RS-232/DMX512 Interface

 Status 25 January 2016

RS-232/DMX512 Control Box (DMX IN+OUT)
and RS-232/DMX512 Generator (DMX OUT only)

The intention of this project is to provide a replacement for discontinued Cinetix products with minimized hardware requirement. New Jan'16: The **MiniDMX** protocol is supported (see p.21)

It is **NOT allowed** to use this instrument together with any safety critical applications, where misfunction could result in personal injury oder noticeable material damage !
All information about this project is provided 'as is' – without any warranty nor responsibility !

Hardware

At the next pages reference schematics and corresponding PCB layouts are shown. These are single layer PCBs which are compatible with a veroboard (2.54 mm raster of round copper pads). So the circuit may be reproduced without etching a PCB. **Both layouts use the same firmware - which is programmed to work with exactly these circuits !!**. Nevertheless details of the peripheral circuits may be modified due to the user's need .

In most cases only the DMX OUT function is used ("DMX Generator"). Though the DMX standard does not recommend a galvanically isolated DMX OUT, some users prefer this. A galvanically isolated DMX transmitter particularly makes sense, if one or more DMX transmitters without galvanically isolated DMX input are connected to the DMX bus.

By default, in the first reference design **DMX OUT is not isolated - but circuitry for optical isolation is provided**. If isolation is preferred, three printed wires - marked with a "c" or "u" symbol at the PCB bottom side - have to be cut (2.5mm drill recommended). One additional jumper wire has to be assembled,too. Optionally a simple not isolated DMX IN is provided. If the use of DMX IN is essential for your application ("DMX Control Box"), **an alternative PCB layout with optoisolated DMX input is available**, as proposed by the DMX standard. No optoisolation for DMX OUT is provided here.

Both PCB designs fit into 3 easily available cabinet models:
DIN rail compatible ("Camdenboss CNMB/4-", Reichelt order code CB HUTKIT 4),
low cost plastic standalone ("Eurobox" from Reichelt or Conrad 52 31 32) and
rugged aluminium ("Fischer Elektronik FRAME" Reichelt order code FR 80 42 100 ME).

Optical isolation of DMX OUT or DMX IN:

For this purpose, layout for an isolated DC/DC converter is provided (standard 4p SIL unregulated 1W model, available from different manufacturers).
Because the output voltage of these DC/DC converters may vary considerably due to different manufacturers, production lots and tolerance of 5V primary supply, sometimes the output voltage has to be adjusted.
For this purpose diode D3 is provided. The adjustment has to be done without installed microcontroller but with assembled IC3 and IC4 respectively IC5. With the recommended Aimtec converter and installed diode D3, the voltage between TP2 and pin 8 of IC4/IC5 usually is about 4.9 to 5.4 Volt. If the voltage is below 4.9 volt, replace D3 with a jumper wire. In cases of very high voltage (isolated voltage with D3 installed >5,4V), a silicon diode like 1N4007 should be used instead of the Schottky 1N5819. At practical DMX operation, the isolated voltage is about 0.1 to 0.2 Volt lower than measured during adjustment.

The power supply is designed for unregulated or regulated **DC supplies with output voltages 8 to 28 Volt** and min. 4 Watt output capability.

The DC input CN1 is designed for a concentric power connector, sleeve 5.0 – 5.5mm, pin(hole) 2.1mm.
The **positive polarity** has to be connected with the **inner (pin) contact**!
As an alternative, layout for a 2 terminal clamp is provided.
The reference circuit is **protected against wrong polarity**: If wrong, the device is not powered.

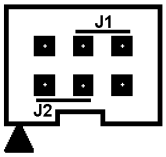
Primarily IC1 converts the input voltage into a regulated 5V supply. To provide an easy supply for external appliances like Ethernet or Wifi converters to RS-232, the 5V supply may be taken from clamp block CN2 (max. 400mA).

Special Parts List:

All resistors are carbon or metal film types, min 0.25W. Precision tolerance is uncritical.
All capacitors should be multilayer ceramic, 5,08mm raster, 50V - except C2, C3: RM2.54mm
Electrolytic capacitor E1 should be a 35V type with 2.54 mm raster
IC1: least expensive P-78E5.0-0.5 (Recom, source RS Components),TSR-1-2450 (Traco Power, source Conrad or Reichelt) or P-785.0-0.5 (Recom)
IC2: ATmega1284P PU, 40p DIL case. Source Reichelt, RS-Components
IC3: 6N137,8pDIL
IC4: MAX487, 8pDIL or equivalent, for example MAX3085, MAX485,MAX483
IC5: MAX3082, MAX3085, 8pDIL or equivalent. Should be "fail safe".
Though MAX3082 is rated for a less data rate it may well be used for DMX IN.
The slew rate limitation is only working on the transmitter
DC/DC converter: 1W, 4p SIL case, AM1S 0505SZ (Aimtec, source Reichelt) or equivalent
LED: V-L-115-WEGW (source Conrad).
Other dual LED types may be used, but then values of R1 and R2 have to be adjusted
CN1: power connector (source Conrad Art.Nr.733980, Reichelt HEBW21 or similar)
CN3: 9 pin female SubD, 7.2 mm raster ("US" type)
CN5, CN7: source Reichelt (PS 25/3G BR) or Conrad (Art.Nr.741221)

Baud rate selection:

Some jumpers may be placed (not while programming) at the Atmel programming connector to set different baud rates. After change of jumper settings a power cycle is necessary.



no jumper (default) 115.200 baud
J1 only 38400 baud
J2 only 19200 baud
J1 and J2 9600 baud

LED color: Idle green. Flashes red when an RS-232 command is received.
Flashes amber/yellow when automatic messages are sent (Control Box only)

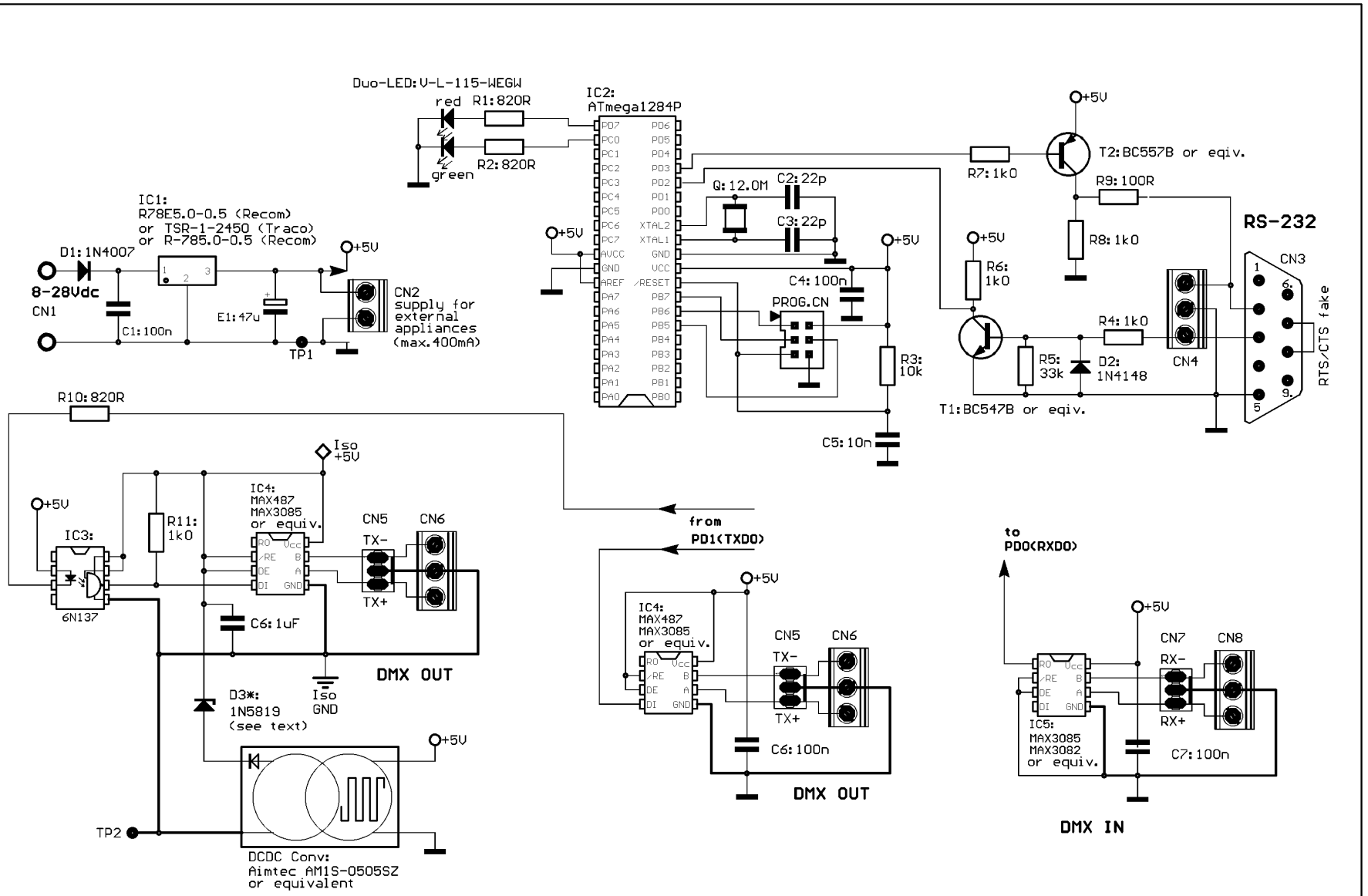
Layout of the DMX512 interface:

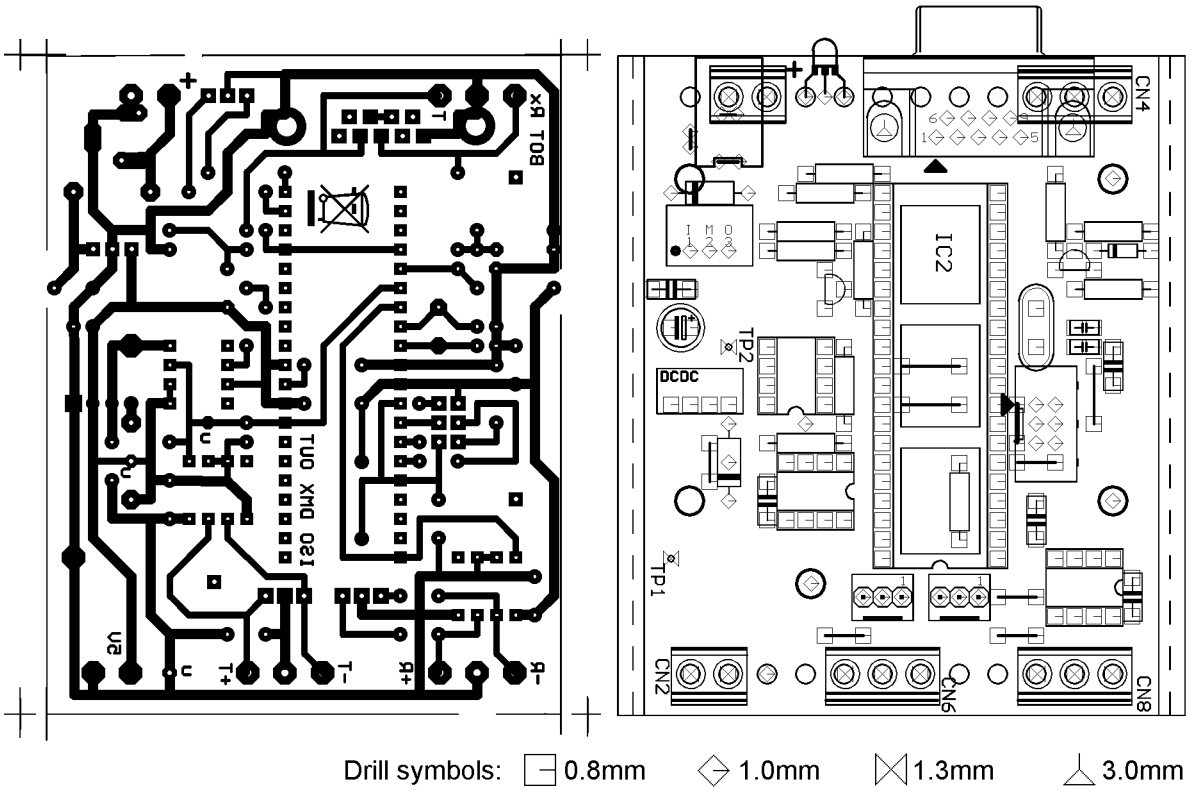
Two sets of connectors are provided for DMX IN and DMX out: one to connect XLR sockets, the other one for directly clamping the bus wires.

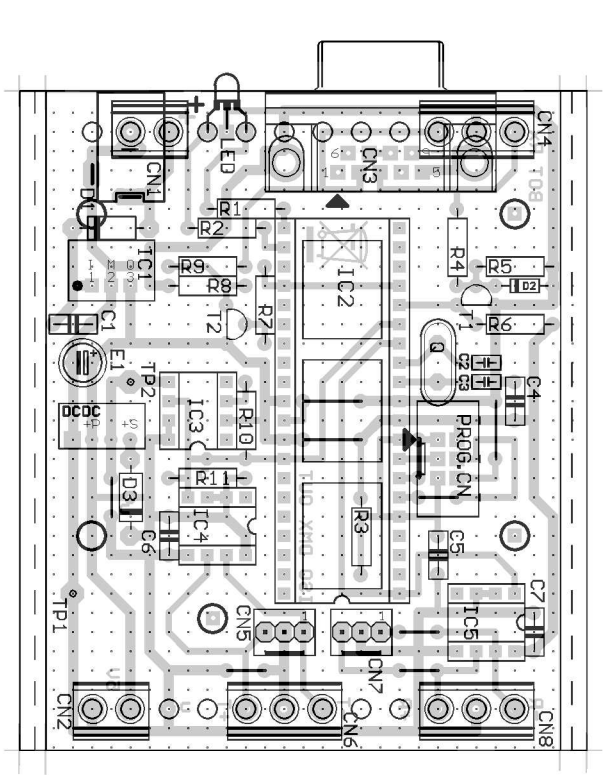
Each of the DMX connectors CN5, CN6, CN7, CN8 has the local signal ground (isolated or or not) at the middle pin. TX+,TX-,RX+ and RX- is marked at the bottom side of the PCB.

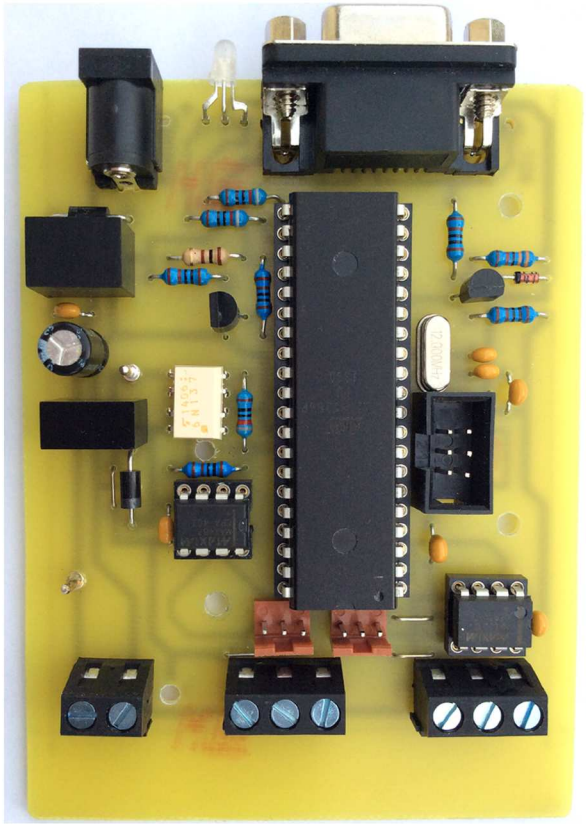
Important safety information:
The **purpose of galvanic isolation** of the DMX input and output is to eliminate DMX data errors due to moderate fault voltages which may result from long loops of signal ground lines and protective earth wires within the complete lighting installation.

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!

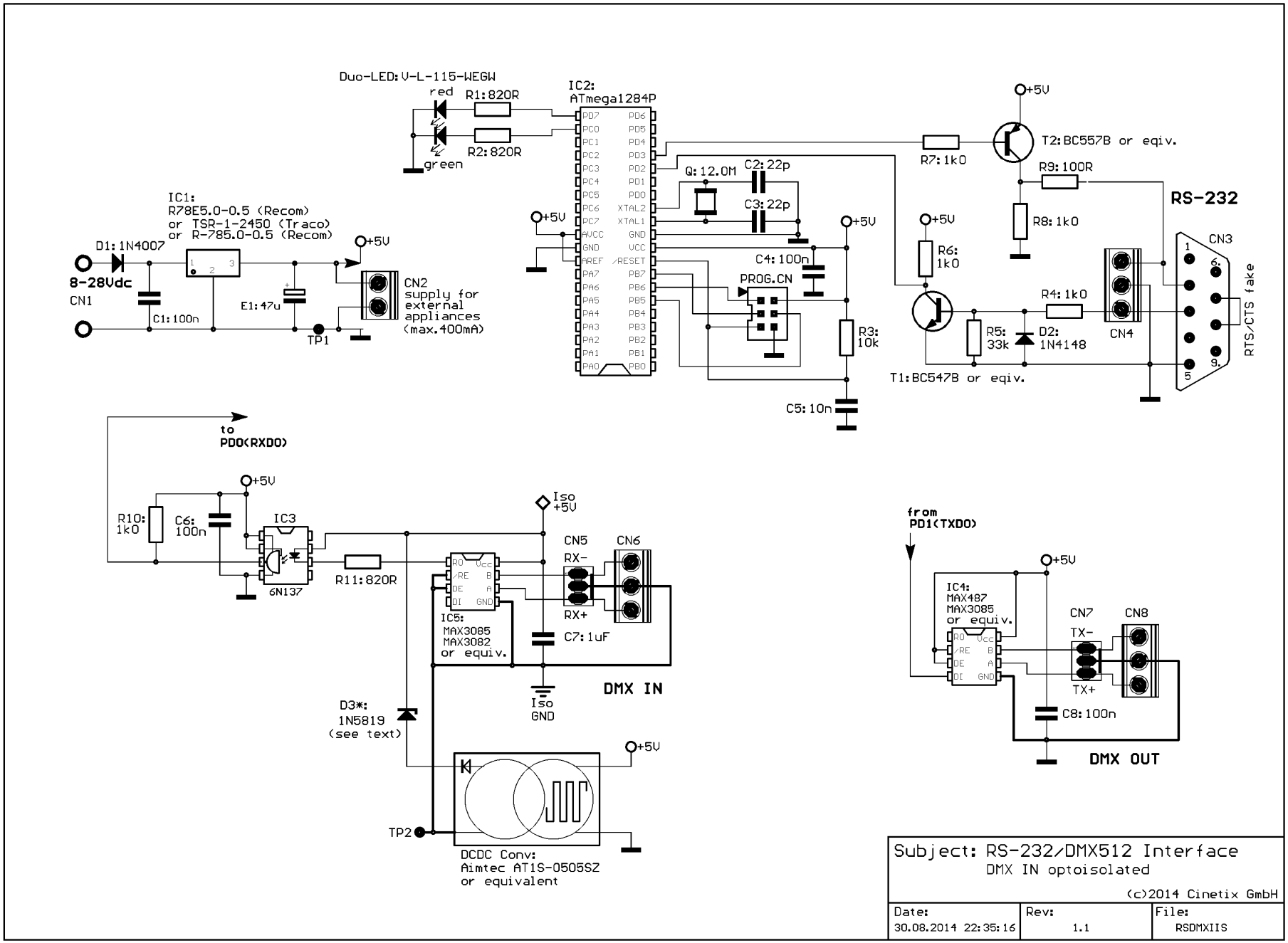

Duo-LED:V-L-115-WEGW
red
green
R1:820R
R2:820R
IC2:
ATmega1284P
IC1:
R78E5.0-0.5 (Recom)
or TSR-1-2450 (Traco)
or R-785.0-0.5 (Recom)
D1:1N4007
8-28Vdc
CN1
C1:100n
E1:47u
TP1
+5V
CN2
supply for
external
appliances
(max.400mA)
Q:12.0M
C2:22p
C3:22p
C4:100n
PROG.CN
R3:
10k
C5:10n
R7:1k0
T2:BC557B or eqiv.
R9:100R
R8:1k0
RS-232
CN3
R6:
1k0
R4:1k0
R5:
33k
D2:
1N4148
CN4
T1:BC547B or eqiv.
RTS/CTS fake
R10:820R
Iso
+5V
IC3:
6N137
R11:
1k0
IC4:
MAX487
MAX3085
or equiv.
CN5
CN6
TX-
TX+
C6:1uF
DMX OUT
D3*:
1N5819
(see text)
Iso
GND
TP2
DCDC Conv:
Aimtec AM1S-0505SZ
or equivalent
from
PD1(TXD0)
C6:100n
DMX OUT
to
PD0(RXD0)
CN7
CN8
RX-
RX+
IC5:
MAX3085
MAX3082
or equiv.
C7:100n
DMX IN
Alternative:
DMX OUT optoisolated
Basic version:
not optoisolated
Subject: RS-232/DMX512 Interface
Option: DMX OUT optoisolated
(c)2014 Cinetix GmbH
Date:
30.08.2014 22:36:42
Rev:
1.1
File:
RSDMXOIS

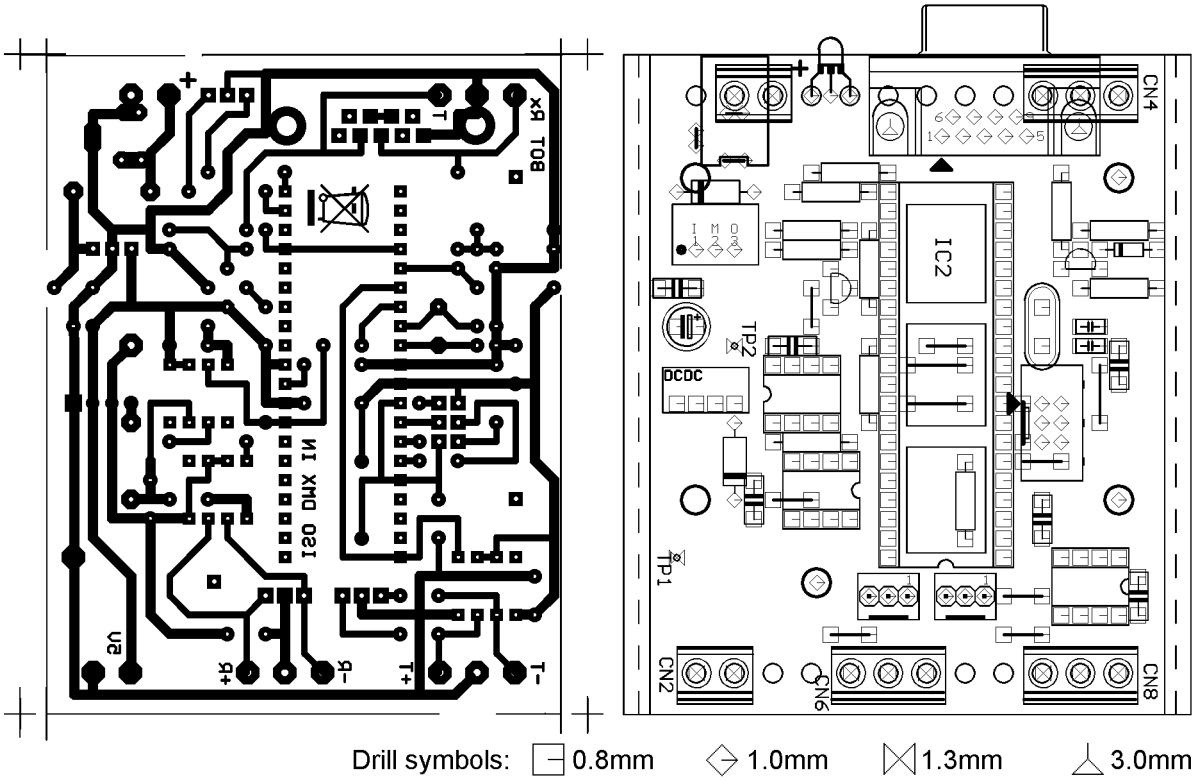


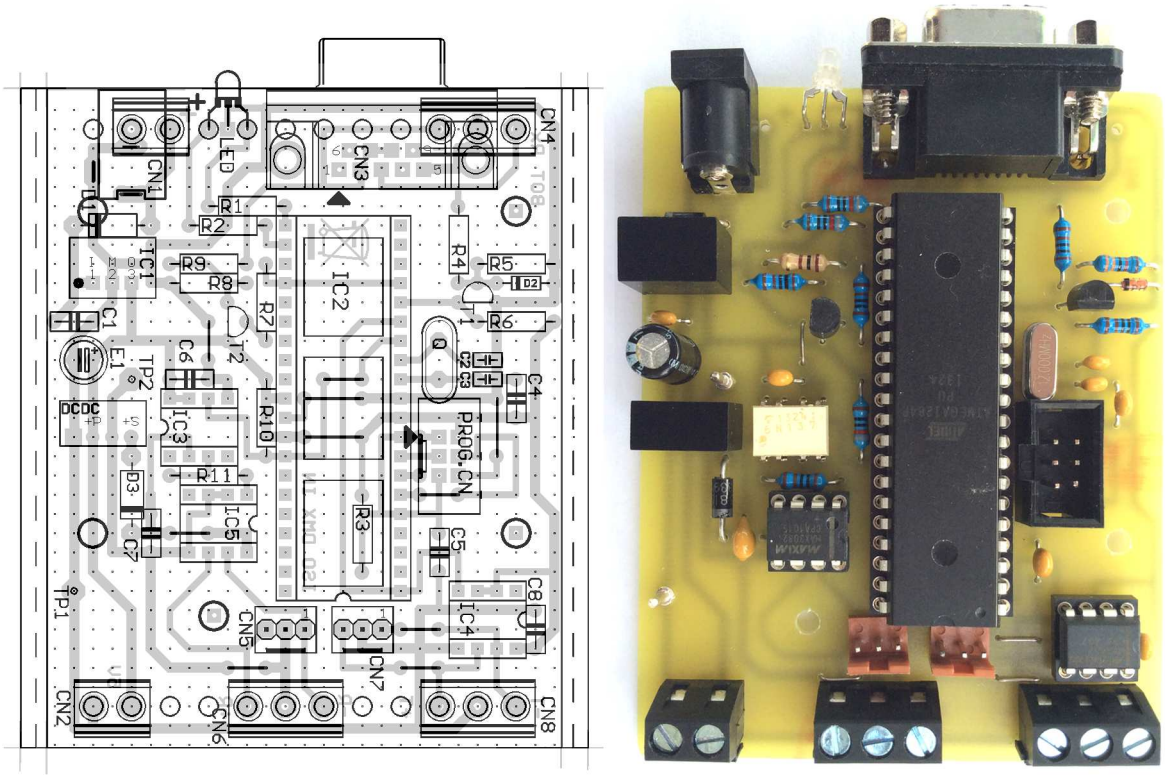




This example PCB is cut 100x75mm to fit into a Fischer Elektronik "FRAME" enclosure. The 75mm width is marked by the outer contour of the PCB layout.
Three holes 3mm diameter are provided for mounting into an "Eurocase" plastic enclosure.
Take care: **some jumper wires are below other components**, have to be assembled first!


Duo-LED: V-L-115-WEGW
red
green
R1: 820R
R2: 820R
IC2:
ATmega1284P
PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
XTAL2 XTAL1 GND VCC /RESET
PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
AVCC GND AREF
Q: 12.0M
C2: 22p
C3: 22p
C4: 100n
PROG.CN
R3: 10k
C5: 10n
+5V
IC1:
R78E5.0-0.5 (Recom)
or TSR-1-2450 (Traco)
or R-785.0-0.5 (Recom)
D1: 1N4007
8-28Vdc
CN1
C1: 100n
E1: 47u
TP1
CN2
supply for
external
appliances
(max.400mA)
R7: 1k0
T2: BC557B or eqiv.
R9: 100R
R8: 1k0
R6: 1k0
R4: 1k0
R5: 33k
D2: 1N4148
T1: BC547B or eqiv.
CN4
CN3
RS-232
RTS/CTS fake
to
PD0(RXD0)
R10: 1k0
C6: 100n
IC3
6N137
R11: 820R
IC5:
MAX3085
MAX3082
or equiv.
RO Vcc
/RE B
DE A
DI GND
Iso
+5V
CN5
RX-
RX+
CN6
C7: 1uF
Iso
GND
DMX IN
D3*:
1N5819
(see text)
TP2
DCDC Conv:
Aimtec AT1S-0505SZ
or equivalent
from
PD1(TXD0)
IC4:
MAX487
MAX3085
or equiv.
CN7
TX-
TX+
CN8
C8: 100n
DMX OUT
Subject: RS-232/DMX512 Interface
DMX IN optoisolated
(c)2014 Cinetix GmbH
Date:
30.08.2014 22:35:16
Rev:
1.1
File:
RSDMXIIS





This example PCB is cut 86.5x68mm to fit into a Camdenboss CNMB/4 DIN rail enclosure.
This size is marked by the inner contour of the PCB layout.
Three holes 3mm diameter are provided for mounting into an "Eurocase" plastic enclosure.
Take care: **some jumper wires are below other components**, have to be assembled first!

Firmware

Short reference of all ASCII commands

Control Box (with DMX IN) only:

Detailled 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.** Detailled 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 synononym 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 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 adressed 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. seleceted 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.

 # <channel number>

The **parameter** (range 1 to 512) **addresses the DMX channel**, where the same DMX level is set which was entered by any previous command.

If FADETIME 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 proces. 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 proces. 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 behavious remains active until the decimal number base is set. Because the number base is stored in presets no. 0 to 3, 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 behavious remains active until the hex number base is set. Because the number base is stored in presets no. 0 to 3, loading of a 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 individuall per DMX channel). During every new setting of hue, saturation and luminance, the stored global values ot the other color components are applied, too.

If set, the fade time gets also 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, disagreable desaturated color tones may appear. To get a perfect color transition, up tp 6 subsequent fade steps between neighboured 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.

" <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

! (no parameter)

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

---

{ ! (no parameter)

same as command '!', but only a fade process on the actually addressed DMX channel (SLOT register, see command S) **is freezed on its present level**

Poll the transmit buffer:

R <number of bytes> (command has changed Jan2016, older versions: command 'Z')

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 'X' command

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.** 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

Comment: 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. 0 to 226 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 !). Any other DXM channels 129 to 512 are not influenced by the chaser and may be operated independently. By use of the **chaser-mask (ASCII {@) individual DMX channels may be excluded from the chaser effect**.

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 laoded. This applies especially for preset no.0, which is loaded when the DMX device is started.

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

Example: if the chaser cycle is set to 4 and the the chaser start is set to 128, then presets no. 128,129,130,131 are loaded partially, then preset no. 128 again and so on

---

{ i <sequence length>

same as command 'i', but the sequence is passed only once, then the chaser stops

---

> <chaser step duration>

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

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 form <chaser start> .
Step duration 0 only virtually stops the chaser. It can be forwarded by one step with command ")".

(<chaser start>

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

Accepted start values: 0 to 226. Default at delivery:128. If the chaser would access a preset beyond 226, the sequence continues with loading preset no. 0 etc..

Comment: By use of this method, a big number of individual chaser effects may be created and run easily . 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. For **creation of appropriate lighting scenes, command ~C... is recommended**

) (no parameter)

Forward the chaser immediately (asynchronous) by one step

{@ <preset#>

Loads a preset as reference for the chaser mask.
The chaser exclusively works on DMX channels, whose mask has level 0.

Parameter: Preset# (range 0 to 226)

Comment: By this command, certain DMX channels will be excluded from the chaser function. This may be useful to handle the timing of special features of complex lamps or some lamps in general independenty from the chaser.

In addition to "full" presets, the partial presets of the types @A, @B and @C as described below can be loaded into the chaser mask. After power cycle or reset of the DMX device and after command | all channels of the chaser mask are initialized with 0. This way by default the chaser operates on all DMX channels 1-128. So, in normal case the chaser mask needs not to be considered by the user.

Like the chaser, so its mask only influences DMX channels 1 to 128. If the use of a chaser mask is intended, it is recommendable to preparate one or a few presets specially for this purpose.

In contrast to the flash pattern described subsequently, the chaser mask is located in nonpermanent memory (SRAM), so it can be modified dynamically without drawback on memory life time.

< <flash duration>

Trigger 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: 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.

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=1/0 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 acual setting of the masterfader. To explain possible differences you are referred to the description of commands H,J,+,-,^,_,T,M, (,). The byte before the slash is the global setting of "automatic messsages" and "DMX triggered messages" (reflects the 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.

~ <preset#>

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

Parameter: preset# (range 0 to 226)

Comment: The setting of LOOP and the number base for DMX levels i.e. **the "system configuration" – is only saved in presets no. 0 to 3.** Thus, in presets no. 0 to 3 preferably different system configurations are stored for quick change. **The parameter value of FADETIME is exclusively stored in preset no. 0. Because this preset is automatically loaded when the 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 from different system configurations.
The masterfader parameter is not stored in presets.

Example: ~23 saves the actual state of the DMX Control Box||Generator as preset no. 23

Special feature: <u>save transmit buffer partially</u>
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#> excusively stores DMX OUT channels 1-128 to preset ch. 1 - 128
~E<Preset#> excusively stores DMX OUT channels 129-384 to preset ch. 129- 384

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: <u>save flash pattern permanently</u>

~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.
Furter details about the flash function see command < above

@ <preset#>

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

Parameter: preset# (range 0 to 226)

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 are formatted to load the default state of the DMX Control Box||Generator. For details see command "|"

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

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

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

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

Special feature: load preset partially

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 - 384 from preset to ch. 129 - 384 DMX OUT
@F<preset#> loads DMX channels 385 -512 from preset to ch. 385- 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 feature exclusively DMX levels are loaded, in no case the system configuration.

| (no parameter)

"clear all memory": all buffers and modes of operation are reset to default
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. The chaser mask is cleared. LOOP = 512, FADETIME = 0.0.
Presets are not deleted or changed otherwise.
Control Box only:
The merger transmits exclusively from the transmit buffer.
All automatic messages are reset to default status.

' (apostrophe, no parameter)

returns the letter "r" (Generator) or "R" (Control Box), followed by a version number of two decimal places

Comment: This command may be used by application software to search automatically for the appropriate COM port where the RS-232 /DMX Generator||Control Box is connected. **This feature is explicitly used by "DMX Control" software**. Else not used during normal operation.

; (Semicolon, no parameter)

Comment: Dummy command.Terminates number input for immediate execution, no effect else.
New as of RevNum 1.6.

} <pulse length> } <pulse length>

Special command to modify the duration of the DMX reset pulse

Comment: This command is intended to solve extreme synchronisation problems with other DMX equipment -- especially if certain types of DMX lamp tend to flicker.

Following the brace a byte is entered which is composed of two hex nibbles 0 .. F, each as a text character. **The same command must be repeated immediately after with the same byte parameter**. The parameter is stored permanently independent of presets and is reloaded during every system start.

1st hex digit (high nibble): Duration of the DMX break pulse: ca. 180us base value + (nibble value * 16 us). This way, the range of the DMX break pulse may be varied between about 180us and 420 us.
2nd hex digit (low nibble): Duration of the mark pulse between DMX break and start byte: ca. 16us + nibble value. So, the setup range is about 16us to 31us.
The command }00}00 resets the DMX timing permanently to its default values.

Control Box (with DMX IN) only:

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).

If <block size> is entered = 0, the **complete receive buffer is copied** into the transmit buffer, independently of the actually addressed DMX channel (SLOT register).

Readout (poll) the receive buffer:

X <number of bytes> (command has changed Jan2016, older versions: command 'R')

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: S100X8 reads a block of 8 bytes starting from DMX channel no.100

Merge (multiplex) DMX levels received at DMX IN with user entered levels:

Throughout this manual, "merge" means switching or multplexing the source of the data transmitted at DMX OUT channel by channel between transmit buffer andi 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 226 does not affect the merge method.

U (no parameter)

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

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

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.

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.

---

o (no parameter)

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

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

P (no parameter)

Transmit from DMX OUT either transmit buffer or receive buffer which has changed last at the DMX channel addressed by SLOT ("last takes precedence")

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".
For any DMX channel, the level of the signal received at DMX-IN **has to be changed at least by 3**, before this signal gets "precedence". This is neccessary to avoid that small signal jitter which is typical for analog potentiometers permanently claims precedence to an external lighting console. This does **not** apply to commands given via RS-232, from here every change of DMX level gains precedence.

Example: S99P causes transmission of of the actual value of the transmit buffer or the 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

---

Trigger data output via RS-232 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 RS-232. 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 to 56 arbitrarily **"preprogrammed" strings** can be stored nonvolatile in the DMX Control Box by the user (each max 63bytes). These are transmitted automatically after certain level changes at the DMX channels 500 to 506 via RS-232. This way external equipment may be controlled by means of a DMX bus to a limited extent.

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 to 127=hex7F), a message is sent when the current DMX value of the addressed DMX slot differs at least the threshold from the value reported or polled before.
For typical applications a good compromise between sensitivity and amount of data is a threshold of 8.

Structure of an automatically generated ASCII message:

S <slot> v [$]DMX level[%] <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.

/ <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 RS-232 interface is limited. So, this command should only be used, when changes at the DMX512 bus take place under controlled conditions. Else messages may be corrupted or lost.

Y <0 , 1 , 2 , 3>

switch messages triggered by DMX IN globally ON and OFF

Y0 automatic messages and transmission of preprogrammed strings globally **OFF**
Y1 automatic messages globally **ON**, preprogrammed strings globally **OFF** (default)
Y2 automatic messages globally **OFF**, preprogrammed strings globally **ON**
Y3 automatic messages and transmission of preprogrammed strings globally **ON**

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 and preprogrammed strings are kept stored. Only their output is suppressed. When switched on again with Y1, Y2 or Y3, the previously stored messages, trigger levels and preprogrammed strings are reactivated exactly and completely.

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

When specific data are received at DMX IN

send preprogrammed strings via RS-232

This allows messaging of user preprogrammed strings (text or binary) via RS-232, if specific DMX levels are received at DMX IN on channels 500 to 506. How to program these strings using a PC keyboard is subsequently described. Maximum length of a string is 63 bytes.

This function is quite useful, for instance, to switch beamers or other media devices with RS-232 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
hex **08 = dec 8**	00	10	20	30	40	50	60
hex **28 = dec 40**	02	12	22	32	42	52	62
hex **48 = dec 72**	04	14	24	34	44	54	64
hex **68 = dec 104**	06	16	26	36	46	56	66
hex **88 = dec 136**	08	18	28	38	48	58	68
hex **A8 = dec 168**	0A	1A	2A	3A	4A	5A	6A
hex **C8 = dec 200**	0C	1C	2C	3C	4C	5C	6C
hex **E8 = dec 232**	0E	1E	2E	3E	4E	5E	6E

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.) 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.

2.) string messaging has to be globally activated (command "Y2" or "Y3").

3.) at the specified DMX channel 500 to 506 **exactly** the specified one of the **sensitive DMX levels** - as listed in the table above - is recognized **new but stable** during 3 subsequent poll cycles. These poll cycles are not synchonized with the incoming DMX signal, i.e. the received DMX level has to be constant long enough (at least 0,1 seconds).

Any string is messaged only once. An example: if the level of DMX channel 500 is pulled from hexA8 to hexA9 and back again to hexA8, the string named 0A is not messaged again. **To release a repeated message, the corresponding DMX channel has to be set to a level above 248 (hex F8) for at least 4 DMX cycles (about 1/10 sec) or another string must be triggered at the same DMX channel before.** 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 **Y2 or Y3** and globally **switched OFF** with the command **Y0 or Y1** (programmed strings are not deleted). This setting is stored in the presets no. 0 to 3. Especially by saving preset no. 0 can be determined if this feature is active when the box is powered on. Default setting is OFF.

How to enter a string and store it:

{ \ <string name> <string> <return>

The command code is the ASCII text sequence {\ followed by the string name (2 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 "carriage return" = ASCII code 13(hexD). The "carriage return" does not become part of the string, of course.

**Example: { ** 2eHello World <return>

"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 byte value else** (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. If the backslash itself shall become part of the transmitted string, it has to be entered in the form \5c.

The backspace key (ASCII code 8) may be used to delete a part of the entered string.

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

Every typed character is echoed via RS-232. "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. For example: \40 would cause the echo @.
As soon as the string is terminated by the return key, it is stored permanently inside the Control Box. The string can be deleted or overwritten at any time (up to 5.000 repetitions). No "editing" is possible.

A string is **deleted** by entering **{\<string name> <return>** , i.e. simply hit the return key directly following 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 RS-232

: (colon) **<string name>**

The command code is a colon (changed as of Rev.Num 1.6, before it was 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 to 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.**

Operation with MiniDMX protocol (new as of version 1.8)

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.

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

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

MiniDMX packets for 256 DMX channels (type A1) as well as for 512 DMX channels (type A2) are accepted. When A1 packets are received, only DMX channels 1 to 256 are updated, channels 257 to 512 keep their actual levels.
When the RS-232 port is set to 115200 baud, reception of 256 DMX channels (one A1 type packet) takes about 25 milliseconds. This corresponds coarsely with the standard DMX512 cycle update rate.

The MiniDMX protocol is supported by a number of good PC based DMX software products like 'DMXControl' and 'Freestyler'. While such a software is active, it occupies the corresponding COM port. Parallel data input by terminal software etc. is impossible then. This way, the 'Z' command differs from the other ASCII commands and effectively constitutes its own mode of operation.

While the MiniDMX mode is active, the **LEDs signal permanent data reception,** i.e. will change color.

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

When commands are formulated as ASCII text they are entered easily with a terminal program or with a comfortably equiped application software. **Frequently however, there is a problem with application programs having a very simple script language or with industrial PLC contollers to convert calculated numbers into their ASCII text representation.** Another problem when using ASCII commands is the 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.
The binary command set uses ASCII Codes < 32(hex20) for synchronisation and differentiation from normal ASCII commands.

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.

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.** If single bytes should have gone lost, every binary transfer is finished automatically after 0.5 seconds to prevent the DMX Control Box||Generator 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 implicitely. All bytes are interpreted as raw binary even if they are "printable".

Binary commands for setting a single DMX channel (command codes 2 and 3) as well as commands for block transfer (command codes 4 and 5) start a fade process.
The binary command 15 for transfer of a complete DMX universe, however, writes directly into the transmit buffer and neutralizes any active fade process.

If a binary command was **entered wrong** or else could not be executed, a **question mark** is echoed just like in the 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||Generator:

<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. Special case: 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 Special case:header byte=0 adresses start channel 512
2nd header byte: number of subsequently transferred data bytes to be written into the transmit buffer. To send a block of 256 data bytes enter 0.
Data bytes: counted sequence of arbitrary bytes**.**

It is absolutely necessary that exactly the number of data bytes is transferred, as entered in the 2nd header byte. If too many bytes are transferred these may be interpreted as ASCII commands and cause unintended operations – a bad source of errors. **If less bytes are transferred** the DMX Control Box||Generator hangs in an endless loop which is resolved automatically after 1 second with an error message. In this case, all previously transferred bytes 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: start channel minus 256. I.e. header byte 0 addresses channel 256
header byte 255 addresses channel 511
2nd header byte: number of subsequently transferred data bytes to be written into the transmit buffer. To send 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 2nd 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 1 second. 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). All active fade processes are cancelled immediately.

<11> (hex B) **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 set.
If the parameter is equal to 255 (hexFF), 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 allowed range **0 to 226**, the **1st data byte is set = 0**.
Though it is redundant in this case, it is maintained for compatibility with other Cinetix products

2nd data byte: preset no. to be loaded 0 to 226 (hex E2)

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

<22> (hex 16) **poll a block of subsequent levels from the DMX transmit buffer in binary format**
The DMX channel no. where polling shall start can be in the range 1 to 255.
This opcode is followed by **2 additional header bytes**.

1st header byte: start channel 1 to 255 Special case:header byte=0 adresses start channel 512

2nd header byte: number of DMX channels to be polled 0 to 255.
To poll 256 channels the 2nd header byte has to be equal 0.

The **first responded byte** is a **header**, which announces the **number of data bytes** which will actually follow.
See comment at command <6>

<23> (hex 17) **poll a block of subsequent levels from the DMX transmit buffer in binary format**
The DMX channel no. where polling shall start can be in the range 256 to 511.
This opcode is followed by **2 additional header bytes**.

1st header byte: start channel minus 256. I.e. header byte 0 addresses channel 256
header byte 255 addresses channel 511

2nd header byte: number of DMX channels to be polled 0 to 255.
To poll 256 channels the 2nd command byte has to be equal 0.

Else see comment at command <6>

<14> (hex B) **poll the complete DMX transmit buffer (channels 1 to 512)**

no further header bytes

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

If automatic messages for received DMX levels are active, care has to be 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.

How to poll a block of subsequent levels from DMX IN: (Control Box only)

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

The DMX channel no. where polling shall start can be in the range 1 to 255.

This opcode is followed by **2 additional header bytes**.

1st header byte: start channel 1 to 255 Special case:header byte=0 adresses start channel 512

2nd header byte: number of DMX channels to be polled 0 to 255.
To poll 256 channels the 2nd header byte has to be equal 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 to be 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.

<7> poll a block of subsequent levels from DMX IN in raw binary format

The DMX channel no. where polling shall start can be in the range 256 to 511.

This opcode is followed by **2 additional header bytes**.

1st header byte: start channel minus 256. I.e. header byte 0 addresses channel 256
header byte 255 addresses channel 511

2nd header byte: number of DMX channels to be polled 0 to 255.
To poll 256 channels the 2nd command byte has to be equal 0.

See comment at command <6>

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

no further header bytes

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

If automatic messages for received DMX levels are active, care has to be 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.

contact: wschemmert@t-online.de