ProviewR

OPEN SOURCE PROCESS CONTROL

Getting Started Guide

2009-05-20

Table of Contents

About this guide....4
Download and installation....6
Create a project....7
Configure the Project....9
Configure the Process station....15
- Create a simple Plc program....17
- Build....20
- Start runtime....21
  - Examine the runtime environment....22
- Process Graphics....23
- Analog programming....27
- Alarms and event....31
- More Process Graphics....34
  - Change the color and text of a pushbutton....34
  - Build a graphic component....36
- Write a help text....39

Appendix....44
- Glossary....44
- Edit functions in the Configurator....45
- Edit functions in the Plc Editor....46

About this guide

This guide will take you through the steps of configuring, developing, simulating and running a small ProviewR project, on a single-computer system. The guide does not intend to be comprehensive. For detailed documentation, please consult the Designer's Guide or the GE (graphical editor) Manual. These documents are available at the ProviewR [http://www.proview.se] site.

Download and installation

To configure a ProviewR system you need to install the ProviewR Development package, pwr46, which is available on the Download page at www.proview.se. Download the package for your Linux release and follow the instructions in the Installation Guide to install for the package.

Create a project

- Login as user pwrp. During installation the user `pwrp` with password `"pwrp"` is added to the system.
- Open the ProjectList by doubleclicking on the ProviewR icon on the desktop. The ProjectList will open up in a new window.
- Enter Edit mode by activating Edit/Edit mode (Ctrl+E) in the menu. Once in Edit mode, an object palette will appear to the left in the window.
- Create a project hierarchy. The ProjectList at first only contains the "Bases" hierarchy. To create a hierarchy for the projects, select Hier in the palette, move the cursor to the "Bases" hierarchy and middle click. A new, empty Hier object is inserted below the existing one.

  The hierarchy object is a container for arbitrary objects and it has two attributes: an ObjectName and a Description. To expand the object and expose the attributes, select the object and use the right arrow key. Now, to edit an attribute, select the attribute and use the right arrow key to open the input field.

  We choose the name "Test" for our hierarchy.
- We then add a ProjectReg object to the "Test" hierarchy and name it "GettingStarted". We want to add the ProjectReg as a child and not as a sibling. To do this, middle click directly on the desired parent leaf, which is our "Test" hierarchy in this case. Open the object (select the object and use the arrow right key) and set the name to "GettingStarted". Note that the project and path attributes also are influenced the name change.
- Save the changes (Ctrl+S) and leave Edit mode (Ctrl+E).

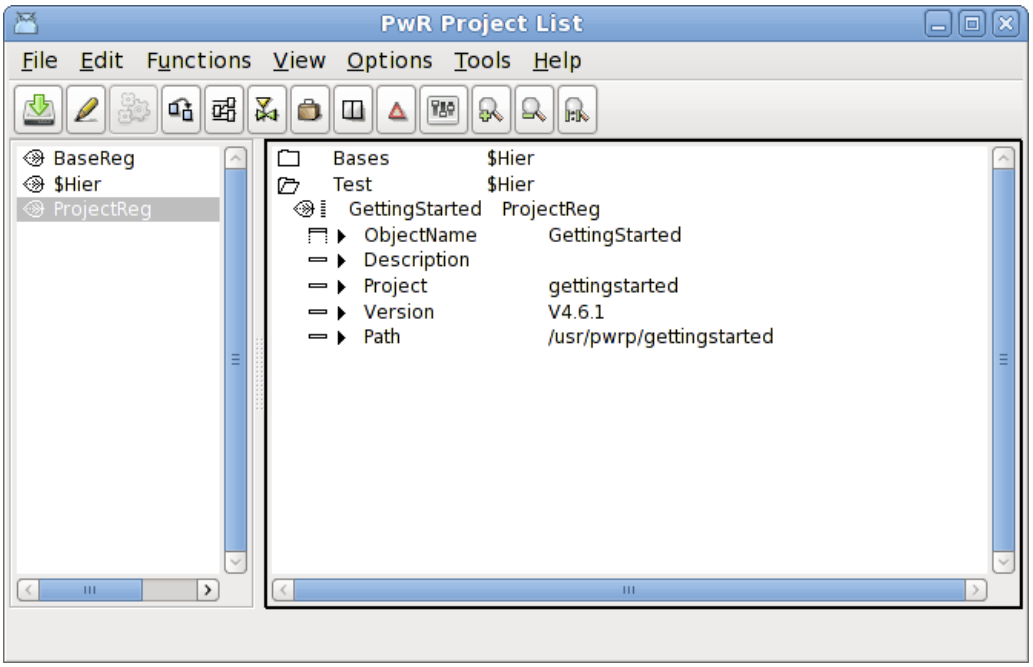


Fig The Project List

Tip

This configurator tool is used to configure most functions of the system, by creating objects and set values to object attributes. See *Edit functions in the Configurator* on how to select/create/delete/move objects and set attribute values. Note that most actions can be activated in different ways: from the menu, from the PopupMenu

(opened by rightclicking on an object), with a single mouseclick or from the keyboard.

Configure the Project

The next step is to attach the project and the Directory Volume by right clicking the ProjectReg object in the ProjectList, and choosing Open Project in the popup menu.

> **Tip**
>
> An alternative way is to open a terminal window and type
>
> ```
> > sdf gettingstarted
> ```
>
> The sdf command sets up environment variables for the project with default values. The Directory Navigator is then started with
>
> ```
> > pwrs
> ```

Starting up the Directory Navigator for the first time in a project will activate a wizard that will configure the nodes and volumes of the project. To create a simple test project you basically just have to press the 'Next' button.

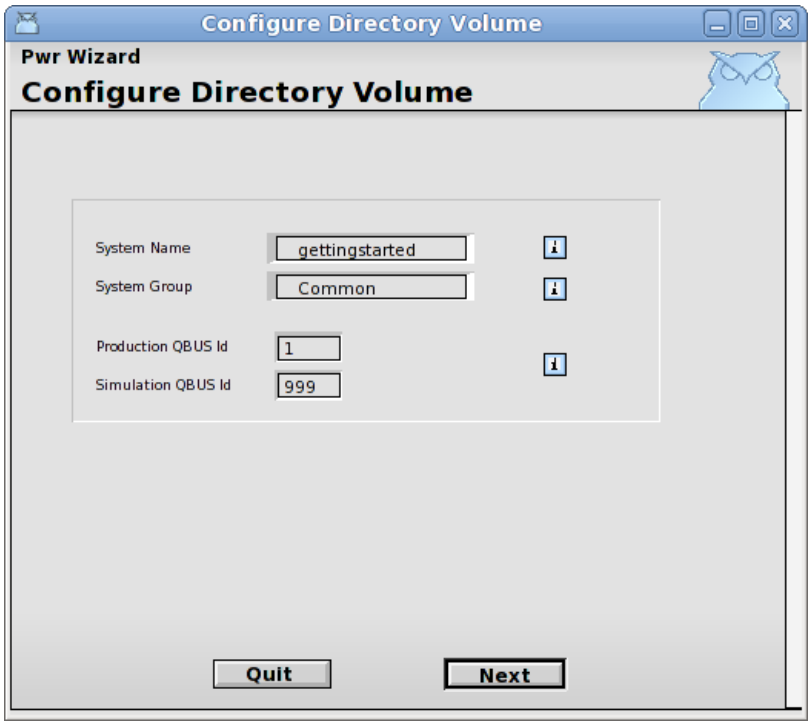


Fig Directory Volume Wizard 1

Press Next.

Comment

The systemgroup defines the users that will have access to the system. Common is a predefined group with the users pwrp and b55. By opening the UserDatabase you can define you can define another systemgroup with users and enter the name of this systemgroup here.

The ProviewR network is divided into buses, and if two nodes should be able to communicate with each other they have to be on the same bus. We will later start the runtime environment on the simulation bus id, which is stated to 999. The environment variable $PWR_BUS_ID controls which bus the runtime is started at, and this is configured in the file /etc/proview.cnf.

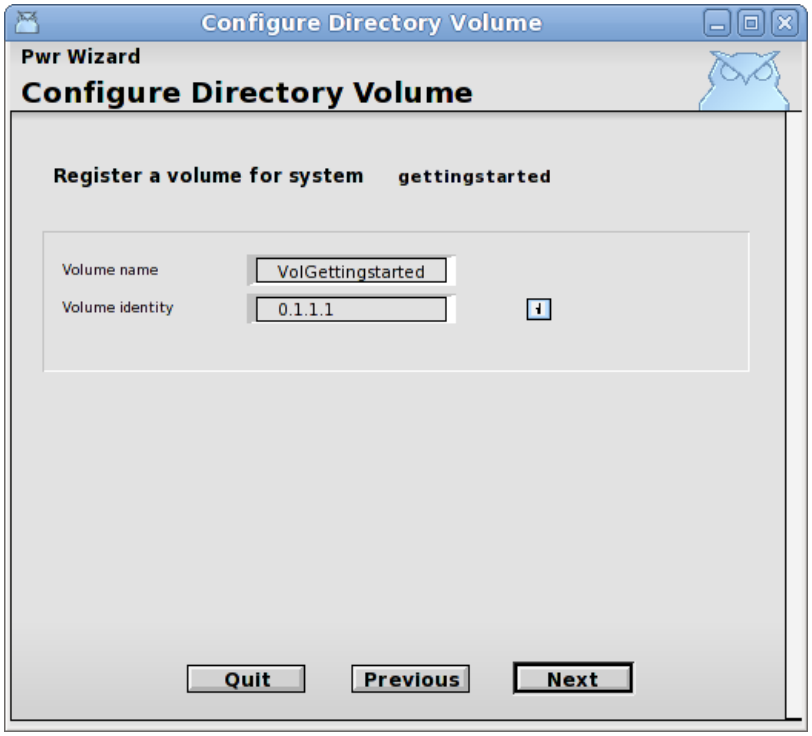


Fig Directory Volume Wizard 2

Press Next

Comment

This will register the volume VolGettingstarted in the GlobalVolumeList. In this volume we will later configure our process station.

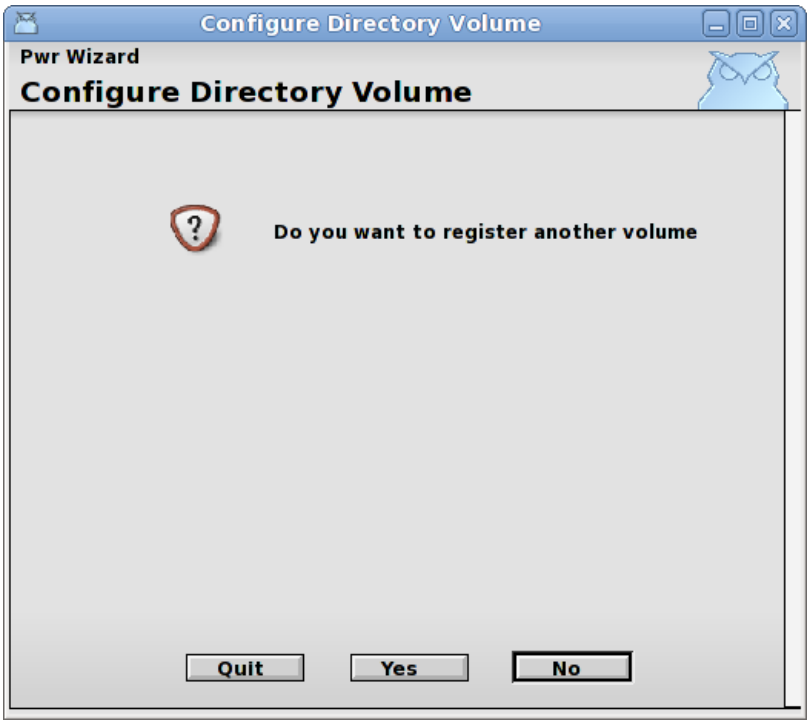


Fig Directory Volume Wizard 3

Press No

Comment

If we have more than one node (process, operator or storage station) in our project we should register one volume for each node. The standard name for a root volume is the nodename with the prefix 'Vol'. For our development and simulation nodes we don't register any volumes. They will load a volume for one of the process, operator or storagestations in the project.

We can also register a classvolume if we need to create specific classes in this project. The standard prefix for classvolumes is 'CVol'.

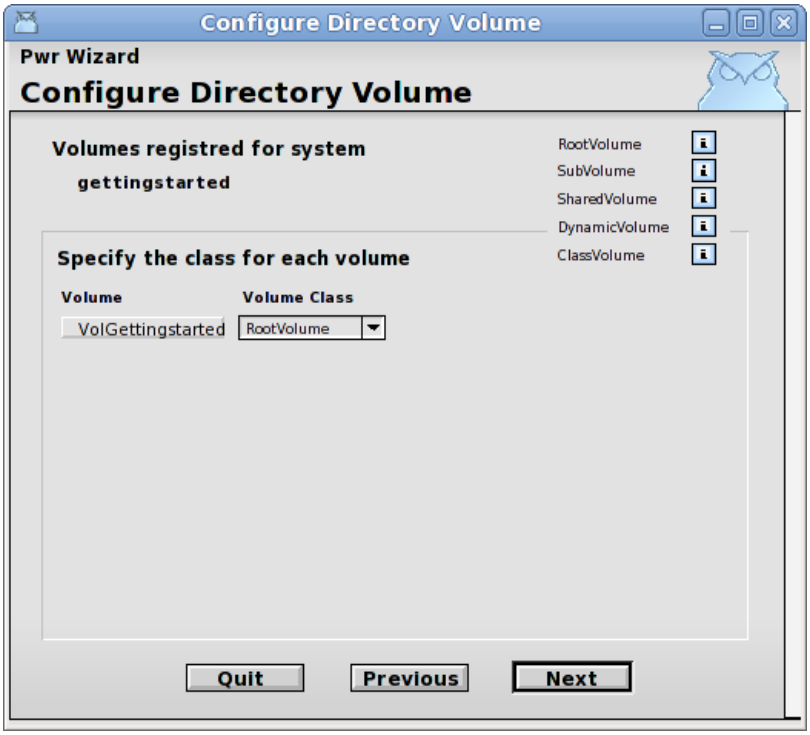


Fig Directory Volume Wizard 4

Press Next

Comment

In this page the type of the registered volumes are specified. In our case we want a root volume which is the default. A root volume is a volume that usually contains all the objects of a node, and is loaded into the node when the runtime environment is started.

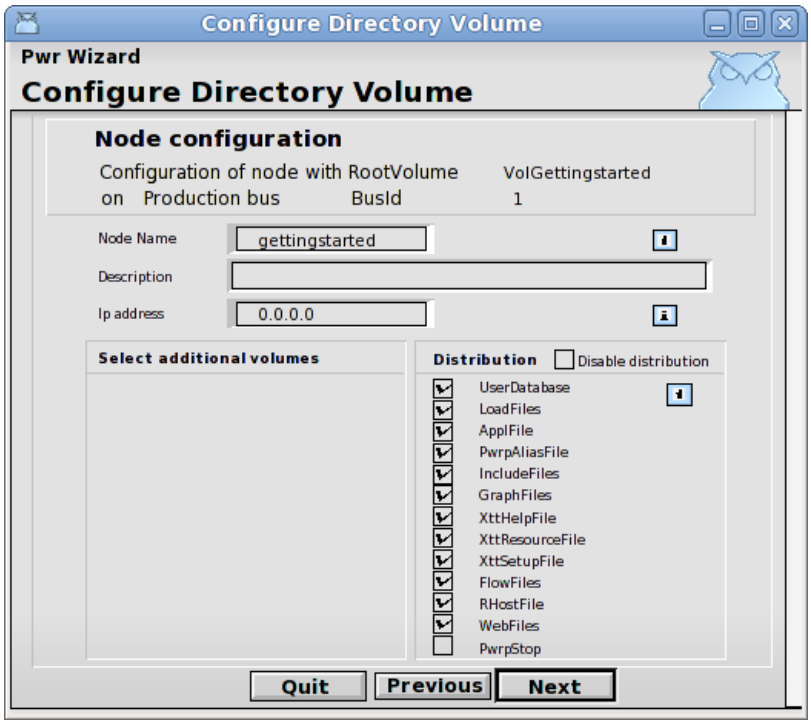


Fig Directory Volume Wizard 5

Comment

This is the configuration of the process station. Normally we would enter a valid IP-address, but in our case, we are never going to start this node, instead we will start the simulation node which is configured on the next page.

Press Next

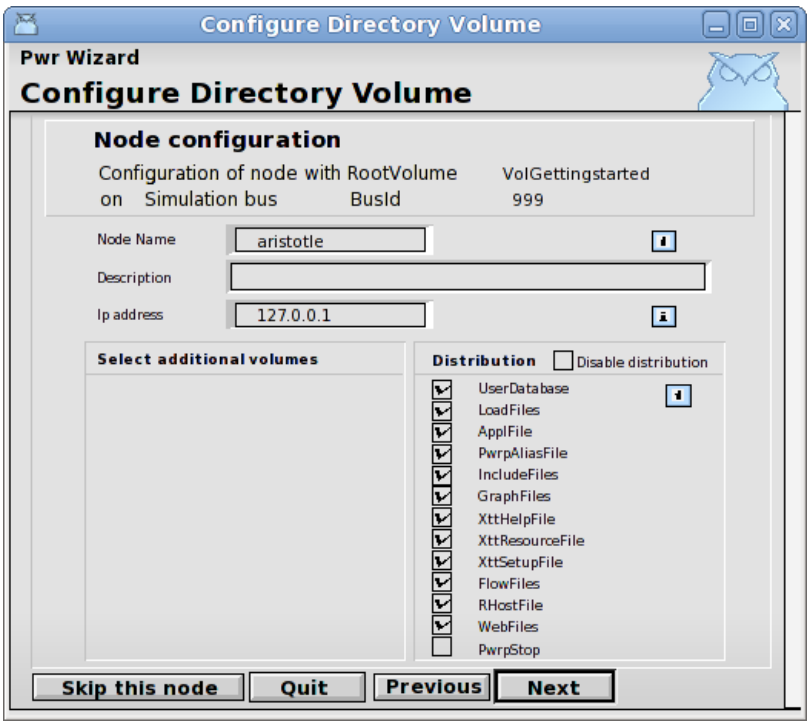


Fig Directory Volume Wizard 6

Comment

This is the configuration of the current node, on which we later will start the runtime environment. Note that the default IP address is the loopback address 127.0.0.1, which will work as long as we don't have any communication with other nodes.

Press Next

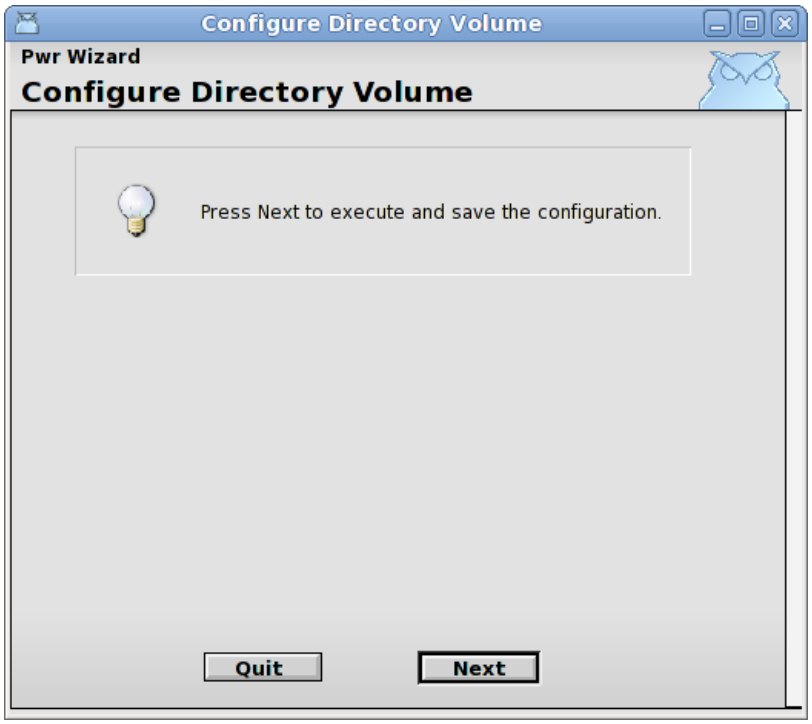


Fig Directory Volume Wizard 7

Press Next

Now the configuration object for volumes and nodes are created and stored in the directoryvolume. Let's take a brief look at the result.

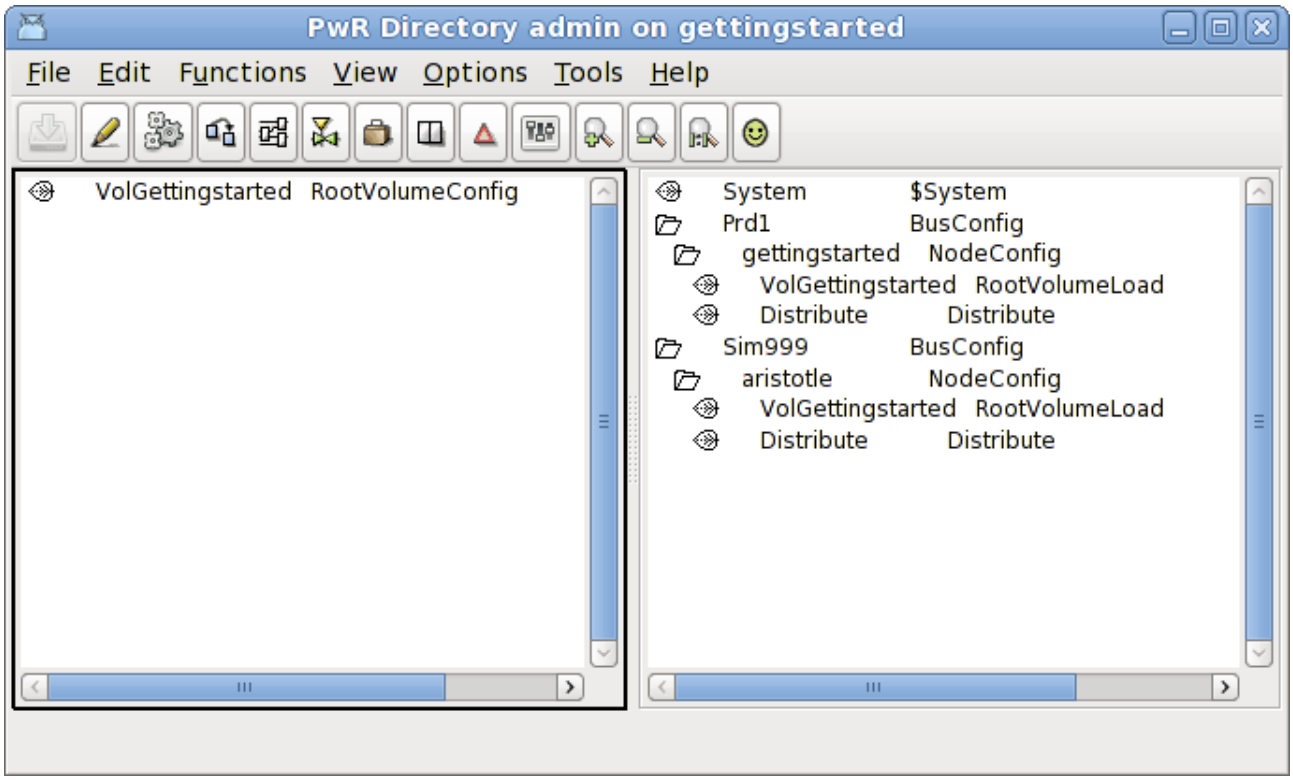


Fig Directory Volume

The view is separated into two windows, the left with the configured volume(s) and the right with the configured nodes. Under the production bus Prd1 the production node is configured and under

the simulation bus Sim999 the current node is configured as a simulation node. Under each node a root volume is stated with a RootVolumLoad object, specifying which volume is loaded at ProviewR runtime startup. There is also a System object containing the systemname and systemgroup. If we later want to add volumes ord nodes to the project, we enter edit mode, and create additional VolumeConfig and NodeConfig objects.

If we return to the wizard, there is still one remaining page.

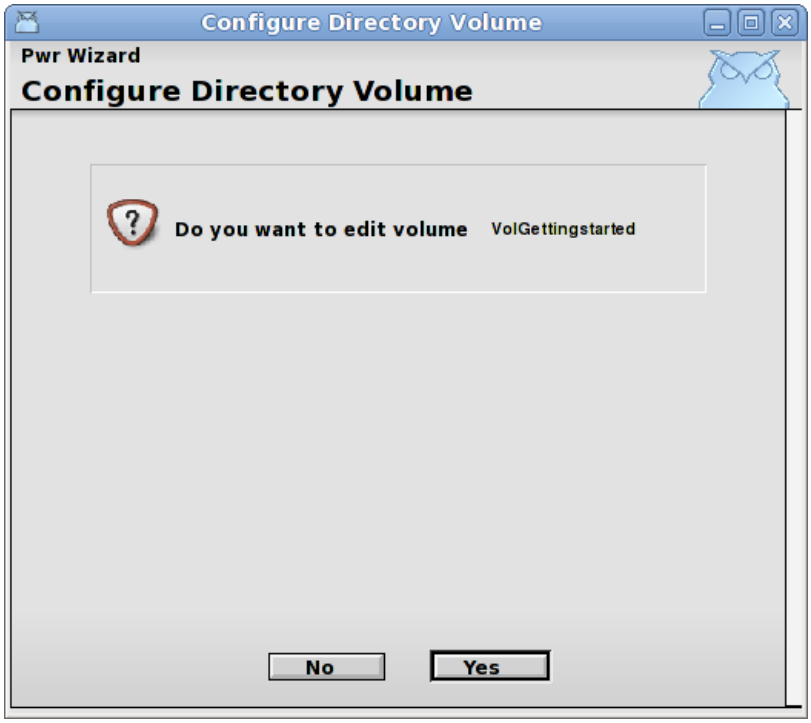


Fig Directory Volume Wizard 8

Press Yes

Comment

We will now open the root volume VolGettingstarted where the configuration of our process station is made.

Tip

This volume can also be opened by rightclicking on the RootVolumeConfig object in the DirectoryVolume, and activating Open Project in the popupmenu.

It can also be opened from a terminal window with the command (do sdf to the project first)

```
> pwrs volgettingstarted
```

Configure the Process station

Opening an empty rootvolume will start the RootVolumeWizard

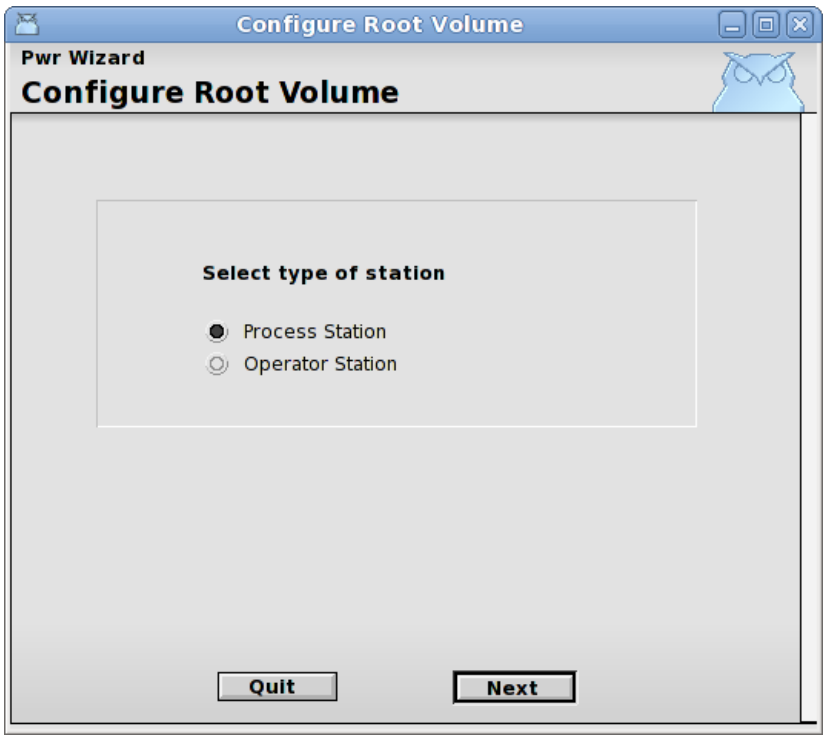


Fig Root Volume Wizard 1

Comment

The process station contains also the functionality of the operator station (and storage station) so we will be able start the operator environment as well.

We will configure a process station, press Next.

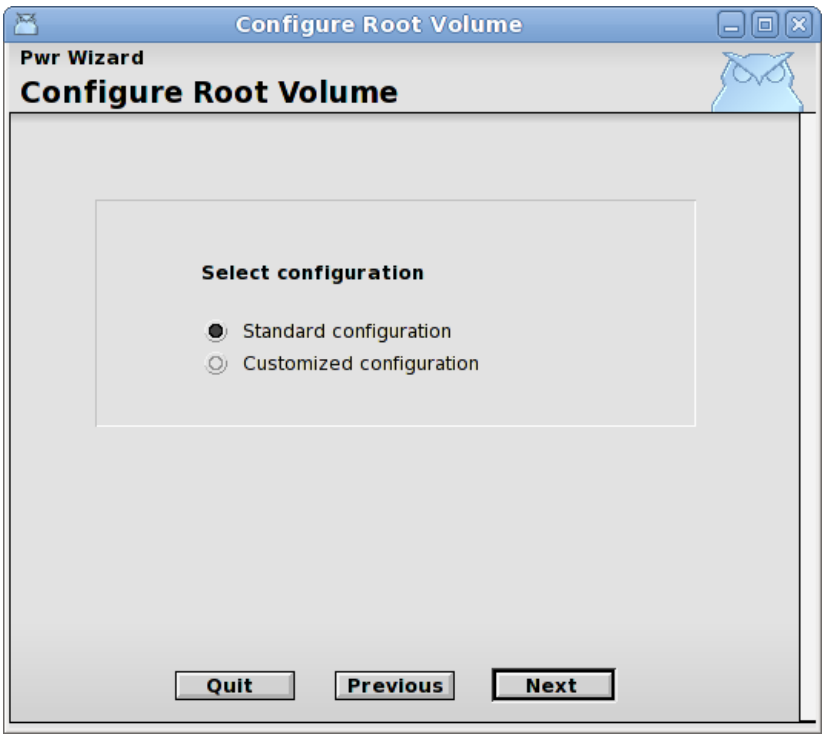


Fig Root Volume Wizard 2

Use the standard configuraton and press Next.

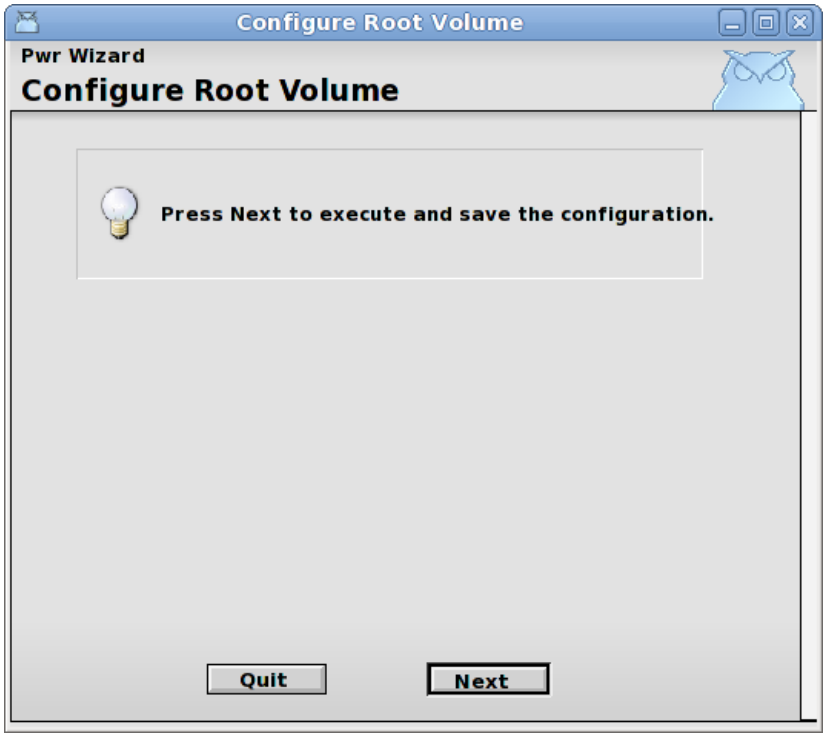


Fig Root Volume Wizard 3

Press Next to save the configuration.

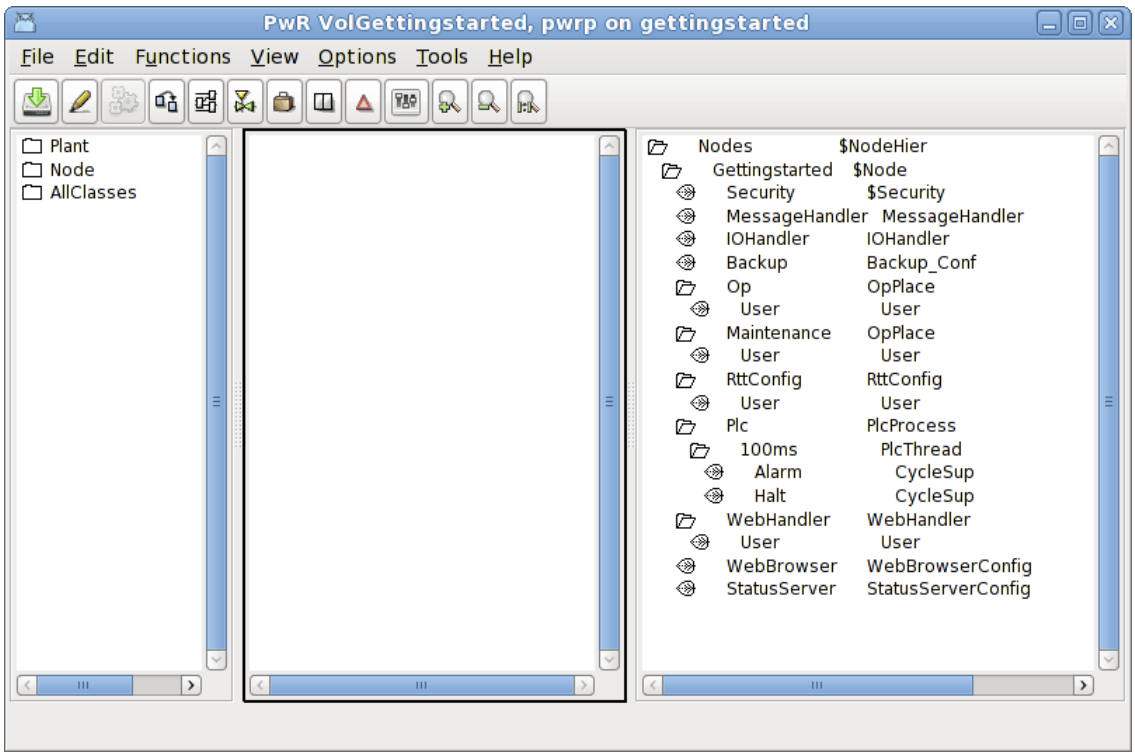


Fig VolGettingstarted initial configuration

Let's have a look at the result in the root volume VolGettingstarted window. The Configurator is separated into two windows, the Plant hierarchy to the left and the Node hierarchy to the right. Note that we already are in edit mode, and the palette is visible to the left.

The node hierarchy contains configuration objects for the hardware and for system processes, and is already configured to some extent, all the objects needed to run our simulation is there.

The planthierachy though, is empty. It will contain objects for the signals, sensors and components

in the plant. We will also add cod form the plc program in the plant hierarchy.

Create a simple Plc program

If you have followed the guide, the Configurator for the rootvolume, VolGettingstarted, is now open, and in edit mode.

> **Tip**
>
> To open the rootvolume again, login as pwrp, doubleclick on the ProviewR icon to open the ProjectList, find the GettingStarted project, rightlclick on it and activate Open Project in the popup menu. This will open the directory volume. Here rightclick one the VolGettingstarted object in the left window, and activate Open Volume in the popup menu. Now the rootvolume will open. Enter edit mode with Ctrl+E.

- Create the hierarchy H1 in the Plant hierarchy: open the Plant map in the palette to the left, and select $PlantHier. Middleclick in the Planthierarchy window (the left) and a $PlantHier object is created. Select the object, and press the Arrow right key to open the object, and set the name to H1 (select ObjectName and use Arrow right key again to open the input field). Close the object with Arrow key left.
- Create four digital value objects (Dv): open the Signal map in the palette and select Dv. Middleclick on the leaf of the H1 object to create a child object.. Continue to create three additional Dv objects and name the Dv objects Dv1, Dv2, Dv3 and Dv4.

  > **Tip** Arrow up when the input field is open recalls the last inserted values.
- Create a PlcPgm object: select PlcPgm in the palette. Create the object as child to H1 and name it Plc. Set also the description to 'A simple program'. Note that the description is displayed en the configurator.

  > **Tip** Most objects have a description field. It's usually a good idea fill in this.
- Save (Ctrl+S) and leave edit mode (Ctrl+E)

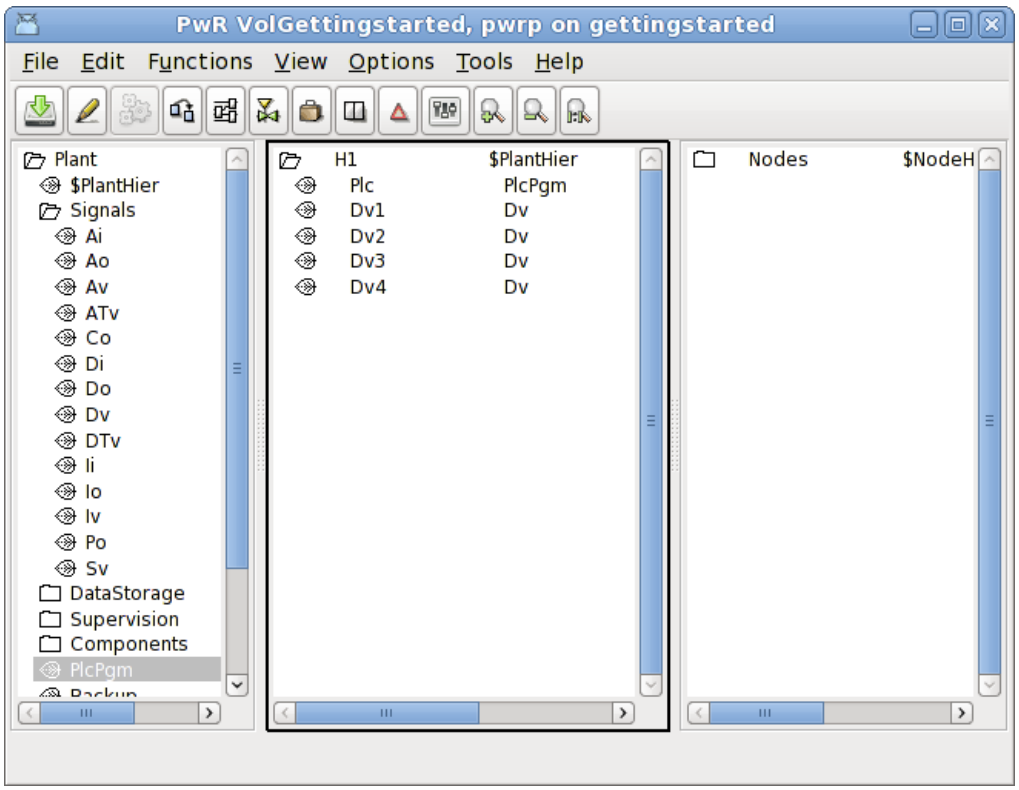
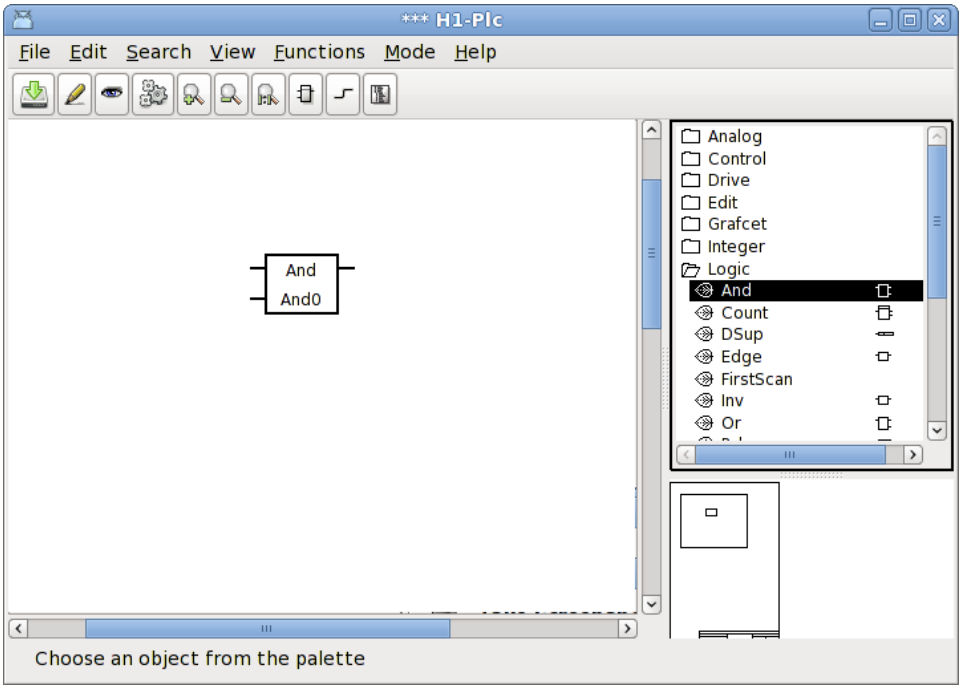


Fig Dv and PlcPgm configured

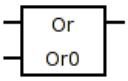
Now it's time to open the plc editor. Rightlclick on the Plc object and activate Open Program. Note that you have to leave edit mode to open the plc editor.

The Plc editor contains a working area with a palette and navigation window to the right. We will create an And gate, an Or gate and a Wait (delay) function object, and we will use the Dv's we have created as inputs and output in the logical schema.

- Create an And gate by opening the Logic map in the palette and select And. Move the cursor to the a position in the working area where you want to place the And gate and click there. The mousebutton should not be pressed when moving the cursor.

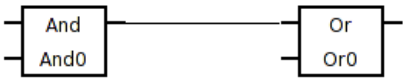


- Create an Or gate by selecting Or and move the cursor to a position in the working area.

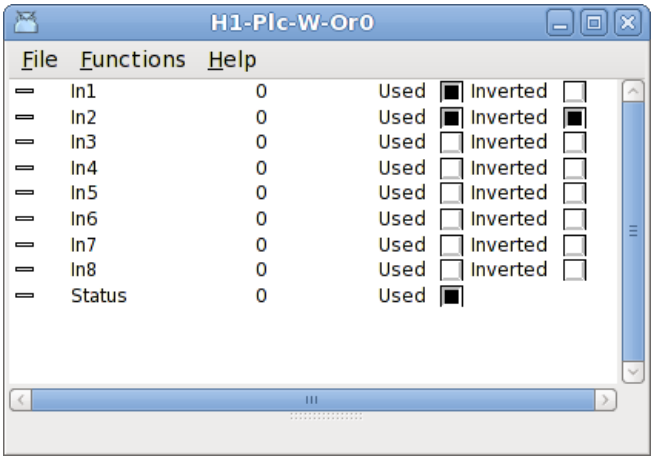


- Connect the output of the And to the first input of the Or by dragging with the middle button from the And output pin to the Or input pin.

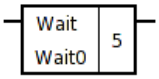
  Tip It can be a little hard to hit a pin. Note that also the area inside the pin is sensitive.



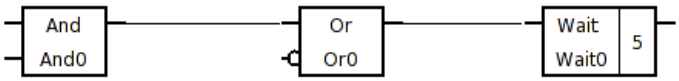
- We will now invert the second input of the Or gate. Rightlclick on the Or gate and activate ObjectEditor in the popup menu. From the object editor we can add new inputs and invert them. Click on the Invert checkbox for In2 to invert the second input. Then close the object editor.



- Create a Wait object by selecting Wait in the palette and place the object in the working area. Open the ObjectEditor (from the popup menu) and enter the delay time in TimerTime to 5 (select TimerTime and press the Arrow right key to open the input field). Close the object editor. Note that the delay time is displayed in the Wait function object.

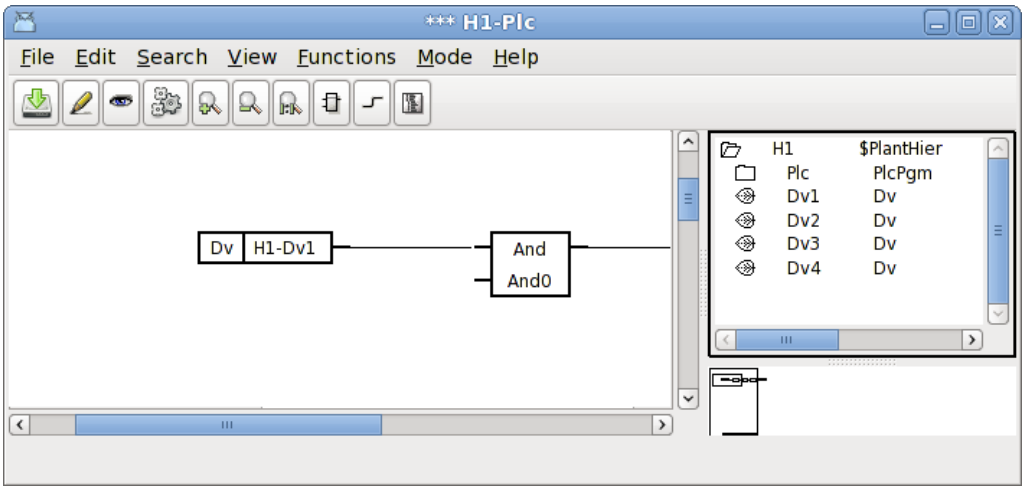


- Connect the output for the Or gate to the input of the Wait object.



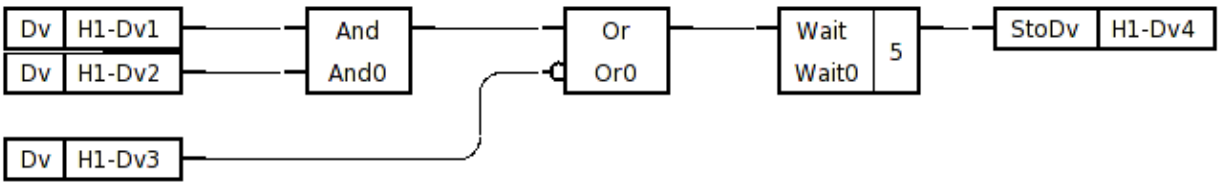
- Next step is to connect the Dv's as input and output to the logical schema we have created. Create a GetDv object, which fetches the value of a Dv object, by dragging with the middle button from the input of the and object (as you created a connection before), and releasing the button in some empty space in the working area. This will create a GetD object that can be connected to the Dv.

Activate View/Palette/Plant in the menu to replace the palette with the plant hierarchy. Select Dv1 and activate Connect in the popup menu for the GetD object.



- Continue to connect the other inputs to Dv2 and Dv3, and the output to Dv4.

Tip You can also use Ctrl+LeftDoubleClick to connect to the plant hierarchy.



Our program is now complete. Save and close the plc editor.

Build

Return to the VolGettingstarted configurator, and activate Functions/Build Node in the menu.

Select the current node, which is our simulation node (not the gettingstarted node which is our non existing process station).

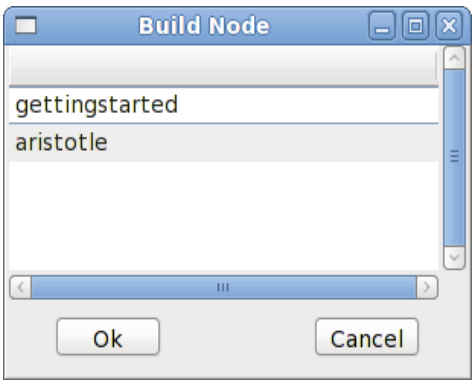


Fig Build node selection

Start runtime

It now time to start the runtime environment. Activate Tools/Runtime Monitor to open the runtime monitor. Click on the 'Start Runtime' button and wait for the 'Down' label to switch to 'Running'.

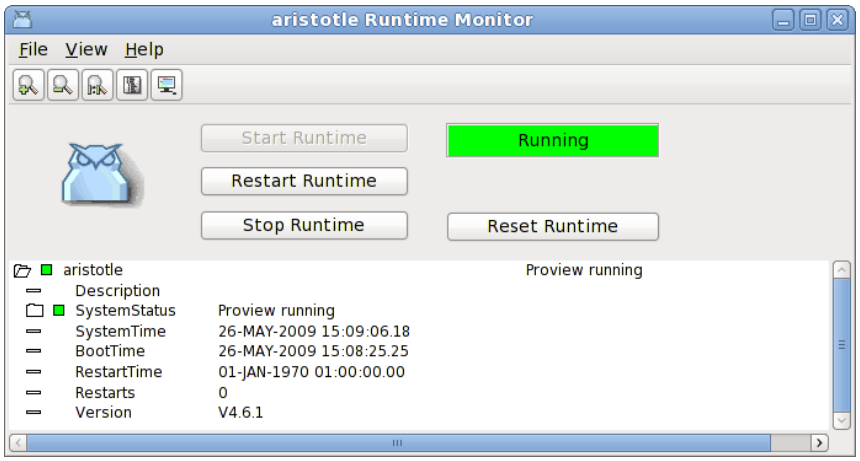


Fig Runtime Monitor

Tip

The runtime environment can also be started from a terminal window with the command (don't forget to do sdf to the project first)

```
> rt_ini &
```

and stopped with the command

```
> . pwr_stop.sh
```

If you have configured another Bus identity than the default (999) you have to set this to the environment variable PWR_BUS_ID

```
> export PWR_BUS_ID=998
```

This can be inserted in the login script $pwrp_login/login.sh.

You can also change the default busid in /etc/proview.cnf.

Examine the runtime environment

The runtime environment is viewed by the runtime navigator (Xtt). It is started from File/Runtime Navigator in the runtime monitor menu.

Tip

You can also start it from a terminal window with

```
> rt_xtt
```

Under the Database map you will find the object tree with the H1 hierarchy and the PlcPgm we created.

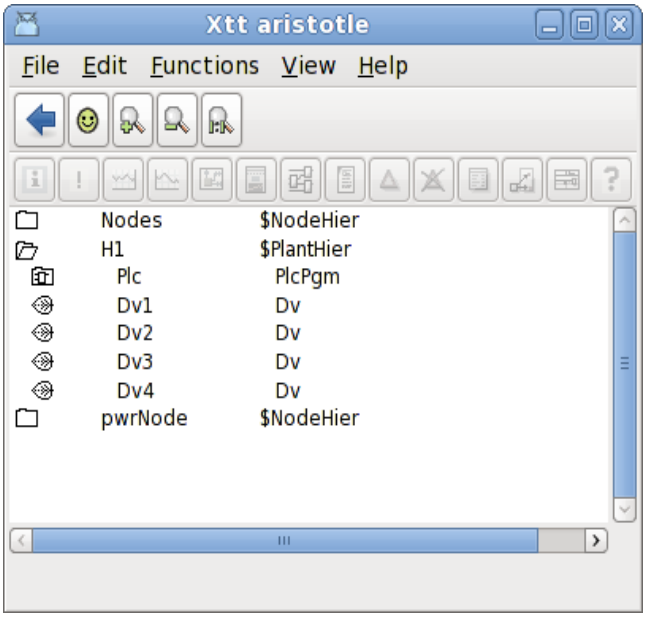


Fig Runtime Navigator Xtt

Rightclick on the Plc object and activate Open Plc in the popup menu to open the program. Now the logical schema is viewed in trace mode, with objects with high status marked with red.

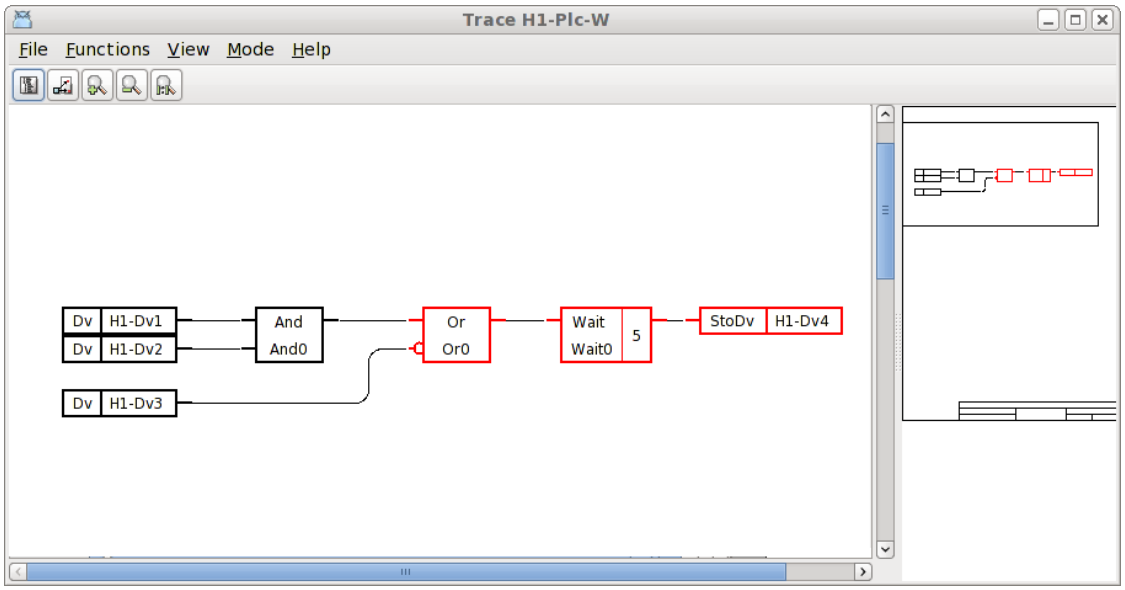
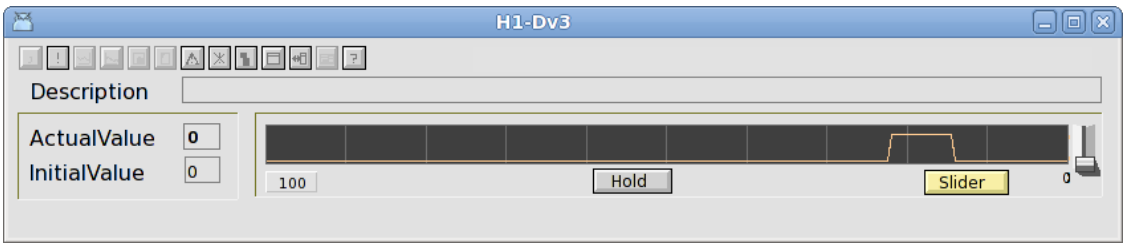


Fig Trace for the plc code

To change one of the input values, for example H1-Dv3, rightclick on the object and activate Object Graph in the popup menu.

Fig Object graph for H1-Dv3



Click on the Slider button to enable the slider to the right and change the value to high with the slider. Notice how the highlight of the Or gate and following objects are removed. Change the value to low and check that the wait object gets high after 5 seconds. Check the program by setting and resetting the other Dv values as well.

Tip

If you change the mode to Simulate (Mode/Simulate in the plc trace menu) you can toggle the Dv values by clicking with Ctrl+Shift+LeftDoubleClick on the function objects.

Process Graphics

We will now draw a graph with some pushbuttons and indicators, and connect them to our Dv objects.

- Enter edit mode in the Configurator for the root volume VolGettingstarted (Ctrl+E) and create a XttGraph object in the node hierarchy under the OpPlace object Nodes-Gettingstarted-Op. You will find XttGraph in the palette under Node/Operator. Select XttGraph in the palette and middleclick on the leaf of the Op object to create it as a child to this object. Open the new XttGraph object (select the object and press the arrow right key) and set the name to TestGraph.

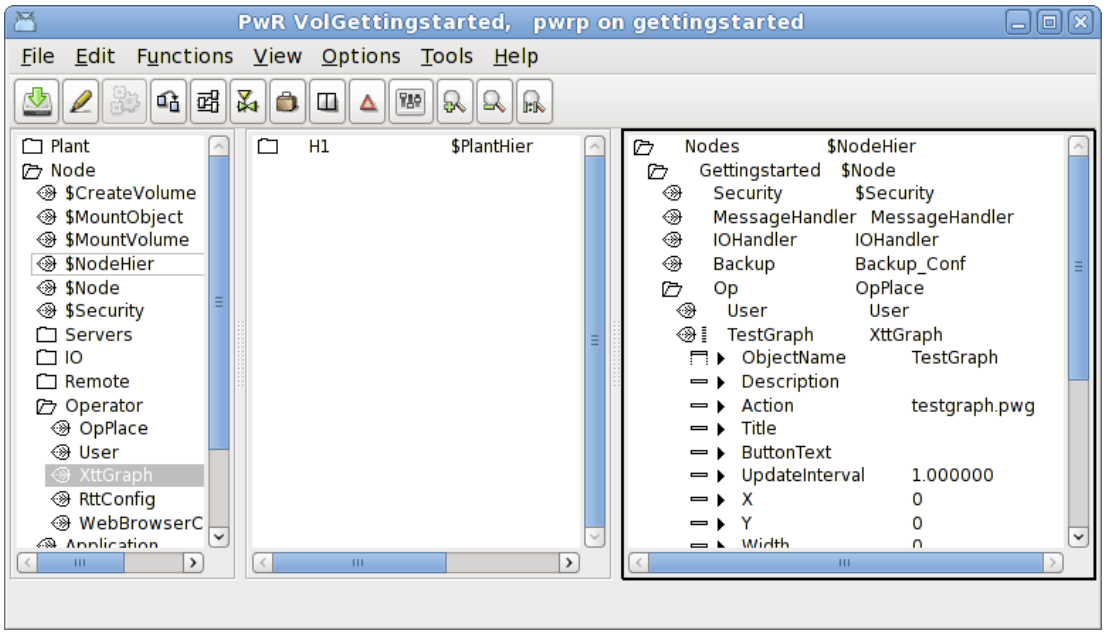
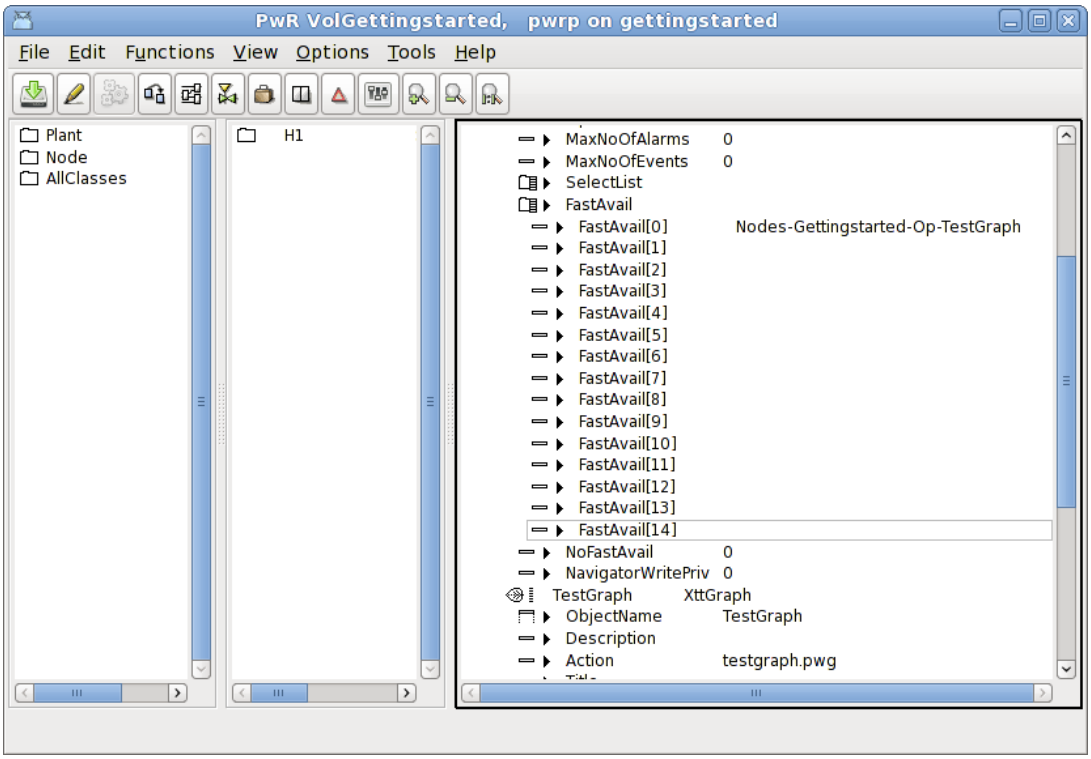


Fig XttGraph object

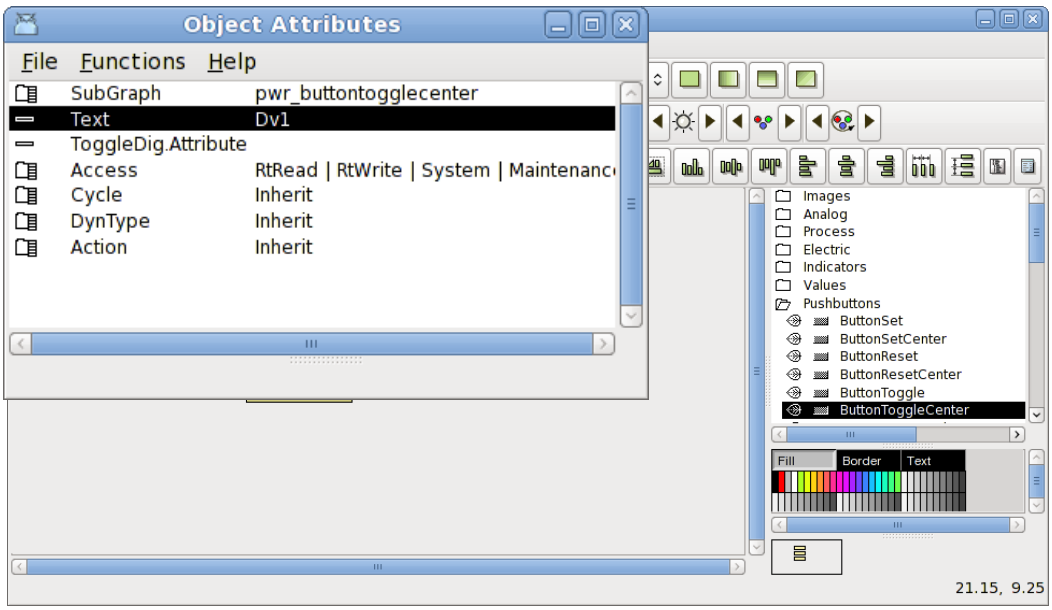
Tip

Note that the Action attributes is set to testgraph.pwg. This is the name of the graph file in the catalog $pwrp_pop.

- To be able to open the graph from the operator window, a link to the XttGraph object should be inserted into the FastAvail array of the Op object. The easiest way to do this is to select the TestGraph object, then right-click on the Op object and activate ConnectFastAvail > Row1 > Column1.

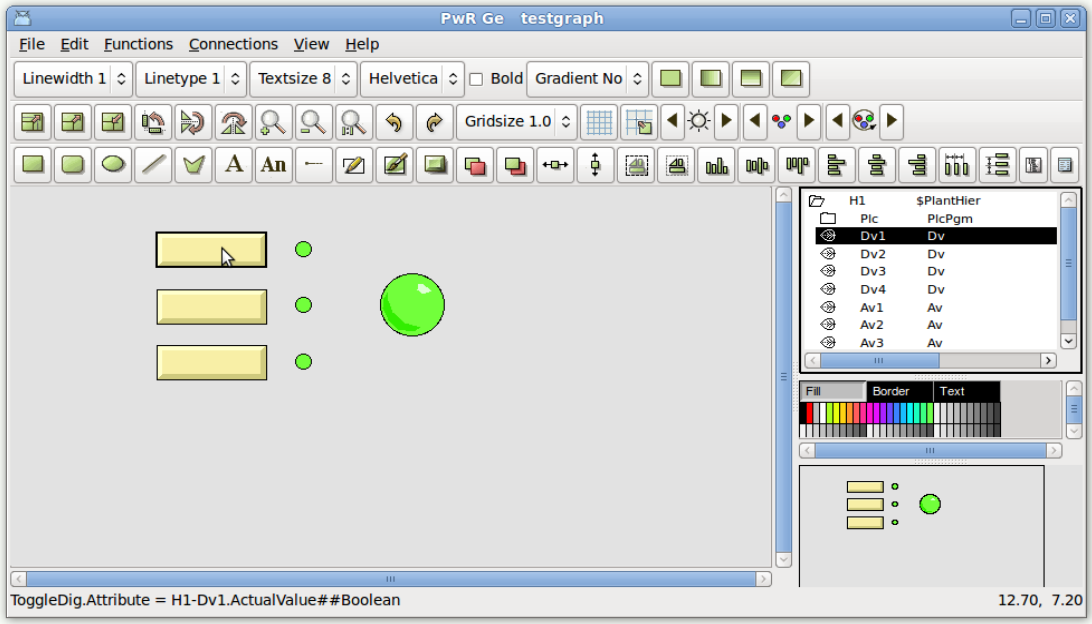


- Save (Ctrl+S) and leave edit mode (Ctrl+E).
- Rightclick on the TestGraph object and activate Open Ge in the popup menu. This will open the graphical editor. The graphical editor contains a working area and a palette and navigation window to the right.
- Create three pushbuttons to toggle the values of Dv1, Dv2 and Dv3. Open the Pushbutton map in the palette and select ButtonToggleCenter. Middleclick in the working area and create three pushbuttons. Doubleclick on a pushbutton to open the attribute editor for the push button. Set the Text attribute to Dv1, Dv2 and Dv3 for the buttons.



Note the ToggleDig.Attribute attribute, this is where we will insert the name of the Dv later.

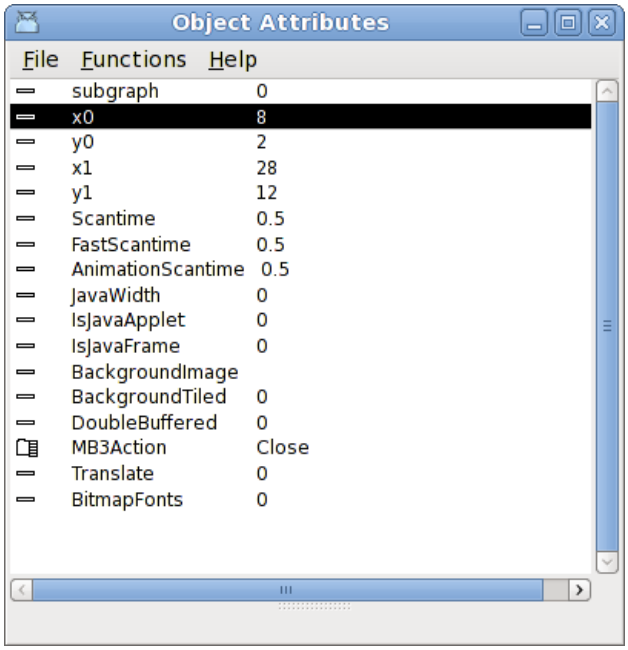
- Create also three indicators to show the values of Dv1, Dv2 and Dv3. Open the Indicator map in the palette and select IndRound. Create three indicators beside the pushbuttons by middleclicking in the working area.
- Create a large indicator for Dv4 by selecting IndRoundLarge in the palette and middleclicking in the working area.
- Now its time to connect the pushbuttons and indicators the the Dv objects. Activate View/View Plant i the menu to replace the palette with the plant hierarchy. Open the H1 map and select Dv1. Select the Dv1 pushbutton also and activate Function/Connect (Ctrl+Q) in the menu. Connect also the small indicator to Dv1, and the corresponding pushbuttons and indicators to Dv2 and Dv3. Finally connect the large indicator to Dv4.



Tip

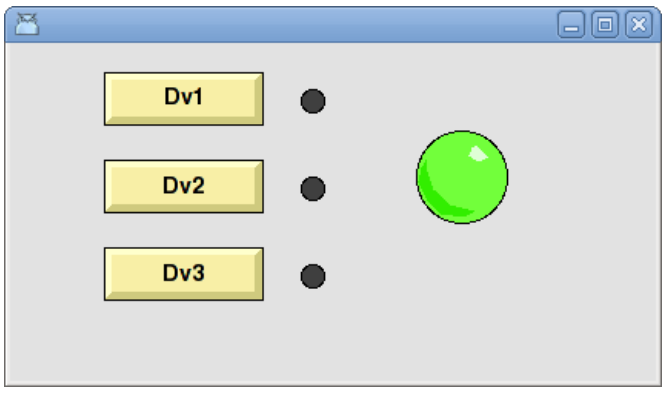
You can also connect by selecting the Dv in the planthierachy and click with Ctrl+LeftDoubleClick on the pushbutton or indicator.

- We should also set the borders of the graph. Activate File/Graph Attributes in the menu to open the attributes for the graph. Measure what you want to be the upper left corner or the graph by placing the cursor there and read the coordinates in the lower right corner of the editor. Insert the coordinates in x0 and y0. Measure the lower right corner and insert these coordinates in x1 and y1.



- Save the graph and close Ge.
- Build the node from the VolGettingstarted Configurator (Functions/Build Node in the menu).
- Open the runtime monitor. Stop runtime and start it again.
- Activate File/Start Operator Environment in the runtime monitor menu.
- Select Nodes-Gettingstarted-Op operator place. The operator window is now opened with the TestGraph button to the right.

- Click on the TestGraph button to open the graph.



- Click on the pushbuttons and check that the indicators are responding.

Tip

Another way to open a graph is to start Xtt from a terminal window

```
> rt_xtt
```

Open the command line in xtt with Ctrl+B and enter the command

```
xtt> open graph testgraph
```

Tip

A XttGraph object is not necessary to open a graph. Open Ge from the Configurator menu, edit and save the graph. Then copy the graph file from the catalog $pwrp_pop to $pwrp_exe

```
> cp $pwrp_pop/testgraph.pwg $pwrp_exe
```

Start xtt and open the graph as described in the tip above.

Finally stop the runtime by pressing the 'Stop Runtime' button in the Runtime Monitor.

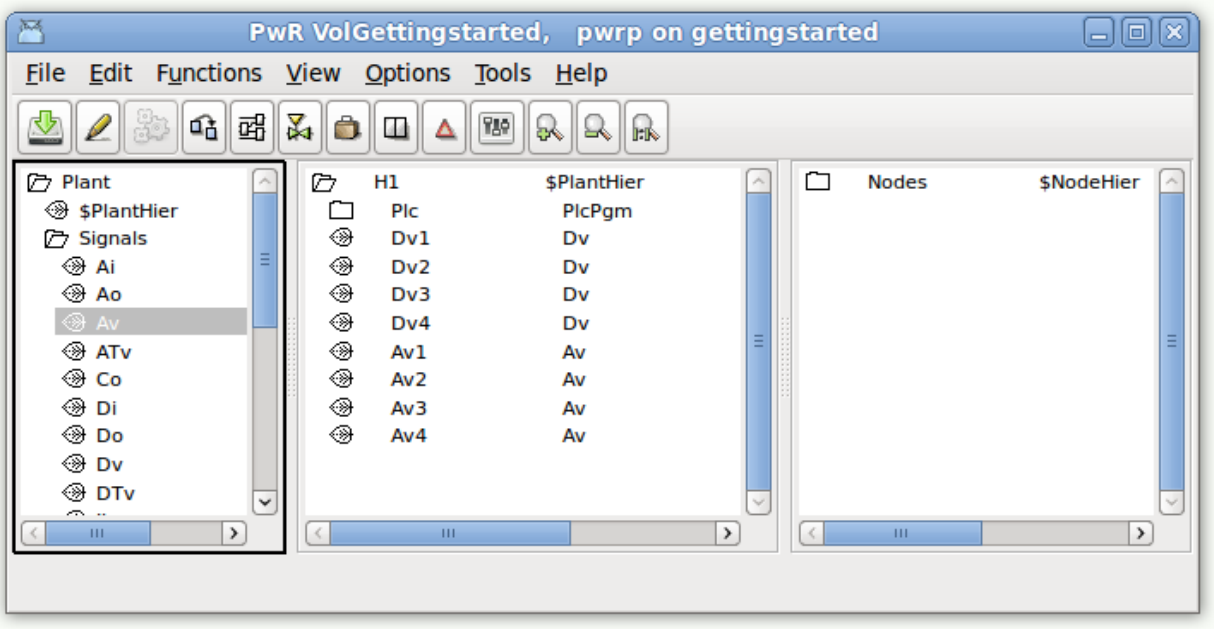
Tip

If runtime doesn't stop, use the 'Reset Runtime' button which is a harder way to stop the runtime. Also use the Reset Runtime after a failed start attempt before trying again.

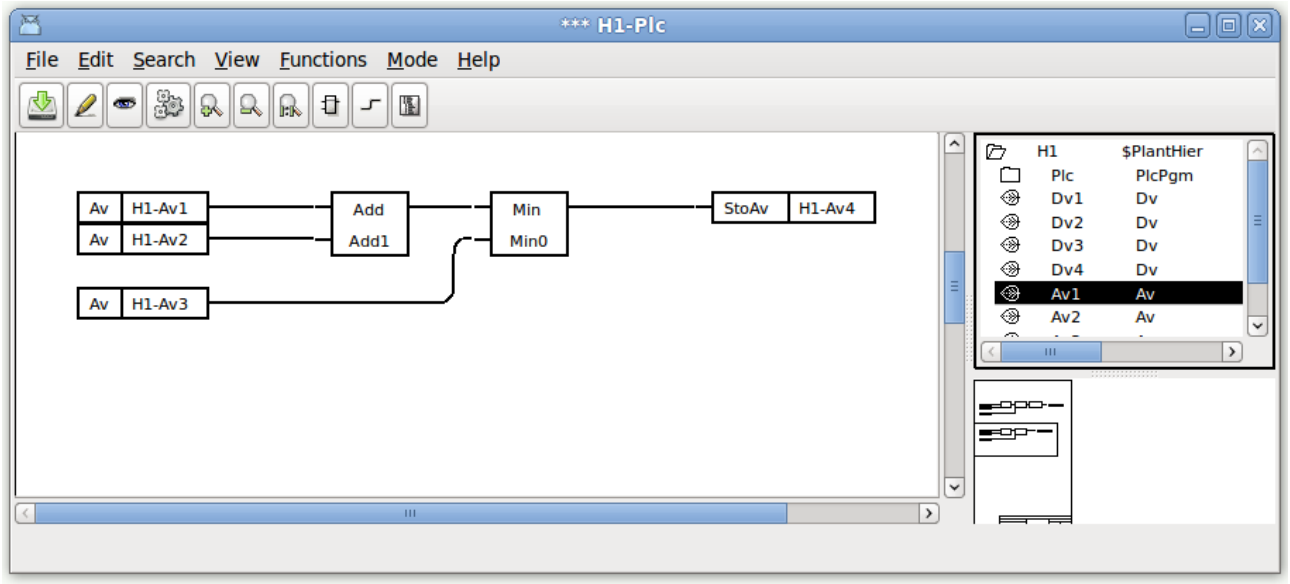
Analog programming

In this section we will add some analog objects to our plc and graph.

Open the configurator for the rootvolume VolGettingStarted again, and enter edit mode (Ctrl+E). Select an analog value object, Av, in the palette (Plant/Signals/Av) and create four Av object in the map H1, and name them Av1, Av2, Av3 and Av4.



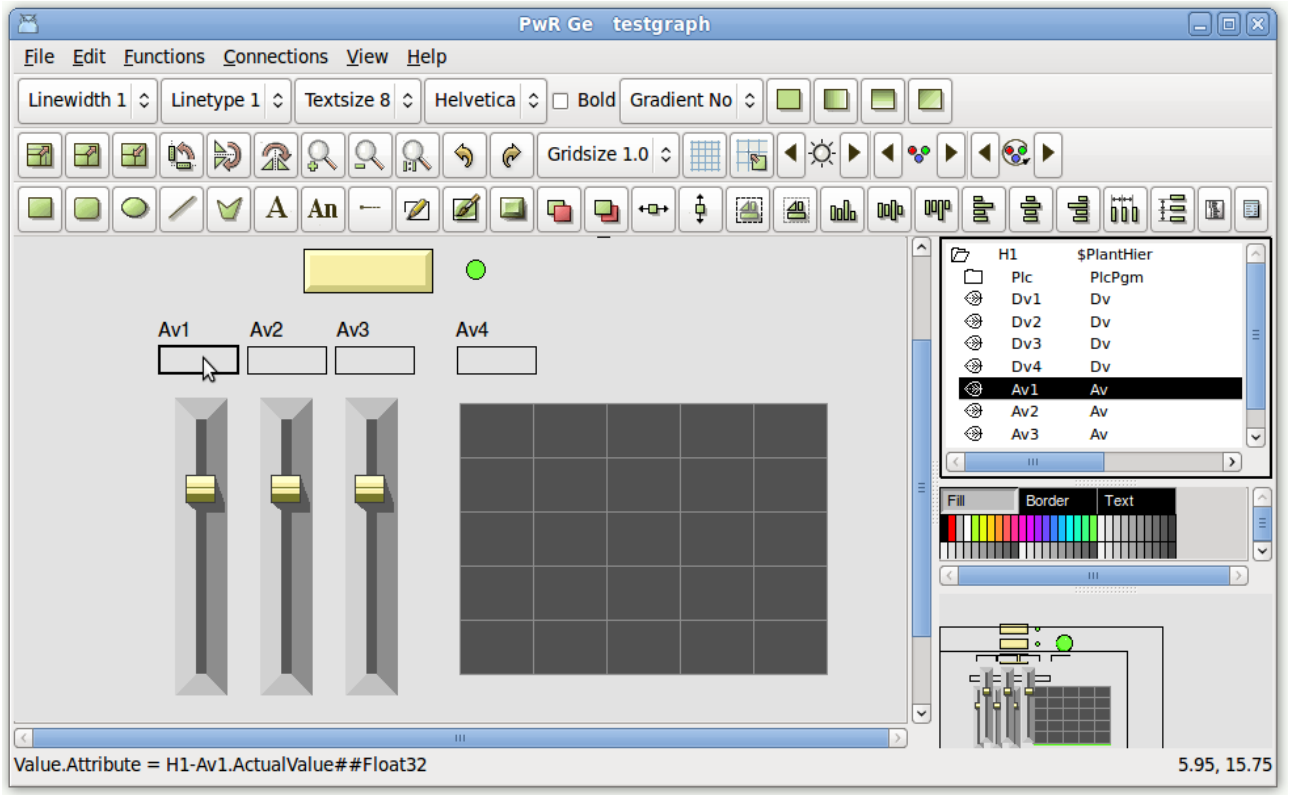
Leave edit mode and open the plcprogram H1-Plc, and enter edit mode (Ctrl+E). Open the map Analog in the palette and create an Add object and a Min object. Connect the output of the Add to the first input of the Min object. Connect the Av objects Av1, Av2 and Av3 to the inputs and Av4 to the output in accordance with the figure below. The connecting to the Av object is done the same way as we did previously with the Dv objects.



Save and close the plc editor. Open Ge from the XttGraph object (Nodes-GettingStarted-Op-TestGraph).

Create three sliders for the three input Av objects, Av1, Av2 and Av3. Open the map Sliders in the palette and select SliderBackground1. The sliders are divided in a background and a slider part. Create three SliderBackground1 objects and then place three Slider1 objects on them. The output Av object, Av4 we will show as a trend. Create a trend object from Analog/Trend in the palette. Resize the trend object by selecting it, click on scale, the upper left pushbutton in the tool panel, and drag with the left mousebutton on the trend object. Create also four value fields from Value/ValueMedium to view the values with figures. Add a text for each value field by clicking the text button marked with an 'A' and leftclick on the spot you want to place the text. Enter the text in the input field. Set the text size with the TextSize option menu.

Connect the graphic object by viewing the plant hierarchy (Ctrl+P). Select Av1 and Ctrl+left doubleclick on the slider to the left. You have to click on the slider object, not the slider background. Connect also the value field. In the value field, the format has to be set. Open the object by doubleclicking on it, and set the Value.Format to '%5.2f'. This is c syntax and means that it is a float value, that will be shown with five characters and two decimals. Continue to connect the second slider and input field to Av2, and the third to Av3. Finally connect the fourth value field and the trend object to Av4. Don't forget to set the format to all the value fields.



Open the graph attributes from File/Graph Attributes in the menu and adjust the graph size in x0,y0 and x1,y1.

Save, exit Ge and build the node from the configurator.

Restart the runtime environment from the Runtime Monitor, or from a terminal window.

Open the Runtime Navigator and open PlcTrace for our plc program (rightclick on the PlcPgm object in the navigator, and activate *Open Plc*).

You are able to see the value of the analog objects by pressing the middle button on an analog output pin and releasing it in an empty space. An analyze node connected to the output pin is now created, viewing the value of the output. We create analyze node for the three Av objects to the left, and for the Add and Min object. To begin with, all the values are zero. To change the values of the Av objects, we open the object graph for them, by rightclicking on them and activating *Open Graph*. To change the value, we can either insert a value in the *ActualValue* field, or we can press the *Slider* button and use the slider to the right.

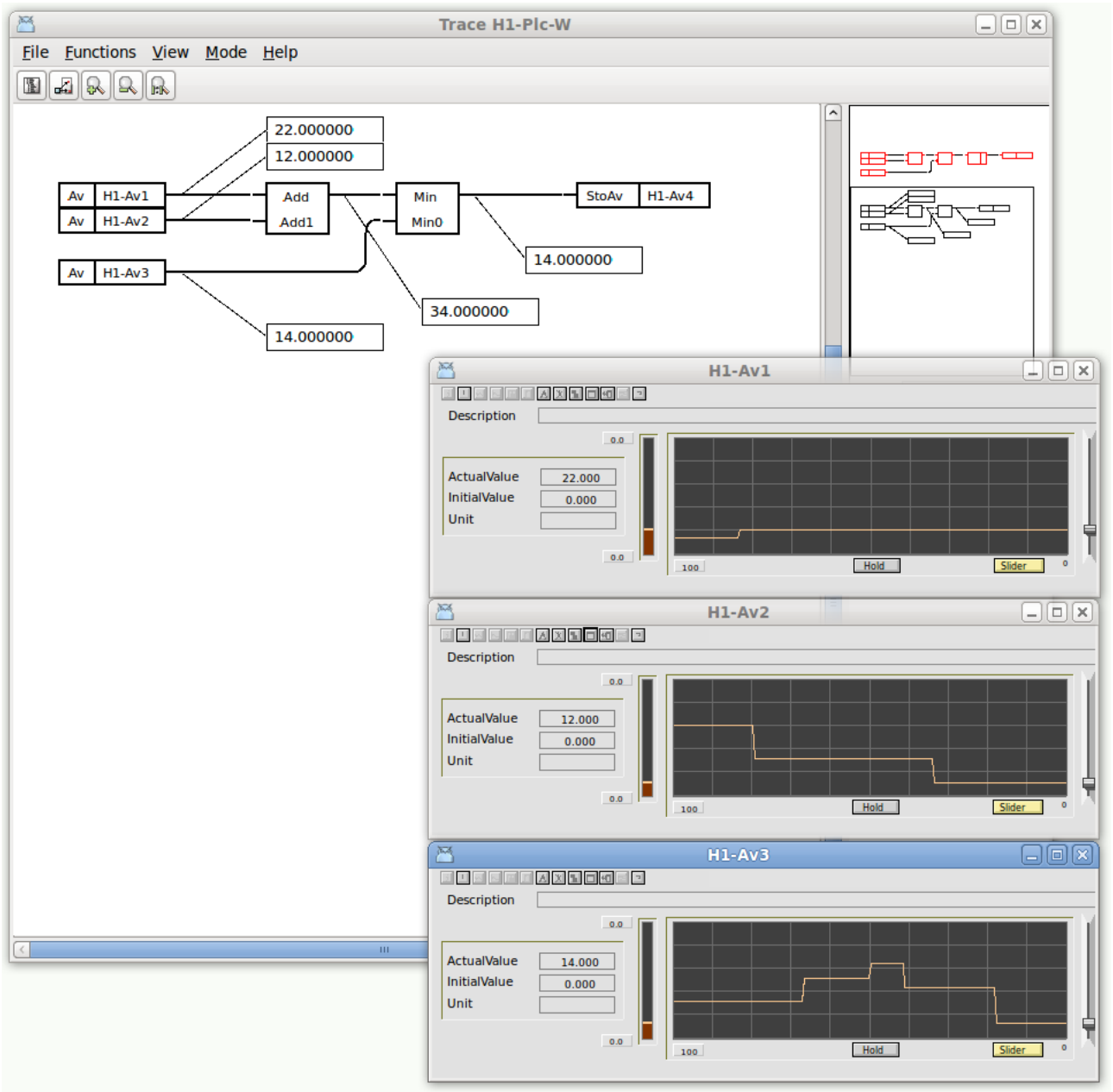


Fig PlcTrace

We also want to know how our graph works, so we open the graph (for example by finding the XttGraph object Nodes-Gettingstarted-Op-TestGraph object, and activating *Graph* in the popup menu). The Av1, Av2 and Av3 values can be changed with the sliders, and the value of Av4 is displayed in the curve.

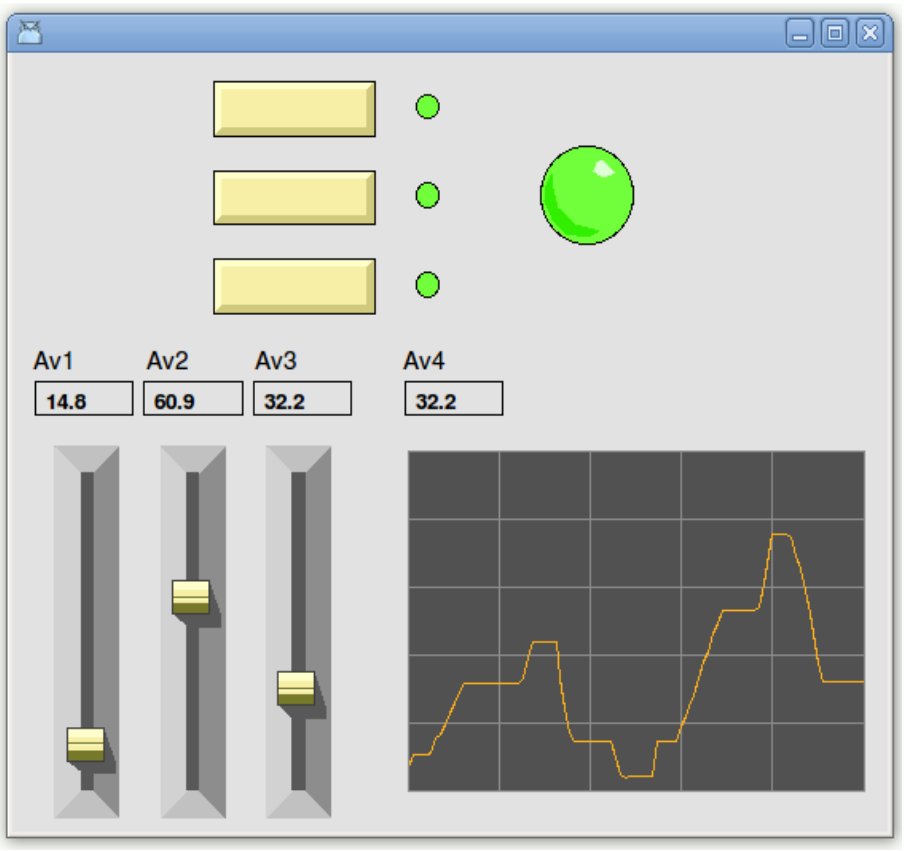


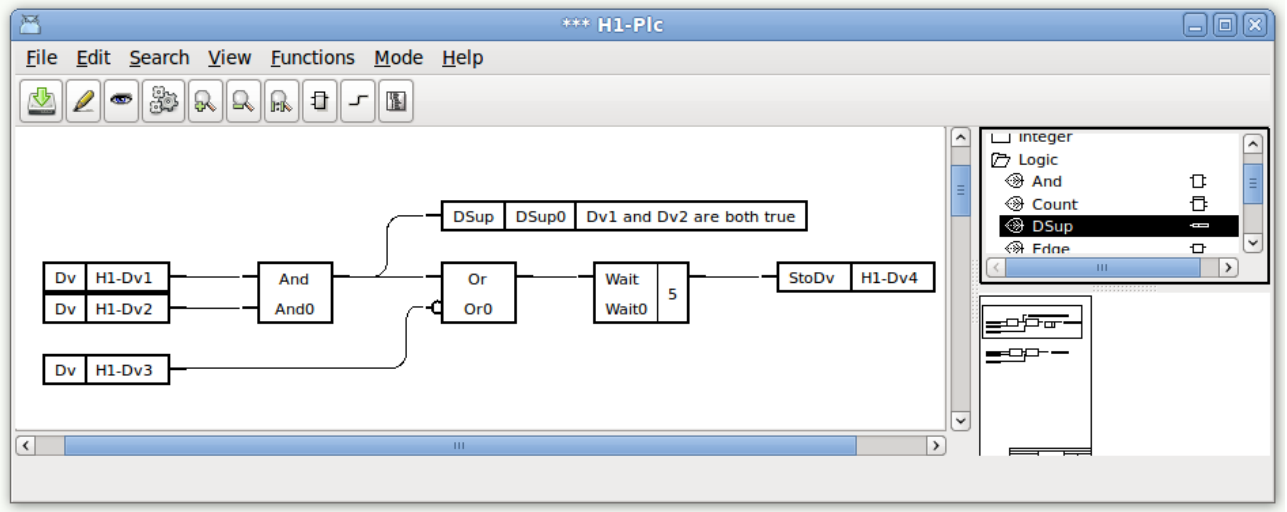
Fig Process Graph with analog values

Alarms and event

Alarms and events are sent with supervision objects. There are two types of supervision objects, DSup that supervises a digital signal, and ASup that supervises an analog signal. They can be configured in the plant hierarchy or in a plc program. In this example we will add a DSup and an ASup to our plc program.

Open the plc editor for our PlcPgm, H1- Plc, and enter edit mode.

Open the Logic map in the palette and create a DSup object. Connect the input of the DSup to the output of the And object.



Open the object editor for the DSup, and insert the alarmtext into the DetectText attribute.

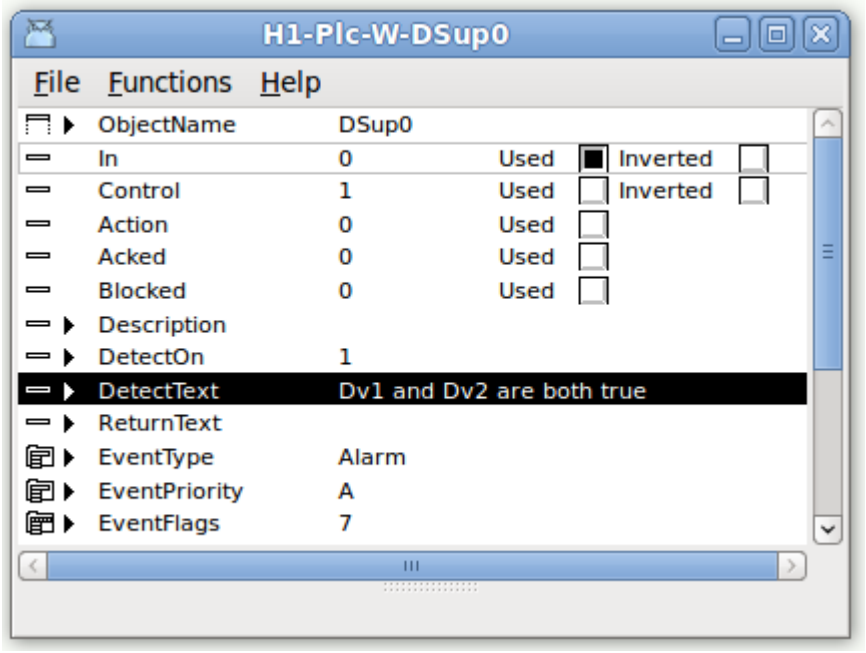
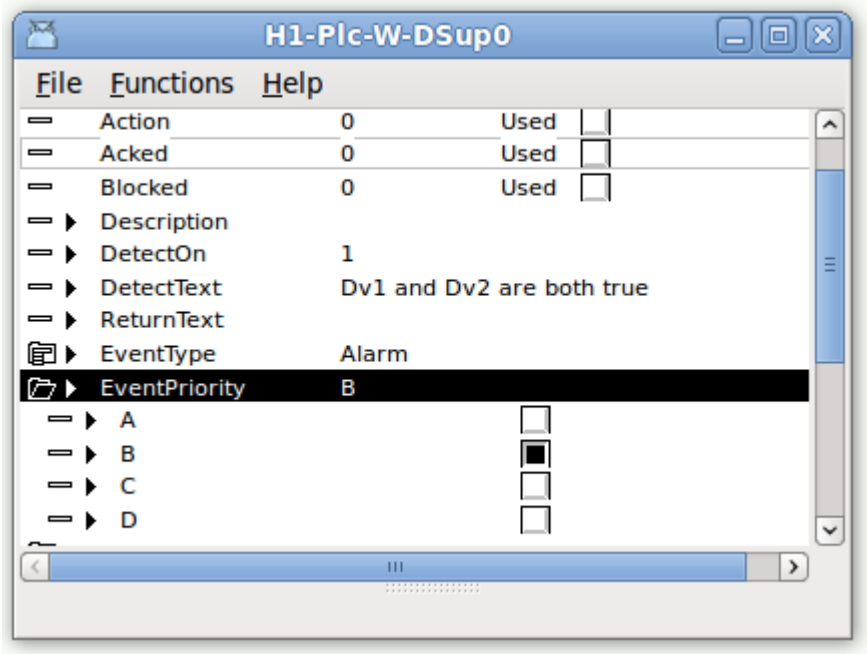
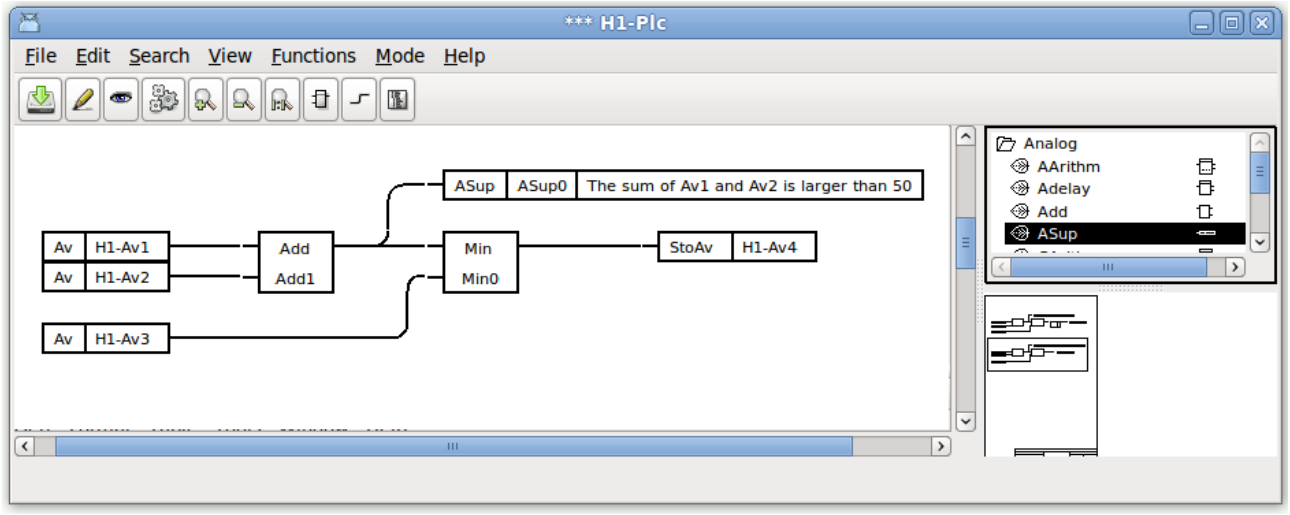


Fig Object editor for DSup

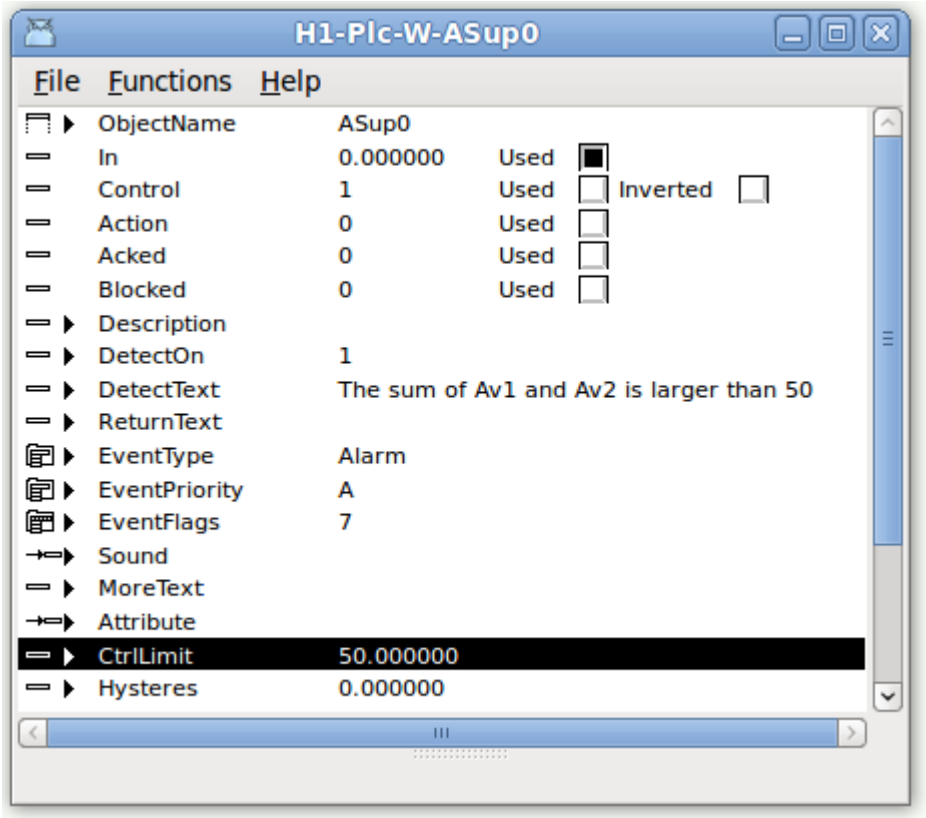
Also set the attribute EventPriority to B.



Create also an ASup object, this is found under the Analog map in the palette, and connect it to the add object



Open the Object editor for the ASup object, and insert the alarm text into the DetectText attributes, and the alarm limit, 50, into the CtrlLimit attribute.



Save and close the plc editor.

We also have to specify that our operator are interested in the alarm from the H1 hierarchy. This is done in the OpPlace object for the operator place. We enter edit mode in the configurator, open the OpPlace object, Nodes-Gettingstarted-Op, and insert H1 into the attribute EventSelectList[0]. The EventSelectList is an array where we can enter several hierarchies from which alarms should be displayed. (In V4.6.1 and prior versions, the selectlist is placed in a User object below the OpPlace object).

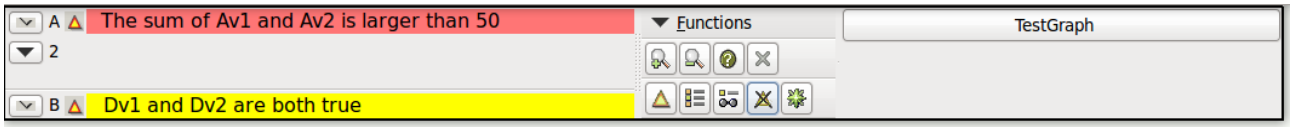
Open the messagehandler object in the node hierarchy, Nodes-Gettingstarted-MessageHandler, and set the attribute EventLogSize to 5000. This will create an database where events and alarms are stored.

Save, leave edit mode and build the node.

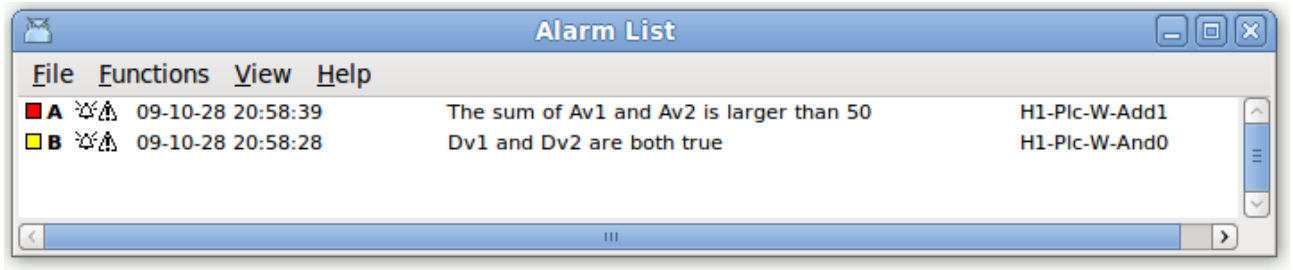
To see how it works in runtime, we open the Runtime Monitor and stop the runtime environment and then start it again.

Then open the operator place by activating *File/Start Operator Environment* in the menu and choosing Nodes-Gettingstarted-Op.

Set the values of H1-Dv1 and H1-Dv2 to 1 to trigger the DSup alarm, and set the value of H1-Av1 to 60 to trigger the Asup alarm. The alarm are viewed in the left part of the operator window, the B alarm yellow and the A alarm red. A beep indicates that there are unacknowledged alarms present. Acknowledge the alarms by pushing the tick buttons to the left.



Open the alarm list from the Alarmlist button in the operator window.



You can also see the alarms as events in the eventlist and search for them in the eventlog database.

More Process Graphics

Now we will work a little more with our process graph. Open the graphical editor again, by finding the XttGraph object in the Configurator and activating *Open Ge* in the popupmenu.

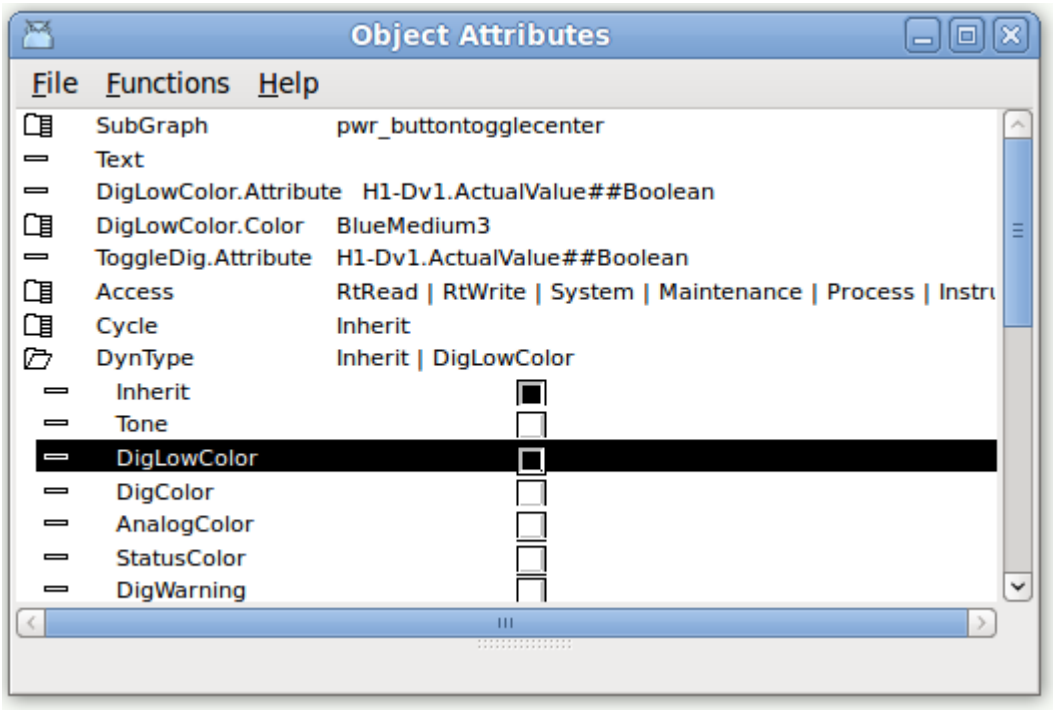
Change the color and text of a pushbutton

So far we have only used the default dynamics of the graphic components. We shall now see how add easy can add dynamics and we start with one of the pushbuttons, the one that toggles the value of Dv1. Lets say that when Dv1 is low, we want the color of the button to be blue, and have the text 'Set Dv1', and when Dv1 is high, the color should be green, and the text 'Reset Dv1'.

Doubleclick on the pushbutton and open the Object Attributes editor.

To change the color of the button we can use the dynamic DigLowColor. Open the map DynType and set the checkbox for DigLowColor. Now the attributes for DigLowColor, DigLowColor.Attribute and DigLowColor.Color, are added to the attribute list.

DigLowColor.Attribute is the value that should change the color, which should be Dv1, and to insert Dv1 we activate View/View Plant i the menu to replace the palette with the plant hierarchy. Open the H1 map and select Dv1. Doubleclick on the DigLowColor.Attribute line and Dv1 should be inserted.



Tip

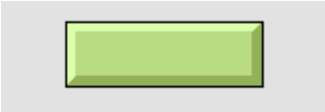
Another way is to use the recall buffer of the input field in the Object Attribute editor. As Dv1 is already present in ToggleDig.Attribute you can select this row, press the *ArrowRight* key to open the input field, and then press *Enter* without changing the value. Now Dv1 is now stored in the recall buffer, so by selecting the DigLowColor.Attribute row, pressing *ArrowRight* to open the input field, and then *ArrowUp* which will fetch the value from the recall buffer, and finally *Enter*, Dv1 is inserted.

The color shall switch between blue and green, and we set the DigLowColor.Color to BlueMedium3. To identify this color you have to look in the color palette. The blue row is divided into three sections, and this it the third color in the middle section.

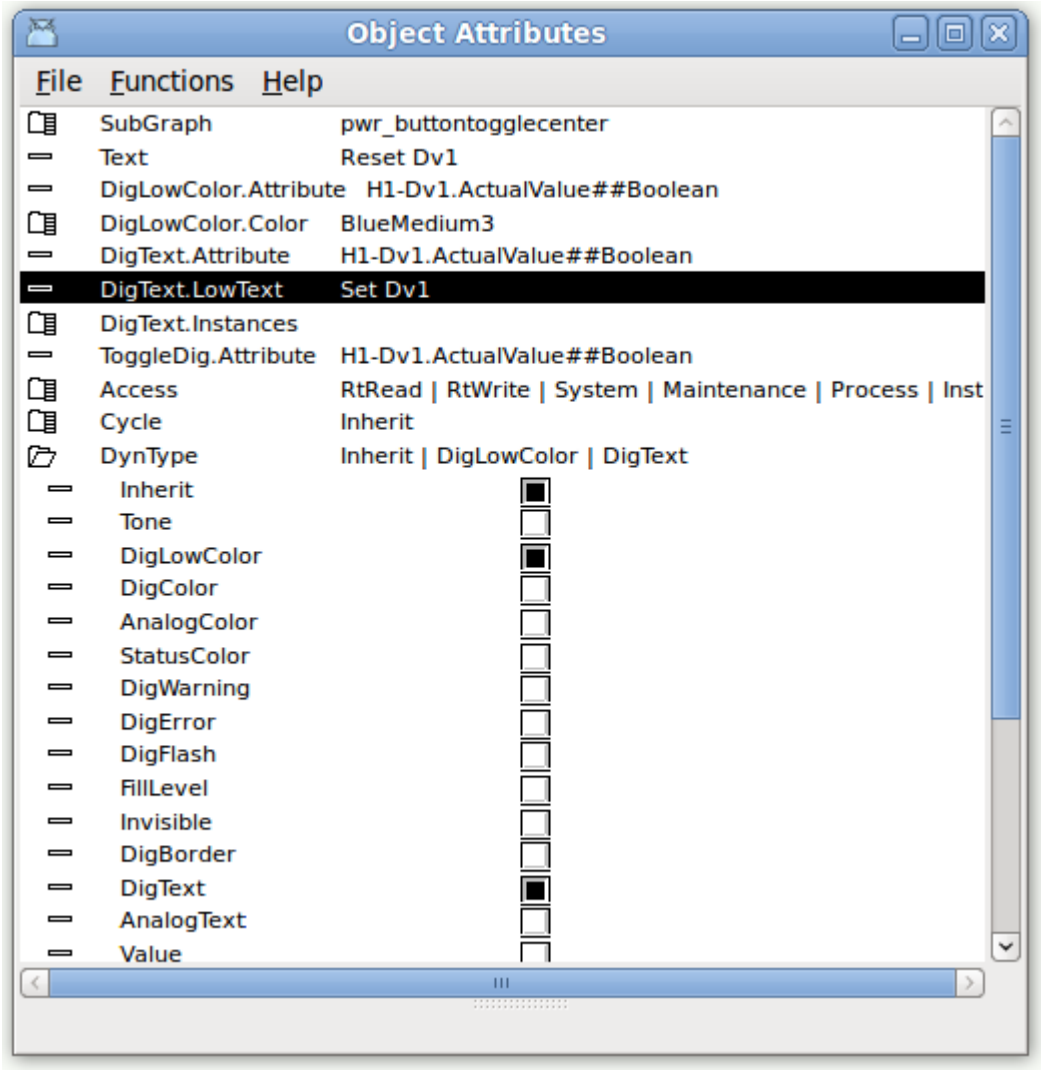
Tip

To insert a color into a color attribute, you can select the color in the color palette, and Ctrl/Doubleclick on the color attribute row in the Object Attributes editor.

The high color should be green, and this is the color we set in the editor. Select the button, and click on a suitable green color in the color palette.



To set a text we use the dynamics DigText. Open the map DynType in Object Attributes and activate the checkbox for DigText. Now the attributes for DigText, DigText.Attribute and DigText.LowText, are viewed. The attribute to change the text is again Dv1, so insert Dv1 in DigText.Attribute in the same way as before. Also insert the the text the should be viewed when Dv1 is low, 'Set Dv1', in DigText.LowText. The high text, 'Reset Dv1', is inserted into the Text attribute.



Close the Object Attribute Editor. And save the graph.

Tip

To check out the result you can activate View/Preview in the Ge menu. Note that the preview can only be used when the runtime environment is started.

The result should be as in the figure below.

Build a graphic component

Lets now see how we can build a graphic component from scratch, for example a flashing indicator.

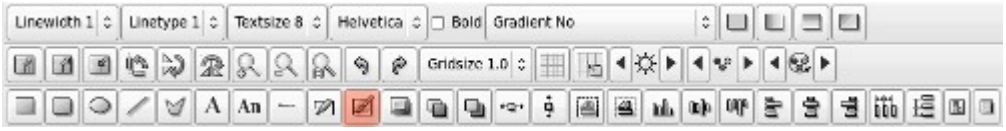
To continue to edit, you first have to stop the preview (View/Preview Stop), if this is started.

We are going to draw two rectangles, one frame and one for the indicator lamp. When we make a group of the we can set the flash dynamic, which we connect to Dv1.

To draw the frame rectangle, push the rectangle button in the toolbar and drag somewhere in the working area.

A rectangle is created.

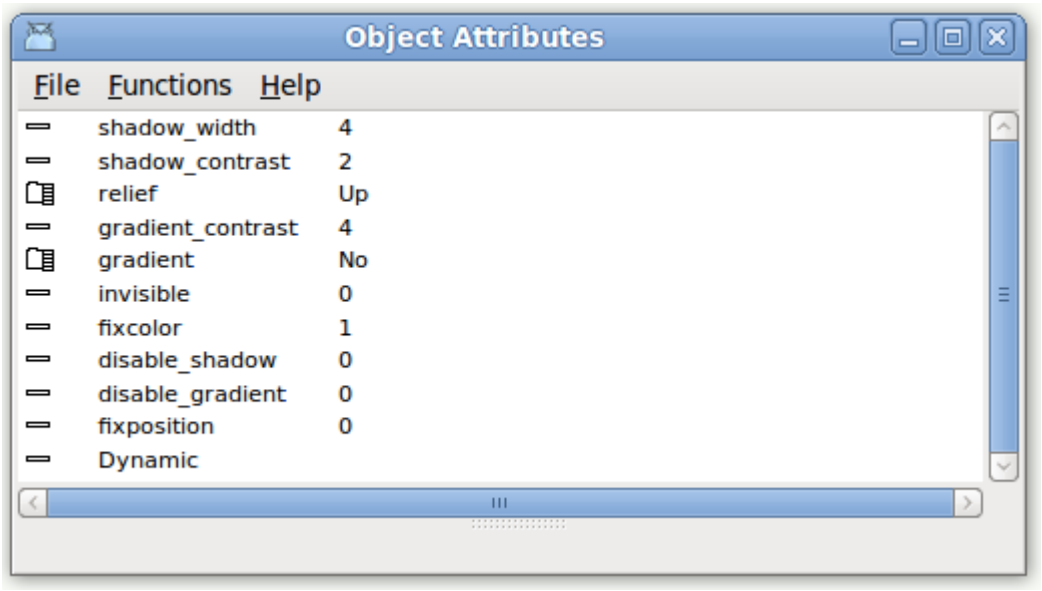
To set a fillcolor, select the rectangle and activate the fillcolor button in the toolbar.



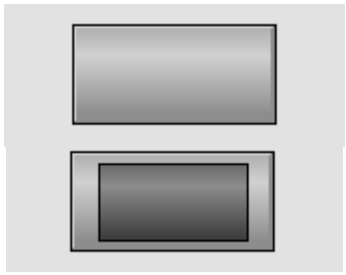
Set a gray color from the second row in the color palette, by selecting the rectangle and click on the desired color in the color palette.

Set a relief shadow on on the rectangle by selecting it and push the 3D button in the toolbar.

Double click on the rectangle and open the Object Attributes editor. Adjust the with of the shadow with the 'shadow_width' property, and set 'fixcolor' to 1, because this frame rectangle should not be affected by the color flashing.

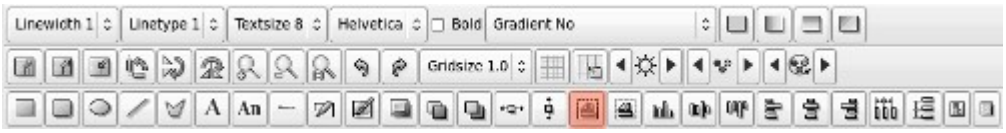


Set a gradient on the rectangle by selecting it and choosing GradientHorizontalTube2 in the toolbar.

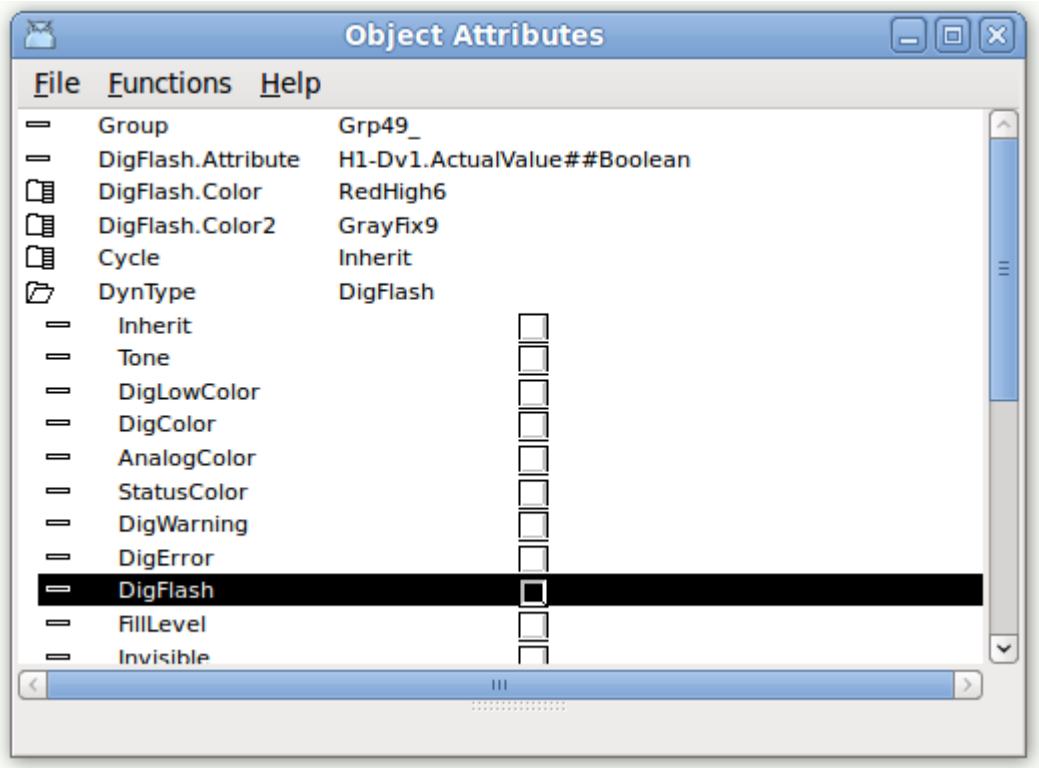


Now, we create another rectangle, a little bit smaller, on top of the first rectangle. This time, we set a darker gray color, but we don't set the 'fixed_color' property, and no 3D property.

Next step is to make a group of the two rectangles. Select them both, and push the group button in the toolbar



When we doubleclick on the group, and open the Object Attributes editor, we can set dynamics on the group. Open the DynType map and set the DigFlash checkbox. Now the attributes for DigFlash are viewed. Insert Dv1 into DigFlash.Attribute, and set the two flashing colors in DigFlash.Color and DigFlash.Color2, for example RedHigh6 and GrayFix9.



Save the graph and start Preview. When Dv1 is set, the indicator should start flashing between red and dark gray.

Write a help text

In this section we will write a help text for the user of the project. The help texts are edited in the file $pwrp_cnf/xtt_help.dat (V4.6.1 and prior versions $pwrp_exe/xtt_help.dat).

A template file is created at project creation. Open the file with you favorite editor, e.g. gedit.

The file is divided in topics by the topic tag, and the main topic for the project is the index topic.

The first row in a topic is a header (in the index topic it is blank).

Now we add a new topic 'testgraph' for our process graph testgraph and create a link in the index topic. By taking a snapshot we create an image of the testgraph which we view in the text with the <image> tag. The imagefile 'testgraph.png' is copied to $pwrp_exe.

```
<topic> index

<image> pwr_logga.gif
```

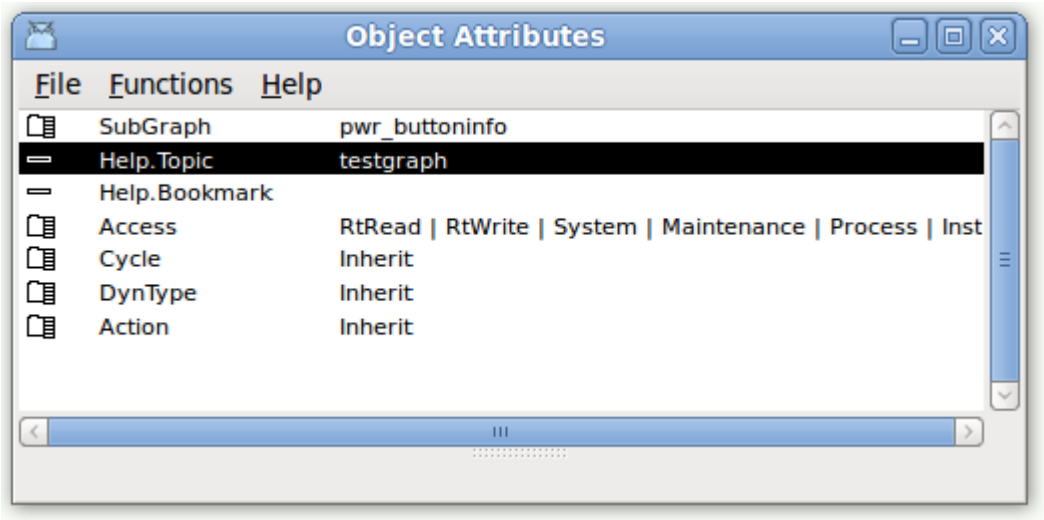
```
<h1>Welcome to gettingstarted
<h2>Description
This a project following the Getting Started Guide in ProviewR.

<h2>Process graphics
Testgraph A test graph <link> testgraph
</topic>

<topic> testgraph
Process graph Testgraph
Push on the three pushbuttons to set Dv1, Dv2 and Dv3.
Activate the slider to set values to Av1, Av2 and Av3.

<image> testgraph.png
<b>Fig The Testgraph graph
</topic>

<include> $pwr_lang/profibus_xtthelp.dat
<include> $pwr_lang/opc_xtthelp.dat
<include> $pwr_lang/basecomponent_xtthelp.dat
<include> $pwr_lang/othermanufacturer_xtthelp.dat
<include> $pwr_lang/abb_xtthelp.dat
<include> $pwr_lang/siemens_xtthelp.dat
<include> $pwr_lang/remote_xtthelp.dat
<include> $pwr_lang/nmps_xtthelp.dat
<include> $pwr_lang/ssabox_xtthelp.dat
```

Now build the node, or copy the file to $pwrp_exe, and you should be able to open the text from Help/Project in rt_xtt or the configurator menu. It can also be opened from the help button in the operator window.

In the testgraph process graph we will now create a button that opens the help text for the graph.

Open the Ge editor for testgraph, and insert an info button from Button/ButtonInfo in the palette.

Open the Object Attributes editor by double clicking on the info button, and insert 'testgraph' in Help.Topic. When the button is pressed, the helptext for the topic 'testgraph' will be displayed.



Open the graph in rt_xtt (this function will not work in Preview), and press the info button.

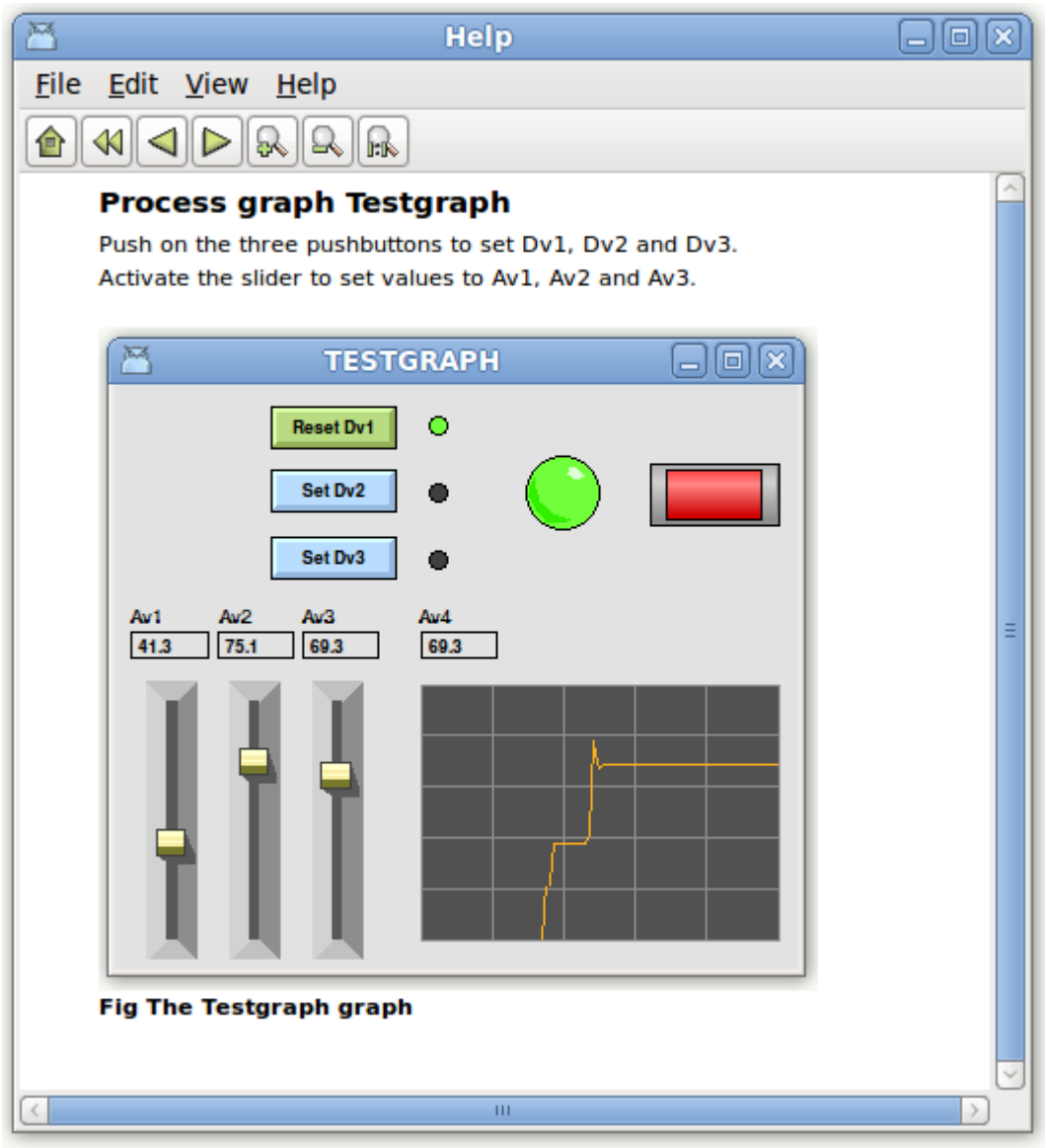


Fig The Testgraph graph

Appendix

Glossary

ProjectList	A list of all projects. Projects are ordered in a tree structure end defined with ProjectReg objects. Projects can be opened from the ProjectsList and this is also where new projects are created.
GlobalVolumeList	A list of all volumes in all projects. All volumes has to be registered in this list.
Volume	A container for objects. There are a number of different types of volumes. A rootvolume, for example, is loaded into a process or operator station at startup and contains all object needed to configure the station.
DirectoryVolume	Every projects contains one directory volume, that configures the nodes and volumes in the project.
Root Volume	The root volume is loaded into a process or operator station at ProviewR startup. It usually contains all the configuration objects for this node.
Object	An object contains a data area divided in attributes. The attributes store data configured in the Configurator, or data calculated in runtime. An object can also have methods, object graphs etc. Every object is specified by a class, where the attributes, methods, plc code etc. are defined. Objects are placed in an object tree, where objects can have parents, siblings and children.
Plcprogram	A control program built by logical and analog functionblocks.
Configurator	A tool to navigate in the object tree and to configure the.
UserDataBase	A database where ProviewR users are defined with username, password and privileges. Contains by default the systemgroup Common with users pwrp and b55. A projects link to a systemgroup and get the users for this group.
Project	A number of processtations, operatorstations and storagestations are gathered int a project. It contains a directory tree with databases and configuration files for the nodes and volumes that belongs to the project.
sdf	A shell command to set up the environment for a project (set default project) `> sdf 'projectname'`
Plant hierarchy	Configures the plant with objects for signals, sensors and components. Also PlcPgm's containing plc code is configured in the plant hierarchy.
Node hierarchy	Configures the hardware, IO-system, and system processes of the node.

Edit functions in the Configurator

Function		Action	Comment
Select Object	Mouse	LeftClick	
	Key	Arrow Up and Down	
Create Object	Menu	Edit/Create Object/After (Ctrl+D) Edit/Create Object/First Child (Shift+Ctrl+D)	Select an object class in the palette first, then select the destination object (parent or sibling), and then activate Create Object in the menu.
	Mouse	MiddleClick	Click on the icon (map or leaf) creates the object as child to the clicked object. Click on the text creates the object as sibling.
Delete Object	Menu	Edit/Delete Object (Delete key)	Deletes the selected objects.
	PopupMe nu	Delete Object	Select the object first.
Move Object	PopupMenu	Move Selected Object	Select the object you want to move the object to first.
	Mouse	MiddleClick	Select the object you want to move, and middleclick on the destination object. Click on the icon to move as child. Click on the text to move as sibling.
Show Children	Mouse	LeftClick on map icon.	
	Key	Arrow Right.	Note that arrow right key also shows the attributes of the object is no children are present.
Open object	Key	Shift+Arrow Right. Arrow Right (if no children).	The arrow key displays the attribute in the current window.
	PopupMenu	Open Object	Opening from the menu opens a new window.
	Menu	Functions/Open Object (Ctrl+A)	
Change attribute value	Key	Arrow Right.	Enter the value into the input field and press enter.
	Menu	Functions/Change Value (Ctrl+Q)	

Table 1 Edit functions in Configurator

Edit functions in the Plc Editor

Function		Action	Comment
Select Object	Mouse	LeftClick	
	Key	Shift+Arrow Keys	
Create Object	Mouse	LeftClick	Select an object class in the palette with the mouse. A functionobject is now moved with the cursor. Place it with a LeftClick.
		MiddleClick	If a class already is selected, or is selected by the Arrow Up and Down keys, Middleclick in the work area to create an object.
	Menu	Edit/Create Object (Alt+D)	
Delete Object	Menu	Edit/Delete (Delete key)	Deletes the selected objects.
	PopupMe nu	Delete	
	Mouse	MiddleDoubleClick	Hit on an object deletes the object. Hit in en empty space deletes all selected objects.
Move Object	Mouse	LeftDrag	If several objects are selected, and a selected objects is dragged, all selected objects will be moved.
	Key	Shift+Control+Arrow keys	
Create Connection	Mouse	MiddleDrag	Drag between two pins. If the connection is dragged from one pin and released in an empty space, a predefined object is created (normally a Get or Sto object).
	Key	Ctrl+D	Select a connection point with Ctrl+Arrow keys, lock this with Shift+Ctrl+D, and select a second connection point with Ctrl+Arrow keys.
Open object	PopupMenu	ObjectEditor	Opening from the menu opens a new window.
	Menu	Functions/Open Object (Ctrl+A)	
Connect	Mouse	Ctrl+LeftDoubleClick	Select an object in the Plant hierarchy.
	Menu	Edit/Connect (Ctrl+Q)	
	PopupMenu	Connect	
Open	Mouse	Shift+LeftDoubleClick	

Subwindow	Menu	Functions/Open Subwindow	
	PopupMenu	SubWindow	

Table 2 Edit functions in the Plc Editor