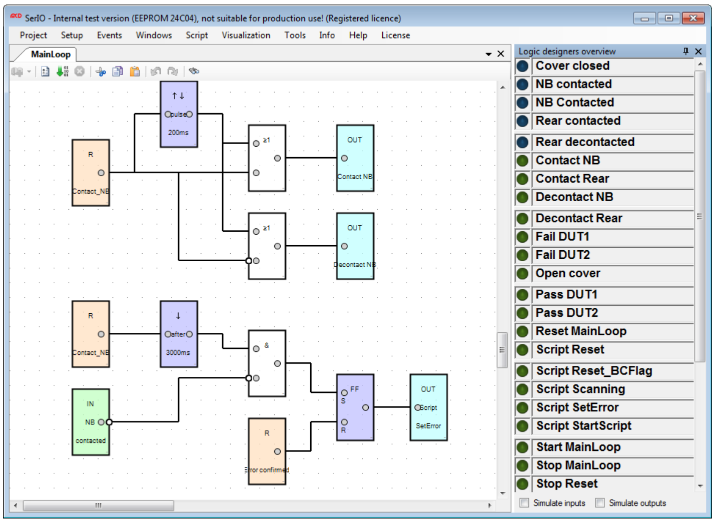
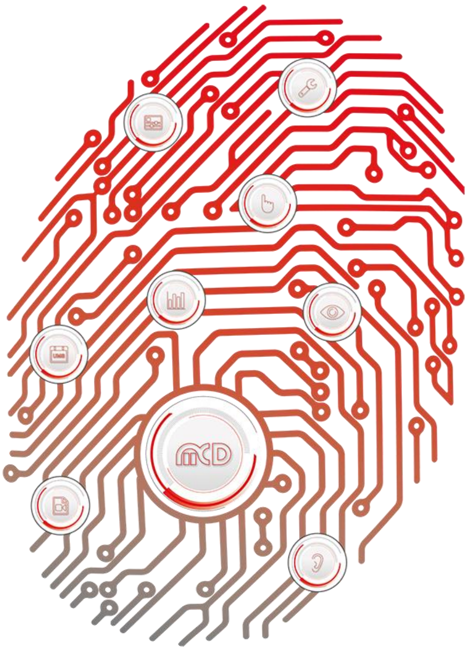
ELEKTRONIK GMBH

GET IN touch WITH SENSITIVE TESTING

Manual

Toolmonitor Logic Designer


SerIO - Internal test version (EEPROM 24C04), not suitable for production use! (Registered licence)



Softline

Modline

Conline

Boardline

Avidline

Pixline

Application

MCD Elektronik GmbH
Hoheneichstr. 52
75217 Birkenfeld
Tel. +49 (0) 72 31/78 405-0
Fax +49 (0) 72 31/78 405-10
info@mcd-elektronik.de
www.mcd-elektronik.com
HQ: Birkenfeld
Managing CEO: Bruno Hörter
Register Court Mannheim
HRB 505692

Template version: 5.4 / 2018-04-09
V1.7 2018-10-09 HM (TS)

Table of Contents

1. General

The Logic Designer permits the graphical programming of inputs and outputs in MCD Toolmonitor setups. General logical relationships between inputs and outputs can be implemented. Here, any value that can be queried in MCD Toolmonitor using the “GetValue“ command can be defined as an input and any value that can be set with the “SetValue“ command can be defined as an output. These inputs and outputs can then be logically combined using the logical components described below. If there are no inputs or outputs available, they can still be simulated for testing. It is also possible to connect the Logic Designer with variables, functions and events from the script engine.

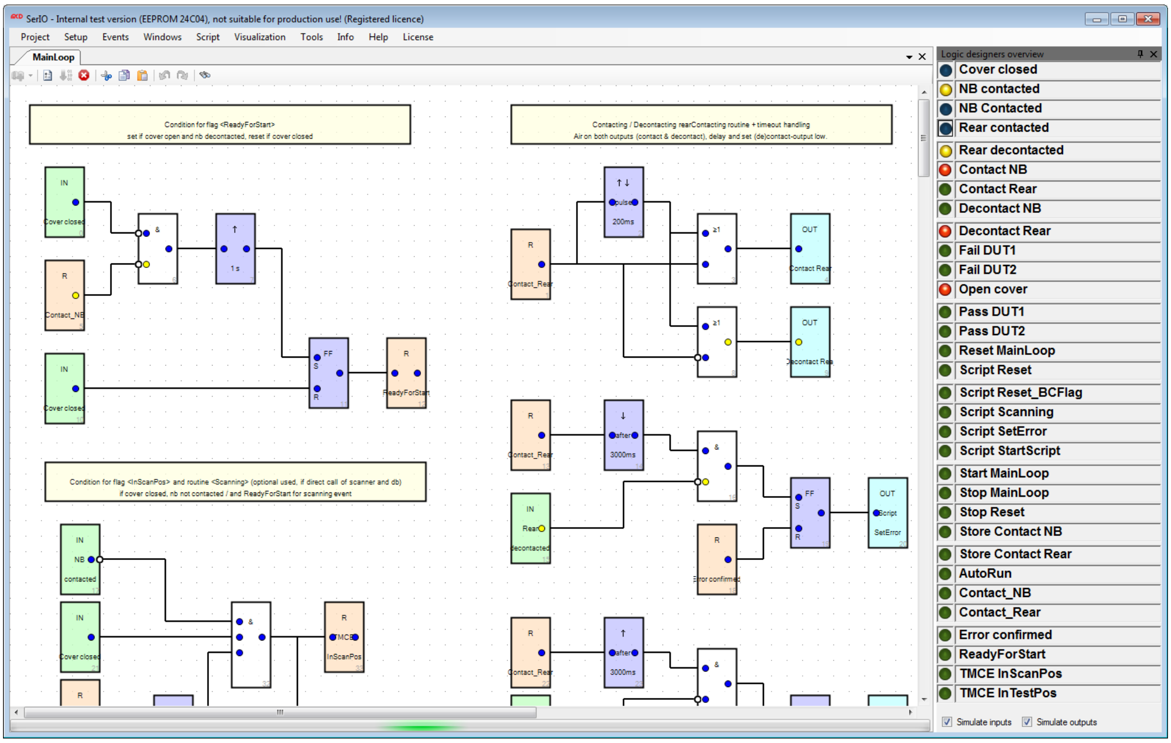


Figure 1: Example of Logic Designer Application

Order number: # 150733

The Logic Designer of the MCD Toolmonitor applications provides a variety of logical elements and standard electronic circuits (such as “FlipFlops“). The logical components “Input“, “Output“, “Global Variable“ and “Script“ also provide seamless integration of logical circuits into MCD Toolmonitor through the “Virtual Interface“.

The following table shows a brief overview of the available logical components.

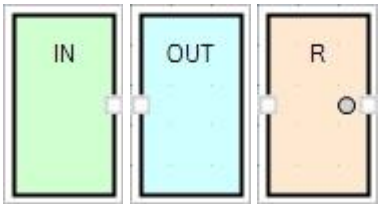
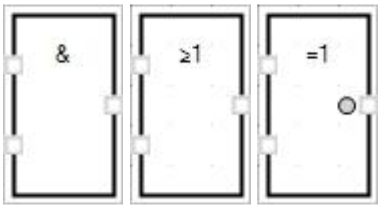
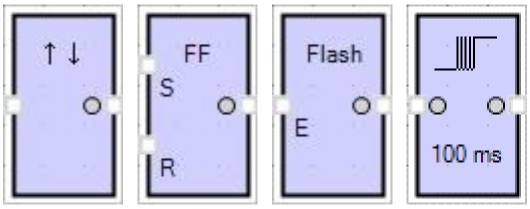
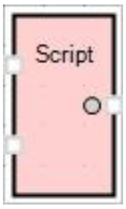
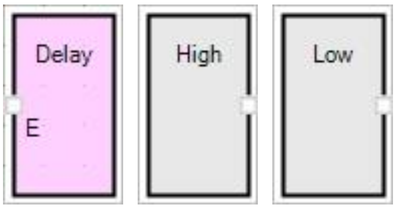
Input, **Output** and **Global Variable** Input, Output and Global Variables permit communication through the *virtual interface* and are used to connect the logical circuit to the accessible values in the Toolmonitor.	
AND, **OR** and **XOR** These gates implement the simplest logical combinations of two (or more) inputs and an output.	
Edge / Pulse gates for edge / pulse detection. **FlipFlop** gates **Flash** component to turn an output on and off periodically. **Debouncer** component to debounce inputs.	
Script component is used for the direct connection of the logic circuit to a method from the script engine and to permit the functionality of the script component to be programmed freely.	
Delay component to include execution delays in the logic circuit. **High** and **Low** provide permanently high and low signals.	
Comment component can be used to embed comments in the circuit.	

Figure 2: Overview of the Provided Logical Components

The functionality and use of the individual logical components are explained in more detail in the section *Logic Elements in Detail*.

2. General Information About the User Interface

A logic design has two basic modes: Design Mode and run / working mode. The creation / editing of a network or circuit is always done in the Design Mode. Testing / execution of the circuit takes place in the working mode. In the run mode, all editing functions are disabled.

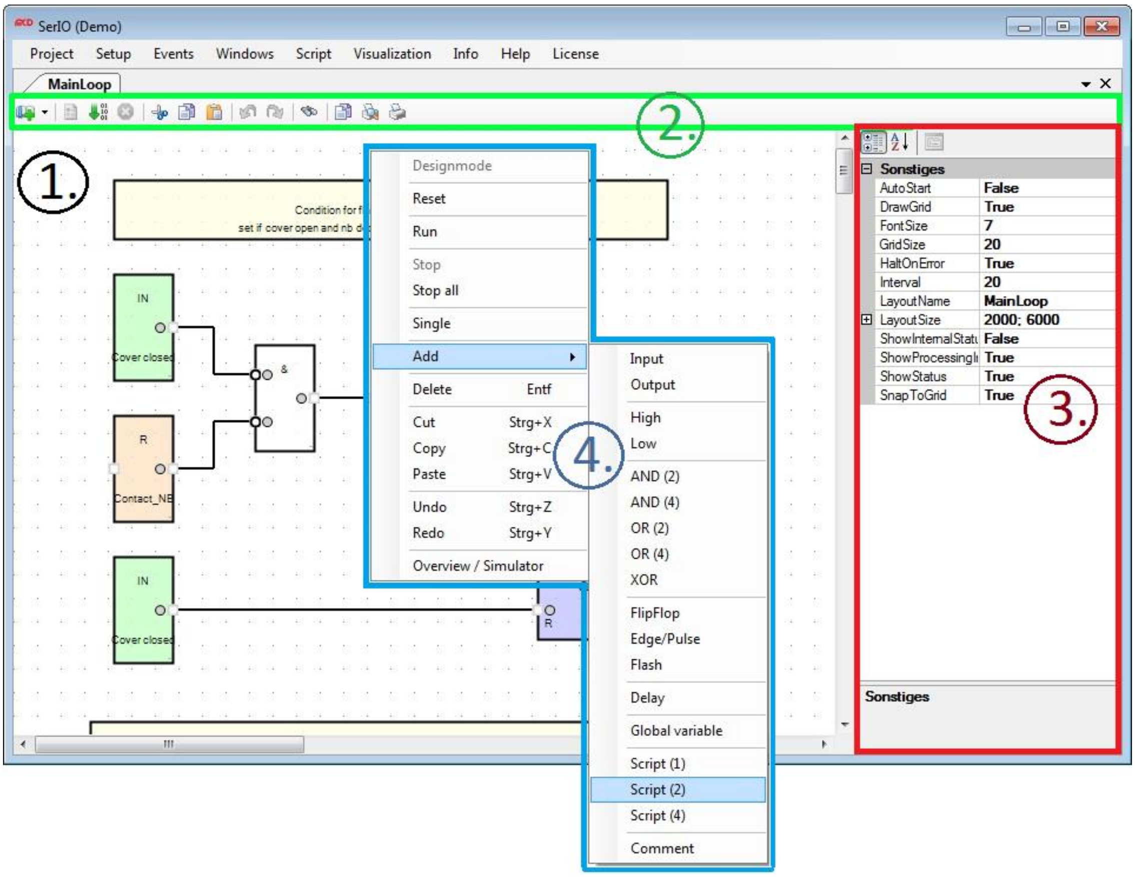
Switching between the two modes takes place using the context menu commands “Design Mode“ and “Run“, or the corresponding toolbar icons and .

2.1. Graphical User Interface Elements in Design Mode

In Design Mode, the work area is divided into:

- Layout area (1.)
- Toolbar (2.)
- Properties window (3.)

The context menu of the layout window (4.) provides extended editing functions.



Sonstiges	
AutoStart	False
DrawGrid	True
FontSize	7
GridSize	20
HaltOnError	True
Interval	20
LayoutName	MainLoop
LayoutSize	2000; 6000
ShowInternalStatu	False
ShowProcessingI	True
ShowStatus	True
SnapToGrid	True

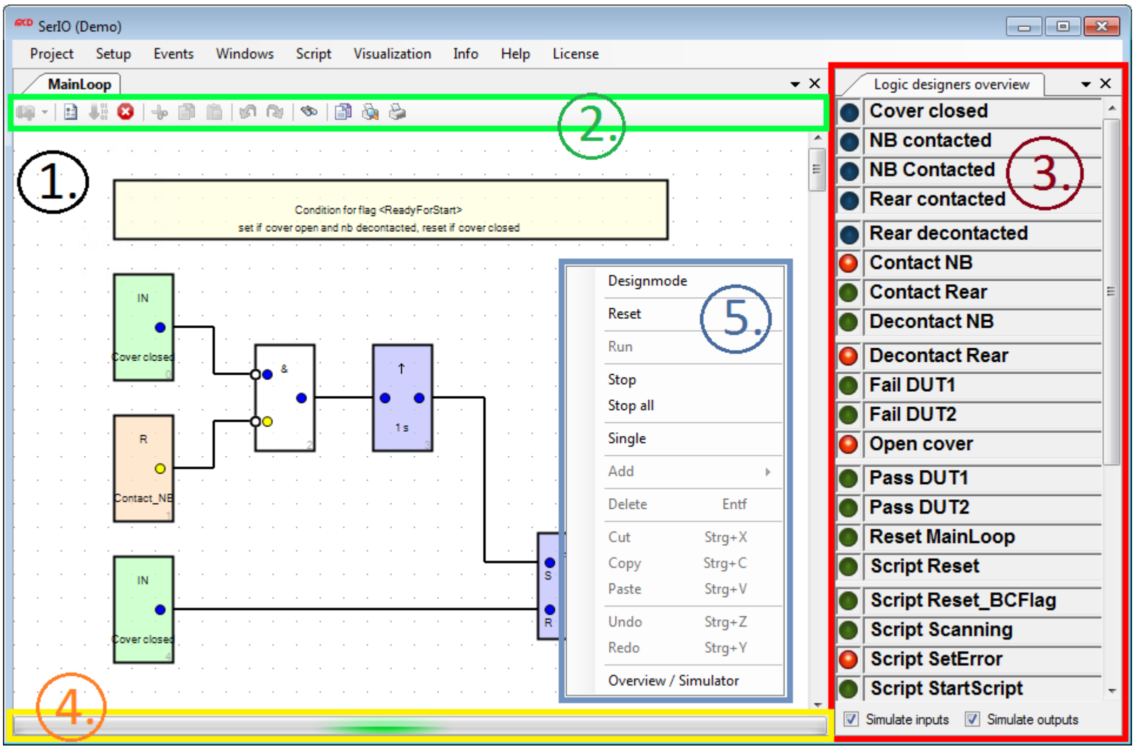
Sonstiges

Figure 3: Overview of the User Interface in the Design Mode

2.2. Graphical User Interface Elements in Run Mode

In the run mode, the work area is divided into:

- Layout area (1.)
- Toolbar (2.)
- Optional window “Logic designers overview“ (3.)
- Progress bar (4.)



Condition for flag <ReadyForStart>
set if cover open and nb decontacted, reset if cover closed

☑ Simulate inputs ☑ Simulate outputs

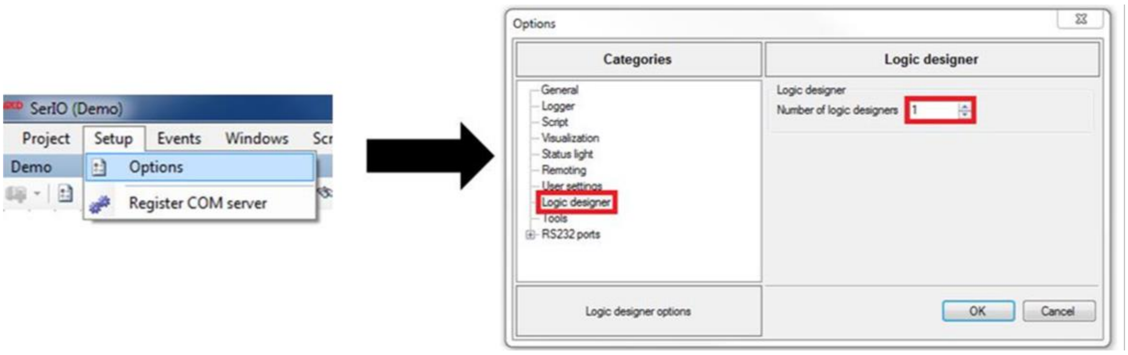
Figure 4: Overview of the User Interface in the Run Mode

Program execution can be seen from the animated progress bar (4.). In the optional window “Logic designers overview“, the current status of all inputs and outputs can be visualized and monitored (3.). All editing functions are disabled in the run mode (5.).

3. Creating a Simple Network

3.1. Creating a Logic Design

To create a new logic design, as shown in the figure below, the number of Logic Designers must be changed in the options for the Toolmonitor to be used. It is also possible to work with multiple designers.



Logic designer

Number of logic designers: 1

Figure 5: Creating the Logic Designer in the Toolmonitor Options

After the Logic Designer has been created in the options, a new menu item now appears under “Script“, named “Logic Designer“. Left - clicking the newly created designer opens it with the default name “Logic Designer1“.

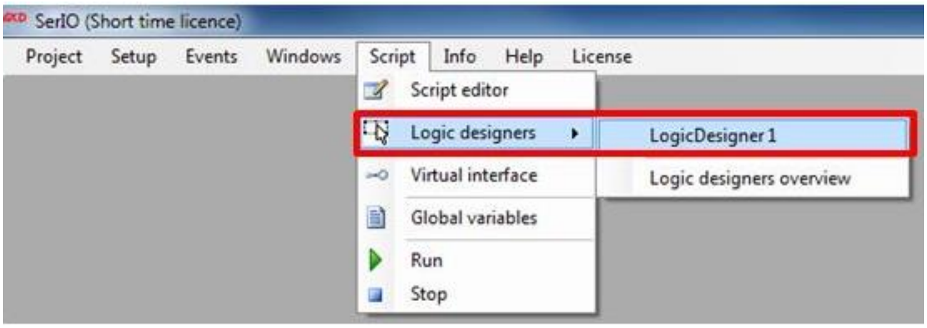


Figure 6: Opening the Logic Designer

When the newly created Logic Designer is opened for the first time, the empty user interface of the designer appears with the grid displayed.

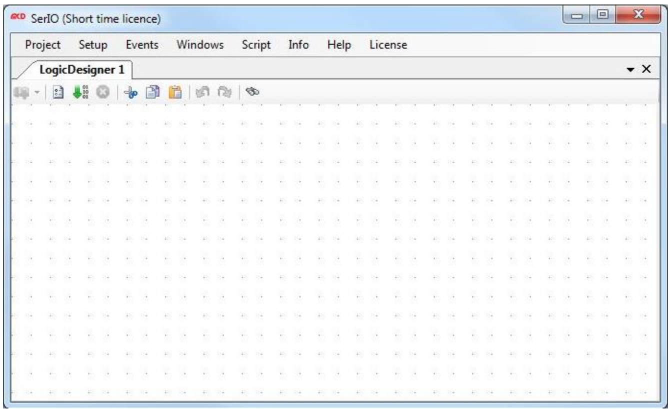


Figure 7: Empty User Interface

3.2. Layout Configuration in Design Mode

To add new components to the user interface or to edit / delete existing components, the user interface must always be switched to Design Mode. You can do this by clicking the context menu item “Design Mode“ or using the button “Design Mode” on the left edge of the toolbar.

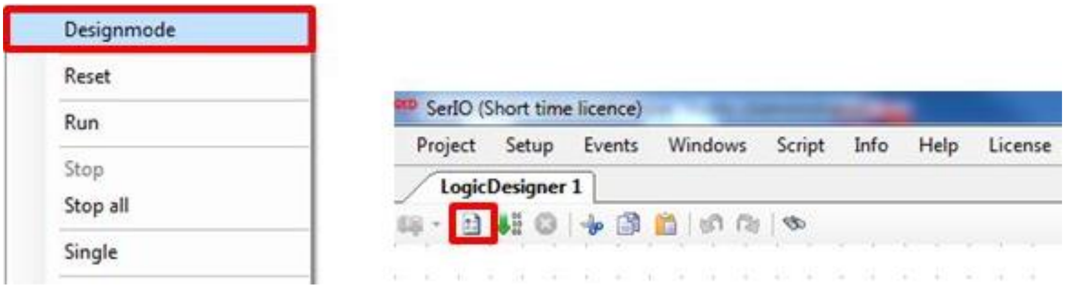
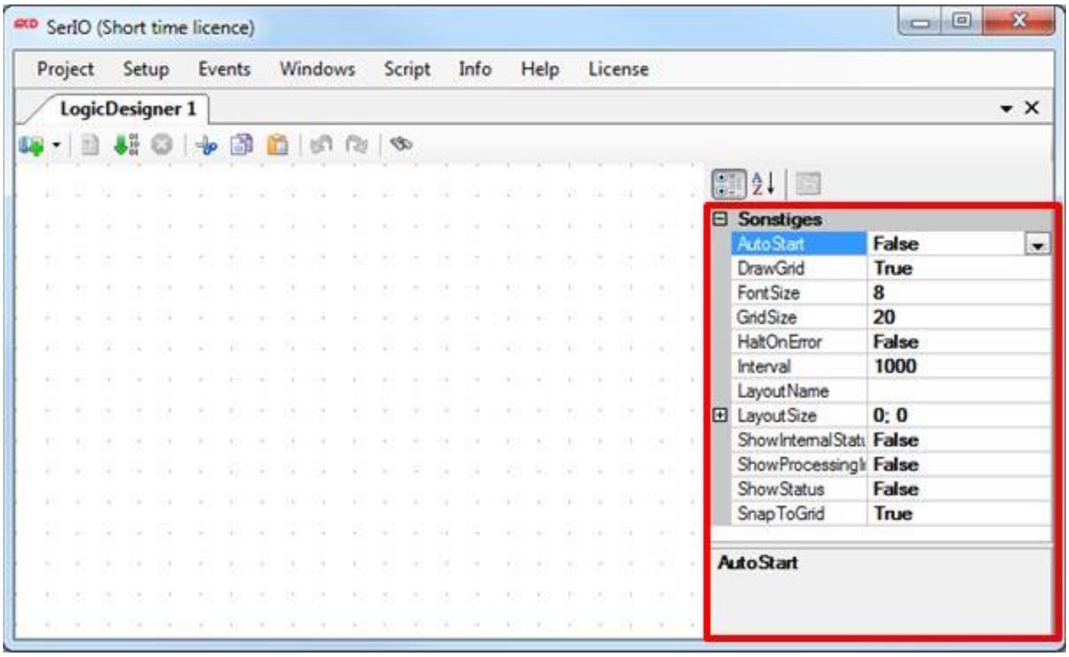


Figure 8: Activating the Design Mode

After switching to the Design Mode, a table appears on the right edge of the screen with configuration options for the user interface.



Sonstiges	
AutoStart	False
DrawGrid	True
FontSize	8
GridSize	20
HaltOnError	False
Interval	1000
LayoutName	
LayoutSize	0; 0
ShowInternalStatı	False
ShowProcessingI	False
ShowStatus	False
SnapToGrid	True

AutoStart

Figure 9: Settings of the User Interface

The following table shows a brief overview of the available settings:

Property / setting	Description
Autostart	Definition whether the content of the Logic Designer should be started automatically when the Toolmonitor / a preset is loaded
DrawGrid	Selection whether the grid should be shown
FontSize	Configuration of font size
GridSize	Configuration of grid spacing
HaltOnError	Definition whether execution should stop on an error / exception
Interval	Configuration of time in milliseconds in which the cycle should repeat
LayoutName	Configuration of layout name (this also changes the window title)
LayoutSize	Configuration of the size of the work (if “0“ is selected, the entire window is used)
ShowInternalStatus	Decision whether the logical state of inputs / outputs or the status on the line should be displayed
ShowProcessingIndex	Displaying the processing order of the individual elements
ShowStatus	Displaying the status of the individual elements
SnapToGrid	Configuration whether components should snap to the grid when being moved

Figure 10: Overview of the Available Configuration Possibilities

There are additional functions for the Logic Designer in the context menu. To open the context menu, right - click the mouse in a free space in the designer.

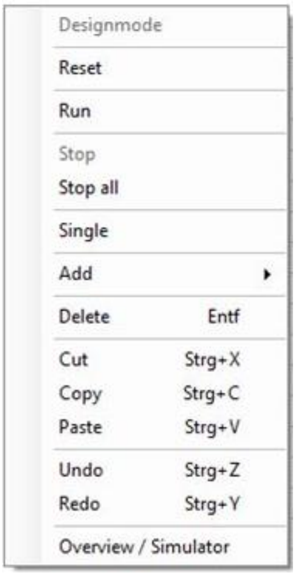


Figure 11: Context Menu of the Logic Designer

Most of the items on the context menu are only available when the Designer is in the Design Mode.

The following table gives a short overview of the items on the context menu. Some of them can also be reached from the toolbar on the upper edge of the screen.

Context menu	Toolbar	Function
Design Mode	📄	Change to Design Mode (only in run mode)
Reset		Reset the logical operations
Run	⬇ 01 10 01	Start the logic circuit
Stop	✖	Stop the logic circuit (only in run mode)
Stop all		Stop all existing forms in the Logic Designer
Single		Execute the entire cycle once
Add	▾	Add new elements
Delete		Delete elements
Cut	✂	Cut elements
Copy	📋	Copy selected elements into the buffer
Paste	📋	Insert the element from the buffer
Undo	↶	Undo the last action
Redo	↷	Repeat the last action
Overview / Simulator	🔭	Open the “Logic designers overview“ window
Copy image	📋	Copy a logic design (or a section of a logic designs) and insert as an image into the buffer
Print preview	🔍	Open the dialog window “print preview“
Print	🖨	Send a logic design to the standard printer

Figure 12: Overview of the Items on the Context Menu

3.3. Adding and Editing Logic Elements

The addition of new elements can either be done from the context menu item “Add“ or using the button with the book icon on the left edge of the toolbar. Note that the Logic Designer must be in Design Mode, since the “Add“ function is only available during design.

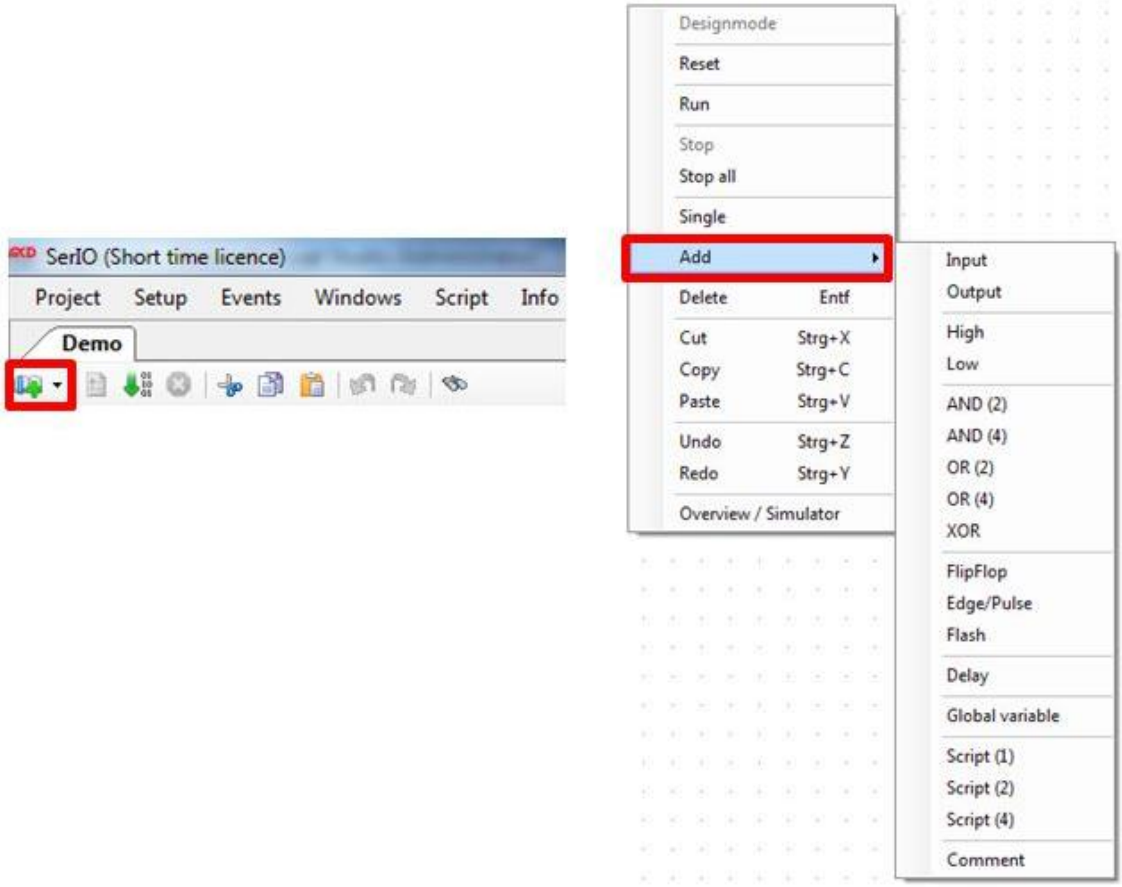
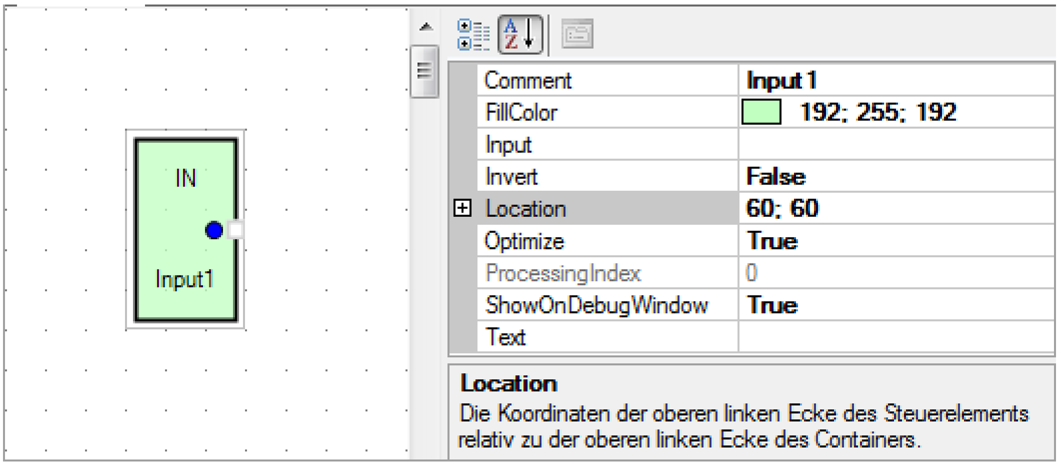


Figure 13: Adding Logic Elements

The appearance and behavior of the added logic components can be adjusted on the properties window. The properties window shows the properties of the currently selected elements. For example, the setting “FillColor“ can be used to change the color of the component.



Comment	Input 1
FillColor	🟩 192; 255; 192
Input	
Invert	False
⊞ Location	60; 60
Optimize	True
ProcessingIndex	0
ShowOnDebugWindow	True
Text	

Location
Die Koordinaten der oberen linken Ecke des Steuerelements relativ zu der oberen linken Ecke des Containers.

Figure 14: Properties of the Selected Logic Component

The “Location“ property can be used to determine the position of the component in the X and Y directions. The component can also be simply dragged to the desired position on the user interface by holding the left mouse button down.

3.4. Connecting Logic Elements

After the elements are added, they must still be connected together. To do this, click the desired white - marked input / output of the one element and connect it to the desired input / output of the other element by left - clicking again. If a connection between two points should be deleted again, you can click on one of the connected points again and then on the empty white space in the user interface. Note that the creation and deletion of connections between logic elements is only possible in Design Mode.

3.5. Deleting Existing Logic Elements

To remove entire logic elements, the corresponding element must be selected and can then be deleted either with the “Del“ key or with the context menu item “Delete“.

4. Testing / Running / Debugging the Circuit in Run Mode

After the logic circuit has been created in Design Mode, it can then be executed with the “Run“ command. This can be started from the context menu of the Logic Design or from the toolbar. This automatically switches the Logic Designer to run mode.

Run mode or execution of the circuit is symbolized by the animated progress bar in the lower part of the logic design. The following functions are available in the run mode:

Context menu	Toolbar	Function
Design Mode	📄	Stop execution of the logic circuit and switch to Design Mode
Reset		Reset the logical operations
Stop	⊗	Stop the logic circuit
Stop all		Stop all existing forms in the Logic Designer
Single		Execute the entire cycle once
Overview / Simulator	🔭	Open the “Logic designers overview“ window

Figure 15: Functions of the Run Mode

In the run mode, all editing functions are disabled.

In the run mode, the “Logic designers overview“ provides an overview of the inputs and outputs of the circuit. As described above, this can be opened using the context menu item “Overview / Simulator“ or the telescope icon on the toolbar. If the inputs or outputs are not connected to a signal, they can also be simulated for testing. To do this, the two checkboxes “Simulate inputs“ and / or “Simulate outputs“ on the lower edge of the “Logic designers overview“ must be checked.

The example below shows two inputs connected by an AND gate to an output. The inputs "Input1" and "Input2" are now displayed in the “Logic designers overview“. “Input1“ is “High“ (yellow light), “Input2“ is “Low“ (blue light) and thus “Output1“ is “Low“ (green light). Both checkboxes are checked to simulate both inputs and outputs.

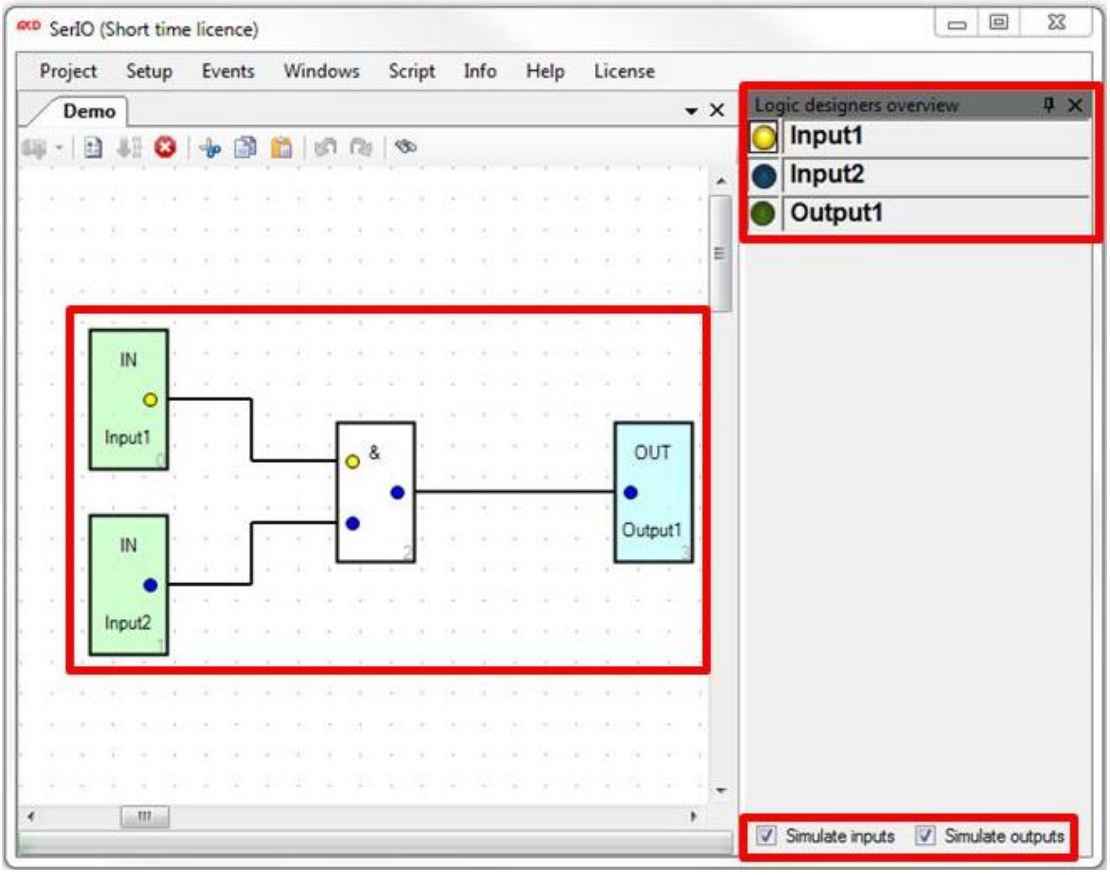
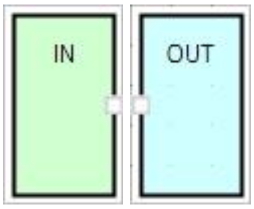
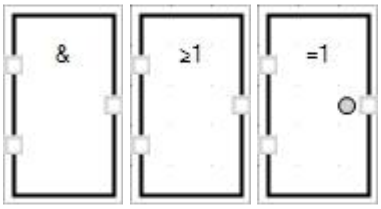
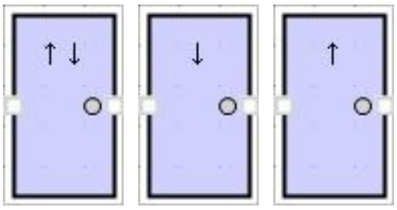
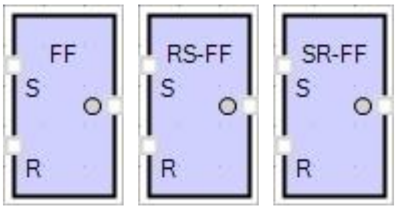
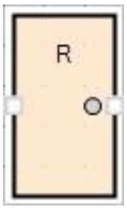


Figure 16: Example of Logic Designer Overview

5. Logic Elements in Detail

The Logic Designer in the MCD Toolmonitors provides a variety of logical elements, a few controls with extended functionality and a few controls for implementing the *virtual interface*.

The following table shows an overview of the available logical components:

Input and **Output** Input and output permit communication through the *virtual interface* and are used to connect the logical circuit to the accessible values in the Toolmonitor. The input can be connected to any value in the Toolmonitor that can be accessed using "Get" methods. The otput can be connected to any value in the Toolmonitor that can be accessed using "Set" methods.	
AND, **OR** and **XOR** These gates implement the simplest logical combinations of two (or more) inputs and an output. AND and OR gates are available with two to four inputs.	
Edge / Pulse gates for edge / pulse detection.	
FlipFlop gates with or without prioritization of set / reset inputs.	
Global Variable This control permits communication via the virtual interface.	

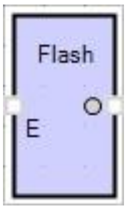
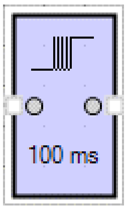
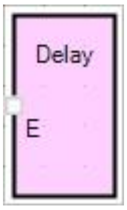
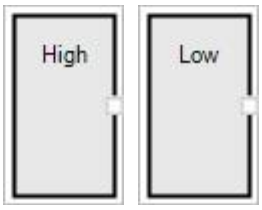
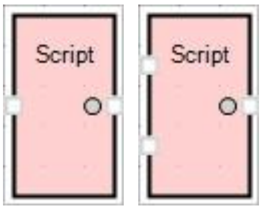
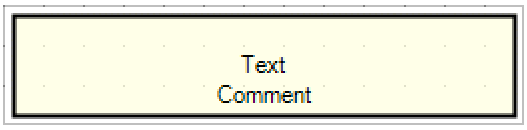
Flash Component This control can be used to turn an output on and off at configurable intervals of time.	
Debouncer Component This component is used to debounce inputs. The debouncing time can be configured arbitrarily.	
Delay Component Using the Delay Component, execution delays can be built into the logic circuit.	
High and **Low** These components provide permanent high and low signals and can be used, for example, in reset or initialization circuits to set an input to a defined state.	
Script Component The Script Component is used for the direct connection of the logic circuit to a method from the script engine and permits the user to create components with specific functionality.	
Comment The Comment control is only used to embed comments in the circuit and has no other functionality.	

Figure 17: Available Logic Components

5.1. General Properties

Logic controls can be configured using the properties window.

All logic controls have a common set of properties: “Location“, “FillColor“, “Comment“, “Text“ and “ProcessingIndex“.

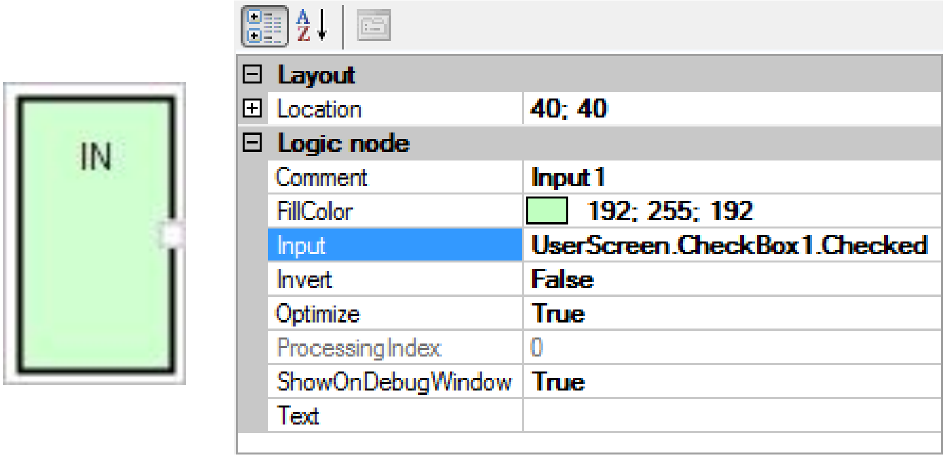
- “Location“: Determination of the position of the component in the X and Y directions. Alternatively, the component can also be simply dragged to the desired position on the user interface by holding the left mouse button down.
- “FillColor“: Configuration of the element‘s color
- “Comment“: Change the text in the upper part of the component
- “Text“: Change the text in the middle part of the component
- “ProcessingIndex“: Turning on and off the display of the processing order in which the nodes are processed on each run.

Otherwise, the configurable properties differ depending on the type and function of the logic control. The following sections cover the control - specific properties.

5.2. Logic Element INPUT

The most important components in a logic circuit are the input and output These logic elements must exist in every circuit. To add an input, click the "Add" menu item in the context menu of the logic design, then the “Input“ menu item**.** Note that the Logic Designer must be in the Design Mode, since the "Add" function is only available during design.

If the newly added input is selected in Design Mode, an overview of the configuration options for the component opens.

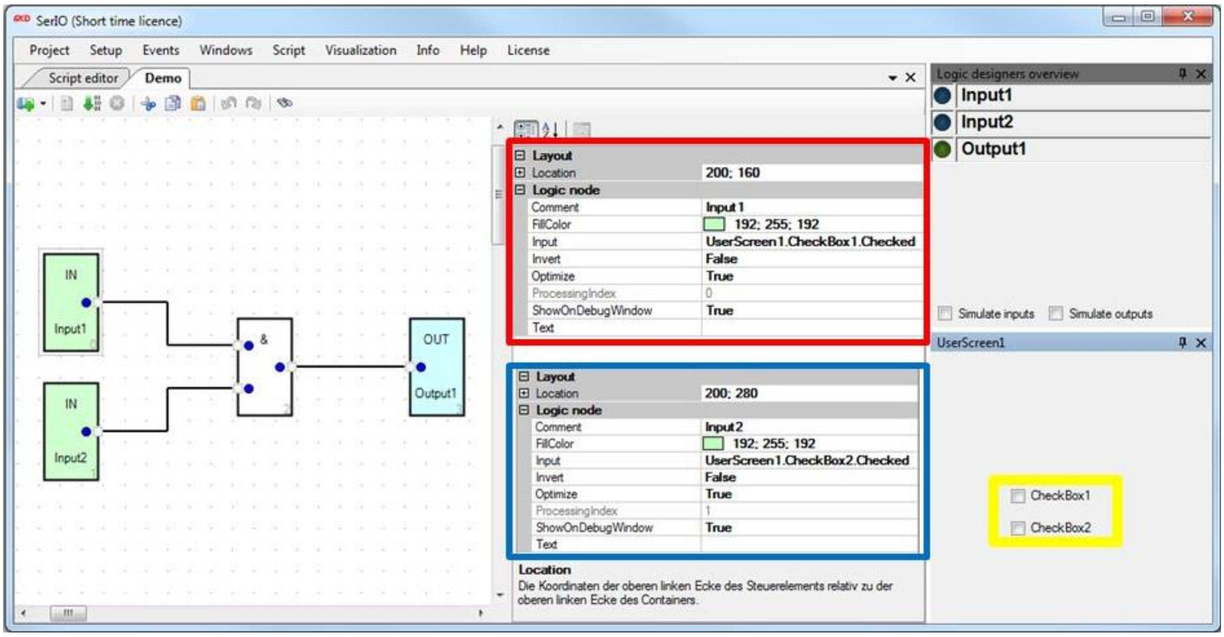


⊟ **Layout**	
⊞ Location	**40; 40**
⊟ **Logic node**	
Comment	**Input 1**
FillColor	🟩 **192; 255; 192**
Input	**UserScreen.CheckBox1.Checked**
Invert	**False**
Optimize	**True**
ProcessingIndex	0
ShowOnDebugWindow	**True**
Text	

Figure 18: Input Control and its Properties

The “Input“ property is used to select which attribute of the “GetValue“ function will be used as the input.

In the example below, the status of the “CheckBox1“ created previously on the “UserScreen“ is connected to “Input1“, and the status of a “Checkbox2“ is connected to “Input2“. To do this, in the“Input“ properties the property of the “CheckBox“ is specified using “UserScreen1.CheckBox1.Checked“ (marked in red) or “UserScreen1.CheckBox2.Checked“ (marked in blue). To test this setup, the “Run“ command on the context menu is used to start the logic circuit. If “CheckBox1“ is checked, “Input1“ goes to “High“. If “CheckBox2“ is checked, “Input2“ goes to „High“. The checkboxes created are marked in yellow. If there is not source available for an input during testing, it can also be simulated by checking the “Simulate inputs“ box in the “Logic Designers Overview“.



⊟ **Layout**	
⊞ Location	200; 160
⊟ **Logic node**	
Comment	Input 1
FillColor	🟩 192; 255; 192
Input	UserScreen1.CheckBox1.Checked
Invert	False
Optimize	True
ProcessingIndex	0
ShowOnDebugWindow	True
Text	

⊟ **Layout**	
⊞ Location	200; 280
⊟ **Logic node**	
Comment	Input 2
FillColor	🟩 192; 255; 192
Input	UserScreen1.CheckBox2.Checked
Invert	False
Optimize	True
ProcessingIndex	1
ShowOnDebugWindow	True
Text	

Location
Die Koordinaten der oberen linken Ecke des Steuerelements relativ zu der oberen linken Ecke des Containers.

Figure 19: Example of Connecting Inputs to Checkboxes

The “Invert“ property allows you to invert the input. If the “Invert“ property is set to “True“, the input is “High“ whenever its source is “Low“, and vice versa.

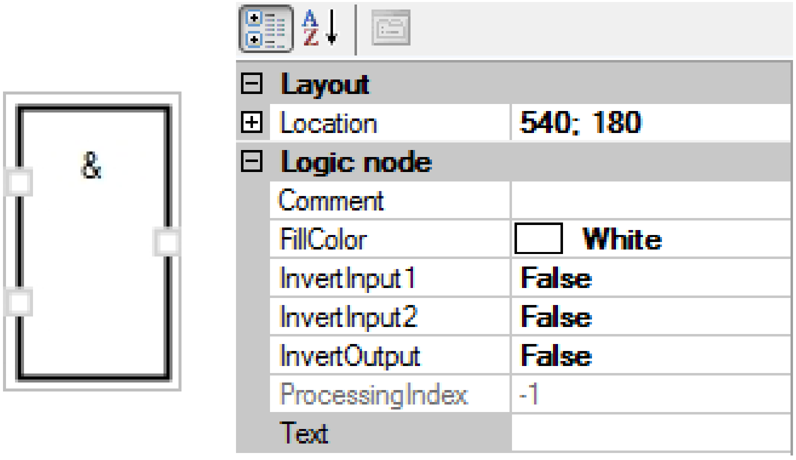
The “Optimize“ property can be used to tell the Logic Designer not to query the input for each new node it processes, but rather to remember the status of the input for each iteration. When an input is reused several times, this can have a significant advantage in terms of performance of the logic circuit.

The “ShowOnDebugWindow“ property can be used to determine whether the component should be displayed in the “Logic Designers Overview“ or not.

5.3. Logic Element AND

An AND gate combines multiple inputs with a single output. To add an AND gate, in the context menu of the logic design click on the submenu “Add“ and then in the “AND(2)“ menu item to add an AND gate with two inputs, or “AND(4)“ to add an AND gate with four inputs. Note that the Logic Designer must be in the Design Mode, since the “Add“ function is only available during design.

If the newly added AND gate is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for an AND gate with four inputs. For an AND gate with two inputs, the two settings for inverting inputs three and four are omitted. Otherwise, the components have identical behavior. If an input on an AND gate with four gates is not used, then this is not taken into consideration when switching the output.



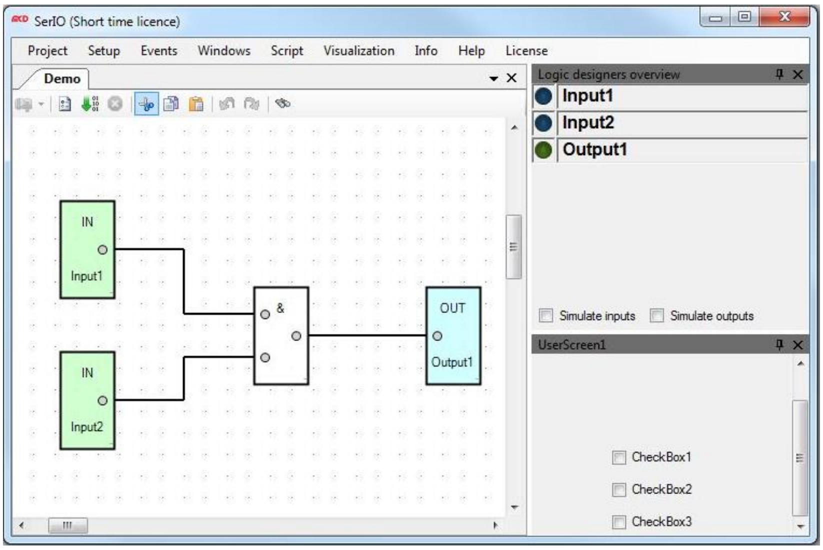
Layout	
Location	540; 180
Logic node	
Comment	
FillColor	□ White
InvertInput1	False
InvertInput2	False
InvertOutput	False
ProcessingIndex	-1
Text	

Figure 20: AND Gate and Available Settings

The “InvertInput1/2/3/4“ properties can be used to invert the different inputs of the component. The topmost input of the component is “Input1“, and the lowest “Input2“.

The “InvertOutput“ property can be used to invert the output of the gate.

The example below shows an AND circuit with two inputs. The inputs are named “Input1“ and “Input2“, and the output is “Output1“. The inputs and outputs are then displayed in the “Logic Designers Overview“ in the upper right corner under these names.



Logic designers overview

- Input1
- Input2
- Output1

□ Simulate inputs □ Simulate outputs

□ CheckBox1

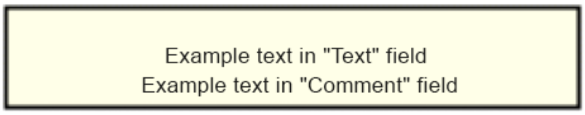
□ CheckBox2

□ CheckBox3

Figure 21: Example of an AND Gate with two Inputs

5.4. Logic Element Comment

The Comment control is used to embed comments in the circuit and has no other functionality.



Example text in "Text" field
Example text in "Comment" field

Figure 22: Example of a Comment Component

The Comment component can be used to add a comment field to the user interface. To add such a comment field, click the “Comment“ menu item in the “Add“ menu. Note that the Logic Designer must be in the Design Mode, since the “Add“ function is only available during design.

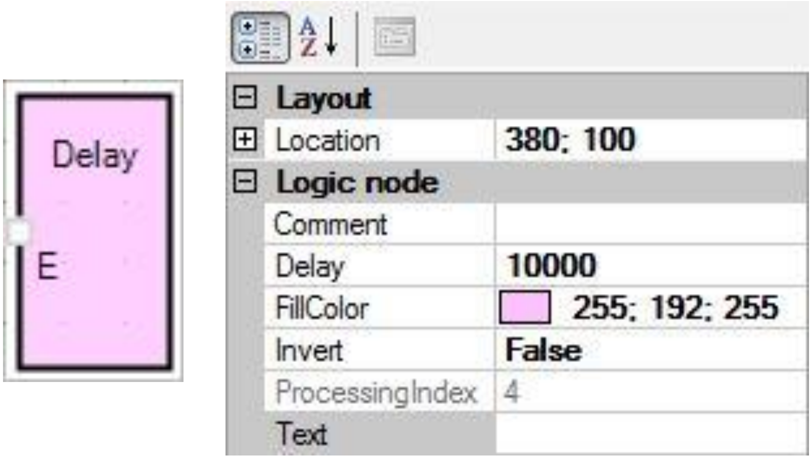
The “Size“ property can be used with width and height to adjust the size of the component. The sizes are specified in pixels. The “Comment“ property can be used to change the text in the upper part of the component. The “Text“ property can be used to change the text in the middle part of the component.

5.5. Logic Element Delay

Using the Delay component, execution delays can be built into the logic circuit.

The “Delay” menu item can be used to add a delay component. If a “High“ signal is on this input, it blocks the entire logic circuit for a certain configurable period of time. To add such a delay, click the “Add“ menu item in the context menu of the logic design, then the “Delay“ menu item. Note that the Logic Designer must be in the Design Mode, since the “Add“ function is only available during design and not in run mode.

If the newly added delay component is selected in Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for a delay component.



⊟ **Layout**	
⊞ Location	**380; 100**
⊟ **Logic node**	
Comment	
Delay	**10000**
FillColor	□ **255; 192; 255**
Invert	**False**
ProcessingIndex	4
Text	

Figure 23: Delay Component and Settings

The “Delay“ property specifies the time in milliseconds for which the logic circuit will be blocked when “High“ is applied to its input.

If the “Invert“ property is set to “True“, the state on the input of the “Delay“ component is inverted.

In the example below, the two inputs “Input1“ and “Input2“ are ANDed together and connected to an output. The “Delay“ component included provides an appropriate delay.

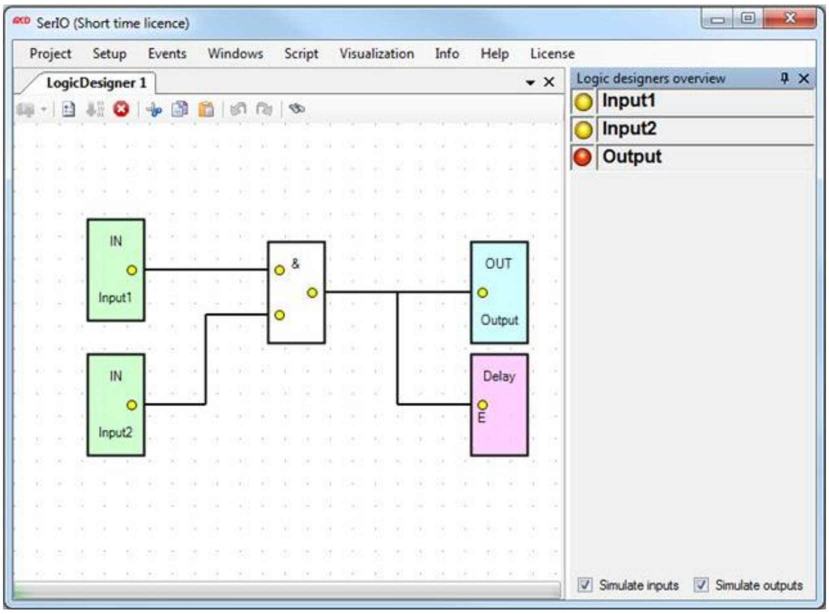


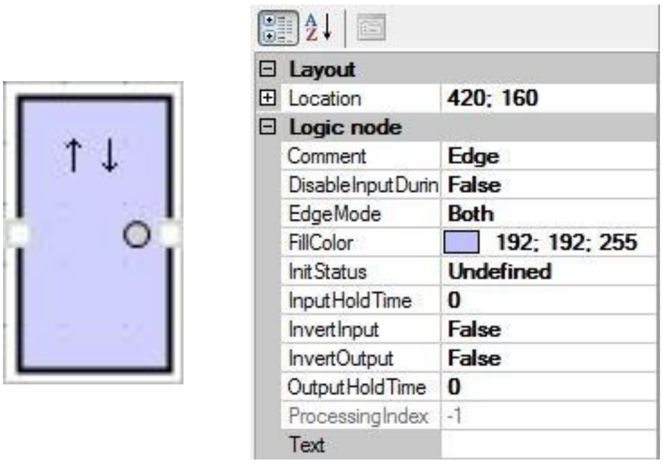
Figure 24: Example of a Circuit with a Delay Component

5.6. Logic Element EDGE / PULSE

The Edge / Pulse component reacts to changing edges by emitting a pulse.

To add an Edge / Pulse component, click the “Add“ menu item in the context menu of the logic design, then the “Edge / Pulse“ menu item. Note that the Logic Designer has to be in the Design Mode, since the “Add“ function is only available during design.

If the newly added Edge / Pulse component is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for an Edge / Pulse component.

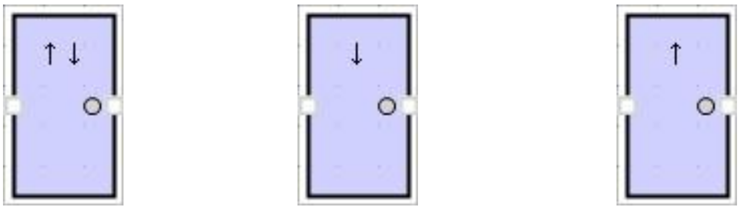


⊟ **Layout**	
⊞ Location	**420; 160**
⊟ **Logic node**	
Comment	**Edge**
DisableInputDurin	**False**
EdgeMode	**Both**
FillColor	▭ **192; 192; 255**
InitStatus	**Undefined**
InputHoldTime	**0**
InvertInput	**False**
InvertOutput	**False**
OutputHoldTime	**0**
ProcessingIndex	-1
Text	

Figure 25: Edge / Pulse Component and Settings

The “DisableInputDuringOutput“ property can be used to disable the input of the component as long as the output is “High“.

The “EdgeMode“ property can be used to specify which edge the component should react to. If “Raising“ is set, the component reacts to rising edges. When a rising edge is detected, a pulse is emitted on the output of the Edge component. If “Falling“ is set, the component reacts to falling edges. If the component should trigger on both falling and rising edges, the “EdgeMode“ must be set to “Both“.



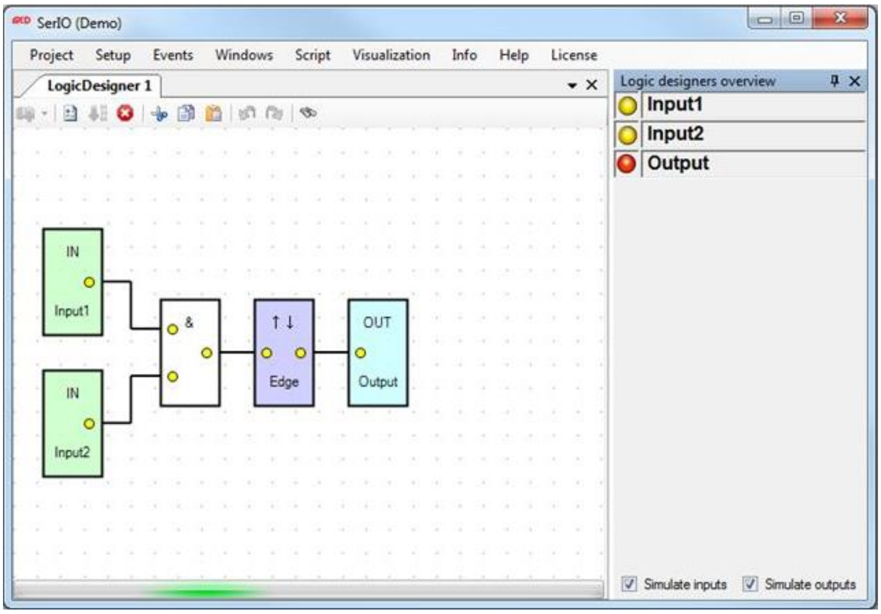
EdgeMode = Both EdgeMode = Falling EdgeMode = Raising

Figure 26: Edge / Pulse Components in Different Modes

“InitStatus“ specifies the state of the output of the component when the logic circuit is initialized. The possible options are “High“, “Low“ and “Undefined“.

The “InputHoldTime“ determines how long the input of the component must be “High“ to trigger a pulse on the output. The time is specified in milliseconds. If this time is “0“, the pulse is emitted as soon as the edge is detected, so at latest by the next execution cycle. The “InvertInput“ property can be used to invert the input of the component. The „InvertOutput“ property can be used to invert the output of the component. The “OutputHoldTime“ specifies the length of the pulse emitted, in milliseconds. If this time is “0“, the pulse will last as long as one run through the logic circuit.

In the example below, the two inputs “Input1“ and “Input2“ are ANDed together and connected to an output through an Edge / Pulse component. Once both inputs are “High“, the Edge / Pulse component detects a rising edge and emits a pulse on the output. If one of the inputs goes “Low“again, the input to the Edge / Pulse components sees a “Low“ signal, the component detects the falling edge, and it emits another pulse on its output. The settings for the component are set as shown above.



○ Input1
○ Input2
● Output

☑ Simulate inputs ☑ Simulate outputs

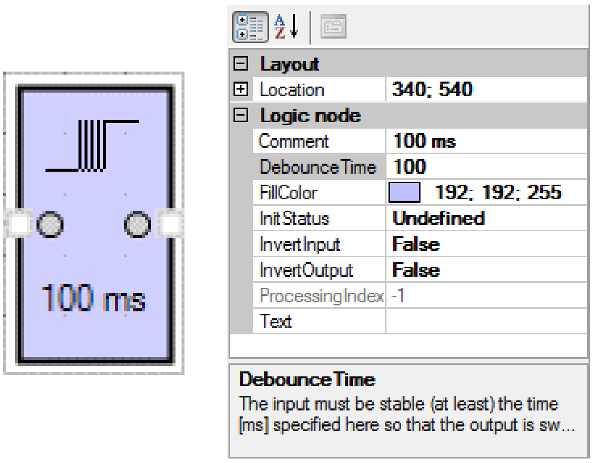
Figure 27: Example of a Circuit with an Edge / Pulse Component

5.7. Logic Element Debouncer

The Debouncer component is used to debounce inputs and can be used for the reliable switching of buttons and switches. The debounce time can be arbitrarily configured with the “DebounceTime“ setting, in milliseconds.

To add a Debouncer component click the “Add“ menu item in the context menu of the logic design, then the “Debouncer“ menu item. Note that the Logic Designer has to be in the Design Mode, since the “Add“ function is only available during design.

If the newly added Debouncer component is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for a Debouncer component.



Layout	
Location	340; 540
Logic node	
Comment	100 ms
DebounceTime	100
FillColor	▭ 192; 192; 255
InitStatus	Undefined
InvertInput	False
InvertOutput	False
ProcessingIndex	-1
Text	

DebounceTime
The input must be stable (at least) the time [ms] specified here so that the output is sw...

Figure 28: Debouncer Component Settings

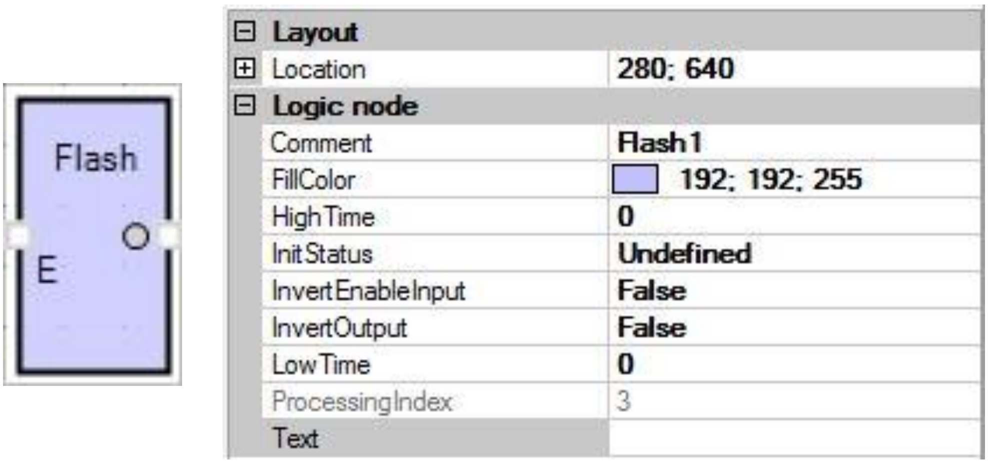
The “DebounceTime“ property specifies how long the signal must be applied to the input of the component without changing before it is sent to the output. The debounce time is specified in milliseconds.

The “InitStatus“ specifies the state of the output of the component when the logic circuit is initialized. The possible options are “High“, “Low“ and “Undefined“. The “InvertInput“ property can be used to invert the input of the component. The “InvertOutput“ property can be used to invert the output of the component.

5.8. Logic Element Flash

The “Flash“ menu item can be used to add a component which, when its input is “High“, turns its output on and off at configurable intervals. In short, the output of the component “flashes“. To add a Flash component, click the “Add“ menu item in the context menu of the logic design, then the “Flash“ menu item. Note that the Logic Designer has to be in the Design Mode, since the “Add“ function otherwise cannot be selected.

If the newly added Flash component is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for a Flash component.



⊟ **Layout**	
⊞ Location	280; 640
⊟ **Logic node**	
Comment	Flash1
FillColor	▭ 192; 192; 255
HighTime	0
InitStatus	Undefined
InvertEnableInput	False
InvertOutput	False
LowTime	0
ProcessingIndex	3
Text	

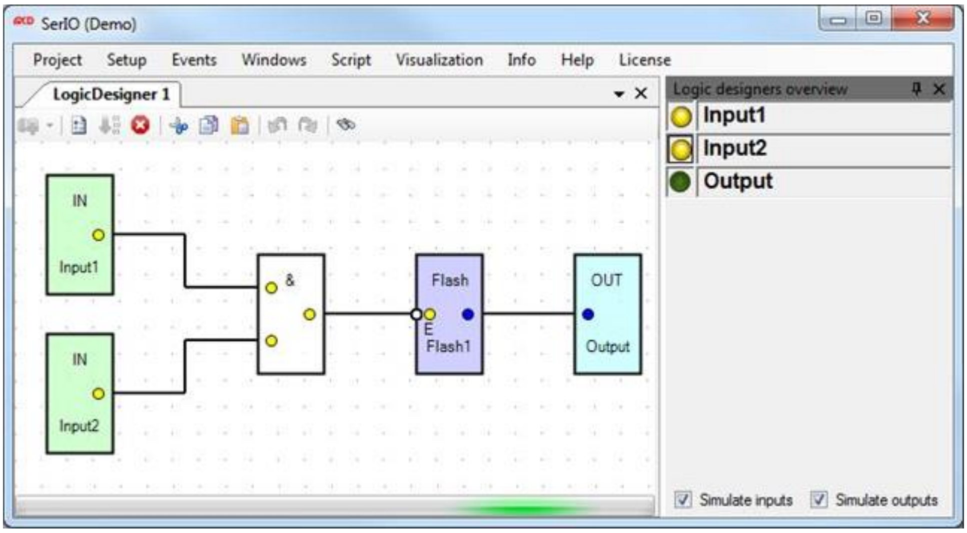
Figure 29: Flash Component and Settings

The “HighTime“ property sets the time in milliseconds for which the “Flash“ component sets its output to “High“ when there is a “High“ signal on its input. If a “0“ appears here, the component holds its output “Low“ for exactly one cycle of the logic circuit, before switching it back to “High“.

The „InitStatus“ property specifies the state of the output of the component when the logic circuit is initialized. The possible options are “High“, “Low“ and “Undefined“. The “InvertEnableInput“ property can be used to invert the input of the component. The “InvertOutput“ property can be used to invert the output of the component.

The “LowTime“ property sets the time in milliseconds for which the Flash component sets its output to “Low“ when there is a “High“ signal on its input. If a “0“ appears here, the component holds its output “Low“, for exactly one cycle of the logic circuit, before switching it back to “High“.

In the example below, the two inputs „Input1“ and „Input2“ are ANDed together and connected to an output through a Flash component. Once “Input1“ and “Input2“ are “High“, then a “High“ signal is also applied to the input of the Flash component, which then switches back and forth between “High“ and “Low“.



Input1
Input2
Output

☑ Simulate inputs ☑ Simulate outputs

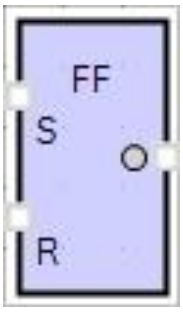
Figure 30: Example of a Circuit with a Flash Component

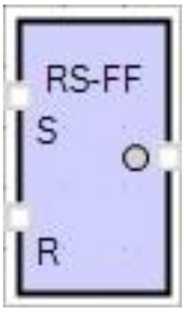
5.9. Logic Element FlipFlop

To add a FlipFlop component, click the “Add“ menu item in the context menu of the logic design, then the “FlipFlop“ menu item.

Three FlipFlop types are supported:
the classical Flipflop (without prioritization of inputs), and RS or SR FlipFlops.

The FlipFlop type can be changed in the settings for the component. Note that the Logic Designer has to be in the Design Mode, since the “Add“ function is only available during design.





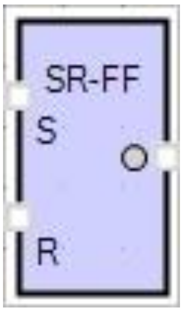
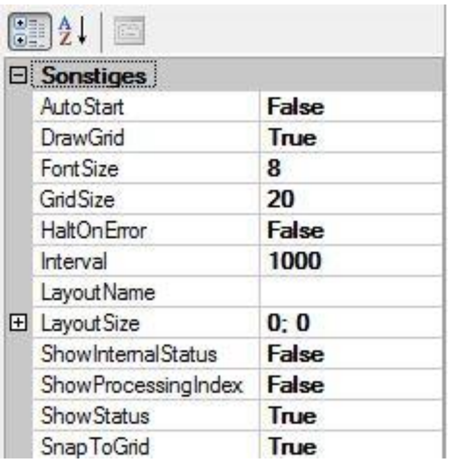


Figure 31: FlipFlop Component without Prioritization, RS FlipFlop with Prioritized Reset Input and SR FlipFlop with Prioritized Set Input

If the newly added “FlipFlop“ is selected in Design Mode, an overview of the configuration options for the component opens.



Sonstiges	
AutoStart	False
DrawGrid	True
FontSize	8
GridSize	20
HaltOnError	False
Interval	1000
LayoutName	
LayoutSize	0; 0
ShowInternalStatus	False
ShowProcessingIndex	False
ShowStatus	True
SnapToGrid	True

Figure 32: FlipFlop Component Settings

The “FlipFlopType“ property sets the type of FlipFlop when both inputs,“Set“ and “Reset“, are “High“. If the option “FF“ is set, then the input that sees a “High“ signal first, “wins“. If ”RSFF“ is set, then the output is reset when both inputs (or only the “Reset“ input) are “High“. So if both inputs are “High“, then the “Reset“ input “wins”. For the “SRFF“ setting, when both inputs are “High“, the “Set“ input “beats” the “Reset“ input.

The “InitStatus“ specifies the state of the output of the component when the logic circuit is initialized. The possible options are “High“, “Low“ and “Undefined“. If the “InvertOutput“ property is set to “True“, then the output of the “FlipFlops“ inverted. If the “InvertReset“ property is set to “True“, then the “Reset“ input of the “FlipFlop“ is inverted. If the “InvertSet“ property is set to “True“, then the “Set“ input of the “FlipFlop“ is inverted.

In the example below, the two inputs “Input1“ and “Input2“ are ANDed together and connected to the “Set“ input of the “FlipFlop“. “Input3“ is connected directly to the “Reset“ input of the “FlipFlop“. The output of the “FlipFlop“ connected to the “Output“ of the circuit. Since this is an “RS FlipFlop“ all three inputs are “High“, the output of the

“FlipFlops“ is in this case “Low“. The “Reset“ input “beats“ the “Set“ input in a “RS FlipFlop“.

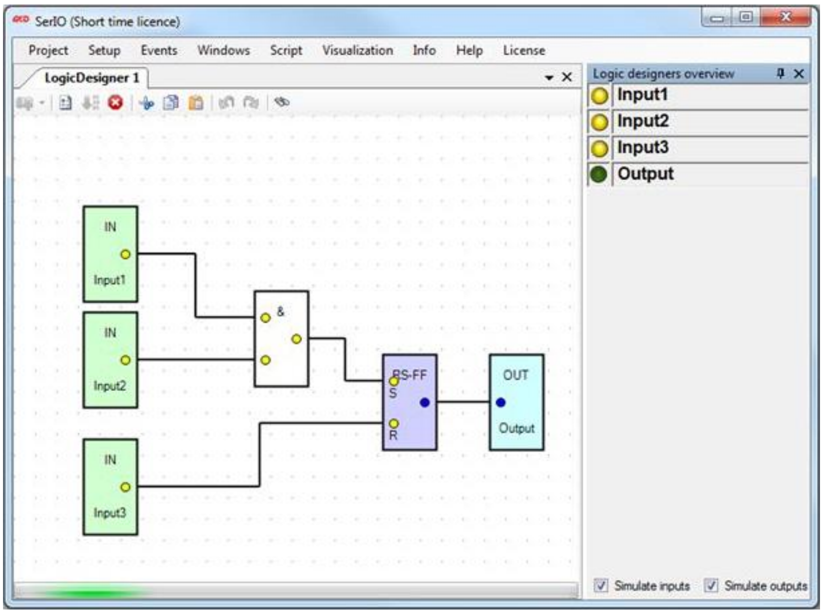
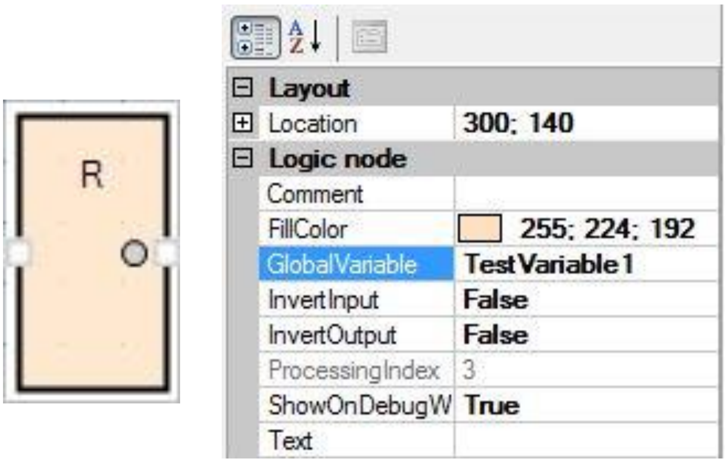

SerIO (Short time licence)
Project Setup Events Windows Script Visualization Info Help License
LogicDesigner 1
Logic designers overview
Input1
Input2
Input3
Output
IN
Input1
IN
Input2
IN
Input3
&
RS-FF
S
R
OUT
Output
☑ Simulate inputs ☑ Simulate outputs

Figure 33: Example of a Circuit with a FlipFlop Component

5.10. Logic Element Global Variable

The component “Global Variable“, or Register Component, makes it possible to connect a logic circuit to a global variable. The options for this component are described in the examples below. To add a Register Component, click the “Add“ menu item in the context menu of the logic design, then the “Global variable“ menu item. Note that the Logic Designer has to be in the Design Mode, since the “Add“ function is only available during design.

If the newly added Register Component is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for a Register Component.

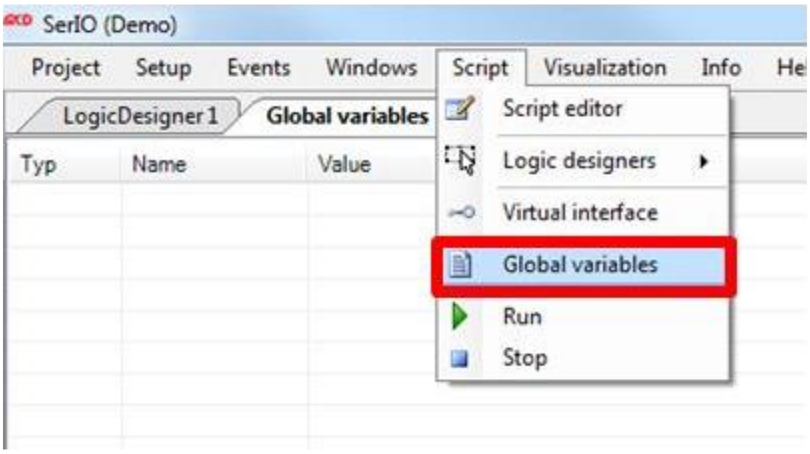


⊟ Layout	
⊞ Location	300; 140
⊟ Logic node	
Comment	
FillColor	▨ 255; 224; 192
GlobalVariable	TestVariable1
InvertInput	False
InvertOutput	False
ProcessingIndex	3
ShowOnDebugW	True
Text	

Figure 34: Global Variable Component and Settings

The “GlobalVariable“ property specifies the name of the global variable to which the component is attached. For the example below, the name “TestVariable1“ was assigned. The “InvertInput“ property can be used to invert the input of the component. The “InvertOutput“ property can be used to invert the output of the component. The “ShowOnDebugWindow“ property determines whether or not the variables are displayed in the debug window.

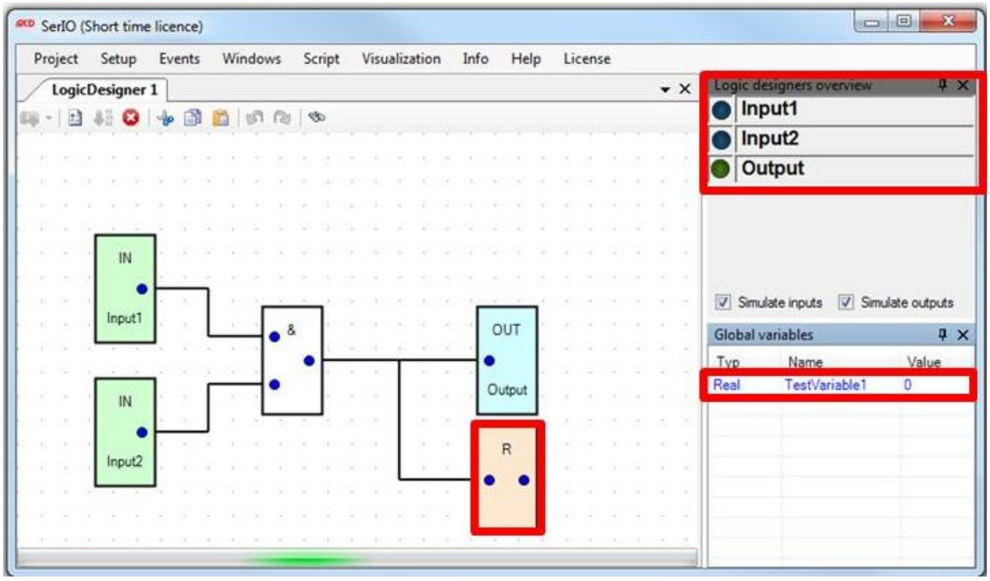
To explain the function of the Register Component in the example below, the overview of the global variables was opened, as can be seen in the figure below. Here, the current values of the global variables created can be monitored.



Typ	Name	Value

Figure 35: Opening the Global Variables Window

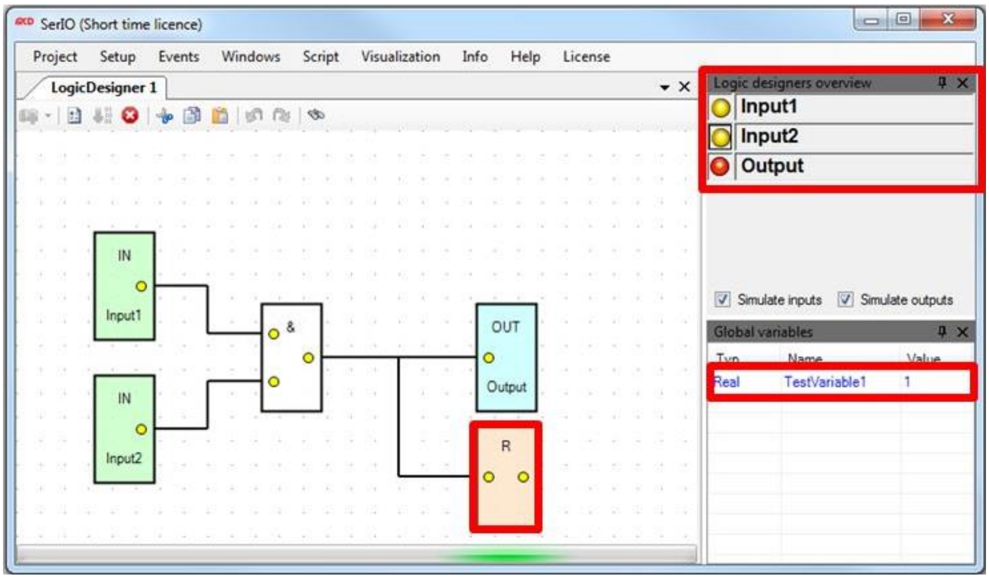
The two inputs “Input1“ and “Input2“ are ANDed together and connected to an output. The value on the output is also connected to the input of the Register Component. This is one way to use this type of component. The value of the variable can then be used in the TestManager or in a script. This allows so - called “shift registers“ to be built, these registers can remember the result or state of a previous run. The two figures below show how the value of the “TestVariable1“ (see above) behaves when the input of the component is set “Low“ (Figure 36) or “High“ (Figure 37).



☑ Simulate inputs ☑ Simulate outputs

Typ	Name	Value
Real	TestVariable1	0

Figure 36: Input of Register Component is Low, TestVariable1 is 0

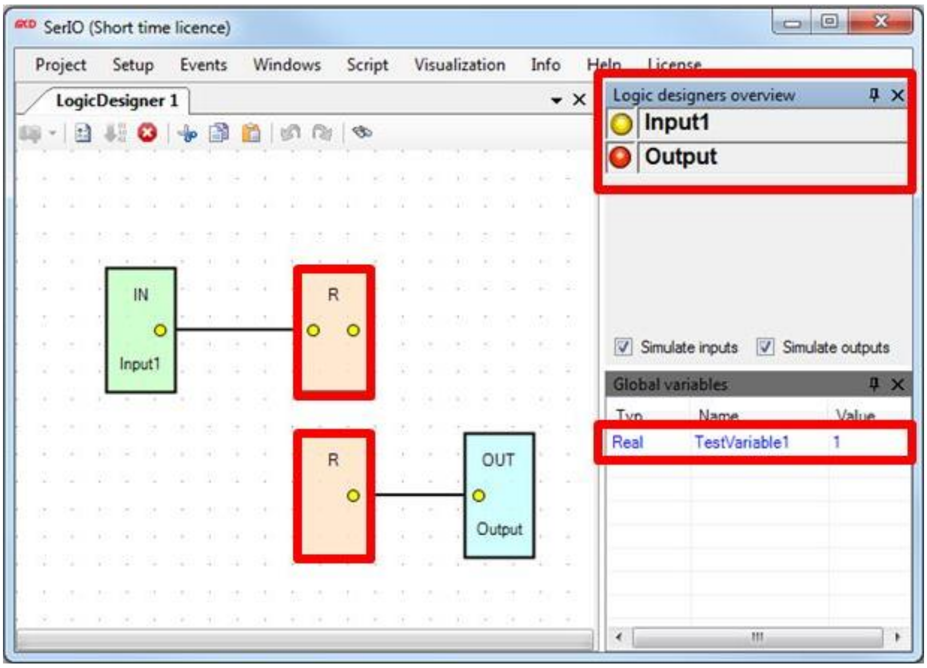


☑ Simulate inputs ☑ Simulate outputs

Typ	Name	Value
Real	TestVariable1	1

Figure 37: Input of Register Component is High, TestVariable1 is 1

In the second example, two Register Components that are connected to the same “TestVariable1“ are added. If “Input1“ is set to “High“, then “TestVariable1“ is also set to “1“ and the output is also “High“. Here, “TestVariable1“ is thus changed by a certain signal, and at the same time it is used to set the output signal.

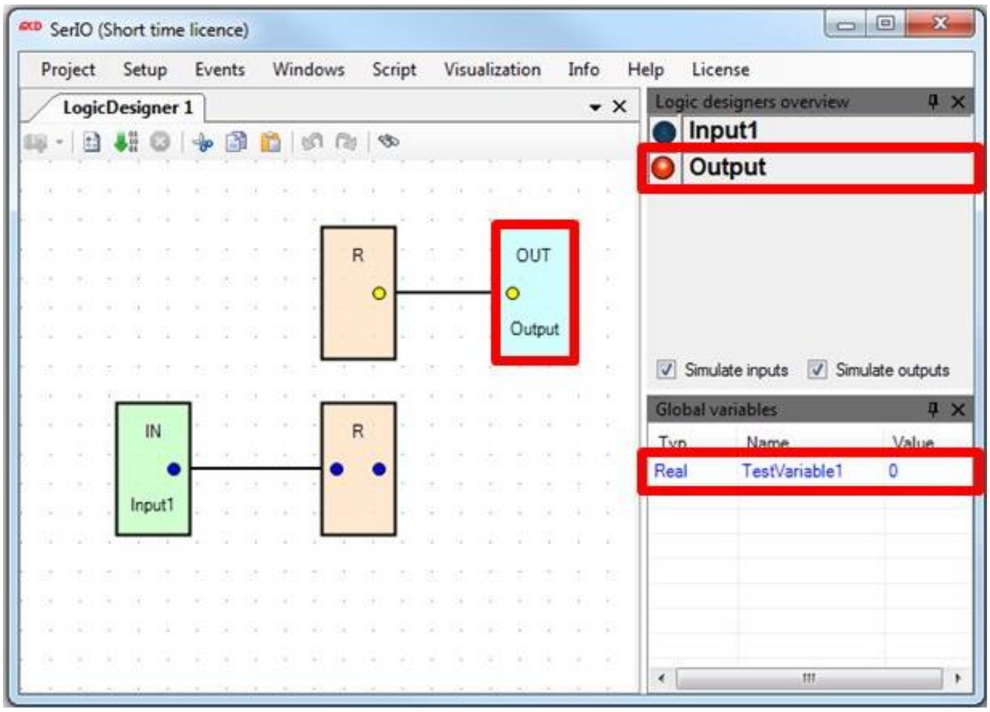


☑ Simulate inputs ☑ Simulate outputs

Typ	Name	Value
Real	TestVariable1	1

Figure 38: Example of a Circuit with Two Register Components

If the order in which the components are processed is now changed as shown in the third example, then a type of “shift register“ can be built, as already mentioned earlier. To clarify this effect (otherwise the switching would happen too quickly), the “Run“ command is not selected from the context menu between the two figures below, but rather the “Single“ command. If a “Single“ step is thus executed using the “Single“ command, it is easy to see the effect described above. The value of “TestVariable1“ is changed from “1“ to “0“, but the output is still “High“. The value from a previous cycle can therefore be “remembered“ using this behavior.



Typ	Name	Value
Real	TestVariable1	0

Figure 39: Shift Register Example

5.11. Logic Elements OR / XOR

An OR Gate performs a logical OR operation on multiple inputs to get an output. An XOR Gate (exclusive OR) performs a logical exclusive OR operation on two inputs to get an output.

To add an OR Gate, in the context menu of the logic design click on the submenu “Add“ and then in the “OR(2)“ menu item to add an OR Gate with two inputs, or “OR(4)“ to add an OR Gate with four inputs. Note that the Logic Designer must be in the Design Mode, since the “Add“ function is only available during design, and not in the run mode.

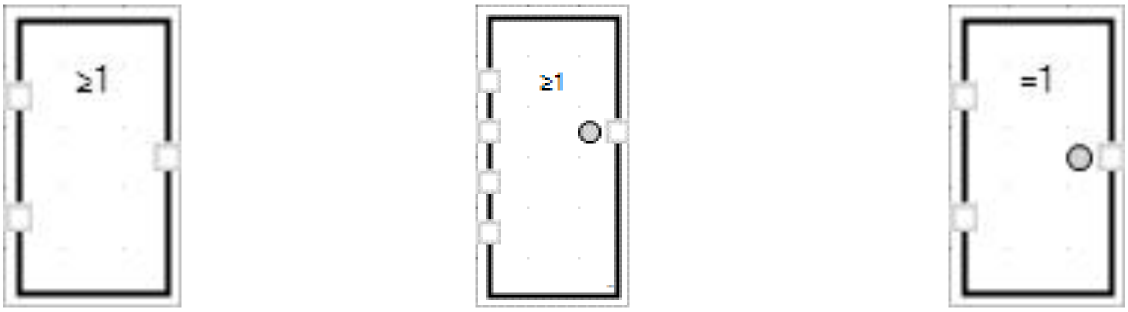
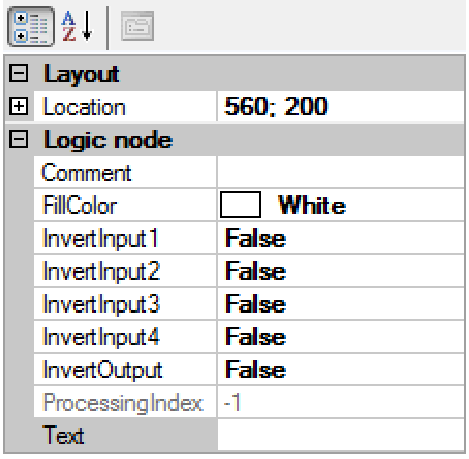


Figure 40: OR Gate with Two Inputs, OR Gate with Four Inputs and a XOR Gate

The “XOR“ menu item can be used to add an “XOR“ operation. The configuration options are identical to those of an OR Gate with two inputs.

If the newly added OR Gate is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for an OR Gate with four. For an OR Gate with two inputs, the two settings for inverting inputs three and four are omitted. Otherwise, the components have identical behavior. If an input on an OR gate with four gates is not used, then it is not taken into consideration when switching the output.

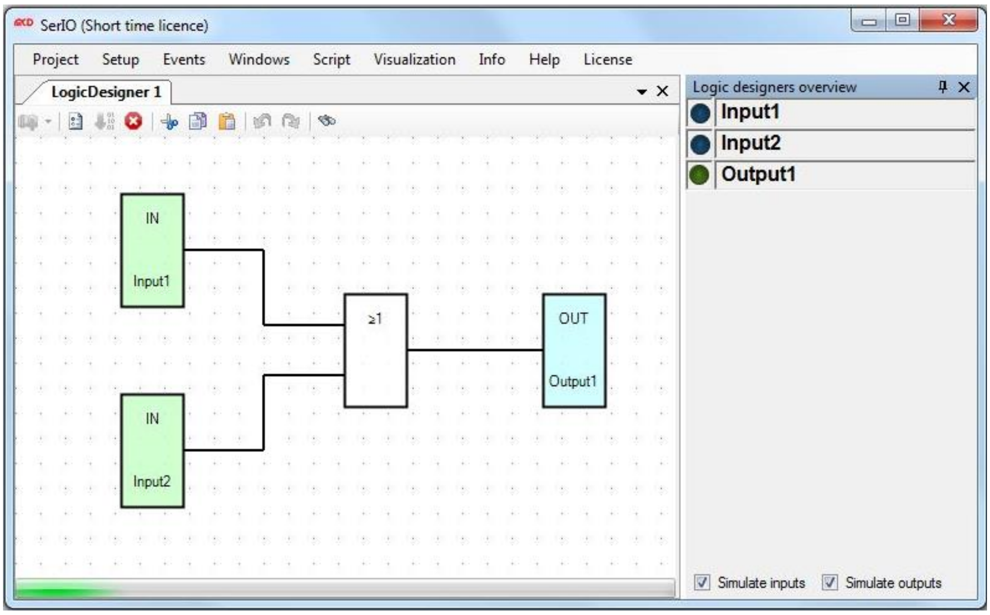


⊟ **Layout**	
⊞ Location	**560; 200**
⊟ **Logic node**	
Comment	
FillColor	□ **White**
InvertInput1	**False**
InvertInput2	**False**
InvertInput3	**False**
InvertInput4	**False**
InvertOutput	**False**
ProcessingIndex	-1
Text	

Figure 41: OR Gate Settings

The "InvertInput1/2/3/4" properties can be used to invert the different inputs of the component. The topmost input of the component is "Input1", and the lowest "Input2". The „InvertOutput" property can be used to invert the output of the gate.

The example below shows an OR circuit with two inputs. The inputs are named "Input1" and "Input2" and the output is "Output1". The inputs and outputs are then displayed in the „Logic Designers Overview" in the upper right corner under these names.



Logic designers overview

- ● **Input1**
- ● **Input2**
- ● **Output1**

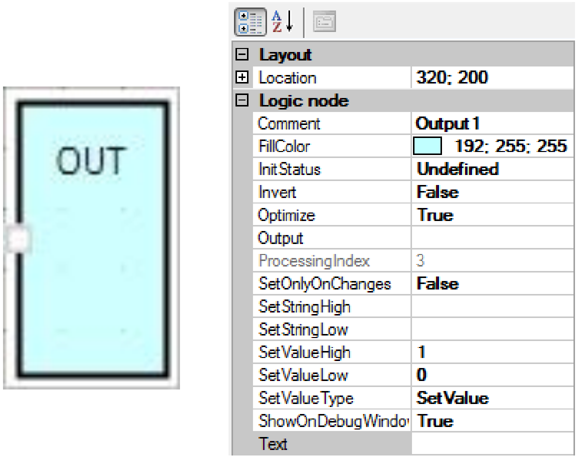
☑ Simulate inputs ☑ Simulate outputs

Figure 42: Example of an OR Gate with Two Inputs

5.12. Logic Element OUTPUT

The most important components in a logic circuit are the "Input" and "Output". These logic elements must exist in every circuit. To add an Output, click the "Add" menu item in the context menu of the logic design, then the "Output" menu item. Note that the Logic Designer has to be in the Design Mode, since the "Add" " function is only available in the Design Mode.

If the newly added Output is selected in the Design Mode, an overview of the configuration options for the component opens.



⊟ **Layout**	
⊞ Location	**320; 200**
⊟ **Logic node**	
Comment	**Output 1**
FillColor	**192; 255; 255**
InitStatus	**Undefined**
Invert	**False**
Optimize	**True**
Output	
ProcessingIndex	3
SetOnlyOnChanges	**False**
SetStringHigh	
SetStringLow	
SetValueHigh	**1**
SetValueLow	**0**
SetValueType	**SetValue**
ShowOnDebugWindo	**True**
Text	

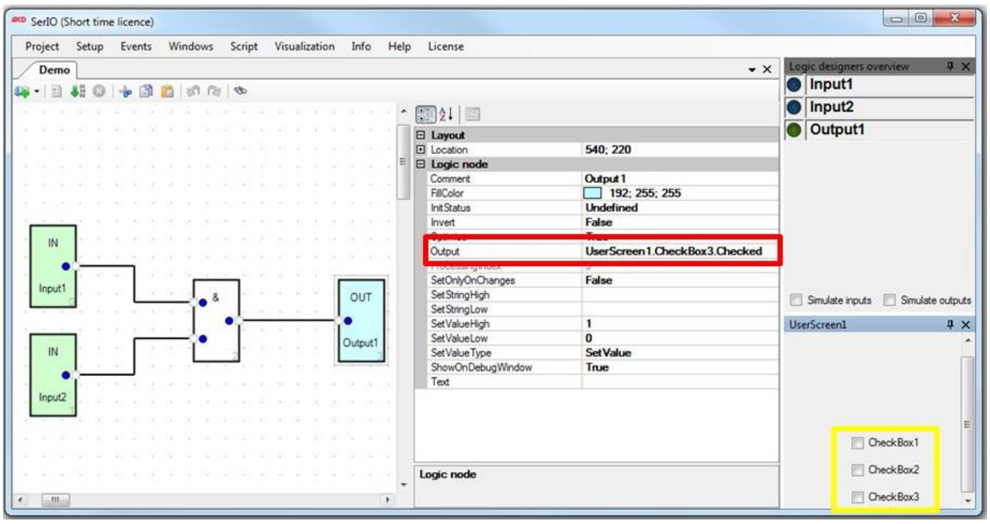
Figure 43: Output Component and Settings

The “InitStatus“ can be used to define the state of the Output when the logic circuit starts. The possible options are “High“, “Low“ and “Undefined“.

The “Invert“ property allows you to invert the input. If the “Invert“ property is set to “True“, the input is “High“ whenever its source is “Low“, and vice versa.

The „Optimize“ property can be used to tell the Logic Designer not to query the output for each new node it processes, but rather to remember the status of the output for each iteration. If “Optimize“ is set to “False“, the output is set immediately when it is determined in the process. If an Output is used more than once, it will be set multiple times during an execution. If “Optimize“ is set to “True“, the Output is only set on the end of an execution. If the state of the Output, theoretically changes multiple times during an execution, the last state “wins“. When an Output is reused several times, this can have a significant advantage in terms of performance of the logic circuit.

The “Output“ property specifies the thing to which the Output will actually be connected through the *virtual interface*. In general, such an Output executes a “SetValue“ command, so the Output can correspondingly be connected to any attribute of the “SetValue“ command. In the example below, the “Output“ is connected to a previously created “CheckBox3“ (marked in yellow) in analogy to the inputs. For the actual connection, the Output setting then specifies the “Checked“ property of “CheckBox3“ using “UserScreen1.CheckBox3.Checked“ (see Figure below).

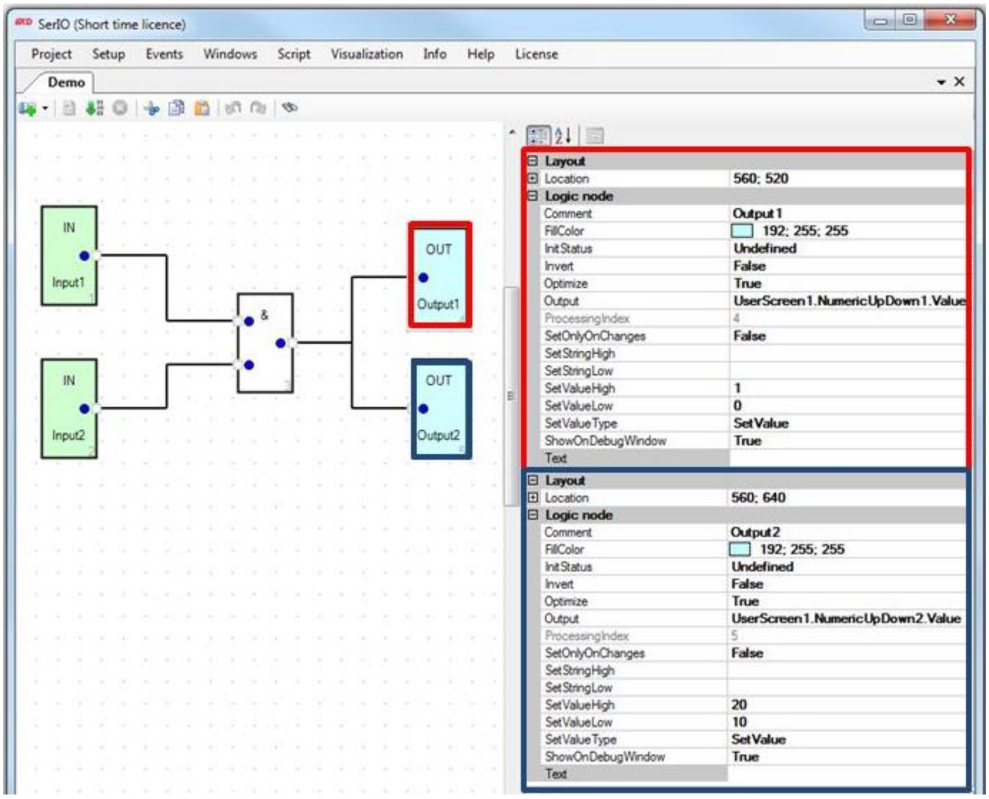


⊟ **Layout**	
⊞ Location	540; 220
⊟ **Logic node**	
Comment	Output 1
FillColor	192; 255; 255
InitStatus	Undefined
Invert	False
Output	UserScreen1.CheckBox3.Checked
SetOnlyOnChanges	False
SetStringHigh	
SetStringLow	
SetValueHigh	1
SetValueLow	0
SetValueType	SetValue
ShowOnDebugWindow	True
Text	

Figure 44: Example of a Circuit between Output to Checkbox

The “SetOnlyOnChanges“ property is also a performance optimization feature. If it is set to “True“, the Output is set only if it has changed values since the last execution. Here, it is important to understand that a state change is not detected if the Output is changed for a reason other than the command from the Logic Designer. If “SetOnlyOnChanges“ is set to “False“, the Output is set again on each execution, whether its value has changed or not.

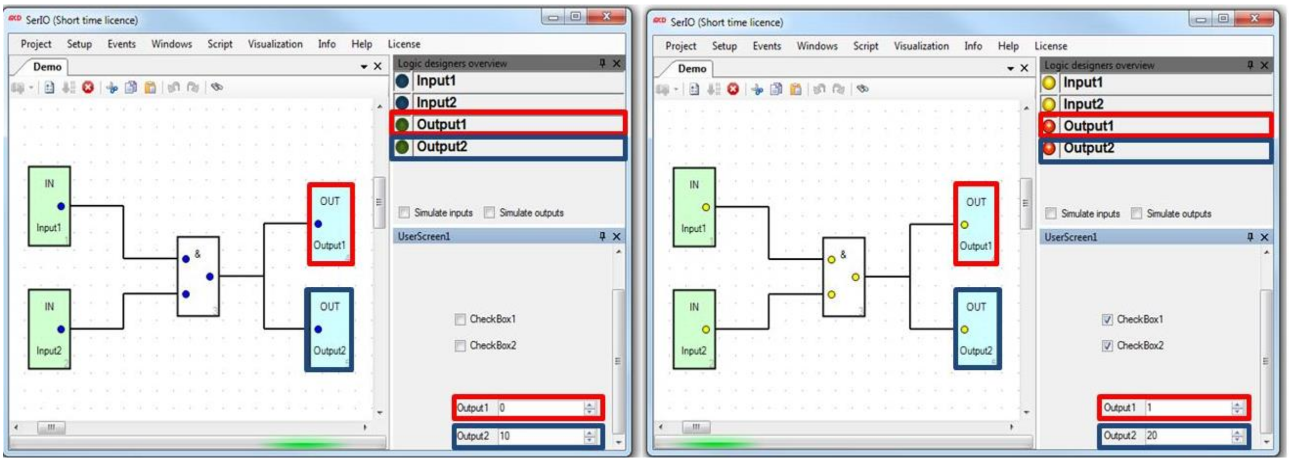
The “SetValueHigh“ and “SetValueLow“ settings can be used to specify the value the “Output“ should return if it is set “Hig or “Low“. In the example below, the ANDed signal from two inputs is connected to two outputs (“Output1“ and “Output2“) for clarification. The two outputs are connected to two „NumericUpDown“ controls created previously in the Designer, which display their return values. The top figure shows the settings for both outputs. The setting for the “Output“ property of each output is the “Value“ property of its “NumericUpDown“ control. “Output1“ returns “0“ for “Low“ and “1“ for “High“. “Output2“ returns “10“ for “Low“ and “20“ for “High“. Here, any “Integer“ values can be used. The two lower figures show the reaction of the controls when the outputs are “Low“ (left) and when the outputs are “High“ (right). Just as in the previous examples, “Input1“ is connected to “CheckBox1“ and „Input2“ to „Checkbox2“.



Layout	
Location	560; 520
Logic node	
Comment	Output 1
FillColor	192; 255; 255
InitStatus	Undefined
Invert	False
Optimize	True
Output	UserScreen1.NumericUpDown1.Value
ProcessingIndex	4
SetOnlyOnChanges	False
SetStringHigh	
SetStringLow	
SetValueHigh	1
SetValueLow	0
SetValueType	SetValue
ShowOnDebugWindow	True
Text	

Layout	
Location	560; 640
Logic node	
Comment	Output 2
FillColor	192; 255; 255
InitStatus	Undefined
Invert	False
Optimize	True
Output	UserScreen1.NumericUpDown2.Value
ProcessingIndex	5
SetOnlyOnChanges	False
SetStringHigh	
SetStringLow	
SetValueHigh	20
SetValueLow	10
SetValueType	SetValue
ShowOnDebugWindow	True
Text	

Figure 45: Settings for Output1 (red) and Output2 (blue)



Left window:

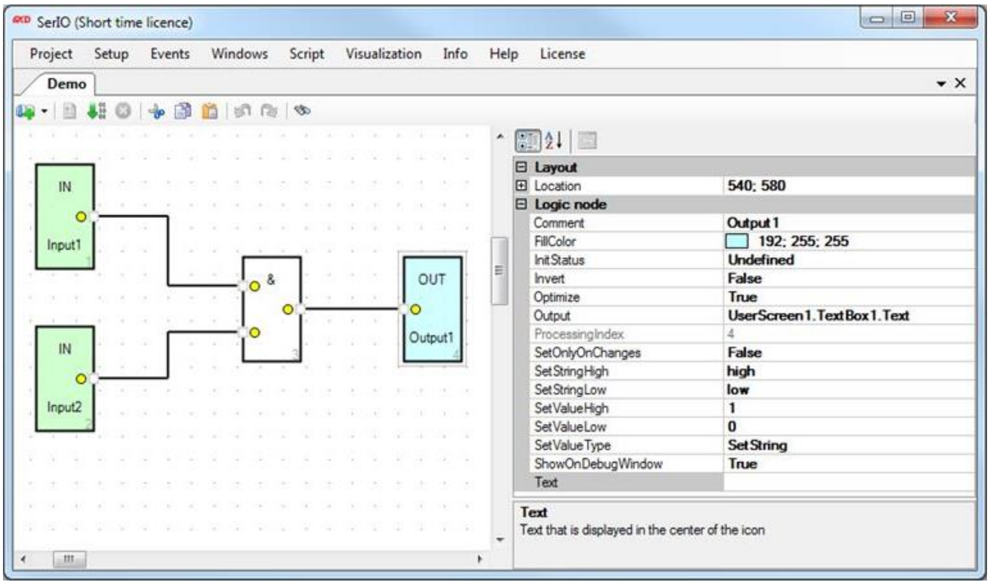
□ CheckBox1
□ CheckBox2
Output1 0
Output2 10

Right window:

☑ CheckBox1
☑ CheckBox2
Output1 1
Output2 20

Figure 46: Output of Return Values of Output1 and Output2 on Numeric UpDown Controls

If the output should not return a number, but a “String“, then this can be determined in the “SetStringHigh“ and “SetStringLow“ properties. The „SetValueType“ property can then be used to select the data type to return. In the example below, there is an output (“Output1“) which writes its return value (“String“) to a “TextBox“. If the state of the “Output“ is “High“, the “TextBox“ shows the word “High“ (see figure below: low state).

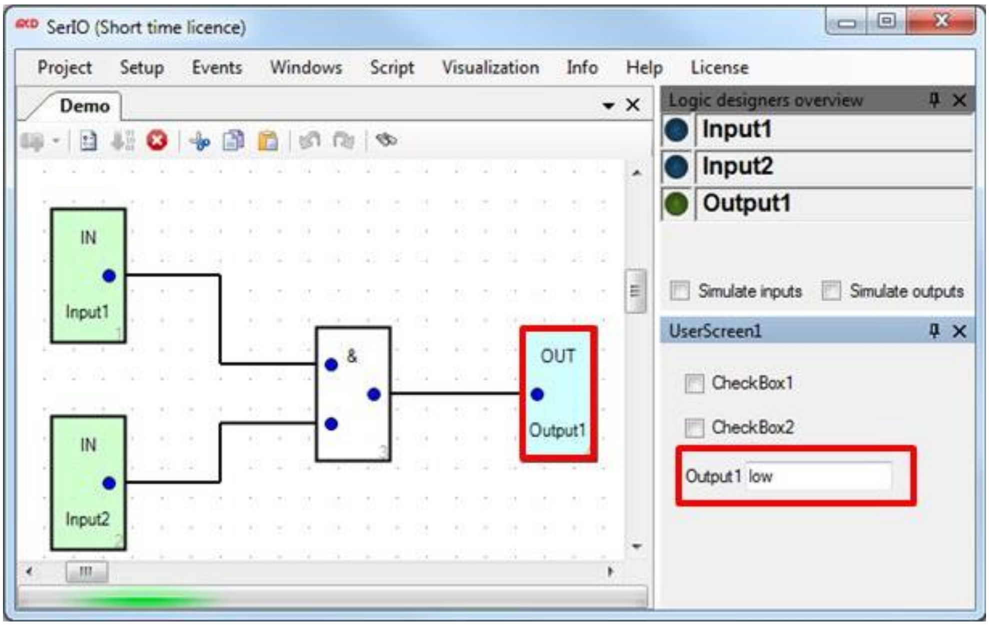

SerIO (Short time licence)
Project Setup Events Windows Script Visualization Info Help License
Demo
IN
Input1
IN
Input2
&
OUT
Output1

Layout	
Location	540; 580
Logic node	
Comment	Output 1
FillColor	192; 255; 255
InitStatus	Undefined
Invert	False
Optimize	True
Output	UserScreen1.TextBox1.Text
ProcessingIndex	4
SetOnlyOnChanges	False
SetStringHigh	high
SetStringLow	low
SetValueHigh	1
SetValueLow	0
SetValueType	SetString
ShowOnDebugWindow	True
Text	

Text
Text that is displayed in the center of the icon

Figure 47: Setting of Output1 for String Return Values and Output to TextBox1 (Low State)

The top figure shows the settings for the “Output“. “Set String“ is specified as the “SetValueType“, and the Output is specified as the “Text“ property of “TextBox1“. If the “Output“ is “Low“, the “TextBox“ displays “Low“ (see Figure 56: Output of string return value of Output1 into TestBox1 (see figure below: here low state).



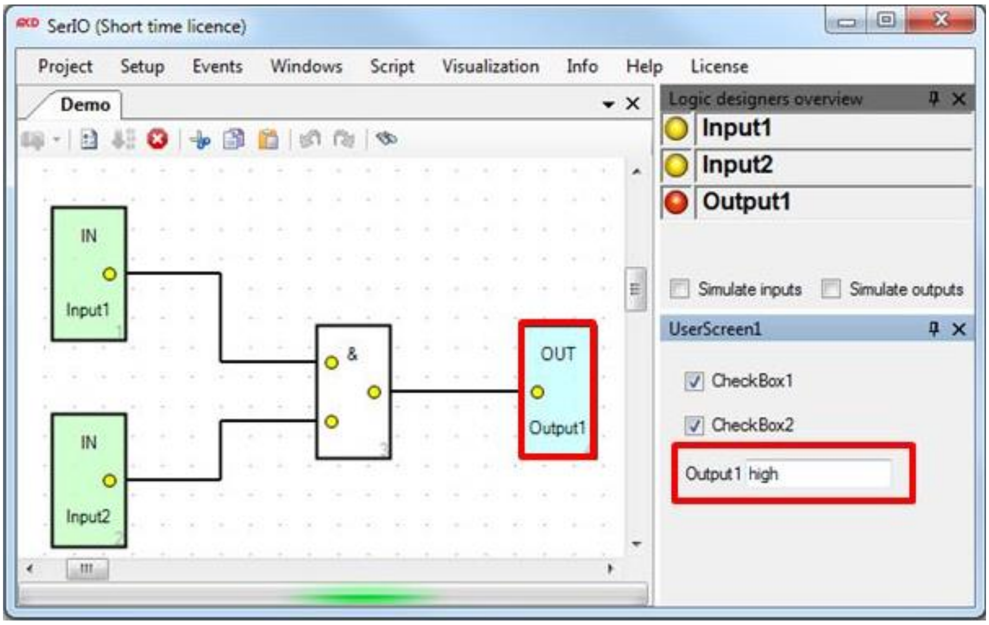
☐ Simulate inputs ☐ Simulate outputs

☐ CheckBox1

☐ CheckBox2

Output1 low

Figure 48: Output of String Return Value of Output1 into TextBox1 (Low State)



☐ Simulate inputs ☐ Simulate outputs

☑ CheckBox1

☑ CheckBox2

Output1 high

Figure 49: Output of String Return Value of Output1 into TestBox1 (here High State)

The “ShowOnDebugWindow“ property can be used to determine whether the component should be displayed in the “Logic Designers Overview“ or not.

5.13. Logic Element HIGH / LOW

The “High“ and “Low“ components provide permanent “High“ / “Low“ signals and can be used, for example, in reset or initialization circuits to set an input to a defined level.

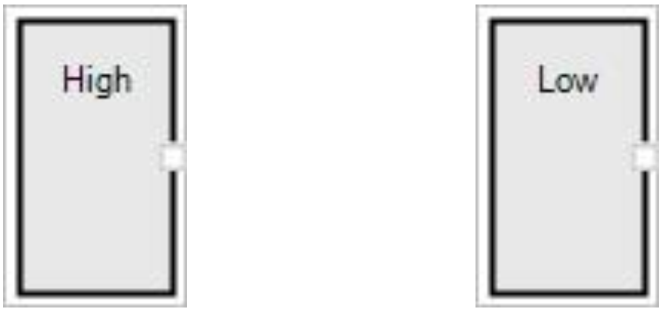


Figure 50: Logic Elements High and Low

5.14. Logic Element Script

To create functionality between different signals that would be too complicated or difficult using conventional components, there is a Script Node available. This node is available in one -, two -, or four - input variants. The Script Node is a connection from the Logic Designer to an arbitrarily configurable Script method. The following examples use a Script method named “MyNode“.

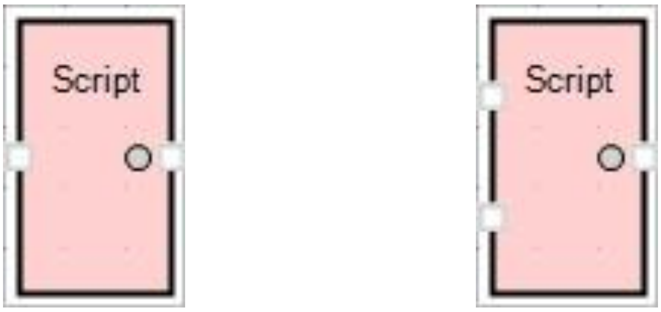


Figure 51: Script Components with One and Two Inputs

To add such a Script Node, click the „Add“ menu item in the context menu of the logic design, then the “Script“ menu item. Note that the Logic Designer has to be in the Design Mode, since the „Add“ function is only available in the Design Mode.

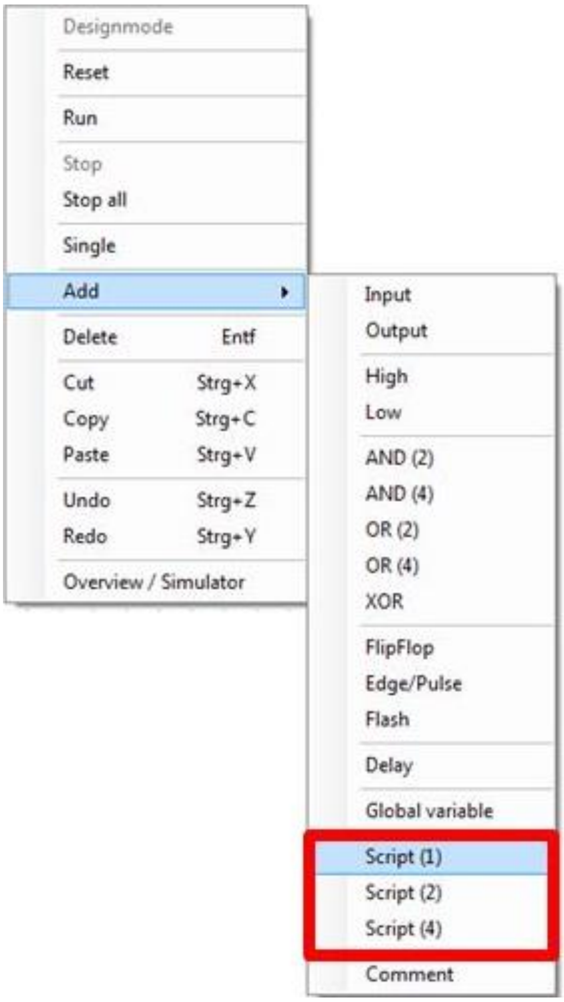
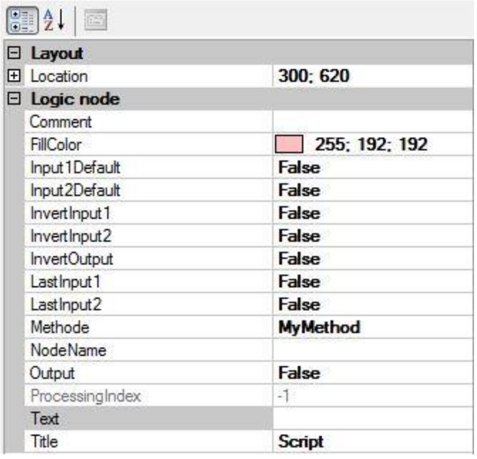


Figure 52: Adding Script Nodes

If the newly added Script Node is selected in the Design Mode, an overview of the configuration options for the component opens. The figure below shows the configuration options for a Script Node with two inputs.



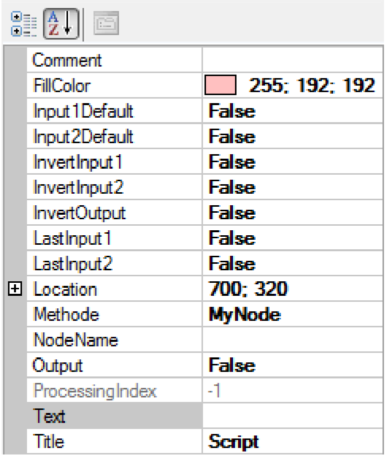
⊟ **Layout**	
⊞ Location	300; 620
⊟ **Logic node**	
Comment	
FillColor	🟥 255; 192; 192
Input1Default	False
Input2Default	False
InvertInput1	False
InvertInput2	False
InvertOutput	False
LastInput1	False
LastInput2	False
Methode	MyMethod
NodeName	
Output	False
ProcessingIndex	-1
Text	
Title	Script

Figure 53: Script Component Settings

The “Input1Default“ property can be used to set a “Default“ value for “Input1“ if it is not connected. The “Input2Default“ property can be used to set a “Default“ value for “Input2“ if it is not connected. The “InvertInput1“ property can be used to invert “Input1“ of the component. The “InvertInput2“ property can be used to invert “Input2“ of the component.“InvertOutput“ can be used to invert the output of the component. “LastInput1“ can be used to set the state of “Input1“ on the last cycle. The “LastInput2“ can be used to set the state of “Input2“ on the last cycle.

„Method“ specifies the Script method to be connected to the Script Node. “NodeName“ can be used to give the Script Node a name. “Output“ can be used to specify the current output state. The “Titel“ property can be used to give the Script Node a title that will then appear in the upper part of the component.

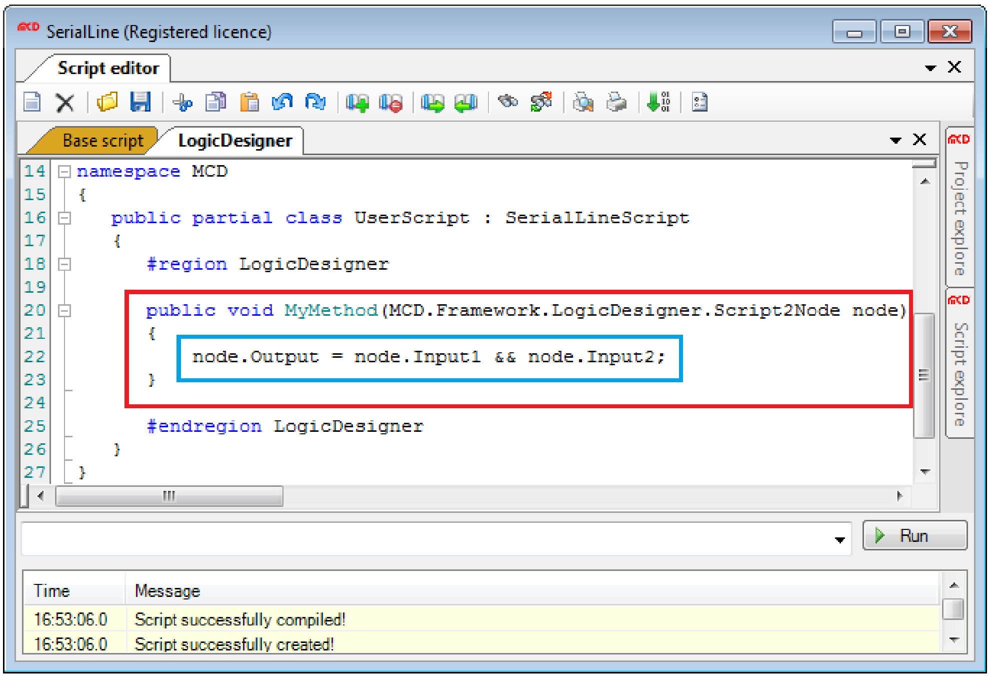
As already mentioned above, a short example will now follow to illustrate the function of the Script Node better. Just as earlier, the Script Node was connected using its “method“ setting to the example method “MyNode“.



Comment	
FillColor	▨ 255; 192; 192
Input1Default	False
Input2Default	False
InvertInput1	False
InvertInput2	False
InvertOutput	False
LastInput1	False
LastInput2	False
⊞ Location	700; 320
Methode	MyNode
NodeName	
Output	False
ProcessingIndex	-1
Text	
Title	Script

Figure 54: Connecting the Script Node to Method MyNode

Since the „MyNode“ method does not exist yet, the “Compiler“ would display an error message if “Run“ were clicked, since it cannot find the function. Once the method name has been assigned, however, a double - click on the pink Script Node automatically creates the method in the script editor (marked in red in the figure below). As the default setting, a logical AND connection is set up between the two settings (marked in blue in the figure below).



```
namespace MCD
{
    public partial class UserScript : SerialLineScript
    {
        #region LogicDesigner

        public void MyMethod(MCD.Framework.LogicDesigner.Script2Node node)
        {
            node.Output = node.Input1 && node.Input2;
        }

        #endregion LogicDesigner
    }
}
```

```
Time        Message
16:53:06.0  Script successfully compiled!
16:53:06.0  Script successfully created!
```

Figure 55: MyNode Method in the Script Editor

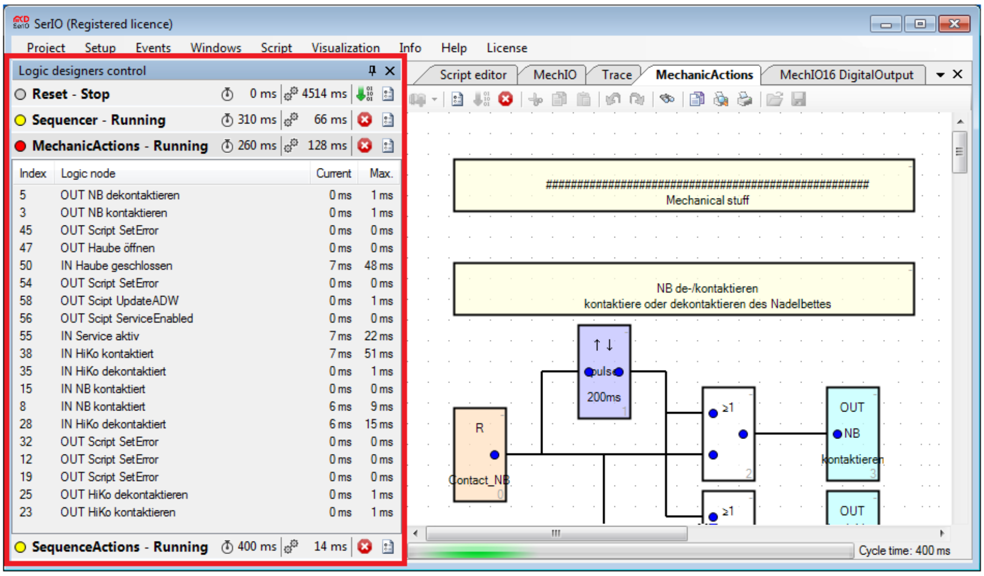
After clicking the “Run“ command, the two inputs “Input1“ and “Input2“ are now connected by a logical AND gate. Instead of the AND connection, any logical operation can be implemented here.

6. Logic Designers Control

6.1. Overview

The „Logic Designers Control“ is a tool of the MCD Toolmonitors for display, control and analysis of logic designs that are used in a project. Features of the „Logic Designers Control“ are:

- Clear / tabular display of all logic designs with current states and processing cycle durations
- Starting / Stopping of logic designs from a central point
- Detailed view of listed relevant “Logic Nodes“ with respective processing cycle duration
- Quick opening of the Logic Designer via a double - click on the name



○ Reset - Stop 0 ms 4514 ms

○ Sequencer - Running 310 ms 66 ms

● MechanicActions - Running 260 ms 128 ms

Index	Logic node	Current	Max.
5	OUT NB dekontaktieren	0 ms	1 ms
3	OUT NB kontaktieren	0 ms	1 ms
45	OUT Script SetError	0 ms	0 ms
47	OUT Haube öffnen	0 ms	0 ms
50	IN Haube geschlossen	7 ms	48 ms
54	OUT Script SetError	0 ms	0 ms
58	OUT Scipt UpdateADW	0 ms	1 ms
56	OUT Scipt ServiceEnabled	0 ms	0 ms
55	IN Service aktiv	7 ms	22 ms
38	IN HiKo kontaktiert	7 ms	51 ms
35	IN HiKo dekontaktiert	0 ms	1 ms
15	IN NB kontaktiert	0 ms	0 ms
8	IN NB kontaktiert	6 ms	9 ms
28	IN HiKo dekontaktiert	6 ms	15 ms
32	OUT Script SetError	0 ms	0 ms
12	OUT Script SetError	0 ms	0 ms
19	OUT Script SetError	0 ms	0 ms
25	OUT HiKo dekontaktieren	0 ms	1 ms
23	OUT HiKo kontaktieren	0 ms	1 ms

○ SequenceActions - Running 400 ms 14 ms

Figure 56: Logic Designers Control (Marked in Red) Docked on the Left

6.2. Opening the Logic Designers Control

Via the menu item **Script → Logic designers → Logic designers control** the „Logic Designers Control“ can be opened.

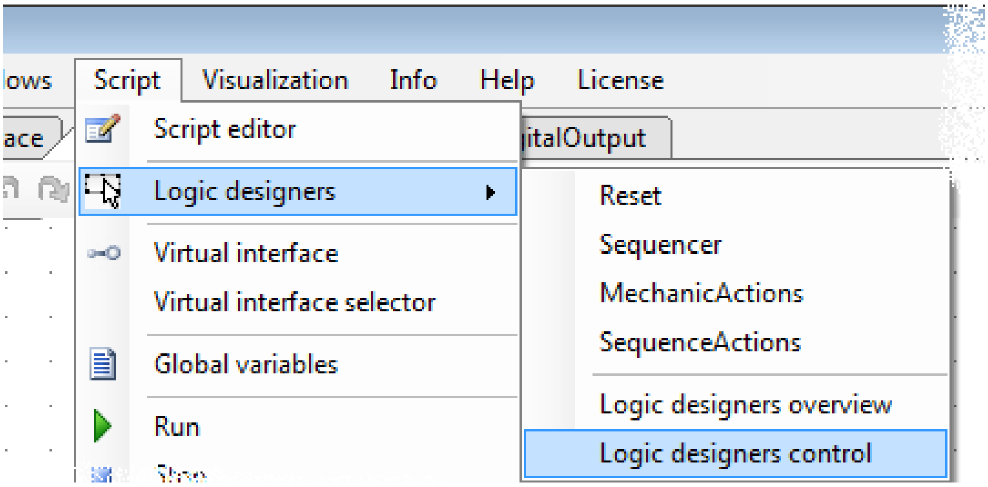
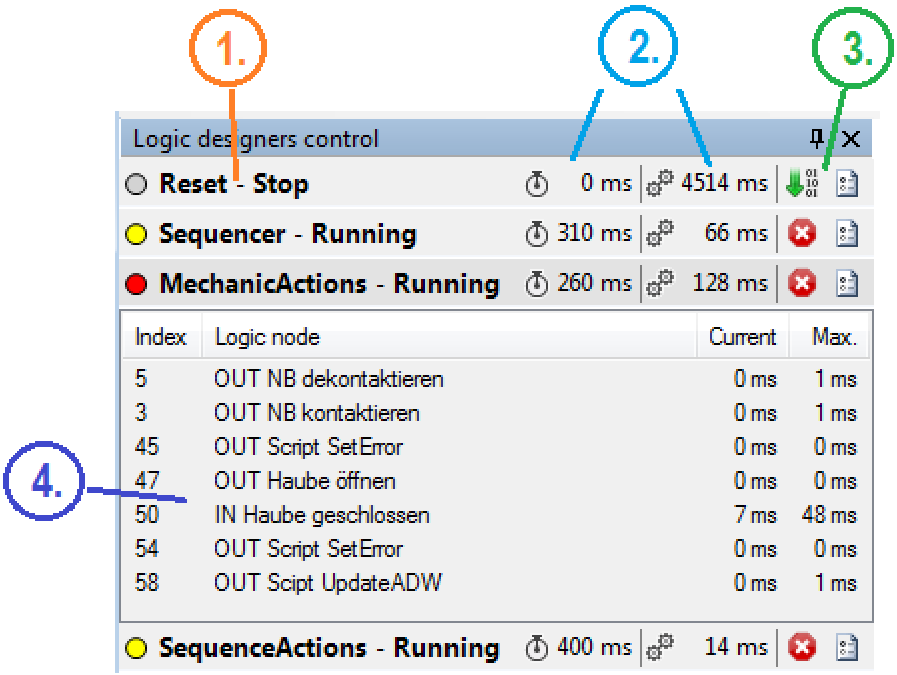


Figure 57: Opening the Logic Designers Control

6.3. Graphical Elements

„Logic Designers Control“ contains all the logic designs that are defined in the application. These are presented line by line amongst each other. An entry of a logic design contains:

- Name and current state of the logic design (1.)
- Last cycle interval and processing cycle duration (2.)
- Buttons for control of the logic designs (3.)



Logic designers control

○ Reset - Stop 0 ms 4514 ms

● Sequencer - Running 310 ms 66 ms

● MechanicActions - Running 260 ms 128 ms

Index	Logic node	Current	Max.
5	OUT NB dekontaktieren	0 ms	1 ms
3	OUT NB kontaktieren	0 ms	1 ms
45	OUT Script SetError	0 ms	0 ms
47	OUT Haube öffnen	0 ms	0 ms
50	IN Haube geschlossen	7 ms	48 ms
54	OUT Script SetError	0 ms	0 ms
58	OUT Scipt UpdateADW	0 ms	1 ms

● SequenceActions - Running 400 ms 14 ms

Figure 58: Display of a Logic Designer Entry

Via left - clicking on a logic design the entry display changes to detail mode (4.) and the “Logic Nodes“ contained inside the logic design are shown.

6.4. Current State

The current state is visualized with a colored symbol on the left side.

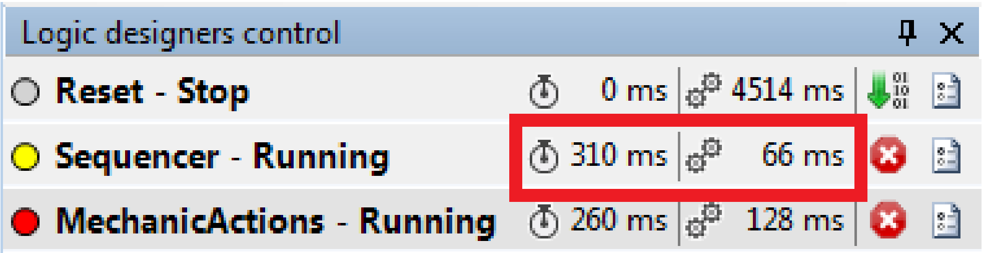
Symbol / Color	State	Description
○ Grey	Stop	Logic design is stopped and is not further processed
🟡 Yellow	Running, Idle	Logic design is executed and awaiting to be processed
🔴 Red	Running, Processing	Logic design is executed and is currently processed

Figure 59: State Visualization

6.5. The Last Cycle Interval and the Processing Cycle Duration

The "Last Cycle Interval" and "Processing Cycle Duration" of a logic design are important parameters for its performance analysis.

The "Last Cycle Interval" means the absolute time in milliseconds that passes between the last and the current execution. That at best corresponds to the defined time from the "Interval" property and is at least as high as the effective processing cycle duration of all active logic designs. The "Last Cycle Interval" is marked with a stopwatch icon (⏱).



Logic designers control

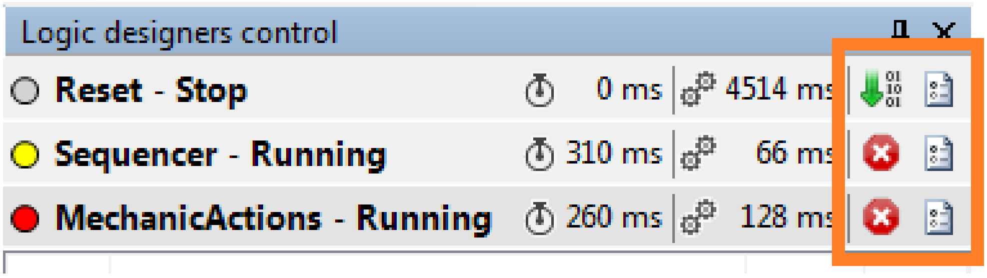
○ Reset - Stop	⏱ 0 ms	⚙ 4514 ms	⬇ 📄
● Sequencer - Running	⏱ 310 ms	⚙ 66 ms	⊗ 📄
● MechanicActions - Running	⏱ 260 ms	⚙ 128 ms	⊗ 📄

Figure 60: Display of the Last Cycle Interval

The "Processing Cycle Duration", marked with a gear icon (⚙), depicts the actual duration of the logic design processing. It consists of the processing time of all "Logic Nodes" belonging to the logical network. The processing time of every single "Logic Node" can be displayed / analyzed in detail view.

6.6. Direct Controlling of Logic Designs: Start / Stop and Open

For direct controlling of logic designs with the "Logic Designers Control" the "Run" (⬇), "Stop" (⊗) and "Design Mode" (📄) buttons are available.



Logic designers control

○ Reset - Stop	⏱ 0 ms	⚙ 4514 ms	⬇ 📄
● Sequencer - Running	⏱ 310 ms	⚙ 66 ms	⊗ 📄
● MechanicActions - Running	⏱ 260 ms	⚙ 128 ms	⊗ 📄

Figure 61: Control Buttons of the Logic Designs

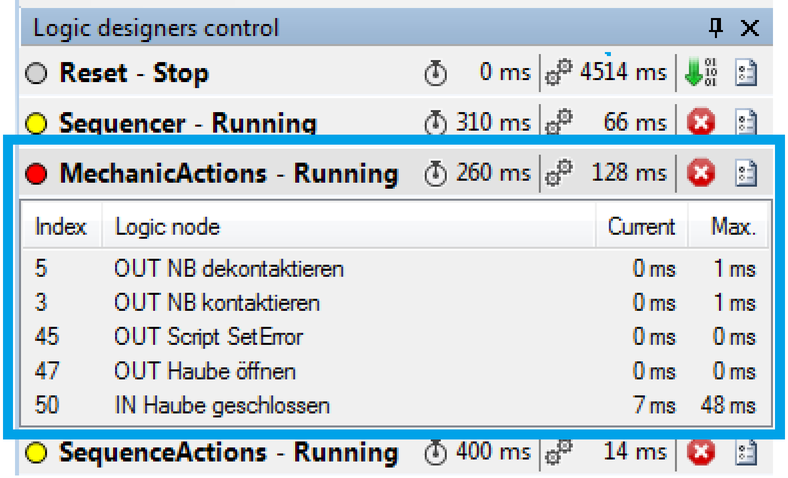
The possibility of quick navigation between the logic designs is a further helpful feature of the “Logic Designers Control“: The logic design is opened and focused on via double left - clicking. The actual run mode is maintained.

Button / Function	Function
Run (⬇)	Starts the logic circuit
Stop (✖)	Stops the logic circuit (only in run mode)
Design Mode (📋)	Opens the logic design to edit (only in run mode)
Double clicking on name	Opens the logic design and sets the focus on it

Figure 62: Summary of the Control Functions

6.7. Detail View

By clicking on an entry of the “Logic Designers Control“, a list of its logic elements is displayed. This list contains all logic elements relevant to the performance analysis: “Inputs“, “Outputs“, “Register“ and “Scripts“. For these logic elements, the following data is expended: “Processing Index“, the name of the logic element, the current pocessing cycle duration and the maximal processing cycle duration (as measured by the last start of the logic design).



Logic designers control

○ **Reset - Stop** 0 ms 4514 ms

○ **Sequencer - Running** 310 ms 66 ms

● **MechanicActions - Running** 260 ms 128 ms

Index	Logic node	Current	Max.
5	OUT NB dekontaktieren	0 ms	1 ms
3	OUT NB kontaktieren	0 ms	1 ms
45	OUT Script SetError	0 ms	0 ms
47	OUT Haube öffnen	0 ms	0 ms
50	IN Haube geschlossen	7 ms	48 ms

○ **SequenceActions - Running** 400 ms 14 ms

Figure 63: List of Logic Elements

Column name	Description
Index	Processing Index: Processing sequence of the logic element within the logic network
Logic node	Name / description of the logic element
Current	Last measured processing cycle duration of the logic element in milliseconds
Max.	Maximal processing cycle duration of this logic element since the last start of the logic design

Figure 64: Summary of the Data Output