

# STM32 based USB to MIDI or RS-232 Interface

©2016-17 Wolfgang Schemmert      Status 06 April 2017

This is a DIY construction manual for a small USB to MIDI(or RS-232) interface based on the **STM32F042F6** microcontroller (20 pin TSSOP).

Alternatively a functionally equivalent device can be built on a Veroboard as a kind of shield (or better say "base board") for the **STM32L053-**, the **STM32F411**, the **STM32F446** and the **STM32L476 Nucleo-64 modules**. This solution is less elegant but takes less "technological skills". Detailed description see below.

Though a number of SMD parts is used, technology is held as simple as possible. The prototype board is bathroom made and freehand assembled.

The interface is "full speed USB2.0" grade. USB class selection between MIDI and RS-232 is made by a jumper near the USB socket (may be connected to a switch). The 5V supply is scaled down by a linear low drop regulator to 3.3V which supplies the complete board (total supply current about 26 mA).

The basic mode of operation is bidirectional transfer of a "transparent" MIDI message stream between USB and the MIDI connectors (respectively RS-232 byte stream). A simple combined MIDI and RS-232 port hardware is provided on the board with separately wired connectors. A dual red/green LED signals presence of power and data flow.

The middle part was originally designed as a kind of "STM32 Stamp" with all IO pins put at two rows of socket arrays (or pinheads, as user prefers). This part still can be cut out and be used as such, in a breadboard, e.g. Or some kind of "shield" can be attached there.

Supported by the provided firmware, up to 6 potentiometers and 1 pushbutton (or no potentiometers and 7 pushbuttons) can be installed there to trigger user programmable MIDI messages laid over the "transparent" data transfer.

The module can be programmed with the STM Flash Loader via the on board RS232 connector. Unfortunately the loader supports only USART1, which is not directly available because USB is mapped on these pins. A workaround is shown below. Alternatively the board can be programmed with STM-Link, which is connected to the pin quadruple next to the USB connector.

In RS-232 operation, a special configuration mode can be activated. Here, for use as USB virtual COM port baud rates 9600, 19200, 38400 , 57600, 115200, 230400 (**and new: MIDI 31250**) are user configurable. **New as of 06 April 2017 these baud rates are configurable independently when operated as USB MIDI class, too.** With the extended baudrate options, the device is useful as a data bridge between RS-232 and MIDI "worlds". Furthermore user specific USB Vid/Pid and MIDI messages to be triggered by pushbuttons or potentiometers are configurable. The user setup is stored nonvolatile in the microcontroller flash.

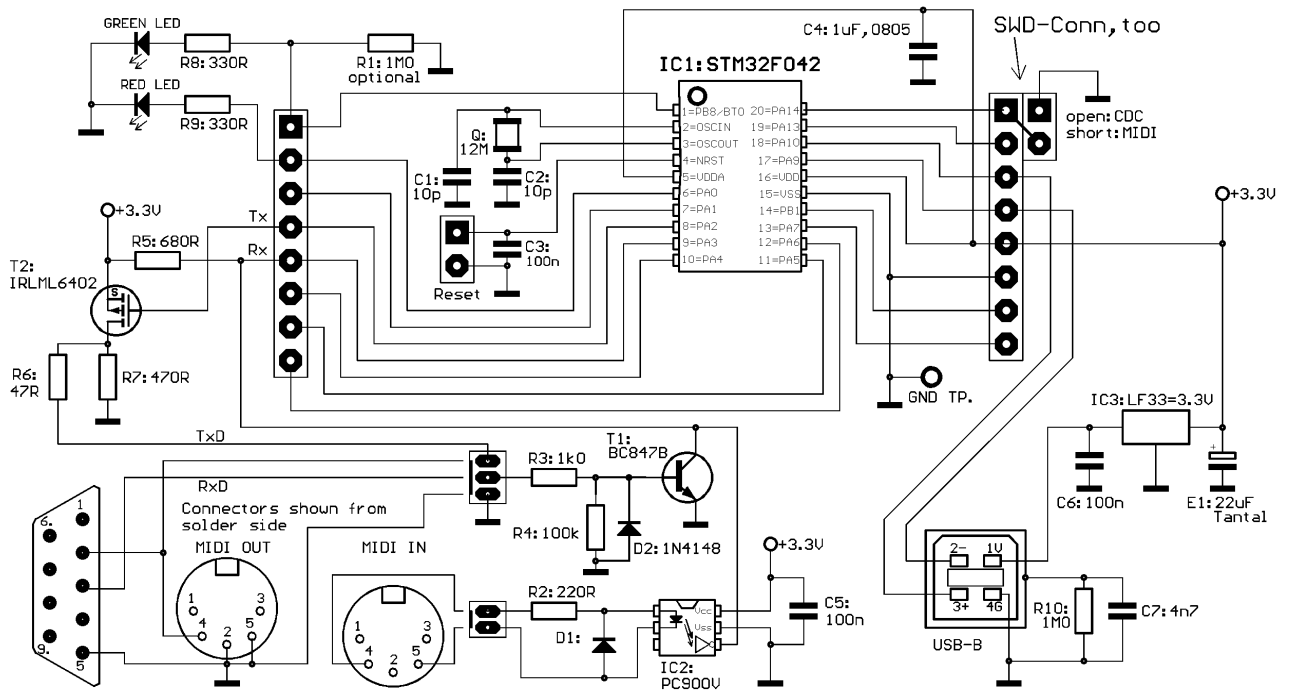
It is **NOT allowed to use these devices together with any safety critical applications**, where misfunction could result in personal injury oder noticeable material damage !

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

## Hardware for STM32F042

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 made pragmatically with a mixture of through hole and SMD parts.

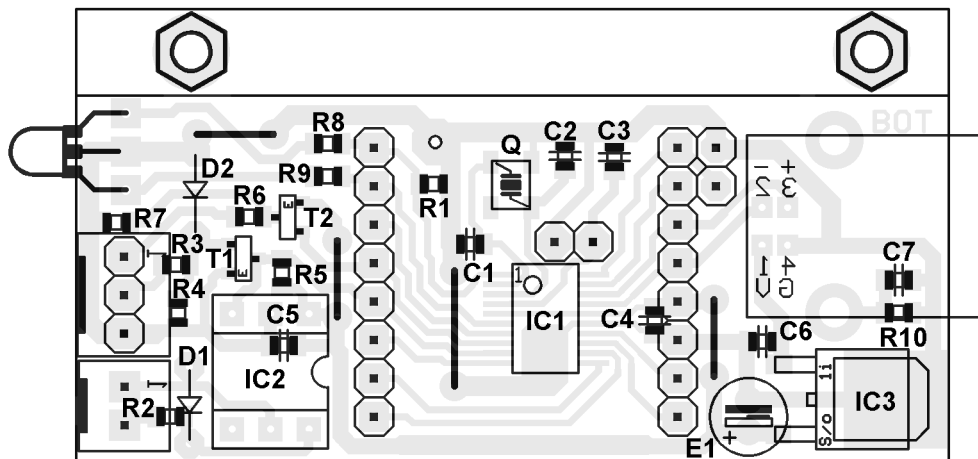
## Schematic diagram:



The circuit of the MIDI interface is completely 3.3V operated. The MOSFET output driver has a threshold voltage of much less than 3 Volt and with the given drain current a drain/source saturation voltage of about 0,1 Volt. This way, the circuit can provide min. 5 mA to a standard MIDI IN where the optocoupler has a voltage drop of max. 1.8 Volt. Most actual optocouplers have a voltage drop of 1.2 to 1.5 volt, so in most cases about 7-8 mA are driven. The 47 Ohm resistor limits the drain current in case of short to ca. 66 mA. The gate capacitance of the MOSFET is a speed limiting criterion.

## Assembly:

This assembly drawing is shown from the **solder side**!  
All SMD resistors and capacitors are size 0805.



R1 is only necessary to pull the BOOT0 (PB8) pin down if the leakage current of the LED is not sufficient. The Reset pinhead is only necessary for programming via USART.

## Special parts:

IC1: STM32F042F6P6, source: RS Components, Farnell ([www.hbe-shop.de](http://www.hbe-shop.de)), [www.tme.eu/de](http://www.tme.eu/de)  
Quartz: Euroquartz MT-12000, source: Reichelt 12,000000-MT, Conrad 1009302  
MOSFET: IRLML6402, source: Reichelt, Conrad 162833

LED: V-L-115-WEGW, source Conrad 187496. Flattened side directed to PCB for correct colours.

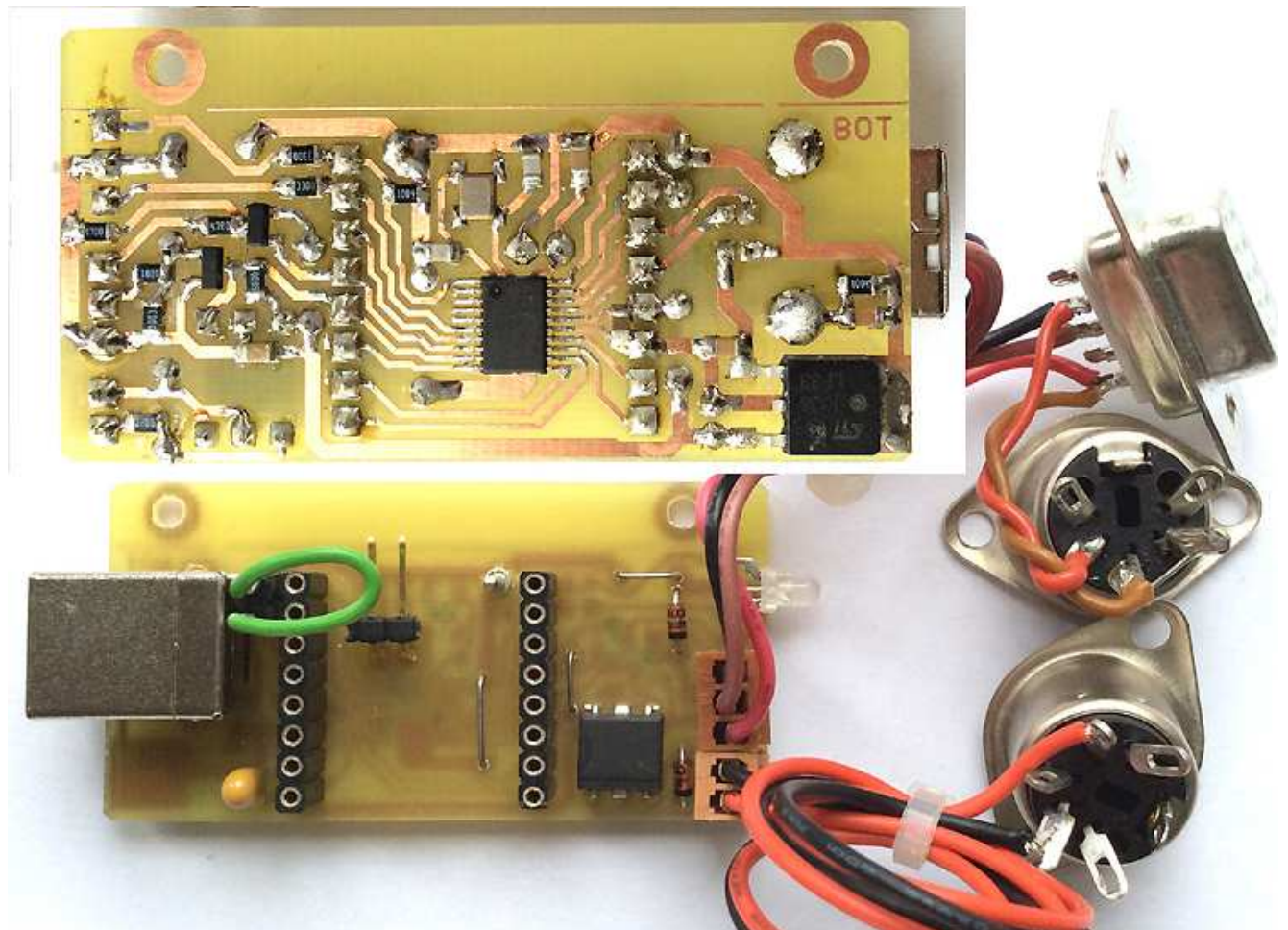
Other dual red/green LED types may be used, but then values of R8 and R9 have to be adjusted

Optocoupler: PC900V, source: Conrad 184098

3.3 V regulator: LF33 CDT, source Reichelt, Conrad 1185435

C4: Murata High Cap, source: Reichelt X7R-G0805 1,0/25

connectors for MIDI/RS-232: source Reichelt PS 25/2G BR, PS 25/3G BR



## Programming

STM provides two quite useful but different programming tools.

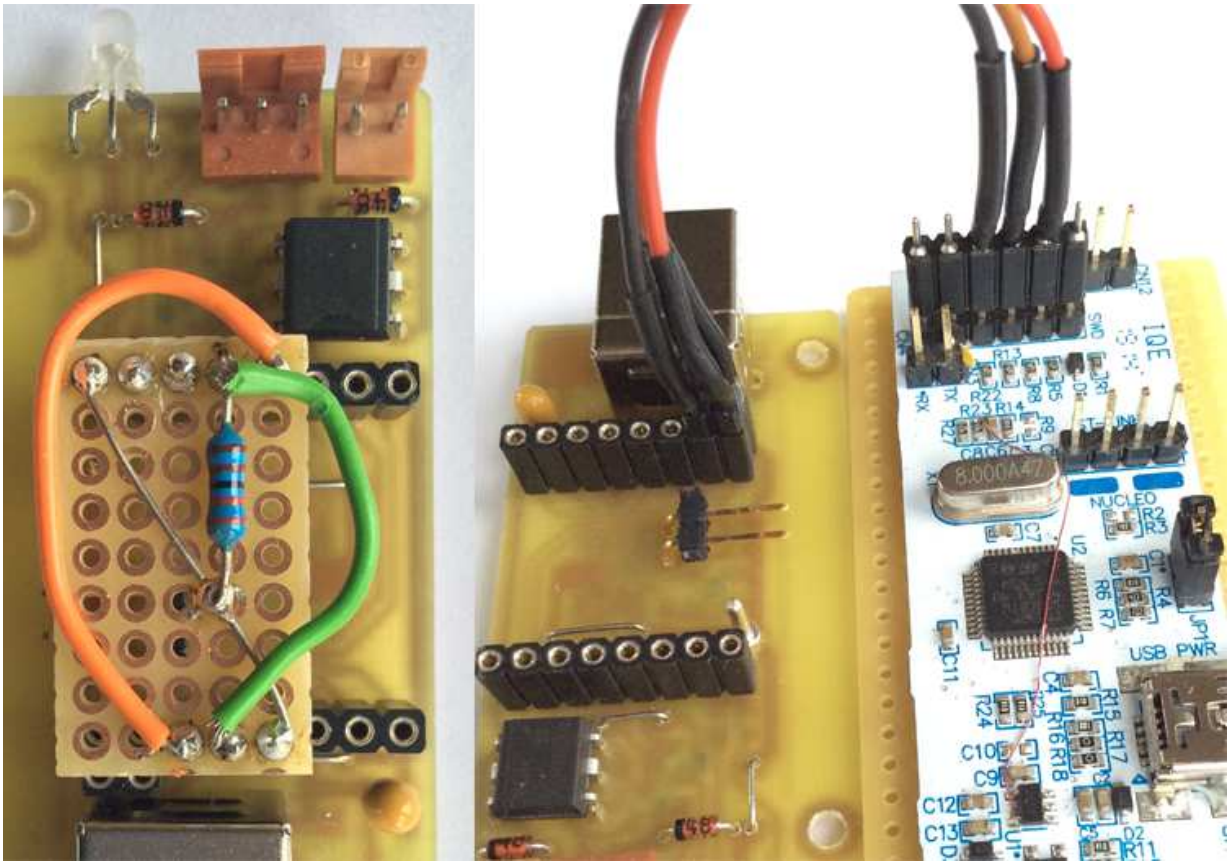
**If no special programmer is available, the module can be programmed with the "STM FlashLoader" via the on board RS232 connector.** Unfortunately the loader supports only USART1, which is not directly available because USB is mapped on these pins. The user RS-232 interface works with USART2. Following practical workaround is proposed:

First download and install the STM Flash Loader. Then you have to build a small Veroboard as follows:

--- connect pin PB8 (red LED) with +3.3Volt (solid wire in the picture).

--- connect the TX of USART1 (= Pin PA9 = USB- pin) with TX of USART2 (PA2) (green wire in the picture). This pin has to be connected to 3.3 Volt with a pull-up resistor of about 2.2 kiloOhm (exact value is uncritical)

--- connect RX of USART1 (=pin PA10= USB+ pin) with RX of USART2 (PA3) (orange wire in the picture).



Now the device can be powered with 5 Volt via the USB connector, **but it is essential, that the USB cable is rather short and there is absolute no digital signal or other electric noise on the USB data wires!** If no USB power supply is available, any d.c. supply 5V up to 9V battery can be used. Take care for correct polarity and connect it at the USB SIDE OF THE REGULATOR ! The LED on our board should light up red.

Connect the onboard RS-232 with a (real or virtual) COM port of your PC. Start the STM Flashloader software. There select the COM port number you are using and optimally 57600 Baud. Leave the other settings as proposed.

Immediately before you click "Next" at the STM Flashloader window, make a reset of our board: short the pinhead in the middle of the top PCB side (this is the only serious reason why it is provided so prominently). When you have clicked "Next" a dialog with a green lamp symbol should appear. If so, follow the hints on the screen. Sometimes the Flashloader does not detect the microcontroller correctly, shows "STM8.." or so. Then disconnect power, restart the Flashloader and repeat the action.

If connection fails, don't resignate, check everything and repeat the action carefully. Sometimes the Flashloader is headstrong.

**More comfortable programming is possible with the ST-LINK hardware and software.**

Instead of buying a dedicated ST-LINK module it is recommended to get a STM Nucleo-64 module, which 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 used standalone 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.

--- connect the 3rd pin with Ground of our board.

--- connect the 4th pin with pin PA13 of our board.

These pins are broken out close together at the socket quadruple near the USB connector.

--- start the ST-LINK software. Click item "Connect" of the "Target" menu. 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" menu and upload the hex code. After programming, remove the programming adaptor, else it will periodically cause a re-enumeration.

## Installation and Operation

**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 has to be installed. If you already have installed the "STM32 Virtual Com Port Driver", 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.

**When a connection is made between pin PA14 and Ground (short the two pins near the USB connector), the board starts as native 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 "USB-MidiCom" or sometimes simply as "USB-Audio Device".

When the jumper is set or removed during operation, the board performs a new USB enumeration within 1-2 seconds. Unfortunately, a MIDI software or Virtual COM port on the PC has to be closed then (best before mode change) and restarted after.

Without further configuration, the local serial I/O works as a RS-232 interface, data format 8N1 (default 115200 baud, independent of the setting at your PC) or as a native MIDI interface (with MIDI baudrate).

**To get into the user setup mode** during RS-232 operation, type three times a plus "+++" within max 1 second and one second no input before and after (as known from legacy modems). A prompt "CMD>" appears. Here you can enter some command- or "opcode" characters followed by numeric input as described below. All input is case independent. Wrong input is refused with a question mark "?". Then simply repeat the command. Else the interface is held quite simple, Backspace is not supported. You can leave the setup mode with opcode "E". Then (and only then) your setup is stored nonvolatile in the microcontroller flash memory.

**opcode "B":** enter the desired serial baud rate for the serial connectors with its two leading characters: 96, 19, 31(=MIDI), 38, 57, 11 or 23. The rest is inserted automatically. Default is 115200 Baud. The new baudrate is not active before the next USB enumeration and only if the setup mode is quitted with "E". Baud rate 230400 is at the edge of the hardware capability and may not work well in some cases. The setting is effective when **operated as virtual COM port.**

**new: opcode "S":** same features as "B", but effective when **operated as USB-MIDI interface**. In this case, the default baud rate is MIDI.

**opcode "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. 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 with an ASCII text editor. Install it "manually".

**opcodes "1,2,3,4,5,6 or 7":** configure which MIDI message shall be sent when the corresponding microcontroller pin is connected to Ground (pushbutton) or, if configured as analog input, when the voltage changes significantly (potentiometer between 3.3V and Ground).

First, letter "S" is prompted. Then enter the **MIDI status byte** you want to be sent in **2 digit hex format**. If you enter only the first nibble, 8,9,A,B,C,D,E plus carriage return or with leading zero, a preconfigured constant MIDI channel is inserted (see opcode M). This way the MIDI channel can be changed easily for all messages with a single command. If you have entered F6,F8,FA,FB,FC,FE,FF, the input is complete now. Undefined MIDI messages (F4,F5,F9,FD) and SysEx messages (F0...F7) are not supported and prompted with "?". If you have entered 8x ,9x, Ax, Bx, Ex or F2 (3 byte MIDI messages) next the letter "N" is prompted and you enter the first MIDI data byte (**note pitch, controller number**) as a **2 digit hex number**. Finally the letter "V" is prompted and you enter the second MIDI data byte to be sent (or the only one for 2 byte MIDI messages) as a **2 digit hex number**. **If the corresponding input shall be driven by a potentiometer, enter "80" as second data byte.** For MIDI Pitch Wheel Change (status Ex), you can enter as special case "81", too. In latter case, the potentiometer is evaluated in 12 bit format, whereas when entered "80", the potentiometer is always evaluated in 7 bit format. If a Note On is configured, the corresponding Note Off is sent when the button is released. The configured MIDI messages are saved permanently when you leave the setup with "E". The message can be deleted (flash reset to empty state) when you enter S00.

**The opcode numbers are assigned as follows:**

**1** corresponds with PA1, **2** with PA4, **3** with PA5, **4** with PA6, **5** with PA7, **6** with PB1 and **7** corresponds with PA13. The latter is the only one which has no analog capability and can exclusively be triggered by a pushbutton.

When the board runs in RS-232 mode, the triggered MIDI messages are sent in ASCII hex format (if activated). Can be immediately observed in setup mode, too.

**opcode "M":** 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)**.

**opcode (letter)"o":** parameter "1" activates trigger of user defined MIDI messages globally, parameter "0" (zero) suppresses them.

**opcode "?:** shows the actual content of the setup memory as a block of raw hex words.

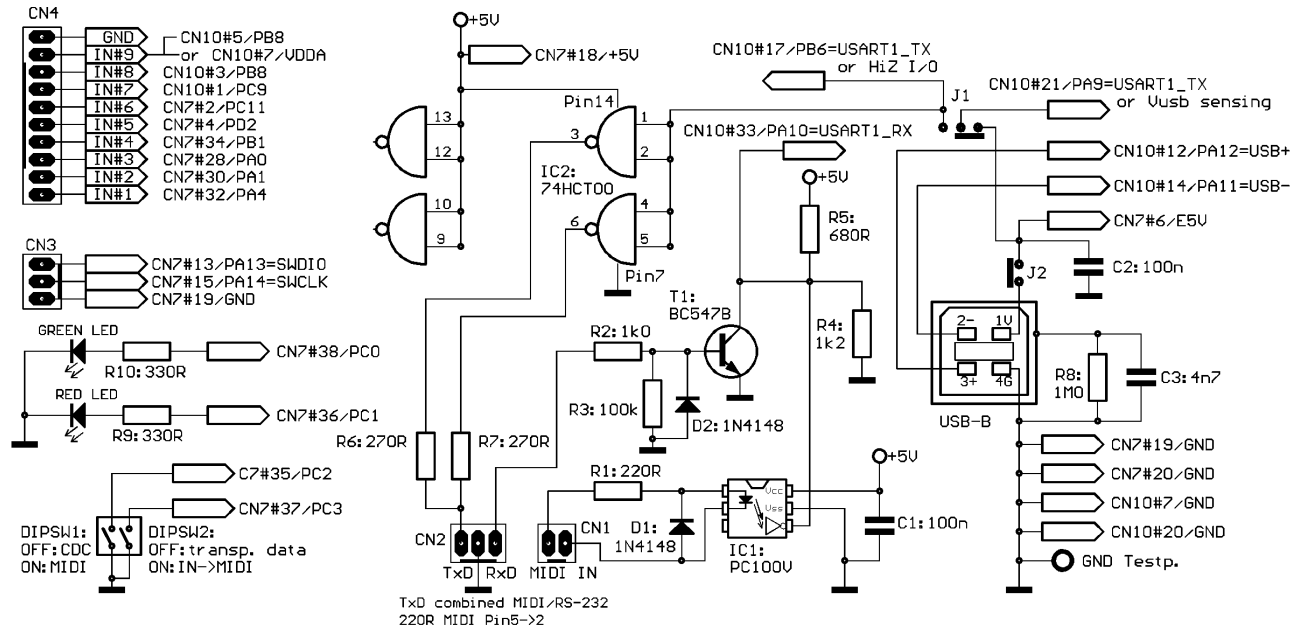
**opcode "|":** resets all user setup to "factory" default. Must be typed twice with >1 second delay!

**opcode "E":** quits the setup mode back to "transparent" mode. **Only in this case the new setup is stored permanently.** Older setup remains unchanged in any case.

# Hardware for STM32L053-, STM32F411-, STM32F446-, STM32L467- Nucleo-64

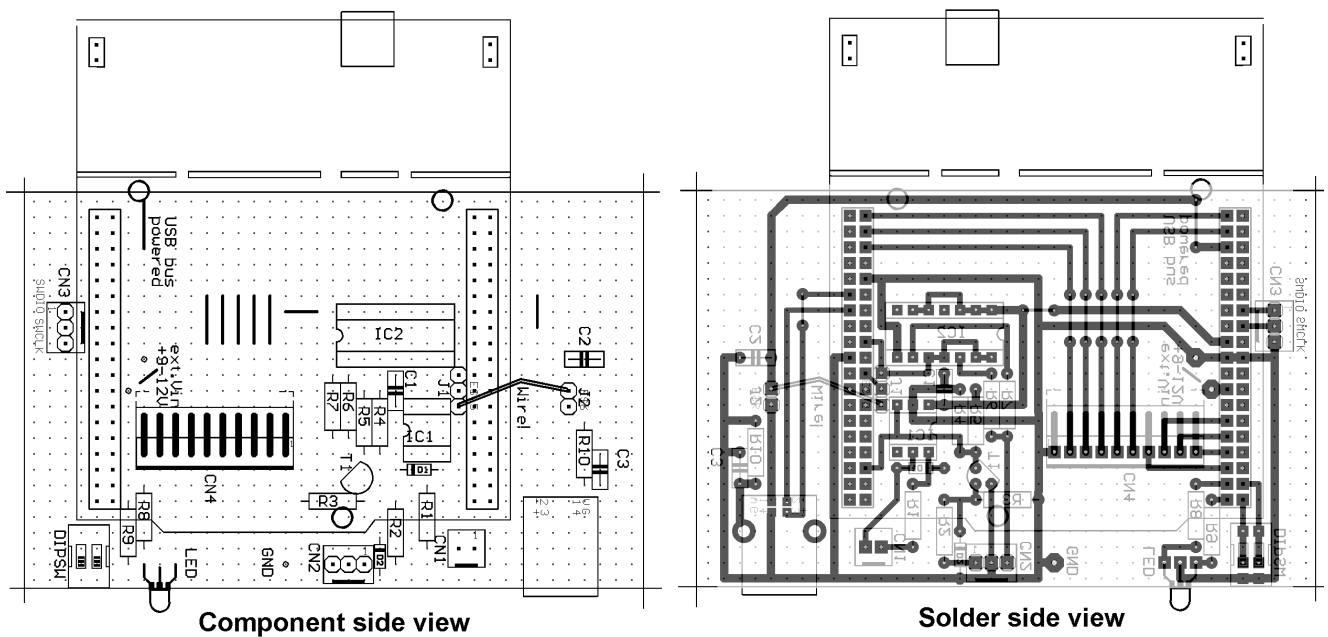
The Veroboard hardware, programming (firmware differs) and operation is the same for all Nucleo versions, but jumper J1 has to be set specifically for "HW Version1" (**STM32L053 no Vusb sensing**) or "HW Version2" (Cortex M4 models perform Vusb sensing).

## Schematic diagram:



## Assembly:

Because etching of PCBs is critical without experience and some tools (especially exposure does not work well with household lamps), a Veroboard design is proposed here. **It is intended to be mounted below the Nucleo board** instead of in "shield" position.



Please note that the assembly diagram is drawn mirrored (solder side view) for easier wiring.

J1 sets the wiring for Vusb sensing (sensing OFF = HW version 1-> towards IC2, sensing ON = HW version 2 -> towards IC1)

J2 is only used for measurement of supply current. For longterm practical use it should be shorted with solder.

If you only need a USB to MIDI or RS-232 converter, it is not necessary to assemble the MIDI trigger wiring

For IC2 it is essential to use a 74HCT00 instead of a 74HC00, because the HCT has a lower input threshold voltage and such works better as voltage level converter.

Take care to assemble CN3 (SWD programming connector) a little bit tilt. Else it would be difficult to connect the programmer there.

The ST-LINK programmer is already provided on the Nucleo board. For practical use and to save supply current, it may be desirable to cut off the ST-LINK part (see programmer picture above). For details see the Nucleo manual. Then the programmer module can be used standalone for external programming as described above together with the 3 pin connector provided on the Veroboard.

### Special parts:

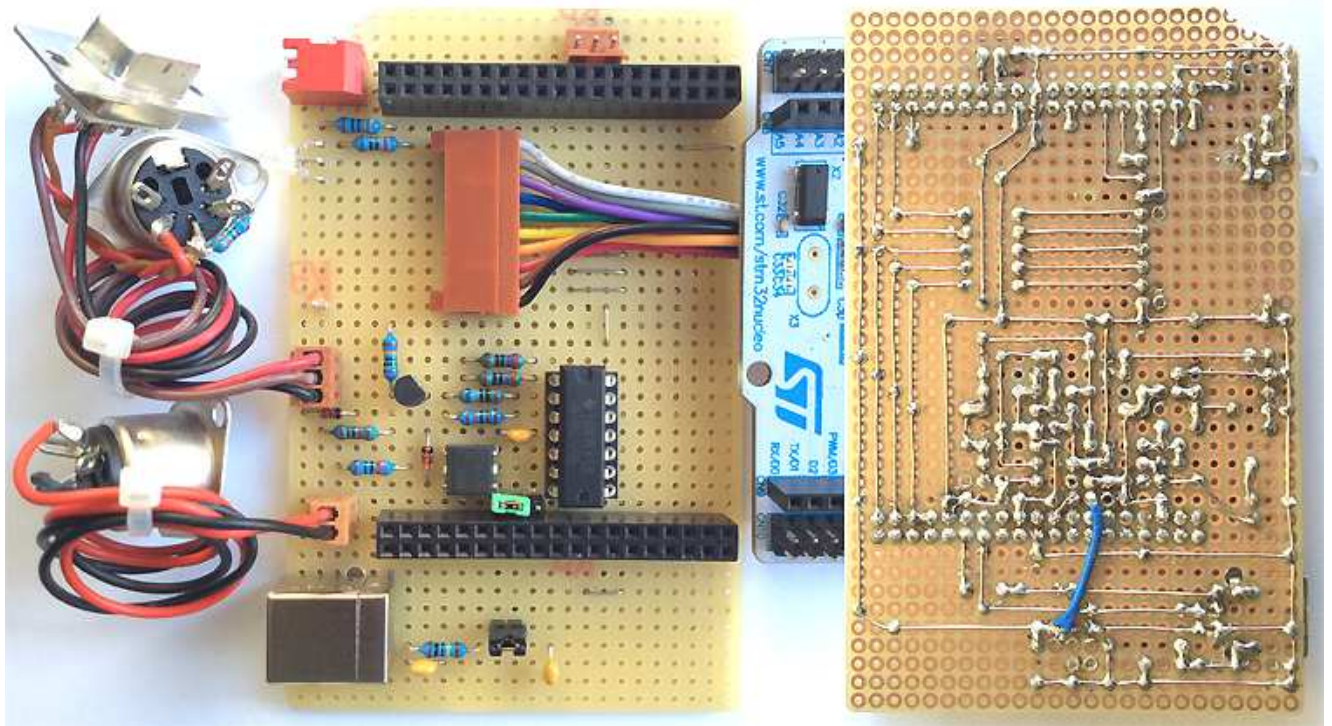
STM32xxxx Nucleo-64 board, source: Farnell ([www.hbe-shop.de](http://www.hbe-shop.de)), Reichelt, [www.tme.eu/de](http://www.tme.eu/de), RS-Components  
LED: V-L-115-WEGW, source Conrad 187496. Flattened side directed to PCB for correct colours.

Other dual red/green LED types may be used, but then values of R9 and R10 have to be adjusted

Optocoupler: PC900V, source: Conrad 184098

socket arrays for Nucleo: source: Reichelt MPE 094-2-50, 2 pcs, have to be shortened manually

connectors for MIDI/RS-232/pushbuttons: source Reichelt PS 25/2G BR, PS 25/3G BR, PS25/10W BR



The USB +5 V is connected with a (blue) wire to jumper J1, because it is impossible to cross it "veroboard style" like over the Nucleo socket. The green jumper is in Vusb sensing position. After a lot of testing and some modifications, the board already looks a little bit damaged.



## Installation and operation:

Is quite similar as described above for the small PCB. **The DIP switch at the outer left board edge selects between RS-232 (open) and MIDI(closed).**

**Trigger of MIDI messages is switched ON and OFF with the other DIP switch.** The messages are laid over the "transparent" data traffic. When the RS-232 mode is active, the MIDI messages are sent as ASCII text.

**The setup mode** is entered the same way with three +++ and the commands or "opcodes" are the same as described above. The **setup is terminated and saved** (exclusively then) when the "E" command is given.

### **The opcode characters are assigned as follows:**

1 corresponds with PA4,

2 with PA0,

3 with PA1,

4 with PB0. Only these have analog capability.

Opcode 5 corresponds with PD2,

6 with PC11,

7 with PC9,

8 with PB8,

9 with PB9. These have no analog capability and can exclusively be triggered by a pushbutton.

A 10 pin connector is provided on the Veroboard to connect these pins with a user designed control panel.

Finally with **opcode "U"** a MIDI message may be assigned to the blue "User Button" on the Nucleo board.

**Attention: when potentiometers shall be used for analog input, PB9 must be connected with the AVDD pin next to it. The connection itself does no harm to PB9, but NO PUSHBUTTON IS ALLOWED TO BE CONNECTED THERE !**

The supply current of all Cortex M4 versions including the ST-LINK part and inclusive some potentiometers varies between about 65 and 80 mA. Particularly each LED of the ST-LINK part uses about 10mA! When the ST-LINK part is cut off, the supply current is in the order of 32-35 mA. The STM32L053 takes about 5 mA less current.

The firmware is programmed straightforward on register level, no "libraries" are used. Especially the USB implementation is totally different for the Cortex M4 microcontrollers (USB\_OTG vs. USB\_FS for STM32F0xx,L0xx).

The STM32L053, STM32F411 and the STM32F446 use a quartz clock oscillator (HFE clock), whereas the STM32L476 is driven by a free running oscillator (MSI clock) which is locked with the 32kHz quartz crystal mounted on the Nucleo board (LSE clock) and provides very good stability.

**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