SOUNDLIGHT
The DMX Company

last updated: 23-05-12

THE DMX RDM MANUAL

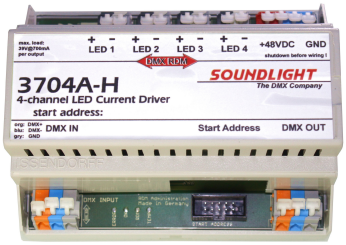
SOUNDLIGHT *The DMX Company* Bennigser Str. 1 D-30974 Wennigsen Tel. 05045-912 93-11

Thank you for choosing SOUNDLIGHT !

Nearly all SOUNDLIGHT devices are supporting DMX RDM, and a number of competitors have also begun to integrate RDM functionality within their products. RDM is coming!

If you have had the chance to try DMX RDM, you won't want to miss it anymore. You will need a capable RDM controller to take advantage of all RDM functons - we will be dealing with this issue later.

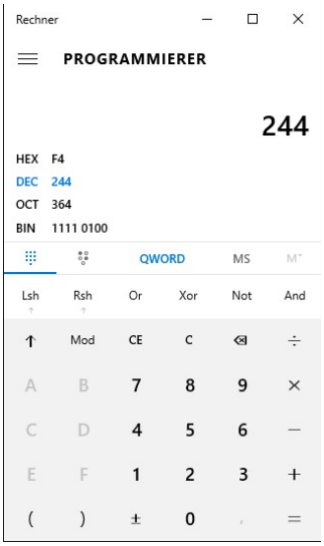
Applications



Let us discuss the advantages of DMX RDM first: the communication on the data bus is bidirectional, taht is, that the responder connected does not only receive control data from the controller (as with standard DMX), but also can return data to the controller. This allows to make steiings and issue queries as well, upon which the response will give a requested response. This is true communication! Getting a response from the connected DMX device, either a simple acknowledge „I have successfully received and understood your command" or a response packed with data „the current temperature is 21 degrees Celsius" is opening a new world to DMX, true **R**emote **D**evice **M**anagement (RDM) of connected devices.

Until now, several RDM standards have been issued. The basic document is ANSI E1-20, which defines the basics and the basic RDM command set. Additionally, some additional documents have been issued. Among these are E1-37.1, E1-37.2, E1-37.4 and E1-37.5, which define many additional commands. SOUNDLIGHT devices make extensive use of E1-37 commands, which allow timing and frequency settings, dimmer curves and more. All DMX standards are available as ANSI standards, which can be purchased from the American National Standards Institute (ANSI, www.ansi.org).

Additionally, there are numerous manufacturer specific commands (PIDs). The number, types and meanings of the parameters used are defined by the respective manufacturer to allow manufacturer-specific configuration of the responder. Unfortunately the RDM standard E1-20 only describes very simple manufacturer specific commands SMSC: Simple Manufacturer Specific Commands) and lacks a definition of multi-parameter command structure. SOUNDLIGHT also specifies multi-parameter commands (CMSC: Complex Manufacturer Specific Commands), CSMC allow text and data including different data structures. All commands used are listed and explained below.

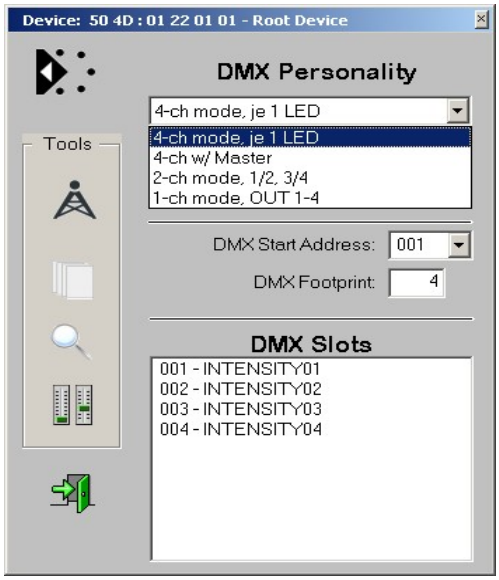


RDM Website

All information regarding DMX RDM can be found on our internet website at: http://www.soundlight.eu/rdm. There you will also find sevarl examples how to issue special commands using different RDM controllers.

Data formats

RDM commands (PIDs) have been defined using hexadecimal notation, that is, digits 0...9, and A,B,C,D,E,F. Use the Windows calculator in Programmer mode to simply convert between hexadecimal and decimal values. Some RDM controllers prefer using decimal entries, since this seems more user-friendly. Other controllers refer to hex format only. Be



careful to also check the separators used when multiple paramaters are required; usually a space or a comma are used to separate entries.

.

Last not least please check the range of commands supported by the controller. There are some controllers just supporting start address and personality, others may support the full 1-37 command set as well as all kinds of manufacturer specific commands.

We recommend the JESE GET/SET Controller, which also displays a number of application-specific menus for easy data entry. For more info pls refer to www.jese.co.uk

The GET/SET controller features several pre-defined masks for swift data entry.

GET and SET

DMX RDM makes use of three different command classes::

1. **DISCOVERY** to automatically identify and retrieve attached DMX RDM responders

2. **GET** to query parameters and settings

3. **SET** to transmit pratmeters and make the desired settings

That's why the JESE controller is called *„GET/SET*". Of course it will also perform a full discovery..

Smart Controller?

Except providing a smart user interface, RDM controllers just have to follow the syntax defined for RDM communiction- they do not need to know anything about the setup they are controlling because all data are stored in the responders. Of course a responder could collect data in advance to have them present when used, but sending a query to the actual device to get the most updated data is a perfact solution. All data, be it integers, floating point or even text data, are stored within the RDM responders and the main job of the controller is to read and display these data, to sort (when needed) or remove/change duplicates (e.g. start adress overlaps) and things like that.

Here's how it works::

1. DISCOVERY

Each DMX RDM responder has a (worldwide) **U**nique **ID**, called UID. This UID is composed of six hexadecimal bytes, with the first two bytes representing the manufacturer ID, and the remaining bytes are the serial identification. Manufacturer IDs are administered and assigned solely by ESTA.org. To dicover a device, the controller defines a address range (e.g. 53 4C 00 00 00 00 ... 53 4C FF FF FF FF and checks the response received. If there is no response, this address range is empty: there are no responders present. If there is a response, but unreadable, there may be multiple responders answering: the address span must be narrowed. If finally there is a readable response, there is just one responder present, the controller can store that UID and start to query the responder for its data.
The manufacturer ID list can be downloaded from the ESTA website (www.esta.org). The

registered SOUNDLIGHT manufacturer ID is „SL“, respective „53 4C“ (see: http://tsp.esta.org/tsp/working_groups/CP/mfctrIDs.php). The unique serial number must be defined by the manufacturer. We use a simple scheme: UID „53 4C 36 03 12 34“ means: this is a SOUNDLIGHT device (53 4C), Device type is „36 03“ and serial is „12 34“.

2. GET-Commands

Each responder can now be indidually addressed using the retrieved UID. Using a GET command allows to read data from the responder. Each Command has a specific number, called „PID“. To read or to set a start addess PID 00F0*hex* must be used. Please refer to standard ANSE E1-20 to retrieve the basic command set. The PID is just for the controller communication; for the user interface, the command should be labelled DMX Startaddress“ instead. The most important PIDs are „DMX Startaddress“ and „DMX Personality“

3. SET-Kommandos

If the controller wishes to set different values, e.g. change the DMX startaddress, it must call the same PID using a SET command. Additionally, the necessary parameters must be passed - so in total the sender UID, the receiver UID, the command class (SET), the PID, a parameter data length (PDL) and the parameters must be sent. Please refer to the standard for a full description of the data packet.

PIDs and Commands

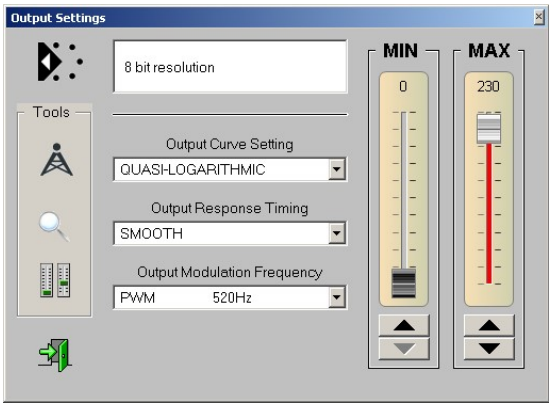
PIDs can be sorted in four groups:

1. Standard commands to be processed by each RDM responder

These PIDs comprise functions necessary for the implementation of DMX RDM. This is the minimum command set a responder has to process. This set includes all discovery commands, DEVICE INFO, the list of SUPPORTED PARAMETERS, and the STARTADDRESS. There are a lot of responders available on the market who can barely do more - we call it „trash“.

2. Standard commands

Standard commands comprise all PIDs described in the actual standard documents. The command syntax and the lsit of parameters are defined in the standard, thus each controller should know how to process these commands (watch out: many controllers do not process all commands!). Smart controllers will offer user-friendly masks to enter data (see picture right, taken from GET/SET Controller). Standard controller may offer a simple generic entry field for a parameter list; in this case the user must refer to the standard to check the requirements.



The PID range for standard commands is 0001*hex* to 7FFF*hex*.

3. Manufacturer Specific Commands

The range from 8000*hex* to FFDF*hex* has been reserved for manufacturer specific commands. MSC are intended for specialfunctions not defined in the standard. All products of one manufacturer must refer to the same PID when calling a special command. This means, that different manufacturer may assign other functions to the same PID: thus be careful

when processing MSCs. Checking a MSC also requires checking the respective manuafacturer ID.

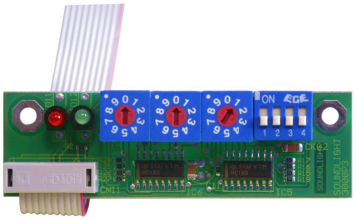
4. PLASA / ESTA commands

The PID range FFE0***hex*** to FFFF*hex* has been reserved for the standards committee and serves for the development of new standard functions (prototyping). Commercially sold products must not contain any of these PIDs.

RDM programming

Our RDM responders allow to set all functional parameters using DMX RDM, but also allow to set some parameter using a attachable manual start address board. Since all settings are stored in nonvolatile memory within the responder, the start address board may be removed after the required setting has been made.

There are different models available::



Startadressboard 3000P:

Uses rotary decimal switches and DIP switches to set DMX startaddress, DMX personality and DMX HOLD mode. The current responder state is being signalled with red/green LEDs.

.

Startadressboard 3006P:

Uses a rotary encoder and a liquid crystal display (LCD) to display current settings. The 3006P also reads actual responder settings in real time (even when modified by RDM).

NOTICE: Any RDM decoder can be operated with or without start address board 3000P connected. Please note that all switches become ***disengaged*** and the respective settings are overridden when remotely programming a DMX start address, DMX personality or HOLD mode via RDM. This is indicated by the yellow LED „RDM“ permanently ON.
To re-engage the switches, set the hundreds position to „9“ temporarily (any address from 900 to 999 will do) and wait for a programming cycle to complete. A programming cycle is indicated by the red and the green LED blinking four times alternatively. LED „RDM“ will extinguish and the address switches will take control again.

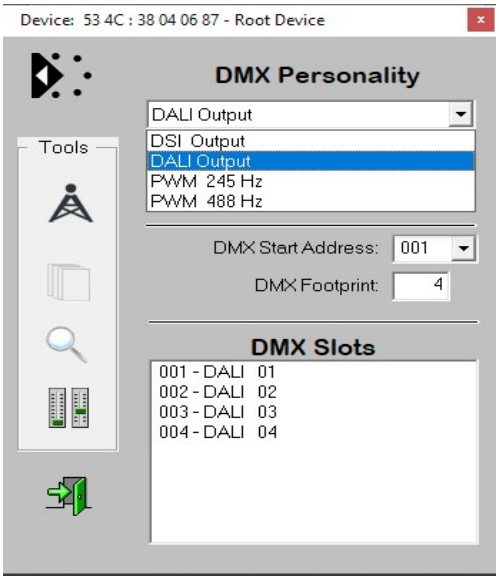
IMPORTANT NOTICE:: All settings can be performed by DMX RDM anytime. Manual settings of the DMX start address, the DMX personality and the DMX HOLD mode can locally be done using any start address board 3000P, 3003P, 3005P, 3006P or 3008P. Addressboards are available as separate accessories. Pls check our website for functions and compatibility.

DMX RDM

Startaddress / Personality

The standard settings are selecting the proper DMX personality (mode of operation) and the DMX startaddress. All controllers should provide masks to enter these data (see picture right: JESE GET/SET controller). Withe the GET/SET controller, both functions are calling the same mask, which allows to check all settings at a glance.

Setting the DMX start address is limited to the allows range 001...512. Many controllers automatically deduct the required device slot count and limit the upper value accordingly, This will result in a maximum DMX start address of 509 for a 4-slot device (pictured).



Special commands

This compilation lists the most important SOUNDLIGHT manufacturer specific commands. While we have made every effort to make sure that this list is complete and up to date we have to reserve the right to make changes or corrections where needed and to add new functions without prior notification. We also do not assume any responsibility for errors and omissions. Please refer to the previously mentioned RDM standards when defining your own MSC. Feel free to copy our implementations for your responder (and let us know). Chances are, there are already controllers supporting our implementation.

PIDs are sorted numerically. Eventually two PDS are listed for the same command. Why this? Some important PIDs from E1-37 have been doubled as manufacturer specific PIDs. Reason:several controllers do not yet know about E1-37 and then reject these PIDs as „unknown PID“. This will not happen when calling the function using a PID in the MSC range. The results are the same, but in this case a generic mask will be presented where parameters can be entered. You may want to refer to the E1-37 standard to check for the parameter lists.

PID 8008: FULL COMMAND LIST VOLLSTÄNDIGE BEFEHLSLISTE

This function defines the length of the manufacturer specific commands section. When selecting „short list“, only DMX HOLD MODE PIDs will be displayed, all other functions will be suppressed or disabled. When selecting „full list“ the complete list will be displayed.

Aufrufe: GET <param = none>
(no parameters required)
Return: <param=Auto_Init [Byte]>

SET <param=List_Type [Byte]>
Return: <param=none>
(no return parameter)

List_Type = $00 short list
List_Type = $FF full list

NOTICE: *Please note, that non-displayed functions will not be available when operating the responder. Additionally, these functions will be disabled.*
Example: enabling „short list“ with a relay module will suppress monostable (and related) functions. Even if the module has been configured as monostable relay, the monostable function will be disabled. This may be used to suppress complex programming with just one command.

NOTICE FOR PROGRAMMERS:
Changing the FULL COMMAND LIST setting may require to run a new discovery to allow the controller refereshing the PID list. Once discovered, however, some controllers do not refresh the PID list. In this case, shut down and restart the controller to read the correct PID list.

PID 8081: DMX HOLD INPUT 1	**Verhalten bei DMX Signalausfall DMX Eingang 1**
PID 8082: DMX HOLD INPUT 2	**Verhalten bei DMX Signalausfall DMX Eingang 2**
PID 8083: DMX HOLD INPUT 3	**Verhalten bei DMX Signalausfall DMX Eingang 3**
PID 8084: DMX HOLD INPUT 4	**Verhalten bei DMX Signalausfall DMX Eingang 4**
PID 8085: DMX HOLD INPUT 5	**Verhalten bei DMX Signalausfall DMX Eingang 5**
PID 8086: DMX HOLD INPUT 6	**Verhalten bei DMX Signalausfall DMX Eingang 6**
PID 80F1: DMX HOLD MODE	**Verhalten bei DMX Signalausfall DMX Eingang allgemein**
PID 80F2: MASTER HOLD MODE	**Verhalten bei DMX Signalausfall Master-Eingang**

The DMX HOLD MODE determines the behaviour at loss of DMX data. There is a similar function (DMX FAIL MODE) in E1-37, but this is much too complex for daily use.
Basically, there are just three (or four) simple options:

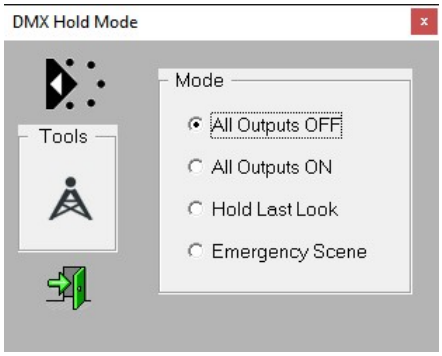
PARAMETER:	1 Byte (HOLD_MODE)
GET:	Aufruf ohne Parameter
Return: Parameter:	1 Byte (HOLD_Modus)
SET:	Calling parameters: 1 Byte (HOLD_Mode)
Return:	Status

Parameter values HOLD_Mode:

Hex	Dec	Function
$00	0	All outputs go to 0% (dark, off)
$01	1	All outputs go to 100% (full on)
$02	2	All outputs remain at the last valid setting „Keep Last Look“
$03*	3*	A predefined scene will be called*

**=not available with all models*

Using a start address board 3000P, the HOLD mode will be set using DIP switches 1 and 2. LCD-Adressboards (3005P, 3006P, 3008P) use a menu to select the requested setting.



Using the GET/SET controller to setup SOUNDLIGHT responders displays a user-friendly mask to select the right DMX HOLD setting.

PID 0141: DMX FAIL MODE **Verhalten bei DMX Signalausfall**
PID 8301: DMX FAIL MODE

The DMX FAIL MODE is a standard PID as defined in E1-37, using multiple parameters for complex HOLD options.
At SOUNDLIGHT, we do not use most of the optional parameters. Below please find a simple conversion table to replace our DMX HOLD command with DMX FAIL MODE settings.

PARAMETER:	7 Bytes, of which: Byte 1,2 = Scene number (16 Bit) Byte 3,4 = Delay after lss of signal in 1/10 seconds (16 Bit) Byte 5,6 = Hold time in 1/10 seconds (16 Bit) Byte 7 = Level
GET:	Called without parameters
Return: Parameter:	7 Byte (Fail-Modus)
SET:	Calling parameters:: 7 Byte (Fail-Modus)
Return:	Status

Conversion from HOLD MODE to FAIL MODE:

HOLD MODE	Function	FAIL MODE
00	all off	00 00 00 00 FF FF 00
01	all on	00 00 00 00 FF FF xx
02	"last look"	00 00 FF FF FF FF 00

When setting the DMX HOLD MODE options only „full on“ can be specified. The DMX FAIL MODE allows to define the output level at loss of signal. Replace the „xx“ with the level to be displayed at signsl loss.
NOTICE. not all responders allow variable output levels at loss of DMX512 control.

All values are given in hexadecimal format!

PID 0640: LOCK PIN
PID 8330: LOCK PIN

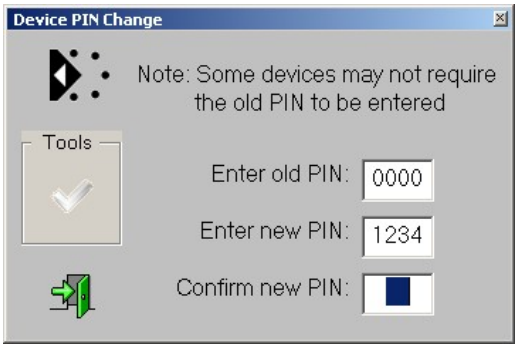
Eingabe einer PIN zur Verriegelung

As with your credit card, you can use a 4-digit PIN to restrict access to various PIDs.
Default PIN is set to 0000 (0000*hex*). The LOCK PIN allows only SET commands, no GET commands (the PIN can never be read!) Using the GET/SET controller, using PID 0640 will display a pre-defined mask, while using PID 8330 will display a generic mask.

To set a new PIN, enter the new PIN followed by the current PIN.
Example: new PIN 0220*hex*, current Pin 1836*hex*:
Using a generic mask, enter 02201836*hex*.
PINs must be within the range 0000*dec* (0000*hex*) to 9999*dec* (270F*hex*), New products are set to start PIN 0000(*dec*) .

IMPORTANT NOTICE: Since PINs cannot be retrieved or reset, make sure you keep your PIN in a safe place. Also please check to use the appropriate numbering system as required by your RDM controller (decimal or hexadecimal) when changing PINs. All modifications are at your own risk!

PARAMETER:	4 Bytes (2 16-bit-words) Byte 1,2: new PIN (16 Bit) Byte 3,4: old PIN (16 Bit))
GET:	no GET call allowed; PIN cannot be retrieved!
SET:	Calling parameters:: 4 Bytes (PIN) Return: Status



Entering a new PIN using the GET/SET Controller

IMPORTANT NOTICE: Make sure to save your PIN in a safe place. PINs cannot be retrieved. Resetting a PIN is only possible at the factory.

PID 0641: LOCK STATE **Verriegelungszustand**
PID 8331: LOCK STATE

Check or change the current lock state.

PARAMETER: 3 Bytes (1x16-bit-word, 1 Byte)
Byte 1,2: PIN
Byte 3: LOCK STATE

GET: Calling parameters: none
Return parameters: 2 Bytes
(LOCK_STATE, TOTAL_LOCK_STATES)

SET: Calling parameters::
3 Bytes (PIN, LOCK_STATE)
Return parameters: Status

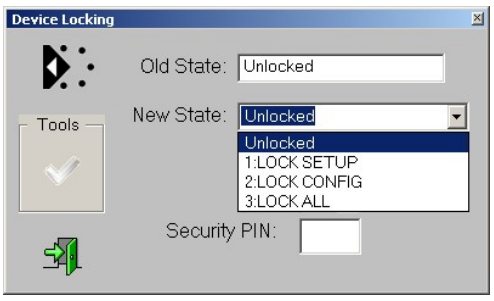
PARAMETERS:
PIN: see PID060 (above)
LOCK_STATE: 00= no lock
01= Setup locked
02= Configuration locked
03= All locked

Example:
Calling GET_LOCK_STATE return two bytes, first byte is current lock state, second byte is the number of available lock states.
- no lock: 0003
- Setup locked: 0103
- Config locked: 0203
- All locked: 0303

Calling SET_LOCK_STATE requires entering the current PIN followed by the desired lock state.
Eingabe: <PIN> <gewünschte Verriegelung>.

Example:
current PIN = 2345dec, Setup Lock required.
Command is: 092901hex.
Make sure to check the numbering system! Since 2345dec is 0929hex. the total command parameter is 092901hex, when the controller expects data entry in hex format.

The LOCK STATE determines the accessibility of several settings (see table in appendix). When locked, the function cannot be acccessed. The responder will display a „Write protected“ message box.



Selecting the LockState using the GET/SET Controller

PID 0642: LOCK STATE DESCRIPTION **Ausgabe einer Beschreibung für die Verriegelung**
PID 8332: LOCK STATE DESCRIPTION

Using a GET command with the respective LOCK_STATE as parameter returns the verbal description of that lock state. There is not SET command available.

PID 1040: IDENTIFY MODE **Identify-Modus**
PID 8340: IDENTIFY MODE

Each DMX RDM responder must be able to accept a IDENTIFY command The identify mode puts the responder in a special state suited to draw the attention to it to be able to identfy it. Thus a moving head may shake its head, while a LED driver may switch to a blinking mode to draw the attention to it.

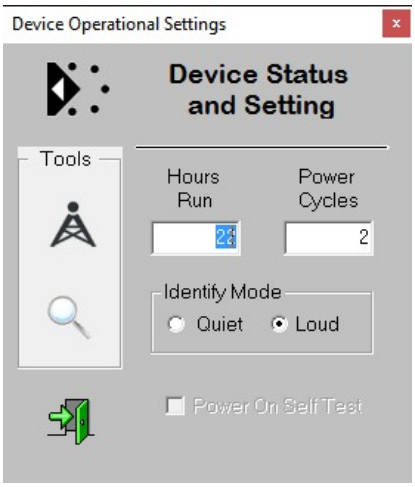
It may, however, be most annoying having a responder to revert to identify mode during showtime. Thus the command IDENTIFY MODE has been created to set the responder to LOUD identify mode (shaking its head) or QUIET identify mode (swich the indicator LEDs or similar). We think, it's a very important command which was not present in the original standard E1-20.

Several responders are preset to QUIET mode to prevent unwanted disruption during showtime (e.g. DMX relays). Check your setup before getting to work.

PARAMETER:	1 Byte (Identify_Mode)
GET:	Called without parameters
Return: Parameter:	1 Byte (Identify_Mode)
SET:	Calling parameters:: 1 Byte (Identify-Modus) Return: Status

Identify-Mode:

Hex	Decimal	Function
00	0	Quiet Mode: e.g. Identify by status LEDs
FF	255	Loud Mode: e.g. Ident switching outputs



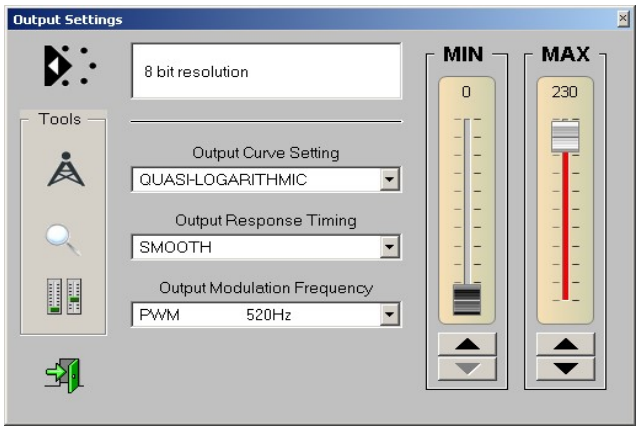
Identify Mode setting using the JESE GET/SET Controller

PID 0341: MINIMUM LEVEL
PID 0342: MAXIMUM LEVEL

Pegel-Begrenzung

Use the functions MINIMUM LEVEL and MAXIMUM LEVEL to limit the allowed output level to the set limits. Using the GET/SET controller, it's a simple task: simply set the faders to the respective values and wait 1...2 seconds until the values have been programmed into the responder. If the outputs are set, you can simply check the results visually.

All settings within the DMX level range 0...255 ($00...$FF) are allowed, but make sure that the maximum level is higher than the minimum level. Also check the correct data format when using a generic mask.



Setting the limits is quite easy when using the JESE GET/SET controller. Just move the faders.

When using a generic mask, 5 Bytes must be transferred as payload:

GET:	Called without parameters	
Return:	5 Bytes, thereof:	
	Byte 1,2:	Minimum-Level upward, 16 Bit
	Byte 3,4:	Minimum-level downward, 16 Bit
	Byte 5:	„On“ Below Minimum
SET:	called with 5 Bytes parameter (see above)	
Return:	Status	

Parameters:
All levels are transferred as 16 bit data. If the level range of the responder is less than 16 bits, the upper n bit are taken into account. The resolution (in bits) must be declared in DIMMER_INFO.

„On Below Minimum“ is either 00*hex* (output goes to zero if level is below minimum level) or 01*hex* (output stays at minimum level when level is below minimum level setting)

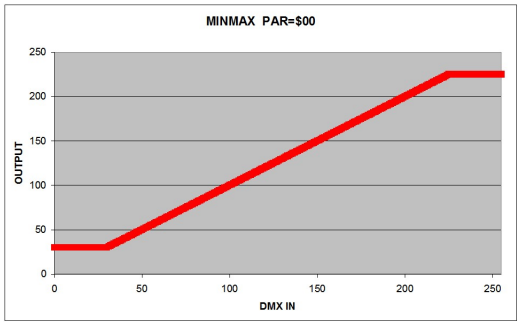
PID 8341: MINMAXMODE MINIMUM-/MAXIMUM-MODUS

This function defines the behaviour of the MINIMUM LEVEL and MAXIMUM LEVEL settings.
There are two basic options for MIN/MAX settings:

a) LIMITING
The output level will be limited to the respective MINIMUM or MAXIMUM settings. Increasing or decreasing the output level will be stopped as soon as the level exceeds the limit values.

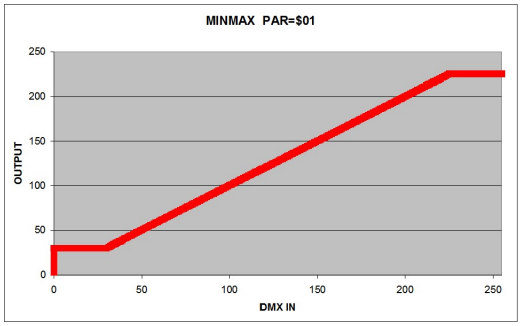
b) SCALING
The output level will be scaled proportionally to not exceed the set MINIMUM or MAXIMUM settings.

Both options can be combined with the „output zero at input zero“ function (see diagrams).

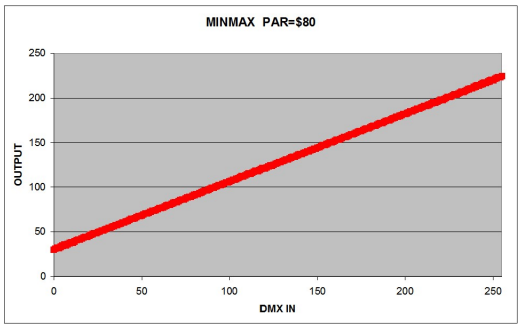


PARAMETER: 1 Byte: MINMAX_MODE

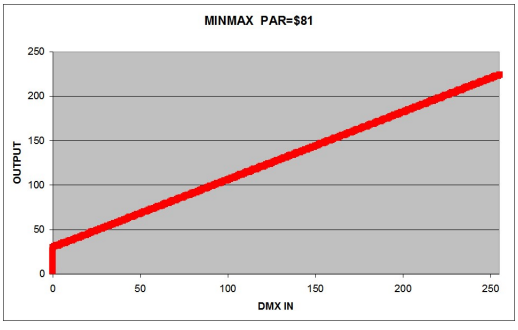
Parameter = $00 (0 dec):
Full Limiting as standard function



Parameter = $01 (1 dec):
Limiting with zero level passing through: higher input levels below set MIN will be output as MINIMUM LEVEL



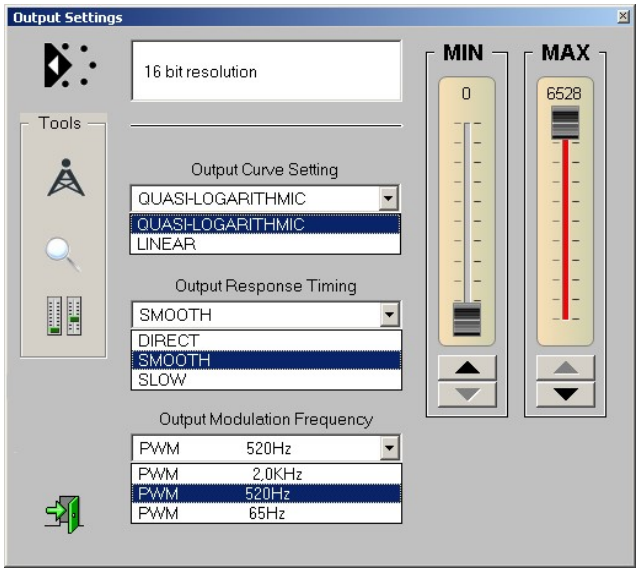
Parameter = $80 (128 dec):
The Limiting will be replaced by continuous scaling: thus the full control range can be used.



Parameter = $81 (129 dec):
Combining scaling and zero pass-through..

PID 0343: CURVE AUSGANGSKURVE

PID 0343 defines the output curve characteristic..



Selecting the output curve using the GET/SET controller: select from the drop-down menu. Done.

The preset curve is "QUASI-LOGARITHMISCH" (curve 01). This applies to most LED drivers, since a logarithmic curve matches the human eye's sensitivity. For compatibility, it may be necessary to change to "LINEAR" (curve 02). Besides, some decoders will offer a „USER CURVE" (curve 03), which (using a free curve editor) can be tailored to your specific needs.

Issue a GET command to retrieve the current setting:

GET	Called without parameters
Return:	2 Bytes: Byte 1: current setting Byte 2: number of available settings
SET:	1 Parameter (1 Byte): new setting

Output Curve settings are counted from 01 onward The example above would allow settings 01, 02 or 03, with the number of available settings being 03. Thus a GET command will automatically return the information how many curves are available. Using that info, you can obtain a curve description by calling the PID OUTPUT CURVE DESCRIPTION and passing the respecting curve number as parameter.

PID 0344: OUTPUT CURVE DESCRIPTION Ausgabe einer Beschreibung für die Kurve

Entering the curve number as parameter will return the curve description in ascii text format. There is no SET command available.

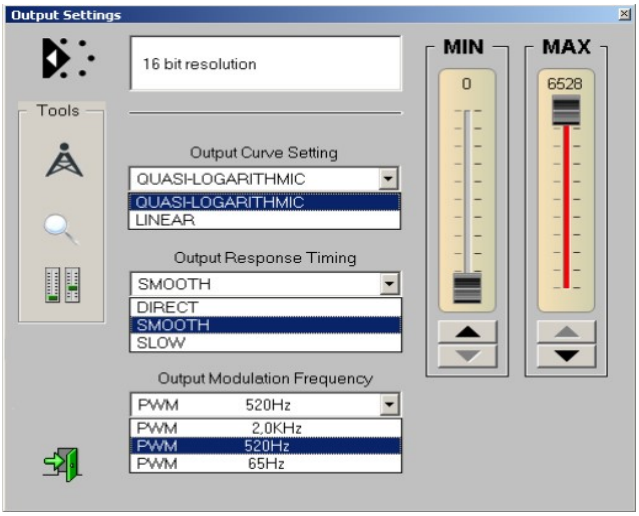
PID 0345: OUTPUT RESPONSE AUSGANGSVERHALTEN

The output behaviour (smoothing) can be set using the OUTPUT RESPONSE TIME function. The preset is "SMOOTH" for optimum output behaviour.
Most responders offer three settings:

1: DIRECT immediate output o received DMX data (fast response but may cause some stepping)

2:SMOOTH Optimum smoothing of incoming data

3: SLOW Slow fade of output.



Selecting the output response using the GET/SET controller: select from the drop-down menu. Done.

Issuing a GET command returns the current setting:

GET	Called without parameters
Return: 2 Bytes:	
	Byte 1: current setting
	Byte 2: number of possible settings
SET:	1 Parameter (1 Byte): new setting
Return: Status	

Output Response settings are counted from 01 onward The example above would allow settings 01, 02 or 03, with the number of available settings being 03. Thus a GET command will automatically return the information, how many output response settings are available. Using that info, you can obtain a output response description calling the PID OUTPUT RESPONSE DESCRIPTION and passing the respecting output response number as parameter.

PID 0346: RESPONSE TIME DESCRIPTION Ausgabe einer Beschreibung für die Glättung

Entering the output response number as parameter will return the output response description in ascii text format. There is no SET command available.

PID 8121: SLOT LABELS **SLOT LABEL SCHREIBEN**
PID 0121: SLOT LABELS*

The PID SLOT LABELS exists in RDM Standard E1-20, but is limited to GET access. Thus slot labels can be read, but cannot be modified by the user. This makes sense for devices with fixed pre-defined slot assignments, but not for general building blocks such as our PWM drivers or relay blocks, where individual slot labelling is essential: e.g. Relay 1 as „Motor“, Relay 2 as „Room Light“, Relay 3 as „Smoke Machine“ etc.

In contrast to the standard our responders allow to write slot labels using PID 0121 or PID 8121 when a SET command is not accepted by the controller. For the command syntax, pls refer to the standard E1-20. Due to its structure, the command is a complex manufacturer specific command (CMSC), which must process several parameters.

Important: Slot labels are limited to 16 characters!

Important: The function must be enabled before writing slot labels is possible.

User Slot Label enable:

Parameter: <SLOT_LABEL_ON:Byte>
$00: User slotLabel active
$FF: User slotLabel off

GET Called without parameters
Return: 1 Byte: current setting

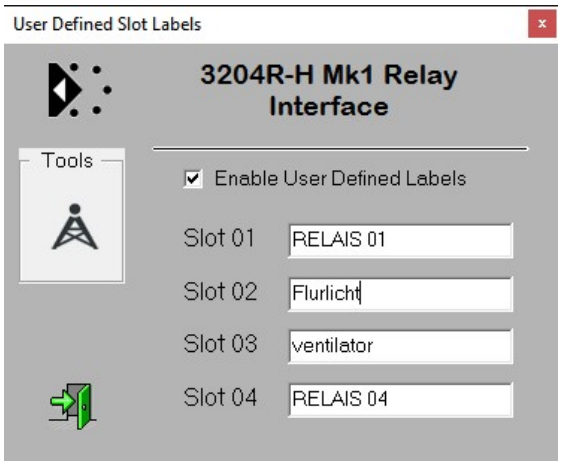
SET: Calling parameters:: 1 Byte
(SLOT_LABEL_ON: Byte)
Return: Status

After User Slot Labels have been enabled, the slot labels displayed may have changed immediately, since data are now read from the user memory.
Example: RGB decoder slot labels may change from „Fader 1“, „Fader 2“ and „Fader 3“ to „Red, Green, Blue“. These labels can be user-modified as „Fire Red“, „Grass Green“ or „SKy Blue“ if you like.

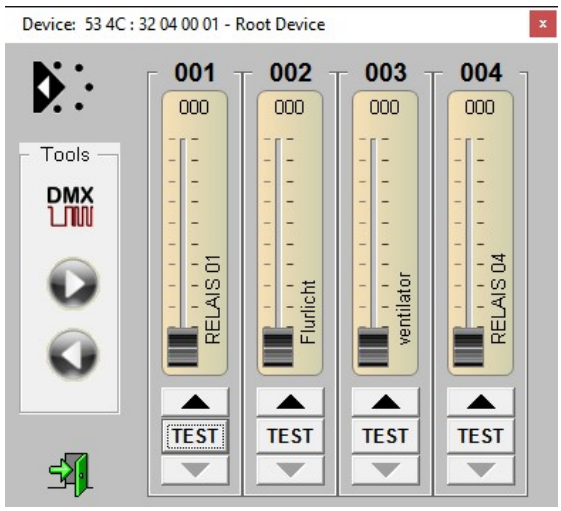
Writing slot labels::
Parameter: <SlotLabel Nr:Word>
<SlotLabel Text: 16 Bytes>
where:
SlotLabelNo: $0000 = Label 1
$0001 = Label 2
$0002 = Label 3
etc
SlotLabelText: ASCII Bytes

GET Calling parameters:: <SlotLabel Nr:Word>
Return: 18 Bytes: <SlotLabel Nr:Word>
<SlotLabel Text: 16 Bytes>

SET:	Calling parameters:: 18 Bytes <SlotLabel Nr:Word> <SlotLabel Text: 16 Bytes>
Return:	Status



The GET/SET controller makes it easy to change slot labels: once enabled, the pre-defined slot labels change to user-modified labels. Simply over-type labels to change contents.



Checking the faderpanel shows the new fader labelling immediately. Remember that fader labels are stored in the RDM responder, not in the RDM controller. Thus shutting down the controller or moving the responder to a different location or setup will keep the user-defined labels intact.

**= Due to RDM Standard E1-20 PID 0121 can only be read, but cannot be written. Our responders also accept SET-commands. If your controller does not accept SET commandsn-for PID 0121, use MSC PID 8121, where a SET access will be possible. Disadvantage: since standard controllers may only offer a generic mask for dataentry, you have to convert text into ASCII codes to write to the responder.*

PID 80FA DMX HOLD LEVEL PEGEL BEI SIGNALAUSFALL

If supported, this fuction allows to set a DMX HOLD LEVEL which is output in case of control signal missing. The level can be set in 8 bit or in 16 bit format. Only the upper n bits are considered, where n is the resolution specified in DEVICE INFO.

Calls: GET <param = none> (no parameter needed)
Return: <param=DMX HOLD LEVEL [1 byte if n<=8, else 2 bytes]>

SET <param=DMX HOLD LEVEL [1 byte if n<=8, else 2 bytes]>
Return: <param=none> (no parameter returned)

PID 8400 SENSOR DEFINITION SENSORTYPE ANPASSEN

This function changes the definition and parameters of the sensor. The format copies the sensor definition as described in RDM standard E1-20. Its purpose is the change the default definition for external sensors. NOTICE: internal sensors (system sensors) cannot be modified.

TIPP: Before trying to make a SET call, issue e GET command and study the result to familiarize yourself with the sensor define data.

Sensors are numerated from 00*hex* until nn , where at least sensor 00 is a system sensor (power supply). System sensors cannot be modified, SET attempts will be refused.

Function: GET / SET

Parameter:

GET: 1 Byte (Sensor-Number, $00 ... $nn)
Return: 46 Bytes Sensordata (see list)

SET: 46 Bytes Sensor data (see list below)
Return: <Status>

Example: Read Sensor 00. (GET) This may yield a data set like this:
00 01 02 01 00 78 01 2C 00 96 00 F5 02 20 50 6F 77 65 72 20 53 75 70 70 6C 79 20 56 20 6F 6C 74 73 20 20 20 20 20 20 20 20 20 20 20 20 20

This is a detailed alalysis of these data (example):

```
00          Sensor Number            ;byte      ;$00  = system sensor
01          Type                     ;byte      ;01=voltage sensor
02          Unit                     ;byte      ,02=volts
01          Prefix                   ;byte      ;01=prefix "dezi"
00 78       Range Minimum Value      ;word      ;0078 = 120dV = 12,0V
01 2C       Range Maximum Value      ;word      ;$012C = 300dV = 30,0V
00 96       Normal Minimum Value     ;word      ;$0096 = 150dV = 15,0V
00 F5       Normal Maximum Value     ;word      ;$00F5 = 245dV = 24,5V
02          Recorded Value Support   ;byte      ;$00= no, $02=yes
20          space                    ;byte      ; "space "
50 6F 77.... 32 chars text label               ;"Power Supply Volts          "
```

NOTICE:
Use these data to set the sensor functionality::

Sensor No.:	01...0F*hex*	(the number of sensors available in the responder can be queried by RDM)
Type:		00*hex*=temperature, 01=voltage, 02=current, 05=power, 09=volume, 1C=contact, 7F=other
Unit:		00=none, 01=degrees C, 02=DCV, 07=DCA, 0A=W
Prefix:		00=non, 01=dezi, 02=centi, 03=milli, 04=micro 11=deka, 12=hekto, 13=kilo, 14=mega *(all data hexadezimal)*

A full description of all parameters available can be found in the standards document ANSI E1.20-2010, which can be donloaded from: *https://webstore.ansi.org*

NOTICE:
The useable length of the text label is limited to just 31 characters, since a space will automatically be added as first character. This is necessary since all data must be stored in words. Unfortunately the standard does not take this into account.

PID 8403 OUTPUT CONFIGURATION AUSGANGS-TYPE FESTLEGEN

This function is available for responders with configurable outputs.

FUNCTION: GET / SET

Aufrufe:

GET:	Parameter:	keine
	Return:	n+2 bytes <control outputs available: word> <out1:byte> <out2:byte> <out3:byte> ..... <out*n*:byte> *n= Anzahl der verfügbaren Ausgänge*
SET:	Pararameter:	3 bytes <control output number: word> <out*x*:byte> *x= Nummer des zugehörigen Ausganges*
	Return:	<status>

Parameter:

OUTPUT CONFIGURATION:
$FF= Output deactivated
$00= Normal Port (TTL-Level: off/on)
$01= PWM output
$02= PWM inverted

Please refer to chapter „HARDWARE DECRIPTION“ in the responder manual. Output topologies and wireing information is given there.

PID 8423 DMX FOOTPRINT SLOTANZAHL FESTLEGEN

DMX FOOTPRINT determines the number of data slots allocated by the responder. The number of slots can differ for each personality

FUNCTION: GET / SET

GET:	Calling:	no parameters
	Return:	*n* bytes <slots<_footprint*1*:byte>....<slots<_footprint*n*:byte> *n= number of available personalities*
SET:	2 bytes:	<personality:byte> <slots:byte>
	Return:	<status>

Parameter:	Personality. byte	01...nn
	Slots: byte	00...nn

PID 8438 INPUT POLARITY EINGANGSPOLARITÄT ÄNDERN

This function sets the polarity of the switch sensor inputs.

Calls: GET <param = none> (no parameter needed)
Return: <param=Polarity [1 Byte]>

SET <param=Polarity [1Byte]>
Return: <param=none> (no parameter returned)

Polarity = $FF all normal polarity (standard mode)
Polarity = $00 all inverted polarity

From firmware version 1.1 onward, single switches can be inverted individually. Where available, use personality 3 „Test Mode“ to check for functionality and proper setting of the switch inputs. Switches should be set to show „L“ when disengaged (operational mode) and „H“ when engaged (error mode).
Changing the polarity of the center switch will result in changing the center point trigger flank.

RIGHT SWITCH: Bit 0 (normal: add value 1, inverted: add value 0)
CENTER SWITCH: Bit 1 (normal: add value 2, inverted: add value 0)
LEFT SWITCH: Bit 2 (normal: add value 4, inverted: add value 0)

Normal Mode: Bit set
Inverted Mode: Bit not set

Example: to invert the right and the left end sensor switch input, add 1+4 = 5.
Thus Polarity = $05

A001: RDM DECADES — ANZAHL DER AUSGEBEDEKADEN

This setting is intended for DALI devices and controls the output data range. The range and the associated characteristic are set using the RDM DECADES function.

Parameter:	<DECADES:Byte>	
	DECADES = 02*hex* (2*dez*):	1%...100%
	DECADES = 03*hex* (3*dez*):	0,1%...100%
GET:	no parameters	
Return:	1 Byte: current setting	
SET:	calling parameters: 1 Byte (DECADES: Byte)	
Return:	Status	

PID C000: SPEED SCALING STEPPER MOTOR GESCHWINDIGKEIT

The PID SPEED SCALING allows to apply a scaling factor (>1) to a stepper motor speed

Parameter:	<SPEED_SCALING:Byte> STEP WIDTH = $01...$FF
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (SPEED_SCALING: Byte) Return: Status

IMPORTANT NOTICE:
For security, the function SPEED SCALING may be locked (LOCK STATE = 02 or 03). If so, no data will be written but a message box "WRITE PROTECTED" will be displayed instead. Change the LOCK STATE to „00“ resp. „unlocked“ using the valid PIN.

PID C001: STEP WIDTH STEPPER MOTOR SCHRITTWEITE

The defaut setting is 1 stepper motor step per DMX step. To increase the step width, multiple motor steps can be assigned per DMX step (1...99*dec*). To do so, just call STEP WIDTH and enter the desired factor.

Parameter:	<STEP WIDTH:Byte> STEP WIDTH = $01...$FF
GET	Called without parameters Return: 1 Byte: derzeitige Einstellung
SET:	Calling parameters:: 1 Byte (STEP WIDTH: Byte) Return: Status

NOTICE:
For security, the function STEP WIDTH may be locked (LOCK STATE = 02 or 03). If so, no data will be written but a message box "WRITE PROTECTED" will be displayed instead. Change the LOCK STATE to „00“ resp. „unlocked“ using the valid PIN.

PID C002: PWM FACTOR STEPPER MOTOR PWM-FAKTOR

When a stepper motor does not turn, motor windings may sink higher currents since there is no induction due to missing movement. This could cause thermal damage within the motor. Setting the motor current to zero, the holding power will also be missing. The PWM FACTOR PID allows to define the holding current (0% [$00] to 99% [$FF]).

The function allows a minimum setting of 0% (0*dec*, 00*hex*, no hold) to 100% (255*dec*, FF*hex*, full hold current).

The factory preset is 60% (153*dec*, 99*hex*).

Parameter:	<PWM FACTOR:Byte> PWM FACTOR = $00...$FF
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (PWM_FACTOR: Byte) Return: Status

NOTICE:
For security, the function PWM FACTOR may be locked (LOCK STATE = 02 or 03). If so, no data will be written but a message box "WRITE PROTECTED" will be displayed instead. Change the LOCK STATE to „00“ resp. „unlocked“ using the valid PIN.

PID C003: LOWER LIMIT — UNTERE FAHRBEREICHS-GRENZE

This function allows to set the "lowest position“ („leftmost position“) in positioning mode. The function is only active in positioning mode.

Parameter:	<LIMIT_LO:Byte> LIMIT_LO = $00...$FF
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (LIMIT_LO: Byte) Return: Status

Limit_Lo = $00...$FE sets the parameter value as leftmost
Limit_Lo = $FF sets the actual position as leftmost

NOTICE:
For security, the function LOWER LIMIT may be locked (LOCK STATE = 02 or 03). If so, no data will be written but a message box "WRITE PROTECTED" will be displayed instead. Change the LOCK STATE to „00“ resp. „unlocked“ using the valid PIN.

PID C004: UPPER LIMIT — OBERE FAHRBEREICHS-GRENZE

This function allows to set the "highest position“ („rightmost position“) in positioning mode. The function is only active in positioning mode.

Parameter:	<LIMIT_HI:Byte> LIMIT_HI = $00...$FF
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (LIMIT_HI: Byte) Return: Status

Limit_Lo = $00...$FE sets the parameter value as rightmost
Limit_Lo = $FF sets the actual position as rightmost

NOTICE:
For security, the function UPPER LIMIT may be locked (LOCK STATE = 02 or 03). If so, no data will be written but a message box "WRITE PROTECTED" will be displayed instead. Change the LOCK STATE to „00“ resp. „unlocked“ using the valid PIN.

PID C005: END SWITCH POLARITY — POLUNG ENDSCHALTER

Function C005 allows to invert the polarity of the end switches-closing contacts (n.o.) can be replaced with openers (n.c.). This simplifies the connection to existing hardware.

Parameter:	<POLARITY:Byte>
	POLARITY = $FF: normal
	POLARITY = $00: inverted
GET	Called without parameters
	Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte
	(POLARITY: Byte)
	Return: Status

PID C006: POS DECEL RANGE — POSITIONIERUNGS-BREMSZONE

This PID allows to set the decelaration zone for electric stepper motors. The decelaration zone should be set for full (maximum) speed. Within the decelaration zone the speed decreases linearly to minimum speed.
The deceleration zone can be set from 15 to about 2000 steps..

Function Calls:
GET <param = none> (no parameter required))
Return: <param=Decel_Range [Byte]>

SET <param=Decel_Range [Byte]>
Return: <param=none> (no return parameters)

Parameter:
Decel_Range = $01...$04
sets the number of stpes as deceleration range:

$01 =	255 Steps
$02 =	512 Steps
$03=	1024 Steps
$04 =	2048 Steps

Decel_Range = $81...$85
sets the number of stpes as deceleration range:

$81 =	255 Steps
$82 =	127 Steps
$83=	63 Steps
$84 =	31 Steps
$85 =	15 Steps

FUNKTION C007 POS DEAD ZONE TOTZONE

This function defines the dead zone (steps) around zero position.

Function Calls:

GET <param = none> (no parameter required)
Return: <param=DeadZone [Byte]>

SET <param=DeadZone [Byte]>
Return: <param=none>
(no Return:parameter)

Parameter: DeadZone = $00...$FF
sets the parameter value as dead zone

NOTICE:
When setting a Pos_Dead_Zone please deactivate positioning accuraca (if available). Both functions are overlapping. When defining a dead zone, positioning accuracy should be set to $00.

PID C008: INIT SPEED INITALISIERUNGS-GESCHWINDIGKEIT

This function sets the speed during initialization (searching zero position). This is a fixed speed. Please note that the initalization routine only starts when allowed by the CONTROL slot (usually DMX slot #1).

Parameter:	<INIT_SPEED:Byte> INIT_SPEED = $00...$FF
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (INIT_SPEED: Byte) Return: Status

Init_Speed = $00...$FF sets the parameter value as initialization speed.

Init_Speed = $00...$FE sets the parameter value as init speed
Init_Speed = $FF sets the actual speed as init speed

TIPP:
Operate the device in „endless mode“ to find the preferred speed. Then switch to „positioning mode“ and save the current speed fader value.

ATTENTION: If mechanical endpoints exist, make sure to stop before reaching the maximum position.

PID C009: ACCURACY BITS POSITIONIERUNGSGENAUIGKEIT (BITS)

This function sets the accuracy when reaching the set position..

This function is only active in positioning mode. The positioning accuracy is defined by a bitmask, which sets the number of steps for „position reached“. The default setting is $03 (8 steps).

Parameter: <ACCURACY_BITS:Byte>
ACCURACY_BITS = $00...$07

GET Called without parameters
Return: 1 Byte: current setting

SET: Calling parameters:: 1 Byte
(ACCURACY_BITS: Byte)
Return: Status

Parameter:

Accuracy_Bits = $00...$07
$00 = full resolution, 0 steps
$01 = dead zone 2 Steps
$02 = dead zone 4 Steps
$03 = dead zone 8 Steps
$04 = dead zone 16 Steps
....etc. until
$07 = dead zone 128 Steps

PID C00A: AUTO_INIT ENABLED AUTOMATISCHE INITIALISIERUNG

This function enables automatic initialization in positioning mode.

Function Calls:

GET <param = none> (no prarameter required)
Return: <param=Auto_Init [Byte]>

SET <param=Auto_Init [Byte]>
Return: <Status>

Parameters:

Auto_Init = $00	Initalization off
Auto_Init = $FF	Initialization on

NOTICE: *Please note that a zero point sensor must be present for automatic initialization. If during initialization no valid zero point can be found, the system stops after three unsuccessful attempts and displays a error message. This requires manual intervention.*

SAFETY NOTICE:
If the automatic initilization has been aborted with error message, the system must only be restartet with an operator present to be able to take control manually.

PID C010 MID POINT OFFSET

This function sets the position offset for the center point sensor. Usually, a inductive sensor is being used, which is detected over a certain span when the nozzle is moving.

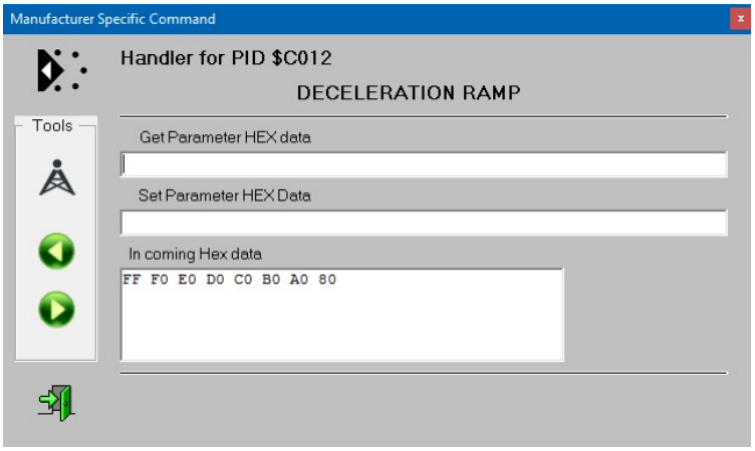
The zero point detector is flank triggered only when the nozzle is moving upward. To compensate for the detection range, a MID POINT OFFSET value can be defined using this function. The data entry range is 000...255 (00*hex*...FF*hex*), with the default set to center position 128 (80*hex*) at the factory. Normally, this will work out fine for all standard applications.

Calls:	GET	<param = none> (no parameter needed)
	Return:	<param=MidPointOffset [1 Byte]>
	SET	<param=MidPointOffset [1 Byte]>
	Return:	<param=none> (no parameter returned)

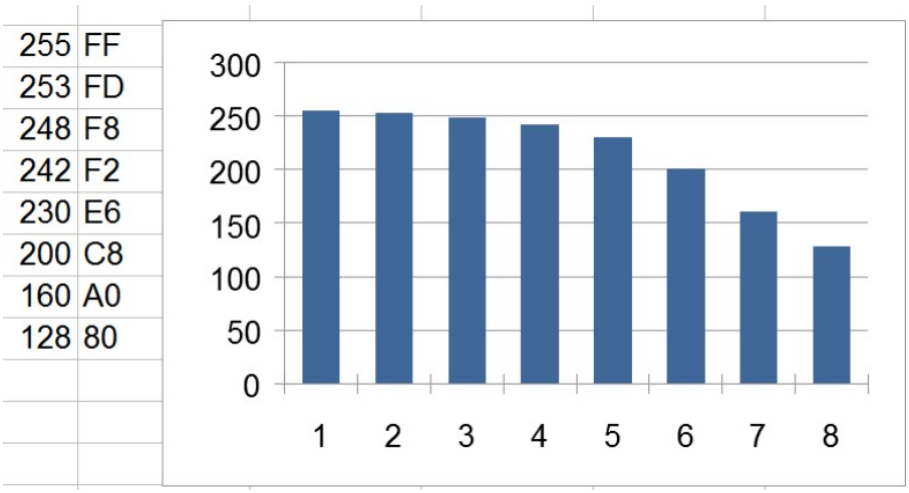
FUNCTION C011 ACCELERATION RAMP
FUNCTION C012 DECELERATION RAMP

This function sets the ramp for acceleration or deceleration of the stepper motor. 8 values must be given to define the acceleration curve, with full speed = $FF and zero speed = $00. See example for more details.

Calls: GET <param = none> (no parameter needed)
Return: <param=Ramp [8 Bytes]>



GET Mask



Ramp design

SET <param=Ramp [8Bytes]>
Return: <param=none> (no parameter returned)

Ramp = $00...$FF the value will be taken as speed factor

PID C013 MOVING RANGE

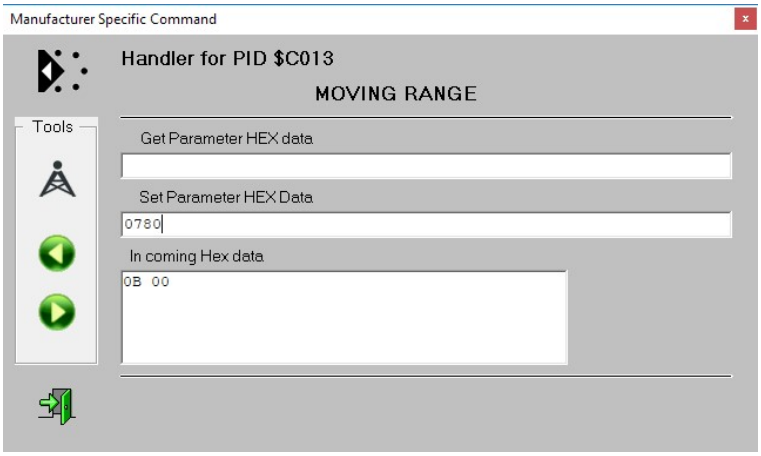
This function sets the total moving range of the nozzle. The standard resolution (when setting the stepper motor driver as stated in the addendum) is 32 steps per degree. This will result in the settings as per table 1:

Angle [°]	Steps	Hexadecimal
30	960	03C0
45	1440	05A0
60	1920	0780
90	2880	0B40
120	3840	0F00
150	4800	12C0
180	5760	1680

Table 1 Angle Step setting

Calls: GET <param = none> (no parameter needed)
Return: <param=StepSetting [2 Bytes]>

SET <param=StepSetting [2Bytes]>
Return: <param=none> (no parameter returned)



Changing from >90° to 60° (+/-30°) moving range

Adjusting the moving range will automatically scale the position faders to cover the full range.

PID C0C0: INTERNAL PATCHING Zuweisung der DMX Datenquelle für das Relais

This function is used with DMX relay modules.

Usually, relay 1 will be activated by DMX slot 1, relay 2 by DMX slot 2 etc. This function introduces a patching between the DMX data source and the relay number, which makes it possible to trigger any relay from any data source. Also, multiple relays can be triggered from the same source.

Parameter:	<relay number:word> <Source:byte>		
GET	Called without parameters		
	Return:	n Bytes (n= number of relays present)	
		Byte 1...n: source / current setting	
GET	Calling parameters: word (relay number [16 Bit])		
	Return:	1 Byte	
		Byte 1: source / current setting	
SET:	Calling parameters:: 3 Bytes		
		(relay number [word], source [byte]	
	Return:	Status	
Parameters:			
	Relays:	0001*hex*	Relay 1
		0002*hex*	Relay 2
		....	
		0006*hex*	Relay 6
		FFFF*hex*	all relays
	Source:	01*hex*	DMX slot 1
		02*hex*	DMX slot 2
		....	
		06*hex*	DMX slot 6

PID C0E0: DMX DATA POLARITY Flanke für die Auslösung des monostabilen Relais

The function DMX DATA POLARITY allows to invert received DMX data. This is a useful feture to invert the triggering of monostable pulse, e.g. change from triggering by the positive flank to the negative flank. The command syntex resembles setting monostable mode.

Parameter:	<relay number:word> <Polarity:byte>		
	Polarity:	00*hex*	inverted
		FF*hex*	normal
GET	Called without parameters Return: n Bytes (n= number of relays) Byte 1...n: <current setting>		
GET	Calling parameters:: 2 Bytes <relay number:word> Return: 1 Byte Byte 1: <current setting>		
SET:	Calling parameters:: 3 Bytes <relay number:word> <Polarity:byte> Return: <Status>		

Relay functions @ FULL COMMAND LIST = 0

NOTICE:
The settings of PIDs C0C0, C0E0 and C0F0 have no effect as long as the responder operates with short command list. No configuration entries can be made.

Relay functions in Detector mode

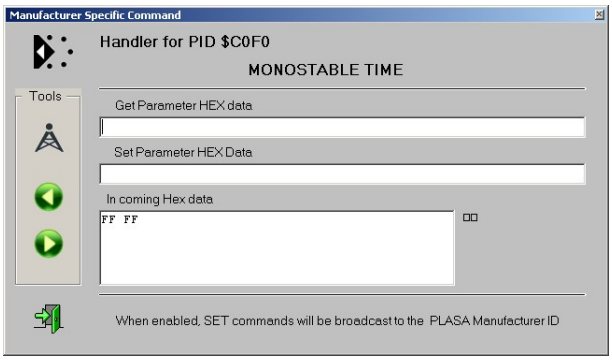
NOTICE:
PIDs C0C0, C0E0 und C0F0 are accessible in Detector mode, but have no effect since no DMX data are received. Since no DMX slots have been assigned, there is no signal source available.

PID C0F0: MONOSTABLE TIME Setzen der monostabilen Impulsdauer

The relays of DMX relay modules 3202R-H, 3204R-H or 3206R-H operate as bistable relays, that is, as long as the relevant DMX data are above the upper threshold level, the output relay is engaged.
The function can be set to monostable mode (inpulse output) with user definable impulse duration using PID C0F0.

Parameter: <Relay number> <Monotime>
wobei: Relay number = 0001*hex*: relay 1
Relay number = 0002*hex*: relay 2
....
Relay number = 0006*hex*: relay 6
Relay number = FFFF*hex*: all relays

Monotime: 01...3F*hex* 25ms...1,5s in 25ms steps
41...7F*hex* 0,25s...15,75s in 250ms steps
FF*hex*: bistabile Mode



GET command using the GET/SET Controller. Data for all existing relays are displayed (shown for 2ch model 3202R-H)

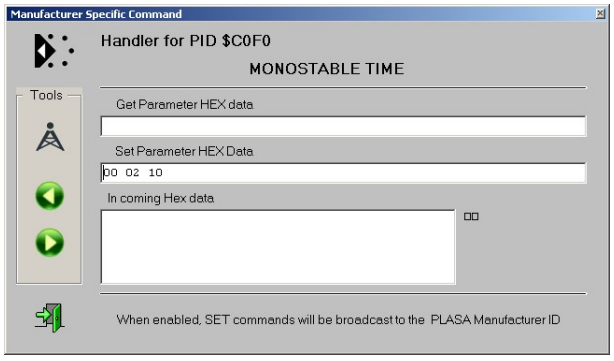
Press GET to retrieve the actual setting, either all relays or one individual relay:

Parameter: <Relay number:word> <Monostable Time:byte>

GET Called without parameters
Return: n Bytes (n= number of relays present)
Byte 1...n: current setting

GET Calling parameters:: Word (Relay number, 16 Bit)
Return: 1 Byte : current setting

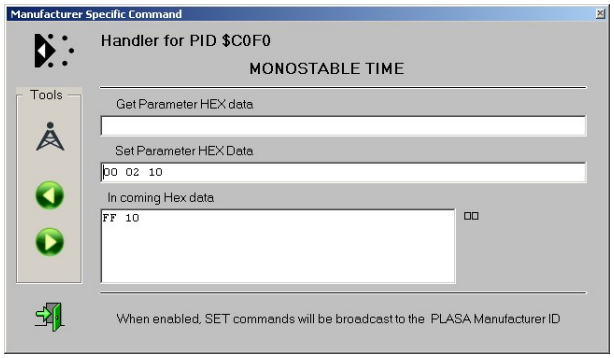
SET: Calling parameters:: 3 Bytes
<Relay number:word> <Monostable Time:byte>
Return: Status



To set relay 2 to monostable mode using a pulse duration of 0,4 sec (400ms) please enter:

0002 10 *hex*
because: 10*hex* = 16*dec*, 16*25ms = 400ms

Press the SET button to execute.



Checking:
When issuing a new GET command, the actual settings will be displayed. Relay 1 is bistable, relay 2 is in monostable mode with 0,4s pulse time.

NOTICE: when set to monostable mode, relays will stay engaged for the set pulse duration even if the DMX control is reset to zero before.

Shortening the pulse duration

You can shorten the pulse duration (pulse ends when driving DMX signal falls back below lower threshold) by adding 80(hex) / 128(dec) to the settings. This can be done for each relay separately. The settings would read:

Monotime:	*81...BF(hex)*	*25ms...1,5s in 25ms steps*
	C1...FE (hex)	*0,25s...15,5s in 250ms steps*
	FF(hex)	*bistable Modus*

PID C0F1: EXCLUSIVE MODE — Relais schalten gegenseitig verriegelnd

EXCLUSIVE MODE means: only one of two relays can be activated momentarily. This locking is done electronically. This is the truth table:

CH1	CH2	RELAY 1	RELAY 2
OFF	OFF	OFF	OFF
ON	OFF	ON	OFF
OFF	ON	OFF	ON
ON	ON	no change	no change

Parameter:	<MODE:Byte>	
	MODE = FF*hex* (255*dec*):	Exclusive ON
	MODE = 00*hex* (0*dec*):	Exclusive OFF

GET	Called without parameters
Return:	1 Byte: current setting
SET:	Calling parameters:: 1 Byte (MODE: Byte)
Return:	Status

When using modules with multiple relays, these pairs are switched to exclusive mode:
- Relay 1 and Relay 2
- Relay 3 and Relay 4
- Relay 5 and Relay 6

...etc...

PID C0F2: FAST MODE **Hohe Schaltgeschwindigkeit setzen**

This mode suppresses the DMX glitch detector and speeds up the signal processing. Each DMX data packet will be routed directly to the outputs, no evaluation of DMX values exists. This is the fastest mode possible.

Parameter:	<MODE:Byte>	
	MODE = FF*hex* (255*dec*):	Fast Mode ON
	MODE = 00*hex* (0*dec*):	Fast Mode OFF
GET	Called without parameters Return: 1 Byte: <current setting>	
SET:	Calling parameters:: 1 Byte <MODE: Byte> Return: <Status>	

PID C0F3: SAFETY MODE / TRIGGER MODE Auf Trigger-Modus umschalten

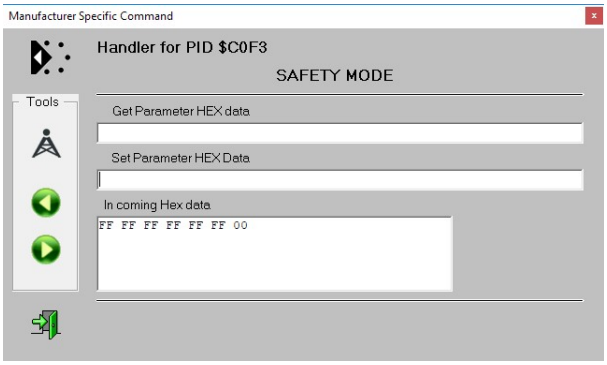
This mode configures individual relays (or all relays) to trigger mode. In trigger mode, the relay will only be engaged if the trigger slot has been set to a special trigger value.

Parameter: <TRIGGER_MODE:Byte>
MODE = FF*hex* (255*dec)*: Trigger_Mode ON
MODE = 00*hex* (0*dec*): Trigger_Mode OFF

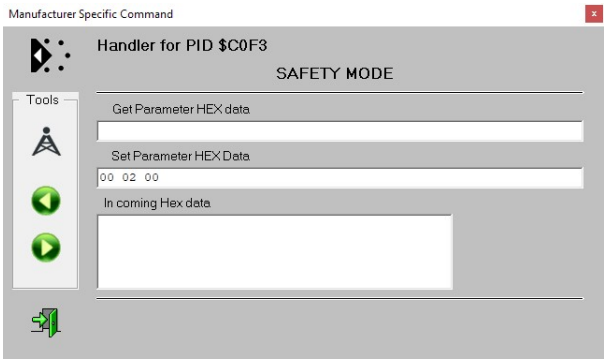
GET — Called without parameters
Return: n Bytes (n= number of relays)
Byte 1...n: <current setting>

GET — Calling parameters:: 2 Bytes <relay number, word>
Return: 1 Byte : <current setting>

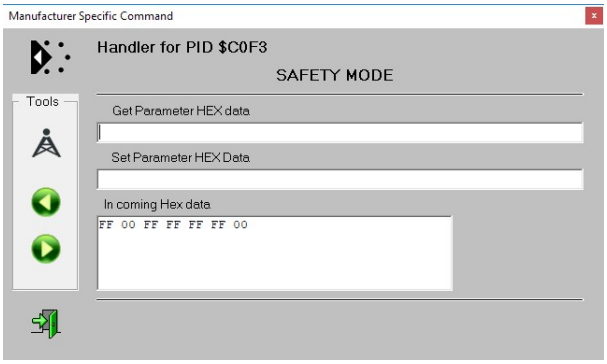
SET: — Calling parameters: 3 Bytes
<relay number: word; Trigger mode: byte>
Return: <Status>



GET without parameters shows all relays (1-6) set to trigger mode. Slot7 is the trigger slot.



SET: relay 2 is no longer controlled by the trigger slot



Checking with GET, no parameters: all relays except relay 2 are still tied to the trigger slot (7).

PID C0F4: SECURITY DELAY **Delay time for sensor evaluation**

The sensor delay time can be activated to delay sensor related action. Thus it will be possible to trigger a relay and to disable the output again as soon as the sensor is triggered. This allows to create applications like automatic filling stations (relay activated and de-activated as soon as sensor reports filling level) or flame detectors (process started but halted if sensor not in range after delay time).

Delay time is defined in increments of 25ms. Thus the total time can be adjusted from 0...6,3 seconds.

Parameter:	<Slot number> <DELAY> [Byte]	
where:	Slot number = 0001*hex*:	Slot no. 1
	Slot number = 0002*hex*:	Slot no. 2
	Slot number = FFFF*hex*:	all

DELAY = 00*hex* (000*dec*) ... FF*hex* (255*dec*)
(timecount * 25ms)

Example: The safety supervision for relay 2 shall be engaged after 2 seconds:

SET PID C0F4: 00 02 50

Calculation: 2 seconds is 80x 25ms, since 80(dec) is 50hex ($50). Most RDM controllers require values to be entered in hex format.

PID C0F5: DELAY TIME — DELAY-ZEIT EINSTELLEN

This mode adds a signal delay. The delay is specified in 25ms steps.

Parameter:	<SIGNAL_DELAY:Byte> SIGNAL_DELAY = $01...$FE
GET	Called without parameters
Return:	n Bytes (n= number of slots) Byte 1...n: <current setting>
GET	Calling parameters: 2 Bytes (slot number, word)
Return:	1 Byte : <current setting>
SET:	Calling parameters:: 3 Bytes (Slot number: word; Signal_Delay: byte>
Return:	<Status>

PID C0F6: SAFETY WINDOW / TRIGGER WINDOW FENSTERBREITE FÜR AUSLÖSUNG

This function defines the trigger window for the trigger slot. A trigger event will only occur when the trigger level is within (including) the lower trigger level and the upper trigger level.

Parameter:	<TRIGLEVEL_LOW: Byte> <TRIGLEVEL_HIGH: Byte> Trigger level can range from 00*hex* to FF*hex*. The higher level must not be lower than the lower level. We recommend to avoid level FF*hex* since a broken DMX line may (false) be interpreted as FF*hex* and then trigger events.
GET	Called without parameters Return: 2 Bytes <Triglevel_Low, Triglevel_High>
SET:	Calling parameters: 2 Bytes <Triglevel_Low: byte , Triglevel_High: byte> Return: <Status>

PID DC01: OUTPUT CURRENT AUSGANGSSTROM FESTLEGEN

This PID sets the output current for LED drivers. The setting is in mA. Depending on the hradware used, setting may be continously or is steps. When steps are used, the selected (or the next lower) step will be activated.

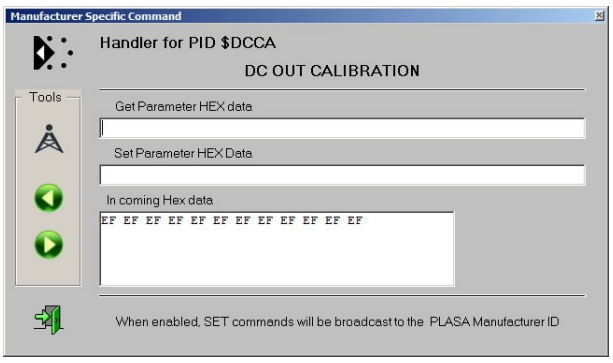
Parameter:	<OUTCURRENT: word> word: current in [mA]
GET	Called without parameters Return: <OUTCURRENT: word>
SET:	Calling parameters:: 2 Bytes (word) OUTCURRENT: word Return: <Status>

Typical values for current settings:

350 mA:	015E*hex*, 350*dec*
500 mA:	01F4*hex*, 500*dec*
600 mA:	0258*hex*, 600*dec*
700 mA:	02BC*hex*, 700*dec*

PID DCCA: OUTPUT CALIBRATION — AUSGANGS-KALIBRIERUNG

This menu allows to set n individual calibration values for outputs 1...n. Data entry uses hexadecimal number format.



(shown: JESE GET/SET controller, setting for 12-channel DMX Demux 3012C-EP)

GET entry values:

none	displays list of calibration factors
00 xx	displays calibration factor for slot xx (xx=01...nn)

SET entry values:

00 xx yy	Sets calibration factor yy for slot xx (yy=00...FF; xx=01...nn)
FF FF yy	Sets same calibration factor yy for all outputs

PID E001: LOWER TEMP TRIP	**UNTERER TEMPERATUR-REGELPUNKT**
PID E002: UPPER TEMP TRIP	**OBERER TEMPERATUR-REGELPUNKT**
PID E003: ALARM TEMP TRIP	**ALARMTEMPERATUR-SCHALTPUNKT**
PID E004: TEMP SAMPLE TIME	**TEMPERATUR-SAMPLEZEIT**
PID E005: TEMP DROP LEVEL	**TEMPERATUR-ABFALLPEGEL**

Diese Funktionen dienen der Einstellung des Temperatur-Managements. Sie sollten vom Anwender nicht verstellt werden. Detailliertere Hinweise entnehmen Sie bitte dem separat erhältlichen Temperaturmanagement-Manual.

PID E001: LOWER TEMP TRIP UNTERER TEMPERATUR-REGELPUNKT

This setting defines the lower temperature management threshold. Outputs will ramp up slowly as soon as the measured temperature falls below the preset temperature. The default setting is 40C. That means: as soon as the LED temperature falls below 40C, the outputs will slowly ramp up to full control range.

The LOWER TEMP TRIP point must be entered in 0,5C steps. That is, to set a lower trip point of 40C, a value of 80 (50*hex*) must be entered. To disable temperature management, enter a value of 0 (00*hex*). Please note, that many RDM editors require values to be entered in hexadecimal format. Please refer to the manual of your preferred RDM editor.

Parameter:	<LOWER TEMP:Byte> LOWER TEMP = $00...$FF, $00=OFF
GET:	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (LOWER TEMP: Byte) Return: <Status>

PID E002: UPPER TEMP TRIP OBERER TEMPERATUR-REGELPUNKT

This setting defines the upper temperature limit. As soon as the set limit is being exceeded, the outputs will be decreased slowly. The default setting is 50C.
The degree of power reduction is derived from the transgression, about 10%/degree. Minimum output level at all times is 20%.

When setting the UPPER TEMP TRIP point, please note that the UPPER TEMP TRIP point must be entered in 0,5C steps. That is, to set a lower trip point of 50C, a value of 100 (64hex) must be entered. To disable temperature management, enter a value of 0 (00hex). Please note, that many RDM editors require values to be entered in hexadecimal format. Please refer to the manual of your preferred RDM editor.

Parameter:	<UPPER TEMP:Byte> UPPER TEMP = $00...$FF, $00=OFF
GET:	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (UPPER TEMP: Byte) Return: <Status>

PID E003: ALARM TEMP TRIP ALARMTEMPERATUR-SCHALTPUNKT

This setting defines the absolute maximum temparature limit. When exceeding the absolute maximum temparature setting, the outputs will immediately be limited to 20% max. output, and a alarm message will be generated. The LED signalling will change to synchronous red-green blinking.

Default setting is 80C. The ALARM TEMP TRIP temperature must be entered in 0,5C steps, thus setting the alarm temp to 80C requires a entry of data value 160 (A0 hex). To disable temperature management, enter a value of 0 (00hex). Please note, that many RDM editors require values to be entered in hexadecimal format. Please refer to the manual of your preferred RDM editor.

Parameter:	<ALARM TEMP:Byte> ALARM TEMP = $00...$FF, $00=OFF
GET:	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (ALARM TEMP: Byte) Return: <Status>

PID E004: TEMP SAMPLE TIME TEMPERATUR-SAMPLEZEIT

The sample interval time between two temperature measurements. The interval is set to 12 seconds default and is optimized for high power LED spots. This ensures a smooth temperature control profile.

ATTENTION: extreme settings may result in thermal oscillations (setting too low) or temperature overshoot, which may burn the LEDs.

The temperature step width is 0.5°C per sample interval.

Parameter:	<SAMPLE TIME:Byte> SAMPLE TIME = $00...$FF, $00=OFF
GET:	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (SAMPLE TIME: Byte) Return: <Status>

PID E005: TEMP DROP LEVEL TEMPERATUR-ABFALLPEGEL

The mimimum level can be set using the TEMP DROP LEVEL command, and is set to 50% (80 hex) default. The scaling matches the DMX value range 0...255 (00 hex to FF hex). We recommend to keep the drop level below 80% (204 dec, CC hex) to ensure enough headroom for smooth temperature regulation. Also please keep in mind that the level setting depends on the curve setting (e.g. linear or logarithmic). The Temp Drop level always refers to the DMX input level, not to the output!

Parameter:	<TEMP DROP:Byte> TEMP DROP = $00...$FF
GET:	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (TEMP DROP: Byte) Return: <Status>

PID E00E: TEMPSENSE ENABLE SENSOR-FREISCHALTUNG

Activation of individual temperature sensors. Up to 16 sensors can be managed (1 bit per sensor, total: 2 bytes).

The TEMP SENSE ENABLE function allows die enable/disable individual sensors as needed.

Default:
FF FF*hex* resp. 255 255*dec* resp. 1111 1111 1111 1111*Bin*

Please refer to the product manual for the individual assignment of temperature sensors.

PID FF08: CONFIG ACCESS. ZUGRIFF AUF KONFGURATION

The PID „CONFIG ACCESS“ starts a automatic down counter which locks all access to configuration parameters as soon as countdown to zero is complete. The timer decreases 1 count per minute, thus times from 1 thru 240 minutes can be set. Additionalle, the counter can be disabled for continuous access.

Parameter:	<Access_Time:Byte> ACCESS_TIME = $01...$FA
GET	Called without parameters Return: 1 Byte: current setting
SET:	Calling parameters:: 1 Byte (Access_Time: Byte) Return: <Status>

HINWEIS: The function CONFIG ACCESS completely blocks all EEPROM access, including DMX startaddress and personality. This applies to both, parameter changes by RDM or lo-callly by a start address board. Thus the function CONFIG ACCESS SET is a mighty tool to automatically protect your responder from unexpected parameter changes- no matter what the source might be.

Parameter: Access_Time:

00*hex* / 0*dec*:	sofortige Verriegelung
01*hex* / 1*dec.* bis F0*hex* / 240*dec*:	Access time in minutes
FA*hex* / 250*dec*:	permanent access

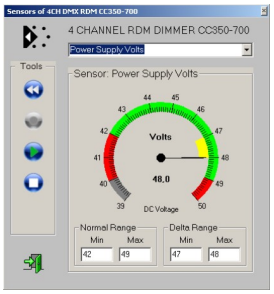
Default: FA*hex*

PID FF7F: RDM TESTAUSGABE / RDM DIAGNOSTIC INFO

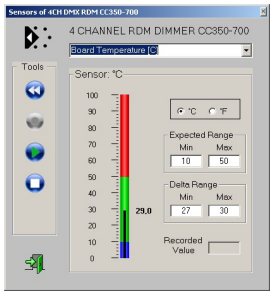
These functions are used for factory setup only.
No user input expected.

Sensors

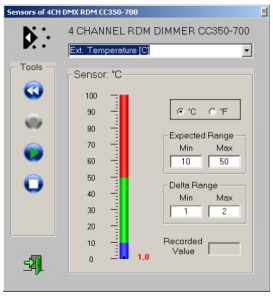
Many responders consist of one or more sensors to measure voltage, temperature, pressure or more. Here are some examples for sensors, which can be queried using DMX RDM:



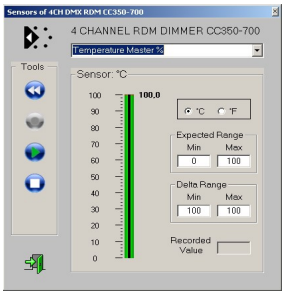
Sensor 1
Power supply



Sensor 2
Elektronics temperature



Sensor 3
LED temperature



Sensor 4
Temperature Master %

The GET/SET Controller automatically assigns a suitable mask to the relevant sensor. Thus all sensor data are easily readable.

Some sensors register variations over time in a special mask, which can be reset using the „<<“ button. To restart registration, press „>“ .

The mask and temperature limits for LED temperature display can be changed by the user using the temperature management PIDs. Changes will be updated with next discovery of the responder.

More RDM Info

For more info regarding DMX RDM, RDM settings and specific RDM commands pls refer to our website at ***www.soundlight.eu/rdm***.

Anhang

PID	NAME	LOCK1	LOCK2
	E1-20 and E1-37 commands		
0015	COMMS STATUS	-	X
0082	DEVICE_LABEL	-	-
0090	FACTORY_DEFAULTS	-	-
00E0	PERSONALITY	X	-
00F0	START_ADDRESS	-	-
0121	SLOT_LABELS	-	-
0140	BLOCK_ADDRESS	-	-
0141	FAIL_MODE	X	-
0201	SENSOR_RESET	-	-
0341	MIN_LEVEL	-	X
0342	MAX_LEVEL	-	X
0343	CURVE	-	X
0345	OUTPUT_RESPONSE	-	X
0347	MODULATION_FREQUENCY	-	X
0603	REAL_TIME_CLOCK	-	-
0640	PIN	X	X
0641	SET_LOCK_STATE	-	-
1001	RESET	-	-
1010	IDENTIFY	-	-
1040	IDENTIFY_MODE	-	-
	DMX Resolution		
80E2	16BIT_MODE	-	X
	DMX Slot Count		
80C1	SET_SLOTCOUNT	-	X
	DMX HOLD Mode		
80F1	DMX_HOLD	X	-
80F2	MASTER_HOLD	X	-
80F3	MASTER_CHANNEL	-	X
80F4	MASTER_TABLE	-	X
80F5	MASTER_CHANNEL	-	X
	Slot Labeling		
8121	SLOT_LABELS	-	X
	E1-20 and E1-37 commands		
8341	MIN_MAX_MODE	-	-
	Sensor Configuration		
8400	SENSOR_DEFINITION_DATA	X	X
8401	SENSOR_LIMITS	X	X
	Slot Configuration		
8403	OUTPUT_CONFIGURATION	-	X
8433	DMX_FOOTPRINT	-	X
	TimerSetup		
9002	TIMEBASE	-	X
9003	TRIGGER_LEVEL	-	X
9004	TRIGGER_COUNT	-	X
	MotorDriver		
C001	STEP_WIDTH	-	X
C002	PWM_FACTOR	-	X
C003	LOWER_LIMIT	-	X
C004	UPPER_LIMIT	-	X
C005	ENDSWITCH_POLARITY	-	X
	Relay Properties		
C0C0	PATCHING	-	X
C0E0	POLARITY	-	X
C0F0	MONOSTABLE_TIME	-	X
C0F1	EXCLUSIVE_MODE	-	X
C0F3	SAFETY_MODE	-	X
C0F4	SAFETY_DELAY	-	X
C0F5	DELAY	-	X

	Output Configuration		
DC01	OUTPUT_CURRENT	X	X
DC0E	DC_OFFSET	X	X
DC0F	OUTPUT_OFFSET	X	X
DC10	DC_SUPPRESS	X	X
DC1F	16BIT_OFFSET	X	X
DCCA	DC_CALIBRATION	-	X
DCCD	USER_CURVE	-	X
DCCE	BENDEC_CURVE	-	X
	Temperature Management		
E001	LOWER_TRIP_POINT	-	X
E002	UPPER_TRIP_POINT	-	X
E003	ALARM_TRIP_POINT	-	X
E004	SAMPLE_TIME	-	X
E005	TEMP_DROPLEVEL	-	X
E00E	TEMP_ENABLE	-	X
	Device Setup		
FF01	FACTORY_SETUP	-	-
FF02	CALIBRATION	-	X
	SubDevices		
FF0E	SUBDEVICE_STARTADDRESS	-	X
FF0F	SUBDEVICE_ENABLE	X	X

GET/SET CONTROLLER

Wir bieten Ihnen mit dem USBRDM-TRI ein Interface an, das komplett mit einer RDM-Controller-Applikation kommt und bestens geeignet ist, um alle unsere DMX RDM Interfaces (siehe: RDM Interfaces) zu verwalten. Mit dem Interface erhalten Sie eine Installations-CD mit Gerätetreibern und die RDM Controllersoftware "GET/SET" für Windows 7,8, und Windows 10. Das Interface und die Software kann auf mehreren Computern installiert werden. Es wird nur dann aktiviert, wenn es angesteckt ist.

Die Besonderheit des USBRDM-TRI ist die interne Signalverbeitung, die die RDM Kommunikation entlastet und beschleunigt. Laufende Updates der Controller-Firmware und der Applikations-Software sind enthalten; neue Versionen können jederzeit einfach von der Hersteller-Website gedownloaded werden. Der benötigte Update-Link ist in der Controller- Software enthalten.

Das USB-RDM TRI wird mit der GET/SET Controller Software gebundlet ausgeliefert. Da das Interface so wohl DMX empfangen als auch DMX senden kann, kann es einfach in eine DMX Leitung *eingeschleift* werden. Jede vorhandene DMX Steuerung kann so einfachst um volle RDM Funktionalität erweitert werden. Einfacher geht es nun wirklich nicht!

Mehr Infos auf:
www.soundlight.eu/produkte/usbrdm-tri2