Chapter 5: Music Synthesis Technologies

For the presentation of sound, music synthesis is as important to multimedia system as for computer graphics to the presentation of image. In this chapter, the basic principles of music synthesis technologies will be introduced and the industry widely adopted MIDI protocol for representing musical performance information will be described.

Principles of Music Synthesis

Characteristic of Music Sounds

Generally musical notes can be divided into three segments according to their time waveforms: the attack, the steady state and the decay.

States of a typical music note
Principles of Music Synthesis

The spectrum of sound signal from a musical instrument is generally a *harmonic rich* signal, e.g., cello, clarinet and trumpet.

During the steady-state portion, the note may be altered by tremolo (an amplitude modulation) or vibrato (a frequency modulation).

There are other musical instruments such as a bass drum, its sound is more or less resembling noise with certain underlying structures.

![Music signal spectrum is harmonic rich](image)

**Music Synthesis Techniques**

- Additive synthesis
- Nonlinear synthesis
- Physical modeling
- Wavetable synthesis
Principles of Music Synthesis

Additive Synthesis

\[ s(t) = e(t) \sum_{m=1}^{M} a_m(t) \cos(2\pi f_m t + \phi_m) \]

- need many oscillators

- \( a_m(t) \) is the time-varying amplitude of the \( m \text{th} \) sinusoid

- \( f_m \) is the frequency of the \( m \text{th} \) sinusoid

- \( \phi_m \) is the phase of the \( m \text{th} \) sinusoid

- \( e(t) \) is the time-varying magnitude envelope of the synthesis signal

Structure of Additive Synthesis
Music Synthesis

Principles of Music Synthesis

Additive Synthesis

Oscillator Implementation

\[ f(t) = \cos(2\pi ft + \phi) \]

(a) Use a mathematical expansion of cosine function

\[
\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots
\]

- too slow for real-time applications!

(b) Table look-up oscillator

Use a wave table to store one period of waveform and index it according to the specified pitch frequency.

To map linearly a value in the range 0 to \(2\pi\) into the range 0 to \(L\), we have

\[
\frac{\theta}{2\pi} = \frac{\text{index}}{L}
\]

index can be obtained through truncation, rounding or linear interpolation. \(L\) is the table size

In order to control the frequency of an oscillator in a time-varying manner, we need to control its instantaneous frequency,
Principles of Music Synthesis

Additive Synthesis

Oscillator Implementation

$$\theta(n + 1) = \theta(n) + \Delta \theta(n)$$

$$\theta(n) = \frac{2\pi fn}{f_s} + \phi$$

$$y(n) = A(n) \cos[\theta(n)]$$

$$\Delta \theta(n) = 2\pi \left[ \frac{\Delta f(n)}{f_s} \right]$$

$$\Delta f(n)$$ is the instantaneous frequency at time $$n$$.

$$\frac{\Delta \theta(n)}{2\pi} = \frac{\Delta \text{index}(n)}{L}$$

$$y(n) = A(n) \, \text{table}[\text{index}(n)]$$

$$\text{index}(n + 1) = \text{index}(n) + \Delta \text{index}(n)$$

$$\Delta \text{index}(n) = \left[ \Delta f(n) \right] \frac{L}{f_s}$$

Table Look-up Algorithm

$$y(n) = A(n) \, \text{table}[\lfloor s(n) \rfloor]$$

$$s(n + 1) = [s(n) + I(n)] \mod \, L$$

$$\lfloor x \rfloor$$ stands for truncation of $$x$$ to the nearest lower integer value.

$$I(n)$$ is the increment to the table index value at time $$n$$, which controls the oscillator’s instantaneous frequency according to

$$I(n) = \frac{\Delta f(n)L}{f_s}$$
Principles of Music Synthesis

Additive Synthesis

Symbol representation
In general, the wavetable can store not only sinusoidal signal but also periodic signal with harmonics, but care must be taken when synthesizing the signal from table at different rates to avoid aliasing effects.

We may use a table lookup oscillator to generate an amplitude envelope signal by defining an appropriate envelope shape and feeding the output of one oscillator into the amplitude input of another.

Music Score File
A file contains, e.g. MIDI

- An initialization section; global parameters, sampling rate, number of channel
- An instrument definition; interconnection of unit generators
- A wave table definition; basis waveform control function, etc
- A note list – define how instruments are played at certain times for certain durations
Principles of Music Synthesis

Additive Synthesis

Amplitude Modulation

(Tremolo effect)

Frequency Control

Time-varying frequency control is useful in the production of pitch glides as portamenti, pitch inflection such as starting note with a slightly flat pitch, converging to and perhaps overshooting the main pitch, and pitch drop off at the end of the note.

Amplitude Modulation
Principles of Music Synthesis

Additive Synthesis
Frequency Modulation

\[ f(t) = f_c + \Delta f \sin 2\pi f_m t \]

e.g., periodic vibrato

Principle of Frequency Modulation

\[ f_1 \text{ a sine waveform} \]
\[ f_2 \text{ an amplitude envelope} \]
Principles of Music Synthesis

Additive Synthesis
Controlling vibrato depth

• How to control both rate and depth?

Random Vibrato
Vibrate rate is centered at a value and added random jitter on it.

Frequency modulation with depth control
Principles of Music Synthesis

Subtractive Synthesis
It is based on the complementary idea of passing a broadband signal through a time-varying filter to produce the desired waveform. The basic processing model of subtractive synthesis consists of an excitation sound source feeding a resonating system.
Principles of Music Synthesis

Nonlinear Synthesis
Nonlinear synthesis does not obey the principle of superposition. The principle is that highly nonlinear devices can generate signals with lots of harmonics. Frequency modulation is a subclass of non-linear synthesis.

Frequency Modulation (FM) Synthesis
- licensed to Yamaha for its DX-7 FM synthesizer introduced in 1983

\[ f(n) = f_c + \Delta f \cos \frac{2 \pi f_m}{R} \]
- \( R \) is sampling rate

\[ y(n) = A \sin((2 \pi f_c / R) + I \sin(2 \pi f_m / R)) \]
- \( I \) is the modulation index,
- \( f_c \) is the carrier frequency
- \( f_m \) is the modulating frequency
Principles of Music Synthesis

Frequency Modulation (FM) Synthesis

**Bessel function**

\[
\sin(\theta + a \sin \beta) = J_0(a) \sin \theta + \sum_{k=1}^{\infty} J_k(a) [\sin(\theta + k\beta) + (-1)^k \sin(\theta - k\beta)]
\]

\( J_k(a) \) is a Bessel function of the first kind of order \( k \) evaluated at the point \( a \). If we let \( \theta = 2\pi f_c / R \), \( \beta = 2\pi f_m / R \), \( a = I = f_c / f_m \), the spectrum of a FM waveform consists of a component at carrier frequency \( f_c \) and an infinite number of sidebands at \( f_c \pm f_m \), \( f_c \pm 2f_m \), \( f_c \pm 3f_m \), and so on.

Bessel function of the first kind are solution to

\[
x^2 y'' + xy' + (x^2 - n^2)y = 0, \quad n \geq 0
\]

\( n \) is the order, recursive

\[
J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x)
\]

**Observation:**

If \( f_m = f_c \), \( f_c - f_m = 0 \), \( f_c - 2f_m = -f_c \) and will warp around at 0 Hz and interfere with the carrier frequency components.

- Increasing the modulating index generally increases the effective bandwidth of the spectrum
- The spacing of the spectrum components is easily controlled by choosing modulating frequencies that have particular relationships to the carrier frequency
Principles of Music Synthesis

Frequency Modulation (FM) Synthesis

Observation:
Harmonic ratio is defined as \( H = f_m / f_c \). If \( H \) has the form \( N_1 / N_2 \), where \( N_1 \) and \( N_2 \) are positive integers, the resulting waveform will be harmonic. If \( H \) is not of this form, the resulting spectrum is not harmonic, e.g., bells, drums, gongs are inharmonic sounds.

For a given \( f_c \) and \( H \), we may control the frequency deviation to generate any kind of amplitude envelope. Because the peak amplitude of the FM waveform is independent of the other FM parameters, we are free to impose on the waveform any kind of amplitude envelope that we like.

Application FM Synthesis

FM technique can produce many attractive and useful musical sounds by controlling its four basic parameters: \( f_c, f_m, \Delta f, A \) or \( f_c, H, I, A \).

where \( I = \Delta f / f_m \)

Extensions to Basic Frequency Modulation
- multiple carriers and/or multiple modulators
Principles of Music Synthesis

Nonlinear Wave Shaping

FM is only a small subset of nonlinear synthesis. The idea of nonlinear wave shaping is to use mathematical composition

\[ y = f(x) \, , \, z = g(y) = g(f(x)) \]

e.g. FM

\[ z = g(f(x)) = a\sin(\alpha + i\sin mx) \]

where \( g(y) = \sin(y) \, , \, f(x) = i\sin mx \)

In general, nonlinear wave shaping deals with identification of useful composing function \( g \) that can accept waveform \( f \) as arguments.

For example, Chebyshev polynomials.

\[(1 - x^2)y'' - xy' + n^2 y = 0, \quad n = 1, 2, 3, \ldots \]

\[ T_n(x) = \cos(n\cos^{-1} x) \]

Recurrence relationship:

\[ T_{n+1}(x) = 2xT_n(x) − T_{n−1}(x) \]

\[ T_0(x) = 1 \]

\[ T_1(x) = x \]

\[ T_2(x) = 2x^2 − 1 \]

\[ T_3(x) = 4x^3 − 3x \]
Principles of Music Synthesis

Nonlinear Wave Shaping
The reason why Chebyshev is so useful for nonlinear wave shaping is because of the nice property,

\[ T_k(\cos \theta) = \cos k \theta \]

i.e., if we feed a cosine waveform at frequency \( f \) into Chebyshev polynomial \( T_k \), the \( k \)th harmonic of the input waveform “pops out”.

\[ T_0(\cos x) = 1 = \cos 0 \]
\[ T_1(\cos x) = \cos x \]
\[ T_2(\cos x) = 2 \cos^2 x - 1 = \cos 2x \]
\[ T_3(\cos x) = 4 \cos^3 x - 3 \cos x = \cos 3x \]

Most implementations employ table look-up oscillator: use a wave table to store one period of waveform and index it according to the specified pitch frequency. Generally, the amplitudes of sinusoids can be modulated to produce tremolo effect. The frequencies can also be controlled (time-varying) for the production of pitch glide effects and vibrato. Advantages: simple and straightforward to implement. Disadvantages: require a large number of oscillators to produce harmonic rich notes.
Principles of Music Synthesis

Physical Modeling

Physical modeling is an approach to model the known or invented musical instruments. The approach is analogous to that of articulatory models of the human speech production mechanism. Generally, physical modeling is computational intensive and also limited to be used to model some string instruments. Recently, the digital waveguide technique developed by researchers in Stanford University can substantially improve sound quality by modeling the sound-generating processes that take place in instruments themselves. The digital waveguide approach is not limited to stringed instruments. Because the sound propagation in tubular instruments like flutes, clarinets and trumpets is very similar to what happens with a string, by adding a simple simulation of what takes place at the mouthpiece or reed, the technique can be used to simulate the sounds of these instruments as well.

e.g. Yamaha VL1 synthesizer introduced in 1994
Principles of Music Synthesis

Wavetable Synthesis

The majority of professional synthesizers available today use some form of wavetable synthesis. The trend for multimedia sound products is also towards wavetable synthesis. It is a relatively new method that uses small digitized recordings of the real-instruments as the basis for the synthesis process. The set of instrument recordings is referred to as the wavetable database.

The quality of the produced sound is affected both by the quality of the database and the quality of the signal-processing algorithms used in the wavetable synthesis process. One of the main drawbacks of this method is that the wavetable database requires dedicated on-board memory which is expensive.

Wavetable music synthesis is similar to simple digital sine wave generation but extended at least two ways. First, the waveform lookup table contains samples for not just a single period of a sine function but for a single period of a more general waveshape. Second, a mechanism exists for dynamically changing the waveshape as the musical note evolves, thus generating a quasi-periodic function in time. This mechanism requires the mixing of a set of well chosen basis wavetables each with their corresponding envelope function.
Principles of Music Synthesis

Wavetable Synthesis
The envelopes of the basis functions are overlapping triangular pulse functions as shown. Note that only two wavetables are being mixed at any one instance of time.

1) Multiple single cycle waveforms are used
2) One or multiple wave modulators control the change between those multiple waveforms or mixtures thereof
3) Modulation rate $<<$ sampling rate

Wavetable synthesis is a static waveform generator that uses a circular table of sequential waveform values, a phase accumulator for address generation, and some means of interpolation between neighboring wavetable samples.
Principles of Music Synthesis

Wavetable Synthesis

The other ingredient for wavetable synthesis is a mechanism to dynamically change the waveform as the musical note proceeds in time. The method involves mixing a finite set of static and phase-locked wavetables, each scaled by individual envelope functions.

The step is to periodically extend the quasi-periodic input. This could be done by simply hacking off all of the input and then periodically repeating the one cycle in both directions of time, ad infinitum. Of course, this would introduce a discontinuity at each splice unless the input was perfectly periodic. To avoid this, one can use a window that more gracefully truncates the input outside of the period in the $t0$ neighborhood. This window must have a complimentary fade-in and fade-out characteristic.
Principles of Music Synthesis

Wavetable Synthesis

A good wavetable synthesizer can produce music which is indistinguishable from a music recording made using real instruments. Wavetable synthesis also provides an opportunity for a larger variety of sounds, since any sound can be made available simply by providing "samples", unlike FM which can only create certain types of sounds based on the physical acoustics of FM sound modeling.

Overlap-added with windowing
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

• developed by MIDI Manufacturers Association in the 1982/1983
• a very efficient method of representing musical performance information
• widely accepted and utilized by musicians and composers
• also widely accepted for computer applications which produce sound, such as multimedia presentations or computer games
• lack of the standardization of synthesizer capabilities. The quality of sound produced is dependent on the synthesizer.

Synthesizer Basics

Polyphony
• The ability to play more than one note at a time. Polyphony is generally measured or specified as a number of notes or voices.
• If the keyboard had more voices (many modern sound modules have 16, 24, or 32 note polyphony), then by pressing, say, five keys on the keyboard you would hear all five of the notes.

Sounds
• The different sounds that a synthesizer can produce are sometimes called "patches", "programs" , "algorithms", or "timbres".
• A patch number is commonly assigned to each sound. For instance, a sound module might use patch number 1 for its acoustic piano sound, and patch number 36 for its fretless bass sound. The association of all patch numbers to all sounds is often referred to as a patch map.

Multitimbral Mode
• A synthesizer is said to be multitimbral if it is capable of producing two or more different instrument sounds simultaneously. If a synthesizer is polyphonic and can produce a piano sound and an acoustic bass sound at the same time, then with enough notes of polyphony and "parts" (multitimbral) a single synthesizer could produce the entire sound of a band or orchestra.
Musical Instrument Digital Interface (MIDI) Protocol

Definition of MIDI

- MIDI information is transmitted in "MIDI messages", which can be thought of as instructions which tell a music synthesizer how to play a piece of music. The synthesizer receiving the MIDI data must generate the actual sounds.
- Extremely economical in terms of memory requirements, about 10 Kbytes of data per minute of sound.
- Post-production editing possible with the MIDI sequencer. The ability to change the playback speed and the pitch or key of the sounds independently.
- Sound dependent on the listener's output device.

The MIDI data stream is a unidirectional asynchronous bit stream at 31.25 kbits/sec. with 10 bits transmitted per byte (a start bit, 8 data bits, and one stop bit). The MIDI interface on a MIDI instrument will generally include three different MIDI connectors, labeled IN, OUT, and THRU. The MIDI data stream is usually originated by a MIDI controller, such as a musical instrument keyboard, or by a MIDI sequencer. A MIDI controller is a device which is played as an instrument, and it translates the performance into a MIDI data stream in real time. A MIDI sequencer is a device which allows MIDI data sequences to be captured, stored, edited, combined, and replayed. The MIDI data output from a MIDI controller or sequencer is transmitted via the devices' MIDI OUT connector.
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

MIDI Terminologies:

*Synthesizer:*
  It is a sound generator (various pitch, loudness, tone color).
  A good (musician's) synthesizer often has a microprocessor, keyboard, control panels, memory, etc.

*Sequencer:*
  It can be a stand-alone unit or a software program for a personal computer. (It used to be a storage server for MIDI data. Nowadays it is more a software *music editor* on the computer.)
  It has one or more MIDI INs and MIDI OUTs.

*Track:*
  Track in sequencer is used to organize the recordings.
  Tracks can be turned on or off on recording or playing back.

*Channel:*
  MIDI channels are used to separate information in a MIDI system.
  There are 16 MIDI channels in one cable.
  Channel numbers are coded into each MIDI message.

*Timbre:*
  The quality of the sound, e.g., flute sound, cello sound, etc.
  Multitimbral -- capable of playing many different sounds at the same time (e.g., piano, brass, drums, etc.)

*Pitch:*
  Musical note that the instrument plays

*Voice:*
  Voice is the portion of the synthesizer that produces sound.
  Synthesizers can have many (16, 20, 24, 32, 64, etc.) voices.
  Each voice works independently and simultaneously to produce sounds of different timbre and pitch.

*Patch:*
  the control settings that define a particular timbre.
Musical Instrument Digital Interface (MIDI) Protocol

Hardware Aspects of MIDI

- MIDI connector is a 5-pin connector. There are three ports on the back of every MIDI unit.
- MIDI IN: the connector via which the device receives all MIDI data.
- MIDI OUT: the connector through which the device transmits all the MIDI data it generates itself.
- MIDI THROUGH: the connector by which the device echoes the data receives from MIDI IN.

Note: It is only the MIDI IN data that is echoed by MIDI through. All the data generated by device itself is sent through MIDI OUT.
Musical Instrument Digital Interface (MIDI) Protocol

MIDI Messages:
MIDI messages are used by MIDI devices to communicate with each other.

Structure of MIDI messages:
MIDI message includes a status byte and up to two data bytes.

Status byte
- The most significant bit of status byte is set to 1.
- The 4 low-order bits identify which channel it belongs to (four bits produce 16 possible channels).
- The 3 remaining bits identify the message.

The most significant bit of data byte is set to 0.

Classification of MIDI messages:
Musical Instrument Digital Interface (MIDI) Protocol

Channel messages: messages that are transmitted on individual channels rather than globally to all devices in the MIDI network.

Channel voice messages:
- Instruct the receiving instrument to assign particular sounds to its voice
- Turn notes on and off
- Alter the sound of the currently active note or notes

<table>
<thead>
<tr>
<th>Voice Message</th>
<th>Status Byte</th>
<th>Data Byte1</th>
<th>Data Byte2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note off</td>
<td>&amp;H8x</td>
<td>Key number</td>
<td>Note Off velocity</td>
</tr>
<tr>
<td>Note on</td>
<td>&amp;H9x</td>
<td>Key number</td>
<td>Note on velocity</td>
</tr>
<tr>
<td>Polyphonic Key Pressure</td>
<td>&amp;HAx</td>
<td>Key number</td>
<td>Amount of pressure</td>
</tr>
<tr>
<td>Control Change</td>
<td>&amp;HBx</td>
<td>Controller number</td>
<td>Controller value</td>
</tr>
<tr>
<td>Program Change</td>
<td>&amp;HCx</td>
<td>Program number</td>
<td>None</td>
</tr>
<tr>
<td>Channel Pressure</td>
<td>&amp;HDx</td>
<td>Pressure value</td>
<td>None</td>
</tr>
<tr>
<td>Pitch Bend</td>
<td>&amp;HEx</td>
<td>MSB</td>
<td>LSB</td>
</tr>
</tbody>
</table>

Notes: `x' in status byte hex value stands for a channel number.
Example: a Note On message is followed by two bytes, one to identify the note, and on to specify the velocity.
To play note number 80 with maximum velocity on channel 13, the MIDI device would send these three hexadecimal byte values: &H9C &H50 &H7F
Musical Instrument Digital Interface (MIDI) Protocol

*Note On / Note Off / Velocity*

When a key is pressed on a MIDI keyboard, the keyboard sends a Note On message on the MIDI OUT port. The status byte indicates the channel number, there are 16 logical MIDI channels. The Note On status byte is followed by one data byte to specify the key number (indicating which key was pressed) and one byte for the velocity (how hard the key was pressed).

The key number is used in the receiving synthesizer to select which note should be played, and the velocity is normally used to control the amplitude of the note. A Note Off message is sent when the key is released. The Note Off message also includes data bytes for the key number and for the velocity with which the key was released.

*Aftertouch*

This is the amount of pressure which is being applied to the keys while they are depressed. This pressure information, commonly called "aftertouch", may be used to control some aspects of the sound produced by the synthesizer (vibrato, for example). If the keyboard has a pressure sensor for each key, then the resulting "polyphonic aftertouch" information would be sent in the form of Polyphonic Key Pressure messages. These messages include separate data bytes for key number and pressure amount. It is currently more common for keyboard instruments to sense only a single pressure level for the entire keyboard. This "Channel aftertouch" information is sent using the Channel Pressure message, which needs only one data byte to specify the pressure value.
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

*Pitch Bend*
The Pitch Bend Change message is normally sent from a keyboard instrument in response to changes in position of the pitch bend wheel. The pitch bend information is used to modify the pitch of sounds being played on a given Channel. The Pitch Bend message includes two data bytes to specify the pitch bend value.

*Program Change*
The Program Change message is used to specify the type of instrument which should be used to play sounds on a given Channel. This message needs only one data byte which specifies the new program number.

*Control Change*
MIDI Control Change messages are used to control a wide variety of functions in a synthesizer. Control Change messages, like other MIDI Channel messages, should only affect the Channel number indicated in the status byte. The Control Change status byte is followed by one data byte indicating the "controller number", and a second byte which specifies the "control value". The controller number identifies which function of the synthesizer is to be controlled by the message.
Musical Instrument Digital Interface (MIDI) Protocol

Channel mode messages: Channel mode messages are a special case of the Control Change message (&HBx or 1011nnnn). The difference between a Control message and a Channel Mode message, which share the same status byte value, is in the first data byte. Data byte values 121 through 127 have been reserved in the Control Change message for the channel mode messages. Channel mode messages determine how an instrument will process MIDI voice messages.

<table>
<thead>
<tr>
<th>1st Data Byte</th>
<th>Description</th>
<th>Meaning of 2nd Data Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;H79</td>
<td>Reset all controllers</td>
<td>None; set to 0</td>
</tr>
<tr>
<td>&amp;H7A</td>
<td>Local control</td>
<td>0 = off; 127 = on</td>
</tr>
<tr>
<td>&amp;H7B</td>
<td>All notes off</td>
<td>None; set to 0</td>
</tr>
<tr>
<td>&amp;H7C</td>
<td>Omni mode off</td>
<td>None; set to 0</td>
</tr>
<tr>
<td>&amp;H7D</td>
<td>Omni mode on</td>
<td>None; set to 0</td>
</tr>
<tr>
<td>&amp;H7E</td>
<td>Mono mode on (Poly mode off)</td>
<td>**</td>
</tr>
<tr>
<td>&amp;H7F</td>
<td>Poly mode on (Mono mode off)</td>
<td>None; set to 0</td>
</tr>
</tbody>
</table>

** if value = 0 then the number of channels used is determined by the receiver; all other values set a specific number of channels, beginning with the current basic channel.
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

System Messages:
System messages carry information that is not channel specific, such as timing signal for synchronization, positioning information in pre-recorded MIDI sequences, and detailed setup information for the destination device.

System real-time messages:
messages related to synchronization

<table>
<thead>
<tr>
<th>System Real-Time Message</th>
<th>Status Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing Clock</td>
<td>&amp;HF8</td>
</tr>
<tr>
<td>Start Sequence</td>
<td>&amp;HFA</td>
</tr>
<tr>
<td>Continue Sequence</td>
<td>&amp;HFB</td>
</tr>
<tr>
<td>Stop Sequence</td>
<td>&amp;HFC</td>
</tr>
<tr>
<td>Active Sensing</td>
<td>&amp;HFF</td>
</tr>
</tbody>
</table>

System common messages:
contain the following unrelated messages

<table>
<thead>
<tr>
<th>System Common Message</th>
<th>Status Byte</th>
<th>Number of Data Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Common Message</td>
<td>Status Byte</td>
<td>Number of Data Bytes</td>
</tr>
<tr>
<td>MIDI Timing Code</td>
<td>&amp;HF1</td>
<td>1</td>
</tr>
<tr>
<td>Song Position Pointer</td>
<td>&amp;HF2</td>
<td>2</td>
</tr>
<tr>
<td>Song Select</td>
<td>&amp;HF3</td>
<td>1</td>
</tr>
<tr>
<td>Tune Request</td>
<td>&amp;HF6</td>
<td>None</td>
</tr>
</tbody>
</table>

System exclusive message:
• Messages related to things that cannot be standardized, (b) addition to the original MIDI specification.
• It is just a stream of bytes, all with their high bits set to 0, bracketed by a pair of system exclusive start and end messages (&HF0 and &HF7).
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

MIDI Sequencers and Standard MIDI Files

- If MIDI messages are generated in real time from musical instrument keyboard, there is no need for timing information to be sent along with the MIDI messages.

- However, if the MIDI data is to be stored as a data file, and/or edited using a sequencer, then some form of "time-stamping" for the MIDI messages is required. The Standard MIDI Files specification provides a standardized method for handling time-stamped MIDI data.

- The specification for Standard MIDI Files defines three formats for MIDI files.
  - Format 0 store all of the MIDI sequence data in a single track.
  - Format 1 files store MIDI data as a collection of tracks.
  - Format 2 files can store several independent patterns.

- Most sophisticated MIDI sequencers can read either Format 0 or Format 1 Standard MIDI Files. Format 0 files may be smaller, and thus conserve storage space. However, Format 1 files may be viewed and edited more directly, and are therefore generally preferred.

General MIDI

General MIDI Level 1 specification, also known as "GM1" was proposed in September of 1991 by the MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC). General MIDI 1 was designed to provide a minimum level of performance compatibility among MIDI instruments.
Music Synthesis

Musical Instrument Digital Interface (MIDI) Protocol

MIDI Sequencers and Standard MIDI Files

General MIDI

The General MIDI (GM) Specification defines a set of general capabilities for General MIDI Instruments. It includes the definition of a General MIDI Sound Set (a patch map), a General MIDI Percussion map (mapping of percussion sounds to note numbers), and a set of General MIDI Performance capabilities (number of voices, types of MIDI messages recognized, etc.). A MIDI sequence which has been generated for use on a General MIDI Instrument should play correctly on any General MIDI synthesizer or sound module.

- MIDI + Instrument Patch Map + Percussion Key Map --> a piece of MIDI music sounds the same anywhere it is played
  - Instrument patch map is a standard program list consisting of 128 patch types.
  - Percussion map specifies 47 percussion sounds.
  - Key-based percussion is always transmitted on MIDI channel 10.

- Requirements for General MIDI Compatibility:
  - Support all 16 channels.
  - Each channel can play a different instrument/program (multitimbral).
  - Each channel can play many voices (polyphony).
  - Minimum of 24 fully dynamically allocated voices.

The General MIDI system specifies which instrument or sound corresponds with each program/patch number, but General MIDI does not specify how these sounds are produced. Thus, program number 1 should select the Acoustic Grand Piano sound on any General MIDI instrument. However, the Acoustic Grand Piano sound on two General MIDI synthesizers which use different synthesis techniques may sound quite different.

General MIDI 2 is a group of extensions made to General MIDI 1 in 1999, which increases both the number of available sounds and the amount of control available for sound editing and musical performance. All GM2 devices are also fully compatible with General MIDI 1.