EN

akYtec ALP

Programming software for akYtec programmable devices

User manual

ALP_2019.03_V12_EN

CE

akYtec GmbH · Vahrenwalder Str. 269 A · 30179 Hannover · Germany · Tel.: +49 (0) 511 16 59 672-0 · www.akytec.de

Contents

1 General

1.1 Abbreviations and terms

Abbreviations and terms used in this manual:

Table 1.1

Abbreviations and terms	Explanation
akYtec ALP	Programming software
FBD	Function Block Diagram is a graphical programming language
Project	A user application created for a programmable device with akYtec ALP software
Target device	A device for which the project is created
Function	A structural unit of a program 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 will return the same output value.
Function block	A structural unit of a program with internal memory and one or more output values. Many instances (copies) of a function block can be created.
Macro	A function block created by the user
Workspace	An area in the software user interface to create the user program by placing graphic components and links between them.
Program cycle	The execution time of the circuit program, which depends on its complexity

1.2 About software

akYtec ALP is a programming software for programmable devices of akYtec GmbH. The programming language is the graphical language FBD (Function block diagram). The software enables testing of the created program and its upload to the non-volatile memory of a programmable device.

Each project contains one or several circuit programs and a device configuration.

The first workspace contains the main circuit program. Macros (user function blocks) can be created as circuit programs on separate workspaces.

If the target device has a display, it can be programmed using display forms, each of them is opened in separate workspace.

Only one project at a time can be opened.

1.3 System requirements

akYtec ALP software runs on Windows XP/Vista/7/8/10. Pre-installed software .NET Framework 4.0 is required. If not installed, you will be asked to install it.

Minimum hardware requirements:

- 1.5 GHz processor
- 1 GB RAM
- 100 MB available hard disk space
- Free USB port
- Keyboard and mouse

- Screen resolution 1024x768

Recommended hardware requirements:

- 3.2 GHz processor
- 4 GB RAM
- 200 MB available hard disk space
- Free USB port
- Keyboard and mouse
- Screen resolution 1280x800

Internet connection is required for:

- Software update
- Device firmware update
- Slave device template download
- Macros download in Component Manager

2 User interface

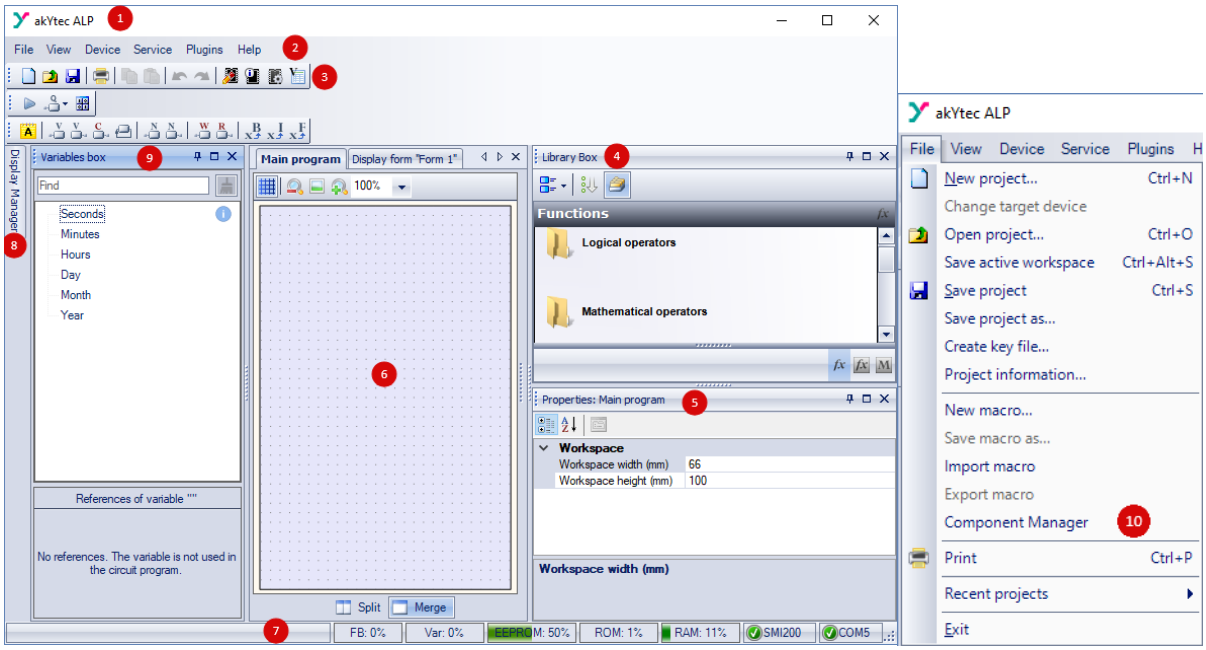


Fig. 2.1

1. ***Title bar*** – shows the name of the software and the path to the open project file
2. ***Menu bar*** – consists of the following groups: File, View, Device, Service, Plugins and Help
3. ***Toolbars*** – Standard, Service and Insert: quick access to the essential functions of ALP
4. ***Library Box*** – a panel that displays all the functions, function blocks, and macros that can be added to the project
5. ***Property Box*** – a panel where the properties of the selected project element can be viewed and modified
6. ***Workspace*** – a field in the user interface where a circuit program, display structure or a display form can be viewed and modified
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 displayed information (available only for devices with display)
9. ***Variable Box*** – a panel in which all project variables with their parameters and references are displayed. Use drag-and-drop to place a variable block in a circuit program.
10. ***Component Manager*** – a special tool in a separate window to access the Online Data Base and to add the elements from Online Data Base to an offline library or to the project library. Internet connection is needed.

2.1 Menu

Table 2.1 Menu "File"

New project	Open a new project. The current project will be closed. (3.3)
Change target device	Change the target device in the project
Open project	Open a previously saved project

Save current work-space	Save the currently opened workspace
Save project	Save the current project
Save project as	Make a copy of the project in a different folder or with a differ-ent name
Create key file	Create a file with a key to protect the project from unauthorized access (in development)
Project information	View and modify the information about the project (3.5)
New macro	Open the new macro in the separate workspace (6.3.1)
Save macro as	Save the current macro under a new name in the project library
Import macro	Import a macro from a file into the project library (6.3.3)
Export macro	Save the current macro as a file (6.3.2)
Component Manager	Open the Component Manager interface (see 2.9)
Print	Open the dialog to set the print options for the current work-space
Recent projects	List of recently opened projects
Exit	Close akYtec ALP

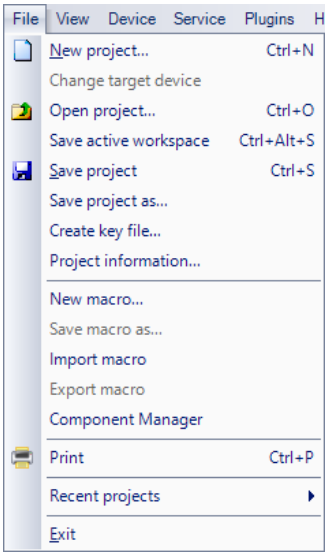


Fig. 2.2 Menu “File”

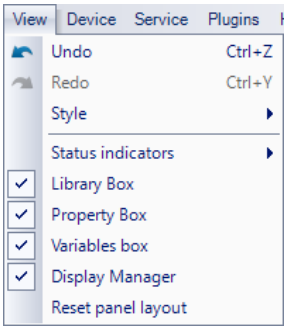


Fig. 2.3 Menu “View”

Table 2.2 Menu “View”

Undo	Undo the last action
Redo	Redo the last undone action
Style	Select the color design of the window
Status indicators	Add / remove the indicators to / from the status bar (2.7)
Library Box	Show / hide Library Box (2.4)
Property Box	Show / hide Property Box (2.5)
Variable Box	Show / hide Variable Box (2.8)
Display Manager *	Show / hide Display Manager (2.6)

** Available only for devices with display*

Table 2.3 Menu “Device”

Transfer application to device	Upload the current project to the device memory (3.7)
Firmware update	Update the firmware of the connected device (3.8)
Device information	Information about the software, the target device and the connected device (3.5)
Variables	The editable table of the project variables with their parameters (5)
Calibration	Start calibration (4.7)
Configuration	Device configuration (4)
Port settings	Settings of the programming interface (3.4)

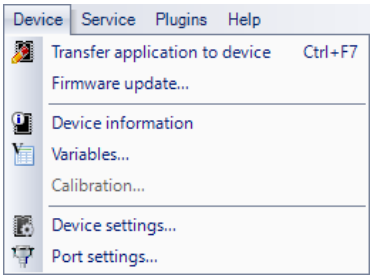


Fig. 2.4 Menu “Device”

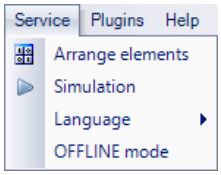


Fig. 2.5 Menu “Service”

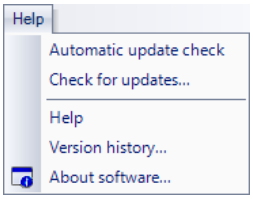


Fig. 2.6 Menu “Help”

Table 2.4 Menu “Service”

Arrange elements	Function blocks of the same type are automatically renumbered in the workspace from top to bottom and from right to left (7.9)
Simulation	Start the simulation mode (7.11)
Language	Select the interface language
OFFLINE mode	Activate OFFLINE mode (3.4.1)

Table 2.5 Menu “Help”

Automatic update check	If activated, the update check is performed on program startup
Check for updates	Check if software updates are available
Help	Online help
Version history	The list of changes for all software versions
About Software	Information about the current software version

2.2 Toolbars

Table 2.6

Standard		
		
	New project	Open a new project. The current project will be closed.
	Open project	Open a previously saved project
	Save project	Save the current project

	Print	Open the dialog to set the print options for the current workspace
	Copy	Copy the element selected in the workspace
	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 (3.5)
	Variables	Open the table of project variables for editing
Service		
		
	Simulation	Start the simulation mode
	Execution order	Change the execution order for the outputs or delay lines in 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		
		
	Comment field	Text field for comments (7.2)
	Variable output block	Variable, which value can be written in the program (7.3)
	Variable input block	Variable, which value can be read in the program (7.3)
	Constant block	Constant value (7.4)
	Delay line	Feedback with one-cycle delay (7.5)
	Network variable output block	Variable, which value can be written via network (7.6)
	Network variable input block	Variable, which value can be read via network (7.6)
	Block WriteToFB	Connects the input value of the block to the selected parameter of the selected function block and used to change the parameter (7.7)
	Block ReadFromFB	Connects the output value of the block to the selected parameter of the selected function block and used to read the parameter (7.7)
	Conversion to BOOL	Conversion of any value to a Boolean (7.8)
	Conversion to INT	Conversion of any value to an Integer (7.8)
	Conversion to REAL	Conversion of any value to a Real (7.8)

2.3 Workspace

When a project is opened, the workspace with the tab ***Main program*** is shown in the middle part of the window (Fig. 2.7).

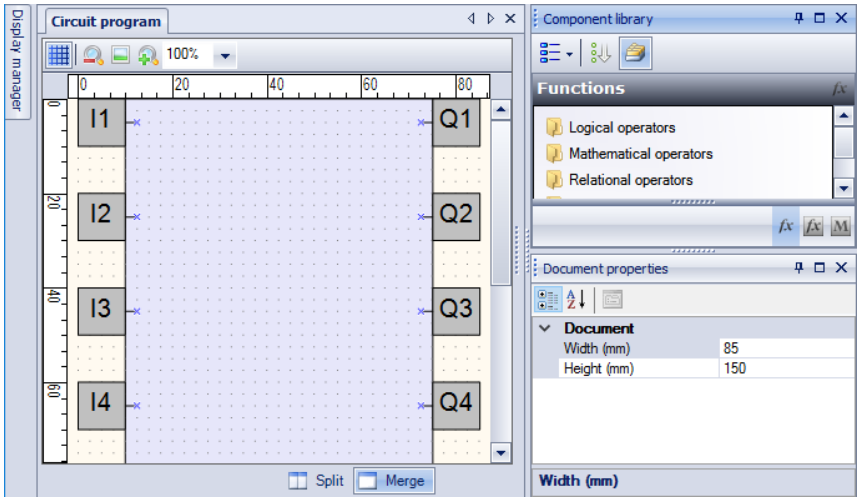


Fig. 2.7 Workspace

Circuit program is built and modified by placing program elements and links 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 inputs and outputs of the target device. Inputs and outputs can be moved up and down along the workspace by drag-and-drop.

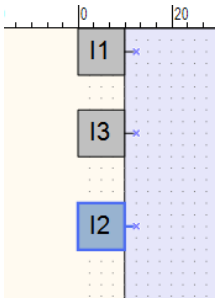
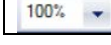


Fig. 2.8

The following icons are located on a toolbar above the workspace (Table 2.7):

Table 2.7

	Show / hide grid	Show / hide vertical and horizontal rulers and a grid in the workspace. If visible, the elements and connecting lines are snapped to the grid.
	Zoom -	Increase the workspace by 10%
	Original size	Return to the original size (100%)
	Zoom +	Decrease the workspace by 10%
	Select scale	Scale list from 20% to 400%

The icons  ***Split*** and  ***Merge*** are located on a toolbar below the workspace. Use the icon ***Split*** 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 ***Merge*** to return to one workspace.

2.4 Library Box

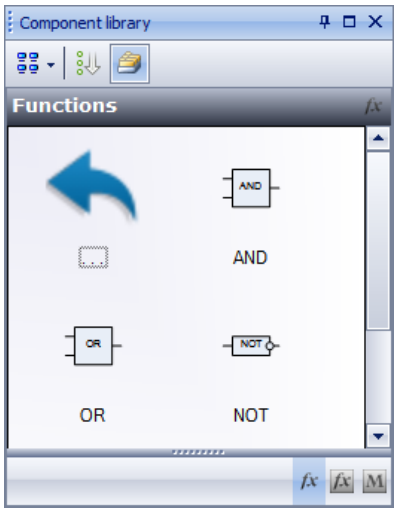


Fig. 2.9 Grouped tiles

The panel Library Box is a summary of program elements available in the project. The panel consists of three libraries: ***Functions***, ***Function blocks*** and ***Project macros.***

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 elements of the selected library (Fig. 2.10):

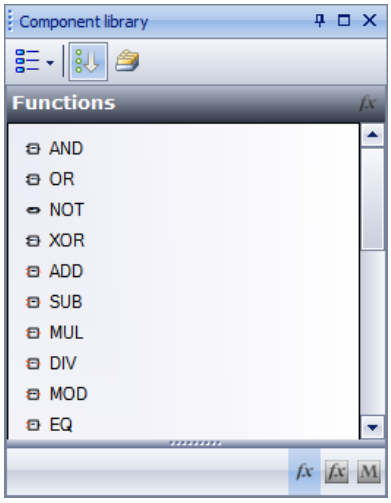


Fig. 2.10 Show all

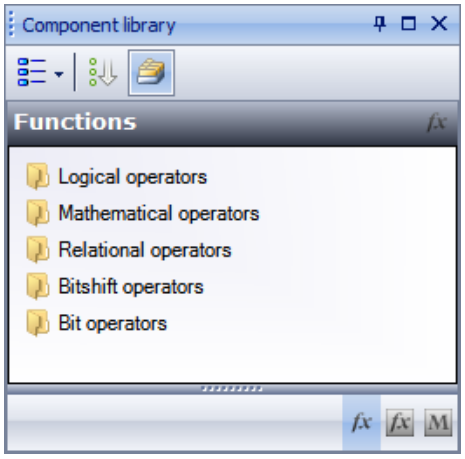


Fig. 2.11 Show grouped

Click the icon ***Show grouped*** to show the elements of the selected library grouped (Fig. 2.11). Double-click the folder to open it.

For descriptions of the library groups and its elements see section 6.

2.5 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.

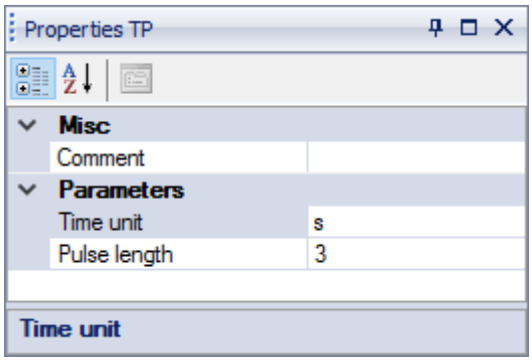


Fig. 2.12

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.

2.6 Display Manager

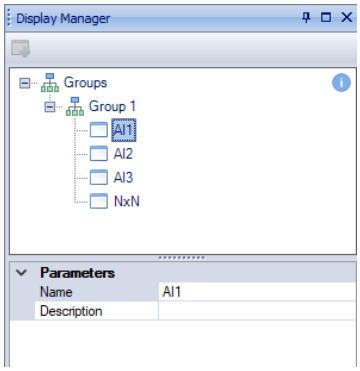


Fig. 2.13

If the target device has a display, the displayed information can be customized using Display Manager. The display can be programmed using one or more display forms with “jumps” between them so that the operator can switch between display forms to see the desired information. For further details about display programming see section 8 “Display programming”.

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 display form hierarchical structure (tree) and a list of properties of the selected object.

The parameters of the selected display form are shown in the lower part of the panel.

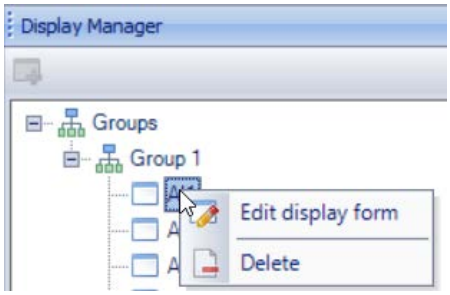


Fig. 2.15

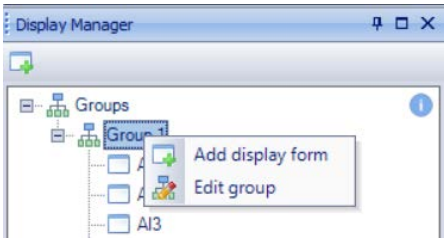


Fig. 2.14

To open a graphical structure of display forms in a separate workspace ***Structure Editor*** (see 2.6.1), use the command “Edit group” in the group context menu (Fig. 2.14).

To open the selected form in a separate workspace (***Display Editor***, see 2.6.2), use the context menu or double-click it in the group (Fig. 2.15).

2.6.1 Display editor

The workspace shows the selected display form with the icons on the right of it, which are used to change the number of the displayed rows. The rows displayed first are bold outlined.

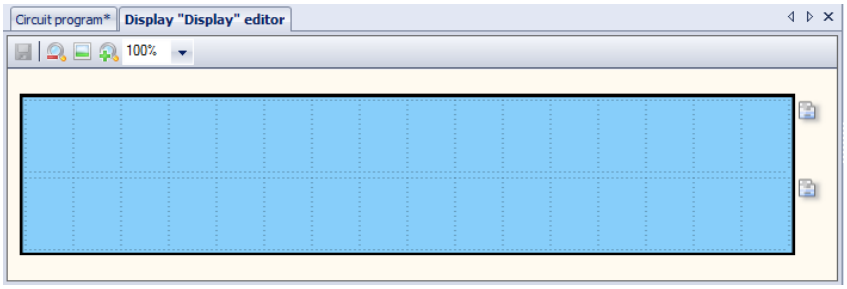


Fig. 2.16

Display Editor has the same toolbar as Structure Editor except the ***Add*** and ***Delete*** items. For use of Display Editor see 8.1.

2.6.2 Structure Editor

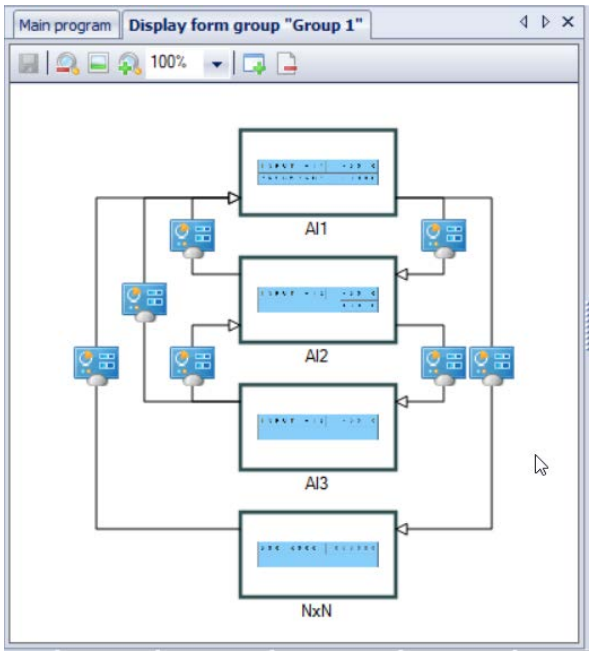


Fig. 2.17

The graphical structure of display forms with “jumps” and their conditions can be edited in Structure Editor.

Table 2.8 Structure Editor toolbar

Common	
	Save workspace
Zoom	
	Zoom out by 10%
	Original size
	Zoom in by 10%
	Select scale
Display form	
	Add display form
	Delete display form

For use of Structure Editor see 8.2

2.7 Status bar

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

Besides that the status bar contains 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. Which indicators are available in the status bar, depends on the type of the target device.

Fig. 2.18

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

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

- ***FB*** – the number of the available and used function blocks
- ***Var*** – the number of the available and used variables

Note: *Some variables can be created by the software automatically, if such elements as delay lines or multiple links with common nodes are used in a project.*

- ***Stack*** – the number of the available and used stack levels
- ***EEPROM*** – the available and used retain memory
- ***ROM*** – the available and used ROM memory
- ***RAM*** – the available and used RAM memory

  Note: *ALP software automatically calculates the available resources of the device and shows a warning if the critical value is reached.*
- ***Device*** – the type of the connected device

  Note: *Click the indicator to switch to* ***OFFLINE*** *mode. In this mode the connection to device is interrupted. The next click deactivate* ***OFFLINE*** *mode (see 3.4.1).*
- ***Port*** – the selected COM port number (programming interface).

  Note: *Click on the indicator to open the window* ***Port settings****.*

2.8 Variable Box

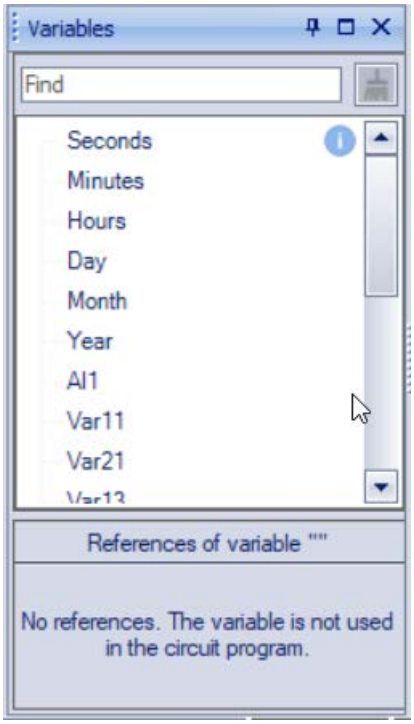


Fig. 2.19

The panel Variable Box shows the project variables from the table of variables. It can be called with the menu item ***Device > Variable Box*** .

The standard position of the panel in the upper left window corner and can be changed.

You can view the information about the variable as a ToolTip text.

The references of the variable in the project 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 on the workspace.

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

To use a variable as an output block, hold the Shift key pressed while drag-and-drop the variable.

If the variable is drag-and-dropped on a connection pin of a program element, a variable block will be created attached to this connection pin.

If the 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 (Fig. 2.20).

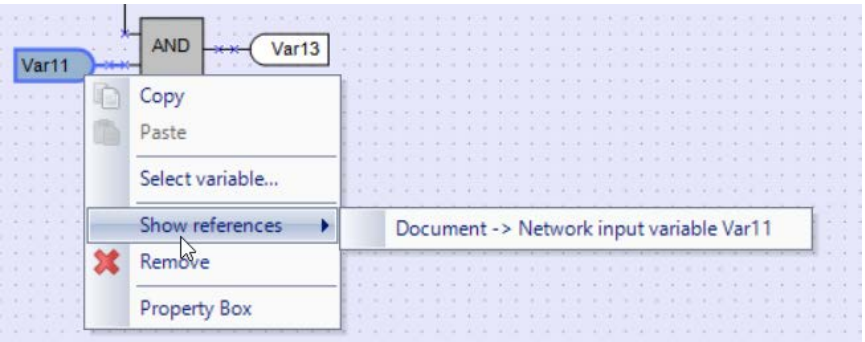


Fig. 2.20

2.9 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. Select the menu item ***File > Component Manager*** to open it in a separate window.

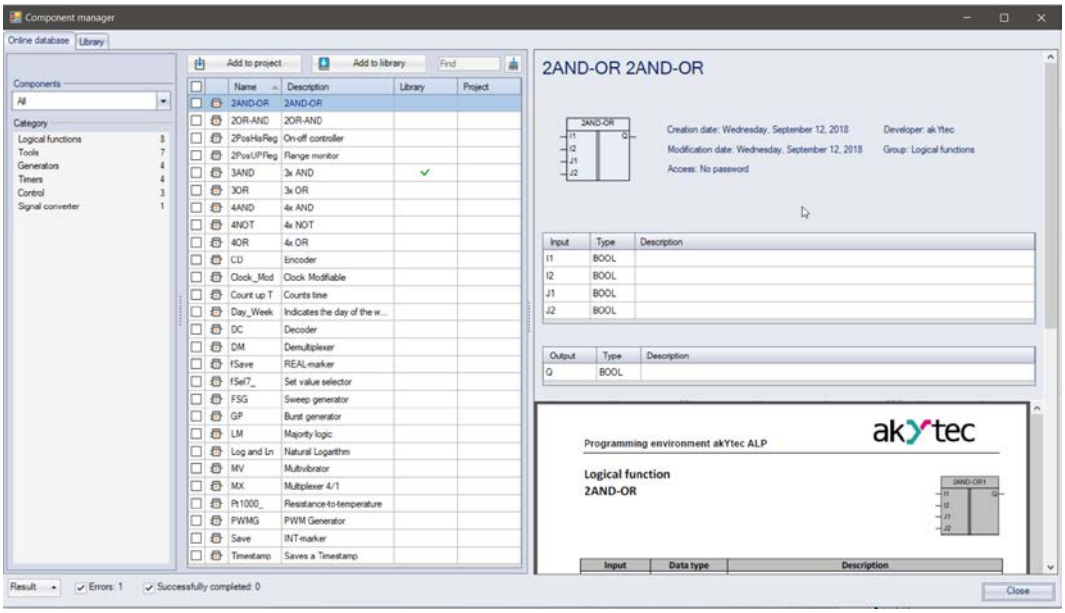


Fig. 2.21

The internet access is necessary for Component Manager to interact with Online Database.

The interface has two tabs: ***Online Database*** and ***Local library***. Elements are shown in categories and can be filtered by name.

- Select the required elements.
- Click the button ***Add to library*** to download the selected elements (macros or device templates) from Online Database to a local library for offline using.
- Click the button ***Add to project*** to add the selected elements from Online Database or from the local library to the project library. The elements are than stored in the project file and can be viewed in Library Box.

A green check mark appears in the column Library or Project depending on the button clicked. This indicates that the element was successfully downloaded / added.

Library files are located at:

***C: \ Users \ [username] \ Documents \ akYtec ALP \ Library ***

A brief description of the selected item is displayed in the upper right field, and a full description in the lower right field. The full description is a PDF document. Scroll the docu-

ment to the end to see the PDF toolbar. Using it, you can download the document or print it.

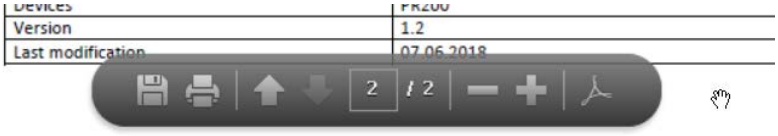


Fig. 2.22

Click the button ***Operation result*** about at the bottom of the window to view the program messages about the performed operations.

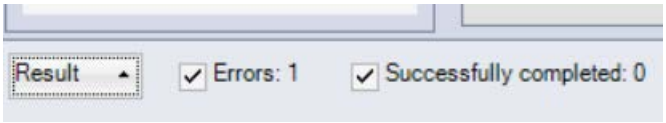


Fig. 2.23

3 Usage basics

To program the device proceed as follows:

- Start akYtec ALP
- Create a new project for the selected target device or open an existing project (see 3.3)
- Save the project on the PC
- Debug the project in the simulation mode (7.11)
- Upload the project to the target device (3.7)

3.1 Program execution

When the target device is selected, the number of available inputs, outputs and the real-time clock availability is already determined. The general structure of the programmable relay is shown in Fig. 3.1.

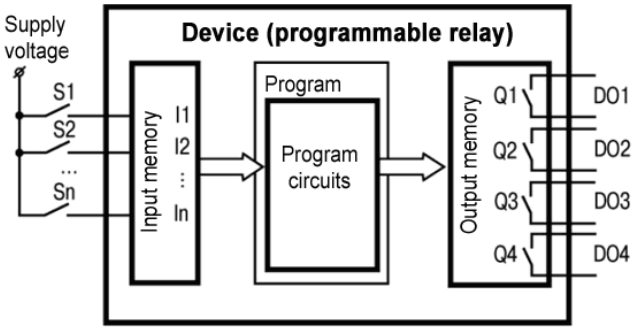


Fig. 3.1 PR operation flowchart

A programmable relay is a kind of PLC with a cyclically executed program:

Step 1 – The status of physical input points is saved to the input memory cells (Input Image Table).

Step 2 – The values from the input memory cells are read and the program is executed from its first instruction to the last one.

Step 3 – The results are saved to the output memory cells (Output Image Table) and applied to the outputs.

When the last step is completed, the program runs again from the first step.

3.2 Cycle time

The time of the operating cycle depends on circuit program complexity. Especially resource-intensive are the following elements:

- FBs (6.2)
- macros (6.3)
- network variables (5.5)
- display elements (6.4).

The cycle time is calculated by the device. It can be viewed on the device display (if exists) using the system menu (see user guide). It can also be viewed in akYtec ALP in simulation mode (see 7.11). The minimum (default) cycle time is 1 ms.

Fig. 3.2

3.3 Project creation

To create a new project select ***File > New project*** in the main menu or use the equivalent icon in the taskbar. Select the target device in the dialog window ***Device selection*** and confirm it with ***OK*** (Fig. 3.3).

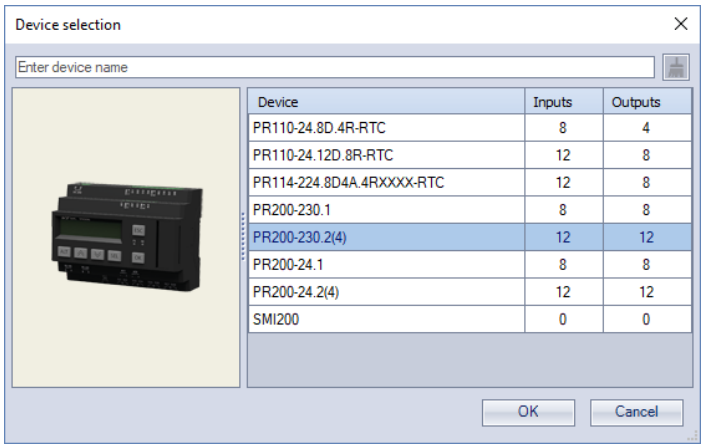


Fig. 3.3

When a new project is created, the workspace with an empty circuit program appears, the status bar is filled with the information about available resources, the panel ***Library Box*** shows the available program elements and the panel ***Property Box*** shows the workspace properties.

Circuit program

Now the main circuit program for the project can be created in the workspace using the common program elements from the toolbar ***Insert*** and the specific program elements from ***Library Box***. Draw connecting lines between inputs, outputs, and program elements to establish logical connections in the program. For details about each program element and connecting lines see section 7.

Display Manager

If the selected device has a display, the ***Display Manager*** tab appears to the left from the workspace. With this tool you can customize the displayed information.

Simulation

Program can be simulated only offline using the menu or toolbar ***Service***. Start the simulation mode, change the state of the inputs and notice the state of the outputs to check the correctness of the program.

3.4 Connection to device

▶ **NOTICE** ***The device must be powered off before connecting to PC.***

The device can be connected to PC directly (PR200) or over the adapter PR-KP20 (for PR110, PR114). The required connection cables are in the package of the PR200 or the adapter included.

Connect the device to a USB port of the PC, switch the power on and select the serial port in the menu ***Device > Port settings***. The number of the emulated COM port can be found in the Windows Device Manager under “Connections (COM and LPT)”.

If the operating system does not find the correct driver, install the driver for PR200 or for the adapter PR-KP20. It can be downloaded from akytec.de.

Select the port number in the dialog window ***Port settings*** and confirm with ***OK***. The only setting available for the serial port is the COM port number, all other settings are fixed and displayed only for your information.

If the connection is established, the information about the connected device and the serial port is shown in the status indicators.

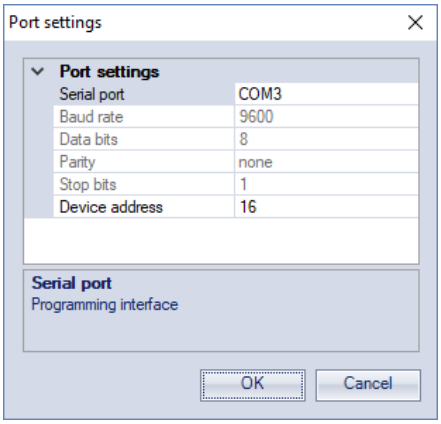


Fig. 3.4

3.4.1 OFFLINE mode

In this mode, the connection between akYtec 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 interrupted.

The application that should not communicate with the device should be switched to OFFLINE mode.

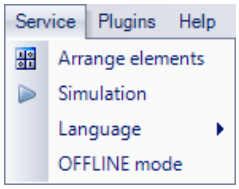


Fig. 3.5

OFFLINE mode can be activated / deactivated using the menu item ***Service > OFFLINE*** mode or by clicking the status indicator ***Device*** (see 2.7). With the next click is OFFLINE mode deactivated.

3.5 Device information

To view information about the software, the target device and the connected device use the menu item ***Device > Information*** or the icon on the toolbar.

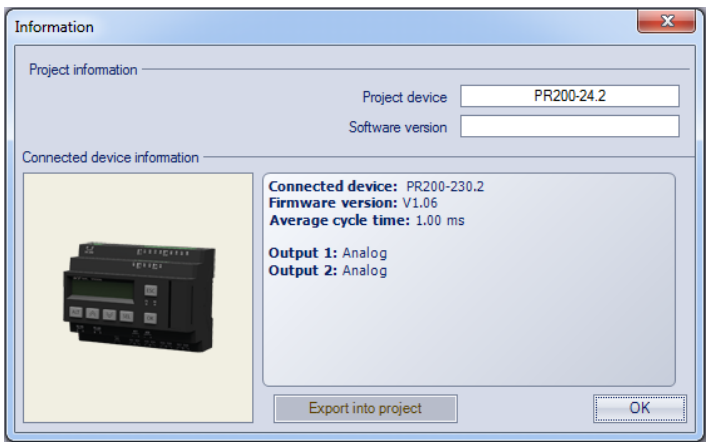


Fig. 3.6

The window ***Device Information*** contains the following information:

Target device – the device selected when the project was created

Software version at project creation – the version of the software in which the project has been created, this version may differ from the current software version

Software version at project modification – the version of the software in which the project has been modified, this version may differ from the current software version

Connected device information – the information about the device connected to the PC

Export into project – the button is active only if the device PR114-224.8D4A.4RXXXX is connected. It can be used to export the real output types of the connected device into the project. Alternatively the type of each output can be manually changed in ***Property Box*** in accordance with the hardware.

3.6 Project information

Use the menu item ***File > Project*** information to view and modify the information about the project. The tab ***General*** contains read-only information about the version of the software in which the project is created and modified.

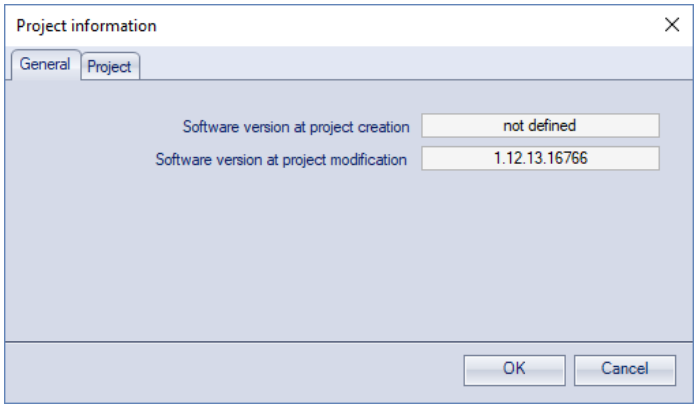


Fig. 3.7

The tab ***Project*** contains information about project version and is only for PR200 available.

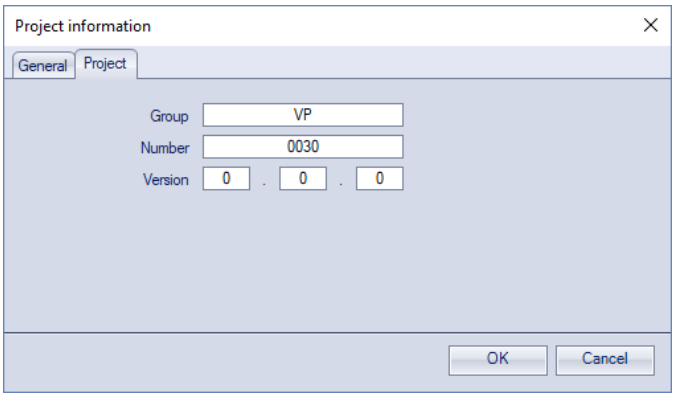


Fig. 3.8

After you complete the desired entries confirm your entries by clicking ***OK***, or click ***Cancel*** to discard them.

3.7 Upload project to device

Attention! The program already stored in the connected device will be replaced by the new one.

Proceed as follows:

– connect the device to PC

- power on the device
- adjust the port settings if necessary
- upload the project to the device

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

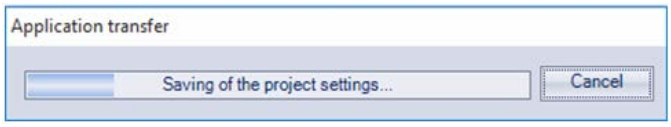


Fig. 3.9

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

Note: After the program transfer is completed, the device goes to the operating mode and the program starts automatically.

3.8 Firmware update / repair

If a new akYtec ALP version includes a new version of the firmware for the connected device or extension module, you will be prompted 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 manual, table “Error indication”).

Note: The user program will not be affected by firmware update.

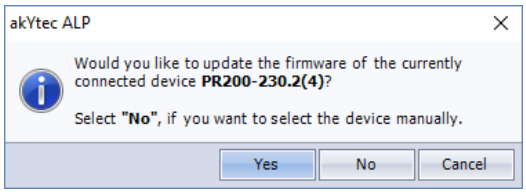


Fig. 3.10

If you select ***Yes***, the firmware of the currently connected and automatically recognized device will be updated (repaired).

If you select ***No***, lists of devices and extension modules will be offered to select from (Fig. 3.10). The window has two tabs: ***Device*** and ***Extension Module***. This way a forced firmware update can be made.

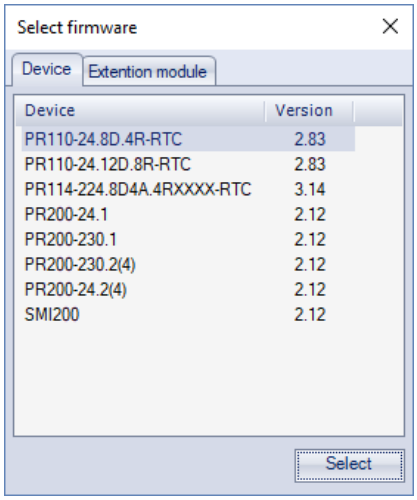


Fig. 3.11

Click ***Select*** to confirm the selection and start the update (repair) process. The message about the update result is shown upon the update completion.

Forced firmware update / repair

If the firmware is damaged (see respective user manual, 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 a 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 base unit and the extension module have incompatible firmware versions and the user program is uploaded to the base unit without the extension module connected, this may lead to an expansion module error displayed. To fix the error, use forced firmware update for the expansion module as described, skipping step 1).

4 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 can include the following groups:

- Display
- Clock
- Interfaces
- Inputs
- Outputs

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

4.1 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 – display brightness 0 100%

Contrast – display contrast 0 100%

The button ***Read*** can be used to read out the current display settings from the connected device.

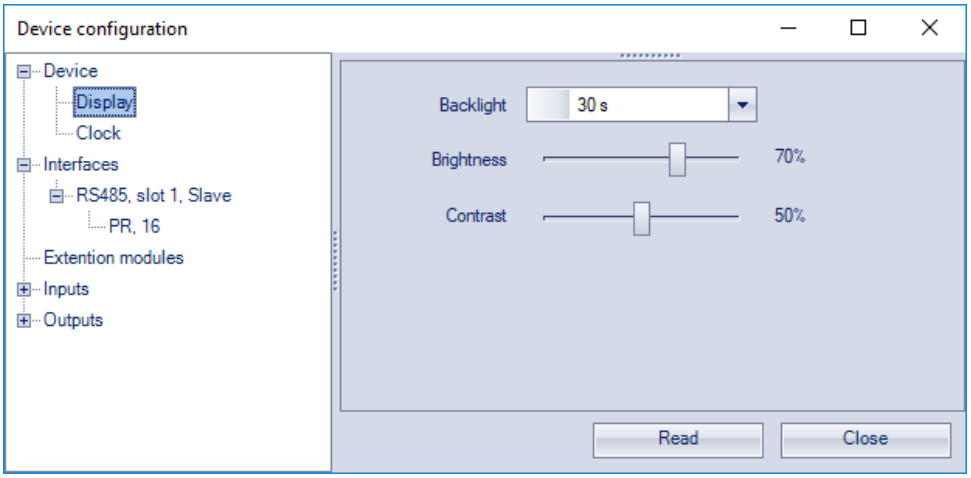


Fig. 4.1

4.2 Clock

The date and the time can be set in the group ***Clock*** if the target device has an integrated real-time clock.

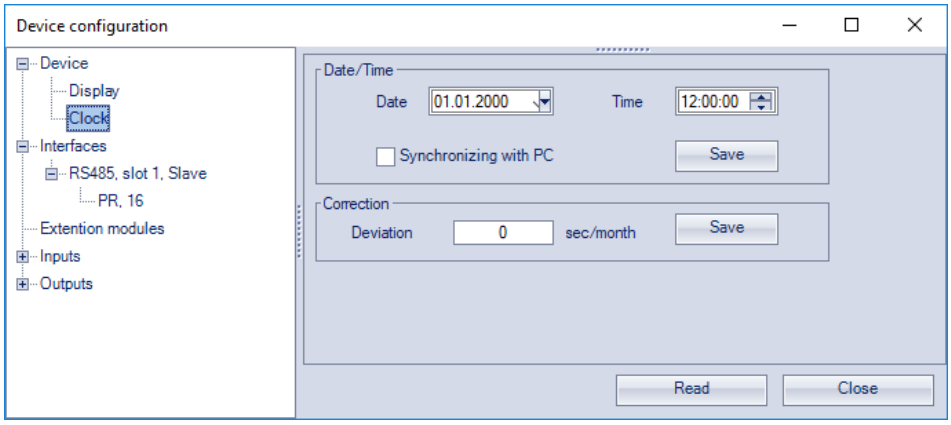


Fig. 4.2

To synchronize the device clock with the PC clock, check the checkbox ***Synchronizing 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***.

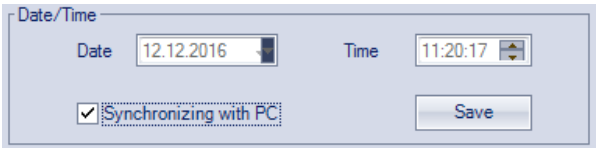


Fig. 4.3

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.

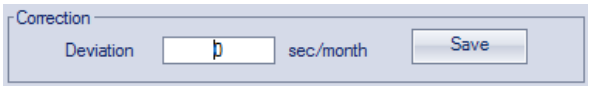


Fig. 4.4

To set the device clock correction, click the button ***Save*** in the section ***Correction***.

The button ***Read*** can be used to read the current time settings from the connected device.

4.3 Interfaces

If the target device has a serial network interface RS485, its parameters can be set in the group ***Interfaces***.

By default, there is one interface configured as a Slave and assigned to the hardware slot 1 with default settings: Master device with the name PR and the network address 16.

If the number of interfaces on the target device can be changed, interfaces can be added or deleted in the configuration, but their number cannot exceed the number of the existing slots (see 4.3.2).

If an interface is configured as a Master, Slaves can be added or deleted in the configuration, but their number cannot exceed 16.

4.3.1 Modbus working

akYtec ALP provides programming of devices that support Modbus-RTU (Master / Slave) or Modbus-ASCII (Master / Slave) protocols.

Note: The devices PR110 and PR114 can operate only in the Slave mode and only with the module PR-MI485 connected.

In order to organize data exchange in the network over the RS485 interface, a Master device is required. There can be only one Master in the network.

Cycle time

The program execution cycle time of the device is automatically adjusted (auto-tuning) depending on the program complexity. Cycle time auto-tuning affects data exchange over Modbus, since the user program execution has a higher priority than request processing. If the program is large, it can take up all the CPU time and Modbus data exchange will not be performed correctly.

To avoid this problem, the lower limit for the volume of the Modbus data exchange is reserved: 50 requests per second. Thus, at least 50 requests per second can be executed even if the user program is large, and even more if the program is small and the processor capacity is sufficient. If there is not enough time to poll all devices, the number of requests should be optimized in the user program.

The ***query cycle*** setting depends on the number of polled variables and the polling frequency in the program. It is recommended to set the ***query cycle*** to 1 s. In this case, the device will be able to request up to 50 variables.

Request time

The device puts each request in the queue. If the queue is short, the device will perform all the request-response cycles and wait for the specified ***query cycle*** time to expire (Fig. 4.5). If the queue is long and exceeds the specified time, the device will poll all slaves with the shortest possible time, but it takes longer than the specified ***query cycle***.

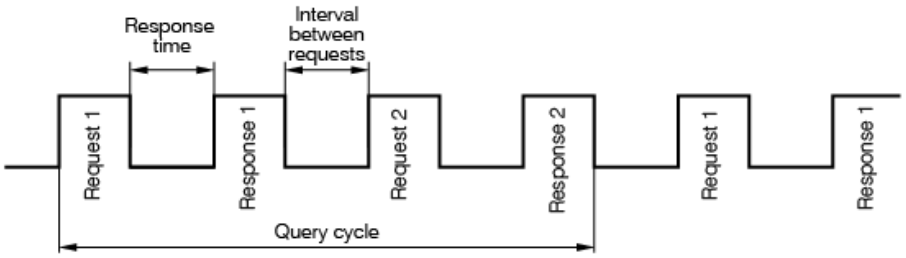


Fig. 4.5 Query time diagram

To minimize the request time, the following is recommended:

- If one or several Slave devices are not connected or not available, it is recommended to foresee blocking of polling of these devices in a program or to minimize the ***Timeout*** parameter for these devices.
- When you set the ***Query cycle*** parameter, the number of Slave devices and the total number of requests should be taken into account. If the processing time of all requests takes longer than the set time, the parameter will be ignored.

Polling of multiple devices on the bus

Slaves are polled according to the generated queue runs from the smallest address to the largest one. In the example below, first the slave with the address 8 will be polled, last - with the address 32.

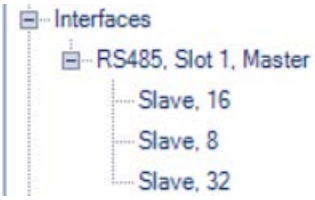


Fig. 4.6

You can set ***Query cycle*** for each slave device individually.

4.3.2 Add / remove interface

If the device has a slot, for which no interface is configured, an appropriate interface can be added using the item ***Add Interface*** in the context menu.

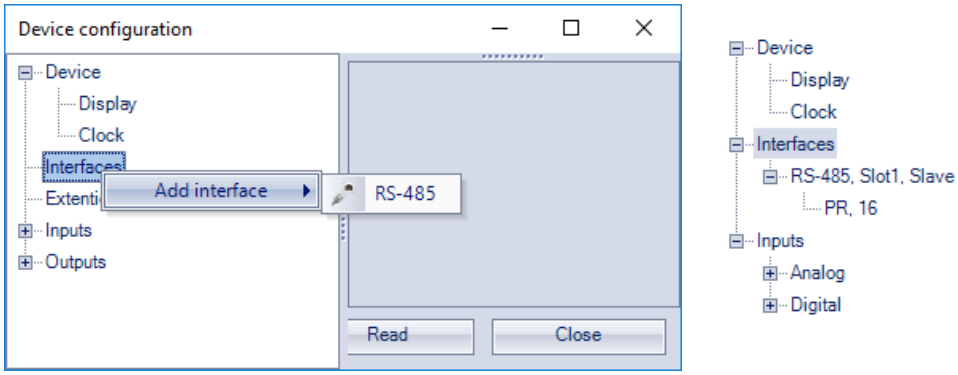


Fig. 4.7

An interface of the selected type with default settings is added.

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

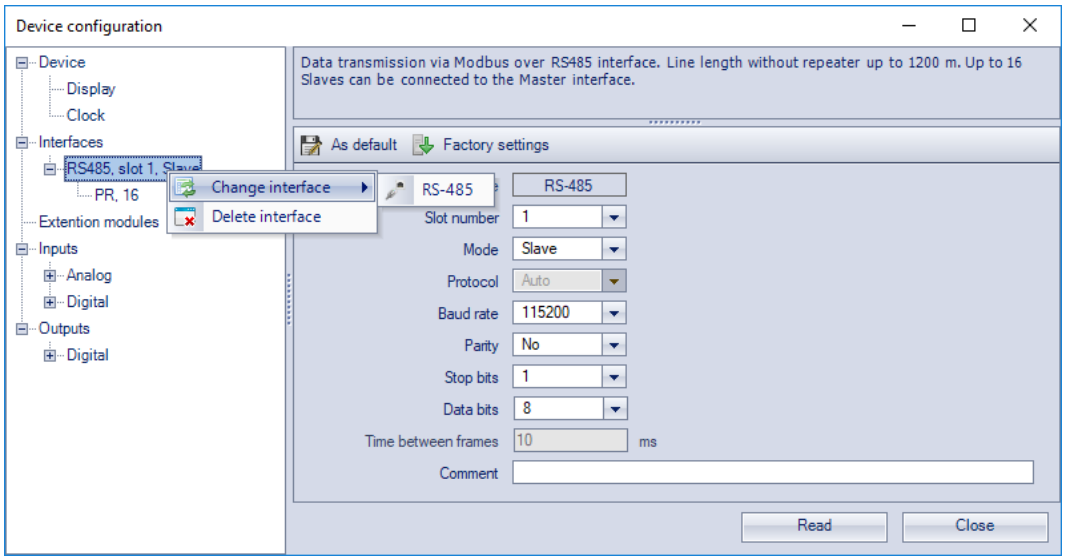


Fig. 4.8

4.3.3 RS485 Interface configuration

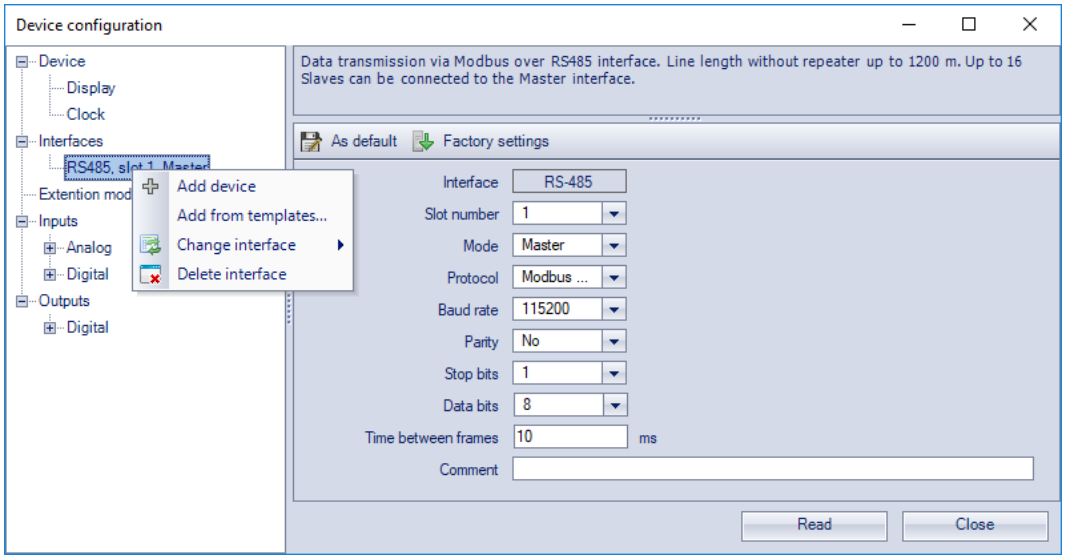


Fig. 4.9 Interface configuration

The type of the interface (RS485), the number of the assigned slot and the mode (Master/Slave) are displayed in the tree.

To establish the connection over the interface, it has to be configured. The parameters of the interface are displayed on the right part of the window. The default value depends on the target device. The parameters ***Protocol*** and ***Interval between requests*** are only in the Master mode available. In the Slave mode they are inactive and grayed out.

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

The icon ***Factory settings*** is used to apply the unchangeable factory settings.

The button ***Read*** is used to read out the current settings from the connected device.

Use the button ***Close*** to save the settings in the project and close the dialog.

4.3.3.1 Master mode

Each interface can control up to 16 slaves. Each slave supports up to 256 variables. The variable addresses and variable names must be unique only if they belong to the same slave.

In the Master mode all the Slaves connected to the interface are sequentially requested. Select the mode ***Master*** in the parameter list, set other connection parameters and add the required number of Slaves using the item ***Add slave*** in the interface context menu.

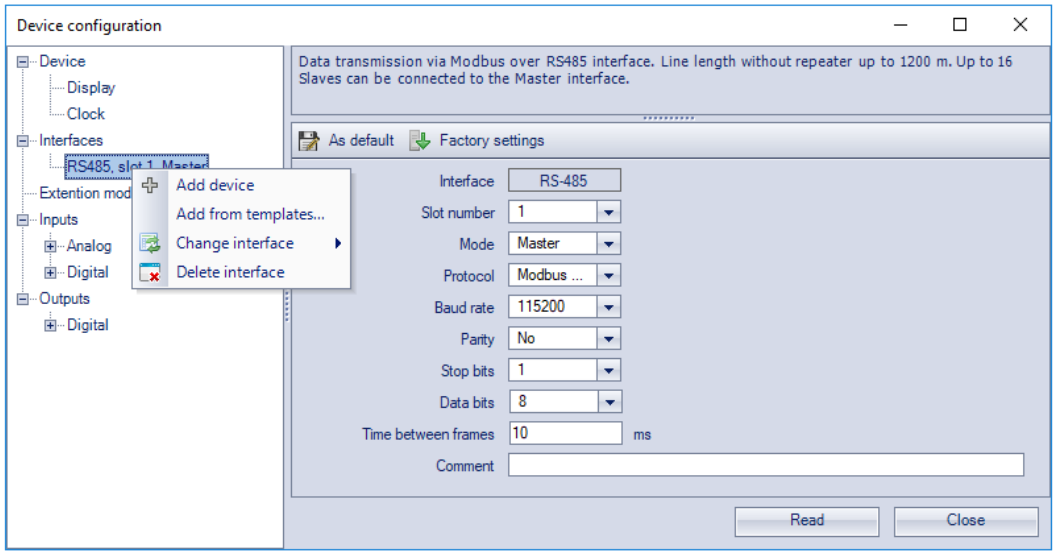


Fig. 4.10 Master configuration in master mode

The added Slave device is displayed with its name and address in the tree below the interface. Select the Slave and configure it in the right part of the window.

In the upper area the parameters of the Slave can be set (Fig. 4.11).

To delete the Slave device use the context menu (Fig. 4.10) or click the icon ***Remove Slave***.

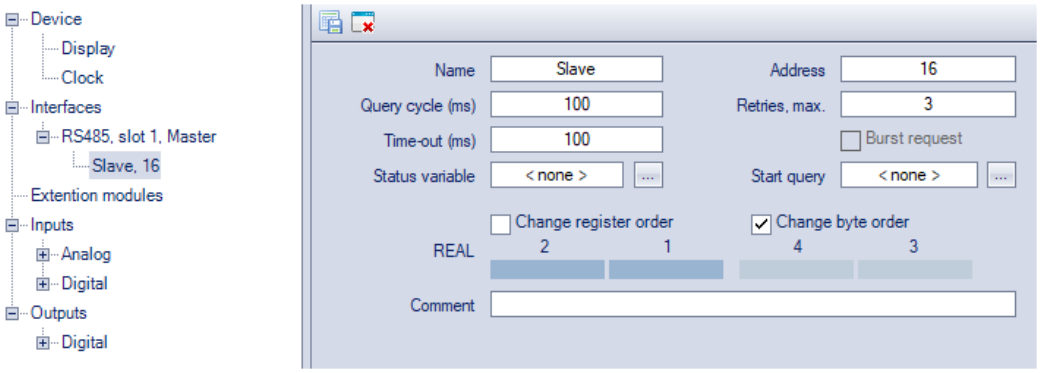


Fig. 4.11 Slave configuration in master mode

- ***Name*** – the name of the Slave displayed in the tree
- ***Address*** – the network address of the Slave
- ***Query cycle (ms)*** – the time interval between queries. A query comprises the number of requests according to the number of variables listed for the Slave. The valid range is 0 65535 ms.
- ***Timeout (ms)*** – the time that request can take before the attempt is considered as failed. The valid range is 0 65535 ms.
- ***Retries, max.*** – the number of the failed request attempts, before query is stopped and the status of the device changes. The valid range is 0 255.
- ***Burst request*** – group request of consecutive registers to increase the data throughput
- ***Status variable*** – the Boolean variable used to record the device status:
  - 1 – the device functions properly
  - 0 – connection with the device is lost. Use the icon ... to select a variable.

- ***Start query*** – the Boolean variable to control the query:
  - 0 – query disabled
  - 1 – query enabled. Use the icon ... to select a variable.
- ***Change register order*** – determines the register order in two-register variables
- ***Change byte order*** – determines the byte order in the register
- ***Comment*** – the text comment to describe the device

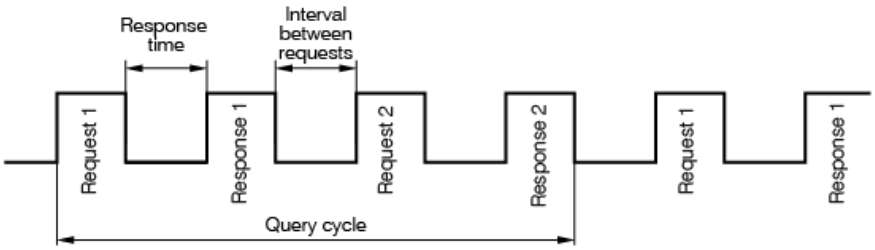


Fig. 4.12 Query time diagram

The list of the variables to be requested from this Slave is in the lower area. Each variable created in this list can be found in the summary table of variables under the tab ***RS485, Slot X*** with a separate list of variables (see 5.5) for each slave device.

Add a variable by clicking the icon ***Add variable*** and set the variable parameters.

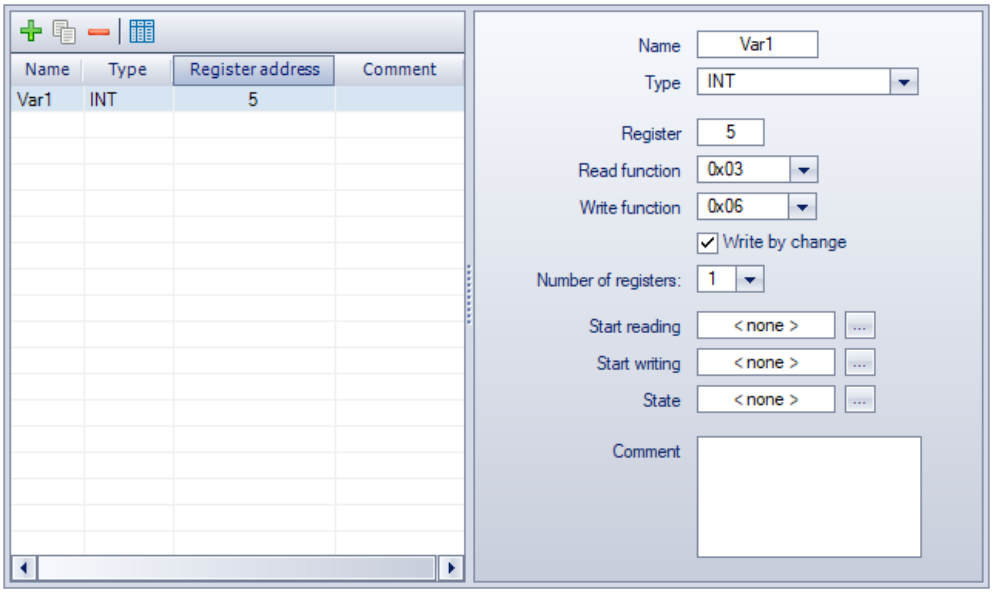


Fig. 4.13 Variable configuration in master mode

- ***Name*** – the name of the variable
- ***Type*** – the data type of the variable: Boolean, Integer or Real
- ***Register*** – the register address
- ***Bit*** – the number of the bit of the register (0 15) (only for Boolean variables)
- ***Read function / Write function*** – selection of the read / write function or disable reading / writing.
- ***Number of registers*** – the number of registers occupied by the variable (only for integer variables)
- ***Start reading*** – the Boolean variable assigned for forced reading of the requested variable

- ***Start writing*** – the Boolean variable assigned for forced writing of the requested variable
- ***Status variable*** – the integer variable used to record the error code in case of error
- ***Comment*** – the text comment to describe the requested variable

To create several variables with similar settings, select a variable and click the icon ***Replication***.

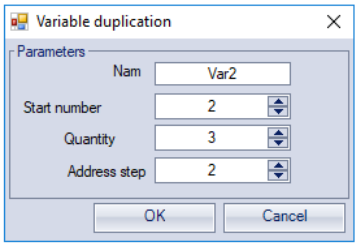


Fig. 4.14 Variable replication

- ***Name*** – the name of the replicated variable
- ***Start number*** – the initial number added to the name of the replicated variable
- ***Quantity*** – the quantity of the replicated variables
- ***Address step*** – the address increment

Click ***OK*** to add the replicated variables to the list of variables. To remove the variable from the list use the icon ***Delete variable***.

4.3.3.2 Templates

To save the Slave device with its configuration as a template use the context menu (Fig. 4.13) or click the icon ***Save Slave as a template*** (Fig. 4.11). The template file with the extension *.dvtp can be used in further projects.

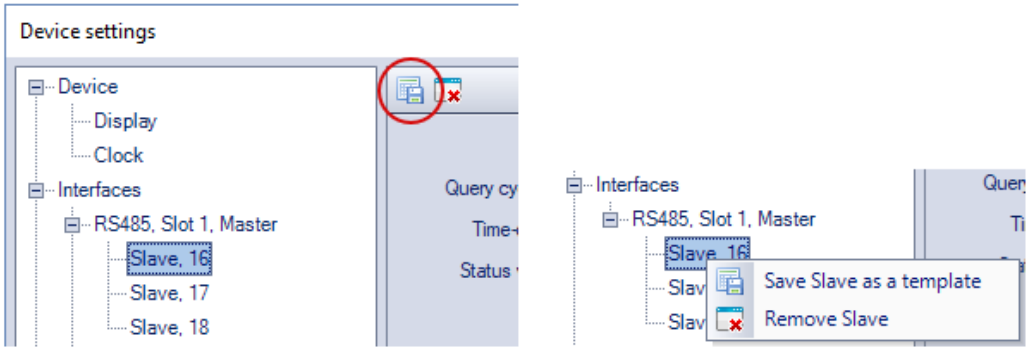


Fig. 4.15 Slave context menu

A Slave as a template can be added to a Master using the command ***Add from templates*** (Fig. 4.16).

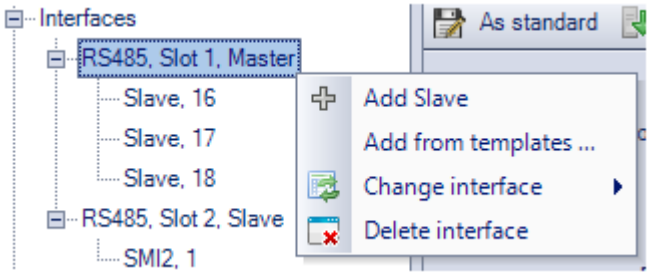


Fig. 4.16 Master context menu

4.3.3.3 Slave mode

If an RS485 interface is added to the tree, the default mode is Slave and the default Master with the name PR and the address 16 is added below. Select the interface to set other connection parameters.

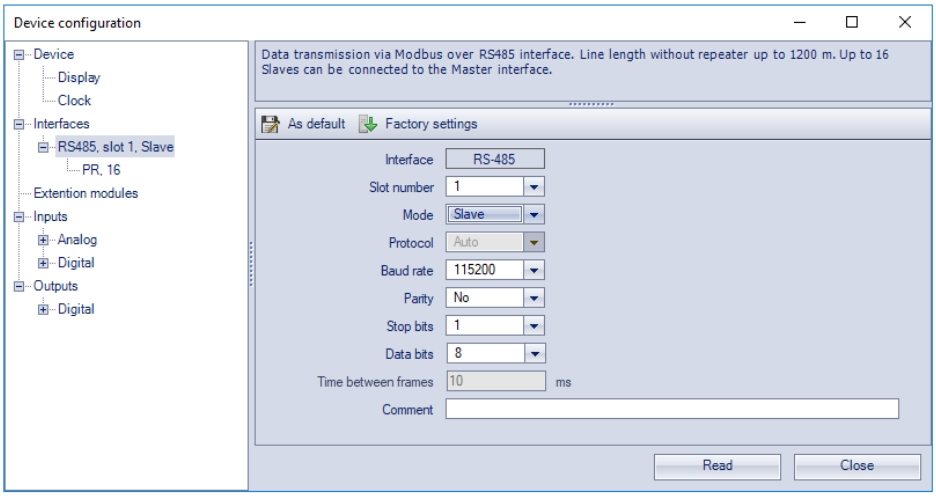


Fig. 4.17 Slave configuration in slave mode

Select the Master to set the parameters for data exchange.

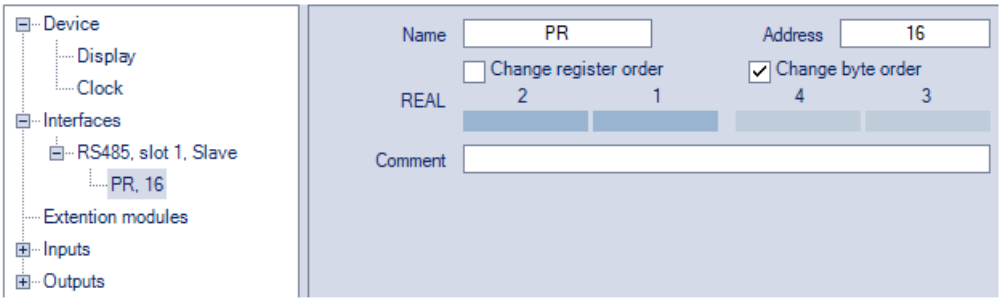


Fig. 4.18 Master configuration in slave mode

The common parameters for data exchange can be set in the upper window part.

- ***Name*** – the name of the Master displayed in the tree
- **Address** – the network address of the Master
- ***Change register order*** – determines the register order in two-register variables
- ***Change byte order*** – determines the byte order in the register
- ***Comment*** – the text comment to describe the device

The list of the variables to be requested by the Master is in the lower part of the window. Each variable created in this list can be found in the summary table of variables under the tab ***RS485, Slot X*** (see 5.5).

Add a variable by clicking the icon ***Add variable*** and set the properties of the variable.

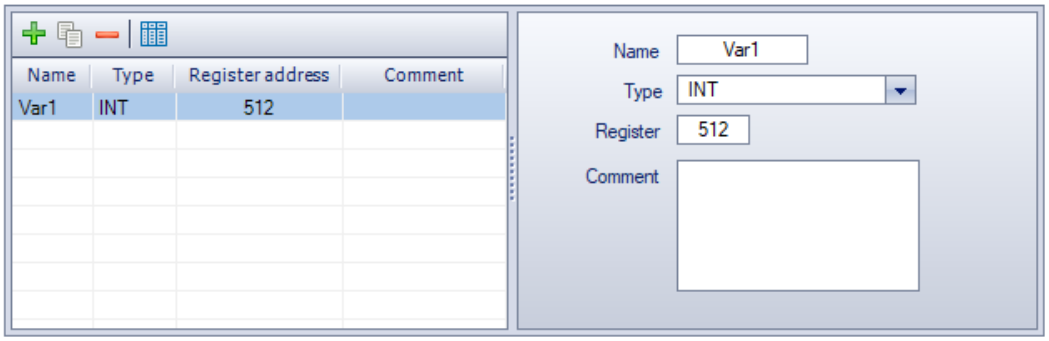


Fig. 4.19 Variable configuration in slave mode

- ***Name*** – the name of the variable
- ***Type*** – the data type of the variable: Boolean, Integer or Real
- ***Register*** – the register address. The range of the available addresses is specified in the user manual of the device.
- ***Comment*** – the text comment to describe the requested variable

To create several variables with similar settings, select a variable and click the icon ***Replication***.

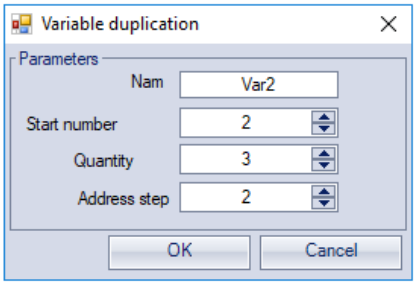


Fig. 4.20 Variable replication

- ***Name*** – the name of the replicated variable
- ***Start number*** – the initial number added to the name of the replicated variable
- ***Quantity*** – the quantity of the replicated variables
- ***Address step*** – the address increment

Click ***OK*** to add the replicated variables to the list of variables. To remove the variable from the list use the icon ***Delete variable***.

4.4 Extension modules

Up to two I/O extension modules of type PRM can be connected to PR200. For further information about extension modules refer to the PRM user manual.
To use module inputs / outputs in the circuit program, add the module to the group ***Extension modules*** using its context menu (Fig. 4.21).

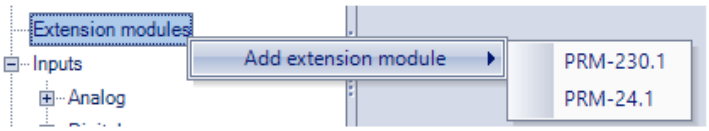


Fig. 4.21

The additional inputs and outputs of the added modules can be configured in branches ***Inputs*** and ***Outputs*** respectively (see 4.5, 4.6). They are displayed in the tree as ***Ix(y)*** or

Qx(y), where ***x*** is the ordinal number of the input (output) on the module and y is the ordinal number of the extension modules counting from the basic device.

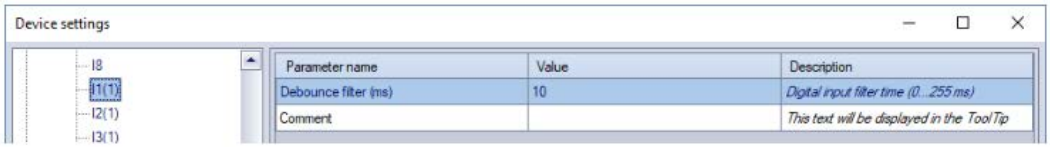


Fig. 4.22

Before uploading the project to the device, all modules must be connected via the internal bus to PR200 and powered. The module firmware is synchronized with the current version of ALP when project uploading.

4.5 Inputs

The content of the branch ***Inputs*** depends on the resources of the target device. It can be analog and/or digital inputs (Fig. 4.23, 4.24).

The parameter ***Comment*** is common for all types of inputs. The text in this field is displayed in the tooltip, when the mouse is over the input in the workspace. It can be entered in Parameter Box too.

For further details about the configuration of the inputs, refer to the user manual of the device.

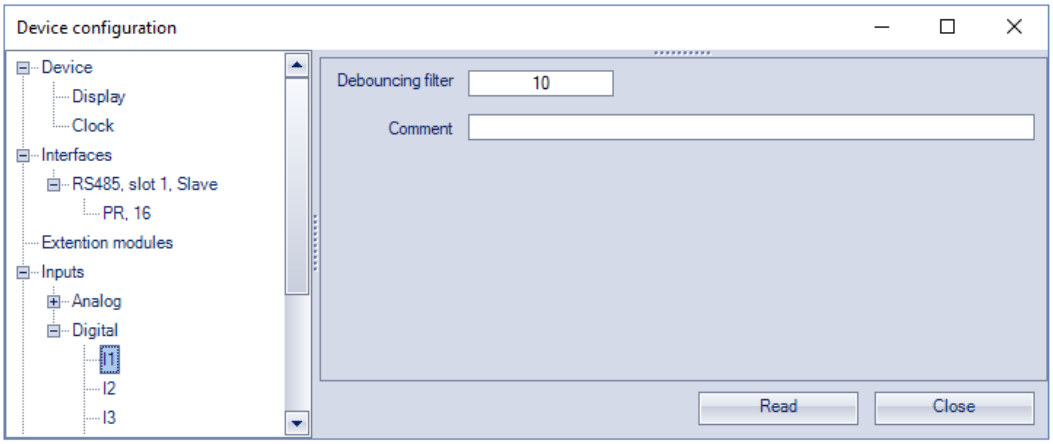


Fig. 4.23 PR200 digital input configuration

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

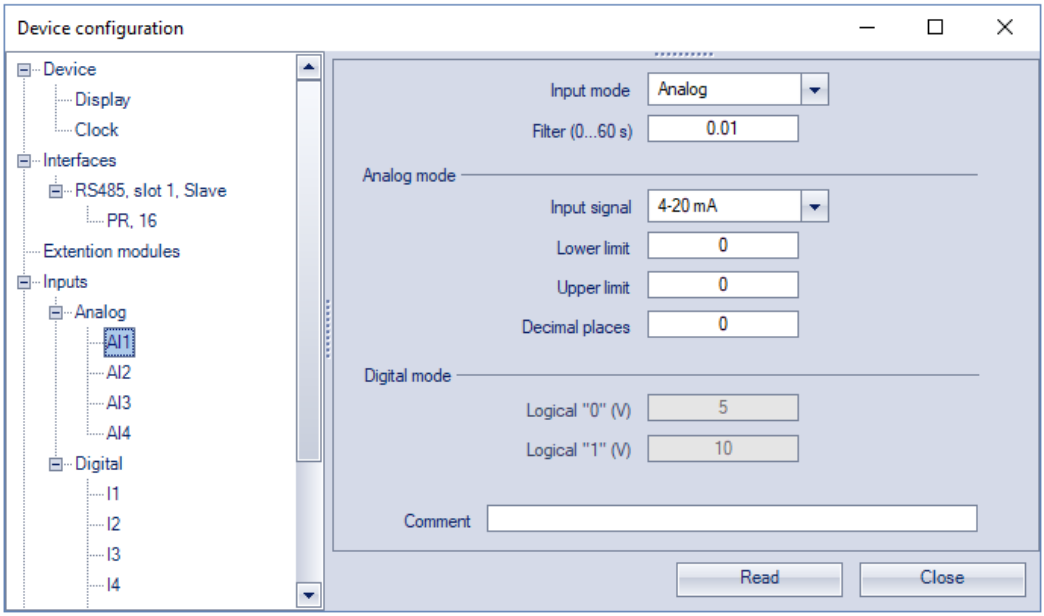


Fig. 4.24 PR200 analog input configuration

4.6 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 the tooltip, when the mouse is over the output in the workspace. It can be entered in Parameter Box too.

For further details about the configuration of the outputs, refer to the user manual of the device.

The digital outputs of the extension module have an add parameter ***Safe condition***. The parameter specifies the output state in case the module loses the connection to the basic device.

Fig. 4.25 PRM output configuration

4.7 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 manual of the respective 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 target to calibrate in the opened dialog.

Fig. 4.26

If the calibration target is selected, the execution of the application on the device is stopped. The application starts again upon the successful completion of the calibration.

4.7.1 Input calibration

Connect a reference signal source to the inputs to be calibrated. Select the type of signal to be connected to the input and the calibration parameters in the opened dialog.

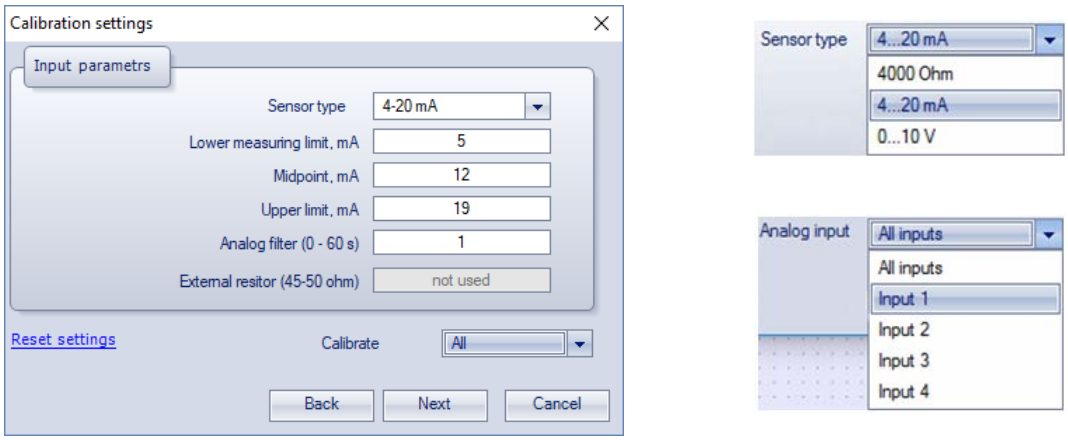


Fig. 4.27 PR200 input calibration

Use the item ***Reset settings*** to apply the default settings for calibration.

Use the list ***Select input*** to select the input to be calibrated, click the button ***Next*** and follow the instructions.

4.7.2 Output calibration

Before calibration of an analog output, prepare the appropriate measuring device and follow the instructions. Measure the value of the signal applied to the output specified in the upper right corner of the window and enter the value into the entry field.

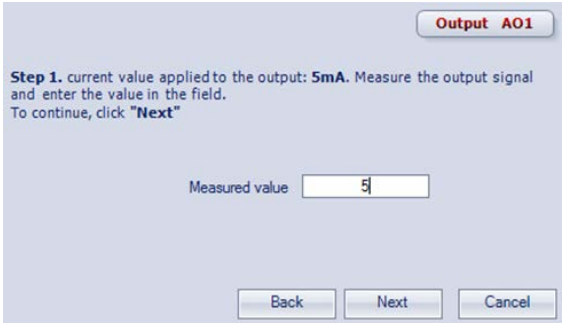


Fig. 4.28 PR200 output calibration

Proceed the same way with the other outputs if needed. The message about the calibration results will appear after the completion of the calibration.

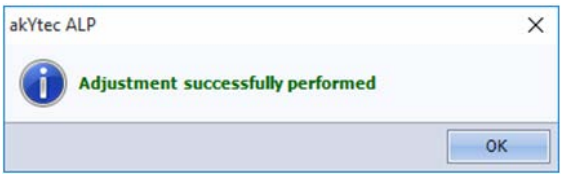


Fig. 4.29

5 Variables

To see all project variables click the icon on the toolbar or use the item ***Device > Variables*** in the main menu.

There are following arts of variables:

- Standard
- Service
- Network

The variables are organized on separate tabs in the table.
Only standard and network variables can be created or deleted.

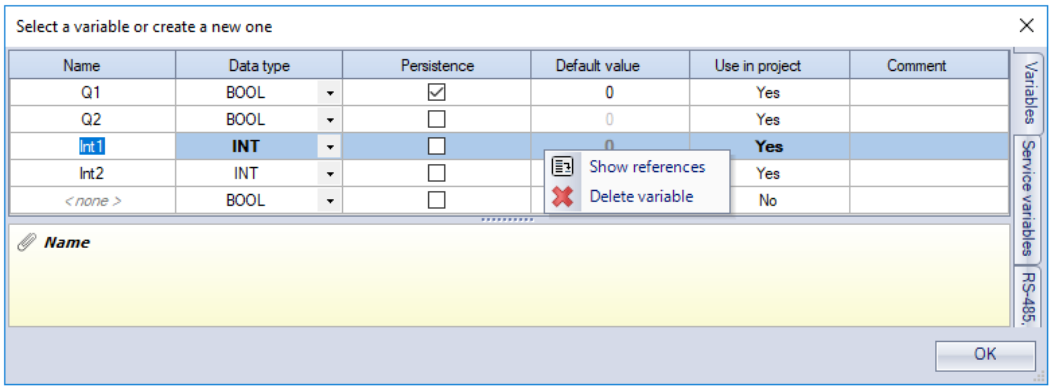


Fig. 5.1

5.1 Properties

Table 5.1 Variable properties

Name	The name of the variable
Data type	Boolean, Integer or Real (see 5.2 “Data type”)
Persistence	Only for standard variables available. The variable is stored in the non-volatile memory of the device and become **retain variable**. For detailed information about storage time and memory size, refer to the user manual of the device.
Default value	Only for retain and network variables available. The value at the fist start of the program, until the new value is assigned to.
Used in project	The variable has a reference to an element in the program
Comment	The text displayed in the tooltip in the workspace, when the mouse is over the variable

5.2 Data type

The variables of the following types can be used in a program:

- Boolean (**BOOL**)
- Integer (**INT**)
- Real (**REAL**)

Note: Different devices can have restrictions related to support of certain types of variables.

1. Boolean (BOOL)

A variable of this type has only two possible values: 1 (True) or 0 (False).

The connecting lines between the Boolean variables in the circuit program are displayed in gray.

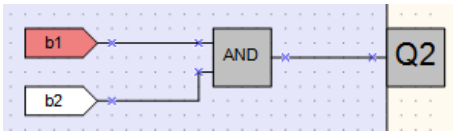


Fig. 5.2 Boolean connections

2. Integer (INT)

A variable of this type has a range 0 4294967295 (4 Byte).

The connecting lines between the Integer variables in the circuit program are displayed in red.

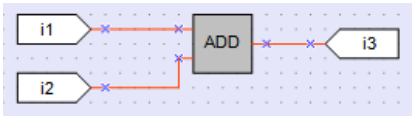


Fig. 5.3 Integer connections

3. Real (REAL)

A variable of this type has a value in the range from ≈1,175e^{-38} to ≈3,4e^{+38}. It is represented by a floating-point number of single-precision (4 Byte).

The connecting lines between the Real variables in the circuit program are displayed in violet.

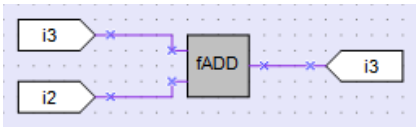


Fig. 5.4 Real connections

5.3 Standard variables

These are common variables used for connection between elements of the circuit program, inputs, outputs and display forms.

Standard variables are listed in the variable table under the tab ***Standard***.

To create a variable select the empty row in the table, enter the variable name and specify its data type. Other parameters are unessential. The created variable can be used in the project.

Use the item ***Show references*** in the variable context menu to see where the variable is used in the project.

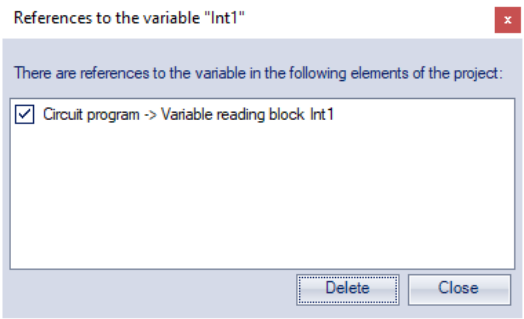


Fig. 5.5

In the dialog window ***References to the variable*** select the reference you want to delete and click ***Delete***.

To remove the variable from the table use the item ***Delete variable*** in its context menu.

5.4 Service variables

Service variables are associated to the device settings and can differ depending on the device. Service variables are related to the hardware features, such as real-time clock, interface card in the slot etc. and cannot be deleted. They can have read/write restrictions.

The service variables are listed in the variable table under the tab ***Service***.

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

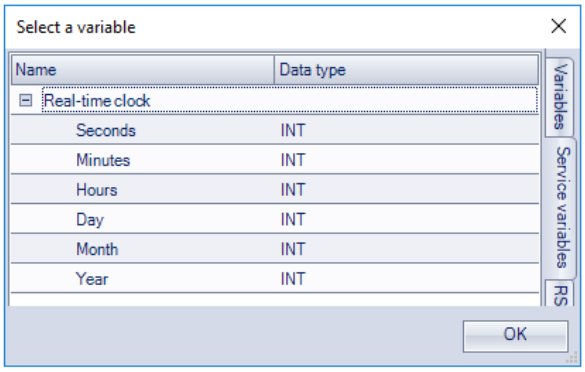


Fig. 5.6 Service variables in the table

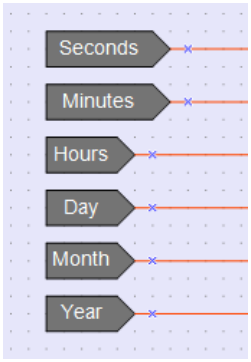


Fig. 5.7 Service variables in a diagram

5.5 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 (Fig. 5.8). The Slave tab contains the variables to be requested for this Slave device.

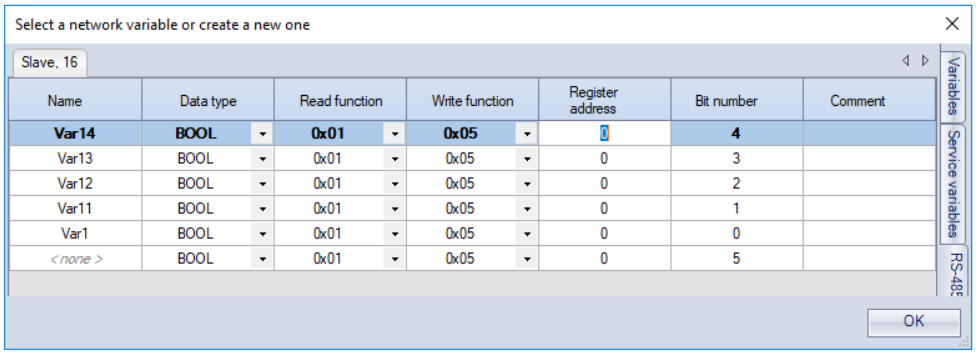


Fig. 5.8 Network variables for Master interface

For more details about network variables for Master interface see 4.3.3.1.

If the interface is configured as a Slave, all network variables to be requested by the Master are shown in one list (Fig. 5.9). For more details about network variables for Slave interface see 4.3.3.3.

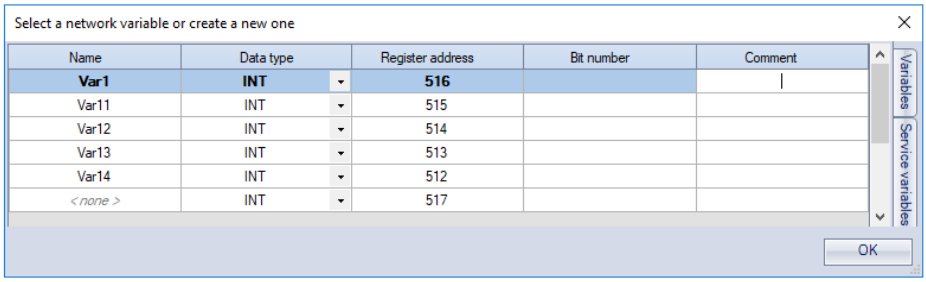


Fig. 5.9 Network variables for Slave interface

6 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.

The library ***Project macros*** comprises the macros created by the user, imported or included to the project from Online Macro Database.

View options can be changed using the icons located on the toolbar of the panel.

6.1 Functions

The library contains the following function groups:

- Logical operators
- Mathematical operators
- Relational operators
- Bitshift operators
- Bit operators

6.1.1 Logical operators

- Conjunction (AND)
- Disjunction (OR)
- Negation (NOT)
- Exclusive OR (XOR)

The logical operators can operate with Boolean or Integer variables.

If the input values are Integer, the operation is performed for each bit separately and the output is also Integer.

6.1.1.1 Conjunction (AND)

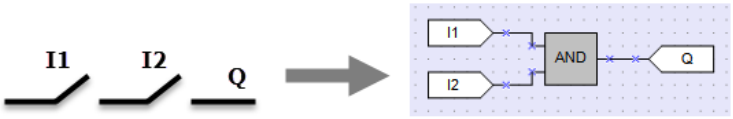


Fig. 6.1

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

Table 6.1 Truth table

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

6.1.1.2 Disjunction (OR)

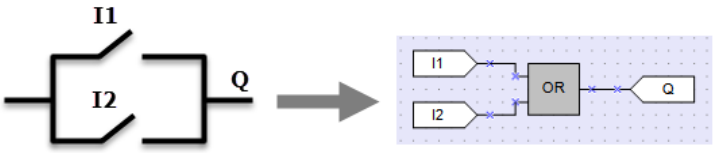


Fig. 6.2

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.

Table 6.2 Truth table

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

6.1.1.3 Negation (NOT)

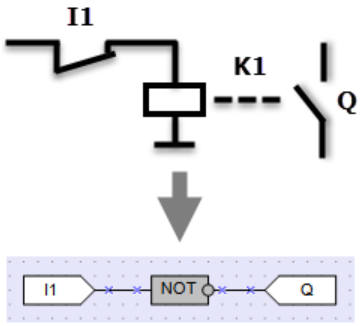


Fig. 6.3

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

Table 6.3 Truth table

I1	Q
0	1
1	0

6.1.1.4 Exclusive OR (XOR)

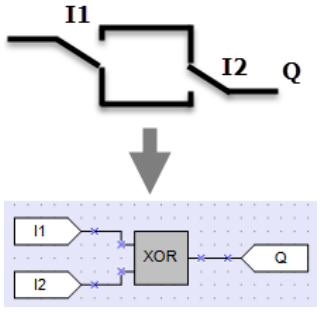


Fig. 6.4

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

Table 6.4 Truth table

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

6.1.2 Mathematical operators

There are different operators for different data types:

Table 6.5

Operator	Integer	Real
Addition	ADD	fADD
Subtraction	SUB	fSUB
Multiplication	MUL	fMUL
Division	DIV	fDIV
Modulo operation	MOD	-
Power function	-	fPOW
Absolute value	-	fABS

6.1.2.1 Addition (ADD, fADD)

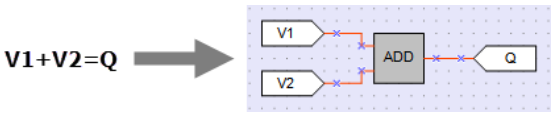


Fig. 6.5

The function **ADD** operates with Integer variables, the function **fADD** operates with Real variables.

The output value **Q** is the sum of the input values.

Example:

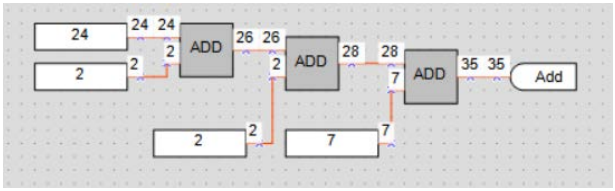


Fig. 6.6

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

6.1.2.2 Subtraction (SUB, fSUB)

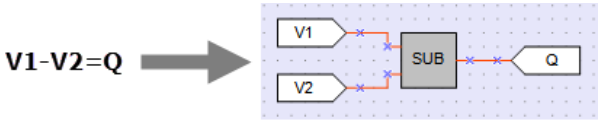


Fig. 6.7

The function **SUB** operates with Integer variables, 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 1:

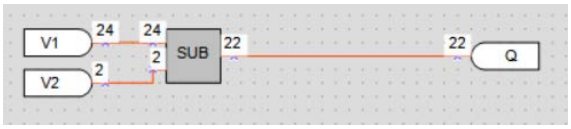


Fig. 6.8

If the value **I1** is less than the value **I2**, the output is calculated as follows:

Q = I1 + 0x100000000 – I2

0x100000000 = 4294967296

Example 2:

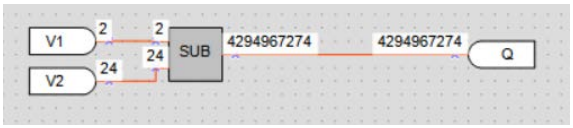


Fig. 6.9

6.1.2.3 Multiplication (MUL, fMUL)

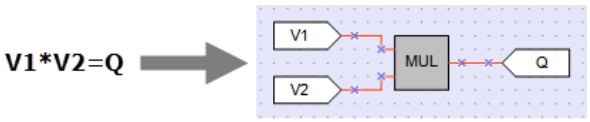


Fig. 6.10

The function **MUL** operates with Integer variables, the function **fMUL** operates with Real variables.

The output value **Q** is the product of the input values.

Example:

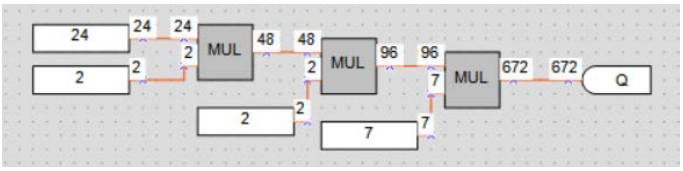


Fig. 6.11

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

6.1.2.4 Division (DIV, fDIV)

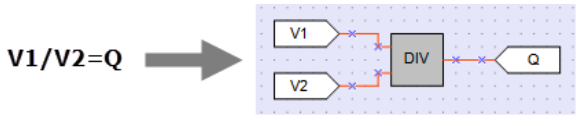


Fig. 6.12

The function **DIV** operates with Integer 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 integer, it is rounded down to an integer.

In case of division by 0 the output value is 0xFFFFFFFF.

6.1.2.5 Modulo operation (MOD)

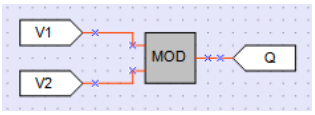


Fig. 6.13

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

$$Q = \left[\frac{V1}{V2}\right]$$

Example:

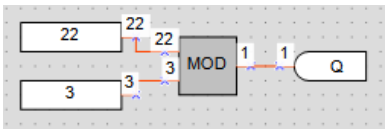


Fig. 6.14

6.1.2.6 REAL-Power function (fPOW)

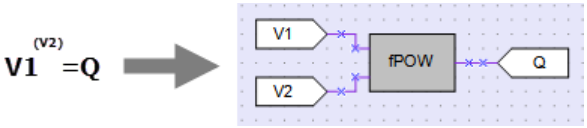


Fig. 6.15

The function **fPOW** operates with Real variables.

The output value **Q** is the value **I1** raised to the power of the value **I2**.

Example:

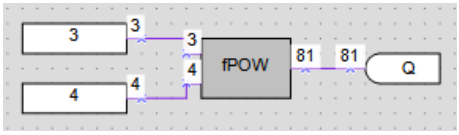


Fig. 6.16

6.1.2.7 REAL-Absolute function (fABS)

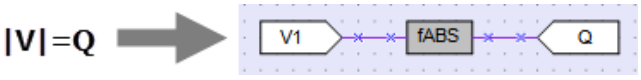


Fig. 6.17

The function **fABS** operates with Real variables.

The output value Q is an absolute value of the input value.

$$Q = |V|$$

Examples:

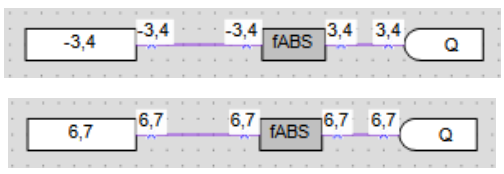


Fig. 6.18

6.1.3 Relational operators

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

6.1.3.1 Equal (EQ)

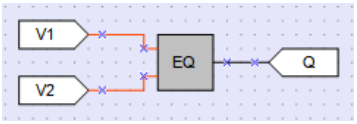


Fig. 6.19

The function **EQ** operates with Integer variables.

The output value **Q** is True if the value **I1** and the value **I2** are equal.

Table 10.6 Truth table

I1 / I2	Q
I1 = I2	1
I1 > I2	0
I1 < I2	0

Examples:

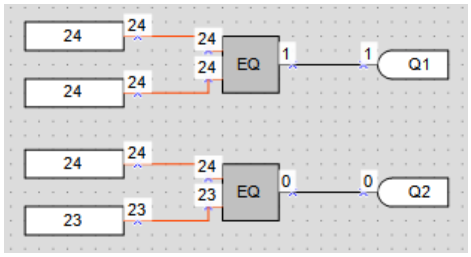


Fig. 6.20

6.1.3.2 Greater than (GT, fGT)

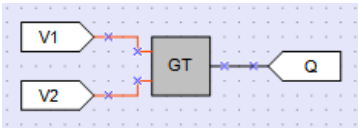


Fig. 6.21

The function **GT** operates with Integer variables, the function **fGT** operates with Real variables.

The output value **Q** is True if the value **I1** is greater than the value **I2**.

Table 6.7 Truth table

I1 / I2	Q
I1 = I2	0
I1 > I2	1
I1 < I2	0

Examples:

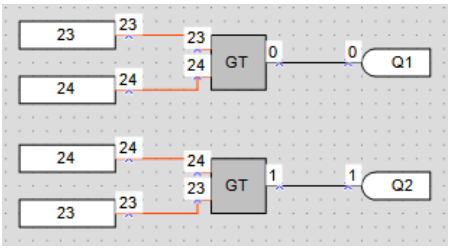


Fig. 6.22

6.1.3.3 Binary selection (SEL)

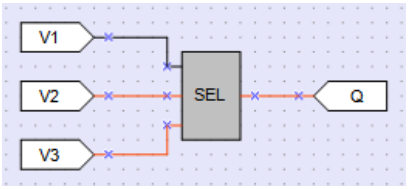


Fig. 6.23

The function **SEL** operates with Integer variables, the function **fSEL** operates with Real variables.

If **I1** = False, the output value **Q** is set to the value **I2**, else to the value **I3**.

Table 6.7 Table of states

I1	Q
0	I2
1	I3

Examples:

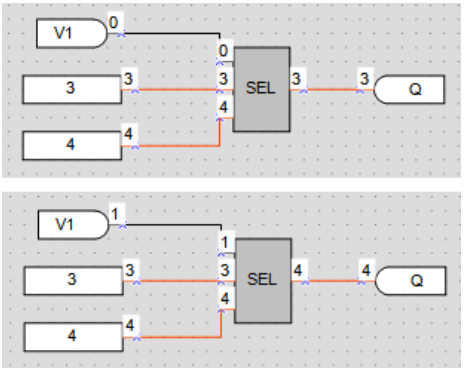


Fig. 6.24

6.1.4 Bitshift operators

The bitshift operators treat a variable as a series of bits that can be moved (shifted) to the left or right.

6.1.4.1 Shift register left (SHL)

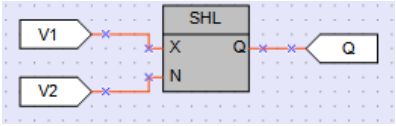


Fig. 6.25

The function **SHL** operates with Integer 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**.

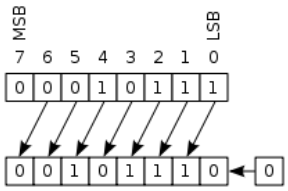


Fig. 6.26

Example: left shift of the number 38 (decimal) = 00100110 (binary) by 2 bits

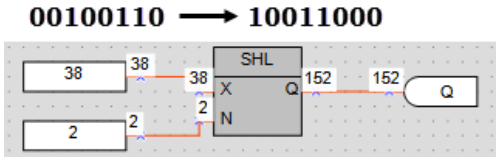


Fig. 6.27

6.1.4.2 Shift register right (SHR)

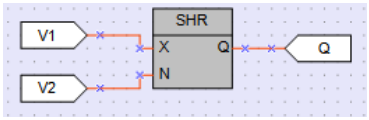


Fig. 6.28

The function **SHR** operates with Integer 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

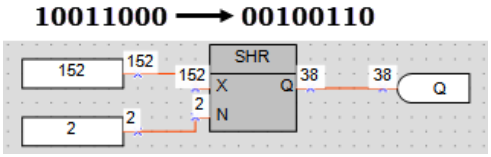


Fig. 6.29

6.1.5 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.

6.1.5.1 Read single bit (EXTRACT)

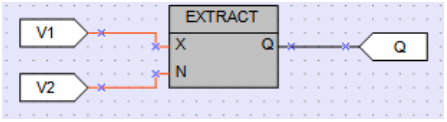


Fig. 6.30

The output value **Q** (Boolean) of the function **EXTRACT** is the value of bit **N** (Integer) in the operand **X** (Integer). The bit numbering is zero-based.

Example: reading of the 5th bit from the number 81 (decimal) = 1010001 (binary):

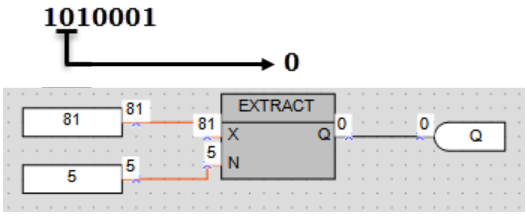


Fig. 6.31

6.1.5.2 Set single bit (PUTBIT)

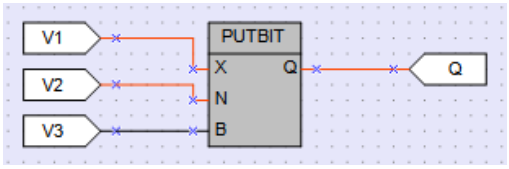


Fig. 6.32

This output value **Q** (Integer) is the value of the operand **X** (Integer) where the bit **N** (Integer) is set to the value at the input **B** (Boolean). The bit numbering is zero-based.

Example: setting of the 4th bit to 1 in the number 38 (decimal) = 100110 (binary):

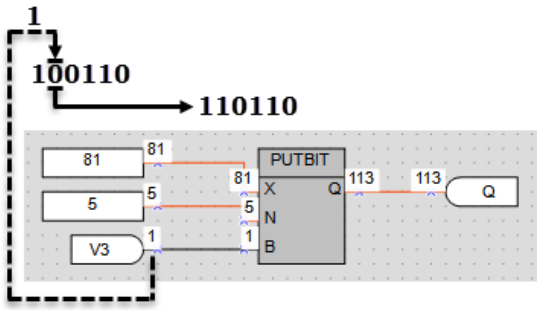


Fig. 6.33

6.1.5.3 Decoder (DC32)

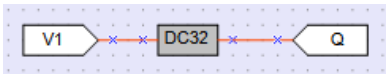


Fig. 6.34

The decoder converts a binary code at the input to a position code at the output.

Decoding is preceded by the bitwise logical operation **AND** with the operand 0x1F (11111b).

Table 6.8 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:

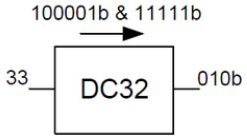


Fig. 6.35

6.1.5.4 Encoder (CD32)

Fig. 6.36

The encoder converts a position code at the input to a binary code at the output.

If there is more than one “1” bits in the position code, the encoder operates only with the most significant “1” bit.

For truth table see table 10.8 for Decoder.

6.2 Function blocks

The library contains the following FB groups:

- Triggers
- Timers
- Generators
- Counters
- Control

6.2.1 Triggers

- RS trigger reset dominant (RS)
- SR trigger set dominant (SR)
- Rising edge (RTRIG)
- Falling edge (FTRIG)
- D-trigger (DTRIG)

6.2.1.1 RS trigger reset dominant (RS)

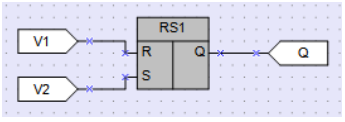


Fig. 6.37

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.

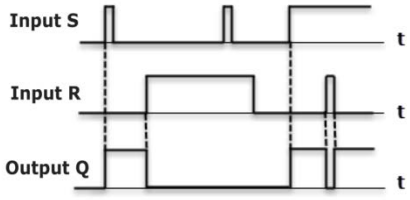


Fig. 6.38

6.2.1.2 SR trigger set dominant (SR)

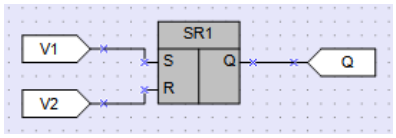


Fig. 6.39

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.

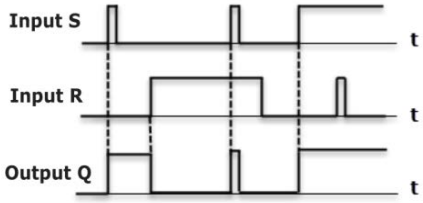


Fig. 6.40

6.2.1.3 Rising edge (RTRIG)

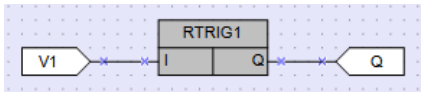


Fig. 6.41

Detector for a rising edge

The output **Q** remains False until a rising edge at the input **I**. As soon as the input **I** becomes True, the output becomes also True and remains for one program cycle.

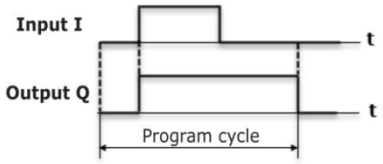


Fig. 6.42

6.2.1.4 Falling edge (FTRIG)

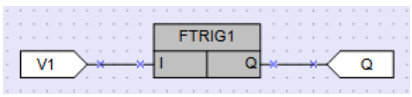


Fig. 6.43

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 for one program cycle.

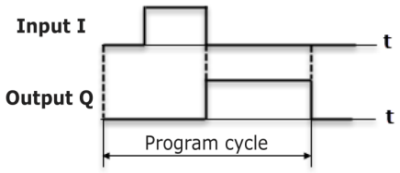


Fig. 6.44

6.2.1.5 D-trigger (DTRIG)

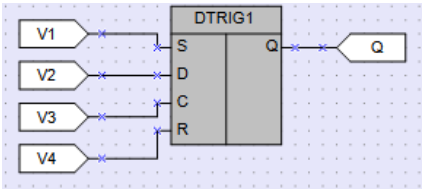


Fig. 6.45

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**.

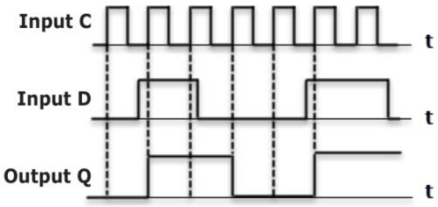


Fig. 6.46

The output **Q** can be forced set to True with a rising edge at the input **S** (Set) and forced 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.

6.2.2 Timers

- Pulse (TP)
- ON-delay timer (TON)
- OFF-delay timer (TOF)
- Timer (CLOCK)
- Weekly timer (CLOCKW)

6.2.2.1 Pulse (TP)

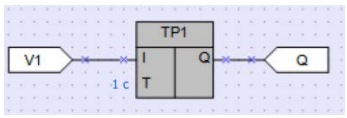


Fig. 6.47

The block **TP** is used to generate one output pulse with the specified pulse duration.

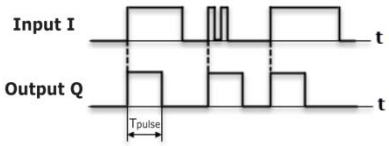


Fig. 6.48

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.

The pulse duration and the time unit can be set in Property Box.

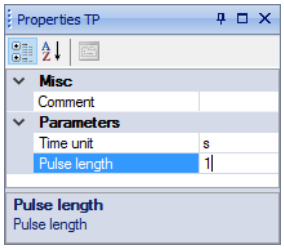


Fig. 6.49

Time range: 0 4147200000 ms or 48 days.

6.2.2.2 ON-delay timer (TON)

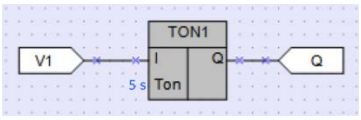


Fig. 6.50

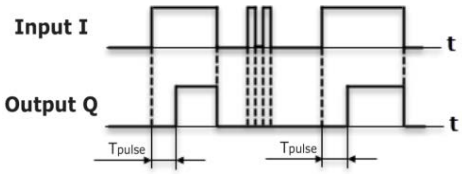


Fig. 6.51

The output **Q** = False if the input **I** = False. The delay time specified at the input $\mathbf{T_{ON}}$ starts with a rising edge at the input **I**. When the time $\mathbf{T_{ON}}$ is elapsed, the output **Q** becomes True and remains until a falling edge at the input **I**. Input changes shorter than $\mathbf{T_{ON}}$ are ignored.

The delay time and the time unit can be set in Property Box.

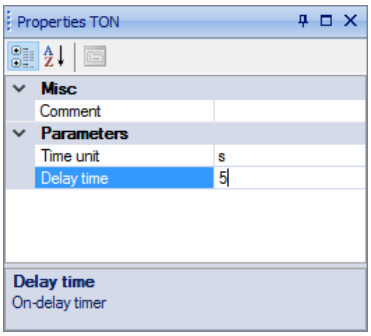


Fig. 6.52

Time range: 0 4147200000 ms or 48 days.

6.2.2.3 OFF-delay timer (TOF)

Fig. 6.53

The output **Q** = False if the input **I** = True. The delay time specified at the input $\mathbf{T_{OFF}}$ starts with a falling edge at the input **I**. When the time $\mathbf{T_{OFF}}$ is elapsed, the output **Q** becomes False and remains until a rising edge at the input **I**. Input changes shorter than $\mathbf{T_{OFF}}$ are ignored.

The delay time and the time unit can be set in Property Box.

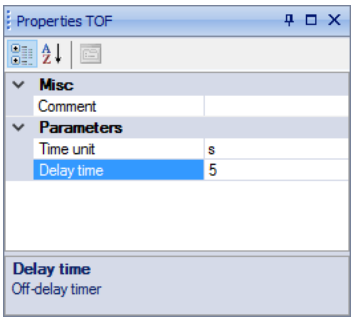


Fig. 6.54

Time range: 0 4147200000 ms or 48 days.

6.2.2.4 Timer (CLOCK)

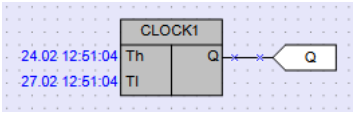


Fig. 6.55

The block **CLOCK** is an interval timer controlled by a real-time clock.

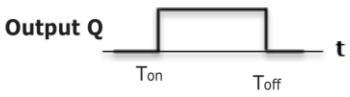


Fig. 6.56

The times $\mathbf{T_H}$ and $\mathbf{T_L}$ can be set in Property Box.

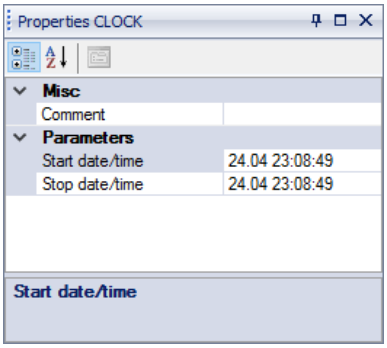


Fig. 6.57

Time range: from 0.00 seconds to 24 hours.

If $T_H < T_L$, the state of the output **Q** is as follows:

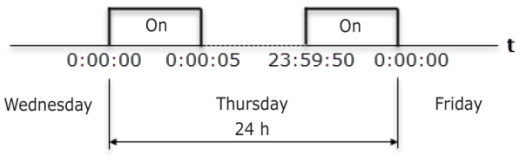


Fig. 6.58

6.2.2.5 Week timer (CLOCKW)

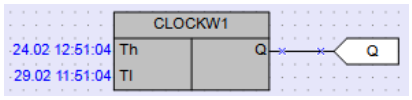


Fig. 6.59

The block **CLOCKW** is an interval timer with the parameter ***Weekday*** controlled by a real-time clock.

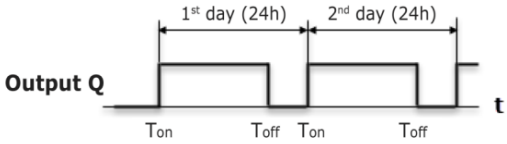


Fig. 6.60

The times T_H and T_L can be set in Property Box.

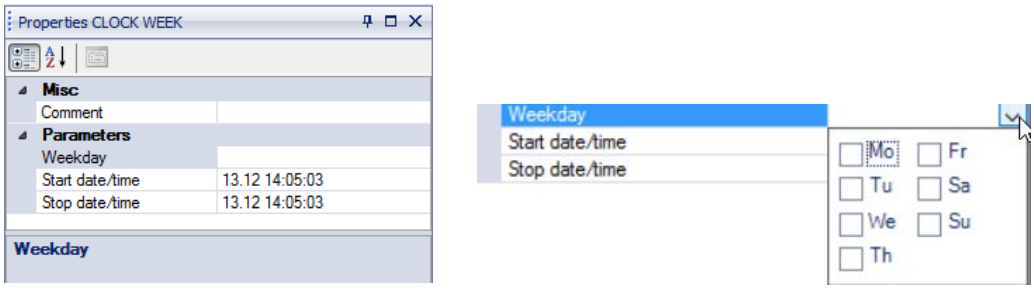


Fig. 6.61

Time range: from 0.00 seconds to 24 hours.

6.2.3 Generators

- Pulse generator (BLINK)

6.2.3.1 Pulse generator (BLINK)

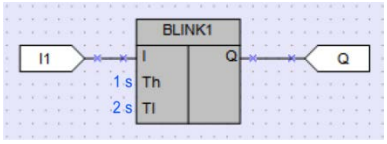


Fig. 6.62

If the input **I** becomes True, the block **BLINK** generates a square wave on the output **Q** with a period of $T_H + T_L$ starting with an interval of the duration of T_L, followed by a pulse of the duration of T_H. It continues that way until the input **I** is False.

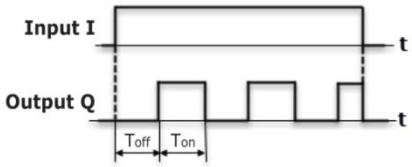


Fig. 6.63

The times T_H and T_L and the time units can be set in Property Box.

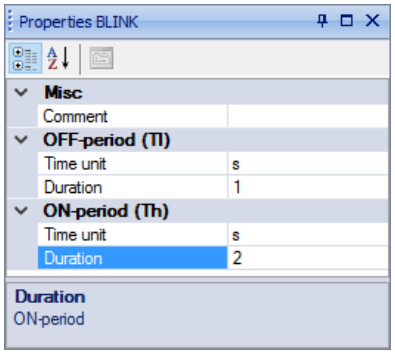


Fig. 6.64

Time range: 0 4233600000 milliseconds or 49 days.

6.2.4 Counters

- Threshold counter with self-reset (CT)
- Universal counter (CTN)
- Threshold counter (CTU)

6.2.4.1 Threshold counter with self-reset (CT)

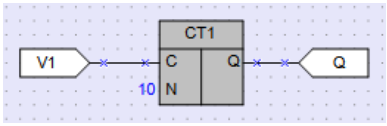


Fig. 6.65

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 for one program cycle.

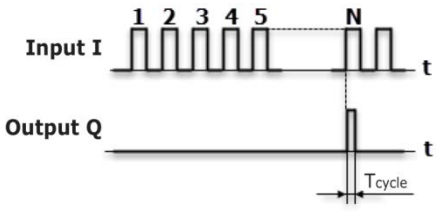


Fig. 6.66

The parameters ***Setting*** and ***Persistence*** can be set in Property Box.

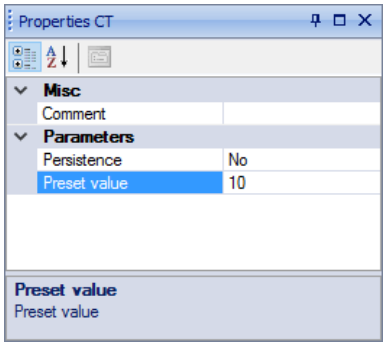


Fig. 6.67

Threshold range: 0 65535.

If ***Persistence*** = Yes, the state of the counter is permanently stored in the non-volatile memory.

6.2.4.2 Universal counter (CTN)

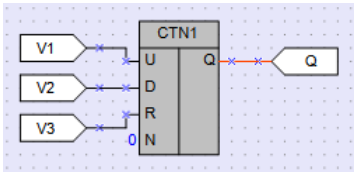


Fig. 6.68

The output **Q** is of type Integer. 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** becomes the value ***Setting*** at the input **N**.

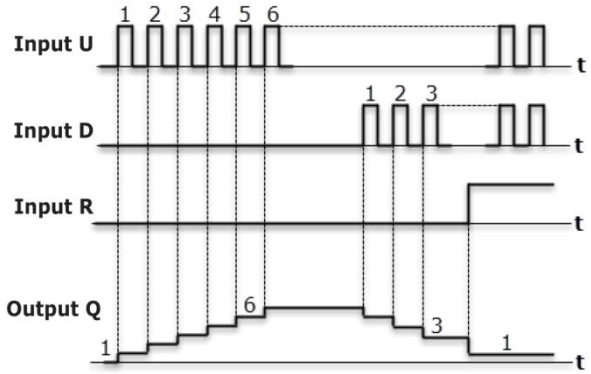


Fig. 6.69

The input **U** has higher priority than the input **D**.

The parameters ***Setting*** and ***Persistence*** can be set in Property Box.

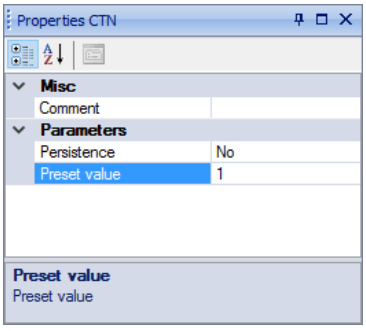


Fig. 6.70

Setting range: 0 65535.

If ***Persistence*** = Yes, the state of the counter is permanently stored in the non-volatile memory.

6.2.4.3 Threshold counter (CTU)

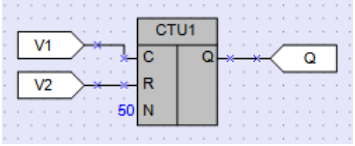


Fig. 6.71

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 until a rising edge at the input **R**. The input **R** has higher priority than the input **C**.

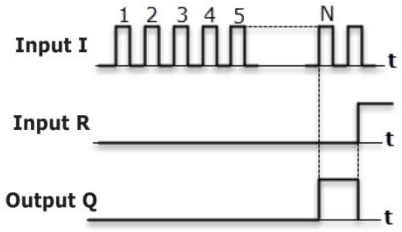


Fig. 6.72

The parameter ***Setting*** can be set in Property Box.

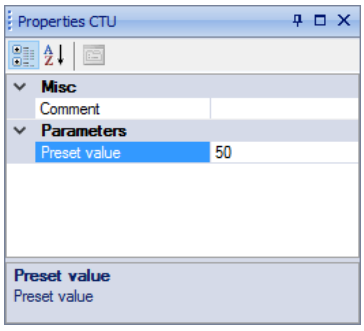


Fig. 6.73

Threshold range: 0 65535.

6.2.5 Controllers

- PID controller (PID)

6.2.5.1 PID controller (PID)

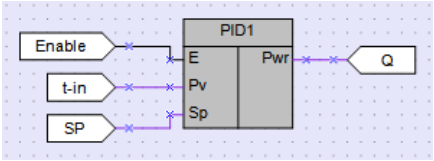


Fig. 6.74

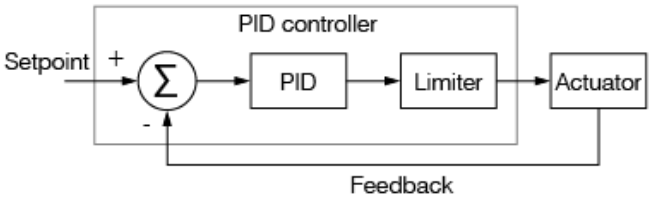


Fig. 6.75

The function block **PID** is used for implementation of the proportional-integral-derivative control.

Table 6.9 PID block inputs/outputs

Name	Type	Description	Values
E	BOOL	Enable control (0 = Off, 1 = On). If disabled, the output becomes the value of the parameter ***Output safe state***.	0/1
Pv	REAL	Process value	
Sp	REAL	Setpoint	
Pwr	REAL	Output power, %	0-100

Table 6.10 PID block parameters

Name	Type	Description	Values
Control mode	BOOL	0-Heating 1-Cooling	0/1
Output safe state	REAL	Output value in off-state, %	0-100
Proportional gain (Kp)	REAL	Coefficient for proportional control	0-100
Integration time (Ti), s	REAL	Time constant for integral control, sec.	0-9999
Derivative time (Td), s	REAL	Time constant for derivative control, sec.	0-100
Max. output value	REAL	Output upper limit, % (default 80%)	0-100
Min. output value	REAL	Output lower limit, % (default 20%)	0-100
Start Auto-Tuning	BOOL	Start auto-tuning if True. The variable can be set only using the block ***WriteToFB***.	0/1

Loop tuning

Tuning of a control loop is the adjustment of its control parameters (proportional gain, integral time, derivative time) to the optimum values for the desired control response.

Manual tuning can be performed using the blocks ***WriteToFB*** and ***ReadFromFB***: 

The parameters can be set using the block ***WriteToFB*** (Fig. 6.76) or in Property Box (Fig. 6.77). The following parameters can be read using the block ***ReadFromFB***:

- Calculated proportional coefficient
- Calculated integration time
- Calculated derivative time
- Flag “Stop Auto-Tuning”

For further instructions see section 7.7 “Reading / writing for function blocks”.

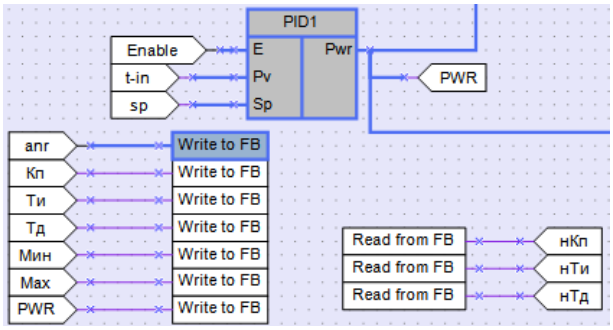


Fig. 6.76

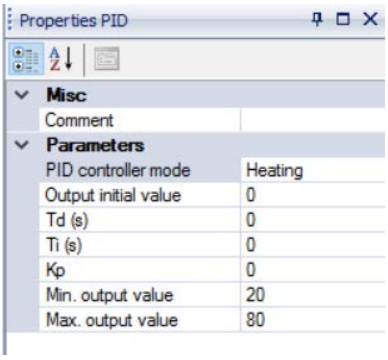


Fig. 6.77

To use auto-tuning, add the block ***WriteToFB*** to the circuit program and set the reference to the variable ***Start Auto-Tuning*** of the FB **PID**. To start the auto-tuning enable control (E = True) and set the variable ***Start Auto-Tuning*** to True.

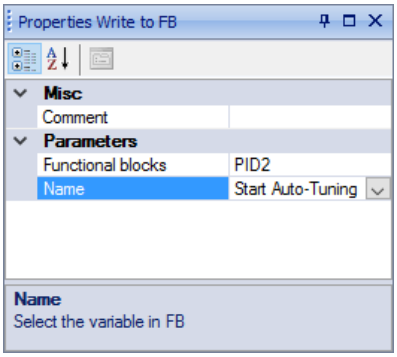


Fig. 6.78

Upon completion of the auto-tuning, the new values of the parameters ***Kp***, ***Ti*** and ***Td*** are calculated and the variable ***Stop Auto-Tuning Flag*** becomes True. If ***Start Auto-Tuning*** becomes False, ***Stop Auto-Tuning Flag*** becomes False as well.

If you reset the variable ***Start Auto-Tuning*** to False before the completion of auto-tuning, the process is stopped, the flag becomes False and new coefficient values remain not calculated.

During the process of the auto-tuning, a test signal limited by parameters ***Max. output value*** and ***Min. output value*** is applied to the block output.

Auto-tuning for Heating Mode

1. The current value is less than the setpoint, the output becomes the maximum value.
2. As soon as the current value exceeds the setpoint, the output becomes the minimum value.
3. Reiteration of the steps 1 and 2.

4. Calculated PID coefficients are saved into appropriate FB variables and the stop flag becomes True.

If the maximum output value is not enough to reach the setpoint, the auto-tuning cannot be completed and will continue until it is stopped manually.

6.3 Project macros

Macro is a function blocks created in the same way as a main program. It can be saved in the project library for further use in the project or exported to a file for use in other projects. Macros can be also imported from a file or downloaded from the Online Macro Database into the project library.

6.3.1 New macro

There are some specific aspects of working with macros:

1. Select ***File > New macro*** in the menu. Specify the number of inputs and outputs in the opened dialog and confirm with ***OK***. The new empty macro is opened in a separate workspace.

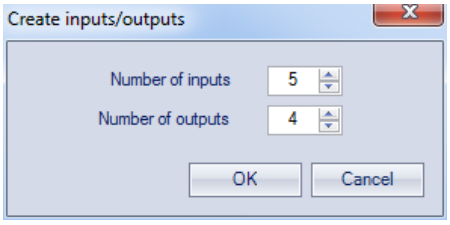


Fig. 6.79

The number of inputs and outputs can be always changed using context menu in the workspace.

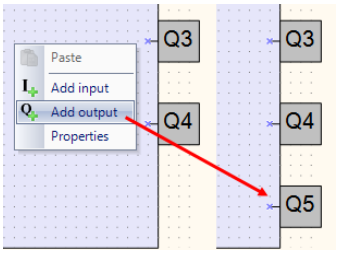


Fig. 6.80

To remove an input or an output, click it with the right mouse button and select ***Delete***.

Data type for each input and output can be selected in Property Box.

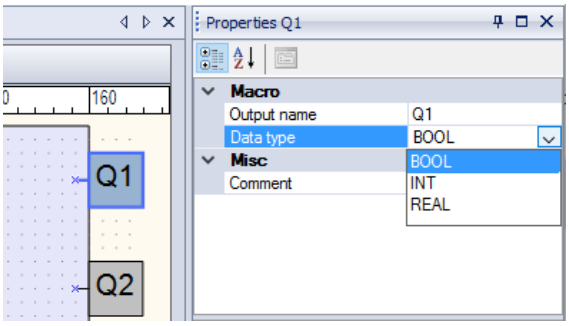


Fig. 6.81

2. Give a name, a description and a group to the macro in Property Box.

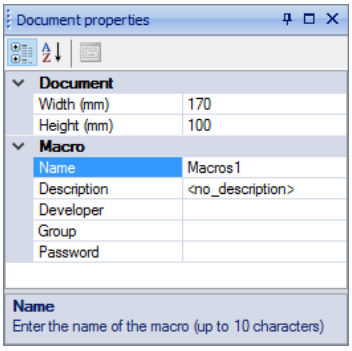


Fig. 6.82

The name is displayed in the workspace tab. It is the name of the macro in the project. The description is displayed in the tooltip, when the mouse is over the macro in the workspace. The group name is used in the library.

3. To use the macro in the project, drag-and-drop it from Library Box into the main program workspace.

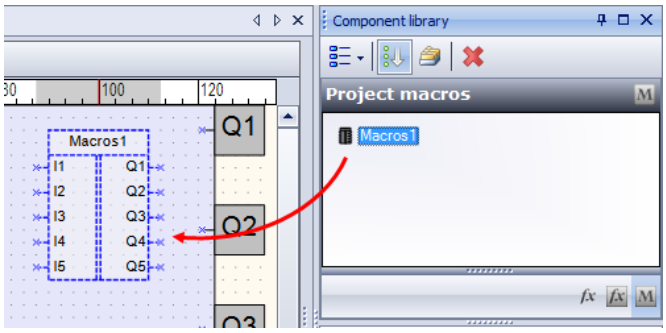


Fig. 6.83

4. The macro can be saved in the project as a new macro with another name using the menu item ***File > Save macro as***

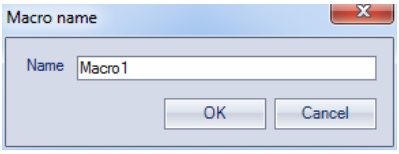


Fig. 6.84

To save a macro as a file see 6.3.2 “Export macro”. To import a macro from a file see 6.3.3 “Import macro”.

5. If a function block is used in the macro, the user can define whether the FB properties are available as properties of the macro in the main program. If the parameter ***Use in macro*** is set to ***Yes***, the parameters of the FB became parameters of the macro and a new option ***Parameter of macro*** is added to the macro. With this option the user can define the name of each FB parameter in the macro.

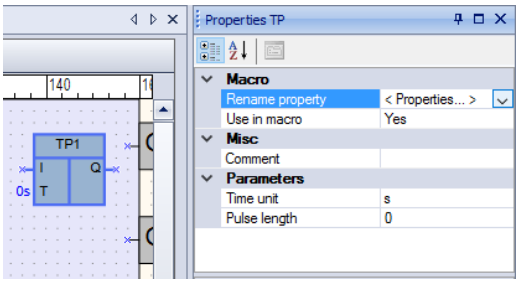


Fig. 6.85

6.3.2 Export macro

A macro can be exported to the file if the macro workspace is active. To open the project macro in the separate workspace, select it in the library and use the item ***Edit macro*** in the macro context menu.

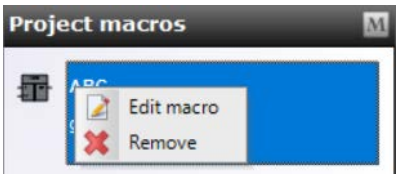


Fig. 6.86

To export the macro use the menu item ***File > Export Macro***, select the path, specify the filename and confirm with ***OK***. The macro file has the extension ****.tpl***. Finally the report about failure/successful export of the macro is displayed.

6.3.3 Import macro

A macro can be imported from the file into the library if the main program is active. Use the menu item ***File > Import macro***. In the opened dialog select the file and confirm with ***OK***. The macro is added to the library into the section ***Macros***.

6.3.4 Download macro

To download macros from ***Online Macro Database*** an internet connection is needed. Select the menu item ***File > Component Manager*** (the main program must be active) to open ***Component Manager*** in a separate window (see 2.9).

6.4 Display elements

Display elements are elements of the library that control the information displayed on the device display. They are available in Library Box if the workspace with a display form is active, and can be placed within a 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*** is the position of the first (left) character placeholder of the element from the left form edge (from 0 to 15).
- ***Coordinate Y*** is the position of the first (left) character placeholder of the element from the upper form edge, depending on the number of the rows in the form.
- There are two options for determining the coordinates: constant (default) or variable. To use the 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.

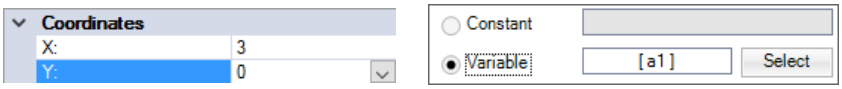


Fig. 6.87

- ***Variable*** – click ***Select*** to select an integer variable from the list and confirm with ***OK***. The display element will move according to the coordinate value controlled by the variable.

– The ***Length*** of the element is the number of the displayed characters (number of placeholders). The display element occupies one display row in height, its length can be changed from 1 to 16 characters maximum.

6.4.1 Text box

Text box is used to display constant text.

Parameters:

Text – text to be displayed. The text length may not exceed the value specified in the parameter ***Length***.

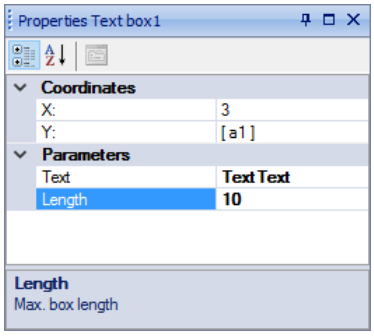


Fig. 6.88

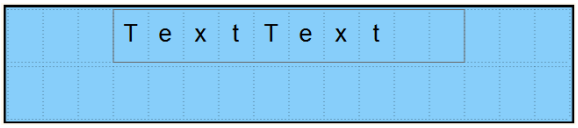


Fig. 6.89

6.4.2 I/O box (INT/REAL)

I/O box (INT/REAL) is used to display or change the selected variable of the type Integer or Real.

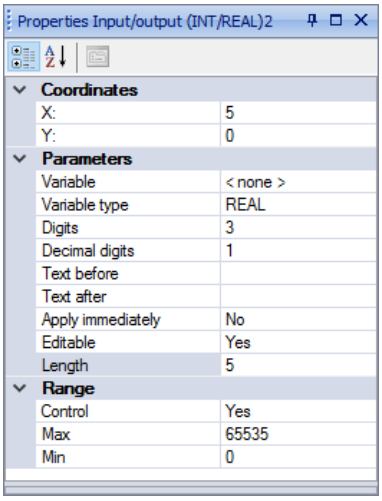


Fig. 6.90

Parameters:

Variable – the reference to a program variable. Use the icon ... to select a variable.

Data type – the data type of the variable: Integer or Real. If the variable has been already selected, its data type is applied.

Digits – the total number of the displayed characters for the variable

Decimal digits – the number of the characters after the decimal point: 0 6 characters or automatic (***Auto***). For detailed information refer to the operation manual of the device.

Text before – the text to the left of the displayed variable value

Text after – the text to the right of the displayed variable value

Editable – if ***Yes***, the displayed value can be changed using the device function buttons

> **Note:** An output variable should be referenced. The option has no effect with an input variable.

Length – the number of characters that can be displayed including the text before, the variable value and the text after

Range:

The group of parameters to control the optional restrictions on the user entered value. If ***Editable*** = ***No***, the parameters of this group have no effect.

Control – if ***Yes***, the value entered using the device function buttons is limited by the parameters ***Max*** and ***Min***

Max – the maximum value for user input

Min – the minimum value for user input

Example:

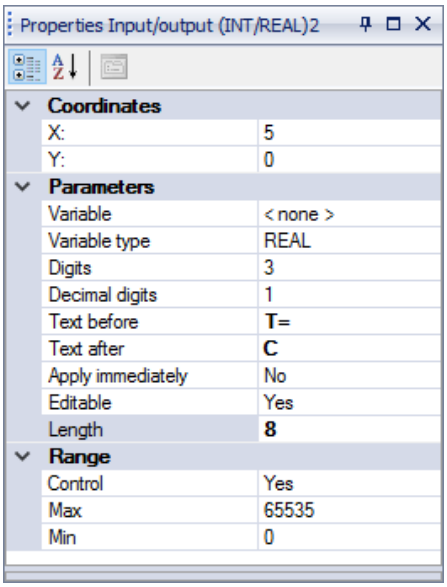


Fig. 6.91

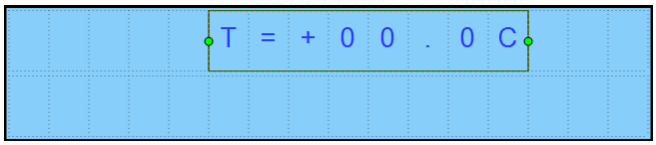


Fig. 6.92

6.4.3 I/O box (BOOL)

I/O box (BOOL) is used to display or change the selected variable of the type Boolean.

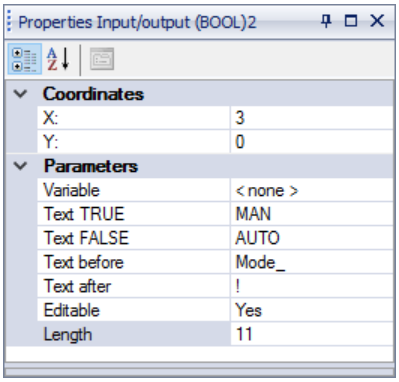


Fig. 6.93

Parameters:

Variable – the reference to a program variable. Use the icon ... to select a variable.

Text TRUE – the text displayed if the referenced variable is True

Text FALSE – the text displayed if the referenced variable is False

Text before – the text to the left of the displayed variable value

Text after – the text to the right of the displayed variable value

Editable – if ***Yes***, the displayed value can be changed using the device function buttons

> **Note:** An output variable should be referenced. The option has no effect with an input variable.

Length – the number of characters that can be displayed including the text before, the variable value and the text after

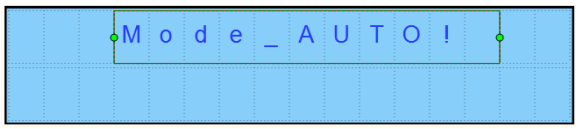


Fig. 6.94

6.4.4 Dynamic box

Dynamic box is used to display one of the text rows from the list depending on the value of the referenced output variable of the type Integer.

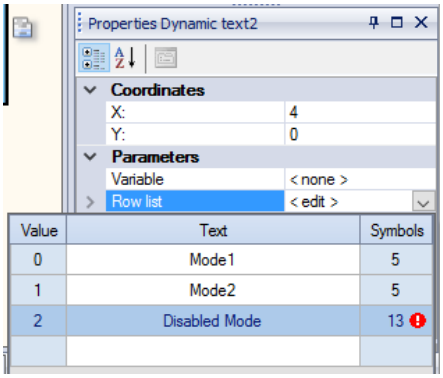


Fig. 6.95

Parameters:

Variable – the reference to a program variable. Use the icon ... to select a variable.

Row list – the list with text rows. The ***Text*** from the row is displayed if the value of the referenced variable is equal to the row ***ID***. The column ***Characters*** shows the number of

characters in the text. An exclamation mark is displayed near the number if the value of the parameter ***Length*** is exceeded.

Length – the number of characters that can be displayed

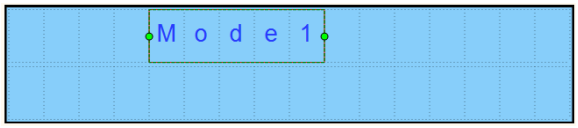


Fig. 6.96

6.4.5 ComboBox

ComboBox is used to select one of the text rows from the list using the device function buttons. The ID of the row is saved in the referenced input variable of the Integer type.

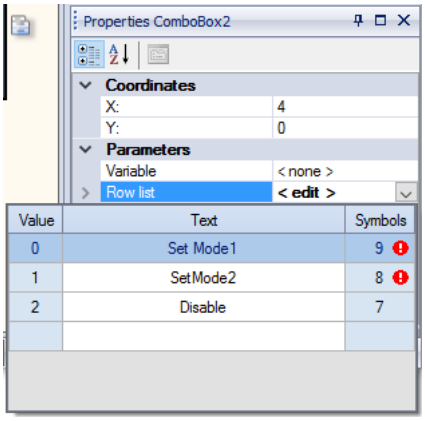


Fig. 6.97

Parameters:

Variable – the reference to a program variable. Use the icon ... to select a 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. An exclamation mark is displayed near the number if the value of the parameter ***Length*** is exceeded.

Length – the number of characters that can be displayed

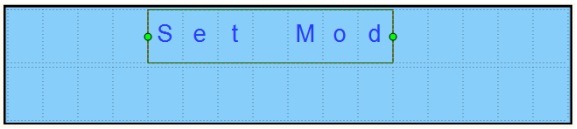


Fig. 6.98

7 Circuit program development

It is recommended to start the creation of a circuit program with planning. The plan must describe all possible states of the device during operation in form of diagram of modes, table of I/O states, electrical or functional diagram, etc.

After all the operation tasks are described, the program can be developed using standard elements from the toolbar ***Insert*** (Table 2.6) and specific elements from the project library (see 6). The project library presented in ***Library Box*** (see 2.4) contains the functions (see 6.1) and the function blocks (see 6.2) available for the target device and the macros added to the project (see 6.3).

For details about using of each element see sections 7.1 – 7.8, for other practices of program development see sections 7.9 – 7.11.

To draw connecting lines use the left mouse button:

- Click the output of the first element. The line is attached to it and follows the mouse cursor.
- To change the line direction, click on the workspace.
- Pull the line up to the input of the second element and click on it to finish the line.

The connecting line can be pulled between the element inputs /outputs associated with the same data type. To connect element inputs /outputs associated with different data types, use conversion blocks (see 7.8).

Click the element to select it. Pull the rectangle around several elements to select a group.

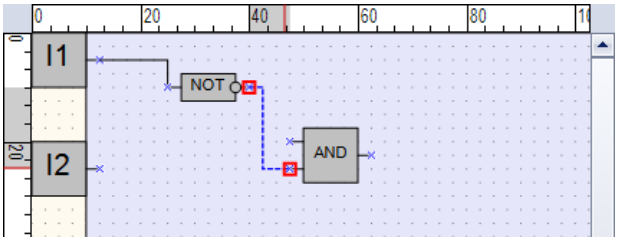


Fig. 7.1

The parameters of the program elements can be set in ***Property Box*** (see 2.5).

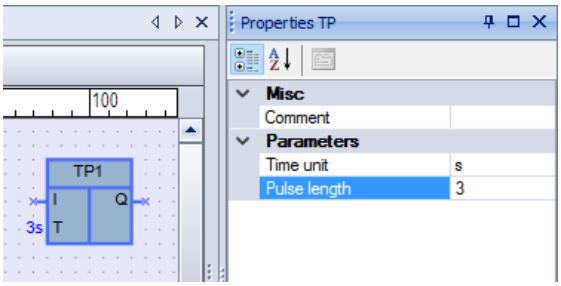


Fig. 7.2

Use element context menu for all available manipulation with the element.

7.1 Using of library elements

To place a library element in the circuit program, select the desired element in ***Library Box*** and move it onto the workspace by drag-and-drop.

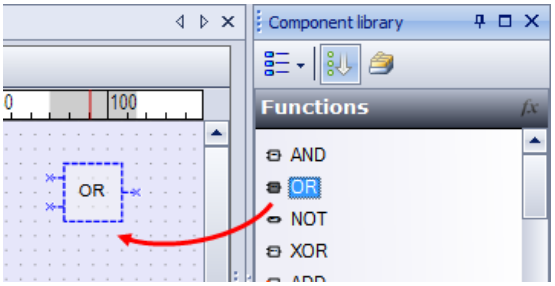


Fig. 7.3

7.2 Using of comments

Comments can be used to explain the program. The content of the comment is shown in the element's tooltip.

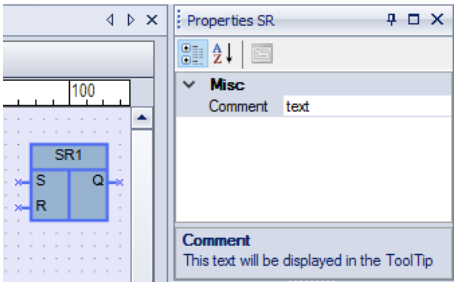


Fig. 7.4

Fig. 7.5

To add a comment to the program, click the icon A on the toolbar ***Insert*** (Table 2.6), then click the place in the workspace to place the comment block and draw a rectangle with the left mouse button.

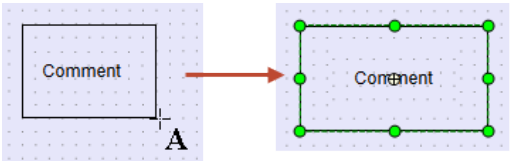


Fig. 7.6

Double-click the text box to write the text.

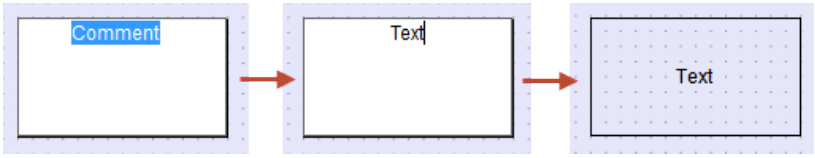


Fig. 7.7

The parameters of the comment can be changed in Property Box.

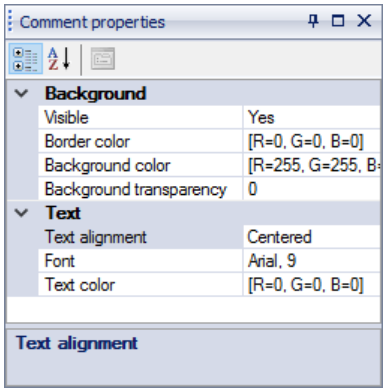


Fig. 7.8

To make the background color visible, set the parameter ***Background transparency*** to more than 20%.

7.3 Using of variables

To add a variable to the program click the corresponding icon on the toolbar ***Insert*** (see Table 2.6), then click the place in the workspace to place the variable block.

Fig. 7.9

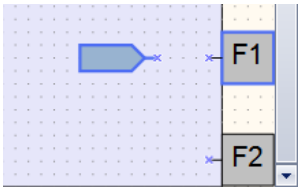


Fig. 7.10

To reference a variable to the block use the icon ... in the row ***Variable*** in Property Box. The table of project variables opens.

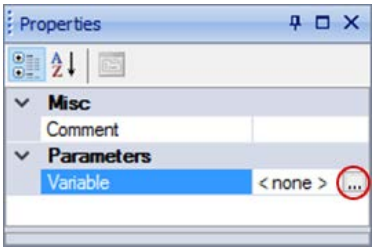


Fig. 7.11

Select a variable or create a new one in the open variable list and confirm with ***OK***.

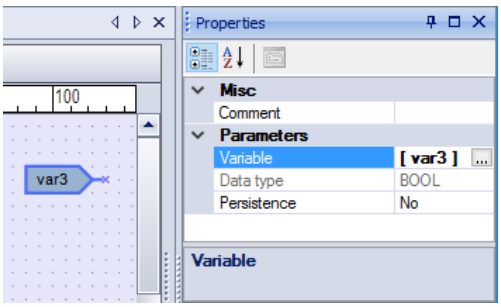


Fig. 7.12

Use network variables to exchange data with other devices connected to the target device over network. For further details about using of network variables, refer to section 7.6 “Network data exchange”.

If a variable block is highlighted in red, it means that the creation is incorrect or not completed.

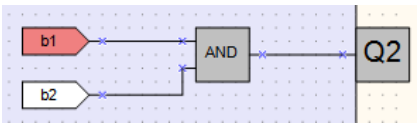


Fig. 7.13

The information about the error is displayed in the status bar.

Network output variable block has an additional parameter ***Write at the end of cycle***. If the parameter is set to ***Yes***, the new value is assigned to the variable only at the end of cycle, when all input variables are already read out.

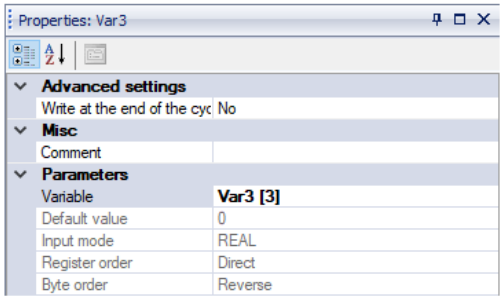


Fig. 7.14

7.4 Using of constants

To add a fixed value to the program click the icon on the toolbar ***Insert*** (Table 2.6), then click the place in the workspace to place the constant block.

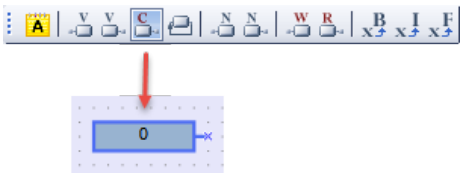


Fig. 7.15

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

7.5 Using of delay lines

A 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 on the toolbar ***Insert*** (Table 2.6) and draw a line from the output to the input of a function or a function block. The delay line is displayed as a red dashed line with an arrow.

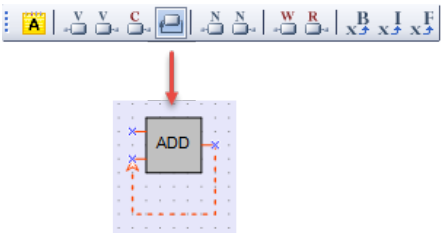


Fig. 7.16

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.

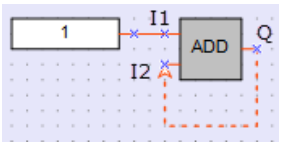


Fig. 7.17

Table 7.1 Cycle signal values

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

7.6 Network data exchange

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

The variables, which can be read via the network, are called Network output variables (N).

The variables, which can be written via the network, are called Network input variables (N).

To add a network variable to the program proceed as follows:

- Click the icon N or N on the toolbar ***Insert*** (Table 2.6).
- Click the place in the workspace to insert the variable.
- Click the icon ... in the row ***Variable*** in Property Box to select a variable for the block.

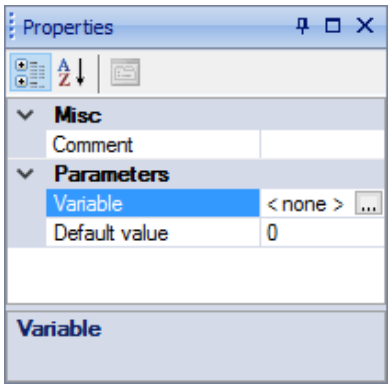


Fig. 7.18

– Select a variable or create a new one in the opened variable list and confirm with OK. The selected variable is referenced to the variable block.

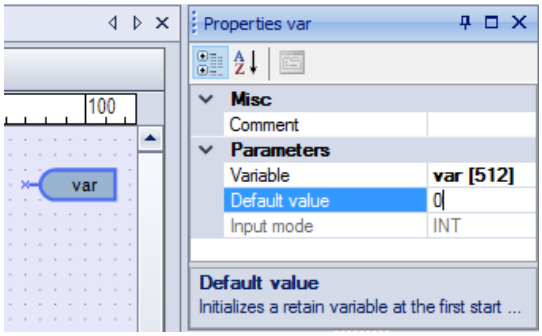


Fig. 7.19

– Connect the network variable to the desired element in the workspace.

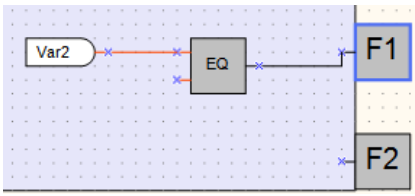


Fig. 7.20

It is recommended to start programming with the creation of variables in the variable table.

7.7 Reading / writing for function blocks

The block ***WriteToFB*** is used to change a parameter of a function block during the process.

Example:

The value of the parameter ***ON-duration*** of the function block ***BLINK1*** should be 2 or 10 depending on the value at the input I5.

To add a block ***WriteToFB*** to the program click the icon on the toolbar ***Insert*** (see Table 2.6), then click the desired place in the workspace.

Go to Property Box, select the function block ***BLINK1*** in the row ***Function block*** and the parameter of the FB in the row ***Parameter in FB*** (Fig. 7.21).

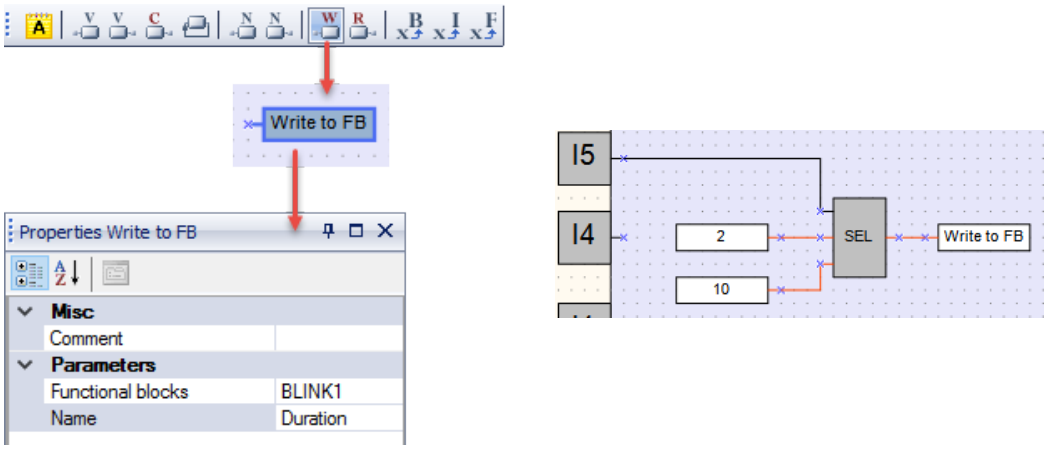


Fig. 7.21

The block ***ReadFromFB*** is used to read the current value of a function block parameter and use it in the program.

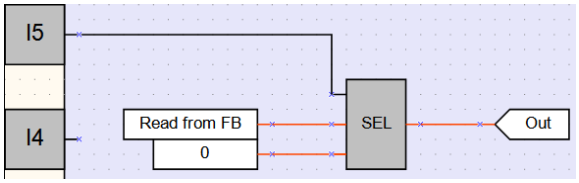


Fig. 7.22

7.8 Conversion blocks

Fig. 7.23

Conversion blocks are universal blocks used to convert an input value of any type into a value of a certain type. There are three blocks available in the ***Insert*** toolbar (Table 2.6):

Table 7.2 Conversion blocks

Conversion to BOOL	Conversion of INT or REAL to BOOL (If the input value is > 0, the output is 1 / True)
Conversion to INTEGER	Conversion of BOOL or REAL to INT (REAL is rounded down to INT)
Conversion to REAL	Conversion of BOOL or INT to REAL

To add the conversion block to the program click one of the three icons on the toolbar ***Insert*** (see Table 2.6), then click the desired place in the workspace.

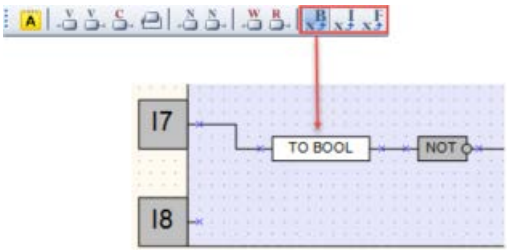


Fig. 7.24

7.9 Arrange elements

Sequence numbers of the function blocks can be automatically re-assigned by clicking the button  ***Arrange elements*** on the toolbar ***Service*** (see Table 3.6). The blocks of the same type are numbered sequentially from top to bottom and from left to right.

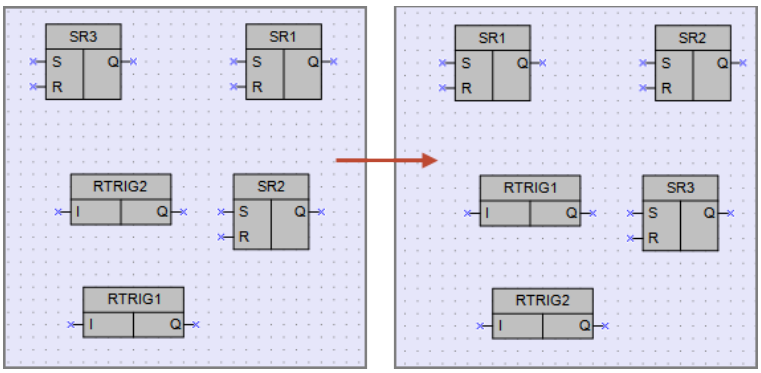


Fig. 7.25

7.10 Execution sequence

Calculation of the values for outputs and delay lines in a circuit program is performed in a certain order. To see this order click the arrow near the icon on the toolbar ***Service*** and select the ***Delay lines*** or ***Outputs*** (Table 2.6).

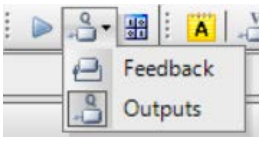


Fig. 7.26

To change the order, double-click an output or a delay line and enter the desired number.

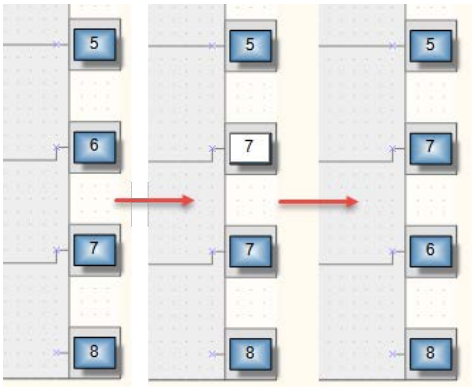


Fig. 7.27

Click the icon to deactivate the edit mode.

7.11 Simulation mode

Use the simulation mode to proof the correctness of the created program. Only offline simulation is provided. The simulation mode enables to analyze the values of all signals within the circuit program. Change the values at the digital and analog inputs as well as of variables and constants and check the values at the outputs.

To start the simulation mode, press the button ***Simulation*** on the toolbar ***Service*** (Table 2.6). A new toolbar ***Simulation*** is displayed with the following controls:

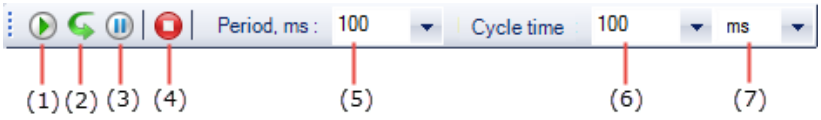


Fig. 7.28

1. Start permanent simulation mode
2. Step-by-step simulation. Click the icon to execute one program cycle.
3. Break simulation. Click the icon once more to continue the simulation.
4. Stop simulation
5. Simulation refresh time (Cycle start period)
6. Cycle time (see 3.2)
7. Selection of measurement units for cycle time: milliseconds, seconds, minutes, hours

If there are function blocks of type CLOCK or CLOCKW in the project (available only for devices with a real-time clock), an additional toolbar ***Calendar*** is displayed during simulation.

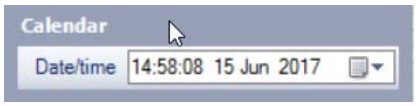


Fig. 7.29

The parameter ***Refresh time (ms)*** specifies the cycle start period.

The parameter ***Cycle time*** specifies the cycle duration.

If you want to change to the program, stop the simulation mode.

To set the value for the input signal click the input block.

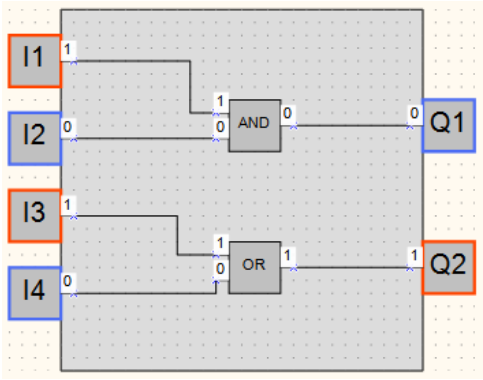


Fig. 7.30

The value of a network variable can be set as well. Click the network variable in the simulation mode, enter the prepared value in the opened dialog and confirm with ***OK***.

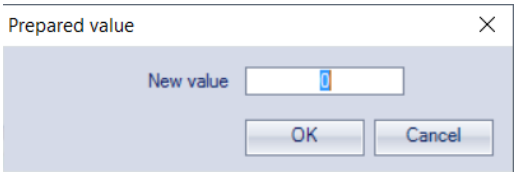


Fig. 7.31

Notes:

- Macros are excluded from simulation. Simulation for macros should be performed separately in the workspace of the macro.
- Simulation cannot be performed for blocks without connection with a device output or a network variable.

8 Display programming

To determine the information to be displayed, use the tab ***Display Manager*** in the upper left corner of the window. Display Manager is available only for target devices with display (for panel description see 2.6).

The display can be programmed using one or more display forms with “jumps” between them so that the displayed information can be changed by program events (variable change) or by operator (button events).

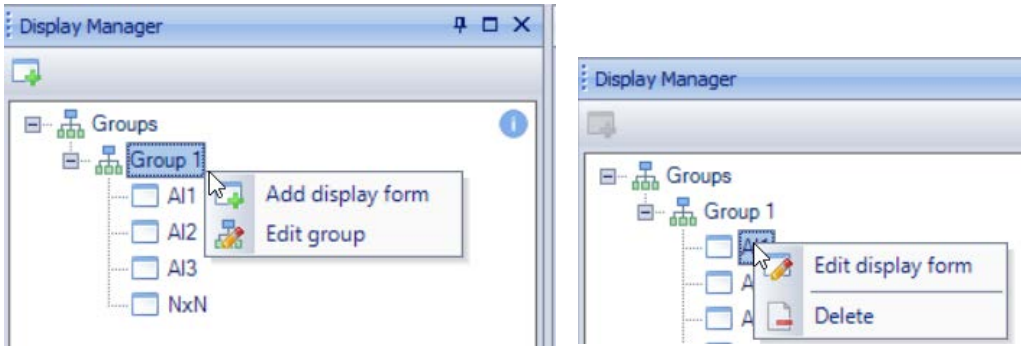


Fig. 8.1 Display Manager

To open a graphical structure of display forms in Structure Editor (see 2.6.1, 8.2), use the command ***Edit group*** in the group context menu (Fig. 8.1).

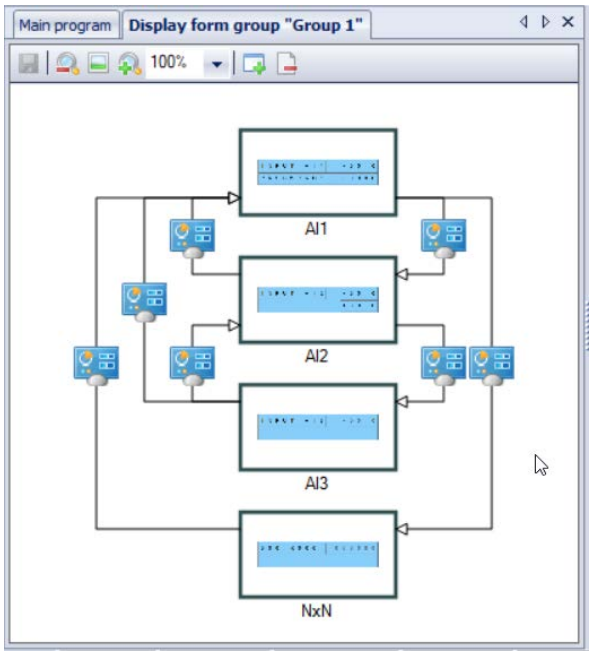


Fig. 8.2 Structure Editor

To open the selected form in Display Editor (see 2.6.2), use the command ***Edit display form*** in the form context menu or double-click the form in the tree (Fig. 8.1).

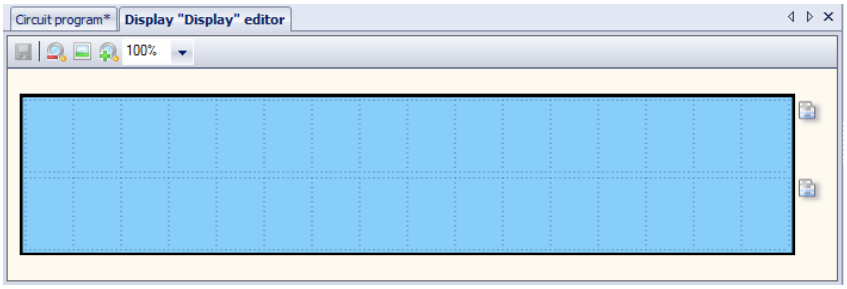


Fig. 8.3 Display Editor

8.1 Display Editor

A form may consist of several rows, at least two. The operator can switch between them using the function buttons on the device. A display form is shown in the workspace with icons on the right side, which are used to change the number of the displayed rows. The rows displayed first are outlined.

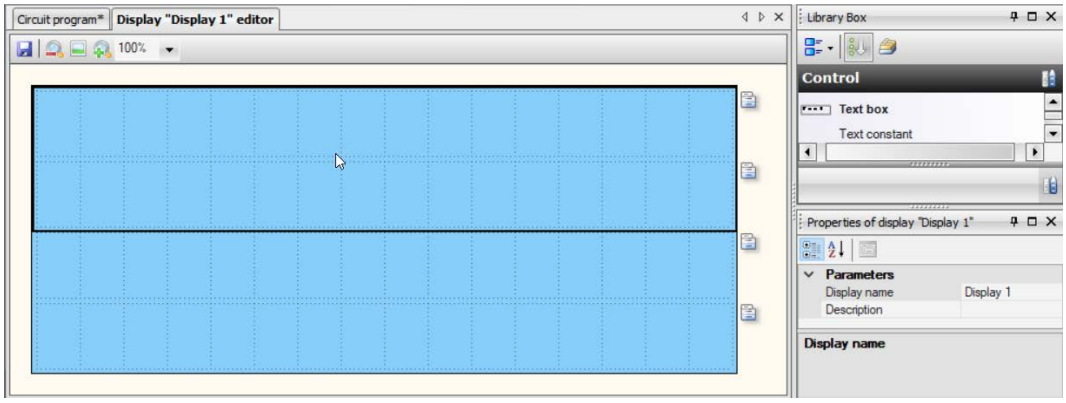


Fig. 8.4

Put the display elements from Library Box onto the form by drag-and-drop. Refer to 6.4 to read about display elements to add.

8.2 Graphical structure

By default, the graphical structure consists of one form. To add a display form to a group:

- In Display Manager, use the group context menu or the toolbar icon (active if the group is selected) (Fig. 8.1).
- In Structure Editor, use the toolbar icon (Fig. 8.2).

To remove the form:

- In Display Manager, use the context menu (Fig. 8.1).
- In Structure Editor, use the toolbar icon or the context menu (Fig. 8.2).

To change the position of a display form within a group in Display Manager, hold the Shift key while you drag-and-drop the form.

If the display structure consists of more than one form, the “jumps” should be specified to enable the navigation between them (see 8.3, Fig 8.2).

8.3 Form properties (jumps)

Select the form in the tree to show its properties:

- Name
- Description

– Jump parameters

Each new display form has a default name that can be changed here. The description is optional.

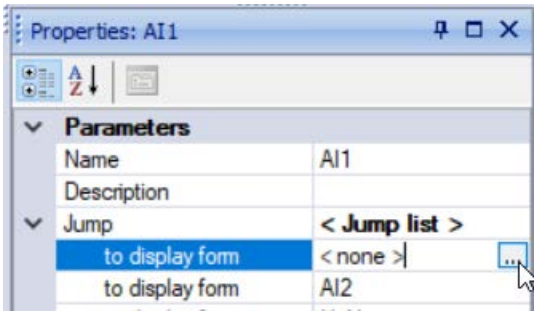


Fig. 8.5

To program the navigation through the display forms using different events, click the ... icon in the row ***to display form***. In the opened dialog ***Jump*** in the section ***to display form***, select the form to which the display should switch if the below specified event occur.

Select the event in the section ***Jump condition***, as a device event or change of a variable by value.

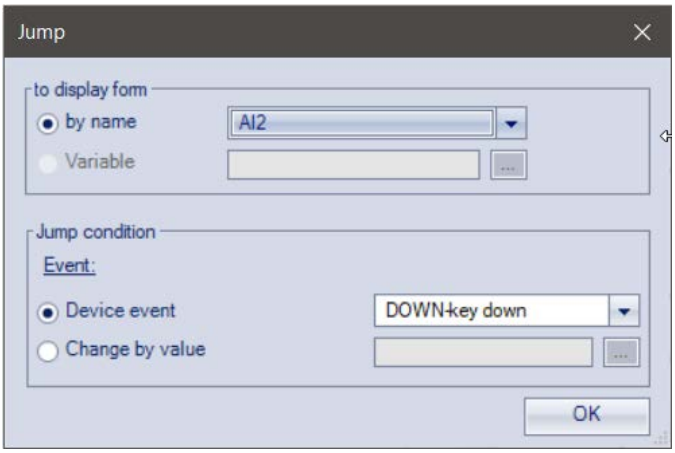


Fig. 8.6

A button event can be selected as ***Device event***.

A Boolean variable can be selected for ***Change by value*** event.

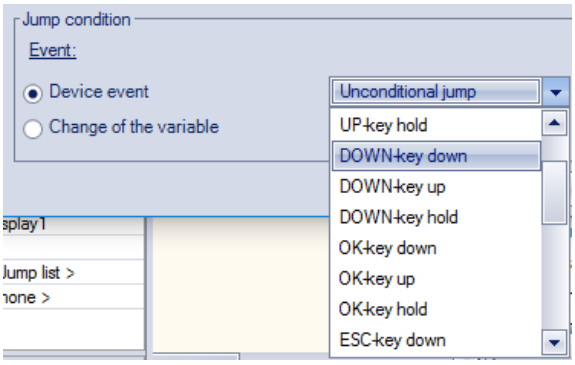


Fig. 8.7 Device events

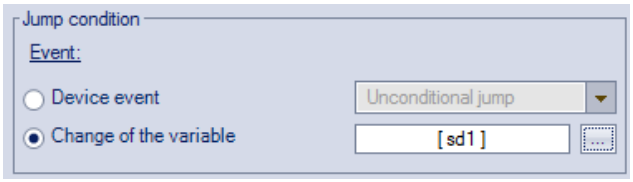


Fig. 8.8 Program events

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

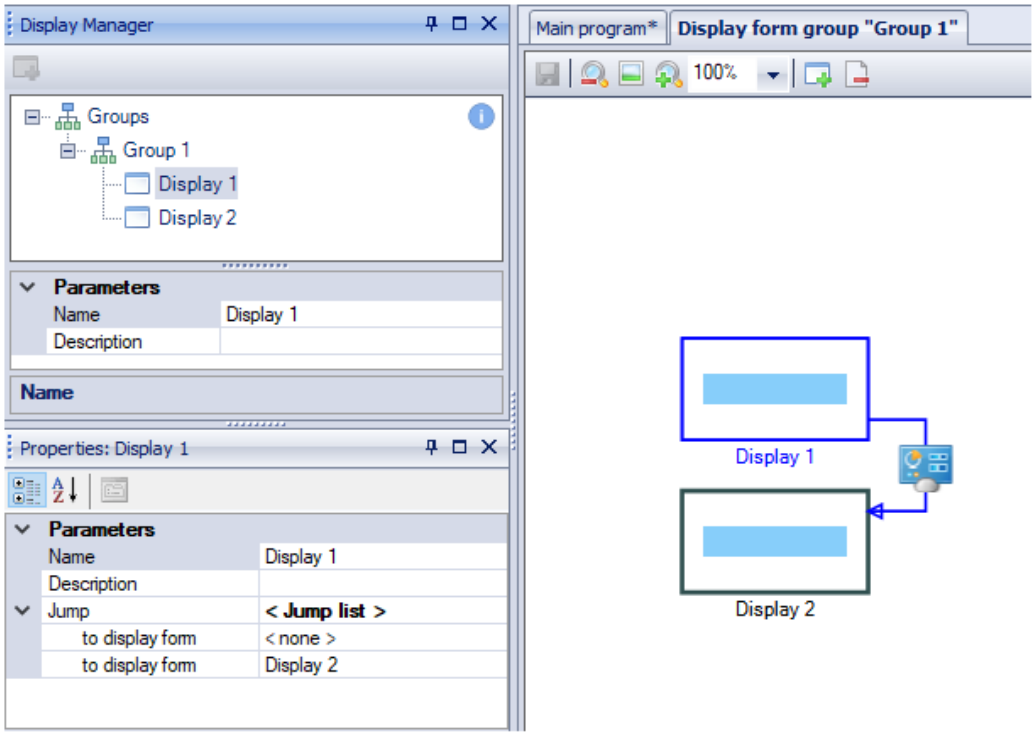


Fig. 8.9 Jump graphical representation

The jump between two display forms can occur by different events, the graphical structure can be of very high complexity level.

9 Keyboard shortcuts

Shortcut	Description	Section
Ctrl + N	New project	Menu / File
Ctrl + O	Open project	Menu / File
Ctrl + Alt + S	Save project as	Menu / File
Ctrl + S	Save project	Menu / File
Ctrl + P	Print	Menu / File
Ctrl + Z	Undo	Menu / View
Ctrl + Y	Redo	Menu / View
Ctrl + F7	Transfer application to device	Menu / Device
F1	Help	Menu / Help
Ctrl + C	Copy	Work with elements
Ctrl + V	Paste	Work with elements
Delete	Delete	Work with elements
Ctrl + →	Increase the element width	Resize element
Ctrl + ←	Decrease the element width	Resize element
Ctrl + ↑	Increase the element height	Resize element
Ctrl + ↓	Decrease the element height	Resize element
Ctrl + wheel up	Zoom+	Resize workspace
Ctrl + wheel down	Zoom-	Resize workspace

10 Program examples

Two examples with simple tasks explain the creation of a circuit program in the ALP programming software.

10.1 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 should be switched 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 should be switched 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 PR110 series with the letters "RTC" in the designation.

The task implementation with the device PR110-24.8D.4R-RTC:

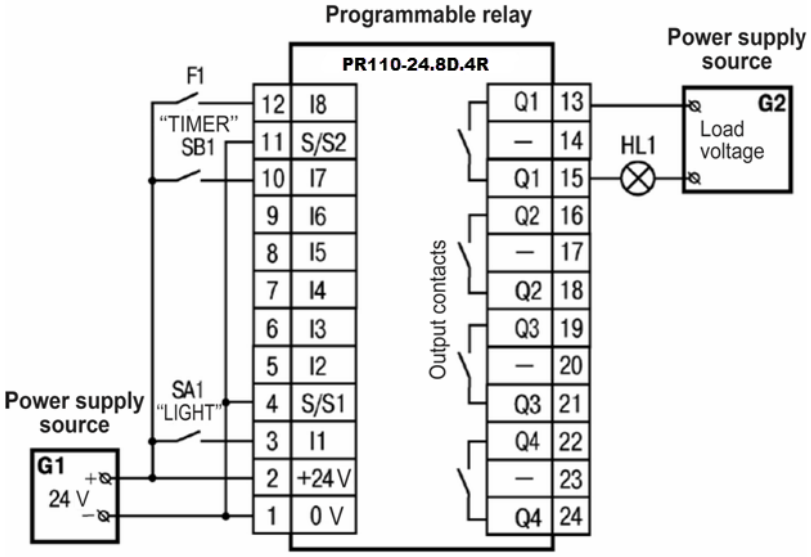


Fig. 10.1

Circuit program

The circuit program can be implemented in the way shown in Fig. 10.2.

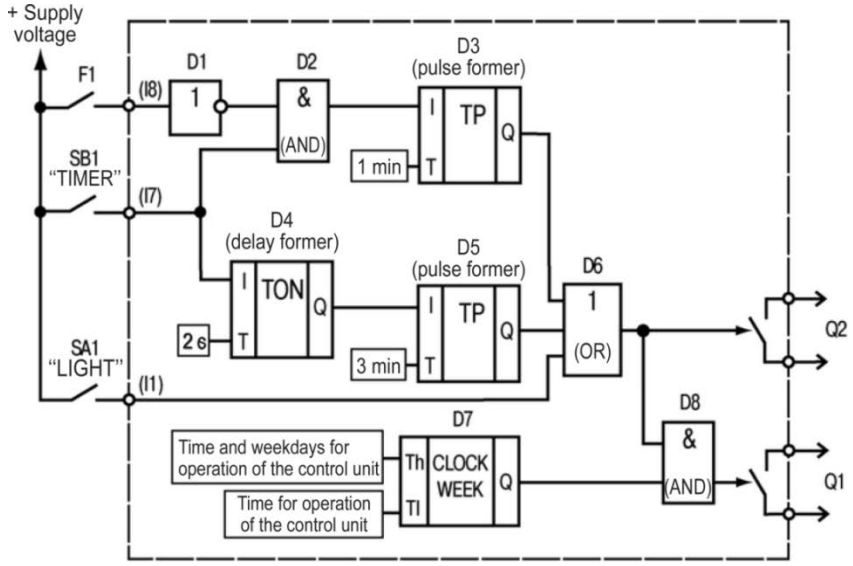


Fig. 10.2

Input I1 – connected to the light sensor F1

Input I2 – connected to the button SB1

Input I3 – connected to the switch SA1

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 is switched 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 is switched on for 3 minutes.
3. If the ambient light is sufficient, the contact of the sensor F1 is closed, 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 is activated 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 Fig. 10.3.

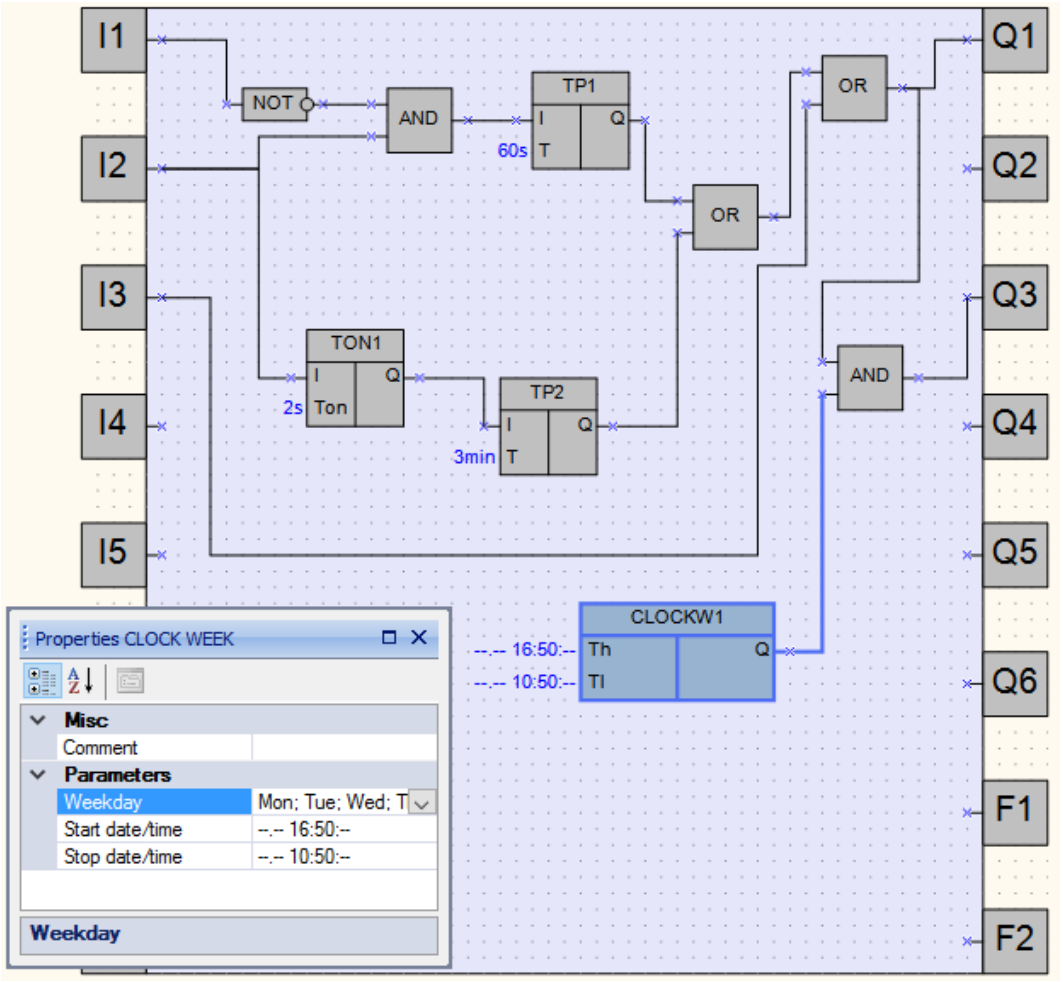


Fig. 10.3

10.2 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 should be switched off automatically, an intermittent acoustic warning signal (HA1) with the 3-second interval should be produced and an operating error should be indicated by the signal lamp HL1 "Overload".
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 PR110 series.

The task implementation with the device PR110-24.8D.4R:

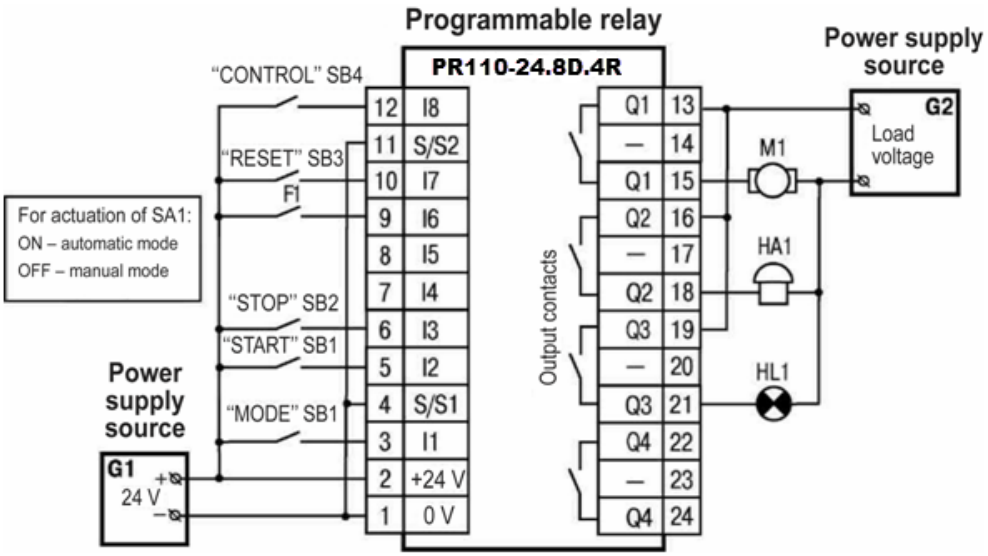


Fig. 10.4

Circuit program

The circuit program can be implemented in the way shown in Fig. 10.5.

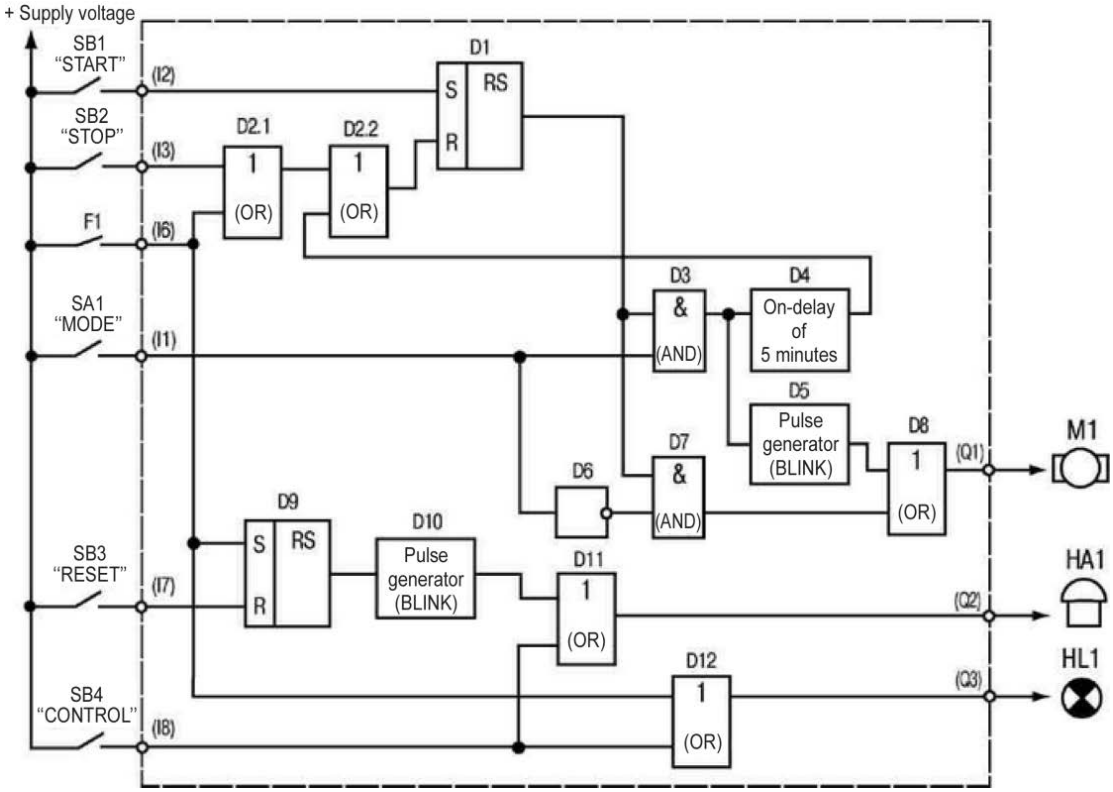


Fig. 10.5

Input I1 – connected to the switch SA1 "MODE"

Input I2 – connected to the button SB1 "START"

Input I3 – connected to the button SB2 “STOP”

Input I6 – connected to the overload switch F1

Input I7 – connected to the button SB3 “RESET”

Input I8 – connected to the button SB4 “TEST”

Output Q1 – connected to the motor

Output Q2 – connected to the acoustic signal HA1

Output Q3 – connected to the signal lamp HL1

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 are switched on.

The circuit program is shown in Fig. 10.6.

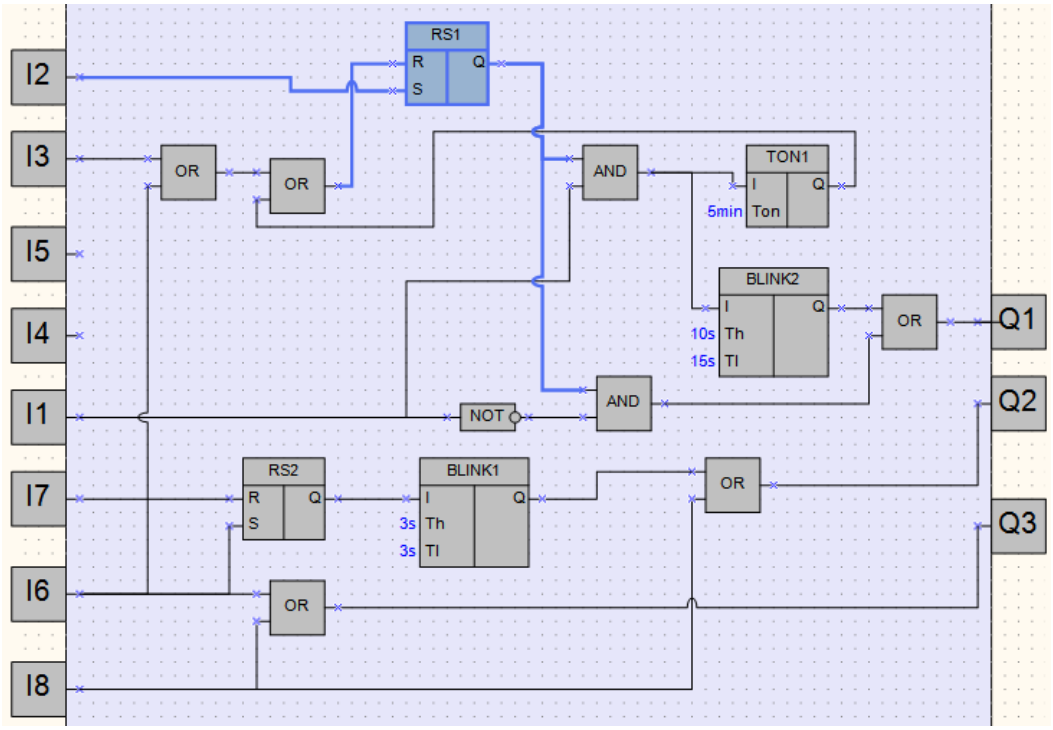


Fig. 10.6

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.