PRELIMINARY DOCUMENTATION
(revision1, 20150731)

DIY Network MIDI Encoder, Gateway, Processor
Free edition, v0.1

PERSONAL USE ONLY

INDEX

Introduction and development prospects

Real-time hardware platforms

A “real-time” requirements for a platform is the specification for a given system to be able to react to an event within a defined time (the deadline) strictly defined. A guarantee to meet real-time requirements can only be made if the behaviour of the operating system's scheduler is deterministic and therefore can be predicted.

In the market, several platforms are available for real-time computation. A standard example is the ARM Cortex-R is a group of 32-bit RISC ARM processor cores licensed by ARM Holdings. The cores are intended for robust real-time use, and consists of the Cortex-R4, Cortex-R5, Cortex-R7.

Another widely used platform in the marked is the x86-platform, but it does not represent the optimal development solution for real-time applications because extremely complex to handle and tune. This does not necessarily mean that it is not suitable for this purpose, but it is worth to consider the response-time of a hardware/firmware/software solution when the response times that you search for your system is very short. This is especially true if you consider that a 32-bit DSP running at 1 GHz has more real-time capability than an 8-bit microcontroller running at 16 Mhz.
The most powerful microcontrollers can nowadays compete and manage large data applications and computations with the fast memories and ports, while the less powerful microcontroller can be limited to less demanding applications requiring a relatively small amount of data and computations.

Latency/Jitter measuring tools

Different hardware platforms, or the same platforms in different hardware configurations, produce different results. For this reason, our DIY system keeps track of every event and measure the performances in order to compare it a posteriori with different hardware configurations. More accurate approaches can use an independently calibrated time controller system.

The Section Performance: Latency/Jitter measuring tools is focused on the real-time performances measurement for every black-box system, from the MIDI classics to the network systems.
For example, the M-Audio PCI Audiophile 2496 board can be used as an excellent measurement tool for the MIDI DIN5 systems. Nevertheless, it grants measure precisions below the 10us only in a real-time fine-tuned system (see results below) while in the official Windows drivers the jitter stays below 1ms.

alsa-midi-latency-test

```
Usage: ./alsa-midi-latency-test -o client:port -i client:port ...
 -o, --output=client:port       port to send events to
 -i, --input=client:port        port to receive events from
 -l, --list                     list available midi input/output ports
 -R, --realtime                 use realtime scheduling (default: no)
 -P, --priority=int             scheduling priority, use with -R (default: maximum)
 -S, --samples=# of samples     to take for the measurement (default: 10000)
 -s, --skip=# of samples        to skip at the beginning (default: 0)
 -w, --wait=ms                  time interval between measurements
 -r, --random-wait              use random interval between wait and 2*wait
 -x,                            disable printf real time display measurements,  improve time accuracy with VERY low latencies,
                                cancels the additional printf delay (<0.1ms) but to compensate with -w option to avoid CPU saturation
 -1,                            1/10^1ms display precision from nanosecond accuracy (default)
 -2,                            1/10^2ms display precision from nanosecond accuracy
 -3,                            1/10^3ms display precision from nanosecond accuracy
 -4,                            1/10^4ms display precision from nanosecond accuracy
 -5,                            1/10^5ms display precision from nanosecond accuracy
 -6,                            1/10^6ms display precision from nanosecond accuracy
 -h, --help                     this help
 -V, --version                  print current version
```

#

alsa-midi-latency-test -o 32:0 -i 32:0 -R -w 3 -r -x -4

> random interval between measurements: 3.000 .. 6.000 ms

> latency distribution:

```
…
0.972300 -0.972400 ms: 1 #
0.972400 -0.972500 ms: 3 #
0.972500 -0.972600 ms: 3 #
0.972600 -0.972700 ms: 11 #
0.972700 -0.972800 ms: 12 #
0.972800 -0.972900 ms: 14 #
0.972900 -0.973000 ms: 16 ##
0.973000 -0.973100 ms: 26 ###
0.973100 -0.973200 ms: 32 ###
0.973200 -0.973300 ms: 38 ####
0.973300 -0.973400 ms: 46 #####
0.973400 -0.973500 ms: 64 ######
0.973500 -0.973600 ms: 49 #####
0.973600 -0.973700 ms: 81 ########
0.973700 -0.973800 ms: 122 ###########
0.973800 -0.973900 ms: 144 ##############
0.973900 -0.974000 ms: 200 ####################
0.974000 -0.974100 ms: 259 #########################
0.974100 -0.974200 ms: 312 ##############################
0.974200 -0.974300 ms: 420 ########################################
0.974300 -0.974400 ms: 399 ######################################
0.974400 -0.974500 ms: 451 ###########################################
0.974500 -0.974600 ms: 414 ######################################
0.974600 -0.974700 ms: 470 ############################################
0.974700 -0.974800 ms: 494 ##############################################
0.974800 -0.974900 ms: 500 ###############################################
0.974900 -0.975000 ms: 469 ############################################
0.975000 -0.975100 ms: 479 #############################################
0.975100 -0.975200 ms: 403 ######################################
0.975200 -0.975300 ms: 454 ###########################################
0.975300 -0.975400 ms: 447 ##########################################
0.975400 -0.975500 ms: 426 ########################################
0.975500 -0.975600 ms: 419 ########################################
0.975600 -0.975700 ms: 422 ########################################
0.975700 -0.975800 ms: 405 ######################################
0.975800 -0.975900 ms: 373 ###################################
0.975900 -0.976000 ms: 305 ##############################
0.976000 -0.976100 ms: 235 ######################
0.976100 -0.976200 ms: 207 ####################
0.976200 -0.976300 ms: 148 ##############
0.976300 -0.976400 ms: 85 #########
0.976400 -0.976500 ms: 49 #####
0.976500 -0.976600 ms: 23 ##
0.976600 -0.976700 ms: 15 ##
0.976700 -0.976800 ms: 7 #
0.976800 -0.976900 ms: 4 #
0.976900 -0.977000 ms: 2 #
0.977000 -0.977100 ms: 5 #
0.977100 -0.977200 ms: 5 #
0.977200 -0.977300 ms: 1 #
0.977300 -0.977400 ms: 2 #
0.977400 -0.977500 ms: 2 #
0.977500 -0.977600 ms: 1 #
0.977600 -0.977700 ms: 3 #
0.977700 -0.977800 ms: 3 #
0.977800 -0.977900 ms: 3 #
0.977900 -0.978000 ms: 1 #
...
0.978200 -0.978300 ms: 1 #
...
0.978400 -0.978500 ms: 1 #
...
0.978600 -0.978700 ms: 4 #
...
0.978800 -0.978900 ms: 1 #
...
0.979100 -0.979200 ms: 1 #
0.979200 -0.979300 ms: 2 #
...
0.979400 -0.979500 ms: 1 #
...
0.979600 -0.979700 ms: 1 #
...
0.980100 -0.980200 ms: 1 #
...
0.981600 -0.981700 ms: 1 #
...
1.015500 -1.015600 ms: 1 #
```

← 0.972300 ms

← 0.973000 ms

← 0.974000 ms

← 0.975000 ms

← 0.976000 ms

← 0.977000 ms

← 0.978000 ms

← 1.015500 ms

> SUCCESS

best latency was 0.972271 ms
worst latency was 1.015542 ms, which is great.

The alsa-midi-latency-test can easily deal with MIDI 1.0 messages network protocols transport. The network permits you to redefine several limits even though its greater complexity does not allows you to grant the same response mentioned above for M-Audio Audiophile 2496 because of the higher complexity of the system. The only way to achieve the test of the performance is to measure the real results.

Time-Sensitive Networking (TSN) is a set of standards developed by the Time-Sensitive Networking Task Group (IEEE 802.1) to avoid these structural problems. The TSN Task Group was formed at November 2012 by renaming the existing Audio/Video Bridging Task Group and continue their work. The name changed as a result of extension of the working area of the standardization group. The standards define mechanisms for the time-sensitive transmission of data over Ethernet networks.

For example, IEEE-1733 defines the means to transport MIDI using IETF's Realtime Transport Protocol (RTP) - including RTP-MIDI - with the improved timing services of AVB. AVB enhances the quality of service and provides guaranteed latency and bandwidth for media streams over an AVB network.

MIDI 1.0 over the network

While the old MIDI DIN5 had universal character, the MIDI 1.0 over the network requires to use a specific transport protocol. Multicast-MIDI and RTP-MIDI are the universal open-standards but different protocols exist and several different implementations are available for them.

The use of the MIDI protocols on the network by Win/Mac/iOS/Android is not homogeneous and often third-part solutions are necessarily applied. Only OSX (after ver. 10.4) and iOS (after ver. 4.2) have native MIDI protocol, the RTP-MIDI protocol. At the present stage our DIY system employs only the Multicast MIDI protocol. In this protocol, the same concept of connection is missing; the broadcasting of the messages happens as soon as you are tuned on the correct station. No IT/Networking expertise.
We plan to support the RTP-MIDI, which is the standard for the Apple’s systems, soon.

Automatic MIDI over the network mapping

With our software you are of course free to connect every your MIDI devices (USB, FireWire, PCI) to the operative system Windows/Mac. Once you connect them to our DIY system, you are exporting them in a bidirectional way on the network. They become single address devices to which you can promptly point. We limited this direct association only to USB-MIDI class-compliant devices that are compatible with a Linux’s kernel.

Multicast MIDI allows you to use up to 20 ports in a one-to-one communication with 20 simple MIDI devices (1xIN, 1xOUT). This number of 20 does not represent a physical barrier but it is the actual limit of the ipMIDI solution operating on Windows/Mac. This put a barriers to the scalability and the multiport MIDI devices can easily reach this upper limit and saturate the available resources.

RTP-MIDI does not present this scalability limits and it allows you to tag the devices present in the network.

However, even though RTP-MIDI is more suitable for complex MIDI networks, Multicast-MIDI is far away simpler and is not overloaded by the growth of the system. With Multicast-MIDI you get rid of the connection concept. To send a MIDI means to broadcast it one-to-all through the networking hardware. Vice versa, RTP-MIDI sets a private connection with every single unit in the network. From one hand it allows you a more precise control but on the other hand it overload the system when the number of connection increase (the IP protocol is indeed slow and heavy because of the high latency processing).

At the moment the DIY supports the simpler solution, i.e. the Multicast-MIDI, and the export of all the resources MIDI on the Multicast (21928 UDP). This means that all the events of the MIDI-OUT of all the connected devices to our DIY system (USB, PCI, FireWire, …) are broadcasted using the first Multicast MIDI port (21928 UDP). Every event received from the first Multicast MIDI port (21928 UDP) is forwarded to all the MIDI-IN ports of the MIDI devices connected to the DIY.

All the MIDI devices are projected neutrally in a transparent way into the network. Nevertheless, the protocol MIDI 1.0 remains unchanged (apart for the SYSEX limited to 512 bytes), and it is changed only the way to export the messages into the network by Multicast MIDI or RTP-MIDI. Different schemes plug-and-play are also available to export your MIDI devices on the network to grant the continuity every time you reboot your system.
Up to now, we implemented only a broadcast-type exporting system because all the system are considered as a global broadcasting system.

Application and future developments

When do you need our solution?
Every time you need performances, integrated solutions and control. Every time you need a MIDI controller for the console of a musical instrument, every time you need a MIDI controller for your synth, every time you need a gateway between different network MIDI protocols, every time you need a software to set-up complex network architecture (including wifi) etc.
We are offering you the ultimate solution for a development platform where every details can be controlled and extended with modular add-on packages way.
Further details in Appendix A represent the first step for the interaction with the black box.

Note that the potentiality of our system are enormous. Despite its association with music devices, MIDI can effectively control any device that can read and process a MIDI command (theatre lighting, sound design, audio processor control, console automation).

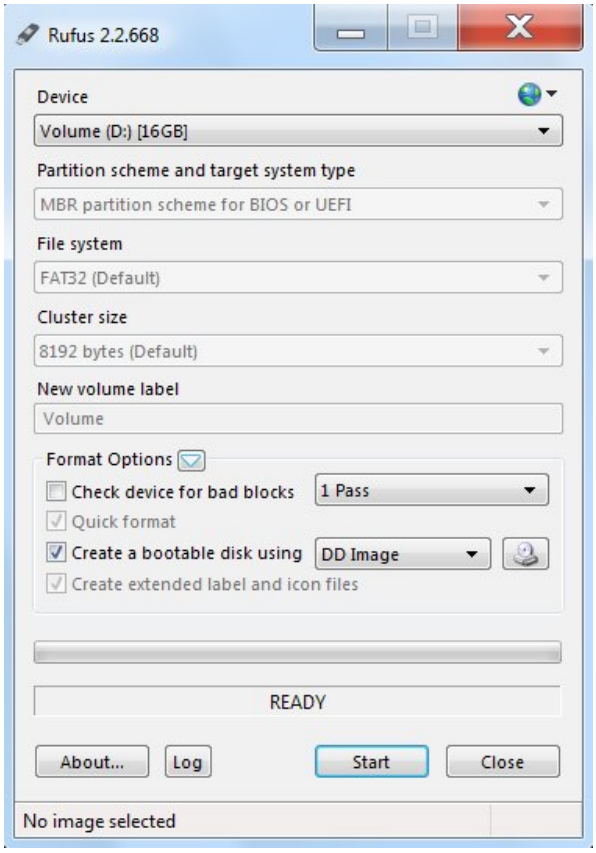
We offer you a versatile meta-product able to be tuned on the final-user necessities. The software implementation allows a deep and rapid development without heavy and expensive modifications, i.e. just and upgrade the software without changing your hardware.

Quick Start

At the moment the free edition runs on any x86 Asus/Asrock platform with TPM support. Some platforms have an integrated TPM (Q87 chipset). Tipically you need to add the TPM as hardware plug-in. Choose freely the CPU and RAM. Faster processors mean more real-time performance. Intel Celeron and Intel Pentium are the best choices.

Be sure to turn on and enable the TPM.

Download Rufus (http://rufus.akeo.ie/) and use it to write the **netmidiegpfree01.img** image file in a USB disk. Make sure to set “DD Image” writing mode. All data on the USB disk will be lost! Boot the DIY system and everything is ready.



If the USB disk has been written correctly then you will see the following banner with the software version:

"DIY NETWORK MIDI ENCODER, GATEWAY, PROCESSOR : FREE EDITION v0.1, PERSONAL USE ONLY!".

The software could fail for the following errors:

FATAL ERROR : TPM NOT FOUND

FATAL ERROR : TPM DISABLED

FATAL ERROR : NO NETWORK FOUND

FATAL ERROR : INTEGRITY ERROR OR UNKNOWN PLATFORM

.

The following table compares the free edition performance on some hardware platforms. The values are only indicative and the performance can be even 10 times higher without any delay added. The free edition is very generous and you can do anything for your personal use only. The DIY system may hang only at startup. Do not worry, reboot the system. This is only a protection mechanism.

Platform x86	**CPU model/cores**	**USB to NET performance**	**NET to USB performance**	**PCI encoder performance with/without velocity**
ASUS Q87T	Intel Pentium G3450 3.40GHz/2	80...25us	75us	90/60us
ASUS H81T	Intel Celeron G1840 2.80GHz/2	85...35us	100us	110/80us
ASRock AM1H-ITX	AMD Athlon 5350 2.0GHz/4	170...120us	250us	320/200us
ASRock AM1H-ITX	AMD Sempron 3850 1.3GHz/4	220...180us	390us	520/320us
ASRock Q1900TM-ITX	Intel J1900 2.0GHz/4	220...150us	250us	380/230us

The DIY system works on the network. You must install the ipMIDI Multicast driver on Windows/Mac. Check the Drivers link for more details.

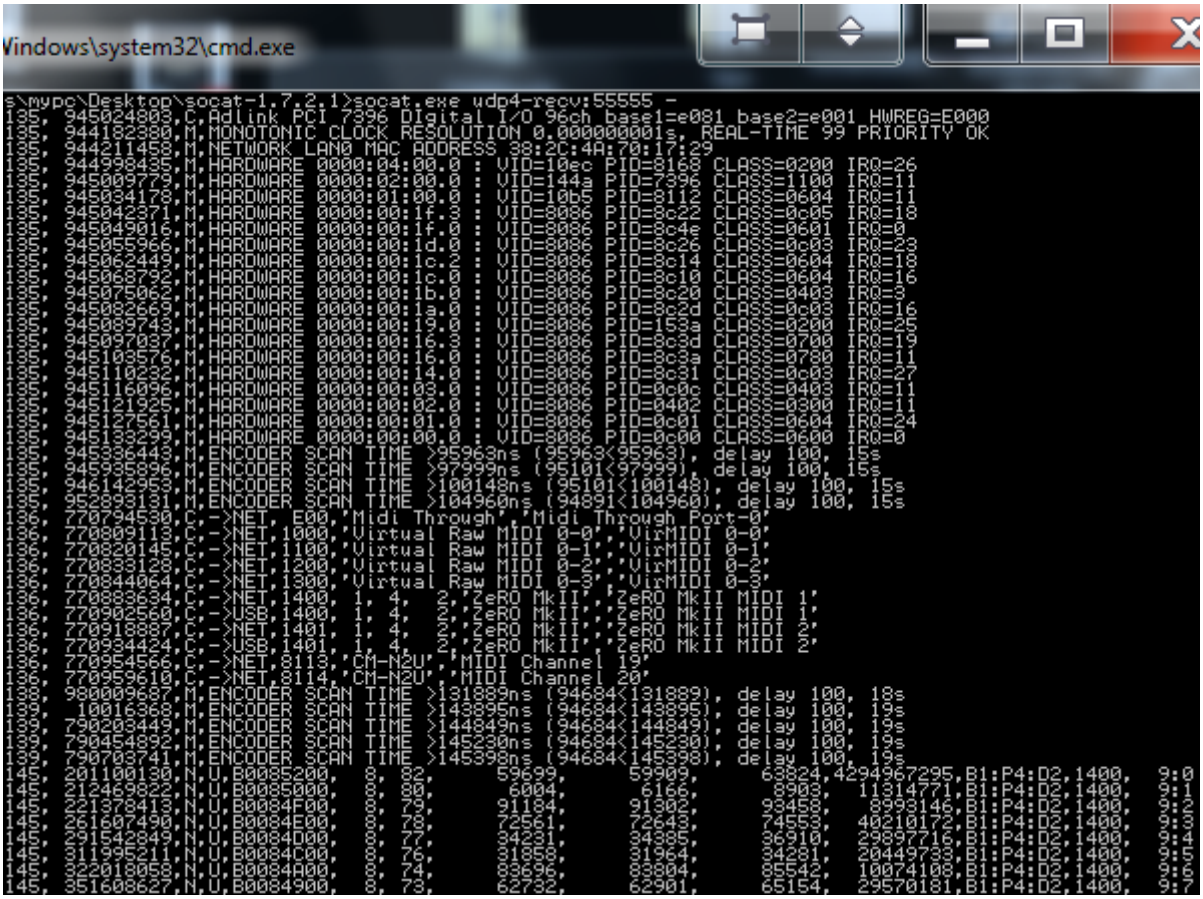
The plug-and-play DIY system export all your MIDI devices on the network in both directions. There are several schemes to export MIDI devices on the network. “Broadcast” is the most simple scheme that is now available :

SOURCE	DESTINATION
USB1, USB2, …, USBn	Multicast MIDI (21928 UDP, MIDI first port)
Multicast MIDI (21928 UDP, MIDI first port)	USB1, USB2, …, USBn

Practically the host acts as a gateway between the USB-MIDI class compliant domain and Multicast MIDI network domain in both directions.

No IT/Networking expertise required. The system works with or without a DHCP server on the network. Just use the ipMIDI driver and consider the network as a special MIDI DIN5 cable.

Live net-console



Logs are simply live broadcast streaming on 55555 UDP port (IPv4 255.255.255.255).

Download socat (http://www.nikhef.nl/~janjust/socat/), extract the contents to your Desktop and open the windows terminal. Finally enter the socat folder with the cd command and run the socat.exe application as follows.
C:\Users\myuser>cd Desktop
C:\Users\myuser\Desktop>cd socat-1.7.2.1
C:\Users\myuser\Desktop\socat-1.7.2.1>**socat.exe udp4-recv:55555 -**

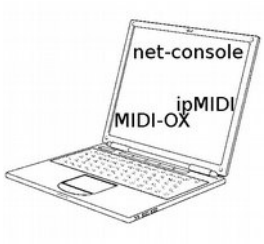
We are now tuned to the default receive channel.

To inspect the low-level USB traffic you must use the following command :

C:\Users\myuser\Desktop\socat-1.7.2.1>**socat.exe udp4-recv:55565 -**

The URB (USB Request Block) log are a kernel feature :
https://www.kernel.org/doc/Documentation/usb/usbmon.txt .

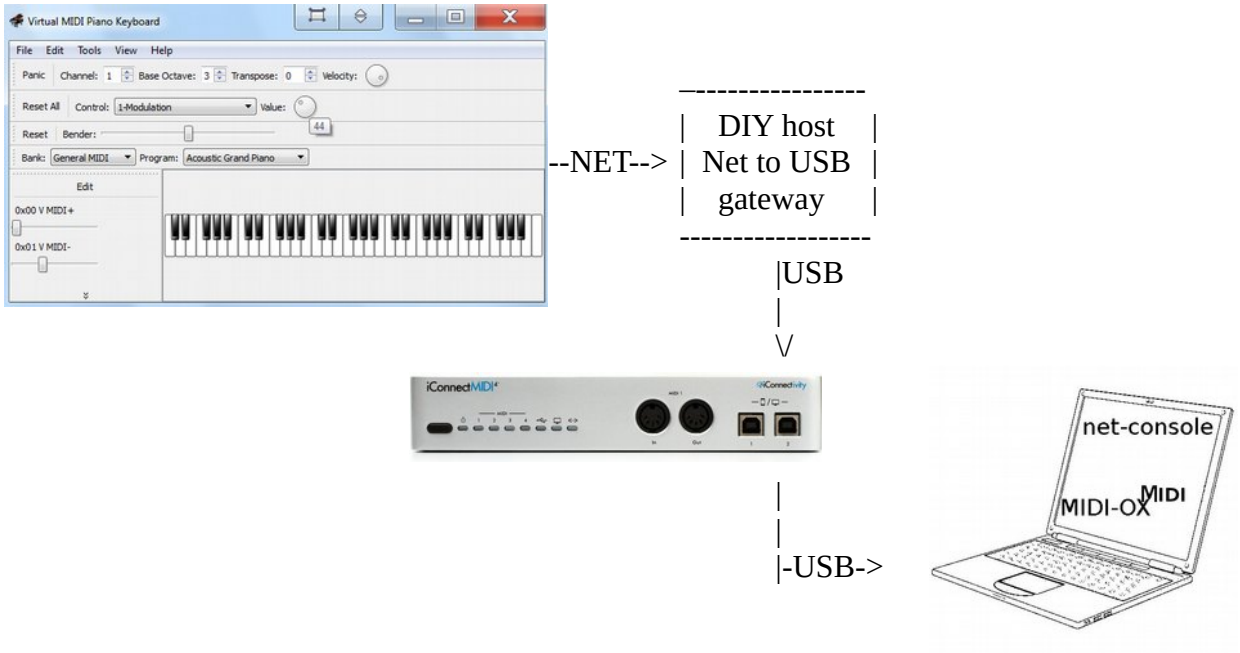
USB to NET gateway net-console example




```
|                  ------------------
|                  |    DIY host    |
|-----USB----->    | USB to Net     |--NET-->
|                  |    gateway     |
                   ------------------
```

Monotonic clock(sec)	(ALSA identification)	Monotonic clock(ns)	to Network	from USB	MIDI event (hex)	MIDI byte2	MIDI byte3	1st processing monotonic delay(ns)	2st processing monotonic delay(ns)	last monotonic delay(ns) after net send	net delay(ns) from the previous packet	USB hw bus/port/device	ALSA dev id	
5074	791175564	N	U	B0026300	2	99	39562	40041	44839	371406	B1:P2:D2	1800	9:1279	
5074	791661893	N	U	B0011F00	1	31	39437	40030	51407	492897	B1:P2:D2	1800	9:1280	
5074	792228862	N	U	B0011E00	1	30	78973	79485	88657	604219	B1:P2:D2	1800	9:1281	
5074	792239241	N	U	B0026500	2	101	216642	217188	225596	147318	B1:P2:D2	1800	9:1282	
5074	792737468	N	U	B0011D00	1	29	91888	92406	100974	373605	B1:P2:D2	1800	9:1283	
5074	792758851	N	U	B0026600	2	102	219544	220116	228917	149326	B1:P2:D2	1800	9:1284	
5074	793298435	N	U	B0026700	2	103	54539	55036	63829	374496	B1:P2:D2	1800	9:1285	
5074	793794610	N	U	B0011A00	1	26	45044	45664	54091	486437	B1:P2:D2	1800	9:1286	
5074	793802028	N	U	B0026800	2	104	188236	188854	197544	150871	B1:P2:D2	1800	9:1287	
5074	794306139	N	U	B0011900	1	25	59567	60113	68644	375211	B1:P2:D2	1800	9:1288	

NET to USB gateway net-console example



Monotonic clock(sec)	(Network identification) Monotonic clock(ns) to USB MIDI-IN	source IP:port	destination IP:port	MIDI event code (hex)	counter processed:lost	net delay(ns) from the previous packet	monotonic delay(ns) after USB send	
8272	514770895 U	169.254.81.73:50979	225.0.0.37:21928	B0016B	000002022:000	7907524	13888
8272	530401989 U	169.254.81.73:50979	225.0.0.37:21928	B00169	000002023:000	15631094	13867
8272	541419760 U	169.254.81.73:50979	225.0.0.37:21928	B00168	000002024:000	11017771	14225
8272	548595281 U	169.254.81.73:50979	225.0.0.37:21928	B00167	000002025:000	7175521	14129
8272	554483118 U	169.254.81.73:50979	225.0.0.37:21928	B00166	000002026:000	5887837	13658
8272	562481978 U	169.254.81.73:50979	225.0.0.37:21928	B00164	000002027:000	7998860	14149
8272	570787861 U	169.254.81.73:50979	225.0.0.37:21928	B00162	000002028:000	8305883	13625
8272	578910364 U	169.254.81.73:50979	225.0.0.37:21928	B00161	000002029:000	8122503	13454
8272	586899984 U	169.254.81.73:50979	225.0.0.37:21928	B00160	000002030:000	7989620	13603
8272	594821912 U	169.254.81.73:50979	225.0.0.37:21928	B0015F	000002031:000	7921928	13709

Live programming

The system configuration is not persistent. Any configuration is lost when the system restarts. You can also reset the configuration to the initial settings. Appendix A lists the programming specifications.

There are three ways to program the system:

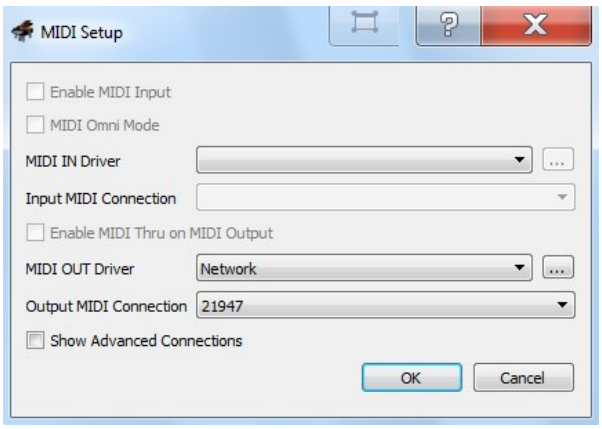
- local : USB-MIDI on a reserved USB bus/port (to be enabled via network , also you need to know the bus/port USB numbers to be activated , the net-console helps you find them)

- network : MIDI Multicast IPv4 225.0.0.37, 21947 UDP port (no feedback). Use the vmpk ***netmidiegpfree01_vmpk060.reg*** template file

- network : OSC Multicast IPv4 225.0.0.37, 21948 UDP port (unicast feedback, requires the dhcp service on the network). Use the lemur ***netmidiegpfree01_lemur.jzml*** template file

The OSC protocol is bidirectional. This means that with OSC you can actively interact with the host. All controls (faders, knobs, ...) can be set directly from the DIY host. In fact with a reset to default this happens with OSC but not with MIDI.

So use vmpk for free programming:



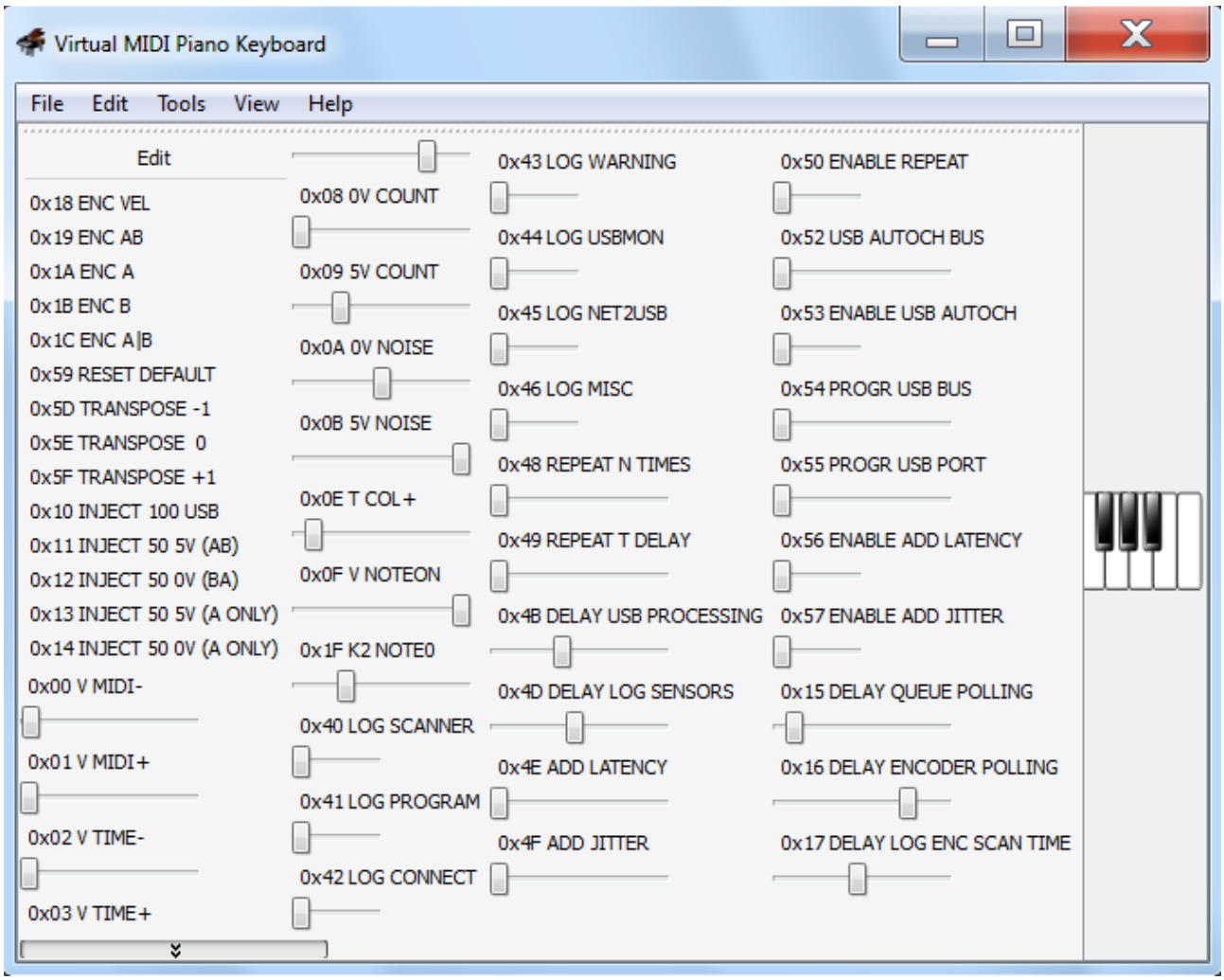
- install windows vmpk version 0.6.0

- configure MIDI OUT to 21947 port

- close vmpk application and load the ***netmidiegpfree01_vmpk060.reg*** template (this is a trick, unfortunately there are no other methods)

- simply use vmpk with the net-console feedback

.



vmpk is a free tool but does not always work optimally. If discomfort persists, you can return to the factory default by deleting the data in the registry and only after closing vmpk : [HKEY_CURRENT_USER\Software\vmpk.sourceforge.net\VMPK-Virtual MIDI Piano Keyboard\]

The next time all data will be recreated. (Uninstall and re-install the vmpk application does not solve this problem.) Perhaps the Mac/Linux version is more stable. Here are the files where vmpk writes configuration data.

OS	vmpk	file/registry path
Windows	0.6.0	\HKEY_CURRENT_USER\Software\vmpk.sourceforge.net\VMPK-Virtual MIDI Piano Keyboard\ExtraControllers
OSX	0.6.0	~/Library/Preferences/net.sourceforge.vmpk.VMPK-Virtual\ MIDI\ Piano\ Keyboard.plist
Linux	0.6.0	~/.config/vmpk.sourceforge.net/VMPK-Virtual\ MIDI\ Piano\ Keyboard.conf

PCI matrix encoder

Nobody forbids you to connect one or more USB keyboards to the system and use it immediately. A PCI(e) Digital I/O card enables the following advantages:
- check/set the scanning process in detail
- increase the real-time performance of all keyboards, synchronously
- set the velocity response curve
- implement new features: toggle/momentary keys, cluster keys, ...
- anything that can be useful
.

The diode keyboards forces pulled-down mode with the PCI(e) Digital I/O cards. The switch under the key provides a resistance of 100Ohm about so it is usually not necessary to apply external resistors but it's good to check.

Most Digital I/O PCI(e) boards emulate the Intel 8255 8-bit chip. This is the Adlink PCI-7296 and Adlink PCIe-7296. PCIe-7296 is not natively PCI Express. It integrates a PCI to PCIe bridge.
The Adlink PCI-7396 is not Intel 8255 compatible and offers increased performance with 32-bit I/O access.
You need to read the manufacturer's specifications for details.

Each group of 8 adjacent pins (A, B or C) can be programmed for input or for output. In the specific case there are two groups of 8 pins used as output. The outputs have been doubled in order to manage and balance a large number of simultaneous events (they may be insufficient for heavy loads).
This means two synchronous scans but there is only one 8x80 matrix!

The matrix is handled in this way.

	A								B								C							
CH1 8x16	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
CH2 8x24	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CH3 8x16	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
CH4 8x16	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95

XX = OUTPUT (WARNING)
XX = Contact A, Keyboard
XX = Contact B, Keyboard
XX = Simple 8x8 matrix (CH0)

If you want to use the velocity then there is only one mode with two contacts. If the velocity is not needed then the two contacts can be used in various configurations: A only, B only, A&B (slow keyboard), A|B (fast keyboard).
You can decide to change the operating modes via software. As easy as pushing a button.

Remember to set the jumpers in the pulled-down position. 7296 and 7396 Adlink cards have a different connection scheme.

Be very careful to identify the correct pins. Use the voltmeter to verify the output pins. They will have a higher voltage than the 0V input pins.

The order of connection goes from the smaller number to the larger number, both for input and outputs. The contact B follows the contact A during the key pressing. If possible, connect both contacts. You can decide to change the operating modes via software. As easy as pushing a button. The default mode is velocity.
If the velocity is not needed then the two contacts can be used in various configurations: A only, B only, A&B (slow keyboard), A|B (fast keyboard).

All keyboards are starting from 36 note number (C1). The starting note can be changed only with the 8x24 matrix (from 0 to 36). Finally the last matrix uses a single contact.

Velocity keyboard log example

```
|---------|→ monotonic clock (s + ns)
|         |   |-|→ scanner event (0=0V, 1=5V)
|         |   | |  |--|--|→ matrix,col,row
|         |   | |  |  |  |     |→ previous full scan time (ns)
|         |   | |  |  |  |     |        |→ current input 80-bit read time (ns)
|         |   | |  |  |  |     |        |
4042, 207276048,S,1, 0, 2, 0,  95921,   7746
4042, 208837737,S,1, 0, 0, 0,  95890,   7768
4042, 218730258,S,1, 1, 0, 0,  95863,   7748
4042, 218730258,N,S,91246E00, 36,110, 73502, 73732, 74805,4294967295,   9892521,110,  1,100, 10,120
4042, 219743889,S,1, 1, 2, 0,  94882,   7775
4042, 219743889,N,S,91266B00, 38,107, 91074, 91347, 93770,   1032596,  12467841,107,  1,100, 10,120
4046, 401936815,S,0, 1, 0, 0,  95833,   7783
4046, 402558981,S,0, 1, 2, 0,  94942,   7747
4046, 413771381,S,0, 0, 0, 0,  94813,   7752
4046, 413771381,N,S,81240000, 36,  0, 95338, 95584, 96631,4294967295,  11834566,  0,  1,100, 10,120
4046, 416952788,S,0, 0, 2, 0,  95836,   7742
4046, 416952788,N,S,81260000, 38,  0, 91644, 91878, 94367,   3179143,  14393807,  0,  1,100, 10,120
|    |     | |    |    |  |    |    |      |       |      |  |  |  |  |
|    |     | |    |    |  |    |    |      |       MIDI velocity min,max ←|--|
|    |     | |    |    |  |    |    | velocity delta time min,max (ms) ←|--|
|    |current velocity delta time (ns) and MIDI velocity ←|-----|
|    |     | net delay(ns) from the previous packet ←|
|    |     | |    |    |  |    |    |
|    |     | |    |    |  |    |----|-----|→ elapsed time from hw event (ns)
|    |     | |    |    |--|→ MIDI byte2, byte3 (decimal)
|    |     | |    |→ MIDI event code (hex)
|    |     |-|→ to network from encoder
|----|→ monotonic clock (s + ns)
```

A|B keyboard log example

```
|---------|→ monotonic clock (s + ns)
|         |   |-|→ scanner event (0=0V, 1=5V)
|         |   | |  |--|--|→ matrix,col,row
|         |   | |  |  |  |     |→ previous full scan time (ns)
|         |   | |  |  |  |     |        |→ current input 80-bit read time (ns)
|         |   | |  |  |  |     |        |
```

```
3792, 755829667,S,1, 0, 2, 0,  95873,   7742
3792, 755829667,N,S,91267F00, 38,127, 16397, 16839, 18280,4294967295
3792, 761148942,S,1, 0, 0, 0,  95795,   7749
3792, 761148942,N,S,91247F00, 36,127, 74690, 74975, 76179,  5377174
3792, 792743168,S,1, 1, 0, 0,  94957,   7751
3792, 795927863,S,1, 1, 2, 0,  95861,   7801
3794, 680321728,N,S,81240000, 36,  0,  5603,  5975,  8229,1919104836
3794, 680321728,S,0, 1, 0, 0,  94998,   7749
3794, 683110398,S,0, 1, 2, 0,  94957,   7787
3794, 683110398,N,S,81260000, 38,  0, 84763, 84899, 85827,  2866268
3794, 684861998,S,0, 0, 0, 0,  94961,   7764
3794, 688040393,S,0, 0, 2, 0,  94924,   7743
```

```
|     |      | |     |     |  |     |     |      |        |
|     |      | net delay(ns) from the previous packet ←|
|     |      | |     |     |  |     |     |      |
|     |      | |     |     |  |     |----|-----|→ elapsed time from hw event (ns)
|     |      | |     |     |  |     |--|→ MIDI byte2, byte3 (decimal)
|     |      | |     |→ MIDI event code (hex)
|     |      |-|→ to network from encoder
|----|→ monotonic clock (s + ns)
```

A & B keyboard log example

```
|---------|→ monotonic clock (s + ns)
|         |   |-|→ scanner event (0=0V, 1=5V)
|         |   | |  |--|--|→ matrix,col,row
|         |   | |  |  |  |     |→ previous full scan time (ns)
|         |   | |  |  |  |     |        |→ current input 80-bit read time (ns)
|         |   | |  |  |  |     |        |
```

```
4281, 110353655,S,1, 0, 2, 0,  94836,   7791
4281, 110530099,S,1, 0, 0, 0,  98025,   7757
4281, 121975784,S,1, 1, 0, 0,  94841,   7777
4281, 121975784,N,S,91247F00, 36,127, 97096, 97310, 98634,4294967295
4281, 124571258,S,1, 1, 2, 0,  94953,   7827
4281, 124571258,N,S,91267F00, 38,127, 88053, 88288, 89713,  2586553
4282, 932531040,S,0, 1, 0, 0,  95759,   7749
4282, 932755452,S,0, 1, 2, 0,  98930,   7749
4282, 941436994,N,S,81240000, 36,  0, 52125, 52371, 53781,1816829804
4282, 941436994,S,0, 0, 0, 0,  94741,   7769
4282, 943431161,N,S,81260000, 38,  0,  5610,  5729,  6969,  1947355
4282, 943431161,S,0, 0, 2, 0,  94807,   7765
```

```
|     |      | |     |     |  |     |     |      |        |
|     |      | net delay(ns) from the previous packet ←|
|     |      | |     |     |  |     |     |      |
|     |      | |     |     |  |     |----|-----|→ elapsed time from hw event (ns)
|     |      | |     |     |  |     |--|→ MIDI byte2, byte3 (decimal)
|     |      | |     |→ MIDI event code (hex)
|     |      |-|→ to network from encoder
|----|→ monotonic clock (s + ns)
```

Appendix A : programming specs

The DIY system can be programmed via MIDI (USB or Multicast) and OSC. The following specifications allow you to use any MIDI/OSC device. For immediate use you can use the preconfigured templates for vmpk and Lemur.

MIDI Control Change programming

CC# command on reserved USB bus/port or UDP Multicast MIDI to IPv4 225.0.0.37:21947 (USB programming is disabled by default and requires activation via UDP Multicast or OSC)

OSC programming

32-bit float on OSC path to IPv4 225.0.0.37:21948

MIDI CC#	**OSC path**	**DESCRIPTION**	**DEFAULT**
0x00	/B000/x	V MIDI- : encoder velocity, min MIDI	20
0x01	/B001/x	V MIDI+ : encoder velocity, max MIDI	100
0x02	/B002/x	V TIME- : encoder velocity, min time (ms)	1ms
0x03	/B003/x	V TIME+ : encoder velocity, max time (ms)	100ms
0x08	/B008/x	0V COUNT : encoder countdown to 0V	32
0x09	/B009/x	5V COUNT : encoder countdown to 5V	32
0x0A	/B00A/x	0V NOISE : tolerance level for 0V noise log	100
0x0B	/B00B/x	5V NOISE : tolerance level for 5V noise log	100
0x0E	/B00E/x	T COL+ : encoder diode delay, time wait after col switch (1=250ns)	10
0x0F	/B00F/x	V NOTEON : encoder, default note-on static velocity	127
0x10	/B010/x	INJECT 100 USB : inject 100 ALSA seq events	-
0x11	/B011/x	INJECT 50 5V(AB) : inject 50 velocity events on, first matrix	-
0x12	/B012/x	INJECT 50 0V(BA) : inject 50 velocity events off, first matrix	-
0x13	/B013/x	INJECT 50 5V (A) : inject 50 events on, first matrix	-
0x14	/B014/x	INJECT 50 0V (A) : inject 50 events off, first matrix	-
0x15	/B015/x	DELAY QUEUE POLLING : scanner/ALSA queue polling delay (0=disable or 1...127us)	10us
0x16	/B016/x	DELAY ENCODER POLLING : next full scan delay (0=disable or 1...127us)	100us
0x17	/B017/x	DELAY LOG ENC SCAN TIME : encoder scan time ciclic log (0=disable or 1...127s)	60s
0x18	/B018/x	ENC VEL : velocity-sensitive keyboard (A&B)	-
0x19	/B019/x	ENC A&B : no velocity (A and B), slow keyboard	-
0x1A	/B01A/x	ENC A : no velocity (only A), fast keyboard	-

0x1B	/B01B/x	ENC B : no velocity (only B), fast keyboard	-
0x1C	/B01C/x	ENC A|B : no velocity (A or B), very fast keyboard	-
0x1F	/B01F/x	K2 NOTE0 : first note 8x12 matrix (0...36)	36
0x40	/B040/x	LOG SCANNER : if >0 net-console 55556 UDP	0
0x41	/B041/x	LOG PROGRAM : if >0 net-console 55557 UDP	0
0x42	/B042/x	LOG CONNECT : if >0 net-console 55558 UDP	0
0x43	/B043/x	LOG WARNING : if >0 net-console 55559 UDP	0
0x44	/B044/x	LOG USBMON : if >0 net-console 55560 UDP	0
0x45	/B045/x	LOG NET2USB : if >0 net-console 55561 UDP	0
0x46	/B046/x	LOG MISC : if >0 net-console 55562 UDP	0
0x48	/B048/x	REPEAT N TIMES : redundant packets to network, all MIDI msg (0=disable or 1...127 times)	0
0x49	/B049/x	REPEAT T DELAY : delay between redundant pakets (0=disable or 1...127us delay)	0
0x4B	/B04B/x	DELAY USB PROCESSING : ALSA delay processing to network (0=disable or 1...127us)	50us
0x4D	/B04D/x	DELAY LOG SENSORS : log cpu load average, sensors (fans, temperatures, voltages)	60s
0x4E	/B04E/x	ADD LATENCY : latency to network (0.1ms)	0
0x4F	/B04F/x	ADD JITTER : random jitter to network (1ms)	0
0x50	/B050/x	ENABLE REPEAT : redundancy to network 0=disable, >1=enable	0
0x52	/B052/x	USB AUTOCH BUS : set usb bus number or 0 to all usb bus (ch MIDI overwritten with usb port number)	0
0x53	/B053/x	ENABLE USB AUTOCH : 0=disable, >1=enable	0
0x54	/B054/x	PROGR USB BUS : set usb bus numer for reserved programming port (0=disable)	0
0x55	/B055/x	PROGR USB PORT : set usb port numer for reserved programming port (0=disable)	0
0x56	/B056/x	ENABLE ADD LATENCY : 0=disable, >1=enable	0
0x57	/B057/x	ENABLE ADD JITTER : 0=disable, >1=enable	0
0x59	/B059/x	RESET DEFAULT : 00, 01, 02, 03, 08, 09, 0A, 0B, 0E, 0F, 18, 1E, 1F, 40, 41, 42, 43, 44, 45, 48, 49, 4B, 4E, 4F, 50, 53, 56, 57, 5E	-
0x5D	/B05D/x	TRANSPOSE -1 : only note-on/note-off	-
0x5E	/B05E/x	TRANSPOSE 0 : only note-on/note-off	0
0x5F	/B05F/x	TRANSPOSE +1 : only note-on/note-off	-
-	/ACT/value	OSC activity led feedback (0.0f to 1.0f)	-