1 Yamaha Keyboards – MIDI Reference

Table of contents

Web site: http://www.jososoft.dk/yamaha
E-mail: js@jososoft.dk

1.1 Introduction

This reference holds a number of articles, which gives an initial understanding of MIDI (Musical Instrument Digital Interface). This understanding will help when creating and editing MIDI and Yamaha keyboard style files.

This reference is in now way meant to be a complete guide to MIDI. For this purpose please check the sources listed at http://www.jososoft.dk/yamaha/music_2.htm

Please mail me (js@jososoft.dk) for questions, comments and suggestions for extending this course. Thank you.

1.2 MIDI - The basic concept

1.2.1 MIDI

MIDI (Musical Instrument Digital Interface) is a communication protocol by which computers and musical instruments can communicate. With MIDI computers, music instruments, sound cards etc. can send and receive instructions from and to each other.

1.2.2 History

MIDI was born in the early 1980's when electronic instrument makers, primarily in the US and Japan, recognized that their instruments must be able to talk to one another. After the details were worked out, manufacturers soon began to include electronic circuitry in their equipment that allowed them to understand the instructions MIDI used.

Before long, nearly every instrument maker in the world had adopted the standard, and though there have been refinements and modifications to MIDI along the way, even the earliest MIDI instruments are still capable enough to be used today. Since its adoption, MIDI has dramatically changed the way music is created, performed and recorded.

1.2.3 What is MIDI

MIDI is a universally accepted standard for communicating information about a musical performance by digital means. It has both a hardware and software component, and though it could be used for sending information about many other things, such as the control of lighting in a theatre, it was originally developed to transmit instructions about music.

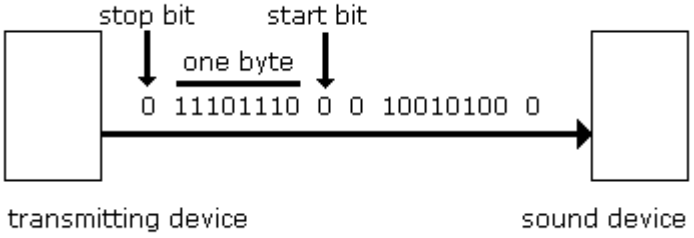
Like music scores MIDI transmits instructions that must be acted on by some device that can make sound. While a piano or guitar player will interpret the music score and produce the sound required, in MIDI it is most likely a synthesizer or drum machine that will react to the information.

MIDI is used both for real-time control of devices; and as files, where all the instructions are saved to a computer file. MIDI does not actually transmit sound electronically. Instead, its information is used in the sound-producing devices (synthesizers, drum machines, sound card etc.), that will create the sound you hear.

1.2.4 How does it Work

A MIDI transmission consists of a series of signals, called bits that pass through a MIDI cable. These signals are electrical impulses that represent the 1's and 0's that make up the language of computers. When the impulses reach their destination (a device) the device interprets them as a series of instructions that usually result in the production of a sound.

The bits in the MIDI transmission move at 31,250 per second, and are transmitted in a serial manner, meaning one after another. Every bit does not represent a different note or event, however. Bits are grouped into strings of 10 to create MIDI messages.



MIDI messages are actually eight bits (= one byte). When messages are being transmitted, two

extra bits, called the stop and start bits, are added to the beginning and end of the byte, hence the 10 bit length of most events.

Some MIDI messages detail specific aspects of a musical performance:

1. what notes should be heard
2. how loud they should be
3. what type of sound (trumpet, drum, flute) should play the notes, etc.

Other MIDI Messages are more general in nature. Together, MIDI messages represent an entire language of musical actions, and can be used to control all details of a complete symphony or a simple hymn.

1.3 MIDI - The structure

1.3.1 What it takes

In order to communicate through MIDI, a device should be able to send and receive MIDI information, though many common devices are created to do primarily one or the other.

1. A sound card must be given instructions that are generated by some other source; it cannot create any MIDI messages on its own.
2. Electronic instruments, known as tone or sound modules, are also only able to respond to messages generated from the "outside."
3. Keyboard controllers are intended for transmitting MIDI data only, and have no way to make sound.

Whatever their capabilities, all MIDI devices must contain a processor, that deciphers and acts upon MIDI messages, as well as physical connections called Ports, for sending and receiving data.

1.3.2 MIDI Channels

One of MIDI's capabilities is its ability to transmit messages to different electronic musical instruments at the same time. Each instrument can distinguish which messages are for it as the messages contain channel information, which acts as an address for the message.

The MIDI channels are not physically separated, i.e., they are not transmitted on separate strands of wire. Rather, the different channel numbers (1-16) are contained in the beginning of the MIDI message, and determine whether an instrument or device will respond to that message.

In this way, messages can be directed to certain devices, while other devices, which might also be receiving the information, will ignore them. Because of this, the user has extensive control over how different instruments react to the information that they receive.

There are certain classes of messages called system messages that don't use a channel, since they are intended for all devices connected to the MIDI chain. Messages that deal with tuning or timing information are in this category.

1.3.3 MIDI Messages

MIDI messages are the language of MIDI; they are the words MIDI uses in a transmission to communicate the information that must pass from a source to a destination. There are many types of MIDI messages, though they all fall into two main categories: channel messages and system messages.

Channel messages are those that carry specific channel information, such as those described above. These include messages such as what note an instrument should play (called a Note Message), and Program Change messages, which tell the instrument what sound it should make while playing the note.

System messages are either intended for all the instruments currently connected to the transmitting device, or are meant to convey information to a specific instrument that is general in nature and doesn't represent specific details of a performance.

Most messages consist of at least two bytes. The first byte is called the status byte, which tells the receiving device what type of message it is. Basically, it identifies the message and prepares the device for a response. MIDI uses the numbers between 128 and 255 for this part of the message.

The status byte can be omitted, if the message status is the same as the previous message status. This is known as "Running Status".

What follows is the actual data the device needs; these bytes are called data bytes. They represent the details of the message; the values the instrument will use to perform its task. MIDI uses the

numbers 0 to 127 for data bytes. Some messages use only one data byte; others need two, while some need none at all.

More about MIDI - Message List (part 10).

1.4 MIDI - Messages

1.4.1 Note On and Note Off Messages

A Note On message is transmitted when a key is pressed on a keyboard, and a Note Off is transmitted when it is released. When a device receives a Note On message, it looks for two data bytes: "which note" and "how loud" it should play it.

The note will continue to play until the Note Off message is received, and it too must contain note and velocity numbers because the device might be playing several notes when the Note Off is received. The Note On / Note Off combination is the most common pair of messages in any MIDI transmission.

1.4.2 Program Change Messages

When a device is turned on, it will load one of its sounds into its RAM and prepare itself to receive note messages. These sounds are permanently stored in the device, and the device operating system tells how to create the required sound.

When the device is ask to load a new sound, it must change the program it is currently running to play notes using the new tone. Hence, the MIDI message that tells the device what sound to make is called a Program Change message.

Program Changes are followed by a single data byte (the Program Number or the "Patch"). Note, that MIDI devices use two different numbering schemes for programs, 0-127 or 1-128. It is important to know which scheme your devices use.

Before General MIDI (GM) there was no consistency in the way manufacturers numbered the sounds in devices. Therefore it was unlikely that music created for one synthesizer would sound correct when performed by another. When using General MIDI, files will sound identical (or nearly so) when played by different instruments. More about General MIDI Voice List (part 6).

In addition to a standardized assignment of program change numbers, General MIDI also defines that Channel 10 are reserved for drum sounds. It also provides a Drum Map, which is the fixed assignment of certain drums sounds too specific MIDI note numbers. More about General MIDI Percussion Key Map (part 7).

1.4.3 Control Change Messages

Control Change messages are used to changes in the status of a physical control, e.g. foot pedals, volume sliders, modulation wheels, and similar peripherals. Controllers are numbered from 0 to 127. These numbers are standardized, for example, controller 10 is used for panning between left and right speakers, while controller 7 is used for volume changes. More about MIDI - Controller List (part 9).

Some control messages act like simple on and off switches. For example, the sustain pedal can only be down or up, so a single status byte can be used to specify which state the pedal is in, and no data bytes are required.

Other controls are continuously changing and are represented by more detailed data. For example, if you move the pitch wheel from its resting position to one extreme, MIDI transmits data representing the wheel's position at numerous points. In this case, the data must be very high in resolution; 2-byte messages are used. This provides a total of 16,384 values.

1.4.4 System Messages

One final category of MIDI messages is called system messages. They all share the characteristic of transmitting information without a channel assignment. As a result, all instruments that receive system messages would act upon them, though one particular type of system message, called system exclusive, is intended for communicating only with a device or devices made by a specific manufacturer.

As all major instrument makers have an ID number (e.g. #67 for Yamaha), a message can be "addressed" to one device and all other receiving instruments will ignore it. For example, all the instructions specifying how a synthesizer makes it sounds could be "dumped" from the device and stored on a computer. This means that a whole new setup of sounds could be sent just before actual note data was transmitted, thereby getting the instrument properly configured.

Other system messages include timing messages, which gives tempo information; and Song Position messages, which indicate where a recorded MIDI sequence should begin playback.

More about MIDI - Message List (part 10).

1.5 MIDI - Files

1.5.1 MIDI File

The Standard MIDI File (SMF) is a file format designed to store the data that a sequencer records and plays. The format stores the standard MIDI messages plus a time-stamp for each message.

The format was designed to be generic so that any sequencer could read or write such a file without losing the most important data, and flexible enough for a particular application to store its own proprietary, "extra" data without disturbing other applications.

1.5.2 What's a Chunk?

Data is always saved within a chunk. There can be many chunks inside of a MIDI file. Each chunk can be a different size (number of bytes in the chunk). A chunk is simply a group of related bytes. Each chunk begins with a 4 character (4 ASCII bytes) ID which tells what "type" of chunk this is.

The next 4 bytes form a 32-bit length (size) of the chunk. All chunks must begin with these two fields (8 bytes), which are referred to as the chunk header. As all data are saved within chunks the format allows proprietary data chunks. An example of this is the additional data chunks in Yamaha keyboard style files (CASM, OTS, and MDB).

NOTE: The Length does not include the 8 byte chunk header. It simply tells you how many bytes of data are in the chunk following this header.

1.5.3 MThd Chunk

The MThd chunk header (with bytes expressed in hex): 4D 54 68 64 00 00 00 06

The first 4 bytes make up the ASCII ID of MThd (the first four bytes are ASCII values for 'M', 'T', 'h' and 'd'). The next 4 bytes tell us that there should be 6 more data bytes in the chunk (and after that we should find the next chunk header or the end of the file).

The first two data bytes tell the Format. There are 3 different formats of MIDI files.

1. Format 0: One single track containing midi data on possibly all 16 midi channels.
2. Format 1: One or more simultaneous (i.e. all start from an assumed time of 0) tracks, perhaps each on a single midi channel.
3. Format 2: One or more sequentially independent single-track patterns.

The next 2 bytes tell how many tracks are stored in the file. Of course, for format type 0, this is always 1. For the other 2 types, there can be numerous tracks.

The last two bytes indicate how many Pulses (i.e. clocks) Per Quarter Note (abbreviated as PPQN) resolution the time-stamps are based upon. For example, if your sequencer has 96 ppqn, this field would be (in hex): 00 60

Data	Meaning
4D 54 68 64	MThd ID
00 00 00 06	Length of the MThd chunk is 6 (it always is).
00 01	The Format type is 1.
00 02	There are 2 MTrk chunks in this file.
00 60	The Pulses Per Quarter Note is 96.

1.5.4 MTrk Chunk

After the MThd chunk, you should find an MTrk chunk, as this is the only other currently defined MIDI chunk. An MTrk chunk contains all of the midi data (with timing bytes).

There will be as many MTrk chunks in the file as the MThd chunk indicates.

The MTrk header begins with the ID of MTrk, followed by the Length (i.e. number of data bytes to read for this track). The Length will likely be different for each track.

1.5.5 Event's Time

A sequencer track contains a series of events: The first event in the track may be to sound a middle C note. The second event may be to sound the E above middle C.

These two events may both happen at the same time. The third event may be to release the middle C note. This event may happen a few musical beats after the first two events.

Each event has a "time" when it must occur, and the events are arranged within a chunk in the order that they occur.

In a MIDI file, an event's "time" precedes the data bytes that make up that event itself i.e. the time-stamp comes before the message.

A given event's time-stamp is referenced from the previous event. For example, if the first event occurs 4 clocks after the start of play, then its "delta-time" is 04. If the next event occurs simultaneously with that first event, its time is 00.

A delta-time is stored as a series of bytes (up to 4 bytes) which is called a variable length quantity.

1.6 General MIDI Voice List

This table lists the 128 voices defined in the MIDI standard.

Notice: The Program Change (or just "P") Numbers are 1-indexed.

P#	Name	P#	Name	P#	Name
1	Acoustic Grand Piano	44	Contrabass	87	Lead 7 (fifths)
2	Bright Acoustic Piano	45	Tremolo Strings	88	Lead 8 (bass + lead)
3	Electric Grand Piano	46	Pizzicato Strings	89	Pad 1 (new age)
4	Honky-tonk Piano	47	Orchestral Harp	90	Pad 2 (warm)
5	Electric Piano 1	48	Timpani	91	Pad 3 (polysynth)
6	Electric Piano 2	49	String Ensemble 1	92	Pad 4 (choir)
7	Harpsichord	50	String Ensemble 2	93	Pad 5 (bowed)
8	Clavi	51	SynthStrings 1	94	Pad 6 (metallic)
9	Celesta	52	SynthStrings 2	95	Pad 7 (halo)
10	Glockenspiel	53	Choir Aahs	96	Pad 8 (sweep)
11	Music Box	54	Voice Oohs	97	FX 1 (rain)
12	Vibraphone	55	Synth Voice	98	FX 2 (soundtrack)
13	Marimba	56	Orchestra Hit	99	FX 3 (crystal)
14	Xylophone	57	Trumpet	100	FX 4 (atmosphere)
15	Tubular Bells	58	Trombone	101	FX 5 (brightness)
16	Dulcimer	59	Tuba	102	FX 6 (goblins)
17	Drawbar Organ	60	Muted Trumpet	103	FX 7 (echoes)
18	Percussive Organ	61	French Horn	104	FX 8 (sci-fi)
19	Rock Organ	62	Brass Section	105	Sitar
20	Church Organ	63	SynthBrass 1	106	Banjo
21	Reed Organ	64	SynthBrass 2	107	Shamisen
22	Accordion	65	Soprano Sax	108	Koto
23	Harmonica	66	Alto Sax	109	Kalimba
24	Tango Accordion	67	Tenor Sax	110	Bag pipe
25	Acoustic Guitar (nylon)	68	Baritone Sax	111	Fiddle
26	Acoustic Guitar (steel)	69	Oboe	112	Shanai
27	Electric Guitar (jazz)	70	English Horn	113	Tinkle Bell
28	Electric Guitar (clean)	71	Bassoon	114	Agogo
29	Electric Guitar (muted)	72	Clarinet	115	Steel Drums
30	Overdriven Guitar	73	Piccolo	116	Woodblock
31	Distortion Guitar	74	Flute	117	Taiko Drum
32	Guitar harmonics	75	Recorder	118	Melodic Tom
33	Acoustic Bass	76	Pan Flute	119	Synth Drum
34	Electric Bass (finger)	77	Blown Bottle	120	Reverse Cymbal

35	Electric Bass (pick)	78	Shakuhachi	121	Guitar Fret Noise
36	Fretless Bass	79	Whistle	122	Breath Noise
37	Slap Bass 1	80	Ocarina	123	Seashore
38	Slap Bass 2	81	Lead 1 (square)	124	Bird Tweet
39	Synth Bass 1	82	Lead 2 (sawtooth)	125	Telephone Ring
40	Synth Bass 2	83	Lead 3 (calliope)	126	Helicopter
41	Violin	84	Lead 4 (chiff)	127	Applause
42	Viola	85	Lead 5 (charang)	128	Gunshot
43	Cello	86	Lead 6 (voice)		

1.7 General MIDI Percussion Key Map

This table lists the percussive sounds in the MIDI standard.
MIDI channel 10 is reserved for percussion. Each key map to a specific percussive sound.
Notice: The Key numbers are 1-indexed.

Key	Nt	Sound	Key	Nt	Sound	Key	Nt	Sound
35	C	Acoustic Bass Drum	51	E	Ride Cymbal 1	67	G#	High Agogo
36	C#	Bass Drum 1	52	F	Chinese Cymbal	68	A	Low Agogo
37	D	Side Stick	53	F#	Ride Bell	69	A#	Cabasa
38	D#	Acoustic Snare	54	G	Tambourine	70	B	Maracas
39	E	Hand Clap	55	G#	Splash Cymbal	71	C	Short Whistle
40	F	Electric Snare	56	A	Cowbell	72	C#	Long Whistle
41	F#	Low Floor Tom	57	A#	Crash Cymbal 2	73	D	Short Guiro
42	G	Closed Hi Hat	58	B	Vibraslap	74	D#	Long Guiro
43	G#	High Floor Tom	59	C	Ride Cymbal 2	75	E	Claves
44	A	Pedal Hi-Hat	60	C#	Hi Bongo	76	F	Hi Wood Block
45	A#	Low Tom	61	D	Low Bongo	77	F#	Low Wood Block
46	B	Open Hi-Hat	62	D#	Mute Hi Conga	78	G	Mute Cuica
47	C	Low-Mid Tom	63	E	Open Hi Conga	79	G#	Open Cuica
48	C#	Hi Mid Tom	64	F	Low Conga	80	A	Mute Triangle
49	D	Crash Cymbal 1	65	F#	High Timbale	81	A#	Open Triangle
50	D#	High Tom	66	G	Low Timbale			

1.8 General MIDI Voices - Natural Range

This table lists the Natural Range of the General MIDI Voices.
Notice: The Voice Numbers are 1-indexed.

#	Note Range	Note Range	Voice	#	Note Range	Note Range	Voice
1	21:108	A-1 - C7	Acoustic Grand Piano	65	56:89	G#2 - F5	Soprano Sax
2	21:108	A-1 - C7	Bright Acoustic Piano	66	52:88	E2 - E5	Alto Sax
3	21:108	A-1 - C7	Electric Grand Piano	67	46:77	A#1 - F4	Tenor Sax
4	21:108	A-1 - C7	Honky-tonk Piano	68	40:74	E1 - D4	Baritone Sax
5	21:108	A-1 - C7	Rhodes Piano	69	58:93	A#2 - A5	Oboe
6	21:108	A-1 - C7	Chorus Piano	70	58:93	A#2 - A5	English Horn
7	29:89	F0 - F5	Harpsichord	71	34:75	A#0 - D#4	Bassoon
8	29:89	F0 - F5	Clavinet	72	50:106	D2 - A#6	Clarinet
9	60:108	C3 - C7	Celesta	73	74:110	D4 - D7	Piccolo
10	79:108	G4 - C7	Glockenspiel	74	60:98	C3 - D6	Flute
11	79:108	G4 - C7	Music Box	75	60:98	C3 - D6	Recorder
12	53:89	F2 - F5	Vibraphone	76	60:98	C3 - D6	Pan Flute
13	36:96	C1 - C6	Marimba	77	60:98	C3 - D6	Bottle Blow

14	79:108	G4 - C7	Xylophone	78	60:98	C3 - D6	Shakuhachi
15	72:91	C4 - G5	Tubular Bells	79	60:98	C3 - D6	Whistle
16	40:76	E1 - E4	Dulcimer	80	60:98	C3 - D6	Ocarina
17	36:96	C1 - C6	Hammond Organ	81	0:127	C-2 - G8	Lead 1 (square)
18	36:96	C1 - C6	Percussive Organ	82	0:127	C-2 - G8	Lead 2 (saw tooth)
19	36:96	C1 - C6	Rock Organ	83	0:127	C-2 - G8	Lead 3 (calliope lead)
20	36:96	C1 - C6	Church Organ	84	0:127	C-2 - G8	Lead 4 (chiff lead)
21	36:96	C1 - C6	Reed Organ	85	0:127	C-2 - G8	Lead 5 (charang)
22	36:96	C1 - C6	Accordion	86	0:127	C-2 - G8	Lead 6 (voice)
23	36:96	C1 - C6	Harmonica	87	0:127	C-2 - G8	Lead 7 (fifths)
24	36:96	C1 - C6	Tango Accordion	88	0:127	C-2 - G8	Lead 8 (bass + lead)
25	40:76	E1 - E4	Acoustic Guitar (nylon)	89	0:127	C-2 - G8	Pad 1 (new age)
26	40:76	E1 - E4	Acoustic Guitar (steel)	90	0:127	C-2 - G8	Pad 2 (warm)
27	40:76	E1 - E4	Electric Guitar (jazz)	91	0:127	C-2 - G8	Pad 3 (poly synth)
28	40:76	E1 - E4	Electric Guitar (clean)	92	0:127	C-2 - G8	Pad 4 (choir)
29	40:76	E1 - E4	Electric Guitar (muted)	93	0:127	C-2 - G8	Pad 5 (bowed)
30	40:76	E1 - E4	Overdriven Guitar	94	0:127	C-2 - G8	Pad 6 (metallic)
31	40:76	E1 - E4	Distortion Guitar	95	0:127	C-2 - G8	Pad 7 (halo)
32	40:76	E1 - E4	Guitar Harmonics	96	0:127	C-2 - G8	Pad 8 (sweep)
33	24:60	C0 - C3	Acoustic Bass	97	0:127	C-2 - G8	FX 1 (rain)
34	24:60	C0 - C3	Electric Bass (finger)	98	0:127	C-2 - G8	FX 2 (sound track)
35	24:60	C0 - C3	Electric Bass (pick)	99	0:127	C-2 - G8	FX 3 (crystal)
36	24:60	C0 - C3	Fretless Bass	100	0:127	C-2 - G8	FX 4 (atmosphere)
37	24:60	C0 - C3	Slap Bass 1	101	0:127	C-2 - G8	FX 5 (bright)
38	24:60	C0 - C3	Slap Bass 2	102	0:127	C-2 - G8	FX 6 (goblins)
39	24:60	C0 - C3	Synth Bass 1	103	0:127	C-2 - G8	FX 7 (echoes)
40	24:60	C0 - C3	Synth Bass 2	104	0:127	C-2 - G8	FX 8 (sci-fi)
41	55:105	G2 - A6	Violin	105	0:127	C-2 - G8	Sitar
42	48:88	C2 - E5	Viola	106	48:69	C2 - A3	Banjo
43	36:84	C1 - C5	Cello	107	0:127	C-2 - G8	Shamisen
44	24:60	C0 - C3	Contra Bass	108	0:127	C-2 - G8	Koto
45	36:105	C1 - A6	Tremolo Strings	109	0:127	C-2 - G8	Kalimba
46	36:105	C1 - A6	Pizzicato Strings	110	0:127	C-2 - G8	Bagpipe
47	23:104	B-1 - G#6	Orchestral Harp	111	0:127	C-2 - G8	Fiddle
48	36:57	C1 - A2	Timpani	112	0:127	C-2 - G8	Shanai
49	36:105	C1 - A6	String Ensemble 1	113	0:127	C-2 - G8	Tinkle Bell
50	36:105	C1 - A6	String Ensemble 2	114	0:127	C-2 - G8	Agogo
51	36:105	C1 - A6	Synth Strings 1	115	0:127	C-2 - G8	Steel Drums
52	36:105	C1 - A6	Synth Strings 2	116	0:127	C-2 - G8	Wood block
53	41:84	F1 - C5	Choir Aahs	117	0:127	C-2 - G8	Taiko Drum
54	41:84	F1 - C5	Voice Oohs	118	0:127	C-2 - G8	Melodic Tom
55	41:84	F1 - C5	Synth Voice	119	0:127	C-2 - G8	Synth Drum
56	24:108	C0 - C7	Orchestra Hit	120	0:127	C-2 - G8	Reverse Cymbal
57	54:86	F#2 - D5	Trumpet	121	0:127	C-2 - G8	Guitar Fret Noise
58	40:77	E1 - F4	Trombone	122	0:127	C-2 - G8	Breath Noise
59	19:72	G-1 - C4	Tuba	123	0:127	C-2 - G8	Seashore
60	54:86	F#2 - D5	Muted Trumpet	124	0:127	C-2 - G8	Bird Tweet
61	38:60	D1 - C3	French Horn	125	0:127	C-2 - G8	Telephone Ringing
62	38:86	D1 - D6	Brass Section	126	0:127	C-2 - G8	Helicopter
63	38:86	D1 - D6	Synth Brass 1	127	0:127	C-2 - G8	Applause
64	38:86	D1 - D6	Synth Brass 2	128	0:127	C-2 - G8	Gunshot

1.9 MIDI - Controller List

This page describes the common types of MIDI Messages.

1. Controller numbers 0 - 31 (Dec) are continuous: Coarse / MSB (most significant byte)
2. Controller numbers 32 - 63 (Dec) are continuous: Fine / LSB (least significant byte)
3. Controller numbers 64 - 97 (Dec) are switches

Name	Hex	Dec	Name	Hex	Dec
Bank Select MSB	00	0	Sound Variation	46	70
Mod Wheel (coarse)	01	1	Sound Timbre	47	71
Breath Controller (coarse)	02	2	Sound Release Time	48	72
Foot Controller (coarse)	04	4	Sound Attack Time	49	73
Portamento Time (coarse)	05	5	Sound Brightness	4A	74
Data Entry (coarse)	06	6	Sound Control 6	4B	75
Volume (coarse)	07	7	Sound Control 7	4C	76
Balance (coarse)	08	8	Sound Control 8	4D	77
Pan (coarse)	0A	10	Sound Control 9	4E	78
Expression Controller (coarse)	0B	11	Sound Control 10	4F	79
Effect Control 1 (coarse)	0C	12	General Purpose 5	50	80
Effect Control 2 (coarse)	0D	13	General Purpose 6	51	81
General Purpose 1	10	16	General Purpose 7	52	82
General Purpose 2	11	17	General Purpose 8	53	83
General Purpose 3	12	18	Ext Effects Depth	5B	91
General Purpose 4	13	19	Tremolo Depth	5C	92
Bank Select LSB	20	32	Chorus Depth	5D	93
Mod Wheel (fine)	21	33	Detune Depth (Celeste Depth)	5E	94
Breath Controller (fine)	22	34	Phaser Depth	5F	95
Foot Controller (fine)	24	36	Data Increment (Data Entry +1)	60	96
Portamento Time (fine)	25	37	Data Decrement (Data Entry -1)	61	97
Data Entry (fine)	26	38	Non-Registered Param LSB	62	98
Volume (fine)	27	39	Non-Registered Param MSB	63	99
Balance (fine)	28	40	Registered Param LSB	64	100
Pan (fine)	2A	42	Registered Param MSB	65	101
Expression Controller (fine)	2B	43	All Sound Off	78	120
Effect Control 1 (fine)	2C	44	Reset All Controllers	79	121
Effect Control 2 (fine)	2D	45	Local Control	7A	122
Sustain	40	64	All Notes Off	7B	123
Portamento	41	65	Omni Mode Off	7C	124
Sustenuto	42	66	Omni Mode On	7D	125
Soft Pedal	43	67	Mono Mode On	7E	126
Hold 2	45	69	Poly Mode On	7F	127

1.10 MIDI - Message List

These tables describe the common types of MIDI Messages.

1. Channel Messages
2. System Messages
3. Real-time Messages
4. Meta Messages

1.10.1 Channel Messages

These messages are addressed to a specific MIDI channel.

In the following table, an "x" specifies any MIDI channel (0 - 15). Thus "status byte 8x" means any value from 80 to 8F.

Note that some message types have two data bytes; others have only one.

Type	Status (Hex)	Data
Note off	8x	note, velocity
Note on	9x	note, velocity (velocity 0 = note off)
Polyphonic pressure	Ax	note, value
Controller change	Bx	controller (more in part 9), value
Program change	Cx	program (more in part 6)
Channel pressure	Dx	value
Pitch bend	Ex	value, value (two bytes). However, many devices accept a single data byte (the MSB byte)

1.10.2 System Messages

System messages are intended for the whole MIDI system - not channel specific.

Note that any non-real-time status byte ends a System Exclusive message; F7 (EOX) is not required at the end of a SysEx message.

Real-time status bytes may appear any time in the MIDI data stream, including in the middle of a SysEx message.

Type	Status (Hex)	Data
System Exclusive	F0	data, then EOX or any status byte. A few examples: ▪ GM System On = F0 7E 7F 09 01 F7 ▪ XG Systen On = F0 43 10 4C 00 00 7E 00 F7 ▪ XG Works On = F0 43 76 1A 10 01 01 01 01 01 01 01 01 F7 ▪ Yamaha = F0 43 73 39 F1 00 46 00 F7 ▪ XG Reset = F0 43 10 4C 00 00 7F 00 00 F7
Time Code	F1	one byte
Song Position Pointer	F2	two bytes: lsb msb
Song Select	F3	one byte: song number 0 - 127
(undefined)	F4	
(undefined)	F5	
Tune Request	F6	no data
EOX (End of System Exclusive)	F7	

1.10.3 Real-time Messages

Real-time messages are not associated with any one MIDI channel.

They can appear in the MIDI data stream at any time.

These messages consist of a single status byte; they have no data bytes.

Type	Status (Hex)
Clock	F8
(undefined)	F9
Start	FA
Continue	FB
Stop	FC
(undefined)	FD
Active Sensing	FE
System Reset	FF

1.10.4 Meta Messages

Meta messages are not associated with any one MIDI channel.

They can appear in the MIDI data stream at any time (except for Time Signature and End of Track).

However it is recommended to insert most of the messages in the beginning of the stream/file.

Note that Meta messages are not for real-time use; but for use in MIDI files.

1. byte: FF = status byte (always FF)
2. byte: Message (event) type
3. byte: Length of the remaining data bytes
4. and following bytes: The data itself.

Type	Status (FF) & Data (Hex)	Comments
Sequence Number	FF 00 02 ss ss	Specifies the number of a sequence. The two data bytes ss ss, are that number which corresponds to the MIDI Cue message.
Text	FF 01 len text	Any amount of text (amount of bytes = len) for any purpose. This event is primarily used to add "comments" to a MIDI file which a program would be expected to ignore when loading that file.
Copyright	FF 02 len text	A copyright message (i.e. text).
Sequence/ Track Name	FF 03 len text	The name of the sequence or track (i.e. text).
Instrument	FF 04 len text	The name of the instrument that the track plays (i.e. text).
Lyric	FF 05 len text	A song lyric (i.e. text) which occurs on a given beat. A single Lyric MetaEvent should contain only one syllable.
Marker	FF 06 len text	A marker (i.e. text) which occurs on a given beat. Marker events might be used to denote a loop start and loop end (i.e. where the sequence loops back to a previous event).
Cue Point	FF 07 len text	A cue point (i.e. text) which occurs on a given beat. A Cue Point might be used to denote where a WAVE (i.e. sampled sound) file starts playing, where the text would be the Wave's filename.
MIDI Channel	FF 20 01 cc	Specifies to which MIDI Channel any subsequent MetaEvent or System Exclusive events are associated. The data byte cc, is the MIDI channel, where 0 would be the first channel.
MIDI Port	FF 21 01 pp	Specifies out of which MIDI Port (i.e. buss) the MIDI events in the MIDI track go. The data byte pp, is the port number, where 0 would be the first MIDI buss in the system.
End of Track	FF 2F 00	This message is NOT optional. It must be the last event in every MIDI track. It's used as a definitive marking of the end of a MIDI track. Only 1 per MIDI track.
Tempo	FF 51 03 tt tt tt	Indicates a tempo change. The 3 data bytes of tt tt tt are the tempo in microseconds per quarter note. In other words, the microsecond tempo value tells you how long each one of your sequencer's "quarter notes" should be. If tt tt tt = 07 A1 20, then each quarter note should be 07 A1 20 (or 500,000) microseconds long. (Check with your Windows Calculator in Scientific Mode). Normally, musicians express tempo as "the amount of quarter notes in

		every minute (i.e. time period)". This is the opposite of the way that the MIDI file format expresses it. To convert the MIDI file format's tempo to BPM (beats per minute): BPM = 60,000,000/(tt tt tt) For example, a tempo of 120 BPM = 07 A1 20 microseconds per quarter note. NOTE: If no Tempo is defined at all, the default value 120 BPM is assumed.
SMPTE Offset	FF 54 05 hr mn se fr ff	Designates the SMPTE start time (hours, minutes, secs, frames, sub frames) of the MIDI track.
Time Signature	FF 58 04 nn dd cc bb	Time signature is expressed as 4 numbers. ▪ nn and dd represent the "numerator" and "denominator" of the signature as notated on sheet music. The denominator is a negative power of 2: 2 = quarter note, 3 = eighth, etc. ▪ cc expresses the number of MIDI clocks in a metronome click. ▪ bb expresses the number of notated 32nd notes in a MIDI quarter note (24 MIDI clocks). This event allows a program to relate what MIDI thinks of as a quarter, to something entirely different. For example, 6/8 time with a metronome click every 3 eighth notes and 24 clocks per quarter note would be the following event: FF 58 04 06 03 18 08 NOTE: If no Time Signature is defined, the default value 4/4 is assumed. Time Signature can only be redefined at measure beginnings.
Key Signature	FF 59 02 sf mi	▪ sf = -7 for 7 flats, -1 for 1 flat, etc, 0 for key of c, 1 for 1 sharp, etc. ▪ mi = 0 for major, 1 for minor
Proprietary Event	FF 7F len data	This can be used by a program to store proprietary data. The first byte(s) should be a unique ID of some sort so that programs can identity whether the event belongs to it, or to some other program. A 4 character (i.e. ascii) ID is recommended for such.

1.11 MIDI - Hex Numbers

This appendix describes Hex numbers and how to convert to/from normal numbers.

The normally used number system is based at 10.

Last digit in a number represents number of "1"; the second last digit represents the number of "10"; the third last digit represents number of "100" (10 * 10) and so on.

35 = (3 * 10) + (5 * 1)
74 = (7 * 10) + (4 * 1) = 74
234 = (2 * 10 * 10) + (3 * 10) + (4 * 1)

Hex values are based at the number 16.

Last digit in a hex number represents number of "1"; the second last digit represents the number of "16"; the third last digit represents number of "256" (16 * 16) and so on.

Hex 35 = (3 * 16) + (5 * 1) = 53
Hex 74 = (7 * 16) + (4 * 1) = 116
Hex 234 = (2 * 16 * 16) + (3 * 16) + (4 * 1) = 564

The basic digits (Notice: A = 10; B = 11 and so on):

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hex values are often written in groups of two characters (from 00 to FF):

Hex 3F A7 = (3 * 16 * 16 * 16) + (15 * 16 * 16) + (10 * 16) + (7 * 1) = 16295

Tip: Use your Windows Calculator in scientific mode to convert to/from Hex numbers.

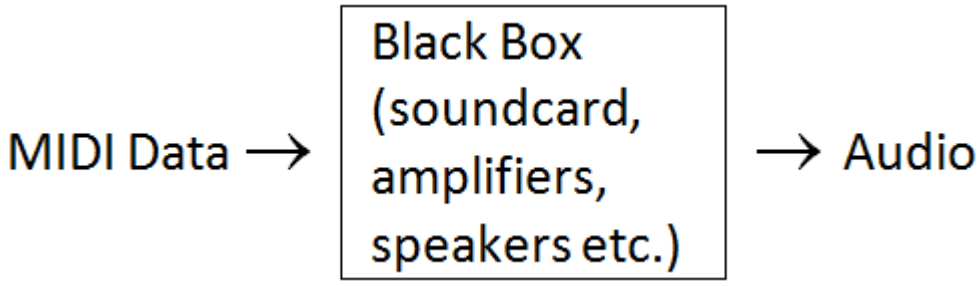
1.12 Audio to MIDI

This article describes why audio files cannot be successfully converted into MIDI files.

1.12.1 MIDI vs. Audio files

MIDI files holds instructions to sound cards in keyboard or computers. The MIDI data is so to speak the "recipe for the music", just like a music sheet is.

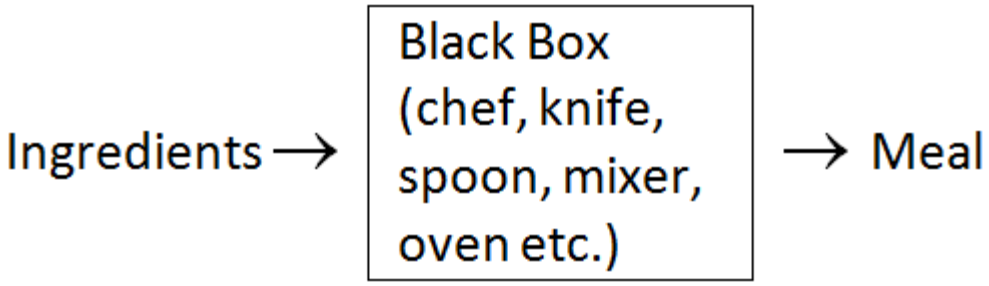
Audio files are samplings of music. Audio data is so to speak "the music". The sound card is the "black box" which reads the recipe for the music; and then creates the audible music.



It has often been discussed if it was possible to reverse the process: Create MIDI data from Audio data. To some degree it is possible. But the limits are quite constricting: Only single instrument MIDI data can be created from audio data.

Complex audio data (and audio data mostly is quite complex) can NOT be turned in usable MIDI data. For details read "Convert from music WAV-MP3 to MIDI" (http://www.skytopia.com/project/articles/mp3-to-midi.html)

To demonstrate this let us look at some analogies: "A rather simple sandwich" and "A rather complex dish". Both cases starts with a recipe to the chef: Which ingredients to use. Next comes the "black box": How to prepare and mix the ingredients; and how to bake or heat the meal.



1.12.2 First analogy: A rather simple sandwich

Consider this recipe from http://food52.com/recipes/1139-mary-s-plain-old-simple-ham-and-cheese-sandwich

Mary's Plain Old Simple Ham and Cheese Sandwich
- 2 slices whole wheat bread
- 4oz thinly sliced ham
- 2 oz sharp cheddar cheese
- 1/2 fresh tomato, sliced
- 1 tbs mayonnaise
- 1 tbs pesto

This rather simple recipe can be transformed into:

I think it is possible to tear the sandwich apart into ingredients. Most can be separated. The cooking process can be reversed; at least mostly. But the toasting of the bread cannot be reversed. If the sandwich was not toasted, it could be transformed "backwards", i.e. into its ingredients.

1.12.3 Second analogy: A rather complex dish

Consider this recipe from http://www.onceuponachef.com/2011/02/beef-stew-with-carrots-potatoes.html

Beef Stew with Carrots & Potatoes

- 3 pounds boneless beef chuck (well-marbled), cut into 1½-inch pieces
- 2 teaspoons salt
- 1 teaspoon freshly ground black pepper
- 3 tablespoons olive oil
- 2 medium yellow onions, cut into 1-inch chunks
- 7 cloves garlic, peeled and smashed
- 2 tablespoons balsamic vinegar
- 1½ tablespoons tomato paste
- ¼ cup all-purpose flour
- 2 cups dry red wine
- 2 cups beef broth
- 2 cups water
- 1 bay leaf
- ½ teaspoon dried thyme
- 1½ teaspoons sugar
- 4 large carrots, peeled and cut into one-inch chunks on a diagonal
- 1 pound small white boiling potatoes (baby yukons), cut in half
- Fresh chopped parsley

This rather complex recipe can be transformed into:

I think it is not possible to tear the dish apart into ingredients. The cooking process cannot be reversed; not at all.

1.12.4 Conclusion

Rather simple audio files can be turned into MIDI data, while complex (e.g. multi instrument) audio files cannot. Just like the sandwich can be transformed into its ingredient; while the dish cannot.

There is no way to get the 2 cups dry red wine back. Sorry, folks...