

USB / DMX512 Control Box

User Manual for products delivered from December 2010

Status 17 May 2011

(The chaser function was improved. New "flash" feature.

If you need manuals for elder product versions, please order by email: cinetix@t-online.de. If available, add date of delivery or the revision number)

Fields of application:

This instrument is - apart from the USB control interface - functionally identical with the RS-232 / DMX512 Control Box

1. Control of a lighting bus connected to DMX OUT with simple ASCII text commands.

As alternative, a set of binary commands for the most essential operations is implemented.

The DMX512 control cycle is transmitted in its whole length of 512 bytes/DMX channels. If needed, the DMX cycle can be shortened – which results in a higher rate of repetition.

2. Multiplex("Merge")-capability: DMX level data, which were ordered via USB, can be merged with data from DMX IN to be transmitted via DMX OUT.

The signal combined of both sources is transmitted from DMX OUT. "Merge" means that the signal transmitted from DMX out is multiplexed or switched channel by channel between the received DMX signal and the one which is generated inside the box.

The data retransmitted this way can be synchronized with an internal timing clock or externally with the signal received at DMX IN.

3. Extraction and analysis of data which are received from an external DMX512 bus connected to DMX IN.

The received data are polled by command via the USB interface as short ASCII text strings or alternatively in a more compact binary format. Furthermore, the box can be configured to send messages automatically when the received DMX level at selected DMX channels has been changed more than a adjustable threshold with respect to the previous message. The device can be operated as a pure DMX receiver and can combine DMX reception with transmission of internally generated DMX data.

4. Output of complex strings (text or binary or mixed) via USB when specific data at selected DMX channels are received at DMX IN

This feature is intended to control media equipment with USB interface (beamers, professional DVD players, e.g.) by means of externally supplied DMX signals. The released strings can be freely programmed by the user. Up to 104 different strings can be transmitted. Each string may contain up to 255 bytes.

It is **NOT allowed** to use this instrument together with all safety critical applications, where misfunction could result in personal injury or noticeable material damage !

Hardware

Elements at the front panel:



The USB connector is located at the left part of the front panel.

The connection to the host PC is made with a common USB cable ("A type" plug to "B type" plug). A virtual serial port is installed on the PC to provide data communication

(for details, see page 4). The USB interface of the USB / DMX Control Box operates essentially independent of the baudrate selected at the users software. The virtual port can be driven with any standard baudrate between 2400 and 115200 baud. In addition to the data transfer, the USB /DMX Control Box is powered through the USB cable (5 Volt, max ca.130mA).

A **dual color LED** signalizes presence of power, the active mode of operation and the data flow via the USB port and DMX interface

--- In the mode of **DMX merge and transmission using internal clock of DMX timing, the base color of the LED is green**. When MIDI commands are received, the led is flashing dark. While changes at DMX IN are messaged via USB; the LED is flashing yellow-orange.

--- In the mode of **DMX merge and transmission snynchronized by the signal at DMX IN, the base color of the LED is yellow-orange**. When MIDI commands are received, the led is flashing dark. While changes at DMX IN are messaged via USB; the LED is flashing red.

--- In **DMX receive mode the the base color of the LED is red**. When MIDI commands are received, the led is flashing dark. While changes at DMX IN are messaged automatically via USB; the LED is flashing yellow-orange.

--- During a reset, the LED is darkened for about one second.

Pinout of the DMX512 interface:

DMX IN:

5-pin male XLR connector (optionally equipped with 3 pin male XLR connector)

DMX IN	XLR Connector
Shield, Signal Ground	Pin 1
DMX- Receiver	Pin 2
DMX+ Receiver	Pin 3

At delivery the DMX input is terminated with a 120 Ohm resistor. If for special applications the termination shall be disabled: remove the 4 upper screws of the enclosure and lift off the top of it. Pull the jumper located near the DMX IN socket.

DMX OUT:

5-pin female XLR connector (optionally equipped with 3 pin female XLR connector)

DMX OUT	XLR Connector
Shield, Signal Ground	Pin 1
DMX- Transmitter	Pin 2
DMX+ Transmitter	Pin 3

The DMX input (receiver) as well as the DMX output (transmitter) each is galvanically isolated from the control circuitry by means of an optocoupler, both terminals are isolated against each other, too.

Important safety information:

The **purpose of galvanic isolation** of the DMX input and output is to eliminate DXM data errors due to moderate fault voltages which result from long loops of signal ground lines and protective earth wires within the complete lighting installation.

In conformance with the DMX standard, isolation is guaranteed only up to 42 Volt. It cannot provide an additional level of electrical safety against broken isolation and faulty safety measures of the connected lighting equipment!

The DMX standard demands that **the complete bus, driven by the DMX transmitter** should be connected with ground or "earth" **at one single point**. In most cases this will be the protective earth of your mains line. **This means: a galvanically isolated DMX transmitter particularly makes sense, if one or more DMX transmitters without galvanically isolated DMX input are connected to the bus.**

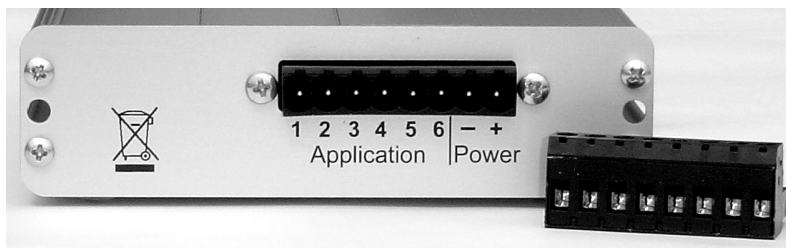
In most cases, even with professional equipment using isolated DMX receivers, a completely galvanically isolated bus obviously provides most reliable operation. **In particular cases ("latch up" of the bus line) however, it may be necessary to eliminate the galvanic isolation of the DMX transmitter.**

For this purpose a jumper is provided inside the box behind the aluminum heat sink. To change it, the box has to be opened: remove the 4 upper screws of the enclosure and lift off the top of it. In state of delivery it is connected towards the heat sink (=isolation present). To bridge the isolation, put it towards the rear panel.

Attention: The metal body of each DMX connector is connected with the metal cabinet of the USB / DMX Control Box and thus with the signal ground of the circuit board, too. Use of DMX cables, whose shielding is connected with the metal body of of the DMX plug, will cancel the galvanic isolation! **The shielding of the DMX cable has to be connected exclusively with pin 1 of the DMX plug.**

More details about the DMX512 standard see appendix B

Optional version: DMX Interface with industrial detachable clamps:



Arrangement of the 8-pin array of clamp terminals:

Terminal	Function
1	- DMX OUT
2r	DMX OUT Signal Ground
3	+ DMX OUT
4	- DMX IN
5	DMX IN Signal Ground
6	+ DMX IN
Power -	not connected at this model
Power +	not connected at this model

Due to the galvanic isolation between DMX IN and DMX OUT, the respective ground terminals have no connection among each other.

All features else are in correspondence with the XLR standard version (see above)

Installation of the USB / DMX512 Control Box

Safety Information: Because data transfer via DMX512 is regarded to be "unreliable" due to missing error checking, **the use of DMX control is explicitly FORBIDDEN together with all safety critical applications**, where malfunction could result in personal injury oder noticeable material damage !

Before the USB / DMX Control Box is capable to control a DMX bus, appropriate **USB drivers have to be installed** on the host PC: the USB / DMX Control Box is connected with a USB cable ("type A"-plug to "type B"-plug) to the host PC. The USB / DMX Control Box is completely powered via the USB cable (5 Volt, max ca. 130mA).

It is not possible to run the the USB / DMX Control Box via a passive hub (for example such one which is built into USB keyboards or similar equipment). For best reliability and stability of the DMX bus, we recommend to connect the Control Box directly to a USB socket at the PC.

Normally the USB / DMX Control Box is controlled via a "virtual COM port" installed on the host PC - this case is decribed below. Some application programs however use different techniques to drive the USB equipment. In case of doubt this item should be studied in the manual of the application software which is intended to communicate with the Control Box.

When the USB / DMX Control Box is plugged in, a message is displayed on the monitor screen (depending on the operating system), that a new hardware has been detected. Normally the user is asked, from which source the drivers shall be copied.

The USB / DMX Control Box is built with USB chips model FT245RL from manufacturer FTDI. The **supplied supportCD contains "VCP" (= virtual Com Port) drivers for Windows versions since Windows2000 in the folder "USB"** (a Zip archive with Windows98 drivers is added too, but this has to be extracted before use). Drivers for other operating systems or updated ones should be downloaded from the FTDI website (www.ftdichip.com) and decompressed.

The driver installation is performed in accordance with the screen dialogue.

It was experienced that Windows98 computers are blocked for about 30 seconds after the USB / DMX Control Box was plugged in. After the Control Box has been unplugged, it is recommend to wait about 30 seconds before re-plugging. Else in some cases the PC has to be rebooted to regain USB connectivity.

The USB / DMX Control Box operates baudrate-independent, i.e. any standard baudrate between 2400 and 115200 baud may be selected at the PC based controller application. But **necessarily** the following data format has to be selected: 8bit data, NO parity bit, 1 stop bit (= 8N1). Handshake is not supported by the USB / DMX Control Box. Due to internal buffers and high processing speed however, this is not necessary.

After the driver is installed, connect the DMX equipment with each other and with the box. In most cases, the USB / DMX Control Box will be used for the **control of lighting equipment** (i.e. as a DMX transmitter). **Then put a 5 pin male plug into the 5 pin female socket "DMX OUT" at the rear panel.** The DMX address switches of all dimmers etc. should be set in a scheme which makes best sense for your specific project.

If you are not sure and don't have experience with DMX, **we recommend for a first startup to use only one dimmer with lamp, set its address switch to "1"** and connect it.

If the box is intended to be used to **read data from an external DMX bus**, then put a 5 pin female plug into the 5 pin male connector "DMX IN" at the rear panel.

If received data are to be **looped through**, make both connections at DMX IN and DMX OUT
Next connect the DMX Control Box to the host PC with an USB cable. For a first test we recommend to start a terminal program (like Hyterterminal which is included in Windows).

Normally the LED at the front panel will be lit green now, i.e. **in default state the box operates as a DMX transmitter. All DMX OUT channels (DMX slang: "slots") should be set to level "0"**. At the terminal program running on the host PC a prompt "DMX Mi" should appear. (Default settings. When the Control Box was already in use, preset no.0 with possibly different behaviour is automatically loaded when the power is switched on). When the return key of the PC is pressed, the LED should be dark for a moment and the box should respond with a <carriage return>.

ASCII-Protocol of the USB / DMX512 Control Box

Two sets of control commands are available: the text oriented ASCII protocol and the binary protocol. Both variants work transparently together, i.e. no explicit mode switching is necessary.

If used together with industrial controllers or with PC based applications which have poor capabilities of data transformation, sometimes the binary protocol (described in the next chapter) is more suitable. Furthermore it is compact and fast, but needs a certain level of programming skills to be used.

In contrast, the ASCII protocol can be operated by hand from a terminal program, which is recommended for a first check of the box.

Every control command and every state message starts with a single characteristic letter. If necessary, it is followed by a number as parameter.

All ASCII characters are interpreted case independent. Feedback - if any - is always given in uppercase letters. Especially for tests the box can completely be operated with a simple terminal program.

Quick start and elementary commands:

The number of the DMX channel (1 - 512) to be addressed by subsequent commands is set with the command "S" followed by the number of the DMX channel as decimal number of one to three digits. This is stored in the internal SLOT register (see appendix A). No action else is directly triggered by this command.

Next enter the level (lighting intensity, "value") for this DMX channel with the command "V" followed by the desired value as a decimal number 0 to 255 (must have three digits or be directly followed by another command or must be terminated with <return>).

Example: S1V45 <return> is equivalent to S001V045

If you want to **start a complete fade process** with one command, first set the fadetime with command "T" (parameter in seconds, optionally tenths of seconds separated by a period). It works on all following commands which set the DMX level until a new fadetime is entered. It may be changed immediately after a level command without retroactive effect on given commands. **Fadetime = 0 switches directly to the final level.** Max. fadetime is 31.9 sec.

Example: T3.5S1V45<return>

If you want to set values of a block of consecutive DMX channels, use the ',' (**comma**) command to pre-increment the DMX address before setting a new value:

Example: S1V45,0,255,36....<return>

modifies DMX channels 1,2,3,4, and fades with the previously set fadetime.

The signal at DMX IN may be evaluated by different methods. The most simple but not most flexible is **polling DMX levels with the command R:**

Example: S3R100 returns the actual DMX IN -levels of channels 3 to 103 .

More elegant is **activation of "automatic messages" with command N or / ("slash"):**

Example: after being activated with S1N8 a message is transmitted automatically via RS-232, as soon as the DMX level of channel no. 1 has changed at least 8 in relationship to the previous message. After command /8 a message is released automatically, when any DMX channel changed by 8 or more.

The command "Q" does not change any values but it informs about the actual settings of the DMX channel which was addressed with the "S" command before. Typical example:

Mi: CH=1 OUT=13(U) TX=27 MF=50% RX=34 MSG=02/0 CS=128/0/20 L=512 T=3.2

OUT reports the presently transmitted DMX level of the channel addressed by CH= SLOT.TX describes the content of the transmit buffer of this DMX channel. Both differ when the global masterfader (item MF) differs from 100%. RX is the level actually received at DMX IN. The first 2 bytes in MSG describe the global setting of DMX-triggered RS-232 messages, the second parameter is the threshold for

"automatic" messages. CS describes start, length and timing of the chaser loop. L shows the number of channels transmitted per DMX cycle and T reports the fade time set by a previous command.

General structure of commands:

CommandCode Parameter [CmdCode Parameter] <CARRIAGE RETURN=hexD>

Every command causes its parameter to be written into the corresponding internal register. Depending on the type of command, when all information is entered the intended action is started using the actual content of ALL OF THE REGISTERS. For this reason, the order of entering data may influence the result. In general **the order FADETIME, SLOT, DMX level should be followed**, but most times not all of the registers have to be updated. (A more detailed description of the internal structure of the DMX Control Box you find in appendix A.)

Parameters (i.e. numeric values) for SLOT, FADETIME and LOOP always have to be entered in **decimal format** and are fed back in decimal.

The parameter of the **masterfader** is always entered in percent (without postponed % sign) in the range 0 to 200.

Parameters for DMX levels may be entered as a **decimal number, a hex number or in percent scale** and are messaged in this active number base. **Any DMX level is internally stored as one byte (8 valid bits)** i.e. takes values between 0 and 255.

Every command is executed as soon as the necessary number of ASCII characters is entered and interpreted. **Input of any number is automatically finished when the maximum number of digits** - context dependent - is entered. Numbers shorter than maximum are finished by starting the next command (input of a command letter) or when "return" is entered or when the number is filled up to its maximum length with leading zeroes.

Example: S1V23<return> is equivalent with **S1V023** or **S001V23Q**

Spaces are not interpreted, i.e. can be used to make the commands more readable

"Carriage Return" (= hexD = dec13) definitively **finishes a command**, starts execution and acknowledges with a <CARRIAGE RETURN> prompt. "Line Feed" (= hexA = dec10) is ignored. This must be taken into account when configuring application programs. If a **typing error** has happened, the pending command should be erased with a <return> keystroke. "Backspace" is out of action.

Feedback response:

The box does not "echo" input commands. If you prefer echoes, activate "local echo" at your terminal. Any number of subsequent commands can be entered per line, but a <CARRIAGE RETURN> should be inserted occasionally to obtain maximum system stability.

Every command line which has been interpreted correctly is acknowledged with a <CARRIAGE RETURN> after typing <return>.

If a command cannot be interpreted (usually due to a syntax error) the box echoes a '?' plus <CARRIAGE RETURN>. If additional commands were already entered, they are lost.

The box sends polled messages immediately after the command is given (Q-, R- or Z- command) or as soon as the corresponding condition is met (automatic messages by N command). **Every requested message is terminated with a <CARRIAGE RETURN>** (=hexD=dec13). This may be used by analyzing software to recognize the end of a syntactically coherent block of data.

Data format of the box "hello prompt" after power on or reset:

operated in DMX merge operation with internal clock:

DMX Mi <CARRIAGE RETURN>

operated in DMX merge operation with external synchronisation:

DMX ME <CARRIAGE RETURN>

operated as pure DMX receiver with received data looped through by hardware:

DMX RX <CARRIAGE RETURN>

This message is sent automatically after power on or after reset of the box. Repeated spontaneous output of this message may be a hint for unstable power supply or extreme immersion of electromagnetic radiation.

Short reference of all ASCII commands

Sn	address DMX channel (write SLOT register) for subsequent action (n=1 - 512)	p.8
Vn	set DMX level at DMX channel=SLOT (n=0 - 255)	p.8
,n	(comma) increment SLOT first then set level at new DMX channel (n=0 - 255)	p.8
=n	fill block of n DMX channels starting from SLOT+1 with level of SLOT (n=1-512)	p.9
+	increase transmit buffer level DMX channel=SLOT by one	p.9
-	decrease transmit buffer level DMX channel=SLOT by one	p.9
^n	add n to transmit buffer level DMX channel=SLOT (n=0 - 255)	p.9
_n	subtract n from transmit buffer level DMX channel=SLOT (n=0 - 255)	p.9
\$	from now DMX level in HEX (only V- , comma- , ^ , _- , R- , Z- , Q- command)	p.9
&	from now DMX level DECIMAL (only V , comma , ^ , _ , R , Z and Q-command)	p.10
%	from now DMX-level in PERCENT (only V , comma , R , Z and Q- command)	p.10
Ts.t	set FADETIME s=seconds t=tenths	p.10
!	stop all fade processes and freeze them at actual DMX level	p.10
Jn	copy DMX receive buffer starting from channel=SLOT into the transmit buffer	p.11
Zn	read n bytes starting at DMX channel=SLOT from transmit buffer to USB	p.11
Rn	read n bytes starting at DMX channel=SLOT from receive buffer to USB	p.11
U	always transmit DMX OUT at channel=SLOT from the transmit buffer (n=0)	p.12
K	transmit less buffer level at channel=SLOT (n=1)	p.12
G	transmit greater buffer level at channel=SLOT (n=2)	p.12
o	always transmit DMX OUT at channel=SLOT from the receive buffer (n=3)	p.12
P	transmit last changed buffer level at channel=SLOT (n=4)	p.13
*n	set merge method (n=0,1,2,3,4 see above) globally for all DMX channels	p.13
Nt	configure automatic message at channel=SLOT with threshold t (t=0-127)	p.13
/t	configure automatic messages for all DMX channels equally with threshold t	p.14
Yn	switch automatic messages globally ON/OFF and select format(n=0 - 4)	p.14
Xn	switch messaging of DMX triggered strings ON/OFF (n=0,1,2,3)	p.15
: <memory name> <string>	" enter and store user programmable string	p.16
; <memory name>	check string, message it for test via USB	p.17
Mn	set the masterfader: n=0 to 200 (in percent, see detailed description below)	p.17
in	set length of chaser cycle (n=2 to 127) and start the chaser	p.18
>t	set duration of chaser step in 1/10 s units	p.18
(n	enter start scene (preset no.) of chaser cycle . See detailed description	p.18
)	forward chaser immediately for one step	p.18
<t	release a lighting flash: all DMX channels are pulsed to 100% for t * 1/10s	p.19
H	set hue (spectral color) for RGB lamp	p.18
W	set color saturation for RGB lamp	p.19
"	set brightness (luminance) for RGB lamp	p.20
Ln	set length of DMX Cycle (n=24 - 512)	p.20
Q	show content of all DMX registers at DMX channel=SLOT via USB	p.20
[run DMX transmitter/merger with internal timing clock	p.20
\	run DMX transmitter/merger synchronised by DMX IN	p.20
]	loop DMX receiver from DMX IN to DMX OUT	p.20
~n	save transmit buffer and configuration as preset no. n	p.21
@n	load preset Nr. n into buffers and update configuration	p.22
{ }	"download", nonvolatile memory (i.e. backup all presets to PC) via RS-232	p.22
{ [}	"upload" saved backup (all presets) via RS-232 into nonvolatile memory	p.23
 	reset all buffers and configuration to default (delivery) state	p.23
'	(Apostroph) return firmware version number via USB	p.23

Detailed description of all ASCII commands:

Every control command and every state message is assigned with a single characteristic letter. If a command expects a parameter, it is listed after the command letter in acute angular brackets <...>. Number values are sent to the box as ASCII text.

This compact format is suitable to enter commands manually as well as for automatic generation and parsing in an application software. In addition there are a **set of compact binary commands available to set levels of any DMX channel or of blocks of DMX channels.** Detailed description see part 2 of the manual.

Addressing the DMX channel ("slot") to be operated with following commands:

S <channel number>

The **parameter addresses a DMX channel**, on which many of the subsequently described commands have an effect. Internally the parameter value is stored in the SLOT register.

In DMX slang sometimes the word "slot" is used as synonym for DMXchannel because during DMX transmission every DMX channel is represented by a specific time slot in the transmission cycle.

Parameter: slot number (range 1 to 512) is the number of the DMX channel to be manipulated with subsequent commands

Comment: No action is started immediately. But the register content will be used for subsequently given commands.

Example: S123 writes 123 into register SLOT

Transmit buffer manipulation (with intermediate storage in the preload buffer):

V <level>

Write parameter into VALUE and simultaneously write it into the transmit buffer of DMX channel = "SLOT".

Parameter: level (range 0 to 255) is the value (lamp intensity, e.g.) which will be transmitted at the DMX channel addressed by SLOT.

Comment: Changes the transmitted DMX packet sequence with the actual content of all 3 registers SLOT, VALUE and FADETIME immediately. It depends on the entry of the action memory of the addressed DMX channel if this new level is transmitted actually.

If FADETIME is equal to zero, the value of the addressed DMX channel is immediately set to <level >

If FADETIME is nonzero, a fade process is started, which begins at the actual value of the addressed DMX channel and finishes, when the value of the addressed DMX slot is equal to <level>.

Example: V34 sets the DMX level to 34 at the DMX channel which is actually addressed by SLOT (i.e. selected before with the "S" command). The parameter is interpreted in the active number base (decimal=default, hex or percent).

, (comma) <level>

First this command increases the SLOT register automatically, then it writes the parameter into VALUE and writes it into the transmit buffer for the new DMX channel = "SLOT".

Parameter: level (range 0 to 255) is the value or intensity which will be transmitted at the DMX slot addressed by the new, incremented SLOT.

Comment: except the fact that the SLOT register is pre-incremented, the ',' (comma) command does the **same as the V command.**

= <block length>

This command writes the final level of the DMX channel addressed by SLOT into the number of <block length> DMX channels starting from (SLOT+1). Starting from the actual level of each of these channels a new fade to this final level is started. The fade time is given by the actual content of the FADETIME register.

Parameter: <block length> (1 to 512) is the number of DMX channels into which the same level is copied. Independent of the value of <block length> DMX channel no.512 is not exceeded.

+ (without parameter)

Increase (add 1 to the) byte addressed by SLOT directly in the DMX stream transmit buffer)

Comment: The byte cannot be made greater than decimal 255. If it is already equal to 255, the + command is ignored. Any fade process currently active on this time slot is cancelled immediately.

- (minus, without parameter)

Decrease (subtract 1 from the) byte addressed by SLOT directly in the DMX stream (transmit buffer)

Comment: The byte cannot be made less than 0. If it is already zero, the - command is ignored. Any fade process currently active on this time slot is cancelled immediately.

^ <summand>

Add summand to the preload buffer addressed by SLOT (and start a fade process)

Comment: The final value cannot be made greater than decimal 255. If the addition would make an overflow, the result is fixed to 255.

The effect is similar to the V command. But instead of an absolute DMX level the sum of (previous entry of VALUE plus <summand>) is restored in the VALUE register and taken as the final level of a new triggered fade process. Any active fade process of this DMX channel is overwritten with the new final level and the actual parameter of FADETIME and restarted.

_ <subtrahend>

subtract subtrahend from the preload buffer addressed by SLOT (and start a fade process)

Comment: The final value cannot be made less than 0. If the subtraction would make a borrow, the result is fixed to 0.

The effect is similar to the V command. But instead of an absolute DMX level the difference of (previous entry of VALUE minus <subtrahend>) is restored in the VALUE register and taken as the final level of a new triggered fade process. Any active fade process of this DMX channel is overwritten with the new final level and the actual parameter of FADETIME and restarted.

\$ (no parameter)

Set number base for input/output of VALUE as **hexadecimal**

Comment: All following parameter values of the commands V, ',' (comma), ^ and _ are interpreted as hexadecimal numbers (0 to FF). This behaviour remains active until a different number base is set with

another command. Because the number base is stored in presets no. 0 to 9, loading of a preset may change the active number base.

All messaged DMX level values are coded as hexadecimal numbers with a prefix "\$".

& (no parameter)

Set number base for input/output of DMX levels as **decimal**

Comment: All following parameter values of the commands V, ',' (comma), ^ and _ are interpreted as decimal numbers (0 to 255). This behaviour remains active until a different number base is set with another command. Because the number base is stored in presets no. 0 to 9, loading of a preset may change the active number base.

All messaged DMX level values are coded as decimal numbers without specifier symbol.

% (no parameter)

Set number base for input/output of DMX levels as **percent**.

Comment: All following parameter values of the commands V, ',' (comma), ^ and _ are interpreted as hexadecimal numbers (0 to 100). This behaviour remains active until a different number base is set with another command. Because the number base is stored in presets no. 0 to 9, loading of a preset may change the active number base.

All messaged DMX level values are coded in percentage scale with a postfix "%".

The use of the **percentage scale is significantly less accurate as the decimal or hex scale**.

Internally the USB/DMX Control Box always operates with the full scale 0-255 while input/output is rounded as integer. Consequently the repeated use of the percentage scale may cause remarkable differences from a precise calculation.

T <seconds.tenths>

Enter parameter into **FADETIME**. No action is started directly.

Parameter: Fadetime is always entered in seconds. **Optionally** - separated by a period - tenths of seconds can be added. Maximum fadetime is 31 seconds plus 9 tenths of a second.

Example: **T13.4** sets FADETIME to 13.4 seconds

! (no parameter)

All fade processes are stopped immediately and all DMX channels are freezed on their present

Special feature: (new from May 2010 / revision number 83)

To ensure optimum compatibility with existing lighting control software - especially "DMX Control", this command was enhanced as follows: If the exclamation mark is followed by a digit 1 to 9 in ASCII text format within 1/10 second, and this sequence is terminated by a "Carriage Return" (decimal 13, hex D), automatic messages are activated in a binary format as described subsequently. Else, if the exclamation mark is not followed by this addendum, the primary operation "FREEZE" is executed .

Not recommended for new development of control software. Instead, we recommend to activate MIDI compatible binary messages as described below on pages 15 and 24.

Binary messages enabled by this command are structured as follows:

If the messaging DMX channel is **between 1 and 255 or equal to 512**, then the message is formatted:

the first byte of the message is always binary <1>

the second byte is always binary <73> = <hex49> = ASCII "I"

the third byte is the number of the messaging DMX channel (=0 for channel 512)

the fourth byte is the actual DMX level of this channel

If the messaging DMX channel is **between 256 and 511**, then the message is formatted:
the first byte of the message is always binary <1>
the second byte is always binary <105> = <hex69> = ASCII "i"
the third byte is the number of the messaging DMX channel minus 256
the fourth byte is the actual DMX level of this channel

If the third or the fourth byte is equal to <1>, then byte stuffing is applied in the message. This means: any <1> is followed by an additionally inserted <0>. The acute brackets used in this notation characterize raw binary numbers (not ASCII) and are not transmitted.

These messages can be terminated with the ASCII command Y0 (see page 15).

J <block size>

Copies a block of <block size> actual DMX levels from the **receive** buffer into the **transmit** buffer starting from the DMX channel which is currently addressed by the SLOT register (S command)

Comment: At elder firmware versions (revision number <83) this command did only copy the **single** DMX level of the addressed DMX channel. From the actual firmware version it combines the features of the previous commands J and H.

Example: S1J512 copies the complete receive buffer into the transmit buffer. This is a useful feature to save to save an externally created lighting scene which is received at DMX IN as a preset of the USB/DMX Control Box, using the ~ command. Replaces command H of earlier firmware versions.

Poll the transmit buffer:

Z <number of bytes>

Poll <number of bytes> of the DMX transmit buffer starting from the DMX level = SLOT and send them via MIDI OUT.

Parameter: number of polled bytes (1 to max. 128)

Syntax of the resulting state message:

s <1st channel no.> v [\$]DMX level[%] [,[\$]DMX level[%]] <CR = dec13 =hexD>

Comment and example see below at the R command

Readout (poll) the receive buffer:

R <number of bytes>

Read out (retrieve) a block of <number of bytes> starting from DMX channel = SLOT and message it to the host PC in ASCII text format

Parameter: number of bytes to be read (1 to max. 128)

Structure of a read out message responding to the R command:

S <1st channel no.> v [\$]DMX level[%] [,[\$]DMX level[%]] <CR =dec13 =hexD>

DMX levels reported as decimal numbers are sent without a specifier symbol

DMX levels reported in hexadecimal are sent with prefix '\$'

DMX levels reported in percent are sent with post added '%'

It is impossible to read data out of DMX packets with nonzero start byte.

Comment: The first byte is read from the DMX channel actually addressed by SLOT. The printout of every polled DMX level is followed by a comma, the end of the message is marked with a <CARRIAGE RETURN =hexD>. This format is suitable for recording, it corresponds with the command sequence to set the same level configuration at DMX OUT.

Automatic messages (see N command) are an alternative to polled messages.

Example: S100R8 reads a block of 8 bytes starting from DMX channel no.100

Manipulation of the action memory / set the merge method:

Throughout this manual, "merge" means switching or multiplexing the source of the data transmitted at DMX OUT channel by channel between transmit buffer and receive buffer. It does not mean linear superposition of both values!

Any of the commands described in this part remains active until it is revised by another command. Note that merge methods and threshold values are stored for every channel in any of the presets no. 0 to 9. So loading of one of these presets can change the merge method and automatic messages. Loading presets 10 to 299 does not affect the merge method.

U (no parameter)

Transmit from the transmit buffer at the DMX channel addressed by SLOT

Comment: This method of merging transmit buffer and receive buffer is valid until the next command is given that changes this feature.

Example: S99U enables transmission of the actual value of the transmit buffer at DMX channel 99

K (no parameter)

Send the the less of transmit buffer and receive buffer at the DMX channel addressed by SLOT

Comment: This method of merging transmit buffer and receive buffer is valid until the next command is given that changes this feature.

Example: S99K enables transmission of the less of the actual values of the transmit buffer and receive buffer at DMX channel no.99

G (no parameter)

Send the the greater of transmit buffer and receive buffer at the DMX channel addressed by SLOT

Comment: This method of merging transmit buffer and receive buffer is valid until the next command is given that changes this feature.

Example: S99G enables transmission of the greater of the actual values of the transmit buffer and receive buffer at DMX channel no.99

o (no parameter)

Transmit from the receive buffer at the DMX channel addressed by SLOT

Comment: This method of merging transmit buffer and receive buffer is valid until the next command is given that changes this feature.

Example: S99M enables transmission of the actual value of the receive buffer at DMX channel 99

P (no parameter)

Send either transmit buffer or receive buffer which has changed last at the DMX channel addressed by SLOT

Comment: This method of merging transmit buffer and receive buffer is valid until the next command is given that changes this feature.

An attempt to reload the existing value to the transmit buffer is handled as "change".

Example: S99P enables transmission of that one of the actual values of the transmit buffer and receive buffer at DMX channel no.99 **which has changed last** (to any value).

* **<method>**

Set the action memory of all DMX channels to an identical merge method:

Parameter: method

0= send all channels from the transmit buffer

1= for all channels send the the less of transmit buffer and receive buffer

2= for all channels send the the greater of transmit buffer and receive buffer

3= send all slots from the receive buffer

4= for all slots either send transmit buffer or receive buffer which has changed last

Example: *3 causes DMX transmission exclusively from the transmit buffer.

Trigger data output from USB by signal change at DMX IN

Specific procedures for different kinds of task are available:

1.) "**Automatic Messages**" are transmitted as soon as the DMX IN signal level at an activated DMX channel changes for a certain amount. Then a specially formatted character string is transmitted via USB. This string has essentially the same format as the command, which would have to be sent to reproduce the corresponding signal level at DMX OUT.

So "Automatic Messages" are especially useful for recording and later playback as well es for analytic evaluation of specific singal levels at DMX IN. As an essential feature of this kind of message **the trigger level can be adjusted arbitrarily** per DMX channel. This way the extracted amount of data can be fitted optimal for the respective application.

2.) Up tp 108 arbitrarily "**preprogrammed**" strings can be stored nonvolatile in the USB / DMX Control Box by the user. These are transmitted automatically after certain level changes at the DMX channels 500 to 512 via USB. This way user software or external equipment with a USB port may be controlled by means of a DMX bus to a limited extent.

3.) New added is a similar freely applicable feature, which "spntaneously" uses the levels of the DMX channels 458 to 499: **A level change at DMX channel 458** (trigger slope from 0 to 100%) **forces transmission of exactly the present levels of DMX channels 460 to 499 as a binary string** via USB. The number of transmitted characters (1 to 40) is defined by the present DMX level at channel 459 ("counted string").

N <threshold>

activate automatic messages at the DMX channel currently addressed by SLOT and set the threshold level of the messages

Parameter n: threshold - describes how much the **received** DMX signal at the DMX channel addressed by SLOT must change with respect to the previous message until an automatic message is released:

n=0: deactivate automatic messages for this slot

n=1: every change of received data releases an automatic message

With all threshold settings else (2 through 127=hex7F,49%), a message is sent when the current DMX value of the respective DMX slot differs at least the threshold from the value reported or polled before. For many typical applications a good compromise between sensitivity and amount of data is a threshold of 8.

Special option: automatic messages in MIDI compatible **binary format:**

are activated with the command Y2. The binary coded messages are better readable with some kind of control equipment. See part 2 of this manual. Command Y1 switches back to output in ASCII format (default).

Structure of an automatically generated ASCII message:

S <slot> v [\$] DMX level [%] <CARRIAGE RETURN=hexD>

Subsequently queued automatic messages are separated with a <CARRIAGE RETURN=hexD>.

Example: S27N8 from now on, every change of the received signal at slot no.27, which differs more than 8 from the previous message, releases an automatic message.

Structure of an automatically generated binary message (after command Y1):

same format as MIDI compatible commands for setting a DMX level, see pages 28/29.

/ <threshold>

activates automatic ASCII style messages for all DMX channels and sets the same threshold level for all channels

Parameter: threshold - describes how much the **received** DMX signal at a given channel must have changed with respect to the previous message until an automatic message is released.

Comment: All details are identical with the N command. Please note that the data capacity of the USB interface is limited. So, this command should only be used, when changes at the DMX512 bus take place under controlled conditions. Else messages may be corrupted or lost.

Y <0 , 1 , 2 , 3 , 4>

switch automatic messages globally ON and OFF and select format

Y0 switch automatic messages globally **OFF**

Y1 switch automatic messages globally **ON** as **MIDI channel messages**

Y2 automatic messages globally **ON** in **ASCII** format **without timestamp** (default)

Y3 automatic messages globally **ON** in **ASCII** format **with relative timestamp**

format: r <thze> S <DMX channel no> v [\$] level [%] <CR>

<thze> is the time difference in 1/25 second units since the pervious automatic message

Y3 automatic messages globally **ON** in **ASCII** format **with absolute timestamp**

format: t <thze> S <DMX channel no> v [\$] level [%] <CR>

<thze> is the time difference in 1/25 second units.

Comment: Settings of these parameters are stored in any of the presets no 0 to 9.

When switched off globally with Y0, all automatic messages are kept stored. Only their output is suppressed. When switched on again with Y1, Y2, Y3 or Y4, the previously stored messages and trigger levels reactivated, but their output format may change.

Comment on timestamps : Following 9999 the count restarts in cyclic repetition from 0. So at maximum time differences up to 6.7 minutes can be distinguished.

Automatic messages with timestamp are not applicable for recording and later 1:1 playback via DMX OUT, but they may be helpful for manual analysis of lighting sequences at DMX IN.

X <0, 1, 2, 3>

Switch emission of preprogrammed strings globally ON and OFF

X0 Switch emission of strings globally **OFF** n

X1 Switch emission of preprogrammed strings (DMX channel 500-512) globally **ON**

X2 Switch emission "spontaneous" of strings (DMX channel 458-499) globally **ON**

X3 Switch emission of strings (DMX channel 458-512) globally **ON**

Important safety information:

Data transmission via DMX512 is technically regarded to be "unreliable". For this reason **it is STRICTLY FORBIDDEN to use the technique described here together with all safety critical applications, where malfunction could result in personal injury oder noticeable material damage !**

Comment: When switched off globally with command X0, all preprogrammed strings to be called with appropriate levels at DMX channels 500-512 are kept stored. Only their output is suppressed. This setting is stored in any of the presets no 0 to 9. When switched on again, the previous configuration is reactivated exactly and completely.

New added is the similar freely useable feature, which exploits "spontaneous" levels of the DMX cannels 458 to 499: if the level of DMX channel no. 458 changes (trigger edge from 0 to 100%) exactly the actual levels of the DMX channels 460 to 499 are sent as binary string via USB. The number of transmitted bytes (1 to 40) is defined by the DMX level at channel 459 ("counted string"). This way any kind and number of strings can be released via USB "spontaneously" at runtime. Because this feature makes only sense when driven by a precise working computer controlled DMX desk, it is deactivated by default and under normal operation. So the DMX channels 458 to 499 may be used for normal operations without limitation. Furthermore free use is possible even in activated state as long as channel no 458 performs no triggering edge 0 to 100%level.

When specific data are received at DMX IN

send preprogrammed strings via USB:

This allows messaging of user preprogrammed strings (text or binary) via USB, if specific DMX levels are received at DMX IN on channels 500 to 512. How to program these strings using a PC keyboard is subsequently described. Maximum length of a string is 255 bytes.

This function is quite useful, for instance, to switch beamers or other media devices with USB input from a separate DMX desk. Even coarse control of brightness and sound level is possible.

Table of "string names" in relationship with DMX channels and DMX levels which trigger messaging of strings:

DMX channel:

DMX level	500	501	502	503	504	505	506	507	508	509	510	511	512
hex 08	000	010	020	030	040	050	060	070	080	090	100	110	120
hex 28	002	012	022	032	042	052	062	072	082	092	102	112	122
hex 48	004	014	024	034	044	054	064	074	084	094	104	114	124
hex 68	006	016	026	036	046	056	066	076	086	096	106	116	126
hex 88	008	018	028	038	048	058	068	078	088	098	108	118	128
hex A8	00A	01A	02A	03A	04A	05A	06A	07A	08A	09A	10A	11A	12A
hex C8	00C	01C	02C	03C	04C	05C	06C	07C	08C	09C	10C	11C	12C
hex E8	00E	01E	02E	03E	04E	05E	06E	07E	08E	09E	10E	11E	12E

In the table above, every DMX "event" which is able to trigger a string message is assigned to a "string name", which is mnemonically oriented at the related DMX parameters.

A specific string is exclusively messaged, if

- 1.) at the specific DMX channel 500 to 512 **exactly** the specific one of the **sensitive DMX levels** as listed in the table above -Werte is recognized **new but stable** during 2 subsequent poll cycles. These poll cycles are not synchronized with the incoming DMX signal, i.e. the received DMX level has to be constant long enough (at least 0,2 seconds).
- 2.) a string has to be programmed by the user for the addressed string name. Unprogrammed or deleted strings are stored with length 0 and are not messaged.
- 3.) string messaging has to be globally activated (command "X1").

Any string is only messaged once. An example: if the level of DMX channel 500 is pulled from hexA8 to hexA9 and back again to hexA8, the string named 00A is not messaged again. For a repeated message, another string must be released at the same DMX channel before. There is a trick to message a certain string repeatedly: "message" an unprogrammed string name in between. This complicated looking procedure serves to avoid unintended messages when corrupted DMX data are received.

Messaging of strings controlled by DMX IN is internally handled with low priority and independently of all other modes of operation. Automatic messages take precedence but cannot slice a string which is in state of transmission.

For clarity we recommend to switch off automatic messages when string messaging is active.

String messaging is **globally switched ON** with the command **X1** and globally **switched OFF** with the command **X0** (programmed strings are not deleted). This setting is stored in the presets no. 0 to 3. Especially by saving preset no. 0 can be determined if this feature is active when the box is powered on. Default setting is OFF.

How to enter a string and store it:

: <string name> <string> "

The command code is a colon followed by the string name (3 places of ASCII text according to table above) of the DMX event which shall be assigned to the message. Letters A,C,E may be entered case independent.

Next the sting itself is entered as a sequence of ASCII text. It is terminated and permanently stored by a final quotation mark (ASCII code hex22). The quotation mark does not become part of the string, of course.

Example: :12eHello World"

"Printable" characters (i.e. all which can be entered directly with a standard keyboard - ASCII code range hex20 to hex7E) **are typed directly.**

To **enter any other byte value** (control characters, country specific letters), first type a backslash \ (ASCII code hex 5C), next type the ASCII code of that byte as **hex number of 2 digits** in ASCII text representation. A "new line" command would be entered as text sequence \0d\0a. No leading and trailing spaces should be typed because they will appear in the stored string.

The third "special key" is backspace (ASCII code hex08), which is used to delete a part of the entered string.

So, if any of the following characters shall become part of the string instead of being used to format the string, it has to be entered by use of the backslash method:

enter **backslash** \ as part of the string: type \5c

enter **quotation mark** " as part of the string: type \22

enter **backspace** as part of the string: type \08

Mouse functions, cursor keys as well as special function keys F1 to F12 are not recognised or evaluated.

Every typed character is echoed via USB. "Printable" characters are echoed 1:1, any else is echoed by backslash plus two hex digits. If a printable character was entered in backslash style it will be echoed as in "printed" style. As an example: \40 would cause the echo @.

As soon as a quotation mark is added to the entered string, it is stored permanently within the Control Box. The string can be deleted or overwritten at any time (up to 10.000 repetitions). No "editing" is possible.

A string is **deleted** by entering : **<string name>**" , i.e. simply type a quotation mark directly after the string name. The string is empty then and has length 0. This configuration is identical with that of a new unprogrammed memory.

Check a user programmed string and view it via USB:

; **<string name>**

The command code is a semicolon followed by the string name according to the table above.

First the number of stored characters is messaged, next the string itself in screen-readable format identical to the echo when programming strings as described above. "Non printable" characters are shown as a combination of a backslash followed by two hex digits. "Printable" characters are shown in "printed" style even if they were entered with backslash.

Attention: This test message is a readable "interpretation" of the memory programmed before. Every byte is stored binary. **When the strings are triggered by signals at DMX IN they are messaged in "raw binary", of course.**

M <percent>

Enter parameter for the masterfader. All DMX levels are modulated immediately

Parameter: The masterfader is always entered in decimal percentage scale (without postponed % sign and independent of the number base for DMX levels).

Default =100, maximum = 200, minimum = 0.

Comment: The masterfader works like a digital signal processor when the **transmit buffer is written into the DMX transmitter hardware**. It is useful for global adjustment of lighting scenes. **It does not change or influence any internal data of the DMX Control Box**. Data looped from DMX-IN to DMX-OUT are **NOT** modified.

The **actually transmitted level of every DMX channel** is the transmit buffer value multiplied by the masterfader factor, i.e. up to 200%. Due to internal fast integer arithmetics, the transmitted level may be slightly lower than exactly calculated (intermediate fractionals lost). Changes of the parameter are applied immediately, not influenced by FADETIME. The masterfader parameter is not stored in presets.

i <cycle length>

Set the length of the chaser cycle (n=2 to 127) and start the chaser

<cycle length> = 0 switches the chaser OFF (new from Dec.2010 / revision no. 90)

Comment: before the chaser can be started, the **step duration** (command >) as well as the **start scene** (command ()) has to be adjusted – **Details see description of these commands**. For an easy start, start at scene 128 and step duration 20 (= 2 seconds) is preset as default.

All settings of the chaser are stored in presets 0 to 3 and reactivated when any of these presets is loaded. This applies especially for preset no.0, which is loaded when the DMX Control Box is started.

The actual settings of fade time and master fader are taken over by the chaser.

The chaser feature works as follows: a sequence of lighting scenes is loaded in a cyclic manner to DMX channels 1 to 128. To obtain this, the DMX levels which are stored in presets no. 128 to 383 for **DMX channels 385 to 512 are exclusively copied to the DMX transmit buffer channels 1 to 128** and transmitted on the DMX bus (**see commands for partial saving and loading of presets below**)

Example: if the chaser cycle is set to 4 and the the chaser start is set to 128, then presets no. 128,129,130,131 are loaded partially, then preset no. 128 again and so on. Any other DMX channels 129 to 512 are not influenced by the chaser and may be operated independently.

> <chaser step duration>

Set duration of chaser step in 1/10 s units

Comment: After the duration of any chaser step is over, the chaser automatically loads a part of the next preset in the cycle: **stored DMX levels from DMX channels 385 to 512 are copied to and transmitted at the DMX bus channels 1 to 128.** After <cycle length> presets were loaded in sequence, the procedure is repeated from <chaser start> .

(<chaser start>

Enter start scene (preset no.) of the chaser cycle

Accepted start values: 128 to 383. Default at delivery: 128

If the chaser would access a preset beyond 383, the sequence continues with loading preset no. 128 etc..

Comment: By use of this method, a big number of individual chaser effects may be created and run easily and flexible. This arrangement is optimized for lighting installations, which use a maximum of 384 DMX channels in normal operation and whose chaser effects are concentrated on DMX channels 1 to 128. Experience has shown that this is the case for stage lighting of most music bands and small theaters.

) (no parameter)

forward the chaser immediately (asynchronous) for one step

< flash duration >

(new from Dec. 2010 / revision no. 90)

Release a lighting flash: all DMX channels are pulsed to 100% for $t * 1/10s$

Comment: with this command a special preset (=special lighting scene) is transmitted at all 512 channels of the DMX bus during a time period given by <flash duration>. After this timing period, all previous DMX levels are restored.

At delivery the flash lighting scene is prestored in a way that all DMX channels are pushed to 100% level. Used together with installation of complex lamp fixtures, this may result in undesirable complications (start stroboscope effect, for example).

To avoid this, a user prepared lighting scene may be stored permanently in a special memory area with the command ~FF. This lighting scene should be designed in a way to avoid these side effects - or a very individual flash pattern may be created. Note: this memory area may be rewritten up to 10.000 times. We recommend not to create new flash configurations dynamically during runtime.

H <hue>

Sets the spectral color (hue) for a group of 3 subsequent DMX channels (RGB lamp)

Comment: The hue may be entered in the range 0 to 255. This will approximately result in following colors. Intermediate hue values will result in intermediate colors:

H0:red, H43:yellow, H85:green, H128:cyan, H170:blue, H213:magenta, H255:red again.

In correspondence with the model of the driven lamp and setting of saturation and brightness the resulting color tint may differ somewhat.

The H command wirkt auf den aktuell adressierteinfluences the actually addressed DMX channel (actual entry to the SLOT register, for example set with command S) and the two next higher neighbours. It is provided that the RGB setting of the respective lamp is done on these 3 successive DMX channels. All features else of a complex lamp ("fixture") may be used independently.

When the H command is applied to a RGB triplet for a first time, previously set (global) values of color saturation (W command) and brightness (" command) are taken over. At the same time this triplet of DMX channels is marked for the RGB mode. From then on hue, saturation and brightness can be set independently, where the command has always to be addressed to the first DMX channel of the triplet. But as soon as a "normal" command to set the DMX level is sent to the first DMX channel of the triplet (especially V, comma ^, _ , = and ! command) the RGB mode is resolved and can only be reactivated with a new H command.

A previously set fade time also works on a RGB tripled marked with the H command. Apart from possibly the first fade operation the fade process then is not simply performed channel by channel to the new final levels, but the 3 grouped channels are faded through the spectral color space. The marker for the RGB feature is only saved in presets no. 0 to 3. If presets are loaded under an incompatible system configuration, this may result in unexpected DMX levels at the corresponding DMX channels

Attention: The marker for the RGB feature is only saved in presets no. 0 to 3 as part of the system configuration. If presets are loaded under an incompatible system configuration, this may result in unexpected DMX levels at the corresponding DMX channels

W <saturation>

Sets the color saturation for a group of 3 subsequent DMX channels (RGB lamp).

Comment: The parameter of <saturation> may take values between 0 and 255. The maximum value 255 sets a pure spectral color, at lower parameter values other color components are partially added which results in a pastel light. When the saturation is set to 0, independently of the hue setting a white or grey light is composed. See comments at commands H and ".

" <brightness>

Sets the resulting brightness for a group of 3 subsequent DMX channels (RGB lamp).

Comment: the <brightness> parameter may take values between 0 and 255. The maximum value 255 sets maximum light intensity, the value 0 switches the light intensity off. Fading down is performed linear, without taking the gamma characteristics of the driven lamp into account. Specially when high performance LEDs are driven most times very strong changes of light intensity are observed at low brightness. So in this range small parameter steps may result heavy changes of the RGB composition.

L <length>

Set the length of the DMX loop.

Parameter: length (range 24 to 512) is the number of user controlled channels to be transmitted per DMX packet

Comment: During **transmission with internal clock**, LOOP sets the number of channels transmitted in every DMX packet. When at later time any DMX channel above this value is addressed or written, the DMX cycle is automatically lengthened. With a new L command the active cycle can be reduced at any time. Due to the regulations of the DMX512 standard the cycle cannot be made shorter than 24 channels.

During transmission with external synchronisation LOOP is automatically fitted to the received DMX signal. In this case, the L command cannot be applied.

During **receive and loop through operation**, the value of LOOP is not important.

Example: L512 sets the length of the transmitted DMX packet to 512

Q (no parameter)

Returns actual settings of all registers of the DMX channel addressed by SLOT. The response is sent as readable ASCII text

Example of a typical message:

```
Mi: CH=1 OUT=13(U) TX=27 MF=50% RX=34 MSG=02/0 CS=128/0/20 L=512 T=3.2
```

Comment about example: OUT reports the presently transmitted DMX level of the channel addressed by CH= SLOT, TX shows the content of the transmit buffer, RX shows the content of the receive buffer. MF reflects the actual setting of the masterfader. To explain possible differences you are referred to the description of commands H,J,+,-,^,_,T,M, (,). The 1st byte after MSG is the global setting of "DMX triggered messages" (X command), the 2nd byte is the global setting of "automatic" messages (Y command). The parameter after the slash is the threshold of "automatic" messages for this DMX channel. CS describes the actual setting of the chaser in the order: start scene, cycle length, step duration. L shows the number of channels transmitted per DMX cycle and T reports the fade time.

[(no parameter)

Start merge operation with internally generated DMX timing clock:

Comment: The merge operation with internal clock should be used when only or essentially DMX data uploaded by the user are to be transmitted at DMX OUT and only few data received at DMX are to be inserted. Timing of DMX OUT is not synchronous with the received DMX packets. This may result in subtle inconsistencies of received and retransmitted DMX packets - especially at fast faders, chasers and strobes - or if pauses between transmitting slots are considerably longer than two stop bits. The base color of the control LED is green.

Note that received DMX packets with nonzero startbytes are ignored and not fed into the receive buffer. Instead the data of the last received packet with zero start byte is kept transmitting.

**** (no parameter)

Start merged operation with DMX clock externally synchronized with the signal received at DMX IN:

Comment: The merge operation with internal clock should be used when essentially DMX data received at DMX IN are to be transmitted at DMX OUT and only few data uploaded by the user are to be inserted. If the signal at DMX IN fails, the transmission at DMX OUT is corrupted, too. **So, this method is not useable for exclusive transmit operation when no signal is fed into DMX IN.** But the external synchronisation has the advantage to retransmit received DMX signals without timing distortion and it is able to forward DMX packets with nonzero startbytes. Problems may arise when the received DMX signal has timing parameters at the lowest edge or below the DMX standard. The base color of the control LED is yellow-orange.

Note that no user data are merged into received DMX packets with nonzero startbytes. These are forwarded as received. It is impossible to read data out of such packets, too.

] (no parameter)

Loop received DMX data directly to DMX OUT by hardware.

Comment: In this mode of operation no user programmed data can be transmitted. It is provided to read out data from a signal at DMX IN and no falsification of this signal retransmitted at DM OUT can be tolerated. The base color of the control LED is red.

~ <preset#>

save current content of the transmit buffer preset (=lighting scene) number <preset#>.

Parameter: preset# (range 0 to 383)

Comment: The setting of LOOP and the number base for DMX levels i.e. the "**system configuration**" – **is only saved in presets no. 0 to 3.** Thus, in presets no. 0 to 3 preferably different system configurations are stored for quick change. **The parameter value of FADETIME is exclusively stored in preset no. 0. Because this preset is automatically loaded when the Generator is powered on, this way a "soft start" may be configured.**

With all presets else only the actual lighting scene of the transmit buffer is saved, so these may be reloaded universally from different system configurations.

The masterfader parameter is not stored in presets.

Example: **~23** saves the actual state of the DMX Generator as preset no. 23

Special feature: save transmit buffer partially (new from May 2010 / revision number 83)
an optionally added hex digit (case independent) expands the command:

~A<preset#> stores DMX OUT channels 1-128 to preset channels 129 - 256

~B<preset#> stores DMX OUT channels 1-128 to preset channels 257 - 384

~C<preset#> stores DMX OUT channels 1-128 to preset channels 385 - 512

~D<preset#> exclusively stores DMX OUT channels 1-128 to preset ch. 1 - 128

any other levels in the preset remain unchanged. This feature is intended as a counterpart to the corresponding @ command to produce and test "long" presets with small lamp configurations.

Attention: with this special feature exclusively DMX levels are re-stored, in no case the channel specific details of the system configuration. Especially when use is made of RGB triplets special care is necessary to keep the markings of these triples aligned with the shifted channel numbers.

Special feature: save flash pattern permanently (new from Dec 2010 / revision number 90)

~FF saves the actual lighting scene in a special memory area,

which is loaded temporarily to DMX-channels 1 to 512 with the "flash" command.

Comment: this memory area may be rewritten up to 10.000 times. We recommend not to create new flash configurations dynamically during runtime.

Further details about the flash function see command < above

@ <preset>

Recalls and activates preset (= lighting scene) number <preset#>

Parameter: preset# (range 0 to 383)

Comment: When one of the presets no 0 to 3 is loaded, the system is reconfigured in correspondence with the stored parameters. When any preset else is loaded, only the content of the transmit buffer is reloaded (lighting scene). At delivery all presets of the DMX Generator are formatted to load the default state of the device. For details see command "|"

After switching power on or after a reset automatically preset no. 0 is loaded.

When FADETIME is set different from 0, the actual lighting scene is faded over into the loading one with this time constant. The parameter of FADETIME is not changed when a preset is loaded.

Exception: when preset no. 0 is loaded (switching the device on, for example), the value of FADETIME which is stored in this preset is applied.

Example: @34 loads preset no. 34 and makes it active

Special feature: load preset partially

(new from May 2010 / revision number 83)

an optionally added hex digit (case independent) expands the command:

@A<preset#> loads DMX channels 129 - 256 from preset to ch. 1 - 128 DMX OUT

@B<preset#> loads DMX channels 257 - 384 from preset to ch.1 - 128 DMX OUT

@C<preset#> loads DMX channels 385 - 512 from preset to ch. 1 - 128 DMX OUT

@D<preset#> loads DMX channels 1 - 128 from preset to ch. 1 - 128 DMX OUT

@E<preset#> loads DMX channels 129 - 256 from preset to ch. 129 - 256 DMX OUT

@F<preset#> loads DMX channels 257 -512 from preset to ch. 257- 512 DMX OUT

any other DMX OUT levels remain unchanged.

Cases A,B,C are intended to get effectively more presets at small installations

Cases D,E,F are intended to handle multi-room installations more comfortable

Attention: with this special feature exclusively DMX levels are loaded, in no case the channel specific details of the system configuration. Especially when use is made of RGB triplets special care is necessary to keep the markings of these triples aligned with the shifted channel numbers. Refer to p.19

{ } (no parameter)

Initiates "download" of the complete nonvolatile memory (all presets) to a PC via USB using the "XMODEM CRC" protocol

Comment: This command allows to make a backup of possibly expensively created presets or exchange different sets of presets. Together with presets no 0 to 3 the system configuration stored in these presets is downloaded, too.

To avoid accidental start of this command, a sequence of both command bytes has to be entered.

On the computer which is used for the backup a software has to be installed and started which is able to perform the transfer als a sequence of defined data packets using the XMODEM CRCprotocol.

Applicable transfer software is for example "Hyperterminal", which is combined with a terminal software to perform the normal operation of the USB/DMX Control Box, too.

First the command sequence { } has to be entered. Then a message is sent back to the controlling PC that the XMODEM CRC 'Receive' function should be started. Now the user opens a corresponding dialog at the XMODEM software where the name of the backup file is entered. The USB/DMX Control Box remains waiting for max 100 seconds while these steps are performed on the PC. When this is finished the PC initiates and controls the download. The download takes about 1 minute.

The command syntax has changed 17 May 2011 (revision number 93 and higher).
For elder firmware the command is { } (no parameter)

{ [(no parameter)

Initiates "upload" of a previously stored backup to the nonvolatile memory (all presets) from a PC via USB using the "XMODEM CRC" protocol

Comment: On the controlling PC a software has to be started, which is able to transfer the backup as a sequence of defined data packets using the "XMODEM CRC" protocol.

To avoid accidental start of this command, a sequence of both command bytes has to be entered.

It is exclusively possible to upload a backup from a USB/DMX Control Box, not from other similar Cinetix products. Under certain conditions after a hardware update (microcontroller module) a backup made with a previous version may be uploaded. Contact Cinetix service to check if this is possible in a specific case.

First the command sequence { [has to be entered. Then a message is sent back to the controlling PC that the XMODEM CRC 'Send' function should be started. Further guide see comment of the complementary download command {].

The command syntax has changed 17 May 2011 (revision number 93 and higher).

For elder firmware the command is { { (no parameter)

| (no parameter)

"clear all memory": all buffers and modes of operation are reset to default

Action memory: the merger transmits exclusively from the transmit buffer.
All automatic messages are switched OFF.

Transmit buffer: All DMX levels of the transmit buffer are reset to "0".
Number base = "decimal". Masterfader = 100% The Chaser is switched OFF.
LOOP = 512, FADETIME = 0.0. Presets are not deleted or changed otherwise.

' (apostrophe, no parameter)

returns the letter "R", followed by a version number of two decimal places

Comment: This command may be used by application software to search automatically for the appropriate COM port where the USB /DMX Control Box is connected. For service requests it is important to have the revision number available. Not used during normal operation.

} <pulse length> } <pulse length>

Special command to modify the duration of the DMX reset pulse

Comment: This command is intended to solve extremely seldom problem cases and **should not be used without recommendation of our service department.**

After the brace a byte is entered which is composed of two hex nibbles 0 .. F, each as a text character. **The same command must be repeated immediately after with the same byte parameter**, no space nor new line are allowed to be put between. The parameter is stored permanently independent of presets and is reloaded during every system start.

1st hex digit (high nibble): Duration of the DMX reset pulse: ca. 90us base value + (nibble value * 8 us). This way, the range of the DMX reset pulse may be varied between about 90us and 210 us.

2nd hex digit (low nibble): Duration of the mark pulse between DMX reset and start byte: ca. 10us + nibble value. So, the setup range is about 10us to 25us.

Exception: The command }00}00 resets the DMX timing permanently to its default values.

"USB / DMX512 Control Box" User Manual part 2

Binary protocol of the USB / DMX512 Control Box

When commands are formulated as ASCII text they are entered easily with a terminal program or with a comfortably equipped application software. **Frequently however, there is a problem with application programs having a very simple script language or with industrial PLC controllers to convert calculated numbers into their ASCII text representation.** Another problem when using ASCII commands is the fact that a bigger number of bytes has to be transferred per command. Binary commands are considerably faster – especially when blocks of DMX levels are to be changed.

So, though less clear and more complicated at a first view, we have added a set of binary commands for the most essential operations.

Two differently designed sets of binary commands are offered.

Binary commands are completely transparent coexistent with the ASCII based set of commands. All types of commands can be mixed arbitrarily - correct code input provided. No explicit switching between modes is.

First the command set based on "**non printable ASCII characters**" is described. This one was already available in previous firmware. **It uses ASCII Codes < 32(hex20) for synchronisation** and differentiation from normal ASCII commands. The widely known DMX-software "Freestyler" and "DMX-Control" control Cinetix devices based on this command set.

Next an alternative set of commands is described, which is formally compatible with **MIDI-channel messages**. This is slightly more complicated to be programmed on a PC or PLC. But due to its construction it distinguishes more robustly from ASCII commands and resynchronizes better in case of data transfer errors.

Binary set of commands based on "non printable" ASCII codes

Binary commands for setting a single DMX channel start a fade process - exactly like their ASCII equivalent.

Binary commands for block transfer however write directly into the preload buffer and neutralize active fade processes on the rewritten DMX channels. As an alternative with fade transition the MIDI CHANNEL PRESSURE compatible command is recommended.

With "printable" ASCII characters it is impossible to get into the binary mode. **The binary mode is automatically finished as soon as the demanded number of bytes was transferred via USB.** If single bytes should have gone lost, every binary transfer is finished automatically after 0.5 seconds to prevent the Control Box from "hanging up".

Every binary command starts with a typical command code which characterizes the kind of command. This way indirectly the number of bytes to be transferred by this command is fixed implicitly. All bytes are interpreted as raw binary even if they are "printable".

If a binary command was **entered wrong** or else could not be executed, a **question mark** is echoed just like in ASCII mode.

In the following characters put between acute brackets mean raw binary values, no ASCII codes. (The brackets themselves are not part of the command.)

How to send DMX data from a PC to the DMX Control Box:

The number in acute brackets is a **command code** which determines the transfer method to be used.

<2> Set DMX level within the channel range 1 - 255 (plus special case slot 512)

followed by 2 bytes:

1st byte: DMX channel 1 - 255. If parameter =0, then channel 512 is set

2nd byte: DMX level value to be entered at the addressed channel

After having received 3 bytes , control is automatically returned to ASCII level

<3> Set DMX level within the channel range 256 - 511

followed by 2 bytes:

1st byte: (channel no. 256 to 511) **minus 256**, resulting in byte1 = 0 to 255

2nd byte: DMX level value to be entered at the addressed channel

<4> Set levels in a subsequent block of DMX channels starting from channels no. 1 to 255.

The command code is followed by two additional **header bytes** and a **fixed number of data bytes**:

1st header byte: **0 to 255** start channel

2nd header byte: number of subsequently transferred data bytes to be written into the preload buffer. For a block of 256 data bytes enter 0.

Data bytes: counted sequence of arbitrary bytes.

It is absolutely necessary that exactly the number of data bytes is transferred, as entered in the 2nd header byte. If too many bytes are transferred these may be interpreted as ASCII commands and cause unintended operations – a bad source of errors. **If less bytes are transferred** the DMX Control Box hangs in an endless loop which is resolved automatically after 0,5 seconds with an error message. In this case, all bytes transferred before the hanging are written into the preload buffer

<5> Set levels in a subsequent block of DMX channels starting from channels no. 256 to 511

The command code is followed by two additional **header bytes** and a **fixed number of data bytes**:

1st header byte: **0 to 255** start channel (calculate: 256 to 511 **minus 256**)

2nd header byte: number of subsequently transferred data bytes to be written into the preload buffer. For a block of 256 data bytes enter 0.

Data bytes: counted sequence of arbitrary bytes.

If DMX channel 512 is exceeded due to a wrong calculation of the 2 nd header byte, surplus received bytes are dropped but the binary command mode is kept busy until the announced number of bytes is received. **Else see comment below command <14> !!**

<7> adjust masterfader and FADETIME.

followed by 2 data bytes:

1st data byte: set new value of the masterfader 0 to 200 (hexC8).

If the parameter is greater than 200, the masterfader is NOT changed by this command.

2nd databyte: set FADETIME in 1/10 seconds units, permissible values 0 to 254.

Differing from the corresponding ASCII command, this way fade times only up to 25.4 seconds can be selected.

If the parameter is equal to 255 (hexFF), the FADETIME is NOT changed by this command.

<12>(hex C) **load a preset (= lighting scene)**

with fade according to the actual value of FADETIME

followed by 2 data bytes:

1st data byte:

To load a preset in the range **0 to 255**, the **1st data byte is set = 0**.

To load a preset in the range **256 bis 383**, the **1st data byte is set = 1**.

and the 2nd data byte has to be (preset number - 256).

E.g.: preset no.299 is loaded with following byte sequence: 12 1 43 (bzw. hex C 1 2B)

2nd data byte: (if the **1st data Byte = 0**) preset no. to be loaded 0 to 255 (hex FF)

or (if the **1st data Byte = 1**) preset no. to be loaded **minus 256**, i.e 0 - 127 (hex 7F)

<14>(hex E) **set RGB hue (spectral color)** (if the first DMX channel of the addressed triplet is between 1 and 255) followed by 2 data bytes:

1st data byte: first DMX channel

2nd data byte: hue (0 - 255), see ASCII command H

<15>(hex F) **set RGB hue (spectral color)** (if the first DMX channel of the addressed triplet is between 256 and 510) followed by 2 data bytes:

1st data byte: (first DMX channel **minus 256**)

2nd data byte: hue (0 - 255), see ASCII command H

<16>(hex 10) **set RGB color saturation and brightness** (if the first DMX channel of the addressed triplet is between 1 and 255) followed by 3 data bytes:

1st data byte: first DMX channel

2nd data byte: saturation 0 - 255, see ASCII command W

3rd data byte: brightness (0 - 255) see ASCII Befehl "

<17>(hex 11) **set RGB color saturation and brightness** (if the first DMX channel of the addressed triplet is between 256 and 510) followed by 3 data bytes:

1st data byte: (first DMX channel **minus 256**)

2nd data byte: saturation (0 - 255), see ASCII command W

3rd data byte: brightness (0 - 255) see ASCII Befehl "

How to poll a block of subsequent levels from DMX IN:

<6>poll a block of subsequent levels from DMX IN in raw binary format

this opcode is followed by **3 additional command bytes**.

1st command byte: HIGH byte of the DMX channel no. where polling shall start

(=0 if channel nuber < 256 or 1, if channel number >= 256)

2nd command byte: LOW byte of the DMX channel no. where polling shall start

(= DMX channel number minus 0 or channel number minus 256 depending on the HIGH byte)

If both address bytes = 0, DMX channel no 512 is read.

3rd command byte: number of DMX channels to be polled .

Per command max. 256 DMX channels can be polled. To poll 256 channels

the 3rd command byte must be equal 0.

Example: to get (dec) 40 levels starting from DMX channel (dec) 370, following

command sequence has to be given: hex 6,1,72,28 - i.e. decimal 6,1,114,40

Example: to get (dec) 256 levels starting from DMX channel 1 auszulesen, following

command sequence has to be given: hex 6,0,1,0 - i.e. decimal 6,0,1,0

The **first responded byte** is a **header**, which announces the **number of data bytes** which will actually follow.

Its value 0 means that 256 bytes of DMX levels will follow. If due to the ordered start address and length of the data block DMX levels above channel no 512 would have been polled, the header byte announces a smaller number of bytes, which is equal to the effectively polled number of DMX levels.

If automatic messages for received DMX levels are active, care has taken not to confuse both types of messages. Each of both message types is sent as a coherent block - but they may follow closely one after the other.

Binary set of MIDI channel messages compatible commands

Every MIDI channel messages starts with a status byte, followed by 1 or 2 data bytes.

Only at the first byte of every message, the **status byte, the most significant bit 7 is set**. So it is clearly different from all ASCII commands.

At all subsequent data bytes, the most significant **bit 7 is cleared**, else their value is context dependend.

Every MIDI style command is not accepted and evaluated before the corresponding number of data bytes has been received. Incomplete commands or surplus data bytes are ignored. If data bytes are missing, decoding of a new command is started from the next valid status byte.

This way, some additional data security is gained in contrast to the previously described style of binary commands. A disadvantage of MIDI compatible commands is their restriction to 7 bits of effective data content per byte. So generally a rearrangement of bits is necessary for transfer of bytes with 8 significant bits.

An example for recalculation of a DMX level into binary command bytes see below.

Select a DMX channel (=SLOT) and set it to a given level

The division **. / 2** is performed following the mathematical rules if "Integer Division", i.e. there are no post point digits, the remainder is dropped and the result is always rounded down to the next lower integer.

DMX channel between 1 and 127

DMX level even number

status byte= 144 (hex 90)
1st data byte = DMX channel
2nd data byte = DMX level **. / 2**

DMX level odd number

status byte= 176(hex B0)
1st data byte = DMX channel
2nd data byte = DMX level **. / 2**

DMX channel between 128 and 255

DMX level even number

status byte = 145 (hex 91)
1st data byte = DMX ch - **128**
2nd data byte = DMX level **. / 2**

DMX level odd number

status byte = 177 (hex B1)
1st data byte = DMX ch - **128**
2nd data byte = DMX level **. / 2**

DMX channel between 256 and 383

DMX level even number

status byte = 146 (hex 92)
1st data byte = DMX ch - **256**
2nd data byte = DMX level **. / 2**

DMX level odd number

status byte = 178 (hex B2)
1st data byte = DMX ch - **256**
2nd data byte = DMX level **. / 2**

DMX channel between 384 and 511

DMX level even number

status byte = 147 (hex 93)
1st data byte = DMX ch - **384**
2nd data byte = DMX level **. / 2**

DMX level odd number

status byte = 179 (hex B3)
1st data byte = DMX ch - **384**
2nd data byte = DMX level **. / 2**

DMX channel equal to 512

DMX level even number

status byte = 144 (hex 90)
1st data byte = 0
2nd data byte = DMX level *.I. 2*

DMX level odd number

status byte = 176 (hex B0)
1st data byte = 0
2nd data byte = DMX level *.I. 2*

Example: Assume DMX channel 450 has to be set to level 101. For that in the table above is looked up in block "DMX channel between 384 and 511". Because the DMX level to be set is odd, we have to look in the right column.

There you find: status byte = 179,
1st data byte = (450-384) = 66,
2nd data byte = (101 integer divided by 2) = 50

Pre-increment addressed DMX channel ("SLOT") and set it to a given level

(useful to **set a block of subsequent** DMX channels, corresponds with the ASCII 'comma' command)

DMX level even number

status byte = 208 (hex D0)
data byte = DMX level *.I. 2*

DMX level odd number

status byte = 209 (hex D1)
data byte = DMX level *.I. 2*

In addition, there is a possibility to transfer levels to subsequent DMX channels without repetition of the status byte ("Running State").

While this method is common use in MIDI environment, **here it is absolutely necessary to finish the "running state" with a final status byte, else following ASCII commands will be misinterpreted.** In total, this procedure results in an efficient method to set a number of subsequent DMX channels **with fade transition.**

Following commands activate the running state, else they are equivalent to the ones described above:

DMX level even number

status byte = 210 (hex D2)
data byte = DMX level *.I. 2*

DMX level odd number

status byte = 211 (hex D3)
data byte = DMX level *.I. 2*

All MIDI compatible status bytes else terminate the running state automatically, especially this applies for all single bytes in the range 246 (hex F6) to 255 (hex FF).

Example: To provide DMX channels 1 to 5 with DMX levels 10,20,30,40,50, channels 6,7,8 with levels 55,65,75 and channel 9 with level 0, following sequence of decimal bytes has to be sent: 144,1,5,210,10,15,20,25,211,27,32,37,210,0,255.

The last byte 255 forces the running state to be terminated, so following ASCII commands will be interpreted correctly.

This sequence of bytes is equivalent: 144,1,5,210,10,15,20,25,211,27,32,37,208,0.

Set the FADETIME

status byte = 160 (hex A0)

1st data byte = **1**, then

2nd data byte = (0 - 127) fade time in 1/10 second units (0 -12,7 seconds).

1st data byte = **2**, then

2nd data byte = (0 - 127) fade time in 1/10 second units

plus 10 seconds (setting range 10,0 - 22,7 seconds).

1st data byte = **3**, then

2nd data byte = (0 - 127) fade time in 1/10 second units
plus 20 seconds (setting range 20,0 - 31,9 seconds)..

1st data byte = **4**, then

2nd data byte = (0 - 127) fade time in **quarter seconds** (0 - 31,8 sec.)

This variant makes it possible to handle the complete range of fading time with a single MIDI controller.

remaining quarter seconds are internally rounded up to the next higher step of 1/10 seconds, i.e. 0.25 to 0.3 and 0.75 to 0.8.

Comment: The actual value of FADETIME is copied into the corresponding fader resource when the fade process is started. Immediately after FADETIME can be modified without retroactivity on running fade processes. Any number of fade processes can be active simultaneously.

Actually recalculated fader levels are written into the transmit buffer.

Load presets (= lighting scenes) with :

status byte = 160 (hex A0)

1st data byte = 96 (hex60): basic value 0 = load preset 0-127

97 (hex61): basic value 128 = load preset 128-255

98 (hex62): basic value 256 = load preset 256-383

2nd data byte = 0 to 127 (hex7F) = preset no. minus basic value

Special cases: load preset partially

(new from May 2010 / revision number 83)

1st data byte = 99 to 101: basic value 0, i.e. partially load preset no 0 - 127

99 (hex63): load DMX channels 129 - 256 from preset to DMX OUT channels 1 - 128

100 (hex64): load DMX channels 257 - 384 from preset to DMX OUT channels 1 - 128

101 (hex65): load DMX channels 385 - 512 from preset to DMX OUT channels 1 - 128

1st data byte = 102 to 104: basic value 128, i.e. partially load preset no 128 - 255

102 (hex66): load DMX channels 129 - 256 from preset to DMX OUT channels 1 - 128

103 (hex67): load DMX channels 257 - 384 from preset to DMX OUT channels 1 - 128

104 (hex68): load DMX channels 385 - 512 from preset to DMX OUT channels 1 - 128

1st data byte = 105 to 107: basic value 256, i.e. partially load preset no 256 - 383

105 (hex69): load DMX channels 129 - 256 from preset to DMX OUT channels 1 - 128

106 (hex6A): load DMX channels 257 - 384 from preset to DMX OUT channels 1 - 128

107 (hex6B): load DMX channels 385 - 512 from preset to DMX OUT channels 1 - 128

1st data byte = 108 to 111: basic value 0, i.e. partially load preset no 0 - 127

108 (hex6C): load DMX channels 1 - 128 from preset to DMX OUT channels 1 - 128

109 (hex6D): load DMX channels 129 - 256 from preset to DMX OUT channels 129 - 256

110 (hex6E): load DMX channels 257 - 384 from preset to DMX OUT channels 257 - 384

111 (hex6F): load DMX channels 385 - 512 from preset to DMX OUT channels 385 - 512

2nd data byte in all cases = 0 to 127 (hex7F) = preset no. minus basic value

any other levels in the preset remain unchanged.

This feature is intended to get more freedom with small installations.

Save, store presets (= lighting scenes) permanently with :

status byte = 160 (hex A0)

1st data byte = 112 (hex70): basic value 0 = save preset no. 0-127

113 (hex71): basic value 128 = save preset no. 128-255

114 (hex72): basic value 256 = save preset no. 256-383

2nd data byte = 0 bis 127 (hex7F) = preset no. minus basic value

Special cases: store preset partially

(new from May 2010 / revision number 83)

1st data byte = 115 to 117: basic value 0, i.e. partially store preset no 0 - 127

115 (hex73): store DMX channels 1-128 from DMX OUT to preset channels 129 - 256

160 (hex74): store DMX channels 1-128 from preset to DMX OUT channels 257 - 384

117 (hex75): store DMX channels 1-128 from preset to DMX OUT channels 385 - 512

1st data byte = 118 to 120: basic value 128, i.e. partially store preset no 128 - 255

118 (hex76): store DMX channels 1-128 from DMX OUT to preset channels 129 - 256

119 (hex77): store DMX channels 1-128 from DMX OUT to preset channels 257 - 384

120 (hex78): store DMX channels 1-128 from DMX OUT to preset channels 385 - 512

1st data byte = 121 to 123: basic value 256, i.e. partially store preset no 256 - 383

121 (hex79): store DMX channels 1-128 from DMX OUT to preset channels 129 - 256

122 (hex7A): store DMX channels 1-128 from DMX OUT to preset channels 257 - 384

123 (hex7B): store DMX channels 1-128 from DMX OUT to preset channels 385 - 512

2nd data byte in all cases = 0 to 127 (hex7F) = preset no. minus basic value

any other levels in the preset remain unchanged.

This feature is intended as a counterpart to the corresponding partial load command to produce and test "long" presets with small lamp configurations.

Set the masterfader with :

status byte = 160 (hex A0)

1st data byte = 7 (changed from Dec 2010 / revision number 90)

then 2nd data byte = 0-127 (hex 7F) masterfader setting in %

1st data byte = 8 (changed from Dec 2010 / revision number 90)

then 2nd data byte = 0-100 (hex 64) masterfader setting in 100-200%
(100 greater than entered in the data byte)

see **comment** at the equivalent ASCII command "M"

Address a DMX channel (write into SLOT register) with:

status byte = 160 (hex A0)

1st data byte = 80 (hex 50): address DMX channels 1-127

then 2nd data byte = DMX channel 1-127

1st data byte = 81 (hex 51): address DMX channels 128 - 255

then 2nd data byte = DMX channel minus 128

1st data byte = 82 (hex 52): address DMX channels 256 - 383

then 2nd data byte = DMX channel minus 256

1st data byte = 83 (hex 53): address DMX channels 384 - 510

then 2nd data byte = DMX channel minus 384

Poll DMX receive buffer from DMX channel ="SLOT" with:

status byte = 160 (hex A0)

1st data byte = 40 (hex28) message even DMX levels with status byte 144-147

message odd DMX levels with status byte 176-179

(like command to set DMX levels, see above.)

1st data byte = 41 (hex29) message the level of the first polled DMX channel with
status byte 144-179 (format see above)
and all following with status byte 208/209 (format see above)

1st data byte = 42 (hex2A) message the level of the first polled DMX channel with
status byte 144-179 (format see above)
and all following with status byte 210/211 (format see above)

the 2nd data byte is coded independent of the 1st data byte:

2nd data byte = number of DMX channels to be polled (1 to 128).

At all format variants the **end of the transferred data block** is marked with a
terminating byte = 247 (hex F7) = MIDI EOX.

Set RGB hue (spectral color) with:

status byte = 160 (hex A0)

1st data byte = 76 (hex 4C): command specifier

2nd data byte = hue 0 - 127

This results approximately in following colors. Intermediate hue values will result in intermediate colors:
2nd data byte= 0:red, 22:yellow, 43:green, 64:cyan, 85:blue, 106:magenta, 127:red again.

In correspondence with the model of the driven lamp and setting of saturation and brightness the
resulting color tone may differ somewhat. See comment at ASCII command H.

Set RGB color saturation with:

status byte = 160 (hex A0)

1st data byte = 77 (hex 4D): command specifier

2nd data byte = saturation 0 - 127 (internally multiplied by 2, see ASCII command W)

Set RGB brightness with::

status byte = 160 (hex A0)

1st data byte = 78 (hex 4E): command specifier

2nd data byte = brightness 0 - 127 (internally multiplied by 2, see ASCII command ")

Appendix A

Concept and data format of DMX512 transmission:

To understand the function of the USB / DMX Control Box and its commands, you should roughly know the concept of internal memories for received and transmitted DMX signals.

For each of the 512 DMX channels three specific memory cells are provided: transmit buffer, receive buffer and action memory.

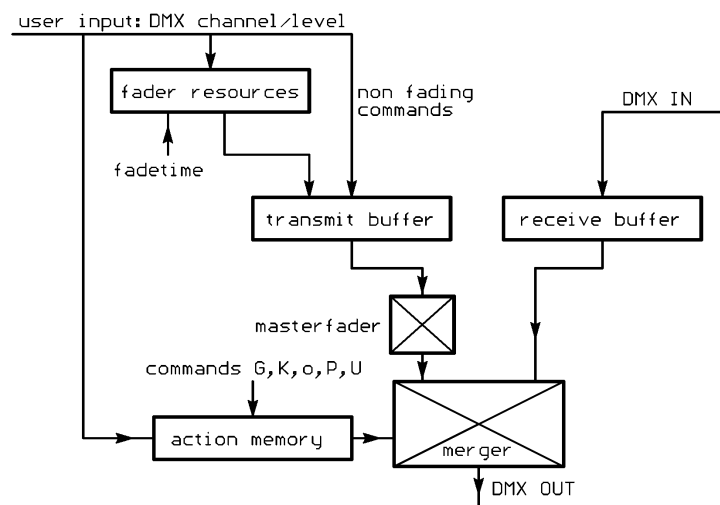
The received DMX signal is permanently written in realtime into the receive buffer.

DMX level data entered by the user are preprocessed if necessary (e.g. a fade process initiated) and then written into the transmit buffer.

The expression "transmit buffer" does not mean that its content is transmitted unconditionally: **Before transmission of any next DMX byte the merger routine looks into the action memory if this byte shall be taken from the receive buffer or from the transmit buffer.**

The **masterfader** modulates the DMX data bytes as a simple signal processor when they are transferred from the **transmit buffer** into the DMX transmitter hardware. This way a complete lighting scene may be "stretched" globally in a simple way.

Internally stored DMX data are not manipulated by the masterfader. The masterfader does not influence any data which are retransmitted from the receive buffer.



The philosophy of commands for writing into the transmit buffer is based on the cooperation of three registers:

- **SLOT register** (written by command "S"),
- **DMX level register** (written by commands "V", comma, "^" and "_") and the
- **FADETIME register** (written by command "T").

The **SLOT register** stores the actually addressed DMX channel (= physical time slot 1 to 512 within the DMX cycle), which shall be written or read subsequently by the **USB command interface**.

For any DMX channel a complete fade process can be performed. The fade time to be used is set in the **FADETIME register**. This fade process is completely controlled by the firmware of the device. Fade jobs are automatically assigned to internal resources. The actual value of FADETIME is copied into the respective fade process when the fade process is started. Straight after then FADETIME can be changed for the next command without influence on running fade processes.

Last not least, a **LOOP register** is implemented, which stores the actual actual number of DMX channels to be transmitted under user control in each DMX packet.

Parameters (i.e. numeric values) for SLOT, FADETIME and LOOP always have to be entered in **decimal format** and are fed back in decimal.

The parameter of **MASTERFADER** is always entered in percent (without postponed % sign) in the range 0 to 200.

Parameters for DMX levels may be entered as a **decimal numbet or as a hex number** (see "system commands") and are messaged in this active number base. **DMX levels are internally stored as one byte (8 valid bits) i.e. can take values between 0 and 255.**

Every command causes its parameter to be written into the corresponding one of the registers listed above. Depending on the type of command, when all information is entered the intended action is started using the actual content of ALL OF THESE REGISTERS. For this reason, the order of entering data may influence the result. In general **the order FADETIME, SLOT, DMX level should be followed**, but most times not all of the registers have to be updated.

After power-up or reset, SLOT is initialized with "1", DMX levels and FADETIME are set to "0", the MASTERFADER is initialized with 100% and LOOP is started with "512" or value read from preset no.0.

Entries to the action memory - which controls the merge behaviour - are controlled with the commands G,K,o,P,U.

Appendix B

Data format DMX512

The "DMX512" standard assumes that a bus master transmits DMX data packets on a DMX data bus in a permanent cyclic repetition. This bus line is looped from the transmitter to the next receiver and from there to the next receiver and so on in a linear bus topology. **Every DMX receiver connected at the bus is permanently supplied with actual control data for all receivers**, no matter if these data have been changed in the meantime.

Every **DMX packet transmission cycle starts with a special reset sequence and a start byte** as header. Subsequently all data bytes are transmitted in 8 bit serial format to all potential receivers. This way every transmitted DMX byte is assigned with a specific time slot within every DMX packet and it can be adressed with the number of this time slot. Therefore the "adresses" or "channels" in a DMX data packet are called "slots". Per DMX slot one data byte is transmitted. The value of this data byte (rage 0 to 255) describes the intensity of a lamp or the position of a "moving head". **Throughout this manual we use the expression "DMX level" or "DMX value" to describe this byte value.**

Adressing of receivers is defined by the time position ("slot") of their data bytes within the cyclic DMX package. By means of a code switch or something similar a number between 1 and 512 can be selected at every DMX receiver. The receiver scans the count of slots during every DMX packet and starts to read out the slot bytes starting from the slot number selected with its switch. The number of evaluated bytes depends on the specific function of the receiver. If, for instance, the switch of a certain receiver is set to 28, this receiver will read the 28th, 29th, 30th ... byte following after the start byte out of the DMX data stream during every DMX cycle. These data are not removed out of the DMX data stream. But all receivers with a different setting of their code switch do ignore them. Depending on the setting of their code switches, several receivers can get the same data - intentionally or by mistake.

Due to its design the "DMX512" standard does not offer inherent error checking and error correction of transferred data and does not supply a usefully standardized feedback channel from receiver to transmitter. While slowly changing actuators like bulb lamps or servo motors are driven, single faulty bytes almost have no effect on the visual behaviour, because they are corrected in the next cyclic transmitted DMX packet with high

probability. **Very critical however is to release singular switching events via the DMX-bus.**

For these reasons the the use of DMX control is explicitly FORBIDDEN together with all safety critical applications, where malfunction could result in personal injury oder noticeable material damage !

To install a DMX512 bus, preferably a special 2 wire twisted and shielded "RS-485" data cable is recommended, which unfortunately is quite expensive and only available from special providers for industrial control. CAT5 ethernet cable has been proved to be a good material for DMX installation, too. For solid installations we have made good experience with shielded ISDN cable, which is traded in germany as type JY(St)Y. The version with 0,8mm wire diameter should be used if available. Using bus lengths up to 100 m, a high quality 2 wire microphone cable will do.

Long bus lines have to be terminated at the most distant receiver. Termination means: DMX+ and DMX- are connected with a resistor of 120 Ohm.

Normally the bus line is "looped through" from receiver to receiver. Most DMX equipment is supplied with a DMX IN and a DMX OUT connector, both are physically wired together inside. For this kind of installation you should have available a 5-pin male XLR-connector prepared with a built-in resistor, which is plugged into the DMX OUT of the last DMX receiver in the chain.

Inside the USB / DMX Control Box the DMX IN signals are electronically refreshed before retransmission. This means, even when the DMX data are "looped through", the **USB / DMX Control Box has to be regarded as terminal device at the bus line. The termination resistor is already built in, no external termination is necessary!** If in special cases this termination is not wanted, a jumper is provided inside the box (close to the DMX IN socket) which has to be pulled then. To open the box: remove 4 screws and pull off the upper part of the metal cabinet.



Cinetix Medien und Interface GmbH
Gemuendenerstr. 27 D-60599 Frankfurt Germany
Phone: +49-69-68 51 05 Fax +49-69-68 600 409
<http://www.cinetix.de/interface/english/>

* Right of technical modifications reserved.

* This description is for information only, no product specifications are assured in juridical sense.

* Trademarks and product names cited in this text are property of their respective owners.