Application Note: AN00103

Enabling DSD256 in the USB Audio 2.0 Device Reference Design Software

The XMOS USB Audio 2.0 device software reference design software supports stereo DSD64 and DSD128 streaming output as standard. This application note describes how, through a few code modifications, support can be extended to DSD256.

Related documents

- USB Audio Software Design Guide[1]

Required tools and libraries

This application note assumes a baseline firmware of the XMOS USB Audio Reference design, version 6.6.1. Later versions of the reference design are likely applicable, however it is possible that code refactoring may alter the code changes required to implement DSD256.

A firmware build with DSD enabled is required as the starting point. A suitable firmware image is 2xoxxd (USB Audio 2.0, no input, output enabled, no midi/SPDIF and DSD enabled), which can be found in the app_usb_aud_xk_u8_2c application folder.

xTIMEcomposer Tools Suite version 13.2 or later is required.

Required hardware

XMOS offers the Mulit-Function Audio platform[2] which supports DSD64 and DSD128. XMOS does not currently provide any development hardware capable of DSD256. It is assumed that the reader has access to either a modified development board connected to suitable DAC, or custom hardware with the appropriate XMOS device, DAC and associated support components.

[1] https://www.xmos.com/download/public/USB-Audio-Software-Design-Guide(6.6.0rc5.a).pdf
[2] https://www.xmos.com/products/reference-designs/mfa

1 Introduction

1.1 DSD Background

DSD stands for “Direct Stream Digital”, a trademark name of Sony and Philips which describes implementation of a pulse-density (delta-sigma) modulation scheme used for representing audio streams.

Standard DSD uses a one bit data stream, generated from an 64x oversampled signal (compared with the base frequency of 44.1KHz), resulting in an overall bit rate of 2.8224Mbps per audio channel. This is often referred to as DSD64.

Increasing demand for higher resolution audio has seen the introduction of DSD128 and now DSD256 which increase the oversample ratio and bit rate to 128x, 5.6448Mbps and 256x, 11.2896Mbps respectively.

1.2 USB DSD Native

In DSD native capable systems, both the host (via the driver) and the audio device recognise DSD as a specific stream format. This means that the host can accept and process requests to play DSD streams from the media application. The device exposes a specific USB interface for playing native DSD content and knows up-front that a DSD stream is incoming and can setup the DAC hardware accordingly, preferably under a mute condition during the stream mode switch.

Native DSD is bandwidth efficient because the data is sent in raw binary format, removing the need for headers. The data is 32b word aligned resulting in no padding. This means a stereo DSD64 stream can be carried using 5.6448Mbps transport bandwidth. This data rate is the equivalent PCM at 88.2KHz sample rate using 32b audio samples, as shown in table 1.

At the time of writing, the only broadly available DSD Native capable host is a Windows machine using an ASIO driver, such as provided by Thesycon. For information about how to setup Native DSD on a Windows host using Foobar2000 player with the SACD plug-in, please refer to the following Q&A post on www.xcore.com[3].

1.3 USB DSD over PCM (DoP)

To support DSD in non-native DSD capable systems, a standard for transporting DSD data Over a PCM (DoP) stream has been created. A useful introduction to the DoP standard can be found at dsd-guide.com[4].

In DoP capable systems, the host has no awareness of DSD at the driver layer. Instead all data is treated as PCM. The media player packages up DSD content into a DoP frame, with a special DoP marker in the MSB. The data is then transmitted to the device, where it is also initially treated as PCM stream. It is only close to the Digital to Analog Converter (DAC) layer that the DoP header is recognised, and the DAC is put into DSD mode. The DoP marker header is stripped and 16 bits of DSD data are extracted from the 24b DoP frame and streamed to the DAC.

The benefit of this approach is that many PCM-only hosts can support DoP (with the appropriate DSD enabled audio device). The downsides of the DoP standard are possible corruption of DSD data if any volume scaling happens in the signal path, the need for stream time configuration of the DAC (with possible clicks/pops) and a doubling of the bandwidth required to transport DSD content.

The last point means that, due to the 8b DoP marker and 8b of padding to align to a long word, DSD64 requires a transport data rate equivalent to 176.4KHz sample rate, 32b audio. See table 1 below. This is exactly double the required bandwidth compared of native DSD. Fortunately, USB Audio Class 2 over High Speed USB offers ample bandwidth for DoP, even when extended to DSD256.

[3] http://www.xcore.com/questions/2673/how-setup-native-dsd-playback-mfa-board
[4] http://dsd-guide.com/dop-open-standard

At the time of writing, XMOS's standard USB Audio 2.0 software reference design does not currently support greater than 384KHz sample rate PCM. Extending the reference design to higher sample rates (up to 764KHz) may be the subject of future application notes, using a 45.1548 / 49.152MHz MCLK, and is expected to be feasible for two channel applications.

DSD speed	Sample rate	Equiv. PCM data rate Native	Equiv. PCM data rate DoP
64 fs	2.8224 MSPS	88200 Hz	176400 Hz
128 fs	5.6558 MSPS	176400 Hz	352800 Hz
256 fs	11.2896 MSPS	352800 Hz	705600 Hz

Table 1: Data rates for DSD native and DoP

Note: The DSD sampling speed is relative to a nominal 44.1KHz rate

2 Code changes to the USB Audio 2.0 Device Reference Design Software

The following section describes the code changes required to enable DSD256.

2.1 Enable higher sample rates

This code change within the file `custom defines.h` has the effect of increasing receive buffer sizes, and exposes higher sample rates to the host through the driver, for both DSD native and PCM interfaces.:

```
#define MAX_FREQ            (384000) //Enables DSD256 native and PCM sample rates supporting DoP DSD128
```

or:

```
#define MAX_FREQ            (768000) //Enables DSD256 native and PCM sample rates supporting DoP DSD256
```

2.2 Implement dummy I2S routine for unsupported PCM sample rates

In many cases, the high PCM sample rate (768KHz) required to carry DoP DSD256 will not actually be supported by the hardware. For example it can not supported using a 22.5792MHz MCLK - see below table.

Sample rate	I2S Bit clock	Divider ratio for MCLK = 22.5792MHz
176400 Hz	11.2896 MHz	2
352800 Hz	22.5792 MHz	1
705600 Hz	45.1584 MHz	not possible

Table 2: MCLK vs BCLK ratio for PCM output

It is therefore necessary to modify the I2S function to handle an advertised 705.6KHz and 768KHz PCM rate in a safe manner. The code below assumes that the maximum PCM sample rate supported by the DAC is 192KHz. The modification below prevents the I2S data output from occurring above the given limit. The following code should be added to the `audio.xc` file, in between the DSD output and I2S output code:

```
if (curSamFreq > 192000){       //Maximum PCM sample rate supported by the DAC.
        samplesOut[0] = 0;      //Above this, any DAC data will be zero
        samplesOut[1] = 0;
}

#pragma xta endpoint "i2s_output_l"
```

Finally, an `default` case needs to be added to the clocks generation code to ensure that the I2S code continues to function, albeit at a lower rate. Note that the I2S rate output to the DAC when the host is set to above 384KHz, will be 176.4KHz or 192KHz, assuming the MCLK is 22.5792MHz or 24.576MHz. However zero data is sent in this case, avoiding output noise:

```
#if (MAX_DIVIDE > 1)
        case 2:
        default:
        p_bclk <: 0xAAAAAAAA;
        p_bclk <: 0xAAAAAAAA;
        break;
```

For further information about how the reference design software handles I2S audio clocks, please see section 3.5.1 (Port Configuration) of the USB Audio Design guide[5].

Outputting of the DSD256 data to the DAC only requires an 11.2896MHz clock, so is acheivable from an MCLK running at 22.5792MHz, as illustrated below.

DSD speed	DSD Clock	Divider ratio for MCLK = 22.5792MHz
64 fs	2.8224 MHz	8
128 fs	5.6558 MHz	4
256 fs	11.2896 MHz	2

Table 3: MCLK vs DSD clock ratio for DSD output

2.3 Write setup code to put your DAC into DSD256 mode

The function `AudioHwConfig()` found in the file `audiohw.xc` is called each time there is a change in the stream type. For example, a sample rate change, bit depth change or stream type change. This code is responsible for configuring the DAC into an appropriate mode. In the test case for this application note, tested using the PCM1795 DAC, no changes to the code were necessary because the DAC will accept DSD64, DSD128 and DSD256 data using the same register settings.

2.4 Other suggested firmware changes

As mentioned earlier in section 2.2, enabling high rate DoP will likely expose invalid PCM sample rates to the host. To further ensure that no noise is output during either the DoP detection phase, or accidental selection of an invalid PCM sample rate, it is recommended to apply a hardware mute when dummy PCM mode is enabled.

To implement this, add a conditional mute to the `AudioHwConfig()` function in the file `audiohw.xc`. This can either be a soft mute using register setting, or preferably using a hard mute line. The following code is suggested:

```
if((dsdMode == DSD_MODE_OFF) && (samFreq > MY_MAXIMUM_SAMP_RATE)){
        <my mute code>
}
```

Tip: Don't forget to add the appropriate "un-mute" code when a valid PCM sample rate is selected. This can be done by adding un-mute code after configuring the DAC for a valid sample rate.

2.5 Results

The test hardware used for developing this application note running the modified firmware successfully advertises the capability of DSD256 native, and DoP256 (via the exposure of 705.6KHz PCM sample rate.

[5] https://www.xmos.com/download/public/USB-Audio-Software-Design-Guide(6.6.0rc5.a).pdf

Playing back DSD256 content via either mode results in the the expected audio output with the following waveform observed on the output pins.

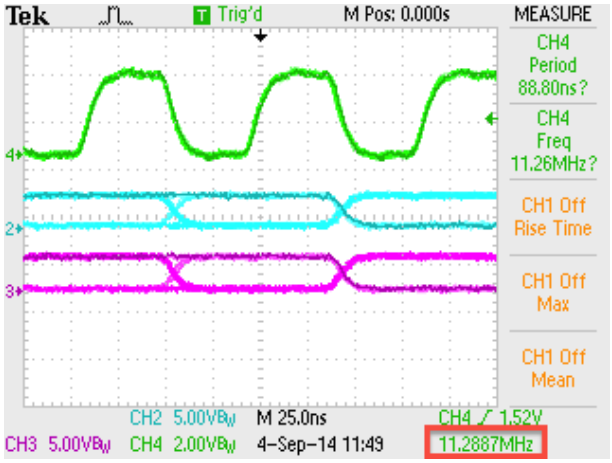

Figure 1: DSD256 scope capture

The DSD data and clock run at the expected data rate of 11.2896 MHz (see table 3). The DSD data is clocked into the DAC on the rising edge of the DSD clock (oscilloscope trace channel 4) and the waveforms are properly aligned and meet the DAC's required setup and hold times. Audio streams without any artefacts.

In some cases, depending on host PC performance, it may be necessary to increase the size of the ASIO buffers due to the high data rate required by DSD256. This will help avoid any dropouts due to the host not keeping up with the device's requests for stream data. Adjustment of the buffer depth can be achieved via settings in the Thesycon control panel, which is supplied with the Thesycon ASIO driver.