Erik Bartmann

Discover modular synthesizer with VCV-Rack 2

What is MIDI?

Author	Erik Bartmann
Internet	https://erik-bartmann.de/
Topic	MIDI basics
Version	1.01
Date	January 21, 2022

Table of contents

Basic knowledge of MIDI

What is it all about?

The following topics are discussed in this paper.

- Historical
- What is MIDI?
- MIDI IN, MIDI OUT, MIDI THRU
- Das MIDI-Protocol
- The MIDI channels
- The MIDI information
- The program MidiView

Electronic musical instruments such as synthesizers are already a fine thing and it is great fun to deal with it. If you build up a small music studio, then this will certainly increase a little over time, because here and there other devices or instruments are added. Here a drum computer, there a mixer or also effect devices round off the happening. Of course, such instruments can certainly provide a lot of joy on their own, but only in the interaction with each other it becomes really exciting. It is then of course necessary that a certain communication takes place among themselves, because if in addition also software in the form of a *DAW* (Digital Audio Workstation) is with the party, then everything should be synchronized in some way. Since the problem of communication between different devices or instruments was recognized very early on, thought was given to an interface through which information could be exchanged. If, for example, different keys are pressed on a synthesizer, then corresponding tones should sound. But how should the whole thing work if, for example, a key is pressed on a master keyboard and a sound generator in the form of a *VCO* (Voltage Controlled Oscillator) has to react to it in a modular system? The abbreviation VCO already contains the solution, because it is about voltages that make sure that something corresponding is triggered at the connected devices. This so-called *CV* (Control-Voltage) is used to influence the pitch, for example, via analog voltages. But not only the pitch, but also other parameters of a synthesizer like volume, filter behavior or envelopes can be manipulated by it. So it is obvious that all communication is done via different voltage levels. Regarding modular systems, this is of course a common practice, although attention must of course be paid to possible losses of voltage levels due to different cable lengths or external influences. When trying to interconnect several different instruments, a different solution must be found. Various manufacturers have joined forces to develop a standardized interface, which should be simple and also inexpensive to implement. A computer scientist by the name of *Dave Smith* was heavily involved in this development and even received an award in the *Mix Foundation TECnology Hall of Fame* in 2005 for the specification of this interface. But which interface is it at all? It is about the so-called *MIDI* interface

What is MIDI

MIDI is the abbreviation for *Musical Instruments Digital Interface* and is a technology that allows electronic musical instruments to communicate with each other. So all devices that speak this language can talk to each other. This connection is quite simple to establish, because there is only one socket with five pins (only 3 are used), which are internally divided into three types of sockets.

- MIDI IN
- MIDI OUT
- MIDI THRU

The *MIDI IN* jack sends information into the respective device and the *MIDI OUT* jack sends information out accordingly. The *MIDI THRU* socket simply forwards the signal from the IN, which can be used to connect several devices in series that are to receive the same information.

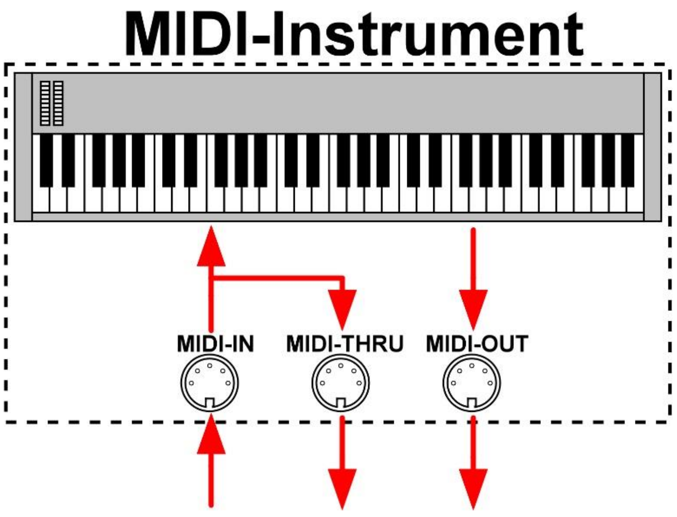


Figure 1 The three MIDI jacks

In the age of USB, MIDI information can also be transmitted via the USB interface. The following figure shows these two possibilities arranged one below the other.

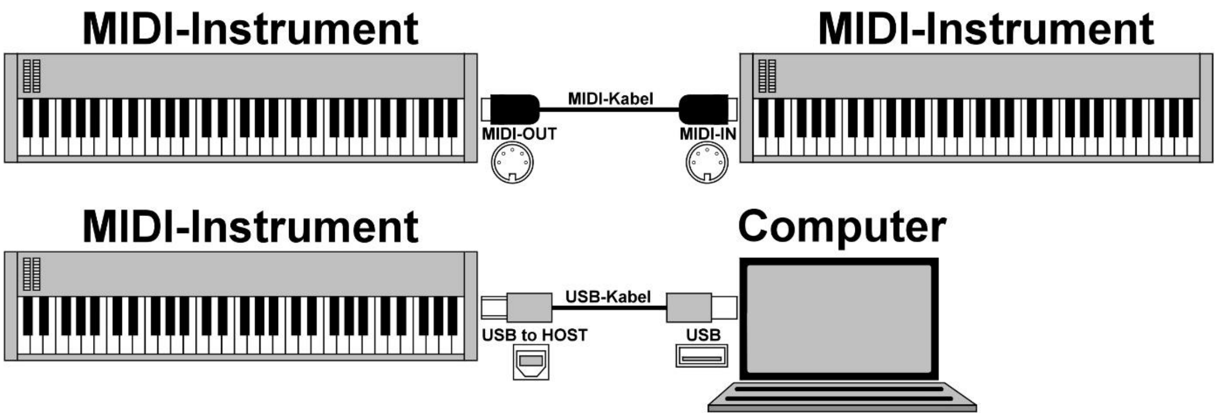


Figure 2 Various MIDI connections

But in what way do the different devices communicate with each other? It is a special protocol that was developed especially for MIDI. The starting point was the situation where it was a matter of controlling other devices such as synthesizers via a keyboard. In the early days of development, the solution was simply to use more or less long cables that transmitted analog signals. Of course, this method entailed the problems already mentioned with regard to voltage losses, which could occur with longer cables and lead to inconsistencies with regard to pitches

The MIDI-Protocol

The communication via the MIDI interface is done in a serial way, i.e. one after the other with a speed of 31250 bit/s. But what exactly is being transmitted and in what form? A MIDI message that is sent consists of three bytes.

What is a byte or a bit?

A *byte* is a unit of measurement used in digital technology and means a sequence of 8 bits. A *bit* is a logical state that can be either 0 or 1. 0 means that no current flows and 1 that a current flows.

These three bytes contain all relevant information that a controlled MIDI device such as a synthesizer needs to know which module it should play in which note, for how long and how loud. No sounds are sent via MIDI, as for example via a speaker line, but only control commands. This should not be forgotten! What do these three bytes look like and what is their meaning? On the following illustration this is clarified.

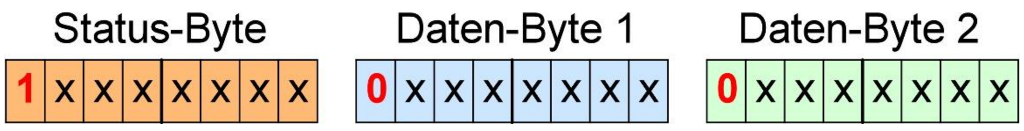


Figure 3 The three MIDI bytes

The transmission of a MIDI signal, which is composed of the three bytes mentioned, is a so-called *event*. Such events can be of different nature. Let's take a closer look.

A MIDI event

If a special event occurs, this can have different causes. A key on the keyboard is pressed and released at some point, or knobs are moved. These events have certain event assignments, which are divided into eight categories or *command types*, as the following table provides.

Command types	Status-Byte (binary)	Status (hex)
Note off	1000 *nnnn*	8*n*
Note on	1001 *nnnn*	9*n*
Poly Pressure	1010 *nnnn*	A*n*
Control Change	1011 *nnnn*	B*n*
Program Change	1100 *nnnn*	C*n*
Channel Pressure	1101 *nnnn*	D*n*
Pitch-Bend	1110 *nnnn*	E*n*

Table 1 MIDI command types

It can be seen that both a binary and a hexadecimal notation is used, which I assume as basic knowledge at this point. The shown information can be found in the first byte, the *status byte*. It transmits the command type and the MIDI channel (1 to 16), which is indicated by *nnnn*. The status byte always starts with a 1 in binary terms. This is used to distinguish it from the data bytes, which start with a 0. What about the mentioned MIDI channels? The following table gives information.

Binary	MIDI channel	Binary	MIDI channel
0000	1	1000	9
0001	2	1001	10
0010	3	1010	11
0011	4	1011	12
0100	5	1100	13
0101	6	1101	14
0110	7	1110	15
0111	8	1111	16

Table 2 MIDI channels

It should be noted that binary counting starts at 0 and is assigned to MIDI channel 1. Let's look at an example for this.

Note on

Assuming the following MIDI event is triggered regarding the status byte. The two data bytes 1 and 2 play no role in this case.

10010000

Let's take a closer look.

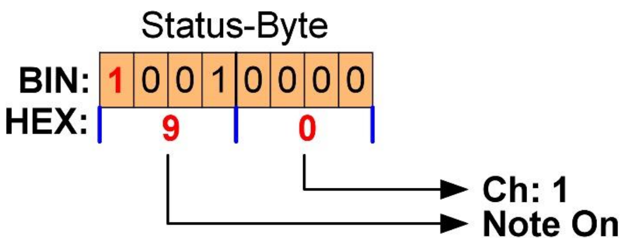


Figure 4 The status byte for Note on

The left 4 bits show that this is the HEX value 9, which according to the first table of MIDI command types stands for the event *Note on*. The right 4 bits represent the desired MIDI channel, which can be seen here with the HEX value 0, which according to the second table for MIDI channels corresponds to channel 1. When there is a note in a MIDI clip in a DAW (Digital Audio Workstation), it is basically two separate MIDI messages. Note on for pressing a key and *Note off* for releasing the key on the keyboard.

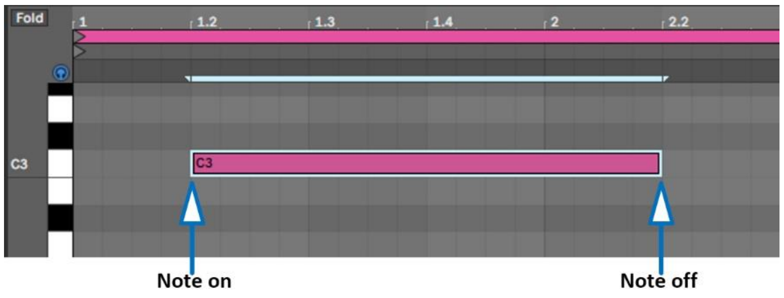


Figure 5 Note on and Note off in a DAW

The length of the note is represented by the length of the graphic bar and is located exactly between the two MIDI commands *Note on* and *Note off*. Now, however, further information is required, because the note number and the velocity (velocity) must also be transmitted with regard to the note.

Control-Change

Let's now move on to another example involving the control of an instrument with regard to a specific parameter. The event is called Control-Change and abbreviated with CC. This is where the two data bytes come into play, because further information such as *controller number* and the corresponding *value* for the change must be transmitted in the case of a *CC*. Let's assume that a knob is moved on a keyboard or a controller as shown in the following figure. How does this look like with the corresponding MIDI event?

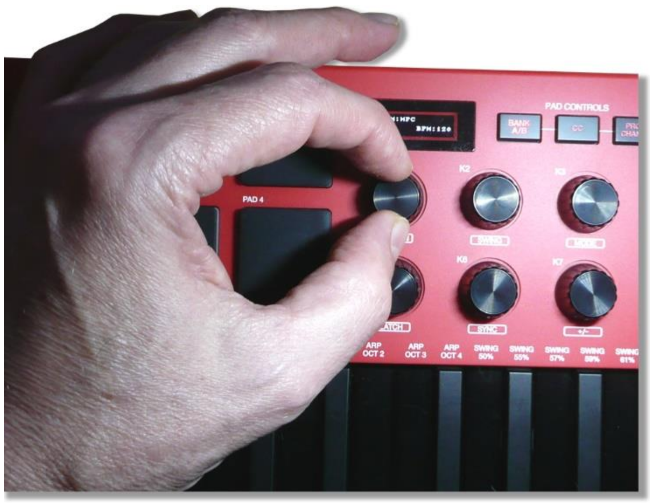


Figure 6 Knobs on the keyboard

As mentioned, all three bytes are required. The following MIDI event is registered.

10110000 01000110 00100110

What does this look like in detail? The first from the left is again the status byte, which looks like this.

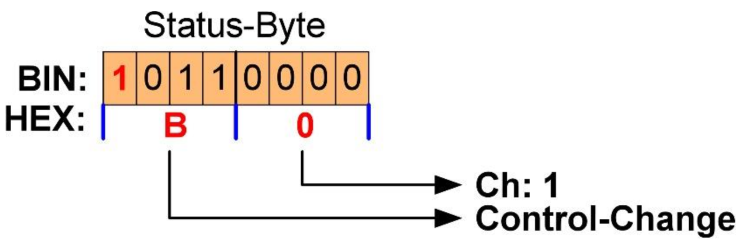


Figure 7 The status byte for CC

The left 4 bits show that this is the HEX value B, which according to the first table for MIDI message types stands for the *Control Change* event. The right 4 bits represent the desired MIDI channel, which can be seen here again with the HEX value 0, which according to the second table for MIDI channels corresponds to channel 1. Put together this results in the hexadecimal value B0, which is 176 in decimal notation. The following can be stated with regard to the individual channels.

- 176 + 0: Channel1
- 176 + 1: Channel2
- 176 + 2: Channel3
- etc.

Now it is important to know which control element (controller) was moved at all, which is determined by the data byte 1. By limiting the value to 7 bits - the left and most significant bit always has the value 0 -, values in the range between 0 and 127 (128 different values) can be achieved.

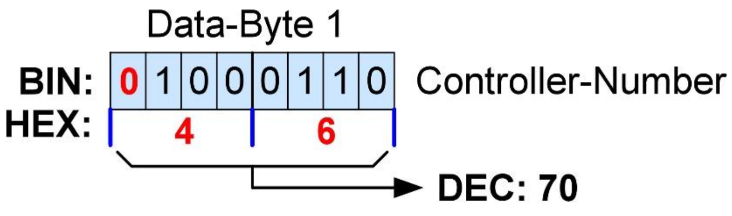


Figure 8 The data byte 1 for the controller number

The hexadecimal value 46 corresponds to the decimal value 70. And the knob I just turned is internally assigned the decimal value 70 as controller number.

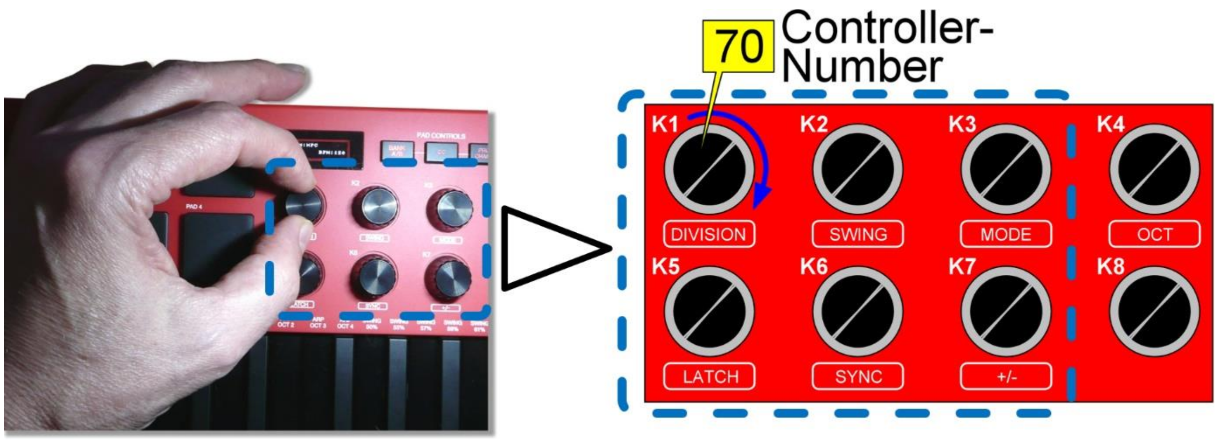


Figure 9 The rotary control with the decimal value 70

Since there are standardized operating elements on synthesizers or controllers, certain values are reserved that cannot be used. An example of this would be the so-called *MOD Wheel* with the value 01. Now it depends on the position of the knob, which is mapped via data byte 2.

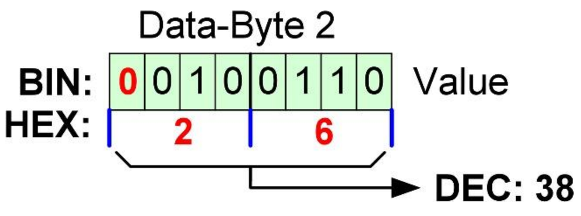


Figure 10 The data byte 2 for the value

Here it is the hexadecimal value 26 which results in 38 in decimal notation. Also in this case by the restriction of the value to 7 bits only values in the range between 0 and 127 (128 different values) can be determined. Regarding the very small value range of 128 different values, there is the possibility to send two identical CC events one after the other. This doubles the value range from 7-bits (128 values) to 14-bits (16,384 values), which is already an enormous increase and allows a much smoother gradation.

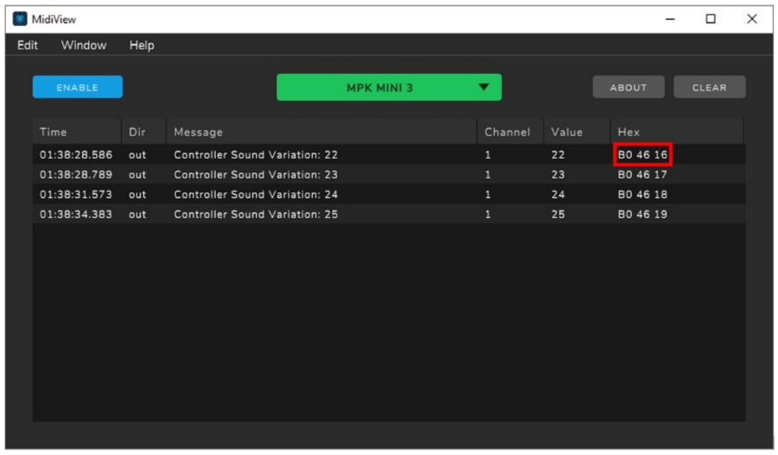
A MIDI monitor program

Maybe it is interesting to see how such MIDI information is sent from a keyboard or controller and for this purpose there is a free program called *MIDIView* available at the following web address.

Hyperlink!

https://hautetechnique.com/midi/midiview/

So I selected my keyboard in MidiView and moved the knob that I just used already.



Time	Dir	Message	Channel	Value	Hex
01:38:28.586	out	Controller Sound Variation: 22	1	22	B0 46 16
01:38:28.789	out	Controller Sound Variation: 23	1	23	B0 46 17
01:38:31.573	out	Controller Sound Variation: 24	1	24	B0 46 18
01:38:34.383	out	Controller Sound Variation: 25	1	25	B0 46 19

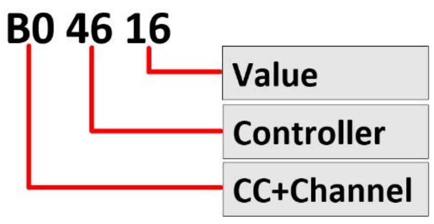


Figure 11 The MidiView program with concrete values

It is very good to observe what information is accumulating there. More information can be found on my website.

Hyperlinks!

https://erik-bartmann.de/

https://erik-bartmann.de/?Musik___VCV-Rack

Happy Frickeling!

Erik Bartmann