



# USER'S MANUAL

**MDU/MES000x-MyA**

**KNX Hotel Solution:**

*Door Unit MDU000x-My*

*Energy Saver MES000x-My*

**Device Manipulation & ETS™ Application  
Description**

August 2017

## Revision Sheet

<b>Release No.</b>	<b>Date</b>	<b>Revision Description</b>
<i>Rev. 0</i>	<i>24/8/2017</i>	<i>User's Manual Created</i>
<i>Rev. 1</i>	<i>14/11/2017</i>	<i>Added more detailed information about parameters</i>

DRAFT

# USER'S MANUAL

## TABLE OF CONTENTS

	<u>Page #</u>
<b>1 GENERAL INFORMATION.....</b>	<b>6</b>
1.1 System Overview.....	6
1.2 Minimum requirements.....	6
1.3 Acronyms and Abbreviations.....	6
<b>2 SYSTEM SUMMARY.....</b>	<b>9</b>
2.1 Application Functions Overview.....	9
2.2 Devices variations.....	10
<b>3 GETTING STARTED.....</b>	<b>12</b>
3.1 Connecting for the first time.....	12
3.2 Downloading application with ETS™.....	12
<b>4 DETAILED PARAMETERS DESCRIPTION.....</b>	<b>14</b>
<b>4.1 DND General Configuration.....</b>	<b>14</b>
4.1.1 DND General Configuration – Door Unit	15
4.1.2 DND General Configuration – Energy Saver	18
4.1.3 DND Confs – Authentication	20
4.1.4 DND Confs – Memory Access Configurations	23
4.1.5 DND Confs – Authentication Ids Configuration	28
<b>4.2 Authentication Actions.....</b>	<b>30</b>
<b>4.3 Logic Channels.....</b>	<b>35</b>
4.3.1 Copy and Forward	35
4.3.2 Logic Operation	39
4.3.3 Comparison	45
4.3.4 Mathematical operation	49
<b>5 OPERATION DESCRIPTION.....</b>	<b>53</b>
<b>5.1 Local vs. Remote authentication.....</b>	<b>53</b>
5.1.1 Local authentication typical usage	55

5.1.2 Remote authentication typical usage	56
<b>5.2 Using device for creating cards.....</b>	<b>57</b>
5.2.1 The sequence for creating the card from brand new card	58
5.2.2 The sequence for creating the card from already used card	59
<b>5.3 Example: changing authentication credentials for Local Authentication.....</b>	<b>59</b>
5.3.1 Making card to not authenticate	59
5.3.2 Setting the room's devices authentication to the created card's credentials	60
<b>5.4 Encryption.....</b>	<b>61</b>
<i>Appendix A -Logic operations.....</i>	<i>65</i>
I - AND (Logical Conjunction).....	65
II - OR (Logical Disjunction).....	66
III - XOR (Exclusive disjunction).....	67
IV - NOT (Negation).....	68
<i>Appendix B -KNX Data types.....</i>	<i>69</i>
<i>Appendix C -Card's memory access conditions (nxp vs fudan).....</i>	<i>72</i>
<i>Appendix D -Mifare Memory Layout.....</i>	<i>74</i>
<i>Appendix E -Encryption, CRC16: Algorithms &amp; Source code.....</i>	<i>75</i>
I - Encryption: RC4 algorithm and source code.....	75
II - Cyclic Redundancy Check: CRC16 algorithm and source code.....	79
<i>Appendix F -Detailed description of Communication objects.....</i>	<i>80</i>
I - DND.....	80
II - Logic Channels.....	82

## **1 GENERAL INFORMATION**

---

## 1 GENERAL INFORMATION

### 1.1 System Overview

This manual refers to the following devices for KNX bus:

- **MDU0001-M**: KNX Hotel Door Unit
- **MDU0001-MR**: KNX Hotel Door Unit with Relay
- **MES0001-M**: KNX Hotel Energy Saver

All of the previous include 1 built-in **Mifare Classic**<sup>1</sup> Card reader, and 4 configurable *Logic Channels* (each of them configurable as *Logic Operation*, *Copy and Forward*, *Comparison* or *Mathematical Operation*), 1 programming button and 1 programming mode indication LED.

The *Logic Channels* are intended to provide flexibility in automation tasks, by allowing the user to reproduce a desired action upon the verification of determined situation. This module will be described in detail ahead in this manual.

### 1.2 Minimum requirements

The ETS<sup>TM</sup> **MDU/MES000x-MyA** application is to be used with KNX Association's ETS4<sup>TM</sup> or higher.

Known to work with:

- ETS4<sup>TM</sup> (version 4.2.0 build 3884)
- ETS5<sup>TM</sup> (version 5.5.2 build 665)

### 1.3 Acronyms and Abbreviations

<b>CO</b>	Communication Object
<b>DPT</b>	Data Point Type
<b>EIB</b>	European Installation Bus (former name to KNX; no longer in use)
<b>GA</b>	Group Address
<b>LED</b>	Light Emitting Diode
<b>DND</b>	Do Not Disturb
<b>ID</b>	Identification (card's identification number)
<b>RC4</b>	Rivest Cipher 4
<b>CRC</b>	Cyclic Redundancy Check

---

<sup>1</sup> Please refer to Mifare Classic technology reference: [http://www.nxp.com/products/identification-and-security/mifare-ics/mifare-classic:MC\\_41863](http://www.nxp.com/products/identification-and-security/mifare-ics/mifare-classic:MC_41863)

DRAFT

## **2 SYSTEM SUMMARY**

---



## 2 SYSTEM SUMMARY

Table 1: Applications specifications

Variant \ Specs	Number of Communication Objects	Maximum number of Group Addresses	Maximum number of Associations
MDU/MES000x-MyA	156 <sup>2</sup>	250	250

### 2.1 Application Functions Overview

The **MDU/MES000x-MyA** ETST<sup>TM</sup> application provides the interface to configure the card authentication method to one of the following:

- Accept all cards;
- Local authentication:
  - card's memory check;
  - card's ID check;
- Remote authentication:
  - card's memory check;
  - card's ID check;

The available buttons can be configured for one of the following functions:

- Button not used;
- ON / OFF;
- Toggle Switch;
- Dimming;
- Shutter / Blinds;
- Heating;
- Priority;
- Scene;
- Value;
- 2-Channel mode.

Additionally, each of the 4 Logic Channels can be configured to one of the following modes:

- Not used;
- Logic operation (binary);
- Copy and Forward;
- Comparison;
- Mathematical Operation.

Many other features are available, and those will be discussed in detail in DETAILED PARAMETERS DESCRIPTION.

---

2 This value is the number of COs reserved in device's memory, however some of them may not become visible, depending on the device's configuration.

## 2.2 Devices variations

There are two hardware that shares the same application:

- **MDU0001-MR**: equipped with Mifare Classic reader and up to 12V, 2A solid state relay;
- **MES0001-M**: equipped with Mifare Classic reader;

Functionally both are similar;

*Table 2: Devices differences*

Variant	MDU000x-My	MES000x-My
Functionality		
Relay	X	
LCD	X	
Buzzer		X
Number of buttons	1	3
Number of LEDs	4	4

### **3 GETTING STARTED**

---

## 3 GETTING STARTED

### 3.1 Connecting for the first time

After connecting the MDU000x-My or MES000x-My device for the first time to the KNX/EIB bus, the user will see the buttons' LEDs turning switching between red and blue one at a time.

This behaviour means that the device hasn't been loaded with a valid ETS™ application yet. The same behaviour may be observed when an invalid application is loaded into the device.

### 3.2 Downloading application with ETS™

If it's the first time that the device will be programmed, you must define the Individual Address via ETS™ interface. You must also press the programming button on the device for allowing ETS™ to identify the target device. You will know that the device is in programming mode when the programming LED turns on. During programming process the programming LED and the programming mode will automatically turn off.

The Individual Address is normally written once, however if it's necessary to re-write the Individual Address, the programming button must be pressed.

Once the device has its Individual Address, the device can be configured according to the project needs using ETS™ application, selecting “Download Application”.

## **4 DETAILED PARAMETERS DESCRIPTION**

---

## 4 DETAILED PARAMETERS DESCRIPTION

In this section all the functions will be introduced and explained in detail, as well as explained the ETS™ Product Database usage. This information should be enough for the installer to understand the device operation in any of the functions and to configure it with the ETS™ database.

**NOTE:** For configuring the Buttons and LEDs the user should refer to the MSW100X-PL manual since the configurations are the same.

The combination of Hotel Room Door Unit with Hotel Room Energy Saver is said to be *Do Not Disturb system* (DND). From this point onwards when referring to DND it is being referred to this group.

### 4.1 DND General Configuration

In ETS™, when you select the general page you will see an environment similar with the one in presented in Figure 1. In this page, in the parameter “**Select target device**” it must be selected the correct Hardware:

- **MDU000x-M** it must be selected *Door Unit* option
- **MES000x-M** it must be selected *ESaver* option

Depending on this selection different group of parameters are going to be available.

### 4.1.1 DND General Configuration – Door Unit

1.1.1 Hotel Room Door Unit > DND General

Information	Select target device	<input checked="" type="radio"/> Door Unit <input type="radio"/> ESaver
Buttons General	Override previously saved settings?	<input type="radio"/> No <input checked="" type="radio"/> Yes
DND General	Room Number	1234
+ DND Confs	At power up	Do nothing
+ Level 1	Enable request card operations (read/write) via COs?	<input checked="" type="radio"/> No <input type="radio"/> Yes
+ Level 2	DISPLAY CONFIGURATIONS:	
+ Button 1	Read success info	
+ LED A	At valid card shown on display	Auth
+ LED B	At invalid valid card shown on display	Erro
+ LED C	Show information on display for (sec.)	10
+ Logic Channels	Text via Communication Object	
	Text cycling rate (when more than 4 chars) (x100ms)	3
	Spaces between text's begining and end while cycling	2
	Text timeout (s) (0=never, clear at empty string)	10
	RELAY CONFS:	
	Relay controlable via Comm. Obj.?	<input checked="" type="radio"/> No <input type="radio"/> Yes
	At power up?	Off
	Relay "Off Pulse" duration (x100ms)	0
	Relay "On Pulse" duration (x100ms)	0
	ANTENNA SETTINGS:	
	!NOTE!: This settings must be changed with values provided by the manufacurer. Wrong settings may result in poor or even non operation of the antenna.	
	#GSNONREG	119
	#GSNOFFREG	0
	#MODGSPREG	17
	#CWGSPREG	9

Figure 1: DND General Configurations page (Door Unit)

Here some parameters that will affect all the system can be configured. All the parameters are explained in the Table 3.

Table 3: Description of parameters from DND General (Door Unit) configurations page

Parameter	Description	Values
Override previously saved settings?	Will reset user settings made after installation upon new ETS download.	- *Yes - No
Room Number	Sets the Room Number	<u>Min:</u> 0 <u>Max:</u> 9999
At power up	Selects the behavior of the DND at device power up after a power failure	- *Do nothing - Valid card presented (level 1) - Valid card presented (level 2) - Valid card presented (level master) - Valid card removed (level 1) - Valid card removed (level 2) - Valid card removed (level master)
Enable request card operations (read/write) via COs?	When active, COs for read/write operations on Mifare Classic cards become available	- Yes - *No
<sup>3</sup> Operation request timeout (s)	Time after which a pending read/write request must be terminated	<u>Min:</u> 0s <u>Max:</u> 255s
Set KeyA and KeyB	When set to "Yes" the access KeyA and KeyB from the card are both modified	- *Yes - No
At valid card show on display	The text to show on the 7-segments display when a valid card is presented	4 characters string
At invalid card show on display	The text to show on the 7-segments display when an invalid card is presented	4 characters string
Show information on display for (s)	Defines for how long the information of card read result remains on display	<u>Min:</u> 0s → won't show <u>Max:</u> 255s
Text cycling rate (x100ms)	When the text string via "Show text on display" CO is longer than the display number of characters, defines the amount of time between shift steps of the string; the text is rolled on display	<u>Min:</u> 0 = 0ms <u>Max:</u> 31 = 3100ms
Spaces between text's beginning and end while cycling	When the text string via "Show text on display" CO is longer than the display number of characters, defines the amount of spaces placed between the string's beginning and end	<u>Min:</u> 0 spaces <u>Max:</u> 4 spaces
Text timeout (s)	When a text string is received via "Show text on display" CO, it defines the amount of time it remains showing;	<u>Min:</u> 0s → until an empty string is received <u>Max:</u> 255s
Relay controllable via Comm. Obj.?	When active, COs for controlling the relay's status become available	- Yes - *No
<sup>4</sup> Relay control type	Selects the contact control type of the relay	- *On=Contact Closed; Off=Contact Open - Off=Contact Open; On=Contact Closed
At power up?	Defines the behavior of the relay when the device comes from a power failure	- Do nothing - *Off - On - Keep previous status - Request status via CO
Keep override after power up?	If active, the override status of the relay (controlled via "Prio On/Off") is kept after a power failure	- Yes - *No
Relay "Off Pulse" duration (x100ms)	The duration of the Off pulse	<u>Min:</u> 0 = 0ms <u>Max:</u> 255 = 25.5s

<sup>3</sup> Visible if "Enable request card operations (read/write) via COs?" is "Yes".

<sup>4</sup> Visible if "Relay controllable via Comm. Obj.?" is "Yes".



Relay “On Pulse” duration (x100ms)	The duration of the On pulse	Min: 0 = 0ms Max: 255 = 25.5s
------------------------------------	------------------------------	----------------------------------

The parameter “**Set KeyA and KeyB**” parameter is specially important when using Mifare Classic compatible cards instead of the NXP ones (see Appendix C).

When in 4.2 in the section “RELAY USAGE” any of the parameters is set to “Pulse On” or “Pulse Off”, the parameters “**Relay “Off Pulse” duration (x100ms)**” and “**Relay “Off Pulse” duration (x100ms)**” define the pulse. This can be used for example to trigger a door lock when a valid card is presented.

The parameters under the section “ANTENNA SETTINGS” shall not be modified, unless explicit instructed by the manufacturer.

### 4.1.2 DND General Configuration – Energy Saver

In this section just the parameters that differs from the ones explained in 4.1.1 are going to be explained.

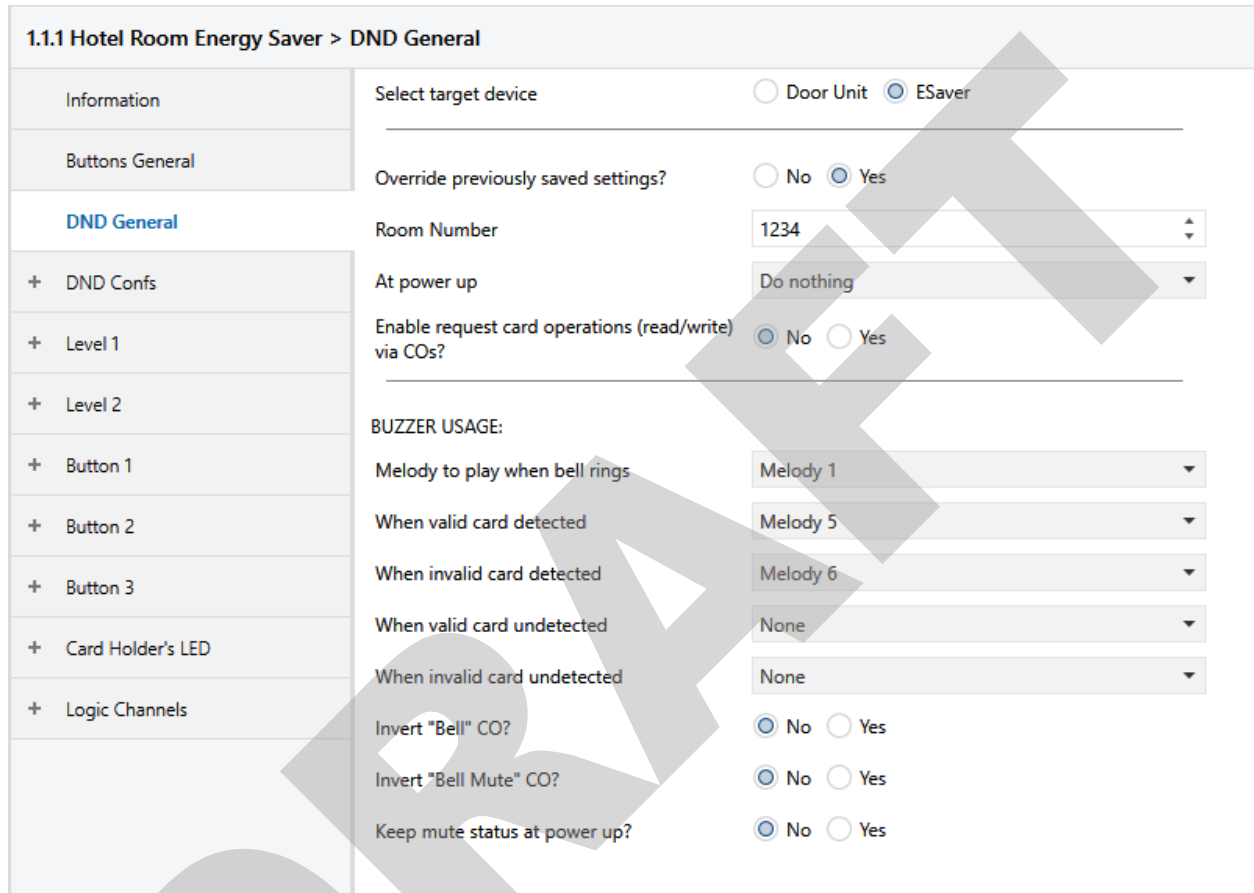


Figure 2: DND General Configurations page (Energy Saver)

A more detailed description of each parameter is found in Table 4.

Table 4: Description of parameters from DND General (Door Unit) configurations page

Parameter	Description	Values
<b>Melody to play when bell rings</b>	Selects the buzzer tone to reproduce when the Bell CO receives and activation value	- None - *Melody 1 - Melody 2 - ... - Melody 6
<b>When valid card detected</b>	Selects the buzzer tone to reproduce when an authenticated card is presented	- None - Melody 1 - ... - *Melody 5 - Melody 6

<b>When invalid card detected</b>	Selects the buzzer tone to reproduce when an unauthenticated card is presented	<ul style="list-style-type: none"> <li>- None</li> <li>- Melody 1</li> <li>- ...</li> <li>- Melody 5</li> <li>- *Melody 6</li> </ul>
<b>When valid card undetected</b>	Selects the buzzer tone to reproduce when an authenticated card is removed from the reading field	<ul style="list-style-type: none"> <li>- *None</li> <li>- Melody 1</li> <li>- ...</li> <li>- Melody 5</li> <li>- Melody 6</li> </ul>
<b>When invalid card undetected</b>	Selects the buzzer tone to reproduce when an unauthenticated card is removed from the reading field	<ul style="list-style-type: none"> <li>- *None</li> <li>- Melody 1</li> <li>- ...</li> <li>- Melody 5</li> <li>- *Melody 6</li> </ul>
<b>Invert "Bell" CO?</b>	When set to "Yes", an <b>Off</b> message via "Bell" CO will trigger the buzzer tone for the bell; otherwise it is triggered by <b>On</b>	<ul style="list-style-type: none"> <li>- *No</li> <li>- Yes</li> </ul>
<b>Invert "Bell Mute" CO?</b>	When set to "Yes", an <b>Off</b> message via "Bell Mute" CO will mute the buzzer tones; otherwise it is muted by <b>On</b>	<ul style="list-style-type: none"> <li>- *No</li> <li>- Yes</li> </ul>
<b>Keep mute status at power up?</b>	When set to "Yes", after a power failure the buzzer mute status is kept	<ul style="list-style-type: none"> <li>- *No</li> <li>- Yes</li> </ul>

### 4.1.3 DND Confs – Authentication

In this section the configurations for the card's authentication are explained. The default configuration page is shown in Figure 3.

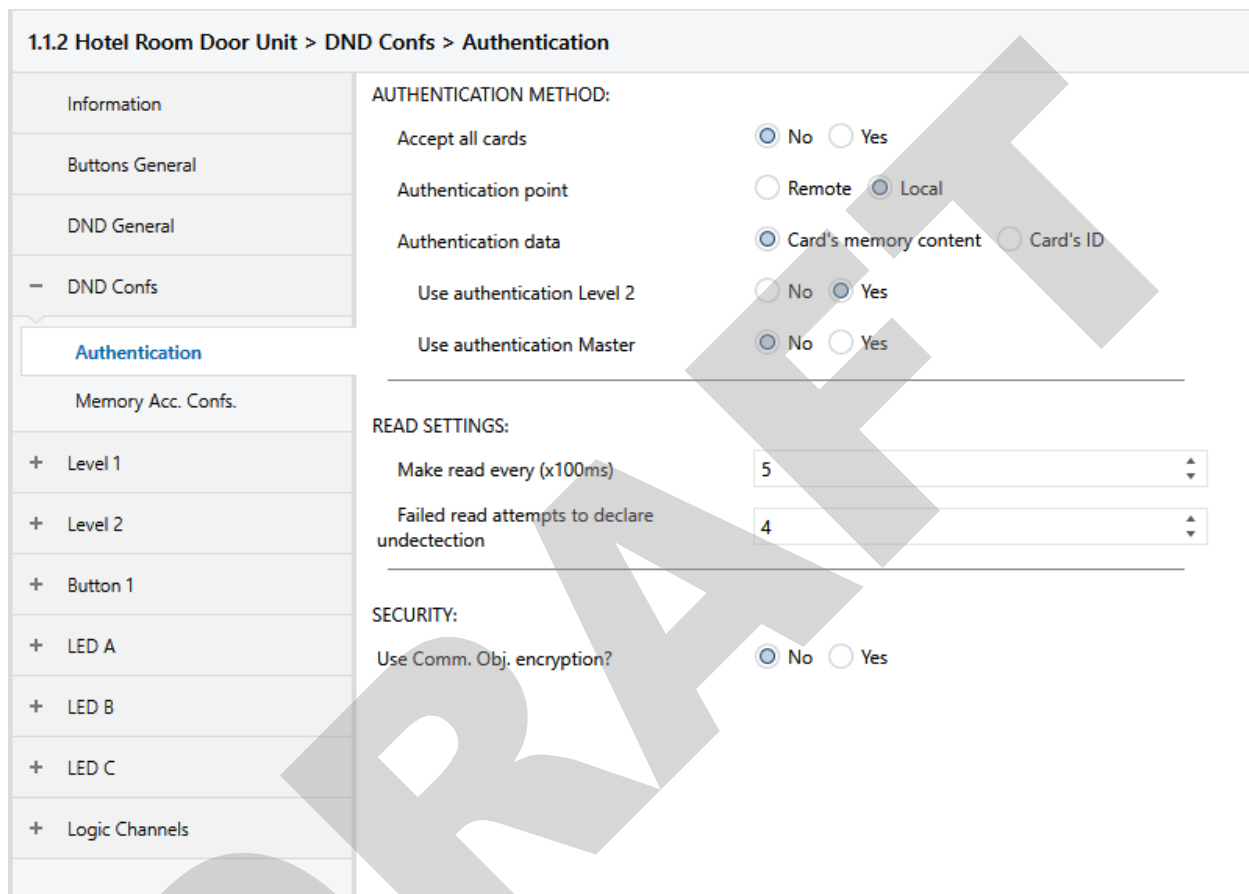


Figure 3: Authentication configurations' page

Table 5: Description of parameters from authentication configurations

Parameter	Description	Values
Accept all cards	If set to "Yes" any card ISO/IEC 14443 compatible that answers to the card select command will be treated as an authenticated card	- *No - Yes
<sup>5</sup> Authentication point	Selects how the authentication is preformed; Remote: the device will get the authentication credentials from the card, issue them to the bus and way acknowledgment; Local: the device itself will have enough information to authenticate the presented card	- Remote - *Local

<sup>5</sup> Visible if "Accept all cards" is "No"

<sup>6</sup> Timeout for remote acknowledge (s)	Sets the maximum amount of time that the acknowledge message can take after the presented card credentials have been sent	Min: 0 = 0s Max: 255 = 255s
Authentication data	Selects which are the card credentials to be tested	- *Card's memory content - Card's ID
Use authentication level 2	Selects whether Level 2 authentication cards are to be tested when a card is on the reading field	- No - *Yes
Use authentication Master	Selects whether Level Master authentication cards are to be tested when a card is on the reading field	- *No - Yes
Make read every (x100ms)	Sets the polling period for attempting to read a card on the reading field	Min: 0 = 0ms Max: 15 = 1.5s
Failed read attempts to declare undetection	Sets how many failed attempts to read a card on the field to declare that the card is not present	Min: 0 Max: 255
Use Comm. Obj. encryption?	When set to "Yes" it is possible to select among a list of COs which ones are to be encrypted	- *No - Yes
<sup>7</sup> Cypher pass as	The cypher pass type	- *ASCII - Decimal
Cypher pass	Sets the encryption key for the selected COs	(see explanation below)
Encrypt CO #?	If set to "Yes" the CO is going to be RC4 encrypted with the "Cypher pass"	- *No - Yes

It shall be noticed that setting "**Accept all cards**" to "Yes" will result in accepting as authenticated any card which answers to the ID request command. This may be used for example for the Energy Saver units in case it is not desired to only allow the room's door access card; a blank general card can be than provided to only activate the Energy Saver.

It's important to understand the meaning of the parameter "**Authentication point**". This parameter defines if the MDU/MES will make a request to a server<sup>8</sup> inquiring the authenticity of a card on the reading field – Remote Authentication Point; or if the MDU/MES possesses enough information for, *per se*, evaluating the authenticity of the card on the reading field. For further insight read 5.1.

The parameter "**Authentication data**" as a very important role as it defines which is the relevant card's details for evaluating it's authenticity. When one selects "Card's ID" the MDU/MES is only going to attempt to evaluate the card's Identification number; this method is very simple and may ease project deployment but has plenty drawbacks: there's a chance, even small, to have more than a card owning the same ID, also cloning a card's ID is an easier process than cloning full memory; the authenticated card's IDs are static and configured as described in 4.1.5. When one selects "Card's memory content" (recommended) the MDU/MES is going to attempt to access the card's memory in order to get its

<sup>6</sup> Visible if "Authentication point" is "Remote"

<sup>7</sup> Visible if "Use Comm. Obj. encryption" is set to "Yes"

<sup>8</sup> The implementation of this "server" shall be your responsibility.

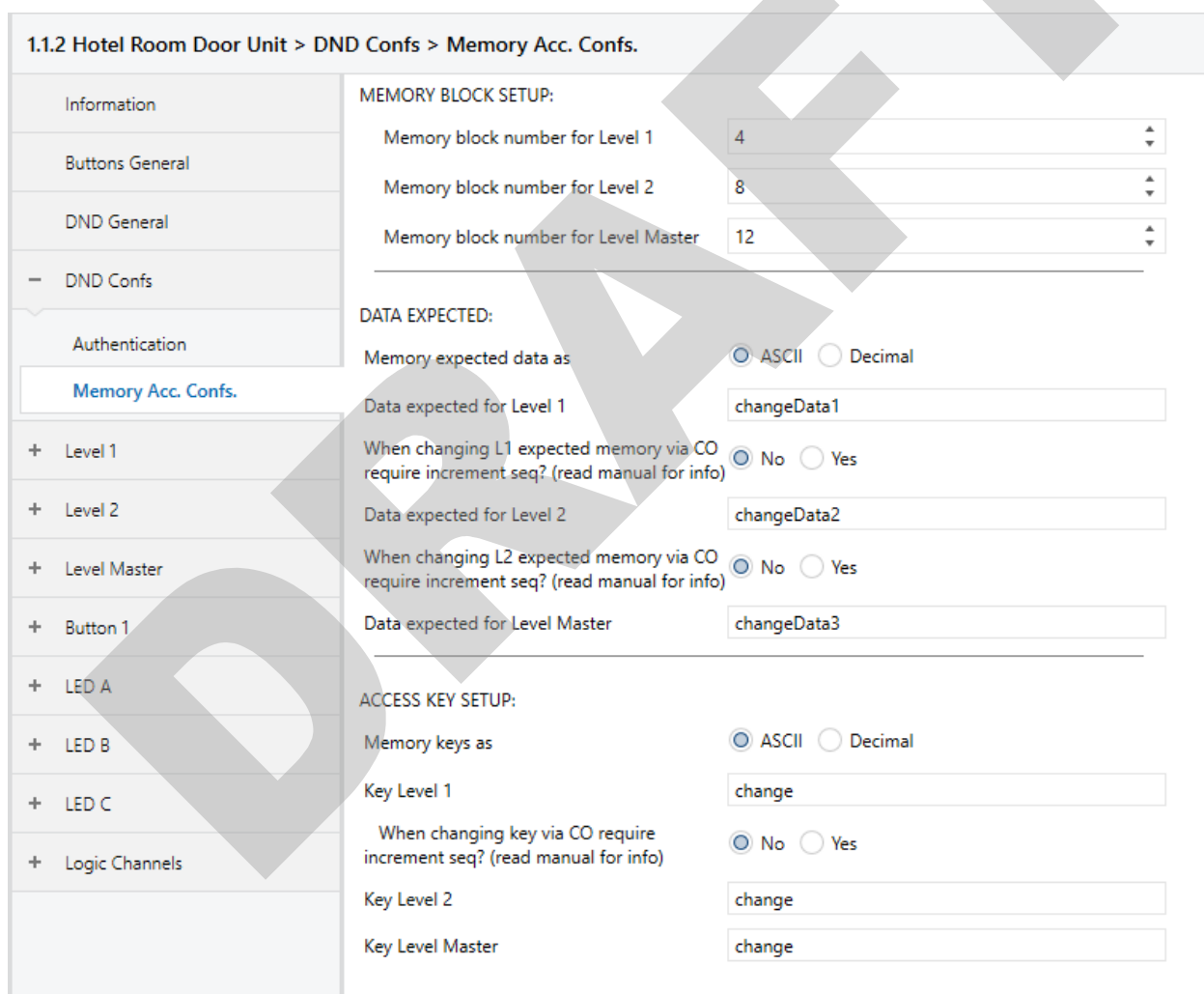
authentication credentials; for this some more parameters must be set configuring how the device will access this memory credentials (see 4.1.4).

For increased security in the system it is possible to encrypt some of the most critical COs in order to protect against hacking attacks. For this the parameter “**Use Comm. Obj. encryption?**” can be activated. This will allow to selective chose from a list which COs will be encrypted (just 14 byte long messages are encrypt-able). The message written to the KNX bus won’t be plain anymore, but will be encrypted with an algorithm and key that its communication partners must implement. This mode is mostly useful for projects where there’s a computer application tailored for the purpose which can include the decryption software. Further details of how to use encryption and how to implement the decryption algorithm are give in 5.4.

#### 4.1.4 DND Confs – Memory Access Configurations

This page contents depends on the settings made on the “Authentication” page. The most relevant parameters affecting this page are “Authentication point” and “Authentication data” (also when “Accept all cards” is set to “Yes” this configuration page is suppressed); this page is only visible if “Authentication data” from the “Authentication” page is set to “Card’s memory content”.

When in the “Authentication” parameters page the “Authentication point” is set to “Local” and “Authentication data” is set to “Card’s memory content” the default settings of “Memory Acc. Confs.” page can be seen in Figure 4.



1.1.2 Hotel Room Door Unit > DND Confs > Memory Acc. Confs.

Information

Buttons General

DND General

– DND Confs

Authentication

Memory Acc. Confs.

+ Level 1

+ Level 2

+ Level Master

+ Button 1

+ LED A

+ LED B

+ LED C

+ Logic Channels

MEMORY BLOCK SETUP:

Memory block number for Level 1: 4

Memory block number for Level 2: 8

Memory block number for Level Master: 12

DATA EXPECTED:

Memory expected data as: ☒ ASCII ☐ Decimal

Data expected for Level 1: changeData1

When changing L1 expected memory via CO require increment seq? (read manual for info): ☒ No ☐ Yes

Data expected for Level 2: changeData2

When changing L2 expected memory via CO require increment seq? (read manual for info): ☒ No ☐ Yes

Data expected for Level Master: changeData3

ACCESS KEY SETUP:

Memory keys as: ☒ ASCII ☐ Decimal

Key Level 1: change

When changing key via CO require increment seq? (read manual for info): ☒ No ☐ Yes

Key Level 2: change

Key Level Master: change

Figure 4: Memory Access Configurations(Local Authentication & Card’s Memory Content)

The description of each of the parameters is found in Table 6.

Table 6: Description of parameters from Memory Access Configurations(Local Authentication & Card's Memory Content).

Parameter	Description	Values
<b>Memory block number for Level #</b>	The card's block to be used for reading and checking for it's data content	Min: 0 Max: 255
<b>Memory expected data as</b>	Selects how the "Data expected for Level #" is going to be entered	- *ASCII - Decimal
<b>Data expected for Level #</b>	The data that is expected to be in the found in the memory block that was read; if the read data matches this parameter the card is considered as authenticated	[14 character/bytes]
<b>When changing L# expected memory via CO require increment seq?</b>	When set to "Yes" and when attempting to change the expected memory data via CO, the newly received data must be greater than the currently set, otherwise it will be rejected	- Yes - *No
<b>Memory keys as</b>	Selected how the "Key Level #" is going to be entered	- *ASCII - Decimal
<b>Key Level #</b>	Defines the key to be used while attempting to access the sector of the memory block to read	[6 character/bytes]
<b>When changing key via CO require increment seq?</b>	When set to "Yes" and when attempting to change the memory block access key via CO, the newly received access key must be greater than the currently set, otherwise it will be rejected	- Yes - *No

Case the parameter "**Authentication point**" is set to "Remote" the section "DATA EXPECTED" won't be displayed; this is due to the fact that the device won't have the responsibility to assert about the authenticity of the card's credentials, it will only have the duty of attempting to read the defined memory block address by using the defined memory key for each of the setup levels.

When setting "**Memory block number for Level #**" one should care about the card's memory layout and select a writable block. This allows to map up to Mifare 4K which has 256 block (from 0 to 255) (refer to Appendix D -). In the Figure 5 this parameter sets the card compartment.

An authenticated card is expected to have "**Data expected for Level #**" value written in "**Memory block number for Level #**" and locked with "**Key Level #**".



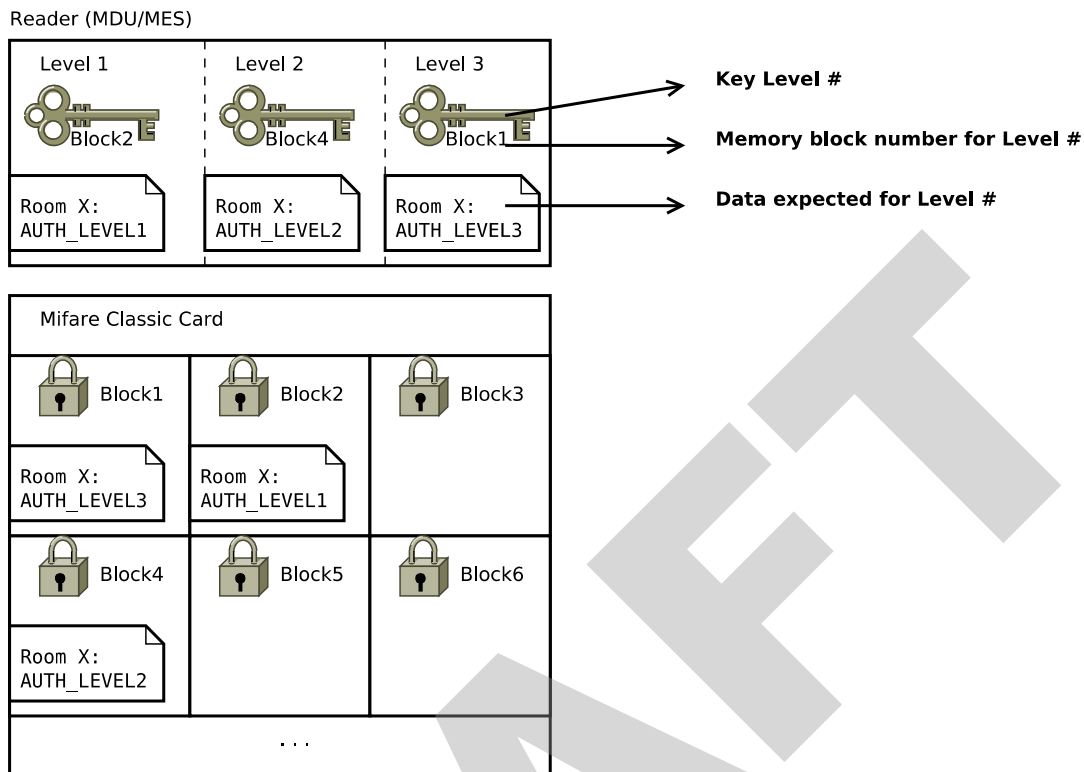


Figure 5: Memory access configurations

It is possible to change the value “**Data expected for Level #**” for levels 1 and 2 on-the-fly via COs “Authentication - [I] Level 1 expected Auth. Data” and “Authentication - [I] Level 1 expected Auth. Data” respectively. This can be used, for example, in case it is desired to change the room’s “credentials” for every new customer; in this case, for minimizing the risk of replay attacks<sup>9</sup> the parameter “**When changing L# expected memory via CO require increment seq?**” can be enabled (this solution works best by also enabling “**Use Comm. Obj. encryption?**” and activating it to “**Encrypt CO 80 ‘Level 1 expected Auth.’**” and “**Encrypt CO 81 ‘Level 2 expected Auth.’**”): every new expected authentication data that is sent to the MDU/MES must be higher than the previously sent, otherwise it will be ignored. The idea is that, if a foreign observer spies on the bus and receives one of the frames setting expected authentication data, if he tries to use it afterward by re-sending it, it will be ignored because it will be equal (or lower in case new authentication data was given) and MDU/MES will discard it. When using encryption, the authentication data frame has 12 useful bytes, so if the password increments by steps of one there are  $2^{(12 \cdot 8)}$ , approximately  $7.9228 \times 10^{28}$  different possible expected data; however notice that it is not required to increment by only one, the steps may be irregular. Notice that the observer won’t be able to see the data and guess it

9 “A replay attack is a category of network attack in which an attacker detects a data transmission and fraudulently has it delayed or repeated. The delay or repeat of the data transmission is carried out by the sender or by the malicious entity, who intercepts the data and retransmits it.” [source: <https://www.techopedia.com/definition/21695/replay-attack>]

is increasing if CO encryption is used; if encryption is used and changing the MDU/MES won't consider the message as well since the CRC of the frame will fail. If for some reason the sequence needs to be reset, this can be done by sending the maximum value (12 bytes with 0xFF) followed by the minimum value (12 bytes with 0x00).

In Figure 6 it's represented an activity diagram showing a situation where a foreign identity (spy) gets access to the bus and intercepts a message that changes the Level 1 expected Authentication Data; the intercepted frame is encrypted however, so it makes it impossible to manipulate it (any modification in the intercepted frame will result in corruption and consequently rejection by the MDU/MES). The spy can however re-send it, pretending to be the reception's server; in this case the re-play of the frame will also be rejected as its sequence number is lower or equal to the previous one.

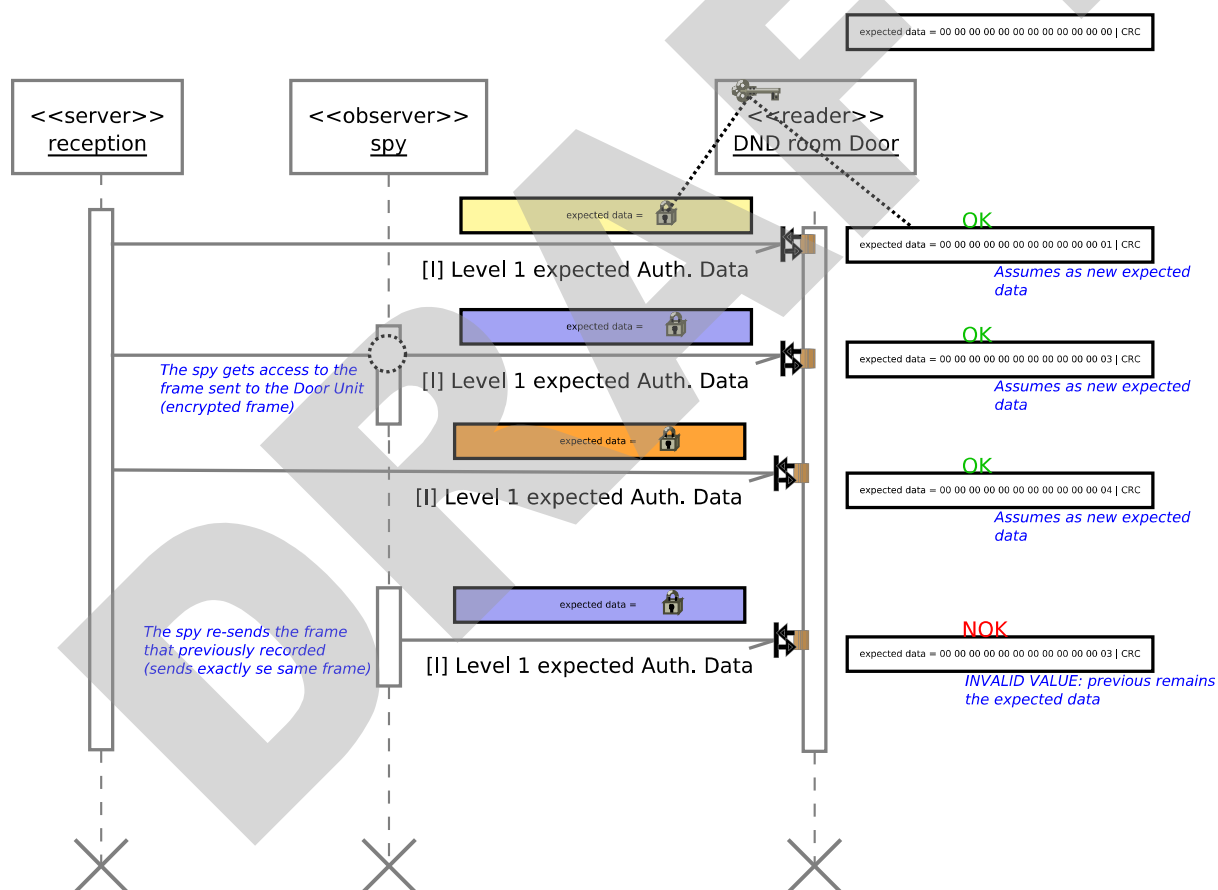


Figure 6: Example of “When changing L# expected memory via CO require increment sequence”

Similarly with the previous, also the card's memory access key (just for level 1 authentication) can be updated at run-time over the KNX bus via CO. For this, the parameter “**When changing key via CO require increment seq?**” must be set to “Yes”; the operation is similar with the one described for the

parameter “**When changing L# expected memory via CO require increment seq?**”, just in this case the number of relevant bytes are 6.

DRAFT

### 4.1.5 DND Confs – Authentication Ids Configuration

This page contents depends on the settings made on the “Authentication” page. The most relevant parameters affecting this page are “Authentication point” and “Authentication data” (also when “Accept all cards” is set to “Yes” this configuration page is suppressed); this page is only visible if “Authentication data” from the “Authentication” page is set to “Card’s ID” and “Authentication point” set to “Local”. In this page it is possible to enable groups of IDs that will be treated as authenticated: three parameters exist in this page (seen in Figure 7) the allows to enable up to 10 Level 1 IDs, 5 Level 2 and 5 Level Master IDs. Per enabled ID a new tab is created were the authentic ID can be defined.

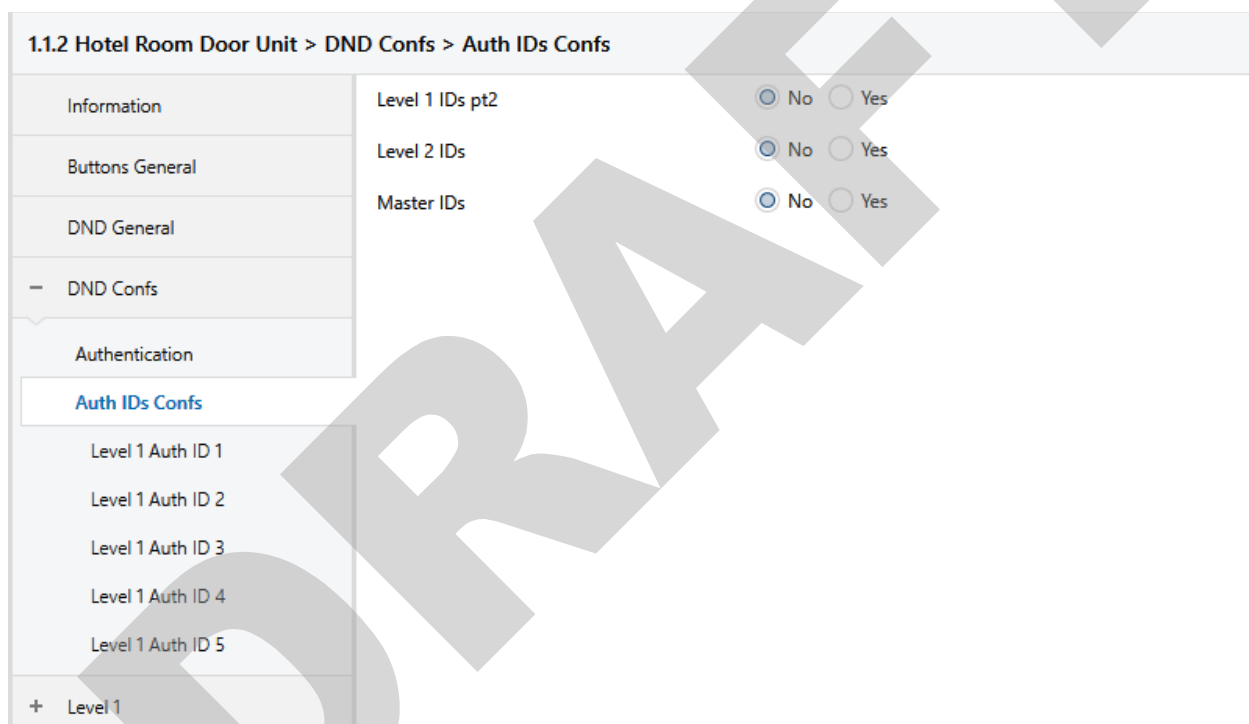


Figure 7: Authenticated Card IDs’ Configuration page (Local Authentication & Card’s ID)

In Figure 8 it is shown a configuration page for setting an authorized card ID; in this case setting a Level 1 authorized ID. It is possible to setup up to:

- Level 1: 10 card IDs
- Level 2: 5 card IDs
- Master: 5 card IDs

1.1.2 Hotel Room Door Unit > DND Confs > Level 1 Auth ID 1

Information	Byte 1	0
Buttons General	Byte 2	0
DND General	Byte 3	0
DND Confs	Byte 4	0
Authentication	Byte 5	0
Auth IDs Confs	Byte 6	0
Level 1 Auth ID 1	Byte 7	0
Level 1 Auth ID 2	Byte 8	0
Level 1 Auth ID 3	Byte 9	0
Level 1 Auth ID 4	Byte 10	0
Level 1 Auth ID 5		
Level 1		

Figure 8: Configuration of one authenticated ID (Local Authentication & Card's ID)

When setting the card's ID the **Byte 1** is the most significant byte, this means, when the card's ID is read, and output like (in hexadecimal, for case of single size ID):

70 BD BF 9F Card's ID in hexadecimal  
 112 189 191 159 Card's ID in decimal  
**B1 B2 B3 B4** Byte's correspondence

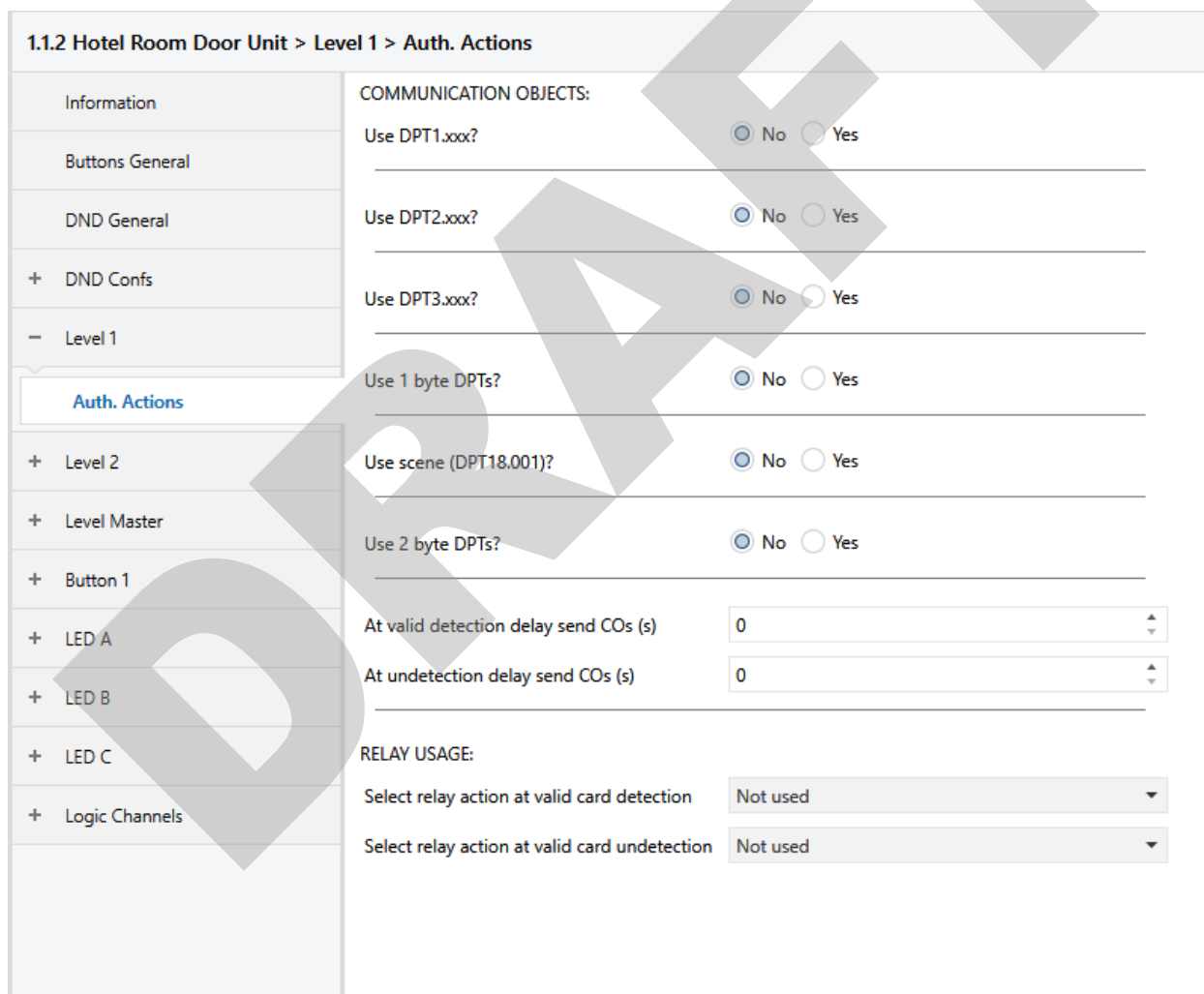
is expected; so the bytes would be filled like (converting the hexadecimal value to decimal, for example with windows calculator):

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
112	189	191	159	0	0	0	0	0	0

## 4.2 Authentication Actions

For each authentication Level it is possible to setup different actions upon presentation of an authenticated card. This means that, when a card is on the reading field, the device will check whether the card is authenticated, and if yes to which authentication Level it belongs; depending on the authentication Level of the card it will react according to the settings made for that Level “Auth. Actions” page.

An example of an authentication actions’ page for a Level (Level 1, in this case, the other Levels are identically configured) is presented in Figure 9.



**1.1.2 Hotel Room Door Unit > Level 1 > Auth. Actions**

Information	COMMUNICATION OBJECTS:	
Buttons General	Use DPT1.xxx?	<input checked="" type="radio"/> No <input type="radio"/> Yes
DND General	Use DPT2.xxx?	<input checked="" type="radio"/> No <input type="radio"/> Yes
+ DND Confs	Use DPT3.xxx?	<input checked="" type="radio"/> No <input type="radio"/> Yes
- Level 1	Use 1 byte DPTs?	<input checked="" type="radio"/> No <input type="radio"/> Yes
Auth. Actions	Use scene (DPT18.001)?	<input checked="" type="radio"/> No <input type="radio"/> Yes
+ Level 2	Use 2 byte DPTs?	<input checked="" type="radio"/> No <input type="radio"/> Yes
+ Level Master	At valid detection delay send COs (s)	0
+ Button 1	At undetection delay send COs (s)	0
+ LED A	RELAY USAGE:	
+ LED B	Select relay action at valid card detection	Not used
+ LED C	Select relay action at valid card undetection	Not used
+ Logic Channels		

Figure 9: Default configuration page for Authentication Actions

Table 7: Description of parameters from Authentication Actions.

Parameter	Description	Values
<b>Use DPT1.xxx?</b>	Selects the usage DPT1.x COs for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is detected	- *OFF - ON
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is undetected	- *OFF - ON
<b>Use DPT2.xxx?</b>	Selects the usage DPT2.x COs for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is detected	- *No Control, 0 - No Control, 1 - Control, 0 - Control, 1
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is undetected	- *No Control, 0 - No Control, 1 - Control, 0 - Control, 1
<b>Use DPT3.xxx?</b>	Selects the usage DPT2.x COs for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Direction</b>	Together with "Amount" parameter defines the value shall be sent when an authenticated Level # card is detected	- *Up   Decrease - Down   Increase
<b>Amount</b>	Together with "Direction" parameter defines the value shall be sent when an authenticated Level # card is detected	- *Break - 1.5%; 3%; 6.25%; 12.5%; 25%; 50%; 100%
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Direction</b>	Together with "Amount" parameter defines the value shall be sent when an authenticated Level # card is undetected	- *Up   Decrease - Down   Increase
<b>Amount</b>	Together with "Direction" parameter defines the value shall be sent when an authenticated Level # card is undetected	- *Break - 1.5%; 3%; 6.25%; 12.5%; 25%; 50%; 100%

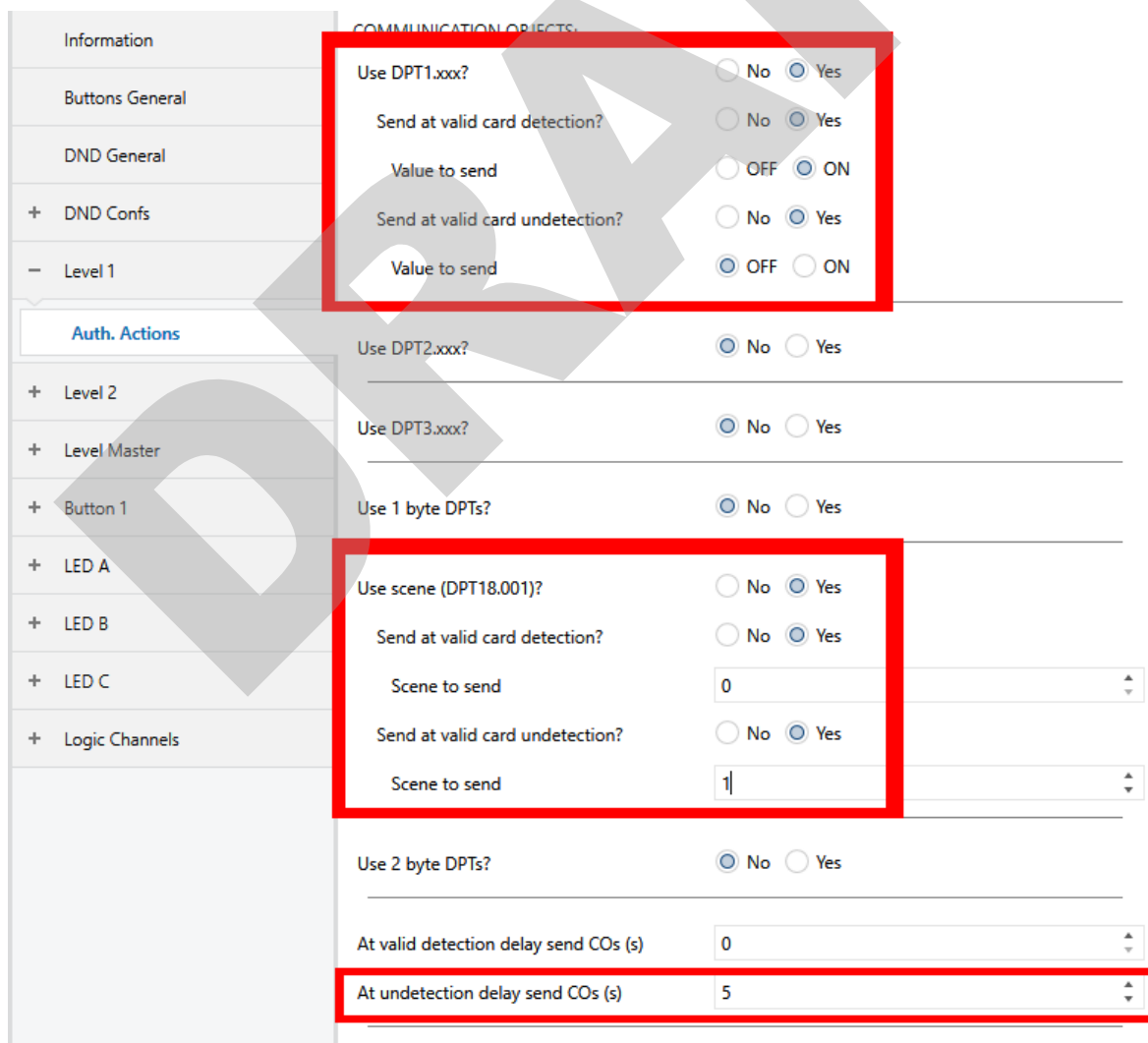
<b>Use 1 byte DPTs?</b>	Selects the usage of 1 byte length COs for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Select type</b>	Defines the DPT of the 1 byte CO to be used	- *DPT4 (char) - DPT5 (1byte unsigned) - DPT6 (1byte signed)
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is detected	[depends on "Select type"]
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is undetected	[depends on "Select type"]
<b>Use scene (DPT18.001)?</b>	Selects the usage of DPT18.001 for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is detected	Min: 0 Max: 63
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is undetected	Min: 0 Max: 63
<b>Use 2 byte DPTs?</b>	Selects the usage of 1 byte length COs for sending messages when card detection events occur for the Level #	- *No - Yes
<b>Select type</b>	Defines the DPT of the 1 byte CO to be used	- DPT7 (2byte unsigned) - DPT8 (2byte signed) - *DPT9 (KNX Float)
<b>Send at valid card detection?</b>	When a Level # valid card is presented will make the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is detected	[depends on "Select type"]
<b>Send at valid card undetection?</b>	When a Level # valid card is removed from the reading field, the CO to send a message if this is set to "Yes"	- *No - Yes
<b>Value to send</b>	Which value shall it send when an authenticated Level # card is undetected	[depends on "Select type"]
<b>At valid detection delay send COs</b>	Sets the delay between Level # authenticated card's detection and COs' emission to the bus	Min: 0 seconds Max: 255 seconds
<b>At undetection delay send COs</b>	Sets the delay between Level # authenticated card's undetection and COs' emission to the bus	Min: 0 seconds Max: 255 seconds
<b>Select relay action at valid card detection</b>	Sets the relay usage when a Level # authenticated card is detected	- *Not used - On - Off - Pulse On - Pulse Off



<b>Select relay action at valid card undetection</b>	Sets the relay usage when a Level # authenticated card is undetected	<ul style="list-style-type: none"> <li>- *Not used</li> <li>- On</li> <li>- Off</li> <li>- Pulse On</li> <li>- Pulse Off</li> </ul>
--	--	---

It is possible to activate more than one of the COs to send when a valid card is presented and/or removed. Also, each of the three available levels owns its own set of COs and parameters, meaning this that for each authorization level a different set of actions can be configured.

Imagine a situation in which it is wanted to trigger *Scene 1* (costumer entrance scenario) when the costumer's card (level 1) is presented and to trigger *Scene 2* (exit scenario) when card is removed; additionally it is wanted to send ON message to some indication LED when costumer's card is presented and OFF when removed. It's also a requirement to send the undetection messages (*Scene 2* and *OFF*)  $D_{und}=5$  seconds after the card being removed. This can be achieved with settings as:



**COMMUNICATION OBJECTS:**

Use DPT1.xxx? ☐ No ☒ Yes

Send at valid card detection? ☐ No ☒ Yes

Value to send ☐ OFF ☒ ON

Send at valid card undetection? ☐ No ☒ Yes

Value to send ☒ OFF ☐ ON

---

Use DPT2.xxx? ☒ No ☐ Yes

---

Use DPT3.xxx? ☒ No ☐ Yes

---

Use 1 byte DPTs? ☒ No ☐ Yes

---

Use scene (DPT18.001)? ☐ No ☒ Yes

Send at valid card detection? ☐ No ☒ Yes

Scene to send

Send at valid card undetection? ☐ No ☒ Yes

Scene to send

---

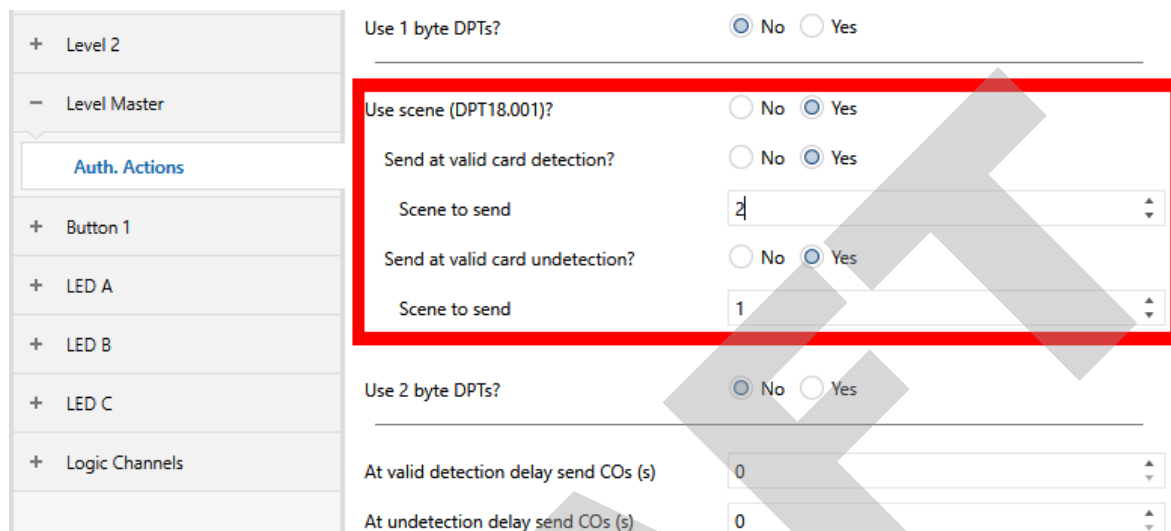
Use 2 byte DPTs? ☒ No ☐ Yes

---

At valid detection delay send COs (s)

At undetection delay send COs (s)

The same device, when Master card is presented, it is desired to trigger *Scene 3* (master entrance scenario) and when it's removed to send immediately *Scene 2* (exit scenario). For this make settings like this:



In Figure 10 one can see the representation of a possible scenario of two different cards being presented and removed for the previously described settings.

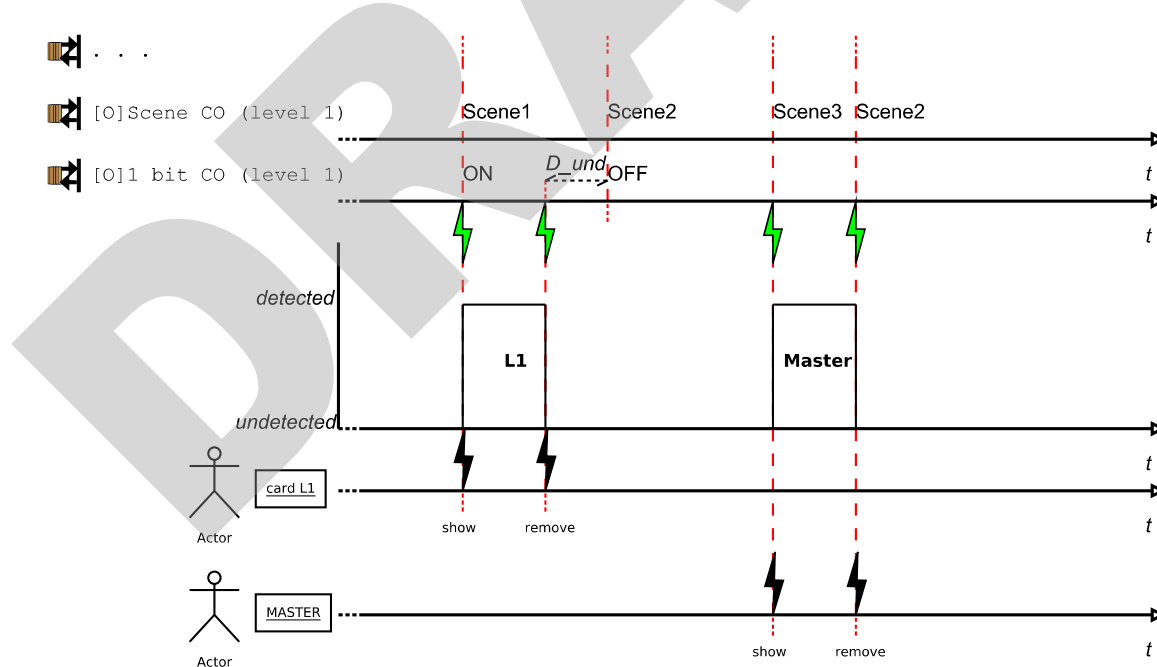


Figure 10: Example of Auth. Actions

### 4.3 Logic Channels

This function block may operate independently from the previously described function block; as a matter of fact, the logic channels is a composition of four independent function blocks. They are intended to solve common installation challenges involving task automation and logic.

Each of the four available logic channels can be configured in one of the following operating modes:

- Copy and forward
- Logic operation(binary)
- Comparison
- Mathematical operation

#### 4.3.1 Copy and Forward

This mode is intended to proportionate you, in certain way, to have one CO sending for more than one Group Address. As defined in KNX standard, one CO can send only to one Group Address. In this way, to allow you to have one CO sending to more than one Group Address it was implemented the “Copy and Forward” function in the Logic Channels.

When assigning the COs to their Group Addresses you should place the input CO in the Group Address from where you want to copy, and place the output in the Group Address to which you want the value to be copied to.

In Figure 11 is presented the configuration page for this function. In Table 8 all the parameters are described in detail.

**IMPORTANT:** if “Emission Delay (s)” is set to a value different than 0s, and more than one “DPT# Copy” is in use (in the same logic channel), than you must be aware that, the emission delay will count from the last received input, and once expired send all waiting “DPT# Copy”. If “DPT[X] Copy” and “DPT[Y] Copy” are in use in the same Logic Channel (say Channel A): “Input DPT[X]” receives a message, starting the timer for emission delay ( $T_D$ ); meanwhile, before  $T_D$ , “Input DPT[Y]” takes a message, what will make the timer to reset, starting a new count to  $T_D$ ; when the timer expires both “Output DPT[X]” and “Output DPT[Y]” are sent. In case you want different timers for different COs you should set one Logic Channel per “DPT# Copy”.

1.1.1 Presence detector with light Regulation > Logic Channel A > CopyForward ChA

General	At power up make read request of COs? <input checked="" type="radio"/> No <input type="radio"/> Yes
Advanced	
Presence	DPT1.x copy <input checked="" type="radio"/> Not used <input type="radio"/> Use
Luminosity Measurement	DPT2.x copy <input checked="" type="radio"/> Not used <input type="radio"/> Use
Lights Control	DPT3.x copy <input checked="" type="radio"/> Not used <input type="radio"/> Use
Scenes	DPT4/5/6.x copy <input checked="" type="radio"/> Not used <input type="radio"/> Use
Advanced	DPT7/8/9.x copy <input checked="" type="radio"/> Not used <input type="radio"/> Use
HVAC Control	Emission delay (s)(Attention: see user's manual) <input type="text" value="0"/>
Logic Channels	Disable copy <input type="text" value="Not used"/>
- Logic Channel A	
CopyForward ChA	
+ Logic Channel B	

Figure 11: Default configuration page for Logic Channels – Copy and Forward

Table 8: Parameters in Logic Channel's configuration page for "Copy and Forward".

Parameter	Description	Values
At power up make read request of COs?	Affects the input COs, and defines if, at power up, the inputs should send a read request	- *No - Yes
DPT# Copy <sup>10</sup>	Affects both, input and output COs and defines if it's activated or not.	- *Not used - Use
Emission Delay (s) <sup>11</sup>	Affects the value to be sent and defines the amount of time, in seconds, that the CO will wait until sending the value.	Min: 0 s Max: 65535 s (~18,2 h)
Disable copy	Affects the values to be sent and defines if the copy and forward must occur or not.	- *Not used - If '1' - If '0'
<sup>12</sup> When Copy Re-enabled	Affects the output CO and defines its behaviour when the "copy" is re-enabled.	Possible values: Do nothing, Send if new Default: Do nothing

From Table 8 must be noticed the effect of **"When Copy Re-enabled"**. By setting this to "Send if new", it means that, if while the "DPT# Copy" is disabled via "Copy Disable" the input value

<sup>10</sup> All of the DPT# Copy parameters have the same configuration options.

<sup>11</sup> The emission delay applies to all of the DPT CO and counts from the last received.

<sup>12</sup> Just present if "Disable copy" is different than "Not used".

changes, when the copy is re-enabled (via "Copy Disable" CO) the output CO will send the last value received; otherwise, if "When Copy Re-enabled" is set to "Do nothing", after copy being re-enabled the output will not send the last received value.

When more than one "DTP# Copy" CO is activated and emission delay is set to a value different than 0 seconds, all the COs are affected by the same value, and counts from the moment that the last input received a message.

As an example, lets imagine that's intended to control two lights via dimmer actuator. The lights can be switched "On" and "Off" independently, but the dimming is wanted to be controlled at the same time to all the lights, but just if the light is "On". Normally this can't be achieved because both lights' dimming COs would be in the same Group Address and by controlling the dimming value both lights would be affected and would turn "On" a light that had been previously turned "Off" via its "ON / OFF" CO. Thanks to this function, it's made possible to achieve such a lightning control. For that one "DTP3 Copy" should be activated per light (you should notice that Control Dimming Data Point Type is 3.007, a 4 bit value). Also the "Copy Disable" should be set to "If '0'" and "When Copy Re-enabled" must be set to "Do nothing". Consider the CO associations presented in Table 9.

Table 9: Possible communication object association for the example.

GA_1	GA_2	GA_3
<ul style="list-style-type: none"> <li>Light1 - ON/OFF: input</li> <li>Button1 - ON/OFF: output</li> </ul>	<ul style="list-style-type: none"> <li>Light2 - ON/OFF: input</li> <li>Button2 - ON/OFF: output</li> </ul>	<ul style="list-style-type: none"> <li>ChannelA - Input DPT3: input</li> <li>ChannelB - Input DPT3: input</li> <li>Button3 - Dimming: output</li> </ul>
GA_4	GA_5	GA_6
<ul style="list-style-type: none"> <li>Light1 - Dimming: input</li> <li>LogicA - Dimming: output</li> </ul>	<ul style="list-style-type: none"> <li>Light2 - Dimming: input</li> <li>LogicB - Dimming: output</li> </ul>	<ul style="list-style-type: none"> <li>Light1 - Status: output</li> <li>LogicA - Copy Disable: input</li> </ul>
GA_7		
<ul style="list-style-type: none"> <li>Light2 - Status: output</li> <li>LogicB - Copy Disable: input</li> </ul>		

In Table 9 the GA\_3 is the "source" from where we want to copy (the dimming value coming from the user interface), so we place both inputs. The GA\_4 is the Group Address of the dimming of one of the lights, where we should place one of the logic channel's output; GA\_5 controls the other light, so we place the other logic channel's output.

In Figure 12 it's suggested a possible operation of the previously described installation having “When Copy Re-enable” set to “Send if new”. Just the COs from the Logic Channels are evidenced.

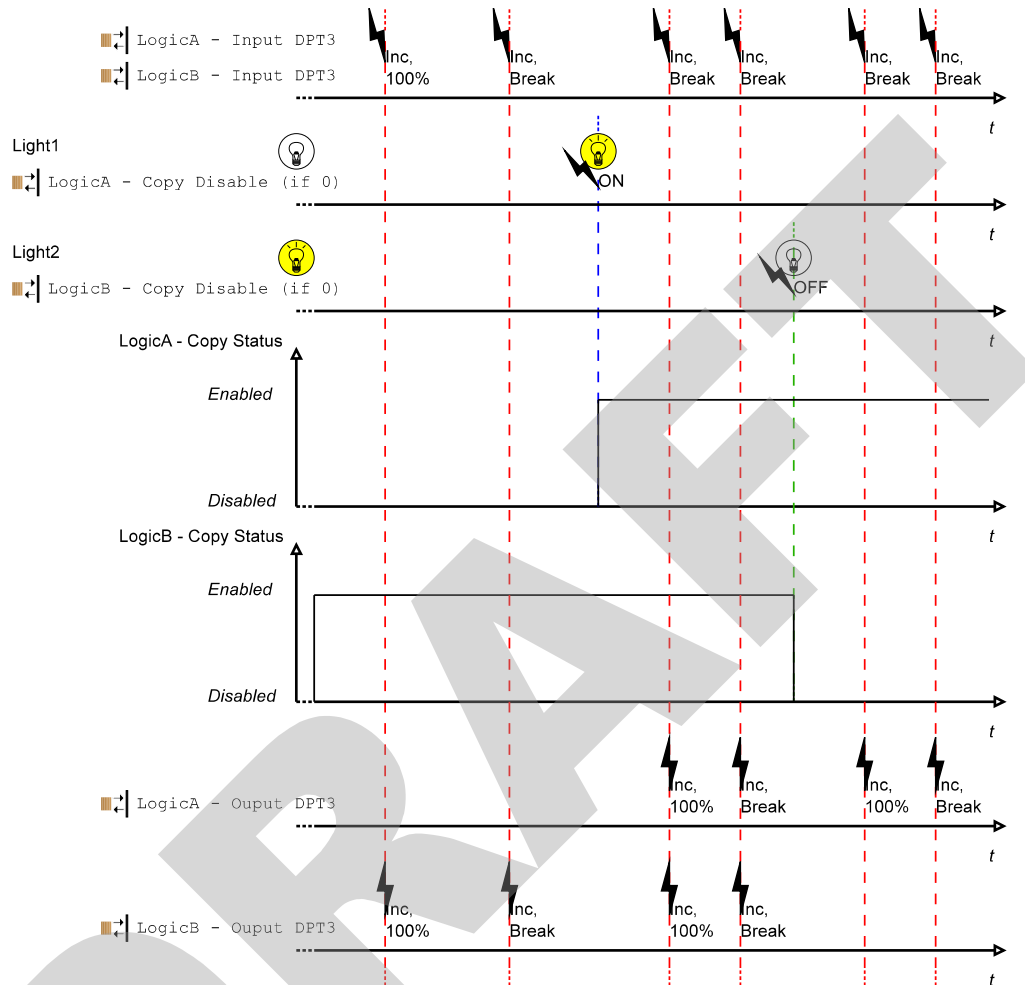


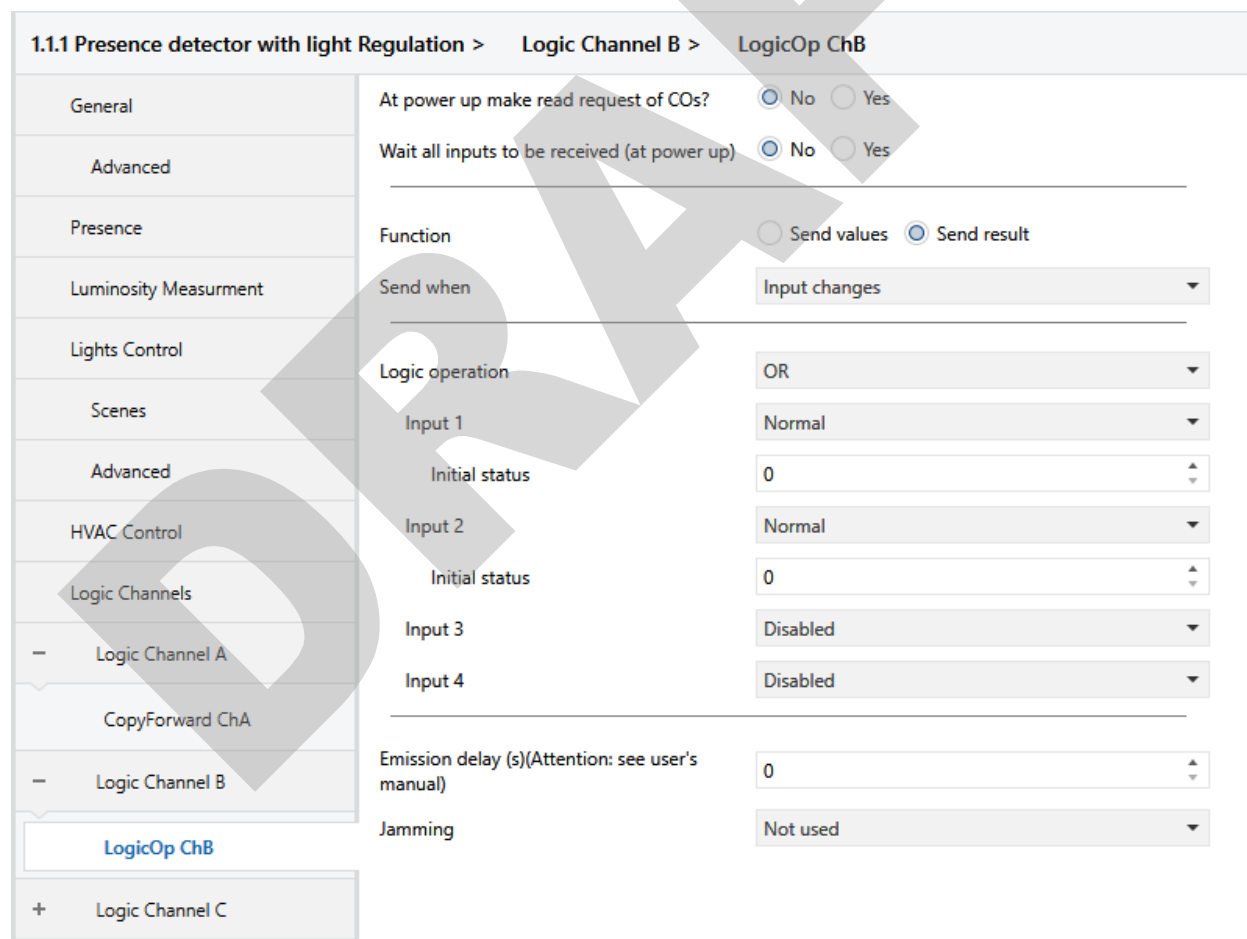
Figure 12: Copy and Forward example.

### 4.3.2 Logic Operation

This mode is intended to proportionate you the possibility of create actions when a group of COs verify a set of conditions that you predefined. The value to be sent can be selected via “Function” parameter and the sending condition is defined by defining the “Send Condition” and “Logic Operation”. In addition you need to select the inputs you want to used (also defining if its value should be negated or not).

With this mode you can define one action to happen always that a logical operation is verified, allowing you to introduce some automation to your KNX installation.

In Figure 13 is shown the default configuration page for the Logic Channels. In Table 10 the parameters are explained in detail.



1.1.1 Presence detector with light Regulation > Logic Channel B > LogicOp ChB	
General	At power up make read request of COs? <input checked="" type="radio"/> No <input type="radio"/> Yes
Advanced	Wait all inputs to be received (at power up) <input checked="" type="radio"/> No <input type="radio"/> Yes
Presence	Function <input type="radio"/> Send values <input checked="" type="radio"/> Send result
Luminosity Measurement	Send when Input changes
Lights Control	Logic operation OR
Scenes	Input 1 Normal
Advanced	Initial status 0
HVAC Control	Input 2 Normal
Logic Channels	Initial status 0
Logic Channel A	Input 3 Disabled
CopyForward ChA	Input 4 Disabled
Logic Channel B	Emission delay (s)(Attention: see user's manual) 0
LogicOp ChB	Jamming Not used
Logic Channel C	

Figure 13: Logic Channel's "Logic Operation" configuration page.

Shall it be noticed that the parameter “**Emission Delay (s)**” applies just to the last event, this means, if one emission delay is in progress but a new value to the output CO comes, the ongoing emission delay is cancelled and a new one is set for the new value (resembling the buttons operability, but

without the emission cancellable by second press (see Figure 3 in 4.1.3 to understand the effect of having new output value for emission delay before the previous being sent)). In case “Jamming” is applied to “Output”, in case output has a value to be sent after jamming, the value is sent after emission delay after jamming end.

Also important to notice the difference in the “**Jamming**” operation. In this function, when the jamming is used, you can choose if just the output is prevented to send while the jamming is set (meaning that when the jamming is cleared the output, if the value changed, will update its value on the bus) or if also the inputs are prevented from receiving its values while jamming is set (meaning that when jamming is cleared the logical channel is in the same condition that it was before the jamming).

The parameter “**Initial Result Status**” is used just for the power-up situation when the “Send condition” is “When result changes”, this because, at power up the “Result” is not yet defined what would create and ambiguous situation. If you set “Initial Result Status” to NONE, whatever the result of first operation, it considers that the result changed and the value is sent to the bus.

Table 10: Parameters in Logic Channel's configuration page for "Logic Operation".

Parameter	Description	Values
At power up make read request of COs?	Affects the input COs, and defines if, at power up, the inputs should send a read request	- *No -Yes
Wait all inputs to be received (at power up)	If enabled the logic channel won't issue any result until all the input COs have received at least 1 value	- *No -Yes
Function	Affects the output CO and defines the action to be executed.	- Send values - *Send result
DPT #	Enables/disables the usage of certain CO for a certain DPT	- *No -Yes
Send when	Affects the output CO and defines in which condition must the output be sent to the bus.	- *Input changes - Result changes (initial result=NONE) - Result changes (initial result=1) - Result changes (initial result=0)
Logic Operation	Affects the operation between the inputs, affecting the result.	- None - And - *Or - Xor
Input # <sup>13</sup>	Affects each of the input COs and defines if it's disabled or used.	- Disabled - Normal - Inverted
Initial status	Defines the input initial status	0/1
Emission Delay (s)	Affects the value to be sent and defines the amount of time, in seconds, that the CO will wait until sending the value.	Min: 0 s Max: 65535 s (~18,2 h)
Jamming	Affects the values to be sent and defines if the COs	- *Not used

<sup>13</sup> The “Input 1” to “Input 4” parameters have the same options.



	can be prevented from sending/receiving their values when an event trigger occurs.	- If '1' - If '0'
<b>When unjammed send newest value?</b>	If active, when unjamming is made the most updated values are sent	- *No -Yes

As a title of example lets consider the following situation: in one home it's intended to turn “On” a small lamp when the kitchen, hall and living room lights are turned off at the same time, but whenever one is turned “On”, the small lamp must be turned “Off”. For this kind of situations you may use Logic Channels configured in “Logic Operation” mode for achieving the solution.

For accomplish the previous described example, you could set the Logic Channel's function to “Switch (ON/OFF)”, with “Send Condition” set to “When result changes” and “Initial Result Status” set to “FALSE” (meaning that the first message will be sent to the bus when the lamp must be turned “On”). You would also allocate one of the inputs per state of light actuator channel (one in Kitchen status indication ( $I_1$ ), other in Hall status indication ( $I_2$ ) and other in Living Room status indication( $I_3$ )). As we want “Result” to be “TRUE” when all the lights are “OFF” (“FALSE”), the logic operation to preform must be the logical AND of the inverse of the inputs:

$$Result = \neg I_1 \wedge \neg I_2 \wedge \neg I_3$$

where "  $\wedge$  " is the logical conjunction operator (AND), and ' $\neg$ ' is the logical negation (NOT) (see Appendix A -Logic operations). Having this, we can consider the truth table of our example (see Table 11).

*Table 11: Truth table for 3 inputs*

$I_1$	$I_2$	$I_3$	$\neg I_1$	$\neg I_2$	$\neg I_3$	$\neg I_1 \neg I_2 \neg I_3$
0	0	0	1	1	1	1
0	0	1	1	1	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
1	0	0	0	1	1	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	0

In Figure 14 is presented a possible operation case for the described situation above. The situation can be interpreted as follows: initially the lights of Kitchen and Living Room are “On” and the lights of the Hall are “Off”. Let's consider that the device has configured to “Read objects” at power up. When the the bus is powered up the inputs will make read requests, and later will receive their answers, but since the initial result status was configured to “FALSE”, and the result after taking the answers is still

“FALSE”, no message is sent to the bus. Later the Kitchen's light is turned “Off”, but since the Living Room's light still “On”, the result is still “FALSE”. When later the Living Room's light is turned “Off”, all the lights are “Off”, which fulfils our condition to turn “On” the small lamp. In this moment the result becomes “TRUE” what makes “ON / OFF” CO to send “On” to the small lamp. If any lamp is turned “On”, say the Hall's light, the result becomes “FALSE” and “ON / OFF” CO sends “Off” to the small lamp.

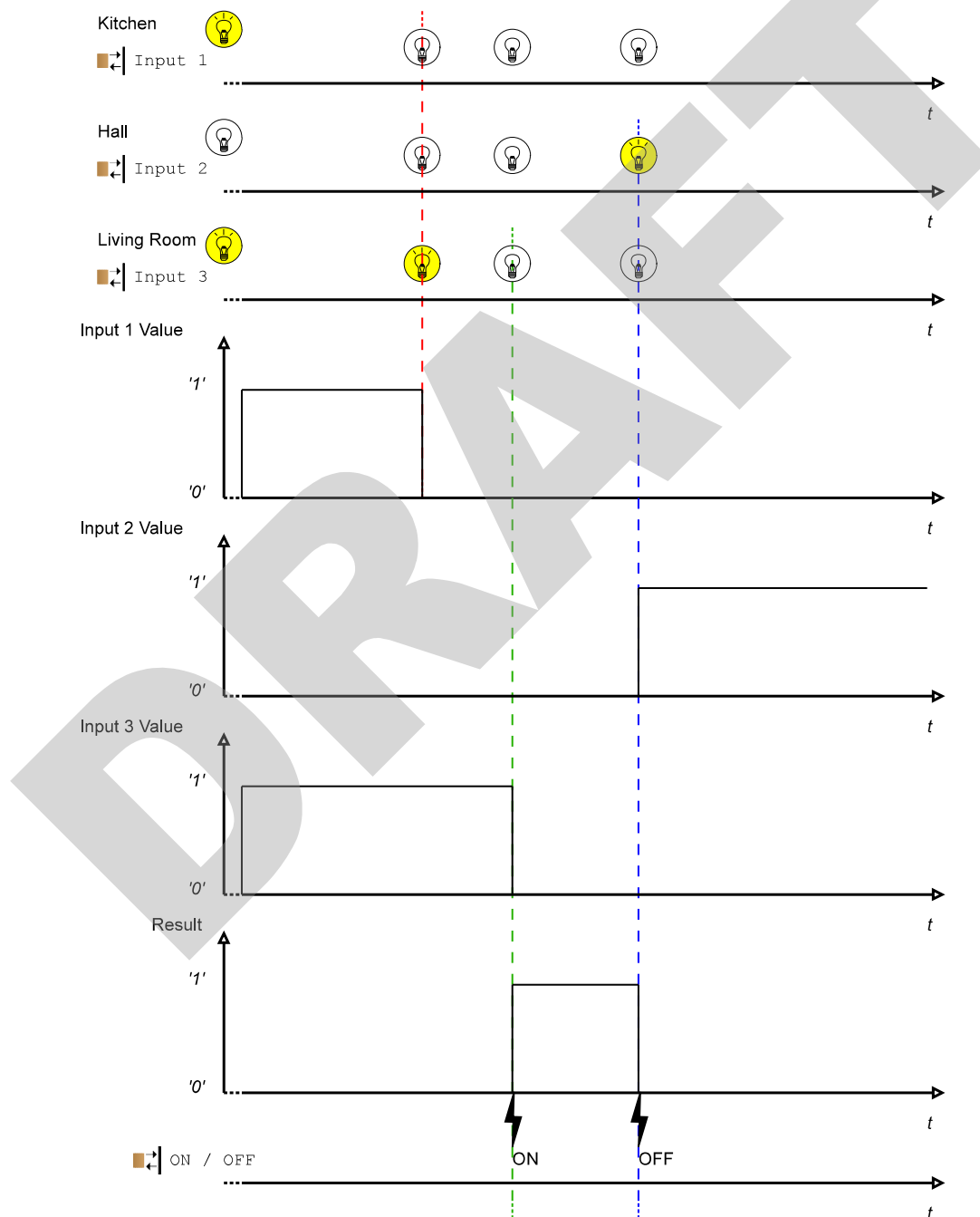


Figure 14: Operation example of "Logic Operation".

Let's suppose now that there's a control button for the small lamp, and it's wanted that when the lamp is turned "On" manually, no matter which is the status of the rest of the lightning, the small lamp must remain "On". For achieving this the Logic Channel's "Jamming" CO may be used. For simplicity let's consider only the Kitchen and Hall's lights as inputs. Is also wanted to make the small lamp to become again controlled according to the Kitchen and Hall's lightning, taking in that moment the value according to the current lightning status. In this way, the small lamp must be controlled via two Group Addresses: one for the manual control (which will send only "On" commands) and other for the logic channel control. For this, you may change "Jamming" to "If '1'" and "Apply jamming to" set to "Jamming Output". When the jamming is applied to the output, the result is kept updating, however the output CO will not send any messages, but as soon as jamming is cleared, if result has changed since it's state before being jammed, the output CO will send its new value, if the result value is still the same the output CO will not send its value.

Please refer to Table 12 for a possible association of COs that would accomplish the solution for the previously exposed problem.

*Table 12: Possible communication object association for the example.*

GA_1	GA_2	GA_3
<ul style="list-style-type: none"> <li>↔ Kitchen - ON/OFF: input</li> <li>↔ Button1 - ON/OFF: output</li> </ul>	<ul style="list-style-type: none"> <li>↔ Hall - ON/OFF: input</li> <li>↔ Button2 - ON/OFF: output</li> </ul>	<ul style="list-style-type: none"> <li>↔ SmallLamp - ON/OFF: input</li> <li>↔ LogicA - ON/OFF: output</li> </ul>
GA_4 <sup>14</sup>	GA_5 <sup>14</sup>	GA_6
<ul style="list-style-type: none"> <li>↔ SmallLamp - ON/OFF: input</li> <li>↔ Button3 - ON/OFF ChA: output</li> <li>↔ LogicA - Jamming: input</li> </ul>	<ul style="list-style-type: none"> <li>↔ Button3 - ON/OFF ChB: output</li> <li>↔ LogicA - Jamming: input</li> </ul>	<ul style="list-style-type: none"> <li>↔ Kitchen - Status: output</li> <li>↔ LogicA - Input1: input</li> </ul>
GA_7		
<ul style="list-style-type: none"> <li>↔ Hall - Status: output</li> <li>↔ LogicA - Input2: input</li> </ul>		

In Figure 15 is shown an operation example for the previously described case. In this Figure just the COs related with the logic channel have been considered.

<sup>14</sup> The Button3, if from MSW100X-PL, would, for example, be configured in "2-Channel mode" set with "ON" to Channel A and "OFF" to Channel B.

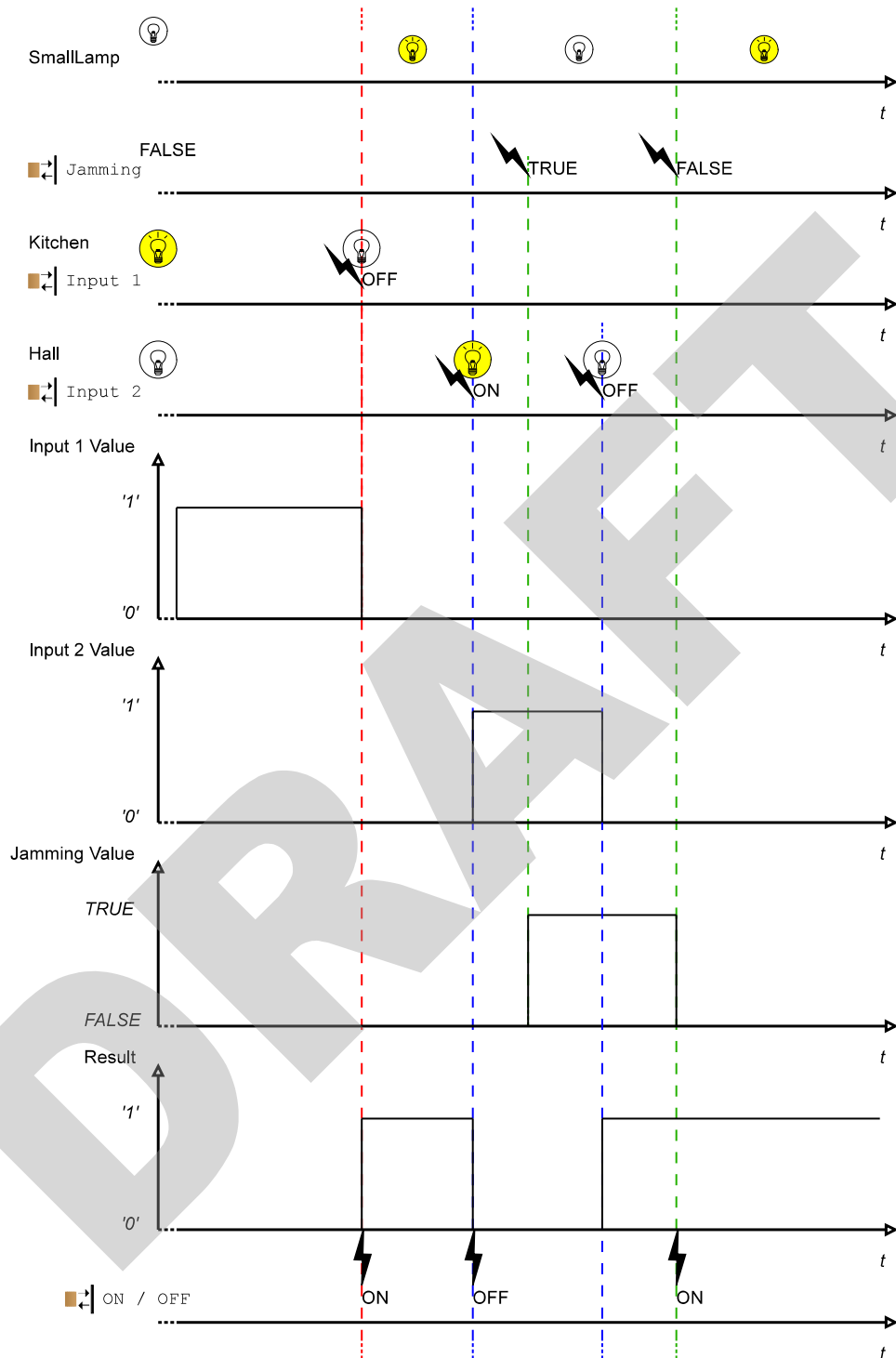


Figure 15: Logic Operation example.

### 4.3.3 Comparison

Automation can be extended by performing automatic operation when a certain variable (CO) is within certain values. This module allows the creation of automatic rules of this sort; when a **CO value matches and/or not matches a comparison a value or the value of other CO** (or group of values) from a pre-defined DPT (or pre-defined DPTs) can be issued to the bus.

1.1.1 Presence detector with light Regulation > Logic Channel C > Comparison ChC

General	At power up make read request of COs? <input checked="" type="radio"/> No <input type="radio"/> Yes	
Advanced	SEND WHEN COMPARISON TRUE:	
Presence	DPT1	<input checked="" type="radio"/> No <input type="radio"/> Yes
Luminosity Measurement	DPT2	<input checked="" type="radio"/> No <input type="radio"/> Yes
Lights Control	DPT3	<input checked="" type="radio"/> No <input type="radio"/> Yes
Scenes	1 Byte	<input checked="" type="radio"/> No <input type="radio"/> Yes
Advanced	2 Byte	<input checked="" type="radio"/> No <input type="radio"/> Yes
HVAC Control	SEND WHEN COMPARISON FALSE:	
Logic Channels	DPT1	<input checked="" type="radio"/> No <input type="radio"/> Yes
Logic Channel A	DPT2	<input checked="" type="radio"/> No <input type="radio"/> Yes
CopyForward ChA	DPT3	<input checked="" type="radio"/> No <input type="radio"/> Yes
Logic Channel B	1 Byte	<input checked="" type="radio"/> No <input type="radio"/> Yes
LogicOp ChB	2 Byte	<input checked="" type="radio"/> No <input type="radio"/> Yes
Logic Channel C	Compare with <input checked="" type="radio"/> Value <input type="radio"/> Communication Object	
Comparison ChC	Data Point Type for comparison (A)	DPT1.x
Logic Channel D	Value to compare with (B)	0
	Comparison operation ( A [op] B )	Equal (==)
	Emission delay (s)(Attention: see user's manual)	0
	Jamming	Not used

Figure 16: Logic Channel's "Comparison Operation" configuration page.

Table 13: Description of parameters from Logic Channel's "Comparison Operation"

Parameter	Description	Values
<b>At power up make read request of COs?</b>	Affects the input COs, and defines if, at power up, the inputs should send a read request	- *No - Yes
<b>Send when Comparison TRUE DPT#</b>	Enables/Disables the usage of the DPT for the case that the comparison being made results TRUE	- *No - Yes
<b>Value</b>	Defines the value to be sent when comparison results TRUE	<u>Depends on the DPT</u>
<b>Send when Comparison FALSE DPT#</b>	Enables/Disables the usage of the DPT for the case that the comparison being made results FALSE	- *No - Yes
<b>Value</b>	Defines the value to be sent when comparison results FALSE	<u>Depends on the DPT</u>
<b>Compare with</b>	Selects with what "Logic X (Compare) – [I] DPT# Input" is going to be compared; if "Value" is selected it will be compared with a constant value; otherwise a CO number must be given	- *Value - Communication Object
<b>Data Point type for comparison (A)</b>	Selects the DPT of "Logic X (Compare) – [I] DPT# Input"	- *DPT1.x ... - DPT9.x
<b>Value to compare with (B)</b>	When "Compare with" is "Value" defines the value for comparison	<u>Depends on Data Point type for comparison (A)</u>
<b>ComObj number to compare with (B)</b>	When "Compare with" is "Communication Object" defines the CO to use for comparison	<u>Must be the CO number from this device</u>
<b>ComObj type (B)</b>	Specifies the DPT of the CO selected in <b>ComObj number to compare with (B)</b>	- *DPT1.x ... - DPT9.x
<b>Comparison operation (A [op] B)</b>	Defines the comparison operator between <b>A</b> and <b>B</b>	- *Equal (==) - Different (!=) - Greater (>) - Less (<)
<b>Emission Delay (s)</b>	Affects the value to be sent and defines the amount of time, in seconds, that the CO will wait until sending the value.	<u>Min:</u> 0 s <u>Max:</u> 65535 s (~18,2 h)
<b>Jamming</b>	Affects the values to be sent and defines if the COs can be prevented from sending/receiving their values when an event trigger occurs.	- *Not used - If '1' - If '0'
<b>When unjammed send newest value?</b>	If active, when unjamming is made the most updated values are sent	- *No - Yes

As title of example let's imagine that one intends to implement a rudimentary heating thermostat function block; in the installation there's:

- a simple temperature sensor that solely sends the room temperature (via DPT9.xxx CO);
- one MPR110x-y with at least a Logic Channel unused;
- On/Off heating actuator (radiator with KNX valve).

In this setup, one could enable a Logic channel in **Comparison** mode, enabling DPT1 for “**Send when Comparison TRUE**” and “**Send when Comparison FALSE**” setting the first to “ON” and the second to “OFF”; set “**Data Point type for comparison (A)**” to “DPT9.x” and the “**Value to compare with (B)**” to the desired setpoint, for instance 24°C and setting “**Comparison operation (A [op] B)**” to “Less (<)”. In this case, when the temperature sensor issues values lower than 24°C the DPT1 output “[O]DPT1.x” will send ON message, opening the heater valve, and when it’s higher it will send OFF closing the heater valve. Please follow with Figure 17.

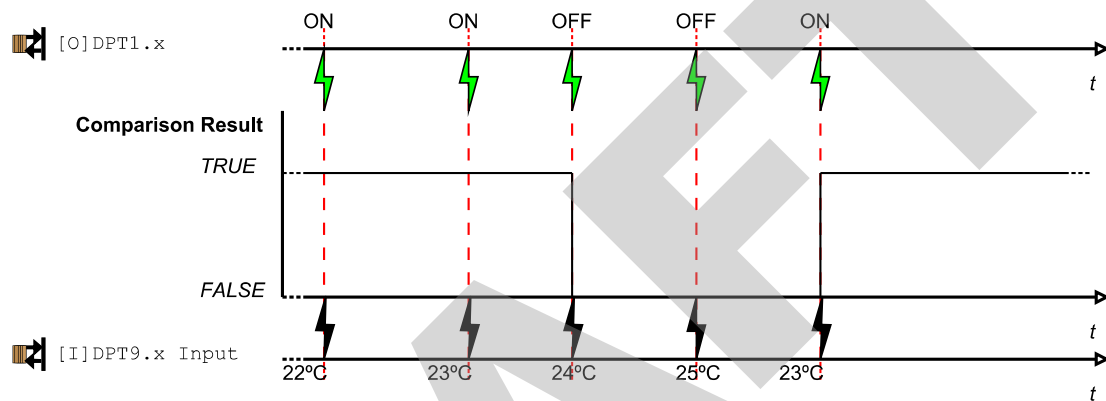


Figure 17: Example of rudimentary thermostat using Comparison (compare with value)

One interesting addition to this system would be to allow changing the setpoint. One way of achieving such would be by changing “**Compare with**” to “Communication Object”; and enabling another dummy Logic channel in “Copy and Forward” mode (for example) and enabling DPT9 CO using it as the input for the setpoint; setting “**ComObj number to compare with (B)**” to the number of the CO enabled and associated with the input of the setpoint<sup>15</sup>; setting “**ComObj type (B)**” to “DPT9.x” (or whatever type is intended to be used as the setpoint input<sup>16</sup>). To note that the comparison is only be triggered when new values are received via input of the Logic Channel. This means that changing the setpoint won’t trigger a comparison; it is required the temperature sensor to send a new value.

15 For example, if Logic Channel A was used for this purpose “Logic A – [I] DPT7/8/9.x Input (2 byte)” would be used as the input of the setpoint: Communication Object number 121.

16 Internally the comparison function block will perform type conversion in order to apply the correct comparison, even between different DPTs; for instance DPT8.x can be compared with DPT9.x

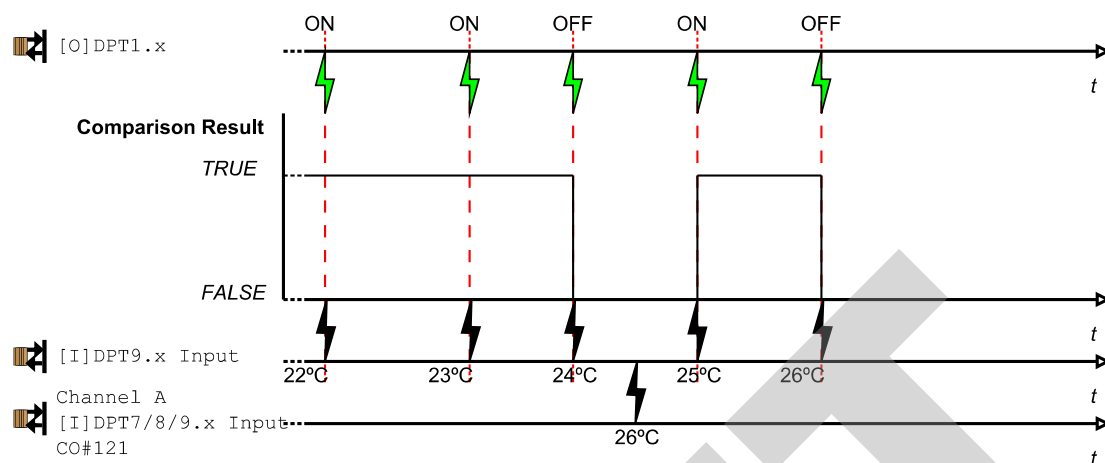
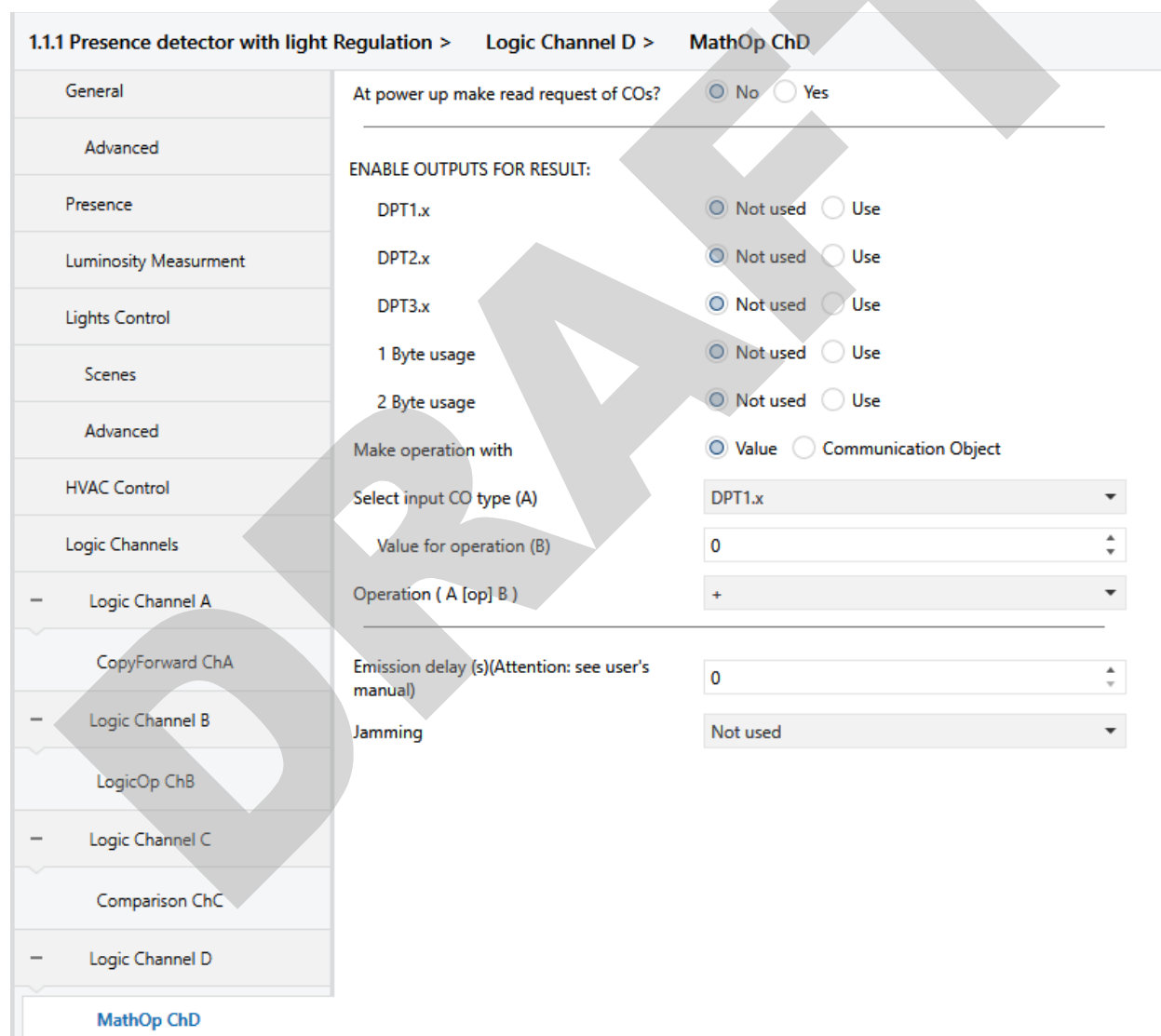


Figure 18: Example of rudimentary thermostat using Comparison (compare with Comm Object)



### 4.3.4 Mathematical operation

Often happens to be needed to apply offsets to available COs' values in order to perform certain action; often happens to be needed to scale values. This functional block is intended exactly to perform this kind of adjustments on-the-fly without using extra, expensive, computing solutions. Calculations between different DPTs is possible, sending result via different DPTs as well<sup>17</sup>. This can even be used as a way of converting between DPTs (by applying a not changing operation, for example "adding 0", and selecting a different output DTP).



**1.1.1 Presence detector with light Regulation > Logic Channel D > MathOp ChD**

**General**

At power up make read request of COs? ☒ No ☐ Yes

---

**ENABLE OUTPUTS FOR RESULT:**

DPT1.x ☒ Not used ☐ Use

DPT2.x ☒ Not used ☐ Use

DPT3.x ☒ Not used ☐ Use

1 Byte usage ☒ Not used ☐ Use

2 Byte usage ☒ Not used ☐ Use

Make operation with ☒ Value ☐ Communication Object

Select input CO type (A) DPT1.x

Value for operation (B) 0

Operation ( A [op] B ) +

---

Emission delay (s)(Attention: see user's manual) 0

Jamming Not used

---

**Logic Channels**

- Logic Channel A
  - CopyForward ChA
- Logic Channel B
  - LogicOp ChB
- Logic Channel C
  - Comparison ChC
- Logic Channel D
  - MathOp ChD**

Figure 19: Logic Channel's "Mathematical Operation" configuration page.

<sup>17</sup> Depending on the DPTs truncation may apply.

Table 14: Description of parameters from Logic Channel's "Mathematical Operation"

Parameter	Description	Values
At power up make read request of COs?	Affects the input COs, and defines if, at power up, the inputs should send a read request	- *No -Yes
Enable output for result: DPT#	Enables/Disables the usage of the DPT for sending the result of the operation; NOTE: depending on the DPT result truncation may apply	- *No - Yes
Make operation with	Selects with what "Logic X (Math) – [I] DPT# Input" is going to make operation with; if "Value" is selected it will make operation with a constant value; otherwise a CO number must be given	- *Value - Communication Object
Select input CO type (A)	Selects the DPT of "Logic X (Math) – [I] DPT# Input"	- *DPT1.x ... - DPT9.x
Value for operation (B)	When "Make operation with" is "Value" defines the value for operation	<u>Depends on Select input CO type (A)</u>
ComObj number to make with (B)	When "Make operation with" is "Communication Object" defines the CO to use for operation	<u>Must be the CO number from this device</u>
ComObj type (B)	Specifies the DPT of the CO selected in <b>ComObj number to make with (B)</b>	- *DPT1.x ... - DPT9.x
Operation (A [op] B)	Defines the mathematical operation to be made between <b>A</b> and <b>B</b>	- + (addition) - - (subtraction) - / (division) - x (multiplication)
Emission Delay (s)	Affects the value to be sent and defines the amount of time, in seconds, that the CO will wait until sending the value.	<u>Min:</u> 0 s <u>Max:</u> 65535 s (~18,2 h)
Jamming	Affects the values to be sent and defines if the COs can be prevented from sending/receiving their values when an event trigger occurs.	- *Not used - If '1' - If '0'
When unjammed send newest value?	If active, when unjamming is made the most updated values are sent	- *No -Yes

Lets suppose a situation where it is desired to get a voltage level, which is provided by a multimeter in DPT9.020 (DPT\_Value\_Volt), provided as an integer (1 byte, unsigned) in Volts; for example, 230000mV(DPT9.020) to be given as 230V(DPT5.x); this value would afterwards be used to show on a display that can only show integer values. For this one could setup the device enabling "Enable output for result: 1 Byte usage" and selecting "DPT5.x"; setting "Make operation with" to "Value"; setting "Select input CO type (A)" to "DPT9.x"; "Value for operation (B)" to 1000; "Operation (A [op] B)" to "/" (division)".

In this configuration every time a voltage value is received via "DPT9.x input (2 byte)" the channel will divide it per 1000 (rounding to closest), resulting the value in volts, will then convert it from DPT9.x to DPT5.x (casting it by truncating) and sending it to the bus via "DPT5 Math op. Result" (consider the possible sequence in Table 15).

Table 15: Example of sequence of DPT9.020 to DPT5.x using Mathematical operation

#	Source Name	Destination Address	Destination Name	DPT	Info
1	Multimeter	0/0/1	[I]DPT9.020 Voltage(mV)	9.020 voltage (mV)	\$75 7B   229867.52 mV
2	Presence detector with light Regulation	0/0/2	[O]DPT5.x (1byte unsigned value)	5.010 counter pulses (0..255)	\$E6   230
3	Multimeter	0/0/1	[I]DPT9.020 Voltage(mV)	9.020 voltage (mV)	\$75 6F   227901.44 mV
4	Presence detector with light Regulation	0/0/2	[O]DPT5.x (1byte unsigned value)	5.010 counter pulses (0..255)	\$E4   228
5	Multimeter	0/0/1	[I]DPT9.020 Voltage(mV)	9.020 voltage (mV)	\$75 81   230850.56 mV
6	Presence detector with light Regulation	0/0/2	[O]DPT5.x (1byte unsigned value)	5.010 counter pulses (0..255)	\$E7   231

Suppose other situation, in which it is desired to get the ration between the “Lights - [O] Presence Lux Setpoint ind.” and “General - [O] Luminosity” in order to show in some sort of display; in this case it is wanted the output value to be DPT9.004 (DPT\_Value\_Lux). In this case “**Enable output for result: 2 Byte usage**” and selecting “DPT9.x”; setting “**Make operation with**” to “Communication Object”; setting “**Select input CO type (A)**” to “DPT9.x”; “**ComObj number to make with (B)**” set to the CO number of “Lights - [O] Presence Lux Setpoint ind.” (35 in this case) and associate “General - [O] Luminosity” and “Logic X - [I] DPT9.x Input (2 byte)” in the same GA of “General - [O] Luminosity” and setting the “**ComObj type (B)**” also to “DPT9.x”. The “**Operation (A [op] B)**” to “/ (division)”.

With this setup for every time “General - [O] Luminosity” issues a value, “DPT9.x input (2 byte)” will receive its value, proceed to the calculation and sending it via “DPT9 Math op. Result”

## **5 OPERATION DESCRIPTION**

---

## 5 OPERATION DESCRIPTION

The MDU000x-M and MES000x-M are able to operate in various configurations, each of them implying different authentication topologies. As explained in 4.1.3 the authentication may be configured to be preformed *locally* or *remotely*.

### 5.1 Local vs. Remote authentication

When the authentication is said to be **Local** it means that the rooms' devices are responsible to alone perform a card authentication; the devices will have enough information to upon card presentation proceed to it's validation. Normally, in such topology, the credentials for a certain room would be static; this would be a great drawback, because the cloning of a card would directly result in the unconditional access to that room. In order to surpass this drawback the information (valid cards' credentials) stored in the rooms' devices can be changed on the fly over KNX bus via COs, as well as the memory access details (memory address/block and memory access key):

```
➡ [I] Level 1 expected Auth. Data
➡ [I] Level 1 Auth. Data Address
➡ [I] Auth. Access key (level 1)
```

When the authentication is said to be **Remote** it means that the rooms' devices aren't responsible *per se* of performing the authentication; they will instead transfer to the bus the relevant data (presented card's credentials) and some other element (normally an authentication server) will evaluate it; upon credentials validation it must inform the rooms' device via CO:

```
➡ [I] Validate authentication
```

Similar with the local authentication, for the remote authentication it is also possible to change the memory access details via COs:

```
➡ [I] Level 1 Auth. Data Address
➡ [I] Auth. Access key (level 1)
```

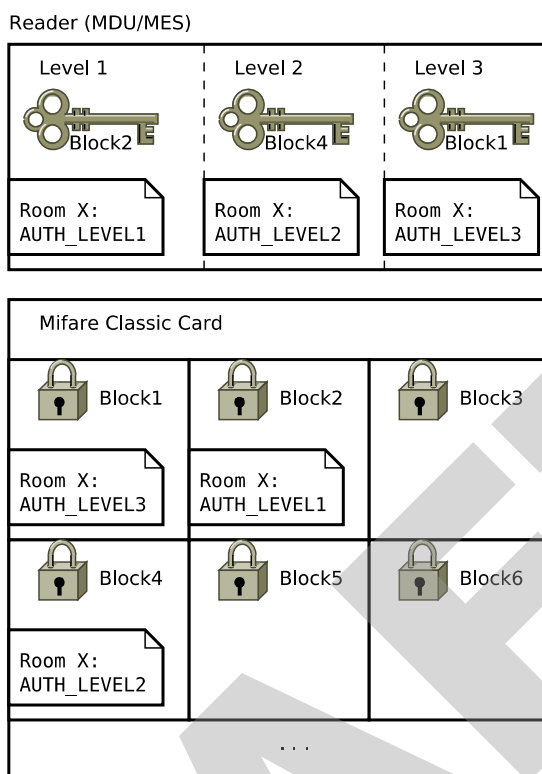


Figure 20: Local authentication

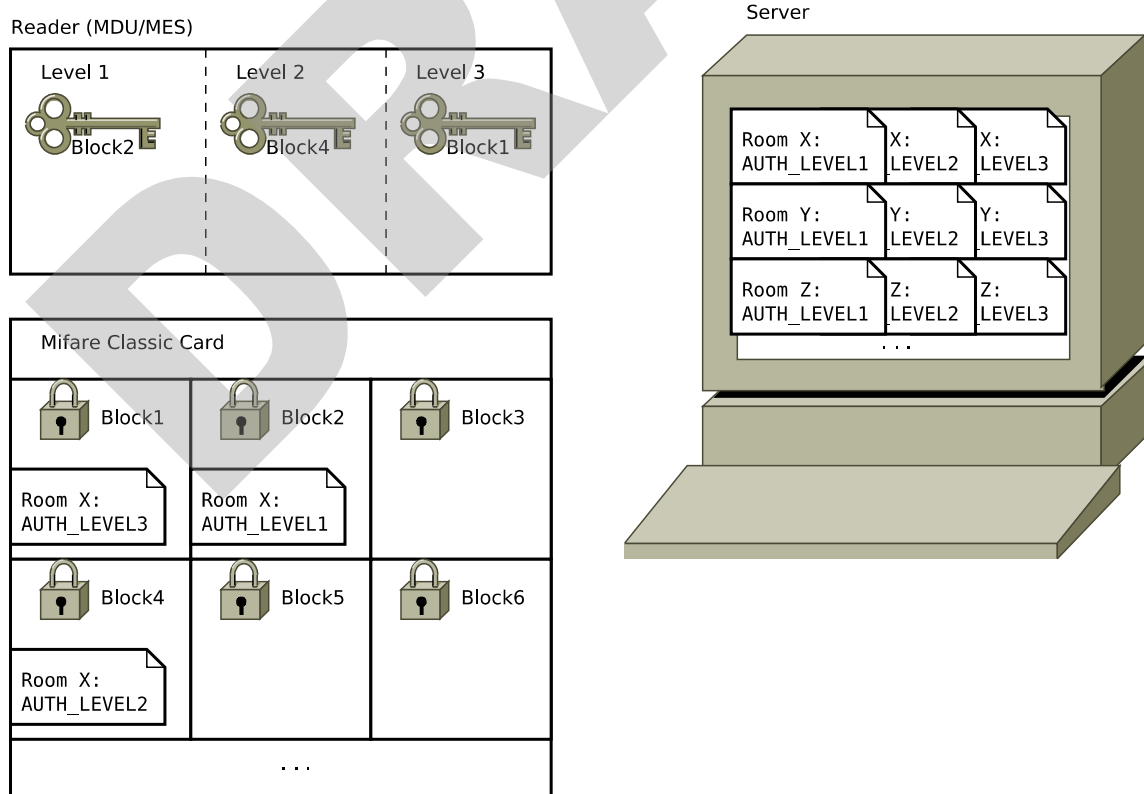


Figure 21: Remote authentication

### 5.1.1 Local authentication typical usage

In the Figure 22 it is presented in an UML activity diagram a typical full operation of the system when the system is configured to make local authentication by using the card's memory data; the card's data must be set to a value that so the rooms' devices will recognize it as authenticated; for this purpose the place in which the data is written must match the place the rooms' devices are going to read, as well as the key used to access the memory for creating the card must be the same used for reading it.

In the diagram not all of the steps are mandatory; for instance, if it is desired to keep the card's sector memory key static and well as the card's memory block number, just changing the expected data between costumers, in the diagram, in the section **"set new credentials"** the messages of the COs "[I] Level 1 Auth. Data Address" and "[I] Auth. Access key (level 1)" can be suppress.

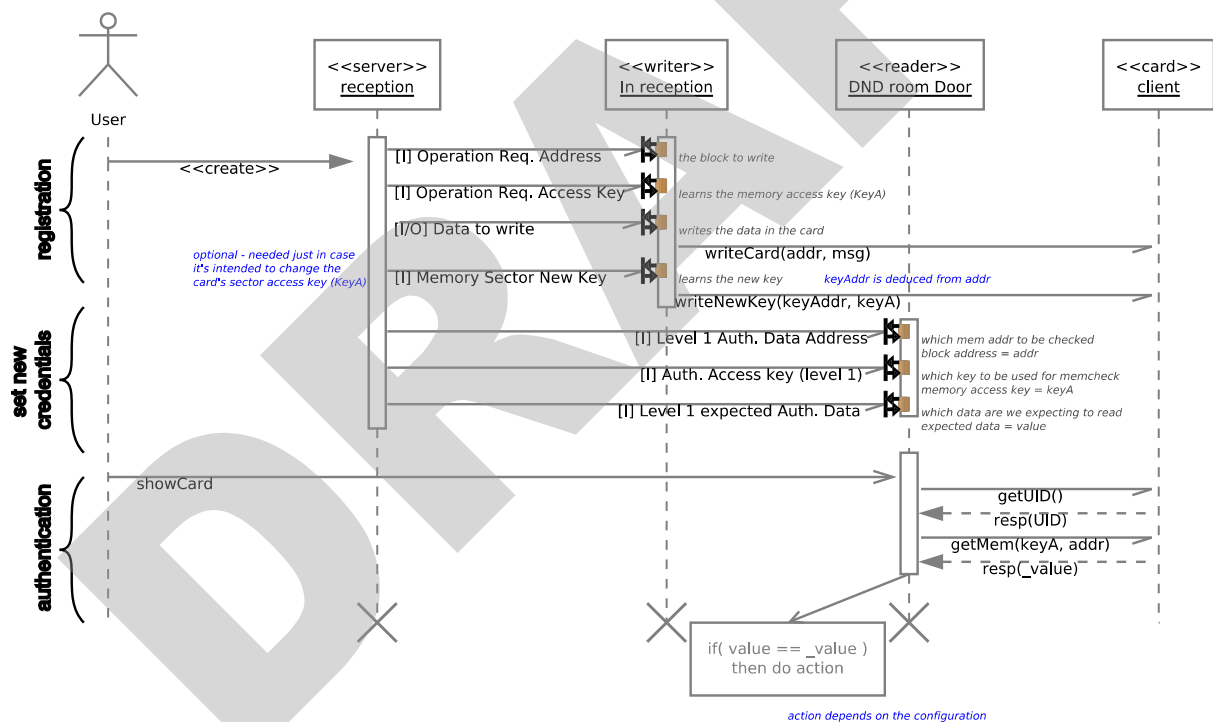


Figure 22: Diagram of a complete usage of Local authentication, using card's memory as authentication data.

From the diagram one may see that for the **"authentication"** section no communication with server is required for declare the card as valid/invalid.

### 5.1.2 Remote authentication typical usage

In the Figure 23 it is presented in an UML activity diagram a typical full operation of the system when the system is configured to make remote authentication by using the card's memory data; the card's data must be set to a value that so the *authentication server*<sup>18</sup> will recognize it as authenticated and inform the rooms' devices about it; for this purpose the place in which the data is written must match the place the rooms' devices are going to read, as well as the key used to access the memory for creating the card must be the same used for reading it.

In the diagram not all of the steps are mandatory; for instance, if it is desired to keep the card's sector memory key static as well as the card's memory block number, just changing the expected data between costumers, in the diagram, the section "**set new credentials**" may be skipped.

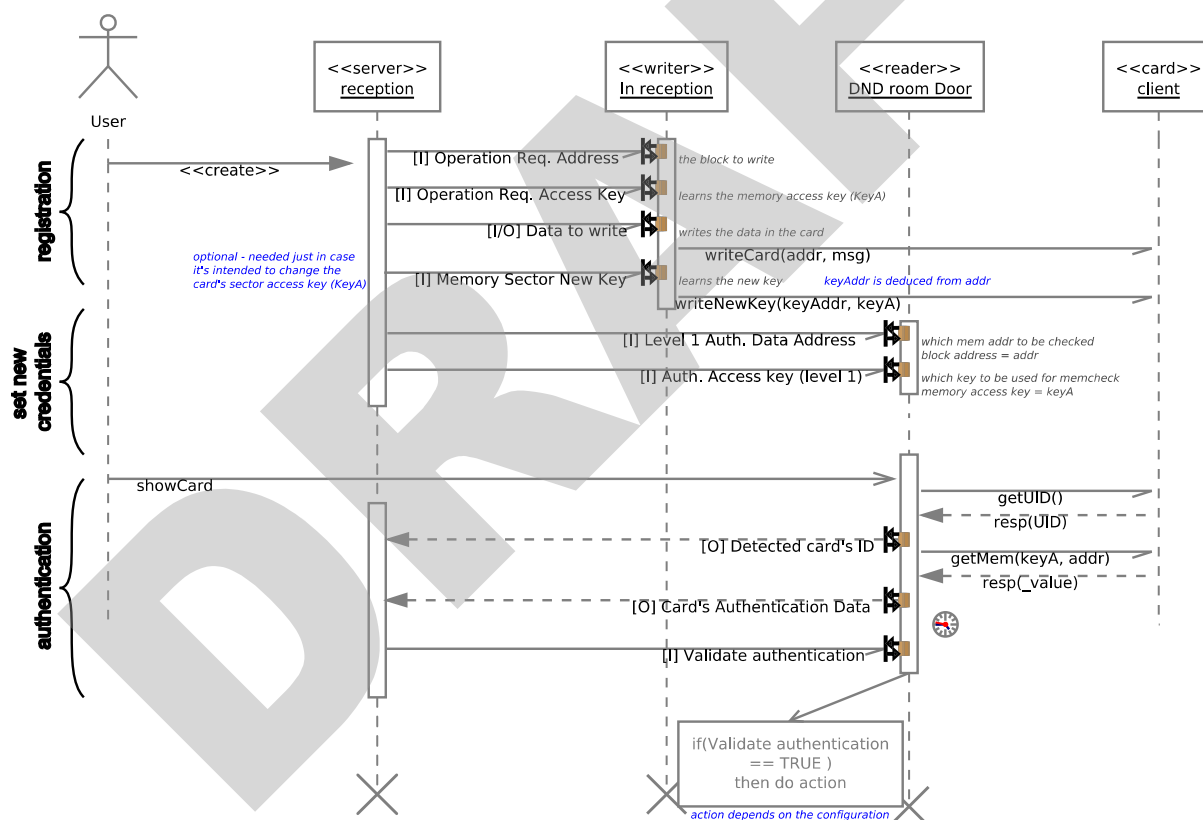


Figure 23 Diagram of a complete usage of Remote authentication, using card's memory as authentication data.

18 The authentication server must have means of reaching the KNX bus and interpret the messages; it must be developed to match the KNX project and must know about the rooms, GAs, and authentication credentials relation.



## 5.2 Using device for creating cards

MDU000x-M and MES000x-M devices are both capable of, via KNX Communication Objects, accessing Mifare Classic cards for write operations. With this, it is possible to use them as a card writer for the project; for example having a device in the lobby/reception intended to create costumer cards for a specific room. The configuration possibilities are vast, therefore here a typical situation is covered.

In order to enable the writing capabilities of the device the parameter “Enable request card operations (read/write) via COs?” from the “DND General” tab, must be set to “Yes”.

As title of example, and for simplifying the explanations let’s here associate the COs to some GAs; only the relevant COs are listed (see Table 16). Notice that the Group Addresses are only for explanation purpose, and in real application they may be what ever the installer wants.

Table 16: Possible GAs for the card writer device

CO #	CO Name	Group Address	Description
87	[I] Operation Req. Address	0/0/1	Setting the card’s memory block to read/write
88	[I] Operation Req. Access Key	0/0/2	Setting the card’s sector key to use in read/write
89	[I/O] Data to write	0/0/3	Data to be written in card’s memory block
90	[I] Memory Sector New Key	0/0/4	Setting new card’s sector key

As title of example, lets also imagine that the Door Unit is configured for local authentication, and that the Memory Access Configurations are as seen in Figure 24. From here it is important to notice that, for the Level 1 authentication (the costumer’s Level), the device is going to look the **card’s block number 4**, and when attempting to read the memory block **it will use the key “HINOX!”**.

The parameter “Data expected for Level 1” can be changed via KNX CO, as we will see in 5.3. For now let’s consider that for the **expected data for Level 1, the Door Unit is expecting to read “name”** from the block 4.

Having in consideration this setup, two possible sequences of KNX GAs communication are going to be presented as a way of creating a card with valid credentials for the Door Unit with configurations as in Figure 24. One of the sequence to execute when creating a card from an unused, brand new card; other sequence for the case that an already existing card (with correct key already set) is to be changed (for example, a card previously used by other costumer).

MEMORY BLOCK SETUP:

Memory block number for Level 1

Memory block number for Level 2

---

DATA EXPECTED:

Memory expected data as ☒ ASCII ☐ Decimal

Data expected for Level 1

When changing L1 expected memory via CO require increment seq? (read manual for info) ☒ No ☐ Yes

Data expected for Level 2

When changing L2 expected memory via CO require increment seq? (read manual for info) ☒ No ☐ Yes

---

ACCESS KEY SETUP:

Memory keys as ☒ ASCII ☐ Decimal

Key Level 1

When changing key via CO require increment seq? (read manual for info) ☒ No ☐ Yes

Key Level 2

Figure 24: Possible room devices' Memory Access Configurations.

### 5.2.1 The sequence for creating the card from brand new card

The card must be placed in the reading field of the writer device.

#### Message

- 1 Send to 0/0/1 message DPT 5.010 with the value 4
- 2 Send to 0/0/2 message DPT 16.\* with the value \$FF \$FF \$FF \$FF \$FF \$FF \$FF \$00 \$00 \$00 \$00 \$00 \$00 \$00 \$00
- 3 Send to 0/0/3 message DPT 16.000 with value "name"
- 4 Send to 0/0/4 message DPT16.000 with value "HINOX!"

#### Explanation

- Set the memory block to be written
- Set the password to be used in the write operations; this is the default password when cards come from factory
- Write "name" in block 4
- Change the card's sector key of the memory block to the one the room's devices are going to use

### 5.2.2 The sequence for creating the card from already used card

The card must be placed in the reading field of the writer device.

Message	Explanation
1 Send to <u>0/0/1</u> message <u>DPT 5.010</u> with the value <u>4</u>	Set the memory block to be written
2 Send to <u>0/0/2</u> message <u>DPT16.000</u> with value " <u>HINOX!</u> "	Set the password to be used in the write operations; assuming the card's sector key was previously set
3 Send to <u>0/0/3</u> message <u>DPT 16.000</u> with value " <u>name</u> "	Send the Level 1 expected data

## 5.3 Example: changing authentication credentials for Local Authentication

In the following of the example in 5.2 it may be desired to eventually change the valid credentials in the Door Unit; for example, after a costumer checkout it may be desirable to disable the costumer's card to authenticate in the room. It would be a good principle to change the "Expected data for Level 1" for every new costumer; this would avoid the risk of an unwanted card duplication resulting in succeeding opening the door.

For this, the devices installed in the rooms can make use of the CO "80 - [I] Level 1 expected Auth. Data".

Table 17: Possible GAs for the room devices (ESaver and Door Unit)

CO #	CO Name	Group Address	Description
80	[I] Level 1 expected Auth. data	1/2/3	Changes the device's expected data in the memory block for an authentic card

### 5.3.1 Making card to not authenticate

In this way in order to, for example disabling the card's authentication after checkout, one can send some data different from the one of the costumer's card (in the previous example "name"):

- send to 1/2/3 message DPT16.000 with value "free"<sup>19</sup>

<sup>19</sup> "free" is only an example, in fact anything different from "name" will result in the costumer's card to be unable to authenticate in the room's devices.

### 5.3.2 Setting the room's devices authentication to the created card's credentials

Still having in mind the previous examples, imagining the room's devices "Expected data for Level 1" is something unknown; we are also changing an already used card (with Level 1 key set to "HINOX!"); also considering that a card for a customer is to be created for authentication in the room; let's also assume that the expected data for Level 1 is to be "John". With the card on the card's writer detection field, a possible messages sequence is:

#### Message

- 1 Send to 0/0/1 message DPT 5.010 with the value 4
- 2 Send to 0/0/2 message DPT16.000 with value "HINOX!"
- 3 Send to 0/0/3 message DPT 16.000 with value "John"
- 4 Send to 1/2/3 message DPT 16.000 with value "John"

#### Explanation

- Set the memory block to be written
- Set the password to be used in the write operations;
- Write "John" in block 4
- Change the room's devices Expected data for Level 1

## 5.4 Encryption

Under some circumstances it may be desirable to prevent foreign partners (let's say *spies*) from having access to some of the more critical information being exchanged on the bus. While KNX secure isn't yet massively available, a security layer can be added on top of the application layer; this is achieved by encrypting the payload of the sent messages.

In the MDU/MES just some of the COs have this option (see Appendix F -I for information of which COs can be encrypted).

In case the *spy* get the chance to intercept the messages, by access the communication bus, in this application there are different aspects to be considered about security:

1. The *spy* **shall not** be able to **read the messages'** contents;
2. The *spy* **shall not** be able to **send manipulated messages** that the system will potentially render as valid;
3. The *spy* **shall not** be able to preform the **replay of a previously intercepted message** rendering him unwanted access.

In order to address the previously mentioned aspects the following techniques may be applied:

1. **Encryption of the transmitted messages:** the *spy* won't be able to decipher the messages unless possessing a valid key;
2. **Cyclic Redundancy Check of the sent messages** (also encrypted): if the *spy* attempts to send a "random" message it will be discarded for it won't contain its CRC together;
3. Usage of **sequence numbers** on the messages: in this way the same message won't be the same twice;

As the implementation of 1. and 2. are more or less straightforward, the implementation of 3. becomes more complex as it requires the communication partners to keep track of each others' communication sequence numbers. For this reason the point 3. was just partially implemented for the most critical CO (see 4.1.4 related with "require increment sequence").

For the encrypting algorithm RC4 is used due to it's simplicity, execution speed while providing a reasonable security level for the kind of application in hands. Many open implementation algorithms can be found on the web.

For the CRC, CRC16 was used.

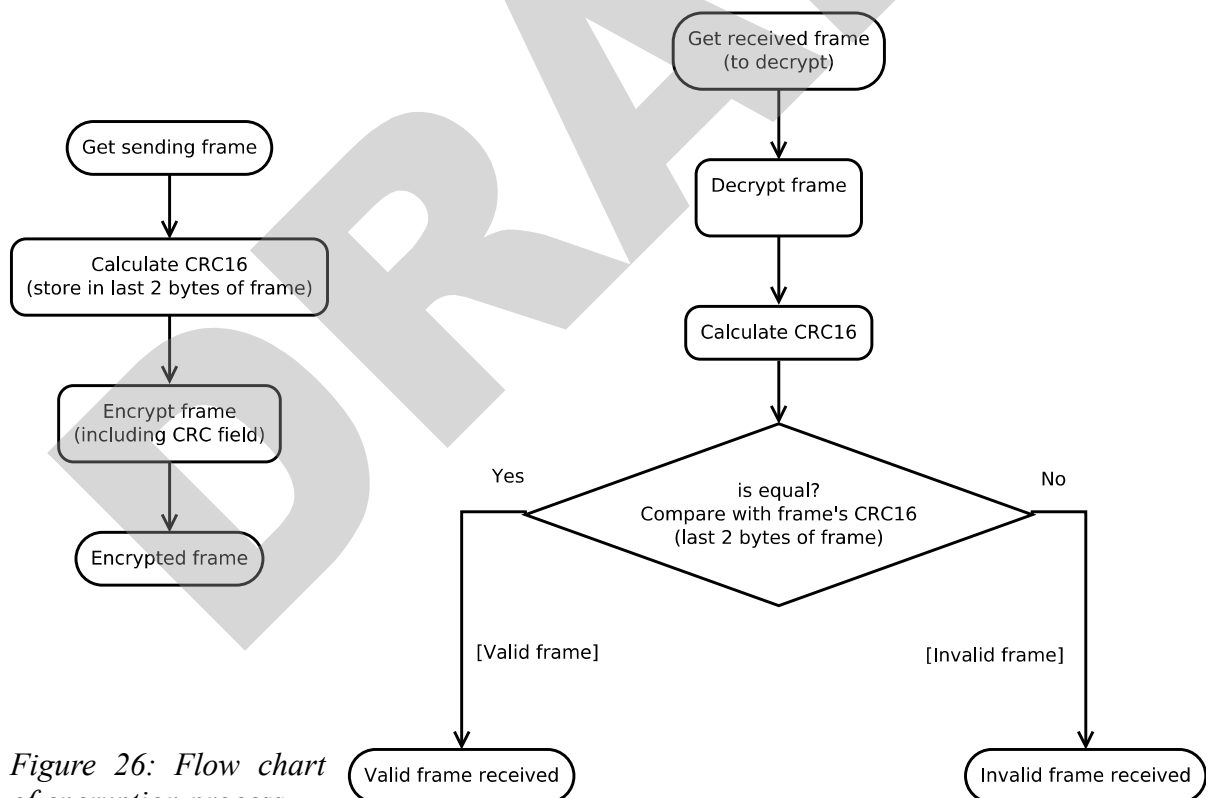
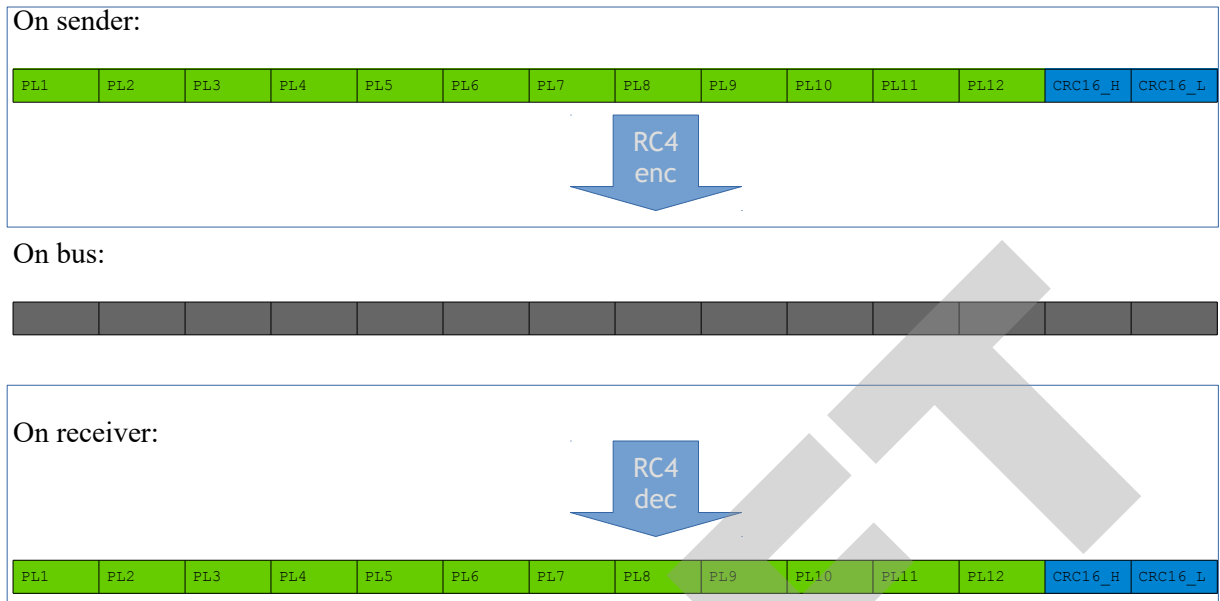


Figure 26: Flow chart of encryption process

Figure 25: Flow chart of decryption

**DRAFT**

## APPENDIXES

---



## APPENDIX A - LOGIC OPERATIONS

Three logic operations are available to be used with up to four binary inputs (see Error: Reference source not found), plus one more (logic NOT) that can be applied to each input independently. In here, useful theoretical information about the four logic operations will be presented.

These functions belongs to the algebra's subarea Boolean algebra, in which the values of the variables are the truth values *TRUE* and *FALSE*, that commonly are denoted by '1' and '0', respectively.

### I - AND (Logical Conjunction)

This operator can be represented by the symbol “ $\wedge$ ”. A  $n$ -place logical operator AND results *TRUE* if  $n$  of its operands are *TRUE*, otherwise the value is *FALSE*.

Main properties:

- Commutativity:  $A \wedge B \Leftrightarrow B \wedge A$  ;
- Associativity:  $A \wedge (B \wedge C) \Leftrightarrow (A \wedge B) \wedge C$  ;
- Distributivity:  $A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$  ;

Table 18: Truth tables for Conjunction Operation

Input			Output
A	B		AB
0	0		0
0	1		0
1	0		0
1	1		1

Input				Output
A	B	C		ABC
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		0
1	0	0		0
1	0	1		0
1	1	0		0
1	1	1		1

Input					Output
A	B	C	D		ABCD
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		0
0	1	0	0		0
0	1	0	1		0
0	1	1	0		0
0	1	1	1		0
1	0	0	0		0
1	0	0	1		0
1	0	1	0		0
1	0	1	1		0
1	1	0	0		0
1	1	0	1		0
1	1	1	0		0
1	1	1	1		1

## II - OR (Logical Disjunction)

This operator can be represented by the symbol “ $\vee$ ”. A  $n$ -place logical operator AND results *TRUE* if at least 1 of  $n$  operands is *TRUE*, if  $n$  operands are *FALSE*, then the result is *FALSE*.

Main properties:

- Commutativity:  $A \vee B \Leftrightarrow B \vee A$  ;
- Associativity:  $A \vee (B \vee C) \Leftrightarrow (A \vee B) \vee C$  ;
- Distributivity:  $A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$  ;

Table 19: Truth tables for Disjunction Operation

Input		Output
A	B	AB
0	0	0
0	1	1
1	0	1
1	1	1

Input			Output
A	B	C	ABC
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Input				Output
A	B	C	D	ABCD
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

### III - XOR (Exclusive disjunction)

This operator can be represented by the symbol “ $\oplus$ ”. A  $n$ -place logical operator XOR results *TRUE* if a odd number of operands is *TRUE*, otherwise then the result is *FALSE*.

Main properties:

- Commutativity:  $A \oplus B \Leftrightarrow B \oplus A$  ;
- Associativity:  $A \oplus (B \oplus C) \Leftrightarrow (A \oplus B) \oplus C$  ;

Table 20: Truth tables for Exclusive Disjunction Operation

Input		Output
A	B	AB
0	0	0
0	1	1
1	0	1
1	1	0

Input			Output
A	B	C	ABC
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Input				Output
A	B	C	D	ABCD
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

## IV - NOT (Negation)

This operator can be represented by the symbol “ $\neg$ ”. Negation is unary (single-argument) logical operator. Negation function takes *Falsity* to *Truth* and vice versa.

Main properties:

- Double negation:  $\neg\neg A \Leftrightarrow A$  and  $\neg\neg\neg A \Leftrightarrow \neg A$  ;
- Distributivity (Morgan's law):  $\neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B)$  and  $\neg(A \wedge B) \Leftrightarrow (\neg A \vee \neg B)$  ;

Table 21: Truth table for NOT Operation

A	$\neg A$
0	1
1	0

## APPENDIX B - KNX DATA TYPES

Table 22: Some of the KNX Data Points Types Table

DPT_ID	DPT_Name	Size (bits)
1.001	DPT_Switch	1
1.002	DPT_Bool	1
1.003	DPT_Enable	1
1.004	DPT_Ramp	1
1.005	DPT_Alarm	1
1.006	DPT_BinaryValue	1
1.007	DPT_Step	1
1.008	DPT_UpDown	1
1.009	DPT_OpenClose	1
1.010	DPT_Start	1
1.011	DPT_State	1
1.012	DPT_Invert	1
1.013	DPT_DimSendStyle	1
1.014	DPT_InputSource	1
1.015	DPT_Reset	1
1.016	DPT_Ack	1
1.017	DPT_Trigger	1
1.018	DPT_Occupancy	1
1.019	DPT_Window_Door	1
1.021	DPT_LogicalFunction	1
1.022	DPT_Scene_AB	1
1.023	DPT_ShutterBlinds_Mode	1
1.100	DPT_eat/Cool	1
2.001	DPT_Switch_Control	2
2.002	DPT_Bool_Control	2
2.003	DPT_Enable_Control	2
2.004	DPT_Ramp_Control	2
2.005	DPT_Alarm_Control	2
2.006	DPT_BinaryValue_Control	2
2.007	DPT_Step_Control	2
2.008	DPT_Direction1_Control	2
2.009	DPT_Direction2_Control	2
2.010	DPT_Start_Control	2
2.011	DPT_State_Control	2
2.012	DPT_Invert_Control	2
3.007	DPT_Control_Dimming	4
3.008	DPT_Control_Blinds	4
4.001	DPT_Char_ASCII	8
4.002	DPT_Char_8859_1	8

5.001	DPT_Scaling	8
5.003	DPT_Angle	8
5.004	DPT_Percent_U8	8
5.005	DPT_DecimalFactor	8
5.010	DPT_Value_1_Ucount	8
6.001	DPT_Percent_V8	8
6.010	DPT_Value_1_Count	8
6.020	DPT_Status_Mode3	8
7.001	DPT_Value_2_Ucount	16
7.002	DPT_TimePeriodMsec	16
7.003	DPT_TimePeriod10MSec	16
7.004	DPT_TimePeriod100MSec	16
7.005	DPT_TimePeriodSec	16
7.006	DPT_TimePeriodMin	16
7.007	DPT_TimePeriodrs	16
7.010	DPT_PropDataType	16
7.011	DPT_Length_mm	16
7.012	DPT_UelCurrentmA	16
7.013	DPT_Brightness	16
8.001	DPT_Value_2_Count	16
8.002	DPT_DeltaTimeMsec	16
8.003	DPT_DeltaTime10MSec	16
8.004	DPT_DeltaTime100MSec	16
8.005	DPT_DeltaTimeSec	16
8.006	DPT_DeltaTimeMin	16
8.007	DPT_DeltaTimers	16
8.010	DPT_Percent_V16	16
8.011	DPT_Rotation_Angle	16
9.001	DPT_Value_Temp	16
9.002	DPT_Value_Tempd	16
9.003	DPT_Value_Tempa	16
9.004	DPT_Value_Lux	16
9.005	DPT_Value_Wsp	16
9.006	DPT_Value_Pres	16
9.007	DPT_Value_umidity	16
9.008	DPT_Value_AirQuality	16
9.010	DPT_Value_Time1	16
9.011	DPT_Value_Time2	16
9.020	DPT_Value_Volt	16
9.021	DPT_Value_Curr	16
9.022	DPT_PowerDensity	16
9.023	DPT_KelvinPerPercent	16
9.024	DPT_Power	16
9.025	DPT_Value_Volume_Flow	16

---

...	...	...
-----	-----	-----

DRAFT

## APPENDIX C - CARD'S MEMORY ACCESS CONDITIONS (NXP VS FUDAN)

Consider the two images below:

Access condition for the Block 3 (X=0-15)

			KEYA	KEYA	Access Con	Access Con	KEYB	KEYB
C1X3	C2X3	C3X3	read	Write	Read	Write	read	Write
0	0	0	never	KEYA B	KEYA B	Never	KEYA B	KEYA B
0	1	0	never	Never	KEYA B	Never	KEYA B	Never
1	0	0	never	KEYB	KEYA B	Never	never	KEYB
1	1	0	never	Never	KEYA B	Never	never	Never
0	0	1	Never	KEYA B	KEYA B	KEYA B	KEYA B	KEYA B
0	1	1	Never	KEYB	KEYA B	KEYB	never	KEYB
1	0	1	Never	Never	KEYA B	KEYB	never	Never
1	1	1	Never	Never	KEYA B	Never	never	Never

Note: KEY A|B means KEY A or KEY B;

Never means can't perform the function.

Figure 27: Fudan Microelectronics FM11RF08 card KeyA and KeyB configurations. Source: Functional Specification

Table 7. Access conditions for the sector trailer

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read <sup>[1]</sup>
0	1	0	never	never	key A	never	key A	never	Key B may be read <sup>[1]</sup>
1	0	0	never	key B	key A B	never	never	key B	
1	1	0	never	never	key A B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration <sup>[1]</sup>
0	1	1	never	key B	key A B	key B	never	key B	
1	0	1	never	never	key A B	key B	never	never	
1	1	1	never	never	key A B	never	never	never	

[1] For this access condition key B is readable and may be used for data

Figure 28: NXP MF1S50yyx/V1 card KeyA and KeyB configurations. Source: Product Datasheet



In the Figure 27 one can see that the card's default access bits configurations allow to Read and Write the KeyA, KeyB and Access bits by using the KeyA or KeyB; in Figure 28 one can see that Read and Write of KeyA, KeyB and Access bits is possible just by using the KeyA, by using KeyB Read and Write operations aren't allowed. This means that in the Fudan's case, if the KeyB is kept in it's default, one may use it for access (reading/writing) the KeyA; as consequence we may potentially get access to the data blocks.

The MDU/MES devices keep the Access bits in it's transport configuration, that so afterwards KeyA, KeyB and data blocks may be altered on the card's memory.

## APPENDIX D - MIFARE MEMORY LAYOUT

[https://commons.wikimedia.org/wiki/File:MiFare\\_Byte\\_Layout.png](https://commons.wikimedia.org/wiki/File:MiFare_Byte_Layout.png)

DRAFT

## APPENDIX E - ENCRYPTION, CRC16: ALGORITHMS & SOURCE CODE

### I - Encryption: RC4 algorithm and source code

“RC4 is an encryption algorithm that was created by Ronald Rivest of RSA Security. It is used in WEP and WPA, which are encryption protocols commonly used on wireless routers. To begin the process of RC4 encryption, you need a key, which is often user-defined and between 40-bits and 256-bits. A 40-bit key represents a five character ASCII code that gets translated into its 40 character binary equivalent (for example, the ASCII key "pwd12" is equivalent to 0111000001110111011001000011000100110010 in binary)”<sup>20</sup>.

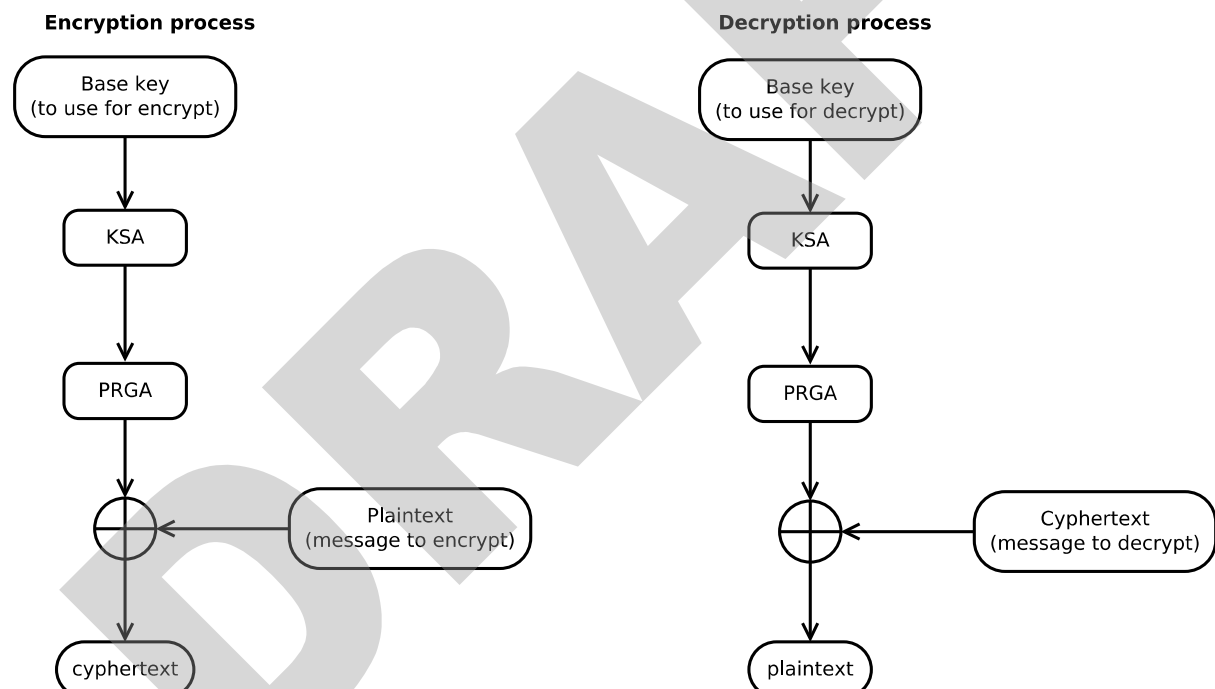


Figure 29: RC4 encryption (left) and decryption (right) flowchart

The processes Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA) processes' implementation are also demonstrated in the figure bellow.

<sup>20</sup> Reference: [https://sites.math.washington.edu/~nichifor/310\\_2008\\_Spring/Pres\\_RC4%20Encryption.pdf](https://sites.math.washington.edu/~nichifor/310_2008_Spring/Pres_RC4%20Encryption.pdf)

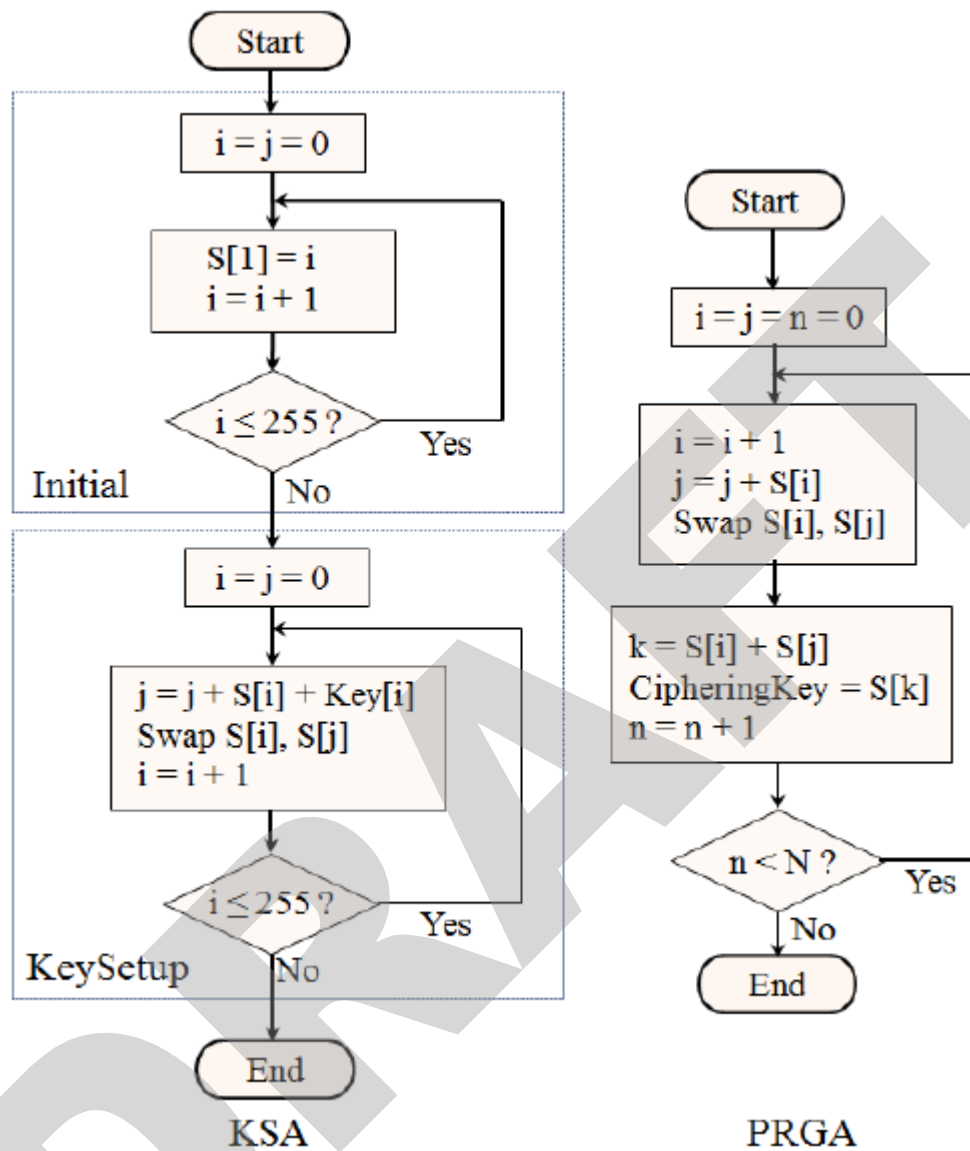


Figure 30: KSA and PRGA processes' flowchart. Ref: [https://www.researchgate.net/figure/261455297\\_fig1\\_Figure-1-RC4%27s-processing-flowchart](https://www.researchgate.net/figure/261455297_fig1_Figure-1-RC4%27s-processing-flowchart)

Bellow it is found the C source code used for implementing RC4 encryption/decryption in the MDU/MES devices:

*rc4.c:*

```
#include <stdint.h>
#include <stdlib.h>

/**
 * Encrypt/Decrypt message
 *
 * \param [in]      key      Base Key
 * \param [in/out]  msg      Plaintext / Cyphertext (after calling contains the output)
 * \param [in]      keyLen   Length of the \ref key in bytes
 * \param [in]      msgLen   Length of the \ref msg in bytes
 * \return         0 if failed, 1 if success
 */
uint8_t rc4_encdec( uint8_t *key, uint8_t *msg, uint8_t keyLen, uint8_t msgLen )
{
    uint8_t bRet;
    uint8_t *pState;

    pState = malloc(256);
    if(NULL == pState)
    {
        /* failed allocating memory for state vector */
        bRet = 0;
    }
    else
    {
        rc4_ksa( pState, key, keyLen );
        rc4_prnga( pState, msg, msgLen );
        free(pState);
        bRet = 1;
    }

    return bRet;
}

/**
 * Key Scheduling Algorithm
 *
 * \param [in/out]  state    Used to generate the keystream
 * \param [in]      key      Key used to initialize the state
 * \param [in]      len      Length of the \ref key in bytes
 */
void rc4_ksa(uint8_t state[], uint8_t key[], uint8_t len)
{
    uint16_t i;
    uint16_t j=0;
    uint16_t t;

    for (i=0; i < 256; ++i)
    {
        state[i] = i;
    }
    for (i=0; i < 256; ++i)
    {
        j = (j + state[i] + key[i % len]) & 0xFF;
        t = state[i];
        state[i] = state[j];
        state[j] = t;
    }
}

/**
 * Pseudo-Random Generator Algorithm
```

---

```

*
* \param [in]      state    Used to generate the keystream
* \param [in,out]  msg      The message to process
* \param [in]      len      Length of the \ref msg in bytes
*/
void rc4_prnga(uint8_t state[], uint8_t msg[], uint8_t len)
{
    uint16_t i=0;
    uint16_t j=0;
    uint16_t x;
    uint16_t t;

    for (x=0; x < len; ++x)
    {
        i = (i + 1) & 0xFF;
        j = (j + state[i]) & 0xFF;
        t = state[i];
        state[i] = state[j];
        state[j] = t;
        msg[x] ^= state[(state[i] + state[j]) & 0xFF];
    }
}

```

---

*end of rc4.c*

A python implementation can be found bellow.

*rc4.py:*

---

```

def rc4(data, key):
    """RC4 encryption and decryption method.
    :param list data: plaintext / cyphertext to encrypt / decrypt
    :param list key: Base key to use for encryption / decryption
    :return list
    """
    # KSA:
    S, j, out = range(256), 0, []
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]
    # PRGA
    i = j = 0
    for ch in data:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        out.append( ch ^ S[(S[i] + S[j]) % 256])
    return out

```

---

*end of rc4.py.*

---

## II -Cyclic Redundancy Check: CRC16 algorithm and source code

Bellow find the python implementation of the CRC16 algorithm. The CRC16 seed used is 0.

*crc16.py:*

```
def crc16(data, dlen, seed=0):  
    """  
    :param list data:  
    :param int dlen:  
    :param int seed:  
    :return int:  
    """  
    reg=seed  
    for i in range(dlen):  
        crc_new = ((reg>>8)&0xFF)|(reg<<8)  
        crc_new = crc_new ^ (data[i])  
        crc_new = crc_new ^ ((crc_new&0xFF)>>4)  
        crc_new = crc_new ^ (crc_new<<12)  
        crc_new = crc_new ^ ((crc_new &0xFF)<<5)  
        reg = crc_new&0xFFFF  
    return reg
```

*end of crc16.py.*

## APPENDIX F - DETAILED DESCRIPTION OF COMMUNICATION OBJECTS

### I - DND

CO n	Text	Function	In/O ut	DPT	Details	Encryption available
57	Relay	[O] Status Indication	Out	1.001 DPT_Switch	Informes about the state of the relay	
58	Authenticati on	[O] Detected card's ID	Out	16.000 DPT_String_ASCII	Sends the read UID	Y
59	Authenticati on	[O] Undetected card's ID	Out	16.000 DPT_String_ASCII	Sends the UID of card undetected	Y
60	Authenticati on	[O] Card's Authentication Data	Out	16.000 DPT_String_ASCII	Sends the read info from the desired Memory address (just the 14 first bytes) MemCheck	Y
61	DND	[O] 1 bit CO (level 1)	Out	1.002 DPT_Bool	COs to send at detection/undetected of Level 1 cards	
62	DND	[O] 2 bit CO (level 1)	Out	2.x		
63	DND	[O] 4 bit CO (level 1)	Out	3.x		
64	DND	[O] 1 byte CO (level 1)	Out	4.x, 5.x, 6.x		
65	DND	[O] Scene CO (level 1)	Out	18.001 DPT_SceneControl		
66	DND	[O] 2 byte CO (level 1)	Out	7.x, 8.x, 9.x	COs to send at detection/undetected of Level 2 cards	
67	DND	[O] 1 bit CO (level 2)	Out	1.002 DPT_Bool		
68	DND	[O] 2 bit CO (level 2)	Out	2.x		
69	DND	[O] 4 bit CO (level 2)	Out	3.x		
70	DND	[O] 1 byte CO (level 2)	Out	4.x, 5.x, 6.x		
71	DND	[O] Scene CO (level 2)	Out	18.001 DPT_SceneControl	COs to send at detection/undetected of Master level cards	
72	DND	[O] 2 byte CO (level 2)	Out	7.x, 8.x, 9.x		
73	DND	[O] 1 bit CO (master)	Out	1.002 DPT_Bool		
74	DND	[O] 2 bit CO (master)	Out	2.x		
75	DND	[O] 4 bit CO (master)	Out	3.x		
76	DND	[O] 1 byte CO (master)	Out	4.x, 5.x, 6.x	COs to send at detection/undetected of Master level cards	
77	DND	[O] Scene CO (master)	Out	18.001 DPT_SceneControl		
78	DND	[O] 2 byte CO (master)	Out	7.x, 8.x, 9.x		
79	Op. Req.	[I/O] Data read (from Rd. Req.)	In/O ut	16.000 DPT_String_ASCII	Sends the Data Read via triggerRead	Y
80	Authenticati on	[I] Level 1 expected Auth. Data	In	16.000 DPT_String_ASCII	Sets Level 1 Auth expected Data	Y
81	Authenticati on	[I] Level 2 expected Auth. Data	In	16.000 DPT_String_ASCII	Sets Level 2 Auth expected Data	Y



82	Authentication	[I] Level 1 Auth. Data Address	In	5.010 DPT_Value_1_Ucount	Level 1 Sets the Addr to be read for Expected Data	
83	Authentication	[I] Level 2 Auth. Data Address	In	5.010 DPT_Value_1_Ucount	Level 2 Sets the Addr to be read for Expected Data	
84	Authentication	[I] Auth. Access key (level 1)	In	16.000 DPT_String_ASCII	Sets the KeyA to be used for memory access for expected data Level 1	Y
85	Op. Req.	[I] Trigger Read Request	In	1.017 DPT_Trigger	Trigger a Read	
86	Op. Req.	[I] Trigger Write Request	In	1.017 DPT_Trigger	Trigger a Write	
87	Op. Req.	[I] Operation Req. Address	In	5.010 DPT_Value_1_Ucount	Sets the Addr to be read at Read trigger	
88	Op. Req.	[I] Operation Req. Access Key	In	16.000 DPT_String_ASCII	Sets the keyA to use at Read/Write Trigger	Y
89	Op. Req.	[I/O] Data to write	In	16.000 DPT_String_ASCII	Sets the data that is to be written (if	Y
90	Op. Req.	[I] Memory Sector New Key	In	16.000 DPT_String_ASCII	Sets a new keyA to the block pointed by memWriteAddr	Y
91	Relay	[I] On/Off	In	1.001 DPT_Switch	Controls the relay	
92	Relay	[I] Prio On/Off	In	2.001 DPT_Switch_Control	Overrides the state of the relay	
93	DND	[I] Room Number	In	7.001 DPT_Value_2_Ucount	Sets the room number (affects the display)	
94	DND	[I] Ring Bell	In	1.017 DPT_Trigger	Triggers the bell ring	
95	DND	[I] Bell melody (prev/next)	In	1.008 DPT_UpDown	Selects previous/next melody	
96	DND	[I] Bell melody number	In	5.010 DPT_Value_1_Ucount	Selects the melody	
97	DND	[I] Bell mute	In	1.001 DPT_Switch	Mutes/Unmutes the bell	
98	Authentication	[I] Validate authentication	In	1.002 DPT_Bool	Informs if the presented auth info (rs_memData + snd_UIDdetected) is valid (NOTE: must be sent within timeOut)	
99	DND	[I] Show text on display	In	16.000 DPT_String_ASCII	Text data to show on the LCD screen (if available)	Y

## II -Logic Channels

There are four logic channels, all of them with same functionality. For determining the Group Object number apply the following offsets:

- Logic Channel A: **X = 0; Y = 0**
- Logic Channel B: **X = 5; Y = 9**
- Logic Channel C: **X = 10; Y = 18**
- Logic Channel D: **X = 15; Y = 27**

Table 23: Logic Channels function specific COs' description

Function			#GO <sup>21</sup>	GO Name	IN/OUT	DPT	
Logic Operation			128+X	[I] Jamming	IN	1.002	
			120+X	[I] DPT1.x Input 1	IN	1.x	
			121+X	[I] DPT1.x Input 2	IN	1.x	
			122+X	[I] DPT1.x Input 3	IN	1.x	
			123+X	[I] DPT1.x Input 4	IN	1.x	
	Send Result		100+Y	[O] DPT1.x	OUT	1.001	
	Value	DPT1	100+Y	[O] DPT1.x	OUT	1.001	
		DPT2	101+Y	[O] DPT2.x	OUT	2.x	
		DPT3	102+Y	[O] DPT3.x	OUT	3.x	
		1 Byte	DPT4	103+Y	[O] DPT4.x	OUT	4.x
			DPT5	103+Y	[O] DPT5.x	OUT	5.x
			DPT6	103+Y	[O] DPT6.x	OUT	6.x
			Scene	103+Y	[O] Scene	OUT	17.001
		2 Byte	DPT7	104+Y	[O] DPT7.x	OUT	7.x
			DPT8	104+Y	[O] DPT8.x	OUT	8.x
			DPT9	104+Y	[O] DPT9.x	OUT	9.x
Copy and Forward/ Mathematical operation/ Comparison			120+X	[I] DPT1.x Input	IN	1.xxx	
			124+X	[I] DPT2.x Input	IN	2.xxx	
			125+X	[I] DPT3.x Input	IN	3.xxx	
			126+X	[I] DPT4/5/6.x Input	IN	4.xxx to 6.xxx	
			127+X	[I] DPT7/8/9.x Input	IN	7.xxx to 9.xxx	
			100+Y	[O] DPT1.x	OUT	1.xxx	
			101+Y	[O] DPT2.x	OUT	2.xxx	
			102+Y	[O] DPT3.x	OUT	3.xxx	
			103+Y	[O] DPT4/5/6.x	OUT	4.xxx to 6.xxx	
			104+Y	[O] DPT7/8/9.x	OUT	7.xxx to 9.xxx	
			128+X	[I] Jamming	IN	1.002	

<sup>21</sup> First CO is the number 0, according to ETS™.