Audio Background

# Introduction

Most of us go through our lives hearing the world around us without really thinking about it. Listening is such an integral part of our existence that its details don’t even begin to surface. But there’s a lot to audio, and also a lot of details that go into audio reproduction on a computer. This document serves as background for committee members who wish to work on the std::audio library.

# What is Sound?

For most human-scale audio, sound can be modeled as waves in an elastic medium. The human ear responds to changes in air pressure, and when you speak, you are vibrating the air around your mouth and sending a pressure wave through the air. The human ear is actually incredibly sensitive, and can be capable of detecting a dynamic range of over 100 decibels (dB)[[1]](#footnote-1).

In fact, any one-dimensional property can be used to represent air pressure levels over time. Some common examples:

|  |  |
| --- | --- |
| Property | Common Usage |
| Voltage | Speakers, Microphones |
| Current | Dynamic microphones |
| Optical transmittivity | Film |
| Magnetic orientation | Tape |
| Physical displacement | Records |
|  |  |

We typically record the air pressure levels over time using a device (a microphone) that converts air pressure to voltage or current, and play it back using a device (a speaker) that converts voltage or current to air pressure.

# Representing Sound in Digital Form

The most common format for representing sound is called Pulse Code Modulation (PCM), which stores the sound pressure as a series of measurements (called samples) at equally-spaced intervals. These measurements can be played back to reproduce an approximation of the original signal. Note that it is an imperfect approximation – there is some amount of loss due to quantization.

The loss is dominated by two factors: the data type used to represent the samples, and the frequency at which the samples have been measured. The loss from the selection of data type is fairly straightforward: a 16-bit integer can store a value that is 256 times as precise as an 8-bit integer. The loss from frequency is a little more subtle. In general, a digital signal can accurately reproduce any analog signal up to ½ of its sampling rate – a value that is called the Nyquist Frequency. For example, if you record an audio signal at 48,000Hz (a common sampling rate), you can accurately reproduce any signal up to 24,000Hz. This means that any data above that frequency will be effectively lost.

The upper threshold for human hearing is around 20,000Hz[[2]](#footnote-2), so typical sampling rates for audio are in the 40,000Hz range (most common are 44,100Hz and 48,000Hz, but typical sample rates range from about 8,000Hz up to 192,000Hz or more).

# Audio Devices

Modern audio devices can be built into the motherboard, socketed into an expansion slot, or plugged into an extension port. Under the hood, an audio device contains a digital-to-analog converter (DAC) chip, which takes samples from the system and converts them to electrical signals that drive a speaker. They are typically abstracted by the operating system through an API such as WASAPI (Windows), CoreAudio (macOS and iOS), or ALSA (Linux).

The audio device contains a small double-buffer of memory, which it is constantly consuming. When a buffer is consumed, it becomes available to be written, and the system has only a limited time to write all of the audio data to the buffer before the audio device begins playing it out. The duration that is available is controlled by the length of the buffer and the sample rate and data format that the buffer is being played back at. Here are a few examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Buffer Length | Sample Rate | Data Format | Number of Channels | Length |
| 512 bytes | 22,050 | float | 4 | 1.45ms |
| 512 bytes | 44,100 | float | 2 | 1.45ms |
| 1024 bytes | 48,000 | short | 1 | 10.67ms |
| 2400 bytes | 96,000 | float | 6 | 1.04ms |
| 4096 | 48,000 | float | 2 | 10.67ms |
|  |  |  |  |  |

The general formula for the length of time that a buffer takes is:

While it is possible to fill the buffers from the main thread, it is extremely uncommon to do so, because the audio buffers are so short that they typically need realtime responsiveness. As such, the audio is typically driven in a dedicated thread.

# Data Formats and Channels

There are three common data types that you see in audio devices: signed 8-bit, signed 16-bit, and floating point. Signed 24-bit and 32-bit are also around, but they are less common[[3]](#footnote-3). For floating point, the values typically range from -1.0f to +1.0f. While these are the most common hardware formats, there may exist esoteric hardware that expects data in a different format.

Channels refer to the number of physical speakers that are connected to the device. A stereo output has two channels (Left and Right), and a 5.1 surround sound system has six channels (Left, Right, Center, Surround Left, Surround Right, and Subwoofer). Within a given buffer that you’re writing to the device, there are two ways that data can be organized: interleaved or isolated.

Interleaved audio data has the channel data interleaved one after the other[[4]](#footnote-4). For a stereo buffer, you would write the sample data thus: LRLRLRLR[[5]](#footnote-5). For a 5.1 buffer, you might write out something like: LRCLsRsSLRCLsRsSLRCLsRsS. Contrariwise, isolated buffers split the channels out to different portions of the buffer. For stereo, you would write LLLRRR, and for 5.1 you would write LLLRRRCCCLsLsLsRsRsRsSSS. The actual format that the device expects is defined by the device driver.

1. A decibel is a unit of measure that expresses the ratio of one value of a physical property to another on a logarithmic scale. In this example, a 100dB dynamic range means that the quietest sound a person can hear is 100,000 times quieter than the loudest sound. (In fact, the “loudest sound” refers to the pain threshold, not an actual loudest sound that can be heard.) [↑](#footnote-ref-1)
2. I myself can hear up to about 17,000Hz. [↑](#footnote-ref-2)
3. 24-bit audio does feature more heavily in wave files. More on that later. [↑](#footnote-ref-3)
4. In all of these examples, L is Left, R is Right, C is Center, Ls is Left Surround, Rs is Right Surround, and S is Subwoofer. [↑](#footnote-ref-4)
5. RLRLRL is also possible, but it is uncommon. [↑](#footnote-ref-5)