NONLINEARLABS

Content

# 1 General

## 1.1 Operation and Usage

The device waits for a USB MIDI data packet to be received on *one port* and when this happens, the packet is send out on the *other port*.

This happens for both directions independently and simultaneously.

The transfer process can be monitored with the two LED indicator lights on the top, one for each port, showing incoming data and data delivery status.

The bottom plate of the MIDI Bridge is equipped with magnets so that you can attach the device to magnetically responsive surfaces, notably the steel chassis of a NonlinearLabs C15 Synthesizer.

WARNING: Those magnets have considable strengths so keep the Bridge away (> 0.5m) from mechanical watches, cathode ray displays/monitors, credit cards etc with magnet strips, audio or video magnetic tapes and recorders/players, and especially from body-implanted medical devices like pacemakers.

## 1.2 Packet Latency

The typical transfer time for the ususal short MIDI packet is about 100µs in either direction, assuming very little other loads on the two USB busses.

When a packet could be sent within *less than 300µs*, the transfer is considered REALTIME.

When a packet could be sent *within 300µs and 2ms*, the transfer is considered LATE.

When a packet could be sent only *after more than 2ms*, the transfer is considered STALE.

## 1.3 Dropped Packets

When a packet cannot be sent out in due time, the transfer is considered DROPPED and will be aborted. This may happen either when the outgoing port is not connected/ready or the host computer is currently not reading data within due time, stalling the transfer (Note: Windows will always accept MIDI data over USB and will never stall whereas on Linux and MacOs a running application that *actually reads* MIDI data is required to avoid the stalling condition).

When the outgoing port is not ready the packet is dropped immediately.

When the port is ready a first stalling condition occurs, a **timeout of 100ms** is used and the packet is dropped, for subsequent stalling packets, the timeout is reduced to **5ms**. It then takes one successful packet delivery to reset the timeout to **100ms** again.

Technical detail: Until a transfer has finished (or was aborted), receiving of further packets is blocked temporarily. There is no internal buffering, rather the transfers are in real-time, one at a time.

# 2 Indicators

Each port side has a RGB (true color) LED indicator which shows both the *port status* and the *packet status* while a packet is running through. Each port LED is referring to the *incoming* data on that port.

The LED color basically shows the latency range measured for current and/or recent packet delivery.

The LED is flashing brigther when an actual packet is running through the device.

## 2.1 Continuous Port Status Display (dim colors)

The dim color of the LED represents the current status of the port :

Pulsing blue (slowly blinking, 3s period) : port is not connected.

Pulsing cyan (slowly blinking, 3s period) : port is connected and receives USB power, but no USB communication is present.

Green : port is connected and USB communication is ready to go.

Yellow : port is connected and USB communication is ready to go, but there were LATE packets within the last two seconds.

Red : port is connected and USB communication is ready to go, but there were STALE packets within the last four seconds.

Magenta : port is connected and USB communication is ready to go, but there were DROPPED packets (with data loss) within the last six seconds.

## 2.2 Flashing Packet Status Display (bright colors)

Green : packet is running for less than **300µs** (REALTIME).

Yellow : packet is running for less than **2ms** (LATE).

Red : packet is running for more than **2ms** (STALE).

Magenta : packet had to be dropped (data loss).

NOTE: Because the actual transfer times are normally very short (< 100µs) they are lengthened for display. Still the short true transfer time is directly indicated with even brighter colors, and notably the normal green color gets brighter and more cyan'ish when very dense traffic is present. In normal MIDI operation the traffic is very sparse, though.

NOTE: As long as you see *any* LED indicator activity (constantly on or blinking) the device is powered up and consumes electrical current. Therefore, to save power, you should unplug the device from computers while those are in standby, hybernate or power-down modes but still apply supply voltage to their USB sockets.

# 3 Error Color/Blink Codes

In normal operation, including a firmware update via MIDI SysEx message, **none** of the below errors will ever occur (except for "Programming Finished")... but things might go wrong on very very seldom occasions.

These are unrecoverable errors in general, device is not operational after an incident. The device must be fully unplugged to reset and return to normal operation.

The LED indicator patterns are meant for post-mortem diagnosis, so please write down the colors and blink states should you ever run into such an error. The blink rate is very fast.

| First LED | Second LED | Meaning |
|---|---|---|
| GREEN blinking | GREEN blinking | Programming Finished Succesfully (NO ERROR) |
| WHITE | WHITE (blinking or not) | Severe Code Error (Lock-Up) *) |
| RED | RED blinking | USB Packet incorrect size |
| RED | YELLOW blinking | Unexpected USB Packet |
| YELLOW | RED blinking | SysEx Data Error |
| YELLOW | YELLOW | Waiting for SysEx End Marker |
| MAGENTA | RED blinking | Programming: Data is too large |
| MAGENTA | GREEN blinking | Programming: Data Length is zero |
| MAGENTA | BLUE blinking | Programming: Erase failed **) |
| MAGENTA | MAGENTA blinking | Programming: WritePrepare failed **) |
| MAGENTA | WHITE blinking | Programming: Write failed **) |

NOTES

*) Software Bugs as well as Broken Code -- for example from an update gone wrong -- will often, but not always, end up with the WHITE-WHITE "code error" pattern.

**) Should one of these severe failures ever happen during a firmware update, it is very likely that the device is now "bricked", containing a partial or broken code update and thus rendered inoperable and refusing to take further updates. It then should be returned to factory for servicing.

# 4 Firmware Version Identification (Blink Pattern after Power Up)

To identify the current firmware version in the device, a specific blink pattern is displayed after power has been applied via one of the USB ports:

The first LED blinking YELLOW for N times, like, say, one time --> Major Revision Number N = 1

Then, second LED blinking CYAN for K times, say, three times --> Minor Revision Number is K = 3

Effective firmware Version is **N.K** , with K displayed with two digits. For the example, Version = 1.03

# 5 Manual Firmware Update

Note that the MIDI Bridge only accepts a firmware update when *no* MIDI traffic has occured since power-up, otherwise it will simply try to deliver the MIDI data on the other port as in normal operation.

1. Fully disconnect the MIDI Bridge.

2. Connect MIDI Bridge to PC only (which port used on the MIDI Bridge does not matter).

3a. For Linux Users, using `amidi` (https://www.systutorials.com/docs/linux/man/1-amidi/)

- find hardware port ID with `amidi -l`, say it was hw:1,0,0 for example

- send SysEx with `amidi -p hw:1,0,0 -s nlmb-fw-update-VX.YZ.syx` (X.YZ replaced with actual firmware number)

3b. For Windows/Mac users:

- use an aplication like "MIDI Tools" (https://mountainutilities.eu/miditools)

- load Firmware SysEx file

- send to MIDI Bridge

If the firmware update was successful, the MIDI Bridge will show that by both LEDs blinking fast in bright green color and then will reset itself after 5 seconds, then showing the new Firmware Version.

If not, try again the full cycle (note: try using also the other port of the MIDI bridge).

4. Optional Firmware Version Check (besides the visual Firmware Version Display):

- Software like "MIDI Tools" must be restarted and then will show the new firmware version of a connected Bridge in the setup screen.

- on Linux, use the command `usb-devices | grep -C 6 -i nonlinear`

Windows hint: To remove stale entries potentionally causing wrong display of the device name, go to device manager, select "show hidden devices", then delete all "NLL-Bridge" entries. Do this while the MIDI Bridge is *not* plugged in, of course.

# 6 Hardware Port Speed Identification

Technically, both ports of the Bridge are USB2.0 compatible but only one port offers the maximum speed of 480Mpbs (“High-Speed”), the other runs at 12Mbps (“Full-Speed”). Both speeds are way beyond the data rates that will normally ever be used or needed by MIDI, though. Only when an USB bus is almost saturated by other than MIDI traffic there could be cases where one wants to connect the High-Speed port of the Bridge to that bus.

The High-Speed port side of the Bridge can by identified by the LED pattern for Firmware Identification, it is located at the side where the *first* blink pulse is seen, in *yellow* (see section *“Firmware Version Identification”*).