Standard MIDI File

craig@ccrma.stanford.edu

Actual MIDI file

```
4d 54 68 64 00 00 00 06 00 00 00 01 00 80 4d 54 72 6b 00 00 00 8c 00 ff 58
04 04 02 30 08 00 ff 59 02 00 00 00 90 3c 28 81 00 90 3c 00 00 90 3c 1e 81
00 90 3c 00 00 90 43 2d 81 00 90 43 00 00 90 43 32 81 00 90 43 00 00 90 45
2d 81 00 90 45 00 00 90 45 32 81 00 90 45 00 00 90 43 23 82 00 90 43 00 00
90 41 32 81 00 90 41 00 00 90 41 2d 81 00 90 41 00 00 90 40 32 40 90 40 00
40 90 40 28 40 90 40 00 40 90 3e 2d 40 90 3e 00 40 90 3e 32 40 90 3e 00 40
90 3c 1e 82 00 90 3c 00 00 ff 2f 00
```

Standard MIDI File Structure

MIDI file contains a header “chunk” and one or more track chunks.

Header Chunk

- Basic information about tracks and timing units.

Track Chunk

- Tracks contain MIDI events

[Track Chunk]

- Multiple tracks can be used for simultaneous parts or a sequence of “songs”.

[Track Chunk]

• • •

Official specifications: http://www.midi.org/techspecs

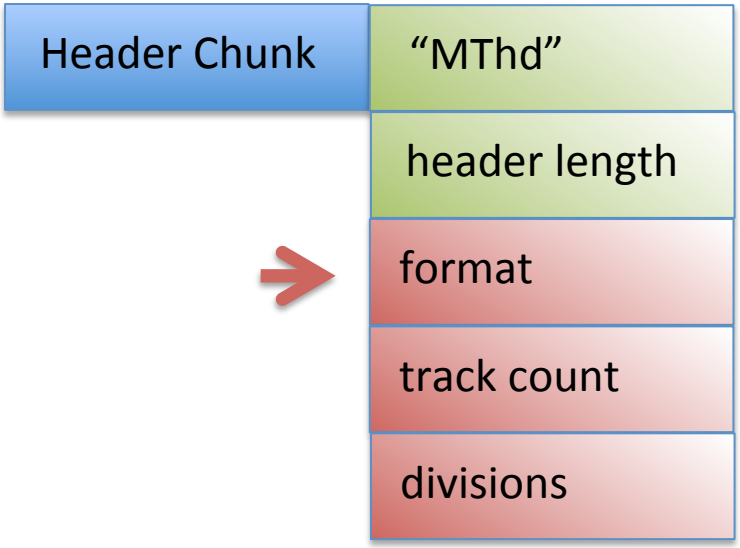
Header Chunk Bytes

bytes	meaning
4	0x4d546864
4	Bytes in header after this field (6).
2	Track configuration
2	# of tracks to follow.
2	Ticks per quarter note (usually)

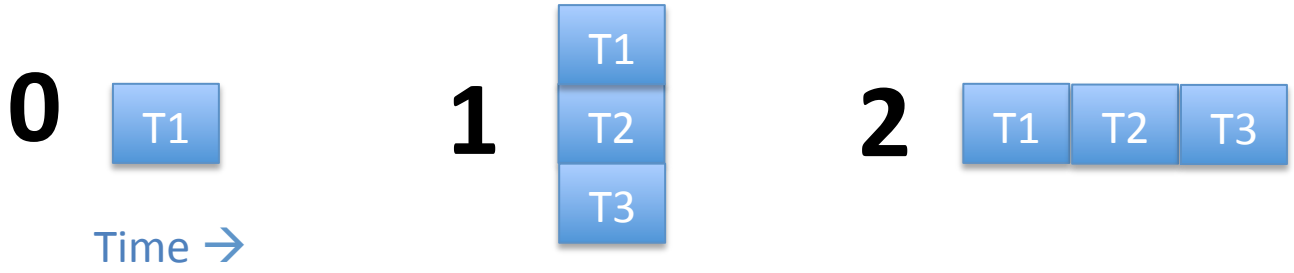
- All MIDI file multi-byte integers are big-endian (MIDI Pitch-bend message is an exception)

Header Chunk Format

- Format parameter indicates the configuration of the track data

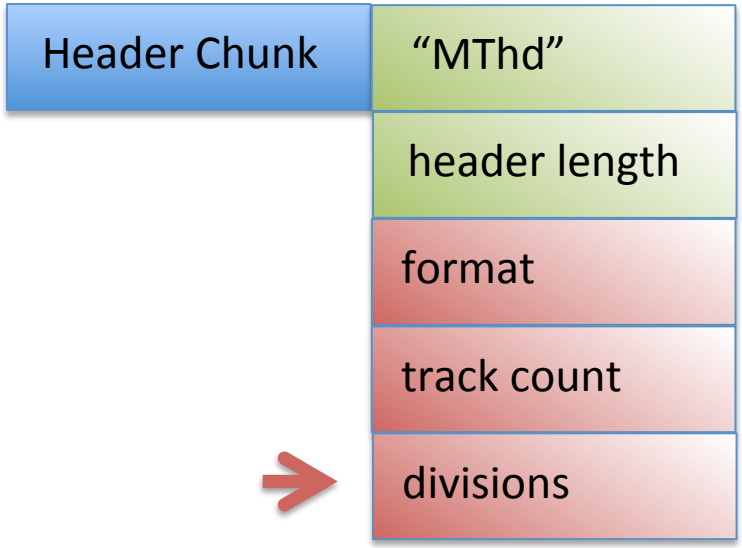


- “**type 0**” indicates a single track to follow (containing all parts interleaved)
- “**type 1**” indicates one or more tracks to be played in parallel.
- “**type 2**” indicates one or more tracks to be played in series.



Header Chunk Divisions

- Divisions parameter usually indicates the number of track ticks per quarter note.



Symbolic Time

- If top bit is 0, then *divisions* is the number of ticks per quarter note.
  Maximum number of tpq is 2^15-1= 32767

Literal Time

- If top bit is 1, then SMPTE time: bottom byte is ticks per frame and top byte is negative (2's compliment) frame rate per second.

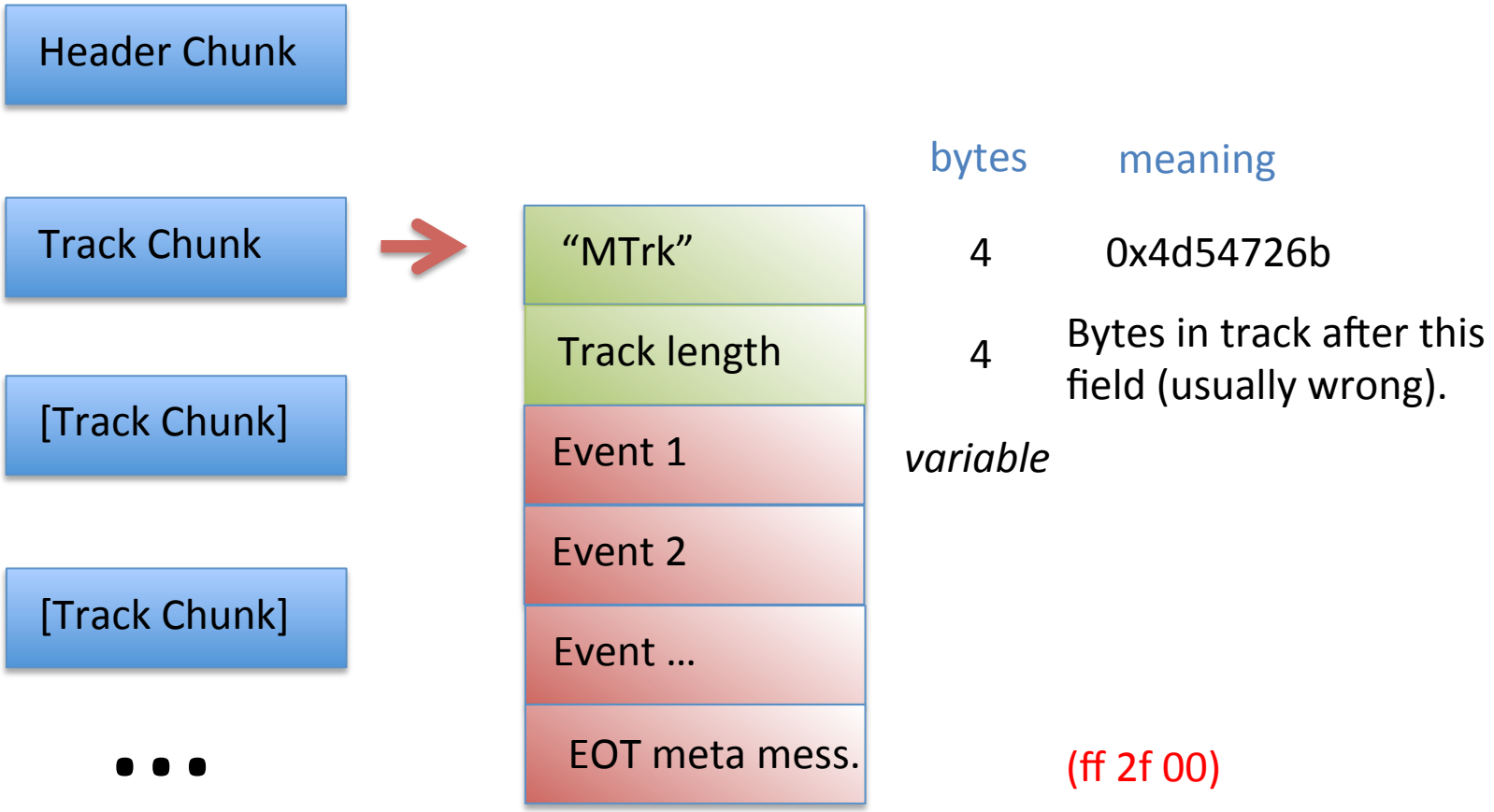
- Tempo meta messages ignored when using SMPTE

1111,1111 = -1
1111,1110 = -2
1111,1101 = -3

1110,1000 = -24 film
1110,0111 = -25 PAL
1110,0011 = -29.97 NTSC
1110,0010 = -30 ATSC

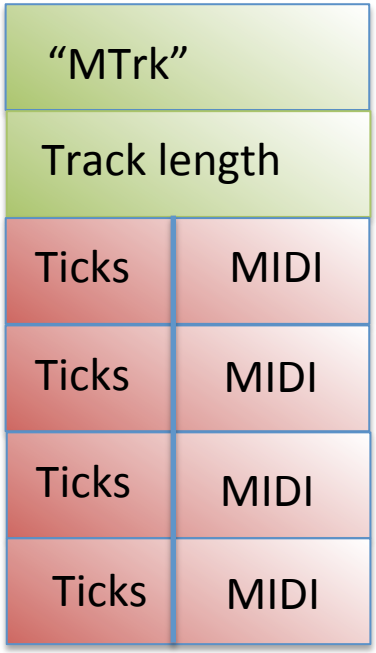
My favorite: 0xE728
25 frames/sec with
40 subframes/frame
1 tick = 1 millisecond

Header Track Chunk



Track Events

- All track events have two components:



Delta Time

- Ticks are integers called "delta times"
- Number of time units since last MIDI message was sent (previous event in track).
- Time units are symbolic (related to TPQ divisions in header), or absolute (SMPTE time code).
- Delta times are encoded as Variable Length Values.

MIDI message

- Literal MIDI bytes to be sent to synthesizer
- or "Meta message" starting with hex FF.

Variable Length Values

- Delta time stamps are compressed using Variable-Length Values system
- If a number is less than 128, then one byte is used for the time stamp
- If a number is greater than 128, the number is cut into 7-bit units, and the most significant bit is used as a continuation bit.
- MIDI files VLVs can be up to 5 bytes long as long as they unpack to 4 bytes.

VLVs (2)

Example: convert 123,456 into VLV bytes

Binary form:	0000 0000 0000 0001 1110 0010 0100 0000
Grouped every 7 bits:	0000000 0000000 0000111 1000100 1000000
Drop leading 0 groups:	0000 0000000 0000111 1000100 1000000
Add continuation bit:	10000111 11000100 01000000
3-byte Hex result:	**87 C4 40**

Example: convert VLV E0 82 7F into integer:

Binary form:	1110 0000 1000 0010 0111 1111
Remove cont. bits:	1110 0000 1000 0010 0111 1111
Regroup:	1 1000 0000 0001 0111 1111
To hex:	1 8 0 1 7 F
To dec:	1,573,247

Meta Messages

- Meta messages are non-MIDI messages stored in a MIDI file.
- Format is (1) hex “FF”,

“type” is the type of meta message. This value is one byte long.
“length” is the number of bytes of data to follow

Meta Message Types

FF **00** 02	**Sequence number.** For type-2 MIDI files. Must come before any non-zero delta times.
FF **01** len txt	**Text event.** Text describing anything.
FF **02** len txt	**Copyright notice.**
FF **03** len txt	**Track name.**
FF **04** len txt	**Instrument name.**
FF **05** len txt	**Lyric.** Syllable which begins at the event's time.
FF **06** len txt	**Marker.** Usually only in first track. Name of start of section (such as "Verse", rehearsal letter, etc.
FF **07** len txt	**Cue point.** Stage direction for current time.
FF **20** 01	**MIDI channel prefix.** Force MIDI messages to given channel.
FF **2F** 00	**End of Track.** All tracks must end with this meta message.

Meta Message Types (2)

FF **51** 03

Tempo. Numbers of microseconds per quarter note. Value is 3-bytes, big-endian. Example: 500000 (0x07a120) is quarter note = 120 BPM (500000 microseconds = 0.5 seconds therefore 120 ½ seconds in a minute).

FF **54** 05

SMPTE offset. Starting time offset of track: Hours, minutes, seconds, frames, fractional frames (1/100th frames). The message must occur in the first track (when type-1 MIDI file); otherwise ignored.

FF **58** 04

Time signature. Numerator, log2 Denominator, clock ticks per beat, 32nd notes per beat (redefine beat to other than quarter note).

FF **59** 02

Key signature. Sharps/flats and mode.

-7 (11111001, 0xF9) = 7 flats
-6 (11111010, 0xFA) = 6 flats
-5 (11111011, 0xFB) = 5 flats
-4 (11111100, 0xFC) = 4 flats
-3 (11111101, 0xFD) = 3 flats
-2 (11111110, 0xFE) = 2 flats
-1 (11111111, 0xFF) = 1 flat

Mode: 0 = major key, 1 = minor key. Dorian?

0 = 0 sharps/flats
1 = 1 sharp
2 = 2 sharps
3 = 3 sharps
4 = 4 sharps
5 = 5 sharps
6 = 6 sharps
7 = 7 sharps

FF **7F** len dta System Exclusive style meta-event.

Running Status

- Dangerous shorthand in MIDI communication protocol and standard MIDI files
- If the command byte of the previous MIDI message (in the track) is the same as the current one, then the command byte can be dropped.

MIDI messages without running status:

0x90 60 127 0x90 60 0 0x90 61 127 0x90 61 0

MIDI messages without running status:

0x90 60 127 60 0 61 127 61 0

Interpretation:

0x90 60 127 0x90 60 0 0x90 61 127 0x90 61 0

Detection algorithm: if a command byte is expected (>= 0x80) but find data byte (< 0x80), then running status is activated.

Parsing a MIDI file

```
4d 54 68 64 00 00 00 06 00 00 00 01 00 80 4d 54 72 6b 00 00 00 8c 00 ff 58

04 04 02 30 08 00 ff 59 02 00 00 00 90 3c 28 81 00 90 3c 00 00 90 3c 1e 81

00 90 3c 00 00 90 43 2d 81 00 90 43 00 00 90 43 32 81 00 90 43 00 00 90 45

2d 81 00 90 45 00 00 90 45 32 81 00 90 45 00 00 90 43 23 82 00 90 43 00 00

90 41 32 81 00 90 41 00 00 90 41 2d 81 00 90 41 00 00 90 40 32 40 90 40 00

40 90 40 28 40 90 40 00 40 90 3e 2d 40 90 3e 00 40 90 3e 32 40 90 3e 00 40

90 3c 1e 82 00 90 3c 00 00 ff 2f 00
```