# DOCUMENTATION

## LCN-binding for openHAB



ISSENDORFF

*Author: Patrik Pastuschek*
*as of 14.04.2015*

## Table Of Contents

# 1 Introduction

The LCN-binding for openHAB was created to enable users to combine supreme LCN technology with the versatility of openHAB. This documentation was designed to guide the user through the installation process of the LCN-binding only, for information about the openHAB installation, please refer to the official openHAB homepage and the openHAB wiki page.
Once you correctly installed the openHAB runtime, the installation of the LCN-binding can begin. Follow the given instructions carefully and make sure that you understand everything before continuing.
Please note that the LCN-binding requires Java 7 (JRE 1.7) or later in order to run properly. Please make sure you have installed the correct version before using the LCN-binding, since the binding will not be able to connect otherwise. On top of that, it is also necessary to install the latest LCN PCHK software (version 2.8 or later).

**The LCN openHAB binding requires PCHK version 2.8 or later!**

If you fail to install the correct version of Java or the PCHK, the binding will not be able to connect to your LCN bus system!

# 2 Setting up the LCN-binding

The following information will help you to set up your LCN-binding according to your needs. In general there are three definition layers that can be used to define items and their bindings within openHAB. The general configuration file contains settings that apply to the LCN-binding as a whole, while the item and sitemap definitions are used to refine the binding for singular items.
Though, before you can start with the configuration, it is necessary to install the binding correctly. In general, it should be sufficient to copy the LCN-binding into the 'addons' folder of your openHAB runtime installation. If you have trouble or any questions about this step, you can find additional information in the openHAB binding installation guide.

## *2.1 General configurations*

Before defining any LCN-bindings, it is necessary to adjust the 'openhab.cfg' located within the 'configurations' folder of the runtime distribution of openHAB. Inside the 'openhab.cfg' you should be able to find a passage for the configuration of the LCN-binding. If no such passage exists, you can still add the configurations by hand. The only mandatory configuration settings that are needed to successfully run the LCN-binding with openHAB are:

```
lcn:id1=local
lcn:address1=127.0.0.1:4114
lcn:username1=max
lcn:password1=maximum
lcn:mode=default
```

Any number of id-address-username-password-combinations is allowed, remember to add incrementing numbers to the identifiers (i.e. id1, id2, … address1, address2, …). Do not skip any numbers while incrementing and always start with 1, otherwise the credentials will not be read correctly. On top of that, all IDs need to be unique.
In the example we would bind the address '127.0.0.1:4114', the username 'max' and the password 'maximum' to the ID 'local'.
If you do not define the port of the address (indicated by the colon followed with the actual port number), the default port 4114 will be assigned.
The 'mode' defines the mode of the LCN-bus, you can choose either 'default' or 'advanced'. Though, the 'advanced' mode is only available for systems which consist **entirely** of modules with a firmware of 160B13 (Dec. 2012) or later.
The difference between the two modes is the resolution of dim values, while the 'default' mode can only handle 0-50 different states, the 'advanced' mode provides 0-200 states. The choice of the mode has no further influence on the usage of the binding.
If you don't know, which mode is currently used by the LCN-bus and want to keep the current mode, you can either set the value to 'unknown' or skip the line entirely.

There is also an optional configuration, which handles the ping behaviour. If the PCK does not receive any orders for a certain time (which can be set inside the PCK software), it will close the connection. While the LCN-binding will automatically reconnect, the reconnection process will cause a brief moment of delay since credentials need to be sent again. To prevent these delays a ping will be sent every x seconds, where x is defined by the following configuration.

```
lcn:ping=600
```

This example will set the ping timer to 600 (10 minutes), thus a ping will be sent to every connected PCK every 10 minutes. If you do not declare this value it will be set to its default value of 600 (10 minutes), which is more than enough for the default configuration of the PCK, which will disconnect after 30 minutes.
If you choose not to send any pings to the PCKs, please assign the value 0.

## *2.2 Item definitions*

Item definitions are configured in '*.items' files inside the '…/configurations/items/' folder. A normal item binding within openHAB for LCN will usually look somewhat like the following:

```
Switch Light_Test "Light" {lcn="[ON:local:ON.0.23.2], [OFF:local:'MAP(test.map)']"}
```

Some of the syntax should be well known from the standard openHAB item definition-syntax. But we will explain every item shortly once again.

```
Switch Light_Test "Light" {lcn="[ON:local:ON.0.23.2], [OFF:lcoal:'MAP(test.map)']"}
```

Defines the item for openHAB. 'Switch' is the item type, while 'Light_Test' is its arbitrary identifier.

```
Switch Light_Test "Light" {lcn="[ON:local:ON.0.23.2], [OFF:local:'MAP(test.map)']"}
```

Applies a label to the item, which will be visible in the actual UI (user interface).

```
Switch Light_Test "Light" {lcn="[ON:local:ON.0.23.2], [OFF:local:'MAP(test.map)']"}
```

This is the actual LCN-binding-definition. A LCN-binding-definition always begins with 'lcn=' and is followed by one or several LCN-bindings. The binding must follow a strict syntax in order to be correctly read by the program.

Syntax: [<openHAB_cmd>:]<id>:<lcn_cmd>

A LCN-binding can contain a openHAB command. This can be used to bind certain LCN-bindings to defined openHAB actions. In the example above one binding is bound to the 'ON' command, while the other one is bound to the 'OFF' command. Since the item is a switch, the 'ON'-binding will be executed when the switch is turned on and vice versa. Remember though, that this openHAB command is optional and is not necessary in order to create a LCN-binding. Especially if you want to receive data for a String item, there is often no reason to bind to an openHAB command.

The ID is required in order to send the LCN-commands to the correct system. IDs are defined in the general configuration, where an IP, port, username and password are mapped to that ID.

The LCN-command can be an actual LCN-command or a mapping via given map. In the example we use the map 'test.map' which has to be located within the

'...\configurations\transform' folder. You can only use the map if you also provide a openHAB-command, because the openHAB-command is used as the key for the map.
The map referred to in the example could look like this:

```
undefined=unknown
ON=ON.0.23.2
OFF=OFF.0.23.2
INCREASE=ADD.0.23.2.4
DECREASE=SUB.0.23.2.4
```

Thus, if the second LCN-binding is called, the actual command will be 'OFF.0.23.2'.
On top of that, LCN-commands can be highly flexible in its use with the placeholder '%i'. Using the placeholder will allow you to change commands at runtime, without the need to define several LCN-bindings or even several items.

```
String Flicker_Test "Flicker" {lcn="[local:FLICKER.0.23.2.%i.%i.5]"}
```

How to define the variables which will replace the '%i' is part of the sitemap definitions in the following chapter. Please note that items, that are supposed to use the placeholder '%i', should almost always be defined as 'String' items. The reason is, that the placeholder is replaced by the openHAB command (or some part of it), which is send to the LCN-binding. Most items can only receive a handful of different openHAB commands, while String items can receive virtually anything, giving you a maximum amount of freedom.
The actual way how to display an item is defined in the sitemap, which is why you should not worry too much about this point.

## *2.3 Sitemap definitions*

In general, the sitemap definitions are straight forward and identical to default openHAB sitemap definitions. Please refer to the openHAB-wiki if you are unfamilar with sitemaps. Though it is possible to define the before mentioned placeholder variables (which is actually done by declaring fitting mappings for your items).
In order to exchange the placeholder with actual values at runtime, a special sitemap definition is required.



```
Switch item=Flicker_Test mappings=[LOW_SLOW="LOW and SLOW", MEDIUM_MEDIUM="MEDIUM and MEDIUM",
HIGH_FAST="HIGH and FAST"]
```

This sitemap definition, wraps our String item in a Switch item. String items can receive any kind of textual command as an openHAB command, while Switch items can only receive the commands 'ON' and 'OFF'. By wrapping the String item in a Switch item, we create a Switch, which can receive any kind of String-command.
Consequently, we are able to freely define new mappings as shown above. Every mapping consists of two elements. An openHAB-command name and a label. The label will be used by the UI to label the different buttons, while the openHAB-command will be send to our LCN-binding and will be used to replace our placeholder.
To increase the amount of flexibiltity, the openHAB-command can contain underscores '_' to devide several openHAB-commands. These commands will later be split up and be placed in our LCN-binding.

A push on the button "LOW and SLOW" would lead to the following LCN-command:
FLICKER.0.23.2.LOW.SLOW.5
On the other hand, pressing "HIGH and FAST" would lead to:
FLICKER.0.23.2.HIGH.FAST.5

Beware, that this technique can have influence on the automatic updates of the item, since the binding may be unable to decide, whether the item is connected to an update (i.e. variable segments, etc.).

# 3 Quick reference

This quick reference can be used to look up the syntax for a certain command. While the syntax is available, there is no explanation for the possible values and their meanings. If you need more information, please refer to chapter 4, “LCN Commands”, where the syntax and use for all commands is explained in depth.

## *3.1 LCN-Commands*

The following table shows all available LCN-Commands. In the left column ('Name') you can find the name of the LCN-Command, while the right column ('Syntax') supplies you with the exact syntax for the complete command. As you will notice, this syntax consists of different identifiers (such as 'segment'). To forge your command you need to replace all identifiers with the necessary values and term, which are listed in the table of '3.2 LCN-Identifiers'.

Note: After each command you can add a target modifier. This modifier can be either 'GROUP' or 'MODULE' (without quotes of course) and is optional. Without any modifier the binding will always assume 'MODULE' was meant.
With the 'GROUP' modifier you can target groups that were defined with the 'GROUPS' command, rather than singular LCN modules.
Example with the 'ON' command:

```
ON.15.35.2
```

Will switch the output 2, of module 35, in segment 15, to on.
Please note, that this command could also be written as 'ON.MODULE.15.35.2', without changing the effect at all.

```
ON.GROUP.15.35.2
```

Will switch the output 2, of any module of group 35, in segment 15, to on.



| Available Commands: | |
|---|---|
| **Name:** | **Syntax:** |
| **ON** | ON.segment.module.output[.ramp] |
| **OFF** | OFF.segment.module.output[.ramp] |
| **DIM** | DIM.segment.module.output.percent.ramp |
| **TOGGLE** | TOGGLE.segment.module.output[.ramp] |
| **ADD** | ADD.segment.module.output.percent |
| **SUB** | SUB.segment.module.output.percent |
| **FLICKER** | FLICKER.segment.module.output.pitch.speed.amount(1...15) |
| **TIMED** | TIMED.segment.module.output.timeAmount(6...240).timeunit.speed |
| **PTIMED** | TIMED.segment.module.output.timeAmount(6...240).timeunit.speed |
| **BINARY_STATE** | BINARY_STATE.segment.module.binarySensor(1...8) |

| | |
|---|---|
| **RELAY** | RELAY.segment.module.binary |
| **RELAY_STATE** | RELAY_STATE.segment.module.relay(1...8) |
| **VAR_VALUE** | VAR_VALUE.segment.module.datatype[.dataID(1...12)][.modifier] |
| **SETPOINT_VALUE** | SETPOINT_VALUE.segment.module.regulator.setpointAction[.operator] [.setpointValue(0...2999)][.modifier] |
| **DIM_VALUE** | DIM_percent.segment.module.output |
| **LIMIT** | LIMIT.segment.module.output.percent.value.timeunit |
| **MEMORY** | MEMORY.segment.module.output.ramp |
| **RAMP_STOP** | RAMP_STOP.segment.module.output |
| **DIM_ALL** | DIM_ALL.segment.module.percent.percent[.percent.percent.ramp] |
| **ALL_BRIGHTNESS** | ALL_BRIGHTNESS.segment.module.percent |
| **ALL_ON** | ALL_ON.segment.module.ramp |
| **ALL_OFF** | ALL_OFF.segment.module.ramp |
| **ALL_TOGGLE** | ALL_TOGGLE.segment.module.ramp |
| **QUICKTIMER** | QUICKTIMER.segment.module.output.percent[.ramp] |
| **SHUTTER** | SHUTTER.segment.module |
| **DALI** | DALI.segment.module.daliTarget[.percent].daliCommand[.percent] |
| **DALI_RAW** | DALI_RAW.segment.module.byteValue(0...255).byteValue(0...255) |
| **LIGHT_SCENE** | LIGHT_SCENE.segment.module.sceneAction.channel(0...7) .( scene[.ramp] \| binaryCall ) |
| **CHOOSE_REGISTER** | CHOOSE_REGISTER.segment.module.register |
| **WRITE_SCENE** | WRITE_SCENE.segment.module.scene.register.percent1.ramp1 .percent2.ramp2[.percent3.ramp3[.percent4.ramp4]] |
| **READ_SCENE** | READ_SCENE.segment.module.scene.register[.output[.type]] |
| **RELAY_TIMER** | RELAY_TIMER.segment.module.byteValue(0...255).binary |
| **MOTOR** | MOTOR.segment.module.motorNumber(1...7).motorAction [.value \| .reportType] |
| **SEND_KEYS** | SEND_KEYS.segment.module.buttonAction.buttonAction.buttonAction [.buttonAction].binary |
| **DELAY_KEYS** | DELAY_KEYS.segment.module.table.value.advTimeUnit.binary |
| **LOCK_KEYS** | LOCK_KEYS.segment.module.table.advBinary |
| **TIMELOCK_KEYS** | TIMELOCK_KEYS.segment.module.value.advTimeUnit.simpleBinary |
| **LED** | LED.segment.module.ledNumber(1...12).ledAction |
| **LED_STATE** | LED_STATE.segment.module[.ledNumber(1...12)] |
| **VAR_ADD** | VAR_ADD.segment.module.dataID(1...12).bigValue[.modifier] |
| **VAR_SUB** | VAR_SUB.segment.module.dataID(1...12).bigValue[.modifier] |
| **THRESHOLD_VALUE** | THRESHOLD_VALUE.segment.module[.thresholdRegister][.thresholdNumber] [.modifier] |
| **MOVE_THRESHOLD** | MOVE_THRESHOLD.segment.module.origin.thresholdValue.operator .thresholdRegister.thresholdNumber[.modifier] |
| **MOVE_THRESHOLD_OLD** | MOVE_THRESHOLD_OLD.segment.module.origin.thresholdValue.operator .binary[.modifier] |
| **GET_INFO** | GET_INFO.segment.module.infoID |
| **GET_OEM** | GET_OEM.segment.module.oemID(1...4) |

| | |
|---|---|
| **GROUPS** | GROUPS.segment.module.operator.group |
| **BEEP** | BEEP.segment.module.beepType.amount(1...15) |
| **TEXT** | TEXT.segment.module.line(1..4).block(1...5).message |
| **TIMED_TEXT** | TIMED_TEXT.segment.module.line(1...4).ramp |
| **MRS** | MRS.segment.module.mrsCommand.<mrsCommand>Action |
| **LANGUAGE** | LANGUAGE.segmant.module.language |
| **GET_COUPLER** | GET_COUPLER |
| **STATUS** | STATUS.segment.module.statusCommand |
| **SN** | SN.segment.module |
| **FW** | FW.segment.module |

## *3.2 LCN-Identifiers*

The following table defines all identifiers used by the different LCN-commands. Identifiers may consist of more identifiers, which need to be resolved aswell.

The name defines the name, which is used in the 'LCN-Command' table, in the 'Syntax' column. In the 'Range:' column, you will either find possible values, indicated by either numbers or words written in capital letters, or other identifiers written in lower letters and surrounded by angle brackets ('<' and '>').

Values and identifiers, which are divided by a '|' show different options. You may use only one of the values (or identifiers) at any time. For example, the identifier 'pitch' can be replaced by either LOW, MEDIUM, HIGH or OFF. Furthermore, blue text in brackets is used to give some very short explanations.

| **Available Identifiers:** | |
|---|---|
| **Name:** | **Range:** |
| **segment** | 0, 3(=all), 5...127 |
| **module** | 5...254 |
| **output** | 1...4 |
| **percent** | 0...100 |
| **ramp** | 0...250 |
| **pitch** | LOW \| MEDIUM \| HIGH \| OFF |
| **speed** | SLOW \| MEDIUM \| FAST \| QUICK |
| **amount** | 1...15 |
| **timeAmount** | 6...240 |
| **timeunit** | S (seconds) \| M (minutes) |
| **binarySensor** | 1...8 |
| **relay** | 1...8 |
| **binary** | 0 (off) \| - (no change) \| 1 (on) |
| **binaryCall** | 0 (don't call) \| 1 (call) |
| **datatype** | VAR \| SETPOINT \| COUNTER |
| **dataID** | 1...12 |
| **modifier** | CELSIUS \| LUX \| LUX_T \| VOLT \| AMP \| WIND \| MOISTURE \| CO2 \| NONE |

| | |
|---|---|
| **regulator** | REGULATOR1 \| REGULATOR2 |
| **setpointAction** | SET \| PUSH_CURRENT \| PUSH_PROG \| ACTIVATE \| DEACTIVATE |
| **setpointValue** | 0...2999 |
| **daliTarget** | SINGLE \| GROUP \| BROADCAST |
| **daliCommand** | LIGHT_SCENE \| SET \| OFF \| UP \| DOWN \| STEP_UP \| STEP_DOWN \| MAX \| MIN \| DOWN_OFF \| ON_UP |
| **byteValue** | 0...255 |
| **value** | 0...999 |
| **sceneAction** | LOAD \| SAVE |
| **scene** | 0...9 |
| **register** | 0...9 |
| **channel** | 0...7 |
| **motorNumber** | 1...7 |
| **motorAction** | CLOSE \| OPEN \| FORCE_OPEN \| STOP \| LIMIT \| GOTO \| GOTO_HIGH \| ADD \| SUB \| LEARN \| REPORT1 \| REPORT2 |
| **buttonAction** | SHORT \| LONG \| RELEASE \| - |
| **advTimeUnit** | S \| M \| H \| D |
| **table** | A \| B \| C \| D |
| **advBinary** | 0 (off) \| 1 (on) \| - (no change) \| U (toggle) |
| **simpleBinary** | 0 (off) \| 1 (on) |
| **ledNumber** | 1...12 |
| **ledAction** | OFF \| ON \| FLASH \| FLICKER |
| **bigValue** | 0...30000 |
| **infoID** | NAME1 \| NAME2 \| COMMENT1 \| COMMENT2 \| COMMENT3 |
| **oemID** | 1...4 |
| **message** | *Up to 12 arbitrary characters* |
| **operator** | + \| - |
| **beepType** | NORMAL \| SPECIAL |
| **statusCommand** | OUTPUTS \| PPORT \| RELAYS \| BINARY \| ALL |
| **line** | 1...4 |
| **block** | 1...5 |
| **mrsCommand** | CALL \| VOLUME \| SOURCE \| RADIO \| EQUALIZER \| IMPRESSION \| STANDBY |
| **callAction** | (START.<band>) \| END |
| **volumeAction** | <mrsOutput>.( ((UP \| DOWN \| ONGOING_UP \| ONGOING_DOWN).<volume>) \| RAMP_STOP \| (SET.<percent>) \| (SET_GROUP.<percent>) \| MUTE ) |
| **sourceAction** | <mrsOutput>.( (SET.<source>) \| PREVIOUS \| NEXT ) |
| **radioAction** | PLAY \| STOP \| PREVIOUS \| NEXT \| (SET.<byteValue>) |
| **equalizerAction** | <mrsOutput>.<band>.<db> |
| **impressionAction** | <mrsOutput>.( LOAD \| SAVE ).<impression> |
| **standbyAction** | (empty) |
| **mrsOutput** | 1...12 \| OPTICAL |
| **impression** | 1...16 |

| | |
|---|---|
| **band** | 1...6 |
| **db** | -15...15 |
| **source** | 1...6 \| OPTICAL \| WEBRADIO |
| **volume** | 0...7 |
| **language** | DE \| EN \| ES \| FR \| RU \| AR \| PL \| TR |
| **group** | 5...254 |
| **origin** | PROG \| CURRENT |
| **thresholdNumber** | 0...4 |
| **thresholdRegister** | 0...4 |
| **thresholdValue** | 0...1000 |
| **type** | RAMP \| VALUE |
| **reportType** | POSITION \| LIMIT \| STEP_OUT \| STEP_IN |

# 4 LCN Commands

In the following chapter all available commands will be explained in detail. Not only how to use the command in general, but also how to define it properly. Therefore, each command will be introduced with information about the usage, which will then be expanded with additional information about exact syntax, examples and some hints on what you can actually achieve with them in openHAB.

| **ON** | The 'ON' command is used to switch a single output of a LCN module on. Thus, it is usually used to switch lights, or similar devices, on. |
|---|---|
| items | Switch, Dimmer |
| syntax | **ON.segment.module.output[.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be switched on, maximum is 4.<br>**ramp:** (optional) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **ON.0.23.2** |
| | `Switch On_Test "Switch" {lcn="[ON:local:ON.0.23.2]"}` |
| | This command will switch the output 2, of module 23, in segment 0, to on. |
| notes | |

| **OFF** | The 'OFF' command is used to switch a single output of a LCN module off. Thus, it is usually used to switch lights, or similar devices, off. |
|---|---|
| items | Switch, Dimmer |
| syntax | **OFF.segment.module.output[.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be switched off, maximum is 4.<br>**ramp:** (optional) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |

| | |
|---|---|
| example | **OFF.15.23.2** |
| | `Switch Off_Test "Switch" {lcn="[OFF:local:OFF.15.23.2]"}` |
| | This command will switch the output 2, of module 23, in segment 0, to off. |
| notes | |

| | |
|---|---|
| **DIM** | The 'DIM' command is used to set a dim value for a single output of a LCN module. It can not only be used to dim lights, but also to dim fans and similar devices. |
| items | Switch |
| syntax | **DIM.segment.module.output.percent[.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be dimmed, maximum is 4.<br>**percent:** is the dim value that the device shall receive, maximum is 100.<br>**ramp:** (optional) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **DIM.12.23.2.75.0** |
| | `Switch Dim_Test "Dimmer" {lcn="[ON:local:DIM.0.23.2.75.0]”}` |
| | This command will instantly dim output 2, of the module 23, in segment 12, to 75%. |
| notes | |



| | |
|---|---|
| **TOGGLE** | The 'TOGGLE' command is used to toggle a single output of a LCN module. |
| items | Switch |
| syntax | **TOGGLE.segment.module.output[.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be toggled, maximum is 4.<br>**ramp:** (optional) is the time that the module should take to achieve given |

| | |
|---|---|
| | value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **TOGGLE.17.89.1** |
| | `Switch Toggle_Test "Toggle" {lcn="[ON:local:TOGGLE.17.89.1]"}` |
| | This command will instantly toggle output 1, of the module 89, in segment 17. |
| notes | |

| | |
|---|---|
| **ADD** | The 'ADD' command is used increase the dim value of a single output of a LCN module. It can only be used to increase the value, but never to decrease the value. |
| items | Switch, Dimmer |
| syntax | **ADD.segment.module.output.percent** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be dimmed, maximum is 4.<br>**percent:** is the dim value that the device shall receive, maximum is 100. |
| example | **ADD.12.35.2.20** |
| | `Dimmer Add_Test "Dimmer" {lcn="[INCREASE:local:ADD.12.35.2.20]"}` |
| | This command will increase the dim value of output 2, of the module 35, in segment 12, by 20%. |
| notes | |



| | |
|---|---|
| **SUB** | The 'SUB' command is used decrease the dim value of a single output of a LCN module. It can only be used to decrease the value, but never to increase the value. |
| items | Switch, Dimmer |
| syntax | **ADD.segment.module.output.percent** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be dimmed, maximum is 4. |

| | |
|---|---|
| | **percent:** is the dim value that the device shall receive, maximum is 100. |
| example | **SUB.12.35.2.15** |
| | `Dimmer Sub_Test "Dimmer" {lcn="[DECREASE:local:SUB.12.35.2.15]"}` |
| | This command will decrease the dim value of output 2, of the module 35, in segment 12, by 15%. |
| notes | |

| **FLICKER** | The 'FLICKER' command is used to repeatedly toggle an output of a LCN module, thereby creating a flickering effect. |
|---|---|
| items | Switch |
| syntax | **FLICKER.segment.module.output.pitch.speed.amount** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be flickered, maximum is 4.<br>**pitch:** is the maximum dim value that shall be used during the flicker. Possible values are: LOW MEDIUM HIGH and OFF.<br>**speed:** is the speed at which the flickering shall be done. Possible values are: SLOW MEDIUM and FAST.<br>**amount:** is the amount of flickers, maximum is 15. |
| example | **FLICKER.45.13.3.HIGH.FAST.7** |
| | `Switch Flicker_Test "Flicker" {lcn="[ON:local:FLICKER.45.13.3.HIGH.FAST.7]"}` |
| | This command will flicker the output 3, of module 13, in segment 45, with a high pitch with fast speed 7 times. |
| notes | This command is almost exclusively used with lights. |

| **TIMED** | The 'TIMED' command is used to switch an output of a LCN module, for a limited amount of time, to on. Once the set amount of time has passed, the output will be switched to off again. |
|---|---|
| items | Switch |
| syntax | **TIMED.segment.module.output.timeAmount.timeunit.speed** |
| | **segment:** is the ID of the segment, where the targeted LCN module is |

| | |
|---|---|
| | located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be switched, maximum is 4.<br>**timeAmount:** is the amount of time, after which the output shall be switched to off again. Values are between 6 and 240.<br>**timeunit:** is the unit of time associated with the timeAmount. Is either S (for seconds) or M (for minutes).<br>**speed:** is the speed at which the dim value shall be reached, when the light is switched on. Possible values are: SLOW MEDIUM and QUICK. |
| example | **TIMED.3.17.2.10.S.FAST** |
| | `Switch Timed_Test "Timed" {lcn="[ON:local:TIMED.3.17.2.10.S.FAST]"}` |
| | This command will quickly switch output 2, of module 17, in segment 3 to on for 10 seconds. |
| notes | |

| | |
|---|---|
| **PTIMED** | The 'PTIMED' command is used to switch an output of a LCN module, for a limited amount of time, to on. Once the set amount of time has passed, the output will be switched to off again. Unlike 'TIMED', 'PTIMED' will extend the duration of other timed commands that are currently in action. |
| items | Switch |
| syntax | **PTIMED.segment.module.output.timeAmount.timeunit.speed** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be switched, maximum is 4.<br>**percent:** is the dim value that the device shall recieve, maximum is 100.<br>**timeAmount:** is the amount of time, after which the output shall be switched to off again. Values are between 6 and 240.<br>**timeunit:** is the unit of time associated with the timeAmount. Is either S (for seconds) or M (for minutes).<br>**speed:** is the speed at which the dim value shall be reached, when the light is switched on. Possible values are: SLOW MEDIUM and QUICK. |
| example | **PTIMED.3.17.2.10.S.FAST** |
| | `Switch PTimed_Test "PTimed" {lcn="[ON:local:PTIMED.3.17.2.10.S.FAST]"}` |
| | This command will quickly switch output 2, of module 17, in segment 3, to on for 10 seconds. If the output is already switched on with by timed command, |

| | |
|---|---|
| | the duration will be increased by 10 seconds. |
| notes | |

| **BINARY_STATE** | The 'BINARY_STATE' command is used to receive the state of a binary sensor of a LCN module. |
|---|---|
| items | String, Contact |
| syntax | **BINARY_STATE.segment.module.binarySensor** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**binarySensor:** is the ID of the binary sensor. Value is between 1 and 8. |
| example | **BINARY_STATE.40.29.4** |
| | `String Binary_Test "Binary [%s] {lcn="[local:BINARY_STATE.40.29.4]"}` |
| | This command will receive the value of the binary sensor number 4, of module 29, in segment 40. And will print the received value ('true' or 'false') at the position of '%s'. |
| notes | For obvious reasons, you can only receive data from a binary sensor and never send data to the sensor. |

| **RELAY** | The 'RELAY' command is used to set the states of one or more relays of a LCN module. |
|---|---|
| items | Switch |
| syntax | **RELAY.segment.module.binary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**binary:** are the binary states of the relays. This argument consists of 8 characters which can be either of the following characters: 1 (switch to on), 0 (switch to off), - (do nothing) or U (toggle). |
| example | **RELAY.0.20.---1----** |
| | `Switch Relay_Test "Relay" {lcn="[ON:local:RELAY.0.20.---1----]` |
| | This command will switch the 4th relay of module 20 in segment 0 to on, while all other relays will keep their current state. |

| | |
|---|---|
| notes | |

| | |
|---|---|
| **RELAY_STATE** | The 'RELAY_STATE' command is used to receive the state of a single relay of a LCN module. |
| items | String |
| syntax | **RELAY_STATE.segment.module.relay** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**relay:** is the ID of the Relais, between 1 and 8. |
| example | **RELAY_STATE.0.20.4** |
| | `String Relay_Sensor "Relay Sensor [%s]" {lcn="[ON:local:RELAY_STATE.0.20.4]` |
| | This command will receive the state of the 4th relay of module 20, in segment 0, and print its current state ('true' or 'false') at the position of '%s'. |
| notes | |

| | |
|---|---|
| **VAR_VALUE** | The 'VAR_VALUE' command is used to receive data from a non binary sensor. |
| items | String, Number |
| syntax | **VAR_VALUE.segment.module.datatype[.dataID][.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**datatype:** is the type of data, that you expect to receive. Possible values are: *VAR, SETPOINT and COUNTER*.<br>VAR: counter- / calculation-variable (dataID: 1 - 12)<br>SETPOINT: setpoint (dataID: 1 - 2)<br>COUNTER: counter variable (dataID: 1 - 4)<br>**[dataID]:** (optional) depending on the datatype. Note that older modules have significantly less variable slots available. Only 1 for VAR, no COUNTER and 2 for the remainder.<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible |

| | |
|---|---|
| | values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |
| example | **VAR_VALUE.0.8.VAR.2.CELSIUS** |
| | `String Temp_Sensor "Temperature Sensor [%s °C]" {lcn="[local:VAR_VALUE.0.8.VAR.2.CELSIUS]"}` |
| | This command will receive transformed temperature values (in °C) from sensor 2, of module 8, in segment 0, and print them at the position of '%s' |
| notes | |

| **SETPOINT_VALUE** | The 'SETPOINT_VALUE' command is used to change the setpoint of a LCN module. |
|---|---|
| items | Switch, Dimmer |
| syntax | **SETPOINT_VALUE.segment.module.regulator.setpointAction[.operator][.setpointValue][.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**regulator:** is the ID of the regulator, either REGULATOR1 or REGULATOR2<br>**setpointAction:** can be either of the following: *SET, PUSH_CURRENT, PUSH_PROG, ACTIVATE, DEACTIVATE*.<br>SET: sets the value of the setpoint to a certain value.<br>PUSH_CURRENT: moves the current value of the setpoint (requires an operator and a setpointValue).<br>PUSH_PROG: moves the programmed value of the setpoint (requires an operator and a setpointValue).<br>ACTIVATE: enables the setpoint for further commands.<br>DEACTIVATE: disables the setpoint for further commands.<br>**operator:** is either + or -<br>**setpointValue:** is the value for SET and PUSH actions, maximum is 2000 for PUSH and 2999 for SET.<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |

| | |
|---|---|
| example | **SETPOINT_VALUE.0.8.REGULATOR1.PUSH_CURRENT.+.1** |
| | `Dimmer Setpoint_Mover {lcn="[INCREASE:local:SETPOINT_VALUE.0.8.REGULATOR1.PUSH_CURRENT.+.1], [DECREASE:local:SETPOINT_VALUE.0.8.REGULATOR1.PUSH_CURRENT.-.1]"}` |
| | This command will increase setpoint 1 by 1 if the up key is pressed, and decrease it by 1 if the down key is pressed. Issuing the command to module 8, in segment 0. |
| notes | |

| **DIM_VALUE** | The 'DIM_VALUE' command is used to receive the value of a dimmed output of a LCN module. |
|---|---|
| items | Number |
| syntax | **DIM_VALUE.segment.module.output** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port, maximum is 4. |
| example | **DIM_VALUE.12.23.2** |
| | `Number Dim_Value "Dim value [%d]" {lcn="[local:DIM_VALUE.12.23.2]"}` |
| | This command will receive the current value of the output 2, of module 23, in segment 12, and print it at the position of '%d'. |
| notes | |



| **LIMIT** | The 'LIMIT' command is used to limit the output of a LCN module by reducing the maximum value that it can receive. |
|---|---|
| items | Switch |
| syntax | **LIMIT.segment.module.output.percent.value.timeunit** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port, maximum is 4.<br>**percent:** the actual limit between 0 and 100 in steps of 4. Illegal values will be automatically adjusted. |

| | |
|---|---|
| | **value:** the amount of time that the limit shall persist. Maximum values depend on the timeunit.<br>**timeunit:** the unit of time, either S (1 - 60 seconds), M (1 – 90 minutes), H (1 – 50 hours) or D (1 – 45 days). |
| example | **LIMIT.0.23.2.25.10.S** |
| | `Switch Limit "Light Limit" {lcn="[ON:local:LIMIT.0.23.2.25.10.S]"}` |
| | This command will limit output 2, of module 23, in segment 0, to a maximum of 24% for 10 seconds. Note how the limit value is automatically adjusted! |
| notes | |

| **MEMORY** | The 'MEMORY' command does either read or load a dim value of an output. The current value will be read (and switched off) if the value is greater than 0, otherwise the last stored value will be loaded. |
|---|---|
| items | Switch |
| syntax | **MEMORY.segment.module.output.ramp** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port, maximum is 4.<br>**ramp:** is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **MEMORY.0.23.2.0** |
| | `Switch Memory_Test "Memory" {lcn="[ON:local:MEMORY.0.23.2.0]"}` |
| | This command will either read or load a dim value for output 2, of module 23, in segment 0. This depends on the current state of output 2. |
| notes | |

| **RAMP_STOP** | Will terminate the ramp of a targeted output of a LCN module. |
|---|---|
| items | Switch |
| syntax | **RAMP_STOP.segment.module.output** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127. |

| | |
|---|---|
| | **module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port, maximum is 4. |
| example | **RAMP_STOP.0.23.2** |
| | `Switch Stop_Test "Ramp Stopper" {lcn="[ON:local:RAMP_STOP.0.23.2]"}` |
| | This command will terminate the current ramp of the output 2, of module 23, in segment 0. |
| notes | |

| | |
|---|---|
| **DIM_ALL** | Dims all outputs of a single LCN module. Output 3 and 4 are only available for firmware 160901 or later. Modules with a firmware between 10061A and 160901 have 4 outputs, but are unable to use the 'DIM_ALL' command for 4 outputs. For that case, please consider using the 'ALL_BRIGHTNESS' command. |
| items | Switch, Dimmer |
| syntax | **DIM_ALL.segment.module.percent.percent2[.percent3.percent4.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**percent:** is the dim value for output 1, maximum is 100.<br>**percent2:** is the dim value for output 2, maximum is 100.<br>**percent3:** (optional) is the dim value for output 3, maximum is 100.<br>**percent4:** (optional) is the dim value for output 4, maximum is 100.<br>**ramp:** (optional) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **DIM_ALL.0.23.100.0.100.0.0** |
| | `Switch Dim_Test "Dim All" {lcn="[ON:local:DIM_ALL.0.23.100.25.100.25.0]"}` |
| | This command will target module 23, in segment 0, and instantly dim the outputs 1 and 3 to 100%, while dimming outputs 2 and 4 to 25%. |
| notes | If you wish to use any of the optional arguments, then you have to use all of them. |

| **ALL_BRIGHTNESS** | The 'ALL_BRIGHTNESS' command is used to dim all outputs of a LCN module at once. |
|---|---|
| items | Switch, Dimmer |
| syntax | **ALL_BRIGHTNESS.segment.module.percent** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**percent:** is the dim value for all outputs, maximum is 100. |
| example | **ALL_BRIGHTNESS.0.23.75** |
| | `Switch On_Test "Switch" {lcn="[ON:local:ALL_BRIGHTNESS.0.23.75]"}` |
| | This command will dim all outputs of module 23, in segment 0, to 75%. |
| notes | The difference to DIM_ALL is, that ALL_BRIGHTNESS can only set one dim value for all outputs, while DIM_ALL can set values for each output independently. |

| **ALL_ON** | The 'ALL_ON' command is used to switch a all outputs of a LCN module on. Thus, it is usually used to switch lights, or similar devices, on. |
|---|---|
| items | Switch, Dimmer |
| syntax | **ALL_ON.segment.module.ramp** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**ramp:** is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **ALL_ON.0.23.0** |
| | `Switch On_Test "Switch" {lcn="[ON:local:ALL_ON.0.23.0]"}` |
| | This command will instantly switch all outputs of module 23, in segment 0, to on. |
| notes | |

| **ALL_OFF** | The 'ALL_OFF' command is used to switch a all outputs of a LCN module off. Thus, it is usually used to switch lights, or similar devices, off. |
|---|---|
| items | Switch, Dimmer |
| syntax | **ALL_OFF.segment.module.ramp** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**ramp:** is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **ALL_OFF.0.23.0** |
| | `Switch Off_Test "Switch" {lcn="[ON:local:ALL_OFF.0.23.0]"}` |
| | This command will instantly switch all outputs of module 23, in segment 0, to off. |
| notes | |

| **ALL_TOGGLE** | The 'ALL_TOGGLE' command is used to toggle all outputs of a LCN module. |
|---|---|
| items | Switch, Dimmer |
| syntax | **ALL_TOGGLE.segment.module.ramp** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**ramp:** is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **ALL_TOGGLE.0.23.0** |
| | `Switch On_Test "Switch" {lcn="[ON:local:ALL_TOGGLE.0.23.0]"}` |
| | This command will instantly toggle all outputs of module 23, in segment 0. |
| notes | |

| **QUICKTIMER** | The 'QUICKTIMER' command immediately sets the value of an output to a given value. Afterwards the value is transitioned to 0% |
|---|---|

| | |
|---|---|
| | with the given ramp. |
| Items | Switch |
| Syntax | **QUICKTIMER.segment.module.output.percent[.ramp]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**output:** is the ID of the output port that is to be dimmed, maximum 4. Older modules may only target 2 outputs.<br>**percent:** is the dim value that the device shall receive, maximum is 100.<br>**ramp:** (optional) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| Beispiel | **QUICKTIMER.120.77.2.50.2** |
| | `Switch Quick "Quicktimer" {lcn="[local:QUICKTIMER.120.77.2.50.2]` |
| | This command will set output 2, of module 77, in segment 120, to 50%. Afterwards the value is changed to 0% with the ramp (see appendix) of 2. |
| Hinweise | |

| **SHUTTER** | The 'SHUTTER' command is used to control shutters. |
|---|---|
| items | Switch |
| syntax | **SHUTTER.segment.module** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254. |
| example | **SHUTTER.120.77** |
| | `String Shutter "Shutter [%s]" {lcn="[local:SHUTTER.120.77]` |
| | This command will control the LCN module 77, in segment 120, as a shutter. |
| notes | |

| **DALI** | The 'DALI' command enables usage of DALI-EVGs. |
|---|---|

| | |
|---|---|
| items | Switch, Dimmer |
| syntax | **DALI.segment.module.daliTarget[.percent].daliCommand[.percent]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**daliTarget:** the target of the command, possible values are: *SINGLE, GROUP, BROADCAST*.<br>SINGLE: just one singular module.<br>GROUP: a DALI group.<br>BROADCAST: all modules.<br>**percent:** (optional) if you use SINGLE or GROUP, you need to specify the target with an address. For SINGLE this can be between 0 and 63 and for GROUP between 0 and 15.<br>**daliCommand:** the actual command, possible values are: *LIGHT_SCENE, SET, OFF, UP, DOWN, STEP_UP, STEP_DOWN, MAX, MIN, DOWN_OFF, ON_UP*.<br>LIGHT_SCENE: calls a certain light scene.<br>SET: sets the brightness.<br>OFF: switches to off.<br>UP: increases brightness.<br>DOWN: decreases brightness.<br>STEP_UP: increases brightness by one step.<br>STEP_DOWN: decreases brightness by one step.<br>MAX: calls the maximum value.<br>MIN: calls the minimum value.<br>DOWN_OFF: decreases the brightness by one step and then switches to off.<br>ON_UP: switches to on and then increases the brightness by one step.<br>**percent:** (optional) if you use LIGHT_SCENE or SET, you need to specify the target (or value). For LIGHT_SCENE the value can be between 0 and 15 and for SET between 0 and 254 (where 254 is 100%). |
| example 1 | **DALI.0.23.BROADCAST.OFF** |
| | `Switch Dali "Dali" {lcn="[ON:local:DALI.0.23.BROADCAST.OFF]"}` |
| | This command will switch all modules at module 23 in segment 0 to off. |
| example 2 | DALI.0.23.SINGLE.45.LIGHT_SCENE.12 |
| | `Switch Dali "Dali" {lcn="[ON:local:DALI.0.23.SINGLE.45.LIGHT_SCENE.12]"}` |
| | This command will select the light scene 12, for DALI-EVG 45, at LCN |

| | |
|---|---|
| | module 23, in segment 0. |
| notes | |

| **DALI_RAW** | The 'DALI_RAW' command is used to send direct commands in form of bytes to DALI-EVGs. |
|---|---|
| items | Switch, Dimmer |
| syntax | **DALI_RAW.segment.module.byteValue.byteValue** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**byteValue:** any byte value (0 – 255). |
| example | **DALI_RAW.0.23.10.20** |
| | `Switch Dali_Raw "Dali Raw" {lcn="[ON:local:DALI_RAW.0.23.10.20]"}` |
| | This command will send the two byte values 10 and 20 to a DALI-EVG at LCN module 23, in segment 0. |
| notes | This command should only be used in combination with a sufficient DALI documentation. |

| **LIGHT_SCENE** | The 'LIGHT_SCENE' command is used to load or save light scenes, via either channels or relays. |
|---|---|
| items | Switch |
| syntax | **LIGHT_SCENE.segment.module.sceneAction.channel**<br>**.( scene[.ramp] \| binaryCall )** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**sceneAction:** is either *LOAD* or *SAVE*.<br>LOAD: loads a light scene.<br>SAVE: saves the current value to a light scene.<br>**channel:** the output for the channel. Values can be between 0 and 7, mapped like the following:<br>0 = use relay information. (requires the binaryCall argument). |

| | |
|---|---|
| | 1 = output 1<br>2 = output 2<br>3 = output 1 and 2<br>4 = output 3<br>5 = output 1 and 3<br>6 = output 2 and 3<br>7 = all outputs<br>**scene:** the scene to load, values between 0 and 9<br>**ramp:** (optional) (not compatible with channel 0) is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255.<br>**binaryCall:** are the binary states of the relays. This argument consists of 8 characters which can be either of the following characters: 1 (call), 0 (don't call). |
| example 1 | **LIGHT_SCENE.0.23.LOAD.2.4.0** |
| | `Switch Load_Scene "Load Light Scene: " {lcn="[ON:local:LIGHT_SCENE.0.23.LOAD.2.4.0]"}` |
| | This command will instantly load light scene number 4, from channel 2, at module 23, in segment 0. |
| example 2 | LIGHT_SCENE.0.23.LOAD.0.2.11110000 |
| | `Switch Load_Scene "Load Light Scene: " {lcn="[ON:local:LIGHT_SCENE.0.23.LOAD.0.2.11110000]"}` |
| | This command will load the light scene number 2 with relays 1 through 4, of module 23 in segment 0. |
| notes | |



| **CHOOSE_ REGISTER** | The 'CHOOSE_REGISTER' command is used to change the register for loading and saving of light scenes. |
|---|---|
| items | Switch |
| syntax | **CHOOSE_REGISTER.segment.module.register** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**register:** is the ID of the register, maximum value is 9. |
| example | **CHOOSE_REGISTER.0.23.0** |

| | |
|---|---|
| | `Switch Register "Choose Register" {lcn="[ON:local:CHOOSE_REGISTER.0.23.0]"}` |
| | This command will select register 0, for module 23, in segment 0. |
| notes | Use this command in combination with LIGHT_SCENE to have an increased capacity for saving and loading light scenes. |

| **WRITE_ SCENE** | The 'WRITE_SCENE' command is used to directly write a light scene. |
|---|---|
| items | Switch |
| syntax | **WRITE_SCENE.segment.module.scene.register.percent1.ramp1.persent2.ramp2[.percent3.ramp3[.percent4.ramp4]]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**scene:** the scene to write, values between 0 and 9<br>**register:** is the ID of the register, maximum value is 9.<br>**percent(i):** is the dim value (in half percent) that the device shall receive, maximum is 200.<br>**ramp(i):** is the time that the module should take to achieve given value. A list about possible values can be found in the appendix, maximum is 255. |
| example | **WRITE_SCENE.0.23.2.1.100.0.50.0** |
| | `Switch Write_Scene "Direct Write Scene" {lcn="[ON:local:WRITE_SCENE.0.23.2.1.100.0.50.0]"}` |
| | This command will write the light scene in register 1, scene 2, of module 23, in segment 0. The light scene will instantly set output 1 to 100%, and output 2 to 50%. |
| notes | |

| **READ_ SCENE** | The 'READ_SCENE' command is used to directly read a light scene. |
|---|---|
| items | String |
| syntax | **READ_SCENE.segment.module.scene.register[.output[.type]]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is |

| | |
|---|---|
| | located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**scene:** the scene to load, values between 0 and 9<br>**register:** is the ID of the register, maximum value is 9.<br>**output:** (optional) is the ID of the output port to read from, maximum is 4.<br>**type:** (optional) either RAMP or VALUE. |
| example | **READ_SCENE.0.23.2.1** |
| | `String Read_Scene "Direct Read Scene: [%s]" {lcn="[local:READ_SCENE.0.23.2.1]"}` |
| | This command will read the light scene in register 1, scene 2, of module 23, in segment 0, and prints it at the position of '%s'. |
| notes | |

| **RELAY_TIMER** | The 'RELAY_TIMER' command will instantly switch a relay to on and, after a set amount of time, to off again. |
|---|---|
| items | Switch |
| syntax | **RELAY_TIMER.segment.module.byteValue.binary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**byteValue:** any byte value (1 – 255). 1 = 0.03s, 255 = 240.96s. Please refer to the appendix for the full table.<br>**binary:** are the binary states of the relays. This argument consists of 8 characters which can be either of the following characters: 1 (switch to on), 0 (switch to off), - (do nothing). |
| example | **RELAY_TIMER.0.23.1.----1---** |
| | `Switch Relay_Timer "Timed Relay" {lcn="[ON:local:RELAY_TIMER.0.23.1.----1---]"}` |
| | This command will target module 23, in segment 0, and instantly switch relay 5 to on. After 0.03 seconds it will switch the relay to off again. |
| notes | |

| **MOTOR** | The 'MOTOR' command controls an engine at a LCN module. |
|---|---|

| | |
|---|---|
| items | Switch, Dimmer |
| syntax | **MOTOR.segment.module.motorNumber.motorAction[.value \| .reportType]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**motorNumber:** the ID of the engine, between 1 and 7. Not required for the motorActions REPORT1 and REPORT2 if you do not use a reportType.<br>**motorAction:** possible values are: *CLOSE, OPEN, FORCE_OPEN, STOP, LIMIT, GOTO, GOTO_HIGH, ADD, SUB, LEARN, REPORT1, REPORT2*<br>CLOSE: closes (i.e. a window).<br>OPEN: opens (i.e. a window).<br>FORCE_OPEN: forces to open (i.e. a window).<br>STOP: stops.<br>LIMIT: sets a maximum value.<br>GOTO: goes to specified position.<br>GOTO_HIGH: goes to specified position, with higher accuracy.<br>ADD: add a certain amount to current position.<br>SUB: subtract a certain amount from current position.<br>LEARN: do a teach run (measures the time needed from open to close).<br>REPORT1: extended report of engine 1 and 2. (this command does not need a motorNumber).<br>REPORT2: extended report of engine 3 and 4. (this command does not need a motorNumber).<br>**value:** (optional) LIMIT, and GOTO need a value between 0 and 100 (in %), while GOTO_HIGH, ADD and SUB require a value between 0 and 200 (in 0.5%).<br>**reportType:** (optional) for REPORT1 and REPORT2 the following types can be used. If no type is specified, all types will be returned in a single message. POSITION, LIMIT, STEP_IN and STEP_OUT.<br>POSITION: the current position of the engine.<br>LIMIT: the limit of the engine.<br>STEP_IN: the step IN of the engine.<br>STEP_OUT: the step OUT of the engine. |
| example | **MOTOR.0.23.1.STOP** |
| | `Switch Motor "MOTOR Stop" {lcn="[ON:local:MOTOR.0.23.1.STOP]"}` |
| | This command will stop the first engine, of module 23, in segment 0. |

| | |
|---|---|
| notes | |

| | |
|---|---|
| **SEND_ BUTTON** | The 'SEND_KEYS' command will send a command to one or more buttons of one ore more tables. |
| items | Switch |
| syntax | **SEND_KEYS.segment.module.buttonAction.buttonAction .buttonAction[.buttonAction].binary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**buttonAction:** The 3 (or 4) buttonActions represent the different button tables A through C (or D). Possible values are: *SHORT, LONG, RELEASE and -*.<br>SHORT: Sends a short command to the table.<br>LONG: Sends a long command to the table.<br>RELEASE: Sends a release command to the table.<br>- : Sends no command to the table.<br>**binary:** are the binary states. This argument consists of 8 characters which can be either of the following characters: 1 (press button), 0 (do nothing). |
| example | **SEND_KEYS.0.23.SHORT.-.-.-.1000000** |
| | `Switch SendButton "Send Button" {lcn="[ON:local:SEND_KEYS.0.23.SHORT.-.-.-.10000000]"}` |
| | This command will send a 'short' command to the first button of Table A. |
| notes | |



| | |
|---|---|
| **DELAY_ KEYS** | The 'DELAY_KEYS' command will send a 'short' command to one or more buttons of a table with a delay. Will always send a 'short' command. |
| items | Switch |
| syntax | **DELAY_KEYS.segment.module.table.value.advTimeUnit.binary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254. |

| | |
|---|---|
| | **table:** defines the target table. Possible values are A, B, C and D.<br>**value:** the amount of time for the delay. Maximum values depend on the timeunit.<br>**advTimeunit:** the unit of time, either S (1 - 60 seconds), M (1 – 90 minutes), H (1 – 50 hours) or D (1 – 45 days).<br>**binary:** are the binary states. This argument consists of 8 characters which can be either of the following characters: 1 (press button), 0 (do nothing). |
| example | **DELAY_KEYS.0.23.B.10.S.00100000** |
| | `Switch DelayButton "Delay Button" {lcn="[ON:local:DELAY_KEYS.0.23.B.10.S.00100000]"}` |
| | After 10 seconds, this command will send a 'short' command to the 3rd button of table B. |
| notes | |

| | |
|---|---|
| **LOCK_KEYS** | The 'LOCK_KEYS' command will (un-)lock one or more buttons of a single table. |
| items | Switch |
| syntax | **LOCK_KEYS.segment.module.table.advBinary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**table:** defines the target table. Possible values are A, B, C and D.<br>**advBinary:** are the binary states. This argument consists of 8 characters which can be either of the following characters: 1 (lock), 0 (unlock) or U (switch lock). |
| example | **LOCK_KEYS.0.23.C.11111111** |
| | `Switch LockButton "Lock Button" {lcn="[ON:local:LOCK_KEYS.0.23.C.11111111]"}` |
| | This command will lock all buttons of table C. |
| notes | |

| | |
|---|---|
| **TIMELOCK_KEYS** | The 'TIMELOCK_KEYS' command will (un-)lock one or more buttons of table A. Does only work with table A! |

| | |
|---|---|
| items | Switch |
| syntax | **TIMELOCK_KEYS.segment.module.value.advTimeUnit .simpleBinary** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**value:** the amount of time for the lock. Maximum values depend on the timeunit.<br>**advTimeUnit:** the unit of time, either S (1 - 60 seconds), M (1 – 90 minutes), H (1 – 50 hours) or D (1 – 45 days).<br>**simpleBinary:** are the binary states. This argument consists of 8 characters which can be either of the following characters: 1 (lock), 0 (dont lock). |
| example | **TIMELOCK_KEYS.0.23.10.M.10000000** |
| | `Switch TimelockButton "Timelock Button" {lcn="[ON:local:TIMELOCK_KEYS.0.23.10.M.10000000]"}` |
| | This command will lock the first button of table A for 10 minutes. |
| notes | |

| **LED** | The 'LED' command will control LEDs, switch them on and off, let them flicker or flash. |
|---|---|
| items | Switch |
| syntax | **LED.segment.module.ledNumber.ledAction** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**ledNumber:** the ID of the LED, values are between 1 and 12.<br>**ledAction:** possible values are: *OFF, ON, BLINK* and *FLICKER.*<br>OFF: switches the LED to off.<br>ON: switches the LED to on.<br>BLINK: blinks the LED.<br>FLICKER: flickers the LED. |
| example | **LED.0.23.2.ON** |
| | `Switch Led "LED" {lcn="[local:LED.0.23.2.ON]"}` |

| | |
|---|---|
| | Switches the 2nd LED of the module 23, in segment 0, to on. |
| notes | |

| | |
|---|---|
| **LED_STATE** | The 'LED_STATE' command will request the current state of the LEDs of a single LCN module. |
| items | String |
| syntax | **LED_STATE.segment.module[.ledNumber]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**ledNumber:** (optional) the ID of the LED, values are between 1 and 12. |
| example | **LED_STATE.0.23** |
| | `String LED_STATE "LED: [%s]" {lcn="[local:LED_STATE.0.23]"}` |
| | Reads the states of the LEDs of module 23, in segment 0, and print them at the location of '%s'. |
| notes | |

| | |
|---|---|
| **VAR_ADD** | The 'VAR_ADD' command will increase the a variable of a LCN module. |
| items | Switch, Dimmer |
| syntax | **VAR_ADD.segment.module.dataID.bigValue[.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**dataID:** is the ID of the variable, maximum is 12. Please note, that older modules may have significantly less variables.<br>**bigValue:** value between 0 and 30000<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |
| example | **VAR_ADD.0.23.1.2500** |

| | |
|---|---|
| | `Switch VAR_ADD "Var Add" {lcn="[ON:local:VAR_ADD.0.23.1.2500]"}` |
| | This command will increase the first variable of module 23, in segment 0, by 2500. |
| notes | |

| **VAR_SUB** | The 'VAR_SUB' command will decrease a variable of a LCN module. |
|---|---|
| items | Switch, Dimmer |
| syntax | **VAR_SUB.segment.module.dataID.bigValue[.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**dataID:** is the ID of the variable, maximum is 12. Please note, that older modules may have significantly less variables.<br>**bigValue:** value between 0 and 30000<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |
| example | **VAR_SUB.0.23.1.720** |
| | `Switch VAR_SUB "Var Sub" {lcn="[ON:local:VAR_SUB.0.23.1.720]"}` |
| | This command will decrease the first variable of module 23, in segment 0, by 720. |
| notes | |



| **THRESHOLD_VALUE** | The 'THRESHOLD_VALUE' command will request a list of all thresholds of a LCN module. |
|---|---|
| items | String |
| syntax | **THRESHOLD_VALUE.segment.module[.thresholdRegister].thresholdNumber[.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254. |

| | |
|---|---|
| | **thresholdRegister:** only for modules from 2012 or later. Value between 1 and 4.<br>**thresholdNumber:** Value between 1 and 4 (5 for modules from before 2012).<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |
| example | **THRESHOLD_VALUE.0.23.1.1** |
| | `String THRESHOLD_VALUE "Threshold Value: [%s]" {lcn="[local:THRESHOLD_VALUE.0.23.1.1]"}` |
| | This command will request threshold 1, in register 1, of module 23, in segment 0, and print it at the location of '%s'. |
| notes | |

| | |
|---|---|
| **MOVE_ THRESHOLD** | The 'MOVE_THRESHOLD' command move a single threshold of a LCN Module. This command is for modules with firmware 160B13 or newer only. |
| items | String |
| syntax | **MOVE_THRESHOLD.segment.module.origin.thresholdValue .operator.thresholdRegister.thresholdNumber[.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**origin:** is either PROG or CURRENT.<br>PROG: moves relative to the programmed threshold.<br>CURRENT: moves relative to the current threshold.<br>**thresholdValue:** is the value, by which the threshold is moved. Values between 1 and 1000 are valid.<br>**operator:** is either + or -<br>**thresholdRegister:** the register of the target threshold, maximum 4.<br>**thresholdNumber:** the ID of the target threshold, maximum 4.<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |

| | |
|---|---|
| example | **MOVE_THRESHOLD.0.23.CURRENT.100.-.2.2** |
| | `Switch Move_Threshold "Move Threshold" {lcn="[local:MOVE_THRESHOLD.0.23.CURRENT.100.-.2.2]"}` |
| | This command will decrease the threshold number 2, in register 2, for module 23, in segment 0, in relation to the current threshold, by 100. |
| notes | |

| | |
|---|---|
| **MOVE_THRESHOLD_OLD** | The 'MOVE_THRESHOLD_OLD' command will move one or more thresholds of a LCN Module. This command is for modules with firmware 140C0D or older only. |
| items | String |
| syntax | **MOVE_THRESHOLD_OLD.segment.module.origin .thresholdValue.operator.binary[.modifier]** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**origin:** is either PROG or CURRENT.<br>PROG: moves relative to the programmed threshold.<br>CURRENT: moves relative to the current threshold.<br>**thresholdValue:** is the value, by which the threshold is moved. Values between 1 and 1000 are valid.<br>**operator:** is either + or -<br>**binary:** marks the thresholds, that are to be moved. This argument consists of 5 characters which can be either of the following characters: 1 (move threshold), - (do nothing), 0 (do nothing).<br>**[modifier]:** ((optional) transforms the output value to the desired unit. Possible values are: CELSIUS, LUX, LUX_T, VOLT, AMP, WIND, MOISTURE, CO2 and NONE. (Attention: for the cases of LUX and LUX_T, it is not possible to convert values, which are to be sent to the LCN bus!) |
| example | **MOVE_THRESHOLD_OLD.0.23.PROG.100.+.01-10** |
| | `Switch Move_Threshold_Old "Move Threshold (old)" {lcn="[local:MOVE_THRESHOLD_OLD.0.23.PROG.100.+.01-10]"}` |
| | This command will increase the thresholds 2 and 4, for module 23, in segment 0, in relation to the programmed threshold, by 100. |
| notes | |

| **GET_INFO** | The 'GET_INFO' command will read the name and/or commentary of a LCN module. |
|---|---|
| items | String |
| syntax | **GET_INFO.segment.module.infoID** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**infoID:** possible values are: *NAME1, NAME2, COMMENT1, COMMENT2* and *COMMENT3.*<br>NAME(i): reads part (i) of the name of the module.<br>COMMENT(i): reads part (i) of the comment of the module. |
| example | **GET_INFO.0.23.NAME1** |
| | `String Get_Info "Get Info [%s]" {lcn="[local:GET_INFO.0.23.NAME1]"}` |
| | This command will read part 1 of the name of module 23, in segment 0, and print them at the position of '%s'. |
| notes | |

| **GET_OEM** | The 'GET_OEM' command will read the OEM information of a LCN module. |
|---|---|
| items | String |
| syntax | **GET_OEM.segment.module.oemID** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**oemID:** the ID of the OEM text, values are between 1 and 4. |
| example | **GET_OEM.0.23.1** |
| | `String Get_OEM "Get OEM: [%s]" {lcn="[local:GET_OEM.0.23.1]"}` |
| | This command will read part 1 of the OEM information of module 23, in segment 0, and print them at the position of '%s'. |
| notes | |

| **GROUPS** | The 'GROUPS' command will add or remove a LCN module to/from a dynamic group. |
|---|---|
| items | Switch |
| syntax | **GROUPS.segment.module.operator.group** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**operator:** is either + or -<br>**group:** the ID of the group. Between 5 and 254. |
| example 1 | **GROUPS.0.23.+.10** |
| | `Switch Groups "Add To Group" {lcn="[ON:local:GROUPS.0.23.+.10]"}` |
| | This command will add module 23, of segment 0, to group 10. |
| example 2 | GROUPS.0.23.-.0 |
| | `Switch Del_Groups "Delete Groups" {lcn="[ON:local:GROUPS.0.23.-.0]"}` |
| | This is a special command and will delete all dynamic groups! |
| notes | |

| **BEEP** | The 'BEEP' command will send a beep signal to a LCN module. |
|---|---|
| items | Switch |
| syntax | **BEEP.segment.module.beepType.amount** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**beepType:** possible values are SPECIAL and NORMAL.<br>**amount:** is the amount of beeps, maximum is 15. |
| example | **BEEP.0.23.NORMAL.5** |
| | `Switch Beeper "Beep" {lcn="[ON:local:BEEP.0.23.NORMAL.5]"}` |
| | This command will send 5 normal beeps to module 23, in segment 0. |
| notes | |

| **TEXT** | The 'TEXT' command will set the text of a displaymodule. |
|---|---|
| items | Switch |
| syntax | **TEXT.segment.module.line.block.message** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**line:** is the ID of the text line, between 1 and 4.<br>**block:** is the ID of the text block, between 1 and 5.<br>**message:** up to 12 arbitrary characters. |
| example | **TEXT.0.23.2.4.welcome** |
| | `Switch Txt "Text" {lcn="[ON:local:TEXT.0.23.2.4.welcome]"}` |
| | This command will set the text of line 2, of block 4, of module 23, in segment 0, to “welcome”. |
| notes | Some characters, like 'ü' and 'ö', take up two or even more character spots. If your text is too long, the excess characters will be automatically discarded. |

| **TIMED_ TEXT** | The 'TIMED_TEXT' command will set the duration of a single text line of a displaymodule. |
|---|---|
| items | Switch |
| syntax | **TEXT_TIMED.segment.module.line.ramp** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**line:** is the ID of the text line, between 1 and 4.<br>**ramp:** is the time, that the line should be displayed for. A list about possible values can be found in the appendix, maximum is 255. |
| example | **TIMED_TEXT.0.23.2.9** |
| | `Switch Timed_Txt "Timed Text" {lcn="[ON:local:TIMED_TEXT.0.23.2.9]"}` |
| | This command will set set the display time of line 2, of module 23, in segment 0, to 5 seconds. |
| notes | |

| **MRS** | The 'MRS' command will control a MRS device. |
|---|---|
| items | Switch |
| syntax | **MRS.segment.module.mrsCommand.<mrsCommand>Action** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**mrsCommand:** possible values are: *CALL, VOLUME, SOURCE, RADIO, EQUALIZER, IMPRESSION* and *STANDBY.*<br>CALL: general call actions<br>START.<band>: starts a certain band. Values for <band> are 1 – 16.<br>END: ends all.<br>VOLUME.<mrsOutput>: volume actions. Values for <mrsOutput> are 1 – 12 or OPTICAL.<br>UP.<volume>: increases volume by one step. Values for <volume> are 0 – 7.<br>DOWN.<volume>: decreases volume by one step. Values for <value> are 0 – 7.<br>ONGOING_UP.<volume>: steadily increases volume. Values for <volume> are 0 – 7.<br>ONGOING_DOWN.<volume>: steadily decreases volume. Values for <volume> are 0 – 7.<br>RAMP_STOP: stops the ramp for ONGOING_UP or ONGOING_DOWN.<br>SET.<percent>: sets the volume to <percent>. Values for <percent> are 0 – 100.<br>SET_GROUP.<percent>: sets the volume for the entire sync group to <percent>. Values for <percent> are 0 – 100.<br>MUTE: toggles the mute state.<br>SOURCE.<mrsOutput>: source actions. Values for <mrsOutput> are 1 – 12 or OPTICAL.<br>SET.<source>: sets the source. Values for <source> are:<br>1 – 6, OPTICAL or WEBRADIO.<br>PREVIOUS: selects the previous source.<br>NEXT: selects the next source.<br>RADIO: web radio actions.<br>PLAY: plays the web radio.<br>STOP: stops the web radio.<br>PREVIOUS: selects the previous radio channel. |

| | |
|---|---|
| | NEXT: selects the next radio channel.<br>SET.<byteValue>: selects a channel. Values for <byteValue> are 0 – 255.<br>EQUALIZER.<mrsOutput>.<band>.<db>: equalizer actions.<br><mrsOutput>: sets the output, possible values are 1 – 12 or OPTICAL.<br><band>: sets the band, possible values are 1 – 6.<br><db>: sets the volume, possible values are -15 to 15.<br>IMPRESSION.<mrsOutput>: impression actions. Values for <mrsOutput> are 1 – 12 or OPTICAL.<br>LOAD.<impression>: loads an impression. Values for <impression> are 1 – 16.<br>SAVE.<impression>: saves an impression. Values for <impression> are 1 – 16.<br>STANDBY: activates the standby mode.<br><mrsCommand>Action: depending on the mrsCommand, an appropriate action has to be chosen. All available commands are listed under mrsCommand. |
| example | **MRS.0.23.VOLUME.7.SET.70** |
| | `Switch Mrs "MRS" {lcn="[ON:local:MRS.0.23.VOLUME.7.SET.70]"}` |
| | This command will set set the volume of the MRS output 7 to 70%, at module 23, in segment 0. |
| notes | |

| **LANGUAGE** | The 'LANGUAGE' command will set the language for the I-port (i.e. MRS, GT4D, GT10D). |
|---|---|
| items | Switch |
| syntax | **LANGUAGE.segment.module.language** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**language:** is the language. Possible values are: *DE, EN, ES, FR, RU, AR, PL,* and *TR*.<br>DE: German<br>EN: English<br>ES: Spanish<br>FR: French |

| | |
|---|---|
| | RU: Russian<br>AR: Arabian<br>PL: Polish<br>TR: Turkish |
| example | **LANGUAGE.0.23.EN** |
| | `Switch Lang "Language" {lcn="[ON:local:LANGUAGE.0.23.EN]"}` |
| | This command will set the language for the I-port of module 23, in segment 0, to English. |
| notes | |

The following commands are used internally and are usually of little use to the user. For the sake of completeness, these commands are listed here anyways.

| **GET_ COUPLER** | The 'GET_COUPLER' command will request information about the segment couplers. |
|---|---|
| items | Switch |
| syntax | **GET_COUPLER** |
| | *no parameter required* |
| example | **GET_COUPLER** |
| | `Switch Get_Coupler "Get Coupler" {lcn="[ON:local:GET_COUPLER]"}` |
| | This command will request information of the segment coupler. |
| notes | |

| **STATUS** | The 'STATUS' command will request status feedback from a LCN module. |
|---|---|
| items | Switch |
| syntax | **STATUS.segment.module.statusCommand** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254.<br>**statusCommand:** possible values are: OUTPUT*S, PPORT, RELAYS, BINARY* and *ALL.*<br>OUTPUTS: requests status feedback from the outputs.<br>PPORT: requests status feedback from the P-port.<br>RELAYS: requests status feedback from the relays.<br>BINARY: request status feedback from the binary sensors.<br>ALL: request status feedback from all devices. |
| example | **STATUS.0.23.ALL** |
| | `Switch Status "Status" {lcn="[ON:local:STATUS.0.23.ALL]"}` |
| | This command will request status feedback from all devices of module 23, in segment 0. |
| notes | |

| **SN** | The 'SN' command is used to receive the serial number of the target LCN module. |
|---|---|
| items | String |
| syntax | **SN.segment.module** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254. |
| example | **SN.140.77** |
| | `String Serial_Test "Serial [%s]" {lcn="[local:SN.140.77]` |
| | This command will receive the serial number of module 77, in segment 140, and print it at the position of '%s'. |
| notes | |

| **FW** | The 'FW' command is used to receive the firmware version of the target LCN module. |
|---|---|
| items | String |
| syntax | **FW.segment.module** |
| | **segment:** is the ID of the segment, where the targeted LCN module is located, valid values are 0 or between 5 and 127.<br>**module:** is the ID of the module itself, values between 5 and 254. |
| example | **FW.140.77** |
| | `String FW_Test "Firmware [%s]" {lcn="[local:FW.140.77]` |
| | This command will receive the firmware version of module 77, in segment 140, and print it at the position of '%s'. |
| notes | |

# 5 Appendix

Inside the appendix you will find some useful materials and information, that may help you with certain aspects of the LCN-binding. Those are a table for different ramp values and their actual meanings, as well as a table for relay times used by the 'RELAY_TIMER' command.

## *5.1 Ramp table*

This table explains the different possible values for ramps. Please note, that the time values are not linear until ramp values of 10 and beyond.

| Ramp | Seconds | | Ramp | Seconds |
|---|---|---|---|---|
| 0 | 0.00 | | 6 | 2.00 |
| 1 | 0.25 | | 7 | 3.00 |
| 2 | 0.50 | | 8 | 4.00 |
| 3 | 0.66 | | 9 | 5.00 |
| 4 | 1.00 | | 10...250 | 2 * (ramp - 10) + 6 |
| 5 | 1.40 | | | |

## 5.2 Relay timer table

This table explains the different possible values for the relay timer command. Just as with the ramp table above, the values are neither linear nor proportional.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0,03 | 32 | 0,96 s | 64 | 2,88 s | 96 | 6,72 s | 128 | 14,4 s | 160 | 29,76 s | 192 | 60,48 s | 224 | 121,92 s |
| 1 | 0,03 s | 33 | 1,02 s | 65 | 3 s | 97 | 6,96 s | 129 | 14,88 s | 161 | 30,72 s | 193 | 62,4 s | 225 | 125,76 s |
| 2 | 0,06 s | 34 | 1,08 s | 66 | 3,12 s | 98 | 7,2 s | 130 | 15,36 s | 162 | 31,68 s | 194 | 64,32 s | 226 | 129,6 s |
| 3 | 0,09 s | 35 | 1,14 s | 67 | 3,24 s | 99 | 7,44 s | 131 | 15,84 s | 163 | 32,64 s | 195 | 66,24 s | 227 | 133,44 s |
| 4 | 0,12 s | 36 | 1,2 s | 68 | 3,36 s | 100 | 7,68 s | 132 | 16,32 s | 164 | 33,6 s | 196 | 68,16 s | 228 | 137,28 s |
| 5 | 0,15 s | 37 | 1,26 s | 69 | 3,48 s | 101 | 7,92 s | 133 | 16,8 s | 165 | 34,56 s | 197 | 70,08 s | 229 | 141,12 s |
| 6 | 0,18 s | 38 | 1,32 s | 70 | 3,6 s | 102 | 8,16 s | 134 | 17,28 s | 166 | 35,52 s | 198 | 72 s | 230 | 144,96 s |
| 7 | 0,21 s | 39 | 1,38 s | 71 | 3,72 s | 103 | 8,4 s | 135 | 17,76 s | 167 | 36,48 s | 199 | 73,92 s | 231 | 148,8 s |
| 8 | 0,24 s | 40 | 1,44 s | 72 | 3,84 s | 104 | 8,64 s | 136 | 18,24 s | 168 | 37,44 s | 200 | 75,84 s | 232 | 152,64 s |
| 9 | 0,27 s | 41 | 1,5 s | 73 | 3,96 s | 105 | 8,88 s | 137 | 18,72 s | 169 | 38,4 s | 201 | 77,76 s | 233 | 156,48 s |
| 10 | 0,3 s | 42 | 1,56 s | 74 | 4,08 s | 106 | 9,12 s | 138 | 19,2 s | 170 | 39,36 s | 202 | 79,68 s | 234 | 160,32 s |
| 11 | 0,33 s | 43 | 1,62 s | 75 | 4,2 s | 107 | 9,36 s | 139 | 19,68 s | 171 | 40,32 s | 203 | 81,6 s | 235 | 164,16 s |
| 12 | 0,36 s | 44 | 1,68 s | 76 | 4,32 s | 108 | 9,6 s | 140 | 20,16 s | 172 | 41,28 s | 204 | 83,52 s | 236 | 168 s |
| 13 | 0,39 s | 45 | 1,74 s | 77 | 4,44 s | 109 | 9,84 s | 141 | 20,64 s | 173 | 42,24 s | 205 | 85,44 s | 237 | 171,84 s |
| 14 | 0,42 s | 46 | 1,8 s | 78 | 4,56 s | 110 | 10,08 s | 142 | 21,12 s | 174 | 43,2 s | 206 | 87,36 s | 238 | 175,68 s |
| 15 | 0,45 s | 47 | 1,86 s | 79 | 4,68 s | 111 | 10,32 s | 143 | 21,6 s | 175 | 44,16 s | 207 | 89,28 s | 239 | 179,52 s |
| 16 | 0,48 s | 48 | 1,92 s | 80 | 4,8 s | 112 | 10,56 s | 144 | 22,08 s | 176 | 45,12 s | 208 | 91,2 s | 240 | 183,36 s |
| 17 | 0,51 s | 49 | 1,98 s | 81 | 4,92 s | 113 | 10,8 s | 145 | 22,56 s | 177 | 46,08 s | 209 | 93,12 s | 241 | 187,2 s |
| 18 | 0,54 s | 50 | 2,04 s | 82 | 5,04 s | 114 | 11,04 s | 146 | 23,04 s | 178 | 47,04 s | 210 | 95,04 s | 242 | 191,04 s |
| 19 | 0,57 s | 51 | 2,1 s | 83 | 5,16 s | 115 | 11,28 s | 147 | 23,52 s | 179 | 48 s | 211 | 96,96 s | 243 | 194,88 s |
| 20 | 0,6 s | 52 | 2,16 s | 84 | 5,28 s | 116 | 11,52 s | 148 | 24 s | 180 | 48,96 s | 212 | 98,88 s | 244 | 198,72 s |
| 21 | 0,63 s | 53 | 2,22 s | 85 | 5,4 s | 117 | 11,76 s | 149 | 24,48 s | 181 | 49,92 s | 213 | 100,8 s | 245 | 202,56 s |
| 22 | 0,66 s | 54 | 2,28 s | 86 | 5,52 s | 118 | 12 s | 150 | 24,96 s | 182 | 50,88 s | 214 | 102,72 s | 246 | 206,4 s |
| 23 | 0,69 s | 55 | 2,34 s | 87 | 5,64 s | 119 | 12,24 s | 151 | 25,44 s | 183 | 51,84 s | 215 | 104,64 s | 247 | 210,24 s |
| 24 | 0,72 s | 56 | 2,4 s | 88 | 5,76 s | 120 | 12,48 s | 152 | 25,92 s | 184 | 52,8 s | 216 | 106,56 s | 248 | 214,08 s |
| 25 | 0,75 s | 57 | 2,46 s | 89 | 5,88 s | 121 | 12,72 s | 153 | 26,4 s | 185 | 53,76 s | 217 | 108,48 s | 249 | 217,92 s |
| 26 | 0,78 s | 58 | 2,52 s | 90 | 6 s | 122 | 12,96 s | 154 | 26,88 s | 186 | 54,72 s | 218 | 110,4 s | 250 | 221,76 s |
| 27 | 0,81 s | 59 | 2,58 s | 91 | 6,12 s | 123 | 13,2 s | 155 | 27,36 s | 187 | 55,68 s | 219 | 112,32 s | 251 | 225,6 s |
| 28 | 0,84 s | 60 | 2,64 s | 92 | 6,24 s | 124 | 13,44 s | 156 | 27,84 s | 188 | 56,64 s | 220 | 114,24 s | 252 | 229,44 s |
| 29 | 0,87 s | 61 | 2,7 s | 93 | 6,36 s | 125 | 13,68 s | 157 | 28,32 s | 189 | 57,6 s | 221 | 116,16 s | 253 | 233,28 s |
| 30 | 0,9 s | 62 | 2,76 s | 94 | 6,48 s | 126 | 13,92 s | 158 | 28,8 s | 190 | 58,56 s | 222 | 118,08 s | 254 | 237,12 s |
| 31 | 0,93 s | 63 | 2,82 s | 95 | 6,6 s | 127 | 14,16 s | 159 | 29,28 s | 191 | 59,52 s | 223 | 120 s | 255 | 240,96 s |