Standard MIDI-File Format Spec. 1.1, updated

Back

Abstract.

A detailed Specification of the Standard MIDI file format

Table of Contents

Return to CSW2 Home Page

Acknowledgement:

0 - Introduction

This document details the structure of MIDI Files. The purpose of MIDI Files is to provide a way of interchanging time-stamped MIDI data between different programs on the same or different computers. One of the primary design goals is compact representation, which makes it very appropriate for disk-based file format, but which might make it inappropriate for storing in memory for quick access by a sequencer program.

MIDI Files contain one or more MIDI streams, with time information for each event. Song, sequence, and track structures, tempo and time signature information, are all supported. Track names and other descriptive information may be stored with the MIDI data. This format supports multiple tracks and multiple sequences so that if the user of a program which supports multiple tracks intends to move a file to another one, this format can allow that to happen.

The specification defines the 8-bit binary data stream used in the file. The data can be stored in a binary file, nibbilized, 7-bit-ized for efficient MIDI transmission, converted to Hex ASCII, or translated symbolically to a printable text file. This spec addresses what's in the 8-bit stream. It does not address how a MIDI File will be transmitted over MIDI.

1 - Sequences, Tracks, Chunks: File Block Structure

In this document, bit 0 means the least significant bit of a byte, and bit 7 is the most significant.

1.1 - Variable Length Quantity

Some numbers in MIDI Files are represented in a form called VARIABLE-LENGTH QUANTITY. These numbers are represented 7 bits per byte, most significant bits first. All bytes except the last have bit 7 set, and the last byte has bit 7 clear. If the number is between 0 and 127, it is thus represented exactly as one byte.

Some examples of numbers represented as variable-length quantities:

00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

The largest number which is allowed is 0FFFFFFF so that the variable-length representations must fit in 32 bits in a routine to write variable-length numbers. Theoretically, larger numbers are possible, but 2×10^8 96ths of a beat at a fast tempo of 500 beats per minute is four days, long enough for any delta-time!

1.2 - Files

To any file system, a MIDI File is simply a series of 8-bit bytes. On the Macintosh, this byte stream is stored in the data fork of a file (with file type 'MIDI'), or on the Clipboard (with data type 'MIDI'). Most other computers store 8-bit byte streams in files.

1.3 - Chunks

MIDI Files are made up of -chunks-. Each chunk has a 4-character type and a 32-bit length, which is the number of bytes in the chunk. This structure allows future chunk types to be designed which may be easily be ignored if encountered by a program written before the chunk type is introduced. Your programs should EXPECT alien chunks and treat them as if they weren't there.

Each chunk begins with a 4-character ASCII type. It is followed by a 32-bit length, most significant byte first (a length of 6 is stored as 00 00 00 06). This length refers to the number of bytes of data which follow: the eight bytes of type and length are not included. Therefore, a chunk with a length of 6 would actually occupy 14 bytes in the disk file.

This chunk architecture is similar to that used by Electronic Arts' IFF format, and the chunks described herein could easily be placed in an IFF file. The MIDI File itself is not an IFF file: it contains no nested chunks, and chunks are not constrained to be an even number of bytes long. Converting it to an IFF file is as easy as padding odd length chunks, and sticking the whole thing inside a FORM chunk.

1.4 - Chunk Types

MIDI Files contain two types of chunks: header chunks and track chunks. A -header- chunk provides a minimal amount of information pertaining to the entire MIDI file. A -track- chunk contains a sequential stream of MIDI data which may contain information for up to 16 MIDI channels. The concepts of multiple tracks, multiple MIDI outputs, patterns, sequences, and songs may all be implemented using several track chunks.

A MIDI File always starts with a header chunk, and is followed by one or more track chunks.

MThd <length of header data>
<header data>
MTrk <length of track data>
<track data>
MTrk <length of track data>
<track data>
. . .

2 - Chunk Descriptions

2.1 - Header Chunks

The header chunk at the beginning of the file specifies some basic information about the data in the file. Here's the syntax of the complete chunk:

<Header Chunk> = <chunk type><length><format><ntrks><division>

As described above, <chunk type> is the four ASCII characters 'MThd'; <length> is a 32-bit representation of the number 6 (high byte first).

The data section contains three 16-bit words, stored most-significant byte first.

The first word, <format>, specifies the overall organisation of the file. Only three values of <format> are specified:

0-the file contains a single multi-channel track
1-the file contains one or more simultaneous tracks (or MIDI outputs) of a sequence
2-the file contains one or more sequentially independent single-track patterns

More information about these formats is provided below.

The next word, <ntrks>, is the number of track chunks in the file. It will always be 1 for a format 0 file.

The third word, <division>, specifies the meaning of the delta-times. It has two formats, one for metrical time, and one for time-code-based time:

bit 15	bits 14 thru 8	bits 7 thru 0
0	ticks per quarter-note	
1	negative SMPTE format	ticks per frame

If bit 15 of <division> is zero, the bits 14 thru 0 represent the number of delta time "ticks" which make up a quarter-note. For instance, if division is 96, then a time interval of an eighth-note between two events in the file would be 48.

If bit 15 of <division> is a one, delta times in a file correspond to subdivisions of a second, in a way consistent with SMPTE and MIDI Time Code. Bits 14 thru 8 contain one of the four values -24, -25, -29, or -30, corresponding to the four standard SMPTE and MIDI Time Code formats (-29 corresponds to 30 drop frame), and represents the number of frames per second. These negative numbers are stored in two's compliment form. The second byte (stored positive) is the resolution within a frame: typical values may be 4 (MIDI Time Code resolution), 8, 10, 80 (bit resolution), or 100. This stream allows exact specifications of time-code-based tracks, but also allows millisecond-based tracks by specifying 25 frames/sec and a resolution of 40 units per frame. If the events in a file are stored with a bit resolution of thirty-frame time code, the division word would be E250 hex.

2.2 - MIDI File Formats 0,1 and 2

A Format 0 file has a header chunk followed by one track chunk. It is the most interchangeable representation of data. It is very useful for a simple single-track player in a program which needs to make synthesisers make sounds, but which is primarily concerned with something else such as mixers or sound effect boxes. It is very desirable to be able to produce such a format, even if your program is track-based, in order to work with these simple programs.

A Format 1 or 2 file has a header chunk followed by one or more track chunks. programs which support several simultaneous tracks should be able to save and read data in format 1, a vertically one dimensional form, that is, as a collection of tracks. Programs which support several independent patterns should be able to save and read data in format 2, a horizontally one dimensional form. Providing these minimum capabilities will ensure maximum interchangeability.

In a MIDI system with a computer and a SMPTE synchroniser which uses Song Pointer and Timing Clock, tempo maps (which describe the tempo throughout the track, and may also include time signature information, so that the bar number may be derived) are generally created on the computer. To use them with the synchroniser, it is necessary to transfer them from the computer. To make it easy for the synchroniser to extract this data from a MIDI File, tempo information should always be stored in the first MTrk chunk. For a format 0 file, the tempo will be scattered through the track and the tempo map reader should ignore the intervening events; for a format 1 file, the tempo map must be stored as the first track. It is polite to a tempo map reader to offer your user the ability to make a format 0 file with just the tempo, unless you can use format 1.

All MIDI Files should specify tempo and time signature. If they don't, the time signature is assumed to be 4/4, and the tempo 120 beats per minute. In format 0, these meta-events should occur at least at the beginning of the single multi-channel track. In format 1, these meta-events should be contained in the first track. In format 2, each of the temporally independent patterns should contain at least initial time signature and tempo information.

Format IDs to support other structures may be defined in the future. A program encountering an unknown format ID may still read other MTrk chunks it finds from the file, as format 1 or 2, if its user can make sense of

them and arrange them into some other structure if appropriate. Also, more parameters may be added to the MThd chunk in the future: it is important to read and honour the length, even if it is longer than 6.

2.3 - Track Chunks

The track chunks (type MTrk) are where actual song data is stored. Each track chunk is simply a stream of MIDI events (and non-MIDI events), preceded by delta-time values. The format for Track Chunks (described below) is exactly the same for all three formats (0, 1, and 2: see "Header Chunk" above) of MIDI Files.

Here is the syntax of an MTrk chunk (the + means "one or more": at least one MTrk event must be present):

<Track Chunk> = <chunk type><length><MTrk event>+

The syntax of an MTrk event is very simple:

<MTrk event> = <delta-time><event>

<delta-time> is stored as a variable-length quantity. It represents the amount of time before the following event. If the first event in a track occurs at the very beginning of a track, or if two events occur simultaneously, a delta-time of zero is used. Delta-times are always present. (Not storing delta-times of 0 requires at least two bytes for any other value, and most delta-times aren't zero.) Delta-time is in some fraction of a beat (or a second, for recording a track with SMPTE times), as specified in the header chunk.

<event> = <MIDI event> | <sysex event> | <meta-event>

<MIDI event> is any MIDI channel message See Appendix 1 - MIDI Messages. Running status is used: status bytes of MIDI channel messages may be omitted if the preceding event is a MIDI channel message with the same status. The first event in each MTrk chunk must specify status. Delta-time is not considered an event itself: it is an integral part of the syntax for an MTrk event. Notice that running status occurs across delta-times.

<sysex event> is used to specify a MIDI system exclusive message, either as one unit or in packets, or as an "escape" to specify any arbitrary bytes to be transmitted. See Appendix 1 - MIDI Messages. A normal complete system exclusive message is stored in a MIDI File in this way:

F0 <length> <bytes to be transmitted after F0>

The length is stored as a variable-length quantity. It specifies the number of bytes which follow it, not including the F0 or the length itself. For instance, the transmitted message F0 43 12 00 07 F7 would be stored in a MIDI File as F0 05 43 12 00 07 F7. It is required to include the F7 at the end so that the reader of the MIDI File knows that it has read the entire message.

Another form of sysex event is provided which does not imply that an F0 should be transmitted. This may be used as an "escape" to provide for the transmission of things which would not otherwise be legal, including system realtime messages, song pointer or select, MIDI Time Code, etc. This uses the F7 code:

F7 <length> <all bytes to be transmitted>

Unfortunately, some synthesiser manufacturers specify that their system exclusive messages are to be transmitted as little packets. Each packet is only part of an entire syntactical system exclusive message, but the times they are transmitted are important. Examples of this are the bytes sent in a CZ patch dump, or the FB-01's "system exclusive mode" in which microtonal data can be transmitted. The F0 and F7 sysex events may be used together to break up syntactically complete system exclusive messages into timed packets.

An F0 sysex event is used for the first packet in a series -- it is a message in which the F0 should be transmitted. An F7 sysex event is used for the remainder of the packets, which do not begin with F0. (Of course, the F7 is not

considered part of the system exclusive message).

A syntactic system exclusive message must always end with an F7, even if the real-life device didn't send one, so that you know when you've reached the end of an entire sysex message without looking ahead to the next event in the MIDI File. If it's stored in one complete F0 sysex event, the last byte must be an F7. There also must not be any transmittable MIDI events in between the packets of a multi-packet system exclusive message. This principle is illustrated in the paragraph below.

Here is a MIDI File of a multi-packet system exclusive message: suppose the bytes F0 43 12 00 were to be sent, followed by a 200-tick delay, followed by the bytes 43 12 00 43 12 00, followed by a 100-tick delay, followed by the bytes 43 12 00 F7, this would be in the MIDI File:

F0 03 43 12 00	
81 48	200-tick delta time
F7 06 43 12 00 43 12 00	
64	100-tick delta time
F7 04 43 12 00 F7	

When reading a MIDI File, and an F7 sysex event is encountered without a preceding F0 sysex event to start a multi-packet system exclusive message sequence, it should be presumed that the F7 event is being used as an "escape". In this case, it is not necessary that it end with an F7, unless it is desired that the F7 be transmitted.

<meta-event> specifies non-MIDI information useful to this format or to sequencers, with this syntax:

FF <type> <length> <bytes>

All meta-events begin with FF, then have an event type byte (which is always less than 128), and then have the length of the data stored as a variable-length quantity, and then the data itself. If there is no data, the length is 0. As with chunks, future meta-events may be designed which may not be known to existing programs, so programs must properly ignore meta-events which they do not recognise, and indeed should expect to see them. Programs must never ignore the length of a meta-event which they do not recognise, and they shouldn't be surprised if it's bigger than expected. If so, they must ignore everything past what they know about. However, they must not add anything of their own to the end of the meta- event. Sysex events and meta events cancel any running status which was in effect. Running status does not apply to and may not be used for these messages.

3 - Meta-Events

A few meta-events are defined herein. It is not required for every program to support every meta-event.

In the syntax descriptions for each of the meta-events a set of conventions is used to describe parameters of the events. The FF which begins each event, the type of each event, and the lengths of events which do not have a variable amount of data are given directly in hexadecimal. A notation such as dd or se, which consists of two lower-case letters, mnemonically represents an 8-bit value. Four identical lower-case letters such as wwww mnemonically refer to a 16-bit value, stored most-significant-byte first. Six identical lower-case letters such as tttttt refer to a 24-bit value, stored most-significant-byte first. The notation len refers to the length portion of the meta-event syntax, that is, a number, stored as a variable- length quantity, which specifies how many bytes (possibly text) data were just specified by the length.

In general, meta-events in a track which occur at the same time may occur in any order. If a copyright event is used, it should be placed as early as possible in the file, so it will be noticed easily. Sequence Number and Sequence/Track Name events, if present, must appear at time 0. An end-of- track event must occur as the last event in the track.

3.1 - Meta-Event Definitions

FF 00 02 Sequence Number
This optional event, which must occur at the beginning of a track, before any nonzero delta-times, and before any transmittable MIDI events, specifies the number of a sequence. In a format 2 MIDI File, it is used to identify each "pattern" so that a "song" sequence using the Cue message can refer to the patterns. If the ID numbers are omitted, the sequences' locations in order in the file are used as defaults. In a format 0 or 1 MIDI File, which only contain one sequence, this number should be contained in the first (or only) track. If transfer of several multitrack sequences is required, this must be done as a group of format 1 files, each with a different sequence number.

FF 01 len text Text Event
Any amount of text describing anything. It is a good idea to put a text event right at the beginning of a track, with the name of the track, a description of its intended orchestration, and any other information which the user wants to put there. Text events may also occur at other times in a track, to be used as lyrics, or descriptions of cue points. The text in this event should be printable ASCII characters for maximum interchange. However, other character codes using the high-order bit may be used for interchange of files between different programs on the same computer which supports an extended character set. Programs on a computer which does not support non-ASCII characters should ignore those characters.

Meta-event types 01 through 0F are reserved for various types of text events, each of which meets the specification of text events (above) but is used for a different purpose:

FF 02 len text Copyright Notice
Contains a copyright notice as printable ASCII text. The notice should contain the characters (C), the year of the copyright, and the owner of the copyright. If several pieces of music are in the same MIDI File, all of the copyright notices should be placed together in this event so that it will be at the beginning of the file. This event should be the first event in the track chunk, at time 0.

FF 03 len text Sequence/Track Name
If in a format 0 track, or the first track in a format 1 file, the name of the sequence. Otherwise, the name of the track.

FF 04 len text Instrument Name
A description of the type of instrumentation to be used in that track. May be used with the MIDI Prefix meta-event to specify which MIDI channel the description applies to, or the channel may be specified as text in the event itself.

FF 05 len text Lyric
A lyric to be sung. Generally, each syllable will be a separate lyric event which begins at the event's time.

FF 06 len text Marker
Normally in a format 0 track, or the first track in a format 1 file. The name of that point in the sequence, such as a rehearsal letter or section name ("First Verse", etc.)

FF 07 len text Cue Point
A description of something happening on a film or video screen or stage at that point in the musical score ("Car crashes into house", "curtain opens", "she slaps his face", etc.)

FF 20 01 cc MIDI Channel Prefix
The MIDI channel (0-15) contained in this event may be used to associate a MIDI channel with all events which follow, including System exclusive and meta-events. This channel is "effective" until the next normal MIDI event (which contains a channel) or the next MIDI Channel Prefix meta-event. If MIDI channels refer to "tracks", this message may be put into a format 0 file, keeping their non-MIDI data associated with a track. This capability is also present in Yamaha's ESEQ file format.

FF 2F 00 End of Track
This event is not optional. It is included so that an exact ending point may be specified for the track, so that an exact length is defined, which is necessary for tracks which are looped or concatenated.

FF 51 03 tttttt Set Tempo (in microseconds per MIDI quarter-note)
This event indicates a tempo change. Another way of putting "microseconds per quarter-note" is "24ths of a microsecond per MIDI clock". Representing tempos as time per beat instead of beat per time allows absolutely exact long-term synchronisation with a time-based sync protocol such as SMPTE time code or MIDI time code. The amount of accuracy provided by this tempo resolution allows a four-minute piece at 120 beats per minute to be accurate within 500 usec at the end of the piece. Ideally, these events should only occur where MIDI clocks would be located -- this convention is intended to guarantee, or at least increase the likelihood, of compatibility with other synchronisation devices so that a time signature/tempo map stored in this format may easily be transferred to another device.

FF 54 05 hr mn se fr ff SMPTE Offset
This event, if present, designates the SMPTE time at which the track chunk is supposed to start. It should be present at the beginning of the track, that is, before any nonzero delta-times, and before any transmittable MIDI events. the hour must be encoded with the SMPTE format, just as it is in MIDI Time Code. In a format 1 file, the SMPTE Offset must be stored with the tempo map, and has no meaning in any of the other tracks. The ff field contains fractional frames, in 100ths of a frame, even in SMPTE-based tracks which specify a different frame subdivision for delta-times.

FF 58 04 nn dd cc bb Time Signature
The time signature is expressed as four numbers. nn and dd represent the numerator and denominator of the time signature as it would be notated. The denominator is a negative power of two: 2 represents a quarter-note, 3 represents an eighth-note, etc. The cc parameter expresses the number of MIDI clocks in a metronome click. The bb parameter expresses the number of notated 32nd-notes in a MIDI quarter-note (24 MIDI clocks). This was added because there are already multiple programs which allow a user to specify that what MIDI thinks of as a quarter-note (24 clocks) is to be notated as, or related to in terms of, something else.

Therefore, the complete event for 6/8 time, where the metronome clicks every three eighth-notes, but there are 24 clocks per quarter-note, 72 to the bar, would be (in hex):

FF 58 04 06 03 24 08

That is, 6/8 time (8 is 2 to the 3rd power, so this is 06 03), 36 MIDI clocks per dotted-quarter (24 hex!), and eight notated 32nd-notes per quarter-note.

FF 59 02 sf mi Key Signature
sf = -7: 7 flats
sf = -1: 1 flat
sf = 0: key of C
sf = 1: 1 sharp
sf = 7: 7 sharps

mi = 0: major key
mi = 1: minor key

FF 7F len data Sequencer Specific Meta-Event
Special requirements for particular sequencers may use this event type: the first byte or bytes of data is a manufacturer ID (these are one byte, or if the first byte is 00, three bytes). As with MIDI System Exclusive, manufacturers who define something using this meta-event should publish it so that others may be used by a sequencer which elects to use this as its only file format; sequencers with their established feature-specific formats should probably stick to the standard features when using this format.

See Appendix 2 - Program Fragments and Example MIDI Files for an example midi file.

Appendix 1 - MIDI Messages

A MIDI message is made up of an eight-bit status byte which is generally followed by one or two data bytes. There are a number of different types of MIDI messages. At the highest level, MIDI messages are classified as being either Channel Messages or System Messages. Channel messages are those which apply to a specific Channel, and the Channel number is included in the status byte for these messages. System messages are not Channel specific, and no Channel number is indicated in their status bytes.

Channel Messages may be further classified as being either Channel Voice Messages, or Mode Messages. Channel Voice Messages carry musical performance data, and these messages comprise most of the traffic in a typical MIDI data stream. Channel Mode messages affect the way a receiving instrument will respond to the Channel Voice messages.

MIDI System Messages are classified as being System Common Messages, System Real Time Messages, or System Exclusive Messages. System Common messages are intended for all receivers in the system. System Real Time messages are used for synchronisation between clock-based MIDI components. System Exclusive messages include a Manufacturer's Identification (ID) code, and are used to transfer any number of data bytes in a format specified by the referenced manufacturer.

Appendix 1.1 - Table of Major MIDI Messages

Channel Voice Messages		
Status D7----D0 nnnn is the MIDI channel no.	**Data Byte(s) D7----D0**	**Description**
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off event. This message is sent when a note is released (ended). (kkkkkkk) is the key (note) number. (vvvvvvv) is the velocity.
1001nnnn	0kkkkkkk 0vvvvvvv	Note On event. This message is sent when a note is depressed (start). (kkkkkkk) is the key (note) number. (vvvvvvv) is the velocity.
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (Aftertouch). This message is most often sent by pressing down on the key after it "bottoms out". (kkkkkkk) is the key (note) number. (vvvvvvv) is the pressure value.
1011nnnn	0ccccccc 0vvvvvvv	Control Change. This message is sent when a controller value changes. Controllers include devices such as pedals and levers. Certain controller numbers are reserved for specific purposes. See Channel Mode Messages. (ccccccc) is the controller number. (vvvvvvv) is the new value.
1100nnnn	0ppppppp	Program Change.

		This message sent when the patch number changes. (ppppppp) is the new program number.
1101nnnn	0vvvvvvv	Channel Pressure (After-touch). This message is most often sent by pressing down on the key after it "bottoms out". This message is different from polyphonic after-touch. Use this message to send the single greatest pressure value (of all the current depressed keys). (vvvvvvv) is the pressure value.
1110nnnn	0lllllll 0mmmmmmm	Pitch Wheel Change. This message is sent to indicate a change in the pitch wheel. The pitch wheel is measured by a fourteen bit value. Centre (no pitch change) is 2000H. Sensitivity is a function of the transmitter. (lllllll) are the least significant 7 bits. (mmmmmmm) are the most significant 7 bits.

Channel Mode Messages (See also Control Change, above)		
1011nnnn	0ccccccc 0vvvvvvv	Channel Mode Messages. This the same code as the Control Change (above), but implements Mode control by using reserved controller numbers. The numbers are: Local Control. When Local Control is Off, all devices on a given channel will respond only to data received over MIDI. Played data, etc. will be ignored. Local Control On restores the functions of the normal controllers. c = 122, v = 0: Local Control Off c = 122, v = 127: Local Control On All Notes Off. When an All Notes Off is received all oscillators will turn off. c = 123, v = 0: All Notes Off c = 124, v = 0: Omni Mode Off c = 125, v = 0: Omni Mode On c = 126, v = M: Mono Mode On (Poly Off) where M is the number of channels (Omni Off) or 0 (Omni On) c = 127, v = 0: Poly Mode On (Mono Off) (Note: These four messages also cause All Notes Off)

System Common Messages		
11110000	0iiiiiii 0ddddddd 0ddddddd 11110111	System Exclusive. This message makes up for all that MIDI doesn't support. (iiiiiii) is usually a seven-bit Manufacturer's I.D. code. If the synthesiser recognises the I.D. code as its own, it will listen to the rest of the message (ddddddd). Otherwise, the message will be ignored. System Exclusive is used to send bulk dumps such as patch parameters and other non-spec data. (Note: Real-Time messages ONLY may be interleaved with a System Exclusive.) This message also is used for extensions called Universal Exclusive Messages.
11110001		Undefined.
11110010	0lllllll 0mmmmmmm	Song Position Pointer. This is an internal 14 bit register that holds the number of MIDI beats (1 beat= six MIDI clocks) since the start of the song. l is the LSB, m the MSB.
11110011	0sssssss	Song Select. The Song Select specifies which sequence or song is to be played.
11110100		Undefined.
11110101		Undefined.

11110110		Tune Request. Upon receiving a Tune Request, all analog synthesisers should tune their oscillators.
11110111		End of Exclusive. Used to terminate a System Exclusive dump (see above).
System Real-Time Messages		
11111000		Timing Clock. Sent 24 times per quarter note when synchronisation is required.
11111001		Undefined.
11111010		Start. Start the current sequence playing. (This message will be followed with Timing Clocks).
11111011		Continue. Continue at the point the sequence was Stopped.
11111100		Stop. Stop the current sequence.
11111101		Undefined.
11111110		Active Sensing. Use of this message is optional. When initially sent, the receiver will expect to receive another Active Sensing message each 300ms (max), or it will be assume that the connection has been terminated. At termination, the receiver will turn off all voices and return to normal (non-active sensing) operation.
11111111		Reset. Reset all receivers in the system to power-up status. This should be used sparingly, preferably under manual control. In particular, it should not be sent on power-up. In a MIDI file this is used as an escape to introduce <meta events>.

Appendix 1.2 - Table of MIDI Controller Messages (Data Bytes)

The following table lists the MIDI Controller messages in numerical (binary) order.

2nd Byte Value			Function	3rd Byte	
Binary	**Hex**	**Dec**		**Value**	**Use**
00000000	00	0	Bank Select	0-127	MSB
00000001	01	1	* Modulation wheel	0-127	MSB
00000010	02	2	Breath control	0-127	MSB
00000011	03	3	Undefined	0-127	MSB
00000100	04	4	Foot controller	0-127	MSB
00000101	05	5	Portamento time	0-127	MSB
00000110	06	6	Data Entry	0-127	MSB
00000111	07	7	* Channel Volume (formerly Main Volume)	0-127	MSB
00001000	08	8	Balance	0-127	MSB
00001001	09	9	Undefined	0-127	MSB
00001010	0A	10	* Pan	0-127	MSB
00001011	0B	11	* Expression Controller	0-127	MSB

00001100	0C	12	Effect control 1	0-127	MSB
00001101	0D	13	Effect control 2	0-127	MSB
00001110	0E	14	Undefined	0-127	MSB
00001111	0F	15	Undefined	0-127	MSB
00010000	10	16	General Purpose Controller #1	0-127	MSB
00010001	11	17	General Purpose Controller #2	0-127	MSB
00010010	12	18	General Purpose Controller #3	0-127	MSB
00010011	13	19	General Purpose Controller #4	0-127	MSB
00010100	14	20	Undefined	0-127	MSB
00010101	15	21	Undefined	0-127	MSB
00010110	16	22	Undefined	0-127	MSB
00010111	17	23	Undefined	0-127	MSB
00011000	18	24	Undefined	0-127	MSB
00011001	19	25	Undefined	0-127	MSB
00011010	1A	26	Undefined	0-127	MSB
00011011	1B	27	Undefined	0-127	MSB
00011100	1C	28	Undefined	0-127	MSB
00011101	1D	29	Undefined	0-127	MSB
00011110	1E	30	Undefined	0-127	MSB
00011111	1F	31	Undefined	0-127	MSB
00100000	20	32	Bank Select	0-127	LSB
00100001	21	33	Modulation wheel	0-127	LSB
00100010	22	34	Breath control	0-127	LSB
00100011	23	35	Undefined	0-127	LSB
00100100	24	36	Foot controller	0-127	LSB
00100101	25	37	Portamento time	0-127	LSB
00100110	26	38	Data entry	0-127	LSB
00100111	27	39	Channel Volume (formerly Main Volume)	0-127	LSB
00101000	28	40	Balance	0-127	LSB
00101001	29	41	Undefined	0-127	LSB
00101010	2A	42	Pan	0-127	LSB
00101011	2B	43	Expression Controller	0-127	LSB
00101100	2C	44	Effect control 1	0-127	LSB
00101101	2D	45	Effect control 2	0-127	LSB
00101110	2E	46	Undefined	0-127	LSB
00101111	2F	47	Undefined	0-127	LSB
00110000	30	48	General Purpose Controller #1	0-127	LSB
00110001	31	49	General Purpose Controller #2	0-127	LSB
00110010	32	50	General Purpose Controller #3	0-127	LSB
00110011	33	51	General Purpose Controller #4	0-127	LSB
00110100	34	52	Undefined	0-127	LSB

00110101	35	53	Undefined	0-127	LSB
00110110	36	54	Undefined	0-127	LSB
00110111	37	55	Undefined	0-127	LSB
00111000	38	56	Undefined	0-127	LSB
00111001	39	57	Undefined	0-127	LSB
00111010	3A	58	Undefined	0-127	LSB
00111011	3B	59	Undefined	0-127	LSB
00111100	3C	60	Undefined	0-127	LSB
00111101	3D	61	Undefined	0-127	LSB
00111110	3E	62	Undefined	0-127	LSB
00111111	3F	63	Undefined	0-127	LSB
01000000	40	64	* Damper pedal on/off (Sustain)	<63=off	>64=on
01000001	41	65	Portamento on/off	<63=off	>64=on
01000010	42	66	Sustenuto on/off	<63=off	>64=on
01000011	43	67	Soft pedal on/off	<63=off	>64=on
01000100	44	68	Legato Footswitch	<63=off	>64=on
01000101	45	69	Hold 2	<63=off	>64=on
01000110	46	70	Sound Controller 1 (Sound Variation)	0-127	LSB
01000111	47	71	Sound Controller 2 (Timbre)	0-127	LSB
01001000	48	72	Sound Controller 3 (Release Time)	0-127	LSB
01001001	49	73	Sound Controller 4 (Attack Time)	0-127	LSB
01001010	4A	74	Sound Controller 5 (Brightness)	0-127	LSB
01001011	4B	75	Sound Controller 6	0-127	LSB
01001100	4C	76	Sound Controller 7	0-127	LSB
01001101	4D	77	Sound Controller 8	0-127	LSB
01001110	4E	78	Sound Controller 9	0-127	LSB
01001111	4F	79	Sound Controller 10	0-127	LSB
01010000	50	80	General Purpose Controller #5	0-127	LSB
01010001	51	81	General Purpose Controller #6	0-127	LSB
01010010	52	82	General Purpose Controller #7	0-127	LSB
01010011	53	83	General Purpose Controller #8	0-127	LSB
01010100	54	84	Portamento Control	0-127	Source Note
01010101	55	85	Undefined	0-127	LSB
01010110	56	86	Undefined	0-127	LSB
01010111	57	87	Undefined	0-127	LSB
01011000	58	88	Undefined	0-127	LSB
01011001	59	89	Undefined	0-127	LSB
01011010	5A	90	Undefined	0-127	LSB
01011011	5B	91	Effects 1 Depth	0-127	LSB
01011100	5C	92	Effects 2 Depth	0-127	LSB

01011101	5D	93	Effects 3 Depth	0-127	LSB
01011110	5E	94	Effects 4 Depth	0-127	LSB
01011111	5F	95	Effects 5 Depth	0-127	LSB
01100000	60	96	Data entry +1	N/A	
01100001	61	97	Data entry -1	N/A	
01100010	62	98	Non-Registered Parameter Number LSB	0-127	LSB
01100011	63	99	Non-Registered Parameter Number MSB	0-127	MSB
01100100	64	100	* Registered Parameter Number LSB	0-127	LSB
01100101	65	101	* Registered Parameter Number MSB	0-127	MSB
01100110	66	102	Undefined	?	
01100111	67	103	Undefined	?	
01101000	68	104	Undefined	?	
01101001	69	105	Undefined	?	
01101010	6A	106	Undefined	?	
01101011	6B	107	Undefined	?	
01101100	6C	108	Undefined	?	
01101101	6D	109	Undefined	?	
01101110	6E	110	Undefined	?	
01101111	6F	111	Undefined	?	
01110000	70	112	Undefined	?	
01110001	71	113	Undefined	?	
01110010	72	114	Undefined	?	
01110011	73	115	Undefined	?	
01110100	74	116	Undefined	?	
01110101	75	117	Undefined	?	
01110110	76	118	Undefined	?	
01110111	77	119	Undefined	?	
01111000	78	120	All Sound Off	0	
01111001	79	121	* Reset All Controllers	0	
01111010	7A	122	Local control on/off	0=off	127=on
01111011	7B	123	* All notes off	0	
01111100	7C	124	Omni mode off (+ all notes off)	0	
01111101	7D	125	Omni mode on (+ all notes off)	0	
01111110	7E	126	Poly mode on/off (+ all notes off)	**	
01111111	7F	127	Poly mode on (incl mono=off +all notes off)	0	

* Equipment must respond in order to comply with General MIDI level 1.
** This equals the number of channels, or zero if the number of channels equals the number of voices in the receiver.

Appendix 1.3 - Table of MIDI Note Numbers

This table lists all MIDI Note Numbers by octave.

The absolute octave number designations are based on Middle C = C4, which is an arbitrary but widely used assignment.

Octave #	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

Appendix 1.4 - General MIDI Instrument Patch Map

- The names of the instruments indicate what sort of sound will be heard when that instrument number (MIDI Program Change or "PC#") is selected on the GM synthesizer.
- These sounds are the same for all MIDI Channels except Channel 10, which has only percussion sounds and some sound "effects". (See Appendix 1.5 - General MIDI Percussion Key Map)

GM Instrument Families

The General MIDI instrument sounds are grouped by families. In each family are 8 specific instruments.

PC#	Family	PC#	Family
1-8	Piano	65-72	Reed
9-16	Chromatic Percussion	73-80	Pipe
17-24	Organ	81-88	Synth Lead
25-32	Guitar	89-96	Synth Pad
33-40	Bass	97-104	Synth Effects
41-48	Strings	105-112	Ethnic
49-56	Ensemble	113-120	Percussive
57-64	Brass	121-128	Sound Effects

GM Instrument Patch Map

Note: While GM does not define the actual characteristics of any sounds, the names in parentheses after each of the synth leads, pads, and sound effects are, in particular, intended only as guides.

PC#	Instrument	PC#	Instrument
1.	Acoustic Grand Piano	65.	Soprano Sax
2.	Bright Acoustic Piano	66.	Alto Sax
3.	Electric Grand Piano	67.	Tenor Sax
4.	Honky-tonk Piano	68.	Baritone Sax
5.	Electric Piano 1 (Rhodes Piano)	69.	Oboe
6.	Electric Piano 2 (Chorused Piano)	70.	English Horn
7.	Harpsichord	71.	Bassoon
8.	Clavinet	72.	Clarinet
9.	Celesta	73.	Piccolo
10.	Glockenspiel	74.	Flute
11.	Music Box	75.	Recorder
12.	Vibraphone	76.	Pan Flute
13.	Marimba	77.	Blown Bottle
14.	Xylophone	78.	Shakuhachi
15.	Tubular Bells	79.	Whistle
16.	Dulcimer (Santur)	80.	Ocarina
17.	Drawbar Organ (Hammond)	81.	Lead 1 (square wave)
18.	Percussive Organ	82.	Lead 2 (sawtooth wave)
19.	Rock Organ	83.	Lead 3 (calliope)
20.	Church Organ	84.	Lead 4 (chiffer)
21.	Reed Organ	85.	Lead 5 (charang)
22.	Accordion (French)	86.	Lead 6 (voice solo)
23.	Harmonica	87.	Lead 7 (fifths)
24.	Tango Accordion (Band neon)	88.	Lead 8 (bass + lead)
25.	Acoustic Guitar (nylon)	89.	Pad 1 (new age Fantasia)
26.	Acoustic Guitar (steel)	90.	Pad 2 (warm)
27.	Electric Guitar (jazz)	91.	Pad 3 (polysynth)
28.	Electric Guitar (clean)	92.	Pad 4 (choir space voice)
29.	Electric Guitar (muted)	93.	Pad 5 (bowed glass)
30.	Overdriven Guitar	94.	Pad 6 (metallic pro)
31.	Distortion Guitar	95.	Pad 7 (halo)
32.	Guitar harmonics	96.	Pad 8 (sweep)
33.	Acoustic Bass	97.	FX 1 (rain)
34.	Electric Bass (fingered)	98.	FX 2 (soundtrack)
35.	Electric Bass (picked)	99.	FX 3 (crystal)
36.	Fretless Bass	100.	FX 4 (atmosphere)
37.	Slap Bass 1	101.	FX 5 (brightness)

38.	Slap Bass 2	102.	FX 6 (goblins)
39.	Synth Bass 1	103.	FX 7 (echoes, drops)
40.	Synth Bass 2	104.	FX 8 (sci-fi, star theme)
41.	Violin	105.	Sitar
42.	Viola	106.	Banjo
43.	Cello	107.	Shamisen
44.	Contrabass	108.	Koto
45.	Tremolo Strings	109.	Kalimba
46.	Pizzicato Strings	110.	Bag pipe
47.	Orchestral Harp	111.	Fiddle
48.	Timpani	112.	Shanai
49.	String Ensemble 1 (strings)	113.	Tinkle Bell
50.	String Ensemble 2 (slow strings)	114.	Agogo
51.	SynthStrings 1	115.	Steel Drums
52.	SynthStrings 2	116.	Woodblock
53.	Choir Aahs	117.	Taiko Drum
54.	Voice Oohs	118.	Melodic Tom
55.	Synth Voice	119.	Synth Drum
56.	Orchestra Hit	120.	Reverse Cymbal
57.	Trumpet	121.	Guitar Fret Noise
58.	Trombone	122.	Breath Noise
59.	Tuba	123.	Seashore
60.	Muted Trumpet	124.	Bird Tweet
61.	French Horn	125.	Telephone Ring
62.	Brass Section	126.	Helicopter
63.	SynthBrass 1	127.	Applause
64.	SynthBrass 2	128.	Gunshot

Appendix 1.5 - General MIDI Percussion Key Map

On MIDI Channel 10, each MIDI Note number ("Key#") corresponds to a different drum sound, as shown below. GM-compatible instruments must have the sounds on the keys shown here. While many current instruments also have additional sounds above or below the range show here, and may even have additional "kits" with variations of these sounds, only these sounds are supported by General MIDI.

Key#	Note	Drum Sound	Key#	Note	Drum Sound
35	B1	Acoustic Bass Drum	59	B3	Ride Cymbal 2
36	C2	Bass Drum 1	60	C4	Hi Bongo
37	C#2	Side Stick	61	C#4	Low Bongo
38	D2	Acoustic Snare	62	D4	Mute Hi Conga
39	D#2	Hand Clap	63	D#4	Open Hi Conga

40	E2	Electric Snare	64	E4	Low Conga
41	F2	Low Floor Tom	65	F4	High Timbale
42	F#2	Closed Hi Hat	66	F#4	Low Timbale
43	G2	High Floor Tom	67	G4	High Agogo
44	G#2	Pedal Hi-Hat	68	G#4	Low Agogo
45	A2	Low Tom	69	A4	Cabasa
46	A#2	Open Hi-Hat	70	A#4	Maracas
47	B2	Low-Mid Tom	71	B4	Short Whistle
48	C3	Hi Mid Tom	72	C5	Long Whistle
49	C#3	Crash Cymbal 1	73	C#5	Short Guiro
50	D3	High Tom	74	D5	Long Guiro
51	D#3	Ride Cymbal 1	75	D#5	Claves
52	E3	Chinese Cymbal	76	E5	Hi Wood Block
53	F3	Ride Bell	77	F5	Low Wood Block
54	F#3	Tambourine	78	F#5	Mute Cuica
55	G3	Splash Cymbal	79	G5	Open Cuica
56	G#3	Cowbell	80	G#5	Mute Triangle
57	A3	Crash Cymbal 2	81	A5	Open Triangle
58	A#3	Vibraslap			

Appendix 2 - Program Fragments and Example MIDI Files

Here are some of the routines to read and write variable-length numbers in MIDI Files. These routines are in C, and use getc and putc, which read and write single 8-bit characters from/to the files infile and outfile.

```c
WriteVarLen(value)
register long value;
{
register long buffer;

buffer = value & 0x7f;
while((value >>= 7) > 0)
 {
 buffer <<= 8;
 buffer |= 0x80;
 buffer += (value &0x7f);
 }

while (TRUE)
 {
 putc(buffer,outfile);
 if(buffer & 0x80) buffer >>= 8;
 else
 break;
 }
}
```

```
doubleword ReadVarLen()
{
register doubleword value;
register byte c;

if((value = getc(infile)) & 0x80)
 {
 value &= 0x7f;
 do
  {
  value = (value << 7) + ((c = getc(infile))) & 0x7f;
  } while (c & 0x80);
 }
return(value);
}
```

As an example, MIDI Files for the following excerpt are shown below. First, a format 0 file is shown, with all information intermingled; then, a format 1 file is shown with all data separated into four tracks: one for tempo and time signature, and three for the notes. A resolution of 96 "ticks" per quarter note is used. A time signature of 4/4 and a tempo of 120, though implied, are explicitly stated.
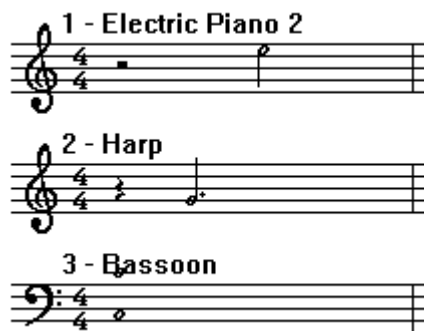
The contents of the MIDI stream represented by this example are broken down here:

Delta-Time (decimal)	Event-Code (hex)	Other Bytes (decimal)	Comment
0	FF 58	04 04 02 24 08	4 bytes; 4/4 time; 24 MIDI clocks/click, 8 32nd notes/ 24 MIDI clocks (24 MIDI clocks = 1 crotchet = 1 beat)
0	FF 51	03 500000	3 bytes: 500,000 usec/ quarter note = 120 beats/minute
0	C0	5	Ch.1 Program Change 5 = GM Patch 6 = Electric Piano 2
0	C1	46	Ch.2 Program Change 46 = GM Patch 47 = Harp
0	C2	70	Ch.3 Program Change 70 = GM Patch 71 = Bassoon
0	92	48 96	Ch.3 Note On C3, forte
0	92	60 96	Ch.3 Note On C4, forte
96	91	67 64	Ch.2 Note On G4, mezzo-forte
96	90	76 32	Ch.1 Note On E5, piano
192	82	48 64	Ch.3 Note Off C3, standard
0	82	60 64	Ch.3 Note Off C4, standard
0	81	67 64	Ch.2 Note Off G4, standard
0	80	76 64	Ch.1 Note Off E5, standard

0	FF 2F	00	Track End

The entire format 0 MIDI file contents in hex follow. First, the header chunk:

4D 54 68 64	MThd
00 00 00 06	chunk length
00 00	format 0
00 01	one track
00 60	96 per quarter-note

Then the track chunk. Its header followed by the events (notice the running status is used in places):

4D 54 72 6B	MTrk
00 00 00 3B	chunk length (59)

Delta-Time	Event	Comments
00	FF 58 04 04 02 18 08	time signature
00	FF 51 03 07 A1 20	tempo
00	C0 05	
00	C1 2E	
00	C2 46	
00	92 30 60	
00	3C 60	running status
60	91 43 40	
60	90 4C 20	
81 40	82 30 40	two-byte delta-time
00	3C 40	running status
00	81 43 40	
00	80 4C 40	
00	FF 2F 00	end of track

A format 1 representation of the file is slightly different. Its header chunk:

4D 54 68 64	MThd
00 00 00 06	chunk length
00 01	format 1
00 04	four tracks
00 60	96 per quarter note

First, the track chunk for the time signature/tempo track. Its header, followed by the events:

4D 54 72 6B	MTrk
00 00 00 14	chunk length (20)

Delta-Time	Event	Comments
00	FF 58 04 04 02 18 08	time signature
00	FF 51 03 07 A1 20	tempo
83 00	FF 2F 00	end of track

Then, the track chunk for the first music track. The MIDI convention for note on/off running status is used in this example:

4D 54 72 6B	MTrk

00 00 00 10	chunk length (16)

Delta-Time	Event	Comments
00	C0 05	
81 40	90 4C 20	
81 40	4C 00	Running status: note on, vel=0
00 FF 2F 00		

Then, the track chunk for the second music track:

4D 54 72 6B	MTrk
00 00 00 0F	chunk length (15)

Delta-Time	Event	Comments
00	C1 2E	
60	91 43 40	
82 20	43 00	running status
00	FF 2F 00	end of track

Then, the track chunk for the third music track:

4D 54 72 6B	MTrk
00 00 00 15	chunk length (21)

Delta-Time	Event	Comments
00	C2 46	
00	92 30 60	
00	3C 60	running status
83 00	30 00	two-byte delta-time, running status
00	3C 00	running status
00	FF 2F 00	end of track

Return to CSW2 Home Page

END.