DoP open Standard

Method for transferring DSD Audio over PCM Frames
Version 1.1

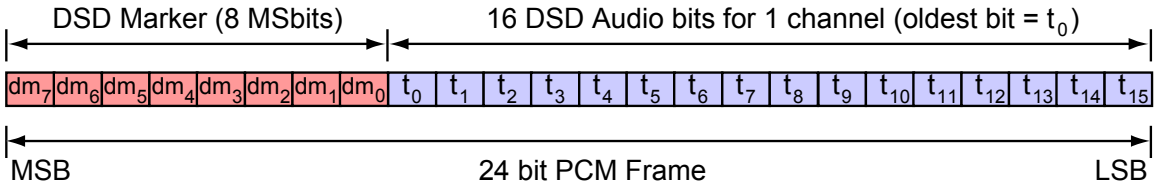
The USB Audio specification 2.0 defines multiple formats for audio of which standard PCM is only one. A general “raw data” format was also defined that can be used for any kind of data including audio, but unfortunately, no specific format was defined for DSD and with the ongoing proliferation of USB converters in the current market it appears that the opportunity for the official USB specification to adopt a single common method of transferring DSD audio via USB is slowly disappearing. This is an attempt of uniting as many manufacturers as possible and jointly defining a method for transferring DSD via USB. While this method is mainly targeted for USB links it is general enough to be applied to other PCM based links such as Firewire, AES/EBU, S/PDIF etc.

1. Motivation

The manufactuers developing audio playback software want to minimize the number of formats they need to support for a USB link. Ideally there is only a single such format. Likewise hardware manufacturers want to make their hardware compatible with as many playback platforms as possible. That, of course, only happens when all use the same format.
As mentioned above, USB Audio already supports a “raw data” format that could be used for DSD and that would create a clear separation to any audio data path containing PCM. However, the latest release of Apple’s operating system OS 10.7 incorporates a USB driver that only supports PCM. Furthermore, the central audio engine, CoreAudio, inside the OS only supports PCM as well, luckily with no limitation on sample rate. (earlier versions supported a mode that was compatible with raw data mode, but that is history). Since the architecture of Apple’s OS forces audio software developers to use CoreAudio for everything audio related there is basically only 1 format left for the Mac platform: PCM. Creating a separate path for DSD would involve a lot of surgery if it is even possible. So we have no choice, but to use a PCM path to transfer DSD audio by using special flags or headers that allow the receiving hardware to detect a format change and switch their decoder accordingly.
When using the Windows platform things are little easier: Windows by nature does not fully support USB Audio 2.0 and what it does support is limited to PCM only at a sample rate of 96kHz or less. There is no native driver support for higher resolution PCM and it is clear from the beginning that a custom driver needs to be created for this platform whether it is for regular PCM or DSD. Luckily a 3rd party software developer (Steinberg Audio) jumped in and created a driver (called ASIO) already many years ago that supports PCM and DSD with no limitation on sample rate or wordlength. It has become quite popular and many software vendors support this in the meantime. ASIO is not directly a hardware driver, but sits between the audio playback application and the hardware driver. Each hardware manufacturer still needs to develop a custom hardware driver for their own hardware, but ASIO then creates a common interface standard for all application software.

2. Solutions

As seen above the Windows platform basically offers a solution with the ASIO driver and the raw data format supported by USB Audio 2.0. Not as ideal as having a dedicated DSD path via USB, but this is safe and straightforward.
Since the Apple OS only allows a PCM path we have to find a way to put DSD audio data into PCM frames that then get sent via the native USB driver. DSD has a sample size of 1 bit and a sample rate of 2.8224MHz. In other words the data rate is 2.8224Mbits/sec. This is equivalent to 16 bit PCM at 176.4kHz. In order to clearly identify when this PCM stream contains DSD and when it contains PCM we will need additional bits. The PCM format with the next higher bit rate is 24 bits at a sample rate of 176.4kHz. This gives us 8 bits for this marker of identifier. It seems a bit overkill if all we need is 2 states (8 bits give us 256 states), but we will see that this extra overhead comes in handy. Here is how we can use the 24 bits in each sample and for each channel:



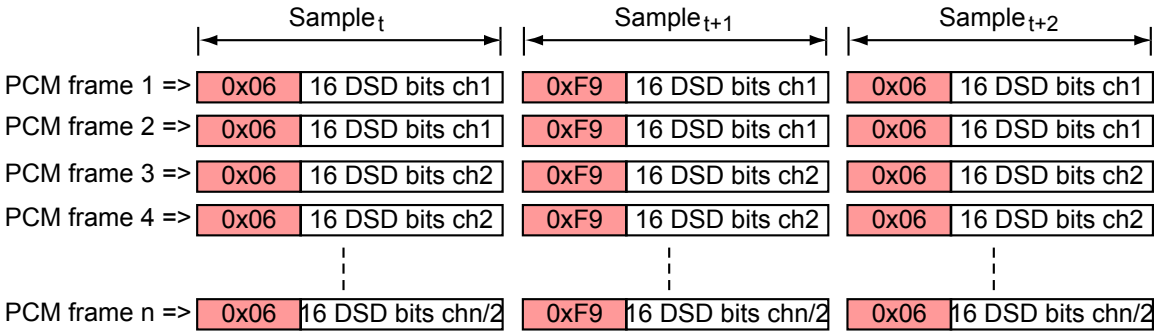
The 8 most significant bits are used for the DSD marker and alternate with each sample between 0x05 and 0xFA. Each channel within a sample contains the same marker. This has been chosen to minimize the click that might be experienced when the receiving hardware misinterpretes the data as PCM when it really is DSD. If this should happen it would create a tone around 88kHz and roughly -34db, nothing harmful and something that most D/A converters would suppress to some degree before it even reaches the loudspeaker. It should be pointed out that hardware manufacturers and software developers alike can easily use common safeguards to prevent such cases of erroneous format switching and that they may only be limited to times during development of hardware and software. It is their responsibility to prevent misinterpreted cases and to test their products thoroughly before release. Misinterpreation of PCM data as DSD may create less predictable clicks.

The remaining 16 lower bits are then used for the DSD data, first or oldest bit in slot t_0. The USB Audio specification assigns each PCM Frame to a specific channel (left, right etc.) and when used for DSD streaming each PCM Frame contains only DSD data corresponding to its assigned channel.

3. Solutions for double rate DSD (128FS) and beyond

Two solutions are possible depending on whether the used PCM transmission scheme is capable of supporting the PCM rate of 352.8kHz or not:

1. The solution described above for 64FS DSD can easily be extended for 128FS by simply raising the underlying PCM sample rate from 176.4kHz to 352.8kHz. All the marker bytes and bit ordering remain the same.
2. For those conduits that do not support 352.8kHz (such as AES/EBU) an alternative method can be used without raising the PCM sample rate:



A PCM channel pair (for instance L/R) is used to transmit 128FS DSD for a single DSD channel. The lower PCM frame number contains the 16 older DSD bits in the same order as in the 64FS case. The higher PCM frame number in the pair then contains the newer 16 DSD bits. A different marker byte is used to distinguish this method from the first one.

Solution 1 can easily be extended to support even higher DSD rates by raising the underlying PCM rate.

4. Recommended implementation

While there is certainly more than one way to implement this solution on the receiver side, the authors found the following implementation to work reliably:

- In order to switch from PCM to DSD mode the receiver has to detect 32 consecutive DSD marker bytes on all channels used.
- In order to switch from DSD to PCM mode the receiver has to detect at least 1 single missing DSD marker byte in at least 1 channel.

This introduces an additional latency of around 180usec. If the USB buffers are accessible for reading while the USB microframe is still being received then no additional delay is necessary.

If the transmission link used is USB it is recommended that for 128FS DSD the first solution described in section 3 is used, or else significant bandwidth is wasted for PCM transmission as the second solution always requires double the amount of channels than is necessary for PCM transmission.

In order to minimize false detection where DSD data would be interpreted as PCM or vice versa it is recommended that the host software verifies the DSD capability of the hardware before exchanging any data. This can be done in various ways and depends on physical link, driver and computer platform.

5. Industry Support

The following have contributed to this document and/or pledged their support for this format (alphabetical order):

Aesthetix	Jim White	Merging Technologies	Dominique Brulhart
Audirvana	Damien Plisson	MSB Technology	Larry S. Gullman
Benchmark Media Systems	John Siau	Mytek Digital	Michal Jurewicz
CEntrance	Michael Goodman	Playback Designs	Andreas Koch
CH-Precision	Thierry Heeb, Florian Cossy	Signalyst	Jussi Laako
		Sonic Solutions	Jon Reichbach
ChannelD	Rob Robinson	Wavelength Audio	Gordon Rankin
dCS	Andy McHarg, David J Steven	XMOS	Ali Dixon
		Vitus Audio	Martin Kristensen
JRiver, Inc.	Matt Ashland	Independent	Dustin Forman
Light Harmonic	Larry Ho		

6. Revision History

Ver	Date	Authors	Description
0.1	2012/01/20	Andreas Koch, Andy McHarg, Rob Robinson	Initial version
0.2	2012/02/07	Andreas Koch, Andy McHarg, Rob Robinson	Updated Industry support section, added recommended implementation
0.3	2012/02/09	Andreas Koch, Andy McHarg, Rob Robinson	Updated industry support section, clarified channel assignment, deleted recommendation about ASIO driver.
1.0	2012/02/17	Andreas Koch, Andy McHarg, Rob Robinson	Release version 1.0
1.0.1	2012/03/08	Andreas Koch, Andy McHarg, Gordon Rankin, Michal Jurewicz	Changed title, added section for 128FS solutions, updated industry support section
1.0.2	2012/03/19	Andreas Koch, Andy McHarg, Gordon Rankin, Michal Jurewicz	Updated section 4 and 5.
1.1	2012/03/30	Andreas Koch, Andy McHarg, Gordon Rankin, Michal Jurewicz	Release version 1.1