Fundamentals of USB-Audio

IN THIS DOCUMENT

USB, the Universal Serial Bus, has been around for decades and is a heavily used standard in the world of personal computers. Memory sticks, external drives, mice, and web cameras are all interfaced over USB. In this article we will look into USB-Audio: a standard for digital audio used in PCs, smart phones and tablets to interface with audio peripherals such as speakers, microphones, or mixing desks. In this article we set out to show how USB-Audio works, what to watch out for, and how to use USB-Audio for high fidelity multi channel input and output.

1 USB basics

USB is a protocol where the PC, the *USB-host*, initiates a *transfer*, and the *device* (for example a USB speaker) responds. Each transfer is addressed to a specific device, and to a specific *endpoint* on the device. *IN-transfers* send data to the PC. When the host initiates an IN-transfer the device has to respond with data for the host. OUT-transfers send data to the device. When the host performs an *OUT-transfer* it sends a packet of data that the device must capture. In the world of USB-Audio, IN and OUT transfers may be used to transport audio samples: an OUT-transfer to send audio data from a PC to a speaker, whereas an IN-transfer is used to send audio data from a microphone to the PC.

There are four sorts of IN and OUT-transfers in USB: *Bulk*, *Isochronous*, *Interrupt*, and *Control* transfers.

A bulk transfer is used to *reliably* transfer data between host and device. All USB transfers carry a CRC (checksum) that indicates whether an error has occurred. On a bulk transfer, the receiver of the data has to verify the CRC. If the CRC is correct the transfer is acknowledged, and the data is assumed to have been transferred error-free. If the CRC is not correct, the transfer is not acknowledged and will be retried. If the device is not ready to accept data it can send a negative-acknowledgment, *NAK*, which will cause the host to retry the transfer. Bulk transfers are not considered time criticial, and are scheduled around the time critical transfers discussed below.

Isochronous transfers are used to transfer data in *real-time* between host and device. When an isochronous endpoint is set up by the host, the host allocates

Publication Date: 2015/3/17 Document Number: XM007723A

a specific amount of bandwidth to the isochronous endpoint, and it regularly performs an IN- or OUT-transfer on that endpoint. For example, the host may OUT 1 KByte of data every 125 us to the device. Since a fixed and limited amount of bandwidth has been allocated, there is no time to resend data if anything goes wrong. The data has a CRC as normal, but if the receiving side detects an error there is no resend mechanism.

Interrupt transfers are used by the host to regularly poll the device to find out whether something worthwhile has happened. For example, a host may poll an audio device to check whether the MUTE button has been pressed. The name *Interrupt* transfer is slightly confusing, since they do not interrupt anything. However, regular polling of data gives the same sort of functionality that an host-interrupt would provide.

Control transfers are very much like bulk transfers. Control transfers are acknowledged, can be NAKed, and are delivered in a non-real-time fashion. Control transfers are used for operations that are outside normal data flow, such as querying the device capabilities, or endpoint status. An explanation on how device capabilities are described is outside the scope of this article, and we just state that there are predefined *classes* such as 'USB Audio Class' or 'USB Mass Storage Class' that enable cross platform interoperability.

All transfers are made in USB *frames*. High Speed USB frames span 125 us (Full Speed USB are 1 ms) and are marked by the host sending a Start-Of-Frame (SOF) message. Isochronous and Interrupt transfers are transmitted at most once a frame.

2 USB-Audio

USB-Audio uses isochronous, interrupt and control transfers. All audio data is transferred over isochronous transfers; interrupt transfers are used to relay information regarding the availability of audio clocks; control transfers are used used to set volume, request sample rates, etc. These are shown in Figure 1.

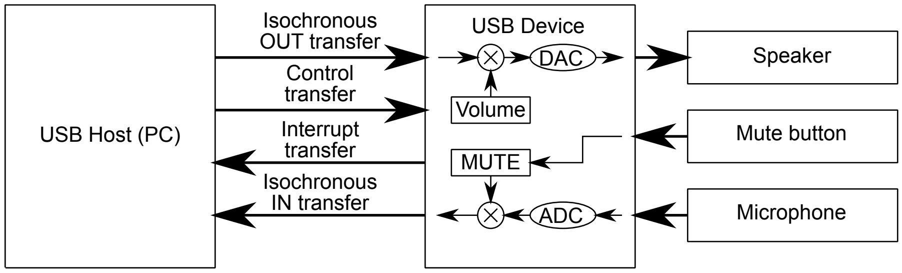


Figure 1: Transfers between a host and a USB device: Isochronous IN and OUT for audio-data, Control for setting parameters, and Interrupt for status monitoring.

The data requirements of a USB-Audio system depends on the number of *channels*, the number of *bits* to represent each sample, and the *sample rate*. Typical channel counts are 2 (stereo), 6 (5.1) or much higher for studio and DJ use. Typically sample size is 24 bits, although 16 bits is available for legacy audio, and 32 bits for high quality audio. Typical sample rates are 44.1, 48, 96, and 192 kHz. The latter is used for high quality audio.

Suppose that we design a stereo audio speaker with a 96 kHz sample rate and 24-bit samples. In order to simplify data marshalling on host and device, 24-bit values are typically padded with a zero byte, so the total data throughput is 96,000 x 2 channels x 4 bytes = 768,000 bytes per second. The isochronous endpoints run at a rate of one transfer per 125 us; or 8,000 transfers per second. Dividing the required byte rate over the frame rate gives us the number of bytes for each isochronous transfer: 768,000/8,000 = 96 bytes per transfer.

When using CD rates, such as 44,100 Hz, the transfer rate works out as 44.1 transfers per second. In USB-Audio each transfer always carries a whole number of samples; alternating transfers carry 48 and 40 bytes (6 and 5 stereo samples), so that the average rate works out as 44.1 bytes per transfer.

A single isochronous transfer can carry 1024 bytes, and can carry at most 256 samples (at 24/32 bits). This means that a single isochronous endpoint can transfer 42 channels at 48 kHz, or 10 channels at 192 kHz (assuming that High Speed USB is used - Full Speed USB cannot carry more than a single stereo IN and OUT pair at 48 kHz).

When transmitting digital audio, latency is introduced. In the case of High Speed USB this latency is 250 us. A packet of data is transferred once in every 125 us window, but given that it may be sent anytime in this window a 250 us buffer is required. On top of this 250 us delay, extra delay may be incurred in the O/S driver, and in the CODEC. Note that Full Speed USB has a much higher intrinsic latency of 2 ms, as data is only sent once in every 1 ms window.

3 What's a second between friends?

The big issue in digital audio is to agree on a common notion of time. Above we have defined USB frames to be transferred 8,000 times per second, and set the speakers to play a sample 96,000 times per second. This will only work if the speaker and the host agree on the length of a second. USB-Audio offers three modes that ensure that the host and the speaker agree on timings:

- In *synchronous mode*, the length of a second is defined by the host device. That is, the host will send data at a rate, and the device has to exactly match that rate.
- In *asynchronous mode* it is the other way around, the device sets the definition of a second, and the host has to match the device.
- In *adaptive mode* the data flow determines the clock.

Adaptive and synchronous mode are not ideal because PCs are notoriously bad at keeping a stable clock, and there are often other audio sources involved, such as an external digital deck. Asynchronous mode enables external clock sources to be used as the master, or a low-jitter clock in the device. Typically, either relies on a crystal based PLL, as shown in Figure 2.

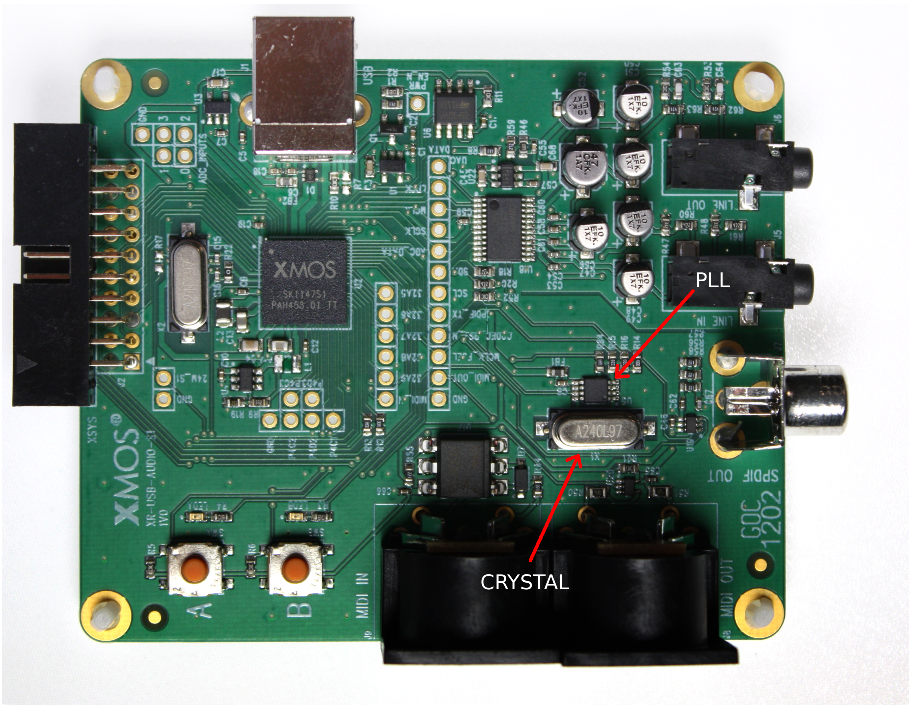


Figure 2: A USB-audio board, with a crystal for stable audio frequencies, and a low-jitter PLL to generate any frequency required.

Hence there are at least two separate clocks in the system, the USB clock with a host driven frequency of 8,000 transfers per second, and a sample clock with an externally driven sample rate of, for example, 96,000 Hz.

These clocks will have subtly different frequencies, and the difference will vary slightly over time. Hence the average number of audio samples per frames will be slightly more or less than the expected rate. For example, in the case of our 96,000 Hz sample rate, the average number of samples may be 12.001. In order to ensure that the host sends the right amount of data, and not too much or too little, the host requests the current sample rate over an interrupt endpoint. Every few milliseconds the average sample rate over the last period is reported back as a 16.16 bit fixed point number. If the last period averaged out as 12.001 frames, then the value 0x000C0041 will be reported (65536 * 12.001).

Given this average rate, the host can work out when to send an extra sample in a transfer; in this example 8 transfers each second will carry one extra sample. In addition to this, the host can use this value to synchronise itself with the audio device. This enables host applications such as a DVD player to keep the video in sync with the audio. If it didn't, the audio would slowly run ahead of the video, and after two hours the audio would be a second out.

In order to keep a short feedback loop, the trick is to not buffer audio packets and feedback packets unnecessarily. Any additional buffering creates latency in the reporting, and this latency makes it more difficult to keep a smooth flow of traffic. This means that the low-level USB stack and the USB-Audio stack should be tightly integrated, without buffering in between. Although this is hard to achieve on an application processor, this is quite easy to achieve if the software is implemented on an embedded processor that has a predictable execution time.

4 Multiple clock sources

The above scheme considers just two clock sources - either the USB device provides the clock, or the host provides the clock. In more complex devices such as mixer desks there may be other devices that provide the sample rate, for example through a digital interface such as ADAT or SPDIF, or through a BNC connector that carries the word clock. For systems like this, the USB-Audio standard allows designers to put a *clock selector* in the device.

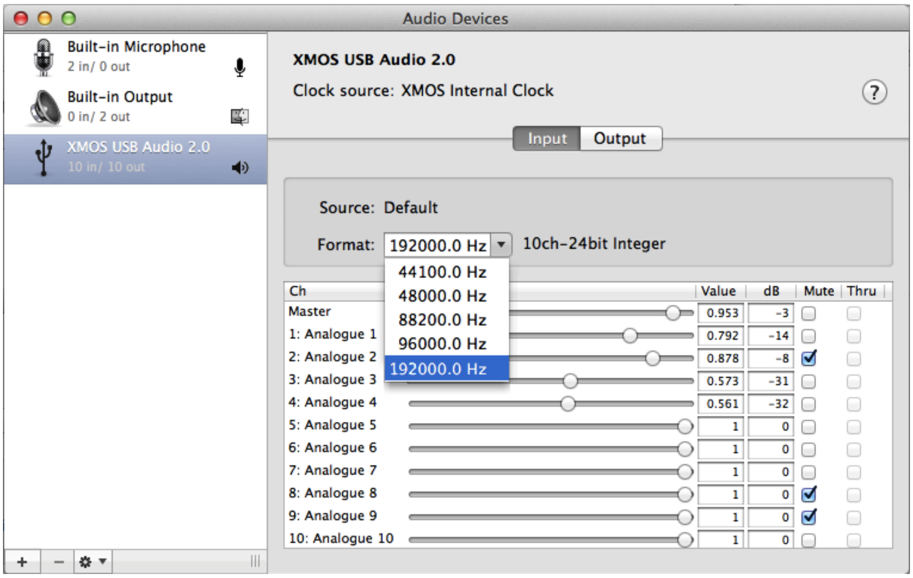
The clock selector states which clock is to be used as the sample rate. The clock selector has multiple input clocks (e.g., the incoming clock on an S/PDIF connection; a local crystal, and the incoming clock on an ADAT connection) and with a control transfer the user selects which clock to use as input, for example the incoming clock of the S/PDIF connection.

5 Compliance and native support

Once a device is USB-Audio Class compliant, it will integrate neatly into the operating system. Figure 3 shows a screenshot of the controls of a USB-Audio device plugged into Mac OS/X. It shows that the clock selection, sample rate selection, channel volume control, and mute control are all controllable just like for any other audio device.

Compliance to the standard makes the device interoperable. O/S vendors can supply a single USB-Audio driver that drives a multitude of devices, with a multitude of capabilities.

Indeed, the same USB-Audio implementation can be parameterised to implement a different number of channels, and the same driver can be used to interface to the device.



XMOS USB Audio 2.0

Clock source: XMOS Internal Clock

Source: Default

Format: 192000.0 Hz 10ch-24bit Integer

Ch	Value	dB	Mute	Thru
Master	0.953	-3	☐	☐
1: Analogue 1	0.792	-14	☐	☐
2: Analogue 2	0.878	-8	✓	☐
3: Analogue 3	0.573	-31	☐	☐
4: Analogue 4	0.561	-32	☐	☐
5: Analogue 5	1	0	☐	☐
6: Analogue 6	1	0	☐	☐
7: Analogue 7	1	0	☐	☐
8: Analogue 8	1	0	✓	☐
9: Analogue 9	1	0	✓	☐
10: Analogue 10	1	0	☐	☐

Figure 3: An interoperable device appears in a standard O/S dialog (Mac OS/X in this example), and the O/S can set volume, sampling rate, etc.

6 Summary

USB-Audio Class 2.0 takes advantage of High Speed USB 2.0, enabling low latency transfer of audio between PC and a connected audio device. The high throughput of High Speed USB 2.0 can be utilised to deliver many audio channels, and with high audio quality. The USB-Audio Class standard caters for a wide range of devices, from complex mixing desks with many channels, multiple clock sources and complex controls, to surround sound systems, PC speakers and microphones.