The MIDI File Format

MIDI File Structure

Chunks

MIDI files are structured into *chunks*.

Each chunk consists of:

type	length	data
4 bytes	4 bytes	*length* bytes

- A 4-byte *chunk type* (ASCII)
- A 4-byte *length* (32 bits, msb first)
- *length* bytes of data

There are two types of chunks:

Header Chunks
: which have a chunk type of "MThd"

Track Chunks
: which have a chunk type of "MTrk"

A MIDI file consists of a single header chunk followed by one or more track chunks.

Since the length-field is mandatory in the structure of chunks, it is possible to accomodate chunks other than "MThd" or "MTrk" in a MIDI file, by skipping over their contents. The MIDI specification *requires* that software be able to handle unexpected chunk-types by ignoring the entire chunk.

	<---Chunk--->				
	type	**length**	**Data**		
MIDI File :	`MThd`	6	*<format>*	*<tracks>*	*<division>*
	`MTrk`	*<length>*	*<delta_time> <event> ...*		
	:				
	`MTrk`	*<length>*	*<delta_time> <event> ...*		

Value Representations

Within a MIDI file, there is a variety of information in addition to the basic MIDI data, such as delta-times and meta-events.

Where such information contains numeric values, these are generally represented in one of two formats:

- binary
- variable length quantity

Binary

Binary values are stored:

- 8-bits per byte
- MSB first (Most Significant Byte first)

(unless otherwise specified)

Variable Length Quantities

The *variable-length quantity* provides a convenient means of representing arbitrarily large integers, without creating needlessly large fixed-width integers.

A *variable-length quantity* is a represented as a series of 7-bit values, from most-significant to least-significant. where the last byte of the series bit 7 (the most significant bit) set to 0, and the preceding bytes have bit 7 set to 1.

Examples:

Value			**Variable Length representation**	
Decimal	Hex	Binary	Binary	Hex
-	abcd	aaaabbbbccccdddd	100000aa 1aabbbbc 0ccccddd	-
0 : 127	00 : 7F	0000 0000 : 0111 1111	0000 0000 : 0111 1111	00 : 7F
128 : 16383	80 : 3FFF	00000000 10000000 : 00111111 11111111	10000001 00000000 : 11111111 01111111	81 00 : FF 7F
1000	03E8	11 1110 1000	10000111 01101000	87 68
100000	0F4240	1111 0100 0010 0100 0000	10111101 10000100 01000000	BD 84 40

As you can see from the above examples, small values (0-127) can be represented by a single byte, while larger values are also accomodated.

The largest value allowed within a MIDI file is 0FFFFFFF. This limit is set to allow variable-length quantities to be manipulated as 32-bit integers.

Header Chunks

The data part of a header chunk contains three 16-bit fields. These fields specify the format, number of tracks, and timing for the MIDI file.

The length of the header chunk is 6-bytes. However, software which reads MIDI files is *required* to honor the length field, even if it is greater than expected. Any unexpected data must be ignored.

Header Chunk				
Chunk Type	**length**	**Data**		
4 bytes (ASCII)	4 bytes (32-bit binary)	<-- *length* (= 6 bytes) -->		

		16-bit	16-bit	16-bit
MThd	*<length>*	*<format>*	*<tracks>*	*<division>*

<length>

length in bytes of the chunk data part.
This is a 32-bit binary number, MSB first.
This will be exactly 6 (bytes) for any MIDI file created under the MIDI 1.0 specification. Nevertheless, any MIDI file reader should be able to cope with larger header-chunks, to allow for future expansion.

<format>

The MIDI file format.
This is a 16-bit binary number, MSB first.
The only valid formats are 0, 1 and 2.

<tracks>

The number of track chunks contained in this MIDI file.
This is a 16-bit binary number, MSB first.

<division>

This defines the default unit of delta-time for this MIDI file.
This is a 16-bit binary value, MSB first.

This may be in either of two formats, depending on the value of MS bit:

Bit:	15	14 ... 8	7 ... 0
<division>	0	*ticks per quarter note*	
	1	*-frames/second*	*ticks / frame*

bit 15 = 0:

bits 0-14
number of delta-time units in each a quarter-note.

bit 15 = 1:

bits 0-7
number of delta-time units per SMTPE frame
bits 8-14
form a negative number, representing the number of SMTPE frames per second. Valid values correspond to those in the MTC Quarter Frame message.

```
-24 = 24 frames per second
-25 = 25 frames per second
-29 = 30 frames per second, drop frame
-30 = 30 frames per second, non-drop frame
```

MIDI File Formats

MIDI files come in 3 variations:

- Format 0
  ...which contain a single track

- Format 1
  ... which contain one or more simultaneous tracks
  (ie all tracks are to be played simultaneously).

- Format 2
  ...which contain one or more independent tracks
  (ie each track is to be played independently of the others).

Format 0 MIDI Files

Format 0 MIDI files consist of a header-chunk and a single track-chunk.

The single track chunk will contain all the note and tempo information.

Format 1 MIDI Files

Format 1 MIDI files consist of a header-chunk and one or more track-chunks, with all tracks being played simultaneously.

The first track of a Format 1 file is special, and is also known as the 'Tempo Map'. It should contain all meta-events of the types Time Signature, and Set Tempo. The meta-events Sequence/Track Name, Sequence Number, Marker, and SMTPE Offset. should also be on the first track of a Format 1 file.

Format 2 MIDI Files

Format 2 MIDI files consist of a header-chunk and one or more track-chunks, where each track represents an independant sequence.

Track Chunks

The data part of a track chunk contains one or more <*delta_time*> <*event*> pairs.
The <*delta_time*> is not optional, but zero is a valid delta-time.

Track Chunk		
type	**length**	**data**

4 bytes (ascii)	4 bytes (32-bit binary)	<-- *length* bytes --> (binary data)
MTrk	*<length>*	*<delta_time>* *<event>* ...

<delta_time>
is the number of 'ticks' from the previous event, and is represented as a variable length quantity

<event>
is one of:

- *<midi_event>*
- *<sysex_event>*
- *<meta_event>*

Events

Notice that there are no explicit delimiters between the *<delta_time>* and *<event>* instances.
This is possible because both the delta-times and events have clearly defined lengths:

- The last byte of a delta-time is identified by having MSbit=0;
- MIDI Channel messages have a pre-defined length (even when running status is used);
- and sysex-events and meta-events contain an explicit length field.

MIDI Events

A *<midi_event>* is any MIDI Channel message. This includes:

- Channel Voice messages
- Channel Mode messages

Running status is applicable within MIDI files in the same manner as usual. Running status is cancelled by any *<sysex_event>* or *<meta_event>*

Sysex Events

Where is is desirable to include messages other than MIDI Channel messages in a MIDI file (System Exclusive messages in particular), the *<sysex_event>* can be used.

Sysex events come in two flavors:

Sysex Events

F0 *<length>* *<sysex_data>*	**F0 Sysex Event**	
	This results in a F0 being sent (Start-Of-Exclusive), followed by *<sysex_data>*	
	<length>	is a variable length quantity, specifying

		the length of <*sysex_data*>
	<*sysex_data*>	`F0` <*sysex_data*> is sent as a MIDI message.
`F7` <*length*> <*any_data*>	**F7 Sysex Event** (or 'escape')	
	This results in exactly <*any_data*> being sent, without anything else being added.	
	<*length*>	is a variable length quantity, specifying the length of <*any_data*>
	<*any_data*>	<*any_data*> is sent as a MIDI message.

In both cases, the End-Of-Exclusive message, `F7`, is not sent automatically. It must be specified explicitly within <*sysex_data*> or <*any_data*>.

Most System Exclusive messages are quite simple, and are sent as a single packet of bytes, starting with `F0` and ending with `F7`. These are easily accomodated using the `F0` form of the <*sysex_event*>

However, some System Exclusive messages are used to control device parameters in real-time, and what is syntactically a single System Exclusive message may consist of a series of small parts which must be sent with appropriate delays. Hence it is necessary to be able to break up a single message into a number of events, with appropriate delta-times. This can be accomodated by using the `F7` form of the <*sysex_event*>.

In fact, the F7 Sysex Event can be used to included any data into the MIDI stream, such as MIDI System Real-Time messages (though this is generally not desirable).

Example 1:

To generate the message:

```
F0 7E 09 03 01 01 F7
```

(Sample Dump Request - Device 9, sample number 257)
the <*sysex_event*> would be:

```
F0 06 7E 09 03 01 01 F7
```

or alternately:

```
F7 07 F0 7E 09 03 01 01 F7
```

Example 2:

The MIDI System Real-Time message 'stop' can be inserted using the <*event*> `F7 01 FC`, while 'continue'. would be `F7 01 FB`.

Suppose you wanted to pause an external drum machine by sending a 'stop', followed 48 delta-time units later by 'continue'. The complete delta-time/event sequence would look something like this:

```
00 F7 01 FC 30 F7 01 FB
```

Meta Events

Meta Events are used for things like track-names, lyrics and cue-points, which don't result in MIDI messages being sent, but are still useful components of a MIDI file.

Meta Events have the general form:

`FF` <*type*> <*length*> <*data*>

where:

<type>
is a single byte, specifying the type of meta-event.
The possible range is `00-7F`. Not all values in this range are defined, but programs must be able to cope with (ie ignore) unexpected values by examining the length and skipping over the data portion.

<length>
is the number of bytes of *<data>* following.
This is a variable length quantity
0 is a valid *<length>*

<data>
zero or more bytes of data

Meta Events

`FF 00 02` *ss ss*	**Sequence Number**	
	This is an optional event, which must occur only at the start of a track, before any non-zero delta-time. For Format 2 MIDI files, this is used to identify each track. If omitted, the sequences are numbered sequentially in the order the tracks appear. For Format 1 files, this event should occur on the first track only.	
	ss ss	Sequence Number, 16 bit binary
`FF 01` *<len> <text>*	**Text Event**	
	This event is used for annotating the track with arbitrary text. Arbitrary 8-bit data (other than ascii text) is also permitted.	
	<len>	length of *<text>* (variable length quantity)
	<text>	*<len>* bytes of ascii text, or 8-bit binary data
`FF 02` *<len> <text>*	**Copyright Notice**	
	This event is for a Copyright notice in ascii text. This should be of the form "(C) 1850 J.Strauss" This event should be the first event on the first track.	
`FF 03` *<len> <text>*	**Sequence/Track Name**	
	Name of the sequence or track	
`FF 04` *<len> <text>*	**Instrument Name**	
	A description of the instrument(s) used on this track. This can also be used to describe instruments on a particular MIDI Channel within a track, by preceding this event with the meta-event MIDI Channel Prefix. (or specifying the channel(s) within the text).	
`FF 05` *<len> <text>*	**Lyric**	
	Lyrics for the song. Normally, each syllable will have it's own lyric-event, which occurs at the time the lyric is to be sung.	

FF 06 *<len> <text>*	**Marker**	
	Normally on the first track of a format 1 or format 0 file. Marks a significant point in the sequence (eg "Verse 1")	
FF 07 *<len> <text>*	**Cue Point**	
	Used to include cues for events happening on-stage, such as "curtain rises", "exit, stage left", etc.	
FF 20 01 *cc*	**MIDI Channel Prefix**	
	Associate all following meta-events and sysex-events with the specified MIDI channel, until the next *<midi_event>* (which must contain MIDI channel information).	
	cc	MIDI channel 1-16 Range: 00-0F
FF 2F 00	**End of Track**	
	This event is *not* optional. It is used to give the track a clearly defined length, which is essential information if the track is looped or concatenated with another track.	
FF 51 03 *tt tt tt*	**Set Tempo**	
	This sets the tempo in microseconds per quarter note. This means a change in the unit-length of a delta-time tick. (note 1) If not specified, the default tempo is 120 beats/minute, which is equivalent to *tttttt*=500000	
	tt tt tt	New tempo, in us/quarter-note 24-bit binary
FF 54 05 *hh mm ss fr ff*	**SMTPE Offset**	
	This (optional) event specifies the SMTPE time at which the track is to start. This event must occur before any non-zero delta-times, and before any MIDI events. In a format 1 MIDI file, this event must be on the first track (the tempo map).	
	hh mm ss fr	hours/minutes/seconds/frames in SMTPE format this must be consistent with the message MIDI Time Code Quarter Frame (in a particular, the time-code type must be present in *hh*)
	ff	Fractional frame, in hundredth's of a frame
FF 58 04 *nn dd cc bb*	**Time Signature**	
	Time signature of the form: *nn*/2^*dd* eg: 6/8 would be specified using *nn*=6, *dd*=3 The parameter *cc* is the number of MIDI Clocks per metronome tick. Normally, there are 24 MIDI Clocks per quarter note. However, some software allows this to be set by the user. The parameter *bb* defines this in terms of the number of	

	1/32 notes which make up the usual 24 MIDI Clocks (the 'standard' quarter note).	
	nn	Time signature, numerator
	dd	Time signature, denominator expressed as a power of 2. eg a denominator of 4 is expressed as *dd*=2
	cc	MIDI Clocks per metronome tick
	bb	Number of 1/32 notes per 24 MIDI clocks (8 is standard)
`FF 59 02` *sf mi*	**Key Signature**	
	Key Signature, expressed as the number of sharps or flats, and a major/minor flag. 0 represents a key of C, negative numbers represent 'flats', while positive numbers represent 'sharps'.	
	sf	number of sharps or flats -7 = 7 flats 0 = key of C +7 = 7 sharps
	mi	0 = major key 1 = minor key
`FF 7F` *<len> <id> <data>*	**Sequencer-Specific Meta-event**	
	This is the MIDI-file equivalent of the System Exclusive Message. A manufacturer may incorporate sequencer-specific directives into a MIDI file using this event.	
	<len>	length of *<id>*+*<data>* (variable length quantity)
	<id>	1 or 3 bytes representing the Manufacturer's ID This value is the same as is used for MIDI System Exclusive messages
	<data>	8-bit binary data

Footnotes

Note 1

In the case where the value of *<division>* (in the header chunk) defines delta-time units in 'ticks per quarter note' (MSbit=0), a change in tempo means a change in the length of a unit of delta-time.

In the case where *<division>* MSbit=1, and the 'ticks' are defined in absolute terms (ticks/frame and frames/second), it is not clear from the specification what effect a new tempo should have.

midi_system_exclusive.html Contents