

Extracting Audio from ISOCH Traffic in a USB Trace

Chetan Kopalle
 Applications Engineer
 February 2024

LeCroy USB traces provide useful visuals to understand traffic. With a few easy steps, users can also hear what was playing on their USB2 Audio Class Endpoint. This paper aims to guide users through the process of identifying and extracting captured audio data and generating a playback file.

We've included an "USB_Audio_Export.usb" trace in the Protocol Suite Examples folder to use for this demo. This example is a Hi-Speed USB2 trace recorded with Protocol Suite v9.30, although the process and principle should apply to any USB trace with ISOCH audio data.

The example trace file can be found at: **C:\Users\Public\Documents\LeCroy\USB Protocol Suite\Examples\2.0 Traces**

Finding the Data

To begin, we need to capture a trace file while audio is being transferred between a Host and an Audio Class device – examples include speakers, microphones, and the DAC inside a phone-to-headset dongle. For this demonstration, we simply played music through a speaker while recording the USB2 traffic with a Teledyne LeCroy analyzer.

Use the hiding tools and view the trace at the Transfer Level for a clearer picture:



Figure 1

Without any previous knowledge of the protocol environment, we can infer where the audio might be streaming. Amidst a series of short Control transfers, we see a couple of Isochronous transfers in a single direction with a significant data payload (fig2). This is our audio stream.

Transfer	F	Control	ADDR	ENDP	bRequest	wIndex	Alt. Setting	wLength	Time	Time Stamp			
42	S	SET	18	0	SET_INTERFACE	1	1	0	497.200 us	3 . 100 011 532			
Transfer	F	Control	ADDR	ENDP	Audio Class	bRequest	wValue	wIndex	tSampleFreq	Time	Time Stamp		
43	S	SET	18	0	SET_CUR	SAMPLING_FREQ_CONTROL	Endpoint 0x0003	0x00BB80 Hz		11.406 ms	3 . 100 508 732		
Transfer	F	Isoch	ADDR	ENDP	Bytes Transferred	Isochronous Packet Info			Time	Time Stamp			
44	S	OUT	18	3	783360	4080 packets ranging from 192 bytes to 192 bytes			4.144 sec	3 . 111 915 066			
Transfer	F	Control	ADDR	ENDP	Audio Class	bRequest	CS & CN	Interface	EntityId	wLength	wVolume	Time	Time Stamp
45	S	SET	18	0	SET_CUR	VOLUME_CONTROL & 0x00	0x00	Feature Unit ID	2	0xFA36		31.624 ms	7 . 256 337 966
Transfer	F	Control	ADDR	ENDP	Audio Class	bRequest	CS & CN	Interface	EntityId	wLength	wVolume	Time	Time Stamp
46	S	SET	18	0	SET_CUR	VOLUME_CONTROL & 0x00	0x00	Feature Unit ID	2	0xF9F7		1.701 ms	7 . 287 962 100
Transfer	F	Control	ADDR	ENDP	Audio Class	bRequest	CS & CN	Interface	EntityId	wLength	wVolume	Time	Time Stamp
47	S	SET	18	0	SET_CUR	VOLUME_CONTROL & 0x00	0x00	Feature Unit ID	2	0xF9B7		517.233 ms	7 . 289 663 332
Transfer	F	Isoch	ADDR	ENDP	Bytes Transferred	Isochronous Packet Info			Time	Time Stamp			
48	S	OUT	18	3	5690781	29640 packets ranging from 192 bytes to 93 bytes			5.624 sec	7 . 806 896 432			

Figure 2

Hints and Tips

We began recording before connecting the USB speaker to the Host through the analyzer. By capturing full enumeration and the audio, Protocol Suite automatically detected and applied the appropriate Decoding settings. In addition to the audio, there are a few Audio Class Volume Control events in this trace (fig 2).

If enumeration was not captured, the Audio Class events will not appear as decoded transfers (fig 3):

Transfer	F	Isoch	ADDR	ENDP	Bytes Transferred	Isochronous Packet Info						Time	
44	S	OUT	18	3	783360	4080 packets ranging from 192 bytes to 192 bytes						4.144 sec	
Transfer	F	Control	ADDR	ENDP	D	TP	R	bRequest	wValue	wIndex	wLength	Bytes Transferred	Time
45	S	SET	18	0	H->D	C	I	0x01	0x0200	0x0600	2	2	31.624 ms
Transfer	F	Control	ADDR	ENDP	D	TP	R	bRequest	wValue	wIndex	wLength	Bytes Transferred	Time
46	S	SET	18	0	H->D	C	I	0x01	0x0200	0x0600	2	2	1.701 ms
Transfer	F	Control	ADDR	ENDP	D	TP	R	bRequest	wValue	wIndex	wLength	Bytes Transferred	Time
47	S	SET	18	0	H->D	C	I	0x01	0x0200	0x0600	2	2	517.233 ms
Transfer	F	Isoch	ADDR	ENDP	Bytes Transferred	Isochronous Packet Info						Time	
48	S	OUT	18	3	5690781	29640 packets ranging from 192 bytes to 93 bytes						5.624 sec	

Figure 3



In this case, we can manually set the decodes using the Decoding Settings:

Capturing enumeration will also have captured Control Transfers. Using the Decoded Fields view, we can quickly browse through the descriptors in these Control Transfers to learn more about the speaker.

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time
3	S	GET	18	0	GET_DESCRIPTOR	CONFIGURATION type, Index 0	0x0000	24 Descriptors	6.221 ms

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bDescriptorType	8	936	0x24	0x24	CS_INTERFACE Descriptor Type
bDescriptorSubType	8	944	0x02	0x02	FORMAT_TYPE descriptor subtype
bFormatType	8	952	0x01	0x01	FORMAT_TYPE_I
bNrChannels	8	960	0x02	0x02	Indicates the number of physical channels in the audio data stream.
bSubframeSize	8	968	0x02	0x02	The number of bytes occupied by one audio subframe. Can be 1, 2, 3 or 4.
bBitResolution	8	976	0x10	0x10	The number of effectively used bits from the available bits in an audio subframe.
bSamFreqType	8	984	0x01	0x01	Indicates how the sampling frequency can be programmed.
tSamFreq[1]	24	992	0x00BB80	0x00BB80	Sampling frequency 1 in Hz for this isochronous data endpoint.

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	1016	0x07	0x07	Size of the descriptor, in bytes
bDescriptorType	8	1024	0x25	0x25	CS_ENDPOINT Descriptor Type
bDescriptorSubType	8	1032	0x01	0x01	

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	1072	0x09	0x09	Descriptor size is 9 bytes
bDescriptorType	8	1080	0x05	0x05	ENDPOINT Descriptor Type
bEndpointAddress	8	1088	0x03	0x03	This is an OUT endpoint with endpoint number 3
bmAttributes	8	1096	0x01	0x01	Types - Transfer: ISOCRONOUS, Sync: No, Usage: Data EP
wMaxPacketSize	16	1104	0x00C0	0x00C0	Maximum packet size for this endpoint is 192 Bytes. If Hi-Speed, 0 additional transactions per frame
bInterval	8	1120	0x01	0x01	The polling interval value is every 1 Frames. If Hi-Speed, 1 uFrames/NAK
bRefresh	8	1128	0x00	0x00	Refresh Rate 2^n ms where n = 0
bSyncAddress	8	1136	0x00	0x00	Synchronization Endpoint (if used) is endpoint 0

Figure 4

This confirms that we're looking for an Isochronous OUT Transfer on Address 18 Endpoint 3.

This document does not contain export-controlled information.

Exporting the raw data

Note down the address and endpoint for the Isochronous transfer – Addr 18 Endpt 3 in the example trace. Confirm that the trace is set to the Transfer Layer Level View, then select “Export->Data” from the “File” dropdown menu:

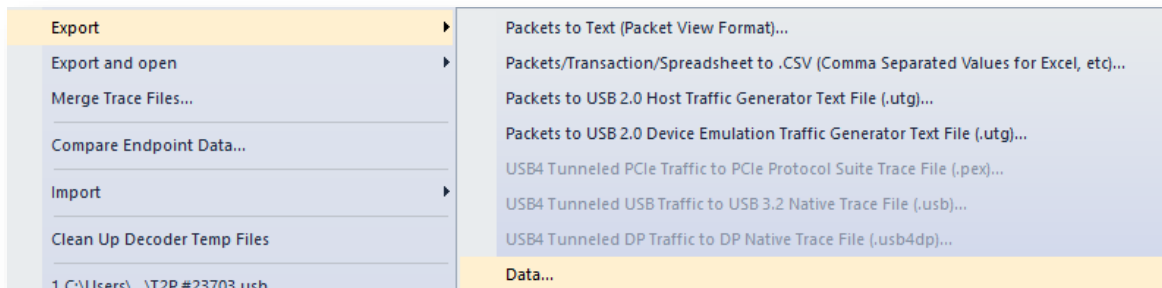


Figure 5

Input the Address, Endpoint, and Direction, then select “Binary” as the Save option. Then select the Range (note: this must be from “Transfer” to “Transfer”):

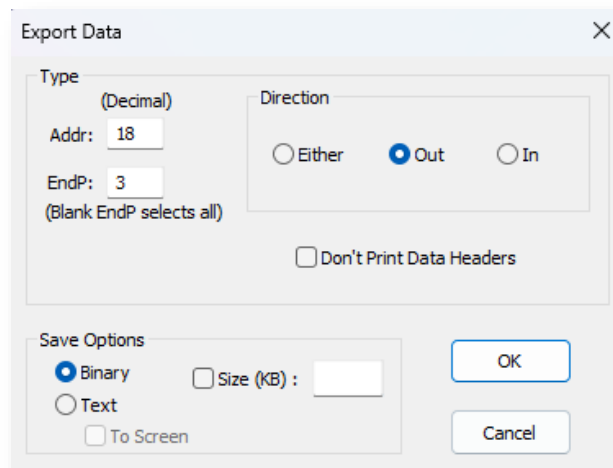


Figure 6

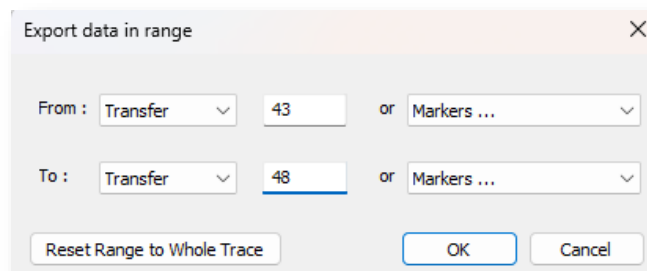


Figure 7

This will output a .dat file with a specified name in the selected location. This is our raw audio stream, but it requires additional meta data to enable playback.

Converting to Audio

Some Digital Audio Workstations (DAWs) provide options to import raw data for audio. This example uses Audacity which, at the time of writing, is a free and open-source DAW available at <https://www.audacityteam.org>.

Launch Audacity and select “Import->Raw Data”, then select the .dat file exported from Protocol Suite:

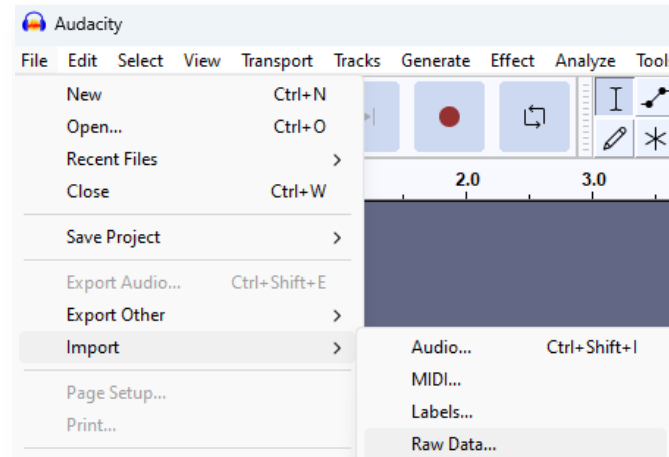


Figure 8

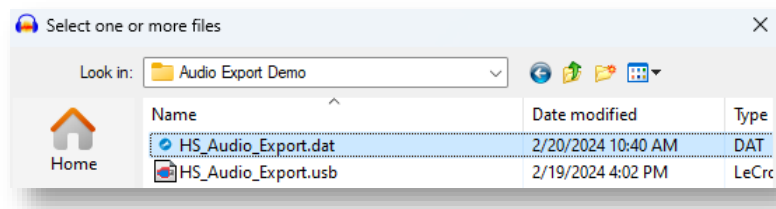


Figure 9

Fill in the Import settings with information found in the Configuration Descriptors and the USB Audio Device Class specifications. The “Detect” option is sometimes accurate. The correct settings for our example file can be found in Figure 10.

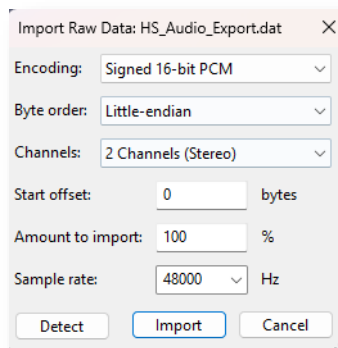


Figure 10

Click the Import button and play back your audio!

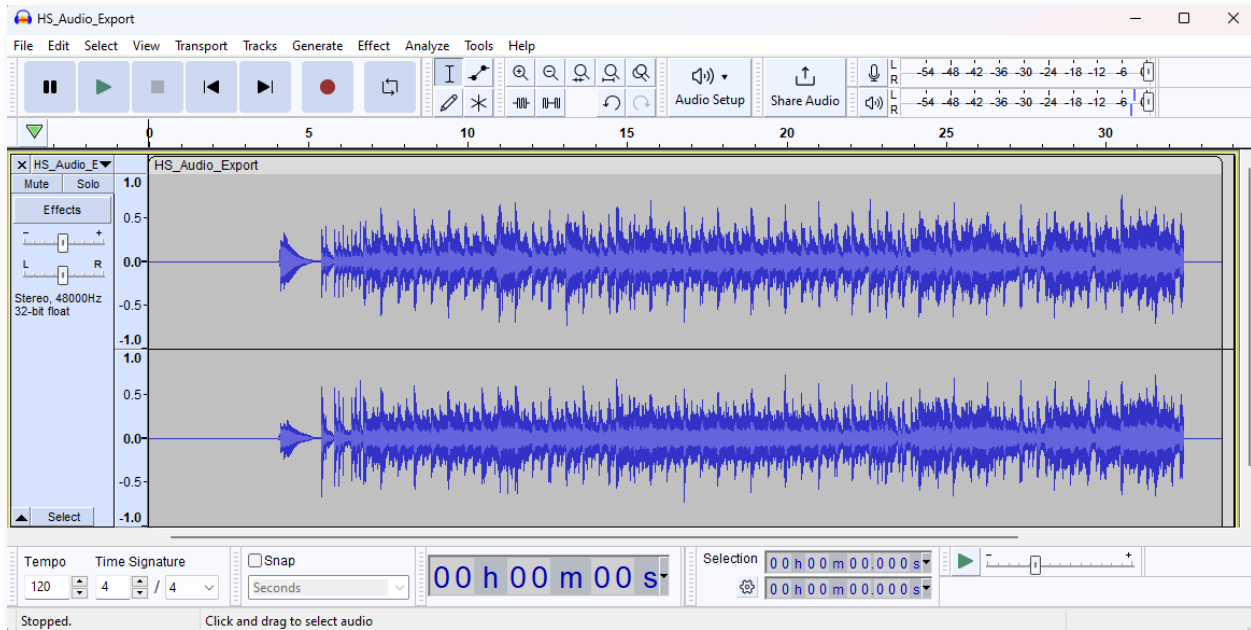
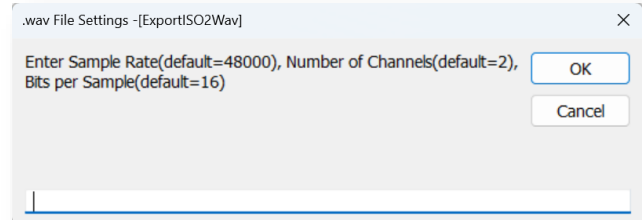
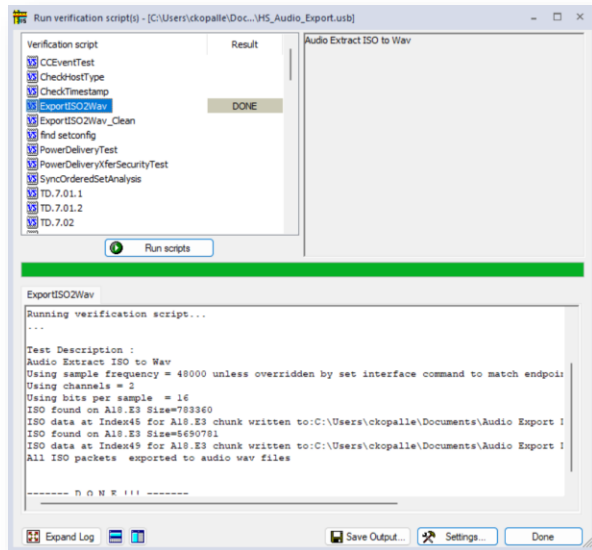


Figure 11

When exporting as a .wav/other file from Audacity, the software will automatically append the necessary metadata (file headers, etc.) to generate a .wav file from the binary data.

Using VSE

This process can be achieved (and simplified) by using the CATC Verification Script Engine. We've included a script in our examples folder that will parse through a trace and automatically export the payload of any Isochronous transfers as a binary file. VSE allows us additional write functions to this binary file — we can write in the header data for a .wav file and save it with a .wav extension. The result is a fully packaged .wav file that's ready for playback. This script has three sections to replicate the manual process described above. Refer to the comments in the script for more information.



This script has been set up to provide an option to manually input settings for the .wav file, but also provides default values. It's possible to build on this script to automatically parse the descriptors for the sample frequency, channels, etc. and insert those values in the .wav header.

This script has three sections to replicate the manual process described above. In “On Start Script” it prompts the user for audio meta data. In “Process Script”, each block of audio is written to a file prefixed by the audio meta data to make it a valid .wav file. In “On Finish Script”, the user is presented with a quick text report as the .wav file is automatically opened in the OS default audio player. Please refer to the comments in the script for more information.

The example script is “**ExportISO2Wav.vse**”, located here by default:

C:\Users\Public\Documents\LeCroy\USB Protocol Suite\Scripts\VFScripts\Examples

In Conclusion

While this paper focused on audio data, the same principle can be applied to any raw data transferred over USB and prepended with an appropriate file header.

For additional examples, resources, and learning opportunities, please visit TeledyneLeCroy.com/Support.