Advanced Logic Module - Manual

Version 1.0

REV04-20240506

GENERAL INFORMATION

DIVUS GmbH
Pillhof 51
I-39057 Eppan (BZ)

The manual is subject to change without notice. We cannot guarantee that the data contained in this document and on the storage media supplied are free of errors and correct. Suggestions for improvements as well as hints on errors are always welcome. The agreements also apply to the specific annexes to this manual.

The designations in this document may be trademarks whose use by third parties for their own purposes may infringe the rights of their owners.

User instructions: Please read this manual before using it for the first time and keep it in a safe place for future reference.

Target group: The manual is written for users with previous knowledge of PC and automation technology.

PRESENTATION CONVENTIONS

[KEY]	Keystrokes of the user are shown in square brackets, e.g. [CTRL] or [DEL].
COURIER	Screen output is described in the Courier font, e.g. C:\>
COURIER FAT	Keyboard input by the user is described in Courier font bold, e.g. C:\> DIR
"„…“	Names of buttons, menus or other screen elements to be selected are displayed in "inverted commas".
PICTOGRAMS	The following pictograms are used in the manual to identify certain sections of text:
	Watch your step! Possibly dangerous situation. Damage to property can be the result.
	Notes Tips and supplementary information
	New Marks changes and new features

TABLE OF CONTENTS

1 General Overview

1.1 INTRODUCTORY REMARKS

The Advanced Logics module allows complex logical networks to be implemented within the DIVUS KNX SERVER/D+ through an integrated graphical editor and an extensive library of logical blocks with which any objects managed by the server can be interconnected.

The module makes it possible to create one or more **logic units,** which can consist of one or more **tasks**; each task is a logical network configurable via a graphic editor, similar to the KNX SERVER's *Programmable Events*, which can exchange data with other tasks by suitable variable type objects, as specified below.

The tasks can be activated or deactivated, and based on this activation, when starting a unit (simulated or "real"), a script in the LUA language is generated that runs in the background and continues to exchange input and output data in a loop, conceptually similar to what happens in PLCs.

Logic units can be run in parallel - they are independent processes; conversely, the *tasks are* parts of a *Logic unit* and flow into the LUA script during compilation. Therefore, all enabled tasks are always executed, from the first to the last, in the order specified on the program configuration page.

Unlike the other logical functions of KNX SERVER, which are processed "on event" (i.e. when a value change occurs at one of the inputs), the LUA Logic units run continuously.

The LUA code is executed cyclically in the background, and at each execution cycle, it processes the logic based on the state of the input objects detected at the beginning of the cycle; the output objects are commanded accordingly, if their state is not already coherent with what the logic predicts.

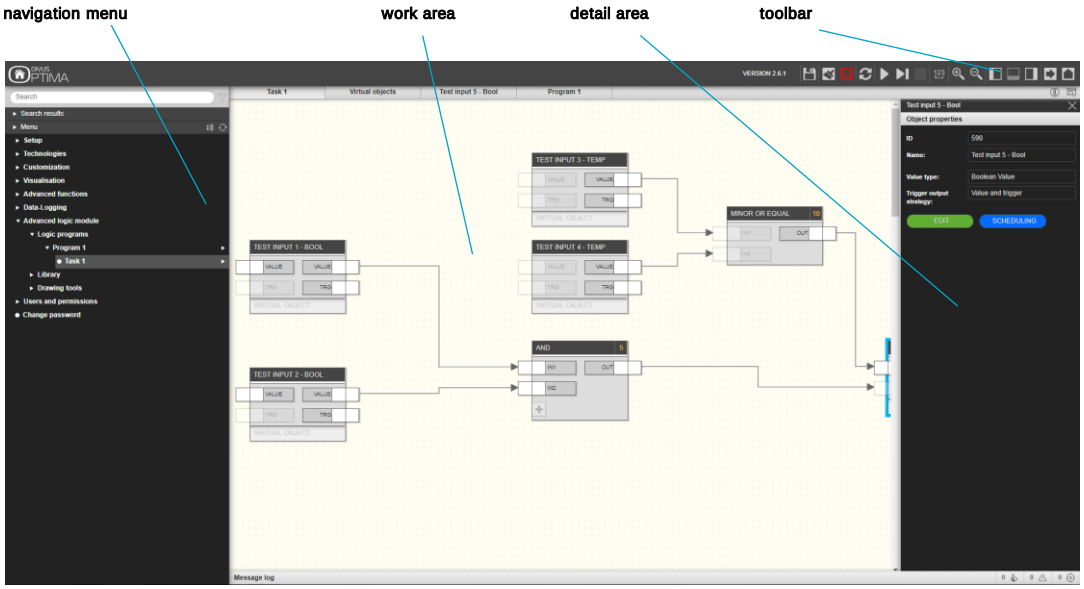
Under normal conditions, therefore, the output objects are kept aligned with the logic even if their state varies for external reasons, by immediately sending a command (to the next LUA execution cycle); this behaviour is very different from other KNX SERVER/D+ event logics, and should be kept in mind when creating a logic network within the advanced logic module.

It is however possible to regulate when the LUA logic is to modify the output objects, e.g., only if the input objects change (thus avoiding "blocking" the outputs if they change for any reasons, for example because they are commanded by the user), using so-called *TRIGGERS*. A special chapter, in the last part of this manual, explains how to use them; a thorough reading of this section is recommended.

2 User interface

2.1 GENERAL LAYOUT

The following figure shows the structure of the graphical user interface of the editor as soon as the window is opened:



2.2 NAVIGATION MENU

The menu contains everything you need to create and manage Logic units. To do this, click on *Advanced logic module*.

2.2.1 LOGIC UNITS

This section contains the list of configured Logic units. Here you can create, edit and delete Logic units.

2.2.2 LIBRARY

This section contains the library of logical blocks that can be inserted into the programs. As described later in more detail, the logical library entries can be inserted into the programs using "Drag and Drop".

2.2.3 DRAWING TOOLS

This section contains the tools we use to customize our programs. As described later in more detail, the drawing tools can be added to the programs by drag & drop.

2.3 TOOLBAR

The toolbar provides the following tools in the middle section, which are available during each phase of the realization of the Logic units:

EXECUTION / BREAK
Start or stop a program

CONTINUOUS SIMULATION
Start the simulation in real-time mode

STEP-BY-STEP SIMULATION
Start the simulation in step-by-step mode

SIMULATION Stop
Stop the current simulation

SORTING
Rearrange the blocks by automatically sorting them by position

ZOOM +
Increase the zoom factor of the work area

ZOOM -
Reduces the zoom factor of the work area.

SHOW / HIDE MESSAGES
Show or hide the notification area below

SHOW / HIDE NAVIGATION MENU
Show or hide the navigation menu on the left side of the screen

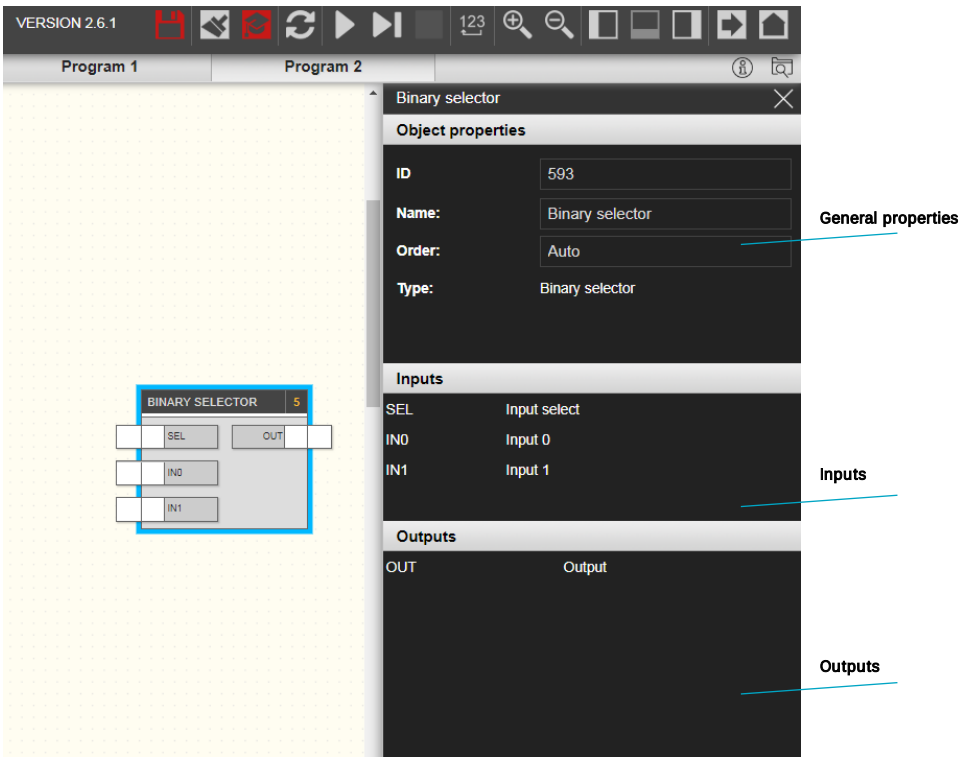
SHOW / HIDE DETAIL AREA
Show or hide the right window with the details

ADVANCED OPTIONS
Allows access to advanced options that are hidden by default

2.4 DETAIL AREA

This normally closed area can be opened using the corresponding button in the toolbar or by right-clicking on the desired object. It contains details about the objects selected in the work area and allows you to change the properties and options.

Depending on the type of object selected, the information can be divided into several sections as shown in the figure below:

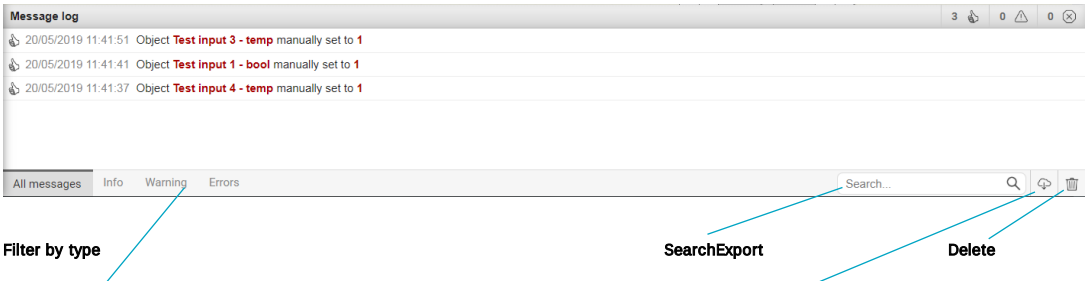


2.5 WORK AREA

The central part of the window is dedicated to the workspace in which the logics are constructed as described below. To extend the usable space, it is recommended to close the navigation area and the lower notification area, especially when editing the Logic units.

2.6 NOTIFICATION AREA

The lower part of the window contains the messages generated by the editor during the realization of the Logic units and especially during the simulation (as described in more detail below).



The messages generated by the editor can have different types depending on their severity and type:

- **Errors**: Reports on transactions or conditions that cause an error and normally require modification or verification by the user.

- **Warning:** Warnings of abnormal conditions, which do not necessarily represent an error or a situation to be changed.

- **Info**: "normal" information messages that report operations performed by the editor that are worth reporting to the user.

- **Debug**: detailed messages of the operations performed by the simulation (only available in "step by step" mode, as described below)

The different types are distinguished by a color highlighted on the page of each message, along with the date and time the message was generated. The Notification Area title bar on the right contains a summary of the number of messages of different types that are visible even when the Notification Area is closed.

The following commands are available at the bottom of the notification area:

- **Filter by message type**: By selecting one of the available entries it is possible to filter the messages on the screen according to the corresponding type

- **Search for messages**: Allows you to filter messages based on one or more keywords

- **Export**: allows the message history (including those related to previous work sessions) to be exported in CSV format, which can be retrieved via external software (e.g. spreadsheets).

- **Empty**: allows deleting messages on the screen (messages still remain archived in the editor and can be exported via the corresponding button for an "Offline" viewing)

3 Logic units

3.1 INTRODUCTION

The logic modules are set up to execute one or more logical networks (tasks) that typically receive one or more pieces of information from the bus, process them through logic blocks and send the results to the bus in the form of commands. Each unit can contain up to 16 tasks.

The editor allows you to configure Logic units by combining blocks and logical functions via drag & drop and simple graphical tools without special programming knowledge. As explained below, the editor also allows you to simulate the behaviour of Logic units.

3.2 CREATE A NEW LOGIC UNIT

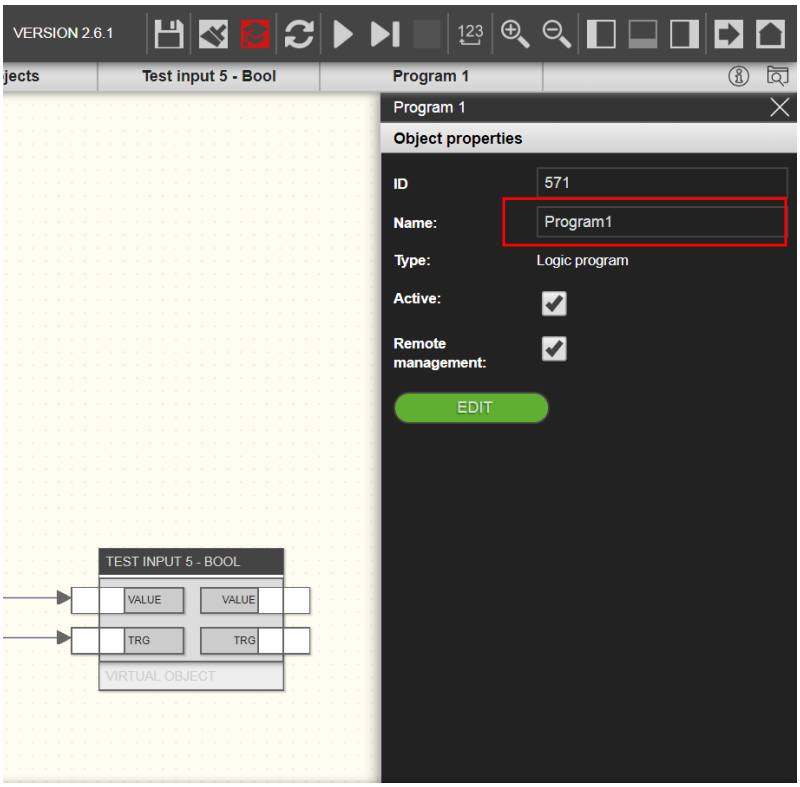
To create a new Logic unit, click on "Logic units" in the "Logic module" section of the main menu and press the corresponding "+" key: a new empty unit called "Logic unit 1" will be created.

3.3 CREATE A NEW TASK

To create a new task, first select the Logic unit you want to work on in the Logic units section of the main menu and press the corresponding + key: a new empty task called Task 1 will be created.

To open the new task, simply click on it: An empty window is displayed in the work area, where you can start building the logic as described below.

To change the name of the task, open the details pane and type the new name in the appropriate text box, as shown in the following figure. The name may not contain any special characters and may be a maximum of 16 characters long.



3.4 REMOVE OR DEACTIVATE A UNIT

To remove an existing unit, simply press the corresponding "X" key in the Logic units window; as soon as the deletion is confirmed, the unit and all its logical functions will be eliminated. This operation cannot be cancelled.

If you do not want a task to be inserted in the logical unit, for example because it is still incomplete, you can deactivate it by deselecting the corresponding "ENABLE" entry in the details area; deactivated tasks are marked with a semi-transparent effect.

3.5 ADDING BLOCKS TO A TASK

The tasks contain the connection of several blocks to a logical network. The blocks can be classic OPTIMA objects, especially KNX objects, or they can be of logical type; the former are useful to read and/or write information on the building automation bus, the latter allow to process and combine this information.

To add a classic block to a task, you must first identify it in the "ETS Project" section located in the "KNX" section below the "Technologies" entry of the main menu. All available KNX objects are listed here.

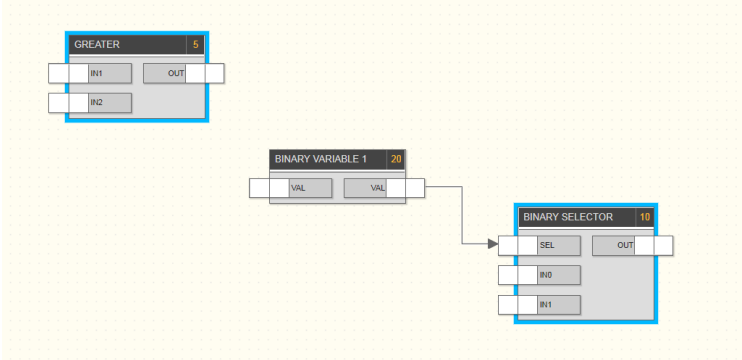
Once the object has been identified, simply drag and drop it into the workspace:

To insert a logic block, you must also identify it in the library under *Logic Module* (for a complete list of available logic blocks, see Chapter 5) and then drag it to the workspace:

3.6 SELECT ONE BLOCK OR MULTIPLE BLOCKS

It is possible to select one or more blocks within a task in different ways:

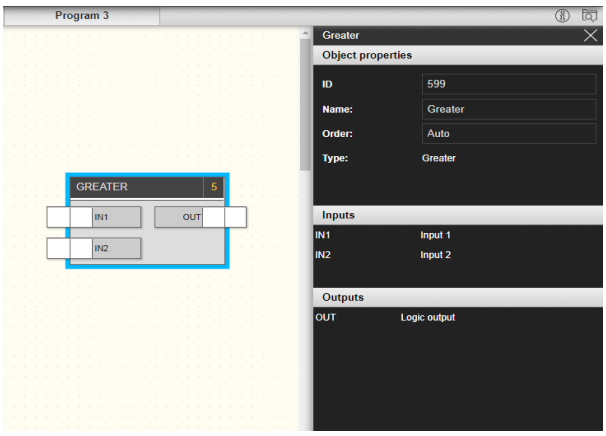
- Click on the "title" of the block (single selection)
- By clicking on the "title" of several blocks while simultaneously pressing the CTRL key (scattered multiple selection)
- By clicking and holding one point of the workspace, you move the cursor by drawing a rectangular selection area (adjacent multiple selection).



The selected blocks are highlighted with a blue border:

The selected blocks can easily be moved within the workspace by dragging and dropping them. By selecting a single block and opening the details window, you can display its properties, the list of input and output nodes, and manage all options as described later for each type.

NOTE: If you select multiple blocks at the same time, it is not possible to see the details because they are different for each one.



3.7 DELETION OF ONE OR MORE BLOCKS

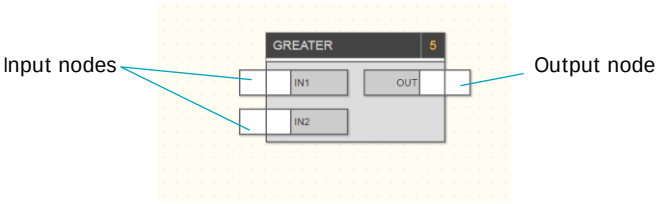
To remove one or more blocks from a Task, do one of the following:

- Select a single block, open the detail window and press the "DELETE" button.
- Select one or more blocks and press the "DELETE" key on the keyboard.

In both cases, the selected blocks are removed from the window after a confirmation message, as are connections to other blocks present in the task itself. This operation cannot be cancelled or undone.

3.8 INPUT AND OUTPUT NODE

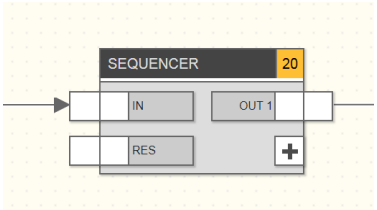
Each block contains at least one input and / or output node, as shown in the following figure:



The input nodes are always on the left side of a block, while the outputs are on the right side. Each node is identified by a synthetic name (e.g. "IN1", "IN2" and "OUT" in the previous figure) that is listed in the Input/Output in the details window, along with a synthetic description of each node.

3.8.1 LOGIC BLOCKS

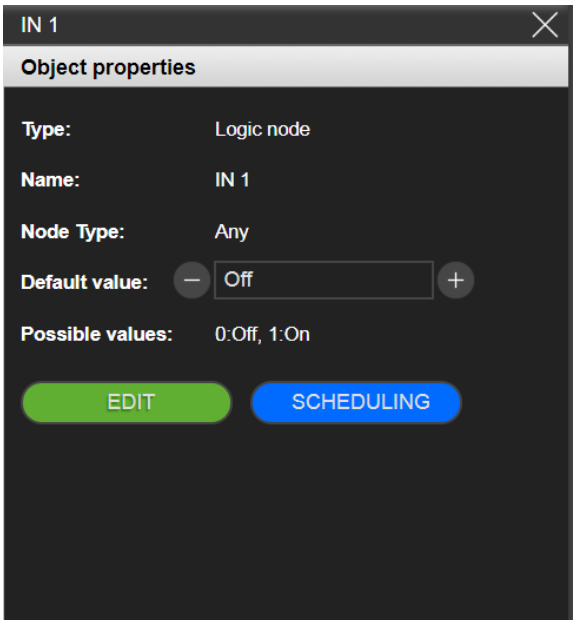
In the case of logic blocks, the input nodes are the 'inputs' to the logic function associated with the block, while the outgoing nodes are the outputs:



In some cases, as in this example, the block provides a variable number of nodes (input or output); in this case, you can use the "+" button to add nodes to the block up to the maximum number.

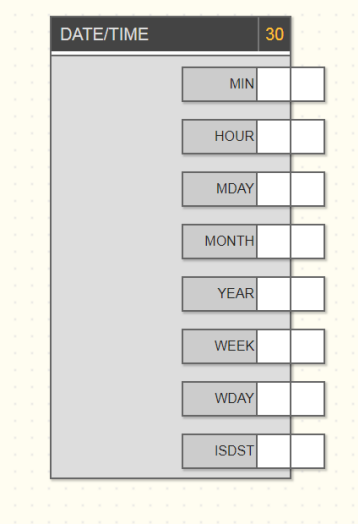
The logic function can only be executed correctly if the input nodes are connected to other blocks and if the output values are transferred to the input nodes of the same number of receiving blocks.

Not all input nodes are absolutely necessary for the correct execution of the logic; if an input node is not connected, the default value is used, which can be changed by selecting the node and opening the corresponding detail area, as shown in the following figure:



The detail window of a node also highlights the possible values it can assume; this information can be useful especially for blocks that have certain combinations or limitations of values.

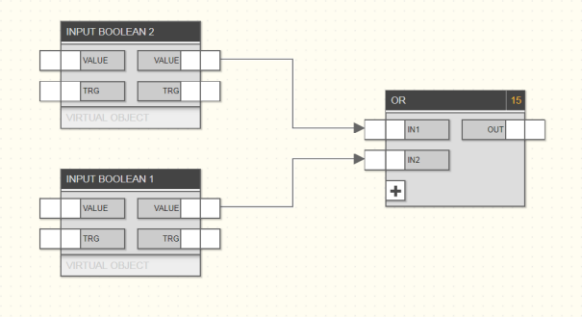
The logic blocks can also only provide outputs, as in the following example (date/time block):



In this case, they can only be used as input for other logics, but they cannot be controlled.

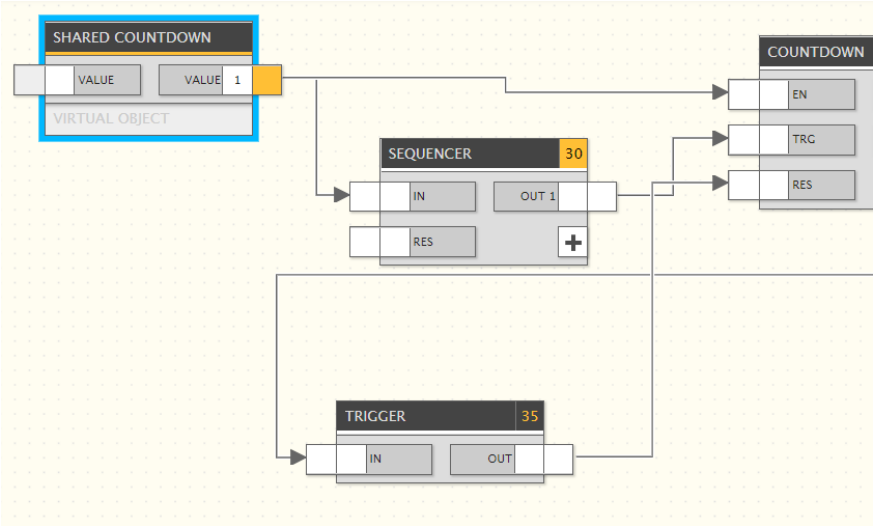
3.9 CONNECTION OF BLOCKS

For the program to actually execute something, it is necessary to provide at least one connection between two nodes of two blocks, so that the value of the first (origin) is passed to the second (destination). To connect two nodes, simply click on the centre of the source node, hold down the mouse button and release it in the centre of the destination node:



If you move the mouse pointer over a connection arrow, it will turn red. By clicking on it, the connection is selected and highlighted in light blue. To delete the selected connection, directly press the "Delete" key on the keyboard.

The origin of a connection must be an output node (right side of a block), while the destination must be an input node (left side); an output node can be the source of multiple connections with different destinations, while an input node is the destination of a single connection.



3.10 EXECUTION SEQUENCE

During the simulation and compilation phases, the editor generates a "list" from the graphically drawn logical networks, which is executed cyclically from start to finish as quickly as possible.

3.10.1 TASK SEQUENCE

Each execution cycle executes the following operations (the cycle time depends on the number and complexity of the tasks):

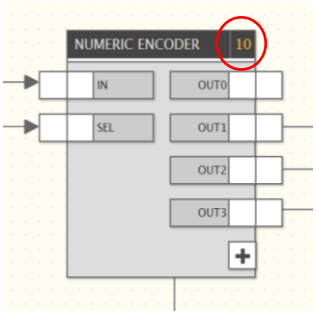
- Reading the inputs from the bus
- Execution of the task 1
- Execution of the task 2
- ...
- Execution of the task n
- Writing commands onto the bus

This means that any interactions between tasks (such as passing values by variables or writing the same node of a block by multiple tasks) are affected by this order, and any actions performed by the tasks towards the end of the queue are not performed by the previous ones until the next execution cycle.

NOTE: If a task is disabled or stopped, it will be "skipped" in the execution cycle, any interaction with the bus and/or other tasks will be suspended in this case.

3.10.2 ORDER OF THE BLOCKS

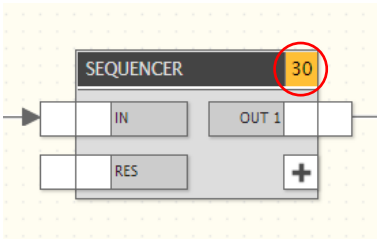
Within each task, the logical blocks also have their own order of execution: the logic module processes the function assigned to the logic blocks in that order. The order of a logic block is shown at its top right, as shown in the following figure:



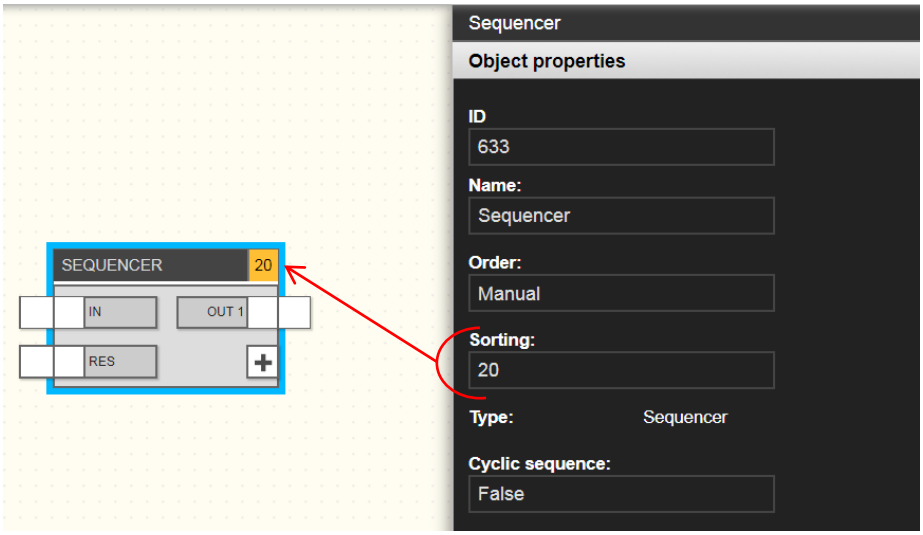
Under normal conditions, the blocks are assigned an ascending order corresponding to the order in which they were inserted into the program; however, it is possible to force a different execution order as follows:

- Select the affected block
- Open the detail window
- Under Order, select "MANUAL".
- Be sure to enter a number that has not already been used.

The blocks with manual sorting are shown as follows:



The following figure shows an example of a logical network with a manual sort block and shows how to change the order in which the blocks are executed:



As mentioned above, the states of the output nodes of all blocks (of all active tasks) are read at the beginning of each execution cycle, and the commands to the input nodes of all blocks (of all active tasks) are sent to the bus at the end of the execution cycle, regardless of the position of the blocks in the tasks and the order of the tasks themselves.

3.11 TRANSFER OF VALUES BETWEEN TASKS

Although each task defines its own logical network, it is possible to pass values between different tasks using certain logical blocks called *variables.* To create a new variable, proceed as follows:

- Open the *library in* the "*Logic Module*" area of the main menu.
- Identify the subitem "*Binary variables*" (if you want to create an ON / OFF variable) or "*Numeric variables*".

- Press the blue "+" key and wait until the new variable has been added to the list.
- Select the new variable and drag it into the first task.

It is possible to assign a name to the variable via the detail window in order to identify it more easily within the programs in which it is used.

If the value of an output node of a logic block is to be assigned to the variable, it is sufficient to connect it to the input node (left side) of the variable; to use this value in other tasks, add the sam variable block to the task(s) and connect the output node (right side) to the input node of another block.

3.12 TYPES OF DATA

The input and output nodes of the blocks can have two types of data:

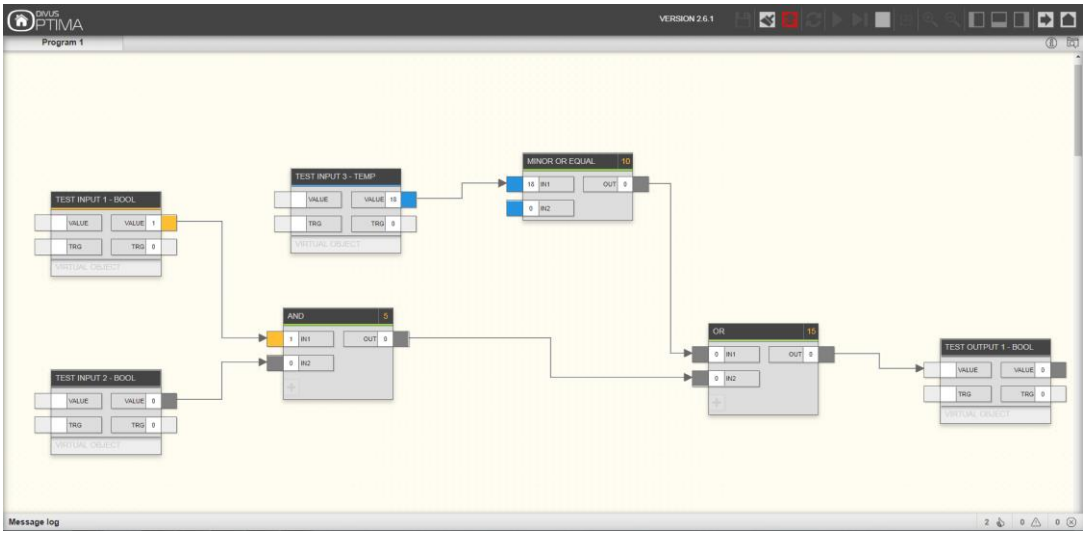
- *Binary*: Only the values 1 (ON) and 0 (OFF) are allowed.
- *Numeric*: Any numeric value is allowed, with certain restrictions depending on the block.

These two data types are not compatible, so the editor prevents the connection of binary nodes to numeric nodes and vice versa: once you start a drag & drop, incompatible nodes become semi-transparent (disabled) and do not accept to create the connection.

3.13 SIMULATION

Before implementing the tasks, it is advisable to test them in the editor via "Simulation" by manually entering values and to check the behaviour of the logical networks - either continuously (repeated execution of the logic in real time) or step by step (i.e. by executing one calculation cycle at a time).

Further information on simulation can be found in Chapter 6.



4 Logics

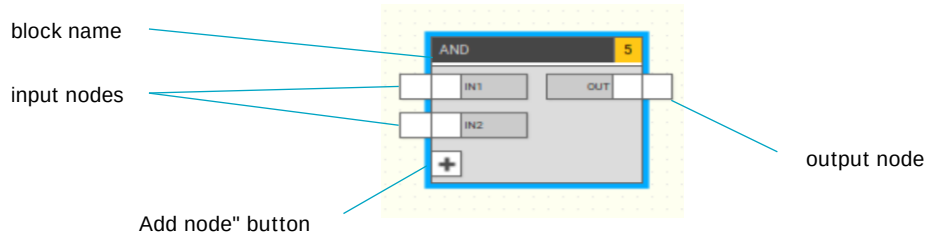
4.1 INTRODUCTION

Logic blocks allow operations to be performed on one or more input values and return one or more output values that can be connected to other logic blocks.

4.2 LOGIC BLOCKS

4.2.1 LAYOUT

As already mentioned, the logic blocks are represented graphically as in the following example:



4.2.2 INPUT NODE

The input nodes allow you to pass values to the logical functions. When you select an input node and open the details pane, you can set the following options:

DEFAULT VALUE	You can specify the value of the node to be used at the beginning of execution until another value is received or if the node is not connected to another block.

In addition to the options mentioned above, the detail window also displays the possible values that the node can assume. In the case of binary nodes, the possible values are only 0 (OFF) or 1 (ON); in the case of numeric nodes, the possible values depend on the type of node and can have specific restrictions.

4.2.3 OUTPUT NODE

The output nodes return the results of the logic function associated with the block and allow it to be passed on to other blocks.

Also, some output nodes of a logical block might be set to have a default value.

4.2.4 ADD AND REMOVE NODES

Some blocks have a variable number of nodes; in these cases, the block taken from the side menu typically contains a minimum set of nodes that can be increased to a maximum number of nodes by pressing the + key.

To remove a previously added node, proceed as follows:

- Select the node
- Open the detail window
- Press the "Delete" button.

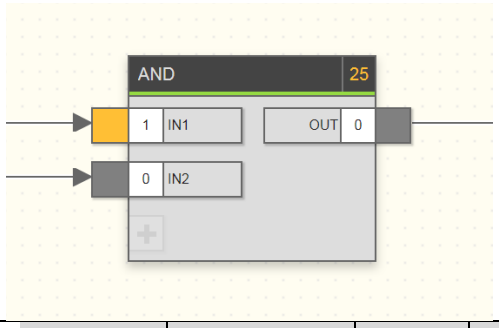
All connections that may be assigned to the node are also deleted.

4.3 COMBINATORIAL LOGIC

4.3.1 AND

DESCRIPTION	Executes the logical AND function between two or more binary inputs (maximum 10).

PREVIEW



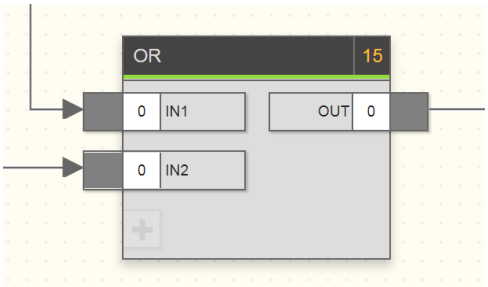
NODES	Tag	Description	Input	Output
	IN1...IN10	Input 1...10 Possible values: 0 → OFF 1 → ON	X	
	OUT	Output Possible values: 0 → OFF 1 → ON		X
	+	Add node	X	

4.3.2 OR

DESCRIPTION

Executes the logical OR function between two or more binary inputs (up to a maximum of 10).

PREVIEW



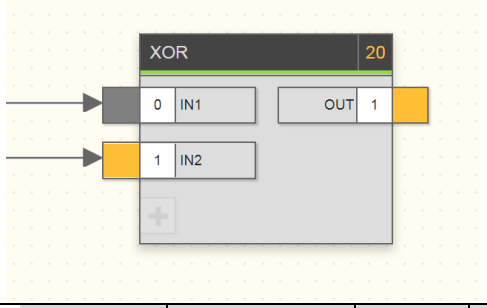
NODES	Tag	Description	Input	Output
	IN1...IN10	Input 1...10 Possible values: 0 → OFF 1 → ON	X	
	OUT	Output Possible values: 0 → OFF 1 → ON		X

	+	Add node	X	

4.3.3 XOR

DESCRIPTION	Perform the logic function XOR between two or more binary inputs (up to a maximum of 10)

PREVIEW



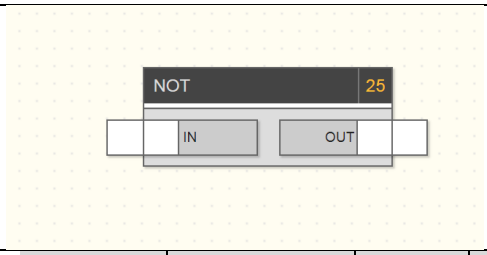
NODES

Tag	Description	Input	Output
IN1...IN10	Input 1...10 Possible values: 0 → OFF 1 → ON	X	
OUT	Output Possible values: 0 → OFF 1 → ON		X
+	Add node	X	

4.3.4 NOT

DESCRIPTION	Executes the logical NOT function of the input.

PREVIEW



NODES

Tag	Description	Input	Output

IN1...IN10	Input 1...10 Possible values: 0 → OFF 1 → ON	X	
OUT	Output Possible values: 0 → OFF 1 → ON		X
+	Add node	X	

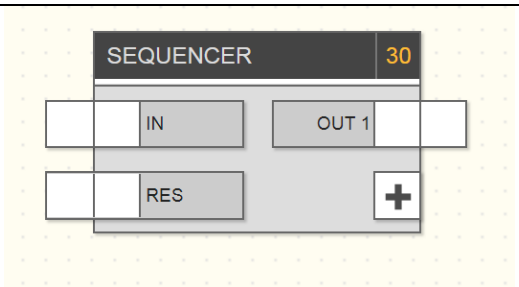
4.4 SCENARIOS AND SEQUENCES

4.4.1 SEQUENCER

DESCRIPTION

When a pulse is received at input IN, it activates and deactivates up to 10 Boolean outputs one after the other, each one remaining active for a specified time.

PREVIEW



NODES

Tag	Description	Input	Output
IN	Sequence start Possible values: 0 → Off 1 → On	X	
RES	Sequence reset Possible values: 0 → Off 1 → On	X	
OUT1...OUT10	Output 1...10 Possible values: 0 → Off 1 → On		X

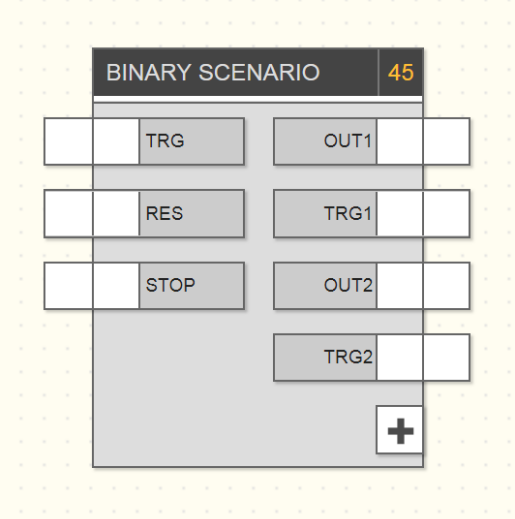
	+	Add node	X
	Options	Cyclic sequence	Defines whether the sequence is to be repeated. Possible values TRUE/FALSE
		Duration of the step 1...10	Waiting time between 2 steps Possible values: from 1 sec. to 12 hrs.

4.4.2 BINARY SCENARIO

DESCRIPTION

When a pulse is received at input TRG, it executes a sequence of Boolean-type commands, each of which can be set, alternating each command with a predefined waiting time, common to all outputs, if necessary.

PREVIEW



NODES

Tag	Description	Input	Output
TRG	Trigger input Possible values: 0 → Off 1 → On	X	
RES	Reset the scene. Possible values: 0 → Off 1 → On	X	

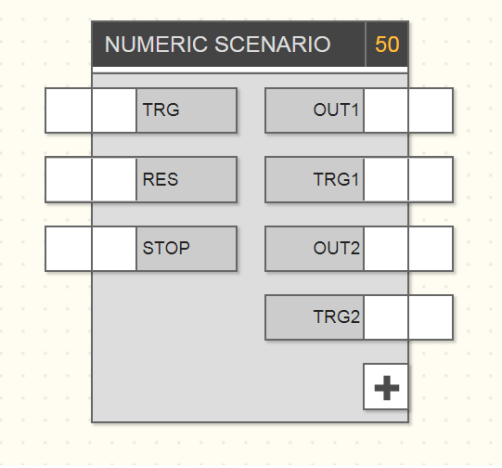
OUT1...OUT10	Output 1...10 Possible values: 0 → Off 1 → On		X
TRG1...TRG10	Trigger 1...10 Possible values: 0 → Off 1 → On		X
+	Add node (and trigger)		X
Options	Output interval	Waiting time between the output commands Possible values: 1...60 sec.	
	Set output 1...10	Value at the outputs 1...10 Possible values: 0 → Wrong (Off) 1 → True (On)	

4.4.3 NUMERIC SCENARIO

DESCRIPTION

When a pulse is received at input TRG, it executes a sequence of commands of the numerical type, each of which can be set, alternating each command with a predefined waiting time, common to all outputs, if necessary.

PREVIEW



NODES

Tag	Description	Input	Output
TRG	Trigger input Possible values: 0 → Off 1 → On	X	
RES	Reset the scene Possible values: 0 → Off 1 → On	X	
STOP	Stop the scene Possible values: 0 → Off 1 → On	X	
OUT1...OUT10	Output 1...10 Possible values: any numeric value		X
TRG1...TRG10	Trigger 1...10 Possible values: 0 → Off 1 → On		X
+	Add Node & Trigger		X
Options	Output interval	Waiting time between the output commands Possible values 1...60 sec.	
	Set output 1...10	Value at the outputs 1...10 Possible values: any numeric value	

4.5 GATES

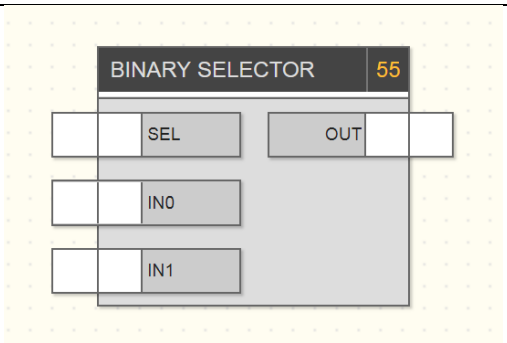
4.5.1 BINARY SELECTOR

DESCRIPTION

Returns the value of one of the inputs based on the value of the SEL input as selector:
If SEL = False → OUT = IN0

If SEL = True → OUT = IN1

PREVIEW



NODES

Tag	Description	Input	Output
SEL	Input selection Possible values: 0 → Off/False i.e. OUT = IN0 1 → On/True i.e. OUT = IN1	X	
IN0 IN1	Inputs 0 and 1 Possible values: 0 → Off 1 → On	X	
OUT	Output Possible values: 0 → Off 1 → On		X

4.5.2 NUMERIC SELECTOR

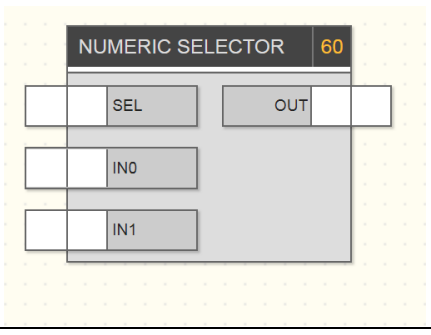
DESCRIPTION

Returns the value of one of the inputs based on the value of the SEL input as selector.

If SEL = False → OUT = IN0

If SEL = True → OUT = IN1

PREVIEW



NODES	Tag	Description	Input	Output
	SEL	Input selection Possible values: 0 → Off/False i.e. OUT = IN0 1 → On/True i.e. OUT = IN1	X	
	IN0 IN1	Inputs 0 and 1 Possible values: Any numeric value	X	
	OUT	Output Possible values: Any numeric value		X

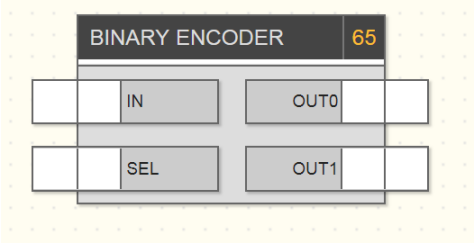
4.5.3 BINARY ENCODER

DESCRIPTION

Sets the value of IN to one of its outputs based on the input value of SEL, which acts as a selector.

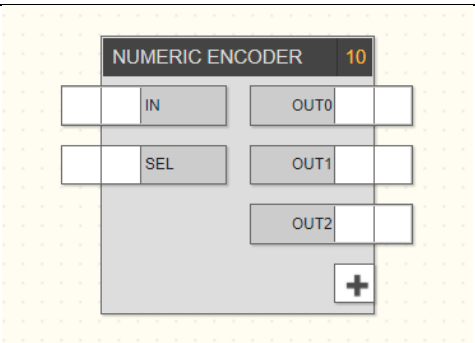
Number of outputs: from 2 to 10

PREVIEW



NODES	Tag	Description	Input	Output
	IN	Input selection Possible values: 0 → Off 1 → On	X	
	SEL	Output selection Possible values: 0 → IN goes to OUT0 1 → IN goes to OUT1	X	
	OUT0...OUT9	Output 0...9 Possible values: 0 → Off 1 → On		X
	+	Add output		X

4.5.4 NUMERIC ENCODER

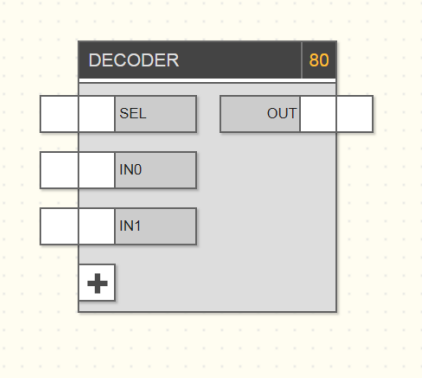
DESCRIPTION	Puts the value of IN (bool) into one of its outputs based on the input value of SEL, which acts as a selector. SEL is numeric. Number of outputs: from 2 to 10
PREVIEW	

NODES	Tag	Description	Input	Output
	IN	Input Possible values: 0 → Off 1 → On	X	
	SEL	Output selector Possible values: 1 → IN goes to OUT0 2 → IN goes to OUT1 … 10 → IN goes to OUT9	X	
	OUT0…OUT9	Output 0…9 Possible values: 0 → Off 1 → On		X
	+	Add output		X

4.5.5 DECODER (OR NUMERIC SELECTOR)

DESCRIPTION	The output returns the value of one of the Boolean inputs based on the input value SEL, which acts as a selector.

PREVIEW



NODES

Tag	Description	Input	Output
SEL	Input selection Possible values: 0 → OUT set to IN0 1 → OUT set to IN1 … 9 → OUT set to IN9	X	
IN0...IN9	Input 0...9 Possible values: 0 → Off 1→ On	X	
OUT	Output Possible values: 0 → Off 1→ On		X
+	Add Input	X	

4.5.6 T FLIP-FLOP

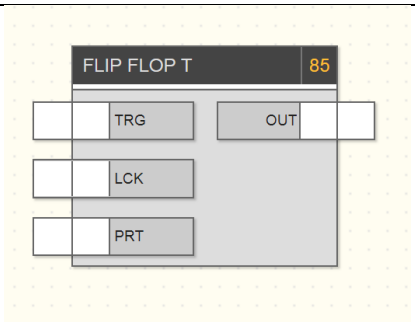
DESCRIPTION

Works like a step-by-step relay. Each time a rising edge (1/On) appears at its input (TRG), the output (OUT) switches state. When the LCK (Lock) input is 1 (True), the TRG effect is locked and there is no output. When the PRT input (priority) is set to 1, the output gets the value set in the parameter VAL (priority value).

Can be used, for example, to control the light of a corridor. You can ensure that the light is normally controlled only when a brightness threshold is met (this condition would be

configured as LCK) and is always on at night (flag that should be connected to the PRT input).

PREVIEW



NODES

Tag	Description	Input	Output
TRG	Trigger Possible values: 0 → Off 1→ On	X	
LCK	Lock Possible values: 0 → Off 1→ On	X	
PRT	Priority flag Possible values: 0 → Off 1→ On	X	
OUT	Output Possible values: 0 → Off 1→ On		X
Options	Priority value	Value of the output if priority flag is set. Possible values: True/False	

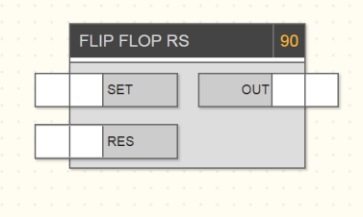
4.5.7 RS FLIP-FLOP (RS → RESET/SET)

DESCRIPTION

Elementary memory block that is loaded with the SET input and reset (meaning the output is set to 0) with the RES (Reset) input. If both inputs are 1, the one set by the "Priority" parameter prevails.

For example, it can be used to manage an alarm signal. An alarm contact must be connected to the SET. Once set to 1, the flipflop will hold the output at 1 until reset by RES. In this way, the information is retained even if the alarm is lowered (to 0).

PREVIEW



NODES

Tag	Description	Input	Output
SET	SET input Possible values: 0 → Off 1 → On	X	
RES	RESET input Possible values: 0 → Off 1 → On	X	
OUT	Output Possible values: 0 → Off 1 → On		X
Options	Select priority	Possible values: 0 → Reset 1 → Set	

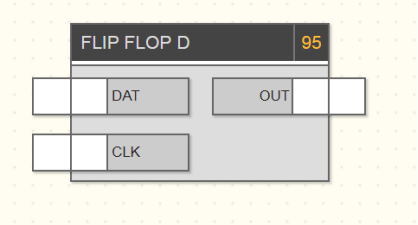
4.5.8 D FLIP-FLOP (D → DATA, AKA DELAY FLIP-FLOP)

DESCRIPTION

The operation is similar to that of the D Latch with the difference that the D Flip-flop acts on the variation of the edge of CLK.

The data in DAT is only sent to OUT on the rising edge of the CLK signal and the value is retained until the next edge of CLK (actually this block forms a memory cell).

PREVIEW



NODES

Tag	Description	Input	Output
DAT	Data Possible values: 0 → Off 1→ On	X	
CLK	Clock Possible values: 0 → Off 1→ On	X	
OUT	Output Possible values: 0 → Off 1→ On		X
Options	All 3 nodes have default value settings (On/Off)		

4.5.9 BINARY D LATCH

DESCRIPTION

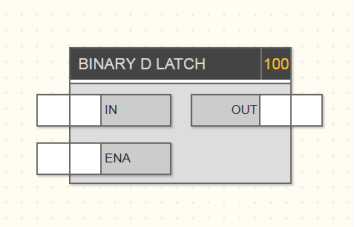
In this block the input signal IN is forwarded to the output OUT when the enable signal ENA is activated (1). When the ENA signal is deactivated, the last state remains at the output (OUT).

If ENA is active again, i.e. (transition 0 -> 1), the last value read in input node IN is sent to output OUT.

If ENA equals 0, the locking block stores the last value read in order to send it at the time when ENA is reactivated.

The data format of IN and OUT is binary.

PREVIEW



NODES	Tag	Description	Input	Output
	IN	Input Possible values: 0 → Off 1→ On	X	
	ENA	Enable Possible values: 0 → Off 1→ On	X	
	OUT	Output Possible values: 0 → Off 1→ On		X
	Options	All 3 nodes have default value settings (On/Off)		

4.5.10 NUMERICAL D-LATCH

DESCRIPTION

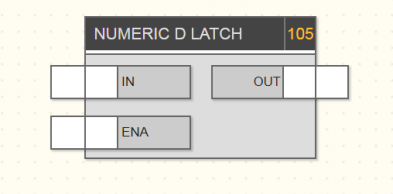
In this block the input signal IN is forwarded to the output OUT when the enable signal ENA is activated (1). When the ENA signal is deactivated, the last state remains at the output (OUT).

If ENA is active again, i.e. (transition 0 1→), the last value read in input node IN is sent to output OUT.

If ENA equals 0, the locking block stores the last value read in order to send it at the time when ENA is reactivated.

The data format of IN and OUT is numeric.

PREVIEW



NODES	Tag	Description	Input	Output

IN	Input Possible values: 0 → Off 1 → On	X	
ENA	Enable Possible values: 0 → Off 1 → On	X	
OUT	Output Possible values: 0 → Off 1 → On		X
Options	The 2 input nodes have default value settings (On/Off)		

4.6 COMPARISONS

RELATIONAL OPERATORS

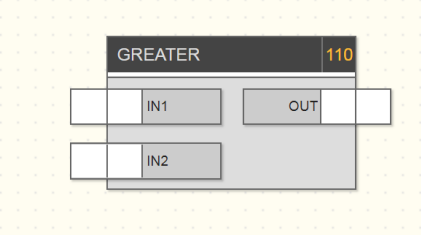
DESCRIPTION

Compares the value of the two inputs and returns TRUE / FALSE as output according to the specific operator.

Available operators:

- Greater
- Equal to or greater than
- Smaller
- Smaller or equal to
- Equal
- Not equal

PREVIEW



NODES

Tag	Description	Input	Output
IN1 IN2	Input 1, 2 Possible values: Any numeric values	X	

	OUT	Comparison result 0 → False/Off 1 → True/On		X

4.7 OPERATIONS

4.7.1 MATHEMATICAL OPERATORS

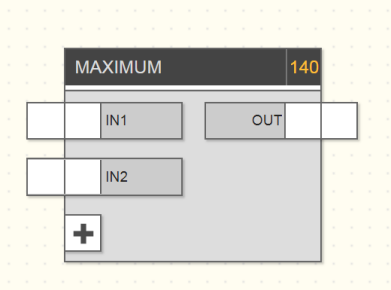
DESCRIPTION

Performs a mathematical operation on the inputs based on the type of the operator

Available operators:

- maximum
- minimum
- average
- sum
- subtraction
- multiplication
- division
- range
- absolute value
- Log10
- integration

PREVIEW (e.g.)



NODES

Tag	Description	Input	Output
IN1 IN2 *	Input 1, 2 Possible values: Any numeric values	X	
OUT	Result Possible values: Any numeric value		X

* The number of inputs can be limited depending on the operation (e.g.: division max. 2, absolute value max. 1).

4.7.1.1 RANGE

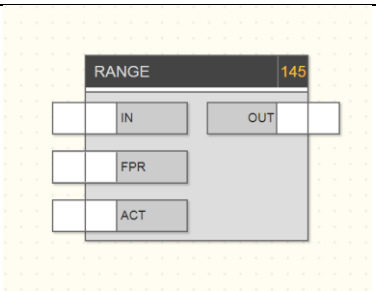
DESCRIPTION

Performs a linear interpolation of the input value IN based on an associated mapping, also called "characteristic" or "feature", defined by two pairs of values (X, Y). The IN value is related between X0 and X1, and this ratio is in turn calculated between the values Y0 and Y1 to determine the initial value.

When the priority mode is set, a pre-set value is returned.

The typical application of this block is the conversion of values between different sizes.

PREVIEW



NODES

Tag	Description	Input	Output
IN	Input Possible values: Any numeric value	X	
FPR	Priority activated Possible values: 0 → no 1 → Priority value is output	X	
ACT	Direct/inverted function Possible values: 0 → direct 1 → inverted	X	
OUT	Output Possible values: Any numeric value		X
OPTIONS	X0 Y0 X1 Y1	Characteristics of linear interpolation Possible values: Any numeric value	
	Priority value	Value to be output if priority is set	

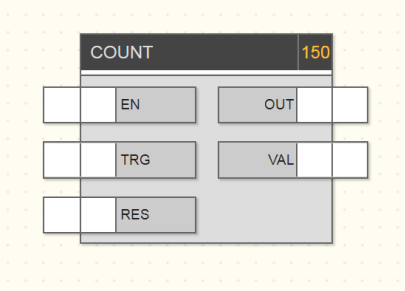
4.8 COUNTERS

4.8.1 COUNTER, COUNTDOWN, UP/DOWN COUNTER

DESCRIPTION

Counts the number of pulses received at the input (TRG) and increases or decreases each time it is received, depending on the counter type.
Counter types: Counter, countdown, up/down counter

PREVIEW



NODES

Tag	Description	Input	Output
EN	Enable Possible values: 0 → not enabled 1 → enabled	X	
TRG	Trigger Possible values: 0 → Off 1 → On (increments counter)	X	
RES	Reset Possible values: 0 → Off 1 → On (counter reset)	X	
OUT	Output Possible values: 0 → Off 1 → On		X
VAL	Current value Possible values: Any numeric value		X
Options	Preset	Default value that is set when the reset is	

		performed or when the logic is started. Possible values: Any numeric value.

4.9 TIMER & SCHEDULES

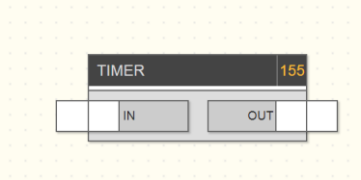
4.9.1 TIMER

DESCRIPTION

Delays the received input value by a pre-set time.

If a 1 is received at the input (IN) (rising edge), an internal counter starts until the time specified as "rising delay", then the output is raised to 1; conversely, if a 0 is received at the input (falling edge), the block waits for the time specified as "falling delay" before setting the output to 0.

PREVIEW

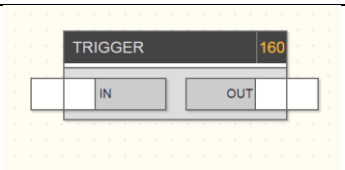


NODES

Tag	Description	Input	Output
IN	Input Possible values: 0 → Off 1 → On	X	
OUT	Output, delayed by timer Possible values: 0 → Off 1 → On		X
Options	Increasing delay	Delay of the forwarding of the rising edge received at the input Possible values: from 1 second to 24 hours	
	Falling delay	Delay of the forwarding of the falling edge received at the input.	

		Possible values: from 1 second to 24 hours

4.9.2 TRIGGERS

DESCRIPTION	Generates a trigger at an edge detected at the input (pulse of one cycle duration). If it receives at input 1, it sets the output to 1 for the duration of a single processing cycle, then the output is reset to 0. In this way it is possible to generate a "pulse" for logic blocks that require it (e.g. scenarios, sequencers, etc.) on the rising edge of the input.
PREVIEW	

NODES

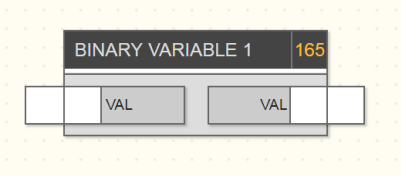
Tag	Description	Input	Output
IN	Input	X	
OUT	Pulse		X
Options	Edge	Rising or falling edge to be detected at the input.	

4.10 VARIABLES

4.10.1 FOREWORD

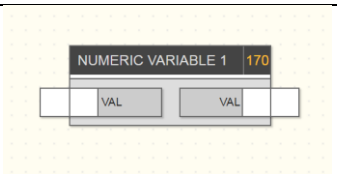
As shown in Section 3.11 variables can pass values between different tasks. The variables must first be created using the "+" button in the corresponding section of the main menu so that they can be dragged into the tasks they are to use.

4.10.2 BINARY VARIABLES

DESCRIPTION	Allow to transfer a Boolean value between different tasks.
PREVIEW	

NODES	**Tag**	**Description**	**Input**	**Output**
	VAL	Input Possible values: 0 → Off 1→ On	X	
	VAL	Current value Possible values: 0 → Off 1→ On		X

4.10.3 NUMERIC VARIABLES

DESCRIPTION	Allow to transfer a numeric value between different tasks.
PREVIEW	

NODES	**Tag**	**Description**	**Input**	**Output**
	VAL	Value the variable receives Possible values: any numeric value	X	
	VAL	Current value of the variable Possible values: any numeric value		X

4.11 CONSTANTS

You may need to use constants for your logics (e.g. for comparisons). Here you can create constants and assign them the desired value. Then drag them to your logic.

5 Triggers

5.1 INTRODUCTION

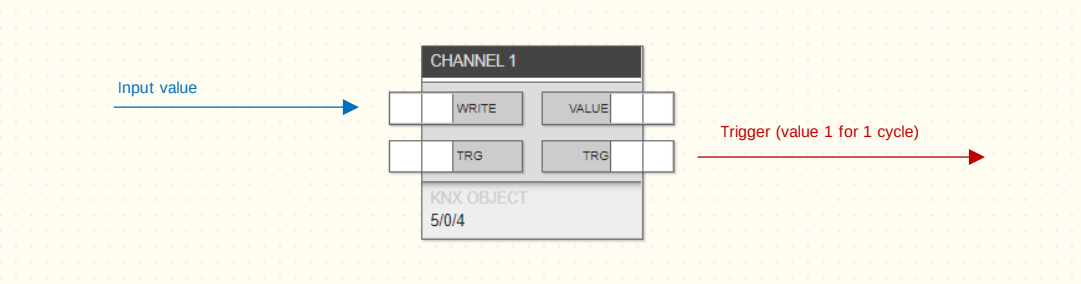
In normal conditions, the logic module keeps the state of the objects at the outputs according to the state of the logic; as soon as these objects change their state (e.g. because they receive a command from the bus, outside of the logic), their state is changed back to the one corresponding to the logic.

This is not always the desired behaviour; in many situations indeed, you'll want to command an object based on the change of state of other objects, but if this object is changed from outside (the logic), you want it to keep the new state. In other words, you'd prefer an event driven logic.

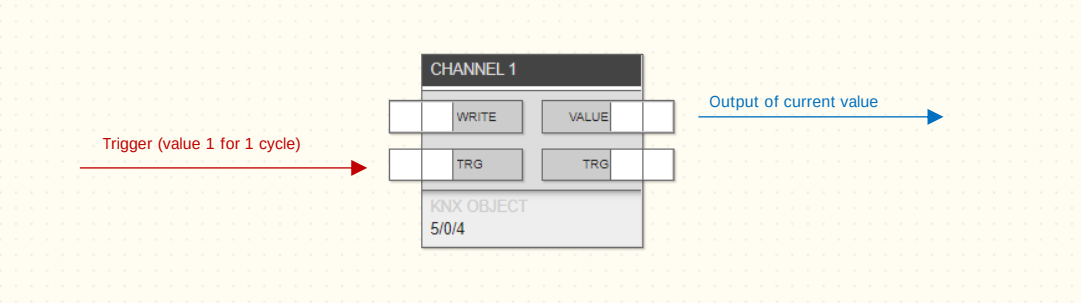
In such cases, the use of triggers for input and output is needed.

5.2 INPUT AND OUTPUT TRIGGERS

Output triggers produce a pulse (i.e. the value 1 for one single execution cycle of the script, then reset to 0) each time a value is received on a certain object (input).



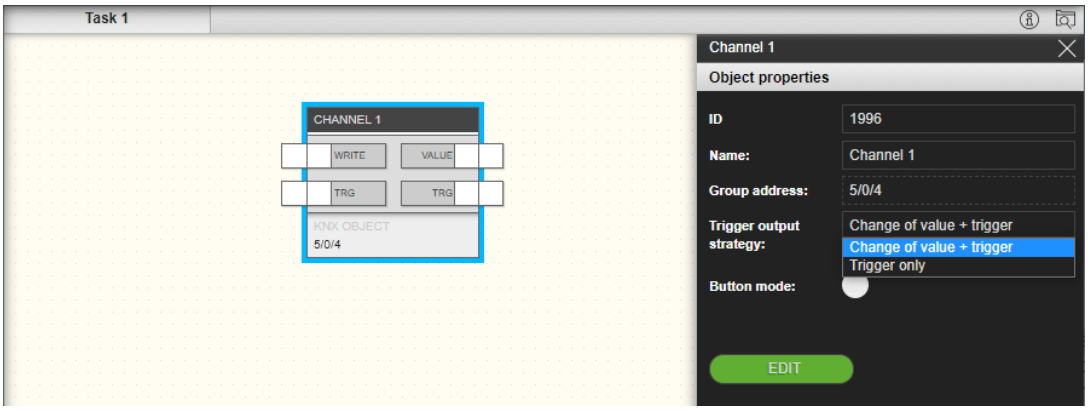
Input triggers, on the other hand, allow to choose when to emit a value: every time a pulse (value 1) is sent to this trigger, the value of the object is sent to the bus – even if it did not change since the previous execution cycle.



5.3 OUTPUT STRATEGY

As explained before, **in normal conditions the output nodes of an object within a task are commanded according to the value change of the object / block**. Moreover, if a pulse is received on the input trigger (TRG node at the left of the block) the value is sent to the bus as well – even if not changed.

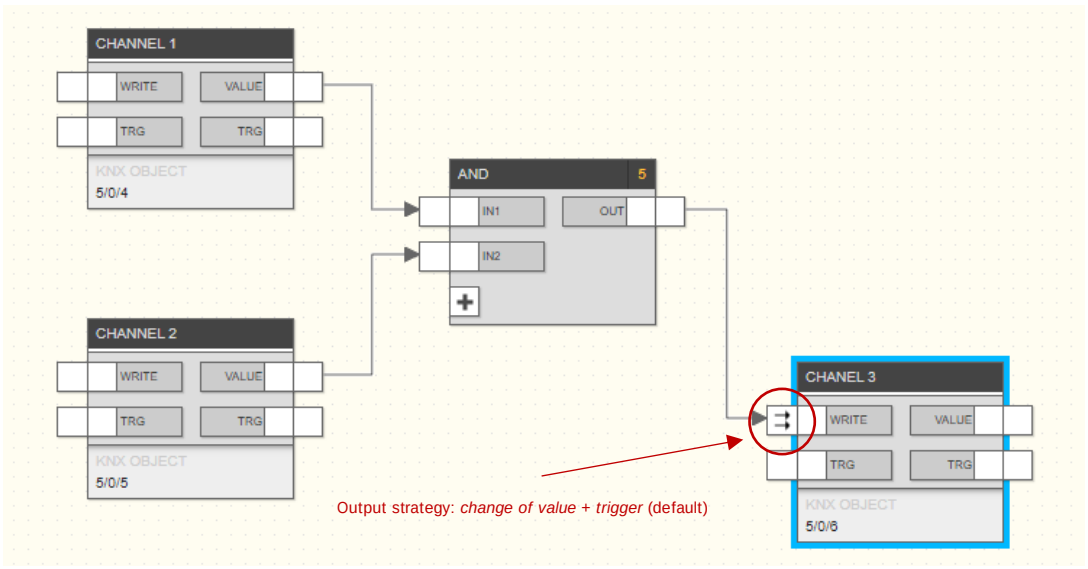
But it is possible to set the output nodes to send the value to the bus only by trigger changing the **output strategy** parameter in the details panel of the block, after selecting the block in the logic editor.



Thus, a value is sent to the bus only when a pulse reaches the block's input trigger node. Otherwise, no values is sent – no matter whether the logic's output upfront has changed or whether the value of the object was changed from outside the logic (e.g. by the user through a physical button).

5.4 TRIGGER CONNECTIONS

Take for example the following simple logic:

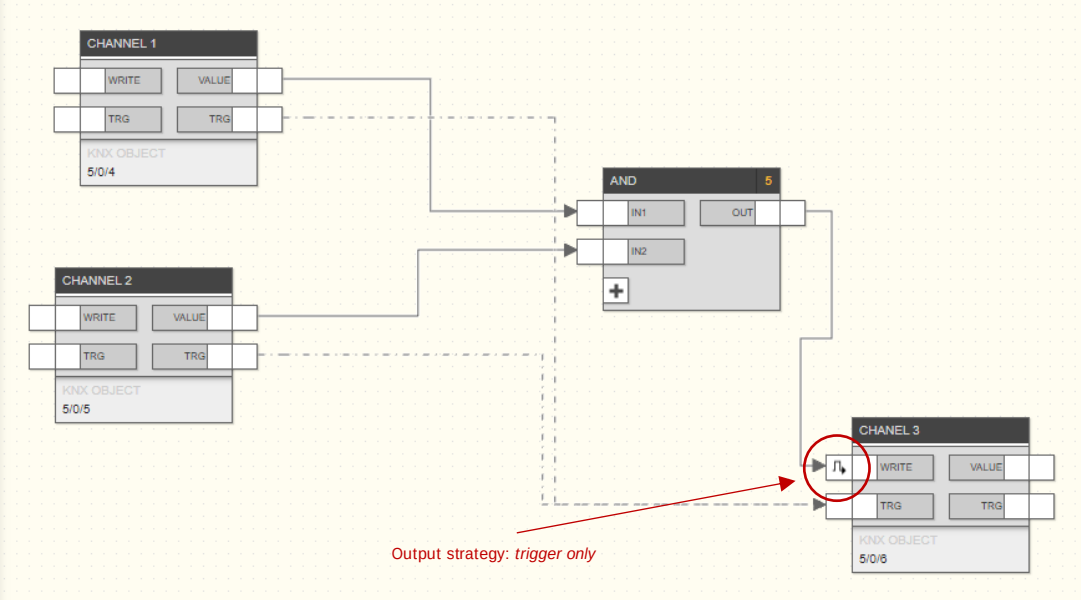


As shown, the output block has change of value + trigger as output strategy, which means that:

- If one of the inputs (Channel 1 or Channel 2) changes, Channel 3 gets the value of the AND between the 2 inputs
- If Channel 3 changes its value for reasons outside of the logic (e.g. a manual command by a user) it is immediately re-aligned to the value of the AND logic.

If you want Channel 3 to be changed **only when one of the inputs changes** but leaving it free to be changed manually (i.e., in any way, from outside the logic), you'll need to:

- Set the output strategy to **trigger only**
- Connect the output triggers of Channel 1 and Channel 2 to the input trigger of Channel 3
- To allow this, output triggers can be linked to multiple inputs whereas usually, input nodes can have a single input connection per node within a task.



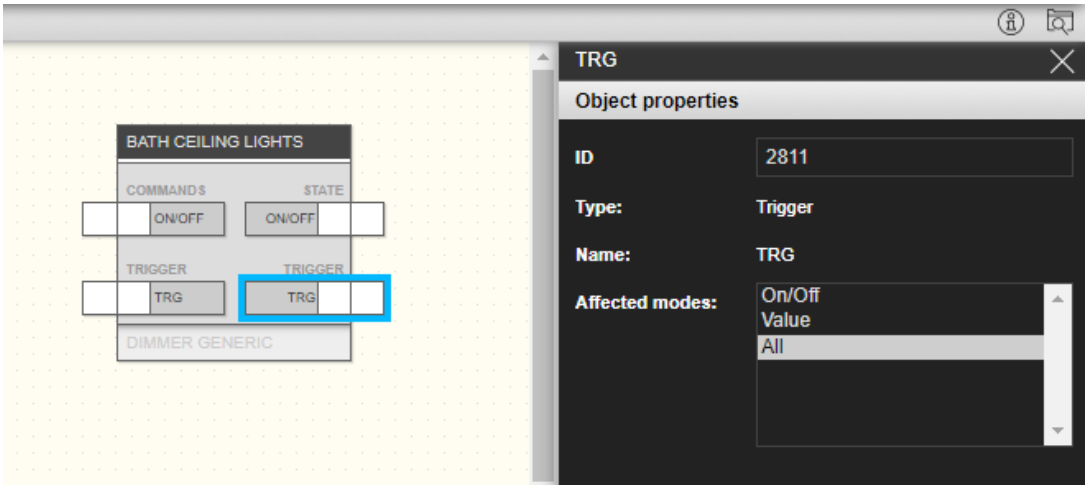
When the input trigger is linked, its output strategy automatically switches to "trigger only" and the shown symbol on the node consequently also.

It is still possible to manually adjust the output strategy; it could be used e.g. to send its value periodically by using a periodic logic upfront, in addition to its value changes, like many KNX devices do.

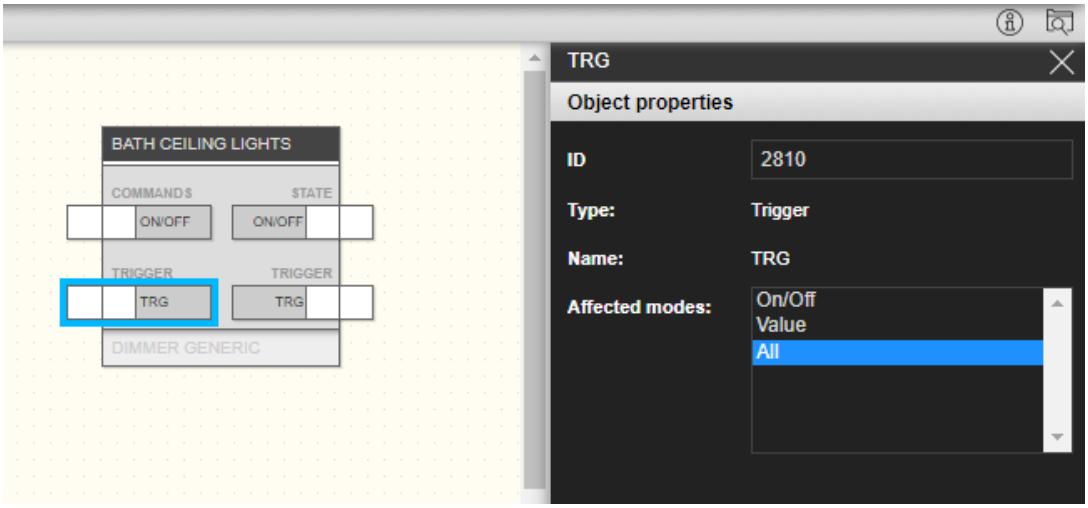
5.5 TRIGGERS AND COMPLEX OBJECTS

For Complex Objects, one single TRG is available (not one for each sub-object). In this case, it is possible to specify to which sub-object the trigger should refer to in the details panel (after selecting the trigger node).

In the case of output triggers, it is possible to select which sub-object will determine a pulse on change of state through the interested nodes selector. As default, all sub-objects are active thus any value change will generate a pulse in the logic, but it is possible to reduce the selection to a sub-set or to a single sub-object.



Similarly, for input triggers it is possible to define which sub-objects should be commanded when a pulse is received.



6 Connections (links)

6.1 INTRODUCTION

As seen before, to create a logical task, you need to connect output nodes to input nodes of different blocks. This chapter discusses the connection types, their options and the impact they have on the LUA code at the base of a Logic unit.

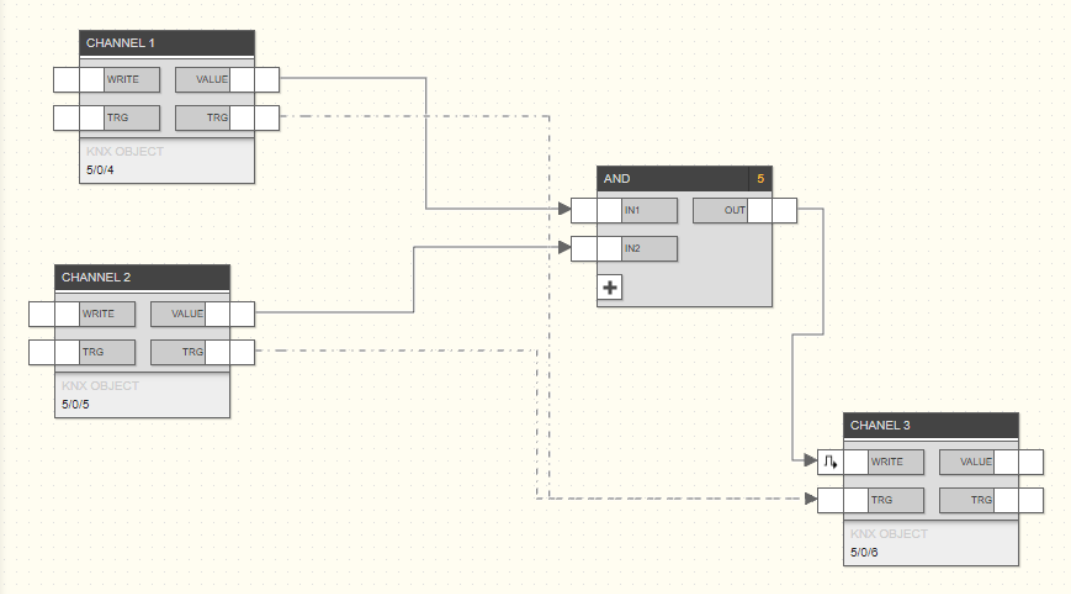
6.2 CONNECTION (LINK) TYPES

Depending on the node type they link, connections may be of 2 types:

- Standard connections
- Trigger connections

The first ones connect output and input nodes, be it Optima objects or logical blocks from the library, passing Boolean or numeric values (depending on the block type). These connection types represent the passage of a value between an output node of one block to the input node of another block during the execution of the Logic unit.

Trigger connections, on the other hand, pass a pulse of the duration of one single execution cycle when it is generated by the TRG node of an Optima object; usually this pulse is linked to one or more other input TRG nodes of other objects to have an output only when triggered (as explained in the chapter about triggers) but it may also be used for any other type of input of the logic's network, together with other triggers or Boolean values. The connections coming from output triggers are shown in the editor as dashed-dotted lines, as you can see here:



6.3 LOGIC BLOCK OUTPUT CONNECTION PROPERTIES

When you select an outgoing connection from a logic block, further options are given in the detail panel to decide when an output should be generated (i.e. move from the logic block to the connected destination block).

To see those options, move over such a connection arrow and click it (it becomes light blue).

Conditions allow to define when an output should move from a logic block to. The options are:

- Every status change (for Boolean and numeric output types)
- Never (for Boolean and numeric output types)
- If Off (only for Boolean types)
- If On (only for Boolean types)

The value field allows to choose what value should be passed for Boolean types. The options are

- Current value
- Inverted value
- On
- Off

So, depending on the outgoing data type, you can easily choose any combination of the above options to fine tune the output as desired.

7 Simulation

7.1 INTRODUCTORY REMARKS

Once a logical task has been realized, it is possible to simulate its operation within the editor by manually inserting the status of the inputs and checking the processing of the outputs in real time, also through the logic blocks that contain a variation of the outputs over time.

7.2 SIMULATION TYPES

Two types of simulation are available:

- **Continuous simulation:** The execution of the task takes place in the background and is influenced by changes in the node status in real time.
- **Step-by-step simulation**: Each task execution cycle must be started manually, and the status of the nodes can be changed between one cycle and the next.

The first type allows a more realistic evaluation of the realized logical networks, the second one allows a thorough and punctual verification of every single value passage between blocks and offers a higher diagnostic level.

7.3 GRAPHICAL SIMULATION ENVIRONMENT

By pressing one of the simulation keys (continuously or step by step), the editor window changes as follows:

- The detail window is closed to provide maximum working space for the simulation.
- Every operation of drag & drop, linking, changing or deleting the contents of Logic units is blocked.
- The nodes take on a colour depending on their state and allow you to force the value manually (as described below).

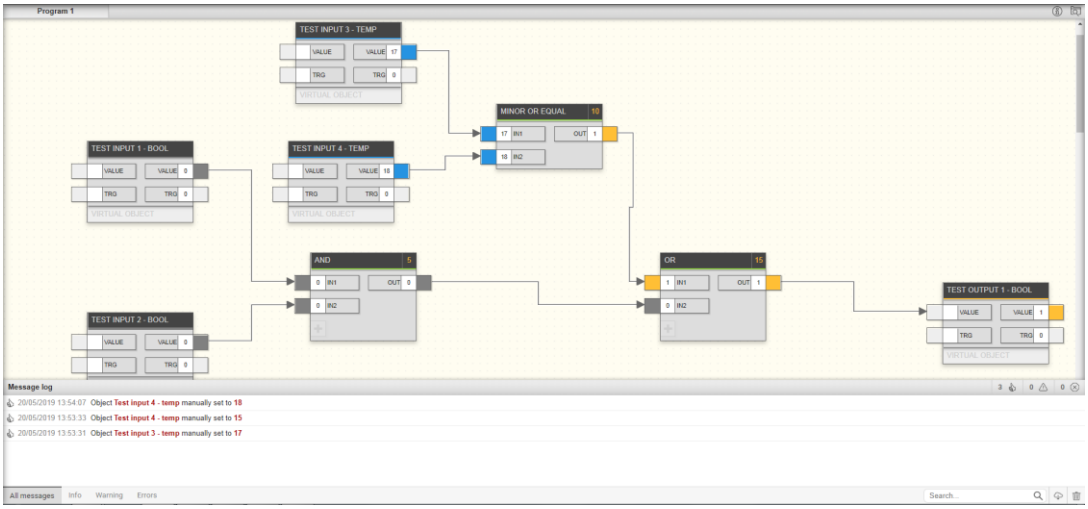
The colour of the nodes follows the following convention:

binary node	grey	Value 0 (OFF)
	yellow	Value 1 (ON)
Numerical nodes	blue	Any value

During simulation, the editor displays a series of information in the notification area about program execution, manual status changes (made by the user), and automatic status changes (detected by logic blocks). In addition, many "debug" messages are reported during the step-by-step simulation, allowing a thorough analysis of the program execution, which is particularly useful in case of errors or malfunctions.

The notification area, which is normally closed to provide maximum space for the simulation, can be opened to read the messages whose number - depending on the type - is visible in the right part of the message bar, even if it is closed. See Section2.6 more details on the notification area.

The following figure shows an example of a simulation with an open notification area:



7.4 MANUAL VALUE INPUT

To set the status of a node manually, proceed as follows:

- Double-click the node value
- delete the current value and enter the new value
- *Enter* Press

The colour of the node (if binary) changes according to the new value and it is passed to the simulator, which propagates it immediately (in case of continuous simulation) or to the next execution cycle (in step-by-step mode).

7.5 STOP SIMULATION

The simulation can be stopped at any time by pressing the "Stop" button in the toolbar (not accessible outside the simulation).

8 Drawing Tools

8.1 INTRODUCTORY REMARKS

To increase the readability of Logic units, especially for complex logical networks, the editor provides several drawing tools that allow the user to enter comments and highlight areas of the program.

These tools are available in the "Logic module" section of the main menu under the "Drawing tools" item. As already mentioned, they can be dragged into the Logic units like the other types of objects.

8.2 LABELS

With the labels you can insert free text into the tasks. It is possible to insert an unlimited number of labels for each logical task.

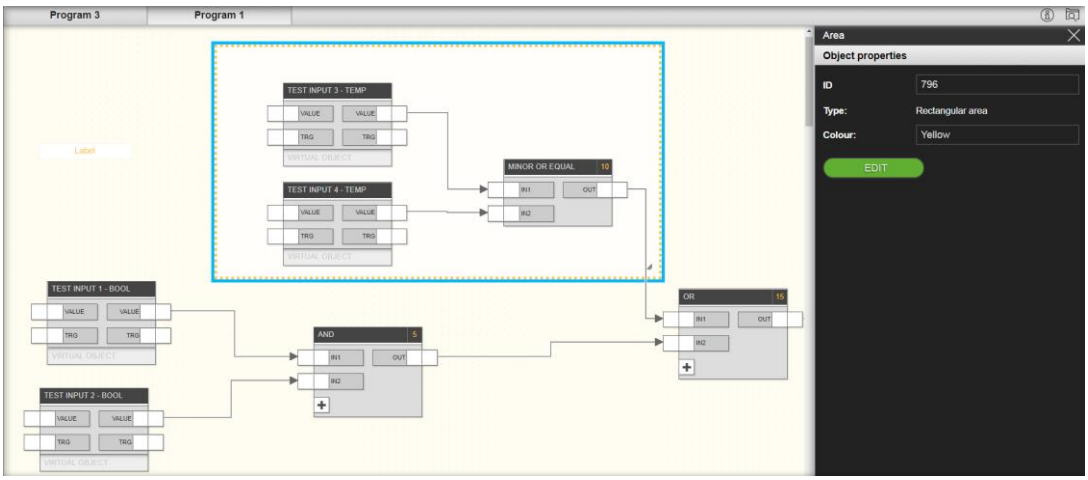
Once a label has been dragged into a task and positioned at the desired point, it is possible to customize it by opening the details window (after it has been selected); the following options are available:

- **text** Text that is displayed in the logical task.
- **colour** Allows you to select the colour of text (black, yellow, red, green, blue, grey)
- **font size** Allows to select the font size (XS, S, N, L, XL)

Labels can be removed from tasks by directly pressing the REMOVE key on the keyboard after they have been selected.

8.3 RECTANGULAR AREAS

It is possible to highlight one or more parts of the logic unit by dragging colored rectangular areas from the main menu, as shown in the following figure:



Once you drag a rectangular area into a program, you can:

- resize by dragging the corresponding cursor in the lower right corner
- Change the border colour using the colour selection field in the details pane.

The rectangular areas are always drawn under the blocks and their connections; they do not support multiple selections such as blocks or labels. To customize them or remove them from the program, you must click them one at a time and use the tools in the Details pane (change colour and button) or press the "Remove" key on the keyboard to remove them from the Task.

9 Appendix

9.1 GLOSSARY

Logical module	The catalogue nomenclature is "Logical Module".
Logic unit	A logical network consisting of one or more logical tasks.
Task	A logical network consisting of one or more linked blocks and logics. Each logical unit can contain up to 16 logical tasks.
Logic block	Block that can be inserted into a Logic unit to perform a particular function and interact with other blocks via input and/or output nodes.
Block	Object which can be inserted into a Logic unit to read and/or write information on the home automation bus and to interact with other blocks via input and/or output nodes
Node	Single element of a logic block that provides certain information in the input or output; the nodes can be connected to other *nodes* via corresponding connections.
Arrow / connector	Link between two nodes of two blocks. The connection has a "direction" that determines the order in which information is exchanged between nodes; in particular, the status of the source node is passed to the target node.
Editor	Graphical configuration environment for Logic units. It enables you to create logical tasks for the logical units present in the project and to obtain the information within the project required for their execution.

9.2 NOTES