CHAPTER 3

Module 3

Digital Audio

Syllabus Topic : Basic Sound Concepts - Computer representation of sound

3.1 Basic Sound Concepts

3.1.1 What is Sound ?

- Sound is a physical phenomenon produced by the vibration of matter. The matter can be almost anything: a violin string or a block of wood, for example.
- As the matter vibrates, pressure variations are created in the air surrounding it. This alternation of high and low pressure is propagated through the air in a wave-like motion. When the wave reaches our ears, we hear a sound. Fig. 3.1.1 graphs the oscillation of a pressure wave over time.

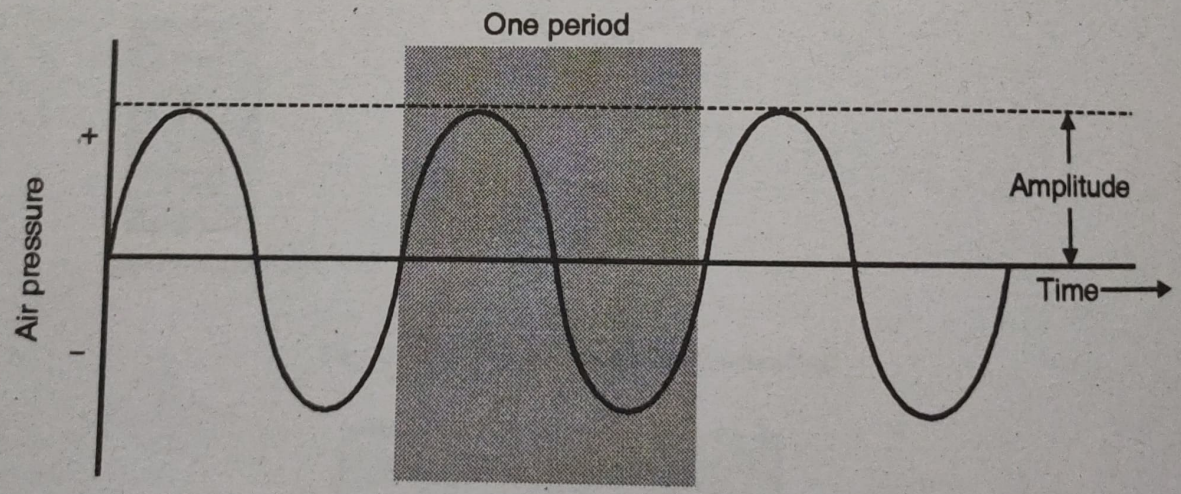


Fig. 3.1.1 : Air Pressure Wave

- The pattern of the pressure oscillation is called a *waveform*. Notice that the waveform in Fig. 3.1.1 repeats the same shape at regular intervals; the shaded area shows one complete shape.
- This portion of the waveform is called a *period*. A waveform with a clearly defined period occurring at regular intervals is called a*periodic waveform*.

- Since they occur naturally, sound waveforms are never as perfectly smooth nor as uniformly periodic as the waveform shown in Fig. 3.1.1.
- However, sounds that display a recognizable periodicity tend to be more musical than those that are nonperiodic.

Here are some sources of periodic and nonperiodic sounds :

☞ **Periodic**

- Musical instruments other than unpitched percussion
- Vowel sounds
- Bird songs
- Whistling wind

☞ **Nonperiodic**

- Unpitched percussion instruments
- Consonants, such as "t," "f," and "s"
- Coughs and sneezes
- Rushing water

3.1.2 Characteristics of Sound Waves

☞ **Frequency**

- **Frequency** is defined as the number of vibrations, oscillations, or cycles in a repeating process occurring per unit time. In the context of sound, it is the number of compressions passing a fixed point of reference in one second. The resulting unit of frequency is called Hertz (Hz).
- Frequency is perceived as pitch. The *frequency* of a sound is the number of times the pressure rises and falls, or oscillates, in a second is measured in *hertz* (Hz). A frequency of 100 Hz means 100 oscillations per second. A convenient abbreviation, kHz for *kilohertz*, is used to indicate thousands of oscillations per second: 1 kHz equals 1000 Hz.
- The frequency range of normal human hearing extends from around 20 Hz up to about 20 kHz.
- The frequency axis is logarithmic, not linear: To traverse the audio range from low to high by equal-sounding steps, each successive frequency increment must be greater than the last.
- For example, the frequency difference between the lowest note on a piano and the note an octave above it is about 27 Hz. Compare this to the piano's top octave, where the frequency difference is over 2000 Hz. Yet, subjectively, the two intervals sound the same.

☞ **Amplitude**

– **Amplitude** is the maximum change in value of a parameter during the oscillation of a wave. In this unit, that parameter will usually be pressure. In practice, this is the distance between a peak or trough and the x-axis on a graph. A sound also has an *amplitude*, a property subjectively heard as loudness. The amplitude of a sound is the measure of the displacement of air pressure from its mean or quiescent state. The greater the amplitude, the louder the sound.

☞ **Phase**

The phase of a wave is an expression of how far through its cycle of oscillation it has progressed. Because the mathematical description of wave motion is similar to the mathematical description of motion in a circle, wave phase is expressed in degrees or radians. A wave completes its cycle in 360°, just as a circle is completed in 360°. Half a cycle is 180°, and a quarter cycle is 90°.

The relative phase of two similar waves is a measure of how synchronized they are :

– **In phase** : Similar waves whose peaks (maxima) coincide are said to be in phase. Their phase difference is 0°.

– **Out of phase** : Similar waves whose peaks (maxima) do not coincide are said to be out of phase. If the peaks of one coincide with troughs (minima) of the other, the two waves are said to be 180° out of phase.

– **Period** - The time required for a single wavelength to pass a fixed point of reference. The period of a sound wave is the inverse of its frequency.

– **Wavelength** - The distance between one peak or crest of a sound wave and the next corresponding peak or crest. The product of the wavelength and the frequency of a sound wave yields the velocity of that wave.

– **Waveform** - The detailed way in which a parameter changes during the oscillation of a wave. For sound, that parameter will usually be pressure. In practice, this is the shape of a wave on a graph. Waveform is perceived as timbre

3.1.3 How the Computer Represents Sound ?

The smooth, continuous curve of a sound waveform isn't directly represented in a computer. A computer measures the amplitude of the waveform at regular time intervals to produce a series of numbers. Each of these measurements is called a *sample*. Fig. 3.1.2 illustrates one period of a digitally sampled waveform.

Audio means "of sound" or "of the reproduction of sound". Specifically, it refers to the range of frequencies detectable by the human ear - approximately 20Hz to 20 kHz. It's not a bad idea to memorise those numbers - 20 Hz is the lowest-pitched (bassiest) sound we can hear, 20 kHz is the highest pitch we can hear. Audio work involves the production, recording, manipulation and reproduction of sound waves.

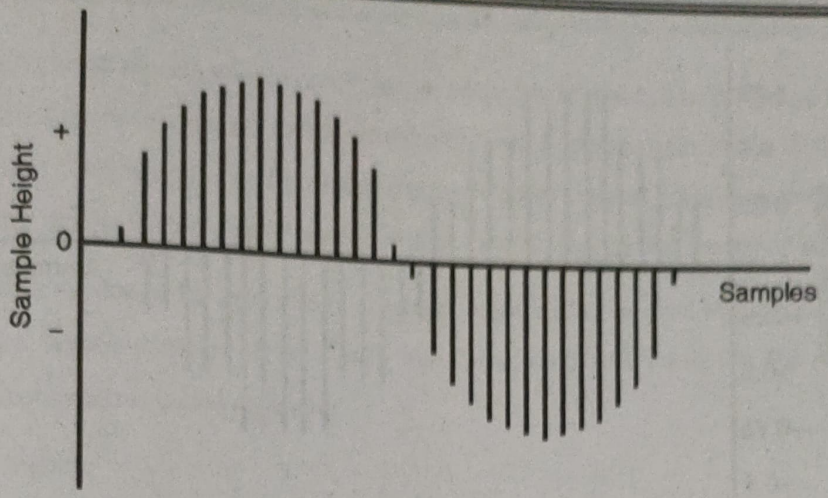


Fig. 3.1.2 : Sampled Waveform

Each vertical bar in Fig. 3.1.2 represents a single sample. The height of a bar indicates the value of that sample.

☞ Sampling Rate

The rate at which a waveform is sampled is called the *sampling rate*. Like frequencies, sampling rates are measured in hertz. The rate can vary typically from 5000-90,000 samples per second. The audio input from a source is sampled several thousand times per second. Each sample is a snapshot of the original signal at a particular time. The CD standard sampling rate of 44100 Hz means that the waveform is sampled 44100 times per second. This may seem a bit excessive, considering that we can't hear frequencies above 20 kHz; however, the highest frequency that a digitally sampled signal can represent is equal to half the sampling rate. So a sampling rate of 44100 Hz can only represent frequencies up to 22050 Hz, a boundary much closer to that of human hearing.

☞ Quantization

A waveform is sampled at discrete times, the value of the sample is also discrete. The *quantization* of a sample value depends on the number of bits used in measuring the height of the waveform. In the case of 8-bit *quantization*, this value is between 0 and 255 (or -128 and 127). In 16-bit digitization, this value is between 0 and 65,535 (or -32,768 and 32,767). Digitized signal can take only certain (discrete) values. The process of digitization introduces noise in a signal. This is related to the number of bits per sample. A higher number of bits used to store the sampled value leads to a more accurate sample, with less noise.

As an extreme example, Fig. 3.1.2 shows the waveform used in the previous example sampled with a 3-bit quantization. This results in only eight possible values : .75, .5, .25, 0, – .25, –.5, –.75, and –1. As you can see, the shape of the waveform becomes less discernible with a coarser quantization. The coarser the quantization, the "buzzier" the sound.

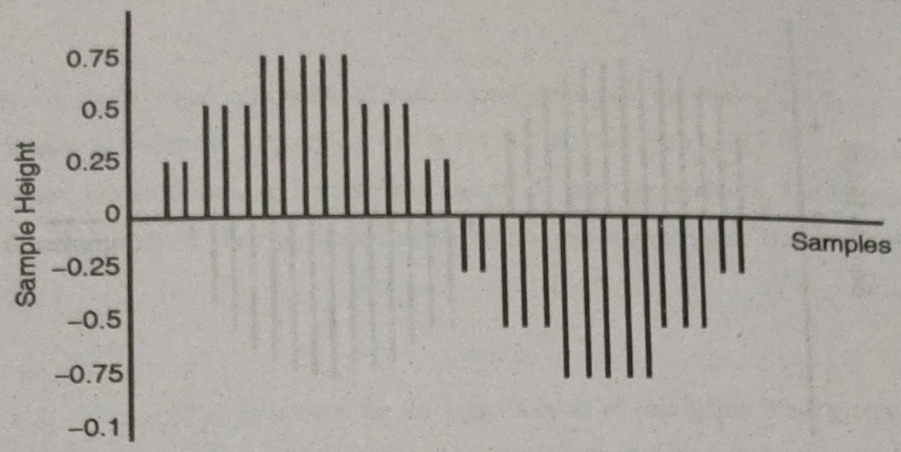


Fig. 3.1.3 : Three-Bit Quantization

☞ Storing Sampled Data

An increased sampling rate and refined quantization improves the fidelity of a digitally sampled waveform; however, the sound will also take up more storage space. Five seconds of sound sampled at 44.1 kHz with a 16-bit quantization uses more than 400,000 bytes of storage a minute will consume more than five megabytes. A number of data compression schemes have been devised to decrease storage while sacrificing some fidelity.

3.1.4 Digitization of Audio

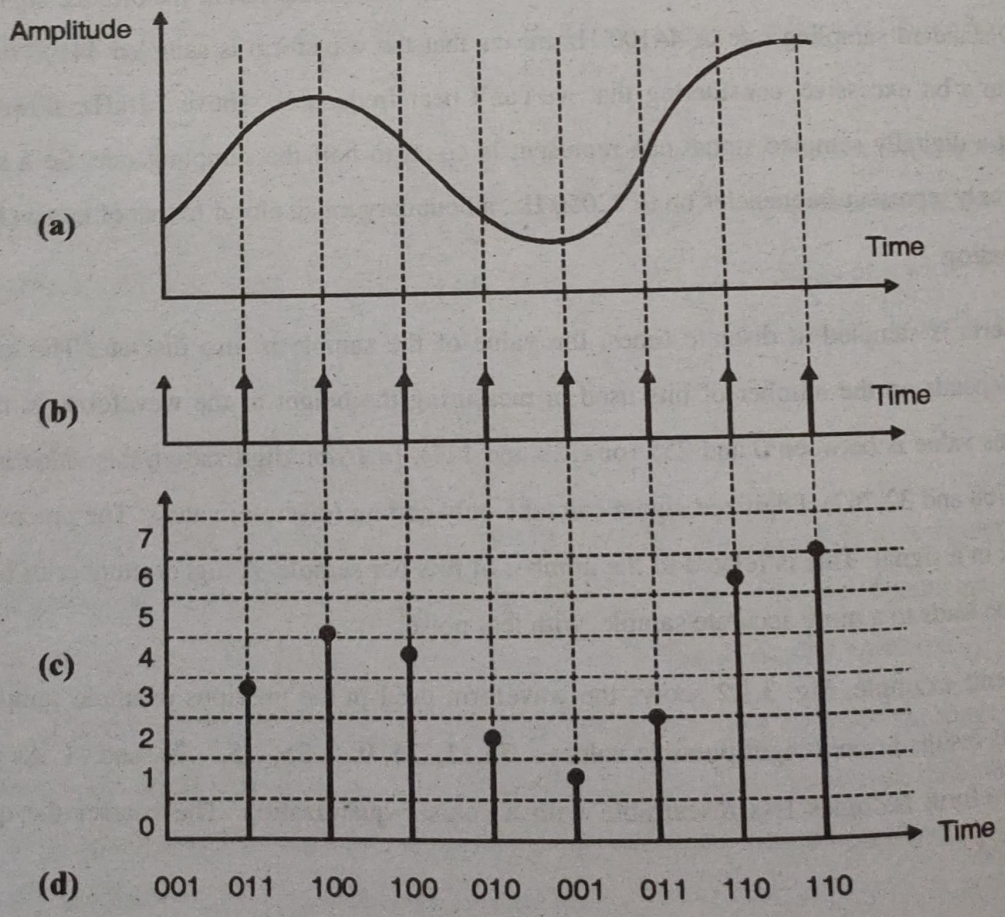


Fig. 3.1.4 : Analog-to-digital conversion process

Sound is produced when objects vibrate producing pressure waves that can be picked up by human's ear. The vibrating pressure waves move in a pattern called waveform. If we graph the intensity or motion of the wave over time, we can get a curve consisting of a series of waveforms. We can say that sound is stored by waves. These are *analog signals*. In other words, analog signals are continuous variable signals that consist of waves.

When the sound is needed to be used in any computer application, we need to converts the air vibrations of sound into an electrical signal, which is called *digital signal* - a stream of 0's and 1's. The process of conversing analog signals to digital signals is called *digitizing*.

(a) Original analog signal;

(b) Sampling pulses;

(c) Sampled values and quantization intervals;

(d) Digitized sequence.

Three stages are involved in digitization of audio : sampling, quantization, and coding

Sampling

The process of converting continuous time into discrete values is called sampling. Fig. 3.1.4(b) and (c) show the sampling process. The time axis is divided into fixed intervals. The reading of the instantaneous value of the analog signal is taken at the beginning of each time interval. This interval is determined by a clock pulse. The frequency of the clock is called the sampling rate or sampling frequency. The sampled value is held constant for the next time interval.

Quantization

The process of converting continuous sample values into discrete values is called quantization. In this process we divide the signal range into a fixed number of intervals. Each interval is of the same size and is assigned a number. In Fig. 3.1.4(c), these intervals are numbered from 0 to 7.

Each sample falls in one of the intervals and is assigned that interval's number. In doing this, each sample has a limited choice of values. In our example, a sample value can only be an integer number between 0 and 7. Before quantization, the last two samples in Fig. 3.1.4(c) have different values. But they have the same value of 6 after quantization. The size of the quantization interval is called the quantization step.

Coding

The process of representing quantized values digitally is called coding Fig. 3.1.4(d). In the above example, eight quantizing levels are used. These levels can be coded using 3 bits if the binary system is used, so each sample is represented by 3 bits. The analog signal in Fig. 3.1.4(a) is represented digitally by the following series of binary numbers: 001, 011, 100, 100, 010, 001, 011, 110, and 110.

Syllabus Topic : File Formats - WAV, MPEG Audio

3.2 Audio Formats

Sound can be stored in many different formats. Some popular audio formats are listed below and most widely used audio format MIDI, MPEG Audio and wave are described in detail.

☞ **The MIDI Format**

- The MIDI (Musical Instrument Digital Interface) is a format for sending music information between electronic music devices like synthesizers and PC sound cards.
- The MIDI format was developed in 1982 by the music industry. The MIDI format is very flexible and can be used for everything from very simple to real professional music making.
- MIDI files do not contain sampled sound, but a set of digital musical instructions (musical notes) that can be interpreted by your PC's sound card.
- The downside of MIDI is that it cannot record sounds (only notes). Or, to put it another way : It cannot store songs, only tunes.
- The upside of the MIDI format is that since it contains only instructions (notes), MIDI files can be extremely small. The example above is only 23K in size but it plays for nearly 5 minutes.
- The MIDI format is supported by many different software systems over a large range of platforms. MIDI files are supported by all the most popular Internet browsers.
- Sounds stored in the MIDI format have the extension .mid or .midi.

☞ **The RealAudio Format**

- The RealAudio format was developed for the Internet by Real Media. The format also supports video.
- The format allows streaming of audio (on-line music, Internet radio) with low bandwidths. Because of the low bandwidth priority, quality is often reduced.
- Sounds stored in the RealAudio format have the extension .rm or .ram.

☞ **The AU Format**

- The AU format is supported by many different software systems over a large range of platforms.
- Sounds stored in the AU format have the extension .au.

The AIFF Format

- The AIFF (Audio Interchange File Format) was developed by Apple.
- AIFF files are not cross-platform and the format is not supported by all web browsers.
- Sounds stored in the AIFF format have the extension .aif or .aiff.

The SND Format

- The SND (Sound) was developed by Apple.
- SND files are not cross-platform and the format is not supported by all web browsers.
- Sounds stored in the SND format have the extension .snd.

The WAVE Format

- The WAVE (waveform) format is developed by IBM and Microsoft.
- It is supported by all computers running Windows, and by all the most popular web browsers .
- Sounds stored in the WAVE format have the extension .wav.

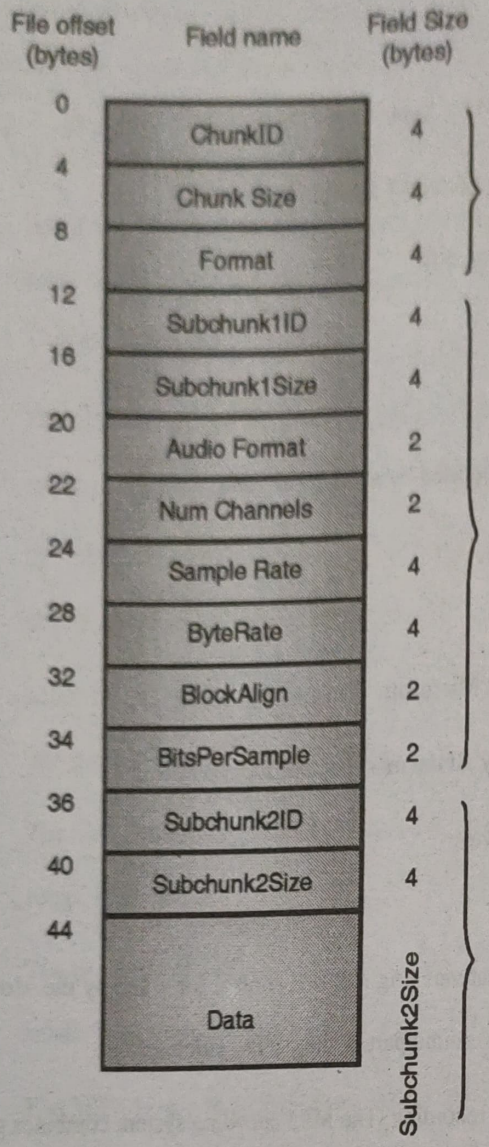
The MP3 Format (MPEG)

- MP3 files are actually MPEG files. But the MPEG format was originally developed for video by the Moving Pictures Experts Group. We can say that MP3 files are the sound part of the MPEG video format.
- MP3 is one of the most popular sound formats for music recording. The MP3 encoding system combines good compression (small files) with high quality. Expect all your future software systems to support it.
- Sounds stored in the MP3 format have the extension .mp3, or .mpga (for MPG Audio).

3.2.1 WAVE File Format

The WAVE file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF file starts out with a file header followed by a sequence of data chunks.

A WAVE file is often just a RIFF file with a single "WAVE" chunk which consists of two sub-chunks - a "fmt " chunk specifying the data format and a "data" chunk containing the actual sample data.



The "RIFF" chunk descriptor

The Format of concern here is "WAVE", which requires two sub-chunks : "fmt" and "data".

The "fmt" sub-chunk

Describes the format of the sound information in the data sub-chunk.

The "data" sub-chunk

Indicates the size of the sound information and contains the raw sound data.

Fig. 3.2.1 : Wave file format

☞ Wave File Header - RIFF Type Chunk

Wave file headers follow the standard RIFF file format structure. The first 8 bytes in the file is a standard RIFF chunk header which has a chunk ID of "RIFF" and a chunk size equal to the file size minus the 8 bytes used by the header. The first 4 data bytes in the "RIFF" chunk determines the type of resource found in the RIFF chunk. Wave files always use "WAVE". After the RIFF type comes all of the Wave file chunks that define the audio waveform.

Offset	Size	Description	Value
0x00	4	Chunk ID	"RIFF"
0x04	4	Chunk Data Size	(file size) - 8
0x08	4	RIFF Type	"WAVE"
0x10	Wave chunks		

Fig. 3.2.2 : RIFF Type Chunk Values

Wave File Chunks

There are quite a few types of chunks defined for Wave files. Many Wave files contain only two of them, specifically the Format Chunk and the Data Chunk. These are the two chunks needed to describe the format of the digital audio samples and the samples themselves.

Format Chunk - "fmt"

The format chunk contains information about how the waveform data is stored and should be played back including the type of compression used, number of channels, sample rate, bits per sample and other attributes.

Offset	Size	Description	Value
0x00	4	Chunk ID	"fmt " (0x666D7420)
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compression code	1 - 65,535
0x0a	2	Number of channels	1 - 65,535
0x0c	4	Sample rate	1 - 0xFFFFFFFF
0x10	4	Average bytes per second	1 - 0xFFFFFFFF
0x14	2	Block align	1 - 65,535
0x16	2	Significant bits per sample	2 - 65,535
0x18	2	Extra format bytes	0 - 65,535

Chunk ID and Data Size

The chunk ID is always "fmt" (0x666D7420) and the size is the size of the standard wave format data (16 bytes) plus the size of any extra format bytes needed for the specific Wave format, if it does not contain uncompressed PCM data. Note the chunk ID string ends with the space character (0x20).

Compression Code

The first word of format data specifies the type of compression used on the Wave data included in the Wave chunk found in this "RIFF" chunk.

Number of Channels

The number of channels specifies how many separate audio signals that are encoded in the wave data chunk. A value of 1 means a mono signal, a value of 2 means a stereo signal, etc.

Sample Rate

The number of sample slices per second. This value is unaffected by the number of channels.

Average Bytes Per Second

This value indicates how many bytes of wave data must be streamed to a D/A converter per second in order to play the wave file. This information is useful when determining if data can be streamed from the source fast enough to keep up with playback. This value can be easily calculated with the formula:

$$\text{AvgBytesPerSec} = \text{SampleRate} * \text{BlockAlign}$$

Block Align

The number of bytes per sample slice. This value is not affected by the number of channels and can be calculated with the formula :

$$\text{BlockAlign} = \text{SignificantBitsPerSample} / 8 * \text{NumChannels}$$

Significant Bits Per Sample

This value specifies the number of bits used to define each sample. This value is usually 8, 16, 24 or 32. If the number of bits is not byte aligned (a multiple of 8) then the number of bytes used per sample is rounded up to the nearest byte size and the unused bytes are set to 0 and ignored.

Extra Format Bytes

This value specifies how many additional format bytes follow. It does not exist if the compression code is 0 (uncompressed PCM file) but may exist and have any value for other compression types depending on what compression information is need to decode the wave data. If this value is not word aligned (a multiple of 2), padding should be added to the end of this data to word align it, but the value should remain non-aligned.

Data Chunk - "data"

The Wave Data Chunk contains the digital audio sample data which can be decoded using the format and compression method specified in the Wave Format Chunk. If the Compression Code is 1 (uncompressed PCM), then the Wave Data contains raw sample values. This document explains how an uncompressed PCM data is stored, but will not get into the many supported compression formats. Wave files usually contain only one data chunk, but they may contain more than one if they are contained within a Wave List Chunk ("wavl").

Offset	Length	Type	Description	Value
0x00	4	char[4]	chunk ID	"data" (0x64617461)
0x04	4	dword	chunk size	depends on sample length and compression
0x08	sample data			

Fig. 3.2.3 : Data Chunk Format

3.2.2 MIDI File Format

The *Standard MIDI File* (SMF) is a file format specifically designed to store the data that a sequencer records and plays (whether that sequencer be software or hardware based).

This format stores the standard MIDI messages (i.e., status bytes with appropriate data bytes) plus a time-stamp for each message. The format allows saving information about tempo, pulses per quarter note resolution (SMPTE setting), time and key signatures, and names of tracks and patterns. It can store multiple patterns and tracks so that any application can preserve these structures when loading the file.

MIDI file format saves data in *chunks* (i.e., groups of bytes preceded by an ID and size) which can be parsed, loaded, skipped, etc. Therefore, it can be easily extended to include a program's proprietary info. For example, maybe a program wants to save a "flag byte" that indicates whether the user has turned on an audible metronome click. The program can put this flag byte into a MIDI file in such a way that another application can skip this byte without having to understand what that byte is for. In the future, the MIDI file format can also be extended to include new "official" chunks that all sequencer programs may elect to load and use.

What's a Chunk ?

Data is always saved within a *chunk*. There can be many chunks inside of a MIDI file. Each chunk can be a different size (i.e., where size refers to how many bytes are contained in the chunk). The data bytes in a chunk are related in some way. A chunk is simply a group of related bytes.

Each chunk begins with a 4 character (i.e., 4 ascii bytes) *ID* which tells what "type" of chunk this is. The next 4 bytes (all bytes are comprised of 8 bits) form a 32-bit length (i.e., size) of the chunk. **All chunks must begin with these two fields** (i.e., 8 bytes), which are referred to as the *chunk header*.

The *Length* does not include the 8 byte chunk header. It simply tells you how many bytes of data are in the chunk *following this header.*

Here's an example header (with bytes expressed in hex):

4D 54 68 64 00 00 00 06

Note that the first 4 bytes make up the ascii ID of **MThd** (i.e., the first four bytes are the ascii values for 'M', 'T', 'h' and 'd'). The next 4 bytes tell us that there should be 6 more data bytes in the chunk

MThd Chunk

The MThd header has an ID of MThd, and a Length of 6.

Let's examine the 6 data bytes (which follow the above, 8 byte header) in an MThd chunk.

The first two data bytes tell the Format. There are actually 3 different types (i.e., formats) of MIDI files. A type of 0 means that the file contains one single track containing midi data on possibly all 16 midi channels. If your sequencer sorts/stores all of its midi data in one single block of memory with the data in the order that it's "played", then it should read/write this type. A type of 1 means that the file contains one or more simultaneous (i.e., all start from an assumed time of 0) tracks, perhaps each on a single midi channel. Together, all of these tracks are considered one sequence or pattern. If your sequencer separates its midi data (i.e. tracks) into different blocks of memory but plays them back simultaneously (i.e., as one "pattern"), it will read/write this type. A type of 2 means that the file contains one or more sequentially independent single-track patterns. if your sequencer separates its midi data into different blocks of memory, but plays only one block at a time (i.e., each block is considered a different "excerpt" or "song"), then it will read/write this type.

The next 2 bytes tell how many tracks are stored in the file, NumTracks. Of course, for format type 0, this is always 1. For the other 2 types, there can be numerous tracks.

The last two bytes indicate how many Pulses (i.e. clocks) Per Quarter Note (abbreviated as PPQN) resolution the time-stamps are based upon, Division. For example, if your sequencer has 96 ppqn, this field would be (in hex):

00 60

Alternately, if the first byte of Division is negative, then this represents the division of a second that the time-stamps are based upon. The first byte will be -24, -25, -29, or -30, corresponding to the 4 SMPTE standards representing frames per second. The second byte (a positive number) is the resolution within a frame (i.e., subframe). Typical values may be 4 (MIDI Time Code), 8, 10, 80 (SMPTE bit resolution), or 100.

You can specify millisecond-based timing by the data bytes of -25 and 40 subframes.

Here's an example of a complete MThd chunk (with header) :

4D 54 68 64 MThd ID

00 00 00 06 Length of the MThd chunk is always 6.

00 01 The Format type is 1.

00 02 There are 2 MTrk chunks in this file.

E7 28 Each increment of delta-time represents a millisecond.

MTrk Chunk

After the MThd chunk, you should find an *MTrk chunk*, as this is the only other currently defined chunk. An MTrk chunk contains all of the midi data (with timing bytes), plus optional non-midi data for **one track**. Obviously, you should encounter as many MTrk chunks in the file as the MThd chunk's NumTracks field indicated. The MTrk header begins with the ID of *MTrk*, followed by the Length (i.e., number of data bytes to read for this track). The Length will likely be different for each track.

3.2.3 MPEG audio

The Motion Picture Experts Group (MPEG) audio compression algorithm is an International Organization for Standardization (ISO) standard for high fidelity audio compressions. It is one of a three-part compression standard, the other two being video and system. The MPEG compression is lossy, but nonetheless can achieve transparent, perceptually lossless compression.

MPEG-Audio allows three sampling rates: 32, 44.1, or 48 kHz. The compressed bitstream can support one or two audio channels. The compressed stream can have one of several predefined fixed bit rates ranging from 32 to 244 kbps per channel. Depending on the audio sampling rate, this translates to compression ratios ranging from 2.7 to 24.

Because MPEG-Audio only removes inaudible parts of the sound, perceptually it is lossless compression, although original audio data cannot be reconstructed bit to bit after compression. The MPEG-Audio committee conducted extensive subjective listening tests during the development of the standard. The tests showed that even with a 6:1 compression ratio and under optimal listening conditions, expert listeners could not distinguish between coded and original audio clips with statistical significance. The audio clips used in the tests were chosen as difficult to compress.

MPEG-Audio is not a single compression algorithm but a family of three audio compression schemes. They are called MPEG-Audio layer 1, layer 2, and layer 3. The complexity of the algorithm increases with the layer number. Thus layer 3 is the most complex. In general, increased complexity produces a higher compression ratio. These three layers are hierarchically compatible; that is, a layer 2 decoder can decode Bit streams encoded with a layer 1 encoder, and a layer 3 decoder can decode bit-streams encoded with layer 2 and layer 1 encoders.

Using MPEG audio coding you can compress the original audio on a CD by a factor of 12 without losing the sound quality. Factors of 24 or even more are also possible, but these are achieved only by getting a sound output of reduced sampling rate and reduced resolution of the samples.

The family of audio coding schemes that achieve compression without losing the CD sound quality, along with their compression ratio are :

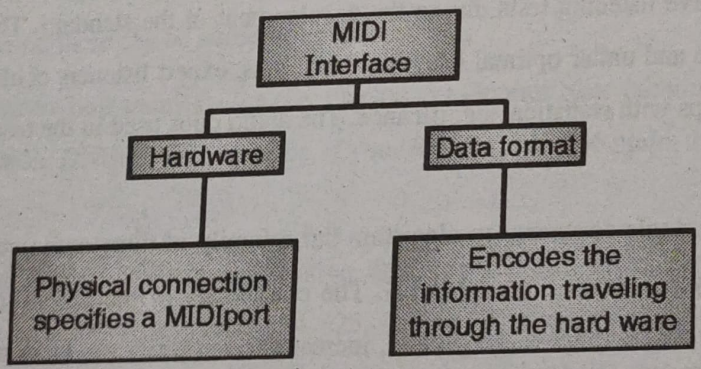
- MPEG Layer 1 (corresponds with 384 Kbps of data transferred to the player in case of a stereo signal) compression ratio is 1:4.
- MPEG Layer 2 (corresponds with 256-192 Kbps of data transferred to the player in case of a stereo signal) compression ratio is 1:6-1:8.
- MPEG Layer 3 (corresponds with 128-112 Kbps of data transferred to the player in case of a stereo signal) compression ratio is 1:10-1:12.

3.3 Basic Concepts of MIDI

Musical Instrument Digital Interface is an industry-standard protocol that enables electronic musical instruments (synthesizers, drum machines), computers and other electronic equipment (MIDI controllers, sound cards, samplers) to communicate and synchronize with each other. Unlike analog devices, MIDI does not transmit an audio signal - it sends event messages about pitch and intensity, control signals for parameters such as volume, vibrato and panning, cues, and clock signals to set the tempo.

The Musical Instrument Digital Interface (MIDI) protocol has been widely accepted and utilized by musicians and composers since its conception in 1983. MIDI data is a very efficient method of representing musical performance information, and this makes MIDI an attractive protocol not only for composers or performers, but also for computer applications which produce sound, such as multimedia presentations or computer games.

However, the lack of standardization of synthesizer capabilities hindered applications developers and presented new MIDI users with a rather steep learning curve to overcome.



(a) Hardware Aspects of MIDI

- Three 5-pin ports found 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.

Fig. 3.3.1 illustrates a typical setup where :

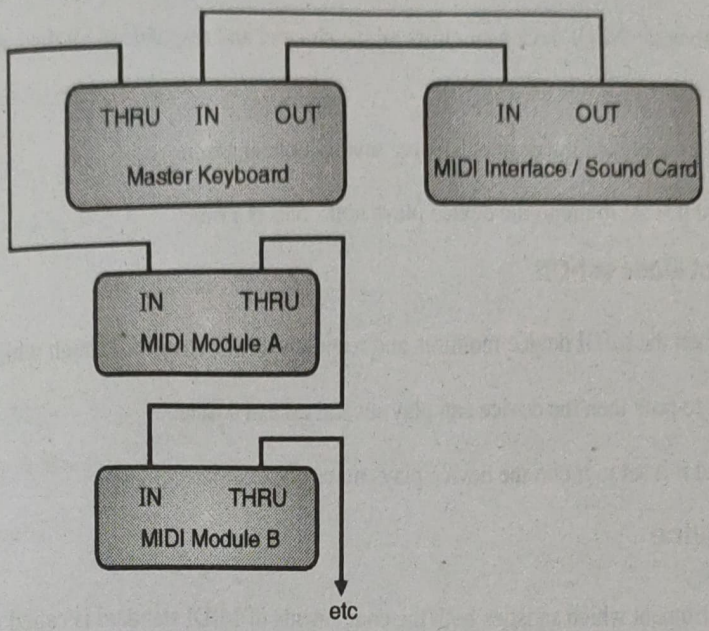


Fig. 3.3.1

A Typical MIDI Sequencer Setup

- MIDI OUT of synthesizer is connected to MIDI IN of sequencer.
- MIDI OUT of sequencer is connected to MIDI IN of synthesizer and through to each of the additional sound modules.
- During recording, the keyboard-equipped synthesizer is used to send MIDI message to the sequencer, which records them.
- During play back : messages are send out from the sequencer to the sound modules and the synthesizer which will play back the music

(b) MIDI standard identifications

Bitstream encoding format for MIDI "messages" that, in the words of the standards document, "can be thought of as instructions which tell a music synthesizer how to play a piece of music."

Three levels have been established to manage player conformance :

- General MIDI System Level 1 (GM1), designed to provide the minimum level of compatibility among MIDI hardware and software; includes 128 presets for instruments and 47 for percussion.
- General MIDI System Level 2 (GM2), extensions to provide greater functionality, may not be as widely supported.

– General MIDI Lite (GM lite), reduced performance, especially in mobile applications.

(c) MIDI Reception Modes

☞ **Functionality of Mode set-ON**

– ON mode is set then the MIDI device monitors all the channel and responds to all the messages irrespective of the channels through which it is transmitted.

– If further it is set to poly then the device can play several note at a time.

– On the other hand if it set to mono the device plays notes one at a time.

☞ **Functionality of Mode set-Off**

– Off mode is set then the MIDI device monitors and responds to that channel through which it is transmitted.

– If further it is set to poly then the device can play several note at a time.

– On the other hand if it set to mono the device plays notes one at a time.

3.3.1 MIDI Device

Any musical instrument which satisfies both the components of MIDI standard is called a MIDI Device. Such a MIDI device is capable of communicating with other MIDI devices through channels. The MIDI standard specifies 16 channels. The heart of any MIDI system is the MIDI synthesizer. A typical synthesizer looks like a simple piano keyboard with a panel full of buttons.

☞ **Sound generators**

It does the actual synthesizing sound. It is to produce the audio signal that becomes a sound when fed into the loud speaker.

☞ **Micro processors**

It communicate with the key board and control panel. It specifies the note and sound command to the sound generators i.e. it sends and receive MIDI messages.

☞ **Key board**

A **MIDI keyboard** is a piano-style digital keyboard device used for sending Musical Instrument Digital Interface (MIDI) signals or commands to other devices connected to the same interface as the keyboard. The basic MIDI keyboard does not produce sound.

Instead, MIDI information is sent to an electronic module capable of reproducing an array of digital sounds or samples that resemble traditional analog musical instruments. MIDI keyboards offers a direct control of the synthesizer to the musician. MIDI keyboards can have different amounts of keys. Classical musicians and purists will insist on all 88 keys being present. Keyboards with 76 keys are also popular.

Control panel

The control panel control those function that are not directly controlled by keyboard. Control panel consist of slider for setting overall volume of synthesizer, button to turn on and off synthesizer and menu for calling different patches for sound generator to play.

Auxillary controller

Auxillary controllers are used to give more control over the notes played on keyboard. Pitch bend controllers can bend pitch up and down, adding portamento to notes. Modulation controller are used to increase or decrease vibrato.

Memory

Synthesizers memory is used to stores patches for sound generator. It can also be used to store setting on control panel. Many synthesizers have slot for external memory.

An MIDI studio consists of :

- **Controller :** Musical performance device that generates MIDI signal when played.
- **MIDI Signal :** A sequence of numbers representing a certain note.
- **Synthesizer :** A piano-style keyboard musical instrument that simulates the sound of real musical instruments.
- **Sequencer :** A device or a computer program that records a MIDI signal.
- **Sound Module :** A device that produces pre-recorded samples when triggered by a MIDI controller or sequencer.

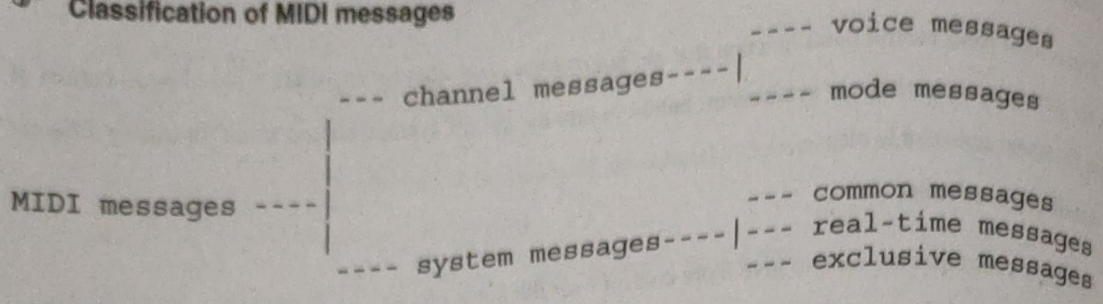
3.3.2 MIDI Messages

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**



A. Channel messages

Each MIDI cable sends 16 channels of information and the channel messages are messages that affect *each channel* independent of one another, the system messages are messages that affect the *entire* MIDI module . Thus Channel messages are messages that affect *each channel* independent of one another.

There are two types of channel messages :

- Channel Voice Messages
- Channel Mode Messages.

1. Channel voice messages

Almost all MIDI devices are equipped to receive MIDI messages on one or more of 16 selectable MIDI channel numbers. A device's particular voice (or patch, program, timbre) will respond to messages sent on the channel it is tuned to and ignore all other channel messages, analogous to a television set receiving only the station it is tuned to and rejecting the others. channel voice messages convey information about whether to turn a note on or off, what patch to change to, how much key pressure to exert (called *aftertouch),* etc.

Message Type	Message Byte (Hex)	Description
Note off	8n kk vv	KK : Note (key) number from 0-127 VV : Key velocity from 0-127
Note on	9 n kk vv	KK : Note (key) number from 0-127 VV : Key velocity from 0-127 0 : Note off
Polyphonic key Pressure	Ankk VV	KK : Same above key pressure value from 0-127 VV : Control no from 0-127

Message Type	Message Byte (Hex)	Description
Control Change	Bn CC VV	CC : Control no from 0-127 VV : Control value from 0-127
Program change	Cn pp	PP : Program number from 0-127
Channel pressure	Dn VV	VV : Channel pressure value from 0-127
Pitch Wheel change	En 1b hb	*l*b : Low byte from 0-127 Hb : High byte from 0-127

2. 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. Mode Messages are used to assign voice relationship for up to 16 channels that is to set the device to MONO mode or POLY mode. Omni mode on enables device to receive voice messages on all channels.

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)	**;
&H7F	Poly mode on (Mono mode off)	None; set to 0

** 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.

B. 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.

1. System real-time messages

- These messages are related to synchronization.
- These messages are used for setting values of real time parameters of a system such as start, stop sequencer.

Message Type	Message Byte (Hex)	Description
Timing clock	F8	Clock is set to 24 clocks/quarter note - System get synchronized to this clock.
Undefined	F9	
Start	FA	This message starts sequencer when play switch is pressed
Continue	FB	This will continue stopped sequencer
Stop	FC	Stop the sequencer when stop switch is pressed
Undefined	FD	
Active sensing	FE	Checks if sequencer is performing an operation.
Reset	FF	It initializes the system.

2. System common messages

These messages are common to complete system. These messages provide for function such as selecting a song, setting song position point with number of beats and sending a tune request to an analog synthesizer.

Message type	Message bytes (Hex)	Description
	F1	Undefined
Song position pointer	F2 IS MS	IS : Least significant byte of ptr ms : Most significant byte of pointer
Song select	F3 SS F4 F5	SS : Song number Undefined Undefined

Message type	Message bytes (Hex)	Description
Tune request	F6	
End of system Exclusive message	F7	EOX : Flag for system exclusive message.

3. System exclusive message

- These messages contain manufacturer specific data such as identification, serial number, model number and other information. *System exclusive message* are related to things that cannot be standardized. 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).

3.3.3 MIDI SMPTE Timing Standard

The **Society of Motion Picture and Television Engineers** or **SMPTE**, founded in 1916 as the Society of Motion Picture Engineers or SMPE is an international professional association, based in the United States of America, of engineers working in the motion imaging industries. An internationally-recognized standards developing organization, SMPTE has over 400 standards, Recommended Practices and Engineering Guidelines for television, motion pictures, digital cinema, audio and medical imaging.

Significant standards promulgated by SMPTE include :

- All film and television transmission formats and media, including digital.
- Physical interfaces for transmission of television signals and related data (such as SMPTE time code and the Serial Digital Interface).

SMPTE timecode is a set of cooperating standards to label individual frames of video or film with a time code defined by the Society of Motion Picture and Television Engineers in the SMPTE 12M specification. SMPTE revised the standard in 2008, turning it into a two-part document: SMPTE 12M-1 and SMPTE 12M-2, including important new explanations and clarifications.

Timecodes are added to film, video or audio material, and have also been adapted to synchronize music. They provide a time reference for editing, synchronization and identification. **Timecode** is a form of media metadata. The invention of timecode made modern videotape editing possible, and led eventually to the creation of non-linear editing systems.

SMPTE con be used with MIDI. The MIDI/ SMPTE synchronizer lets the user specify different tempos and exact points in SMPTE timing at which each tempo is to start, change and stop. The synchronizer keeps these tempos and timing points in memory. As a SMPTE video deck plays and sends a streamed of SMPTE times to the synchronizer, the synchronizer checks the incoming time and sends out MIDI clocks at a corresponding tempo.

SMPTE counts off the passage of time in terms of seconds, minutes, and hours (i.e., the way that non-musicians count time). It also breaks down the seconds into smaller units called "frames". The movie industry created SMPTE, and they adopted 4 different frame rates. You can divide a second into 24, 25, 29, or 30 frames. Later on, even finer resolution was needed by musical devices, and so each frame was broken down into "subframes". So, SMPTE is not directly related to musical tempo. SMPTE time doesn't vary with "musical tempo".

Many devices use SMPTE to sync their playback. If you need to sychronize with such a device, then you may need to deal with SMPTE timing. Of course, you're probably still going to have to maintain some sort of PPQN clock, based upon the passing SMPTE subframes, so that the user can adjust the tempo of the playback in terms of BPM, and can consider the time of each event in terms of bar:beat:tick. But since SMPTE doesn't directly relate to musical tempo, you have to interpolate (i.e., calculate) your PPQN clocks from the passing of subframes/ frames/ seconds/ minutes/hours

3.3.4 MIDI Software

There are hundreds if not thousands of software applications that involve MIDI, either actively or passively. Included are MIDI Sequencers (now commonly combined with audio recording into Digital Audio Workstations or 'DAWs'.), auto accompaniment applications, notation programs, music teaching software, games, DJ/remix environments, and more.

There are four types of software applications and they are :

1. Musical notations and printing applications
2. Music recording and performances applications
3. Synthesizer patch editors and librarians
4. Music education application

1. Musical notations and printing applications

This type of application software provides functions such as recording of MIDI messages and Playing back the messages in performance.

2. Musical notations and printing applications

This type of application software allows writing music using traditional musical notation. User can play back music or print the music on paper for live performance or publication.

3. Synthesizer patch editors and librarians

This type of application software allows storing and editing the information of synthesizer pitches.

4. **Music education application**

This type of application software teaches different aspect of music using computer and other controller of attached MIDI instruments.

Syllabus Topic : Compression - PCM, DM, DPCM

3.4 Audio Compression

- Audio consists of analog signals of varying frequencies. Audio signal are converted to digital form and then processed, stored and transmitted.
- Audio compression allows storing audio in less space and if there is need to transmit the signal, transmitting is more efficient and at lower cost. **Audio compression** is a form of data compression designed to reduce the transmission bandwidth requirement of digital audio streams and the storage size of audio files. Audio compression algorithms are implemented in computer software as audio codecs.

Schemes of Audio Compression

The most commonly used compression schemes for audio are :

- Pulse Code Modulation
- ADPCM (Adaptive Differential Pulse Code Modulation)
- Delta Modulation

3.4.1 Pulse Code Modulation

A signal is pulse code modulated to convert its analog information into a binary sequence, i.e., 1s and 0s. The output of a PCM will resemble a binary sequence.

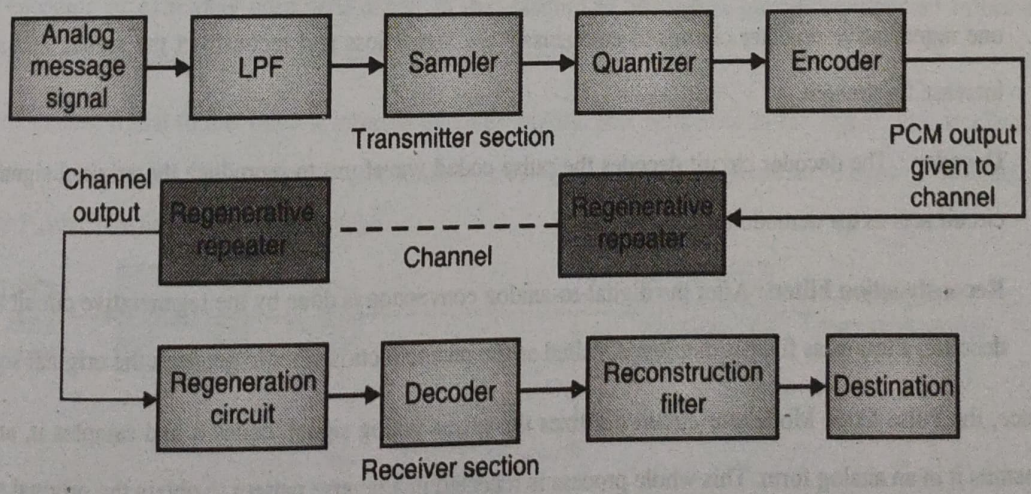


Fig. 3.4.1

- The transmitter section of a Pulse Code Modulator circuit consists of **Sampling, Quantizing** and **Encoding**, which are performed in the analog-to-digital converter section. The low pass filter prior to sampling prevents aliasing of the message signal.
- The basic operations in the receiver section are **regeneration of impaired signals, decoding,** and **reconstruction** of the quantized pulse train.
- Following is the block diagram of PCM which represents the basic elements of both the transmitter and the receiver sections.
  - **Low Pass Filter :** This filter eliminates the high frequency components present in the input analog signal which is greater than the highest frequency of the message signal, to avoid aliasing of the message signal.
  - **Sampler :** This is the technique which helps to collect the sample data at instantaneous values of message signal, so as to reconstruct the original signal. The sampling rate must be greater than twice the highest frequency component W of the message signal, in accordance with the sampling theorem.
  - **Quantizer :** Quantizing is a process of reducing the excessive bits and confining the data. The sampled output when given to Quantizer, reduces the redundant bits and compresses the value.
  - **Encoder :** The digitization of analog signal is done by the encoder. It designates each quantized level by a binary code. The sampling done here is the sample-and-hold process. These three sections (LPF, Sampler, and Quantizer) will act as an analog to digital converter. Encoding minimizes the bandwidth used.
  - **Regenerative Repeater** : This section increases the signal strength. The output of the channel also has one regenerative repeater circuit, to compensate the signal loss and reconstruct the signal, and also to increase its strength.
  - **Decoder :** The decoder circuit decodes the pulse coded waveform to reproduce the original signal. This circuit acts as the demodulator.
  - **Reconstruction Filter :** After the digital-to-analog conversion is done by the regenerative circuit and the decoder, a low-pass filter is employed, called as the reconstruction filter to get back the original signal.
- Hence, the Pulse Code Modulator circuit digitizes the given analog signal, codes it and samples it, and then transmits it in an analog form. This whole process is repeated in a reverse pattern to obtain the original signal.

3.4.2 Adaptive Differential Pulse Code Modulation (ADPCM)

ADPCM stands for Adaptive Differential Pulse Code Modulation. **ADPCM** is a very widely used lossy coding scheme. The acronym refers to waveform codecs which instead of quantizing the sound signal directly, like PCM codecs, quantize the difference between the sound signal and a prediction that has been made of the sound signal. If the prediction is accurate then the difference between the real and predicted sound samples will have a lower variance than the real sound samples, and will be accurately quantized with fewer bits than would be needed to quantize the original sound samples. At the decoder the quantized difference signal is added to the predicted signal to give the reconstructed sound signal. Using this technique one can achieve about 40-80% compression.

For the samples that are highly correlated, when encoded by PCM technique, leave redundant information behind. To process this redundant information and to have a better output, it is a wise decision to take a predicted sampled value, assumed from its previous output and summarize them with the quantized values. Such a process is called as **Differential PCM (DPCM)** technique. DPCM differs from PCM because it quantizes the difference of the actual sample and predicted value. That is the reason it is called as differential PCM.

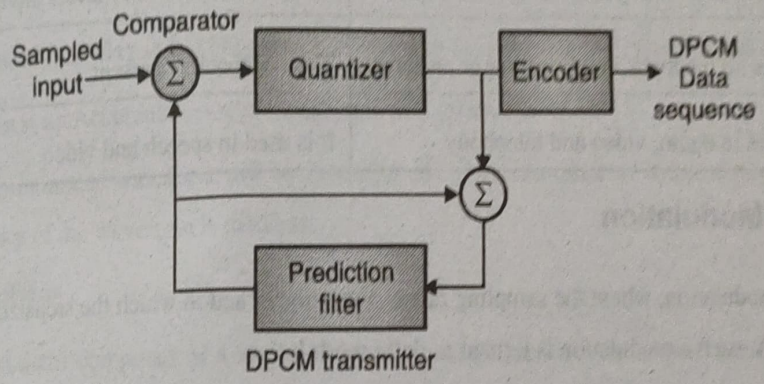


Fig. 3.4.2

The encoding of extremely correlated signal in the standard PCM system produces redundant information. Through eliminating redundancy more efficient signal can be reduced.

The redundant signal future value is inferred by analysing the past behaviour of the signal. This prediction of future value gives rise to differ rential quantization technique. When the quantizer output is encoded, the Differential Pulse Code Modulation is obtained.

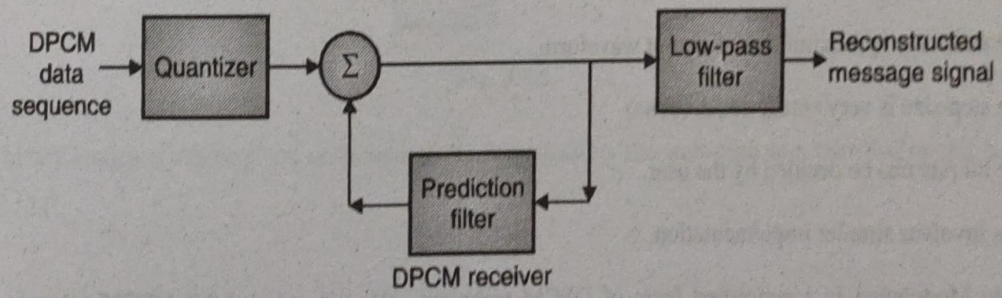


Fig. 3.4.3

3.4.3 PCM vs DPCM

Sr. No.	PCM	DPCM
1.	The PCM stands for pulse code modulation.	The DPCM stands for differential pulse code modulation.
2.	The PCM (pulse code modulation) can be use 4, 8 or 16 bit per sample.	The DPCM (differential pulse code modulation) can be use 2 or 3 bit per sample.
3.	It has required highest bandwidth.	It has required small bandwidth.
4.	The quantization error depends on number of levels.	Slope over load distortion and quantization noise is present.
5.	The number level depends on number of bit.	It has fixed number of levels are used.
6.	There is no feedback in transmitter or receiver.	The Feedback is present.
7.	It is used in audio, video and telephony.	It is used in speech and video.

3.4.4 Delta Modulation

The type of modulation, where the sampling rate is much higher and in which the stepsize after quantization is of a smaller value Δ, such a modulation is termed as **delta modulation**.

Following are some of the features of delta modulation :

- An over-sampled input is taken to make full use of the signal correlation.
- The quantization design is simple.
- The input sequence is much higher than the Nyquist rate.
- The quality is moderate.
- The design of the modulator and the demodulator is simple.
- The stair-case approximation of output waveform.
- The step-size is very small, i.e., Δ (delta).
- The bit rate can be decided by the user.
- This involves simpler implementation.

Delta Modulation is a simplified form of DPCM technique, also viewed as **1-bit DPCM scheme**. As the sampling interval is reduced, the signal correlation will be higher.

Delta Modulator

- The Delta Modulator comprises of a 1-bit quantizer and a delay circuit along with two summer circuits. Following is the block diagram of a delta modulator.

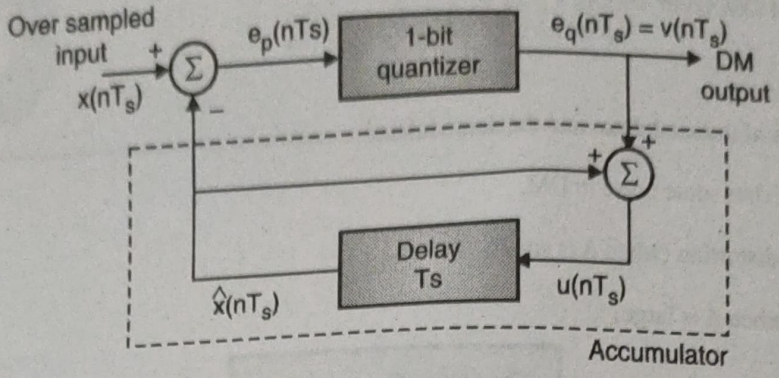


Fig. 3.4.4

- The predictor circuit in DPCM is replaced by a simple delay circuit in DM.
- The present input of the delay unit = (The previous output of the delay unit) + (The present quantizer output)
- Delay unit output is an Accumulator output lagging by one sample.
- A Stair-case approximated waveform will be the output of the delta modulator with the step-size as delta (Δ). The output quality of the waveform is moderate.

Delta Demodulator

- The delta demodulator comprises of a low pass filter, a summer, and a delay circuit. The predictor circuit is eliminated here and hence no assumed input is given to the demodulator.
- Following is the diagram for delta demodulator.

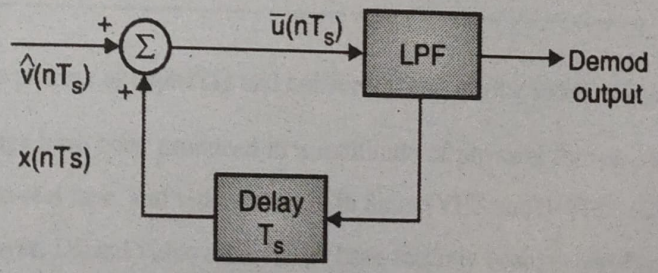


Fig. 3.4.5

- A binary sequence will be given as an input to the demodulator. The stair-case approximated output is given to the LPF.

- Low pass filter is used for many reasons, but the prominent reason is noise elimination for out-of-band signals. The step-size error that may occur at the transmitter is called **granular noise**, which is eliminated here. If there is no noise present, then the modulator output equals the demodulator input.

☞ **Advantages of DM Over DPCM**

- 1-bit quantizer
- Very easy design of the modulator and the demodulator
- However, there exists some noise in DM.
- Slope Over load distortion (when Δ is small)
- Granular noise (when Δ is large)

Review Questions

Q. 1 Explain computer representation of sound.

Q. 2 Explain MIDI file format.

Q. 3 Explain WAVE file format.

Q. 4 Explain MIDI messages.

Q. 5 What are the MIDA devices ?

Q. 6 Explain speech analysis.

Q. 7 Explain speech transmission.

Q. 8 Explain different audio compression algorithms.

Chapter Ends...

□□□