ekinex
CONTROL YOUR LIVING SPACE

Application manual

KNX pushbutton unit

EK-EH2-TP
6-rocker

Contents

Revision	Changes	Data	Drafted by	Verified by
1.0	First draft	2019-11-04	G. Schiochet	
2.0	Removed all references for EK-EA2-TP. Major updates for the new EK-EH2-TP	2019-12-18	G. Schiochet	

1 Scope of the document

This application manual describes application details for the 4.0 release of the ekinex® KNX pushbutton interface EK-EH2-TP (6 rockers).

The document is intended for the system configurator as a description and reference of device features and application programming. For installation, mechanical and electrical details of the device please refer to the technical description datasheet.

Application manual and application programs for ETS are available for download at www.ekinex.com.

Item	File name (## = release)	Version	Device rel.	Update
Product datasheet	STEKEH2TP_EN_v.2.0.pdf	EK-EH2-TP	2.0	12 / 2019
Application manual	MAEKEH2TP_EN_v.2.0.pdf	EK-EH2-TP	2.0	
Application program	APEKEH2TP##.knxprod	EK-EH2-TP	4.0	

2 Product description

The ekinex® 6-rocker pushbutton unit EK-EH2-TP is an S-mode KNX devices for on/off switching of loads, dimming of lighting devices, control of motor drives or other programmable switching and control functions.

This unit is equipped with an integrated KNX bus communication module and it is designed for wall installation on flush mounting box. The device has two programmable LEDs for each function which can be used for instance as a status signal or orientation nightlight.

The pushbutton unit has to be completed with three-position rocker faceplates with central resting position and a frame (which must be ordered separately). Different styles of rocker faceplates can be fitted (e.g. square or rectangular ones) and combined in a most flexible arrangement.

This device is also fitted with a temperature sensor, whose measurements can be read and used as desired by any KNX device connected to the bus.

The device is powered by the KNX bus line with a 30 VDC SELV voltage and does not require auxiliary power.

Product code	Number and type of rockers	Rocker size	Frame
EK-EH2-TP	6 square 4 square + 2 rectangular 2 square + 4 rectangular	Square 40x40 mm Rectangular 80x20 mm	Form or Flank series

i **Note on mounting screws**

The screws for the metal support must be tightened with a max. torque of 1.0 Nm.

2.1 Completion of the device

For full installation and operation, an ekinex® pushbutton must be completed with a separate ordered set of:

- A set of 6 rocker faceplates;
- A frame from the ekinex® *Form* or *Flank* series.

Additional required parts, all of which are supplied with the device, are:

- Metallic support
- Fixing screws (two pairs)
- Protection lid for temporary construction site installment
- KNX terminal block for the connection of the bus line.

2.2 Rocker placement

Different combinations of rockers of the two types are allowed.

The possible combinations are shown in Fig. 1.

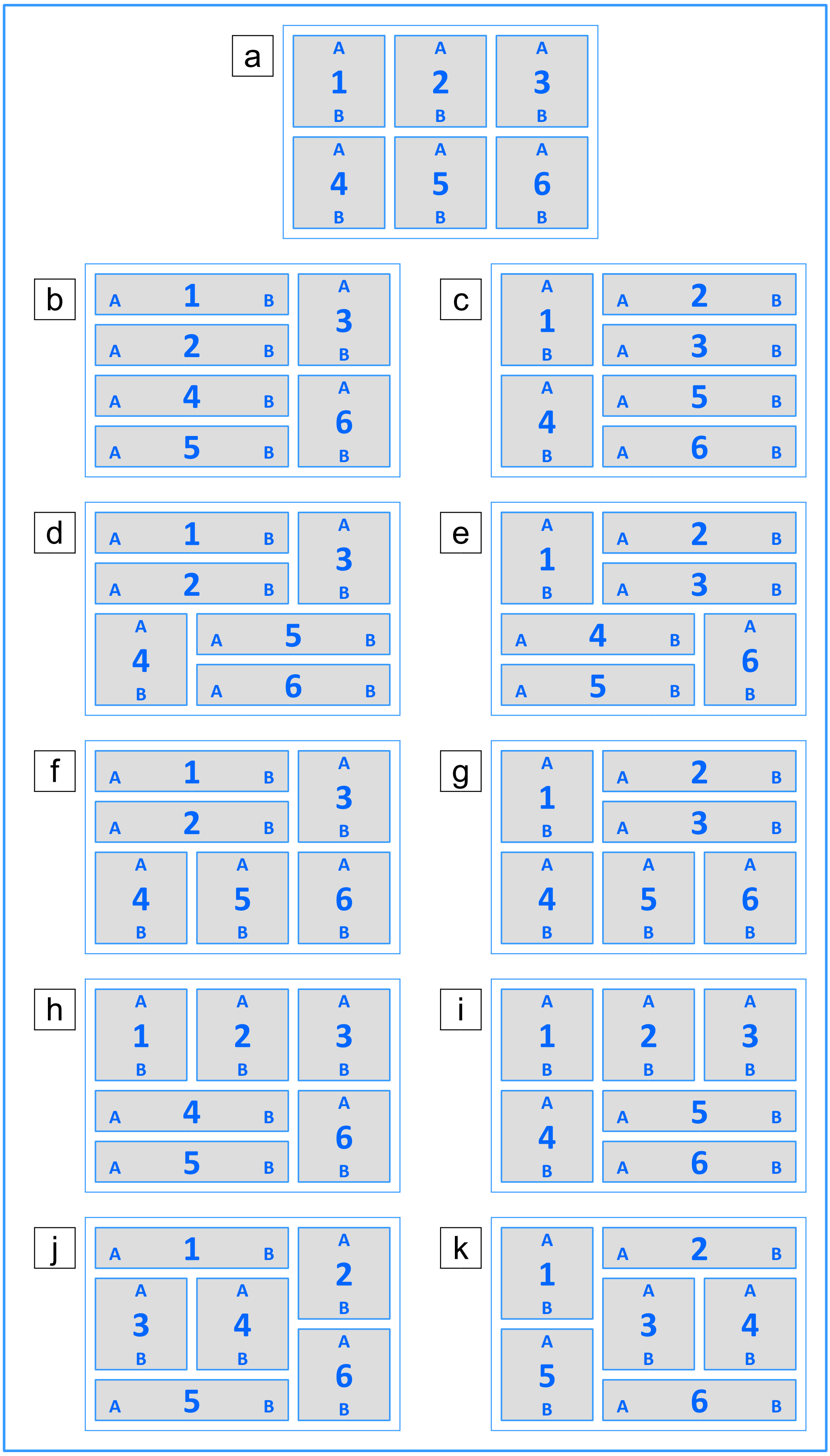


Fig. 1 - Rocker combinations (EK-EH2-TP 6-rocker)

2.3 Rocker functions

Each of the two non-neutral, temporary side positions of a rocker corresponds to an input or *function* (labeled with letters *A* and *B*) that can be activated by pressing the side of the rocker.

When one of the two sides of a rocker is pressed, the device sends on the KNX bus the telegram (or sequence) that has been associated to the corresponding function according to how the device is programmed.

In the most common situation, for instance, one side of the rocker might send an “ON” telegram for a lighting unit, while the other side would send the “OFF” telegram for the same unit. Another typical application would be for one side of the rocker to increase the brightness of a dimmed light (and respectively decrease it for the opposite side), or to raise / lower a curtain or blind and so forth.

The two functions associated with a rocker can also be programmed to perform exactly the same operation, thereby effectively causing one rocker to act as a single pushbutton.

2.4 LED indicators

Each rocker side is equipped with two LEDs, one green and one blue (red and white in other versions), which can be freely programmed e.g. as status feedbacks of the loads or as orientation nightlight.

For a more detailed description of LED position and related settings please refer to the application section of this manual.

2.5 Temperature sensor

The temperature sensor is embedded inside the plastic casing, close to the bottom edge of the front plate.

The sensor is filtered by the software of the device in order to supply a stable measurement of real environmental parameters, avoiding short transitory disturbances.

The rocker plate assembly is arranged in such a way as to prevent that the sensor is obstructed in its function.
Customization of rocker plates

Rocker plates can be customized with predefined symbols and texts; for more information see the standard library on the ekinex® catalog or the website www.ekinex.com. On request, a customization is also possible with symbols and texts chosen by the customer.

3 Switching, display and connection elements

The front side of the device is fitted with mounting hooks for the rocker faceplates; between the hooks, the pushbuttons and the LEDs for status indication are placed.

On the rear side, the device is equipped with a programming pushbutton, a programming status LED and terminals for connecting the KNX bus line.

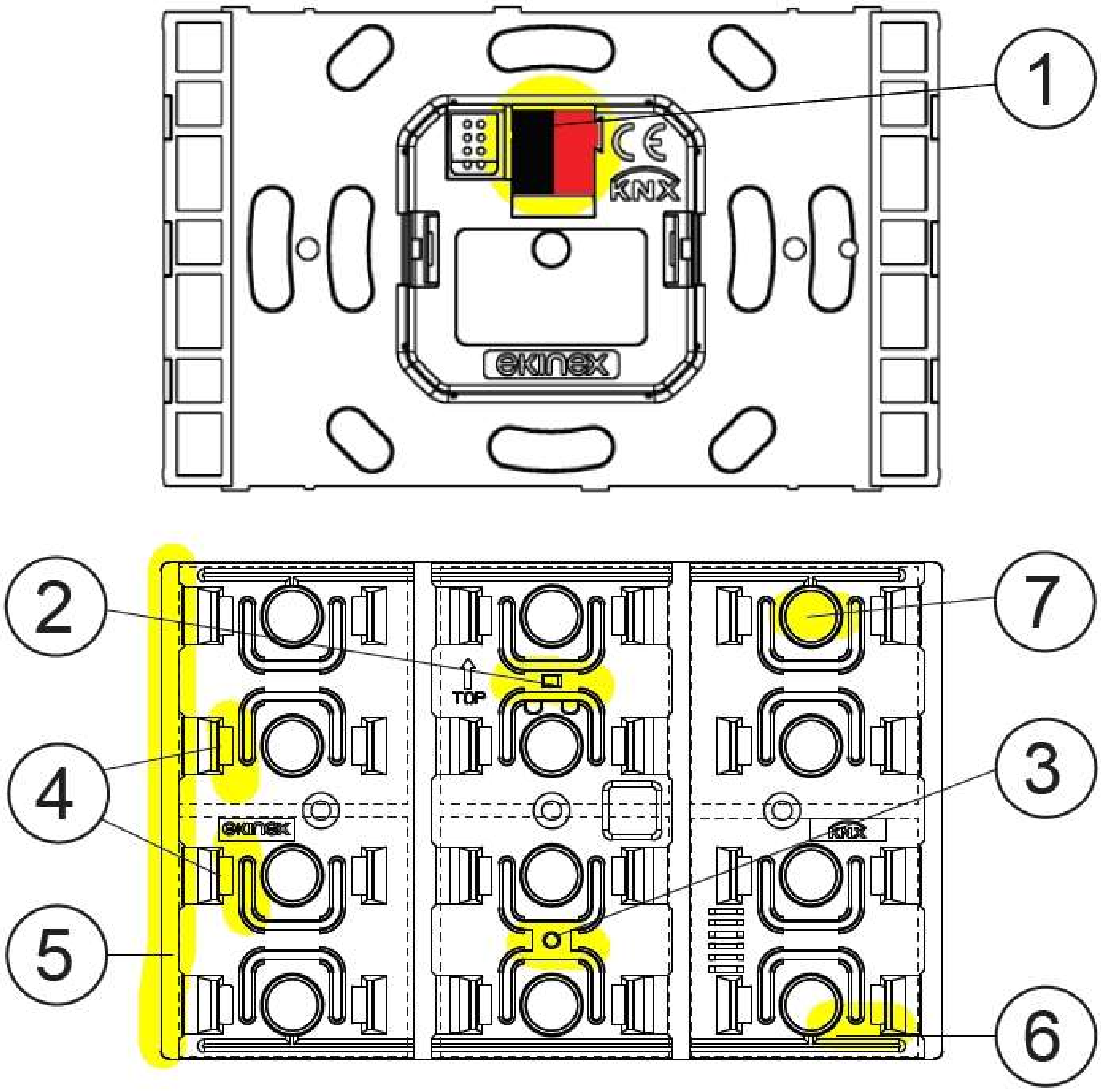


Fig. 2 - Switching, display and connection elements

1. Terminal block for KNX bus line
2. Programming LED
3. Programming pushbutton
4. Rocker faceplate hooks
5. LED diffusers
6. Temperature sensor
7. Pushbuttons

4 Configuration

The exact functionality of the device depends on the software settings.

In order to configure and commission the device you need ETS4 or later releases and the ekinex® application program **APEKEH2TPxx.knxprod**, that can be downloaded from the ekinex website www.ekinex.com.

The application program allows the configuration of all working parameters for the device.

The device-specific application program has to be loaded into ETS or, as alternative, the whole ekinex® product database can be loaded; at this point, all the instances of the selected device type can be added to the project.

For every single device, ETS allows to set the operating parameters separately for each function as described in detail in the following chapters.

The configuration can, and usually will, be performed completely offline; the actual transfer of the programmed configuration to the device takes place in the commissioning phase.

Product code	EAN	No. of channels	ETS application software (## = release)	Communication objects (max nr.)	Group adresses (max nr.)
EK-EH2-TP	8018417181061	12	APEKEH2TP##.knxprod	156	254

i

Configuration and commissioning of KNX devices require specialized skills; to acquire these skills, you should attend training courses at a training center certified by KNX.

For further information: www.knx.org

5 Commissioning

After the device has been configured within the ETS project according to user requirements, the commissioning of the device requires the following activities:

- electrically connect the device, as described in the product datasheet, to the bus line on the final network or through a purposely setup network for programming;
- apply power to the bus;
- switch the device operation to programming mode by pressing the programming pushbutton located on the rear side of the housing. In this mode of operation, the programming LED is turned on steady;
- upload the configuration (including the physical address) to the device with the ETS program.

At the end of the upload, the operation of the device automatically returns to normal mode; in this mode the programming LED is turned off. Now the device is programmed and ready for use on the bus.

6 Function description

After switching on the bus, which also acts as a power supply, the device becomes fully functional after a very short time needed for reinitialization. A delay is programmable for the device to become active on the bus in order to avoid a bus traffic overload during the first moments of startup of the whole network.

In case of a bus power failure (voltage lower than 19 V for 1 s or more), the device becomes unreactive: the timing functions are not active, neither are the programmed group addresses. As soon as the bus voltage is restored, the device will resume operation in its previous state (which is saved on power fail), unless different initialization settings are programmed.

6.1 Offline operation

A fully unprogrammed device does not operate in standby mode. Since the operation relies entirely on the exchange of information through communication objects, there is no part of the device that can operate independently from a KNX bus.

6.2 Online operation

In general the device works like a configurable digital sensor that is listening to own inputs or outputs of other devices. On input events the device performs output functionality over KNX bus like sending values or controlling external devices like KNX actuators.

In the following section, the 4-rocker device will be referenced for the sake of clarity, but all information also applies to the 6-rocker unit, unless explicitly specified.

6.2.1 Software working cycle

The main purpose of the software is following:

- Handle user pushbutton presses and generate bus telegrams according to the assigned functions;
- Implement pushbutton interlock and timing functions;
- Handle incoming bus messages in order to update the status of pushbutton activations and LED indicators;
- Respond to bus messages requesting feedback on the status of the inputs.

The status of the device and specifically of its entities (pushbutton / rocker activation status and LED indicators) relies on KNX *communication objects*, which can be freely defined and bound in various ways to the physical elements of the device.

There are also special events on which it is possible to trigger additional features. These events are the bus failure and recovery, and the download of a new configuration with ETS.

6.2.2 Pushbutton inputs

The press of a pushbutton can be bound to different effects on a state variable.

6.2.2.1 Pushbutton input events

A button press can be handled either as an “on-off” event (“on” means when the button is pushed, “off” when it is released), or as a “short press - long press” event (whereby a time period can be defined to discriminate the duration of the “long” from the “short” press).

In both cases, for each of the two available events a separate action can be assigned that operates on a selected variable (actually, more than one; see below for details).

6.2.2.2 Lock function

For each input (or channel if inputs are coupled, see below), a lock feature can be enabled which allows to block the operation of an input through a message on a communication object.

When in a locked state, the input is effectively disabled.

A value (for each transition) can be specified to be assigned to the communication object upon entering or exiting the locked state.

The locked state can also be automatically activated when the bus is connected.

The locked state can be deactivated by sending another telegram.

6.2.2.3 State variables (Communication objects)

The variable that is changed by the input events can be one of the types available for KNX communication objects, i.e. for instance a 1 bit value (on-off), a 2 bit value or an integer value of larger size.

In all cases, each of the two events can:

- change the value of the variable to one of two definable values within its range (which is trivial in the case of the 1-bit value);
- toggle between the two defined values
- do nothing (value is unaffected)

This state variable, once assigned a group address, is actually a **KNX communication object**; as such, it undergoes the usual rules for communication objects, among which – for instance – the effect of flags to determine how the change of value affects the transmission of the objects.

This obviously implies that, if the value of a communication objects changes due to the effect of a bus telegram, the corresponding change will register in the device, according to its associated flags.

6.2.2.4 Binding between Events and Communication objects

The above description is a little simplified in order to ease comprehension; as a matter of fact, to each event can be assigned not just one, but several communication objects (up to 8), even of different types. Each of these communication objects can have its own behavior and its own associated value set.

6.2.2.5 Repeated send

For most features, is it possible to set the device to send a telegram not just when a value changes as a consequence of an input transition, but also at regular intervals whenever that value setting is active.

This behavior, also referred to as Cyclical Transmission, can be set separately for each of the two values that are associated to an input (or both, or none of them).

If an input is set to “*send values or sequences*” mode, repeated send is not available if more than 1 Communication Object is assigned to that input.

6.2.2.6 Input coupling

The 8 / 12 pushbutton inputs described can be considered, and used, as independent; however, due to the physical structure of the device and the nature of the functions it most frequently performs, these inputs can be naturally grouped in pairs, which in the application program are referred to as *channels*. Each channel is made of a pair of inputs, and is physically associated to a rocker.

Since the channels of the device are labelled 1 to 6 (or 1 to 4), the inputs are labelled 1A / 1B for channel 1, 2A / 2B for channel 2 and so on. The same numbering is used whether the channel pairing is used or not.

In order to specify channel pairings, each rocker can be configured in two ways: single mode and coupled mode. This setting appears among rocker-level settings rather than input-level settings, because only inputs belonging to the same rocker can be coupled. The only combinations allowed for coupling are in fact 1A with 1B, 2A with 2B, and so on.

- In *single mode*, each input operates independently, has its own parameters and communication objects. This is the mode of operation described so far.
- In *coupled mode*, 2 inputs operate logically grouped under a channel in order to perform a common functionality; therefore, they operate on shared communication objects.

It is possible to configure some of the inputs in single mode and the others in coupled mode, with the pairing constraints just described.

It must be mentioned that there is actually a third way to configure an input pair, which lies somehow halfway between the two modes above (although it is considered as a variation of the single mode): each second input, i.e. inputs 1B, 2B, 3B etc., can be configured to perform exactly the same function as its first input. In this fashion, both pushbuttons associated with a rocker are effectively operated “in parallel”, so as to operate the whole rocker as a single, larger control (either pushbutton or switch, according to programmed operation).

Single and coupled modes have a similar functionality, but differ for the configuration.

6.2.2.7 Single Input mode

Each single input can be configured for one of following different features:

1. *Send values or sequences*
   An event triggers the transmission on the bus of configurable values or sequence of values.
   These values can be of a logical type or a numerical type with a different size.
   A sequence of values can be made of up to 8 communication objects of different value types.
   Time delays can set between values in the sequence.
2. *Dimmer control*
   This mode is intended to be used with dimming actuators for the control of lighting devices.
   The functionality is triggered on short press and long press events.
   On short press events, the device sends on/off telegrams to the dimming actuator.
   On long press events, the dimming percentage is varied up or down until the button is released.
3. *Shutter or Venetian blind control*
   This mode is intended to be used together with actuators for the control of motorized blinds, shutters and similar devices. These actuators have functions for blind opening and closing; two movement types are selectable, i.e. continuous movement and stepwise movement. On input events, the device sends operation telegrams to the actuators.
   The operation is configurable through following parameters:
   - If *toggle* mode is enabled, on each activation of the same input the movement direction is inverted; if it is disabled, the movement direction is fixed and it can be set to “up” or “down”.
   - If *blinds* mode is enabled, the device sends “full movement” telegrams on long press and “step” telegrams on short press; if it is disabled, the device sends “full movement” telegrams on long press and “stop” telegrams on short press.

4. *Scene function output*
   This mode is intended to be used together with several KNX actuators that support using a scene function; this function allows to store and recall a communication object value on an actuator.
   In this mode, the role of the device is to send a “store / recall scene” telegram to the actuator on a long / short press event.
   This mode has two possible configurations:
   - Activate preset scene on short press, and store current setting as scene value on long press
   - Activate two different scenes on long and short press.

6.2.2.8 Coupled Input mode

Each pair of coupled inputs, corresponding to the two sides of a same rocker, can be configured for one of following different features (only the differences from the single mode are highlighted):

1. *Switch control*
   Both inputs in a pair are bound to the same communication object; unlike single mode, the object can only be of the 1-bit type (on-off), therefore building a conventional switching behaviour.
   The user can configure which of the two inputs sets the “off” or resp. “on” value.
2. *Dimmer control*
   The functionality is triggered on short press and long press events of the inputs in the pair.
   The user can configure which of the two inputs sets the “up” or resp. “down” value.
   On short press events, the input configured as “up” sends an “on” switching telegram to the dimming actuator; while the “down” input sends an “off” telegram.
   On long press events, the dimming percentage is varied up or down until the button is released.
3. *Shutter or Venetian blind control*
   The two inputs of a pair are assigned to opposite movement directions; these can be assigned to inputs as desired, i.e. A up / B down or the other way around.
   The *blinds* mode can also be set, and it works exactly as in single mode.

In coupled mode there is no provision for a *scene* control feature.

6.2.2.9 Dimmer control function

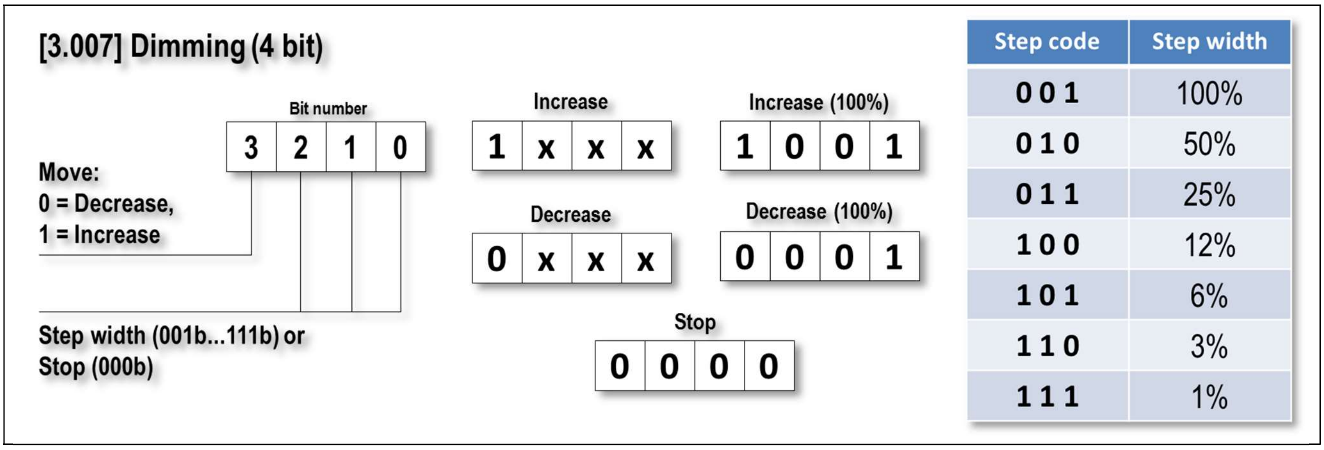
The Dimmer control function is a standard device application profile defined by the KNX specifications. These specifications define several basic requirements regarding interface mechanisms; in addition to that, specific operational details must be defined for the particular device (both for control devices and actuators).

i

All information listed in this section has the purpose of illustrating the operation of the specific device, and are not to be necessarily considered fully comprehensive or realatable to any other device. It is therefore recommended, in order to obtain a complete and generally applicable documentation, to refer to official KNX documents.

For further information please visit the KNX website www.knx.org.

The Dimmer type control is essentially based on a 4-bit Communication Object whose format is illustrated below:



Step code	Step width
0 0 1	100%
0 1 0	50%
0 1 1	25%
1 0 0	12%
1 0 1	6%
1 1 0	3%
1 1 1	1%

The transmission of a telegram containing data with this format instructs the actuator either to effect an increase (or decrease) of the output intensity with the specified amplitude step, ot to stop an ongoing variation.

The increase / decrease are not abrupt but rather progressive; therefore, an increase / decrease with a step width corresponding to the full available range has the effect of starting the intensity variation in the chosen direction. The variation will continue until the maximum (minimum) intensity value is reached, unless it is interrupted at a desired point by sending a telegram with the “stop” value.

It is usually possible, and desired, to be able to switch the load immediately on or off (i.e. take the intensity value abruptly to 0% or 100%). In order to obtain this, another telegram is used which is based on a different communication object of binary “On/Off” type; this is nothing else than the same object used for ordinary load switching, which is normally available even in absence of the dimming mechanism.

The control device – in this case, the pushbutton unit – defines the operations required to originate a sequence of the above described commands, in the desired order and with the proper timing required to obtain the desired command.

The operations and associated commands for the unit are illustrated below:

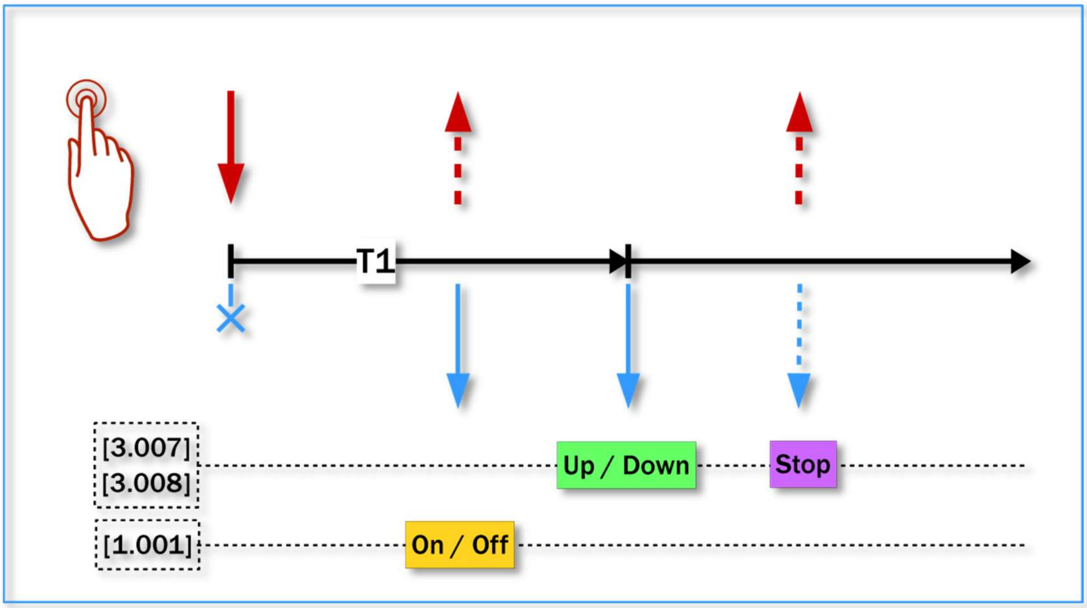


Fig. 3 – Dimmer command sequence

- Short press: immediate switching, on or off (through an on/off value toggle on the switch communication object)
- Long press: Value increase / decrease to 100% (resp. 0%)
- Release: Stop variation

Please notice that the very same mechanism can be applied for the control of shutters and venetian blinds' slats (whereby "higher / lower intensity" should be more properly replaced by "opening / closing"). For this purpose, the Datapoint Type (DPT) 3.008 exists, which has identical structure and values as data described above.

In order to control a shutter or blinds with the same operations, it it therefore possible to "connect" a communication object of type 3.007 on the control side with a corresponding object of type 3.008 on the actuator's side (provided that this is made available by the actuator). In this case, obviously, the "On/Off" switching object for immediate switching is not used.

6.2.2.10 Shutter / Blinds control

The Shutter/Blinds function is a standard device application profile defined by the KNX specifications. These specifications define several basic requirements regarding interface mechanisms; in addition to that, specific operational details must be defined for the particular device (both for control devices and actuators).

i

All information listed in this section has the purpose of illustrating the operation of the specific device, and are not to be necessarily considered fully comprehensive or relatable to any other device. It is therefore recommended, in order to obtain a complete and generally applicable documentation, to refer to official KNX documents.

For further information please visit the KNX website www.knx.org.

In the case of a shutter, the actuator moves a mechanical element from a stroke endpoint to another in a progressive manner, allowing stops in intermediate points; there are two command lines (which may be activated one at a time) that control movement in each of the two directions.

The Venetian blind is basically a shutter which, in addition to the up / down movement, also has a set of louvers or slats which are opened / closed in the same fashion as the main blinds' panel (progressive movement). In standard actuators, both the blinds' panel and the slats are driven through the same pair of lines, which are in common; therefore, in order to achieve a separate setting of the two elements, the lines must be activated in a peculiar sequence. Further details can be found in the documentation of the actuators; suffice to say here that, on the command side, the control sequences are defined regardless to the way the actual device works.

The basic control for a Shutter or Venetian blinds is based on three communication objects (all of 1-bit size):

- [1.008] Move Up/Down
- [1.007] Stop – Step Up/Down
- [1.017] Dedicated Stop

The effect of the commands associated to these object is following:

- The "Move" command, as soon as it is issued, starts the actuator movement in the specified direction;
- The "Stop / Step" command has a dual purpose. If the actuator is at rest, it starts a short movement in the specified direction (the duration is parametrized in the actuator); otherwise, it just stops the current movement with no further action.
- The "Stop" command simply stops whatever movement is currently ongoing, if any.

Usually other type of control objects are made available to allow for alternate controlling mechanisms (Dimmer-style, absolute position etc.), but their description is out of the scope of the pushbutton-based control illustrated in this section. Further details can be found in the manuals of actuator devices or in KNX specifications.

In the most simple style of operation, on the control side:

- for a Shutter control, at least the "Move" and "Stop" objects are needed (and made available);
- for Venetian blinds control, at least the "Move" and "Step/Stop" objects are needed (and made available).

For the sake of completeness it must be noted that, on the actuator side – whether for Shutter or Venetian blinds – the availability of both "Move" and "Stop/Step" objects must be guaranteed, whereas the "Stop" object is optional (albeit nearly always available).

There are many possible variations about the operation of the control device – in this case, a pushbutton unit - in order to generate a command sequence with the proper sequence and timing.

For ekinex pushbuttons, two different modes of operation are available: these are labeled as “Shutter” and “Venetian blinds” according to their main (albeit not exclusive) intended purpose.

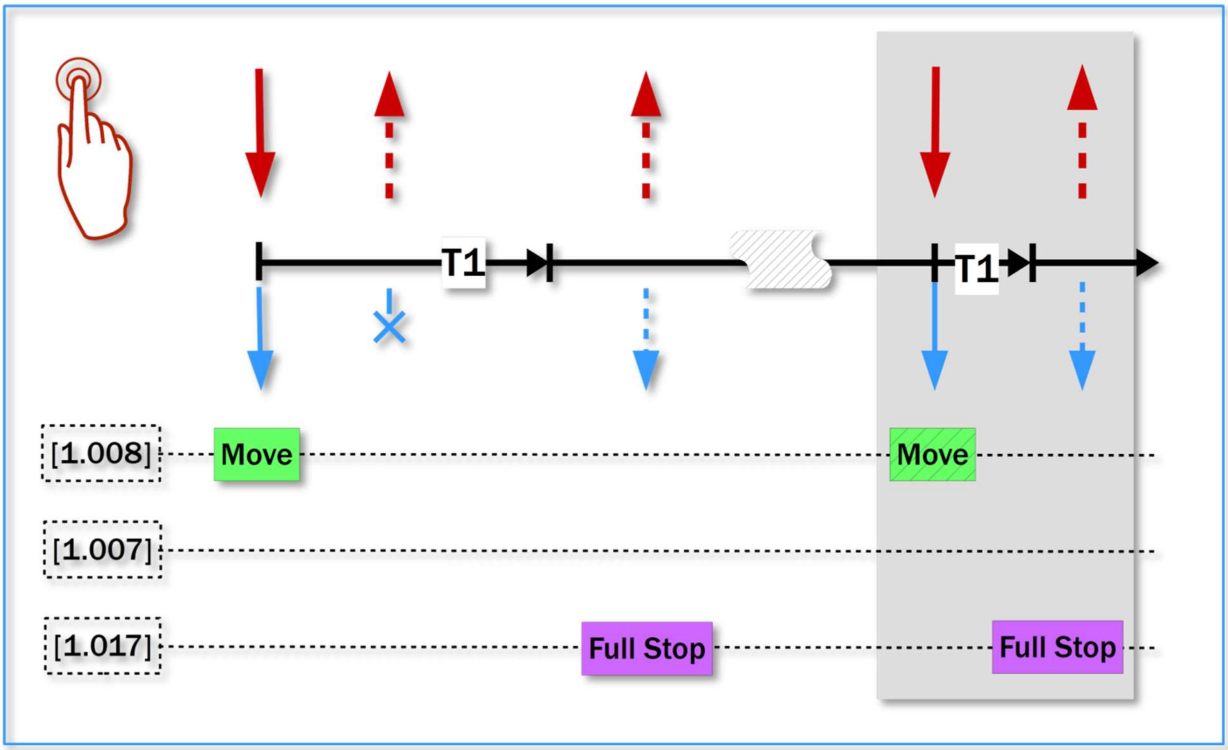


Fig. 4 – Command sequence in “Shutter” operation mode

In “Shutter” mode, a key press – or the activation of a digital input, generally speaking – starts the actuator movement in the corresponding direction. The direction can be either fixed or alternating between different activations, according to how the pushbutton is programmed.

If the button is released quickly, the shutter will continue its movement until the endpoint is reached; if desired, though, it can be stopped at any time with a long press.

If the keypress is longer, the shutter stops as soon as the button is released (which is supposed to happen once the shutter has reached the desired position).

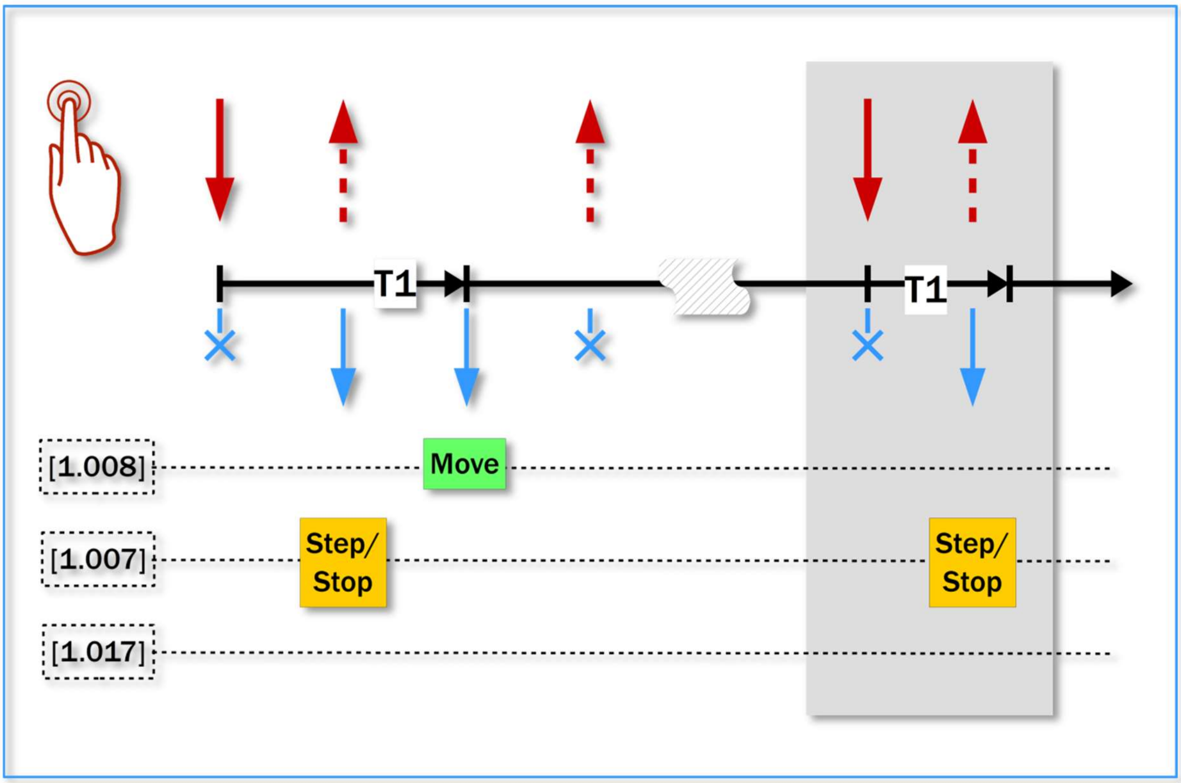


Fig. 5 - Command sequence in “Venetian blinds” operation mode

In “Venetian blinds” mode, on a short key press (upon release, to be exact) the actuator moves by a step duration; this operation is normally (i.e. if the actuator is actually configured for Venetian blinds) used for setting the slats’ position.

If the keypress is longer, as soon as the threshold time for long press is reached a “Move” command is issued; this will take the blinds to the fully opened or closed position. If an intermediate position is desired, the blinds can be stopped with another short press activation.

6.2.3 LED indicators

The LED indicators (one green, one blue) associated with each input can be singularly addressed, even if the corresponding inputs are coupled.

6.2.3.1 General parameters

All LEDs have a common intensity value, which can be set from the bus through a communication object or with a fixed setting from 0 to 100% in 10% steps.

6.2.3.2 Individual parameters

Each LED can be driven in one of following ways:

- Fixed value (always on or always off);
- Lit when the corresponding input is activated. In this option, an additional off-delay can be specified after the button is released;
- Status set from the bus through a communication object. In this case, the LED can be set to be flashing when active (with a choice of different on/off time combinations), and the on/off light status can be inverted with respect to the communication object status (so as to have the LED on when the CO has an “off” value).

6.2.3.3 Technical Alarm indicator

The device has a peculiar indicator feature called *Technical Alarm*: if it is enabled, all LEDs (green and blue) at the four corners of the device can be activated flashing through a KNX bus telegram.

This feature is meant as an indicator for a generic alarm condition, but it can be used in a custom way as the user sees fit.

6.2.4 Temperature sensor

The value from the temperature sensor, unless disabled, can be read from the bus by other devices. In addition, its behaviour can be modified through following parameters.

The raw value read from the sensor can be corrected with a small offset (-5 °C to +5 °C in steps of 0.5 °C), in order to compensate for environmental factors and achieve a better precision.

The value read by the temperature sensor can periodically be sent on the bus with a specified transmission interval, and also whenever a specified variation occurs.

6.3 Device settings

This section lists all configurable parameters and describes related communication objects.

Every channel, and every input or input pair under a channel, offers the same set of communication objects and parameters, but they may all be independently configured.

Hereafter, all channel-specific settings are listed grouped by channel; a generic channel number is referenced as “x” (where x = 1…6), while a generic input is referenced as “xx” (xx = 1A, 1B, 2A, ... 6B).

> i The parameter values highlighted in bold represent the default value.

The device settings are divided in two main groups: the *general* settings and the *channel-specific* settings.

6.3.1 General settings

The parameters in this section define the overall behavior of the device.

Parameter name	*Conditions*	*Settings*
Rocker configuration	-	*See Fig. 3a / 3b* *for available choices*
	Specifies the configuration of installed rocker plates, thereby determining how physical pushbuttons will be associated to logical inputs and coupled in rocker pairs.	
LED position	*EK-EH2-TP only* Rocker configuration = 6 square rockers	*See Fig. 4* *for available choices*
	Specifies the position of LEDs corresponding to each rocker	
Temperature sensor	-	enabled / **disabled**
	Enables the temperature sensor by making the corresponding communication object available.	
Temperature offset	**Temp. sensor = enabled**	-5°C... +5°C (**0°C**)
	Correction offset to be applied to the raw measured temperature value	
Minimum change of value to send	**Temp. sensor = enabled**	enabled / **disabled**
	Minimum change in temperature required to trigger the transmission of a new value.	
Transmission interval	**Temp. sensor = enabled**	hh:mm:ss (**00:05:00**)
	Interval between cyclical transmissions. A value of 00:00:00 disables cyclical transmission.	
LED intensity from bus	-	yes / **no**
	Specifies whether the intensity value of LEDs should be set through a communication object.	
LED intensity	**LED intensity from bus = no**	0%..100% (**50%**)
	Fixed intensity value of LEDs.	
Delay after bus voltage recovery	-	hh:mm:ss.fff (**00:00:04.000**)
	Delay before bus telegrams can be sent after a recovery of the bus voltage. The delay time affects the transmission generated by an event as well as the cyclical transmission. For the cyclical transmission: after the delay time finished, the cycle restarts and the first telegram will be sent after the cycle time.	

Parameter name	*Conditions*	*Settings*
Technical alarm	-	enabled / **disabled**
	Enables a communication objects that activates an alarm indication through a bus telegram. The indication is made by flashing the four LEDs at the corners of the device. *This indication is made available to the user for any purpose he sees fit (not necessarily an actual alarm).*	

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Technical alarm	Technical alarm = enabled	1 bit	C-W--	[1.005] alarm	0
Brightness value	Not used (previous versions only)	2 Byte	CR-T-	[9.004] lux (Lux)	1
LED intensity percentage	LED intensity from the bus = yes	1 Byte	C-W--	[5.001] Percentage (0..100%)	2
Temperature value	Temp. sensor = enabled	2 Byte	CR-T-	[9.001] temperature (°C)	3

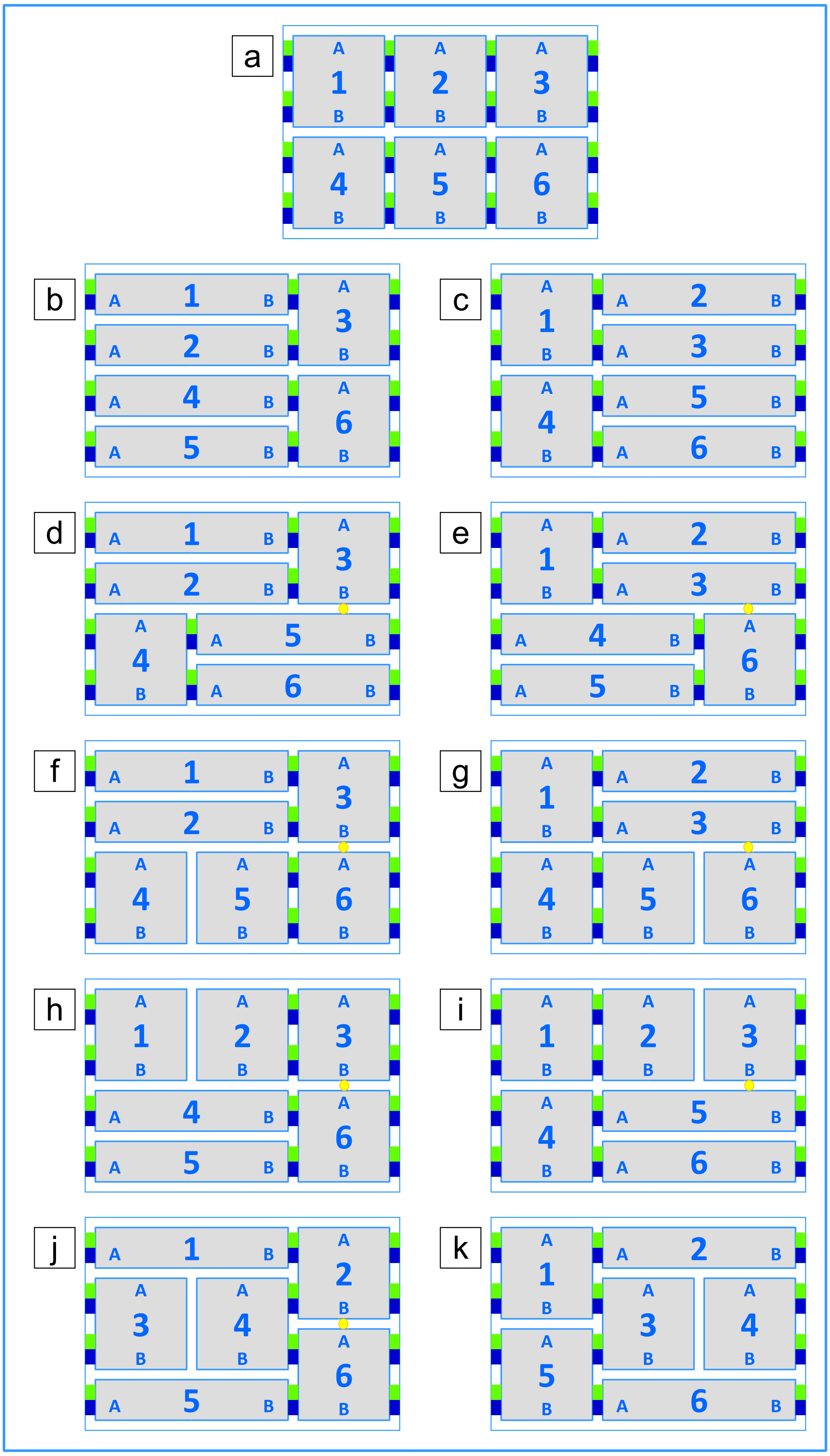

a
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
b
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
c
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
d
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
e
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
f
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
g
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
h
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
i
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
j
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B
k
A 1 B
A 2 B
A 3 B
A 4 B
A 5 B
A 6 B

Fig. 6 - Rocker combinations (EK-EH2-TP 6-function)

Available rocker plate configuration options for EK-EH2-TP 6-rocker unit:

a. 6 square buttons*
b. 4 rectangular buttons on the left
c. 4 rectangular buttons on the right
d. 2 rectangular buttons on top left and 2 on bottom right
e. 2 rectangular buttons on top right and 2 on bottom left
f. 2 rectangular buttons on top left
g. 2 rectangular buttons on top right
h. 2 rectangular buttons on bottom left
i. 2 rectangular buttons on bottom right
j. 2 rectangular buttons on top and bottom left
k. 2 rectangular buttons on top and bottom right

* In this configuration, the position of the LEDs associated with each rocker can be changed; see the corresponding section for details.

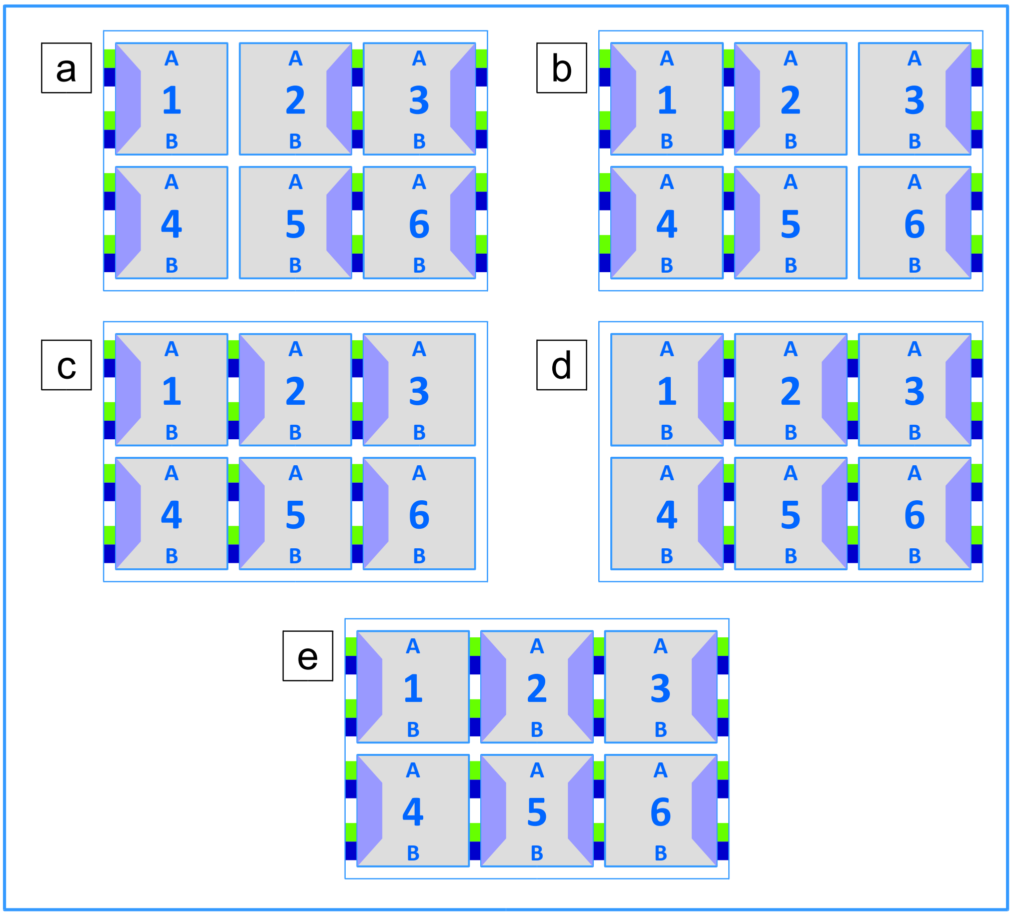


Fig. 7 - Configurable LED positions *for 6 square rocker combination (EK-EH2-TP 6-function only)*

Available options for LED positions for EK-EH2-TP 6-rocker unit and 6 square rocker combination:

a.	Left	Right	Right
b.	Left	Left	Right
c.	Left	Left	Left
d.	Right	Right	Right
e.	Left	Left + Right	Right

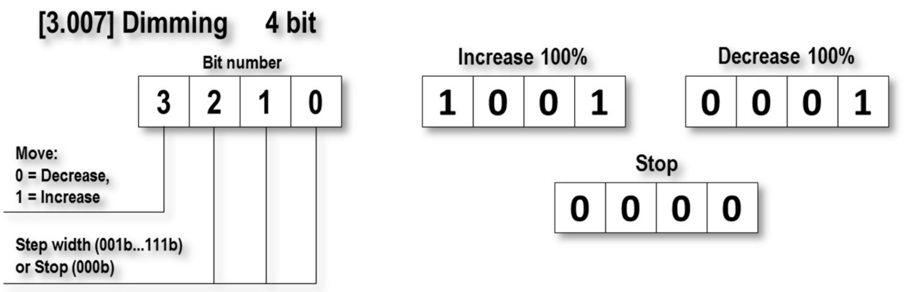
6.3.2 Rockers configuration

Parameter name	*Conditions*	*Settings*
Rocker *x*	-	disabled **independent or single** coupled copy parameters from rocker*
	Set operation mode for inputs corresponding to Rocker x. *The identification of which Rocker and associated input pushbuttons are corresponding to a given number (e.g. 1A – 2B – 4A etc.) is done according to parameter "General / Button configuration".* ** This option is only available for rockers nr. 2 and above. If selected, the corresponding rocker can be made to perform the exact same kind of function as another specified rocker, but basing on different communication objects.* *This allows to spare time in configuring the device, at the same time assuring that there is no inconsistency between two rockers that are meant to be configured in exactly the same way.* *To assign the same configuration is just a shortcut for the selection of configuration options; it is in no way implied that the two rockers share any of the involved communication objects (each rocker has its own independent objects).*	
Function A	**Rocker *x* = independent or single**	**enabled** / disabled
	Enables or disables the capability to generate events for the first pushbutton of the rocker.	
Type	**Rocker *x* = independent or single** **Function A = enabled**	**send values or sequences** dimming shutter or venetian blind scene
	Determines the kind of function performed by the FIRST rocker input. *Further parameters for the selected function will appear in the individual rocker configuration sections (see below).*	
Function B	**Rocker *x* = independent or single**	disabled enabled in parallel with function A, as a single function **copy parameters from function A**
	Enables or disables the capability to generate events for the second pushbutton of the rocker. *If not disabled, the pushbutton can be given an own independent function (enabled),used as an "alias" of the first input (in parallel), or perform the exact same kind of function as first input (copy parameters), but possibly basing on a different communication object.*	
Type	**Rocker *x* = independent or single** **Function B = enabled**	**send values or sequences** dimming shutter or venetian blind scene
	Determines the kind of function performed by the SECOND rocker input. *Further parameters for the selected function will appear in the individual rocker configuration sections (see below).*	
Type	**Rocker *x* = coupled**	switch **dimming** shutter or venetian blind
	Determines the kind of function performed by the FIRST and SECOND rocker input. *Further parameters for the selected function will appear in the individual rocker configuration sections (see below).*	
Rocker to copy from	**Rocker *x* = copy parameters from rocker** (x > 1)	1..4*
	** The values that can be chosen obviously do not include the number of the rocker for which the selection is made.*	

6.3.2.1 Independent or single: send values or sequences

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Switching status *[type]*, object *n*	Rocker *x* = indep. or single Function *x* = enabled Type = send values or sequences	*According to configuration* (**1-bit**)	CRWTU	*According to configuration* (**[1.001] switch**)	5, 22 (1A, 1B) 43, 60 (2A, 2B) 81, 98 (3A, 3B) 119, 136 (4A, 4B) 157, 174 (5A, 5B) 195, 212 (6A, 6B)
	Up to 8 objects can be defined for binding with the same event. *The listed CO numbers are those referring to object nr.1; the COs for each subsequent object are following in sequence.* *To obtain the CO numbers for object number n, just add (n-1) to the listed numbers.* *E.g.: COs associated to input 3A (of Rocker 3) have numbers from 81 to 89. The number of CO nr. 5 is therefore 81+(5-1) = 85.* ***The size and type of the individual objects can be configured as described in following sections***.				

6.3.2.2 Independent or single: dimming

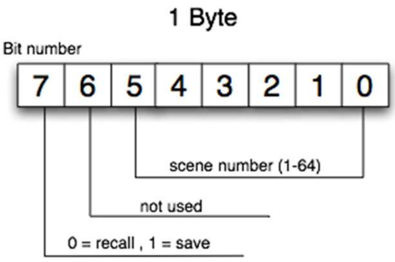
Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Switching command	Rocker *x* = indep. or single Function *x* = enabled Type = dimming	1 bit	CRWTU	[1.001] switch	13,30 (1A, 1B) 51,68 (2A, 2B) 89,106 (3A, 3B) 127,144 (4A, 4B) 165,182 (5A, 5B) 203,220 (6A, 6B)
	Send a command to a dimming actuator to switch the light on or off. *The command is triggered by a short press on the input.* *The value sent can be a fixed value or it can be toggled at each input activation.*				
Rocker *x* – Dimming up / down / stop command	Rocker *x* = indep. or single Function *x* = enabled Type = dimming	4 bit	CR-T-	[3.*] 3-bit control	14,31 (1A, 1B) 52,69 (2A, 2B) 90,107 (3A, 3B) 128,145 (4A, 4B) 166,183 (5A, 5B) 204,221 (6A, 6B)
	Send a command to a dimming actuator to change dimming intensity (brighter or darker). *Three values are used which mean start increase, start decrease or stop the change.*  [3.007] Dimming 4 bit Bit number 3 2 1 0 Move: 0 = Decrease, 1 = Increase Step width (001b...111b) or Stop (000b) Increase 100% 1 0 0 1 Decrease 100% 0 0 0 1 Stop 0 0 0 0 *Increase/decrease values are sent when a long press action occurs and stop value on press release.* *The value sent can be a fixed value or it can be toggled at each input activation.*				

6.3.2.3 *Independent or single: shutter or venetian blind*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Dedicated stop command	Rocker *x* = indep. or single Function *x* = enabled Type = shutter or venetian blind **Blind mode = Disabled**	1 bit	CRWTU	[1.017] trigger	13,30 (1A, 1B) 51,68 (2A, 2B) 89,106 (3A, 3B) 127,144 (4A, 4B) 165,182 (5A, 5B) 203,220 (6A, 6B)
	Immediately stop any movement of the blind. The object is sent on release after a long press (if the venetian blind mode is disabled).				
Rocker *x* – Stop – step up/down command	Rocker *x* = indep. or single Function *x* = enabled Type = shutter or venetian blind **Blind mode = enabled**	1 bit	CR-T-	[1.007] step	16,33 (1A, 1B) 54,71 (2A, 2B) 92,109 (3A, 3B) 130,147 (4A, 4B) 168,185 (5A, 5B) 206,223 (6A, 6B)
	Increase or decrease the opening of the blind stepwise, or interrupt an ongoing movement. *The object is sent on release after a short press (if the venetian blind mode is enabled).*				
Rocker *x* – Move up / down command	Rocker *x* = indep. or single Function *x* = enabled Type = shutter or venetian blind	1 bit	CRWTU	[1.008] up/down	17,34 (1A, 1B) 55,72 (2A, 2B) 93,110 (3A, 3B) 131,148 (4A, 4B) 169,186 (5A, 5B) 207,224 (6A, 6B)
	Move the blind to fully open or fully closed position. *The object is sent either on first press (if the venetian blind mode is disabled) or on long press (if the venetian blind mode is enabled)*				

6.3.2.4 *Independent or single: scene*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Scene number	Rocker *x* = indep. or single Function *x* = enabled Type = scene	1 Byte	CR-T-	[17.*] Scene number [18.*] Scene control	18,35 (1A, 1B) 56,73 (2A, 2B) 94,111 (3A, 3B) 132,149 (4A, 4B) 170,187 (5A, 5B) 208,225 (6A, 6B)
	Store or recall a scene. The lowest 6 bits in the byte form the code of the scene, while the highest bit is the operation code (store or recall).				



6.3.2.5 *Coupled: switch*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Switching command	Rocker *x* = coupled Function *x* = enabled Type = switch	1-bit	CRWTU	[1.001] switch	13 51 89 127 165 203

6.3.2.6 *Coupled: dimming*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Switching command	Rocker *x* = coupled Function *x* = enabled Type = dimming	1 bit	CRWTU	[1.001] switch	13 51 89 127 165 203
	See notes as for independent command.				
Rocker *x* – Dimming up / down / stop command	Rocker *x* = indep. or single Function *x* = enabled Type = dimming	4 bit	CR-T-	[3.*] 3-bit control	14 52 90 128 166 204
	See notes as for independent command.				

6.3.2.7 *Coupled: shutter or venetian blind*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Dedicated stop command	Rocker *x* = coupled Function *x* = enabled Type = shutter or venetian blind **Blind mode = disabled**	1 bit	CRWTU	[1.017] trigger	13 51 89 127 165 203
	See notes as for independent command.				
Rocker *x* – Stop – step up/down command	Rocker *x* = coupled Function *x* = enabled Type = shutter or venetian blind **Blind mode = enabled**	1 bit	CR-T-	[1.007] step	16 54 92 130 168 206
	See notes as for independent command.				

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Move up / down command	Rocker *x* = coupled Function *x* = enabled Type = shutter or venetian blind	1 bit	CRWTU	[1.008] up/down	17 55 93 131 169 207
	See notes as for independent command.				

6.3.3 Rocker *x:* Function *A/B* configuration

6.3.3.1 Independent or single

For the *independent or* single channel setting, all parameters listed below are referred to either Function A or Function B (whichever are enabled).

In the following sections, it is implicitly understood that for the listed parameters to appear, the corresponding functions xA and/or xB must be enabled.

The entries assigned to "Object *n*" are repeated so many times as the number of configured objects according to the *Number of Communication Objects* parameter.

For all Type values:

Parameter name	*Conditions*	*Settings*
Lock function	-	enabled / **disabled**
	Enables or disables the capability of locking the input through a remote command (telegram).	
Lock function – Invert lock device signal	Lock function = enabled Rocker *x* = independent or single	**not inverted** / inverted
	Allows to interpret a "lock activate" telegram as unlock and vice-versa.	
Lock function – Lock after bus recovery	Lock function = enabled Rocker *x* = independent or single	**no** / yes
	If active, after returning from a bus failure or power-off the device will retain the lock status it had before. Otherwise (in the default case), the device will restart in the non-locked condition.	

When the lock function is enabled, for each input or channel a behaviour can be defined to be followed when the locking or unlocking command is received.

The details will be listed in the following sections; the different behaviours are summarized in the table below.

Channel mode	*Input type*	*Behaviour at locking*	*Behaviour at unlocking*
independent 	send values or sequences	**none** as close or short press as open or long press	
 coupled	switching	**none** off on toggle	**none** off on as previous
independent coupled	dimming		
independent 	scene	none send first scene send second scene	
independent coupled	shutter or venetian blind	none up down	

6.3.3.2 Independent or single: Lock function enabled

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *xx* – Lock command	Rocker *x* = Indep. Or single Lock function = enabled	1 bit	`C-W--`	[1.003] enable	4,21 (1A, 1B) 42,59 (2A, 2B) 80,97 (3A, 3B) 118,135 (4A, 4B) 156,173 (5A, 5B) 194,211 (6A, 6B)

6.3.3.3 Independent or single: send values or sequences

Parameter name	*Conditions*	*Settings*
Number of Communication Objects	Rocker *x* = independent or single Type = send values or sequences	1…8 (**1**)
	Number of communication objects configured in association with the button event.	
Lock function – Behaviour at locking	Rocker *x* = independent or single Type = send values or sequences	**none** as close or short press as open or long press
	Allows to perform the operation associated to the specified event when a locking command is received.	
Lock function – Behaviour at unlocking	Rocker *x* = independent or single Type = send values or sequences	**none** as close or short press as open or long press
	Allows to perform the operation associated to the specified event when an unlocking command is received.	
Event	Rocker *x* = independent or single Type = send values or sequences	**close / open contact** short / long press
	Type of event that should be used as trigger for an action	
Long press time	Rocker *x* = independent or single Type = send values or sequences Event = short /long press	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
Object *n* – send delay	Rocker *x* = independent or single Type = send values or sequences	hh:mm:ss.ff (**00:00:00.00**)
	Delay before the object is transmitted on the bus. *By defining a delay after the event occurs and before the object value is sent, it is possible to associate a time defined sequence of values to an input event.*	
Object *n* – Cyclical transmission	Rocker *x* = independent or single Type = send values or sequences **Number of Comm. Objects = 1**	none **off / value 1** on / value 2 both off and on / both values
	Defines which of the values, if any, must be cyclically retransmitted whenever activated.	
Object *n* – Cyclic sending interval	Rocker *x* = independent or single Type = send values or sequences **Number of Comm. Objects = 1** **Send cyclically ≠ none**	hh:mm:ss (**00:02:00**)
	Interval between cyclical transmissions.	
Object *n* – send dimension	Rocker *x* = independent or single Type = send values or sequences	**1 bit value** 2 bits value 1 byte unsigned value 1 byte percentage 1 byte signed value 2 bytes unsigned value 2 bytes signed value 2 bytes floating value
	Defines size and type of the values to be sent when an event occurs.	

Parameter name	*Conditions*	*Settings*
Object *n* – Close *or* Short Press	Rocker *x* = independent or single Type = send values or sequences **send dimension = 1 bit value**	none on off **toggle**
	Rocker *x* = independent or single Type = send values or sequences **send dimension = 2 bit value**	none disable enable off / up **enable on / down** enable off / up ↔ disable enable on / down ↔ disable enable off / up ↔ enable on / down
	Rocker *x* = independent or single Type = send values or sequences **send dimension = *any byte value***	none **send value 1** send value 2 send value 1 ↔ send value 2
	Value change behaviour caused by either a Close or a Short Press event (according to event configuration)	
Object *n* – Open *or* Long Press	Rocker *x* = independent or single Type = send values or sequences **send dimension = 1 bit value**	**none** on off toggle
	Rocker *x* = independent or single Type = send values or sequences **send dimension = 2 bit value**	none **disable** enable off / up enable on / down enable off / up ↔ disable enable on / down ↔ disable enable off / up ↔ enable on / down
	Rocker *x* = independent or single Type = send values or sequences **send dimension = *any byte value***	**none** send value 1 send value 2 send value 1 ↔ send value 2
	Value change behaviour caused by either an Open or a Long Press event (according to event configuration)	
Object *n* – Value 1	Rocker *x* = independent or single Type = send values or sequences **send dimension = *any byte value***	0...255 (1 byte unsigned value) 0…100 (1 byte percentage) -128...127 (1 byte signed value) 0...65535 (2 bytes unsigned value) -32768... 32767 (2 bytes signed value) -671088.64...670760.96 (2 bytes floating value)
	First value available for association in send events	
Object *n* - Value 2	Rocker *x* = independent or single Type = send values or sequences **send dimension = *any byte value***	*same as value 1*
	Second value available for association in send events	

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *xx* – Switching status *[type]* Object *n*	Rocker *x* = Indep. Or single Type = send values or sequences	*See table below*	CRWTU	*See table below*	5,22 (1A, 1B) 43,60 (2A, 2B) 81,98 (3A, 3B) 119,136 (4A, 4B) 157,174 (5A, 5B) 195,212 (6A, 6B)
	The listed CO numbers are those referring to object nr.1; the COs for each subsequent object are following in sequence. *To obtain the CO numbers for object number n, just add (n-1) to the listed numbers.* *E.g.: COs associated to input 3A (of Rocker 3) have numbers from 81 to 89. The number of CO nr. 5 is therefore 81+(5-1) = 85.*				

Sizes and DPTs are as follows:

Size	*DPT*
1 bit	[1.001] switch
2 bits	[2.*] 1-bit controlled
1 byte unsigned	[4.*] character [5.*] 8-bit unsigned value [20.*] 1-byte
1 byte percentage	[4.*] character [5.*] 8-bit unsigned value [20.*] 1-byte
1 byte signed	[6.*] 8-bit signed value
2 bytes unsigned	[7.*] 2-byte unsigned value
2 bytes signed	[8.*] 2-byte signed value
2 bytes floating	[9.*] 2-byte float value

6.3.3.4 Independent or single: dimming

Parameter name	*Conditions*	*Settings*
Long press time	Rocker *x* = independent or single Type = dimming	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
Toggle mode	Rocker *x* = independent or single Type = dimming	**enabled** / disabled
	When enabled, causes the short press to toggle the on-off status of the destination CO; otherwise, a fixed status can be assigned to the short press.	
Long action	Rocker *x* = independent or single Type = dimming **Toggle mode = enabled**	**darker** brighter darker ↔ brighter
	Defines the function to be assigned to the long press. If the toggle mode is enabled, the Short press action is already defined as toggle.	
Short / Long action	Rocker *x* = independent or single Type = dimming **Toggle mode = disabled**	**off / darker** on / brighter off / darker ↔ brighter on / darker ↔ brighter
	Defines the function to be assigned to the long and short press.	
Send cyclically	Rocker *x* = independent or single Type = dimming	**none** off / value 1 on / value 2 both off and on / both values
	Defines which of the values, if any, must be cyclically retransmitted whenever activated.	
Cyclic sending interval	Rocker *x* = independent or single Type = dimming **Send cyclically ≠ none**	hh:mm:ss (**00:02:00**)
	Interval between cyclical transmissions.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = independent or single Type = dimming	**none** off on toggle
	Value to be assigned to the object when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = independent or single Type = dimming	**none** off on as previous
	Value to be assigned to the object when an unlocking command is received.	

6.3.3.5 Independent or single: shutter or venetian blind

Parameter name	*Conditions*	*Settings*
Long press time	Rocker *x* = independent or single Type = shutter or venetian blind	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
Toggle mode	Rocker *x* = independent or single Type = shutter or venetian blind	**enabled** / disabled
	When enabled, causes each subsequent press to invert the direction of movement; otherwise, a fixed direction can be assigned.	
Up / Down action	Rocker *x* = independent or single Type = shutter or venetian blind **Toggle mode = disabled**	**up** down
	Defines the movement direction to be assigned to the button press.	
Blind mode	Rocker *x* = independent or single Type = shutter or venetian blind	enabled / **disabled**
	If blinds mode is enabled, the device sends "full movement" telegrams on long press and "step" telegrams on short press; if it is disabled, the device sends "full movement" telegrams on long press and "stop" telegrams on short press.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = independent or single Type = shutter or venetian blind	**none** up down
	Operation to perform when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = independent or single Type = shutter or venetian blind	**none** up down
	Operation to perform when an unlocking command is received.	

6.3.3.6 Independent or single: scene

Parameter name	*Conditions*	*Settings*
First scene number	Rocker *x* = independent or single Type = scene	1…64 (**1**)
	Main scene number to be assigned to button press. It is named “first” for the case that an alternative scene number is used.	
Learning mode	Rocker *x* = independent or single Type = scene	enabled / **disabled**
	When enabled, a long key press can be used to program the selected scene by storing the current parameters.	
Long press time	Rocker *x* = independent or single Type = scene **Learning mode = enabled**	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
Scene activation	Rocker *x* = independent or single Type = scene **Learning mode = disabled**	**send first scene only** toggle between two scenes
	Allows the key to be used to alternate between two different scenes.	
Second scene number	Rocker *x* = independent or single Type = scene **Learning mode = disabled** **Scene activation = toggle between two scenes**	1…64 (**2**)
	Alternate scene number to be assigned to button press.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = independent or single Type = scene	**none** send first scene send second scene
	Allows to perform the specified operation when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = independent or single Type = scene	**none** send first scene send second scene
	Allows to perform the specified operation when an unlocking command is received.	

6.3.3.7 *Coupled*

For a *coupled* channel, all the parameters are referred to the single menu entry for Function xA and xB.

In the following sections, it is implicitly understood that for the listed parameters to appear, the corresponding functions xA and xB must be enabled.

For all Type values:

Parameter name	*Conditions*	*Settings*
Lock function	Rocker *x* = coupled	enabled / **disabled**
	Enables or disables the capability of locking the input through a remote command (telegram).	

When the lock function is enabled, for each input or channel a behaviour can be defined to be followed when the locking or unlocking command is received.

The details will be listed in the following sections; for a table summarizing the different behaviours, please refer to the corresponding section about the independent channels.

6.3.3.8 *Coupled: Lock function enabled*

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *xx* – Lock command	Rocker *x* = coupled Lock function = enabled	1 bit	`C-W--`	[1.003] enable	4 42 80 118 156 194

6.3.3.9 *Coupled: switch*

Parameter name	*Conditions*	*Settings*
xA and xB use	Rocker *x* = coupled Type = switch	**A on, B off** A off, B on
Send cyclically	Rocker *x* = coupled Type = switch	none **off / value 1** on / value 2 both off and on / both values
	Defines which of the values, if any, must be cyclically retransmitted whenever activated.	
Cyclic sending interval	Rocker *x* = coupled Type = switch **Send cyclically ≠ none**	hh:mm:ss (**00:02:00**)
	Interval between cyclical transmissions.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = coupled Type = switch	**none** on off toggle
	Value to be assigned to the object when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = coupled Type = switch	**none** on off as previous
	Value to be assigned to the object when an unlocking command is received.	

6.3.3.10 Coupled: dimming

Parameter name	*Conditions*	*Settings*
Long press time	Rocker *x* = coupled Type = dimming	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
xA and xB use	Rocker *x* = coupled Type = dimming	**A increases, B decreases** A decreases, B increases
Send cyclically	Rocker *x* = coupled Type = dimming	none **off / value 1** on / value 2 both off and on / both values
	Defines which of the values, if any, must be cyclically retransmitted whenever activated.	
Cyclic sending interval	Rocker *x* = coupled Type = dimming **Send cyclically ≠ none**	hh:mm:ss (**00:02:00**)
	Interval between cyclical transmissions.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = coupled Type = dimming	**none** off on toggle
	Value to be assigned to the object when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = coupled Type = dimming	**none** off on as previous
	Value to be assigned to the object when an unlocking command is received.	

6.3.3.11 Coupled: shutter or venetian blind

Parameter name	*Conditions*	*Settings*
Long press time	Rocker *x* = coupled Type = shutter or venetian blind	hh:mm:ss.fff (**00:00:03.000**)
	Minimum push time for a press in order to be recognized as a long press.	
xA and xB use	Rocker *x* = coupled Type = shutter or venetian blind	**A up, B down** A down, B up
Blind mode	Rocker *x* = coupled Type = shutter or venetian blind	enabled / **disabled**
	If blinds mode is enabled, the device sends "full movement" telegrams on long press and "step" telegrams on short press; if it is disabled, the device sends "full movement" telegrams on long press and "stop" telegrams on short press.	
Lock function – Behaviour at locking	Lock function = enabled Rocker *x* = coupled Type = shutter or venetian blind	**none** up down
	Operation to perform when a locking command is received.	
Lock function – Behaviour at unlocking	Lock function = enabled Rocker *x* = coupled Type = shutter or venetian blind	**none** up down
	Operation to perform when an unlocking command is received.	

For other communication objects related to *coupled* mode, please refer to the general *Rockers Configuration* section.

6.3.4 Rocker *x:* LED configuration

Following parameters are repeated for each of the available LEDs.

LED parameters settings are always listed grouped by rocker (regardless whether the inputs are coupled or not): for each rocker *x*, available LEDs are marked Green *x*A, Blue *x*A, Green *x*B, Blue *x*B.

Parameter name	*Conditions*	*Settings*
Green/Blue LED *XX*	-	fixed **when contact closed** status from bus
Always	Green/Blue LED *XX* = fixed	on / **off**
	Status for a fixed LED condition	
Off delay	Green/Blue LED *XX* = when contact closed	hh:mm:ss.ff (**00:02:00.00**)
	Delay before LED goes off	
Blinking	Green/Blue LED *XX* = status from bus	**no** / yes
Signal from bus	Green/Blue LED *XX* = status from bus	**not inverted** / inverted
	Specifies whether the LED status from the bus should be inverted, i.e. LED on when an "off" command is received on the communication object.	

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Rocker *x* – Led *[type]* A/B *command*	Green/Blue LED *XX* = status from bus	1 bit	CRWTU	[1.001] switch	*See table below*

CO numbers for bus LED switches:

	Green A	Blue A	Green B	Blue B
Rocker 1	38	39	40	41
Rocker 2	76	77	78	79
Rocker 3	114	115	116	117
Rocker 4	152	153	154	155
Rocker 5	190	191	192	193
Rocker 6	228	229	230	231

7 Appendix

7.1 Summary of KNX Communication Objects

Following is a summary of all KNX Communication Objects (CO) and corresponding Data Point Types (DPT) defined by the application program according to configuration options.

The listing order is generally by CO number (in case of COs repeated by channel, the CO number for the first channel is taken as relevant).

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Technical alarm	Technical alarm = enabled	1 bit	C-W--	[1.005] alarm	0
Brightness value	*Not used (previous versions only)*	*2 Byte*	*CR-T-*	*[9.004]* *lux (Lux)*	*1*
LED intensity percentage	LED intensity from the bus = yes	1 Byte	C-W--	[5.001] Percentage (0..100%)	2
Temperature value	Temp. sensor = enabled	2 Byte	CR-T-	[9.001] temperature (°C)	3
Rocker *xx* – **Lock command**	Rocker *x* = Independent or single, Channel A Lock function = enabled	1 bit	C-W--	[1.003] enable	4, 42, 80, 118, 156, 194
	Rocker *x* = coupled Lock function = enabled				
Rocker *xx* – **Switching status *[type]* Object *n****	Rocker *x* = Independent or single, Channel A **Type = send values or sequences**	***See table 1***	CRWTU	***See table 1***	5..12, 43..50, 81…88, 119..126, 157..164, 195..202
		** The listed CO numbers are starting from object nr.1; the COs for each subsequent object are following in sequence.* *To obtain the CO numbers for object number n, just add (n-1) to the listed numbers.E.g.: COs associated to input 3A (of Rocker 3) have numbers from 81 to 89. The number of CO nr. 5 is therefore 81+(5-1) = 85.*			
Rocker *xx* – **Switching command**	Rocker *x* = Independent or single, Channel A **Type = dimming**	1 bit	CRWTU	[1.001] switch	13, 51, 89, 127, 165, 203
	Rocker *x* = coupled **Type = switch**				
	Rocker *x* = coupled **Type = dimming**				
Rocker *xx* – **Dedicated stop command**	Rocker *x* = Independent or single, Channel A **Type = shutter or venetian blind**	1 bit	CRWTU	[1.017] trigger	13, 51, 89, 127, 165, 203
	Rocker *x* = coupled **Type = shutter or venetian blind** **Blind mode = disabled**				

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Technical alarm	Technical alarm = enabled	1 bit	`C-W--`	[1.005] alarm	0
Rocker *xx* – **Dimming up / down / stop command**	Rocker *x* = Independent or single, Channel A **Type = dimming**	4 bit	`CR-T-`	[3.*] 3-bit control	14, 52, 90, 128, 166, 204
Rocker *xx* – **Stop – step up/down command**	Rocker *x* = Independent or single, Channel A **Type = shutter or venetian blind** **Blind mode = enabled**	1 bit	`CR-T-`	[1.007] step	16, 54, 92, 130, 168, 206
	Rocker *x* = coupled **Type = shutter or venetian blind** **Blind mode = enabled**				
Rocker *xx* – **Move up / down command**	Rocker *x* = Independent or single, Channel A **Type = shutter or venetian blind**	1 bit	`CRWTU`	[1.008] up/down	17, 55, 93, 131, 169, 207
	Rocker *x* = coupled **Type = shutter or venetian blind**				
Rocker *xx* – **Scene number**	Rocker *x* = Independent or single, Channel A **Type = scene**	1 Byte	`CR-T-`	[17.*] Scene number [18.*] Scene control	18, 56, 94, 132, 170, 208
*Light threshold 1 - **Switch***	*Not used (previous versions only)*	*1 bit*	*`CR-T-`*	*[1.001] switch*	*20*
Rocker *xx* – **Lock command**	Rocker *x* = Independent or single, Channel B **Lock function = enabled**	1 bit	`C-W--`	[1.003] enable	21, 59, 97, 135, 173, 211
Rocker *xx* – **Switching status *[type]* Object *n****	Rocker *x* = Independent or single, Channel B **Type = send values or sequences**	***See table 1***	`CRWTU`	***See table 1***	22..29, 60..67, 98..105, 136..143, 174..181, 212..219
		** The listed CO numbers are starting from object nr.1; the COs for each subsequent object are following in sequence.* *To obtain the CO numbers for object number n, just add (n-1) to the listed numbers. E.g.: COs associated to input 3B (of Rocker 3) have numbers from 98 to 105. The number of CO nr. 5 is therefore 98+(5-1) = 102.*			
Rocker *xx* – **Switching command**	Rocker *x* = Independent or single, Channel B **Type = dimming**	1 bit	`CRWTU`	[1.001] switch	30, 68, 106, 144, 182, 220
Rocker *xx* – **Dedicated stop command**	Rocker *x* = Independent or single, Channel B **Type = shutter or venetian blind** **Blind mode = Disabled**	1 bit	`CRWTU`	[1.017] trigger	30, 68, 106, 144, 182, 220

Object name	*Conditions*	*Size*	*Flags*	*DPT*	*CO number(s)*
Technical alarm	Technical alarm = enabled	1 bit	`C-W--`	[1.005] alarm	0
Rocker *xx* – **Dimming up / down / stop command**	Rocker *x* = Independent or single, Channel B **Type = dimming**	4 bit	`CR-T-`	[3.*] 3-bit control	31, 69, 107, 145, 183, 221
Rocker *xx* – **Stop – step up/down command**	Rocker *x* = Independent or single, Channel B **Type = shutter or venetian blind** **Blind mode = enabled**	1 bit	`CR-T-`	[1.007] step	33, 71, 109, 147, 185, 223
Rocker *xx* – **Move up / down command**	Rocker *x* = Independent or single, Channel B **Type = shutter or venetian blind**	1 bit	`CRWTU`	[1.008] up/down	34, 72, 110, 148, 186, 224
Rocker *xx* – **Scene number**	Rocker *x* = Independent or single, Channel B **Type = scene**	1 Byte	`CR-T-`	[17.*] Scene number [18.*] Scene control	35, 73, 111, 149, 187, 225
*Light threshold 2 - **Switch***	*Not used (previous versions only)*	*1 bit*	*`CR-T-`*	*[1.001] switch*	*37*
Rocker x – **Led [type] command**	Green/Blue LED XX = status from bus	1 bit	`CRWTU`	[1.001] switch	***See table 2***

Table 1. Independent/single channel object sizes and DPTs:

Size	*DPT*
1 bit	[1.001] switch
2 bits	[2.*] 1-bit controlled
1 byte unsigned	[4.*] character [5.*] 8-bit unsigned value [20.*] 1-byte
1 byte percentage	[4.*] character [5.*] 8-bit unsigned value [20.*] 1-byte
1 byte signed	[6.*] 8-bit signed value
2 bytes unsigned	[7.*] 2-byte unsigned value
2 bytes signed	[8.*] 2-byte signed value
2 bytes floating	[9.*] 2-byte float value

Table 2. CO numbers for bus LED switches:

	Green A	Blue A	Green B	Blue B
Rocker 1	38	39	40	41
Rocker 2	76	77	78	79
Rocker 3	114	115	116	117
Rocker 4	152	153	154	155

7.2 Warning

- Installation, electrical connection, configuration and commissioning of the device can only be carried out by qualified personnel.
- Opening the housing of the device causes the immediate end of the warranty period.
- ekinex® KNX defective devices must be returned to the manufacturer at the following address: Ekinex S.p.A. Via Novara, 37, I-28010 Vaprio d'Agogna (NO) Italy.

7.3 Other information

- This application manual is aimed at installers, system integrators and planners
- For further information on the product, please contact the ekinex® technical support at the e-mail address: support@ekinex.com or visit the website www.ekinex.com
- KNX® and ETS® are registered trademarks of KNX Association cvba, Brussels