

# The module <Vision> of subsystems "User Interfaces"

<i>Module:</i>	Vision
<i>Name:</i>	Operation user interface (QT)
<i>Type:</i>	User interfaces
<i>Source:</i>	ui_Vision.so
<i>Version:</i>	1.2.0
<i>Author:</i>	Roman Savochenko
<i>Developers:</i>	Roman Savochenko, Maxim Lysenko, Ksenia Yashina
<i>Translated:</i>	Maxim Lysenko
<i>Description:</i>	Visual operation user interface.
<i>License:</i>	GPL

## Contents table

<a href="#">The module &lt;Vision&gt; of subsystems "User Interfaces"</a>	1
<a href="#">Introduction</a>	2
<a href="#">1. Purpose</a>	3
<a href="#">2. Tool of the graphical formation of the VCA interface</a>	5
<a href="#">2.1. Styles</a>	14
<a href="#">2.2. Linkage with the dynamics</a>	15
<a href="#">3. Execution of the VCA interfaces</a>	17
<a href="#">4. Conception of basic elements (primitives)</a>	19
<a href="#">4.1. Elementary figure primitive (ElFigure)</a>	20
<a href="#">4.2. Text primitive (Text)</a>	21
<a href="#">4.3. Primitive of the form element (FormEl)</a>	22
<a href="#">4.4. Primitive of the displaying the media materials (Media)</a>	23
<a href="#">4.5. Primitive of the construction of diagrams/graphs (Diagram)</a>	24
<a href="#">4.6. Primitive of the protocol formation (Protocol)</a>	24
<a href="#">4.7. Primitive of the report formation (Document)</a>	25
<a href="#">4.8. Primitive of the box container (Box)</a>	26
<a href="#">5. Vector graphics editor</a>	27
<a href="#">5.1. Purpose</a>	27
<a href="#">5.2. Principles and functions of the graphic editor</a>	27
<a href="#">5.3. Basic principles of operation in the graphic editor</a>	29
<a href="#">5.4. Popup menu of the graphic editor</a>	32
<a href="#">5.5. Properties dialog of the elementary figure</a>	33
<a href="#">6. The overall configuration of the module</a>	35

# Introduction

Vision module provides a mechanism of the final visualization control area (VCA) into the OpenSCADA. The module is based on the multi-platform library of graphical user interface (GUI) of firm TrollTech — QT (<http://www.trolltech.com/qt>). In its work, the module uses the data of the VCA engine (module [VCAEngine](#)).

Visual control area (VCA) is an integral part of the SCADA system. It applies to the client stations with a view to providing accessible information about the object and to for the the issuance of the control actions to the object. In various practical situations and conditions the VCA, based on different principles of visualization may by applied. For example, this may be the library of widgets QT, GTK+, ~ wxWidgets or hypertext mechanisms based technologies HTML, XHTML, XML, CSS, and JavaScript, or third-party applications of visualization, realized in various programming languages Java, Python, etc. Any of these principles has its advantages and disadvantages, the combination of which could become an insurmountable obstacle to the use of VCA in a practical case. For example, technologies like the QT library can create highly-reactive VCA, which will undoubtedly important for the operator station for control of technological processes (TP). However, the need for installation of that client software in some cases may make using of it impossible. On the other hand, Web-technology does not require installation on client systems and is extremely multi-platform (it is enough to create a link to the Web-server at any Web-browser) that is most important for various engineering and administrative stations, but the responsiveness and reliability of such interfaces is lower that actually eliminates the using of them at the operator of the TP stations.

OpenSCADA system has extremely flexible architecture that allows you to create external interfaces, including user and in any manner and for any taste. For example, the system configuration OpenSCADA as now available as by means of the QT library, and also the Web-based.

At the same time creation of an independent implementation of the VCA in different basis may cause the inability to use the configuration of one VCA into another one. That is inconvenient and limited from the user side, as well as costly in terms of implementation and follow-up support. In order to avoid these problems, as well as to create as soon as possible the full spectrum of different types of VCA [проект создания концепции СБУ](#) is established. The result of this project — the direct visualization module (based on the library QT), direct visualization module [WebVision](#) and VCA engine [VCAEngine](#).

# 1. Purpose

This module of the direct visualization of the VCA is designed for the formation and execution of VCA interfaces among the graphic library QT.

The final version of the VCA module, built on the basis of this module, will provide:

- three levels of complexity in the formation of visualization interface which let organically to develop and apply the tools of the methodology from simple to complex:
  - formation from the template frames through the appointment of the dynamics (without the graphical configuration);
  - graphical formation of new frames through the use of already made visualization elements from the library (mimic panel);
  - formation of new frames, template frames of the visualization elements in the libraries.
- building of the visualization interfaces of various complexity, ranging from simple flat interfaces of the monitoring and finishing with the full-fledged hierarchical interface used in SCADA systems;
- providing of the different ways of formation and configuration of the user interface, based on different graphical interfaces (QT, Web, Java ...) and also through the standard management interface of OpenSCADA system;
- change of dynamics in the process of execution;
- building of the new template frames on the user level and the formation of the frames libraries, specialized for the area of application (eg the inclusion of frames of parameters, graphs and other items linking them to each other) in accordance with the theory of secondary using and accumulation;
- building of the new user elements of the visualization and the formation of the libraries of frames, specialized for the area of application in accordance with the theory of secondary using and accumulation;
- description of the logic of new template frames and user visualization elements as with the simple links, and also with the laconic, a full-fledged programming language;
- the possibility of the inclusion of the functions (or frames of computing of the functions) of the object model of OpenSCADA to the user elements of the visualization, actually linking the presentation of the algorithm of computing (for example, by visualizing the library of models of devices of TP for following visual modeling TP);
- separation of user interfaces and interfaces of visualization of data provides building the user interface in a single environment, and performance of it in many others (QT, Web, Java ...);
- the possibility to connect to the performing interface for monitoring and corrective actions (for example, while operator training and control in real time for his actions);
- Visual building of various schemes with the superposition of the logical links and the subsequent centralized execution in the background (visual construction and performance of mathematical models, logic circuits, relay circuits and other proceedings);
- providing of the the functions of the object API to the OpenSCADA system, it can be used to control the properties of the visualization interface from the user procedures;
- building of the servers of frames, of elements of the visualization and of the project of the interfaces of the visualization with the possibility to serve the great number of the client connections;
- simple organization of client stations in different basis (QT, Web, Java ...) with the connection to the central server;
- full mechanism of separation of privileges between the users which allows to create and execute projects with the various rights of access to its components;
- adaptive formation of alarms and notifications, with the support of different ways of notification;
- support of the user formation of the palettes and font preferences for the visualization of the interface;
- support of the user formation of maps of the events under the various items of equipment management and user preferences;

- support for user profiles, allowing to define various properties of the visualization interface (colors, font characteristics, the preferred maps of events);
- flexible storage and distribution of libraries of widgets, frames, and projects of the visualization interfaces in the databases, supported by OpenSCADA; actually users need only to register the database with data.

## 2. Tool of the graphical formation of the VCA interface

Development of the VCA interface is performed in a single window, realizing many documents interface (MDI) interface (Fig. 2.a). This approach allows you to simultaneously edit multiple frames of various sizes. The following mechanisms for managing the development are used: toolbars, menus and context menus. Most actions are duplicated by different mechanisms, that allows you to quickly find the tool by the preferred method. Navigational interfaces are implemented by the attached windows. Configuration of the toolbars and attached windows is saved on exit and restored at startup that lets you to customize the interface for yourself.

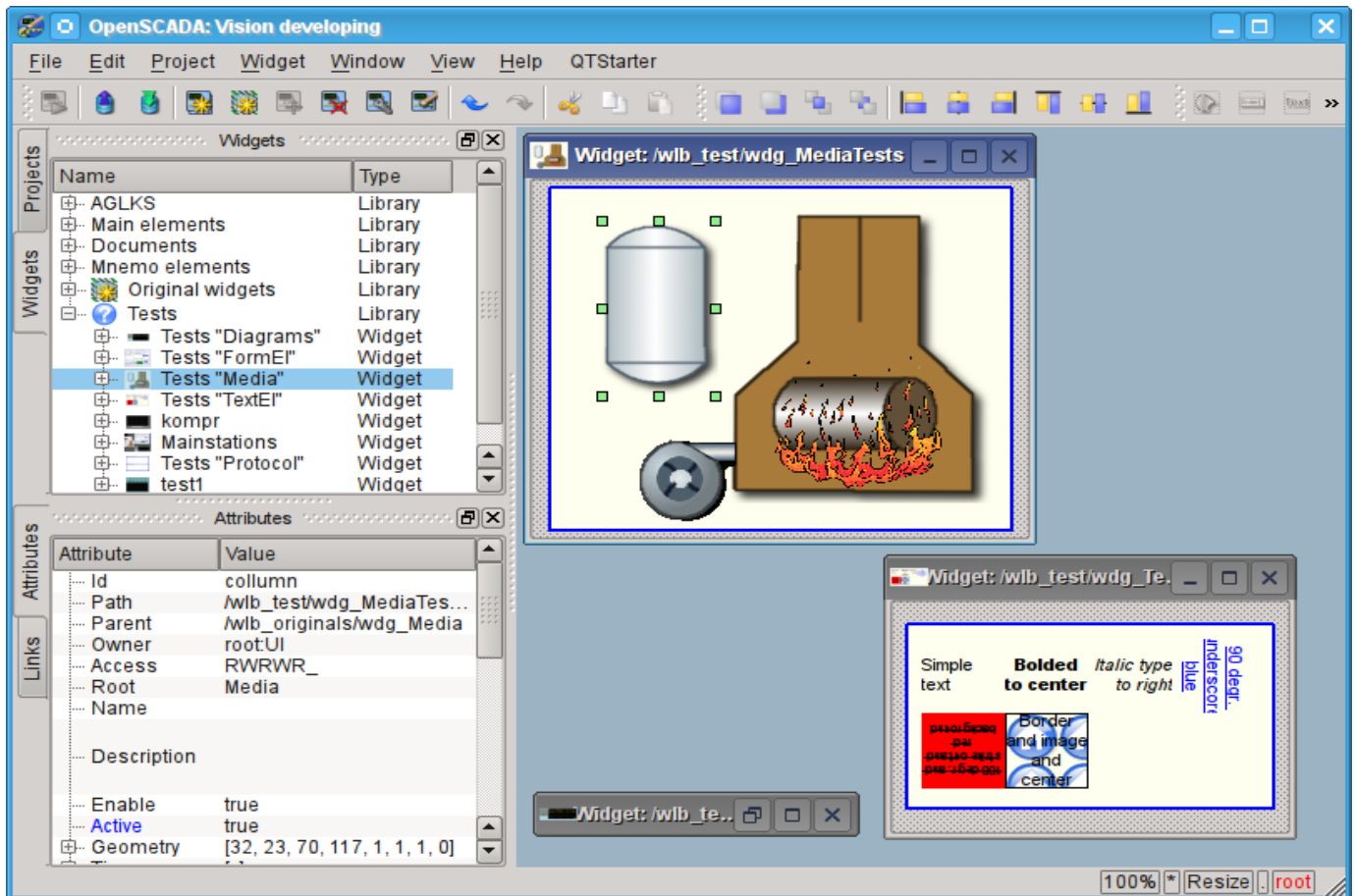


Fig.2.a. The window of the VCA interface development.

Access to major components of the VCA is made by attached windows, in the Figure 2.a these windows are shown on the left side. These windows contain:

- Tree of the widget libraries. Using the navigator you can quickly find the needed widget or library and to do with them necessary operations. The following operations are implemented: add, delete, copy, settings of the widgets and libraries, as well as cleaning and visual editing of the widget. For adaptive management the context menu is supported with the following items:
  - "New library" — creation of the new library.
  - "Add visual item" — adding of the visual element to the library.
  - "Delete visual item" — deleting of the visual element from the library.
  - "Visual item changes clear" — cleaning of the visual element with inheritance of modified properties or setting them by default.
  - "Visual item properties" — configuration of the visual element.
  - "Visual item edit" — visual editing of the element.
  - "Visual item cut" — cut/move of the visual element at the time of paste.
  - "Visual item copy" — copy of the visual element at the time of paste.
  - "Visual item paste" — paste of the visual element.
  - "Load from DB" — uploading the data of the visual element from the database.
  - "Save to DB" — saving data of visual element to the database.

- "Refresh libraries" — performs rereading of the configuration and composition of the libraries of the data model.
- The tree of pages of the project. Provides the mechanism for "Drag and drop" for creation of the user frames based on the elements of libraries. In order to provide the adaptive management the context menu is supported with the following items:
  - "Run project execution" — starting of the execution of the chosen project.
  - "New project" — creation of the new project.
  - "Add visual item" — adding of the visual element to the project/page.
  - "Delete visual item" — deleting of the visual element from the project/page.
  - "Visual item changes clear" — cleaning of the visual element with inheritance of modified properties or setting them by default.
  - "Visual item properties" — configuration of the visual element.
  - "Visual item edit" — visual editing of the element.
  - "Visual item cut" — cut/move of the visual element at the time of paste.
  - "Visual item copy" — copy of the visual element at the time of paste.
  - "Visual item paste" — paste of the visual element.
  - "Load from DB" — uploading the data of the visual element from the database.
  - "Save to DB" — saving data of visual element to the database.
  - "Refresh libraries" — performs rereading of the configuration and composition of the libraries of the data model.
- attributes of widgets;
- external links of widgets.

In the main space of the working window the pages of projects, frames of the widgets' libraries, user elements and elements of primitives at the time of their visual editing are placed.

At the top of the working window there is the menu. All the tools needed for development the VCA interfaces are placed in the menu. Menu has the following structure:

- "File" — General operations.
  - "Load from DB" — uploading the data of the visual element from the database.
  - "Save to DB" — saving data of visual element to the database.
  - "Close" — close the editor's window
  - "Quit" — quit from the OpenSCADA system.
- "Edit" — Editing operations of the visual elements.
  - "Make visual item changes UnDo" — make visual item last change undo.
  - "Make visual item changes ReDo" — make repeat for visual item change.
  - "Visual item cut" — cut/move of the visual element at the time of paste.
  - "Visual item copy" — copy of the visual element at the time of paste.
  - "Visual item paste" — paste of the visual element.
- "Project" — Operations over the projects.
  - "Run project execution" — starting of the execution of the chosen project.
  - "New project" — creation of the new project.
  - "Add visual item" — adding of the visual element to the project.
  - "Delete visual item" — deleting of the visual element from the project.
  - "Visual item changes clear" — cleaning of the visual element with inheritance of modified properties or setting them by default.
  - "Visual item properties" — configuration of the visual element.
  - "Visual item edit" — visual editing of the element.
- "Widget" — Operations over the widgets and the libraries of widgets.
  - "New library" — creation of the new library.
  - "Add visual item" — adding of the visual element to the library.
  - "Delete visual item" — deleting of the visual element from the library.
  - "Visual item changes clear" — cleaning of the visual element with inheritance of modified properties or setting them by default.
  - "Visual item properties" — configuration of the visual element.
  - "Visual item edit" — visual editing of the element.
  - "View" — Management of the arrangement of visual elements on the frame.

- "Rise widget" — rising the widget to the top.
- "Lower widget" — lowering the widget to the very bottom.
- "Up widget" — to rise the widget above.
- "Down widget" — to lower the widget below.
- "Align to left" — alignment of the widget to the left.
- "Align to vertical center" — alignment of the widget vertically to the center.
- "Align to right" — alignment of the widget to the right.
- "Align to top" — alignment of the widget to the top.
- "Align to horizontal center" — horizontal alignment of the widget in the center.
- "Align to bottom" — alignment of the widget to the bottom.
- "Library: {Name of the library}" — menu items to access the frames/widgets in the library.
- "Window" — Management of the windows of MDI-interface.
  - "Close" — to close the active window.
  - "Close all" — to close all the windows.
  - "Tile" — to tile all the windows for visibility at the same time.
  - "Cascade" — to cascade all the windows.
  - "Next" — to activate the next window.
  - "Previous" — to activate the previous window.
  - "Widget: {Name of the widget}" — items of activation of the specific window.
- "View" — Management of the visibility of the working window and the toolbars on it.
  - "Visual items toolbar" — visual element toolbar.
  - "Widgets view functions" — the toolbar for management of the visibility and arrangement of widgets on the panels.
  - "Elementary figures tools" — Additional toolbar for the editing the primitive of elementary figures ("ElFigure").
  - "Projects" — attached window of management of projects' tree.
  - "Widgets" — attached window of management of widgets' libraries tree.
  - "Attributes" — attached window of the attributes' manager.
  - "Links" — attached window of the links' manager.
  - "Library: {Name of the library}" — management of the visibility of widgets' libraries toolbars.
- "Help" — Help for OpenSCADA and fro Vision module.
  - "About" — information about this module.
  - "About QT" — Information about the QT library, used by this module.
  - "What's this" — query of the description of the elements of the window's interface.

Above, under menu, or on the sides, there are the toolbars. Toolbars can be hidden or located, which is controlled in the menu item "View". The following toolbars are present:

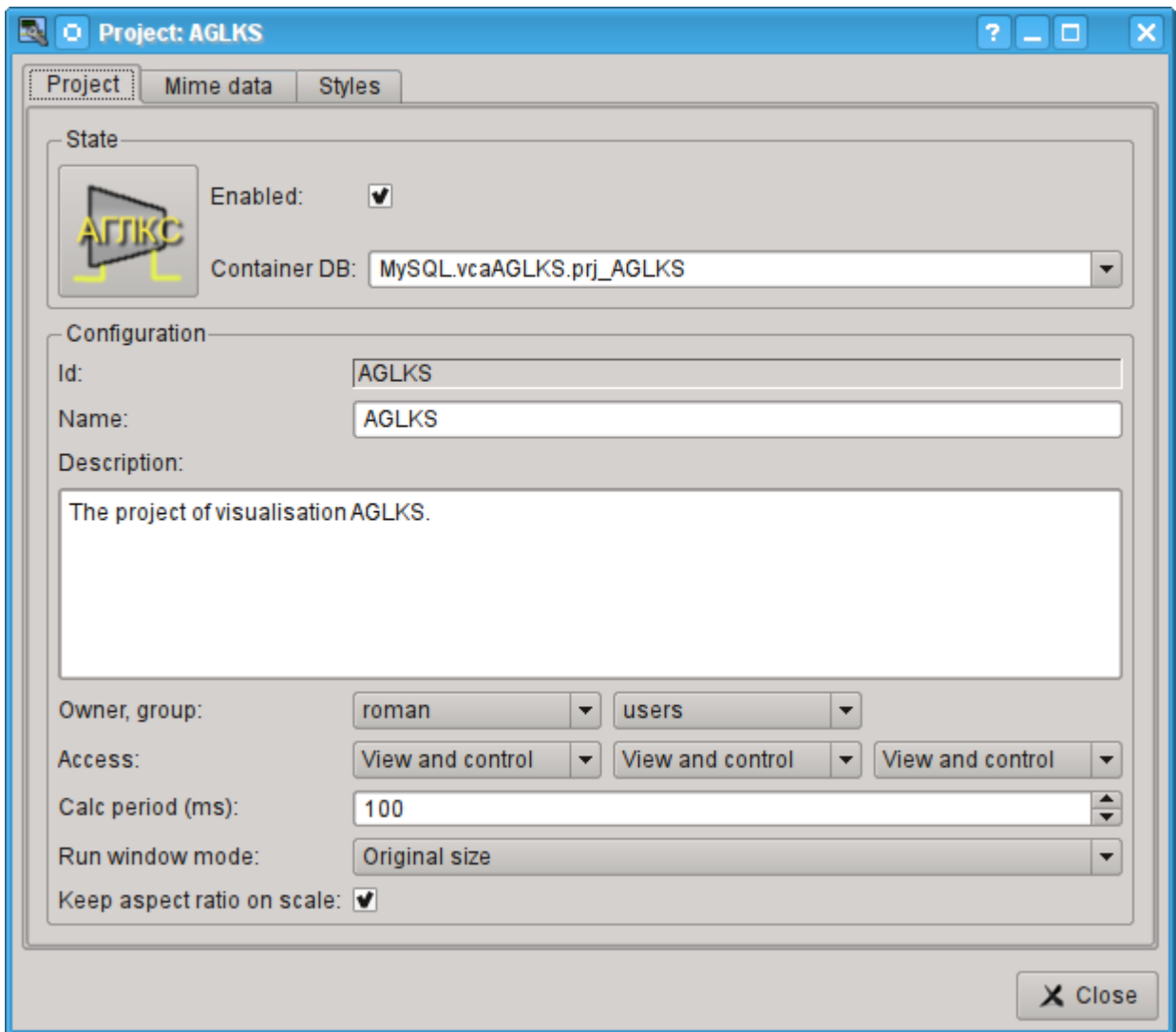
- "Visual items toolbar" — Management toolbar of the visual items:
  - "Run project execution for selected item" — runs the project for execution and activates the selected page of the project.
  - "Load item data from DB" — uploading the data of the chosen elements from the database.
  - "Save item data to DB" — saving data of chosen elements to the database.
  - "New project" — creation of the new project.
  - "New library" — creation of the new library.
  - "Add visual item" — adding of the visual element to the project.
  - "Delete visual item" — deleting of the visual element from the project.
  - "Visual item's properties" — configuration of the visual element.
  - "Visual item edit" — visual editing of the element.
  - "Make visual item changes UnDo" — make visual item last change undo.
  - "Make visual item changes ReDo" — make repeat for visual item change.
  - "Visual item cut" — cut/move of the visual element at the time of paste.
  - "Visual item copy" — copy of the visual element at the time of paste.
  - "Visual item paste" — paste of the visual element.

- "Widgets view functions" — The toolbar of visibility and arrangement management of widgets on the panels:
  - "Rise widget" — rising the widget to the top.
  - "Lower widget" — lowering the widget to the very bottom.
  - "Up widget" — to rise the widget above.
  - "Down widget" — to lower the widget below.
  - "Align to left" — alignment of the widget to the left.
  - "Align to vertical center" — alignment of the widget vertically to the center.
  - "Align to right" — alignment of the widget to the right.
  - "Align to top" — alignment of the widget to the top.
  - "Align to horizontal center" — horizontal alignment of the widget in the center.
  - "Align to bottom" — alignment of the widget to the bottom.
- "Elementary figure tools" — Additional toolbar of the editing of the elementary figures primitive ("ElFig").
  - "Cursor" — return to the cursor for the action over the figures on the widget.
  - "Add line" — adding the line to the elementary figure.
  - "Add arc" — adding the arc to the elementary figure.
  - "Add besier curve" — adding the Bézier curve to the elementary figure.
  - "Connections" — the enabling of the of connections at the elementary figure.
- "Library: {Name of the library}" — Management of the visibility of toolbars of the widget libraries. The contents of the panel depends on the contents of the library and includes call buttons of the library items.
- "QTStarter toolbar" — The toolbar, created by the module of the module of starting the QT library modules. It contains buttons to start the UI modules of OpenSCADA, based on the QT Library. With this toolbar you can open multiple copies of the windows of the module or other modules.

At the bottom of the development window of the VCA there is the status line. On the right side of the status line there are indicators of the visual scale of the edited frame, of the mode of changing of the size of the elements, of the mode of the current page of the of the VCA engine station and the user on whose behalf the development of the VCA interface is done. By double-clicking on the indicator of the user it can be changed the current user, enter the new username and password. In the main field of the status line it is displayed various information and assistance messages.

To edit the properties of the visual elements there are two dialogues. The first dialogue allows you to edit the properties of containers of visual elements (widget libraries and projects), figure 2.b. The second dialogue serves to edit the properties of the visual elements, Fig. 2.c. Changes, made in the dialogues, at once, get to the VCA engine. To save these changes to the database or restore from the database it is necessary to use the appropriate tools of the main development window.

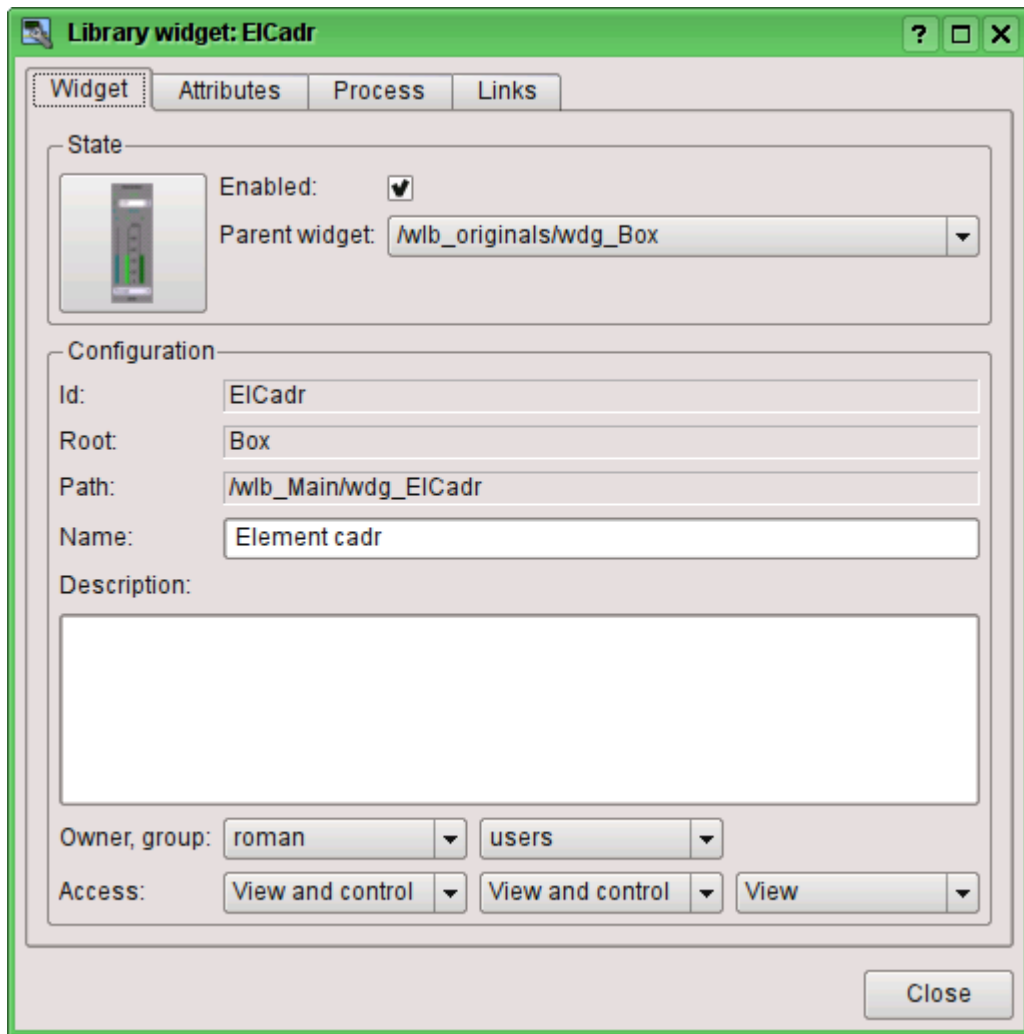




*Fig.2.b. Dialogue of the editing the properties of the containers of visual elements.*

With the help of the main tab of that dialog you can set:

- The state of the elements' container, namely: "Enabled", the database container.
- Id, name and description of the container.
- For project: user, group of users and user access, users' group and all the rest.
- For the project: the period for calculating of the project and the mode of opening the windows in the execution.

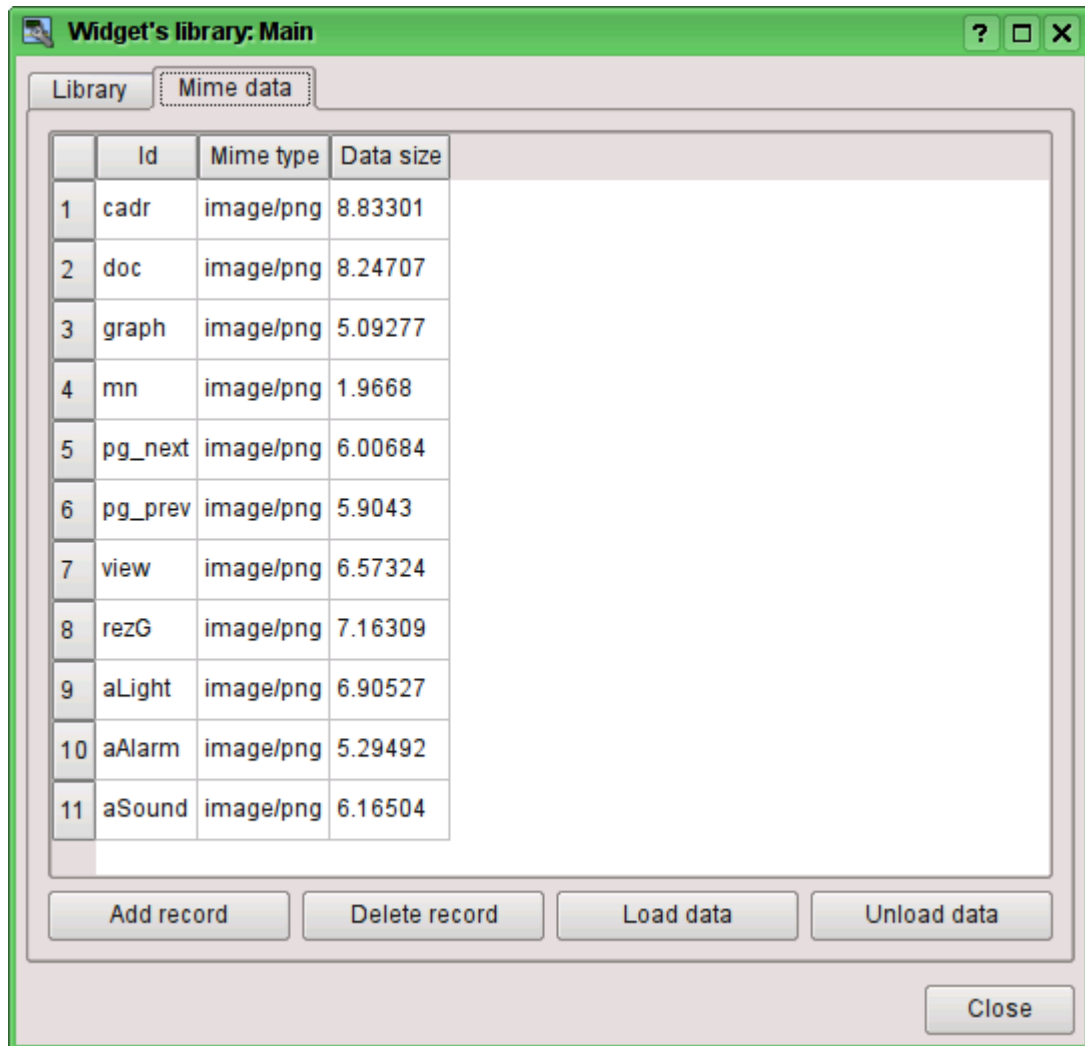


*Fig.2.c. Dialogue of editing the properties of visual elements.*

With the help of the main tab of that dialog you can set:

- The state of element, namely: "Enabled", the parent widget.
- Id, root, path, name and description of the element.
- User, group of users of the element and user access, user groups and all the rest.

Dialogue of editing the properties of the containers of visual elements contains two tabs: configuration tab of the the main parameters (Fig.2.b) and the configuration tab of the mime-data of containers (Fig. 2.d).



*Fig.2.d. Editing tab of the mime-data of the container of visual elements.*

Dialogue of the editing the properties of the visual elements contains four tabs: configuration tab of the main parameters (Fig.2.b), the tab of attributes of the element (Fig. 2.e), the tab of the processing of the element (Fig. 2.f) and the tab of links of the elements (Fig.2.g). At different levels of the hierarchy of visual elements any tabs can be available, but some are not.

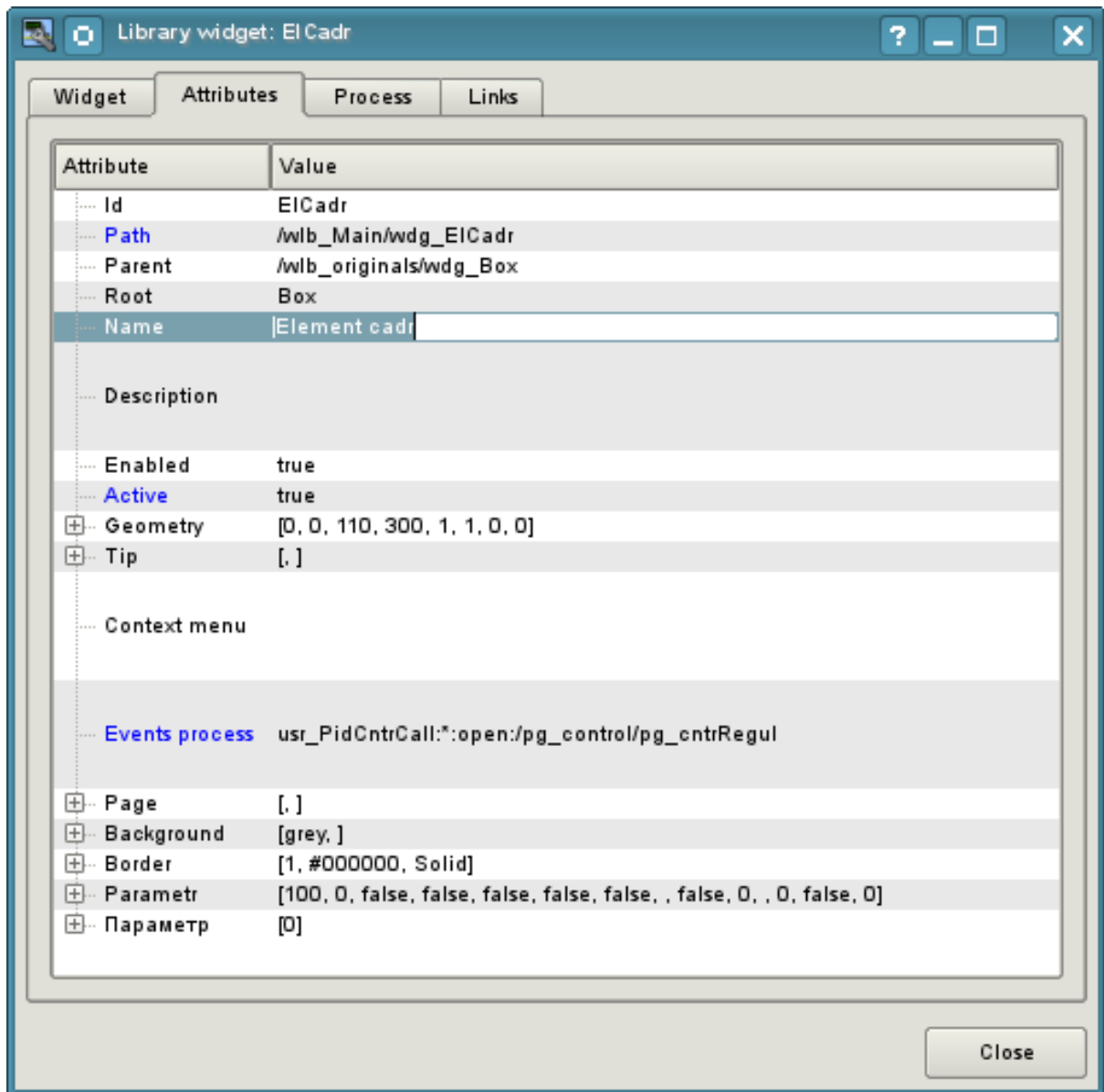


Fig.2.e. Attributes of the editing dialogue of the properties of the visual element tab.



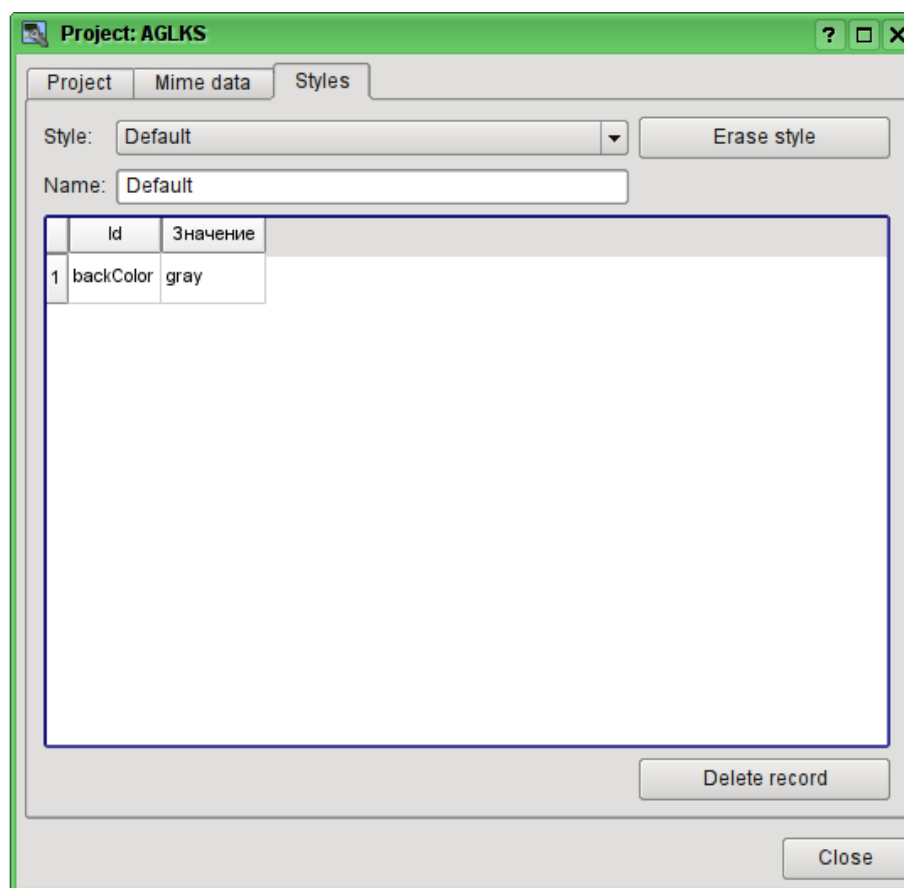
## 2.1. Styles

It is known that people can have individual characteristics in the perception of graphical information. If these features are not taken into account it is possible to get the rejection and exclusion of the user to the VC interface. Such rejection and exclusion can lead to fatal errors in the management of TP, as well as traumatize the human by the permanent working with the such interface. In SCADA systems it is accepted the agreement, which regulate the requirements for establishing a unified VC interface which is normally perceived by most of people. The people with some deviations are not taken into account.

To take this into account, and provide the ability to centrally and easily change the visual properties of the interface, the project provides the implementation of visualization interface styles manager.

User can create many styles, each of which will hold the color, font and other properties of the elements of the frame. A simple change of style will quickly transform the VC interface, and the possibility of appointing an individual style to the user will take into account his individual characteristics.

To realize this opportunity, when you create a frame, it is necessary for the properties of color, font and others set the «Config» ( of the table if the «process» tab) in the value of «From style» (Fig. 2.f). And in the parameter «Config template» to specify the identifier of the style field. Further, this field will automatically appear in the Style Manager and will be there to change. Style Manager is available on the project configuration page in the tab «Styles» (Fig. 2.1). On this tab you can create new styles, delete old ones, change the field of the style and delete unnecessary.



*Fig. 2.1. Styles tab of the configuration page of the project.*

In general the styles are available from the project level. At the level of libraries of widgets you can only define styles fields of widgets. At the project level, at the choice of style it is started the work with styles, which includes access to the fields of styles instead of direct attribute values. In fact, this means that when reading or writing a widget attribute these operations will be carried out with the corresponding field of the chosen style.

When you run the project execution it will be used the set in the project style. Subsequently, the user can select a style from the list of available ones. The user's style will be saved and used next time you run the project.

## 2.2. Linkage with the dynamics

To provide relevant data in the visualization interface the data of subsystems "Data acquisition (DAQ)" must be used. The nature of these data as follows:

1. parameters that contain some number of attributes;
2. attributes of the parameter can provide information of four types: Boolean, Integer, Real and String;
3. attributes of the parameter can have their history (archive);
4. attributes of the parameter can be set to read, write, and with full access.

Considering the first paragraph it is necessary to allow the possibility of the group of destination links. To do this we use the conception of [of the logic level](#).

In accordance with paragraph 2, links provide transparent conversion of connection types and do not require special configuration.

To satisfy the opportunities for access to archives, in accordance with paragraph 3, links make check of the type of the attribute, and in the case of connection to the "Address", the address of linkage is put into the value.

In terms of the VCA, the dynamic links and configuration of the dynamics are the one process, to describe a configuration of which the tab "Processing" of the widgets is provided (Fig.2.f). The tab contains a table of configuration of the properties of the attributes of the widget and the text of calculation procedure of the widget.

In addition to configuration fields of the attributes the column "Processing" in the table is provided, for selective using of the attributes of the widgets in the computational procedure of the widget, and the columns "Configuration" and "Configuration template", to describe the configuration of links.

Column "Configuration" allows you to specify the linkage type for the attribute of the widget:

- *Constant* — in the tab of widget links the field for indication of a constant appears, for example of the special color or header for the template frames;
- *Input link* — linkage with the dynamics for a read-only;
- *Output link* — linkage with the dynamics just for the record;
- *Full link* — complete linkage with dynamic (read/write).

Column "Configuration template" makes it possible to describe the groups of dynamic attributes. For example it may be different types of parameters of subsystem "DAQ". Furthermore, in the case of correct formation of this field, the mechanism of automatically assign of the attributes with the only indication of the parameter of subsystem "DAQ" is working, which simplifies and accelerates the configuration process. The value of this column has the following format: **<Parameter>|<identifier>**, where:

- *<Parameter>* — the group of the attribute;
- *<Identifier>* — identifier of the attribute, this value is compared with the attributes of the DAQ parameters with automatic linkage, after the group link indication.

Installation of the links may be of several types, which are determined by the prefix:

- *val:* — Direct download of the value through the links mechanism. For example, link: "val:100" loads in the attribute of the widget the value of the 100. It is often used in the case of absence of end point of the link, in order to direct value indicating.
- *prm:* — Link to the attribute of the parameter or parameter, in general, for a group of attributes, of subsystem "Data acquisition". For example, the link "prm:/LogicLev/experiment/Pi/var" implements the access of the attribute of the widget to the attribute of the parameter of subsystem "Data acquisition". Sign "(+)" at the end of the address signals about successful linking and presence of the target.
- *wdg:* — Link to an attribute of another widget or a widget, in general, for a group of attributes. For example, the link "wdg:/ses\_AGLKS/pg\_so/pg\_1/pg\_ggraph/pg\_1/a\_bordColor" implements the access of the attribute of one widget to the attribute of another one. At that moment this type of link is not intended for installation by the user manually, and is installed automatically in the mode of dynamic linkage!

Processing of the links occurs at a frequency of calculating the widget in the following order:

- Receiving of the data from input links.
- The implementation of calculating of the script.
- Transmission of the values by the output links.

Fig. 2.g presents the tab with the possibility of group and individual assignment of attributes.

When the widget that contains the configuration of links is placed to the container of widgets, all links of the source widget is added to the list of resulting links of the widgets' container.

The aforesaid shows that the links are set by the user in the configuration interface. However, for the possibility of creation of the frames for general use, with the function of providing detailed data of various sources of the same type, a dynamic linkage mechanism is necessary. Such an mechanism is provided through a reserved key identifier "<page>" of the group of attributes of links in the frames of general purpose and dynamic linkage with the identifier "<page>" in the process of opening of the frame of general purpose by means of the signal from another widget.

Lets examine the example when we have the frame of general-purpose "Control panel of graph" and a lot of "Graphs" in different tabs. "Control panel of graph" has links with the templates:

- tSek -> "<page>|tSek"
- tSize -> "<page>|tSize"
- trcPer -> "<page>|trcPer"
- valArch -> "<page>|valArch"

At the same time, each widget "Graph" has the attributes tSek, tSize, trcPer and valArch. In the case of a calling of the opening signal of "Control panel of graph" from any widget "Graph" it is happening the linkage of the attributes of the "Control panel of graph" in accordance with the attribute specified in the template with the attribute of the widget "Graph". As a result, all changes in the "Control panel of graph" will be displayed on the graph by means of the link.

In the case of presence of external links to the parameters of subsystem "Data acquisition" in the widget "Graph", the links of "Control panel of graph" will be installed on an external source. In addition, if in the "Control panel of graph" will be declared the links to the missing attributes directly in the widget "Graph", it will be made the search for the availability of such attributes from an external source, the first to which the link is directed, performing, thus, the addition of missing links.

To visualize this mechanism the table 2.2 is cited.

**Table 2.2.** The mechanism of the dynamic linkage.

Attributes of the "Control panel of graph" (the template of dynamic linkage)	"Graph" attributes	Attributes of an external "Parameter"	The resulting link or an value of the linking attribute
tSek (<page> tSek)	tSek	-	"Graph".tSek
tSize (<page> tSize)	tSize	-	"Graph".tSize
trcPer (<page> trcPer)	trcPer	-	"Graph".trcPer
valArch (<page> valArch)	valArch	-	"Graph".valArch
var (<page> var)	var	var	"Parameter".var
ed (<page> ed)	-	ed	"Parameter".ed
max (<page> max)	-	-	EVAL
min (<page> min)	-	-	EVAL



### 3. Execution of the VCA interfaces

Execution of the VCA interface is to run a new project session or connect to the existing one on the level of VCA engine. Then the module of direct visualization represents and manages the data of the session. The main window mode of execution mode of this module has the form presented at Fig.3.

Update of the contents of the open pages of the visualization interface with the frequency of the project session execution. In the updating process it is performed:

- request a list of opened pages, with a sign of page modification, at the model and consistency checking of the actually opened pages to that list;
- request of the branch of the modified pages;
- update of the contents of the modified pages and their widgets, in accordance with the received modified data.

At the closure of "RunTime" window closing of the session of the project is done in the VCA engine.

The mechanism of the request of the only modified data is based on an absolute counter of the session execution. If you want to make real changes in the attributes of widgets the memorizing of the value of this counter is done, which allows the identification of modified attributes. This approach can increase productivity and reduce the load on network sharing in the case of access to the VCA engine via network.

Hierarchically the module provides an opportunity to accommodate the project pages in the main execution window (Fig.3), as well as putting them inside of the container widgets, as well as by the opening of additional windows over the main.

When you expand the main execution window, or when moving to the full-screen mode the scaling of the page content of the VCA interface is done, filling the entire space of the window and allowing to execute the projects, developed on one screen resolution, at different resolutions.

The main window consists of menu (top) status line (bottom), and the executable contents of the session between them. Menu in the execution mode is positioned as the OpenSCADA administrator tool, containing the self-system functions and it is available only to privileged users, occupying the group "root". Menu has the following structure:

- "File" — General operations.
  - "Print" — Print:
    - "Page" — page of the user interface;
    - "Diagram" — diagram on the user interface;
    - "Document" — document on the user interface.
  - "Export" — Export:
    - "Page" — page of the user interface;
    - "Diagram" — diagram on the user interface;
    - "Document" — document on the user interface.
  - "Close" — Close the editor window.
  - "Quit" — Quit from the OpenSCADA system.
- "Alarm" — Alarm quittance:
  - "Alarm level" — all alarms;
  - "Light alarm" — lighting notification;
  - "Speaker alarm" — notification with the whistle;
  - "Sound/speech alarm" — sound/speech notification.
- "View" — Display options of the project session.
  - "Full screen" — Switcher of the full screen execution mode.
- "Help" — Help through the OpenSCADA and Vision module.
  - "About" — Information about this module.
  - "About QT" — Information about the QT library, used by the module.

On the right side of the status line the indicators of the time, the current VCA engine station and users on whose behalf the VCA interface is executed, as well as the panel with the alarm quittance buttons, print and export. By double-clicking on the indicator of the user it can be changed by the typing of the new username and password, and by clicking on the quittance button — to quit alarms completely or only the

desired notification. In the main field of the status line various messages and assistance messages are displayed.

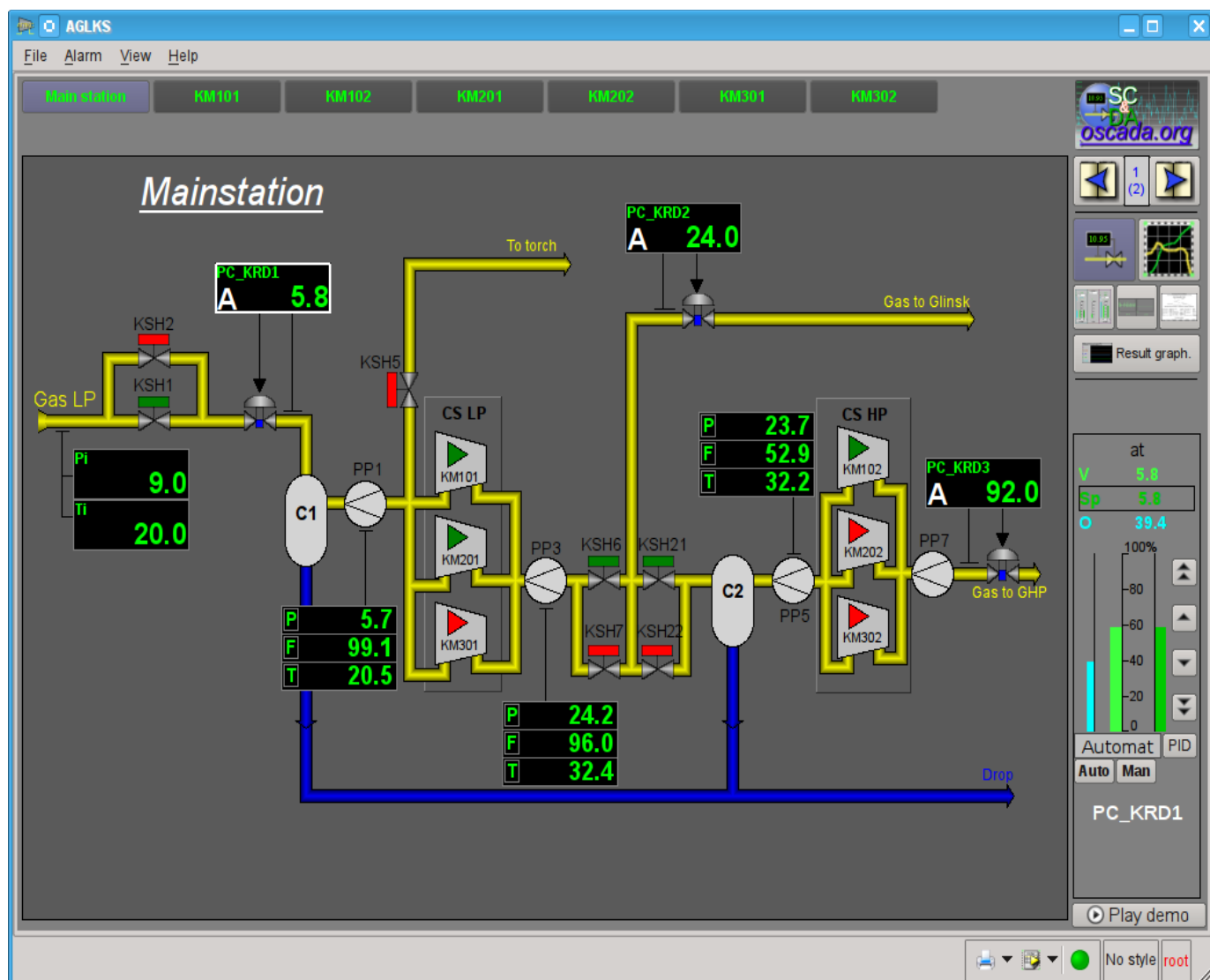


Fig.3. The main window if the execution mode.

## 4. Conception of basic elements (primitives)

In this version of that module not all the primitives' images of this project are implemented. In general the project provides the following primitives:

<b>Id</b>	<b>Name</b>	<b>Purpose</b>
ElFigure	Elementary graphic figures	<p>Primitive is the basis for drawing basic graphic shapes with their possible combinations in the single object. The support of the following elementary figures is provided:</p> <ul style="list-style-type: none"><li>• Line.</li><li>• Arc.</li><li>• Bézier curve.</li><li>• Fill of the closed circuit.</li></ul> <p>For all figures contained in the widget common properties of thickness, color, etc. are set, but this does not exclude the possibility of indicating of aforementioned attributes specific to each figure separately.</p>
FormEl	Form elements.	<p>Includes support of standard form components:</p> <ul style="list-style-type: none"><li>• Line edit.</li><li>• Text edit.</li><li>• Check box.</li><li>• Button.</li><li>• Combo box.</li><li>• List.</li><li>• Slider.</li><li>• Scroll bar.</li></ul>
Text	Text	Text element(labels). It is characterized by the type of font, color, orientation and alignment.
Media	Media	Element of representation of raster and vector images of various formats, playback of the animated images, playback of audio segments and view of video segments. Perhaps it will be useful to include the OpenGL support for it!
Diagram	Diagram	Element of the diagram with the support of the possibility of displaying multiple streams of trends and different modes of display, from minimalist to full, two-, three-dimensional, circular, etc.
Protocol	Protocol	Element of the protocol, visualizer of the system messages, with support for multiple operating modes with the different sizes and settings
Document	Document	The element of generating the reports, journals and other documentation on the basis of specified data.
Function	Function of API of the object model of OpenSCADA	Not visual, on the side of execution, widget which allows to include a computing function of the object model of OpenSCADA in the VCA.
Box	Box	Contains the mechanism for other widgets placement with the purpose of creation of new, more complex widgets and pages of final visualization.

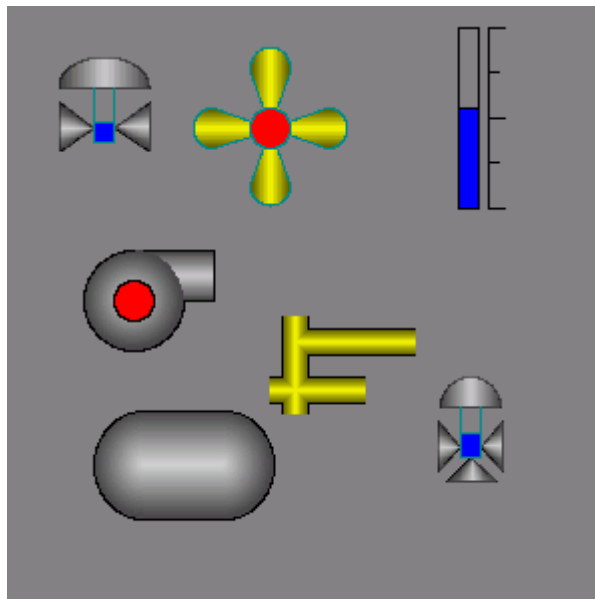
Lets examine the implementation of each primitive.

## 4.1. Elementary figure primitive (ElFigure)

Support of the following elementary figures is provided: lines, elliptical arcs, Bézier curves and fill of the closed circuit with the color and/or image. For the elementary figures the following operations are provided:

- creation/deleting of the figures;
- copying of the figure(s);
- moving and resizing of the figures by mouse and keyboard;
- possibility to connect the elementary figures to each other, getting more complex figures, for which all the properties of the source elementary figures are available;
- possibility of simultaneous movement of several figures;
- fill of the closed circuit with the color and/or image;
- generation of mouse key events at the time of the mouse-click on the filled spaces;
- scaling;
- rotation.

Fig. 4.1 shows a part of the screen with a frame containing the elementary figures.



*Fig.4.1. Realization of elementary figures in the Vision.*

The figures underlying this widget, containing the points (the start and end ones) that can be connected with the according points of other figures; and the points with the help of which the geometry of the figure can be changed.

It is possible to add the figure using the mouse:

1. Select the desired figure from the context menu.
2. Set with the left mouse-button start and end points (for line with the SHIFT key hold its orthogonal drawing is made).

The deleting of the figure(s) it is possible by pressing "Del", having selected figure(s).

The copying of the figure(s) it is possible by pressing keys "Ctrl"+"C", having selected figure(s).

Moving/resizing of the figure it is possible by using the mouse or keyboard:

1. Select the figure, by clicking on it with the left mouse button.
2. Drag (with the help of mouse or control keys) the figure or one of its control points in the desired location and release the mouse button (key).

It is possible to move several figures, selected by means of holding "Ctrl" and clicking on the desired figures (this option works when the button Connections (Connections) is disabled) or by mouse selection.

The connection of the figures with each other it is possible by the following way:

1. Press the Connections button.
2. Select one of the figures and move its start or end point to the desired start or end point of the

other figure so that it will get to the appeared circle, release the left mouse button. Connected figures are moving as well as the individual, the general point is moved for all connected figures, to which it refers(priority is given to the arc, two arcs can't be connected directly with each other ).

To fill the closed circuit from the figures it is possible with the following way:

1. Press the Connections button.
2. Create the closed circuit.
3. Make the double-click of the left mouse button inside of it.

To delete the fill of the closed circuit it is possible from the context menu of the widget; by braking the closed circuit or by double-click of the left mouse button on the already existing filled space.

Rotation of the figure is made around the center of the widget.

## 4.2. Text primitive (Text)

Support of the text element with the following properties is provided:

- Font with the properties: type/class of the font, size, bold, italic, strikeout and underline.
- Text color.
- Text orientation.
- Automatic word wrap.
- Alignment of the text horizontally and vertically with all options..
- Displaying the background as the color and/or image.
- Display the border around the text, with the specified color, width and style.
- Formation of the text from the attributes of different types and properties.

Fig. 4.2 represents a part of the screen with the frame containing the text examples using various parameters.

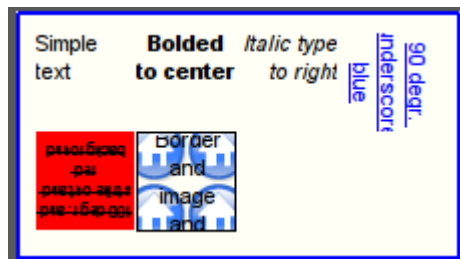


Fig.4.2. Realization of the basic text element in the Vision.

### 4.3. Primitive of the form element (FormEl)

Support of the form elements on the VCA frames is provided. The following form elements are included:

*Line edit* — It is represented by the following types: "Text", "Combo", "Integer", "Real", "Time", "Date", "Date and time". All kinds of line editor support the confirmation of entry.

*Text edit* — It is the flat-text editor with the confirmation or denial of entry.

*Check box* — Provides a field of binary flag.

*Button* — Provides the button with the support of: the color of the button, the image of the button, and mode of fixation.

*Combo box* — Provides the selection field of the element from the list of the items.

*List* — Provides the list box with the control of the current element.

*Slider* — Slider element.

*Scroll bar* — Strip of the scroll bar.

The following modes are realized: «Enable» and «Active», as well as transfer of changes and events to the data model of the VCA (engine).

Fig. 4.3 represents a part of the screen with the frame containing the above-listed elements of the form.

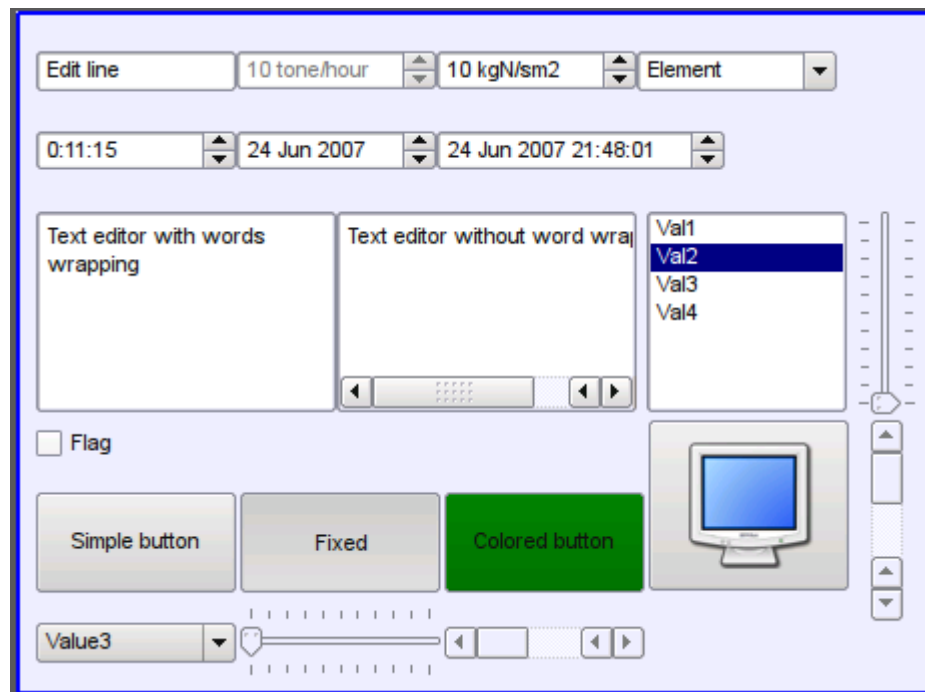


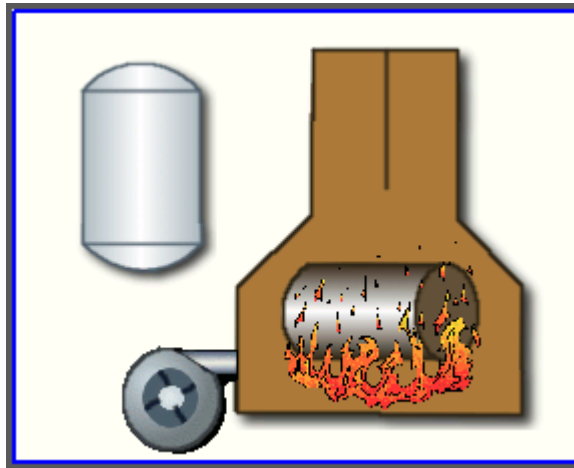
Fig.4.3. Realization of the form elements in the Vision.

#### 4.4. Primitive of the displaying the media materials (Media)

Support of the element of the displaying of media materials with the following properties is provided:

- The indication of the source of media data (images or video material).
- View of the images of most well-known formats with the possibility of inscribing of it in the size of the widget.
- Playback of the simple animated images and video formats with the possibility to control the playback speed.
- Displaying of the the background as a color and/or image.
- Display the border around the text, with the specified color, width and style.
- Formation of the active areas and generating the events when they are activated.

Fig. 4.4 represents a part of the screen with the frame containing examples of viewing/playback of media data.



*Fig.4.4. Realization of the basic element of the displaying of media materials in the Vision.*

## 4.5. Primitive of the construction of diagrams/graphs (Diagram)

Support of the element of the construction of diagrams/graphs with the following properties is provided:

- Construction of graphs/trends:
  - Construction graph for: archive data, current data and the formation of an intermediate buffer for the display of the parameters without archive.
  - Construction of a single graphs with the value of the parameter on the ordinate axis, and the combined graphs of up to 10 parameters, with the percentage scale.
  - Ability to adapt the parameter's graph to the value, the regrowth of scale.
  - Wide range of scalability and adaptation of the horizontal scale, with automatic averaging at the server level and the primitive itself.
  - Ability to display the size grid and markers on the horizontal and vertical, with adaptation to the displaying range.
  - Support of the active mode, with the cursor and getting values under the cursor.

Fig. 4.5 represents a part of the screen with the frame containing examples of the trend-diagrams.

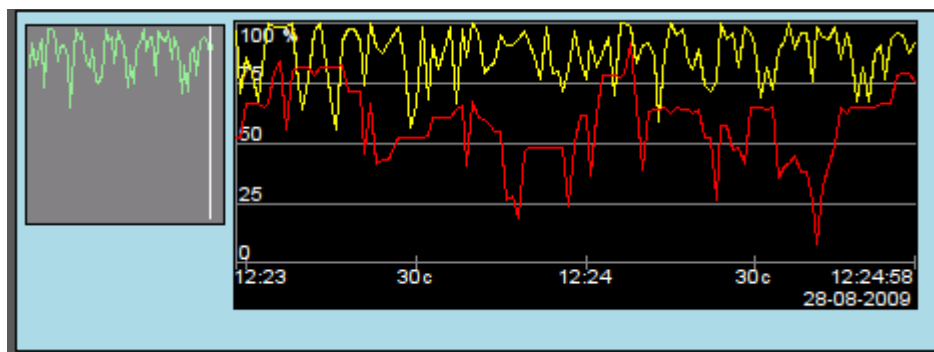


Fig.4.5. Realization of the basic element of a diagram-trend displaying in the Vision.

## 4.6. Primitive of the protocol formation (Protocol)

Support of the element of the formation of the protocol with the following properties is provided:

- Formation of the protocol from the archive of messages for the specified time and depth.
- Request of the data from the messages archivers.
- Selection of data from the archives by the level of importance and the category of messages template.
- Support the tracking mode for the appearance of messages in the archive of messages.

Fig. 4.6 represents a part of the screen with the frame containing an example of the protocol.

	Time	mcsec	Level	Category	Me
1	28.08.2009 13:14:23	8275	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/	Start controller!
2	28.08.2009 13:14:23	3510	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/	Start controller!
3	28.08.2009 13:14:23	25948	1	/DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/	Start controller!
4	28.08.2009 13:14:23	23775	1	/DemoStation/sub_DAQ/mod_System/cntr_AutoDA/	Start controller!
5	28.08.2009 13:14:23	21291	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/	Start controller!
6	28.08.2009 13:14:23	18554	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/	Start controller!
7	28.08.2009 13:14:23	1676	1	/DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/	Start controller!

Fig.4.6. Realization of the basic element of a protocol displaying in the Vision.



## 4.7. Primitive of the report formation (*Document*)

Support element of the report formation with the following properties is provided:

- Adaptive formation of a document structure based on Hypertext Markup Language. This provides support for the broad features of formatting of the documents.
- Formation of the documents on command or on schedule. It is necessary for creation of reports into the archive and then view the archive.
- Formation of a document in real time mode. It is necessary to form documents completely dynamically, and based on the archives for the specified time.
- Using of the the attributes of the widget for transmission of values and addresses to the archives in the report. It allows you to use the widget of the document as a template when generating reports with other input data.

The basis of any document is XHTML-template. XHTML-template is the tag "body" of the WEB-page which contains the document's static in the standard XHTML 1.0 and elements of the executable instructions in one of the languages of the user programming of OpenSCADA in the form of `<?dp {procedure} ?>`. The resulting document is formed by the execution of procedures and insert of their result into the document.

The source for values of the executable instructions are the attributes of the widget of the primitive, as well as all the mechanisms of the user programming language. Attributes may be added by the user and they can be linked to the actual attributes or parameters or they can be autonomous, values of which will be formed in the script of the widget. In the case of linked attributes the values can be extracted from the history, archive.

Fig. 4.7 shows the frame containing a sample of the document.

"Dneprovskij metkombinat" LTD  
(enterprise name, where flowmeter located)

**DAY REPORT**

over " \_\_\_\_\_ " 2008  
made \_\_\_\_\_  
(date) (time)

"FLOWTEC-TM" \_\_\_\_\_ Pipeline \_\_\_\_\_  
(calculator or corrector name) (thread name)

**The flowmeter characterisation included standard narrowing:**

Contract hour	<b>12 25</b>	Sensor's type	<b>Угловой</b>	Atm. pressure, kPa	<b>95</b>
Molar part of N <sub>2</sub> , %	<b>70</b>	Ct. of harshness	<b>0.2</b>	Cuting, kPa	<b>80</b>
Molar part of CO <sub>2</sub> , %	<b>10</b>	Ct of blunting	<b>0.1</b>	Upper border, kPa	<b>150</b>
Pipe diameter, mm	<b>100</b>	Coefficient L (alpha)	<b>0.3</b>	Trigger threshold, kPa	<b>85</b>
Diameter of narrowing, mm	<b>60</b>	Relative square of narrowing	<b>0.6</b>	Dynamic ductility, kgF/m <sup>2</sup>	<b>32</b>

**Hours data**

Date	Time	Capacity, 1000x m3	Aver. pressure diff., pPa	Average pressure, MPa	Average temperatura, °C	Average density , kg.m <sup>3</sup>	
	begin	end					
28 08 2009	09:00	10:00	Empty	Empty	Empty	Empty	
28 08 2009	10:00	11:00	Empty	Empty	Empty	Empty	
28 08 2009	11:00	12:00	2341.62	6.02	9.25	15.44	1.01
28 08 2009	12:00	12:25	2331.34	6.02	9.62	15.48	0.97
All over day			4672.96	12.04	18.86	30.92	1.98

Fig.4.7. Implementation of the basic visualization element of the report documentation in the Vision.

## 4.8. Primitive of the box container (Box)

Support of the primitive of the container concurrently serves as the project pages is provided. This primitive is the only element-container, which may include links to frames from the library, thereby creating the user elements of desired configuration. Primitive implements the provided by the project properties. The properties of this primitive are:

*Container* — Allows you to form the desired objects by grouping in the limits of the primitive.

*Page* — Elements constructed on the basis of the primitive may serve as a page of user interface.

*Container of pages* — Property of substitution of its own contents by another page in the execution process. Used to create frames on the pages of user interface. For example, the main page of traditional SCADA system with alarm objects is constructed in this way.

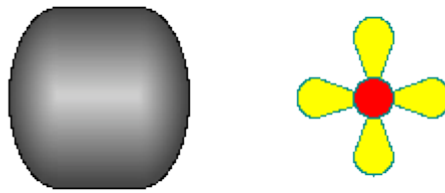
*Background* — Supports ability to specify the background as color or image.

*Border* — Supports the displaying of the border, with the specified color, width and style.

Example of editing of the frame, based on the primitive, is shown in Fig. 2.a, and Fig. 3 shows a page containing the container of the pages, built on the basis of the primitive.



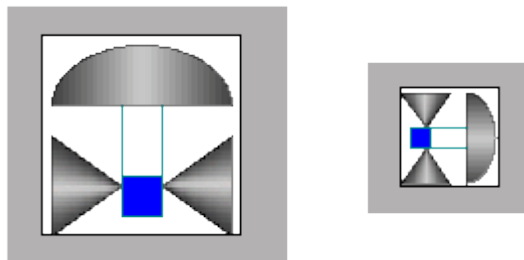
It is possible to connect the various graphic primitives for creation of complex graphic objects. If the connected primitives make the closed circuit, it can be filled with color and/or image (Fig. 5.2.c).



*Fig. 5.2.c. Fills of the closed circuit with the colors and/or images.*

Graphics editor allows you to zoom and rotate the figures (primitives and complex graphic objects) (Fig. 5.2.d).



The features of the editor also include the selection, moving, copying and deleting the figures.



*Fig. 5.2.d. Zoom and rotate the figures.*

### 5.3. Basic principles of operation in the graphic editor

To get to work with a graphical editor after the OpenSCADA is started it is necessary to call "Operation user interface". The "Widgets" tab contains a list of existing graphic libraries and their elements.

Lets suppose that we need to add a graphic element to the one of the existing libraries. To do this, select the name of the library, and click the  on the toolbar. In the appeared window enter the ID and name of the new graphic element. After that lets select the created item and click  button. The drawing area will appear on the right. By double-clicking on that field or either using the context menu lets enter the editing mode - the mode of the graphical editor that allows you to perform all provided manipulations with the figures. At a time when we have the created graphic element in focus, the "Attributes" tab takes the form, shown in Fig. 5.3.a.

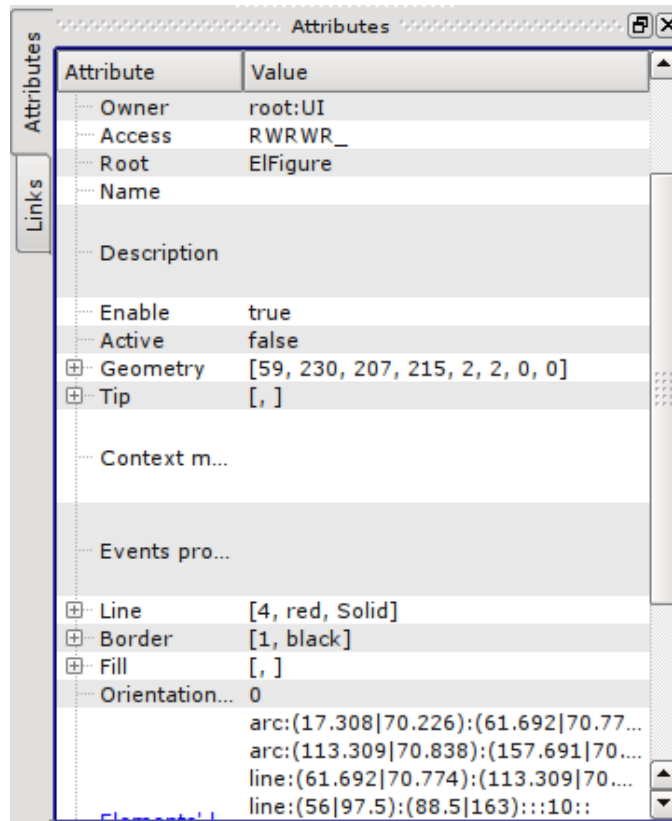


Fig. 5.3.a. Attributes of elementary figure.

With the help of mouse or the "Geometry" field of the "Attributes" tab lets define the drawing area size and scale coefficients.

Using the "Line" field of the "Attributes" tab, lets define the width, color, lines' style of the figures which we'll draw. With the "Border" field lets define the width and color of the border. The "Fill" field allows you to specify color, and image for the fills. "Elements' list" contains a list of primitives used to create the graphic object. Elements of all fields of the "Attributes" tab can be dynamically changed in the scripts (programs) of the user.

Graphics primitives can be drawn with the mouse or by specifying the list of graphical primitives ("Elements' list"). In the first case, the coordinates of control points of the primitive are computed automatically, width, color, style of the line, width and color of the border are set by defaults from the "Attributes" tab. In the second case, the primitive must be described in the "Elements' list" as follows:

*line:(x1|y1):(x2|y2):width:color:border\_width:border\_color:style (1)*

*arc:(x1|y1):(x2|y2):(x3|y3):(x4|y4):(x5|y5):width:color:border\_width:border\_color:style (2)*

*bezier:(x1|y1):(x2|y2):(x3|y3):(x4|y4):width:color:border\_width:border\_color:style, (3)*

Where:

*(x1|y1)* — first control point's coordinates of the primitive;

$(x2|y2)$  — second control point's coordinates of the primitive;  
 $(x3|y3)$  — third control point's coordinates of the primitive;  
 $(x4|y4)$  — fourth control point's coordinates of the primitive;  
 $(x5|y5)$  — fifth control point's coordinates of the primitive;  
 $width$  — line width, with which the primitive will be drawn;  
 $color$  — line color, with which the primitive will be drawn;  
 $border\_width$  — border width;  
 $border\_color$  — border color;  
 $style$  — line style ("0" — solid, "1" — dashed, "2" — dotted).

The priority values are the width, color, style of the line, width and color of the border, specified in the "Elements' list". If you want to use any of the properties that are set in the fields "Line" or "Border" of the "Attributes" tab, then during the description of the primitive they should be skipped. For example, we want to create a line with a width of 3, red color, solid and without border. At the same time in the "Line" field the following properties are set: the line width — 3, black color of the line, solid line style, and in the "Border" field — the width of the border — 5, border's color — green. Then the description of the primitive in the "Elements' list" will be as follows:

`line:(x1|y1):(x2|y2)::red:0:: (4)`

Expressions (1) - (4) define the static properties of the primitives that can not be changed by user's program. To specify the dynamic properties it is necessary to use the following expression:

`line:1:2:w1:c1:w2:c2:s1 (5)`

Then the "Attributes" tab will be appended by the fields: Point 1 (1), Point 2 (2), Width 1(w1), Width 2(w2), Color 1(c1), Color 2(c2), Style 1 (s1) (Fig. 5.3.b), values of these fields can be changed programmatically, by using a programming language of the OpenSCADA project. It is obvious that if desired not all properties of the primitive can be declared dynamic, but one or several, besides one dynamic property can be used repeatedly.

Editing the coordinates of control points is made either with the mouse or through changes in the "Elements' list" for static points, or directly in the "Attributes" tab, having the dynamic ones ("Point 1"...). Editing the other properties is made by means of changing the contents of the fields "Line", "Border", "Fill" of the "Attributes" tab or the "Elements' list".

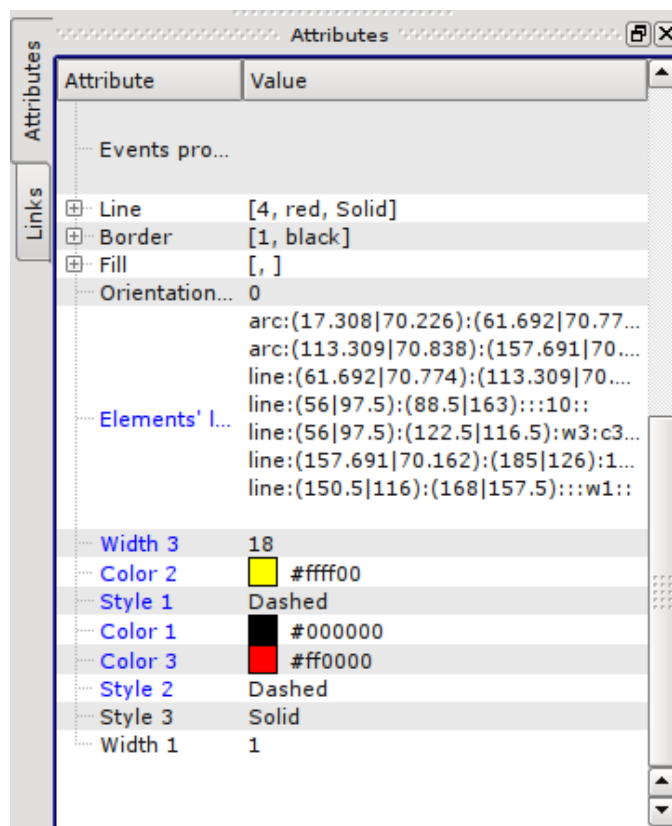


Fig. 5.3.b. Dynamic attributes of elementary figure.

There is the possibility of selection (left click on the figure) the primitive, the joint selection of primitives (left mouse button + "Ctrl" key pressed having disabled the "Connections" button), a joint selection by the frame, drawn by holding down the left mouse button; moving them (keyboard/mouse); copy and paste («Ctrl» + «C», «Ctrl» + «V» or from the popup menu); delete («Del»).

To connect the primitives with each other it is necessary:

- 1) push the "Connections" button;
- 2) select one of the figures and move its start or end point to the desired start or end point of the other figure so that it will get to the appeared circle. Connected figures are moving as well as the individual ones, the general point is moved for all connected figures, to which it refers (priority is given to the arc, two arcs can't be connected directly with each other ).
- 3) release the left mouse button.

To fill the closed circuit it is necessary to make double click of the left mouse button inside it or you can specify a fill in the "Elements' list" as follows:

*fill:(x1:y1):(x2:y2):...:(xn:yn):color:image* – (statics);

*fill:1:2:3:...:c1:i1* – (dynamics).

Where:

*(x1:y1):(x2:y2):...:(xn:yn)* - coordinates of the start/end points of the primitives that form a closed circuit;

*color* - fill color;

*image* - fill image.

The priority values are the color and image of the fill, set in the "Elements' list. " If you want to use the properties set in the "Fill" field of the "Attributes" tab, then they should be skipped in the description of the primitive.

To scale the figures it is necessary in the "Geometry" field to set the scale the for the "X" and "Y" axes. In addition, there is the ability to visually zoom in/out the widget without changing the scale in the "Geometry" field. You must exit the editing mode by the right click on the widget field, select in the popup menu "Zoom in (+10%)" / "Zoom out (-10%)", or rotate the mouse wheel while holding the "Ctrl" key pressed. Exit the editing mode is made either by pressing the "Esc", or using the popup menu of the widget.

It is possible to rotate the contents of the widget. To rotate the created objects you should set the "Orientation angle" from -360 to 360 in the "Attributes" tab.

Graphical editor supports color transparency, defined as follows: color-t, where the color - the color itself, and t - transparency from 0 (fully transparent) to 255 (opaque).

## 5.4. Popup menu of the graphic editor

The popup menu by the right-clicking in the editable widget is provided. Depending on what an object is under the mouse popup menu can take several different forms. Below, in Figure 5.4, there are examples of the popup menu.

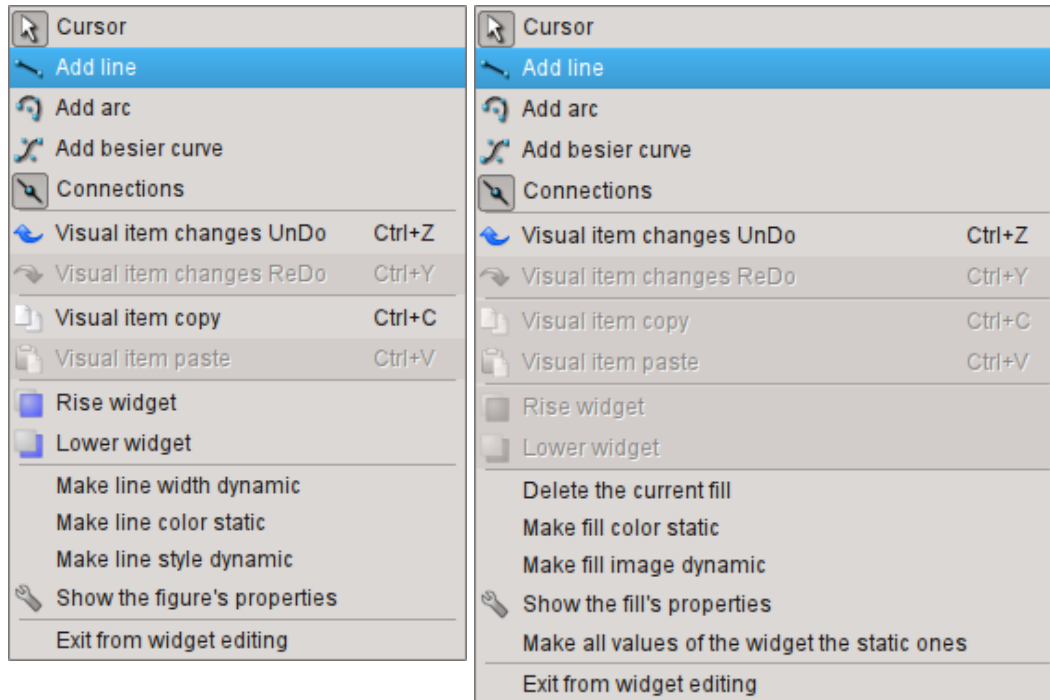


Fig. 5.4. The popup menu for the lines (line, arc, Bézier curve) and for the fill from left to right.

The popup menu includes the following sections (from top to bottom):

- drawing section, allows you to select the figure to paint, the "bindings" mode by the "Connections" item and to return to selection mode - "Cursor";
- undo/redo section;
- copy/paste the selected figures section;
- rising/lowering the selected figures along "Z" section;
- control section (to make static or dynamic) different properties of primitives and also for the call the properties dialog of the elementary figure;
- exit the editing mode section.



## 5.5. Properties dialog of the elementary figure

Dialog, versions of which are shown in Figures 5.5.a, 5.5.b, 5.5.c, is implemented for interactive and comfortable user control of the properties of the figure(s).

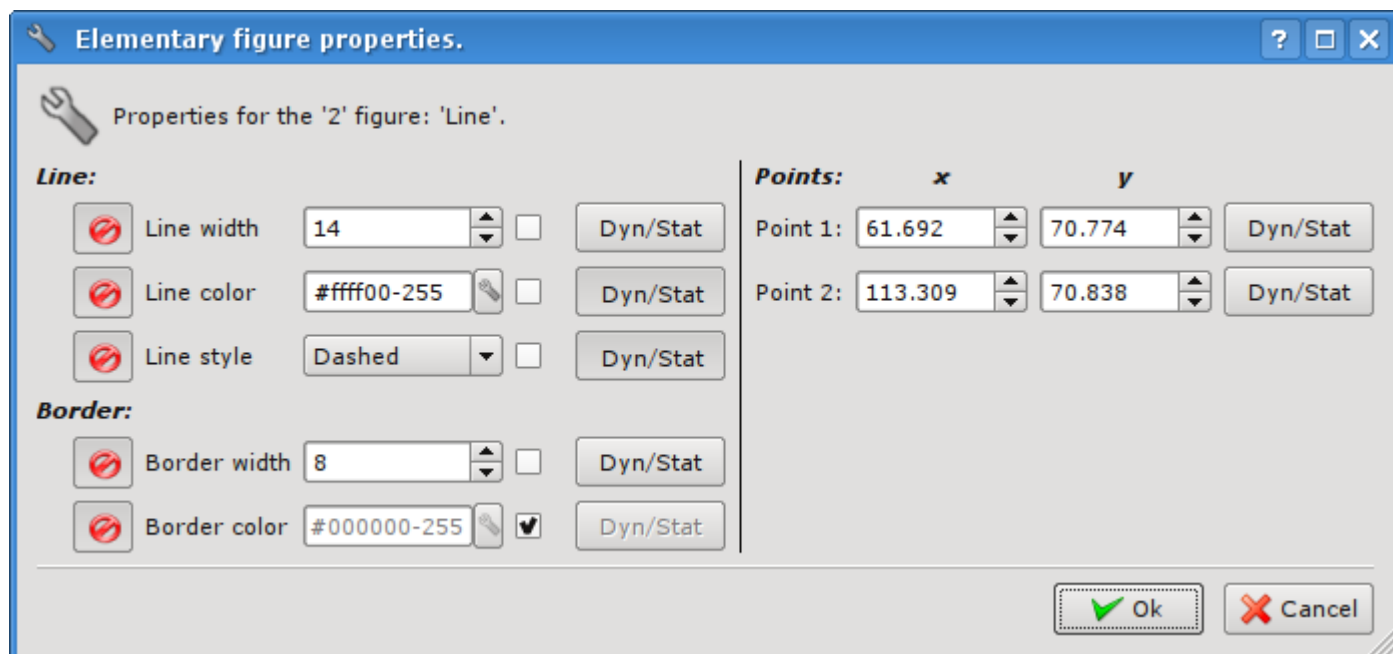
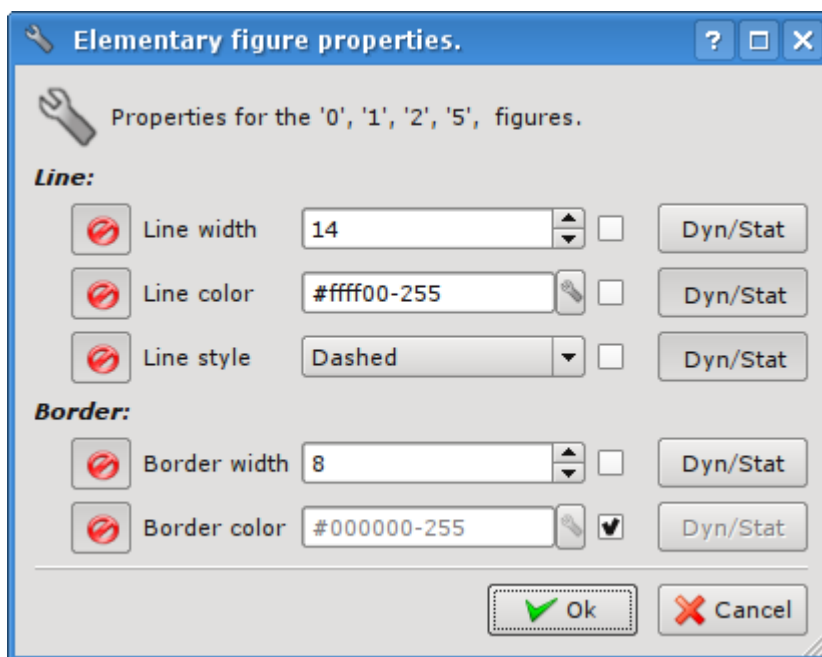


Fig. 5.5.a. Elementary figure's properties dialog for single figure (line, arc or a Bézier curve).



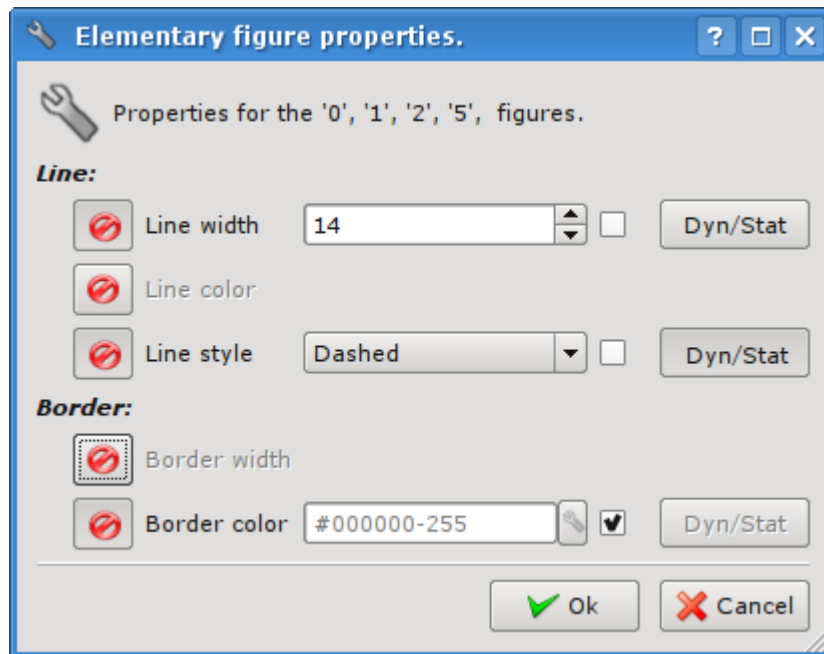


Fig. 5.5.b. Elementary figure's properties dialog for the group of selected figures.

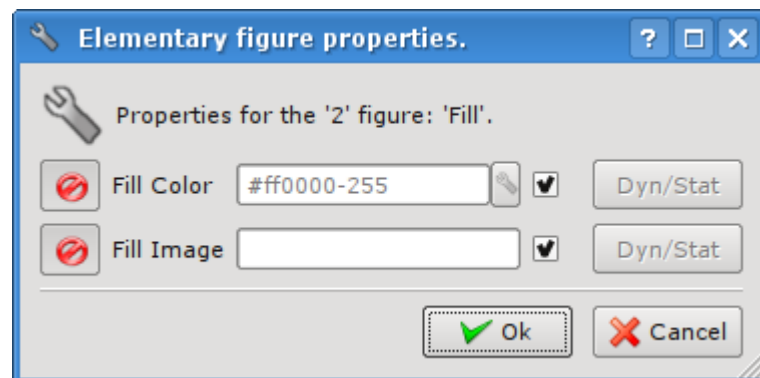



Fig. 5.5.c. Elementary figure's properties dialog for the fill.

As can be seen from the figures above, the dialog can be of three different forms, depending on the object for which it is called. In the title of the dialog there are the numbers of figures, for which it is called. These numbers correspond to the positions of figures in the "Elements" list", starting from the top.

If the dialog called for a single figure (line, arc or a Bézier curve), then it is possible to edit the points' coordinates of the figure (Fig. 5.5.a). If the point is connected to the point of another figure or figures, and "bindings" ("Connections") are enabled, then entered coordinates will be applied to all connected figures and fills, which involved the figure, will be redrawn accordingly.

If the dialog is called for the group of selected figures (Fig. 5.5.b), properties (attributes) listed in the dialog will be applied to all figures listed in the title of the dialog. When you call the dialog in the fields of the properties will be shown the data of the figure from the group of selected ones, for which it was called the popup menu. There is the possibility to include/exclude certain properties of the dialog. For this the (  ) button is provided. In the case of exclusion of individual properties, they will not be processed upon acceptance of the dialog ("Ok" button). After the acceptance of the dialog, all of the data of the included properties will be applied to the whole group of figures.

Dialog for the fill's properties (Fig. 5.5.c) allows you to manage the properties of the specific fill.

When selecting a check box to the right of a property, it (the property) after the acceptance of the dialog is set to the default value, which is listed in the "Attributes" tab. The "Dyn/Stat" buttons make the appropriate properties dynamic or static.

## 6. The overall configuration of the module

To adjust your own behavior in not obvious situations the module provides the ability to customize individual settings through the management interface of the OpenSCADA (Fig. 6.a). These settings are:

- Module Status: running and the number of available screens in the library of QT4.
- The name of the remote OpenSCADA station with visualization engine VCA.
- Initial user of the configurator — points on behalf of what user to open configurator without requiring the password.
- The lifetime of the pages in the cache. Visualizer provides acceleration of the user interface by placing a previously opened pages in the cache, to control the lifespan of pages in which the property and intended. Specifying a value of zero generally excludes cleaning the cache.
- The list of projects, by separate symbol ';', for their automatic execution with the launch of the module. To provide the possibility to indicate the opening of the window of the project execution on the desired display of many display systems the recording format of the project "PrjName-1" is provided, where 1 — the number of the target display.
- The link to the configuration page of the external OpenSCADA stations.

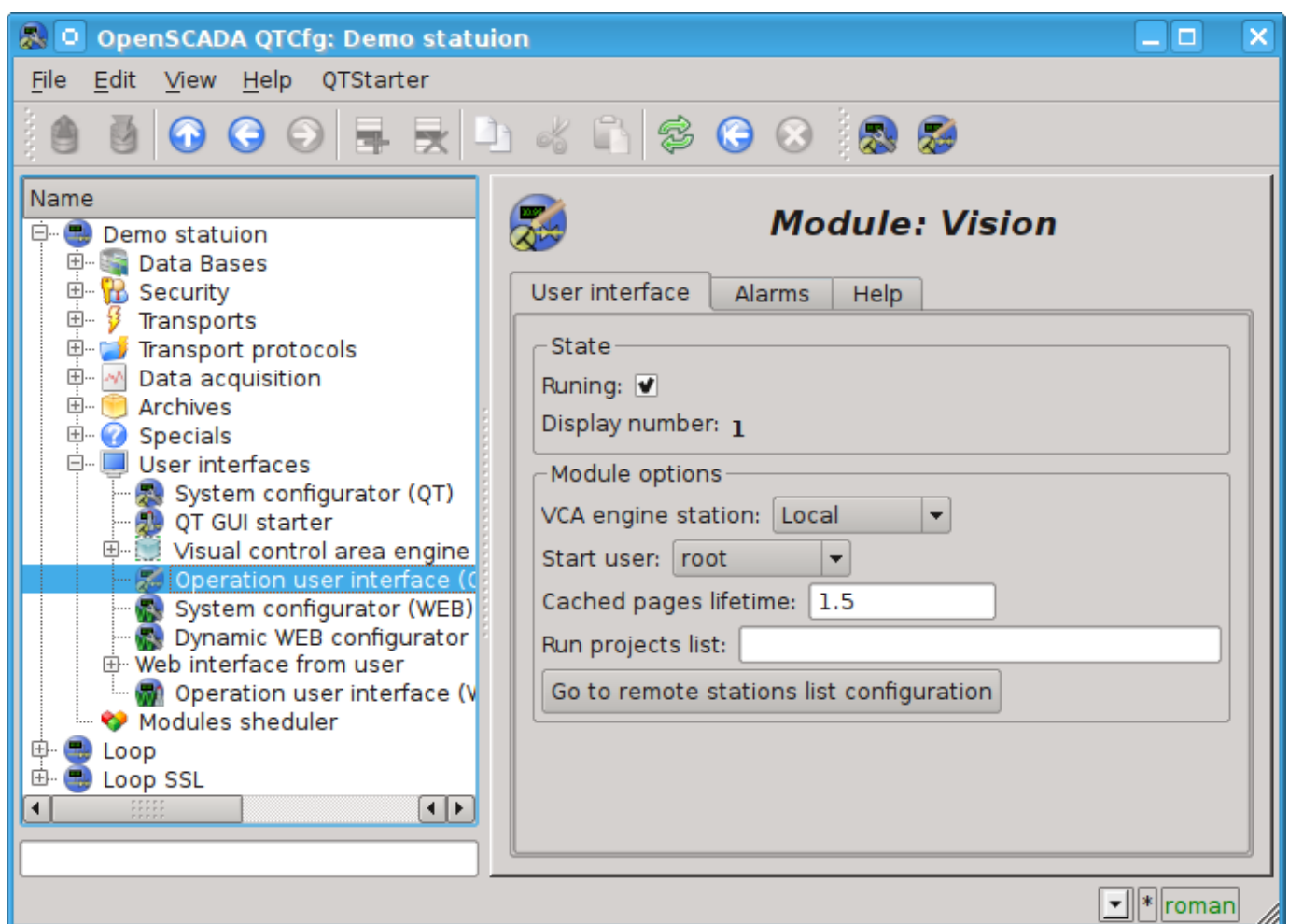
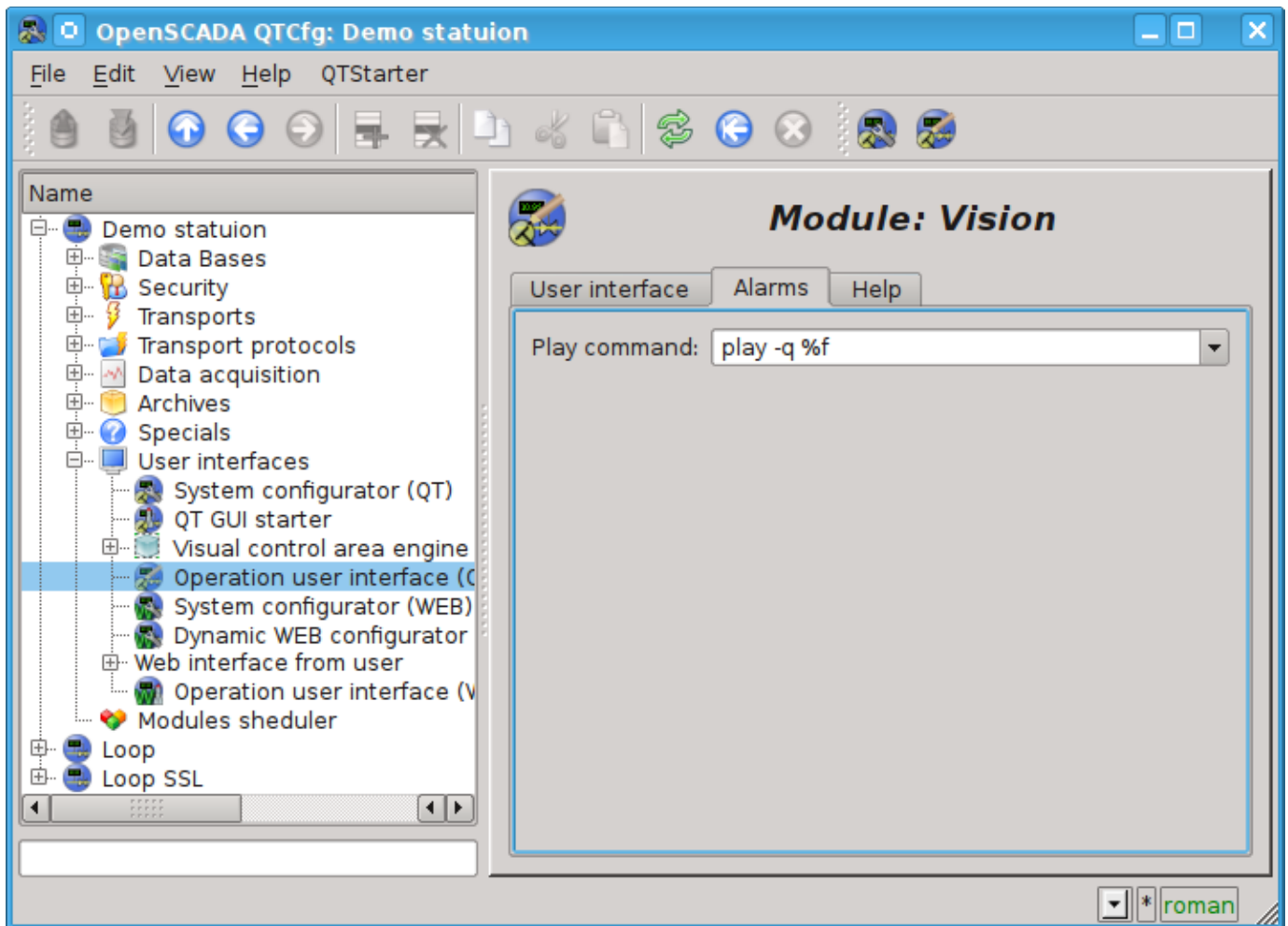


Fig.6. The configuration page of the module.

Setting for processing of alarms can be made in the "Alarms" tab (Fig. 6.b). In this tab you can specify a string of command to play audio files, usually a "play-q %f".



*Fig.6.b. Tab "Alarms" of the configuration page of the module.*