

# The MIDI Architecture and Device Connectivity

## 3.1 Introduction.

An important distinction when discussing sound recording is to distinguish between computer generated sounds via MIDI devices, and computer generated sounds through digital recordings; either pre-recorded, such as on an audio CD, or 'captured' to the computer's hard drive by connecting a device into the computer's sound card input sockets. MIDI generated sound is often referred to as 'music', where digital recordings is referred to as 'sound'. Thus there are two kinds of sound a computer can make, digital, and synthesized. Digital sounds are recordings of real sounds, while synthesized sounds are programmed.

### 3.1.2 Digital Sound

Digital sound involves digitization, which means to encode data in the form of ones and zeros, such as 1001001100110001. This is how a CD player works. The information on CDs is in digital form, whereas information on audio tapes is analog. For a computer to process sound, the sound must be converted into a digital data stream with an analog-to-digital converter. Once the sound is recorded onto the memory of the computer, it can be processed or played back with a digital-to-analog converter. These converters are devices that exist on the computers sound cards/modules, and are controlled by the software you use to process the sound, SoundForge on the PC, Sound Edit on the Apple Macintosh , for example.

Sound cards will have prerecorded digital samples stored on them. These 'samples' can then be played by sequencing software that will play a number of tracks simultaneously in an ensemble. Software such as Cakewalk, Steinberg's Cubasis, and eMagic's Logic Audio.

### 3.1.3 Synthesized Sound

Synthesized sound isn't digitally recorded; it's a mathematical reproduction of a sound based on a description. Synthesizers use hardware and algorithms to generate sounds on-the-fly from a description of the desired sound. Synthesized sound is the reproductions of sounds based on algorithms and hardware tone generators.

## 3.2 An Introduction to MIDI

MIDI is an acronym for Musical Instrument Digital Interface. MIDI is a set of agreed communication protocols and device connectivity standards established in 1983.

MIDI provides for a common interconnection of musical equipment such as keyboards and other devices including sound modules, sequencers and computers. MIDI provides the framework for music to be played on a music keyboard and recorded to a device in real time. The device that allows a collection of notes to be recorded in the order played is called a *sequencer*. Once recorded (sequenced) the piece can then be played back and the corresponding notes will sound on a keyboard or any other device that is connected to the sequencer device, such as a sound module. MIDI does not produce sounds, MIDI is a sort of computer coded language that sends messages to devices. The device that receives the MIDI message will respond accordingly, by sounding a musical note for example, based on the current selected program or *voice* (a voice is the term used to describe a particular instrument's type, such as a piano, or organ). A MIDI signal or *message* can be from a large collection of available messages defined in the MIDI protocol .

### 3.2.1 General MIDI

General MIDI or 'GM' MIDI is an agreed set of protocols that all manufacturers must implement. In particular the association or mapping of program numbers to voices. General MIDI establishes 128 (0-127) mappings between program number and voice type. From program 0, acoustic piano, to program 127, gunshot. The GM Percussion Map also provides access to a standard drum kit mapped to a predefined set of keys ranging from B0 through to A4.

### **3.2.2 The MIDI Architecture**

The MIDI architecture consists of channels, messages, and program numbers that are mapped to particular voices. A collection of interconnected MIDI devices can be thought of as a network of musical devices which are transmitting and or receiving messages for the purposes of decoding sequences of data that will result in an interpreted performance of real-time or pre-recorded music. Its also useful to consider MIDI as having a layered architecture. The *device layer* consists of the hardware and associate connectivity, cables, footswitch etc. The *communication layer*, which consists of the channels, and messages, and the *music layer* which consists of the actual voices and sound quality of these voices produced by hardware device tone generators. In some ways this music layer could be described as the presentation layer - as in 'what the user hears'.

### **3.2.3 Channels**

MIDI messages can be transmitted on 16 individual channels from 1 to 16 (channel 10 is reserved as the drum channel). Thus MIDI has the potential to communicate with 16 individual MIDI devices (where a MIDI device is a keyboard, sound module, drum machine, computer, etc.) In the early days of MIDI (early to mid 1980s) sound modules and other devices had only mono timbre capabilities, that is, the device could only play one voice at any given time.

### **3.2.4 Multi-Timbre MIDI Devices**

As hardware technology improved so did the multi-timbre capabilities of sound modules and related devices. Today (April 2005), most MIDI devices can support up to 16 individual voices and any limitations occur in sound processing hardware devices such as computers, that do not have enough system resources available (CPU, RAM) to support the simultaneous playback of 16 channels of instrument sounds.

### 3.2.5 MIDI Messages

A computer processes data which is fundamentally a collection of electrical states being either on or off. These states are represented by using a binary number system whereby, depending on the agreed representation, a 0 is off and a 1 is on. These are known collectively as one *bit*, one bit being either on or off. Eight bits are grouped into a structure known as a *byte*. Also, a byte will have a leftmost bit, and a rightmost bit. The leftmost bit is called the Most Significant Bit (MSB), as it represents the column which has the highest value ( *as in denary where the number 234 has 2 representing the hundreds column and thus is the most significant value*).

A MIDI message consists of a number of *bytes*, typically two or three. The first byte in a MIDI message indicates the type of byte as either a status byte, or a data byte. A status byte is indicated by the MSB of the byte being set to 1, a data byte is indicated by the MSB being set to 0. There are four categories of MIDI message: *Channel, System Exclusive, System Common, and System Realtime*.

#### 3.2.5.1 System Exclusive Messages

System Exclusive messages or *SysEx* messages are the MIDI provision for manufactures to include MIDI messages that will access their hardware devices with their own *exclusive* messages. Thus some manufactures can provide a way of accessing different features of their equipment that are not available using channel messages. Surprisingly however, not all SysEx messages are ‘exclusive’. There are *Universal* SysEx messages, an example of which is the Master Volume SysEx message governing global volume setting of a device. A further example of a universal message is the GM Reset, this message is recognized by all GM-compatible devices regardless of manufacturer.

### **3.2.5.2 System Common Messages**

System Common messages are used to transmit commands to one or more devices and achieve a common result. For example, a program change command sent to a keyboard which is also controlling another device, or slave, the System Common message would make the program change occur on both devices. System Common message are also used within the context of the synchronization of MIDI devices with a non-MIDI device such as a video or tape recorder usually via the MIDI Time Code message which is an adaptation of SMPTE time code.

### **3.2.5.3 System Real-Time Messages**

The typical use of a System Real-Time message is when it's necessary to achieve the synchronization of MIDI devices. One of the simplest applications of a System Real-Time message is where the drum tempo on the slave device is synchronized to the drum tempo of the master device.

### **3.2.5.4 Channel-Voice Messages**

Channel-voice messages include note on, note off, and other messages, including for example, to change the program (voice) from a piano to a guitar. Channel voice messages are used to transmit real-time performance data throughout a connected MIDI system. A channel voice message is generated whenever a MIDI instrument is played, selected or modified by the performer. Each channel voice message is assigned a channel number within its status byte, so that only devices that have the channel number matching that transmitted will respond to the commands. There are seven types of channel voice messages: note-on, note-off, polyphonic-key pressure, program change, control change, and pitch-bend change.

### 3.2.5.5 Note-On Messages

A note-on message is used to indicate the beginning of a MIDI note. A note-on message is generated whenever a note is triggered on a MIDI keyboard, drum machine or any MIDI instrument, or when playing a MIDI sequence from within a sound module or computer generated MIDI song. Note-on messages are made up of three bytes of information as shown in Figure 3.1 below.

Status / Channel	Note Value	Attack Velocity
(0-15)	0-127	0-127
(1001 0010)	(0100 0000)	(0101 1001)

Figure 3.1: Format of a MIDI note-on message.

The first byte in the message indicates to the receiving device that a MIDI note has been played, and the channel that is to receive the command. The second byte carries the actual MIDI value of the note that has been played. On an 88 note keyboard MIDI numbers range from 21 in the deep base to 108 in the far treble. With each natural key (white) and sharp/flat (black) being numbered in sequence across the entire keyboard. The final byte of the note-on message is the attack velocity the note will be played at. The attack velocity will determine how loud the note will sound within the context of the overall volume level set at the device. An attack velocity of 127 is loud as compared with an attack velocity of 50, say.

So the message shown in Figure 3.1 gives the first byte as note-on, channel 2 byte, second byte note value 64, and byte three indicates a velocity of 89

*Appendix One includes a diagram of MIDI note values mapped to full-size (88-note) keyboard pitch values.*

### **3.2.5.6. Note-Off messages**

Once a device has received a note-on message it will continue to play the note until it receives a note-off command. Each note that received a note-on message will continue to play until a corresponding note-off message for that note has been received. Thus the basis of musical composition can be encoded as a series of MIDI note-on note off events. The note-off structure consists of three bytes of information, note of status, the MIDI channel number, and a release velocity value. The release velocity value indicates the speed at which the key was released. Low values indicating the key was released slowly, high values indicating the key was released quickly. Not all MIDI devices can respond to the release velocity feature. Also, the note-off command does not cut the sound, if the sound has a release or final decay slope it will begin the stage once the note-off message is received by the device.

### **3.2.5.7 MIDI Sequences**

A collection of MIDI note-off and note-off commands together with other MIDI device control information may be recorded as a MIDI *sequence*. This MIDI sequence may then be loaded into a hardware sequencer or more typically a software sequencer and then played as via MIDI instruments and the corresponding musicality of the piece will be performed. Sequences may also be recoded in a MIDI file format to secondary storage devices such as computer hard disks and CDROMs. The MIDI file format is well documented and standardized across the music industry.

### **3.3.1 MIDI Device Connectivity**

A MIDI compatible device will have at least two sockets or *ports*. One to receive MIDI-IN commands and one to transmit MIDI-OUT commands. Some devices also have a MIDI THRU port which enables the commands being received at the MIDI-IN port to be sent as-is to another MIDI device. This allows one master MIDI device to control one or several MIDI slave devices. Where a device can be any MIDI capable musical instrument, controller or sound module etc. If a MIDI-THRU port is not available it may be implemented via software where the MIDI system incorporates a computer as one of the control devices.

The MIDI devices are interconnected with cables that have a five-pin DIN connector at each end. Most PC soundcards will have a so-called Game Port connector where an adaptor lead can be attached which will feature a Game Port connector for connecting to the soundcard and a MIDI-IN connector and MIDI-OUT connector at the other end for connecting MIDI devices into the computer system. Figure 3.2.

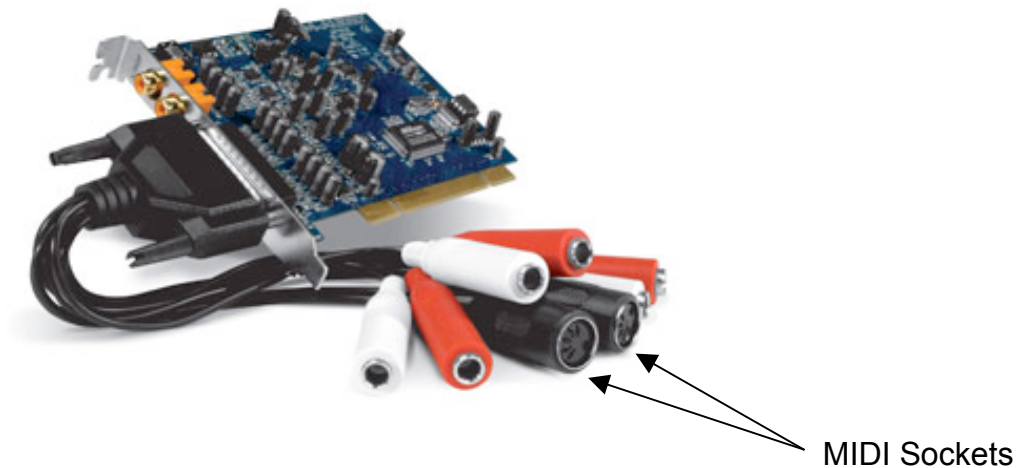


Figure 3.2: PC soundcard with MIDI and audio connectivity.

*Image courtesy M-Audio*

Laptop computers generally won't provide any on-board connectivity to external MIDI devices. This connectivity must be achieved by using so-called audio interfaces which are available with various levels of connectivity and sophistication. Most will provide at least the MIDI IN and MIDI OUT ports as shown in Figure 3.3 below. More sophisticated interfaces provide audio connectivity for microphones and other instruments such as guitars. Some also have integrated keyboard controller as shown in Figure 3.4 below. The connectivity from interface to computer is via the universal serial bus interface (USB). The USB connectivity is beginning to replace traditional five-pin DIN MIDI cables.



Figure 3.3 : Basic MIDI USB interface.

*Image courtesy M-Audio*



Figure 3.4 : MIDI keyboard controller with MIDI audio and USB interfaces.

*Image courtesy M-Audio*

### 3.3.2 Typical MIDI Configurations

A typical MIDI setup is shown in Figure 3.5 below. It consists of a PC system connected to a MIDI keyboard. The keyboard will receive and transit MIDI commands to and from the computer system typically from sequencing software. The MIDI out commands sent to the keyboard from the computer system are also sent on as-is from the keyboards MIDI-THRU port. Thus the computer system will control the keyboard and MIDID module simultaneously with one set of commands.

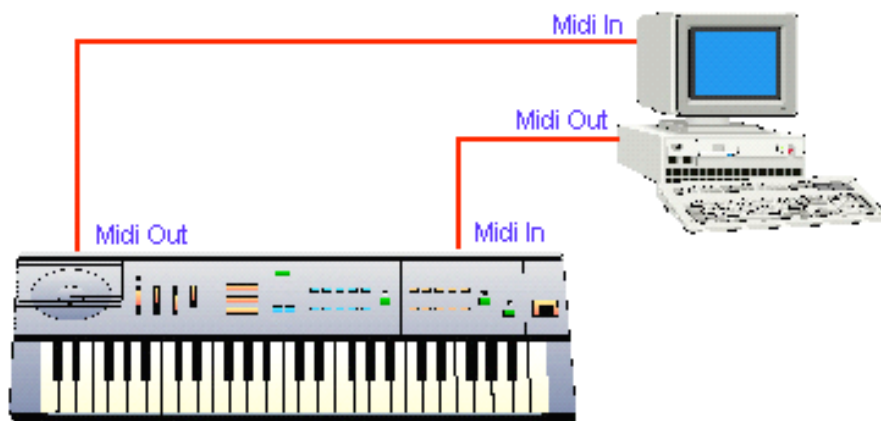


Figure 3.5: A typical MIDI setup.

A slightly more sophisticated MIDI setup is shown in Figure 3.6 below. It consists of a PC system connected to a MIDI keyboard which itself is connected to another MIDI device. The keyboard will receive and transit MIDI commands to and from the computer system (typically from sequencing software). The MIDI out commands sent to the keyboard from the computer system are also sent on as-is from the keyboard's MIDI-THRU port. Thus the computer system will control the keyboard and MIDI module simultaneously with one set of commands.

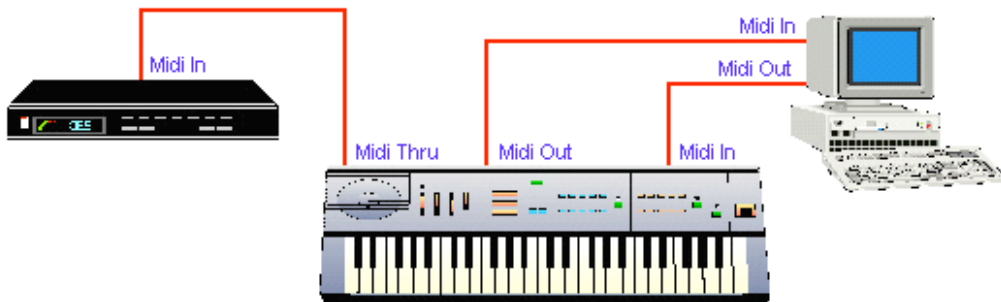


Figure 3.6: MIDI setup including MIDI THRU connectivity.

A combination of audio and MIDI devices is shown in Figure 3.7 below. It features a laptop which is connected to a keyboard controller via the USB port, and an audio interface which is connected to an amplifier and monitor speakers

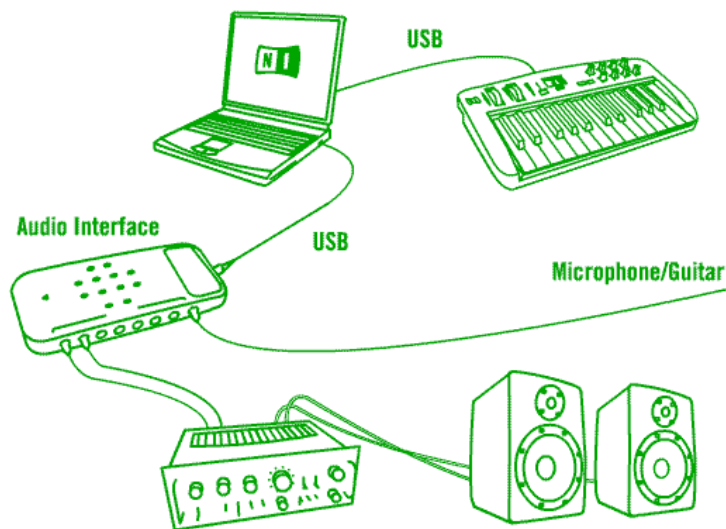


Figure 3.7: MIDI setup including audio connectivity.