Why do you need USB Audio Class 2?

Introduction

Since its launch nearly 20 years ago, Universal Serial Bus (USB) has become the de facto digital communications interface for connecting wired peripherals to consumer and professional electronic devices. Integrated into products such as PCs, laptops, tablets, phones, games consoles and televisions, USB is truly ubiquitous and provides a simple-to-use, plug-and-play user experience.

A very successful application space for USB continues to be audio. From basic USB speakers to high-resolution audiophile headphone amplifiers, and from a USB microphone to a multi-channel mixing and recording desk, USB connectivity has enabled a rich and diverse range of audio products. Behind this success is the USB Audio Class (UAC) standard, which defines an extensive and robust set of audio control and streaming features to ensure inter-operability of USB audio devices with any USB host.

This article discusses the key features of UAC, explains the differences between revision 1 and 2 of the standard, and outlines the benefits of choosing USB Audio Class 2 (UAC2) for your next audio product. It also shows how UAC2 continues to be important in the ever evolving world of audio formats and content delivery methods.

Aside from the USB and UAC specifications themselves, there are already many excellent articles explaining USB and UAC. Rather than repeat the content here, we recommend you read the article: "Fundamentals of USB Audio" [Henk Muller, XMOS Ltd][1] for an introduction to USB and UAC terminology.

USB and UAC background

First, let's look at how the USB standard has evolved and which features are relevant to USB audio. Since the USB specification was first released in January 1996 there have been several updates to enhance the standard and meet the growing needs of consumers. With each major revision of the standard the bus speed has increased significantly from the original 12Mbit/s bandwidth up to 10Gbit/s today. Table 1 below shows the speed characteristics for each revision of the USB specification.

[1] http://www.edn.com/design/consumer/4376143/Fundamentals-of-USB-Audio

			USB standard version			
Name	**Raw bus speed**	**Service interval**	**1.0, 1.1**	**2.0**	**3.0**	**3.1**
Low Speed	1.5Mbit/s	1ms frame	✓	✓	✓	✓
Full Speed	12Mbit/s		✓	✓	✓	✓
High Speed	480Mbit/s	125µs microframe	-	✓	✓	✓
SuperSpeed	5GBit/s		-	-	✓	✓
SuperSpeed+	10Gbit/s		-	-	-	✓

Table 1: USB versions

It is important to note that a USB host compliant to any given version of the USB standard must maintain backward compatibility to all previous versions of the USB standard. The host is required to detect the capabilities of an attached device and configure itself to operate at the correct speed. For example, a USB 3.0 host must provide full backwards-compatibility to control a High Speed, Full Speed, or Low Speed device.

Now let's take a moment to understand the evolution of UAC and its relationship with the USB standard. UAC was developed to address the very specific needs of audio applications and version 1 of the specification was released shortly before the release of USB1.1. Consequently a UAC1 device connects as a USB 1.x device and only operates at Full Speed. The UAC2 specification was released several years after the USB 2.0 specification and takes advantage of the increased bus speed. A UAC2 device can connect as a USB 2.0 device and can operate at either Full Speed or High Speed. However the UAC2 specification states that, "the changes introduced in UAC2 are not generally backwards compatible to UAC1." Therefore a USB audio device must choose to declare itself as either a UAC1 device or a UAC2 device at enumeration.

Table 2 summarizes the interoperability between different classes of hosts and UAC devices.

USB Host	UAC1 device	UAC2 device
1.x	Full speed	Full speed
2.0	Full speed	High speed / Full speed
3.x	Full speed	High speed / Full speed

Table 2: UAC interoperability for USB bus speeds

What about USB 3 audio?

First we need to interpret the question.

As already discussed, when a UAC1 or UAC2 device is connected to a USB 3.x host, then the host will fall back to operate at the appropriate speed (either full speed or high speed). So if the question means "Can USB 3 support audio?" the answer is yes - a UAC1 or UAC2 device is fully compatible with a USB 3.x host.

But if the question means "What about USB Audio Class 3?" then there is no such specification. And, at the time of writing, no work has even begun on a version 3 of the UAC specification. As this article will show, UAC2 continues to meet the requirements for USB audio applications.

Analogue to USB digital

Before we dig into the details of USB audio, let's consider how audio data is represented in the digital world.

CDs were the first digital audio format to be deployed to the mass-market, with their audio data stored using Linear Pulse Code Modulation (PCM) encoding. PCM is a raw, uncompressed format where each audio channel is sampled at a fixed rate and bit-depth; 16bit stereo at 44.1kHz in the case of a CD.

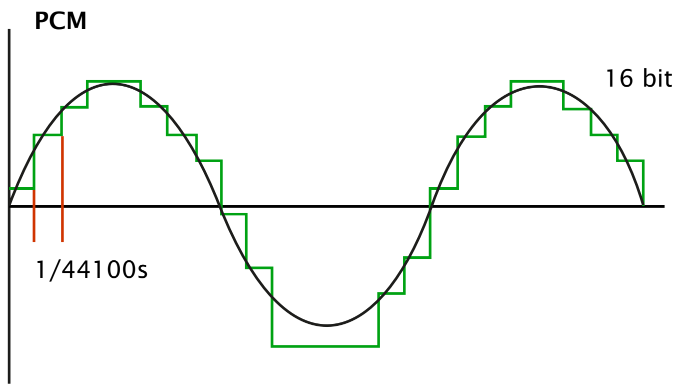

Figure 1: PCM encoding

PCM is a direct digital representation of the sampled analogue audio signal and produces a constant bitrate data stream (see figure 1). No processing is required to interpret PCM data which provides low-latency access and makes it relatively easy to manipulate the audio signal in hardware and/or software. PCM encoding is ideally suited to the transfer of audio over USB and is the default audio format in both UAC1 and UAC2.

Both UAC1 and UAC2 define the use of isochronous transfers for audio data. Isochronous transfers guarantee bandwidth availability on the USB link and so ensure delivery of the audio data with a bounded latency. But how much data, and with what latency? This is one of the key areas where UAC1 and UAC2 differ.

Bandwidth and channel counts

Let's start by looking at the data bandwidth offered by UAC1 and UAC2.

As mentioned earlier, UAC uses isochronous transfers to guarantee bandwidth availability and every UAC device must therefore implement Isochronous Endpoints. The amount of data that can be transferred by any given Endpoint is a function of the data payload or packet size and the USB frame period (or service interval).

Table 3 below summarises the bandwidth available for UAC1 and UAC2. For UAC1, a Full Speed isochronous endpoint can transfer a single packet containing up to 1023bytes per 1ms frame, which equates to 8.2Mbit/s. A UAC2 High Speed isochronous endpoint can transfer a single packet of up to 1024bytes per 125µs microframe, which equates to 65.5Mbit/s. USB 2.0 also defines a high bandwidth isochronous endpoint that can transfer up to 3 packets per 125µs microframe (192 Mbit/s). Note that high bandwidth transfers are only supported on High Speed devices.

What is an Endpoint?

USB uses "Endpoints" to facilitate the transfer of data between hosts and devices. The USB specification specifies an Endpoint as "A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device." There are four basic types of endpoints defined for different types of data transfer: Control, Bulk, Interrupt and Isochronous.

UAC	USB	Isochronous Endpoint
UAC1	Full Speed	≤8.2Mbit/s
UAC2	High Speed	Standard: ≤65.5Mbit/s High-BW: ≤196.6Mbit/s

Table 3: Maximum data transfer rates for UAC isochronous endpoint

While the maximum data rate of an interface is interesting, what really matters in audio applications is the number of audio channels that an interface can support. Since UAC uses PCM encoding the translation from data rate to the theoretical maximum number of audio channels a UAC endpoint can carry is straightforward. The bandwidth required for a single PCM channel is simply the sample rate multiplied by the bit-depth; table 4 below details the number of channels supported by each class of UAC endpoint. Note that since an isochronous endpoint provides unidirectional data flow the following channel count values are per audio direction.

Sample rate (kHz)	44.1			48			88.2			96			176.4			192			352.8			384		
Bit-depth	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**	**16**	**24**	**32**
BW per ch (Mbit/s)	0.71	1.06	1.41	0.77	1.15	1.54	1.41	2.12	2.82	1.54	2.30	3.01	2.82	4.23	5.64	3.07	4.61	6.14	5.64	8.47	11.3	6.14	9.22	12.3
UAC 1	*11*	*7*	*5*	*10*	*7*	*5*	*5*	*3*	*2*	*5*	*3*	*2*	*2*	*1*	*1*	*2*	*1*	*1*	*1*	*-*	*-*	*1*	*-*	*-*
UAC 2	92	61	46	85	56	42	46	30	23	42	28	21	23	15	11	21	14	10	11	7	5	10	7	5
UAC2 high BW	*278*	*185*	*139*	*256*	*170*	*128*	*139*	*92*	*69*	*128*	*85*	*64*	*69*	*46*	*34*	*64*	*42*	*32*	*34*	*23*	*17*	*32*	*21*	*15*

Table 4: Maximum channel counts for UAC1 and UAC2

In most real-world audio applications channel count requirements are even numbers, with a minimum of stereo. Also, where 24bit audio is used, for ease of compute on a 32-bit machine, this is usually processed in a zero-padded 32bit word. This allows us to construct the following comparison, as shown in table 5.

		Channel count (per isochronous endpoint)																										
		UAC1									**UAC2 (standard endpoint)**									**UAC2 (high BW endpoint)**								
SR (kHz)	**Bit-depth**	**2**	**4**	**6**	**8**	**16**	**32**	**48**	**64**	**128**	**2**	**4**	**6**	**8**	**16**	**32**	**48**	**64**	**128**	**2**	**4**	**6**	**8**	**16**	**32**	**48**	**64**	**128**
44.1	**16**	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	24/32	✓	✓								✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓
48	**16**	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	24/32	✓	✓								✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	
88.2	**16**	✓	✓								✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓
	24/32	✓									✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	
96	**16**	✓	✓								✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	
	24/32	✓									✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓		
176.4	**16**	✓									✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	
	24/32										✓	✓	✓	✓						✓	✓	✓	✓	✓	✓			
192	**16**	✓									✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓		
	24/32										✓	✓	✓	✓						✓	✓	✓	✓	✓				
352.8	**16**										✓	✓	✓	✓						✓	✓	✓	✓	✓	✓			
	24/32										✓	✓								✓	✓	✓	✓	✓				
384	**16**										✓	✓	✓	✓						✓	✓	✓	✓	✓				
	24/32										✓	✓								✓	✓	✓	✓					

Table 5: UAC1 and UAC2 channel count comparison

This demonstrates the higher data capabilities of UAC2, which enables high-resolution multi-channel audio applications.

Latency

A standard isochronous endpoint assembles the streaming data into packets, and transmits one packet every (micro)frame. Since this data packet may occur anywhere within a given (micro)frame, to ensure a continuous stream of data buffers are required at both the sending and receiving endpoints. A minimum of a 2 (micro)frame buffer needs to be used at both the sending and receiving endpoints to account for worst case packet jitter. This introduces latency. Table 6 below defines the minimum round-trip latency for UAC1 and UAC2. In applications where streaming latency is important, UAC2 offers up to an 8x reduction over UAC1.

UAC	Streaming latency
UAC1	4ms
UAC2	500µs

Table 6: Minimum latency per endpoint/direction

Note that in a complete USB audio system there will be additional latencies due to the system hardware (interrupts, DMA, digital/analogue conversion, etc.) and host software (interrupts, drivers, audio stack, user application, buffering etc.).

Timing is everything

The importance of a clean stable clock in digital audio is well documented, and transferring audio over USB is no exception. UAC offers three clocking mechanisms:

- Synchronous: The audio clock is derived from the USB start-of-frame signal and so is synchronous to the USB host.
- Asynchronous: The audio clock is derived from an independent clock source, typically a crystal oscillator in the USB audio device, or an external clock supplied to the USB audio device.
- Adaptive: The audio clock is derived from the flow of audio data itself.

Each clocking method has pros and cons and best-fit applications. For example, asynchronous is particularly favoured for audiophile playback systems where a high stability clock source is implemented in the device, or audio is synchronised to an external digital stream such as S/PDIF. Whereas synchronous enables audio content from multiple USB devices to be more easily aggregated by a host, without the need for complex sample-rate conversion and so is often used in professional mixing systems.

UAC1 only allows for a single audio clock. UAC2 introduces the concept of ‘clock domains’ (one or more sample clocks synchronized to a common master clock) and the ability to support one or more clock domains, each derived from a different ‘clock entity’.

Both UAC1 and UAC2 offer the same three clocking schemes. However, UAC2 offers 8x the timing resolution in synchronous and adaptive modes, and allows complex multi-clocking domains to be configured and controlled.

Staying in control

So far we have presented UAC purely as a standardized transport mechanism to stream audio data between a host and a connected peripheral. But UAC provides much more.

A typical USB audio device will implement some form of audio control and processing functionality. This might be something as simple as a mute or volume control, or a complex channel routing and mixing matrix with sample rate conversion and DSP effects.

UAC provides standardized methods to describe and present such features to a host, allowing a common host driver and API to configure and control the audio functionality of any USB audio device. UAC2 extends the control capabilities of UAC1 with, amongst other things: richer audio effects functionality, physical and logical channel clusters, more accurate device descriptions and the ability to map physical user controls directly to audio functions without the need to implement a separate USB Human Interface Class (HID).

The host with the most

For UAC to actually work in the real-world requires compliant implementations of UAC in both the host and the device. Fortunately many common operating systems (OS) include native UAC support.

OS	UAC1	UAC2
Mac OS X	✓	✓
Windows *	✓	-
Linux (e.g. Ubuntu)	✓	✓
iOS	✓	✓
Android **	✓ (since 5.0)	-

Table 7: Operating System native support for UAC

* Windows only supports UAC1 natively. However, independent drivers are available that add UAC2 support to Windows.
** Android only supports UAC1 natively. However, several manufacturers have added UAC2 support to their implementations of Android.

Market relevance

So far we've shown how UAC provides robust audio connectivity and an extensive set of standardized audio controls with support across a wide range of platforms. But how is this useful and why should we care?

Digital audio content is created and consumed. Creation implies recording, consumption implies playback, and sometimes they are combined, for example, a live performance or a Karaoke system. UAC enables audio connectivity for all of these applications, even though the requirements are quite different.

Playback

Since the launch of CDs over 30 years ago, the consumer market has embraced the convenience and reliability of digital audio. However, storing PCM data directly results in large files – for example, a 60 minute CD equates to a 650Mbyte audio .wav file. The growth of computer based audio storage and the explosion in portable music players would not have been possible without another innovation; audio compression. MP3 (or MPEG-2 Audio Layer 3 to give it its proper name) dominates this space and enables file sizes to be reduced by up to 10x.

The shift in how we source our digital music - firstly by ripping CDs, then to digital downloads and more recently towards online streaming - has been enabled by MP3, and other compression formats.

Here it is important to note that whilst audio data may exist on the host in a compressed format, both UAC1 and UAC2 require that audio data streamed across the USB connection is PCM encoded. Any audio compression/decompression is therefore always carried out by the host.

As Table 5 shows, playback of stereo, CD quality or decompressed MP3 audio is perfectly feasible using UAC1 or UAC2.

Ever increasing data storage capabilities coupled with always-on high-bandwidth internet connections have enabled another trend; high-resolution audio. Some studios are also releasing content as recorded at higher sampling rates (typically 96kHz or 192kHz) and greater bit depth (typically 24bits); distributing as either raw uncompressed data, or compressed using a lossless codec such as FLAC.

Many high-resolution audio playback devices are implemented as add-on DACs or headphone amplifiers, which require a connection to the audio content source. This connection needs to provide robust, bit-accurate audio transfer with enough bandwidth to accommodate the high-resolution audio data-rate. And wouldn't it be convenient if this connection could also provide power and control? UAC provides the ideal connectivity solution, and more specifically the use of UAC2 becomes mandatory for high-resolution audio.

DSD and DoP

Earlier we stated that UAC requires audio data be PCM encoded. Like any rule, there is always an exception. UAC2 introduced a special 'Raw Data' transfer mode which allows for non-PCM encoded data to be transferred. Many high-resolution audio playback devices use this mode to stream Direct Stream Digital (DSD) encoded audio.

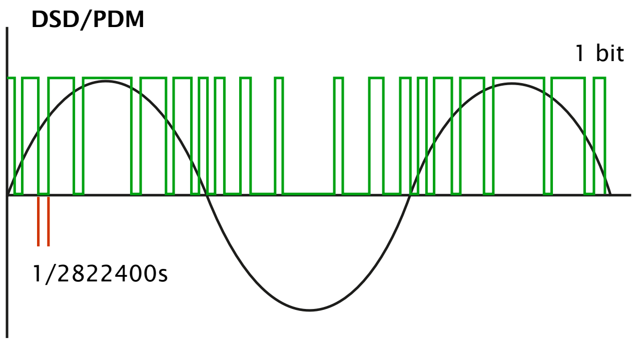

Figure 2: DSD encoding

DSD encoding is used by Super Audio CD (SACD) and employs Pulse Density Modulation (PDM) encoding, as shown in Figure 2. The equivalent to CD quality in DSD encoding is a 2.28224MHz single-bit stream. Since this bit-rate is 64 times the equivalent CD sample-rate, 2.28MHz DSD is also referred to as DSD64. To increase the resolution of DSD one simply increases the sample-rate; typically by a factor of 2. For example, DSD128 is 5.6MHz and DSD256 is 11.3MHz.

Since streaming DSD content uses UAC2 'Raw Data' mode, and the format of the data in this mode is undefined in the UAC2 specification, both the system host and the audio device have to agree on what the raw data represents. As well as specific support in the audio device, implementing 'native DSD' therefore requires a custom UAC2 driver on the system host.

For systems where the host UAC2 driver does not offer native DSD support, a work-around solution exists in the form of DSD-over-PCM (DoP), where the DSD encoded data is encapsulated in a normal PCM stream. A DoP audio source adds special markers to the data, which when received by a DoP audio sink causes the sink to extract the embedded DSD content.

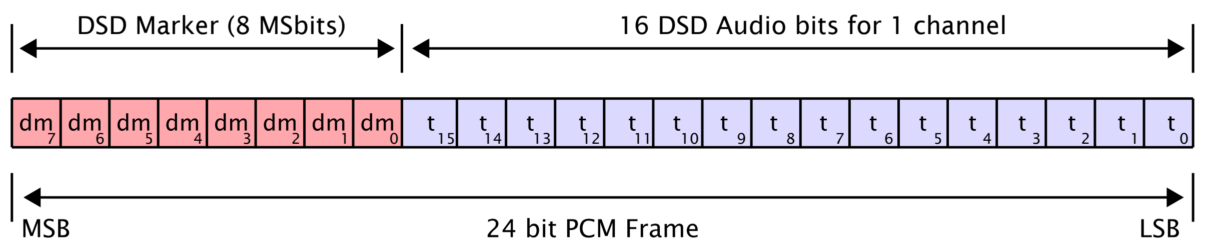
DSD Marker (8 MSbits)
16 DSD Audio bits for 1 channel
dm7 dm6 dm5 dm4 dm3 dm2 dm1 dm0
t15 t14 t13 t12 t11 t10 t9 t8 t7 t6 t5 t4 t3 t2 t1 t0
MSB
24 bit PCM Frame
LSB

Figure 3: DoP encapsulation

Since 16bits of DSD audio in encapsulated in to a 24bit DoP frame and 24bit data is typically transferred as a 32bit word, DoP encoding requires twice the bandwidth as native DSD. Figure 3 shows how the DSD data is encapsulated in a PCM frame.

Note that when streaming DSD in Raw Data mode, standard UAC2 controls cannot be used, as they only understand PCM encoded data.

Similarly, when streaming DSD as DoP, any audio processing or volume controls must be disabled, or be set to pass audio data through transparently, so as not to break the DoP encoding.

UAC provides an ideal audio playback connectivity solution and, by choosing UAC2, USB audio devices can support high-resolution audio, streamed as either PCM or DSD data.

Recording and live performance

The processing capability available in desktops, laptops and even tablets has spawned a rich variety of powerful Digital Audio Workstation (DAW) software. But how do you get the audio into and out of the DAW? Again UAC provides an ideal audio connectivity solution. But compared to our previous example of playback only, the requirements are quite different.

Firstly, channel count. A simple solo artist configuration might only require 2 input channels. For a small band 6 inputs could be sufficient. In larger applications, channel counts of 16 or 32 are not unusual. Recording at 96kHz, 24bits is commonplace. Multiple output channels are also required: as monitoring feedback to the performers and sound engineers, and as the main audience feed at a live performance. Together these factors increase the required data bandwidth.

Secondly, latency. In a DAW-based system, delays accumulate along the audio path. For example: from microphone, ADC, USB, DAW, USB, DAC, amplifier, to speaker. Or for a performer, to in-ear monitors. Minimizing this delay is critical when hearing one's own voice, where delays above 10-15ms become detectible. Since system latency is a sum of its parts, and USB streaming latency occurs twice in the chain (into the DAW and back out again), keeping USB latency to a minimum is important.

As Table 5 and Table 6 demonstrate, only UAC2 provides the capability to handle higher channel counts and deliver the low-latency streaming demanded by recording and live audio applications.

UAC1 or UAC2?

So, as you embark on the design of your next USB audio device, should you choose UAC1 or UAC2? Table 8 below summarizes what we've learnt about UAC 1 and UAC2.

Feature		UAC1		UAC2
Channel count		Ok for low channel counts	✓	Enables multi-channel bi-directional audio
Latency		Ok for playback only	✓	Required for low latency live monitoring applications
Audio resolution		Limited to maximum of 96kHz, 24bit stereo playback	✓	Enables high resolution bi-directional audio
Power consumption	✓	Full speed USB consumes less power		High speed USB consumes more power
Host support	✓	Native support in all OSs	✓	Support available in all OSs

Table 8: UAC1 or UAC2?

As always, one has a choice. For basic applications with low channel count(s) at up to CD quality audio, then UAC1 continues to be sufficient. But for an application that requires higher-resolution audio, or higher channel counts or where audio latency is important then UAC2 really is the only choice.

In conclusion

This article has explored the features of the USB Audio Class, demonstrating how the robust audio streaming connectivity and standardized audio control capabilities meet the requirements of real-world audio applications. We have seen how UAC2 extends the capabilities of UAC1 and enables whole new categories of audio products; from high-resolution audio playback systems to live multi-channel audio recording workstations.

In the continually evolving world of digital audio formats, content creation techniques and online music delivery methods, UAC2 continues to be as relevant now, if not more so, than when it was introduced. UAC2 enables your design today, future-proofs for tomorrow and so should be the standard of choice for your next audio product.

Laurence Strong
Technical Marketing and Applications Specialist
uk.linkedin.com/in/laurencestrong