Tutorial I: Basic ASIO app

This tutorial will help you to design a layout for a sound-based app in Visual C++ by using a simple sample app.

Pre-requisites:
Windows Programming in Visual C++

Stuff you will need:
Visual Studio 2013: trial version available at
https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx
ASIO SDK: download at http://www.steinberg.net/en/company/developers.html
ASIO4ALL driver: download at http://www.asio4all.com/ (Not necessary if you already have another audio driver supporting ASIO. Almost all the audio drivers support ASIO.)

Where Can I learn Visual C++?
There are a number of online resources where you can easily learn programming in Visual C++. MSDN itself has a satisfactorily documented reference for Visual C++ available at:
https://msdn.microsoft.com/en-us/library/60k1461a.aspx
You can also refer to **Programming Windows Fifth Edition** by ***Charles Petzold***. This is an excellent book written in simple language for all those passionate beginners who want to learn Windows programming.

ASIO:
ASIO stands for Audio Stream Input/Output. As Wikipedia says "**Audio Stream Input/Output** (**ASIO**) is a computer sound card driver protocol for digital audio specified by Steinberg, providing a low-latency and high fidelity interface between a software application and a computer's sound card. Whereas Microsoft's DirectSound is commonly used as an intermediary signal path for non-professional users, ASIO allows musicians and sound engineers to access external hardware directly."
Extract the ASIOSDK2.3.zip on your PC. I assume that you extract it in C:.
Inside the extracted ASIOSDK2.3 folder you will find a pdf file by the same name i.e. 'ASIO SDK2.3.pdf'. This file gives a detailed description by the Steinberg, the makers of ASIO. If you don't want to read the whole document then just read the Overview to get a basic idea.

Making an ASIO integrated Win32(Windows) Project:
Follow these steps:

- Open Visual Studio 2015 version.

- File->New->Project
- Visual C++ -> Win32 Project -> Click on OK
- A window pops. Click Next -> Check Empty project ->Click Finish.
- In the Solution Explorer, right click on Source Files and add the file hostsample_basic.cpp.
- Extract asiosdk2.3 into C:.
- Right-Click on project name in Solution Explorer and click properties.
- Configuration Properties -> General -> Character Set -> Use Multi-Byte Character Set.
- C/C++ -> Preprocessor -> Preprocessor Definitions -> after the ';' add _CRT_SECURE_NO_WARNINGS
- C/C++ -> Additional Include Directories -> add the following directories:
    1. C:\asiosdk2.3\host\pc
    2. C:\asiosdk2.3\host
    3. C:\asiosdk2.3\common
- Click on OK.
- Add the following cpp files in your project:
    1. C:\asiosdk2.3\common\asio.cpp
    2. C:\asiosdk2.3\host\asiodrivers.cpp
    3. C:\asiosdk2.3\host\pc\asiolist.cpp
- Click on Build -> Build Solution. The project will build successfully.
- Debug -> Start Debugging (to run the app).

App Description:

The app allows you to select an audio driver and then after that a blank space appears. Press space and a tone (a sine wave) of 275Hz starts playing. Press space again and the sine wave will stop into silence.

Reading time!

So now we have successfully installed the app and before proceeding further you have to study some stuff regarding sampling of audio signal. There will be a few terms which you should be knowing like:

1. 8bit/16bit/24bit/32bit (or sample size)
2. Sampling frequency
3. Buffer size
4. Granularity
5. Input and Output Latency
6. Amplitude
7. Frequency
8. Intensity
9. Noise

10. Distortion
11. Dynamic Range

You might just GOOGLE all these terms to get a quick idea. Or you can also refer to the ASIO SDK pdf. Besides here are some suggested links:

Suggested links: https://en.wikipedia.org/wiki/Digital_audio
http://blog.prosig.com/2008/04/14/what-is-db-noise-floor-dynamic-range/

How to Use the app:

After you have got a basic idea of all above mentioned stuff you can use our sample app for your purpose. After starting the app and selecting a driver you can use Alt+Tab to go to the Visual Studio window (since the app is in full-screen mode without the title bar) and press Alt+2 to see the Output window displaying Min. buffer size, max buffer size, granularity, sampling frequency, etc.

The BufferSwitchTimeInfo function:

This function is a part of the 'Boiler plate' code of the ASIO setup. It gives us access of all the i/o channels. The function does the following for every channel:

- è Checks if the channel is input or output
- è Checks the size i.e. bit-depth of the channel buffer (which can be 8bit/16 bit/24bit/32bit etc)
- è If output then calls the function load_buffer in which we load a buffer, passing it into the particular output channel. Example: Say we write a sine wave and then send it buffer by buffer to play a tone of a particular frequency.
- è If input then calls a function store_buffer which stores the buffer sent by the particular output channel. This Buffer can be then dissected/analyzed the way we want to get different characteristics of the incoming signal.

Note that we have two store buffer functions in our code: store_buffer_left for left input channel and store_buffer_right for right input channel. But we have only one load_buffer function which loads a sine wave into the stereo system.

So you can just close your eyes and use the load_buffer function to load a sound into the output of your sound system and store_buffer function to store and analyse the recorded sound.

Generating 275Hz Sine wave:

To generate a Sine wave of 275Hz you have to access the output channels of your system. Hence the function of interest is load_buffer. Here we assume that you are having a stereo system on your PC i.e. two output channels. So we load the same sine wave in both the channels. In case of mono-output system you can just use the same code just replace line number 916 to 919 by:

nBufferIndex++;

Now the tone of frequency f0 is represented by:

$X = A*\sin(2*PI*f0*t)$

Here Sample size is the size of your sample i.e. 2 raised to the bit depth. But the values vary from negative to positive hence we have used half of sample size – 1 as the amplitude(A) of our sine wave. In our case the buffSize was 512 so we used this number directly in the for loop. You can replace it by buffSize however If you want. Now, f0 will be 275, and the t will be written in terms of no. of samples i.e. N/44100, where N = i+nOffset. nOffset is updated by buffSize after each iteration to make it continuous.