ProviewR

OPEN SOURCE PROCESS CONTROL

Operator's Guide

2024-01-17

Version 6.1.5

1

2 Introduction

ProviewR Operator's Guide is intended for persons who will be in contact with running ProviewR systems in their daily work. It could be

- operators that supervise and handle a process.
- maintenance personnel troubleshooting the plant.
- process engineer fetching information about the process.
- system managers maintaining and debugging the system.

Function for all these groups are available in the ordinary operator environment, and who belongs to a certain group are determined from the privileges the user is granted.

3 About ProviewR

What is ProviewR

ProviewR is a modern, powerful and general process control system. It contains all the functions normally required for successful sequential control, adjustment, data acquisition, communication, supervision, HMI/SCADA, history data storage etc.

ProviewR is a distributed system, it consists of a number of computers (nodes) connected in a network. The nodes can be of type process station, operator station or storage station.

3.1 Stations

Process stations

A process station collects measuring data from different sensors and switches, it can be analog measuring data as temperatures, flows, levels, or digital measuring data from for example photocells or pressure switches. A control program is executed in the process station, that from the measuring data calculates control data for the process that is sent out to motors and actuators affecting the process.

Process stations often contains special hardware to read input data and write output data, but this can also be performed over the network with protocols as Modbus/TCP and Profinet.

Operator station

The operator station is the interface between the operator and the process. The operator supervises the process with the help of process graphs, displaying the state of the process by presenting data in the shape of bars, curves, indicators etc. The operator can influence the process, for example by entering data into input fields or pushing buttons in the process graph.

Storage station

Some measuring data has to be stored, in order to view trends and changes, or to be able to get back and analyze the state of the process at a certain point of time. The storage time can vary from hours to several years. The storage is done on storage stations, that have sufficient disk space to be able to store data for long periods, and also have backup functions.

Development station

There can also be development stations on the network, where process, operator and storage stations are configured and programmed. On the development stations there are tools to draw process graphs, program sequences, logical schemas, control loops etc. When the configuration is modified for a station, the new configuration is distributed to the station

via the network.

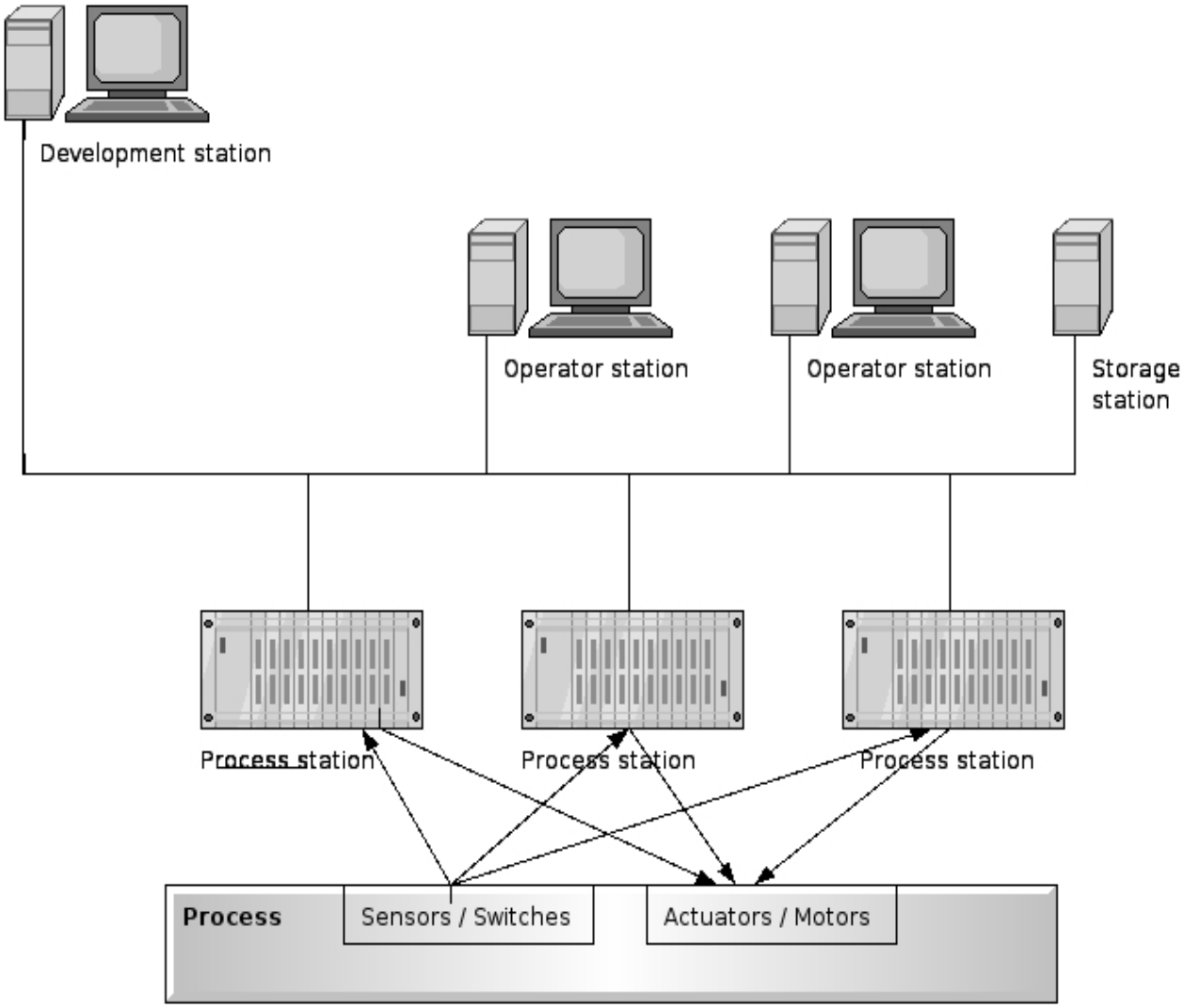


Fig Components in a ProviewR system

3.2 Objects

The object tree

Object is a central concept in ProviewR. Sensors, valves, motors, controllers etc. are all represented by objects in a database. The objects are ordered in a tree structure where different hierarchy levels in the tree represent parts of the plant or process. If we look at a furnace, the top level can represent the whole furnace. The furnace consists of four warming zones, which are represented by objects on the next level in the object tree. Each zone has two electric heaters and two temperature sensors, still one level down. Here are also some object that don't have any physical equivalent in the plant, a PID controller and a control program for the zone.

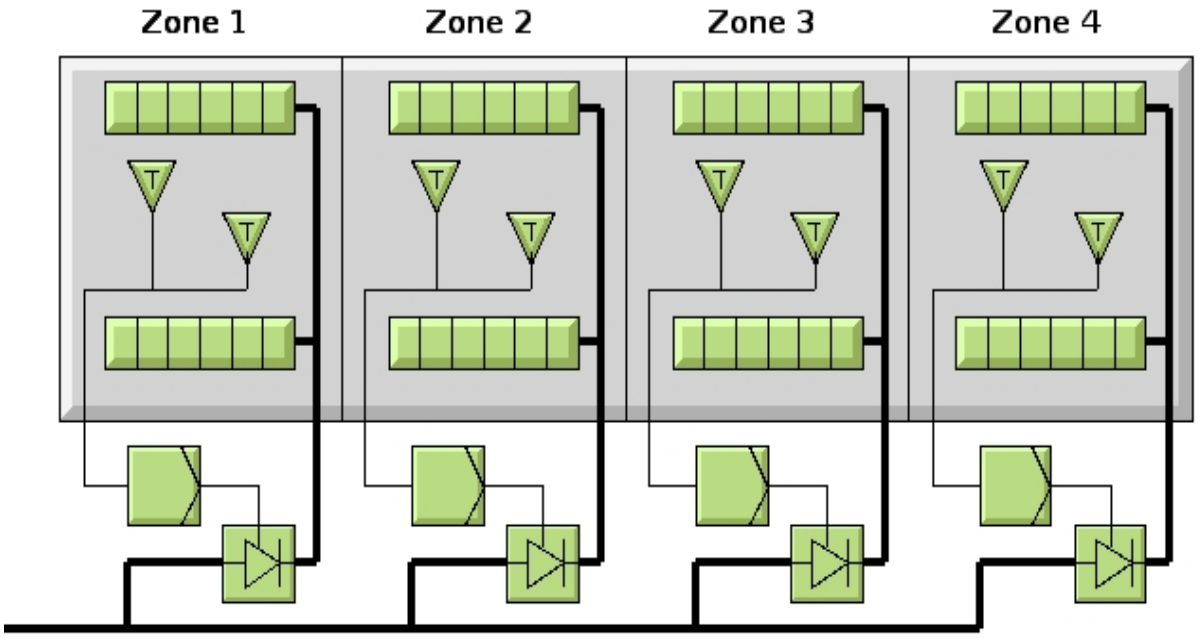


Fig Furnace with 4 zones

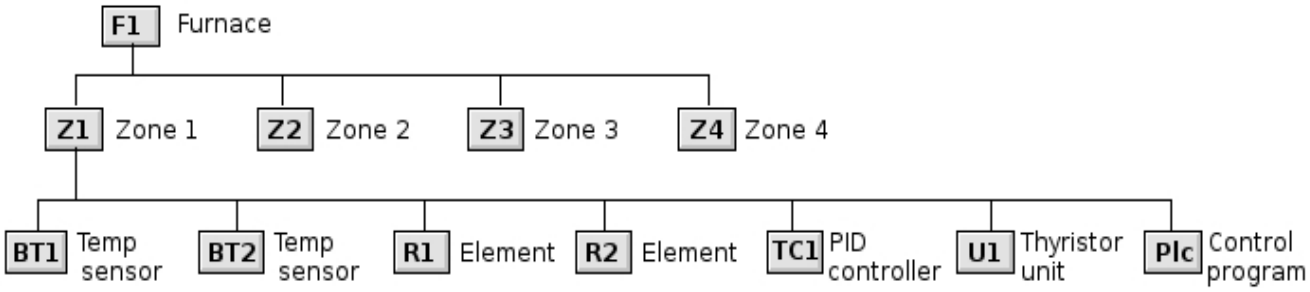


Fig Object tree for the furnace

Note that several objects have quite cryptic names that follow a notation standard, but to each object there are a more detailed description that is often viewed with the object.

Let us have a closer look at an object, for example one of the temperature sensors. The object is shown in an overview process graph with the graphical symbol of the object, a triangle with the letter T.

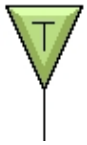


Fig Graphical symbol for a temperature sensor

Object graph

If you click on the symbol the object graph of the object is opened. From the object graph you can see the properties of the object. The most interesting for a temperature sensor is of course the measured temperature, that is viewed both in figures and as a bar

(thermometer), see 'Fig Object graph for a temperature sensor' below. The temperature sensor object also contains four alarm limits, highhigh, high, low and lowlow. The level of the alarm limits can be set from the graph, and also a hysteresis on the limits. With check boxes you can also disable the alarm limits.

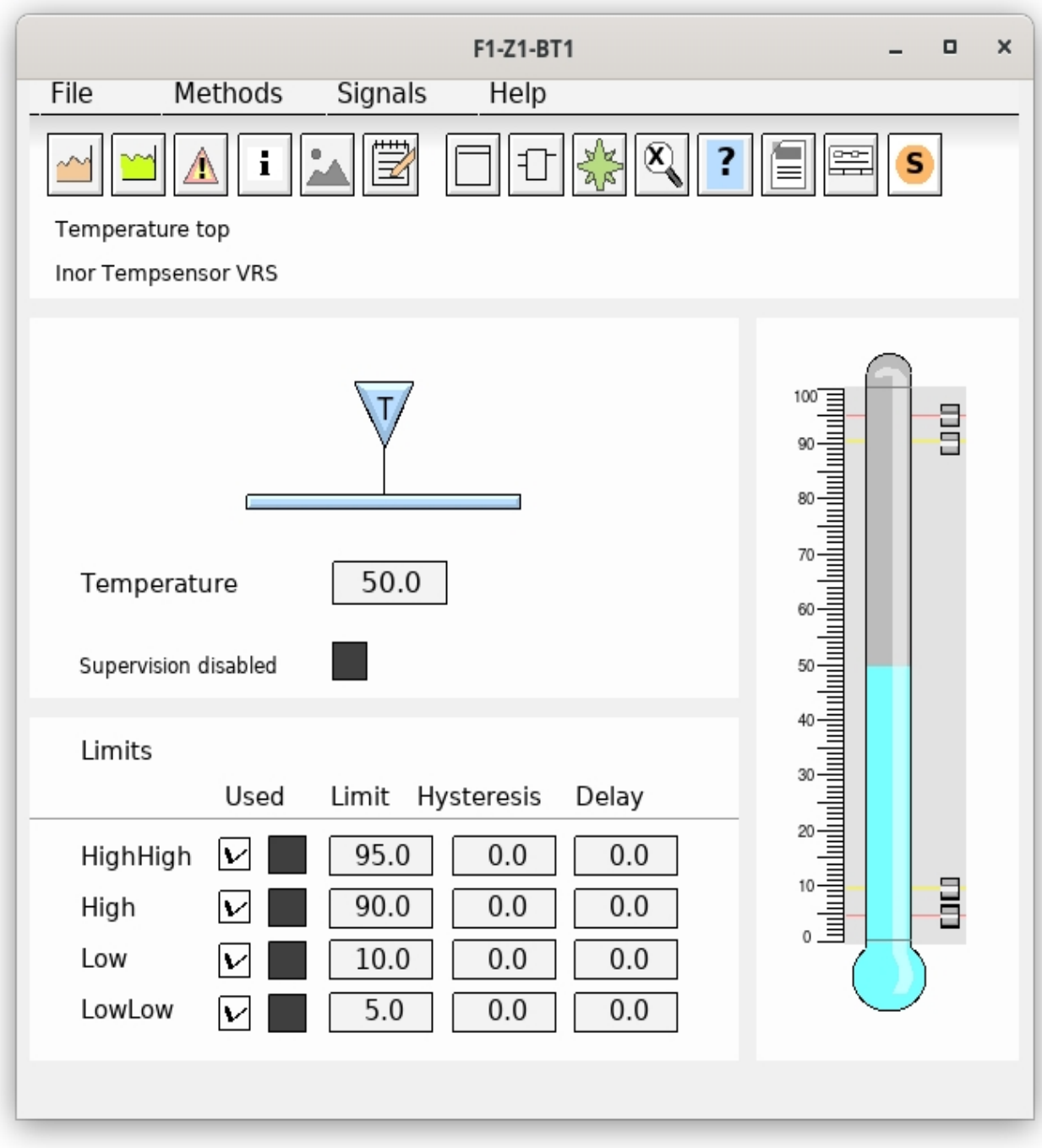


Fig Object graph for a temperature sensor

Methods

In the upper edge of the object graph there are a number of pushbuttons that activate the methods of the object. By the methods you can reach all the information there is on the objects in the control system. The methods can also be activated from the menu in the objects graph, or by right clicking on the graphical symbol in the overview graph, opening a popup menu with the methods. Actually, wherever you find the object, in alarm lists, process graphs, in the plc code etc, you can by right clicking on the object open the popup

menu with the methods or the object. Here follows a list of different methods, but which methods are relevant for a specific object, depends on the type of object and how it is configured.

Method	**Description**
Object Graph	Open the object graph.
Trend	Open a trend curve.
History	Fetch history data from a storage station and view in a curve.
Hist Event	View a historical list of alarms and events for the object.
Block Events	Block alarms and events.
Help	View a help text for the object.
Note	Write a note for the object. The note is viewed in the object graph.
Open Object	Show the content of the object.
RtNavigator	Show the object in the object tree.
Crossreferences	Show were the object is referenced in the plc code and process graphs.
Help Class	View help for this object type.
Datasheet	Show the datasheet for the component in the plant that corresponds to the object.
CircuitDiagram	Show the circuit diagram for the object.

4 Operator place

4.1 Start the operator environment

Usually the operator environment will be started automatically at login of the workstation. In the login configuration, a operator place object is added to the starting command. In the operator place object the properties of the operator place are configured.

At times though, you may want to start another way, for example from a terminal window.

The operator environment is started by the program rt_xtt, and you specify an OpPlace object as argument. The command to start with the OpPlace object Nodes-OpgNode-b55 is

```
> rt_xtt Nodes-OpgNode-b55
```

There are also a number of options that can be added to the argument list

-l	Language. One the following languages is specified: en_us English (default). sv_se Swedish. de_de German. fr_fr French.
-q	The license window is not opened at startup.
-c	A pushbutton to close the operator environment is shown in the operator window.
-u	Start with a opplace object with the same name as the current linux user.
-s	Displays a list of available OpPlace objects.

4.2 The Operator Window

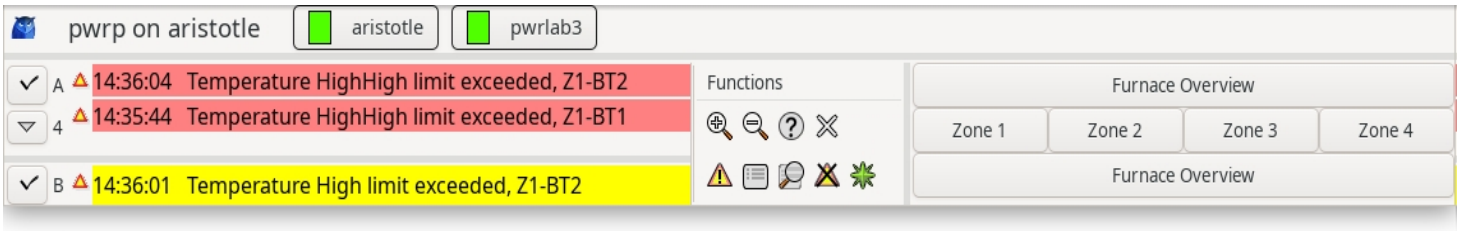


Fig The operator window

The operator window is placed at the upper edge of the screen, and lacks title row and pushbuttons to iconify and delete the window. It is divided in tree parts, to the left current alarms and messages are viewed, in the middle there are pushbuttons for some base functions, and to the left there are pushbuttons configured for this operator place.

Status bar

The top row is a status bar. It shows who is logged in on which node. For tha current node, and for all nodes that there are node supervision configured, an indicator for system status is viewed. Green states that everything is all right, yellow indicates warning, red error and flashing red fatal error. If you click on a node, the status graph for the node is opened.

Alarms and messages

In the left part of the operator window alarm and messages are displayed.

The alarms are divided into four priority levels A, B, C and D, where A has the highest priority and D the lowest. A alarms are marked red, B alarms yellow, C alarms blue and D alarms violet. Furthermore there is the category Info messages that are marked with green.

In the upper field, the latest unacknowledged A alarm are displayed. Depending on the height of the operator window, 2 to 5 alarms are viewed. There is also a button marked with a tick to acknowledge A alarms.

In the lower field the latest unacknowledged B, C, D alarms, and info messages are viewed. These alarms are displayed in priority order rather than in chronological order, i.e. if there are unacknowledged B alarms, the latest of these will be displayed. Only if there are no acknowledged B alarms, possible C alarms are viewed etc. There is also an acknowledge button to acknowledge the B, C, D alarms and the info messaged.

An alarm is prevailing if the alarm condition still is fulfilled. This is marked with a warning triangle to the left of the alarm text.

Menu and pushbuttons to open base functions

In the middle part of the operator window there are a menu and some buttons to open base functions.

Menu
Below is a description of the menu items in the Functions menu.

Functions/Alarm/AlarmList	Open the alarm list with prevailing and unacknowledged alarms.
Functions/Alarm/EventList	Open the event list, a list of the latest alarms and events.
Functions/Alarm/EventLog	Open the event log, where you can display stored events.
Functions/Alarm/BlockList	Open the list of blocked alarms.
Functions/Curves/Trends	Display a list of all trendcurves.
Functions/Curves/Fast	Display a list of all fastcurves.
Functions/Curves/Process History	Display a list of process history.
Functions/Process Graphics	Display a list of all process graphs, defined by XttGraph objects.
Functions/Navigator	Open the navigator.
Functions/View/Zoom in	Increase the text size in the alarmtext.
Functions/View/Zoom out	Decrease the text size in the alarmtext.
Functions/User/Switch User	Open the login window to login as another user.
Functions/User/Show User	Show the current user.
Functions/User/Return	Return to the original user.
Functions/Help/Project	Show helptexts for the project.
Functions/Help/Overview	Show an overview of helptexts.
Functions/Help/Operator Window	Display help on the operator window.
Functions/Help/About ProviewR	Display ProviewR into, for example version of install package.
Functions/Close	Close down the operator place.

Pushbuttons

Below the menu there is a tool panel with a number of pushbuttons that works as shortcuts to menu items:

- zoom in and out.
- show helptexts for the project.
- close.
- show alarmlist.
- show eventlist.
- show eventlog.
- show list of blocked alarms.
- open the navigator.

Function buttons

To the right there are a number of pushbuttons that are used to open process graphs or execute various orders or commands. How these buttons are used depends of the configuration of the operator place.

5 Alarms and Events

5.1 About alarm and events

Alarms

Alarms are sent to the operator station when something occurs that operator has to pay attention to. It could be a temperature crossing a limit, or a bad value entered in a process graph. In ProviewR there are special supervision objects that supervise signals and generates alarms.

Alarm are viewed in the alarmlist.

Priority

The alarms are grouped in four priority levels, A, B, C and D, where A is the highest priority and D the lowest. How the priority levels are used, depends on the configuration of the system. It is common that the alarm priorities are used in the following way.

A alarms are marked with red. They have the highest priority and indicate that a serious error in the plant has occurred that immediately should be taken care of. Often it causes a stop in the production as long as the alarms are prevailing.

B alarms are marked with yellow. They have a little lower priority, implying an error that that should soon be taken care of, but the production can continue still some time.

C alarms are marked with blue and D alarms with violet. They indicates minor errors that are not acute. In many system two alarm levels are enough, and only A and B alarms are used.

Alarmtext

Alarms contain text that is viewed in alarm and event lists. This text is in one row and maximum 80 characters long. There is also room for longer text, a moretext, that for example, can contain further explanation of the cause of the alarm, or how it should be taken care of. The moretext is displayed in the alarm and event lists if the cursor is placed on the alarmtext.

Acknowledge

An alarm has to be acknowledged by the operator. As long as the alarm is unacknowledged, it remains in the alarmlist and gives rise to a beep. If an alarm is displayed at several operator places it is enough for one of the operators to acknowledges the alarm.

Blocking

If an alarm is not relevant during a period, it is possible to block the alarm. You can, for example, block the alarms for a part of the plant that is not in production. The

blocking can be performed on a separate alarm object, or for a hierarchy. Blocking is a method of an object, and performed from a blocking window that is opened from the popup menu for the object. Only users that are granted the privilege RtEvents are allowed to block alarms.

Blocked objects are displayed in a blocklist.

Messages

Info messages are a group that has the same functionality as alarms. They have lower priority than alarms and are marked with green.

Events

An event is generated in the same way as alarms, by supervision objects. Also alarms generates events. Activation of an alarm is counted as an event, as are the return of the alarm state and the acknowledgement of the alarm.

Events are stored in an eventlist, that contains the latest events. They are also stored in the event log, where you can go back and look at events for some period of time, and also see event statistics. For the eventlog there is a search dialog, where you can search for events with certain search criteria, for example time, eventtype or eventname.

Selectlist

The operator place has a selectlist that contains a list of plant parts in the object tree. Only alarms from these plant parts will be viewed in the alarm and event lists. The other alarms and events are filtered away.

5.2 Alarmlist

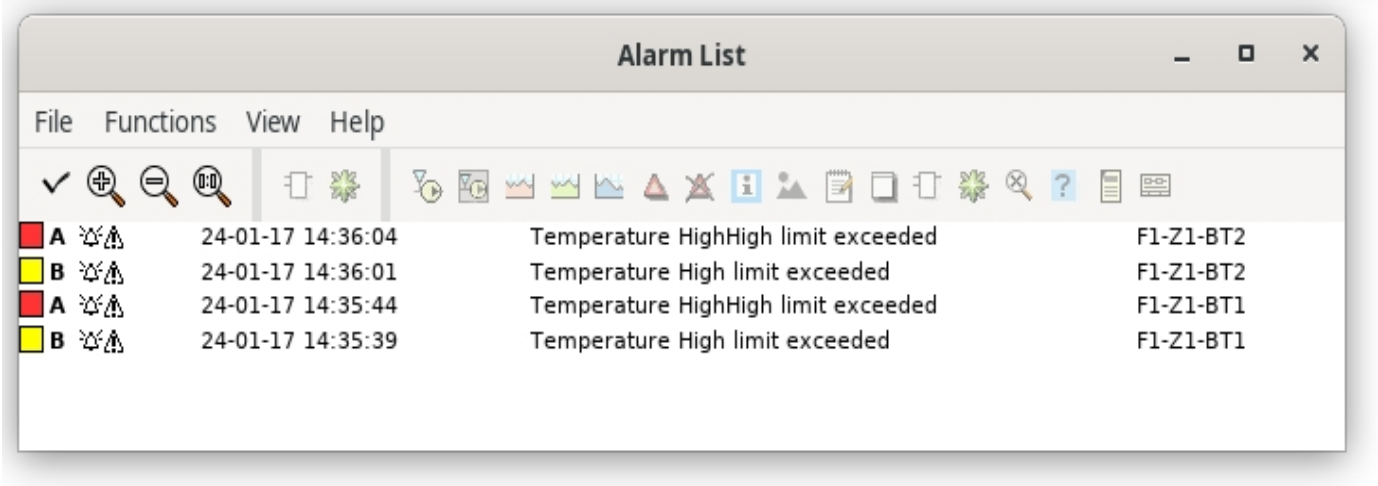


Fig Alarmlist
The alarmlist is opened by activating the "Alarmlist" button in the operator window. You can also open the alarmlist from the menu in the navigator, 'Alarm/Alarm List', or by the xtt command 'show alarmlist'.

In the alarmlist prevailing and unacknowledged alarms are viewed. Unacknowledged alarms are

marked with a bell and prevailing alarms with a warning triangle. Only alarms from the plant parts stated in the selectlist are viewed.

The moretext of an alarm is displayed when the cursor is placed on the alarmtext.

Alarms can be acknowledged by activating Functions/Acknowledge (Ctrl+K) in the menu.

If you rightclick on an alarmtext, the methods of the alarm objects are displayed. Thus making it easy to open the object graph or find where in the plc code the alarm is generated.

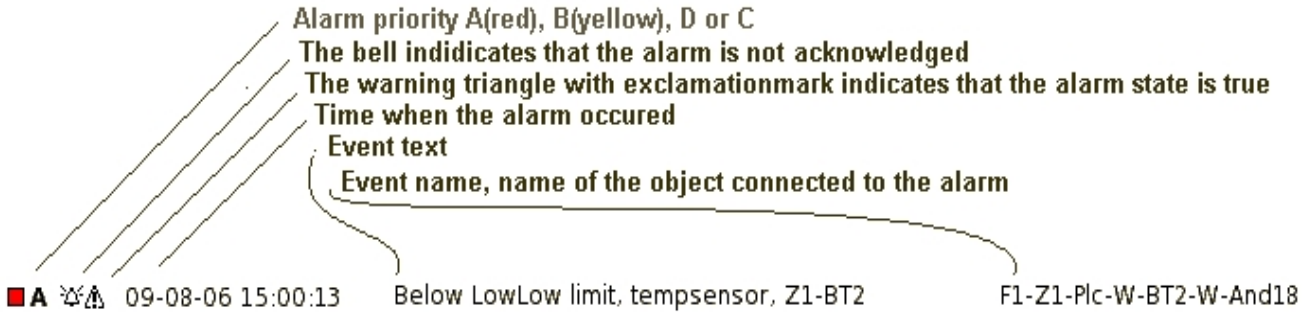


Fig Alarmtext

5.3 Eventlist

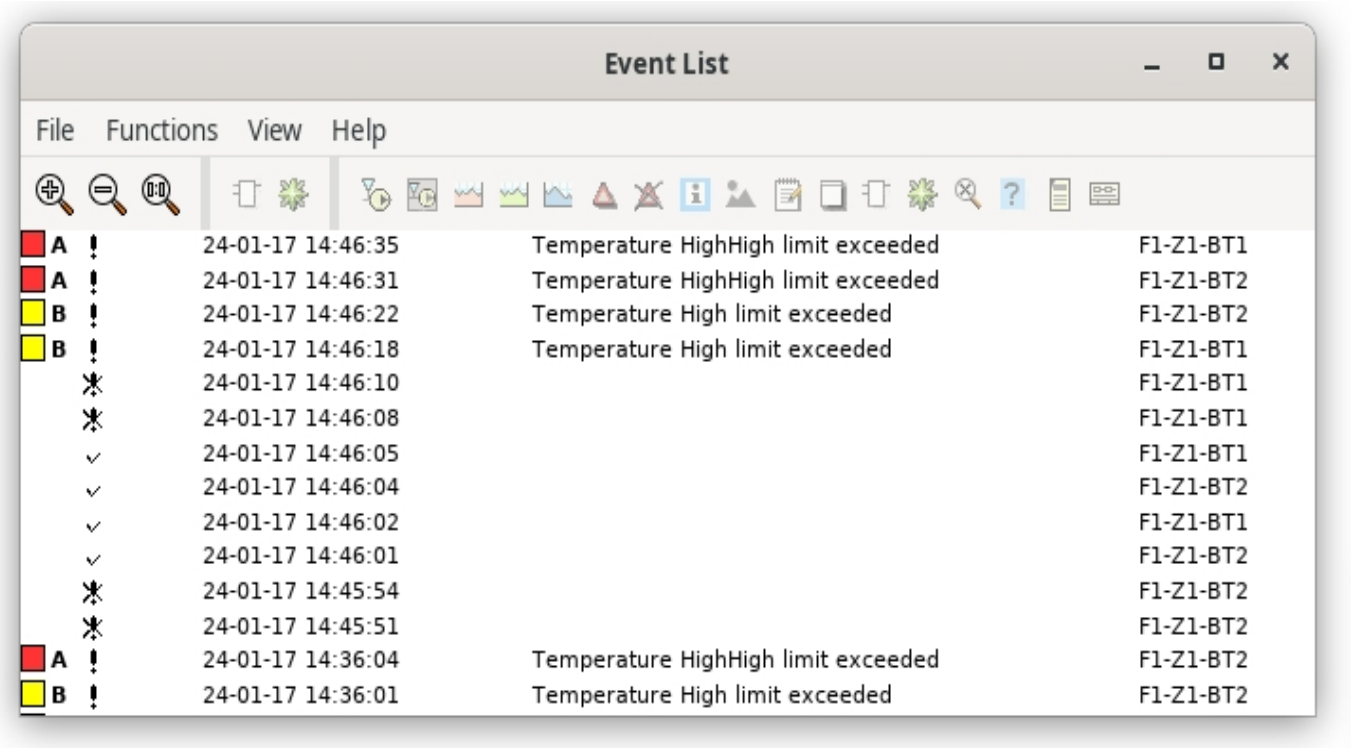


Fig Eventlist

The eventlist is opened from the "Eventlist" button in the operator window. It can also be opened from the navigator menu or by the command 'show eventlist'.

The events from the plant parts stated in the selectlist are displayed in the eventlist.

The maximum number of events are configured in the User object. When the maximum number of events is reached, old events are removed when new ones arrives.

Each event indicates
- Color indication for alarm priority.
- Event type.
- Time of event.
- Event text.
- Event name (object/signal name).

The moretext for an event is viewed when the cursor is placed on the event text. If you rightclick on the eventtext, the methods of the objects are displayed. Thus making it easy to open the object graph or find where in the plc code the alarm is generated.

5.4 Eventlog

The eventlist has a limited length and displays only the latest event. All events though are stored in a database, the eventlog. From this it is possible to look at events for a longer period of time.

As the number of stored events often are several thousands, you search for the events you are interested in with different search criteria. The events matching the search critiera are diplayed in a list.

Search criteria

The event log dialog is viewed in figure 'Eventlog' below.

Time

On the upper row you can specify a time interval of the search. State requested interval with the optionsmenu to the right. If you select 'Time' the time interval is inserted into the input fields.

Eventtype

An eventtype is specified. If no type is selected, all types are present in the search.

- Active	An alarm is activated.
- Message	An info message.
- Return	An alarm returns from active state.
- Ack	En alarm is acknowledged.

Priority

Search for alarms with a specific priority. If no priority is selected, the search is made for all priorities.

Eventname

Name of the object the event is connected to. Wildcard (*) is allowed, for example F1-Z1* will search for all events in the hierarchy F1-Z1. If no object name is supplied, all

objects are searched for.

Eventtext

Search for an event text. Wildcard (*) is allowed. If no text is specified, all texts are searched for.

When the search criteria are stated, the 'Search' pushbutton is pressed. The search is performed and the result is presented in a list in the lower part of the window.

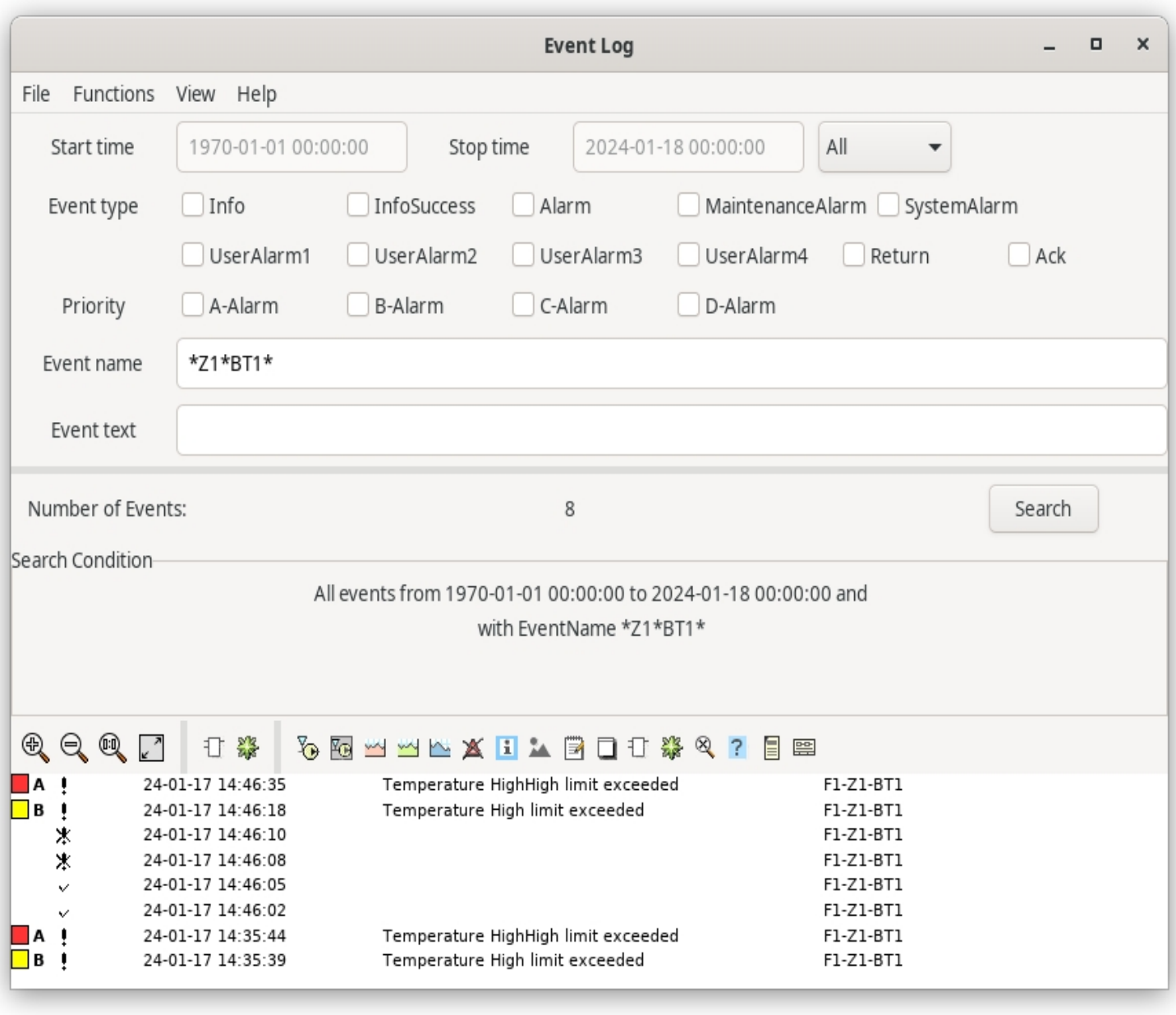


Fig Eventlog

In the list of events an event is displayed with

- Priority — The priority is marked with color and a letter.
- Eventtype — Active is marked with an exclamation mark, Return with an crossed-out exclamation mark and Ack with a tick.
- Eventtext
- Eventname

If you rightclick on a row the methods of the eventname object are displayed.

5.5 Blocklist

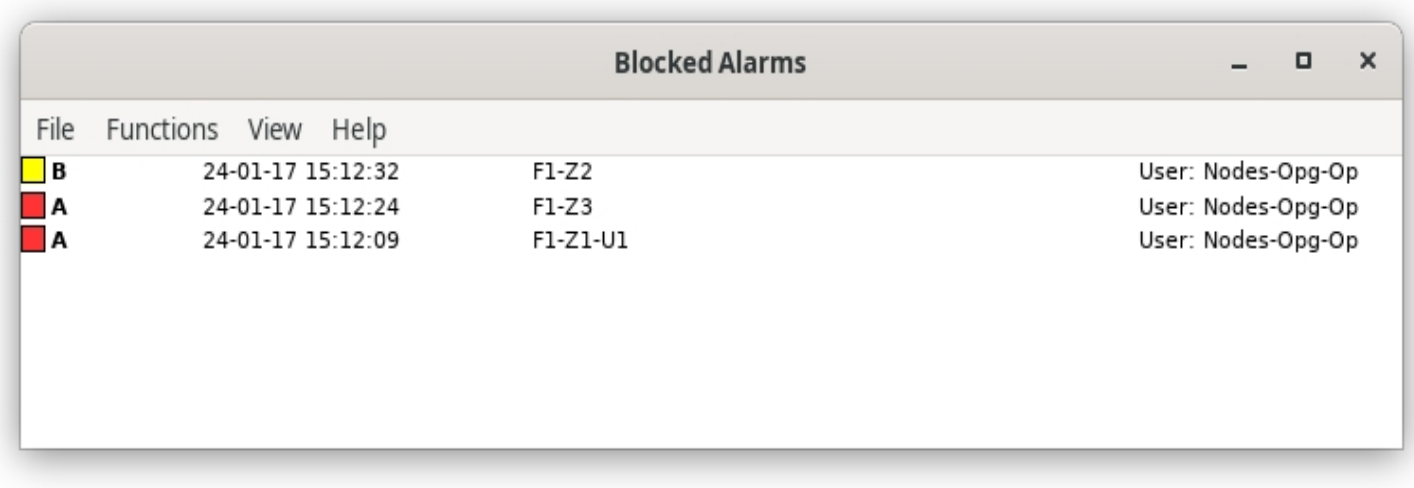


Fig Blocklist

The list of blocked alarms is opened from the menu in the navigator, Alarm/Blocket Alarms, or by the xtt command 'show blocklist'.

For each blocked objects are displayed:
- Color indicating the blocked priority level.
- Object name.
- User that has made the blocking.

A blocking is removed by selecting the blocking and activating Functions/Remove Blocking in the menu.

Alarm blocking is performed by the 'Block Events' method for an object.

5.6 Alarm blocking

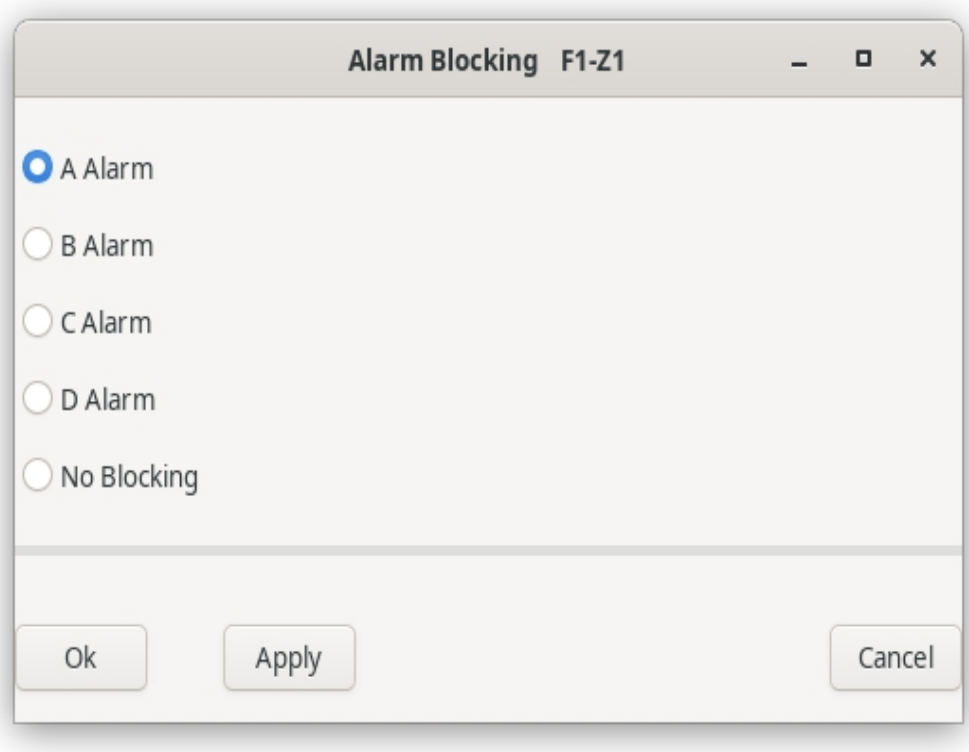


Fig Dialog for alarm blocking

Alarm blocking is activated by the 'Block Events' method, i.e. from the popup menu for an object, or with the xtt command 'eventlist block'.

The window is used to block alarms for an object. All alarms with the specified priority, and with lower priorities, are blocked for the object and all its descendants.

The priority is specified by radio buttons. You can also remove blocking by the 'No Blocking' button.

Only users with the privilege 'RtEvent' or 'System' are authorized to block alarms.

6 Curves

6.1 Trends

Trendcurves are opened from the 'Trend' method in for example the popup menu. They can also be opened by the Xtt command 'open trend'. Trend curves are configured by DsTrend objects and, if several curves are viewed in the same curve window, by PlotGroup objects.

A trendcurve is a signal value, for example of type temperature, pressure or flow, that is stored with a specific time interval. A trendcurve can store 478 values, and so the total time window depends on the time resolution. If a new value is stored every second, a total of 8 minutes can be stored, and with a new value every minute, eight hours can be stored. The values are stored in the RAM memory, and will be lost at a reboot.

The trendcurves are viewed in a curve window. A curve window can display 20 different curves.

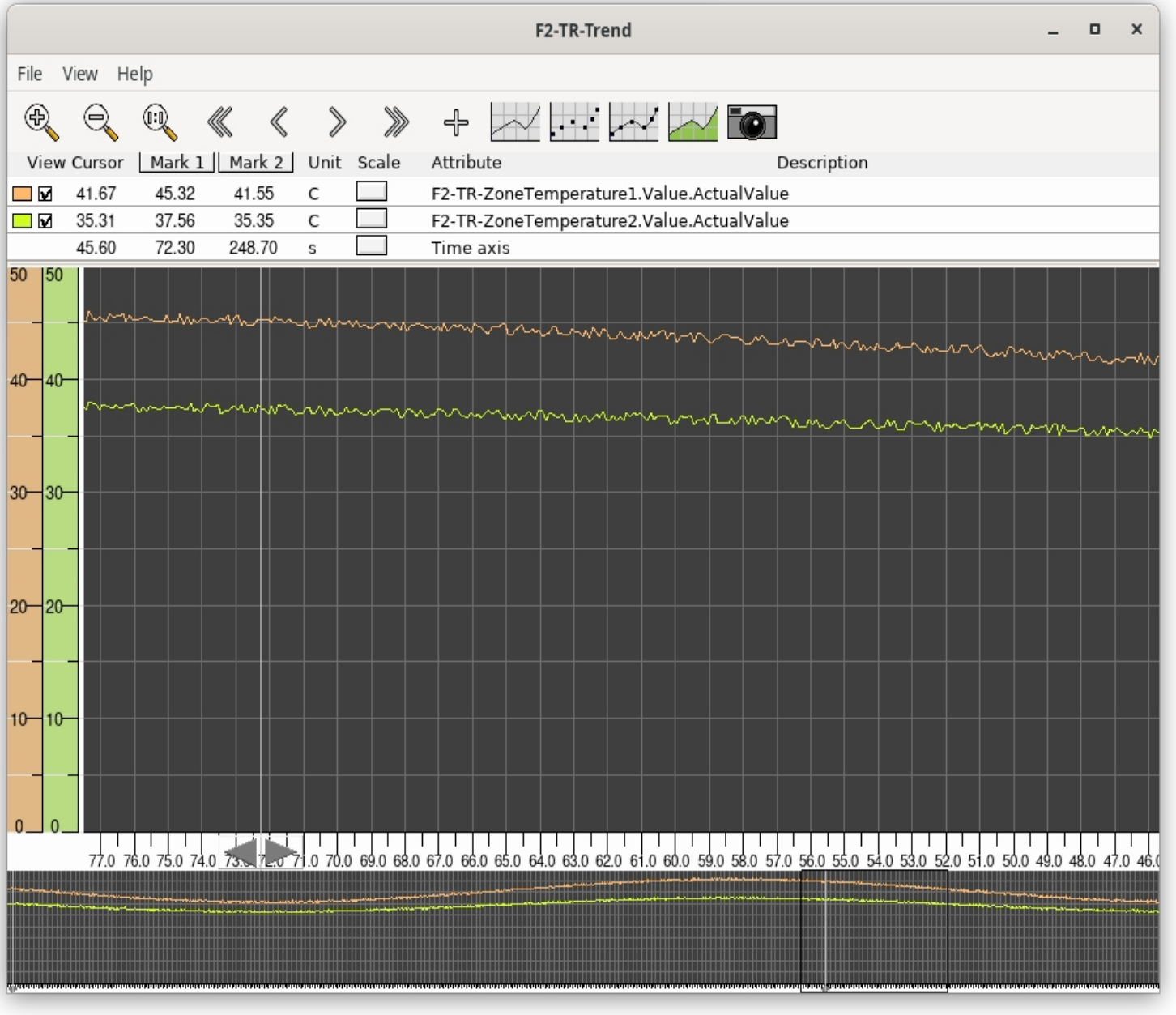


Fig Trendcurves

The upper part of the curve window views a table over the curves. The first column is a color indication to identify the corresponding curve.

Description of columns in the table

View	A checkbox from which you can choose to view or hide the curve.
Cursor	Shows the curve value where the cursor is placed.
Mark1	Shows the curve value where the first marker is placed.
Mark2	Shows the curve value where the second marker is placed.
Unit	Displays the unit of the curve.
Scale	From 'Scale' you can change the scale in y direction (Scale on the Time axis row changes in x direction).
Attribute	States the attribute of the trend curve.
Description	Description of the attribute.

Navigation

The tool panel contains buttons to zoom and move the view to the left or to the right.

You can also navigate with the navigation window in the lower part of the window. The part of the curve that is displayed, is marked with a black square. With drag MB1 you kan move the view in x direction, and by dragging the middle button you can zoom in and out.

Navigation from the keyboard is accomplished with the arrow keys. With arrow up and down the zoom factor is changed, and arrow left and right moves the view to the left and to the right.

Markers

There are two markers in the window that can be positioned at specific time values. The curve value for the position is displayed in the table. The markers can be moved with the cursor, or by clicking with MB1 (first marker) or with Ctrl+MB1 (second marker) in the curve window.

Snapshot

For some trends it is possible to take snapshot of the current trend, in order to examine it more carefully or save it to file. The snapshot is activated with the camera button in the trend toolbar, and a new curve window is opened with the snapshot.

Add curves

Curves can be added to the current trend window by selecting a DsTrendCurve object in the runtime navigator and activating the '+' button in the toolbar.

Graph grid

When several curves are present, the graph grid is adapted to the scale of a specific curve by clicking on the colored scale for this curve.

Export to text file

A trend can be exported to text file from File/Export in the menu. In the export dialog, a specific attribute can be selected for export, or all attributes. To avoid overwriting previous exported files, the filename can contain the string '$date' which will be replaces by the current date in the actual file name.

6.2 Trend snapshot

The trend snapshot curve is a snapshot of a trend taken at a specific time.

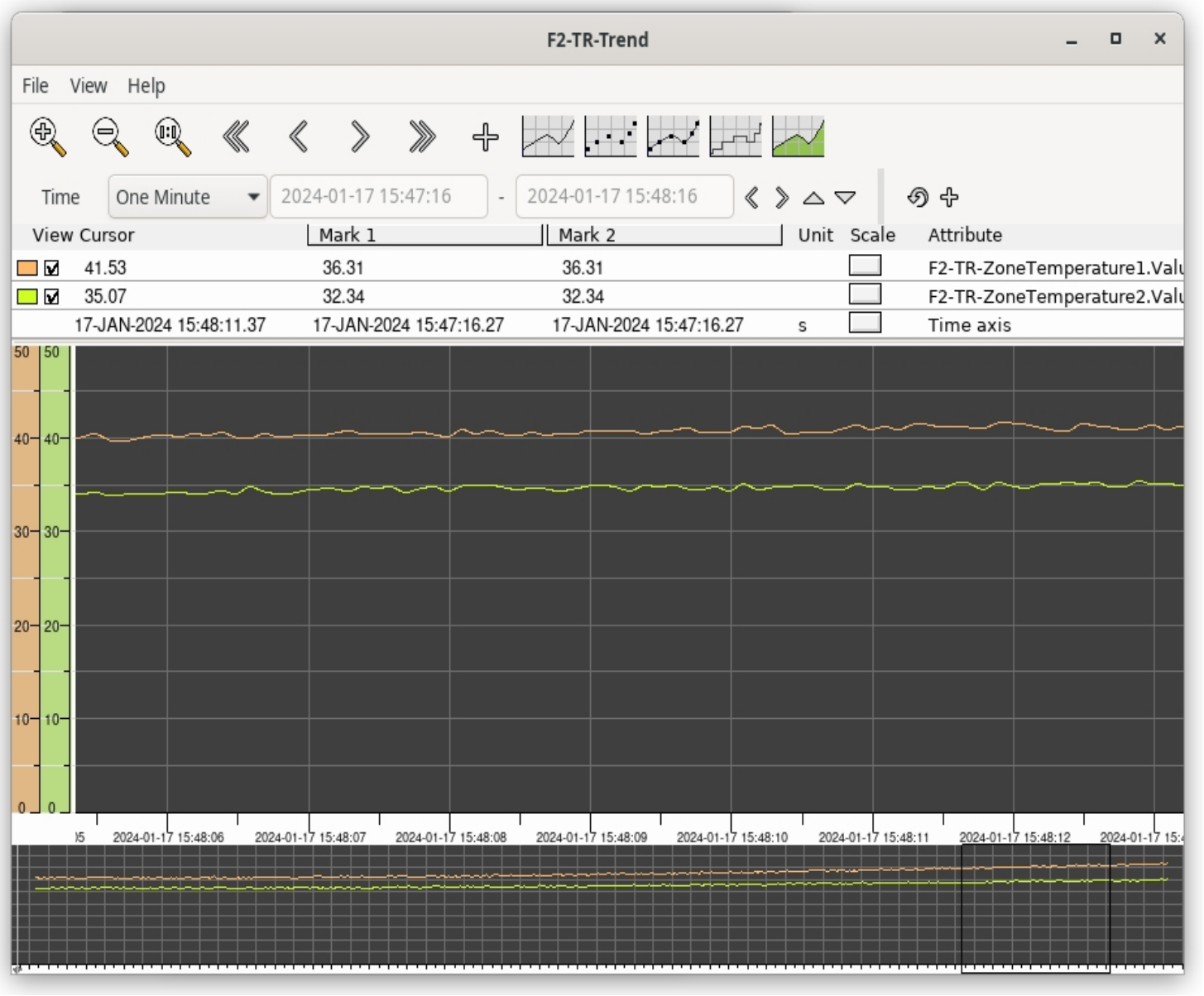


Fig Trend snapshot

A time interval can be selected in the time toolbar. When a new time intervall is selected the 'Update curve' button is pressed to redraw the curve.

Store curve

A snapshot curve can be stored by activating File/Save in the menu.

Open stored curve

A saved snapshot curve is opened from File/Open in the menu. To open in a new window, first activate File/New to open an empty snapshot window, and then activate File/Open in this window.

6.3 Fastcurves

Fastcurves are opened from the 'Fast' method in for example the popup menu. They can also be opened by the Xtt command 'open fast'. Fastcurves are configured by DsFastCurve objects and, if several curves are viewed in the same windows, by PlotGroup objects.

Fastcurves are used to display fast sequences during a shorter period of time. The storage of the fastcurve is started by a trigger signal and continues a specific time. When the sequence is finished the curve is viewed, and remains on the screen until the trigger signal is set again to activate a new recording of the sequence.

The fastcurves are viewed in a curve window. A curve window can display 20 different curves.

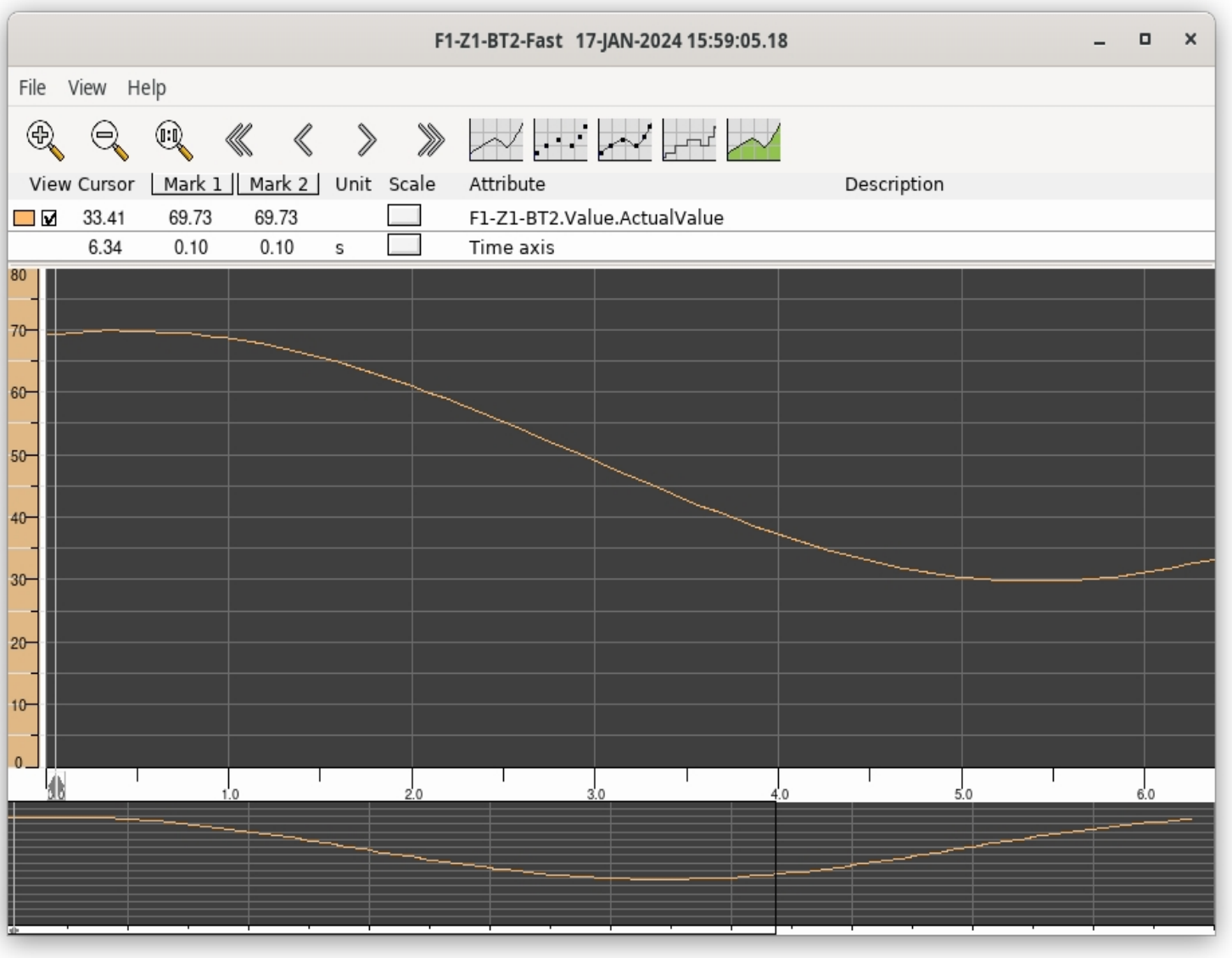


Fig Fastcurve

6.4 Process history

Process history is opened from the 'History' method in for example the popup menu. It can also be opened by the Xtt command 'open history'. The history is configured by SevHist

objects.

Process history means that the value of a signal, for example of type temperature, pressure or flow, is stored in a database with a specific frequency during a period of time.
The history can extend over several years and the number of values for one signal can reach several millions.

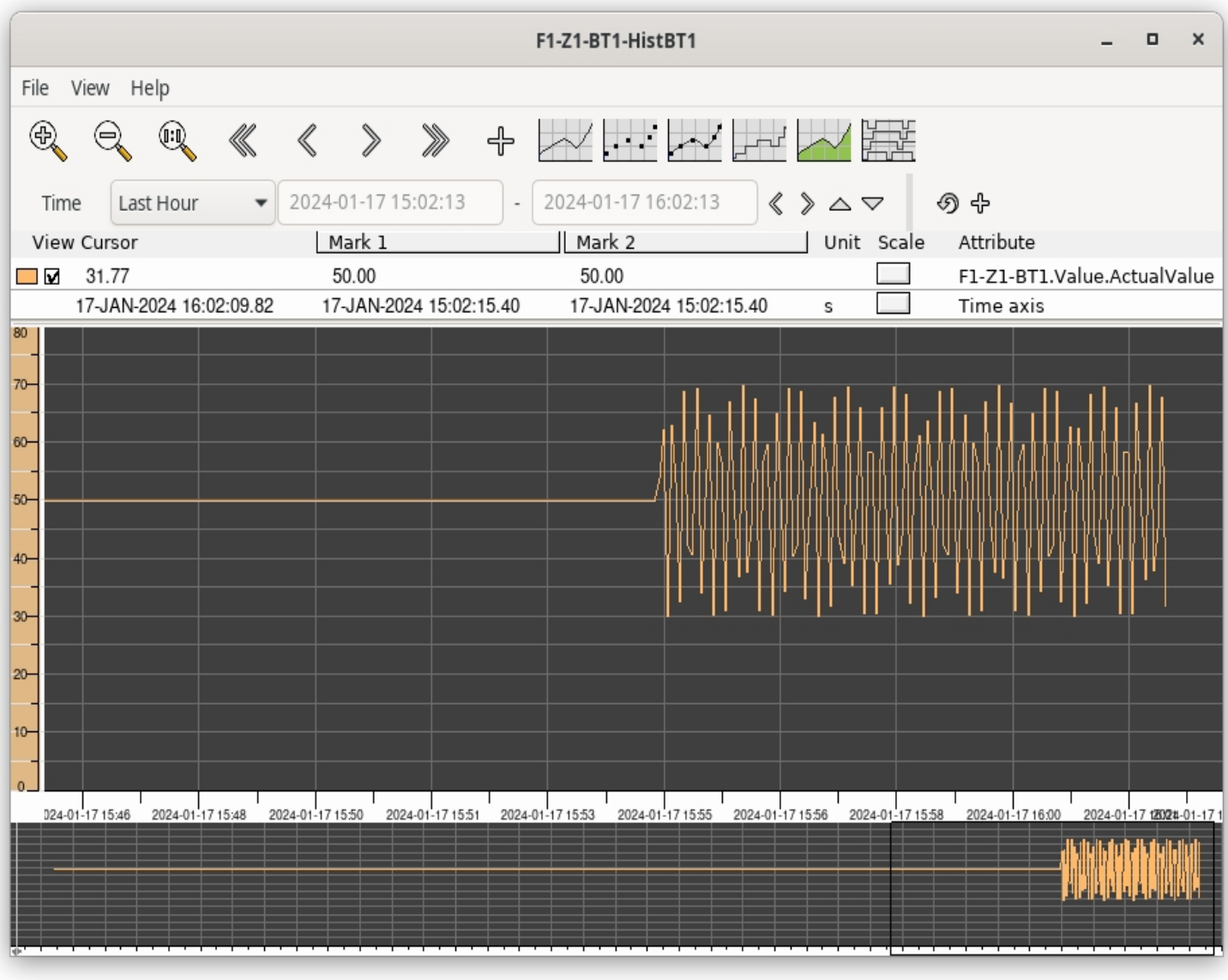


Fig Process history

In the upper part of the curve window views a table over the curves. The first column is a color indication to identify the corresponding curve.

Description of columns in the table

View	A checkbox from which you can choose to view or hide the curve.
Cursor	Shows the curve value where the cursor is placed.
Mark	Shows the curve value where the marker i placed.
Unit	Displays the unit of the curve.
Scale	From 'Scale' you can change the scale i y direction (Scale on the Time axis row changes i x direction).
Attribute	States the attribute of the trend curve.

Navigation

The tool panel contains buttons to zoom and move the view to the left or to the right.

You can also navigate with the navigation window in the lower part of the window. The part of the curve that is displayed, is marked with a black square. With drag MB1 you can move the view in x direction, and by dragging the middle button you can zoom in and out.

Navigation from the keyboard is accomplished with the arrow keys. With arrow up and down the zoom factor is changed, and arrow left and right moves the view to the left and to the right.

Resolution

As the number of values for an attribute can reach several millions, only a fraction of the values are viewed in the curve. When first opening the curve, the whole time range is viewed with a couple of hundred points. To gain higher resolution you use the arrow up and arrow down buttons in the tool panel. Arrow up will zoom in and increase the resolution, i.e. new values are fetched in a smaller time interval. Arrow down will zoom out and decrease the resolution.

7 Help window

Getting help

The help function can be opened in a number of different ways.

Operator window

In the middle part of the operator window there is a help button that opens the help texts for the project.

The Navigator menu

The menu entry Help/Overview gives help on ProviewR and the operator environment. Help/Project gives help on the project.

Command

A help text is opened by the Xtt command 'help'. To help you add the topic you are interested in. The command 'help help' shows for example how you use help.

Object methods

For an object there are two help methods.
- Help, views help for the object or plant part.
- Help Class, views information about the object class.

Help and info buttons in process graphs

Pushbuttons in process graphs can be connected to helptexts. These are usually marked with a question mark or an 'I'.

Fig Help buttons

Navigate in the help text

Rows with links are marked with an arrow in the left edge of the row.
The first row is a link to the previous topic.

Navigate from the keyboard

With PageUp and PageDown you can scroll up and down in the text.
Links are followed by selecting the link row with the arrow up and down keys, and then press arrow right.
Arrow left returns to the previous page.

Ctrl+N views the next topic, and Ctrl+P the previous topic. Ctrl+A returns to the start page.

Navigate with the mouse

To follow a link, click on the arrow, or double click on the row.
By clicking on the arrow in the first row, you return to the previous page.

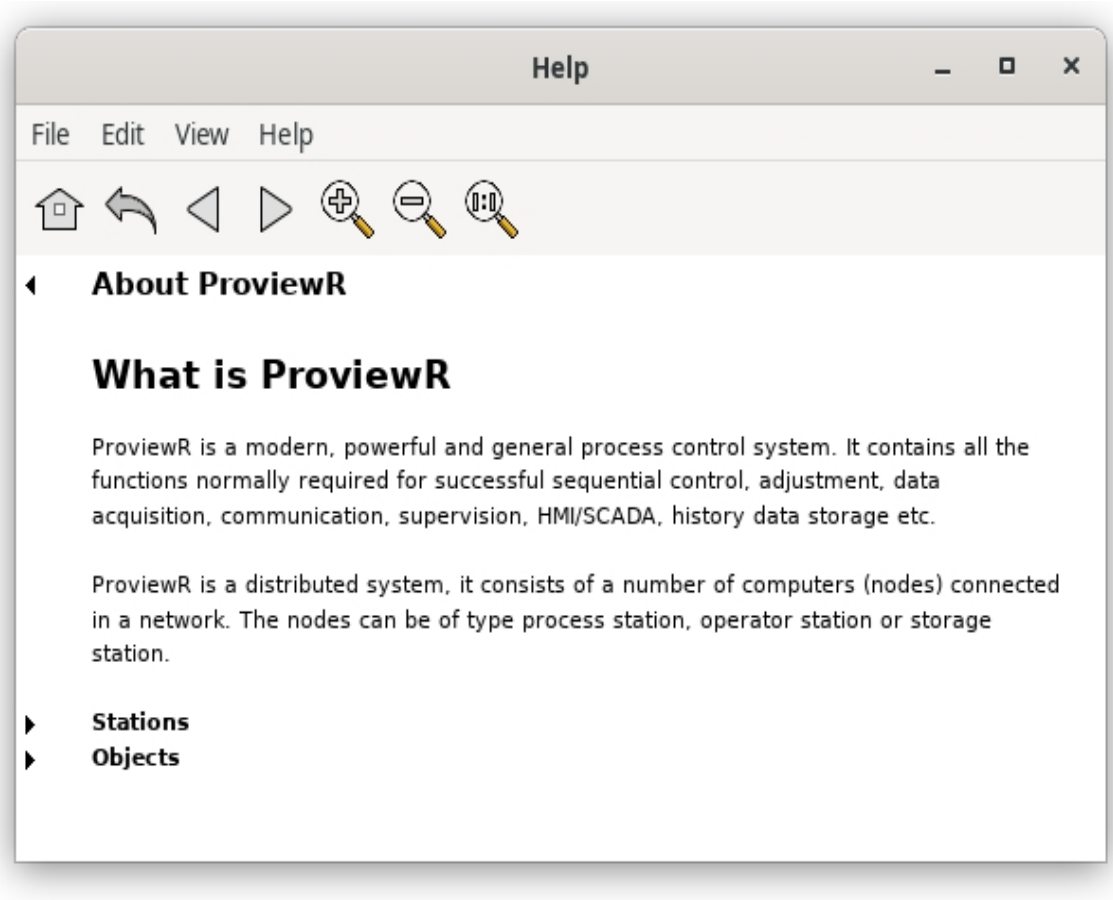


Fig Help window

8 Navigator

The navigator, also called Xtt, is started from the 'Navigator' button in the operator window. It can also be started as a separate program, independent of the operator environment. This is useful at maintenance or troubleshooting of the system. You then start from a terminal window, see section 3.1.

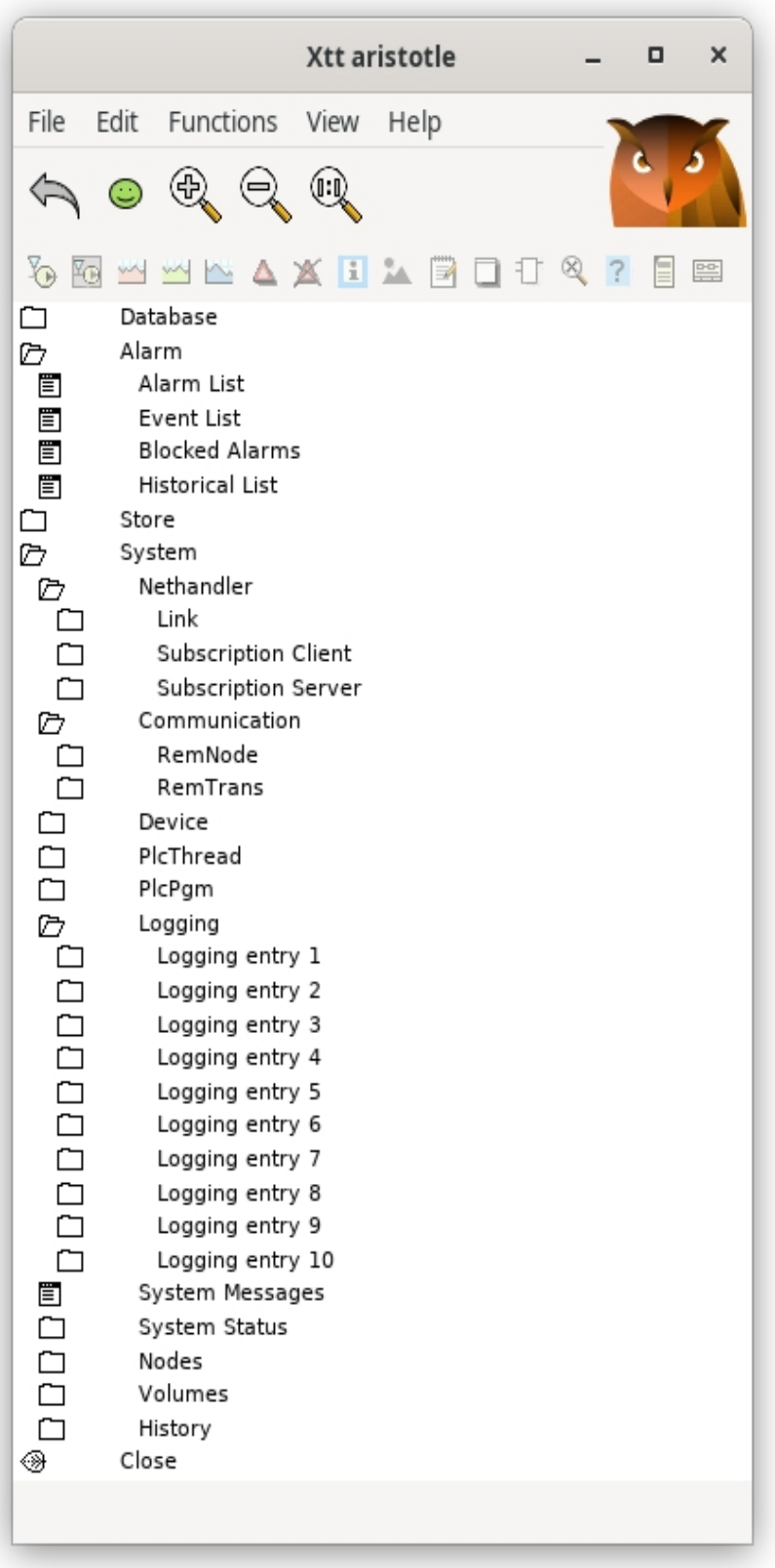


Fig The navigator menu

The figure above shows the navigator menu. The upper map, 'This Project' in configurable and can contain a map structure that is specific for the project, and by default it is missing. The other maps though are always present.

Database	Shows the object tree and the objects in the database.

Alarm / Alarm List	Opens the alarm list.
Alarm / Event List	Opens the event list.
Alarm / Blocked Alarms	Shows the list of blocked alarms.
Alarm / Historical List	Shows the event log.
Store	Shows stored collection views.
System / Nethandler / Link	Shows links to other ProviewR systems.
System / Nethandler / Subscription Client	Shows subscriptions (client).
System / Nethandler / Subscription Server	Shows subscriptions (server).
System / Communication / RemNode	Shows remote communication (nodes).
System / Communication / RemTrans	Shows remote communication (transactions).
System / Device	Shows I/O devices.
System / PlcThread	Shows Plc threads.
System / PlcPgm	Shows PlcPgm objects.
System / Logging	Dialog to start logging of data.
System / System Messages	Views system messages.
System / System Status	Views system status for the node.
System / Nodes	Shows the status graph for nodes.
System / Volumes	Shows loaded volumes.
Close	Close the navigator.

Navigate

Navigate from the keyboard

The most ergonomic way to navigate in the map structure is to use the arrow keys. With Arrow Up and Arrow Down you select a map or a leaf. With Arrow Right you open a map and with Arrow Left you close.

An object in the database is opened by Shift Arrow Right, i.e. you first press the Shift button and keep it down while pressing the Arrow Right key. If the object doesn't have any children, it is enough to press Arrow Left.

If you have the privilege RtWrite it is possible to change attribute values in the database. By opening an object, selecting an attribute and pressing Arrow Right, an input field is opened where a new value can be entered.

Navigate with the mouse

Of course you can also navigate with the mouse. A map is opened by clicking on the map (or doubleclicking on the text). Then you close by clicking on the map again.

An object in the database is opened by pressing the Shift key and click on the map/leaf for the object.

Commands

From Functions/Command in the menu, or Ctrl+B, a command prompt is opened in the bottom row of the navigator. Here you can enter commands and execute scripts. Available commands are described in the chapter Commands and script in the chapter Script.

Collection view

By Functions/Collect/Show, Ctrl+N, you look at collected objects and attributes. For trouble shooting you often need to look at a number of different attributes simultaneously. By traversing the object tree and collecting attributes with Functions/Collect/Insert, Ctrl+V, and then activating Functions/Collect/Show these attributes are viewed on one page.

A collection view can be saved to a later session by the 'save' command. To save a collection view with the name 'mycollection' you write the command

```
> store/collect mycollection
```

The name can be found in the 'Store' map, and can also be restored by the command

```
> @mycollection
```

Collection window

A collection view can be copied to a separate window with Functions/Collect/Copy To Window.

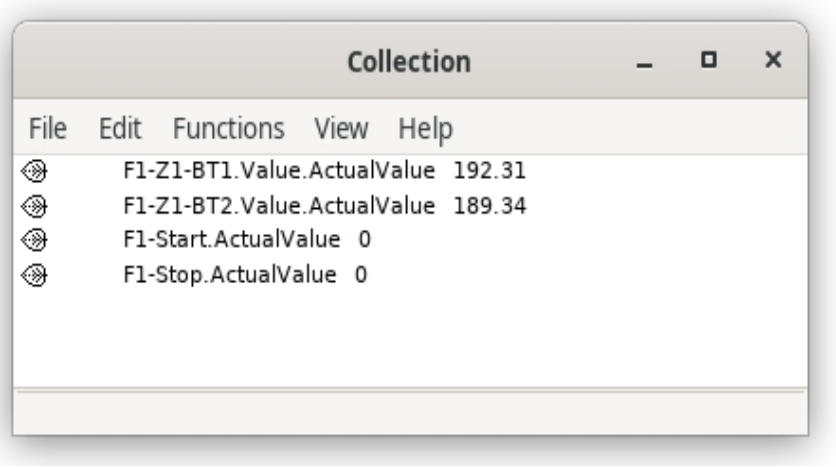


Fig Separate collection window

Also an empty window can be created from Functions/Collect/New Window. New attributes are added to the window by selecting the attribute in the runtime navigator, then selecting the collection window and activating Edit/Insert (Ctrl+V). With Shift+ArrowUp and Shift+ArrowDown attributes can be moved up and down in the window.

A collection window is saved from File/Save (Ctrl+S) with a specified name. It can later be opened from Functions/Collect/Open in the runtime navigator, or File/Open in an collection window. The desired window is opened from a list containing all saved collection windows.

8.1 Database

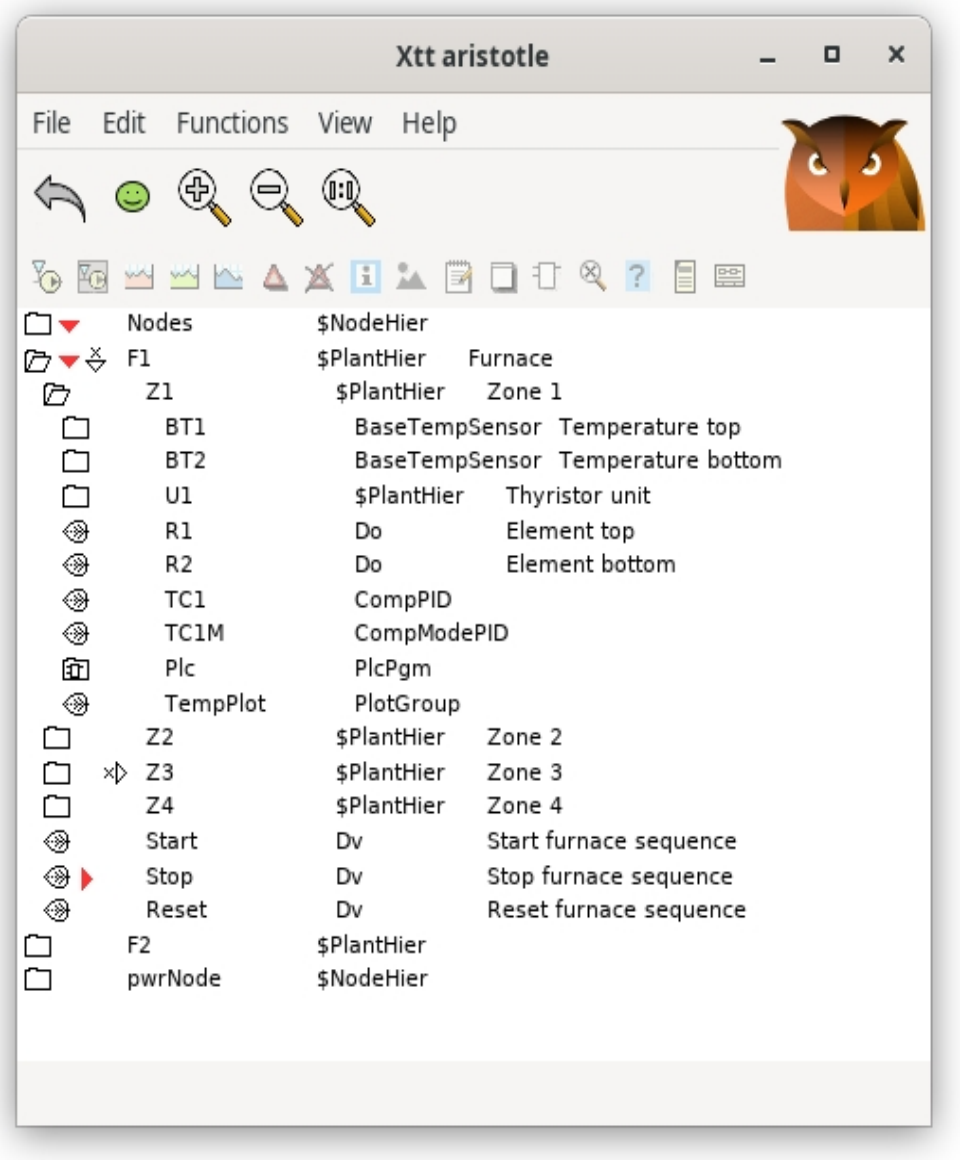


Fig The object tree from the navigator

Under the map 'Database' the object tree is shown. Here you can navigate in the object three, open objects and examine the content of different attributes in the object. If you have privileges, you can also change attribute values.

If you rightclick on an object, an popup menu with the object methods is opened. Buttons to activate the methods of the objects are also found in the lower tool panel in the navigator. Only methods that are valid for an object are sensitive.

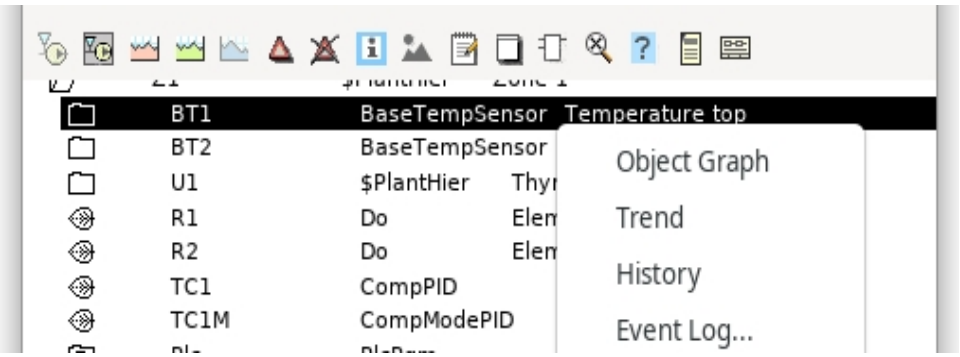


Fig Methods can be activated both from the tool panel and the popup menu

8.2 System / Nethandler / Link

The map 'System / Nethandler / Link' shows links to other ProviewR stations. The current station is also viewed in the list.

In the figure below you can see that the local node aristotle has contact with the node pwrutv.

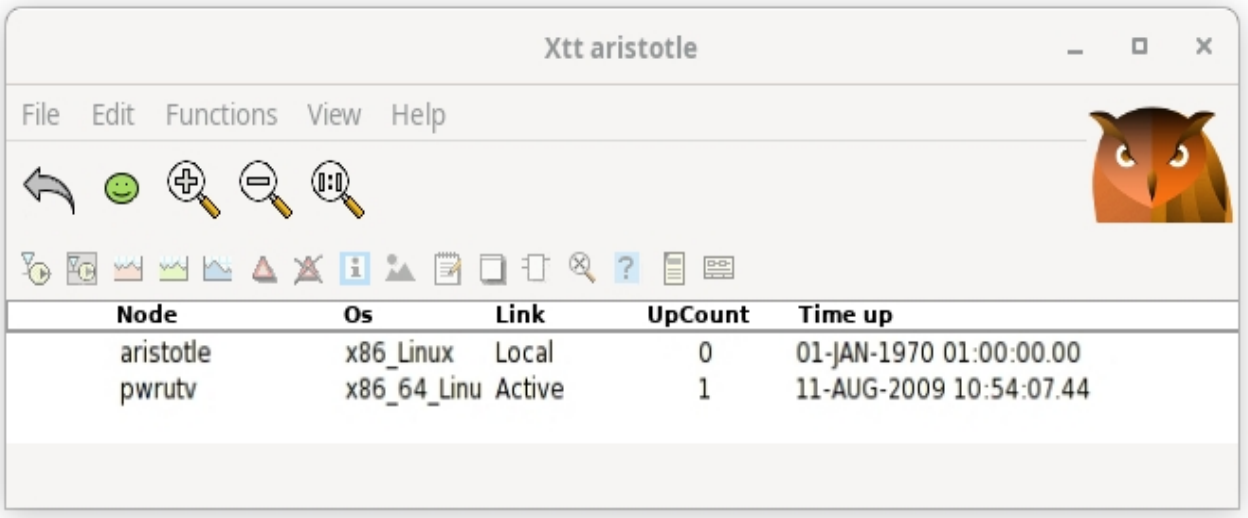


Fig Links to other nodes

Description of the columns in the table

Node	Nodename
Os	Operating system and hardware for the node.
Link	Link status. Can be Up, Active, Connected, Down or Local.
UpCount	The number of times the link is established.
TimeUp	Time for the last establishment of the link.

8.3 System / Nethandler / Subscription Client

Shows the subscriptions the current node has to other stations.

When an operator station displays data from process stations in an process graph, a subscription on each data viewed in the graph is created. This implies that the process stations at regular intervals sends new data to the operator stations, which updates the process graph with the new data.

If there is a field in a process graph that is not updated. you can enter the Subscription Client table to see if the subscription is present and if it is updated. Each time new data arrives for a subscription, 'Count' increments and the time is inserted in 'Time'.

In 'Unknown' on the top row, you can see the number of subscriptions where the attribute is unknown. This can be a subscription to a station that is not yet running, but it can also be a subscription that is misspelled, or to objects that no longer exist.

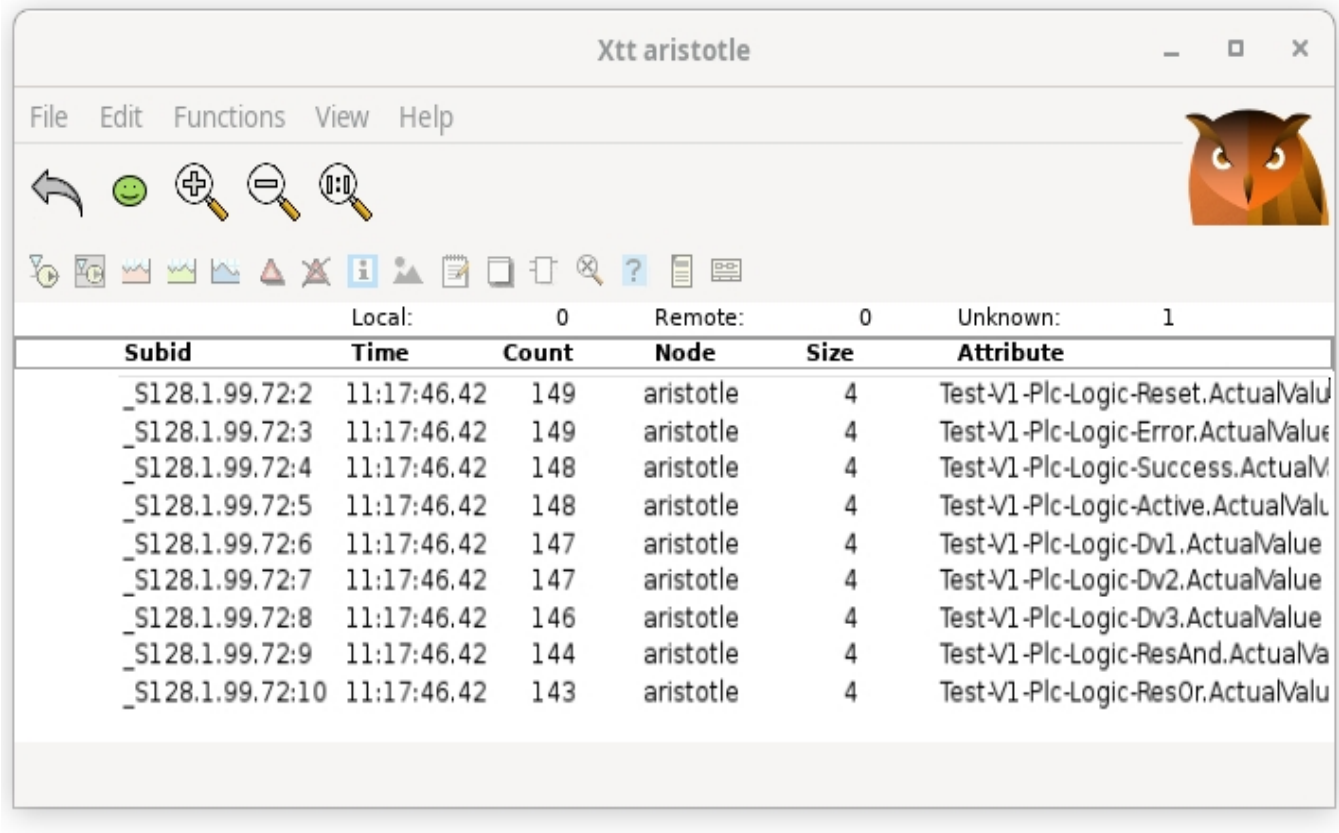


Fig Subscription client

Description of the columns

Subid	The subscription identity.
Time	Latest arrival of subscription data.
Count	Number of times data has been received.
Node	Node sending subscription data.
Size	Size in bytes of subscription data.
Attribute	Database attribute for the subscription.

8.4 System / Nethandler / Subscription Server

The list shows subscriptions other stations have requested from the current node, i.e. subscription sent cyclically from the current node to other nodes, usually from process stations to operator stations.

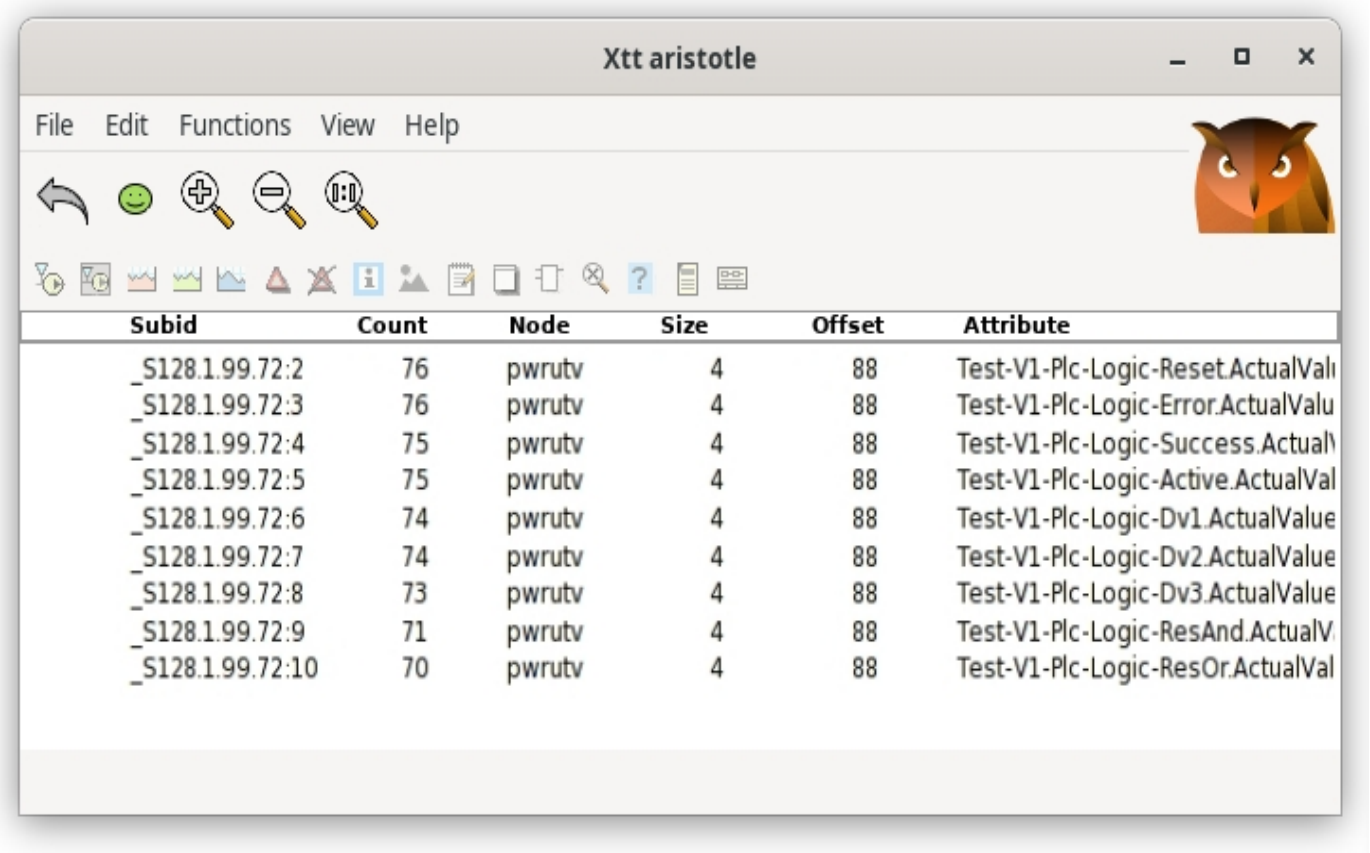


Fig Subscription server

Description of the columns

Subid	The subscription identity.
Count	Number of times data has been sent.
Node	Node to whom subscription data is sent.
Size	Size in bytes of subscription data.
Offset	Offset in the object from where data is fetched.
Attribute	Database attribute for the subscription.

8.5 System / Communication / RemNode

Displays a list of nodes that the current node communicates with via Remote.

Remote is a function to send data between nodes. Nodes communicating with each other are configured by RemNode objects, and a transaction sent between two nodes by the Remote function are configured by RemTrans objects. A number of different protocols for the data exchange are supported, Modbus, MQ, Serial, TCP, UDP, 3964R and ALCM.

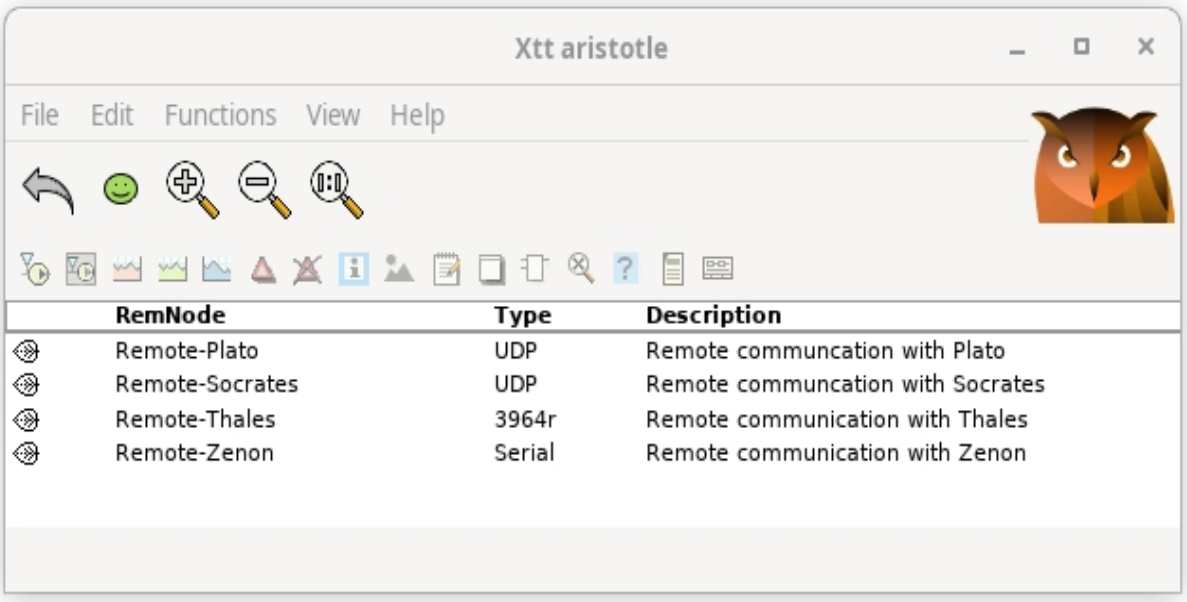


Fig List of RemNodes

Description of the columns

RemNode	Name of the RemNode object, only the two last name segments are displayed.
Type	Type of communication, Modbus, MQ, Serial, TCP, UDP, 3964R or ALCM.
Description	Optional description.

By selecting a RemNode and pressing the Arrow Right key, the RemTrans objects for the node are viewed (you can also click on the map).
By Shift/Arrow Right the RemNode object is opened.

8.6 System / Communication / RemTrans

Shows a list of all RemTrans objects.

A RemTrans object specifies a transaction sent between two node by the Remote function. The transaction implies that the current node receives or sends data.

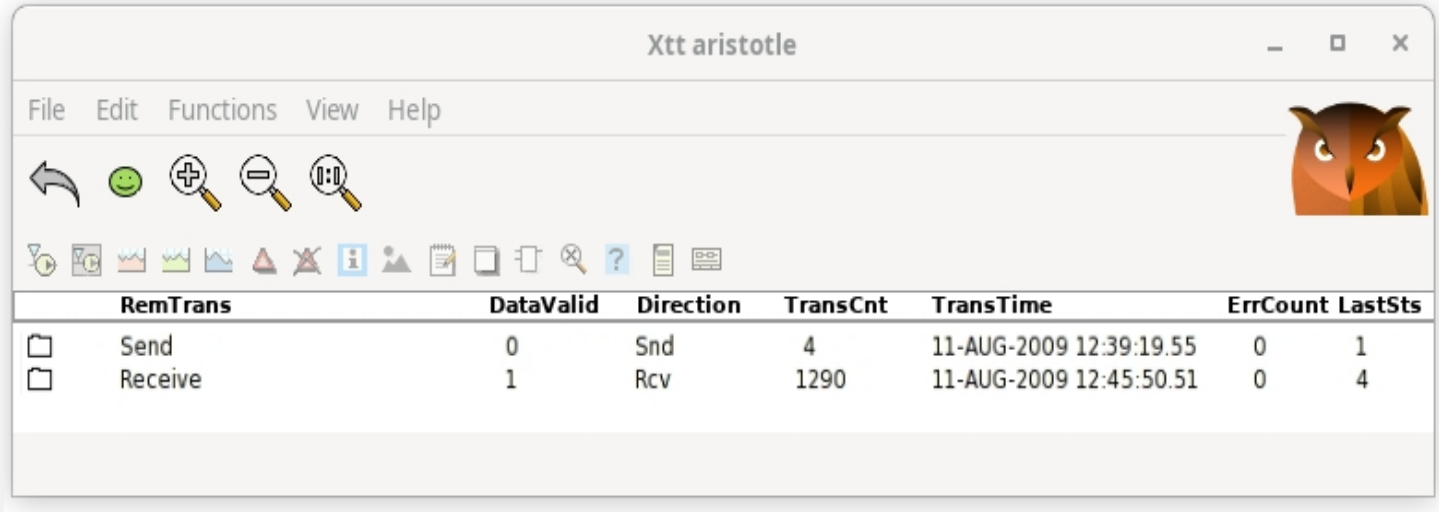


Fig List of RemTrans-objecs

Description of the columns

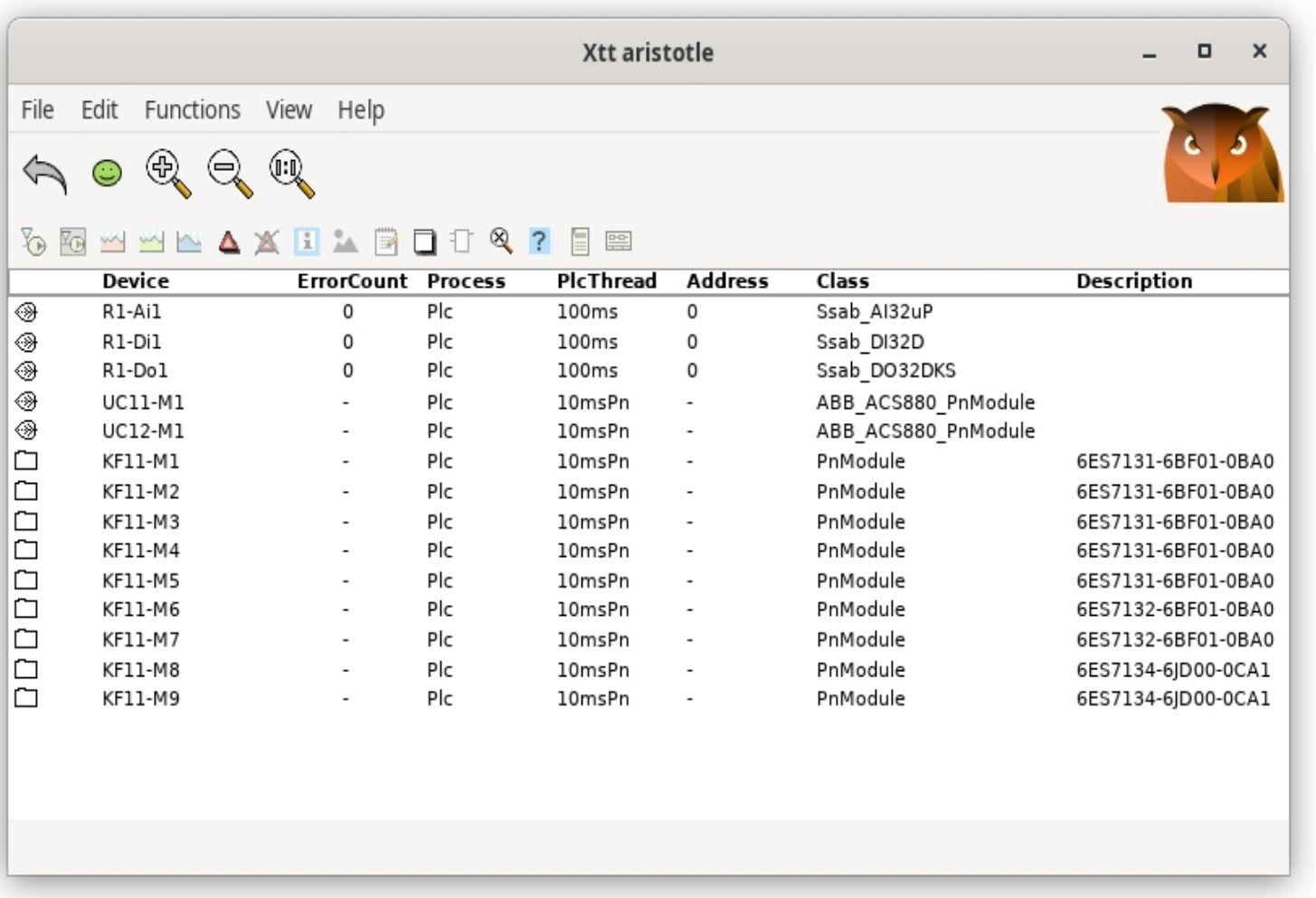
RemTrans	Name of the RemTrans object, only the two last name segments are shown.
DataValid	States that data that is not yet sent is present (when sending) or that received data that has not been taken care of is present.
Direction	Direction of the transaction, Snd (send) or Rcv (receive).
TransCnt	Number of sent or received transactions.
TransTime	Time for the last transaction.
ErrCount	Number of unsuccessful transactions.
LastSts	Status for the last transaction.

By selecting a transaction and pressing the Arrow Right key, the content of the transaction is viewed (you can also click on the map). If the attribute StructFile is supplied with the include file containing the data structures for the transactions, the data content is viewed in this format.
By Shift/Arrow Right the RemTrans object is opened.

8.7 System / Device

Shows a list of I/O devices.

The I/O system in ProviewR is divided into four levels, agent, rack, card and channel.
In the device list, all the card objects are displayed.



	Device	ErrorCount	Process	PlcThread	Address	Class	Description
	R1-Ai1	0	Plc	100ms	0	Ssab_AI32uP	
	R1-Di1	0	Plc	100ms	0	Ssab_DI32D	
	R1-Do1	0	Plc	100ms	0	Ssab_DO32DKS	
	UC11-M1	-	Plc	10msPn	-	ABB_ACS880_PnModule	
	UC12-M1	-	Plc	10msPn	-	ABB_ACS880_PnModule	
	KF11-M1	-	Plc	10msPn	-	PnModule	6ES7131-6BF01-0BA0
	KF11-M2	-	Plc	10msPn	-	PnModule	6ES7131-6BF01-0BA0
	KF11-M3	-	Plc	10msPn	-	PnModule	6ES7131-6BF01-0BA0
	KF11-M4	-	Plc	10msPn	-	PnModule	6ES7131-6BF01-0BA0
	KF11-M5	-	Plc	10msPn	-	PnModule	6ES7131-6BF01-0BA0
	KF11-M6	-	Plc	10msPn	-	PnModule	6ES7132-6BF01-0BA0
	KF11-M7	-	Plc	10msPn	-	PnModule	6ES7132-6BF01-0BA0
	KF11-M8	-	Plc	10msPn	-	PnModule	6ES7134-6JD00-0CA1
	KF11-M9	-	Plc	10msPn	-	PnModule	6ES7134-6JD00-0CA1

Fig List of I/O units

Description of the columns

Device	Name of device object. Only the two last name segments are viewed.
Class	Card object class.
ErrorCount	Counter for read or write failures.
Process	Process handling the card. 1: plc, 2 io_comm.
PlcThread	Plc thread handling the card.
Address	Address configured for the card.

By selecting a device and press the Arrow Right key the channels of the device are displayed (you can also click on the leaf).

8.8 System / PlcThread

Displays a list of threads in the plc program.

Different plc programs are running with different priority and cycletime. They are connected to PlcThread objects that determines the priority and cycletime for the execution.

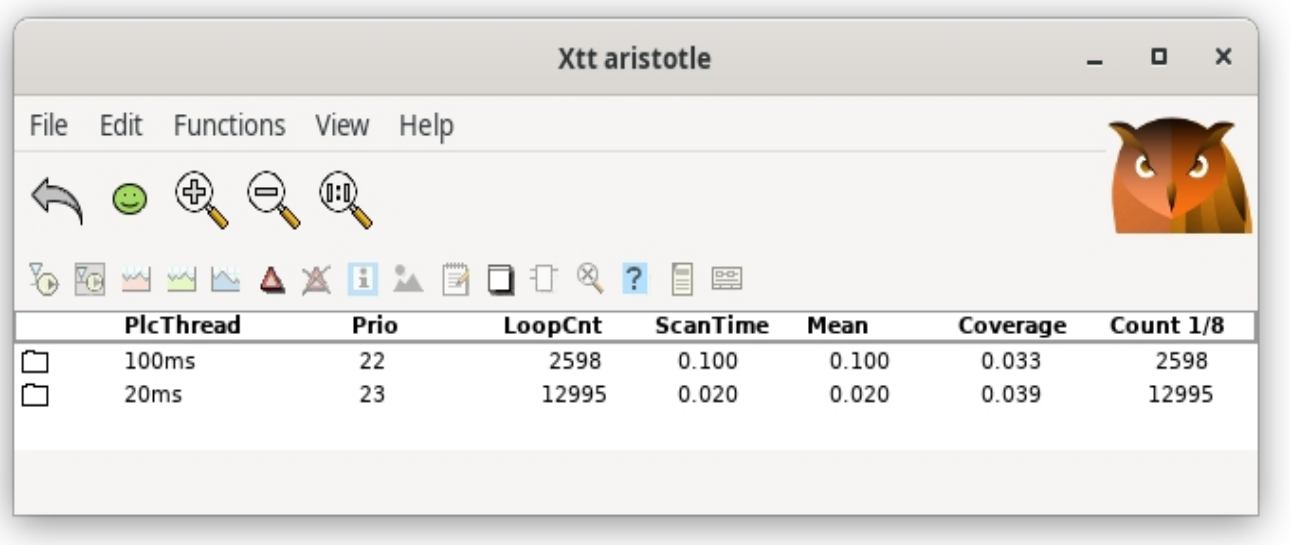


Fig List of plc threads.

Description of the columns

PlcThread	Name of thread object.
Prio	Thread priority.
LoopCnt	Counter of number of cycles since ProviewR startup.
ScanTime	Configured cycletime.
Mean	Meanvalue of measured cycletime.
Coverage	The execution time for the thread in proportion to the cycletime in percent.
Count 1/8	Number of cycles where the execution time is less than 1/8 of the cycle time.
Count 1/4	Number of cycles where the execution time is less than 1/4 of the cycle time and larger than 1/8.
Count 1/2	Number of cycles where the execution time is less than 1/2 of the cycle

time and larger than 1/4.

Count 1/1 — Number of cycles where the execution time is less than the cycle time and larger than half the cycletime.

8.9 System / PlcPgm

Displays a list of all PlcPgm objects.

A PlcPgm object contains plc code controlling a plant part, or executing other types of calculations. The PlcPgm object is connected to a thread object specifying the priority and cycletime.

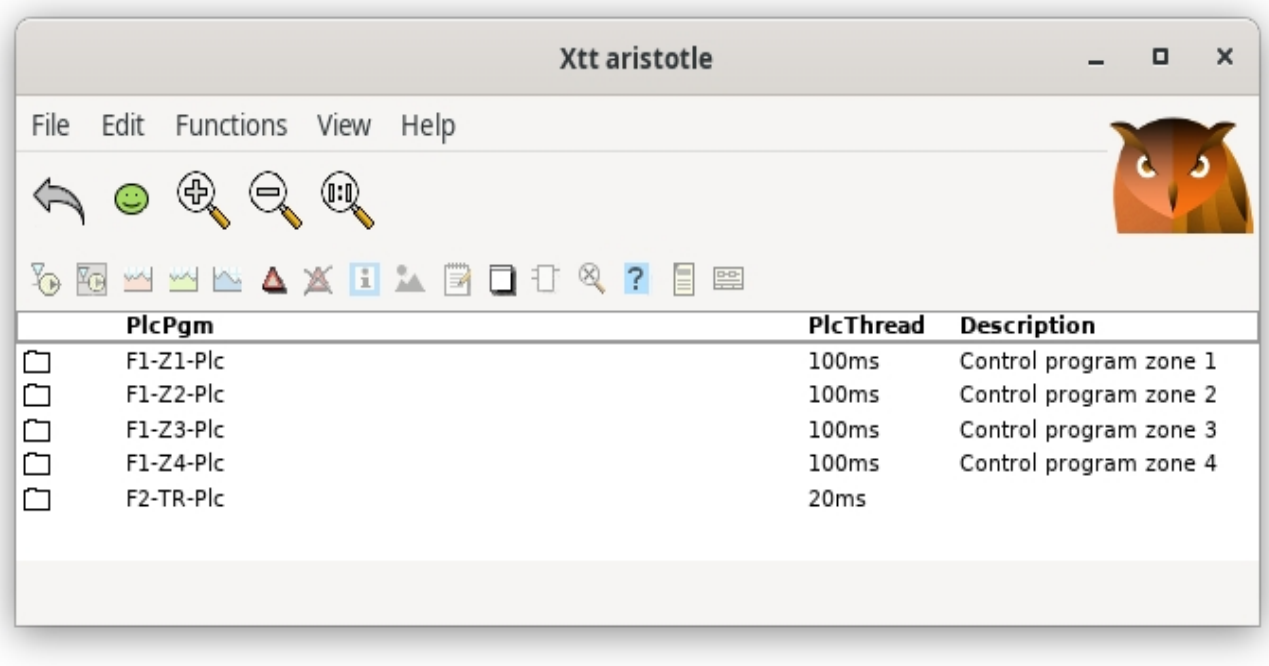


Fig List of PlcPgm objects

Description of the columns

PlcPgm	Name of PlcPgm object.
PlcThread	Plc thread the object is connected to.
Description	Optional description.

PlcTrace for a PlcPgm is opened by selecting the PlcPgm object and press Ctrl+L. Arrow Right key will display all the windows in the plc program.

8.10 System / Logging

With the logging function it is possible to log signals and values in the database to a text file. The content of the textfile can be viewed in a curve window, or imported into for example Calc or Excel.

Maximum 100 attributes can be logged in one logfile, and there are 10 entries, where each entry handles one logfile.

Start the logging

- Collect the attributes that are to be logged, into the collection view, for example by finding and selecting the attributes in the database and press Ctrl+V.

- Check with Ctrl+N that the correct attributes are present in the collection view.

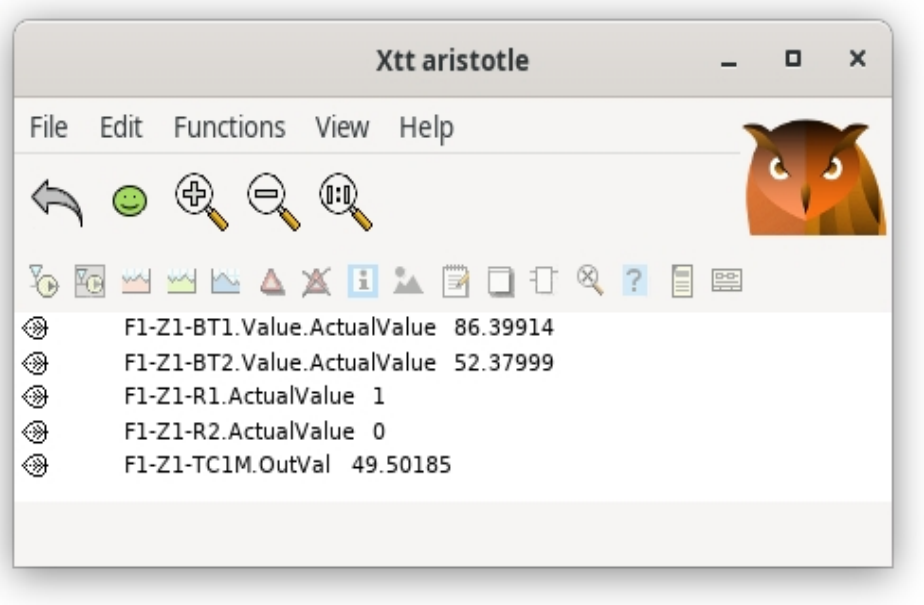


Fig Attributes collected in the collection view

- Open Logging entry 1 (System / Logging / Logging Entry 1).

- Click on the 'Insert' button, The attributes are now transferred from the collection view to the list of attributes in the logging entry.

- Specify cycle time in ms, and filename if needed.

- Start the logging by clicking on the 'Start' button.

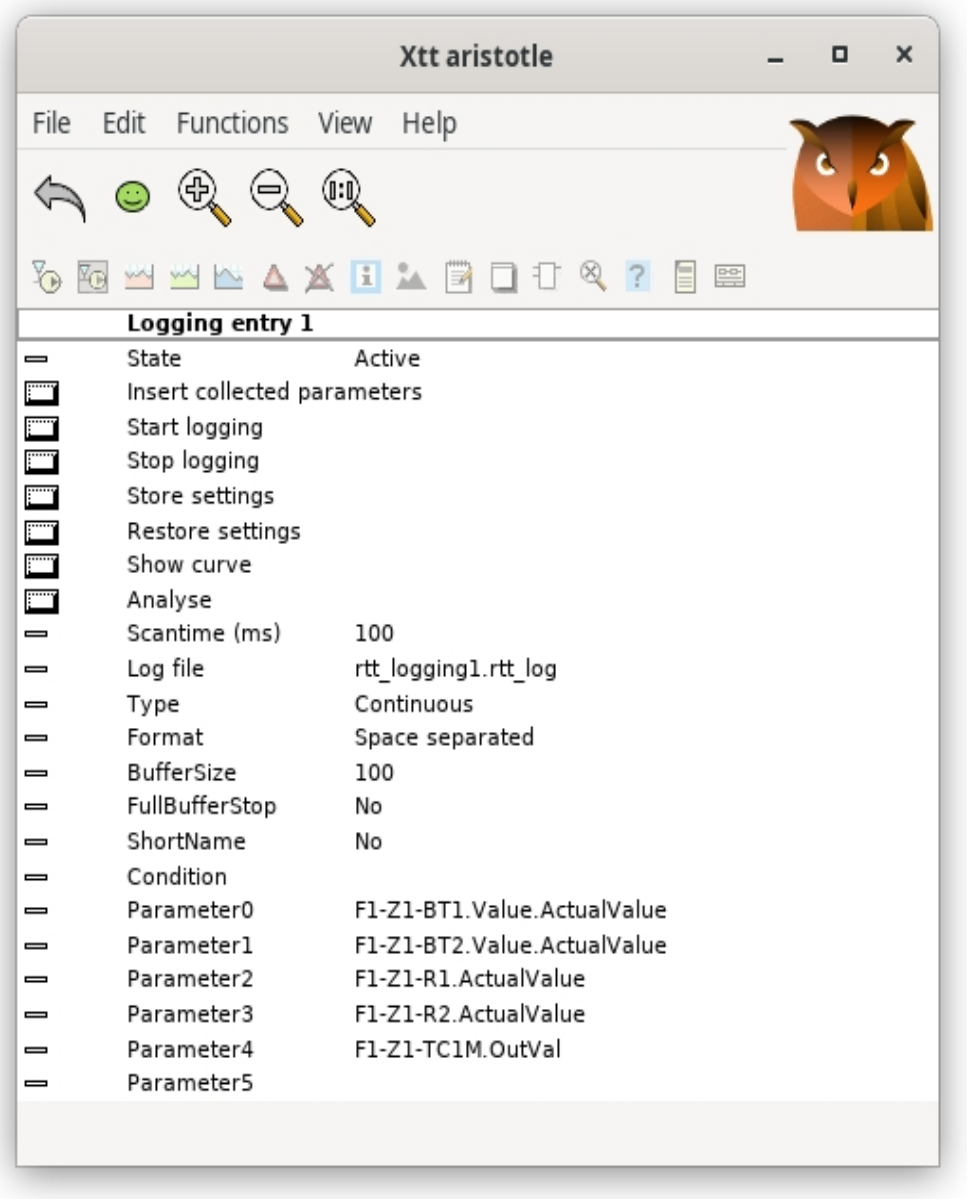


Fig Logging entry with logging started

- When the logging is complete, you click on the 'Stop' button. Note that you must not terminate the navigator during the period of time the logging is active. Though you can leave the loggentry page.

- Look at the logging result in the curve window by activating 'ShowFile'.

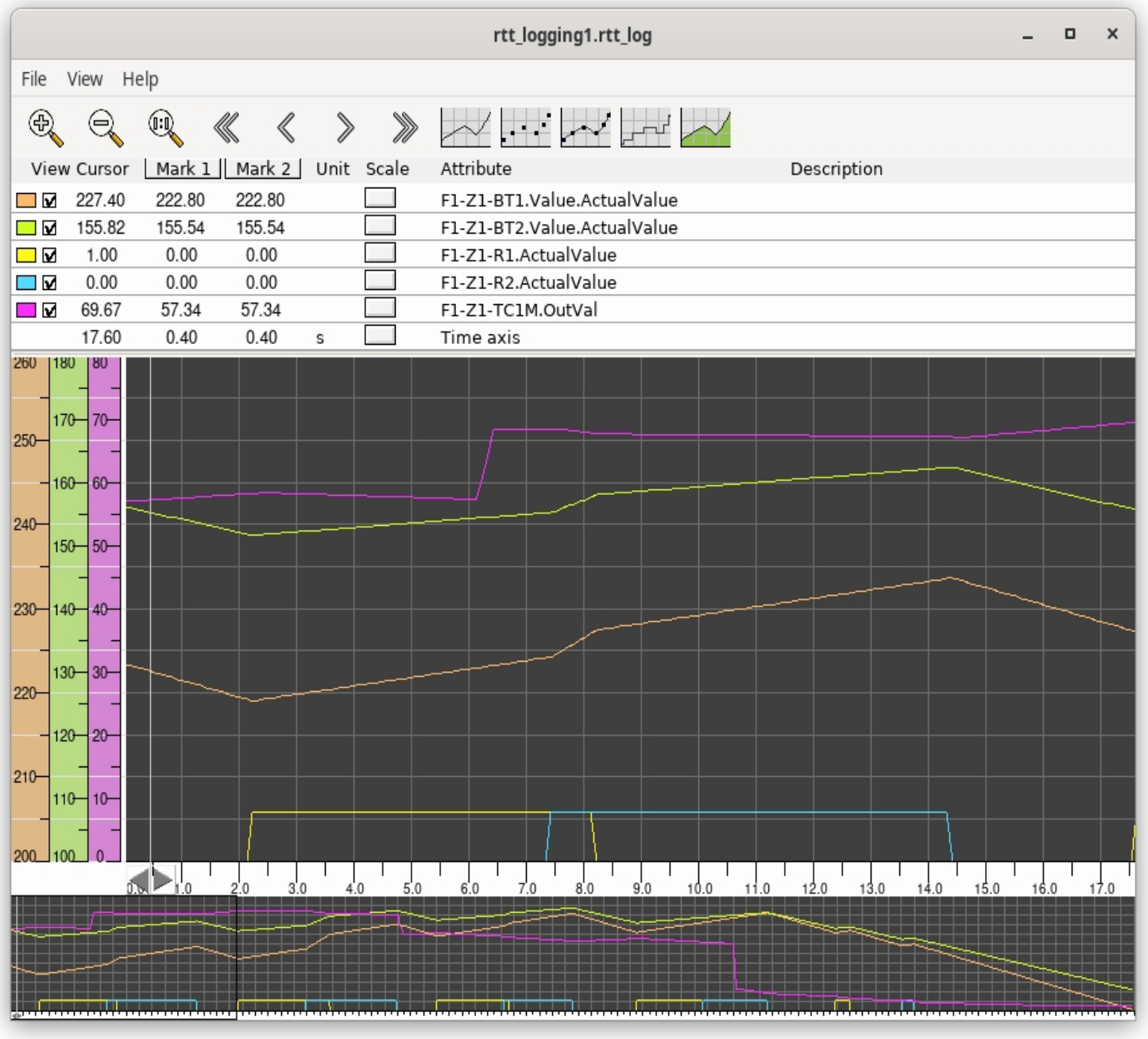


Fig Logging viewed as a curve

There are two types of logging. Either the value of the attributes are logged continuously with a certain frequency (continuous logging), or an attribute is logged every time its value is changed (event triggered logging).

Continuous logging

The values of the attributes in the entry are logged every cycle. In the file, also the time since the logging started is written. The file can be viewed in the Xtt curve window from 'ShowFile' or imported into a spreadsheet program.

Event triggered logging

An attribute in the entry is logged if the value of the attribute is changed. Into the logfile the time of the change, and the new value is written. This type of logging can

not be viewed from 'ShowFile'.

Buffer

The logging is first written into an intern buffer. When the buffer is full it is written to file. At fast time critical sequences, you can increase the buffer size to avoid interruption when the buffer i written to file.

Conditional logging

In 'Condition' a digital attribute controlling the logging can be specified. The logging is only performed when the ConditionParameter is true.

It is also possible to insert an expression into 'Condition'. The expression starts with 'expr' followed by the condition enclosed with parenthesis. In the expression, attribute values can be fetched with the GetA function for analog attributes, GetD function for digital attributes and GetI function for integer attribute.

Expression example
expr(GetD("H1-H2-Dv1.ActualValue") && GetA("H1-H2-Av2.ActualValue") > 2.5)

The logging is performed when the Dv 'H1-H2-Dv1' is true and the Av 'H1-H2-Av2' is larger than 2.5.

Store

The logging configuration is stored with the 'Store' button, and can be restored later with the 'Restore' button.

8.11 System / System Messages

The ProviewR system logg contains information, warning and error messages, written by the server processes of the system. Also applications can write to the system log by using the errh interface.

Messages in the loggfile has different degrees of severity. The severity is marked with a letter and color:

F	red	Fatal, the most serious type of error message.
E	red	Error.
W	yellow	Warning.
I	green	Information.
S	green	Success.

Also the ProviewR startup is marked with a yellow colored row.

Above all you should notice red messages implying something is wrong in the system.

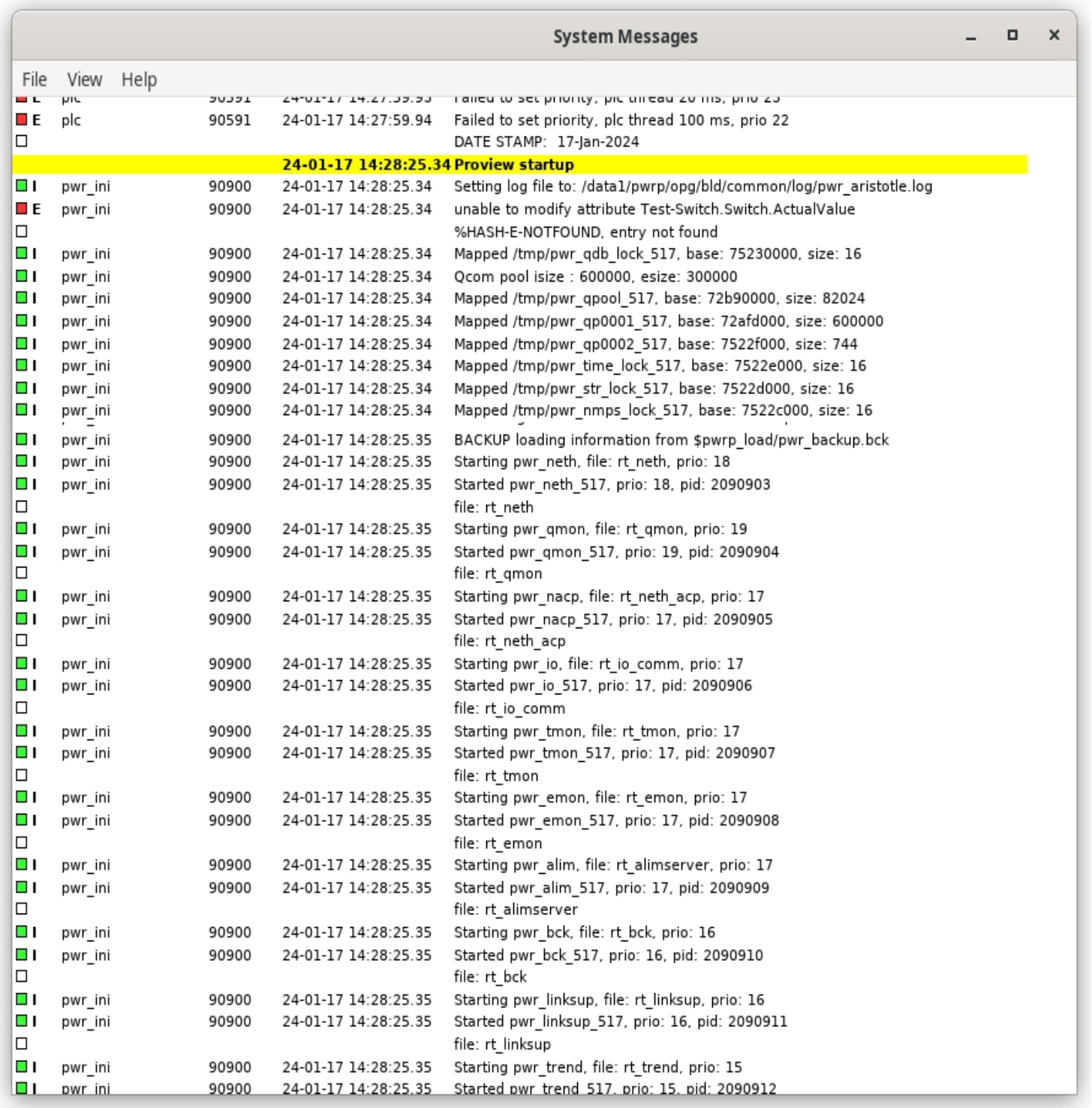


Fig System log

8.12 System / System Status

The system status displays a list of nodes in the system, and by opening a node the status graph for the node is shown.

The most interesting in the graph is the SystemStatus. This is an indication of the

condition of the system. If system status is green, its OK, if it is yellow or red, something is wrong in the system.

The color markings has the same severity as in the System log above, i.e.

F	flashing red	Fatal.
E	red	Error.
W	yellow	Warning.
I	green	Information.
S	green	Success.
	gray	Null status. The function is not activated.

The system status is a summary status for the system and application processes. Below system status there is a list of all these processes, and the status for each process is displayed. Furthermore is the latest or most severe system message in the system log from the processes are displayed.

If the system status is for example yellow, this implies that some system or application process has a yellow status. Usually by reading the status text and look in the system log is is possible to figure out what the error is.

rt_ini	Initialization	Startup process starting the system. Also handles the system log.
rt_qmon	QCom monitor	Handles the communication with other nodes.
rt_neth	Nethandler	Handles exchanged of database information between nodes.
rt_neth_acp		Utility process for the nethandler.
rt_io	I/O	Process for I/O handling that is asynchronous with the plc program
rt_tmon	Timer monitor	Handles sending of subscriptions.
rt_emon	Event monitor	Handles alarms and events.
rt_bck	Backup process	Handles backup of attributes and objects.
rt_linksup	Link supervision	Supervision of links to other nodes.
rt_trend	Trend process	Handles storing of trend curves.
rt_fast	Fast process	Handles storing of fast curves.
rt_elog	Event log process	Logging of events.
rt_webmon	Web monitor	Supplies the web interface with database information.
rt_webmonmh	Web mh monitor	Supplies the web interface with alarms and events.
rt_sysmon	System monitor	Supervises the system.
plc	Plc process	Handles I/O and executes the plc code.
rs_remote	Remote process	Handles remote communication.
opc_server	Opc server process	Handles Opc server communication.
rt_statussrv	Status server	Webservice for the runtime monitor and supervision center.
rt_post	Post server	Sends alarms via email or sms.
rt_report	Report server	Generates reports.
rt_sevhistmon	Storage monitor	Collects and sends process history data to storage server.
rt_powerlink	Ethernet powerlink server	Handles Powerlink communication.

8.13 System / Volumes

Displays the loaded volumes.
For each volume is viewed: name, class, version and an letter combination LCM where L means loaded, C cached and M mounted.

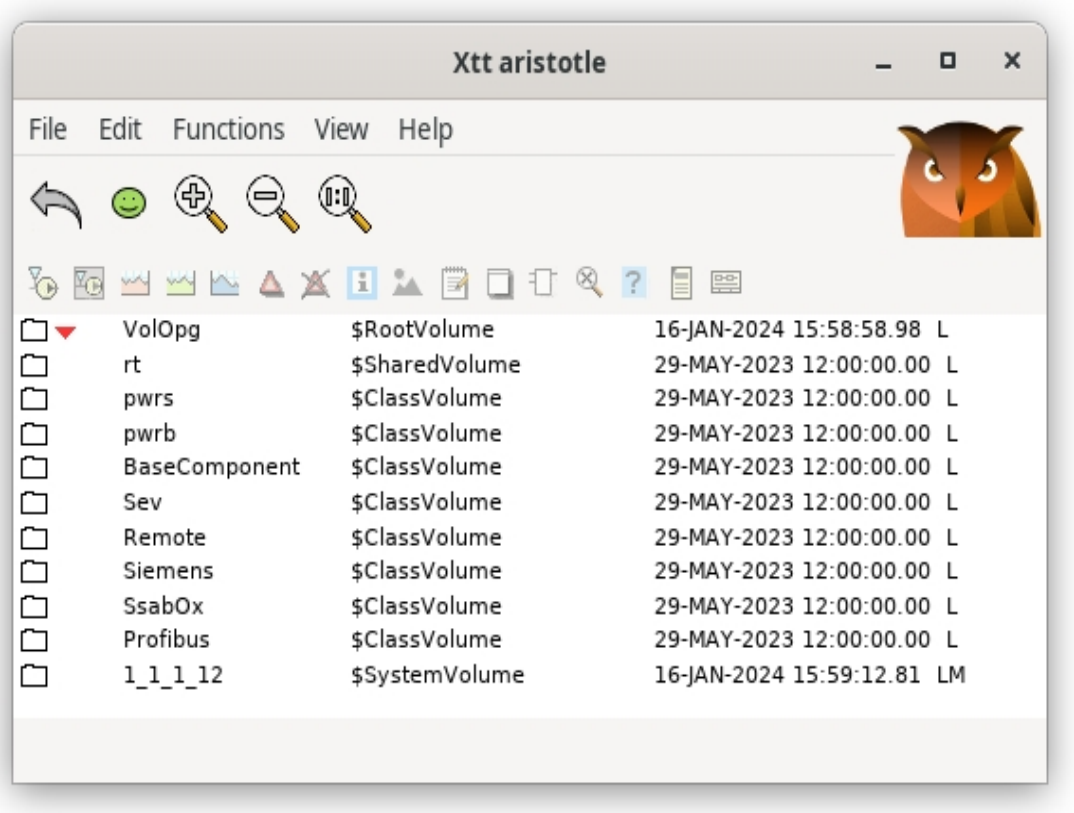


Fig Loaded volumes

9 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 dashboard is divided in rows and columns where the cells are positioned. A cell initially has the height of one row and the width of one column, but can be expanded over several rows and columns.

The cell can contain several elements. A bar cell for example, can contain three elements. Each element is connected to an analog attribute that are displayed with a bar. A trend cell can contain two elements and show two curves.

There are three major cell types, analog, digital and object.

Analog dashboard cells

Bar

A bar cell displays an analog value with a bar and a value. I can contain three elements.

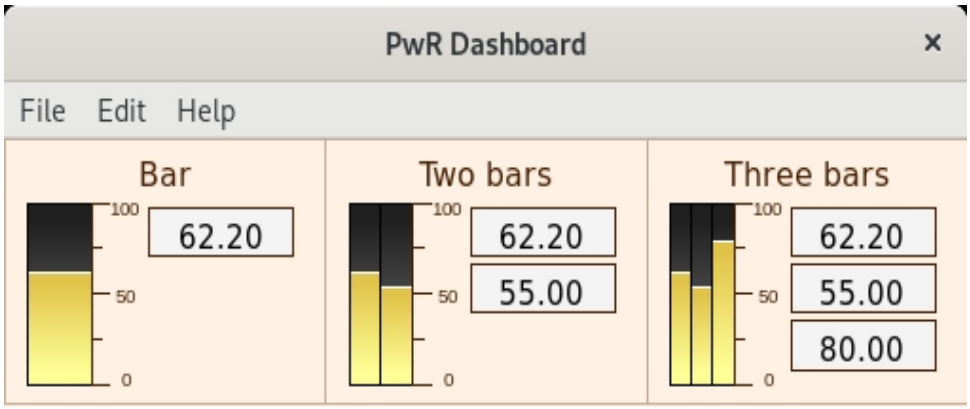
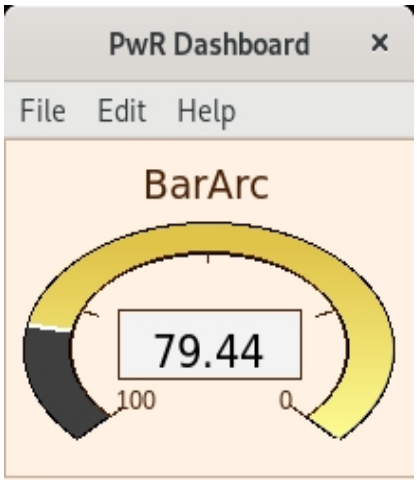
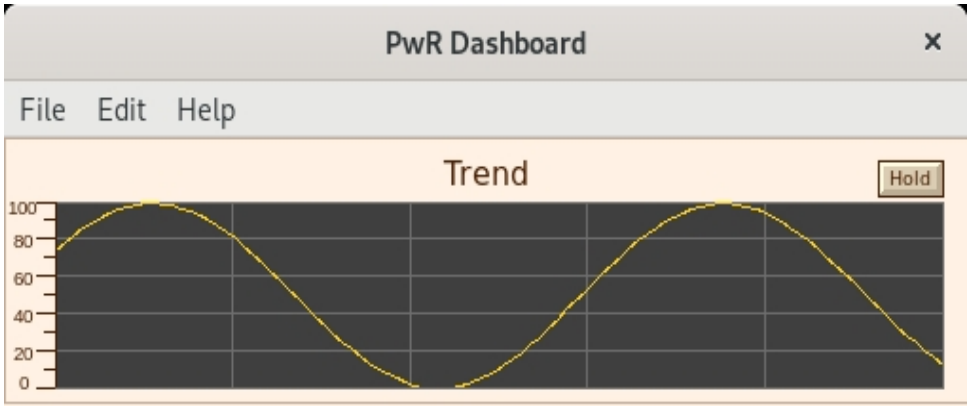


Fig Bar cells with one, two and three elements

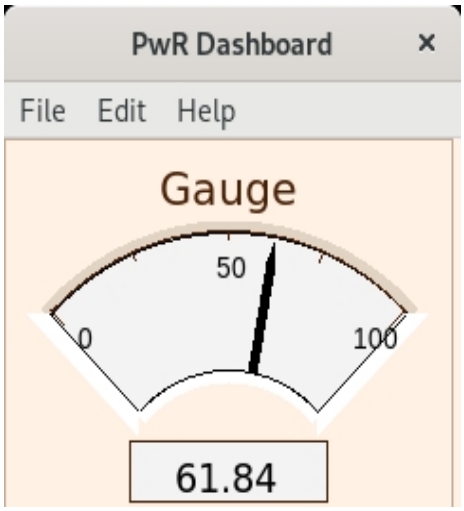
BarArc



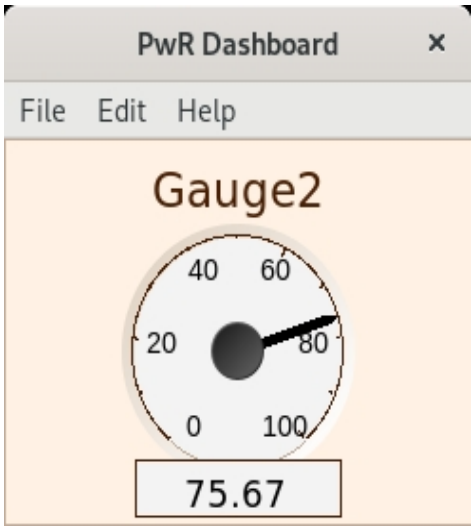
Trend



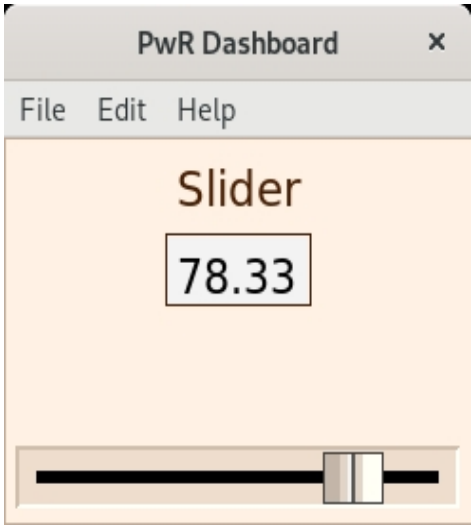
Gauge



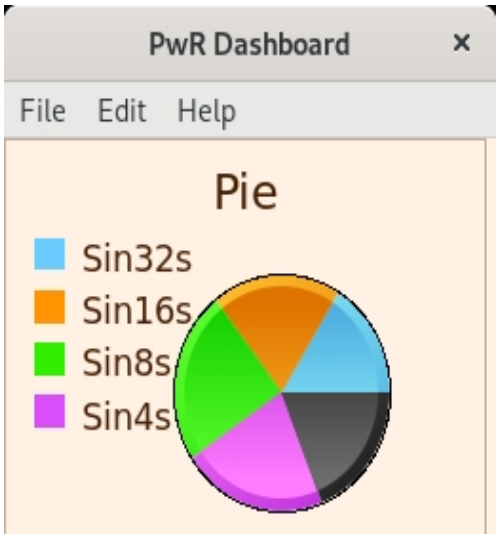
Gauge2



Slider

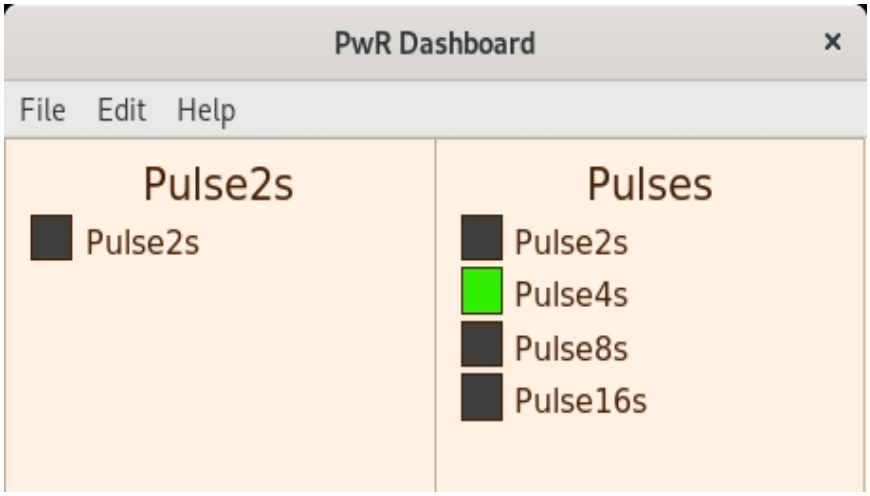


Pie

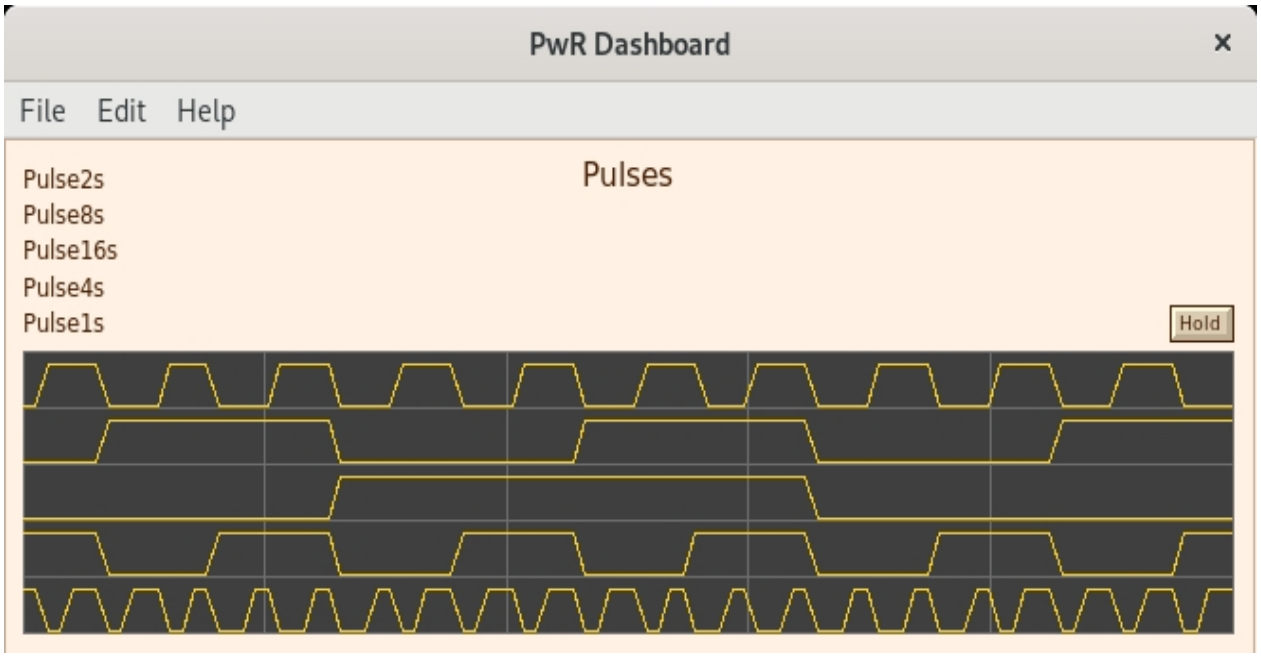


Digital dashboard cells

Indicator



DigitalTrend



Object dash cells

Object cells exist for some object, eg DsTrendCurve.

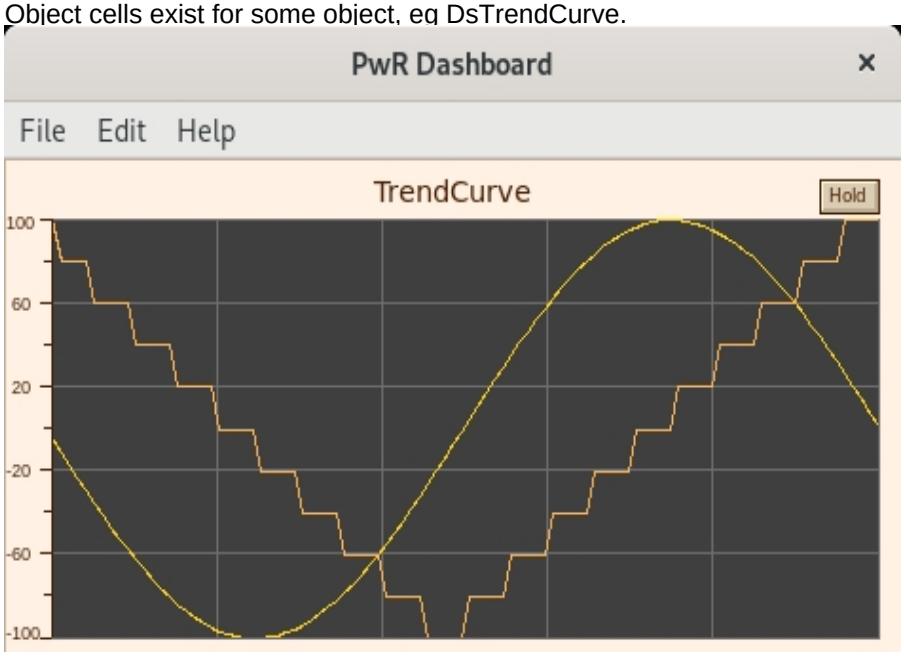


Fig Object cell for DsTrendCurve

Create a dashboard in rt_xtt

By selecting and attribute in the navigator, and activating Functions/Dashboard/Insert (Ctrl+D) in the menu, the attribute is added to a dashboard. By default it's added to the 'PwR Dashboard', but if there is a dashboard that is in edit mode, it will be inserted in this dashboard instead. It's possible to add new attributes to the dashboard until all rows and columns are filled.

For an analog attribute a bar cell is created. This can be changed by entering edit mode in the dashboard from File/Edit (Ctrl+E) in the dashboard menu. Open the cell attributes (double click) and select another type in Dash.Type. Leave the edit mode with Ctrl+E.

When entering edit mode, the Ge editor is opened in restricted mode. Some functionality in edit mode is

- Add. Create an empty cell.
- Delete. Delete cell.
- Copy. Copy a cell.
- Paste. Paste previously copied cells.
- Connect. Connect selected attribute in the navigator to cell.
- Merge. Merge cells. The selected cells will be merged into the one that was first selected.
- From GraphAttributes, the scan time and dashboard size can be modified.

10 Plc trace

Plc trace is a tool to study and debug the plc code.
Trace is opened from the method 'Open Plc', or from a crossreference in the Plc code.
You can also open trace by selecting a PlcPgm in the navigator and activate 'Functions/OpenProgram' (Ctrl+L) in the menu.

To the left there are a navigation window where the displayed area is marked with a square. By moving the square with drag MB1 you move the displayed area. You can also zoom by dragging with the middle button.

Digital objects with height status are marked with red in the code.

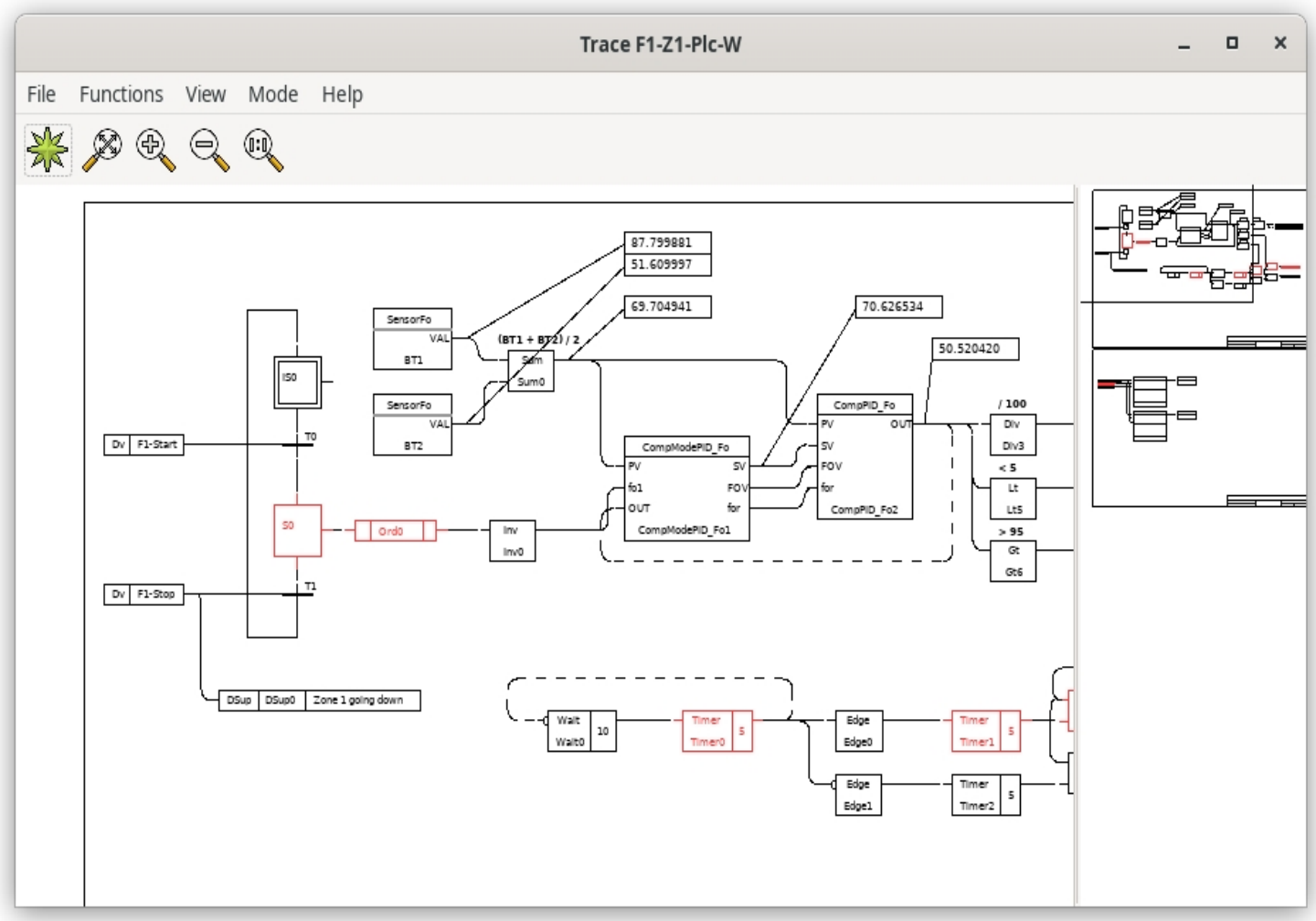


Fig Plc Trace

You can look at the value of analog or digital outputs by creating so called analyze nodes. They are created by dragging with the middle button from the output pin.

A set of analyze nodes can be stored and restored by 'File/Save Trace' and 'File/Restore Trace' in the menu.

When rightclicking on an object the methods of the objects are displayed. A very useful method in this context is crossreferences, The crossreferences are also displayed when doubleclicking on the object.

Crossreferences

The crossreferences are a list of the places an object or attribute occur in the code or in process graphs. If the reference implies reading the value, it is marked with a book, if it implies writing the value, it is marked with a pen. Reading can be that the value is displayed in a graph or fetched in the plc code. A write reference can be a pushbutton in a graph or a Set or Sto object in the plc code.

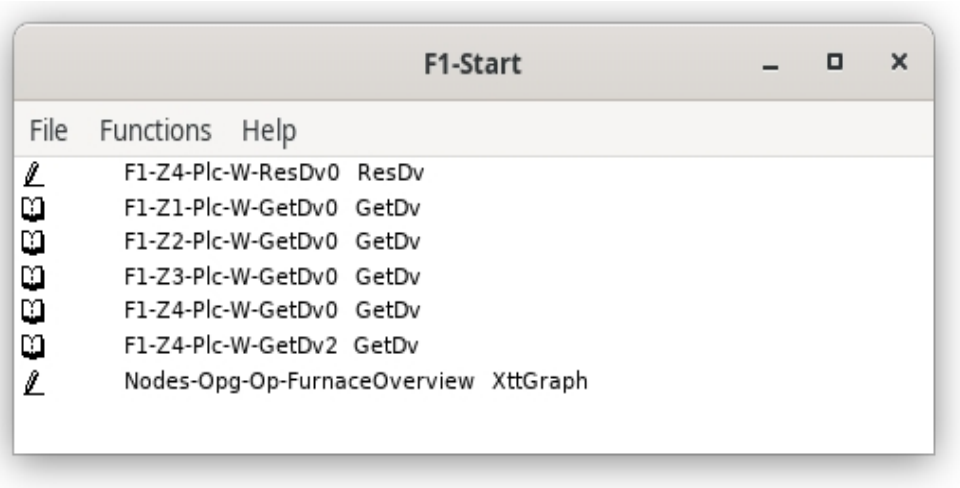


Fig Crossreferences

By rightclicking on the crossreference, the plc or process graph the reference is pointing at is opened. On plc references you can open PlcTrace with doubleclick. When Plc Trace is opened the reference is marked and centered.

Debugging with Plc Trace and Crossreferences

Plc Trace and Crossreferences are a very efficient tool for troubleshooting.
A troubleshooting often starts out from an alarm or error indication in a process graph. By activating the methods 'Crossreferences' or 'Open Plc' you find the place or places in the code where the object is written. By analysing the code you conclude which signal is missing, open the crossreferences for this and open PlcTrace where it is written. In this way you can search in the code and finally find the cause of the problem.

Simulation

At simulation, i.e. when testing a system without the I/O system, you can set PlcTrace in simulate mode with 'Mode/Simulate' in the menu. In simulate mode you can toggle digital signals by clicking with Ctrl/Shift MB1 on them in the Plc Trace window.

11 Setup script

A setup script can be created on the home directory with the name

```
xtt_setup.rtt_com
```

This will be executed when the operator environment (or Xtt) is started, and can contain script statements and xtt commands.

Here are some examples of useable commands.

Open a graph

```
open graph my_graph
```

Create a maintenance menu in the navigator

```
create item/text="Maintenance"/menu/destination="DataBase"/before
create item/text="My graph"/command="open graph my_graph"/pixmap=graph
        /dest=Maintenance/first
```

Remove menu items from the standarde menu of the navigator

```
delete item /name=close
delete item /name=system-nethandler
```

Short command for an hierarchy in the database

```
define rb9 "show children /name=hql-rb9"
```

Short command for a graph

```
define my_graph "open graph my_graph"
```

11.1 Symbol

An xtt symbol can be used as a short command or a variable in a command. If the symbol is used as a variable in a command in should be surrounded by apostrophes.

Symbols are created by the define command.

Example of a symbol used as a short command

```
xtt> define p1 "show child/name=hql-hvk-pumpar-pump1"
xtt> p1
```

Example of a symbol used as a variable

```
xtt> define p1 hql-hvk-pumpar-StartPump1
xtt> open trace 'p1'
```

12 Users and privileges

ProviewR contains a system with users that are granted privileges. To perform specific tasks certain privileges are needed, and only if the user has this privileges he is allowed to perform them.

The following privileges are used in the operator environment:

RtRead	Read authority in runtime.
RtWrite	Write authority in runtime. Authorized to change values of attributes from the navigator.
RtEvents	Privilege to handle alarm and event.
System	System manager privilege. Authorized for most things.
Maintenance	Privilege for maintenance personnel.
Process	Privilege for process engineer.
Instrument	Privilege for instrument technician.
Operator1	Privilegie for operator.
Operator2	Privilegie for operator.
Operator3	Privilegie for operator.
Operator4	Privilegie for operator.
Operator5	Privilegie for operator.
Operator6	Privilegie for operator.
Operator7	Privilegie for operator.
Operator8	Privilegie for operator.
Operator9	Privilegie for operator.
Operator10	Privilegie for operator.

Users

When the operator environment or Xtt is started, you login either as a user, or you are assigned the default privileges stated in the Security object.

If the startup is made with an OpPlace object, the user is fetched from the UserName attribute in the User object. If a user is missing or invalid an login window is opened and the user has to login with username and password.

If Xtt is started without OpPlace object, the privileges are fetched from the attribute DefaultXttPriv in the Security object. If there are no privileges that gives read authority, the login window is opened.

Navigator
To make changes in the database from the navigator, RtWrite or System privileges are required. This is also valid for the command 'set parameter'.

Plc Trace
To make changes in the database from PlcTrace in simulate mode, RtWrite or System are required.

Object graphs
Generally the privileges RtWrite or System are required to make changes in the database from an object graph. There are though some exceptions. In ChanAi and ChanAo also

Instrument has authority to change signal ranges, and to set an Ao in test mode. In the PID and Mode graphs, Process are authorized to change beside RtWrite and System. In the Mode graph you can furthermore control the authority by opening the graph with the command 'open graph' and use the /access option.

Process graphs

In the process graphs the designer chooses for each input field or pushbutton which privileges are required to make a change.

13 Function keys

You can use the function keys as shortcuts to open graphs or influence objects in the database.

Function calls

The following functions can be called when an function key is pressed.

SetDig()

Sets an attribute of type Boolean. The name of the attribute is specified as argument.

ResDig()

Resets an attribute of type Boolean. The name of the attribute is specified as argument.

ToggleDig()

Toggles an attribute of type Boolean. The name of the attribute is specified as argument.

Command()

Executes a Xtt command. The command is specified as an argument.

Setup file

The connection between the keys and the functions is specified in the file Rt_Xtt on the login directory. In the file there are one row for each key. On the row i written
- Possible modifiers (Shift, Control or Alt).
- <Key> followed by the name of the key, e.g. <Key>F7 or <Key>m.
- a colon followed by the function that is to be called with arguments.

A '#' sign in the first position denotes a comment.

Example

```
#
# Configuration of Global function keys
     <Key>F5: Command(event ack /prio=A)
     <Key>F6: Command(event ack /prio=NOA)
     <Key>F7: Command(show alarm)
     <Key>F8: Command(show event)
     <Key>F9: Command(close all/except=navigator)
     Shift Control <Key>m: SetDig(F1-Start.ActualValue)
     Shift Control <Key>n: SetDig(F1-Stop.ActualValue)
     Shift Alt <Key>n: SetDig(F1-Reset.ActualValue)
```

14 Status Monitor

The Status Monitor is a tool to supervise and handle process- and operator stations. You add nodes that you want to supervise and the statusmonitor views status for the node, with color indication and status information.

By default the SystemStatus is displayed for the nodes, but with the -m option another status assigned by the application can be displayed instead.

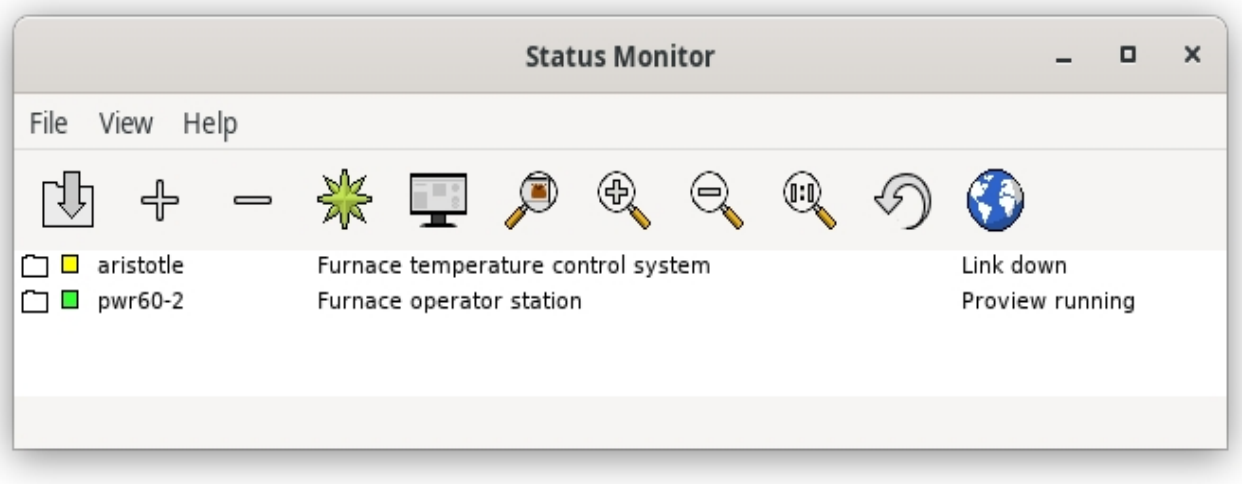


Fig The Status Monitor

The command to start the status monitor is

```
> rt_statusmon [-l 'language'][-c 'config-file'][-m 'status-index'][-e][-t][-o][-g]

-l  Language, en_us, sv_se, de_de or fr_fr.
-m  Display a user defined status insteadad of system status.
-c  Configuration file. Default $HOME/rt_statusmon.dat.
-e  Show node description from setup file.
-t  Hide buttons to start Runtime Monitor in toolbar and menu.
-o  Open map at start.
-g  Don't connect to ProviewR runtime.
```

Add and remove nodes

Nodes are added and removed to the list with the '+' and '-' buttons in the tool bar. When adding a node the nodename, ip address, bus id, description and operator place should be specified.

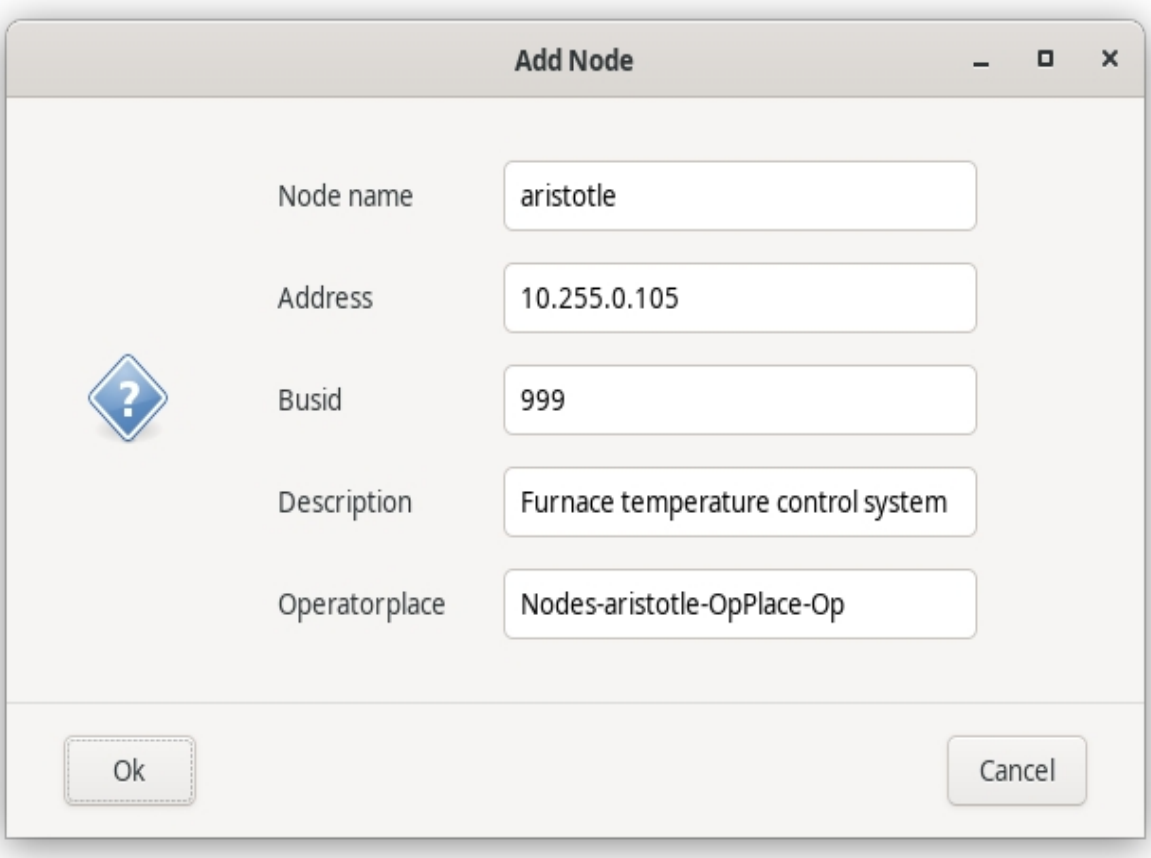


Fig Add node

More information is displayed by opening a node by clicking on the map.

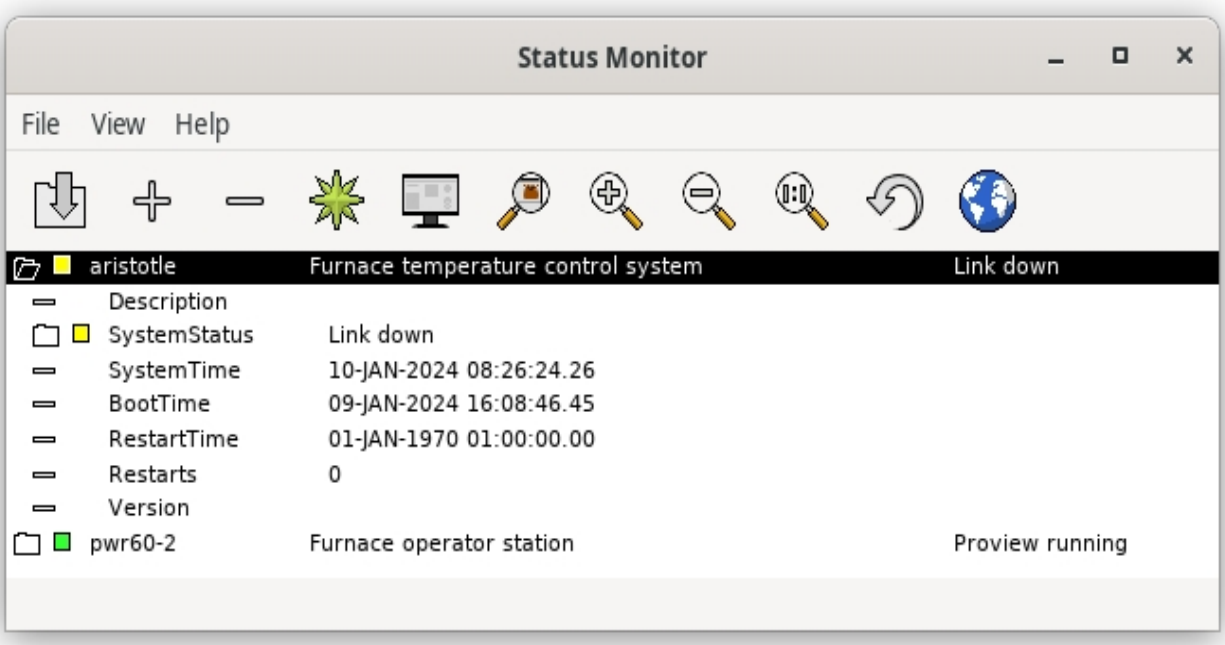


Fig Node info

By opening the SystemStatus map, the status for the individual processes are displayed.

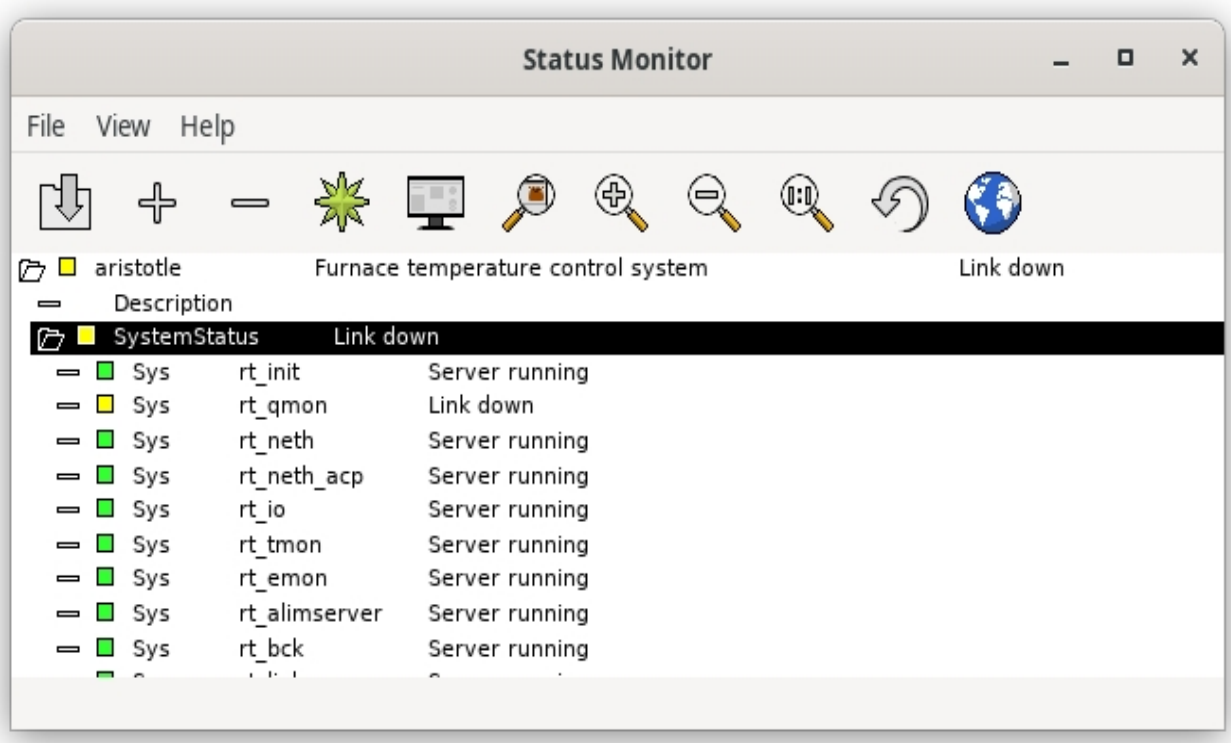


Fig Process status for a node

If a map is drawn over the plant, this can be opened from the globe in the toolbar. Indicators will show the status of each node, and a popup menues can be configured to open oeratorplace and xtt.

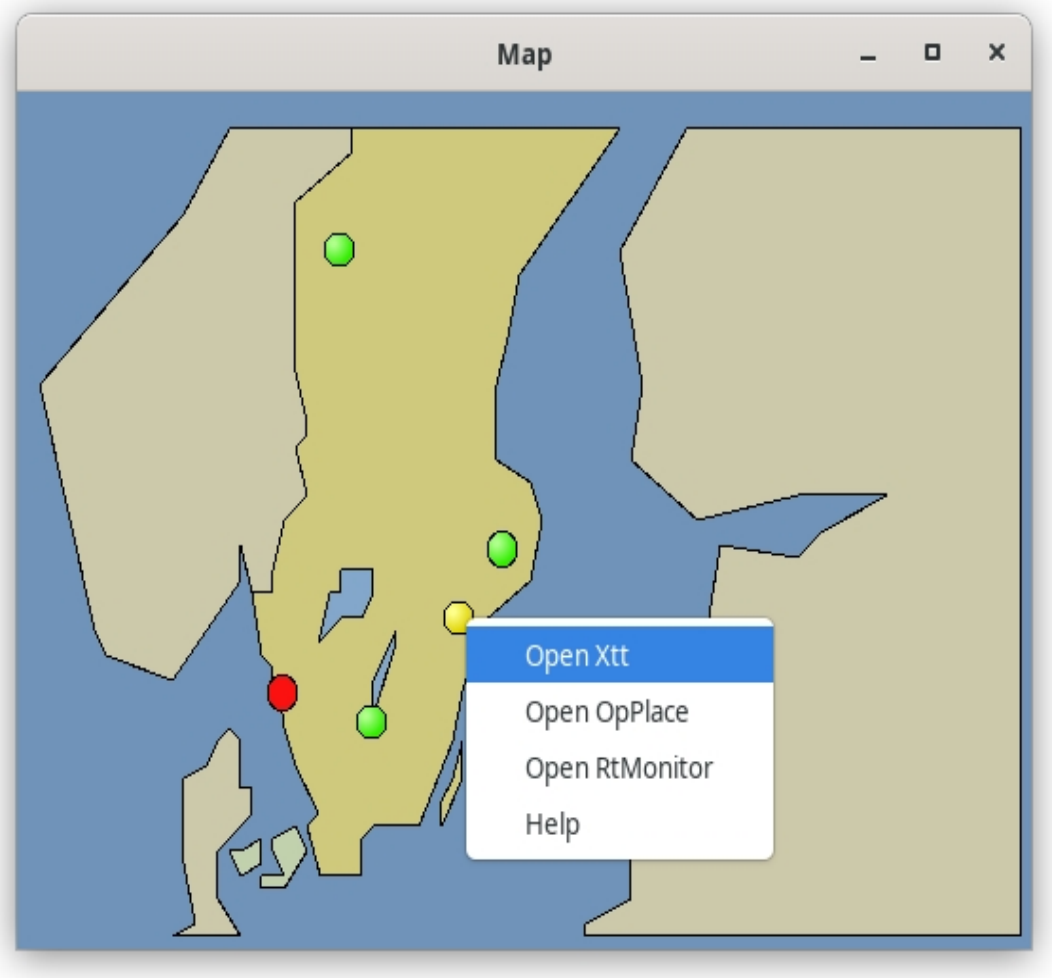


Fig Map

Commands

The status monitor contains a command line that is opened with Ctrl+B. The commands can be used for map popup menues in particular.

The following commands are defined

help
Display the help text for a topic.

```
help 'topic'
```

For qualifiers, see the Xtt help command

select node
Select a node in the node list.

```
select node /node=

/node             Name of the node to select.
```

open graph

Open a Ge graph.

```
open graph 'graph'
```

open opplace
Open operator place for specified node.

```
open opplace /node=

/node              Name of node.
```

open xtt
Open runtime navigator for specified node.

```
open xtt /node=

/node              Name of node.
```

open rtmon
Open runtime monitor for specified node.

```
open rtmon /node=

/node              Name of node.
```

15 Xtt commands

add parameter	Add an attribute to a picture
add menu	Add a menu item to a xtt menu
call method	Call a xtt method for an object
check method	Call a xtt method filter for an object
collect	Add an attribute to collect list
collect show	Show the collect list
collect clear	Clear the collect list
collect remove	Remove an attribute from the collect list
collect open	Open a stored collect window
close alarmlist	Close alarmlist
close eventlist	Close eventlist
close graph	Close a graph
close multiview	Close a multiview
close navigator	Close the navigator
create item	Create a menu item in the Xtt menu
create opmenuitem	Create a menu item in the operator window menu
crossreference	Show crossreferenses
define	Define a symbol
delete item	Delete a menu item in the Xtt menu
delete opmenuitem	Delete a menu item in the operator window menu
emit signal	Emit a Ge signal
eventlist	Handle the eventlist
exit	close xtt
help	Display help
login	User login
logout	User logout
logging	Logg attributes to file
open	Open various windows
plcscan	Set plc scan on or off
read object	Read the content of an object from file
search	Search for an object or a string
set advanceduser	Set advanced user
set display	Set format of display for selected attribute
set folder	Set a folder in a tabbed window.
set parameter	Set the value of an attribute
set signal	Set invert, conversion and test for signals.
set subwindow	View a graph in a Ge window object.
set subevents	Disable events in a Ge window or table object.
set graph	Handling of a graph
setup	Xtt setup
show version	Show xtt version
show symbol	Show a symbol
show plcpgm	Show PlcPgm objects
show plcthreads	Show PlcThread objects
show links	Show links
show logfiles	Show rtt loggfiles
show subsrv	Show subscription server
show subcli	Show subscription client
show device	Show devices
show remnode	Show RemNode objects
show remtrans	Show RemTrans objects

show database	Show the runtime database
show file	Show files
show graph	Show graph-files
show time	Show current time
show default	Show default directory
show children	Show the children of an object
show message	Show a message
show objectlist	Show all objects of a specific class
show objid	Show object identity
show parameter	Show an object attribute
show logging	Show a logging entry
show eventlist	Show the event list
show alarmlist	Show the alarm list
show user	Show current user
sound	Play a sound defined by a sound object
store	Store an attribute list in a file
write object	Write the content of an object to file

15.1 Command call method

Call a xtt method for an object.
Xtt methods are usually activated from the popup menu for an object.
This is an additional way to activate a methods, from a command
pushbutton in a Ge graph.

xtt> call method /method= /object=

/method=	the name of the method, i.e. the attribute ButtonName in the method definition.
/object=	the name of the object.

15.2 Command check method

Call a xtt method filter for an object.
Method filter are used to decide whether to display a method button or not.
This command is used to disable a method button in a Ge graph. It returns
the filter value to the caller.
It is only valid in a Ge graph.

xtt> check method /method= /object=

/method=	the name of the method, i.e. the attribute ButtonName in the method definition.
/object=	the name of the object.

15.3 Command add parameter

Add an attribute to a list of attributes, and display the value of
the attribute.
A list of attributes will be created with the command 'show parameter',
and more attributes can be added to the list with 'add parameter'.

Object that matches the class, name and hierarchy description are displayed
on the screen.

xtt> add parameter /parameter= /name= /class= /hierarchy=

/parameter=	the name of the parameter to be displayed.
/name=	the name of the object. Wildcard is allowed.
/class=	displays objects of this class.
/hierarchy=	displays objects below this object in the hierarchy.

related subjects

show parameter

15.4 Command show parameter

Create a new list, add an attributes to the list, and display the value of the attributes.
More attributes can be added to the list with 'add parameter'.

Object that matches the class, name and hierarchy description are displayed on the screen.

xtt> show parameter /parameter= /name= /class= /hierarchy=

/parameter=	the name of the parameter to be displayed.
/name=	the name of the object. Wildcard is allowed.
/class=	displays objects of this class.
/hierarchy=	displays objects below this object in the hierarchy.

related subjects

add parameter

15.5 Command set parameter

Set the value of an attribute.

xtt> set parameter /name= /value=

/name=	the name of the parameter to be set.
/value=	value to set.
/bypass=	Bypass any access restrictions.

15.6 Command set folder

Set a folder in a tabbed window object.

xtt> set folder 'graphname' /name= /index=

/name=	the name of the tabbedwindow object.
/index=	index of tab to set. Starting from 0.

15.7 Command set signal

Set invert, conversion, test and testvalue for a signal.

Conversion can be set on Di, Ii and Ai,
Invert can be set on Di and Do.
Test can be set on Do, Ao and Io.
Testvalue can be set on Do.

xtt> set signal conversion /on [/name=]
xtt> set signal conversion /off [/name=]
xtt> set signal invert /on [/name=]
xtt> set signal invert /off [/name=]
xtt> set signal test /on [/name=]
xtt> set signal test /off [/name=]
xtt> set signal testvalue /on [/name=]
xtt> set signal testvalue /off [/name=]

/name	Signal name. If not supplied the currently selected signal is used.
/on	Set signal entity on.
/off	Set signal entity off.

15.8 Command set subwindow

View a graph in a Ge window object. This command is used on command buttons to change the content of a window object. /source contains the name of the graph to be viewed. Also object graphs can be viewed by supplying the object with the /object qualifier.

xtt> set subwindow 'graphname' /name= /source= [/object=]

'graphname'	The graph where the window object resides. '$current' denotes the current graph.
/name	The name of the window object.
/source	Name of the Ge graph to view in the window object.
/object	Specifies the object when the graph is an object graph.
/x0	Upper left x Ge coordinate for graph border. If x0, y0, x1, y1 are supplied, another part of the graph than the one specified in the Graph attributes x0, y0, x1, y1 will be displayed. Implemented for graphs in a multiview cell, not for window objects.
/y0	Upper left y Ge coordinate for graph border.
/x1	Lower right x Ge coordinate for graph border.
/y1	Lower right y Ge coordinate for graph border.
/continue	By default other actions for the command button are discarded as the current graph may have replaced itself. To also execute the succeeding actions for the button, the /continue qualifier is added.

If the 'set subwindow' command is used in a script and replaces the current graph, the script should be terminated with a call to exit(GLOW__SUBTERMINATED).

```
main()
  set subwindow motor1 /name=subw1 /source=motor1.pwg
  ...
  exit( GLOW__SUBTERMINATED);
endmain
```

15.9 Command set subevents

Disable events in a Ge window or table object.
Window and table objects grab all events inside their area, and click sensitive objects can not be placed on top of them. If they temporarily should be covered by a sensitive object though, the eventhandling can be turned off for window and table objects in the graph.

Events will be disabled for all window, tabbed window and table objects in the graph.

xtt> set subevents 'graphname' [/on] [/off]

'graphname'	the graph where the window object resides. '$current' denotes the current graph.
/off	Disable events in window and table objects.
/on	Enable events in window and table objects.

15.10 Command add menu

Add a menu item to the xtt menu hierarchy.
The menu is placed last in the current menu.
The action of the item can be to execute an xtt
command, or display the attributes of an object.

xtt> add menu /text= /command=
xtt> add menu /text= /object=

/text	Text of the menu item.
/command	Xtt command to be executed when the item is activated.
/object	Object that will be displayed when the item is activated.

related subjects

create item
delete item

15.11 Command collect

Add an attribute to the collection list.
If the name qualifier is omitted the current selected attribute will
be collected.

The /newwindow and /addwindow qualifiers are usually used in command
files for stored collections windows (files of type .rtt_col)

xtt> collect
xtt> collect /name=
xtt> collect /name= /newwindow /width= /height= /zoomfactor= /scantime=
/title= /last
xtt> collect /name= /addwindow /last

/name	Name of the attribute
/newwindow	Creates a new window and inserts the attrbute into this window.
/width	Window width.
/height	Window height.
/zoomfactor	Text zoom factor in new window.
/scantime	Update frequency for values.
/title	Window title.
/last	This is the last attribute to add for this window.
/addwindow	The attribute is added to the last created collect window.

Example

xtt> collect /name=hql-hvk-Start.ActualValue

related subjects

collect show
collect clear
collect remove
collect open

15.12 Command collect show

Display the collect list.

xtt> collect show

related subjects

collect
collect clear

15.13 Command collect clear

Clear the collect list.

xtt> collect clear

related subjects

collect
collect show
collect remove
collect open

15.14 Command collect remove

Clear an item in the collect list.

xtt> collect remove

related subjects

collect
collect show
collect open

15.15 Command collect open /file=

Open a stored collection window. If file is not specified, a list of store windows are displayed.

xtt> collect open
xtt> collect open /file=Motor2

related subjects

collect
collect show
collect remove
collect open

15.16 Command create opmenuitem

Create a menu item in the operator window menu.
The action of the item can be either to execute a command.

xtt> create opmenuitem /name= /command= /pixmap=

/name	Menu item name, with the name of above menues included. The name should begin with 'Functions-' or its translation. For other name segments submenues are created if they don't already exist. Ex Functions-MyMenu-MyGraph
/command	Xtt command that will be executed when the menu item is activated.
/before	The menu item is positioned first in the menu.
/after	The menu item is positioned last in the menu.
/pixmap	Pixmap for the menu item: 'graph' eller 'curve'.

Example

xtt> create opmenuitem /name="Functions-MyMenu-MyGraph"/after/pixmap=graph

related subjects

delete opmenuitem

15.17 Command create item

Create a menu item in the xtt menu hierarchy.
The action of the item can be either to execute
a command, or to open a menu.

xtt> create item /text= /command= /pixmap= /destination= /after /before /firstchild /lastchild
xtt> create item /text= /menu /destination= /after /before /firstchild /lastchild

/text	Item text.
/command	Xtt command to be executed when the item is activated.
/menu	The item will display a menu.
/destination	The destination item. Can be the parent or a sibling. If nullstring the item is created on the top level.
/after	Positions the item after the destination item.
/before	Positions the item before the destination item.
/firstchild	Positions the item as first child of the destination item.
/lastchild	Positions the item as last child of the destination item.
/pixmap	Pixmap for the item: 'map', 'leaf', 'graph', 'list' or 'script'.

Example

xtt> create item /text="Motor1"/dest=Maintenance-Motors/command="open graph motor1"/first

related subjects

add menu
delete item

15.18 Command crossreference

Show crossreferences.
Crossreferences can be displayed for
- Signals.
- Object references with GetData.
- c-functions or strings used in the code of CArithm- or DataArithm-objects.

If qualifiers are omitted the selected object will be examined.

xtt> crossreference
xtt> crossreference /name=
xtt> crossreference /function= [/brief]
xtt> crossreference /string= [/brief]

/name	Name of an object.
/function	Name of a c-function referenced in CArithm or DataArithm.
/string	String that will be searched for in the code of CArithm- and DataArithm-object.

Examles

xtt> cross /name=hql-hvk-Start
xtt> cross /function="CreateHvkObject"

15.19 Command emit signal

Emit a Ge signal.
The signal is be emitted to a specific graph, or to all graphs and multiviews if graph is not specified.

xtt> emit signal /signalname= {/graph=] [/instance=]

/signal_name	Name of signal.
/graph	Graph or multiview to which the signal is directed.
/instance	Instance object if the graph is an object graph.

15.20 Command exit

Close xtt.

xtt> exit

15.21 Command define

Define a symbol.

xtt> define 'symbolname' 'text'

related subjects

symbol
show symbol
symbolfile

15.22 Command delete opmenuitem

Delete a menu item in the operator window menu.
The menu item can be created by the user or an item of the standard configuration.

xtt> delete opmenuitem /name=

/name Name of the menu item.

Example

xtt> delete opmenuitem /name=Functions-Curves

related subjects

create opmenuitem

15.23 Command delete item

Delete a menu item.
The menu item can be created by the user or an item of
the xtt standard configuration.

xtt> delete item /name=

/name Name of the menu item.

Example

xtt> delete item /name=exit

related subjects

add menu
create item

15.24 Command eventlist

Handle the eventlist

xtt> eventlist load

Load the eventlist.

xtt> eventlist unload

Unload the eventlist.

xtt> eventlist acknowledge /priority=

Acknowledge last alarm of specified priority. Priority can be A, B, C, D, I or NOA. NOA acknowleges the last not A alarm.

xtt> eventlist acknowledge /all

Acknowledge all alarms.

xtt> eventlist acknowledge /autoacknowledge=

Automatically acknowlege all alarms. The acknowlegde will be made cyclically with the cycletime specified by /autoacknowledge. This function is made for test purposes and requires System privilege.

related subjects

show alarmlist
show eventlist

15.25 Command help

Display help information for a subject.
The help information will be searched for in a help file. The file can be the base helpfile, the project helpfile or another help file.

If no helpfile is supplied the subject will be searched for in the base and project helpfiles.

xtt> help 'subject' [/popnavigator] [/bookmark=] [/helpfile=] [/returncommand=][/width=][/height=]

/popnavigator	The help window (the navigator) will be displayed on top of the other window and gain input focus. This is useful if the command is used in a ge graph.
/bookmark	The name of a bookmark inside the topic. The display will be positioned at the bookmark.
/helpfile	A help file that contains information of the help subject.
/returncommand	A command that is executed when returning from the help subject. This is usually used to return to a main topic when help is called from a command button in a ge picture.
/width	Desired width of the help window.
/height	Desired height of the help window.

related subjects

helpfile

15.26 Command login

Login with username an password. The privileges of the user will be fetched from the user database, and affect the access to the system.

xtt> login 'username' 'password'

related subjects

logout

15.27 Command logout

Logout a user, and return to the original user.

xtt> logout

related subjects

login

15.28 Command open

Open various windows.

open attribute	Open a value change dialog for an attribute
open consolelog	Display system messages
open fast	Open fast curve
open file	Open a file or URL
open fileview	Open a fileview
open graph	Open a graph
open history	Open process history curve
open multiview	Open a multiview
open jgraph	Open a java graph
open navigator	Open the navigator
open operatorwindow	Open the operator window
open shistory	Open a list with all history curves for an object
open trace	Open trace for a plc window
open trend	Open DsTrend or PlotGroup graph
open tcurve	Open a trend snapshot
open url	Open an URL in a web browser

15.29 Command open graph

Open a ge graph. The graph can be supplied as a pwg-file or as a XttGraph object.

xtt> open graph 'filename' /width= /height= /scrollbar /menu /navigator
/instance= /focus= /inputempty

xtt> open graph /object= /focus= /inputempty
xtt> open graph /class /instance=

/width	Width in pixel.
/height	Height in pixel.
/scrollbar	Scrollbars will be displayed in the graph window.
/menu	A menu will be displayed in the graph window.
/navigator	A navigator window will be opened.
/object	An XttGraph-object. If the name starts with '$node' or '*' this will be replaced by the node object.
/instance	Open the object graph or an hierarchy graph for this instance. If /class is supplied the object graph is opened, otherwise a filename can be supplied.
/class	Open the object graph for the instance object.
/parent	Specifies that the object graph for the parent of the instance object should be opened.
/focus	Name of a value input object. The object will gain input focus when the graph is opened.
/inputempty	The input label of the input focus object will be empty.
/pwindow	Filename of another window that this graph should be placed on top of. The string $current will denote the current window when the graph is opened from a push button in this window.
/pinstance	Instance name for the pwindow graph, if pwindow is an object graph.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.
/hide	Don't display the graph on the screen.
/x0	Upper left x Ge coordinate for graph border. If x0, y0, x1, y1 are supplied, another part of the graph than the one specified in the Graph attributes x0, y0, x1, y1 will be displayed.
/y0	Upper left y Ge coordinate for graph border.
/x1	Lower right x Ge coordinate for graph border.
/x1	Lower right y Ge coordinate for graph border.

15.30 Command open multiview

Open a multiview window specified by an XttMultiView object.

xtt> open multiview 'objectname' /width= /height= /xposition= /yposition=
/fullscreen /maximize /iconify /hide

'objectname'	Name of and XttMultiView object. If the name starts with '$node' or '*' this will be replace by the node object.
/width	Window width in pixels.
/height	Window height in pixels.
/xposition	Window x position in pixels.
/yposition	Window y position in pixels.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.

15.31 Command open shistory

Open a list of history curves for an object.
Displays a list of all SevHist and SevHistObject storing process history for different attributes in an object.

xtt> open shistory /name=

/name	Object name.

15.32 Command open history

Open process history for an object. Data is fetched from the storage server, but can also read from a file containing history data previously stored with the export function.

xtt> open history [/name=] [/title=]
xtt> open history /file=

/name	An object with stored process history.
/file	Open a file containing history data previously stored with the export function.
/title	Title of curve window.
/width	Window width in pixels.
/height	Window height in pixels.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.

Example

xtt> open history /name=hql-hvk-flow

15.33 Command open attribute

Open a value input dialog for an attribute.

xtt> open attribute /name= [/title=]

/name	Attribute name.
/title	Window title.

15.34 Command open consolelog

Display system messages.

xtt> open consolelog

15.35 Command open url

Open an URL in a web browser.

xtt> open url 'URL'

Example
xtt> open url http://www.proview.se

15.36 Command open jgraph

Open a java frame. The java frame can be exported from Ge or drawn in JBuilder.

xtt> open jgraph 'name'

15.37 Command open navigator

Open the runtime navigator. If an object is supplied the object is selected in the navigator object tree.

xtt> open navigator [/object=]

/object	Name of an object or attribute that is viewed and selected. Optional.

15.38 Command open file

Opens a file or URL in the user's preferred application.
If the filename contains a slash it has to be surrounded by quotes.

xtt> open file 'filename'

Example
xtt> open file "$pwrp_doc/manual.pdf"

15.39 Command open fileview

Opens a fileview that displays files that matches the specified pattern.
The fileview can be opened in 'Open File' or 'Save File' mode.

In open mode, a file is selected and the selected filename is written to a string attribute in the database (target). Simultaneously a digital attribute is set (trigger) to indicate that an open should be performed. The actual opening can be done by a DataFRead object in the plc, but has to be programmed by the user.

In save mode, a file is selected, or a filename is entered. The filename is written to a string attribute and a digital signal is set. The actual save can be performed by a DataFWrite object in the plc.

xtt> open fileview /file= /target= /trigger= /type=save [/ftype=]
xtt> open fileview /file= /target= /trigger= /type=open

/file	File pattern, containing wildcard '*', for files that is to be viewed in the fileview.
/ftype	Default file type for input file. If an entered filename (in save mode) doesn't contain a filetype, the default filetype will be added to the filename.
/target	String attribute where the selected filename will be stored.
/trigger	Boolean attribute that is set to indicate that a save or open should be performed.
/type	'save' or 'open'

Example
open fileview /file="$pwrp_load/*.txt"/ftype="txt"/target=P1-Sv.ActualValue/trigger=P1-Dv.ActualValue/type=save

15.40 Command open trace

Open trace for a plc window. If the name qualifier is omitted,
trace will be opened for the selected PlcPgm or PlcWindow.

xtt> open trace [/name=] [/center=]

/name	PlcPgm or PlcWindow.
/center	Object in the plc-window that will be centered and selected. Use the last segment of the name.

Examples

xtt> open trace /name=hql-hvk-flow /center=Pid0

15.41 Command open trend

Open a graph for a DsTrend, DsTrendCurve or PlotGroup object.

xtt> open trend [/name=] [/title=]

/name	DsTrend, DsTrendCurve or PlotGroup. Can be a list of DsTrend-objects separated by comma.
/title	Title of curve window.
/width	Window width in pixels.
/height	Window height in pixels.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.

Examples

xtt> open trend /name=hql-hvk-flow-Trend,hql-hvk-temperature-Trend

15.42 Command open fast

Open a fast curve specfied with a DsFastCurve object.

xtt> open fast /name= [/title=]
xtt> open fast /file=

/name	DsFastCurve or PlotGroup. Can be a list of DsFast-objects separated by comma.
/file	Opens a file previously created with the export function for fast curves.
/title	Title of curve window.
/width	Window width in pixels.
/height	Window height in pixels.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.

Examples

xtt> open fast /name=hql-hvk-flow-Fast

15.43 Command open tcurve

Takes a snapshot of a trend curve and displays the snapshot. With the /new qualifier
an empty curve window is opened where a previously stored snapshot curve can be viewed.

xtt> open tcurve /name= [/title=] [/height=] [/width=]
xtt> open tcurve /new [/title=]

/name	DsTrendCurve object to take a snapshot from.
/new	Opens an empty tcurve window where a stored snapshot curve can inserted from File/Open in the menu.
/title	Title of graph window.
/width	Window width in pixels.
/height	Window height in pixels.
/fullscreen	Open the window fullscreen without frame.
/maximize	Open the window maximized with frame.
/iconify	Open the window iconified.

Example

xtt> open tcurve /name=hql-hvk-flow-Trend

15.44 Command open operatorwindow

Open the operatorwindow.

xtt> open operatorwindow 'opplace-object'

15.45 Command close graph

Close a ge graph. The graph can be supplied as a pwg-file or as a XttGraph object. To close a graph from a button inside the graph $current can be used, 'close graph $current'.

xtt> close graph 'filename' [/instance=] [/iconify]
xtt> close graph /object=

/object	A XttGraph-object. If the namestring begins with '*', * is replaced by the node object.
/instance	Close the class graph for this instance.
/iconify	The graph is iconified instead of closed.

15.46 Command close multiview

Close a multiview window.
To close a multiview from a button inside the multiview $current can be used 'close multiview $current'.

xtt> close multiview 'objectname'

'objectname' Name of the XttMuliView object.

15.47 Command close alarmlist

Close the alarmlist.

xtt> close alarmlist

15.48 Command close eventlist

Close the eventlist.

xtt> close eventlist

15.49 Command close navigator

Close the navigator.

xtt> close navigator

15.50 Command search

Search for an objectname or a string.

xtt> search 'object'
xtt> search /regularexpression 'expression'
xtt> search /next

15.51 Command plcscan

Set plc scan on or off for all plc programs or for selected or specified program.

System privilege is required.

xtt> plcscan [/on] [/off] [/all] [/name=]

/on	Set scan on.
/off	Set scan off.
/all	Set scan on or off for all plc programs.
/name	Set scan on or off for specified plc program or window.

15.52 Command read object

Read the content of an object from file.
Reads the content of a file, normally created by command 'write object',
and insert the value into the specified object.

xtt> read object /object= /file=

/object	Name of object.
/file	Name of file to read from.

15.53 Command set advanceduser

Set or reset advanced user.

xtt> set advanceduser
xtt> set noadvanceduser

related subjects

advanced user

15.54 Command set display

Set format for display of the selected attribute. The attribute can be displayed as decimal, hexadecimal, octal, binary, integer or float. Object identities and attribute references can be displayed as identity.

xtt> set display default
xtt> set display hexdecimal
xtt> set display decimal
xtt> set display octal
xtt> set display binary
xtt> set display integer
xtt> set display float
xtt> set display identity

15.55 Command set graph

Executes a storage of open inputfields with the EscapeStore property set.

xtt> set graph 'filename' /escapestore [/instance=]

/escapestore	Executes a storage of open inputfields with the EscapeStore property set.
/instance	Instance specified for the graph.

15.56 Xtt setup

Setup of xtt properies

ConfigureObject	RttConfigure object.
DefaultDirectory	Default directory for commandfiles.
Scantime	Scan time for update fileds.
AlarmMessage	Display last alarm in the Xtt-message field.
AlarmBeep	Beep if unacknowledged alarms exist.
AlarmReturn	Display return events in the eventlist.
AlarmAck	Display acknowledged events in the eventlist.
SymbolFilename	Symbolfile.
Verify	Verify commandfile execution.
AdvancedUser	User is advanced.

15.57 Command show version

Show the xtt version

xtt> show version

15.58 Command show symbol

Show one symbol, or all symbols

xtt> show symbol 'symbol'	**Show symbol 'symbol'**
xtt> show symbol	**Show all symbols**

related subjects

define
symbol

15.59 Command show plcpgm

List the PlcPgm-objects in the system.

xtt> show plcpgm

15.60 Command show plcthreads

Display the PlcThread-objects.
The PlcThreads contains information and statistics for the plc-threads.

xtt> show plcthreads

15.61 Command show links

Show the links to other proview systems.

xtt> show links

15.62 Command show logfiles

Show the rtt log-files in the current work directory.
A log-file is created by the logging function in rtt or xtt and has the filetype .rtt_log.

A logfile is opened with double-click, or by selecting the file and then pressing return.

xtt> show logfiles

15.63 Command show alarmlist

Open the alarmlist window.
With the 'show alarmlist satellite' command multiple alarmlists can be opened with differnt alarmview.

xtt> show alarmlist
xtt> show alarmlist satellite [/alarmview=] [/width=] [/height=]
[/xposition=] [/yposition=]

/alarmaview	Name of an AlarmView object.
/width	Window width in pixels.
/height	Window height in pixels.
/xposition	Window x position in pixels.
/yposition	Window y position in pixels.

15.64 Command show eventlist

Open the eventlist window.

xtt> show eventlist

15.65 Command show message

Open a message window.

xtt> show message /text= [/title=]

/text	Message text.
/title	Window title.

15.66 Command show objectlist

Open a list dialog with all objects of a specific class or
a number of specific classes. When an object in the list is
activated, the object graph for the object is opened.

xtt> show objectlist /class= [/title=] [/sort]

/class	The name of a class, or serveral class names separated by comma.
/title	Title of the list dialog.
/sort	The objects are sorted in alphabetical order.

15.67 Command show objid

Display object identity for an object.
If name is not specified, the identity for the selected object is displayed.

xtt> show objid [/name=] [/hexadecimal]

/name	The name of the object.
/hexadecimal	Displays the object index in hex.

15.68 Command show user

Show the current user and his privileges.

xtt> show user

15.69 Command sound

Play a sound defined by a Sound or SoundSequence object.

xtt> sound 'object'

15.70 Command store

Store an attribute list to a scrip-file, or store the current
symbol table to a scrip-file.
The generated script-file can be executed from the command-prompt
with @'filename', or it can be found among the script-files in the
menu beneath 'Store'.

If the /collect is used, the attribute-list
will be restored in the collect list.

xtt> store 'filename' [/collect]
xtt> store 'filename' /symbols

/collect	The list is restored in the collection list.
/symbols	The current symbol table is stored.

15.71 Command show logging

Show a logging entry.

xtt> show logging /entry=

15.72 Command logging

A number of commands handle the xtt logging.

logging start
logging stop
logging set
logging show
logging store

15.73 Command write object

Write the content of an object to file.

xtt> write object /object= /file=

/object	Name of object.
/file	Name of file to write to.

16 Xtt script

Execute script

Datatypes and declarations

Datatypes
Datatype conversions
Variable declarations
Operators

Statements

main-endmain
function-endfunction
if-else-endif
while-endwhile
for-endfor
break
continue
goto
include

Input/output functions

ask()
printf()
say()
scanf()

File handling functions

fclose()
felement()
fgets()
file_search()
fopen()
fprintf()
fscanf()
translate_filename()

String functions

edit()
element()
extract()
sprintf()
strchr()
strrchr()
strlen()
strstr()

toupper()
tolower()

Database functions

GetAttribute()
SetAttribute()
GetChild()
GetParent()
GetNextSibling()
GetClassList()
GetClassListAttrRef()
GetNextObject()
GetNextAttrRef()
GetObjectClass()
GetNodeObject()
GetRootList()

System functions

exit()
get_namespace()
set_namespace()
system()
terminate()
time()
tzset()
verify()

Array functions

arrayclear()
arraypush()
arraysize()
sort()

Math functions

cos()
EVEN()
MAX()
MIN()
ODD()
random()
sin()

Miscellaneous functions

CutObjectName()
ConfirmDialog()
TextDialog()
MessageError()
MessageInfo()
GetCurrentText()
GetCurrentObject()
get_pwr_config()
get_node_name()
getmsg()

EVEN()
ODD()
get_language()
GetUser()
GetPrivileges()
GetGraphInstance()
GetGraphInstanceNext()
SetSubwindow()
Quit()

xtt-commands

xtt-commands

16.1 Execute a script

A script-file will be executed from the command-line with the command

xtt> @'filename'

16.2 Datatypes

The datatypes are float, int and string.

int	integer value.
float	32-bit float value.
string	80 character string (null terminated).

There are three different tables in which a variable can be declared: local, global and extern. A local variable is known inside a function, a global is known in all functions in a file (with include-files), an external is known for all files executed in a session.

16.3 Datatype conversions

If an expression consists of variables and functions of different datatypes the variables will be converted with the precedence string, float, int. If two operands in an expression is of type int and float, the result will be float If two operands is of type float and string, or int and string, the result will be string. In an assignment the value of an expression will be converted to the type of the assignment variable, even if the result is a string and the variable is of type float or int.

Example

```
string str;
int    i = 35;
```

```
str = "Luthor" + i;
```
The value in str will be "Luthor35".

```
float   f;
string  str = "3.14";
int     i = 159;
f = str + i;
```
The value in f will be 3.14159.

16.4 Variable declarations

A variable must be declared before it is used.
A declaration consists of
- the table (global or extern, if local the table is suppressed)
- the datatype (int, float or string)
- the variable name (case sensitive)
- if array, number of elements in square brackets, or if array with zero lenght, only brackets
- equal mark followed by an init value, if omitted the init value is zero or null-string
- semicolon

An extern variable should be deleted (by the delete statement).
Global variables can also be deletet with the deletegbl statement.

Example

```
int             i;
float           flow = 33.4;
string          str = "Hello";
float           width[5] = (1.20, 2.44, 4.81, 7.77, 9.20);
extern int      jakob[20];
string          simon[];
global float    ferdinand = 1234;
...
delete jakob[];
deletegbl ferdinand;
```

Namespace

Extern variables can be declared in different namespaces by setting the namespace with set_namespace(). This can be used for running sets of scripts concurrently in the same process. A set of script can handle common extern variables, and by using different namespaces, the same set can be run in several instances without mix-up of the extern variables.

16.5 Operators

The operators have the same function as i c, with some limitations. All operators are not implemented. Some operators (+,=,==) can also operate on string variables. Precedence of operators is similar to c.

Operator	Description	Datatypes
+	plus	int, float, string
-	minus	int, float
*	times	int, float
/	divide	int, float
++	increment, postfix only.	int, float
--	decrement, postfix only	int, float
>>	bits right-shifted	int
<<	bits left-shifted	int
<	less than	int, float
>	greater than	int, float
<=	less equal	int, float
>=	greater equal	int, float
==	equal	int, float, string
!=	not equal	int, float, string
&	bitwise and	int
|	bitwise or	int
&&	logical and	int
||	logical or	int
!	logical not	int
=	assign	int, float, string
+=	add and assign	int, float
-=	minus and assign	int, float
&=	logical and and assign	int
|=	logical or and assign	int

16.6 Script statements

main-endmain	Main function.
function-endfunction	Function declaration.
if-else-endif	Conditional execution.
while-endwhile	While loop.
for-endfor	For loop.
break	Terminate while or for loop.
continue	Continue while or for loop.
goto	Jump to label.
include	Include script file.

16.6.1 main-endmain

The main and endmain statements controls where the execution starts and stops
If no main and endmain statements will be found, the execution will start
att the beginning of the file and stop at the end.

Example

```
main()
  int a;

  a = p1 + 5;
  printf( "a = %d", a);
endmain
```

16.6.2 function-endfunction

A function declaration consists of
- the datatype of the return value for the function
- the name of the function
- an argumentlist delimited by comma and surrounded by parenthesis. The argumentlist must include a typedeclaration and a name for each argument.

The arguments supplied by the caller will be converted to the type of the
to the type declared in the argument list. If an argument is changed inside
the function, the new value will be transferred to the caller. In this way
it is possible to return other values then the return value of the function.
A function can contain one or several return statements. The return will hand
over the execution to the caller and return the supplied value.

Example

```
function float calculate_flow(float a, float b)
  float c;
  c = a + b;
  return c;
endfunction

...
flow = korr * calculate_flow( v, 35.2);
```

16.6.3 if-else-endif

The lines between a if-endif statement will be executed if the expression in the if-statement is true. The expression should be surrounded by parentheses. If an else statement is found between the if and endif the lines between else and endif will be executed if the if-expression is false.

Example

```
if ( i < 10 && i > 5)
  a = b + c;
endif

if ( i < 10)
  a = b + c;
else
  a = b - c;
endif
```

16.6.4 while-endwhile

The lines between a while-endwhile statement will be executed as long as the expression in the while-statement is true. The expression should be surrounded by parentheses.

Example

```
while ( i < 10)
  i++;
endwhile
```

16.6.5 for-endfor

The lines between a for-endfor statement will be executed as long as the middle expression in the for-statement is true. The for expression consists of three expression, delimited by semicolon and surrounded by parentheses. The first expression will be executed the before the first loop, the third will be executed after every loop, the middle is executed before every loop and if it is true, another loop is done, if false the loop is leaved.

Example

```
for ( i = 0; i < 10; i++)
  a += b;
endfor
```

16.6.6 break

A break statement will search for the next endwhile or endfor statement continue the execution at the line after.

Example

```
for ( i = 0; i < 10; i++)
  a += b;
  if ( a > 100)
    break;
endfor
```

16.6.7 continue

A continue statement will search for the previous while or for statement continue the loop execution.

Example

```
for ( i = 0; i < 10; i++)
  b = my_function(i);
  if ( b > 100)
    continue;
  a += b;
endfor
```

16.6.8 goto

A goto will cause the execution to jump to a row defined by label.
The label line is terminated with colon.

Example

```
  b = attribute("MOTOR-ON.ActualValue", sts);
  if (!sts)
    goto some_error;
  ...
some_error:
  say("Something went wrong!");
```

16.6.9 include

An script include-file containing functions can be included with the #include statement. The default file extention is '.rtt_com'

Example

```
#include <my_functions>
```

16.7 Input/Output functions

Function	Description
ask	Print a question an read an answer.
printf	Formatted print.
say	Print a text.
scanf	Formatted read.

16.7.1 ask()

int ask(string question, (arbitrary type) reply)

Description

Prompts for input with supplied string.
Returns number of read tokens, 1 or 0.

Arguments

string	question	Prompt.
arbitrary type	reply	Entered reply. Can be int, float or string.

Example

```
string reply;

ask( "Do you want to continue? [y/n] ", reply);
if ( reply != "y")
  exit();
endif
```

16.7.2 printf()

int printf(string format [, (arbitrary type) arg1, (arbitrary type) arg2])

Description

Formatted print. C-syntax. Format argument and non, one or two value arguments. Returns number of printed characters.

Arguments

string	format	Format.
arbitrary type	arg1	Value argument. Optional. Can be int, float or string.
arbitrary type	arg2	Value argument. Optional. Can be int, float or string.

Example

```
printf( "Watch out!");
printf( "a = %d", a);
printf( "a = %d and str = %s", a, str);
```

16.7.3 say()

int say(string text)

Description

Prints a string.

Arguments

string	text	Text to print.

Example

```
say( "Three quarks for Muster Mark!");
```

16.7.4 scanf()

int scanf(string format , (arbitrary type) arg1)

Description

Formatted input. C-syntax.
Returns number of read characters.

Arguments

string	format	Format.
arbitrary type	arg1	Value argument. Returned. Can be int, float or string.

Example

```
scanf( "%d", i);
```

16.8 Input/Output functions

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 files.
fopen	Open a file.
fprintf	Formatted write to file.
fscanf	Formatted read from file.
translate_filename	Replace environment variables in a file name.

16.8.1 fclose()

int fclose(int file)

Description

Closes an opened file.

Arguments

int	file	file-id returned by fopen.

Example

```
int infile;
infile = fopen("some_file.txt","r");
...
fclose( infile);
```

16.8.2 felement()

string felement(int file, int number, string delimiter, string str)

Description

Extracts one element from a string of elements read from a file with the fgets() function. felement() can be used in favor of element() when the read string is larger than the string size 256. felement() can parse lines up to 1023 characters.

Arguments

int	number	the number of the element.
string	delimiter	delimiter character.

Returns

string	The extracted element.

Example

```
string elem1;
int file;
string line;

file = fopen( "my_file.txt", "r");
while( fgets( line, file))
  elem1 = felement( 1, " ");
endwhile
```

16.8.3 fgets()

int fgets(string str, int file)

Description

Reads a line from a specified file.
Returns zero if end of file.

Arguments

string	str	Read line. Returned.
int	file	file returned by fopen.

Example

```
file = fopen("some_file.txt","r");
while( fgets( str, file))
  say( str);
endwhile
fclose( file);
```

16.8.4 file_search()

int file_search(string pattern, string found_file, int pass)

Description

Search for files.
A pattern with wildcard can be specified to search for several files.
The search sequence is divided in the passes init, next and end.
At the first call pass init (1) is specified. At search of more
files with the same patter the pass next (1) is specified. The
search is closed with the pass end (2).
Returns odd status if a file is found, else even status.

Arguments

string	pattern	Name of file to search for. Can contain wild card ('*').
string	found_file	Found file.
int	pass	Pass. Init (1), next (0) or end (2).

Example

```
string pattern = "*.txt";
string found_file;
int sts;

sts = file_search(pattern, found_file, 1);
while (sts & 1)
  printf("Processing %s\n", found_file);
  ...
  sts = file_search(pattern, found_file, 0);
endwhile
file_search(pattern, found_file, 2);
```

16.8.5 fopen()

int fopen(string filespec, string mode)

Description

Opens a file for read or write.
Returns a file identifier. If the file could not be opened, zero is returned.

Arguments

string	filespec	Name of file.
string	mode	Access mode

Returns

int	File identifier, or zero on error.

Example

```
int infile;
int outfile;

infile = fopen("some_file.txt","r");
outfile = fopen("another_file.txt","w");
...
fclose( infile);
fclose( outfile);
```

16.8.6 fopen()

int fopen(string filespec, string mode)

Description

Opens a file for read or write.
Returns a file identifier. If the file could not be opened, zero is returned.

Arguments

string	filespec	Name of file.
string	mode	Access mode

Returns

int	File identifier, or zero on error.

Example

```
int infile;
int outfile;

infile = fopen("some_file.txt","r");
outfile = fopen("another_file.txt","w");
...
fclose( infile);
fclose( outfile);
```

16.8.7 fprintf()

int fprintf(int file, string format [, (arbitrary type) arg1,
 (arbitrary type) arg2])

Description

Formatted print on file. C-syntax. Format argument and non, one or two value arguments.
Returns number of printed characters.

Arguments

int	file	File id returned by fopen.
string	format	Format.
arbitrary type	arg1	Value argument. Optional. Can be int, float or string.
arbitrary type	arg2	Value argument. Optional. Can be int, float or string.

Example

```
int outfile;
outfile = fopen( "my_file.txt", "w");
if (!outfile)
  exit();
fprintf( outfile, "Some text");
fprintf( outfile, "a = %d", a);
fclose( outfile);
```

16.8.8 fscanf()

int fscanf(int file, string format , (arbitrary type) arg1)

Description

Formatted read from file. C-syntax.
Returns number of read characters.

Arguments

int	file	File id.
string	format	Format.
arbitrary type	arg1	Value argument. Returned. Can be int, float or string.

Example

```
int file;
int i;

file = fopen( "my_file.txt", "r");
if (file)
  fscanf( file, "%d", i);
  fclose( file);
endif
```

16.8.9 translate_filename()

string translate_filename(string fname)

Description

Replace environment variables in filename.

Arguments

string	fname	A filename.

Returns

string	String with expanded env variables.

Example

```
string fname1 = "$pwrp_db/a.wb_load";
string fname2;
fname2 = translate_filename( fname1);
```

16.9 String functions

Function	Description
edit	Remove superfluous spaces and tabs.
element	Extract one element from a string.
extract	Extract a substring from a string.
sprintf	Formatted print to a string.
strchr	Return the first occurence of a character in a string.
strrchr	Return the last occurence of a character in a string.
strlen	Calculate the length of a string.
strstr	Return the first occurence of a substring in a string.
tolower	Convert string to lower case.
toupper	Convert string to upper case.

16.9.1 edit()

string edit(string str)

Description

Removes leading and trailing spaces and tabs, and replaces multiple tabs and spaces with a single space.
Returns the edited string.

Arguments

string	str	string to be edited.

Example

```
collapsed_str = edit(str);
```

16.9.2 element()

string element(int number, string delimiter, string str)

Description

Extracts one element from a string of elements.
Returns the extracted element.

Arguments

int	number	the number of the element.
string	delimiter	delimiter character.
string	str	string of elements.

Example

```
string str = "mary, lisa, anna, john";
string elem1;
elem1 = element( 1, ",", str);
```

16.9.3 extract()

string extract(int start, int length, string str)

Description

Extracts the specified characters from the specified string.
Returns the extracted characters as a string.

Arguments

int	start	start position of the first character. First character has position 1.
int	length	number of characters to be extracted.
string	str	string from which characters should be extracted.

Example

```
extracted_str = extract( 5, 7, str);
```

16.9.4 sprintf()

int sprintf(string str, string format [, (arbitrary type) arg1, (arbitrary type) arg2])

Description

Formatted print to buffer. C-syntax. Format argument and non, one or two value arguments. Returns number of printed characters.

Arguments

string	str	String to print to.
string	format	Format.
arbitrary type	arg1	Value argument. Optional. Can be int, float or string.
arbitrary type	arg2	Value argument. Optional. Can be int, float or string.

Example

```
string str;
int items;

sprintf( str, "Number of items: %d", items);
```

16.9.5 strchr()

int strchr(string str, string c)

Description

Return the first occurence of a character in a string.

Arguments

string	str	String to search in.
string	c	Character to search for.

Returns

int	Index for first occurence of character. First character has index 1. Returns zero if the character is not found.

Example

```
string str = "index.html";
int idx;

idx = strchr( str, ".");
```

16.9.6 strrchr()

int strrchr(string str, string c)

Description

Return the last occurence of a character in a string.

Arguments

string	str	String to search in.
string	c	Character to search for.

Returns

int	Index for last occurence of character. First character has index 1. Returns zero if the character is not found.

Example

```
string str = "/usr/local/pwrrt";
int idx;

idx = strrchr( str, "/");
```

16.9.7 strlen()

int strlen(string str, string c)

Description

Calculates the length of a string.

Arguments

string	str	String to calculate length for.

Returns

int	Length of string.

Example

```
string str = "/usr/local/pwrrt";
int len;

len = strlen( str);
```

16.9.8 strstr()

int strstr(string str, string substr)

Description

Return the first occurence of a substring in a string.

Arguments

string	str	String to search in.
string	substr	Substring to search for.

Returns

int	Index for first occurence of substring. First character has index 1. Returns zero if the substring is not found.

Example

```
string str = "index.html";
int idx;

idx = strstr( str, ".html");
```

16.9.9 toupper()

string toupper(string str)

Description

Convert string to upper case.

Arguments

string	str	String to convert.

Returns

string	String in upper case.

Example

```
string str1 = "Buster Wilson";
string str2;
str2 = toupper( str);
```

16.9.10 tolower()

string tolower(string str)

Description

Convert string to lower case.

Arguments

string	str	string to convert.

Returns

string	string in lower case.

Example

```
string str1 = "Buster Wilson";
string str2;
str2 = tolower( str);
```

16.10 System functions

Function	Description
exit	Exit script.
get_namespace	Get current namespace.
set_namespace	Set namespace for extern variables.
system	Execute shell command.
terminate	Terminate the process.
time	Get system time.
tzset	Set time zone.
verify	Print executed lines.

16.10.1 exit()

int exit()

Description

Terminates executions of the file.

Example

```
exit();
```

16.10.2 get_namespace()

string get_namespace()

Description

Returns the current namespace.

Returns

string	Current namespace.

Example

```
string current_namespace;
current_namespace = get_namespace();
```

16.10.3 set_namespace()

set_namespace(string namespace)

Description

Set namespace for extern variables.
The maximum size of the namespace is 31 characters. If the length of the input string exceeds the maximum size, the last 31 characters of the string is used.

Arguments

string	namespace	New namespace.

Example

```
set_namespace(p1);
```

16.10.4 system()

int system(string cmd)

Description

Execute a shell command.

Arguments

string	cmd	Shell command to execute.

Returns

int	The return value is -1 on error and the return status of the command otherwise.

Example

```
string cmd;

cmd = "firefox http://www.proview.se";
system( cmd);
```

16.10.5 time()

string time()

Description

Returns the current time in string format.

Example

```
string t;
t = time();
```

16.10.6 terminate()

int terminate()

Descriptions

Terminate the process.

16.10.7 tzset()

string tzset(string timezone)

Description

Set time zone.

Example

```
tzset("Europe/Stockholm");
```

16.10.8 verify()

int verify([int mode])

Description

Sets or shows verification mode. If verification is on all executed lines will be displayed on the screen.
Returns the current verification mode.

Arguments

int	mode	verification on (1) or off (0). Optional.

Example

```
verify(1);
```

16.11 Array functions

Function	Description
arrayclear	Clear a dynamic array.
arraypush	Push a value to a dynamic array.
arraysize	Get size of an array.
sort	Sort an array.

16.11.1 arrayclear()

int arrayclear((arbitrary array type) array)

Description

Remove all elements of an array and set size to 0.
The array argument should be specified with square brackets.

Returns the status of the operation.

Arguments

(arbitrary array type)	array	Array name. Should be specified with square brackets.

Example

```
float temp[];
...
arrayclear(temp[]);
```

16.11.2 int arraypush((arbitrary array type) array,

Description

Add an element at the back of an array and insert the specified value in the element.
The array argument should be specified with square brackets.

Returns the status of the operation.

Arguments

(arbitrary array type)	array	Array name. Should be specified with square brackets.
(arbitrary type)	value	Value to push to the array.

Example

```
float x[];

arraypush(x[], 29.2);
```

16.11.3 int arraysize((arbitrary array type) array)

Description

Get the size of an array, i e number of elements in the array.
The array argument should be specified with square brackets.

Returns the size of the array.

Arguments

(arbitrary array type)	array	Array name. Should be specified with square brackets.

Example

```
float x[];
int size;
...
size = arraysize(x[]);
```

16.11.4 int sort((arbitrary array type) array1 [,(arb

Description

Sort an string array in alpabetical order, or an int or float array in numerical order.
Up to seven additional arrays can be specified that will be sorted the same way as the first array. These arrays should have the same size, or larger size than the first array.
The array arguments should be specified with square brackets.

Returns the status of the operation.

Arguments

(arbitrary array type)	array1	Name of array that will be sorted. Should be specified with square brackets.
(arbitrary array type)	array2 - array8	Additional optional arrays that will be sorted the same way as the first array. Should be specified with square brackets.

Example

```
string name[];
string description[];
int sts;
...
sts = sort(name[], description[]);
```

16.12 Math functions

Function	Description
cos	Cosine function.
EVEN	Check if value is even.
MAX	Get larges value.
MIN	Get smallest value.
ODD	Check of value is odd.
random	Get a random value.
sin	Sine function.

16.12.1 cos()

float cos(float angle)

Description

Returns the cosine value for an angle in degrees.

Arguments

float	angle	Angle in degrees.

Example

```
float x;
float angle = 45;

x = cos(angle);
```

16.12.2 EVEN()

int EVEN(int val)

Description

Test if a value is even.

Returns 1 if the value is even, else 0.

Arguments

int	val	Value.

16.12.3 MAX()

float MAX(float v1, float v2)

Description

Returns the largest of two values.

Arguments

float	v1	Value.
float	v2	Value.

Example

```
float f1;
float f2;
float max;

max = MAX(f1, f2);
```

16.12.4 MIN()

float MIN(float v1, float v2)

Description

Returns the smallest of two values.

Arguments

float	v1	Value.
float	v2	Value.

Example

```
float f1;
float f2;
float min;

min = MIN(f1, f2);
```

16.12.5 ODD()

int ODD(int val)

Description

Test if a value is odd.

Returns 1 if the value is odd, else 0.

Arguments

int	val	Value.

16.12.6 random()

float random(float min_value, float max_value)

Description

Returns a random value in the specified range.

Arguments

float	min_value	Min value for random value.
float	max_value	Max value for random value.

Example

```
float val;

val = random(0.0, 100.0);
```

16.12.7 sin()

float sin(float angle)

Description

Returns the sine value for an angle in degrees.

Arguments

float	angle	Angle in degrees.

Example

```
float x;
float angle = 45;

x = sin(angle);
```

16.13 Database functions

Function	Description
GetAttribute()	Get attribute value.
SetAttribute()	SGet attribute value.
GetChild()	Get object child.
GetParent()	Get object parent.
GetNextSibling()	Get object sibling.
GetClassList()	Get first instance of a class.
GetClassListAttrRef()	Get first attribute instance of a class.
GetNextObject()	Get next instance of a class.
GetNextAttrRef()	Get next attribute instance of a class.
GetObjectClass()	Get class of an object.
GetNodeObject()	Get node object.
GetRootList()	Get first object in root list.

16.13.1 GetAttribute()

(variable type) GetAttribute(string name [, int status])

Description

Get the value of the specified attribute. The returned type is dependent of the attribute type. The attribute will be converted to int, float or string.

Arguments

string	name	name of the attribute to be fetched.
int	status	status of operation. Returned. If zero, the attribute could not be fetched. Optional.

Example

```
int alarm;
int sts;

alarm = GetAttribute("Roller-Motor-Alarm.ActualValue");
on = GetAttribute("Roller-Motor-On.ActualValue", sts);
if ( !sts)
  say("Could not find motor on attribute!");
```

16.13.2 SetAttribute()

int SetAttribute(string name, (arbitrary type)value [, int publicwrite])

Description

Set the value of the specified attribute.

To set the value of a normal attribute the RtWrite privilege is required. It is though possible to write to attributes defined as PublicWrite without this privilege, if the publicwrite argument is added. PublicWrite attribute can be found in the PublicAv, PublicIv and PublicDv classes.

Returns the status of the operation.

Arguments

string	name	name of the attribute to write to.
<arbitrary type)	value	the value to set.
int	publicwrite	if 1, write to an attribute defined as PublicWrite is allowed also without the RtWrite privilege.

Example

```
float value = 22.2;
int sts;

sts = SetAttribute("Roller-Motor-Reference.ActualValue", value);
if ( !(sts & 1))
  printf( "SetAttribute error %d\n", sts);
endif
```

16.13.3 GetChild()

string GetChild(string name)

Description

get the first child of an object. The next children can be fetched with GetNextSibling().
Returns the name of the child. If no child exists a null-string is returned

Arguments

string	name	name of object.

Example

```
string child;

child = GetChild("Roller-Motor");
```

16.13.4 GetParent()

string GetParent(string name)

Description

Get the parent of an object.
Returns the name of the child. If no parent exists a null-string is returned.

Arguments

string	name	name of object.

Example

```
string parent;

parent = GetChild("Roller-Motor");
```

16.13.5 GetNextSibling()

string GetNextSibling(string name)

Description

Get the next sibling of an object.
Returns the name of the sibling. If no next sibling exists a null-string is returned.

Arguments

string	name	name of object.

Example

```
string name;
int not_first;

name = GetChild("Rt");
not_first = 0;
while ( name != "")
  if ( !not_first)
    create menu/title="The Rt objects"/text="'name'"/object="'name'"
  else
    add menu/text="'name'"/object="'name'"
  endif
  not_first = 1;
  name = GetNextSibling(nname);
endwhile
if ( !not_first )
  MessageError("No objects found");
```

16.13.6 GetClassList()

string GetClassList(string class)

Description

Get the first object of a specified class. The next object of the class can be fetched whith GetNextObject().
Returns the name of the first object. If no instances of the class exist, a null-string is returned.

Arguments

string	name	name of class.

Example

```
string name;

name = GetClassList("Dv");
```

16.13.7 GetClassListAttrRef()

string GetClassListAttrRef(string class)

Description

Get the first object or attribute object of a specified class. The next attribute object of the class can be fetched whith GetNextAttrRef().
Returns the name of the first object or attribute object. If no instances of the class exist, a null-string is returned.

Arguments

string	name	name of class.

Example

```
string name;

name = GetClassListAttrRef("Dv");
```

16.13.8 GetNextObject()

string GetNextObject(string name)

Description

Get the next object in a classlist.
Returns the name of the object. If no next object exists, a null-string is returned.

Arguments

string	name	name of object.

Example

```
string name;

name = GetClassList("Di");
while ( name != "")
  printf("Di object found: %s", name);
  name = GetNextObject(name);
endwhile
```

16.13.9 GetNextAttrRef()

string GetNextAttrRef(string name)

Description

Get the next object or attribute object in a classlist.
Returns the name of the attribute object. If no next object exists,
a null-string is returned.

Arguments

string	classname	name of class.
string	name	name of attribute object.

Example

```
string name;

name = GetClassListAttrRef("Di");
while ( name != "")
  printf("Di object found: %s", name);
  name = GetNextAttrRef("Di", name);
endwhile
```

16.13.10 GetObjectClass()

string GetObjectClass(string name)

Description

Get the class of an object.
Returns the name of the class.

Arguments

string	name	name of object.

Example

```
string class;

class = GetObjectClass("Motor-Enable");
```

16.13.11 GetNodeObject()

string GetNodeObject()

Description

Get the node object.
Returns the name of the node object.

Example

```
string node;
node = GetNodeObject();
```

16.13.12 GetRootList()

string GetRootList()

Description

Get the first object in the root list.
Returns the name of the root object. The next object in the root list can be fetched with GetNextSibling().

Example

```
string name;

name = GetRootList();
while( name != "")
  printf( "Root object found: %s", name);
  name = GetNextSibling(name);
endwhile
```

16.14 Miscellaneous functions

Function	Description
CutObjectName()	Cut off an object name.
ConfirmDialog()	Open a confirm dialog.
TextDialog()	Open text info window.
MessageError()	Print error message.
MessageInfo()	Print info message.
GetCurrentText()	Get selected text.
GetCurrentObject()	Get selected object.
get_pwr_config()	Get configuration values.
get_node_name()	Get node name.
getmsg()	Get status text.
EVEN()	Check if value is even.
ODD()	Check if value is odd.
get_language()	Get the current language
GetUser()	Get the current user.
GetPrivileges()	Get the privileges for the current user.
GetGraphInstance()	Get the instance object for a graph.
GetGraphInstanceNext()	Get the instance object for next graph.
SetSubwindow()	Set graph for a Ge Window objekt.
Quit()	Quit the operator environment.

16.14.1 CutObjectName()

string CutObjectName(string name, int segments)

Description

Cut the first segments of an object name.
Returns the last segments of an object name. The number of segments left is specified by the second argument

Arguments

string	name	Path name of object.
int	segments	Number of segments that should be left.

Example

```
string path_name;
string object_name;

path_name = GetChild("Rt-Motor");
object_name = CutObjectName( path_name, 1);
```

16.14.2 ConfirmDialog()

int ConfirmDialog(string title, string text)

Description

Display a confirm dialog box.
Returns 1 if the yes-button is pressed, 0 if the no-button i pressed.

Arguments

string	title	Title.
string	text	Confirm text.

Example

```
if ( ! ConfirmDialog( "Confirm", "Do you really want to..."))
  printf( "Yes is pressed\n");
else
  printf( "No is pressed\n");
endif
```

<TOPIC> textdialog() <style> function
TextDialog()

int TextDialog(string title, string text, [int image])

Description

Display a dialog with a text and an image of type error, warning, info
or question.

Arguments

string	title	Title.
string	text	Text.
int	image	Type of image, 1 error, 2 warning, 3 info, 4 question.

Example

```
TextDialog( "Error", "Motor is already started", 1);
```

16.14.3 MessageError()

string MessageError(string message)

Description

Print an rtt error message on the screen.

Example

```
MessageError("Something went wrong");
```

16.14.4 MessageInfo()

string MessageInfo(string message)

Description

Print an rtt info message on the screen.

Example

```
MessageInfo("Everything is all right so far");
```

16.14.5 GetCurrentText()

string GetCurrentText()

Description

Get the text of the current menu item or update field.

Example

```
string text;

text = GetCurrentText();
```

16.14.6 GetCurrentObject()

string GetCurrentObject()

Description

Get the object associated with the current menu item.
If no object is associated, a null-string i returned.

Example

```
string object;

object = GetCurrentObject();
```

16.14.7 get_pwr_config()

string get_pwr_config(string name)

Description

Get the value of a configuration variable.
Configuration values are set in /etc/proview.cnf.
Returns the value of the configuration variable.

Example

```
string id;

id = get_pwr_config( "qcomBusId");
```

16.14.8 get_node_name()

string get_node_name()

Description

Get the host name for the current node.
Returns the host name.

Example

```
name = get_node_name();
```

16.14.9 getmsg()

string getmsg(int status)

Description

Get the corresponding text for a status variable.
Returns the text.

Example

```
msg = getmsg(sts);
```

16.14.10 EVEN()

int EVEN(int sts)

Description

Check is an integer is even.
Returns 1 if even and 0 if odd.

Example

```
sts = SetAttribute( "Pump-V1-Switch.Description", "Valve switch open");
if ( EVEN(sts))
  printf("Couldn't set attribute\n");
endif
```

16.14.11 ODD()

int ODD(int sts)

Description

Check is an integer is odd.
Returns 1 if odd and 0 if even.

Example

```
sts = SetAttribute( "Pump-V1-Switch.Description", "Valve switch open");
if ( ODD(sts))
  printf("Set operation successful\n");
endif
```

16.14.12 get_language()

string get_language()

Description

Get the current language.

Returns

string	The current langage, e.g en_us, sv_se, de_de.

Example

```
string lng;

lng = get_language();
if ( lng == "sv_se")
  create opmenu/name="Funktioner-Bilder-Översikt"/command="open graph overview"
endif
if ( lng == "en_us")
  create opmenu/name="Functions-ProcessGraphs-Overview"/command="open graph overvie
endif
```

16.14.13 GetUser()

string GetUser()

Description

Get the current user.

Example

```
string user;

user = GetUser();
```

16.14.14 GetPrivileges()

int GetPrivileges()

Description

Get the privileges for the current user.

Example

```
int priv;

priv = GetPrivileges();
if ( priv & 4)
  # Has system privilege
  ...
endif
```

16.14.15 GetGraphInstance()

string GetGraphInstance(string graph)

Description

Get the instance object for an open object graph.
Returns the instance object, or an empty string if the
graph is node open.

Arguments

string	graph	Graph file name.

Example

```
string instance;

instance = GetGraphInstance( "$pwr_exe/pwr_c_dv.pwg");
```

16.14.16 GetGraphInstanceNext()

string GetGraphInstanceNext(string graph, string previous)

Description

Get the next instance object for the specified object graph.
Used when several versions of the same object graph is open for
different objects.
Returns the instance object, or an empty string if there is
no next instance.

Arguments

string	graph	Graph file name.
string	previous	Previous instance.

Example

```
string instance;

instance = GetGraphInstance( "$pwr_exe/pwr_c_dv.pwg");
while ( instance != "")
  printf( "Instance %s\n", instance);
  instance = GetGraphInstanceNext( "$pwr_exe/pwr_c_dv.pwg", instance);
endwile
```

16.14.17 SetSubwindow()

string SetSubwindow(string graph, string window, string source, [string object, int self])

Description

View a graph in a Ge window object. This command is used on command buttons to change the content of a window object. source contains the name of the graph to be viewed. Also object graphs can be viewed by supplying the object in the object argument.
If the function is executed in the window that is replaced, the self argument should be 1.

Arguments

string	graph	Graph name.
string	window	Name of Ge window object.
string	source	pwg-file or ge scrip that should be inserted into the window.
string	object	Instance object if source is an object graph.
int	self	Should be 1 if the function is executed in the window that is replaced.

Example pwg file

```
SetSubwindow("$current", "W1", "motor1.pwg", 0, 1);
```

Example script file

```
SetSubwindow("$current", "W1", "@m1script", 0, 1);
```

16.14.18 Quit()

Quit()

Description

Quit the operator environment.

16.15 Xtt commands

All the xtt-commands is available in the script code. An rtt-command line should NOT be ended with a semicolon. Variables can be substituted in the command line by surrounding them with apostrophes.

Example

```
string name = "PUMP-VALVE-Open.ActualValue";
float  value = 2.2;
set parameter/name='name'/value='value'
```

Example

```
string name;
string parname;
int j;
int i;
for ( i = 0; i < 3; i++)
  parname = "vkv-test-obj" + (i+1);
  create obj/name='parname'
  for ( j = 0; j < 3; j++)
    name = parname + "-obj" + (j+1);
    create obj/name='name'
  endfor
endfor
```