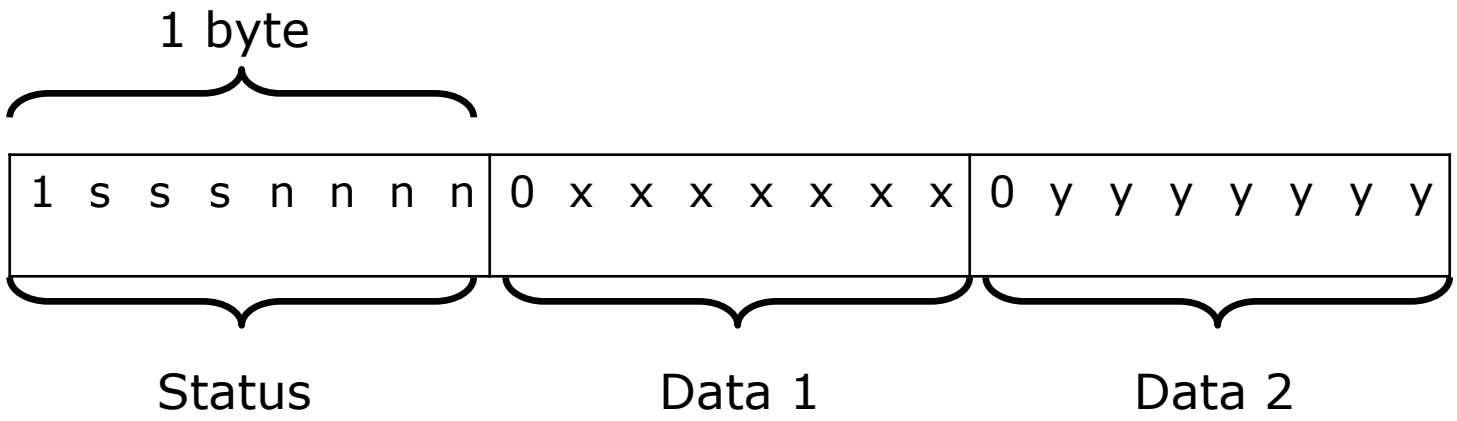
MIDI Code

Juan P Bello

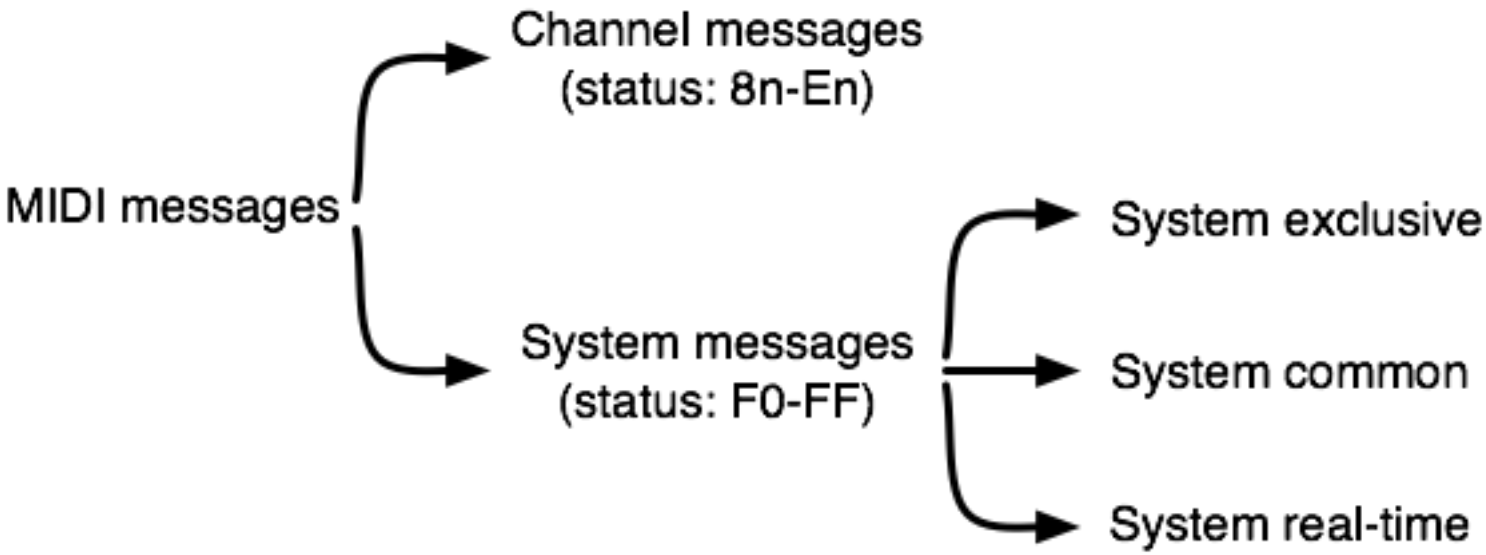
MIDI Code: the message format

- 2 types of MIDI message bytes: the status byte and the data byte
- Status bytes always begin with 1, and data bytes with 0. That leaves only 7 bits per byte to represent the message (128 possible values).
- MIDI messages begin with the status byte, where 3 bits (*sss*) are used to denote the type of message, and 4 bits (*nnnn*) to denote the channel number to which the message apply (max. 16 channels).



MIDI Messages

- There are two main types of MIDI messages: channel and system



- As their names indicate they are addressed to individual channels or the whole system (exception: “omni on” channel messages)

Channel messages

- MIDI channel numbers (n) are referred as 1 to 16, while in reality they are represented by binary values 0 to 15 (0-F).
- Example: the status byte of a note off message for channel 7 is “86”

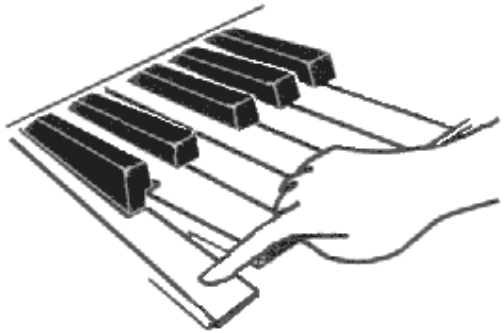
Message	Status	Data 1	Data 2
Note off	8n	Note number	Velocity
Note on	9n	Note number	Velocity
Polyphonic aftertouch	An	Note number	Pressure
Control change	Bn	Controller number	Data
Program change	Cn	Program number	-
Channel aftertouch	Dn	Pressure	-
Pitch wheel	En	LSbyte	MSbyte

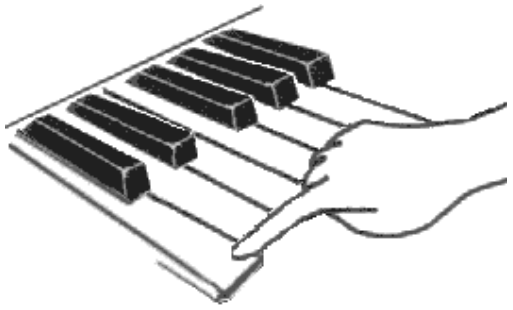
Note on / Note off (1)

- They make the bulk of the information commonly sent through a MIDI channel

Message	Status	Data 1	Data 2
Note off	8n	Note number	Velocity
Note on	9n	Note number	Velocity

- "Note on" triggers a musical note, "note off" turns it off

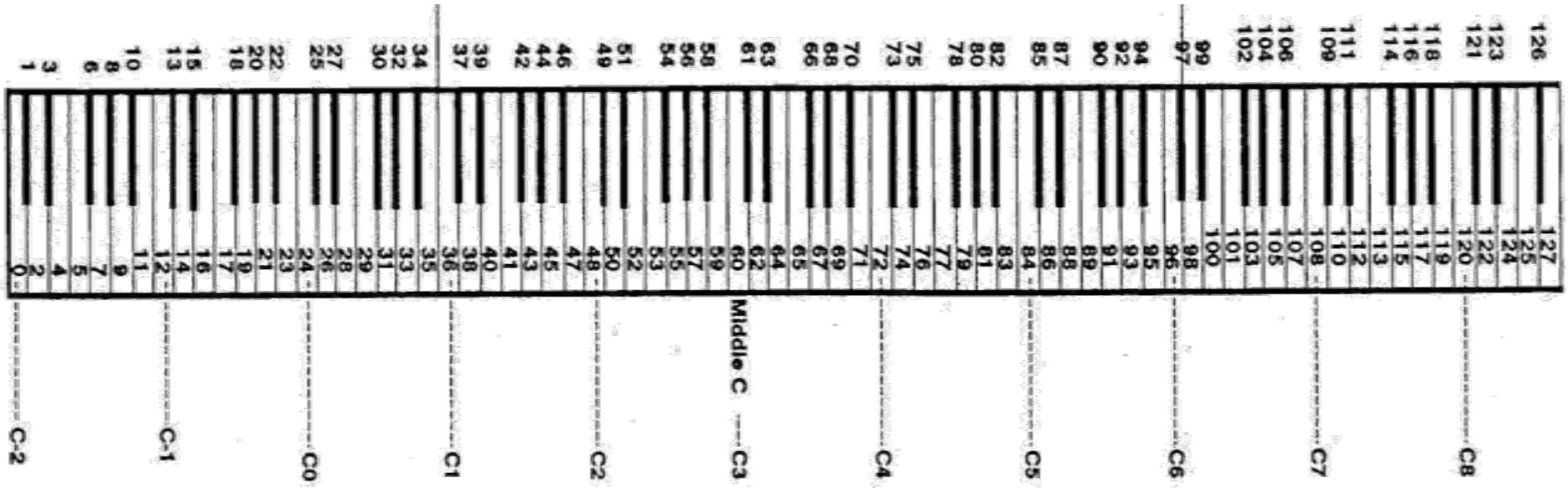




- All notes MUST be turned off (otherwise they'll sound indefinitely)

Note on / Note off (2)

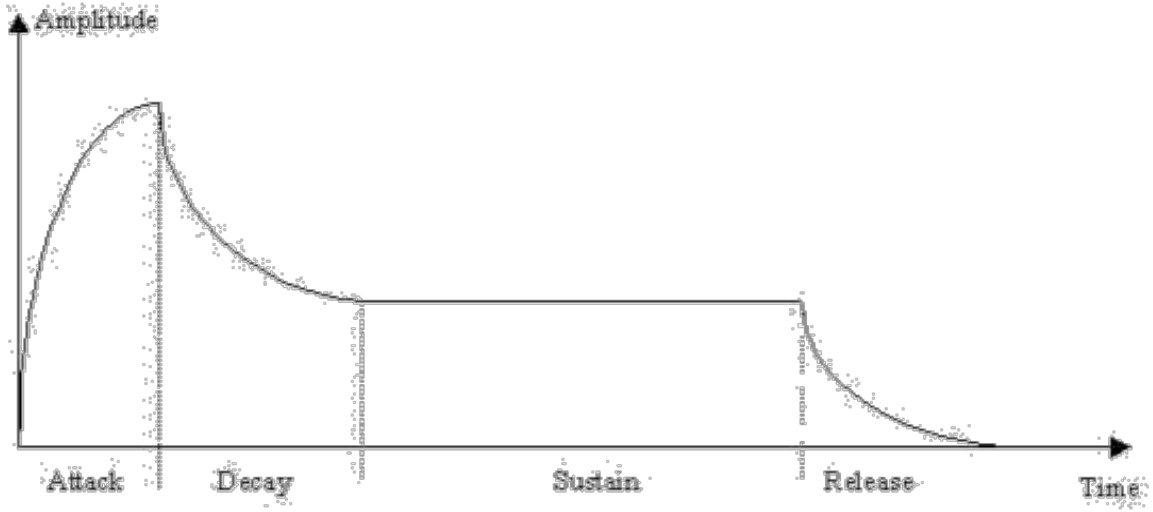
- There are 128 (0-127) possible note values (~10 octaves) mapped to the chromatic western music scale.



- Commonly, middle C is mapped to MIDI's C-3 (note number 60, 6th octave).

Note on / Note off (3)

- Note on messages are also associated to a “velocity” value, characterizing how hard the key was hit.
- Note on velocity can be used to control volume and timbre of a sound (e.g. by controlling the scaling of an envelope generator)



- The mapping between velocity and the parameter it controls is often logarithmic.
- Note off velocity relates to the speed at which a note was released
- It could be used to affect a sound, but it is not normally used.

Note on / Note off (4)

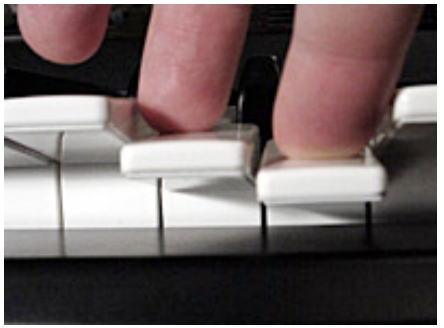
- Note on, velocity zero is equivalent to note off.
- It is convenient when large amounts of data are sent to the MIDI bus (e.g. a high-polyphony chord)
- Normally we will need 6 bytes for each note of a chord: [9n] [pitch][velocity] and [8n][pitch][velocity]
- Instead we can clutter note on and off messages together: [9n] [pitch][vel][pitch][vel]…[pitch][vel]
- This is known as running status
- For a 4-note chord it means 17 bytes are transmitted rather than 24 bytes (assuming running status remains unchanged).

Aftertouch

- Key pressure messages are called aftertouch
- It refers to the amount of pressure placed on a key at the bottom of its travel (triggering performance parameters, e.g. vibrato)

Message	Status	Data 1	Data 2
Polyphonic aftertouch	An	Note number	Pressure
Channel aftertouch	Dn	Pressure	-

- Polyphonic key pressure transmits a separate value per key (thus requiring separate sensors)
- This is expensive as most players do not maintain constant pressure on the bottom of the key
- Most instruments use a single sensor, thus one message is sent with the approx. total pressure sensed (channel aftertouch)



Control change (1)

- MIDI is also capable of transmitting orders for the controllers (e.g. pedals, switches and wheels) of a given device
- All these controllers are addressed by using the same status byte, with the first data byte determining the specific controller

Message	Status	Data 1	Data 2
14-bit controllers MSbyte	Bn	00-1F (controllers)	Data
14-bit controllers LSbyte	Bn	20-3F (same order)	Data
7-bit controllers/switches	Bn	40-65	Data
Undefined	Bn	66-77	-
Channel mode	Bn	78-7F	Data

- The number of controllers has augmented significantly since the introduction of MIDI
- Thus, although not originally specified, the MMA administers an agreement as to which ID corresponds to which controller

Control change (2)

Controller number (hex)	*Function*
00	Bank select
01	Modulation wheel
02	Breath controller
03	Undefined
04	Foot controller
05	Portamento time
06	Data entry slider
07	Main volume
08	Balance
09	Undefined
0A	Pan
0B	Expression controller
0C	Effect control 1
0D	Effect control 2
0E–0F	Undefined
10–13	General purpose controllers 1–4
14–1F	Undefined
20–3F	LSbyte for 14 bit controllers (same function order as 00–1F)

The first 64 numbers are used for 32 physical controllers at greater control resolution

Thus 2 data bytes (14 bits) are used to represent the controller's position (16384 possible values)

Control change (3)

Controller number (hex)	*Function*
40	Sustain pedal
41	Portamento on/off
42	Sostenuto pedal
43	Soft pedal
44	Legato footswitch
45	Hold 2
46–4F	Sound controllers
50–53	General purpose controllers 5–8
54	Portamento control
55–5A	Undefined
5B–5F	Effects depth 1–5
60	Data increment
61	Data decrement
62	NRPC LSbyte (non-registered parameter controller)
63	NRPC MSbyte
64	RPC LSbyte (registered parameter controller)
65	RPC MSbyte

7-bit controllers use only 1 data byte for their position

On/off switches are represented with values 00-3F for off, and 40-7F for on.

Controller messages (1)

- Control messages (status byte Bn) can be implemented in many ways in sound generators
- Although standard definitions exist for many of these controllers, it is possible to remap them in sequencers and sound devices
- 14-bit continuous controllers are rarely used. Oftentimes only the MSbyte of the 32 14-bit controllers is used
- The internal mixing of voices can be controlled (independent from velocity) using volume (07) and pan (0A) channel controllers.

Controller messages (2)

- A number of alternative expressive controllers have been proposed over the years.
- Examples include so-called breath controllers (controller ID 02), which were originally intended for wind controllers but that are often used individually to add expressiveness.
- The sensed blowing pressure can be applied to envelope generators or modulation parameters to affect the sound.

Sound controllers (1)

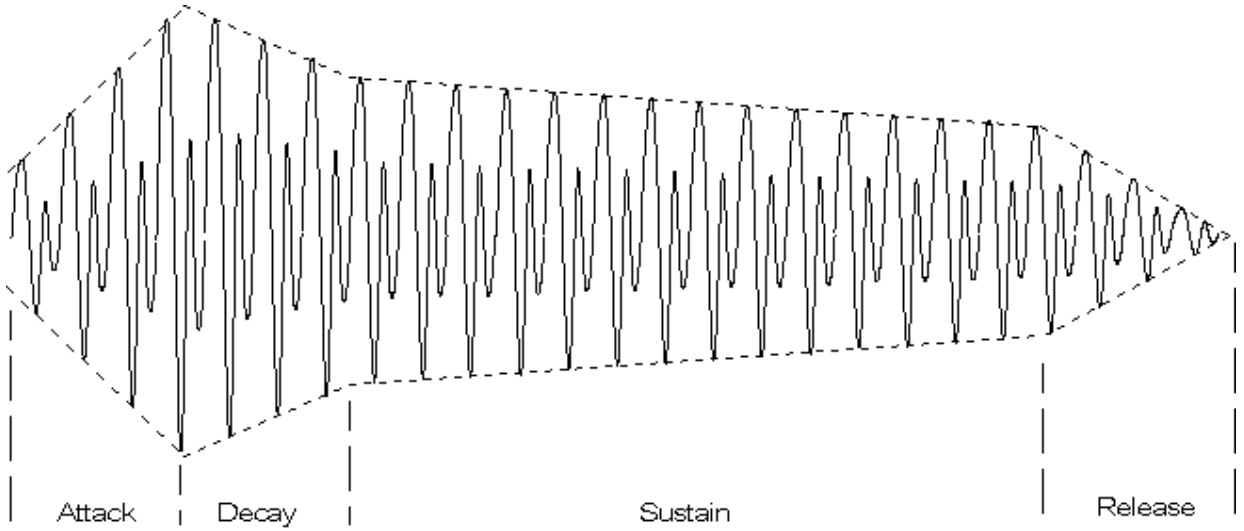
- Sound and effect controllers provide control over the sound quality of a device and the parameters of built-in effects
- They are device dependent, intended for real-time and high-level (abstract) control of the synthesis and effect processes

Controller #	Function	Controller #	Function
46	Sound variation	5B	External effects depth
47	Timbre/harmonic content	5C	Tremolo depth
48	Release time	5D	Chorus depth
49	Attack time	5E	Detune depth
4A	Brightness	5F	Phase depth
4B-4F	No default		

- Sound variations refer to pre-programmed variants of a basic sound (e.g. piano, lid open/closed, honky tonk).

Sound controllers (2)

- Timbre and brightness are used to affect the spectral content of the sound (e.g. by controlling filtering characteristics)
- Envelope controllers modify the attack and release time of a sound (following the ADSR envelope approximation).



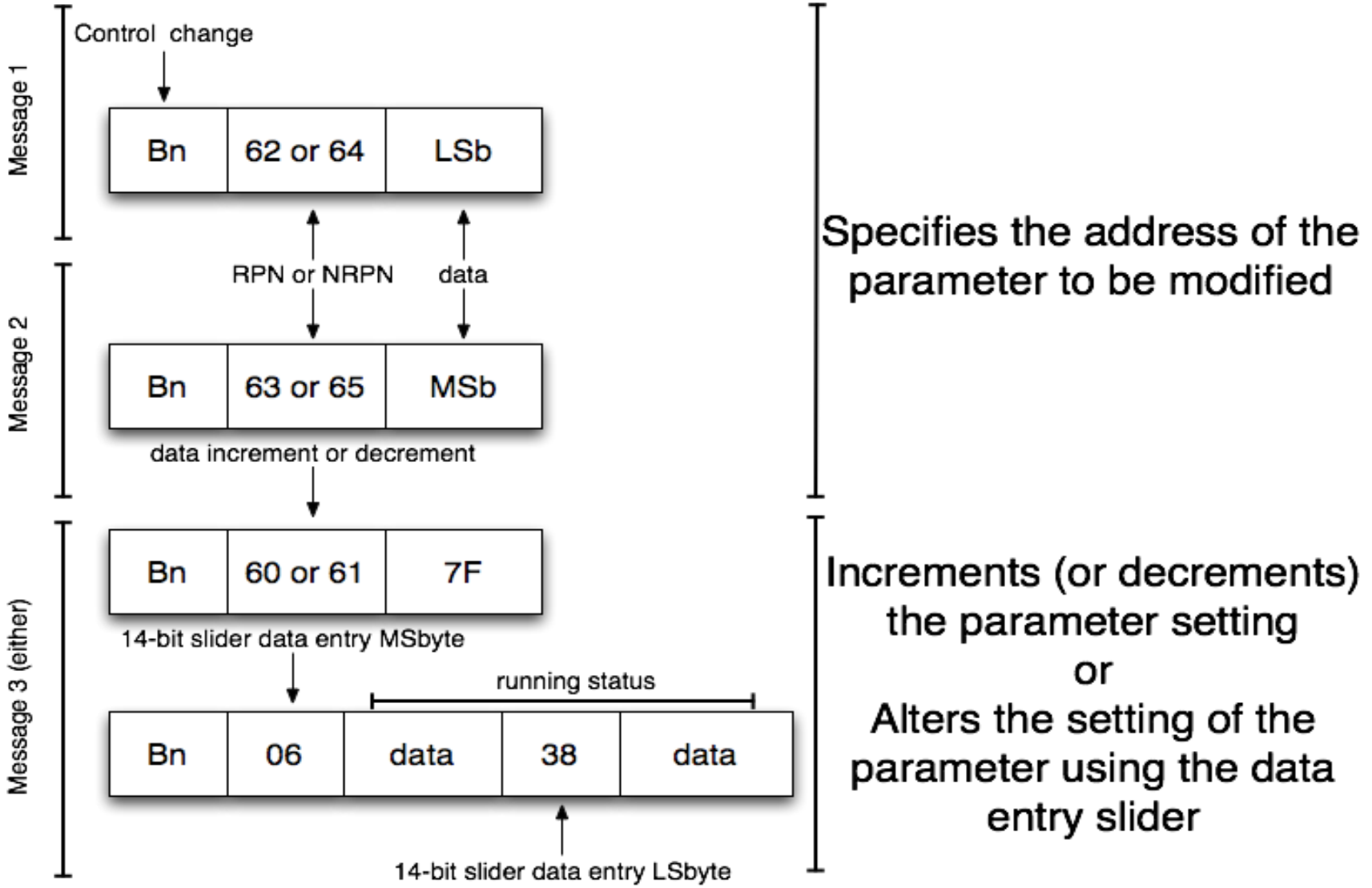
- Both progressively shorter (data byte < 40) and longer (data byte > 40) times can be assigned

RPN and NRPN (1)

- Registered (RPN) and non-registered (NRPN) parameter numbers are specific control change messages that allow for the control of the internal parameters of a voice patch
- Any aspect of a voice (e.g. velocity sensitivity of an envelope generator, range of modulation or bending, etc) can be modified remotely via these messages
- RPNs are universal and should be registered with the MMA while NRPNs can be manufacturer specific

RPN	Parameter
00 00	Pitch bend sensitivity
00 01	Fine tuning
00 02	Coarse tuning
00 03	Tuning program select
00 04	Tuning bank select
7F 7F	Cancels RPN or NRPN

RPN and NRPN (2)



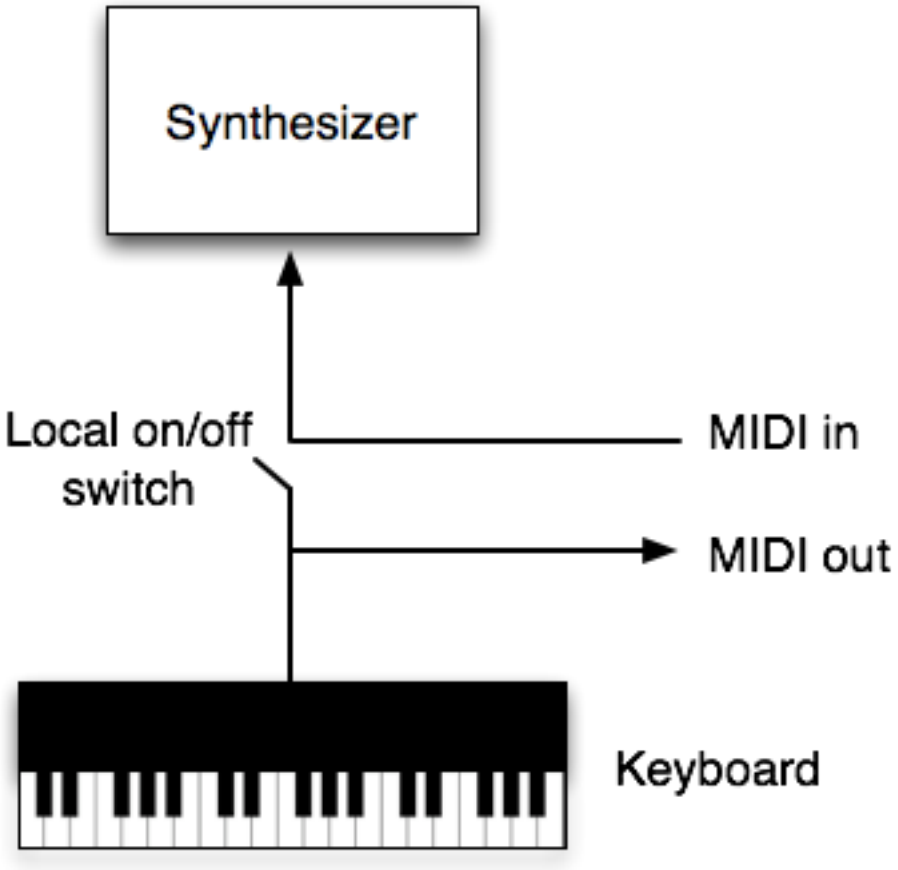
Channel mode messages (1)

- Channel Mode Messages are a special case of Control Change Messages (status byte Bn).
- They set the mode of operation of the instrument receiving on channel n.
- A change of channel mode should turn all notes off automatically

Status	Index	Argument	Description
Bn	78	00	All Sounds Off
Bn	79	7F	Reset All Controllers
Bn	7A	7F on/00 off	Local Control On/Off
Bn	7B	00	All Notes Off
Bn	7C	00	Omni Mode Off (All Notes Off)
Bn	7D	00	Omni Mode On (All Notes Off)
Bn	7E	channels	Mono Mode On (Poly Mode Off)
Bn	7F	00	Poly Mode On (Mono Mode Off)

Channel mode messages (2)

- All sounds off turns off all sound generators
- Reset all controllers returns a device to its standard settings
- Local on/off breaks the link between the instrument's keyboard and its sound generator
- Omni off (default) sets the instrument to receive on the appropriate channel
- Omni on sets the instrument to receive in all channels
- Mono mode sets the instrument to monophonic
- Poly mode sets it to polyphonic



Program change

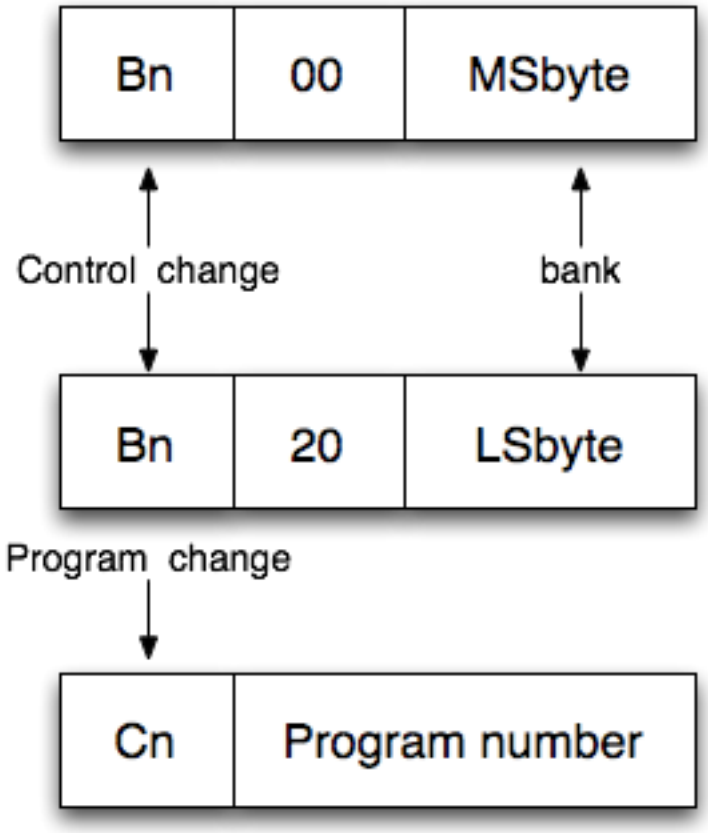
- Program change is used to change between patches or presets of an instrument: the set-up of stone generators and the way they are interconnected
- The message is channel specific
- There is only one byte of data used (only 128 presets to choose from)

Message	Status	Data 1	Data 2
Program change	Cn	Program number	-

- In other devices, e.g. effect processors, the program change is used to switch between effects
- On some instruments, programs are organized in “banks” of 8, 16 or 32 presets

Voice selection

- The program change message (Cn) only allows for the selection of one of 128 voices.
- A sound generating device may allow the user to define a program change map to decide which voice correspond to which message
- Commonly, current devices have very large (>> 128) program memories
- A solution is to precede program changes with a 14-bit “bank select” control change message



Pitch bend

- The pitch wheel is the only controller with a status byte of its own.
- It uses 2 data bytes (14 bits of resolution)
- Such resolution ensures smooth changes of pitch
- It is channel specific

Message	Status	Data 1	Data 2
Pitch wheel	En	LSbyte	MSbyte

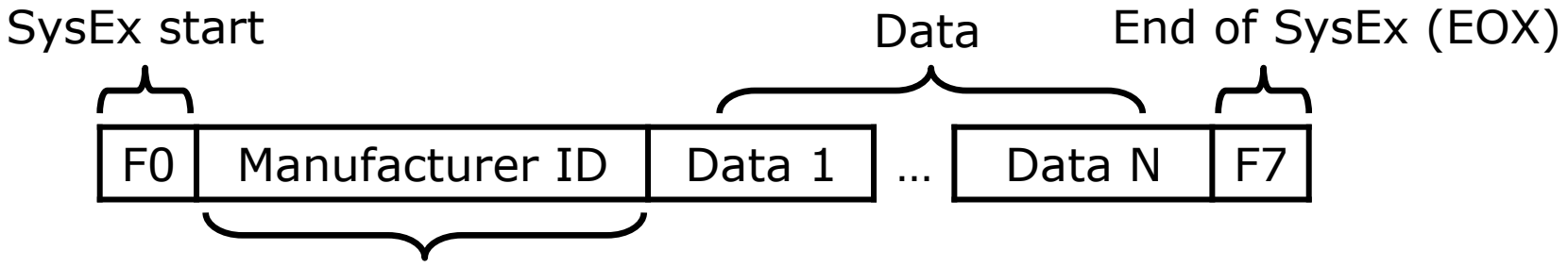
- In default position, the pitch bend value is in the middle of its range ([En] [00] [40]). This allows pitch bending both up and down.
- The controlled pitch range is set in the receiving device

System messages

Message	Status	Data 1	Data 2
System exclusive			
System exclusive start	F0	Manufacturer ID	Data … (Data)
End of system exclusive	F7	-	-
System common			
Quarter frame (MTC)	F1	Data	
Song pointer	F2	LSbyte	MSbyte
Song select	F3	Song number	-
Tune request	F6	-	
System Real-time			
Timing clock	F8	-	-
Start	FA	-	-
Continue	FB	-	-
Stop	FC	-	-
Active Sensing	FE	-	-
Reset	FF	-	-

System Exclusive (1)

- SysEx messages are used for device-specific data transfer.
- Except for SysEx for Universal Info, the standard only defines how messages begin and end.
- A return link is necessary for *Handshaking*.
- Error checking (checksum) is applied to long data dumps



Manufacturers ID

Yamaha	43
Roland	41
Akai	47

Universal information

Universal non-commercial	7D	Ed&R
Universal non-realtime	7E	Sample dumps
Universal realtime	7F	MTC

System Exclusive (2)

- Manufacturer-specific SysEx messages are mainly used for dumping/loading the settings of a device and for remotely controlling its parameters.

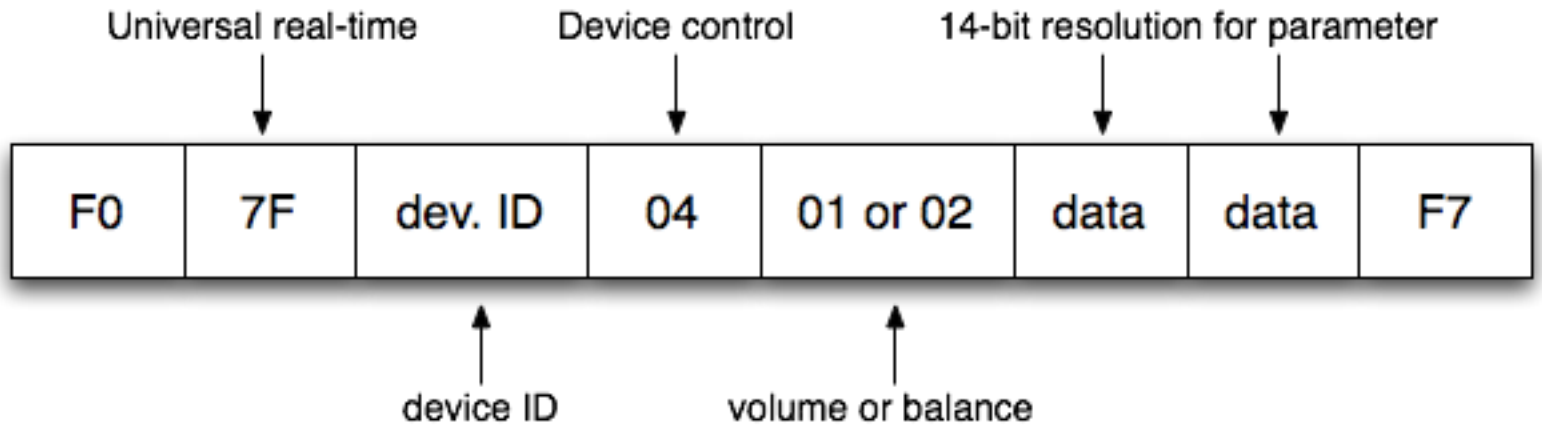
- Universal non-realtime messages include Sample/MIDI file dump, General MIDI on/off, Inquiry requests, MIDI Tuning standard.

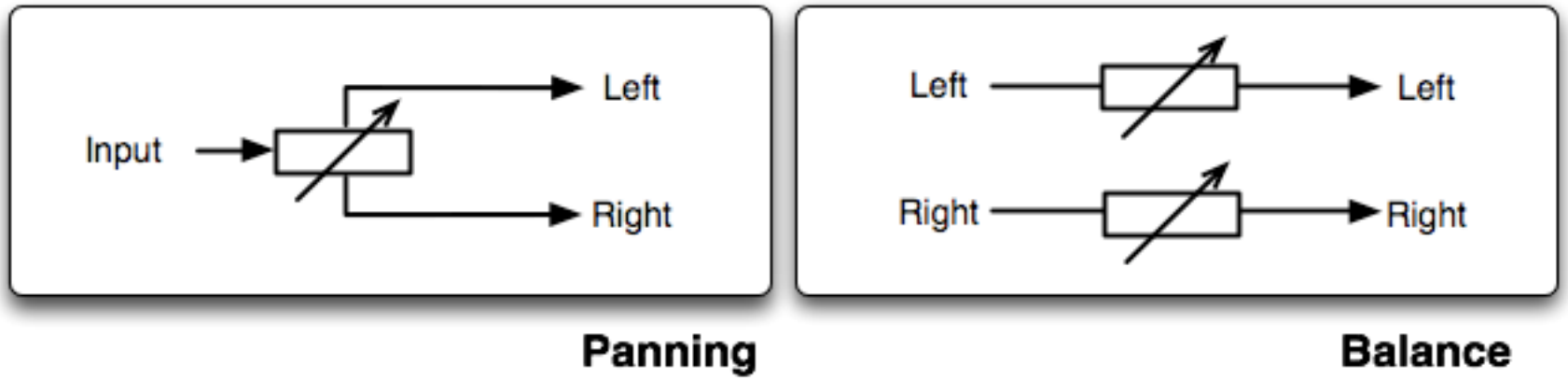
- Universal realtime messages include MIDI timecode (MTC), MIDI Machine/Show control, Master Volume/Balance control, Bar/Time-signature markers.

- Timecode is a digital code, used in the audiovisual industries, that represents time in terms of hours, minutes, seconds and frames (hh:mm:ss:ff)

System Exclusive (3)

- Example: master volume and balance messages





System common messages

- Like SysEx Universal messages, SCM are intended for the attention of all systems

Status	Data 1	Data 2	Description
F1	Data	-	Quarter-frame (MTC)
F2	LSbyte	MSbyte	Song Pointer
F3	Song number	-	Song Select
F6	-	-	Tune Request

- Song select determines which pre-recorded sequence of MIDI data (song) from a collection is to be accessed.
- Song pointer directs devices towards a particular location in a song (in beats from the beginning of the song).
- Tune request asks synthesizers to re-tune themselves to a pre-specified reference.

MIDI and synchronization

- An important function of MIDI is the handling of timing and synchronization data between devices
- MIDI sends sync info through the same data stream it uses for control
- Sync info comes in two types: music-related timing data (related to bars and beats) and timecode information, necessary to synchronize to audio-visual devices in “real” time (seconds, hours, etc).
- The first type is handled by the song pointer message in combination with system real-time messages.

Note value	# of MIDI beats	# of MIDI clocks
Whole note	16	96
Half note	8	48
Quarter note	4	24
Eight note	2	12
Sixteenth note	1	6

System real-time messages

- They control the execution of timed sequences in a MIDI system

```
System Real-Time Messages
STATUS Description          Details
=============================================================
F8     Timing Clock
FA     Start                from beginning of song
FB     Continue             from stop position
FC     Stop                 stops song's execution
FE     Active Sensing       flags active connection (3/sec)
FF     System Reset         Reset all devices to init state
```

- The MIDI clock is a single status byte (F8) to be sent 6 times per MIDI beat (defined as a sixteenth note).
- Thus it represents musical tempo rather than real time.
- It is the only MIDI byte that can interrupt the current status
- The internal clock of the receiver is incremented with each clock

MIDI timecode (MTC)

- MIDI timecode (MTC) is an alternative to MIDI clocks and song pointers when real-time synchronization is important
- It is used to distribute SMPTE/EBU timecode data thoughout a MIDI system (for synchronization with audiovisual equipment)
- It is also used to transmit information about “cue points” at which certain events are to take place
- There are two types of MTC sync messages: quarter frame messages (used to update the receiver continuously with running timecode) and full-frame messages (used for one-off updates of the MTC position)

Quarter frame messages

- The system common *quarter-frame* message can be used to transmit MTC info.
- This message consists of 1 status byte (F1) and 1 data byte.
- As timecode info is much longer than this, then we split the message into 8 separate messages transmitting the frames' LS/ MSnybbles, seconds' LS/MSnybbles, minutes' LS/MSnybbles and hours' LS/MSnybbles.
- The message type is encoded in the first data nybble (as 0-7)
- Hour information is encoded as in SysEx MTC, including frame rate info.
- Despite the message name, the MTC is updated every two frames (causing a delay at the receiver).
- At 30 fps, and 8 messages to represent a frame, we need to send 120 messages per second.
- If transmitted continuously this takes ~7.7% of the data bandwidth
- SysEx full-frame messages can be used to avoid this.

Full frame messages

F0	7F	Chan	01	subID	hh	mm	ss	ff	F7

- A SysEx Full-frame message transmits MTC: 7F is the universal realtime identifier, channel number (set to 7F for the whole system), 01 identifies the message as a MTC message, and a subID of 01 refers to full-frame message.
- Hours are coded as [0 qq ppppp], where *qq* represents the time-code frame rate (in fps):

  00 = 24 frame (used in films)

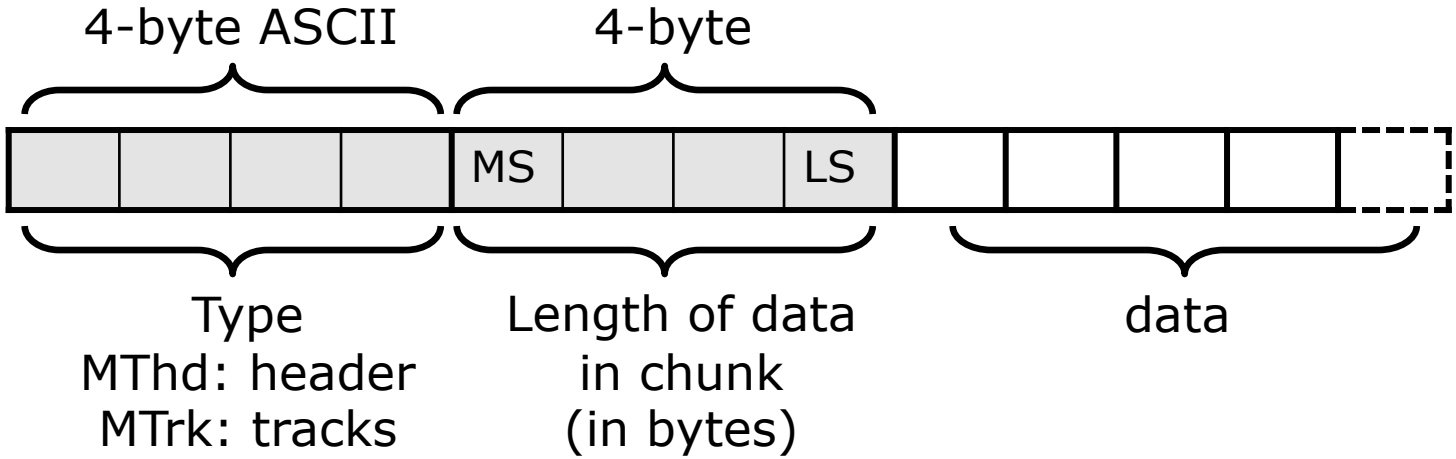
  01 = 25 frame (EBU: PAL and SECAM TV pictures in Europe/Australia)

  10 = 30 drop-frame (SMPTE drop frame: NTSC color TV in the US/Japan)

  11 = 30 non-drop-frame (SMTPE: monochrome US TV pictures)
- *ppppp* represents hours from 00 to 23. Minutes, seconds and frames are represented by a byte each
- SysEx can also transmit MTC origination information (user bits) and MTC cue messages.

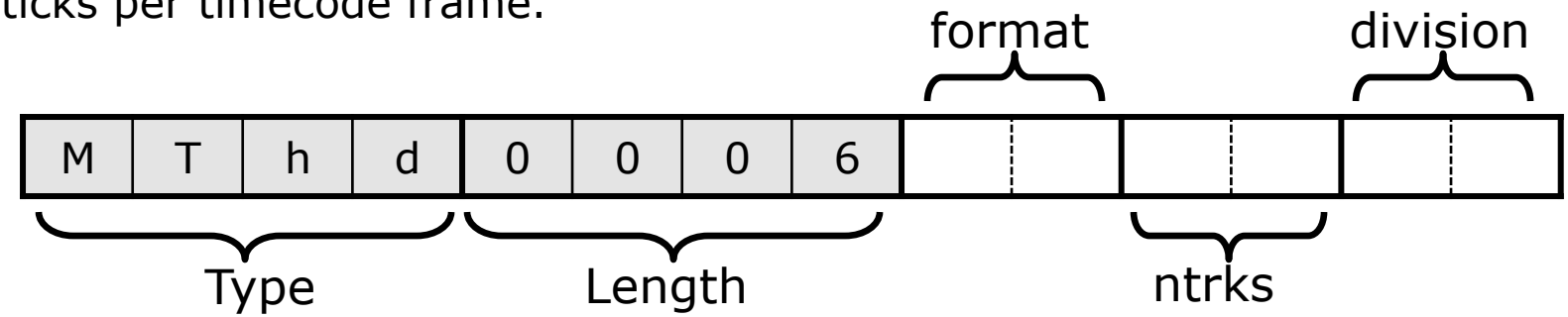
Standard MIDI files (1)

- Standard MIDI files were designed to allow exchange of sequenced data between devices / SW sequencers.
- These files represent data as events belonging to individual sequencer tracks, plus info such as track/instrument names and time signatures.
- There are 3 types of MIDI files for representing: single-track data (type 0), synchronous multi-track data (type 1) and asynchronous multi-track data (type 2).
- Data is organized as bytes grouped into *header* and *track chunks*.

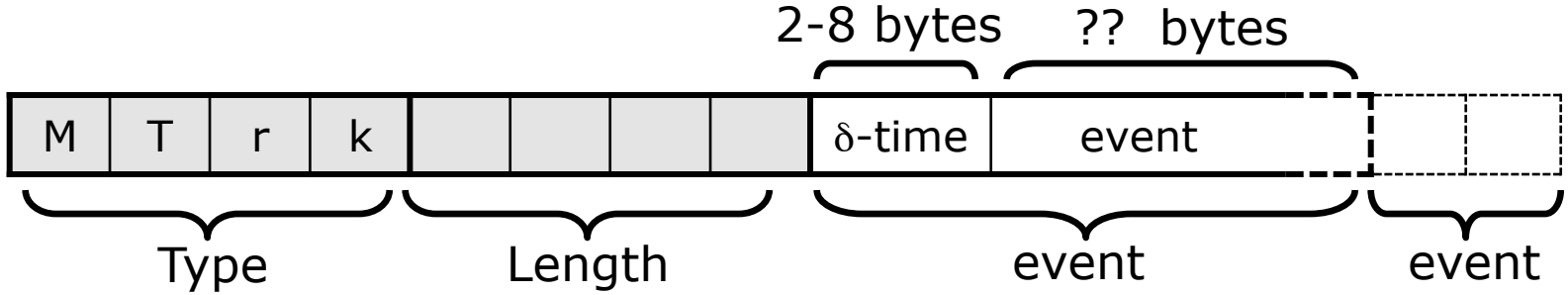


Standard MIDI files (2)

- Header chunk: *format* defines the MIDI file type (0, 1 or 2), *ntrks* defines the number of track chunks and *division* defines the timing format.
- The timing format is defined by the MSB of the 2-byte *division* word. *0* indicates a division of ticks per quarter note, while *1* indicates a division of ticks per timecode frame.



- Track chunks contains strings of MIDI events, each labeled with a *δ-time* (ticks since the last event) at which the event occurs.
- MIDI events can be channel messages, SysEx and meta-events (containing labels and internal data)



Useful References

- Francis Rumsey and Tim McCormick (2002). "Sound and Recording: An Introduction", Focal Press.
  - Chapter 13: MIDI

- MIDI Manufacturers Association (2002). The complete MIDI 1.0 detailed specification (www.midi.org)