PRODUCT MANUAL

ACTinBOX

ACTinBOX CLASSIC KNX

ZN1I0-AB46

Edition 4

Versión 3.2

INDEX:

1. INTRODUCTION

1.1. PRODUCT

The **ACTinBOX CLASSIC** is a KNX actuator that combines in a same device:

- **4 x 10A** multifunction (**INDIVIDUAL** or **SHUTTER**) binary **OUTPUTS.**
- **6** Voltage free contact **INPUTS** to connect **sensors** and/or **push buttons.**
- **Advanced LOGICAL FUNCTIONS** included.

These 3 sections work independently, and can interact the others as if there were 3 different autonomous devices connected to the **KNX BUS**.

1.2. COMMUNICATION OBJECTS

The **ACTinBOX CLASSIC** has **151 Communication Objects** divided into four different sections:

INDIVIDUAL OUTPUTS	**INPUTS**
SHUTTER CHANNELS	**LOGICAL FUNCTIONS**

NOMENCLATURE: To easily find the appropriate Communication Objects during the Group Addressing process, these are named depending on the Section they belong to, as:

["Group Type"] "Function to be executed"

Following abbreviations are associated to the different sections:

- Individual Outputs:

[O1] → Output 1
[O2] → Output 2
[O3] → Output 3
[O4] → Output 4

- Inputs:

[I1] → Input 1
... …….
... …….
[I6] → Input 6

- Shutter Channels:

[CA] → Channel A
[CB] → Channel B

- Logical Functions:

[LF] → Logical Function

Examples: *See (**Figure 1.2**)*

- **[O3] Status:** Output 3 Status object
- **[CA] Stop:** Channel A Shutter Stop Control
- **[I3] Block:** Object to block input 3 control
- **[LF] FUNCTION 5 RESULT (1 bit):** Object to store the 1 bit Logical Function 5 result.

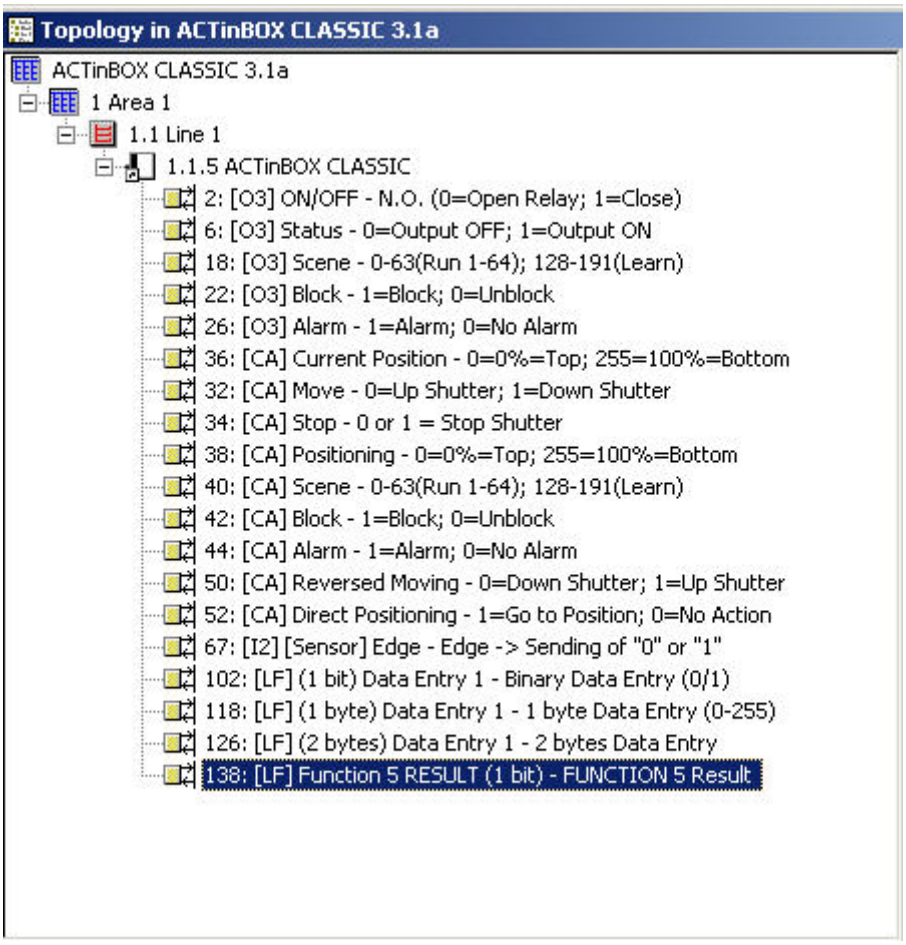


Figure 1.2

2. OUTPUTS

The **ACTinBOX CLASSIC** has **four 10A binary outputs** (see product Datasheet). These 4 outputs are divided into 2 groups (called Channels) with 2 outputs each.

- Channel A: "Output 1" & "Output 2"

- Channel B: "Output 3" & "Output 4"

Both channels can be independently parameterized as:

- **Individual Outputs:** Every output is independent from the others, so all of them are independently managed. To control electrical loads…
- **Shutter Channels:** Outputs in the same channel are jointly managed. To control shutter drives, sun blinds or similar…
  - **In the shutter channel case:**
    - ✓ **« Output 1 » (3 in Channel B):** In charge of raising the shutter.
    - ✓ **«Output 2 » (4 in Channel B):** In charge of lowering the shutter.

Configuration example: *Consider a facility where we need to manage one shutter drive and a light spot.*

*In this case, the **ACTinBOX CLASSIC** could be parameterized as follows:*

- ***Channel A** = Shutter Channel*
- ***Channel B** = Individual Outputs*
  - ✓ ***Output 3** = normally open*
  - ✓ ***Output 4** = Disabled*

2.1.SHUTTER CHANNELS

The **ACTinBOX** CLASSIC allows installing any type of shutter drive control (or similar) on its Channels. To operate them, two basic control objects (**"Move"** & **"Stop/Step"**) together with a set of additional functions (with their own Communications objects) are available.

The Basic shutter control with the above mentioned Communications objects is made as follows:

- **Raise Shutter:** Send value "**0**" to the object "**Move**".

- **Lower Shutter:** Send value "**1**" to the object "**Move**".

Note: *When the object "**Move**" gets a "**0**" or a "**1**", the shutter will start moving, and won't stop unless it reaches its lowest or highest position (depending on the order received), or that receives some other order annulling the previous one.*

- **To stop a shutter in motion:** A "**0**" or a "**1**" must be sent to the object "**Stop**".

2.1.1. TYPE

- **ROLLER SHUTTER (No lamellas) / SUNBLINDS** → These are the typical revolving shutters, with a simple (Raise/Lower) movement.

- **SHUTTER WITH LAMELLAS** → Special shutters with a secondary movement managed by the same drive.
  The **ACTinBOX** in this case, allows controlling both movements, lamella rotation (getting more or less incident light from the outside), and the Raise/Lower movement.
  By selecting this Shutter control, the **"Stop"** object is replaced with the **"Stop/Step"** one.
  This way, if the device receives a **"0"** or a **"1"** via this object when the shutter is in motion, it shall stop; while if the shutter is stopped, receiving a **"0"** via this communication object will make lamellas to pull up; on the contrary, receiving a **"1"** will make lamellas to pull down.

2.1.2. TIMES

It is necessary to set two different times (three when working with shutters with moveable lamellas) for a proper channel working.

- **MAIN TIME (Shutter Height)**: This is the time the shutter needs to cover its height completely. Both times can be used in this field (Raising time or Lowering time); but if there was some difference between these two times, **"Lowering time"** will be considered as "master", and should be used to fill in this field.
  "Raising time" in this case will be set in the **"Total Time up"** field, enabled for this purpose.
  This variable won't need to be periodically calibrated since the exact shutter position remains on the ACTinBOX (even when Power Failure occurs).

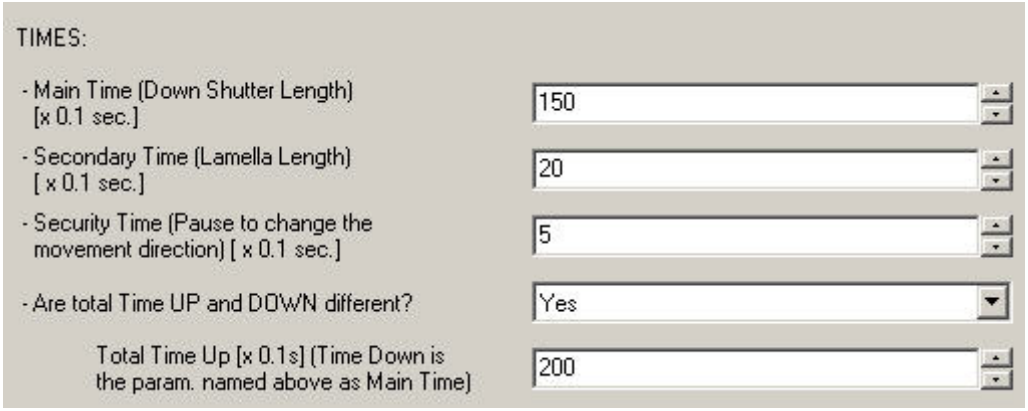
  Note: *After programming the device with the ETS, the ACTinBOX considers the shutter is completely raised.*

- **SECONDARY TIME (Lamellas movement) :** (**Only for Lamella Shutters)** Time used by lamellas to cover a complete deployment (up or down).

- **SECURITY TIME (before changing the movement direction) :** This time is applied by the actuator to protect the drive when the movement direction of the shutter is changed. If the device receives the order to **"Lower"** the shutter while this is being raised, the ACTinBOX will stop it for a while (security time), to later **"Lower"** it. It is recommended to keep the default value in this field: 5 [x 0.1 sec]

- **DIFFERENT UP / DOWN TIMES?** : Whenever the shutter raising and lowering times are different (i.e. heavy shutters), this field should be dropped down to set the raising shutter time in this field, as mentioned above the lowering one must be set in the **"Main Time field"**.

Configuration Example: *Shutter in Channel B takes 15 seconds to be lowered and 20 to be raised. In this case the ACTinBOX parameterization should remain as follows:*



- **ADDITIONAL TIME WHEN SHUTTER GETS THE LIMIT:** This parameter guarantees the shutter always gets its lowest or highest level by setting an extra time for the drive to keep moving once the shutter took up its raising/lowering times, preventing small maladjustments.

2.1.3. FUNCTIONS

Following parameters add functionality or special features to a Shutter Channel Control.

2.1.3.1. STATUS OBJECT

This function provides a communication object **"Current Position"**, to indicate the user the exact position the shutter is at all times.
This is a 1 byte object measured in **"%"**. The object value is in the range [0…255]:

- 0= 0% -> Shutter completely raised
- …….

- …….
- 255=100% -> Shutter completely lowered

When the shutter is in motion, and eventually stops, the CLASSIC actuator can send (via this Object) the exact position of the shutter to update the rest of devices in the BUS when needed

Note: *The shutter "**Current Position**" status object has been programmed so that every time the shutter is in motion, this is transmitted to the BUS to update its position in real time (every second).*

2.1.3.2. PRECISE CONTROL

This function makes possible to move the shutter to any position on its length, via the 1 byte **"Positioning"** communication object.

Every time the **ACTinBOX** gets a new value through this object (e.g. 50%), the shutter is moved to the corresponding position (the middle in the example).

2.1.3.3. SCENES

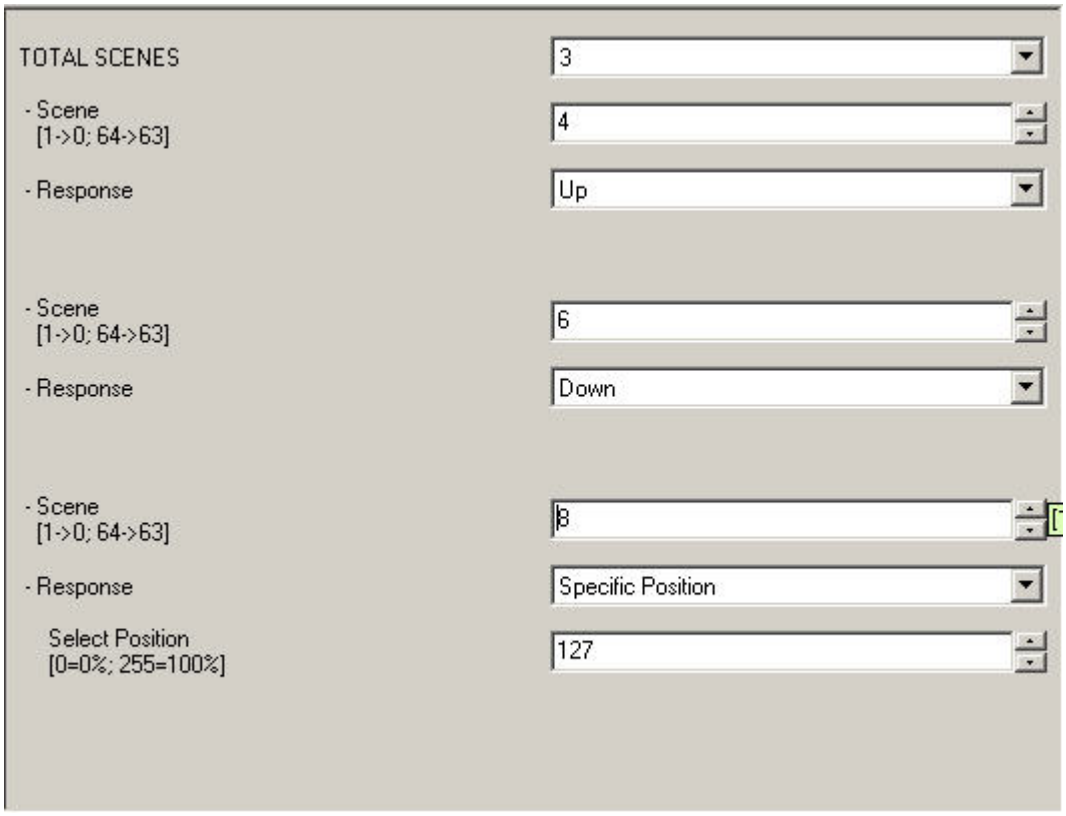
This function makes possible to use a standard (1 byte) scene object to control shutters, giving the users the possibility to choose a precise position where to locate the shutter depending on the scene number received by the ACTinBOX through the object **"Scenes"**.

- **TOTAL SCENES** → To choose the total number of scenes to be used; up to a maximum of five scenes can be set in this field.
- **SCENE**→ Set the scene number.
- **RESPONSE**→ To set the precise position where to locate the shutter when the corresponding scene number is called from the BUS.

Example: *Consider a facility where 4 scenes will be used (**4, 6, 8 & 9**), but only three of them will be controlled from the ACTinBOX CLASSIC to locate the shutter in a precise position:*

- ***Scene 4*** → *Up*
- ***Scene 6*** → *Down*
- ***Scene 8*** → *In the middle (50%)*

Parameterization in this case should remain as follows:



2.1.3.4. BLOCK

This function makes possible to **"block"** an output by disabling its control. Output will be blocked by sending a "**1"** to the corresponding object in the channel.

Note: *Only the Alarm function is higher priority than the block one. This means that , if shutter in the channel is blocked and the Alarm goes On, the shutter shall be carried to the Alarm parameterized position. When Alarm goes off, the shutter will recover its blocked status.*
To unblock the shutter it's necessary to send a ***"0"*** *to the object* ***"Block"****.*

2.1.3.5. ALARM

This function is designed for cases in which the ACTinBOX must response to an alarm situation. When an alarm occurs, this function locates the shutter in the parameterized position, and after this, shutter is blocked (it won´t be controlable until Alarm goes off).

- **NUMBER OF ALARMS** → Set whether to use one or two alarms. Both of them can be independently managed through their corresponding communication objects.

Note: *"**Alarm 1**" is higher priority than **"Alarm 2"**. This means that if channel is in **"Alarm 2"** status and **"Alarm 1"** occurs the shutter will change to **"Alarm 1"** status and it shall not come back to **"Alarm 2"** until **"Alarm 1"** goes off. Whereas if the Channel is in **"Alarm 1"** status, and **"Alarm 2"** occurs , **"Alarm 1"** prevails.*

- **TRIGGER VALUE →** Set the value to activate the Alarm status. An Alarm status will be activated when the value set in this field is sent to the object Alarm (or Alarm 2).
  The opposite of the **"trigger"** value is the **"Passive"** value.

- **CYCLICAL MONITORING →** To be sure that the sensor works properly and that no alarm is active, the sensor to send the **"Trigger Value"** to the ACTinBOX, shall continuously send the **"Passive Value"** to the BUS when there is no alarm active. It is in this case that this parameter should be enabled; this way, if the ACTinBOX doesn't get the **"passive value"** during the parameterized

- **CYCLE TIME**, alarm will be automatically activated.

Note: *It's recommended to set a time higher than double sensor cycle time, to avoid alarm frames missing.*

- **RESPONSE→** Set the response of the actuator channel output when the alarm is activated.

- **DEACTIVATION →** Two different methods to deactivate an active Alarm:

  - ✓ **NORMAL→** By sending the **"Passive value"** (opposite to the Trigger one) to the object **"Alarm"**.
  - ✓ **FROZEN→** Consists in applying the normal method to later send a **"1"** to the object "**Unfreeze alarm**". This method makes the channel output remains blocked even when the alarm status is finished; in this case it will be necessary then that the output is manually enabled from another point in the installation.

  - **END →** This parameter sets the output response when the alarm status is finished.

2.1.3.6. REVERSED MOVING

This function makes possible to control shutters the other way around from usual; this means **"1"** to raise the shutter and **"0"** to lower it. This control is made through the object **"Reversed Moving"** and is compatible with the usual control.

This is really useful when a **"Central Off"** is required in the installation, i.e. to turn the lights off & lower the shutters. In this case, a **"0"** should be sent

to the light **"ON/OFF"** objects, and to the shutter **"Reversed moving"** objects.

2.1.3.7. DIRECT POSITIONING

Function to move a shutter to a prefixed specific position via 1 bit objects.

When value **"1"** is received through one of these objects ("**Direct positioning**" or "**Direct positioning 2**"), the shutter will be moved to the parameterized position.

- **TOTAL DIRECT POSITIONINGS** → Set the number of direct positioning to be used.
- **POSITION** → Choose the exact position to move the shutter to (Remember that 0=0% and 255=100%).
- **NEW POSITIONS SAVING** → Set whether to allow or not new positions saving:
  - ✓ **Save Position 1**
  - ✓ **Save Position 2**

A "**1**" must be sent to these objects in order to save new positions.

2.1.3.8. START UP CONFIGURATION

This function is meant to define the behaviour of the shutter channel output after a BUS Power Failure, or after programming the device with the ETS.

- **INITIAL POSITION** → This field is to define the exact position the shutter should be located after a BUS Power Failure. After programming the device with the ETS, **"current position"** option means the shutter will remain the position it was before de programming.
- **UPDATE** → By enabling this field, the output initial status can be sent to the BUS to update the rest of devices in the installation if needed.
- **START UP SENDING DELAY**→ As some devices in the installation may take longer to restart after a Power Failure, this field allows setting a delay in seconds for the initial configuration to be sent to the BUS, in order to ensure the rest of devices in the installations are ready to receive the corresponding telegrams.

Note: *The initial position is always sent through the object "**Current Position**".*

2.2.INDIVIDUAL OUTPUTS

When this option is selected, both outputs in the channel are completely independent from each other.

2.2.1. TYPE

It is necessary to indicate whether the outputs are **"Normally open"** or **"Normally closed"**.

- Normally Open:	**- Normally Closed:**
ON → Close Relay **OFF** → Open Relay	**ON** → Open Relay **OFF** → Close Relay

2.2.2. FUNCTIONS

Following parameters add functionality or special features to Shutter Channels Control.

2.2.2.1. STATUS OBJECT

The "**status**" object always shows the current status output, and is meant to feedback any other device in the installation when needed.

- ⇨ Output ON -> status output = "**1**".
- ⇨ Output OFF -> status output = "**0**".

This object will be sent to the BUS every time the output status changes.

- o **INTERNAL LINKS** → To internally link the output **"status object"** with:
  - ⇨ **1 bit input objects**
  - ⇨ **1 bit logical function objects** (data entry objects).

  This means that, if an internal link is set, every time the output changes and the new **"Status"** object value is sent to the BUS through the corresponding Group address (when used), the same effect over the internally linked Input &/or Logical function objects will take place; in fact if no feedback signal (to update any other device in the installation) is needed, no group addresses are necessary to make internal links work. This feature is especially useful to feedback the inputs when these control the outputs by means of a "switching" control.

- ✓ **INPUT 1** → When enabled, the output **"status"** object is internally linked with the "1 bit" object in the Nº1 input. Possible options are:
  - Nothing
  - Short Press
  - Long Press
  - Short & Long Press
  - Edge
- ✓ …….. ……..
- ✓ …….. ……..
- ✓ **INPUT 6** → When enabled, the output **"status"** object is internally linked with the "1 bit" object in the Nº6 input. Possible options are:
  - Nothing
  - Short Press
  - Long Press
  - Short & Long Press
  - Edge
- ✓ **LOGICAL FUNCTIONS** → When enabled, the output **"status"** object is internally linked with the "1 bit" logical function data entry.

  Possible options in this case are:
  - Data (1bit)1
  - …… ……
  - …… ……
  - Data (1bit)16

Configuration Example: *Consider a facility where Output 1* ***[O1]*** *is connected to a spot light.*

Inputs 1 and 2 ***[I1]*** *&* ***[I2]*** *control Output 1 with short presses:* ***Input 1*** *to turn off the light, and* ***Input 2*** *to switch alternatively the spot light (ON-OFF-ON).*

When input 2 is pressed, light will turn ON. A bit later, by pressing input 1 light will turn OFF. But….how about if we push Input 2 again?…., this will send an OFF order again (as the last value sent was ON) and the light spot in this case will do nothing.

To avoid this erroneous behaviour, an internal link between the Output 1 status object and the Input 2 short press should be set, this way when the light spot is turned off from Output 1, Output 2 will be feedbacked solving the above mentioned problem.

2.2.2.2. TIMERS

This section is meant to control the outputs by mean of a timer.

Two different timers can be selected:

- **Simple Timer:** Through the object "**Timer**".
- **Flashing:** Through the object "**Flasing**".

Note: *Both timers (Simple and Flashing) work independently from each other as from the "**On/Off**" control, since these are all managed from three different Communications objects.*

*By sending an ON order to the object "**Timer**", a scheduled ON begins. If before the time for the ON comes to its end a new OFF order is sent to the object "**ON/OFF**", the output will turn off finishing the previous set timer.*

- **SIMPLE TIMER** → This is applied in the output when an ON or OFF order is received through the objet **"Timer"**.
  - ✓ **ON DELAY**→ Time to pass since the ON order is sent (through the object **"Timer")** and the (ON) response in the output takes place.
  - ✓ **OFF DELAY**→ Time to pass since the OFF order is sent (through the object **"Timer")** and the (OFF) response in the output takes place.
  - ✓ **ON DURATION**→ Time the output remains ON before recovering the OFF status. A "**0**" set in this field means the Output will remain always ON, no timing is applied in the output.

Note: *Sections mentioned above are detailed next:*

*- By sending a "**1**" to the object" **Timer**", an order to apply the **"On Delay"** and the **"On Duration"** in the corresponding output is sent.*

*- By sending a "**0**" to the object"**Timer**", an order to apply the **"Off Delay"** in the corresponding output is sent.*

  - ✓ **MULTIPLY**→ Consists in multiplying a timer as many times as a "**1**"or a "**0**" is received through the object "**Timer**".

Note: *The way the multiply parameter works is detailed next:*

- <u>No Multiply:</u> When value "1" is received by object "Timer" while a simple ON timer is still running, the ACTinBOX resets its counter to 0 to restart counting again. The same happens with "0" and timer off.

- <u>Multiply:</u> When value "1" is received by object "Timer"while a simple ON timer is still running, the ACTinBOX will count double time. If another "1" is received before the ON timer ends, the ACTinBOX will count triple time and so on…. The same happens with "0" and timer off.

- **<u>FLASHING</u>** → This function is meant for the output to run the sequence ON-OFF-ON-OFF…. when needed
  - ✓ **<u>ON DURATION</u>**→ Set the On duration time in the sequence.
  - ✓ **<u>OFF DURATION</u>**→ Set the off duration time in the sequence.
  - ✓ **<u>REPETITIONS</u>**→ Set the number of repetitions in the sequence. For an unlimited value, set **"0"** in this field.
  - ✓ **<u>STATUS AFTER LAST REPETITION</u>**→ To set the output status after the last repetition in the sequence took place.

***<u>Note:</u>** Flashing starts by sending a "**1**" to the object "**Flashing**", and stops by sending a **"0"**.*

2.2.2.3. SCENES

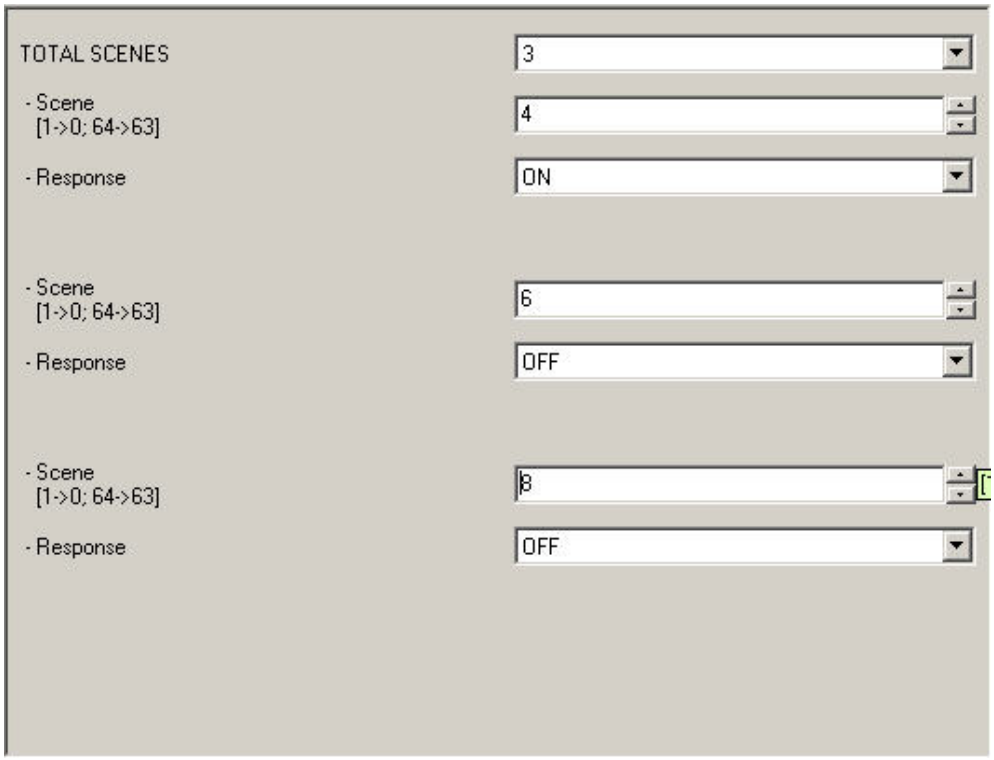
This function makes possible to use a standard (1 byte) scene object to control the outputs.

- **<u>Nº OF SCENES</u>** → To choose the total number of scenes to be used; up to a maximum of five scenes can be set in this field.
- **<u>SCENE</u>**→ Set the Scene number.
- **<u>RESPONSE</u>→** To set the exact output response (On / Off) when the corresponding scene number is called from the BUS.

***<u>Example:</u>** Consider a facility where 4 scenes will be used (**4, 6, 8 & 9**), but only three of the outputs willl be controlled from the ACTinBOX CLASSIC*

- **Scene 4** → *ON*
- ***Scene 6*** → *ON*
- ***Scene 8*** → *OFF*

Parameterization of the device in this case should remain as follows



2.2.2.4. BLOCK

This function makes possible to **"block"** an output by disabling its control. Output will be blocked by sending a "**1"** to the corresponding object in the channel.

Note: *Only the Alarm function is higher priority than the block one. This means that , if output in the channel is blocked and the Alarm goes On, the output shall be carried to the Alarm parameterized position. When Alarm goes off, the output is blocked again. To unblock the output it's necessary to send a* ***"0"*** *to the object* ***"Block"****.*

2.2.2.5. ALARM

This function is designed for cases in which the ACTinBOX must response to an alarm situation. When an alarm occurs, this function sets the output in the previously parameterized position, and after this, output is blocked (it won't be controllable until Alarm goes off).

- **TRIGGER VALUE** → Set the value to activate the Alarm status. An Alarm status will be activated when the value set in this field is sent to the object Alarm (or Alarm 2). The opposite of the **"trigger"** value is the **"Passive"** value.

- **CYCLICAL MONITORING** → To be sure that the sensor works properly and the alarm is not activated, the sensor to send the **"Trigger Value"** to the ACTinBOX, shall continuously send the **"Passive Value"** to the BUS when there is no alarm active. It

is in this case that this parameter should be enabled; this way, if the ACTinBOX doesn´t get the **"passive value"** during the parameterized

- **CYCLE TIME**, alarm will be automatically activated

Note: *It's recommended to set a time higher than double sensor cycle time, to avoid alarm frames missing.*

- **RESPONSE** → Set the response of the actuator channel output when the alarm is activated. When the **"Flashing"** response is set in this field, new parameters will appear in the ETS parameterization environment:
  - ✓ **ON DURATION**→ Set the On duration time in the sequence.
  - ✓ **OFF DURATION**→ Set the Off duration time in the sequence.
  - ✓ **REPETITIONS**→ Set the number of repetitions in the sequence. For an unlimited value, set **"0"** in this field.
  - ✓ **STATUS AFTER LAST REPETITION** → To set the output status after the last repetition in the sequence took place.
- **DESACTIVACIÓN →** Two different methods to activate an alarm:
  - ✓ **NORMAL**→ By sending the **"Passive value"** (opposite to the Trigger one) to the object **"Alarm".**
  - ✓ **FROZEN**→ Consists in applying the normal method to later send a **"1"** to the object "**Unfreeze alarm**". This method makes the channel output remains blocked even when the alarm status is finished; in this case it will be necessary then that the output is manually enabled from another point in the installation.
- **END** → This parameter sets the output response when the alarm status is finished.

2.2.2.6. START UP CONFIGIGURATION

This function is meant to define the behaviour (ON / OFF) of the channel output after a BUS Power Failure, or after programming the device with the ETS.

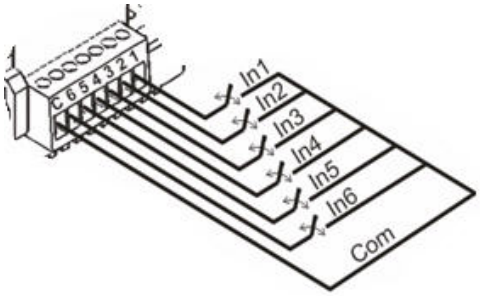
- **INITIAL POSITION** → This field is to define the exact initial position for the channel output after a BUS Power Failure. After programming the device with the ETS, option **"current position"** means output OFF (relay open) for normally open outputs, and output OFF (relay closed) for normally closed outputs.

- **UPDATE** → By enabling this field, the output initial status signal can be sent to the BUS to feedback the rest of devices in the installation when needed.

- **START UP SENDING DELAY**→ As some devices in the installation may take longer to restart after a Power Failure, this field allows setting a delay in seconds for the initial configuration to be sent to the BUS, in order to ensure the rest of devices in the installation are ready to receive the corresponding telegrams.

3. 3. INPUTS

The **ACTinBOX CLASSIC** has 6 **Voltage Free contact Inputs** to be individually configured. Every input can be connected to a **"Push Button"** or a **"Switch/Sensor"**.



Following functions can be selected through a **Push Button input type**:

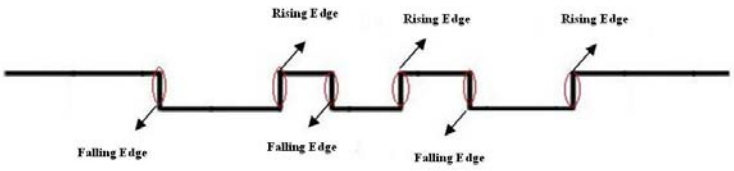
- **« 0 / 1 Sending »:** To send "1 bit" values to the BUS.
- **Shutter Control:** This function results on sending a **"1 bit"** object to the BUS in order to control shutters.
- **Dimmer Control:** This function results on sending a **"4 bits"** Dimming Control Object to the BUS.
- **Scene Sending:** This function results on sending a **"1 byte"** Scene Control Object to the BUS; a scene on the BUS, can be then managed from the input through this Object.

It is possible for the CLASSIC to carry out a function with a short press and another different function with a long press. The CLASSIC then can control up to 12 different functions through its inputs.

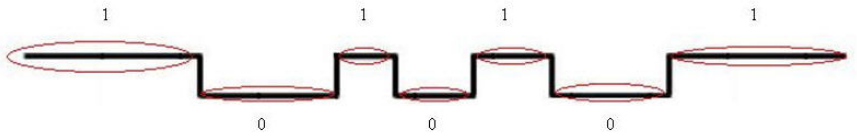
> ***Example:*** *Input 3 can control a light with a short press and run a scene with a long press.*

For a **sensor input type** the following function is available:

- **« 0 / 1 Sending »:** For every rising/falling edge detected on an input, user can select whether to send a "**0**", a "**1**", or an alternative switching of **"0"** and **"1"**



Sensor/Switch → *Detects the transition of the digital signal from low to high (rising edge) or from high to low (falling edge)*



Push Button → *Detects the status of the pulse*

3.1. PUSH BUTTON

A Push Button connected to an input consists of an electrical mechanism which keeps its contacts open under normal conditions. While keeping it pressed, contacts remain closed to later recover its normal position when released.

Depending on the Threshold Time, two different actions can be distinguished:

- **Shot Press**
- **Long Press**

Note: *Normally closed Push buttons cannot be connected to the ACTinBOX inputs.*

3.1.1. "0/1" FUNCTION

This function results on sending 1 bit to the BUS

- **RESPONSE:** This parameter sets whether the value sent is **"0"**, **"1"** or an alternative switching between **"0"** y **"1"**.
- **CYCLICAL RESPONSE SENDING:** Users can ask the ACTinBOX to periodically send **"0"** or **"1"**, or even both (when "always" is selected).
  - ✓ **CYCLE TIME:** Set the time between two consecutive telegrams when using the Cyclical Sending.
- **INTERNAL LINKS** → Through this parameter, users can internally link input objects with other objects in the own ACTinBOX. No group addressing process is needed when internal links are used.

Note: *Internal links will only work when the corresponding sections involved in the link are enabled, this means that an internal link with an* ***"output 3"*** *object will not work if channel B has not been previously enabled as* ***" individual outputs".***

3.1.2. SHUTTER FUNCTION

This function results on sending 1 bit to the BUS in order to control shutters

- **Response**: The corresponding control object can be used to:
  - ✓ **Raise:** Raise the shutter. "0" is sent to the BUS.
  - ✓ **Lower:** Lower the shutter. "1" is sent to the BUS.
  - ✓ **Raise/Lower (toggle switch):** Alternative switching between the Raise/Lower orders (to manage the shutter with an only input).
  - ✓ **Stop/Step Up:** Stops the shutter ("0" is sent to the BUS). When talking about shutters with lamellas, this mode also allows users to control them by pulling lamellas a step up.
  - ✓ **Stop/Step Down:** Stops the shutter ("1" is sent to the BUS). When talking about shutters with lamellas, this mode also allows the user to control them by pulling lamellas a step down.
  - ✓ **Stop/Step (toggle switch) :** Stops the shutter. When talking about shutters with lamellas, this mode also allows users to control them; this parameter alternatively switches the lamellas up/down steps.

Note: *When no directional lamellas are present, any of the 3 last options will "stop" the shutter.*

Note: *When up/down (toggle) option is selected for a short press, the user will not be able to stop the shutter with another short press on the same input.*

- **INTERNAL LINKS** → This parameter is meant to internally link the corresponding input object to the prefered ACTinBOX Shutter Channel (or even both), no group addresses are necessary to make internal links work.

Note: *Internal links will only work when the corresponding sections involved in the link are enabled; this means that an internal link with "**Channel A**" will not work unless channel A has not been previously enabled in the **"Outputs"** section*

3.1.3. DIMMER FUNCTION

This function results on sending a (4 bits) Dimming Control Object to the BUS.

- **Response :** Depending on the chosen option, the Control Object may be:
  - ✓ **Light ON:** Turn the light ON. "1" is sent to the BUS.

- ✓ **Light OFF:** Turn the light OFF. “0” is sent to the BUS
- ✓ **Light ON/OFF (toggle):** Alternative switching between the ON/OFF orders (to manage the lighting level with an only input).
- ✓ **Brighter:** Depending on the “Dimming Step” set, every press on the input will make the brightness level go up. The first short press on the input will increase the brightness level ; a second press stops the “Increase”
- ✓ **Darker:** Depending on the “Dimming Step” set, every press on the input will make the brightness level go down The first short press on the input will decrease the brightness level ; a second press stops the “Decrease”
- ✓ **Brighter/Darker (toggle):** Alternative switching between the Brighter/Darker orders.
- ✓ **Dimming Step:** Depending on the value selected, different brightness levels are offered. Once the “DIMMER CONTROL” option is selected, it is necessary to set this parameter to fix the increasing/decreasing percentage level in every dimming step.

Dimming Step	Necessary button presses for a complete regulation (0 – 100%)
6. 100%	1
5. 50%	2
4. 25%	4
3. 12.5%	8
2. 6.25%	16
1. 3.1%	32
0. 1.5%	64

3.1.4. SCENE FUNCTION

This function results on sending a (1 byte) Scene Control Object to the BUS; a scene on the BUS can be managed with an input through this object.

- o **RESPONSE:** Choose whether the scene will be “Run” or “Saved”.
- o **SCENE:** This parameter identifies the scene to Run/Save with the corresponding input.

- **INTERNAL LINKS** → This parameter is meant to internally link the corresponding input object to the **"Scene"** objects in the **ACTinBOX** outputs.

3.1.5. ADDITIONAL CONFIGURATION

- **THRESHOLD TIME →** This parameter defines the time limit where a short press turns into a long press. If a press on the display ends before the long press time, then it is a short press. This value must be set with precision to tenths of a second (i.e. to get "0.5" seconds, set "5")

- **RESPONSE DELAY →** This parameter sets the time to wait for the object to be sent to the BUS since the action on the input took place. This value must be set with precision to tenths of a second (i.e. to get "1" second, set "10").

  To get an immediate sending (no delay), set value "0" in this field.

- **BLOCK →** By selecting "Yes" on this field, a new Communications object "Block" appears on the ETS. This object can be used to disable an input:

  - ✓ On receiving a "1" through this object, the QUAD will be ignoring any press on the input.
  - ✓ On receiving a "0" through this object, the input recovers its enabled status, so that it is ready again to receive orders. Actions taken on the input while being disabled will not be taken into account.

3.2.SWITCH / SENSOR

A Switch/Sensor connected to an input, consists of an electrical mechanism which may have its contacts open or closed under normal conditions. These mechanisms don't recover their normal position automatically as with the push button.

A transition of a digital signal from low/high/low is called "Edge":

- **Falling Edge:** Closed contact to Open Contact.
- **Rising Edge:** Open contact to Closed Contact.

By selecting a Switch/Sensor input, "[Switch/Sensor] Edge" Communication object will be sent to the BUS every time a rising/falling Edge is detected

- **FALLING EDGE →:** Set the value to be sent to the BUS in the transition of the digital signal from high to low
- **RISING EDGE →:** Set the value to be sent to the BUS in the transition of the digital signal from low to high

- **SENDING OF "0" DELAY→** Time for the ACTinBOX to wait before sending value "0" through the "[Switch/Sensor] Edge" communication object when this value has been detected on an incoming edge.

- **SENDING OF "1" DELAY →** Time for the ACTinBOX to wait before sending value "1" through the "[Switch/Sensor] Edge" communication object when this value has been detected on an incoming edge.

- **PERIODICAL SENDING OF "0" →** Set a period of time to cyclically send value "0" to the BUS when object "[Switch/Sensor] Edge" detects this value on an incoming edge. If not cyclically sending is needed, please select value "0" in this field.

- **PERIODICAL SENDING OF "1" →** Set a period of time to cyclically send value "1" to the BUS when object "[Switch/Sensor] Edge" detects this value on an incoming edge. If not cyclically sending is needed, please select value "0" in this field.

- **BLOCK →** By selecting "Yes" on this field, a new Communications object "Block" appears on the ETS. This object can be used to disable an input:

    - On receiving a "1" through this object, the ACTinBOX will be ignoring any Edge on the input.

    - On receiving a "0" through this object, the input recovers its enabled status, so that it is ready again to receive orders.

  Actions taken on the input while being disabled will not be taken into account.

- **INTERNAL LINKS →** This functionality allows complete independence between different sections in the ACTinBOX. This way, setting an internal link is like if an internal group address is associated with the corresponding enabled function.

4. LOGICAL FUNCTIONS

This section in the **ACTinBOX** is meant to perform **binary logic operations with incoming data from the BUS**, to send the RESULT through other Communication Objects specifically enabled in the actuator for this operation. These Functions work with two different types of data:

- **BUS**, through special Communication Objects enabled for these functions.

- **Internal variables**, to store the intermediate partial operation RESULTS.

- **LOGICAL FUNCTIONS SELECTION** → up to five different logical functions can be enabled.
    - FUNCTION 1……5
- **TOTAL DATA ENTRY OBJECTS** → It is necessary to define the number of Data Entry Objects of each type necessary to be used in all functions.
    - **1 BIT (16 available objects)** → It is necessary to previously define the number of 1 bit objects to be used as data entry in the function operations.
    - **1 BYTE (8 available objects)** → It is necessary to previously define the number of 1 byte objects to be used as data entry in the function operations.
    - **2 BYTES (8 available objects)** → It is necessary to previously define the number of 2 byte objects to be used as data entry in the function operations.

Note I: *Also available as internal variables to store partial results in the operations:*

- 16 "1 bit" variables
- 8 "1 byte" variables
- 8 "2 bytes" variables

Note II: *It is necessary to previously define by parameter the number of data entry objects to be used in the functions before these appear on the ETS environment.*

Note III: *It is always recommended to define more data entries than needed, as a later redefinition involves the deletion of the possible Group addresses association previously made; with the consequent loss of time to re-associate them all again.*

4.1. CALL

This section is meant to select the objects to trigger the FUNCTION execution. Up to 8 different objects may be selected.

Note: *For the FUNCTION to be executed, it will be necessary that at least one of the enabled objects in this section is updated. It is not necessary that the objects in charge of triggering the function execution are included in it.*

4.2. OPERATIONS

This section is meant to define the operations to be performed in every enabled FUNCTION. Up to 4 different operations can be enabled

- **OPERATION**→ Enable the corresponding operation
- **TYPE** → 4 different operation types:
    - **Logic**→: 1 bit available logical operations are **ID, AND, OR, XOR, NOT, NAND, NOR y NXOR**. All of them work with 2 different values (except **ID** and **NOT**). These values can be chosen from the available **16 1 bit objects**, and the **16 1 bit internal variables**. In this case, the operation RESULT is also a 1 bit value that can be stored in any of the 16 available 1 bit internal variables.
    - **Arithmetic (1 byte/2bytes (unsigned integer)/2bytes (Floating point)→:** Depending on the chosen type, these operations will work with 1 byte or 2 bytes values. Users can choose among the following arithmetic operations: **ID, ADD, SUBSTRACT, MULTIPLY, DIVIDE, MAXIMUM y MINIMUM**. All of them work with two values (excepto **ID**); these can be chosen from the available objects, variables or a constant value chosen by parameter. The RESULT in the arithmetic opeation will be 1byte or 2 bytes, (depending on the operation). This RESULT can be stored in any of the 8 corresponding variables.

Note: *Arithmetic Operations (2 bytes unsigned integer) work with data in the range (0……65535). Constants set in the corresponding parametrizable field use format 1X (i.e. Value=4000→Parameter=4000).*

Note I: *Arithmetic operations (2 bytes Floating point) work with data in the range (0……120). Constants set in the corresponding parametrizable field use format 0.1X (i.e. Value=22.5→Parameter=225).*

Note II: *If the RESULT in the 2 bytes Arithmetic operations exceed the permitted range, this will be converted to the corresponding limit in the range. Dividing by "0" doesn´t send anything to the BUS.*

- **Comparison (1 byte/2bytes (unsigned integer)/2bytes (Floating point)** →: These operations work with 1 byte or 2 bytes values, depending on the chosen type. Users in this case can choose among the following comparision operations: **HIGHER, HIGHER OR EQUAL, LOWER, LOWER OR EQUAL, EQUAL, UNEQUAL**. All of them work with two values to be chosen hmong the available objects, values or constant values chosen by parameter. The RESULT in the operation is a 1 bit type ("1" si se cumple la comparación y "0" si no se cumple).This RESULT can be stored in any of the 16 1 bit available variables.

- **Conversión (1 bit/1 byte/2bytes (unsigned integer)/2bytes (Floating point)→:** To convert the Communication Objects between formats.

CONVERSION FUNCTION DESCRIPTION

Specific information on the conversion function is detailed next:

- "**CONVERSION**" (1 bit → 1byte)

1bit	1byte
0	00000000
1	00000001

- **"CONVERSION"** (1bit → 2bytes unsigned integer)

1bit	2bytes unsigned integer
0	00000000 00000000
1	00000000 00000001

- **"CONVERSION"** (1 bit → 2 bytes Floating point)

1bit	2 bytes Floating point
0	0
1	0,1

- **“CONVERSION”** (1 byte → 1 bit)

1byte	1bit
0	0
1..255	1

- **“CONVERSION”** (1 byte → 2 bytes unsigned integer)

1byte	2bytes
$00	$00 00
$01	$00 01
…	…
$FF	$00 FF

- **“CONVERSION”** (1 byte → 2 bytes Floating point)

1byte	2 bytes Floating point
0	0
1	0.1
255	25.5

Note: *Conversion limit in this case is 25.5*

- **“CONVERSION”** (2 bytes unsigned integer → 1 bit)

2bytes unsigned integer	1bit
0	0
1..65535	1

- **“CONVERSION”** (2 bytes unsigned integer →1 byte)

2 bytes unsigned integer	1byte
$00 00	$00
$00 01	$01
…	…
$00 FF	$FF
> $00 FF	$FF

- **“CONVERSION”** (2 bytes unsigned integer→ 2 bytes Floating point)

2bytes unsigned integer	2 bytes Floating point
0	0
1	0.1
…	…
1200	120
>1200	120

- **“CONVERSION”** (2 bytes Floating point → 1 bit)

2 bytes Floating point	1bit
0	0
0,1…....120	1

- **“CONVERSION”** (2 bytes Floating point →1 byte)

2 bytes Floating point	1byte
0	0
0,1... 25,5 .	1..255
> 25,5	255

- **“CONVERSION”** (2 bytes Floating point → 2 bytes unsigned integer)

2 bytes Floating point	2bytes unsigned integer
0	0
0.1	1
…	
120	1200
>120	1200

- **OPERATION RESULT**→ To define the variable to store the operation result.

4.3. RESULT

This section is meant to tell the ACTinBOX where to store and what to do with the RESULT obtained in the previous sections.

- **TYPE**→ Choose among 1 bit, 1 byte or 2 bytes (Unsigned integer) / (Floating point).
- **VALUE**→ Set the variable where the RESULT will be stored.

Note: *Please notice that all the storing variables are shared with all the possible functions/operations in the ACTinBOX, this means that a specific variable used to store the RESULT in an operation/function, should not be used to store a different result.*

- **SENDING**→ Set the conditions to send the RESULT to the BUS.
    - **Result is different from last sent**→: The RESULT will be sent every time the final RESULT in the operations changes.
    - **Whenever the function is executed**→: The RESULT will be sent every time the FUNCTION is executed.

*Note: This parameter is related with section **CALL** (see pag Nº 35); actualy the RESULT will be sent every time the FUNCTION is executed, but the FUNCTION will only be executed when at least one of the enabled objects in the section CALL is updated.*

    - **Periodical sending**→: The result will be periodically sent depending on the time set in the **CYCLE TIME** field.

- **RESTRICTION**→ The sending of the **1 bit functions RESULT** can be restricted to ("**0**" or "**1**"). **1 byte and 2 bytes functions RESULT** sending can be also restricted depending on the following options:
    - Values equals reference one
    - Values not equal to reference one
    - Values lower than reference one
    - Values higher than reference one
    - **Reference Value**→: For the **RESULT Type = 1 byte**, possible reference value range is **[0…..255]**. For the **RESULT Type = 2 bytes**, possible reference value range is **[0……65535]**

- **DELAY**→ Time to pass before sending the RESULT to the BUS. If no delay is needed please set value “0” in this field.

- **INTERNAL LINKS**→ To internally link the function result with the rest of the objects in the ACTinBOX.
    - **Outputs**→: When RESULT is 1 bit type, this can be sent to any 1 bit objects in the outputs.
    - **Shutter Channel**→: When RESULT is 1 bit type, this can be sent to any shutter channel control object.
    - **Inputs**→: When RESULT is 1 bit type, this can be sent to any shutter channel control object.

- **Logical Functions**→: The function RESULT can be always sent to any of the Logical Functions data entry objects (this way, the result can be also used by any other function in the actuador).

5. COMMUNICATION OBJECTS

Communication Objects in the Logical Function section can be two types:

- **DATA**→ Data coming from the BUS, these are the data the operations work with.
- **RESULTS**→ These are the Functions RESULTS. Depending on its size, these are divided in 3 types: 1 bit, 1 byte and 2 bytes.

5.1.NOMENCLATURE:

- **DATA OBJECT TYPE**

  [LF] Data ("size") "X" where size can be 1 bit, 1 byte or 2 bytes; and "**X**" is the Data number (1…..16 for 1 bit data, 1…..8 for 1 byte & 2 bytes).

- **RESULT OBJECT TYPE**

  [LF] RESULT FUNCTION "X" ("size"). Where size can be 1 bit, 1 byte or 2 bytes (depending on the data function result), and "**X**" is the function number (1….5).

- **INTERNAL VARIABLES**

  b1,……, b16 (1 bit type)

  n1,……, n8 (1 byte type)

  x1,……, x8 (2 bytes type)

ANNEX I: COMMUNICATION OBJECTS

SECTION	NUMBER	SIZE	IN/OUT	FLAGS	VALUE			NAME	DESCRIPTION
					RANGE	1º TIME	RESET		
INDIVIDUAL OUTPUTS	0-3	1bit	I	W	0/1	Any	Any	[Sx] ON/OFF	N.O. (0=Open ¨Relay; 1=Close) N.C. (0=Close Relay; 1=Open)
	4-7	1 bit	O	R-T	0/1	Parameter	Parameter	[Sx] Status	0=Output OFF; 1=Output ON
	8-11	1 bit	I	W	0/1	Any	Any	[Sx] Timer	0=OFF Timer;1=ON Timer
	12-15	1 bit	I	W	0/1	Any	Any	[Sx] Flasing	1=Flasing; 0=Flasing end
	16-19	1byte	I	W	0-63 128-192	Any	Any	[Sx] Scenes	0-63(Scene. 1-64);128-191(Save.)
	20-23	1 bit	I	W	0/1	0	Previous	[Sx] Block	1=Block; 0=Unblock
	24-27	1bit	I	W	0/1	Parameter	Previous	[Sx] Alarm	1=Alarm; 0=No Alarm 0=Alarm; 1=No Alarm
	28-31	1 bit	I	W	0/1	Any	Any	[Sx] Frozen	Alarm=0+Froz.=1 -> Alarm end
SHUTTER CHANNEL	32-33	1 bit	I	W	0/1	Any	Any	[Cx] Raise / Lower	0=Raise shutter; 1=Lower
	34-35	1bit	I	W	0/1	Any	Any	[Cx] Stop	0 ó 1 = Stop Shutter
								[Cx] Stop/Step	0=Stop/step up; 1=Stop/Step down
	36-37	1byte	O	R-T	0/1	0	Calculate	[Cx] Current Position	0=0%=Up; 255=100%=down
	38-39	1byte	I	W	0-255	Any	Any	[Cx] Direct Positioning	0=0%=Up; 255=100%=down
	40-41	1byte	I	W	0-255	Any	Any	[Cx] Scenes	0-63(Scen. 1-64);128-191(save.)
	42-43	1 bit	I	W	0-63 128-192	0	Previous	[Cx] Block	1=Bloquear; 0=Desbloquear
	44-45	1bit	I	W	0/1	Parameter	Previous	[Cx] Alarm	1=Alarm; 0=No Alarm 0=Alarm; 1=No Alarm
	46-47	1bit	I	W	0/1	Parameter	Previous	[Cx] Alarm 2	1=Alarm; 0=No Alarm 0=Alarm; 1=No Alarm
	48-49	1 bit	I	W	0/1	Any	Any	[Cx] Frozen	Alarm=0+Froz.=1 -> Alarm end
	50-51	1 bit	I	W	0/1	Any	Any	[Cx] Reversed Moving	0=Lower Shutter; 1=Raise
	52-53	1 bit	I	W	0/1	Any	Any	[Cx] Direct Positioning	1= Go to position; 0=Nothing
	54-55	1 bit	I	W	0/1	Any	Any	[Cx] Direct Positioning 2	1=Go to position 2; 0=Nothing
	56-57	1 bit	I	W	0/1	Any	Any	[Cx] Save Position	1=Save position; 0=Nothing
	58-59	1 bit	I	W	0/1	Any	Any	[Cx] Save Position 2	1=Save Position 2; 0=Nothing
INPUTS	60-65	1 bit	I	W	0/1	0	Previous	[Ex] Block	1=Input blocked; 0=Free
	66-71	1 bit	I	R-W-T	0/1	0	Previous	[Ex] [Short Press] "0"	Short Press -> Send "0"
								[Ex] [Short Press] "1"	Short Press -> Send "1"
								[Ex] [Short Press] switch	Short Press -> Switching 0/1
								[Ex] [Short Press] Raise Shutter	Short Press -> Send 0 (Raise)
								[Ex] [Short Press] Lower Shutter	Short Press -> Send 1 (Lower)
								[Ex] [Short Press] Raise/Lower Shutter	Short Press -> 0/1 switching
								[Ex] [Short Press] stop shutter/step up	Short Press -> Send 0
								[Ex] [Short Press] stop/shutter/step down	Short Press -> Send 1
								[Ex] [Short Press] Stop Shutter	Short Press -> Switching 0/1
								[Ex] [Short Press] Dimmer ON	Short Press -> Send 1 (ON)
								[Ex] [Short Press] Dimmer OFF	Short Press -> Send 0 (OFF)
								[Ex] [Short Press] Dimmer ON/OFF	Short Press ->Switching 0/1
								[Ex] [Switch/Sensor] Edge	Edge -> Send "0" or "1"
	72-77	4bits	O	R-T	0-15	0	Previous	[Ex] [Short Press] Brightness up	SPress->Up SPress->Stop
								[Ex] [Short Press] Brightness down	SPress->Down SPress-> stop
								[Ex] [Short Press] Brightness up/down	SPress-> +/- Luz; SPress->stop
	78-83	1byte	O	R-T	0-63 128-192	Any	Any	[Ex] [Short Press] Run Scenes	Short Press -> Send 0-63
								[Ex] [Short Press] Save Scenes	Short Press -> Send128-191

INPUTS	84-89	1bit	O	**R-W-T**	0/1	0	Previous	**[Ex] [Long Press] "0"** **...** **[Ex] [Long Press] Dimmer ON/OFF**	Long Press -> Send "0" ... Long Press -> switching 0/1
	90-95	4bits	O	**R-T**	0-15	0	Previous	**[Ex] [Long Press] Brightness up** **[Ex] [Long Press] Brightness down** **[Ex] [Long Press] Brightness up/down**	LPress->Up SPress->Stop LPress->down; LPress->Stop LPress-> +/- Luz; LPress->Stop
	96-101	1byte	O	**R-T**	0-63 128-192	Any	Any	**[Ex] [Long Press] Run Scene** **[Ex] [Long Press] Learn Scene**	Long Press -> Send 0-63 L Press -> Send 128-191
LOGICAL FUNCTIONS	102-117	1bit	I	**W**	0/1	0	Previous	**[LF] Data (1bit) 1** **...** **[LF] Data (1bit) 16**	Binary data entry (0/1) ... Binary data entry (0/1)
	118-125	1byte	I	**W**	0-255	0	Previous	**[LF] Data (1byte) 1** **...** **[LF] Data (1byte) 8**	1 byte data entry (0-255) ... 1 byte data entry (0-255)
	126-133	2bytes	I	**W**	0-FFFF	0	Previous	**[LF] Data (2bytes) 1** **...** **[LF] Data (2bytes) 8**	Temperature data entry ... Temperature data entry
	134-138	1bit	O	**R-T**	0/1	0	Previous	**[LF] RESULT FUNCTION 1 (1bit)** **...** **[LF] RESULT FUNCTION 5 (1bit)**	FUNCTION RESULT 1 ... FUNCTION RESULT 5
	139-143	1byte	O	**R-T**	0-255	0	Previous	**[LF] RESULT FUNCTION 1 (1byte)** **...** **[LF] RESULT FUNCTION 5 (1byte)**	FUNCTION RESULT 1 ... FUNCTION RESULT 5
	144-148	2bytes	O	**R-T**	0ºC-120ºC	25ºC	Previous	**[LF] RESULT FUNCTION 1 (2bytes)** **...** **[LF] RESULT FUNCTION 5 (2bytes)**	FUNCTION RESULT 1 ... FUNCTION RESULT 5
RESET	149	1bit	O	**T**	O	O	O	**Reset 0**	Power comes back->Send 0
	150	1bit	O	**T**	1	1	1	**Reset 1**	**Power comes back->Send 1**

ZENNIO TECHNICAL DOCUMENTATION

¡BECOME A USER!
http://zennioenglish.zendesk.com
TECHNICAL SUPPORT