Musical Instrument Digital Interface (MIDI)

[The MIDI Protocol](#)
[MIDI Computer Interface](#)
[General MIDI](#)
[MIDI Message Data Format](#)
[Who Owns a MIDI File?](#)
[Updates to MIDI Specification 1.0](#)

MIDI is a communications protocol that allows digital instruments to interact with each other and with computers. MIDI has become the primary digital production tool for musicians since its invention in 1983. A MIDI file contains no sounds, just instructions describing the notes played in a performance and related information.

A large percentage of professionals working in new media have a background in the field of music, and many of them had their first creative experience with computers using MIDI. The protocol was initially designed to control digital keyboards, but as soon as computers entered the studio, they were connected to the MIDI chain. Then software became available for recording, printing, and editing musical symbols, just as word processors and graphic design programs proliferated for working with other media types.

Keyboard synthesizer technology made major advances and became very popular in the 1980s. New methods of generating sounds were the focus of considerable research and development. The synthesizer joined the world of widely used musical instruments. One desirable method of creating sounds with synthesizers was to “layer,” or combine the timbres of more than one instrument. A small group of synthesizer design technicians from different manufacturers met in 1983 to discuss a communications protocol to control a number of synthesizers from one keyboard. They developed a method of connecting two synthesizers from competing manufacturers with cables that allowed either instrument to control the other. They called it the Musical Instrument Digital Interface, or MIDI.

The MIDI Protocol

Two synthesizers can communicate using MIDI in the same way that two computers can communicate over modems. The data exchanged between MIDI devices describes the performance of musical notes. MIDI information contains commands that instruct an instrument when to start and stop playing a specific note. Additional information translates the velocity of a keystroke into the volume of a note. MIDI information can be hardware-specific. It can tell a synthesizer to change sounds, which are referred to as instruments, programs, patches, voices, or timbres. Master volume, modulation of tones, and other types of data can be transmitted. MIDI information can start and stop a song, or sequence of events, and identify a location within a song. Computers can edit and store information that defines the sounds that reside in a synthesizer. A distinction may be made between a synthesizer that uses oscillators to electronically create a sound and a sampler that plays back a looped recording of a sound wave. Memory in samplers and sound cards holds a “wave table” of samples, containing short recordings of live instrument sounds.

The basic unit of communication used in MIDI is the byte. Each MIDI command has its own particular byte sequence. The first byte is the status byte, which tells the MIDI device what function to perform. The status byte contains the MIDI channel that is being addressed. MIDI data can flow on 16 different channels simultaneously. Depending on the mode of reception and the channel to which a MIDI unit is set to receive, it will accept or ignore a status byte. The bytes that follow the status byte address the particular channel indicated by the status byte until another status byte is received.

The status byte sends commands such as Note On, Note Off, and Patch Change. Depending on the status byte, a number of different byte patterns will follow. The Note On status byte tells the MIDI device to play a note. This status byte requires a note-number byte to identify the note and a velocity byte to define the volume. These bytes are required to complete the Note On transmission.

A separate Note Off command is sent to stop the note, which is not part of the Note On command. This command also requires the same two additional bytes as the Note On byte.

Another example of a status byte is the Patch Change byte. The additional byte required by this command is the number of the new patch or voice on the synthesizer. It is important to select the desired channel when sending a Patch Change command. Patch Change data is different on every synthesizer. The International MIDI Association (IMA) has set standards, and each manufacturer has an ID number.

The SysEx status byte, which requires at least three additional bytes, can perform a variety of functions. The first additional byte is the manufacturer's ID number, the second is a data format byte, and the third is an end of transmission (EOX) byte.

IN, OUT, and THRU

There are three five-pin ports on a typical MIDI unit for connecting a MIDI interface: IN, OUT, and THRU. The IN port accepts MIDI data that comes to the unit from an external source. These are the MIDI commands that control the instrument. The OUT port sends MIDI data from the unit, such as Note On and Patch Change messages. The THRU port sends an exact copy of the data received at the IN port. There is no change made to the data; however, a brief delay occurs in transmission.

Only three of the five conductors in a MIDI cable are used. The cable is terminated on both ends with a Deutsche Industrie Norm (DIN) plug. Data passes through the cable on pins 1 and 3, and pin 2 is shielded and connected to a common ground. Pins 4 and 5 are not used. A MIDI cable is specially grounded and shielded for efficient data transmission. The length of the cable is limited. The IMA specification allows a maximum cable length of 50 feet. The total length of a MIDI chain is unlimited, as long as no link is longer than 50 feet. Commercially available cables usually range from five to ten feet in length.

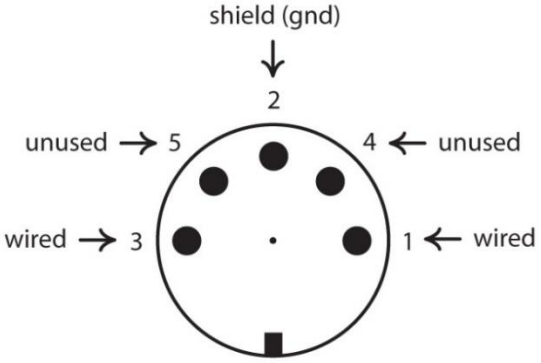


Figure 1—MIDI cable with connectors showing pin-out

MIDI Chains and Loops

A MIDI chain is a series of one-way connections between MIDI equipment. The basic link is a connection between two devices. The MIDI OUT port of one device is connected to the MIDI IN port of the other. A key pressed on the first unit causes both units to sound. A key press on the second unit causes only it to sound. Several instruments may be chained together with a series of one-way links. In this type of setup, the OUT of the first unit is connected to the IN of the second, and the THRU of the second is connected to the IN of a third. If all units are set to receive on the same channel, pressing a key on the first one will cause all units to sound. Pressing a key on any of the other units will make a sound only on that device.

A MIDI loop is a MIDI chain configured for two-way transmission. A single element loop is made of two interconnecting links. The OUT port of the first unit is connected to the IN port of the second, and the OUT port of the second is connected to the IN port of the first. A key pressed on either unit will cause both units to sound, provided they are set to receive on the same channel. A feedback loop does not occur because data going into the second unit from the first is not sent from the OUT port back to the first unit. This is a configuration with two one-way links, not a multilink chain.

MIDI Computer Interface

A special hardware interface is required to connect a computer to a MIDI device because the MIDI data transmission rate is 31.5 Kbps. This data rate is different from any other computer interface rate. Apple Computer and Commodore were the first companies to provide MIDI interface hardware. Roland Corporation later developed an interface for IBM-compatible computers, the MPU-401. Atari designed the ST series computer with MIDI ports built in. A wide range of interfaces is available for all types of systems. Some come with software to handle an entire MIDI setup and route signals on different channels to different devices in the chain. Mark of the Unicorn and Opcode make professional quality interfaces that generate their own time code for synchronization. Interfaces are available that connect through either the parallel port or a USB port on the computer.

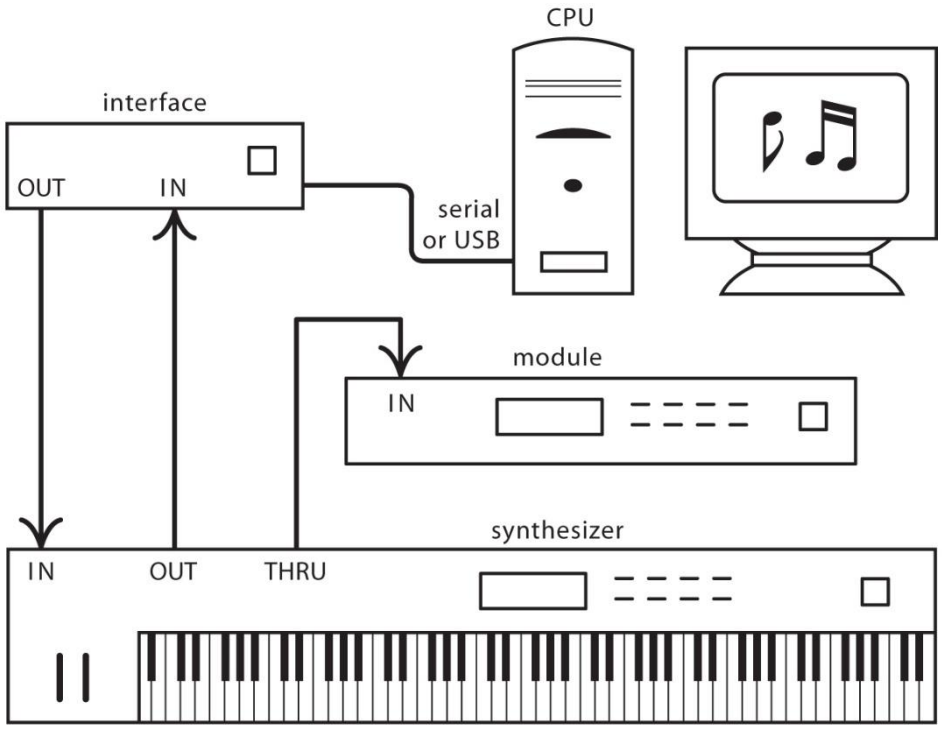


Figure 2—Diagram of MIDI setup with synth, interface, CPU, cables, etc.

Software Applications

An abundance of software applications available serving a variety of functions using the MIDI interface. One of the most widely used is the sequencer, which turns a computer into a multitrack recording studio for MIDI tracks. Sequencers allow the computer to record, store, edit, and replay MIDI data. The data can be saved in the Standard MIDI File (SMF) format as a song and realized by any sound card or synthesizer. There are thousands of MIDI files free for downloading on the World Wide Web. Most sequencers provide extensive editing capabilities as well as synchronization using MIDI Time Code (MTC) or SMPTE Time Code. In recent years, the sequencer has been endowed with multitrack audio recording functions. These programs allow the user to mix and edit live recorded tracks with MIDI tracks in a virtual studio environment, known as a Digital Audio Workstation (DAW). Some of the more popular applications are ProTools from Avid, Digital Performer from MOTU, Cubase from Steinberg, Logic Pro from Apple, Studio One from Presonus, and Ableton Live.

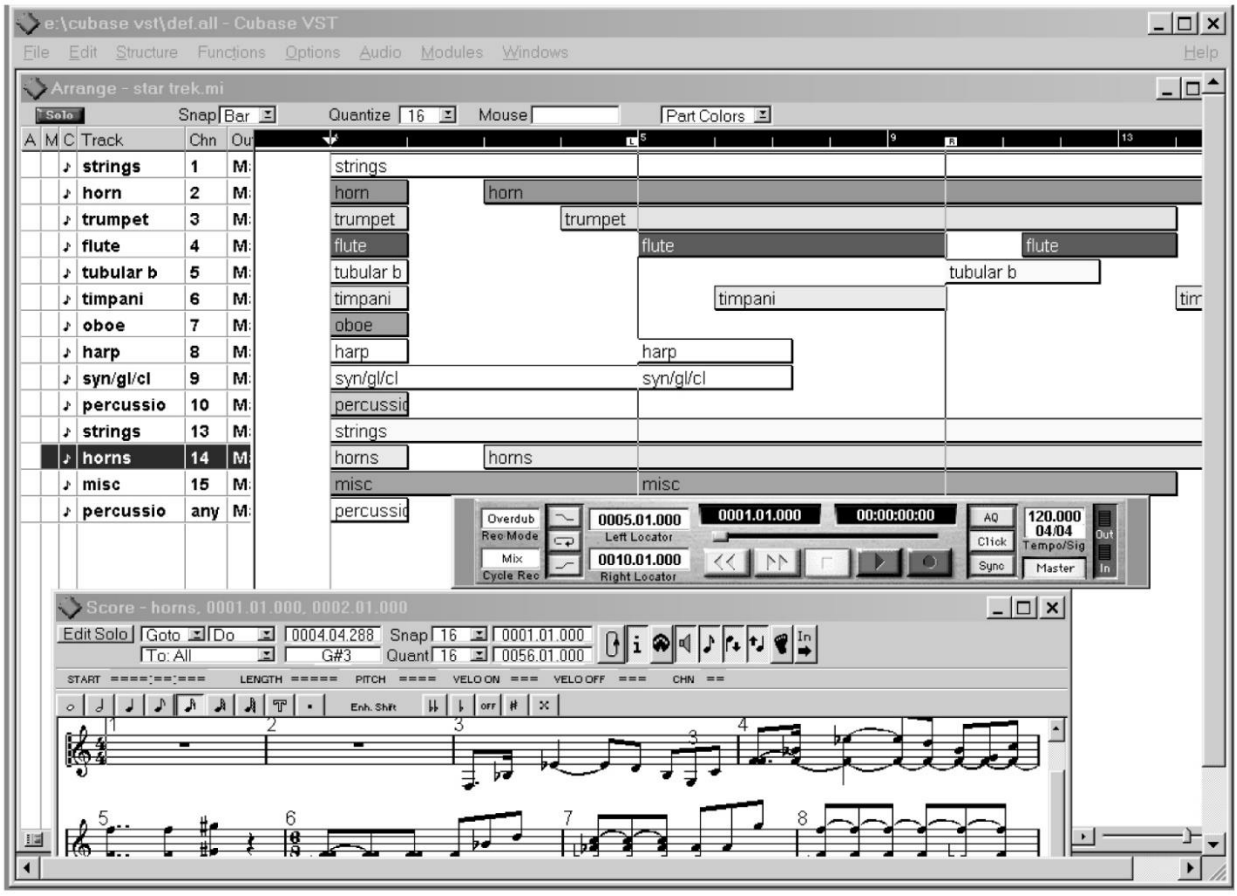


Figure 3—Screen from Cubase

Music notation programs are another popular category of MIDI applications. They display MIDI data as a musical score on the monitor, which can be edited and printed. Notes on the staff can be entered with or without a MIDI interface or keyboard. Most of the music that is composed and published today is rendered in this way. Some of the commonly used notation programs are Finale from Coda, Sibelius

from Avid, Dorico from Steinberg, Notion from Presonus, and the free open source program MuseScore.

Computer-based sample editors and librarians are often used to develop sounds that may be transferred to an instrument after they are edited. Patch librarians allow banks of sounds to be edited, stored on disk, and moved between the computer and the synthesizer via MIDI.

Computers in MIDI Chains

A computer functions in the same way as any other unit in a MIDI chain or loop. Most interfaces have the standard three ports: IN, OUT, and THRU. A computer can serve as a MIDI data driver and supply the MIDI data for the rest of the chain. It can also receive and record MIDI data from other devices.

If the device receiving data from the computer is multitimbral, meaning that it can allocate a different sound to each MIDI channel, data sent on all 16 MIDI channels simultaneously creates an electronic orchestra. A computer-controlled MIDI chain is often used to emulate a recording studio. Scratch tracks for film scores are typically done in this environment.

General MIDI

Compared to audio files, MIDI files are extremely small in size. This is a big advantage when transferring MIDI files over a network like the web. A music track can be embedded in a web page and begin playing after a very brief download. Sound card manufacturers have adopted a specification called “General MIDI” (GM). This standard describes a set of 128 instrument samples that appear in a specific order. The disadvantage for GM files is that the composer has little control over how the MIDI file will sound on playback because the file does not contain that actual sample that is played, just the name of the instrument on each channel.

General MIDI is really just a minor qualification to the MIDI specification that assigns a particular instrumental sound to each MIDI program number, so that an individual musical part plays back on the type of instrument for which it was intended. A well-designed MIDI file includes a Program Change message at the beginning that tells the playback device to switch to the appropriate set of instruments (programs, voices, or patches). Almost all sound cards support General MIDI, but the quality of the sound depends on the quality of the samples stored in the sound card (wave table) and on the speakers used to realize the sounds.

Program number 1 on all GM sound modules is an Acoustic Grand Piano. Patch number 25 is a Nylon String Guitar. The programs are arranged in 16 families of instruments, with each family containing eight instruments. For example, the reed family includes the Saxophone, Oboe, and Clarinet programs. The program number assigned to each instrument in General MIDI is shown in the table below.

A GM sound module that is multitimbral can play MIDI messages on all 16 channels simultaneously, with a different program sounding for each channel. All programs should sound an A440 pitch when they receive the MIDI note number 69.

The Drum Part is sent and received on MIDI channel 10. Each of the MIDI notes triggers a different drum sound on the keyboard. The assignment of drum sounds to MIDI note numbers is also shown in the table below.

The GM standard allows for Program Change messages in a MIDI song file, which applies the correct instrumentation automatically. The specification requires that a GM module be able to respond to the Pitch and Modulation controllers on a synthesizer and to play 24 notes simultaneously with dynamic voice allocation between the 16 channels. This means that the first note played is replaced by the 25th note if more than 24 notes are held at the same time.

The GM specification spells out some global settings. A module should respond to velocity data to control the volume of a note. The pitch wheel bend range should default to +/- 2 semitones. The module also should respond to Channel Pressure as well. It should respond to numbered MIDI controller messages for Modulation (1), Channel Volume (7), Pan (10), Expression (11), Sustain (64), Reset All Controllers (121), and All Notes Off (123). Channel Volume should default to 90, with all other controllers and effects off and the pitch wheel offset at 0. The module should respond to Registered Parameter Numbers that control Pitch Wheel Bend Range (0), Fine Tuning (1), and Coarse Tuning (2). Initial tuning should be the standard A440 reference. A MIDI System Exclusive message can be used to turn a module's General MIDI mode on or off.

In the accompanying tables, "Prog#" refers to the MIDI Program Change number that causes the instrument to be selected. These decimal numbers are shown on a module's display or in a sequencer's Event List. MIDI modules count the first Patch as 0, not 1. The value sent in the Program Change message is actually one less than the program number from the list. A GM module automatically adds a digit when it generates the MIDI Program Change message.

Table of General MIDI Programs

This chart shows the names of all 128 GM instruments and the MIDI Program Change numbers used to select those instruments.

Table of General MIDI Programs

This chart shows the names of all 128 GM instruments and the MIDI Program Change numbers used to select those instruments.

Piano

Prog#	Instrument
1	Acoustic Grand
2	Bright Acoustic
3	Electric Grand
4	Honky-Tonk
5	Electric Piano 1
6	Electric Piano 2
7	Harpsichord
8	Clavinet

Chromatic Percussion

Prog#	Instrument
9	Celesta
10	Glockenspiel
11	Music Box
12	Vibraphone
13	Marimba
14	Xylophone
15	Tubular Bells
16	Dulcimer

Organ

Prog#	Instrument
17	Drawbar Organ
18	Percussive Organ
19	Rock Organ
20	Church Organ
21	Reed Organ
22	Accordion
23	Harmonica
24	Tango Accordion

Guitar

Prog#	Instrument
25	Nylon String Guitar
26	Steel String Guitar
27	Electric Jazz Guitar
28	Electric Clean Guitar
29	Electric Muted Guitar
30	Overdriven Guitar
31	Distortion Guitar
32	Guitar Harmonics

Bass

Prog#	Instrument
33	Acoustic Bass
34	Electric Bass (finger)
35	Electric Bass (pick)
36	Fretless Bass
37	Slap Bass 1
38	Slap Bass 2
39	Synth Bass 1
40	Synth Bass 2

Solo Strings

Prog#	Instrument
41	Violin
42	Viola
43	Cello
44	Contrabass
45	Tremolo Strings
46	Pizzicato Strings
47	Orchestral Strings
48	Timpani

Ensemble

Prog#	Instrument
49	String Ensemble 1
50	String Ensemble 2
51	Synth Strings 1
52	Synth Strings 2
53	Choir Aahs
54	Voice Oohs
55	Synth Voice
56	Orchestra Hit

Brass

Prog#	Instrument
57	Trumpet
58	Trombone
59	Tuba
60	Muted Trumpet
61	French Horn
62	Brass Section
63	Synth Brass 1
64	Synth Brass 2

Reed

Prog#	Instrument
65	Soprano Sax
66	Alto Sax
67	Tenor Sax
68	Baritone Sax
69	Oboe
70	English Horn
71	Bassoon
72	Clarinet

Pipe

Prog#	Instrument
73	Piccolo
74	Flute
75	Recorder
76	Pan Flute
77	Blown Bottle
78	Skakuhachi
79	Whistle
80	Ocarina

Synth Lead

Prog#	Instrument
81	Lead 1 (square)
82	Lead 2 (sawtooth)
83	Lead 3 (calliope)
84	Lead 4 (chiff)
85	Lead 5 (charang)
86	Lead 6 (voice)
87	Lead 7 (fifths)
88	Lead 8 (bass+lead)

Synth Pad

Prog#	Instrument
89	Pad 1 (new age)
90	Pad 2 (warm)
91	Pad 3 (polysynth)
92	Pad 4 (choir)
93	Pad 5 (bowed)
94	Pad 6 (metallic)
95	Pad 7 (halo)
96	Pad 8 (sweep)

Synth Effects

Prog#	Instrument
97	FX 1 (rain)
98	FX 2 (soundtrack)
99	FX 3 (crystal)
100	FX 4 (atmosphere)
101	FX 5 (brightness)
102	FX 6 (goblins)
103	FX 7 (echoes)
104	FX 8 (sci-fi)

Ethnic

Prog#	Instrument
105	Sitar
106	Banjo
107	Shamisen
108	Koto
109	Kalimba
110	Bagpipe
111	Fiddle
112	Shanai

Percussive

Prog#	Instrument
113	Tinkle Bell
114	Agogo
115	Steel Drums
116	Woodblock
117	Taiko Drum
118	Melodic Tom
119	Synth Drum
120	Reverse Cymbal

Sound Effects

Prog#	Instrument
121	Guitar Fret Noise
122	Breath Noise
123	Seashore
124	Bird Tweet
125	Telephone Ring
126	Helicopter
127	Applause
128	Gunshot

MIDI Message Data Format

This chart shows the drum sound assigned to each MIDI note number. Typically, Channel 10 is the default channel for a set of drums.

In the standard MIDI protocol one device is the "transmitter" and another is the "receiver." Messages include "status" bytes and "data" bytes. This arrangement is similar to common computer networking protocols. Unfortunately, there is no provision for handshaking between connected units in the MIDI protocol.

Status bytes and data bytes are easily distinguished. In all status bytes, bit 7 is a 1. All data bytes must contain a 0 in bit 7 and lie in the range between 0 and 127.MIDI applies a logical channel concept. There are 16 logical channels, encoded into bits 0 through 3 of the status bytes of messages for which a channel number is significant.

General MIDI Drum Sounds

This chart shows the drum sound assigned to each MIDI note number. Typically, Channel 10 is the default channel for a set of drums.

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

Voice Messages

In messages with channel numbers, the status byte determines the number of data bytes for a single message. The specification divides these into "voice" and "mode" messages. The mode messages are for control of the logical channels, and the control codes are added onto the data bytes for the parameter message. The voice messages are as follows:

Status Byte	Data Bytes
Note On	2 each; 1 byte pitch, followed by 1 byte velocity
Note Off	2 each; 1 byte pitch, followed by 1 byte velocity
Key Pressure	2 each; 1 byte pitch, 1 byte pressure (after-touch)
Parameter	2 each; 1 byte parameter number, 1 byte setting
Program	1 byte; program selection
Channel Pressure	1 byte; channel pressure (after-touch)
Pitch Wheel	2 bytes; a 14-bit value, least significant 7 bits first

For all of these messages, a convention called the "running status byte" may be used. If the transmitter wishes to send another message of the same type on the same channel under the same status byte, the status byte need not be resent.

A Note On message with a velocity of zero is synonymous with a Note Off message. Combined with the previous feature, this allows long strings of notes to be sent without repeating status bytes. The "zero velocity Note On" feature is frequently used. The pitch bytes of notes correspond to the half steps on a keyboard, with middle C = 60.

The velocity bytes for velocity sensing keyboards represent a logarithmic scale. Non-velocity sensing devices send a velocity of 64. The pitch wheel value is an absolute setting. The receiver determines the increments. The default value corresponds to a centered pitch wheel (unmodified notes).

Parameter messages are used to set controller dials, the purpose of which is left to the given device, except as noted below. The first data bytes correspond to the following controllers:

Data Byte	Parameter Governed
0-31	Continuous controllers 0–31, most significant byte
32-63	Continuous controllers 0–31, least significant byte
64-95	On/off switches
96–121	Unspecified, reserved for future
122–127	The "channel mode" messages

The second data byte contains the seven-bit setting for the controller. The switches have data byte 0 set to OFF, 127 set to ON, with 1 through 126 undefined. If a controller only needs seven bits of resolution, it uses the most significant byte. If both are needed, the order is specified as most significant followed by least significant. With a 14-bit controller, it is legal to send only the least significant byte if the most significant doesn't need to be changed. Controller number 1 is standardized to be the modulation wheel.

MIDI Mode Messages

These messages begin with status bytes, followed by data bytes 122 through 127. The data bytes function as further data for a group of messages that control the combination of voices and channels to be accepted by a receiver. There is an implicit "basic" channel over which a given device is to receive these messages. The receiver ignores mode messages over any other channels, no matter what mode they might be in. The basic channel for a given device may be fixed or set in some manner outside the scope of the MIDI standard.

The meaning of the values 122 through 127 is as follows:

First Data Byte	Second Data Byte	
122	Local control	0 = local control off, 127 = on
123	All notes off	0
124	Omni mode off	0
125	Omni mode on	0
126	Mono mode.	The number of monophonic channels
127	Poly mode	0

Note: 124 through 127 also turn all notes off.

Local control determines whether notes played on an instrument's keyboard are sounded on the instrument or not. With local control off, the host is able to read input data if desired and to send notes to another instrument.

The mode setting messages control what channels and how many voices the receiver recognizes. There is always a basic channel. "Omni" refers to the ability to receive voice messages on all channels. "Mono" and "Poly" refer to whether multiple voices are allowed to sound at once. Unfortunately, the omni on/off state and the mono/poly state interact with each other. There are four possible settings, called "modes," with given numbers in the specification:

Mode 1 **Omni on/poly**—Voice messages are received on all channels and assigned polyphonically. Any notes received are played, up to the maximum capacity.

Mode 2 **Omni on/mono**—A monophonic instrument will receive single notes to play in one voice on all channels.

Mode 3 **Omni off/poly**—A polyphonic instrument will receive voice messages on only the basic channel.

Mode 4 **Omni off/mono**—The "mono" part is a misnomer. To operate in this mode, a receiver is supposed to receive one voice per channel. The number of channels recognized is given by the second data byte, or the maximum number of possible voices if this byte is 0. The set of channels thus defined is a sequential set, starting with the basic channel.

A receiver may ignore any mode that it cannot honor or switch to an alternate mode (typically mode 1). Receivers are supposed to default to mode 1 when they power up. The original 1.0 specification states that power-up conditions are supposed to place a receiver in a state where it will only respond to Note On/Note Off messages, requiring a setting of some sort to enable the other message types. (Current manufacturers default to "Multi" mode for startup, which is similar to Omni on/poly for multitimbral modules.)

System Messages

In system messages, the status bytes and data bytes are used as follows:

Message Purpose	Data Bytes
System Exclusive	Variable length
Song Position	2 bytes; a 14-bit value, least significant byte first
Song Select	1 byte; a song number
Tune Request	0
EOX (terminator)	0

Song Position and Song Select are for controlling sequencers. The Song Position is measured in beats, which count every six MIDI clock pulses. These messages determine what is to be played on receipt of a start real-time message. The Tune Request tells analog synthesizers to tune their oscillators.

The System Exclusive message is intended for manufacturers to insert any specific messages that apply to their own product. The following data bytes lie in the range of 0 to 127. The System Exclusive is to

be terminated by the EOX byte. The first data byte is to be a manufacturer's ID, assigned by the MIDI standards committee. The terminator byte is optional. A System Exclusive may also be terminated by the status byte of the next message.

Common MIDI Manufacturer ID Numbers

"American Group"

1	Sequential Circuits (originator of spec)
4	Moog
5	Passport Designs
6	Lexicon
7	Kurzweill
8	Fender
10	Oberheim
11	Apple Computers
15	JL Cooper
18	Emu Systems
21	Orban
31	Voce

"European Group"

24	Hohner
29	PPG
39	Soundcraft

"Japanese Group"

40	Kawai
41	Roland
42	Korg
43	Yamaha
44	Casio

Real-time Messages

These messages are called "real-time" messages because they may be sent anytime anywhere. This includes between data bytes of other messages. A receiver is supposed to be able to receive and process (or ignore) these messages and resume collection of the remaining data bytes for the message that was in progress. Real-time messages do not affect any running status byte in effect.

All of these messages are followed by no data bytes to prevent them from being interrupted. The real-time messages are as follows:

- Timing Clock
- Start
- Continue
- Stop
- Active Sensing
- System Reset

The Timing Clock message is sent at the rate of 24 clocks per quarter note and is used to synchronize devices, particularly drum machines. Start, Continue, and Stop are for the control of sequencers and drum machines. The Continue message causes a device to pick up at the next clock mark.

The Active Sensing byte is to be sent once at least every 300 milliseconds, if it is used. Its purpose is to implement a time-out mechanism for a receiver to revert to a default state. A receiver is to operate normally until it receives one, activating the time-out mechanism from the receipt of the first Active Sensing byte.

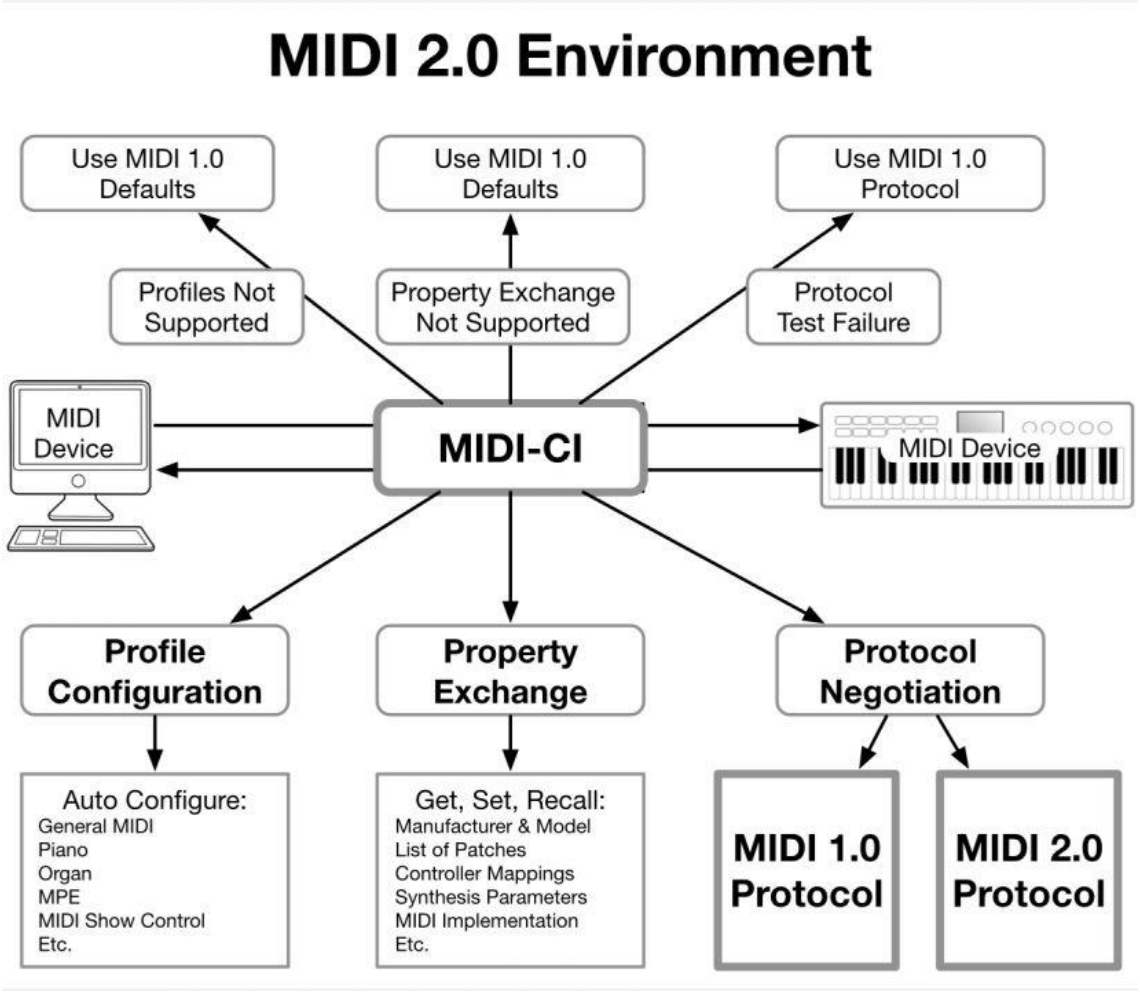
The System Reset initializes to power-up conditions. It should be used sparingly and never sent automatically on power up.

Who Owns a MIDI File?

As with any performance of a musical composition, the composer controls rights to the content and the performer controls rights to a particular performance of the work. Although a piece may be in the public domain, the file is still the property of the person who created it. It is legal to use a MIDI song on your web site as background music, providing the tune is in the public domain or permission has been secured from the copyright owner. Permission should also be secured from the creator of the MIDI file itself since the creation is akin to a performance. There are no restrictions on MIDI files based on public domain material other than those claimed by the creator of the file.

Updates to MIDI specification 1.0

Until 2020, the original MIDI specification changed very little, although some of the initial status bytes were not originally defined. The architecture of MIDI did not allow for expansion without a complete redesign of the system. Any new design would need to be backwards- compatible and operate with legacy MIDI hardware. The only major enhancement to General MIDI was the capacity to send sample data along with a Standard MIDI File as a downloadable sound (DLS). Not all sound cards or modules were equipped to handle this information. Below is a diagram from the MMA showing the effect of Compatibility Inquiry made in 2020, which is at the heart of MIDI 2.0.



MIDI 2.0 advances the specification, while retaining backward compatibility with MIDI 1.0 gear and software already in use. The advances in MIDI 2.0 are explained below.

MIDI 2.0 Allows Two-way MIDI Communication

MIDI 1.0 messages went in one direction: from a transmitter to a receiver. MIDI 2.0 is bi-directional and changes MIDI communication from a monologue to a dialog. For example, with the new MIDI-CI (Capability Inquiry) messages, MIDI 2.0 devices can talk to each other, and auto-configure themselves to work together. They can also exchange information on functionality, which is key to backward compatibility—MIDI 2.0 gear can find out if a device supports MIDI 2.0, and then simply communicate using MIDI 1.0 if it does not.

Higher Resolution, More Controllers and Better Timing

To deliver a higher level of nuanced musical and artistic expressiveness, MIDI 2.0 re-imagines the role of performance controllers, the aspect of MIDI that translates human performance gestures to data computers can understand. Controllers are now easier to use, and there are more of them: over 32,000 controllers, including controls for individual notes. Enhanced, 32-bit resolution gives controls a smooth,

continuous, "analog" feel. New Note-On options were added for articulation control and precise note pitch. In addition, dynamic response (velocity) was upgraded. Major timing improvements in MIDI 2.0 can apply to MIDI 1.0 devices. Some MIDI 1.0 gear can even retrofit certain MIDI 2.0 features.

Profile Configuration

MIDI gear can now have Profiles able to dynamically configure a device for a particular use case. If a control surface queries a device with a "mixer" Profile, then the controls will map to faders, panpots, and other mixer parameters. But with a "drawbar organ" Profile, that same control surface can map its controls automatically to virtual drawbars and other keyboard parameters—or map to dimmers if the profile is a lighting controller. This saves setup time, improves workflow, and eliminates manual programming.

Property Exchange

While Profiles set up an entire device, Property Exchange messages provide specific, detailed information sharing. These messages can discover, retrieve, and set many properties like preset names, individual parameter settings, and unique functionalities. For example, recording software could display data about a synthesizer onscreen, effectively bringing hardware synths up to the same level of recallability as their software counterparts.

Built for the Future

MIDI 2.0 is the result of a global, decade-long development effort. Unlike MIDI 1.0, which was initially tied to a specific hardware implementation, a new Universal MIDI Packet format makes it easy to implement MIDI 2.0 on any digital transport (like USB or Ethernet). To enable future applications that we cannot envision today, there's ample space reserved for new MIDI messages.

Further development of the MIDI specification, as well as safeguards to ensure future compatibility and growth, will continue to be managed by the MIDI Manufacturers Association working in cooperation with the Association of Musical Electronics Industry (AMEI), the Japanese trade association that oversees the MIDI specification in Japan.

MIDI is intended to continue to serve musicians, DJs, producers, educators, artists, and anyone who creates, performs, learns, and shares music and artistic works in the decades to come.