DKM API

DKM FX AND DKM FXC SERIES APPLICATION PROGRAMMING INTERFACE

BLACK BOX®

Introduction

This manual contains important safety instructions as well as instructions for setting up the product and operating it. Please read the general safety instructions (see chapter 1, page 8) and additional notices in the respective chapters. Read carefully through the User Manual before you switch on the product.

Trademarks and Trade Names

All trademark and trade names mentioned in this document are acknowledged to be the property of their respective owners.

Validity of this Manual

This manual applies to all DKM FX and DKM FXC matrices of the series. Please note the change log for this manual in Chapter 11, page 125).

The manufacturer reserves the right to change specifications, functions, or circuitry of the series described here without notice. Information in this manual can be changed, expanded, or deleted without notice. You can find the current version of the manual on our website.

Copyright

Table of Contents

1 Important Information

1.1 Symbols for Warnings and Helpful Information

The meaning of the symbols used for warnings and helpful information in this manual is described below:

NOTICE
NOTICE identifies information, if not observed, that endangers the Functionality of your device or the security of your data.

This symbol indicates instructions for procedures recommended by the manufacturer to use the device effectively.

This symbol indicates information about special features on the device or when using device and Function variants.

1.2 Terms and Spellings

Uniform terms are used in this manual for better readability or easier assignment.

The following terms are used for products and descriptions:

Term	Description
Matrix	DKM FX and DKM FXC enterprise/flex/compact
Tera Tool	Management software
Source	Computer, graphics card (USB, video, audio, data)
Sink	Console (monitor, keyboard, mouse, video, audio, data)
CPU Unit	Encoder to connect to the source
CON Unit	Decoder to connect at the console peripherals
EXT Unit	Logical object for representing a CPU or CON Unit in the matrix
CPU Device	Logical object for switching EXT Units of CPU Units via matrix
CON Device	Logical object for switching EXT Units of CON Units via matrix

The following formats are used for keyboard commands:

Keyboard command	Description
key	Key on the keyboard
key + key	Press keys simultaneously.
key, key	Press keys successively.
2x key	Press key quickly, twice in a row (like a mouse double-click).
Number/number on the keyboard	Numeric key at the top end of the alphanumeric keyboard usually used for described operations
Number on the numerical pad	Numeric key on the numeric pad. If the use of the numeric pad is required, it is explicitly stated in the instructions.

The following formats are used for descriptions, such as descriptions of editing files or updating firmware:

Keyboard command	Description
`Config.txt`	For instance, file name.
`#CFG`	For instance, file content.

The following formats are used for software descriptions:

Spelling	Description
Bold print	Description of terms that are used in the device firmware or the management software
Bold print > Bold print	Selection of a menu item in the working area of OSD, such as **Configuration > System**

Mouse button	Description
Left mouse button	Primary mouse button* (default in most operating systems)
Right mouse button	Secondary mouse button*
* Unless you have customized your mouse settings in the operating system.	
Descriptions containing "click...", "mouse click," or "double-click" mean a click with the primary (left) mouse button. If the right mouse button must be used, this is explicitly stated in the description.	

1.3 Intended Use

The DKM FX and DKM FXC API (application programming interface) is used to control a DKM FX and DKM FXC matrix externally by network (TCP/IP) or RS-232 serial interface connection. Both the RS-232 serial interface and the TCP/IP interface use the same API telegram commands for the operation of the DKM FX and DKM FXC matrix.

The DKM FX and DKM FXC API has been successfully implemented with various common media control systems. The DKM FX and DKM FXC API provides the full scope of switching functionality. It does not support the configuration of a DKM FX and DKM FXC system.

2 Safety Instructions

To ensure reliable and safe long-term operation of your device:

- ➡ Read this user manual carefully.
- ➡ Only use API telegram commands according to this user manual. Failure to follow the instructions described can endanger the security of your data.

3 Description

3.1 System Overview

3.1.1 Application

The matrix operating system offers various functions for an operation via DKM FX and DKM FXC API. The API telegram commands are mainly intended to switch sinks (CON Devices) to other sources (CPU Devices) with selectable access rights. API telegram commands are also used, for example, to activate and deactivate Multi-Screen-Control (MSC), to send messages to monitor displays of addressed CON Devices, or for an overall definition of the total switching status and for saving and loading such switching states. In addition, matrix clones can be parallelly switched as synchronized matrices (Stacking) via serial network interface.

Via API a telegram response is automatically sent to confirm the preservation of the telegram commands.

The matrix optionally provides an echo via API of all affected matrix processes that are executed, for example, via macros or functions key switching commands. This allows to continuously track changes of a matrix configuration and enables your own applications to be updated. To receive the API echo via TCP/IP, the sockets have to be kept open continuously (see chapter 5.1.3, page 23).

3.1.2 DKM FX and DKM FXC Matrix System

A DKM FX and DKM FXC matrix system consists of a DKM FX and DKM FXC matrix and, for KVM applications, one or more CPU Units/CON Units. The DKM FX and DKM FXC matrix is connected to the CPU Units/CON Units by interconnect cables or directly to the video devices when used as a video matrix.

All available ports of the matrix can be used either as an input or output port depending on components and equipment. Non-blocking access is available for all users. For example, a user's access is not limited by the activities of another user.

CPU Units are connected directly to the sources by the provided cables. Monitor(s), keyboards, and mice are connected to the CON Units. The communication between the DKM FX and DKM FXC matrix and the CPU Units/CON Units occurs over the respective interconnect cables.

The DKM FX and DKM FXC matrix supports a wide and flexible range of system configurations:

A part of the DKM FX and DKM FXC can be configured, for example, as a Single-Head workstation, a part as Dual-Head workstation, or even as Quad-Head workstation. In addition, there are configurations with KVM and USB 2.0 available.

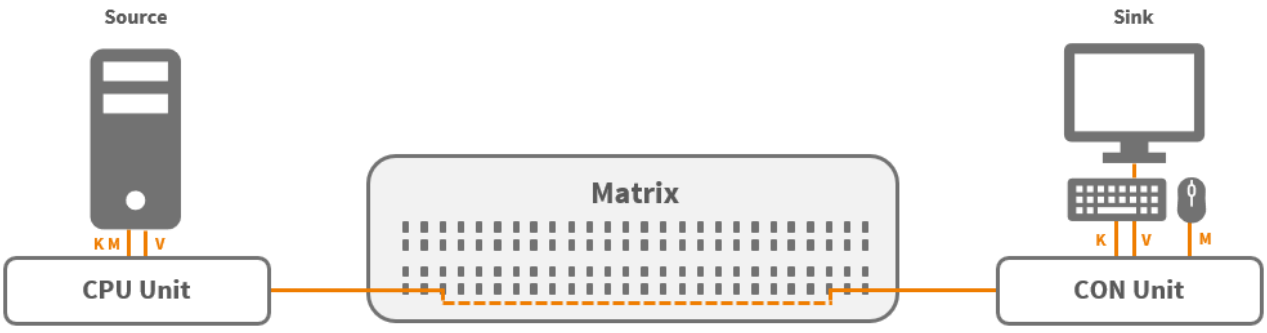


Fig. 1 Example - Matrix system with connected hardware

3.1.3 Matrix System Hardware and Logical Objects

On all DKM FX and DKM FXC matrices, switching extender modules follows the same principle:

- A CON/CPU Unit (hardware) is represented by an EXT Unit (logical object) in the matrix.
- This EXT Unit needs to be assigned to a CON or CPU Device (logical object).
- The actual switching takes place on the level of the CPU and CON Devices.

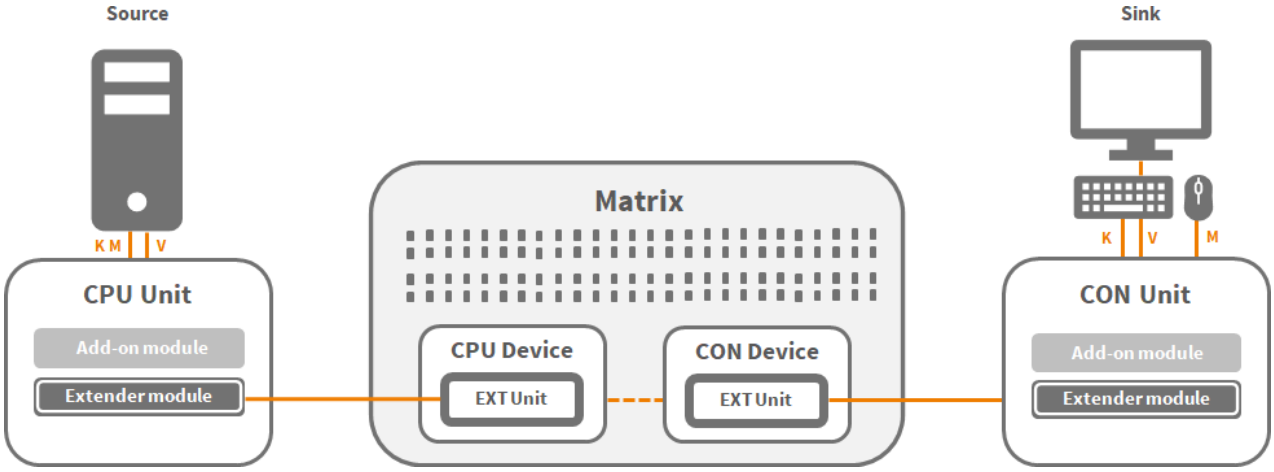


Fig. 2 Example - Matrix system with connected hardware and logical objects

3.1.4 Matrix Switching Possibilities

There are several possibilities to switch CON Devices to CPU Devices depending on the access rights of the user or CON Device and the configuration.

- Full Access (FA): The video is displayed on the monitor of the associated CON Device with USB HID control of the switched CPU Device. With enabled Sharing option, several users may have Full Access, but only one at a time. Others will remain in video only.
- Video only (VO): The video is displayed on the monitor connected to the switched CON Device without USB HID control of the switched CPU Device.
- Private Mode (PM): With enabled Private Mode, only one CON Device can be switched to the respective CPU Device. The CPU Device is not available for other connections.

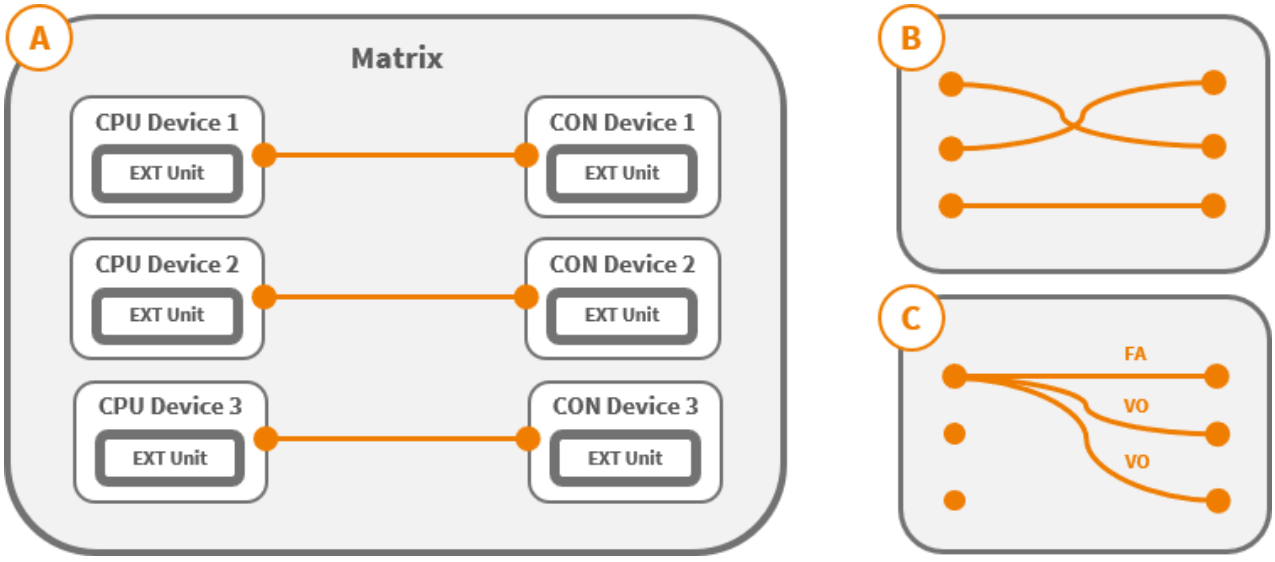


Fig. 3 Example - Matrix switching possibilities

3.1.5 Installation Example - Single-Head with External Control

An external serial control can be connected to the DKM FX and DKM FXC matrix via the CPU board and its connectors. The CPU board provides the possibility for both serial and TCP/IP connections.

- The serial connection to an external serial control is established by using a serial cable with D-Sub 9 connectors or a D-Sub 9 to RJ45 adapter cable (DKM FX and DKM FXC compact).
- The TCP/IP connection is established by using a Cat X network cable.

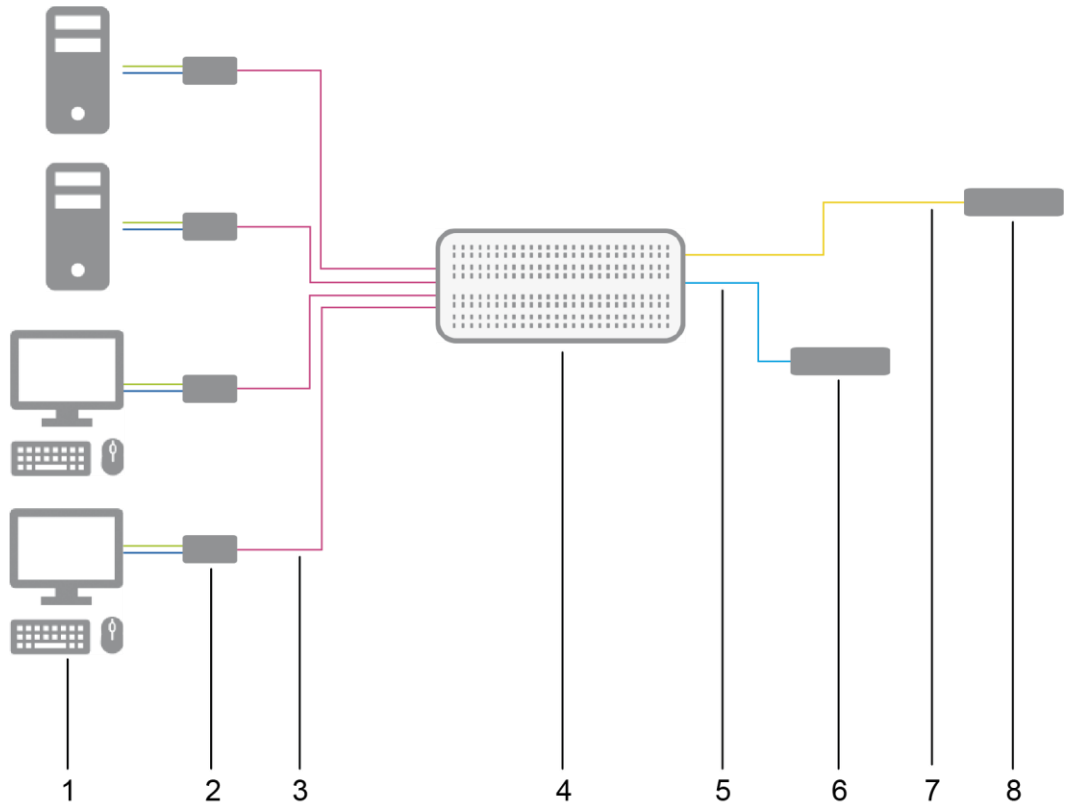


Fig. 4 Example - Matrix system with connected hardware and connected external control

1 Single-Head sources and sinks
2 Single-Head CPU Units and CON Units
3 Interconnect cable
4 DKM FX and DKM FXC matrix
5 Serial connection cable (D-Sub 9 connectors or a D-Sub 9 to RJ45 adapter cable
6 External serial control (RS232, option 1)
7 Network connection cable (Cat X)
8 External control (TCP/IP, option 2)

3.1.6 Extended Connection Examples

By sending API telegram commands to set extended connections, Full Access, Private Access, or Video-only connections can be set.

Full Access and Private Access Connection

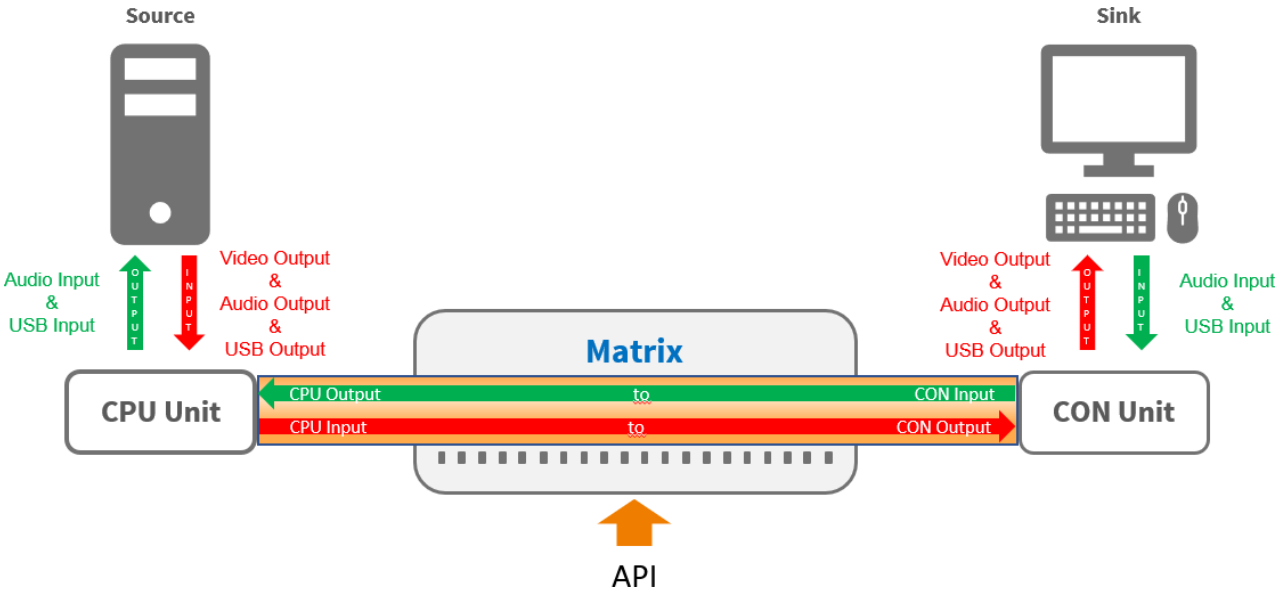


Fig. 5 Full Access and Private Access connection controlled by API telegram commands

Video-only Connection

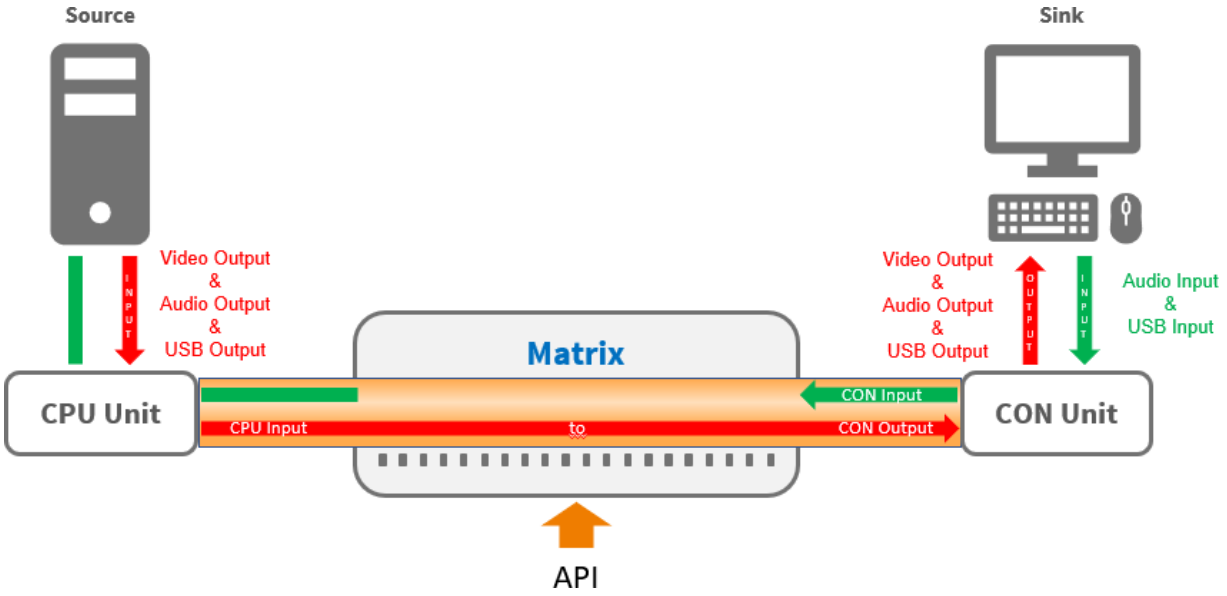


Fig. 6 Video-only connection controlled by API telegram commands

USB HID Control Connection

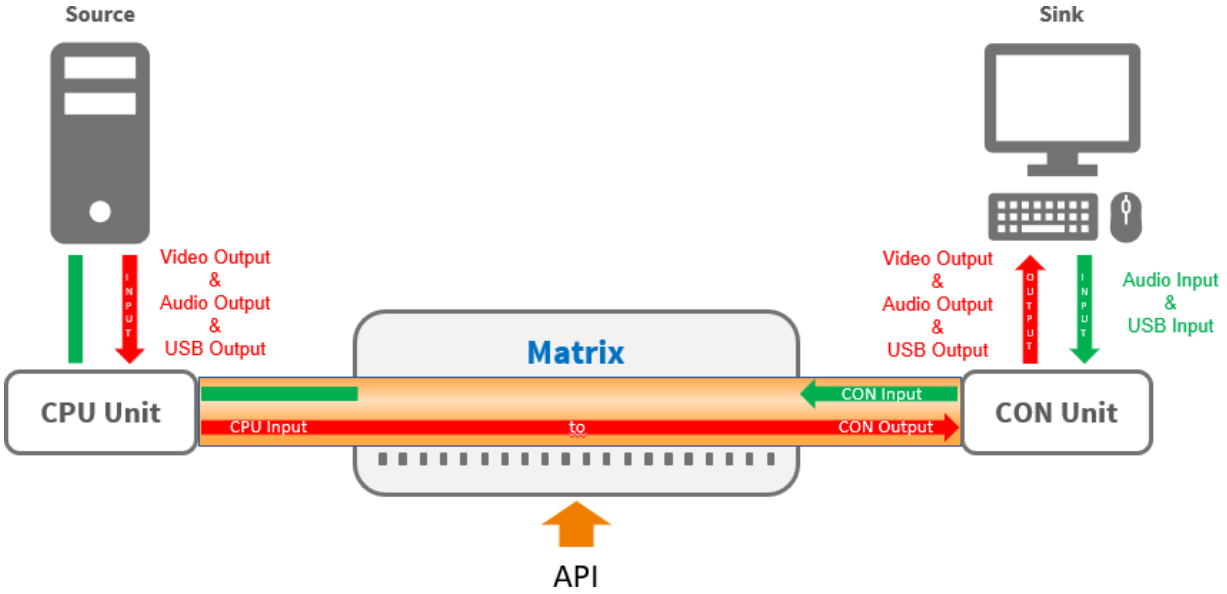


Fig. 7 USB HID Control connection controlled by API telegram commands

3.2 Telegram Structure

Constraints

- Minimum telegram length is 3 bytes
- Maximum buffer size for data transfer is 10240 bytes
- 16 sockets for TCP/IP communication over port 5555 are available. Ensure that there will be at least one socket left for the communication with the management software.
- Wait for a response before sending another request to the matrix.

Definition

The Little Endian format is used as byte order (valid for all commands larger than 1 byte).

Example: 1012 -> `0xF4 0x03` (not `0x03 0xF4`)

3.2.1 GET Request

Optional elements

Description	Control Character	Server identification	Command	Total length of telegram		Data
Hex coding	0x1B	1 Byte	1 Byte	1 Byte	1 Byte	n Bytes

Request

Type	Bytes	Description
Control character	1	Start byte always: ESC (`0x1B`)
Server identification	1	Identification of service = request: [(`0x5B`)
Command	1	A special command
Size	2	Optional, if telegram size > 3 bytes
Data	n	Optional, n bytes of data

Example

Telegram: `0x1B 0x5B 0x48 0x07 0x00 0xC9 0x0B`

Response

Type	Bytes	Description
Control character	1	Start byte always: ESC (`0x1B`)
Server identification	1	Identification of service = response:] (`0x5D`)
Command	1	A special command
Size	2	Optional, if telegram size > 3 bytes
Data	n	Optional, n bytes of data

Example

Telegram: `0x1B 0x5D 0x48 0x09 0x00 0xC9 0x0B 0xF4 0x03`

3.2.2 SET Command

Optional elements

Description	Control Character	Server identification	Command	Total length of telegram		Data
Hex coding	0x1B	1 Byte	1 Byte	1 Byte	1 Byte	n Bytes

Request

Type	Bytes	Description
Control character	1	Start byte always: ESC (0x1B)
Server identification	1	Identification of service = request: [(0x5B)
Command	1	A special command
Size	2	Optional, if telegram size > 3 bytes (see example in chapter 3.2.3, page 16)
Data	n	Optional, n bytes of data

Example

Telegram: 0x1B 0x5B 0x49 0x09 0x00 0xC9 0x0B 0xF4 0x03

Set CPU Device connection (input) to CON Device (output).

Response

Type	Bytes	Description
<ACK>	1	Acknowledge (0x06), no error. The matrix will send a confirmation if the SET command has been received and the matrix settings have been successfully changed.
<NAK>	1	Negative Acknowledge (0x15), error occurred. The matrix will send an error message if the SET command has been received but the matrix settings have not been successfully changed.
<BEL>	1	Busy (0x07), no time. The matrix will send a busy message if the matrix is busy and receiving the SET command is not possible.
[ECHO]	n	Reports the matrix sequences solicited by a command and, therefore, the matrix's new switching status. The echo can be used to update user applications and to operate several matrices in parallel. To activate the Echo Mode, see chapter 5.1.2, page 22 (for network interface) or chapter 5.2.2, page 24 (serial interface).

Example

<ACK>[ECHO]

[ ] = Optional elements

✅ Use the [ECHO] reports to verify that the switch commands have been executed as requested. Update the external switch status according to the [ECHO] reports rather than according to your commands.

3.2.3 Size of Telegram

The size of the telegram contains all bytes of the request or the response. See the following example:

Ctrl Char	Server Ident	Command	Total length of telegram	CpuID	ConID	Mode
0x1B	0x5B	0x62	0x0B 0x00	0xE9 0x03	0xB9 0x0B	0x00 0x00

Byte count	1	2	3	4	5	6	7	8	9	10	11
Hex coding	0x1B	0x5B	0x62	0x0B	0x00	0xE9	0x03	0xB9	0x0B	0x00	0x00

3.2.4 Strings

All strings are NUL-terminated, for example, the output of names that end with a NUL byte. After the NUL byte, the interpretation must be ended. All subsequent bytes are undefined and must not be interpreted.

Example:

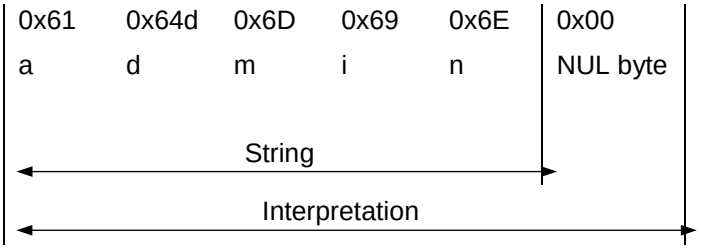


Fig. 8 String example

3.3 Data Communication

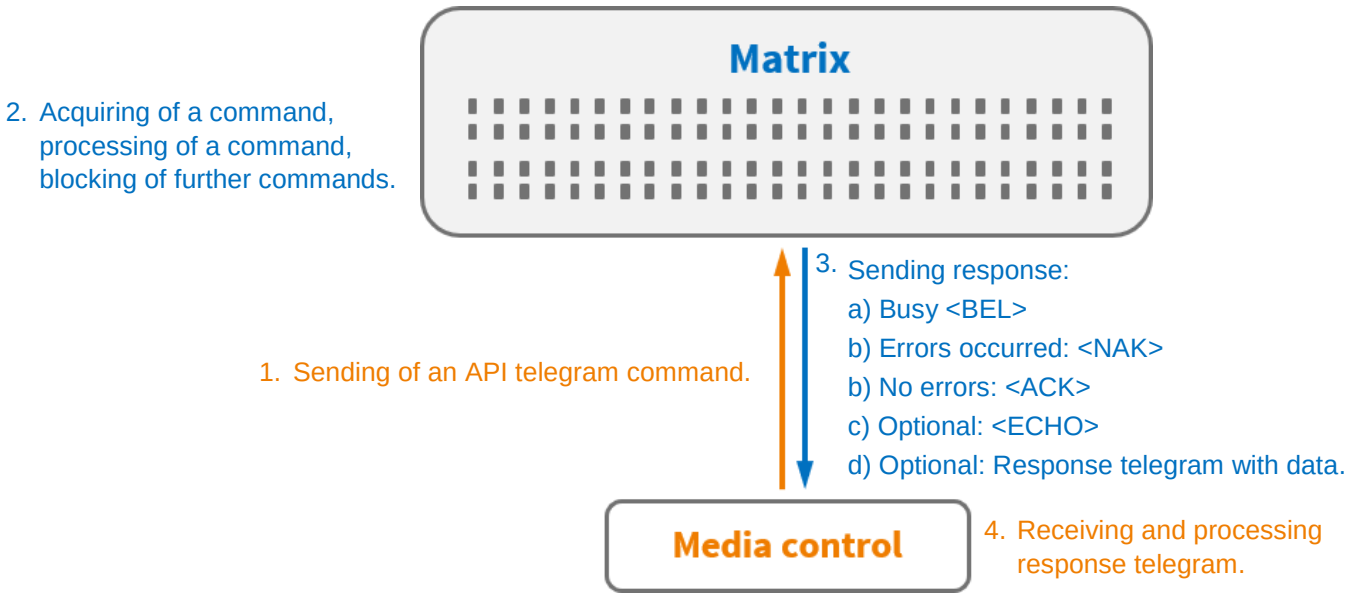


Fig. 9 Process sequence of a data communication

3.4 DKM FX and DKM FXC API Basics

Converting from decimal to hexadecimal

Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal
1	00 01	254	00 FE	1001	03 E9
2	00 02	255	00 FF	1002	03 EA
3	00 03	256	01 00	1003	03 EB
4	00 04	257	01 01	1004	03 EC
5	00 05	258	01 02	1005	03 ED
6	00 06	259	01 03	1006	03 EE
7	00 07	260	01 04	1007	03 EF
8	00 08	261	01 05	1008	03 F0
9	00 09	262	01 06	1009	03 F1
10	00 0A	253	01 07	1010	03 F2
11	00 0B	264	01 08	1011	03 F3
12	00 0C	265	01 09	1012	03 F4

Converting Device IDs for API Use

To switch CON Devices to CPU Devices, the Device IDs have to be converted from decimal to hexadecimal values. The nibbles (half bytes) of the hexadecimal ID values have to be swapped for API use (Little Endian format).

CPU IDs

ID (decimal)	ID (hex)	ID (hex) in API
1001	03 E9	E9 03
⋮	⋮	⋮
1999	07 CF	CF 07

CON IDs

ID (decimal)	ID (hex)	ID (hex) in API
3001	0B B9	B9 0B
⋮	⋮	⋮
3999	0F 9F	9F 0F

For example, the hex coding for the CON Device 3017 is `0xC9 0x0B`.

The nibbles (half bytes) of the hexadecimal ID values of all command types larger than 1 byte must be swapped for API use (Little Endian format).

3.5 API Telegram Command Structure with different Applications

Here are some examples for sending API telegram commands with different applications.

API Switch Telegram Command										
ESC	**SRV**	**CMD**	**Size**		**CPU ID High-, Low-Byte**		**CON ID High-, Low-Byte**		**Mode**	
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
0x1B	0x5B	0x62	0x0B	0x00	0xF4	0x03	0xC9	0x0B	0x01	0x00

API Switch Telegram Command for use in Packet Sender App (www.Packet Sender.com)

1B 5B 62 09 00 00 00 B9 0B

API Switch Telegram Command for use with Linux (with netcat, Nmap.org)

Installing nmap in debian linux: "apt-get install nmap"

- Not encrypted API

  echo -ne "\x1B\x5B\x62\x09\x00\x00\x00\xB9\x0B\x\x | ncat 192.168.100.200 5555 -d 0.1
- Encrypted (SSL) API

  echo -ne "\x1B\x5B\x62\x09\x00\x00\x00\xB9\x0B\x\x | ncat -C --ssl 192.168.100.200 5565 -d 0.1

4 Installation of External Control

✅ We recommend that first-time users set up the system in a test environment that is limited to a single room. This makes it easier to identify and solve any cabling problems, and experiment with your system more conveniently.

Requirements

The matrix system has been connected and basically configured according to the DKM FX and DKM FXC user manual:

- The hardware of the matrix system has been physically connected (matrix, controller board(s), I/O boards, CON Units, CPU Units, sources, and sinks, etc.).
- The initial configuration has been set (system settings, for example, network settings).
- The logical devices have been created and configured (EXT Units and CON/CPU Devices, etc.).
- A status of the matrix configuration has been stored as backup file.

Setting up an External Control

1. Connect a monitor, a keyboard, and a mouse to the CPU board.

   This is required to enable the API Service in the matrix.
2. Connect the matrix to the power sockets.
3. Connect the power supply voltage to the matrix.
4. Wait until the matrix's boot process is finished and the status LED of the controller board(s) flashes green.
5. Connect the external control either via RS-232 or TCP/IP to the matrix.

ℹ If using a TCP/IP connection, the matrix's IP address and the external controller's IP address must be in the same subnet, or a corresponding gateway must be configured

5 Configuration

In the following section, all relevant settings via OSD or management software are described. For a detailed explanation about OSD or management software, please refer to the DKM FX and DKM FXC user manual.

Enabling Network Connection as API

To enable sending API telegram commands and sending echoes via network interface, some settings must be configured first:

- Enabling sending API telegram commands (see chapter 5.1.1, page 21).
- Enabling sending echoes (see chapter 5.1.2, page 22).
- Keeping the IP socket open (see chapter 5.1.3, page 23).

Enabling Serial Connection as API

To ensure sending API telegram commands and sending echoes via serial interface, some settings must be configured first:

- Setting the format for serial data transmission (see chapter 5.2.1, page 24).
- Enabling sending echoes (see chapter 5.2.2, page 24).

NOTICE

Loss of configuration changes

By clicking on **Okay** (OSD) or **Apply** (management software), changes are applied to the active configuration and saved in the matrix's volatile memory. In the event of a sudden power failure, these changes are lost. To save changes permanently:

➡ save the configuration changes into the active configuration (**Remote Save**, only management software), save a predefined configuration (**Save as…,** both configuration options), perform a restart (**Restart Matrix** (OSD) or **Restart Device** (management software)). For more information, please refer to the DKM FX and DKM FXC matrix user manual.

After changing the system's configuration, we recommend that you de-register the primary controller board and boot to the secondary controller board until the boot process is finished.

Configurations can be saved as a file that can be stored independently of the matrix.
We recommend saving a matrix status every time when a configuration has been changed.

5.1 Enabling Network Connection as API

5.1.1 Enabling API Telegram Commands

Before you can operate the DKM FX and DKM FXC matrix via API telegram commands, you need to configure the matrix via OSD settings.

The serial interface can be blocked while OSD has been opened.

To enable the operation via API telegram commands:

1. Select **Configuration > Network** in the task area.
2. Set **Y** in the **API Service #1** field to activate the network interface at the matrix for accessing via API telegram commands or via management software (default, API service port 5555/5565).

   Note: If **Dual Interface** is activated for a matrix with two network ports, both network ports can be activated for the API service.
3. Click **Okay** to confirm your entries.

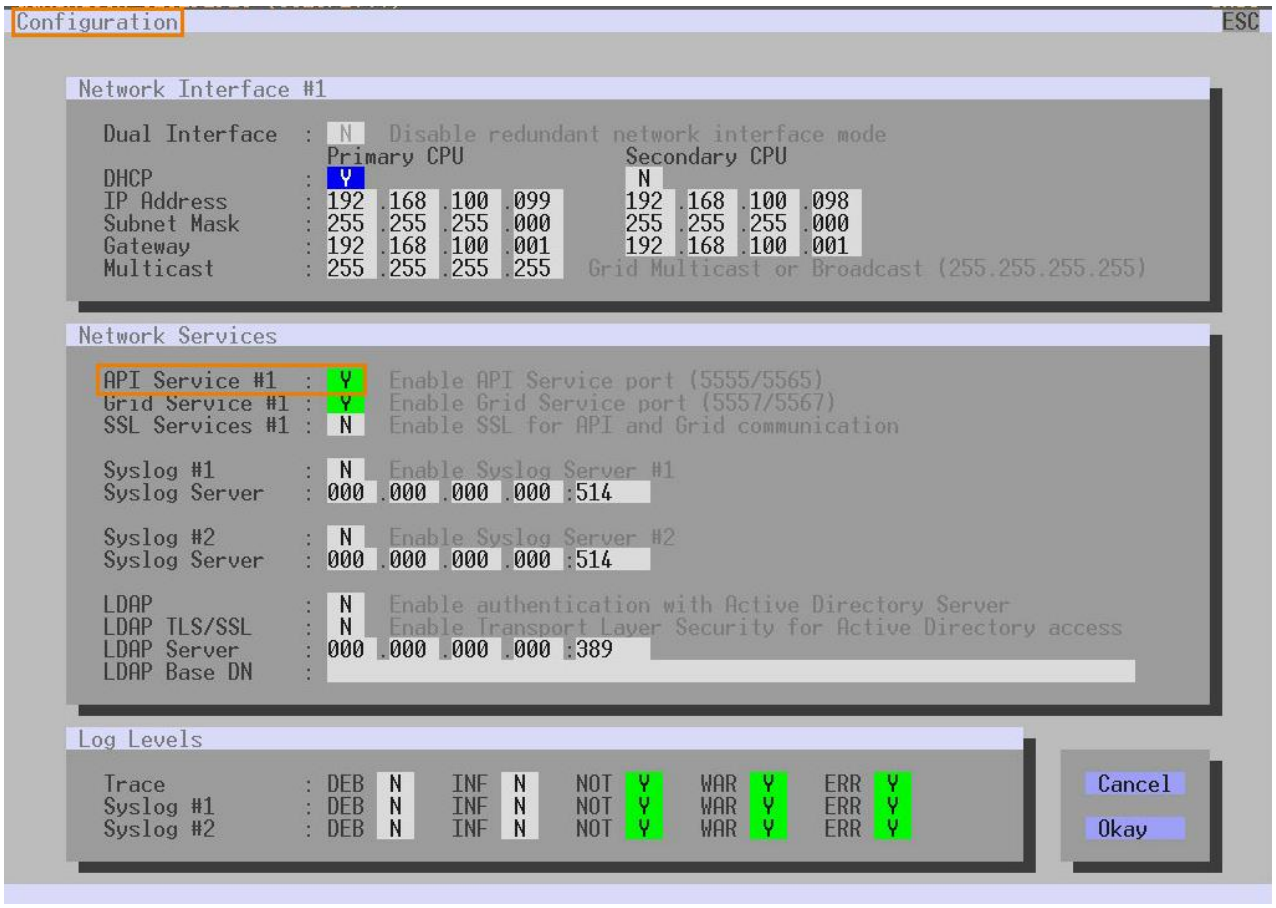


*Fig. 10 OSD Menu **Configuration - Network***

5.1.2 Enabling of Sending Echoes for Switching Processes

To send all performed switching commands both via API telegram command and without API telegram commands in the matrix as an echo, the API Service has to be enabled (see previous chapter).

Configuration via OSD

To send all performed switching commands in the matrix as an echo, the following options have to be enabled, depending on the interface used:

1. Select **Configuration > System** in the main menu.
2. Enter Y in the **Enable LAN Echo** field to enable sending echoes via network interface (usable with DKM FX and DKM FXC enterprise, flex and compact).
3. Enter Y in the **Enable old Echos** field to enable sending echoes to media control systems with API control configured with matrix firmware version older than V02.09. Current switching commands are internally translated into the old, already known, switching commands.
4. Click **Okay** to confirm your entries.

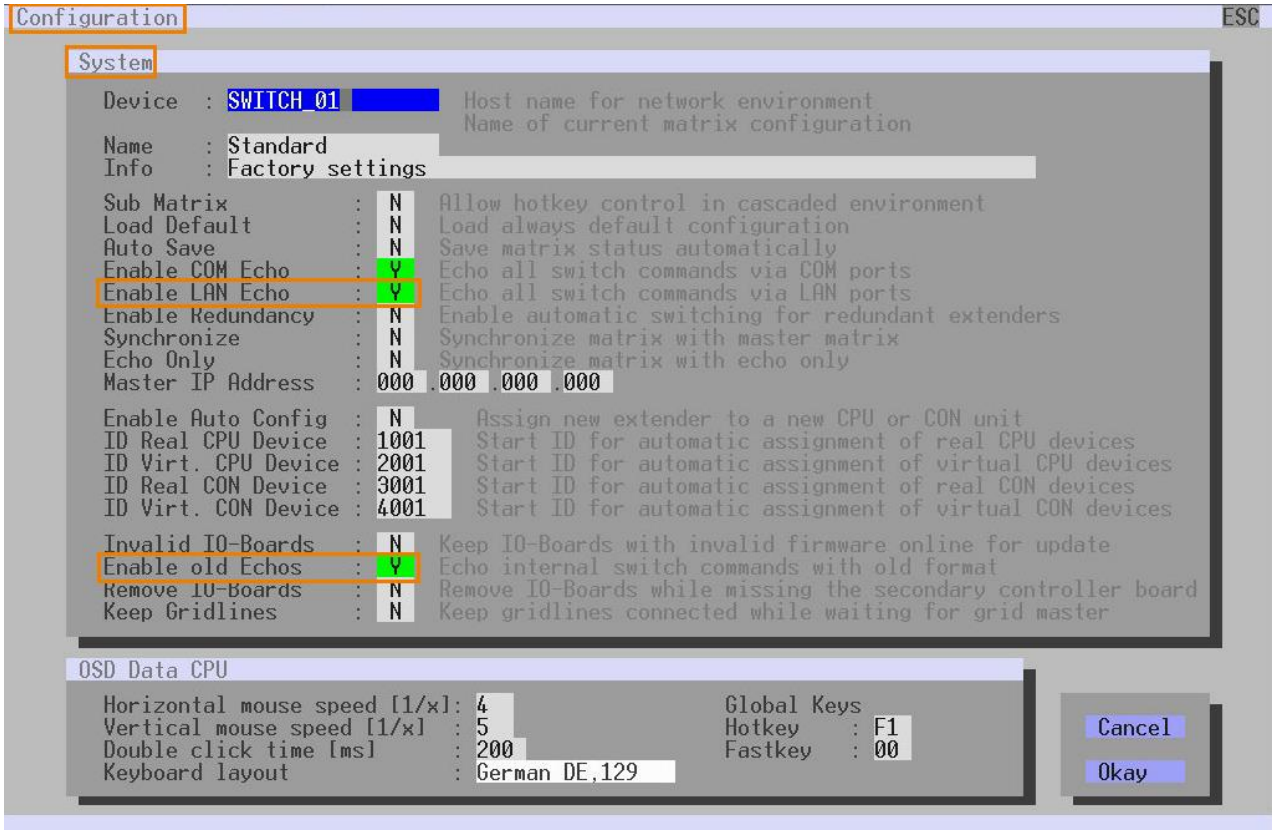


*Fig. 11 OSD Menu **Configuration - System***

5.1.3 Keeping IP Socket Connection

Communication	IP Sockets
Non-encrypted	Matrix IP address with TCP Port 5555.
SSL-encrypted	Matrix IP address with TCP Port 5565.

Keeping IP Sockets open for Sending Echo for Switching Processes without API Telegram Commands

The DKM FX and DKM FXC matrix includes 16 network sockets. These sockets are kept open for 30 seconds. If there is no keep alive signal in between this period, the socket will be closed again.

To send echoes for all affected matrix processes that are executed without API telegram commands, for example, switching via function keys as an echo, the IP socket connection must be kept open.

To keep the IP socket connection to the matrix open, there are two possibilities:

- Repetitive sending the API telegram command **Get System Time**
- Repetitive sending the API telegram command **Get System Status**

5.1.3.1 Get System Time

Request

Get system time (“ESC (S”). For detailed description, see chapter 6.1.1, page 25.

Control Character	Server identification	Command
0x1B	0x28	0x53
ESC	(	S

5.1.3.2 Get System Status

Request

Get system status (“ESC [z”). For detailed description, see chapter 6.1.2, page 26.

Control Character	Server identification	Command
0x1B	0x5B	0x7A
ESC	[	z

5.2 Enabling Serial Connection as API

5.2.1 Setting Serial Data Transmission Format

To establish the serial communication to the DKM FX and DKM FXC enterprise or compact, set the format for serial data transmission to the following parameters:

115.2K, 8, 1, NO (115.2 KBAUD, 8 data bits, 1 stop bit, no parity)

5.2.2 Enabling of Sending Echoes for Switching Processes

To send all performed switching commands both via API telegram command or without API telegram commands in the matrix as an echo, some settings have to be made in the matrix via OSD or via management software (chapter 5.1.2, page 22).

Configuration via OSD

The serial interface can be blocked while OSD has been opened.

To send all performed switching commands in the matrix as an echo, the following options have to be enabled, depending on the interface used:

1. Select **Configuration > System** in the main menu.
2. Enter Y in the **Enable COM Echo** field to enable sending echoes via RS-232 interface (usable with DKM FX and DKM FXC enterprise and compact).
3. Enter Y in the **Enable old Echoes** field to enable sending echoes to media control systems with API control configured with matrix firmware version older than V02.09. Current switching commands are internally translated into the old, already known, switching commands.
4. Click **Okay** to confirm your entries.

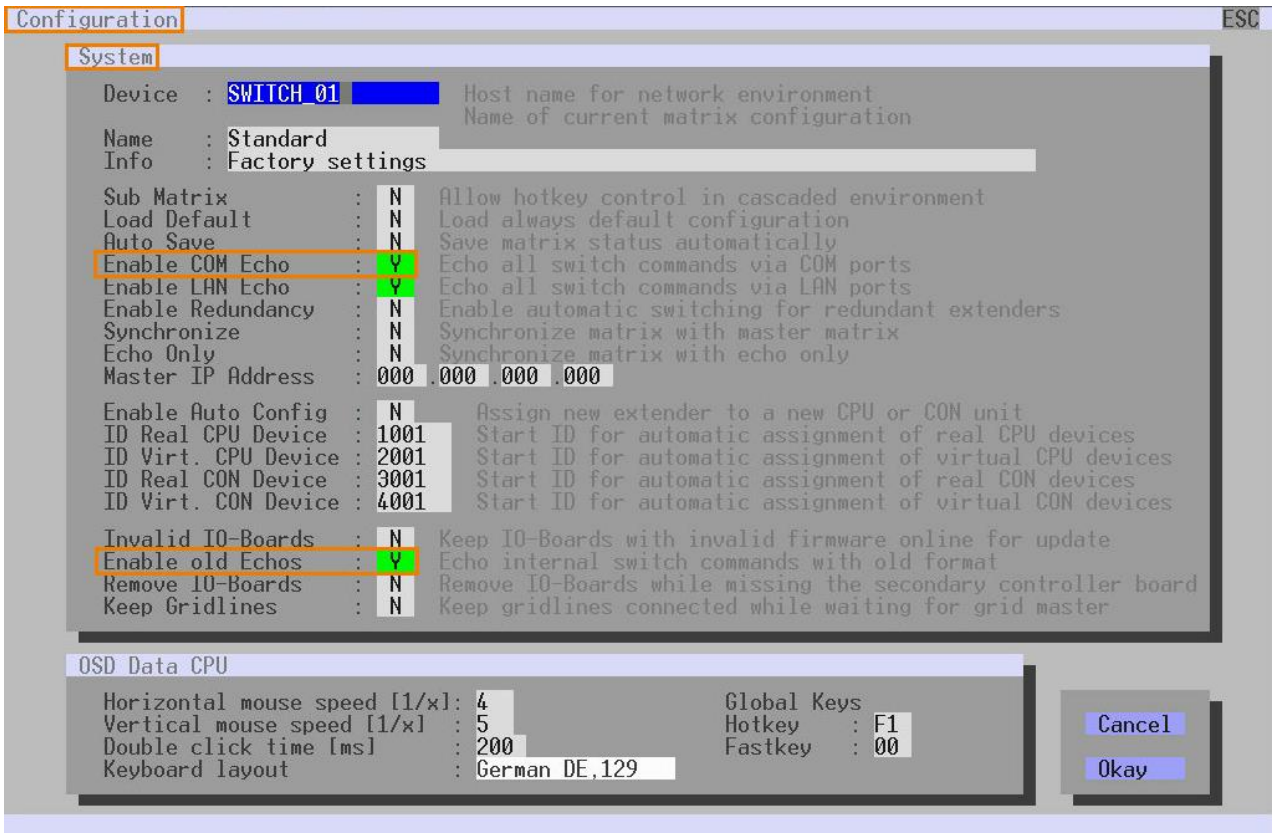


*Fig. 12 OSD Menu **Configuration - System***

6 Basic API Commands and Best Practice

This chapter provides an overview of the most commonly used switching commands and how they can be operated by using proven code examples of the external serial control.

6.1 Basic Commands

6.1.1 Get System Time

Request

Get system time via API telegram command: `ESC ( S`

Control Character	Server identification	Command
0x1B	0x28	0x53
ESC	(	S

Response Example

Return system time (“ESC) S”, example)

Control Character	Server identification	Command	Total length of telegram		Seconds	Minutes	Hours	Weekday	Calendar Day	Calendar Month	Calendar Year (2000+)
0x1B	0x29	0x53	0x0C	0x00	0x48	0x27	0x15	0x06	0x28	0x01	0x12
ESC	)	S	Size								

	Hex coding
Seconds (0-59)	0x00 – 0x59
Minutes (0-59)	0x00 – 0x59
Hours (0-23)	0x00 – 0x23
Weekday (1-7, Monday = 1)	0x01 – 0x07
Calendar Day (1-21)	0x01 – 0x31
Calendar Month (1-12)	0x01 – 0x12
Calendar Year (+2000)	0x00 – 0x23

Translation: 2012-28-01, Saturday, 15:27:48

6.1.2 Get System Status

Request

Get system status via API telegram command: `ESC [ z`

Control Character	Server identification	Command
0x1B	0x5B	0x7A
ESC	[	z

Response Example

Return system status (“ESC] z”, example).

The system status is encoded in the CRC format, see following page.

Telegram: `0x1B 0x5D 0x7A 0x19 0x00 0x2D (00101101) 0x0D (00001101) 0xdb (11011011) 0x50 (01010000) 0x00 0x00 0x00 0x00 0x08 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x06 0x00 0x00 0x00`

Control Character	Server identification	Command	Total length of telegram		Task active	Switch CPU status	PSU status	PSU FAN status	Not in use				Grid lines		
0x1B	0x5D	0x7A	0x19	0x00	0x2D	0x0D	0xDB	0x50	0x00	0x00	0x00	0x00	Active	Busy	Free
													0x08	0x02	0x06
													0x00	0x00	0x00
													0x00	0x00	0x00
													0x00	0x00	0x00
ESC	]	z	Size												

00000001	Main active	Switch IC active	PSU1 available	PSU3 Error
00000010	LAN active	Switch over-temp	PSU1 voltage OK	PSU4 available
00000100	LAN Grid active	Grid active	PSU1 Error	PSU4 voltage OK
00001000	Switch active	Master Matrix	PSU2 available	PSU4 Error
00010000	Sync active	576 Master CPU	PSU2 voltage OK	Fan 1 OK
00100000	UART active	576 Sub CPU	PSU2 Error	Fan 1 Error
01000000	INT active	Red. primary CPU	PSU3 available	Fan 2 OK
10000000	OSD1 active	Red. secondary CPU	PSU3 voltage OK	Fan 2 Error

2D = 00101101 : Main active, LAN Grid active, Switch active, UART active

0D = 00001101 : Switch IC active, Grid active, Master Matrix

DB = 11011011: PSU 1,2, and 3 OK, PSU 1,2, and 3 voltage ok

50 = 01010000 : Fan1 and 2 ok

8 Grid lines active, 2 Grid lines busy, 6 Grid lines free,

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
z	1	Command	0x7A
Size	2	Total length of telegram (25 bytes)	0x19 0x00
Bitset 1	1		0x00 to 0xFF
		Bit 00: taskMAIN active	00000001
		Bit 01: LANAPI active	00000010
		Bit 02: LANGRID active	00000100
		Bit 03: taskSWITCH active	00001000
		Bit 04: taskSYNC active	00010000
		Bit 05: taskUART active	00100000
		Bit 06: taskINT active	01000000
		Bit 07: taskOSD1 active	10000000
Bitset 2	1		0x00 to 0xFF
		Bit 08: Switch IC active	00000001
		Bit 09: Switch over-temp.	00000010
		Bit 10: Grid active	00000100
		Bit 11: Master Matrix	00001000
		Bit 12: 576er Master CPU	00010000
		Bit 13: 576er Sub CPU	00100000
		Bit 14: Redundancy primary CPU	01000000
		Bit 15: Redundancy secondary CPU	10000000
Bitset 3	1		0x00 to 0xFF
		Bit 16: PSU 1 available	00000001
		Bit 17: PSU 1 voltage ok	00000010
		Bit 18: PSU 1 error	00000100
		Bit 19: PSU 2 available	00001000
		Bit 20: PSU 2 voltage ok	00010000
		Bit 21: PSU 2 error	00100000
		Bit 22: PSU 3 available	01000000
		Bit 23: PSU 3 voltage ok	10000000
Bitset 4	1		0x00 to 0xFF
		Bit 24: PSU 3 error	00000001
		Bit 25: PSU 4 available	00000010
		Bit 26: PSU 4 voltage ok	00000100
		Bit 27: PSU 4 error	00001000
		Bit 28: Fan 1 ok	00010000
		Bit 29: Fan 1 error	00100000
		Bit 30: Fan 2 ok	01000000
		Bit 31: Fan 2 error	10000000
Bitset 5	1	Not in use	0x00
Bitset 6	1	Not in use	0x00
Bitset 7	1	Not in use	0x00

Type	Bytes	Description	Hex coding
Bitset 8	1	Not in use	`0x00`
Bitset 9	4	GLActive: Total number of Grid Lines in the system	E.g., 8 Grid Lines `0x08 0x00 0x00 0x00`
Bitset 10	4	GLBusy: Number of Grid Lines in use	E.g., 2 Grid Lines `0x02 0x00 0x00 0x00`
Bitset 11	4	GLFree: Number of unused Grid Lines [GLActive - GLBusy]	E.g., 6 Grid Lines `0x06 0x00 0x00 0x00`

6.1.3 Disconnect all Ports

Request

Telegram: `ESC [ A`

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
A	1	Command	0x41

Request Example

Telegram: `0x1B 0x5B 0x41`

Disconnect all ports.

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

6.2 Set Extended Connection

Extended commands represent the Hotkey commands for Full Access, Video-only and Private Access within the API.

Telegram: `ESC [ b Size CpuID ConID Mode`

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
b	1	Command	0x62
Size	2	Total length of telegram (11 bytes)	0x0B 0x00
CpuID	2	ID of a CPU Device	E.g., 1012 = 0xF4 0x03
		0 = Disconnect connection	0x00 0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B
Mode	2	Connection mode	
		0 = Full Access	0x00 0x00
		1 = Video-only	0x01 0x00
		2 = Private Mode	0x02 0x00
		3 = USB HID Control	0x03 0x00

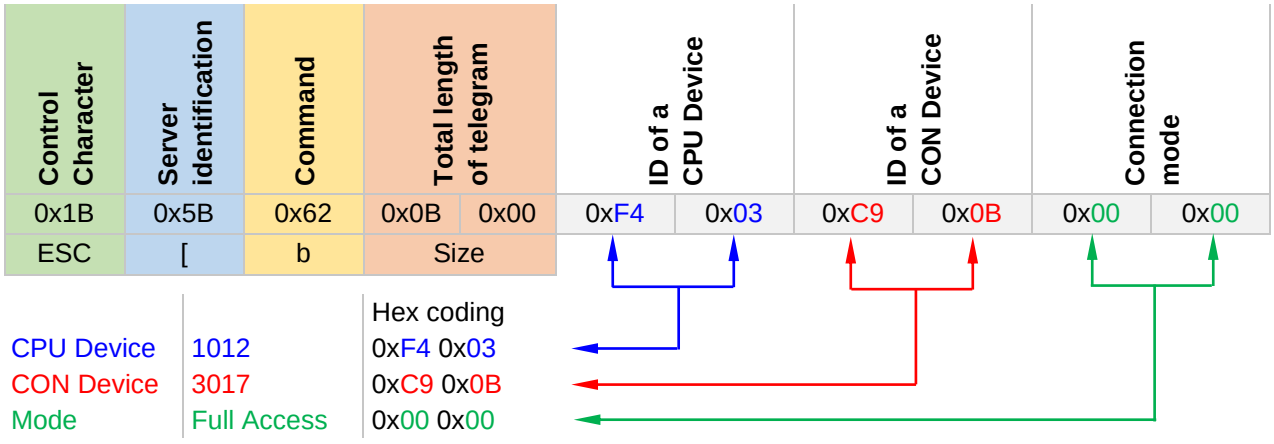
Disconnecting KVM Connection

For disconnecting just use `0x00 0x00` instead of the concrete CpuID.

6.2.1 Establishing a KVM Connection (Full Access)

- Set CON Device (input) connection to CPU Device (output).
  Data of CON Device (USB, Audio, …) is transmitted to a CPU Device.
- Set CPU Device (input) connection to CON Device (output)
  Data of CPU Device (Video, USB, Audio, …) is transmitted to a CON Device.

For example, switch the CON Device (ConID = 3017) with Full Access to the CPU Device (CpuID = 1012) by sending the following API telegram command:

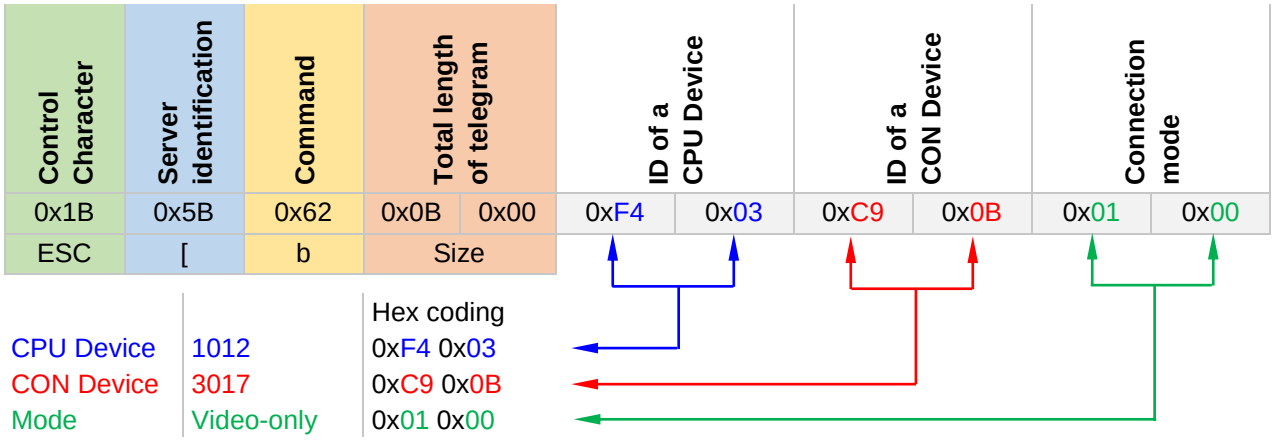


6.2.2 Establishing a Video Connection (Video-only Access)

Set CPU Device (input) connection to CON Device (output)

Data of CPU Device (Video, USB, Audio, …) is transmitted to a CON Device.

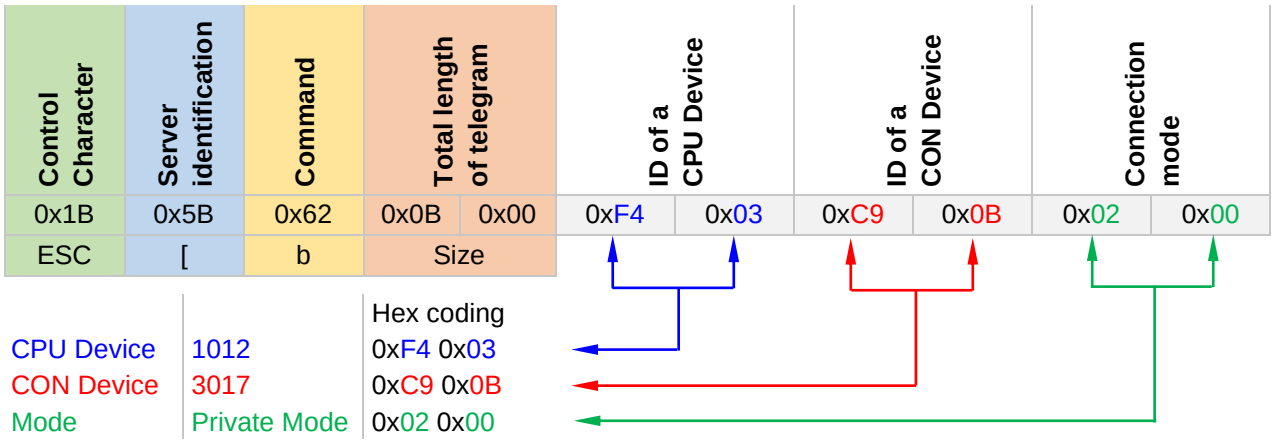
For example, switch the CON Device (ConID = 3017) with Video-only Access to the CPU Device (CpuID = 1012) by sending the following API telegram command:



6.2.3 Establishing an exclusive KVM Session (Private Access)

- Set CON Device (input) connection to CPU Device (output).
  Data of CON Device (USB, Audio, …) is transmitted to a CPU Device.
- Set CPU Device (input) connection to CON Device (output)
  Data of CPU Device (Video, USB, Audio, …) is transmitted to a CON Device.

For example, switch the CON Device (ConID = 3017) with Private Access to the CPU Device (CpuID = 1012) by sending the following API telegram command:



6.3 Establishing a USB 2.0 Data Connection (USB 2.0 Access)

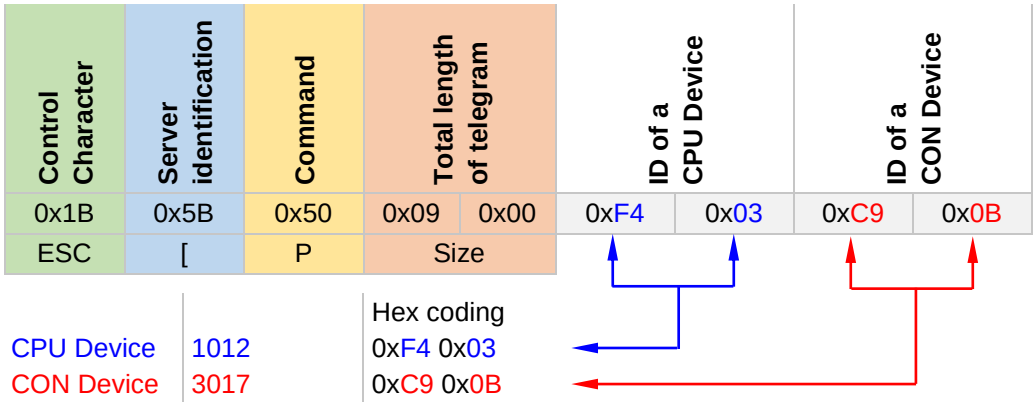
To set a USB 2.0 connection based on devices that only consists of USB 2.0 stand-alone extender modules, the bidirectional switching command is required.

- Set CON Device (input) connection to CPU Device (output).
  Data of CON Device (USB, Audio, …) is transmitted to a CPU Device.
- Set CPU Device (input) connection to CON Device (output)
  Data of CPU Device (Video, USB, Audio, …) is transmitted to a CON Device.

Telegram: `ESC [ P Size CpuID ConID`

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
P	1	Command	0x50
Size	2	Total length of telegram (9 bytes)	0x09 0x00
CpuID	2	ID of a CPU Device	E.g., 1012 = 0xF4 0x03
		0 = Disconnect connection	0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B

For example, set CON Device (ConID = 3017) with Full Access to the CPU Device (CpuID = 1012) by sending the following API telegram command:



Disconnecting USB 2.0 Connection

0x1B	0x5B	0x50	0x09	0x00	0x00	0x00	0xC9	0x0B

Switching a CON Device from an existing Connection

To switch from a device within an existing USB 2.0 connection to another device requires closing the current USB 2.0 connection at first. The disconnect must be performed by using the bidirectional command:

1. Disconnect the CPU Device (CpuID = 1012) from the CON Device (ConID = 3017):
   `0x1B 0x5B 0x50 0x09 0x00 0x00 0x00 0xC9 0x0B`

After disconnecting the existing connection, a switching break of 1 to 2 seconds is strongly recommended before the next switching operation is executed.

2. Connect to the new CON Device.

7 Operation

7.1 General Spellings for Telegram Data

7.1.1 Commands for Switching for CON and CPU Devices

Type	Description
ConCount	0 = All CON-CPU Device connections will be returned
	1...n = Number of CON-CPU Device connections will be returned
CpuCount	0 = All CPU-CON Device connections will be returned
	1...n = Number of CPU-CON Device connections will be returned
Count	Number of CPU-CON Device and CON-CPU Device connections will be returned
ConID	ID of a CON Device, example 0xC9 0x0B
ConID[1…n]	1...n = ID numbers of CON Devices in sequence, example 0xC9 0x0B 0xD4 0x0B
CpuID	ID of a CPU Device, example 0xF4 0x03
CpuID[1…n]	1...n = ID numbers of CPU Devices in sequence, example 0xF4 0x03 0xF5 0x03
<ConID CpuID>[1 …n]	1...n = Pairs of ID numbers of CON Devices connected to CPU Devices in sequence, example: 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03
<CpuID ConID>[1 …n]	1...n = Pairs of ID numbers of CPU Devices connected to CON Devices in sequence, example 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B

7.1.2 Commands for Assigning Virtual/Real CON and CPU Devices

Type	Description
RConCount	0 = All real CON Devices with assignments to virtual CON Devices will be returned
	1...n = Number of real CON Devices with assignments to virtual CON Devices will be returned
VCpuCount	0 = All virtual CPU Devices with assignments to real CPU Devices will be returned
	1...n = Number of virtual CPU Devices with assignments to real CPU Devices will be returned
RConID	ID of a real CON Device, example 0xC9 0x0B
RConID[1…n]	1...n = ID numbers of real CON Devices in sequence, example 0xC9 0x0B 0xD4 0x0B
VConID	ID of a virtual CON Device, example 0xC2 0x0F
VConID[1…n]	1...n = ID numbers of virtual CON Devices in sequence, example 0xC2 0x0F 0xCA 0x0F
RCpuID	ID of a real CPU Device, example 0xF4 0x03
RCpuID[1…n]	1...n = ID numbers of real CPU Devices in sequence, example 0xF4 0x03 0xF5 0x03
VCpuID	ID of a virtual CPU Device, example 0xE2 0x07
VCpuID[1…n]	1...n = ID numbers of virtual CPU Devices in sequence, example 0xE2 0x07 0xEE 0x07
<RConID VConID>[1…n]	1...n = Pairs of ID numbers of real CON Devices assigned to virtual CON Devices in sequence, example 0xC9 0x0B 0xC2 0x0F 0xD4 0x0B 0xCA 0x0F
<VCpuID RCpuID>[1…n]	1...n = Pairs of ID numbers of virtual CPU Devices assigned to real CPU Devices in sequence, example 0xE2 0x07 0xF4 0x03 0xEE 0x07 0xF5 0x03

7.1.3 Commands for Switching Input and Output Ports

Type	Description
PortCount	0 = All port connections will be returned
	1...n = Number of port connections will be returned
OutPort	Port number of an output port, example `0x02 0x00`
OutPort[1 …n]	1...n = Output port numbers in sequence, example `0x02 0x00 0x04 0x00`
InPort	Port number of an input port, example `0x08 0x00`
InPort[1 …n]	1...n = Input port numbers in sequence, example `0x08 0x00 0x0A 0x00`
<OutPort InPort>[1 …n]	1...n = Pairs of output port numbers connected to input port numbers in sequence, example `0x02 0x00 0x08 0x00 0x04 0x00 0x0A 0x00`

7.2 Bidirectional Commands

7.2.1 Set Connection of CON Device to CPU Device

Request

Telegram: `ESC [ P Size Count CpuID ConID`

Set or disconnect CON Device (input) connection to/from CPU Device (output) and CPU Device (input) connection to/from CON Device (output).
Input data of CON Device (USB, Audio, …) will be transmitted to CPU Device.
Input data of CPU Device (Video, USB, Audio, …) will be transmitted to CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
P	1	Command	`0x50`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
CpuID	2	ID of a CPU Device	E.g., 1012 = `0xF4 0x03`
		0 = Disconnect connection	`0x00 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`

Request Example

Telegram: `0x1B 0x5B 0x50 0x09 0x00 0xF4 0x03 0xC9 0x0B`

Set CON Device (ConID = 3017) connection to CON Device (CpuID = 1012) and CPU Device (CpuID = 1012) to CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.2 Set Connection of CON Devices to CPU Devices

Request

Telegram: `ESC [ Q Size Count <CpuID ConID>[1…n]`

Set or disconnect CON Devices (input) connection to/from CPU Devices (output) and CPU Devices (input) connection to/from CON Devices (output).
Input data of CON Device (USB, Audio, …) will be transmitted to CPU Device.
Input data of CPU Device (Video, USB, Audio, …) will be transmitted to CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
Q	1	Command	0x51
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for Count = 3 0x13 0x00
Count	2	1…n = Number of bidirectional CPU-CON Device connections	E.g., 3 = 0x03 0x00
CpuID	2	IDs of CPU Devices (maximum 1024 entries)	E.g., 1012 = 0xF4 0x03, 1013 = 0xF5 0x03, 1020 = 0xFC 0x03
		0 = Disconnect connection	0x00 0x00
ConID	2	IDs of CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B

Request Example for Connecting

Telegram: `0x1B 0x5B 0x51 0x13 0x00 0x03 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0xE0 0x0B`

Set connections of CON Devices to CPU Devices and CPU Devices to CON Devices, for example, the following devices:

CpuID[1] = 1012, ConID[1] = 3017;

CpuID[2] = 1013, ConID[2] = 3028;

CpuID[3] = 1020, ConID[3] = 3040;

Request Example for Disconnecting

Telegram: `0x1B 0x5B 0x51 0x13 0x00 0x03 0x00 0x00 0x00 0xC9 0x0B 0x00 0x00 0xD4 0x0B 0x00 0x00 0xE0 0x0B`

Disconnect connections of CON Devices from CPU Devices and CPU Devices from CON Devices, for example, the following devices:

CpuID[1] = 0, ConID[1] = 3017;

CpuID[2] = 0, ConID[2] = 3028;

CpuID[3] = 0, ConID[3] = 3040;

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.3 Get Connections for CON Devices and CPU Devices

Request

Telegram: `ESC [ R`

Get all CPU Device - CON Device connections.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
R	1	Command	0x52

Request Example

Telegram: `0x1B 0x5B 0x52`

Get all CPU Device - CON Device connections.

Response

Telegram: `ESC ] R Size CpuCount ConCount <CpuID ConID>[1…n] <ConID CpuID>[1…n]`

Return all CPU Device - CON Device connections for all specified CPU and CON Devices in pairs.

For each defined CPU Device, the ConID of the connected CON Device will be added, or 0 if the CPU Device is disconnected. For each defined CON Device, the CpuID of the connected CPU Device will be added, or 0 if the CON Device is disconnected.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
R	1	Command	0x52
Size	2	Total length of telegram (9 bytes + data = 29 bytes)	E.g., for CpuCount = 3 and ConCount = 2 0x1D 0x00
CpuCount	2	1...n = Number of CPU-CON Device connections will be returned	E.g., 3 = 0x03 0x00
ConCount	2	1...n = Number of CON-CPU Device connections will be returned	E.g., 2 = 0x02 0x00
CpuID	2	IDs of CPU Devices (maximum 1024 entries)	E.g., 1012 = 0xF4 0x03, 1013 = 0xF5 0x03, 1020 = 0xFC 0x03
		0 = No connection	0x00 0x00
ConID	2	IDs of CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B
		0 = No connection	0x00 0x00

Response Example

Telegram: 0x1B 0x5D 0x52 0x1D 0x00 0x03 0x00 0x02 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0xE0 0x0B 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0x00 0x00

Return CPU Device - CON Device connections in pairs, for example:

CpuID[1] = 1012, ConID[1] = 3017;
CpuID[2] = 1013, ConID[2] = 3028;
CpuID[3] = 1020, ConID[3] = 3040;

ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[2] = 0, no connection available;

Or <NAK>.

7.2.4 Set Connections for CON Devices and CPU Devices

Request

Telegram: `ESC [ S Size CpuCount ConCount <CpuID ConID>[1…n] <ConID CpuID>[1…n]`

Set or disconnect a connection for all defined CON Devices and CPU Devices.

Input data of CON Device (USB, Audio, …) will be transmitted to CPU Device.
Input data of CPU Device (Video, USB, Audio, …) will be transmitted to CON Device.

For each defined CPU Device add the ConID, or 0 if the CPU Device is disconnected. For each defined CON Device add the CpuID, or 0 if the CON Device is disconnected.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
S	1	Command	`0x53`
Size	2	Total length of telegram (9 bytes + data = 29 bytes)	E.g., for CpuCount = 3 and ConCount = 2 `0x1D 0x00`
CpuCount	2	1...n = Number of CPU-CON Device connections will be returned	E.g., 3 = `0x03 0x00`
ConCount	2	1...n = Number of CON-CPU Device connections will be returned	E.g., 2 = `0x02 0x00`
CpuID	2	IDs of CPU Devices (maximum 1024 entries)	E.g., 1012 = `0xF4 0x03`, 1013 = `0xF5 0x03`, 1020 = `0xFC 0x03`
		0 = Disconnect connection	`0x00 0x00`
ConID	2	IDs of CON Devices (maximum 1024 entries)	E.g., 3017 = `0xC9 0x0B`, 3028 = `0xD4 0x0B`, 3040 = `0xE0 0x0B`
		0 = Disconnect connection	`0x00 0x00`

Request Example for Connecting

Telegram: `0x1B 0x5B 0x53 0x1D 0x00 0x02 0x00 0x02 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03`

Set a connection for all defined CON Devices and CPU Devices:

CpuID[1] = 1012, ConID[1] = 3017;
CpuID[2] = 1013, ConID[2] = 3028;
ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[2] = 1013;

Request Example for Disconnecting

Telegram: `0x1B 0x5B 0x53 0x1D 0x00 0x01 0x00 0x01 0x00 0xFC 0x03 0xE0 0x0B`

Disconnect Full Access connection for CON Device (ConID = 3017) from CPU Device (CpuID = 1028) and CPU Device (CpuID = 1028) from CON Device (ConID = 3017):

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.5 Set Connection of Input Port to Output Port

Request

Telegram: `ESC [ C Size OutPort InPort`

Set connection of input port to output port and output port to input port.

Data of input port will be transmitted to output port, and output port will be transmitted to input port.

To set the connection, a CON Device has to be configured at the output port, and a CPU Device has to be configured to the input port. If no CPU Device is found, an existing connection will be disconnected.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
C	1	Command	`0x43`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
OutPort	2	1…2032 = Output port number connected to a CON Device	E.g., 10 = `0x0A 0x00`
InPort	2	1…2032 = Input port number connected to a CPU Device	E.g., 20 = `0x14 0x00`
		0 = Disconnect port	`0x00 0x00`

Request Example for Connecting

Telegram: `0x1B 0x5B 0x43 0x09 0x00 0x0A 0x00 0x14 0x00`

Set connection from output port (OutPort = 10) to input port (InPort = 20) and from input port (InPort = 20) to output port (OutPort = 10).

Request Example for Disconnecting

Telegram: `0x1B 0x5B 0x43 0x09 0x00 0x0A 0x00 0x00 0x00`

Disconnect input port (InPort = 0) from output port (OutPort = 10).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.6 Set Connection of Input Ports to Output Ports

Request

Telegram: `ESC [ E Size PortCount OutPort[1…n] InPort[1…n]`

Set connection of input ports to output ports and output ports to input ports.

Data of input ports will be transmitted to output ports, and output ports will be transmitted to input ports.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
E	1	Command	`0x45`
Size	2	Total length of telegram (11 bytes + data = 27 bytes)	E.g., for PortCount = 5 `0x1B 0x00`
PortCount	2	0 = All port connections will be returned	`0x00 0x00`
		1…n = Number of port connections will be returned (maximum 2032 port connections)	E.g., 5 = `0x05 0x00`
OutPort	2	Output port numbers	E.g., 2 = `0x02 0x00`, 4 = `0x04 0x00`, 5 = `0x05 0x00`
InPort	2	1…2032 = Input port numbers whose input signals are output to output port numbers	E.g., 8 = `0x08 0x00`, 10 = `0x0A 0x00`, 12 = `0x0C 0x00`
		0 = Disconnect port	`0x00 0x00`

Request Example

Telegram: `0x1B 0x5D 0x45 0x1b 0x00 0x05 0x00 0x02 0x00 0x08 0x00 0x04 0x00 0x0A 0x00 0x05 0x00 0x0C 0x00 0x08 0x00 0x02 0x00 0x0A 0x00 0x04 0x00`

Set a connection for all defined input ports to output ports:

OutPort[1] = 2, InPort[1] = 8;
OutPort[2] = 4, InPort[2] = 10;
OutPort[3] = 5, InPort[3] = 12;
InPort[1] = 8, OutPort[1] = 2;
InPort[2] = 10, OutPort[2] = 4;

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.7 Set the KVM Link of a CON Device with Forcing

With this command, the KVM connection of one or all CON Device is switched to one of the two KVM link ports. At the same time, it is stored in the matrix configuration that this link is desired and forced. This remains also after a restart.
If Enable Redundancy is active in the matrix configuration with forced KVM link switching, the redundancy option then no longer affects the respective CON Device. To enable the deactivated Enable Redundancy option again, the KVM connection must be reset back to default.

Request

Telegram: `ESC [ f Size ConID KVM Force`

Force KVM connection of all or specified CON Devices to defined link port or resets the forced KVM connections back to default.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
f	1	Command	`0x66`
Size	2	Total length of telegram (11 bytes)	`0x0B 0x00`
ConID	2	0 = All CON Devices are switched to the same KVM link.	`0x00 0x00`
		ID of CON Device	E.g., 3017 = `0xC9 0x0B`, 3008 = `0xC0 0x0B`
KVM	2	KVM link of the CON Device to be forced	
		0 = Default	`0x00 0x00` (no KVM link is forced)
		1 = Primary link port	`0x01 0x00`
		2 = Secondary link port	`0x02 0x00`
Force	2	0 = Link switching without forcing Enabled automatic redundancy remains activated and enabled automatic redundancy remains deactivated.	`0x00 0x00`
		1 = Link switching with forcing	`0x01 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x66 0x0B 0x00 0xC9 0x0B 0x01 0x00 0x01 0x00`

Force (Force = 1) KVM connection of CON Device (ConID = 3017) to primary link port (KVM = 1).

Request Example 2

Telegram: `0x1B 0x5B 0x66 0x0B 0x00 0xC9 0x0B 0x01 0x00 0x00 0x00`

Switch KVM connection of CON Device (ConID = 3017) to primary link port (KVM = 1) without forcing (Force = 0).

Enabled automatic redundancy (**Enable Redundancy** option) remains activated and disabled automatic redundancy remains deactivated.

Request Example 3

Telegram: `0x1B 0x5B 0x66 0x0B 0x00 0xC0 0x0B 0x02 0x00 0x01 0x00`

Force KVM connection of all CON Devices (ConID = 3008) to secondary link port (KVM = 2).

Request Example 4

Telegram: `0x1B 0x5B 0x66 0x0B 0x00 0x00 0x00 0x02 0x00 0x01 0x00`

Force (Force = 1) KVM connection of all CON Devices (ConID = 0) to secondary link port (KVM = 2).

Request Example 5

Telegram: `0x1B 0x5B 0x66 0x0B 0x00 0x00 0x00 0x00 0x00 0x01 0x00`

Set all CON Devices (ConID = 0) back to default. The force byte will be ignored and no KVM link is forced.

Enabled automatic redundancy (**Enable Redundancy** option) remains activated and disabled automatic redundancy remains deactivated.

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.8 Set KVM Connection of Single CON EXT Units

Request

Telegram: `ESC [ j Size ConID ExtID KVM`

Set CON EXT Unit KVM connection (input) to specified KVM link port (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
j	1	Command	`0x6A`
Size	2	Total length of telegram (11 bytes)	`0x0B 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
ExtID	2	1…n = Number of EXT Unit of the CON Device to be switched	E.g., 2 = `0x02 0x00`
		0 = Switch all EXT Units of a CON Device	`0x00 0x00`
KVM	2	KVM link of the CON Device to be forced	1 = Primary link port = `0x01 0x00`
			2 = Secondary link port = `0x02 0x00`
			3 = Local input port = `0x03 0x00`

Request Example

Telegram: `0x1B 0x5B 0x6A 0x0B 0x00 0xC9 0x0B 0x02 0x03 0x02 0x00`

Set CON EXT Unit KVM connection (ExtID = 2) of CON Device (ConID = 3017) to secondary link port (KVM = 2) and deactivate **Enable Redundancy** option if it has been activated.

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.2.9 Set KVM Connection to local Input Port

Request

Telegram: `ESC [ f Size ConID KVM`

Set KVM link of CON Device to local input port.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
f	1	Command	`0x66`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
		0 = All CON Devices switch to the same link	`0x00 0x00`
KVM	2	Active KVM link of the CON Device	3 = Local input port = `0x03 0x00`

Request Example

Telegram: `0x1B 0x5B 0x66 0x09 0x00 0xC9 0x0B 0x03 0x00`

Set active KVM link (KVM = 3) of CON Device (ConID = 3017) to local input port.

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.3 Unidirectional Commands from CPU Device to CON Device

7.3.1 Get CPU Device connected to CON Device

Request

Telegram: `ESC [ H Size ConID`

Get CPU Device (input) connected to CON Device (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
H	1	Command	0x48
Size	2	Total length of telegram (7 bytes)	0x07 0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B

Request Example

Telegram: `0x1B 0x5B 0x48 0x0 0x00 0xC9 0x0B`

Get CPU Device connected to CON Device (ConID = 3017).

Response

Telegram: `ESC ] H Size ConID CpuID`

Return CPU Device (input) connected to CON Device (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
H	1	Command	0x48
Size	2	Total length of telegram (9 bytes)	0x09 0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B
CpuID	2	ID of a CPU Device	E.g., 1012 = 0xF4 0x03
		0 = No connection	0x00 0x00

Response Example 1

Telegram: `0x1B 0x5D 0x48 0x09 0x00 0xC9 0x0B 0xF4 0x03`

Return CPU Device (CpuID = 1012) connected to CON Device (ConID = 3017).

Or <NAK>

Response Example 2

Telegram: `0x1B 0x5D 0x48 0x09 0x00 0xC9 0x0B 0x00 0x00`

No CPU Device (CpuID = 0) connected to CON Device (ConID = 3017).

Or <NAK>

7.3.2 Set Connection of CPU Device to CON Device

Request

Telegram: `ESC [ I Size ConID CpuID`

Set or disconnect CPU Device connection (input) to/from CON Device (output).
Input data of CPU Device (Video, USB, Audio, …) will be transmitted to CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
I	1	Command	`0x49`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
CpuID	2	ID of a CPU Device	E.g., 1012 = `0xF4 0x03`
		0 = Disconnect	`0x00 0x00`

Request Example for Connecting

Telegram: `0x1B 0x5B 0x49 0x09 0x00 0xC9 0x0B 0xF4 0x03`

Set CPU Device (CpuID = 1012) connection to CON Device (ConID = 3017).

Request Example for Disconnecting

Telegram: `0x1B 0x5B 0x49 0x09 0x00 0xC9 0x0B 0x00 0x00`

Disconnect CPU Device (CpuID = 0) connection from CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.3.3 Get CPU Devices connected to CON Devices

Request

Telegram: `ESC [ J Size ConCount ConID[1…n]`

Get CPU Devices (input) connected to specified or all CON Devices (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
J	1	Command	0x4A
Size	2	Total length of telegram (7 bytes + data = 13 bytes)	E.g., for ConCount = 3 0x0D 0x00
ConCount	2	0 = All CON-CPU Device connections will be returned	0x00 0x00
		1…n = Number of CON-CPU Device connections will be returned	E.g., 3 = 0x03 0x00
ConID	2	IDs of CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B

Request Example for Getting specified CPU Devices

Telegram: `0x1B 0x5B 0x4A 0x0D 0x00 0x03 0x00 0xC9 0x0B 0xD4 0x0B 0xE0 0x0B`

Get CPU Devices (CpuIDs) connected to specified CON Devices (ConID = 3017, 3028, 3040).

Request Example for Getting all CPU Devices

Telegram: `0x1B 0x5B 0x4A 0x07 0x00 0x00 0x00`

Get all CPU Devices (CpuIDs) connected to all CON Devices (ConCount = 0).

Response

Telegram: `ESC ] J Size ConCount <ConID CpuID>[1…n]`

Return a list of CPU Devices (input) connected to CON Devices (output) as pairs.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
J	1	Command	0x4A
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for ConCount = 3 0x13 0x00
ConCount	2	1…n = Number of CON-CPU Device connections (maximum 1024 connections)	E.g., 3 = 0x03 0x00
ConID	2	IDs of CON Devices	E.g., 1012 = 0xF4 0x03, 1013 = 0xF5 0x03, 1020 = 0xFC 0x03
CpuID	2	IDs of CPU Devices	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B
		0 = No connection	0x00 0x00

Response Example 1

Telegram: 0x1B 0x5D 0x4A 0x13 0x00 0x03 0x00 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03 0xE0 0x0B 0xFC 0x03

Return CPU Devices connected to CON Devices, for example:

ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[2] = 1013;
ConID[3] = 3040, CpuID[3] = 1020;

Or <NAK>.

Response Example 2

Telegram: 0x1B 0x5D 0x4A 0x13 0x00 0x03 0x00 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03 0xE0 0x0B 0x00 0x00

Return CPU Devices connected to CON Devices, for example:

ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[2] = 1013;
ConID[3] = 3040, CpuID[3] = 0, no connection available;

Or <NAK>.

7.3.4 Set Connections of CPU Devices to CON Devices

Request

Telegram: `ESC [ K Size ConCount <ConID CpuID>[1…]`

Set or disconnect connections of CPU Devices (input) to/from CON Devices (output).
Input data of CPU Devices (Video, USB, Audio, …) will be transmitted to CON Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
K	1	Command	`0x4B`
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for ConCount = 3 `0x13 0x00`
ConCount	2	1…n = Number of CON-CPU Device connections	E.g., 3 = `0x03 0x00`
ConID	2	IDs of CON Devices	E.g., 1012 = `0xF4 0x03`, 1013 = `0xF5 0x03`, 1020 = `0xFC 0x03`
CpuID	2	IDs of CPU Devices	E.g., 3017 = `0xC9 0x0B`, 3028 = `0xD4 0x0B`, 3040 = `0xE0 0x0B`
		0 = Disconnect	`0x00 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x4B 0x13 0x00 0x03 0x00 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03 0xE0 0x0B 0xFC 0x03`

Set connections of CPU Devices to CON Devices, for example, the following devices:

ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[1] = 1013;
ConID[3] = 3040, CpuID[1] = 1020;

Request Example 2

Telegram: `0x1B 0x5B 0x4B 0x13 0x00 0x03 0x00 0xC9 0x0B 0xF4 0x03 0xD4 0x0B 0xF5 0x03 0xE0 0x0B 0x00 0x00`

Set connections of CPU Devices to CON Devices, for example, the following devices:

ConID[1] = 3017, CpuID[1] = 1012;
ConID[2] = 3028, CpuID[1] = 1013;
ConID[3] = 3040, CpuID[1] = 0, disconnect connection;

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.3.5 Set Connection of Single CPU EXT Units to Single CON EXT Units in Multi-Head Devices

Request

Telegram: `ESC [ l Size CpuID CpuExt ConID ConExt`

Set connection for EXT Unit (input) of a CPU Device to an EXT Unit (output) of a CON Device.
Input data of CPU EXT Unit (Video, USB, Audio, …) will be transmitted to CON EXT Unit.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
l	1	Command	`0x6C`
Size	2	Total length of telegram (13 bytes)	`0x0D 0x00`
CpuID	2	ID of a CPU Device	E.g., 1012 = `0xF4 0x03`
CpuExt	2	1…8 = Number of EXT Unit of the CPU Device to be switched	E.g., 4 = `0x04 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
ConExt	2	1…8 = Number of EXT Unit of the CON Device to be switched	E.g., 2 = `0x02 0x00`

Request Example

Telegram: `0x1B 0x5B 0x6C 0x0D 0x00 0xF4 0x03 0x04 0x00 0xC9 0x0B 0x02 0x00`

Set connection for EXT Unit (CpuExt = 4) of CPU Device (CpuID = 1012) to EXT Unit (ConExt = 2) of CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.4 Unidirectional Commands from CON Device to CPU Device

7.4.1 Get CON Device connected to CPU Device

Request

Telegram: `ESC [ L Size CpuID`

Get CON Device (input) connected to CPU Device (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
L	1	Command	0x4C
Size	2	Total length of telegram (7 bytes)	0x07 0x00
CpuID	2	ID of a CPU Device	E.g., 1012 = 0xF4 0x03

Request Example

Telegram: `0x1B 0x5B 0x4C 0x07 0x00 0xF4 0x03`

Get CON Device connected to CPU Device (CpuID = 1012).

This command will not work if a Video-only connection has been set (see chapter 6.2.2, page 31or chapter 7.3.2, page 47).

Response

Telegram: `ESC ] L Size CpuID ConID`

Return CON Device (input) connected to CPU Device (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
L	1	Command	0x4C
Size	2	Total length of telegram (9 bytes)	0x09 0x00
CpuID	2	ID of a CPU Device	E.g., 1012 = 0xF4 0x03
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B
		0 = No connection	0x00 0x00

Response Example 1

Telegram: `0x1B 0x5D 0x4C 0x09 0x00 0xF4 0x03 0xC9 0x0B`

Return CON Device (ConID = 3017) connected to CPU Device (CpuID = 1012).

Or <NAK>.

Response Example 2

Telegram: `0x1B 0x5D 0x4C 0x09 0x00 0xF4 0x03 0x00 0x00`

No CON Device (ConID = 0) connected to CPU Device (CpuID = 1012).

Or <NAK>.

7.4.2 Set Connection of CON Device to CPU Device

Request

Telegram: `ESC [ M Size CpuID ConID`

Set or disconnect CON Device (input) connection to/from CPU Device (output).
Input data of CON Device (USB, Audio) will be transmitted to CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
M	1	Command	`0x4D`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
CpuID	2	ID of a CPU Device	E.g., 1012 = `0xF4 0x03`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
		0 = Disconnect	`0x00 0x00`

Request Example for Connection

Telegram: `0x1B 0x5B 0x4D 0x09 0x00 0xF4 0x03 0xC9 0x0B`

Set CON Device (ConID = 3017) connection to CPU Device (CpuID = 1012).

Request Example for Disconnection

Telegram: `0x1B 0x5B 0x4D 0x09 0x00 0xF4 0x03 0x00 0x00`

Disconnect CON Device (ConID = 0) connection from CPU Device (CpuID = 1012).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.4.3 Get CON Devices connected to CPU Devices

Request

Telegram: `ESC [ N Size CpuCount CpuID[1…n]`

Get CON Devices (input) connected to specified or all CPU Devices (output).

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
N	1	Command	0x4E
Size	2	Total length of telegram (7 bytes + data = 13 bytes)	E.g., for CpuCount = 3 0x0D 0x00
CpuCount	2	0 = All CPU-CON Device connections will be returned	0x00 0x00
		1...n = Number of CPU-CON Device connections will be returned	E.g., 3 = 0x03 0x00
CpuID	2	IDs of CPU Devices (maximum 1024 entries)	E.g., 1012 = 0xF4 0x03

Request Example for Getting specified CON Devices

Telegram: `0x1B 0x5B 0x4E 0x0D 0x00 0x03 0x00 0xF4 0x03 0xF5 0x03 0xFC 0x03`

Get CON Devices connected to specified CPU Devices (CpuID = 1012, 1013, 1020).

Request Example for Getting all CON Devices

Telegram: `0x1B 0x5B 0x4E 0x07 0x00 0x00 0x00`

Get all CON Devices (ConIDs) connected to all CPU Devices (CpuCount = 0).

Response

Telegram: `ESC ] N Size CpuCount <CpuID ConID>[1…n]`

Return CON Devices (input) connected to CPU Devices (output) as pairs.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
N	1	Command	0x4E
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for CpuCount = 3 0x13 0x00
CpuCount	2	1…n = Number of CPU-CON Device connections (maximum 1024 connections)	E.g., 3 = 0x03 0x00
CpuID	2	IDs of CPU Devices	E.g., 1012 = 0xF4 0x03, 1013 = 0xF5 0x03, 1020 = 0xFC 0x03
ConID	2	IDs of CON Devices	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B
		0 = No connection	0x00 0x00

Response Example 1

Telegram: 0x1B 0x5D 0x4E 0x13 0x00 0x03 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0xE0 0x0B

Return CON Devices connected to CPU Devices, for example:

CpuID[1] = 1012, ConID[1] = 3017;
CpuID[2] = 1013, ConID[2] = 3028;
CpuID[3] = 1020, ConID[3] = 3040;

Or <NAK>.

Response Example 2

Telegram: 0x1B 0x5D 0x4E 0x13 0x00 0x03 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0x00 0x00

Return CON Devices connected to CPU Devices and CPU Devices without CON Device connection, for example:

CpuID[1] = 1012, ConID[1] = 3017;
CpuID[2] = 1013, ConID[2] = 3028;
CpuID[3] = 1020, ConID[3] = 0, no connection available;

Or <NAK>.

7.4.4 Set Connection of CON Devices to CPU Devices

Request

Telegram: `ESC [ O Size CpuCount[1…] <CpuID ConID>[1…n]`

Set or disconnect CON Devices (input) connection to/from CPU Devices (output).
Input data of CON Device (USB, Audio) will be transmitted to CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
O	1	Command	0x4F
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for CpuCount = 3 0x13 0x00
CpuCount	2	1…n = Number of CPU-CON Device connections	E.g., 3 = 0x03 0x00
CpuID	2	IDs of CPU Devices (maximum 1024 entries)	E.g., 1012 = 0xF4 0x03, 1013 = 0xF5 0x03, 1020 = 0xFC 0x03
ConID	2	IDs of CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B, 3028 = 0xD4 0x0B, 3040 = 0xE0 0x0B
		0 = Disconnect	0x00 0x00

Request Example 1

Telegram: `0x1B 0x5B 0x4F 0x13 0x00 0x03 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0xE0 0x0B`

Set connections of CON Devices to CPU Devices, e.g., the following devices:

CpuID[1] = 1012, ConID[1] = 3017;

CpuID[2] = 1013, ConID[2] = 3028;

CpuID[3] = 1020, ConID[3] = 3040;

Request Example 2

Telegram: `0x1B 0x5B 0x4F 0x13 0x00 0x03 0x00 0xF4 0x03 0xC9 0x0B 0xF5 0x03 0xD4 0x0B 0xFC 0x03 0x00 0x00`

Set connections of CON Devices to CPU Devices and disconnect CON Device connection from CPU Device, for example, the following devices:

CpuID[1] = 1012, ConID[1] = 3017;

CpuID[2] = 1013, ConID[2] = 3028;

CpuID[3] = 1020, ConID[3] = 0, disconnect connection;

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.4.5 Set USB HID Control in Multi-Screen Control Configuration

Firmware Requirement

MATAPP F04.00.20210521

Request

Telegram: `ESC [ w Size Module Port MConID SConID`

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
w	1	Command	`0x77`
Size	2	Total length of telegram (13 bytes)	`0x0D 0x00`
Module	2	1...254 = Slot number of an I/O board within a single matrix or global slot number within a Matrix Grid	E.g., slot number 4 = `0x04 0x00`
Port	2	1…1032 = Port number of the Control CON Device within a single matrix or global slot number within a Matrix Grid	E.g., port number 32 = `0x20 0x00`
MConID	2	ID of a Control CON Device within a MSC configuration with connected USB HID devices	E.g., 3006 = `0xBE 0x0B`
SConID	2	ID of a CON Device within a MSC configuration to which the USB HID control must be switched to	E.g., 3004 = `0xBC 0x0B`

Example 1 - Management Software with MSC Group

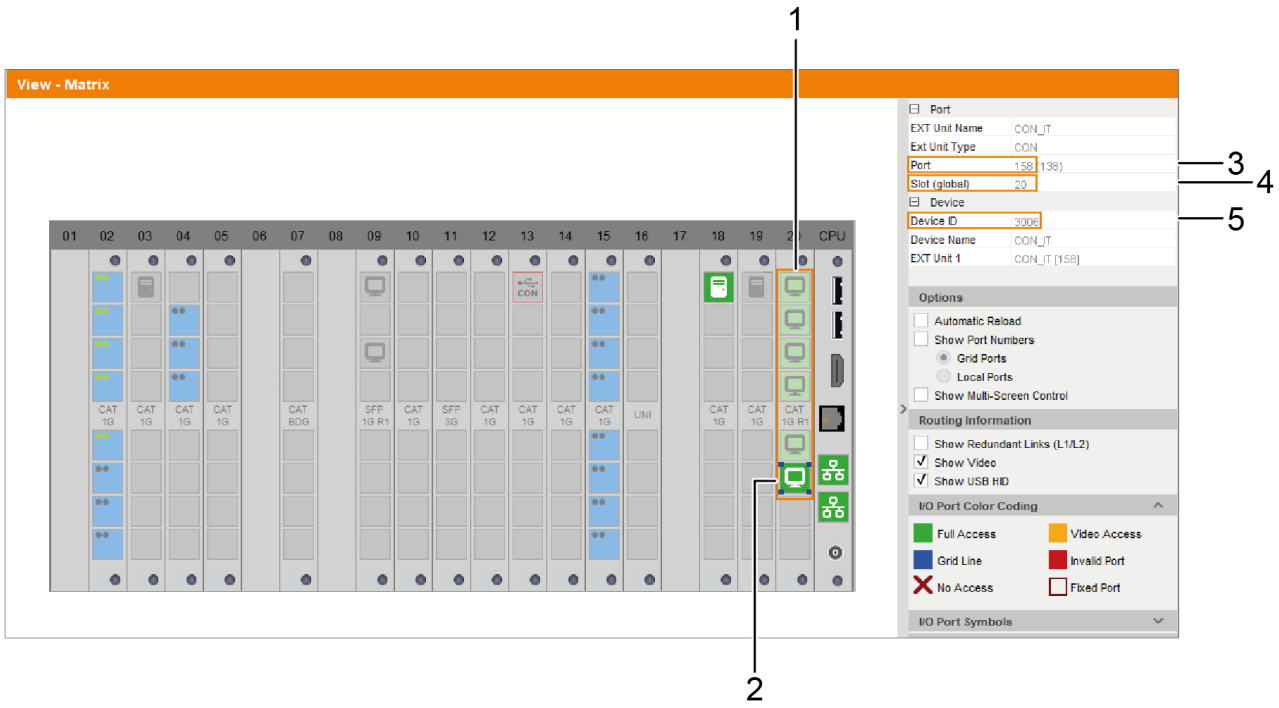


Fig. 13 Example 1 - Matrix without Grid configuration or Master matrix in a Grid environment

1 MSC Configuration with the following CON Devices 3001, 3002, 3003, 3004, 3005, 3006)
2 Control CON Device 3006 with connected USB HID devices (keyboard and mouse)
3 Port number of the Control CON Device
4 Slot number of the I/O board
5 Device Number of the Control CON Device

Example 2 - Management Software with MSC Group

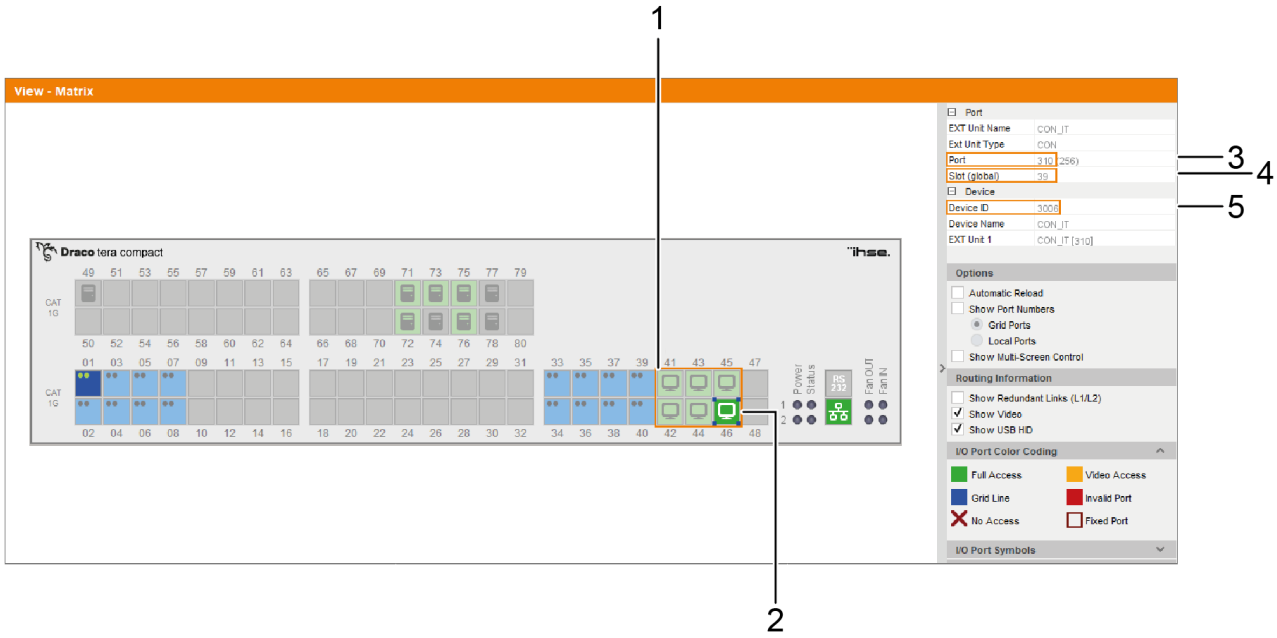


Fig. 14 Example 2 - Matrix without Grid configuration or Master matrix in a Grid environment

1 MSC Configuration with the following CON Devices 3001, 3002, 3003, 3004, 3005, 3006)
2 Control CON Device 3006 with connected USB HID devices (keyboard and mouse)
3 Port number of the Control CON Device
4 Slot number of the I/O board
5 Device Number of the Control CON Device

Request Example with Firmware MATAPP F04.00.20210521

Telegram: `0x1B 0x5B 0x77 0x0D 0x00 0x04 0x00 0x20 0x00 0xBE 0xC0 0xBC 0x0B`

Switch USB HID control within the MSC configuration of the I/O board plugged in slot number 4 (Module = 4). Change USB HID control from Control CON Device (MConID = 3006) connected to port number 32 (Port = 32) to another CON Device (SConID = 3004).

Request Example with Firmware MATAPP F04.01.20220726

Changes with firmware MATAPP F04.01.20220726: The data of the I/O board index and the port number of the Control CON Device can also be omitted and is then simply specified with zero.

Telegram: `0x1B 0x5B 0x77 0x0D 0x00 0x00 0x00 0x00 0x00 0xBE 0xC0 0xBC 0x0B`

Switch USB HID control within the MSC configuration of the I/O board plugged in slot number 4 (Module = 4). Change USB HID control from Control CON Device (MConID = 3006) connected to port number 32 (Port = 32) to another CON Device (SConID = 3004).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.5 Unidirectional Commands for Ports

7.5.1 Get Input Port connected to Output Port

Request

Telegram: `ESC [ B Size OutPort`

Get input port connected to output port.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
B	1	Command	0x42
Size	2	Total length of telegram (7 bytes)	0x07 0x00
OutPort	2	1…2032 = Output port number	E.g., 5 = 0x05 0x00

Request Example

Telegram: `0x1B 0x5B 0x42 0x07 0x00 0x05 0x00`

Get input port number connected to output port number (OutPort = 5).

Response

Telegram: `ESC ] B Size OutPort InPort`

Return input port connected to output port.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
B	1	Command	0x42
Size	2	Total length of telegram (9 bytes)	0x09 0x00
OutPort	2	1…2032 = Output port number	E.g., 5 = 0x05 0x00
InPort	2	0 = Output port without connection	0x00 0x00
		1…2032 = Input port number whose input signal is output to OutPort	E.g., 3 = 0x03 0x00

Response Example 1

Telegram: `0x1B 0x5D 0x42 0x09 0x00 0x05 0x00 0x03 0x00`

Return input port number (InPort = 3) connected to output port number (OutPort = 5).

Or <NAK>

Response Example 2

Telegram: `0x1B 0x5D 0x42 0x09 0x00 0x05 0x00 0x00 0x00`

No input port (InPort = 0) connected to output port (OutPort = 5).

Or <NAK>

7.5.2 Set Connection of Input Port to Output Port

Request

Telegram: `ESC [ F Size OutPort InPort`

Set connection of input port to output port.

Data of input port will be transmitted to output port.

To set the connection, a CON Device has to be configured at the output port, and a CPU Device has to be configured to the input port. If no CPU Device is found, an existing connection will be disconnected.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
F	1	Command	`0x46`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
OutPort	2	1…2032 = Output port number (CON Device)	E.g., 10 = `0x0A 0x00`
InPort	2	1…2032 = Input port number (CPU Device) whose input signal is output to OutPort (CON Device)	E.g., 20 = `0x14 0x00`
		0 = Disconnect port	`0x00 0x00`

Request Example for Connecting

Telegram: `0x1B 0x5B 0x46 0x09 0x00 0x0A 0x00 0x14 0x00`

Set connection from output port number (OutPort = 10) to input port number (InPort = 20).

Request Example for Disconnecting

Telegram: `0x1B 0x5B 0x46 0x09 0x00 0x0A 0x00 0x00 0x00`

Disconnect input port (InPort = 0) from output port (OutPort = 10).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.5.3 Get Input Ports connected to Output Ports

Request

Telegram: `ESC [ D Size PortCount OutPort[1…n]`

Get input ports connected to specified or all output ports.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
D	1	Command	0x44
Size	2	Total length of telegram (7 bytes + data = 13 bytes)	E.g., for PortCount = 3 0x0D 0x00
PortCount	2	0 = All port connections will be returned	0x00 0x00
		1…n = Number of port connections will be returned	E.g., 3 = 0x03 0x00
OutPort	2	1…2032 = Output port numbers	E.g., 2 = 0x02 0x00, 4 = 0x04 0x00, 5 = 0x05 0x00

Request Example for Getting specified Input Ports

Telegram: `0x1B 0x5B 0x44 0x0D 0x00 0x03 0x00 0x02 0x00 0x04 0x00 0x05 0x00`

Get input ports connected to specified output ports (PortCount = 3, OutPort = 2, 4, 5)

Request Example for Getting all Input Ports

Telegram: `0x1B 0x5B 0x44 0x07 0x00 0x00 0x00`

Get all input ports connected to all output ports (PortCount= 0).

Response

Telegram: `ESC ] D Size PortCount <OutPort InPort>[1…n]`

Return input ports connected to output ports as pairs.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
D	1	Command	0x44
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for PortCount = 3 0x13 0x00
PortCount	2	0 = All port connections will be returned	0x00 0x00
		1…n = Number of port connections will be returned	E.g., 3 = 0x03 0x00
OutPort	2	1…2032 = Output port numbers	E.g., 2 = 0x02 0x00, 4 = 0x04 0x00, 5 = 0x05 0x00
InPort	2	0 = Output port without connection	0x00 0x00
		1…2032 = Input port numbers whose input signals are output to output port numbers	E.g., 8 = 0x08 0x00, 10 = 0x0A 0x00, 12 = 0x0C 0x00

Response Example 1

Telegram: 0x1B 0x5D 0x44 0x13 0x00 0x03 0x00 0x02 0x00 0x08 0x00 0x04 0x00 0x0A
0x00 0x05 0x00 0x0C 0x00

Return input port numbers (InPort) connected to output port number (OutPort), for example:

OutPort[1] = 2, InPort[1] = 8;
OutPort[2] = 4, InPort[2] = 10;
OutPort[3] = 5, InPort[3] = 12;

Or <NAK>.

Response Example 2

Telegram: 0x1B 0x5D 0x44 0x13 0x00 0x03 0x00 0x02 0x00 0x08 0x00 0x04 0x00 0x0A
0x00 0x05 0x00 0x00 0x00

Return input port numbers (InPort) connected to output port number (OutPort), for example:

OutPort[1] = 2, InPort[1] = 8;
OutPort[2] = 4, InPort[2] = 10;
OutPort[3] = 5, InPort[3] = 0, no connection available

Or <NAK>.

7.5.4 Set Connection of Input Ports to Output Ports

Request

Telegram: `ESC [ G Size PortCount OutPort[1…n] InPort[1…n]`

Set connection of input port to output port.

Data of input ports will be transmitted to output ports.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
G	1	Command	0x47
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for PortCount = 3 0x13 0x00
PortCount	2	0 = All port connections will be returned	0x00 0x00
		1…n = Number of port connections will be returned	E.g., 3 = 0x03 0x00
OutPort	2	1…2032 = Output port numbers	E.g., 2 = 0x02 0x00, 4 = 0x04 0x00, 5 = 0x05 0x00
InPort	2	1…2032 = Input port numbers whose input signals are output to output port numbers	E.g., 8 = 0x08 0x00, 10 = 0x0A 0x00, 12 = 0x0C 0x00
		0 = Disconnect connection	0x00 0x00

Request Example 1

Telegram: `0x1B 0x5D 0x47 0x13 0x00 0x03 0x00 0x02 0x00 0x08 0x00 0x04 0x00 0x0A 0x00 0x05 0x00 0x0C 0x00`

Set a connection for all defined input port numbers to output port numbers:

OutPort[1] = 2, InPort[1] = 8;
OutPort[2] = 4, InPort[2] = 10;
OutPort[3] = 5, InPort[3] = 12;

Request Example 2

Telegram: `0x1B 0x5D 0x47 0x13 0x00 0x03 0x00 0x02 0x00 0x08 0x00 0x04 0x00 0x0A 0x00 0x05 0x00 0x00 0x00`

Set a connection for two defined input port numbers (InPort) to output port numbers (OutPort), for example:

OutPort[1] = 2, InPort[1] = 8;
OutPort[2] = 4, InPort[2] = 10;
OutPort[3] = 5, InPort[3] = 0, no connection available

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.6 Get Commands for Lists and Status

7.6.1 Get CPU Device List

Request

Telegram: `ESC [ g Size First`

Get list of CPU Devices including ID, name, and online status.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
g	1	Command	`0x67`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
First	2	0 = List all CPU Devices (recommended for first request)	`0x00 0x00`
		1...n = Index of CPU Device from which the list scan will start	E.g., 3 = `0x03 0x00`

Request Example for CPU Device List from Index

Telegram: `0x1B 0x5B 0x67 0x07 0x00 0x03 0x00`

Get list of CPU Devices starting from the third CPU Device (First = 3).

Request Example for all CPU Devices

Telegram: `0x1B 0x5B 0x67 0x07 0x00 0x00 0x00`

Get list of all CPU Devices (First = 0).

Response

Telegram: `ESC ] g Size Count Next <ID Name Status Info>[1…n]`

Return list of all or specified CPU Devices including ID, name, and status.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
g	1	Command	`0x67`
Size	2	Total length of telegram (7 bytes + data = 33 bytes)	E.g., 33 bytes for Count = 1 `0x21 0x00`
Count	2	1…n = Number of CPU Devices in the list (max. 256 entries will be returned)	E.g., 1 = `0x01 0x00`
Next	2	ID of first CPU Device in next list	E.g., 0 = `0x00 0x00` (no further CPU Device)
ID	4	IDs of CPU Devices	E.g., 1000 = `0xE8 0x03 0x00 0x00`
Name	17	Name of CPU Device	E.g., CPU_Video1 = 0x43 0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Type	Bytes	Description	Hex coding
Status	1	Status of CPU Device	
		0 = Offline	0x00
		1 = Online	0x01
		2 = Private Mode and offline	0x02
		3 = Private Mode and online	0x03
Info	2	Empty bytes	0x00 0x00

Response Example

Telegram: 0x1B 0x5D 0x67 0x21 0x00 0x01 0x00 0x00 0x00 0xE8 0x03 0x00 0x00 0x43
0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00

Return list of one CPU Device including ID, name, and status.

Or <NAK>.

7.6.2 Get CON Device List

Request

Telegram: `ESC [ h Size First`

Get list of CON Devices including ID, name, and online status.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
h	1	Command	`0x68`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
First	2	0 = List all CON Devices (recommended for first request)	`0x00 0x00`
		1...n = Index of CON Device from which the list scan will start	E.g., 5 = `0x05 0x00`

Request Example for CON Device List from Index

Telegram: `0x1B 0x5B 0x68 0x07 0x00 0x05 0x00`

Get list of CON Devices starting from the fifth CON Device (First = 5).

Request Example for all CON Devices

Telegram: `0x1B 0x5B 0x68 0x07 0x00 0x00 0x00`

Get list of all CON Devices (First = 0).

Response

Telegram: `ESC ] h Size Count Next <ID Name Status Info>[1…n]`

Return list of all or specified CON Devices including ID, name, status, and logged in user.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
h	1	Command	`0x68`
Size	2	Total length of telegram (7 bytes + data = 33 bytes)	E.g., 33 bytes for Count = 1 `0x21 0x00`
Count	2	1…n = Number of CON Devices in the list (max. 256 entries)	E.g., 1 = `0x01 0x00`
Next	2	ID of first CON Device in next list	E.g., 0 = `0x00 0x00` (no further CON Device)
ID	4	IDs of CON Devices	E.g., 3000 = `0xB8 0x0B 0x00 0x00`
Name	17	Name of CON Device	E.g., CON_Video1 = 0x43 0x4F 0x4E 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Type	Bytes	Description	Hex coding
Status	1	Status of CON Device	
		0 = Offline	0x00
		1 = Online	0x01
		2 = Private Mode and offline	0x02
		3 = Private Mode and online	0x03
		4 = Video-only and offline	0x04
		5 = Video-only and online	0x05
Info	2	Info about logged-in user	E.g., user with ID = 1 =0x01 0x00

Response Example

Telegram: 0x1B 0x5D 0x68 0x21 0x00 0x01 0x00 0x00 0x00 0xB8 0x0B 0x00 0x00 0x43
0x4F 0x4E 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x01 0x00

Return list of one CON Device including ID (ID = 3000), name (Name = CON_Video1), and online status (Status = 1) with logged in user (Info = 1).

Or <NAK>.

7.6.3 Get User List

Request

Telegram: `ESC [ i Size First`

Get list of users including ID and name.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
i	1	Command	`0x69`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
First	2	0 = List all users (recommended for first request)	`0x00 0x00`
		1...n = Index of user from whom the list scan will start	E.g., 1 = `0x01 0x00`

Request Example for Listing Users starting from Index

Telegram: `0x1B 0x5B 0x69 0x07 0x00 0x01 0x00`

Get list of users starting from the first user (First = 1).

Request Example for Listing all Users

Telegram: `0x1B 0x5B 0x69 0x07 0x00 0x00 0x00`

Get list of all users (First = 0).

Response

Telegram: `ESC ] i Size Count Next <ID Name Info>[1…n]`

Return list of all or specified users including ID and name.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
i	1	Command	`0x69`
Size	2	Total length of telegram (7 bytes + data = 33 bytes)	E.g., 33 bytes for Count = 1 `0x21 0x00`
Count	2	Number of users in the list (max. 256 entries)	E.g., 1 = `0x01 0x00`
Next	2	ID of first user in next list	E.g., 0 = `0x00 0x00` (no further user)
ID	4	ID of CPU Devices	E.g., 1 = `0x01 0x00 0x00 0x00`
Name	17	Name of CPU Device	E.g., admin = 0x61 0x64 0x6D 0x69 0x6E 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Info	3	Empty bytes	`0x00 0x00 0x00`

Response Example

Telegram: 0x1B 0x5D 0x69 0x21 0x00 0x01 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x61
0x64 0x6D 0x69 0x6E 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00

Return list of one user including ID and name.

Or <NAK>.

7.6.4 Get Active KVM Link of CON Device

Request

Telegram: `ESC [ m Size ConID`

Get active KVM link of CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
m	1	Command	0x6D
Size	2	Total length of telegram (7 bytes)	0x07 0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B

Request Example for CPU Device List from Index

Telegram: `0x1B 0x5B 0x6D 0x07 0x00 0xC9 0x0B`

Get active KVM link of a CON Device (ConID = 3017).

Response

Telegram: `ESC ] m Size ConID KVM`

Return active KVM link for a CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
m	1	Command	0x6D
Size	2	Total length of telegram (9 bytes)	0x09 0x00
ConID	2	ID of a CON Device	E.g., 3017 = 0xC9 0x0B
KVM	2	Active KVM link of the CON Device	1 = Primary link = 0x01 0x00
			2 = Secondary link = 0x02 0x00
			3 = Local CPU = 0x03 0x00

Response Example

Telegram: `0x1B 0x5D 0x6D 0x07 0x00 0xC9 0x0B 0x01 0x00`

Return list with the CON Device (ConID = 3017) with active primary KVM link.

Or <NAK>.

Only the first assigned EXT Unit of a CON Device can be queried.

7.6.5 Get Active KVM Link List of CON Devices

Request

Telegram: `ESC [ t Size First`

Get active KVM link list of CON Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
t	1	Command	`0x74`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
First	2	0 = List all CON Devices (recommended for first request)	`0x00 0x00`
		1...n = Index of CON Device from which the list scan will start	E.g., 5 = `0x05 0x00`

Request Example for Active KVM Link List from Index

Telegram: `0x1B 0x5B 0x74 0x07 0x00 0x03 0x00`

Get active KVM link list of CON Device starting from the fifth CON Device (First = 5).

Request Example for Active KVM Link List of all CPU Devices

Telegram: `0x1B 0x5B 0x74 0x07 0x00 0x00 0x00`

Get active KVM link list of all CON Devices (First = 0).

Response

Telegram: `ESC ] t Size Count Next <ConID KVM>[1…n]`

Return active KVM link for all or specified CON Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
t	1	Command	`0x74`
Size	2	Total length of telegram (9 bytes + data = 17 bytes)	E.g., 17 bytes for Count = 1 `0x11 0x00`
Count	2	Number of CON Devices in the list	E.g., 2 = `0x02 0x00`
Next	2	ID of first CON Device in next list (max. 256 entries)	E.g., 0 = `0x00 0x00` (no further CON Device)
ConID	2	IDs of CON Devices	E.g., 3017 = `0xC9 0x0B`, 3018 = `0xCA 0x0B`
KVM	2	Active KVM link port of the CON Device	1 = Primary link port = `0x01 0x00`
			2 = Secondary link port = `0x02 0x00`
			3 = Local input port = `0x03 0x00`

Response Example 1

Telegram: `0x1B 0x5D 0x74 0x11 0x00 0x02 0x00 0x00 0x00 0xC9 0x0B 0x02 0x00 0xCA 0x0B 0x01 0x00`

Return list with one CPU Device (ConID = 3017) with active primary KVM link (KVM = 1) and a second CPU Device (ConID = 3018) with active secondary KVM link (KVM = 2).

Or <NAK>.

Response Example 2

Telegram: `0x1B 0x5D 0x74 0x0D 0x00 0x01 0x00 0x00 0x00 0xC9 0x0B 0x01 0x00`

Return the CON Device (ConID = 3017) with active primary KVM link (KVM = 1).

Or <NAK>.

Only the first assigned EXT Unit of a CON Device can be queried.

7.6.6 Get Multi-Screen Control Configuration List

Request

Telegram: `ESC [ u Size First`

Get list of MSC configurations available on I/O boards.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
u	1	Command	`0x75`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
First	2	0 = List all MSC configurations (recommended for first request)	`0x00 0x00`
		1...2031 = Index of port from which the list scan will start	E.g., 3 = `0x03 0x00`

Request Example for MSC Configuration List from Index

Telegram: `0x1B 0x5B 0x75 0x07 0x00 0x03 0x00`

Get list of MSC configurations starting from the third I/O board (First = 3).

Request Example for all MSC Configurations

Telegram: `0x1B 0x5B 0x75 0x07 0x00 0x03 0x00`

Get list of all MSC configurations (First = 0).

Response

Telegram: `ESC ] u Size Count Next Module Port ConID Status Owner`

Return list of all or specified CON Devices including ID, name, and online status.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
u	1	Command	`0x75`
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., 19 bytes for Count = 1 `0x13 0x00`
Count	2	Number of MSC configurations in the list (maximum 64 entries)	E.g., 1 = `0x01 0x00`
Next	2	ID of first MSC configuration in next list	E.g., 0 = `0x00 0x00` (no further CON Device)
Module	2	1...254 = Slot number of an I/O board within a single matrix or global slot number within a Matrix Grid	E.g., slot number 4 = `0x04 0x00`
Port	2	1…1032 = Port numbers of the Control CON Device within a single matrix or global slot number within a Matrix Grid	E.g., port number 32 = `0x20 0x00`
ConID	2	IDs of CON Devices within a MSC configuration	E.g., 3006 = `0xBE 0x0B`

Type	Bytes	Description	Hex coding
Status	2	1 = Control CON Device with connected USB HID devices	0x01 0x00
		0 = Control CON Device without connected USB HID devices	0x00 0x00
Owner		ID of a CON Device if exclusive	0x01 0x00

Response Example

Telegram: 0x1B 0x5D 0x75 0x13 0x00 0x01 0x00 0x00 0x00 0x04 0x00 0x20 0x00 0xBE 0x00 0x01 0x00 0xBE 0x0B

Return list with one MSC configuration.

Or <NAK>.

7.6.7 Get Status Information

Command available from FW 4.0_20201201

Request

Telegram: `ESC [ y Size Type ID`

Get list of specified system data.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
y	1	Command	0x79
Size	2	Total length of telegram (11 bytes)	0x0B 0x00
Type	2	Data range of matrix components	
		1 = System data	0x01 0x00
		2 = Module data (I/O board)	0x02 0x00
		3 = Port data	0x03 0x00
		4 = EXT Unit data	0x04 0x00
		5 = CPU Device data	0x05 0x00
		6 = CON Device data	0x06 0x00
		7 = User data	0x07 0x00
		8 = Extended CON Device data (since FW 4.0 2021-05-21)	0x08 0x00
ID	4	ID of matrix component	E.g., 40172072 = 0x28 0xFA 0x64 0x02
		In case of system data, enter 0x00 0x00 0x00 0x00	

Request Example

Telegram: `0x1B 0x5B 0x79 0x0B 0x00 0x04 0x00 0x28 0xFA 0x64 0x02`

Get information of EXT Unit data (Type = 4, ID = 40172072).

Response

Telegram: `ESC ] y Size Type ID Error Data`

Data contains extended information about the status of the requested matrix component.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
y	1	Command	`0x79`
Size	2	Total length of telegram (12 bytes + data)	E.g., 60 bytes = `0x3C 0x00`
Type	2	Type of matrix component	E.g., EXT Unit = `0x04 0x00`
ID	4	ID of matrix component	E.g., 40172072 = `0x28 0xFA 0x64 0x02`
Error	1	Error number 0 = Without error = `0x00` 1 = Invalid type = `0x01` 2 = Invalid ID = `0x02`	E.g., Valid request = 0 = `0x00`
Data*	20 to 172	All available data regarding the requested matrix component	...

* Data size, see table below.
Data content, see details in the respective chapter for response definition.

Response Example for valid Request

Telegram:
```
0x1B 0x5B 0x79 0x3C 0x00 0x04 0x00 0x28 0xFA 0x64 0x02 0x00 0x28 0xFA
0x64 0x02 0x01 0x00 0x0C 0x80 0x02 0x00 0x01 0x00 0x45 0x58 0x54 0x5F
0x30 0x34 0x30 0x31 0x37 0x32 0x30 0x37 0x32 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x09 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0xBC 0x0B 0x00 0x00
```

Response Example for invalid Request

Telegram:
```
0x1B 0x5B 0x79 0x0C 0x00 0x04 0x00 0x28 0xFA 0x64 0x02 0x01
```

Sizes of Response Telegram and Data

The size of the telegrams depend on the requested matrix component, in every case multiples of 4.

Type	Component	Telegram size	Data size
1	System data	124 bytes	112 bytes
2	Module data (I/O board)	52 bytes	40 bytes
3	Port data	32 bytes	20 bytes
4	EXT Unit data	60 bytes	48 bytes
5	CPU Device data	144 bytes	132 bytes
6	CON Device data	152 bytes	140 bytes
7	User data	52 bytes	40 bytes
8	Extended CON Device data (since FW 4.0 2021-05-21)	184 bytes	172 bytes

7.6.7.1 System Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error Data <Status Name Info Device>`

Type	Bytes	Description	Hex coding
Status	4	System status bits	0x01 0x00 0x00 0x04
Name	16 + 4	Name of the configuration	E.g., Master Matrix = 0x4d 0x61 0x73 0x74 0x65 0x72 0x20 0x4d 0x61 0x74 0x72 0x69 0x78 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Info	64 + 4	Short description	E.g., Grid Master = 0x47 0x72 0x69 0x64 0x20 0x4d 0x61 0x73 0x74 0x65 0x72 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Device	16 + 4	Hostname for network	E.g., KVM_M_M = 0x4b 0x56 0x4d 0x5f 0x4d 0x5f 0x4d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Response Example

Telegram:
```
0x1B 0x5D 0x79 0x7C 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x01 0x00
0x00 0x04 0x4d 0x61 0x73 0x74 0x65 0x72 0x20 0x4d 0x61 0x74 0x72 0x69
0x78 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x47 0x72 0x69 0x64 0x20 0x4d
0x61 0x73 0x74 0x65 0x72 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x4b 0x56 0x4d 0x5f 0x4d 0x5f 0x4d 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

Definition of System Status Bits

The binary interpretation of the received system status bytes `0x01 0x00 0x00 0x04` sent in little endian byte order is 00000100 00000000 00000000 00000001.

Bit by bit

0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

User configuration loaded, no default configuration.

Primary controller board of a matrix with redundant controller boards.

System Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	User configuration
02	0x00000002	00000000 00000000 00000000 00000010	Chinese OSD translation
21	0x00100000	00000000 00010000 00000000 00000000	Controller board is on hold because of invalid firmware
22	0x00200000	00000000 00100000 00000000 00000000	Redundant controller board - Second controller board available
23	0x00400000	00000000 01000000 00000000 00000000	576 only - Master active
24	0x00800000	00000000 10000000 00000000 00000000	576 only - Sub active
25	0x01000000	00000001 00000000 00000000 00000000	576 only - Master controller board
26	0x02000000	00000010 00000000 00000000 00000000	576 only - Sub controller board
27	0x04000000	00000100 00000000 00000000 00000000	Redundant controller board - Primary controller board
28	0x08000000	00001000 00000000 00000000 00000000	Redundant controller board - Secondary controller board

7.6.7.2 Module Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error Data <ID Status Version>`

Type	Bytes	Description	Hex coding
ID	4	1...254 = Slot number of the I/O board	E.g., 36 = `0x24 0x00 0x00 0x00`
Status	4	Module status bits	`0x4F 0x00 0x00 0x00`
Version	32	Firmware version of the I/O board	E.g., MATXCAT IO8 8 F04.00 12.11.20 = `0x4D 0x41 0x54 0x58 0x43 0x41 0x54 0x20 0x20 0x49 0x4F 0x38 0x20 0x38 0x20 0x46 0x30 0x34 0x2E 0x30 0x30 0x20 0x31 0x32 0x2E 0x31 0x31 0x2E 0x32 0x30 0x00 0x00`

Response Example

Telegram: `0x1B 0x5D 0x79 0x34 0x00 0x02 0x00 0x24 0x00 0x00 0x00 0x00 0x24 0x00 0x00 0x00 0x4F 0x00 0x00 0x00 0x4D 0x41 0x54 0x58 0x43 0x41 0x54 0x20 0x20 0x49 0x4F 0x38 0x20 0x38 0x20 0x46 0x30 0x34 0x2E 0x30 0x30 0x20 0x31 0x32 0x2E 0x31 0x31 0x2E 0x32 0x30 0x00 0x00`

Definition of Module Status Bits

The binary interpretation of the received module status bytes `0x4F 0x00 0x00 0x00` sent in little endian byte order is 00000000 00000000 00000000 01001111.

Bit by bit

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

The I/O board is connected, online, configured, active and supports high speed rate.

Module Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	Module is available.
02	0x00000002	00000000 00000000 00000000 00000010	Module is online.
03	0x00000004	00000000 00000000 00000000 00000100	Module is configured.
04	0x00000008	00000000 00000000 00000000 00001000	Module is active.
05	0x00000010	00000000 00000000 00000000 00010000	Module is running in service mode.
07	0x00000040	00000000 00000000 00000000 01000000	Module supports high baud rate.
08	0x00000080	00000000 00000000 00000000 10000000	Module has activated trace function.
27	0x04000000	00000100 00000000 00000000 00000000	Module has an error.
31	0x40000000	01000000 00000000 00000000 00000000	Module is stopped.
32	0x80000000	10000000 00000000 00000000 00000000	Module has invalid firmware.

7.6.7.3 Port Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error Data <ID Status Type Ext Output>`

Type	Bytes	Description	Hex coding
ID	4	1…2032 = Port number	E.g., 36 = `0x24 0x00 0x00 0x00`
Status	4	Status register	`0x1F 0x28 0x00 0x00`
Type	4	Extender module type respectively port number	`0x02 0x00 0x01 0x00`
Ext	4	ID of EXT Unit respectively IP address	E.g., 561151012 = `0x38 0x73 0x65 0x02`
Output	4	1…2032 = Port number of the port whose signal is output	E.g., 5 = `0x05 0x00 0x00 0x00`

Response Example

Telegram: `0x1B 0x5D 0x79 0x20 0x00 0x03 0x00 0x24 0x00 0x00 0x00 0x00 0x24 0x00 0x00 0x00 0x1F 0x28 0x00 0x00 0x02 0x00 0x01 0x00 0x38 0x73 0x65 0x02 0x05 0x00 0x00 0x00`

Definition of Port Status Bits

The binary interpretation of the received port status bytes `0x1F 0x28 0x00 0x00` sent in little endian byte order is 00000000 00000000 00101000 00011111.

Bit by bit

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	1	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

This port is connected to a powered extender module and fully functional.
Switching to link 2 or local possible.

Port Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	Port is available.
02	0x00000002	00000000 00000000 00000000 00000010	Initialization is completed.
03	0x00000004	00000000 00000000 00000000 00000100	Tx is enabled.
04	0x00000008	00000000 00000000 00000000 00001000	Rx is enabled.
05	0x00000010	00000000 00000000 00000000 00010000	Port is connected.
12	0x00000800	00000000 00000000 00001000 00000000	Possibility of switching to link 1, link 2, or local
13	0x00001000	00000000 00000000 00010000 00000000	Port is connected to link 2 of an extender module.
14	0x00002000	00000000 00000000 00100000 00000000	Port is connected to an extender module.
15	0x00004000	00000000 00000000 01000000 00000000	Port is connected to a matrix.
16	0x00008000	00000000 00000000 10000000 00000000	Port is invalid.

7.6.7.4 EXT Unit Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error Data <ID Status Type Name PPort SPort Cpu Con>`

Type	Bytes	Description	Hex coding
ID	4	ID of EXT Unit	E.g., 40172072 = `0x28 0xFA 0x64 0x02`
Status	4	Status register	`0x01 0x00 0x0C 0x80`
Type	4	Extender module type	`0x02 0x00 0x01 0x00`
Name	16 + 4	Name of EXT Unit	E.g., EXT_040172072 = `0x45 0x58 0x54 0x5F 0x30 0x34 0x30 0x31 0x37 0x32 0x30 0x37 0x32 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
PPort	4	1…2032 = Matrix port number of the primary EXT Unit port	E.g., 9 = `0x09 0x00 0x00 0x00`
SPort	4	1…2032 = Matrix port number of the secondary EXT Unit port	`0x00 0x00 0x00 0x00`
Cpu	4	ID of CPU Device assigned to this EXT Unit	`0x00 0x00 0x00 0x00`
Con	4	ID of CON Device assigned to this EXT Unit	E.g., 3017 = `0xC9 0x0B 0x00 0x00`

Response Example

Telegram: `0x1B 0x5D 0x79 0x3C 0x00 0x04 0x00 0x28 0xFA 0x64 0x02 0x00 0x28 0xFA 0x64 0x02 0x01 0x00 0x0C 0x80 0x02 0x00 0x01 0x00 0x45 0x58 0x54 0x5F 0x30 0x34 0x30 0x31 0x37 0x32 0x30 0x37 0x32 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x09 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xC9 0x0B 0x00 0x00`

Get EXT Unit (Type = 4) data of a redundant CON extender module which is currently working on link 2. The CON extender module contains a video output, a USB HID input, and a USB 2.0 embedded input.

Definition of EXT Unit Status Bits

The binary interpretation of the received EXT Unit status bytes `0x01 0x00 0x0C 0x80` sent in little endian byte order is 10000000 00001100 00000000 00000001.

Bit by bit

1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

Definition of EXT Unit Type Bits

The binary interpretation of the received EXT Unit type bytes `0x12 0x00 0x01 0x00` sent in little endian byte order is 00000000 00000001 0000000 00010010.

Bit by bit

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

EXT Unit Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	Extender module is connected and powered on.
02	0x00000002	00000000 00000000 00000000 00000010	Extender modules support HDCP.
03	0x00000004	00000000 00000000 00000000 00000100	Extender modules have active HDCP.
04	0x00000008	00000000 00000000 00000000 00001000	Extender modules communicates with SNMP components.
17	0x00010000	00000000 00000001 00000000 00000000	Extender module is connected to fix port.
18	0x00020000	00000000 00000010 00000000 00000000	Extender module is connected to UNI board.
19	0x00040000	00000000 00000100 00000000 00000000	Redundant extender module
20	0x00080000	00000000 00001000 00000000 00000000	Extender module is internally switched to KVM link 2 (only for CON extender modules).
21	0x00100000	00000000 00010000 00000000 00000000	Extender module is internally switched to local KVM link (only for CON extender modules).
32	0x80000000	10000000 00000000 00000000 00000000	Extender module is configured properly.

EXT Unit Type Definition

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	Extender module with video input
02	0x00000002	00000000 00000000 00000000 00000010	Extender module with USB HID input
03	0x00000004	00000000 00000000 00000000 00000100	Extender module with audio input
04	0x00000008	00000000 00000000 00000000 00001000	Extender module with serial input
05	0x00000010	00000000 00000000 00000000 00010000	Extender module with input USB 2.0 embedded
06	0x00000020	00000000 00000000 00000000 00100000	Extender module with input USB 2.0 stand-alone
07	0x00000040	00000000 00000000 00000000 01000000	Extender module with universal input
08	0x00000080	00000000 00000000 00000000 10000000	Extender module with cascaded input
09	0x00000100	00000000 00000000 00000001 00000000	Extender module with second video input
10	0x00000200	00000000 00000000 00000010 00000000	Extender module with second USB HID input
11	0x00000400	00000000 00000000 00000100 00000000	Extender module with second audio input
12	0x00000800	00000000 00000000 00001000 00000000	Extender module with second serial input
13	0x00001000	00000000 00000000 00010000 00000000	Extender module with second input USB 2.0 embedded
14	0x00002000	00000000 00000000 00100000 00000000	Extender module with second input USB 2.0 stand-alone
15	0x00004000	00000000 00000000 01000000 00000000	Extender module with second universal input
16	0x00008000	00000000 00000000 10000000 00000000	Extender module with second cascaded input
17	0x00010000	00000000 00000001 00000000 00000000	Extender module with video output
18	0x00020000	00000000 00000010 00000000 00000000	Extender module with USB HID output
19	0x00040000	00000000 00000100 00000000 00000000	Extender module with audio output
20	0x00080000	00000000 00001000 00000000 00000000	Extender module with serial output
21	0x00100000	00000000 00010000 00000000 00000000	Extender module with output USB 2.0 embedded
22	0x00200000	00000000 00100000 00000000 00000000	Extender module with output USB 2.0 stand-alone
23	0x00400000	00000000 01000000 00000000 00000000	Extender module with universal output
24	0x00800000	00000000 10000000 00000000 00000000	Extender module with cascaded output
25	0x01000000	00000001 00000000 00000000 00000000	Extender module with second video output

Bit	Bits in hexadecimal	Binary bits	Description
26	0x02000000	00000010 00000000 00000000 00000000	Extender module with second USB HID output
27	0x04000000	00000100 00000000 00000000 00000000	Extender module with second audio output
28	0x08000000	00001000 00000000 00000000 00000000	Extender module with second serial output
29	0x10000000	00010000 00000000 00000000 00000000	Extender module with second output USB 2.0 embedded
30	0x20000000	00100000 00000000 00000000 00000000	Extender module with second output USB 2.0 stand-alone
31	0x40000000	01000000 00000000 00000000 00000000	Extender module with second universal output
32	0x80000000	10000000 00000000 00000000 00000000	Extender module with second cascaded output

Definition Example of EXT Unit Type Bits for an IP CPU Extender Module

For a CON extender module, the EXT Unit type bytes `0x00 0x01 0x00 0x92` are sent in little endian byte order and the binary interpretation is 00000000 00000001 0000000 00000010.

Bit by bit

0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

Definition Example of EXT Unit Type Bits for a CON Extender Module with local Output

For a CON extender module, the EXT Unit type bytes `0x00 0x02 0x00 0x01` are sent in little endian byte order and the binary interpretation is 00000000 00000001 0000000 00000010.

Bit by bit

0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

7.6.7.5 CPU Device Data Response Definition

Response

Telegram: ESC] y Size Type ID Error **Data <ID Status Type Name Ext PPort SPort Con>**

Type	Bytes	Description	Hex coding
ID	4	ID of CPU Device	E.g., 1012 = 0xF4 0x03 0x00 0x00
Status	4	Status register	0x01 0x00 0x02 0x80
Type	4	Type of CPU Device	0x00 0x00 0x00 0x00
Name	16 + 4	Name of the CPU Device	E.g., CPU_Video1 = 0x43 0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Ext	32	List of assigned EXT Units (maximum 8 entries à 4 bytes)	ID 90000001 = 0x81 0x4A 0x5D 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
PPort	32	Matrix port list for primary ports of EXT Units (maximum 8 entries à 4 bytes)	0x2F 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
SPort	32	Matrix port list for secondary ports of EXT Units (maximum 8 entries à 4 bytes)	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Con	4	ID of CON Device connected with the CPU Device	E.g., 3017 = 0xC9 0x0B 0x00 0x00

Response Example

Telegram: 0x1B 0x5D 0x79 0x90 0x00 0x05 0x00 0xF4 0x03 0x00 0x00 0x00 **0xF4 0x03**
0x00 0x00 0x01 0x00 0x02 0x80 0x00 0x00 0x00 0x00 0x43 0x50 0x55 0x5F
0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x81 0x4A 0x5D 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x2F 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0xC9 0x0B 0x00 0x00

Definition of CPU Device Status Bits

The binary interpretation of the received CPU Device status bytes `0x01 0x00 0x02 0x80` sent in little endian byte order is 10000000 00000010 00000000 00000001.

Bit by bit

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

CPU Device Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	CPU Device is online
02	0x00000002	00000000 00000000 00000000 00000010	CPU Device is connected private
16	0x00008000	00000000 00000000 10000000 00000000	Defined as reference CPU Device
17	0x00010000	00000000 00000001 00000000 00000000	Defined as virtual CPU Device
18	0x00020000	00000000 00000010 00000000 00000000	Allow private enabled
19	0x00040000	00000000 00000100 00000000 00000000	Force private enabled
20	0x00080000	00000000 00001000 00000000 00000000	Fix frame enabled
21	0x00100000	00000000 00010000 00000000 00000000	Is CPU Device group
22	0x00200000	00000000 00100000 00000000 00000000	Is CPU Device switch
23	0x00400000	00000000 01000000 00000000 00000000	2 step access enabled
24	0x00800000	00000000 10000000 00000000 00000000	Is SIRA CPU Device (IP CPU)
25	0x01000000	00000001 00000000 00000000 00000000	Exclusive access enabled
26	0x02000000	00000010 00000000 00000000 00000000	Multi-Screen Control mode disabled

This CPU Device is online, active, and allows a private connection.

7.6.7.6 CON Device Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error` **`Data <ID Status Type Name Ext PPort SPort PCpu SCpu User>`**

Type	Bytes	Description	Hex coding
ID	4	ID of CON Device	E.g., 3017 = `0xC9 0x0B 0x00 0x00`
Status	1	Status register	`0x01 0x00 0x00 0x80`
Type	4	0	`0x00 0x00 0x00 0x00`
Name	16 + 4	Name of CON Device	E.g., CON_3017 = `0x43 0x4f 0x4e 0x5f 0x33 0x30 0x31 0x37 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
Ext	32	List of assigned EXT Units (maximum 8 entries à 4 bytes)	ID 90000007 = `0x87 0x4A 0x5D 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
PPort	32	Matrix port list for primary ports of EXT Units (maximum 8 entries à 4 bytes)	`0x0F 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
SPort	32	Matrix port list for secondary ports of EXT Units (maximum 8 entries à 4 bytes)	`0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
PCpu	4	ID of CPU Device connected to CON Device via primary ports	E.g., 1012 = `0xF4 0x03 0x00 0x00`
SCpu	4	ID of CPU Device connected to CON Device via secondary ports	`0x00 0x00 0x00 0x00`
User	4	ID of user logged in to CON Device	E.g., 3 = `0x03 0x00 0x00 0x00`

Response Example

Telegram:

```
0x1B 0x5D 0x79 0x98 0x00 0x06 0x00 0xC9 0x0B 0x00 0x00 0x00 0xC9 0x0B
0x00 0x00 0x01 0x00 0x00 0x80 0x00 0x00 0x00 0x00 0x43 0x50 0x55 0x5F
0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x87 0x4A 0x5D 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x0F 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0xF4 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x03 0x00 0x00 0x00
```

Definition of CON Device Status Bits

The binary interpretation of the received CON Device status bytes `0x01 0x00 0x00 0x80` sent in little endian byte order is 10000000 00000000 00000000 00000001.

Bit by bit

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

This CON Device is configured and online.

CON Device Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
01	0x00000001	00000000 00000000 00000000 00000001	CON Device is online.
02	0x00000002	00000000 00000000 00000000 00000010	CON Device is connected private
03	0x00000004	00000000 00000000 00000000 00000100	CON Device is connected to video only.
04	0x00000008	00000000 00000000 00000000 00001000	CON Device was manually switched to link 1.
05	0x00000010	00000000 00000000 00000000 00010000	CON Device was manually switched to link 2.
16	0x00008000	00000000 00000000 10000000 00000000	Defined as reference CON Device
17	0x00010000	00000000 00000001 00000000 00000000	Defined as virtual CON Device
18	0x00020000	00000000 00000010 00000000 00000000	Allow login is enabled.
19	0x00040000	00000000 00000100 00000000 00000000	Force login is enabled.
20	0x00080000	00000000 00001000 00000000 00000000	Los frame is enabled.
21	0x00100000	00000000 00010000 00000000 00000000	Allow CPU scan is enabled.
22	0x00200000	00000000 00100000 00000000 00000000	Force CPU scan is enabled.
23	0x00400000	00000000 01000000 00000000 00000000	CON Device is a MSC CON Device.
24	0x00800000	00000000 10000000 00000000 00000000	CON Device is a MSC Control CON Device.
25	0x01000000	00000001 00000000 00000000 00000000	Port mode is enabled.
26	0x02000000	00000010 00000000 00000000 00000000	Redundancy is disabled.
27	0x04000000	00000100 00000000 00000000 00000000	Show macro list is enabled.
28	0x08000000	00001000 00000000 00000000 00000000	OSD is disabled.
32	0x80000000	10000000 00000000 00000000 00000000	CON Device is configured properly.

7.6.7.7 User Data Response Definition

Response

Telegram: `ESC ] y Size Type ID Error Data <ID Status Type Name Cpu Mode>`

Type	Bytes	Description	Hex coding
ID	4	User ID	E.g., 3 = `0x03 0x00 0x00 0x00`
Status	4	Status register	`0x00 0x00 0x00 0x80`
Type	4	0	`0x00 0x00 0x00 0x00`
Name	16 + 4	Username	E.g., John Doe = `0x4A 0x6F 0x68 0x6E 0x20 0x44 0x6F 0x65 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
Cpu	4	CPU Device ID of active respectively last CPU Device connection	E.g., 1012 = `0xF4 0x03 0x00 0x00`
Mode	4	Mode of CPU Device ID of active respectively last CPU Device connection	
		0 = Full Access = `0x00 0x00 0x00 0x00`	E.g., Full Access = 0 = `0x00 0x00 0x00 0x00`
		1 = Video-only = `0x01 0x00 0x00 0x00`	
		2 = Private Mode = `0x02 0x00 0x00 0x00`	
		3 = USB HID Control = `0x03 0x00 0x00 0x00`	

Response Example

Telegram: `0x1B 0x5D 0x79 0x34 0x00 0x07 0x00 0x03 0x00 0x00 0x00 0x00 0x03 0x00 0x00 0x00 0x00 0x00 0x80 0x00 0x00 0x00 0x00 0x4A 0x6F 0x68 0x6E 0x20 0x44 0x6F 0x65 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xF4 0x03 0x00 0x00 0x00 0x00 0x00 0x00`

Definition of CPU Device Status Bits

The binary interpretation of the received User status bytes `0x00 0x00 0x00 0x80` sent in little endian byte order is 10000000 00000000 00000000 00000000.

Bit by bit

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

This user is a standard matrix user, no LDAP user.

User Status Bits

Bit	Bits in hexadecimal	Binary bits	Description
17	0x00010000	00000000 00000001 00000000 00000000	Is user group
18	0x00020000	00000000 00000010 00000000 00000000	Is LDAP user
19	0x00040000	00000000 00000100 00000000 00000000	LDAP user sync

32	0x80000000	10000000 00000000 00000000 00000000	User is configured properly

7.6.7.8 Extended CON Device Response Definition

Response

Telegram: `ESC ] y Size Type ID Error` **`Data <ID Status Type Name Ext PPort SPort PCpu SCpu User ECpu>`**

Type	Bytes	Description	Hex coding
ID	4	ID of CON Device	E.g., 3017 = `0xC9 0x0B 0x00 0x00`
Status	1	Status register	`0x01 0x00 0x00 0x80`
Type	4	0	`0x00 0x00 0x00 0x00`
Name	16 + 4	Name of CON Device	E.g., CON_3017 = `0x43 0x4f 0x4e 0x5f 0x33 0x30 0x31 0x37 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
Ext	32	List of assigned EXT Units (maximum 8 entries à 4 bytes)	ID 90000007 = `0x87 0x4A 0x5D 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
PPort	32	Matrix port list for primary ports of EXT Units (maximum 8 entries à 4 bytes)	`0x0F 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
Sport	32	Matrix port list for secondary ports of EXT Units (maximum 8 entries à 4 bytes)	`0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
PCpu	4	ID of CPU Device connected to CON Device via primary ports	E.g., 1012 = `0xF4 0x03 0x00 0x00`
SCpu	4	ID of CPU Device connected to CON Device via secondary ports	`0x00 0x00 0x00 0x00`
User	4	ID of user logged in to CON Device	E.g., 3 = `0x03 0x00 0x00 0x00`
ECpu	32	List of assigned CPU Device IDs if individually connected by using the 0x6C command (maximum 8 entries à 4 bytes)	E.g., 1012= `0xF4 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

The same definitions as described in 7.6.7.6 (CON Device data definition).
Additionally from byte 141 to 172 the CPU Device IDs of individually connected extender modules of a multi head CON Device are listed.

7.7 Assignment Commands for CON Devices

7.7.1 Get Virtual CON Device

Request

Telegram: `ESC [ T Size RConID`

Get virtual CON Device of a real CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
T	1	Command	0x54
Size	2	Total length of telegram (7 bytes)	0x07 0x00
RConID	2	ID of a real CON Device	E.g., 3017 = 0xC9 0x0B

Request Example

Telegram: `0x1B 0x5B 0x54 0x07 0x00 0xC9 0x0B`

Get virtual CON Device of a real CON Device (RConID = 3017).

Response

Telegram: `ESC ] T Size RConID VConID`

Return virtual CON Device assigned to a real CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
T	1	Command	0x54
Size	2	Total length of telegram (9 bytes)	0x09 0x00
RConID	2	ID of a real CON Device	E.g., 3017 = 0xC9 0x0B
VConID	2	ID of a virtual CON Device	E.g., 4034 = 0xC2 0x0F
		0 = No assignment	0x00 0x00

Response Example 1

Telegram: `0x1B 0x5D 0x54 0x09 0x00 0xC9 0x0B 0xC2 0x0F`

Return virtual CON Device (VConID = 4034) assigned to real CON Device (RConID = 3017).

Or <NAK>

Response Example 2

Telegram: `0x1B 0x5D 0x54 0x09 0x00 0xC9 0x0B 0x00 0x00`

Return real CON Device (RConID = 3017) without assignment to a virtual CON Device (VConID = 0).

Or <NAK>

7.7.2 Assign/Remove Virtual CON Device to/from a Real CON Device

Request

Telegram: `ESC [ U RConID VConID`

Set virtual CON Device to a real CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
U	1	Command	0x55
Size	2	Total length of telegram (9 bytes)	0x09 0x00
RConID	2	ID of a real CON Device	E.g., 3017 = 0xC9 0x0B
VConID	2	ID of a virtual CON Device	E.g., 4034 = 0xC2 0x0F
		0 = Remove assignment	0x00 0x00

Request Example 1

Telegram: `0x1B 0x5B 0x55 0x09 0x00 0xC9 0x0B 0xC2 0x0F`

Assign virtual CON Device (VConID = 4034) to real CON Device (RConID = 3017).

Request Example 2

Telegram: `0x1B 0x5B 0x55 0x09 0x00 0xC9 0x0B 0x00 0x00`

Remove virtual CON Device (VConID = 0) assignment from real CON Device (RConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.7.3 Get Virtual CON Devices

Request

Telegram: `ESC [ X Size RConCount RConID[1…n]`

Get virtual CON Devices assigned to specified or all real CON Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
X	1	Command	0x58
Size	2	Total length of telegram (7 bytes + data = 13 bytes)	E.g., for RConCount = 3 0x0D 0x00
RConCount	2	0 = All real CON Devices with assignments to virtual CON Devices will be returned	0x00 0x00
		1...n = Number of real CON Devices with assignments to virtual CON Devices will be returned	E.g., 3 = 0x03 0x00
RConID	2	IDs of a real CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B 3028 = 0xD4 0x0B 3040 = 0xE0 0x0B

Request Example for Requesting specified Real CON Devices

Telegram: `0x1B 0x5B 0x5A 0x0D 0x00 0x03 0x00 0xC9 0x0B 0xD4 0x0B 0xE0 0x0B`

Get real CON Devices assigned to specified virtual CON Devices (RConID = 3017, 3028, 3040).

Request Example for Requesting all Real CON Devices

Telegram: `0x1B 0x5B 0x5A 0x07 0x00 0x00 0x00`

Get all real CON Devices assigned to all virtual CON Devices (RConCount = 0).

Response

Telegram: `ESC ] X Size RConCount <RConID VConID>[1…n]`

Return virtual CON Devices of real CON Devices as pairs.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
X	1	Command	0x58
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for RConCount = 3 0x13 0x00
RConCount	2	1...n = Number of CON Devices with assignments to virtual CON Devices will be returned	E.g., 3 = 0x03 0x00
RConID	2	IDs of a real CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B 3028 = 0xD4 0x0B 3040 = 0xE0 0x0B

Type	Bytes	Description	Hex coding
VConID	2	ID of a virtual CON Device (maximum 1024 entries)	E.g., 4034 = 0xC2 0x0F 4042 = 0xCA 0x0F 4045 = 0xCD 0x0F
		0 = No assignment	0x00 0x00

Response Example 1

Telegram: 0x1B 0x5D 0x58 0x13 0x00 0x03 0x00 0xC9 0x0B 0xC2 0x0F 0xD4 0x0B 0xCA 0x0F 0xE0 0x0B 0xCD 0x0F

Return virtual CON Devices of real CON Devices as pairs, for example:

RConID[1] = 3017, VConID[1] = 4034;
RConID[2] = 3028, VConID[2] = 4042;
RConID[3] = 3040, VConID[3] = 4045;

Or <NAK>.

Response Example 2

Telegram: 0x1B 0x5D 0x58 0x0F 0x00 0x03 0x00 0xC9 0x0B 0x00 0x00 0xD4 0x0B 0x00 0x00

Return real CON Devices without assignment to virtual CON Devices, for example:

RConID[1] = 3017, VConID[1] = 0, no assignment;
RConID[2] = 3028, VConID[2] = 0, no assignment

Or <NAK>.

7.7.4 Assign/Remove Virtual CON Devices to/from a Real CON Devices

Request

Telegram: `ESC [ Y Size RConCount <RConID VConID>[1…n]`

Assign or remove virtual CON Devices to/from real CON Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
Y	1	Command	0x59
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for RConCount = 3 0x13 0x00
RConCount	2	1...n = Number of real CON Devices with assignments to virtual CON Devices will be returned	E.g., 3 = 0x03 0x00
RConID	2	IDs of real CON Devices (maximum 1024 entries)	E.g., 3017 = 0xC9 0x0B 3028 = 0xD4 0x0B 3040 = 0xE0 0x0B
VConID	2	IDs of virtual CON Devices (maximum 1024 entries)	E.g., 4034 = 0xC2 0x0F 4042 = 0xCA 0x0F 4045 = 0xCD 0x0F
		0 = Remove assignment	0x00 0x00

Request Example 1

Telegram: `0x1B 0x5B 0x59 0x13 0x00 0x03 0x00 0xC9 0x0B 0xC2 0x0F 0xD4 0x0B 0xCA 0x0F 0xE0 0x0B 0xCD 0x0F`

Assign virtual CON Devices to real CON Devices as pair, for example:

RCpuID[1] = 1012, VCpuID[1] = 2018;
RCpuID[2] = 1013, VCpuID[2] = 2030;
RCpuID[3] = 1020, VCpuID[3] = 2035;

Request Example 2

Telegram: `0x1B 0x5B 0x59 0x0F 0x00 0x03 0x00 0xC9 0x0B 0x00 0x00 0xD4 0x0B 0x00 0x00`

Remove virtual CON Device (VConID = 0) assignment from real CON Devices (RConID = 3017, 3028) as pairs, for example:

RCpuID[1] = 3017, VCpuID[1] = 0, no assignment;

RCpuID[2] = 3028, VCpuID[2] = 0, no assignment

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.8 Assignment Commands for CPU Devices

7.8.1 Get Real CPU Device

Request

Telegram: `ESC [ V Size VCpuID`

Get real CPU Device of a virtual CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
V	1	Command	`0x56`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
VCpuID	2	ID of a virtual CPU Device	E.g., 2018 = `0xE2 0x07`

Request Example

Telegram: `0x1B 0x5B 0x56 0x07 0x00 0xE2 0x07`

Get real CPU Device of a virtual CPU Device (VCpuID = 2018).

Response

Telegram: `ESC ] V Size VCpuID RCpuID`

Return real CPU Device of a virtual CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
V	1	Command	`0x56`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
VCpuID	2	ID of a virtual CPU Device	E.g., 2018 = `0xE2 0x07`
RCpuID	2	ID of a real CPU Device	E.g., 1012 = `0xF4 0x03`
		0 = No assignment	`0x00 0x00`

Response Example

Telegram: `0x1B 0x5D 0x56 0x09 0x00 0xC9 0x0B 0xC2 0x0F`

Return real CPU Device (RCpuID = 1012) assigned to virtual CPU Device (VCpuID = 2018).

Or <NAK>

Response Example 2

Telegram: `0x1B 0x5D 0x56 0x09 0x00 0xC9 0x0B 0x00 0x00`

Return virtual CON Device (VConID = 2018) without assignment to a real CON Device (RConID = 0).

Or <NAK>

7.8.2 Assign/Remove Real CPU Device to/from a Virtual CPU Device

Request

Telegram: `ESC [ W Size VCpuID RCpuID`

Assign or remove real CPU Device to/from virtual CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
W	1	Command	`0x57`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
VCpuID	2	ID of a virtual CPU Device	E.g., 2018 = `0xE2 0x07`
RCpuID	2	ID of a real CPU Device	E.g., 1012 = `0xF4 0x03`
		0 = Remove assignment	`0x00 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x57 0x09 0x00 0xE2 0x07 0xF3 0x07`

Assign real CPU Device (RCpuID = 1012) to virtual CPU Device (VCpuID = 2018).

Request Example 2

Telegram: `0x1B 0x5B 0x57 0x09 0x00 0xE2 0x07 0x00 0x00`

Remove real CPU Device (RConID = 0) assignment from virtual CPU Device (VCpuID = 2018).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.8.3 Get Real CPU Devices

Request

Telegram: `ESC [ Z Size VCpuCount VCpuID`

Get real CPU Devices assigned to specified or all virtual CPU Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
Z	1	Command	`0x5A`
Size	2	Total length of telegram (7 bytes + data = 13 bytes)	E.g., for VCpuCount = 3 `0x0D 0x00`
VCpuCount	2	0 = All virtual CPU Devices with assignments to real CPU Devices will be returned	`0x00 0x00`
		1...n = Number of virtual CPU Devices with assignments to real CPU Devices will be returned (maximum 1024 assignments)	E.g., 3 = `0x03 0x00`
VCpuID	2	IDs of virtual CPU Devices	E.g., 2018 = `0xE2 0x07` 2030 = `0xEE 0x07` 2035 = `0xF3 0x07`

Request Example for Requesting some Real CPU Devices

Telegram: `0x1B 0x5B 0x5A 0x0D 0x00 0x03 0x00 0xE2 0x07 0xEE 0x07 0xF3 0x07`

Get real CPU Devices assigned to specified virtual CPU Devices (VCpuID = 2018, 2030, 2035).

Request Example for Requesting all Real CPU Devices

Telegram: `0x1B 0x5B 0x5A 0x07 0x00 0x00 0x00`

Get all real CPU Devices assigned to all virtual CPU Devices (VCpuCount = 0).

Response

Telegram: `ESC ] Z Size VCpuCount <VCpuID RCpuID>[1…n]`

Return real CPU Devices of a virtual CPU Devices as pairs.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
Z	1	Command	`0x5A`
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for VCpuCount = 3 `0x13 0x00`
VCpuCount	2	1...n = Number of virtual CPU Devices with assignments to real CPU Devices will be returned	E.g., 3 = `0x03 0x00`
VCpuID	2	IDs of virtual CPU Devices (maximum 1024 entries)	E.g., 2018 = `0xE2 0x07` 2030 = `0xEE 0x07` 2035 = `0xF3 0x07`

Type	Bytes	Description	Hex coding
RCpuID	2	IDs of real CPU Devices (maximum 1024 entries)	E.g., 1012 = 0xF4 0x03 1013 = 0xF5 0x03 1020 = 0xFC 0x03
		0 = No assignment	0x00 0x00

Response Example 1

Telegram: 0x1B 0x5D 0x5A 0x13 0x00 0x03 0x00 0xE2 0x07 0xF4 0x03 0xEE 0x07 0xF5 0x03 0xF3 0x07 0xFC 0x03

Return real CPU Devices assigned to virtual CPU Devices as pair, for example:

VCpuID[1] = 2018, RCpuID[1] = 1012;
VCpuID[2] = 2030, RCpuID[2] = 1013;
VCpuID[3] = 2035, RCpuID[3] = 1020;

Or <NAK>.

Response Example 2

Telegram: 0x1B 0x5D 0x5A 0x0f 0x00 0x03 0x00 0xE2 0x07 0x00 0x00 0xEE 0x07 0x00 0x00

Return virtual CPU Devices without assignment to real CPU Devices, for example:

VCpuID[1] = 2018, RCpuID[1] = 0, no assignment;
VCpuID[2] = 2030, RCpuID[2] = 0, no assignment

Or <NAK>.

7.8.4 Assign/Remove Real CPU Devices to/from Virtual CPU Devices

Request

Telegram: `ESC [ a Size VCpuCount <VCpuID RCpuID>[1…n]`

Assign or remove real CPU Devices to/from virtual CPU Devices.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
a	1	Command	`0x61`
Size	2	Total length of telegram (7 bytes + data = 19 bytes)	E.g., for VCpuCount = 3 `0x13 0x00`
VCpuCount	2	1...n = Number of virtual CPU Devices with assignments to real CPU Devices will be returned	E.g., 3 = `0x03 0x00`
VCpuID	2	IDs of virtual CPU Devices (maximum 1024 entries)	E.g., 2018 = `0xE2 0x07` 2030 = `0xEE 0x07` 2035 = `0xF3 0x07`
RCpuID	2	IDs of real CPU Devices (maximum 1024 entries)	E.g., 1012 = `0xF4 0x03` 1013 = `0xF5 0x03` 1020 = `0xFC 0x03`
		0 = Remove assignment	`0x00 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x61 0x13 0x00 0x03 0x00 0xE2 0x07 0xF4 0x03 0xEE 0x07 0xF5 0x03 0xF3 0x07 0xFC 0x03`

Assign real CPU Devices to virtual CPU Devices as pair, for example:

VCpuID[1] = 2018, RCpuID[1] = 1012;
VCpuID[2] = 2030, RCpuID[2] = 1013;
VCpuID[3] = 2035, RCpuID[3] = 1020;

Request Example 1

Telegram: `0x1B 0x5B 0x61 0x0f 0x00 0x03 0x00 0xE2 0x07 0x00 0x00 0xEE 0x07 0x00 0x00`

Remove real CPU Device (RCpuID = 0) assignment from virtual CPU Devices (VCpuID = 2018, 2030) as pairs, for example:

VCpuID[1] = 2018, RCpuID[1] = 0, no assignment;
VCpuID[2] = 2030, RCpuID[2] = 0, no assignment

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.9 Assignment Commands for CPU Groups

7.9.1 Get CPU Group of a CPU Device

Request

Telegram: `ESC [ p Size RCpuID`

Get CPU Group of a CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
p	1	Command	`0x70`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
RCpuID	2	ID of a CPU Device	E.g., 1006 = `0xEE 0x03`

Request Example

Telegram: `0x1B 0x5B 0x70 0x07 0x00 0xEE 0x03`

Get CPU Group of a CPU Device (RCpuID = 1006).

Response

Telegram: `ESC ] p Size RCpuID GCpuID`

Return CPU Group of a CPU Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
p	1	Command	`0x70`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
RCpuID	2	ID of a CPU Device	E.g., 1006 = `0xEE 0x03`
GCpuID	2	ID of a CPU Group	E.g., 2003 = `0xD3 0x07`

Response Example

Telegram: `0x1B 0x5D 0x70 0x09 0x00 0xEE 0x03 0xD3 0x07`

Return CPU Group (GCpuID = 2003) of a CPU Device (RCpuID = 1006).

Or <NAK>.

7.9.2 Get Members of a CPU Group

Request

Telegram: `ESC [ 2 Size GCpuID`

Get all CPU Device of a CPU Group.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
2	1	Command	0x32
Size	2	Total length of telegram (7 bytes)	0x07 0x00
GCpuID	2	ID of a CPU Group	E.g., 2003 = 0xD3 0x07

Request Example

Telegram: `0x1B 0x5B 0x32 0x07 0x00 0xD3 x0x07`

Get CPU Devices of a CPU Group (GCpuID = 2003).

Response

Telegram: `ESC ] 2 Size GCpuID Count <ID Name>[1…n] NUL`

Return list of CPU Devices of a CPU Group including ID and name.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
]	1	Server identification	0x5D
2	1	Command	0x32
Size	2	Total length of telegram (7 bytes + data = 53 bytes)	E.g., for Count = 2 0x35 0x00
GCpuID	2	ID of a CPU Group	E.g., 2003 = 0xD3 0x07
Count	2	1...n = Number of CPU Devices with assignments to CPU Groups will be returned	E.g., 2 = 0x02 0x00
ID	4	IDs of CPU Devices	E.g., 1001 = 0xE9 0x03 0x00 0x00, 1025= 0x01 0x04 0x00 0x00
Name	16	Name of CPU Device	E.g., for ID = 1001: CPU_Video1 = 0x43 0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00 0x00 E.g., for ID = 1025: CPU_Video2 = 0x43 0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x32 0x00 0x00 0x00 0x00 0x00 0x00
NUL	4	End character	0x00 0x00 0x00 0x00

Response Example

Telegram: 0x1B 0x5D 0x32 0x35 0x00 0xD3 0x07 0x02 0x00 0xE9 0x03 0x00 0x00 0x43
0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F 0x31 0x00 0x00 0x00 0x00 0x00
0x00 0x01 0x04 0x00 0x00 0x43 0x50 0x55 0x5F 0x56 0x69 0x64 0x65 0x6F
0x32 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Return list of all CPU Devices of a CPU Group including ID and name.

Or <NAK>.

7.9.3 Assign/Remove CPU Device to/from a CPU Group

Request

Telegram: `ESC [ q Size RCpuID GCpuID`

Assign CPU Device to a CPU Group.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5B`
q	1	Command	`0x71`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
RCpuID	2	0 = All CPU Devices	`0x00 0x00`
		ID of a CPU Device	E.g., 1006 = `0xEE 0x03`
GCpuID	2	ID of a CPU Group	E.g., 2003 = `0xD3 0x07`
		0 = Remove assignment	`0x00 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x71 0x09 0x00 0xEE 0x03 0xD3 0x07`

Assign CPU Device (RCpuID = 1006) to a CPU Group (GCpuID = 2003).

Request Example 2

Telegram: `0x1B 0x5B 0x71 0x09 0x00 0xEE 0x03 0x00 0x00`

Remove the CPU Group assignment of a CPU Device (RCpuID = 1012) with GCpuID = 0

Request Example 3

Telegram: `0x1B 0x5B 0x71 0x09 0x00 0x00 0x00 0xD3 0x07`

Remove all CPU Devices from a CPU Group (GCpuID = 2003) with RCpuID = 0

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.10 Restart and Shutdown Commands

7.10.1 Restart and Shut down of single Components of a Matrix (Controller Board and I/O Board)

Firmware Requirement

MATAPP 03.08

Request

This command must be sent to the matrix where the component to be restarted or shut down is located. For a Matrix Grid, the global index of the I/O board must be used.

Telegram: `ESC [ s Size Command P1 P2`

Restart and shut down of single components of a matrix.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
s	1	Command	0x73
Size	2	Total length of telegram (17 bytes)	0x11 0x00
Command	4	2 = Restart	0x02 0x00 0x00 0x00
		8 = Shut down	0x08 0x00 0x00 0x00
P1	4	0 = Controller board	0x00 0x00 0x00 0x00
		1…254 = Index of an I/O board	E.g., I/O board #7 = 0x07 0x00 0x00 0x00
P2	4	Reserved	0x00 0x00 0x00 0x00

Request Example 1 - Restart only the Controller Board of a Matrix

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Restart (Command = 2) controller board (P1 = 0) of a matrix with no effect for other components of the matrix or of other matrices of a Matrix Grid no matter if it's a single matrix, a Master Matrix, or Sub Matrix. Send the command to the matrix where the controller board is located (no distribution via the Master Matrix to Sub Matrices).

Request Example 2 - Shut down only the Controller Board of a Matrix

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Shut down (Command = 8) controller board (P1 = 0) of a matrix with no effect for other components of the matrix or of other matrices of a Matrix Grid no matter if it's a single matrix, a master matrix, or sub matrix. Send the command to the matrix where the controller board is located (no distribution via the Master Matrix to Sub Matrices).

Request Example 3 - Restart I/O Board #7 of a Matrix

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x02 0x00 0x00 0x00 0x07 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Restart (Command = 2) the single I/O board #7 (P1 = 7) of a matrix. Send the command to the matrix where the I/O board is located (no distribution via the Master Matrix to Sub Matrices)

Request Example 4 - Shut down I/O Board #7 of a single Matrix

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x08 0x00 0x00 0x00 0x07 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Shut down (Command = 8) the single I/O board #7 (P1 = 7) of a matrix. Send the command to the matrix where the I/O board is located (no distribution via the Master Matrix to Sub Matrices).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.10.2 Restart and Shut down of the whole Matrix (Controller Boards and I/O Boards)

Firmware Requirement

MATAPP 03.08

Request

Telegram: `ESC [ s Size Command P1 P2`

Restart controller board with no effect for other components of the matrix or other matrices of a Matrix Grid.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
s	1	Command	0x73
Size	2	Total length of telegram (17 bytes)	0x11 0x00
Command	4	2 = Restart	0x02 0x00 0x00 0x00
		8 = Shut down	0x08 0x00 0x00 0x00
P1	4	255 = Whole matrix	0xFF 0x00 0x00 0x00
P2	4	0 = Master matrix passes command to all sub matrices (all matrices of the Matrix Grid are affected) Note: For a single matrix without Grid configuration and for Sub Matrices of a Matrix Grid always use “0”. Sub Matrices never pass the command to other matrices.	0x00 0x00 0x00 0x00
		1 = Master matrix does not pass command to Sub Matrices (only the Master Matrix is affected, Sub Matrices are not affected)	0x01 0x00 0x00 0x00

Request Example 1 - Restart a Single Matrix or a Sub Matrix of a Matrix Grid

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x02 0x00 0x00 0x00 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Command must not be sent to a master matrix.
Restart (Command = 2) single matrix (P1 = 255) but not the master matrix of a matrix Grid (P2 = 0).

Request Example 2 - Shut down a Single Matrix or a Sub Matrix of a Matrix Grid

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x08 0x00 0x00 0x00 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00`

Command must not be sent to a master matrix.
Shut down (Command = 8) single matrix (P1 = 255) but not the master matrix of a matrix Grid (P2 = 0).

Request Example 3 - Restart only the Master Matrix of a Matrix Grid

Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x02 0x00 0x00 0x00 0xFF 0x00 0x00 0x00 0x01 0x00 0x00 0x00`

Command must be sent to a master matrix.
Restart (Command = 2) whole matrix (P1 = 255) which is the Master of a Matrix Grid without effect to the Sub Matrices (P2 = 1).

Request Example 4 - Shut down only the Master Matrix of a Matrix Grid

Telegram: 0x1B 0x5B 0x73 0x11 0x00 **0x08** 0x00 0x00 0x00 **0xFF** 0x00 0x00 0x00 **0x01** 0x00 0x00 0x00

Command must be sent to a master matrix.
Shut down (Command = 8) whole matrix (P1 = 255) which is the Master of a Matrix Grid without effect to the Sub Matrices (P2 = 1).

Request Example 5 - Restart the whole Matrix Grid (Master Matrix and all Sub Matrices simultaneously)

Telegram: 0x1B 0x5B 0x73 0x11 0x00 **0x02** 0x00 0x00 0x00 **0xFF** 0x00 0x00 0x00 **0x00** 0x00 0x00 0x00

Command must be sent to a master matrix.
Restart (Command = 2) whole matrix (P1 = 255) and the Master Matrix will pass the command to all Sub Matrices of the Matrix Grid to restart the whole matrix Grid (P2 = 0).

Request Example 6 – Shut down the whole Matrix Grid (Master and all Sub Matrices simultaneously)

Telegram: 0x1B 0x5B 0x73 0x11 0x00 **0x08** 0x00 0x00 0x00 **0xFF** 0x00 0x00 0x00 **0x00** 0x00 0x00 0x00

Command must be sent to a master matrix.
Shut down (Command = 8) whole matrix (P1 = 255) and the Master Matrix will pass the command to all Sub Matrices of the Matrix Grid to shut down the whole matrix Grid (P2 = 0).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.10.3 Restart a single Extender Module

Firmware Requirement

MATAPP 04.01.20220726

In order to restart an extender module, it is mandatory to send two different commands.

- The first command switches on the service mode for the device's specified EXT Unit.
- The second command restarts the extender module for the device's specified EXT Unit.

Please note that you have to wait 1 second between the first and the second command.

If you send the commands to the Master Matrix of a Matrix Grid, the command will automatically be passed to all the other matrices. Therefore, you can restart any extender module within a matrix Grid by sending the API commands to the Master Matrix.

Request

Telegram: `ESC [ s Size Command P1 P2`

Restart a single extender module.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
s	1	Command	`0x73`
Size	2	Total length of telegram (17 bytes)	`0x11 0x00`
Command	4	16 = Switch on the service mode of the specified EXT Unit of the CON/CPU Device	`0x10 0x00 0x00 0x00`
		17 = Restart the specified EXT Unit of the CON/CPU Device	`0x11 0x00 0x00 0x00`
P1	4	CON Device or CPU Device	E.g., CON Device 3017 = `0xC9 0x0B 0x00 0x00`
			E.g., CPU Device 1012 = `0xF4 0x03 0x00 0x00`
P2	4	1…8 = Index of the EXT Unit within the CON Device or CPU Device	E.g., 1 = `0x01 0x00 0x00 0x00`

Request Example Restart the first Extender Module of a CON Device

1. Switch on the service mode of the first EXT Unit of the CON Device 3017.

   Telegram: `0x1B 0x5B 0x73 0x11 0x00` **`0x10`** `0x00 0x00 0x00` **`0xC9 0x0B`** `0x00 0x00` **`0x01`** `0x00 0x00 0x00`

2. Sleep 1 second.
3. Restart the first extender module of the CON Device 3017.

   Telegram: `0x1B 0x5B 0x73 0x11 0x00` **`0x11`** `0x00 0x00 0x00` **`0xC9 0x0B`** `0x00 0x00` **`0x01`** `0x00 0x00 0x00`

Request Example - Restart the second Extender Module of a CPU Device

1. Switch on the service mode of the second EXT Unit of the CPU Device 1012.

   Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x10 0x00 0x00 0x00 0xF4 0x03 0x00 0x00 0x02 0x00 0x00 0x00`

2. Sleep 1 second.
3. Restart the second extender module of the CPU Device 1012.

   Telegram: `0x1B 0x5B 0x73 0x11 0x00 0x11 0x00 0x00 0x00 0xF4 0x03 0x00 0x00 0x02 0x00 0x00 0x00`

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.11 Execute Macros at a CON Device

Request

Telegram: `ESC [ o Size Key KeyUserID KeyConID ConID`

Execute a CON macro or a user macro.

Type	Bytes	Description	Hex coding
ESC	1	Control character	0x1B
[	1	Server identification	0x5B
o	1	Command	0x6F
Size	2	Total length of telegram (13 bytes)	0x0D 0x00
Key	2	Macro key (F1 to F32) F1 = 0x00...F16 = 0x0F Shift+F1 = 0x10...Shift+F16 = 0x1F	E.g., F3 = 0x02 0x0B
KeyUserID	2	1…n = Enable User ID (from matrix) for user macros	E.g., KeyUserID 5 = 0x01 0x00
		0 = No user macros executable	0 = 0x00 0x00
KeyConID	2	1…n = Enable ID of CON Device for CON macros	E.g., 3017 = 0xC9 0x0B
		0 = No CON macros executable	0 = 0x00 0x00
ConID	2	1…n = Enable ID of CON Device for executing user macros	E.g., 3017 = 0xC9 0x0B
		0 = No user macros executable	0 = 0x00 0x00

Request Example for Executing a User Macro

Telegram: `0x1B 0x5B 0x6F 0x0D 0x00 0x02 0x00 0x05 0x00 0x00 0x00 0xC9 0x0B`

Execute user macro (KeyUserID = 5) F3 at CON Device (ConID = 3017).

Request Example for Executing a CON Device Macro

Telegram: `0x1B 0x5B 0x6F 0x0D 0x00 0x02 0x00 0x00 0x00 0xC9 0x0B 0x00 0x00`

Execute CON Device macro F3 at CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.12 Setting Commands

7.12.1 Get logged in User of a CON Device

Request

Telegram: `ESC [ d Size ConID`

Get the user logged in to a CON.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
d	1	Command	`0x64`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
ConID	2	ID of the CON Device the user is logged in	E.g., 3017 = `0xC9 0x0B`

Request Example

Telegram: `0x1B 0x5B 0x64 0x07 0x00 0xC9 0x0B`

Get the user logged in to a CON Device (ConID = 3017).

Response

Telegram: `ESC [ d Size ConID User`

Return the user logged in to a CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
d	1	Command	`0x64`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
ConID	2	ID of the CON Device the user is logged in	E.g., 3017 = `0xC9 0x0B`
UserID	2	User ID from matrix	E.g., 5 = `0x05 0x00`
		0 = No user logged in to the CON Device	`0x00 0x00`

Response Example

Telegram: `0x1B 0x5B 0x64 0x09 0x00 0xC9 0x0B 0x05 0x00`

Return the user logged in to a CON Device (ConID = 3017).

Or <NAK>.

7.12.2 Login/Logout User at CON Device

Request

Telegram: `ESC [ e Size ConID UserID`

Login a user at a CON Device. Access to CPU Devices is immediately available.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
e	1	Command	`0x65`
Size	2	Total length of telegram (9 bytes)	`0x09 0x00`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
UserID	2	ID of a user to be logged in	E.g., 1 = `0x01 0x00`
		0 = Logout user	`0x00 0x00`

Request Example for Login

Telegram: `0x1B 0x5B 0x65 0x09 0x00 0xC9 0x0B 0x01 0x00`

Login user (UserID = 1) at CON Device (ConID = 3017).

Request Example for Logout

Telegram: `0x1B 0x5B 0x65 0x09 0x00 0xC9 0x0B 0x00 0x00`

Logout user (UserID = 0) from CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.12.3 Set User for API Context

Request

Telegram: `ESC [ n Size UserID`

Set user for API context. Available after setting as long as the socket is opened.

When a CPU list is queried, only the CPUs to which the user has access are returned.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
n	1	Command	`0x6E`
Size	2	Total length of telegram (7 bytes)	`0x07 0x00`
UserID	2	User ID from matrix	E.g., 5 = `0x05 0x00`

Request Example

Telegram: `0x1B 0x5B 0x6E 0x07 0x00 0x05 0x00`

Set user for API access (UserID = 5).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.12.4 Set Fix Frame Color

Request

Telegram: `ESC [ r Size CpuID ConID Color`

Set or delete specific fix frame color to/from CON Device or CPU Device. The fix frame color property has priority over the CPU_FIXFRAME option, if enabled.

A permanently frame is shown at CPU Devices with an active connection and CON Device with/without connection.

If both have a color frame assignment, the color of the CPU Device is shown for both the CPU Device and the CON Device.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
r	1	Command	`0x72`
Size	2	Total length of telegram (11 bytes)	`0x0B 0x00`
CpuID	2	ID of a CPU Device	E.g., 1012 = `0xF4 0x03`
ConID	2	ID of a CON Device	E.g., 3017 = `0xC9 0x0B`
Color	2	Color code	0 = Off `0x00 0x00`
			0 = 1 Blue `0x01 0x00`
			0 = 2 Green `0x02 0x00`
			0 = 3 Cyan `0x03 0x00`
			0 = 4 Red `0x04 0x00`
			0 = 5 Magenta `0x05 0x00`
			0 = 6 Yellow `0x06 0x00`
			0 = 7 White `0x07 0x00`

Request Example 1

Telegram: `0x1B 0x5B 0x72 0x0B 0x00 0xF4 0x03 0x00 0x00 0x04 0x00`

Set red fix frame color for CPU Device (CpuID = 1012).

Request Example 2

Telegram: `0x1B 0x5B 0x72 0x0B 0x00 0x00 0x00 0xC9 0x0B 0x02 0x00`

Set green fix frame color for CON Device (ConID = 3017).

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

7.12.5 Send OSD Message

Request

Telegram: `ESC [ 1 Size ConID Row Col ForeColor BackColor Time Text NUL`

Show message box at one or at all CON Devices.

Type	Bytes	Description		Hex coding
ESC	1	Control character		0x1B
[	1	Server identification		0x5B
1	1	Command		0x31
Size	2	Total length of telegram (depends on text length)		E.g., 32 bytes 0x20 0x00
ConID	2	0 = All CON Devices		0x00 0x00
		ID of CON Device		E.g., 3017 = 0xC9 0x0B, 3008 = 0xC0 0x0B
Row	4	Row padding. value x 10 pixels		
		0…127	• Message box is shifted X pixels from top	E.g., 14 x 10 pixels = 140 pixels from top = 0x0E 0x00 0x00 0x00
		-127…-1	• Message box is shifted X pixels from bottom	
		-128	• Message box centered	E.g., center position = -128 = 0x80 0xFF 0xFF 0xFF
Col	4	Column padding. value x 10 pixels		
		0…127	• Message box is shifted X pixels from left	
		-127…-1	• Message box is shifted X pixels from right	E.g., -14 x 10 pixels = 140 pixels from right = 0xF2 0xFF 0xFF 0xFF
		-128	• Message box centered	E.g., center position = -128 = 0x80 0xFF 0xFF 0xFF
ForeColor	4	Color of the font (0 = black)		E.g., yellow = 0x0D 0x00 0x00 0x00
BackColor	4	Color of the background (0 = semi-transparent like OSD)		E.g., blue = 0x0E 0x00 0x00 0x00
Time	4	Display time in seconds 0 = unlimited		E.g., 18 seconds = 0x12 0x00 0x00 0x00
Text	1 - 62	Text to be displayed in the message box		E.g., Test = 0x54 0x65 0x73 0x74
NUL	1	End character		0x00

Color Coding

Color	Hex coding
black / transparent	0 = 0x00 0x00
grey 1	1 = 0x01 0x00
grey 2	2 = 0x02 0x00
grey 3	3 = 0x03 0x00
grey 4	4 = 0x04 0x00
grey 5	5 = 0x05 0x00
grey 6	6 = 0x06 0x00
grey 7	7 = 0x07 0x00
grey-blue	8 = 0x08 0x00
light-orange	9 = 0x09 0x00
light-blue	A = 0x0A 0x00
red	B = 0x0B 0x00
green	C = 0x0C 0x00
yellow	D = 0x0D 0x00
blue	E = 0x0E 0x00
white	F = 0x0F 0x00

Request Example 1

Telegram: 0x1B 0x5B 0x31 0x20 0x00 0xC9 0x0B 0x0E 0x00 0x00 0x00 0xF2 0xFF 0xFF
0xFF 0x0D 0x00 0x00 0x00 0x0E 0x00 0x00 0x00 0x12 0x00 0x00 0x00 0x54
0x65 0x73 0x74 0x20

Send dialog message (Text = Test) to CON Device (ConID =3017) with text color yellow (ForeColor = D) on background color blue (BackColor = E) for 18 seconds in a message box 140 px from the top and 140 px from the right.

Request Example 2

Telegram: 0x1B 0x5B 0x31 0x5A 0x00 0xC0 0x0B 0x80 0xFF 0xFF 0xFF 0x80 0xFF 0xFF
0xFF 0x0D 0x00 0x00 0x00 0x0E 0x00 0x00 0x00 0x0A 0x00 0x00 0x00 0x22
0x54 0x68 0x69 0x73 0x20 0x69 0x73 0x20 0x61 0x20 0x74 0x65 0x73 0x74
0x20 0x6D 0x65 0x73 0x73 0x61 0x67 0x65 0x20 0x74 0x6F 0x20 0x43 0x4F
0x4E 0x20 0x44 0x65 0x76 0x69 0x63 0x65 0x20 0x49 0x44 0x20 0x33 0x30
0x30 0x38 0x20 0x66 0x6F 0x72 0x20 0x31 0x30 0x20 0x73 0x65 0x63 0x6F
0x6E 0x64 0x73 0x2E 0x22 0x00

Send dialog message (Text = "This is a test message to CON Device ID 3008 for 10 seconds.") to CON Device (ConID =3008) with text color yellow (ForeColor = D) on background color blue (BackColor = E) for 10 seconds in a centered message box.

7.12.6 Send OSD Message with Confirmation Button

Request

Telegram: `ESC [ 0 Size GetConID RetConID DialogID Row Col ForeColor BackColor Type Time Text NUL`

Show message box at one or at all CON Devices with the possibility to confirm the message via button.

Type	Bytes	Description		Hex coding
ESC	1	Control character		`0x1B`
[	1	Server identification		`0x5B`
0	1	Command		`0x30`
Size	2	Total length of telegram (depends on text length)		E.g., 32 bytes `0x20 0x00`
GetConID	2	0 = All CON Devices display the dialog		`0x00 0x00`
		ID of CON Device on which the dialog is displayed		E.g., 3017 = `0xC9 0x0B`, 3008 = `0xC0 0x0B`
RetConID	2	0 = Return response via API		`0x00 0x00`
DialogID	4	Dialog ID for identification		E.g., `0x04 0x00 0x00 0x00`
Row	4	Row padding. value x 10 pixels		
		0…127	Message box is shifted X pixels from top	E.g., 14 x 10 pixels = 140 pixels from top = `0x0E 0x00 0x00 0x00`
		-127…-1	Message box is shifted X pixels from bottom	
		-128	Message box centered	E.g., center position = -128 = `0x80 0xFF 0xFF 0xFF`
Col	4	Column padding. value x 10 pixels		
		0…127	Message box is shifted X pixels from left	
		-127…-1	Message box is shifted X pixels from right	E.g., -14 x 10 pixels = 140 pixels from right = `0xF2 0xFF 0xFF 0xFF`
		-128	Message box centered	E.g., center position = -128 = `0x80 0xFF 0xFF 0xFF`
ForeColor	4	Color of the font (0 = black)		E.g., yellow = `0x0D 0x00 0x00 0x00`
BackColor	4	Color of the font background (0 = semi-transparent like OSD)		E.g., blue = `0x0E 0x00 0x00 0x00`
Type	4	Dialog type		
		0: without button		`0x00 0x00 0x00 0x00`
		1: **Cancel** + Okay		`0x01 0x00 0x00 0x00`
		2: **No** + Yes		`0x02 0x00 0x00 0x00`
		3: **Deny** + Allow		`0x03 0x00 0x00 0x00`
		-1: Cancel + **Okay**		`0xFF 0xFF 0xFF 0xFF`
		-2: No + **Yes**		`0xFF 0xFF 0xFF 0xFE`
		-3: Deny + **Allow**		`0xFF 0xFF 0xFF 0xFD`
Time	4	Display time in seconds after the dialog will be closed and the default button will be sent back as response		E.g., 18 seconds = `0x12 0x00 0x00 0x00`

Type	Bytes	Description	Hex coding
Text	1 - 62	Text to be displayed in the message box	E.g., Test = 0x54 0x65 0x73 0x74
NUL	1	End character	0x00

Color Coding

Color code	Hex coding	Color
0	0 = 0x00 0x00	black / transparent
1	1 = 0x01 0x00	grey 1
2	2 = 0x02 0x00	grey 2
3	3 = 0x03 0x00	grey 3
4	4 = 0x04 0x00	grey 4
5	5 = 0x05 0x00	grey 5
6	6 = 0x06 0x00	grey 6
7	7 = 0x07 0x00	grey 7
8	8 = 0x08 0x00	grey-blue
9	9 = 0x09 0x00	light-orange
10	A = 0x0A 0x00	light-blue
11	B = 0x0B 0x00	red
12	C = 0x0C 0x00	green
13	D = 0x0D 0x00	yellow
14	E = 0x0E 0x00	blue
15	F = 0x0F 0x00	white

Request Example 1

Telegram: 0x1B 0x5B 0x30 0x2A 0x00 0xC9 0x0B 0x00 0x00 0x04 0x00 0x00 0x00 0x0E
0x00 0x00 0x00 0xF2 0xFF 0xFF 0xFF 0x0D 0x00 0x00 0x00 0x0E 0x00 0x00
0x00 0xFF 0xFF 0xFF 0xFE 0x12 0x00 0x00 0x00 0x54 0x65 0x73 0x74 0x00

Send dialog message (Text = Test) to CON Device (ConID =3017) with text color yellow (ForeColor = 13) on background color blue (BackColor = E). Display the message with buttons No and Yes (default) (Type = -2) for 18 seconds in a message box 140 px from the top and 140 px from the right.

Request Example 2

Telegram: 0x1B 0x5B 0x30 0x64 0x00 0xC0 0x0B 0x00 0x00 0x04 0x00 0x00 0x00 0x80
0xFF 0xFF 0xFF 0x80 0xFF 0xFF 0xFF 0x0D 0x00 0x00 0x00 0x0E 0x00 0x00
0x00 0x02 0x00 0x00 0x00 0x0A 0x00 0x00 0x00 0x22 0x54 0x68 0x69 0x73
0x20 0x69 0x73 0x20 0x61 0x20 0x74 0x65 0x73 0x74 0x20 0x6D 0x65 0x73
0x73 0x61 0x67 0x65 0x20 0x74 0x6F 0x20 0x43 0x4F 0x4E 0x20 0x44 0x65
0x76 0x69 0x63 0x65 0x20 0x49 0x44 0x20 0x33 0x30 0x30 0x38 0x20 0x66
0x6F 0x72 0x20 0x31 0x30 0x20 0x73 0x65 0x63 0x6F 0x6E 0x64 0x73 0x2E
0x22 0x00

Send dialog message (Text = "This is a test message to CON Device ID 3008 for 10 seconds.") to CON Device (ConID =3008) with text color yellow (ForeColor = D) on background color blue (BackColor = E). Display the message with buttons No (default) and Yes (Type = 2) for 10 seconds in a centered message box.

Response

Telegram: `ESC ] 0 Size GetConID RetConID DialogID Type`

Return message response.

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
]	1	Server identification	`0x5D`
0	1	Command	`0x30`
Size	2	Total length of telegram (17 bytes)	`0x11 0x00`
GetConID	2	0 = All CON Devices	`0x00 0x00`
		ID of CON Device that received the dialog	E.g., 3017 = `0xC9 0x0B`
RetConID	2	0 = Send response extern via API	`0x00 0x00`
DialogID	4	Dialog ID for identification	E.g., `0x04 0x00 0x00 0x00`
Type	4	Response	
		0: undefined	`0x00 0x00 0x00 0x00`
		1: Cancel/No/Deny	`0x01 0x00 0x00 0x00`
		2: Okay/Yes/Allow	`0x02 0x00 0x00 0x00`
		3: Timeout	`0x03 0x00 0x00 0x00`
		4: Request denied	`0x04 0x00 0x00 0x00`

Response Example

Telegram: `0x1B 0x5D 0x30 0x11 0x00 0xC9 0x0B 0x00 0x00 0x04 0x00 0x00 0x00 0x02 0x00 0x00 0x00`

Return message response with Yes (Type = 2) from CON Device (GetConID = 3017).

7.12.7 Toggle activated Multi-Screen Control temporarily off/on

This command can be used if a configured and activated MSC configuration should be temporarily toggled off and to toggle on the MSC configuration again. The information whether MSC is temporarily toggled off is part of the matrix configuration and remains valid after a restart.

A prerequisite for temporarily toggling off MSC is that an owner is configured and not set to "shared." By disabling MSC for a control CON, MSC is disabled for all CON Devices with the same owner.

If MSC is disabled, it is not possible to use MSC neither by moving the mouse nor by keyboard commands. For CON Devices where MSC is temporarily disabled, relative mouse coordinates are used.

Firmware Requirement

MATAPP F04.01.20221017

Request

Telegram: `ESC [ w Size Module Port MConID SConID`

Type	Bytes	Description	Hex coding
ESC	1	Control character	`0x1B`
[	1	Server identification	`0x5B`
w	1	Command	`0x77`
Size	2	Total length of telegram (13 bytes)	`0x0D 0x00`
Module	2	Not required = 0	`0x00 0x00`
Port	2	Not required = 0	`0x00 0x00`
MConID	2	ID of a Control CON Device within a MSC configuration with connected USB HID devices	E.g., 3006 = `0xBE 0x0B`
SConID	2	-1 = Toggle MSC on/off	`0xFF 0xFF`

Request Example for temporarily Toggling off an MSC configuration

Telegram: `0x1B 0x5B 0x77 0x0D 0x00 0x00 0x00 0x00 0x00 0xBE 0xC0 0xFF 0xFF`

Temporarily toggle off (SConID = -1) the MSC configuration of the Control CON Device (MConID = 3006).

To toggle on the temporarily toggled off MSC configuration for the Control CON Device (MConID = 3006), send the same command again.

Response

<ACK>[<ECHO>] or <NAK>.

[ ] = Optional elements

8 Troubleshooting

In the following chapters, support for problems with the DKM FX and DKM FXC API is provided. If you have problems regarding the involved devices, especially the DKM FX and DKM FXC matrix, refer to the device’s manual.

8.1 Failure at the Serial Interface

Diagnosis	Possible reason	Resolution
Serial control is not possible or only restrictedly possible.	Computer and matrix are operating at different Baud rates.	Change the Baud rate in the matrix and computer (see chapter 5.2.1, page 24).

9 Technical Support

Prior to contacting support, please ensure you have read this manual, and then installed and set-up your matrix as recommended.

10 Glossary

The following terms are commonly used in this manual or in video and KVM technology.

Term	Description
Cat X	Interface to connect any Cat 5e (Cat 6, Cat 7) cable
CON Device	Logical object that summarizes several EXT Units of physical extender modules (CON Units) to switch more complex sink systems via matrix
CON Unit	Decoder extender module to connect to the console (monitor(s), keyboard, and mouse; optionally also with USB 2.0 devices)
Console	Monitor, keyboard, mouse, media control, external switching solution, etc.
CPU Device	Logical object that summarizes several EXT Units of physical extender modules (CPU Units) to switch more complex source systems via matrix
CPU Unit	Encoder extender module to connect to a source
EXT Unit	Logical object for representing and managing an extender module physically connected directly to the matrix. Add-on modules, if applicable, are included in the EXT Unit of the respective extender module. Dual-Head extender modules will be managed as two independent EXT Units.
KVM	Keyboard, video, and mouse
MSC	Control of USB HID of up to eight sources at one sink with only one connected mouse or keyboard. The sink can consist of up to eight monitors, or up to sixteen monitors when operating Dual-Head Sources. In a matrix system, Multi-Screen Control (MSC) can be set up at multiple sinks.
OSD	The On-Screen-Display is used to display information or to operate a device.
USB HID	USB HID devices (Human Interface Device) allow for data input. There is no need for a special driver during installation; “New USB HID device found” is reported. Typical USB HID devices include keyboards, mice, graphics tablets, and touch screens. Storage, video, and audio I/O devices are **not** HID.

11 Change log

This table offers an overview of the most important changes available through firmware updates, such as new functions, changed configuration or operation.

Version	Date	Chapter	New functions/changes
REV1.00	02/2023		Initial user manual