RadiCS Platform high-level software

u7 Integrated Development Environment

User Manual

[1. u7-ide program’s functions and fearures 3](#_Toc419277533)

[2. PostgreSQL Server setup 4](#_Toc419277534)

[3. Installing and setting up u7set 9](#_Toc419277535)

[4. Project Managing 13](#_Toc419277536)

[5. Hardware configuration 18](#_Toc419277537)

[6. Сигналы приложения (Application signals). Добавление, редактирование и удаление. 25](#_Toc419277538)

[7. FBL-element library 26](#_Toc419277539)

[8. Application Logic Editor 34](#_Toc419277540)

[9. Редактор технологических видеокадров (Workflow schemes) 45](#_Toc419277541)

[10. Редактор диагностических видеокадров (Diag schemes) 45](#_Toc419277542)

[11. Компиляция проекта (Build) 45](#_Toc419277543)

# u7-ide program’s functions and fearures

* 1. The u7-ide program is an integrated development environment for the RadiCS Platform. It is a main tool for creating projects.

The program uses the client-server architecture. All project data is stored on a database server on local or remote machine. The program connects to the server and works with the database.

The u7-ide program allows many users to work with the project at the same time. Every user has a login and a password and can change any allowed part of the project.

The project database keeps the history of all changes. To edit some project data, user should make the “check-out” operation for the item needed to be edited. If some other user has already checked out this item, the operation will fail. After finishing the editing process, user should do “check-in” operation to place his local changes to the project database. If needed, user can take any previous version of the changes.

After building the project the program creates a set of files. Some of them are used by other software, and some of them contain hardware information and must be programmed to logic modules’ flash memory.

* 1. The u7-ide program has following main functions:

–creating projects, editing and deleting them;

–project users management;

– creating a hardware configuration database;

– creating application signals database;

– creating and using the FBL-elements library;

– creating application logic schemes;

– creating workflow schemes;

– creating diagnostic schemes;

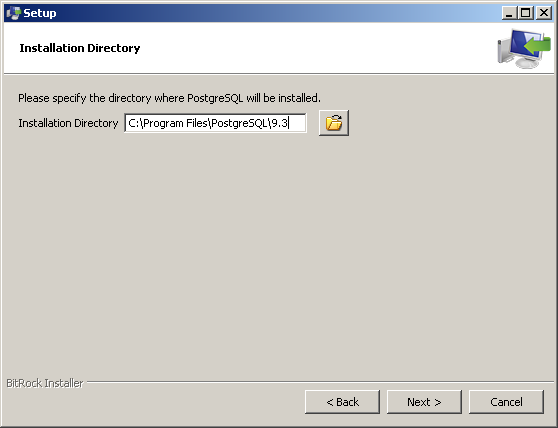
– building the project.

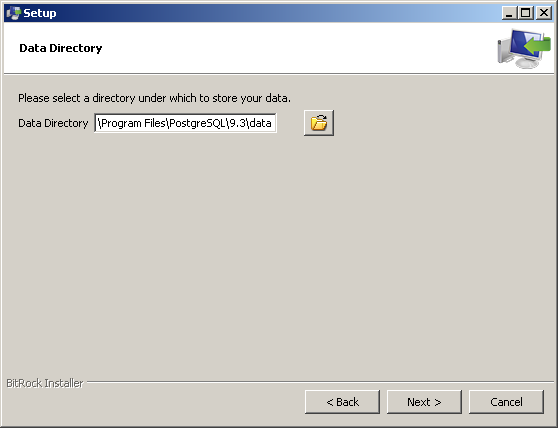
# PostgreSQL Server setup

* 1. Server installation
     1. To install the database server for u7-ide projects, run the install file of the PostgreSQL Server, for example, postgresql-9.3.5-1-windows-x64.exe. A welcome screen will appear. On every screen, modify the data if needed, and press the “Next” button.

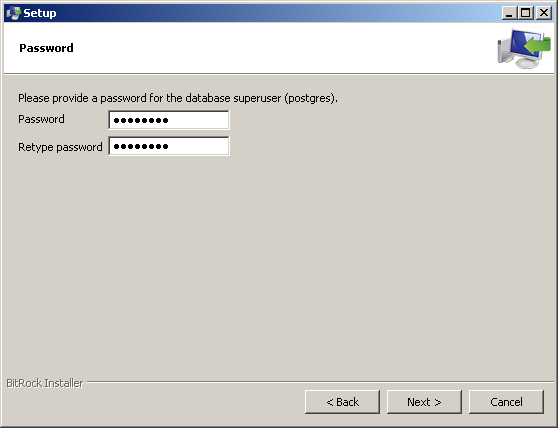


* + 1. Choose the installation path and data path.

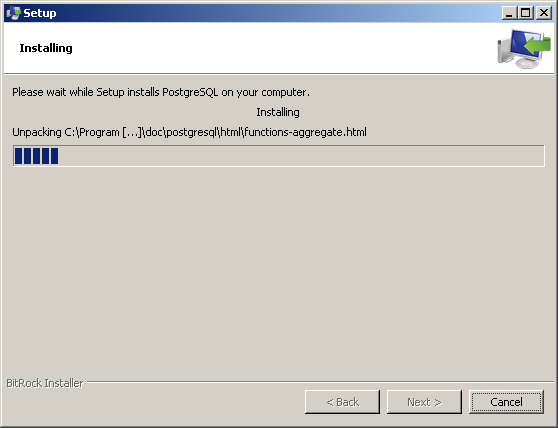




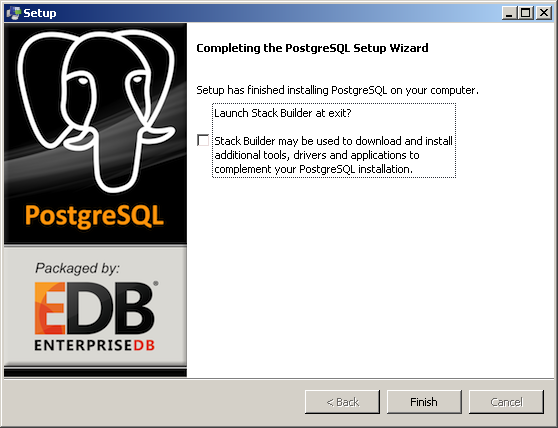
* + 1. Provide a superuser password.



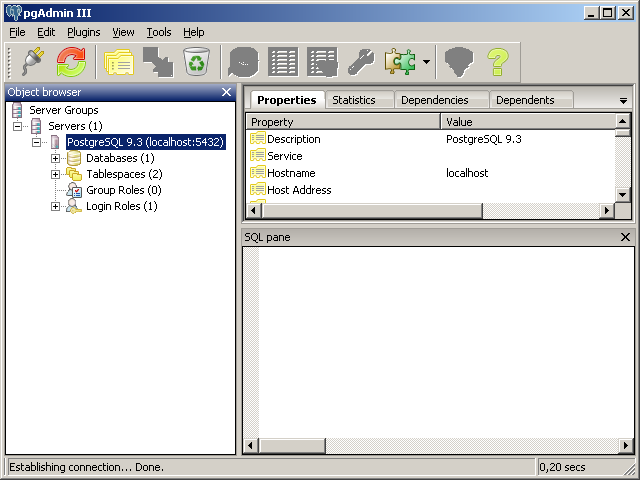
* + 1. The installation process will begin.



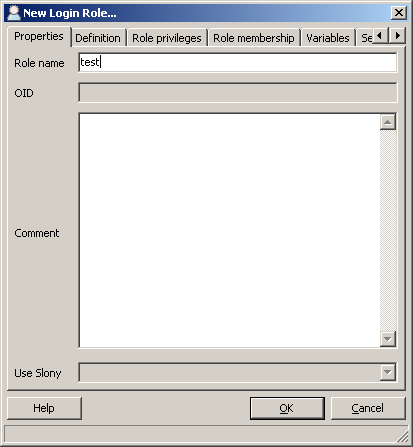
* + 1. At the end of the installation, press the “Finish” button. The server will start its service at background.



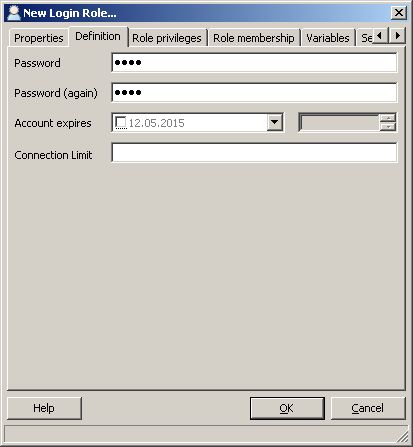
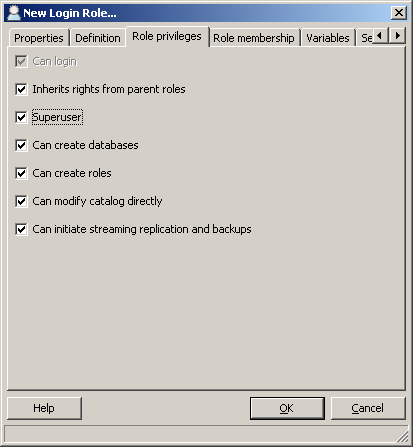
* 1. Server configuration
     1. To configure the server, run the program called “pgAdmin III” and enter the superuser password. The following screen will appear.



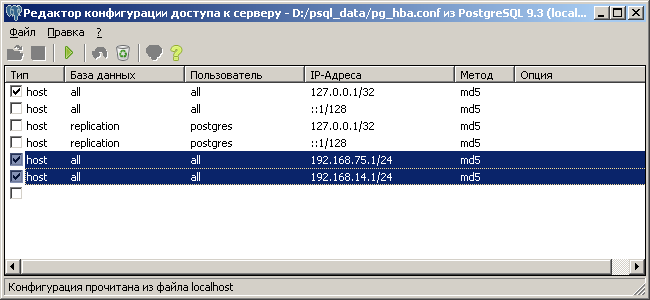
* + 1. In the “Login Roles”, select “New Login Role…” menu item and create a user which will have full access to the database. This login will be used by the u7-ide program.



* + 1. Supply a password and full access rights for this user.

* 1. Choose the “pg\_hba.conf” menu item in the “Tools” - “Server Configuration” menu and add IP address masks to allow remote hosts to connect to this server. For example, for IP range “192.168.75.1 – 192.168.75.254” write “192.168.75.1/24”.



# Installing and setting up u7set

* 1. General settings
     1. System requirements:

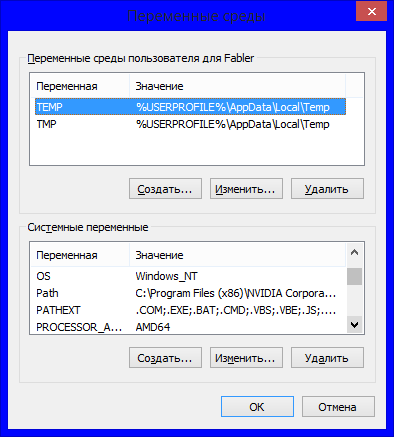
– Qt Library (up to 3.1.2);

– Database Management System PostgreSQL (up to 9.3.5);

– Executable file u7.exe;

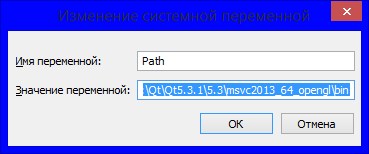
– DLL-module VFrame30.dll

* + 1. u7set requires additional libraries to be installed into system variable "Path". To do this, open context menu of "My computer", and choose "Properties". After that, open "Additional system settings". Then a new dialog window called "System Properties" will be shown. In this window, choose parameter "System Variables". Sample of the window shown on the figure 3.1.



|  |  |
| --- | --- |
| Figure | 3. |
|  |  |

After that, list of all system variables will be shown. Scroll it down to variable called "Path". Click on it, and push "Change" button. New dialog will be opened (fig. 3.2).



|  |  |
| --- | --- |
| Figure | 3. |
|  |  |

Insert in the end of line "Variable value" symbol ";" and path to your PostgreSQL dir. To insert second path of your Qt libraries, add ";" symbol first.

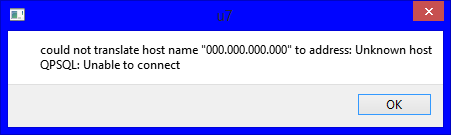
WARNING: Don't add ";" symbol after second Path.

Example:

";C:\Qt\Qt5.3.1\5.3\msvc2013\_64\_opengl\bin;C:\ PostgreSQL\9.3\bin".

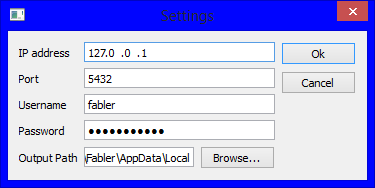
Close all windows for next step.

* 1. Starting program
     1. After starting up, program connects to server. If server address not set, program will show appropriate message (fig. 3.3).



|  |  |
| --- | --- |
| Figure | 3. |

To change server address, open "Tools" bookmark in the main window, and select parameter "Settings". Sample of "Settings" windows shown on figure 3.4.



|  |  |
| --- | --- |
| Figure | 3. |

Settings window contains:

– IP address - server address to connect for;

– Port - server port to connect for;

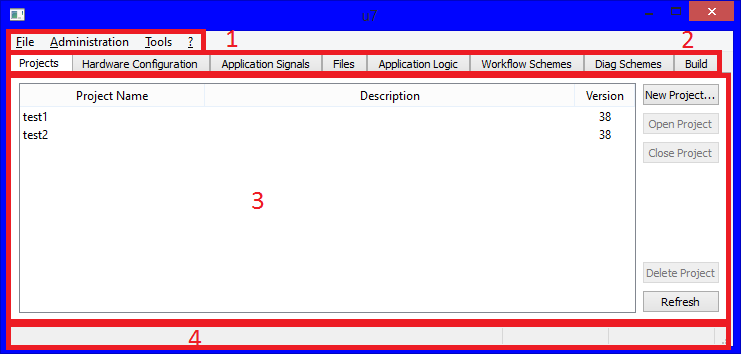
– Username - Server database user name;

– Password - User's database password;

– Output path - path, where program will store building result.

* 1. Program GUI

The sample of program main window shown on figure 3.5.



|  |  |
| --- | --- |
| Figure | 3. |

Program's main window contains:

– Main menu(1);

– Inset panel(2); (панель вкладок)

–Program workspace(3);

–Program status bar.

* + 1. Main menu contains:

– Files (Exit from program)

– Administration (Project logs, and user administration tool);

– Tools (Program general settings);

– ? (Program help).

* + 1. Inset panel contains:

– Projects (Project managing);

– Hardware configuration (Preset and hardware settings);

– Application signals (Signal settings);

– Files (Project Files);

– Application Logic (Project logic settings);

– Workflow Schemes ();

– Diag Schemes ();

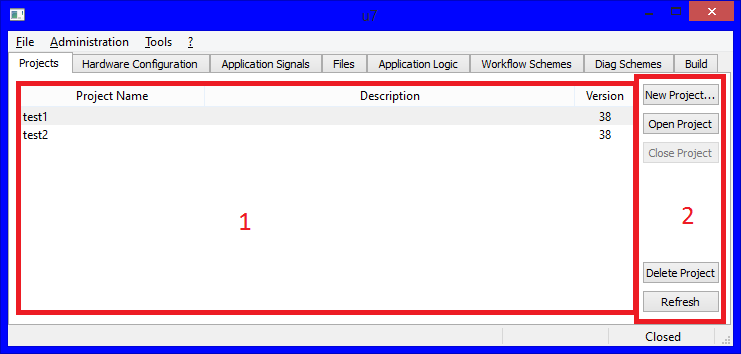
– Build (Project build settings).

Workspace can show different information in case of selected work mode.

Status bar shows progress of connecting to server.

# Project Managing

* 1. Project bookmark was shown on figure 4.1.



|  |  |
| --- | --- |
| Figure | 4.1 |

Project panel contains:

– workspace (1);

– control panel(2).

Workspace contain list of projects on server. List made of columns:

– Project Name (Name of the project);

– Description (Project description);

– Version (u7-ide version, which project was built).

Control Panel contain functional buttons for work with projects:

– Create project (Creates project);

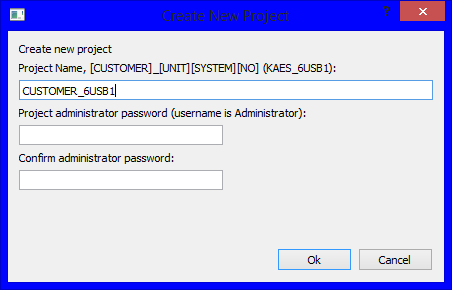
– Open project (Opens selected project);

– Close project (Close opened project);

– Delete project (removes selected project from server);

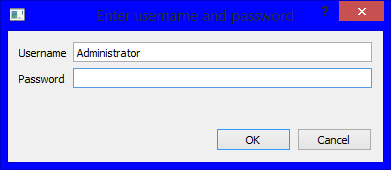
– Refresh (Refresh projects info);

* 1. To create project push button "Create project". "New project" window will be shown on the screen (fig. 4.2). In dialog window fill parameters "Project name" and Administrator password (key-word to access project). Password must be typed two times. After typing press "Ok" button to accept changes, or "Cancel" button to decline.

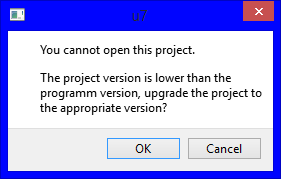


|  |  |
| --- | --- |
| Figure | 4. |

* 1. To open project press "Open Project" button. New dialog window will be shown (fig. 4.3). Type login and password to continue. If project's version is not similar to your u7-ide version, new message dialog will be shown (fig. 4.4). In this case, program will offer to update your project. Press "OK" button to continue.

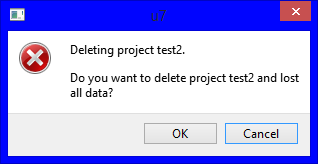


|  |  |
| --- | --- |
| Figure | 4. |



|  |  |
| --- | --- |
| Figure | 4.4 |

* 1. To close selected project press "Close Project" button. Your project will be closed.
  2. To remove selected project, press "Delete Project" button. New dialog window will be shown (fig. 4.5). Press "OK" button to remove all project data from server, or "Cancel" to decline.



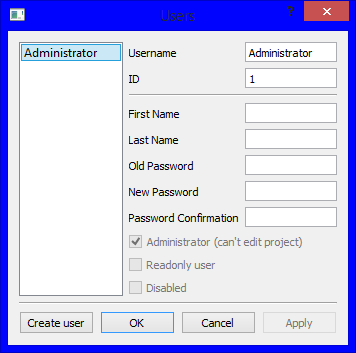
|  |  |
| --- | --- |
| Figure | 4. |

ATTENTION: Selected project must be closed, to be removed.

* 1. To refresh list of projects press "Refresh" button.
  2. To add users select parameter "Administration" from "Users" in main menu.

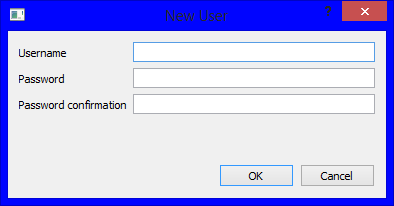
ATTENTION: You can add users only in opened project.

New dialog window will be shown (fig. 4.6)



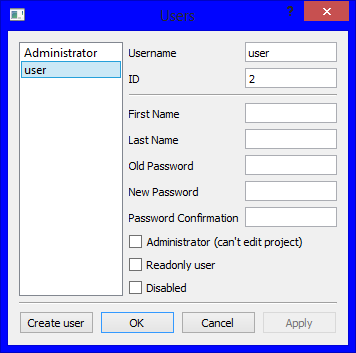
|  |  |
| --- | --- |
| Figure | 4.6 |

Here shown all project users. To create new user press "Create user" button. New dialog window will be shown (fig. 4.7).



|  |  |
| --- | --- |
| Figure | 4.7 |

Fill first row with username, second and third must be filled with password. After pressing "OK" button new user will be created, and user list in previous window will be refreshed (fig. 4.8).



|  |  |
| --- | --- |
| Рисунок | 4.8 |

To activate created user select one of work modes:

– Administrator (Read-Write Administrator rights);

– Readonly user (Read-Only user);

– Disabled (Blocked user).

Also "Users" window includes next options:

– Change Username of selected user (Value Username);

– Change Name and Surname of selected user (First name and Second name values);

– Change Password of selected user (Type old password, then two times type new password).

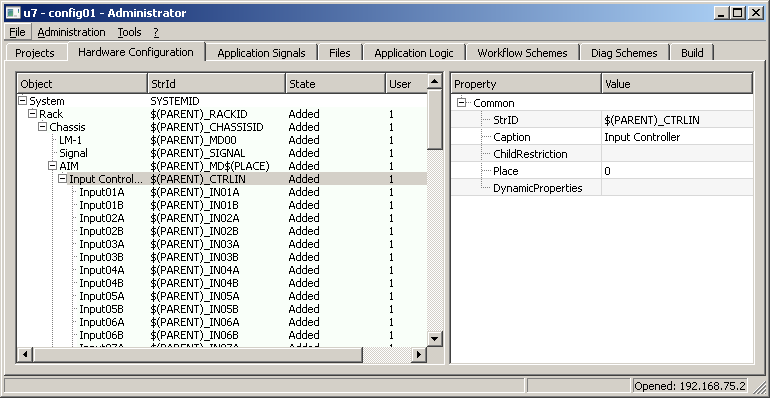
To accept changes press "Apply" button, or "Cancel" to discard them.

# Hardware configuration

* 1. Hardware configuration is a part of a project that describes the structure of the system. It contains different kinds of objects: system, rack, chassis, modules, controllers and signals. It has hierarchical structure. A system is always a top-level object that contains a set of child objects called racks. A rack contains child objects called chassis. And chassis contains a set of modules.

Every module can contain two types of child objects: controllers and signals. Controllers can contain only signals.

The “Hardware configuration” tab with sample configuration tree is shown on the figure.



Each hardware configuration object has a set of properties. Different types of objects may have same and different properties.

The most important common properties are:

– StrID – a string identifier of an object. It has common structure: $(PARENT)\_ID, where $(PARENT) is the identifier of the parent object, and ID is an own identifier. Systems have no parent objects, so they have simple identifiers. When the project compiles, $(PARENT) macro are replaced by real parent identifiers. $(PLACE) macro means an object place and also is replaced by real number.

Here are some examples of identifiers. The first column is an identifier with a macro, and the second column is an identifier with macro replaced by parent id.

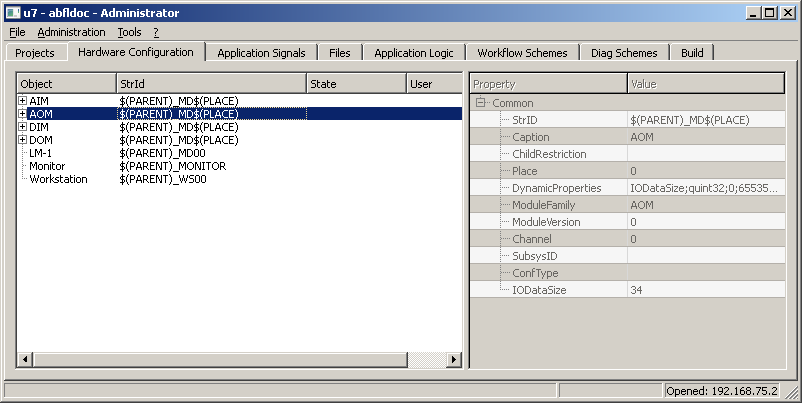
|  |  |  |
| --- | --- | --- |
| USB | USB | System ID |
| $(PARENT)\_RACK01 | USB\_RACK01 | Rack ID |
| $(PARENT)\_CHAS00 | USB\_RACK01\_CHAS00 | Chassis ID |
| $(PARENT)\_MD$(PLACE) | USB\_RACK01\_CHAS00\_MD00 | Module ID |
| $(PARENT)\_CTRLIN | USB\_RACK01\_CHAS00\_CTRLIN | Controller ID |
| $(PARENT)\_IN01 | USB\_RACK01\_CHAS00\_CTRLIN\_IN01 | Signal ID |

– Caption – a name of an object. It contains a string description of the object;

– Place – an object position. This property is important for racks, chassis, modules and signals. For example, this property means a module’s position in a rack.

* 1. Hardware configuration database contains two main parts: hardware configuration itself and a list of presets. User can switch between editing these parts by right-clicking the mouse on the tree area and choosing the “Swich to Preset” or “Switch to Configurations” menu commands.

A sample preset editing window is shown on the figure.

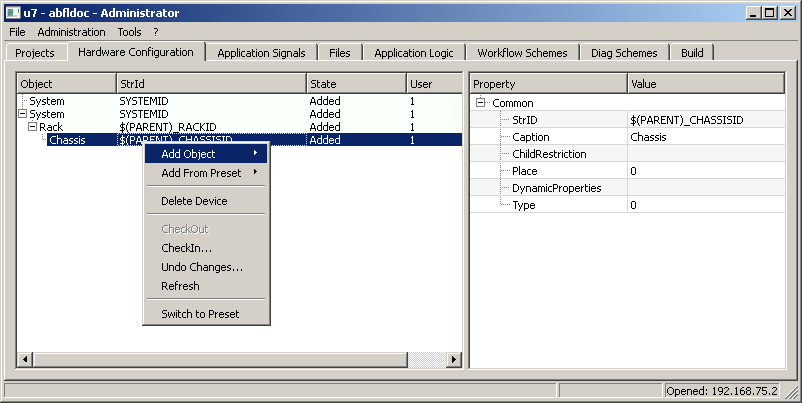


A preset is an object that contains other objects. For example, a module is a preset containing controllers and signals. Presets can be added to the hardware configuration. When preset is added, all its child objects are also added automatically.

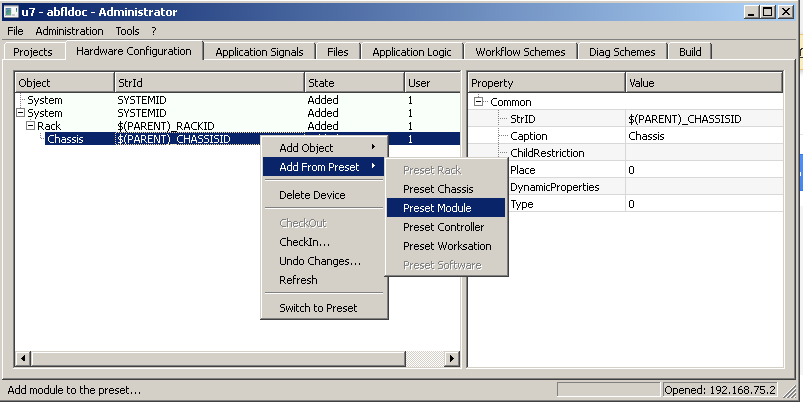
For example, user can create a hardware configuration, add a system, rack and chassis objects. Then he can add few module presets to the rack. In this case, all modules’ signals and controllers will be added also.

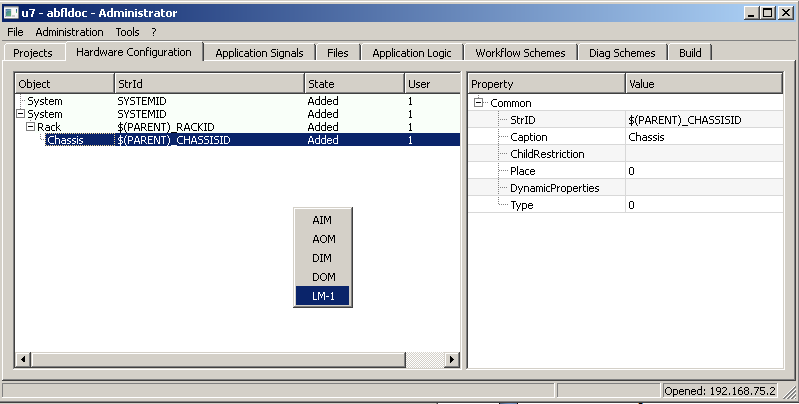
The hardware configuration may contain many instances of presets, for example, many same modules. If it is necessary to make changes in a module, these changes should be done only in the preset library. And then, after user command, all of the instances of this module will be updated to the preset contents automatically.

* 1. Editor commands in both hardware configuration and presets editing modes are the same. On the left side of the “Hardware configuration” tab there is an object tree, and on the right side there is a properties window.
     1. To add an object to a tree, select the parent object and click the right mouse button on a tree area and choose the kind of an object in the “Add Object” submenu as shown on the figure. Note that objects should be added following the hierarchy: rack is a child of a system, chassis is a child of a rack and so on.



* + 1. To add a preset to a tree, select the parent object and choose a preset type in the “Add From Preset” submenu, as shown on a figure. A menu with available presets list will appear. Choose needed preset from this menu, as shown on the figure.





* + 1. To delete an object, select it in a tree and choose the “Delete Object” command.
    2. Before editing an existing object, check it out by choosing the “CheckOut” menu command. After editing is finished, check it in by choosing the “CheckIn” menu command. By this, your changes will be placed in the project database. To undo changes that were not checked in yet, use the “Undo” command.
    3. The “Refresh” command is used to request hardware configuration data from the project database and to redraw tree contents.
    4. To edit object properties, select needed object in the tree and check it out if necessary. On the right side of the window the properties list will appear. To change a property, click on it in a table and enter a value in a “Value” column.
    5. A system object has following properties:

|  |  |
| --- | --- |
| StrID | String identifier |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Reserved |
| DynamicProperties | Dynamic properties list |

* + 1. A rack object has following properties:

|  |  |
| --- | --- |
| StrID | String identifier, must contain $(PARENT) macro |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Reserved |
| DynamicProperties | Dynamic properties list |

* + 1. A chassis object has following properties:

|  |  |
| --- | --- |
| StrID | String identifier, must contain $(PARENT) macro |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Reserved |
| DynamicProperties | Dynamic properties list |
| Type | A type of the chassis |

* + 1. A module object has following properties:

|  |  |
| --- | --- |
| StrID | String identifier |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Module place in a chassis, 1-based. |
| DynamicProperties | Dynamic properties list |
| ModuleFamily | A family of a module, for example “AIM” or “LM-1” |
| ModuleVersion | Module software version |
| Channel | Number of a system channel, which this module belongs to. Used only in LM modules family. |
| SubsysID | Subsystem name |
| ConfType | Reserved |
| IODataSize | Size, in words, of module’s data packet. |

* + 1. A controller object has following properties:

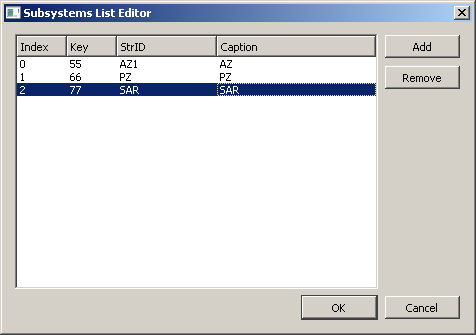
|  |  |
| --- | --- |
| StrID | String identifier, must contain $(PARENT) macro |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Reserved |
| DynamicProperties | Dynamic properties list |

* + 1. A signal object has following properties:

|  |  |
| --- | --- |
| StrID | String identifier, must contain $(PARENT) macro |
| Caption | Caption |
| ChildRestriction | Reserved |
| Place | Index of the signal, may be used in some families of modules |
| DynamicProperties | Dynamic properties list |
| Type | Signal type: input, output, analog, discrete, diagnostic etc. |
| ByteOrder | Bytes order: big endian or little endian |
| Format | Data format: signed, unsigned etc. |
| Size | Size, in bits, of a signal |
| ValidityOffset | Validity offset in a data packet |
| ValidityBit | Validity bit number in a data packet |
| ValueOffset | Value offset in a data packet |
| ValueBit | Value bit number in a data packet (always 0 for analog signals, because they have whole word or byte) |

* 1. After adding a module to the hardware configuration it is necessary to fill its Place property. For LM is also necessary to fill its SubsysID property. Without this, compile errors will appear.
  2. SubsysID is an identifier, which shows the subsystem code for an LM module. This code depends on the hardware set. A set of subsystems need to be created before building.

To edit this set, choose the “Subsystem List Editor” command in “Tools” menu. Editor window is shown on a figure. To add and remove items, use “Add” and “Remove” buttons. To edit values in “Key”, “StrID” and “Caption” columns, double click left mouse button on them and enter the data.



Values of SubsysID properties in LM modules should exist in StrID column of the subsystems list. Otherwise, compile errors will appear.

# Сигналы приложения (Application signals). Добавление, редактирование и удаление.

# FBL-element library

* 1. Using of FBL-elements

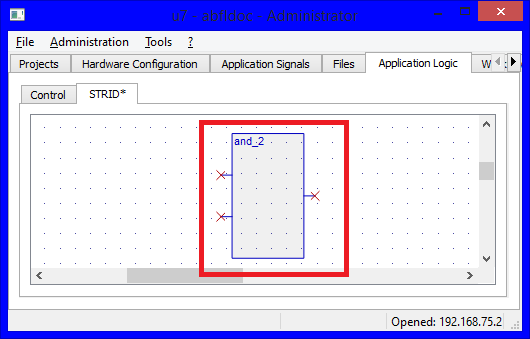
FBL (Functional Block Logic) element - text-based file with block settings, which will be used for creating and optimization program logic.

It's possible to create a library of FBL-elements. Every element is XML-based file. Example of such file shown on fig. 7.1.



|  |  |
| --- | --- |
| Figure | 7.1 |

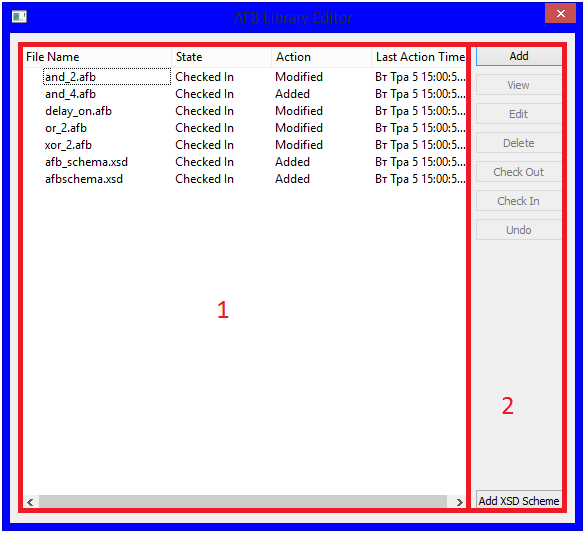
FBL-elements from library can be added to logic construction schemes, and then converted to command lines in compilation time. Example of FBL-element on scheme shown on fig. 7.2.



|  |  |
| --- | --- |
| Figure | 7.2 |

* 1. FBL-element editor. Creating, editing and removing elements.

To open FBL-elements editor, select "AFBL Editor..." from "Tools" menu in main menu. New window will be shown (fig. 7.3).



|  |  |
| --- | --- |
| Figure | 7.3 |

Editor consist of workspace (1) and control panel (2).

* + 1. Workspace keep list of elements in the project. List includes next columns:

– Element name ("File Name");

– Element status ("State");

– Last action with element ("Action");

– Element last action time (Last action time).

* + 1. Control panel includes next buttons:

– "Add" - add new FBL-element;

– "View" - open selected FBL-element for viewing;

– "Edit" - open selected FBL-element for editing;

ATTENTION: Button can be available only if element is "Checked In" ("Check In" button is activated).

– "Delete" - remove selected element;

– "Check Out" - prepare selected element for editing;

– "Check In" - close selected element for editing;

– "Undo" - cancel last action;

Push "View" button to open selected element for viewing. New editor window will be shown (fig. 7.4).



|  |  |
| --- | --- |
| Figure | 7.4 |

To edit selected element, push "Check Out" button first. Then push "Edit" button, new editor window will be shown (fig. 7.5).



|  |  |
| --- | --- |
| Figure | 7.5 |

ATTENTION: When element "Checked Out", it's became unavailable for other users.

Editor window include workspace (1) and Control panel(2). In workspace FBL-element code will be shown.

Control panel include:

– Validate... - Check code validity;

– OK - save code;

– Cancel - close code without saving.

To save changes push "OK" button, or press "Cancel" button to decline changes. Then push "Check In" button if needed (Element became available for editing for others).

* 1. FBL-element structure. Properties, inputs and outputs.

FBL-element creates with XML-script. Example of FBL-element code with tag description shown on figure 7.6.



|  |  |
| --- | --- |
| Figure | 7.6 |

First tag <?xml...?> include properties description of xml-file: xml-version and encoding.

Description of all FBL-elements keeps inside ApplicationFunctionalBlocks block. FBL-element code keeps in AfbElement block. Afbelement has next attributes:

– Unique key (Guid);

– Unique identificator (StrId);

– Name (Caption);

– Operation code (OpCode).

ATTENTION: Guid and StrId will be unavailable for editing after saving script.

AfvElement block include next parts:

– InputSignals – keep all input signals description, marked with AfbElementSignal tags;

– OutputSignals – keep description of all output signals, marked with AfbElementSignal tags;

– Params – keep element parameters, which are available for editing in scheme editor. Elements are described with AfbElementParam tags. Example of script shown on figure 7.7.



|  |  |
| --- | --- |
| Figure | 7.7 |

– ConstParams – parameters, which are disabled for editing by user. Described with AfbElementParam tags.

AfbElementSignal tag include next attributes:

– Signal name (Caption);

– Signal type (Type). Can be changed to "Analog" or "Discrete";

– Signal number (Index);

– Signal Size in bytes (Size).

AfbElementSignal include next attributes:

– Parameter name (Caption);

– Parameter visibility (Visible);

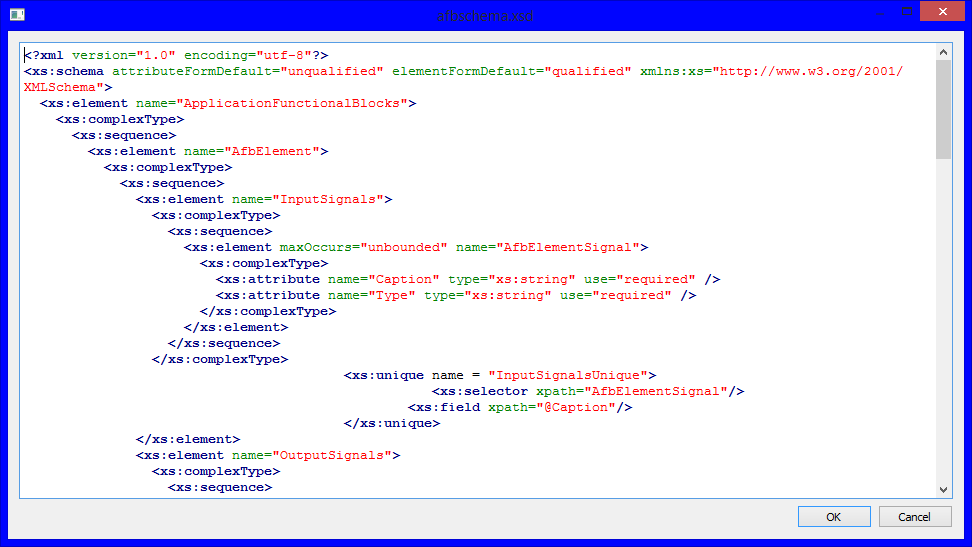
– Parameter type (Type). Can be changed to "AnalogInternal", "AnalogFloatingPoint", or "DiscreteValue";

– Parameter number (Index);

– Parameter Size in bytes (Size).

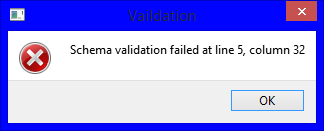
* 1. Press "Validate..." button to check code validation. Then, list of .xsd-files will be shown under the button. Choose file, which needed.

.xsd-file (scheme file) holds up rules of AFBL-element syntax. It's can be edited if needed, and stores in the same list, where AFBL-elements keep. Example of .xsd-file shown on figure 7.8.



|  |  |
| --- | --- |
| Figure | 7.8 |

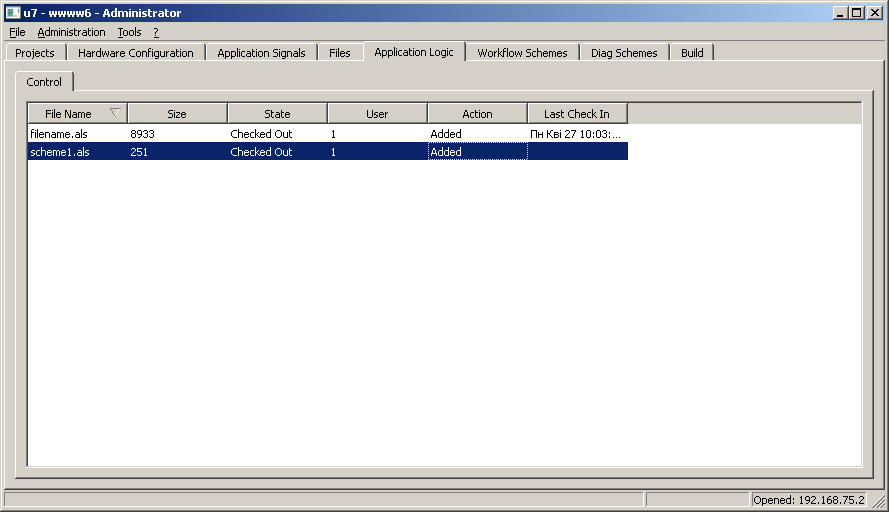
In case of error in FBL-element syntax, new dialog window will be shown (fig. 7.9). In this window place of error will be written.



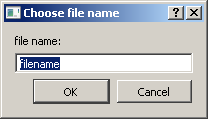
|  |  |
| --- | --- |
| Figure | 7.9 |

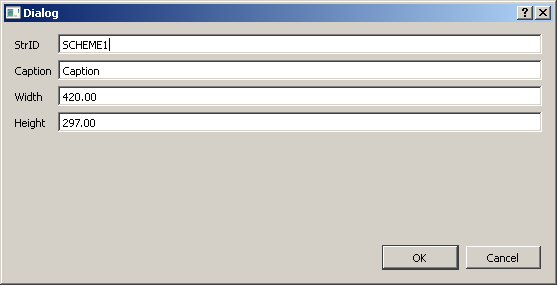
# Application Logic Editor

* 1. To edit application logic, switch to the “Application Logic” tab. This tab contains a list of files with application logic schemes, as shown on the figure.

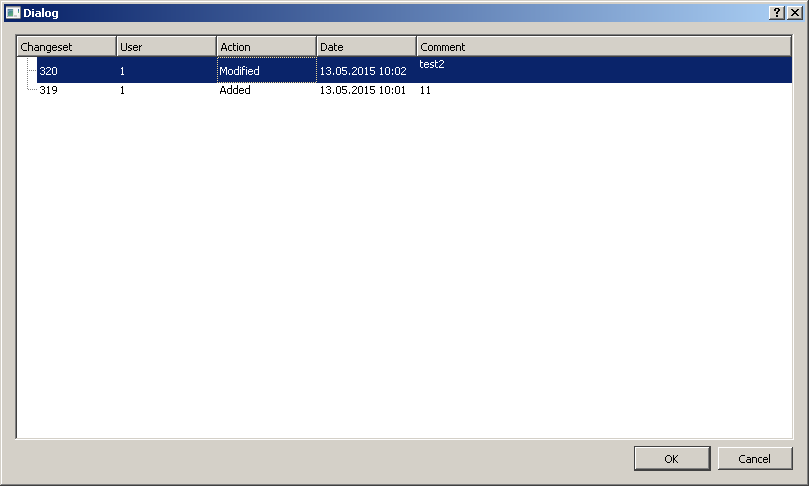


* + 1. To add a new scheme, press the right mouse button on the list and choose the “Add File.,,” item. Enter the file name in the dialog shown on the figure. Enter scheme parameters in the next dialog shown on the figure. A new scheme will be created.

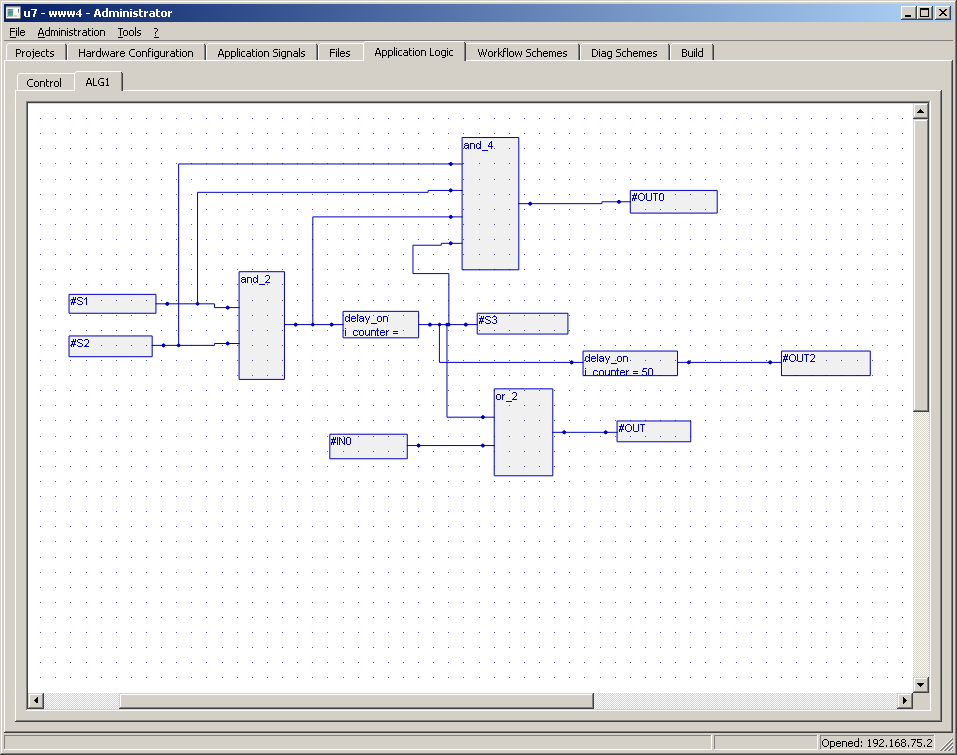




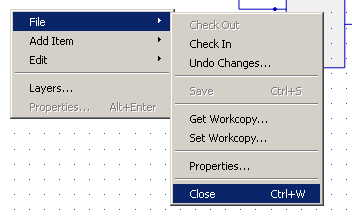
* + 1. To check out an existing scheme for editing, click the right mouse button on a file name and select “Check Out…” command. After editing finished, to check in an existing item and place changes to server, choose “Check In…” command. To undo changes, use the “Undo Changes…” command.
    2. To delete an existing file, select “Delete File…” command.
    3. To view file contents, click the right mouse button on a file and select “View…” command. A window with the file check-in history will appear, shown on a figure. Choose a file changeset to view and press OK.



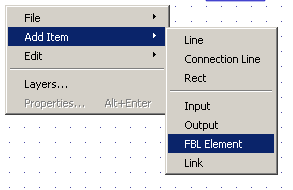
* + 1. To edit an existing scheme, click the right mouse button on the file name in a list and choose “Open…”. Only checked out files can be edited.
  1. When a scheme file is opened for viewing or editing, a new tab with file contents is created. In view mode, no changed can be made to the file. An application logic editing tab is shown on the figure.

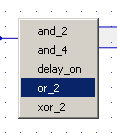


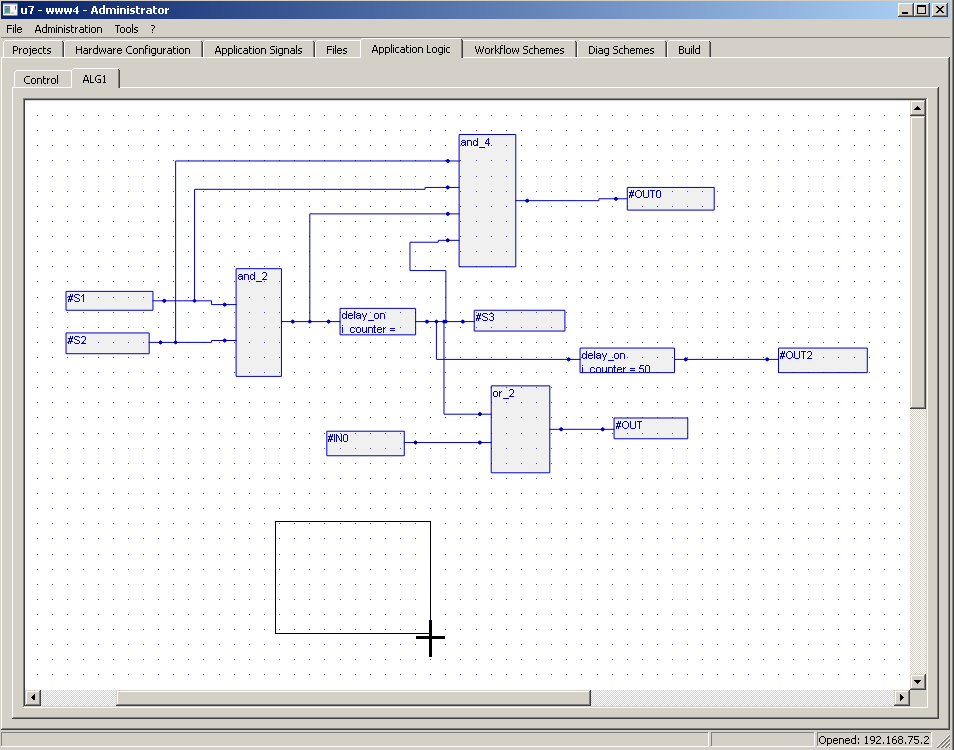
* + 1. For opened scheme control, click the right mouse button on the white space and choose the “File” submenu. A list of commands is shown on the figure. To save changes, choose the “Save…” command. To close a tab, choose the “Close” command.

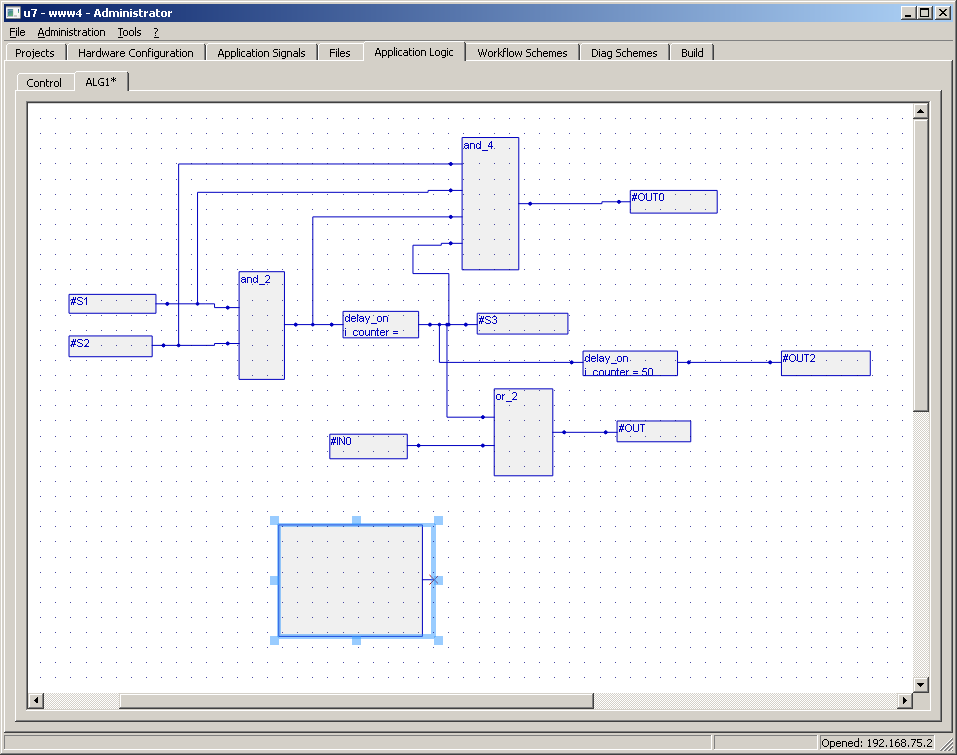


* + 1. To add a new item to the scheme, click the right mouse button on the white space and choose the “Add Item” submenu. A list of available items will appear as shown on the figure. If user chooses the “FBL Element” item, an additional menu will appear to choose the type of an element, as shown on the figure. Select the necessary element in the menu, press the left mouse button on the scheme white space, move the mouse and release the left button, as shown on the figure. A newly created item will appear, as shown on the figure.

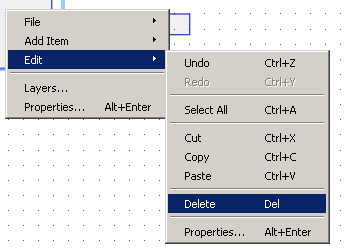




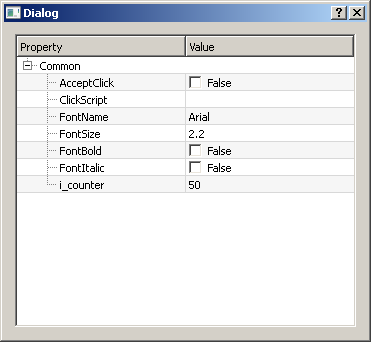




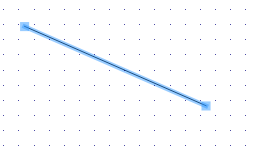
* + 1. To move an item, press the left mouse button on it and move the mouse. Then release the button.
    2. For item editing commands, select the “Edit” submenu in the scheme popup-menu. The “Edit” submenu contains “Undo”, “Redo”, “Select All”, “Delete” and clipboard commands, as shown on the figure.



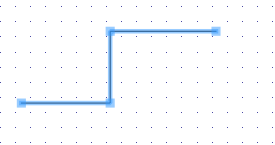
* + 1. To edit the item’s properties, select an item and choose the “Properties” command in the “Edit” submenu. The properties dialog will appear, as shown on the figure.



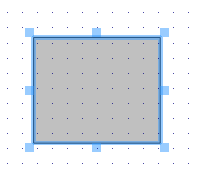
* 1. Static scheme items are used to display static scheme elements. They do not change their appearance.
     1. A “Line” element is a simple line, as shown on the figure.



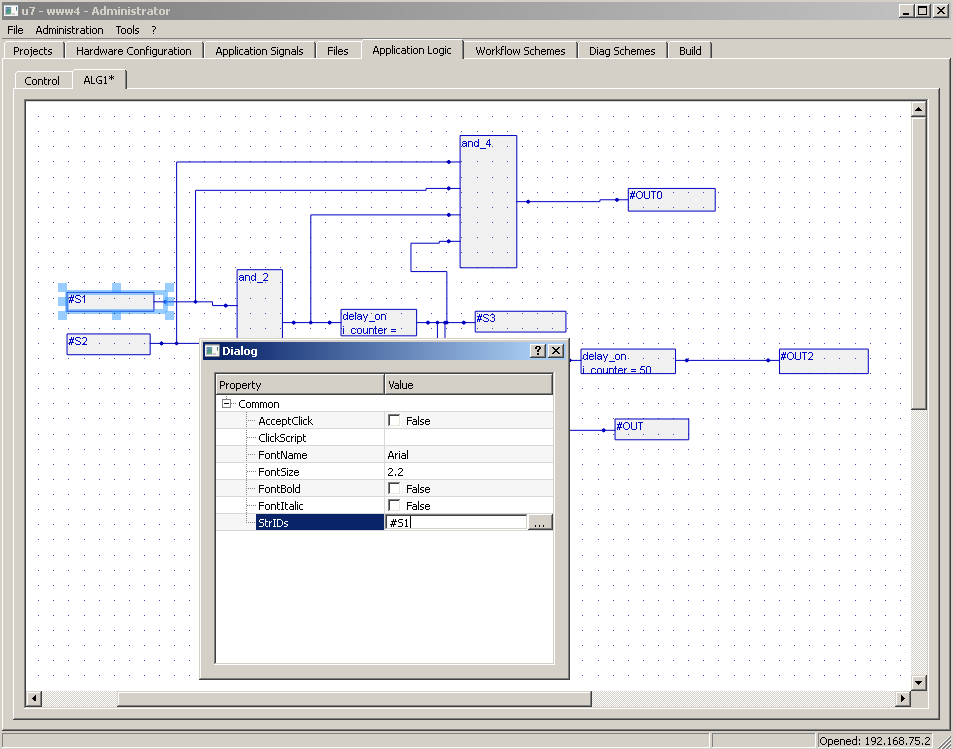
* + 1. A “Connection Line” is a line contains a few sections. To add a connection line to the scheme, select the “Connection Line” command in the menu. Then press the left mouse button and move the mouse. To make a connection point, hold the left mouse button and click the right mouse button. Then move the mouse again. After adding all connection, release the left mouse button. A connection line is shown on the figure.



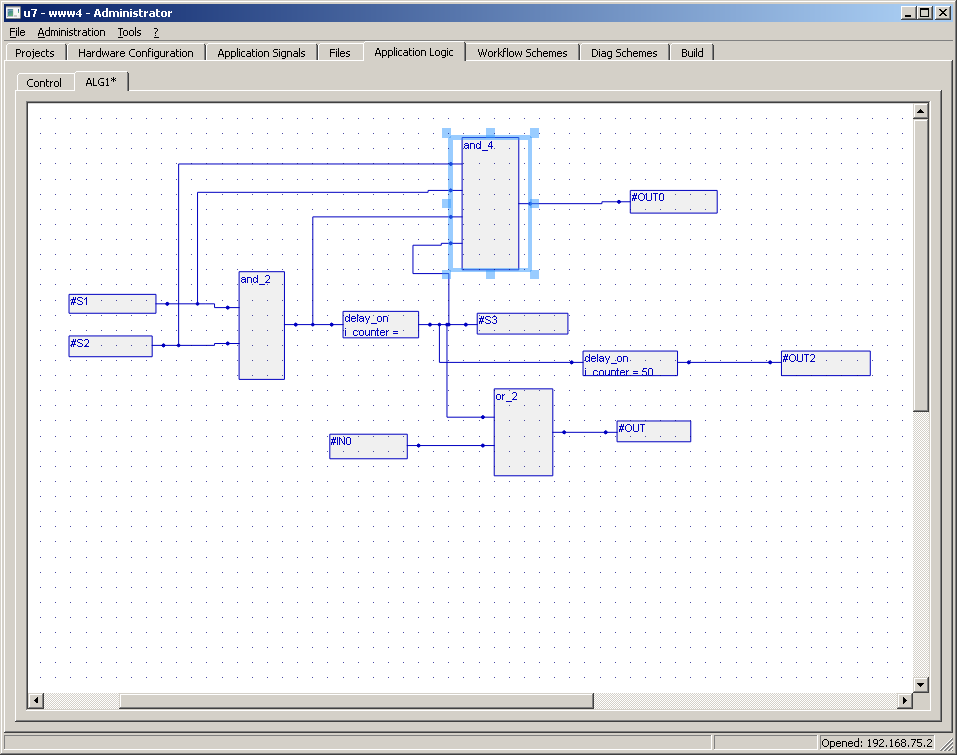
* + 1. A “Rectangle” item is a simple rectangle, shown on the figure.



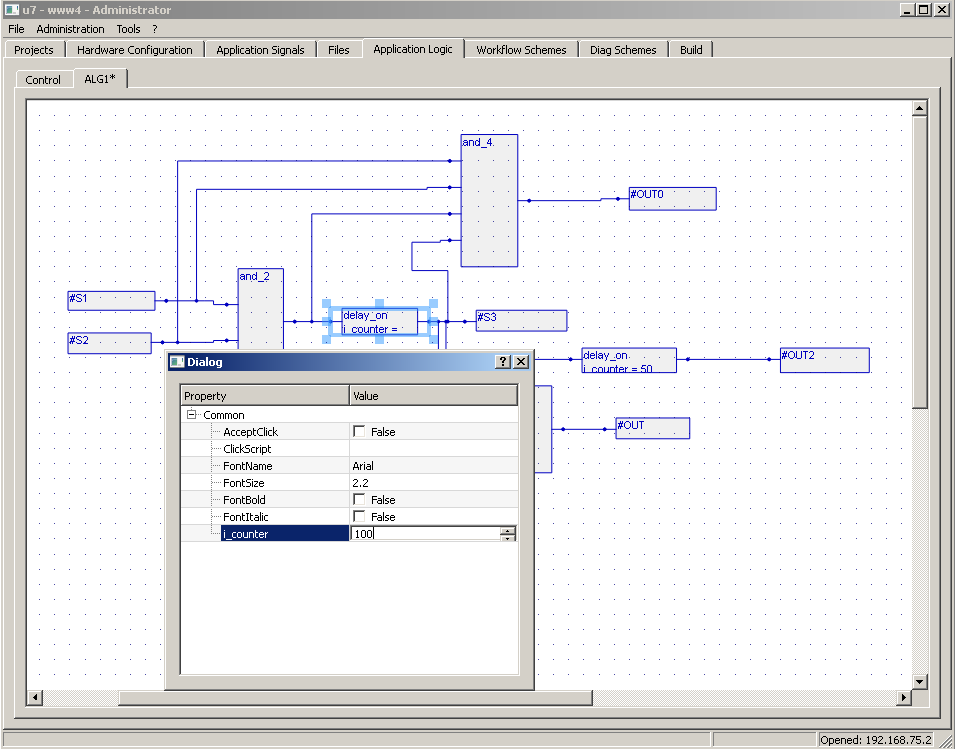
* 1. Dynamic item describe application logic itself.
     1. “Input” item is selected on the figure. It contains input signal of the algorithm. The “StrIDs” property conatins a signal identifier this item belongs to.
     2. “Output” item is similar to “Input” item. It contains output signal of the algorithm. “StrIDs” property contains a signal identifier this item belongs to.



* + 1. FBL elements describe application logic. They are described in FBL element library and are implemented in the hardware. Most FBL elements have inputs and outputs, as shown on the figure.

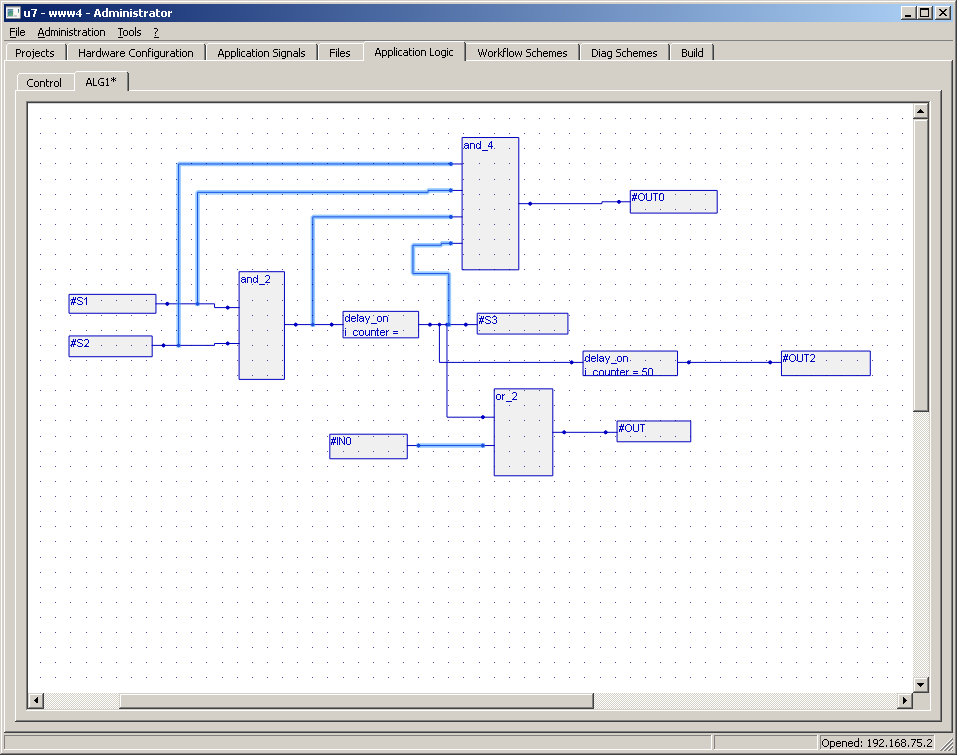


FBL elements may also contain parameters that user can change. For example, a “Delay” FBL element contains the “i\_counter” property that user can change to control the delay time. An example is shown on the figure.



* + 1. A “Link” item is a connection item that connects input, output and FBL items. An example is shown on the figure.

NOTE – to compile the project, there should be no unconnected pins.



# Редактор технологических видеокадров (Workflow schemes)

# Редактор диагностических видеокадров (Diag schemes)

# Компиляция проекта (Build)

* 1. Файлы, создаваемые при компиляции проекта
  2. Возможные ошибки при компиляции проекта