KNX 8-channel ac/dc voltage input module - DIN rail mounting

I1 GND(N) I2 GND(N) I3 GND(N) I4 GND(N)
I1 I2 I3 I4
GW 90729
Input Module 8 in
in: 24..48Vdc 24..230Vac
KNX
GEWISS
I5 I6 I7 I8
I5 GND (N) I6 GND (N) I7 GND (N) I8 GND (N)

GW 90729

Technical manual

Contents

1 Introduction

This manual explains the functions of the **KNX 8-channel ac/dc voltage input module - DIN rail** (GW 90729) devices and how they are set and configured with the aid of the ETS configuration software.

2 Application

The KNX 8-channel ac/dc voltage input module - DIN rail makes it possible to connect up to 8 (24..48Vdc or 24..230Vac) voltage input push-buttons or contacts and sent the relative commands to actuator devices via the KNX BUS.
The module is powered by the BUS line and equipped with 8 amber LEDs signalling the status of the inputs.
The module is assembled on the DIN rail, inside the electric boards or junction boxes.

Each input is configured with the ETS software to create one of the functions listed below.

Input management / Transmission of objects on the BUS:
- management of edges with sending of sequences (1 bit, 2 bit, 1 byte, 2 byte, 3 byte, 4 byte, 14 byte) with 8 communication objects and timed intervals
- management of short/long contact closure with command transmission (1 bit, 2 bit, 1 byte, 2 byte, 3 byte, 4 byte, 14 byte)
- enabling/locking of inputs

Scenes:
- management of scenes with 1-byte items
- sending of scene storing commands
- scene sequence management

Priority commands:
- sending of priority commands

Roller shutters/curtain command:
- with single or double push-button

Dimmer command:
- with single or double push-button
- with stop telegram or cyclical send
- with sending of the light intensity value (0%-100%)

Pulse count:
- on ascent or descent fronts, or on both
- counter for 1 byte, 2 byte, 4 byte
- transmission on variation and/or cyclical (value counted on the BUS)
- overflow signalling on the BUS

multiple press/closing contact
- management of contact closure with consecutive (max. 4) pressing operations (1 bit, 2 bit, 1 byte, 2 byte, 3 byte, 4 byte, 14 byte)

Switchover sequences:
- with 1-bit objects on BUS (from 2 to 8)

2.1 Association limits

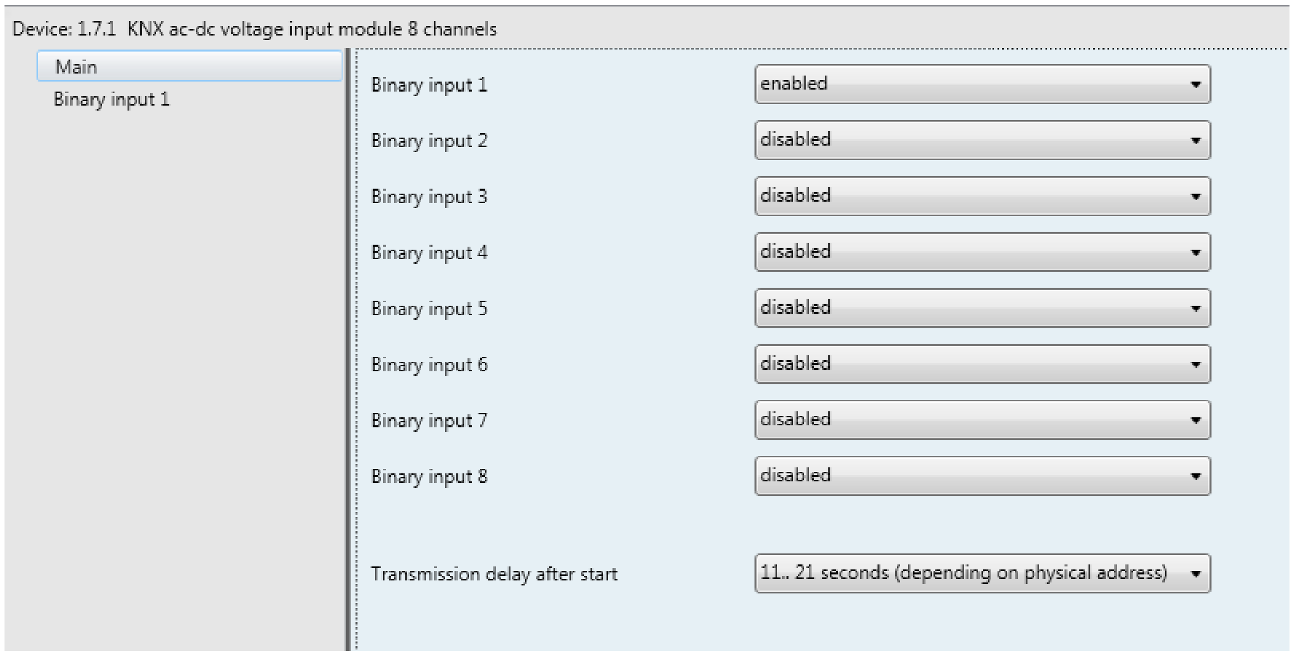
Maximum number of group addresses:	254
Maximum number of associations:	254

This means that up to 254 group addresses can be defined, and up to 254 associations can be made (communication objects and group addresses).

3 *“Main”* menu

The **Main** menu contains the application parameters for all the 8 input channels implemented by the device.

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Binary input 1	enabled
Binary input 2	disabled
Binary input 3	disabled
Binary input 4	disabled
Binary input 5	disabled
Binary input 6	disabled
Binary input 7	disabled
Binary input 8	disabled
Transmission delay after start	11.. 21 seconds (depending on physical address)

Fig. 3.1

3.1 Parameters

➢ *3.1.1 Binary input X*

Each of the 8 binary input channels implemented by the module can be managed autonomously, carrying out an autonomous function with respect to the others. The parameters **“Binary input 1”**, **“Binary input 2”**, **“Binary input 3”**, **“Binary input 4”**, **“Binary input 5”**, **“Binary input 6”**, **“Binary input 7”**and **“Binary input 8”** make it possible to enable the configuration of the relative input channels, displaying the configuration menus. The values that can be set are:

- **disabled** **(default value)**
- enabled

selecting **enabled**, displays the configuration menu **Binary input 1**, **Binary input 2**, **Binary input 3**, **Binary input 4**, **Binary input 5**, **Binary input 6**, **Binary input 7** or **Binary input 8** (see chap. 4 “Binary input X”).

➢ *3.1.2 Transmission delay after start*

To ensure that, with multiple devices in the line, the telegrams sent by the various devices do not collide when the bus voltage is restored, it is possible to define the time that must pass after which the device may transmit the telegrams on the bus following a drop/recovery of the bus supply voltage. The parameter **“Transmission delay after start”** is used to set this delay. The values that can be set are:

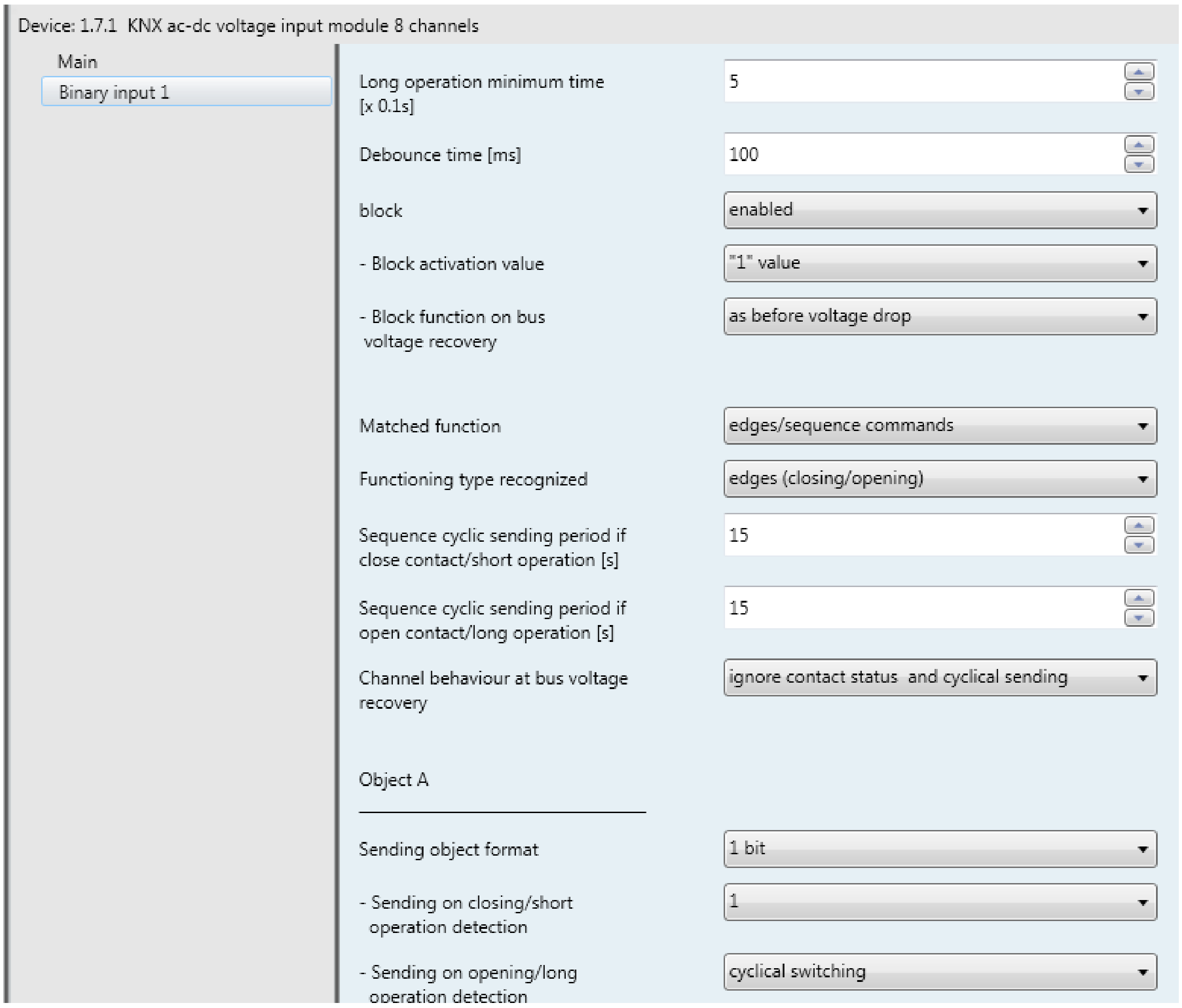
- **11.. 21 seconds (depending on physical address)** **(default value)**
- 5.. 9 seconds
- 11 seconds
- 13 seconds
- 15 seconds
- 17 seconds
- 19 seconds
- 21 seconds
- no delay

Setting the values **11.. 21 seconds (depending on physical address)** and **5.. 9 seconds**, the device automatically calculates the transmission delay according to an algorithm that examines the physical address of the device itself; the presented values (11/21 or 5/9) indicate the extremes of the value interval that can be calculated.
The switch-on time of the device is estimated as approx. 8 seconds.

4 *“Binary input X”* menu

If a binary input is enabled, a dedicated menu is displayed for each input called **Binary input x** (x = 1 .. 8, is the input index). The menu structure changes based on the value set for the **“Matched function”** parameter. For the sake of simplicity, the parameters enabled according to the value set for the above parameter are listed in the following paragraphs.

The basic structure of the menu is as follows:



Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	edges/sequence commands
Functioning type recognized	edges (closing/opening)
Sequence cyclic sending period if close contact/short operation [s]	15
Sequence cyclic sending period if open contact/long operation [s]	15
Channel behaviour at bus voltage recovery	ignore contact status and cyclical sending

Object A

Sending object format	1 bit
- Sending on closing/short operation detection	1
- Sending on opening/long operation detection	cyclical switching

Fig. 4.1: “Binary input X” menu (some parameters)

4.1 Parameters

➢ *4.1.1 Long operation minimum time [x 0.1s]*

Many of the functions that the binary inputs can perform require differentiation between short and long operations. The parameter **“Long operation minimum time [x 0.1s]”** can be used to define the minimum effective time the device must detect the closure of the contact in order to distinguish a short operation from a long one. The possible values are:

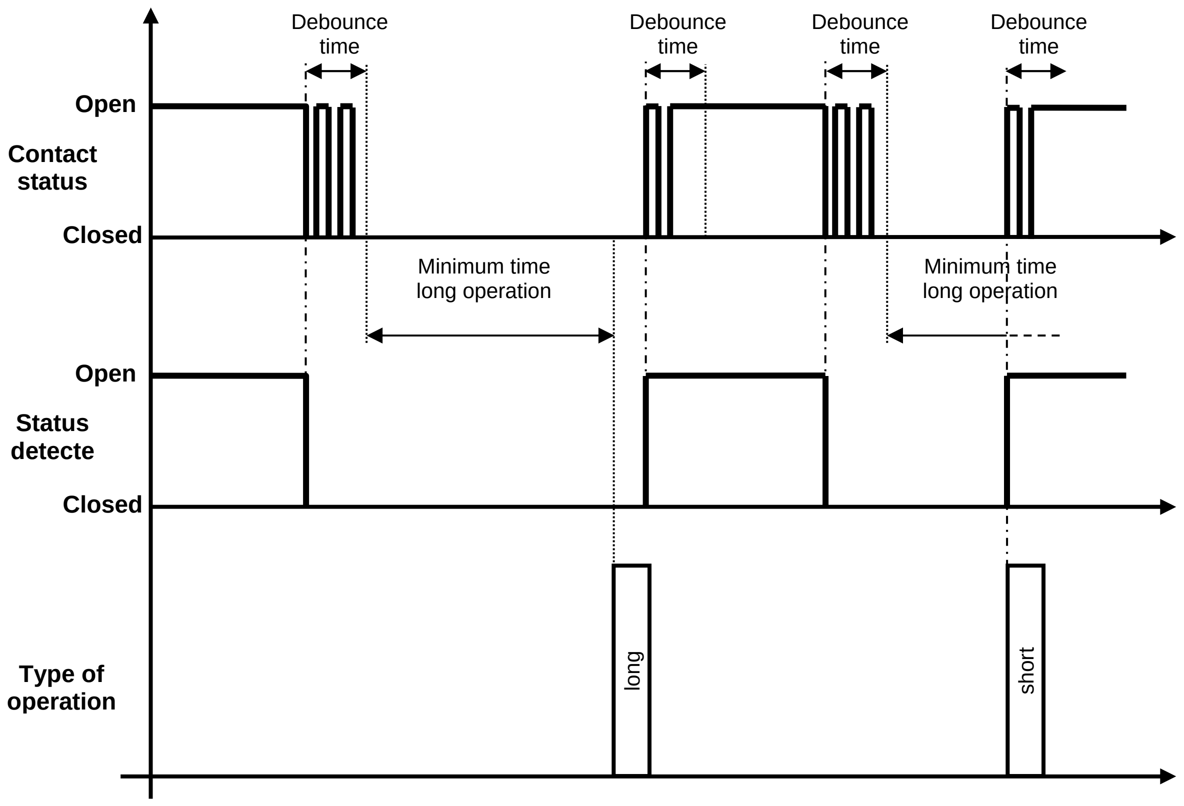
- from 3 to 150 with step 1, **5 (default value)**

➢ *4.1.2 Debounce time [ms]*

When an electro-mechanical device such as a push-button is pressed, there is a series of brief bounces (quick closing and opening of the contact) before the contact shifts definitively to the open or closed status; if suitable precautions are not taken, these bounces may be detected by the application software and interpreted as multiple command activations, causing subsequent device malfunctioning.
Given that the duration of these bounces depends on the type of device used, a function has been added to the device software to avoid the problem. It basically involves inserting a delay time between the reading moments of the push-button contact status so that when a contact status variation is detected, a specific time must pass before the device can detect another variation.
This value can be set in the **"Debounce time [ms]"** parameter. The values that can be set are:

- from 10 to 255 with steps of 1, **100 (default value)**

The following chart summarises the concepts of **"Long operation minimum time [x 0.1s]"** and **"Debounce time [ms]"** explained above.



Starting from the top, the first chart shows a simulation of the time trend of the push-button status. The second chart shows the time trend of the push-button status detected by the device software that filters the contact disturbance (bounce) for a time equal to **T**$_{\textbf{debounce}}$ starting from the moment when the first variation is detected.
At the end of the debounce time, the software re-reads the contact status and, if it is the same as the last status detected and if the variation is from open status to closed status (push-button pressed), it activates a timer whose initial value is the one set in **"Long operation minimum time [x 0.1s]"**. If the timer expires before the status variation from closed to open is detected, the software interprets this as a long operation; otherwise, the timer is blocked and the action is considered a short operation, as shown in the third chart.

➢ *4.1.3 Block*

To inhibit the binary input when sending commands associated with the closure/opening or long/short enabling of the contact, the block function must be activated: this function inhibits the detection of the closure/opening or long/short enabling of the contact, thereby preventing the device from sending the telegrams associated with these events on the BUS. If it is activated, any possible status variation will not be interpreted until a block deactivation command is received.
The parameter for enabling the function is the **"Block"** parameter that can take the following values:

- **disabled (default value)**
- enabled

If **enabled** is selected, this displays the parameters **"Block activation value"** and **"block function on bus voltage recovery"** and the communication object ***IN.x - Block*** (Data Point Type: 1.003 DPT_Enable), with which you can activate the function via the BUS command.

In particular cases where a front (opening or closure) or operation (short or long) is associated with the cyclical sending of a command/value, the block works in the following way:

a. if the block is activated while the cyclical sending is active, the device continues to send cyclically throughout the period in which the block is active. When the block is deactivated, the activation condition of the cyclical sending will be checked again. if it continues to be checked, the cyclical sending will continue, otherwise, the cyclical sending will end (even if the variation occurred while the block was active, so the sending of the telegram on front detection was inhibited).
b. if the block is activated while the cyclical sending is not active, the device does not react. When the block is deactivated, the cyclical sending condition will be checked and the necessary actions will be taken (even if the variation occurred while the block was active).

The parameter **"Block activation value"** makes it possible to set which logic value the bit received via BUS telegram should assume to activate the block function; the values that can be set are:

- value "0"
- **value "1" (default value)**

The parameter **"Block function on bus voltage recovery"** is used to set the status of the block function on BUS voltage recovery. The values that can be set are:

- disabled
- enabled
- **as before voltage drop (default value)**

➢ *4.1.4 Matched function*

The parameter used to define the function implemented by the binary input is **"Matched function".**
The values that can be set are:

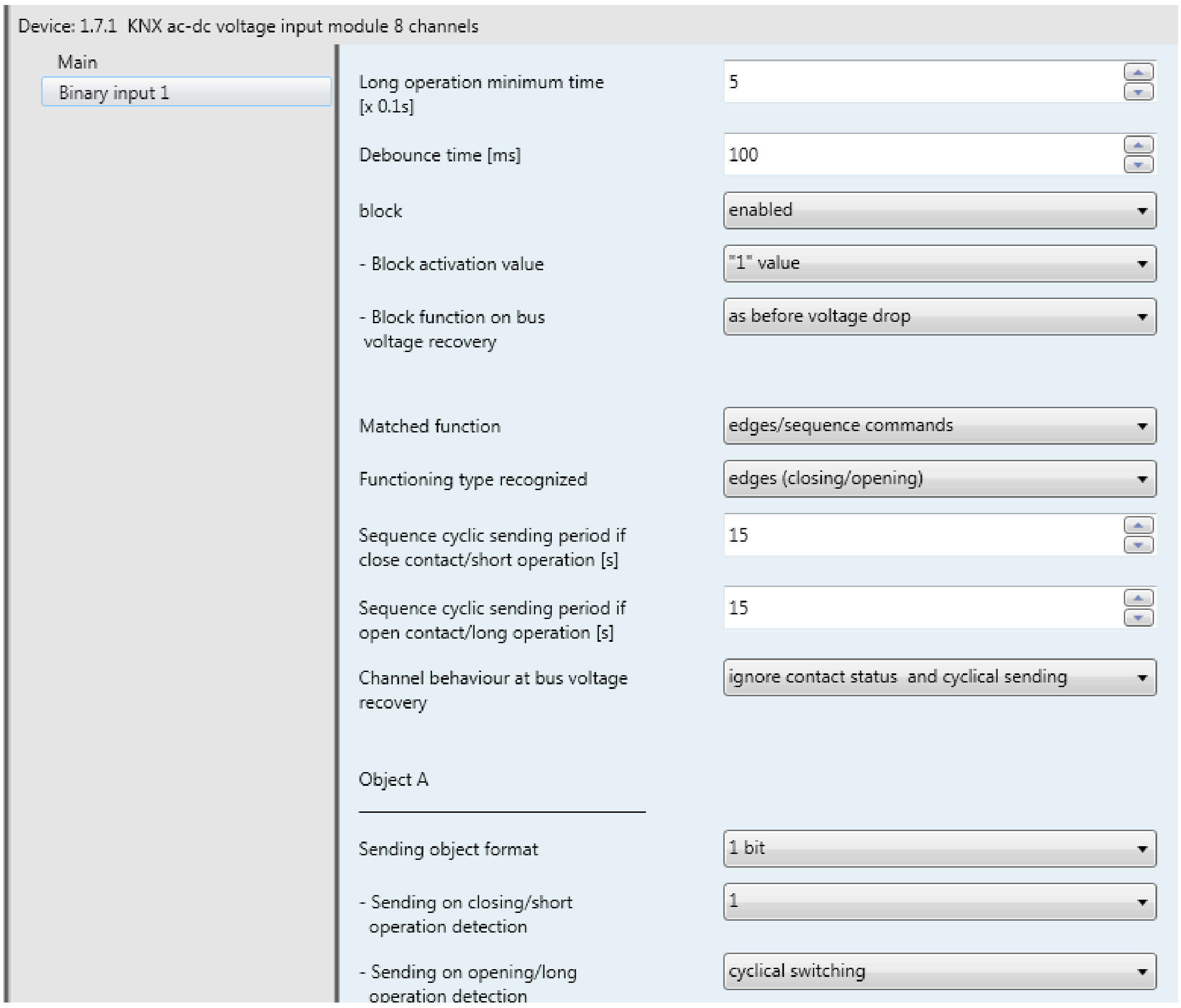
- **edges/sequence commands (default value)**
  (See paragraph 4.1.4.1. "edges/sequence commands" function)
- 1 push-button + stop dimmer
  (See paragraph 4.1.4.2. "1 push-button + stop dimmer" function)
- cyclic sending 1 push-button dimmer
  (See paragraph 4.1.4.3. "Cyclic sending 1 push-button dimmer" function)
- 1 push-button shutter control
  (See paragraph 4.1.4.4. "1 push-button shutter control" function)
- 2 push-button + stop dimmer
  (See paragraph 4.1.4.5. "2 push-button + stop dimmer" function)
- cyclic sending 2 push-button dimmer
  (See paragraph 4.1.4.6. "Cyclic sending 2 push-button dimmer" function)
- 2 push button roller shutters control
  (See paragraph 4.1.4.7. "2 push button roller shutters control" function)

- scene management
  (See paragraph 4.1.4.8. ”Scene management” function)
- switching sequences
  (See paragraph 4.1.4.9. “Switching sequences” function)
- pulse counter
  (See paragraph 4.1.4.10. “Pulse counter” function)
- multiple press/closing contact
  (See paragraph 4.1.4.11. “Multiple press/closing contact” function)

➢ *4.1.4.1 “Edges/Sequence commands” function*

This function is used to set the type and number of commands to send after a status change has been detected, for up to a total of 4 commands per input. The value of the command can be differentiated according to the event detected (closure/opening, or short/long operation). The sending of commands can also be delayed with a set fixed time, and the cyclical sending of command telegrams can be enabled.

The basic structure of the menu is as follows:



Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	edges/sequence commands
Functioning type recognized	edges (closing/opening)
Sequence cyclic sending period if close contact/short operation [s]	15
Sequence cyclic sending period if open contact/long operation [s]	15
Channel behaviour at bus voltage recovery	ignore contact status and cyclical sending
Object A	
Sending object format	1 bit
- Sending on closing/short operation detection	1
- Sending on opening/long operation detection	cyclical switching

Fig. 4.2: “Edges/Sequence commands” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.1.1 Type of recognised activation (Functioning type recognized)*

The parameter **"Functioning type recognized"** is used to define which type of contact operation generates the sending of the sequence commands. The values that can be set are:

- **edges (closure/opening)** **(default value)**
- short operation/long operation

➢ *4.1.4.1.2 Sequence cyclic sending period if close contact/short operation [s]*

The parameter **"Sequence cyclic sending period if close contact/short operation [s]"** parameter is used to set the repeat period for the sequence commands associated with the closed contact (or short operation) event; The values that can be set are:

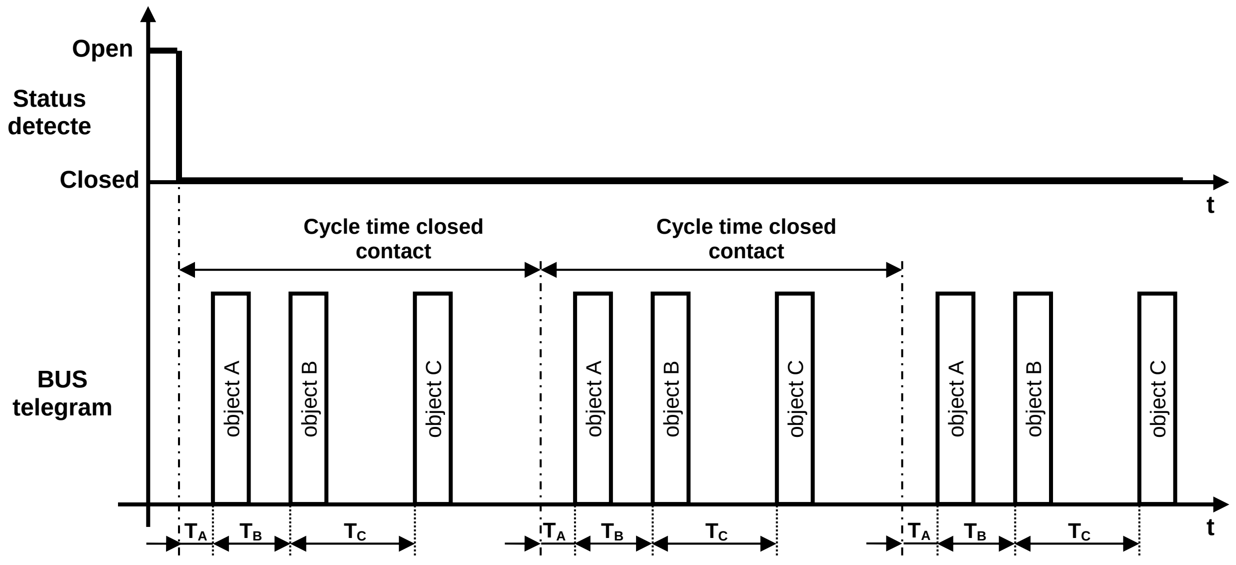
- from 1 to 65535 with steps of 1, **15** **(default value)**

➢ *4.1.4.1.3 Sequence cyclic sending period if open contact/prolonged selection [s]*

The parameter **"Sequence cyclic sending period if open contact/short operation [s]"** parameter is used to set the repeat period for the sequence commands associated with the open contact (or long operation) event; The values that can be set are:

- from 1 to 65535 with steps of 1, **15 (default value)**

The count of the cyclical sending time is initialised in the moment when the operation associated with cyclical sending is detected. The commands are repeated at the end of the cycle time, on the basis of the delays set (the entire command sequence is repeated). The following chart summarises the concept.



T_A= Delay on sending object A
T_B= Delay on sending object B
T_C= Delay on sending object C

The chart shows that, once contact closure has been detected, the cycle time counter is initialised along with the delay on the sending of the first object (in this case, object A). At the end of the cycle time, the whole sequence (including delays) is repeated. Throughout the repeat, the contact remains closed.

➢ *4.1.4.1.4 Channel behaviour at BUS voltage recovery*

When the BUS voltage is restored, the behaviour of the binary input (with regard to the sending of the sequence and the cyclical sending of telegrams) can be defined via the **“Channel behaviour at bus voltage recovery”** parameter. The values that can be set are:

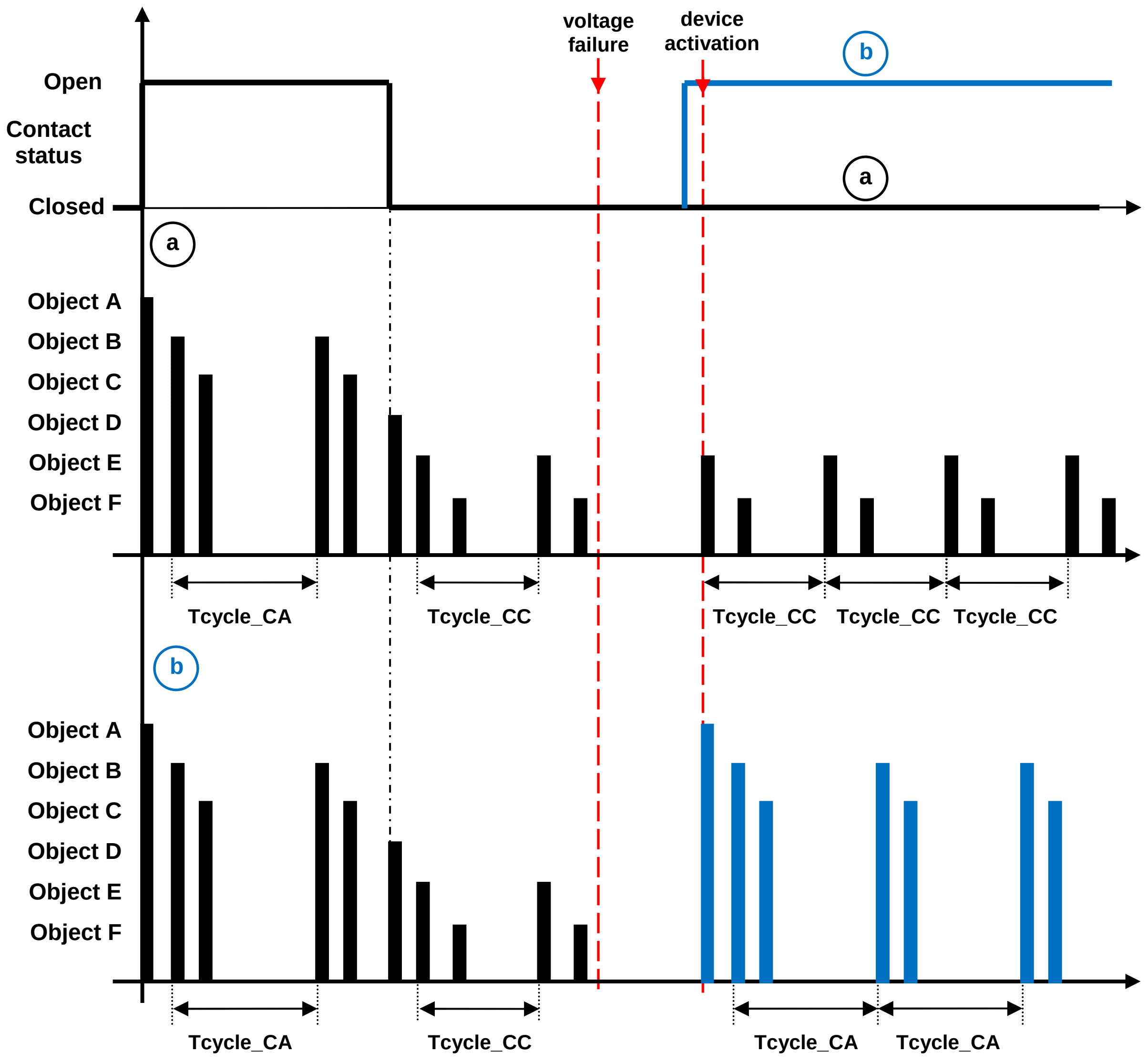
- **ignore contact status and cyclical sending** **(default value)**
- evaluate contact status and cyclical sending

By selecting **evaluate contact status and cyclical sending**, the device behaves in the following way:

- if the recognised type of operation is **edges (closing/opening)**, the device checks the contact status and:
  a) if the current status is the same as before the voltage failure, the device evaluates the value set in the **“Sending object condition**” items of all the objects of the sequence, and sends only those telegrams for which cyclical sending is enabled (as if the voltage failure had not occurred).
  b) if the current status is different from the one before the voltage failure, the device interprets the event as a new edge (occurring at switch-on) and consequently initialises the sending of the entire sequence.
- if the recognised type of operation is **short operation/long operation**, the device checks the last operation recognised before the voltage failure and, after evaluating the value set for the **“Sending object condition**” items of all the objects of the sequence, it sends only those telegrams for which cyclical sending is enabled (as if the voltage failure had not occurred).

If the value **ignore contact status and cyclical sending** is selected, no telegram is sent when the BUS voltage is restored; the status variation or a short/long operation must be detected in order to reactivate the sending of the sequence.

The following chart helps you to understand the behaviour of the device upon BUS recovery if the value "evaluate contact status and cyclical sending" is selected and the type of operation recognised is "edges" (closure/opening).



In the example above, objects A, B, C are sent on the contact opening edge and objects B and C are also sent cyclically. Objects D, E, F are sent on the contact closure edge and objects E and F are also sent cyclically. Chart "a" shows the condition in which the contact status when the device is activated following BUS voltage failure is the same as before that failure; vice versa, in chart "b" the contact status when the device is activated is different from that prior to the failure.

Chart "a"

- On the opening of the contact, the device sends the sequence of telegrams A, B and C on the basis of the set sending delays
- after a period of time equal to the period of cyclical telegram sending with an open contact (Tcycle_CA), the device again sends objects B, C for which cyclical sending is enabled
- on the closure of the contact, the device sends the sequence of telegrams D, E and F on the basis of the set sending delays
- after a period of time equal to the period of cyclical telegram sending with a closed contact (Tcycle_CC), the device again sends objects E, F for which cyclical sending is enabled
- upon recovery after a BUS voltage failure, the device detects that the contact status is "closed", as it was prior to the failure. At this point, it sends telegrams E, F for which cyclical sending is enabled. Object D is not sent
- After a period of time equal to the period of cyclical telegram sending with a closed contact (Tcycle_CC), the device again sends objects E, F for which cyclical sending is enabled. This condition continues until contact opening is detected.

Chart "b"

- On the opening of the contact, the device sends the sequence of telegrams A, B and C on the basis of the set sending delays
- after a period of time equal to the period of cyclical telegram sending with an open contact (Tcycle_CA), the device again sends objects B, C for which cyclical sending is enabled
- on the closure of the contact, the device sends the sequence of telegrams D, E and F on the basis of the set sending delays
- after a period of time equal to the period of cyclical telegram sending with a closed contact (Tcycle_CC), the device again sends objects E, F for which cyclical sending is enabled
- upon recovery after a BUS voltage failure, the device detects that the contact status is "open", unlike the condition prior to the failure. At this point, it sends telegrams A, B and C on the basis of the set sending delays, as if it had detected an opening edge at the time of activation
- after a period of time equal to the period of cyclical telegram sending with an open contact (Tcycle_CA), the device again sends objects B, C for which cyclical sending is enabled. This condition continues until contact closure is detected.

➢ *4.1.4.1.5 Object A-B-C-D*

For each binary input, up to 4 different objects can be sent (distinguished by the letters A, B, C and D) on the basis of the closure (or short operation) or opening (or long operation) of the contact; object A is always enabled, whereas the parameter **"Object z"** (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

If **enable** is selected, the following parameters will be visualised: **"Sending object format"**, **"Sending on closing/short operation detection"**, **"Sending on opening/long operation detection"** and **"Object sending delay [s]"**. These are grouped together in the **z object** sub-group (where z indicates the object associated with the binary input, included between **A** and **D)**.

➢ *4.1.4.1.6 Sending object format*

The parameter **"Sending object format"** makes it possible to set the format and code of the bus telegram that will be sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit
- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value
- 3 bytes RGB colour
- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes
-

The value set for this item will alter the values that can be set for the **"Sending on closing/short operation detection"** and **"Sending on opening/long operation detection"** parameters.
The parameter **"Sending on closing/short operation detection"** parameter is used to set the command or value to be sent following the detection of the closure or short operation of the contact (depending on the type of operation selected) associated with the binary input.

The parameter **“Sending on opening/long operation detection”** parameter is used to set the command or value to be sent following the detection of the opening or long operation of the contact (depending on the type of operation selected) associated with the binary input.

- If the format of the object to send is **1 bit**, this displays the communication object ***IN.x - z object 1 bit value*** (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the two parameters above are:

  - **no action/stop cyclic sending object** **(default value on detection of opening)**
  - 0
  - **1** **(closing detection default value)**
  - cyclical switching

  Selecting the value **cyclical switching**, the **“Object status feedback”** parameter will be shown, which makes it possible to enable and display the ***IN.x - z object status feedback*** communication object (Data Point Type: 1.001 DPT_Switch); by enabling this object, when the status feedback telegram is received for the object in question, the command that the device will send (via the ***IN.x - z object 1 bit value*** object) when the event associated with the cyclical switching detected will be the opposite of the value generated by the most recent event between the BUS value received on the ***IN.x - z object status feedback*** object and the last value sent (via the ***IN.x - z object 1 bit value*** object).
  The **“Status feedback object”** parameter may have the following values:

  - **disabled** **(default value)**
  - enabled

  Selecting the value **enabled** displays the ***IN.x - z object status feedback*** communication object. In this case, every time the BUS voltage is restored you must send a status read request on this object in order to update the interface about the status of the devices connected.

- If the format of the object to send is **2 bit**, this displays the communication object ***IN.x - z object 2 bit value*** (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default opening value)**
  - **on forcing active (down)** **(default closing value)**
  - activate OFF forcing (up)
  - deactivate forcing [=forcing deactivation]
  - cyclical switching - ON forcing /OFF forcing
  - cyclical switching - ON forcing/forcing deactivation
  - cyclical switching - OFF forcing/forcing deactivation

  By selecting **cyclical switching**, in this case no communication object will be displayed as the device is always updated about the function activation status.

- If the format of the object to send is **1 byte unsigned**, the ***IN.x - 1 byte value z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default opening value)**
  - **send value** **(default closing value)**

  By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **“Value (0 .. 255)”** which can assume the following values:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte signed**, the ***IN.x - 1 byte value z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the two parameters listed above are:

- **no action/stop cyclic sending object** **(default opening value)**
- **send value** **(default closing value)**

By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **"Value (-128 .. 127)"** which can assume the following values:

- from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - 1 byte value z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default opening value)**
  - **send value** **(default closing value)**

  By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **"Value (0% .. 100%)"** which can assume the following values:

  - from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - 1 byte value z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default opening value)**
  - auto
  - **comfort** **(closure default value)**
  - pre-comfort
  - economy
  - off (building protection)
  - cyclical switching (thermostat)
  - cyclical switching (timed thermostat)

  By selecting **cyclical switching**, in this case no communication object will be displayed as the device is always updated about the function activation status.
  By selecting **cyclical switching (thermostat)**, each time the associated event (closing/opening or short/long operation) is detected, the device will send a new temperature adjustment mode (HVAC) in the order *Comfort→ Precomfort→ Economy→ Off→ Comfort* … By selecting **cyclical switching (timed thermostat),** each time **the** associated event (closing/opening or short/long operation) is detected, the device will send a new temperature adjustment mode (HVAC) in the order *Comfort→ Precomfort→ Economy→ Off→ Auto → Comfort* …

- If the format of the object to send is **2 byte unsigned**, the ***IN.x 2 byte value z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default value on detection of opening)**
  - **send value** **(closing detection default value)**

  By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **"Value (0 .. 65535)"** which can assume the following values:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte signed**, the ***IN.x 2 byte value z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object 0** **(default value on detection of opening)**
  - **send value 1** **(closing detection default value)**

By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **“Value (-32768 .. +32767)”** which can assume the following values:

- from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the ***IN.x 3 byte value z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the two parameters above are:

  - **no action/stop cyclic sending object** **(default value on detection of opening)**
  - **send value** **(closing detection default value)**

  By setting **send value**, you can select the colour to be sent via the **“Colour”** dummy parameter; The values that can be set are:

  - **white** **(default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

  By selecting **customise**, the following parameters are made visible: **“Value of RED component (0 .. 255)”**, **“Value of GREEN component (0 .. 255)”** and **“Value of BLUE component (0 .. 255)”**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
  The values that can be set are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte unsigned**, the ***IN.x 4 byte value z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default value on detection of opening)**
  - **send value** **(closing detection default value)**

  By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **“Value (0 .. 4294967295)”** which can assume the following values:

  - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte signed**, the ***IN.x 4 byte value z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the two parameters listed above are:

  - **no action/stop cyclic sending object** **(default value on detection of opening)**
  - **send value** **(closing detection default value)**

  By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **“Value (-2147483648 .. 2147483647)”** which can assume the following values:

  - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

- If the format of the object to send is **14 bytes**, the ***IN.x 14 byte value z object*** communication object will be visible (Data Point Type: 16.001 DPT_String_8859_1) and the values that can be set for the two parameters listed above are:

- **no action/stop cyclic sending object** **(default value on detection of opening)**
- **send value** **(closing detection default value)**

By setting **send value**, it is possible to define the value to be sent via the newly displayed parameter **"Value (ISO characters 8859-1)"** which can assume the following values:

- 14 alphanumeric characters with ISO/IEC coding 8859-1

NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with "unsigned and signed value" coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format "2/4 byte signed/unsigned value".

➢ *4.1.4.1.7 Object sending delay (0..255 seconds)*

The parameter **"Object sending delay (0.. 255 seconds)"** parameter sets the delay between the detection of the event associated with the sending of the command, and the effective sending of the command/value on the BUS. With regard to the objects that range from index B to index D, this parameter indicates the delay between sending the command/value associated with the object with the previous index (z-1) and sending the command/value associated with the object to which the parameter refers; the delay in these cases is calculated from the moment when the command/value associated with the object with the previous index (z-1) is sent, not from the moment of detection of the event that generated the sending (closure/opening or short/long operation).
The set delay will only be executed if the event in progress, associated with the object to which the parameter refers, is associated with any value other than **no action**; otherwise, the delay is ignored.
The parameter may assume the following values:

- from **0 (default value)** to 255 seconds, with steps of 1.

Note: if a sequence of commands with delays - activated by the detection of a specific event (closure/opening or short/long operation) - is being sent, then the detection of the opposite event will cause the termination of the sending of that sequence, but only if at least one of the actions associated with the detection of the latter event is different from **no action**; otherwise, the command/value sequence will be continue to be sent until the last command/value has been sent.

➢ *4.1.4.1.8 Cyclic sending object condition*

Given the possibility to interface various devices with the device input contacts, it may be useful to repeat the command telegrams at pre-set intervals (especially if there is a sensor interface). The **"Cyclic sending object condition"** parameter defines the conditions for the cyclical sending of the command telegrams. The values that can be set are:

- **never** **(default value)**
- in the case of an open contact/long operation
- in the case of a closed contact/short operation
- always

By selecting **never**, the device will only send the telegram with the set value on the BUS when the contact changes from closed to open or vice versa (or when a short/long operation is detected on the contact).

By selecting **in the case of an open contact/long operation**, the device will only send the telegram with the set value on the BUS when the contact changes from closed to open (or when a long operation is detected on the contact). As long as the contact remains open (or no other operation is recognised), the device will occasionally send the value associated with the event; if a new long operation is recognised, this cyclical sending is interrupted and the sending of the sequence associated with the detected operation restarts.

By selecting **in the case of a closed contact/short operation**, the device will only send the telegram with the set value on the BUS when the contact changes from open to closed (or when a short operation is detected on the contact). As long as the contact remains closed (or no other operation is recognised), the device will occasionally send the value associated with the event; if a new short operation is recognised, the sending of the sequence associated with the detected operation restarts.

By selecting **always**, the device will only send the telegram with the set value on the BUS when the contact changes from closed to open or vice versa (or when a short/long operation is detected on the contact). The command telegram associated with the detected event is repeated at regular intervals. If a short/long operation is recognised, this cyclical sending is interrupted and the sending of the sequence associated with the detected operation restarts.

If the value **no action/stop cyclic sending object** is associated with a specific operation for all the objects enabled, then the cyclical condition will be ignored even if it is enabled. If cyclical sending is active (determined by the setting of the other operation), this is terminated.

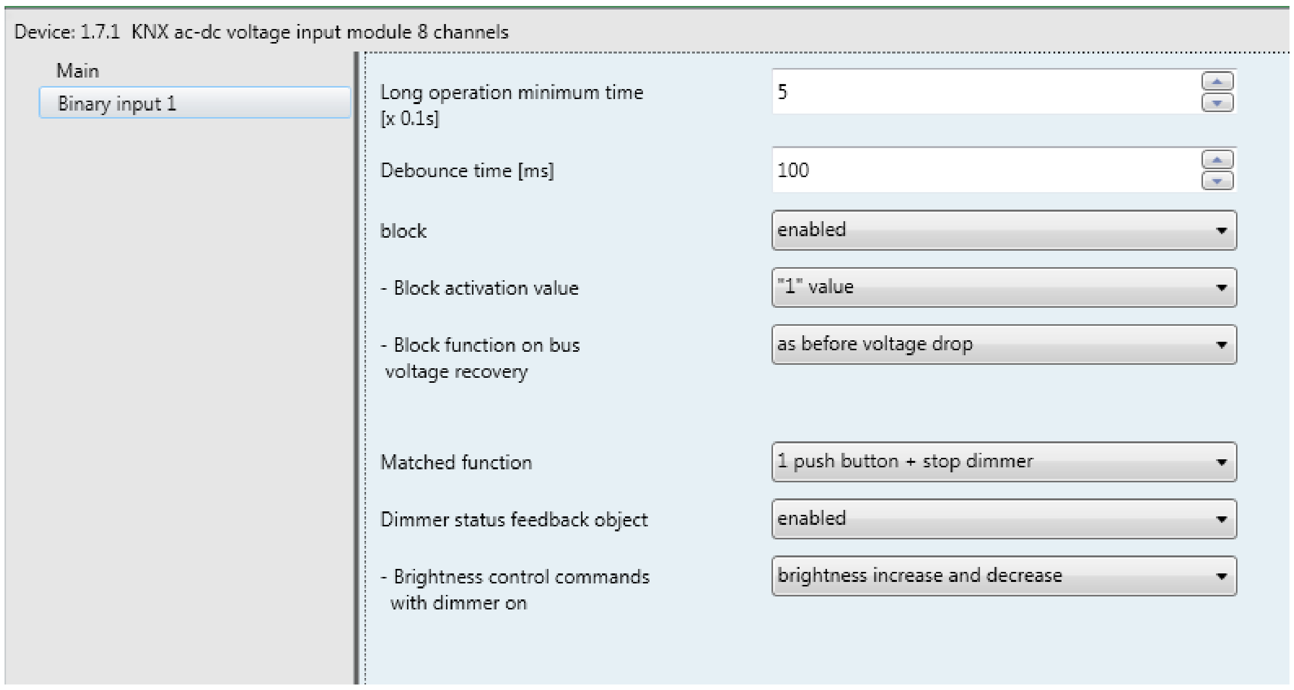
➢ *4.1.4.2 “1 push-button + stop dimmer" function*

This is used to configure the binary input to control a dimmer with a single button, increasing and decreasing dimmer brightness always using the same input.
For sending on/off telegrams and brightness control telegrams.
As there is only one input to manage the On/Off and brightness control functions, the operation is managed by differentiating between short operations and long operations:

- a long operation is interpreted as a brightness control command. When the contact is opened, an adjustment stop telegram is sent to stop the brightness increase/decrease operation for the dimmer and to fix the brightness value reached at the moment the stop control command was received.
- a short operation is interpreted as an on/off command.

Using this type of function, brightness control depends on the so-called brightness control characteristic curve, which varies from actuator to actuator, based on how the manufacturer designed the curve that regulates power, and as a result brightness. This means that the speed with which brightness reaches its maximum and minimum value does not depend on the commands sent from the device, but the latter regulates the brightness itself by stopping its increase/decrease based on the desired value. The communication objects enabled by this function are ***IN.x - Switching*** (Data Point Type: 1.001 DPT_Switch) and ***IN.x - Brightness dimming*** (Data Point Type: 3.007 DPT _Control_Dimming).

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	1 push button + stop dimmer
Dimmer status feedback object	enabled
- Brightness control commands with dimmer on	brightness increase and decrease

Fig. 4.3: “1 push button + stop dimmer” matched function menu

The normal behaviour of the device foresees that if the command to be sent is the opposite of the last command sent, this is transformed into:

- long operation: if the last sent command was an off command or a decrease brightness command, the new command will be an increase brightness command; vice versa, if the last sent command was an on command or an increase brightness command, the new command will be a decrease brightness command. In both cases, when the contact is opened, an adjustment stop telegram is sent to stop the brightness increase/decrease operation for the dimmer and to fix the brightness value reached at the moment the stop control command was received.
- short operation: if the last sent command was an on command, the new command will be an off command; vice versa, if the last sent command was an off command, the new command will be an on command; the brightness increase/decrease control commands in this case do not determine the value of the last command sent to distinguish the value of the new command to be sent.

This behaviour is modified if the user enables the ***IN.x - Dimmer status feedback*** communication object (Data Point Type: 1.001 DPT_Switch), via the **“Dimmer status feedback object”** parameter.

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.2.1 Dimmer status feedback object*

This parameter may have the following values:

- **disable** **(default value)**
- enable

If **enabled** is selected, the **“Brightness control commands with dimmer on”** parameter is visualised, along with the communication object ***IN.x - Dimmer status feedback***, which makes it possible to receive status feedback from the controlled dimmer actuator; The behaviour of the push-button panel is modified as follows:

- long operation: the commands that the device sends depend on the parameter **“Brightness control commands with dimmer on”**, which can assume the following values:

  - only brightness increase
  - only brightness decrease
  - **brightness increase and decrease** **(default value)**

  By setting **brightness increase and decrease**, if the value of the last two events "last sent command" and "dimmer status feedback" is ON, the new brightness control command to be sent will be the opposite of the last sent command. When the contact is opened, an adjustment stop telegram is sent to stop the brightness increase/decrease operation for the dimmer and to fix the brightness value reached at the moment the stop control command was received. If the value of the last of the two events "last sent command" and "dimmer status feedback" is OFF, the first command to be sent is increase brightness value, followed by sending the command opposite of the last one sent.
- short operation: if the value of the last of the two events "last sent command" and "dimmer status feedback" is ON, the new command will be an off command. Vice versa, if the value of the last of the two events "last sent command" and "dimmer status feedback" is OFF, the new command will be an on command.

If the feedback object is enabled, every time the BUS voltage is restored you must send a status read request on this object in order to update the device about the status of the devices connected.

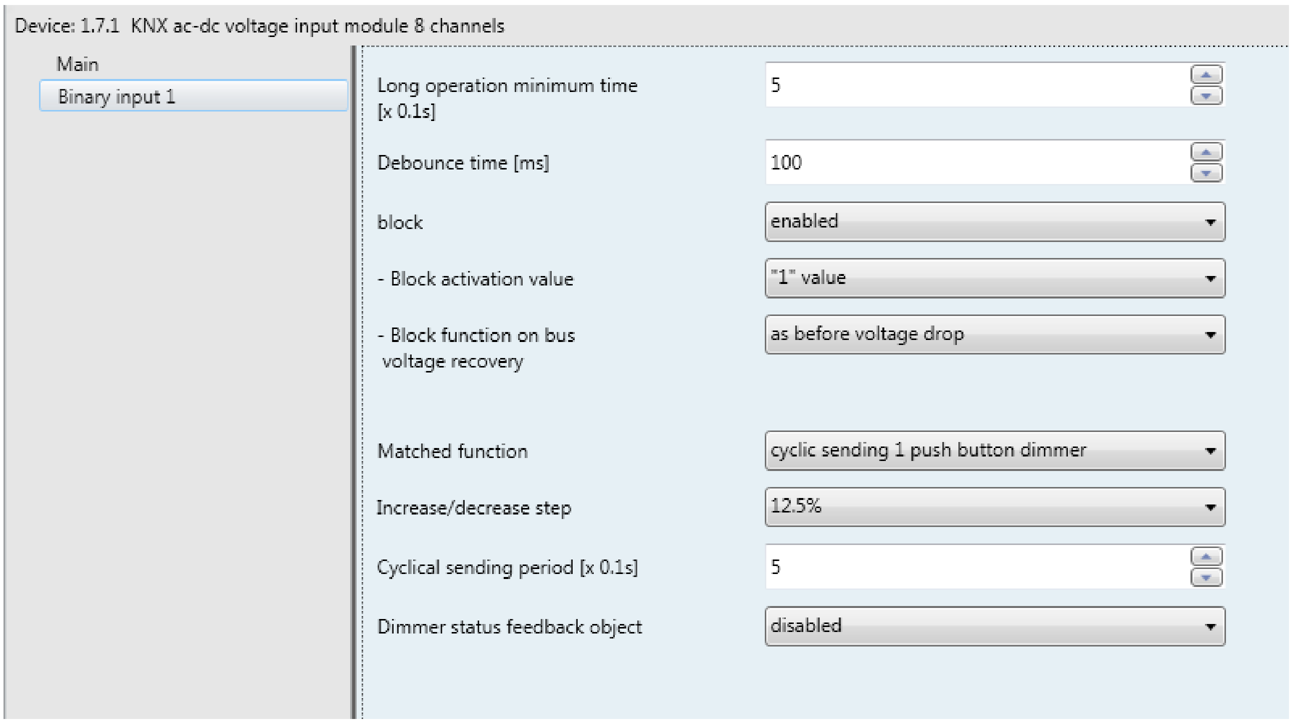
➢ *4.1.4.3 “Cyclic sending 1 push-button dimmer" function*

This is used to configure the binary input to control a dimmer with a single button, increasing and decreasing dimmer brightness always using the same button, with defined and settable control steps.
As there is only one input to manage the On/Off and brightness control functions, the operation is managed in the following way: with each activation, the command sent is the opposite to the last one sent. Furthermore, a distinction is made between short operations and long operations:

- a long operation is interpreted as a brightness control command. No telegram is sent when the contact is opened.
- a short operation is interpreted as an on/off command.

Unlike the ***1 push-button + stop dimmer*** function, it is possible to define both the brightness variation steps and the time that must elapse between the sending of one command and another when the long operation is drawn out over time. The sending of the "regulation stop" telegram on contact opening is not therefore necessary, because although the regulation does follow the characteristic power/brightness curve, it is the command sent by the device that determines the percentage variation. The communication objects enabled by this function are ***IN.x - Switching*** (Data Point Type: 1.001 DPT_Switch) and ***IN.x - Brightness dimming*** (Data Point Type: 3.007 DPT _Control_Dimming).

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	cyclic sending 1 push button dimmer
Increase/decrease step	12.5%
Cyclical sending period [x 0.1s]	5
Dimmer status feedback object	disabled

Fig. 4.4: "Cyclic sending 1 push button dimmer" matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.3.1 Increase/decrease step*

The parameter**"Increase/decrease step"** is used to set the percentage value of the brightness variation associated with the brightness increase/decrease commands. In this way, as soon as a long operation is detected, the device sends the first increase/decrease command with the set percentage. The values that can be set are:

- 100%
- 50%
- 25%
- **12.5%** **(default value)**
- 6.25%
- 3.125%
- 1.56%

➢ *4.1.4.3.2 Cyclical sending period [x 0.1s]*

If the contact remains closed after recognising the long operation, the device sends the command cyclically until contact opening is detected. The **"Cyclical sending period [x 0.1s]"** parameter is used to set the time that must pass between the sending of one increase/decrease command and another, if the contact remains closed after the recognition of a long operation. When the contact is opened, no telegram is sent; the cyclical sending of the brightness control commands is merely stopped.

The values that can be set for the parameter **”Cyclical sending period [x 0.1s]”** are:

- from 3 to 50 in steps of 1 **(default value 5)**

To sum up, when a long operation is detected, the device sends the first increase/decrease command with the set percentage and, if the contact remains closed, it sends the command cyclically until it detects the opening of the contact.

EXAMPLE: if long operation minimum time is set to ***0.5 sec***, and the **Increase/decrease step** parameter is set to **12.5%** and the **Cyclical sending period [x 0.1s]** parameter is set to **3** (0.3 sec) and contact closure is detected:
- 0.5 seconds after the detection of the contact closure, a long operation is detected and so the first 12.5% brightness increase/decrease telegram is sent
- from this moment, for every 0.3 seconds that contact remains closed, the device will send the 12.5% brightness increase/decrease command again and again until opening is detected
- when the contact is opened, no telegram is sent; the cyclical sending is merely stopped

➢ *4.1.4.3.3 Dimmer status feedback object*

As for the ***1 push button + stop dimmer*** function, it is possible to enable the dimmer status feedback object by changing the behaviour of the switching and control commands as described in paragraph 4.1.4.2 “1 push button + stop dimmer” function.
The parameter used to enable the feedback object is **“Dimmer status feedback object”** which can have the following values

- **disable (default value)**
- enable

If **enable** is selected, the **“Brightness control commands with dimmer on”** parameter is visible along with the ***IN.x - Dimmer status feedback*** communication object (Data Point Type: 1.001 DPT_Switch), which is used to receive the status feedback from the controlled dimmer actuator.
The parameter **“Brightness control commands with dimmer on”** can have the following values:

- only brightness increase
- only brightness decrease
- **brightness increase and decrease (default value)**

If the feedback object is enabled, every time the BUS voltage is restored you must send a status read request on this object in order to update the device about the status of the devices connected.

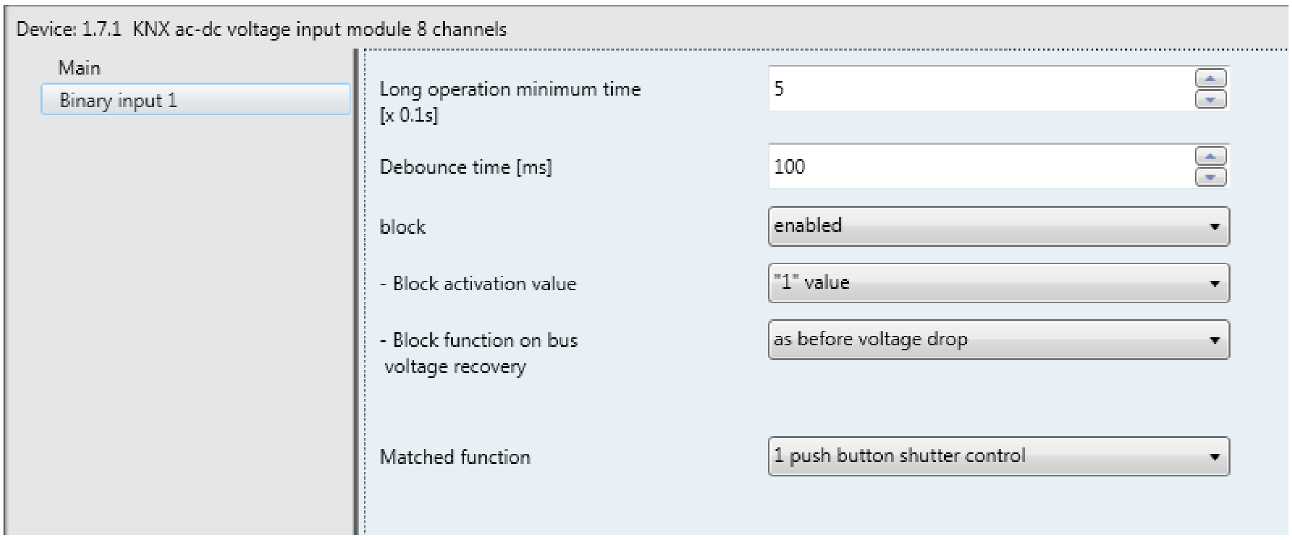
➢ *4.1.4.4 “1 push-button shutter control” function*

This is used to configure the binary input to control a shutter with a single button, regulating the upward and downward travel of the shutter and, depending on the device version, controlling louvres opening/closing.
As only one input manages the louvre up/down and control functions, operation is managed so that with each activation, a command is sent that is the opposite to the last movement signal received by the actuator that manages the shutter. There is a difference between short and long operations:
- a long operation is interpreted as an up/down movement command. The new value to be sent is the opposite of the last value sent via the ***IN.x - Shutter movement*** object or of the movement feedback received via the ***IN.x - Movement feedback*** object, depending on which of the two events occurred last. If the last event that occurred is “upward movement feedback reception” or “sending upward movement command”, the new command will be a "downward movement" command and vice versa.

- a short operation is interpreted as a louvre control command. The new value to be sent depends on the last value sent via the ***IN.x - Shutter movement*** object or the movement feedback received via the ***IN.x - Movement feedback*** object, depending on which of the two events occurred last; if the last event that occurred is “upward movement feedback reception” or “send upward movement command”, the command will be a "closing louvres adjustment" command, and vice versa. If the shutter is moving, the louvre adjustment command will only stop the shutter up/down movement.

The communication objects enabled by this function are ***IN.x - Shutter movement*** (Data Point Type: 1.008 DPT_UpDown), ***IN.x - Louvre stop/adjustment*** (Data Point Type: 1.007 DPT_Step) and ***IN.x - Movement feedback*** (Data Point Type: 1.008 DPT_UpDown).

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]: 5
Debounce time [ms]: 100
block: enabled
- Block activation value: "1" value
- Block function on bus voltage recovery: as before voltage drop
Matched function: 1 push button shutter control

Fig. 4.5: “1 push button shutter control” matched function menu

No new parameters are enabled with this function.

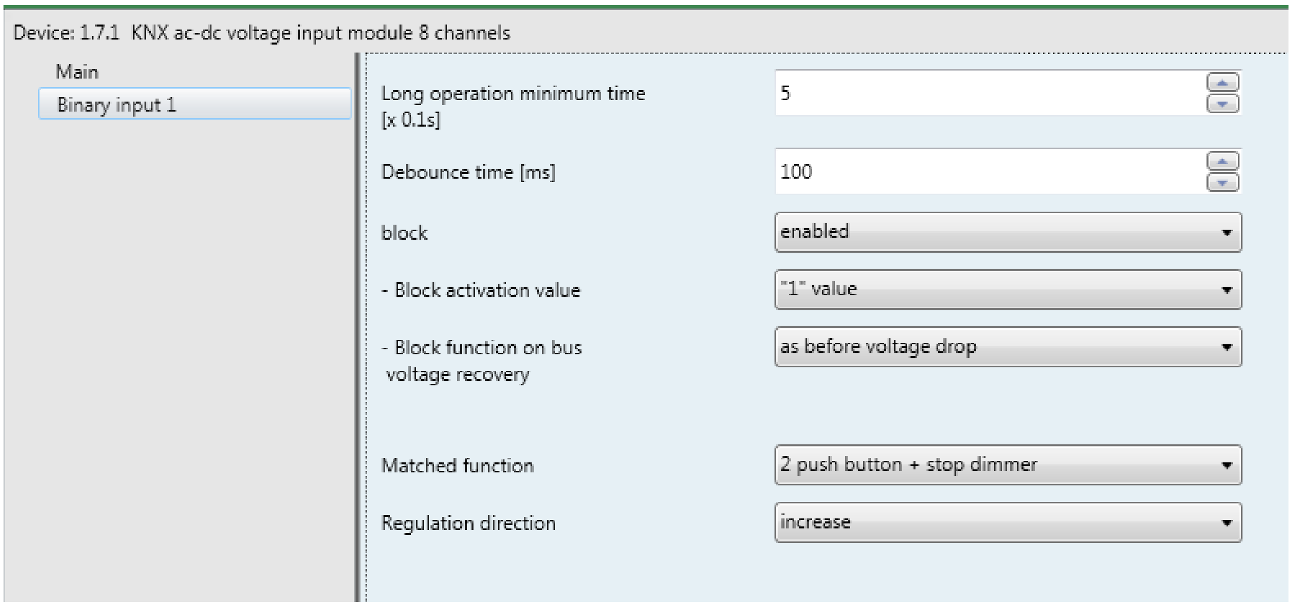
➢ *4.1.4.5 “2 push-button + stop dimmer" function*

This is used to configure the channel to control a dimmer with two buttons, managing in this case only one of the two control directions (brightness increase/decrease).
On or off telegrams and brightness increase or decrease telegrams can be sent, based on the configured control direction. Also in this case, there is a difference between short and long operations:

- a long operation is interpreted as a brightness control command. If the set control direction is "increase", the control will only be increasing, otherwise if the set control direction is "decrease" the control will be decreasing. In both cases, when reopening the contact, an adjustment stop telegram is sent to stop the brightness increase or decrease operation for the dimmer and to fix the brightness value reached at the moment the stop control command was received.
- a short operation is transformed in to an on or off command depending on the set control direction. If the set control direction is "increase" the sent command will only be an ON command. If the set control direction is "decrease" the sent command will only be an OFF command.

Using this type of function, brightness control depends on the so-called brightness control characteristic curve, which varies from device to device, based on how the manufacturer designed the curve that regulates power, and as a result brightness. The communication objects enabled by this function are ***IN.x - Switching*** (Data Point Type: 1.001 DPT_Switch) and ***IN.x - Brightness dimming*** (Data Point Type: 3.007 DPT _Control_Dimming).

The structure of the menu is as follows:



Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	2 push button + stop dimmer
Regulation direction	increase

Fig. 4.6: “2 push button + stop dimmer” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.5.1 Regulation direction*

The parameter **“Regulation direction”** configures the control direction of the brightness that the channel controls; The values that can be set are:

- **Increase**	**(uneven channel default value)**
- **Decrease**	**(even channel default value)**

Selecting **increase**, the sent commands will be 'increase brightness 100%' or ON, depending on the recognized activation; otherwise, selecting **decrease** the sent commands will be 'decrease brightness 100%' or OFF.

➢ *4.1.4.6 “Cyclic sending 2 push-button dimmer” function*

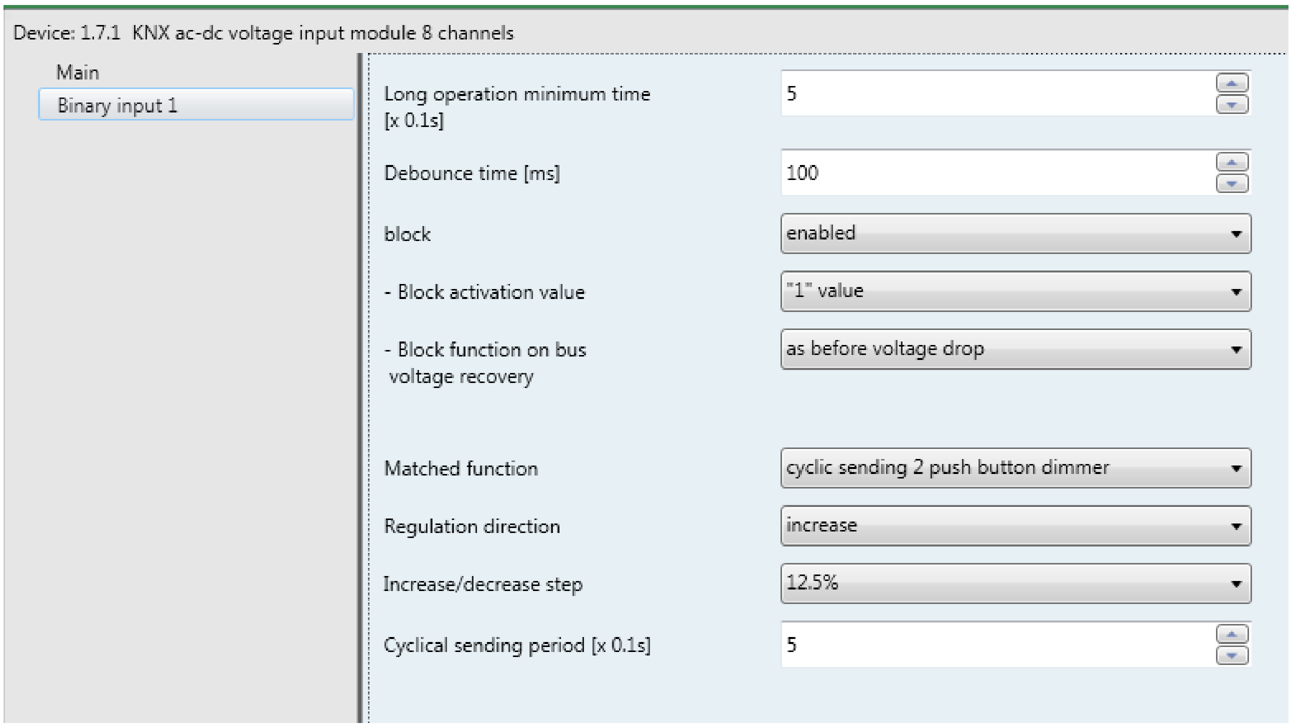
This is used to configure the channel to control a dimmer with two buttons, managing in this case only one of the two control directions (brightness increase/decrease).
On or off telegrams and brightness increase or decrease telegrams can be sent, based on the configured control direction. Also in this case, there is a difference between short and long operations:

- a long operation is interpreted as a brightness control command. If the set control direction is "increase", the control will only be increasing, otherwise if the set control direction is "decrease" the control will be decreasing. In both cases, no telegram is sent when the contact is opened again.
- a short operation is transformed in to an ON or OFF command depending on the set control direction. If the set control direction is "increase" the sent command will only be an ON command. If the set control direction is "decrease" the sent command will only be an OFF command.

Unlike the ***2 push-button + stop dimmer*** function, it is possible to define both the brightness variation steps of the brightness increase/decrease commands as well as the time that must elapse between the sending of one command and another when the push-button remains pressed. The sending of the "regulation stop" telegram on push-button release is not therefore necessary, because although the regulation does follow the characteristic power/brightness curve, it is the command sent by the device that determines the percentage variation. The communication objects enabled by this function are ***IN.x -***

Switching (Data Point Type: 1.001 DPT_Switch) and ***IN.x - Brightness dimming*** (Data Point Type: 3.007 DPT _Control_Dimming).

The structure of the menu is as follows:



Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	cyclic sending 2 push button dimmer
Regulation direction	increase
Increase/decrease step	12.5%
Cyclical sending period [x 0.1s]	5

Fig. 4.7: “Cyclic sending 2 push button dimmer” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.6.1 Control direction (=Regulation direction)*

The parameter **“Regulation direction”** configures the control direction of the brightness that the channel controls; The values that can be set are:

- **Increase** **(uneven channel default value)**
- **Decrease** **(even channel default value)**

Selecting **increase**, the sent commands will be 'increase brightness 100%' or ON, depending on the recognized activation; otherwise, selecting **decrease** the sent commands will be 'decrease brightness 100%' or OFF.

➢ *4.1.4.6.2 Increase/decrease step*

The parameter**“Increase/decrease step”** is used to set the percentage value of the brightness variation associated with the brightness increase/decrease commands. In this way, as soon as a long operation is detected, the device sends the first increase/decrease command with the set percentage. The values that can be set are:

- 100%
- 50%
- 25%
- **12.5%** **(default value)**
- 6.25%
- 3.125%
- 1.56%

If the contact remains closed after recognising the long operation, the device sends the command cyclically until contact opening is detected.

➢ 4.1.4.6.3 *“Cyclical sending period [x 0.1s]”*

The **”Cyclical sending period [x 0.1s]”** parameter is used to set the time that must pass between the sending of one increase/decrease command and another, if the contact remains closed after the recognition of a long operation. When the contact is opened, no telegram is sent; the cyclical sending of the brightness control commands is merely stopped.
The values that can be set for the **”Cyclical sending period [x 0.1s]”** parameter are:

- from 3 to 50 in steps of 1 **(default value 5)**

To sum up, when a long operation is detected, the device sends the first increase/decrease command with the set percentage and, if the contact remains closed, it sends the command cyclically until it detects the opening of the contact.

EXAMPLE: if long operation minimum time is set to ***0.5 sec***, and the **Increase/decrease step** parameter is set to **12.5%** and the **Cyclical sending period [x 0.1s]** parameter is set to **3** (0.3 sec) and contact closure is detected:
– 0.5 seconds after the detection of the contact closure, a long operation is detected and so the first 12.5% brightness increase/decrease telegram is sent
– from this moment, for every 0.3 seconds that contact remains closed, the device will send the 12.5% brightness increase/decrease command again and again until opening is detected
– when the contact is opened, no telegram is sent; the cyclical sending is merely stopped

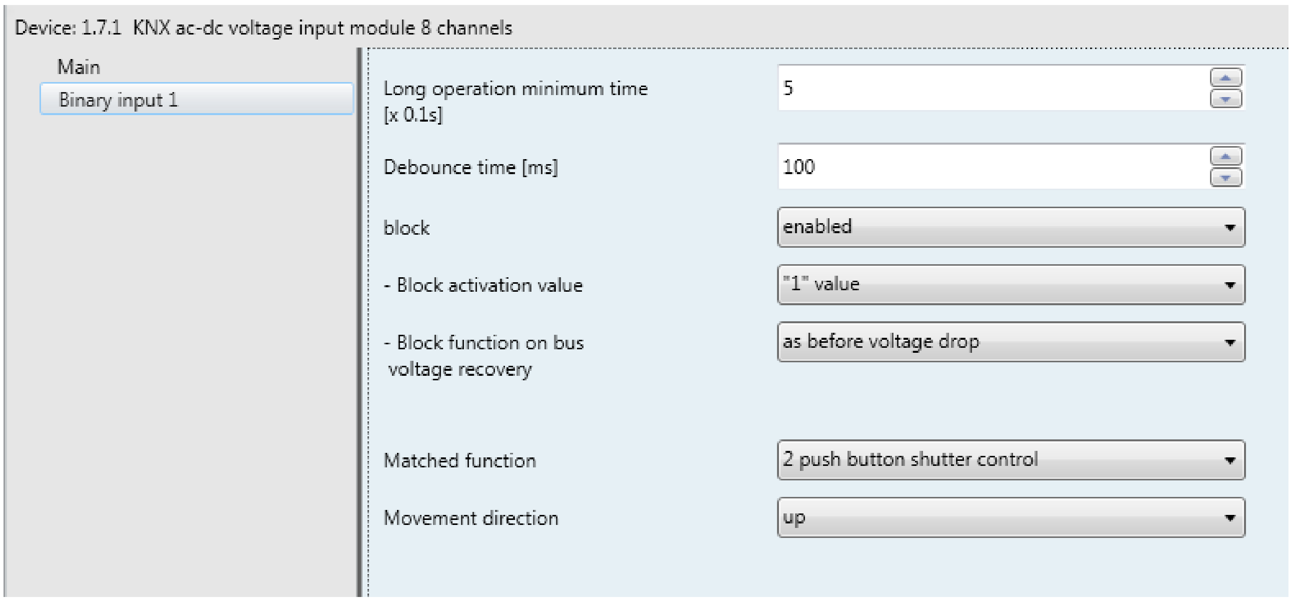
➢ 4.1.4.7 *“2 push-button shutter control” function*

This is used to configure the binary input to control a shutter/venetian blind with two buttons, managing in this case only one of the two movement directions (down or up).
Up or down movement telegrams or louvres open or close control telegrams can be sent. Also in this case, there is a difference between short and long operations:

- a long operation is transformed into a movement command. If the set movement direction is "up", the movement will only be up; vice versa if the set direction is "down" the movement will be down. When the contact reopens, the device does not perform any action.
- a short operation is transformed into a louvres control command (stop movement if the shutter is moving), depending on the set movement direction. If the set movement direction is "up" the sent command will only be a louvres opening control command (or stop movement); If the set adjustment direction is "down" the sent command will only be a louvres closing control command (or stop movement).

The communication objects enabled by this function are ***IN.x - Shutter movement*** (Data Point Type: 1.008 DPT_UpDown) and ***IN.x - Louvre stop/adjustment*** (Data Point Type: 1.007 DPT_Step).

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	2 push button shutter control
Movement direction	up

Fig. 4.8: “2 push button shutter control” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.7.1 Movement direction*

The parameter **“Movement direction”** is used to configure the direction of movement of shutter the input controls; The values that can be set are:

– **Up**	**(uneven channel default value)**
– **Down**	**(even channel default value)**

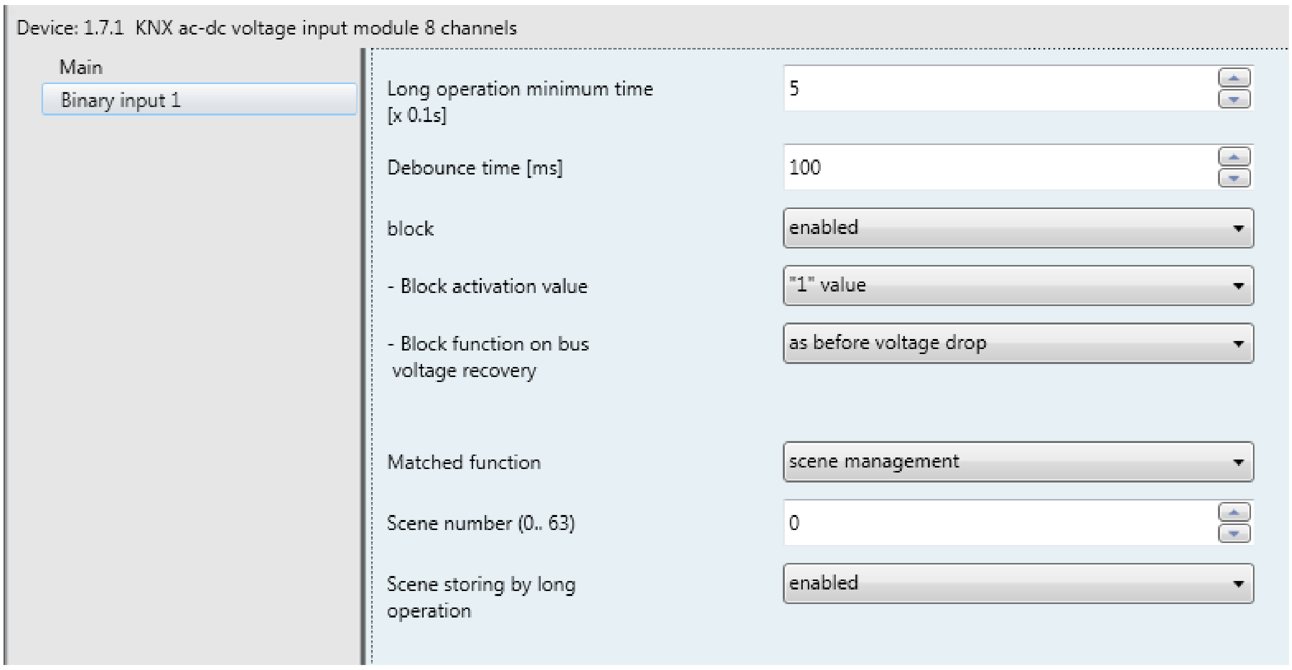
selecting **up**, the sent commands will be up movement or louvres opening control (stop movement), depending on the recognised activation; vice versa, selecting **down**, the sent commands will be down movement or louvres closing control (stop movement).

➢ *4.1.4.8 “Scene management” function*

This is used to configure the binary input to send scene memorising and execution commands, with the possibility of sending the scene memorising command following a command received from the BUS. Only one scene can be managed for each input.
There is a difference between short and long operations:
- a long operation is interpreted as a scene storing command.
- a short operation is interpreted as a scene execution command.

The communication objects enabled by this function are ***IN.x - Scene*** (Data Point Type: 18.001 DPT_SceneControl) and ***IN.x - Scene storing trigger*** (Data Point Type: 1.017 DPT_Trigger).

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	scene management
Scene number (0.. 63)	0
Scene storing by long operation	enabled

Fig. 4.9: “Scene management” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.8.1 Scene number (0..63)*

The **“Scene number (0.. 63)”** parameter is used to set the value of the scene to be recalled/stored and as a result the relative values that are sent via the ***IN.x - Scene*** object. The possible values are:

- from **0 (default value)** to 63, in steps of 1

➢ *4.1.4.8.2 Scene storing by long operation*

The parameter **“Scene storing by long operation”** enables the sending of a scene memorising command when a long operation is recognised. The values that can be set are:

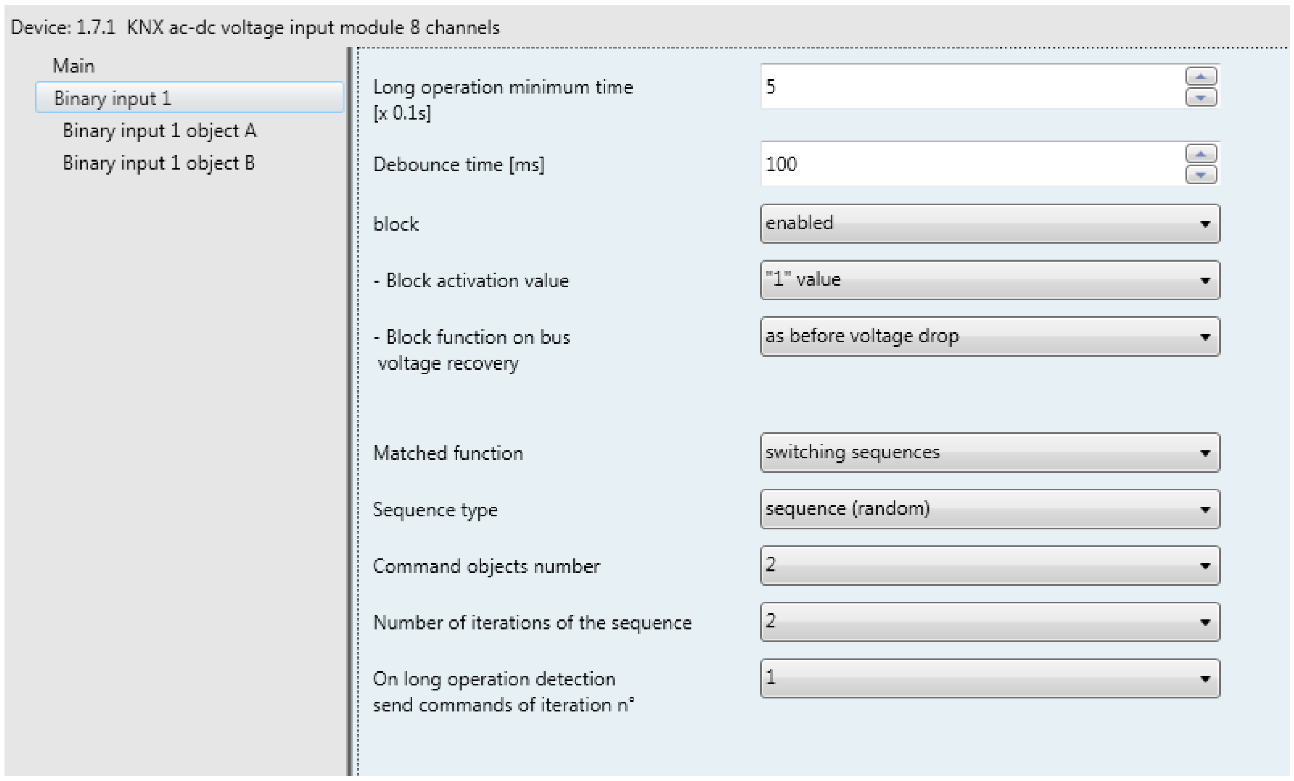
- disabled
- **enabled** **(default value)**

Only if **enabled** is selected, the device will send the scene storing command when a long operation is detected; if **disabled** is selected, a long operation is not recognised and only causes the sending of the scene execution command (like the short operation).
Independently of the value set for the above parameter, it is possible to indirectly generate the sending of the scene memorising command after receiving a bus telegram on the object ***IN.x - Scene storing trigger*** (both with value “1” as well as with value “0”); each time the device receives a telegram on that object, it must immediately send a scene memorisation telegram.

➢ *4.1.4.9 “Switching sequences” function*

Used to send a sequence of commands following the detection of a specific operation.

The structure of the menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 object A
Binary input 1 object B

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	switching sequences
Sequence type	sequence (random)
Command objects number	2
Number of iterations of the sequence	2
On long operation detection send commands of iteration n°	1

Fig. 4.10: “Switching sequences” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.9.1 Number of objects to send (Command objects number)*

The parameter **“Command objects number”** is used to set the number of commands that make up the sequence itself; Depending on the value set for this item, the ***IN.x - Sequence z*** communication objects are enabled (Data Point Type: 1.001 DPT_Switch) (with **z** included between A and D). The values that can be set are:

- from **2 (default value)** to 4, in steps of 1

➢ *4.1.4.9.2 Sequence type*

The parameter **“Sequence type”** is used to set the type of sequence to be sent. The values that can be set are:

- **sequence 1 (filling)** **(default value)**
- sequence 2 (sum)
- sequence 3 (free)

Sequence 1 (filling) consists in: each time a closure (edge) is detected, the device sends - on the enabled communication objects - a sequence that follows the filling progress. This sequence consists in activating one communication object a time, in cascade, until all the objects have the logical value “1” and in deactivating the objects in cascade until they again have the logical value “0”. Taking into consideration a sequence that includes 3 commands, at each iteration, the sent commands will be:

Edge no.	Value sent on ***IN.x – Sequence C***	Value sent on ***IN.x – Sequence B***	Value sent on ***IN.x – Sequence A***
1st edge	0	0	1
2nd edge	0	1	1
3rd edge	1	1	1
4th edge	0	1	1
5th edge	0	0	1
6th edge	0	0	0

Note: Once the 6th edge is detected, the sequence will restart from the beginning

The table shows how, considering the increasing/decreasing trend of the sequence, the most significant bit of the sequence, in this particular case, is the one for the communication object ***IN.x - C sequence*** whereas the least significant is always the one for the object ***IN.x - A sequence***.

Sequence 2 (sum) consists in: each time a closure (edge) is detected, the device sends - on the communication objects - a sequence that follows the sum progress. This sequence consists in counting the detected edges and converting this value into a binary format, distributing it on the enabled communication objects. Taking into consideration a sequence that includes 3 commands, at each iteration, the sent commands will be:

Edge no.	Value sent on ***IN.x – Sequence C***	Value sent on ***IN.x – Sequence B***	Value sent on ***IN.x – Sequence A***
1st edge	0	0	1
2nd edge	0	1	0
3rd edge	0	1	1
4th edge	1	0	0
5th edge	1	0	1
6th edge	1	1	0
7th edge	1	1	1
8th edge	0	0	0

NB: Once the 8th edge is detected, the sequence will restart from the beginning

The table shows how the trend of the sent commands depends on the count of the detected edge; in fact it starts with the binary coding of value 1 up to (in this specific case) the coding of value 7 and then the count starts again with the next edge. Also in this case, the most significant bit in the sequence is the one for the communication object ***IN.x - C sequence*** whereas the least significant is always the one for object ***IN.x - A sequence***.

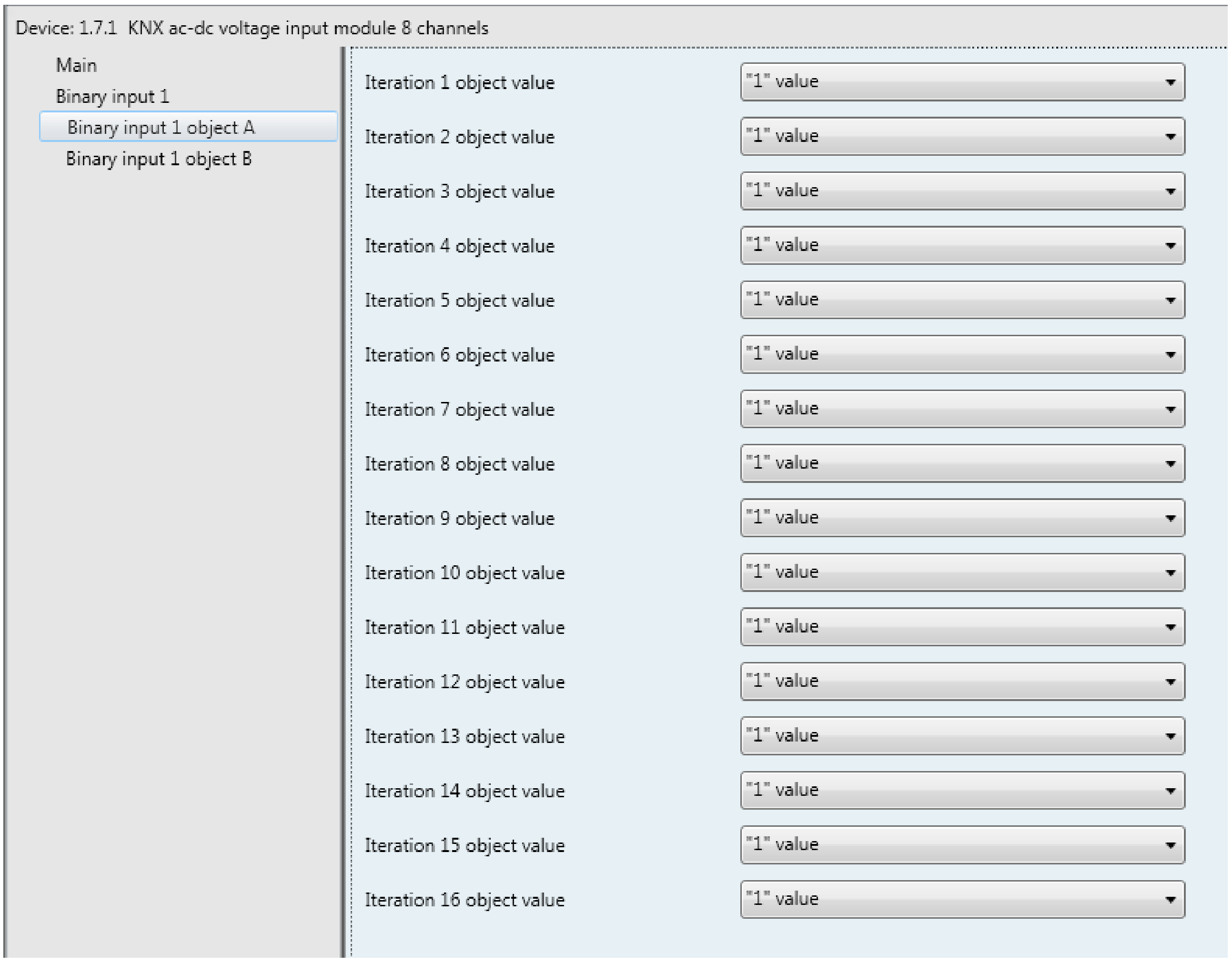
Sequence 3 (free) allows the user to directly set the value for each command for each set edge; this setting enables the parameter **"Number of sequence iterations"** and the configuration menu **z object channel x** (one for each enabled command).
The parameter **"Number of sequence iterations"** allows to set the number of iterations (edges) that make up the sequence; The values that can be set are:

- from **2 (default value)** to 16 with steps of 1

Based on the value set for this item, the **Channel x z object** menu will display or hide the parameters **"Iteration 1 object value"**, **"Iteration 2 object value"**, **"Iteration 3 object value"**, **"Iteration 4 object value"**, **"Iteration 5 object value"**, **"Iteration 6 object value"**, **"Iteration 7 object value"**, **"Iteration 8 object value"**, **"Iteration 9 object value"**, **"Iteration 10 object value"**, **"Iteration 11 object value"**, **"Iteration 12 object value"**, **"Iteration 13 object value"**, **"Iteration 14 object value"**, **"Iteration 15 object value"** and **"Iteration 16 object value"**, which can assume the following values:

- value "0"
- **value "1" (default value)**

The structure of the **Binary input x object z** menu is as follows:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

- Main
- Binary input 1
- Binary input 1 object A
- Binary input 1 object B

Iteration 1 object value	"1" value
Iteration 2 object value	"1" value
Iteration 3 object value	"1" value
Iteration 4 object value	"1" value
Iteration 5 object value	"1" value
Iteration 6 object value	"1" value
Iteration 7 object value	"1" value
Iteration 8 object value	"1" value
Iteration 9 object value	"1" value
Iteration 10 object value	"1" value
Iteration 11 object value	"1" value
Iteration 12 object value	"1" value
Iteration 13 object value	"1" value
Iteration 14 object value	"1" value
Iteration 15 object value	"1" value
Iteration 16 object value	"1" value

Fig. 4.11: “Binary input x object z” menu

Regardless of the type of sequence selected, the “**On long operation detection, send commands of iteration n°**” parameter is used to define which sequence iteration to send if a long operation is detected; The values that can be set are:

- from 1 to 16 in steps of 1 **(default value 1)**

EXAMPLE: with reference to the above tables, let's suppose that the value set by the user is **3**. When a long operation is detected, the device will send:

Edge no.	Value sent on ***IN.x – Sequence C***	Value sent on ***IN.x – Sequence B***	Value sent on ***IN.x – Sequence A***
1st edge	0	0	1
2nd edge	0	1	1
3rd edge	1	1	1
4th edge	0	1	1
5th edge	0	0	1
6th edge	0	0	0

"Filling" sequence

Edge no.	Value sent on ***IN.x* – *Sequence C***	Value sent on ***IN.x* – *Sequence B***	Value sent on ***IN.x* – *Sequence A***
1st edge	0	0	1
2nd edge	0	1	0
3rd edge	0	1	1
4th edge	1	0	0
5th edge	1	0	1
6th edge	1	1	0
7th edge	1	1	1
8th edge	0	0	0

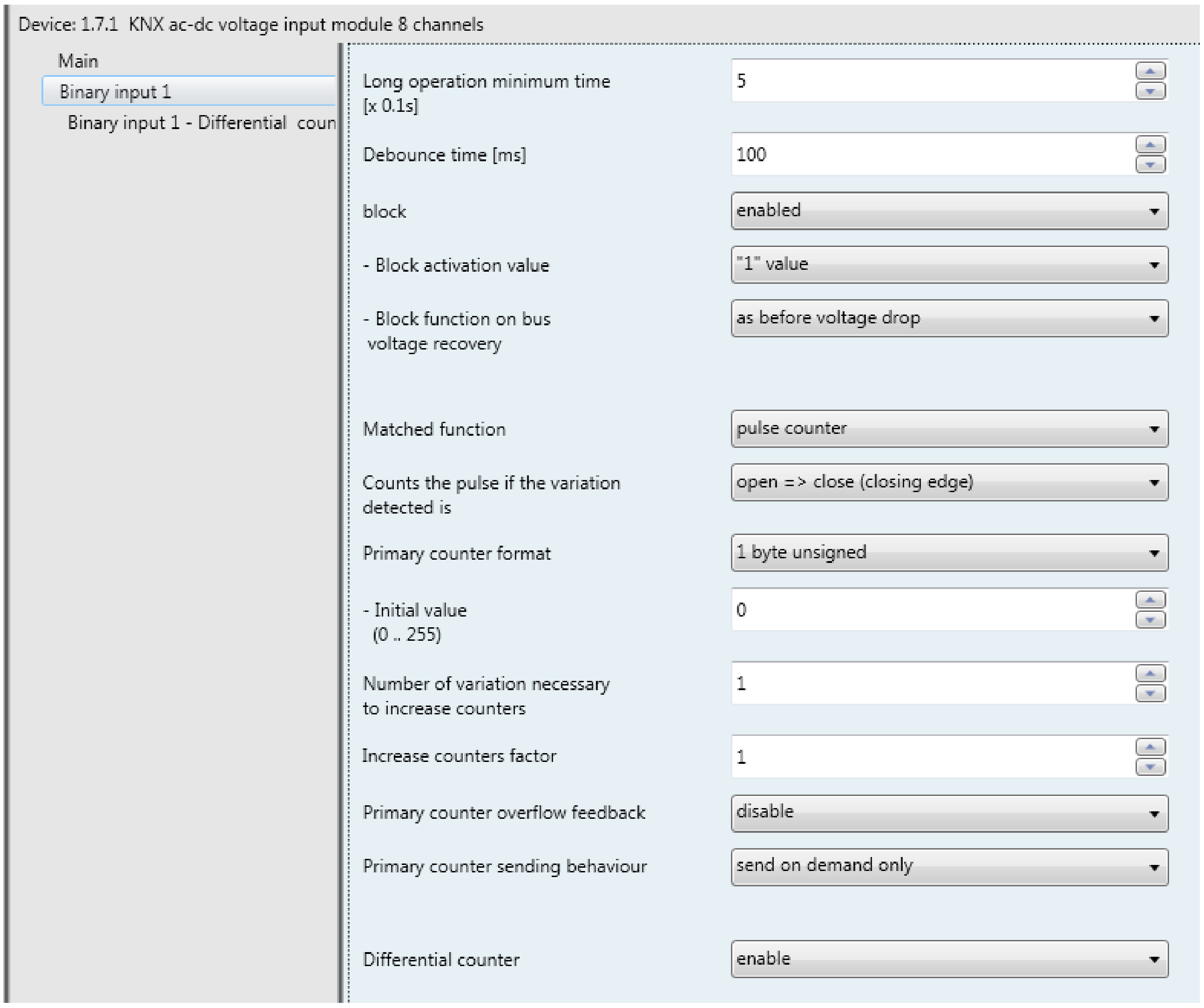
"Sum" sequence

Once a long operation has been detected and the sequence relating to the set iteration has been sent, then when the next short operation is detected, the sequence relating to the iteration immediately after the one associated with the long operation will be sent (in the example given here, the sequence associated with iteration no. 4 will be sent).
To sum up, the value set for the "**On long operation detection**, **send commands of iteration n°**" parameter defines both the sequence to be sent and the value with which to initialise the iterations counter when a long operation is detected.
Make sure the selected iteration number associated with the sequence to be sent with a long operation is less than - or equal to - the maximum number of iterations associated with the sequence; otherwise, the iteration to be taken into consideration is the maximum one.

➢ 4.1.4.10 ”Pulse counter” function

Used to configure the channel for counting the number of contact status variations (edges) by setting the parameters that characterise the count.

The structure of the menu is as follows:



Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Differential coun

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	pulse counter
Counts the pulse if the variation detected is	open => close (closing edge)
Primary counter format	1 byte unsigned
- Initial value (0 .. 255)	0
Number of variation necessary to increase counters	1
Increase counters factor	1
Primary counter overflow feedback	disable
Primary counter sending behaviour	send on demand only
Differential counter	enable

Fig. 4.12: ”Pulse counter” matched function menu

Enabling this function makes the following parameters available for configuration:

➢ 4.1.4.10.1 Counts the pulse if the variation detected is

In this mode, each contact can count the incoming pulses. The count is based on the detection of the edges of the input signal. There are 2 edges that can be detected: contact closure and opening. The **“Counts the pulse if the variation detected is”** parameter is used to set the type of contact status variation to be considered for increasing the count of the primary and differential counters. The values that can be set are:

- **open => close (closing edge)** **(default value)**
- close => open (opening edge)
- both

By selecting **open => close (closing edge)**, only the variation from open contact to closed contact (closing edge) will be considered by the device as a pulse, so it is this variation that produces an increase in the count value; the opposite status variation will have no effect.

By selecting **close => open (opening edge)**, only the variation from closed contact to open contact (opening edge) will be considered by the device as a pulse, so it is this variation that produces an increase in the count value; the opposite status variation will have no effect.
By selecting **both**, the variation from closed contact to open contact (opening edge) and the variation from open contact to closed contact (closing edge) will both be considered by the device as a pulse, producing an increase in the count value.

➢ *4.1.4.10.2 Primary counter format*

The primary counter used for the pulse count must be of a sufficient capacity to count the maximum required number of pulses. With the **"Primary counter format"** parameter, you can define the size and code of the communication object used to communicate the value of the primary counter. The values that can be set are:

- **1 byte value without sign** **(default value)**
- 1 byte signed value
- 2 bytes unsigned value
- 2 bytes signed value
- 4 bytes unsigned value
- 4 bytes signed value

Depending on the value set for this item, the values that can be set for the **"Initial value"** parameter will be different.
The parameter **"Initial value"** is used to set the initial value of the primary counter; When the primary counter reaches its overflow - or maximum value - point (or minimum value, depending on the counter increase factor set), it is re-initialised to the set initial value.
Depending on the value set for the **Primary counter format** parameter, the values that can be set for this item will be different.

- If the format of the primary counter is **1 byte unsigned**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the primary counter is **1 byte signed**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the primary counter is **2 byte unsigned**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the primary counter is **2 byte signed**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the primary counter is **4 byte unsigned**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the primary counter is **4 byte signed**, the ***IN.x - Primary counter*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

  - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

➢ *4.1.4.10.3 Number of variations necessary to increase counters*

The parameter **“Number of variations necessary to increase counters”** is used to set the number of edges necessary to increase the counters (both primary and differential). This means that, if a value of 2 is set (for example), two edges are needed to increase the value of the counters (both primary and differential). The values that can be set are:

- from **1 (default value)** to 32767 with step of 1

➢ *4.1.4.10.4 Increase counters factor*

The parameter **“Increase counters factor”** is used to establish by how many units the counters (both primary and differential) must increase when counter increase conditions occur (number of edges detected equal to the number of variations needed for a counter increase). This means that, if a value of 2 is set (for example), the counters (both primary and differential) will be increased by two units every time increase conditions occur.

- from - 32768 to +32767 in steps of 1 **(default value 1)**

if a negative value is selected, the counters are decreased and the overflow value of the primary counter is the minimum value of the range defined by the selected format.

To better understand the meaning of the **“Number of variations necessary to increase counters”** and **“Increase counters factor”** parameters, let's consider the case where the increase factor is 2 and the number of variations necessary to increase the counter is 5. With this configuration, the value of the counters (both primary and differential) will be increased by two units for every five count increase edges detected. Of course, the counter value is not modified until 5 increase edges are detected.

➢ *4.1.4.10.5 Primary counter overflow feedback*

The parameter **“Primary counter overflow feedback”** is used to enable the display - and hence the use - of the communication objects that indicate when the primary counter has exceeded its maximum (or minimum) value. The values that can be set are:

- **disable** **(default value)**
- enable object of 1 bit
- enable objects of 1 bit and 1 byte

Selecting a value other than **disable** displays the communication object ***IN.x – Primary counter overflow bit*** (Data Point Type: 1.002 DPT_Bool) via which the device indicates the overflow of the primary counter. When the overflow occurs, a value of “1” is sent; a value of “0” is never sent.

Selecting **enable objects 1 bit and 1 byte** displays the communication object ***IN.x - Primary counter overflow byte*** (Data Point Type: 5.010 DPT_Value_1_Ucount) via which the device indicates the overflow of the primary counter. When the overflow occurs, the value defined by the new **"Send the value with primary counter overflow"** parameter is sent. This parameter may assume the following values:

- from **0 (default value)** to 255, in steps of 1

Once the maximum (or minimum) value has been reached, the primary counter restarts from the value set in **"Initial value"**.
If the value set in **"Increase counters factor"** is greater than 1, the number of units needed to trigger the overflow may be less than the increase factor; as the primary counter is circular, it is re-initialised when the overflow value is exceeded and the supplementary units are calculated.

Example: increase counters factor of *7*, the counter is *1 unsigned byte* and the initial value is 50. If the counter value is 253 and the counter increase condition is detected, the overflow telegram is sent and the new counter value is 54 (the initial value is also counted).

➢ *4.1.4.10.6 Primary counter sending behaviour*

This parameter **"Primary counter sending behaviour"** is used to define the conditions for sending the current value of the primary counter. The values that can be set are:

- send on demand only
- **send in case of change** **(default value)**
- send periodically
- send on change and periodically

Selecting a value other than **send on demand only** displays the communication object ***IN.x - Primary counter sending trigger*** (Data Point Type: 1.017 DPT_Trigger). Selecting **send in case of change** or **send on change and periodically**, the **"Minimum primary counter variation for sending value"** parameter will be visible, whereas by selecting **send periodically** or **send on change and periodically** the **"Primary counter sending period (seconds)"** parameter will be visible.

Selecting the value **send on demand only**, no new parameter will be enabled because the primary counter value is not sent spontaneously by the device; only in the case of a status read request will it send the user a telegram in response to the command received, giving information about the current value of the primary counter.

If the primary counter sending condition is different from **on demand only**, there is the possibility of indirectly generating the sending of the current counter value following receipt of a BUS telegram on the object ***IN.x - Primary counter sending trigger*** (both with value "1" as well as with value "0"); Every time the device receives a telegram on that object, it must immediately send the current value of the primary counter. After a BUS voltage recovery, the value of the primary counter should be sent in order to update any connected devices.

The parameter **"Minimum primary counter variation for sending value"** is visible if the primary counter value is sent with a change. It is used to define the minimum count variation (in relation to the last value sent) that causes the new measured value to be spontaneously sent; The values that can be set are:

- from 1 to 100 with steps of 1, **10 (default value)**

The parameter **"Primary counter sending period (seconds)"** is visible if the primary counter value is sent periodically. It is used to define the period with which telegrams indicating the current primary counter value are spontaneously sent; The values that can be set are:

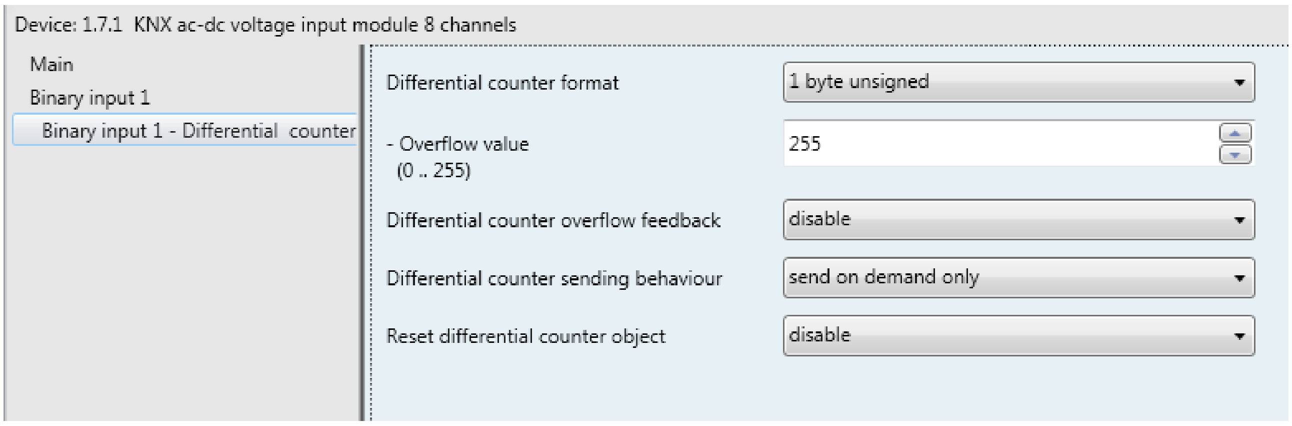
- from 1 to 255 with steps of 1, **15 (default value)**

In the event of a BUS voltage failure, the primary counter value must be saved in a non-volatile memory and restored when the BUS voltage is recovered.

4.1.4.10.7 Differential counter

The parameter **“Differential counter”** is used to enable the display - and hence the use - of the communication object ***IN.x - Differential counter*** and display the configuration menu ***Binary input x - Differential counter*** (see figure below).
Unlike the primary counter, the differential counter: can be reset, can indicate an overflow value different from the maximum coded value, and has an initial value of 0. The two counters both have: a counter increase edge, an increase factor, and a number of variations for counter increase.
The values that can be set are:

- **disable (default value)**
- enable


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Differential counter

Differential counter format: 1 byte unsigned
- Overflow value (0 .. 255): 255
Differential counter overflow feedback: disable
Differential counter sending behaviour: send on demand only
Reset differential counter object: disable

Fig. 4.13: “Binary input x – Differential counter” menu

The differential counter used for the pulse count must be of a sufficient capacity to count the maximum required number of pulses. With the **“Differential counter format”** parameter, it is possible to define the size and code of the communication object used to communicate the value of the primary counter. The values that can be set are:

- **1 byte value without sign (default value)**
- 1 byte signed value
- 2 bytes unsigned value
- 2 bytes signed value
- 4 bytes unsigned value
- 4 bytes signed value

The initial value is always 0, regardless of the format selected.
Depending on the value set for this item, the values that can be set for the **“Overflow value”** parameter will be different.

The **“Overflow value”** parameter is used to set the maximum value of the differential counter; in fact, unlike the primary counter, it is possible to set the maximum count value - i.e. the value beyond which the differential counter is in an overflow condition.

Depending on the value set for the **Differential counter format** parameter, the values that can be set for this item will be different.

- If the format of the differential counter is **1 byte unsigned**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

  - from 0 to **255 (default value)** with steps of 1

- If the format of the differential counter is **1 byte signed**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to **127 (default value)** with steps of 1

- If the format of the differential counter is **2 byte unsigned**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from 0 to **65535 (default value)** with steps of 1

- If the format of the differential counter is **2 byte signed**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from-32768 to **+32767 (default value)** with steps of 1

- If the format of the differential counter is **4 byte unsigned**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

  - from 0 to **4294967295 (default value)** with steps of 1

- If the format of the differential counter is **4 byte signed**, the ***IN.x - Differential counter*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

  - from -2147483648 to **2147483647 (default value)** with steps of 1

The parameter **“Differential counter overflow feedback”** is used to enable the display - and hence the use - of the communication objects that indicate when the differential counter has exceeded its maximum (or minimum) value. The values that can be set are:

- **disable** **(default value)**
- enable object of 1 bit
- enable objects of 1 bit and 1 byte

Selecting a value other than **disable** displays the communication object ***IN.x – Differential counter overflow bit*** (Data Point Type: 1.002 DPT_Bool) via which the device indicates the overflow of the differential counter. When the overflow occurs, a value of “1” is sent; a value of “0” is never sent.
Selecting **enable objects 1 bit and 1 byte** displays the communication object ***IN.x - Differential counter overflow byte*** (Data Point Type: 5.010 DPT_Value_1_Ucount) via which the device indicates the overflow of the differential counter. When the overflow occurs, the value defined by the new **“Send the value with differential counter overflow”** parameter is sent. This parameter may assume the following values:

- from **0 (default value)** to 255, in steps of 1

Once the maximum value has been reached, the differential counter restarts from 0.
If the value set in **“Increase counters factor”** of the **Channel x** menu is greater than 1, the number of units needed to trigger the overflow may be less than the increase factor; as the differential counter is circular, it is re-initialised when the overflow value is exceeded and the supplementary units are calculated. Example:

increase counters factor of *7* and the counter is *1 byte unsigned*. If the differential counter value is 253 and the counter increase condition is detected, the overflow telegram is sent and the new counter value is 4 (the initial value is also counted).

The parameter **"Differential counter sending behaviour"** is used to define the conditions for sending the current value of the differential counter. The values that can be set are:

- **send on demand only** **(default value)**
- send on change
- send periodically
- send on change and periodically

Selecting a value other than **send on demand only** displays the communication object ***IN.x - Differential counter sending trigger*** (Data Point Type: 1.017 DPT_Trigger). Selecting **send in case of change** or **send on change and periodically**, the **"Minimum differential counter variation for sending value"** parameter will be visible, whereas by selecting **send periodically** or **send on change and periodically** the **"Differential counter sending period (seconds)"** parameter will be visible.
Selecting the value **send on demand only**, no new parameter will be enabled because the differential counter value is not sent spontaneously by the device; only in the case of a status read request will it send the user a telegram in response to the command received, giving information about the current value of the differential counter.
If the differential counter sending condition is different from **on demand only**, there is the possibility of indirectly generating the sending of the current counter value following receipt of a BUS telegram on the object ***IN.x - Differential counter sending trigger*** (both with value "1" as well as with value "0"); Every time the device receives a telegram on that object, it must immediately send the current value of the differential counter. After a BUS voltage recovery, the value of the differential counter should be sent in order to update any connected devices.

This parameter is visible if the differential counter value is sent with a change. It is used to define the minimum count variation (in relation to the last value sent) that causes the new measured value to be spontaneously sent. The values that can be set are:

- from 1 to 100 with steps of 1, **10 (default value)**

This parameter is visible if the differential counter value is sent periodically. It is used to define the period with which telegrams indicating the current differential counter value are spontaneously sent. The values that can be set are:

- from 1 to 255 with steps of 1, **15 (default value)**

The parameter **"Differential counter reset object"** is used to enable the display - and hence the use - of the communication object ***IN.x - Differential counter reset*** (Data Point Type: 1.017 DPT_Trigger), to receive - via BUS - the differential counter reset command for resetting the value of the differential counter. The values that can be set are:

- **disable** **(default value)**
- enable

Selecting **enable**, the ***IN.x - Differential counter reset*** communication object is made visible, via which the device receives the differential counter reset command. If a value of "1" or "0" is received, the differential counter is re-initialised at 0.

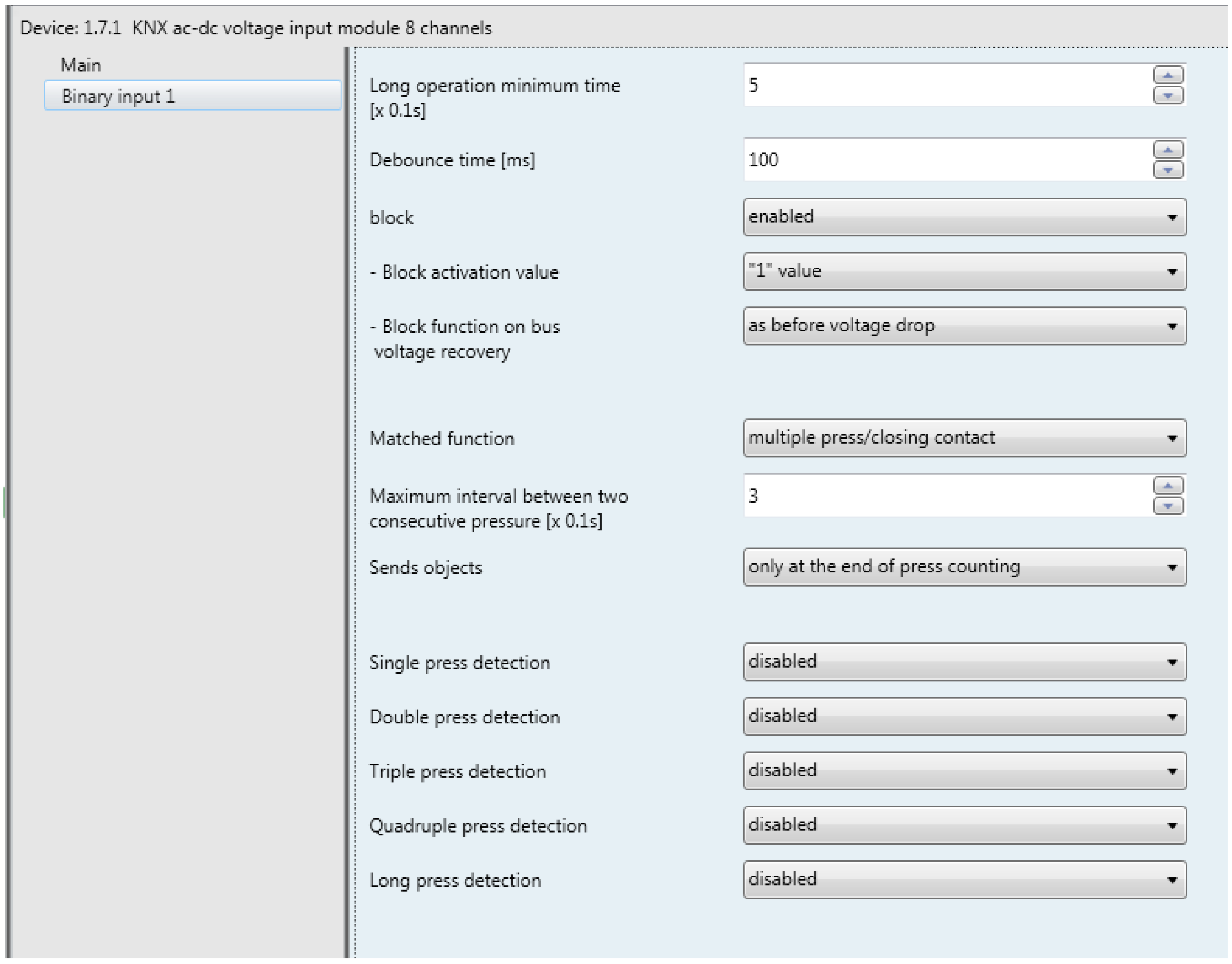
In the event of a BUS voltage failure, the differential counter value must be saved in a non-volatile memory and restored when the BUS voltage is recovered.

➢ *4.1.4.11 “Multiple press/closing contact” function*

This function is used to set the type and number of commands to send after a series of consecutive pressing operations has been detected, for up to four commands per binary input.

The structure of the menu is as follows:

-


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Long operation minimum time [x 0.1s]	5
Debounce time [ms]	100
block	enabled
- Block activation value	"1" value
- Block function on bus voltage recovery	as before voltage drop
Matched function	multiple press/closing contact
Maximum interval between two consecutive pressure [x 0.1s]	3
Sends objects	only at the end of press counting
Single press detection	disabled
Double press detection	disabled
Triple press detection	disabled
Quadruple press detection	disabled
Long press detection	disabled

Fig. 4.14: "Multiple press/closing contact" function menu

In this mode, every input can send a series of KNX telegrams following the detection of several consecutive contact pressing operations; a pressing is recognised when the contact re-opens after a closure (open→closed→open). In particular, the device is able to distinguish the following consecutive pressings:

- single press → one pressing of the push-button
- double press → two consecutive pressings of the push-button
- triple press → three consecutive pressings of the push-button
- quadruple press → four consecutive pressings of the push-button
- long press → long contact closure

Five consecutive presses or more are interpreted as a “quadruple press”.
In order to recognise two consecutive presses, it is necessary to define the maximum gap between the detection of one press and the next; if the time between two presses (not counting the debounce time) is less than the maximum gap, the count of multiple presses is increased. When the time that elapses after the detection of a pressing (not counting the debounce time) exceeds the maximum gap, the device recognises

a number of consecutive multiple presses equal to the value counted and, after sending the telegrams associated with this action, it resets their counter.

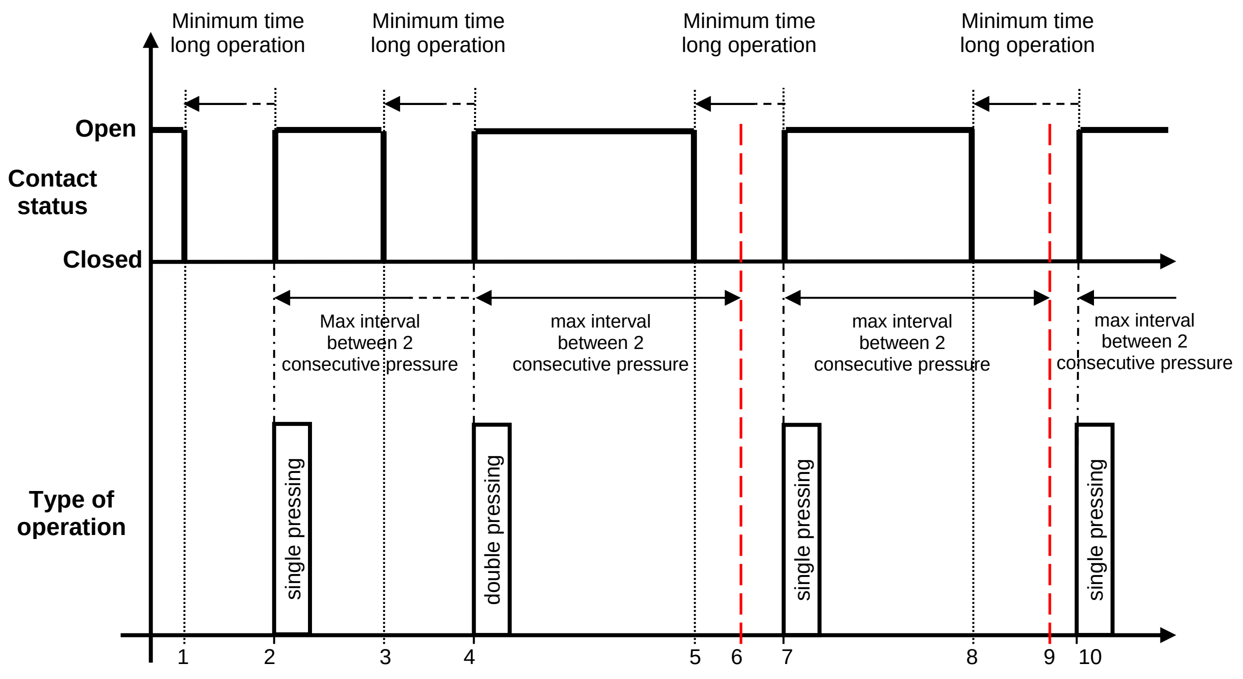
Enabling this function makes the following parameters available for configuration:

➢ *4.1.4.11.1 Maximum interval between two consecutive pressure [x 0.1s]*

The parameter **"Maximum interval between two consecutive pressure [x 0.1s]"** is used to define the maximum gap between the detection of one press and the next, so that they are recognised as consecutive presses. The values that can be set are:

- from **3 (default value)** to 100 seconds, with steps of 1.

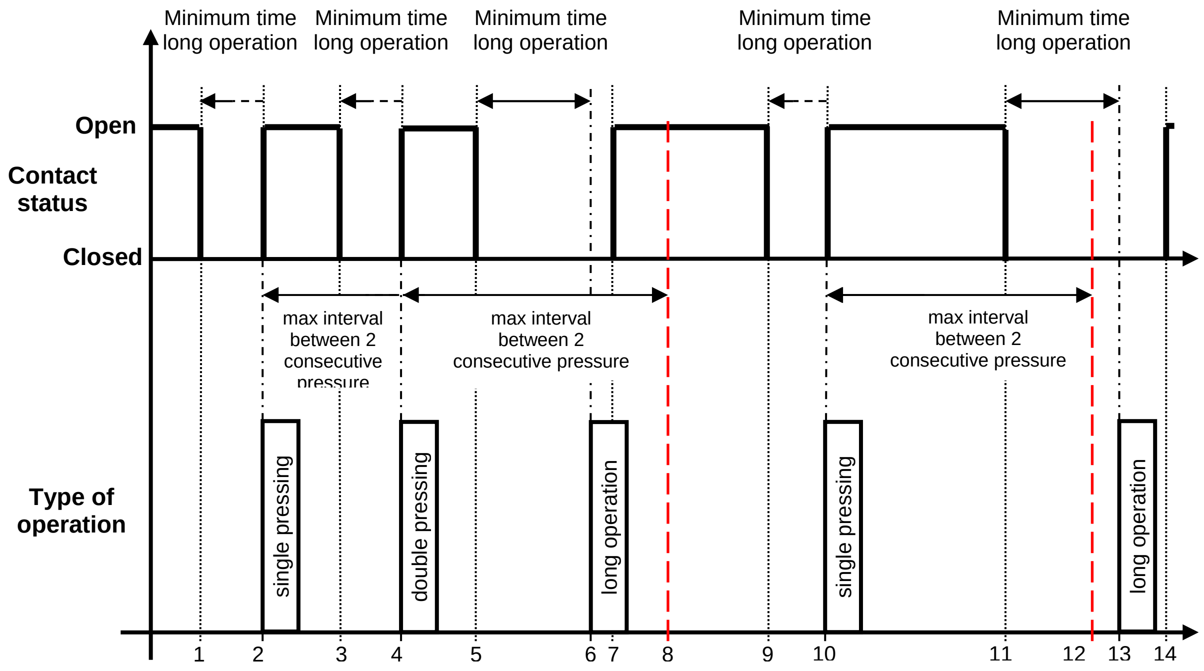
The following chart shows some situations that summarise the concept of multiple presses (the debounce time is not shown).


Minimum time long operation
Minimum time long operation
Minimum time long operation
Minimum time long operation
Open
Contact status
Closed
Max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
Type of operation
single pressing
double pressing
single pressing
single pressing
1
2
3
4
5
6
7
8
9
10

1. Once the closure of the contact has been detected, the contact closure time is calculated in order to distinguish a short press from a long one.
2. When the re-opening of the contact is detected before the long operation time, a short press is recognised and the count of the gap between two consecutive presses is started. The multiple press count is increased.
3. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
4. The re-opening of the contact before the long operation time and before reaching the maximum gap between two consecutive presses means the detection of a new short press that increases the multiple press count and re-initialises the calculation of the gap between two consecutive presses.
5. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
6. Once the maximum gap between two consecutive presses (dotted red line) has elapsed, the multiple press count is terminated and, after sending the KNX commands relating to this action, the counter is reset.

7. The re-opening of the contact before the long operation time means the detection of a new short press that increases the multiple press count and initialises the count of the gap between two consecutive presses.
8. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
9. Once the maximum gap between two consecutive presses (dotted red line) has elapsed, the multiple press count is terminated and, after sending the KNX commands relating to this action, the counter is reset.
10. The re-opening of the contact before the long operation time means the detection of a new short press that increases the multiple press count and initialises the count of the gap between two consecutive presses.

The detection of a long press in no way alters the multiple press count or any calculation of the gap between two consecutive presses, even if the minimum duration of the long operation is less than the maximum gap between two consecutive presses. See below (the debounce time is not shown).


Minimum time long operation
Minimum time long operation
Minimum time long operation
Minimum time long operation
Minimum time long operation
Open
Contact status
Closed
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
Type of operation
single pressing
double pressing
long operation
single pressing
long operation
1
2
3
4
5
6
7
8
9
10
11
12
13
14

1. Once the closure of the contact has been detected, the contact closure time is calculated in order to distinguish a short press from a long one.
2. When the re-opening of the contact is detected before the long operation time, a short press is recognised and the count of the gap between two consecutive presses is started. The multiple press count is increased.
3. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
4. The re-opening of the contact before the long operation time and before reaching the maximum gap between two consecutive presses means the detection of a new short press that increases the multiple press count and re-initialises the calculation of the gap between two consecutive presses.
5. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
6. If the contact remains closed for a time greater than the minimum duration of a long operation, a long press is recognised and the KNX commands for that action are sent, but neither calculation of the gap between two consecutive presses nor the multiple press count is modified in any way.
7. The re-opening of the contact following the recognition of a long press does not lead to any action.
8. Once the maximum gap between two consecutive presses (dotted red line) has elapsed, the multiple press count is terminated and, after sending the KNX commands relating to this action, the counter is reset.

9. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
10. The re-opening of the contact before the long operation time means the detection of a new short press that increases the multiple press count and initialises the count of the gap between two consecutive presses.
11. A new contact closure leads to the initialisation of the contact closure time count (to distinguish a short press from a long one), but this does not modify in any way the calculation of the gap between two consecutive presses and the multiple press count.
12. Once the maximum gap between two consecutive presses (dotted red line) has elapsed, the multiple press count is terminated and, after sending the KNX commands relating to this action, the counter is reset.
13. If the contact remains closed for a time greater than the minimum duration of a long operation, a long press is recognised and the KNX commands for that action are sent, but neither calculation of the gap between two consecutive presses nor the multiple press count is modified in any way.
14. The re-opening of the contact following the recognition of a long press does not lead to any action.

➢ *4.1.4.11.2 Send objects*

The commands associated with the “multiple press” function can be sent in two different ways:

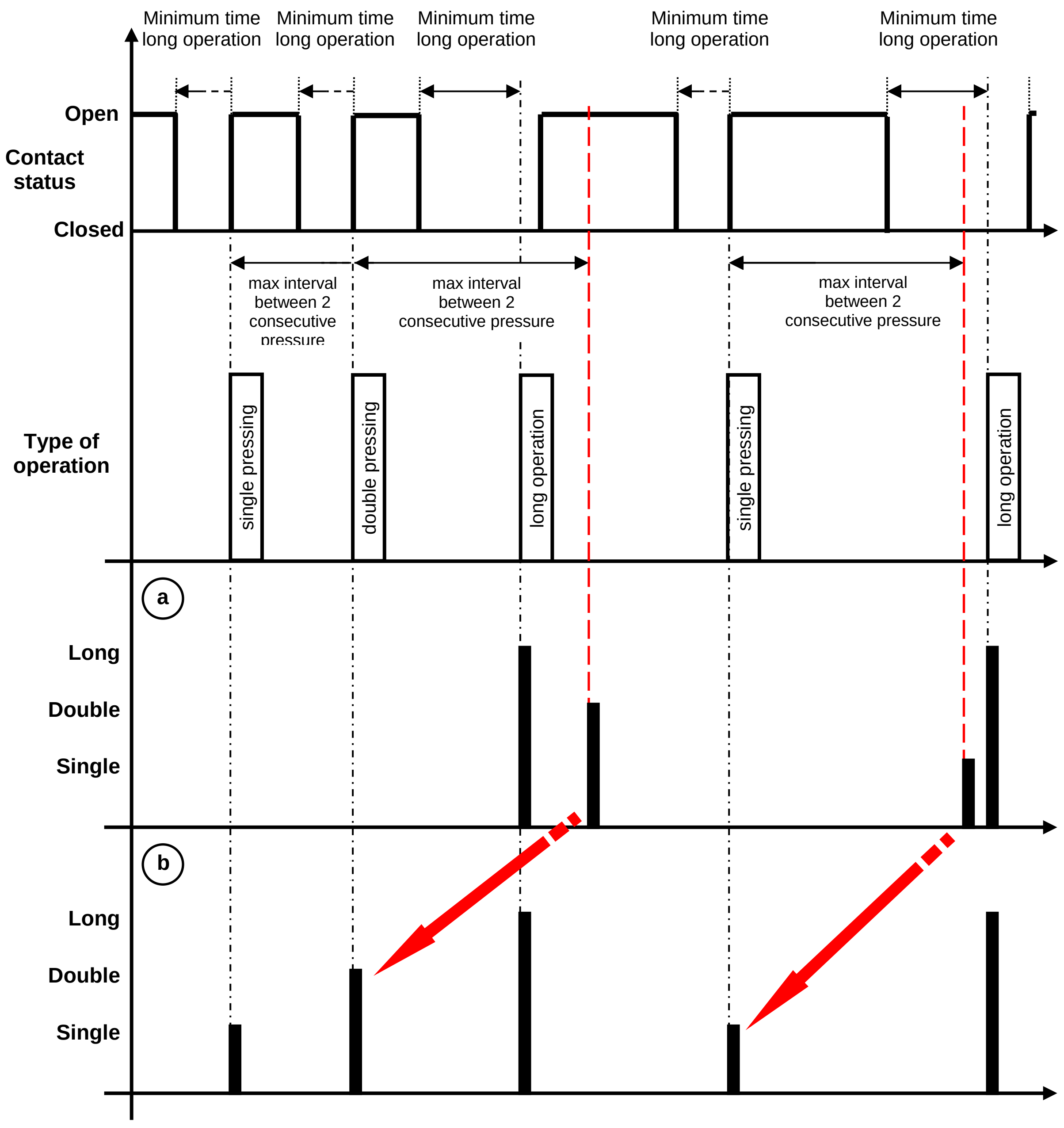
1. the device waits for the gap between two consecutive presses to exceed the maximum value, consequently interrupting the multiple press count and sending the commands associated with the number of presses detected
2. every time the multiple press count is increased, the device sends the telegrams associated with the number of presses detected

The commands associated with a “long press” are always sent as soon as the long press is detected.
The **“Send objects”** parameter is used to define the sending conditions of the objects associated with multiple presses. The values that can be set are:

- with every press detected
- **only at the end of the press count** **(default value)**

Selecting **only at the end of the press count**, the device behaves as described in point “a”. Selecting **with every press detected**, the device behaves as described in point “b”.

The following chart summarises the behaviour of the device on the basis of the set sending condition.


Minimum time long operation
Minimum time long operation
Minimum time long operation
Minimum time long operation
Minimum time long operation
Open
Contact status
Closed
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
max interval between 2 consecutive pressure
Type of operation
single pressing
double pressing
long operation
single pressing
long operation
a
Long
Double
Single
b
Long
Double
Single

The chart resumes the situation shown previously, introducing the long press and its effect on counters and timers. The two sections at the bottom show the commands sent on the KNX BUS if the sending is **only at the end of the press count** (case “a”) or **with every press detected** (case “b”). The main difference between the two cases is that in case “b”, every time a multiple press is counted, the associated telegrams are sent, while in case “a” it is necessary to wait until the time between two consecutive presses exceeds the maximum value in order to end the multiple press count, and the telegrams sent are only those associated with the last press detected.
The red arrows highlight the differences between the moments when the telegrams associated with the same long presses are actually sent.

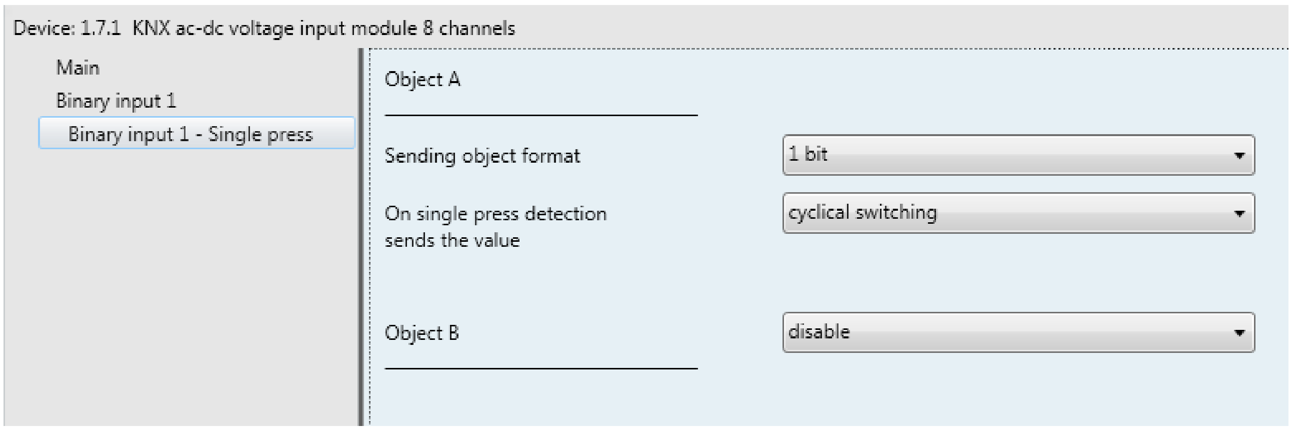
➢ 4.1.4.11.3 Single press detection

The parameter **"Single press detection"** is used to enable the recognition of a single press, and to visualise the **Binary input x - Single press** menu for enabling and configuring the commands that will be sent following the recognition of a single press.

The values that can be set are:

- disabled
- **enabled** **(default value)**

selecting **enabled**, displays the menu **Binary input x - Single press** as shown in the following figure:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Single press

Object A

Sending object format: 1 bit
On single press detection sends the value: cyclical switching

Object B: disable

Fig. 4.15: "Binary input x – Single press" menu

Upon detection of the single press, it is possible to send up to 4 different objects (which are distinguished by the letters A, B, C and D); object A is always enabled, whereas the parameter **"Object z"** (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

selecting **enable** displays the parameters **"Sending object format"** and **"On single press detection sends the value"** grouped in the subgroup **Object z** (where z indicates the object associated with the binary input, included between **A** and **D)**.

The parameter **"Sending object format"** makes it possible to set the format and code of the object "z" of input "x" that is sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit
- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value
- 3 bytes RGB colour
- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes

Depending on the value set for this item, the values that can be set for the **“On single press detection sends the value”** parameter will be different.
The **“On single press detection sends the value”** parameter is used to set the command or value to send following the detection of a single press (on the basis of the set sending conditions) associated with the binary input. The values that can be set are:

- If the format of the object to send is **1 bit**, the ***IN.x - Single press 1 bit z object*** communication object will be visible (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the above parameter are:

    - 0
    - 1
    - **cyclical switching (default value)**

    selecting **cyclical switching**, the command that the device will send (via the object ***IN.x - Single press 1 bit z object***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Single press 1 bit z object***).

- If the format of the object to send is **2 bit**, the ***IN.x - Single press 2 bit z object*** communication object will be visible (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the above parameter are:

    - forcing active on (down)
    - activate OFF forcing (up)
    - deactivate forcing [=forcing deactivation]
    - cyclical switching - ON forcing /OFF forcing
    - **cyclical switching - forcing ON / deactivate forcing (default value)**
    - cyclical switching - OFF forcing/forcing deactivation

    selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Single press 2 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Single press 2 bit z object z***).

- If the format of the object to send is **1 byte value unsigned**, the ***IN.x - Single press 1 byte z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

    - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte value signed**, the ***IN.x - Single press 1 byte z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

    - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - Single press 1 byte z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the above parameter are:

    - from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - Single press 1 byte z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the above parameter are:

    - auto mode
    - comfort mode
    - pre-comfort mode
    - economy mode

- off mode (building protection)
- **cyclical switching (thermostat)** **(default value)**
- cyclical switching (timed thermostat)

By selecting the value **cyclic switching (thermostat)**, each time the associated event is detected (single press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Comfort* …;
By selecting the value **cyclic switching (timed thermostat),** each time the associated event is detected (pressing/releasing) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Auto→ Comfort* …

- If the format of the object to send is **2 byte value unsigned**, the ***IN.x - Single press 2 byte z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte value signed**, the ***IN.x - Single press 2 byte z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the **"On single press detection sends the value"** parameter is a dummy one, used to select the colour to be sent. The real value, downloaded from the memory, will depend on the three parameters that represent the colour components (see below). In addition, the ***IN.x - Single press 3 byte z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the above parameter are:

  - **white (default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

  By selecting **customise**, the following parameters are made visible: **"Value of RED component (0 .. 255)"**, **"Value of GREEN component (0 .. 255)"** and **"Value of BLUE component (0 .. 255)"**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
  The values that can be set are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte value unsigned**, the ***IN.x - Single press 4 byte z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte value signed**, the ***IN.x - Single press 4 byte z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

  - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

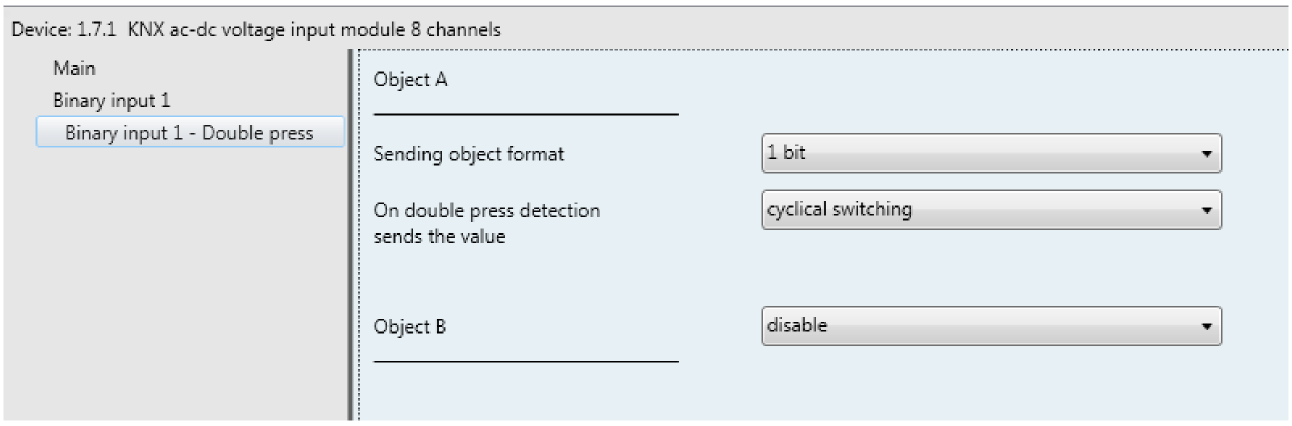
NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with "unsigned and signed value" coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format "2/4 byte signed/unsigned value".

➢ *4.1.4.11.4 Double press detection*

The parameter **"Double press detection"** is used to enable the recognition of a single press, and to visualise the **Binary input x - Double press** menu for enabling and configuring the commands that will be sent following the recognition of a double press.
The values that can be set are:

- disabled
- **enabled** **(default value)**

selecting **enabled**, displays the menu **Binary input x - Double press** as shown in the following figure:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Double press

Object A

Sending object format: 1 bit

On double press detection sends the value: cyclical switching

Object B: disable

Fig. 4.16: "Binary input x – Double press" menu

Upon detection of the double press, it is possible to send up to 4 different objects (which are distinguished by the letters A, B, C and D); object A is always enabled, whereas the parameter **"Object z"** (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

selecting **enabled** displays the parameters **"Sending object format"** and **"On double press detection sends the value"** grouped in the subgroup **Object z** (where z indicates the object associated with the binary input, included between **A** and **D)**.

The parameter **"Sending object format"** makes it possible to set the format and code of the object "z" of input "x" that is sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit
- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value

- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes
- 3 bytes RGB colour

Depending on the value set for this item, the values that can be set for the **“On double press detection sends the value”** parameter will be different.
The **“On double press detection sends the value”** parameter is used to set the command or value to send following the detection of a double press (on the basis of the set sending conditions) associated with the binary input. The values that can be set are:

- If the format of the object to send is **1 bit**, the ***IN.x - Double press 1 bit z object*** communication object will be visible (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the above parameter are:

  - 0
  - 1
  - **cyclical switching (default value)**

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Double press 1 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Double press 1 bit z object z***).

- If the format of the object to send is **2 bit**, the ***IN.x - Double press 2 bit z object*** communication object will be visible (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the above parameter are:

  - forcing active on (down)
  - activate OFF forcing (up)
  - deactivate forcing [=forcing deactivation]
  - cyclical switching - ON forcing /OFF forcing
  - **cyclical switching - forcing ON / deactivate forcing (default value)**
  - cyclical switching - OFF forcing/forcing deactivation

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Double press 2 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Double press 2 bit z object z***).

- If the format of the object to send is **1 byte unsigned**, the ***IN.x - Double press 1 byte z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte signed**, the ***IN.x - Double press 1 byte z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - Double press 1 byte z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - Double press 1 byte z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the above parameter are:

- auto mode
- comfort mode
- pre-comfort mode
- economy mode
- off mode (building protection)
- **cyclical switching (thermostat)** **(default value)**
- cyclical switching (timed thermostat)

By selecting the value **cyclic switching (thermostat)**, each time the associated event is detected (double press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Comfort* …;
By selecting the value **cyclic switching (timed thermostat),** each time the associated event is detected (double press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Auto→ Comfort* …

- If the format of the object to send is **2 byte unsigned**, the ***IN.x - Double press 2 byte z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte signed**, the ***IN.x - Double press 2 byte z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the **"On double press detection sends the value"** parameter is a dummy one, used to select the colour to be sent. The real value, downloaded from the memory, will depend on the three parameters that represent the colour components (see below). In addition, the ***IN.x - Double press 3 byte z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the above parameter are:

  - **white (default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

  By selecting **customise**, the following parameters are made visible: **"Value of RED component (0 .. 255)"**, **"Value of GREEN component (0 .. 255)"** and **"Value of BLUE component (0 .. 255)"**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
  The values that can be set are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte unsigned**, the ***IN.x - Double press 4 byte z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte signed**, the ***IN.x - Double press 4 byte z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

  - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

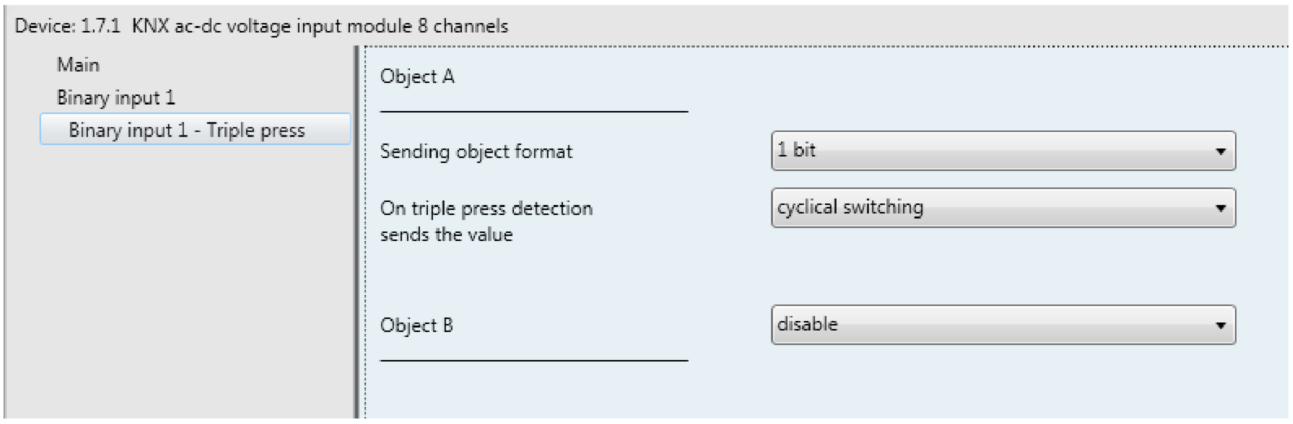
NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with "unsigned and signed value" coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format "2/4 byte signed/unsigned value".

➢ *4.1.4.11.5 Triple press detection*

The parameter **"Triple press detection"** is used to enable the recognition of a triple press, and to visualise the **Binary input x - Triple press** menu for enabling and configuring the commands that will be sent following the recognition of a triple press.
The values that can be set are:

- disabled
- **enabled** **(default value)**

selecting **enabled**, displays the menu **Binary input x - Triple press** as shown in the following figure:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Triple press

Object A

Sending object format	1 bit
On triple press detection sends the value	cyclical switching
Object B	disable

Fig. 4.17: "Binary input x – Triple press" menu

Upon detection of the triple press, it is possible to send up to 4 different objects (which are distinguished by the letters A, B, C and D); object A is always enabled, whereas the parameter **"Object z"** (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

selecting **enabled** displays the parameters **"Sending object format"** and **"On triple press detection sends the value"** grouped in the subgroup **Object z** (where z indicates the object associated with the binary input, included between **A** and **D)**.

The parameter **"Sending object format"** makes it possible to set the format and code of the object "z" of input "x" that is sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit

- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value
- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes
- 3 bytes RGB colour

Depending on the value set for this item, the values that can be set for the **“On triple press detection sends the value”** parameter will be different.
The **“On double triple detection sends the value”** parameter is used to set the command or value to send following the detection of a triple press (on the basis of the set sending conditions) associated with the binary input. The values that can be set are:

- If the format of the object to send is **1 bit**, the ***IN.x - Triple press 1 bit z object*** communication object will be visible (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the above parameter are:

  - 0
  - 1
  - **cyclical switching (default value)**

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Triple press 1 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Triple press 1 bit z object z***).

- If the format of the object to send is **2 bit**, the ***IN.x - Triple press 2 bit z object*** communication object will be visible (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the above parameter are:

  - forcing active on (down)
  - activate OFF forcing (up)
  - deactivate forcing [=forcing deactivation]
  - cyclical switching - ON forcing /OFF forcing
  - **cyclical switching - forcing ON / deactivate forcing (default value)**
  - cyclical switching - OFF forcing/forcing deactivation

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Triple press 2 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Triple press 2 bit z object z***).

- If the format of the object to send is **1 byte unsigned**, the ***IN.x - Triple press 1 byte z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte signed**, the ***IN.x - Triple press 1 byte z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - Triple press 1 byte z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the above parameter are:

- from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - Triple press 1 byte z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the above parameter are:

  - auto mode
  - comfort mode
  - pre-comfort mode
  - economy mode
  - off mode (building protection)
  - **cyclical switching (thermostat) (default value)**
  - cyclical switching (timed thermostat)

  By selecting the value **cyclic switching (thermostat)**, each time the associated event is detected (triple press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Comfort …*;
  By selecting the value **cyclic switching (timed thermostat),** each time the associated event is detected (triple press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Auto→ Comfort …*

- If the format of the object to send is **2 byte unsigned**, the ***IN.x - Triple press 2 byte z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte signed**, the ***IN.x - Triple press 2 byte z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the **"On triple press detection sends the value"** parameter is a dummy one, used to select the colour to be sent. The real value, downloaded from the memory, will depend on the three parameters that represent the colour components (see below). In addition, the ***IN.x - Triple press 3 byte z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the above parameter are:

  - **white (default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

  By selecting **customise**, the following parameters are made visible: **"Value of RED component (0 .. 255)"**, **"Value of GREEN component (0 .. 255)"** and **"Value of BLUE component (0 .. 255)"**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
  The values that can be set are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte unsigned**, the ***IN.x - Triple press 4 byte z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte signed**, the ***IN.x - Triple press 4 byte z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

  - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

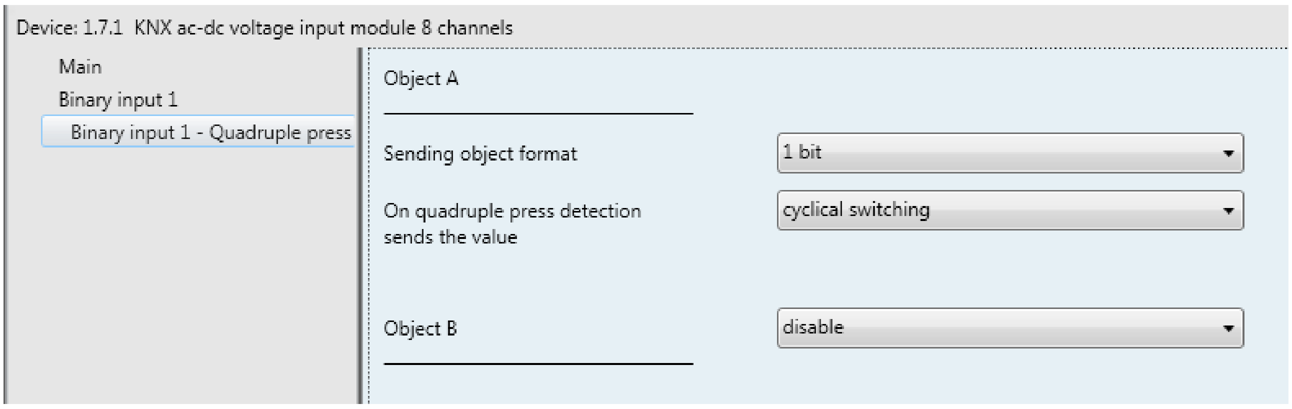
NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with "unsigned and signed value" coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format "2/4 byte signed/unsigned value".

➢ *4.1.4.11.6 Quadruple press detection*

The parameter **"Quadruple press detection"** is used to enable the recognition of a quadruple press, and to visualise the **Binary input x - Quadruple press** menu for enabling and configuring the commands that will be sent following the recognition of a quadruple press. The values that can be set are:

- disabled
- **enabled** **(default value)**

selecting **enabled**, displays the menu **Binary input x - Quadruple press** as shown in the following figure:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Sending object format: 1 bit
On quadruple press detection sends the value: cyclical switching
Object B: disable

Fig. 4.18: "Binary input x – Quadruple press" menu

Upon detection of the quadruple press, it is possible to send up to 4 different objects (which are distinguished by the letters A, B, C and D); object A is always enabled, whereas the parameter **"Object z"** (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

selecting **enabled** displays the parameters **"Sending object format"** and **"On quadruple press detection sends the value"** grouped in the subgroup **Object z** (where z indicates the object associated with the binary input, included between **A** and **D)**.

The parameter **"Sending object format"** makes it possible to set the format and code of the object "z" of input "x" that is sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit
- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value
- 3 bytes RGB colour
- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes

Depending on the value set for this item, the values that can be set for the **"On quadruple press detection sends the value"** parameter will be different.
The **"On double quadruple detection sends the value"** parameter is used to set the command or value to send following the detection of a quadruple press (on the basis of the set sending conditions) associated with the binary input. The values that can be set are:

- If the format of the object to send is **1 bit**, the ***IN.x - Quadruple press 1 bit z object*** communication object will be visible (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the above parameter are:

  - 0
  - 1
  - **cyclical switching** **(default value)**

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Quadruple press 1 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Quadruple press 1 bit z object z***).

- If the format of the object to send is **2 bit**, the ***IN.x - Quadruple press 2 bit z object*** communication object will be visible (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the above parameter are:

  - forcing active on (down)
  - activate OFF forcing (up)
  - deactivate forcing [=forcing deactivation]
  - cyclical switching - ON forcing /OFF forcing
  - **cyclical switching - forcing ON / deactivate forcing** **(default value)**
  - cyclical switching - OFF forcing/forcing deactivation

  selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Quadruple press 2 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Quadruple press 2 bit z object z***).

- If the format of the object to send is **1 byte unsigned**, the ***IN.x - Quadruple press 1 byte z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte signed**, the ***IN.x - Quadruple press 1 byte z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - Quadruple press 1 byte z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - Quadruple press 1 byte z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the above parameter are:

  - auto mode
  - comfort mode
  - pre-comfort mode
  - economy mode
  - off mode (building protection)
  - **cyclical switching (thermostat)** **(default value)**
  - cyclical switching (timed thermostat)

  By selecting the value **cyclic switching (thermostat)**, each time the associated event is detected (quadruple press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Comfort* …;
  By selecting the value **cyclic switching (timed thermostat),** each time the associated event is detected (quadruple press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Auto→ Comfort* …

- If the format of the object to send is **2 byte unsigned**, the ***IN.x - Quadruple press 2 byte z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte signed**, the ***IN.x - Quadruple press 2 byte z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the **"On quadruple press detection sends the value"** parameter is a dummy one, used to select the colour to be sent. The real value, downloaded from the memory, will depend on the three parameters that represent the colour components (see below). In addition, the ***IN.x - Quadruple press 3 byte z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the above parameter are:

  - **white (default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

  By selecting **customise**, the following parameters are made visible: **"Value of RED component (0 .. 255)"**, **"Value of GREEN component (0 .. 255)"** and **"Value of BLUE component (0 .. 255)"**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
  The values that can be set are:

- from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte unsigned**, the ***IN.x - Quadruple press 4 byte z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

    - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte signed**, the ***IN.x - Quadruple press 4 byte z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

    - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

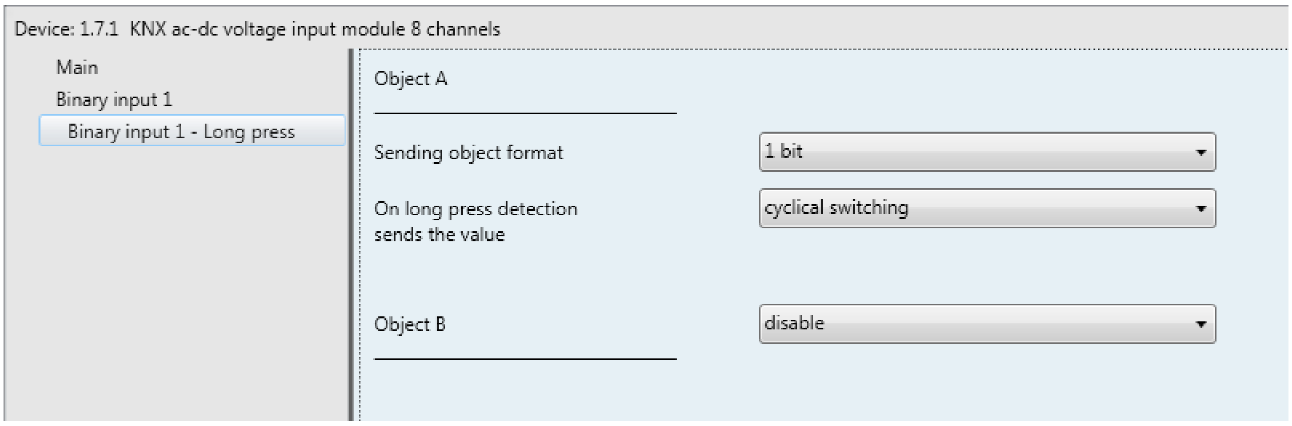
NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with “unsigned and signed value” coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format “2/4 byte signed/unsigned value”.

➢ *4.1.4.11.7 Long press detection*

The parameter **“Long press detection”** is used to enable the recognition of a long press, and to visualise the **Binary input x - Long press** menu for enabling and configuring the commands that will be sent following the recognition of a long press. The values that can be set are:

- disabled
- **enabled** **(default value)**

selecting **enabled**, displays the menu **Binary input x - Long press** as shown in the following figure:


Device: 1.7.1 KNX ac-dc voltage input module 8 channels

Main
Binary input 1
Binary input 1 - Long press

Object A

Sending object format: 1 bit

On long press detection sends the value: cyclical switching

Object B: disable

Fig. 4.19: “Binary input x – Long press” menu

Upon detection of the long press, it is possible to send up to 4 different objects (which are distinguished by the letters A, B, C and D); object A is always enabled, whereas the **“Object z”** parameter (z is the index of the object associated with the threshold, between **A** and **D)** can be used to enable a new object to be sent. The parameter may assume the following values:

- **disable** **(default value)**
- enable

selecting **enabled** displays the parameters **“Sending object format”** and **“On long press detection sends the value”** grouped in the subgroup **Object z** (where z indicates the object associated with the binary input, included between **A** and **D)**.

The parameter **“Sending object format”** makes it possible to set the format and code of the object “z” of input “x” that is sent by the device. The values that can be set are:

- **1 bit** **(default value)**
- 2 bit
- 1 byte unsigned value
- 1 byte signed value
- 1 byte percentage value
- 1 byte HVAC mode
- 2 bytes unsigned value
- 2 bytes signed value
- 3 bytes RGB colour
- 4 bytes unsigned value
- 4 bytes signed value
- 14 bytes

Depending on the value set for this item, the values that can be set for the **“On long press detection sends the value”** parameter will be different.
The **“On long press detection sends the value”** parameter is used to set the command or value to send following the detection of a long press (on the basis of the set sending conditions) associated with the channel. The values that can be set are:

- If the format of the object to send is **1 bit**, the ***IN.x - Long press 1 bit z object*** communication object will be visible (Data Point Type: 1.002 DPT_Bool) and the values that can be set for the above parameter are:

    - 0
    - 1
    - **cyclical switching** **(default value)**

    selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Long press 1 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Long press 1 bit z object z***).

- If the format of the object to send is **2 bit**, the ***IN.x - Long press 2 bit z object*** communication object will be visible (Data Point Type: 2.001 DPT_Switch_Control) and the values that can be set for the above parameter are:

    - forcing active on (down)
    - activate OFF forcing (up)
    - deactivate forcing [=forcing deactivation]
    - cyclical switching - ON forcing /OFF forcing
    - **cyclical switching - forcing ON / deactivate forcing** **(default value)**
    - cyclical switching - OFF forcing/forcing deactivation

    selecting **cyclic switching**, the command that the device will send (via the object ***IN.x - Long press 2 bit z object z***) when the event associated with the cyclical switching detected will be the opposite of the last value sent (via the object ***IN.x - Long press 2 bit z object z***).

- If the format of the object to send is **1 byte unsigned**, the ***IN.x - Long press 1 byte z object*** communication object will be visible (Data Point Type: 5.010 DPT_Value_1_Ucount) and the values that can be set for the above parameter are:

    - from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **1 byte signed**, the ***IN.x - Long press 1 byte z object*** communication object will be visible (Data Point Type: 6.010 DPT_Value_1_Count) and the values that can be set for the above parameter are:

  - from -128 to 127 in steps of 1 **(default value 0)**

- If the format of the object to send is **1 byte percentage value**, the ***IN.x - Long press 1 byte z object*** communication object will be visible (Data Point Type: 5.001 DPT_Scaling) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 100, in steps of 1

- If the format of the object to send is **1 byte HVAC mode**, the ***IN.x - Long press 1 byte z object*** communication object will be visible (Data Point Type: 20.102 DPT_HVACMode) and the values that can be set for the above parameter are:

  - auto mode
  - comfort mode
  - pre-comfort mode
  - economy mode
  - off mode (building protection)
  - **cyclical switching (thermostat) (default value)**
  - cyclical switching (timed thermostat)

  By selecting the value **cyclic switching (thermostat)**, each time the associated event is detected (long press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Comfort* …;
  By selecting the value **cyclic switching (timed thermostat),** each time the associated event is detected (long press) the device sends a new thermoregulation mode (HVAC), following the order *Comfort→ Precomfort→ Economy→ Off→ Auto→ Comfort* …

- If the format of the object to send is **2 byte unsigned**, the ***IN.x - Long press 2 byte z object*** communication object will be visible (Data Point Type: 7.001 DPT_Value_2_Ucount) and the values that can be set for the above parameter are:

  - from **0 (default value)** to 65535, in steps of 1

- If the format of the object to send is **2 byte signed**, the ***IN.x - Long press 2 byte z object*** communication object will be visible (Data Point Type: 8.001 DPT_Value_2_Count) and the values that can be set for the above parameter are:

  - from -32768 to +32767 in steps of 1 **(default value 0)**

- If the format of the object to send is **3 bytes RGB colour**, the **“On long press detection sends the value”** parameter is a dummy one, used to select the colour to be sent. The real value, downloaded from the memory, will depend on the three parameters that represent the colour components (see below). In addition, the ***IN.x - Long press 3 byte z object*** communication object will be visible (Data Point Type: 232.600 DPT_Colour_RGB) and the values that can be set for the above parameter are:

  - **white (default value)**
  - yellow
  - magenta
  - red
  - turquoise
  - green
  - blue
  - customise

By selecting **customise**, the following parameters are made visible: **"Value of RED component (0 .. 255)"**, **"Value of GREEN component (0 .. 255)"** and **"Value of BLUE component (0 .. 255)"**; The combination of the three colour components determines the actual value sent on the BUS. If you select any of the other values, these parameters will still be visible but with pre-set values that cannot be modified.
The values that can be set are:

- from **0 (default value)** to 255, in steps of 1

- If the format of the object to send is **4 byte unsigned**, the ***IN.x - Long press 4 byte z object*** communication object will be visible (Data Point Type: 12.001 DPT_Value_4_Ucount) and the values that can be set for the above parameter are:

    - from **0 (default value)** to 4294967295, in steps of 1

- If the format of the object to send is **4 byte signed**, the ***IN.x - Long press 4 byte z object*** communication object will be visible (Data Point Type: 13.001 DPT_Value_4_Count) and the values that can be set for the above parameter are:

    - from -2147483648 to 2147483647 in steps of 1 **(default value 0)**

NOTE: to remedy the problem of coding and the method for inserting values with 2/4 byte floating point format (DPT 9.0xx and 14.0xx), there is an external transformation tool that makes it possible to enter a value in the floating format and obtain the corresponding value with "unsigned and signed value" coding, and vice versa. In this way, the user obtains the value to be entered in the ETS database, selecting the format "2/4 byte signed/unsigned value".

5 Communication objects

Communication objects with **output** functions

#								Object name	Object function	Description	Datapoint type
IN 1	IN 2	IN 3	IN 4	IN 5	IN 6	IN 7	IN 8				
1	26	51	76	101	126	151	176	IN.x - Switch	On/Off	Dimmer switching on/off commands	1.001 DPT_Switch
1	26	51	76	101	126	151	176	IN.x - Shutter movement	Up/Down	Moves up/down the shutter	1.008 DPT_UpDown
1	26	51	76	101	126	151	176	IN.x - Scene	Execute/Store	Send learn/execute scene commands	18.001 DPT_SceneControl
1	26	51	76	101	126	151	176	IN.j - Sequence A	On/Off	Sends On/Off commands associated of the sequence with object A	1.001 DPT_Switch
1	26	51	76	101	126	151	176	IN.x - A object 1 bit value	1/0 value	Send 1/0 values associated to object A	1.002 DPT_Bool
1	26	51	76	101	126	151	176	IN.x - Primary counter	1 byte unsigned value	Send unsigned value (0..255) of the primary counter	5.010 DPT_Value_1_Ucount
1	26	51	76	101	126	151	176	IN.x - Primary counter	1 byte signed value	Send signed value (-128..127) of the primary counter	6.010 DPT_Value_1_Count
1	26	51	76	101	126	151	176	IN.x - Primary counter	2 byte unsigned value	Send unsigned value (0..65535) of the primary counter	7.001 DPT_Value_2_Ucount
1	26	51	76	101	126	151	176	IN.x - Primary counter	2 byte signed value	Send signed value (-32768..32767) of the primary counter	8.001 DPT_Value_2_Count
1	26	51	76	101	126	151	176	IN.x - Primary counter	4 byte unsigned value	Send unsigned value (0.. 4294967295) of the primary counter	12.001 DPT_Value_4_Ucount
1	26	51	76	101	126	151	176	IN.x - Primary counter	4 byte signed value	Send signed value (-2147483648.. 2147483647) of the primary counter	13.001 DPT_Value_4_Count
1	26	51	76	101	126	151	176	IN.x - A object 2 bits value	On/Off forced positioning	Send priority commands associated to object A of the sequence	1.002 DPT_Switch_Control
1	26	51	76	101	126	151	176	IN.x - A object 1 byte value	Unsigned value	Send unsigned values (0..255) associated to object A	5.010 DPT_Value_1_Ucount
1	26	51	76	101	126	151	176	IN.x - A object 1 byte value	Signed value	Send signed values (-128..127) associated to object A	6.010 DPT_Value_1_Count
1	26	51	76	101	126	151	176	IN.x - A object 1 byte value	% value	Send percent values (0%..100%) associated to object A	5.001 DPT_Scaling
1	26	51	76	101	126	151	176	IN.x - A object 1 byte value	HVAC mode	Send HVAC mode (auto/comfort/precomfort/economy/off)	20.102 DPT_HVACMode

										associated to object A	
1	26	51	76	101	126	151	176	IN.x - A object 2 bytes value	Unsigned value	Send unsigned values (0..65535) associated to object A	7.001 DPT_Value_2_Ucount
1	26	51	76	101	126	151	176	IN.x - A object 2 bytes value	Signed value	Send signed values (-32768..32767) associated to object A	8.001 DPT_Value_2_Count
1	26	51	76	101	126	151	176	IN.x - A object 3 byte value	RGB color	Send RGB color components value associated to object A	232.600 DPT_Colour_RGB
1	26	51	76	101	126	151	176	IN.x - A object 4 bytes value	Unsigned value	Sends unsigned values (0.. 4294967295) associated to object A	12.001 DPT_Value_4_Ucount
1	26	51	76	101	126	151	176	IN.x - A object 4 bytes value	Signed value	Sends signed values (-2147483648.. 2147483647) associated to object A	13.001 DPT_Value_4_Count
1	26	51	76	101	126	151	176	IN.x - A object 14 bytes value	Characters ISO 8859-1	Send characters codified with ISO 8859-1 standard associated to object A	16.001 DPT_String_8859_1
1	26	51	76	101	126	151	176	IN.x - Single press 1 bit A object	1/0 value	Send 1/0 values associated to object A of the single press	1.002 DPT_Bool
1	26	51	76	101	126	151	176	IN.x - Single press 2 bit A object	On/Off forced positioning	Send priority commands associated to object A of the single press	1.002 DPT_Switch_Control
1	26	51	76	101	126	151	176	IN.x - Single press 1 byte A object	Unsigned value	Send unsigned values (0..255) associated to object A of the single press	5.010 DPT_Value_1_Ucount
1	26	51	76	101	126	151	176	IN.x - Single press 1 byte A object	Signed value	Send signed values (-128..127) associated to object A of the single press	6.010 DPT_Value_1_Count
1	26	51	76	101	126	151	176	IN.x - Single press 1 byte A object	% value	Send percent values (0%..100%) associated to object A of the single press	5.001 DPT_Scaling
1	26	51	76	101	126	151	176	IN.x - Single press 1 byte A object	HVAC mode	Send HVAC mode (auto/comfort/precomfort/economy/off) associated to object A of single press	20.102 DPT_HVACMode
1	26	51	76	101	126	151	176	IN.x - Single press 2 byte A object	Unsigned value	Send unsigned values (0..65535) associated to object A of the single press	7.001 DPT_Value_2_Ucount
1	26	51	76	101	126	151	176	IN.x - Single press 2 byte A object	Signed value	Send signed values (-32768..32767) associated to object A of the single press	8.001 DPT_Value_2_Count

1	26	51	76	101	126	151	176	IN.x - Single press 3 byte A object	RGB color	Send RGB color components value associated to object A of the single press	232.600 DPT_Colour_RGB
1	26	51	76	101	126	151	176	IN.x - Single press 4 byte A object	Unsigned value	Sends unsigned values (0.. 4294967295) associated to object A of the single press	12.001 DPT_Value_4_Ucount
1	26	51	76	101	126	151	176	IN.x - Single press 4 byte A object	Signed value	Sends signed values (-2147483648.. 2147483647) associated to object A of the single press	13.001 DPT_Value_4_Count
2	27	52	77	102	127	152	177	IN.x - Brightness dimming	Increase/Decrease	Dimmer brightness increasing/decreasing commands	3.007 DPT_Control_Dimming
2	27	52	77	102	127	152	177	IN.x - Shutter stop/Louvres control	Stop/Step	Send stop movement/slat regulation commands	1.007 DPT_Step
2	27	52	77	102	127	152	177	IN.x - B sequence	On/Off	Send On/Off commands associated to object B of the sequence	1.001 DPT_Switch
2	27	52	77	102	127	152	177	IN.x - Primary counter bit overflow	Overflow status	Send the primary counter overflow feedback	1.002 DPT_Bool
2	27	52	77	102	127	152	177	IN.x - B object 1 bit value	1/0 value	Send 1/0 values associated to object B	1.002 DPT_Bool
2	27	52	77	102	127	152	177	IN.x - Single press 1 bit B object	1/0 value	Send 1/0 values associated to object B of the single press	1.002 DPT_Bool
3	28	53	78	103	128	153	178	IN.x - C sequence	On/Off	Send On/Off commands associated to object C of the sequence	1.001 DPT_Switch
3	28	53	78	103	128	153	178	IN.x - Primary counter byte overflow	Overflow status	Send the value associated to the primary counter overflow feedback	5.010 DPT_Value_1_Ucount
3	28	53	78	103	128	153	178	IN.j - 1 bit value associated to object C	1/0 value	Sends values 1/0 associated with object C	1.002 DPT_Bool
3	28	53	78	103	128	153	178	IN.x - C object 1 bit value	1/0 value	Send 1/0 values associated to object C	1.002 DPT_Bool
4	29	54	79	104	129	154	179	IN.x - D sequence	On/Off	Send On/Off commands associated to object D of the sequence	1.001 DPT_Switch
4	29	54	79	104	129	154	179	IN.x - Differential counter	1 byte unsigned value	Send unsigned value (0..255) of the differential counter	5.010 DPT_Value_1_Ucount
4	29	54	79	104	129	154	179	IN.x - Differential counter	1 byte signed value	Send signed value (-128..127) of the differential counter	6.010 DPT_Value_1_Count
4	29	54	79	104	129	154	179	IN.x - Differential counter	2 byte unsigned value	Send unsigned value (0..65535) of the differential counter	7.001 DPT_Value_2_Ucount
4	29	54	79	104	129	154	179	IN.x - Differential	2 byte signed	Send signed value (-	8.001

								counter	value	32768..32767) of the differential counter	DPT_Value _2_Count
4	29	54	79	104	129	154	179	IN.x - Differential counter	4 byte unsigned value	Send unsigned value (0.. 4294967295) of the differential counter	12.001 DPT_Value _4_Ucount
4	29	54	79	104	129	154	179	IN.x - Differential counter	4 byte signed value	Send signed value (-2147483648.. 2147483647) of the differential counter	13.001 DPT_Value _4_Count
4	29	54	79	104	129	154	179	IN.x - D object 1 bit value	1/0 value	Send 1/0 values associated to object D	1.002 DPT_Bool
4	29	54	79	104	129	154	179	IN.x - Single press 1 bit D object	1/0 value	Send 1/0 values associated to object D of the single press	1.002 DPT_Bool
5	30	55	80	105	130	155	180	IN.x - Differential counter bit overflow	Overflow status	Send the differential counter overflow feedback	1.002 DPT_Bool
5	30	55	80	105	130	155	180	IN.x - Double press 1 bit A object	1/0 value	Send 1/0 values associated to object A of the double press	1.002 DPT_Bool
5	30	55	80	105	130	155	180	IN.x - Double press 2 bit A object	On/Off forced positioning	Send priority commands associated to object A of the double press	1.002 DPT_Switch _Control
5	30	55	80	105	130	155	180	IN.x - Double press 1 byte A object	Unsigned value	Send unsigned values (0..255) associated to object A of the double press	5.010 DPT_Value _1_Ucount
5	30	55	80	105	130	155	180	IN.x - Double press 1 byte A object	Signed value	Send signed values (-128..127) associated to object A of the double press	6.010 DPT_Value _1_Count
5	30	55	80	105	130	155	180	IN.x - Double press 1 byte A object	% value	Send percent values (0%..100%) associated to object A of the double press	5.001 DPT_Scalin g
5	30	55	80	105	130	155	180	IN.x - Double press 1 byte A object	HVAC mode	Send HVAC mode (auto/comfort/precom fort/economy/off) associated to object A of double press	20.102 DPT_HVAC Mode
5	30	55	80	105	130	155	180	IN.x - Double press 2 byte A object	Unsigned value	Send unsigned values (0..65535) associated to object A of the double press	7.001 DPT_Value _2_Ucount
5	30	55	80	105	130	155	180	IN.x - Double press 2 byte A object	Signed value	Send signed values (-32768..32767) associated to object A of the double press	8.001 DPT_Value _2_Count
5	30	55	80	105	130	155	180	IN.x - Double press 3 byte A object	RGB color	Send RGB color components value associated to object A of the double press	232.600 DPT_Colour _RGB
5	30	55	80	105	130	155	180	IN.x - Double press 4 byte A object	Unsigned value	Sends unsigned values (0.. 4294967295) associated to object A of the double press	12.001 DPT_Value _4_Ucount
5	30	55	80	105	130	155	180	IN.x - Double press 4	Signed value	Sends signed values	13.001

								byte A object		(-2147483648.. 2147483647) associated to object A of the double press	DPT_Value_4_Count
6	31	56	81	106	131	156	181	IN.x - Differential counter byte overflow	Overflow status	Send the value associated to the differential counter overflow feedback	5.010 DPT_Value_1_Ucount
6	31	56	81	106	131	156	181	IN.x - Double press 1 bit B object	1/0 value	Send 1/0 values associated to object B of the double press	1.002 DPT_Bool
7	32	57	82	107	132	157	182	IN.x - Double press 1 bit C object	1/0 value	Send 1/0 values associated to object C of the double press	1.002 DPT_Bool
8	33	58	83	108	133	158	183	IN.x - Double press 1 bit D object	1/0 value	Send 1/0 values associated to object D of the double press	1.002 DPT_Bool
9	34	59	84	109	134	159	184	IN.x - Triple press 1 bit A object	1/0 value	Send 1/0 values associated to object A of the triple press	1.002 DPT_Bool
9	34	59	84	109	134	159	184	IN.x - Triple press 2 bit A object	On/Off forced positioning	Send priority commands associated to object A of the triple press	1.002 DPT_Switch_Control
9	34	59	84	109	134	159	184	IN.x - Triple press 1 byte A object	Unsigned value	Send unsigned values (0..255) associated to object A of the triple press	5.010 DPT_Value_1_Ucount
9	34	59	84	109	134	159	184	IN.x - Triple press 1 byte A object	Signed value	Send signed values (-128..127) associated to object A of the triple press	6.010 DPT_Value_1_Count
9	34	59	84	109	134	159	184	IN.x - Triple press 1 byte A object	% value	Send percent values (0%..100%) associated to object A of the triple press	5.001 DPT_Scaling
9	34	59	84	109	134	159	184	IN.x - Triple press 1 byte A object	HVAC mode	Send HVAC mode (auto/comfort/precomfort/economy/off) associated to object A of triple press	20.102 DPT_HVACMode
9	34	59	84	109	134	159	184	IN.x - Triple press 2 byte A object	Unsigned value	Send unsigned values (0..65535) associated to object A of the triple press	7.001 DPT_Value_2_Ucount
9	34	59	84	109	134	159	184	IN.x - Triple press 2 byte A object	Signed value	Send signed values (-32768..32767) associated to object A of the triple press	8.001 DPT_Value_2_Count
9	34	59	84	109	134	159	184	IN.x - Triple press 3 byte A object	RGB color	Send RGB color components value associated to object A of the triple press	232.600 DPT_Colour_RGB
9	34	59	84	109	134	159	184	IN.x - Triple press 4 byte A object	Unsigned value	Sends unsigned values (0.. 4294967295) associated to object A of the triple press	12.001 DPT_Value_4_Ucount
9	34	59	84	109	134	159	184	IN.x - Triple press 4 byte A object	Signed value	Sends signed values (-2147483648.. 2147483647)	13.001 DPT_Value_4_Count

										associated to object A of the triple press	
10	35	60	85	110	135	160	185	IN.x - Triple press 1 bit B object	1/0 value	Send 1/0 values associated to object B of the triple press	1.002 DPT_Bool
11	36	61	86	111	136	161	186	IN.x - Triple press 1 bit C object	1/0 value	Send 1/0 values associated to object C of the triple press	1.002 DPT_Bool
12	37	62	87	112	137	162	187	IN.x - Triple press 1 bit D object	1/0 value	Send 1/0 values associated to object D of the triple press	1.002 DPT_Bool
13	38	63	88	113	138	163	188	IN.x - Quadruple press 1 bit A object	1/0 value	Send 1/0 values associated to object A of the quadruple press	1.002 DPT_Bool
13	38	63	88	113	138	163	188	IN.x - Quadruple press 2 bit A object	On/Off forced positioning	Send priority commands associated to object A of the quadruple press	1.002 DPT_Switch_Control
13	38	63	88	113	138	163	188	IN.x - Quadruple press 1 byte A object	Unsigned value	Send unsigned values (0..255) associated to object A of the quadruple press	5.010 DPT_Value_1_Ucount
13	38	63	88	113	138	163	188	IN.x - Quadruple press 1 byte A object	Signed value	Send signed values (-128..127) associated to object A of the quadruple press	6.010 DPT_Value_1_Count
13	38	63	88	113	138	163	188	IN.x - Quadruple press 1 byte A object	% value	Send percent values (0%..100%) associated to object A of the quadruple press	5.001 DPT_Scaling
13	38	63	88	113	138	163	188	IN.x - Quadruple press 1 byte A object	HVAC mode	Send HVAC mode (auto/comfort/precomfort/economy/off) associated to object A of quadruple press	20.102 DPT_HVACMode
13	38	63	88	113	138	163	188	IN.x - Quadruple press 2 byte A object	Unsigned value	Send unsigned values (0..65535) associated to object A of the quadruple press	7.001 DPT_Value_2_Ucount
13	38	63	88	113	138	163	188	IN.x - Quadruple press 2 byte A object	Signed value	Send signed values (-32768..32767) associated to object A of the quadruple press	8.001 DPT_Value_2_Count
13	38	63	88	113	138	163	188	IN.x - Quadruple press 3 byte A object	RGB color	Send RGB color components value associated to object A of the quadruple press	232.600 DPT_Colour_RGB
13	38	63	88	113	138	163	188	IN.x - Quadruple press 4 byte A object	Unsigned value	Sends unsigned values (0..4294967295) associated to object A of the quadruple press	12.001 DPT_Value_4_Ucount
13	38	63	88	113	138	163	188	IN.x - Quadruple press	Signed value	Sends signed values	13.001

								4 byte A object		(-2147483648.. 2147483647) associated to object A of the quadruple press	DPT_Value_4_Count
14	39	64	89	114	139	164	189	IN.x - Quadruple press 1 bit B object	1/0 value	Send 1/0 values associated to object B of the quadruple press	1.002 DPT_Bool
15	40	65	90	115	140	165	190	IN.x - Quadruple press 1 bit C object	1/0 value	Send 1/0 values associated to object C of the quadruple press	1.002 DPT_Bool
16	41	66	91	116	141	166	191	IN.x - Quadruple press 1 bit D object	1/0 value	Send 1/0 values associated to object D of the quadruple press	1.002 DPT_Bool
17	42	67	92	117	142	167	192	IN.x - Long press 1 bit A object	1/0 value	Send 1/0 values associated to object A of the long press	1.002 DPT_Bool
17	42	67	92	117	142	167	192	IN.x - Long press 2 bit A object	On/Off forced positioning	Send priority commands associated to object A of the long press	1.002 DPT_Switch_Control
17	42	67	92	117	142	167	192	IN.x - Long press 1 byte A object	Unsigned value	Send unsigned values (0..255) associated to object A of the long press	5.010 DPT_Value_1_Ucount
17	42	67	92	117	142	167	192	IN.x - Long press 1 byte A object	Signed value	Send signed values (-128..127) associated to object A of the long press	6.010 DPT_Value_1_Count
17	42	67	92	117	142	167	192	IN.x - Long press 1 byte A object	% value	Send percent values (0%..100%) associated to object A of the long press	5.001 DPT_Scaling
17	42	67	92	117	142	167	192	IN.x - Long press 1 byte A object	HVAC mode	Send HVAC mode (auto/comfort/precomfort/economy/off) associated to object A of long press	20.102 DPT_HVACMode
17	42	67	92	117	142	167	192	IN.x - Long press 2 byte A object	Unsigned value	Send unsigned values (0..65535) associated to object A of the long press	7.001 DPT_Value_2_Ucount
17	42	67	92	117	142	167	192	IN.x - Long press 2 byte A object	Signed value	Send signed values (-32768..32767) associated to object A of the long press	8.001 DPT_Value_2_Count
17	42	67	92	117	142	167	192	IN.x - Long press 3 byte A object	RGB color	Send RGB color components value associated to object A of the long press	232.600 DPT_Colour_RGB
17	42	67	92	117	142	167	192	IN.x - Long press 4 byte A object	Unsigned value	Sends unsigned values (0.. 4294967295) associated to object A of the long press	12.001 DPT_Value_4_Ucount
17	42	67	92	117	142	167	192	IN.x - Long press 4 byte A object	Signed value	Sends signed values (-2147483648.. 2147483647)	13.001 DPT_Value_4_Count

										associated to object A of the long press	
18	43	68	93	118	143	168	193	IN.x - Long press 1 bit B object	1/0 value	Send 1/0 values associated to object B of the long press	1.002 DPT_Bool
19	44	69	94	119	144	169	194	IN.x - Long press 1 bit C object	1/0 value	Send 1/0 values associated to object C of the long press	1.002 DPT_Bool
20	45	70	95	120	145	170	195	IN.x - Long press 1 bit D object	1/0 value	Send 1/0 values associated to object D of the long press	1.002 DPT_Bool

The object variants shown in blue in the table above are not shown for objects B (objects 2/27/52/77/102/127/152/177), C (objects 3/28/53/78/103/128/153/178) and D (objects 4/29/54/79/104/129/154/179) due to space problems, but are still present.
The object variants shown in red in the table above are not shown for objects B (objects 6/31/56/81/106/131/156/181), C (objects 7/32/57/82/107/132/157/182) and D (objects 8/33/58/83/108/133/158/183) due to space problems, but are still present.
The object variants shown in orange in the table above are not shown for objects B (objects 10/35/60/85/110/135/160/185), C (objects 11/36/61/86/111/136/161/186) and D (objects 12/37/62/87/112/137/162/187) due to space problems, but are still present.
The object variants shown in green in the table above are not shown for objects B (objects 14/39/64/89/114/139/164/189), C (objects 15/40/65/90/115/140/165/190) and D (objects 16/41/66/91/116/141/165/191) due to space problems, but are still present.
The object variants shown in grey in the table above are not shown for objects B (objects 18/43/68/93/118/143/168/193), C (objects 19/44/69/94/119/144/169/194) and D (objects 20/45/70/95/120/145/170/195) due to space problems, but are still present.

Communication objects with **input** functions

#								Object name	Object function	Description	Datapoint type
IN 1	IN 2	IN 3	IN 4	IN 5	IN 6	IN 7	IN 8				
0	25	50	75	100	125	150	175	IN.x - Block	Switching On/Off	Allows to activate/deactivate the block function	1.003 DPT_Enable
2	27	52	77	102	127	152	177	IN.x - Scene storing trigger	Store	Receive the sending learn scene message request (trigger)	1.017 DPT_Trigger
21	46	71	96	121	146	171	196	IN.x - Dimmer status feedback	On/Off status	Dimmer activation status feedbacks	1.001 DPT_Switch
21	46	71	96	121	146	171	196	IN.x - A object status feedback	On/Off status	Receive status feedback from actuator for cyclical commutation function of object A	1.001 DPT_Switch
21	46	71	96	121	146	171	196	IN.x - Movement feedback	Increase/Decrease	Receive movement direction feedback from shutter actuator	1.008 DPT_UpDown
21	46	71	96	121	146	171	196	IN.x - Primary counter sending trigger	Counter value transmission	Receive the sending primary counter actual value request (trigger)	1.017 DPT_Trigger
22	47	72	97	122	147	172	197	IN.x - B object status feedback	On/Off status	Receive status feedback from actuator for cyclical commutation function of object B	1.001 DPT_Switch
22	47	72	97	122	147	172	197	IN.x - Differential counter sending trigger	Counter value transmission	Receive the sending differential counter actual value request (trigger)	1.017 DPT_Trigger
23	48	73	98	123	148	173	198	IN.x - C object status feedback	On/Off status	Receive status feedback from actuator for cyclical commutation function of object C	1.001 DPT_Switch
23	48	73	98	123	148	173	198	IN.x - Differential counter reset	Reset value	Receives the differential counter value reset	1.017 DPT_Trigger
24	49	74	99	124	149	174	199	IN.x - D object status feedback	On/Off status	Receive status feedback from actuator for cyclical commutation function of object D	1.001 DPT_Switch

code 7.01.7.563.3

LAST REVISION 03/2015