Chapter 6
Basics of Digital Audio

6.1 Digitization of Sound

What is Sound?

- Sound is a wave phenomenon like light, but is macroscopic and involves molecules of air being compressed and expanded under the action of some physical device.
  a) For example, a speaker in an audio system vibrates back and forth and produces a *longitudinal* pressure wave that we perceive as sound.
  b) Since sound is a pressure wave, it takes on continuous values, as opposed to digitized ones.

c) Even though such pressure waves are longitudinal, they still have ordinary wave properties and behaviors, such as reflection (bouncing), refraction (change of angle when entering a medium with a different density) and diffraction (bending around an obstacle).

d) If we wish to use a digital version of sound waves we must form digitized representations of audio information.

→ Link to physical description of sound waves.

- Signals can be decomposed into a sum of sinusoids. Fig. 6.1 shows how weighted sinusoids can build up quite a complex signal.

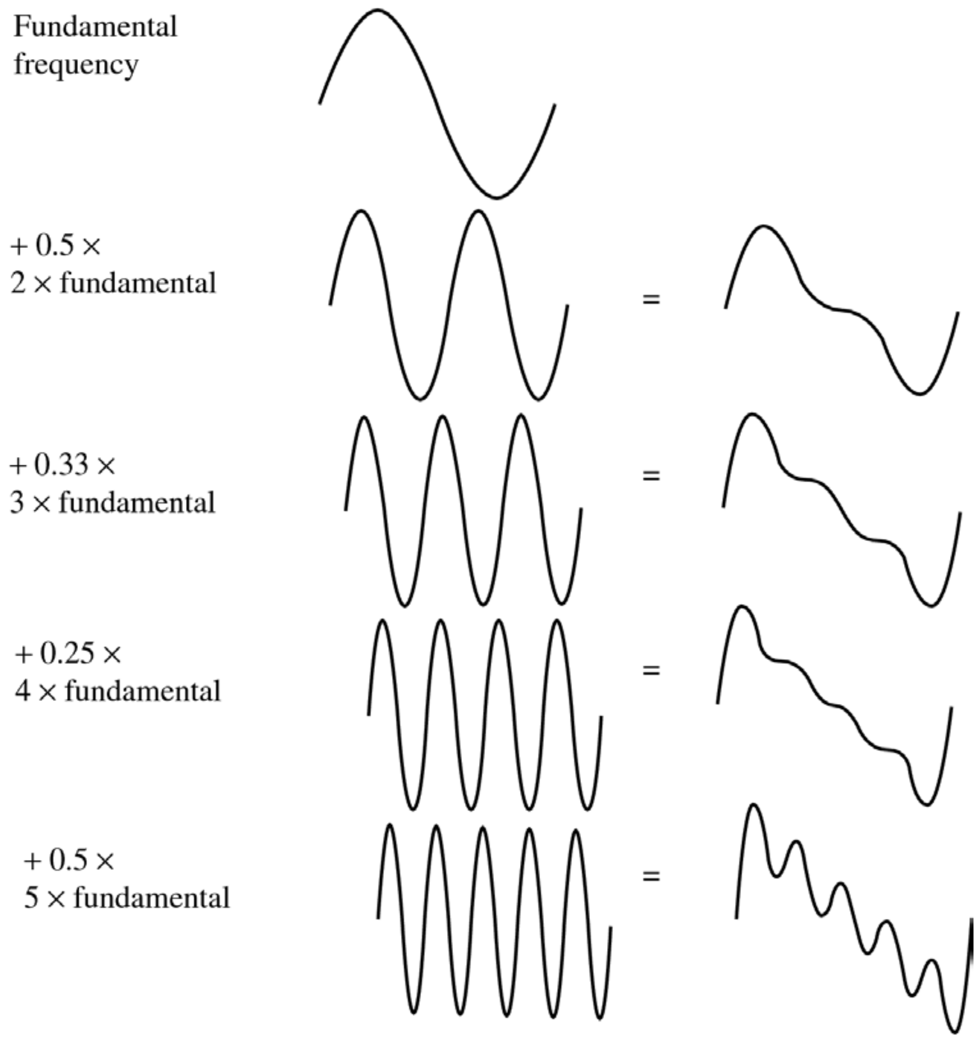


Fig. 6.1: Building up a complex signal by superposing sinusoids

- Whereas **frequency** is an absolute measure, **pitch** is generally relative — a perceptual subjective quality of sound.

  (a) Pitch and frequency are linked by setting the note A above middle C to exactly 440 Hz.

  (b) An **octave** above that note takes us to another A note. An octave corresponds to *doubling the frequency*. Thus with the middle "A" on a piano ("A4" or "A440") set to 440 Hz, the next "A" up is at 880 Hz, or one octave above.

- **Harmonics**: any series of musical tones whose frequencies are integral multiples of the frequency of a fundamental tone.

- If we allow non-integer multiples of the base frequency, we allow **overtones**, and have a more complex resulting sound.

- Together, the fundamental frequency and overtones are referred to as **partials**. The harmonics can also be referred to as **harmonic partials**.

6.1.2 Digitization

- **Digitization** means conversion to a stream of numbers, and preferably these numbers should be integers for efficiency.

- Fig. 6.2 shows the 1-dimensional nature of sound: **amplitude** values depend on a 1D variable, time. (And note that images depend instead on a 2D set of variables, x and y).

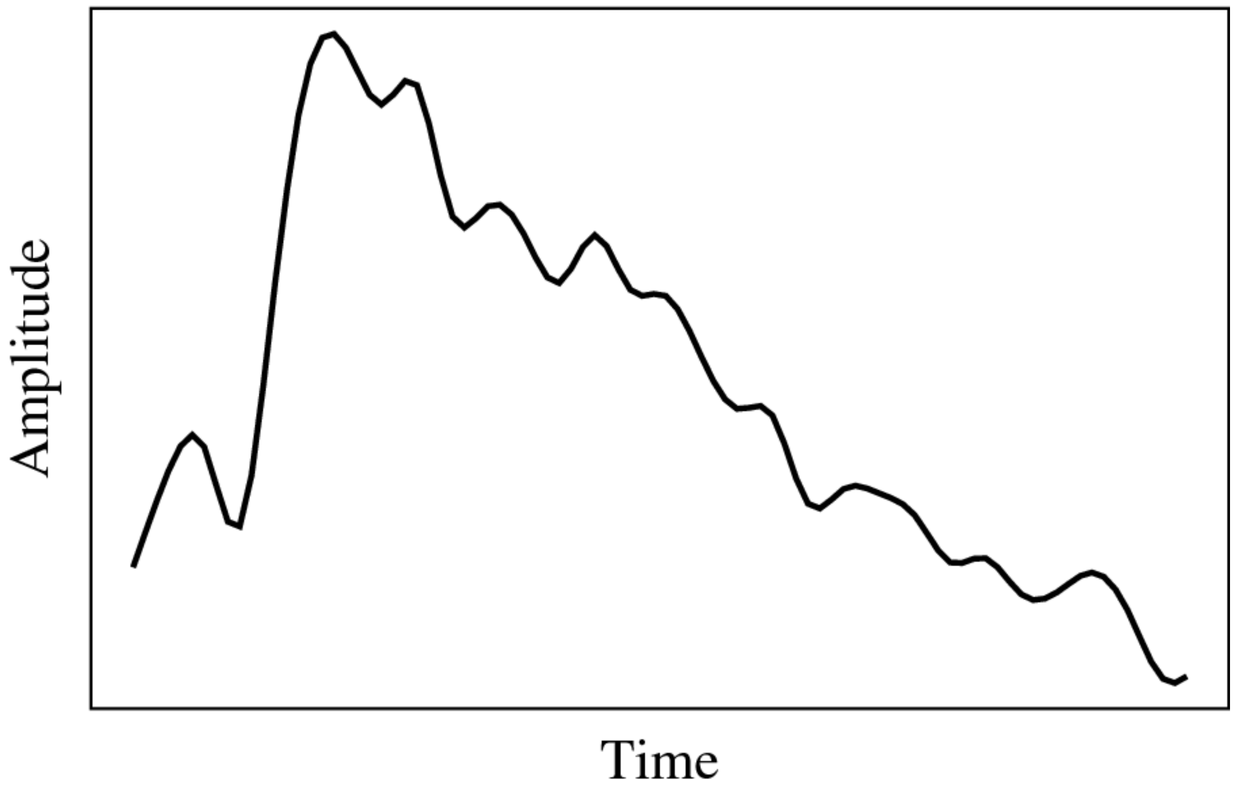


Fig. 6.2: An analog signal: continuous measurement of pressure wave.

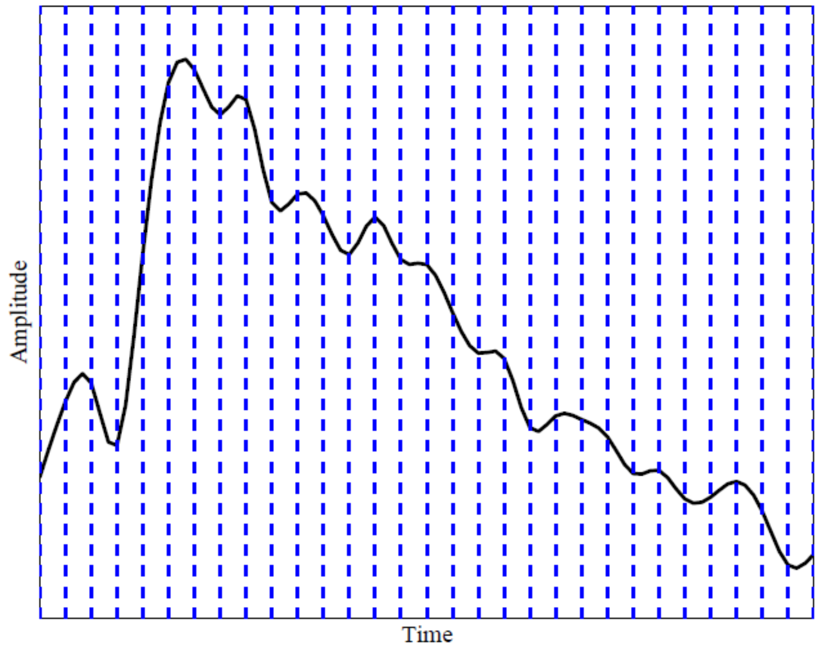
- To digitize, the signal must be **sampled** in each dimension: in time, and in amplitude.

  (a) Sampling means measuring the quantity we are interested in, usually at evenly-spaced intervals.

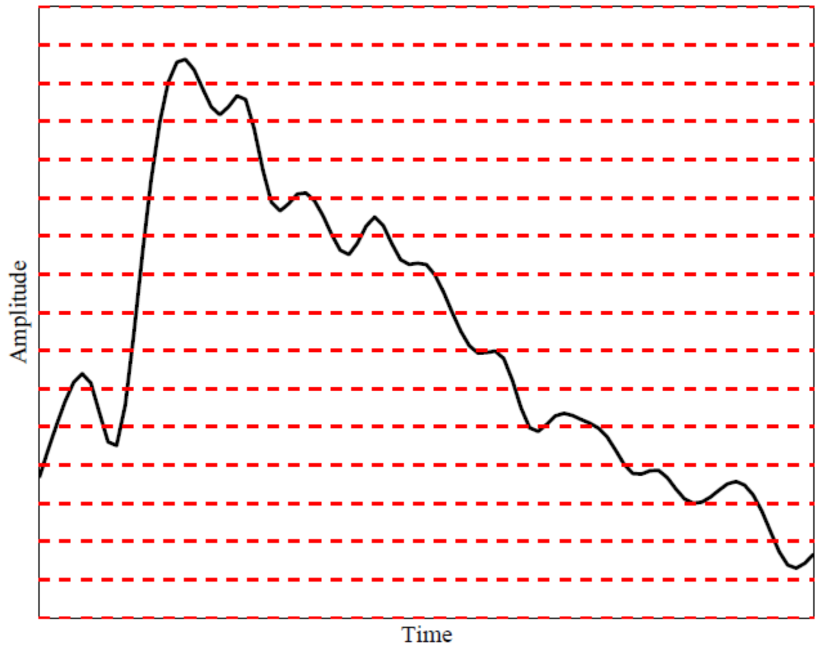
  (b) The first kind of sampling, using measurements only at evenly spaced time intervals, is simply called, sampling. The rate at which it is performed is called the *sampling frequency* (see Fig. 6.3(a))

  (c) For audio, typical sampling rates are from 8 kHz (8,000 samples per second) to 48 kHz. This range is determined by the Nyquist theorem, discussed later.

  (d) Sampling in the amplitude or voltage dimension is called **quantization**. Fig. 6.3(b) shows this kind of sampling.



(a)



(b)

Fig. 6.3: Sampling and Quantization.

(a):Sampling the analog signal in the time dimension.

(b):Quantization is sampling the analog signal in the amplitude dimension.

- Thus to decide how to digitize audio data we need to answer the following questions:

1. What is the sampling rate?
2. How finely is the data to be quantized, and is quantization uniform?
3. How is audio data formatted? (file format)

6.1.3 Nyquist Theorem

- The Nyquist theorem states how frequently we must sample in time to be able to recover the original sound.

  (a) Fig. 6.4(a) shows a single sinusoid: it is a single, pure, frequency (only electronic instruments can create such sounds).

  (b) If sampling rate just equals the actual frequency, Fig. 6.4(b) shows that a false signal is detected: it is simply a constant, with zero frequency.

(a) Now if sample at 1.5 times the actual frequency, Fig. 6.4(c) shows that we obtain an incorrect (**alias**) frequency that is lower than the correct one — it is half the correct one (the wavelength, from peak to peak, is double that of the actual signal).

(b) Thus for correct sampling we must use a sampling rate equal to at least twice the maximum frequency content in the signal. This rate is called the **Nyquist rate**.

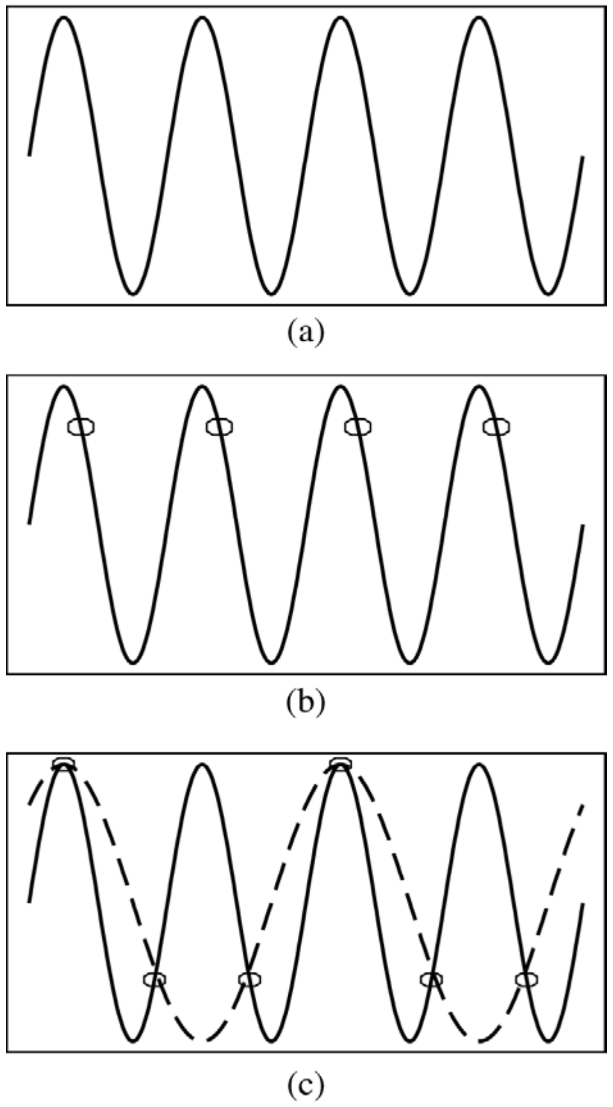


Fig. 6.4: Aliasing.

(a): A single frequency.

(b): Sampling at exactly the frequency produces a constant.

(c): Sampling at 1.5 times per cycle produces an *alias* perceived frequency.

- **Nyquist Theorem**: If a signal is **band-limited**, i.e., there is a lower limit f_1 and an upper limit f_2 of frequency components in the signal, then the sampling rate should be at least $2(f_2 - f_1)$.

- **Nyquist frequency**: half of the Nyquist rate.
  - Since it would be impossible to recover frequencies higher than Nyquist frequency in any event, most systems have an **antialiasing filter** that restricts the frequency content in the input to the sampler to a range at or below Nyquist frequency.

- The relationship among the Sampling Frequency, True Frequency, and the Alias Frequency is as follows:

$$f_{alias} = f_{sampling} - f_{true}, \text{ for } f_{true} < f_{sampling} < 2 \times f_{true} \qquad (6.1)$$

- In general, the apparent frequency of a sinusoid is the lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid. Fig. 6.5 shows the relationship of the apparent frequency to the input frequency.

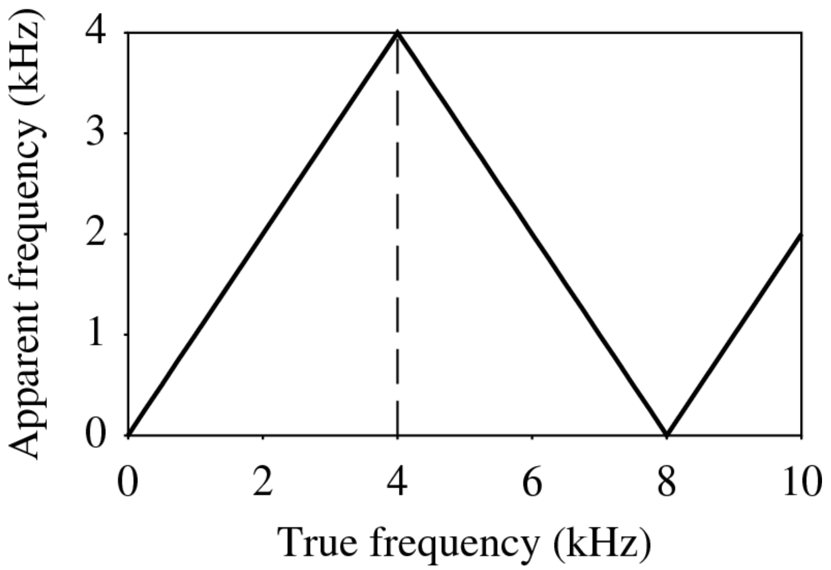


Fig. 6.5: Folding of sinusoid frequency which is sampled at 8,000 Hz. The folding frequency, shown dashed, is 4,000 Hz.

6.1.4 Signal to Noise Ratio (SNR)

- The ratio of the power of the correct signal and the noise is called the *signal to noise ratio* (**SNR**) — a measure of the quality of the signal.

- The SNR is usually measured in decibels (**dB**), where 1 dB is a tenth of a **bel**. The SNR value, in units of dB, is defined in terms of base-10 logarithms of squared voltages, as follows:

$$SNR = 10\log_{10}\frac{V_{signal}^2}{V_{noise}^2} = 20\log_{10}\frac{V_{signal}}{V_{noise}} \qquad (6.2)$$

a) The power in a signal is proportional to the square of the voltage. For example, if the signal voltage V_{signal} is 10 times the noise, then the SNR is

$$20 * \log_{10}(10) = 20\text{dB}.$$

To know: Power → 10; Signal Voltage → 20.

- The usual levels of sound we hear around us are described in terms of decibels, as a ratio to the quietest sound we are capable of hearing. Table 6.1 shows approximate levels for these sounds.

Table 6.1: Magnitude levels of common sounds, in decibels

Threshold of hearing	0
Rustle of leaves	10
Very quiet room	20
Average room	40
Conversation	60
Busy street	70
Loud radio	80
Train through station	90
Riveter	100
Threshold of discomfort	120
Threshold of pain	140
Damage to ear drum	160

6.1.5 Signal to Quantization Noise Ratio (SQNR)

- Aside from any noise that may have been present in the original analog signal, there is also an additional error that results from quantization.

(a) If voltages are actually in 0 to 1 but we have only 8 bits in which to store values, then effectively we force all continuous values of voltage into only 256 different values.

(b) This introduces a roundoff error. It is not really “noise”. Nevertheless it is called **quantization noise** (or quantization error).

- The quality of the quantization is characterized by the Signal to Quantization Noise Ratio (**SQNR**).

  (a) **Quantization noise**: the difference between the actual value of the analog signal, for the particular sampling time, and the nearest quantization interval value.

  (b) At most, this error can be as much as half of the interval.

c) For a quantization accuracy of N bits per sample, the SQNR can be simply expressed:

$$SQNR = 20\log_{10}\frac{V_{signal}}{V_{quan\_noise}} = 20\log_{10}\frac{2^{N-1}}{\frac{1}{2}}$$

$$= 20 \times N \times \log 2 = 6.02\,N\,(\text{dB}) \qquad (6.3)$$

- Notes:

(a) We map the maximum signal to $2^{N-1} - 1$ (and the most negative signal to -2^{N-1}.

- 6.02N **is the worst case**. If the input signal is sinusoidal, the quantization error is statistically independent, and its magnitude is uniformly distributed between 0 and half of the interval, then it can be shown that the expression for the SQNR becomes:

$$SQNR = 6.02N + 1.76(dB) \qquad (6.4)$$

6.1.6 Linear and Non-linear Quantization

- **Linear format**: samples are typically stored as uniformly quantized values.
- **Non-uniform quantization**: set up more finely-spaced levels where humans hear with the most acuity.
  - Weber's Law stated formally says that equally perceived differences have values proportional to absolute levels:

$$\Delta\text{Response} \propto \Delta\text{Stimulus}/\text{Stimulus} \tag{6.5}$$

  - Inserting a constant of proportionality k, we have a differential equation that states:

$$dr = k\,(1/s)\,ds \tag{6.6}$$

with response r and stimulus s.

– Integrating, we arrive at a solution

$$r = k \ln(s) + C \tag{6.7}$$

with constant of integration C.

Stated differently, the solution is

$$r = k \ln\left(s/s_0\right) \tag{6.8}$$

s_0 = the lowest level of stimulus that causes a response (r = 0 when $s = s_0$).

- Nonlinear quantization works by first transforming an analog signal from the raw s space into the theoretical r space, and then uniformly quantizing the resulting values.
- Such a law for audio is called ***μ*-law** encoding, (or **u-law**). A very similar rule, called

  ***A*-law**, is used in telephony in Europe.
- The equations for these very similar encodings are as follows:

μ-law:

$$r = \frac{\operatorname{sgn}(s)}{\ln(1+\mu)} \ln\left\{1+\mu\left|\frac{s}{s_p}\right|\right\}, \qquad \left|\frac{s}{s_p}\right| \le 1 \tag{6.9}$$

A-law:

$$r = \begin{cases} \dfrac{A}{1+\ln A}\left(\dfrac{s}{s_p}\right), & \left|\dfrac{s}{s_p}\right| \le \dfrac{1}{A} \\ \dfrac{\operatorname{sgn}(s)}{1+\ln A}\left[1+\ln A\left|\dfrac{s}{s_p}\right|\right], & \dfrac{1}{A} \le \left|\dfrac{s}{s_p}\right| \le 1 \end{cases} \tag{6.10}$$

$$\text{where } \operatorname{sgn}(s) = \begin{cases} 1 & \text{if } s > 0, \\ -1 & \text{otherwise} \end{cases}$$

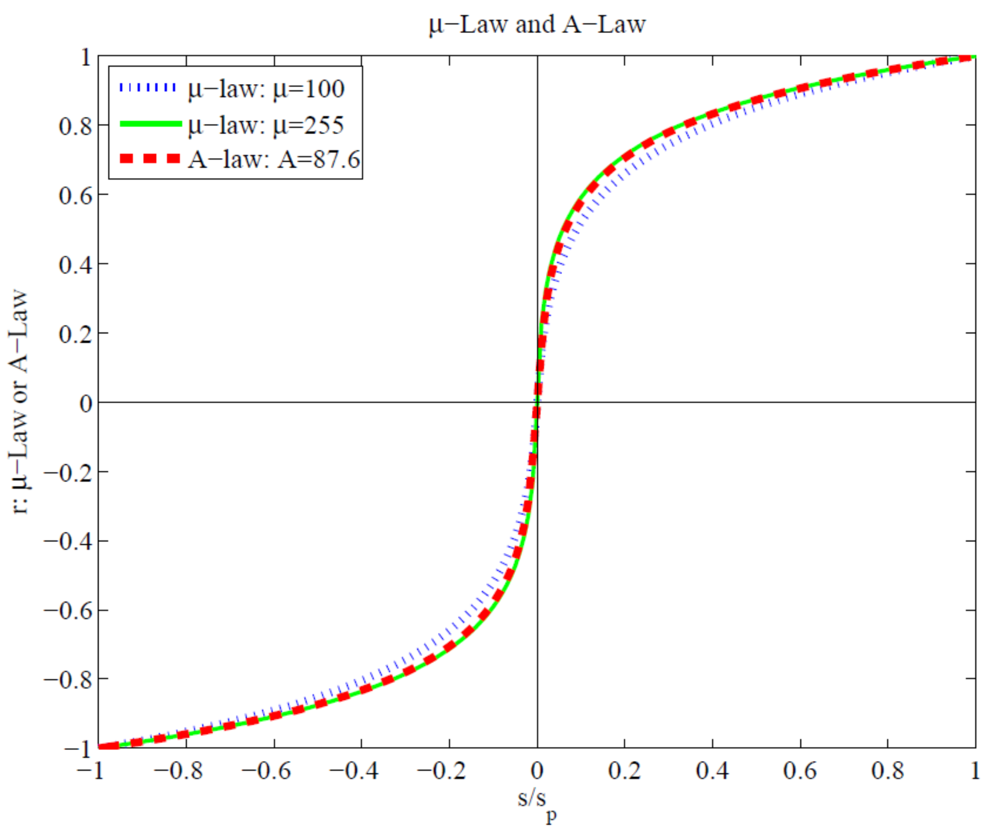


Fig. 6.6: Nonlinear transform for audio signals.

The parameter μ is set to μ = 100 or μ = 255; the parameter A for the A-law encoder is usually set to A = 87.6.

- The μ-law in audio is used to develop a nonuniform quantization rule for sound: uniform quantization of r gives finer resolution in s at the quiet end.

Bit allocation:

- In μ-law, we would like to put the available bits where the most perceptual acuity (sensitivity to small changes) is.
    1. Savings in bits can be gained by transmitting a smaller bit-depth for the signal.
    2. μ-law often starts with a bit-depth of 16 bits, but transmits using 8 bits.
    3. And then expands back to 16 bits at the receiver.

- Let μ=255.
- Now, we want s in $[-1, 1]$. The input is in -2^{15} to $(+2^{15} - 1)$, we divide by 2^{15} to normalize.
- Then the μ-law is applied to turn s into r.
- Now go down to 8-bit samples, using $\hat{r} = \text{sign}(s) * \text{floor}(128 * r)$.
- Now the 8-bit signal $\hat{r}$ is transmitted.
- At the receiver side, we normalize $\hat{r}$ by dividing by 2^7, and then apply the inverse μ-law function:

- at the receiver side, we normalize $\hat{r}$ by dividing by 2^7, and then apply the inverse *μ*-law function:

$$\hat{s} = \text{sign}(s)\left(\frac{(\mu+1)^{|\hat{r}|}-1}{\mu}\right)$$

- Finally, we expand back up to 16 bits:

$$\tilde{s} = ceil\left(2^{15} \times \hat{s}\right)$$

Companding puts the most accuracy at the quiet end near zero.

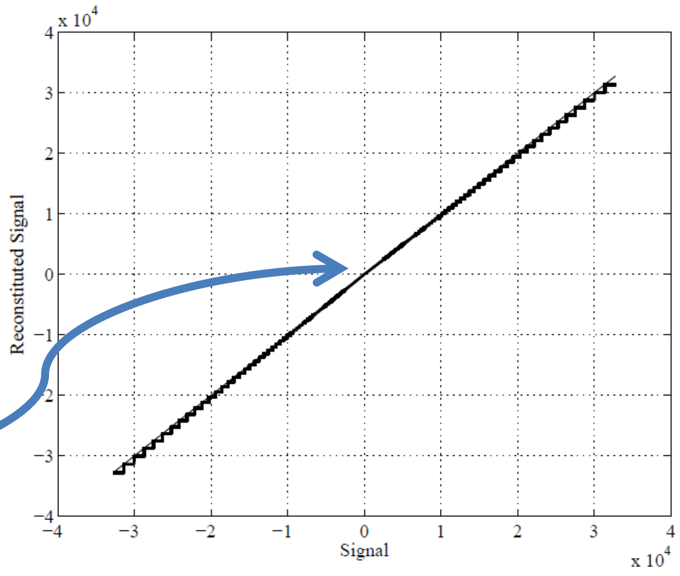


Fig. 6.7: Nonlinear quantization by companding.

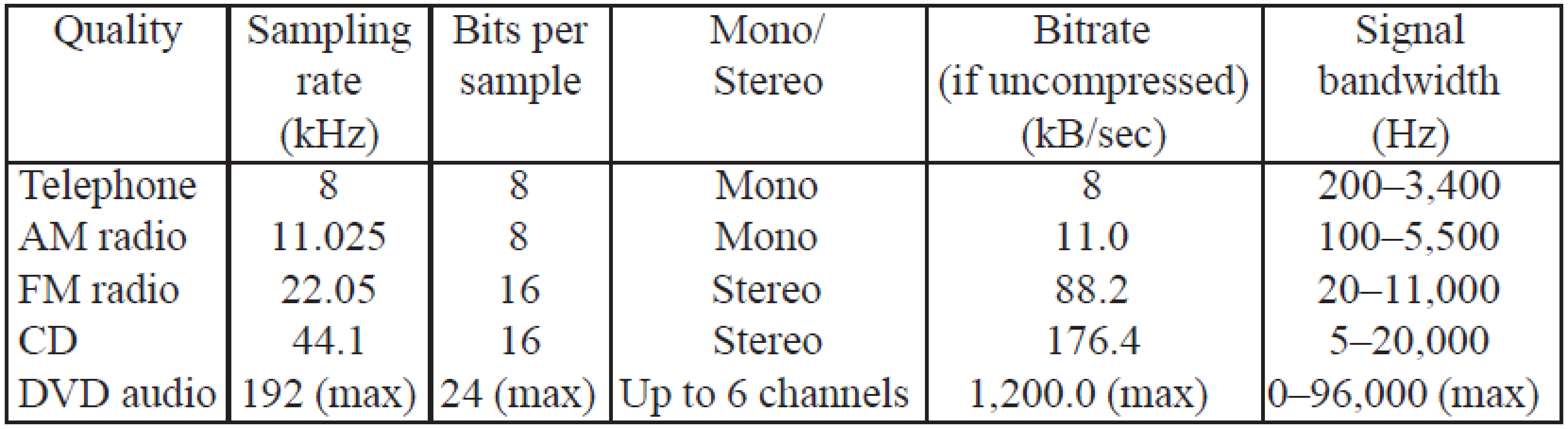
6.1.7 Audio Filtering

- Prior to sampling and AD conversion, the audio signal is also usually *filtered* to remove unwanted frequencies. The frequencies kept depend on the application:
  a) For speech, typically from 50Hz to 10kHz is retained, and other frequencies are blocked by the use of a **band-pass filter** that screens out lower and higher frequencies.
  b) An audio music signal will typically contain from about 20Hz up to 20kHz.
  c) At the DA converter end, high frequencies may reappear in the output — because of sampling and then quantization, smooth input signal is replaced by a series of step functions containing all possible frequencies.
  d) So at the decoder side, a **lowpass** filter is used after the DA circuit.

6.1.7 Audio Quality vs. Data Rate

- To transmit a digital audio signal, the uncompressed data rate increases as more bits are used for quantization. Stereo: double the bandwidth.

Table 6.2: Bitrate and bandwidth in sample audio applications



Quality	Sampling rate (kHz)	Bits per sample	Mono/ Stereo	Bitrate (if uncompressed) (kB/sec)	Signal bandwidth (Hz)
Telephone	8	8	Mono	8	200–3,400
AM radio	11.025	8	Mono	11.0	100–5,500
FM radio	22.05	16	Stereo	88.2	20–11,000
CD	44.1	16	Stereo	176.4	5–20,000
DVD audio	192 (max)	24 (max)	Up to 6 channels	1,200.0 (max)	0–96,000 (max)

6.1.9 Synthetic Sounds

1. **FM** (Frequency Modulation): one approach to generating synthetic sound:

$$x(t) = A(t)\cos[\omega_c \pi t + I(t)\cos(\omega_m \pi t + \phi_m) + \phi_c] \qquad (6.12)$$

→ Link to details

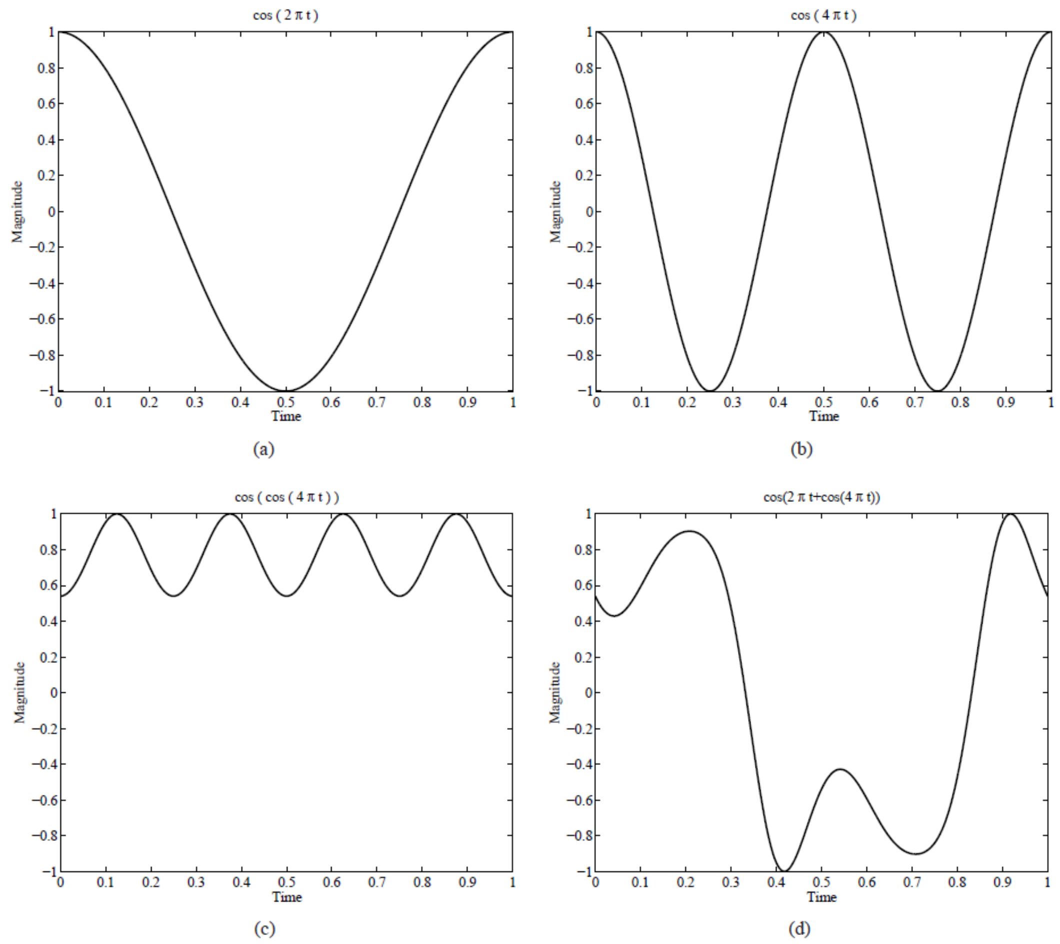


Fig. 6.8: Frequency Modulation. (a): A single frequency. (b): Twice the frequency. (c): Usually, FM is carried out using a sinusoid argument to a sinusoid. (d): A more complex form arises from a carrier frequency, $2\pi t$ and a modulating frequency $4\pi t$ cosine inside the sinusoid.

2. **Wave Table synthesis**: A more accurate way of generating sounds from digital signals. Also known, simply, as **sampling**.

- In this technique, the actual digital samples of sounds from real instruments are stored. Since wave tables are stored in memory on the sound card, they can be manipulated by software so that sounds can be combined, edited, and enhanced.

→ Link to details.

6.2 MIDI: Musical Instrument Digital Interface

- Use the sound card's defaults for sounds: ⇒ use a simple scripting language and hardware setup called **MIDI**.

- **MIDI Overview**

  a) MIDI is a scripting language — it codes "events" that stand for the production of sounds. E.g., a MIDI event might include values for the pitch of a single note, its duration, and its volume.

  b) MIDI is a standard adopted by the electronic music industry for controlling devices, such as synthesizers and sound cards, that produce music.

a) The MIDI standard is supported by most synthesizers, so sounds created on one synthesizer can be played and manipulated on another synthesizer and sound reasonably close.

b) Computers must have a special MIDI interface, but this is incorporated into most sound cards. The sound card must also have both D/A and A/D converters.

MIDI Concepts

- MIDI **channels** are used to separate messages.

  (a) There are 16 channels numbered from 0 to 15. The channel forms the last 4 bits (the least significant bits) of the message.

  (b) Usually a channel is associated with a particular instrument: e.g., channel 1 is the piano, channel 10 is the drums, etc.

  (c) Nevertheless, one can switch instruments midstream, if desired, and associate another instrument with any channel.

- **System messages**

  (a) Several other types of messages, e.g. a general message for all instruments indicating a change in tuning or timing.

  (b) If the first 4 bits are all 1s, then the message is interpreted as a **system common** message.

- The way a synthetic musical instrument responds to a MIDI message is usually by simply ignoring any **play sound** message that is not for its channel.
  - If several messages are for its channel, then the instrument responds, provided it is **multi-voice**, i.e., can play more than a single note at once.

- It is easy to confuse the term **voice** with the term **timbre** — the latter is MIDI terminology for just what instrument that is trying to be emulated, e.g. a piano as opposed to a violin: it is the quality of the sound.
  - (a) An instrument (or sound card) that is **multi-timbral** is one that is capable of playing many different sounds at the same time, e.g., piano, brass, drums, etc.
  - (b) On the other hand, the term **voice**, while sometimes used by musicians to mean the same thing as timbre, is used in MIDI to mean every different timbre and pitch that the tone module can produce at the same time.
- Different timbres are produced digitally by using a **patch** — the set of control settings that define a particular timbre. Patches are often organized into databases, called **banks**.

- **General MIDI**: A standard mapping specifying what instruments (what patches) will be associated with what channels.

  a) In General MIDI, channel 10 is reserved for percussion instruments, and there are 128 patches associated with standard instruments.

  b) For most instruments, a typical message might be a Note On message (meaning, e.g., a keypress and release), consisting of what channel, what pitch, and what “velocity” (i.e., volume).

  c) For percussion instruments, however, the pitch data means which kind of drum.

  d) A Note On message consists of “status” byte — which channel, what pitch — followed by two data bytes. It is followed by a Note Off message, which also has a pitch (which note to turn off) and a velocity (often set to zero).

→ Link to General MIDI Instrument Patch Map

→ Link to General MIDI Percussion Key Map

- The data in a MIDI status byte is between 128 and 255; each of the *data bytes* is between 0 and 127. Actual MIDI bytes are 10-bit, including a 0 start and 0 stop bit.

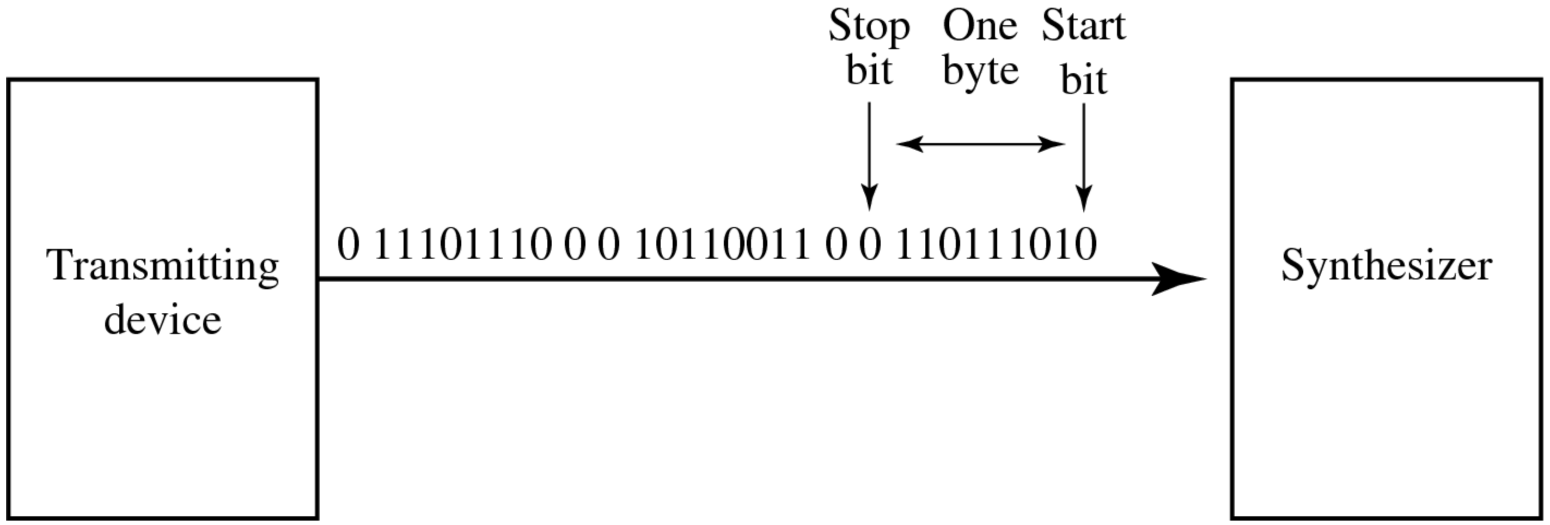


Fig. 6.9: Stream of 10-bit bytes; for typical MIDI messages, these consist of {Status byte, Data Byte, Data Byte} = {Note On, Note Number, Note Velocity}

- A MIDI device often is capable of **programmability**, and also can change the **envelope** describing how the amplitude of a sound changes over time.
- Fig. 6.10 shows a model of the response of a digital instrument to a Note On message:

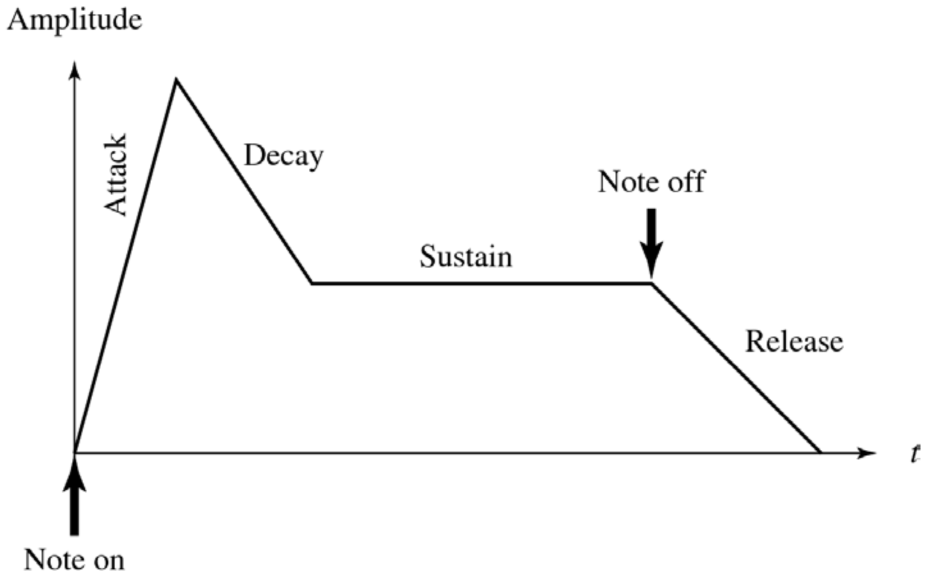


Fig. 6.10: Stages of amplitude versus time for a music note

6.2.2 Hardware Aspects of MIDI

- The MIDI hardware setup consists of a 31.25 kbps serial connection. Usually, MIDI-capable units are either Input devices or Output devices, not both.

- A traditional synthesizer is shown in Fig. 6.11:

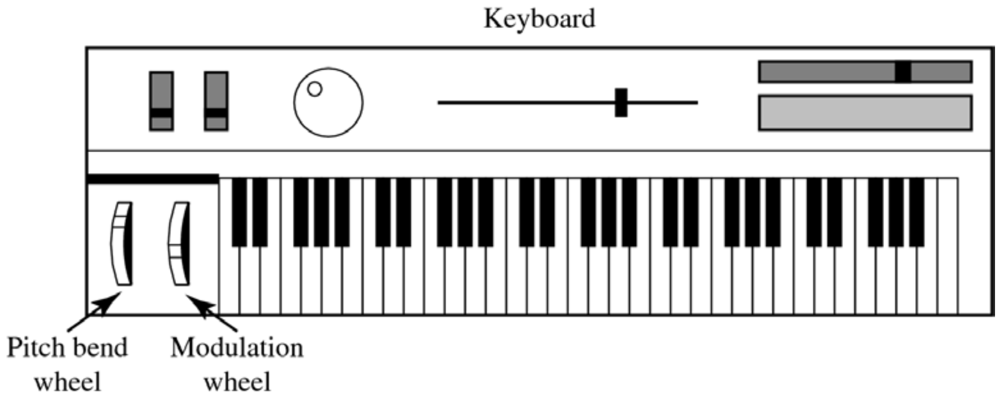


Fig. 6.11: A MIDI synthesizer

- The physical MIDI ports consist of 5-pin connectors for IN and OUT, as well as a third connector called THRU.

  a) MIDI communication is half-duplex.

  b) MIDI IN is the connector via which the device receives all MIDI data.

  c) MIDI OUT is the connector through which the device transmits all the MIDI data it generates itself.

  d) MIDI THRU is the connector by which the device echoes the data it receives from MIDI IN. Note that it is only the MIDI IN data that is echoed by MIDI THRU — all the data generated by the device itself is sent via MIDI OUT.

- A typical MIDI sequencer setup is shown in Fig. 6.12:

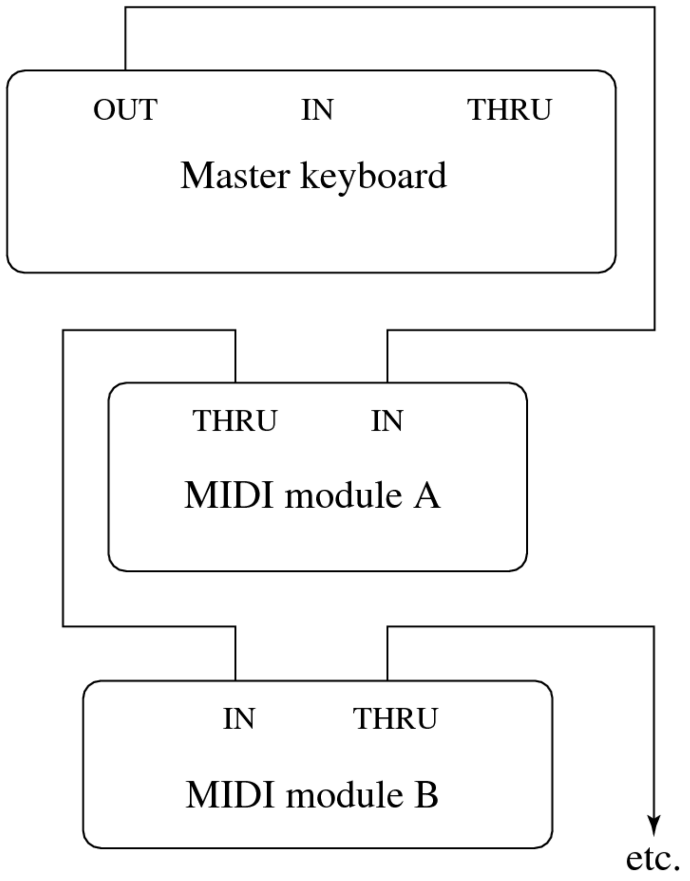


Fig. 6.12: A typical MIDI setup

6.2.3 Structure of MIDI Messages

- MIDI messages can be classified into two types: channel messages and system messages, as in Fig. 6.13:

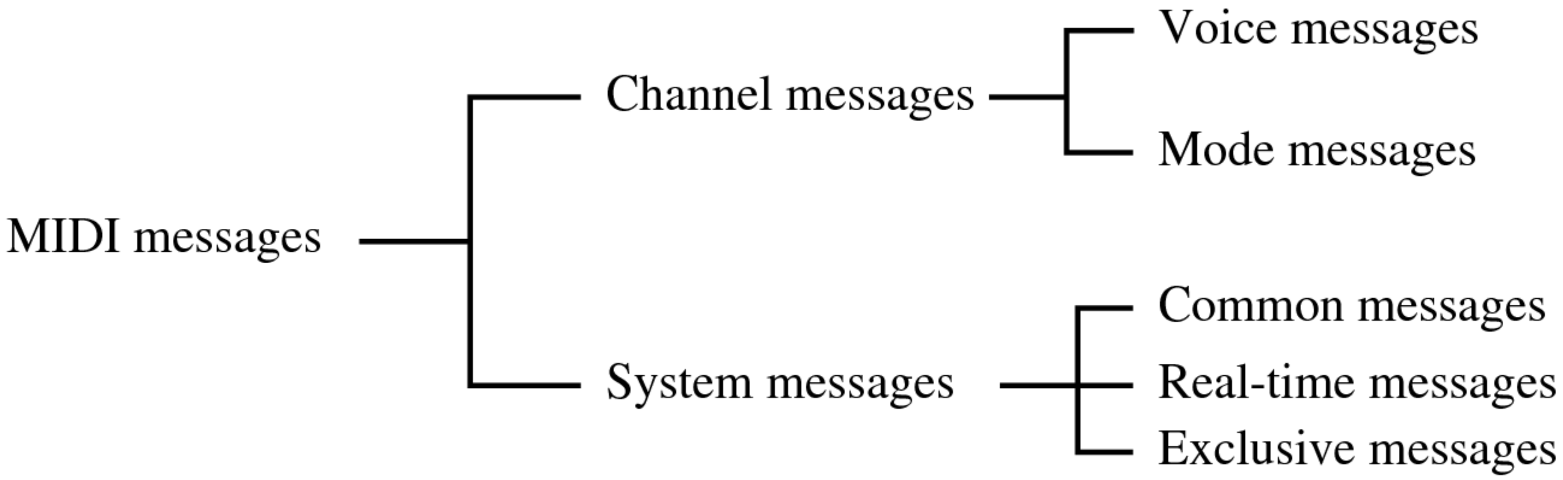


Fig. 6.13: MIDI message taxonomy

- **A. Channel messages**: can have up to 3 bytes:

  a) The first byte is the status byte (the opcode, as it were); has its most significant bit set to **1**.

  b) The 4 low-order bits identify which channel this message belongs to (for 16 possible channels).

  c) The 3 remaining bits hold the message. For a data byte, the most significant bit is set to **0**.

- **A.1. Voice messages**:

  a) This type of channel message controls a voice, i.e., sends information specifying which note to play or to turn off, and encodes key pressure.

  b) Voice messages are also used to specify controller effects such as sustain, vibrato, tremolo, and the pitch wheel.

  Table 6.3 lists these operations.

Table 6.3: MIDI voice messages

Voice Message	Status Byte	Data Byte1	Data Byte2
Note Off	&H8n	Key number	Note Off velocity
Note On	&H9n	Key number	Note On velocity
Poly. Key Pressure	&HAn	Key number	Amount
Control Change	&HBn	Controller num.	Controller value
Program Change	&HCn	Program number	None
Channel Pressure	&HDn	Pressure value	None
Pitch Bend	&HEn	MSB	LSB

(** &H indicates hexadecimal, and 'n' in the status byte hex value stands for a channel number. All values are in 0..127 except Controller number, which is in 0..120)

- **A.2. Channel mode messages**:

  a) Channel mode messages: special case of the Control Change message → opcode B (the message is &HBn, or 1011nnnn).

  b) However, a Channel Mode message has its first data byte in 121 through 127 (&H79–7F).

  c) Channel mode messages determine how an instrument processes MIDI voice messages: respond to all messages, respond just to the correct channel, don't respond at all, or go over to local control of the instrument.

  d) The data bytes have meanings as shown in Table 6.4.

Table 6.4: MIDI mode messages

1st Data Byte	Description	Meaning of 2nd Data Byte
&H79	Reset all controllers	None; set to 0
&H7A	Local control	0 = off; 127 = on
&H7B	All notes off	None; set to 0
&H7C	Omni mode off	None; set to 0
&H7D	Omni mode on	None; set to 0
&H7E	Mono mode on (Poly mode off)	Controller number
&H7F	Poly mode on (Mono mode off)	None; set to 0

- **B. System Messages:**

  a) System messages have no channel number — commands that are not channel specific, such as timing signals for synchronization, positioning information in pre-recorded MIDI sequences, and detailed setup information for the destination device.

  b) Opcodes for all system messages start with &HF.

  c) System messages are divided into three classifications, according to their use:

B.1. **System common messages**: relate to timing or positioning.

Table 6.5: MIDI System Common messages.

System Common Message	Status Byte	Number of Data Bytes
MIDI Timing Code	&HF1	1
Song Position Pointer	&HF2	2
Song Select	&HF3	1
Tune Request	&HF6	None
EOX (terminator)	&HF7	None

B.2. **System real-time messages**: related to synchronization.

Table 6.6: MIDI System Real-Time messages.

System Real-Time Message	Status Byte
Timing Clock	&HF8
Start Sequence	&HFA
Continue Sequence	&HFB
Stop Sequence	&HFC
Active Sensing	&HFE
System Reset	&HFF

B.3. **System exclusive message**: included so that the MIDI standard can be extended by manufacturers.

a) After the initial code, a stream of any specific messages can be inserted that apply to their own product.

b) A System Exclusive message is supposed to be terminated by a terminator byte &HF7, as specified in Table 6.

c) The terminator is optional and the data stream may simply be ended by sending the status byte of the next message.

6.2.4 General MIDI

- General MIDI is a scheme for standardizing the assignment of instruments to patch numbers.

  a) A standard percussion map specifies 47 percussion sounds.

  b) Where a “note” appears on a musical score determines what percussion instrument is being struck: a bongo drum, a cymbal.

  c) Other requirements for General MIDI compatibility: MIDI device must support all 16 channels; a device must be multitimbral (i.e., each channel can play a different instrument/program); a device must be polyphonic (i.e., each channel is able to play many voices); and there must be a minimum of 24 dynamically allocated voices.

- **General MIDI Level2**: An extended general MIDI was defined in 1999 and updated in 2003, with a standard `.smf` “Standard MIDI File” format defined — inclusion of extra character information, such as karaoke lyrics.

6.2.5 MIDI to WAV Conversion

- Some programs, such as early versions of Premiere, cannot include `.mid` files — instead, they insist on `.wav` format files.

  a) Various shareware programs exist for approximating a reasonable conversion between MIDI and WAV formats.

  b) These programs essentially consist of large lookup files that try to substitute pre-defined or shifted WAV output for MIDI messages, with inconsistent success.