WaveIO Version 1.04
A Soundcard interface for LabView with ASIO support

C. Zeitnitz

© C. Zeitnitz 2005-2015
Web-page: http://www.zeitnitz.eu/waveio

1. Introduction

The WaveIO package consists of a DLL and a set of interface LabView VIs to access the different functions of the package. In addition you will find some examples how to use the sound card interface. WaveIO allows to use nearly all features of the sound card hardware:

- Use all supported sampling formats of the sound card
- Allow to use all channels provided by the sound card
- Capture the signals from any sound source, e.g. media player (only non DRM protected media)
- Use multiple sound cards simultaneously and with different formats
- Control the sound mixer of the devices
- Utilize ASIO[1] devices (useful for applications with a high channel count)

The connection between the LabView VI and actual sound hardware is done via the standard Windows API. So ALL sound systems providing drivers for Windows should be accessible via the WaveIO interface. This applies to on-board sound, PCI sound cards as well as USB sound systems.
The parallel operation of multiple sound cards is supported.
Additional VIs allow to control the Windows Mixer in order to select the source and destination of soundcard, mute or un-mute and monitor/control the volume level.

2. Requirements

The only requirement for the usage of the WaveIO components is currently LabView 7.1 or above running on a Windows 2000/XP/Vista/7/8 system with at least one soundcard. The requirements on the system performance are low, but depend on the used parameters. For high channel counts and high sampling rates the load on the system can be significant.

3. Installation

To use WaveIO just copy the files waveio.dll and waveio.llb into the LabView directory user.lib. At the next LabView start the Vis should show up in the function palette under User Libraries. The example VI should be copied to a directory of your choice.

4. Features of the WaveIO Interface

The interface allows the user to define the following parameters (applies to input and output of sound data):

- Select the soundcard
- Number of I/O buffers (multi-buffering)
- Size of the I/O buffers (given in time or samples)
- Number of channels
- Sampling rate in 1Hz Steps
- Resolution up to 32Bits
- Channel map

---

[1] ASIO is a trademark and software of Steinberg Media Technologies GmbH

- Timeout for watchdog thread
- Sound Mixer control

4.1. I/O Buffers

Buffers are used to pass the sound data from the application to the Windows API and vice versa. The total size of the buffer(s) defines the time when the next data have to be available. Since other applications require some CPU time as well or the application itself might be busy, multiple buffers allow an asynchronous access.

A quasi real-time access is still possible, because the individual buffer size can be small (e.g. 50msec). The total buffer time is given by the number of buffers times the time per buffer. Example: 50msec buffer size and 10 buffers allow for ½ second total buffer time and should avoid any buffer over- or under-run problem. Buffer times below 20msec should be avoided!

Short buffer times should be avoided, if system performance is an issue. This is due to the fact that the provided VIs try to access the DLL within the individual buffer time (50msec in the above example). So, if the load on the system seems high increase the buffer time per channel!

The size of the buffers can be defined in terms of the time (in msec) per channel, or in number of samples per channel.

4.2. Sound Parameters

For the access of the soundcard some parameters have to be specified: Number of channels, sampling rate and resolution per sample.

Channels

The numbers of channels depend on the capability of the sound card. A standard 7.1 card, can in principle have 8 output channels and most likely 2 input channels.

For ASIO devices, the maximal number of channels is currently 32. Instead of the number of channels a channel mask can be provided to the open vi (set "Channel Mask?" input to true).

Sampling Rate

The sampling rate for all channels can be set in 1Hz steps. Most current sound cards are able to use sampling rates up to 100kHz in 1Hz steps. Only very old cards stick to the standard rates of 44.1kHz, 22.05kHz etc. Some special cards can handle higher rates up to 192kHz or above. It has been observed in the past, that the timing of some sound cards is incorrect, when deviating from the standard settings (e.g. 44.1 kHz).

Resolution

Most current soundcards will work with any resolution between 8Bit and 16Bit and some cards even with 24Bits. The WaveIO DLL can handle resolutions per sample up to 32Bits. Be aware that at 8 Bit resolution the sound card is working with unsigned values and for all other resolution with signed samples.

ASIO devices are only supported with samples as 32 bit integer. This is automatically selected by the DLL. The input value for bits per sample is ignored.

5. *LabView Components of the Interface*

5.1. Soundcard Interface VIs

The WaveIO package contains the following VIs in waveio.llb:

- **WaveIO_GetInfo:** extract the names of devices available for the provided device type (Play, Record, Loopback, ASIO input, ASIO output)

- **WaveIO_GetDeviceID:** retrieve hardware related information from the attached audio devices, e.g. vendor and product identification number, serial number and a unique identification number.

- **Waveio_GetInfo_ASIO:** extract the names and some properties of ASIO devices. An array of cluster is returned containing the information about the buffer size, some tested sampling rates and the sample format

- **WaveIO_OpenControlPanel:** Open the Control Panel for the defined device type (record, playback, ASIO)

- **WaveIO_Open**: opens a sound device for recording or playback. The device has to be started before actually reading/writing data to it. The open will return an error if the device is not existing, already open or the selected sound format is not supported by the device. A watchdog thread is started, if the specified timeout is greater than zero. The watchdog will close the device, if the no call to DLL for the specific device is recorded within the given time window. For each device (and I/O mode) a separate watchdog thread is started.

- **WaveIO_Start**: starts the opened sound device

- **WaveIO_Play**: sends the provided data to the soundcard. An error will occur, if the size of the provided data does not match the buffer setting of the open command.

- **WaveIO_Record**: waits and retrieves data from the soundcard.

- **WaveIO_Stop**: stops the sound device. Required before closing the device.

- **WaveIO_Close**: closes the sound device.

5.2. WAV File VIs

The DLL contains routines to directly send the received data from the soundcard to a standard wave file, which can be played with any media player. Example code can be found in RecordWave_and_WAVFile.vi

- **WaveIO_FileOpen:** Open a file for a given device. The Soundcard handle of an opened soundcard has to be passed to the VI. In addition the size of the used buffers is specified. The size of the buffer allows to record data, which have been taken some time ago.

- **WaveIO_FileStatus:** Check the status of a file. Return the current buffer range stored for recording and returns flags about the status of the file.

3

- **WaveIO_FileStop:** Stop the writing of data to the file. This does NOT close the file yet! A subsequent call of WaveIO_FileRec will continue the writing of data to the same file.
- **WaveIO_FileRec:** Start the actual writing of data to the file. The VI requires a time where to start the writing and a duration. The start time of a given buffer (nbuf), as returned by FileStatus or WaveIO_Record, is given by nbuf•(samples/buf)/(sampling rate)•1000msec.
  All times given in milliseconds!
- **WaveIO_FileClose:** Close the file. This will create the actual *.wav file

- **WaveIO_AddWaveFileChunk:** Adding data (array of bytes) to the header of the wave file

Be aware, that the times required by WaveIO_FileRec have to be in sync with the buffer numbering in WaveIO!

The above Vis call the corresponding routines in the waveio.dll. The dll itself does not call any LabView libraries.


5.3. Sound Mixer VIs

The DLL contains routines to directly access some features of the sound mixer. These Vis work for Windows XP as well as Vista/7, even though the sound API changed completely.
- **WaveIO_FindRecordingDevice:** Find the recording devices for given criteria. These are the recording source (Mic, Line …) defined by an input enumerator and an optional card name, which can be part of the actual card name. The device returns arrays of names and device IDs.
- **WaveIO_FindPlaybackDevice:** Find the playback devices for given criteria. These are the playback destinations (Speaker, Line …) defined by an input enumerator and an optional card name, which can be part of the actual card name. The device returns arrays of names and device IDs.
- **WaveIO_SelectRecordSource:** Select a given source (Mic, Line …) defined by an input enumerator for the selected device. Optionally the source can be un-muted and all volume controls set to 100%.
- **WaveIO_SelectPlayDestination:** Select a given destination (Speaker, Line …) defined by an input enumerator for the selected device. Optionally the destination can be un-muted and all volume controls set to 100%.
- **WaveIO_Mute:** Mute or un-mute the given device (source or destination). The device type (recording or playback) has to be specified by an enumerator. The input value "Get State" return true for a muted device.
- **WaveIO_Volume:** Set or extract the volume of the recording or playback device. The destination can be specified.

6. Using the Interface

The following picture shows the signal flow when recording sound with the WaveIO VIs. The Buffer constant specifies two buffers with 100msec per channel for each buffer. The constant specifying the sound format sets the number of channels to two with a sample rate of 44.1kHz and 16Bit resolution.
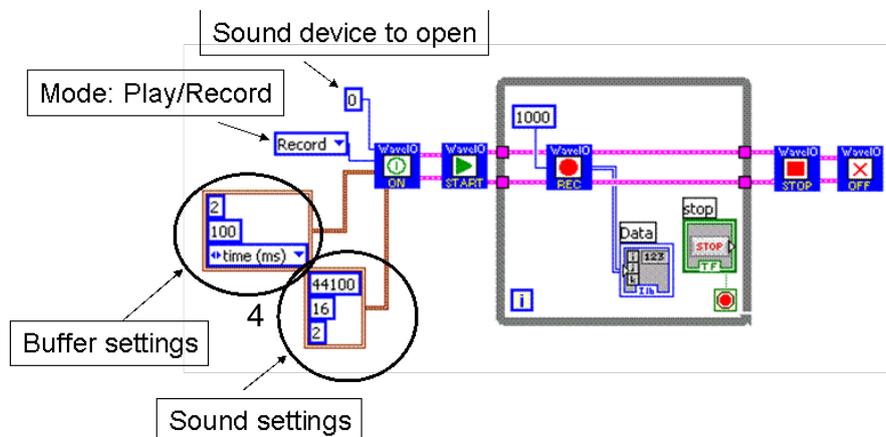
The number of the sound device corresponds to the Windows enumeration of the available soundcards starting at zero! The names of the corresponding soundcard can be retrieved with the GetInfo Vi.

After the device has been opened successfully the device has to be started. In the while loop the data are retrieved from the soundcard. The timeout is set to 1000msec.

ALL data which are played or recorded are returned or have to be provided as a multi-dimensional array! Each dimension corresponds to one channel. For mono data the second dimension should be set to 1 as shown in the sound output example below. Be aware, that these code snippets are not complete!

In order to decouple the actual data transfer from/to the soundcard from the rest of the application, the actual interface part should be run in a separate thread and communicate via queues (for data and commands) with the main application.

Pack mono data in multi-dim array
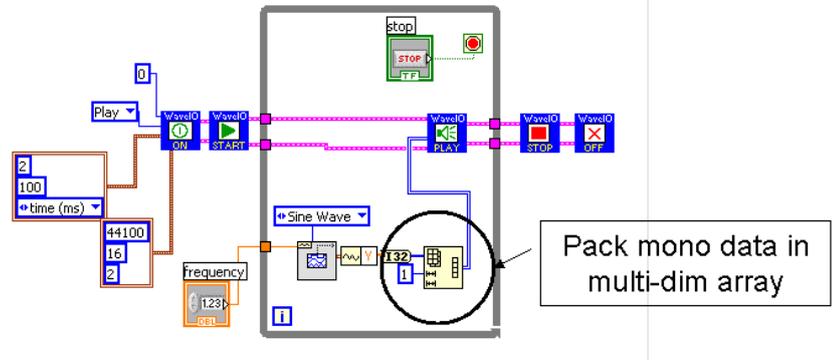
7. Enclosed examples

The enclosed examples show the operation of the sound card interface for input and output.

- **PlayWave.vi –** Simple sine wave generator for multiple channels (default setting for two channels). You can define the sound card to use and set the format and buffer parameters. Works for device type selector Play and ASIO_ouput.
- **RecordWave.vi –** Record multiple channels from a single sound card and display the output in a graph. Selectable sound card and buffer parameters. Works for device type selector Record, Loopback and ASIO_input
- **PlayWave_multi_cards.vi –** Generator for sine/triangular wave on two sound cards simultaneously. Selectable sound cards, format and buffer parameters. In the example both cards use the same parameters, but this can easily be changed.
  RecordWave_multi_cards.vi – Record the signals from two cards simultaneously. Selectable sound cards and buffer parameters.
- **RecordWave_and_WAVFile.vi** – Record the signal from a single sound cards and allow to write data to a wave file as well.
- **TestMixer.vi –** test the mixer functionality (mute, set volume, card selection)
- **ASIOTest.vi** – testing the ASIO device with output to a text file (asiotest.log)
- **ASIO_Input_Output.vi** – record and play sound from/to an ASIO device
- **ASIOInfo.vi** – just extract and display information about found ASIO devices

The examples are just meant as a starting point for your own application. Be aware, that the generation and recording should be decoupled from your application. So start the VIs as independent tasks (dynamic VI) and transfer the data from the recorder to the application via a queue. This will ensure, that no buffer over or under runs can occur.

8. Signal sources

The signal source recorded with the WaveIO interface can be selected manually or via the above Vis in the Windows mixer. Regularly the following sources are available:
- Line-In (stereo)
- Microphone (mono for most soundcards)
- Wave – PC internal sound from a MP3 or media player
- CD-ROM – signal directly connected from the CD-ROM to the soundcard

5

Sometimes additional inputs like AUX are available.

On most cards only one of the above signals is selectable at a given time. In order to select the signal source open in the sound mixer the recording control.
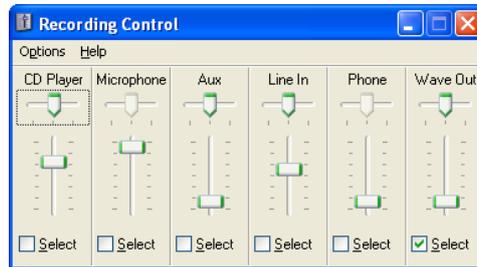
Sound Control of Windows XP

Note for Windows Vista/7/8
Be aware, that in Windows Vista/7 the mixer has been changed completely. Usually the mixer is not accessible directly, but for each source/destination a separate device is available. Auto jack connection detection complicates things. This can be disabled by a registry hack (search the Web for "jack sensing EnableDynamicDevices").

Loopback-Devices

In addition to the regular signal sources, you will find in the list of input devices something like "Loopback: Speaker". This is a pseudo-device, which is capturing the sound send to the speaker in the Windows mixer. Be aware, that only non DRM protected media can be captured, otherwise you will receive only silence!

9. ASIO Support

In order to utilize the ASIO interface, a card with an ASIO driver is required.
Most standard sound cards can be operated via ASIO by installing the ASIO4ALL driver.
The ASIO support is currently only limited to input and output with a data format of 32 bit integer per sample and the maximal number of channels is 32. There are no special VIs for ASIO. Just use the same as for standard Windows devices and change the device type to ASIO input or output. The current interface does not provide information about latencies and time stamps. It is mainly useful to support devices with high channel count.
Be aware, that the ASIO API does not allow to use multiple devices simultaneously. So you can only open a single device for both input and output. WaveIO will return an error, if you try to open a second device.

10. Final Remark

This Interface provides an easy access to the soundcard and allows to utilize most of its features. Some limitations exist:
- Only PCM data are supported by the current DLL. In principle other codecs could be interfaced, but this would require some serious re-write of the code.
- Some parameters have to be set manually in the properties of the device. For this you need to access the Windows sound properties and mixer. This can be open the from Labview via the shell command VI:
  - Windows XP: "sndvol32". This will bring up the volume control and the selector for the signals that are mixed for the speakers. Adding the option for recording (sndvol32 /r) will start the recording control as seen above
  - Windows Vista/7/8: "control.exe mmsys.cpl,,0"
    The trailing number selects the device type 0: playback, 1: recording

If you have any additional questions please contact: info@zeitnitz.eu

11.   Term of Usage

This Software is provided as is and can freely be distributed for private and non-commercial usage.
In case of a commercial usage a license is required. Please obtain the license from
https://secure.shareit.com/shareit/product.html?productid=300424978
or contact info@zeitnitz.eu

Troubleshooting

The code has been tested with quite a variety of soundcards, but nevertheless problems could show up on some cards. Hang ups in Windows can cause the code to run into the timeout and buffer over- and underruns can occur. In the following some hints what to do under some standard conditions.

Sound format is not supported

Any soundcard I know supports 44.1kHz sampling rate at 16Bit resolution and two channel input/output. So if the interface complaints even at these settings, there is definitely something wrong with the sound driver! Other formats, especially higher channels counts require the corresponding settings in the driver software to be consistent with the selected value. E.g. if you try to output 6 channels on a 5.1 system you have to enable all channels in the soundcard control.

Buffer overrun/underrun occurred

Buffer over- or underruns occur, when the time between calls to the Windows API are in excess of the total buffer time per channel (time per buffer time the number of buffers). This can happen, when some other process takes up too much CPU time, or the machine is busy with disk access.
An increase of the total buffer time can help to remedy this problem. The number of buffers should be at least two, but a number of four buffers is more reliable. Higher numbers are usually not necessary. If the load on the system is high as well, increase the size of samples per channel in order to reduce the number of calls to the DLL.

Device already open

If a device has not been closed correctly and you try to re-run your application you might get this error. This is due to the fact, that LabView does not unload the DLL when stopping the VI. You'll have to close LabView completely and re-start it in order to get rid of this problem. If you experience this problem frequently you can allow the DLL to start a watchdog thread. This thread will close the device automatically after the given timeout period. Per default no watchdog is started. The watchdog thread is started for each successfully opened device, if the specified time (wired to the open VI) is greater than zero.

Input data do not match Format

The WaveIO_play VI will return this error if the number of samples in the data array does not match the sound format specified when the device was opened. If this error shows up, please check, that you provide sufficient values: Multi-dimensional array with a total size of number of samples per channel times the number of channels.

My old LabView program is no longer working with the new WaveIO version

Some VIs have a slightly changed interface, so you have to re-link for example the WaveIO_Open VI to the new version in WaveIO.llb, since some new options have been added

ASIO: Signal amplitudes are strange

The ASIO assumes 32bit integer data. The input for the number of bits per sample is ignored.

ASIO: Second device cannot be opened

The ASIO API does not allow to open multiple ASIO devices. So, first close the open device before opening another one. This is a shortcoming of the ASIO API and not under the control of the author of this program.

Copyrights

1. All rights of the WaveIO package lie with Christian Zeitnitz
2. ASIO is a trademark and software of Steinberg Media Technologies GmbH
3. The ASIO interface has been developed with the help of the SDK provided by Steinberg Media Technologies GmbH, Hamburg, Germany