EN CE UK CA

akytec



akYtec ALP v3.1

Programming software

User manual

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

akYtec ALP v3.1_3-EN-130795-2.4
© All rights reserved
Subject to technical changes and misprints

Contents

1. Introduction 5
1.1. System requirements 5
1.2. Terms and abbreviations 5
2. Installation 7
3. User interface 10
3.1. Main menu 10
3.2. Toolbars 12
3.3. Library Box 13
3.4. Property box 16
3.5. Variable Box 16
3.6. Workspace 18
3.7. Status bar 19
3.8. Display manager 20
4. General information 23
4.1. Program execution 23
4.2. Project creation 23
4.2.1. Protecting a project with a key 24
4.3. Program creation 26
4.3.1. Text field 28
4.3.2. Standard variable block 29
4.3.3. Constant block 33
4.3.4. Delay line 34
4.3.5. Network variable 35
4.3.6. Read from/Write to FB 36
4.3.7. Conversion block 37
4.3.8. Arrange elements 38
4.3.9. Execution sequence 38
4.4. Display programming 39
4.4.1. Monochrome text LCD 40
4.4.2. Graphic color LCD 45
4.5. Simulation 52
4.5.1. Visualization Simulation 55
4.5.2. ST code simulation 56
4.6. Connection to device 60
4.7. Upload project to device 61
4.8. Online debugging 62
4.9. Project information 65
4.10. Component manager 67
4.11. Macro development 70
4.12. Using ST function 79
4.13. ST function blocks 84
5. Device configuration 89
5.1. Display 89
5.2. Clock 90
5.3. Data exchange 91

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3.1. Interfaces 91
5.3.2. Modbus working 97
5.4. Extension modules 115
5.5. Inputs and outputs 116
5.6. Password 117
5.7. Connecting to akYtec Cloud 118
6. Variables 120
6.1. Data types 127
6.2. Service variables 128
6.3. Network variables 129
6.4. Binding variables to parameters 130
6.5. Copy-paste variable block 131
7. Library 133
7.1. Functions 133
7.1.1. Logical operators 133
7.1.2. Mathematical operators 136
7.1.3. Relational operators 139
7.1.4. Bitshift operators 141
7.1.5. Bit operators 143
7.2. Function blocks 145
7.2.1. Triggers 145
7.2.2. Timers 147
7.2.3. Generators 151
7.2.4. Counters 152
7.2.5. Analog 155
7.3. Project macros 162
7.4. ST functions 163
7.5. ST function block 166
7.6. Display elements 170
7.6.1. Text box 171
7.6.2. I/O box (INT/REAL) 171
7.6.3. I/O box (BOOL) 172
7.6.4. Dynamic box 173
7.6.5. ComboBox 174
7.7. Basic graphic elements 175
7.7.1. Text 181
7.7.2. Indicator 182
7.7.3. Progress bar 183
7.7.4. Dynamic box 184
7.7.5. REAL/INT input/output 187
7.7.6. Time input/output 191
7.7.7. Input/output of IP address 192
7.7.8. Line 194
7.7.9. Polygon 195
7.7.10. Circle 196
7.7.11. Button 197
7.7.12. Switch 199

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7.13. Switch group 201
7.7.14. Image 204
7.7.15. Menu 206
7.7.16. Chart 208
8. Device 213
8.1. Device information 213
8.2. Cycle time 214
8.3. Firmware update / repair 214
8.4. Calibration 215
8.4.1. Input calibration 216
8.4.2. Output calibration 217
8.5. Change target device 217
9. Plugins 219
9.1. Replication master 220
9.1.1. Writing a program to the device via the Replication master on Windows 224
9.1.2. Writing a program to a device via the Replication master on Linux 227
9.2. Exporting the device to akYtec Cloud 228
10. Keyboard shortcuts 230
11. Program examples 232
11.1. Task 1: Light switch with automatic switch-off 232
11.2. Task 2: Mixer control 234
11.3. Task3. Direct connection of PR103 to akYtec Cloud 237
12. ST language 242
12.1. Syntax 242
12.1.1. Bitwise access to variables 242
12.1.2. Using functions in other ST program elements 242
12.1.3. Using one function block in another 243
12.1.4. Comments in ST editor 244
12.1.5. Copying ST elements between projects 244
12.2. Documentation in the ST editor 245
12.3. Data types 246
12.3.1. Reserved keywords 247
12.3.2. Arrays 248
12.4. Language structures 250
12.4.1. Operations 251
12.4.2. Assignment operation 253
12.4.3. IF statement 253
12.4.4. CASE statement 254
12.4.5. RETURN statement 255
12.4.6. FOR statement 255
12.4.7. WHILE statement 257
12.4.8. REPEAT UNTIL statement 257
12.5. System functions 258
12.5.1. Date and Time functions 258
12.5.2. Mathematical functions 260
12.5.3. Logarithmic functions 262
12.5.4. Trigonometric functions 264

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.6. System Function Blocks 265
12.6.1. Triggers 265
12.6.2. Timers 269
12.6.3. Generators 275
12.6.4. Counters 277

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1 Introduction

ALP is a programming software for programmable devices of akYtec GmbH. ALP uses graphical language FBD (Function block diagram) and ST (Structure Language) as programming languages, according to IEC 61131-3. The software enables simulation and debugging of the created program and its upload to the non-volatile memory of the device.
Created project contains minimum one circuit program and device configuration. The first workspace contains the main circuit program. In separate workspaces user can create macros as circuit programs. If the target device has a display, it can be programmed using display forms in separate workspaces. Only one project can be opened in ALP at a time.
akYtec ALP has different functions for the following groups of devices:

– first generation devices (PR100, PR102, PR200 and SMI200)
– second generation devices (PR103, and devices with a graphical color LCD (PR205, PR225))

Basic akYtec ALP functionality is available for all devices, functionality and interfaces for devices on the second generation are not available for first generation devices.

1.1 System requirements

Operation systems:

– Windows 7 (SP1+)
– Windows 8.1
– Windows 10
– Windows 11

System libraries:

– Microsoft .NET Framework 4.8
– Microsoft .NET Desktop Runtime 6.0.8
– Microsoft Visual C++ 2015-2022

Recommended hardware requirements:

– 3.2 GHz processor
– 4 GB RAM
– 700 MB available hard disk space
– Free USB port
– Keyboard and mouse
– Screen resolution 1280×800

Internet connection is required for:

– Software update
– Device firmware update
– Macros download in Component manager

1.2 Terms and abbreviations

– **ALP** — programming software for programmable devices of akYtec GmbH.
– **EEPROM** — electrically erasable programmable read-only memory of the device.
– **FBD (Function Block Diagram)** — graphical programming language supported by IEC 61131-3.
– **Function** — Structural program unit with one return value. The function does not store information about its internal state, i.e. if the function is called with the same input values, it returns the same output value.
– **Function block** — Structural program unit with internal memory and one or more output values. It is used in program as an instance, i. e. a copy with its own memory.
– **LoRa (Long Range)** — wireless data transmission technology.
– **LoRaWAN (Long Range Wide Area Network)** — The protocol that manages communication over the LoRa network.
– **Macro** — user function block.
– **Program cycle** — execution time of the circuit program, which depends of its complexity.
– **Project** — user application created for a specific device with ALP software, includes the circuit program.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **Slot** — a connection point in the device, enabling the attachment of an interface or an extension module.
- **ST (Structured Text)** — a text programming language supported by IEC 611131-3.
- **Target device** — device type for which the project is created.
- **Workspace** — graphic area in the user interface for the program creation, modification and debugging.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2 Installation

1. Download and run the *akYtecALP.exe* file on your PC.
2. Select setup language.

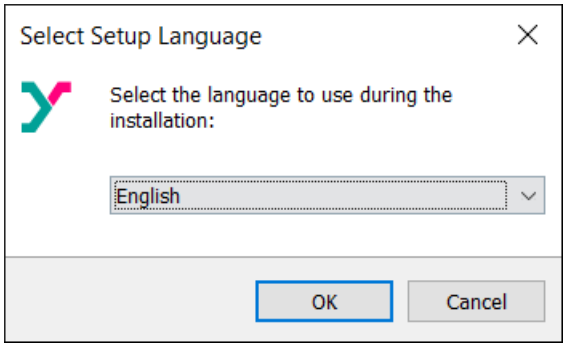


Fig. 2.1 Setup language selection

DANGER
Limitations: ALP must be installed and run under the same user account.
During installation, ALP creates a folder in the AppData directory (for example, *C:\Users \user\AppData\Roaming\akYtec\akYtec ALP*) that contains files required for the proper operation of the program. This directory is accessible only to the user who performed the installation. This behavior is a limitation of the Windows operating system

3. The installation wizard window will open. Review the information and click the **Next** button.

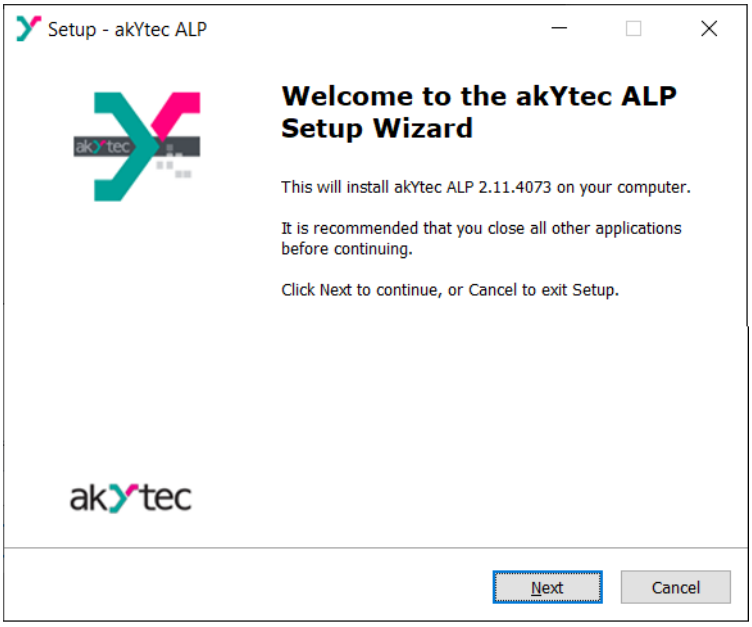


Fig. 2.2 Installation Wizard Window

4. Select the installation folder for akYtec ALP and the folder for the Start menu shortcut. Click the **Next** button.
5. If necessary, enable the corresponding checkboxes to create a desktop shortcut, create a quick launch panel shortcut, associate files with the .owl and .owle extensions with akYtec ALP, and install the .NET6 SDK library for creating a Replication Wizard for Linux OS. Click the **Next** button.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

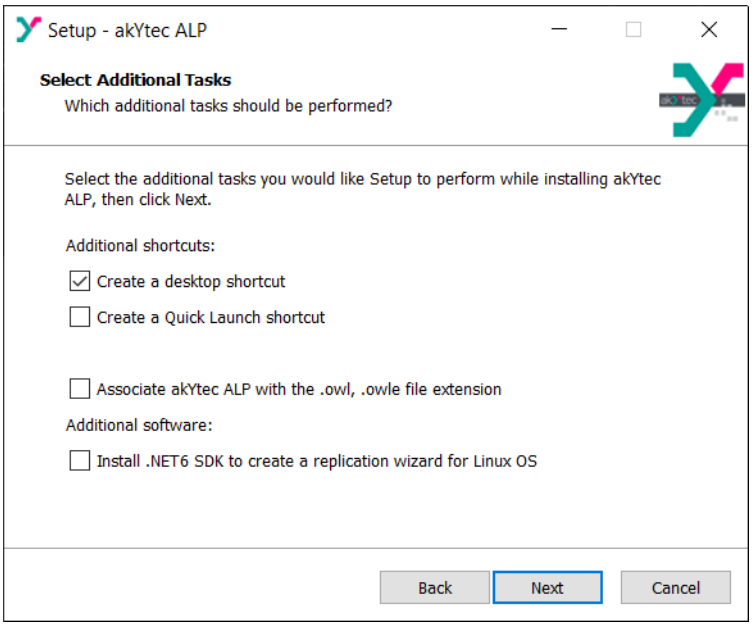


Fig. 2.3 Selecting Additional Tasks

6. Review the installation information and click the **Install** button.

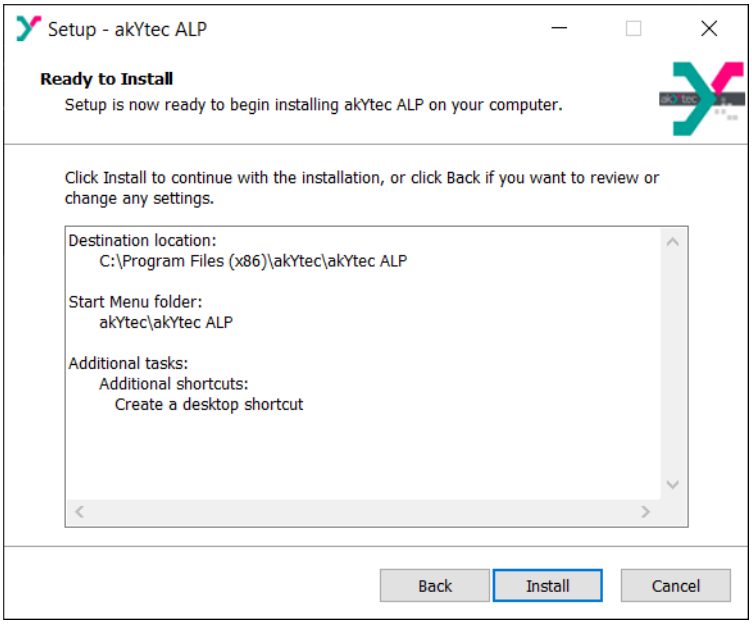


Fig. 2.4 Installation launch window

7. A window will open displaying the installation progress.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

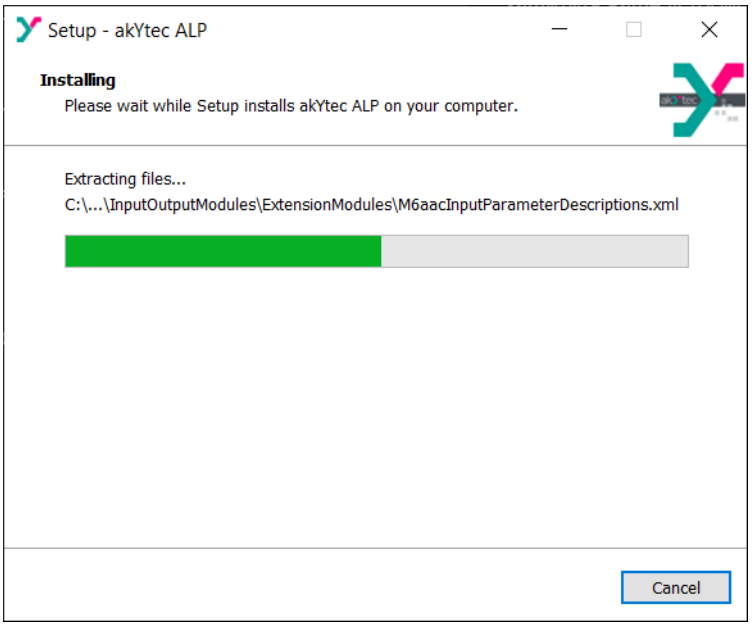


Fig. 2.5 Program installation process

8. Wait for the installation to complete and, if necessary, check the **Launch akYtec ALP** checkbox.

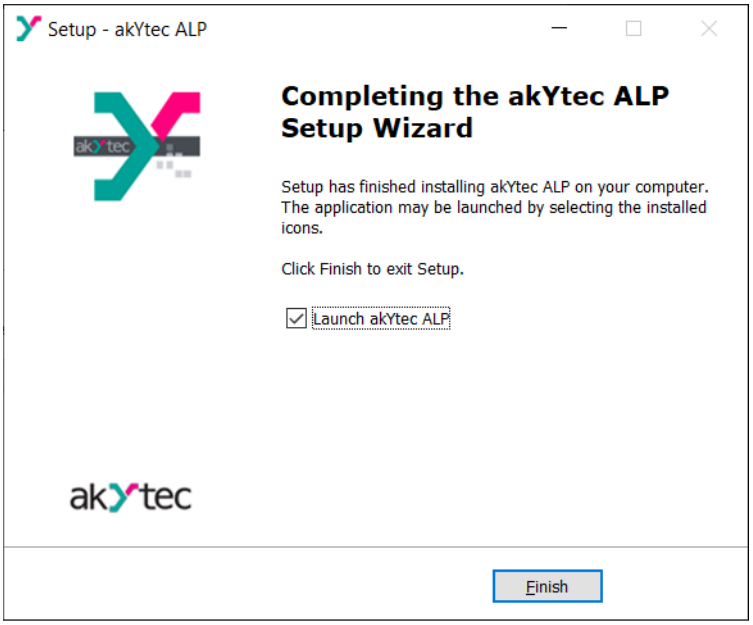
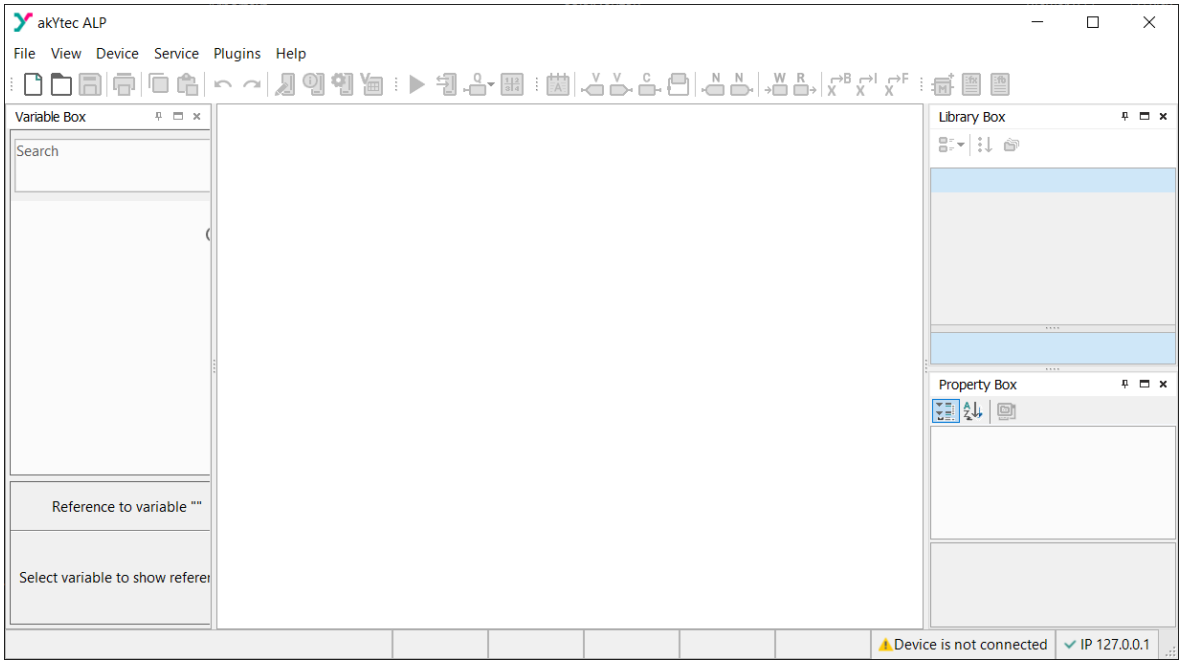


Fig. 2.6 Installation completion

9. Click the **Finish** button.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3 User interface



1. *Title bar* — shows the name of the software and the path to the open project file.
2. *Menu bar* — contains following groups: File, View, Device, Service, Plugins and Help.
3. *Tool bars* – Standard, Service and Insert: quick access to the essential functions of ALP.
4. *Library box* – the panel shows all the functions, function blocks, and macros that can be used in the project (drag-and-drop using).
5. *Property Box* — panel for viewing and modifying properties of the selected project element
6. *Workspace* — graphic area for creating and modifying circuit programs, ST programs, display elements or display forms.
7. *Status bar* — shows status and error messages, target device memory usage, status of the connected device and the programming interface.
8. *Display Manager* — a tool to program the display information (available only for the devices with display).
9. *Variable Box* — this panel shows all project variables with their parameters and references (drag-and-drop using).
10. *Component manager* — special tool in a separate window to access Online Database and to add the elements from the Online Database to the offline library or to the project library. Internet connection is needed.

3.1 Main menu

File

New project	Open a new project. The current project will be closed
Change target device	Change the target device in the project
Open project	Open a previously saved project
Save active workspace	Save the active workspace
Save project	Save the current project
Save project as…	Make a copy of the project in a different folder or under a different name

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Create key file…	Create a file with a key to protect the project from unauthorized access
Project information	View and modify the *information about the project*
New macro	Open a new macro in the separate *workspace*
Import	Import a macro, an ST function or an ST function block from a file into the project library
Export	Save the current macro, ST function or ST function block as file
Component manager	Open the *Component manager* interface
Print	Open the dialog to set the print options and print the active workspace
Recent projects	List of recently opened projects
Exit	Close ALP

View

Undo	Undo the last action
Redo	Redo the last undone action
Status indictors	Add / remove the in indicators to / from the *status bar*
Library Box	Show / hide *Library Box*
Property Box	Show / hide *Property Box*
Variable Box	Show / hide *Variable Box*
Display manager	Show / hide *Display Manager*
Reset layout	Restore the default panel layout

Device

Transfer application to device	Upload the current project to the device memory
Firmware update	Update the firmware of the connected device
Device information	*Information* about the software, the target device and the connected device
Variable table	The *editable table* of the project variables with their parameters
Calibration	Start analog I/O *calibration*
Device configuration	Open *Device configuration* interface
Port configuration	Settings of the *programming interface*

Service

Arrange elements	Function blocks of the same type are automatically renumbered in the workspace from top to bottom and from right to left
Simulation mode	Start / stop *simulation* mode

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Language	Select the interface language
OFFLINE mode	Activate OFFLINE mode

Help

Automatic update check	If activated, the update check is performed on program startup
Check for updates...	Check for software updates
Help	Open Help window
Version history	Running list of the changes have been made in all software versions
About Software	Information about the current software version

3.2 Toolbars

Standard



	New project	Open a new project. The currently opened project will be closed
	Open project	Open a previously saved project
	Save project	Save the current project
	Print	Set the print options for the active workspace
	Copy	Copy the selected element
	Paste	Paste the copied element
	Undo	Undo the last action
	Redo	Redo the last undone action
	Transfer application to device	Upload the current project to the device memory
	Device information	Information about the software, the target device and the connected device
	Device configuration	Device configuration
	Variable table	Table of project variables

Service



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

	Simulation	Start / stop simulation
	Online debugging	Start / stop online debugging
	Execution order	Change the execution order for the outputs or delay lines on a circuit program or in a macro
	Arrange elements	Function blocks of the same type are automatically renumbered in the workspace from top to bottom and from right to left

Insert



	Text field	Text field to comment the program
	Variable output block	Block for writing a variable into the program
	Variable input block	Block for reading a variable from the program
	Constant block	Constant value
	Delay line	Feedback with one-cycle delay
	Network variable output block	Block for writing a variable via network
	Network variable input block	Block for reading a variable via network
	WriteToFB block	Connects the output value of the block to the selected parameter of the selected function block. Allows you to edit the parameter
	ReadFromFB block	Connects the output value of the block to the selected parameter of the selected function block. Allows you to read the parameter
	Conversion to BOOL	Conversion of any values to a BOOL value
	Conversion to INT	Conversion of any values to an INT value
	Conversion to REAL	Conversion of any values to a REAL value
	New macro	New user macro
	New ST function	New user function in ST language
	New ST function block	New user function block in ST language

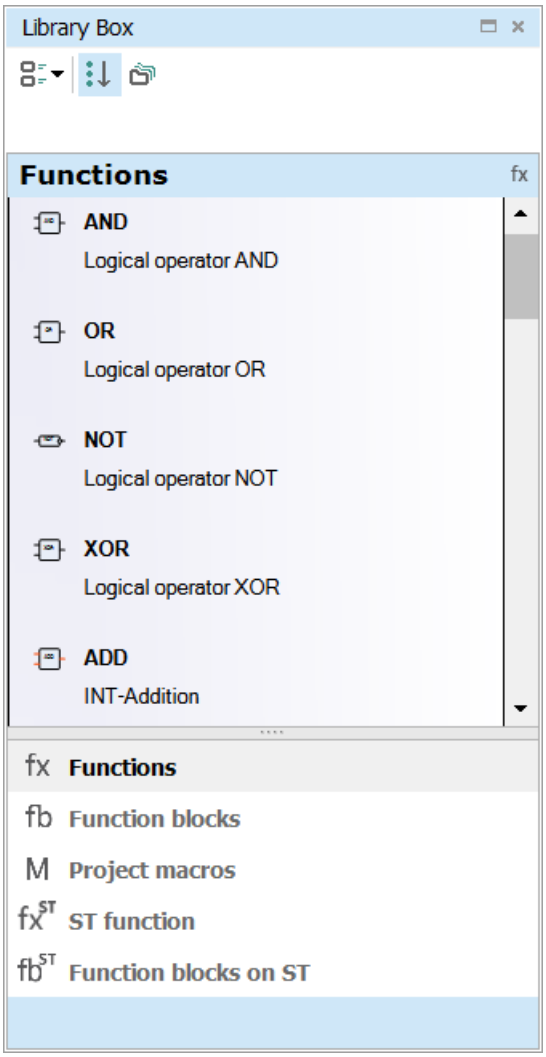
3.3 Library Box

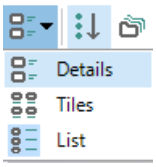
The ***Library Box*** panel is a summary of program blocks available in the project. The panel consists of libraries:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– *Functions*
– *Function blocks*
– *Project macros*
– *ST functions*
– *ST function bocks*

Select an item on the lower toolbar of the panel to show the respective content.
The standard position of the panel is the upper right window corner (can be changed).

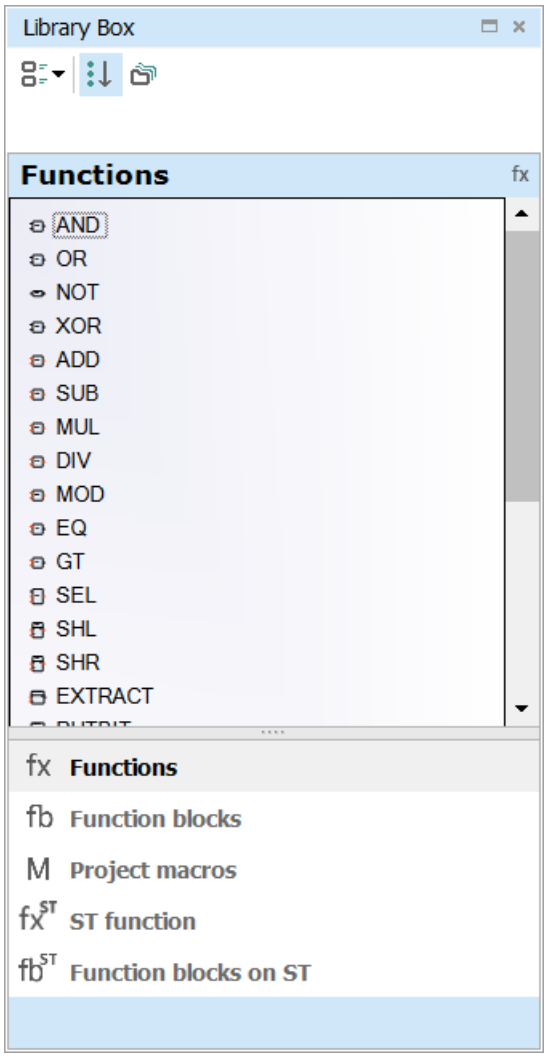




The panel view can be changed using the icons on the upper toolbar.

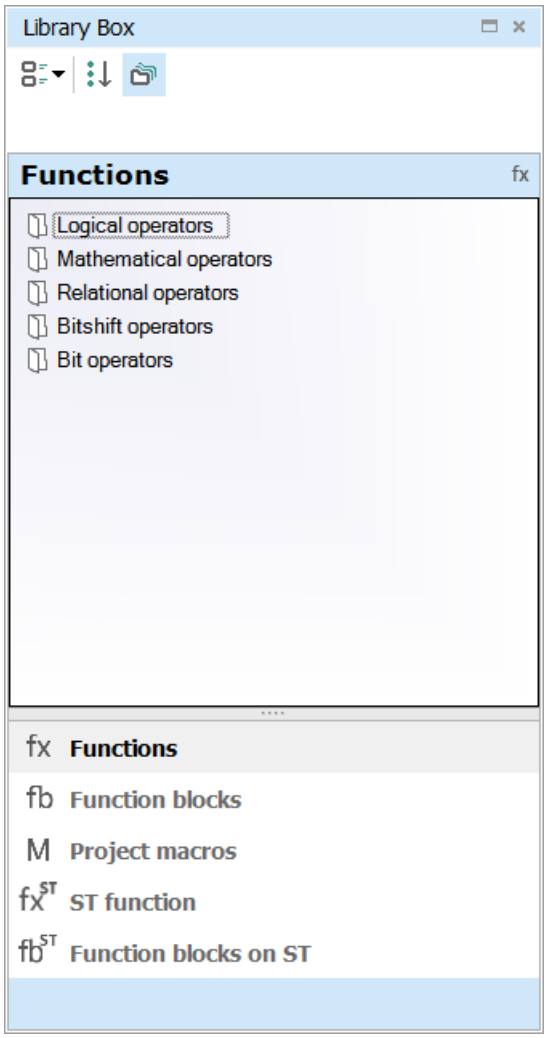
– Click the icon **Show all** to show all the blocks of the selected library.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



– Click the icon **Show grouped** to show the blocks of the selected library group. Double-click the folder to open it.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



For description of the library groups and individual blocks see section *Library*.

3.4 Property box

The panel Property Box is used to view and modify the parameters of the program elements. Select the element to view its properties. If no item is selected, the panel displays the properties of the workspace.
The standard position of the panel is the lower right window corner (can be changed).
The parameters are shown grouped by categories by default.

To show them in alphabetical order, click the icon .

To show them grouped, click the icon .
Select the parameter input field to edit its value.

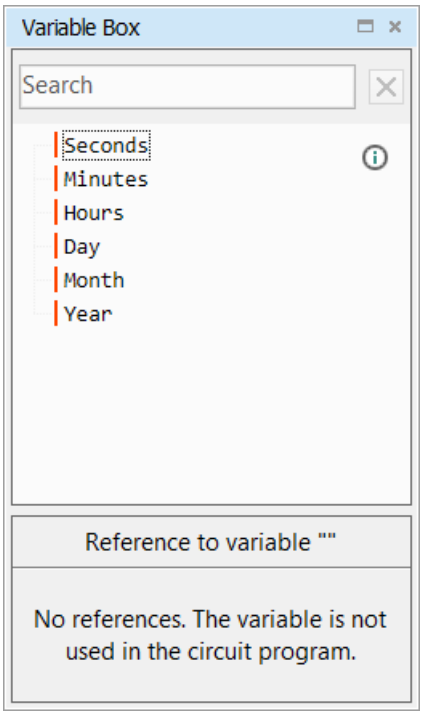
3.5 Variable Box

The panel ***Variable Box*** shows the project variables from the variable table.
The standard position of the panel is the upper left window corner (can be changed).
The variable properties are availiable in a tooltip text.

Variable block in the workspace

Drag-and-drop a variable to place it in the circuit program as an input block.

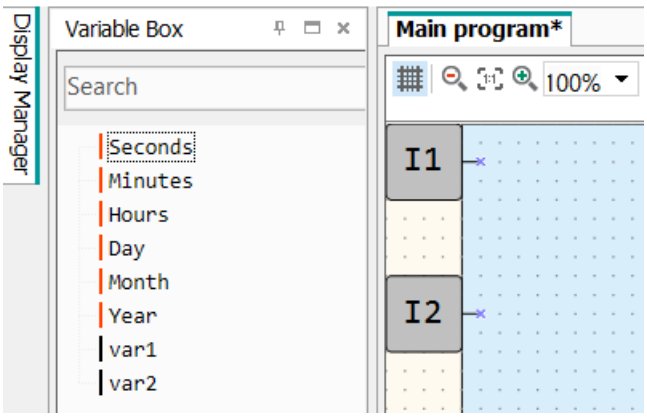
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



To use a variable as an output block, hold the Shift key pressed as you drag-and-drop the variable. If a variable is drag-and-dropped onto a connection pin of a block, the created variable block is connected to this connection pin.

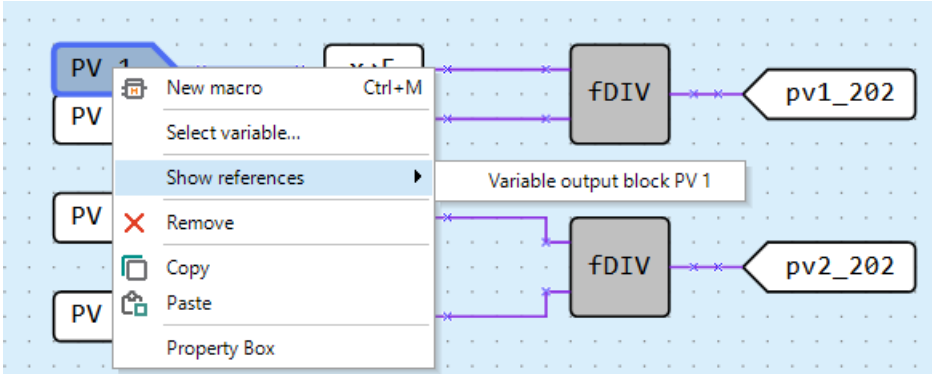
References

The variable references are shown as links in the lower panel part. If you click on the link, the block to which the variable is referred is highlighted in the workspace.



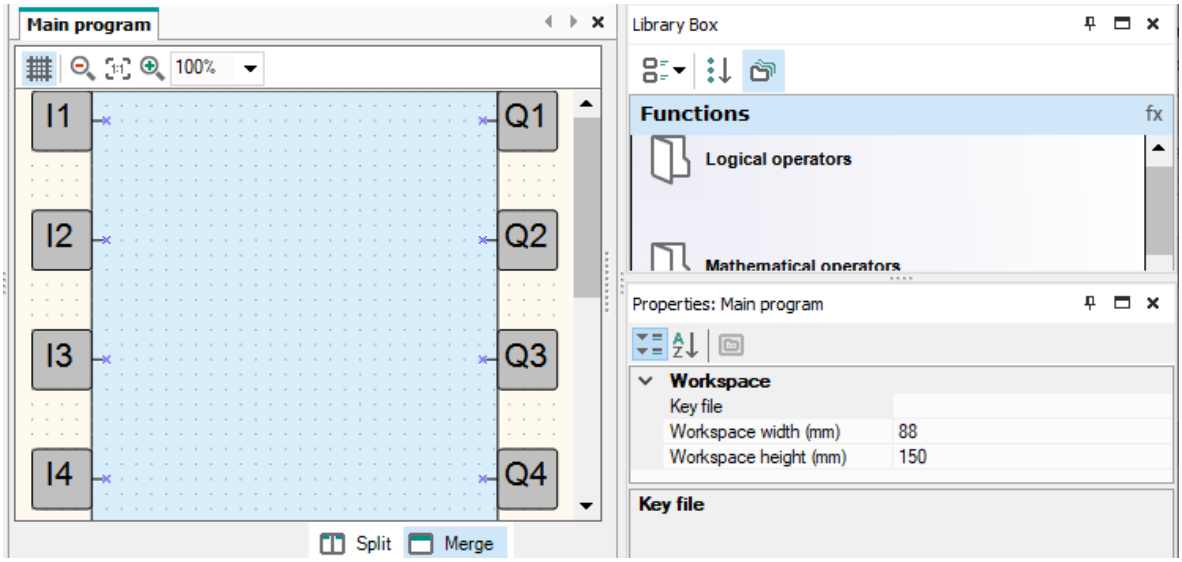
If a variable is used at more than one place in the project, all the references can be viewed with the item ***Show references*** in the variable block context menu. Click on the link to view the reference.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



3.6 Workspace

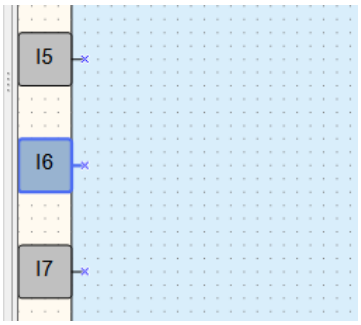
When a project is open, the workspace with the tab ***Main program*** is shown in the middle of the window.
If your device supports ST functions, the **Function Editor** tab appears.



Circuit program is built by placing program blocks and connecting lines between them in the workspace. The size of the workspace can be changed in Property Box. The inputs (left) and outputs (right) are signed as follows:

– **Ix** – digital inputs
– **AIx** – analog inputs
– **Qx** – relay outputs
– **AOx** – analog outputs
– **Fx** – LED indicators

The numbers (x) correspond to the ordinal numbers of physical I/O points of the target device. I/O points can be moved up and down along the workspace by drag-and-drop.

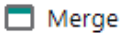
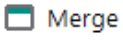


Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Workspace toolbar

	**Show / Hide grid**	Show / hide vertical and horizontal rulers and a grid in the workspace. If the grid is visible, the blocks and connecting lines are snapped to the grid
	**Zoom –**	Decrease the workspace by 10 %
	**Original size**	Return to the original size (100 %)
	**Zoom +**	Increase the workspace by 10 %

You can set the required scale using the drop-down menu to the right of the buttons described above.

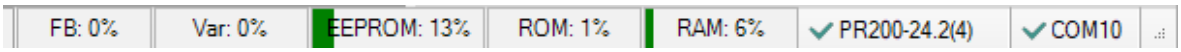
The icons  and  are located on a toolbar below the workspace. Use the icon  to show the same circuit program in two workspaces. It can be useful if the program is too large and you want to view two different parts of the program at the same time. Use the icon  to return to one workspace.

3.7 Status bar

Status and error messages are displayed in the left field of the status bar.

View

The right field of the Status bar has different status indicators that show information about the memory usage of the target device (resource indicators), status of the connected device and the programming interface. Contents of the status bar depend on the type of the target device.



Resource indicators show the used resource in percent of the total available amount. Move the mouse cursor over the indicator to see the absolute amount of the resource.

Indicators

If the device is connected, Status bar contains the following information:

– **FB** — number of the used and available function blocks.
– **Var** — number of the used and available variables.
– **Stack** — percentage of the available and used stack memory. The stack is used for intermediate calculations in the program.
– **Sys EEPROM** — percentage of used and available system non-volatile memory. The indicator is full if the project uses network variables, if variables are linked to visualization/device parameters, or if the project uses expansion modules.
– **EEPROM** — percentage of used and available non-volatile memory. The indicator is full if the project uses standard non-volatile variables.
– **ROM** — used and available ROM memory.
– **Sys RAM** — used and availiable system RAM memory. This indicator appears when the RAM is over 80 % full.

> **NOTE**
> *ALP software automatically calculates the available resources of the device and shows a warning if the critical value is reached.*

– **RAM** — used and available RAM memory.
– **System Visualization ROM** — the available visualization ROM of the device as a percentage of the total volume (only for devices with a graphic color LCD).

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **User Visualization ROM** — the available visualization ROM for bitmap images (only for devices with a graphic color LCD).
- **Visualization RAM** — the available visualization RAM of the device as a percentage of the total volume. The calculation is performed for the most heavily loaded screen (only for devices with a graphic color LCD).
- **Device** — device model, if connected

NOTE
*Click the Device indicator to switch to **OFFLINE** mode. In this mode the connection to device is interrupted. The next click disables the **OFFLINE** mode.*

- **COMx** — the selected COM port number (programing interface)

NOTE
*Click on the indicator to open the window **Port settings**.*

NOTICE
If any of the indicators overflow, writing a program to the device is disabled.

You can customize the number of indicators on the Status bar in the ***View*** menu.

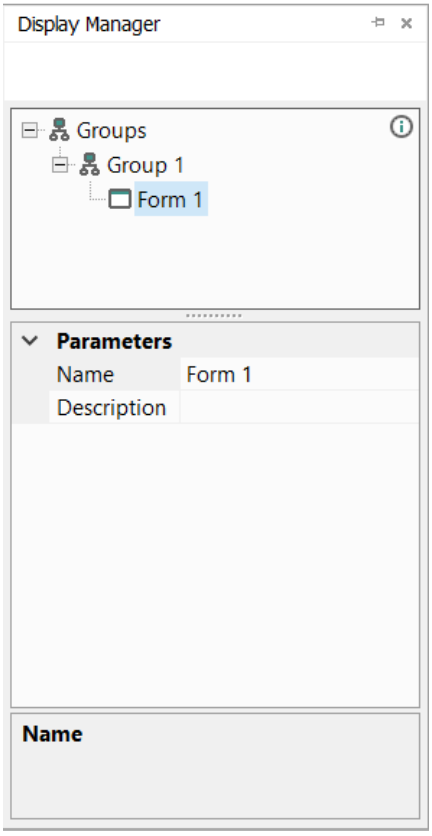
OFFLINE mode

OFFLINE mode interrupts the connection between ALP and the device.
OFFLINE mode can be activated / deactivated using the menu item ***Service > OFFLINE mode*** or by clicking the status indicator ***Device*** . The next click disables the OFFLINE mode.
For more details see section *Upload project to device*.

3.8 Display manager

If the target device has a display, you can program the displayed information using one or more display forms. For further details about display programming see section *Display programming*
The programming is carried out using the programming tool ***Display Manager***. The tab Display Manager is located in the upper left corner of the window. Click the tab to open the panel. The panel contains a toolbar, a hierarchical structure (tree) of display forms and the parameters of the selected object.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

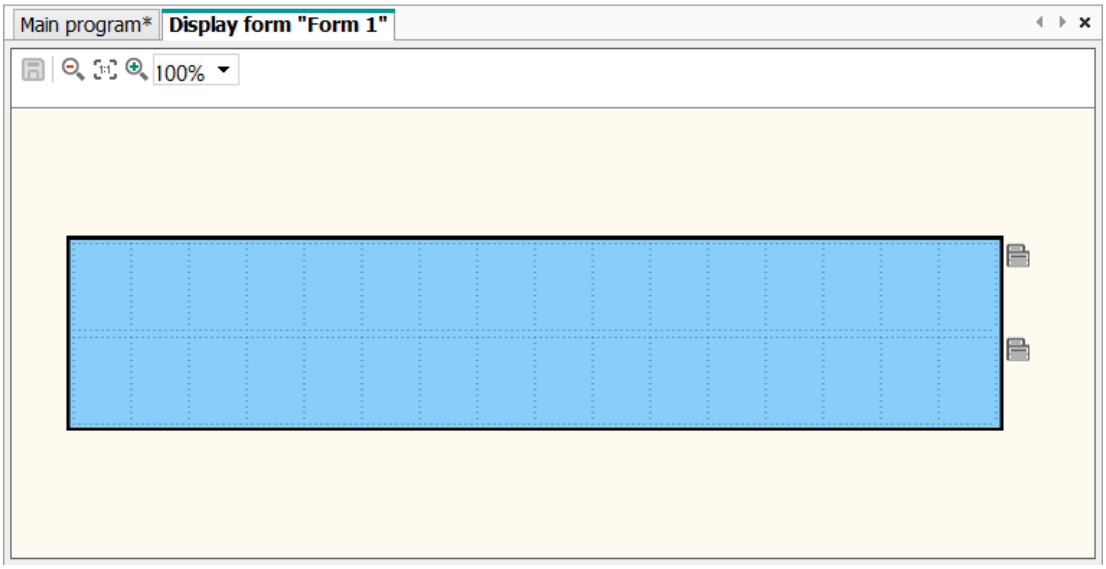


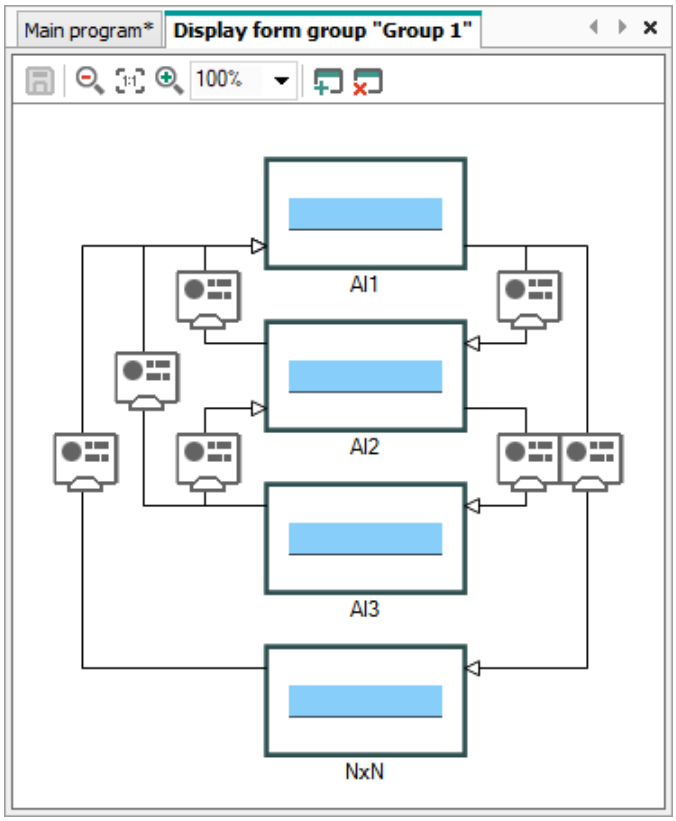
The parameters of the selected display form are shown in the lower part of the panel.
To program the selected form, open it in a separate workspace ***Display Editor***, using the context menu or double-click the form in the group.

	Add display form
	Delete display form
	Edit display form

The workspace shows the selected display form with the icons  to the right of it, which are used to change the number of displayed rows. The rows displayed first are bold outlined.
Working with screens is discussed in detail in the *section 4.4.1*.
If there are more than one display forms, you should specify “jumps” between them so that the operator can switch between forms to see the required information. It can be done in a separate workspace ***Structure Editor***, which represents the graphical structure of display forms with “jumps” and their conditions. To open it, use the command ***Edit group*** in the group context menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000





Workspace Structure Editor has the following functions (buttons at the top):

	**Save workspace**
	**Zoom out by 10 %**
	**Original size**
	**Zoom in by 10 %**

The scale can be changed with the drop-down menu to the right of the buttons.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

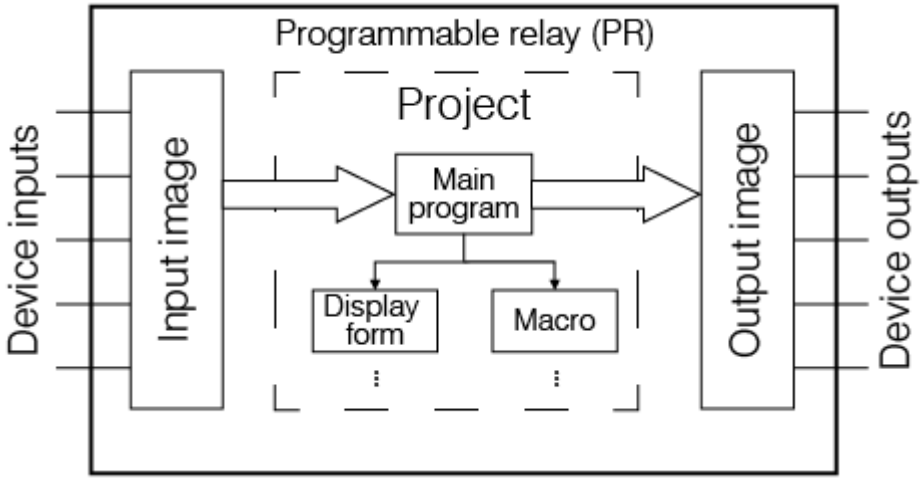
4 General information

This section describes the basic principles of operating a programmable relay (further referred to as "device") and programming this device:

- *Program execution*
- *Project creation*
- *Program creation*
- *Display programming*
- *Simulation*
- *Connection to device*
- *Upload project to device*
- *Online debugging*
- *Project information*
- *Component manager*
- *Macro development*
- *ST function*
- *ST function block*

4.1 Program execution

The selected target device determines the number of available inputs and outputs and the availability of a real-time clock.
General structure of a programmable relay:



Programmable relay is a programmable logical controller (PLC) with a cyclically executed project:
Step 1 – Device saves the status of physical inputs to the input memory cells (Input Image Table).
Step 2 – Device reads out the input memory cells and the main program is executed from its first command to the last one.
Step 3 – Device saves the results to the output memory cells (Output Image Table) and applies them to the outputs. When the last step is completed, the project runs again from Step 1.
When the last step is completed, the program runs again from the first step.

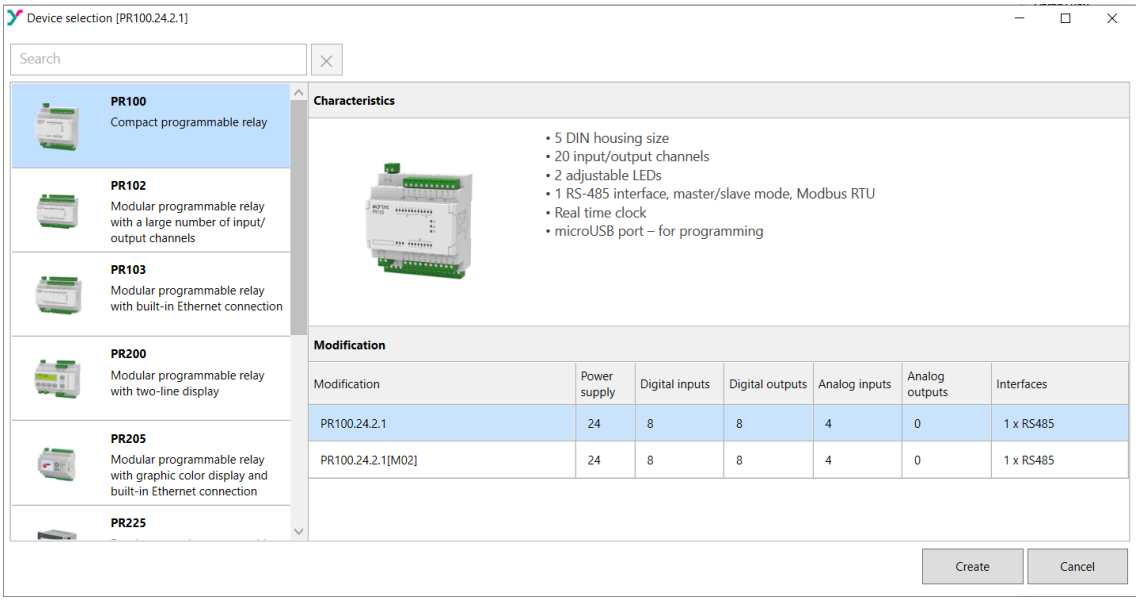
4.2 Project creation

Project creation

To create a new project:

1. Click icon in the taskbar or select ***File*** →***New project…*** in the main menu.
2. Select the target device in the dialog window ***Device selection*** and confirm it with ***OK***.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The new project appearance:

– Workspace – empty circuit program
– *Status bar* – information about available resources
– *Library Box* – available program blocks
– *Property Box* – workspace properties

NOTE
If the device is already connected to PC, ALP will offer the device model in the selection window.

If the selected device has a display, the *Device Manager* tab appears to the left from the workspace. With this tool you can *program* the displayed information.
You can save the current project or open a saved project using the corresponding buttons on the toolbar or in the main ***File*** menu.

Circuit program development

Now you can create the *main circuit program* in the workspace using the common program blocks from the toolbar ***Insert*** and the specific program blocks from *Library Box*. Draw connecting lines between inputs, outputs and blocks to establish logical connections in the program.

Simulation

Program can be simulated offline. Start the *simulation mode* using the menu item ***Service***

→***Simulation*** or the toolbar icon , change the state of the inputs and notice the state of the outputs to check how the program works.

Online debugging

If the device is connected and the program in the device and in ALP is the same, you can use *online debugging* to check the correctness of the program in the device.

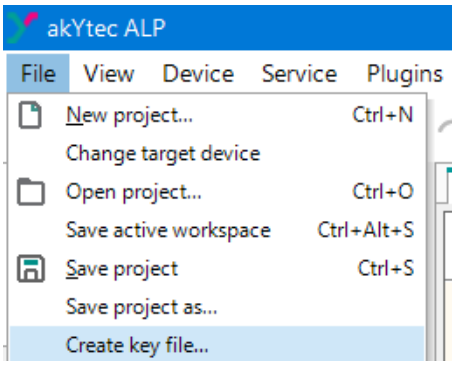
4.2.1 Protecting a project with a key

Creating a file with a key

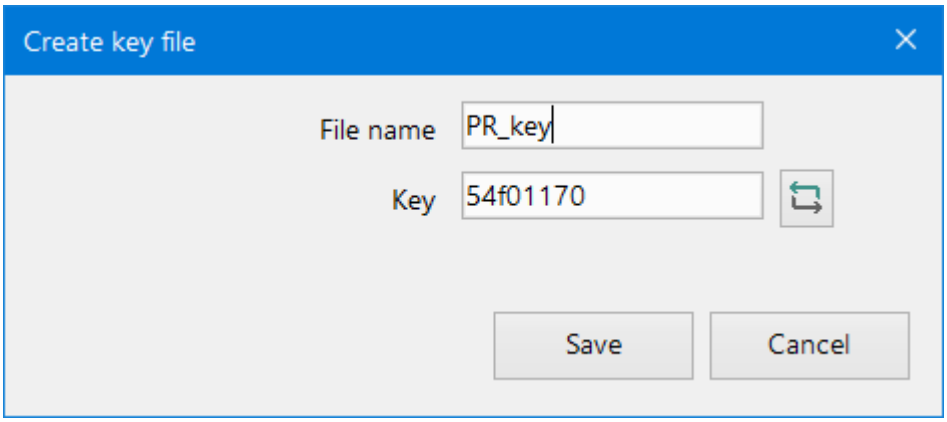
To create a file with a key:

1. Select from the main menu **File → Create key file…**

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. In the dialog box that opens, enter file name and generate the key. You can edit the key by clicking the button or manually.

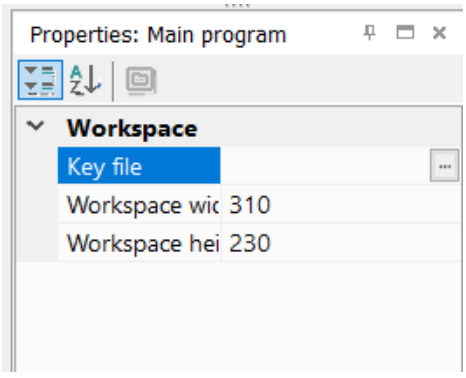


3. Save file. By default ALP saves the key file to folder *C:\Users\User name\documents\ALP\Keys*.

Binding a key file

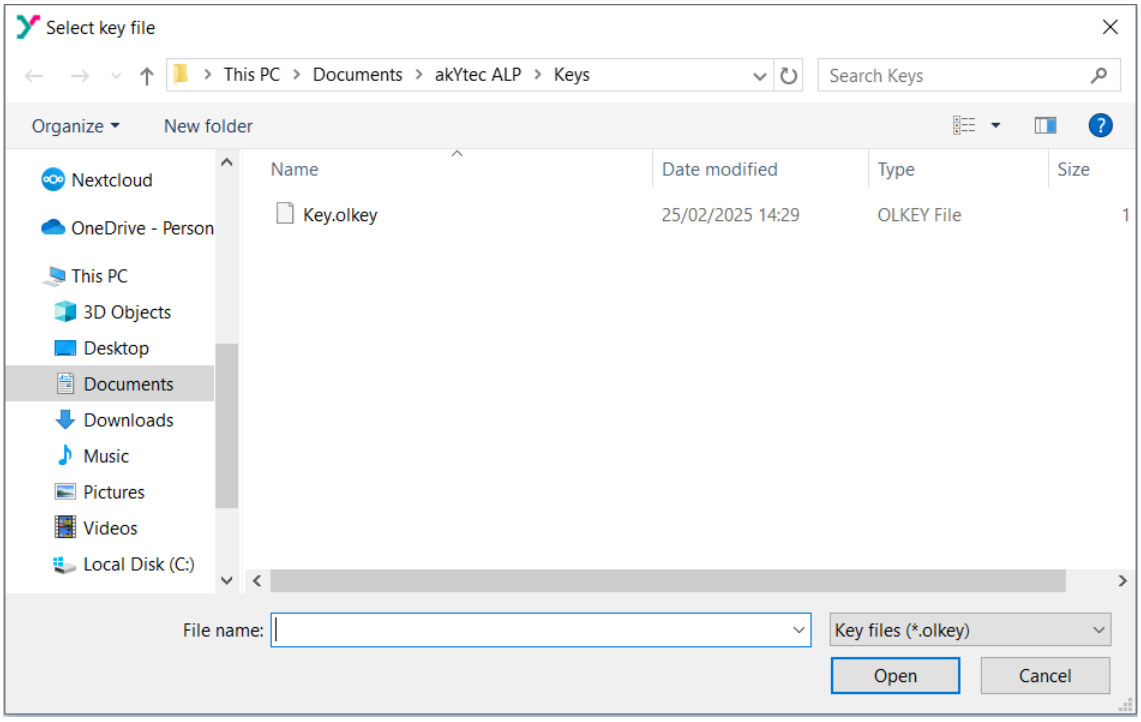
To bind a key file to a project:

1. Click on icon "**…**" in the file properties (**Properties: Main program** panel).

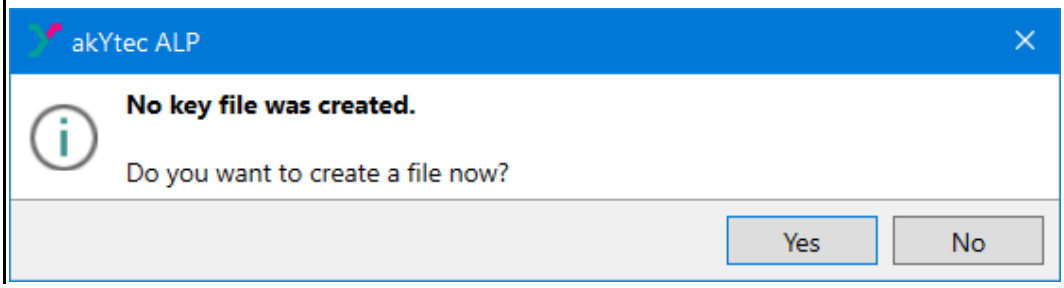


2. Select the key file in the new window.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



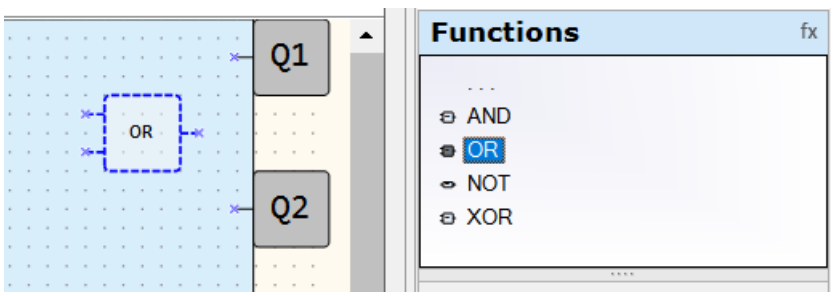
NOTE
If there is no key in the folder *C:\Users\User name\documents\ALP\Keys*, ALP will offer to create a new key.



3. Press the **Open** button.

4.3 Program creation

Before creating a program it is recommended to plan it. The plan should describe all possible states of the device during operation in form of a mode diagram, a table of I/O states, an electrical or functional diagram, etc.
When all the operation tasks are defined, you can create the program using the standard block from the toolbar Insert and the specific blocks from the project library. The project library presented in *Library Box* contains the functions and the function blocks available for the target device, as well as the macros added to the project.
To add a library block into the circuit program, select the block in Library Box and move it to the workspace by drag-and-drop.

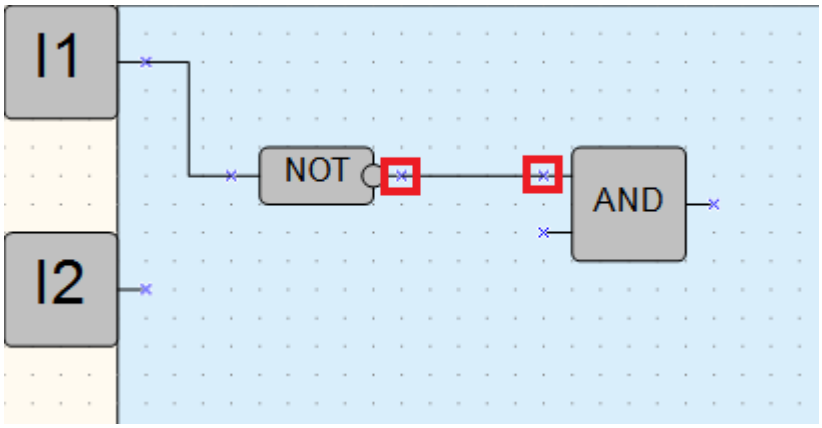


Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

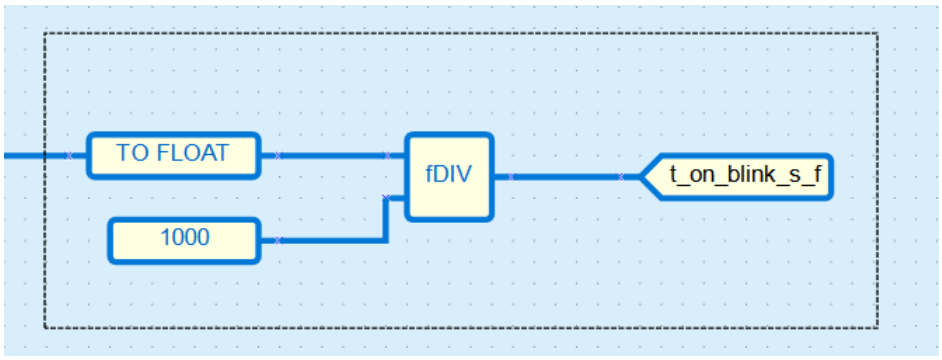
To draw connecting lines between inputs and outputs of the device and program blocks, use the left mouse button:

– Click the input pin of the device. The line is attached to it and follows the mouse cursor.
– To change the line direction, click a point in the workspace.
– Pull the line up to the input pin of a block and click on it to finish the line.

The connecting line can be drawn only between connection pins assigned to the same data type.



If the input and output data types of components do not match, the connection line cannot be created. For more information, refer to the *Variable types* section.
Click the block to select it. To select a group, pull the rectangle around several blocks.

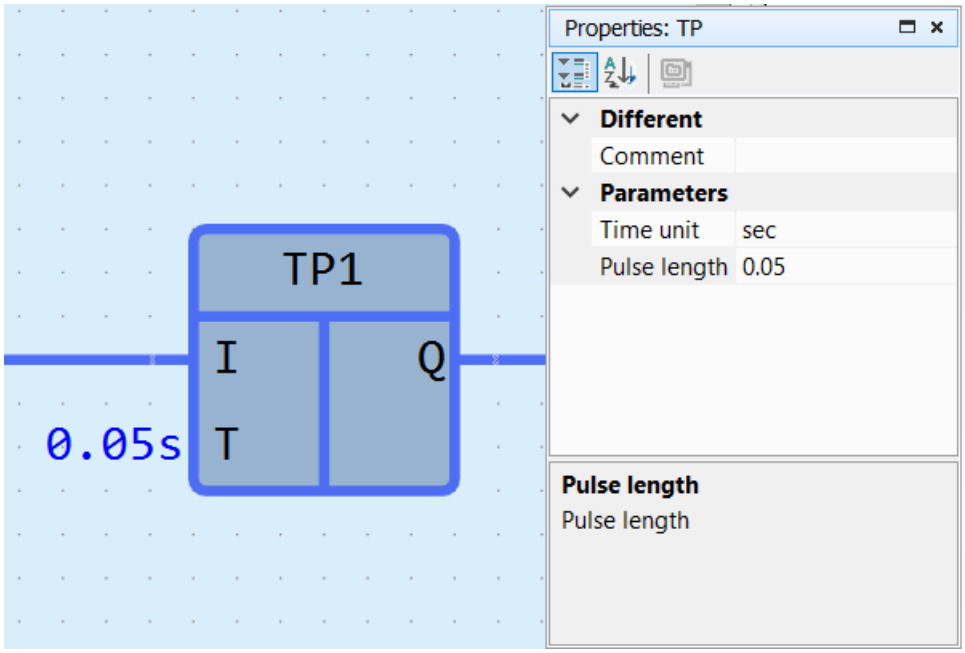


***Unused** variables* do not participate in the program logic and do not occupy device memory.

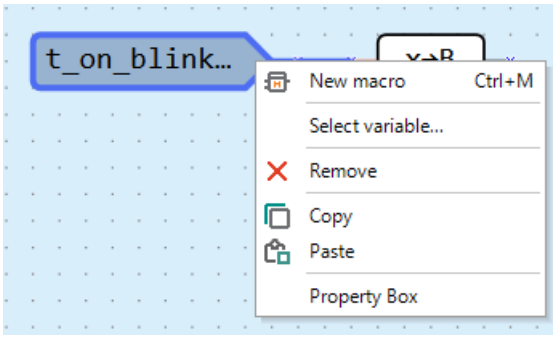
Component settings

You can edit properties of each element in *Property Box*.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Use block context menu for all manipulations available for the element.



Toolbar **Insert** has following blocks and functions for program development:

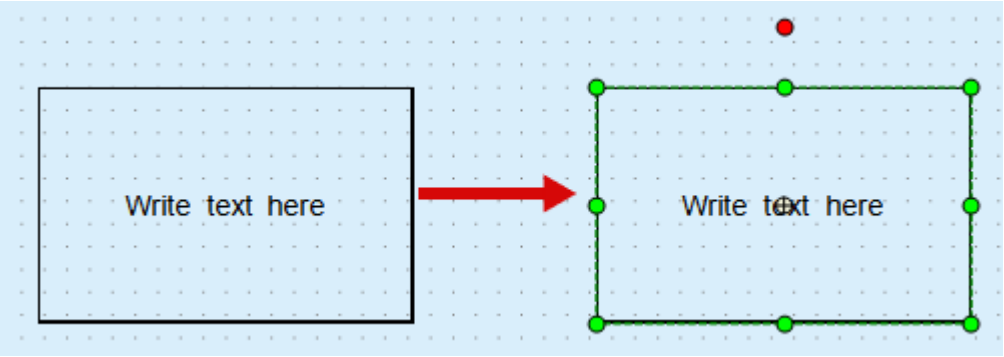
Text field	Placing a text comment on the circuit program
Variable block	Placing a variable block for writing or reading program values
Constant block	Placing a block with a fixed numeric value
Delay lines	Creating a delay for one cycle of transferring a value from the component's output to its input
Network variable block	Placing blocks for data exchange between devices connected to a common network
Read / write to FB	Writing/reading the values of individual parameters from the FB to a variable and vice versa
Conversion blocks	Converting values of different types for transmission
Arrange elements	Reassignment of sequence numbers of FB schemes
Execution sequence	Changing the order in which program output values are calculated

4.3.1 Text field

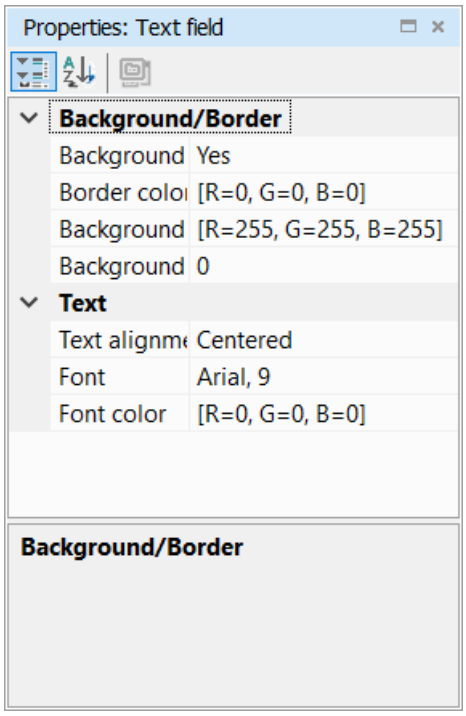
The text fields are used to explain the program.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

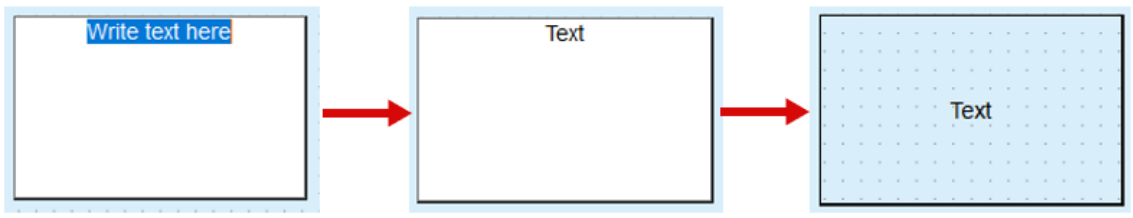
To add a text field to the program, click the icon in the toolbar ***Insert***, then click a point in the workspace to place the upper/left corner of the text field and draw a rectangle to set its size.



The parameters of the text field can be changed in ***Property Box***.



To make the background color of the text block visible, it is recommended to set the ***Background transparency*** greater than 20 %.
Double-click the text field to write the text.



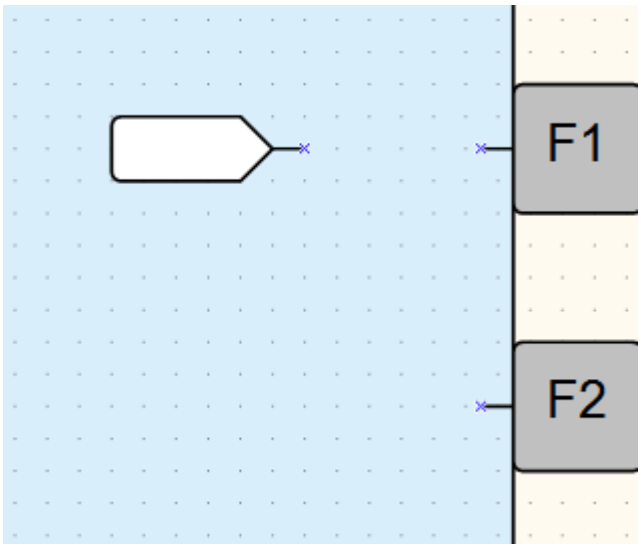
4.3.2 Standard variable block

The variable block enables the use of a variable in a circuit program.
To add a variable block to the program, click on its icon in the insert panel:

– **input variable block** — this block writes a variable into the program;

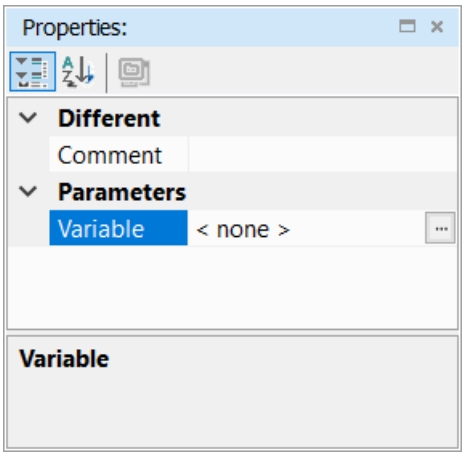
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

–  **output variable block** — block reads a variable from the program.
Click the point in the workspace to place the variable block.



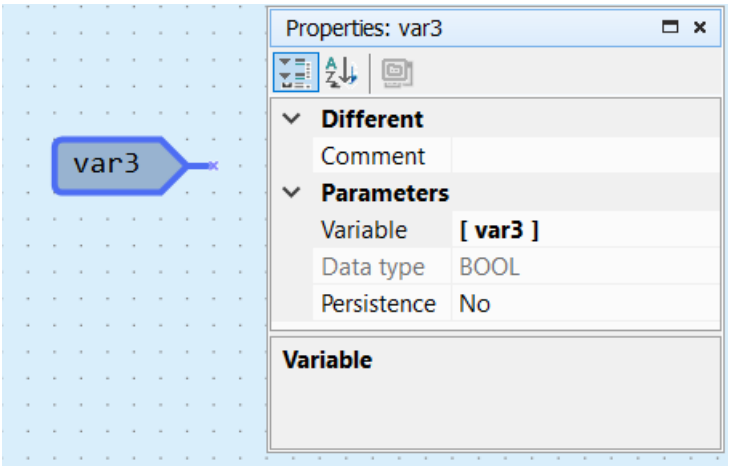
You can also add a variable block to the program from *Variable table*.
To assign a variable to a variable block:

1. Select a variable block.
2. Double click the block or click on icon "..." in the Variable row in the Property Box. *Variable table* opens. Select a variable or create a new one. Only the editable tabs in the table are available for selection. The availability is limited by the block type.

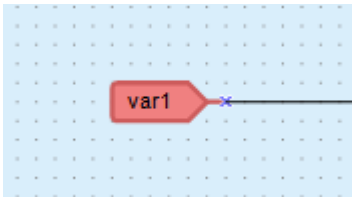


3. Confirm the selection with ***OK***. The variable is assigned to the block.

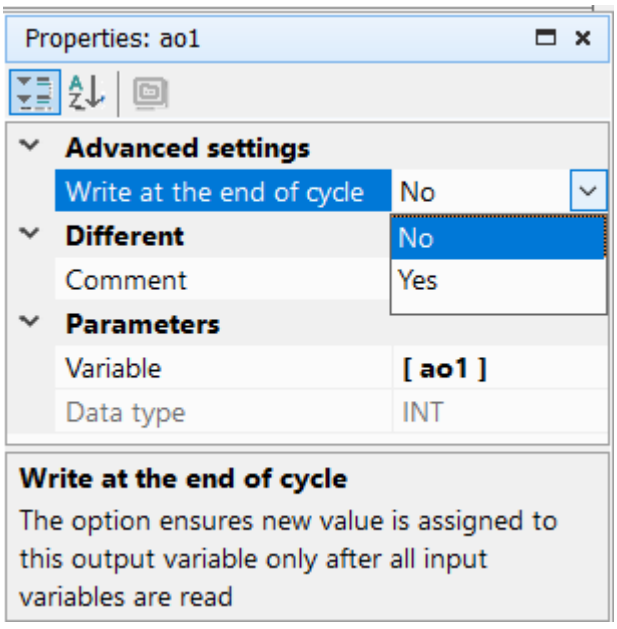
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Connect the variable block to the necessary element in the workspace.
If the variable block is highlighted in red, it means that the creation is incorrect or not completed. Information about the error is shown in the status bar.



Creating variables in the variable table is the recommended first step in programming.
If the variable is used more than once in a project, all references can be followed with the item **Show references** in the variable block context menu. The function is also available in simulation and online debugging modes.
For an output variable block that is bound to a device parameter or a visualization element, the **Write at the end of cycle** setting is available. This setting writes the value to the variable after all read operations in the *working cycle 4.1*.



The default value for the **Write at the end of cycle** setting is determined based on the **Persistence** property of the variable bound to the output block:

– if the **Persistence** checkbox is unchecked, then **Write at the end of cycle** is set to **"No"**;
– if the **Persistence** checkbox is checked, then **Write at the end of cycle** is set to **"Yes"**.

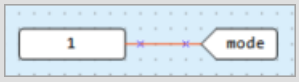
The default value for the **Write at the end of cycle** setting is updated in the following cases:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

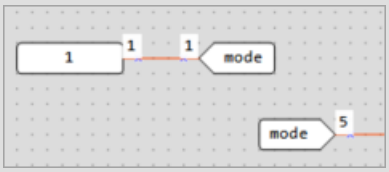
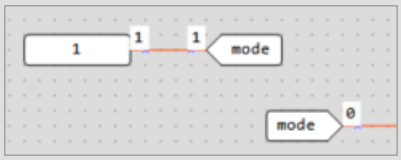
– each time the variable binding to the output block is changed;
– each time the **Persistence** property of the variable bound to the output block is changed.

Examples of using the **Write at the end of cycle** parameter setting with recommended values:

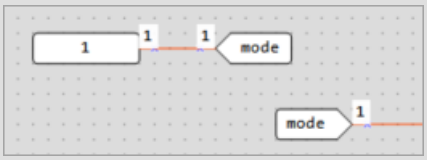
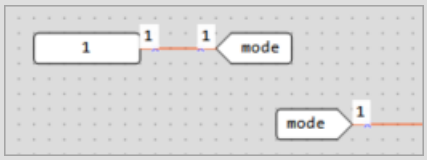
Example:
A variable named *mode* has been created and bound to a visualization element. The variable is initialized with a value from the canvas.



An operation needs to be performed on this variable in the first *working cycle 4.1*.
When **Write at the end of cycle** is set to **"Yes"**:

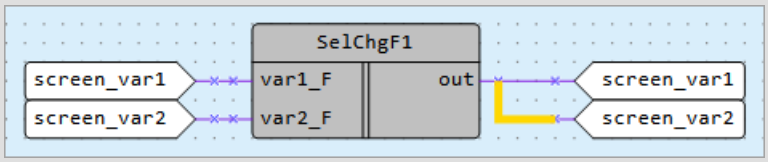
Non-Persistence	Persistence
The *mode* variable in the input variable block will be initialized with the default value (e.g., 5): 	The *mode* variable in the input variable block will be initialized with a zero value: 

When **Write at the end of cycle** is set to **"No"**:

Non-Persistence	Persistence
The *mode* variable in the input variable block will be initialized with the value from the canvas: 	The *mode* variable in the input variable block will be initialized with the value from the canvas: 

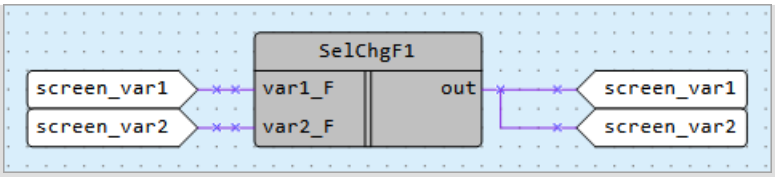
It is recommended to use **Write at the end of cycle "No"**.

Example:
Two standard variables, *screen_var1* and *screen_var2*, have been created and bound to different visualization elements that can be modified from the device screen.
Synchronization of these variables' values is required. This can be achieved using the SelChg/ SelChgF macros from the component manager.
The SelChgF macro is designed to modify an int/float type variable using two variables. The output of the macro records the most recent change from either of the two variables.
When **Write at the end of cycle** is set to **"No"** (default for volatile variables):



A cyclic dependency appears on one of the variables (in this example, *screen_var2*, highlighted in yellow), which prevents modifying the variable's value from the screen, as *screen_var2* will take the value of *screen_var1*.
When **Write at the end of cycle** is set to **"Yes"** (default for non-volatile variables) for the *screen_var2* variable, the cyclic dependency disappears, and both variables become modifiable from the screen:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

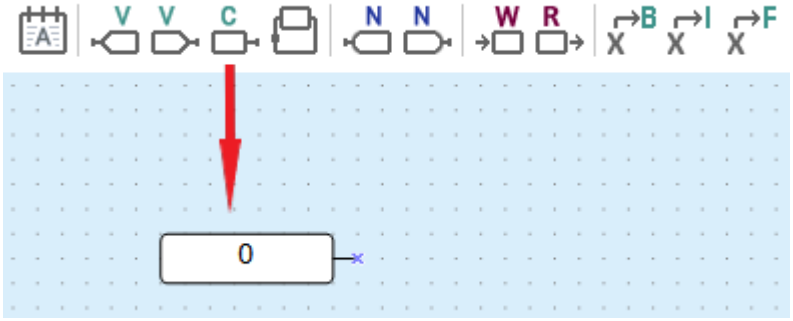


It is recommended to use **Write at the end of cycle "Yes"**.

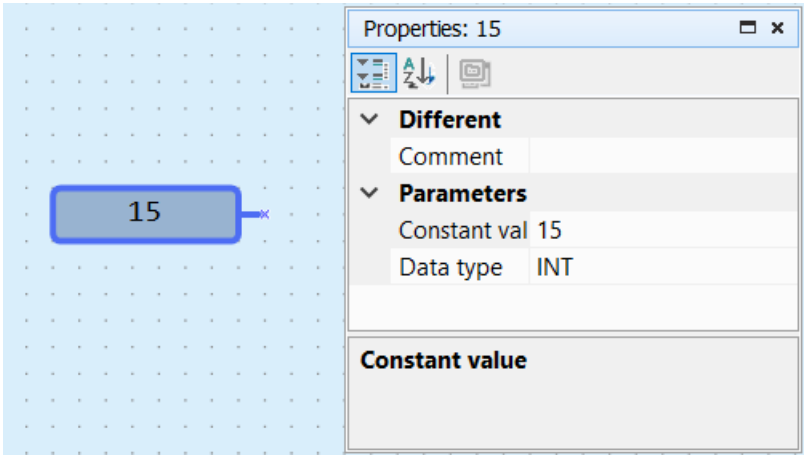
The information provided in the examples is advisory in nature. The user has the option to set the exact value that ensures the required behavior in this part of the diagram.

4.3.3 Constant block

To add a constant value to the program, click on icon C in the ***Insert***toolbar, then click the point in the workspace to place the constant block.

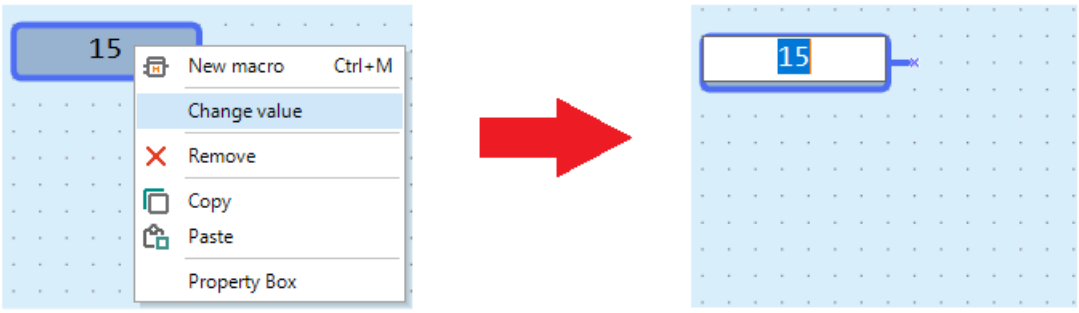


Select the data type using "**…**" in row ***Data type*** and enter the value on row ***Constant value*** in ***Property Box***.



The value of the constant block will not change throughout the program execution.
You can edit it by double-clicking on the constant block, in ***Property Box*** or by selecting ***Change value*** in the block context menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



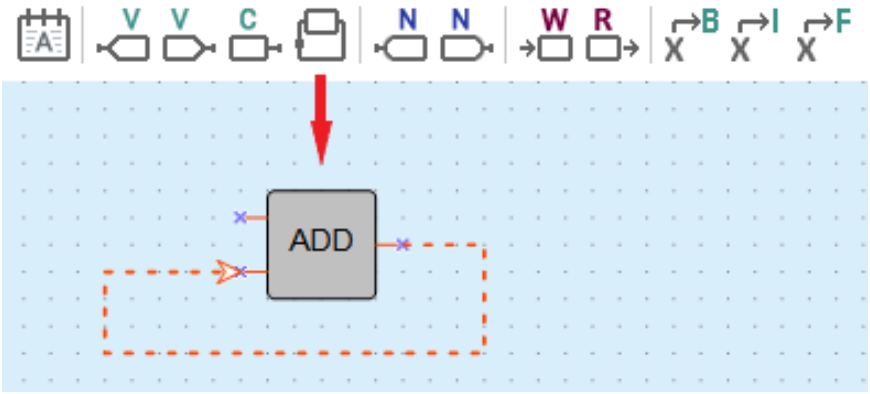
Data types valid values:

BOOL	0 / 1
INT	0 … 4,294,967,295
REAL	–3.402823e+38 … 3.402823e+38

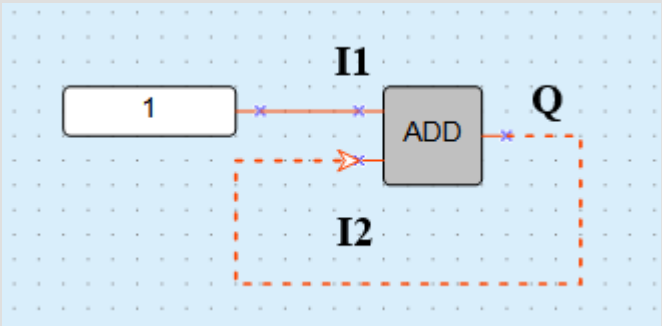
4.3.4 Delay line

The delay line is used to transfer the value from the block output to the block input, delayed for one cycle. The output and input may belong to different blocks.

Click the icon in the toolbar ***Insert*** and draw a line from the output to the input of a function block. The delay line is displayed on the diagram as a dashed line with an arrow, with the color corresponding to the variable's data type.



Example:
A constant value 1 is transferred to the input I1 of the addition block ADD (Integer). A value from the block output (Q) calculated in the previous cycle is transferred to the input I1 over delay line.



Cycle signal values:

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Cycle	1	2	3	4	5	6	7	8	9	10
I2	0	1	2	3	4	5	6	7	8	9
Q	1	2	3	4	5	6	7	8	9	10

4.3.5 Network variable

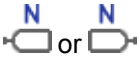
The network input and output variable blocks are special type of variable blocks for data exchange between devices connected to a common network.

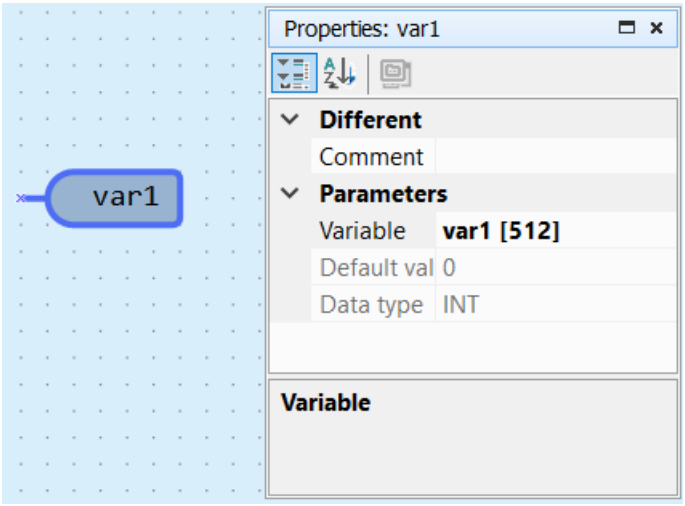
–  – Network output variables are the variables that can be read via the network.

–  – Network input variables are the variables that can be written via the network.

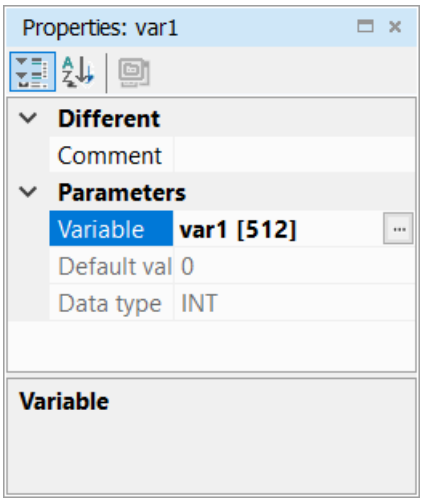
NOTE
A variable cannot be assigned to the block if there are no communication interfaces in the device configuration.

To add a network variable to the program:

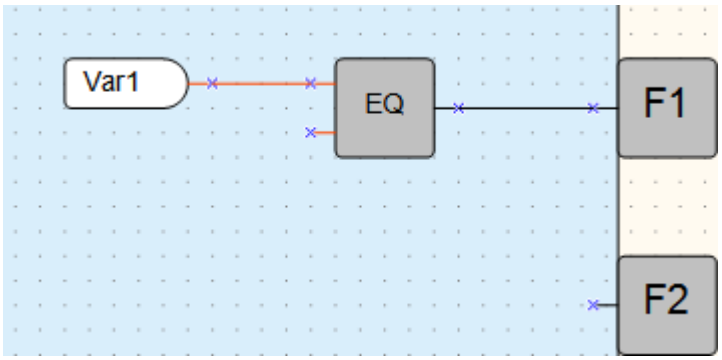
1. Click on icon  in toolbar ***Insert***.
2. Click the point in the workspace to place the variable block.
3. Double click the block or click on icon "..." in the Variable row in the Property Box. *Variable table* opens. Select a variable or create a new one. Only the editable tabs in the table are available for selection. The availability is limited by the block type.
4. Confirm the selection with ***OK***. The variable is assigned to the block.



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Connect the network variable block to the necessary element in the workspace.



If the variable block is highlighted in red, it means that the creation is incorrect or not completed.
The information about the error is shown in the status bar.
Creating variables in the variable table is the recommended first step in programming.
If the variable is used more than once in a project, all references can be followed with item ***Show references*** in the variable block context menu.

4.3.6 Read from/Write to FB

The ReadFromFB/WriteToFB blocks are used to read or write a parameter value of a function block during the program execution.
The following blocks can be added to the circuit:

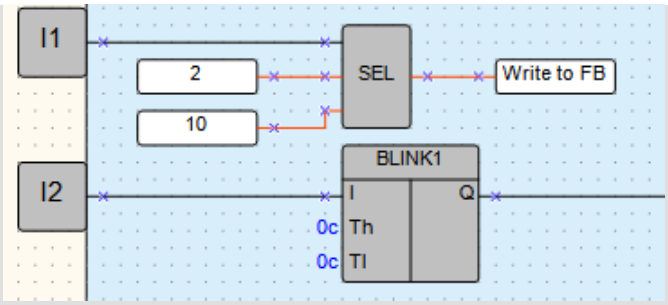
– W — to write the value to a FB;

– R — to read the value from a FB.

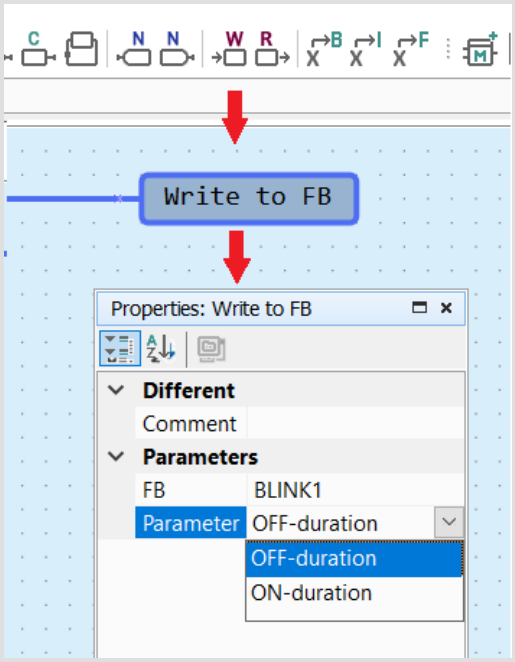
The block ***WriteToFB*** (W) is used to change an FB parameter during the process.

Example:
The value of the parameter ***ON-duration*** of the FB ***BLINK1*** should be 2 or 10 depending on the value at the input ***I1***.

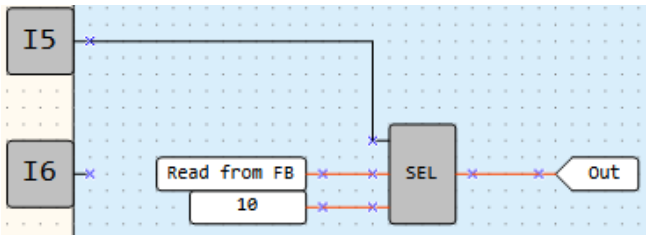
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Go to ***Property Box***, select the FB ***BLINK1*** in the row ***Function block*** and the parameter of the FB in the row ***Parameter in FB***.



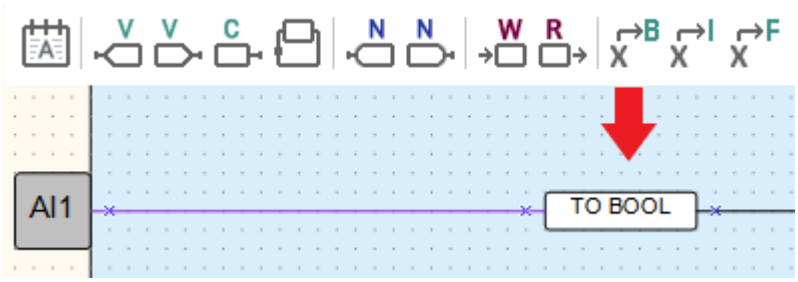
The block ***ReadFromFB*** (R) reads the current value of an FB parameter and uses it in the program. Editing is similar to the **WriteToFB** block.



4.3.7 Conversion block

A connection line between program components can only be created for an input and an output of the same type: Boolean, Integer, Floating Point, or Array (with the same number and type of elements). To create a connection between an input and an output of different types, use conversion blocks.
Using conversion blocks with variables of the Array type is not possible. Conversion of individual array elements is allowed.
To add the conversion block to the program, click the corresponding icon in the toolbar ***Insert***, then click the required place in the workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

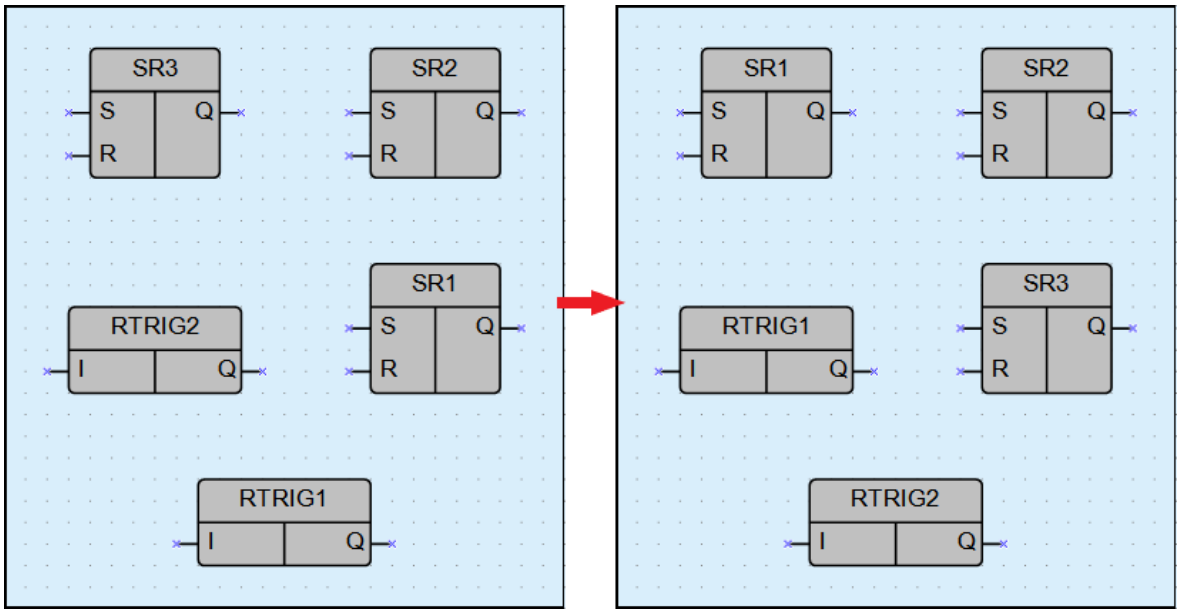


Conversion blocks:

	Conversion to BOOL	Converts INT or REAL to BOOL If the input value > 0, the output = 1 (True)
	Conversion to INT	Converts BOOL or REAL to INT REAL is rounded down to INT, negative value is converted to 0
	Conversion to REAL	Converts BOOL or INT to REAL

4.3.8 Arrange elements

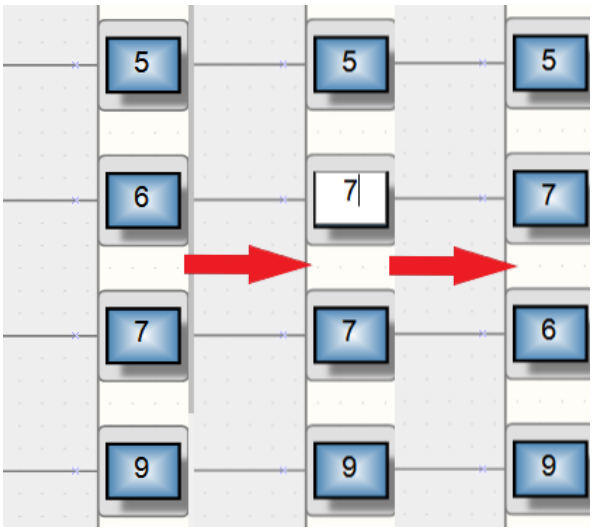
The sequence numbers of the function blocks can be automatically reassigned by clicking the button ***Arrange elements*** in the toolbar ***Service***. The blocks of the same type are numbered sequentially from top to bottom and from left to right.



4.3.9 Execution sequence

Calculation of the values for outputs and delay lines is performed in a certain order. To see this order, click the arrow near the icon Q in the toolbar ***Service*** and select ***Delay lines*** or ***Outputs***.
ALP will switch to the execution order setting mode – the sequence numbers of the execution order will be displayed near the outputs and feedbacks.
To change the order, double-click an output or a delay line and enter the required number.

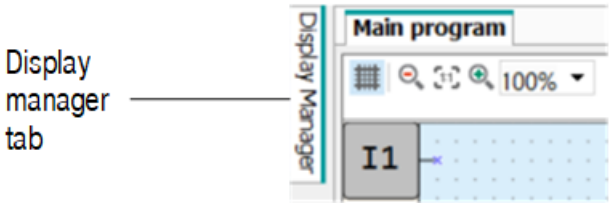
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



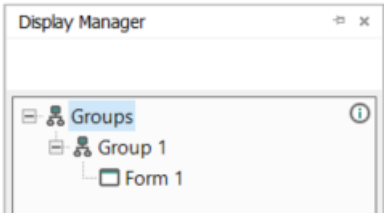
Click the icon  once more to deactivate the edit mode.

4.4 Display programming

If the device is equipped with a display, it can be used to show various information according to the algorithm defined in the program. The screen can display text, graphic components, and program variable values. The *Display Manager* is used for programming the display. To open the Display Manager, click its tab title on the left side of the workspace.

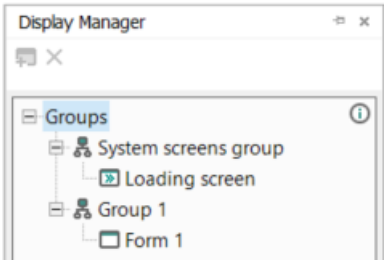


The display manager differs between devices with graphical color LCDs and those with monochrome text LCDs as follows:



By default, Form 1 is added to Group 1:
- Group 1 can be edited
- Group 1 cannot be deleted

Fig. 4.1 Display manager for devices with a monochrome text LCD



By default, a Loading screen is added to the System screens group:
- The System Screens group and the Loading screen cannot be edited or deleted.

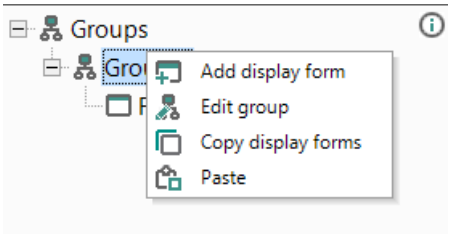
By default, Form 1 is added to Group 1:
- Group 1 can be edited
- Group 1 cannot be deleted

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

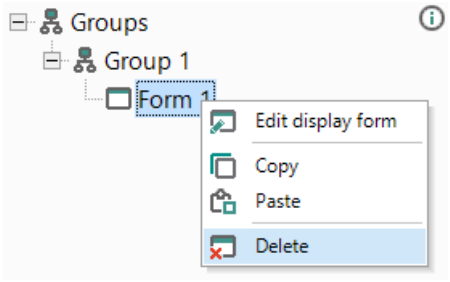
Fig. 4.2 Display manager for devices with a graphical color LCD

Adding Screens

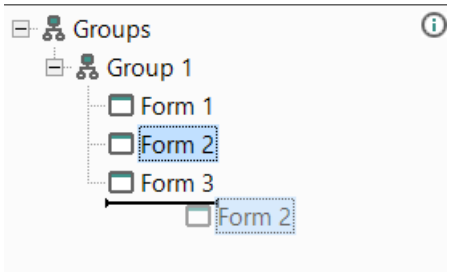
You can add multiple display forms in the display manager. It is possible to configure display form jumps on the device's display triggered by button presses or variable changes. The configuration of jumps differs between *monochrome text LCDs* and *graphical color LCDs*.
To add a screen, right-click on the **Group 1** element and select **Add display form** from the context menu.



To delete a screen, right-click on the desired screen and select **Delete** from the context menu.



To change a display form's position, use the drag-and-drop method while holding down the **Shift** key.



When moving a display form , the new position will be indicated by a horizontal marker.

Display form editor

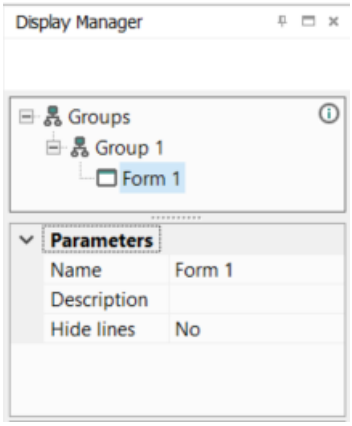
The functionality of the display form editor depends on the type of display in the device:
– *Monochrome text LCD*
– *Graphical color LCD*

4.4.1 Monochrome text LCD

Display parameters

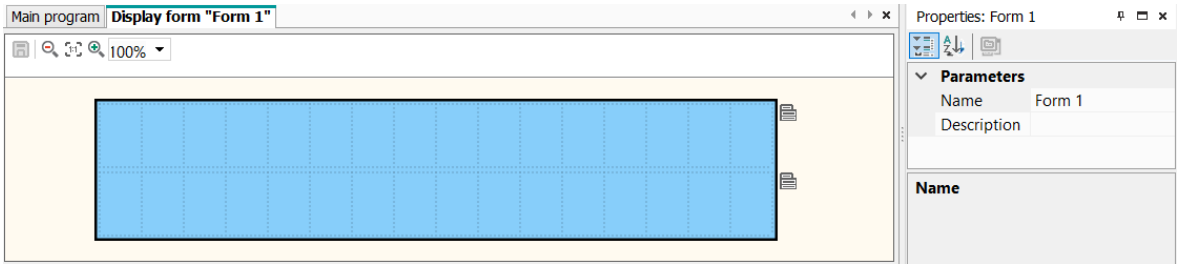
Display parameters are displayed in the properties box of the **Display manager**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



– **Name** – displayed in the **Display manager** and in the display editor's title bar.
– **Description** – a text description of the display form.
– **Hide lines** – hides or shows empty rows on the device's screen.

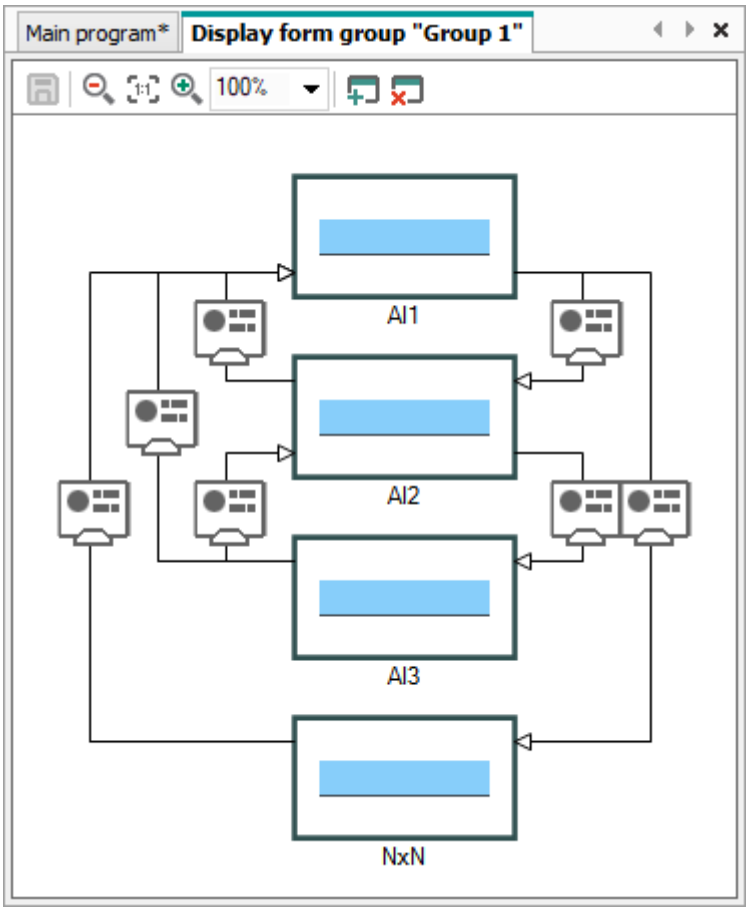
To display the **parameters** of a display form in the Property Box panel, double-click the desired display form in the **Display manager**.



Display form group editor

For devices with a monochrome text LCD, the Screen Group Editor is also available. It is opened by selecting the **Edit Group** command from the context menu of a screen group in the **Display manager**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



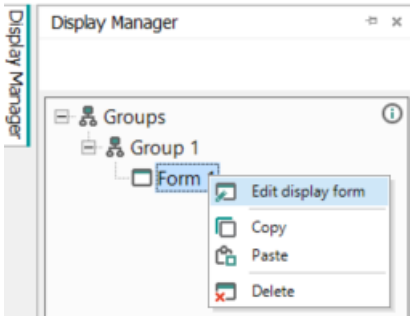
At the top of the Screen Editor and Screen Group Editor windows, the following buttons are available:

	**Save Document**
	**Zoom Out**
	**Actual Size**
	**Zoom In**

The zoom level can also be changed using the dropdown menu to the right of these buttons.

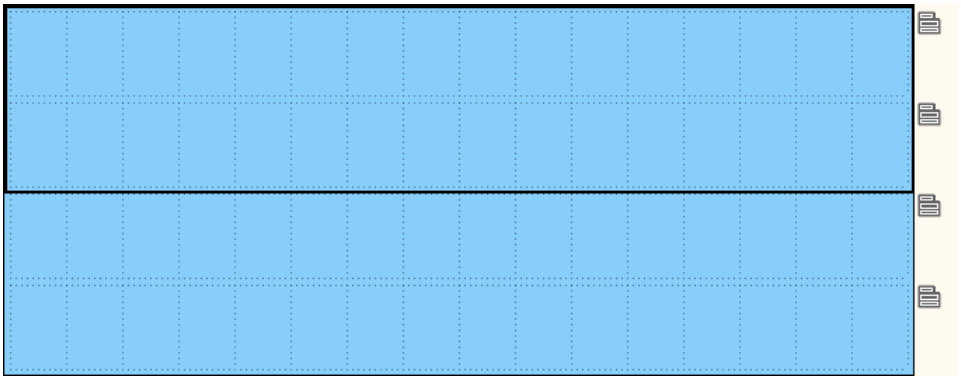
Display editor

To open the selected form in ***Display Editor***, use the command  ***Edit display form*** in the form context menu or double-click the form in the tree.



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

An icon on the right edge of each row represents the row context menu, which is used to change the number and the order of the displayed rows.

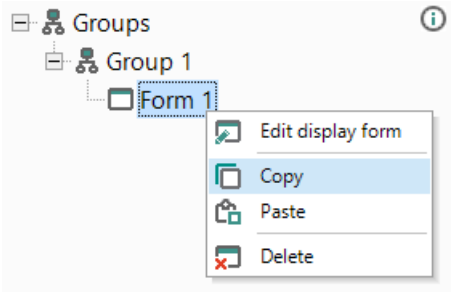


Put the *display elements 7.6* from ***Library Box*** by drag-and-drop onto the form.

NOTE
The character set is implemented within the Windows-1251 encoding.

Copy-paste display form

In the Display Manager, you can copy forms for pasting into the current or another project. To copy selected forms, select the **Copy** command in the context menu of a form or group of forms, or press the **Ctrl + C** key combination. Multiple screens can be selected by holding down the **Ctrl** or **Shift** key.



To paste copied forms, select the **Paste** command in the context menu of a form or group of forms, or press the key combination **Ctrl + V**.
All controls and screen properties placed on the form are copied along with the form. The variables associated with the form are also copied, according to the rules described in the section *Copy-paste variables*.
Jumps between the selected forms are copied as well. If the form has a related jump outside the selected area, this jump is deleted.

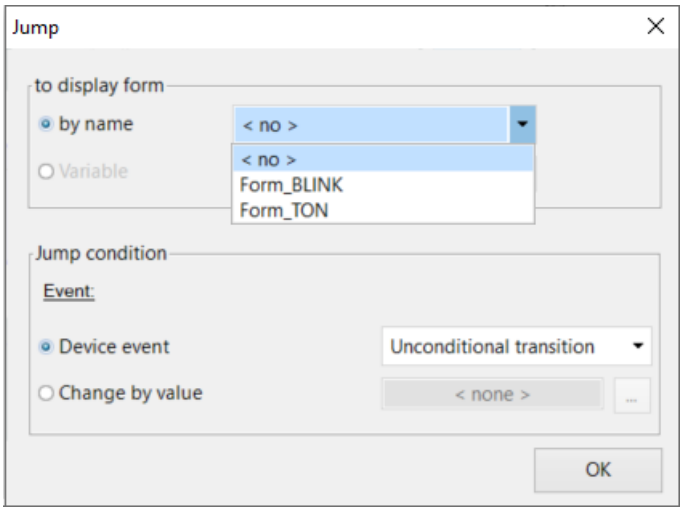
Jumps

If the display structure consists of more than one form, “jumps” should be defined to enable the navigation between forms.
To create a jump:

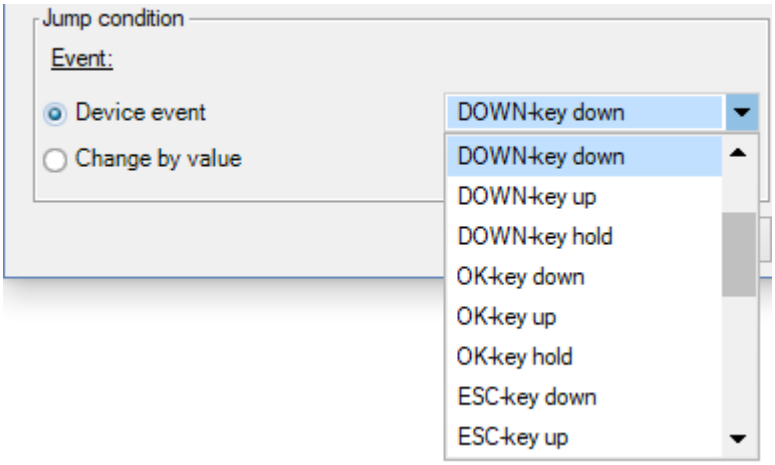
1. Right-click on the **Group 1** element in the display manager tree and select **Edit group** in the context menu. The screen group editor tab will open.
2. Select the start form in the form group editor.
3. Click on icon “**…**” in row ***to display form*** in ***Property box***. Jump dialog window will open.

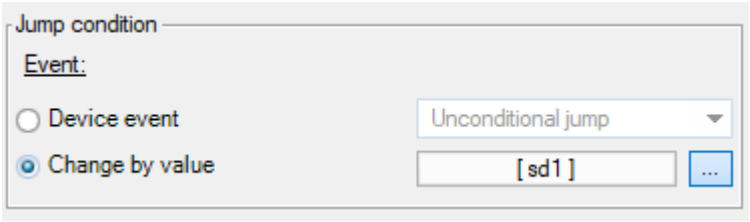
   In the drop-down menu, select another form to go to.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



4. Select the event in section ***Jump condition*** as device event or change of a variable by value.
   A button event can be selected as ***Device event***.
   A BOOL variable can be selected for ***Change by value*** event.

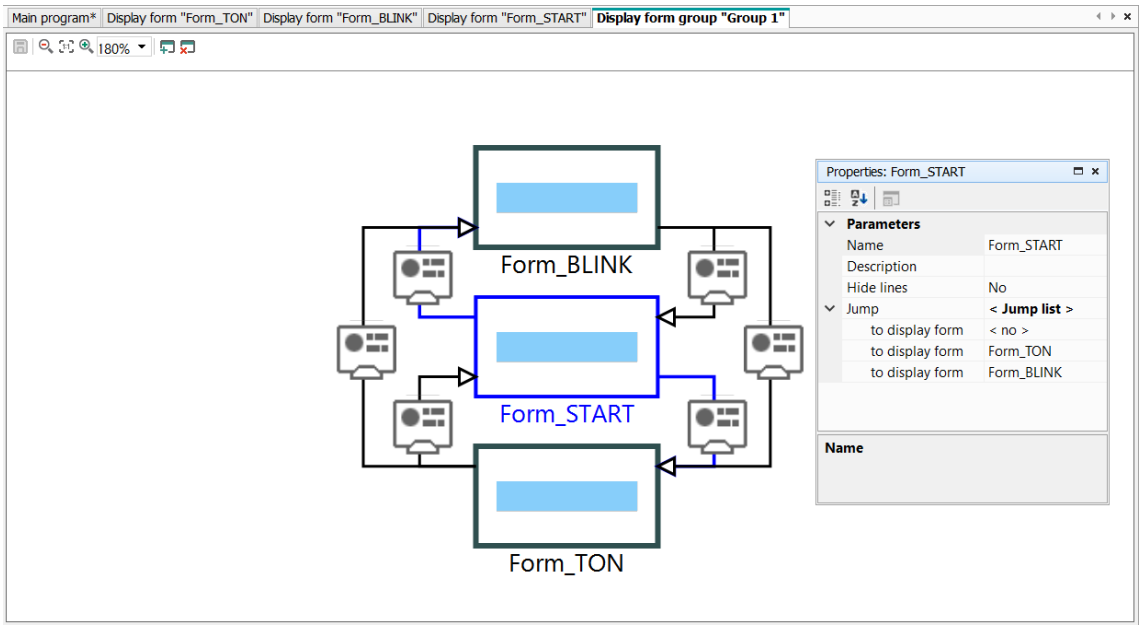




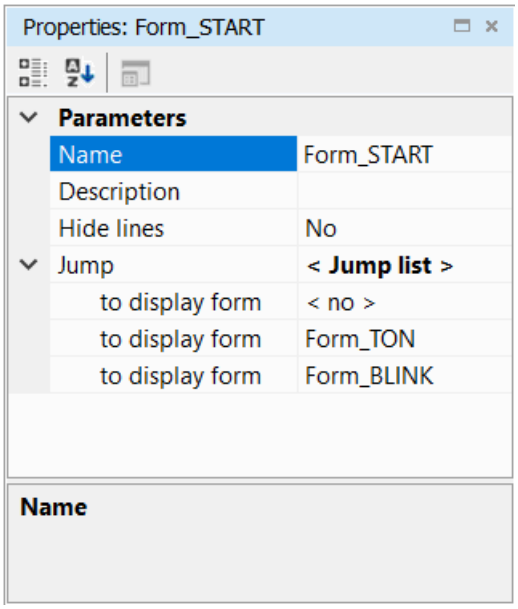
NOTE
An unconditional transition is used to configure a jump when none of the specified events or variable value changes occur, for example, to switch to an alarm screen.

5. Confirm with ***OK***. The created jump is shown in the structure.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



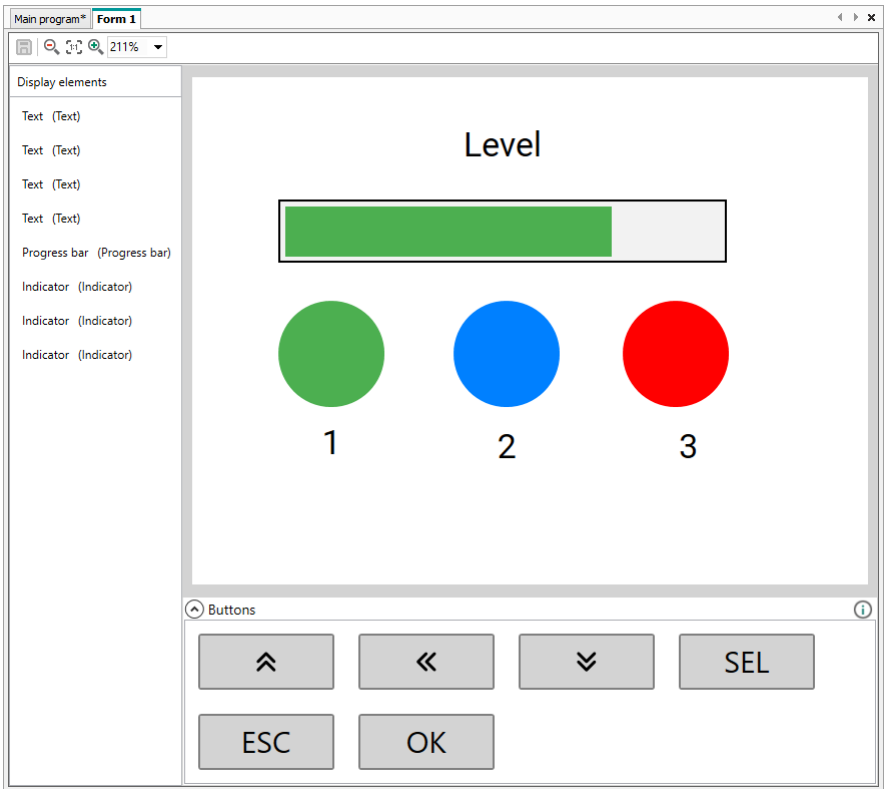
The jumps can occur based on various events. To accommodate this, multiple jumps **collection** can be configured. Jumps are added in the same way as described above.
To edit or delete a transition, select it in the screen's properties and configure the corresponding parameters. A transition will be deleted if the value in the **Jump list** field of the transition creation menu is set to **<no>**.



4.4.2 Graphic color LCD

For devices with graphic color LCD, the display output is programmed using graphic elements.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

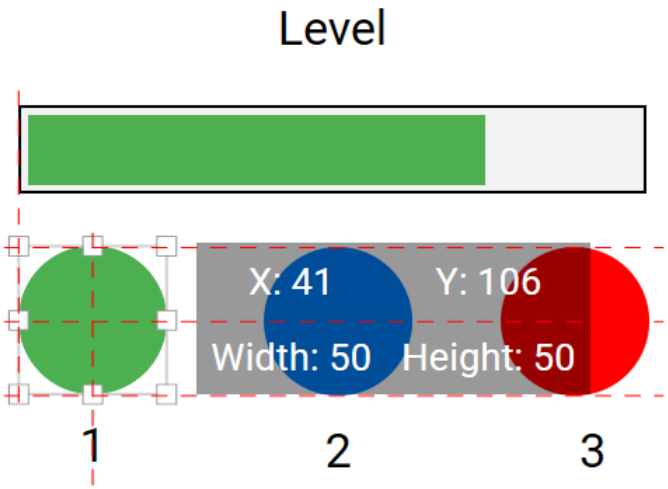


Screen Editor Areas:

– **Screen elements panel** is used to display the list of components located in the screen field. In the screen elements panel, you can also change the order of graphic elements and assign names to them;
– **Screen field** is used for arrangement of graphic elements;
– **Buttons** are used to program the device buttons.

Graphic elements to be placed into the screen field can be selected from the **Library Box**. Element descriptions are given in *Basic Elements* section.
A screen simulation window is available in *simulator mode*.

Screen field



When graphic elements are moved into the field, tooltips appear with coordinates and dimensions of the element and guide lines relative to other elements.
The following context menu actions are available for graphic elements and graphic element groups:

– Copy

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

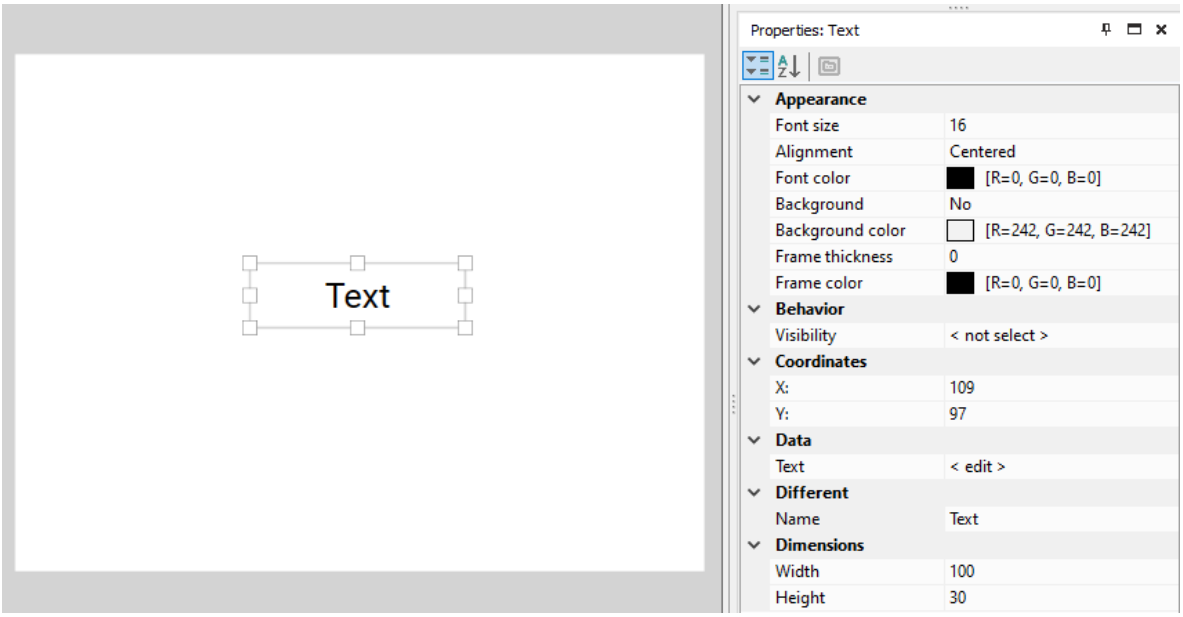
– Insert
– Delete
– Move to the foreground and background
– Align

Double-click on the graphical elements **Text** and **Dynamic Text** to activate the text input mode or on other elements to open the variable binding window.
Graphical elements can be moved using the arrows on the keyboard, and other actions can be performed using *keyboard shortcuts*.

Properties of graphic elements

The properties of a graphical element are displayed when:
– selecting the graphical element located on the screen canvas.
– selecting the graphical element located in the Display elements panel.

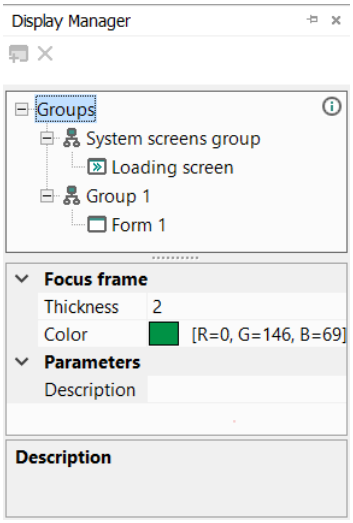
Properties common to all graphic elements:
– **Element name**;
– **Size** - width and height in pixels;
– **Coordinates** - on X and Y axes, the reference point is the upper left edge of the element.



Screen Groups

For the **Groups** folder, the properties panel of the **Display manager** allows you to configure parameters that will be applied to all visualization elements on every screen within that group. Changing settings at the group level automatically affects all graphical elements located on all screens in the group.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



– **Parameters:**

  Description – a text description of the screen group.

– **Focus frame:**

  Thickness – the thickness of the frame around a graphical element in focus, both in simulation mode and on the device screen.

  Color – the color of the frame around a graphical element in focus, both in simulation mode and on the device screen.

Loading screen

For devices with a graphical color LCD, you can upload a custom image for the screen that is displayed during device initialization after power-up.

NOTE
By default, a pre-installed image is shown on the loading screen.

To add a custom image, double-click the left mouse button on the **Loading screen** element in the **Display manager** tab. The **Loading screen image** window will open:

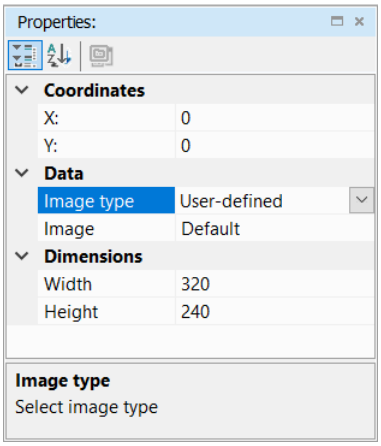


Fig. 4.3 Loading screen image properties

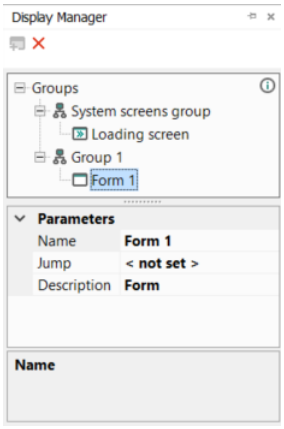
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional Options
Coordinates	X	The X-coordinate position of the element. The element's position is determined by the coordinates of its top-left corner. Coordinates start from 0 and increase from left to right. The maximum value is determined by the element's width and the screen's width.	Coordinates can be set by dragging the object on the screen canvas.
	Y	The Y-coordinate position of the element. The element's position is determined by the coordinates of its top-left corner. Coordinates start from 0 and increase from top to bottom. The maximum value is determined by the element's height and the screen's height.	
Data	Image Type	– Selecting "Default" disables further parameter configuration – the startup screen will display the pre-installed image. – Selecting "User-defined" enables configuration of the image parameters.	
	Image	Select the image file. The following file types are available for selection: *.jpg; *.jpeg; *.jpe, *.bmp. The maximum image resolution is limited by the device's screen resolution. Image scaling down is allowed within the ALP screen editor	
Dimensions	Width	The width of the element along the X-axis. The maximum value cannot exceed the width of the loaded image.	Element size can be set by resizing the object's boundaries on the screen canvas. The width and height of the element change proportionally to the dimensions of the loaded image.
	Height	The height of the element along the Y-axis. The maximum value cannot exceed the height of the loaded image.	

When the device boots up, the first screen from the Screen Group 1 list is displayed by default.

Display form parameters

Display form parameters are displayed in the properties panel of the **Display manager**.



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

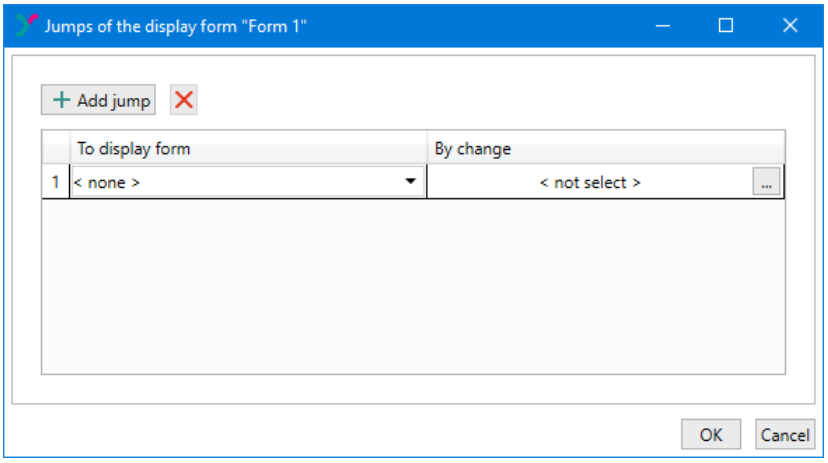
– **Name** – displayed in the **Display manager** and in the display editor's title bar.
– **Jump** – for creating transitions between display forms.
– **Description** – a text description of the display form.

Jumps between display forms

NOTE
Configuring transitions to the **Loading screen** is not available.

To create a jump for a graphical display:

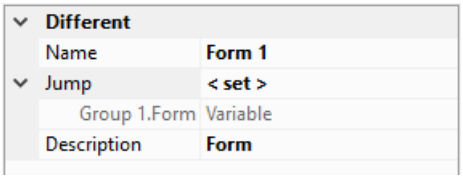
1. Right-click on the screen in the screen manager tree and select **Set up jumps** in the context menu or click on the **…** button in the **Jump** field of the display form properties. The Jump Editor window opens. The selected screen will be the starting point for the transition.



2. From the **To display form** drop-down menu, select another display form to transition to.
3. In the **By change** drop-down menu, specify a Boolean variable, which, when its value is changed, triggers the jump. After the transition is executed, the value of the bound variable is changed to "0".

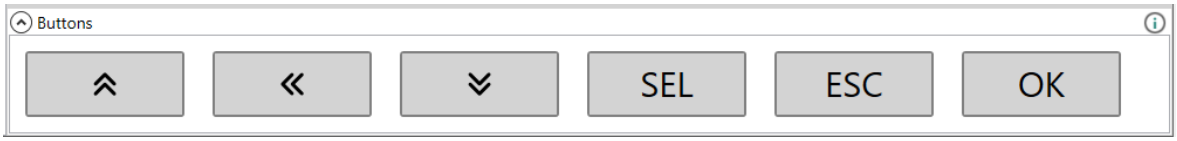
   NOTE
   To create a button-triggered jump, see below.

4. To add another jump, click **Add jump**, to delete a selected jump, click .
5. Click **OK**. The created jumps will be displayed in the display form properties. The order in which the jumps are executed corresponds to their order in the display form properties.



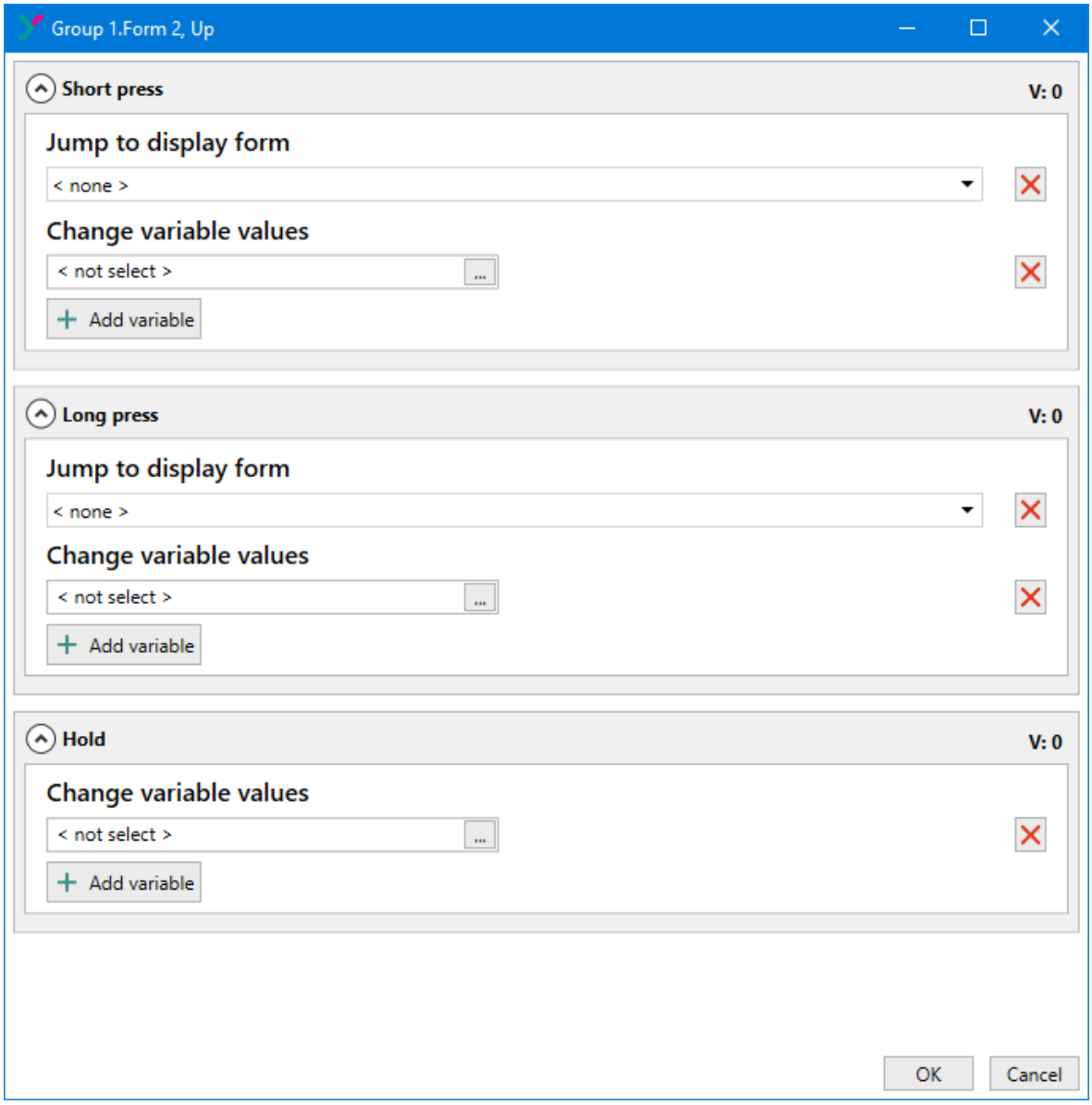
Configuring Screen Jumps Using a Button

At the bottom of the visualization editor, there is a "Buttons" panel that displays all the physical buttons available on the device:



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Double-click a button to open the event programming window. The buttons are programmed separately for each display form.



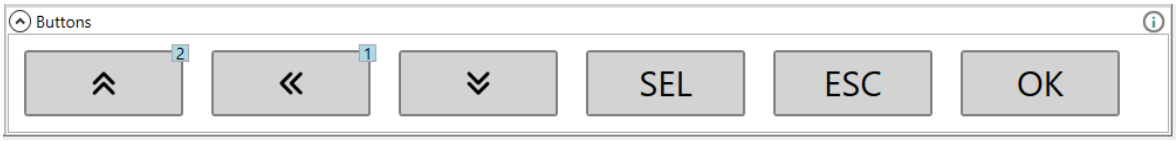
The following events are available for programming the button:

– short press (less than 3 seconds)
– long press (3 seconds or more)
– hold (5 seconds or more)

You can program several handlers for short and long presses (jumps to other display forms and changes in the value of Boolean variables). Only a handler for changing the values of Boolean variables can be programmed to hold. Handlers are executed in the program in the order in which they are added.

To add a variable, click + **Add Variable**. To delete a programmed action, press ✕.
Programming overrides the default button behaviors. To return to the default behavior of the buttons, delete the event handlers and write the program to the device.
If no actions have been set by the user for the **Up** and **Down** buttons, then when they are pressed the device will move to other screens according to the order in the screen manager window.
Once programmed, the button menu will indicate the number of events for each button.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



4.5 Simulation

The correctness of the created program can be verified using the simulation mode, which models how output states change in response to changes in input states. Simulation allows you to analyze the state of signals within the program.

To enter simulator mode, click the ▶ button on the toolbar or select **Tools** → **Simulator Mode** from the main menu. The simulation panel will open.

Simulation toolbar

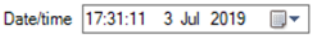


▶	**Start**	Start the permanent simulation
↪	**Single cycle**	Step-by-step simulation. Click the icon to execute one program cycle
⏸	**Pause**	Interrupt the simulation. Clock the icon once more time to continue the simulation
■	**Stop**	Stop simulation
	Refresh time	Input field for setting the information refresh period on the scheme in milliseconds
	Cycle time	Input fields for setting the cycle time of program execution in simulation mode. Cycle time units: milliseconds, seconds, minutes, hours
	Watch Window	Open/close the window to watch the variables vaues at each program step
	Display form simulation	Open the display simulation window

CAUTION
Cycle time *in simulation mode is not the same as the cycle time calculated by the actual device. The* ***cycle time*** *can be set arbitrarily and is required to simulate the operation of timing function blocks – TON, TOF, BLINK. All timing parameters of these function blocks will be processed during the simulator's intermediate steps.*

Simulation of Clock Blocks

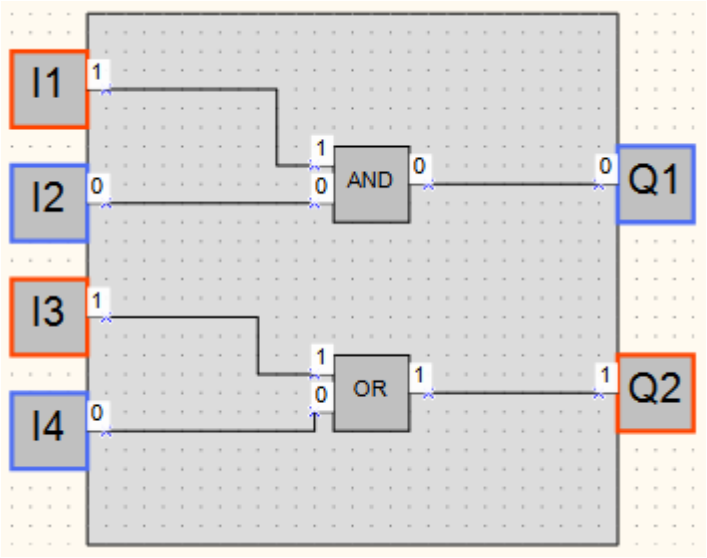
For devices with a real-time clock function, an additional panel opens. This panel allows you to simulate the behavior of the *CLOCK* and *CLOCK WEEK* function blocks over time.



Simulation procedure

1. Run simulation in one of the modes: real time (▶) or step-by-step (↪).
2. Set the input values on program blocks.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



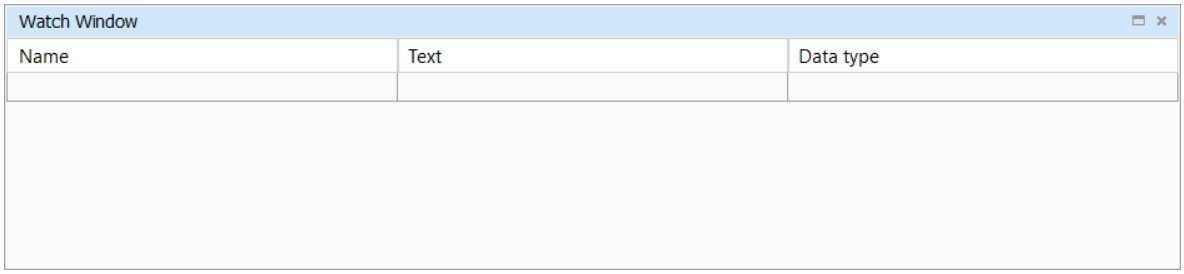
3. Select values of parameters ***Refresh time***, ***Cycle time*** and **Cycle unit time** for convenient simulation.
4. Exit simulation mode to correct the program.

The simulation mode operates according to the following rules:

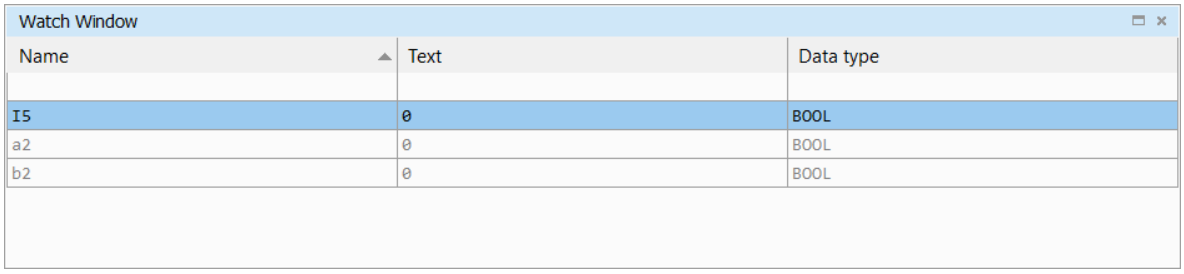
– The program (on the **Program** tab) and macros are simulated separately.
– The operation of blocks that are not connected to any of the device's outputs or to an output block of a network variable is not simulated.
– Simulation will not work for volatile variables or variables that are linked incorrectly.

Watch window

Click icon  on the simulation toolbar to watch the input, output or variable values at every program step.

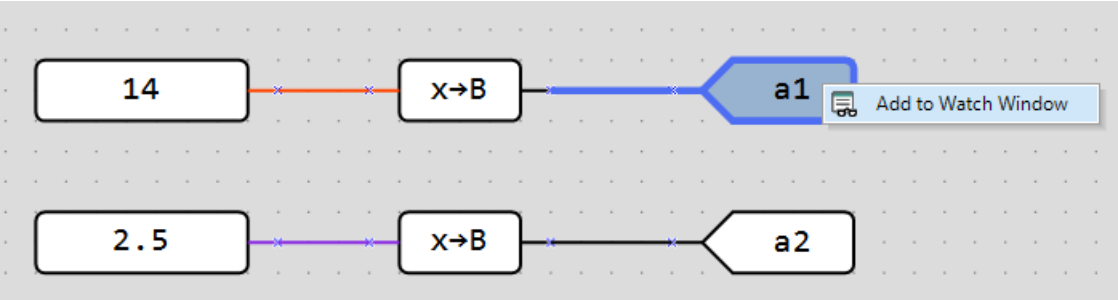


To add an input, an output or a variable to the ***Watch list*** , click the empty field in the ***Name*** column and then click on the "**...**" icon appeared to the left.

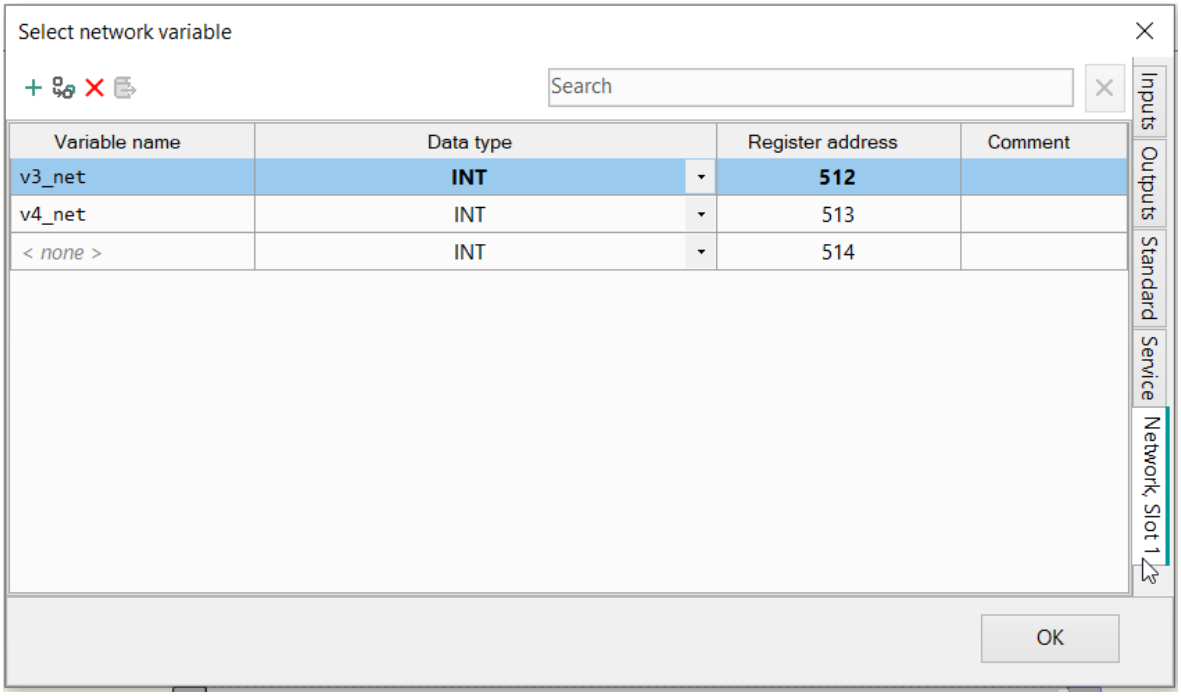


Also to add a variable to the**Watch Window**, right-click on the variable in the diagram and select **Add to Watch Window** from the context menu.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



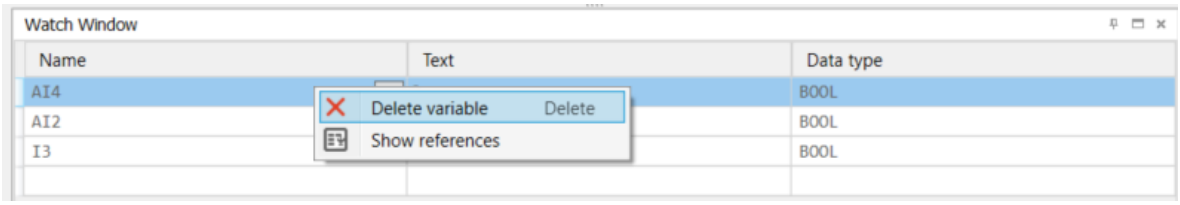
The ***Variable table*** will open. Here project variables, inputs and outputs can be selected.



The selected variables are added to the preview window.



Unused variables will appear grayed out in the watch window, their value is always shown as 0, and it is not possible to enter a value for such variables.
To delete a variable from the watch window, select the variable and press Delete, or use the variable's context menu in **Watch Window**:

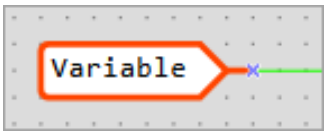


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

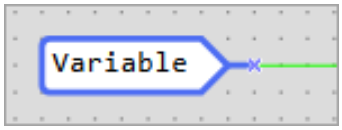
It is not possible to delete an array element that is nested within another array.

Entering Variable Values

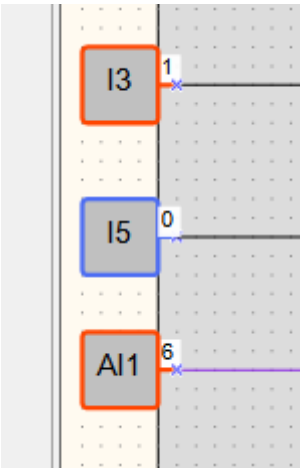
In simulator mode, variable values can be set in the watch window under the **Values** column. Variable values can also be set directly on the diagram. Double-clicking a variable opens a window with a field for entering a new value.
After entering a non-zero value for a variable, the block on the diagram will turn red for one program execution cycle. If simulation is stopped, the block will remain red until simulation is started again or the **Single Cycle Step** command is executed:



After entering a zero value for a variable (for an array variable, all elements must have the value 0/ False), the block on the diagram will turn blue for one program execution cycle. If simulation is stopped, the block will remain blue until simulation is started again or the **Single Cycle Step** command is executed:



In simulator mode, the values of device inputs can also be changed by clicking on them. Clicking on an input toggles its color and boolean value. For analog inputs, the value is set similarly to variables, using a window with an input field.



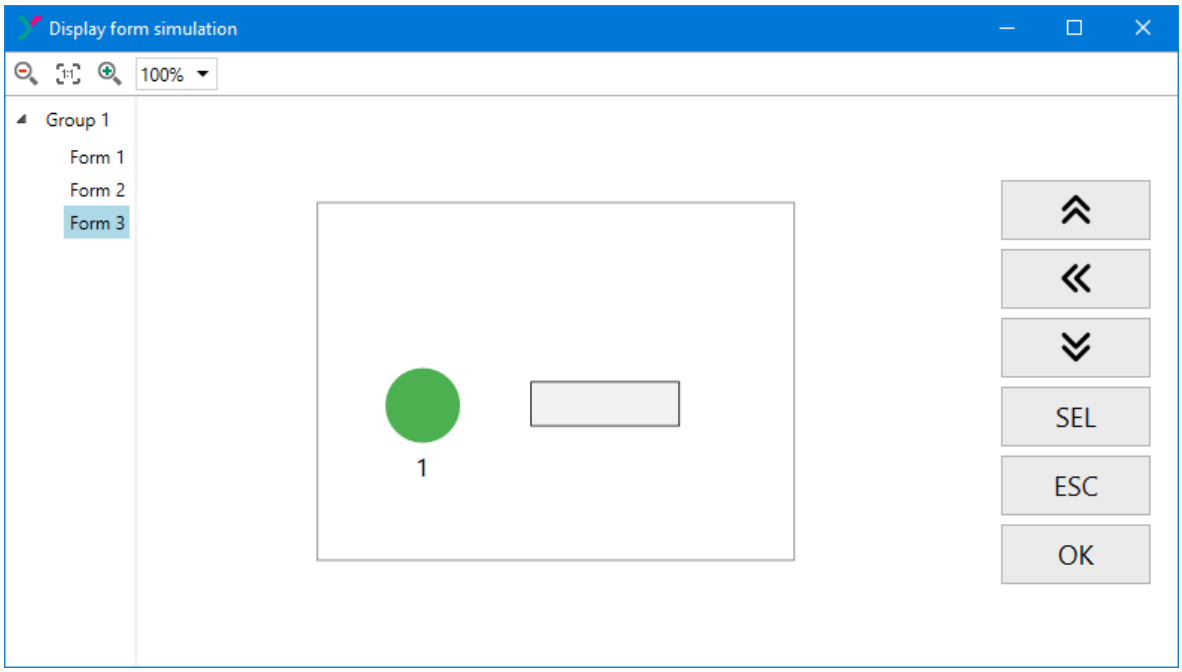
The values of the variables, inputs and outputs can be set in the ***Value*** column during simulation.

4.5.1 Visualization Simulation

In simulator mode for devices with graphic color LCD, simulation of the device display and buttons is available. After entering the mode, a window displaying the device start screen will open.

NOTE
The Graph visualization element is not supported in simulation mode.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Zooming buttons are located in the upper left corner of the window.
A list of screens available in the project is located in the left part of the window.
The center part of the window displays a simulation of the device screen.
Buttons for simulation of actions with real buttons of the device are located in the right part of the window.

Working with the screen

The following visualization elements can be edited in simulation mode:
– *Dynamic Text*
– *Int/float I/O forms*
– *Time I/O forms*
– *IP I/O forms*

Working with buttons

Buttons in the right part of the window are intended for simulation of actions with real buttons of the device.
The button actions are set for each screen. Button actions are processed for the screen that is currently displayed on the main area of the window.
Simulation buttons can be controlled with the keys ***Q, W, E, A, S, D, Z, X, C***. Pressing buttons on the keyboard is enabled only if the visualization simulation window is in focus.
In a step-by-step simulation, button hold is not enabled.

4.5.2 ST code simulation

Simulation of function or function block code written in the ST language is available in ALP. To use it, go to the **Function Editor** (or **Function Block Editor**) window and click the ▶ button on the toolbar, or in the main menu **Service** → **Simulator Mode.** The simulation panel will open. In the **Editor** window, the variable values will be displayed in the view boxes next to the variable. If arrays are declared in the program, the word **Modify** is displayed in the view window.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Main program | Function Editor "funcB" | Function block editor "all_in_one"

```
27        Timer T#0ms : time;
28        DTimer DT#2000-01-01-0:00:00 : dt;
29        MassB Change : array [0..4] of bool;
30        MassUd Change : array [0..4] of udint;
31        MassF Change : array [0..4] of real;
32        MassT Change : array [0..4] of time;
33        i 0 : udint;
34    end_var
35
36    locU FALSE := U FALSE and udint_to_bool(Res 0 );
37    locRes 0 := Res 0 *2 + 4 - time_to_udint(Timer T#0ms ) + dt_to_udint(DTimer DT#2000-01-01-0:00:00 );
38    locN 0 := N 0 /2 - 5;
39
40    if locU FALSE or U FALSE then
41        Resq 0 :=2;
42    else
43        Resq 0 :=3;
44    end_if
```

Fig. 4.4 Function block editor window in simulator mode

Color coding of the view windows:

13.3 — the current value of the variable;

12 — the value of the variable has been changed by the user;

13.3 — the value of the variable has been fixed by the user

Change — the value of one or more elements of the array (not all) has been fixed by the user;

??? — error: incorrect variable value (for example, when a non-existent array element is specified using a variable).

If you left-click on the view window to enter the value of an integer variable (udint) or a floating point variable (real) in the simulation mode, a window will appear in which you can enter a new value of the variable and, if necessary, fix the value.

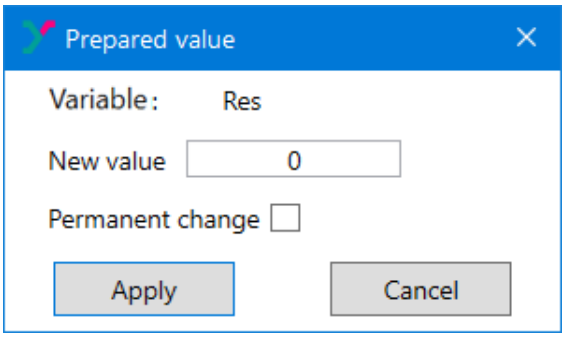


Fig. 4.5 Changing the value of a udint variable

When you left-click on the view window to enter the value of a Boolean variable (bool) in simulation mode, a window will appear where you can select true or false and, if necessary, fix the value.

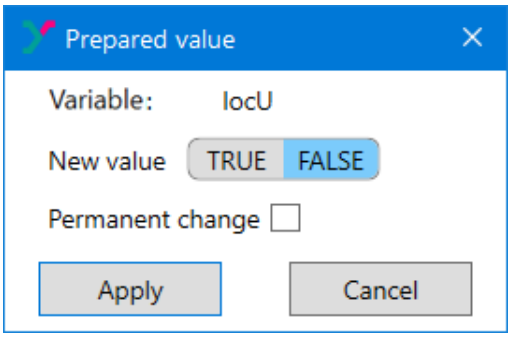


Fig. 4.6 Changing the value of a bool variable

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

When you left-click the view window to enter the value of a time variable (time or DT) in the simulation mode, a window will appear where you can enter the value in the format T#0d0h0m0s0ms (for time) or DT#2000-01-01-0:00:00 (for dt), and fix the value if necessary.

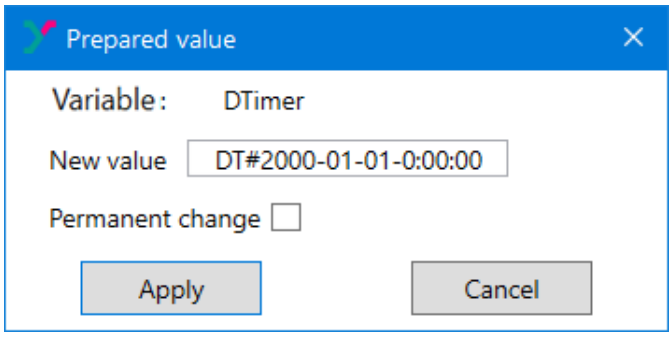


Fig. 4.7 Changing the value of a time variable

NOTE
The time is recalculated automatically after the **Accept** button is pressed. For example, the entered value of the variable T#80m will look like T#1h20m when the **Change Value** window is reopened.

If an incorrect value or no value is entered, an error message will appear. The **Accept** button will be disabled.

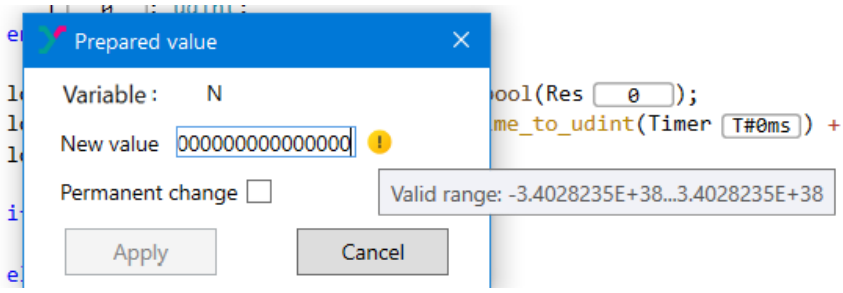


Fig. 4.8 Value entry error

To enter values in the array elements, left-click on the view window labeled **Modify**:

– If the element index is specified by a constant or using a variable, the Change Value for Specific Array Element window will appear;
– If the index of an array element is specified as the result of a mathematical expression or as the result of a function/function block, a window with all elements of the array and variable values available for input will appear:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

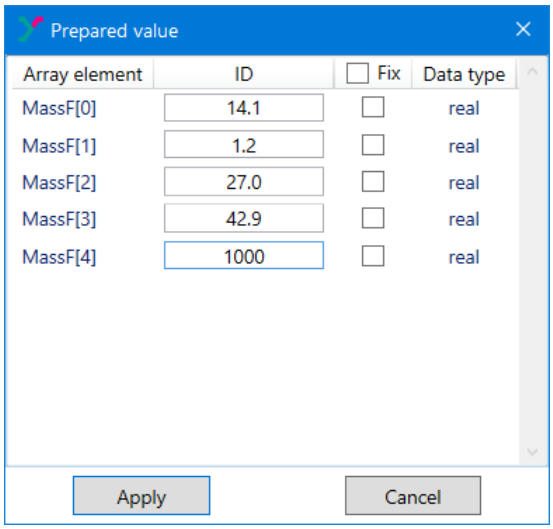


Fig. 4.9 Changing values of array elements with real variables

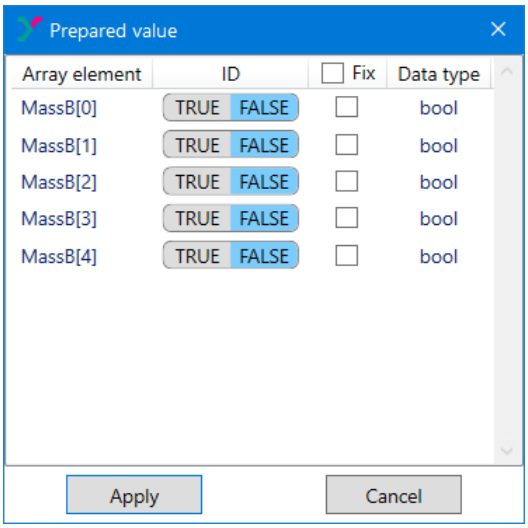


Fig. 4.10 Changing values of array elements with bool variables

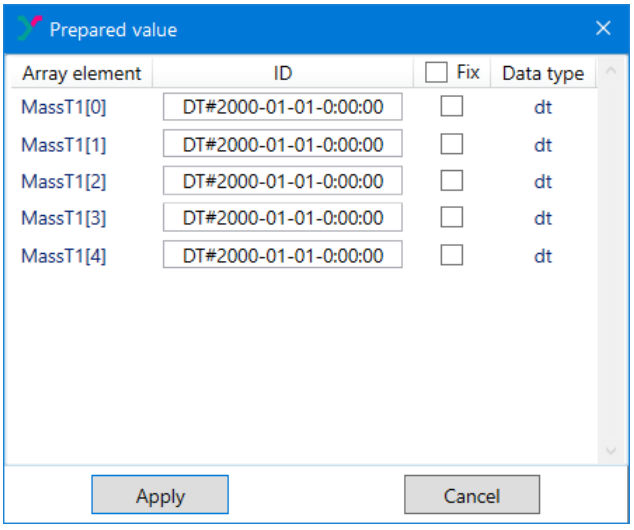


Fig. 4.11 Changing values of array elements with dt variables

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

If an incorrect value or no value is entered, an error message will appear. The **Accept** button will be disabled.

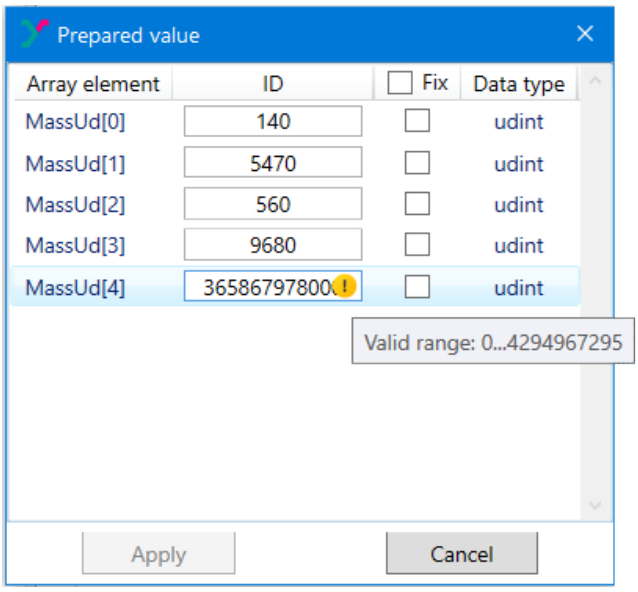


Fig. 4.12 Value entry error

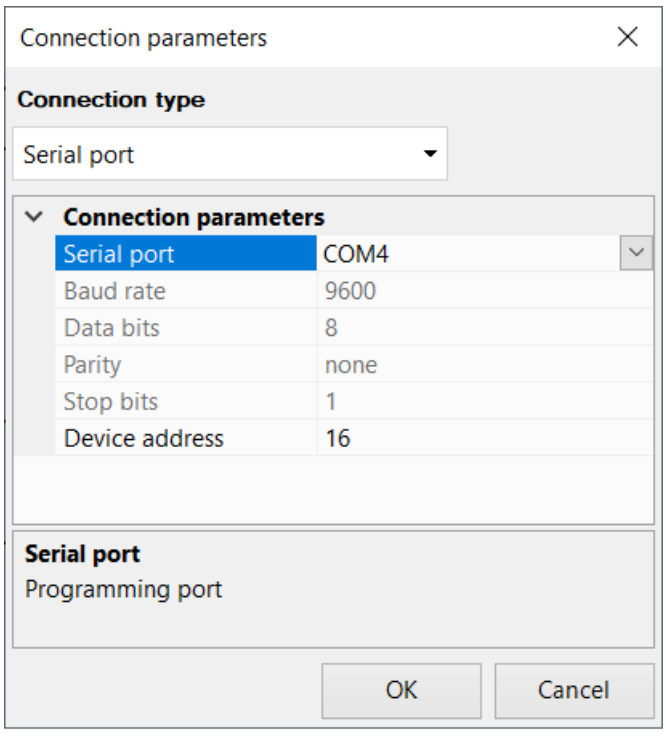
4.6 Connection to device

CAUTION
The device must be powered off before connecting to PC.

All devices can be connected to PC over USB. If the device has an Ethernet interface, it can be connected over Ethernet. To temporarily interrupt the connection, use ***OFFLINE mode***.

NOTE
For second generation devices (PR103, PR205, PR225), connection via RS-485 is available using the IC4M USB converter. In this case, the device's operating speed is fixed at 9600 bps.

Connection over USB



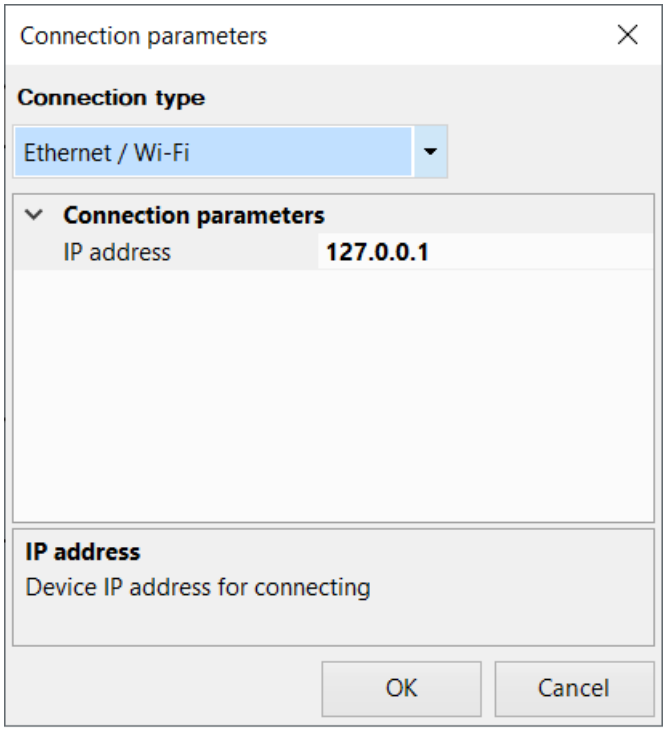
Devices can be connected to PC over USB.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. Connect the device to a USB port of the PC and switch the device on.
2. Start ALP and select the menu item ***Device > Port settings***.
3. Select ***Serial port*** for ***Connection type***.
4. Select the serial port in the opened dialog. The number of the emulated COM port can be found in the Windows Device Manager under “Connections (COM and LPT)”.
5. Enter the ***Device address*** (16 by default) and confirm with ***OK***. All other parameters are displayed only for your information.

If connection is established successfully, status indicators will show the information about connected device and serial port.

Connection over Ethernet



To connect the device to a PC via Ethernet interface, consider the following steps:

1. Connect the device to the same local network as the PC.
2. Find out the IP address of the connected device. The default IP address is specified in User Manual for the device.. The current IP address of the device can be read using the software.
3. Select ***Ethernet / Wi-Fi*** for ***Connection type***.
4. Enter ***IP address*** of the connected device and confirm with ***OK***.

If connection is established successfully, status indicators will show the information about connected device and serial port.

4.7 Upload project to device

Upload project to device

CAUTION
When a new project is uploaded to the device, the program already stored in the device memory (ROM) will be replaced by the new one.

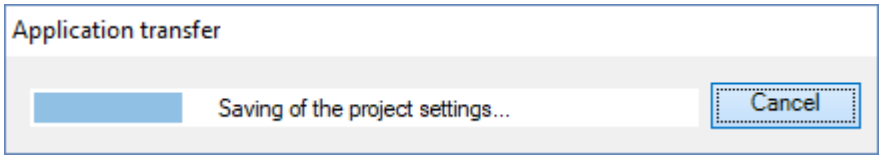
Proceed as follows:

1. *Connect* the device to PC.
2. Power on the device.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3. Adjust the port settings if necessary.
4. Upload project to the device.

   The project can be uploaded to the device using the menu item ***File → Transfer application to device*** or clicking the icon in the toolbar. When the upload is completed, the device can be powered off and disconnected from the PC.



If the target device does not match the connected device, a warning message will be displayed.

NOTE
When the program transfer is completed, the device goes to the operating mode and the program starts automatically.

OFFLINE mode

In the ***OFFLINE*** mode, the connection between ALP and the device is interrupted.
The mode is helpful when you work with two ALP instances running on PC and trying to communicate with the same device. Both applications will alternately occupy the port and the connection to the device will be constantly interrupted.
The ALP instance that is not to interact with the device has to be set to OFFLINE mode.
OFFLINE mode can be activated / deactivated using the menu item ***Service→ OFFLINE mode*** or by clicking the *status indicator 3.7* ***Device***. With the next click OFFLINE mode deactivated.

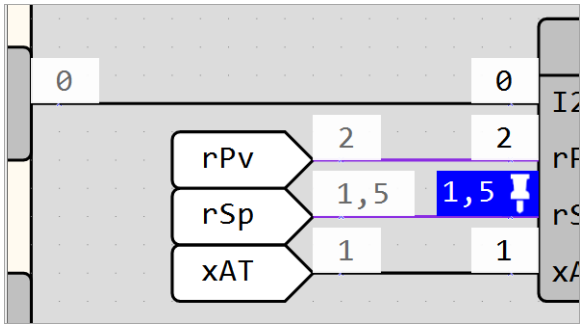
4.8 Online debugging

The online debug mode is designed for testing the program in real time with actual values from the device inputs. Online debugging is carried out only with the device connected to a PC. In online debugging mode, it is not possible to make changes to the project.

Sequence and rules of work in online debugging mode

Online debugging can only be performed after the program has been loaded into the device. The program in the device must match the program in the project diagram. The device must have the same version *firmware*.
To perform online debugging you should:

1. Start online debugging by clicking the button on the toolbar - the online debugging panel will open.
2. Check the logic of the program operation with real values at the device inputs, which are displayed on the diagram.



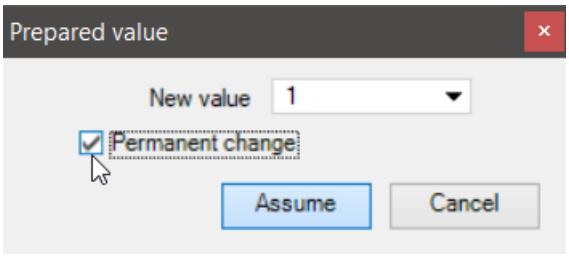
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3. To make adjustments, exit the online debugging mode by pressing the button .

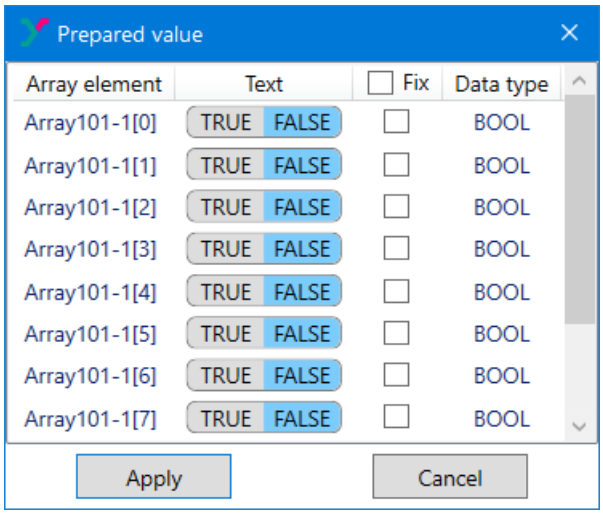
Online debugging is available only for the project diagram (to check the operation of macros, functions and function blocks in the ST language, you should use *Simulation*).

Manual assignment of values

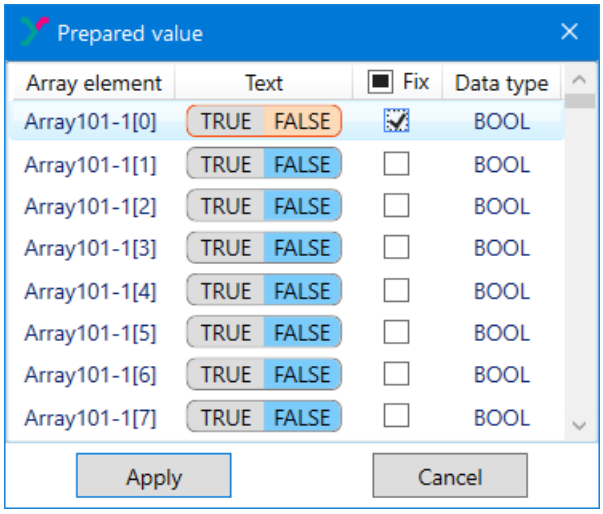
In online debugging mode, you can set the required values for output variable blocks, as well as for function inputs, function blocks, and macros. To set a value in online debugging mode, click on the variable value plate and enter the new value in the window that opens.



For a variable of the array type, it is possible to enter the values of all array elements:



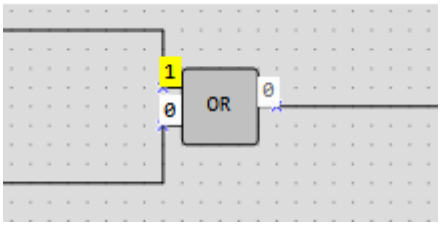
The entered values of an array variable element in the "Change Value" window are highlighted in orange:



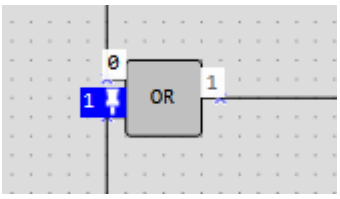
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

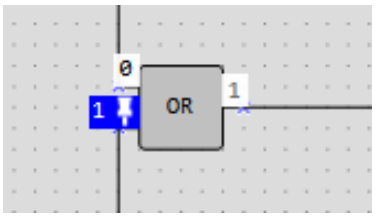
Displaying values on a diagram

If you do not check the box **Fix the values**, then the new value will be valid for one working cycle. At the end of the current cycle or at the beginning of the next one, the specified value can be overwritten. During the action, the set value is highlighted in yellow.

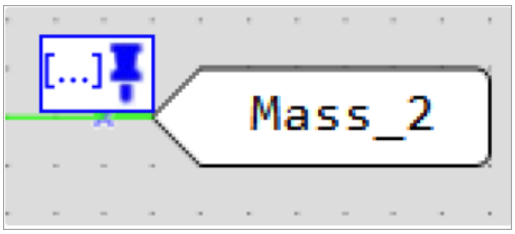


If you lock a set value, it will remain in effect every work cycle until you manually change it to another value, or until you release the lock. Value fixing is disabled when exiting online debugging mode. For array type variables, it is possible to fix one, several, or all values of the array elements. The fixed value (for arrays - fixing the values of all elements) is displayed on the diagram as a blue box:

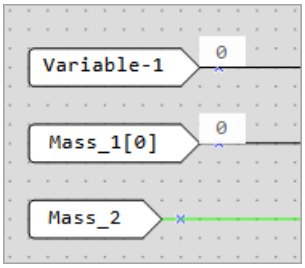




If you fix the value of one or more elements of the array (not all), the bar is displayed in white with a blue border:



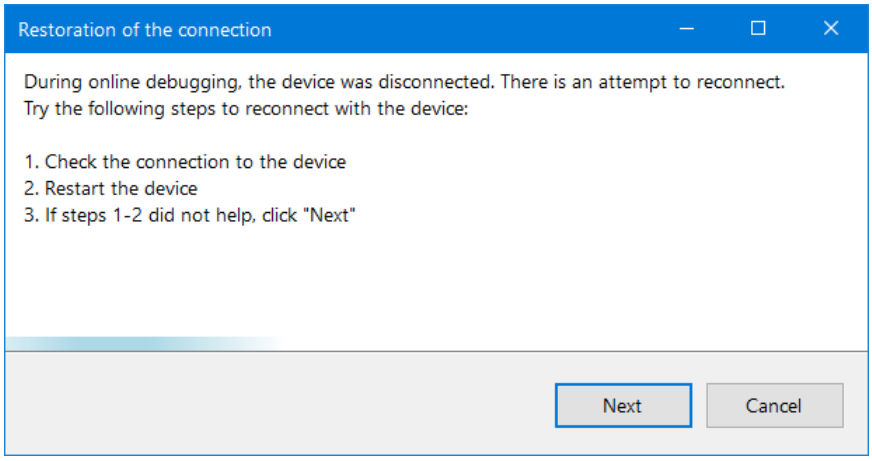
It is not possible to set the values of the input variable blocks, so the values of these input variables are displayed in gray on the diagram. For input blocks of array type variables, the value is not displayed.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

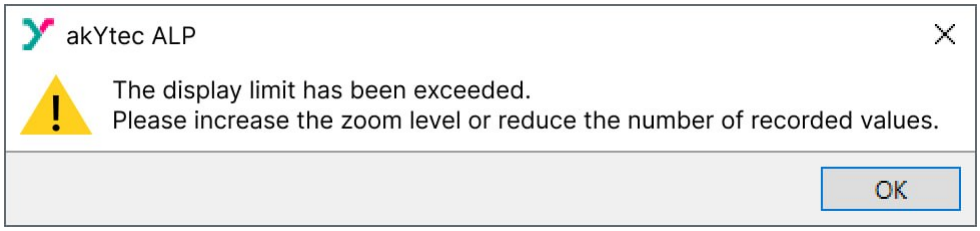
Troubleshooting

If the connection with the device is lost, the online debugging mode will be disabled after 10 seconds, and the device will switch to operating mode. If you manage to restore the connection, online debugging will continue, but previously recorded values will be disabled.



Limit of transmitted values

Each device modification has its own limit for the memory buffer used for data transfer in online debugging mode. If the diagram displays empty bars, then a limitation is triggered, and you should increase the diagram scale to reduce the number of values displayed in the diagram window. Committed values remain committed even if they are not displayed in the schematic window, and continue to occupy the memory buffer. If the device's memory buffer is completely full, then clicking on an empty plate will not open the "Change Value" window. Instead, an information message will appear about exceeding the limit of displayed values:



If the device memory buffer is full, but the array type variable has at least one element with a fixed value, then:

– The fixed value is displayed in the “Change Value” window and can be changed or unfixed;
– The values of uncommitted array elements in the “Change Value” window are not displayed and are not available for input.

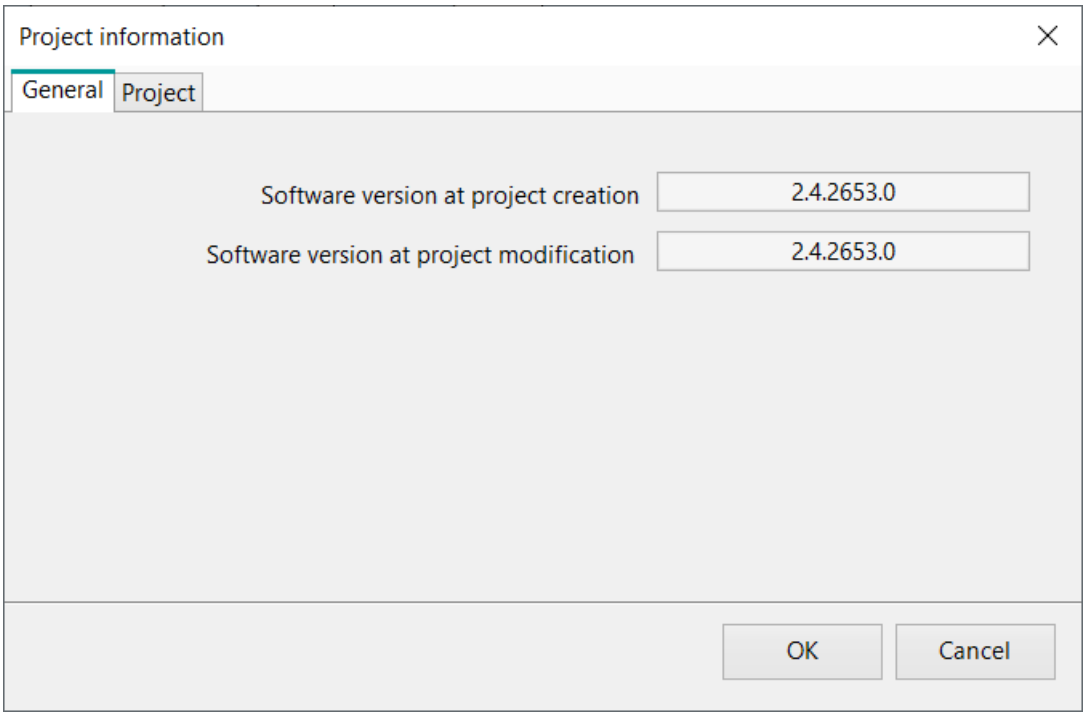
4.9 Project information

Use the menu item ***File*** → ***Project information*** to view and modify the information about the program.

General

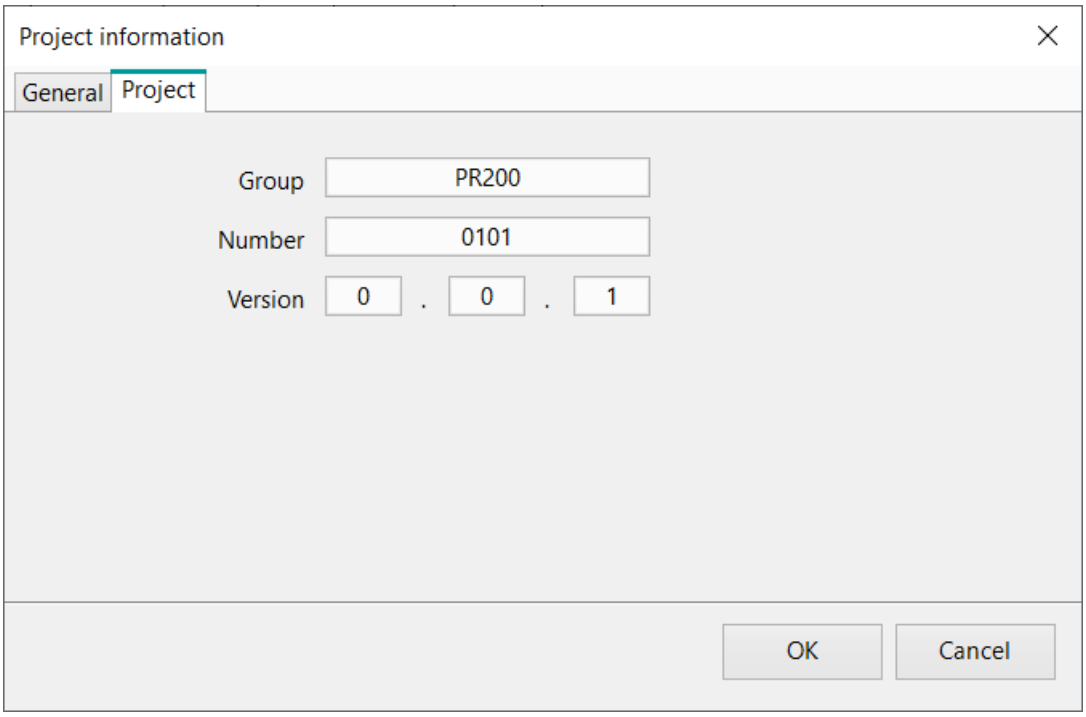
The tab ***General*** contains the information about the software version.
Software version at project creation – the version of the software in which the project has been created.
Software version at project modification – the version of the software in which the project has been modified.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Project

The tab ***Project*** is not available for each device. In the tab you can specify information about the group, number and version of the program to be displayed in the ***Device information*** window of the connected device after the project is saved to it.



– ***Group*** –project group name
– ***Number*** – project number within the group
– ***Version*** – project version

Click ***OK*** to save the information in the project, or click ***Cancel*** to discard it.

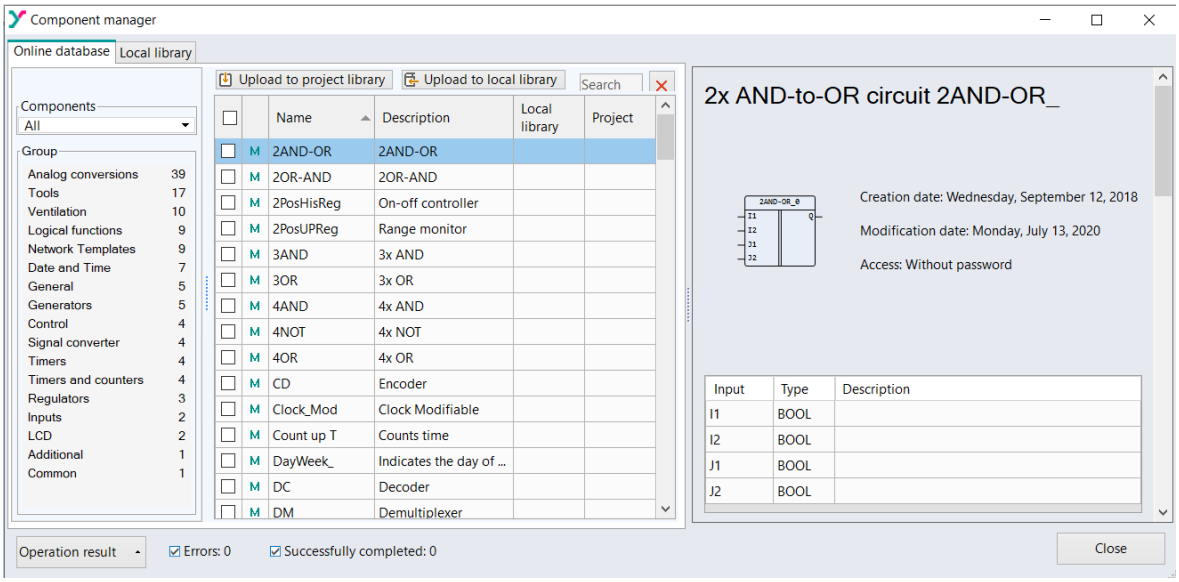
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.10 Component manager

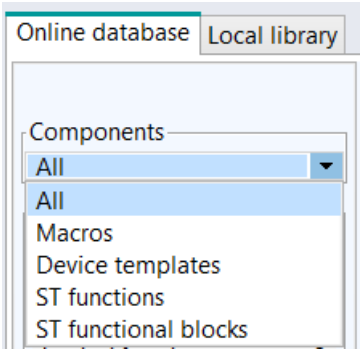
New macros and device templates can be downloaded from akYtec Online Database. Component Manager is the tool for all interactions with this database. The internet access is necessary for this interaction.
Select the menu item ***File*** → ***Component manager*** to open it in a separate window.

Online database

The **Online database** tab displays a list of components available from the akYtec company's online database.

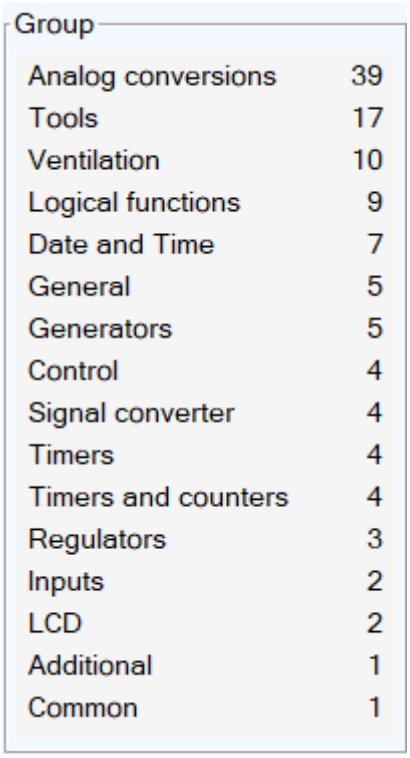


The **Components** dropdown menu allows you to filter the list by type:

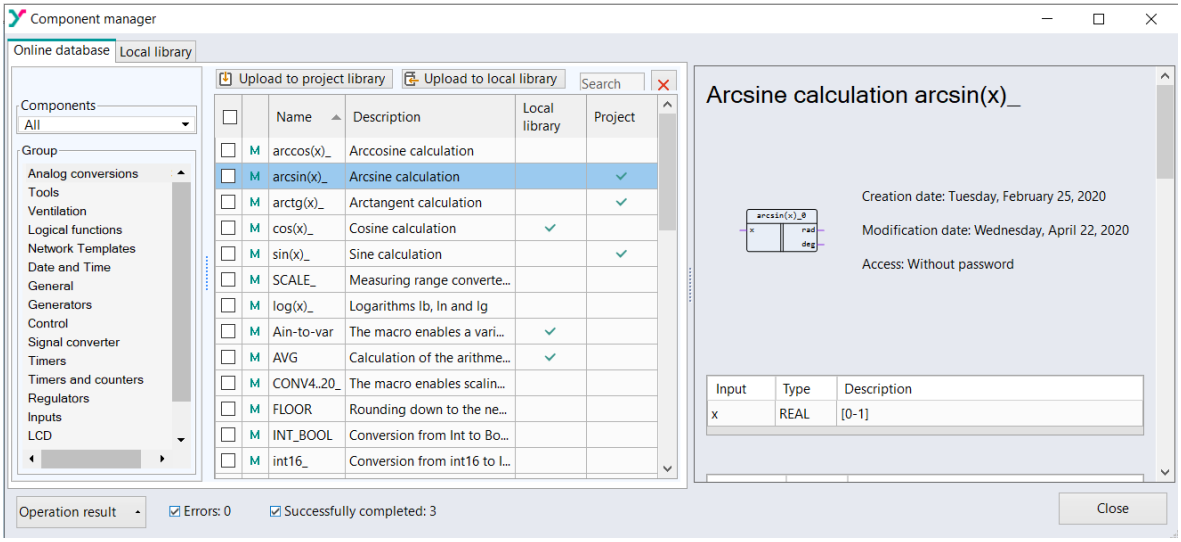


Macros are further divided into categories based on their function:

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



A full description of a component can be viewed by selecting it in the list. The component description is displayed in the right part of the window. Acrobat Reader is required to view the complete component documentation.



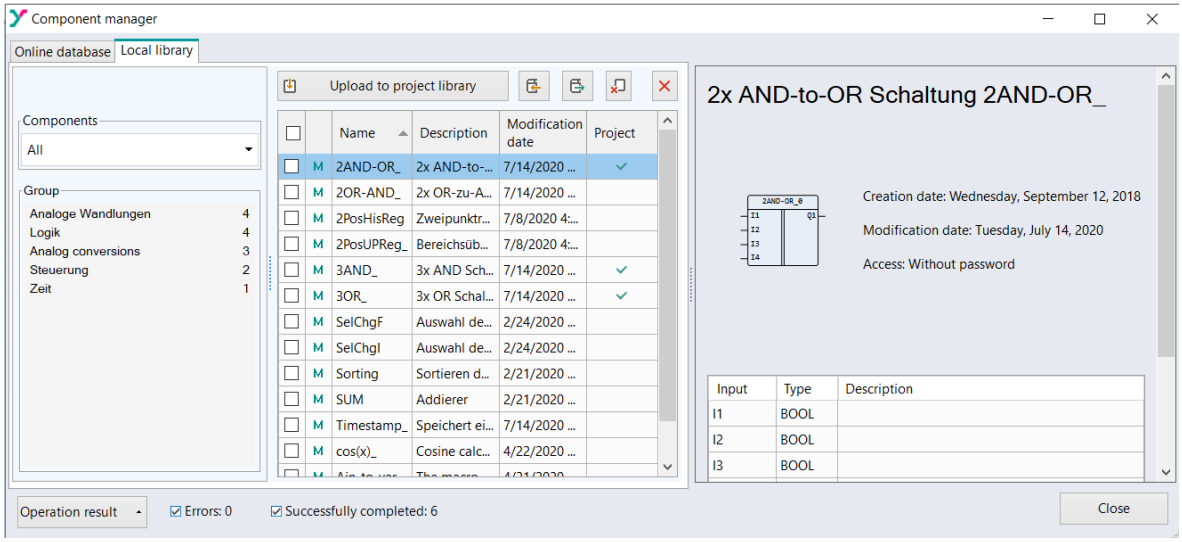
To make components available without an internet connection, check the boxes next to them and click the **Upload to local library** button – the components will be added to the local storage and will appear in the **Library** tab.
To immediately load components into the current project, check the boxes next to them and click the **Upload to project library** button – the components will be added to the **Component library** panel under the **Project macros** section.
Successful download of components to local storage or the current project is indicated by a green checkmark in the **Library** and **Project** columns, respectively.
To connect a network template to the device configuration, follow the recommendations in the *Network device templates* section.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Local library

The **Local library** tab displays the contents of the component local storage on the user's PC. To load components into the current project, select the desired components and click the **Upload to project library** button – the components will be added to the **Component library** panel under the **Project macros** section. Successful loading of components into the current project is indicated by a green checkmark in the **Project** column of the list.

NOTE
Loading components containing arrays into the **Project Library** is possible starting from ALP version 3.0.



NOTE
The component local storage is located at: *C:\Users\[username]\Documents\akYtec ALP \Library*

The Component Manager supports the following component management functions:

– – import a file into local storage.

– – export a file from local storage.

– – delete selected components.

Component import

Import allows you to load components that are stored on your PC but are not yet part of the local storage.
To import a component:

1. Click the button.
2. In the window that appears, select the files to import.
3. Click the **Open** button. The files will appear in the **Library** tab.

Import Restrictions for Earlier Versions of ALP

When importing a component containing functionality not supported by the target version:

– Importing into ALP version 2.11.370 or higher – the component will be loaded but will appear with restrictions in the **Component manager**. The description will indicate the ALP version from which the component is supported.

– Importing into a ALP version lower than 2.11.370 – the component will be loaded, but an error message window will appear when you try to select it from the library.

Any operations performed with such components will result in errors. It is not possible to use components not supported by the current version of ALP in a project.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Component export

Export allows you to save components to your PC for further modification or transfer.
To export a component:

1. Check the box next to the desired component in the **Library** tab.
2. Click the button.
3. In the window that appears, select the export path.
4. Click the **OK** button. The component will be saved as a file to the specified location.

Further information

Functions for the component documentation are available according to the installed version of Acrobat Reader, including saving the file in PDF format and printing.

To view the results of operations in the Component manager, click the Operation result button – a message line will expand at the bottom of the window.

```
cos(x)_ - Component has been successfully added to the library
Ain-to-var - Component has been successfully added to the library
AVG - Component has been successfully added to the library
2AND-OR_ - The macro is added to the project
3AND_ - The macro is added to the project
3OR_ - The macro is added to the project
```

4.11 Macro development

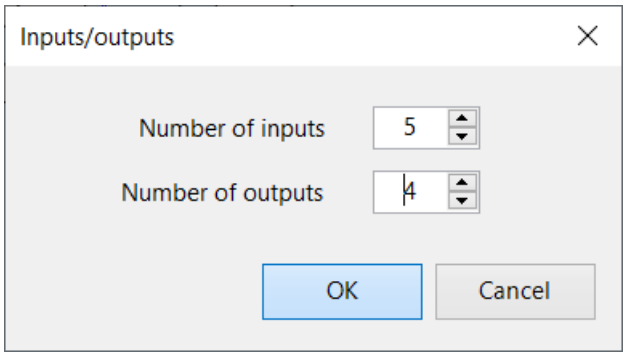
A macro is a user-defined algorithm for use in a project.
Main operations with macros:

– **Saving** to the *Library* for future use in the project
– **Exporting** to a file for use in other projects
– **Importing** previously created macros
– **Downloading** macros from the online database into the project

New macro using main menu

To create a macro:

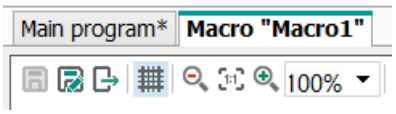
1. Select **File** → **Create Macro** from the main menu. In the dialog box that opens, specify the number of inputs and outputs for the macro:



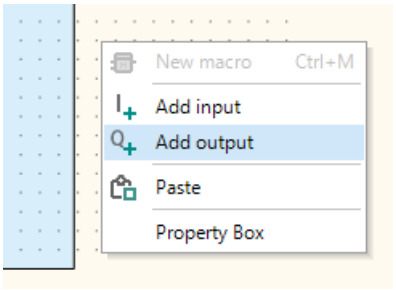
NOTE
The number of inputs and outputs can be changed after the macro is created.

2. Develop the macro's algorithm in the **Macro editor** tab, similarly to developing a program on the diagram.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

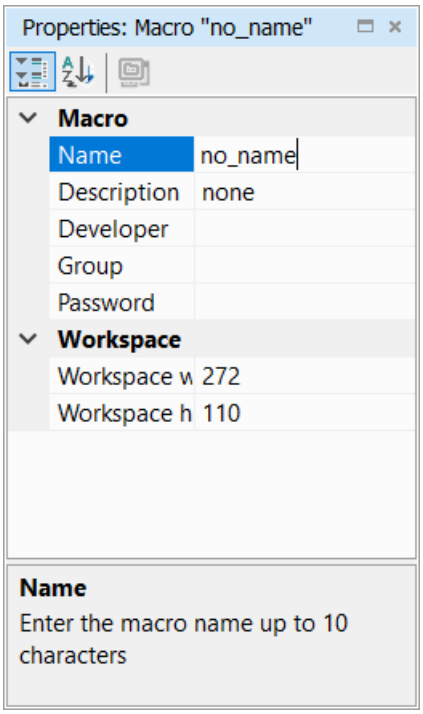


The number of inputs and outputs can be increased using the editor's context menu or the and buttons.



To delete an input or output, right-click on it and select the **Delete** command.

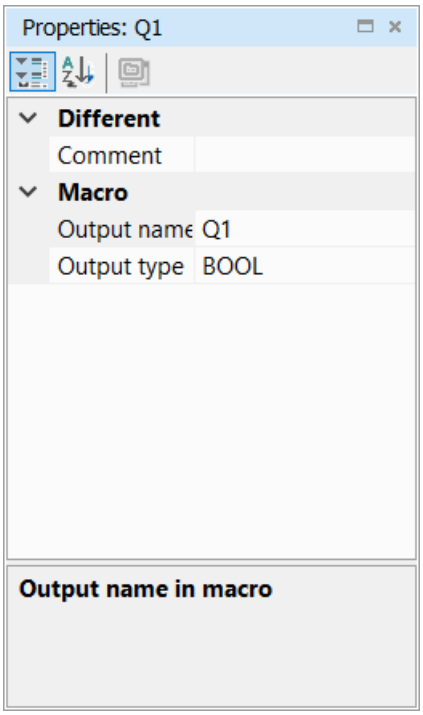
3. Set a name and description for the macro in the properties panel.



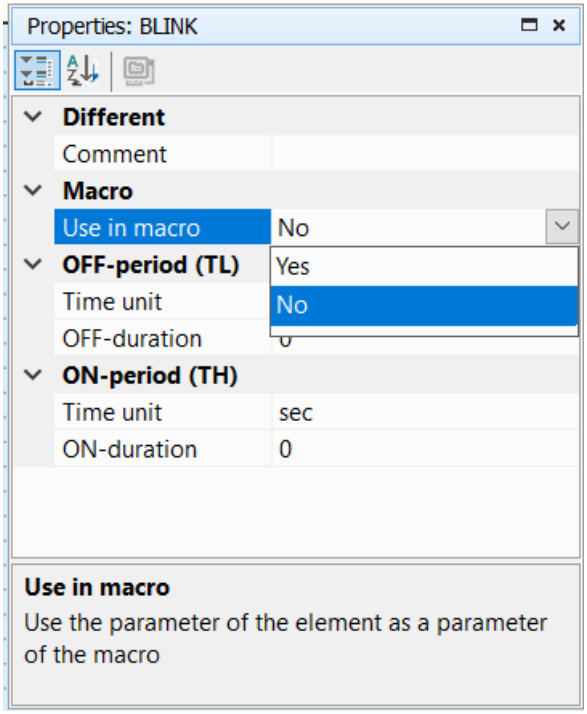
The entered name will be shown in the macro editor's title bar and on the diagram.

4. To define the type and name for inputs/outputs, select an input or output and configure it in the properties panel.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



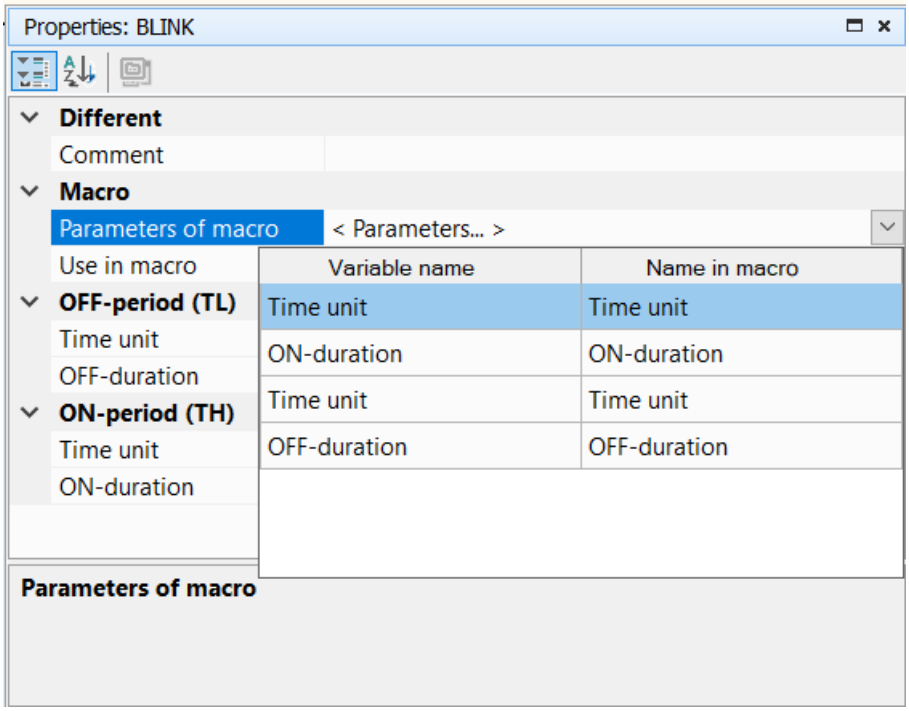
5. Next, you can set the "visibility" of the settings for Function Blocks (FBs) used within the macro from the main program.



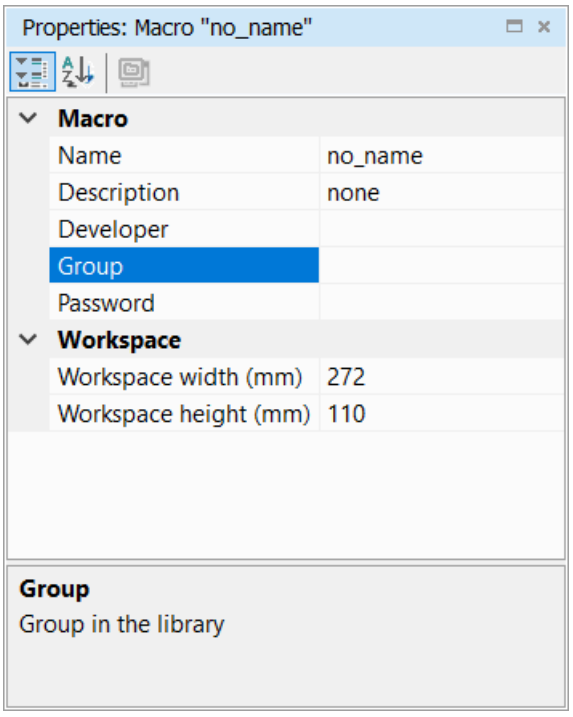
To make a selected FB's settings available when the FB is used in the program, configure the block's property **Use in Macro – Yes** in its properties panel.

After selecting **Yes**, a dropdown list **Rename variables** will appear, listing the available variables. For example, for the BLINK FB:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



6. After creating the macro, start the *simulator mode 4.5* and verify its logic.
7. Before saving the macro, you can fill in the fields: **Name**, **Description**, **Developer**, **Group**, and **Password**.



It is recommended to give the macro a short and clear name. The content of the **Description** field is displayed under the macro's image in the **Library Box** panel.

If no password is set (the **Password** field is left blank), access to edit the macro is open to everyone.

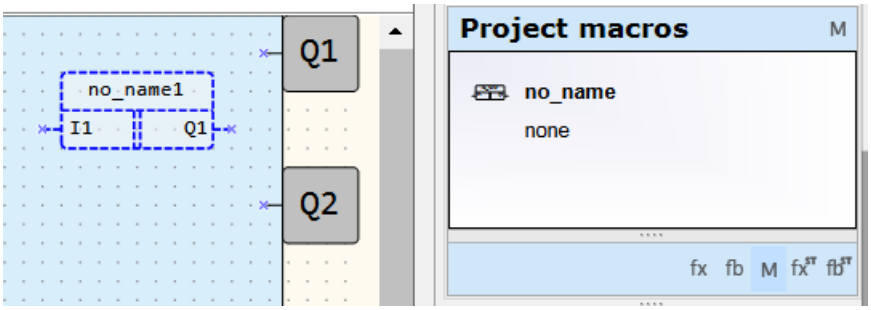
If the **Group** field is not filled, the macro is saved in the project library under the **Other** group.

The macro can be saved by selecting **File** → **Save Macro As** from the main menu or by clicking the button on the macro editor toolbar.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

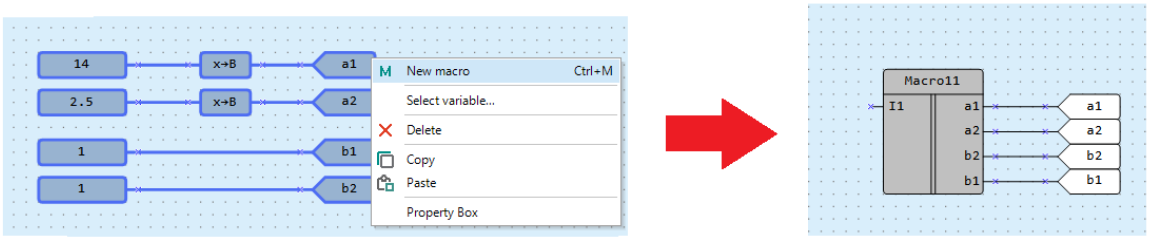
A saved macro is only available for the currently open project. To use the macro in other projects, it must be exported.

8. Select the **Project Macros** section in the **Component Library** panel and drag the macro onto the project canvas.

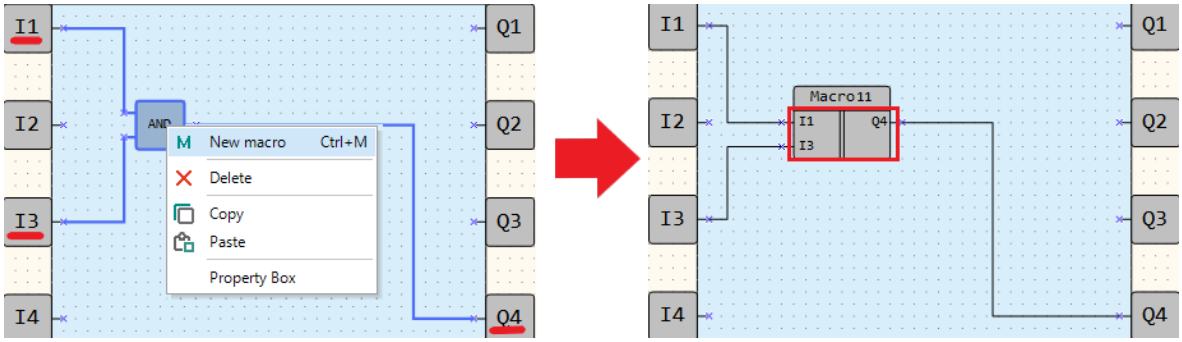


New macro using context menu

A macro can be created by selecting elements on the project diagram and choosing **New macro** from the context menu. A macro block will appear in place of the selected elements.



The macro's inputs and outputs are named according to the elements and blocks connected to them at the time of creation. If an output is connected to multiple elements, it is named by default (*Qn*, where *n* is the sequential number of the macro output).



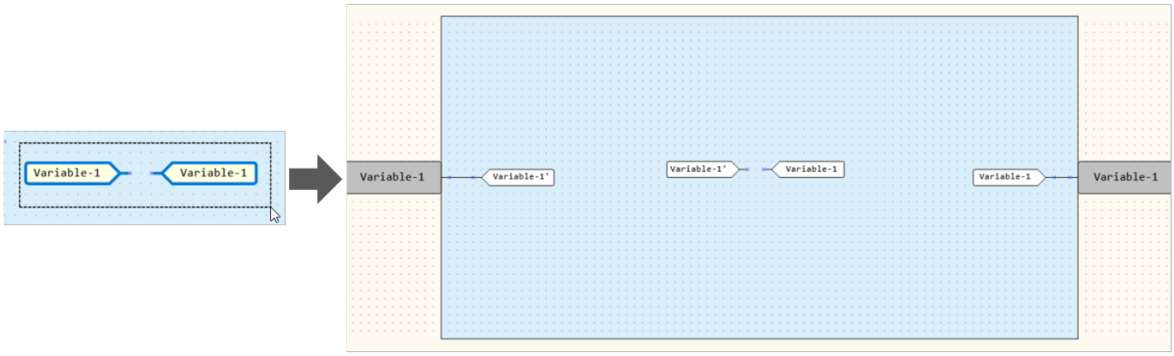
The created macro and its input/output names can be edited in the **macro editor** by selecting **Edit** from the block's context menu.
The number of inputs and outputs of the macro will correspond to the number of connected input and output links in the selected diagram area. If elements without any connections are selected, a macro with one input and one output is created.
Elements that are not included in the macro when created on the diagram:

– Inputs and outputs
– Service variables
– Network variables
– The PID controller block (PID)

If any of the above elements were selected during macro creation, they will be connected to the corresponding input/output of the macro block. If the selected area includes a Function Block (FB)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

from which a value is read or to which a value is written, the read/write block for that FB is included in the macro, even if it is located outside the selected area.
If standard variables fall within the selection area, they are copied into the macro with the same name. Variables inside the macro are not identical to the variables on the main diagram. If all blocks associated with a single variable are included inside the macro, they disappear from the main diagram.
If, during macro creation, both the input and output blocks of an ***in-use*** variable fall within the selected area, a new variable will be created inside the macro. This new variable inherits all properties of the original selected variable: type, non-volatility, and default value. Its name is formed based on the original variable's name with an apostrophe (') added at the end.



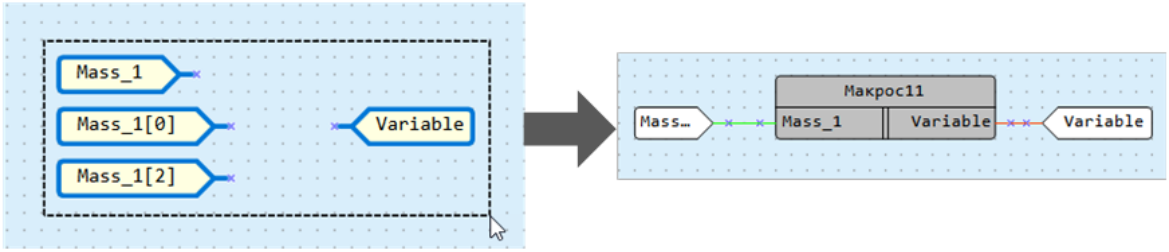
The same rules apply if, during macro creation, the selected area includes:

– Both the input and output blocks of an **in-use** array element.
– Both the input and output blocks of an array that contains at least one **in-use** element.
– An input/output block of an array containing at least one **in-use** element, along with an input/output block of an element from that **in-use** array.

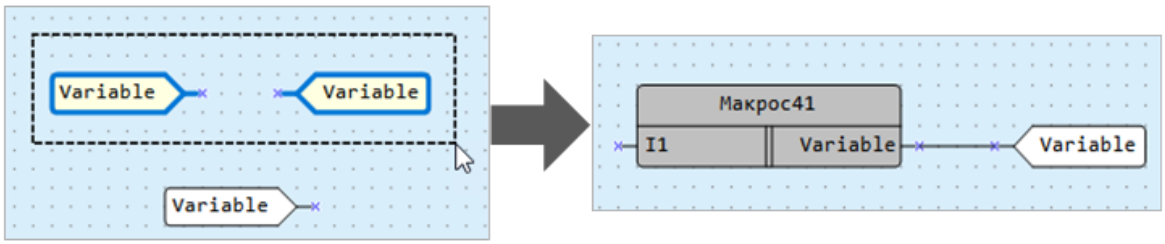
The name for an array element created under these rules will be formatted as follows: square brackets ([ ]) will be replaced with curly braces ({ }), and an apostrophe (') will be added at the end of the name.

Example:
Mass_1[0] → Mass_1{0}'

If, during macro creation, the selected area includes the input block of an array and input blocks for elements of that same array, only one input of the Array type, named after the array, will be created inside the macro:



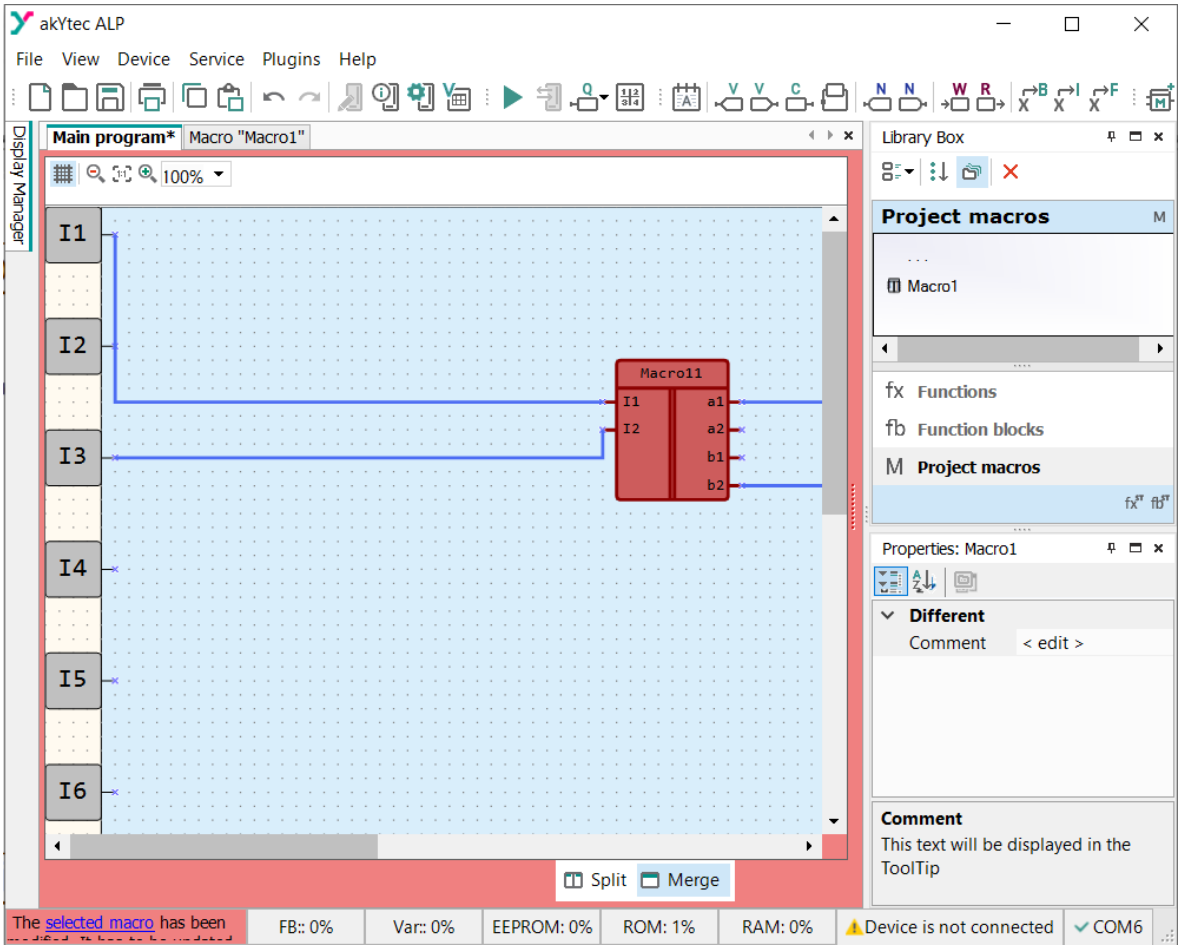
If, during macro creation, the selected area includes both the input and output blocks of a single **unused** variable that also exists outside the selected area on the diagram, these blocks are copied into the macro with the same name, and only the output variable block remains on the original diagram.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

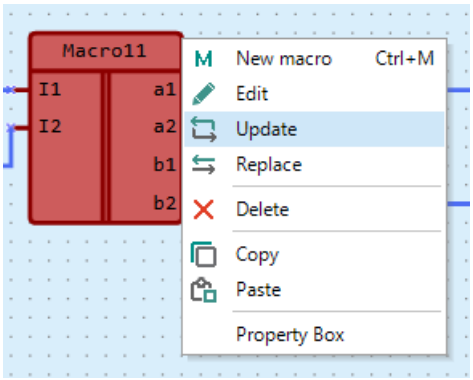
Update macro

If a macro used in the program has had its name, type, number of inputs/outputs changed, or if an element (variable, constant) with the property **Use in Macro – Yes** has been added, ALP will require it to be updated and will highlight it on the canvas. A macro is considered changed after saving edits in the **Macro editor**.



To update a macro on the canvas:

1. Right-click on the highlighted macro and select **Update** from the menu that appears.



NOTE
In the case of a nested macro, ALP will prompt you to navigate to it. Actions for a nested macro are similar.

2. After updating the macro, ALP will automatically proceed to the next macro requiring an update.

A macro is updated in the project according to the following rules:

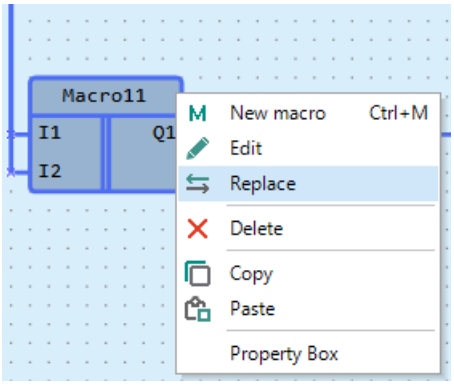
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– If the type or name of an input/output with an attached connection has been changed in the macro, that connection will be broken upon update;
– If inputs/outputs have been added to the macro, the connections of existing inputs/outputs are not broken during the update.

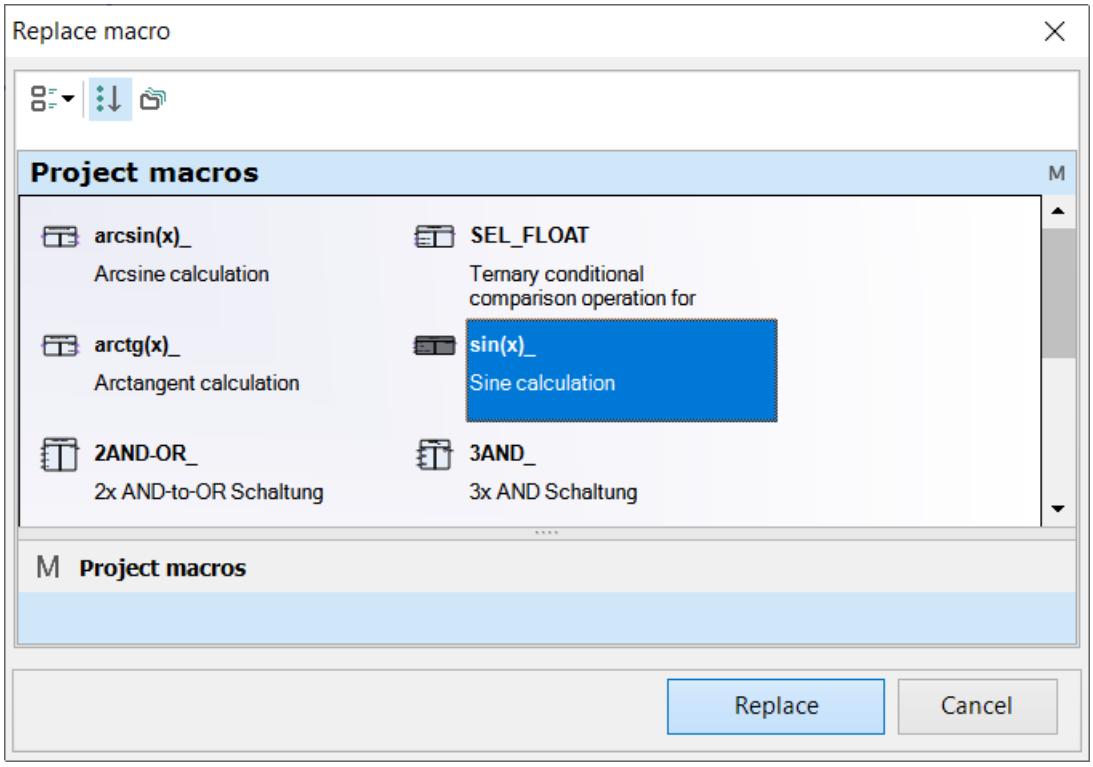
NOTE
ALP identifies connections to macro inputs/outputs by name and type. If you change the name or type of an input/output with an attached connection in the macro and create a new input/output with the same name and type, ALP will automatically reconnect the connection to the new input/output when the macro is updated.

Quick macro replacement

If you need to replace a macro with another one from the project macros, click the **Replace** button in the macro's context menu.



The "Replace Macro" window will open.



If the type or name of an input/output with an attached connection has been changed in the macro, that specific connection will be broken upon replacement, but all other connections will be preserved.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

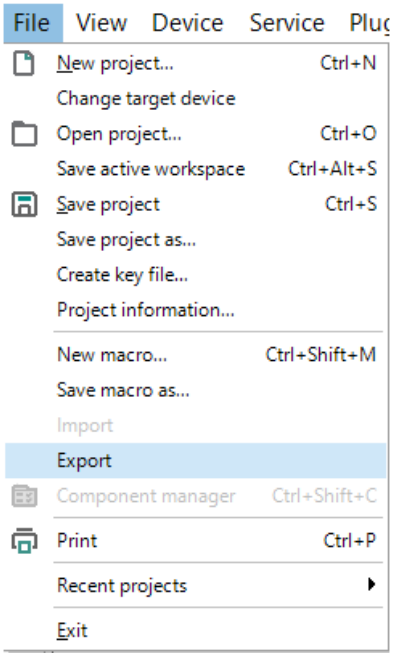
Export macro

Exporting a macro to a file is only possible when the macro editor window is open. To export a macro, select **File** → **Export** in the main menu.
To export a macro:

1. Open the macro in the editor.

   If you need to edit the macro before saving, you should drag it onto the project canvas and select **Edit** and make changes in the macro context menu.

2. Select the main menu item **File** → **Export**.



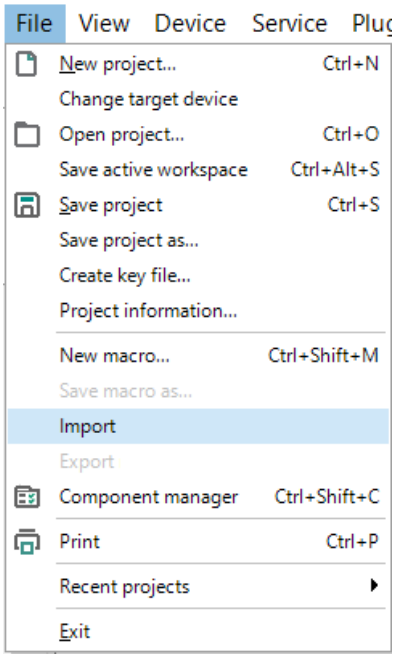
3. In the window that opens, select a location and save the macro file with the extension *.*tpl*or *.*tple*. After saving, a message indicating that the macro was exported successfully will be displayed.

Import macro

If you need to use a macro created in another project to create a program, you can import the required macro into the project.
To import a macro use the main menu item **File** → **Import**.

NOTE
The **Import** item is active only when the focus is on the workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



In the window that opens, select the desired file and click the **OK** button. The macro will be added to the ***Library Box*** in the ***Project macros*** section, and can now be used in the project.

NOTE
Importing macros that contain variables of the Array type or array elements is possible in ALP version 3.0 and higher.

Copy macro

Macros can be copied from project to project for reuse and reduced development time. To copy a macro, select the macro block in the source project and click the on the toolbar or select the ***Copy*** command in the block context menu. The macro is inserted into another project by clicking the button on the toolbar or by selecting the ***Insert*** in the canvas context menu. You can also use keyboard shortcuts to copy and paste, see *Keyboard shortcuts*.
Once inserted, the macro will be available in the ***Project macros*** section of the ***Library Box***.

NOTE
Inserting macros that contain variables of the Array type or array elements is possible in ALP version 3.0 and higher.

4.12 Using ST function

In ALP, it is possible to create user-defined functions in the ST language.

NOTICE
ST funtions reserve space in ROM memory after they are added to the project library, regardless of whether they are used in the project or not.

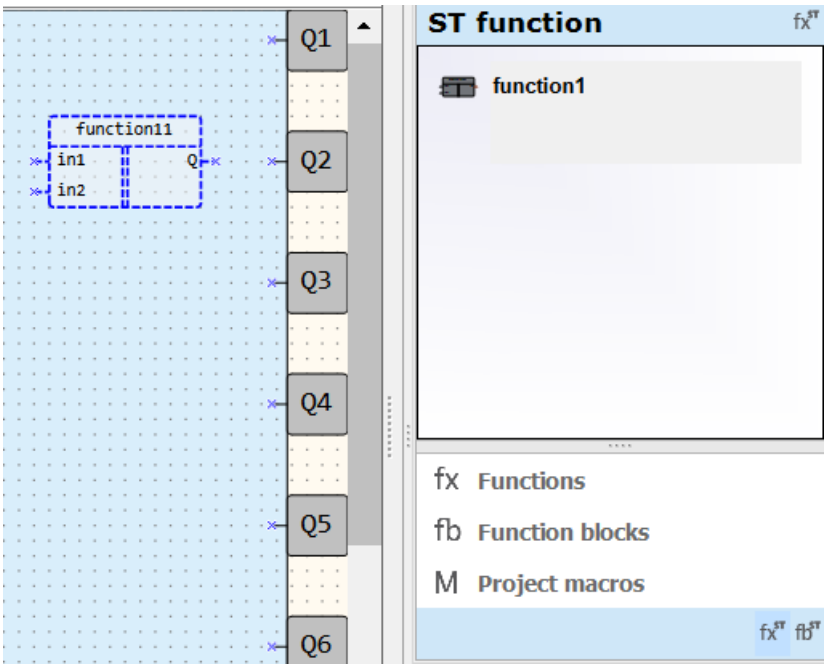
Creation of ST function

1. Click the toolbar icon ***New ST function***. Function editor with an ST function template opens in a new workspace.

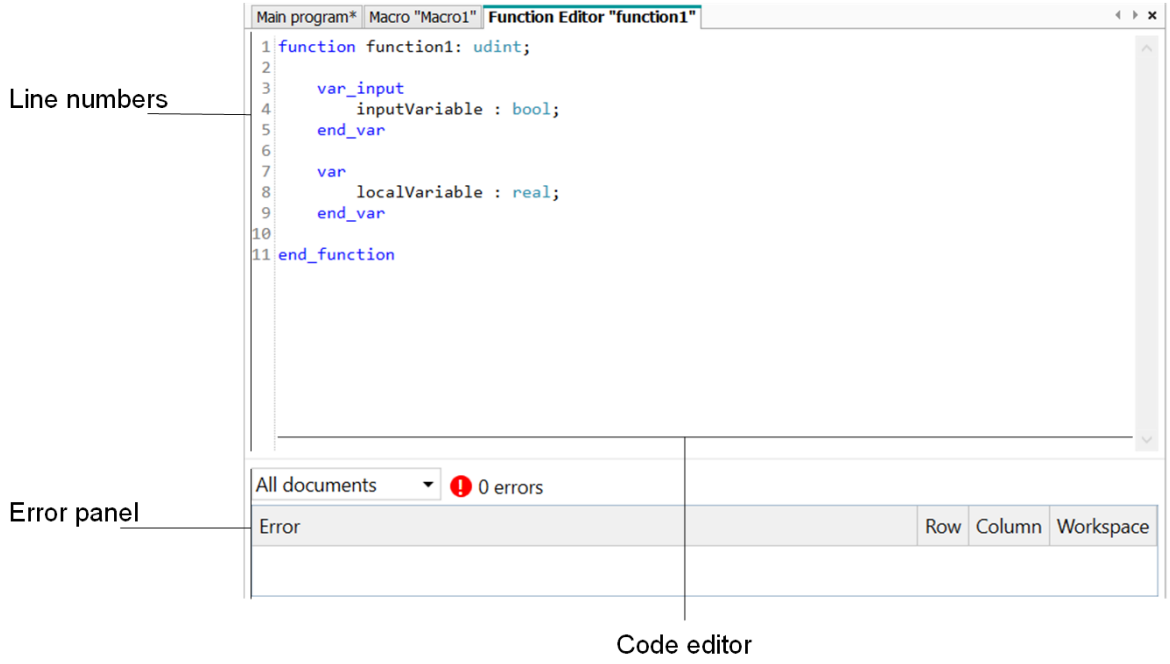
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. Specify the function name and the output data type in the first line.
3. Specify all required inputs variables in the input variable declaration block ***var_input***.
4. Specify all required local variables in the local variable declaration block ***var***.
5. Develop a function algorithm in accordance with the ST syntax rules.
6. Switch to the ***Main program*** tab or close the ***Function editor*** tab. The function will be saved automatically.
7. Select the section ***ST functions*** in ***Library Box*** and drag the saved function onto the project workspace.



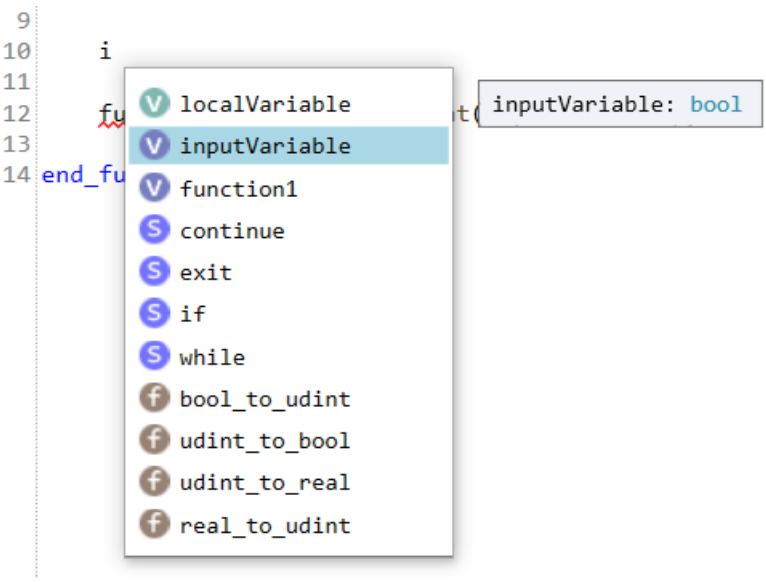
Function editor interface



1. **Line numbers** – sequential line numbers in the program code.
2. **Code editor** – code editing area with automatic syntax highlighting.
3. **Error panel** – error display area.

Snippet management

Snippet management is a text editor feature that allows easy insertion of content from a catalogue of repeatedly used text. If you enter the first character in the editor, a context menu opens with focus on the first line. Use the cursor keys to select a snippet. To insert the selected snippet into the code, press ***Enter*** or ***Tab*** or double-click on the list item.



Snippet groups:
– V – local variables
– S – statements (***while***, ***for*** etc.)
– K – keywords (***true***, ***false***)
– F – built-in functions

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

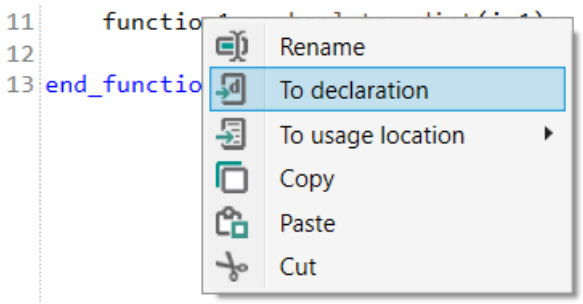
– T – other functions

Within groups, snippets are arranged alphabetically.

Jump to declaration or usage location

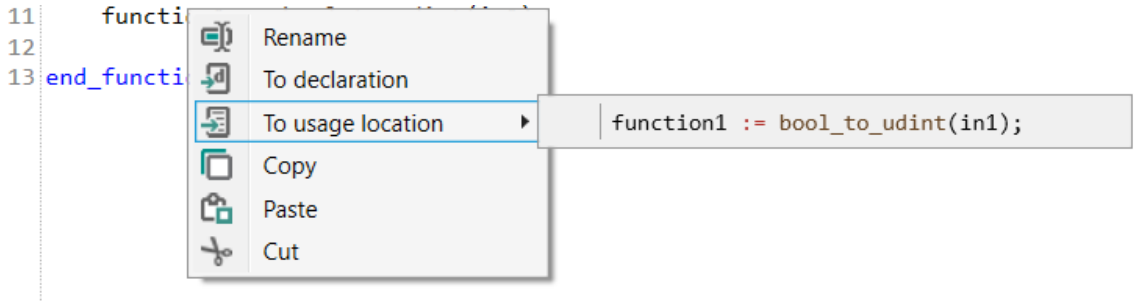
For convenient work with the code, a search is implemented to find the places in the program code where a function or variable is declared or used.
To jump to declaration:

1. Place the cursor on the name of a function or variable in the program code.
2. Right-click on the name.
3. Select ***To declaration*** in the context menu.



To jump to usage location:

1. Place the cursor on the name of a function or variable in the program code.
2. Right-click on the name.
3. Select ***To usage location*** in the context menu. A list of places in the code opens the selected function or variable is used.



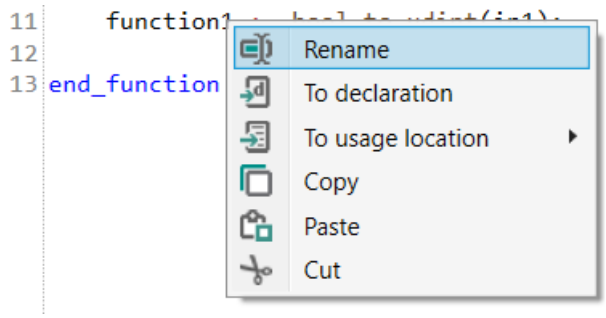
4. Left-click on the selected usage location. The cursor will move to the line where the function or variable is used.

Rename variable or function

Centralized change of the name of a variable or function throughout the code is available.
Proceed as follows:

1. Place the cursor on the name of a function or variable in the program code.
2. Right-click on the name.
3. Select ***Rename*** in the context menu. There is a list of places in the code, where the selected function or variable is used.

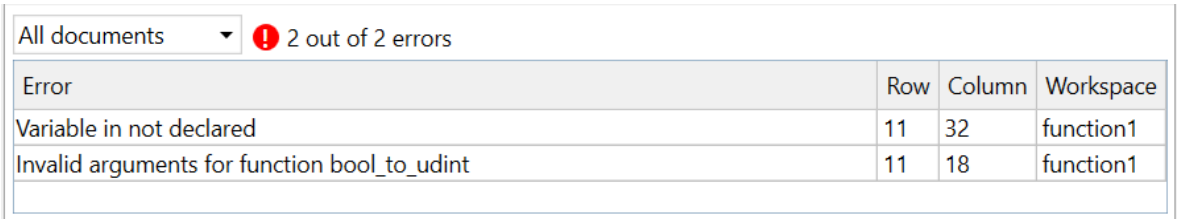
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



4. Enter a new name in one of the green marked locations and click on another place in the code. Now the name is changed in the whole program.

Error panel

All errors occurred during the code writing are listed In ***Error panel***. Left-click on a row in the list to jump to the error in the code.

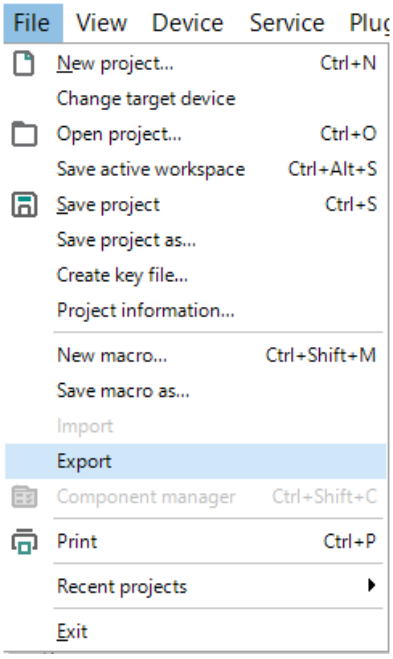


Error	Row	Column	Workspace
Variable in not declared	11	32	function1
Invalid arguments for function bool_to_udint	11	18	function1

Export ST function

Exporting a function to a file is only possible when the function editor tab is open. To export a function, select **File** → **Export** in the main menu.
To export function proceed as follows:

1. Open the function in the editor.
2. Select **File** → **Export** from the main menu.



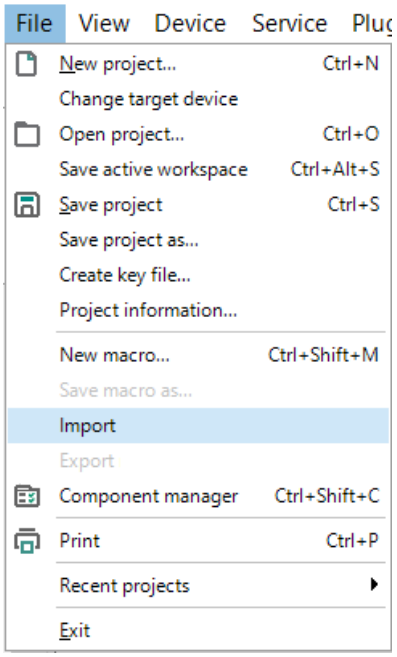
3. In the window that opens, select a location and save the function file with the extension **.fst*. Once saved, a message indicating that the function was exported successfully will be displayed.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Import ST function

If you need to use a function created in another project to create a program, you can import it into the project.
To import a function block, select **File** → **Import** in the main menu.

NOTE
The **Import** item is active only when the focus is on the project workspace.



In the window that opens, select the desired file and click the ***OK*** button. The function will be added to the ***Library Box*** in the ***ST functions*** section, now it can be used in the project.

4.13 ST function blocks

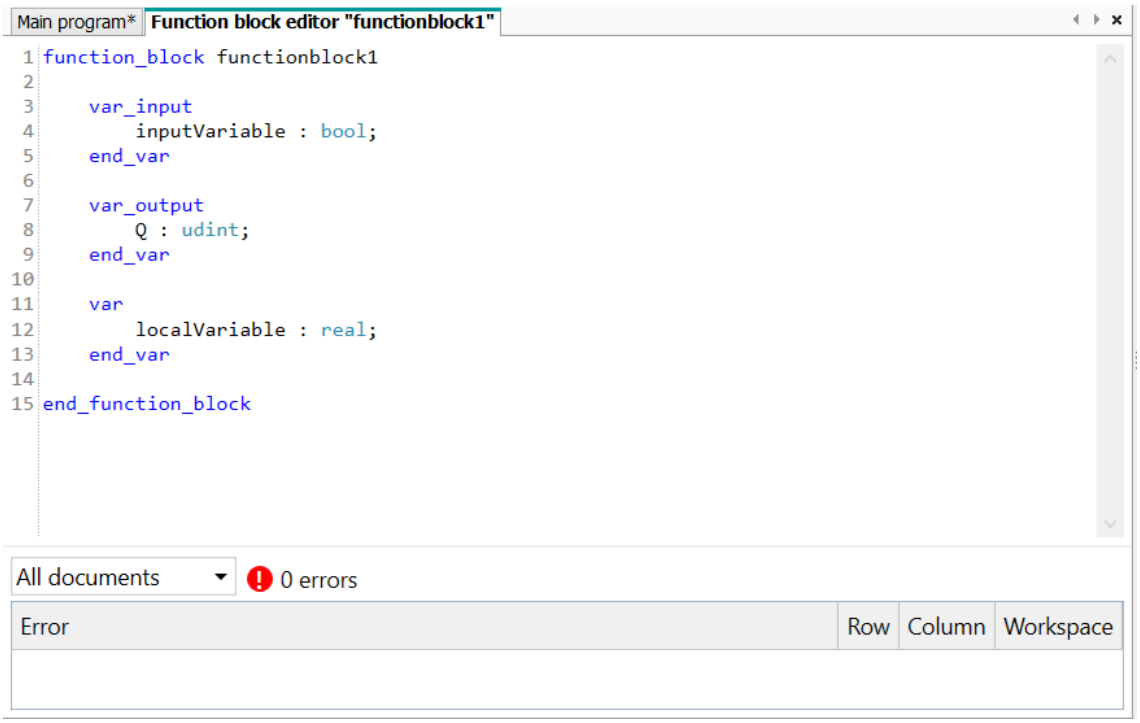
In ALP, it is possible to create user-defined function blocks in the ST language.

Creating an ST function block

To create an ST function block:

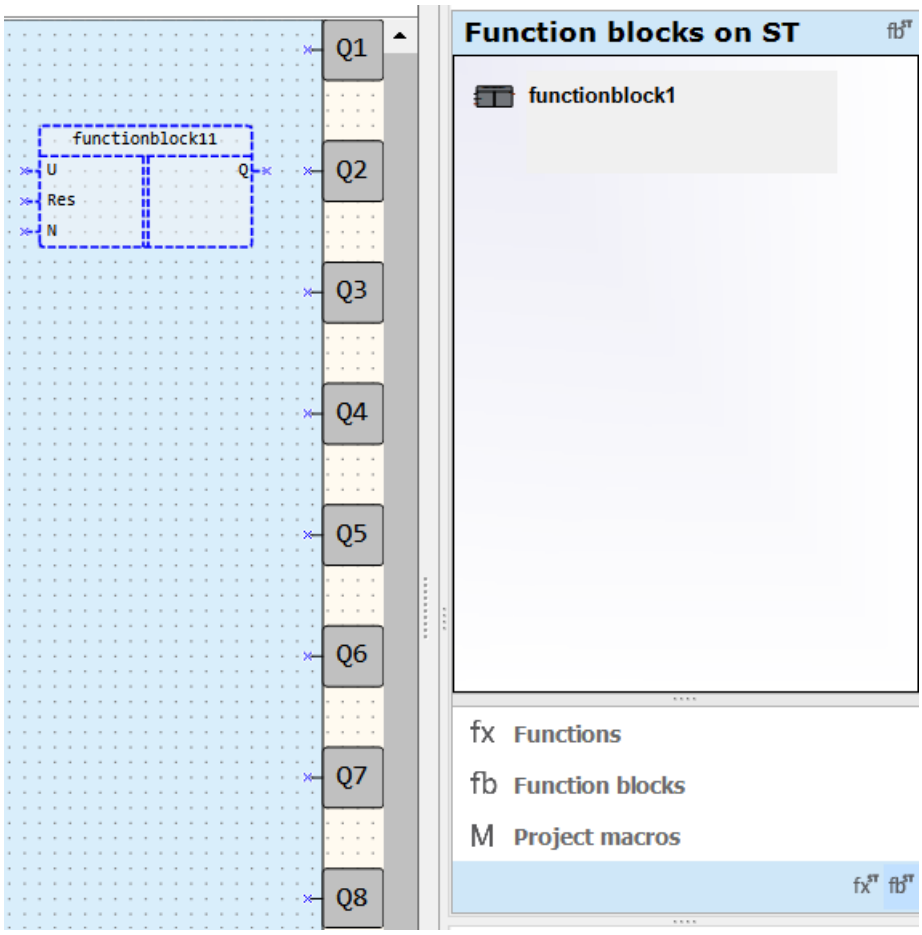
1. Click the toolbar icon  ***New ST function block***. Function block editor with an ST function block template opens in a new workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. Specify the function block name and the output data type in the first line.
3. Specify all required inputs variables in the input variable declaration block ***var_input***.
4. Specify all required local variables in the local variable declaration block ***var***.
5. Develop a function block algorithm in accordance with the ST syntax rules.
6. Switch to the ***Main program*** tab or close the ***Function block editor*** tab. The function block will be saved automatically.
7. Select the section ***ST function blocks*** in ***Library Box*** and drag the saved block onto the project workspace.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Function block editor interface



1. **Line numbers** – sequential line numbers in the program code.
2. **Code editor** – code editing area with automatic syntax highlighting.
3. **Error panel** – error display area.

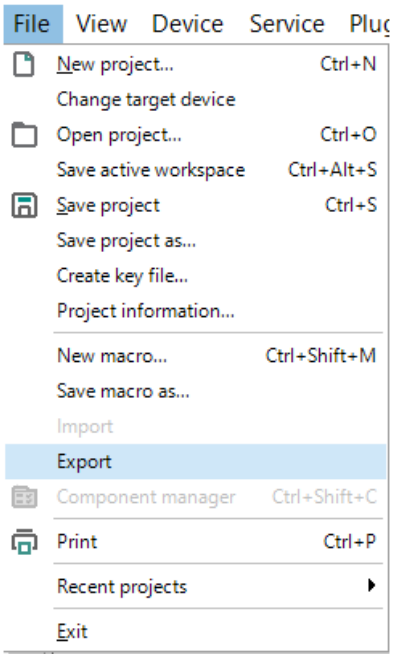
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
Similar to the function editor, the function block editor supports the functions *snippets* , *tracking the location of declaration and use* , *renaming* and *error tracking* .

Export ST function block

Exporting a function block to a file is only possible when the function block editor tab is open. To export a function, select **File** → **Export** in the main menu.
To export function block proceed as follows:

1. Open the function block in the editor.
2. Select **File** → **Export** from the main menu.



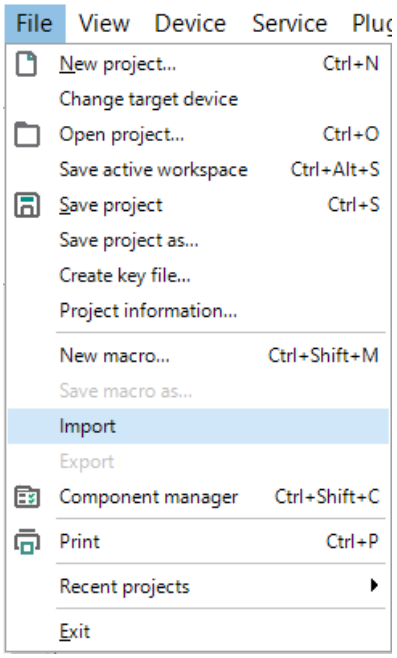
3. In the window that opens, select a location and save the function file with the extension **.fbst*. Once saved, a message indicating that the function was exported successfully will be displayed.

Import ST function block

If you need to use a function block created in another project to create a program, you can import it into the project.
To import a function block , select **File** → **Import** in the main menu.

NOTE
The **Import** item is active only when the focus is on the project workspace.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

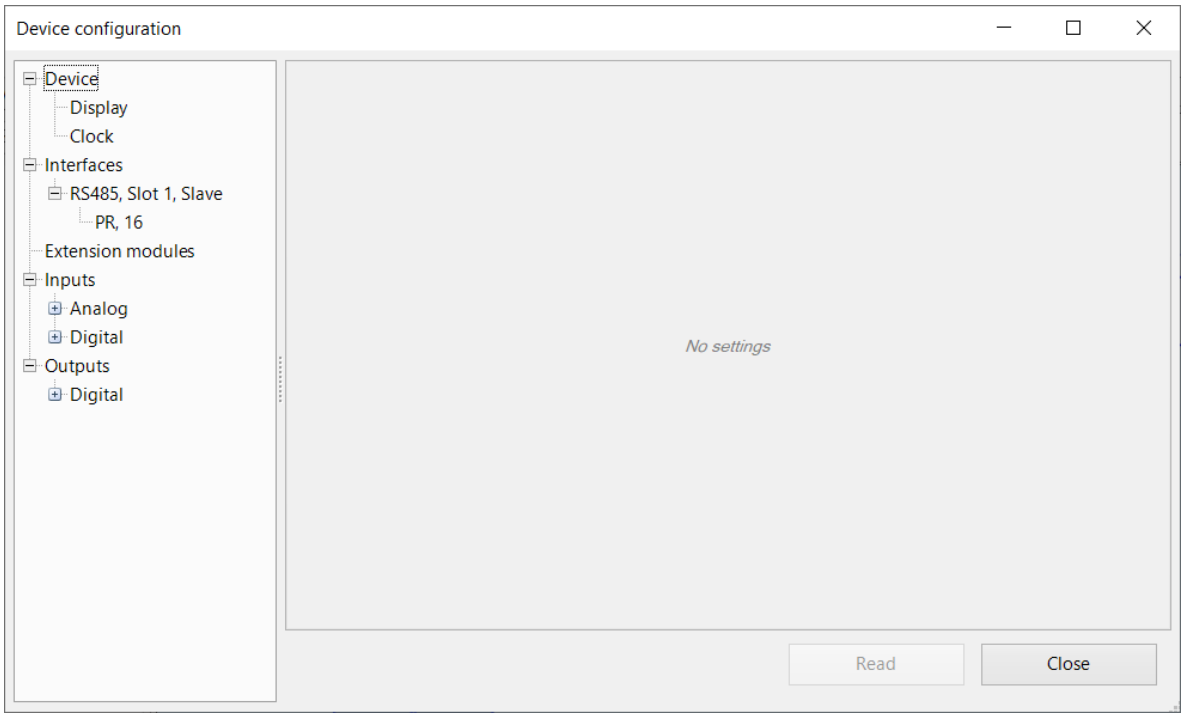


In the window that opens, select the desired file and click the ***OK*** button. The function block will be added to the ***Library Box*** in the ***ST function blocks*** section, now it can be used in the project.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5 Device configuration

The configuration of the device is a part of a project and can be set using the menu item ***Device→ Configuration***. The dialog window ***Device configuration*** consists of two parts. The configurable parameters of the device are presented in the parameter tree in the left part of the window. The content of a group is presented in the right part.



The content of the parameter tree depends on the target device and may include the following groups:

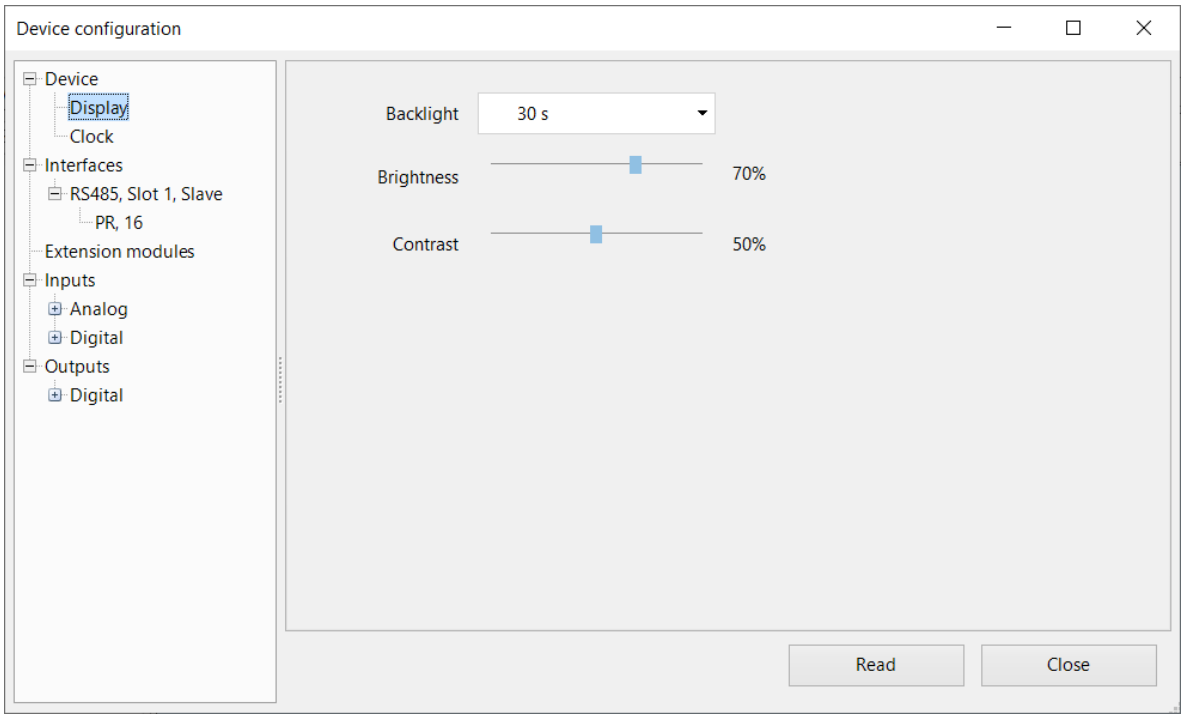
– *Display*
– *Clock*
– *Interfaces*
– *Extension modules*
– *Inputs and outputs*

All the settings are saved in the project, except the clock settings. The configuration is also possible without connecting the device.

5.1 Display

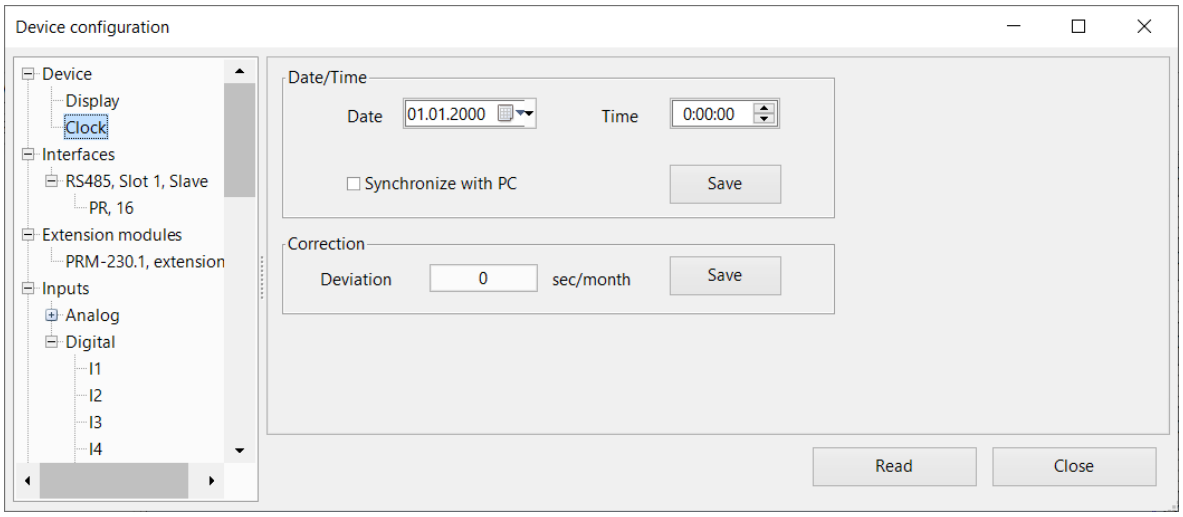
If the target device has a display, the following parameters can be set:
Backlight – the duration of the backlight since the last user activity
Brightness – 0…100%
Contrast – 0…100%
The button ***Read*** can be used to read out the current display settings from the connected device.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



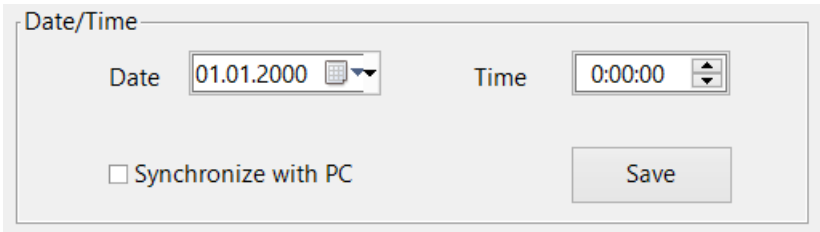
5.2 Clock

If the target device has a built-in real-time clock, the date and time can be set in the ***Clock*** group.



Date and time

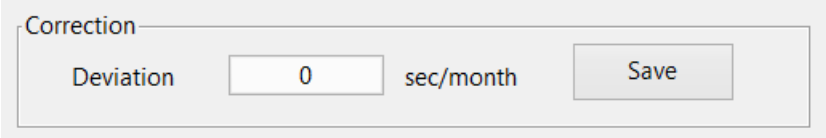
To synchronize the device clock with the PC clock, check the checkbox ***Synchronize with PC***. In this case the fields ***Date*** and ***Time*** become inactive. To set the device clock to the new values, click the button ***Save*** in the section ***Date/Time***.



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Correction

Specify the clock error in seconds per month in the field ***Deviation*** to set the clock correction. Enter a negative value if the device clock is too fast.



To save the clock correction in the device, click the button ***Save*** in the section ***Correction***.
The button ***Read*** can be used to read the current time settings from the connected device.

Clock configuration for the new hardware devices

The clock settings window for devices on the new hardware platform has a different interface and does not have time correction (their hardware provides greater accuracy). Setting the time zone is required to display local time correctly, since the device stores the time value as Greenwich Mean Time (GMT). Enabling the **Set computer time zone** option synchronizes the real world clock device time with PC clock.

5.3 Data exchange

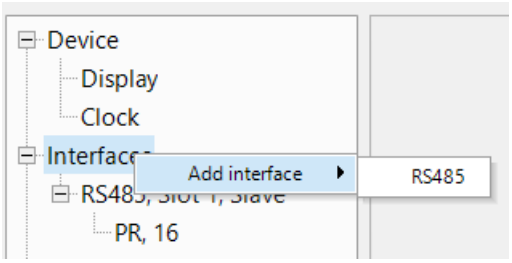
– *Interfaces*
– *Modbus*

5.3.1 Interfaces

In **Interfaces**, the parameters of interfaces for device communication with other devices on the network are configured.
The following interfaces are available:
– *RS485*
– *Ethernet*
– *LoRa*
Communication interfaces RS485 and Ethernet are used with the *Modbus* protocol.
Data communication via LoRa is performed using the LoRaWAN protocol.
Adding, replacing, and removing an interface are available for first-generation devices. In second-generation devices, all available interfaces are located in the configuration tree.

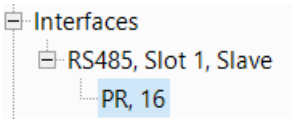
Add interface

If the Add interface function is available in the device, right-click on **Interfaces** in the configuration tree and select **Add Interface**.



An interface of the selected type with default settings is added. The interface name consists of the number of the slot occupied in the device and the interface mode (master/slave).
The LoRa interface in PR102 and PR200 can be located only in slot 2.

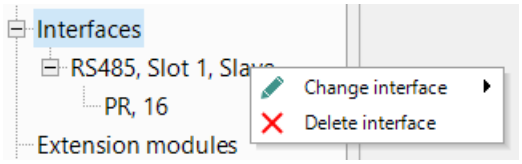
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The number of communication interfaces for the device cannot exceed the number of slots in the device.

Replace/remove interface

Depending on device, the interface can be replaced by another type of interface or removed using the context menu.



If device variables of the interface are used in the project, the interface will not be replaced or removed until you delete the variable references.

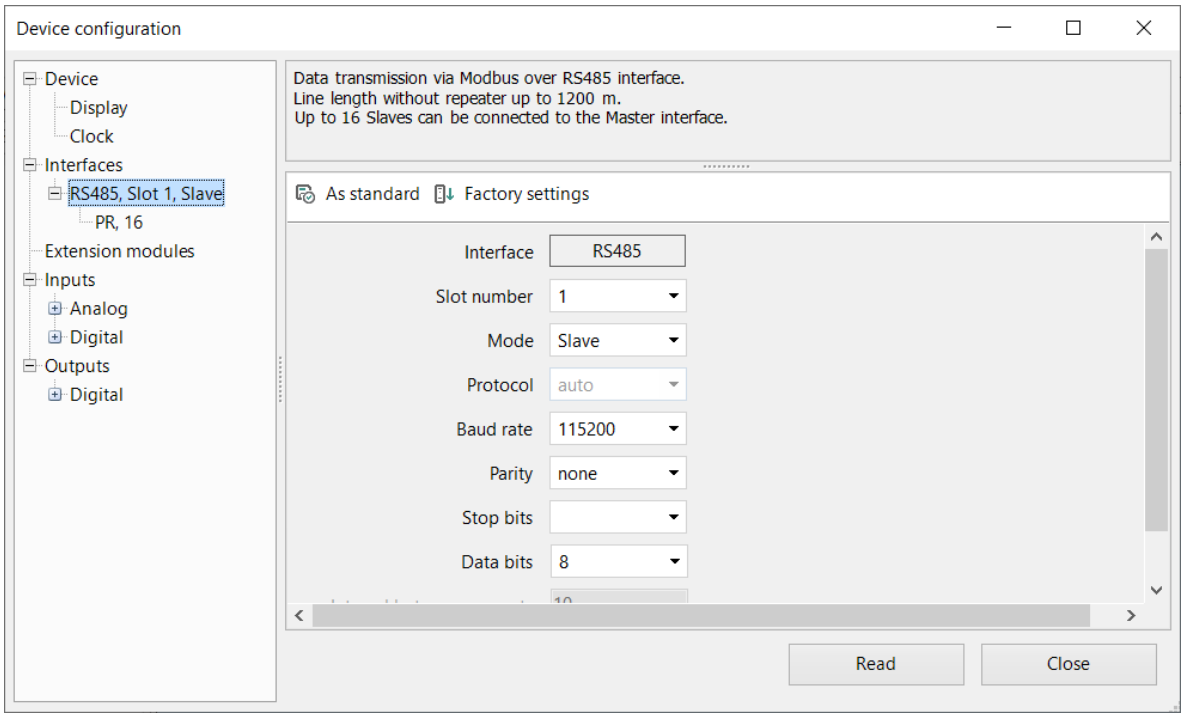
NOTE
Adding, replacing, and removing interfaces is not available for PR100.

5.3.1.1 RS485 interface configuration

To change the interface settings, select it in the configuration tree.

Interface parameters for the first-generation devices

The interface settings will be displayed in the right part of the device configuration window.



The type of the interface (RS485), the number of the assigned slot and the mode (master/slave) are displayed in the tree. If there are LoRa settings in the configuration tree, you cannot select a slot for the RS485 interface.
Interface **Mode**:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– *Slave* – In Slave mode, the device provides values to be read by another device on the network. It does not initiate polling itself.
– *Master* – In Master mode, the device reads and writes the values from other slave devices connected via the interface.

Icon ***As standard*** is used to save the settings as default values for future projects.

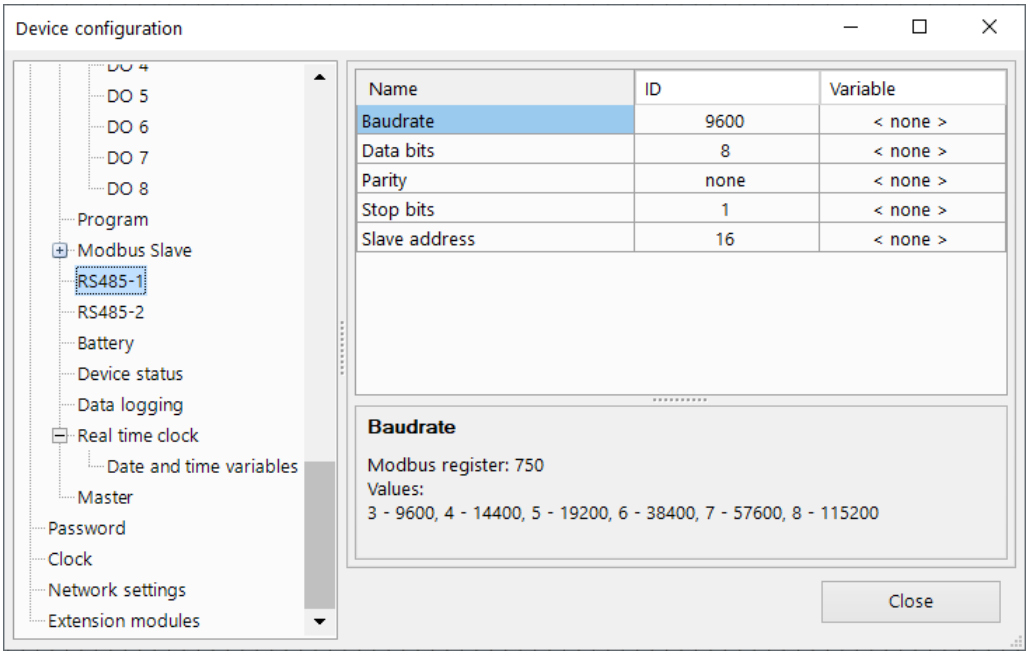
Icon ***Factory settings*** is used for applying the unchangeable factory settings.
The remaining parameters are entered in compliance with the project objectives and the specifications of other devices on the network.

NOTE
Parameters **Protocol** and **Interval between requests** are only available in the master mode.

Button ***Read*** is used to read out the current settings from the connected device.
Use button ***Close*** to save the settings in the project and close the dialog.

Interface parameters for the second-generation devices

For the second-generation devices, the interface parameters are located in the **RS485 port settings** section in the configuration tree. The right side of the device settings window displays interface parameters. The settings window looks like the figure below.



The list of interface parameters depends on the device.
Choose values from the drop-down list or enter them into the **ID** column. It is possible to assign a variable to the parameter in the **Variable** column.

5.3.1.2 Ethernet interface configuration

Ethernet settings are available only for second generation devices in the **Ethernet Settings** menu in the **Network settings** section of the settings tree.
The settings window displays current network parameters of the device, and also sets new ones.
After saving the settings with the new IP address, the device should be rebooted.

NOTE
After setting a new IP address, the device will lose connection with the PC. For the new connection you need to specify a new IP address (see *Connection to device*).

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3.1.3 LoRa interface configuration

LoRa settings are available only for device models with the LoRa module, in the configuration tree: **Interfaces** → **LoRa**. Data exchange via the LoRa interface is carried out using the LoRaWAN protocol. The programmable relay is the end node of the LoRaWAN network.

Network Settings

The LoRa interface and the LoRaWAN connection can be configured on the **Network parameters** tab.

Activation — Select the activation method in the network (ABP/OTAA):

– **OTAA** — network connection with the settings received from the server

– **ABP** — network connection with the settings specified manually

Region — Region-specific frequency band of LoRaWAN.

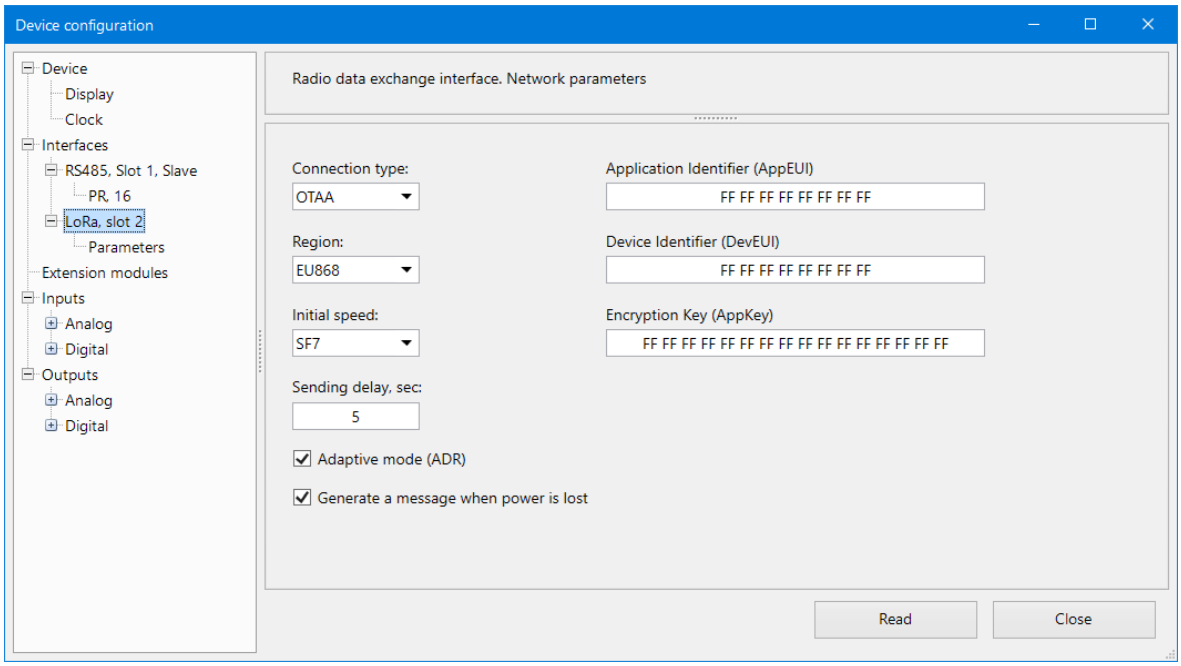
Initial data rate — Spreading Factor (SF) which determines the data rate, transmission range, and noise immunity. SF7: maximum rate, minimum range. SF12: minimum rate, maximum range.

Transmission delay, s — It is the delay time (in seconds) before sending packets. Transmission delay allows you to postpone the transfer of a ready parcel.

Adaptive data rate (ADR) — When selected, it automatically changes **Initial data rate** to optimal rate, depending on communication quality. The specified **Initial data rate** is used as the starting value for optimizing.

Generate a message when power is lost — When the parameter is selected, the device will send a message to the server in case of power supply failure.

OTAA



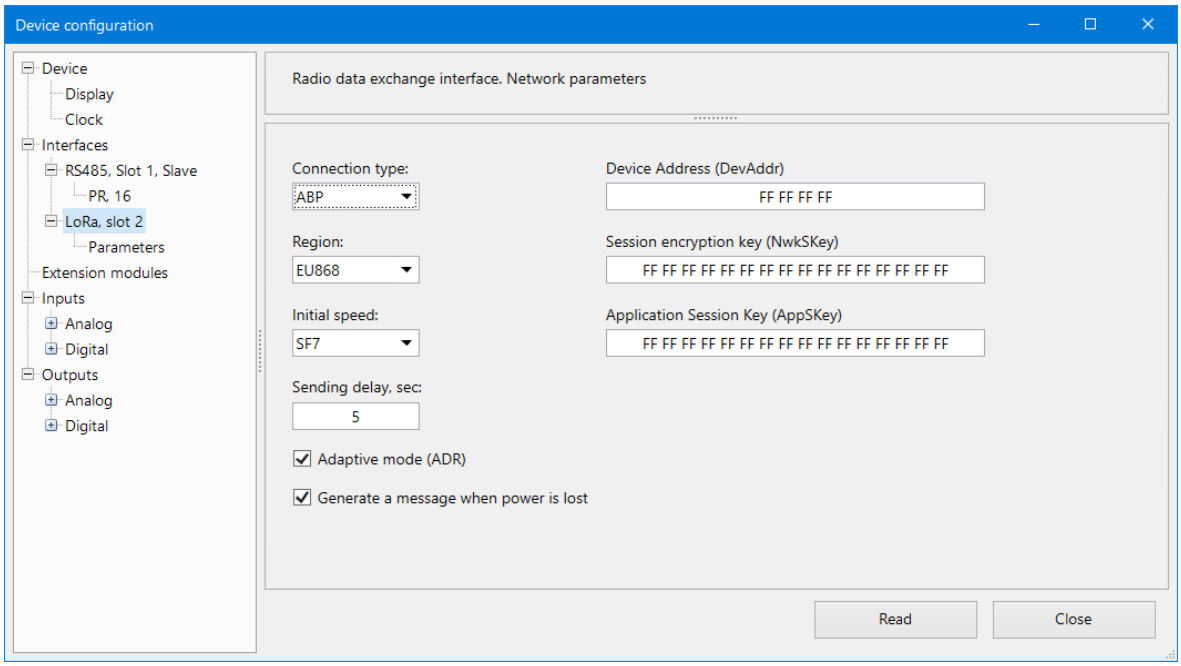
Application Identifier (AppEUI) — Globally unique 64-bit (16-character) identifier that uniquely identifies the application provider of the end-device. The device uses the hexadecimal code to identify the application that receives and processes data from the device.

Device Identifier (DevEUI) — Globally unique 64-bit (16-character) identifier that specifically identifies an end-device. It is primarily used in the network to securely register and activate a connected end-device.

Application key (AppKey) — Server key (128 bits). It is used to encrypt information that will be transmitted over the network. It must be kept confidential and match on the device and on the server side.

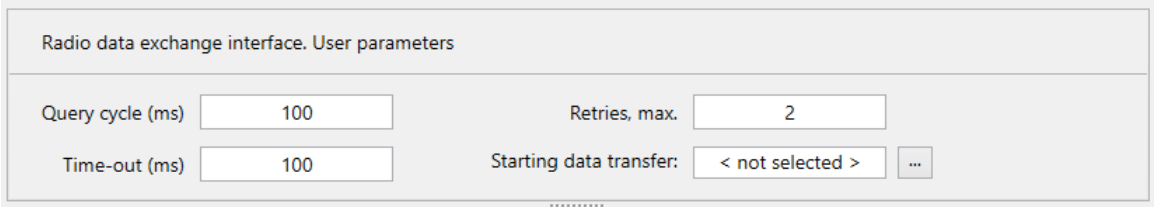
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

ABP



Device Address (DevAddr) — Local network address of a device. It is used for message routing and device identification during data communication.
Network session key (NwkSKey) — Network session key (128 bits) is a hexadecimal code used to encrypt/decrypt data communicated between the device and the network server.
Application session key (AppSKey) — Application session key (128 bits) is a hexadecimal code used to encrypt/decrypt data communicated between the device and the network server during a specific communication session.
Read button reads all the current parameters of the connected device.
Close button saves the entered parameters in the device and closes the Device configuration window. The parameters take effect after the power supply is restarted or after the program is loaded into the device.

User parameters



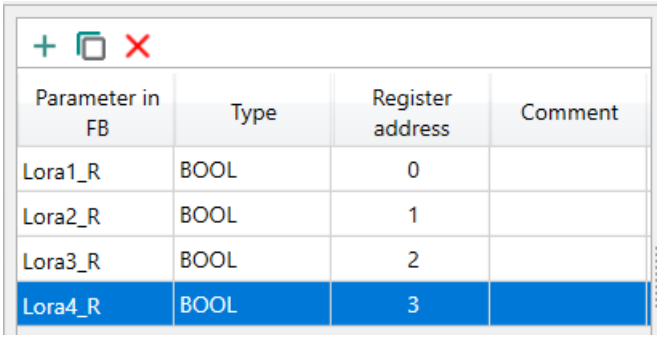
Query cycle (ms) — the time interval between polling cycles.
Time-out (ms) — the time after which a polling attempt is considered failed.
Retries, max. — the number of failed polling attempts.
Starting data transfer — specifies a boolean variable that controls polling access: 0 – polling disabled, 1 – polling enabled. After polling, the value is reset to 0. The variable is selected in the variable table, which is opened by clicking the "..." button.
Group queries are not supported in LoRa modules.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

User-defined variables

A table of device variables is located at the bottom of the window.

To add a variable, click on the button. The variable will be added to the table as shown in the figure below.



Parameter in FB	Type	Register address	Comment
Lora1_R	BOOL	0	
Lora2_R	BOOL	1	
Lora3_R	BOOL	2	
Lora4_R	BOOL	3	

The variables added to the Lora interface automatically get a postfix to the name:

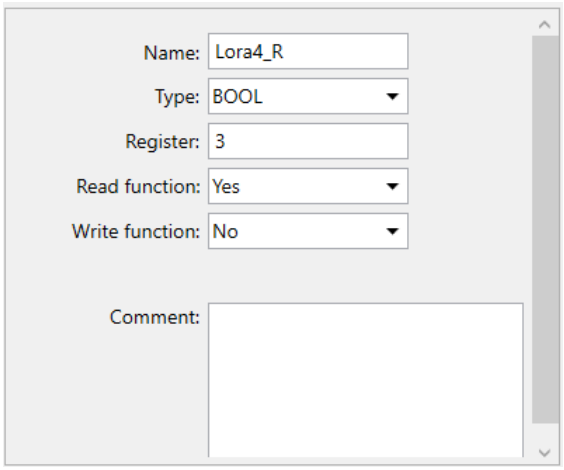
– _R if the read function is selected in the variable properties
– _W if the write function is selected in the variable properties

To create several variables with similar settings, use the Duplicate variable function. For more information about **Duplicate variable**, see section *Slave mode*.

To delete a variable from the table, select the required variable and click .

Device variable properties

The device variable properties are configured to the right of the table.



– **Name** — name of the device variable (set by the user)
– **Type** — type of the device variable: Integer or Floating Point
– **Register** — register address of the device variable (set by the user). The range of available addresses is specified in the User guide on the device.
– **Read function/Write function** — enables or disables the selection of a read/write function
– **Number of registers** (for integer variables only) — number of registers occupied by a variable: 1 or 2
– **Comment** — a text description of a variable in the variable table

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3.2 Modbus working

General

For devices in Master mode, you must select a protocol in *settings*.

Modbus Master	**PR200, PR100, PR102, SMI200 First Generation**	**PR225, PR205, PR103 Second Generation**
Protocols	Modbus RTU, Modbus ASCII (user-selectable)	Modbus RTU, Modbus ASCII (user-selectable) Modbus TCP
Maximum Number of Slave Devices	16 per interface	32 across all interfaces
Maximum Number of Requests	255 variables per device Depending on the configuration, a single variable can correspond to a read request, a write request, or both a read and write request simultaneously.	192, starting from ALP version 2.7.349 and firmware versions: PR205: 1.5.0, PR103: 1.7.0 64 (in earlier software versions). Depending on the configuration, a single variable can correspond to either a read request or a write request.
Group Queries	Supported, starting from ALP version 2.11 and firmware versions: PR100: 3.08, PR100: 2.70, PR102: 2.70, PR200: 2.73, IPP120: 2.73. Up to 48 registers per request (configurable).	Supported, starting from ALP version 2.10 and firmware versions: PR225: 1.9.0, PR205: 1.9.0, PR103: 1.11.0. Up to 16 registers per request (configurable).
Modbus Functions	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x0F, 0x10	0x01, 0x02, 0x03, 0x04, 0x06, 0x0F, 0x10
Data Types	BOOL, UINT16, UINT32, REAL	BOOL, UINT16, UINT32, REAL
Byte/Register Order Swapping	Supported, affects only REAL and UINT32 variables.	Supported, affects only REAL and UINT32 variables.
Broadcast Requests	Not supported	Not supported
Polling Templates	Supported	Supported starting from ALP version 2.10
Polling Control	At the device level (enable/ disable polling) – see the **Polling** parameter. At the request level – triggering on the edge of a boolean variable (parameters **Read Trigger**/**Write Trigger**) + the **Write on Change** checkbox.	At the device level (enable/ disable polling) – see the **Polling** parameter. At the request level – triggering on the edge of a boolean variable (parameters **Read Trigger**/**Write Trigger**) + the **Write on Change** checkbox.
Management of Desired Polling Period	At the device level	At the request level

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Modbus Master	PR200, PR100, PR102, SMI200 First Generation	PR225, PR205, PR103 Second Generation
Diagnostics	At the device level (communication established/ lost) – see the **Status** parameter. At the request level – Modbus error code - see the **Status** parameter.	At the device level (communication established/ lost) – see the **Status** parameter.
Modbus TCP	Not supported	Supported Polling of all slave devices is sequential (only one client connection is active at a time). Requests are sent sequentially within a connection. Valid range for the slave device address (Unit ID): 1…247.

Devices in Slave mode automatically determine the protocol.

Modbus Slave	PR200, PR100, PR102, SMI200 First Generation	PR225, PR205, PR103 Second Generation
Protocols	Modbus RTU, Modbus ASCII (auto-detection – response is sent in the format of the request)	Modbus RTU, Modbus ASCII (auto-detection – response is sent in the format of the request) Modbus TCP
Memory Area Model	Common, with input/holding register overlay	Common, with input/holding register overlay
Memory Size	64 registers (128 bytes)	1020 registers (2040 bytes)
Group Queries	Supported without artificial limitations	
Modbus Functions	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x0F, 0x10	0x03, 0x04, 0x06, 0x0F, 0x10, 0x14, 0x15
Working with Bit Masks	Bit mask parameters can be read using either function 0x03 or 0x01. If using 0x01, the register number should be multiplied by 16 and the bit number added.	Bit mask parameters can only be read using function 0x03.
Data Types	Uint16, REAL	Uint16, REAL
Byte/Register Order Swapping	Supported, but affects only REAL and UINT32 system variables.	Not supported. When polling variables that occupy two or more registers, the following settings should be used: – Byte order – big-endian (most significant byte first) – Register order – little-endian (least significant register first)
Broadcast Requests	Supported	Not supported
Modbus TCP	Not supported	Supported, up to 4 simultaneous client connections (a fifth connection

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Modbus Slave	PR200, PR100, PR102, SMI200 First Generation	PR225, PR205, PR103 Second Generation
		is independent and reserved for akYtec Cloud). The Unit ID in the request is not validated.
RETAIN	All network variables are always non-volatile (retained).	Starting from ALP version 2.9, the non-volatility of each network variable is user-configurable. In earlier versions, all network variables are always non-volatile (retained).
Register Map Export	Export to akYtec Cloud, and a *.csv* file is supported.	Export to akYtec Cloud, and a *.csv* file is supported.

To organize data exchange on a network via a communication interface, a Network Master is required – a device operating in Master mode. The main function of the Network Master is to initiate data exchange.
Only one device in Master mode is allowed on the communication line.

Number of Communication Interfaces

The number of communication interfaces a device has depends on the number of available connection slots. If the device supports up to two communication interfaces, they can be configured to operate on two independent networks.

Standard Modbus Exception Codes

- **01** – The function code received in the query is not an allowable action for the server (Illegal Function).
- **02** – The data address received in the query is not an allowable address for the server (Illegal Data Address).
- **03** – A value contained in the query data field is not an allowable value for the server (Illegal Data Value).

Functions and Data Areas

When making a request, the Master accesses one of the Slave's **memory areas** using a **function**. A **memory area** is characterized by the type of values it stores (bits/registers) and the access type (read-only/read-write).

Table 5.1 Modbus Protocol Data Areas

Data Area	Designation	Data Type	Access Type
Coils	0x	Boolean	Read/Write
Digital Inputs	1x	Boolean	Read Only
Input Registers	3x	Integer	Read Only
Holding Registers	4x	Integer	Read/Write

Each data area consists of a specific number of cells, which depends on the particular device. Each cell has a unique address. For configurable devices, the manufacturer provides a **register map** containing information about the correspondence between device parameters and their addresses. For programmable devices, the user creates such a map independently using a programming environment. Some devices combine both of the cases described – their register map has a fixed part and a part that the user can supplement according to their task.
In some devices, memory areas are overlaid on top of each other (e.g., **0x** and **4x**), meaning the user can access the same registers using different functions.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

A **function** defines the operation (read/write) and the memory area on which this operation will be performed.

Table 5.2 Main Modbus Protocol Functions

Function Code	Function Name	Command Executed
1 (0x01)	Read Coil Status	Reads values from multiple coils
2 (0x02)	Read Discrete Inputs	Reads values from multiple discrete inputs
3 (0x03)	Read Holding Registers	Reads values from multiple holding registers
4 (0x04)	Read Input Registers	Reads values from multiple input registers
5 (0x05)	Force Single Coil	Writes a value to a single coil
6 (0x06)	Preset Single Register	Writes a value to a single holding register
15 (0x0F)	Force Multiple Coils	Writes values to multiple coils
16 (0x10)	Preset Multiple Registers	Writes values to multiple holding registers

In various documents, identical designations may have different meanings depending on the context. For example, the prefix **0x** is often used to indicate a hexadecimal number system. Therefore, in one case, **0x30** might denote "the 30th bit of the coils memory area," and in another, it might mean "address 30 in the hexadecimal (HEX) number system" (and this address could refer to any memory area).
Polling a Slave can be **single** or **group**. In **single polling**, the Master reads each of the Slave's parameters using a separate command.
In **group queries**, the Master reads several parameters at once with a single command. This is possible only when the addresses of the parameters in the register map are strictly consecutive with no gaps. Group queries help reduce network traffic and the time spent polling a device, but in some cases, it cannot be used (or can only be used with limitations) due to the individual characteristics of the device.

Query Cycle and Response Time-out

The device sends requests at intervals specified by the **Query cycle** parameter of the device being polled. The time allowed to wait for a response to a request is set by the **Time-out** parameter. If no response is received within this time, the device will retry the request N times, where N is the number of **Retries**. If there is still no response, the **Status variable** will be set to 0 (False).

Network Variable Types

Data Type		Size in Slave Mode	Size in Master Mode
Integer	UINT	16 bits (1 register)	16 bits (1 register)
	UDINT	-	32 bits (2 registers)
Floating Point	REAL	32 bits (2 registers)	32 bits (2 registers)
Boolean	BOOL	-	1 bit

Register Order and Byte Order

When the device operates in Slave mode, the register/byte order settings only affect how the Master reads floating-point system network variables (analog inputs/outputs).
For user-defined network variables in Slave mode, data is always transmitted in the following format:

– Most significant register first: No

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

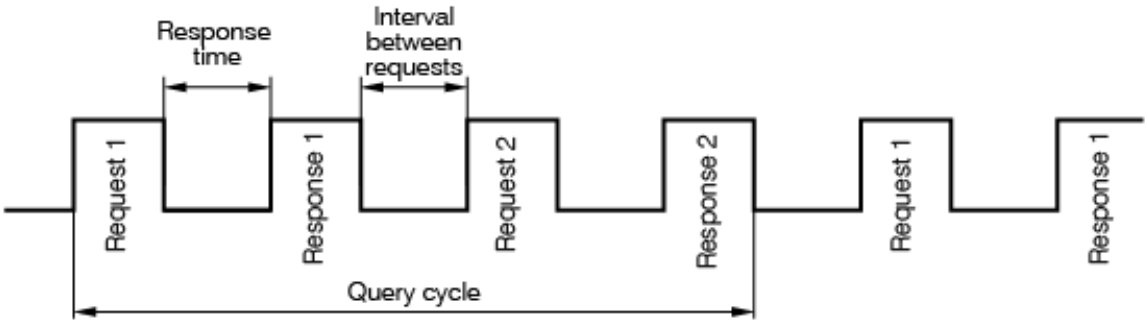
– Most significant byte first: Yes

Automatic Program Cycle Time Adjustment

The devices can automatically adjust the program cycle time based on the complexity of the algorithm. This automatic adjustment of the program cycle time affects the operation of the communication interface because requests are processed during the remaining time of the cycle after the program has finished executing.
According to the cycle time adjustment algorithm, the minimum number of Master calls is up to 50 times per second. If the Master cannot poll all devices within this timeframe, the algorithm should be modified to optimize the number of requests.

Query Cycle and Assignment Rules

The device places all requests into a queue. If the queue is short, the device will complete all request-response cycles and then wait until the specified period elapses. If the queue is long and cannot be processed within the specified period, the device will poll all necessary parameters with the maximum possible period, which will be longer than the period set in the configuration.
Polling timing diagram:



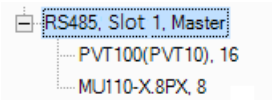
For a maximum speed of 115200 bps, when polling two REAL variables from a single slave device under "ideal" conditions (short communication line, no interference), the following should be established:

– Response time from the start of the request: 2.64 ms
– The next request starts no earlier than after 12 ms
– Total time for one REAL register request: 3.4 ms

Under the conditions mentioned above, the device can send 83 requests per second. This value also applies to other devices with similar timing characteristics. As the algorithm becomes more complex during development, the cycle time increases, and the number of requests per second will decrease.
The polling period value depends on the algorithm and how often specific parameters need to be polled. It is recommended to set the polling period to 1 second. In this case, the device will be able to poll up to 50 variables.

Polling of multiple devices in the network

Slaves are polled according to the generated queue from the smallest to the largest address. In the following example, the slave with the address 8 is polled first, while the one with address 32 is polled last.



Query cycle can be set for each slave individually.

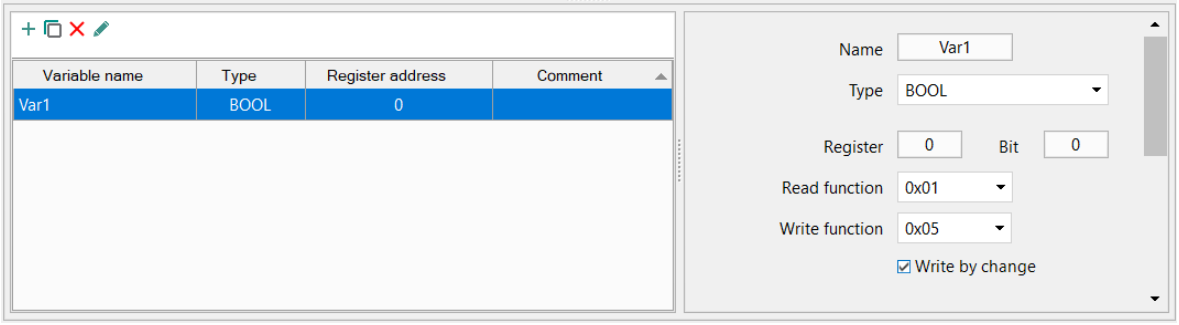
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Calculating the Address and Bit for Reading a Boolean Variable in Master Mode

In some cases, it is necessary to calculate the register address and bit number of a slave device. To determine the register address and bit number from the documentation of the device being polled, take the location of the bit variable, for example, *bit 1400*. Then proceed as follows:

1. Calculate the register address: 1400 ÷ 16 = 87.
2. Calculate the bit number: 1400 mod 16 = 8.

The obtained address and number are entered into the ***Register*** and ***Bit*** fields when configuring the device polling.



5.3.2.1 Master mode

In Master mode, the device polls other slave devices connected via the interface. The **Group queries** option is available for polling similar parameters of the connected slave devices.
To poll, devices must be added and configured. To add a device, right-click on the name of the Master interface in the device configuration window - for *first-generation* devices, or on the Modbus Master item for *second-generation* devices, and select **Add Device** from the context menu.

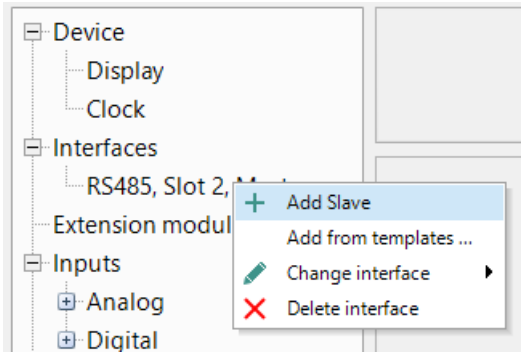


Fig. 5.1 Adding a Slave device (first-generation device)

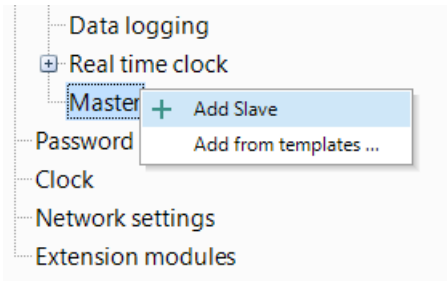


Fig. 5.2 Adding a Slave device (second-generation device)

The added device will appear in the tree as a new branch for the corresponding interface. The device name consists of the name, device address, and communication interface (for second-generation devices).

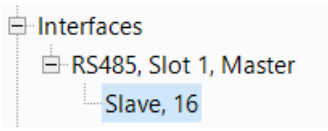


Fig. 5.3 Slave device in the parameter tree (first-generation devices)

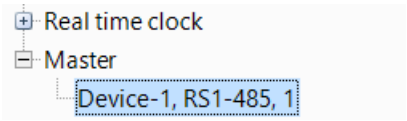


Fig. 5.4 Slave device in the parameter tree (second-generation devices)

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group polling

Group polling allows reading multiple registers with a single command. The maximum number of registers in a group poll depends on the device's buffer size, the Modbus protocol type, and the read/write function used. For *first-generation* devices, the maximum group poll size is 48 registers. For *second-generation* devices, it is 16 registers. The formation of a group poll follows these rules:

1. Variables must be located strictly sequentially in the register map – with no gaps or unused registers/bits between them.
2. The variables in the poll must have identical read/write function codes assigned.

   Exception: Functions **0x05** and **0x06** are not allowed – group requests are not supported for them.
3. Variables are combined into a group poll only if they have the same polling type (periodic, on-command, or on-change).
4. Only variables of compatible sizes can be combined into a group poll:
   - **Boolean variables** – can only be grouped with other Boolean variables.
   - **Integer (1 register)** – can only be grouped with other integer (1 register) variables.
   - **Integer (2 registers)** – can be grouped with integer (2 registers) and floating-point variables.
   - **Floating-point** – can be grouped with integer (2 registers) and floating-point variables.

To minimize the time required to obtain data from all devices, the following is recommended:

- If one or more slave devices are disconnected or unavailable, it is recommended to implement logic in the program to disable polling for those devices or to minimize the " Time-out" parameter for them. Otherwise, the polling time will increase proportionally to the number of disconnected devices and the value set for their ***Time-out*** parameter.
- When setting the "Query cycle" parameter, consider the number of slave devices and the total number of requests. If the processing time for all requests exceeds the time set by the "Query cycle" parameter, this parameter will be ignored.

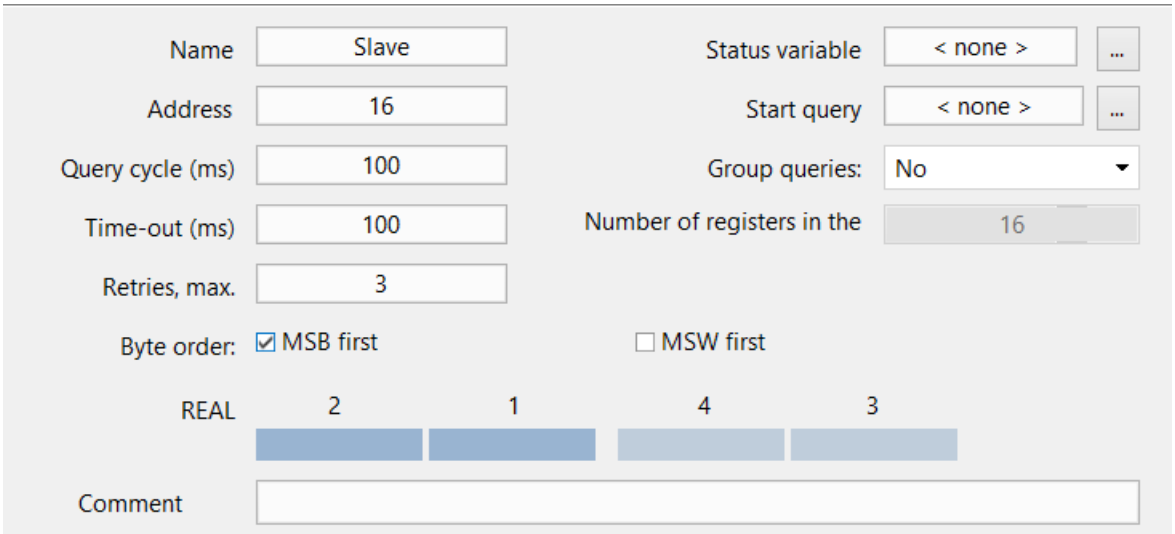
Using identical variable addresses and names for each device is allowed.

Master Mode for First-Generation Devices

NOTE
For a list of first and second-generation devices, see the *Introduction* section.

Slave device parameters

To change device parameters, click on its name in the configuration tree – the parameters will be displayed in the right part of the device configuration window.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **Name** – the device name displayed in the configuration tree.
- **Address** – the network address of the device.

- **Query cycle (ms)** – the time interval between polling cycles. Valid range: 0 to 65535 ms.
- **Time-out (ms)** – the time after which a polling attempt is considered failed. Valid range: 0 to 65535 ms. It is recommended to set the response time-out between 300–500 ms.
- **Retries, max.** – the number of failed polling attempts after which the device's Status changes. Valid range: 0 to 255.
- **Status variable** – specifies a boolean variable that stores the device status: 1 – device is functioning normally, 0 – communication with the device is lost. The variable is selected in the *variable table*, which is opened by clicking the "**...**" button.
- **Start query** — specifies a boolean variable that controls polling access: 0 – polling disabled, 1 – polling enabled. The variable is selected in the *variable table*, which is opened by clicking the "**...**" button.

  NOTE
  Variables linked to the **Status variable** and **Start query** parameters can be used in the program logic.

- **Group queries** – enables polling of more than one variable per request.
- **Number of registers in request** – from 2 to 48.
- **Byte Order** – determines the sequence of bytes in the packet.
- **MSW first** – determines the order of register transmission when working with two-register variables.
- **MSB first** – determines the byte order in the transmission.
- **Comment** – a text description of the device.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

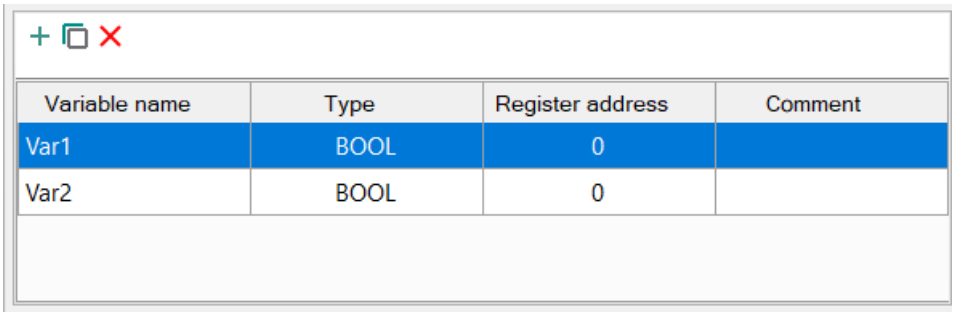
Example:
Goal: Read the number "-250.123".
Depending on the register/byte order settings, it will be transmitted in different ways.

Register/Byte Order Setting	Transmitted Number
Byte order: ☐ MSB first ☐ MSW first REAL 1 2 3 4	Least significant byte 01111101 · Most significant byte 00011111 · Least significant byte 01111010 · Most significant byte 11000011 Least significant word · Most significant word
Byte order: ☑ MSB first ☐ MSW first REAL 2 1 4 3	Most significant byte 00011111 · Least significant byte 01111101 · Most significant byte 11000011 · Least significant byte 01111010 Least significant word · Most significant word
Byte order: ☐ MSB first ☑ MSW first REAL 3 4 1 2	Least significant byte 01111010 · Most significant byte 11000011 · Least significant byte 01111101 · Most significant byte 00011111 Most significant word · Least significant word
Byte order: ☑ MSB first ☑ MSW first REAL 4 3 2 1	Most significant byte 11000011 · Least significant byte 01111010 · Most significant byte 00011111 · Least significant byte 01111101 Most significant word · Least significant word

Slave device variables

The polled device variables table is located at the bottom of the window.

To add a variable, click the + button. The variable will be inserted into the table, as shown in the figure below.



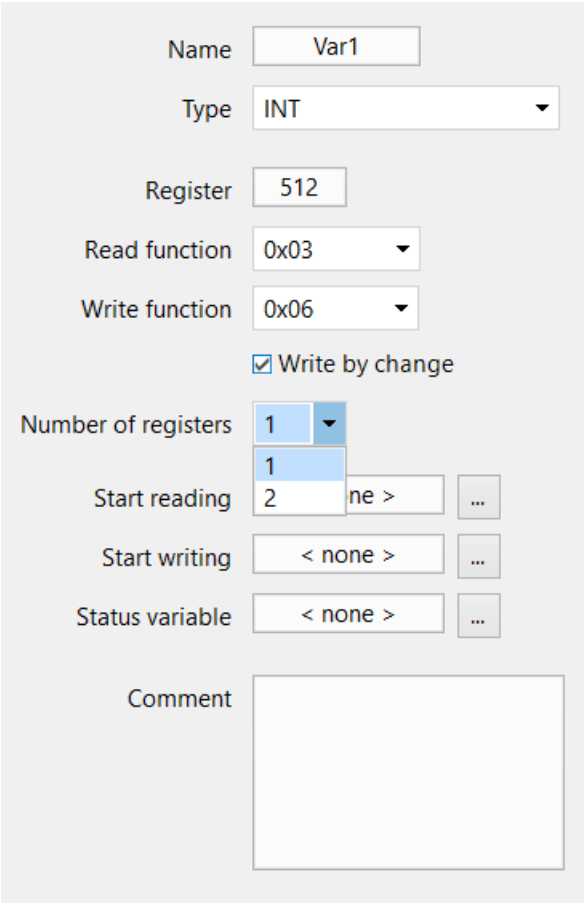
To delete a variable from the table, select it and click the ✕ button.

To create several variables with similar settings, select a variable in the table and click the **Duplicate** button.

NOTE
For more information about **Duplicate variable**, see section *Slave mode*.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Properties of network variables for Slave device



The properties of a polled device variable are configured to the right of the table:

– **Name** – the name displayed in the *variable table*;

> **NOTE**
> Creating variables with identical names is not recommended.

– **Type** – the *variable type*: BOOL, INT, or REAL;
– **Bit (for Boolean variables only)** – the bit number to read;
– **Register/Register Address** – the register value accessed by the device, displayed in the table;
– **Read Function/Write Function** – enables or disables the selection of a read/write function;

> **NOTE**
> **Write by change** – if this function is activated, the master will initiate a write request to the slave device whenever the variable's value changes.

– **Number of registers** – the number of registers occupied by the variable: 1 or 2;
– **Start reading** – assigns a boolean variable (command variable) to force a read of the configured variable. When the command variable changes from 0 to 1, the master initiates a read request. After the request is successfully completed, the command variable value automatically changes from 1 back to 0;
– **Start writing** – assigns a boolean command variable to force a write of the configured variable. When the command variable changes from 0 to 1, the master initiates a write request. After the request is successfully completed, the command variable value automatically changes from 1 back to 0;
– **Status variable** – assigns an integer variable that will store the *Modbus* error code, should one occur;
– **Comment** – a text description of the variable for display in the *variable table*;

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
Command-based reading and writing allow for forced reads or writes of a variable's value, providing more flexible control over these operations.
Different read and write modes can be combined, such as:
– Timer-based reading
– Event-based reading/writing
– Write on change (used by default)

Master Mode for Second-Generation Devices

To configure polling parameters for connected devices, select the **Master** node in the device's parameter tree.

NOTE
The maximum number of devices across all interfaces is 32.

To change the parameters of a polled device, click on its name in the parameter tree. The available parameters will be displayed in the right part of the window: device parameters at the top, and the device's network variables at the bottom.

NOTE
It is possible to modify the network settings of Slave devices from external software (*akYtec ToolPro*).

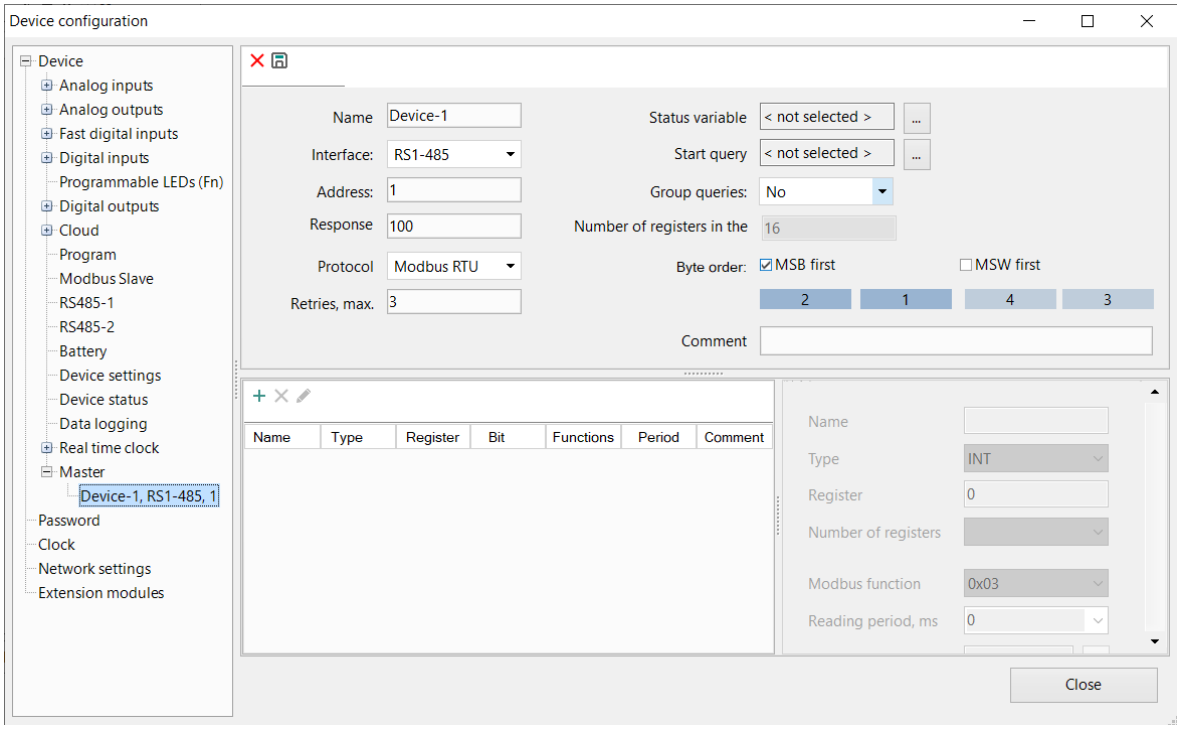


Fig. 5.5 Slave device parameters

– **Name** – the device name displayed in the configuration tree.
– **Interface** – the interface through which the polled device is connected. The list of available parameters depends on the selected interface.
– **Address** – the network address of the device.
– **Response time-out, ms** – the time after which a polling attempt is considered failed. Valid range: 10 to 10,000 ms.
– **Protocol**
– **Retries, max.** – the number of failed polling attempts allowed. Valid range: 0 to 3.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– **Status variable** – specifies a boolean variable that stores the device status: 1 – device is functioning normally, 0 – communication with the device is lost. The variable is selected in the *variable table*, which is opened by clicking the "**…**" button.
– **Start query** – specifies a boolean variable that controls polling access: 0 – polling disabled, 1 – polling enabled. The variable is selected in the *variable table*, which is opened by clicking the "**…**" button.
– **Group queries** – enables polling of more than one variable per request.
– **Number of registers in the request** – from 2 to 16.
– **Byte order** – determines the sequence of bytes in the packet.
– **Comment** – a text description of the device.

Specific parameters for a polled device connected via Ethernet:

– **IP Address** – the unique network address of the device, valid range from 0.0.0.0 to 255.255.255.255.
– **Port** – the port number, valid range from 0 to 65535.

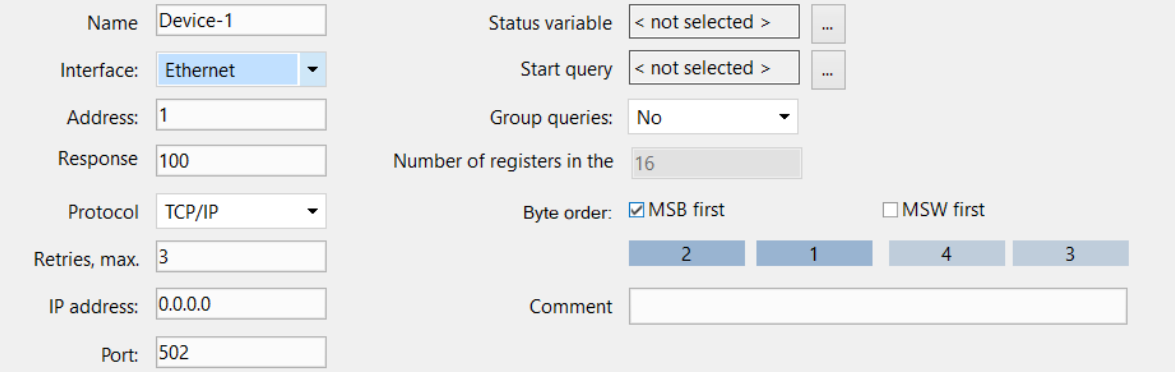


Fig. 5.6 Specific parameters for device connected via Ethernet

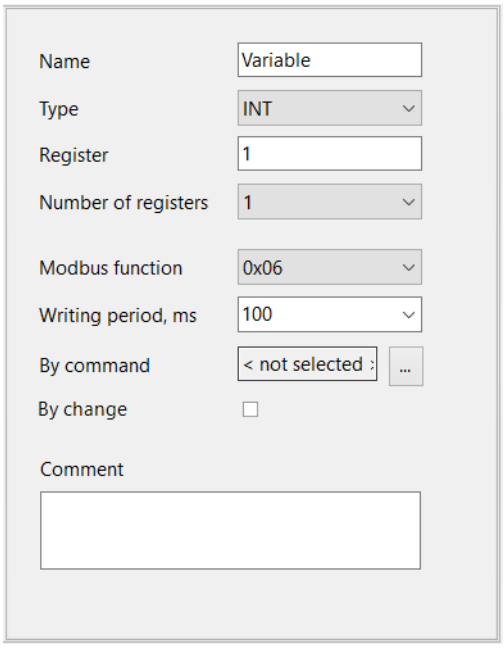


Fig. 5.7 Properties of network variables for the Slave device

– **Name** – the name displayed in the *variable table*.

NOTE
Creating variables with identical names is not recommended.

– **Type** – the *variable type*: BOOL, INT, or REAL.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– **Register** – the register value accessed by the device, displayed in the table.
– **Bit (for Boolean variables only)** – the bit number to read.
– **Number of registers (for Integer variables only)** – the number of registers occupied by the variable: 1 or 2.
– **Modbus function** – enables or disables the selection of a read/write function.

The list of configurable parameters depends on the selected read/write function.
Parameters for Read Functions:

– **Reading period** – the time interval between read requests.
– **By command** – a boolean variable whose edge triggers a read of the parameter.

Parameters for Write Functions:

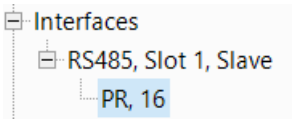
– **Writing period** – the time interval between write operations.
– **By command** – a boolean variable whose edge triggers a write of the parameter.
– **By change** – if this function is activated, the master will initiate a write request to the slave device whenever the variable's value changes.
– **Comment** – a text description of the variable for display in the *variable table*.

5.3.2.2 Slave mode

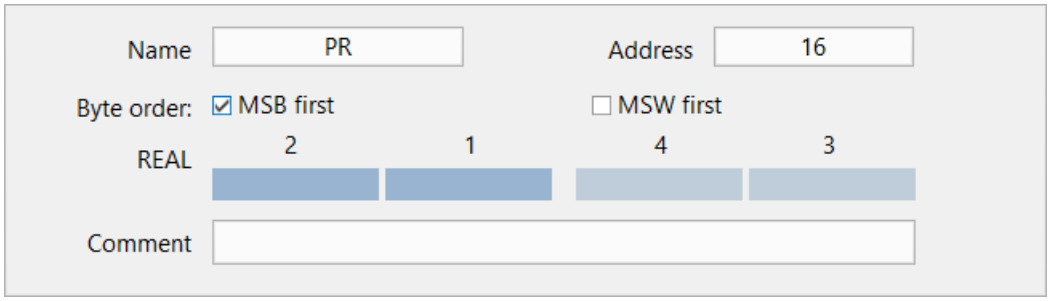
In Slave mode, the device provides data to be read by another device on the network; it does not initiate polling itself.

Slave mode for first generation devices

To configure the data transmission parameters, click on the device name (default is **PR, 16**) under the interface name in the configuration tree.



Device parameters



The right part of the configuration window contains device parameters common to all variables:

– **Name** – the device name displayed in the configuration tree.
– **Address** – the network address of the device.
– **Comment** – a text description of the device.

NOTE
In Slave mode, the **MSW first** and **MSB first** settings only affect the register/byte packing order for the device's system network variables.
For user-defined project network variables in Slave mode, data is always transmitted in the following format:

– **Most significant word first**: No
– **Most significant byte first**: Yes

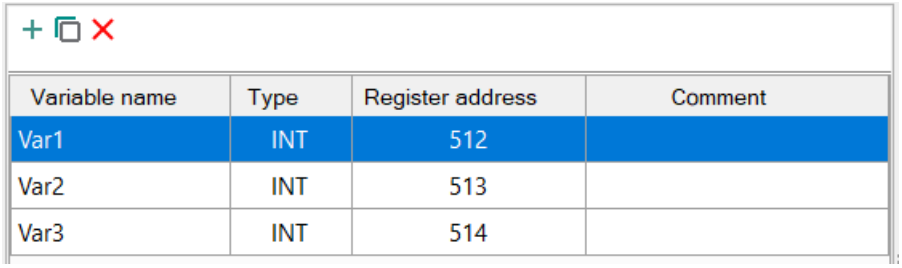
A device in Slave mode can support polling of up to 64 registers during group polling.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Device variables

The device variables table, located at the bottom of the window, contains the variables whose values will be read by another device.

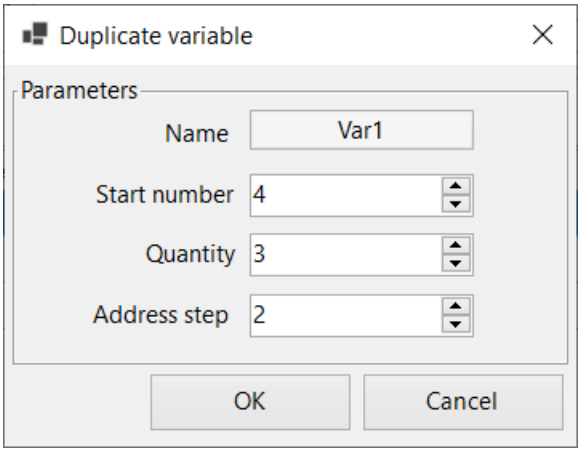
To add a variable, click the + button. The variable will be inserted into the table, as shown in the figure below.



To delete a variable from the table, select it and click the ✕ button.

To create several variables with similar settings, select a variable in the table and click the **Duplicate** button.

NOTE
Duplicating variables is only available for first-generation devices.

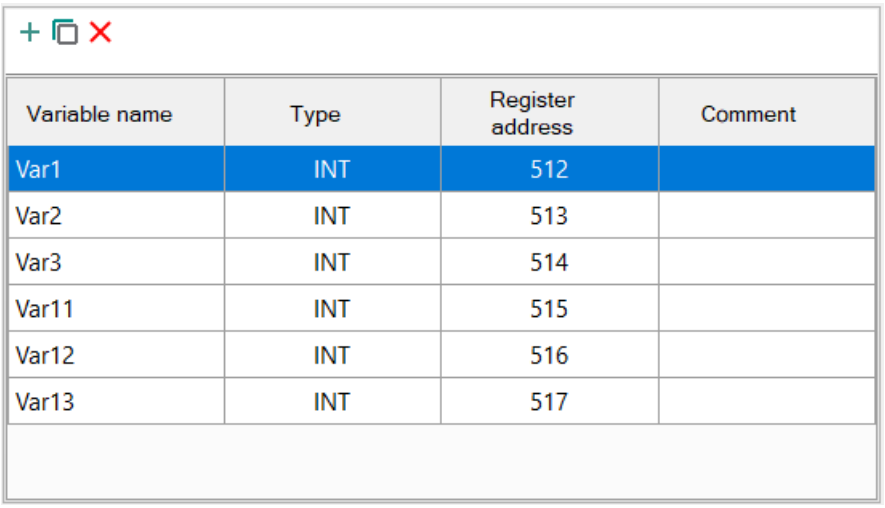


Variable duplication settings:

– **Name** – the name of the variable to be duplicated.
– **Start number** – the initial number that will be appended to the duplicated variable's name.
– **Quantity** – the number of variables to add.
– **Address step** – the increment step for the **Register** parameter value.

The variables will be added to the table.

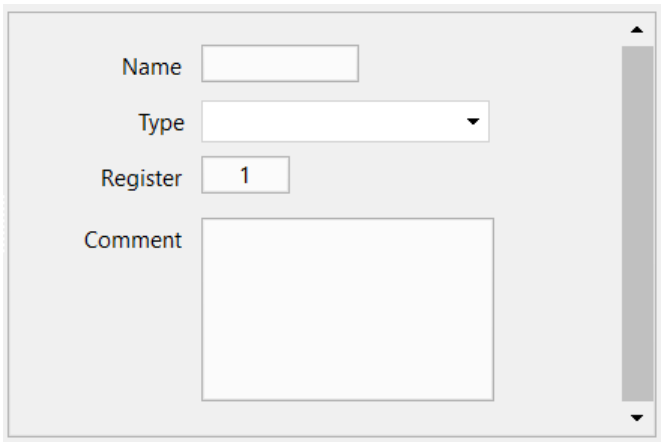
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Variable name	Type	Register address	Comment
Var1	INT	512	
Var2	INT	513	
Var3	INT	514	
Var11	INT	515	
Var12	INT	516	
Var13	INT	517	

The duplicated variables added into the LoRa interface with postfix _R or _W have a Start number before the postfix.

Device variable properties



The device variable properties are configured to the right of the table:

– **Name** – Name of the device variable (set by the user)
– **Type** – *type* of the device variable: integer or floating-point

> **NOTE**
> Boolean variables can be read into an integer variable. The state of a discrete input can be extracted using the *EXTRACT* block or a suitable macro from the *Component manager*.

– **Register** – register address of the device variable (set by the user). The range of available addresses is specified in the device's *user manual*.
– **Comment** – a textual description of the variable's value in the *variable table*.

Slave mode for the second generation devices

Second-generation devices operate in Slave mode by default. Device variables are configured through the *variable table* on the **Network, Slave** tab.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

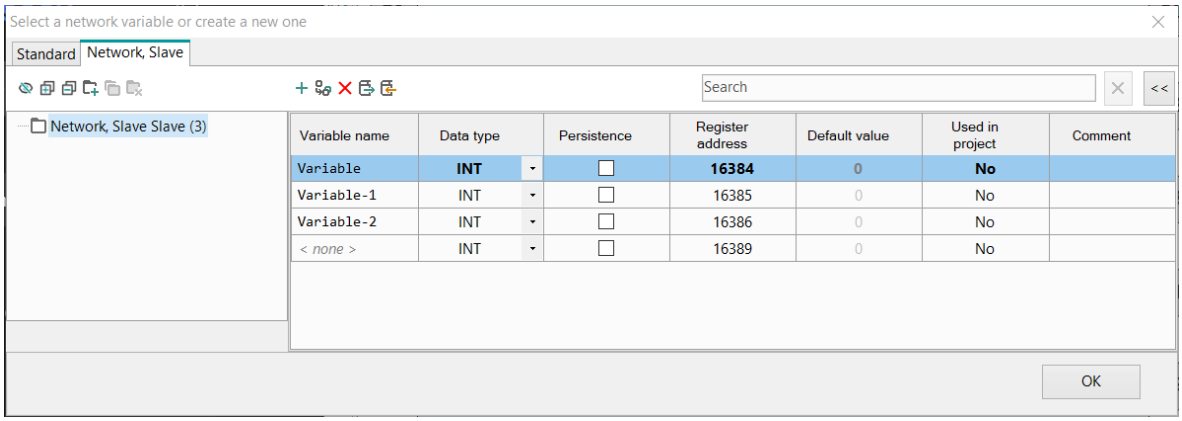


Fig. 5.8 Variable table

– **Variable name** – name of the device variable (set by the user)
– **Data type** – integer or floating-point

> **NOTE**
> Boolean variables can be read into an integer variable. The state of a discrete input can be extracted using the *EXTRACT* block or a suitable macro from the *Component Manager*.

– **Persistence** – when the checkbox is enabled, the variable is stored in the non-volatile system EEPROM area. Information about variable persistence is saved with the project.
– **Register address** – address of network variables in the range from 16,384 to 20,479
– **Default value** – variable value that will be initialized when the program starts
– **Used in project** – indicates if the variable is linked to blocks in the program. If the variable is bound, the value is **Yes**.
– **Comment** – a textual description of the variable for display in the variable table

After connecting the device to akYtec ToolPro or akYtec Cloud, the network variables will be displayed in the parameter tree.

To configure the display, select a variable in the table and click the >> **Parameter Settings** button in the upper right corner of the window. The **Parameter Configuration** panel will appear.

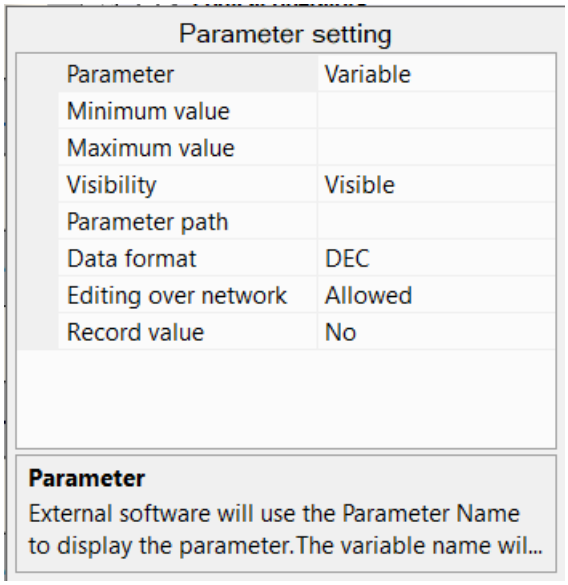


Fig. 5.9 Parameter setting

– **Parameter** – parameter name that will be displayed in the parameter tree when working with akYtec ToolPro and akYtec Cloud.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **Minimum value** – minimum parameter value for display in akYtec ToolPro and akYtec Cloud, does not affect program execution logic.
- **Maximum value** – maximum parameter value for display in akYtec ToolPro and akYtec Cloud, does not affect program execution logic;
- **Visibility** – parameter visibility when working with akYtec ToolPro and akYtec Cloud: **Visible** – parameter is displayed in the parameter tree, **Hidden** – the parameter is not displayed in the parameter tree;
- **Parameter path** – ´parameter will be displayed when the device works with external software. Directory nesting is added using a backslash – \.
- **Data format** – format in which the parameter value will be displayed; relevant for integer variables; available formats: **DEC** (decimal), **BIN** (binary, displayed in akYtec ToolPro as a bit mask), **hex** (hexadecimal), and **enum** (enumeration, for uint16 only).
- **Editing over network** – ability to edit the parameter value from akYtec ToolPro and akYtec Cloud.
- **Record value** – whether parameter value needs to be archived: **Yes** – the parameter value is recorded in the device archive, **No** – the parameter value is not recorded in the device archive.

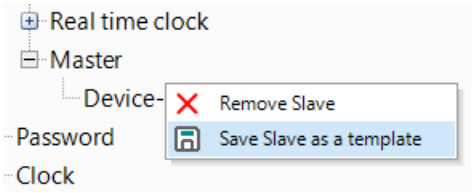
5.3.2.3 Network device templates

Network Device Templates

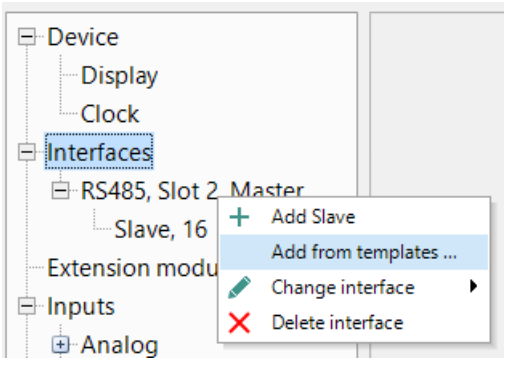
For interfaces operating in Master mode, the creation and use of network device templates is available. The parameters and variables of a configured device can be saved as a template file.

Creating a Template

To create a template, select **Save Slave as a template** in the device's context menu or in the upper part of the device configuration window.



In the opened file explorer window, select the location and enter the file name. The file will be saved with the **.dvtpe* extension. The saved template can be used for other interfaces and projects.
To use a saved template, right-click on the interface name in the configuration tree and select **Add from templates…** In the opened file explorer window, select the file containing the template.

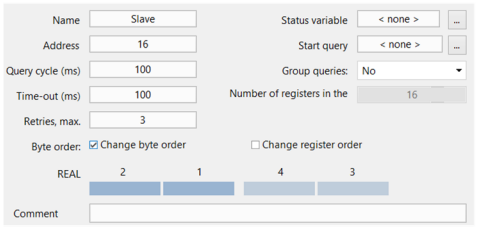


The device from the template will be added to the interface along with all its parameters.
For a number of akYtec devices, there are ready-made templates that can be downloaded from the online database using the *Component Manager*.
When using network device templates created in projects for *first-generation* devices in projects with *second-generation* devices:

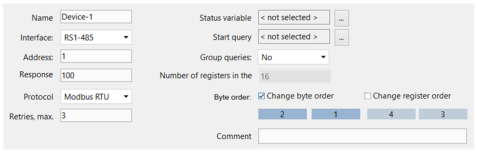
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– When **group polling** is selected, the number of registers in the request will either retain its value (if the entered value is from 2 to 16), or will be set to 16 if the value stored in the template is greater than 16:

Slave parameters
for first-generation devices

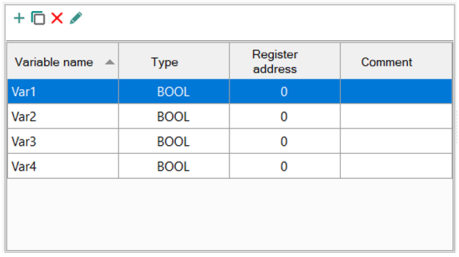


Slave parameters
for second-generation devices



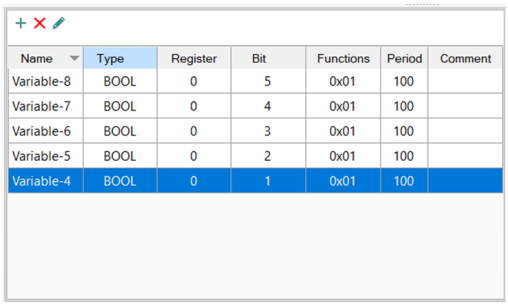
– A variable with read and write function is split into two variables: a read function _read and a write function _write:

Variable table of Slave
for first-generation devices



Variable name	Type	Register address	Comment
Var1	BOOL	0	
Var2	BOOL	0	
Var3	BOOL	0	
Var4	BOOL	0	

Variable table of Slave
for second-generation devices

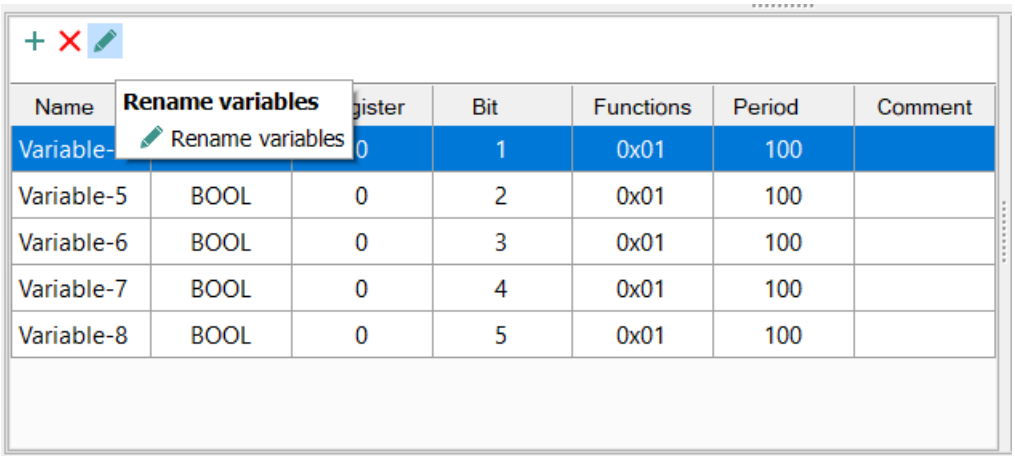


Name	Type	Register	Bit	Functions	Period	Comment
Variable-8	BOOL	0	5	0x01	100	
Variable-7	BOOL	0	4	0x01	100	
Variable-6	BOOL	0	3	0x01	100	
Variable-5	BOOL	0	2	0x01	100	
Variable-4	BOOL	0	1	0x01	100	

When using network device templates created in projects for second-generation devices in projects with first-generation devices, the polling period parameter will revert to its default value (100 ms).

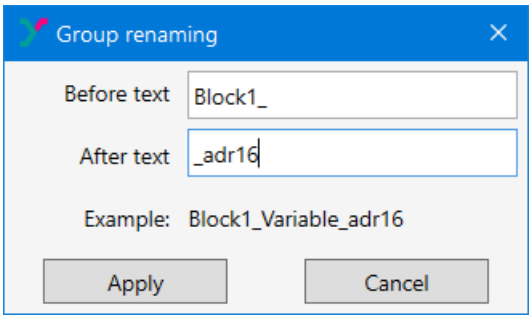
Group variable renaming

To batch rename variables in the variable table of the Slave device in the device configuration window, click the button.

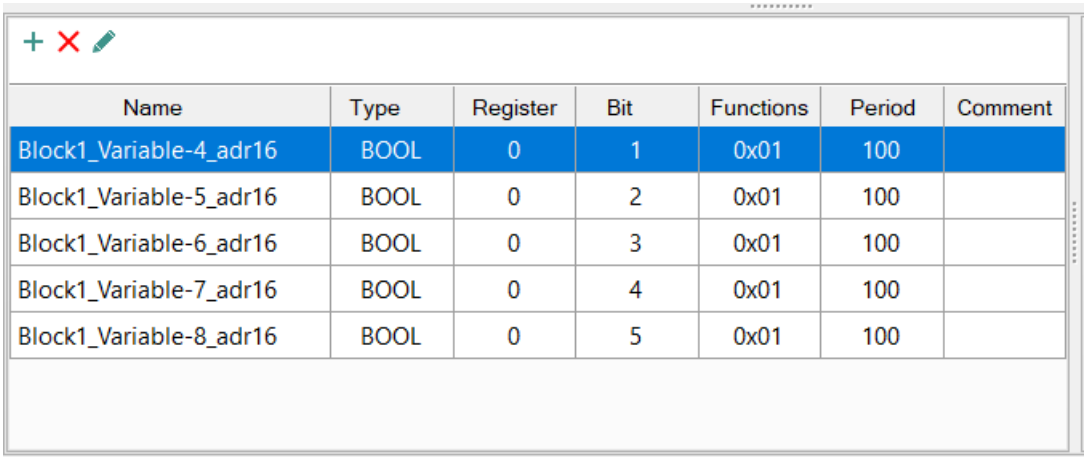


In the window that appears, enter the text to display before and after the variable name, and click the **Apply** button.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



As a result, all variables will be renamed according to the entered data:



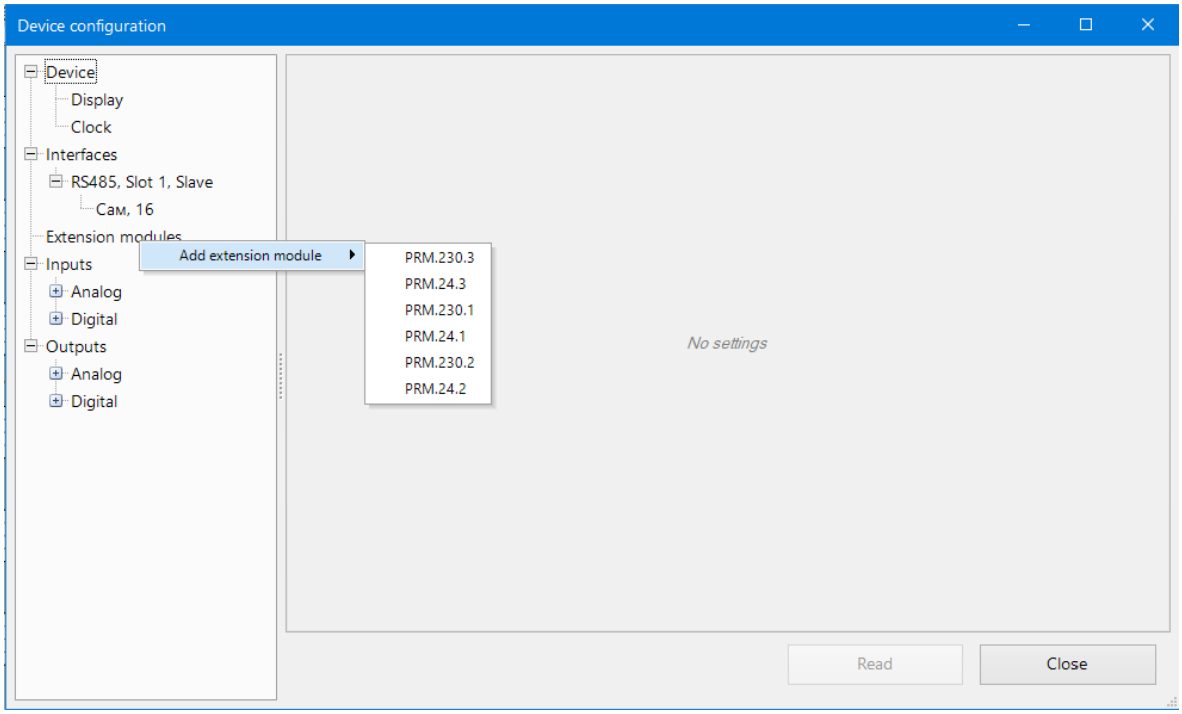
Name	Type	Register	Bit	Functions	Period	Comment
Block1_Variable-4_adr16	BOOL	0	1	0x01	100	
Block1_Variable-5_adr16	BOOL	0	2	0x01	100	
Block1_Variable-6_adr16	BOOL	0	3	0x01	100	
Block1_Variable-7_adr16	BOOL	0	4	0x01	100	
Block1_Variable-8_adr16	BOOL	0	5	0x01	100	

When replicating a variable that was renamed using batch renaming, the "Text after" will not change; the initial and subsequent numbers will be added to the name of the replicated variable.
The batch rename settings are saved in the template and are available for editing when the template is subsequently used.
Renaming each variable individually is available in the *variable table* and in the *program code*.

5.4 Extension modules

Up to two I/O extension modules of type PRM can be connected to base device. For further information about extension modules refer to the PRM user guide.
To use module I/O points in the circuit program, add the module to the group ***Extension modules*** using its context menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



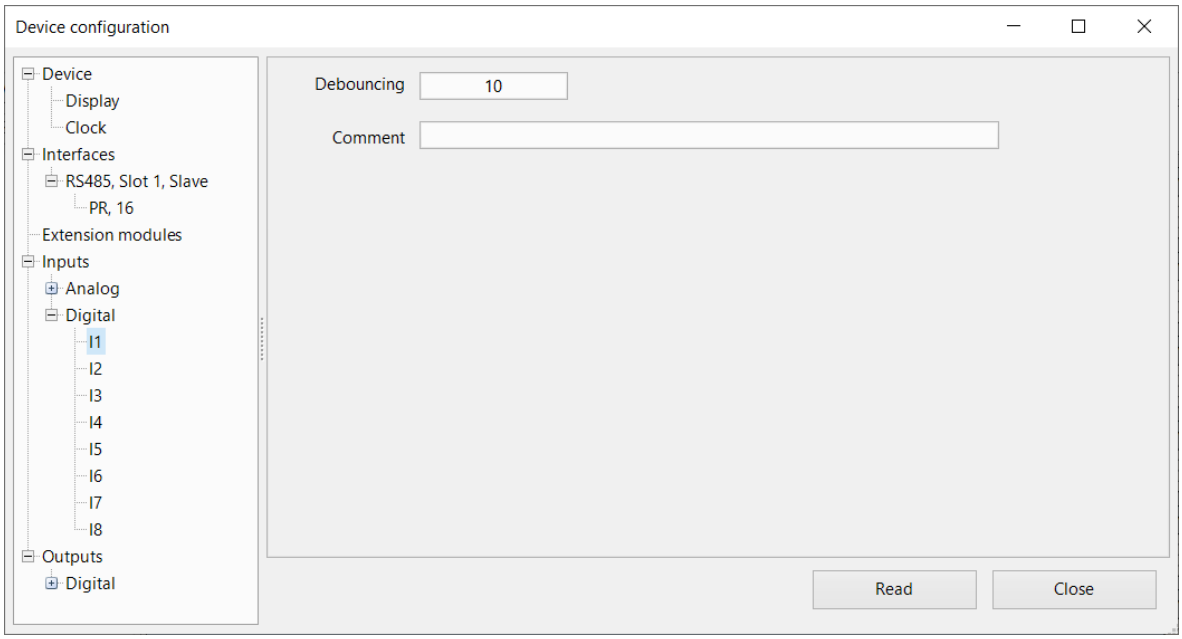
The additional I/O points of the added modules can be configured in branches ***Inputs*** and ***Outputs*** respectivel. They are displayed in the tree as ***Ix(y)*** and ***Qx(y)*** respectively, where ***x*** is the ordinal number of the I/O point on the module and ***y*** is the ordinal number of the module counting from the base device.
Before uploading the project to the base device, all modules must be connected via the internal bus to base device and powered on. The module firmware is synchronized with the current version of ALP when uploading a project.

5.5 Inputs and outputs

Inputs

The content of the branch ***Inputs*** depends on the resources of the target device. It can be analog and/or digital inputs.
The parameter ***Comment*** is common for all types of inputs. The text in this field is displayed in a tooltip, when the mouse cursor is over the input in the workspace. The text can be entered in Property Box too.
For further details about the configuration of the inputs, refer to the device user guide.

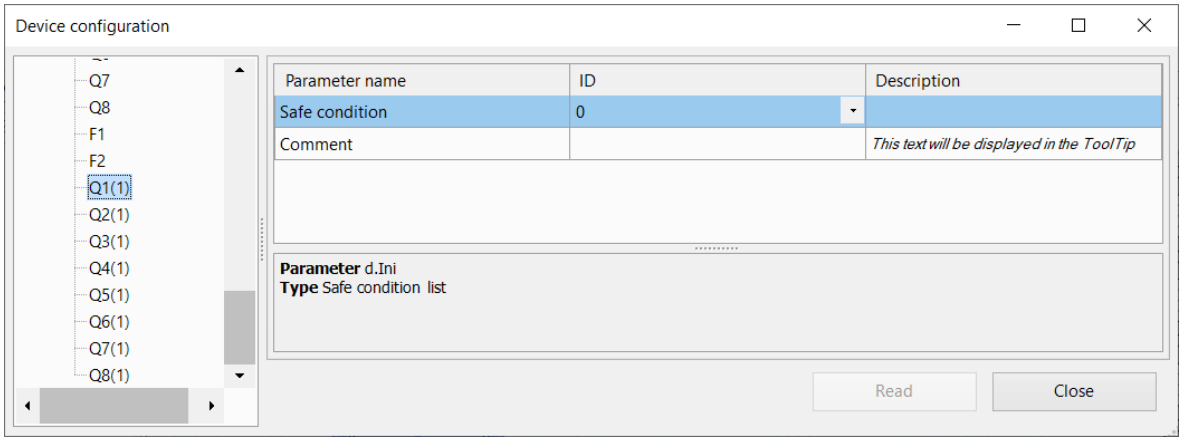
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Other input parameters depend on the types of the input and the device.

Outputs

The content of the branch ***Outputs*** depends on the resources of the target device. It can be analog and/or digital outputs.
The parameter ***Comment*** is common for all types of outputs. The text in this field is displayed in a tooltip, when the mouse cursor is over the output in the workspace. It can be entered in Property Box too.
For further details about the configuration of the outputs, refer to the device user guide.
The digital outputs of the extension module have an additional parameter ***Safe condition***. The parameter specifies the output state in case the connection between the module and the base device is lost.



Settings for devices on the new hardware platform

The window for setting up inputs and outputs for devices on the new platform has a different interface, and the parameters on the right side of the window are presented in a table.
For devices on the new platform, the menu for setting the safe state of the outputs is located in the branch of the added expansion module.

5.6 Password

For devices on the new platform, you can set a password to protect the device.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The password is set in the **Password** section in the settings tree only for the device connected to PC.

Create password

If a password is not set in the device, then password creation will be active in the settings window. Enter your password and confirm to continue.

Changing and resetting password

If the device already has set a password, you can change or reset it.
To change the password, enter current password and new password in the **Change password** columns.
To reset your password, enter your current password in the **Reset password** column.
If you have lost your password, see the device user guide in order to reset it.
For a password-protected device, a password is required when transferring a program, see *Upload project to device*.

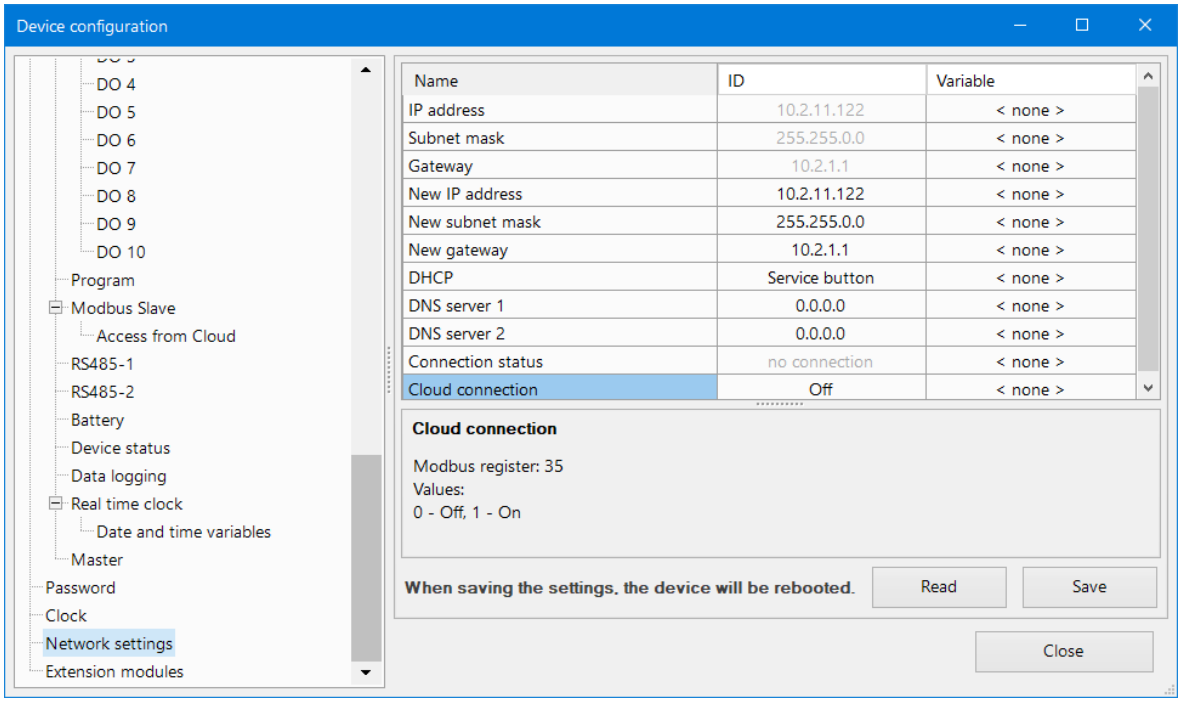
5.7 Connecting to akYtec Cloud

akytec Cloud connectivity is available for devices on the new platform (i.e. PR103).

NOTE
For a list of devices on the new platform, see *About*.

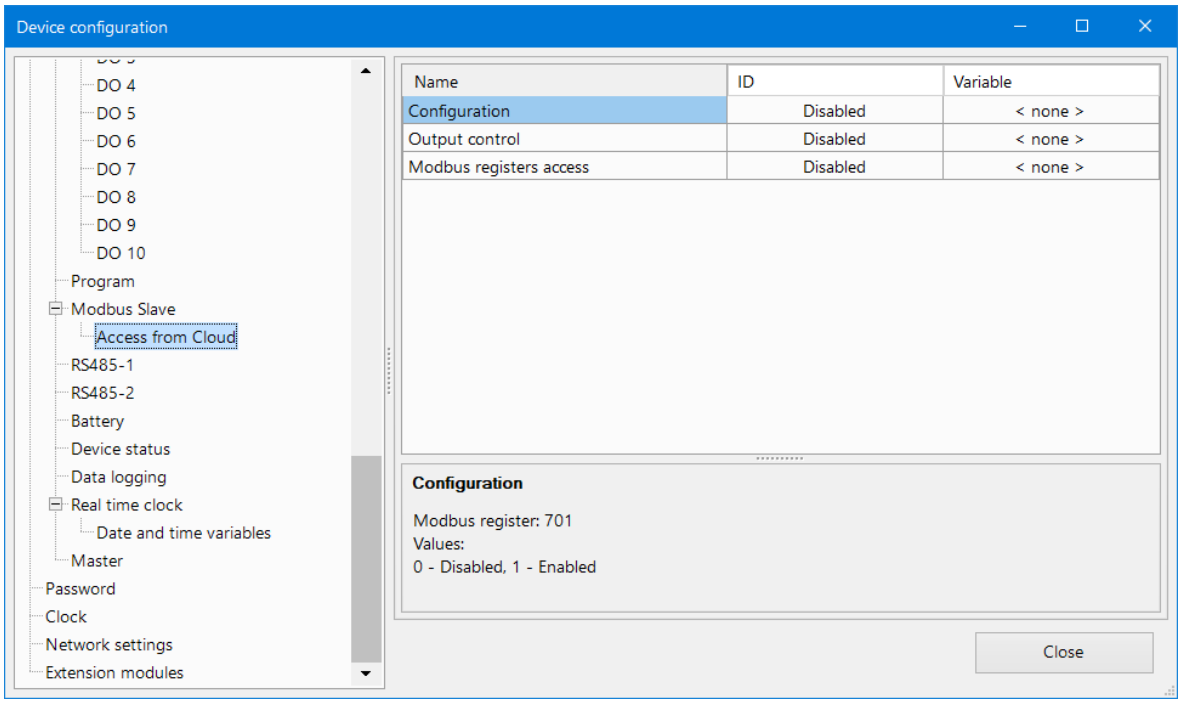
NOTE
To connect to akYtec Cloud, the device must be password protected. Without the password, the device will not connect to akytec Cloud. For setting the password, see *Password*.

Before setting up connection to akYtec Cloud, you should set up authorization of the device in akYtec Cloud, for details see *akYtec Cloud User Manual* and device *User Manual*. During operation, the device must be connected to the Internet for data transfer.
In the **akYtec Cloud Connection Settings** section of the Settings tree, you can set the connection and akYtec Cloud connection status.



In the **Access from Cloud** section of the configuration tree, you can set the akYtec Cloud access levels to the device.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



For an example of working with akYtec Cloud, see *section Task 3. Direct connection of PR103 to akYtec Cloud*.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6 Variables

Variables are used for writing and reading values on the project diagram and when programming *device display*.

In-Use Variables

A variable is considered **in-use** if it meets **at least one** of the following conditions:

- General conditions for all variable types:
  - It is linked to a visualization element.
  - It is linked to a device parameter.
  - It is linked to physical buttons.
  - It is used in logical transitions between screens.
  - An output block for a non-volatile variable (Non-volatility = On) exists for the variable on the diagram.
- Additional conditions for variables of the Array type:
  - For an array with **Boolean elements** and non-volatility set to "Off" – an output block for the array or its elements exists on the diagram.
  - For an array with **Integer or Real elements** and non-volatility set to "Off" – an input block for the array or its elements exists on the diagram.

If none of the conditions are met, the variable is considered **unused**.
Unused variables:

- do not participate in the program logic.
- are not involved in program simulation.
- do not occupy device memory.

Variable table

Project variables are displayed in the Variable table.

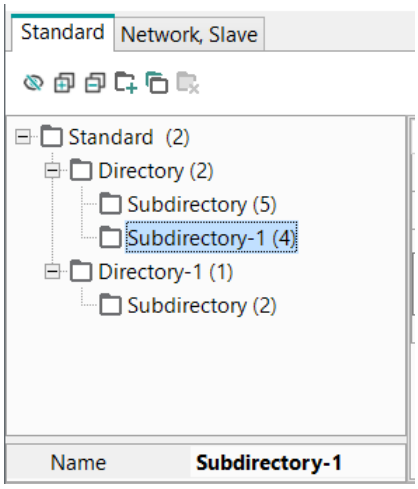
To see all project variables, click the icon  in the toolbar or use the menu item ***Device*** → ***Variable table***.
The variables are divided into three groups, each of which has a separate tab in the table:

- **Standard**
- *Service*
- *Network* – a separate tab for each communication interface.

Variable directory tree

The **Variable directory tree** allows you to create catalog sections for organizing variables. A separate catalog tree is displayed for each variable category. For standard and network variables, you can create additional catalog sections (maximum nesting depth is 3). The catalogs for service variables are fixed and cannot be modified.
Next to each catalog name, the number of variables within that catalog is shown in parentheses. This count either includes or excludes variables in nested catalogs, depending on the "Show nested variables" setting.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



	Enable/disable the display of nested variables
	Expand nested sections
	Collapse nested sections
	Create a variable catalog section. Creating a section is also available through the catalog's context menu.
	Duplicate catalog — the selected folder will be duplicated in the parent catalog with all nested sections and added variables. Copying a section is also available through the catalog's context menu. Select **Copy**, then select the catalog where you want to add the copied section, right-click and select **Paste**. **NOTE** The target catalog for adding information must be one level above the copied section.
	Delete catalog. Deleting a catalog section is also available through the catalog's context menu. A catalog section containing variables cannot be deleted.

To move directory sections, press and hold the **Shift** key, and use the mouse to move the section to a new location.

NOTE
Moving sections is only available within the parent directory.

To rename a directory section, enter a new name on the **Name** line at the bottom of the **Variable Directory Tree**.

NOTE
The directory section name field cannot be empty or duplicate the name of another section within the parent directory. If an incorrect name is entered, an error message box will appear.

To determine the directory section in which a variable resides, highlight the variable in the table and press ***Ctrl + H***. The section where the variable is located highlights in blue in the directory tree. The line below shows name of the section, and the section with the variable opens in the variable table. You may also hover your mouse over a variable to summon a pop-up window with the variable's name and location.

Variables

To use a variable in a project, you must first create it in the variable table.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

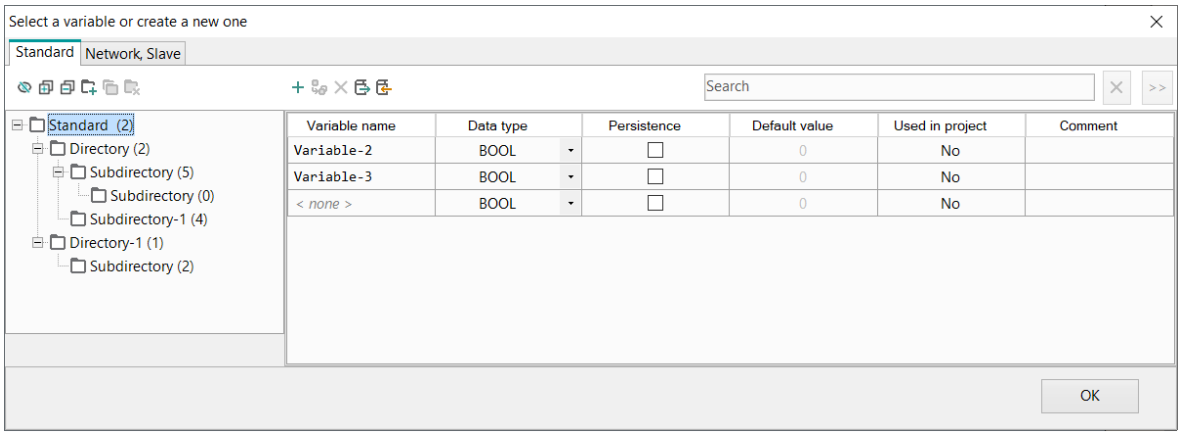


Fig. 6.1 Variable table

+	Create a variable
	Duplicate a variable
×	Delete a variable
	Export standard variables to a CSV file
	Import standard variables from a CSV file
	Expand all arrays
	Collapse all arrays

To create a variable, click the + button in the upper left part of the table, or enter a variable name in the "Variable name" column. Select the *type* and fill in the remaining parameters according to your operational needs.

– **Variable name** – name to be displayed on the Variables panel and on the variable block in the project.
– **Data type** – one of three types can be assigned: Boolean, integer, floating point. See the *Data Types 6.1* section for more details.

  When selecting a variable of the Array type, enter the following in the **Array Settings** window that appears:

  – **Size of array** – the number of elements in the array, maximum 256. Values are specified in the format [x..y], where x is the index of the first element and y is the index of the last element.
  – **Element type** – the data type of the array elements: BOOL, INT, REAL.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

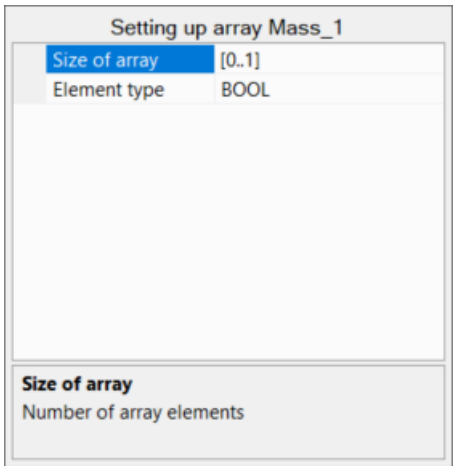


Fig. 6.2 Array settings

– **Persistence** – enable the checkbox in the setting field to save the variable value in the device ROM in case of power failure. The storage time of the variable in ROM depends on the device type, see device *operation manual*

NOTE
For devices on the new platform, when the Persistense checkbox is enabled in the variable table, the variable value is saved:

– To the system EEPROM for a standard variable bound to an device parameter or visualization element. For the network Slave variable.

– To the EEPROM for a standard variable not bound to the device parameter or to the visualization element.

When you save a project, the information about the non-volatile variables is also saved.

– **Default value** – the value that the variable will keep until a new value is written to it.
– **Used in project** – indication of binding to blocks in the program. If the variable is bound, the value is **Yes**.
– **Comment** – text description to be displayed in the tooltip on the project diagram when the cursor is placed over the variable block.

To duplicate a variable, right-click on the variable line and select the **Duplicate a variable** from the context menu or click the button at the top of the table.

NOTE
Duplicating an array element is not possible.

To remove a variable from the table, right-click on the variable row and select **Delete Variable** in the context menu or click the button at the top of the table.

NOTE
Deleting an array element is not possible.

Exporting variables to a file

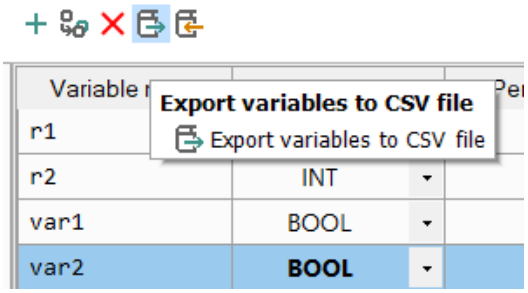
To export a variable tab as a table in ***.csv*** format:

1. Click **Export Variables to CSV File** at the top of the table.
2. In the window that opens, specify the location to save the file.
3. Click the **Save** button.

NOTE
The file name is generated based on the exported tab following the pattern ***ProjectName_Tab_Variables***.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
For second-generation devices, the Slave network variables tab is exported along with the variable parameter settings.



Importing variables from a file

To import a variable table from a ***.csv*** file, click the **Import Variables from CSV File** button at the top of the table. Select the file from which you want to import data and click **Open**. Any catalogs present in the file will also be imported, preserving the variable nesting. If the file contains incorrect data, such as a missing column for the variable name, type, or register, an error message will be displayed. After correcting the error, perform the import again. If there are matches in variable names and/or registers between the project and the imported file, an information window will appear listing all name and/or register conflicts and prompt you to either continue with or cancel the import.

Expand/Collapse arrays

If your project contains a variable of the Array type, you can configure the display of its elements in the variable table. An array is displayed in the variable table as an array tree. To expand the array tree for all array-type variables in the variable table, click the **Expand all arrays** button at the top of the table. This setting is preserved the next time you open the variable table within the same session. To display the elements of a single array-type variable, click the individual "+" button located in front of the array's name.

To collapse the array tree for all array-type variables in the variable table, click the **Collapse all arrays** button at the top of the table. This setting is preserved the next time you open the variable table within the same session. To hide the elements of a single array-type variable, click the individual "-" button located in front of the array's name.

Searching for Variables

The variable table supports searching and filtering by name. The space character acts as a logical OR, allowing you to filter variables by multiple criteria.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

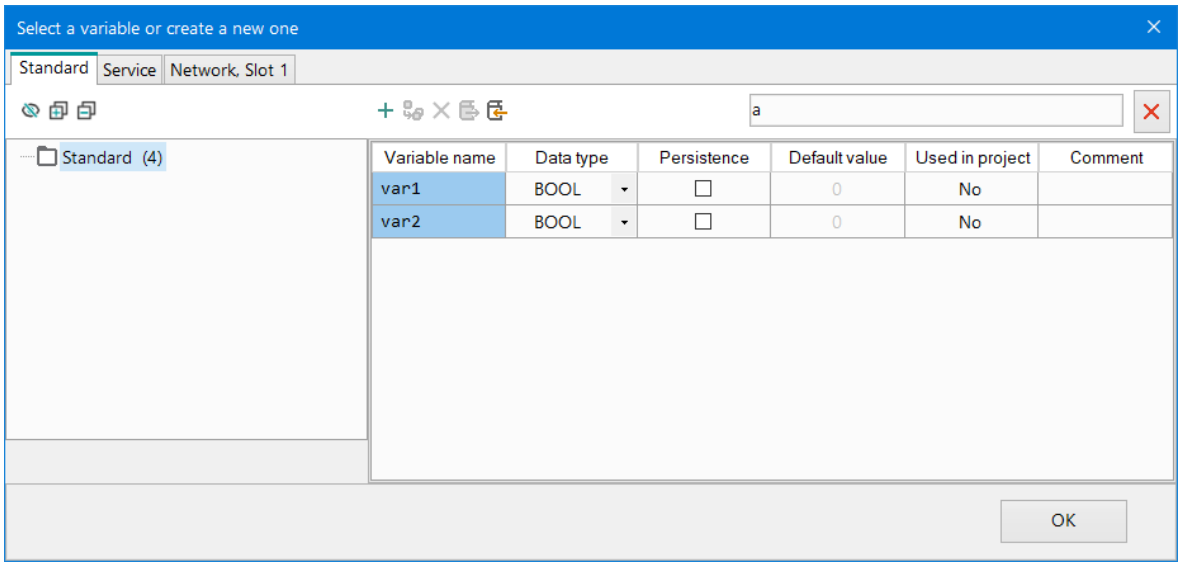


Fig. 6.3 Search for variables

If the variable entered in the search line does not exist, you can create it by clicking the + button at the top of the table. The variable is automatically assigned the name entered in the search box.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

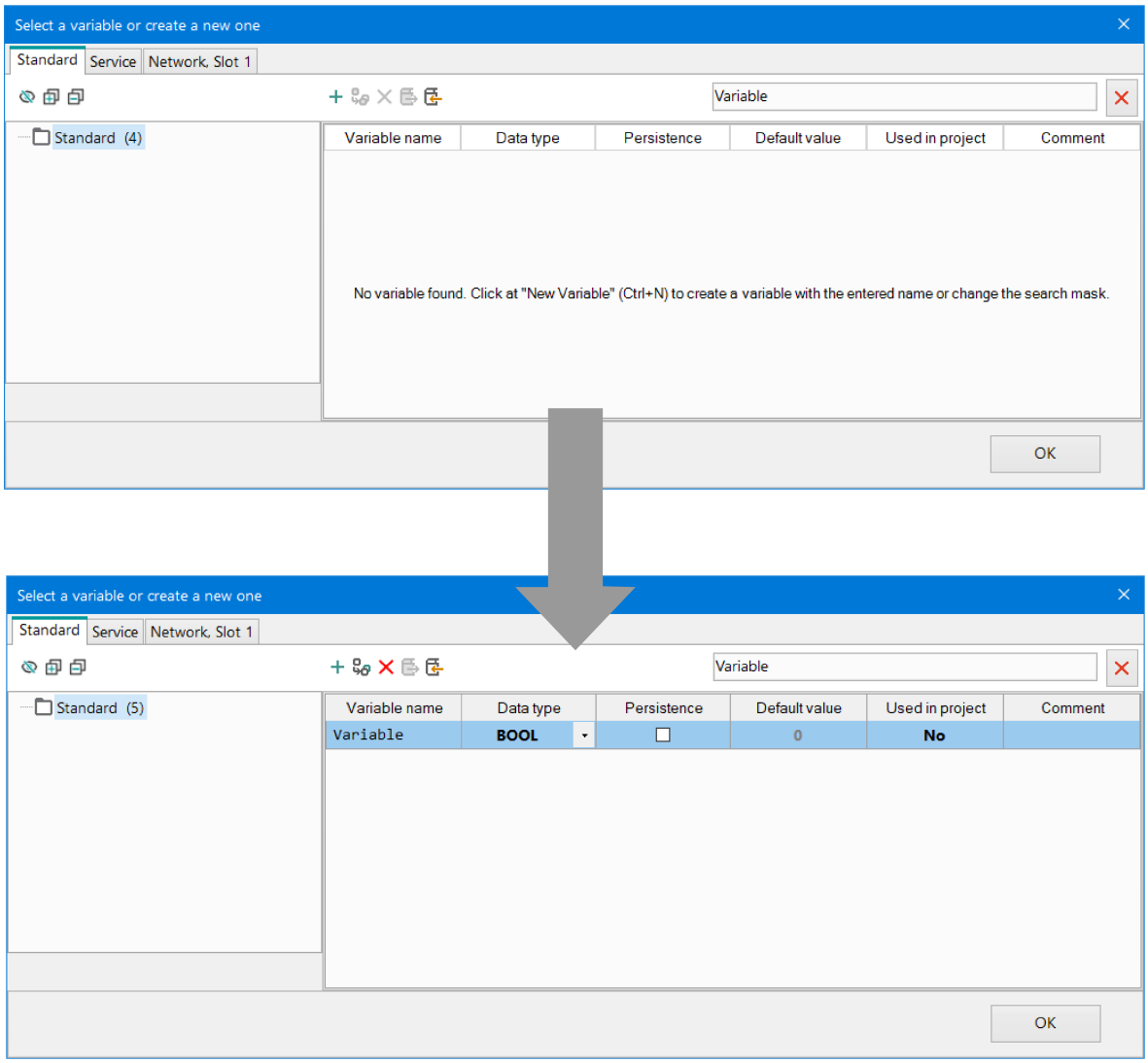
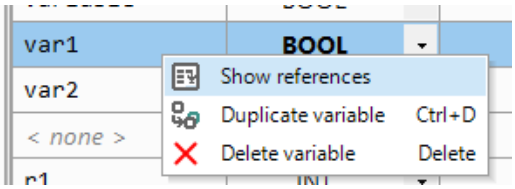


Fig. 6.4 Creating a variable

Variable Table Context Menu

To find where a variable is used in the project, right-click on the variable's row in the table and select **Show References** from the variable's context menu.



The new window shows information about the selected variable.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

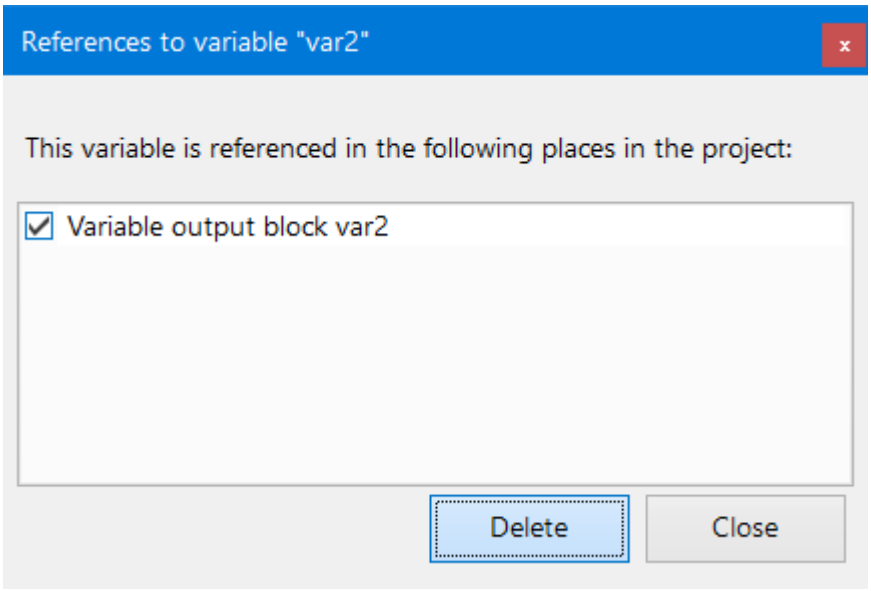


Fig. 6.5 References to a variable

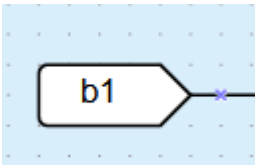
For an Array type variable, references are shown both for the array itself and for its individual elements.
To remove a variable block's binding within the project, check the boxes next to the desired variables and click the **Delete** button. The variable will remain in the table but will no longer be used in the project.

6.1 Data types

The variables of the following types can be used in a program:
– Boolean (***BOOL***)
– Integer (***INT***)
– Real (***REAL***)
– Array
Values can only be passed from one variable to another if the variable types match.

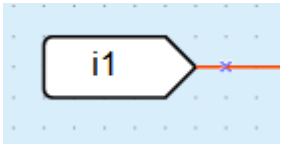
BOOL

A variable of this type has only two possible values: 1 (**True**) or 0 (**False**).
The connecting lines between the BOOL variables in the circuit program are black.



INT

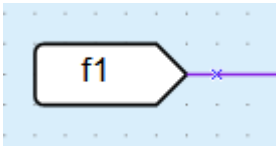
A variable of this type is an unsigned integer in the range of 0…4,294,967,295 (4 Byte).
The connecting lines between the INT variables in the circuit program are red.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

REAL

A variable of this type has a value in the range of -3.402823e+38…3.402823e+38. It is represented by a floating-point number of single-precision (4 Byte).
Single-precision floating-point numbers provide a relative precision of 7–8 decimal digits within a range from 10^{-38} to approximately 10^{38}.
The connecting lines between the REAL variables in the circuit program are purple.

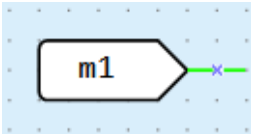


Array

An Array type variable stores an ordered set of elements, all of the same type: BOOL, INT, or REAL. The maximum number of elements in a single array is 256.

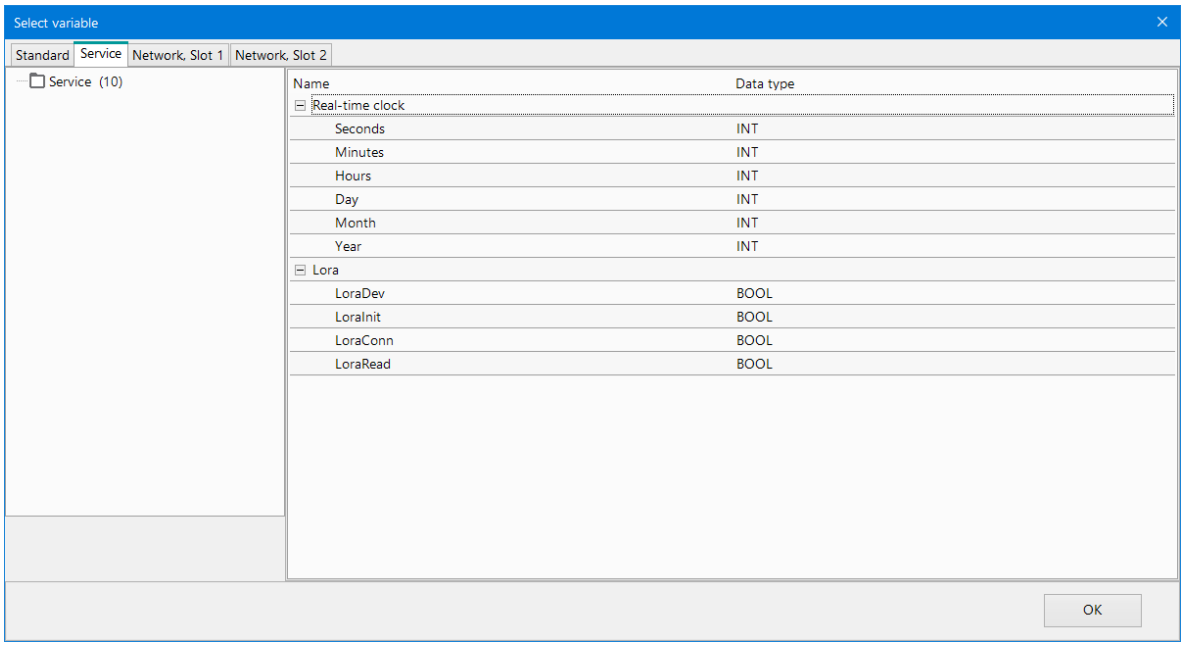
NOTE
Working with arrays is available in ALP starting from version 3.0. In the current implementation, only one-dimensional arrays are supported.

The connecting lines between the Array variables in the circuit program are green.



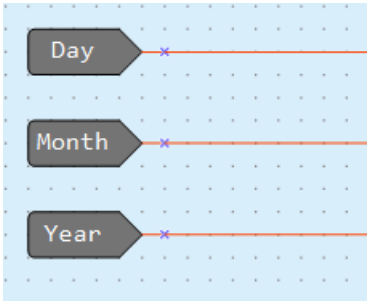
6.2 Service variables

Service variables are associated with the device settings and can only be read. Writing to service variables is not available. The set of service variables can differ depending on the device. If the device has a LoRa communication interface, then the service variables of the interface are also displayed in tab ***Service***.



The blocks of service variables are shown in the circuit program with a gray background.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

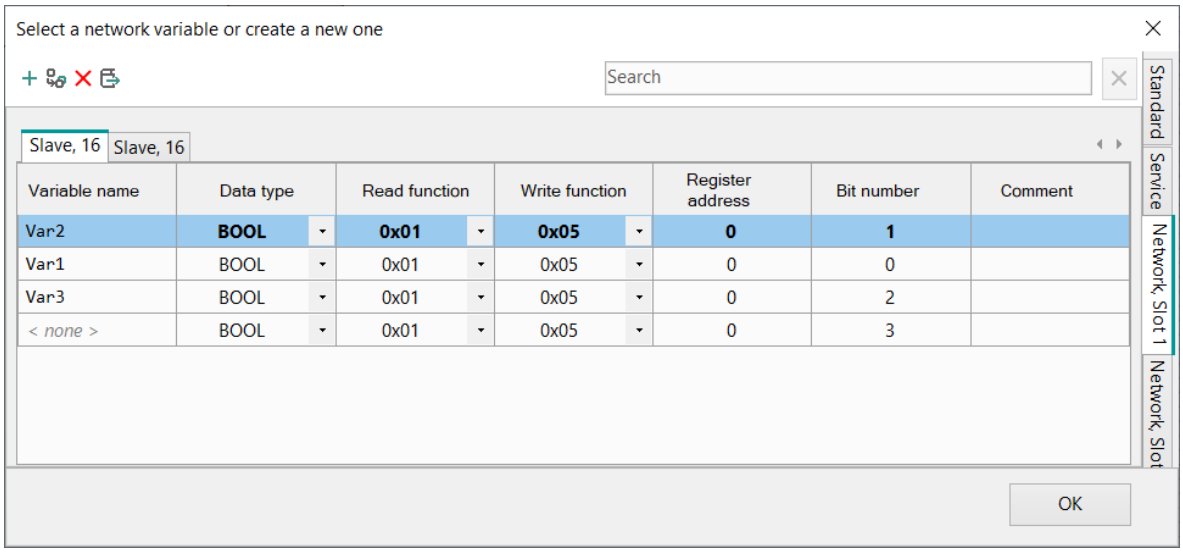


Service variables for the second-generation devices

There are no service variables in the *second-generation* devices. Instead of service variables, variables are bound to device parameters (see section *Binding variables to parameters*).

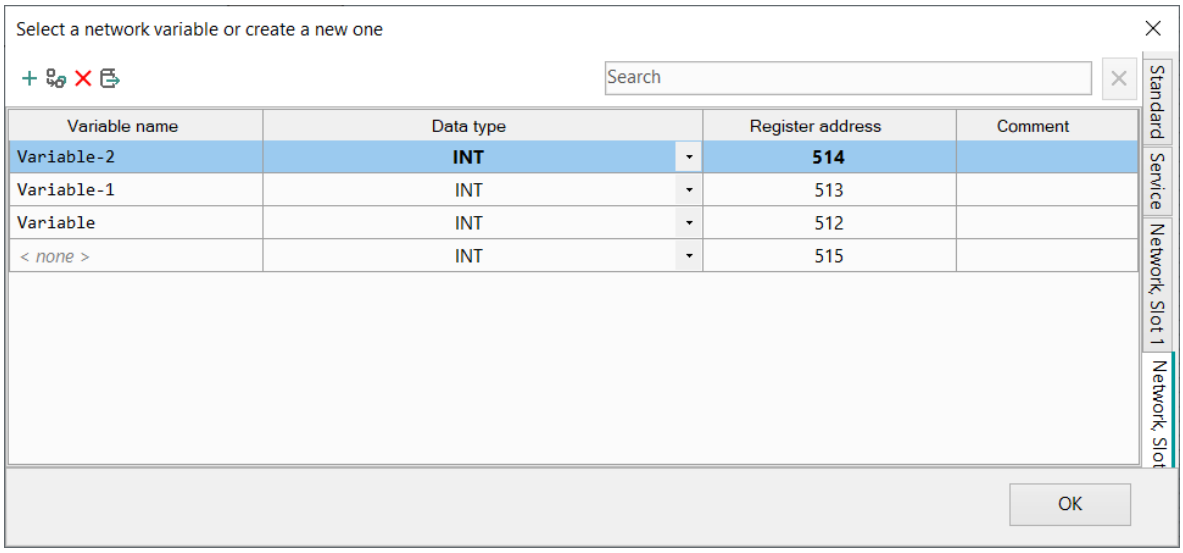
6.3 Network variables

Each interface slot has a separate tab in the table.
If the interface is configured as a master, there are separate tabs for each slave device within the slot tab. The Slave tab contains the variables to be requested for this slave device.



Network variables and their references are deleted in the same way as standard variables.
For more details about network variables for master interface see *Master mode 5.3.2.1* section.
If the interface is configured as a Slave, all network variables to be requested by the master are shown in one list.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



6.4 Binding variables to parameters

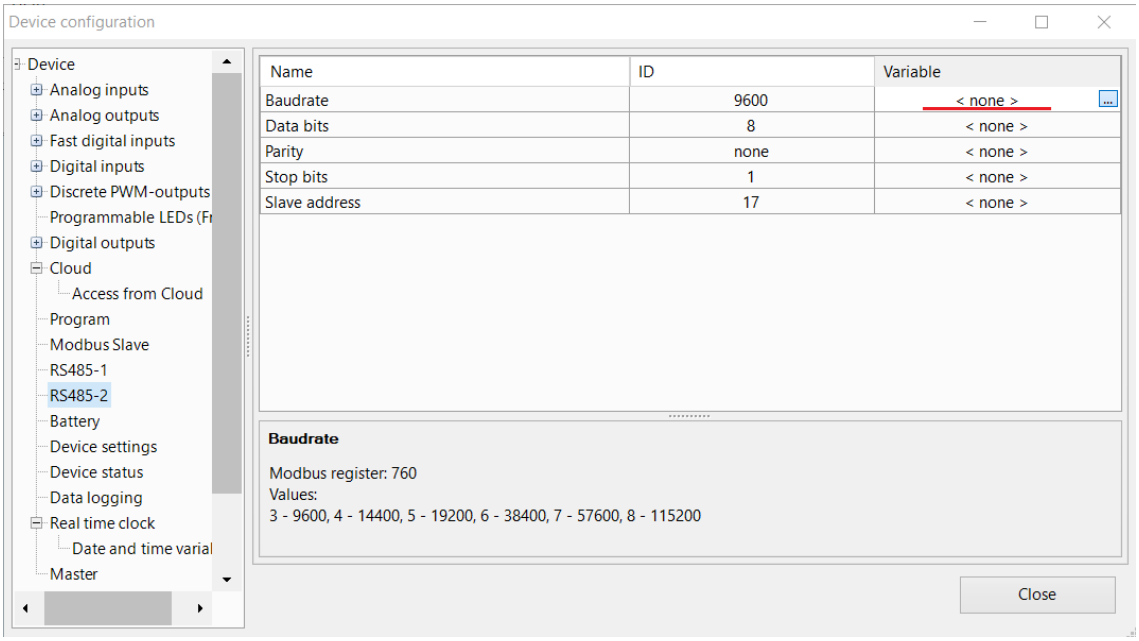
NOTE
Variable binding to parameters is only available for **second-generation devices**. For a list of second-generation devices, see the *About* section.

In second-generation devices, you can bind variables to device parameters and use them in the program. Variables bound to parameters are available for reading and modification.
To bind a variable to a device parameter:

1. Open **Device Settings**.

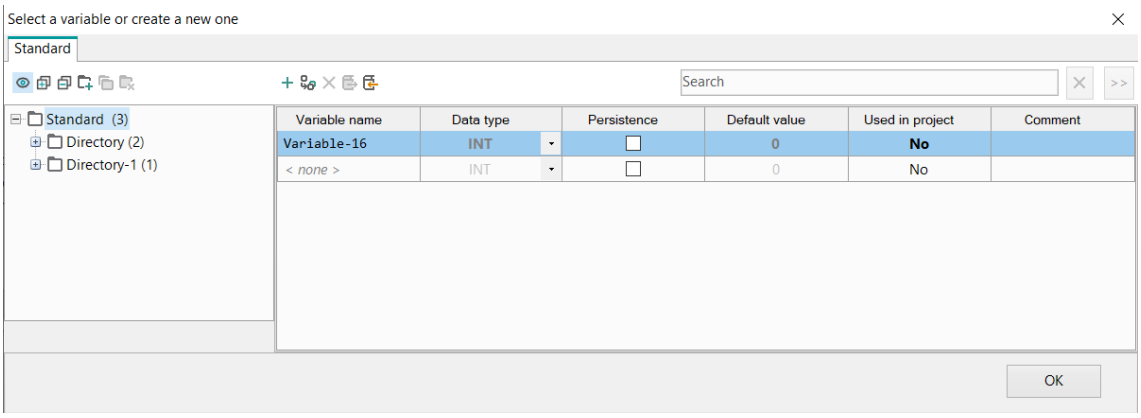
   NOTE
   Variable binding is not available for all parameters.

2. Select the parameter for binding and click the "…" button in the **Variable** column of the settings window.



3. In the Variables window that opens, select or create a variable for binding. Variables highlighted in gray are read-only.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The name of the bound variable will be displayed in the "Variable" column next to the parameter.

After binding to parameters, variables can be used in the program. This allows parameter values to be read and modified within the program.

NOTE
If a variable bound to a parameter is used for writing via a block, and the parameter value has not changed, the value written through the block will be stored in the variable for one cycle. In the next cycle, the variable value will be read from the parameter.

6.5 Copy-paste variable block

Variables can be copied from project to project for reuse and to reduce development time. To copy a variable, select the variable block in the source project and click the button on the toolbar or select the **Copy** command from the block's context menu. The variable is pasted into another project by clicking the button on the toolbar or by selecting the **Paste** command from the canvas context menu.

Variables

When a copied variable block is pasted into a project, all its properties are checked. If a variable with exactly matching properties already exists in the project, it is linked to the block. Variables with differing properties are automatically created in the project.
Copying an array element from one project to another follows these rules:

– If the project where the element is being pasted does not contain an array with properties (type, size) identical to those of the array to which the copied element belongs, a new array is created along with the element, fully replicating the properties of the original array.
– If the target project already contains an array with identical properties, a new array is not created.

The rules for copying Array type variables are the same as those for copying standard variables.
If variable names match but other properties differ, a new variable is created. However, the name conflict is indicated in the variable table, and one of the variables must be renamed.

NOTE
It is not possible to paste REAL type variables into a project for a device that does not support this data type.

NOTE
Non-volatile variables cannot be pasted into a project for a device that does not support them.

Service variables

Service variables cannot be copied into a project for a device that does not support real-time clocks.

Network variables

Only network variables in Slave mode for an identical slot number can be copied to another project. Network variables in Master mode must be created manually. When pasting a copied network

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

variable block into a project, all variable properties are checked. If the project contains a variable with completely matching properties, it is duplicated into the block. Variables with differing properties are automatically created in the project. If variable names or registers match but other properties differ, a new variable is created, but a register or name conflict occurs. The conflict can be resolved in the variable table by assigning a different name or registers.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7 Library

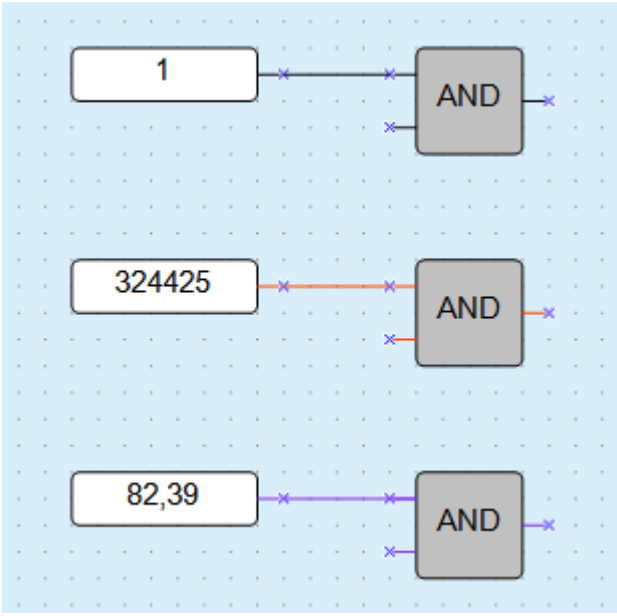
If a project is open, the panel ***Library Box*** contains the following libraries:
– *Functions*
– *Function blocks*
– *Project macros*
Select an icon in the lower part of the panel to show the respective content.
Project macros library comprises the macros that have been created, imported or included to the project from Online Database.
View options can be changed using the icons located in the panel toolbar.

7.1 Functions

The library contains the following function groups:
– *Logical operators*
– *Mathematical operators*
– *Relational operators*
– *Bitshift operators*
– *Bit operators*

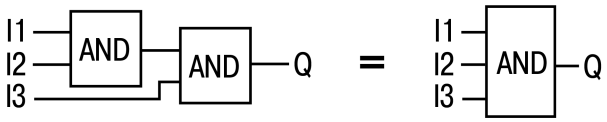
7.1.1 Logical operators

– *Conjunction (AND)*
– *Disjunction (OR)*
– *Negation (NOT)*
– *Exclusive OR (XOR)*
The logical operators can operate with BOOL or INT variables.
If the input values are INT, the operation is performed bitwise and the output is also an INT.

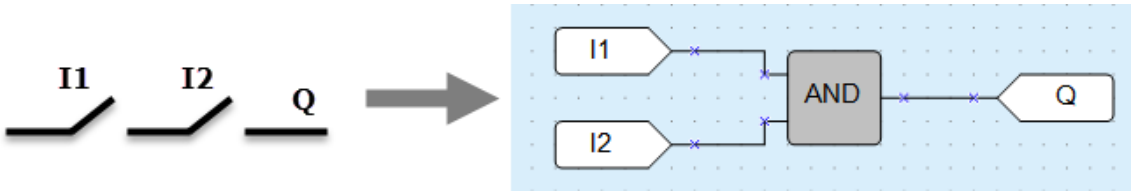


For **AND** and **OR** operators, it should be taken into account that unconnected block inputs will have the following states:
– for **AND** – TRUE
– for **OR** – FALSE
In this case, the blocks act as a signal repeater. To increase the number of inputs for logical operators, their cascade connection is used:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



7.1.1.1 Conjuction (AND)



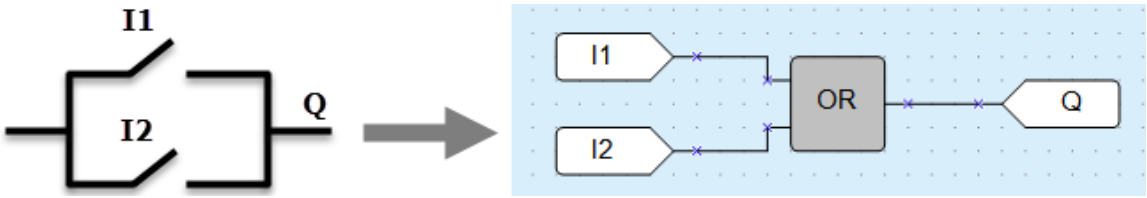
The output **Q** is **True** if both inputs are **True**. The function **AND** represents a serial connection in an electrical circuit.
Truth table:

I1	I2	Q
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise operation example with integer inputs:

AND	0011 (decimal 3)
	0101 (decimal 5)
	0001 (decimal 1)

7.1.1.2 Disjunction (OR)



The output **Q** is **True** if at least one of the inputs is **True**. The function **OR** represents a parallel connection in an electrical circuit.
Truth table:

I1	I2	Q
0	0	0
0	1	1
1	0	1
1	1	1

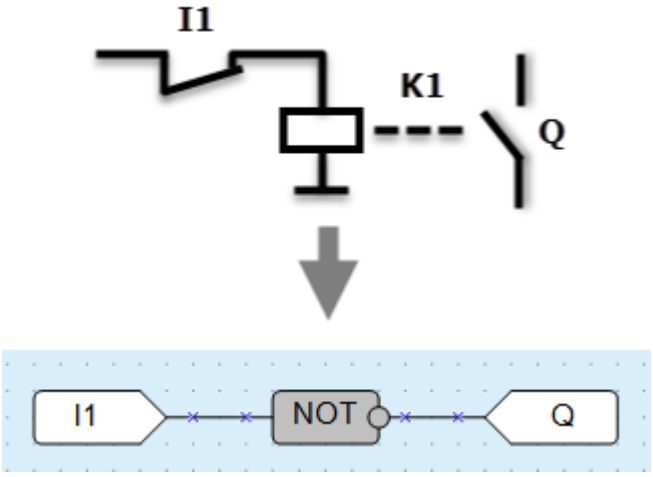
Bitwise operation example with integer inputs:

OR	0011
	0101
	0111

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.1.1.3 Negation (NOT)

The function **NOT** inverts the signal. The output **Q** is **True** if the input is **False** and vice versa.



Truth table:

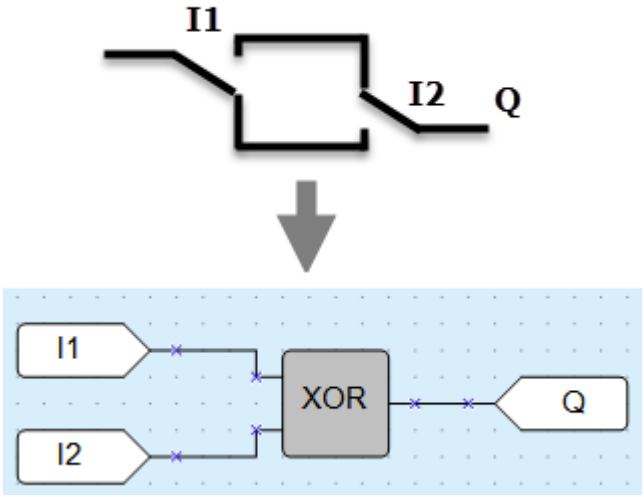
I1	Q
0	1
1	0

Bitwise operation example with integer inputs:

NOT	01
	10

The bitwise NOT, or complement, is a unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value.

7.1.1.4 Exclusive OR (XOR)



The output **Q** is **True** if only one of the inputs is **True**.
Truth table:

I1	I2	Q
0	0	0
0	1	1

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

I1	I2	Q
1	0	1
1	1	0

Bitwise operation example with integer inputs:

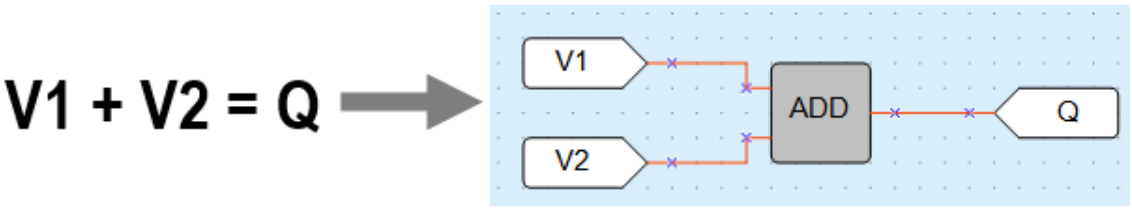
XOR	0011
	0101
	0110

7.1.2 Mathematical operators

There are different operators for different data types:

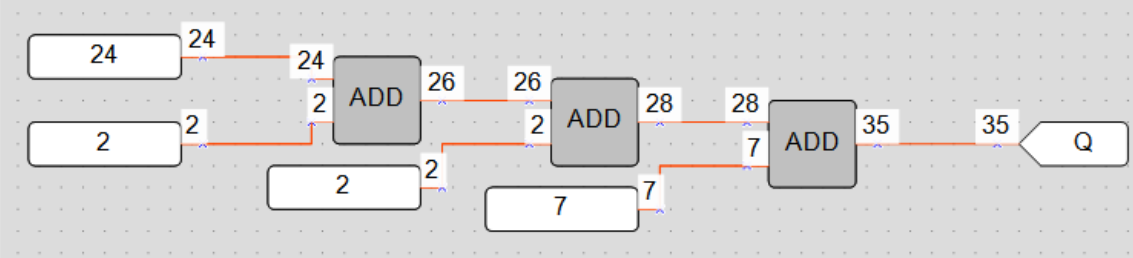
Operator	INT	REAL
Addition	*ADD*	*fADD*
Subtraction	*SUB*	*fSUB*
Multiplication	*MUL*	*fMUL*
Division	*DIV*	*fDIV*
Modulo operation	*MOD*	–
Power function	–	*fPOW*
Absolute value	–	*fABS*

7.1.2.1 Addition (ADD, fADD)



The function ADD operates with INT variables, while the function fADD operates with REAL variables. The output value Q is the sum of the input values.

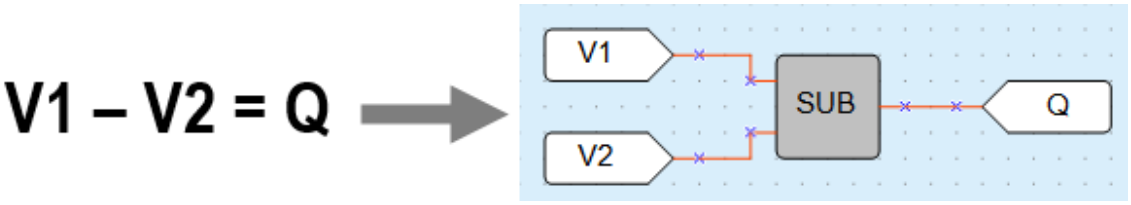
Example:



The output value may not exceed 4294967295 (32 bits). Otherwise the extra bits will be truncated.

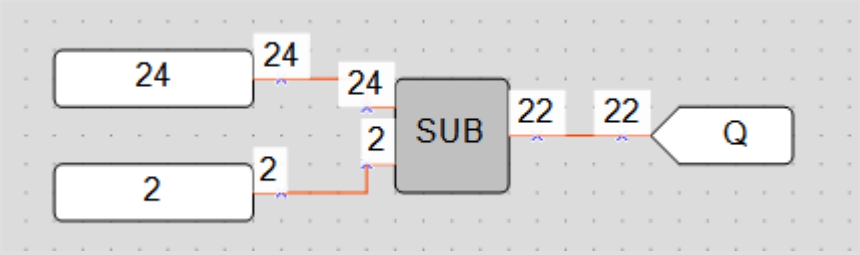
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.1.2.2 Subtraction (SUB, fSUB)

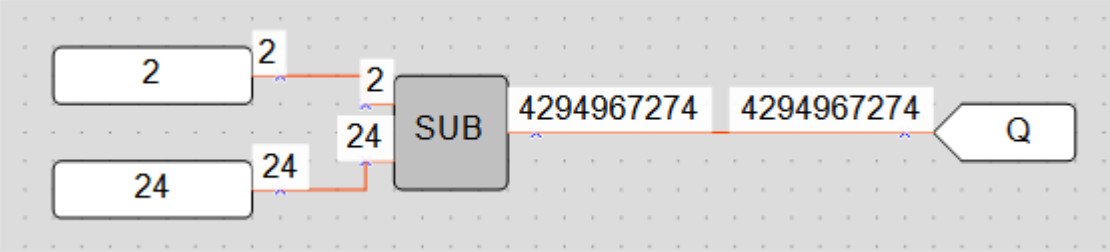


The function **SUB** operates with INT variables, while the function **fSUB** operates with REAL variables.
The output value **Q** is the result of subtraction of the value **I2** from the value **I1**.

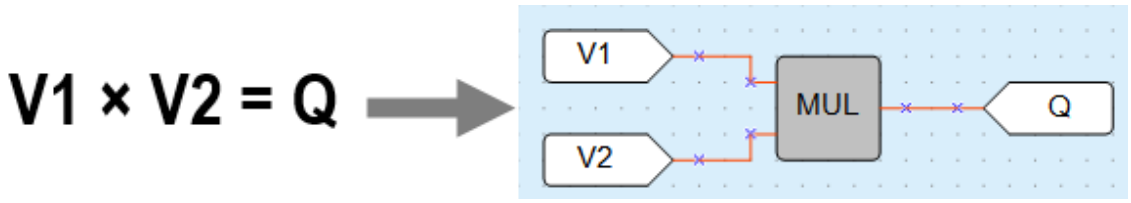
Example:



If the value **I1** is less than the value **I2**, the output is calculated as follows:
Q = I1 + 0x100000000 – I2
0x100000000 = 4294967296



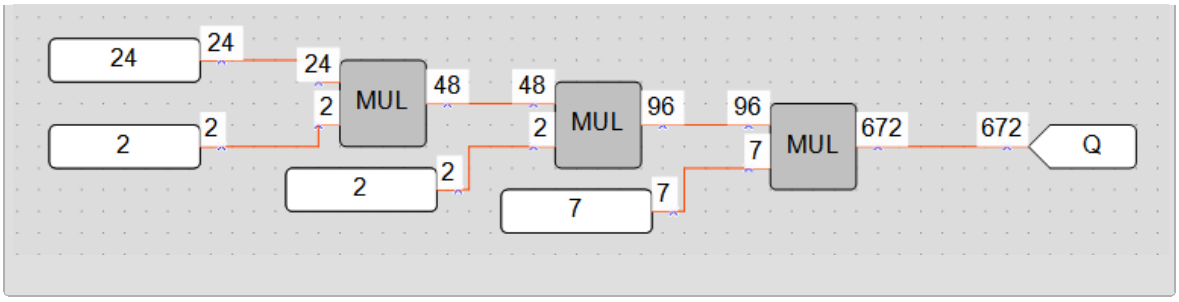
7.1.2.3 Multiplication (MUL, fMUL)



The function **MUL** operates with INT variables, while the function **fMUL** operates with REAL variables.
The output value **Q** is the product of the input values.

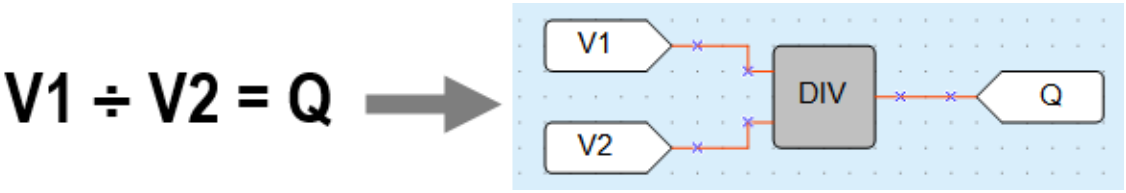
Example:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



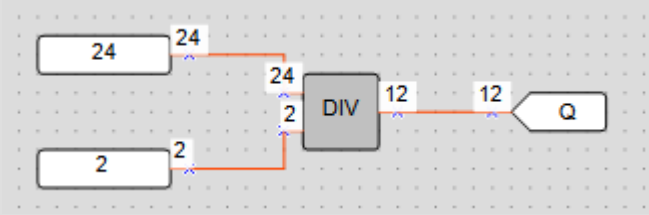
The output value may not exceed 4294967295 (32 bits). If it does happen, the extra bits will be truncated.

7.1.2.4 Division (DIV, fDIV)

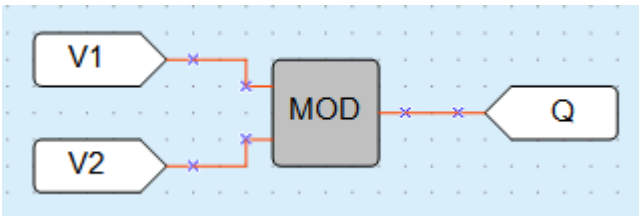


The function **DIV** operates with INT variables, the function **fDIV** operates with REAL variables.
The output value **Q** is the quotient of the input values, where the value **I1** is the dividend and the value **I2** is the divisor.
If the quotient is not an INT, it is rounded down to an INT.
In case of division by 0 the output value is 0xFFFFFFFF.

Example:

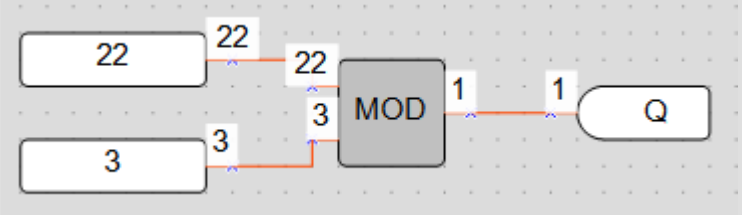


7.1.2.5 Modulo operator (MOD)



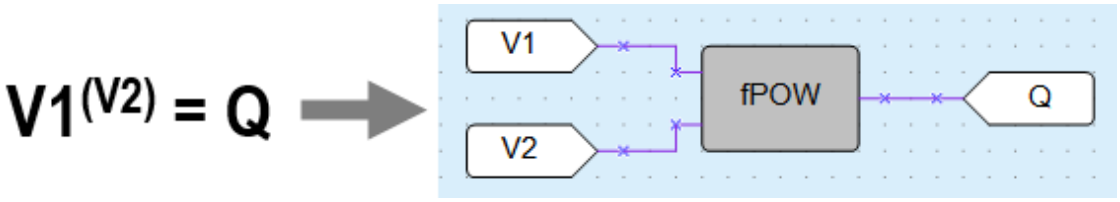
The function **MOD** operates with INT variables. The output **Q** is a remainder of the division of input values.

Example:



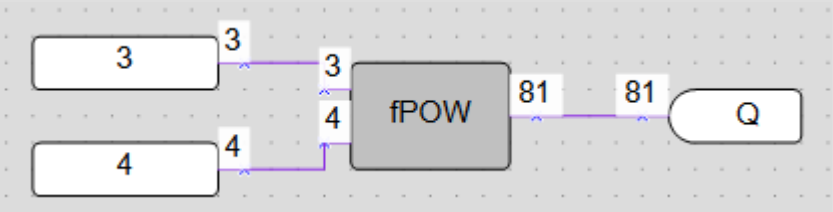
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.1.2.6 REAL-Power function (fPOW)

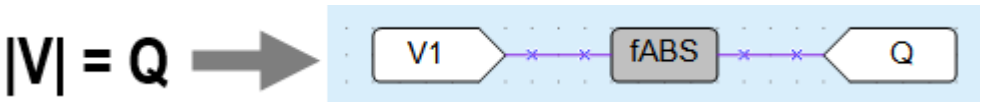


The function **fPOW** operates with REAL variables.
The output value **Q** is the value **I1** raised to the power of the value **I2**.

Example:

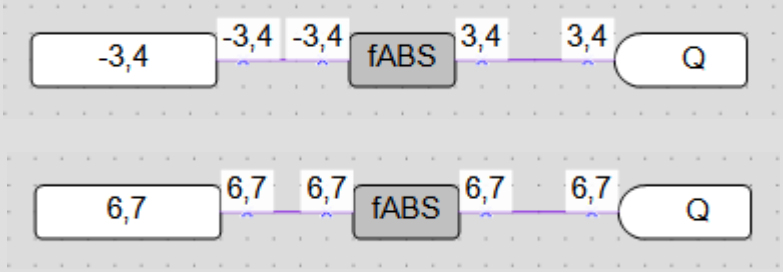


7.1.2.7 REAL-Absolute function (fABS)



The function **fABS** operates with REAL variables.
The output value **Q** is an absolute value of the input value.

Example:



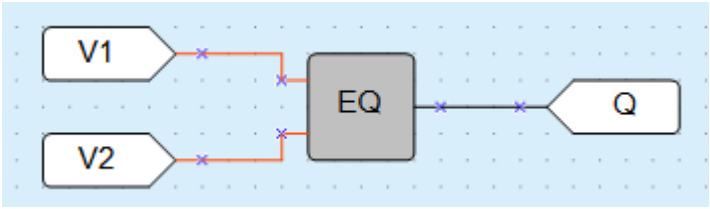
7.1.3 Relational operators

The relational operators are functions that test or define some kind of relation between two or more values.

– *Equal (EQ)*
– *Greater than (GT, fGT)*
– *Binary selection (SEL)*

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

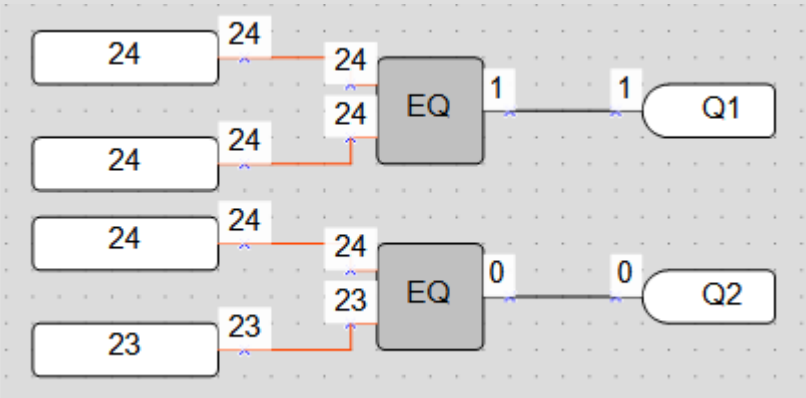
7.1.3.1 Equal (EQ)



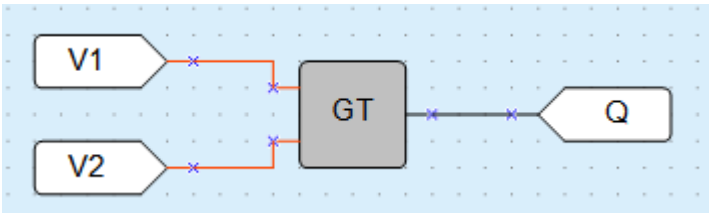
The function **EQ** operates with INT variables.
The output value **Q** is **True** if values **I1** and **I2** are equal.

– V1 = V2 → Q = 1;
– V1 > V2 → Q = 0;
– V1 < V2 → Q = 0.

Example:



7.1.3.2 Greater than (GT, fGT)

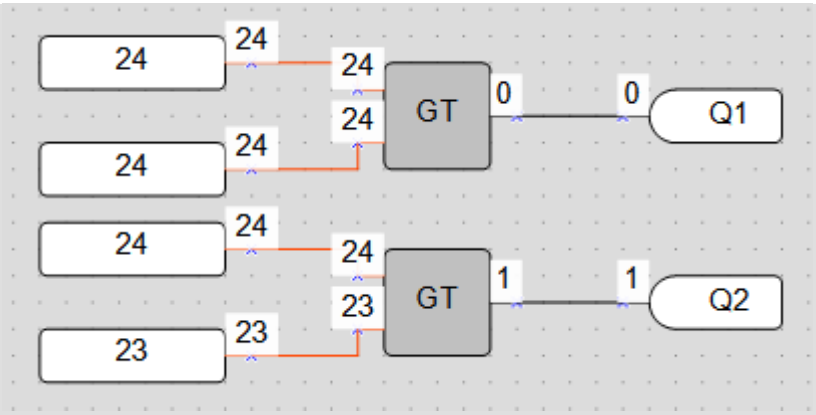


The function **GT** operates with INT variables, while the function **fGT** operates with REAL variables.
The output value **Q** is **True** if the value **I1** is greater than the value **I2**.

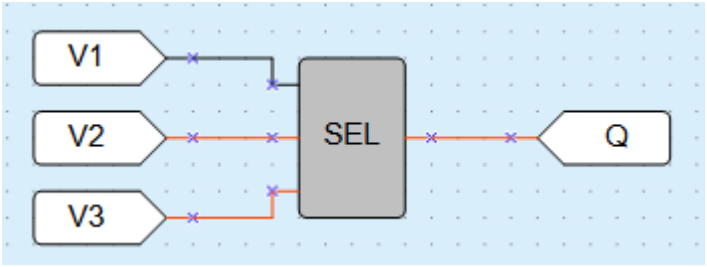
– V1 = V2 → Q = 0;
– V1 > V2 → Q = 1;
– V1 < V2 → Q = 0.

Example:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



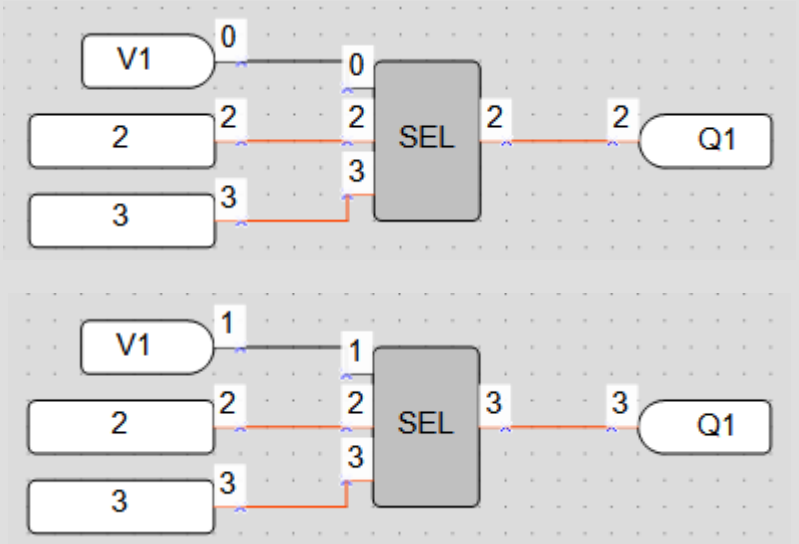
7.1.3.3 Binary selection (SEL, fSEL)



The function **SEL** operates with INT variables, the function **fSEL** operates with REAL variables. If the value **I1** is **False**, the output value **Q** is set to the value **I2**, else to the value **I3**.

– V1 = 0 → Q = V2;
– V1 = 1 → Q = V3.

Example:



7.1.4 Bitshift operators

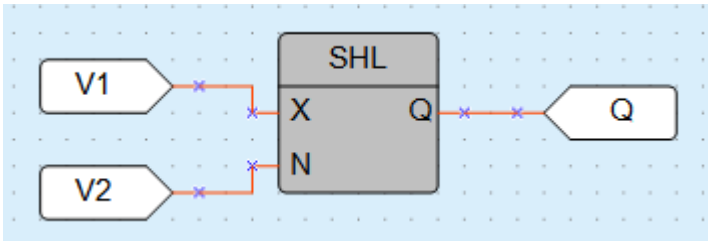
The bitshift operators treat a variable as a series of bits that can be moved (shifted) to the left or right.

– *Shift register left (SHL)*
– *Shift register right (SHR)*

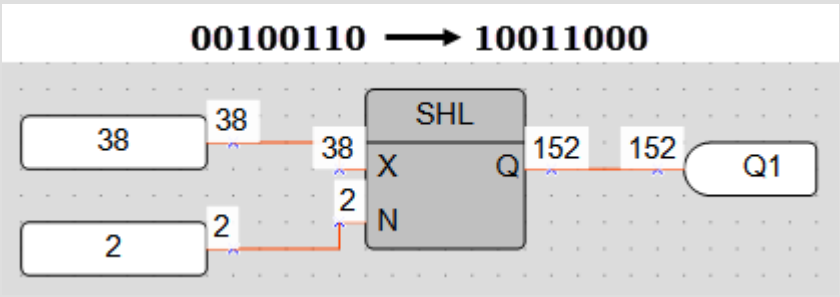
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.1.4.1 Shift register left (SHL)

The function **SHL** operates with INT variables. It is used to shift all bits of the operand **X** to the left by the **N** number of bits; vacated bits are zero-filled. The result is set to the output **Q**.

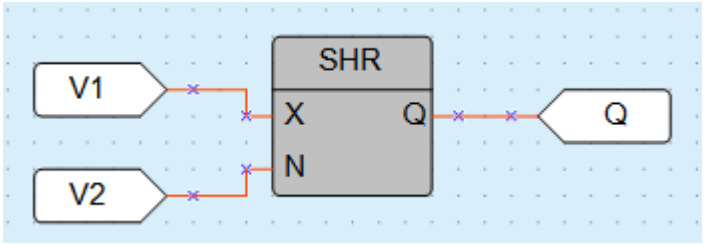


Example:
Left shift of the number 38 (decimal) = 00100110 (binary) by 2 bits

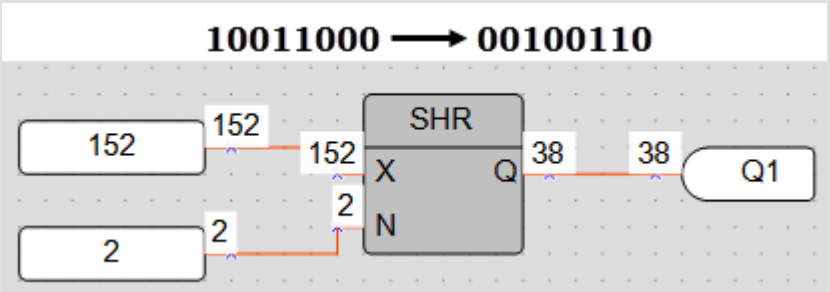


7.1.4.2 Shift register right (SHR)

The function **SHR** operates with INT variables. It is used to shift all bits of the operand **X** to the right by the **N** number of bits; vacated bits are zero-filled. The result is set to the output **Q**.



Example:
Right shift of the number 152 (decimal) = 10011000 (binary) by 2 bits



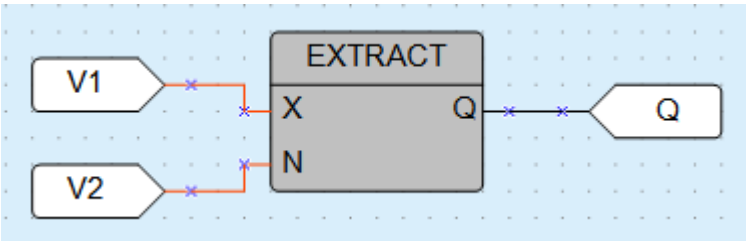
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.1.5 Bit operators

The bit operator treats a value as a series of bits to perform operations on one or more individual bits of an operand.

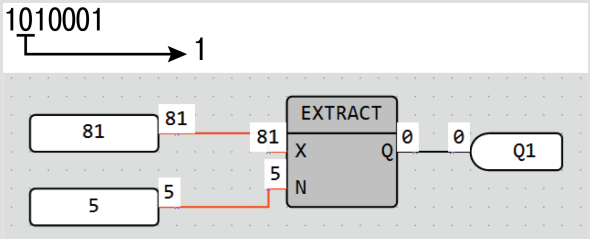
– *Read single bit (EXTRACT)*
– *Set single bit (PUTBIT)*
– *Decoder (DC32)*
– *Encoder (CD32)*

7.1.5.1 Read single bit(EXTRACT)

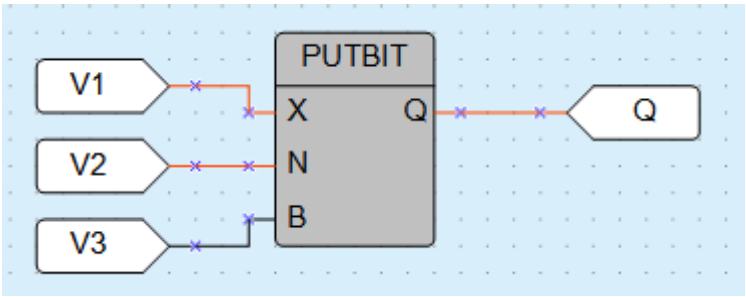


The output value **Q** (BOOL) of the function **EXTRACT** is the value of bit **N** (INT) in the operand **X** (INT). The bit numbering is zero-based.

Example:
Reading of the 5th bit from the number 81 (decimal) = 1010001 (binary):



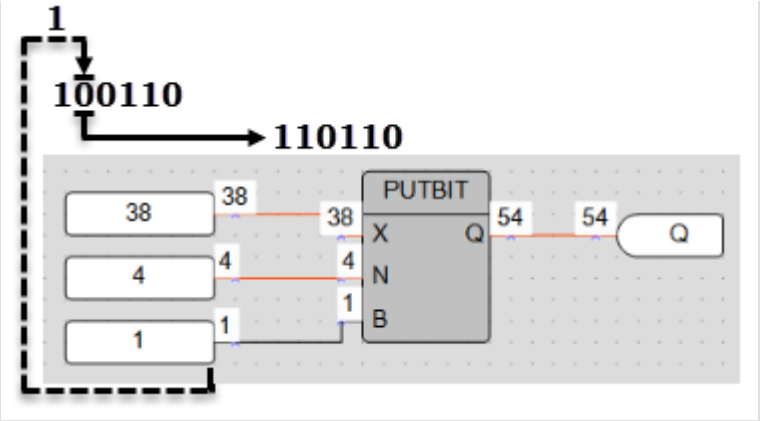
7.1.5.2 Set single bit (PUTBIT)



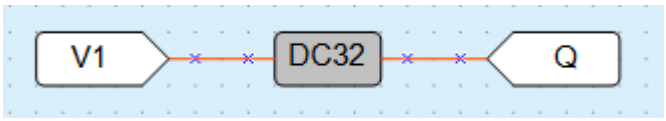
This output value **Q** (INT) is the value of the operand **X** (INT) where the bit **N** (INT) is set to the value at the input **B** (BOOL). The bit numbering is zero-based.

Example:
Setting of the 4th bit to 1 in the number 38 (decimal) = 100110 (binary):

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



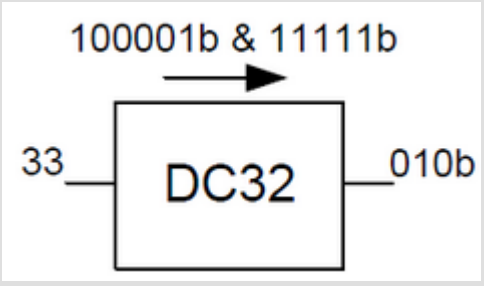
7.1.5.3 Decoder (DC32)



The decoder converts a binary code at the input to a position code at the output.
Decoding is carried out bitwise by the logical operation **AND** with the operand 0x1F (11111b).
Truth table:

Binary code					Position code									
5	4	3	2	1	32	31	…	6	5	4	3	2	1
0	0	0	0	0	0	0		0	0	0	0	0	1
0	0	0	0	1	0	0		0	0	0	0	1	0
0	0	0	1	0	0	0		0	0	0	1	0	0
0	0	0	1	1	0	0		0	0	1	0	0	0
0	0	1	0	0	0	0		0	1	0	0	0	0
…					…			…					
1	1	1	0	1	0	0		0	0	0	0	0	0
1	1	1	1	0	0	1		0	0	0	0	0	0
1	1	1	1	1	1	0		0	0	0	0	0	0

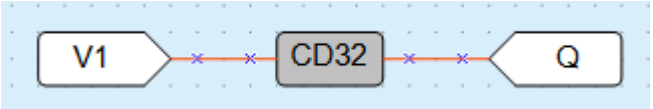
Example:



7.1.5.4 Encoder

Encoder (CD32) is used to perform the operation of converting the positional code from the input into binary code at the output.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



If there is more than one “1” bits in the position code, the encoder operates only with the most significant “1” bit.
Truth table:

Binary code					Position code									
5	4	3	2	1	32	31	…	6	5	4	3	2	1
0	0	0	0	0	0	0		0	0	0	0	0	1
0	0	0	0	1	0	0		0	0	0	0	1	0
0	0	0	1	0	0	0		0	0	0	1	0	0
0	0	0	1	1	0	0		0	0	1	0	0	0
0	0	1	0	0	0	0		0	1	0	0	0	0
…					…			…					
1	1	1	0	1	0	0		0	0	0	0	0	0
1	1	1	1	0	0	1		0	0	0	0	0	0
1	1	1	1	1	1	0		0	0	0	0	0	0

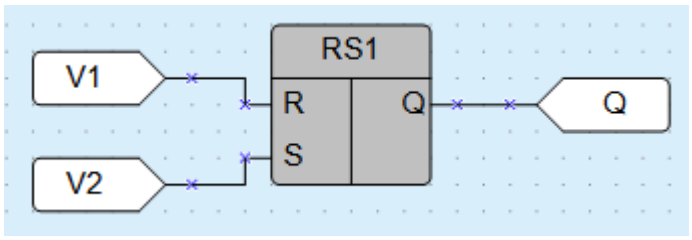
7.2 Function blocks

- *Triggers*
- *Timers*
- *Generators*
- *Counters*
- *Controllers*

7.2.1 Triggers

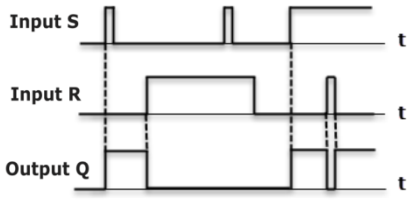
- *RS trigger reset dominant (RS)*
- *SR trigger set dominant (SR)*
- *Rising edge (RTRIG)*
- *Falling edge (FTRIG)*
- *D-trigger (DTRIG)*

7.2.1.1 RS trigger reset dominant

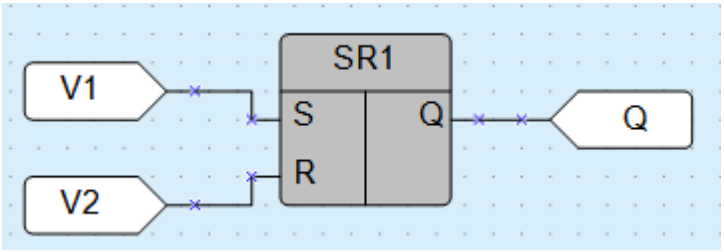


The output **Q** is **True** with a rising edge at the input **S** (Set) and **False** with a rising edge at the input **R** (Reset). The input **R** has higher priority.

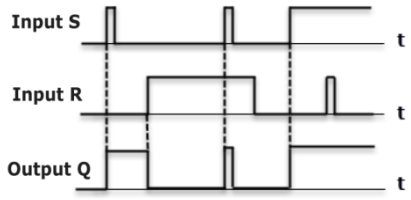
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



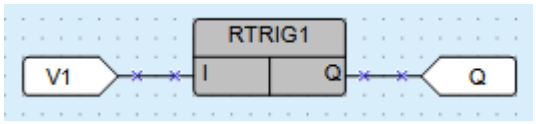
7.2.1.2 SR trigger set dominant



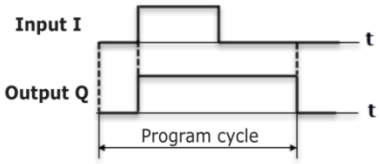
The output **Q** is **True** with a rising edge at the input **S** (Set) and **False** with a rising edge at the input **R** (Reset). The input **S** has higher priority.



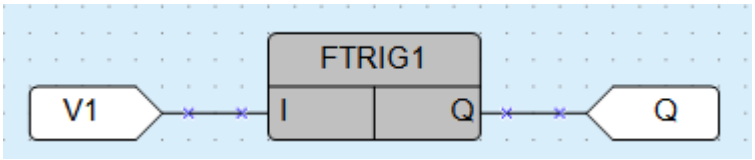
7.2.1.3 Rising edge (RTRIG)



Detector for a rising edge appears.
The output **Q** remains **False** until a rising edge at the input **I**. As soon as the input **I** becomes **True**, the output becomes **True** and remains **True** for one program cycle.

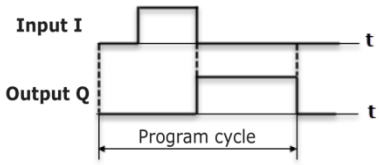


7.2.1.4 Falling edge (FTRIG)

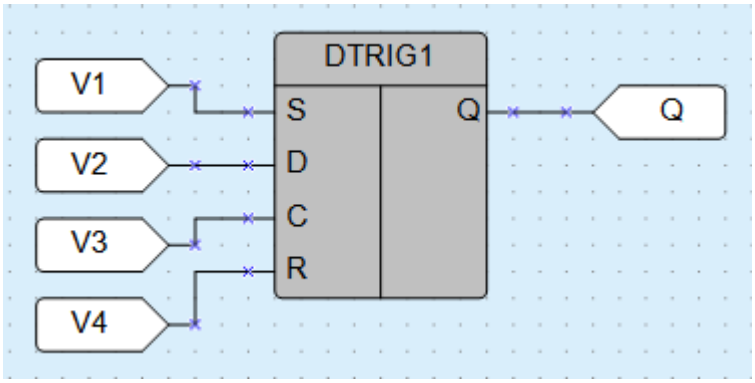


Detector for a falling edge.
The output **Q** remains **False** until a falling edge at the input **I**. As soon as the input **I** becomes **False**, the output becomes **True** and remains **True** for one program cycle.

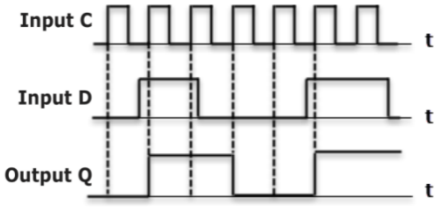
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



7.2.1.5 D-trigger (DTRIG)



D-trigger generates a pulse at the output **Q** with the pulse duration specified at the input **D** and synchronized with the clock pulse at the input **C**.
If the input **D** is **True**, the output **Q** becomes **True** with a rising edge of the clock pulse at the input **C**.
If the input **D** is **False**, the output **Q** becomes **False** with a rising edge of the clock pulse at the input **C**.



The output **Q** can be forced to set to **True** with a rising edge at the input **S** (Set) and forced to reset to **False** with a rising edge at the input **R** (Reset), regardless of the states of the inputs **C** and **D**. The input **R** has higher priority.

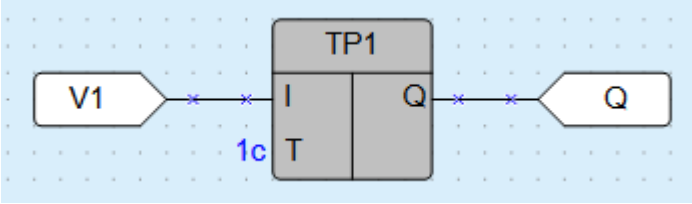
7.2.2 Timers

- *Pulse (TP)*
- *ON-delay timer (TON)*
- *OFF-delay timer (TOF)*
- *Timer (CLOCK)*
- *Weekly timer (CLOCKWEEK)*

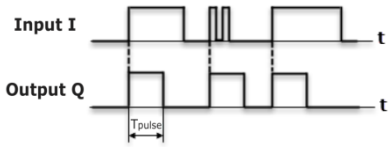
7.2.2.1 Pulse (TP)

The Pulse Timer (TP) is used to generate an output pulse of a specified duration. The output **Q** becomes **True** with a rising edge at the input **I** for the time specified at the input **T**. During this time, the output **Q** remains **True** regardless of the signal change at the input **I**. The output **Q** is reset to **False** with the end of pulse.

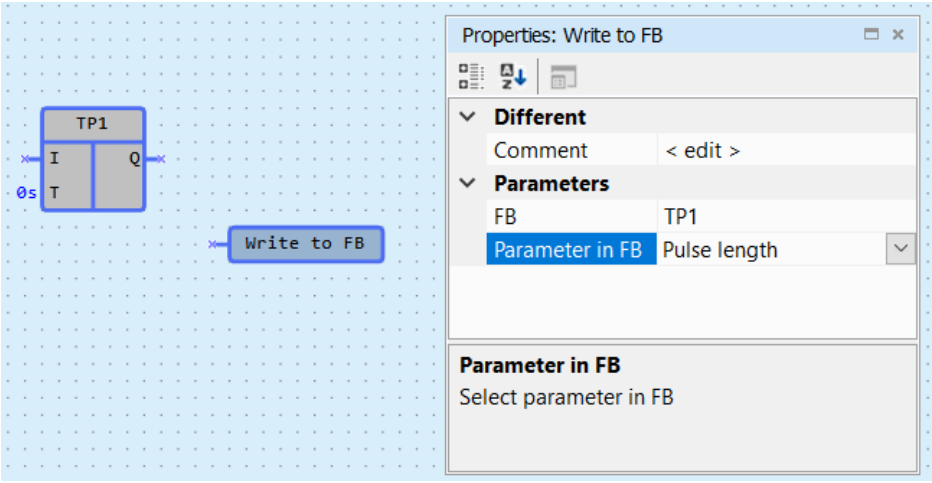
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



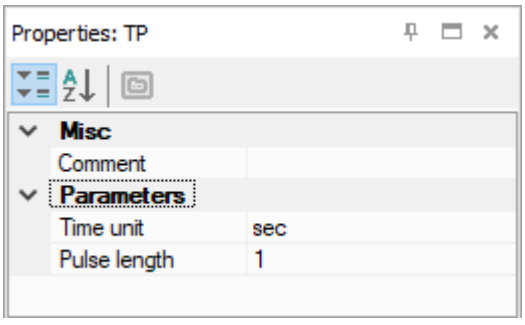
The operation of the pulse timer is illustrated by the diagram shown in the figure below.



Time range: 0…4,147,200,000 ms or 48 days.

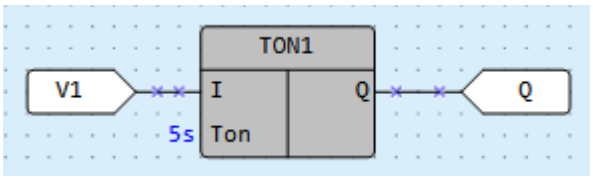


The values of the function block's parameters can be set using a Write to FB block, as shown in the figure above, or configured in Property Box:



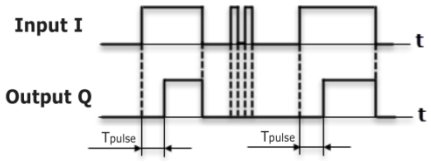
7.2.2.2 ON-delay timer (TON)

The On-Delay Timer (TON) is used to delay the transmission of a signal. The output **Q** = **False** if the input **I** = **False**. The delay time specified at the input **TON** starts with a rising edge at the input **I**. When the time **TON** is elapsed, the output **Q** becomes **True** and remains **False** until a falling edge appears at the input **I**. Input changes shorter than **TON** are ignored.

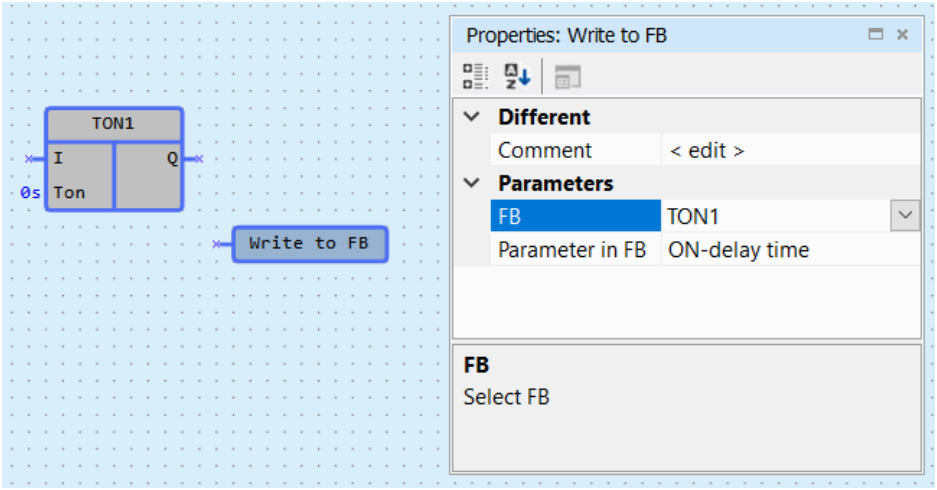


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

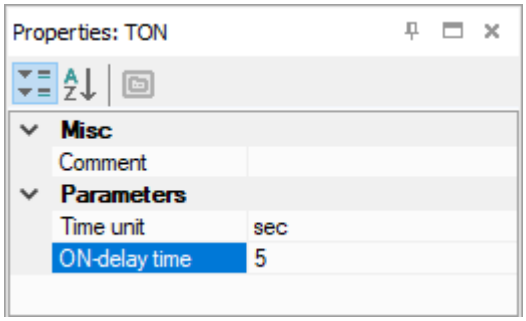
The operation of the timer is explained by the diagram shown in the figure below.



Time range: 0…4,147,200,000 ms or 48 days.

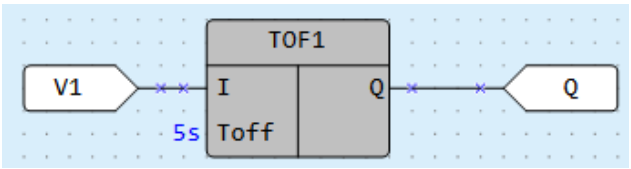


The values of the function block's parameters can be set using a Write to FB block, as shown in the figure above, or configured in Property Box:

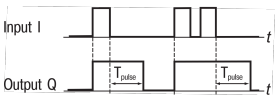


7.2.2.3 OFF-delay timer (TOF)

The Off-Delay Timer (TOF) is used to delay turning off the output. The output **Q** = **False** if the input **I** = **False**. The delay time specified at the input **TOFF** starts with a falling edge at the input **I**. When the time **TOFF** is elapsed, the output **Q** becomes **False** and remains **False** until a rising edge appears at the input **I**. Input changes shorter than **TOFF** are ignored.

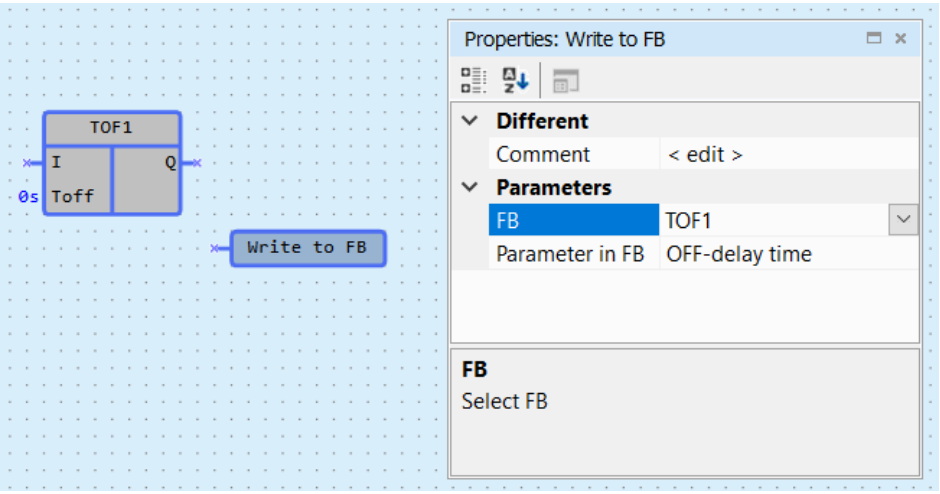


The operation of the timer is explained by the diagram shown in the figure below.

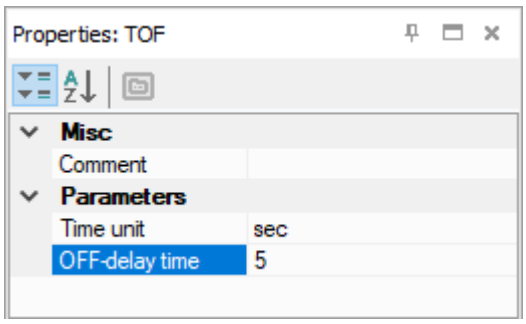


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

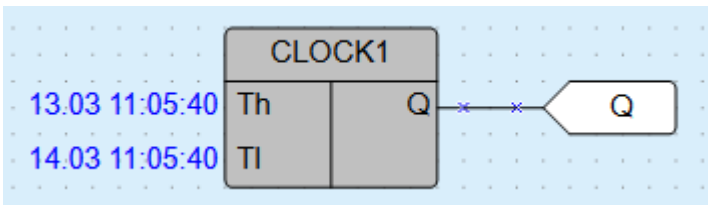
Time range: 0…4,147,200,000 ms or 48 days.



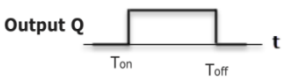
The values of the function block's parameters can be set using a Write to FB block, as shown in the figure above, or configured in Property Box:



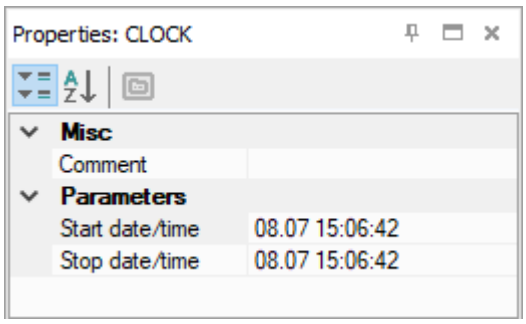
7.2.2.4 Timer (CLOCK)



The block **CLOCK** is an interval timer controlled by a real-time clock.

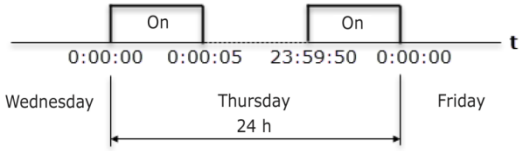


The times **TH** and **TL** can be set in Property Box.

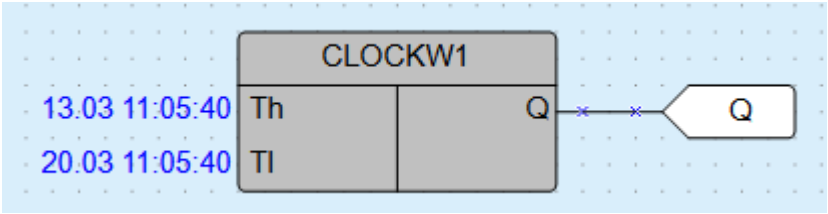


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

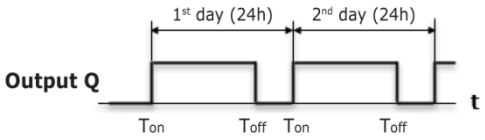
Time range: from 0.00 seconds to 24 hours.
If **TH** < **TL**, the state of the output **Q** is as follows:



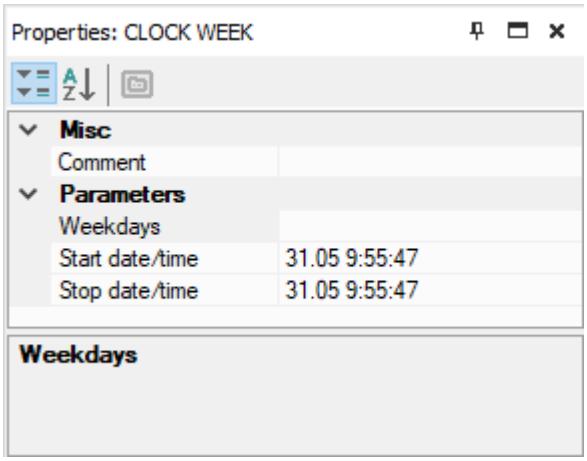
7.2.2.5 Weekly timer (CLOCKWEEK)



The block **CLOCKWEEK** is an interval timer with the parameter ***Weekdays*** controlled by a real-time clock.



The times **TH** and **TL** can be set in Property Box.



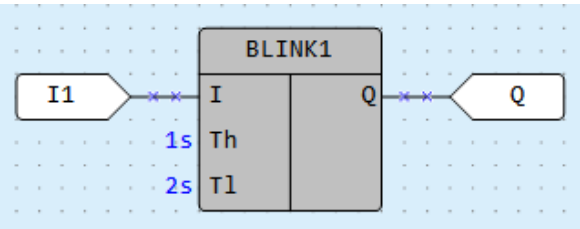
Time range: from 0.00 seconds to 24 hours.

7.2.3 Generators

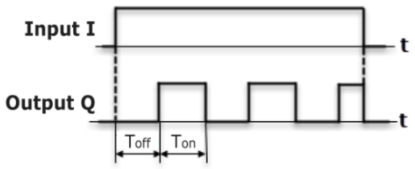
– *Pulse generator (BLINK)*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

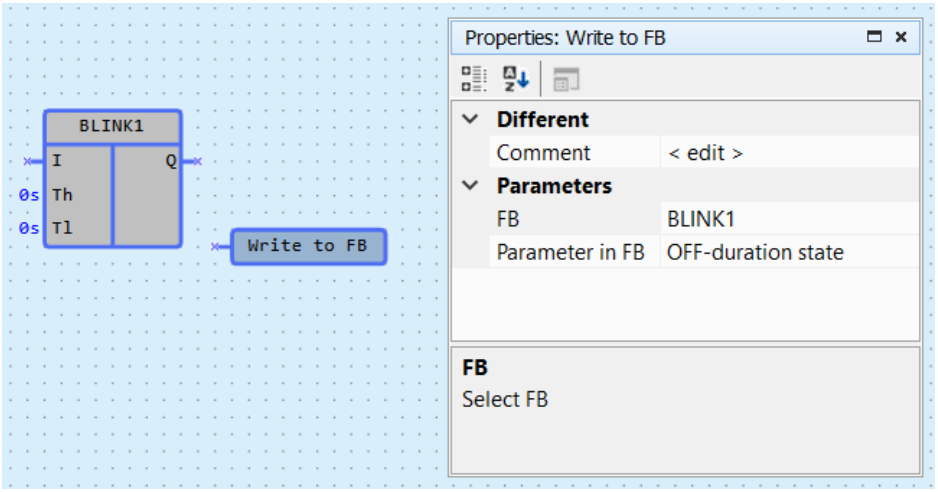
7.2.3.1 Pulse generator (BLINK)



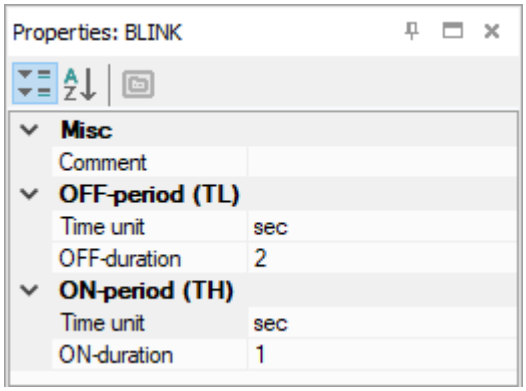
If the input **I** becomes **True**, the block **BLINK** generates a square wave on the output **Q** with a period of **TH** + **TL**, starting with an interval of the duration of **TL**, followed by a pulse of the duration of **TH**. It continues that way until the input **I** is **False**.



Time range: 0…4,233,600,000 milliseconds or 49 days.



The values of the function block's parameters can be set using a Write to FB block, as shown in the figure above, or configured in Property Box:



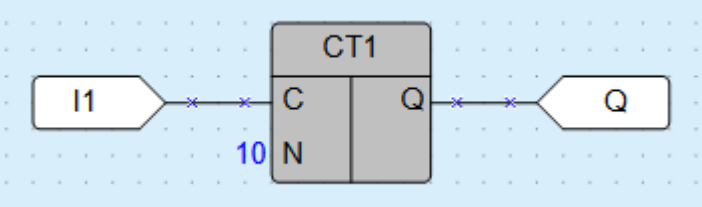
7.2.4 Counters

– *Threshold counter with self-reset (CT)*
– *Universal counter (CTN)*

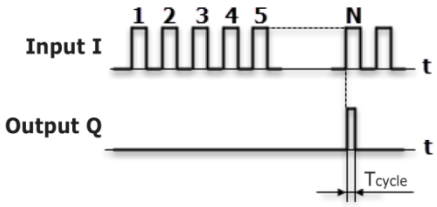
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– *Threshold counter (CTU)*

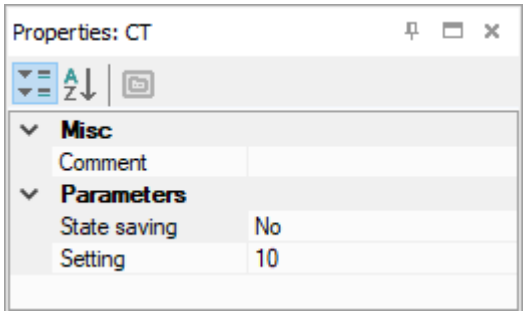
7.2.4.1 Threshold counter with self-reset (CT)



The output **Q** is of type BOOL. If the number of pulses counted on the input **C** exceeds the threshold (***Setting***) specified at the input **N**, the output **Q** becomes **True** and remains for one program cycle. The operation of the counter is explained in the diagram below.

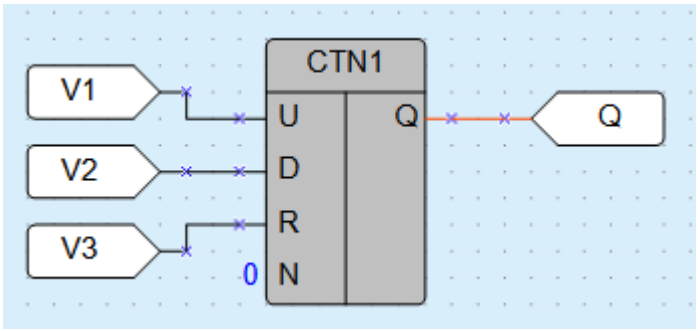


The parameters ***Setting*** and ***State saving*** can be set in Property Box.



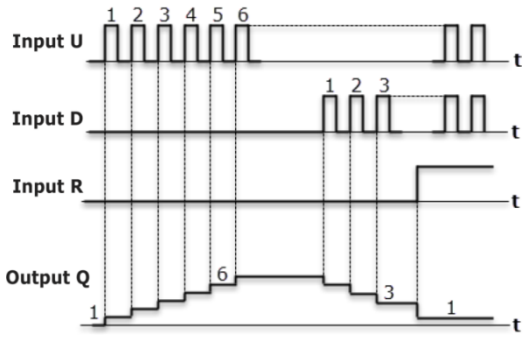
Threshold range: 0…65535.

7.2.4.2 Universal counter (CTN)

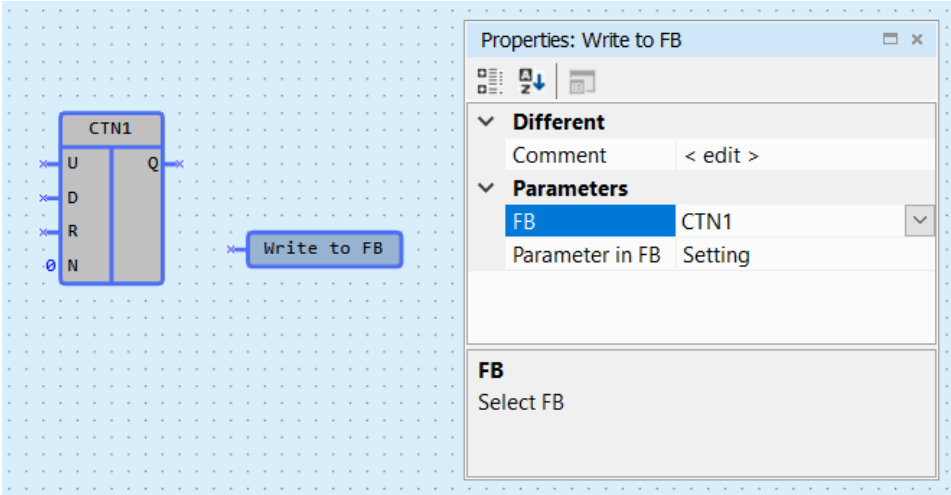


The output **Q** is of type INT. A rising edge at the input **U** increases the value at the output **Q** by 1. A rising edge at the input **D** decreases the value at the output **Q** by 1.
If the input **R** = **True**, the output **Q** gains the value ***Setting*** from the input **N**.

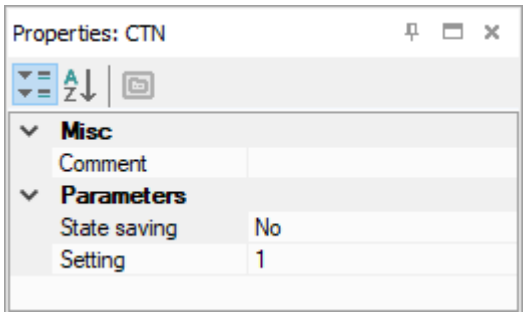
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The input **U** has higher priority than the input **D**.
Setting range: 0…65,535.

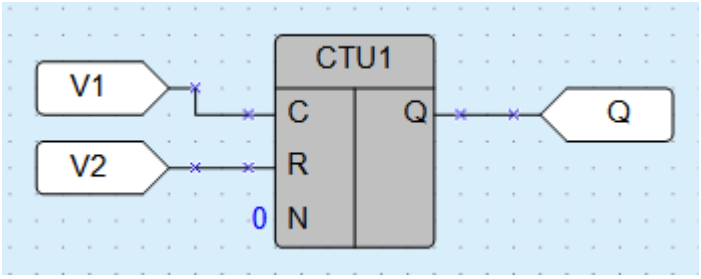


The values of the function block's parameters can be set using a Write to FB block, as shown in the figure above, or configured in Property Box:



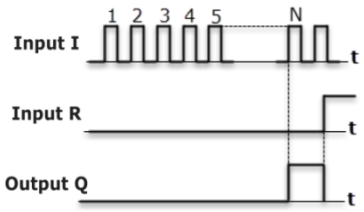
If ***State saving*** = ***Yes***, the state of the counter is permanently stored in the non-volatile memory.

7.2.4.3 Threshold counter (CTU)

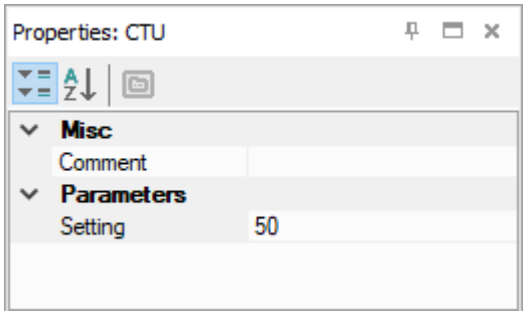


Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The output **Q** is of type Boolean. If the number of pulses counted on the input C exceeds the threshold (***Setting***) specified at the input **N**, the output **Q** becomes **True** and remains **True** until a rising edge at the input **R**. The input R has higher priority than the input **C**.
The operation of the counter is explained in the diagram below.



The parameter ***Setting*** can be set in Property Box.

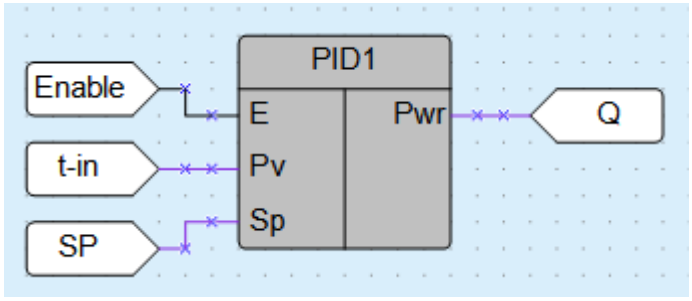


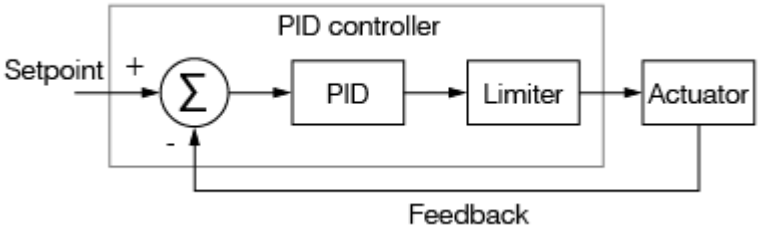
Threshold range: 0…65535.

7.2.5 Analog

– *PID-controller (PID) for fist generation devices*
– *PID-controller (PID_REG) for second generation devices*

7.2.5.1 PID-controller (PID) for first generation devices





The function block **PID** is used to implement of the proportional-integral-derivative control.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Table 7.1 PID block inputs/outputs

Name	Type	I/O	Description	Values
E	BOOL	I	Enable control (0 = Off, 1 = On). If disabled, the parameter ***Pwr*** takes the value of the parameter ***Output safe state***.	0 – Off 1 – On
Pv	REAL	I	Process value	
Sp	REAL	I	Setpoint	
Pwr	REAL	O	Output power, %	0…100

Table 7.2 PID block parameters

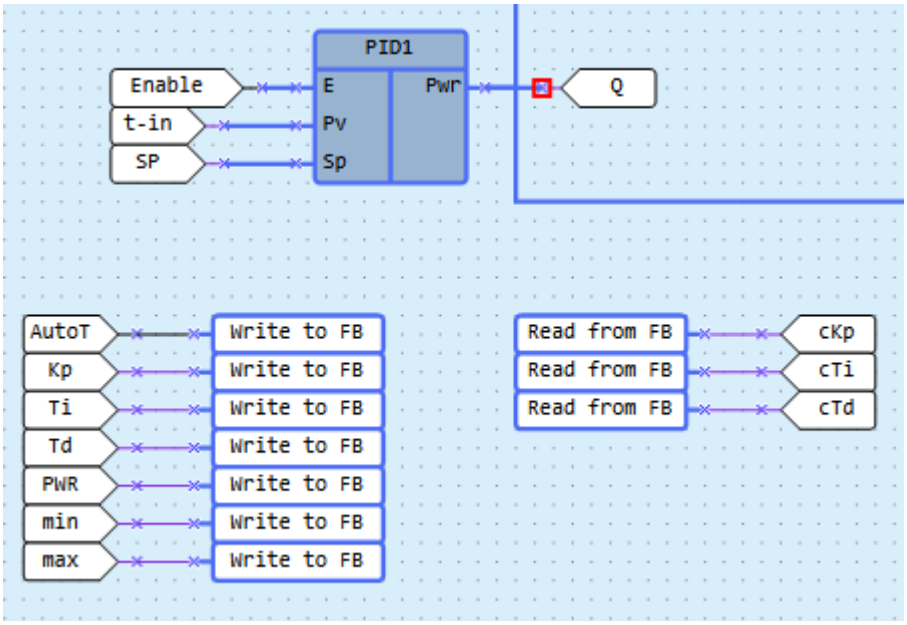
Name	Type	Description	Values	Access		
				Property Box	Write ToFB	Read FromFB
Control mode	BOOL	0 – Heating 1 – Cooling	0/1	X	X	
Output safe state	REAL	Output value when control is disabled, %	0…100	X	X	
Kp	REAL	Proportional gain, multiplication factor for proportional control	0…100	X	X	
Ti (s)	REAL	Integral time, time constant for integral control in seconds	–3,402823E +38… 3,402823E+38	X	X	
Td (s)	REAL	Derivative time, time constant for derivative control in seconds	–3,402823E +38… 3,402823E+38	X	X	
Output max.	REAL	Output upper limit, % (default 80 %)	0…100	X	X	
Output min.	REAL	Output lower limit, % (default 20 %)	0…100	X	X	
Start AT	BOOL	0 – stop auto-tuning 1 – start auto-tuning	0/1		X	
AT completed	BOOL	Flag: 0 – auto-tuning stopped 1 – auto-tuning started	0/1			X
Kp calculated	REAL	Calculated proportional gain	–3,402823E +38… 3,402823E+38			X
Ti calculated	REAL	Calculated integral time	–3,402823E +38… 3,402823E+38			X
Td calculated	REAL	Calculated derivative time	–3,402823E +38… 3,402823E+38			X

Tuning of a control loop is the adjustment of its control parameters (***Kp***, ***Ti***, ***Td***) to the optimal values for the desired control response.

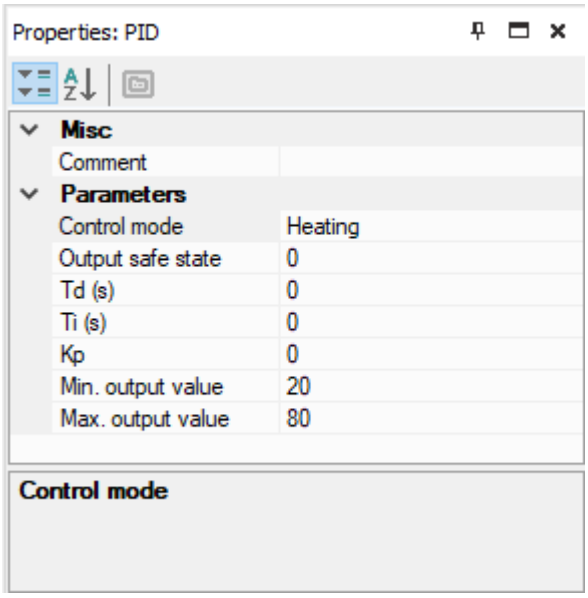
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Auto-tuning

Programmable loop tuning can be performed using the blocks *WriteToFB* and *ReadFromFB*.
To write the parameters, use the block WriteToFB or Property Box.
To read the parameters, use the block ReadFromFB.



To use auto-tuning, add the block WriteToFB to the circuit program and set the reference to the parameter ***Start AT*** of the PID block.
To start the auto-tuning, enable control (***E*** = **1**) and set the parameter ***Start AT*** = **1**.



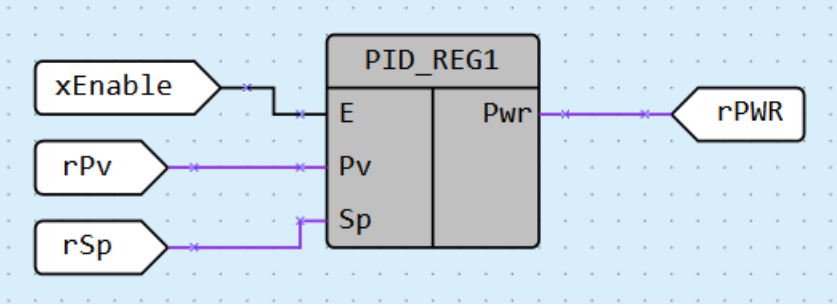
Upon completion of the auto-tuning, the new values of the parameters ***Kp***, ***Ti*** and ***Td*** are calculated and the flag ***AT completed*** becomes **1**.
If ***Start AT*** = **0**, the flag ***AT completed*** = **0** as well.
If you set ***Start AT*** = **0** before the completion of auto-tuning, the auto-tuning is stopped, the flag ***AT completed*** becomes **0** and no new coefficients are calculated.
During the auto-tuning, a test signal limited by parameters ***Output max.*** and ***Output min.*** is applied to the output **Pwr**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
*If the maximum gain is not sufficient to reach the setpoint, the auto-tuning cannot be completed and will continue until it is stopped with **Start AT** = 0.*

7.2.5.2 PID-controller (PID_REG) for second generation devices

NOTE
The PID REG is not available for PR103 devices.



The PID controller is used to implement the proportional-integral-derivative (PID) control law. The controller's output power is calculated using the formula:

$$MV(t) = \frac{1}{X_p} \cdot \left(e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)\, d\tau + T_d \cdot \frac{de(t)}{dt} \right)$$

, where
Xp – proportional band (measurement units)
Ti – integral time constant [s]
Td – derivative time constant [s]
e(t) – error (deviation)
MV(t) – PID output

Block Inputs

Name	Type	Description	Values
E	BOOL	Controller enable. When disabled, the output is set to the value specified in the 'Output Power in Off State' property	0 – Off; 1 – On
Pv	REAL	Current value of the controlled variable	–9999...9999*
Sp	REAL	Setpoint value of the controlled variable	

*If a value outside this range is applied to the input, the regulator's Pwr output will be reset to the value specified in the 'Output Power in Off State' parameter.

Block Outputs

Table 7.3 Block Outputs

Name	Type	Description	Values
Pwr	REAL	Controller output power	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Inputs Available via Write to FB

Table 7.4 Inputs Available via Write to FB

Name	Type	Description	Values
Control mode	BOOL	Controller operating modes: – 'Heater' mode: used for controlling actuators that increase the controlled variable; – 'Cooler' mode: used for controlling actuators that decrease the controlled variable	0 – Heater; 1 – Cooler
Start AT	BOOL	Controller auto-tuning control	0 – Auto-tuning off 1 – Auto-tuning started
Proportional band	REAL	PID controller proportional band, measurement units	0..9999
Integral time constant	REAL	Integral time constant, sec	0..4000
Derivative time constant	REAL	Derivative time constant, sec	0..4000
Output power in OFF state	REAL	Pwr output state when input E is 0	-9999..9999
Output max.	REAL	Upper limit of output power	-9999..9999
Output min.	REAL	Lower limit of output power	-9999..9999
Dead zone	REAL	Dead band around setpoint/2 where controller output remains unchanged, measurement units	
Output type	BOOL	Controller output signal selection	0 – Power 1 – Discrete CRV
Full travel length	REAL	Time for valve stem to move from fully closed to fully open position, sec	5..999**
Minimum pulse time	REAL	Minimum control pulse duration for the valve, sec	0.001..100
Backlash compensa-tion time	REAL	Valve backlash sampling time, sec	0..10
Setpoint change rate	REAL	Setpoint rate of change, measurement units/sec	0 – Function disabled 0...9999
Manual mode of operation of the regulator	BOOL	Flag for switching controller to manual mode	0 – Automatic mode 1 – Manual mode
Manual output control "More"	BOOL	In manual mode: command to send "Increase" signal to block output	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Name	Type	Description	Values
Manual output control "Less"	BOOL	In manual mode: command to send "Decrease" signal to block output	
Initial value of controlled variable for PID-autotuning (Pv_0)	REAL	Controlled variable value that will be at input Pv when zero power is applied (object at rest). This parameter is required for correct auto-tuning	-9999..9999

**To apply a new full stroke time for the valve, restart the block by toggling the signal at input E.

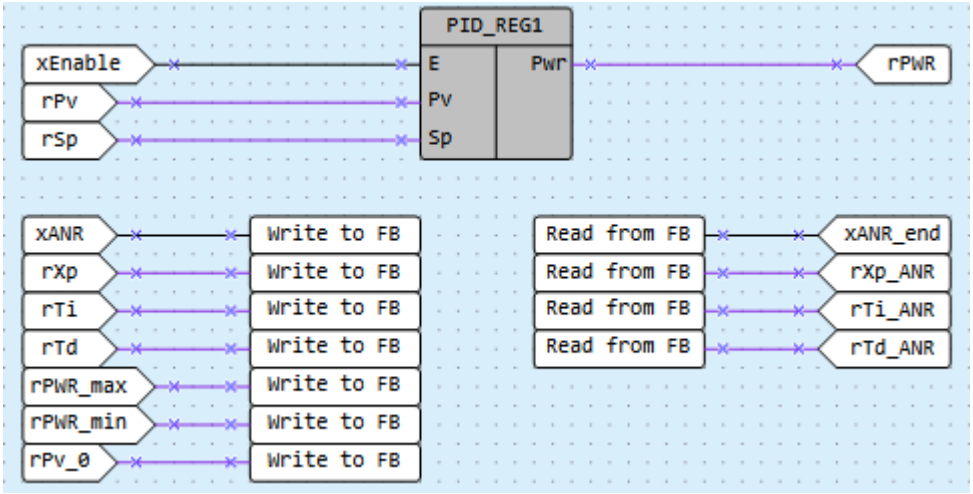
Outputs Available via Read from FB

Table 7.5 Outputs Available via Read from FB

Name	Type	Description	Values
AT completed	BOOL	Auto-tuning completion flag	0 – AT not completed 1 – AT completed
Calculated proportional band	REAL	Proportional band calculated after auto-tuning, measurement units	0..9999
Calculated integral time constant	REAL	Integral time constant calculated after auto-tuning, sec	0..4000
Calculated differential time constant	REAL	Derivative time constant calculated after auto-tuning, sec	0..4000
"More"	BOOL	Valve open signal	
"Less"	BOOL	Valve close signal	

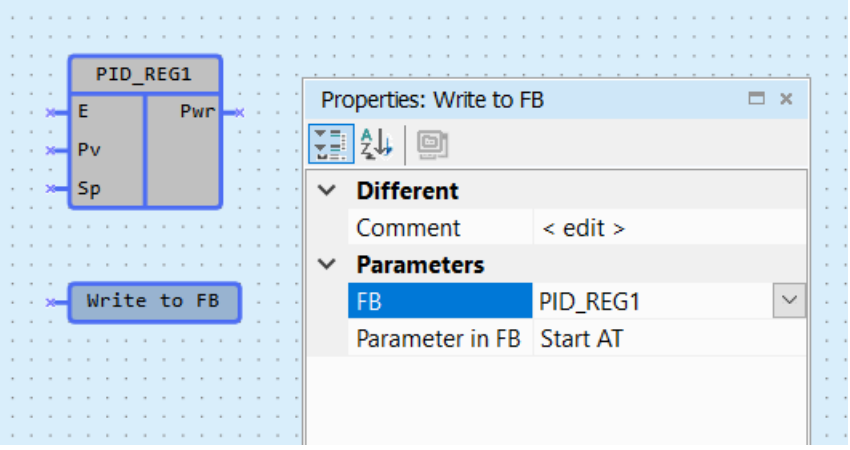
Auto-Tuning

PID controller auto-tuning is performed using *Read from and Write to FB* blocks.



To start auto-tuning, add a **Write to FB** block and bind it to the PID controller's **Start Auto-Tuning** variable.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



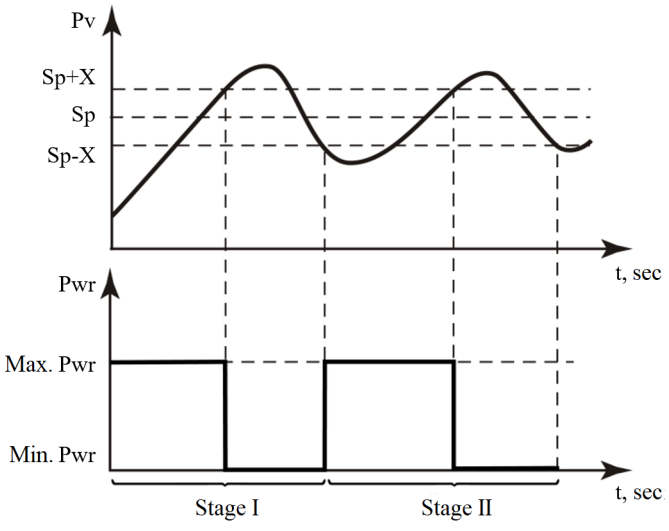
Values for other FB parameters can be set using the **Write to FB** block, as shown in the figure above, or configured in the properties panel.
Use **Read from FB** blocks to read the values of parameters **Calculated proportional band**, **Calculated integral time constant**, **Calculated differential time constant**, and **AT completed**.
To start auto-tuning, apply a logical "1" to input **E**.
Upon completion of the auto-tuning process, new coefficient values are available for reading: **Calculated proportional band**, **Calculated integral time constant**, and **Calculated differential time constant**. The **AT completed** parameter is set to logical one.

CAUTION
The auto-tuning completion flag remains at logical one for one cycle only.

If the **Start AT** input is reset to logical zero before tuning completion, the process stops, the completion flag is not set, and new coefficient values are not calculated.

CAUTION
For correct auto-tuning, the **Initial value of controlled variable for PID-autotuning** parameter must be set. It should equal the controlled variable value corresponding to zero actuation (object at rest).

During the tuning process, test power limited by the **Output max.** and **Output min.** parameter values is applied to the PID controller output.
Auto-tuning sequence for 'Heater' mode:



1. When the current value is below the setpoint by more than *X = 0.04*(Sp-Pv_0)*, maximum power (according to settings) is applied to the block output.
2. Once the current value exceeds the setpoint by more than *X = 0.04*(Sp-Pv_0)*, minimum power is applied to the block output.
3. Steps 1 and 2 are repeated once more.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4. The calculated PID controller parameters are sent to the corresponding outputs, and the completion flag is set (for one cycle).

If the maximum power setting is insufficient to reach the setpoint, the auto-tuning process will not complete until manually reset.

Working with digital CRV

To work with a discrete control valve, set the value to **"Control valve (digital)"** in the corresponding parameter (Output Type), or, when working with this parameter via **Write to FB**, write the value "1" to it.
For correct operation, the valve parameters must be set: **Full travel time**, **Minimum pulse length**, and **Backlash compensation time**.
Access to control signals (Increase/Decrease) is through the corresponding outputs of the controller's **Read from FB** block.
In this mode, the **Pwr** output displays the virtual valve position (which may differ from the actual position).
When logical "1" is removed from input E, the valve remains in its current position.
For quick setup, we recommend using ready-made macros from the Component Manager:

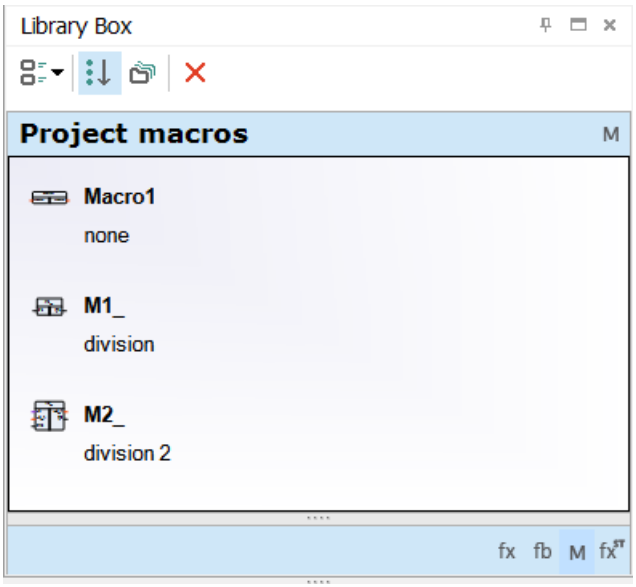
– For power control — **PID AT_** block;
– Macro with full functionality in power control mode — **PID AT_F_**;
– For discrete CRV control – **PID Valve_** block;
– Macro with full functionality in discrete CRV control mode — **PID Valve_F_**.

Manual Mode

Operation in power control mode: when switching the controller to manual mode (the **Manual mode of operation of the regulator** input is set to 1), the calculated power value is fixed.
Operation in discrete CRV control mode: when switching the controller to manual mode (the **Manual mode of operation of the regulator** input is set to 1), the calculated valve position is fixed, and the manual valve open/close control inputs become available.

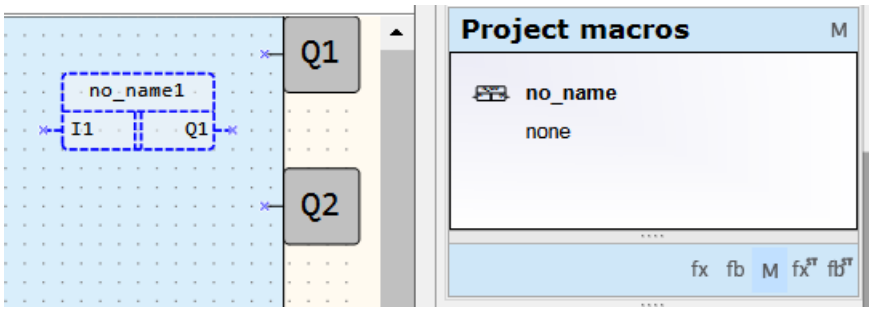
7.3 Project macros

Project Macros section contains macros *created by the user* or downloaded from Online Database using *Component Manager*.



To add a macro to a project, drag-and-drop the macro from the *Library Box* to the workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



To open the project macro in the separate workspace for editing, select it in the workspace or in the library and use the item **Edit macro** in the macro context menu.

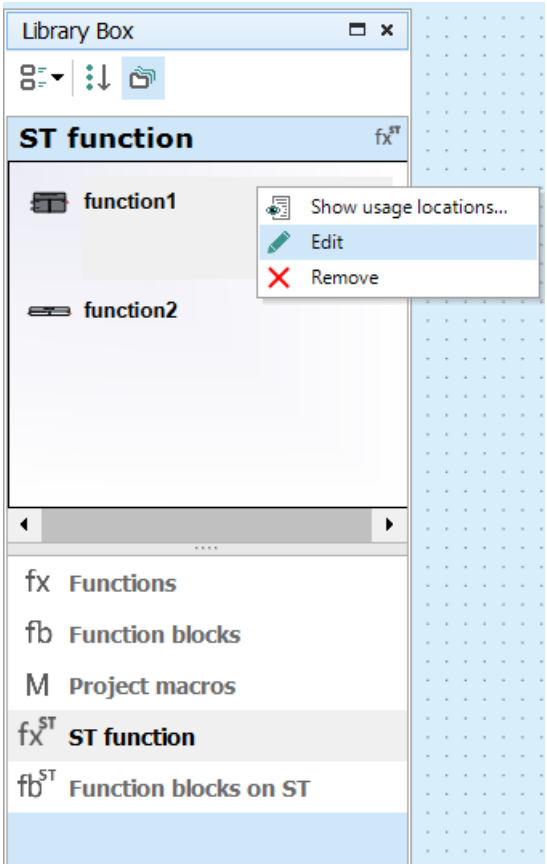
To remove the macro from the Library Box, select the macro and click the ✕ icon in the panel toolbar.

For details about macros creation, development and handling see section *Macro development*.

7.4 ST functions

NOTE
User can create functions in ST language for PR100(M02), PR102, PR200, PR103, PR205, PR225 and SMI200 devices.

If you have *created ST functions* in your project, they will be available in the Library Box.

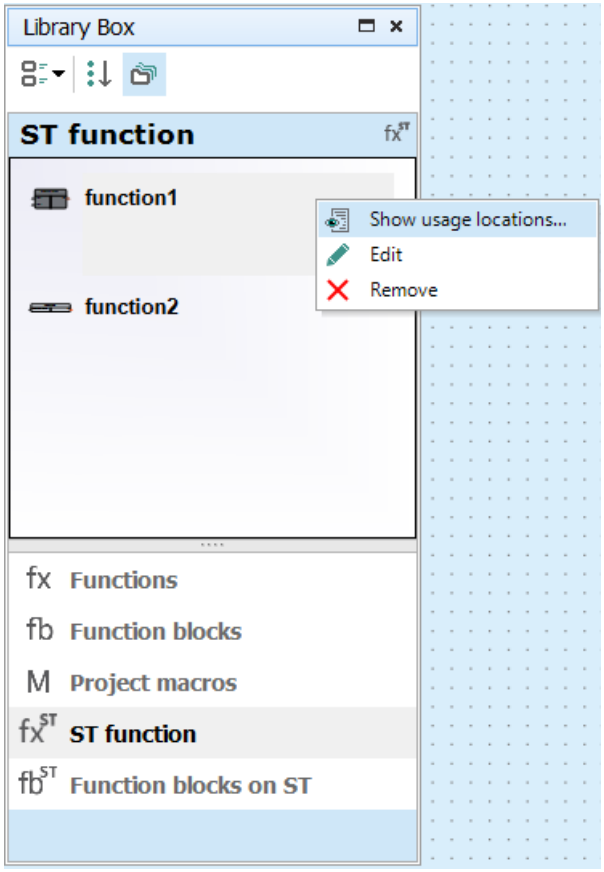


Usage locations

To view all places where the function is used:

1. Right-click on the function name
2. Select item ***Show usage location…***.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The **Function usage locations** panel will open at the bottom of the window, displaying where the function is used in the diagram and in the function editor.



3. Right-click on the line that indicates where the function is used.
4. Select item **To usage location**.



The focus will shift to where the function is used in the diagram or in the function editor.

NOTE
Double-click leads to the same result.

If the places where functions are used have changed while working with the program, you should update the **Function usage locations** panel:

1. Right-click on any line of the panel.

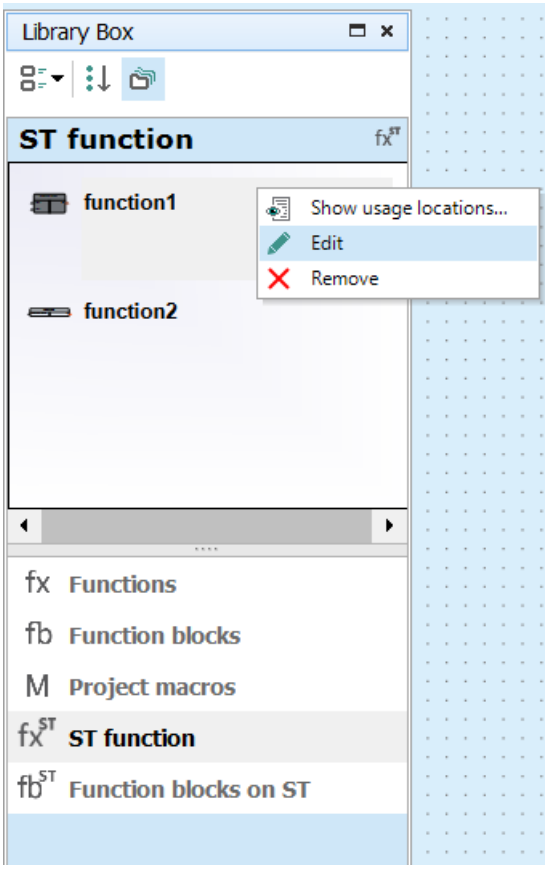
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2. Select the item ***Refresh usage locations***.

Go to the function editor

To go to the *function editor*:

1. Right-click on the function name.
2. Select the item ***Edit***.



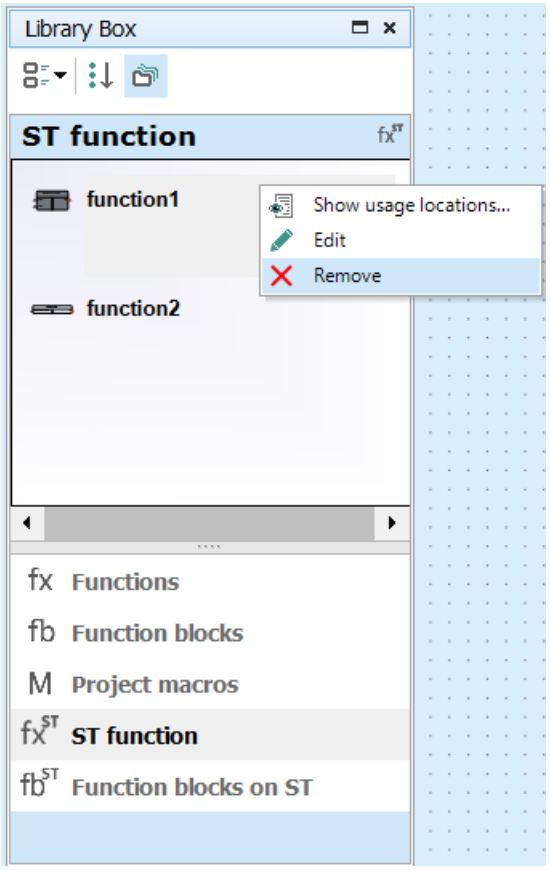
The *function editor* will open.

Delete function

To remove a function from a project:

1. Right-click on the function name.
2. Select the item **Remove**.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



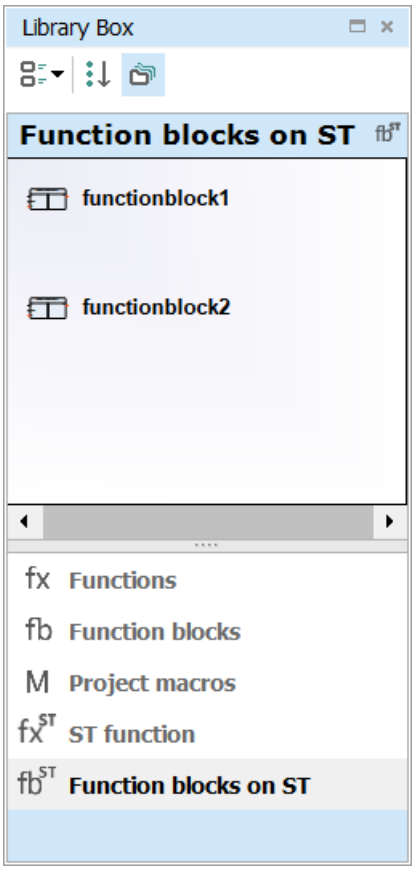
NOTICE
If the function is used in the diagram and/or in other functions, deletion may result in compilation errors.

7.5 ST function block

NOTE
User can create functions in ST language for PR100(M02), PR102, PR200, PR103, PR205, PR225 and SMI200 devices.

If ST function blocks are created in the project, they will be available in the component library.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

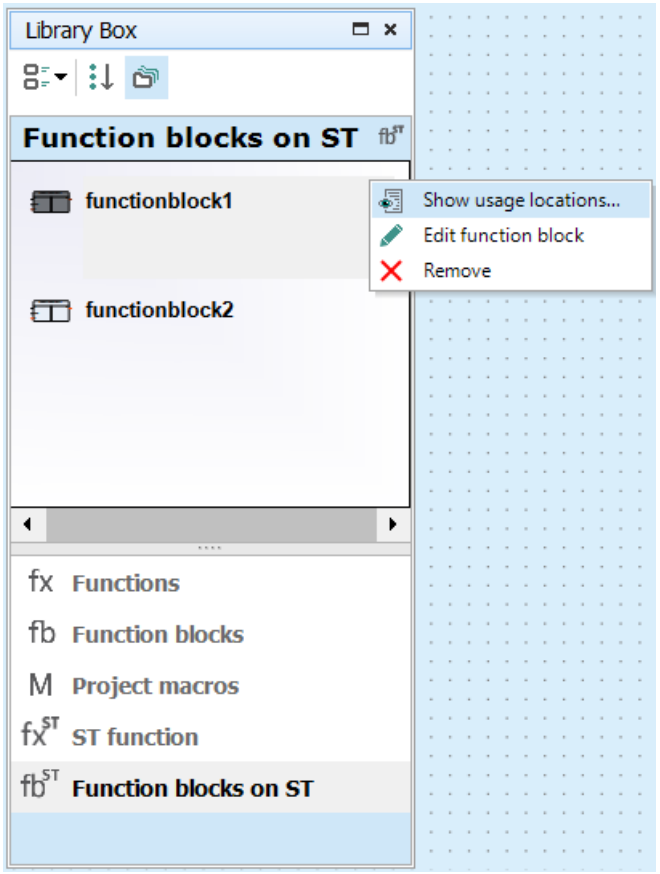


Usage location

To view all places where the function is used:

1. Right-click on the name of the function block.
2. Select the item **Show usage locations...**.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The **Function block usage locations** panel will open at the bottom of the window, showing where the function block is used in the main program and in the editor.



3. Right-click on the line with the location where the function block is used.
4. Select the item **To usage location**.



The focus will shift to where the function block is used on the diagram or in the editor.

NOTE
Double-click leads to the same result.

If the places where function blocks are used have changed while working with the program, you should update the **Function block usage location** panel:

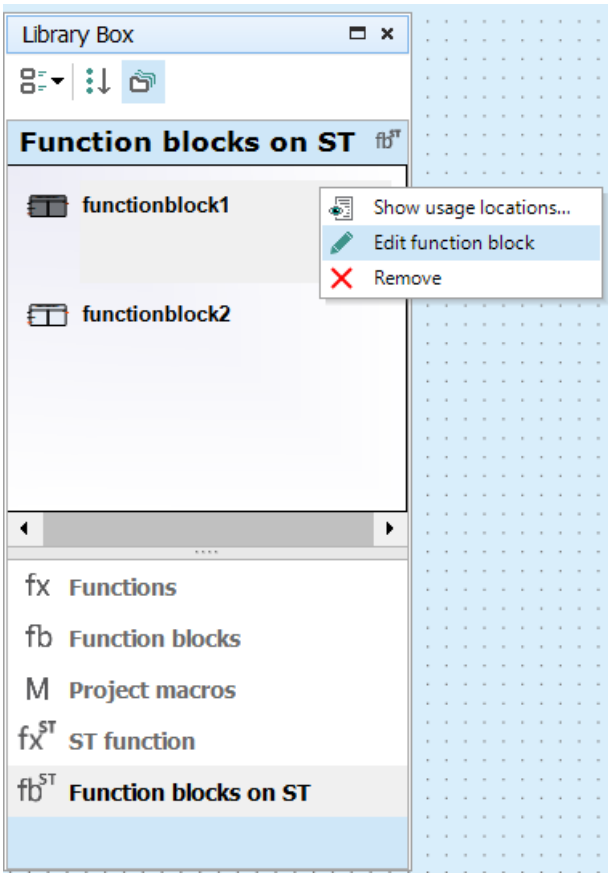
1. Right-click on any line of the panel.
2. Select the item **Refresh usage locations**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Go to the function block editor

To go to the *function block editor*:

1. Right-click on the function block name.
2. Select the item **Edit**.



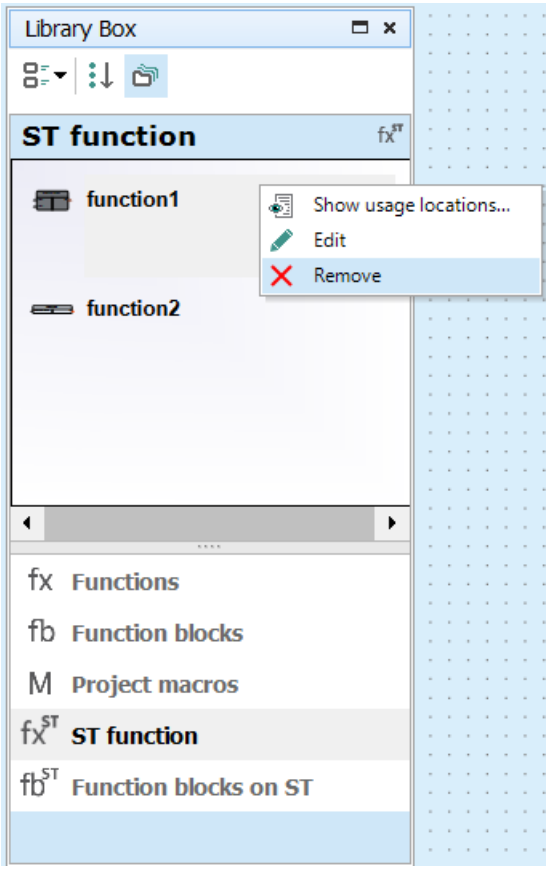
The *function block editor 4.13* will open.

Delete function block

To delete a function block:

1. Right-click the function block name.
2. Select the item **Remove**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



NOTICE
If the function block is used in the main program and/or in other functions, deletion may result in compilation errors.

7.6 Display elements

If the workspace with a display form is active, only display elements are available in Library Box. With these blocks, the information on the device display can be controlled. The display elements can be placed within the display form by drag-and-drop. The following elements are available:

– *Text box*
– *I/O box (INT/REAL)*
– *I/O box (BOOL)*
– *Dynamic box*
– *Combobox*

Use Property Box to customize an element.
Common parameters for all elements:

– ***Coordinate X*** – the position of the first (left) character placeholder of the element from the left edge of the form (from 0 to 15).
– ***Coordinate Y*** – the position of the first (left) character placeholder of the element from the upper edge of the form, depending on number of the rows in the form.
– There are two ways to determine coordinates: constant (default) or variable. To use a coordinate dependent on a variable, select the coordinate and open the list on the right of the input field.
  – ***Constant*** – specify the coordinates in Property Box or place the element within the form by drag-and-drop.
  – ***Variable*** – click ***Select*** to select an INT variable from the list and confirm with ***OK***. The display element will move according to the coordinate value controlled by the variable.
– ***Length*** – the number of reserved characters. The display element occupies one display row in height, its length can be from 1 to 16 characters.

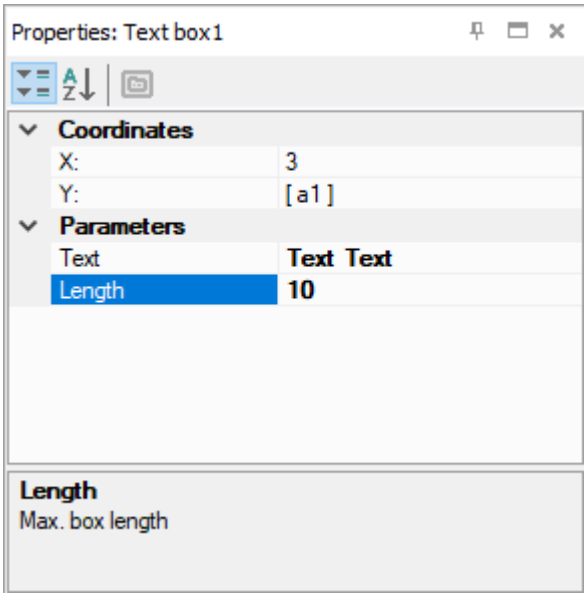
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.6.1 Text box

Text box is used to display plain text.

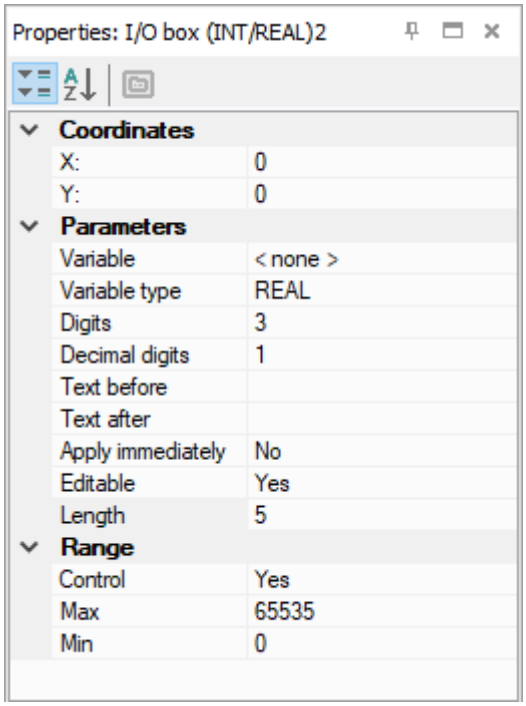
Parameters

Text – text to display. The parameter ***Length*** specifies the number of the reserved characters.



7.6.2 I/O box (INT/REAL)

I/O box (INT/REAL) is used to display a variable of type INT or REAL. The variable value can be changed with the device function buttons.



Parameters

– ***Variable*** – the reference to a variable. Use icon "**…**" in the input field to select the variable.
– ***Data type*** – INT or REAL. If the variable has been already selected, its data type will be accepted.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

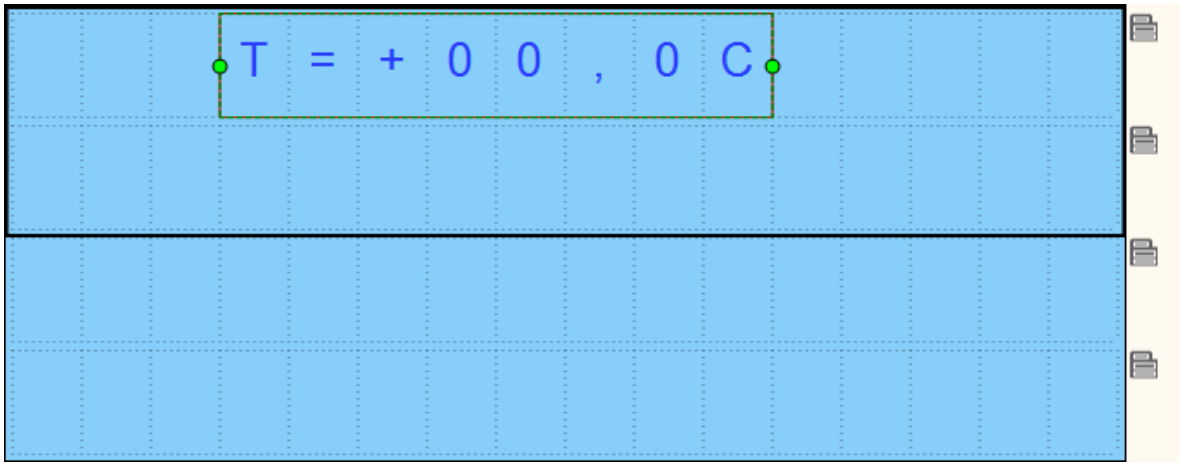
– ***Digits*** – the total number of displayed digits.
– ***Decimal digits*** — the number of the characters after the decimal point: 0…6 characters or ***Auto*** for Auto-precision*.
– ***Text before*** – the text to the left of the displayed variable.
– ***Text after*** – the text to the right of the displayed variable.
– ***Editable*** – if set to ***Yes***, the displayed value can be changed using the device function buttons.

NOTE
An output variable should be selected. The option has no effect with an input variable.

– ***Length*** – the total number of reserved characters including both the text before and after.

Range:
The group of parameters is used to limit the input value. If ***Editable*** = ***No***, the parameters of this group have no effect.

– ***Limit*** – if set to ***Yes***, the value entered using the device function buttons is limited by the user parameters ***Max*** and ***Min***, else it is limited only by the available memory area.
– ***Max*** – the maximum input value.
– ***Min*** – the minimum input value.



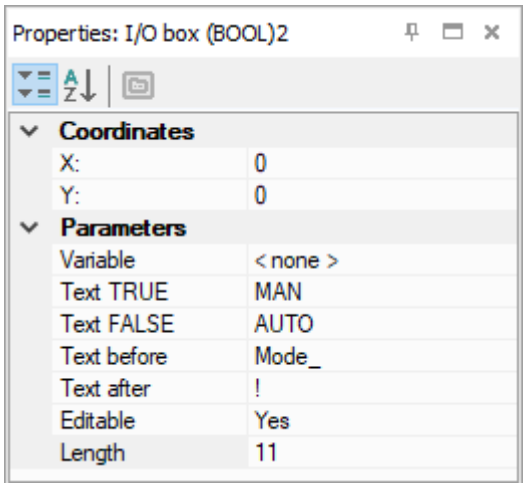
*** Auto-precision**
The option enables to display a REAL variable most precisely for the set number of reserved characters (parameter ***Digits***). To use the option, select in the workspace an I/O-Box display element with associated variable of REAL type and select ***Auto*** for the parameter ***Decimal digits*** in the Property Box.

Example:
To display the variable VAR1, 4 digits with Auto-precision are reserved. The value 1.546745 will be displayed rounded as 1.547. If the value changes to 110.478696, it will be displayed as 110.5.

7.6.3 I/O box (BOOL)

I/O box (BOOL) is used to display a variable of BOOL type. The value of the variable can be changed with the device function buttons.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

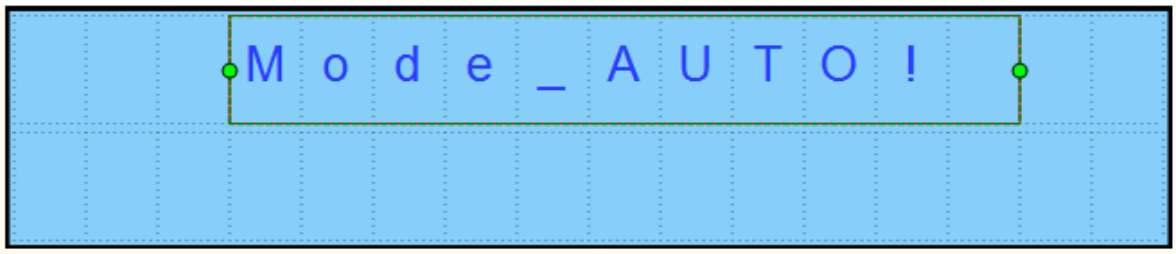


Parameters

- ***Variable*** – the reference to a variable. Use icon "**…**" in the input field to select the variable.
- ***Text TRUE*** – the text displayed if the variable is ***True***.
- ***Text FALSE*** – the text displayed if the variable is ***False***.
- ***Text before*** – the text to the left of the displayed variable.
- ***Text after*** – the text to the right of the displayed variable.
- ***Editable*** – if set to ***Yes***, the displayed value can be changed using the device function buttons.

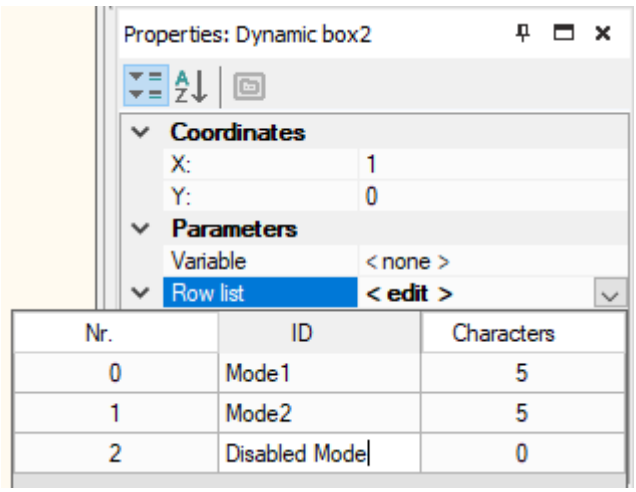
  NOTE
  An output variable should be selected. The option has no effect with an input variable.

- ***Length*** — the total number of reserved characters including both the text before and after.



7.6.4 Dynamic box

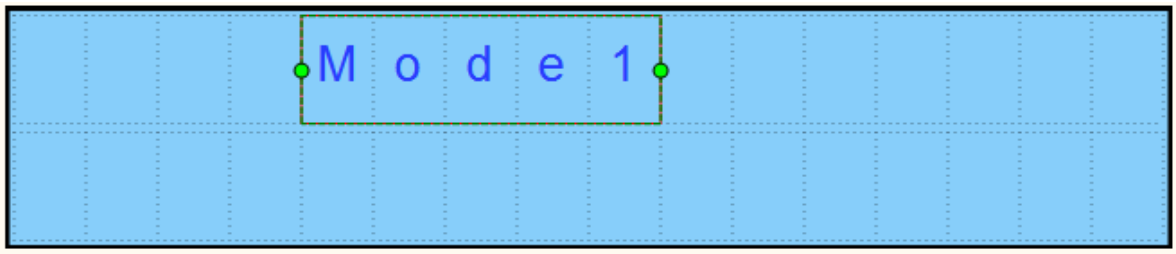
Dynamic box is an output field. It is used to display one of the text rows from a list depending on a row ***ID***. The row ***ID*** is saved in a referenced variable of INT type.



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

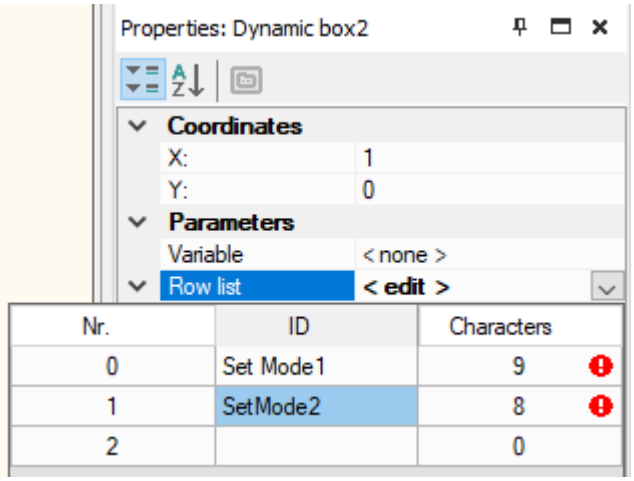
Parameters

- ***Variable*** — the reference to an integer project variable. To select the variable, click the "..." button and select from the *variable table 6*;
- ***Row list*** — the list with text rows. The ***Text*** from the row is displayed if the value of the referenced variable iequals to the row ***ID***. The column ***Characters*** shows the number of characters in the text. If the value of ***Length*** parameter is exceeded, there will be an exclamation mark near the number.
- ***Length*** — the number of reserved characters.



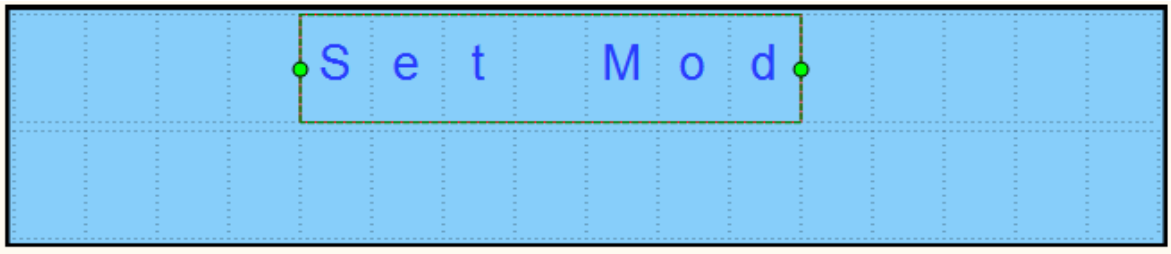
7.6.5 ComboBox

ComboBox is an input / output field. It is used to display one of the text rows from a list depending on a row ***ID***. The row ***ID*** is saved in a referenced variable of INT type. The ***ID*** can also be selected using the device function buttons.



Parameters

- ***Variable*** — the reference to a program variable. Use the icon «**...**» in the input field to select the variable.
- ***Row list*** — the table with text rows. The ***Text*** of the selected row is displayed and the row ***ID*** is saved in the referenced output variable. The column ***Characters*** shows the number of characters in the text. If the value of ***Length*** parameter is exceeded, there will be an exclamation mark near the number.
- ***Length*** — the number of the reserved characters.

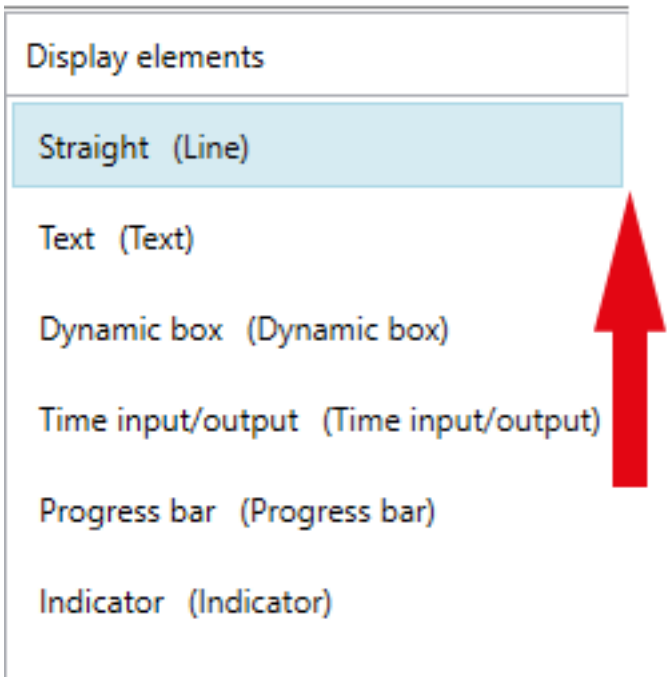


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7 Basic graphic elements

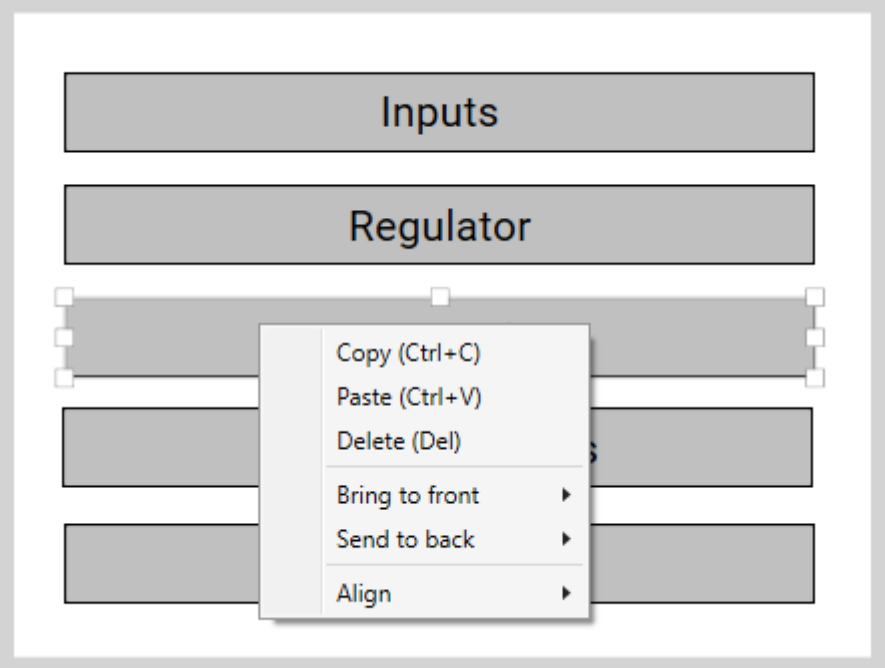
- *Text*
- *Indicator*
- *Progress bar*
- *Dynamic text*
- *Input/output int/float*
- *Time input/output*
- *IP Input/Output*
- *Line*
- *Polygon*
- *Circle*
- *Button 7.7.11*
- *Switch 7.7.12*
- *Switch group 7.7.13*
- *Image*
- *Menu*
- *Graph*

These elements are used to design the display of devices with a graphic color LCD. If several editable visualization elements are added to the screen, the order of their selection using the SEL button will be determined by the order in the "Screen Components" list from bottom to top (the lower elements will be highlighted first):



You can change the order of elements by dragging and dropping an element in the list or using the context menu by right-clicking on an element on the canvas:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The following data provides information on the amount of RAM (visualization RAM) required for displaying visualization elements on devices with a graphical color LCD:

Visualization Element	Visualization RAM Used, bytes	Additional Usage	Note
Empty Screen	252		
Physical Buttons	80	RAM calculation: 80 bytes (each button) + 20 bytes * number of variables + 10 bytes * number of actions + 10 bytes (mandatory reserve) Calculation examples: 1. Changing a variable on short press = 80+20*1 +10*1+10 = 120 bytes 2. Screen transition on short press = 80 +20*0+10*1+10 = 100 bytes 3. Changing a variable on short and long press, and screen transition on short press = 80 +20*2+10*3+10 = 160 bytes	
Text	464	If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	
Indicator	Default size, without frame – 630	If a variable is used: +20 bytes per variable.	The amount of RAM used directly depends on the element's

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visualization Element	Visualization RAM Used, bytes	Additional Usage	Note
		If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	dimensions and frame thickness: – Larger element dimensions require more memory for display; – Frame 0 px (disabled) - minimum RAM usage for an element of this size; – Frame 1 px - RAM usage increases compared to an element without a frame; – Frame 2 px or more - RAM usage increases compared to an element with a 1 px frame. When the frame thickness exceeds 2 px, the amount of RAM used does not increase further and corresponds to the usage for an element with a 2 px thick frame.
Progress bar	408	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Value visibility: +16 bytes for displaying the value (Value, visible = Yes).	
Dynamic box	836 (for one line)	+4 bytes for each additional line. If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Input control: +12 bytes for allowing editing (Disable editing = No).	
REAL/INT input/ output	776	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visualization Element		Visualization RAM Used, bytes	Additional Usage	Note
			Input control: +24 bytes for allowing editing (Disable editing = No).	
Time input/output		688 for date **or** time only. 704 for date **and** time.	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Input control: +28 bytes for allowing editing (Disable editing = No).	
Input/Output of IP Address		676	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Input control: +24 bytes for allowing editing (Disable editing = No).	
Line	Straight line	182	If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	
	Broken line	190		
Polygon	Rectan-gle	338	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Fill: +132 bytes (Appearance, Fill = Yes). Action configuration (screen transition or variable change) = 10 bytes * number of actions + 26 bytes.	
	Triangle	334		
	Diamond	338		
Circle	Circle	Default size, without fill – 978 Default size, with fill – 1346	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Action configuration (screen transition or variable change) = 10 bytes * number of actions + 26 bytes.	The amount of RAM used directly depends on the element's dimensions and the presence of a fill: – Larger element dimensions require more memory for display; – The RAM usage for an element with a fill is greater than for an element without a fill.
	Circle sector	Default size, without fill – 1502 Default size, with fill – 1878		

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visualization Element	Visualization RAM Used, bytes	Additional Usage	Note
			For certain combinations of settings, the "Thickness" parameter of the focus frame may slightly affect the visualization RAM used.
Button	Rectangular without rounded corners, without frame – 894	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding. Action configuration (screen transition or variable change) = 10 bytes * number of actions + 26 bytes. +6 bytes for adding a 1px frame (for a rectangular button without rounded corners). +20 bytes for adding a frame with a thickness of 2 px or more (for a rectangular button without rounded corners).	For a round button or a rectangular button with rounded corners, the amount of RAM used directly depends on the element's dimensions: – Larger element dimensions require more memory for display. For certain combinations of settings, the "Thickness" parameter of the focus frame may slightly affect the visualization RAM used.
Switch	Default size, without frame – 1192	If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	The amount of RAM used directly depends on the element's dimensions: – Larger element dimensions require more memory for display. For certain combinations of settings, the "Thickness" parameter of the focus frame may slightly affect the visualization RAM used.
Switch group	1688 (for a group with one cell)	+ 208 bytes for each additional cell. If a variable is used: +20 bytes per variable. If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	
Image	372	If visibility is controlled by a variable: +20 bytes for the visibility variable binding.	Calculation of user visualization flash memory usage = Width * Height * 2.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visualization Element		Visualization RAM Used, bytes	Additional Usage	Note
			Action configuration (screen transition or variable change) = 10 bytes * number of actions + 26 bytes. + 32 bytes for adding a frame.	The calculation uses the width and height (in pixels) of the original image loaded into the visualization editor. If the same image is copied and linked to multiple visualization elements, no additional user visualization flash memory is used.
Menu		1870 (for a menu with one item)	+ 298 bytes for each additional row/line. If visibility is controlled by a variable: +20 bytes for the visibility variable binding of the element or row.	
Chart	Background operation disabled	Without pens, time scale 2 sec – 982	Additional cost for increasing display time range: From 2 bytes for 3 seconds up to 916 bytes for 23:59:59. Additional cost per pen: + 20 bytes (first pen) + each subsequent pen depending on the time range (from 74 bytes for 2 seconds up to 990 bytes for 23:59:59). If visibility is controlled by a variable: +20 bytes for the visibility variable binding of the element or pen.	The amount of RAM used directly depends on the element's width: – A wider element requires more memory for display.
	Background operation enabled	Without pens, time scale 2 sec – 1002	Additional cost for increasing display time range: From 6 bytes for 3 seconds up to 2748 bytes for 23:59:59. Additional cost per pen: + 20 bytes (first pen) + each subsequent pen depending on the time range (from 94 bytes for 2 seconds up to 2842 bytes for 23:59:59). If visibility is controlled by a variable: +20 bytes for the visibility variable binding of the element or pen.	

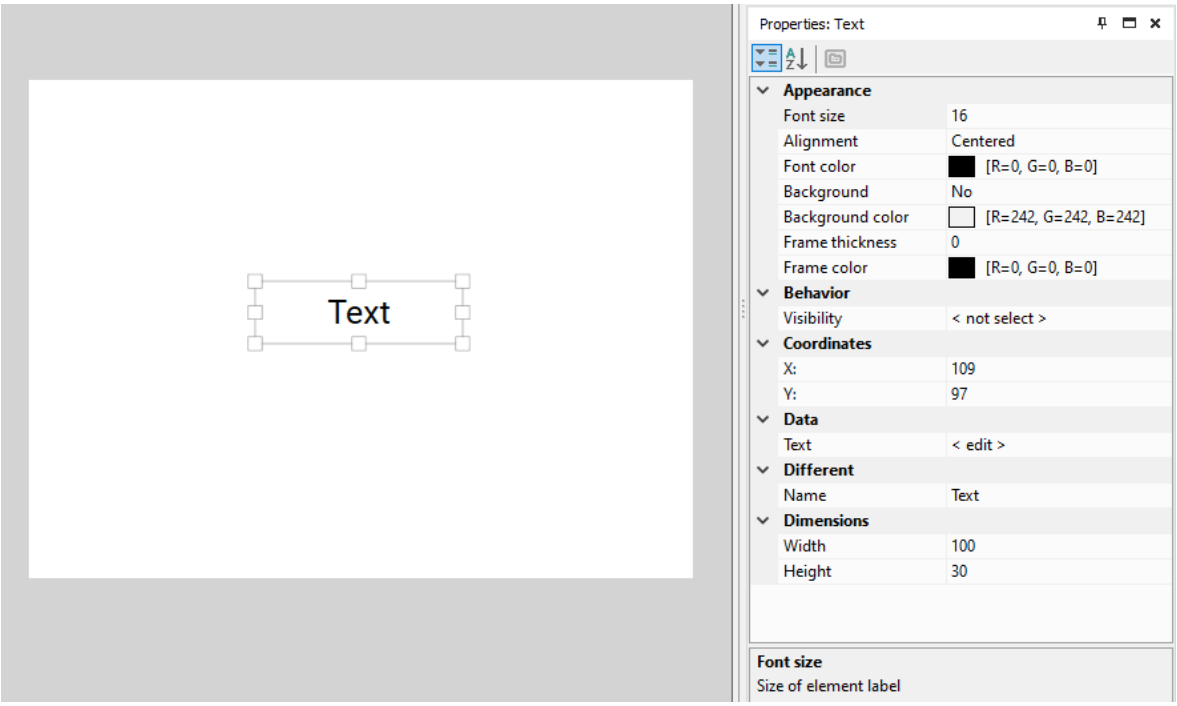
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7.1 Text

The **Text** element is intended for placing a text block in the visualization screen.

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions, enter text and set other parameters on the properties panel.



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis – from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis – from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Parameters

Group	Parameter	Description
Appearance	**Font size**	Size of text displayed in element: 16, 32, 48
	Alignment	Alignment of the text displayed in the element: left, center, right
	Font color	The color of the text displayed in the element
	Background	The background of the text displayed in the element
	Background color	The background color of the text displayed in the element

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

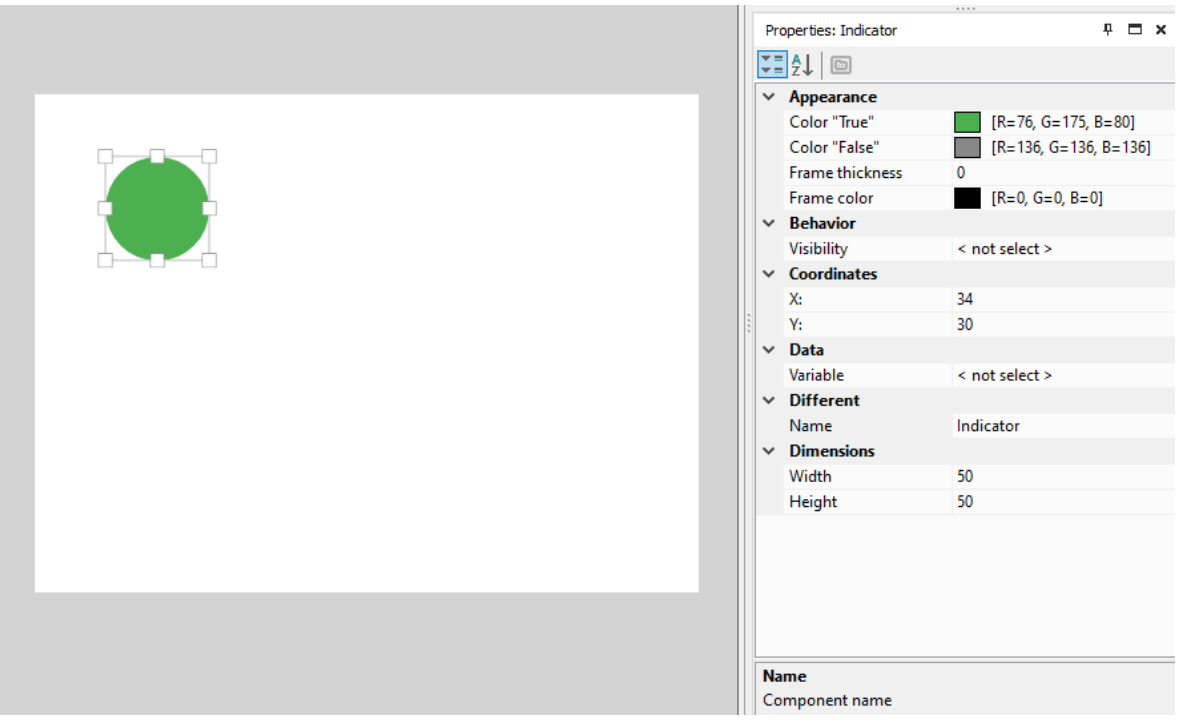
Group	Parameter	Description
	Frame thickness	Element frame thickness
	Frame color	Element border color
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	**Text**	The field contains the text that will be displayed in the element
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

7.7.2 Indicator

The **Indicator** element is designed to display the value of a Boolean variable in the visualization screen. If the variable value is **1**, the indicator takes the color "True". If the variable value is **0**, the indicator takes the color "False".

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions and set the parameters on the properties panel.



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Parameters

Group	Parameter	Description
Appearance	Color “True”	The color of the element if the value of the bound variable is TRUE
	Color “False”	The color of the element if the value of the bound variable is FALSE
	Frame thickness	Element frame thickness
	Frame color	Element border color
Behavior	Visibility	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	Variable	A boolean variable is bound to the parameter for display
Different	Name	Displayed in the list of used elements on the left side of the visualization editor

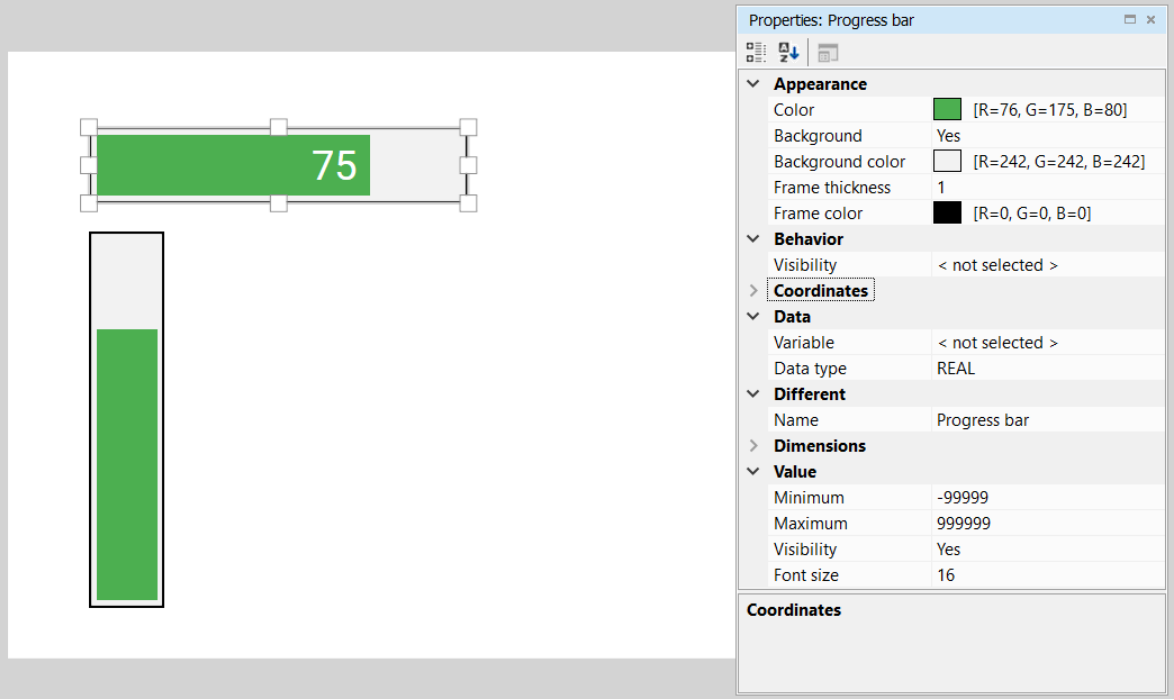
NOTE
If no variable is bound to the **Variable** parameter, the color of the element will always be **True**.

7.7.3 Progress bar

The Progress bar element is designed to display the value of an integer or floating-point variable as a scale on the visualization screen.

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions and set the parameters on the properties panel.



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The element's dimensions are defined by the X and Y axes and are set in the properties panel or by changing the element's borders on the screen field. For vertical orientation of the element, the Y value must be greater than the X value.

Parameters

Group	Parameter	Description
Appearance	**Color**	Element scale color
	Background	Background of the displayed element
	Background color	Background color of the displayed element
	Frame thickness	Element frame thickness
	Frame color	Element border color
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	**Variable**	An integer or floating point variable is bound to the parameter for display.
	Variable type	Type of variable bound to element: floating point, integer
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor
Value	**Minimum**	0 % of element scale
	Maximum	100 % of the measurement scale
	Visibility	Displaying the scale value in an element
	Font size	The font size of the text displayed in the element: 16, 32, 48

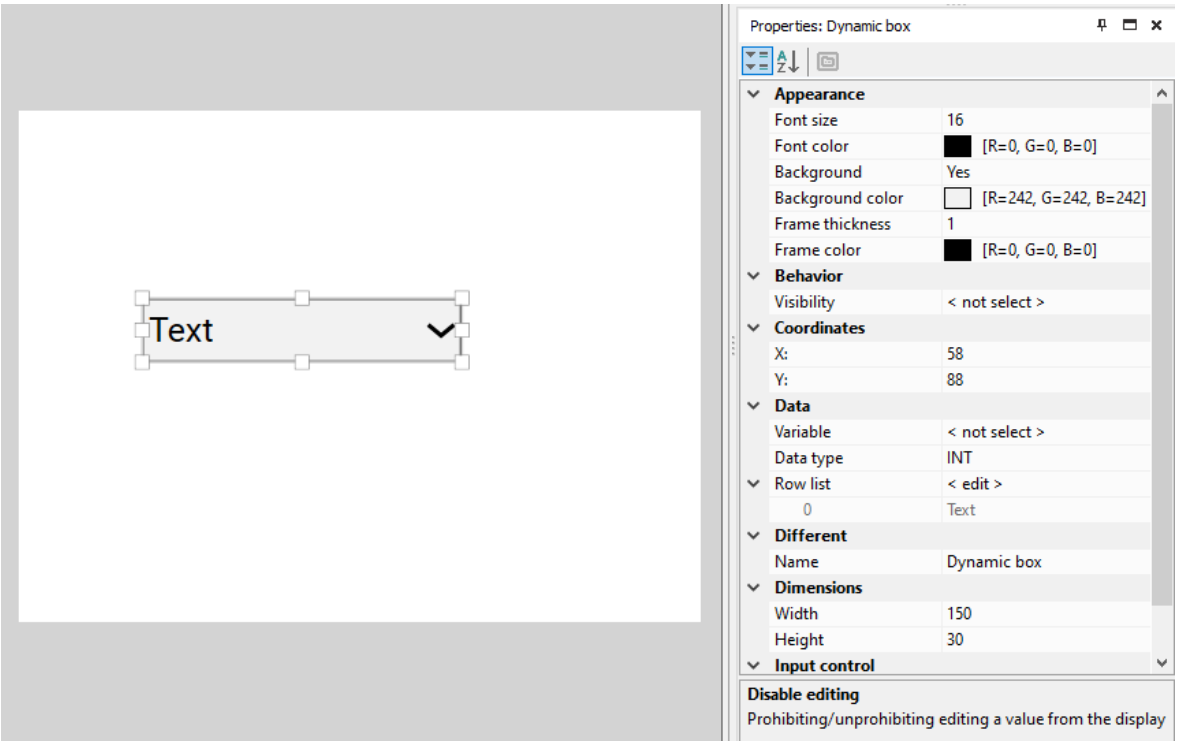
7.7.4 Dynamic box

The **Dynamic box** element is designed to select and display the value of an integer variable from a drop-down list on the visualization screen. The integer value can be assigned a text designation, which will be displayed in the drop-down list.

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions, and set the parameters and data on the properties panel.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Parameters

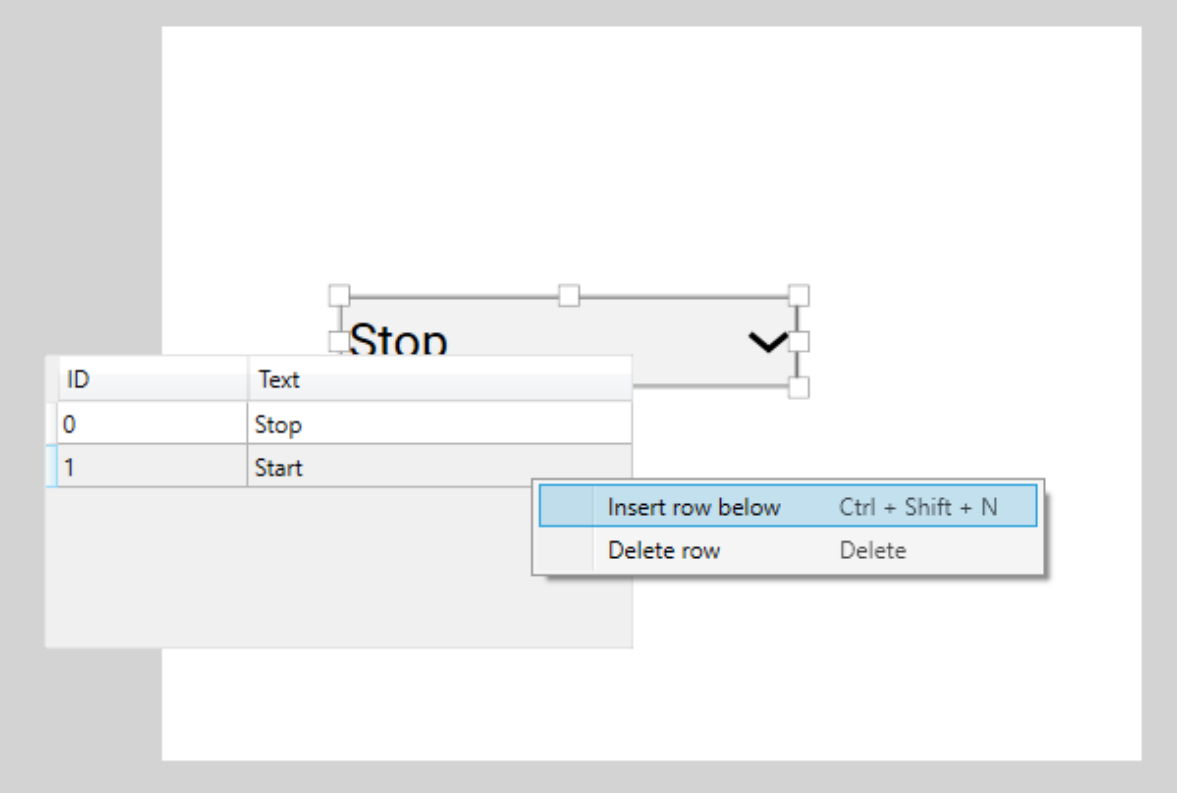
Group	Parameter	Description
Appearance	**Font size**	The size of the text displayed in the element: 16, 32, 48
	Font color	The color of the text displayed in the element
	Background	The background of the text displayed in the element
	Background color	The background color of the text displayed in the element
	Frame thickness	Element frame thickness
	Frame color	Element border color
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element.
Data	**Variable**	A variable is bound to the parameter to store values
	Data type	Selecting the data type of the bound variable: INT and BOOL

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description
		If a Boolean variable type is selected, the element's drop-down list can contain only two lines. If an integer variable type is selected, the drop-down list can contain a maximum of 128 rows.
	Row list	In tabular form, the variable values and their corresponding names, which will be displayed in the element's drop-down list
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor
Input control	**Disable editing**	If set to ***Yes***, the value of the bound variable can only be viewed, the input option will not be available. Otherwise, entering a value in the element overwrites the variable value

To add a new line to an element's drop-down list, follow these steps:

1. Double-click the **Dynamic box** element in the screen editor.
2. In the window that opens, right-click on one of the rows in the table.
3. Select **Insert row below**.



Example:

Table 7.6 List of lines

0	Stop
1	Start
2	Pause

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7.5 REAL/INT input/output

The **REAL/INT input/output** element is intended for inputting and displaying the value of an integer or floating-point variable on the visualization screen.

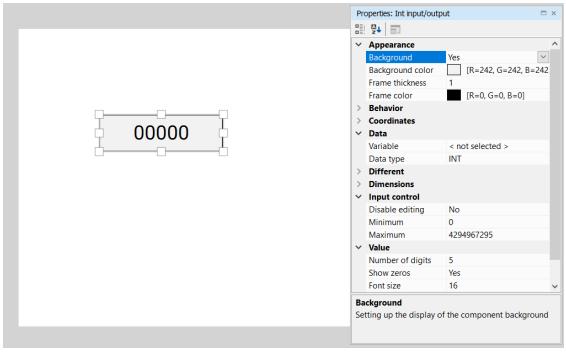


Fig. 7.1 Input/Output int

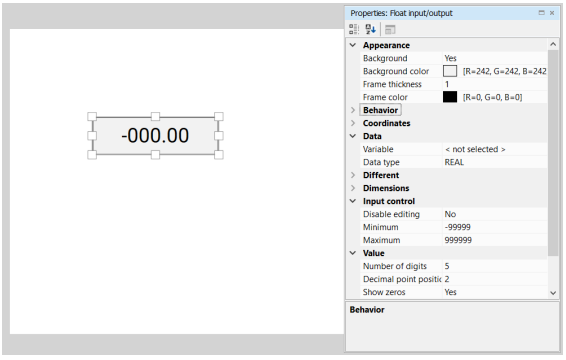


Fig. 7.2 Input/Output float

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions, and set the parameters and data on the properties panel.

Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen.
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

NOTE
If the output variable is outside the limits specified in the **Minimum** / **Maximum** parameters, then the element on the device will display the current state of the variable and highlight the frame of this element in red.

Configuring Properties

Group	Parameter	Description	Additional Options
Appearance	**Background**	The background of the text displayed in the element	
	Background color	The background color of the text displayed in the element	
	Frame thickness	Element frame thickness	
	Frame color	Element border color	
Data	**Variable**	A variable is bound to the parameter	
	Data type	To input/output an integer value, specify the Integer type.	

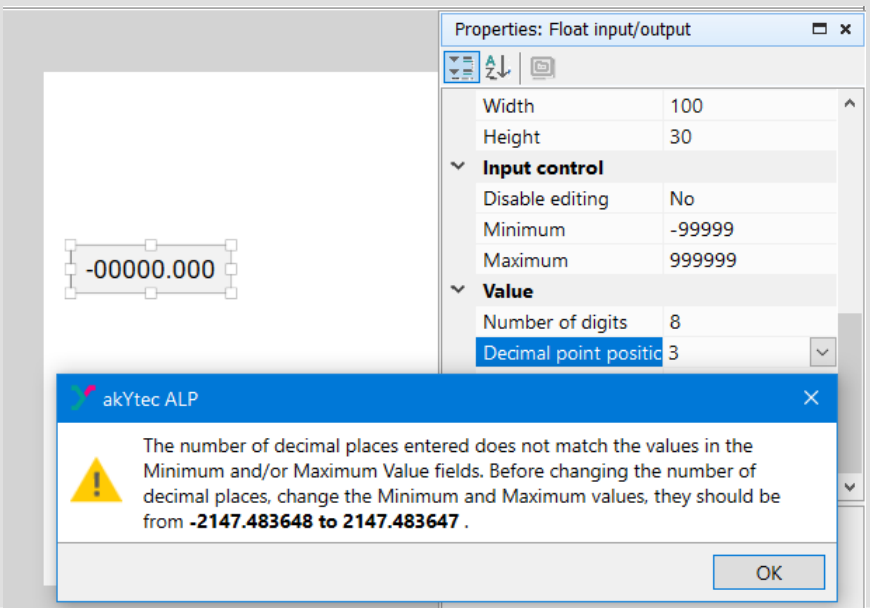
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional Options
Value	**Number of digits**	The number of digits displayed in the element	When selecting a float variable: Cannot be less than or equal to the value of **Decimal point position**.
	Decimal point position (setting available only when selecting a float variable)	The number of digits in the fractional part of the value displayed in the element	Changing this value may require adjusting the **Maximum** and **Minimum** values. (See example after the table)
	Show zeros	Setting up the display of trailing zeros	
	Font size	The size of the text displayed in the element: 16, 32, 48	
	Font color	The color of the text displayed in the element	
	Alignment	Alignment of the text displayed in an element	
Input control	**Disable editing**	If set to **Yes**, the value of the bound variable can only be viewed, the input option will not be available. Otherwise, entering a value in the element overwrites the variable value	
	Minimum	Minimum value to enter in an element	When selecting a float variable: The limit for the value is calculated using the formula: 2147483648 / 10^*Number of Decimal Places* Changing this value may require adjusting the **Number of Decimal Places** property. (See example after the table)
	Maximum	Maximum value for input in element	When selecting a float variable: The limit for the value is calculated using the formula: 2147483647 / 10^*Number of Decimal Places* Changing this value may require adjusting the **Number of**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

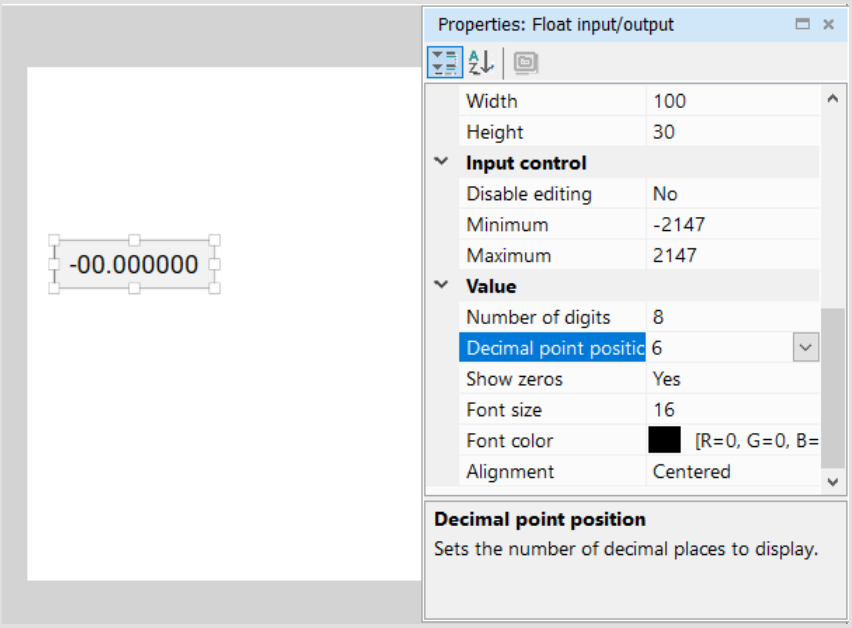
Group	Parameter	Description	Additional Options
			Decimal Places property. (See example after the table)
Coordinates	**X**	Position value of the element along the X-axis. The element position is determined by the coordinates of the top-left corner. Coordinate counting starts at 0 and proceeds from left to right; the final value is determined by the element dimensions and screen width.	Coordinates can be set by moving the object across the screen.
	Y	Position value of the element along the Y-axis. The element position is determined by the coordinates of the top-left corner. Coordinate counting starts at 0 and proceeds from top to bottom; the final value is determined by the element dimensions and screen height.	
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element	
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor	
Dimensions	**Width**	Width of the element along the X-axis	Element dimensions can be set by changing the object boundaries on the screen field
	Height	Height of the element along the Y-axis	

Example:
For the **Input/Output float** visualization element with 8 total digits displayed, it is necessary to set 6 decimal places:

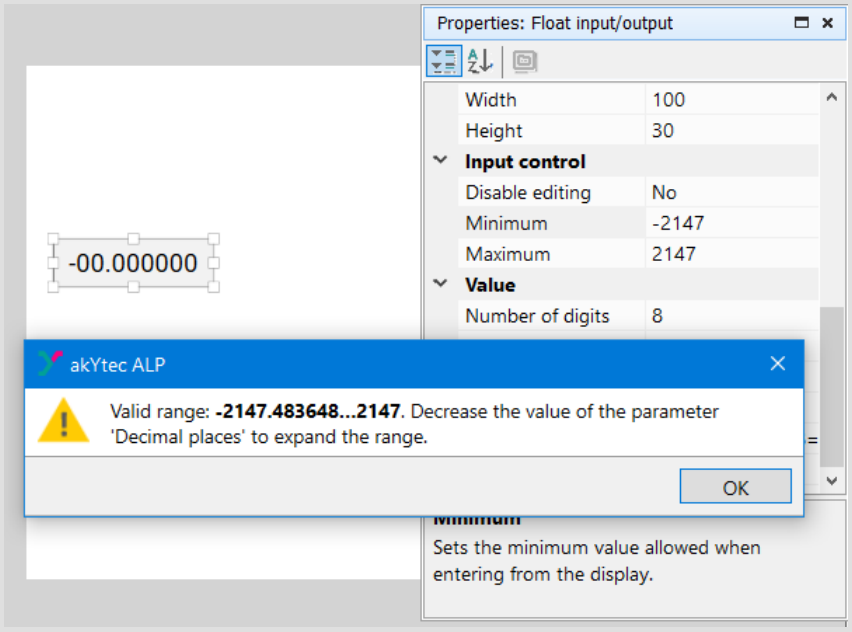


Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

When selecting 6 for the **Number of Decimal Places** property, an error will be displayed indicating the limits of the **Minimum** and **Maximum** values, calculated using the formula:
Minimum value = -2147483648 / 10^6 = -2147.483648
Maximum value = 2147483647 / 10^6 = 2147.483647
Enter the **Minimum** and **Maximum** values within the specified range, and select 6 for the **Number of Decimal Places** property:



The value has changed; the visualization element now displays 6 decimal places.
When attempting to change the **Minimum** or **Maximum** value to a value outside the range calculated by the formula above, an error message will be displayed:



The **Number of Decimal Places** property and the **Minimum** and **Maximum** values are interrelated.

NOTE
If the output variable exceeds the boundaries specified in the **Minimum** / **Maximum** parameters, the device will display the current state of the variable and highlight the element border in red.

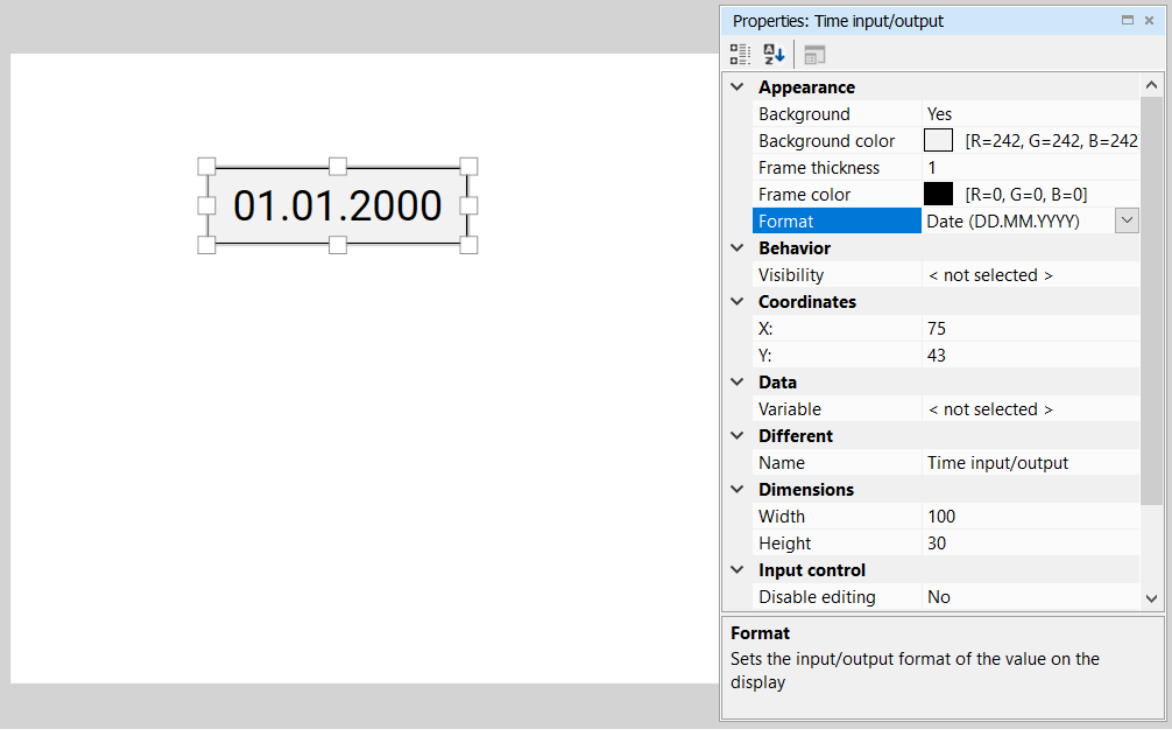
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7.6 Time input/output

The **Time input/output** element is used to input and display the date and time value on the visualization screen. An integer variable is used to store the date and time value, which stores the number of seconds since 00:00:00 January 01, 2000.

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions, and set the parameters and data on the properties panel.



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Parameters

Group	Parameter	Description
Appearance	**Background**	The background of the text displayed in the element
	Background color	The background color of the text displayed in the element
	Frame thickness	Element frame thickness
	Frame color	Element border color

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description
	Format	Select from the drop-down list the format for displaying the date and time. Format notations: – **ss** – seconds; – **mm** – minutes; – **hh** – hours; – **DD** – day; – **MM** – month; – **YYYY** – year.
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	**Variable**	A variable is bound to the parameter
Input control	**Disable editing**	If set to **Yes**, the value of the bound variable can only be viewed, the input option will not be available. Otherwise, entering a value in the element overwrites the variable value
Value	**Font size**	The size of the text displayed in the element: 16, 32, 48
	Font color	The color of the text displayed in the element
	Alignment	Alignment of the text displayed in an element
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

7.7.7 Input/output of IP address

The **Input/output of IP address** element is used to enter and display the IP address (IPv4) value on the visualization screen. An integer variable is used to store the IP address value, the value of which is calculated using the formula:

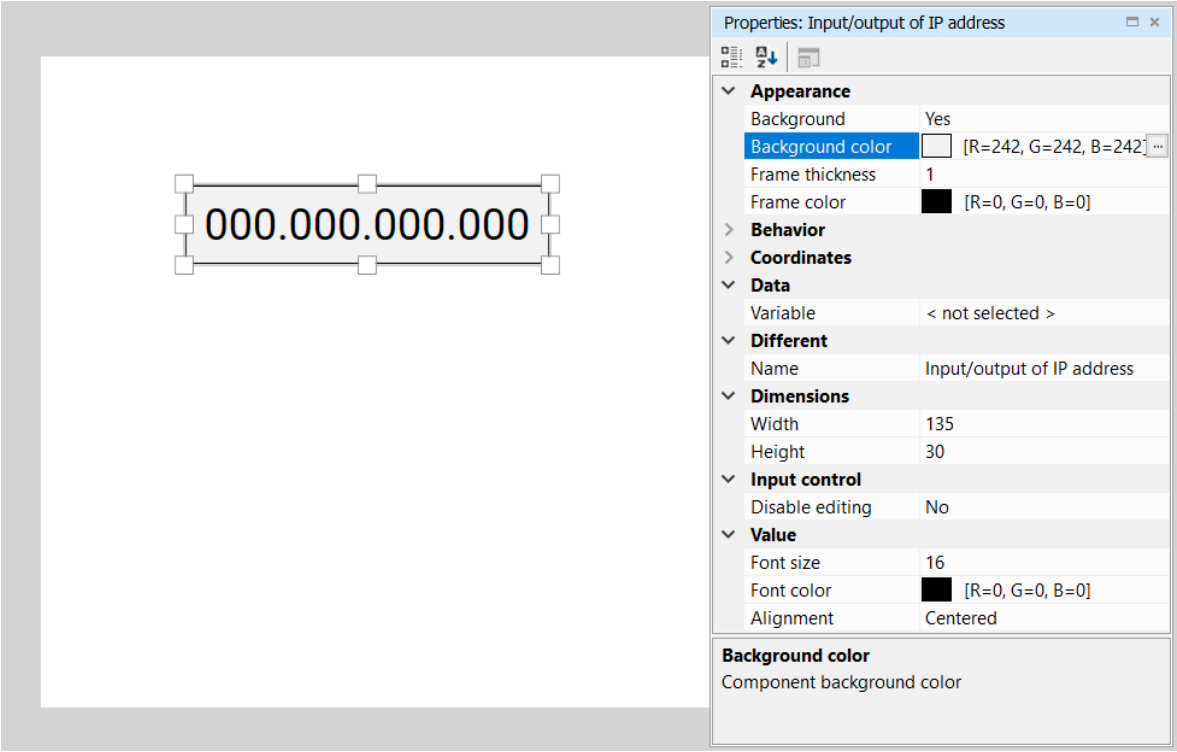
$$int = o1 * 256^3 + o2 * 256^2 + o3 * 256^1 + o4 * 256^0,$$

where *int* is the value of an integer variable;
o1, o2, o3, o4 – IP address octets.

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions, and set the parameters and data on the properties panel.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Coordinates

The element's location is determined by the X and Y axes. The coordinates are set in the properties panel or by moving the element across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Parameters

Group	Parameter	Description
Appearance	**Background**	The background of the text displayed in the element
	Background color	The background color of the text displayed in the element
	Frame thickness	Element frame thickness
	Frame color	Element border color
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	**Variable**	A variable is bound to the parameter
Input control	**Disable editing**	If set to **Yes**, the value of the bound variable can only be viewed, the input option will not be available. Otherwise, entering a value in the element overwrites the variable value
Value	**Font size**	Size of text displayed in element: 16, 32, 48

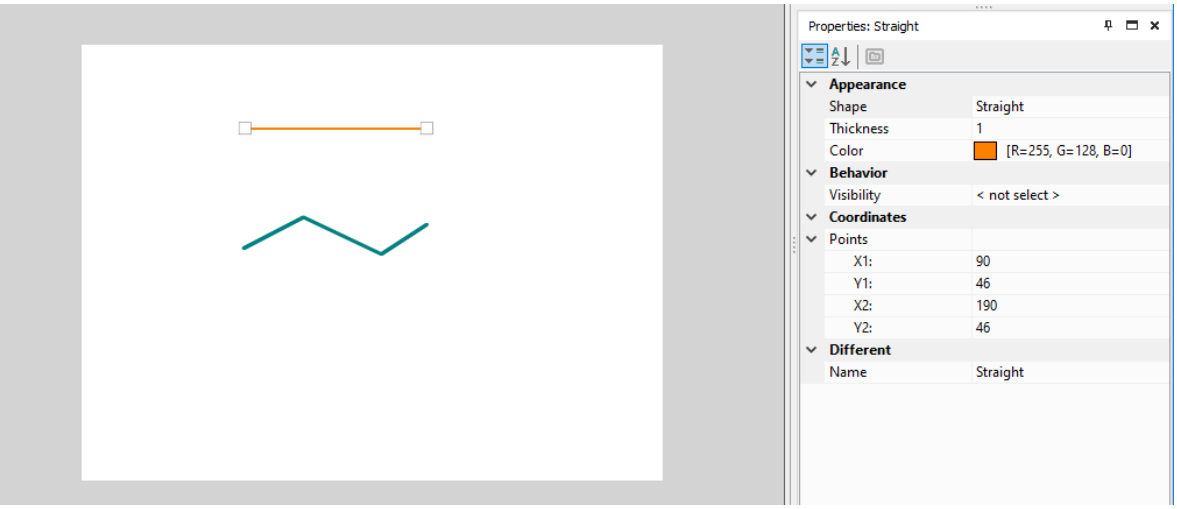
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description
	Font color	The color of the text displayed in the element
	Alignment	Alignment of the text displayed in an element
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

7.7.8 Line

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions and set the parameters on the properties panel.



Coordinates

The location of the straight line is determined by two points, the coordinates of which along the X and Y axes are specified in the properties panel or by moving the points across the screen. The coordinates along both axes start from 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

The location of the broken line is determined by four points.

Parameters

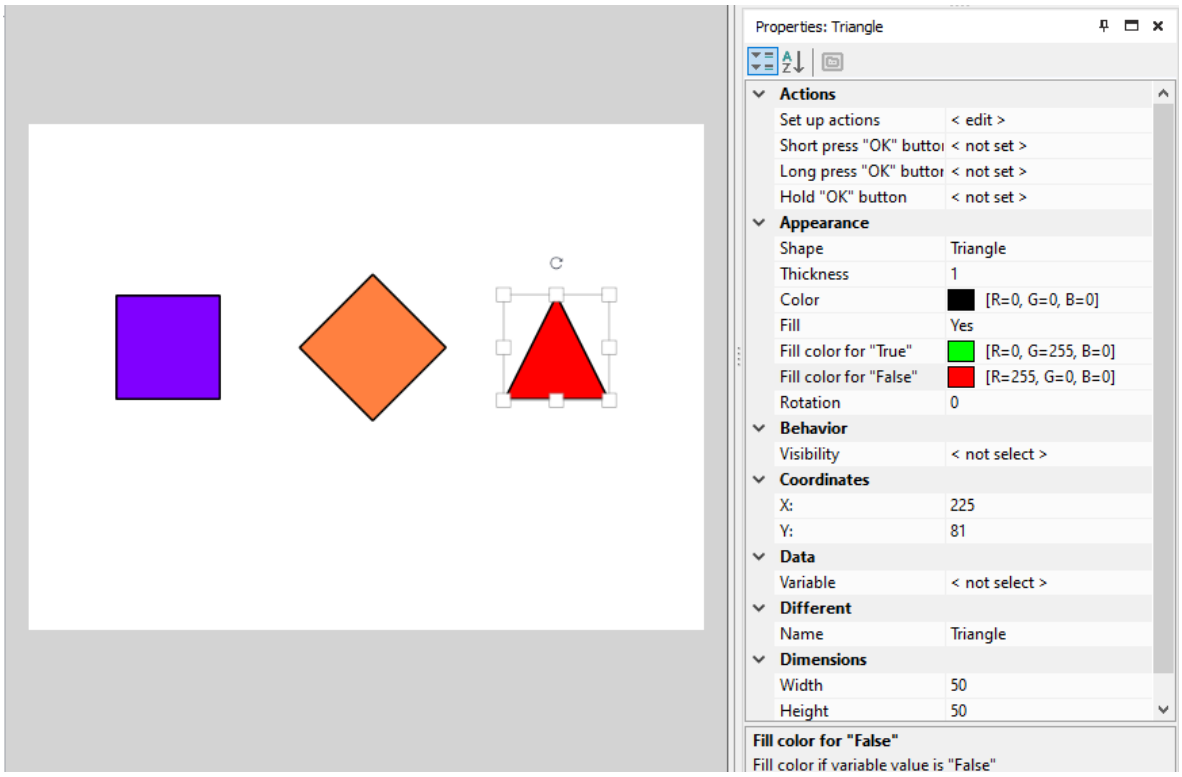
Group	Parameter	Description
Appearance	**Shape**	Display line shape: straight, polyline
	Thickness	Line thickness
	Color	Element color
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.7.9 Polygon

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions and set the parameters on the properties panel.



Coordinates

The polygon's location is determined by the coordinates of its center along the X and Y axes. The coordinates are set in the properties panel or by moving the object across the screen. The coordinates along both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Parameters

Group	Parameter	Description
Actions	**Set up actions**	Select an action to navigate between screens
	Short press “OK” button	List of actions performed by short pressing the “OK” button
	Long press “OK” button	List of actions performed by long pressing the “OK” button
	Hold “OK”	List of actions performed when holding the “OK” button
Appearance	**Shape**	The shape of the displayed polygon: triangle, rectangle, rhombus

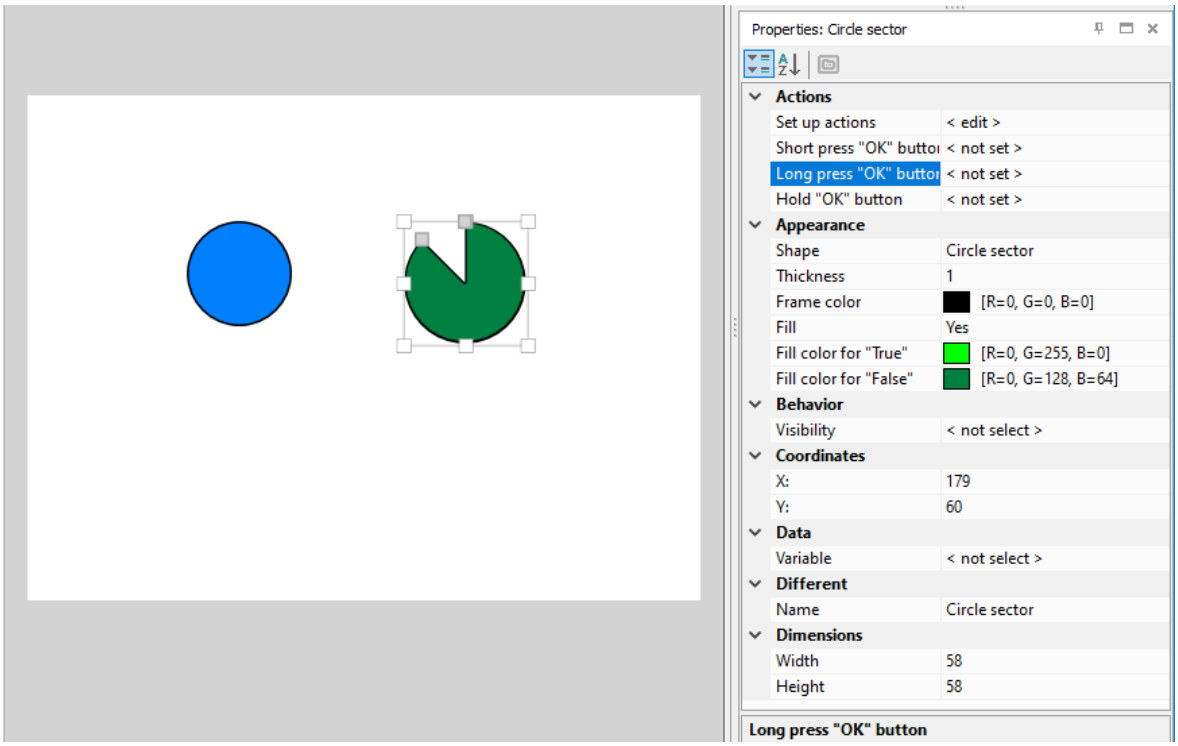
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description
	Thickness	Polygon outline thickness
	Color	Polygon outline color
	Fill	Presence of fill in the figure
	Fill color for «True»	Fill color of the shape if the bound variable is ***True***
	Fill color for «False»	The fill color of the shape if the bound variable is ***False***
	Rotation	Rotate an element around its center in degrees
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element
Data	**Variable**	A variable is bound to the parameter
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

7.7.10 Circle

Setting up properties

For correct display on the device screen, you should specify the location coordinates, element dimensions and set the parameters on the properties panel.



Coordinates

The location of the circle is determined by the coordinates of its center along the X and Y axes. The coordinates are set in the properties panel or by moving the object across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

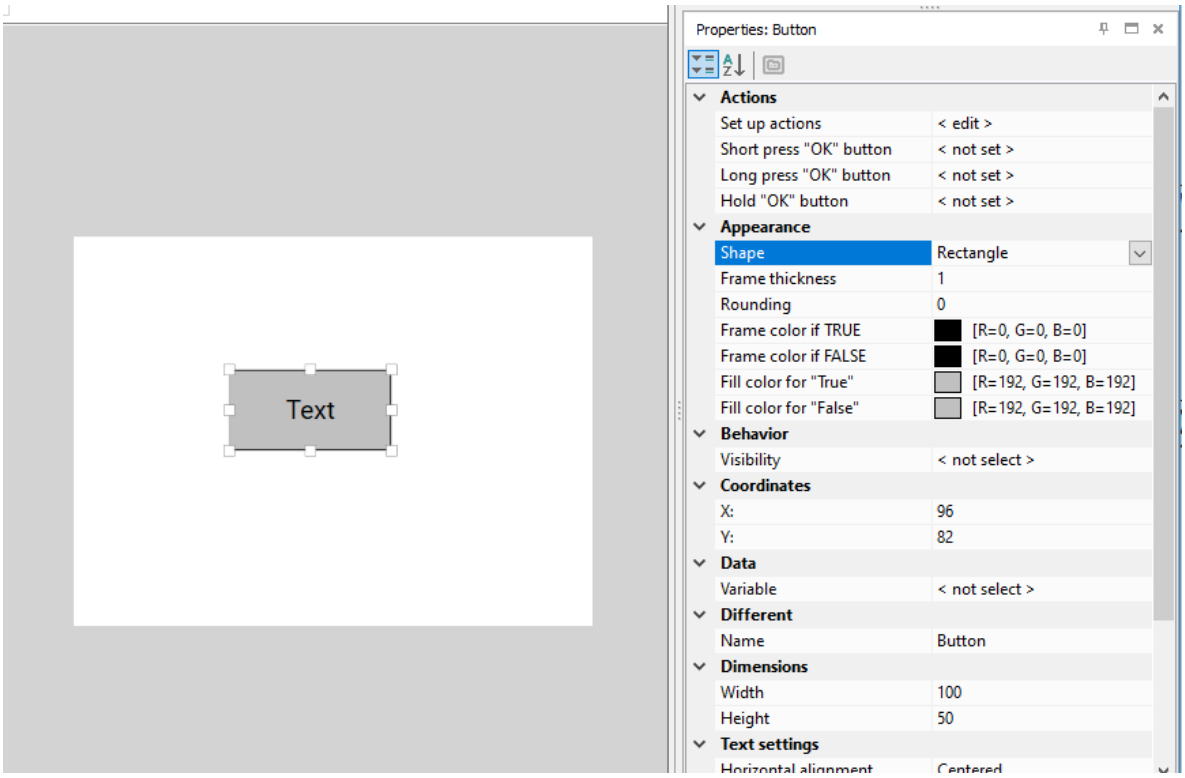
Parameters

Group	Parameter	Description
Actions	**Set up actions**	Select an action to navigate between screens
	Short press “OK” button	List of actions performed by short pressing the ***OK*** button
	Long press “OK” button	List of actions performed by long pressing the ***OK*** button
	Hold “OK” button	List of actions performed when holding the ***OK*** button
Appearance	**Shape**	The shape of the displayed circle: circle, circle sector
	Thickness	Thickness of the outline of the figure
	Frame color	Color of the outline of the figure
	Filling	Presence of fill in the figure
	Fill color for "True"	The fill color of the shape if the bound variable is ***True***
	Fill color for "False"	The fill color of the shape if the bound variable is ***False***
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element.
Data	**Variable**	A variable is bound to the parameter
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor

7.7.11 Button

The **Button** element is designed to switch the state of a Boolean variable.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Configuring Properties

Group	Parameter	Description	Additional features
Appearance	**Shape**	The shape of the button to display: **rectangle** or **circle**	
	Frame thickness	Button outline thickness	
	Rounding	Rounding the corners of the frame. Available only when the Rectangle shape is selected	
	Frame color if TRUE	The fill color of the frame if the value of the bound variable is "True"	
	Frame color if FALSE	The fill color of the frame if the bound variable is "False"	
	Fill color for "True"	The fill color of the shape if the value of the bound variable is "True"	
	Fill color for "False"	The fill color of the shape if the bound variable is "False"	
Data	**Variable**	A variable is bound to the parameter	
Actions	**Set up actions**	Select an action to navigate between screens	
	Short press “OK” button	List of actions performed by short pressing the “OK” button	
	Long press “OK” button	List of actions performed by long pressing the “OK” button	
	Hold “OK” button	List of actions performed when holding the “OK” button	
Coordinates	**X**	Position value of the element along the X-axis. Element position is determined by top-	Coordinates can be set by moving

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional features
		left corner coordinates. Coordinate counting starts at 0 and goes left to right; final value is determined by element dimensions and screen width	the object on the screen field
	Y	Position value of the element along the Y-axis. Element position is determined by top-left corner coordinates. Coordinate counting starts at 0 and goes top to bottom; final value is determined by element dimensions and screen height	
Text settings	**Horizontal alignment**	Alignment of the text displayed in the element: left, center, right	
	Vertical alignment	Alignment of the text displayed in the element: top, center, bottom	
	Font size	Size of text displayed in element: 16, 32, 48	
	Text "True"	The text to display when the button is pressed	
	Text "False"	The text to display when the button is released	
	Text color "True"	The color of the text displayed when the button is pressed	
	Text color "False"	The color of the displayed text when the button is released	
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which will determine the visibility of the element.	
Different	**Name**	Displayed in the list of used elements on the left side of the visualization editor	
Dimensions	**Width**	Width of element along X-axis	The size of an element can be set by changing the boundaries of the object on the screen field.
	Height	Height of element along Y-axis	

7.7.12 Switch

The **Switch** element is designed to control the state of a Boolean variable, as well as to display its current state.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

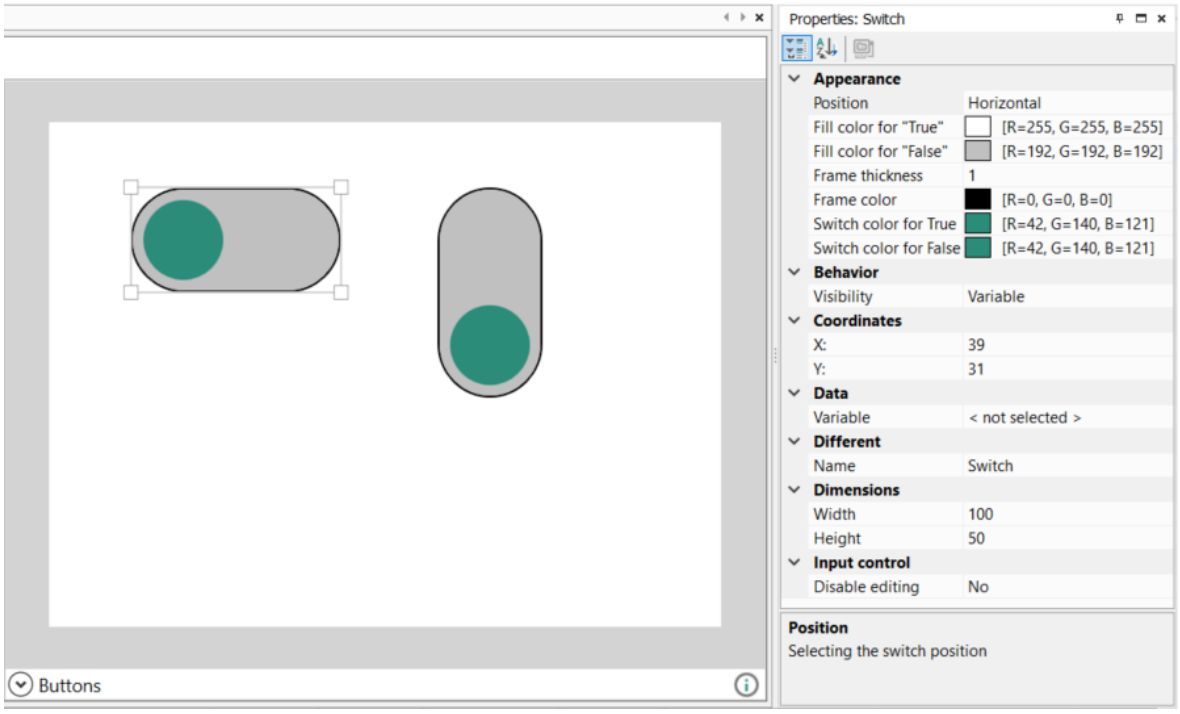


Fig. 7.3 Appearance and Properties of the Switch element

Coordinates

The location of the button is determined by the coordinates of its center along the X and Y axes. The coordinates are set in the properties panel or by moving the object across the screen. The coordinates for both axes start at 0:

– along the X axis - from left to right, the final value is determined by the size of the element and the width of the screen;
– along the Y axis - from top to bottom, the final value is determined by the size of the element and the height of the screen.

Dimensions

The dimensions of an element are determined by the X and Y axes and are set in the properties panel or by changing the boundaries of the element on the screen.

Properties

To configure the required type of **Switch** element on the device screen, the following properties are available:

Group	Parameter	Description	Additional Options
Appearance	**Position**	Vertical or horizontal switch placement	
	Fill color for “True”	Fill color of the element if the switch is on	
	Fill color for “False”	Fill color of the element when the switch is off	
	Frame thickness	Element outline thickness	
	Frame color	Element outline color	
	Switch color for True	The color of the switch when it is on	
	Switch color for False	The color of the switch when it is off	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional Options
Data	**Variable**	A Boolean variable is bound to the parameter	
Input control	**Disable editing**	Enabling or disabling the ability to change the switch value from the device screen	
Coordinates	**X**	Position value of the element along the X-axis. Element position is determined by top-left corner coordinates. Coordinate counting starts at 0 and proceeds left to right; final value is determined by element dimensions and screen width	Coordinates can be set by moving the object on the screen field.
	Y	Position value of the element along the Y-axis. Element position is determined by top-left corner coordinates. Coordinate counting starts at 0 and proceeds top to bottom; final value is determined by element dimensions and screen height	
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which controls the element's visibility	
Different	**Name**	It appears in the list of used elements on the left side of the visualization editor	
Dimensions	**Width**	Width of element along X-axis	Element dimensions can be set by changing object boundaries on the screen field. Switch proportions are maintained according to the following rules: – horizontal position: Length = Width × 2 – vertical position: Length = Width / 2
	Height	Height of element along Y-axis	

7.7.13 Switch group

The **Switch group** element is designed to manage parameter lists and display the status of these lists:

– **Radio button** – allows selection of only one parameter from the list.
– **Checkbox** – allows selection of one or multiple parameters from the list.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

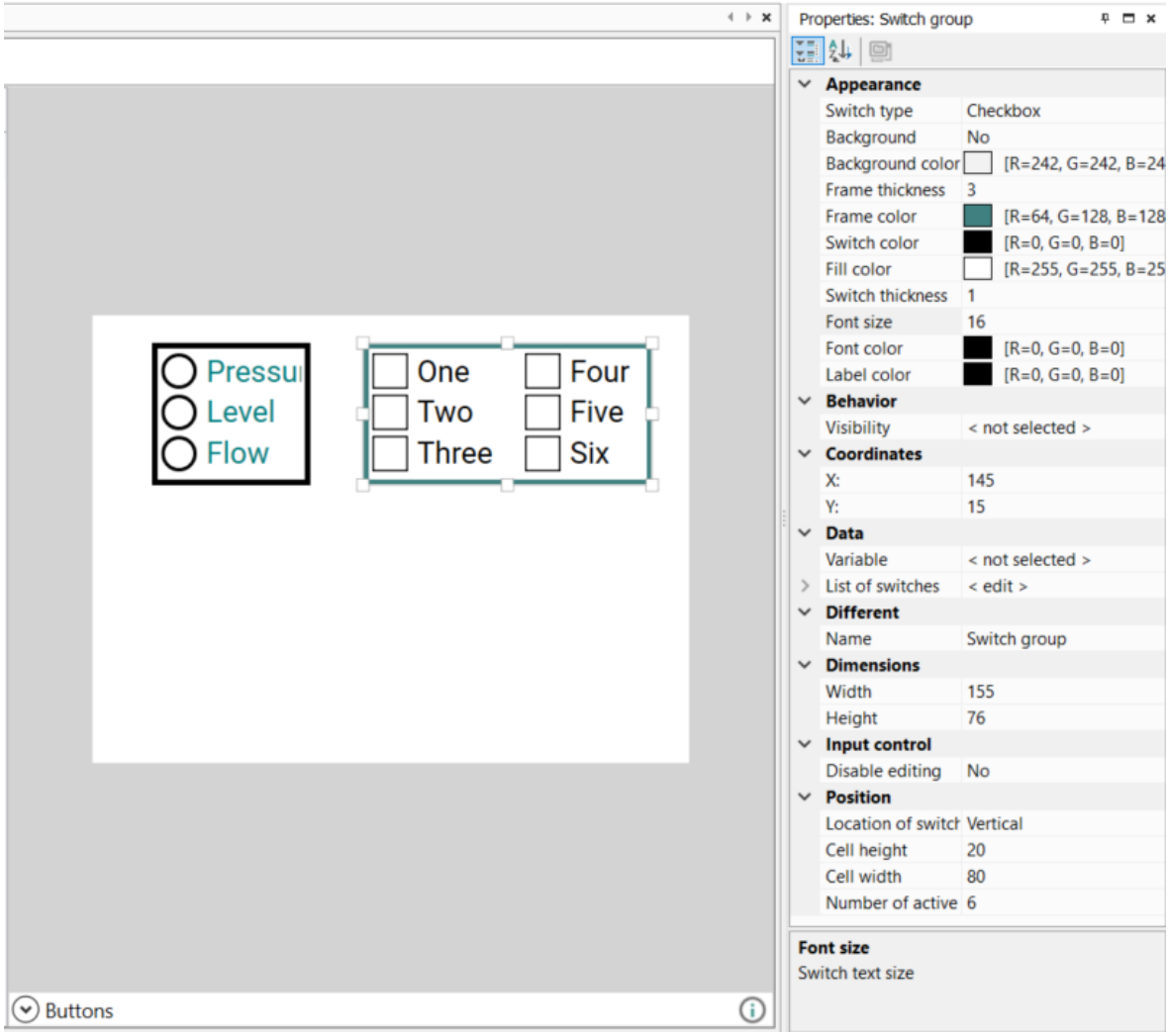
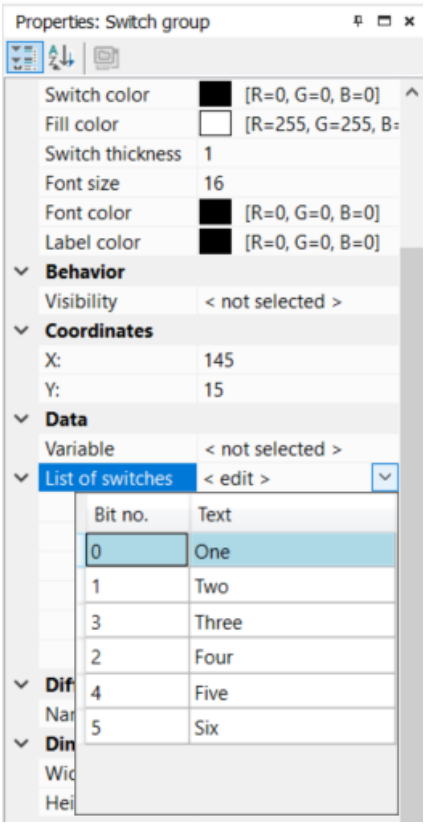
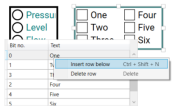


Fig. 7.4 Appearance and Properties of the Switch Group element

Configuring Properties

Group	Parameter	Description	Additional Options
Appearance	**Switch type**	Select the switch type: radio button or checkbox	
	Background	Presence or absence of a background for the text displayed in the element.	
	Background color	The background color of the text displayed in the element	
	Frame thickness	Element frame thickness	
	Frame color	Element frame color	
	Switch color	Label field border color	
	Fill color	Radio button/checkbox fill color	
	Switch thickness	Label field border thickness	
	Font size	Size of text displayed in element: 16, 32, 48	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional Options
	Font color	The color of the text displayed in the element	
	Label color	The color of the label being set	
Data	**Variable**	An integer variable is bound to the parameter.	
	List of switches (max. 32 lines)	The variable's values and their corresponding text to be displayed in the element. To add rows, click "Edit" and use the context menu of the opened table:  Specify the variable value, as well as text for display in the element	Adding rows to the parameter list is also available on the screen field: 1. Double-click LMB on the element placed on the screen 2. To add rows, use the context menu of the opened table:  3. Specify the variable value, as well as text for display in the element
Input control	**Disable editing**	Enabling/disabling the ability to interact with the element from the device screen: – Switching between list parameters (for the radio button element) – Enabling/disabling list parameters (for the checkbox element)	
Coordinates	**X**	Position value of the element along the X-axis. Element position is determined by top-left corner coordinates. Coordinate counting starts at 0 and proceeds left to right; final value is	Coordinates can be set by moving the object on the screen field

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional Options
		determined by element dimensions and screen width	
	Y	Position value of the element along the Y-axis. Element position is determined by top-left corner coordinates. Coordinate counting starts at 0 and proceeds top to bottom; final value is determined by element dimensions and screen height	
Behavior	**Visibility**	A Boolean variable is bound to the parameter, which controls the element's visibility	
Position	**Location of switches**	Selection of the list layout: horizontal or vertical	
	Cell height	The height of the cell for arranging list items	
	Cell width	The width of the cell for arranging list items	
	Number of active checkboxes	Restricts the simultaneous selection of multiple parameters from the list. The limit cannot exceed the number of rows added to the Switch List. Available only when the switch type "Checkbox" is selected.	
Different	**Name**	It appears in the list of used elements on the left side of the visualization editor	
Dimensions	**Width**	Width of element along X-axis	Element dimensions can be set by changing object boundaries on the screen field
	Height	Height of element along Y-axis	

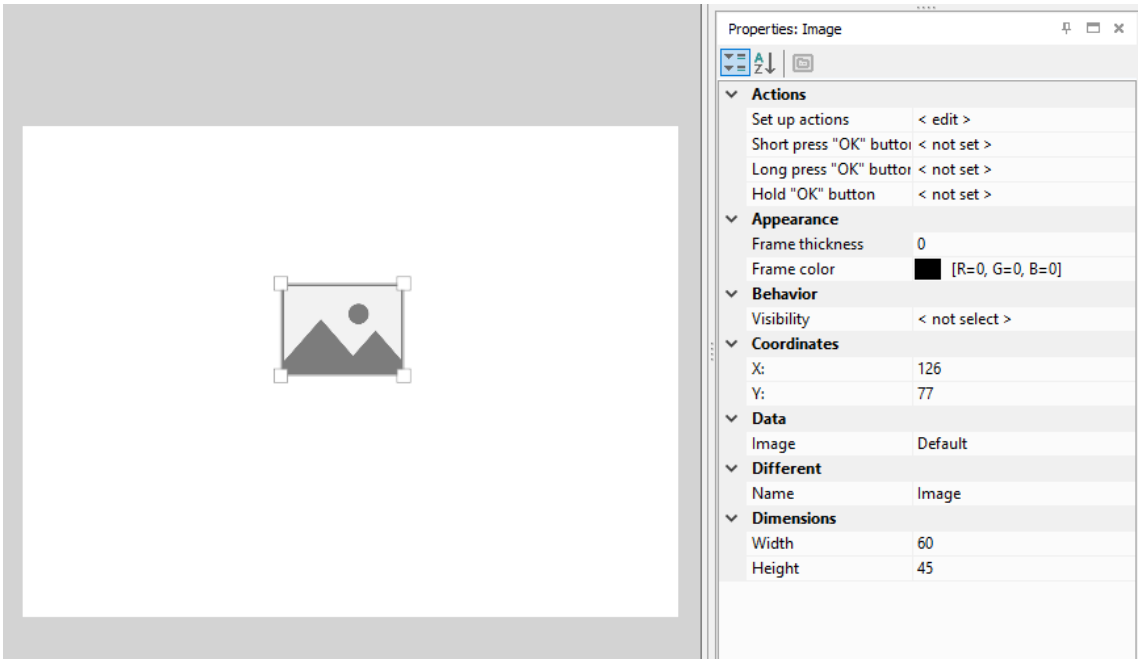
7.7.14 Image

Adding a Custom Image to a Project

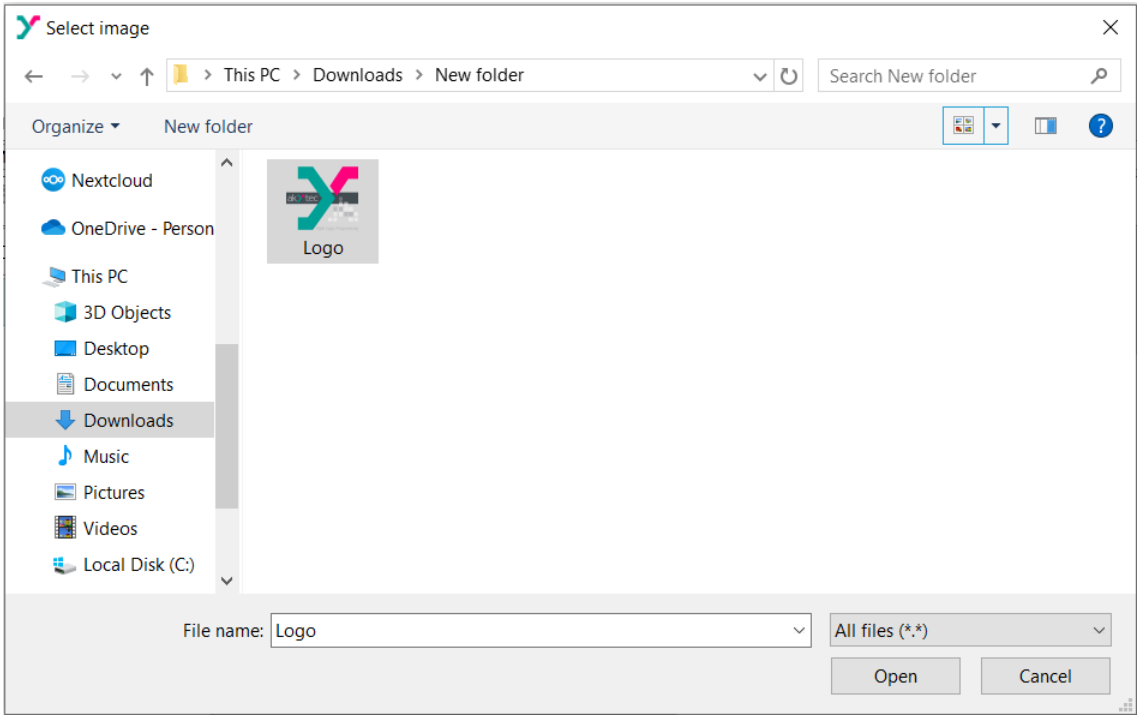
To add a custom image:

1. Place an **Image** object on the workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. Double-click the placed object with the left mouse button. The image selection window will open.



NOTE
Supported file types: *.jpg, *.jpeg, *.jpe, *.bmp.
The maximum image size is limited by the device's screen resolution. Images can be scaled down within the ALP editor if necessary.

3. Select an image and click **Open**.

Configuring Properties

To ensure the image displays correctly on the device screen, configure its location, dimensions, and other parameters in the properties panel.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Coordinates

The element's position is determined by the coordinates of its center along the X and Y axes. Coordinates can be set in the properties panel or by dragging the object on the screen. The coordinate origin (0,0) is the top-left corner:

– **X-axis:** Increases from left to right. The maximum value is constrained by the element's size and the screen width.
– **Y-axis:** Increases from top to bottom. The maximum value is constrained by the element's size and the screen height.

Dimensions

The element's size is defined by its width (X-axis) and height (Y-axis). Dimensions can be set numerically in the properties panel or adjusted visually by dragging the element's boundaries on the screen.

Parameters

Group	Parameter	Description
Appearance	**Frame thickness**	Sets the thickness of the image's border.
	Frame color	Sets the color of the image's border.
Behavior	**Visibility**	Bind a Boolean variable to control the element's visibility. The element is visible when the variable is TRUE.
Data	**Image**	Selects the image file to display.
Different	**Name**	The name displayed for this element in the list of used elements on the left side of the visualization editor.
Actions	**Set up actions**	Select an action to execute, such as navigating to another screen.
	Short press “OK”	Defines actions triggered by a short press of the ***OK*** button.
	Long press “OK”	Defines actions triggered by a long press of the ***OK*** button.
	Hold "OK"	Defines actions triggered by holding down the ***OK*** button.

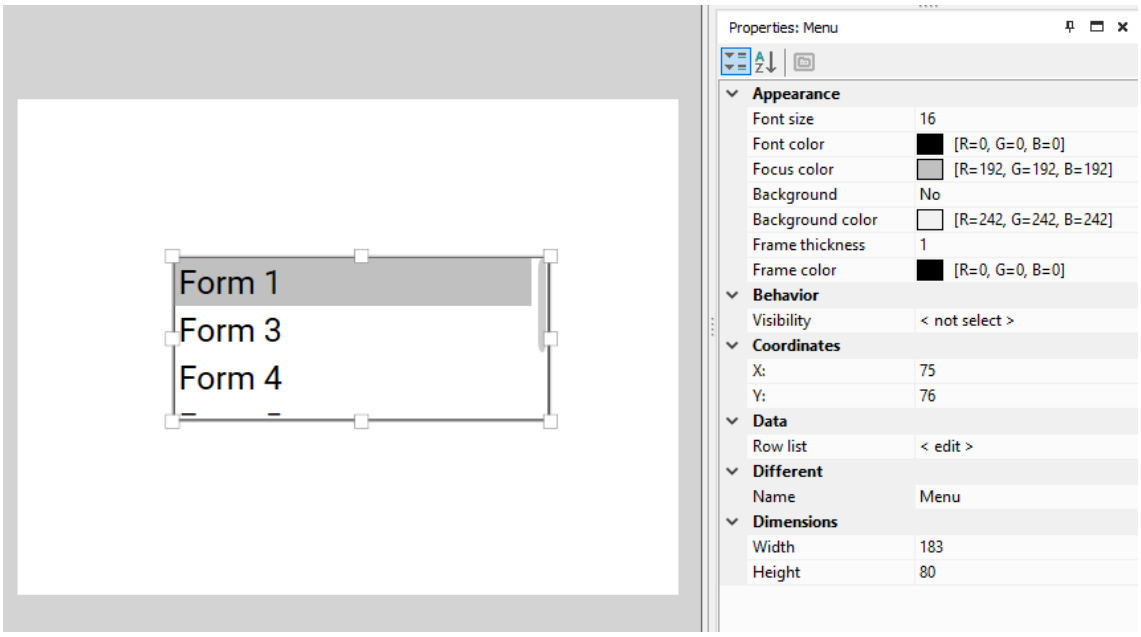
NOTE
If the same image file is linked to multiple elements, it only occupies space once in the user visualization ROM. This is reflected in the **User visualization ROM** indicator in the status bar.

7.7.15 Menu

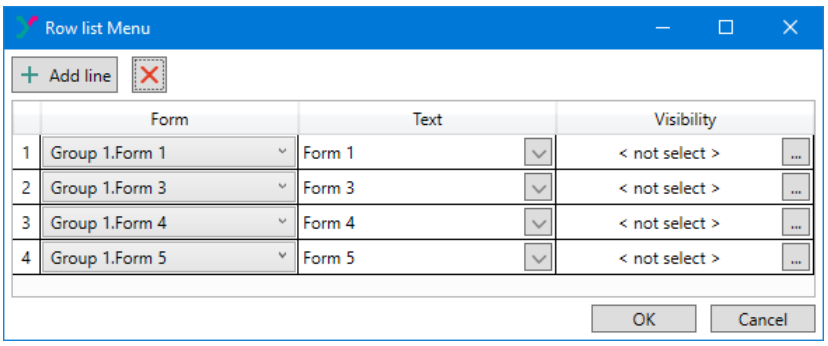
The **Menu** element is designed for navigation between visualization screens. Each menu item is linked to a specific screen.
To add a **Menu** object to a visualization screen:

1. Place a **Menu** object on the workspace.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. Double-click the placed object with the left mouse button (LMB). The line editing window will open.



3. Add the required number of menu items (lines).

NOTE
Maximum: 128 lines.

4. In the **Screen** column, select the screen to navigate to when the menu item is selected.

NOTE
All project screens except the current one are available for selection.

5. The name of the selected screen is automatically displayed in the **Text** column. To change the text, click the marker located at the end of the row, and enter the text to be displayed for the menu item in the field that opens.
6. Set the visibility for each item by binding Boolean variables in the **Visibility** column.
7. Click the **OK** button to confirm and close the window.

Configuring Properties

Group	Parameter	Description	Additional features
Appearance	**Font size**	Size of the text displayed in the element: 16, 32, 48	
	Font color	Color of the text displayed in the element	

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Group	Parameter	Description	Additional features
	Focus color	Background color of the focused menu row	
	Text background	Enables or disables a background behind the text	
	Background color	Color of the text background	
	Frame thickness	Thickness of the element's border	
	Frame color	Color of the element's border	
Data	**Row list**	Opens the line editing window to configure menu items	
Coordinates	**X**	The value of the element's position along the X axis. The location of the element is determined by the coordinates of the upper left corner. The coordinates are counted starting from 0 and going from left to right, the final value is determined by the size of the element and the width of the screen.	Coordinates can be set by moving the object across the screen.
	Y	The value of the element's position along the Y axis. The location of the element is determined by the coordinates of the upper left corner. The coordinates are counted starting from 0 and going from top to bottom, the final value is determined by the size of the element and the height of the screen.	
Behavior	**Visibility**	If you select **Yes**, the **Menu** element will have focus after entering the screen. Only one element with the value **Focused by default** is allowed on a single screen - **Yes**	
Different	**Name**	The name displayed for this element in the list of used elements on the left side of the visualization editor.	
Dimensions	**Width**	Element width along the X axis	
	Height	Element width along the Y axis	

7.7.16 Chart

The **Chart** element is designed to display the change in variable values in graphical form on the device screen.

– Up to four variables can be displayed simultaneously (data types – int and float).
– The polling time interval for drawing the chart can be configured (the maximum allowed value is 23h59m59s).
– The polling period is calculated automatically, depends on the maximum value of the x-axis, the width of the **Chart** element, and the border thickness, and must not be less than 1 second.
– The number of slices is determined by the following formula:

  Number of slices = max. X / t poll, where:

  – max. X – maximum value of the x-axis;
  – t poll – polling time.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
Chart plotting is only available for devices with color LCD. Chart operation in simulation mode is not supported.

To add a graph to the screen, select the **Chart** element in the Library Box and place it in the working area.
The coordinate axes will appear with the scale and grid displayed. The X axis is the time axis, the Y axis is the axis of variable values (udint or float).

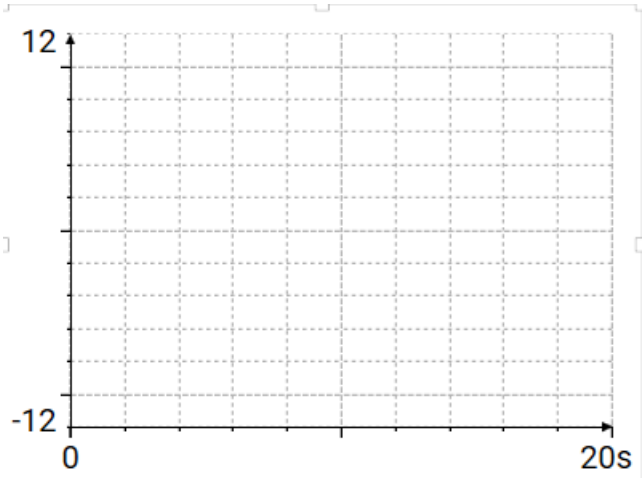


Fig. 7.5 Coordinate axes with default values

To configure parameters and add variables, use the **Property Box**:

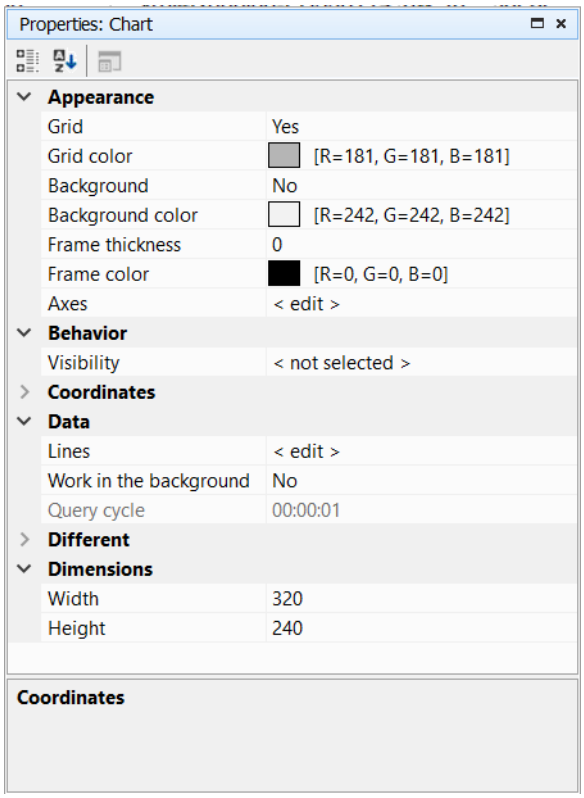


Fig. 7.6 Graph Properties Window

Customize the **Appearance** of the graph:
– presence and color of the grid

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– presence and color of background
– presence, thickness and color of the frame
– axes:

Select the Axes line and left-click on <Edit> and then on the button that appears ... . The Axes Graph window will open:

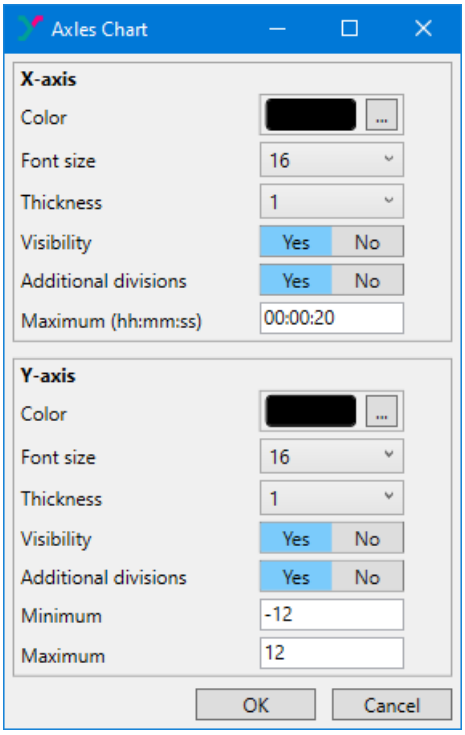


Fig. 7.7 Axis Window

If necessary, adjust the color and thickness of the axes, font size, visibility of the axes and additional tick marks. Enter the desired maximum value of the X-axis in the format hh:mm:ss. The maximum allowed value is 23:59:59. The default value is 20 seconds.

NOTE
Minimum value of X axis = 0 – non-editable parameter.

Enter the minimum and maximum values for the Y-axis. Default value: -12, 12.

If you enter an incorrect value, an error message will appear and the OK button will become inactive:

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

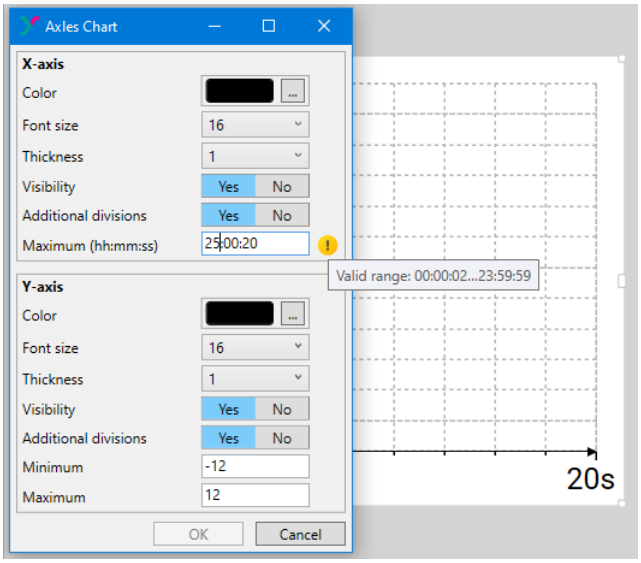


Fig. 7.8 Data entry error

To add variables to the chart, in the **Data** section of the Graph Properties window, select the Lines line and left-click on <Edit> and then on the button that appears ..., or double-click on the chart field. The List of lines Chart window will open:

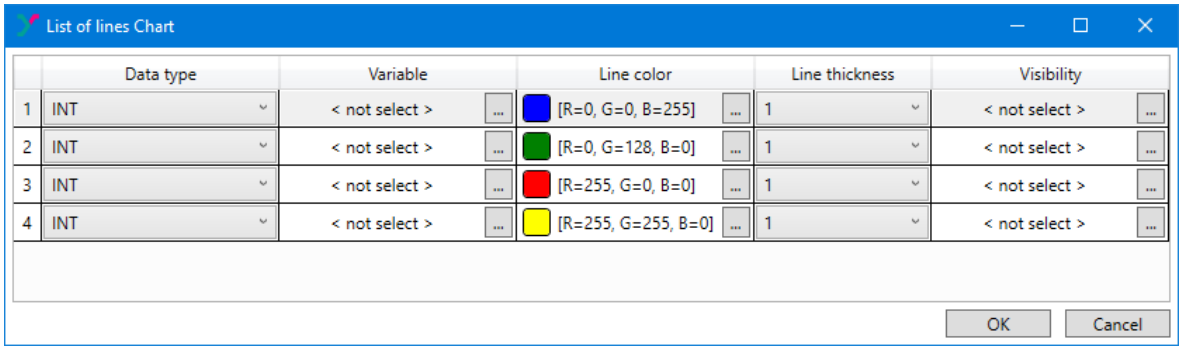


Fig. 7.9 List of lines – Chart window

In the window that opens, select:

– data type – integer / floating point
– variable
– color and thickness of the line
– visibility – it is possible to select a Boolean type variable from the Standard Variable Tables category.

NOTE
Maximum number of variables on a graph - 4.

In order to easily show the number of added variables and the settings for the lines' appearance on the chart, the lines are displayed with an offset along the Y axis relative to each other.
The Run in background (Yes/No) option in the **Data** section affects the saving of data when switching to another screen.
The polling period is a non-editable parameter, it depends on the maximum value of the X-axis, the width of the Chart element and the thickness of the frame.

NOTE
The polling period cannot be less than 1 second.

The position of the chart is determined by the coordinates of the upper left corner along the X and Y axes. The coordinates are set by moving the object across the screen. If you need a precise position

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

of the graph on the device screen, set the coordinates of the Property Box in the **Coordinates** section.
If necessary, in the **Behavior** section, configure the visibility of the entire graph depending on the Boolean variable.
You can change the name of a chart in the **Different** section of the Property Box.
To scale the chart, move the mouse over the corner or side marker holding down the left mouse button or set the required graph size on the screen. You can also change the size in the **Dimensions** section: the width and height are set in pixels.
After *uploading the program* the device screen will display a graph with the added variables:

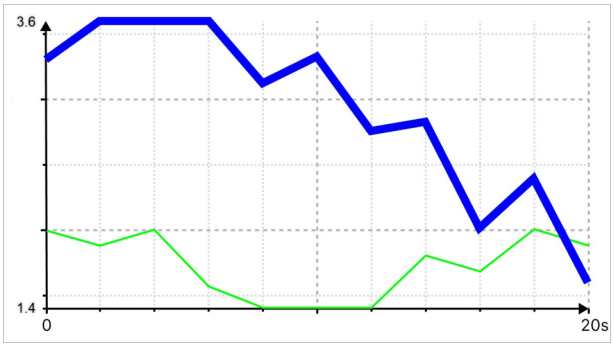


Fig. 7.10 Chart on the device screen

NOTE
If the parameter value goes beyond the minimum or maximum value of the Y axis on the device screen, the line on the chart will continue to move along the corresponding boundary.

If a variable added to the graph is deleted from the project, an information window will appear:

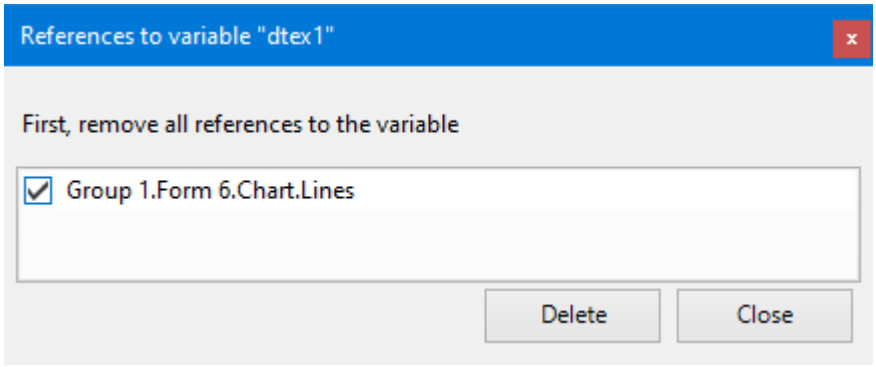


Fig. 7.11 Information window

If you select **Delete** in the Lines Graph window, the variable binding is deleted.

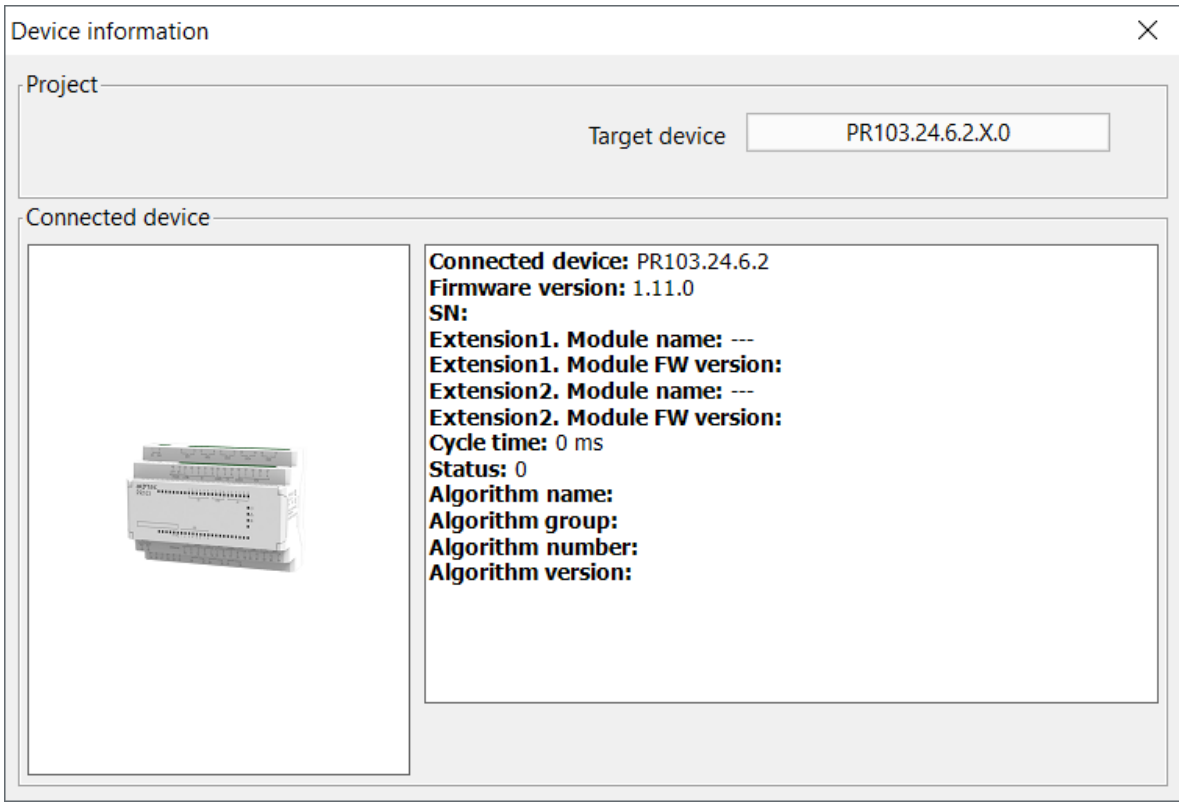
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

8 Device

This section describes the operating functions and configuration of the device:
– *Device information*
– *Cycle time*
– *Firmware update*
– *Calibration*

8.1 Device information

To view information about the software, the target device and the connected device use the menu item ***Device*** → ***Information...*** or the icon in the toolbar.
A window containing information about the connected device appears:



The window ***Device Information*** contains the following information:
Target device – the device for which the project was created
Connected device – the information about the device connected to the PC
Alternatively, the type of each output can be manually changed in ***Property Box*** in accordance with the hardware.

Information about the device on the new hardware platform

For devices on the new platform, the information displayed in the window differs.
Project information:
– ***Selected device model*** - the model and modification of the device selected when creating the project.

Information about the connected device:
– ***Device name*** – model and modification of the connected device
– ***Firmware version*** – firmware version of the connected device
– ***S/N*** – unique device identifier
– **PRM Slot. Module name** – model of the *extension module*, connected to the device

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– **PRM Slot. Module Firmware Version** – firmware version of the *extension module*, connected to the device.

8.2 Cycle time

Cycle time is the time it takes to complete the operating cycle of the device, namely:
– polling the state of the physical inputs of the device and copying their values into memory cells
– program processing
– read/write program network variables
– writing the results of the program to the physical outputs of the device

The default cycle time is **1 ms**. The device adjusts the cycle time depending on the complexity of the program.
Conditions for increasing cycle time:
– the complexity of the algorithm increases (a large number of FBs and macros are involved)
– the program uses a large number of network variables
– the project uses a large number of data controls via the device display

The user cannot set the cycle time. If the device is equipped with a display, the current cycle time can be viewed in the system menu of the device. If the device is connected to a PC, the cycle time is shown in the *Device Information* window.

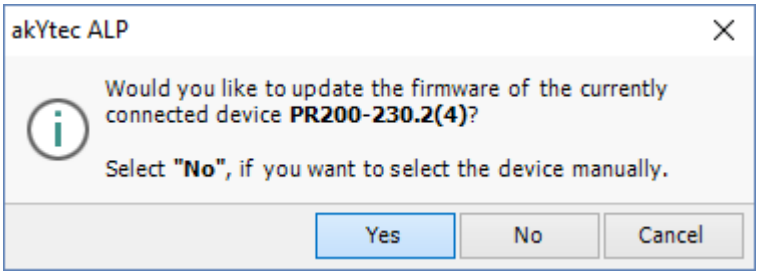
8.3 Firmware update / repair

If a new ALP version includes a new version of the firmware for the connected device or extension module, a prompt appears to update the firmware before uploading a user program to the device. No internet connection is needed. Click ***Yes*** to start the update.

NOTE
Ensure the power supply of the device and extension modules (if any) and the safe connection between the PC, the device and the extension modules (if any) during the update process.

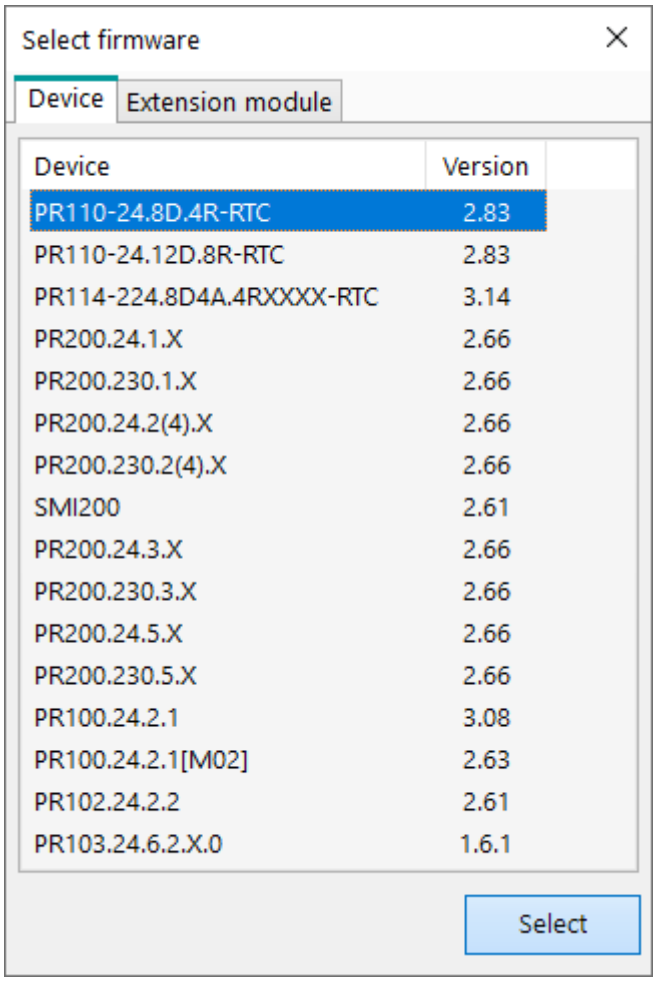
You can also update the firmware manually using the menu item ***Device > Firmware update***. This way the firmware can be repaired when the firmware damage is detected (see respective user guide, table “Error indication”).

NOTE
The user program will not be affected by firmware update.



If you select ***Yes***, the firmware of the currently connected and recognized device will be updated (repaired).
If you select ***No***, lists of devices and extension modules will be offered to select from. The opened window has two tabs: ***Device*** and ***Extension Module***. This way a forced firmware update can be made.
Click ***Select*** to confirm the selection and start the update (repair) process. The message about the update result is shown upon the update completion.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Forced firmware update / repair

If the firmware is damaged (see respective user guide, table “Error indication”) and device automatic recognition is not possible, a forced firmware update should be used. Proceed as follows:

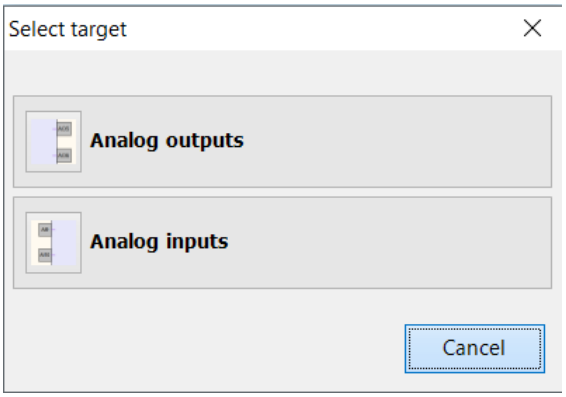
1. set the device in the forced download mode (see the device user guide)
2. select the menu item ***Device > Firmware update***, lists of devices and extension modules will be offered to select from
3. select the device (extension module)
4. click ***Select*** to confirm the selection and start the update (repair) process.

The message about the update result is shown upon the update completion.
If the device and the extension module have incompatible firmware versions and the user program is uploaded to the device without the extension module connected, this may lead to an expansion module error being displayed. To fix the error, use forced firmware update for the expansion module as described, skipping step 1.

8.4 Calibration

Only general information about calibration of analog inputs or outputs is given in this section. For detailed information about calibration refer to the user guide of the device.
If calibration of analog inputs or outputs is necessary, use the menu item ***Device*** → ***Calibration…***.
The item is active only if a device is connected. Select the calibration target (inputs or outputs) in the opened dialog.

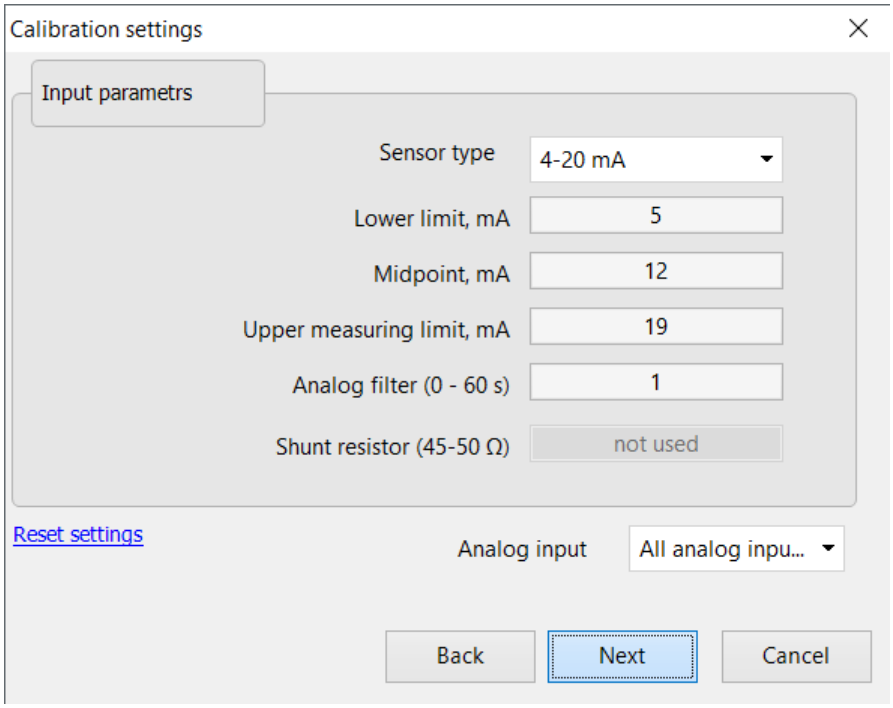
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

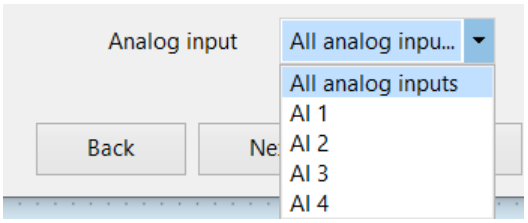


After selecting the calibration target, device stops the execution of the program. The program starts again when the calibration completes successfully.

8.4.1 Input calibration

To calibrate inputs, connect a reference signal source to them. Start calibration, select the type of signal connected to the input and set the calibration parameters in the opened dialog.



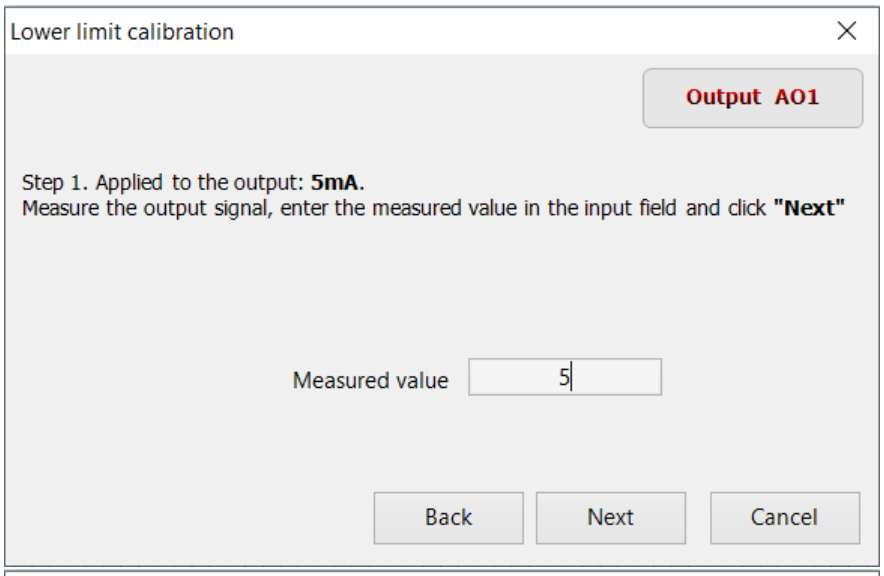


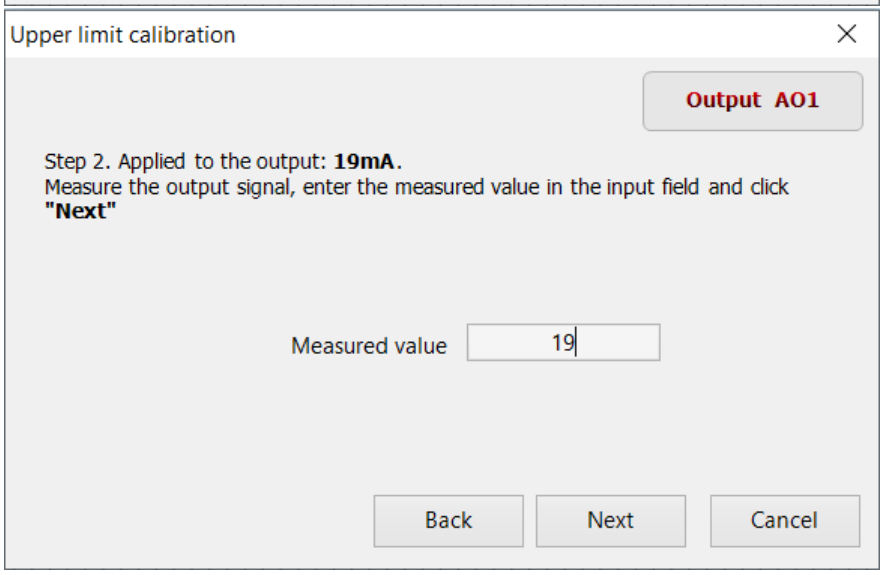
Use the item ***Reset settings*** to apply the default settings for calibration.
Use the list ***Select input*** to select the input to calibrate, click the button ***Next*** and follow the instructions.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

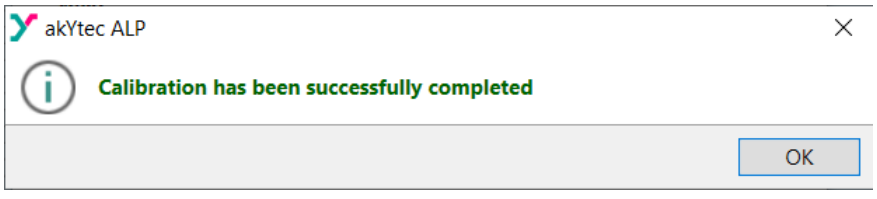
8.4.2 Output calibration

Before calibrating an analog output, prepare the appropriate measuring device, than start calibration and follow the instructions. Measure the signal at the output indicated at the top right of the window and enter the value in the input field.
Proceed the same way with the other outputs if needed. The message about the calibration results will appear after the completion of the calibration.





Proceed the same way with the other outputs if needed. The message about the calibration results will appear when the calibration is complete.



8.5 Change target device

The **Change Target device** function is designed to transfer a project to another model of a device from the same generation of devices.
The change of target platform occurs according to the following rules:

– The canvas automatically adjusts to changes in the number of inputs/outputs.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

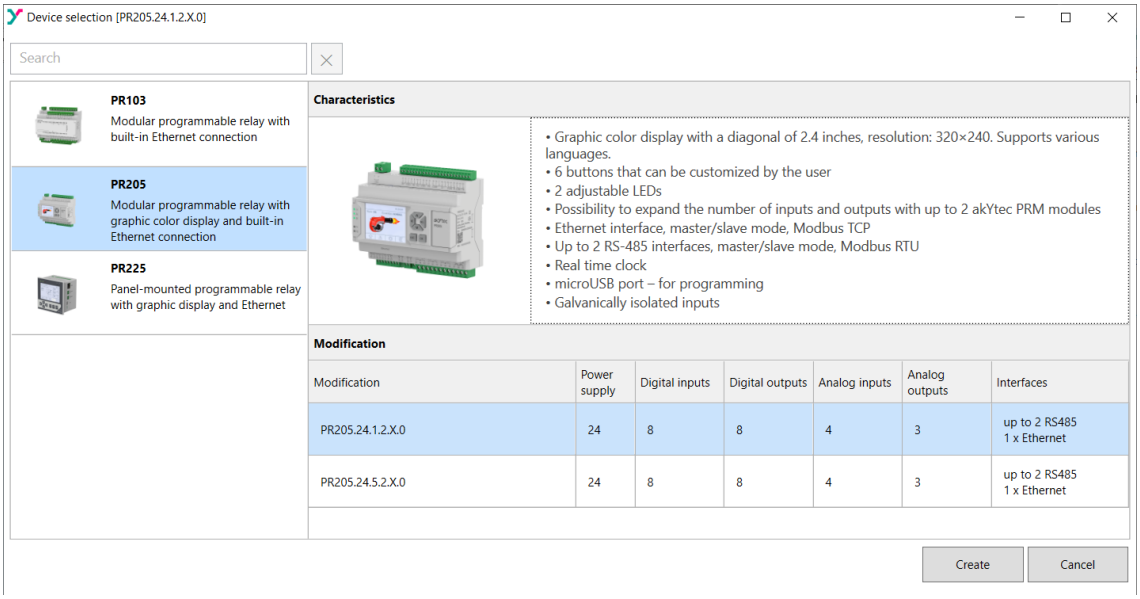
- The user-configured I/O layout remains. New I/O are added after the existing I/O of the original project.
- Input/output connections whose data type has changed are broken.
- If the number of inputs/outputs increases but the data type of the original set of inputs/outputs does not change, then the connections remain.
- If the number of inputs/outputs decreases compared to the original, then the connections of remote inputs/outputs are broken.
- If extension modules were added to the original device, they are transferred to the target device, the connections for them remain. If an extension module cannot be connected to the target device, a window will open with the information that the references to the variables bound to the extension modules will be deleted.
- Analog input/output settings are transferred (if there are analog outputs on the target device).
- Communication interfaces are transferred without changes. If the target device for the project transfer does not support an interface, a window will open with the information that the interface, network variables and references to the variables will be deleted.
- All screens and their settings are transferred from one target platform to another. If the target device for the project transfer does not have a display, a window will open with information about the removal of references to variables bound to the visualization elements.

  NOTE
  When changing the target platform from PR225 to PR205, visualization elements will be scaled proportionally to the change in screen size.
  When changing the target platform from PR205 to PR225, visualization elements are transferred without scaling.

- All variables are transferred to the target device. If the variables are assigned to the parameters not available in the target device, then a window will open with the information that the references to the variables bound to the parameters will be deleted.
- If the target device does not support the FB contained in the source project, an information window will appear notifying you of the need to delete the FB.

To change the target platform, you should:

1. Open the project to be transferred to another device or device model.
2. Select ***File*** → ***Change Target device*** from the menu. ALP will prompt you to save the project before changing the device. Next, a menu will appear with a list of available device models:



3. Select the new device. Click the ***OK*** button.
4. Check and restore broken connections, if any. You can check program's operation in the *Simulator*.
5. Save the modified project.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

9 Plugins

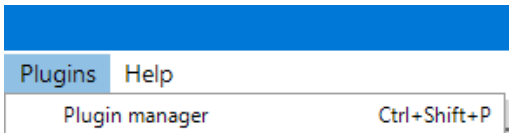
Plugins are used to create complex projects and integrate them with other akYtec services.
Plugins in ALP:

– *Replication master*. With this plugin you can load a program into the device without ALP and protect it from copying
– *Exporting the device to akYtec Cloud*. Allows you to export network variable configuration for polling in akYtec Cloud service

Installing plugins

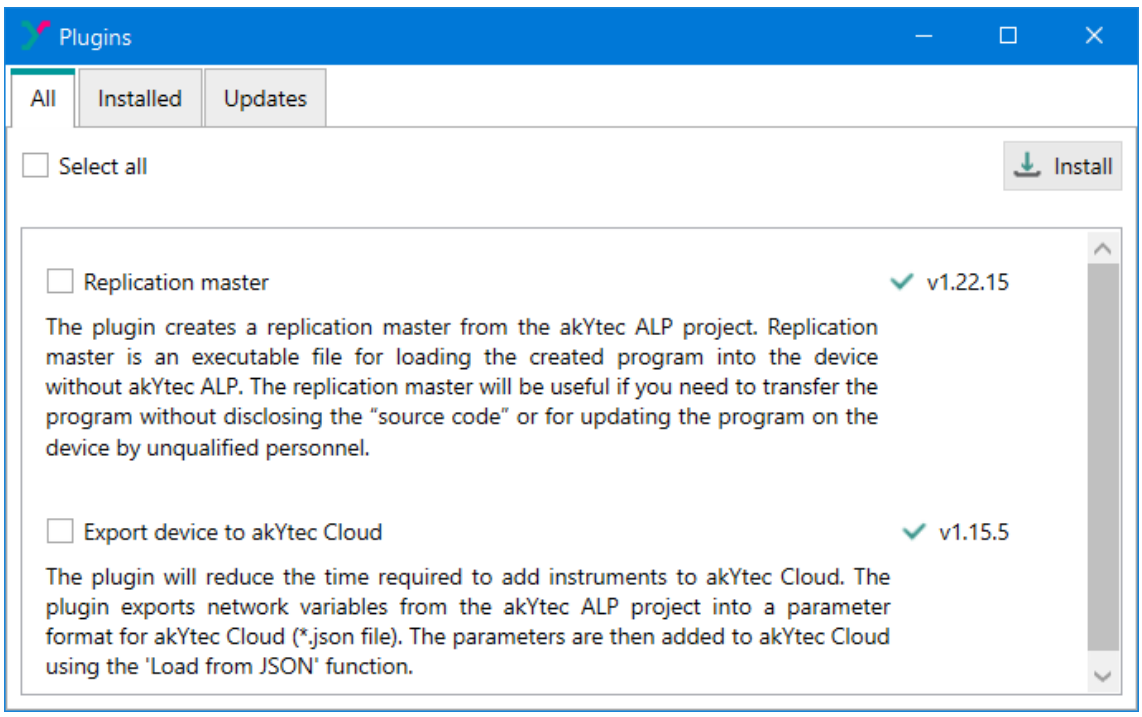
To install plugins:

1. Select ***Plugins*** → ***Plugin manager*** at the main menu.

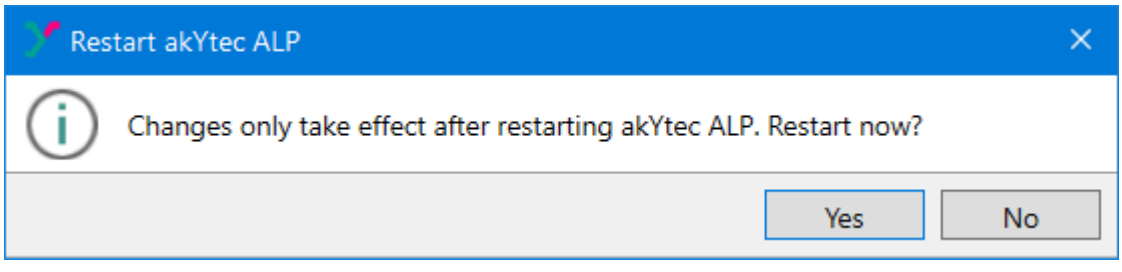


.

2. Check the required extensions in the ***Plugins*** window.



3. Click the **Install** button.
4. A green check mark ✓ next to the selected extensions shows that the installation is complete.
5. Close ***Plugins*** window. ALP restart dialog box appears.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

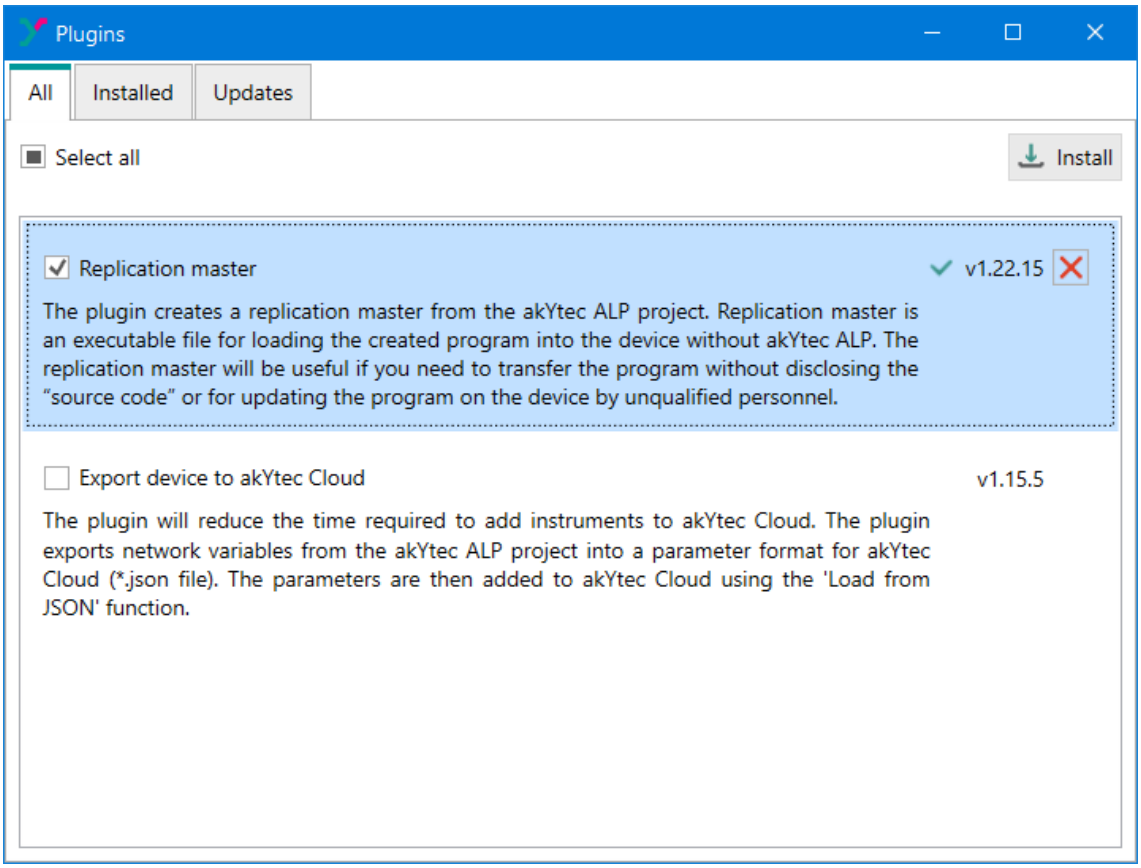
After restarting ALP, the extensions will be ready for use.

NOTE
If no extensions are installed, you should check if port 8084 is open.

Deleting extensions

To delete an extension:

1. Select ***Plugins*** → ***Plugin Manager*** at the main menu.
2. Check the box ✕ next to the desired extension in the ***Plugins*** window.



3. The delete icon will change to the install icon.
4. Close ***Plugins*** window. ALP restart dialog box appears. After ALP is restarted, the deleted extensions will no longer be available in the main menu .

9.1 Replication master

The Replication master allows you to create a program file from an ALP project with one of the extensions:
**.exe* – for Windows;
**.dll* – for Linux.
The file can be used to load the program into the device without using the ALP.

CAUTION
The program can only be replicated between identical device modifications.

The result of writing a program to the device using the Replication master depends on whether the key is available in the device and the Replication master. The key can be written to the device using ALP.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

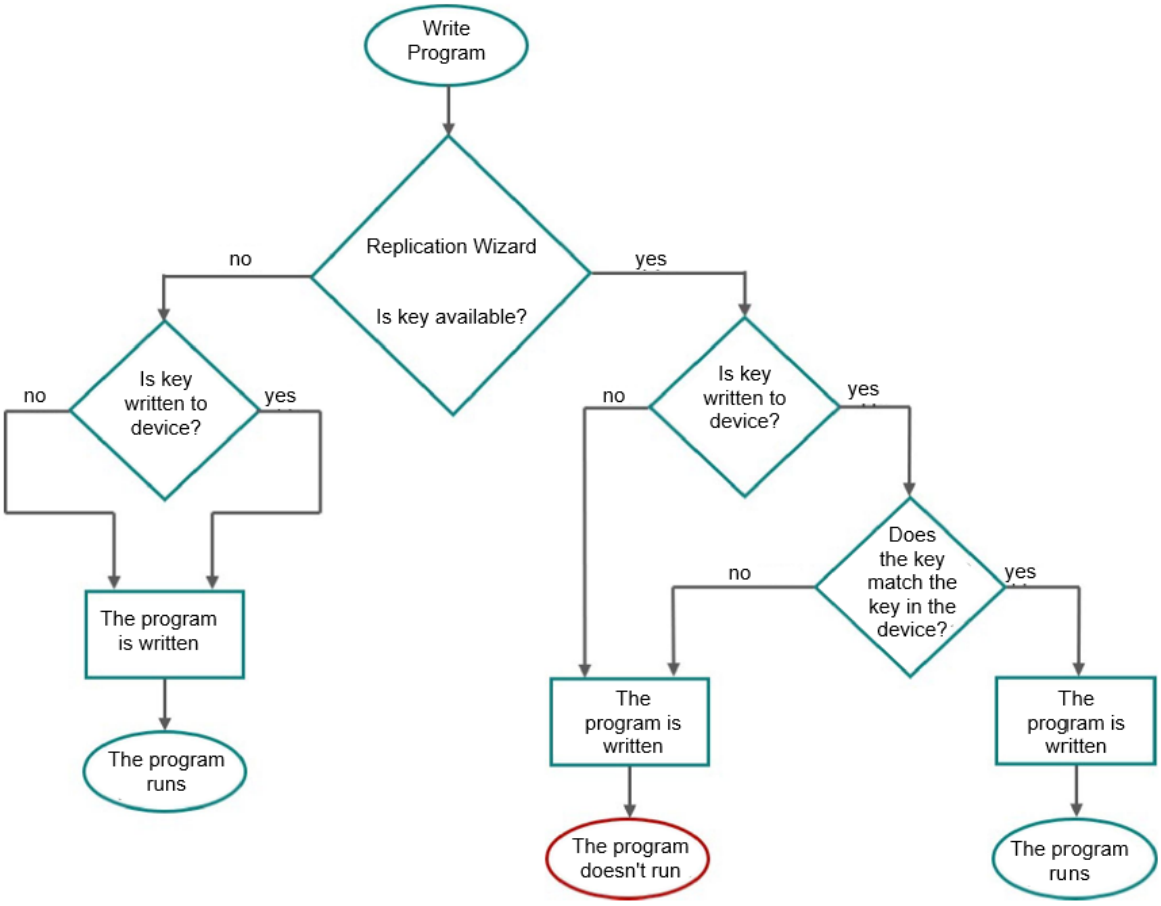


Fig. 9.1 Writing a program using the Replication master

Creating a Replication master

Before creating the Replication master for Windows, it is recommended to create an **.rtf* text file with the project description. The project description will be displayed during the Replication master startup before the program is loaded into the device.
To create a Replication master for Linux, you must install the .NET 6 SDK. This can be done when installing ALP. To do this, check the **Install .NET6 SDK** checkbox in the installation window:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

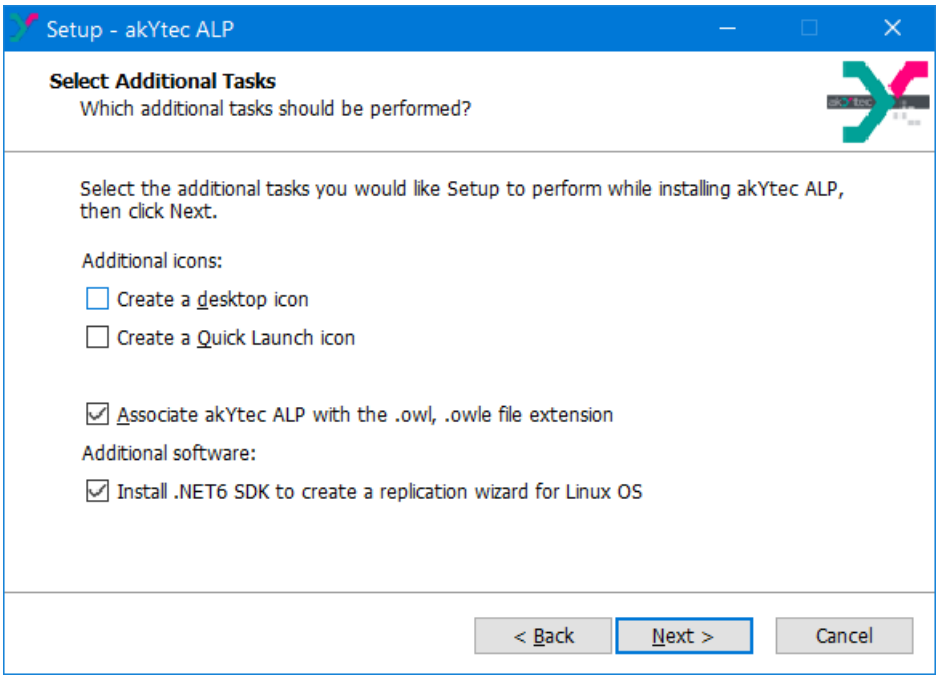


Fig. 9.2 ALP installation window

If the .NET 6 SDK has not been installed, an error will occur when creating the Replication master for Linux and the file will not be created. At the bottom of the window you will see a note about the installation being required and a link to download the package:

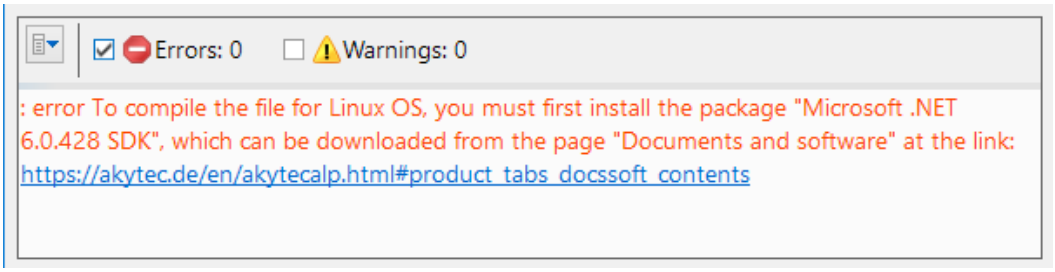
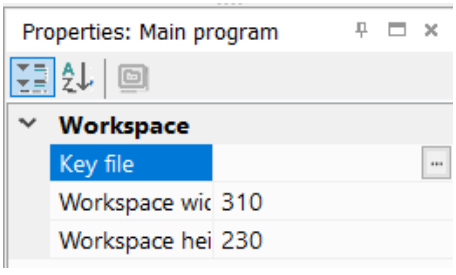


Fig. 9.3 File creation error

To create a Replication master:

1. In the **Key file name** field of the project properties panel, specify the *file containing the key*, stored on the PC. To download the program without a key, leave this field blank.



   If after specifying the path to the key file, the key is deleted, the Replication master will not be created, and error information will be displayed at the bottom of the window:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

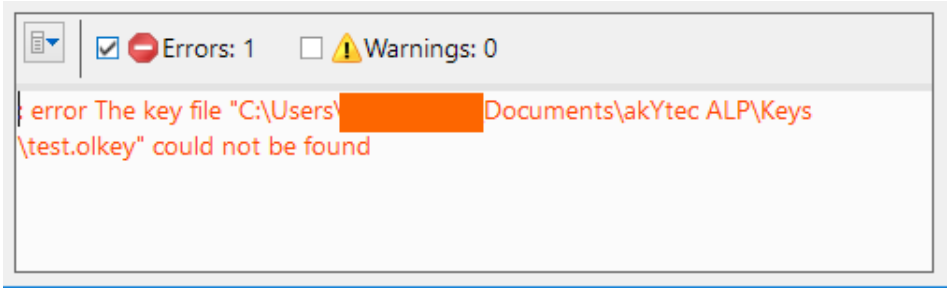


Fig. 9.4 File creation error

2. Select **Create Replication master** from the **Extensions** main menu.
3. A window will be opened for the Replication master creation. If a key is connected to the project, the message **User program is protected by a key!** will be displayed. Otherwise, the message **Device supports user program protection, but no key is selected for the project** will be summoned.

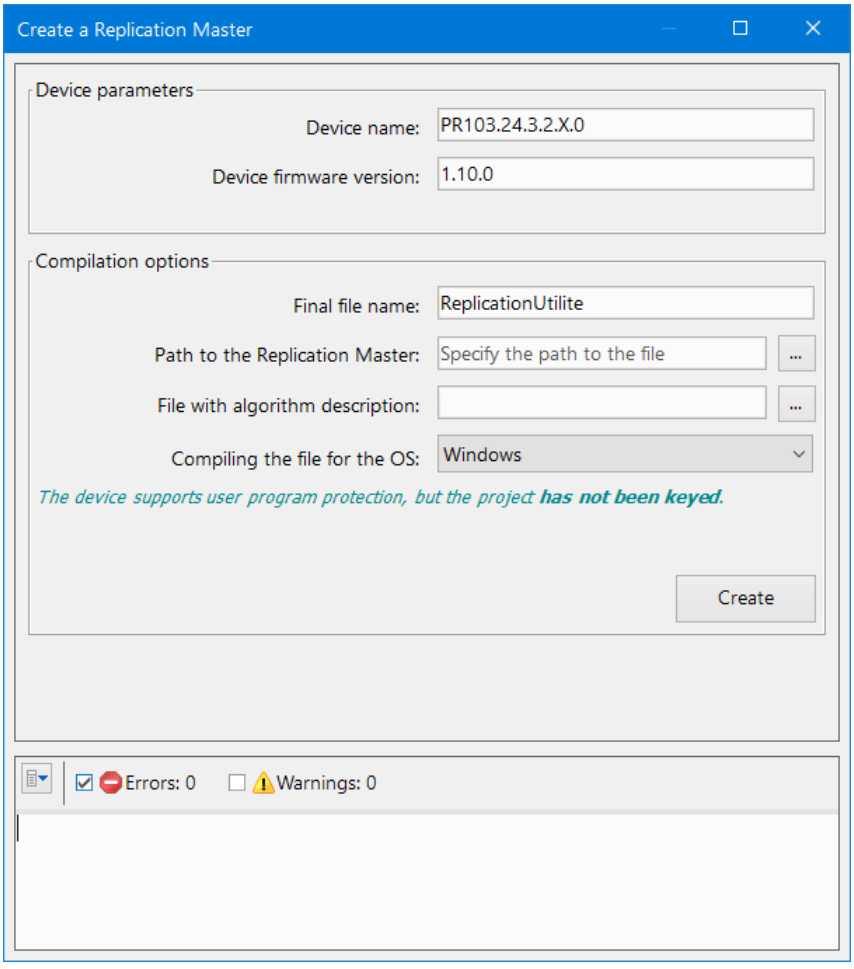


Fig. 9.5 Creating the Replication master

The **device parameters** are read from the device and filled in automatically.

In the compilation parameters window, fill in the fields:

– *Final file name* – name of new Replication master file
– *Path to the Replication master* – location of new file
– *File with algorithm description (For Windows)* – path to the text file with the project description in **.rtf* format (optional)
– *Compiling the file for the OS* – target operating system

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4. Click the **Create** button. If there are no errors, a file with the specified name and extension **.exe* or **.dll* will be created.

NOTE
Creating a Replication master for Linux can take a long time.

NOTE
The **.dll* file extension may not be displayed in Windows unless the option to display system files is enabled:

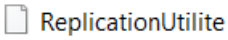


*Fig. 9.6 Displaying a file with ***.dll extension***

The file creation report is displayed at the bottom of the window.

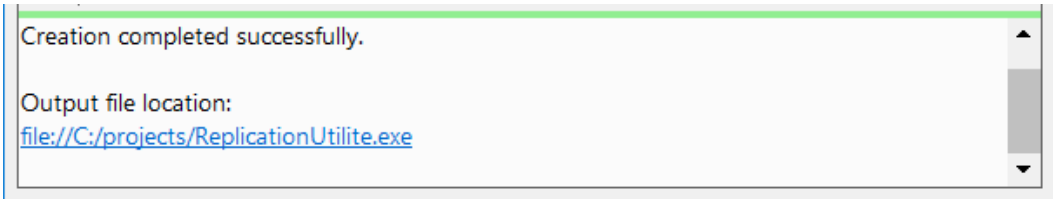


Fig. 9.7 File Creation Report

To hide or unhide the report display area, click the . button. Information about warnings and errors is displayed if the corresponding checkboxes are checked.

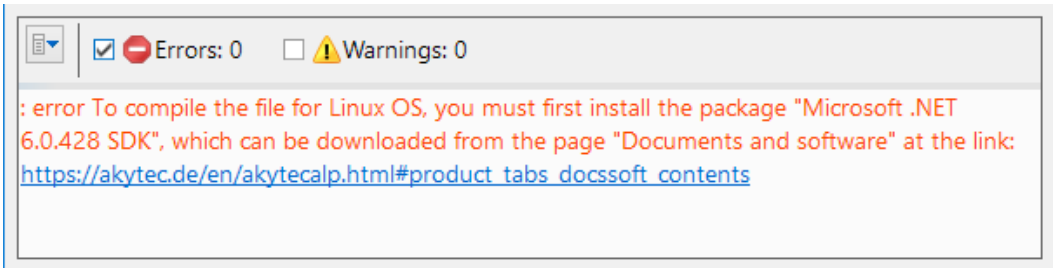


Fig. 9.8 Examples of warnings

9.1.1 Writing a program to the device via the Replication master on Windows

To write a program to the device using the Replication master, you should:

1. Run the created Replication master file.

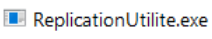


Fig. 9.9 Replication Master File

NOTE
When the master opens, the project description text from the attached file with the **.rtf* extension will be displayed. If the file was not attached when the master was created, the project description will be empty.

2. Follow the instructions of the replication master.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

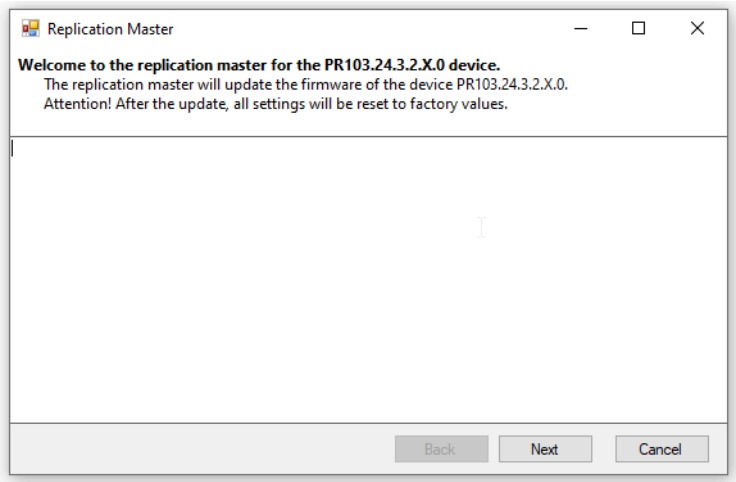


Fig. 9.10 Replication master window

3. In the Port settings section Replication master shows the number of the connected COM port. How to find out the port number is described in the section *Connection to device*.
4. If the device is password protected, enter the password in the Replication master window. If the password is missing or entered incorrectly, the **Next** button will be inactive. When the port is selected, connection status is shown in the lower right corner.

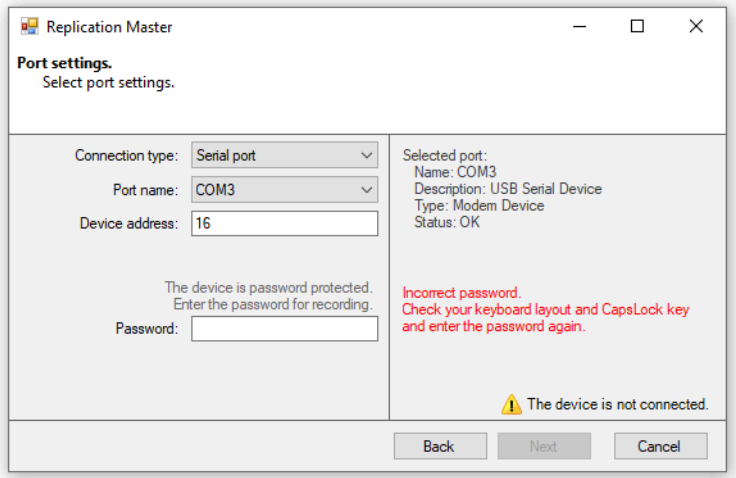


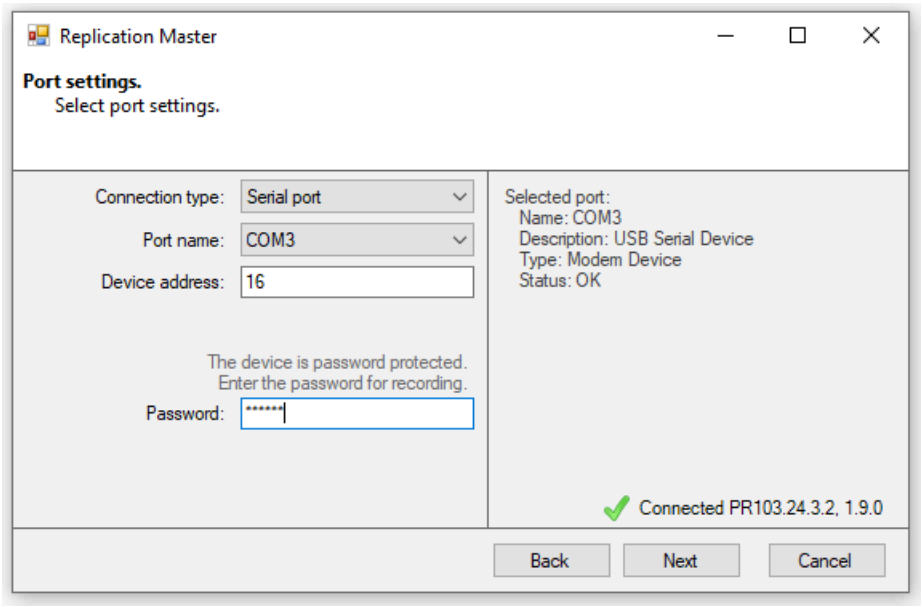
Fig. 9.11 Device connection error

In case of connection error, the error sign and text will be displayed. When you hover over the sign a tooltip will appear describing the error.

Error	Possible reasons
The device is not connected	– COM port selected incorrectly; – USB driver not installed; – connection break
Another device is required	The connected device does not match the project. You need to replace the device in the project and create a new replication master, or connect a suitable device

If there are no errors, there is a green check mark near the device name.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



If the connection with the device is constantly interrupted (the check mark is blinking), then another application may be occupying the USB port. This may happen if an ALP program is running and a connection to the same port is configured in it. In this case, to load the project into the device, the other application should be closed or switched to *OFFLINE*.

5. If the connection is stable, you should click the **Next** button.

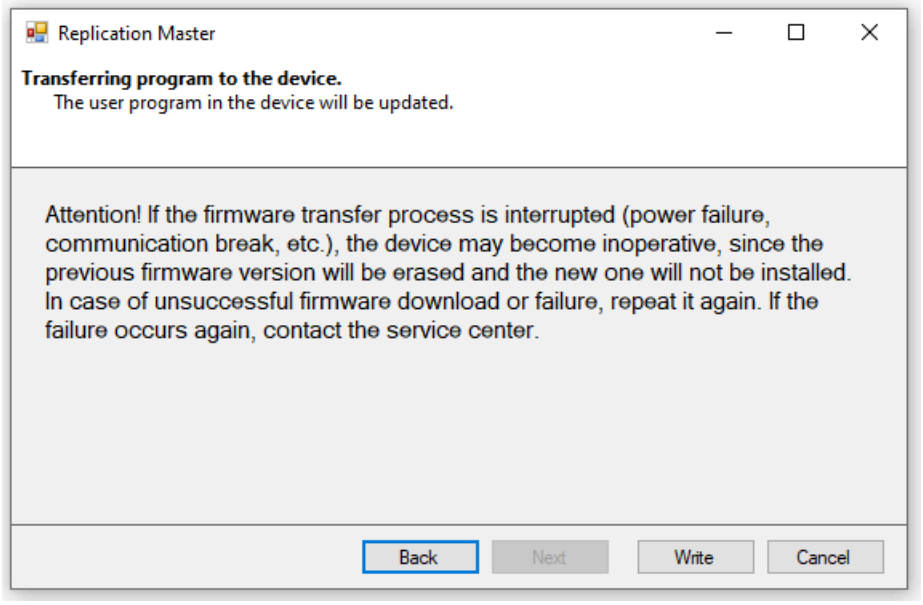


Fig. 9.12 Writing the program

6. To write the program into the device, click the **Write** button. The writing status line will be displayed at the bottom of the window. During the process, the connection between the PC and the device must not be interrupted. When the transfer is complete, the message will be displayed.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

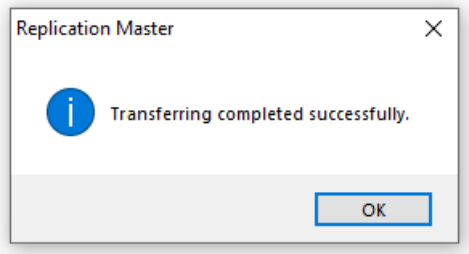


Fig. 9.13 Download complete

“Silent” installation mode

NOTE
The "silent" installation mode is available when working with the Replication master version 1.21.14 and later.

The Replication master supports the "silent" firmware mode for devices. To launch it, use the following command line arguments:

– ***/silent*** – "silent" installation argument;
– ***/password:*** – password, required if the device was password protected;
– **/ip:** – IP address of the connected device (if the device is connected via the Ethernet interface);
– **/portname:** – COM port number (if the device is connected via USB interface);
– **1>> File_Name.txt** – text file displaying the firmware process in percentage;
– **2>> File_Name.txt** – text file displaying errors during the firmware process.

An example of a command to run Replication master file (*ReplicationUtilite.exe*) located on the desktop when connected via an Ethernet interface:
C:\Users\User_name\Desktop\ReplicationUtilite.exe /silent /IP:10.2.3.204
Example of a command to run Replication master file (*ReplicationUtilite.exe*) located on the desktop when connected via USB to a device with password *1234*:
C:\Users\User_name\Desktop\ReplicationUtilite.exe /silent /portname:COM7 /password:1234
Example of a command to run the ReplicationUtilite.exe Replication master file located on the desktop when connected via USB to create a text file displaying errors during the firmware process:
C:\Users\User_name\Desktop\ReplicationUtilite.exe /silent /portname:COM7 2>> File_Name.txt

9.1.2 Writing a program to a device via the Replication master on Linux

PC requirements:

– Ubuntu 22.04 or 24.04, x86/32 or x64 processor architecture
– .NET 6

To write a program to a device on Linux, open a command line and enter the command:

– **./ReplicationUtilite -ip:192.168.1.100** – when connecting the device via Ethernet, where 192.168.1.100 is the IP address of the device.
– **./ReplicationUtilite -portname:/dev/ttyACM0** – when connected via a serial port, where ACM0 is the port address.

NOTE
To determine the device port, use the commands:
– **lsusb** – for USB;
– **dmesg | grep tty** – for serial port.

An example of a command to launch the replication master file when connecting the device via a serial port with the address ACM0:

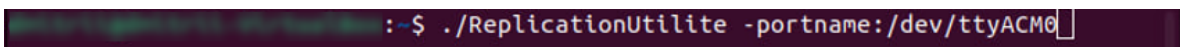


Fig. 9.14 Command to write a program

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

If the actions are performed correctly, the command line will display the status line of the program loading into the device:

```
[########################################-----------]  77% /
```

Fig. 9.15 Status bar

Table 9.1 Possible errors

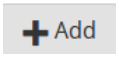
Error	Solution
You do not have permission to run the file `Permission denied`	– contact the administrator – use the command **sudo ./ReplicationUtilite -portname:/dev/ttyACM0**, where ACM0 is the port address
Entering an invalid command `Incorrect string arguments are specified`	Please enter a valid command to run the file
The device is password protected `Incorrect string arguments are specified`	Enter the command **-password:**.

9.2 Exporting the device to akYtec Cloud

Exporting a device to akYtec Cloud reduces the time to add a device configured in Slave mode to the akYtec Cloud service.
Working with akYtec Cloud requires a special device, a gateway, to be connected to the device. Let's assume that the gateway is already configured and connected to the control panel, the settings are given in the *Operation Manual* for the gateway.
To export the configuration of Modbus parameters of the device, proceed as follows:

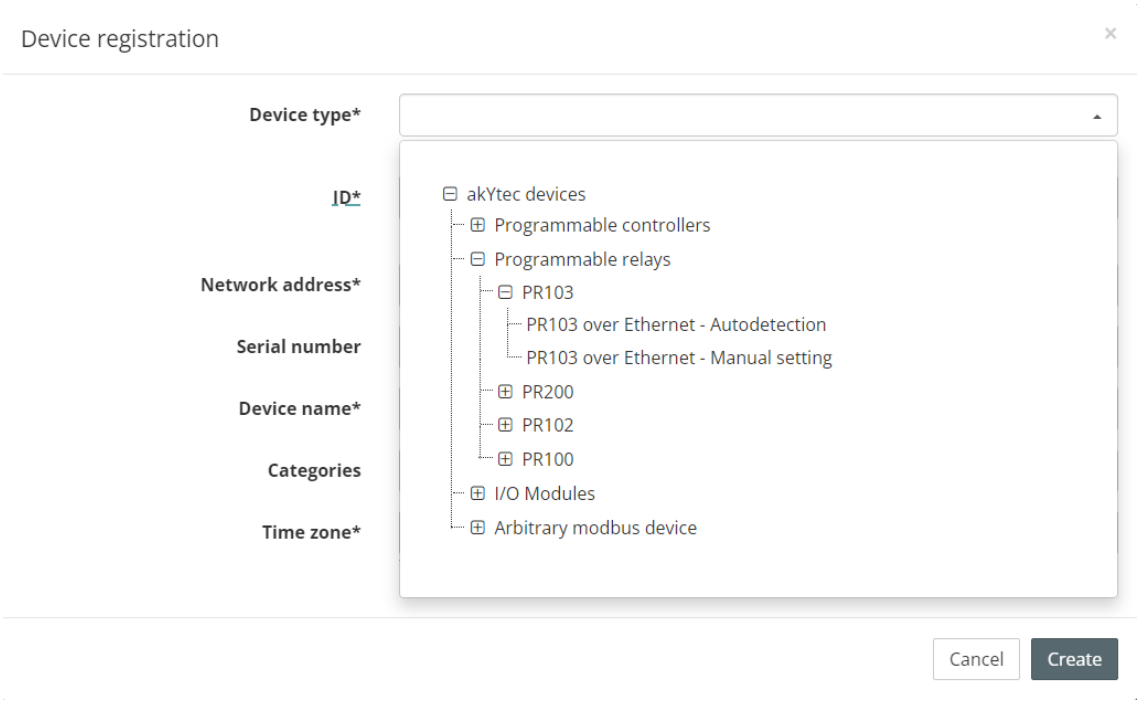
1. Create a project with settings in Slave mode and network variables.
2. Select **Export Device to akYtec Cloud** from the **Plugins** main menu.
3. In the menu that opens, select the location and specify the file name in **.json* format.

Further steps require a personal account in akYtec Cloud.
After logging in to your personal account:

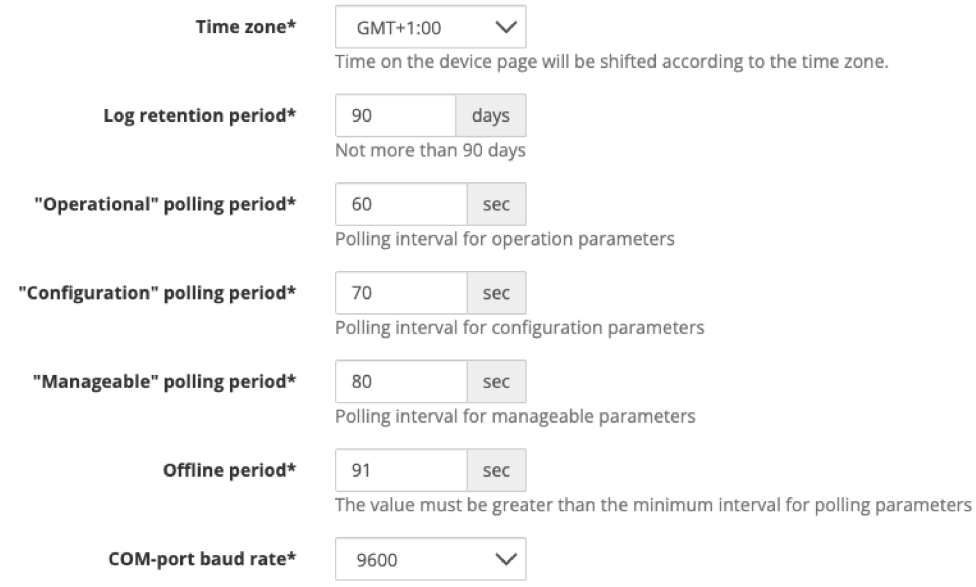
1. Click the button .
2. On the page that opens, click the 

button.
3. In the menu that appears, set the **ID**: IMEI / serial number of the network gateway (indicated on the gateway housing). In the **Device Type** drop-down list, select **PR103** in list **Programmable relays**. Enter the address of the device that was set in ALP. Fill in the remaining fields. Next, click the **Create** button.

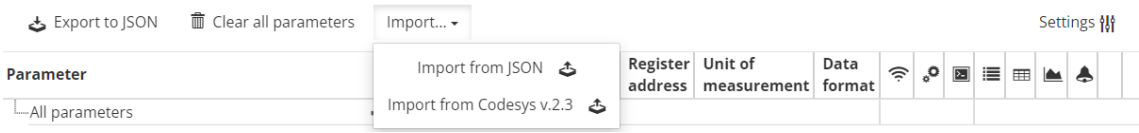
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



4. On the **General/General Settings** tab, specify the polling rate and COM port settings of the device according to the settings in ALP. Click the **Save** button to apply the new settings.



5. On the page that appears, click the **Parameter Settings** button. Click on the **Import** drop-down list and select the **Load from JSON** option. In the menu that opens, select the previously created file in **.json* format and click **Load Parameters**.



6. Modbus variables of the device will be added to akYtec Cloud.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

10 Keyboard shortcuts

Keyboard shortcut	Action
Menu/File	
Ctrl + N	Create a new project
Ctrl + O	Open an existing project
Ctrl + Alt + S	Save an open project under a different name
Ctrl + S	Save an open project
Ctrl + P	Print
Ctrl + Shift + C	Open Component Manager
Menu/View	
Ctrl + Z	Undo last change
Ctrl + Y	Return (restore) a canceled action
Menu/Device	
Ctrl + F7	Transfer the application to the device
Ctrl + Shift + V	Open Variable Table
Ctrl + Shift + S	Open device configuration
Menu/Service	
Shift + F5	Go to simulation mode
F6	Start simulation
F7	Stop simulation
F8	Pause simulation
F10	Single cycle
Ctrl + F5	Go to debug mode
Menu/Plugins	
Ctrl + Shift + P	Open plugins manager
Menu/Help	
F1	Open Help
Insert panel	
Ctrl + Shift + F	Create an ST function
Ctrl + Shift + M	Create a macro
Ctrl + Shift + B	Create an ST function block
Ctrl + M	Create a macro from a selection
Usage location panel	
F5	Refresh usage locations
Keys for working with elements	
Ctrl + C	Copy an element
Ctrl + V	Paste from clipboard
Delete	Deleting a selected item

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Keyboard shortcut	Action
Element resizing keys	
Ctrl + →	Increasing the width of a selected element
Ctrl + ←	Decreasing the width of a selected element
Ctrl + ↓	Increasing the height of a selected element
Ctrl + ↑	Decreasing the height of a selected element
Scaling the workspace	
Ctrl + Mouse wheel	When you rotate the mouse wheel away from you, the scale of the workspace increases. When you rotate the mouse wheel toward you, the scale of the workspace decreases
Ctrl + «+»	Increase scale
Ctrl + «–»	Decrease scale
Switch between tabs	
Tab + →	Switch between tabs
Tab + ←	

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

11 Program examples

These examples with simple tasks explain the creation of a circuit program in the ALP programming software.

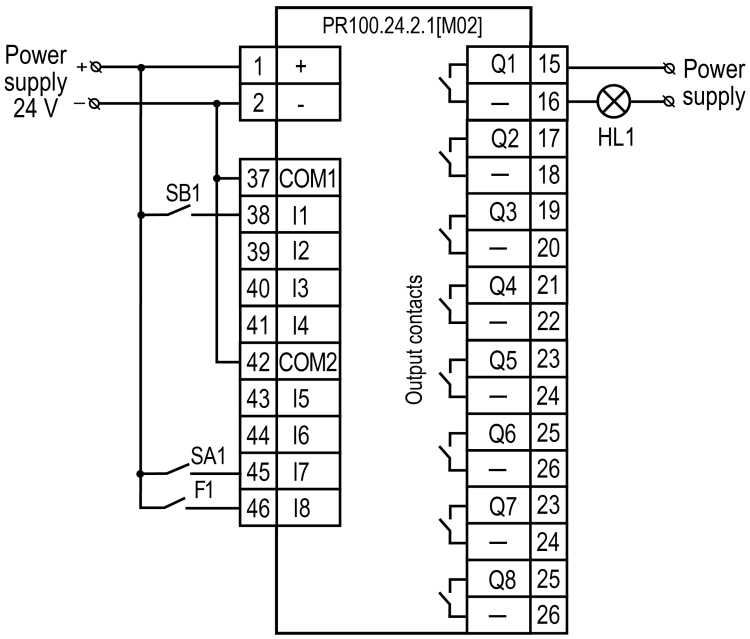
– *Light switch with automatic switch-off*
– *Mixer control*
– *Direct connection of PR103 to akYtec Cloud*

11.1 Task 1: Light switch with automatic switch-off

The task is to switch the light on for a certain time, e.g. for a house entry.
Task definition:

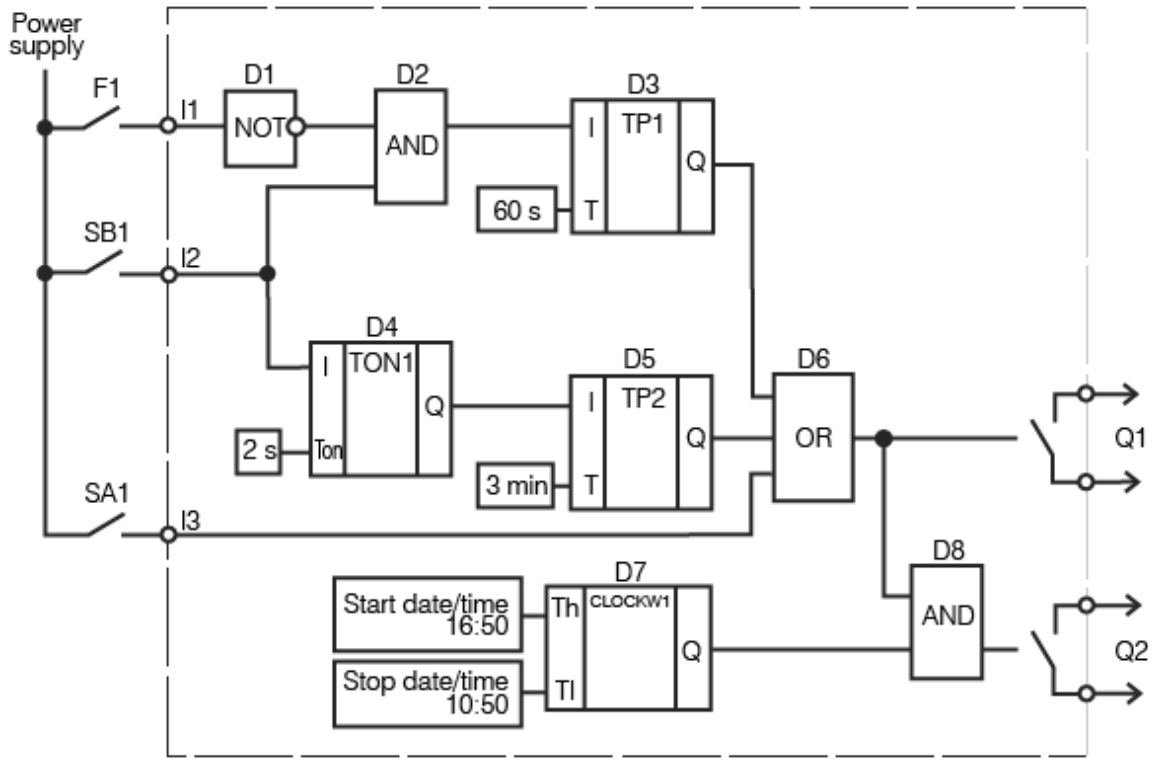
1. The light sensor F1 and the light button SB1 “TIME” are installed in front of the entrance door.
2. If the button SB1 is shortly pressed and the ambient light is insufficient, the light switches on for 1 minute – this time should be enough to find a key hole and to open the door.
3. If the button SB1 is pressed for 2 seconds, the light switches on for 3 minutes regardless of the ambient light – this mode can be useful for entrance cleaning.
4. Provide the possibility to control the light by commands from external devices or with the switch SA1 “CONST” regardless of the ambient light. This mode can be useful during the reception of guests or for further automation of the apartment as part of the "smart house" program.
5. Provide the possibility to switch on the light only at a certain time.

Device selection:
The control device must have minimum two digital inputs, one digital output and an integrated real-time clock to implement this task. These features can be provided by devices of PR100 series.
The task implementation with PR100.24.2.1(M02):



Circuit program
The circuit program can be implemented in the way shown in figure below.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

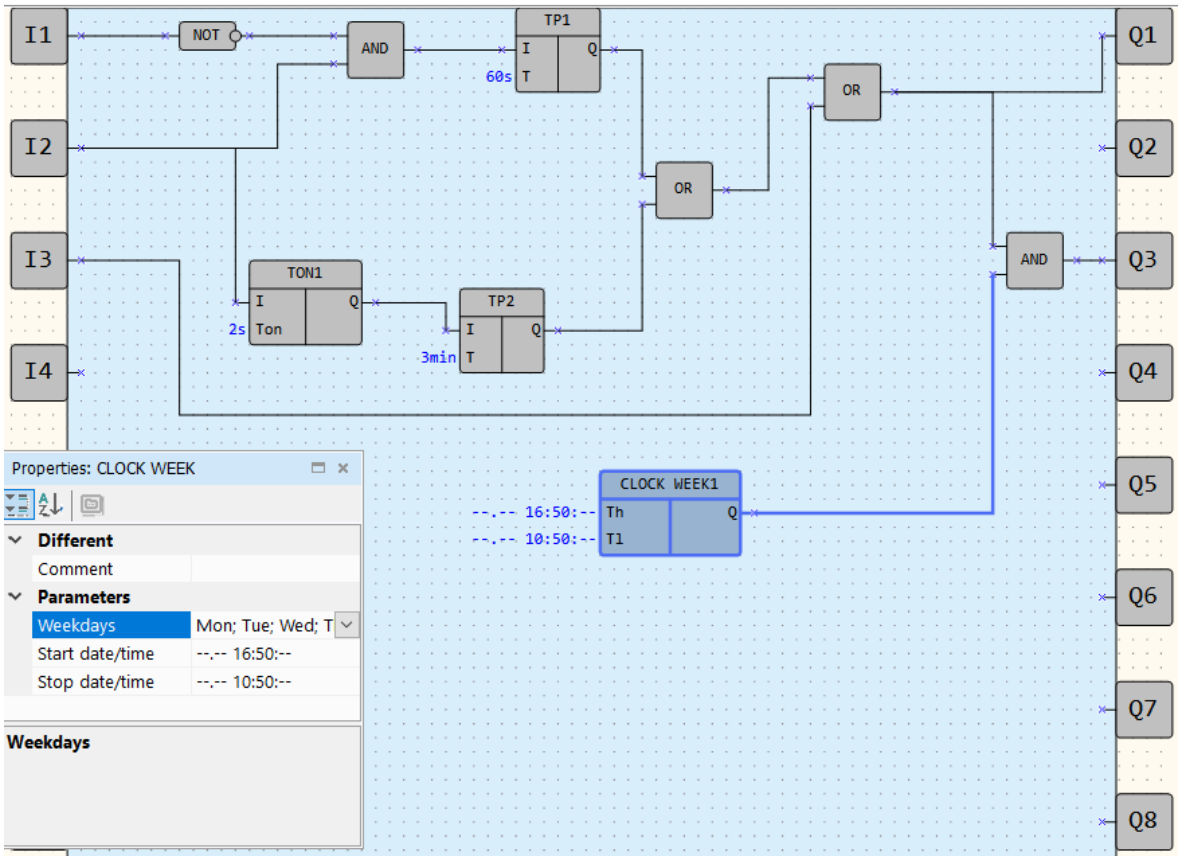


Input I1 – connected to light sensor F1
Input I2 – connected to SB1 button
Input I3 – connected to SA1 switch
Output Q1 – output to implement the task points 1-4
Output Q2 – output to implement the task point 5
Program description:

1. If the button SB1 is shortly pressed (< 2 s), the logical AND (D2) is enabled. If the ambient light is insufficient, the first input of D2 is also **True** and the timer TP “Pulse” (D3) forms a pulse with 1 minute duration. This pulse activates the output Q1 over the logical OR (D6) and the light switches on for 1 minute.
2. If the button SB1 is pressed for > 2 s, the on-delay timer TON (D4) activates the timer TP “Pulse” (D5), a pulse with the duration of 3 minutes activates the output Q1 over logical OR (D6) and the light switches on for 3 minutes.
3. If the ambient light is sufficient, the contact of the sensor F1 closes, the logical AND (D2) is disabled and the timer TP “Pulse” (D3) is blocked.
4. If the switch SA1 “CONST” is closed, the output Q1 activates over the logical OR (D6) and the light is switched on constantly.
5. If you want to use the light only on certain weekdays at certain times, you can use the output Q2. With the weekly timer CLOCKW (D7) you can set the start and the stop time and the weekdays for lighting.

The circuit program created in ALP is shown in figure below.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



11.2 Task 2: Mixer control

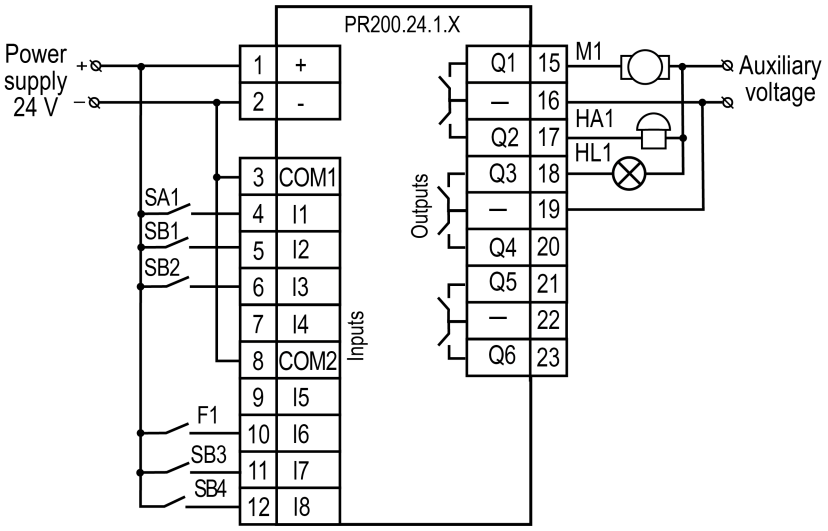
The task is to implement an industrial mixer with simple control functions.
Task definition:

1. Automatic and Manual operation modes are required. The switch SA1 “MODE” is installed to switch between the modes.
2. In Automatic mode the operating cycle can be started with the button SB1 “START” and stopped automatically with the end of the cycle or manually with the button SB2 “STOP”. The cycle duration is 5 minutes. During the cycle the motor of the mixer is on for 15 seconds and off for 10 seconds alternately. All settings can be changed in the program.
3. In Manual mode the motor can be started with the button SB1 “START” and stopped with the button SB2 “STOP”.
4. When the motor is overloaded (overload switch F1), it switches off automatically. Signal lamp HL1 "Overload" indicates an operating error. Intermittent warning signal (HA1) sounds with 3-second interval.
5. The acoustic signal can be switched off with the button SB3 “RESET”.
6. The button SB4 “CONTROL” is used for the functional test of the lamp HL1 and the acoustic signal HA1.

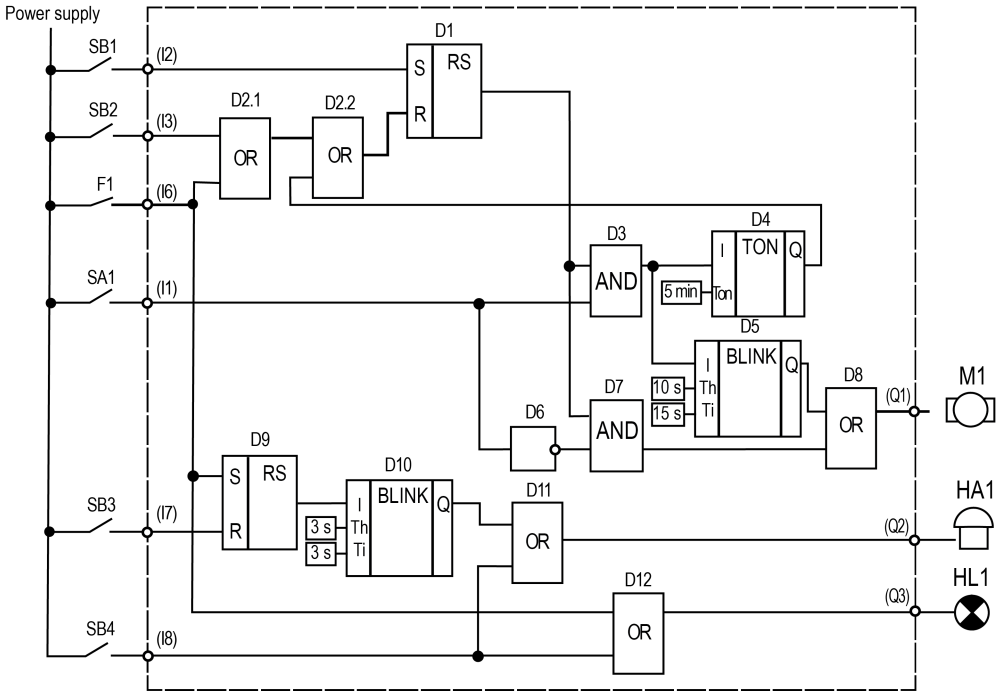
Device selection:
The control device must have minimum 6 digital inputs and 3 digital outputs to implement this task. These features can be provided by devices of PR200 series.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The task implementation with PR200.230.1.X:



Circuit program
The circuit program can be implemented in the way shown in figure below.



Input I1 – connected to SA1 “MODE” switch
Input I2 – connected to SB1 “START” button
Input I3 – connected to SB2 “STOP” button
Input I6 – connected to F1 overload switch
Input I7 – connected to SB3 “RESET” button
Input I8 – connected to SB4 “TEST” button
Output Q1 – connected to motor
Output Q2 – connected to HA1 acoustic signal
Output Q3 – connected to HL1 signal lamp

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Program description:

1. Input I2 (SB1 “START”)

   If the button SB1 is pressed, the RS trigger D1 becomes **True** as long as there is no reset signal at the input R. Subsequent signal path depends on the state of the switch SA1 “MODE”:

   – If SA1 is open (Manual mode), the logical AND (D7) and the logical OR (D8) are enabled and the motor M1 (output Q1) is switched on.
   – If SA1 is closed (Automatic mode), the logical AND (D7) is disabled and the start signal can only activate the pulse generator BLINK (D5) to start the operating cycle (15 s on / 10 s off) and the on-delay timer TON (D4) to stop it (in 5 min).

2. Input I3 (SB2 “STOP”)

   If the button SB2 is pressed or the switch F1 is activated, the RS trigger D1 is reset over the input R and the output Q1 is disabled.

3. Input I1 (SA1 “MODE”)
   – If the switch SA1 is open (Manual mode), the logical AND D3 is disabled and D7 is enabled, the timer D4 and the pulse generator D5 are disabled and the motor M1 can be only started with SB1 and stopped with SB2.
   – If the switch SA1 is closed (Automatic mode), the logical AND D3 is enabled and D7 is disabled, thus the motor M1 can be only started by the pulse generator D5 (15 s on / 10 s off cycle) and stopped by the timer D4 in 5 minutes.

4. Input I6 (overload switch F1)

   When the motor is overloaded, the F1 contact is closed, the RS trigger D1 is reset and the motor is stopped.

   Concurrently the signal lamp HL1 is switched on over the logical OR (D12) and the acoustic signal HA1 is activated over the RS trigger D9. The pulse generator D10 provides an intermittent acoustic signal with the cycle 3 s on / 3 s off.

5. Input I7 (SB3 “RESET”)

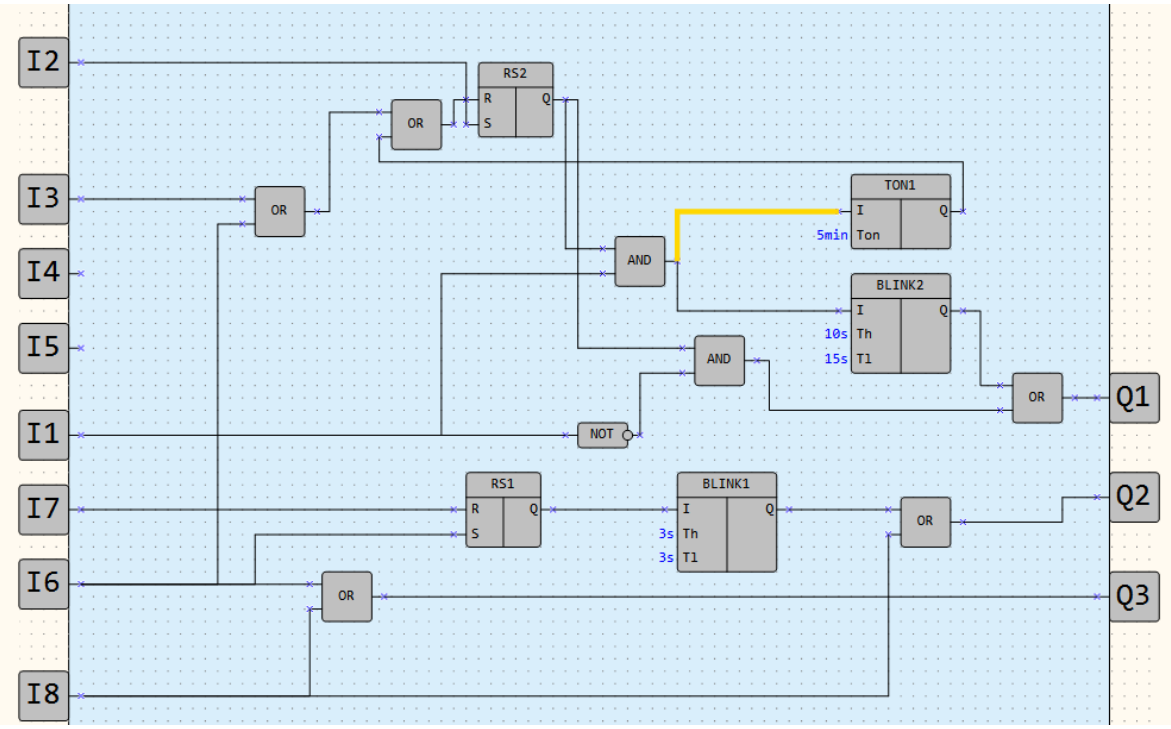
   The button RESET is used to reset the acoustic signal HA1. If the button SB3 is pressed, the RS trigger D9 is reset and the pulse generator D10 for the acoustic signal HA1 is stopped.

6. Input I8 (SB4 “TEST”)

   The button TEST is used to test the acoustic signal HA1 and the signal lamp HL1. If the button SB4 is pressed, the logical ORs D11 and D12 are enabled, the outputs Q2 and Q3 activated, the acoustic signal and the lamp switch on.

The circuit program is shown in the figure below.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



NOTE

1. *The remaining two unused inputs and one output can be used for implementation of additional functions. For example, to switch between different time settings for automatic motor operation or to switch other operating parameters of the mixer.*
2. *The technological cycle of operation can be completely automated by implementation of an incremental counter (CT) to switch off the RS trigger D1.*

11.3 Task3. Direct connection of PR103 to akYtec Cloud

Direct connection to akYtec Cloud is available for devices on the new platform

NOTE
For a list of devices on the new platform, see the *About* the program section..

The device must be configured for operation and connected to the network.
To connect to akYtec Cloud you should:

1. Connect the device to the PC and create a project using network variables.
2. Set a *password* to access the device.
3. In the *settings* device window, allow access to akYtec Cloud.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

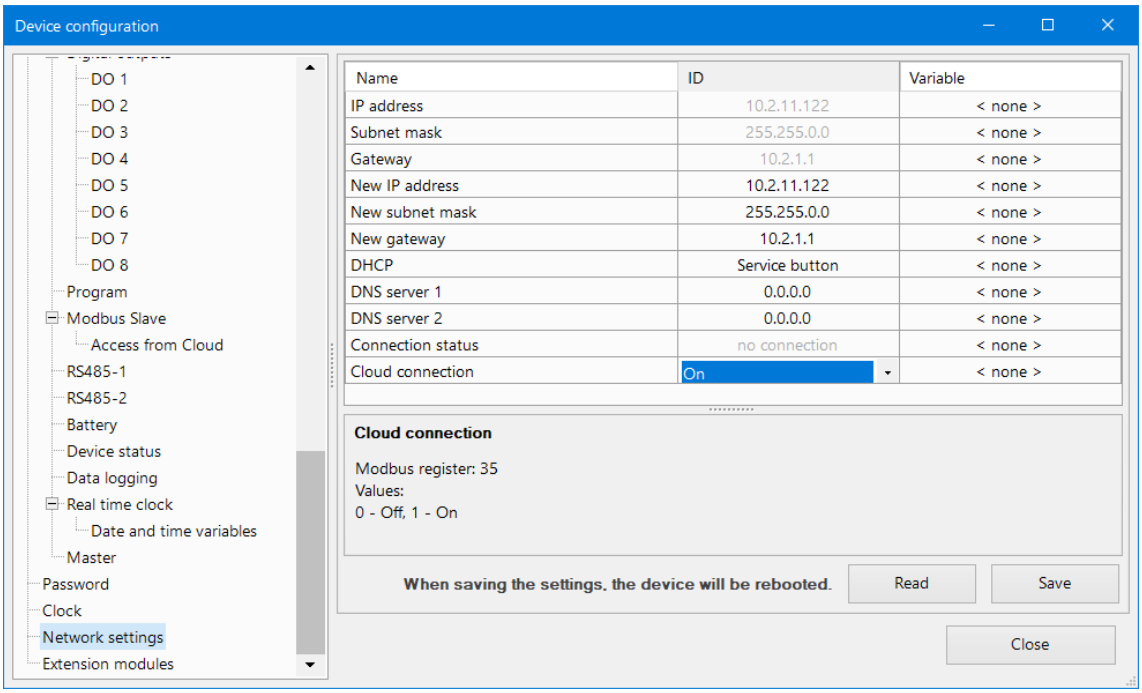


Fig. 11.1 Device configuration window

NOTE
The value of the **Cloud connection** status parameter can be read via Modbus or a user program variable can be linked. The parameter description is given in the table below..

Table 11.1 Possible states of the Cloud connection parameter

State	Value	Description
no connection	0	Cloud sharing is disabled
identification	1	Establishing connection to Cloud
normal operation	2	Cloud controls the device, no errors
network error	3	The device cannot establish a connection to the Cloud server
no password	4	The password for the device is not set

4. Allow remote access to Modbus registers.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

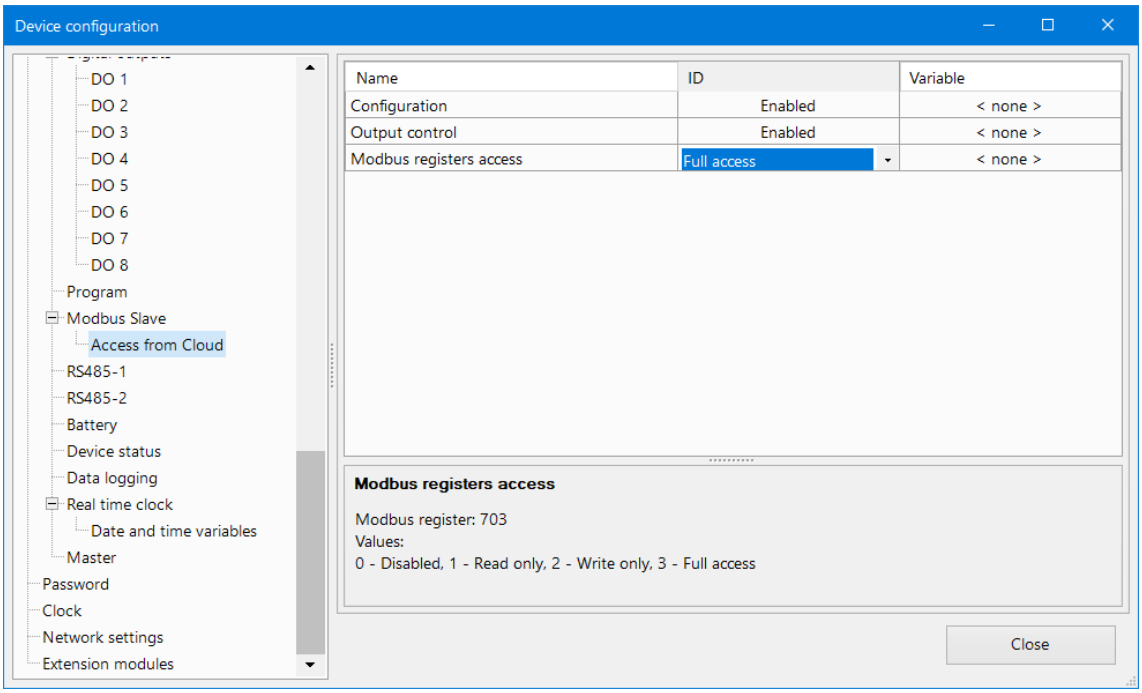
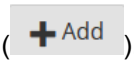


Fig. 11.2 Device configuration window

5. Load the program into the device **(Device → Load the program into the device**).
6. Go to the main page of the *Cloud* service. If you don't have an akYtec Cloud account, then go through the registration procedure.
7. Go to the **Administration** tab, open the **Devices** tab and click the **Add device** button (

).
8. A window will appear with a choice of the device type.

NOTE
Direct connection to Cloud is performed with PR103 as an example. To connect other second generation devices, follow similar steps.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

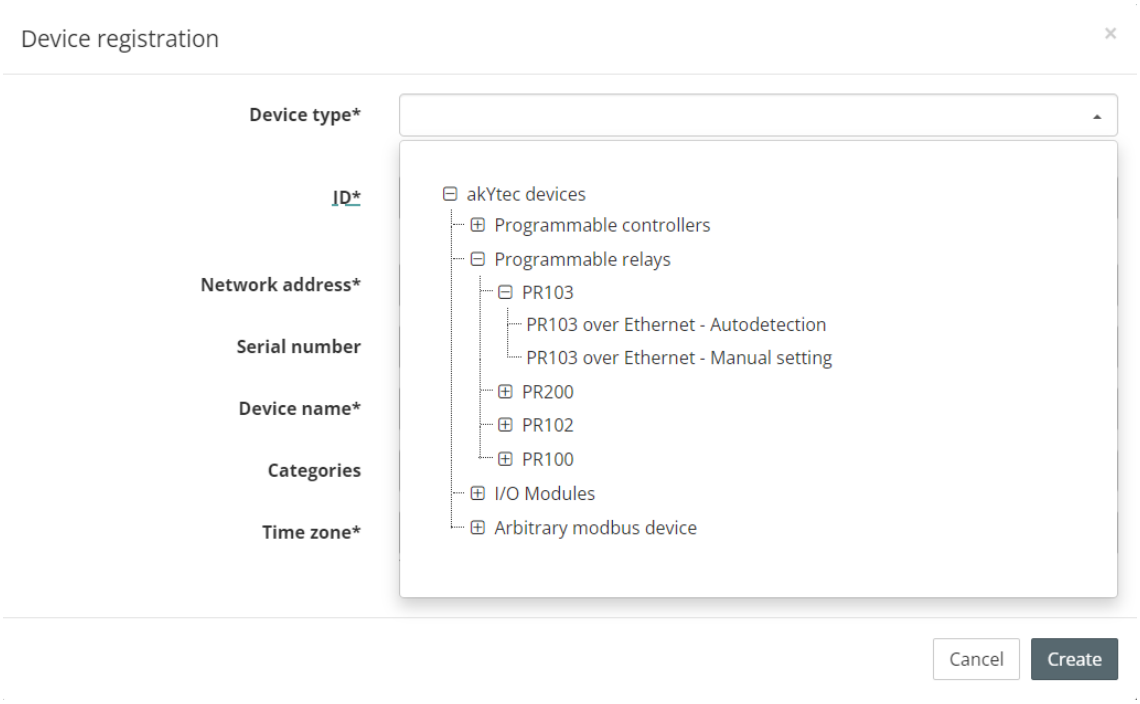


Fig. 11.3 Selecting the device type

Option **PR103 via Ethernet – Autodetection**:

NOTE
With this method of adding a device, not only network variables are displayed, but also the entire tree of device parameters.

a. Enter in the new window:
   – Identifier – device serial number
   – Network address – address 1, leave unchanged
   – Device name – name of the device;
   – Time zone – time zone in which the device is located
b. Click **Add**. The basic settings interface of the device will open.
c. Enter the password for the device. If necessary, you can change other settings (for example, the polling period).
d. Click **Save** to apply the new settings.
e. Cloud will connect to the device and read all parameters from it.

Option **PR103 via Ethernet – Manual setup**:

a. Enter in the new window:
   – Identifier – device serial number
   – Network address – address 1, leaves unchanged
   – Device name – name of the device;
   – Time zone – time zone in which the device is located.
b. Click **Add**. The general settings interface of the device will open.
c. Enter the password for the device. If necessary, you can change other settings. Click **Create** to apply the new settings.
d. Next, in the Parameters/Parameter settings tab, add the network parameters of the device using one of the available methods:
   – manually, in accordance with the network variables specified in PR103. It should be noted that the register addresses are specified in hexadecimal - therefore the values used differ from those given in ALP - 16384 (DEC) = 4000 (HEX).

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– as a *.json file, if you use the Device Export extension in akYtec Cloud ALP. To add parameters, click on the Import drop-down list and select the Load from JSON option. In the menu that opens, select the previously created file in *.json format and click the Load parameters button.

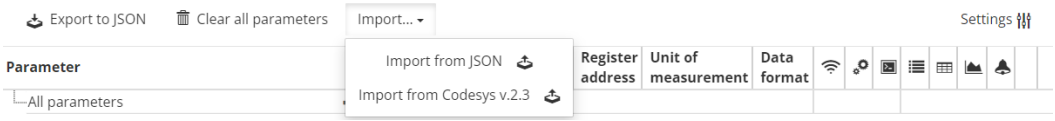


Fig. 11.4 Import parameters

If all settings were correct, then the **General Data** tab will show data from the device. If the device is not polled from Cloud, then you should check the network settings, the connection status to Cloud.



Fig. 11.5 Example display of connection to Cloud

If the values are not written to the device from Cloud, you should check if the checkbox in the **Managed Parameter** column is checked:



Fig. 11.6 Controlled parameter

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12 ST language

ST (Structured Text) is a high-level text language. It is one of the five languages supported by the IEC 61131-3 standard.
ALP allows you to create functions and function blocks in the ST language.

– *Syntax*
– *Data types*
– *Language structure*

12.1 Syntax

Keywords can be entered in upper and lower case characters. Spaces and tabs do not affect the syntax and can be used everywhere.
The names of variables, functions and function blocks should follow the rules:

– The name must not contain spaces or special characters (for example, !, @, etc.). The exception is the underscore character (_)
– The name must start with a letter
– The variable name can only contain letters of the Latin alphabet
– The name must not contain multiple underscore (_) characters in a row (i.e. the name **i__Test** is not valid, but the name **i_Te_st** is valid)
– Object names are case insensitive (**ITest** and **iTEST** will be interpreted as the same name)
– There are no restrictions on name length
– The name must not match one of the reserved keywords (eg VAR, INT, etc.)
– It is recommended to use Hungarian notation and lowerCamelCase style for variable names

12.1.1 Bitwise access to variables

Bitwise access is available for variables of type UDINT. To access a bit, use the following format:

```
Variable.Bit_number
```

Example

```
Variable.1 := TRUE;
Variable.31 := FALSE;
```

The bit number must be specified as an integer from 0 to 31. Using variables as bit numbers is not permitted.

12.1.2 Using functions in other ST program elements

A function can be called within another function or function block. To do this, you need to use the following format (informal call):
Function name (comma separated list of function inputs)
Example

```
FUNCTION rFun1: REAL;

    VAR_INPUT
        rIn1 : REAL;
        rIn2 : REAL;
    END_VAR

    rFun1 := rIn1 + rIn2;

END_FUNCTION
```

There is a function **rFun1**:
Calling **rFun1** from **rFun2** function will look like this:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
FUNCTION rFun2: REAL;

    VAR_INPUT
        rIn1_0 : REAL;
        rInf1_1 : REAL;
        rInf1_2 : REAL;
    END_VAR

    rFun2 := rIn1_0 * rFun1(rInf1_1, rInf1_2);

END_FUNCTION
```

rFun2 function will return a number equal to the product of **rIn1_0** and the sum of **rInf1_1** and **rInf1_2**, which are specified at the input of this function.

12.1.3 Using one function block in another

Instances of one function block can be created in another function block, namely in the local variable declaration area in the format:

(short designation):(function block name)

Once a function block is instantiated, you can begin working with its data. The inputs of a block instance are externally writable. Outputs are for reading.

You can call an instance in different ways. As an example, consider the call to a counter for forward counting, which is created as a template when creating a function block:

1. Through a formal call:

```
FUNCTION_BLOCK fb2

    VAR_INPUT
        xIn : BOOL;
    END_VAR

    VAR_OUTPUT
        xAlarmMax : BOOL;
        udiQ : UDINT;
    END_VAR

    VAR
        fb1 : functionblock1; //declaration of a function block instance
    END_VAR

    fb1 (U := xIn, Q => udiQ); //calling a function block instance
    IF fb1.Q > 10 THEN
        xAlarmMax := TRUE;
    END_IF

END_FUNCTION_BLOCK
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
Pay attention to the specific operator for copying a value from the block's output variables (“=>”).

2. By accessing the inputs/outputs of a function block (variable declarations are similar):

```
fb1.U := xIn; //write data to FB input
fb1 (); //call a FB instance
udiQ := fb1.Q; //read the output of the FB instance
IF fb1.Q > 10 THEN
    xAlarmMax := TRUE;
END_IF
```

The **fb2** output will indicate that 10 pulses have been exceeded and the accumulated counter value.

NOTICE
You cannot declare an instance of a function block in the body of a function.

NOTICE
In an ST functional block, the maximum nesting of blocks is no more than 8.

NOTICE
ST function blocks do not support RETAIN type variables.

NOTICE
ST function blocks reserve space in ROM memory after they are added to the project library, regardless of whether they are used in the project or not.

12.1.4 Comments in ST editor

The ST editor in ALP supports the commenting feature. Two types of comments are available:

1. Single-line. Its marker is a double slash – //
2. Multiline. Its marker is:
   – (* – beginning of comment
   – *) – end of comment

12.1.5 Copying ST elements between projects

To copy ST components between projects:

1. Select the ST elements that need to be copied on the original project diagram.
2. Copy the selected elements using the keyboard shortcut **Ctrl+ C** or the context menu.
3. Open the project to paste the copied elements.
4. Paste elements into the second project diagram using the keyboard shortcut **Ctrl + V** or the context menu

As a result, templates of all copied elements will be added to the corresponding section of the project component library.

NOTE

1. If the component library does not contain the group that was specified for the copied component, it will be created.
2. If the copied component group is not specified, the template will be added to the **Other** folder.

When copying, all connections between all components that were included in the copy operation are preserved.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTICE
If any of the copied elements contains an error, the pasting of the elements into the project diagram will be canceled.
An error message will be displayed indicating the name of the incorrect element.
Other components will appear in the component library in the appropriate sections.

12.2 Documentation in the ST editor

ST editor supports documentation function.
The documentation marker is the triple slash "///".
Documentation is added above objects: function/function block declaration, input/output variable declaration.
Tags for documentation are listed in the table below.

Tag	Description
<Description>… </Description>	Description of the program element (function, function block, input and output (function block only) variable)
<Author>…</Author>	Name of the creator of the function or function block
<GroupName>… </GroupName>	Group name for grouping a function or function block in a content library
<OutputDescription>… </OutputDescriotion>	Function output description

Documenting the function

```
///<Description>Resistance temperature detector (Pt1000)</Description>
///<OuptutDescription>Temperature</OutputDescription>
///<Author>akYtec</Author>
///<GroupName>Temperature sensor</GroupName>

FUNCTION f_PT1000: REAL; // function for PT1000 RTD sensor

    VAR_INPUT

        ///<Description>Resistance</Description>
        R : REAL;

    END_VAR

    VAR_OUTPUT

        ///<OutputDescription>Temperature</OutputDescription>
        Q : REAL;

    END_VAR
```

Documenting the function block

```
///<Description>Counter for direct counting</Description>
///<Author>akYtec</Author>
///<GroupName>Timers and counters</GroupName>
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
FUNCTION_BLOCK fb_Counter

    VAR_INPUT

        ///<Description>Pulse detector</Description>
        U : BOOL; //bool input variable

        ///<Description>Counter reset flag</Description>
        Res : BOOL; //bool input variable

        ///<Description>Preset counter value</Description>
        N : UDINT; //bool input value

    END_VAR

    VAR_OUTPUT

        ///<Description>Counter value</Description>
        Q : UDINT; //udint output value

    END_VAR
```

12.3 Data types

Data types, supported in akYtec ALP:

Data type	Description	Valid range	Size
BOOL	Boolean	FALSE, TRUE	4 bytes
UDINT	Unsigned double integer	0…4294967295	4 bytes
REAL	Floating-point	-1.2×10^{-38}…3.4×10^{38}	4 bytes
TIME	Time interval	T#0…4294967295ms T#0..4294967s T#0..71582m T#0..1193h T#0..49d T#0..49d17h02m47s295ms	4 bytes
DT	Time of day and date	DT#2000-01-01-00:00:00..2136-02-07-6:28:15	4 bytes
ARRAY	Array	—	—

The data type of a variable determines the type of information, the range of representations, and the set of allowed operations.
The variable can be used only after its declaration. It is possible to assign the value of one variable to another variable only if they are of the same type. Otherwise, type converter should be used.

NOTE
Converting a larger type to a smaller one can result in loss of information.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.3.1 Reserved keywords

NOTE
In the table, words available for use are highlighted in **bold**.

ABS	END_TRANSITION	**RETURN**	**SYS.TOF**
ACTION	END_TYPE	SEL	**SYS.TON**
AND	**END_VAR**	**SHL**	**SYS.TP**
ARRAY	**END_WHILE**	**SHR**	TASK
AT	ENO	SINT	**THEN**
BEGIN	**EXIT**	STEP	**TIME**
BOOL	F_EDGE	STRING	TIME_OF_DAY
BOOL_TO_REAL	**FALSE**	STRUCT	**TIME_TO_UDINT**
BOOL_TO_UDINT	**FOR**	**SYS.ACOS**	**TO**
BY	**FROM**	**SYS.ASIN**	TOD
CASE	**FUNCTION**	**SYS.ATAN**	TRANSITION
CD32	**FUNCTION_BLOCK**	**SYS.BLINK**	**TRUE**
CONFIGURATION	**GET_DATE_TIME**	**SYS.CLOCK**	TYPE
CONTINUE	**GET_TIME**	**SYS.CLOCKWEEK**	**UDINT**
DATE	**IF**	**SYS.COMPARE_ DATE_TIME**	**UDINT_TO_BOOL**
DATE_AND_TIME	INITIAL_STEP	**SYS.COS**	**UDINT_TO_DT**
DC32	INT	**SYS.CT**	**UDINT_TO_REAL**
DINT	LINT	**SYS.CTN**	**UDINT_TO_TIME**
DO	LREAL	**SYS.CTU**	UINT
DT	LWORD	**SYS.DTRIG**	ULINT
DT_TO_UDINT	**MOD**	**SYS.EXP**	**UNTIL**
DWORD	NON_RETAIN	**SYS.FACT**	USINT
ELSE	**NOT**	**SYS.FMAX**	**VAR**
ELSIF	**OF**	**SYS.FMIN**	VAR_ACCESS
EN	ON	**SYS.FTRIG**	VAR_CONFIG
END	**OR**	**SYS.IS_LEAP_YEAR**	VAR_EXTERNAL
END_ACTION	**POW**	**SYS.LG**	VAR_GLOBAL
END_CASE	PROGRAM	**SYS.LN**	VAR_IN_OUT
END_ CONFIGURATION	R_EDGE	**SYS.LOG**	**VAR_INPUT**
END_FOR	READ_ONLY	**SYS.MAX**	**VAR_OUTPUT**
END_FUNCTION	READ_WRITE	**SYS.MIN**	VAR_TEMP
END_FUNCTION_ BLOCK	**REAL**	**SYS.RS**	**WHILE**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

END_IF	**REAL_TO_BOOL**	**SYS.RTRIG**	WITH
END_REPEAT	**REAL_TO_UDINT**	**SYS.SIN**	WORD
END_RESOURCE	**REPEAT**	**SYS.SQRT**	WSTRING
END_STEP	RESOURCE	**SYS.SR**	**XOR**
END_STRUCT	RETAIN	**SYS.TAN**	

12.3.2 Arrays

Array declaration syntax

The following syntax is used to declare an array:

```
<Array name>:[Index1..IndexN]OF <ELEMENT TYPE>:=[Default value of the element1,
Default value of the element2, Default value of the elementN];
```

Where:
Index1 – number of the first element of the array;
IndexN – number of the last element of the array.
Rules for indices:

– The value of IndexN cannot be less than or equal to the value of Index1.
– Indices must be of integer type only and cannot take negative values.

Examples of array declaration:

```
        Mass1: ARRAY[0..1] OF UDINT; //declaration of an array consisting of
two elements
UDINT type
        Mass1: ARRAY[0..1] OF UDINT :=[1, 5]; //initialization of array elements
when declaring an array
        Mass1: ARRAY[0] OF UDINT :=[1, 5]; /error: array dimension must be
specified as <udint>...<udint>
        Mass1: ARRAY[] OF UDINT; //error: array dimension must be specified in
 <udint>...<udint>
```

Errors when declaring an array:

```
        Mass1: ARRAY[0] OF UDINT :=[1, 5]; //error: array dimension
must be specified as <integer>...<integer>
        Mass1: ARRAY[] OF UDINT; //error: array dimension
must be specified as <integer>...<integer>
```

Syntax for accessing array elements

To access array elements, use the following syntax:

```
<Array name>[Index1];
```

Examples of working with array elements:

```
VAR:=Mass1[5]; //assigning the value from the fifth element
of the array to the variable Var
VAR:=Mass1[5]+5*2;
VAR:=UDINT_TO_BOOL(Mass1[5]); //assigning the converted
value to the variable Var
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
frome the five element
VAR:=(Mass1[i]); //assigning the value from the i element
of the array to the variable Var,
where i is a variable (local/input)
```

The following rules apply when working with arrays:

1. An array can only be one-dimensional.
2. The number of array elements is limited to 32768 elements.
3. Array elements can be of the following types: BOOL, UDINT, FLOAT, TIME, and DT.
4. Array support for functions:
   – A function can return a variable of the Array data type.

     Example of a function that returns an array:

```
//function MyFunction will return a variable of Array type
function MyFunction: ARRAY [0..1] OF  UDINT ;
   var_input
      //input variables of array type
       Mass1 : ARRAY [0..1] OF  UDINT ;
   end_var
      //forming the return array in the function body
MyFunction [0]:= Mass1 [0];
MyFunction [1]:= Mass1 [1];
end_function
```

   – Declaring an array-type variable within a function occurs in the VAR...END_VAR and/or VAR_INPUT...END_VAR sections.

     Example of declaring array-type variables in different sections of a function:

```
//function MyFunction will return a variable of array type
function MyFunction: ARRAY [0..1] OF  UDINT ;
   var_input
      //input variables of array type
       Mass1 : ARRAY [0..1] OF  UDINT ;
   end_var
   var
      //local variables of array type
       Mass2 : ARRAY [0..1] OF  UDINT ;
   end_var
end_function
```

   For function blocks:

   – Declaring an array-type variable in an FB occurs in the VAR...END_VAR, VAR_INPUT...END_VAR, and/or VAR_OUTPUT...END_VAR sections.

```
function_block MyFunctionblock
   var_input
      //declaration of an input variable of array type
       Mass1 : ARRAY [0..1] OF  UDINT ;
   end_var
   var_output
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
        //declaration of an output variable of array type
         Mass2 : ARRAY [0..1] OF  UDINT ;
    end_var
        //declaration of a local variable of array type
         Mass3 : ARRAY [0..1] OF  UDINT ;
    end_var
  end_function_block
```

5. The sequence of default values for array elements is enclosed in square brackets.
6. When initializing array elements, the number of default values must match the number of array elements.

   Example:

```
VAR
        Mass1: ARRAY[0..1] OF UDINT:=[1,5]; //no error
        Mass1: ARRAY[0..1] OF UDINT:=[1,5,7]; //error
        Mass1: ARRAY[0..1] OF UDINT:=[1]; //error
END_VAR
```

Limitations when working with arrays:

1. Bitwise access to array elements is not supported.

   Example:

```
        Mass1: ARRAY[0..1] OF UDINT:=[1,5];
        Mass1[1].1 : = TRUE; //error
```

2. Repeating a sequence of array elements is not supported.

   Example:

```
        Mass1: ARRAY[0..1] OF UDINT:=[1,5,7(2)]; //where (2) stands
for repetition
sequences [1, 2, 7] 2 times)
```

3. Repeating a sequence of array elements is not supported.

   Example:

```
       Mass1[1, 5]:=1; //error: assigning value 1 to first and fifth
array elements
```

Example of *using arrays*.

12.4 Language structures

ST language structures include:

– *arithmetic operations*
– *bit operations*
– *data type conversion operations*
– *logical operations*
– *relational operations*
– *assignment operation*
– ***IF – ELSIF – ELSE** statement*
– ***CASE** statement*
– ***RETURN** statement*
– ***FOR** statement*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

– ***WHILE** statement*
– ***REPEAT – UNTIL** statement*

NOTE
When writing expressions, it is permissible to use variables (input, output and local) and constants.

12.4.1 Operations

12.4.1.1 Arithmetic operations

Operation	Operator	Data types	Example
addition	+	IN, OUT: UDINT/ REAL	OUT := IN1 + IN2 + …
multiplication	*		OUT := IN1 * IN2 * …
subtraction	-		OUT := IN1 - IN2
division	/		OUT := IN1 / IN2
modulo	MOD		OUT := IN1 MOD IN2
absolute value	ABS (IN)	IN, OUT: REAL	OUT := ABS (IN1)
exponentiation	POW (IN, N) IN – base N – exponent	IN, N, OUT: REAL	OUT := POW (IN1, N)

The result of the arithmetic operation is the mathematical result of the expression.
The priority of an operation determines the order of its execution in the expression. Parentheses are allowed to define the calculation order in arithmetic expressions.

12.4.1.2 Bit operations

Operation	Operator	Data types	Example
Bitwise shift left	SHL (IN, N)	IN, OUT: UDINT N: 1..32	OUT := SHL (IN1, N)
Bitwise shift right	SHR (IN, N)		OUT := SHR (IN1, N)
Decoder. Converts binary code to positional code	DC32 (IN)	IN, OUT: UDINT	OUT := DC32 (IN1)
Encoder Converts positional code to binary code	CD32 (IN)	IN, OUT: UDINT	OUT := CD32 (IN1)

12.4.1.3 Data type conversion operations

Operation	Operator	Data types	Example
UDINT to REAL	UDINT_TO_REAL (IN)	IN: UDINT OUT: REAL	OUT := UDINT_TO_ REAL (IN)
UDINT to BOOL	UDINT_TO_BOOL (IN)	IN: UDINT OUT: BOOL	OUT := UDINT_TO_ BOOL (IN)
UDINT to TIME	UDINT_TO_TIME (IN)	IN: UDINT OUT: TIME	OUT := UDINT_TO_ TIME (IN)
UDIMT to DT	UDINT_TO_DT (IN)	IN: UDINT OUT: DT	OUT := UDINT_TO_ DT (IN)

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Operation	Operator	Data types	Example
REAL to UDINT	REAL_TO_UDINT (IN)	IN: REAL OUT: UDINT	OUT := REAL_TO_ UDINT (IN)
REAL to BOOL	REAL_TO_BOOL (IN)	IN: REAL OUT: BOOL	OUT := REAL_TO_ BOOL (IN)
BOOL to REAL	BOOL_TO_REAL (IN)	IN: BOOL OUT: REAL	OUT := BOOL_TO_ REAL (IN)
BOOL to UDINT	BOOL_TO_UDINT (IN)	IN: BOOL OUT: UDINT	OUT := BOOL_TO_ UDINT (IN)
TIME to UDINT	TIME_TO_UDINT (IN)	IN: TIME OUT: UDINT	OUT := TIME_TO_ UDINT (IN)
DT to UDINT	DT_TO_UDINT (IN)	IN: DT OUT: UDINT	OUT := DT_TO_ UDINT (IN)

12.4.1.4 Logical operations

Operation	Operator	Data type	Example
Logical negation	NOT	IN, OUT: BOOL	OUT := NOT IN1
Boolean multiplication	AND &		OUT := IN1 AND IN2 OUT := IN1 & IN2
Boolean addition	OR		OUT := IN1 OR IN2
logical (bitwise) "exclusive OR"	XOR		OUT := IN1 XOR IN2

12.4.1.5 Relational operations

Operation	Operator	Data types	Example
greater than	>	IN: UDINT/REAL OUT: BOOL	OUT := IN1 > IN2
greater than or equal to	>=		OUT := IN1 >= IN2
equal to	=		OUT := IN1 = IN2
less than or equal to	<=		OUT := IN1 <= IN2
less than	<		OUT := IN1 < IN2
not equal to	<>		OUT := IN1 <> IN2

12.4.1.6 Operation priorities

NOTE
The operations in the table are ordered **from highest to lowest priority**. The higher the priority of an operation, the sooner it is executed.

Operation	Operator
Brackets	(expression)
Calling a function and function block	Example: fb1(); function1 := … ;
Bit operations	
Unary minus	–

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Operation	Operator
Logical negation	NOT
Exponentiation	POW
Multiplication	*
Division	/
Modulo	MOD
Addition	+
Subtraction	–
Relational operations	>.< <=, >=
Equal to	=
Not equal to	<>
Conjunction Logical multiplication "AND"	& AND
Exclusive OR	XOR
Disjunction Logical addition "OR"	OR

12.4.2 Assignment operation

The paired symbol "**:=**" is used to indicate assignment. The right and left sides of the expression must contain operands of the same type (automatic type casting is not provided). On the left side of the expression (receiving side) only a variable can be used. The right side can contain an expression or a constant.

12.4.3 IF statement

The **IF** operator allows you to test one or more conditions, and, if at least one of the conditions is true, execute the specified expression conditions. After executing the expressions, the operator exits the statement – that is, the remaining conditions are no longer checked.
Let's consider the operator's work using the example of signaling that the temperature value exceeds the permissible limits:

```
FUNCTION_BLOCK fb //function block name

    VAR_INPUT //declaration of inpur variables
        rTemp : REAL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        xHigh : BOOL;
        xLow : BOOL;
    END_VAR

    VAR //declaration of local variables
        rHighTemp : REAL := 20;
        rLowTemp : REAL := 10;
    END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    //coding area

    IF rTemp > rHighTemp THEN
        xHigh := TRUE;
    ELSIF rTemp < rLowTemp THEN
        xLow := TRUE;
    ELSE
        xHigh := FALSE;
        xLow := FALSE;
    END_IF

END_FUNCTION_BLOCK
```

If the condition in the **IF** statement is true (the value of the variable **rTemp** is greater than **rHighTemp**), then the variable **xHigh** will be assigned the value **TRUE** and the statement will exit the statement (the next condition will not be checked). If the condition is not met, then the next condition placed in the nested **ELSIF** statement will be checked. If the condition in **ELSIF** is satisfied (the value of the variable **rTemp** is less than **rLowTemp**), then the variable **xLow** will be assigned the value **TRUE** and the statement will exit the statement (the next condition will not be checked). If none of the conditions in **IF** and **ELSIF** are met (that is, the temperature value is within acceptable limits), then the expressions placed in the nested **ELSE statement will be executed – the assignment of value FALSE to the xHigh and xLow variables.**
The use of nested **ELSIF** and **ELSE** statements is optional. An arbitrary number of **ELSIF** statements can be placed inside an **IF** statement.
The construction allows nesting, that is, inside one **IF** there can be another one, etc. Also inside the **IF** operator, loops and the ***CASE*** operator can be used.

12.4.4 CASE statement

The **CASE** operator allows you to compare the value of a given integer variable (selector) with a set of constants or integer values (labels), and if there is a match, execute the expressions specified for this label. After executing the expressions, the operator exits the statement.
Example:

```
FUNCTION_BLOCK fb1 //function block name

    VAR_INPUT //declaratio of input variables
        udiSel : UDINT;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        xOut1 : BOOL;
        xOut2 : BOOL;
        xOut3 : BOOL;
        xOut4 : BOOL
    END_VAR

    //coding area

    xOut1 := FALSE;
    xOut2 := FALSE;
    xOut3 := FALSE;
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    xOut4 := FALSE;

    CASE udiSel OF
        0:
            xOut1 := TRUE;
        1..3:
            xOut2 := TRUE;
        4, 6:
            xOut3 := TRUE;
    ELSE
        xOut4 := TRUE;
    END_CASE

END_FUNCTION_BLOCK
```

If the **udiSel** value is:

– Equal to 0, then **xOut1** will take the value **TRUE**;
– Falling into the range 1..3, then **xOut2** will take the value **TRUE**;
– Equal to 4 or 6, then **xOut3** will take the value **TRUE**;
– Not falling into any of the specified values, then **xOut4** will take the value **TRUE**;

As can be seen from the example, a label can include several values, listed, separated by commas “4, 6”, or the range “1..3”. In this case, values of one of the labels should not coincide with the values of the others. Also, when specifying a range of values, the beginning of the range must be less than its end.
The nested **ELSE** statement is optional; the expressions placed in it are executed if the selector value does not match any of the labels.
The actions provided to handle each of the **CASE** statement cases can use loops, **IF** statements, and **CASE** statements.

12.4.5 RETURN statement

The **RETURN** statement allows you to exit a program object.
Usage example:

```
IF xDone THEN
    RETURN;
END_IF;

udiCounter := udiCounter + 1;
```

If the variable **xDone** takes the value **TRUE**, then the expression “**udiCounter** := **udiCounter** + 1” will not be executed will be (like everything that will be located below in the body of the program).

12.4.6 FOR statement

The **FOR** operator is used to organize a loop with a predetermined number of iterations. It is usually used for operations on arrays of data.
The **IF** and **CASE** operators, as well as other loop operators, can be used inside a loop.
As an example, consider the implementation of bubble sort from smallest to largest:

```
FUNCTION_BLOCK MaxI_MinI
//Maximum and minimum numbers using bubble sort from smallest to largest
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    VAR_INPUT //declaration of input variables
        udiX1, udiX2, udiX3, udiX4, udiX5 : UDINT;
```

```
//Adding the required number of input variables and defining the data type
    END_VAR

    VAR_OUTPUT //declaration of output variables
        udiMaxI, udiMinI : UDINT;
    END_VAR

    VAR //declaration of local variables
        udiI, udiJ, udiN, udiK : UDINT;
        audiX : ARRAY [1..5] OF UDINT;
//Specifies the range of the array and the data type of the array
    END_VAR

    //coding area
    //array declaration
    audiX[1] := udiX1;
    audiX[2] := udiX2;
    audiX[3] := udiX3;
    audiX[4] := udiX4;
    audiX[5] := udiX5;
    udiN := 5; // the number of numbers to sort is specified (array size)

    FOR udiI := 1 TO udiN-1 DO
        FOR udiJ := 1 TO udiN-udiI DO
            IF audiX[udiJ] > audiX[udiJ+1] THEN
                udiK := audiX[udiJ];
                audiX[udiJ] := audiX[udiJ+1];
                audiX[udiJ+1] := udiK;
            END_IF;
        END_FOR;
    END_FOR;

    udiMaxI := audiX[udiN]; //the last (maximum) number of the array is displayed
    udiMinI := audiX[1]; //the first (minimum) number of the array is displayed

END_FUNCTION_BLOCK
```

The **udiI** variable is called the loop iterator (counter). This variable must be a signed integer type (**UDINT**). After each execution of the loop body, the iterator value changes - by default to **+1**. The user can set the iterator “step” using the nested **BY** operator. After this, the transition immediately occurs to the next iteration - that is, the entire cycle is executed “from beginning to end”. The structure will look like this:

```
// udiI will take the values 1, 4, 7, 10, etc.
FOR udiI := 1 TO udiN-1 BY 3 DO
    .. // loop code
END_FOR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The loop iterator (counter), its start and end values, and its increment are integer values. They are calculated before entering the loop, and changing the values of the variables included in any of these expressions will not change the number of iterations.
You can exit a loop early using the **EXIT** operator. Example structure:

```
FOR udiI := 1 TO udiN-1 DO
    IF audiX[2] > 100 THEN
        EXIT;
    ELSE
        .. // loop code
    END_IF
END_FOR
```

You can skip a loop step using the **CONTINUE** operator.

```
FOR udiI := 1 TO udiN-1 DO
    IF udiI = 3 THEN
        CONTINUE;
    ELSE
        .. // loop code
        END_IF
END_FOR
```

12.4.7 WHILE statement

The **WHILE** operator is used to create a loop with an unknown number of iterations. The loop will terminate if the condition being tested returns **FALSE**. In this case, the condition is checked BEFORE the expression is executed (a loop with a precondition). Thus, if the condition immediately returns **FALSE**, then the loop will not be executed even once.
The **IF** and **CASE** operators, as well as other loop operators, can be used inside a loop.
Example

```
WHILE rVar < 100 DO
    rVar := rVar + 1;
END_WHILE
```

The result of executing this loop (with the initial value of the variable **rVar** := 10) will be the number 100.
You can exit a loop early using the **EXIT** statement.
You can skip a loop step using the **CONTINUE** operator.

12.4.8 REPEAT UNTIL statement

The **REPEAT** statement is used to create a loop with an unknown number of iterations. The loop will terminate if the condition being tested returns **TRUE**. In this case, the condition is checked AFTER the expression is executed (a loop with a postcondition). Thus, if the condition immediately returns **TRUE**, then the loop will be executed once.
The **IF** and **CASE** operators, as well as other loop operators, can be used inside a loop.
Example:

```
REPEAT
    IF rVar > 100 THEN
        EXIT;
    END_IF;
    rVar := rVar + 1;
UNTIL rVar > 180
END_REPEAT;
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The result of executing this loop (with the initial value of the variable **rVar** := 10) will be the number 101.
To exit a loop early, you can use the **EXIT** operator.
You can skip a loop step using the **CONTINUE** operator.

12.5 System functions

The ST editor supports the following system functions:
– *Date and Time functions*
– *Mathematical functions*
– *Logarithmic functions*
– *Trigonometric functions*

12.5.1 Date and Time functions

– *GET_TIME function*
– *GET_DATE_TIME function*
– *SYS.COMPARE_DATE_TIME function*
– *SYS.IS_LEAP_YEAR function*

12.5.1.1 GET_TIME function

The **GET_TIME** function returns a **TIME** type value (4 bytes) containing the time elapsed from the moment the device was last turned on, in milliseconds.
Example

```
VAR
    Time_1 : TIME := T#0ms;
    Time_2 : TIME := T#0ms;
    Q :      BOOL := FALSE;
END_VAR

IF Time_1 = T#0ms THEN
   Time_1 := GET_TIME();
END_IF

Time_2 := GET_TIME();

IF (Time_2 - Time_1) >= T#1000ms THEN
    Q      := NOT Q;
    Time_1 := T#0ms;
    Time_2 := T#0ms;
END_IF
```

12.5.1.2 GET_DATE_TIME function

The **GET_DATE_TIME** function returns a **DT** type value (4 bytes) containing real time clock data, in seconds from 00:00:00 01/01/2000, taking into account the time zone set in the device.
Example

```
VAR
    Ton_UDINT : UDINT;
    Ton_DT : DT;
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
END_VAR

Ton_DT := GET_DATE_TIME();

Ton_UDINT := DT_TO_UDINT(Ton_DT);
```

12.5.1.3 SYS.COMPARE_DATE_TIME function

The **SYS.COMPARE_DATE_TIME** function compares two UDINT values given as input using a given date/time mask and returns a UDINT value that evaluates to:

– 1 – value 1 is greater than value 2
– 2 – value 1 is less than value 2
– 0 – value 1 is equal to value 2

The comparison is made by the number of seconds starting from 00:00:00 01.01.2000

Example

```
FUNCTION udiCompare: UDINT;

    VAR_INPUT
        udiValue1:  UDINT; //value 1
    END_VAR

    VAR
        udiValue2 : UDINT; //value 2
        udiMask :   UDINT := 63; //date/time mask
    END_VAR

    udiValue2  := DT_TO_UDINT (GET_DATE_TIME ());
    udiCompare := SYS.COMPARE_DATE_TIME (udiValue1, udiValue2, udiMask);

END_FUNCTION
```

Mask = 63 (0b111111) – all bits are used:
0 bit – if 1, then seconds are used.
1 bit – if 1, then minutes are used.
2 bits – if 1, then the hours are used.
3 bits – if 1, then days are used.
4 bits – if 1, then months are used.
5 bits – if 1, then years are used.

12.5.1.4 SYS.IS_LEAP_YEAR function

The **SYS.IS_LEAP_YEAR** function returns a BOOL value containing data on whether the UDINT number given as an input corresponds to a leap year (1 - leap year, 0 - not).

Example

```
FUNCTION xLeapYear: BOOL; //function name and output data type

    VAR_INPUT //declaration of input variables
        udiYear : UDINT; //year being checked
    END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    xLeapYear := SYS.IS_LEAP_YEAR (udiYear);

END_FUNCTION
```

12.5.2 Mathematical functions

- *Mathematical constants*
- *MIN/FMIN*
- *MAX/FMAX*
- *SQRT*
- *EXP*
- *FACT*

12.5.2.1 Mathematical constants (CONST.*)

Name	Value	Description
CONST.PI	3.1415927	the number Pi (π)
CONST.E	2.7182817	Euler number (e)
CONST.POS_INF	+ ∞	positive infinity
CONST.NEG_INF	– ∞	negative infinity

12.5.2.2 MIN/FMIN

The **MIN/FMIN** functions return the smallest value of two input variables.
Input variables and a return value for the function:

- **MIN** — for type UDINT (both arguments and a result)
- **FMIN** — for type REAL (both arguments and a result)

Example 1

```
VAR
    Var_1: UDINT;
    Var_2: UDINT := 7;
    Var_3: UDINT := 5;
END_VAR
Var_1 := SYS.MIN(Var_2, Var_3); // return value is 5
```

Example 2

```
VAR
    Var_1: REAL;
    Var_2: REAL := 7.1;
    Var_3: REAL := 8.2;
END_VAR
Var_1 := SYS.FMIN(Var_2, Var_3); // return value is 7.1
```

12.5.2.3 MAX/FMAX

The **MAX/FMAX** functions return the largest value of two input variables.
Input variables and a return value for the function:

- **MAX** — for type UDINT (both arguments and a result)
- **FMAX** — for type REAL (both arguments and a result)

Example 1

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
VAR
    Var_1: UDINT;
    Var_2: UDINT := 7;
    Var_3: UDINT := 8;
END_VAR
Var_1 := SYS.MAX(Var_2, Var_3); // return value is 8
```

Example 2

```
VAR
    Var_1: REAL;
    Var_2: REAL := 7.5;
    Var_3: REAL := 5.2;
END_VAR
Var_1 := SYS.FMAX(Var_2, Var_3); // return value is 7.5
```

12.5.2.4 SQRT

The **SQRT** function returns the square root of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 4.0;
END_VAR
Var_1 := SYS.SQRT(Var_2); // return value is 2.0
```

12.5.2.5 EXP

The **EXP** function returns the exponent value raised to the power of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 1.0;
END_VAR
Var_1 := SYS.EXP(Var_2); // return value is 2.7182817
```

12.5.2.6 FACT

The **FACT** function returns the factorial of the input number. The function returns the correct values when the input variable value is within the range from 0 to 12. In all other cases, the function returns 0.
An input variable and a return value of type UDINT.
Example

```
VAR
    Var_1: UDINT;
    Var_2: UDINT := 5;
END_VAR
Var_1 := SYS.FACT(Var_2); // return value is 120
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.5.3 Logarithmic functions

– *LOG*
– *LG*
– *LN*

12.5.3.1 LOG

The **LOG** function returns the logarithm of the input number with an arbitrary base.
An input variable and a return value of type REAL.

Base of logarithm	Number of logarithm	Result
< 0	Any	NaN*
NaN	1	0
+ inf**	0	NaN
+ inf	> 0	0
+ inf	+ inf	NaN
0	> 0, ≠ 1	0
0	0	NaN
0	+ inf	NaN
0	1	0
1	Any	NaN
NaN	NaN	NaN
> 0, ≠ 1	> 0	Number
> 0, ≠ 1	0	- inf***
> 0, ≠ 1	+ inf	+ inf

*NaN (Not-a-Number) — not a number
**+ inf (infinity) — positive infinity
***- inf (infinity) — negative infinity

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 2; // Base of logarithm
    Var_3: REAL := 8; // Number of logarithm
END_VAR
Var_1 := SYS.LOG(Var_2, Var_3); // return value is 3
```

12.5.3.2 LG

The **LG** function returns the decimal logarithm of the input number.
An input variable and a return value of type REAL.

Number of logarithm	Result
0	- inf*
< 0	NaN**
NaN	NaN
+ inf***	+ inf

*- inf (infinity) — negative infinity
**NaN (Not-a-Number) — not a number
**+ inf (infinity) — positive infinity

Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 10;
END_VAR
Var_1 := SYS.LG(Var_2); // return value is 1
```

12.5.3.3 LN

The **LN** function returns the natural logarithm of the input number.
An input variable and a return value of type REAL.

Number of logarithm	Result
0	- inf*
< 0	NaN**
NaN	NaN
+ inf***	+ inf

*- inf (infinity) — negative infinity
**NaN (Not-a-Number) — not a number
**+ inf (infinity) — positive infinity

Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 2.718281;
END_VAR
Var_1 := SYS.LN(Var_2); // return value is ~1
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.5.4 Trigonometric functions

- *SIN*
- *COS*
- *TAN*
- *ASIN*
- *ACOS*
- *ATAN*

12.5.4.1 SIN

The **SIN** function returns the sin (sine) value of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 1.5708; // in radians
END_VAR
Var_1 := SYS.SIN(Var_2); // return value is 1
```

12.5.4.2 COS

The **COS** function returns the cos (cosine) value of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 1.0472; // in radians
END_VAR
Var_1 := SYS.COS(Var_2); // return value is 0.49999794
```

12.5.4.3 TAN

The **TAN** function returns the value of the tan (tangent) of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 1.3734; // in radians
END_VAR
Var_1 := SYS.TAN(Var_2); // return value is 4.999982
```

12.5.4.4 ASIN

The **ASIN** function returns the value of the arcsin (arcsine) of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 0.5;
END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
Var_1 := SYS.ASIN(Var_2); // return value is 0.52359825
```

NOTE
The relative error of the function result is 1e-5.

12.5.4.5 ACOS

The **ACOS** function returns the arccos (arccosine) value of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 0.5;
END_VAR
Var_1 := SYS.ACOS(Var_2); // return value is 1.047198
```

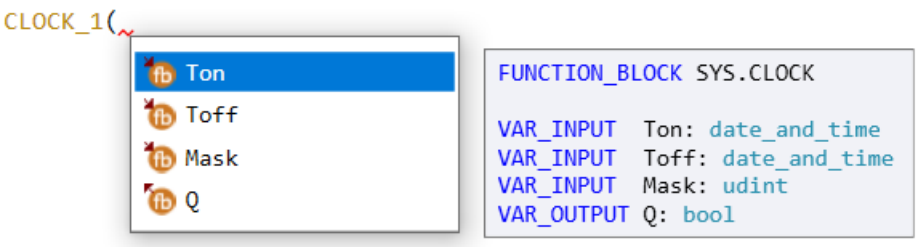
12.5.4.6 ATAN

The **ATAN** function returns the arctg (arctangent) value of the input number.
An input variable and a return value of type REAL.
Example

```
VAR
    Var_1: REAL;
    Var_2: REAL := 1.73205; // 3^(1/2)
END_VAR
Var_1 := SYS.ATAN(Var_2); // return value is 1.0471973
```

12.6 System Function Blocks

The ST editor supports the following system function blocks:
– *Triggers*
– *Timers*
– *Generators*
– *Counters*
The use of system function blocks in other program elements is similar to *using custom function blocks*.
For each system function block, the editor provides a hint on how to use it.



12.6.1 Triggers

– *RS trigger reset dominant (SYS.RS)*
– *SR trigger set dominant (SYS.SR)*
– *Rising edge (SYS.RTRIG)*
– *Falling edge (SYS.FTRIG)*
– *D-trigger (SYS.DTRIG)*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.6.1.1 RS trigger reset dominant (SYS.RS)

The RS trigger reset dominant (SYS.RS) is used to switch with state fixation during the receipt of short pulses at the corresponding input. The output Q will show HIGH level signal at the edge of the signal from the input S.

Designation	Data type	Description
Inputs		
S	BOOL	SET input
R	BOOL	RESET input
Outputs		
Q	BOOL	Trigger output

```
FUNCTION_BLOCK RS_trigger //Function block name

    VAR_INPUT //declaration of input variables
        R_in : BOOL;
        S_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        RS_1: SYS.RS;
    END_VAR

    //code area

    RS_1(R := R_in, S := S_in, Q => Q_out);

END_FUNCTION_BLOCK
```

To prevent simultaneous receipt of signals at both inputs, the signal of input R takes priority.

12.6.1.2 SR trigger set dominant (SYS.SR)

The SR trigger set dominant (SYS.SR) is used to switch with state fixation during the receipt of short pulses at the corresponding input. The output Q will show HIGH level at the edge of the signal from the input S.

Designation	Data type	Description
Inputs		
S	BOOL	SET input
R	BOOL	RESET input
Outputs		
Q	BOOL	Trigger output

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
FUNCTION_BLOCK SR_trigger //Function block name

    VAR_INPUT //declaration of input variables
        S_in : BOOL;
        R_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        SR_1: SYS.SR;
    END_VAR

    //code area

    SR_1(S := S_in, R := R_in, Q => Q_out);

END_FUNCTION_BLOCK
```

To prevent simultaneous receipt of signals at both inputs, the signal of input S takes priority.

12.6.1.3 Rising edge (SYS.RTRIG)

The rising edge (SYS.RTRIG) is used when it is necessary to have a reaction to a change in the state of a digital input signal. Output Q generates a single pulse on the rising edge of the input I.

Designation	Data type	Description
Input		
I	BOOL	Trigger input
Output		
Q	BOOL	Trigger output

```
FUNCTION_BLOCK R_trigger //Function block name

    VAR_INPUT //declaration of input variables
        RT_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        RT_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        RTrig_1: SYS.RTRIG;
    END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    //code area

    RTrig_1(I := RT_in, Q => RT_out);

END_FUNCTION_BLOCK
```

12.6.1.4 Falling edge (SYS.FTRIG)

The falling edge (SYS.FTRIG) is used when it is necessary to have a reaction to a change in the state of a digital input signal. Output Q generates a single pulse on the leading edge of the input I.

Designation	Data type	Description
Input		
I	BOOL	Trigger input
Output		
Q	BOOL	Trigger output

```
FUNCTION_BLOCK F_trigger //Function block name

    VAR_INPUT //declaration of input variables

        FT_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        FT_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        FTrig_1: SYS.FTRIG;
    END_VAR

    //code area

    FTrig_1(I := FT_in, Q => FT_out);

END_FUNCTION_BLOCK
```

12.6.1.5 D-trigger (SYS.DTRIG)

D-trigger (SYS.DTRIG) is used to generate a pulse to turn on the output for the time interval of the pulse at the D input. Output interval synchronizes with the clock frequency at the C input.
At the Q trigger output, a HIGH level signal will appear on the front of the clock pulses at the C input if there is a HIGH level signal at the D input. Output Q returns to the LOW level signal on the front of the clock pulses at the C input if there is a HIGH level signal at the D input.
Input S forces output Q to a HIGH level state.
Input R is the priority input and sets output Q to LOW level.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Designation	Data type	Description
Inputs		
S	BOOL	SET input
D	BOOL	Trigger input
C	BOOl	Clock frequency
R	BOOL	RESET input
Oupputs		
Q	BOOL	Trigger output

```
FUNCTION_BLOCK D_trigger //Function block name

    VAR_INPUT //declaration of input variables
        S_in : BOOL;
        D_in : BOOL;
        C_in : BOOL;
        R_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        DTrig_1: SYS.DTRIG;
    END_VAR

    //code area

    DTrig_1(S := S_in, D := D_in, C := C_in, R := R_in, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.2 Timers

- *Pulse (SYS.TP)*
- *ON-delay timer (SYS.TON)*
- *OFF-delay timer (SYS:TOF)*
- *Timer (SYS.CLOCK)*
- *Weekly timer (SYS:CLOCKWEEK)*

12.6.2.1 Pulse (SYS.TP)

The pulse (SYS.TP) is used to generate a pulse to turn on the output for a specified time interval. A HIGH level signal appears at the output Q of the block at the edge of the input signal I. After starting, the output Q does not respond to a change in the value of the input signal during the interval T. When the interval T has expired, the output signal is reset to LOW level.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Designation	Data type	Description
Inputs		
I	BOOL	Turning on the timer
T	TIME	Pulse duration
Outputs		
Q	BOOL	Timer output

```
FUNCTION_BLOCK TP_timer //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        TP_1: SYS.TP;
    END_VAR

    //code area
    TP_1(I := I_in, T := T#1000ms);
    //where ms is milliseconds, s is seconds, m is minutes, h is hours, d is days
    Q_out := TP_1.Q;
END_FUNCTION_BLOCK
FUNCTION_BLOCK TP_timer //milliseconds

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
        T_in : UDINT := 5000;//milliseconds
    END_VAR

    VAR_OUTPUT //declaration of output variables

        Q_out : BOOL;
    END_VAR

    VAR
        TP_1: SYS.TP;
        T_time: TIME;
    END_VAR

    //code area
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    T_time := UDINT_TO_TIME(T_in);
    TP_1(I := I_in, T := T_time, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.2.2 ON-delay timer (SYS.TON)

The ON-delay timer (SYS.TON) is used to delay the signal transmission.. The timer output Q will produce a HIGH level signal with a delay relative to the input signal front I for time more than T and will turn off at the input signal fall.

Designation	Data type	Description
Inputs		
I	BOOL	Timer start (on rising edge)
T	TIME	Delay on power-on
Outputs		
Q	BOOL	Timer output

```
FUNCTION_BLOCK TON_timer //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        TON_1: SYS.TON;
    END_VAR

    //code area

    TON_1(I := I_in, T := T#1000ms);
    //where ms is milliseconds, s is seconds, m is minutes, h is hours, d is days
    Q_out := TON_1.Q;

END_FUNCTION_BLOCK
FUNCTION_BLOCK TON_timer //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
        Ton_in : UDINT := 5000;//milliseconds
    END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        TON_1: SYS.TON;
        Ton_time: TIME;
    END_VAR

    //code area

    Ton_time := UDINT_TO_TIME(Ton_in);
    TON_1(I := I_in, T := Ton_time, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.2.3 OFF-delay timer (SYS.TOF)

The OFF-delay timer (SYS.TOF) is used to delay the output off. The timer output Q will show a HIGH level signal on the rising edge of the signal at the input I, the countdown of the off-delay time T will start on each falling edge of the input signal. After the input signal is off, the output will show a LOW level signal with a delay of T.

Designation	Data type	Description
Inputs		
I	BOOL	Timer start (on falling edge)
T	TIME	Delay on power-on
Outputs		
Q	BOOL	Timer output

```
FUNCTION_BLOCK TOF_timer //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        TOF_1: SYS.TOF;
    END_VAR

    //code area
```

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    TOF_1(I := I_in, T := T#1000ms);
    //where ms is milliseconds, s is seconds, m is minutes, h is hours, d is days
    Q_out := TOF_1.Q;

END_FUNCTION_BLOCK
FUNCTION_BLOCK TOF_timer //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
        Tof_in : UDINT := 5000;//milliseconds
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        TOF_1: SYS.TOF;
        Tof_time: TIME;
    END_VAR

    //code area

    Tof_time := UDINT_TO_TIME(Tof_in);
    TOF_1(I := I_in, T := Tof_time, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.2.4 Timer (SYS.CLOCK)

The timer (SYS.CLOCK) is used to generate a pulse to turn on the Q output according to the real-time clock. The output turn-on time T_{oT} and turn-off time T_{off} are set as timer parameters.

Designation	Data type	Description
Inputs		
Ton	DT	Turn-on time
Toff	DT	Shutdown time
Mask	UDINT	Selection of quantities to be used
Output		
Q	BOOL	Timer output

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

NOTE
Specifying the Mask variable is optional.
If the value of the Mask variable is not specified, the block defaults to a Mask = 63 (0b111111),
Where:
Mask = 63 (0b111111)
0 bit – if 1, then seconds are used
1 bit – if 1, then minutes are used
2 bits – if 1, then the hours is used
3 bits – if 1, then days are used
4 bits – if 1, then months are used
5 bits – if 1, then years are used

```
FUNCTION_BLOCK CLOCK_timer //Function block name

    VAR_INPUT //declaration of input variables
       I_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        CLOCK_1: SYS.CLOCK;
    END_VAR

    //code area


    CLOCK_1(Ton := DT#2023-09-28-7:20:55, Toff := DT#2023-09-28-12:30:59);

    Q_out := CLOCK_1.Q;

END_FUNCTION_BLOCK
```

12.6.2.5 Weekly timer (SYS.CLOCKWEEK)

The weekly timer (SYS.CLOCKWEEK) is used to generate a pulse to turn on the output Q according to the real-time clock, taking into account the days of the week. The time of turning on T_{on} and turning off T_{off} from the output Q and the days of the week of operation are set as timer parameters.

Designation	Data type	Description
Inputs		
Ton	DT	Turn-on time
Toff	DT	Shutdown time
DayOfWeekMask	UDINT	Selecting the days to use
DateTimeMask	UDINT	Selection of quantities to be used

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Designation	Data type	Description
Outputs		
Q	BOOL	Timer output

NOTE
Specifying the DayOfWeekMask and DateTimeMask variables is optional.
If the value of the DayOfWeekMask variable is not specified, then the block defaults to a DayOfWeekMask = 127 (0b1111111),
If the value of the DateTimeMask variable is not specified, then the block defaults to a DateTimeMask = 63 (0b111111),
Where:
DayOfWeekMask = 127 (0b1111111)
0 bit – if 1, then Mondays are taken into account
1 bit – if 1, then Tuesdays are taken into account
2 bits – if 1, then Wednesdays are taken into account
3 bits – 1, then Thursdays are taken into account
4 bits – if 1, then Fridays are taken into account
5 bits – if 1, then Saturdays are taken into account
6 bits – if 1, then Sundays are taken into account

DateTimeMask = 63 (0b111111)
0 bit - if 1, then seconds are used
1 bit - if 1, then minutes are used
2 bits - if 1, then hours are used
3 bits - if 1, then days are used
4 bits - if 1, then months are used
5 bits - if 1, then years are used

```
FUNCTION_BLOCK CLOCKWEEK_timer //Function block name

    VAR_INPUT //declaration of input variables
       I_in : BOOL;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        CLOCKWEEK_1: SYS.CLOCKWEEK;
    END_VAR

    //code area

    CLOCKWEEK_1(Ton := DT#2023-09-28-7:20:55, Toff := DT#2023-09-28-12:30:59);
    Q_out := CLOCKWEEK_1.Q;
END_FUNCTION_BLOCK
```

12.6.3 Generators

– *Pulse generator (SYS.BLINK)*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

12.6.3.1 Pulse generator (SYS.BLINK)

The pulse generator (SYS.BLINK) is used to form rectangular pulses. At the output Q of the generator, pulses are formed with specified parameters of the duration of the on (Th – HIGH level signal) and off (Tl – LOW level signal) state for the duration of the control signal at the input I (HIGH level signal).

Designation	Data type	Description
Inputs		
I	BOOL	Work permission
Th	TIME	Duration of a logical unit
Tl	TIME	Logical zero duration
Outputs		
Q	BOOL	Generator output

```
FUNCTION_BLOCK BLINK_generator //Function block name

    VAR_INPUT //declaration of input variables
       I_in : BOOL := FALSE;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR
        BLINK_1: SYS.BLINK;
    END_VAR

    //code area

    BLINK_1(I := I_in, Th := T#1000ms, Tl := T#1000ms);
    //where ms is milliseconds, s is seconds, m is minutes, h is hours, d is days

    Q_out := BLINK_1.Q;

END_FUNCTION_BLOCK
FUNCTION_BLOCK BLINK_generator //Function block name

    VAR_INPUT //declaration of input variables
        I_in : BOOL := FALSE;
        Th_in : UDINT := 5000;//milliseconds
        Tl_in : UDINT := 5000;//milliseconds
    END_VAR
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    VAR_OUTPUT //declaration of output variables
```

```
        Q_out : BOOL;
    END_VAR

    VAR
        BLINK_1: SYS.BLINK;
        Th_time: TIME;
        Tl_time: TIME;
    END_VAR

    //code area

    Th_time := UDINT_TO_TIME(Th_in);
    Tl_time := UDINT_TO_TIME(Tl_in);
    BLINK_1(I := I_in, Th := Th_time, Tl := Tl_time, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.4 Counters

- *Threshold counter with self-reset (SYS.CT)*
- *Universal counter (SYS.CTN)*
- *Threshold counter (SYS.CTU)*

12.6.4.1 Threshold counter with self-reset (SYS.CT)

The threshold counter with self-reset (SYS.CT) is used to count a specified number of pulses N (input N is the pulse number setting). At the output Q of the counter, a pulse of the HIGH level signal with the duration of the device working cycle (cycle time) will appear if the number of pulses arriving at the input C reaches the set value N.

Designation	Data type	Description
Inputs		
C	BOOL	Counter input
N	UDINT	Counter setting
Outputs		
Q	BOOL	Signaling of the setpoint reached (for one cycle)

```
FUNCTION_BLOCK CT_counter //Function block name

    VAR_INPUT //declaration of input variables
        C_in : BOOL;
        N_in : UDINT := 10;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : BOOL;
    END_VAR

    VAR //declaration of local variables
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
        CT_1: SYS.CT;
    END_VAR

    //code area

    CT_1(C := C_in, N := N_in, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.4.2 Universal counter (SYS.CTN)

The universal counter (SYS.CTN) is used for direct and indirect counting. The "direct counting" operation is performed by the rising edge of the pulse at the direct counting input U, which increases the value of the output signal Q. Incoming pulse on the input D ("decrease counting") decrease the value of the output Q. If a logical "1" signal comes to the input R, the output of the counter Q is set to the value of the input N.

Designation	Data type	Description
Inputs		
U	BOOL	Direct Counting
D	BOOL	Countdown
R	BOOL	Reset the output state to value N
N	UDINT	Setpoint
Outputs		
Q	UDINT	Cumulative value of pulses

```
FUNCTION_BLOCK CTN_counter //Function block name

    VAR_INPUT //declaration of input variables
        U_in : BOOL;
        D_in : BOOL;
        R_in : BOOL;
        N_in : UDINT := 10;
    END_VAR

    VAR_OUTPUT //declaration of output variables
        Q_out : UDINT;
    END_VAR

    VAR //declaration of local variables

        CTN_1: SYS.CTN;
    END_VAR

    //code area
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
    CTN_1(U := U_in, D := D_in, R := R_in, N := N_in, Q => Q_out);

END_FUNCTION_BLOCK
```

12.6.4.3 Threshold counter (SYS.CTU)

The threshold counter (SYS.CTU) is used to count the number of incoming pulse on the C input. A pulse of HIGH level signal will appear at the Q counter output if the number of incoming pulses on the input reaches the set value at the N input (N is the setpoint).

Designation	Data type	Description
Inputs		
U	BOOL	Direct Counting
R	BOOL	Reset the counter state to 0
N	UDINT	Setpoint
Outputs		
Q	BOOL	Signaling of setpoint reached

```
FUNCTION_BLOCK CTU_counter //Function block name

    VAR_INPUT //declaration of input variables
        C_in : BOOL;
        R_in : BOOL;
        N_in : UDINT := 10;
    END_VAR

    VAR_OUTPUT //declaration of output variables

        Q_out : BOOL;
    END_VAR

    VAR //declaration of local variables
        CTU_1: SYS.CTU;
    END_VAR

    //code area

    CTU_1(C := C_in, R := R_in, N := N_in, Q => Q_out);

END_FUNCTION_BLOCK
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000