# STM32 based MIDI Merger to USB and Ethernet

 Status 15 October 2017

This is a DIY construction manual for a MIDI Merger based on the **STM32 Nucleo-L476** board which is plugged on a specific PCB.

**The essential job of the MIDI Merger is to collect/bundle the message streams** from up to 4 conventional MIDI IN connectors and transmit the combined stream via Ethernet or USB to a PC or another MIDI capable host device. The MIDI bytes from each MIDI IN are stored intermediately until a complete MIDI message is received and then the complete message is sent as compact data block to the host. This way integrity and wholeness of the different inputs is preserved independent of their asynchron timing of MIDI byte reception.

**Vice versa the MIDI message stream from the host is distributed** to up to 4 conventional MIDI OUT: For each MIDI OUT a channel filter is provided, so that only messages with user configured MIDI channels are retransmitted to a specific MIDI OUT. Because the transfer speed of USB and Ethernet is much faster than that of a conventional MIDI OUT, the effective transfer speed of each MIDI OUT is significantly increased by the filter.

Different **modes of operation** are selected by a dual contact DIP switch:

--- **both switches OFF**: setup via RS232 or USB in parallel.
--- **left switch ON**: Ethernet is merge collector/master
--- **right switch ON**: USB is merge collector/master
--- **both switches ON**: Ethernet is merge collector/master for MIDI 1 and 3
USB is merge collector/master for MIDI 4 and 5
i.e. 2 independent virtual MIDI interfaces are provided
(the naming of MIDI interfaces corresponds with the number of the STM32 USART which is used for this conventional MIDI I/O.
USART2 is not used because it has a special function on the Nucleo board)

The **USB interface** is "full speed USB2.0" grade. During setup it operates in USB CDC Class, i.e. provides a virtual COM port at the host. During MIDI transfer it is configured as USB MIDI Class Audio device.

The **wired Ethernet** interface is realized with a compact **Wiznet WIZ850io** module. By default it operates in UDP Multicast mode, which provides a very good MIDI interface on the host together with the ipMIDI driver (available from www.nerds.de). As an alternative, standard UDP can be configured by setup. TCP is not possible because it makes less sense in this case.
Additionally and more or less experimentally for evaluation of the Wiznet chip, a special configuration is available to use TCP together with specially configured MIDI4 as virtual Telnet interface. Details see below.

The **STM32 Nucleo-L476** was chosen for this project because it has several U(S)ARTs and does not need a hardware modification when the ST-LINK part is cut off.

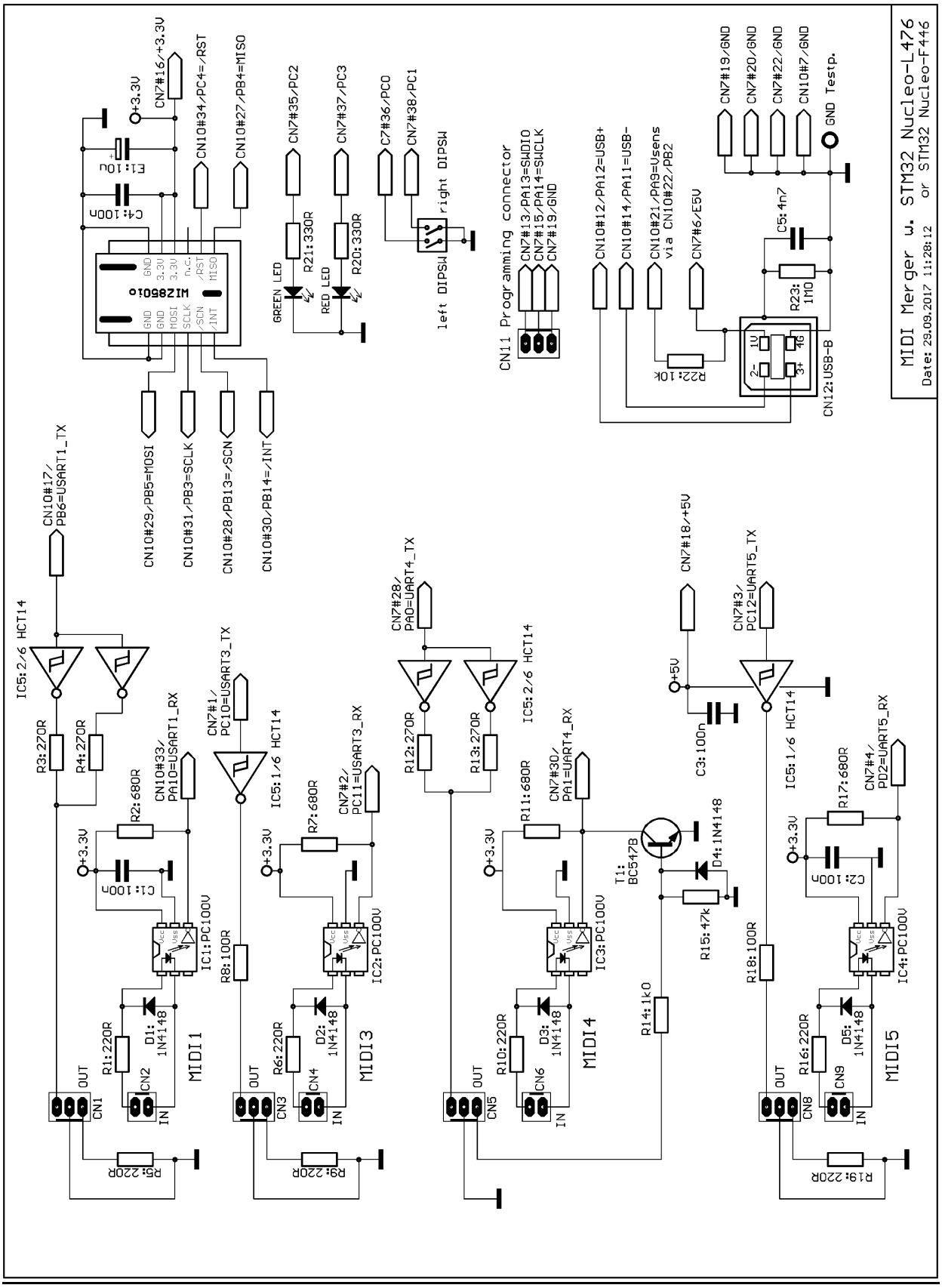
**The additional circuitry** for the conventional MIDI interfaces, the WIZ850io, USB connector, DIP switch and LED is located on a 10x8.6 cm **single layer PCB**. Its design is less elegant, but held strictly in 2.54mm raster (except USB connector). So it is possible to rebuild it on a Veroboard with round copper dots, though this would be a heavy task. For reasons of better soldering of the single layer PCB and component placement as well as better "stand alone" usefulness, it is not designed as a "shield", but as a " base board" instead. It can be used with or without the ST-LINK part of the Nucleo module.

**5 Volt power is supplied through the USB connector**, even possible by a USB power bank. USB connectivity is not necessary when Ethernet works as MIDI Merge collector/master.

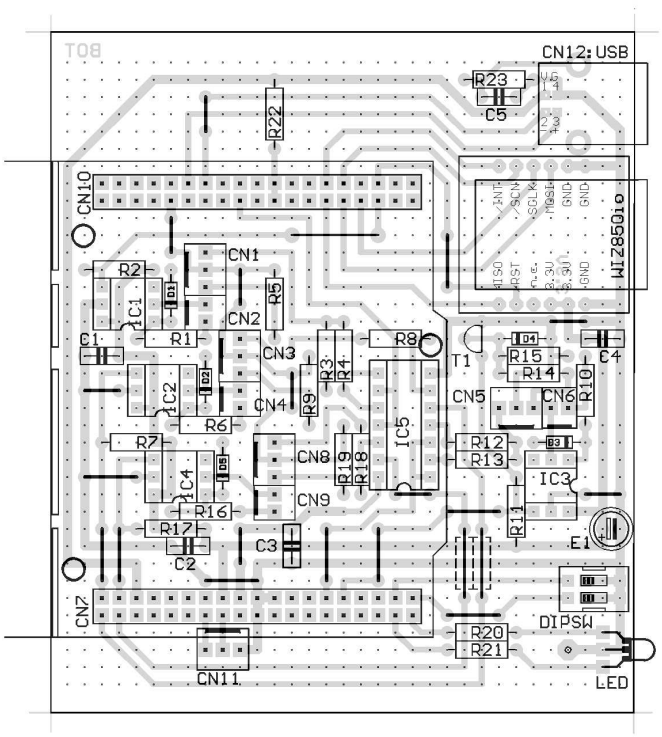
# Hardware of the additional PCB

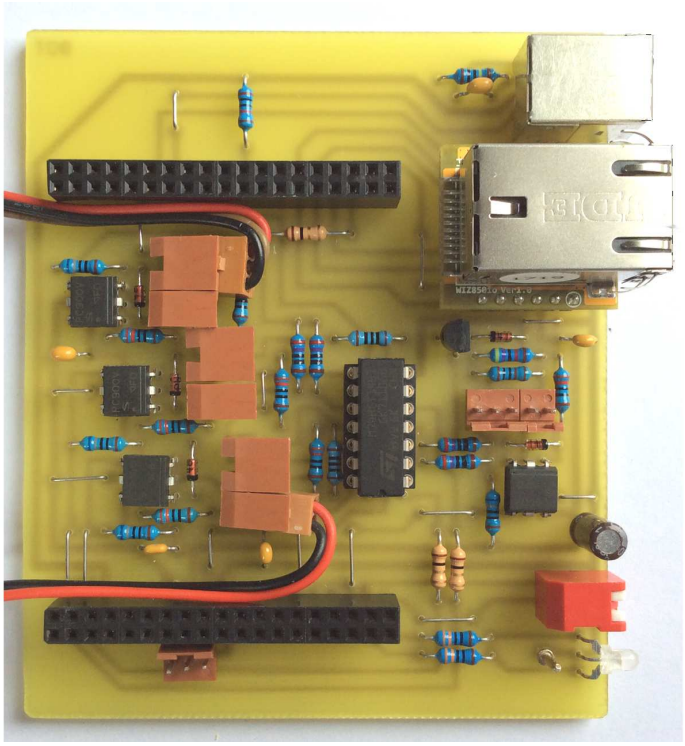
## Schematic diagram:

The connector CN7,CN10#pin numbers refer to the Nucleo board. For details see the Nucleo User Manual:



## Assembly:



Recommended drill diameters:
resistors, capacitors, diodes, ICs, jumper wires, DIPSW, USB socket = 0.8mm, else 1.0mm

For MIDI connectors below the Nucleo board angled connectors are used to keep the wires below the Nucleo board. They must be mounted slightly tilt to let the plug counterpart fit into. Alternatively, the wires can be soldered directly, of course.

MIDI4 I/O is wired somewhat different from the other ones. The second 220 Ohm resistor from pin5 of the MIDI connector to Ground has to be soldered directly on the connector. For my prototype I have connected the open end of R14 with pin 3 of the MIDI4 OUT connector. With a specially wired cable I have RS-232 access here, which was quite useful for debugging during programming work.

A 1:1 TIF file with the PCB bottom layer is available for download at my webpage <www.midi-and-more.de/midi-merger.htm>

## Special parts:

STM32 Nucleo-L476RG: sources www.funduinoshop.com, Mouser, RS-Components, Farnell
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 R20 and R21 have to be adjusted
Optocoupler: PC900V; source Conrad 184098
connectors CN7,CN10 for Nucleo: source Reichelt MPE 094-2-40, must be shortened
connetors for MIDI/RS-232: source Reichelt PS 25/2G BR, PS 25/3G BR, PS 25/2W BR, PS 25/3W BR

## Programming

**With the unmodified STM32 Nucleo the onboard ST-LINK module can be used to program the STM32L476 processor.** Follow the instructions of the Nucleo manual.

To get a practically useable MIDI Merger, it is advised to cut the ST-LINK part from the module and use it as a separate programmer. CN11 is provided as programmer connector on the PCB.

How to program external parts with the Nucleo ST-LINK module, read its manual. When the programmer is cut off and used standalone, the programming software possibly sends a problem like "no target voltage". Then 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 programmer adaptor:**
--- connect the 2nd pin of the Nucleo SWD connector (counted from the side towards the Mini USB connector) with pin PA14 of the microcontroller on the remaining Nucleo board.
--- connect the 3rd pin with Ground.
--- connect the 4th pin with pin PA13 of the microcontroller
These pins are broken out on the "base board" PCB as **connector CN11**. (PA13 leftmost at CN11, see assembly drawing above).
--- 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. In very rare cases 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.

## Configfuration

**In default state** (both DIP switches OFF)**, the MIDI Merger runs in configuration mode and the board may be connected via USB to a virtual COM port at the PC.** Success of USB connection can be observed with 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. In MIDI mode the PC installs another driver, but usually this is handled automatically

Parallel to the USB connection, MIDI4 I/O (i.e. UART4) gets configurated with 115.2 kBaud and on the PCB a transistor for RS-232 input is assembled wired-or with the MIDI optocoupler (note the different wiring of the 3 pin connector with respect to the other MIDI I/O). **So MIDI4 can be used as a simple RS-232 port if USB is not available**. This is also helpful as debug port when the firmware shall be modified.

When the USB or RS-232 connection is installed, enter '?' and you will get a list of the setup commands: (input is case independent)

```
!: Show actual Configuration
V: set USB Vid/Pid of this device (hex
E: set Ethernet MAC ID (hex)m5.m4.m3.m2.m1.m0)
G: set Gateway IP (ip3.ip2.ip1.ip0)
I: set IP of this device (ip3.ip2.ip1.ip0)
S: set Subnet Mask of this device (m3.m2.m1.m0)
M: set Multicast UDP MIDI Merge IP(ip3.ip2.ip1.ip0)
D: set MIDI(in)->Host Merge Port number (Multicast only)
H: set Host->MIDI(out) Distribution/Split Port number
U: set Std.UDP/Com Destination IP incl.Broadcast(ip3.ip2.ip1.ip0)
R: set UDP Local(Rx) Port number (std.UDP, not Multicast)
T: set UDP Destination(Tx) Port number
L: set Lan Mode: 0=UDP Multicast Merge, 1=std.UDP Merge, 2=TCP&UDP COM
P: set TCP/Com Port number (Telnet/COM port only, not for MIDI) !!
X: set TCP Disconnect Time (0 ... 65535 seconds, 0=no auto disconnect)
B: set COM4 Baudrate: first 2 digits,MIDI=31 (UDP&TCP COM + Setup only)
J: set Split/Distribute MIDI4 IN to MIDI1,3,5 OUT: 0=NO, 1=YES (new in version 1.2)
K: set Split/Distribute MIDI5 IN to MIDI1,3,4 OUT: 0=NO, 1=YES (new in version 1.2)
1: set MIDI1 OUT Filter for CH.1-16 (4 hex nibbles XXXX)
```

```
3: set MIDI3 OUT Filter for CH.1-16 (4 hex nibbles XXXX)
4: set MIDI4 OUT Filter for CH.1-16 (4 hex nibbles XXXX)
5: set MIDI5 OUT Filter for CH.1-16 (4 hex nibbles XXXX)
Y: set Ethernet OUT Filter for CH.1-16 (4 hex nibbles XXXX)
Z: set USB OUT Filter for CH.1-16 (4 hex nibbles XXXX)
|: save new setup permanently (will become active next power up !)
```

When you enter command code '!' the actual settings are displayed (i.e. when called first = default settings). The user interface is held very simple; Backspace is not supported. If a mistake happens, repeat the complete command.

To avoid unexpected communication breakdown, **the new setup is not activated before during the next power up. So new entries are lost, when you leave Setup mode without saving the new setup permanently with command '|' !!** Exception: commands J,K,1,3,4,5,Y,Z are active immediately but are stored permanently, too.

**Factory Reset:** Use the ST-LINK utility and "Erase Sectors" page 24 (address 0x0800C000)

If the command parameter L (Lan Mode) is set to 2, somewhat different behaviour is active:
--- both DIP switches OFF: setup as described above with 115.2 kBaud
--- left DIP switch. ON: serial communication Ethernet UPD ←→ UART4 as MIDI/COM port (user config. baudrate)
--- right DIP switch ON: serial communication TCP (Telnet) ←→ UART4 as MIDI/COM port (user config. baudrate)
--- both DIP switches ON: setup **with user configured baud rate**

**Please note that the default values for USB Vid, Pid and Ethernet MAC ID are allowed for device test and evaluation only! For any use else, you have to enter your own Vid, Pid and unique MAC ID !**

The default Vid/Pid is the same as installed with ST-LINK, no additional Windows driver installation is necessary.
The default unique MAC ID is provided by courtesy of Cinetix GmbH i.L.

**MIDI OUT messages are filtered** (command codes 1,2,3,4,5,X,Y). The filters are active too, when MIDI 4 or 5 are splitted ( commands J, K ).
The filters are **configured as follows** (one individual filter for every MIDI OUT) :
In the sketch below set a 1 for every MIDI channel which shall be delivered and a 0 for every channel which shall be blocked:

**|ch1 ch2 ch3 ch4|ch5 ch6 ch7 ch8|ch9 ch10 ch11 ch12|ch13 ch14 ch15 ch16|**

Then rewrite every block of 4 bits into a "hex nibble" as follows and use the 4 nibbles for setup:

0 0 0 0 = nibble 0
1 0 0 0 = nibble 1
0 1 0 0 = nibble 2
1 1 0 0 = nibble 3
0 0 1 0 = nibble 4
1 0 1 0 = nibble 5
0 1 1 0 = nibble 6
1 1 1 0 = nibble 7
0 0 0 1 = nibble 8
1 0 0 1 = nibble 9
0 1 0 1 = nibble A
1 1 0 1 = nibble B
0 0 1 1 = nibble C
1 0 1 1 = nibble D
0 1 1 1 = nibble E
1 1 1 1 = nibble F

Programmers will see, that this is the mirrored order bits are organized normally. But starting with MIDI channel 1 seems to be more plausible for practical use in musical context.

**During configuration mode, a very simple "legacy" MIDI Merger is active** in the background: MIDI1 IN and MIDI3 IN are merged to MIDI5 OUT. Vice versa: MIDI5 IN is split / distributed to MIDI1 OUT and MIDI3 OUT. No channel filters are active in this case. This feature may be helpful for example when a MIDI File Player and a Keyboard shall be connected to a synthesizer expander at the same time or to replace a simple MIDI distributor. A reverse example: keyboard and drum set are sent through the merger to a DAW and the DAW sends to a synthesizer expander at MIDI5 OUT. This way in Setup mode the DAW is bypassed - keyboard and drums are sent directly to the synthesizer.
A similar direct split/merge behaviour is obtained in main Merge mode with commands J and K.

## Operation

After setup the Merger essentially works quietly in the background.

One special, but **possibly interesting feature** was not introduced yet:
-- left DIP switch closed (Ethernet is MIDI collector/master): then USB is active as 5th MIDI I/O
-- right DIP switch closed (USB is MIDI collector/master):then Ethernet is active as 5th MIDI I/O

As long as simple MIDI channel messages and short Sysex messages up to say 50 bytes are exchanged in a speed which is generated by hand operated musical instruments or something similar, everything is expected to work without conflicts.

**About latency**: the essential task for a MIDI Merger is to file messages from different inputs in a way that every message is buffered up to be retransmitted as an entity. Corresponding with MIDI baudrate, the systemic latency is about 1/3 millisecond per byte, i.e. about 1 millisecond for buffering one MIDI channel message. The internal processing time is neglegible compared with this. The additional latency by USB or Ethernet transfer is in the order of a simple commercial MIDI interface, depending on PC speed and CPU load by other active programs. However a Sysex message of 60 bytes for example, will have a latency of ca. 20ms due to intermediate buffering of the complete message in the MIDI Merger. The opposite direction from PC to MIDI OUT is less critical, because every byte is re-transmitted immediately (except J=1 or K=1). Here the native MIDI baudrate is the bottleneck.

**Problems may arise under heavy duty situations**: Every I/O buffer has a size of 5600 bytes. Ethernet and USB can transmit much faster than conventional MIDI OUT - and no handshake is implemented for these channels. **So in extreme cases buffers may overrun or noticeable delay/latency may be observed at MIDI OUT**.

Other problems are possible when **MIDI dumps are transmitted or firmware shall be updated via MIDI**: The **maximum data block** which can be transferred under merge conditions is 1 buffer size = 5600 bytes. Bigger data blocks should be transferred correctly when no data traffic is active over the other MIDI I/O. When a .SYX dump file consists of a series of packed Sysex messages, every Sysex message is transferred as a separate block - this may lower the problem level. MIDI files cannot be transferred directly. Must be converted to a series of Sysex or other MIDI messages by means of an appropriate MIDI file player.

Due to the large number of possible combinations and the limits of nonprofessional test tools which only were available during program evaluation, a **secure behaviour under any heavy duty load could not be verified**.
**So it is absolutely recommended not to use the MIDI Merger for firmware/ software updates and transfer of precious dumps !!!**

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