# STM32 based DMX controller with USB Interface

©2017 Wolfgang Schemmert **New features added**, Status12 October 2017

This is a DIY construction manual for a simple and compact DMX512 controller based on the **STM32F042F6** microcontroller (20 pin TSSOP).

**DMX commands and power supply are provided via USB**. **It is seen by a host PC as a MIDI interface or as a virtual COM port**. Accordingly, DMX control is possible by MIDI messages as well as by ASCII text based commands. As a special case, the **MiniDMX protocol** is implemented, which is supported by a number of DMX control software, for example "DMXControl3".

The interface is "full speed USB2.0" grade. USB class selection between MIDI and virtual COM port (USB CDC class) is made by a jumper (may be connected to a switch). Detailled description see below.

It is **NOT allowed to use this device 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 or responsibility

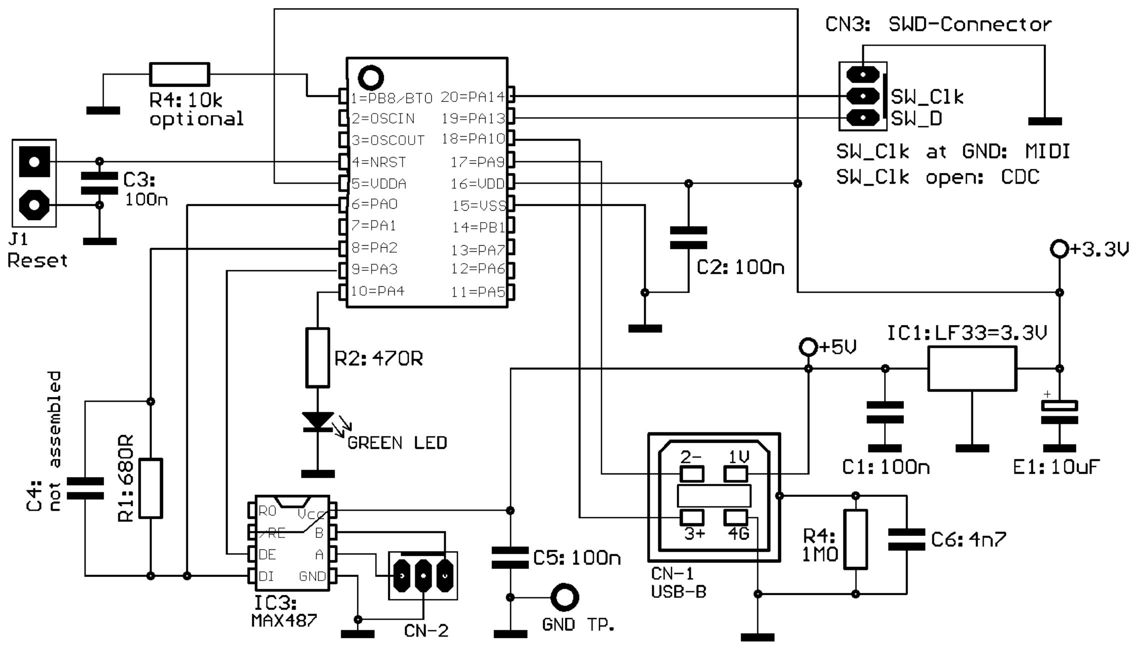
## Hardware

The microcontroller is clocked by an internal oscillator (HSI48), whose frequency is synchronized with the the USB signal. This technique provides a very stable DMX output, too.

Because it is not possible to re-start the STM32 microcontroller USART immediately after the DMX reset pulse, the USART Tx is kept running in idle state during DMX reset. It is fed via a resistor to the DMX driver MAX487, which is pulled low by another Open Drain output of the microcontroller during DMX reset.
The total supply current without DMX load is ca 20 mA, with terminated DMX bus up to 50 mA.

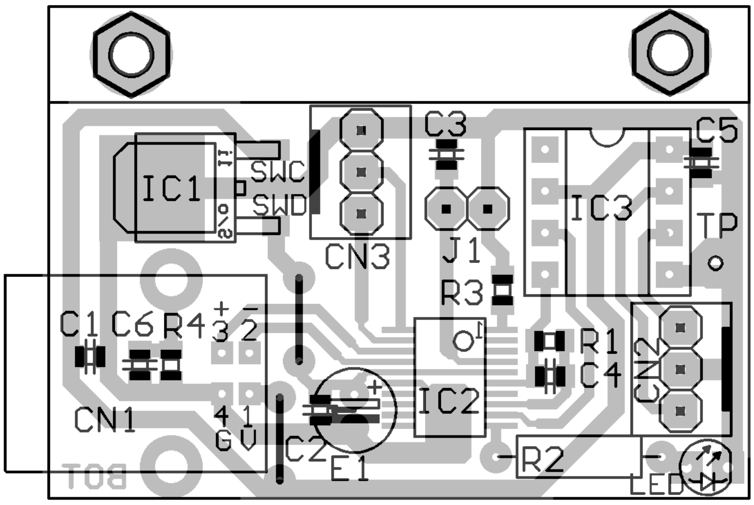
For easy reproduction with simple tools, the hardware is built on a single layer PCB with a small number of jumper wires. Thickness and spacing of the printed wires are designed for hobbyist technologies. Assembly is designed pragmatically with a mixture of through hole and SMD parts.

## Schematic diagram:



## Assembly:

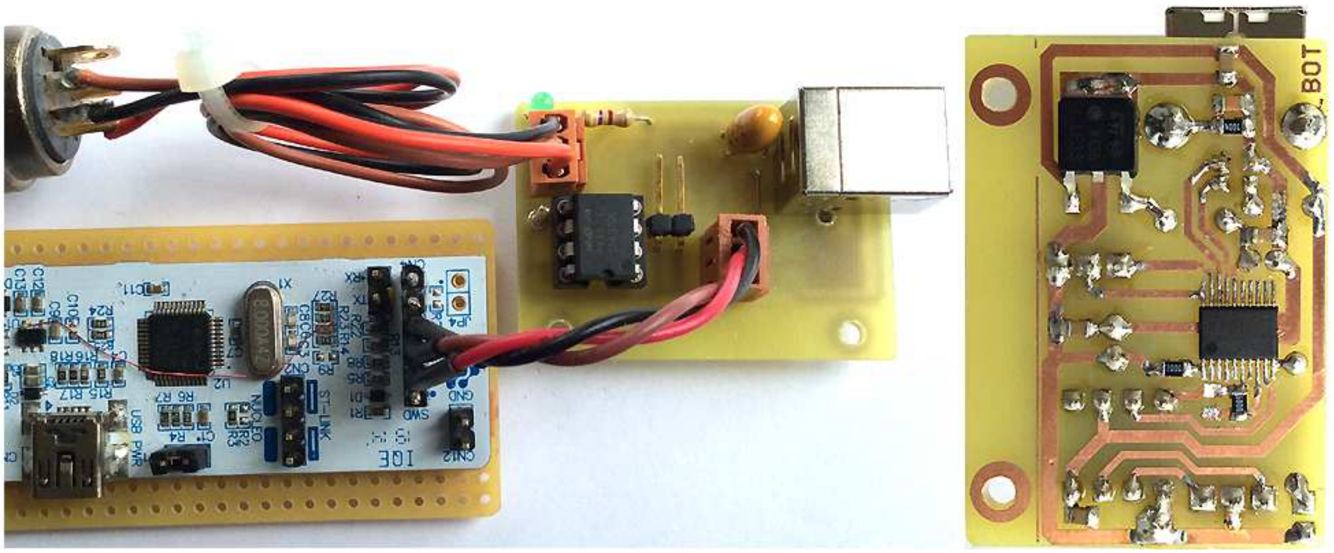
This assembly drawing is shown from the **component side**! IC1, IC2 and all SMD parts are viewed through the PCB. All SMD resistors and capacitors are size 0805.



The board size is 43x30mm. R3 is only necessary to pull the BOOT0 (PB8) pin down, usually not needed. The Reset pinhead is only necessary in seldom cases of problems with starting the SWD programmer.

## Special parts:

IC1: 3.3 V regulator: LF33 CDT, source Reichelt, Conrad 1185435
IC2: STM32F042F6P6, source: www.tme.eu/de, Mouser, RS Components, Farnell
IC3: MAX487 CPA, source Reichelt, RS Components, Farnell
Connectors CN2, CN32: source Reichelt PS 25/3G BR or Conrad 741221
LED: 3mm, low current (specified for 2mA), source Reichelt and others
E1: Tantal 10uF, min 10V. A standard Electrolytic Capacitor may be used, too



## Programming:

**Because no RS-232 port is available on this board, programming is possible only with an SWD programmer.** Instead of buying a dedicated ST-LINK module it is recommended to get a STM Nucleo-64 module, which is less expensive, can be configured as programmer and additionally be used for other experiments with STM32 microcontrollers. How to program external parts with the Nucleo, read its manual. When the programmer is cut off (i.e. used standalone, see picture above) and the programming software sends a problem like "no target voltage", connect the 3.3V output (pin next to text U1) of the regulator (5 pins) on the ST-Link part with R23 (4.7 kOhm, pad directed towards the SWD connector) to pretend a supply voltage of the programmed device.

**Build a short programming adaptor** (see picture above, wire length max 10 cm)**:**
--- connect the 2nd pin of the Nucleo SWD connector (counted from the side towards the Mini USB connector) with pin PA14 of our board (red in the photo above).
--- connect the 3rd pin with Ground of our board (brown in the photo above).
--- connect the 4th pin with pin PA13 of our board (black in the photo above).
These pins are broken out at connector CN3.
--- **start the ST-LINK software**. Click item "Connect" of the "Target" menue. After some seconds, a screen with the connection report and a listing should appear. Sometimes, but usually not, it is necessary to reset our board for a moment while you connect it with ST-LINK. When you are connected, select "Program&Verify" from the "Target" menue and upload the hex code. After programming, click item "Disconnect" of the "Target" menue and remove the ST-Link adaptor.

## Installation:

By default the USB interface works with the STM Vid/Pid, **but this is only allowed for test and evaluation inside your shack! For any public use, your individual Vid/Pid must be activated !** (how to do this see below under "Setup" page 6)

**In default state, the board is connected to a virtual COM port at the PC.** This can be observed with the Windows Device Manager. When USB is connected the first time, the driver gets installed. If you already have installed the "STM32 Virtual Com Port Driver" for the ST-Link programmer, by default our board uses the same driver and will be listed under "COM and LPT ports" as "STMicroelectronics Virtual COM Port". Else download the "STM32 Virtual Com Port Driver" from the STM website and install it. You can change the COM port number under "advanced settings" in the Device Manager. Else keep the default settings. The baud rate is not relevant in this case, any works. Handshake is not supported.

**When a connection is made between pin PA14** (SWD-Clk pin) **and Ground (e.g. place a jumper on CN3), the board is seen by the PC as a standard USB / MIDI interface.** When the USB cable is connected first, a driver gets installed automatically (Windows XP or later). In the Windows Device Manager it appears under "Audio-,Video & Game Controller" as "STM32-UsbDmx" or simply as "USB-Audio Device".
By default, the device is sensitive at MIDI Channel 1 (Status Byte low nibble = 0 ! )

For a functional test and other manual MIDI DMX operation, the simple console software "MidiTerm" is available from my website <www.midi-and-more.de>.

In ASCII as well as in MIDI mode, the DMX transmitter starts automatically with all 512 DMX channels at zero level (or with a user designed lighting scene, see ASCII command ~ page 5).

**Attention:** when the jumper is set or removed during operation, the board performs a new USB enumeration within 1-2 seconds. Unfortunately (at least under Windows) a MIDI software or Virtual COM port on the PC has to be closed then and restarted after. The software or COM port should be **closed before mode change**, else restart of this software may be blocked. In this case, another reset/enumeration of the USB device is necessary to re-enable the software.

## Modes of Operation:

## ASCII command set

is active when **PA14 (SWD-Clk pin) is NOT connected with Ground** (jumper or switch).

### Short reference of all ASCII commands

### Short reference of all ASCII commands

| | | |
|---|---|---|
| **S**n | address DMX channel for subseqent action (n=1 - 512) | p.4 |
| **V**n | set DMX level at addressed DMX channel (n=0 - 255) | p.4/5 |
| **,**n (comma) | increment DMX channel address first, there set level (n=0 - 255) | p.5 |
| **+** | increase addressed transmit buffer level DMX channel by one | p.5 |
| **-** | decrease addressed transmit buffer level DMX channel by one | p.5 |
| **=**n | set n DMX channels starting from (addressed ch.+1) to level of addressed ch. | p.5 |
| **T**n | soft Fade Time n in 1/10 seconds | p.5 |
| **R**n | read n bytes from transmit buffer starting from the addressed DMX channel | p.5 |
| **C**n | limit DMX cycle length to n DMX channels (n=24 to 512) | p.5 |
| **D** | set output of all DMX channels immediately to 0 (Panic function) | p.6 |
| ~n | save transmit buffer a as preset no. n | p.6 |
| @n | load preset Nr. n into buffers | p.6 |
| **\** | start SETUP mode | p.6 |
| MiniDMX protocol | | p.6 |
| SETUP mode | | p.7 |

## 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 always in decimal format and are sent via USB 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 to be manipulated with following commands:

#### S <DMX channel number>

The **parameter addresses a DMX channel**, on which subsequently described commands have an effect.

**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:** DMX address (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 address content will be applied to subsequently given commands.

**Example: S123** addresses DMX channel number 123

---

### Transmit buffer manipulation:

#### V <level>

**Write parameter into the transmit buffer of the actually adressed DMX channel**.

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

**Example: V34** sets the DMX level to 34 at the DMX channel which is actually addressed

---

### **,** (comma) **<level>**

**First this command increases the actually addressed DMX channek automatically, then it writes the parameter into the transmit buffer for the new DMX channel'**.

**Parameter:** level (range 0 to 255) is the value or intensity which will be immediately transmitted at the incremented and now actual DMX channel

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

---

### **+** (no parameter)

**Increase (add 1 to the) level** of the actually addressed DMX channel

**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 actually addressed DMX channel

**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.

---

### **= <block length>**

**This command writes the final level of the addressed DMX channel into the number of <block length> DMX channels starting from (addressed channel+1)**. A new fade to this final level is started from the actual level of each of these channels. 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 <block length>, channel 512 is never exceeded

---

## Additional features:

### **T < tenths of seconds>**

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

**Parameter:** FADETIME is always entered in tenths of seconds. Maximum fadetime is 12.7 seconds. This is valid for all subsequent level changes until a new T command is given. The Fade Time is stored together with preset no.0 and reloaded together with it, explicitly after power up.

**Example: T113** sets the Fade Time to 11.3 seconds

---

### **R <number of bytes>**

**Poll <number of bytes> of the DMX transmit buffer starting from the actually addressed DMX channel and send them via USB.**

**Parameter:** number of polled bytes (1 to max. 512) . After possibly the level of DMX channel no.512 is sent, the output stops.

**Syntax of the resulting state message:**

R<number of bytes> from S <1st channel no.>: V <DMX level> [**,<**DMX level>… ] <CR >

---

**C <number of DMX channels>**
**limits the DMX cycle to a specific number of DMX channels.**

Minimum is 24, maximum is 512

---

**D** (no parameter)

**Sets all DMX channels immediately to level 0. Panic function, no soft fade.**

---

**~ <preset number>**
**save the current content of the transmit buffer permanently in the microcontroller flash memory.**

**Comment:** This lighting scene is loaded automatically after switching power on or after changing the mode of operation. Default when not yet saved: all DMX channels zero level.

---

**@ <preset number>**
**Loads the permanently stored lighting scene into the DMX transmit buffer.**

---

**\ (Backslash)** (no parameter)

**opens the Setup/Configuration command interpreter.**

The next command codes and numeric values are interpreted as setup commands. The Setup/Configuration command interpreter can only be quit by command 'E'

---

## 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' (or which command ever). 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 detailled description of the MiniDMX protocol see <https://www.dzionsko.de/pmwiki.php?n=MiniDMX.Startseite>)

MiniDMX packets for 256 DMX channels (type A1), for 512 DMX channels (type A2) and for 96 DMX channels (legacy type A0) are accepted. When A1 (or A0) packets are received, the levels on DMX channels 257 (or 97) to 512 remain unchanged.

**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.

**Any software else using the virtual COM port at the PC must be closed before the lighting software is started, else the port is blocked !**

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

## SETUP mode:

**is activated during ASCII mode with command code [**

To **display a list of all command codes, type '?'** in your terminal.

**Command code "V":** enter your own USB Vid as a **4 digit hex number** (leading zeroes must be typed). Then a letter "P" appears and you enter your own USB Pid the same way as a 4 digit hex number.

By default the STM Vid/Pid is used, **but this is only allowed for test and evaluation inside your shack! For any public use, your individual Vid/Pid must be activated!**

It is stored permanently when you leave the setup with opcode "E" and can be reset to the default by entry of Vid = 0000 and Pid= 0000. **In MIDI mode the Pid is always 1 higher** than set up, else the host PC may get confused. To install the device with your own Vid/Pid, for example you could copy and rename the file "stmcdc.inf" and change the respective entries to your Vid/Pid with an ASCII text editor. Install it "manually".

**Command code "C":** insert the commonly used MIDI channel **1…16 in decimal format**. The MIDI channel inserted as low nibble into the MIDI **status byte - is always one less (0…F) !**

**Command code "!":** shows the actual setup entries.

**Command code "|":** stores the setup/configuration permanently in the microcontroller flash memory. The new MIDI base channel is active immediately. The new Vid/Pid are activated after the next power cycle or reset (when stored), not to kick you out of the active USB session.

**Command code "E": quits the setup mode back to standard ASCII mode.** If not stored permanently, the new Vid/Pid is lost. The new MIDI channel is active only during this session then.

## MIDI Channel Message protocol of the DMX512 transmitter

is active when **PA14 (SWD-Clk pin) is connected with Ground** (jumper or switch).

**By default MIDI channel no. 1 is the base channel for command input. This may be changed** in the SETUP mode. Details see above.

### Quick start and basic commands:

The most frequently used application is **lighting control with NOTE ON messages from a sequencer**.

**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.
**Exception**: 2nd data byte 127 sets the DMX level to 255
Or described in the opposite way of thinking: **to set a certain DMX level ( 0 to 255) with a simple MIDI command, HALF OF the intended DMX level has to be entered in the 2nd MIDI data byte.**.

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

| **coded MIDI channel** | **1st data byte** | **addresses DMX channel #-** | **calculation of 1st data byte** |
|---|---|---|---|
| as selectred 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 + 2 | 0 to 127 | 256 to 383 | = DMX channel minus 256 |
| as base channel + 3 | 0 to 127 | 384 to 511 | = DMX channel minus 384 |
| **as base channel** | **0 special case !** | **512** | **= 0** |

This means: on a sequencer program you have to **reserve a block of 4 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 CONTROL CHANGE commands, which allow to adjust any DMX channel 1 to 512 with full 8 bit accuracy using only the MIDI base channel**. See description of the appropriate 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 and allowed again (default state) with PROGRAM CHANGE command 120. This way you can "play in" DMX level timing with NOTE ON on a keyboard or sequencer without care about note ends.

### Survey of all MIDI Channel Commands (MIDI Implementation Chart)

**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 physically send the data byte value "0" when "program no.1" is selected and so on ! **In the table "DB" denotes the physically transferred data byte.**

| **MIDI message**<br>configured MIDI channel plus up to 3 additional MIDI channels | **special data values** | **function /effect** | **p.** |
|---|---|---|---|
| **NOTE OFF** | **DB1** = DMX channel ++<br>**DB2** = ignored | DMX level --> 0. | 9<br>12 |
| **NOTE ON**<br>MIDI channel = base ch. + up to next 3 channels | **DB1** = DMX channel ++<br>**DB2** = DMX level ./. 2 | set DMX channel and level<br>(only 1 MIDI message / 7bit resolution)<br>**DMX level = DB2 * 2** | 9<br>12 |
| **CONTROL CHANGE**<br>MIDI channel = base ch. | **DB1**= 1<br>**DB2** = 0..127 | set Fade Time 0 …12.7 sec (DB2 / 10)<br>in steps of 1/10 sec | 10 |
| | **DB1** =36 (hex24)<br>**DB2** = 1 to 10 | poll (entry of DB2) levels of DMX OUT starting from the actually addressed DMX channel | 11 |
| | **DB1**= 64 (hex40)<br>**DB2** = 0..127 | limit number of transmitted DMX channels to (DB2+1) * 4 min24 max512 | 11 |
| | **DB1**= 72 (hex480)<br>**DB2** = 0..127 | fill DB2 DMX channels starting from (addressed ch.+1) to the level of actually addressed channel | 11 |
| | **DB1**= 80–83 (hex50-53)<br>**DB2** = 0..127 | address DMX channel using only one single MIDI channel. " | 10 |
| | **DB1** = 84 ( hex 54)<br>**DB2** = 0..127 | set DMX level at addressed DMX ch. to **DB2** *2<br>adjust DMX level with 8 bit resolution to **even** value | 10 |
| | **DB1** = 85 ( hex 55)<br>**DB2** = 0..127 | set DMX level at addressed DMX ch to **DB2** *2 **+ 1**<br>adjust DMX level with 8 bit resolution to **odd** value | 10 |
| | **DB1** = 86 ( hex 56)<br>**DB2** = 0..127 | First **increase addressed DMX channel** there set DMX level to **DB2** *2**.**<br>adjust DMX level with 8 bit resolution to **even** value | 10 |
| | **DB1** = 87 ( hex 57)<br>**DB2** = 0..127 | First **increase addressed DMX channel** there set DMX level to **DB2** *2 **+ 1**<br>adjust DMX level with 8 bit resolution to **odd** value | 10 |
| | **DB1** = 96 (hex 60)<br>**DB2**= preset number | load preset 0-12 from Flash memory | 12 |
| | **DB1** = 112 (hex70)<br>**DB2** = preset number | save preset 0-12 nonvolatile in flash memory | 12 |
| | **DB1** = 119 (hex 77) | **DB2** has the same meaning as the data byte of the corresponding PROGRAM CHANGE command. | |
| **PROGRAM CHANGE**<br>MIDI channel = base ch. | DB = 8 | decrease DMX level (minus 1) at addressd channel | 11 |
| | DB = 9 | increase DMX level (plus 1) at addressed channel | 11 |
| | DB = 120 (hex 78) | Note velocity 0 is transferred to DMX level (default) | 12 |
| | DB = 121 (hex 79) | Note velocity 0 and Note Off is ignored | 12 |
| | DB = 127 (hex7F) | Panic: set DMX level of all channels to zero | 10 |

## Simple setting of DMX channel and DMX transmit level

## NOTE ON (NOTE OFF)

**The first MIDI data byte** (note value/pitch) **sets the DMX channel.**

**The 2nd data byte** (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:**

Velocity 127 sets the DMX level to 255

after PROGRAM CHANGE 121 is sent, Note OFF and NOTE ON with velocity = 0 are ignored

**BUT:** NOTE ON with velocity = 1 sets DMX level to 0

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

**To address DMX channels 128 to 512**, the USB / DMX Interface expects control commands on a higher MIDI channel corresponding with following table:

| **coded MIDI channel** | **1st data byte** | **addresses DMX channel-** | **calculation of 1st 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 + 2 | 0 to 127 | 256 to 383 | = DMX channel minus 256 |
| as base channel + 3 | 0 to 127 | 384 to 511 | = DMX channel minus 384 |
| **as base channel** | **0 special case !** | **512** | **= 0** |

---

## <u>Address the active DMX channel</u> 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 channel no.** | **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 |
| 82 (hex53) | 256 to 2383 | 0 to 127 | = DMX channel minus 256 |
| 83 (hex53) | 384 to 511 | 0 to 127 | = DMX channel minus 384 |
| **80 (hex50)** | **512** | **0 (special case!)** | **= 0** |

This command does not directly trigger any action. But the updated active DMX address register will be executed together with subsequent commands.

---

## <u>Set DMX level with 8 bit resolution at the addressed DMX channel</u> with:

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**

---

## <u>Set Fade Time</u> with:

CONTROL CHANGE

MIDI channel as position of rotary switch

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

2nd data byte = 0 – 127: fade time in 1/10second units
(setting range 0 –12.7 seconds).

The actual value of 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.

---

<u>Simple modifications of the DMX level</u> with:

PROGRAM CHANGE

MIDI channel as selected base channel
data byte = 8 decrease (subtract 1 from) the level of the addressed DMX channel
data byte = 9 increase (add 1 to) the level of the addressed DMX channel

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

data byte = 127 (hex 7F) sets all DMX channels to level 0

---

<u>Fill block of DMX channels starting from (addressed channel+1) with the final level of the actually addressed DMX channe l</u>with:

CONTROL CHANGE

MIDI channel as position of rotary switch
1st data byte = 72 (hex48)
2nd data byte: block length (1 to 127)

**Comment:** The fade duration is given by the actual setting of the Fade Time.

---

<u>Poll DMX transmit buffer at the actually addressed DMX channel</u>:

CONTROL CHANGE

MIDI channel as selected base channel
1st data byte = 72 (hex 48)
2nd data byte 1 to 10
The response comes as a sequence of CONTROL CHANGE messages:
**first CC**: actually addressed DMX channel, format as described for CC 80-83
**second CC**: 1st data byte = 1, 2nd data byte = actual Fade Time/4
**following CCs**:DMX level at this and subsequent channels,
format as described for CC 84,85

---

<u>Limit the number of transmitted DMX channels </u>with:

CONTROL CHANGE

MIDI channel as selected base channel
1st data byte = 64 (hex40)
2nd data byte = 5 … 127 (hex 7F)
The number of transmitted DMX channels (DMX cycle length) is calculated as follows:
cycle = (2nd data byte+1) * 4, i.e. the minimum cycle length is 24 DMX channels

---

Save, store preset (= actual lighting scene) with:
CONTROL CHANGE

MIDI channel as selected base channel
1st data byte = 112 (hex70)
2nd data byte = 0 … 12 ( preset number)
The actual lighting scene is stored nonvolatile in the microcontroller flash memory.

---

Load preset (= lighting scene) with:
CONTROL CHANGE

MIDI channel as selected base channel
1st data byte = 112 (hex70)
2nd data byte = 0 … 12 ( preset number)
Preset number 0 is automatically loaded after power on. Together the Fade Time and the max number of transmitted DMX channels is loaded and activated. This way the first lighting scene may be faded softly out of dark.

---

Change mode of operation with :
PROGRAM CHANGE

MIDI channel as selected base channel

data byte = 120 (hex 78**)** NOTE ON messages with 2nd data byte (velocity) = 0 are accepted and set 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 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.**
This mode is active, until it is revised with PROGRAM CHANGE 120 or device new start

**General comment on PROGRAM CHANGE messages:**

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**

---

**contact:** wschemmert@t-online.de