

Simple USB / DMX512 Interface with Arduino Micro or Arduino Leonardo

©2015-17 Wolfgang Schemmert Status 10 January 2017

The circuit described here provides an easy to build USB controller for the DMX512 lighting bus.

Three different command sets are selectable, the command set is selected by a jumper, which may be replaced by a switch:

- When the **ASCII text based command set** is selected, the USB interface gets configured as "USB Communication Device" (CDC/ACM class). It provides a virtual RS-232 port.
- The **MiniDMX protocol is implemented** – formally implemented as ASCII text command, i.e. configuration as "USB Communication Device", too
- When the **MIDI channel based command set** is selected, the USB interface is configured conforming to the "USB MIDI Streaming Class", i.e. is seen as a virtual MIDI port.

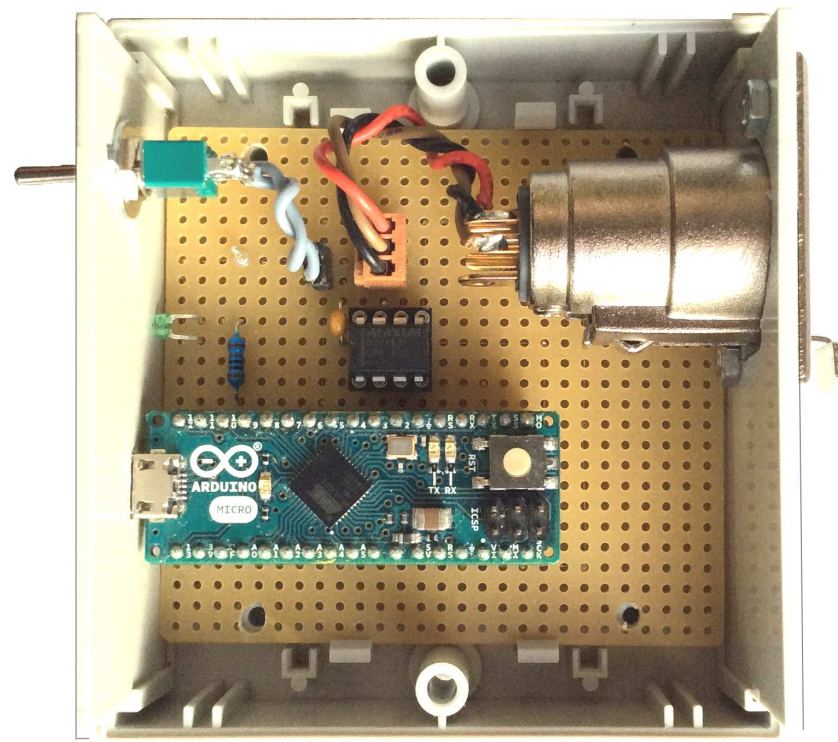
Due to limited SRAM of the ATmega32U4 processor, max. 256 DMX channels are supported. In contrast to Arduino IDE compatible DMX libraries, **special DMX features** like soft fading between lighting scenes, permanent storage of lighting scenes and chaser effect are offered. An **alternative firmware** is available, which supports the **MiniDMX protocol for 512 DMX channels** instead of the MIDI command set.

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

All information about this project is provided 'as is' – without any warranty

Unfortunately the downloadable hex code cannot be programmed with the Arduino IDE **you will need an external ISP programmer**

Hardware for Arduino Micro

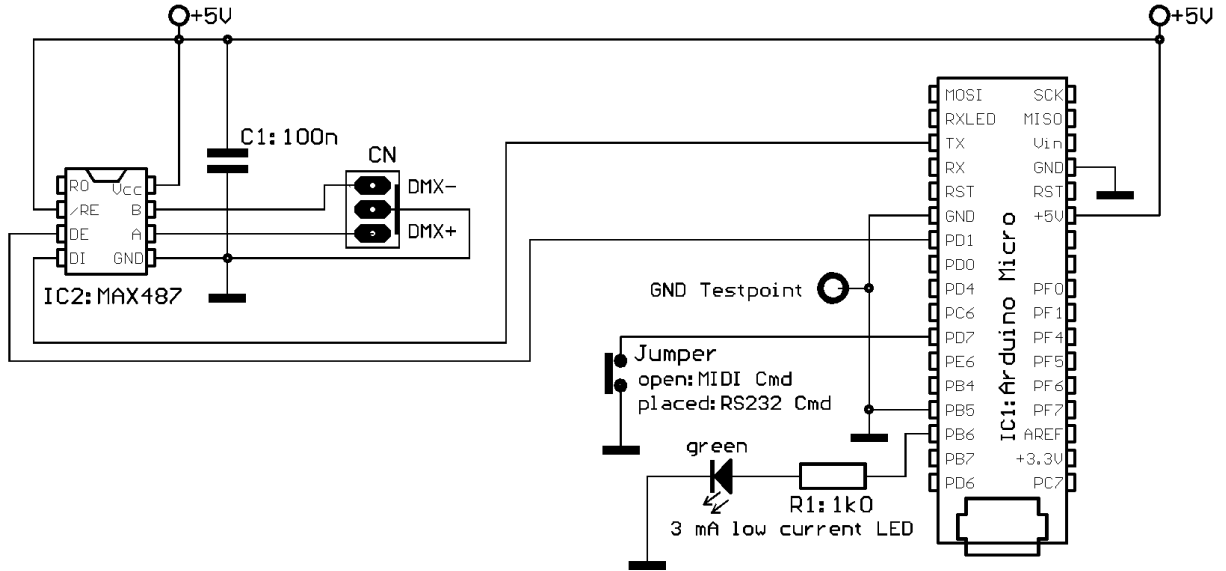


For easy reproduction with simple tools, **the hardware is designed to be built on a Veroboard around an Arduino Micro module** using a minimum of additional parts.

The USB interface is "full speed" grade, the device is **USB bus powered**. The supply current is less than 100mA. A green LED signals presence of power and data flow.

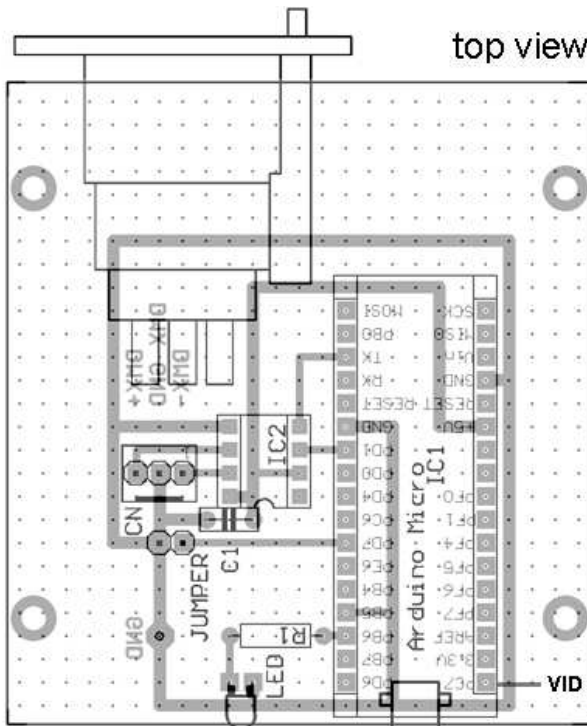
Schematic diagram:

DMX512 OUT (256 channels)

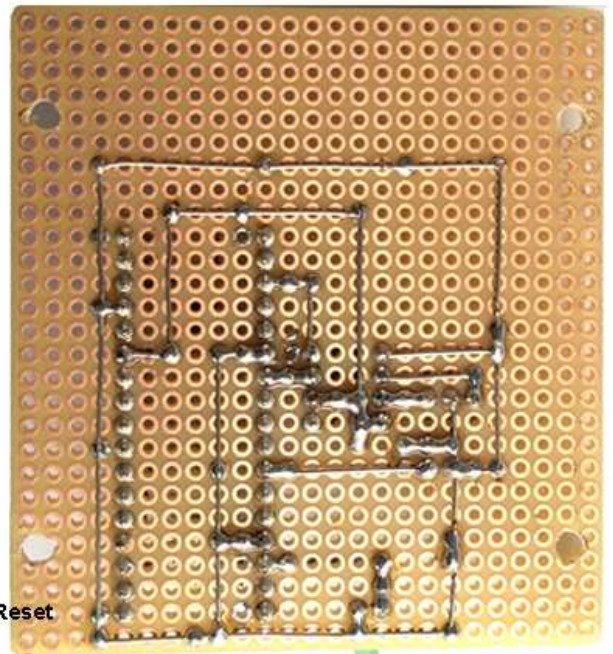


Assembly:

To build an etched PCB instead of the Veroboard, a 1:1 TIF file is provided for download at the website <www.midi-and.more.de/armicdmx.htm>.



mirrored, solder view



The PCB is designed to fit into a "Bopla Unimas85" enclosure, source Reichelt (BOPLA U 85) or Conrad 540803 It also fits acceptably well into an "Eurobox" enclosure, source Reichelt (EUROBOX) or Conrad part no. 523132 Resistor R1 should be carbon or metal film type, min 0.25W, max 5% tolerance. The optimum value may be chosen higher or lower than 1 kiloOhm depending on efficiency of the used LED Capacitor C1 should be multilayer ceramic 5.08mm raster, 50V

IC1: Arduino Micro module

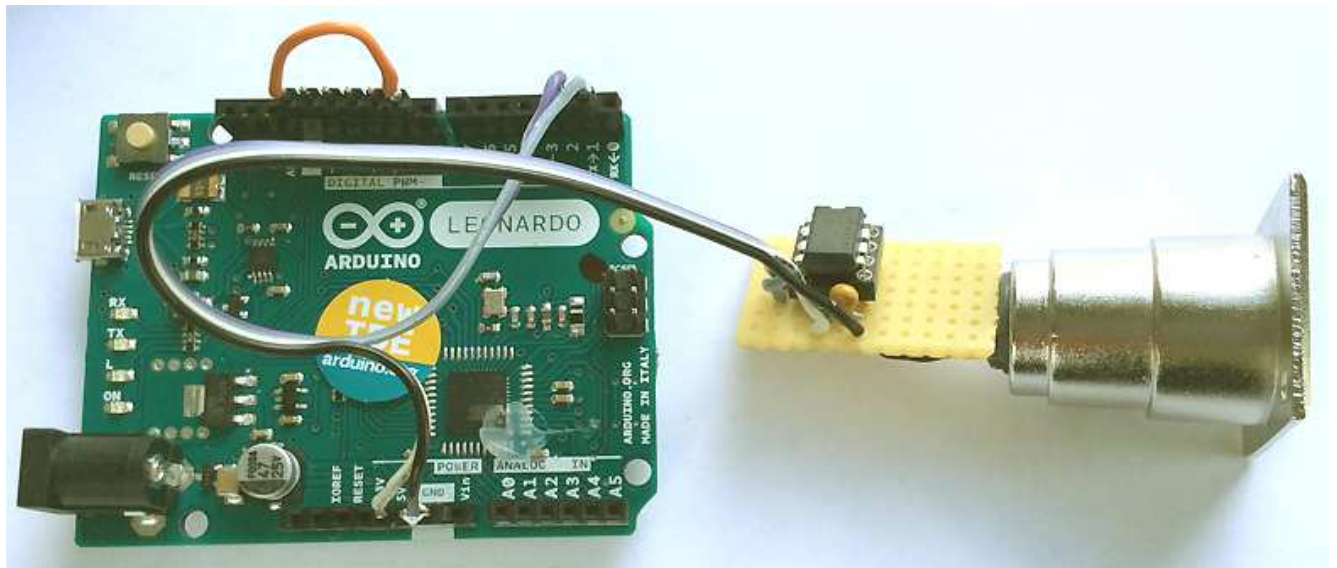
IC2:MAX487 8pDIL or equivalent (e.g. MAX3085, MAX485)

LED: green, preferably 3mm low current type (2-4mA).

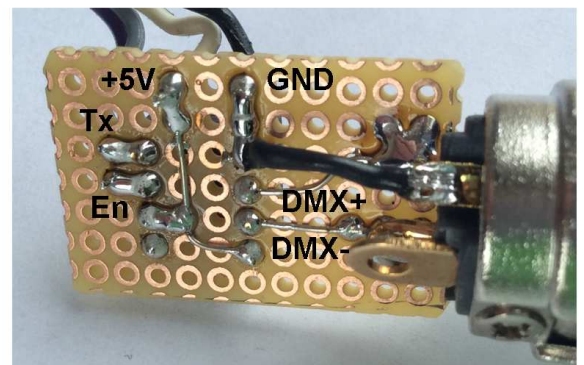
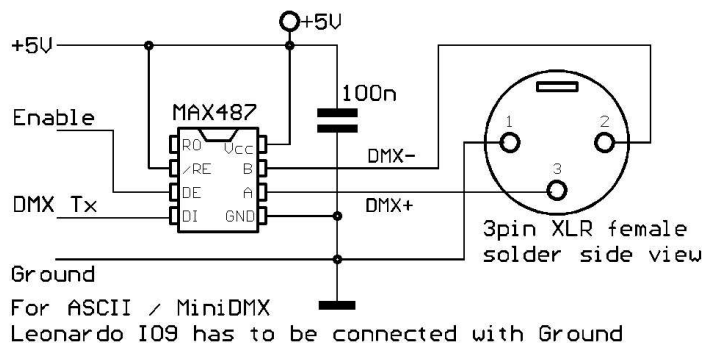
CN: source Reichelt (PS 25/3G BR) or Conrad. The XLR connector may be soldered directly, too.

Hardware for Arduino Leonardo

The DMX transmitter is mounted on a small PCB, which is directly fitted at the XLR connector. For this example, a 3 pin socket was chosen, but the concept works with a 5 pin connector too. If the XLR connector is a type which is mounted through the hole, it is essential that the Veroboard is slim enough to fit through the hole too.



In this wiring example, the black wire is connected with Ground, the white wire with +5V. The grey wire transports DMX data from Leonardo TX (D1) to the MAX487. The violet wire enables/disables the MAX487. It is switched low during USB enumeration to minimize supply current. It is connected with Leonardo D2. Leonardo IO9 is connected to ground with an orange jumper (or switch). This causes the Leonardo configured as virtual serial port at the PC and is used for the ASCII/MiniDMX command set. When the jumper is removed, the Leonardo is configured as virtual MIDI interface and the MIDI command set for DMX is activated. All connections to the Leonardo are made unsoldered with 2.54 mm pinheads.



In the following text only Arduino Micro is referred. But commands and firmware features are the same for Arduino Leonardo. By default, will be registered as Arduino Micro at the PC too, but you are enabled to change this.

Installation and Operation

Depending on the setting of the jumper JP (or switch), the device behaviour is different:

--- If the jumper is open or not installed, the firmware is automatically configured as a USB Communication Device (CDC/ACM class), i.e. will be installed as a virtual RS-232 compatible COM port.

When the microcontroller is programmed new, the default USB VID/PID for the RS-232 compatible mode is the same as for Arduino Micro (VID=2341,PID=0037).

When you start the new programmed Arduino Micro in RS-232 compatible mode first time, the LED initially is dark and the Windows Device manager shows "Unkown Device". Then install "Arduino Micro.inf" manually as guided by Windows navigation.

For this purpose **additionally the appropriate "Arduino Micro.inf" file has to be available**. This is found best the following way: Download the Arduino Software package and install it. Open folder "Drivers". The actual file "Arduino.inf" does not work very well, so it is proposed to unzip "Old_Arduino_Drivers.zip" into new folder named "Old_Arduino_Drivers". File "Arduino Micro.inf" is contained there, it is even compatible with Windows2000

--- If the jumper is set or a connected switch is closed, the firmware is automatically configured as a USB MIDISream device, i.e. most operating systems will recognize it as a virtual MIDI device. In Windows Device Manager (XP and newer only, Windows98/2000 has no built-in USB/MIDI support) it will be listed under "Audio-Video-Game Controller" as "USB-MidiDMX" or simply as "USB Audio Device". If you open any MIDI software, it will be seen as MIDI port "USB-MidiDMX" or simply as "USB Audio Device".

When the interface is connected first time to a PC, it will take some seconds to install the driver. During this period, the LED will glow dark red.

When the microcontroller is programmed new, the default USB VID for the MIDI mode is the same as for Arduino Micro (VID=2341) but the PID is =0038.

Please note that the default values for VID and PID are allowed for device test and evaluation only! For any use else, you have to enter your own VID and PID as follows:

Start the interface in RS-232 compatible mode (no jumper/switch open). You are operating with the ASCII text based command set now. Start a terminal program on your PC (any baud rate works), select the virtual COM port which is assigned to the interface. Enter exactly the sequence:

[V yourVID (as sequence of 4 hex digits) . When this is ok, the letter 'P' will be prompted.

Now enter yourPID (as sequence of 4 hex digits) <return>

(Case independent. The hex words are entered without type specifier like 0x. Leading zeroes of the VID/PID MUST be entered, no spaces between the items). The input will be echoed.

After ASCII command sequence [? <return> the actual VID/PID setting is returned.

The MIDI PID is always one higher.

For your own VID/PID modify the .inf file correspondingly with an ASCII text editor.

To reset VID/PID to default values, connect Arduino PC7 (=IO13) with GROUND during power up! See marker at the assembly figure above.

Hints for operation with the MiniDMX protocol are found below at page 11.

Description of the MIDI command set is found below starting from page 12.

ASCII command set

is active when NO jumper JP is placed or a switch attached there is open. The associated COM port may be set in the Device Manager. Baud rate selection of your control software does not matter, communication is baud rate independent. Handshake is not supported.

Short reference of all ASCII commands

S n	address DMX channel (write SLOT register) for subsequent action (n=1 - 256)	p.5
V n	set DMX level at DMX channel=SLOT (n=0 - 255)	p.6
,n (comma)	increment SLOT first then set level at new DMX channel (n=0 - 255)	p.6
=n	fill block of n DMX channels starting from SLOT+1 with level of SLOT (n=1-256)	p.6
#n	set DMX channel no. n (n=1-256) to DMX level entered by previous command	p.6
+	increase transmit buffer level DMX channel=SLOT by one	p.7
-	decrease transmit buffer level DMX channel=SLOT by one	p.7
^n	add n to transmit buffer level DMX channel=SLOT (n=0 - 255)	p.7
_n	subtract n from transmit buffer level DMX channel=SLOT (n=0 - 255)	p.7
\$	from now DMX level in HEX (only V- , comma- , ^ , _- , R- , Z- , Q- command)	p.7
&	from now DMX level DECIMAL (only V , comma , ^ , _- , R , Z and Q-command)	p.7
Hn	set hue (spectral color) for RGB lamp	p.7/8
Wn	set color saturation for RGB lamp	p.8
Ln	set brightness (luminance) for RGB lamp	p.8
Ts.t	set FADETIME s=seconds t=tenths	p.8
X	stop fade processes and freeze them at actual DMX level	p.8
Rn	read n bytes starting at DMX channel=SLOT from transmit buffer	p.9
Mn	set the masterfader: n=0 to 200 (in percent, see detailed description below)	p.9
in	set length of chaser cycle (n=2 to 89) and start the chaser	p.9
Pt	set duration of chaser step in 1/10 s units	p.9
Gn	enter start scene (preset no.) of chaser cycle . See detailed description	p.9
N	forward chaser immediately by one step	p.10
Q	show content of all DMX registers at DMX channel=SLOT	p.10
~n	save transmit buffer a as preset no. n	p.10
@n	load preset Nr. n into buffers	p.10
	reset all buffers and configuration to default (=delivery) state	p.10
;	(semicolon) dummy command terminates number input (new as of version 1.6)	p.11

MiniDMX protocol

p.11

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.

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 256) is the number of the DMX channel to be manipulated with subsequent commands

Comment: No action is started immediately. But the SLOT 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 selected merge method of the addressed DMX channel if this new level gets actually transmitted.**

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 256) is the number of DMX channels into which the same level is copied. Independent of the value of <block length> DMX channel no.256 is not exceeded.

<channel number>

The **parameter** (range 1 to 256) **addresses the DMX channel**, where the same DMX level is set which was entered by any previous command.

If the fade time is nonzero, a fade process is started, which begins at the actual level of the addressed DMX channel and finishes, when the level is equal to the previously entered level.

Example: S1v35 #5 first sets DMX channel no 1 to level 35 and next channel no 5 to level 35, too.

+ (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. If a fade process is active at this DMX channel, only the final value is increased.

- (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. If a fade process is active at this DMX channel, only the final value is decreased.

^ <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 fade time 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 fade time 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 the decimal number base is set. Because the number base is stored in preset no. 0, loading of this 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 the hex number base is set. Because the number base is stored in preset no. 0, loading of this preset may change the active number base. All messaged DMX level values are coded as decimal numbers without specifier symbol.

H <hue>

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

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

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

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

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

Every new setting of RGB-hue, color saturation and luminance is applied immediately to the 3 DMX channels addressed by SLOT, furthermore each is stored in a global register (not individual per DMX channel). During every new setting of hue, saturation and luminance, the stored global values of the other color components are applied, too.

If set, the fade time gets also applied in combination with the H command. But the fade transition from the previous color tone to the new one is performed along a straight line through the color space, not along the spectral color circle. So, if is faded between very different colors, disagreeable desaturated color tones may appear. To get a perfect color transition, up to 6 subsequent fade steps between neighbored colors have to be performed. The technical handling can be simplified by use of the chaser effect.

W <saturation>

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

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

L <brightness>

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

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

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

X (no parameter)

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

Poll the transmit buffer:

R <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 >

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 device.**

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

Comment: The chaser feature works as follows: a sequence of presets (=lighting scenes) is loaded in a cyclic manner to DMX channels 1 to 128.

Before the chaser can be started, the **step duration** (command P) as well as the **start scene** (command G) has to be adjusted. **Start at scene 50 and step duration 20 is loaded by default.**

DMX channels 129 to 256 are not modified by the chaser and may be used for chaser independent steady lighting.

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

Example: if the chaser cycle is set to 4 and the the chaser start is set to 64, then presets no. 64,65,66,67 are loaded partially, then preset no. 64 again and so on

P <chaser step duration>

Set duration of chaser step in 1/10 s units (step duration = 0 to 25.5 seconds)

Comment: Default at power on: 20 (= 2 seconds). After the duration of a chaser step is over, the chaser automatically loads the next preset in the cycl. After <cycle length> presets were loaded in sequence, the procedure is repeated form <chaser start> .

Step duration 0 only virtually stops the chaser. It can be forwarded by one step with command "N".

G <chaser start>

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

Accepted start values: 0 to 89. Default at power on:50. If the chaser would access a preset beyond 89, the sequence continues with loading preset no. 0 etc..

N (no parameter)

Forward the chaser immediately (asynchronously) by one step

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:

CH=1 TX=27 R=13 G=0 B=0 MF=50% CS=50/0/20 T=3.2

Comment about example: TX reports the present level of the DMX transmit buffer at the channel addressed by CH= SLOT, R shows the actually transmitted level, modified by the Master Fader, G and B show the levels of the next 2 subsequent DMX channels (i.e. RGB show the output of a RGB fixture mit DMX start address= SLOT). MF reflects the actual setting of the masterfader. To explain possible differences you are referred to the description of commands H,W,L,+,-,^,_,T,M, (,). CS describes the actual setting of the chaser in the order: start scene, cycle length, step duration. T reports the fade time.

~ <preset#>

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

Parameter: preset # (range 0 to 89)

Comment: The parameter value of FADETIME and the number base is exclusively stored in preset no. 0. Because this preset is automatically loaded when the device 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 without change of system parameters.

@ <preset#>

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

Parameter: preset# (range 0 to 89)

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

When the fade time is set different from 0, the actual lighting scene is faded over into the loading one with this time constant.

Exception: when preset no. 0 is loaded (switching the device on, for example), the permanently stored value of the fade time and the number base are updated.

| (no parameter)

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

This command has to be entered twice to protect the lighting scene against lapse of command entry

Comment: this is a kind of warm start, not a reset!

All DMX levels of the transmit buffer are reset to "0".

Number base = "decimal".

Masterfader = 100%

The chaser is switched OFF and parameters are reset to their default values

Fade time = 0.0.

Presets are not deleted or changed otherwise.

; (Semicolon, no parameter)

Comment: Dummy command. Terminates number input for immediate execution, no effect else. Is needed when the PD patcher is used for DMX control ("Fudi" protocol). New as of RevNum. 1.7.

Operation with MiniDMX protocol

The MiniDMX protocol is formally implemented as ASCII command 'Z' here.

The practical reason is that each MiniDMX data packet starts with the character hex5A='Z'. This way, **no switch or other configuration is necessary to start or terminate operation of the MiniDMX protocol.** After the data packet is completely received and processed successfully (i.e. put into the DMX transmission cycle), the firmware returns back to the ASCII command main loop and waits for the next ASCII command (or MiniDMX data packet).

If no MiniDMX compatible data byte is received within max. 100 milliseconds, processing of this data packet is cancelled, the firmware jumps back to the main ASCII command interpreter and waits for the next 'Z'.

Once active in MiniDMX mode, this mode is locked for about 1/2 sec as safety against faulty ASCII commands released by incomplete packets. After this timeout, the complete ASCII command set works again. This way, the 'Z' command and MiniDMX constitutes a distinct mode of operation, though it is embedded in the ASCII protocol.

When MiniDMX data packets are received, each active fade process is terminated immediately, the chaser is stopped.

(For a detailed description of the MiniDMX protocol see
<<https://www.dzionsko.de/pmwiki.php?n=MiniDMX.Startseite>>)

MiniDMX packets for 256 DMX channels (type A1) and for 512 DMX channels (type A2) are accepted, but when A2 packets are received, the data for DMX channels 257 to 512 are discarded.

The MiniDMX protocol is supported by a number of good PC based DMX software products like preferably 'DMXControl 3'. It is supported by 'DMXControl 2' and 'Freestyler' too, but the operation is less smooth then. While such a software is active, it occupies the corresponding COM port. Parallel data input by terminal software etc. is impossible then.

While the MiniDMX mode is active, the **LED is blinking** permanently

When the alternative firmware is installed and the jumper is set (or switch is closed) **exclusively the MiniDMX protocol is active supporting 512 DMX channels.** All MiniDMX features else are the same as described above.

The complete ASCII command set and the related additional features - like smooth fade transitions and user predefined lighting scenes - are only available when NO jumper is set. Only 256 DMX channels are supported then due to limited SRAM resources of the microcontroller. The MIDI command set is not available in this case.

MIDI Channel Message protocol of the DMX512 transmitter

By default MIDI channel no. 1 is the base channel for command input. This may be changed at any time by sending a MIDI CONTROL CHANGE message to controller no.124 (hex7C) with controller value equal to the new MIDI base channel (1 to 16 (hex10)) **twice in immediate sequence**: Not to cut the current session, this new channel will become **active not before the next power cycle** or microcontroller reset else. Details see page 14.

Quick start and basic commands:

The most frequently used application is **lighting control with NOTE ON messages from a sequencer**. Precision is improved by additional use of POLY KEY PRESSURE messages. Depending on available equipment and personal liking the POLY KEY PRESSURE messages may be exchanged with CONTROL CHANGE messages. **See description of PROGRAM CHANGE 62 and 63 below.**

To address any of the DMX channels 1 to 127, control data have to be sent on the selected MIDI base channel. How to access DMX channels 128 to 256 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 USB/DMX Interface.

Following **exceptions have to be taken into account**:

If a **POLY KEY PRESSURE** 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 modified (use a PROGRAM CHANGE command with data byte=60)
then for NOTE ON the DMX level will be 255 if the 2nd MIDI data byte is 127 (hex7F)
and for POLY KEY PRESSURE the DMX level will be 0 if the 2nd MIDI data byte is 0

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 POLY KEY PRESSURE, even DMX levels are set with NOTE ON.

Example: To set DMX channel no. 35 to level 200 send NOTE ON, note 35, velocity 100.
To set DMX channel no. 35 to level 201 send POLY KEY PRESSURE, note 35, velocity 100.

To write data into DMX channels 128 to 256 with these commands, the MIDI commands are sent on the **next higher MIDI channel** as described in the table:

coded MIDI channel	1 st data byte	sets SLOT address to-	calculation of 1 st data byte
as selected base channel	1 to 127	1 to 127	= DMX channel
as base channel + 1	0 to 127	128 to 255	= DMX channel minus 128
as base channel	0 special case !	256	= 0

This means: on a sequencer program you have to **reserve a block of 2 MIDI channels** for full control of the USB / DMX Interface 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.

This limitation can be worked around with somewhat **more complex commands, which allow to adjust any DMX channel 1 to 256 with full 8 bit accuracy.** See description of the appropriate CONTROL CHANGE and PITCH WHEEL CHANGE commands below.

The response of the USB/DMX Interface to NOTE ON messages with **velocity=0** or to any **NOTE OFF** messages is prevented with PROGRAM CHANGE command 121. This way you can "play in" DMX level timing with NOTE ON on a keyboard without care about note ends.

Survey of all MIDI Channel Commands (MIDI Implementation Chart)

With PROGRAM CHANGE 62 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.

Abbreviations:

DB means "data byte", DB1 means "1st data byte" (note value, controller number), DB2 means "2nd data byte" (velocity, controller value)

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.**

MIDI message and coded MIDI channel	special data values	function /effect	p.
NOTE OFF	see detailed description	DMX level --> 0. May be blocked with PROGRAM CHANGE 121	21
NOTE ON MIDI channel = base channel.+ next channel	DB1 = DMX channel DB2 = DMX level ./ 2	set DMX channel and level (only 1 MIDI message / 7bit resolution) DMX level = DB2 * 2	15
MIDI channel = base ch.	see detailed description	load preset 0-89 by MIDI note (i.e. directly with a keyboard key stroke).	21
POLY KEY PRESSURE MIDI channel = base channel.+ next channel	DB1 = DMX channel DB2 = DMX level ./ 2+1	set DMX channel and level (only 1MIDI message / 7bit resolution) DMX level = DB2 * 2 +1	15
CONTROL CHANGE MIDI channel = base ch.	DB1 = 1 DB2 = tenth sec.0-127	set fade time 0 to 12.7 seconds fade time = DB2 ./ 10	18
notice PROGRAM CHANGE 62 !	DB1 = 2 DB2 = tenth sec.0-127	set fade time 10 to 22.7 seconds fade time = 10 + DB2 ./ 10 seconds	18
	DB1 = 3 DB2 = seconds 0-32	set fade time in whole seconds. DB2 = seconds (0 to 31,9 seconds)	18
	DB1 = 4 DB2 = quarter sec.0-127	set fade time in the range 0 to 31.8 seconds with a single controller. Fade time = DB2 ./ 4 seconds	18
	DB1 = 7 DB2 = masterfader %	set masterfader in the range 0 – 127 %	18
	DB1 = 8 DB2 = masterfader-100	set masterfader in the range 100-200 %	18
	DB1 = 16 (hex 10) DB2 = step in 1/10 sec	set step duration of the chaser. 0 = chaser OFF. Controlled by an internal timer.	18
	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	19
	DB1 = 19 (hex 13) DB2 = start preset	set start scene (preset no.) 0-89 of the chaser cycle. See detailed description	19
	DB1 = 21 (hex 15) DB2 = 1 -127	DB2 specifies a note value, which is used to forward the chaser by 1 step (with a MIDI Base Drum for example) instead of setting a DMX level	19
	DB1 = 40 (hex 28)	fill a block of DB2 DMX channels starting from ch. SLOT+1 with the final level of DMX ch. "SLOT"	17
	DB1 = 64 (hex 40) DB2 = 0 to 127	set hue (color tone) of an RGB lamp (3 consecutive DMX channels) see detailed description !	16

MIDI message and coded MIDI channel	special data values	function /effect	p.
CONTROL CHANGE MIDI channel = base ch.	DB1 = 65 (hex 41) DB2 = 0 to 127	set color saturation of an RGB lamp i.e.add grey- or white component (3 consecutive DMX channels) see detailed description !	17
notice PROGRAM CHANGE 62 !	DB1 = 66 (hex42) DB2 = 0 to 127	set brightness/luminance of an RGB lamp (3 consecutive DMX channels) see detailed description !	17
	DB1 = 80 – 81 (hex 50 – 51)	address DMX channel(SLOT) using only one single MIDI channel. For loading data see "pitch change"	15
	DB1 = 84 (hex 54)	set DMX level (@SLOT) to DB2 *2 adjust DMX level with 8 bit resolution to even value	16
	DB1 = 85 (hex 55)	set DMX level (@SLOT)to DB2 *2 + 1 adjust DMX level with 8 bit resolution to odd value	16
	DB1 = 86 (hex 56)	First increase DMX channel (SLOT reg.) there set DMX level (@SLOT)to DB2 *2. adjust DMX level with 8 bit resolution to even value	16
	DB1 = 87 (hex 57)	First increase DMX channel (SLOT reg.) there set DMX level (@SLOT)to DB2 *2 + 1 adjust DMX level with 8 bit resolution to odd value	16
	DB1 = 96 (hex 60) DB2 = preset no.	load preset no. 0 – 89. The fade time is exclusively loaded with preset no. 0	21
	DB1 = 112 (hex 70) DB2 = preset no.	save preset no. 0 - 89. The fade time is exclusively saved with preset no. 0	20
	DB1 =122 (hex7A) DB2 = 0 to 127	poll (entry of DB2) levels of DMX OUT starting from DMX channel=SLOT (DB2=0: poll 128 channels)	19/ 20
	DB1 =124 (hex7C) DB2 = 1 to 16	change active MIDI channel at next power cycle. See detailed description ! (New as of firmware 1.6)	21
	DB1 = 127 (hex 7F)	DB2 has the same meaning as the data byte of the corresponding PROGRAM CHANGE command.	
PROGRAM CHANGE MIDI channel = base ch.	DB = 8	decrease DMX level (minus 1) at channel "SLOT"	17
	DB = 9	increase DMX level (plus 1) at channel "SLOT"	17
	DB = 16 (hex 10)	forward chaser immediately by 1 step	19
	DB = 60 (hex 3C)	Special behaviour NOTE ON:velocity127>DMX 255 and POLY KEY PRESSURE: velocity 0→ DMX 0	20
	DB = 61 (hex 3D)	default: NOTE ON sets DMX even DMX level POLY KEY PRESSURE sets DMX odd DMX level	20
	DB = 62 (hex 3E)	exchange functions of CONTROL CHANGE and POLY KEY PRESSURE	20
	DB = 63 (hex 3F)	don 't exchange CONTROL CHANGE and POLY KEY PRESSURE (default)	20
	DB = 72 (hex 48)	Stop all fade processes, freeze at momentary level	17
	DB = 96 (hex 60)	NOTE ON 0-89 (keyboard) loads preset no 0-89	21
	DB = 120 (hex78)	NOTE ON sets the level of a DMX channel Velocity=0 and NOTE OFF are accepted (default)	21
	DB = 121 (hex79)	NOTE ON sets the level of a DMX channel Velocity=0 is ignored	21
	DB = 122 (hex7A)	ask actual channel configuration (get SysEx message as response)	22
	DB = 127 (hex7F)	Clear All Memory	22
CHANNEL PRESSURE MIDI channel = base ch.	DB= DMX level /.2	increase addressed DMX channel and set there DMX level = DB*2.	16
CHANNEL PRESSURE MIDI ch.= base channel +1	DB= DMX level /. 2	increase addressed DMX channel and set there DMX level = DB*2 +1. (ATmega32U4 only !)	16
PITCH CHANGE MIDI channel = base ch.	DB1 = less than 64 ? DB2 = DMX level /. 2	set DMX level at position SLOT (8bit resolution) If DB1 >= 64, then DMX level = DB2*2 + 1	15

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

NOTE ON or POLY KEY PRESSURE (=POLYPHONIC AFTERTOUCHE)

The first MIDI data byte (controller number or note value/pitch) **sets the DMX channel.**
The 2nd data byte (controller value / note velocity) **describes the light intensity to be set.**
 The DMX level is **twice of** the second MIDI databyte.
 Or vice versa: the **the 2nd MIDI data byte is HALF OF** the intended DMX level (0 to 255).
Following exceptions do apply:
 If the status byte is a POLY KEY PRESSURE,
 then the DMX level is incremented by one, i.e. the next higher odd DMX level is sent.
 This behaviour may be **deactivated** with a PROGRAM CHANGE (data byte = 60).
Exclusively then POLY KEY PRESSURE behaves like NOTE ON:
 if the 2nd. data byte is equal to 127, the DMX level is set to 255.

This simple method works only for DMX channels 1 to 127.

To address DMX slots 128 to 256, the USB / DMX Interface expects control commands on a higher MIDI channel corresponding with following table (only valid for MIDI base channels 1 - 13):

coded MIDI channel	1 st data byte	addresses DMX channel-	calculation of 1 st data byte
as selected base channel	1 to 127	1 to 127	= DMX channel
as base channel + 1	0 to 127	128 to 255	= DMX channel minus 128
as base channel	0 special case !	256	= 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:

CONTROL CHANGE

This alternative command version is used, **when all MIDI messages shall be sent on the MIDI channel = selected base channel.**

The coarse range is selected by 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 2nd data byte
80 (hex50)	1 to 127	1 to 127	= DMX channel
81 (hex51)	128 to 255	0 to 127	= DMX channel minus 128
80 (hex50)	256	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 selected base channel

1st data byte: if equal or greater than 64, "1" is added to the DMX level
 if less than 64 (hex40), nothing is added to the DMX level

2nd data byte: desired DMX level divided by 2

inside the USB / DMX Interface, 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" of 14bit data into the second data byte zu. So it can be used together with simple MIDI equipment, which has only 7 bit capability of PITCH WHEEL CHANGE operation.

Method 2:

CONTROL CHANGE

MIDI channel as selected base channel

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

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

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

Comment: with this method a block of consecutive DMX channels can be set with different levels in a quite compact way.

To set the start channel, the first DMX channel is addressed with NOTE ON, POLY KEY PRESSURE or CONTROL CHANGE, all following ones with a CHANNEL PRESSURE command.

If it is sufficient to set the DMX levels to the nearest even value, any number of DMX channels can be set without repetition of the status byte.

Set hue (color tone) of a RGB lamp with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 64 (hex 40): set hue at the addressed DMX channel + next 2

2nd databyte = hue (color tone) 0 - 127

This will approximately result in following colors. Intermediate hue values will result in intermediate colors:

2nd databyte= 0:red, 22:yellow, 43:green, 64:cyan, 85:blue, 106:magenta, 127:red again.

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

Comment: The command to controller number 76 influences the actually addressed DMX channel (actual entry to the SLOT register set first by a NOTE ON, CONTROL CHANGE or POLY KEY PRESSURE message) and the two next higher DMX channels. It is provided that the RGB setting of the

respective lamp is done on these 3 subsequent DMX channels. All features else of a complex lamp ("fixture") may be used independently.

If set, the fade time gets also applied in combination with these commands. But the fade transition from the previous color tone to the new one is performed along a straight line through the color space, not along the color circle. So, if is faded between very different colors, disagreeable intermediate color tones may appear. To get a perfect color transition, up to 6 subsequent fade steps between neighbored colors have to be performed. The technical handling can be simplified by use of the chaser effect.

Set color saturation of a RGB lamp with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 65 (hex 41): set saturation at the addressed DMX channel + next 2

2nd data byte = color saturation 0 - 127

Comment: "Saturation" describes the amount of white or grey in a color tone (pastel shade). Saturation=127 gets a pure color, saturation=0 gets white or grey without specific color tone.

Set brightness/luminance of a RGB lamp with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 66 (hex 42): set luminance at the addressed DMX channel + next 2

2nd data byte = brightness/luminance 0 - 127

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

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 40 (hex28)

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 selected base channel

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

data byte = 9 increase (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. If a fade process is active at the addressed DMX channel, only the final DMX level is decreased or increased

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

Freeze all DMX levels at their present state.

Set the Fade Time with:

CONTROL CHANGE

MIDI channel as selected base channel

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

2nd data byte = 0 – 127: fade time in 1/10 second units
(setting range 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 – 22.7 seconds).

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

2nd data byte = 0 – 32: fade time in **whole seconds** (0 – 31.97 sec.)
if the 2nd data byte = 32, it is internally limited to 31.9

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

2nd data byte = 0 – 127: fade time in **1/4 second units** (0 – 31.7 sec.)

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

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

Set the MASTERFADER with:

CONTROL CHANGE

MIDI channel as selected base channel

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: 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 buffer of the DMX interface.**

Set chaser step duration with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 16 (hex 10)

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

or = 0 switches the chaser **OFF**

by 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 the next preset in the cycle.

Step duration 0 only virtually stops the chaser. It can be forwarded by one step with PROGRAM CHANGE 16

Set chaser cycle length and start it with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 18 (hex12)

2nd data byte = 2 - 127 (hex 7F) set chaser cycle length 2-89

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

Comment: before the chaser can be started, the **step duration** (CONTROL CHANGE, 1st data byte= 16) as well as the **start scene** (CONTROL CHANGE 1st data byte= 19) has to be adjusted – **Details see description of these commands.**

As soon as <cycle length> presets are loaded in sequence, the procedure repeats from <chaser start>. If the chaser would access a preset beyond 89, the sequence continues with loading preset no. 0 etc..

Example: if the chaser cycle is set to 4 and the the chaser start is set to 64, then presets no. 64,65,66,67 are loaded partially, then preset no. 64 again and so on

Set chaser start preset (lighting scene) with :

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 19 (hex13) ,

2nd data byte: = chaser cycle start preset (0 - 89)

Comment: For an easy start, start at scene 50 and step duration 20 (= 2 seconds) is preset as default. **The chaser loads only the first 128 DMX channels of the preset.** The remaining upper DMX channels can be used for chaser-independent steady light.

Forward chaser immediately (asynchronously) by 1 step with :

PROGRAM CHANGE

MIDI channel as selected base channel

data byte = 16 (hex 10) forward chaser immediately (asynchronously) by 1 step

or activation of a special feature with

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 21 (hex15) ,

2nd data byte = 1 – 126: this note value is reserved to forward the chaser

= 127 deactivates the feature

Comment: Controller value 1-126 of this command specifies, by which MIDI NOTE ON value (pitch) 1-126 the chaser shall be forwarded by one step (**use of this feature disables setting DMX levels with NOTE ON at this DMX channel !**). Velocity of the NOTE ON command does not matter, only velocity=0 is ignored. **May be used to lock the chaser to a GM Drum and Percussion Set.**

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

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 122 (hex 7A)

2nd data byte = (0 bis 127) count of DMX channels to be polled.

= 0: poll 128 DMX channels

This command is essentially used for tests. For better readability, the response comes as a SysEx message which reports the actual DMX levels in ASCII text format. The content of the SysEx message is identical with the response to the ASCII command 'Z', which has the following syntax:

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

Specific behaviour of POLY KEY PRESSURE and NOTE ON:

PROGRAM CHANGE

MIDI channel as selected base channel

data byte = 60 (hex3C) Special behaviour of NOTE ON (Velocity127-->DMX level 255) and POLY KEY PRESSURE (Velocity0-->DMX level 0)

data byte = 61 (hex3D) **Default** behaviour of NOTE ON (DMX level = Velocity*2) and POLY KEY PRESSURE (DMX level = Velocity*2+1)

Comment: in some cases, the final DMX levels 0 and 255 have special meaning at lamp fixtures. IF activated, the special behaviour described above provides better fitness for these situations.

Exchange CONTROL CHANGE and POLY KEY PRESSURE functionally with:

PROGRAM CHANGE

MIDI channel as selected base channel

data byte = 62 (hex3E) **all functions of CONTROL CHANGE and POLY KEY PRESSURE are exchanged with respect to the description in this manual**

data byte = 63 (hex3F) CONTROL CHANGE and POLY KEY PRESSURE have the same meaning as described in this manual (default)

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 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.

The alternative setup with exchanged functions shows advantages if 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.

Save, store presets (= lighting scenes) with:

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 112 (hex70)

2nd data byte = 0 to 89 (hex5A): preset number to be saved

Comment: Together with preset no.0, the actual fade time, special behaviour of POLY KEY PRESSURE and NOTE ON messages (see PROGRAM CHANGE 60,61,62,63,96,120,121) and the ASCII number base (see PROGRAM CHANGE 122 and ASCII commands) is stored permanently.

Load presets (= lighting scenes) with:

CONTROL CHANGE

MIDI channel as selected base channel

1st data byte = 96 (hex60)

2nd data byte = 0 to 127 (hex7F): preset number to be loaded

Comment: After the USB / DMX Interface is powered on or the jumper setting was changed, generally preset no.0 is loaded. Together with this preset the permanently stored fade time, special behaviour of POLY KEY PRESSURE and NOTE ON messages (see PROGRAM CHANGE 60,61,62,63,96,120,121) and the ASCII number base is updated.

Alternatively load presets (= lighting scenes) with :

NOTE ON

MIDI channel as selected base channel

1st data byte (0 - 89): preset number to be loaded

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

This mode of operation has to be activated before with a PROGRAM CHANGE command, data byte 96 (hex60). It remains active until it is changed by a new PROGRAM CHANGE command. The default setting for NOTE ON is "short command" to simply set a DMX channel and level by one single MIDI message.

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.

Select active MIDI channel with:

CONTROL CHANGE

MIDI channel as currently selected base channel

1st data byte = 124 (hex7C)

2nd data byte = 1 to 16 (hex10): MIDI channel to be selected after next power cycle

Comment: This command sets the MIDI channel exclusively while dip switches 2 and 3 both are OFF. It has no effect if any of dip switches 2 and 3 is ON. **The command has to be entered twice in immediate sequence.** Not to disturb the current session, the new MIDI channel gets valid not before the power is cycled or the microcontroller is reset otherwise. New as of firmware version 1.6 (Dec'2015)

Change mode of operation with :

PROGRAM CHANGE

MIDI channel as selected base channel

data byte = 96 (hex 60): NOTE ON loads preset no. 0-89 (see above)

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"

A System Exclusive message will be returned with typically following ASCII content:

CH=1 TX=27 R=13 G=0 B=0 MF=50% CS=50/0/20 T=3.2

Comment: the content of the SysEx message is identical with the response to the ASCII command 'Q':

TX reports the present level of the DMX transmit buffer at the channel addressed by CH= SLOT, R shows the actually transmitted level, modified by the Master Fader, G and B show the levels of the next 2 subsequent DMX channels (i.e. RGB show the output of a RGB fixture mit DMX start address= SLOT). MF reflects the actual setting of the masterfader. CS describes the actual setting of the chaser in the order: start scene, cycle length, step duration. T reports the fade time: full seconds and tenths separated by a dot.

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

This command has to be entered twice to avoid destruction of setup by lapse of command entry

Comment: this is a kind of warm start, not a reset!

All DMX levels of the transmit buffer are reset to "0".

Number base = "decimal".

Masterfader = 100%

The chaser is switched OFF and parameters are reset to their default values

Fade time = 0.0.

Presets are not deleted or changed otherwise.

General comment on PROGRAM CHANGE messages:

As an alternative, all commands which can be performed with a PROGRAM CHANGE can be sent as a CONTROL CHANGE command with 1st data byte = 127 (hex 7F). Then its 2nd data byte is equal to the data byte of the corresponding PROGRAM CHANGE command.

When formatting a PROGRAM CHANGE message, many MIDI sequencers and other MIDI control equipment demand that the data value (program number) has to be entered one higher than the physically transmitted data byte. (Example: user entered program #1 is transmitted as hex C0 00), **The USB / DMX Interface evaluates PROGRAM CHANGE messages with the physical parameter as described in the manual. Check your equipment if the parameter has to be entered by 1 higher**

Some MIDI devices send together with a PROGRAM CHANGE Befehl automatically a BANK SELECT command (=CONTROL CHANGE to controller no.32(hex20)).

If you notice this conflict: don't address any DMX receiver at DMX channel no 32.

Appendix

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

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

Determination of MIDI note values from common note names:

	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	-	-	-	-

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).

contact: wschemmert@t-online.de

* Right of technical modifications reserved. Provided 'as is' - without any warranty. Any responsibility is excluded.

* 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