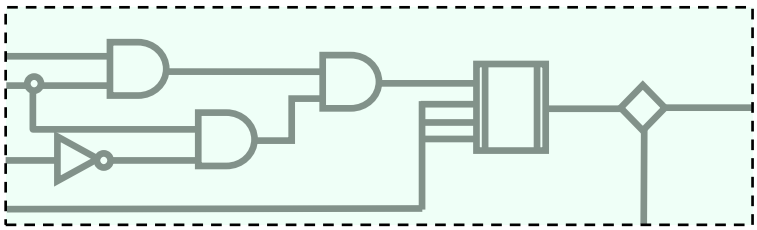
Zennio®



Logic Functions

Logic-Mathematical Functions Module

USER MANUAL

User Manual Version: [1.0]_a

www.zennio.com

CONTENTS

DOCUMENT UPDATES

Version	Changes	Page(s)
[1.0]_a	**Software changes**: • Internal optimisation.	-
[0.3]_a	**Software changes**: • Internal delay between simultaneous sendings. • Periodic and delayed sendings of previous results are cancelled upon re-triggering of the function.	-

1 INTRODUCTION

1.1 LOGIC FUNCTIONS MODULE

A variety of Zennio devices (among many others, all the actuators of the ACTinBOX and MAXinBOX series) incorporate a Logic Functions module, which makes them capable of performing **mathematical** and **binary logic** operations with data received from the KNX bus, as well as of sending the results through specific communication objects of different sizes.

The operands of these functions may be of the following types:

- **Communication objects** received through the KNX bus.
- **Internal variables** containing partial results from previous operations.
- **Constant values**, defined by parameter in ETS.

Up to ten different and independent functions can be configured, each of which can consist itself in up to four successive operations, all of which can share a set of internal variables so that the result of one operation can be used as the input for the next one.

Important: *depending on the particular device, the number of functions available and the functionality itself may differ slightly. Therefore, the user manual for the Logic Functions module has been particularised for each device. It is highly encouraged to make use of the links provided at the Zennio homepage (www.zennio.com) within the download section of the particular device being configured.*

2 CONFIGURATION

2.1 GENERAL APPROACH

The Logic Functions module permits enabling and configuring up to ten independent functions, the behaviour of which is typically divided into four stages:

- **Triggers**: the first step to trigger the execution of a function consists in *calling* it. With that aim, one or more communication objects can be configured, so that whenever any of them updates its value from the KNX bus, the function will be triggered (provided that the execution condition is found to be true).

- **Execution Condition**: after triggering the function, it will be checked whether the execution condition is true or not. This condition involves the comparison of two input operands. If the comparison is true, the operations will be executed. If no execution condition is configured, the function will be executed whenever it is triggered.

- **Operations**: triggering the function will itself initiate the execution of up to four mathematical or binary operations. The following needs to be configured for each of them:

  - **Operation type**: desired action (addition, subtraction, negation, etc.).

  - **Operands**: values to operate on. They can be input communication objects, internal variables containing the result of previous operations, or constants predefined in ETS.

  - **Result**: internal variable where to store the result of the operation.

- **Result**: it is necessary to set which internal variable contains the global result of the function, so its value will be sent through the corresponding communication object once the execution of all the operations ends.

Figure 1 shows a diagram of the operation of logical functions.

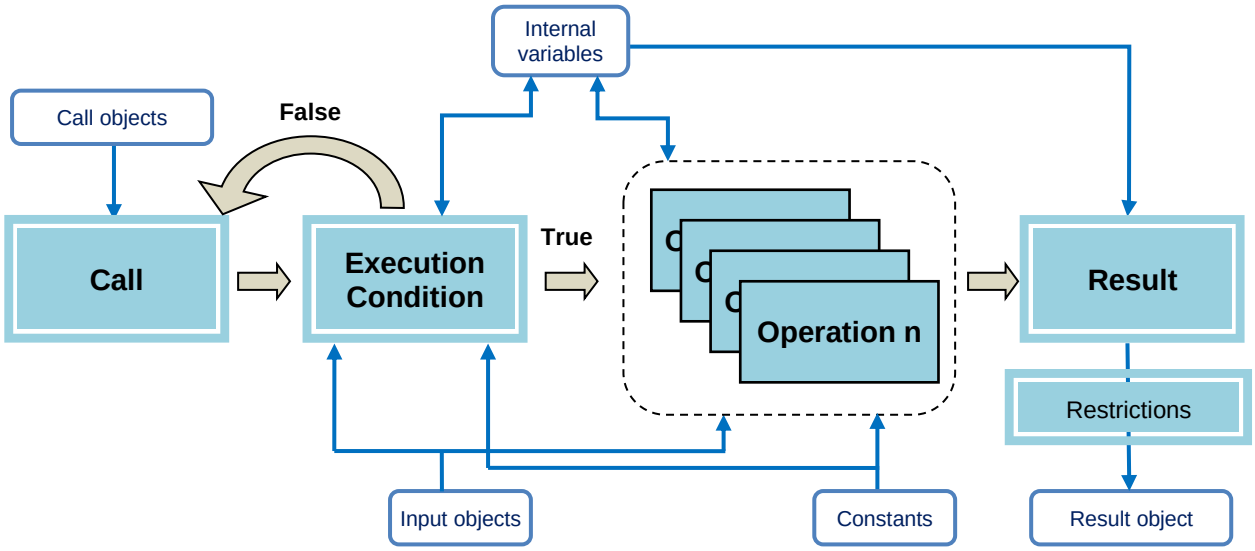


Figure 1. Logic Functions operation diagram.

2.2 TRIGGERS

Up to eight trigger objects (with a size of one bit, one byte, two bytes or four bytes) are available for each function, each of which will trigger the function as soon as it receives a value from the bus. These objects do not necessarily need to be used as operators as well.

Note that the actual execution of the operations after triggering the function will depend on the evaluation of the execution condition, if any.

2.3 EXECUTION CONDITION

Once the trigger is activated, the next step is to check whether the current situation meets the conditions for executing the operations.

If no execution condition has been configured, the operations will be executed always. If configured, the operations will be executed only if the comparison of **two input operands** (constant values, communication objects or internal variables) is true.

The **comparison operations** available are (depending on the operation size): equal to; not equal to; greater than; greater than or equal to; less than; and less than or equal to.

2.4 OPERATIONS

As already stated, each logic function consists in the execution of up to four consecutive operations, which can be any of the following:

- **Logic**: ID, NOT, AND, OR, XOR, NAND, NOR and NXOR.
- **Arithmetic**: ID, addition, subtraction, multiplication, division, maximum and minimum.
- **Comparison**: is equal to, not equal to, is greater than, is greater than or equal to, is less than, is less than or equal to..
- **Conversion**: *cast* operations to convert a certain operand from one size to another (e.g., to convert a 1-bit value into a 1-byte value).

The Logic Functions module can operate in the following value ranges (for either communication objects, internal variables with intermediate results, or numerical constants defined by parameter in ETS):

- Binary values: **0** and **1**.
- Unsigned integers (one byte): **0 – 255**.
- Percentage values (one byte): **0 – 100**.
- Unsigned integers (two bytes): **0 – 65535**.
- Signed integers (two bytes): **-32768 – 32767**.
- Floating point values (two bytes): **-671088.64 – 671760.96**.
- Signed integers (four bytes): **-2147483648 – 2147483647**.

For further information in relation to these operations, size conversions and how values are truncated, please consult *ANNEX I: Operations Reference*.

2.5 INPUT OBJECTS

Multiple specific objects can be enabled to use them with the Logic Functions module:

- Up to 32 one-bit objects,
- Up to 16 one-byte objects,
- Up to 16 two-byte objects.
- Up to 8 four-byte objects.

The value of the above objects can work, for example, as operands for the operations of the enabled functions.

2.6 INTERNAL VARIABLES

Additionally, the integrator will have the following at their disposal:

- 32 one-bit internal variables (b1, …, b32),
- 16 one-byte internal variables (n1, …, n16),
- 16 two-byte internal variables (x1, …, x16).
- 8 four-byte internal variables (y1, …, y8).

All of them may be used to temporarily store intermediate results, which themselves will be available as input values for later operations.

2.7 RESULT OBJECTS

Every logical function comes with a specific object (one-bit, one-byte, two-bytes or four-bytes size, depending on the parameterisation of the function) through which the final value of a certain internal variable (selectable by parameter) will be sent to the bus, as the result of the sequence of operations that make up the function.

The integrator has the chance to set whether this sending should happen every time the function is executed, or periodically, or only in case the function throws a result that differs from that of the previous execution.

On the other hand, the results to be sent can be restricted, so the KNX bus is only notified when the result meets a certain restriction or range of values. Finally, it is also possible to configure in ETS a certain delay for the transmission of the result.

3 ETS PARAMETERISATION

3.1 CONFIGURATION

The 'Configuration' screen of the Logic Functions module contains the options shown below.

Note: *the parameter tabs of the Logic Functions module may not show in ETS by default – it may be necessary to enable this module from the General parameter tab of the device itself. Please refer to the user manual of the device for more details*.

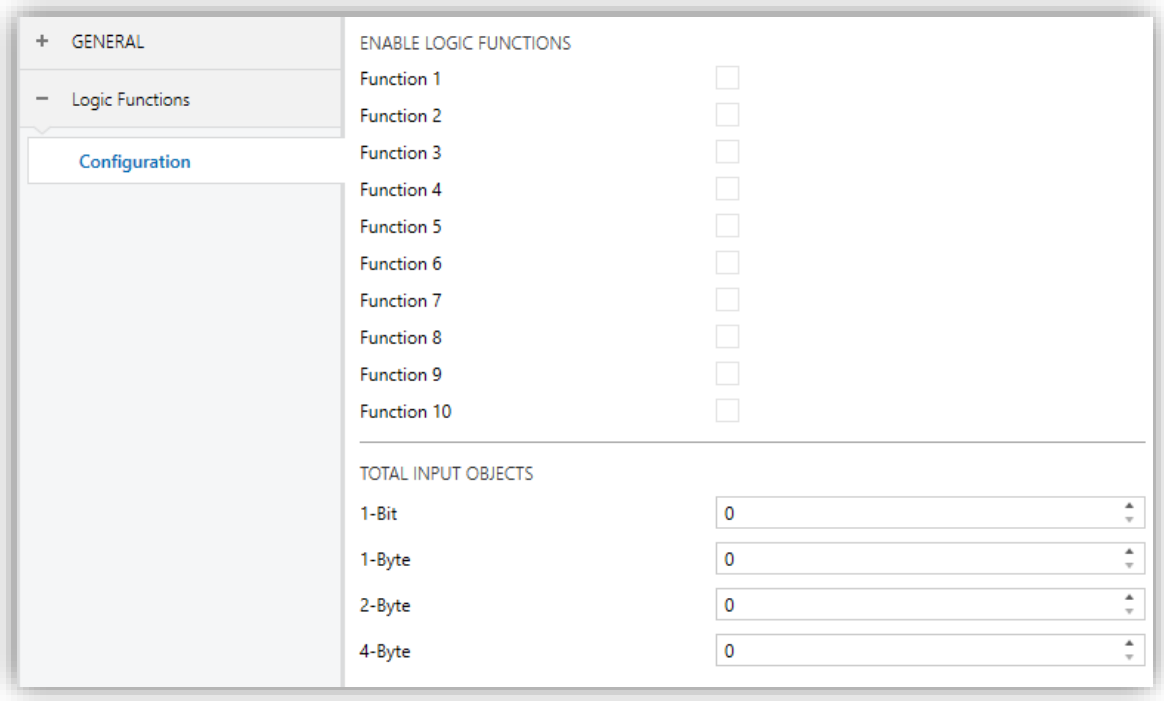


Figure 2. General Parameter Tab of the Logical Functions Module.

As illustrated, none of the ten functions are enabled by default. As they become enabled by the integrator, additional tabs will be included into the tab list on the left.

Moreover, in this tab the integrator sets the number of total input objects needed for all the logic functions. As explained in section 2.5, it is possible to enable up to 32 one-bit objects, 16 one-byte objects, 16 two-byte objects or 8 four-byte objects.

The next sections cover the purpose of every tab and of the parameters they contain.

3.2 FUNCTION n

A specific tab per enabled function (see section 3.1) will be included into the tab list on the left, being itself divided into four more tabs.

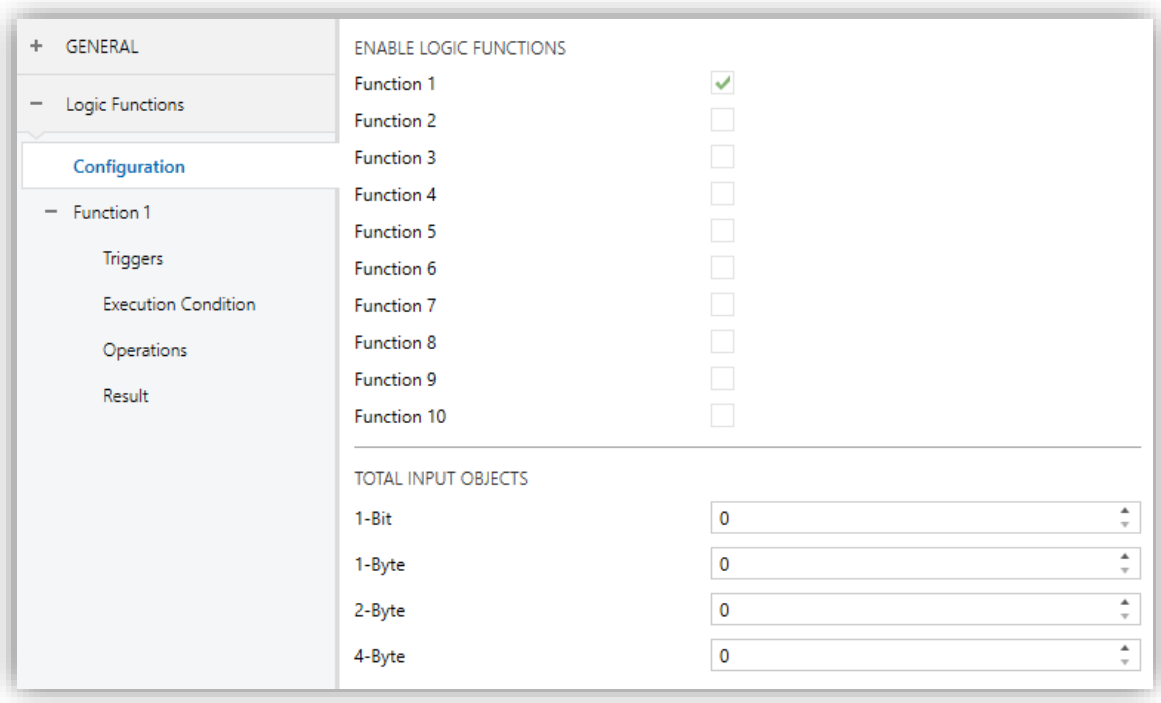


Figure 3. Function 1 enabled.

3.2.1 TRIGGERS

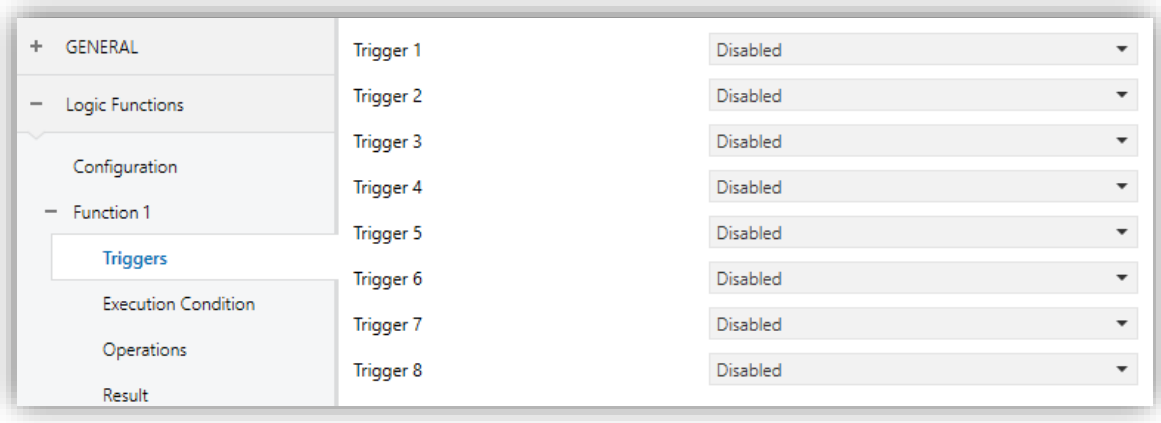


Figure 4. "Triggers" Tab

This section allows selecting up to **eight objects** to work as call objects. Call objects trigger the execution of a function whenever any of them receives a value from the bus, as long as the execution condition is true. Of course, setting one particular object as

the call object of multiple functions will make them trigger consecutively whenever the object is written a value.

Note: *objects being used as triggers must have been enabled previously (see 3.1)*.

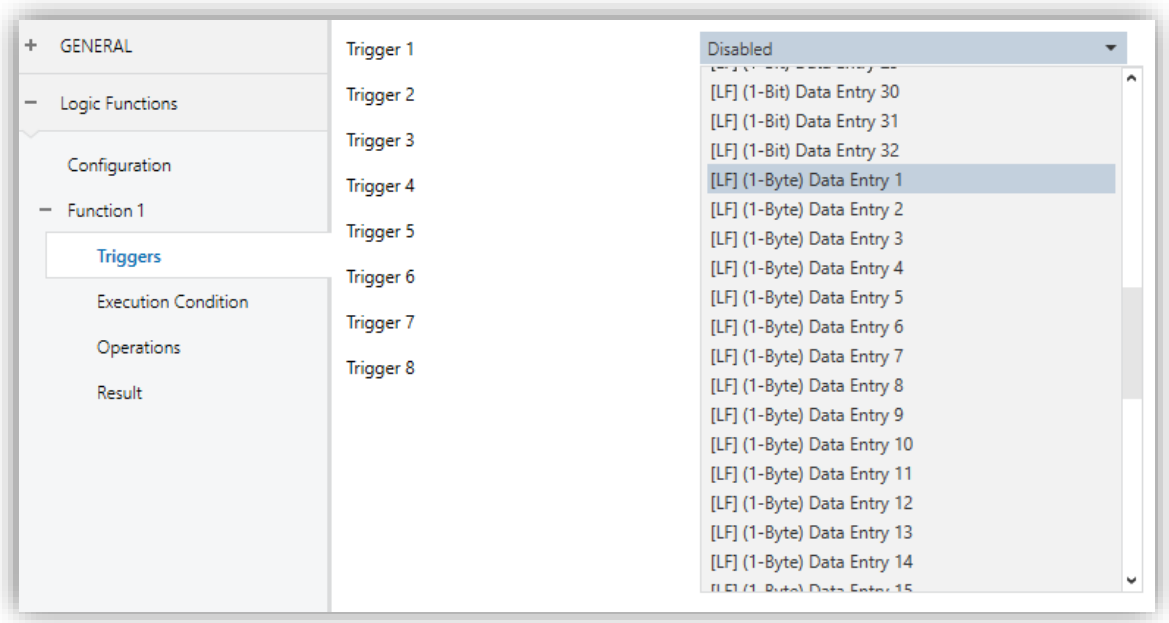


Figure 5. Selecting the trigger objects

3.2.2 EXECUTION CONDITION

From this section it is possible to configure the execution condition of the function through the following parameters:

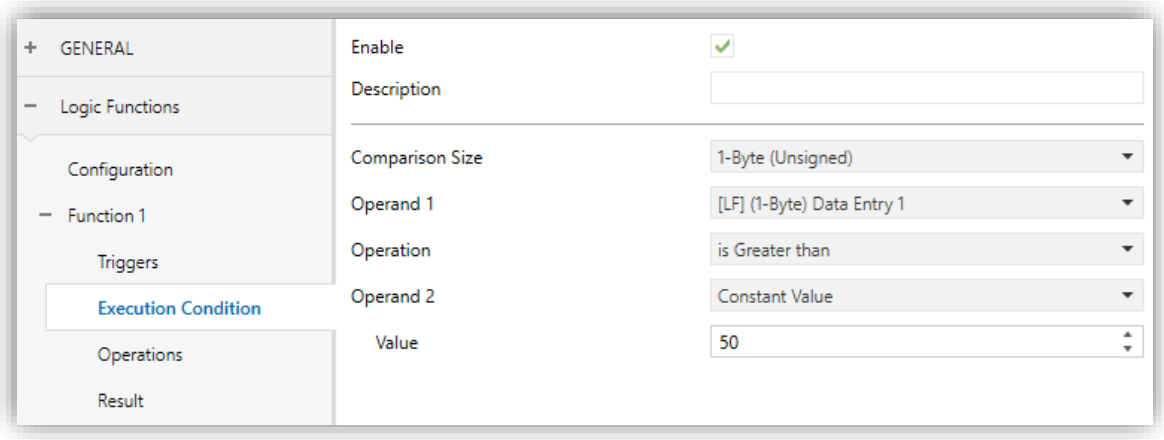


Figure 6. "Execution Condition" Tab.

- **Enable**: sets whether configuring an execution condition is required or not.
- **Description**: sets a brief description (up to 100 characters). This field simply helps identify the function, and does not have a practical implication – it is provided for the convenience of the integrator.
- **Comparison Size**: sets the size of the involved input operands: 1 bit, 1 byte (unsigned), 1 byte (percentage), 2 bytes (unsigned), 2 bytes (signed), 2 bytes (float), 4 bytes (signed).
- **Operand 1**: source of the value: communication objects, internal variables or constant values. If "Constant Value" is selected, "**Value**" will be shown in order to set the desired constant value.
- **Operation**: it depends on the Comparison Size selected. If 1 bit selected, this field will only show the options "Is equal to" and "Is not equal to". If Comparison Size is set to other than "1 bit", this field will show the entire option list (see section 2.4).
- **Operand 2**: source of the value: communication objects, internal variables or constant values. If "Constant Value" is selected, "**Value**" will be shown in order to set the desired constant value.

3.2.3 OPERATIONS

The maximum number of operations per logic function is four. All of them are activated individually and executed sequentially. If any intermediate operation is left disabled, it will be ignored.

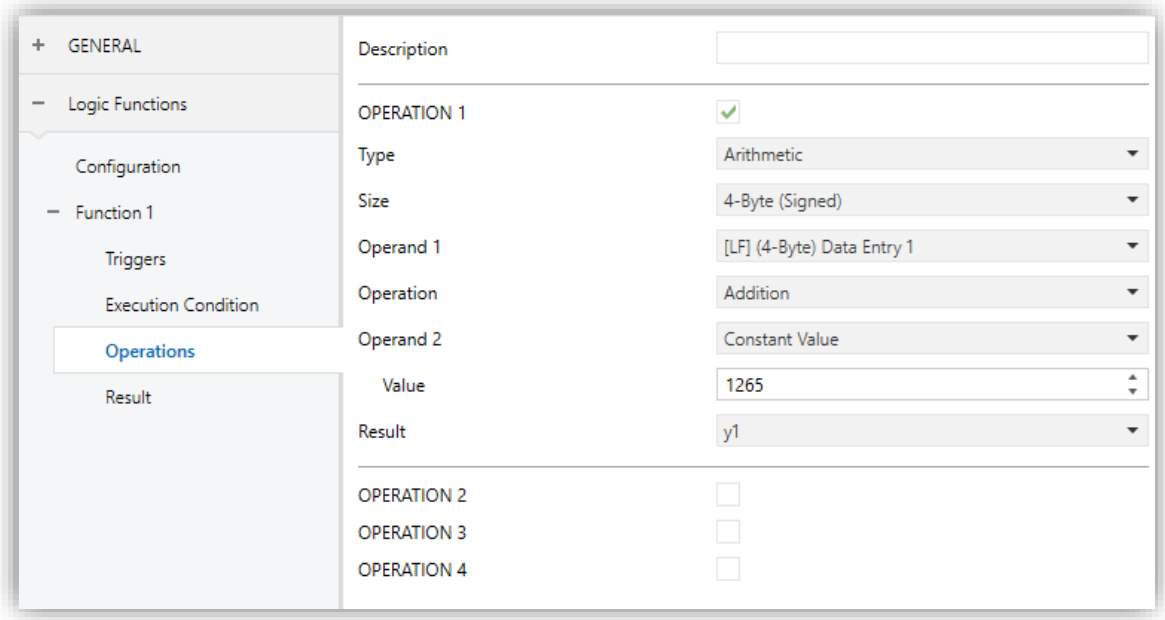


Figure 7. "Operations" Tab

For each operation it is possible to configure the following parameters:

- **Description**: sets a brief description (up to 100 characters) for the logic function. This field simply helps identify the function, and does not have a practical implication – it is provided for the convenience of the integrator.

- **Operation "i"**: enables or disables the operation number "i" (1-4). Every enabled operation offers the following parameters:

  - **Type**: sets the type of the operation (logic, arithmetic, comparison or conversion). If set to other than "logic", an additional parameter ("**Size**") will show up for the selection of the size of the operation result ("1 bit", "1-byte (unsigned)", "1-byte (percentage)", "2-byte (unsigned)", "2-byte (signed)", "2-byte (float)" and "4-bytes (signed)"). See section 2.4.

  - **Operation**: sets the particular action executed by operation number "i". Depending on the selected operation type (logical, arithmetical,

comparison or conversion), this parameter will display different options. For further information, please refer to *ANNEX I: Operations Reference*.

- **Operand "j"**: depending on the operation selected, one or more additional parameters named "Operand j" will show up, thus letting the user set the input values (operands) of the operation. These may be communication objects, internal variables or constant values (see sections 2.4 and later).

- **Operation Result**: sets the internal variable where the result of the operation will be stored. This partial result may be configured afterwards as the final function result or take part in later operations as an operand, if desired.

Note: *all the logic functions share together the same set of internal variables. This means that, for example, if function no. 1 stores a partial result into variable "n1" and afterwards function no. 2 reads that variable (to perform an operation on its value), it will find the value that was written there by function no. 1*.

3.2.4 RESULT

From this section it is possible to establish which internal variable should be considered as the one that contains the final result of the function, so that after executing all the operations that make up the function, the value of that variable can be reported to the bus through the "**[FL] Function *n* – Result**" object.

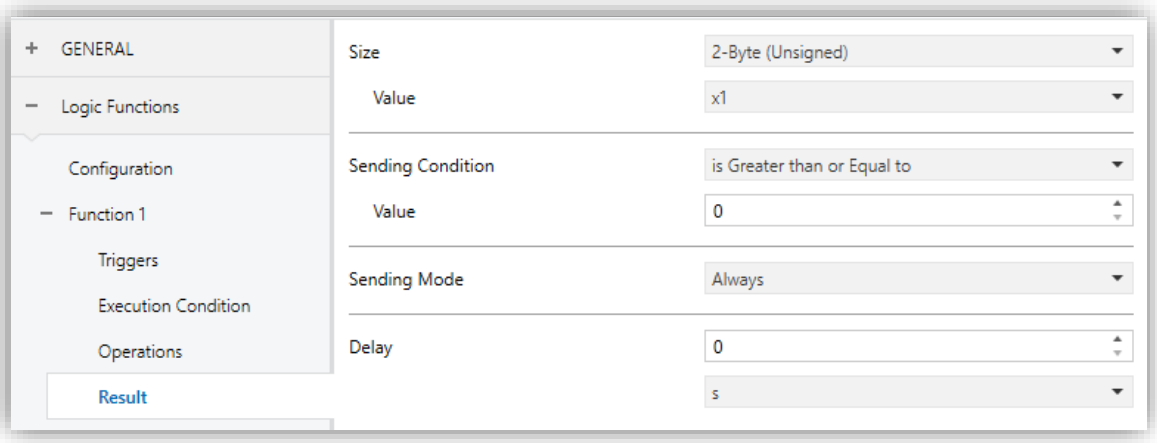


Figure 8. "Result" Tab.

- **Size**: sets the size of the function result. The available options are: "1 bit", "1 byte (unsigned)", "1 byte (percentage)", "2 bytes (unsigned)", "2 bytes (signed)", "2 bytes (float)", "4 bytes (signed)".

- **Value**: sets the particular internal variable whose value will be sent to the bus, through the function result object, after performing the operations.

- **Sending condition**: sets restrictions over the results to be sent to the KNX bus, so that only those that do meet the restriction will be sent. The available options are:

  - **For a one-bit result**:
    - No restriction,
    - Is equal to,
    - Is not equal to.
  - **For results of other sizes (1 byte, 2 bytes, 4 bytes)**:
    - No restriction,
    - Is equal to,

- Is not equal to,
- Is greater than,
- Is greater than or equal to,
- Is less than,
- Is less than or equal to.

After selecting a restriction, it will be also necessary to fill in the "**Value**" parameter. The range of values for this parameter depends on the size selected (see section 2.4).

- **Sending Mode**: this field defines under what circumstances the result of the function should be sent to the KNX bus, provided that it meets the above restriction.

  - Always.

  - Result is Different from Last One: the result of the function will only be sent to the bus when it differs from the value previously sent.

  - Periodic Sending: the result object will be re-sent to the bus every certain time, once the function is triggered for the first time. "**Periodicity**" sets the desired period. The range is 10 to 600 tenths of a second, 1 to 3600 seconds, 1 to 1440 minutes, or 1 to 24 hours.

    Bear in mind that re-sending a result periodically **does not mean that it is calculated again**, so any change in the input objects will not alter the value being sent – triggering the function again is necessary to update the result. Also note that triggering a function again will interrupt the previous periodic sending and reset the period time counter, even in case the new result **does not apply** for being sent to the bus – in such case, the previous periodic sending will be interrupted as well.

- **Delay**: sets a delay to be counted after the execution of the function and before sending the result to the bus. If the result is preferred to be sent immediately, this should be set to "0".

  The range of this parameter depends on the selected time unit: 0 to 600 tenths of a second, 0 to 3600 seconds, 0 to 1440 minutes, or 0 to 24 hours.

As explained, the delay is applied at the time of sending the object to the bus: the function is executed immediately after the call, regardless of whether the result will be sent delayed or not. Thus, the result sent to the bus will not be affected by changes in the operands during the delay time count.

Note: *in case of being a function triggered again with a previous result pending to be sent to the bus (due to the delay), the function will execute again and thus overwrite the previous result and restart the time count. in case the new result* ***does not meet the conditions*** *for being sent to the bus, the previous delayed sending will be cancelled as well*.

ANNEX I: OPERATIONS REFERENCE

OPERATIONS IN BOOLEAN LOGIC (1 BIT)

- **ID (identity)**

1st Operand	*2nd Operand*	*Result*
0	-	0
1	-	1

- **AND (logical conjunction)**

1st Operand	*2nd Operand*	*Result*
0	0	0
0	1	0
1	0	0
1	1	1

- **OR (logical disjunction)**

1st Operand	*2nd Operand*	*Result*
0	0	0
0	1	1
1	0	1
1	1	1

- **XOR (exclusive OR)**

1st Operand	*2nd Operand*	*Result*
0	0	0
0	1	1
1	0	1
1	1	0

- **NOT (negation)**

1st Operand	*2nd Operand*	*Result*
0	-	1
1	-	0

- **NAND (negated AND)**

1st Operand	*2nd Operand*	*Result*
0	0	1
0	1	1
1	0	1
1	1	0

- **NOR (negated OR)**

1st Operand	*2nd Operand*	*Result*
0	0	1
0	1	0
1	0	0
1	1	0

- **NXOR (negated XOR)**

1st Operand	*2nd Operand*	*Result*
0	0	1
0	1	0
1	0	0
1	1	1

ARITHMETIC OPERATIONS

- **ID (identity)**

1st Operand	*2nd Operand*	*Result*
v1	-	*v1*

- **ADDITION**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*v1 + v2*

- **SUBTRACTION**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*v1 - v2*

- **MULTIPLICATION**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*v1 * v2*

- **DIVISION**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*v1 / v2*

- **MAXIMUM**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*max(v1, v2)*

- **MINIMUM**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*min(v1, v2)*

Note: *it is advisable to read the Additional Remarks for further information about specific cases and overflows*.

COMPARISON OPERATIONS

- **HIGHER**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 > v2.* *0 in any other case.*

- **HIGHER OR EQUAL**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 ≥ v2.* *0 in any other case.*

- **LOWER**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 < v2.* *0 in any other case.*

- **LOWER OR EQUAL**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 ≤ v2.* *0 in any other case.*

- **EQUAL**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 = v2.* *0 in any other case.*

- **NOT EQUAL**

1st Operand	*2nd Operand*	*Result*
v1	*v2*	*1 ↔ v1 ≠ v2.* *0 in any other case.*

Note: *it is advisable to read the Additional Remarks for further information about specific cases and overflows*.

CONVERSION OPERATIONS

1-Bit Operand

	1 byte (unsigned)	*2 bytes (unsigned)*	*2 bytes (signed)*	*2 bytes (float)*	*4 bytes (signed)*
0	*0*	*0*	*0*	*0.00*	*0*
1	*1*	*1*	*1*	*1.00*	*1*

One-Byte Unsigned Operand

	1 bit	*2 bytes (unsigned)*	*2 bytes (signed)*	*2 bytes (float)*	*4 bytes (signed)*
0	*0*	*0*	*0*	*0.00*	*0*
10	*1*	*10*	*10*	*10.00*	*10*
180	*1*	*180*	*180*	*180.00*	*180*
255	*1*	*255*	*255*	*255.00*	*255*

One-Byte Percentage Operand

	1 bit	1 byte (unsigned)	2 bytes (unsigned)	2 bytes (signed)	2 bytes (float)	4 bytes (signed)
0	0	0	0	0	0.00	0
10	1	10	10	10	10.00	10
80	1	80	80	80	80.00	80
100	1	100	100	100	100.00	100

Two-Byte Unsigned Operand

	1 bit	1 byte (unsigned)	2 bytes (signed)	2 bytes (float)	4 bytes (signed)
0	0	0	0	0.00	0
10	1	10	10	10.00	10
500	1	255	500	500.00	500
65535	1	255	32767	65535.00	65535

Two-Byte Signed Operand

	1 bit	1 byte (unsigned)	2 bytes (unsig.)	2 bytes (float)	4 bytes (signed)
-32768	0	0	0	-32768.00	-32768
-12000	0	0	0	-12000.00	-12000
0	0	0	0	0.00	0
12345	1	255	12345	12345.00	12345

Two-Byte Floating-Point Operand

	1 bit	1 byte (unsigned)	2 bytes (unsig.)	2 bytes (signed)	4 bytes (signed)
-671088,64	0	0	0	-32768	-671088
-321654,98	0	0	0	-32768	-321654
0,5	0	0	0	0	0
0,00	0	0	0	0	0
12345,67	1	255	12345	12345	12345

Four-Byte Signed Operand

	1 bit	1 byte (unsigned)	2 bytes (unsig.)	2 bytes (signed)	2 bytes (float.)
-2147483648	0	0	0	-32768	0xF800 (*)
-1	0	0	0	-1	-1,00
0	0	0	0	0	0,00
123456	1	255	65535	32767	123456,00
2147483647	1	255	65535	32767	0x7FFF (*)

(*) The minimum and maximum 2-byte floating point values in the KNX standard will operate, respectively, as –infinity and +infinity, as explained in Additional Remarks.

ADDITIONAL REMARKS

As already stated, the Logic Functions module can operate on the following data types:

- Binary: **0** and **1**.
- Unsigned integers.
  - One byte: **0 – 255**.
  - Two bytes: **0 – 65535**
- Percentage values (one byte): **0 – 100**.
- Signed integers.
  - Two bytes: **-32768 – 32767**.
  - Four bytes: **-2147483648 – 2147483647**.
- Floating point (two bytes): **-671088.64 – 671760.96**.

These operands may be communication objects, internal variables where to store partial results, or even, for certain operations, numerical constants set by parameter in ETS.

On the other hand, it is important to bear in mind the following remarks:

- **Overflows in arithmetic operations** are dealt with by returning the value of the surpassed limit. For instance, a one-byte addition between the values 250 and 10 will return 255, as 255 is the limit of the range permitted for one byte.
- **Power failures** in the bus do not imply the loss of the values of the objects nor of the internal variables: after the power recovery, they will still maintain their previous values.
- **Divisions by zero** do not return a result; a function containing a divide-by-zero operation will be interrupted after the execution of the previous operation.
- **Multiplication and division of percentage values together** are performed according to the following examples:

- Two constants:

  Multiplying or dividing two percentage constants is done by considering that one of them is an integer. Therefore, if operand no. 1 is set to “25” and operand no. 2 is set to “2”, the result will be “50%” (note that the result of this operation is already known in advance).

- One constant and one object:

  This option is intended for cases where, for example, a shutter drive is required to move always to “k” times (being “k” a constant) the position of another shutter (which is received through a communication object). For example, if “k” is set to “3” and the current position of the latter shutter is “30%”, the former will move to 90%.

- Two objects:

  - To make –as in the above case– a shutter named “B” move to “k” times the position of another shutter named “A”, being “k” in this case a one-byte unsigned integer value (and not a constant), it is first necessary to convert “k” into an internal variable, being afterwards possible to multiply it with the status object of B.

  - In the same scenario, let “k%” be the value of a 1-byte percentage object (and not a constant). To make B move to “k%” of the position of A (e.g., 50% of 40%, thus, 20%), it is necessary to convert both into a pair of 2-byte floating point internal variables, multiply them, and finally convert the result into a 1-byte percentage value.

- The upper (most positive) and lower (most negative) extremes of the range of floating point values allowed by KNX (i.e., -671088.64 represented as 0xF800, and +670760.96 represented as 0x7FFF) operate as **+∞ (plus infinity) and –∞ (minus infinity)** respectively, as in the following examples:

  - (670760.96) * (-671088.64) = (-671088.64)
  - (-671088.64) * (-671088.64) = (670760.96)
  - (670760.96) * (34.00) = (670760.96)
  - (670760.96) * (-34.8) = (-671088.64)
  - (670760.96) / (65.2) = (670760.96)

- (670760.96) / (-341.12) = (-671088.64)
- (120000.00) / (670760.96) = (0.00)
- (-671088.64) / (670760.96) = (-1.00)
- (-671088.64) / (3500.66) = (-671088.64)

- If both a **re-send period** and a **delay time** are parameterised, the latter will only be taken into account before the first sending to the bus, after the execution of the operations. Once the delay time elapses, the result is sent to the bus and the re-send period count starts. The delay is not applied to the re-sending of the result, unless the function is triggered again and a new result applies for being sent to the bus – in such case, the delay time will be applied again to the first sending.

Join and send us your inquiries
about Zennio devices:

http://support.zennio.com

Zennio Avance y Tecnología S.L.
C/ Río Jarama, 132. Nave P-8.11
45007 Toledo (Spain).

Tel. +34 925 232 002.

www.zennio.com
info@zennio*.com*