

Ethernet / DMX512 Control Box

User Manual for products delivered from May 2011

Status 05 May 2011

New: UDP default mode is Multicasting (e.g. MIDI over Ethernet), XMODEM protocol) If you need manuals for earlier models, please order by email: cinetix@t-online.de. Please add date of delivery or revision number.

The Ethernet / DMX512 Control Box is preferably designed to control a DMX512 digital lighting bus by means of an Ethernet TCP or UDP connection and to check or poll levels of another DMX 512 bus.

To manage this, the Ethernet / DMX-512 Control Box is operated as TCP server - or as an UDP serving device - selectable with a rotary switch.

In **it's native mode of operation**, a set of commands based on **short ASCII text prases** is implemented. An alternative set of **binary commands** is available, which is based on MIDI channel messages. All commands are documented, no proprietary drivers are needed.

This **open concept** makes the Ethernet/DMX Control Box especially **useful to enhance possibilities of industrial PLC, programmable media controllers and user's software.**

The signal transmitted at **DMX OUT may be merged** (switched in realtime) configurable channel by channel **between the level set by Ethernet command and the one received at DMX IN.** Further merge options: least or greatest of both is transmitted or "last changed takes precedence". Up to 384 presets (=lighting scenes) may be stored and recalled with adjustable fade-in time.

To evaluate or analyse the signal received at DMX IN **different tyes of automatically reported messages can be activated**, which are sent as TCP or UDP data packets as soon as any activated channel at DMX IN changes more than a user settable threshold or takes specific values. Because the UDP protocol works connection-less and allows broadcast and multicast messages, these messages can be received by several control stations (clients) and a single controller can receive messages from a number of automatically serving UDP transmitters.

Operation of the Ethernet / DMX512 Control Box is designed to be essentially free of interference with other communication running over the same network. Low level control of the DMX bus and complete fade processes are completely managed inside the Control Box. This way commands are quite compact and the Ethernet is burdened with relatively low data traffic.

Additionally, the Ethernet / DMX Control Box can be switched to some special modes of operation:

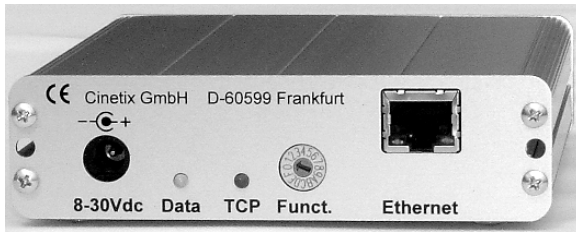
--- **Art-Net™ node: Full support to send to one universe DMX OUT and read from one universe DMX IN.** The "Subnet" and "Universe" for DMX OUT and DMX IN are configurable via TCP/Telnet, UDP or serial interface. In addition to the standard Art-Net network IP configurations, a custom IP configuration is available, which makes the box smoothly compatible and cooperative with existing small class C networks.

--- **Serial communication through TCP or UDP:** bytes received via Ethernet are retransmitted via serial port (RS-232, RS-422, MIDI) and vice versa serially received bytes are transferred via Ethernet to a corresponding client. **In addition to standard PC baudrates especially the MIDI interface is supported.** An optoisolated MIDI -IN is provided by hardware, which may easily be reconfigured as a standard RS-232 input. This way industrial PLC, media controllers or MIDI sequencers - which don't have their own Ethernet access - are brought to communicate over Ethernet.

Optionally a set of transformations between binary data into their ASCII notation and vice versa is selectable.

It is **NOT allowed** to use this instrument together with any safety critical application, where malfunction could result in personal injury oder noticeable material damage !

Hardware



front panel 11 x 3 cm, depth 11 cm

The **power supply** is designed for unregulated or regulated **DC supplies with output voltages 8 to 30 Volt** and min 250 mA output. A simple in-plug supply with "europlug" is delivered along with the DMX Control Box.

The DC input is designed for a concentric low voltage connector, external 5,0 - 5,5mm, internal 2,1mm. The **positive polarity** has to be connected with the **inner contact!** Internally the DMX Control Box is **protected against wrong polarity**: If wrong, the device is not powered.

The **front panel rotary switch "Funct."** selects between different modes of operation:

Position	Function
0	Configuration with ASCII text via TCP(Telnet)
1	TCP(Telnet) <--> DMX server, only ASCII commands in 7bit format
2	TCP <--> DMX server, ASCII commands and binary commands
3	TCP <--> DMX server, binary commands (MIDI message style)
4	UDP <--> DMX server, ASCII commands and binary commands
5	UDP <--> DMX server, binary commands (MIDI message style)
6	Art-Net node, broadcast IP 2.xxx.xxx.xxx, subnet mask 255.0.0.0, port 0x1936
7	Art-Net node, broadcast IP 10.xxx.xxx.xxx, subnet mask 255.0.0.0, port 0x1936
8	Art-Net node, custom IP and subnet mask, port 0x1936
9	transparent TCP server for 1:1 data transfer (MIDI and RS-232)
A	transparent TCP server with byte format conversion
B	transparent UDP node (client & server) with byte format conversion
C	transparent UDP node (client & server) for 1:1 data transfer (MIDI and RS-232)
D	TCP client connection control by single byte control characters or MIDI SysEx
E	Configuration with ASCII text via UDP multicast
F	Configuration with ASCII text via RS-232, 19200 Baud

The red/green **dual LED "Data"** next to the power connector shows the presence of supply voltage, the selected mode of operation, reception of commands and transmission of DMX level change messages or other data via Ethernet.

Modes "TCP – DMX server", "UDP – DMX server" and "Art-Net node":

- **During DMX transmission and merge operation with internal clock the LED shines green.** When commands are transferred via Ethernet the LED darkens for a moment. During automatic transmission of level changes at DMX IN the LED turns into yellow-orange for a moment.
- **During DMX transmission and merge operation with external synchronisation the LED shines yellow-orange.** When commands are transferred via Ethernet the LED darkens for a moment. During automatic transmission of level changes at DMX IN the LED turns into red for a moment.
- **In DMX reception only mode (received DMX signal is looped to the DMX transmitter by hardware) the LED shines red.** When commands are transferred via Ethernet the LED darkens for a moment. During automatic transmission of level changes at DMX IN the LED turns into yellow-orange for a moment.
- During a reset procedure the LED goes off for about one second.

Mode "Configuration":

The LED shines red and darkens for a moment when data are received via the selected configuration channel (Ethernet or serial)

Modes "TCP client" or "transparent":

While communication is in idle state, the LED shines yellow-orange. When data packets are transmitted to the Ethernet, it blinks green. When data packets are received from Ethernet, it blinks red. The LED may turn off while data are transferred in both directions simultaneously.

The blue LED "TCP" placed right beside is lit as long as the TCP channel is in "connected" state. In the UDP mode it flashes blue when data are exchanged.

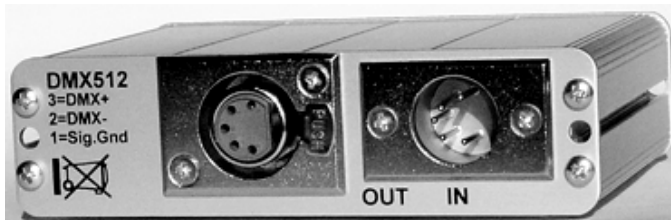
At the right side of the front panel the **RJ45 Ethernet socket** is placed:

The connection to the network is established with a standard Ethernet patch cable.

At the lower part of the RJ45 socket 2 small LEDs are integrated. The green LED is the "Link LED", it shines as long as a physical connection with the network is present. The yellow/orange LED goes off while data are transferred over the network.

According to the type of the connected network a connection with 10MBit/s or 100MBit/s is established.

Pinout of the DMX512 interface:



DMX IN: 5 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 (see appendix B, too). If the termination has to be taken away in a special situation: remove the top part of the enclosure (4 screws) and pull of the jumper near the DMX IN connector.

DMX OUT: 5-pin female XLR connector

DMX OUT	XLR connector
Shield, signal ground	Pin 1
DMX- transmitter	Pin 2
DMX+ transmitter	Pin 3
Serial data input + (MIDI compatible)	Pin 4
Serial data input - (MIDI compatible)	Pin 5

The DMX output is only active in the rotary switch positions 1 to 8. In all other positions the DMX OUT socket is configured as a serial port.

In configuration mode (rotary switch position F) the interface is activated with 19200 baud.

The active baud rate at rotary switch positions 9 to D can be set to 9600, 19200, 38400, 57600, 115200 baud as well as to MIDI (31250 baud) during the configuration session.

By means of an adaptor cable (build your own or optionally available) a data connection to a standard RS-232 interface can be set up. How to build this cable by your own see appendix D.

If you use an **alternatively wired adaptor**, a **standard MIDI interface** is provided. The serial data input (Pin 4 and 5) is galvanically isolated with an opto coupler in compliance with the MIDI standard. How to build this cable by your own see appendix D

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!

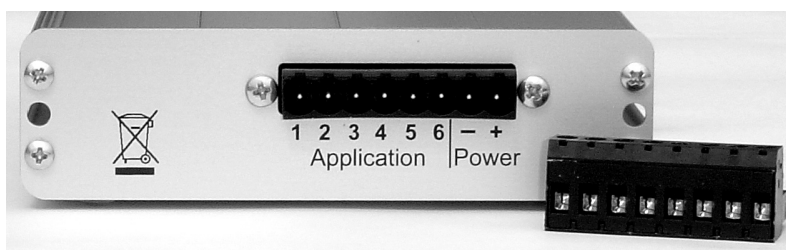
The DMX standard demands that **the complete bus, driven by the DMX transmitter** should be connected with ground or "earth" **at one single point**. In most cases this will be the protective earth of your mains power line. **This means: a galvanically isolated DMX transmitter particularly makes sense, 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 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 Ethernet / DMX Control Box and thus with the signal ground of the circuit board, too. Use of DMX cables, whose shielding and/or signal ground is connected with the metal body of of the DMX plug, will cancel the galvanic isolation! **The shield / signal ground 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:



This version is **preferably intended to be used as DMX server**. When used in other modes of operation, where the XLR connectors provide a serial interface, a fully equipped RS-422 interface is available, which is quite well prepared for connection to many PLC controllers. Connection to RS-232 or MIDI interfaces however, is only recommended for configuration etc. when no Ethernet access is available. A wiring proposal is presented in appendix D.

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 -	Power Supply Minus (= Device Ground)
Power +	Power Supply Plus

Due to the galvanic isolation between DMX IN and DMX OUT, the 3 ground terminals have no connection among each other. The power supply terminals are internally directly connected with the power connector at the front panel and protected against wrong polarity.

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

Configuration of the Ethernet / DMX512 Control Box

Before practical use the device has to be configured. This may be achieved via Ethernet as well as with an RS-232 adaptor cable.

Configuration via Ethernet (Telnet) when the rotary switch is in **position "0"**:

A PC-sided network configuration which allows access to **IP address 192.168.0.240 / port 23** is required (default at delivery, IP may be changed to user values after configuration).

The Control Box is connected to the network and power is supplied. At the PC a Telnet terminal software is started and a TCP connection is established to the actual IP address (see above) / port 23. "Hyperterminal" which is included in Windows, provides a good Telnet terminal. Useful alternatives are the freeware programs "Teraterm" and "PuTTYtel".

After the IP address of the Ethernet/DMX Control Box has been changed, subsequent configuration sessions will take place under this new IP address. Independent of the port set with the command P, TCP configuration will always take place at port 23.

Configuration using RS-232 (rotary switch **position "F"**):

Using a special adaptor cable (see appendix D) the serial port of the PC is connected with DMX-OUT. On the PC a terminal software (like Hyperterminal or Teraterm) is started with **19200 Baud**. With the RS-232 connection reconfiguration is always possible if the actual network settings are unknown.

Configuration using UDP broadcast (rotary switch **position "E"**):

This method is useful too, if the actual IP setting is unknown. No cable has to be assembled.

The simple UDP terminal software "UDPTERM" can be downloaded from the Cinetix website. For configuration by UDP the Ethernet / DMX Control Box always uses multicast IP 225.0.0.37 and port 21928, independently of its stored UDP configuration. About 4 seconds after power on or turning the rotary switch, the Box automatically transmits its own IP address. If "UDPTERM" is configured for multicast IP 225.0.0.37 and port 21928 too, this address will be received and displayed. An advantage of the multicast connection lies in the fact, that it is laid over the existing network IP, i.e. this has not to be changed temporarily for configuration.

Devices delivered before May 2011 have to be configured by UPD broadcast.

The configuration procedure is identical for all of these connection methods:

In some cases the terminal software has to be configured to insert a "Line Feed" after every received "Carriage Return".

After a 'L' command has been typed at the terminal, a list of commands characters and the necessary input to change the corresponding parameter is returned:

```
List of Setup Commands:
?          list setup entries
G hh.hl.lh.ll set gateway IP
I hh.hl.lh.ll set IP of this module
M hh.hl.lh.ll set subnet mask of this module
# hh.hl.lh.ll set TCP default destination IP (client only)
P          set TCP port number (client&server)
N hh.hl.lh.ll set UDP default destination IP (1:1, broadcast, multicast)
D          set UDP destination(Tx)port number
S          set UDP source(Rx)port number
Z          show IP of UDP msgs:0=No 1=ASCII 2=SysEx(client&transparent)
A          set Art-Net 'Subnet'for this box (0-F)
U          set Art-Net "Universe" for DMX OUT (0-F)
V          set Art-Net "Universe" for DMX IN (0-F)
K          config TCP/UDP<->MIDI/RS-232/RS-422 Konverter
B          set MIDI/RS-232/RS-422 baudrate
C          set MIDI channel (1-16)
Q          set TCP disconnect code (TCP client)
T          set TCP/UDP packet delay(0-254ms). 0=send immediately
E          set server idle disconnect timeout (1-255s). 0=permanent
F          new line: 0=CR, 1=CR+LF
X          set DMX timing (SPECIAL!! see manual)
```

Commands and their subsequent parameters may be entered in arbitrary order or as a single item. Every input is finished with a "Carriage Return" (= dec13=hexD) or by entering next command letter. All settings are stored permanently inside the Ethernet / DMX Control Box.

The command "?" displays the actual parameter settings as a list. Hexadecimal numbers are marked with a preceding dollar sign \$

Commands **G,I,M** and **P** define the **network address of this Ethernet/DMX Control Box.**

Ethernet IP addresses are entered in common style as 4 decimal numbers 0 to 255 separated by periods. **Port numbers** are entered as a decimal number 0 to 65535. Dynamic IP configuration is not supported by the DMX Control Box. If a typing error happens repeat the complete command. Backspace is not supported.

To make the first operation as easy as possible, **at delivery following default parameters are configured**, which are compatible with many small networks:

IP address 192.168.0.240, subnet mask 255.255.255.0, TCP port 23 (=Telnet standard).

The TCP **port number** is always the same for received and transmitted data packets and used in server and client mode. It is set with **command P**.

When operated as a **TCP client**, the **command code #** sets the IP address of the server which will be addressed by default. More flexible options see part 4 of the manual.

The **parameter set by command Q** specifies the control code **to establish and to terminate a connection** by the client.

The parameter of Q is entered as a hex byte without preceding "\$" or "0x". It's optimum value is application dependent. When control is performed with ASCII commands, in most cases a value of the range 1 to 24(hex18) is recommendable. These values can be entered on a keyboard by typing a letter while the CTRL key is pressed. The default termination/disconnect code is CTRL-N = dec14(hexE)

A standard connection is established with (termination code + 2) i.e. default is CTRL-P =16(hex10)

It has to be taken into account that control software may use some of these "nonprintable" ASCII codes for hotkeys and terminal control. With MIDI or binary applications however, it is recommend to set the parameter = 0 to get a binary-proof disconnect.

Detailed description of connection handling and further options see part 4 of the manual

When operated as an **UDP node** (rotary switch positions 4,5,B,C), the **commands N and D** (destination) set the default network address and port where UDP datagrams are sent to. The command S (source) sets the port for reception of UDP messages. Normally D and S should be set to the same number.

In complex installation with a number of UDP serving nodes and controllers, the flexibility may be enhanced when both are set differently.

Three **UDP special cases** should be mentioned:

--- At delivery, the IP address for UDP is set to 225.0.0.37 and both port numbers to 21928 .

With this **multicast-configuration** the Control Box can immediately be operated manually with the simple terminal console "UDPTERM", which is available for download from the Cinetix website. Furthermore: this default setup can be directly used with the MIDI driver ipMIDI to control the device from a MIDI sequencer or MIDI file player.

--- **UDP datagrams can be transmitted by broadcast**. To use this mode, in simple class C networks the **last IP byte is set to 255** with the configuration **command N**

--- If the last **IP byte is set= 254**, the Ethernet / DMX Control Box always **responds to the IP address of the last received datagram** (i.e. to the sender of the last command)

By means of these special cases, the Ethernet / DMX Control Box can serve a number of "clients" (i.e. controllers) during the same session.

When the parameter of command **Z is configured =1** the UDP node returns the IP address of the datagram sender as ASCII text, but only once after this has been changed with respect to the previous datagram. After configuration with command **Z2** , the responded ASCII text is embedded into a MIDI SysEx frame. This feature is switched off with command **Z0**, which is configured as default.

The **Art-Net mode of DMX control** is activated with **front panel positions 6,7 or 8** which implies appropriate IP and port setting. In addition, the Art-Net specific "**Subnet**" has to be configured by **command A**. Furthermore the "**Universe**" for **DMX OUT** is configured with **command U**, the "**Universe**" for **DMX IN** is configured with **command V**.

The **Subnet** is valid for the complete module, while DMX OUT (configured with command U) and DMX IN (configured with V) may address different **Universes**. Each item is entered as a single hex digit ("nibble") in the range 0 to F without leading \$ or 0x. Default is "0" for all parameters.

If supported by the Art-Net "server", reconfiguration via Ethernet is possible during operation with the ArtAddress protocol.

To avoid possible irritation: in some Art-Net context a byte combined of Subnet (high nibble) and Universe (low nibble) - each set as described above, is referred as a "Universe".

Parameters set with B and C are only relevant, when a serial or MIDI interface is connected to DMX-OUT. The MIDI baud rate has to be entered as text "31250".

Setting of **parameters T and E** however influences the operational behaviour in any application: Default at delivery is T = 30ms and E = 255s.

To achieve an acceptable efficiency on the Ethernet while operating with binary MIDI channel messages, data packets are not transmitted per MIDI command but in timed sequences. The timing interval is set with command "T" in milliseconds. A lower value increases precision of timing at costs of network burden. If parameter T is set to 0 , the buffer is emptied via Ethernet as frequently as technically possible. If predictable by context, matching clusters of bytes (MIDI channel messages for example) are transmitted within one packet.

Independent of the T setting, the buffer is transmitted via Ethernet when the last entered byte is "line Feed" (parameter F=0) or "Carriage Return" (parameter F=1) or if MIDI specific EOX is sent (which terminates MIDI SysEx-messages) or when the buffer is full.

The parameter "E" controls the timing of auto-disconnect in all modes of TCP server operation.

While at the client side the presence of a control software or human operator may be assumed to manage and control the TCP connection, this is not evident with a remotely located server. Due to

certain errors of connection (e.g. client terminates the connection irregularly when power fails) deadlock situations are possible if the server does not break after a while. Whenever a data packet is received, the server resets timeout. This way a permanent connection is maintained while commands are received. If this parameter is set to zero, the server holds the connection permanently until it is terminated by the client.

The **command F** controls the **new line** sequence (= delimiter of all sent feedback and messages), which is sent by the Ethernet / DMX Control Box to a client (terminal or other controller):

As default at delivery (parameter F=1) the sequence "Carriage Return + Line Feed" is sent, which is compatible with the default setting of most terminal software. To obtain a more compact output format, it may be changed to "Carriage Return" only (parameter F=0). Within received commands, the Ethernet / DMX Control Box only evaluates "CR". "LF" is ignored.

The **parameter X** modifies the timing of the DMX transmission cycle.

This is only necessary and recommended to patch extreme synchronisation problems with other DMX equipment and **should not be applied without having contacted Cinetix**. To avoid changes of DMX parameters by accident, this command must be entered **twice** in immediate sequence!

Following the command code X another byte composed of 2 hex digits (nibbles) 0...F is entered in text form. The entered value is stored permanently and reloaded during every system start.

1st hex digit (high nibble): duration of the DMX reset pulse: 90us basic value + (nibble value * 8 us). Thus, the range of setting is 90us to 210 us.

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

Exception: The command X00X00<enter> resets DMX timing permanently to factory default.

After this basic setup the final mode of operation has to be selected with the rotary switch. When the switch is turned, after a short while the Ethernet/DMX Control Box performs a reset and reorganises itself in correspondence with the new selected mode.

First operation

Subsequently hints are given to use the Control Box in its primarily intended **mode of operation TCP <--> DMX server** or **UDP <--> DMX server**. In these modes of operation a **DMX lighting bus is controlled via Ethernet**.

User informations related to **the other modes of operation see part 4 of this manual**

A TCP server is initialized **passively waiting in the "listen" state**. At the other side of the network a TCP client (=control device) is necessary, which actively establishes and terminates the connection. After the connection is terminated, the TCP <--> DMX server automatically falls back into the "listen" state and waits for a new connection request by a client. A connection may be installed permanently or only to submit a single command. During "connected" state, data exchange is exclusive between client and server.

The **UDP protocol** however is "**connection-less**", only datagrams are sent to a given IP and port. Any device receives every message which is directed to its IP address and UDP port number. If configured correspondingly, a DMX server (= Ethernet<-->DMX bus coupler) can be controlled with UDP commands from a number of clients (= controller devices) and can submit UDP messages simultaneously to several clients without need of reconfiguration. Reliability of data transfer with UDP is less than with TCP, however.

Following applies to both modes, this will be described as DMX server:

Control is possible with text oriented ASCII commands (rotary switch positions 1, 2, 4 - see part 1 of the manual) or mixed with binary commands (see part 2 and 3 of the manual) whose format is composed in a way that normally they can be merged conflict-free with the ASCII commands. **Because data transfer via Ethernet is independent of baudrates, MIDI-oriented commands are generally suitable for binary control**. This may be useful with control equipment with no or bad possibilities to transform numerical values into ASCII strings and vice versa.

The essentially **MIDI-oriented** rotary switch positions 3 and 5 offer similar possibilities of operation. ASCII commands, however, can only be used then as the body of System Exclusive MIDI messages.

The DMX equipment is connected to the box with DMX bus cables. During cabling work all equipment should be powered off. Additional information concerning the DMX lighting bus see appendix B.

In most cases the DMX Control Box is used to **control lamps and moving lights**. For this kind of application plug the DMX cable into the female socket "DMX OUT" in the middle of the rear panel. The DMX address switches of all dimmers etc. are to be set in a scheme which makes best sense for your specific project.

If you are not sure and have no 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 you intend to use the Control Box **to extract data from an external DMX bus (DMX receiver)**, plug the incoming DMX cable into the male socket "DMX IN" placed at the right side of the rear panel. If the signals received at "DMX IN" shall be **looped** to "DMX OUT" of the other DMX bus, both connections are made.

The Ethernet RJ45 socket is connected to an **Ethernet with 10 or 100 MBit/s**. To connect it to a hub or switch, use a 1:1 patch cable. For direct connection to a PC use a cable with crossed pairs of wire.

After these preparations are finished, **power is switched on**: first the DMX equipment then the Ethernet/DMX Control Box. Now a client (terminal software recommended for first use) establishes a connection, which has to use the previously configured IP address and port number of the Ethernet / DMX Control Box.

In default state the dual LED at the front panel will shine green, i.e. the box operates as DMX transmitter with internal clock. All DMX output channels should be set to zero level (or to a previously stored preset if the box was already used). If a "new line" is sent via Ethernet, the box should respond with a "new line". During a TCP connection the blue LED is permanently lit, during UDP data exchange it flashes.

ASCII protocol of the Ethernet / DMX512 Control Box

Especially for tests the DMX Control Box can completely be operated manually with a terminal program (Telnet capable). **In DMX server mode, all commands are processed inside the Ethernet / DMX512 Control Box.** The client at the other end of the network only sends and receives short phrases of ASCII text as described below.

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. For systematic reasons in this manual all command codes are written in capitals (except i and o to be better recognized). Manual input, of course, will be made with lowercase letters. Feedback - if any - is always given in uppercase letters.

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 messages via Ethernet, 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. (For a more detailed description of the internal structure of the see 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 or / 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 immission of electromagnetic radiation.

Short reference of all ASCII commands

Sn	address DMX channel (write SLOT register) for subsequent action (n=1 - 512)	p.13
Vn	set DMX level at DMX channel=SLOT (n=0 - 255)	p.13
,n	(comma) increment SLOT first then set level at new DMX channel (n=0 - 255)	p.13
=n	fill block of n DMX channels starting from SLOT+1 with level of SLOT (n=1-512)	p.12
+	increase transmit buffer level DMX channel=SLOT by one	p.14
-	decrease transmit buffer level DMX channel=SLOT by one	p.14
^n	add n to transmit buffer level DMX channel=SLOT (n=0 - 255)	p.14
_n	subtract n from transmit buffer level DMX channel=SLOT (n=0 - 255)	p.14
\$	from now DMX level in HEX (only V- , comma- , ^ , _- , R- , Z- , Q- command)	p.14
&	from now DMX level DECIMAL (only V , comma , ^ , _ , R , Z and Q-command)	p.15
Ts.t	set FADETIME s=seconds t=tenths	p.15
!	stop all fade processes and freeze them at actual DMX level	p.15
Jn	copy DMX receive buffer starting from channel=SLOT into the transmit buffer	p.15
Zn	read n bytes starting at DMX channel=SLOT from transmit buffer to Ethernet	p.15
Rn	read n bytes starting at DMX channel=SLOT from receive buffer to Ethernet	p.15
U	always transmit DMX OUT at channel=SLOT from the transmit buffer (n=0)	p.15
K	transmit less buffer level at channel=SLOT (n=1)	p.16
G	transmit greater buffer level at channel=SLOT (n=2)	p.16
o	always transmit DMX OUT at channel=SLOT from the receive buffer (n=3)	p.17
P	transmit last changed buffer level at channel=SLOT (n=4)	p.17
*n	set merge method (n=0,1,2,3,4 see above) globally for all DMX channels	p.17
Nt	configure automatic message at channel=SLOT with threshold t (t=0-127)	p.18
/t	configure automatic messages for all DMX channels equally with threshold t	p.18
Yn	switch automatic messages globally ON/OFF and select format (n=0 - 4)	p.18
Xn	switch messaging of DMX triggered strings ON/OFF (n=0,1)	p.19
:	<memory name> <string> " enter and store user programmable string	p.20
;	<memory name> check string, message it for test via Ethernet	p.21
Mn	set the masterfader: n=0 to 200 (in percent, see detailed description below)	p.21
in	set length of chaser cycle (n=2 to 127) and start the chaser	p.21
>t	set duration of chaser step in 1/10 s units	p.22
(n	enter start scene (preset no.) of chaser cycle . See detailed description	p.22
)	forward chaser immediately for one step	p.22
<t	release a lighting flash: all DMX channels are pulsed to 100% for t * 1/10s	p.22
Ln	set length of DMX Cycle (n=24 - 512)	p.23
Q	show content of all DMX registers at DMX channel=SLOT via Ethernet	p.23
[run DMX transmitter/merger with internal timing clock	p.23
\	run DMX transmitter/merger synchronised by DMX IN	p.23
]	loop DMX receiver from DMX IN to DMX OUT	p.24
~n	save transmit buffer and configuration as preset no. n	p.24
@n	load preset Nr. n into buffers and update configuration	p.24
 	reset all buffers and configuration to default (=delivery) state	p.26
?	show list of actual configuration parameters	p.26

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 a **set of compact binary commands available to set levels of any DMX channel or of blocks of DMX channels.** Detailed description see parts 2 and 3 of the manual.

Address 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 applied to subsequently given commands.

Example: S123 writes 123 into register SLOT

Transmit buffer manipulation:

V <level>

Write parameter 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 in accordance with previously entered values in the SLOT and FADETIME register. **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 .

, (comma) <level>

First this command increases the SLOT register automatically, then it writes the parameter 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.

+ (no parameter)

Increase (add 1 to) the level of the DMX channel addressed by SLOT

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, no parameter)

Decrease (subtract 1 from) the level of the DMX channel addressed by SLOT

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 transmit 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 transmit 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.

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

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 commands J and H used in previous firmware versions.

Example: **S1J512** copies the complete receive buffer into the transmit buffer. This is a useful feature to save an externally created lighting scene which is received at DMX IN as a preset of the Ethernet/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 + LF>

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 + LF>

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

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

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 3. So loading of one of these presets can change the merge method and automatic messages. Loading presets 4 to 384 does not affect the merge method.

Note that "merge" has different meaning in this context from the way it is used with the Art-Net protocol !

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

○ (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: S99o 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 via Ethernet 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 Ethernet. 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.

This way "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 Ethernet/DMX Control Box by the user. These are transmitted automatically after certain level changes at the DMX channels 500 to 512 via Ethernet. This way external equipment may be controlled by means of a DMX bus to a limited extent.

Additionally, this feature is useable in the "transparent" modes of operation, see part 4 of the manual.

N <threshold>

activates automatic messages from 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 27/28.

/ <threshold>

activates automatic 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 Ethernet interface may be 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** (see part 3)

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 3. 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>

Switch emission of preprogrammed strings globally ON and OFF

X0 Switch emission of strings globally **OFF** (default)

X1 Switch emission of preprogrammed strings (DMX channel 500-512) 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 3. When switched on again, the previous configuration is reactivated exactly and completely.

When specific data are received at DMX IN

send preprogrammed strings via Ethernet:

This allows messaging of user preprogrammed strings (text or binary) via Ethernet, if specific MDX 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 Ethernet 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 9. 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 for example. 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 beeing 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 Ethernet. "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 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 Ethernet

; **<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. 41)

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 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 219. 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. 41)

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.

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** (new from firmware revision no. 73). **Because this preset is automatically loaded when the Control Box 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 Control Box as preset no. 23

Special feature: save transmit buffer partially (new from Dec.2010 / revision number 41)

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.

Special feature: save flash pattern permanently (new from Dec 2010 / revision number 41)

~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 Control Box 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 (new from firmware revision no. 73): 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 Dec.2010 / revision number 41)

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.

{) (no parameter)

Initiates "download" of the complete nonvolatile memory (all presets and preprogrammed strings) via Ethernet using the XMODEM CRC protocol

Comment: new at devices delivered from May 2011 (revision number >= 46)

By use of this command you can 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.

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.

Though terminal programmms "Hyperterminal" and "Teraterm" provide data transfer based on the XMODEM CRC protocol, but this does not work with a TCP connection. Probably because during XMODEM transfer necessarily some bytes are transmitted which are interpreted as Telnet Control codes.

As far as known here, the only reliably working transfer procedure is achieved with the UDP terminal software "UDPTERM" which may be downloaded from the Cinetix website. The "UDPTERM" command code CTRL-G calls the sequence {) in the background and handles the handshake, which is necesseary for XMODEM, automatically. The handshake cannot be managed by manual terminal operation.

{ ((no parameter)

Initiates "upload" of a previously stored backup to the nonvolatile memory (all presets and preprogrammed strings) via Ethernet using the "XMODEM CRC" protocol

Comment new at devices delivered from May 2011 (revision number >= 46)

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

As far as known here, the only reliably working transfer procedure is achieved with the UDP terminal software "UDPTERM" which may be downloaded from the Cinetix website. The "UDPTERM" command

code CTRL-U calls the sequence { (in the background and handles the handshake, which is necessary for XMODEM, automatically. The handshake cannot be managed by manual terminal operation.

Further comment see comment of the complementary download command {).

| (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" gesetzt.
Number base = "decimal". Masterfader = 100% The Chaser is switched OFF.
LOOP = 512, FADETIME = 0.0. Presets are not deleted or changed otherwise.

? (no parameter)

read setting of configuration parameters

Comment: This command corresponds with the '?' command used in configuration mode. The response is returned via Ethernet to the client.

"Ethernet / DMX512 Control Box" User Manual part 2

Binary protocol of the Ethernet / 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.

If the Ethernet/DMX Control Box shall be used in combination with the "DMX-Control" software, we recommend strongly to use the Art-Net Mode (rotary switch pos. 8).

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.

In part 3 of the manual another set of commands is described, which is formally compatible with **MIDI-channel messages**. Due to its construction it distinguishes more robustly from ASCII commands and resynchronizes better in case of data transfer errors. Furthermore, it provides complete control of the Ethernet / DMX512 Control Box. **With special PC-installed drivers** - for example with "ipMIDI" (see www.nerds.de)- **the Ethernet / DMX512 Control Box may be driven directly from MIDI-sequencers or similar equipment.**

Set of binary 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 transmit 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 Ethernet.** If single bytes should have gone lost, every binary transfer is finished automatically after 0.5 seconds to prevent the Ethernet/DMX 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.

Subsequently, where numbers are put between acute brackets, this means 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

<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 transmit 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 Ethernet/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 transmit 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 transmit 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 <4> !!**

<15>(hex F) Send and set the complete DMX universe (channels 1 to 512 to DMX OUT)

no further header bytes

Data bytes: sequence of 512 arbitrary bytes.

It is absolutely necessary that exactly 512 data bytes is transferred. 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. In this case, all bytes transferred before the hanging are written into the transmit buffer

The complete transmit buffer is overwritten. This has no influence on the actual length of the DMX transmit cycle (command L) and on the actual merge setting. All active fade processes are cancelled immediately.

<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)

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 has to be equal 0.

Example: to get (dec) 40 levels starting from DMX channel (dec) 370, following command sequence has to be sent: hex 6,1,72,28 - i.e. decimal 6,1,114,40

Example: to get (dec) 256 levels starting from DMX channel 1, following command sequence has to be sent: 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.

<16>(hex 10) **poll a complete DMX universe (channels 1 to 512 from DMX IN)**

no further header bytes

Immediately the complete DMX receive buffer (= a block of 512 bytes) is returned-regardless of the actually received length of the DMX cycle.

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.

"Ethernet / DMX512 Control Box" User Manual part 3

MIDI protocol of the Ethernet / DMX512 Control Box

To run the Ethernet / DMX Control Box under different conditions, a binary set of commands is implemented, which is based on MIDI channel messages.

While describing the operative handling with these MIDI commands we assumed that it is possible at client side to send and receive binary MIDI data via Ethernet. The realisation of this task is not easy in a common MIDI environment. The special PC driver "ipMIDI" (see www.nerds.de) together with the new implemented multicast mode of operation of the Ethernet/DMX Control Box provides a reasonable solution for this task.

Because data transfer via Ethernet is independent of conventional baud rates, **this set of commands is useful too for control equipment and software which is not designed to be used in MIDI context**, but has problems with conversion of numeric values into ASCII text and vice versa.

General structure of MIDI messages: The **status byte** (where the type of command and MIDI channel is coded) is followed - depending on the type of the MIDI message - by up to 2 **data bytes**. Contained in NOTE ON, NOTE OFF or POLY KEY PRESSURE messages, the **1st data byte** is the note value ("pitch", see table at the last page of the manual), the **2nd data byte** is the strength ("velocity") of the key stroke. With CONTROL CHANGE messages the 1st data byte addresses the controller number, the 2nd data byte is the value to be applied to this controller. In the given DMX context, you have to ignore the musical meaning of these terms.

The content of MIDI messages used here cannot follow the conventions which have been established in the world of musicians, but has been optimized for tasks of lighting control. We have taken care to design the codes in a way they can be reproduced with common MIDI equipment like sequencers and programmable keyboards.

First is described how the Ethernet / DMX Control Box can be operated with simple commands for frequently used applications.

Subsequently the complete set of MIDI commands is described in detail.

Quick start:

One of the most frequently used applications is **lighting control based on a MIDI sequencer**, which in most cases is available as software running on a PC. Depending on the specific model and personal liking, **NOTE ON or CONTROL CHANGE messages are used to program the lighting events**. Apart from the different type of command (MIDI status byte) the commands are constructed identically for both methods. **Both types of command can be mixed arbitrarily.**

To address any of the DMX channels (in DMX slang "slots") 1 to 127, control data have to be produced on the track of that MIDI channel which has been selected during configuration. Advice to access DMX channels 128 to 512 see below.

The 1st data byte of the MIDI command defines the DMX channel to be addressed.
The 2nd data byte of the command describes the DMX level (light intensity) to be set.
the 2nd MIDI data byte is multiplied by 2 inside the Ethernet/DMX Control Box.

Following exceptions have to be regarded

If a **CONTROL CHANGE** command was sent, the resulting 8 bit DMX level is additionally **incremented**,
i.e. **the next higher odd level is set.**

This behaviour can be deactivated (use a PROGRAM CHANGE command with data byte=60)
then for NOTE ON as well as for CONTROL CHANGE commands is valid:
if the 2nd MIDI data byte is equal to 127, the DMX level is set to its max. value 255

Or described in the opposite way of thinking: **to set a certain DMX level (0 to 255) with a simple MIDI command, HALF OF the intended DMX level has to be entered in the 2nd MIDI data byte.** Odd DMX levels are set with CONTROL CHANGE, even DMX levels are set with NOTE ON.

Example: MIDI channel configured as "1". Then DMX channel no 1 is **set to level 64** (corresponds to 25%) with the MIDI message: NOTE ON (status byte = hex 90). 1st data byte = 1, 2nd data byte = dez.32 (hex 20). Correspondingly, the DMX **level is set to 65** with following MIDI message: CONTROL CHANGE (status byte = hex B0). 1st data byte = 1, 2nd data byte = dez.32 (hex 20).

Some further annotations:

Many MIDI instruments - like keyboards and sequencers - automatically terminate a NOTE ON with a corresponding NOTE OFF message or by a NOTE ON message with velocity = 0. In the standard configuration of the DMX Control Box this turns the DMX channel and the attached lamp off immediately. This mode of operation is useful, if - e.g. - a keyboard shall be used as a "light organ" or similar actions are programmed on a sequencer.

In other applications, where NOTE ON messages simply shall be used to control steady lighting scenes, this behaviour is unpleasant.

The response of the Ethernet/DMX Control Box to NOTE ON messages with velocity=0 or to any NOTE OFF messages can be prevented with PROGRAM CHANGE command 121, details see below.

The advantage of setting DMX levels with NOTE ON messages is the possibility to "play in" the timing interactively and intuitively with a keyboard.

When - alternatively - CONTROL CHANGE messages are used to set DMX levels, some sequencer software offers the feature to create fade ramps graphically. In this case the very limited data capacity of a MIDI line has to be taken into account. Another approach is the use of controller pads, which offer direct visual feedback of intensity and colour settings, but provide less feeling for intuitive handling of timing. You should decide to use one method at your taste.

Different kinds of sequencer software use different nomenclatures to name MIDI data bytes. "Cubase" by Steinberg calls the first data byte "Wert1" and the second one "Wert2" (german versions). However, "Logic Audio" names the first data byte of a CONTROL CHANGE message "Num" and the second one "Val".

Get informed about the naming conventions of your sequencer. Unfortunately many sequencer programs don't offer the feature of entering the first data byte as a plain number, but only as a note name as used by musicians.

A transformation table is published in appendix C.

To write data into DMX channels 128 to 512 with these simple commands, the Ethernet / DMX Control Box is sensitive to the next three MIDI channels too: (not valid if the basic MIDI channel is configured as 14,15 or 16):

MIDI-Kanal in status byte	1st data byte	addresses DMX channel	calculation of. 1st data byte
= configured MIDI channel	1 to 127	1 to 127	= DMX channel
following options are only available if MIDI channel is configured as 1 to 13 :			
as configured + 1	0 to 127	128 to 255	= DMX channel minus 128
as configured + 2	0 to 127	256 to 382	= DMX channel minus 256
as configured + 3	0 to 127	384 to 511	= DMX channel minus 384
as configured	0 special case !	512	= 0

Thus, if MIDI channel no.1 was selected during configuration, the Ethernet/DMX Control Box is sensitive for the MIDI channels 2,3 and 4, too. This means: on a sequencer program you always have to reserve a block of 4 MIDI channels to fully control the Ethernet/DMX Control Box and the corresponding edit tracks have to be initialized.

Attention: data which are meant for other MIDI equipment which works on these channels may be misinterpreted.

If one of the MIDI channels 14,15 or 16 was selected during configuration, the Ethernet/DMX Control Box is exclusively sensitive to the selected channel. This is an advantage in applications which use almost the complete set of MIDI channels for musical tracks or similar. With simple commands (NOTE ON or CONTROL CHANGE), however, only DMX channels 1 to 127 are addressable.

This can be worked around with somewhat more complex commands, which **allow to adjust any DMX channel 1 to 512 with full 8 bit accuracy.** See details at the description of the POLY KEY PRESSURE and the PITCH WHEEL CHANGE command below.

Survey of all MIDI Channel Commands (MIDI Implementation Chart)

The basic MIDI channel is entered in configuration mode. The MIDI channels given in the table have to be sent in MIDI commands **relatively to this preselected basic MIDI channel**. If the basic MIDI channel is configured as 14,15 or 16, the commands will work only at the preselected channel, but not on any additional channels.

Abbreviations: DB means "data byte", DB1 means "1st data byte", DB2 means "2nd data byte",
When using **PROGRAM CHANGE** commands you should take into account, that most MIDI devices and software send the data byte value "0" when "program No.1" is selected! **In the table "DB" denotes the physically transferred data byte!**

With PROGRAM CHANGE 63 the functions of **CONTROL CHANGE** and **POLY KEY PRESSURE** commands (as they are described in this manual) **may be exchanged**.
This option provides better flexibility to work with different kind of MIDI control equipment.

MIDI message and coded MIDI channel	special data values	function /effect	p.
NOTE OFF	all,see detail description	DMX level --> 0 May be blocked by ProgramCh. 79	47
NOTE ON MIDI channel = Config. + next 3 channels	DB1 = DMX channel DB2 = DMX level ./ 2	set DMX channel and level (only 1 MIDI message / 7bit resolution) DMX level = DB2 * 2	36
MIDI channel = Config.	s. detailed description	load preset 0 - 299	46
POLY KEY PRESSURE MIDI channel = Config.	DB1 = 1 DB2 = 1/10 sec. 0-127	set FADETIME to 0-12.7 seconds fade time = DB2 ./10 seconds	38
notice PROGRAM CHANGE 63 !	DB1 = 2 DB2 = 1/10 sec. 0-127	set FADETIME to 10 - 22.7 seconds fade time = 10 + DB2 ./10 seconds	38
	DB1 = 3 DB2 = 1/10 sec. 0-119	set FADETIME to 20 - 31.9 seconds fade time = 20 + DB2 ./10 seconds n	38
	DB1 = 4 DB2 = 1/4 sec. 0-127	set FADETIME in the range 0 to 31.8 seconds with a single controller. Fade time = DB2 ./4 seconds	38
	DB1 = 7 DB2 = masterfader %	set masterfader in the range 0 – 127 % Note: DB1 was changed with respect to previous versions	39
	DB1 = 8 DB2 =masterfader-100	set masterfader in the range 100-200 % Note: DB1 was changed with respect to previous versions	39
	DB1 = 15 (hex F) DB2 = duration *1/10s	release flash pulse: all DMX channels are set to 100% during a short time. See detailed description	39
	DB1 = 16 (hex 10) DB2 = step in 1/10 s	set step duration of the chaser. Is controlled by an internal timer. 0 = chaser OFF.	40
	DB1 = 18 (hex 12) DB2 = count of scenes per cycle	set cycle length of the chaser and start it: display any preset (scene) 'step duration' long, then next preset is loaded. Repeat cycle after 'count' steps	39
	DB1 = 19 (hex 13) DB2 = start-128 (0-127)	set start scene (preset no.) of the chaser cycle. 128 is added internally. See detailed description	40
	DB1 = 20 (hex 14) DB2 = start-256 (0-127)	set start scene (preset no.) of the chaser cycle. 256 is added internally. See detailed description	40
	DB1 = 36,37 (hex24,25) DB2 = 0 to 127	poll (entry of DB2) levels of DMX OUT starting from DMX channel=SLOT (DB2=0: polls 128 channels)	42
	DB1 =40-43 (hex28-2B) DB2 = 0 to 127	poll (entry of DB2) levels of DMX IN starting from DMX channel=SLOT. Selectable data formats	43
	DB1 = Mode 48-51 (hex 30 - 33) DB2 = treshold	(de)activate automatic MIDI messages generated from input at DMX-IN for DMX channel addressed by "SLOT"	41
	DB1 = 58 - 61(hex 3A-D) DB2 = threshold	(de)activate automatic MIDI messages generated from input at DMX-IN for all DMX channels equally	41
	DB1 = 72 (hex 48)	fill a block of DB2 DMX channels starting from ch. SLOT+1 with the final level of DMX ch. "SLOT"	38

MIDI message and coded MIDI channel	special data values	function /effect	p.
POLY KEY PRESSURE MIDI channel = Config.	DB1 = 80 – 83 (hex 50–53)	address DMX channel using a single MIDI channel.	36
notice PROGRAM CHANGE 63 !	DB1 = 84 (hex 54)	set DMX level (@SLOT) to DB2 *2 adjust DMX level with 8 bit resolution to even value	37
	DB1 = 85 (hex 55)	set DMX level (@SLOT)to DB2 *2 + 1 adjust DMX level with 8 bit resolution to odd value	37
	DB1 = 86 (hex 56)	set DMX level (@SLOT) to DB2 *2 first increment DMX channel (add 1), then adjust DMX level with 8 bit resolution to even value	37
	DB1 = 87 (hex 57)	set DMX level (@SLOT)to DB2 *2 + 1 first increment DMX channel (add 1), then adjust DMX level with 8 bit resolution to odd value	37
	DB1 = 96 (hex 60) DB2 = preset no.	load preset no. 0 – 127. The system configuration is exclusively loaded with preset no. 0 - 3	45
	DB1 = 97 (hex 61) DB2 = preset no. -128	load preset no. 128 – 255	45
	DB1 = 98 (hex 62) DB2 = preset no. -256	load preset no. 256 – 383	45
	DB1 = 99 (hex 63) DB2 = preset no.	load preset no. 0–127 partially. Preset DMX channels 129-256 are loaded into DMX OUT channels 1-128	45
	DB1 = 100 (hex 64) DB2 = preset no.	load preset no. 0–127 partially. Preset DMX chann. 257-384 are loaded into DMX OUT channels 1-128	45
	DB1 = 101 (hex 65) DB2 = preset no.	load preset no. 0–127 partially. Preset DMX chann. 385-512 are loaded into DMX OUT channels 1-128	45
	DB1 = 102 (hex 66) DB2 = preset no. - 128	load preset no. 128–255 partially. Preset DMX ch. 129-256 are loaded into DMX OUT channels 1-128	45
	DB1 = 103 (hex 67) DB2 = preset no. - 128	load preset no. 128–255 partially. Preset DMX ch. 257-384 are loaded into DMX OUT channels 1-128	45
	DB1 = 104 (hex 68) DB2 = preset no. - 128	load preset no. 128–255 partially. Preset DMX ch. 385-512 are loaded into DMX OUT channels 1-128	45
	DB1 = 105 (hex 69) DB2 = preset no. - 256	load preset no. 256–383 partially. Preset DMX ch. 129-256 are loaded into DMX OUT channels 1-128	46
	DB1 = 106 (hex 6A) DB2 = preset no. - 256	load preset no. 256–383 partially. Preset DMX ch. 257-384 are loaded into DMX OUT channels 1-128	46
	DB1 = 107 (hex 6B) DB2 = preset no. - 256	load preset no. 256–383 partially. Preset DMX ch. 385-512 are loaded into DMX OUT channels 1-129	46
	DB1 = 108 (hex 6C) DB2 = preset no.	load preset no. 0–127 partially. Only DMX channels 1-128 are loaded, 129-512 remain unchanged	46
	DB1 = 109 (hex 6D) DB2 = preset no.	load preset no. 0–127 partially. Only DMX channels 129-256 are loaded, others remain unchanged	46
	DB1 = 110 (hex 6E) DB2 = preset no.	load preset no. 0–127 partially. Only DMX channels 257-384 are loaded, others remain unchanged	46
	DB1 = 111 (hex 6F) DB2 = preset no.	load preset no. 0–127 partially. Only DMX channels 385-512 are loaded, others remain unchanged	46
	DB1 = 112 (hex 70) DB2 = preset no.	save preset no. 0 - 127. The system configuration is exclusively saved with preset no. 0 - 3	44
	DB1 = 113 (hex 71) DB2 = preset no. - 128	save preset no. 128 - 255	44
	DB1 = 114 (hex 72) DB2 = preset no. - 256	save preset no. 256 - 383	44
	DB1 = 115 (hex 73) DB2 = preset no.	save preset no 0–127 partially. DMX OUT channels 1-128 are copied into preset channels 129-256	45
	DB1 = 116 (hex 74) DB2 = preset no.	save preset no 0–127 partially. DMX OUT channels 1-128 are copied into preset channels 257-384	45

MIDI message and coded MIDI channel	special data values	function /effect	p.
POLY KEY PRESSURE MIDI channel = Config.	DB1 = 117 (hex 75) DB2 = preset no.	save preset no 0–127 partially. DMX OUT channels 1-128 are copied into preset channels 385-512	45
notice PROGRAM CHANGE 63 !	DB1 = 118 (hex 76) DB2 = preset no. - 128	save preset no 128-255 partially. DMX OUT chann. 1-128 are copied into preset channels 129-256	45
	DB1 = 119 (hex 77) DB2 = preset no. - 128	save preset no 128-255 partially. DMX OUT chann. 1-128 are copied into preset channels 257-384	45
	DB1 = 120 (hex 78) DB2 = preset no. - 128	save preset no 128-255 partially. DMX OUT chann. 1-128 are copied into preset channels 385-512	45
	DB1 = 121 (hex 79) DB2 = preset no.- 256	save preset no 256-383 partially. DMX OUT chann. 1-128 are copied into preset channels 129-256	45
	DB1 = 122 (hex 7A) DB2 = preset no. - 256	save preset no 256-383 partially. DMX OUT chann. 1-128 are copied into preset channels 257-384	45
	DB1 = 123 (hex 7B) DB2 = preset no. - 256	save preset no 256-383 partially. DMX OUT chann. 1-128 are copied into preset channels 385-512	45
	DB1 = 126 (hex 7E) DB2 = only15(hex F)	Store actual lighting scene as a pattern which is loaded during the flash effect	45
	DB1 = 127 (hex 7F)	DB2 has the same meaning as the data byte of the corresponding PROGRAM CHANGE command.	47
CONTROL CHANGE MIDI channel = Config. + next 3 channels	DB1 = DMX channel DB2 = DMX level ./ 2	set DMX channel and level (only 1MIDI message / 7bit resolution) DMX level = DB2 * 2 +1	36
PROGRAM CHANGE MIDI channel = Config.	DB = 0	always send DMX level defined by MIDI command to DMX-OUT	37
	DB = 1	Merge: always send less of DMX IN and MIDI cmd	41
	DB = 2	Merge: always send greater of DMX IN and MIDI	41
	DB = 3	always send level from DMX IN to DMX OUT	41
	DB = 4	Merge: "last takes Precedence"	41
	DB = 8	decrease DMX level (minus 1) at channel "SLOT"	38
	DB = 9	increase DMX level (plus 1) at channel "SLOT"	38
	DB = 16 (hex 10)	forward chaser immediately by 1 step	38
	DB = 32 (hex 20)	copy receive buffer (DMX IN) to transmit buffer	38
	DB = 36 (hex 24)	block execution of MIDI commands	44
	DB = 37 (hex 25)	stop command blocking and execute last received	44
	DB = 40 (hex 28)	deactivate automatic messages globally	42
	DB = 41 (hex 29)	activate automatic messages globally format as MIDI channel messages (default)	42
	DB = 42 (hex 2A)	activate automatic messages globally format as SysEx/ASCII	42
	DB = 50 (hex 32)	deactivate DMX triggered msgs globally (default)	42
	DB = 51 (hex 33)	activate DMX triggered msgs channel 500-512	42
	DB = 56 (hex 38)	No Running State allowed in commands	43
	DB = 57 (hex 39)	accept Running State in commands (default *)	43
	DB = 60 (hex 3C)	CONTROL CHANGE behaves like NOTE ON Controller value 127 is transformed to DMX 255	43
	DB = 61 (hex 3D)	CONTROL CHANGE sets DMX level +1 (default)	43
	DB = 62 (hex 3E)	don 't exchange CONTROL CHANGE and POLY KEY PRESSURE (default)	44
	DB = 63 (hex 3F)	exchange CONTROL CHANGE and POLY KEY	44
	DB = 64 (hex 40)	set LOOP=SLOT	43
	DB = 72 (hex 48)	Stop all fade processes, freeze at momentary level	38
	DB = 80-84 (hex50-54)	global merge configuration,see detailed description	41
	DB = 96 (hex 60)	NOTE ON 0-127 (keyboard) loads preset no 0-127	46
	DB = 97 (hex 61)	NOTE ON 0-127 loads preset no 128-255	46
	DB = 98 (hex 62)	NOTE ON 0-127 loads preset no 256-383	46

MIDI message and coded MIDI channel	special data values	function /effect	p.
PROGRAM CHANGE MIDI channel = Config.	DB = 99 (hex 63)	NOTE ON 0-127 loads preset no 0-127 partially See POLY KEY PRESSURE DB1=99	46
	DB = 100 (hex 64)	NOTE ON 0-127 loads preset no 0-127 teilweise Siehe POLY KEY PRESSURE DB1=100	46
	DB = 101 (hex 65)	NOTE ON 0-127 loads preset no 0-127 partially See POLY KEY PRESSURE DB1=101	46
	DB = 102 (hex 66)	NOTE ON 0-127 loads preset no 128-255 partially See POLY KEY PRESSURE DB1=102	46
	DB = 103 (hex 67)	NOTE ON 0-127 loads preset no 128-255 partially See POLY KEY PRESSURE DB1=103	46
	DB = 104 (hex 68)	NOTE ON 0-127 loads preset no 128-255 partially See POLY KEY PRESSURE DB1=104	47
	DB = 105 (hex 69)	NOTE ON 0-127 loads preset no 256-383 partially See POLY KEY PRESSURE DB1=105	47
	DB = 106 (hex 6A)	NOTE ON 0-127 loads preset no 256-383 partially See POLY KEY PRESSURE DB1=106	47
	DB = 107 (hex 6B)	NOTE ON 0-127 loads preset no 256-383 partially See POLY KEY PRESSURE DB1=107	47
	DB = 108 (hex 6C)	NOTE ON 0-127 loads preset no 0-127 partially See POLY KEY PRESSURE DB1=108	47
	DB = 109 (hex 6D)	NOTE ON 0-127 loads preset no 0-127 partially See POLY KEY PRESSURE DB1=109	47
	DB = 110 (hex 6E)	NOTE ON 0-127 lädt Preset Nr. 0-127 partially See POLY KEY PRESSURE DB1=110	47
	DB = 111 (hex 6F)	NOTE ON 0-127 loads preset no 0-127 partially See POLY KEY PRESSURE DB1=111	47
	DB = 120 (hex78)	NOTE ON sets the level of a DMX channel Velocity=0 and NOTE OFF are accepted (default)	47
	DB = 121 (hex79)	NOTE ON sets the level of a DMX channel Velocity=0 is ignored	47
	DB = 122 (hex7A)	ask actual channel configuration (get SysEx msg)	47
	DB = 124 (hex7C)	DMX OUT and Merge w. internally generated clock	47
	DB = 125 (hex7D)	DMX OUT and Merge synchronized by DMX-IN	47
	DB = 126 (hex7E)	loop DMX IN to DMX OUT (receiver function only)	47
	DB = 127 (hex7F)	Clear All Memory	47
CHANNEL PRESSURE MIDI channel = Config.	DB= DMX level ./2	increase adressed DMX channel and set there DMX level = DB*2 (seeASCII comma command)	37
MIDI channel = Config.+1	DB= DMX level ./ 2	increase adressed DMX channel and set there DMX level = DB*2 +1	37
MIDI channel = Config.+2	DB= DMX level ./ 2	increase adressed DMX channel and set there DMX level = DB*2. Running state temporarily ON	37
MIDI channel = Config.+3	DB= DMX level ./ 2	increase adressed DMX channel and set there DMX level = DB*2+1 einstellen. Running state ON	37
PITCH CHANGE MIDI channel = Config.	DB1 = less 64 or not DB2 = DMX level ./ 2	set DMX level at position SLOT (8bit resolution) add one if DB1 is greater or equal 64 (hex 40)	36
SYSTEM EXCLUSIVE	channel independent	encloses all ASCII commands for MIDI	48

Commands where is annotated " MIDI channel as configured and following 3 channels " are able to address all 512 DMX slots (but only if MIDI channel no. 1 to 13 has been configured).

The relationship between the MIDI channel coded in the command and the corresponding entry written into the SLOT register is explained in the following table.

Express setting of DMX channel (= "SLOT") with a single MIDI message:

NOTE ON or CONTROL CHANGE

The 1st data byte of the MIDI command defines the DMX channel to be addressed and manipulated.

The 2nd data byte of the MIDI command describes the light intensity to be set.

The 2nd MIDI data byte is multiplied by 2 inside the ETHERNET/DMX Control Box.

Following **exceptions** do apply:

When setting is done with a **CONTROL CHANGE** command, the 8-bit DMX level is additionally incremented, i.e. the next higher odd DMX level is set.

This behaviour can be **deactivated** with a PROGRAM CHANGE command, data byte=60)

only then is valid for NOTE ON as well as for CONTROL CHANGE commands:

if the 2nd MIDI data byte is = 127, then the DMX level is set to its maximum value 255.

This simple method works only for DMX channels 1 to 127. To address DMX channels 128 to 512, the ETHERNET/DMX Control Box expects control commands on a higher MIDI channel corresponding with following table (only valid for configured MIDI channel 1 - 12):

coded MIDI channel	1 st data byte	addresses DMX channel	calculation of 1 st data byte
as configured	1 to 127	1 to 127	= DMX channel
next options are only available for configured MIDI channel 1-12:			
as configured + 1	0 to 127	128 to 255	= DMX channel minus 128
as configured + 2	0 to 127	256 to 382	= DMX channel minus 256
as configured + 3	0 to 127	384 to 511	= DMX channel minus 384
as configured	0 special case !	512	= 0

Special reaction on NOTE ON messages with velocity=0 and on NOTE OFF messages see PROGRAM CHANGE commands 120 and 121.

Address the active DMX channel (= "SLOT" register) with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

This alternative command version is used, **when all MIDI messages shall be sent on the single MIDI channel which has been entered in configuration mode.**

The coarse range is selected with the first data byte, which has to be chosen according to this table:

1st data byte	addresses DMX chan.= SLOT	2nd data byte	calculation of 1st data byte
80 (hex50)	1 to 127	1 to 127	= DMX channel
81 (hex51)	128 to 255	0 to 127	= DMX channel minus 128
82 (hex52)	256 to 382	0 to 127	= DMX channel minus 256
83 (hex53)	384 to 511	0 to 127	= DMX channel minus 384
80 (hex50)	512	0 (special case!)	= 0

Comment: This command does not directly trigger any action. But the updated content of the SLOT register will be executed together with subsequent commands. In DMX slang a DMX channel is called a "SLOT" (physically it denotes a time slot in the DMX transmit cycle, therefore this strange name)

Set DMX level with 8 bit resolution at DMX channel = SLOT with:

Method 1:

PITCH WHEEL CHANGE

MIDI channel as configured

1st data byte: if greater or equal 64, "1" is added to the DMX level
else nothing is added to the DMX level

2nd data byte: desired DMX level divided by 2
inside the Ethernet/DMX Control Box, this data is multiplied by 2 before it is written into the DMX transmit buffer

Comment: This coding scheme looks strange at first glance. But it is compatible with the standard MIDI method to put the 7 "most significant bits" into the second data byte. So it can be used together with simple MIDI equipment, which has only 7 bit capability of PITCH WHEEL CHANGE control.

Method 2:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCH)

MIDI channel as configured

1st data byte = 84 (hex54) adjusts to an **even** DMX level

2nd data byte: desired DMX level divided by 2
i.e vice versa: DMX level= 2nd data byte * 2

1st data byte = 85 (hex55) adjusts to **next odd** DMX level

2nd data byte: desired DMX level divided by 2
i.e vice versa: DMX level= 2nd data byte * 2 **plus 1**

1st data byte = 86 (hex56) **first increases the addressed DMX channel**
and adjusts this one to an **even** DMX level

2nd data byte: desired DMX level divided by 2
i.e vice versa: DMX level= 2nd data byte * 2

1st data byte = 87 (hex57) **first increases the addressed DMX channel**
and adjusts this one to **next odd** DMX level

2nd data byte: desired DMX level divided by 2
i.e vice versa: DMX level= 2nd data byte * 2 **plus 1**

Increase addressed DMX channel ("SLOT") and set DMX level with:

CHANNEL PRESSURE (equivalent to the ASCII "comma" command)

MIDI channel as configured: DMX level = data byte*2

MIDI channel as configured +1: DMX level = data byte*2 +1

MIDI channel as configured +2: DMX level = data byte*2, activates Running State

MIDI channel as configured +3: DMX level = data byte*2 +1 activates Running State

Comment: with this method a block of consecutive DMX channels can be set with different levels in a quite compact way. To maintain the start channel, the first DMX channel is set with NOTE ON or CONTROL CHANGE, all following ones with a CHANNEL PRESSURE command.

If the command is sent on the MIDI channel which is configured or the next higher one, running state in the command sequence is accepted in correspondence with the general system setup. However, if the 2nd or 3rd higher MIDI channel is coded into the status byte, running state is temporarily accepted until reception of the next status byte which differs from this rule.

Example: Assume code swith in position 2 (status byte Low Nibble =1). The DMX channels 1to 5 shall be set to DMX levels 10,20,30,40,50 and the channels 6,7,8 shall be set to levels 55,65,75. Follwing sequence of decimal bytes has to be transmitted: 145,1,5,211,10,15,20,25,212,27,32,210,37. With the last status byte 210 the acceptance of running state is reset to the previous system setup.

Fill block of DMX channels from "SLOT+1" with the final level of DMX channel "SLOT" with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 72 (hex48)

2nd data byte: block length (1 to 127)

Comment: Every DMX channel in the commanded range is faded or switched from its present level to the final state of the DMX channel which is preselected by the SLOT register. The fade duration is given by the actual setting of the FADETIME register.

Simple modifications of the DMX level with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 8 decrement (subtract 1 from) the level of DMX channel "SLOT"

data byte = 9 increment (add 1 to) the level of DMX channel "SLOT"

Comment: With these commands the disadvantage of lower accuracy of 7 bit MIDI data can be compensated. Furthermore it is useful to perform extremely slow fade transitions.

data byte = 16 (hex10) forward chaser immediately by 1 step

data byte = 32 (hex 20) the present content of the **receive buffer is completely copied into the transmit buffer** (and appropriate merge configuration provided: transmitted via DMX-OUT immediately)

Also see the corresponding ASCII/SysEx command 'J'

Comment: use this command to poll and save a complete lighting scene from an external console.

data byte = 72 (hex48) **Stop all fade processes immediately.**

Freeze all DMX levels at their present state.

Set FADETIME with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

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 respective 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.

In earlier **firmware versions (< 20)** the FADETIME was set differently: POLY KEY PRESSURE, 1st data byte 0 to 31:fade time in seconds, 2nd data byte 0 to 9: additional 1/10 seconds

Set the MASTERFADER with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 7

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

1st data byte = 8

then 2nd data byte = 0-100 (hex 64) masterfader setting 100-200%
(internally 100 is added to the data byte)

Comment: from December 2010 (revision number 41) the 1st data byte was changed to obtain better compatibility with MIDI Volume Control. At earlier firmware versions the 1st data byte was 8 resp.9

Also see comment at the equivalent ASCII/SysEx command 'M'

Release "flash" effect with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE) (new from Dec 2010 / rev.no 41)

MIDI channel as configured

1st data byte = 15 (hex F)

2nd data byte = flash duration in 1/10 s units (0,1 to 12,7 seconds)

all DMX channels are pulsed to 100%

= 0 terminate flash immediately, independently from a previously entered duration

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 the 2nd data byte. 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 segment with a POLY KEY PRESSURE command 1st data byte 126 (hex 7E), 2nd data byte =15 (hex F). This lighting scene should be designed in a way to avoid these side effects - or even a very individual flash pattern may be created.

Set chaser cycle length and start it with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE) (new from Dec 2010 / rev.no 41)

MIDI channel as configured

1st data byte = 18 (hex12)

2nd data byte = 2 - 127 (hex 7F) set chaser cycle length 2-127

or = 0: switches the chaser **OFF**

Comment: before the chaser can be started, the **step duration** (POLY KEY PRESSURE, 1st data byte= 16) as well as the **start scene** (POLY KEY PRESSURE 1st data byte= 19 or 20) 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 maintain 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 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.

Set chaser step duration with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 16 (hex10)

2nd data byte = (1 - 127) chaser step duration in 1/10 second units

or = 0 switches the chaser **OFF**

the duration of chaser steps is controlled by an internal timer at delivery, a default step duration of 2 seconds is active to provide an easy start

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.** As soon as <cycle length> presets are loaded in sequence, the procedure is repeated from <chaser start>.

Set chaser start preset (lighting scene) with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 19 (hex13) ,

2nd data byte: **offset = start preset minus 128** (0 - 127)

alternatively the chaser cycle may start with a preset in the range 256-383:

1st data byte = 20 (hex14)

2nd data byte: **offset = start preset minus 256** (0 - 127)

Comment: due to limited coding possibilities with MIDI channel messages, the number of the start preset of the chaser cycle cannot be entered directly. Instead, the **difference to preset no.128 is entered as the 'offset'.**

If start preset no.128 (offset=0) is chosen (default setting at delivery) the chaser cycle always starts with loading preset no. 128. **With an offset unequal to 0 the chaser cycle starts with preset no. (128 + offset).** If the chaser would access a preset beyond 383, the sequence continues with loading preset no. 128 etc..

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.

Set the merge method (transmission from DMX OUT) for DMX channel = "SLOT" with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 0 always transmit the MIDI activated level from the transmit buffer

data byte = 1 transmit the less of transmit buffer and DMX IN

data byte = 2 transmit the greater of transmit buffer and DMX IN

data byte = 3 always forward the level from DMX IN to the transmit buffer

data byte = 4 transmit the last changed one of MIDI activated transmit buffer or DMX IN ("last takes precedence")

Set the merge method (transmission from DMX OUT) equally for DMX channels with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 80 (hex 50) transmit the MIDI activated level from the transmit buffer

data byte = 81 (hex 51) transmit the less of transmit buffer and DMX IN

data byte = 82 (hex 52) transmit the greater of transmit buffer and DMX IN

data byte = 83 (hex 53) always forward the level from DMX IN to the transmit buffer

data byte = 84 (hex 54) transmit the last changed one of MIDI activated transmit buffer or DMX IN receive buffer ("last takes precedence")

Activate "automatic" messages for DMX channel = "SLOT" with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCH)

MIDI channel as configured

1st data byte: select MIDI format of "automatic" messages

Following **options are available:**

1st data byte = **48** (hex 30) send a mix of NOTE ON and CONTROL CHANGE messages (default)

This format is identical to the one described above under "quick start".

EVEN DMX levels are sent in 7 bit format as NOTE ON messages

ODD DMX levels are sent in 7 bit format as CONTROL CHANGE messages

1st data byte = **49** (hex 31) send a CONTROL CHANGE message

The DMX level is reported in 7 bit format.

Optimum for quick and easy post-editable recording.

1st data byte = **50** (hex 32) send a NOTE ON message

The DMX level is reported in 7 bit format. **When suppression of NOTE OFF is**

activated (PROGRAM CHANGE / data byte = 121,) the 2nd data byte (velocity) will be set to 1, if the DMX level is equal to 0 or 1.

1st data byte = **51** (hex 33) send a pair of messages contained of

POLY KEY PRESSURE and PITCH CHANGE message

The modified DMX channel is transferred as a POLY KEY PRESSURE message, the corresponding DMX level is transferred with 8 bits of resolution as a PITCH CHANGE message. Both are sent in the same format as the command described above for transmission of DMX levels via DMX OUT. This data format is most useful, when only one MIDI channel is available for the DMX Control Box.

2nd data byte: desired **threshold (0 to 127)**

threshold = 0 causes deactivation of this message

Comment: If the activation of these messages was stored in one of the presets 0 to 3, they are reactivated when the preset is loaded and sent on the MIDI channel which is presently configured (if applicable: on additional MIDI channels, too), independent of which MIDI channel was configured when the preset was saved. But all messages are suppressed which would be sent on MIDI channels >16.

Activate "automatic" messages equally for all DMX channels with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

Same activation and annotations as described above for an individual DMX channel **but the**

1st data byte is 58,59,60,61 (hex 3A-3D) to activate the message types described above under 48,49,50,51 equally for all DMX channels.

Switch "automatic" messages globally ON and OFF with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 40 (hex28) switch automatic messages globally **OFF**

data byte = 41 (hex29) switch automatic messages globally **ON**,

formatted as **MIDI channel messages (default)**

the automatically sent MIDI messages are formatted corresponding to the configuration made last with POLY KEY PRESSURE DB1= 48, 49, 50, 51, 52 or 58, 59, 60, 61, 62 details see above

data byte = 42 (hex2A) switch automatic messages globally **ON**,

message formatted as **SysEx/ASCII**

When globally switched off, all automatic messages are kept stored individually per DMX channel. Only their output is suppressed. This setting is stored in any of the presets no 0 to 3. When switched on again, the previous configuration including threshold levels is reactivated - but the message format may differ.

Switch emission of "preprogrammed" strings globally ON and OFF with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 50 (hex32) switch emission of strings globally **OFF**

= 51 (hex33) switch emission of strings (DMX channel 500-512) globally **ON**

Comment: When globally switched off, all preprogrammed strings which are released by changes of DMX channels 500-512 are kept stored. Only their emission due to special levels at DMX IN is suppressed until they are activated again.

Poll DMX transmit buffer at DMX channel ="SLOT" and subsequent ones with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 36 (hex24) Response: mixture of NOTE ON and CONTROL CHANGE
1st data byte = 37 (hex25) Response: sequence of POLY KEY PRESS (1 MIDI channel)
2nd data byte (0 - 127) = number of polled DMX channels (1 -128)
2nd data byte=0 polls 128 channels

Poll DMX receive buffer at DMX channel ="SLOT" and subsequent ones with :
POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

1st data byte = 40 (hex28) Resonse: sequence of MIDI NOTE ON and/or CONTROL CHANGE messages corresponding to "express setting" described above.

1st data byte = 41 (hex 29) Response: first a POLY KEY PRESSURE (no. of 1st DMXchannel) followed by POLY KEY PRESSURE messages (successive DMX level values).

1st data byte = 42 (hex 2A) Response: first a MIDI NOTE ON or CONTROL CHANGE followed by CHANNEL PRESSURE messages **without** running state.

1st data byte = 43 (hex 2B) Response: first a MIDI NOTE ON or CONTROL CHANGE followed by CHANNEL PRESSURE messages **WITH** running state.

2nd data byte (0 - 127) = number of polled DMX channels (1 -128)

2nd data byte=0 polls 128 channels

Set length of DMX cycle to the actual value of "SLOT" with
PROGRAM CHANGE

MIDI channel as configured

data byte = 64 (hex40)

Comment: The minimum length of the DMX cycle (= number of DMX channels transmitted between two DMX reset pulses) is 24.

Switch acceptance of Running State in commands with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 56 (hex38) Running State in commands **forbidden**

= 57 (hex39) Running State in commands is **accepted** (default)

Comment: this command is locked in ASCII preferred modes of operation (rotary switch pos 1,2,4). Then running state is forbidden: Exception: CHANNEL PRESSURE command.

CONTROL CHANGE sets odd DMX levels: activate and deactivate with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 60 (hex3C) DMX level = MIDI data byte *2

data byte = 61 (hex3D) DMX level = MIDI data byte*2+1 (default)

Comment: when this effect is deactivated, CONTROL CHANGE commands have the same effect as NOTE ON commands. As a special case with NOTE ON as well as with CONTROL CHANGE the MIDI-data byte 127 is transformed into DMX level 255.

Exchange CONTROL CHANGE and POLY KEY PRESSURE functionally with:
PROGRAM CHANGE

MIDI channel as configured

data byte = 62 (hex3E) CONTROL CHANGE and POLY KEY PRESSURE
have the same meaning as described in this manual
(default)

= 63 (hex3F) **all functions of CONTROL CHANGE
and POLY KEY PRESSURE are exchanged with respect
to the description in this manual**

Comment: With this option, the command set may be better fitted to the particular control task or features of the given control equipment.

The default setting as described in the manual is most appropriate when the control equipment (sequencer for example) provides good possibilities to work with POLY KEY PRESSURE messages and/or simultaneous fade ramps shall be (graphically) executed on a sequencer with CONTROL CHANGE messages. Recommended for recording and control with sequencer.

The alternative setup with exchanged functions shows advantages if many settings are made preferably with a controller pad or with a programmable keyboard - especially when the POLY KEY PRESSURE messages, which are less frequently used in normal MIDI operation, are not or badly supported. Recommended for control by keyboard or controller pad and evaluation of lighting scenes (presets).

Block command execution with:
PROGRAM CHANGE

MIDI channel as configured

data byte = 36 (hex 24)

Comment: this command is especially useful for example, when lighting scenes (presets, see below) to be loaded or saved are selected with a controller pad or similar. The blocking feature prevents that all intermediate values, which are emitted during motion of the controller, are executed. When programming of lighting scenes is made with a controller pad, we recommend that a pair of two key buttons is reserved for the block and unblock function.

Terminate blocking of commands and execute last received command
immediately with:

PROGRAM CHANGE

MIDI channel as configured

data byte = 37 (hex 25)

Comment: While blocking of commands is active, the Ethernet / DMX Control Box remembers the last received MIDI channel message. This is executed immediately after the blocking is terminated. This feature is helpful to set controllers etc. fast and precisely without sending intermediate values.

Save, store presets (= lighting scenes) with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

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 to 127 (hex7F): add 0 - 127 to basic value

Comment: After the DMX Control Box is switched on, the code switch was rotated or a MIDI RESET message was received, generally preset no.0 is loaded. Together with presets no. 0 bis 3 the system configuration is stored in addition to the actual lighting scene. This includes especially: format of DMX levels reported via MIDI OUT, running state behaviour, reaction on CONTROL CHANGE commands, LOOP, chaser loop. Together with presets no. 4 to 383 only the lighting scene is saved.

Special cases: store preset partially (new from Dec.2010 / revision number 41)

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.

Attention: with this special feature exclusively DMX levels are re-stored, in no case the channel specific details of the system configuration.

Save flash pattern with :

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE) (new from Dec.2010 / rev.no 41)

MIDI channel as configured

1st data byte = 126 (hex 7E)

2nd data byte = 15 (hex F) any values else are ignored

Comment: For detailed description, see flash start command above

Load presets (= lighting scenes) with:

POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

MIDI channel as configured

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): add 0 - 127 to basic value

Special cases: load preset partially (new from Dec. 2010 / revision number 41)

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 of the DMX bus 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.

Attention: with this special feature exclusively DMX levels are loaded, in no case the channel specific details of the system configuration.

Alternatively load presets (= lighting scenes) with :

NOTE ON

MIDI channel as configured

1st data byte (0 -127): preset number to be loaded **minus basic value**
corresponding with the range setting which was made before with a
PROGRAM CHANGE command

2nd data byte: not relevant, except when velocity = 0, then no preset is loaded

Comment: this method is useful to change lighting scenes quickly during "live operation", but helpful too for simple programming of a lighting track with a sequencer. As preparatory work an appropriate set of presets has to be produced. To manage this best, the number of keys of the keyboard which is used for this task has to be considered and possibly the pitch has to be transposed. If organized well, this way lighting design of a bigger repertoire may be performed quite efficient.

This mode of operation has to be activated before with a PROGRAM CHANGE command, data byte 96 to 111 (hex 60 to hex 6F). It remains active until it is changed with new PROGRAM CHANGE command. The default setting for NOTE ON is "short command" to simply set a DMX channel and level with one MIDI message.

Change mode of operation with :

PROGRAM CHANGE

MIDI channel as configured

data byte 96 to 111 (hex 60 to 6F) **NOTE ON commands load presets (see above)**

listed more specifically:

data byte = 96 (hex 60): **basic value 0**, i.e. NOTE ON loads preset no. 0-127

97 (hex 61): **basic value 128**, i.e. NOTE ON loads preset no. 128-255

98 (hex 62): **basic value 256**, i.e. NOTE ON loads preset no. 256-383

and additional **special functions: load presets partially**

data byte = 99 ... 101: basic value 0, i.e. loads preset no. 0 - 127 partially

99 (hex63): NOTE ON loads DMX channels 129 - 256 from preset to DMX OUT chan. 1 - 128

100 (hex64): NOTE ON loads DMX channels 257 - 384 from preset to DMX OUT chan. 1 - 128

101 (hex65): NOTE ON loads DMX channels 385 - 512 from preset to DMX OUT chan. 1 - 128

data byte = 102 ... 104: basic value 128, i.e. loads preset no. 128 - 255 partially

102 (hex66): NOTE ON loads DMX channels 129 - 256 from preset to DMX OUT chan. 1 - 128

103 (hex67): NOTE ON loads DMX channels 257 - 384 from preset to DMX OUT chan. 1 - 128

104 (hex68): NOTE ON loads DMX channels 385 - 512 from preset to DMX OUT chan. 1 - 128
data byte = 105 ... 107: basic value 256, i.e. loads preset no. 256 - 383 partially
105 (hex69): NOTE ON loads DMX channels 129 - 256 from preset to DMX OUT chan. 1 - 128
106 (hex6A): NOTE ON loads DMX channels 257 - 384 from preset to DMX OUT chan. 1 - 128
107 (hex6B): NOTE ON loads DMX channels 385 - 512 from preset to DMX OUT chan. 1 - 128
data byte = 108 ... 111: basic value 0, i.e. loads preset no. 0 - 127 partially
108 (hex6C): NOTE ON loads DMX channels 1 - 128 from preset to DMX OUT channels 1 - 128
109 (hex6D): NOTE ON loads DMX channels 129 - 256 from preset to DMX OUT ch. 129 - 256
110 (hex6E): NOTE ON loads DMX channels 257 - 384 from preset to DMX OUT ch. 257 - 384
111 (hex6F): NOTE ON loads DMX channels 385 - 512 from preset to DMX OUT ch. 385 - 512

data byte = 120 (hex 78) **NOTE ON messages set DMX channel and level.**

velocity = 0 is accepted and sets the DMX level to 0 (=default)

Any NOTE OFF message (with arbitrary velocity) sets the DMX level to 0

data byte = 121 (hex 79) **NOTE ON messages set DMX channel and level.**

NOTE ON messages with 2nd data byte (velocity) = 0 are ignored

and all NOTE OFF messages are ignored

but: NOTE ON messages with velocity = 1 set the DMX level to 0.

data byte = 122 (hex 7A) ask configuration of DMX channel= "SLOT"

The response comes as a SysEx message with content equal to ASCII command "Q".

data byte = 124 (hex 7C) activates merge operation with **internal DMX clock**

data byte = 125 (hex 7D) activates merge operation with
external synchronisation by the signal at DMX IN

data byte = 126 (hex 7E) activates **DMX receive** operation.

The **signal is looped through** from DMX IN to DMX OUT.

It can be polled and automatic messages can be activated

data byte = 127 (hex 7F) **clears** all buffers and working registers

This implies: the actual mode of operation is reset to the default setup.

Presets are not modified

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.

Comment: Many MIDI sequencers and other MIDI control equipment demand that when formatting a PROGRAM CHANGE message the data value (program number) has to be entered one higher than physically transmitted as data byte. (Example: user entered program #1 is transmitted as hex C0 00)
The ETHERNET/DMX Control Box evaluates PROGRAM CHANGE messages always according to the physically received data bytes.

Some MIDI devices automatically send together with every PROGRAM CHANGE message a BANK SELECT message (=CONTROL CHANGE to controller no.20.). If this conflict is noticed in your installation, we recommend not to address any DMX receiver to DMX channel no. 20.

As an alternative, all commands which can be performed with a PROGRAM CHANGE can be sent as a POLY KEY PRESSURE command, whose 1st data byte is =127. Then its 2nd data byte is equal to the data byte of the corresponding PROGRAM CHANGE command.

System Exclusive Commands enclosing ASCII text

Any ASCII text based command or sequence of commands - as described in part 1 of this manual - **may be used in a MIDI context**. For this reason it is put or embedded into a

System Exclusive message frame:

The SysEx command frame embeds ASCII text messages as a kind of brace to make them compatible with the MIDI format. Any SysEx command consists of a SysEx Header, the data content and the terminating EOX byte.

The **SysEx header** always is the same 4 byte sequence, which contains the Cinetix SysEx manufacturer ID:

hexadecimal: **F0 00 20 5D**

or decimal: **240 0 32 93**

EOX is a single byte, which has the value **hexF7=decimal 247** generally throughout MIDI.

Into this System Exclusive command frame different ASCII text commands can be packed or embedded.

--- **First the SysEx header opens the ASCII command interpreter.**

--- **Then every ASCII 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 smaller than maximum are finished by starting the next command (input of a command letter) or when an adequate number or leading zeroes is put ahead or the SysEx packet is finished with EOX.

--- **The EOX-message causes** immediate execution of the last entered command and **closes the ASCII command interpreter. From now on incoming data are evaluated as MIDI channel messages** (until a next SysEx header).

Within a SystemExclusive command frame or packet an arbitrary number of ASCII commands can be contained, but the text length per SysEx packet should not exceed 80 characters.

"Ethernet / DMX512 Control Box" User Manual part 4

Operation as Art-Net™ node

(Art-Net™ is Designed by and Copyrighted for Artistic Licence(UK)Ltd)

This mode of operation is active when the rotary switch is in one of the positions 6,7,8.

Basic features are supported like they are used in most Art-Net "server" applications, but no DHCP, no RDM, no firmware/UBEA update and no video/media support.

In **positions 6 and 7** the primary IP and subnet mask configurations proposed by Artistic Licence are active.

In **position 8 the custom setting is active** which is effectively configured with commands N (default UDP destination IP) and M (subnet mask). To enable Art-Net broadcast, the UDP destination IP is automatically converted in a way, that those bits of the subnet mask which are set to zero, are converted to one in the Art-Net destination IP (except the highest byte).

For example, the default UDP destination IP setting =192.168.0.1 and subnet mask 255.255.255.0 will result to the custom Art-Net IP 192.168.0.255.

With the custom setup, the Ethernet/DMX Control Box and a corresponding control software fit better into small networks, which carry other data simultaneously. Has been tested and verified with "DMX-Control", "vvvv" and "DMX-Workshop".

In addition to the IP, the Art-Net specific "**Subnet**" and "**Universe**" have to be edited during configuration with the commands A, U and V. Each can take values between 0 and 15(hexF).

Command "A" sets the Art-Net specific "Subnet" equally for DMX OUT and DMX IN. Command "U" sets the Art-Net specific "Universe" for DMX OUT, while command "V" sets the Art-Net specific "Universe" for DMX IN. Default is "0" for all of these settings. These settings may be overridden temporarily by commands of the "ArtAddress" protocol.

Notice: in some context, Art-Net uses the "Subnet" setting as high nibble of a combined term called "universe". In the configuration mode of the Ethernet/DMX Control Box only the **low nibble** of the combined term is entered / edited as "Universe" - do "Subnet" separately!

Operation as UDP <--> DMX "server"

The corresponding commands were already described in parts 1 to 3 of this manual. Here **some specialities of communication with an UDP "client"** (=control device) are lined out.

In its basic configuration, the **UDP <--> DMX server** accepts all commands, which are directed to **its own IP address and its configured UDP source port** (config. command S). Any response is retransmitted to the configured **destination IP and port (config. command D)**. In simple configurations it is recommended to configure source and destination port equally.

At devices delivered before May 2011 the source and destination port is always configured equally with command P.

In normal case the **IP addresses and ports of both communication partners have to be configured mirror-like to each other** and merely a point- to-point data exchange is possible

When operated as an UDP<-->DMX server the **ability of communication can be extended by 3 methods:**

--- Broadcast:

The UDP protocol generally includes that all datagrams which **have set all IP bits =1** where the subnet mask bit is =0, are accepted by every UDP receiver with adequate port setting. Most simple networks are of class C (subnet mask usually 255.255.255.0). **In this case the last byte of the destination IP has to be set = 255 to achieve transmission and reception of a datagram in broadcast style.**

Broadcast transmission is "network public", i.e. will be received by all UDP devices with the same port.

--- Multicast:

(new from Dec. 2010, revision number 41)

If the first byte of the UDP destination IP is configured in the range 224 to 239, the Ethernet / DMX Control Box automatically initiates itself as a member of the corresponding multicasting group.

The main effect of this setting is that UDP datagrams which do not fit to the IP range of the local network, are not sent to the gateway IP, but instead to the multicasting address, which is laid over the general IP configuration of the network.

At devices delivered from May 2011 for UDP modes of operation (switch positions 4,5,B;C;E) the default IP is set to multicasting group 225.0.0.37 and destination as well as source port 21928
 This feature is especially useful to make the ethernet available as a MIDI port for PC applications when the ipMIDI driver is used (see "www.nerds.de" for download)

--- Adaptive destination IP:

If in UDP <--> DMX server operation the **last byte of the destination IP is configured =254, the Ethernet/DMX Control Box adopts the last sender-IP-byte of every datagram which is directed to it.** This is copied as destination IP into any responded message, including acknowledgement prompts. **I.e. the answer is always returned to that client, which has sent the last command.** In contrast to broadcast, this mode of communication is "network-private", so other network participants cannot receive it. **Adoption of the destination IP is temporary, no setup is changed.**

Dynamic redirection of UDP messages (rotary switch positions B and C only)

This is performed with a MIDI SysEx style message **fed into the serial interface** which has to be formatted as follows:

hex F0 0 20 5D 23 **IP_(low)** F7 - or decimal 240 0 32 93 35 **IP_(low)** 247

The term IP_(low) is entered as a decimal number 0 to 255 in ASCII text format.

Only the lowest byte of the UDP destination IP may be changed. **This redirection is temporary** until it is changed by a new command. When the rotary switch is changed or a reset is performed by any reason, the permanently configured value will be reactivated.

Operation as TCP server or UDP node for serial data transfer

When any of these modes of operation is used, a serial or MIDI adaptor cable is plugged into the DMX OUT socket (see Appendix D). No signal input is allowed at DMX IN (except when connected to RS-422 directly there). During configuration special care is necessary for the parameters B (baud rate), T (packet timing interval), E (server connection timeout) and **K (byte to ASCII/MIDI conversion).**

Transfer buffers are provided. But if data are transferred permanently via Ethernet at higher speed than the serial port can transmit, any buffer will overflow sooner or later.

A TCP server can only "listen" for a connection request from a TCP client. It cannot initiate a connection by itself. To use it as a TCP client, take rotary switch position D. An UDP node however, does not technically distinguish between "server" and "client", but context dependent it may be regarded as one of these.

Rotary switch position 9 and C:

TCP Transparent mode: all bytes are transferred bidirectionally without change. Only the baudrate at the serial port is configurable (including MIDI bit rate).

Rotary switch position A and B:

These modes of operation are **specially useful to control MIDI or other binary equipment by means of an ASCII capable controller and vice versa.**

K ="Konverter" configuration:	data via Ethernet bidirectional <-->	Serial&MIDI OUT	Serial&MIDI IN
0	ASCII triplet *	byte 0 ... 255	byte 0 ... 255
1	byte 0 ... 255	ASCII triplet *	ASCII triplet *
2	byte 0 ... 255	MIDI Note On **	MIDI channel message ^{5*}
3	byte 0 ... 255	MIDI Control Change ***	MIDI channel message ^{5*}
4	byte 0 ... 255	ProgrCh, ChannPress ^{4*}	MIDI channel message ^{5*}
5	string ^{6*}	---	MIDI Control Change***
5	MIDI Control Change***	string ^{6*}	---

* "ASCII triplet": any **binary byte is described as a text sequence** as follows:the hexadecimal binary code is represented by 2 ASCII digits (0-9,A,a,B,b,C,c,D,d,E,e,F,f). For binary codes less than 16 (hex10) a leading zero has to be inserted. Byte entry is terminated by a SPACE (ASCII Code 32 /hex20) or by a CARRIAGE RETURN 13 (hex D). Text input which cannot be interpreted or is incorrect is responded by a question mark '?'.
 ** MIDI Note On
 *** MIDI Control Change
^{4*} Program Change, Channel Pressure
^{5*} MIDI channel message
^{6*} string

** The MIDI NOTE ON message is built as follows: MIDI channel as configured. Bits 0..6 of the Ethernet byte become the 1st MIDI data byte (pitch). If bit7 of the Ethernet byte is =0, the 2nd MIDI data byte (velocity) becomes 64(hex40). Else, if bit7 of the Ethernet byte is =1 , the 2. MIDI data byte becomes 127(hex7F)

*** The MIDI CONTROL CHANGE message is built as follows: MIDI channel as configured. Bits 0..6 of the Ethernet byte become the 2nd MIDI data byte (controller value). If bit7 of the Ethernet byte is =0, the =0 ,the 1st MIDI data byte becomes 80(hex50) Else, if bit7 of the Ethernet byte is =1 , the 1st MIDI data byte becomes 81(hex51).

4* If bit 7 of the Ethernet byte is = 0, a PROGRAM CHANGE message is sent, if bit 7 is= 1 ist, a CHANNEL PRESSURE message is sent. MIDI channel as configured. The data byte of both types of messages contains bits 0..6 of the Ethernet byte.

5* The **byte which is transmitted via Ethernet** is built as follows: MIDI are only accepted at the configured MIDI channel, any else are ignored. Running state is accepted.

NOTE ON: The 1st MIDI data byte (pitch) becoems bits 6...0 of the Ethernet byte. if the 2nd MIDI data byte (velocity) is < 65 (hex41), then bit 7 of the Ethernet byte becomes =0, else =1.

MIDI messages with velocity =0 are ignored.

CONTROL CHANGE: the 2nd MIDI data byte (controller value) becomes bits 6...0 of the Ethernet byte. If the 1stMIDI data byte(controller No.) is even, then bit7 of the Ethernet byte becomes = 0, if the controller no. is odd, then bit7 of the Ethernet byte becomes = 1

PROGRAM CHANGE: The MIDI data byte becomes bits 6...0 of the Ethernet byte, **bit 7 becomes = 0.**

CHANNEL PRESS: The MIDI data byte becomes bits 6...0 of the Ethernet byte, **bit 7becomes = 1.**

6* Any of the strings stored with the ASCII : (colon) command (see page19) may be triggered for transmission via Ethernet or via serial port by a MIDI Control Change message which is built as follows:

1st data byte (controller no.) 0 - 12 (hex 0 .. C), 2nd data byte (controller value) hex 0,2,4,6,8,A,C,E.

To make this concept more transparent:

Configuration modes 0 and 1 are primarily intended to control MIDI equipment from an ASCII text based software and to display MIDI feedback messages in ASCII text format.

Mode 0 is specially useful, when the Ethernet / DMX Control Box gets ASCII commands via Ethernet and shall control a MIDI instrument (or binary controlled device else) which is connected to the serial interface. Vice versa binary feedback messages then shall be sent as text via Ethernet ("**Server**").

In **Mode 1** however the Ethernet / DMX Control Box translates ASCII text data which are input at the serial interface binary format to transmit them via Ethernet and vice versa ("**Client**").

Modes 2 ,3, 4 are specially useful, to control a non-MIDI application (binary or ASCII text based) from a MIDI environment with easily generated MIDI messages ("**Client**"). Any of the described MIDI message types may be entered at the serial interface at any configuration of parameter K. The serial output format however is fixed by configuration of parameter K.

Operation as TCP Ethernet client

Depending on the configured the baud rate and the attached RS-232- or MIDI cable, a **bidirectional converter between this kind of serial port and the Ethernet** is established. **Data transfer is transparent**, that means the client transfers data without change from one medium to the other one.

Communication is **handled with the TCP protocol**. Connection parameters (own IP address, subnet mask, IP address of the called server, communication ports, baudrate, MIDI channel) are set in the configuration phase.

Connection control:

In addition to data transfer, the client has to handle the TCP connection. For this purpose a single control code is reserved, which is not available for transparent data transfer. Different options are available, which are optimized for specific scenarios, but can be used universally.

It has to be taken into account, that all commands to establish a TCP connection are only executed in the "disconnected" state, thus are available freely during connection. **The only code which is forbidden during transparent operation, is the command to terminate the connection.**

It has to be taken into account, that during certain applications "invisible" control characters are transferred. Examples "Tab", "Backspace", "Carriage Return", "Line Feed", "ACK", "NAK" or XON / XOFF characters.

Possible conflicts during transfer of arbitrary binary (MIDI) data are avoided best when disconnect code "0" is configured.

TCP-DISCONNECT:

Arbitrary byte, which is entered during configuration.
the default value, which works well with ASCII applications is: hex E (=dec 14, key CTRL-N).

Following special cases do apply:

--- When the disconnect code is **set to "0"**, for disconnection **exactly the following byte sequence** (= MIDI SysEx style) has to be sent

hex F0 0 20 5D 0 F7 or in decimal notation: 240 0 32 93 0 247

This option is preferable when not ASCII but digital data are transmitted. Though it is not absolutely impossible that payload data make a disconnect, it is very improbable.

--- **Codes hex FE and FF (dec 254, 255) are forbidden.**

TCP-CONNECT:

When the connection was established successfully, the default connect code is prompted. If no TCP- could be established, the Ethernet / DMX Control Box returns the disconnect code through the serial interface. This way with repeated trials a connection to a temporarily otherwise connected server is installed quickly.

Optionally all subsequently described commands to establish connection may be embedded into SysEx frame:

hex F0 0 20 5D **<command>** F7 or decimal 240 0 32 93 **<command>** 247

In a "pure" MIDI context <command> may exclusively contain bytes with values 0 to127, for proper function of the Control Box any value is accepted. Only EOX=247(hexF7) is not allowed in <command> because this would terminate the SysEx frame.

TCP-connection to the default IP address, which was entered during configuration:

TCP disconnect code **+ 2** (changed with respect to firmware versions before Aug.2010!)

Temporary select another IP address and perform TCP-connect with it

ASCII text sequence: **# <IP = 1-3digits. decimal number of the range 0 to 255> .**
(notice to insert the period at the end !)

Only the last = least significant IP byte may be exchanged. The first 3 IP bytes of the addressed TCP server have to remain equal. This complies automatically with class C networks.

TCP-connection to the previously active IP address (by default or after # - command):

TCP disconnect code **+ 1** (changed with respect to firmware versions before Aug.2010 !)

Appendix A

Concept and data format of DMX512 transmission: (applies not for Art-Net)

To understand the function of the Ethernet/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.

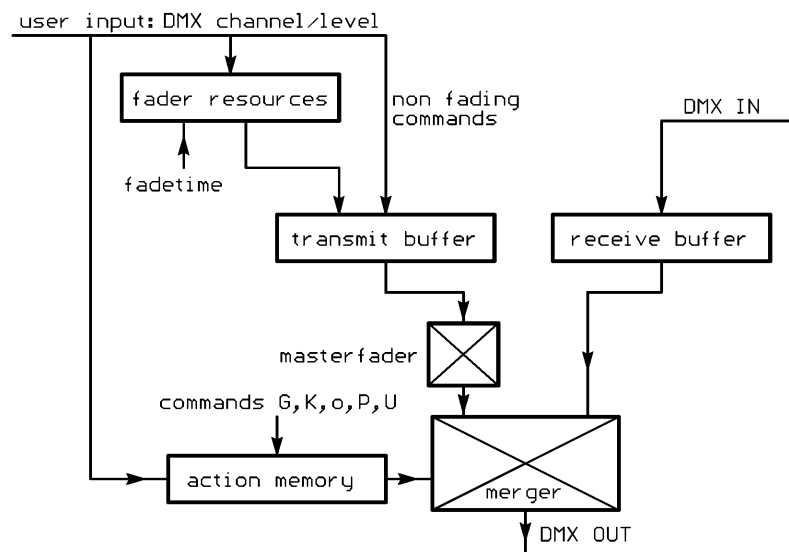
The received DMX signal is permanently written in realtime into the receive buffer.

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 subsequently be written, read or modified otherwise by **user commands**.

For every DMX channel a complete fade process can be performed. The fadetime to be used is set in the **FADETIME register**. This fade process is automatically controlled by the firmware of the box. **Fade processes can be active simultaneously on all DMX channels** and even presets (=stored lighting scenes) are faded over from the previous scene with FADETIME while they are loaded. The actual value of FADETIME is copied

into the respective fader resource when a fade process is started. Straight after then FADETIME can be changed for the next command without influence on any running fader process.

Last not least, a LOOP register is implemented, which stores the actual actual number of slots to be transmitted under user control in each DMX packet. During transmission with external synchronisation LOOP is automatically fitted to the received DMX signal !

After power on or reset, SLOT is initialized with "1", DMX levels and FADETIME are set to "0", the masterfader is set to 100% and LOOP is started with "512".

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

Appendix B

Data format DMX512

The "DMX512" standard respectively its renewed issue ANSI E1.11-2004 ("DMX512-A") 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 addressed with the number of this time slot. Therefore the "addresses" or "channels" in a DMX data packet are called "slots". Per DMX slot one data byte is transmitted. The value of this data byte (range 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.**

Addressing of receivers is defined by the time position ("slot") of their data bytes within the cyclic DMX package. By means of a rotary 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 rotary switch do ignore them. Depending on the setting of their rotary 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 keep 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 Ethernet/DMX Control Box the DMX IN signals are electronically refreshed before retransmission. This means, even when the DMX data are "looped through", the **Ethernet/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.

Appendix C

Structure of MIDI channel messages (super compact crash course)

The first byte is the **status byte, only this one has bit 7 set.**

It is composed of 2 hex nibbles. The most significant (high) nibble describes the message type, the least significant (low) nibble contains the MIDI channel. The MIDI channel is always physically coded one less than written in any MIDI manual or literature. Putting both together: (MIDI channel-1) has to be added to the message type status body.

For MIDI channel no. 1 the different types of MIDI messages show following status byte (=status body):

NOTE OFF:hex80,dec128

NOTE ON:hex90,dec144

POLY KEY PRESSURE(=POLY KEY AFTERTOUCH):hexA0,dec160

CONTROL CHANGE:hexB0,dec176

PROGRAM CHANGE:hexC0,dec192

CHANNEL PRESSURE:hexD0,dec208

PITCH CHANGE:hexE0,dec224

The status byte of PROGRAM CHANGE and CHANNEL PRESSURE is followed by exactly 1 data byte

The status byte of all other message types is followed by exactly 2 data bytes

At all data bytes the most significant bit7 is cleared, the other bits are content specific.

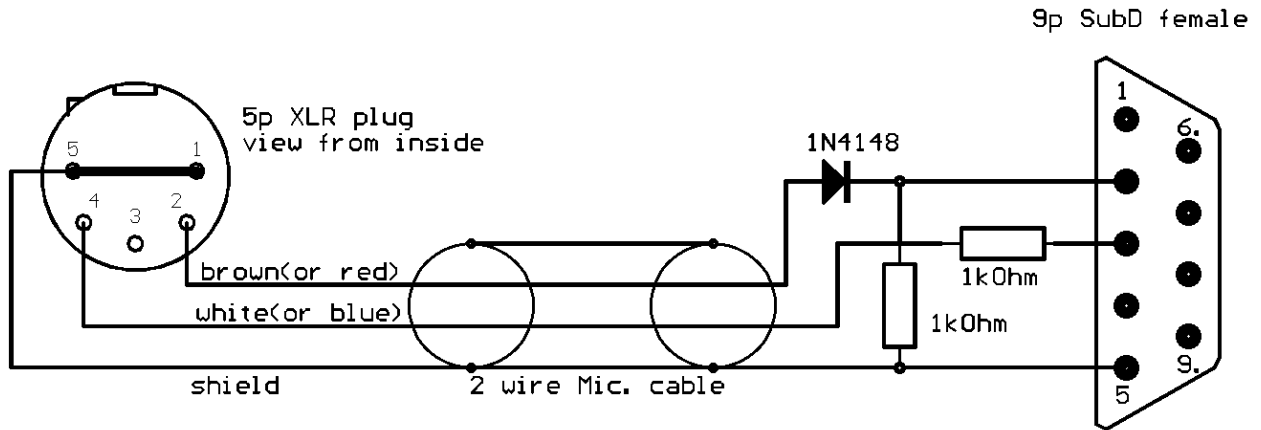
Determination of MIDI note values from common note names:

This scheme is used by several MIDI sequencers. But deviating names of the different octaves are in use, too. Following unique relationship may be helpful: the concert pitch a (440Hz) -- in the table above "A3" - is always corresponding with MIDI note value 69 (hex 45).

	C	C#	D	D#	E	F	F#	G	G#	A	B	H
-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127	-	-	-	-

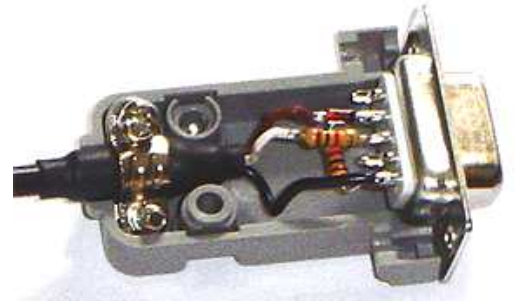
Appendix D

Adaptor cable to connect DMX OUT with a RS-232 COM port:



Build yourself: Use 2-wire shielded microphone cable. Solder a resistor 1kiloOhm from pin5 to pin2 of the SubD connector. Bend the resistor somewhat down to make some space for mounting the following part above: Solder a diode 1N4148 at pin2 of the SubD connector, the ring printed on the diode must show towards the connector socket. Solder a resistor 1 kiloOhm at pin3 of the connector. Shorten the wire ends of resistor and diode to 1mm length. Now solder the brown (red) wire of the mikrophone cable at the diode, the white (blue) wire at the resistor towards pin3 and solder the shield at pin5 of the SubD connector.

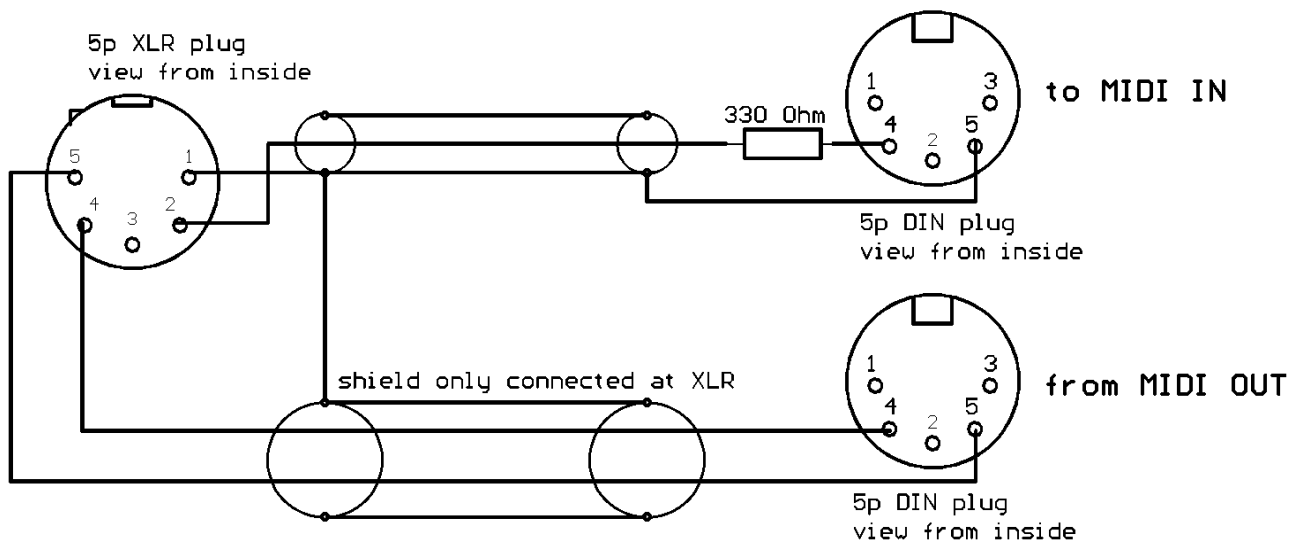
The construction may be stabilized with hot melting glue.



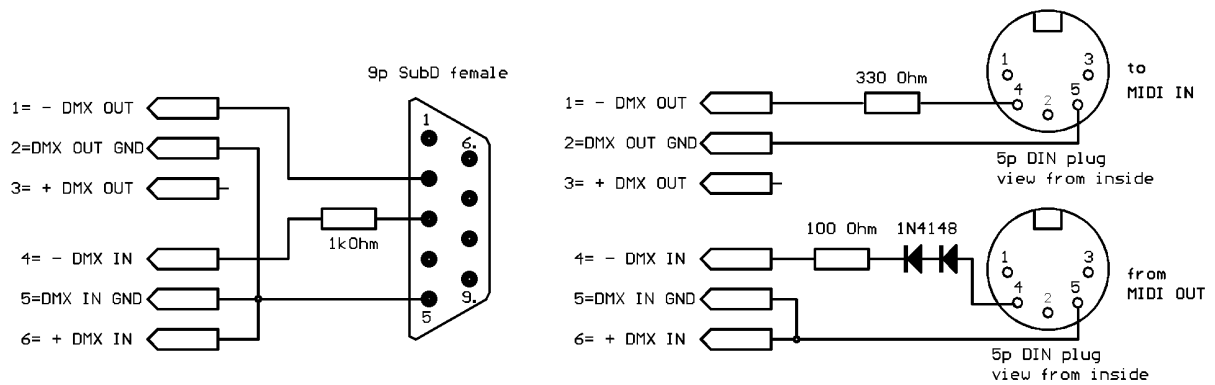
The cable shown in the picture was prepared before as follows: a black wire was soldered to the shield and all was isolated with a heat-shrunk plastic tube.

(In the cables supplied by us, the parts inside the SubD connector are soldered on a small PCB.)

Adaptor cable to connect DMX OUT with a MIDI Interface:



Adaptor for RS-232 and MIDI interface with the clamp version:



Both circuits will only work correctly when the internal termination resistor at DMX IN is active (jumper set = default at delivery).

With an additional combination of diode and resistor for decoupling (as drawn above at the version with XLR-connectors) the noise immunity of the RS-232 output will be improved.

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