



Advanced Logic Module - Manual

Version 1.0

REV01-20200518

---

## GENERAL INFORMATION

---

DIVUS GmbH  
 Pillhof 51  
 I-39057 Eppan (BZ)

Operating instructions, manuals and software are protected by copyright. All rights reserved. Copying, duplicating, translating, translating in whole or in part is not permitted. An exception applies to the creation of a backup copy of the software for personal use.

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	7
1.1	INTRODUCTORY REMARKS	7
2	USER INTERFACE	8
2.1	GENERAL LAYOUT	8
2.2	NAVIGATION MENU	8
2.2.1	LOGIC PROGRAMS	8
2.2.2	LIBRARY	8
2.2.3	DRAWING TOOLS	8
2.3	TOOLBAR	9
2.4	DETAIL AREA	9
2.5	WORK AREA	10
2.6	NOTIFICATION AREA	10
3	LOGICAL PROGRAMS	12
3.1	INTRODUCTORY REMARKS	12
3.2	CREATE A NEW LOGICAL PROGRAM	12
3.3	CREATE A NEW TASK	12
3.4	REMOVE OR DEACTIVATE A PROGRAM	13
3.5	ADDING BLOCKS TO A TASK	13
3.6	SELECT ONE BLOCK OR MULTIPLE BLOCKS	14
3.7	DISTANCE OF ONE OR MORE ORDER BLOCKS	15
3.8	INPUT AND OUTPUT NODE	15
3.8.1	LOGICAL BLOCKS	16
3.9	CONNECTION OF BLOCKS	17
3.10	EXECUTION SEQUENCE	18
3.10.1	TASK SEQUENCE	18

3.10.2	ORDER OF BLOCKS	19
3.11	TRANSFER OF VALUES BETWEEN TASKS	20
3.12	TYPES OF DATA	21
3.13	EXECUTION OF LOGIC ON BUS	21
3.14	SIMULATION	21
4	LOGICS	23
4.1	INTRODUCTORY REMARKS	23
4.2	LOGIC BLOCKS	23
4.2.1	LAYOUT	23
4.2.2	INPUT NODE	23
4.2.3	EYE NODE	24
4.2.4	ADD AND REMOVE NODES	24
4.3	COMBINATORIAL LOGIC	24
4.3.1	AND	24
4.3.2	OR	25
4.3.3	XOR	26
4.3.4	NOT	26
4.4	SCENARIOS AND SEQUENCES	27
4.4.1	SEQUENCER	27
4.4.2	BINARY SCENARIO	28
4.4.3	NUMERIC SCENARIO	29
4.5	GATE	31
4.5.1	BINARY SELECTOR SWITCH	31
4.5.2	NUMERICAL SELECTOR SWITCH	32
4.5.3	BINARY ENCODER	32
4.5.4	DECODER	33
4.5.5	T FLIP-FLOP	34

4.5.6	RS FLIP-FLOP	35
4.5.7	D FLIP-FLOP	36
4.5.8	BINARY D-LOCKING	36
4.5.9	NUMERICAL D-LOCK	37
4.6	COMPARISONS	38
4.7	OPERATIONS	39
4.7.1	MATHEMATICAL OPERATORS	39
4.7.1.1	RANGE	40
4.8	COUNTERS	41
4.8.1	COUNTER, COUNTDOWN, UP/DOWN COUNTER	41
4.9	TIMER & SCHEDULES	42
4.9.1	TIMER	42
4.9.2	TRIGGERS	43
4.10	VARIABLES	44
4.10.1	FOREWORD	44
4.10.2	BINARY VARIABLES	44
4.10.3	NUMERIC VARIABLES	45
5	SIMULATION	46
5.1	INTRODUCTORY REMARKS	46
5.2	SIMULATION TYPES	46
5.3	GRAPHICAL SIMULATION ENVIRONMENT	46
5.4	MANUAL VALUE INPUT	47
5.5	STOP SIMULATION	48
6	DRAWING TOOLS	49
6.1	INTRODUCTORY REMARKS	49
6.2	INSCRIPTIONS	49
6.3	RECTANGULAR AREAS	50

7    APPENDIX \_\_\_\_\_ 51

7.1   GLOSSARY \_\_\_\_\_ 51

7.2   NEED \_\_\_\_\_ 52

# 1 General Overview

---

## 1.1 INTRODUCTORY REMARKS

---

The Advanced Logics module allows complex logical networks to be implemented within the DIVUS KNX SERVER 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 **logical programs, which** can consist of one or more **tasks**; each task is a logical network configurable via a graphic editor, similar to the KNX SERVER *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 program (simulated or "real"), a script in the LUA language is generated that runs in the background and continues by exchanging input and output data in a loop, conceptually similar to what happens in PLCs.

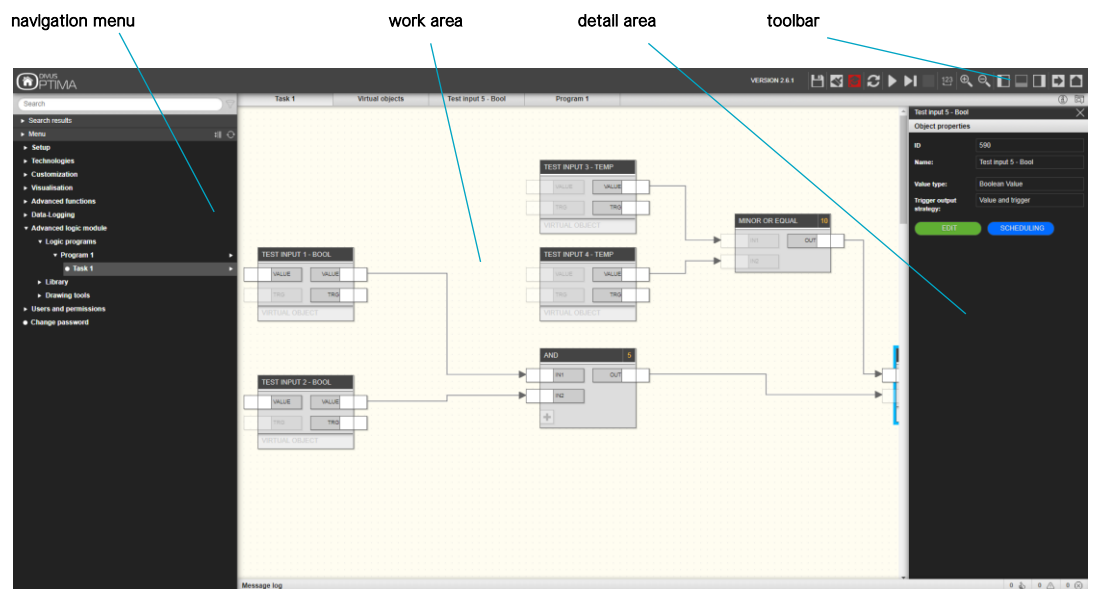
*Logical programs* can be run in parallel - they are independent processes; conversely, the *tasks are* parts of a *logical program* and flow into the LUA script during compilation. Therefore, all enabled tasks are always executed, from the first to the last, in the order specified on the program configuration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

Unlike the other logical functions of KNX SERVER, which are processed "on event" (i.e. when a status change occurs at one of the inputs), the LUA logic programs run continuously and keep the output objects in their desired state if they change state due to an external event.

## 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 logical programs. To do this, click on *Advanced logic module*.

#### 2.2.1 LOGIC PROGRAMS

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

#### 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 logical programs:



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

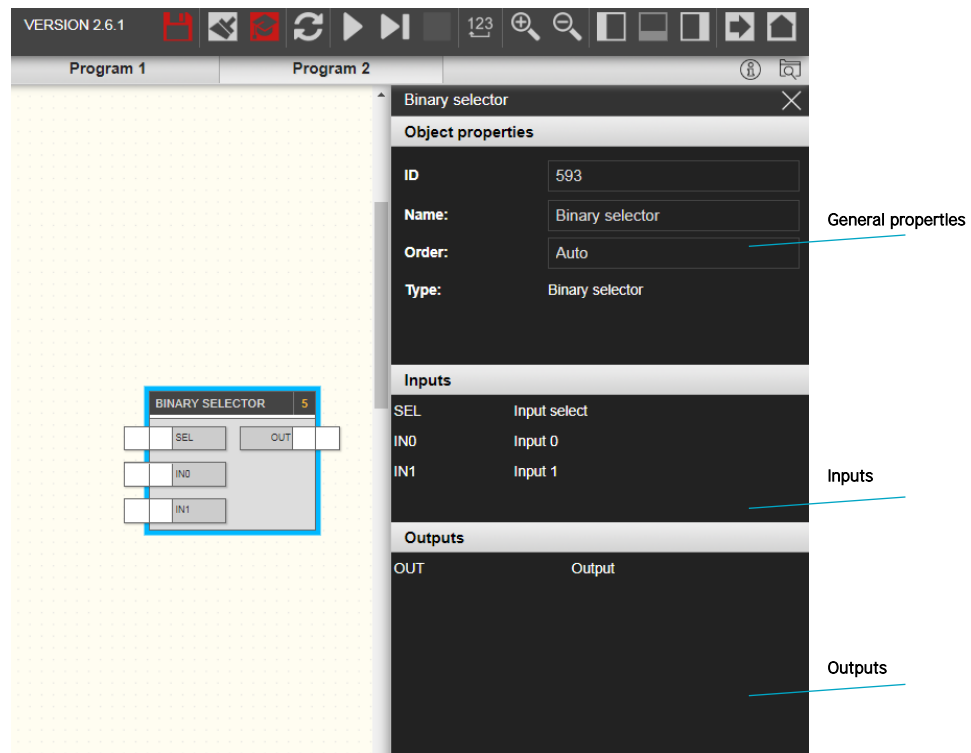
---

## 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:

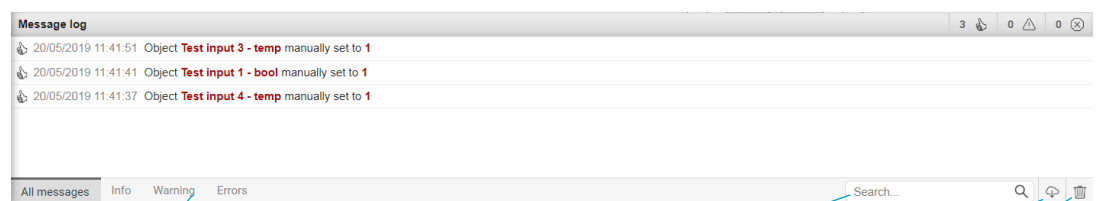


## 2.5 WORK AREA

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 programs.

## 2.6 NOTIFICATION AREA

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



Filter by type

SearchExport

Delete

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 Logical Programs

---

## 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 program can contain up to 16 tasks.

The editor allows you to configure logic programs 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 logical programs.

## 3.2 CREATE A NEW LOGICAL PROGRAM

---

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

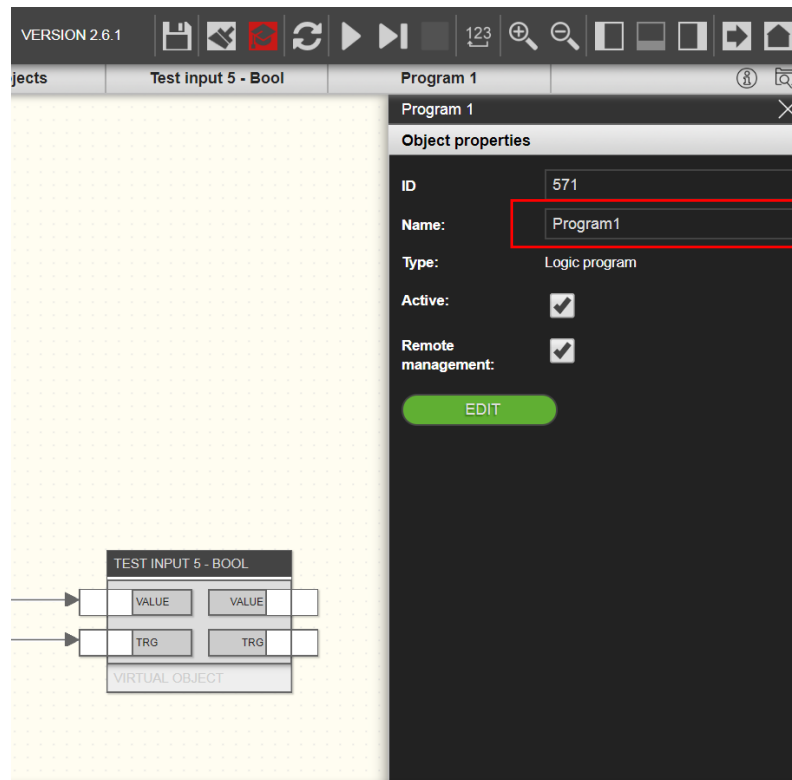
## 3.3 CREATE A NEW TASK

---

To create a new task, first select the logical program you want to work on in the Logic Programs 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 PROGRAM

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

If you do not want a program 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 programs are marked with a semi-transparent effect.

### 3.5 ADDING BLOCKS TO A TASK

The programs 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 objects are listed here.

Once the block 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:

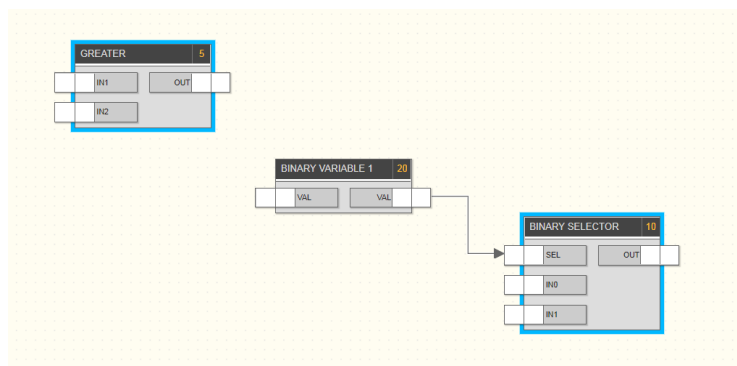
---

### 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).

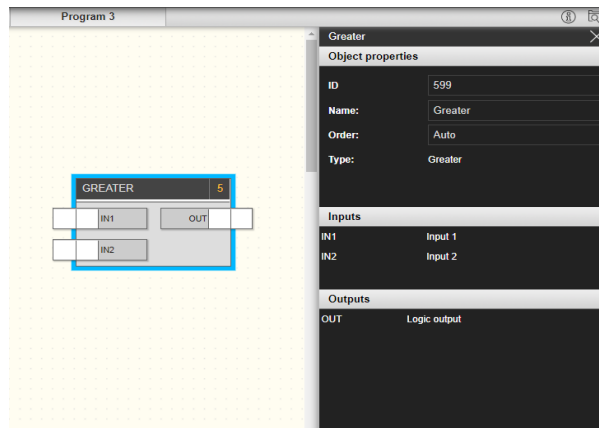


The selected blocks are highlighted with a blue border:

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 DISTANCE OF ONE OR MORE ORDER BLOCKS

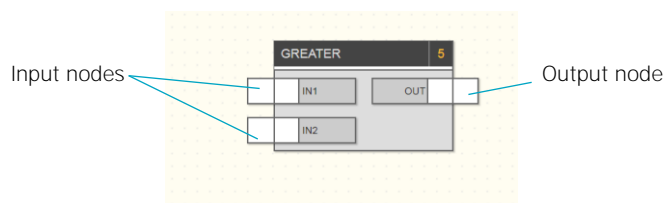
To remove one or more blocks from a Task, do one of the following:

- 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 canceled or undone.

### 3.8 INPUT AND OUTPUT NODE

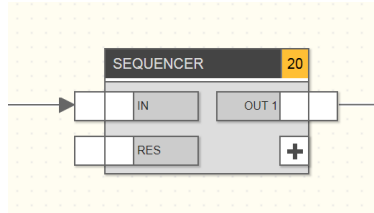
Each block contains at least one input and / or output node, as shown in the following figure:



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 LOGICAL BLOCKS

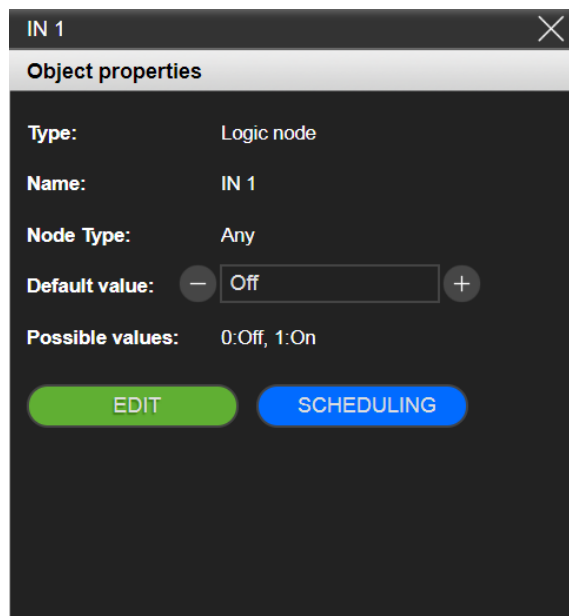
–while the output nodes are the outputs:



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.

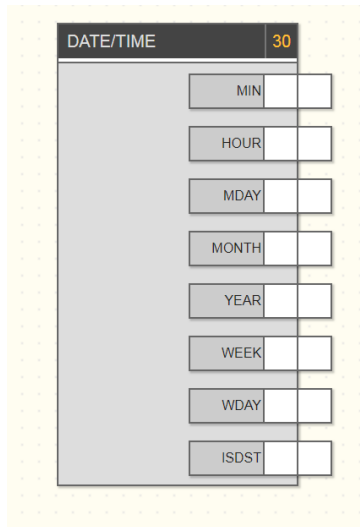
Not all input nodes are absolutely necessary for the correct execution of the logic; if an input node is not connected, the default value is used, which can be changed by selecting the node and opening the corresponding detail area, as shown in the following figure:





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.

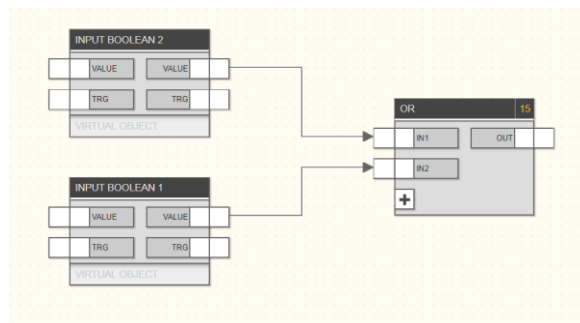
The logic blocks can also only provide outputs, as in the following example (date/time block):



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

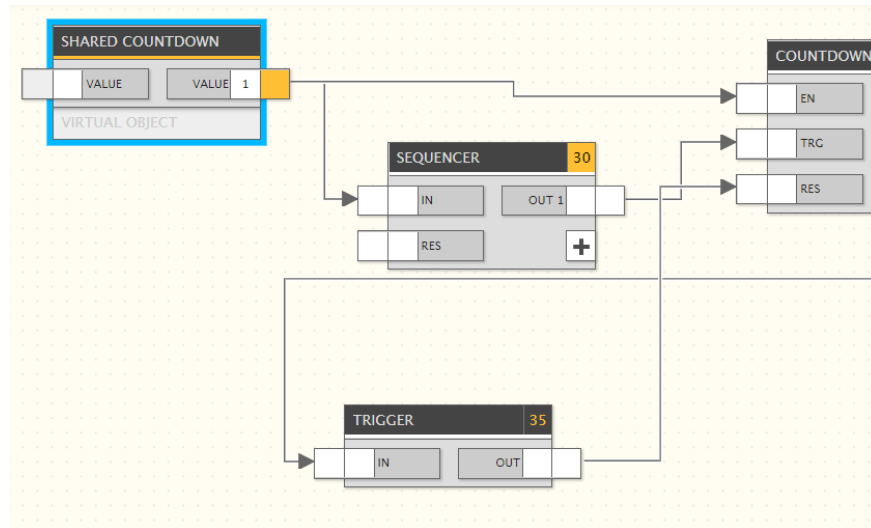
### 3.9 CONNECTION OF BLOCKS

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 center of the source node, hold down the mouse button and release it in the center 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 on the bus

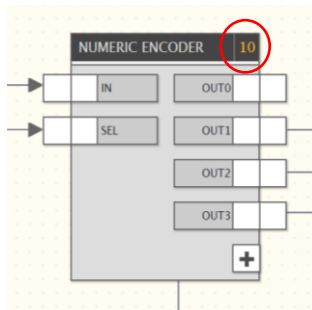
This means that any interactions between tasks (such as passing values by variables or writing the same node of a block by multiple tasks) are affected by this order, and any actions performed by the tasks towards the end of the queue are not performed by the previous ones until the next execution cycle.



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

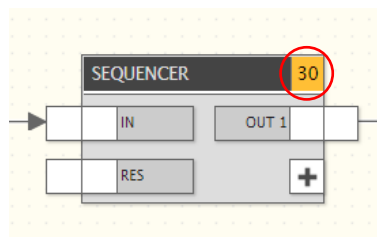
Within each task, the logical blocks also have their own order of execution: the logic module processes the function assigned to the logic blocks in that order. The order of a logic block is highlighted at the top right, as shown in the following figure:



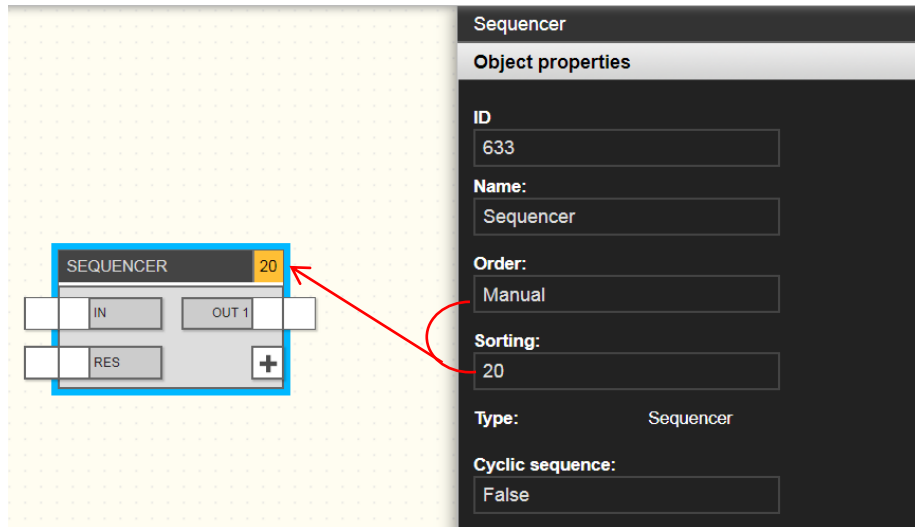
Under normal conditions, the blocks are assigned an ascending order corresponding to the order in which they were inserted into the program; however, it is possible to force a different execution order as follows:

- 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 marked 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 programs 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 program

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 programs, 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 and do not accept omissions to create the connection.

---

### 3.13 EXECUTION OF LOGIC ON BUS

---

To be removed?

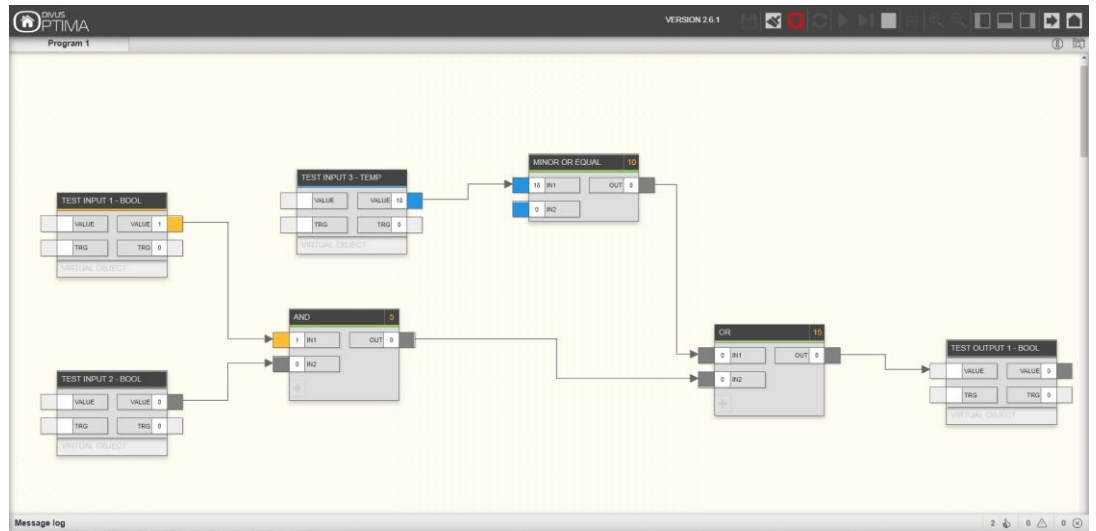
---

### 3.14 SIMULATION

---

Before implementing the programs, 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

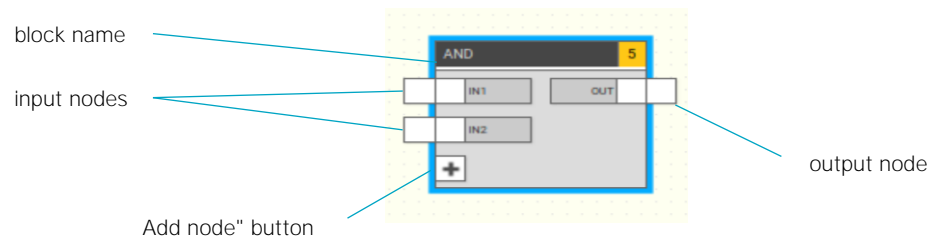
## 4.1 INTRODUCTION

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.

There are no configurable options for output nodes of a logical block.

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 deleted.

---

4.3        COMBINATORIAL LOGIC

---

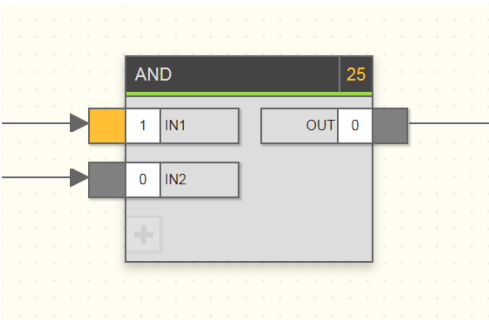
4.3.1    AND

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

---



PREVIEW



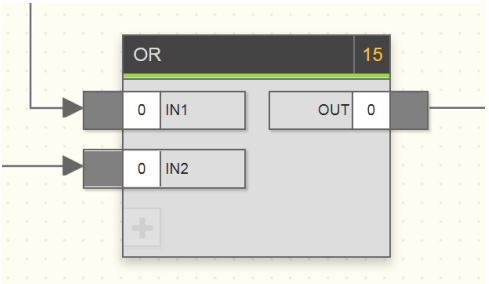
NODES	day	Description of the	entrance	exit
	ON1...ON10	Input 1...10Possible values: 0 → OFF 1 → ON	X	
	OFF	OutputPossible values: 0 → OFF 1 → ON		X
	+	Add nodes	X	

4.3.2 OR

DESCRIPTION

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

PREVIEW



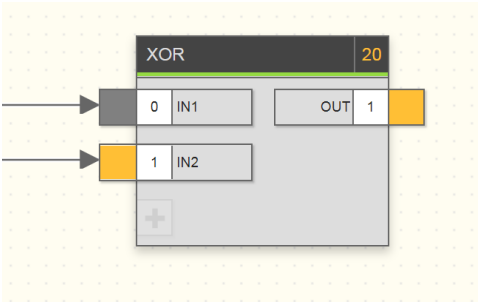
NODES	day	Description of the	entrance	exit
	ON1...ON10	Input 1...10Possible values: 0 → OFF 1 → ON	X	
	OFF	Output Possible values:		X

	0 → OFF 1 → ON		
+	Add nodes	X	

4.3.3 XOR

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

**PREVIEW**

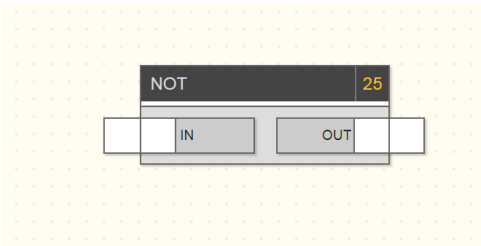


NODES	day	Description of the	entrance	exit
	ON1...ON10	Input 1...10Possible values: 0 → OFF 1 → ON	X	
	OFF	OutputPossible values: 0 → OFF 1 → ON		X
	+	Add nodes	X	

4.3.4 NOT

**DESCRIPTION** Executes the logical NOT function of the input.

PREVIEW



NODES	day	Description of the	entrance	exit
	ON1...ON10	Input 1...10Possible values: 0 → OFF 1 → ON	X	
	OFF	OutputPossible values: 0 → OFF 1 → ON		X
	+	Add nodes	X	

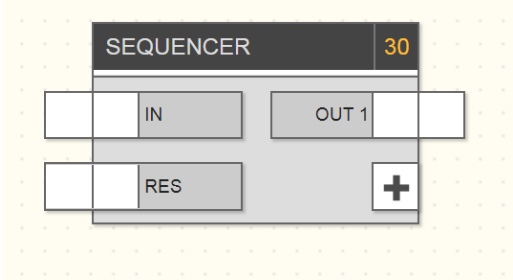
4.4 SCENARIOS AND SEQUENCES

4.4.1 SEQUENCER

DESCRIPTION

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

PREVIEW



NODES	day	Description of the	entrance	exit
	ON	Sequence startPossible	X	

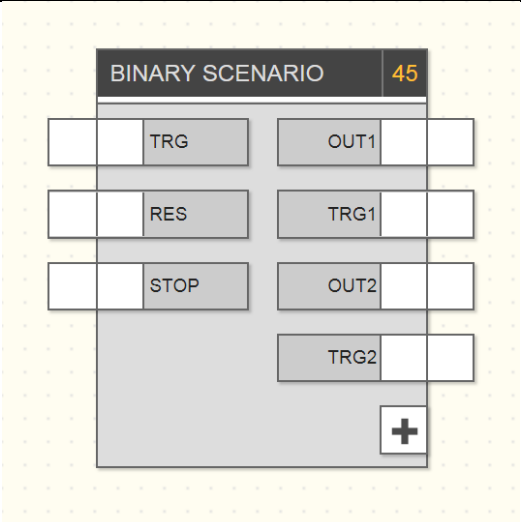
	values: 0 Off1→ On→		
RES	Reset the Seq. Possible values: 0 Off1→ On→	X	
OFF1...OFF10	Output 1...10 Possible values: 0 Off1→ On→		X
+	Add nodes		X
Options for the	Cyclic sequence	Defines whether the sequence is to be repeated. Possible values TRUE/FALSE	
	Duration of the step 1...10	Waiting time between 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, with each command possibly changing with a fixed time common to all outputs.

PREVIEW



## NODES

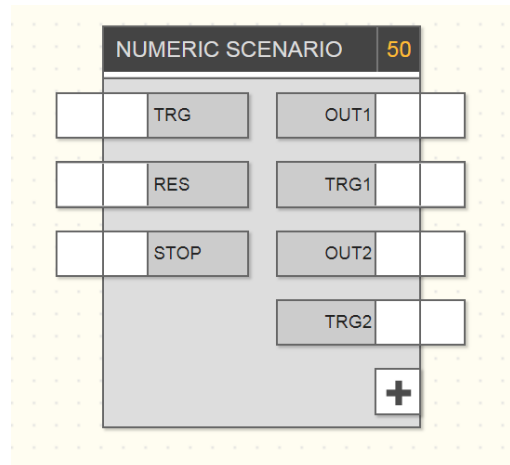
day	Description of the	entrance	exit
TRG	Trigger input Possible values: 0 Off1 → On→	X	
RES	Reset the scene. Possible values: 0 Off1 → On→	X	
OFF1...OFF10	Output 1...10 Possible values: 0 Off1 → On→		X
TRG1...TRG10	Trigger 1...10 Possible values: 0 Off1 → On→		X
+	Add node (and trigger)		X
Options for the	output interval	Waiting time between the output commandsPossible values1 ...60 sec.	
	Set output 1...10	Value at the outputs 1...10Possible 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, with each command possibly alternating with a preset time common to all outputs.

## PREVIEW



NODES	day	Description of the	entrance	exit
	TRG	Trigger input Possible values: 0 Off1 → On→	X	
	RES	Reset the scene. Possible values: 0 Off1 → On→	X	
	STOP	Stop the scene. Possible values: 0 Off1 → On→	X	
	OFF1...OFF10	Output 1...10 Possible values: any numeric value		X
	TRG1...TRG10	Trigger 1...10 Possible values: 0 Off1 → On→		X
	+	Add Node & Trigger		X
	Options for the	output interval	Waiting time between the output commands Possible values 1 ...60 sec.	

	Set 1...10	output	Value at the outputs 1...10Possible values: any numeric value
--	---------------	--------	--

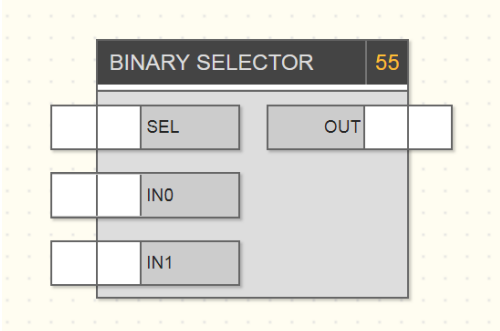
4.5 GATE

4.5.1 BINARY SELECTOR SWITCH

DESCRIPTION

Returns the value of one of the inputs based on the value of the SEL input as selector:  
If SEL = Wrong → OFF = ON0  
  
If SEL = True → OFF = ON1

PREVIEW



day	Description of the	entrance	exit
SEL	Input selectionPossible values: 0 →Off i.e. OFF = ON01→ On i.e. OFF = ON1	X	
ON0EIN1	Inputs 0 and 1 Possible values: 0 Off1 → On→	X	
OFF	Output Possible values: 0 Off1 → On→		X

4.5.2 NUMERICAL SELECTOR SWITCH

DESCRIPTION

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

If SEL = Wrong → OFF = ON0

If SEL = True → OFF = ON1

PREVIEW				
NODES	day	Description of the	entrance	exit
	SEL	Input selectionPossible values: 0 → Off i.e. OFF = ON01 → On i.e. OFF = ON1	X	
	ON0EIN1	Inputs 0 and 1Possible values: Any numeric value	X	
	OFF	OutputPossible values: Any numeric value		X

4.5.3 BINARY ENCODER

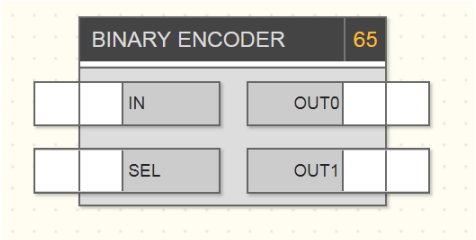
DESCRIPTION

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

Number of outputs: from 2 to 10



PREVIEW



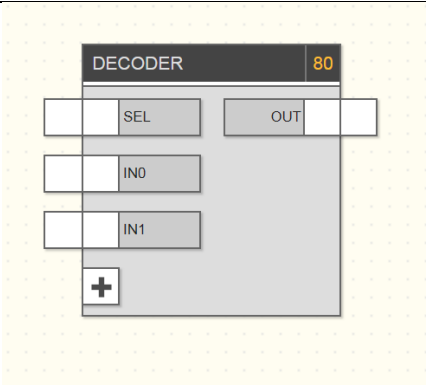
NODES	day	Description of the	entrance	exit
	ON	Input selectionPossible values: 0 Off1 → On→	X	
	SEL	Output selectionPossible values: 1...10	X	
	OFF0...OFF9	Output 0...9 Possible values: 0 Off1 → On→		X
	+	Add output		X

4.5.4 DECODER

DESCRIPTION

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

PREVIEW



NODES	day	Description of the	entrance	exit
	ON	Input selectionPossible	X	

	values: 0...9		
ON0...ON9	Input Possible values: 0 Off1 → On→	X	
OFF	Output Possible values: 0 Off1 → On→		X
+	Add Input	X	

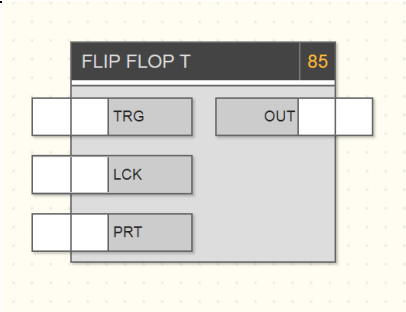
4.5.5 T FLIP-FLOP

DESCRIPTION

Works like a step-by-step relay. Each time a rising edge appears at its input (TRG), the output (OFF) changes state. When the LCK (Lockout) input is 1 (True), the TRG effect is locked, so the output never changes. When the PRT input (priority) is set to 1, the output assumes the value set in 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	day	Description of the	entrance	exit
	TRG	Trigger Possible values: 0 Off1 → On→	X	

LCK	LockPossible values: 0 Off1 → On→	X	
PRT	Priority flagPossible values: 0 Off1 → On→	X	
OFF	Output Possible values: 0 Off1 → On→		X
Options for the	priority value	Value of the output if priority flag is set. Possible values: True/False	

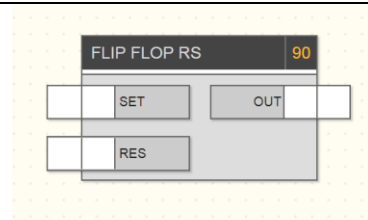
## 4.5.6 RS FLIP-FLOP

## DESCRIPTION

Elementary memory block that is "loaded" with the SET input and reset with the RES (Reset) input. If both inputs are 1, the priority set by the "Priority" parameter has priority.

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	day	Description of the	entrance	exit
	SET	SET inputPossible values: 0 Off1 → On→	X	
	RES	RESET inputPossible values: 0 Off1 → On→	X	

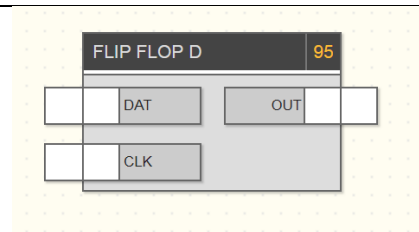
OFF	Output Possible values: 0 Off1 → On→		X
Options for the	Select priority	Possible values: 0 Off1 → On→	

## 4.5.7 D FLIP-FLOP

**DESCRIPTION**

The operation is similar to that of the D-Lock with the difference that the D-Flipflop acts on the variation of the front of CLK.

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

**PREVIEW****NODES**

day	Description of the	entrance	exit
DAT	DataPossible values: 0 Off1 → On→	X	
CLK	ClockPossible values: 0 Off1 → On→	X	
OFF	Output Possible values: 0 Off1 → On→		X
Options for the	Select priority	Possible values: DAT / CLK	

## 4.5.8 BINARY D-LOCKING

**DESCRIPTION**

In this block the input signal ON is forwarded to the output OFF when the enable signal ENA is activated (1). When the

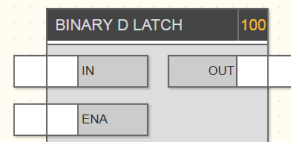
ENA signal is deactivated, the last state remains at the output (OFF).

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

If ENA = 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 ON and OFF is binary.

#### PREVIEW



#### NODES

day	Description of the	entrance	exit
ON	InputPossible values: 0 Off1 → On→	X	
ENA	EnablePossible values: 0 Off1 → On→	X	
OFF	Output Possible values: 0 Off1 → On→		X
Options for the	Select priority	Possible values: ON / ENA	

#### 4.5.9 NUMERICAL D-LOCK

#### DESCRIPTION

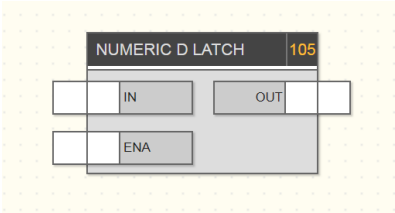
In this block the input signal ON is forwarded to the output OFF when the enable signal ENA is activated (1). When the ENA signal is deactivated, the last state remains at the output (OFF).

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

If ENA = 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 ON and OFF is numeric.

PREVIEW



NODES	day	Description of the	entrance	exit
	ON	InputPossible values: 0 Off1 → On→	X	
	ENA	EnablePossible values: 0 Off1 → On→	X	
	OFF	Output Possible values: 0 Off1 → On→		X
	Options for the	Select priority	Possible values: ON / ENA	

4.6 COMPARISONS

RELATIONAL OPERATORS

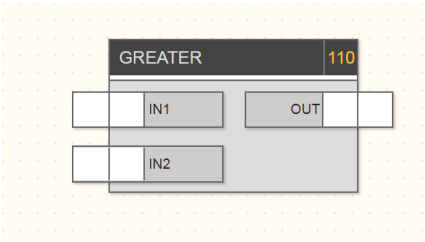
DESCRIPTION

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

Available operators:

- Bigger
- Greater than or equal to
- kid
- Less than or equal to
- Equal
- Not equal

PREVIEW



NODES	day	Description of the	entrance	exit
	ON1 ON2	Input 1, 2Possible values: Any numeric values	X	
	OFF	Comparison result0 Off1→ On→		X

4.7 OPERATIONS

4.7.1 MATHEMATICAL OPERATORS

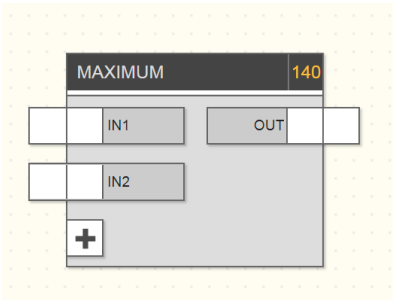
DESCRIPTION

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

Available operators:

- maximum
- minimum
- mean
- sum
- subtraction
- multiplication
- division
- range
- Absolute value
- Log10
- integration

PREVIEW



NODES	day	Description of the	entrance	exit
	ON1 ON2 *	Input 1, 2Possible values: Any numeric values	X	
	OFF	ResultPossible 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

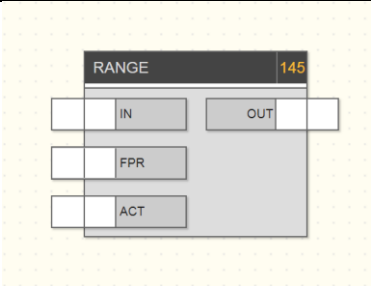
DESCRIPTION

Performs a linear interpolation of the input value ON based on an associated mapping, also called "characteristic" or "feature", defined by two pairs of values (X, Y). The ON 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 preset value is returned.

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

PREVIEW



NODES	day	Description of the	entrance	exit
-------	-----	--------------------	----------	------



ON1	InputPossible values: Any numeric value	X	
FPR	Priority activatedPossible values: 0 no1→ Priority value→ is output	X	
ACT	Direct/inverted function Possible values: 0 direct1→ inverted→	X	
OFF	OutputPossible values: Any numeric value		X
OPTIONS	X0Y0X1Y1	Characteristics of linear interpolation Possible values: Any numeric value	
		Value to be output if priority is set	

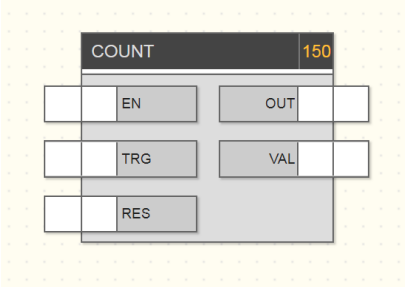
4.8 COUNTERS

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.  
Meter types: Counter, countdown, up/down counter

PREVIEW



## NODES

day	Description of the	entrance	exit
FINAL INSPECTION	ActivationPossible values: 0 →not activated1 activated→	X	
TRG	TriggerPossible values: 0 Off1→→ On (increments counter)	X	
RES	ResetPossible values: 0 Off1→→ On (counter reset)	X	
OFF	Output Possible values: 0 Off1→ On→		X
VAL	Current valuePossible values: Any numeric value		X
Options for the	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 &amp; SCHEDULES

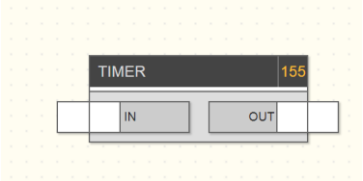
## 4.9.1 TIMER

## DESCRIPTION

Delays the received input value by a preset time.

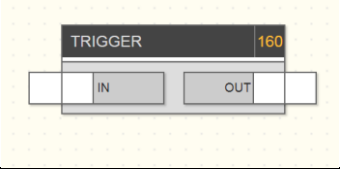
If a 1 is received at the input (ON) (rising edge), an internal counter starts until the time specified as "rising delay", then the output is raised to 1; conversely, if a 0 is received at the

input (falling edge), the block waits for the time specified as "falling delay" before setting the output to 0.

PREVIEW				
NODES	day	Description of the	entrance	exit
	ON	InputPossible values: 0 Off1 → On→	X	
	OFF	Output, delayed by timer Possible values: 0 Off1 → On→		X
	Options for the	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	<p>Generates a trigger at an edge detected at the input (pulse of one cycle duration).</p> <p>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.</p>
-------------	--

PREVIEW	
---------	--

## NODES

day	Description of the	entrance	exit
ON	entrance	X	
OFF	impulse		X
Options for the	flank	Rising or falling edge to be detected at the input.	

## 4.10 VARIABLES

## 4.10.1 FOREWORD

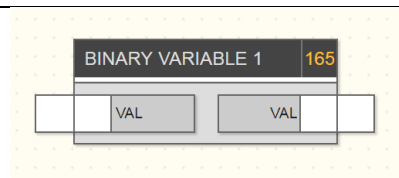
As shown in Section 3.11 variables can pass values between different tasks. The variables must first be created using the "+" button in the corresponding section of the main menu so that they can be dragged into the tasks they are to use.

## 4.10.2 BINARY VARIABLES

## DESCRIPTION

Allow to transfer a Boolean value between different tasks.

## PREVIEW



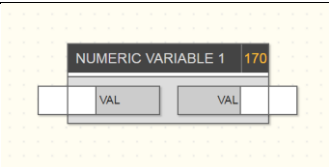
## NODES

day	Description of the	entrance	exit
VAL	InputPossible values: 0 Off1 → On→	X	
VAL	Current valuePossible values: 0 Off1 → On→		X

4.10.3 NUMERIC VARIABLES

DESCRIPTION

Allow to transfer a numeric value between different tasks.

PREVIEW				
NODES	day	Description of the	entrance	exit
	VAL	Value the variable receivesPossible values: any numeric value	X	
	VAL	Current value of the variablePossible 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 simulation

---

## 5.1 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, even through the logic blocks that contain a variation of the outputs over time.

---

## 5.2 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.

---

## 5.3 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 logical programs is blocked.
- The nodes take on a colour depending on their state and allow you to force the value manually (as described below).

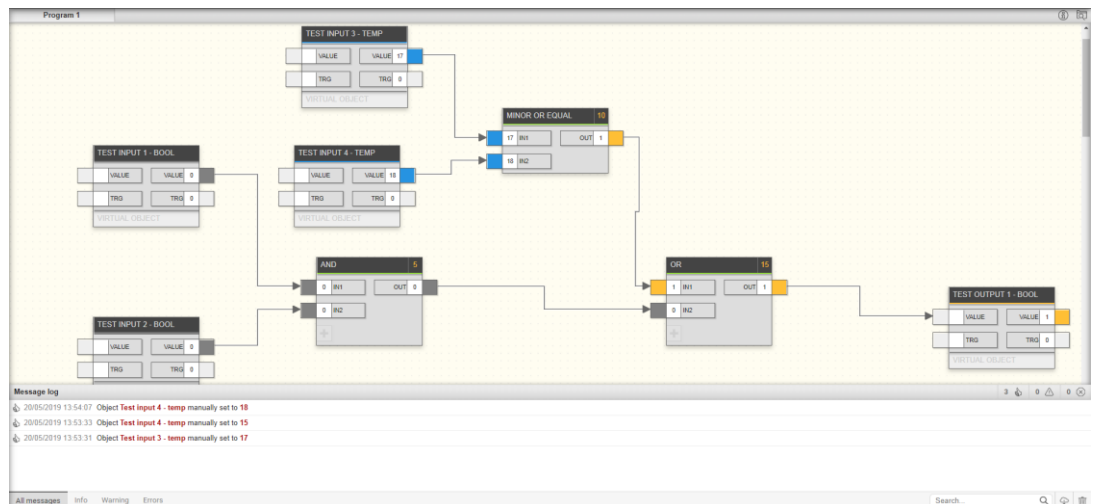
The colour of the nodes follows the following convention:

binary node	grey	Value 0 (OFF)
	yellow	Value 1 (ON)
<hr/>		
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 Section 2.6 more details on the notification area.

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



## 5.4 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).

---

## 5.5 STOP SIMULATION

---

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



# 6 drawing tools

---

## 6.1 INTRODUCTORY REMARKS

---

To increase the readability of logical programs, 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 logical programs like the other types of objects.

---

## 6.2 INSCRIPTIONS

---

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.

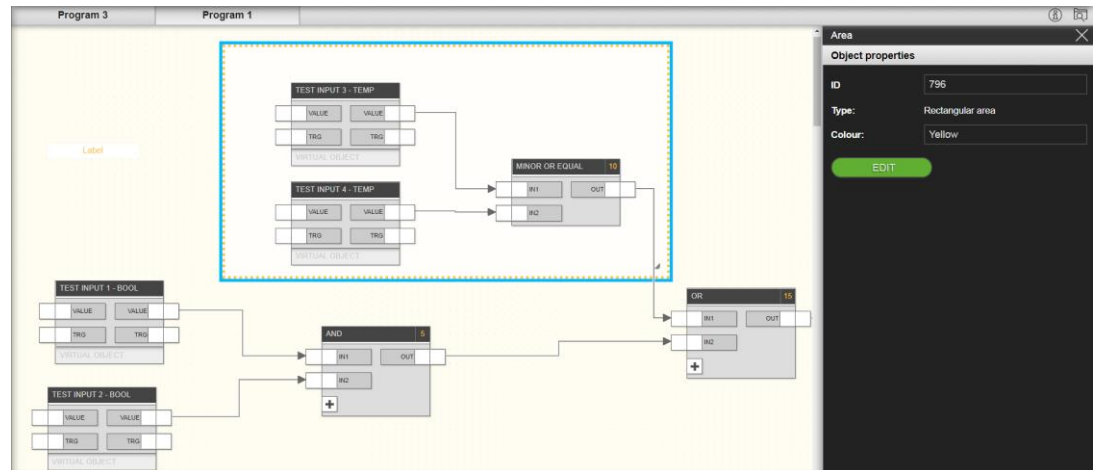
Once a label has been dragged into a Task and positioned at the desired point, it is possible to customize it by opening the details window (after it has been selected); the following options are available:

- **text** Text that is displayed in the logical task.
- **colour** Allows you to select the color of text (black, yellow, red, green, blue, gray)
- **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.

### 6.3 RECTANGULAR AREAS

It is possible to highlight one or more parts of the logic program 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.

# 7 appendix

## 7.1 GLOSSARY

<b>Logical module</b>	The catalog nomenclature is "Logical Module".
<b>Logical Program</b>	A logical network consisting of one or more logical tasks.
<b>Logical Task</b>	A logical network consisting of one or more linked blocks and logics. Each logical unit can contain up to 16 logical tasks.
<b>Logical block</b>	Block that can be inserted into a logic program to perform a particular function and interact with other blocks via input and/or output nodes.
<b>Block</b>	Device which can be inserted into a logic program to read and/or write information on the home automation bus and to interact with other blocks via input and/or output nodes
<b>knot</b>	Single element of a logic block that provides certain information in the input or output; the nodes can be connected to other <i>nodes</i> via corresponding connections.
<b>liaison</b>	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.
<b>editor</b>	Graphical configuration environment for logical programs. 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.

## 7.2 NEED

[illegible]