Binary Inputs

Input Module for
Push buttons, Switches and Sensors.

Software Library Version: 0.1.5
User Manual Version: [0.1.2]_a

www.zennio.com

Contents

1 INTRODUCTION

A variety of Zennio devices incorporate an input interface where it is possible to connect one or more push buttons, switches or on-off sensors, among other accessories.

Please refer to the specific user manual and datasheet of each Zennio device in order to confirm whether this feature is available or not, and for instructions on how to connect these accessories to the input interface of the device.

On the other hand, keep in mind that even if the model of the input accessory itself may be the same for all devices, **the functionality and the ETS configuration may slightly differ depending on the device and the version of the application program**. Please always ensure to download from the Zennio homepage (www.zennio.com) the user manual and annexes that correspond to the specific device and application program being configured.

2 CONFIGURATION

Inputs configured as binary inputs let the device perform the following tasks:

- Retrieving the **state** (1/0) of the input line and detecting **changes** (i.e., button presses, sensor changes, etc.).

- **Reporting** the KNX bus about the above states/changes and triggering the corresponding **actions**, depending on the case.

- Detecting **sabotage** (i.e., unexpected voltage levels on the line) on inputs configured as switch/sensor.

Every binary input needs itself to be configured as one of these two types: **push button** and **switch/sensor**.

2.1 PUSH BUTTON

The actions to be triggered on both, short and long presses (and even on the release of the push button) are independent and parameterisable. It is even possible to set **how long** a press must be to be considered as long.

These actions can consist in **sending the KNX bus** a binary value (0, 1 or an alternating value), a shutter control order, a dimmer control order, a scene run/save order or a constant numeric value (a 1-byte integer, a percentage value, a 2-byte integer or a 2-byte floating point).

A certain **delay** before sending the value to the bus can be configured in some cases. In the case of the binary values, a periodical re-sending can be also configured, which may be useful if intending to link such value to an alarm monitor or similar.

2.2 SWITCH/SENSOR

Binary values (configurable) will be sent to the bus whenever rising or falling edges are detected in the input line. In this case, it is the fact of switching from one state to the other one what matters, and not whether it is for a longer or shorter time. A simple example would be a sensor that commutes between two states depending on certain

physical conditions (light, floods, weight…), being one of which undesired. The usual behaviour is that a certain binary value is sent to the KNX bus whenever the switch/sensor commutes from one state to the other one.

It is possible to introduce a certain **delay** prior to sending these values to the KNX bus – a delay for the "0" and a delay for the "1", no matter which one is sent after which edge (rising / falling). Moreover, **periodically re-sending** the last value is possible by configuring the desired period.

Security checks can optionally be performed for inputs of the Switch/Sensor type, as long as an **end-of-line resistor** has been connected to the switch/sensor. The value of such resistor needs to be configured by parameter (the available values are 2.2 kΩ, 2.7 kΩ, 3.3 kΩ, 4.7 kΩ and 10 kΩ) as well as whether it has been connected in parallel or serially, which depends on the switch/sensor type (*normally open* or *normally closed*).

> ➢ If **normally open**, the line will remain at a low voltage level in the absence of the undesired condition. The occurrence of that situation, however, will cause a rising edge (the switch/sensor gets closed). This type of sensor requires connecting the end-of-line resistor **in parallel**.

> ➢ On the other hand, if **normally closed**, the line remains at a high voltage level until the occurrence of the undesired situation, which will cause a falling edge (the switch/sensor will open). This requires connecting the end-of-line resistor **in series**.

By means of this resistor, it will be possible to distinguish not only the two states of the switch/sensor, but also additional (unexpected) voltage levels (e.g., short-circuits and open circuits due to a **breakdown** or a **sabotage**), which will be reported to the bus through alarm objects.

Figure 1. Left: normally open Sensor (parallel resistor). Right: Normally Closed Sensor (serial resistor).

2.3 LOCKING A BINARY INPUT

It is possible to **lock/unlock** each input independently by writing to the proper objects. While an input remains locked, the application will ignore further switches that may take place on the line, however the **periodical sending** of values, if parameterised, will not be interrupted (the last value will still be re-sent, even if the input switches the state). On the other hand, when the **unlock** event occurs:

- **Push Button**: a fresh evaluation of the current state (high / low) of the line will be performed, as well as the corresponding action.

- **Switch/Sensor**: if specifically enabled in ETS, the current state of the line will be compared to that prior to the lock event. If they are different, it can be assumed that an edge (rising or falling) has taken place while the input remained locked, and therefore the associated action will be triggered.

2.4 INITIAL STATES

Finally, regarding the **initial states**, it is important to keep in mind the following:

- The **lock** state is maintained after a bus power failure. However, a download from ETS sets every input to *unlocked*.

- On switches/sensors with security checks, the **alarm** conditions are evaluated at the start-up of the device, and the alarm objects updated when required.

- The **periodical sending** of values, if configured in ETS, will be resumed after a power loss.

- When the device recovers from a power loss, the **new state of the push buttons** will always be re-evaluated, while that of the **switches/sensors** will be compared the previous one only if configured by parameter (similar behaviour as for the lock function).

- **Toggle** binary objects (0, 1, 0, 1…) are always sent with value "1" on the very first time after a download from ETS. This applies to switches/sensors configured with action "Switching 0/1".

3 ETS PARAMETERISATION

Please note that the screenshots and object names shown next may be slightly different depending on the device and on the application program.
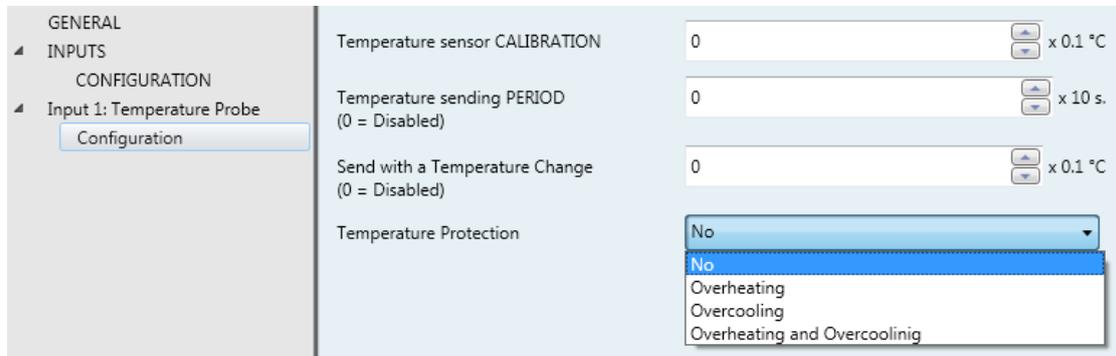

Figure 2. Temperature Probe - Configuration

When an input has been configured as Binary Input, the object "**[Ix] Input Lock**" turns visible (when it receives a "1", the input will become locked, while a "0" will unlock it).

Apart from that, a specific entry will become visible in the menu on the left of the parameters window. This entry will itself comprise a tab called **Configuration**, containing the following parameter:

- **Type**: sets whether the input is a "Push button" (default) or a "Switch/sensor".

3.1 PUSH BUTTON

When **Type** is set to "Push button", the following parameters are available:
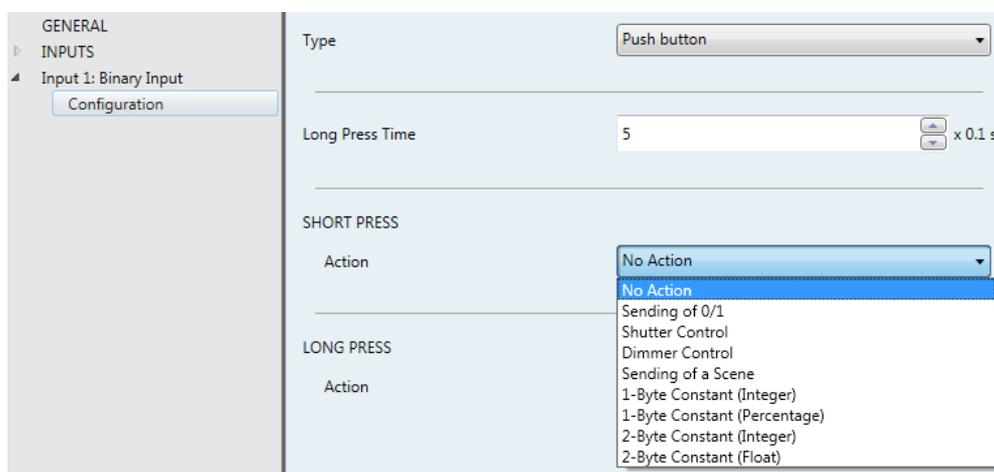

Figure 3. Push Button - Action for Short Press.

- **Long Press Time**: sets the minimum time (1 to 50 tenths of a second) the user press should last before it is considered a long press.

- **Short Press – Action**: defines the action to be triggered when a short press is detected. The available actions are:

  ➢ Nothing

  ➢ Sending of 0/1.

  - **Response**: sets whether to send (through object "**[Ix] Short Press B**") a value (B), which can be "0", "1" or an alternation of "1" and "0".

  - **Delay**: sets a delay between the detection of the button press and the actual sending of the response. The delay must be set in seconds (0-255), minutes (0-255) or hours (0-18).

  - **Periodical Response Sending**: sets whether to re-send the response values periodically or not. The options are "Always", "Only for 0", "Only for 1" and "No" (default). The cycle time must be set in seconds (0-255), minutes (0-255) or hours (0-18).

  ➢ Shutter Control.

  - **Response**: sets the particular order to be sent to the shutter actuator. The options are:

    o "Up": one "0" will be sent through "**[In] [Short Press] Move Up Shutter**",

    o "Down": one "1" will be sent through "**[In] [Short Press] Move Down Shutter**",

    o "Up/Down (switched)": values "1" and "0" will be sent alternatively through "**[Ix] [Short Press] Move Up/Down Shutter**". In this case an additional, writable object ("**[Ix] [Short Press] Shutter Status (input)**") will be available; it may be linked to the shutter status object from the shutter actuator to receive feedback about the current state of the shutter. This will avoid sending move-up orders if the shutter is already at 0% or move-down orders if it is already at 100%.

- o "Stop/Step Up": one "0" will be sent through "**[Ix] [Short Press] Stop/Step Up Shutter**",

- o "Stop/Step Down": one "1" will be sent through "**[Ix] [Short Press] Stop/Step Down Shutter**",

- o "Stop/Switched Step": values "1" and "0" will be sent (alternatively with every press) through "**[Ix] [Short Press] Stop/Step Shutter (switched)**".

- **Delay**: similar as for "Sending of 0/1".

➢ Dimmer Control.

- **Response**: sets the particular order to be sent to the light dimmer:

- o "Light ON": one "1" will be sent through the "**[Ix] [Short Press] Dimmer ON**" binary object,

- o "Light OFF": one "0" will be sent through the "**[Ix] [Short Press] Dimmer OFF**" binary object,

- o "Light ON/OFF (switched)": values "1" and "0" will be sent alternatively through the "**[Ix] [Short Press] Dimmer ON/OFF**" binary object,

- o "Brighter": on every even press, one 4-bit order will be sent (through "**[Ix] [Short Press] Brighter**") to increase the light level by a certain percentage, which needs to be configured through parameter "**Dimming Step**". On the other hand, on every odd press, an order to stop the dimming will be sent. The sequence is therefore Brighter → Stop → Brighter → Stop → etc.

- o "Darker": analogous to the previous one, but for decreasing the light level,

- o "Brighter/Darker (switched)": analogous to the above two, although the dimming orders will be in this case increase/decrease orders (alternatively). The sequence will be therefore Brighter → Stop → Darker → Stop → Brighter → etc. These orders will be sent through object "**[Ix] [Short Press] Brighter/Darker**". In this case there will be an additional, writable object ("**[Ix] [Short Press] Dimming Status (input)**"

that may be linked to the dimming status object from the dimmer, in order to receive feedback about the current light level. This will avoid increase orders if the current level is 100% or decrease orders if it is already 0%

- **Delay**: similar as for "Sending of 0/1".

➢ Sending of a Scene.

- **Response**: sets the particular order to be sent to the bus. The options are "Run Scene" (an order to run a specific scene will be sent through "**[Ix] [Short Press] Run Scene**") and "Save Scene" (an order to save the current state as a specific scene will be sent through "**[Ix] [Short Press] Save Scene**").

- **Scene**: sets the desired scene number (1 to 64) for the above run/save orders.

- **Delay**: similar as for "Sending of 0/1".

➢ 1-Byte Constant (Integer):

- **Response**: sets a constant value to be sent to the bus (through "**[Ix] [Short Press] Constant Value (Integer)**"), in the range 0 to 255.

➢ 1-Byte Constant (Percentage):

- **Response**: sets a constant value to be sent to the bus (through "**[Ix] [Short Press] Constant Value (Percentage)**"), in the range 0 to 100.

➢ 2-Byte Constant (Integer):

- **Response**: sets a constant value to be sent to the bus (through "**[Ix] [Short Press] Constant Value (Integer)**"), in the range 0 to 65535.

➢ 2-Byte Constant (Float) :

- **Response**: sets a constant value to be sent to the bus (through "**[Ix] [Short Press] Constant Value (float)**"), in the range -671088.64 to 670760.96.

- **Long Press – Action**: defines the action to be triggered when a long press is detected. The available actions are **analogous to those for the short press**, except for the following remarks:

  ➢ In the <u>shutter control</u>, if a move-up or move-down (or an alternating move up/down) order is configured as the response, then together with the usual object through which the parameterised order is sent after pressing the button, an additional object will show up: "**[Ix] [Long Press] Stop Shutter**", which will send an order to stop the motion of the shutter as soon as the button is released, thus making it possible to (optionally) implement a hold & release shutter control.

> **Example**: **Long-Press Shutter Control**.
>
> *"Shutter Control" has been set as the desired reaction to long presses, and "Move up" as the response. When a long press is detected, the value "0" will be sent through "**[Ix] [Long Press] Move up Shutter**", while after the release of the push button the value "0" will be sent through "**[Ix] [Long Press] Stop Shutter**", which will only take effect if it gets linked to the corresponding object from the shutter actuator.*

  ➢ In the <u>dimmer control</u>, if an order to increase or decrease the light level (or to increase / decrease it alternatively) is configured as the response, then the usual object through which the parameterised order is sent after pressing the button **will also send an order to stop the light regulation** as soon as the button is released (thus permitting a hold & release dimmer control), which does not take place when this type of response is assigned to short presses.

> **Example**: **Long-Press Dimmer Control**.
>
> *"Dimmer Control" has been set as the desired reaction to long presses, and "Brighter" (with a dimming step of 50%) as the response. When a long press is detected, the value "0xA" will be sent through "**[Ix] [Long Press] Brighter**", while after the release of the push button the value "0x8" will be sent, which will interrupt the light regulation.*

3.2 SWITCH/SENSOR

When **Type** has been set to "Switch/Sensor", the following parameters are also available:

Figure 4. Switch/Sensor.

- **Security**: selecting or unselecting this checkbox determines whether the input line includes a security end-of-line resistor so it is therefore possible to monitor sabotage or breakdown situations (which will be notified by periodically sending the value "1" through object "**[Ix] [Switch/Sensor] Alarm: Breakdown or Sabotage**"; once they are over, this object will send one "0"). When selected, two more parameters come up:

- ➤ **Switch/Sensor Type**: sets whether the switch/sensor is normally open and therefore with a resistor connected in parallel ("N.O. (parallel resistor)") or normally closed and therefore with a resistor connected in series ("N.C. (serial resistor)").

- ➤ **Resistor Value**: sets the value of the resistor, which should be one of the following: 2.2 kΩ, 2.7 kΩ, 3.3 kΩ, 4.7 kΩ and 10 kΩ.
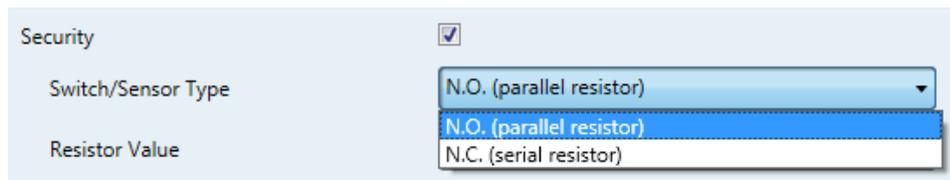

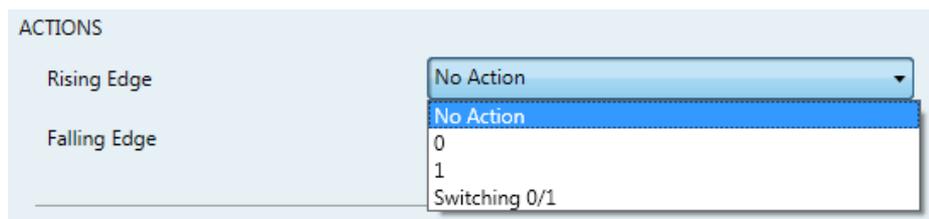**Figure 5**. Switch/Sensor - Security.

.⁕ **Actions.**

Figure 6. Switch/Sensor - Actions.

- ➤ **Rising Edge**: indicates what should be sent to the KNX bus when a rising edge takes place on the input line. The options are "No Action" (default), "0", "1" and "Switching 0/1" (i.e., the values "1" and "0" will alternate with every rising edge detected). These values are sent through object "**[Ix] [Switch/Sensor] Edge**".

- ➤ **Falling Edge**: analogous to the above parameter. The response to falling edges will be sent through the same object ("**[Ix] [Switch/Sensor] Edge**").

.⁕ **Periodical Sending.**

- ➤ **Sending of "0"**: sets every how much time (0 to 255 seconds, 0 to 255 minutes or 0 to 18 hours) the value "0" should be periodically sent once the corresponding edge has been detected. If no periodical re-sending is required, just leave this at 0.

- ➤ **Sending of "1"**: analogous to the above one, but for the value "1".

• **Delay.**

  ➢ **Sending of "0"**: sets a certain delay (0 to 255 seconds, 0 to 255 minutes or 0 to 18 hours) before sending the value "0" once the corresponding edge has been detected. For an immediate sending, just leave this at 0.

  ➢ **Sending of "1"**: analogous to the above one, but for the value "1".

• **Evaluate the Input State after Unlock or Reset**: sets whether the state of the input line should be evaluated or not whenever it gets unlocked (through object "**[Ix] Input Lock**") or when the device recovers from a power loss, so this new state can be compared to the last known, making the device trigger the proper response in case the two are different.

• **Sending Status (0 and 1) on Bus Voltage Recovery**: sets if the status of the line (i.e., the action that corresponds to the fact of switching to that state) should always be sent to the bus when the device recovers from a power loss, even if the status is the same as before the power loss.

Join and send us your inquiries
about Zennio devices:

http://zennioenglish.zendesk.com