USB AUDIO INTERFACE GUIDE

FOR DIGITAL SENSORS AND SIGNAL CONDITIONERS

THE MODAL SHOP
AN MTS COMPANY

modalshop.com | 513 351 9919

USB AUDIO INTERFACE GUIDE

FOR DIGITAL SENSORS AND SIGNAL CONDITIONERS

REVISION HISTORY

This document was derived from and supersedes the “ED-0295 USB Audio Interface Guide” and the “Digiducer – Sensitivity White Paper” documents. This revision adds references to models V485B39 and IV 485B39, and defining Version 3 format – high sensitivity input range for digital signal conditioner.

PRODUCT SUPPORT

For additional questions, contact The Modal Shop at 800.860.4867 or 513.351.9919, 9 a.m. to 5 p.m. EST. If it is more convenient, fax your questions or comments to The Modal Shop at 513.458.2172 or email our technical staff at info@modalshop.com.

COPYRIGHT

DISCLAIMER

The following paragraph does not apply in any state or country where such statements are not agreeable with local law: The Modal Shop, Inc. provides this publication “as is” without warranty of any kind, express or implied, including but not limited to, the implied warranties of merchantability or fitness for a particular purpose. This document is subject to change without notice, and should not be construed as a commitment or representation by The Modal Shop, Inc.

This publication may contain inaccuracies or typographical errors. The Modal Shop, Inc. will periodically update the material for inclusion in new editions. Changes and improvements to the product described in this manual may be made at any time.

TRADEMARKS

THE MODAL SHOP
AN MTS COMPANY

10310 Aerohub Boulevard
Cincinnati, OH 45215 USA
Phone: 513.351.9919
FAX: 513.458.2172
www.modalshop.com
info@modalshop.com

TABLE OF CONTENTS

1. INTRODUCTION

This is a guide on how to interface to a growing family of digital sensors and digital signal conditioners that use the USB Audio format as the core communication protocol. This includes the Digiducer branded digital sensors such as the Model 333D01, Model 333D02, Cosen MB63, and the PCB Piezotronics Model 633A01. The Digital ICP® - USB Signal Conditioner Model 485B39 is included because it interfaces in a very similar manner.

The primary audience is those who are interested in developing software applications to acquire and scale sensor data. A user that is primarily interested in performing analysis may want to consider an existing set of applications on various platforms that have already been optimized for use with these devices.

While this document may provide some advice on the audio software interfaces available with different operating systems, it is beyond its scope to provide full details given the diversity of operating systems and interfaces. Any reference for accessing audio data from a USB microphone will provide a good starting point.

The focus of this document is extensions that have been implemented to support the simple acquisition and scaling of sensor data.

1.1 Reference Material

Audio Device Document 1.0
: The low level USB audio protocol. Typically operating systems provide device drivers and audio interfaces so this level of information is not usually needed.
http://www.usb.org/developers/docs/devclass_docs/audio10.pdf

Thesycon USB Descriptor Dumper
: This is a useful Windows utility to access USB descriptor data.
http://www.thesycon.de/eng/usb_descriptordumper.shtml.

Digiducer GitHub
: Examples for a few different languages and environments.
https://github.com/digiducer

Sensitivity Decoder Spreadsheet
: Caution: This contains VBA macros that are used if scanning for attached devices is used. Calculations can still be performed in Excel without the macro if the sensitivity is entered manually.
https://github.com/Digiducer/General/blob/master/Digiducer%20Calibration%20Decoder.xlsm

WAV file format definition
: This tends to move around so the link may not always be valid.
http://soundfile.sapp.org/doc/WaveFormat/

2. BACKGROUND

Because the integrated sensors are digital accelerometers, traditional analog sensitivity (scaling factor) units such as mV/g are not the most accurate. We have determined that units of Digital Counts (24 bit resolution) per m/s^2 at 100Hz at 48 kHz sample rate would be the most universally accepted and accurate.

The digital signal conditioners will use units of Digital Counts (24 bits resolution) per 1 Vpeak at 100Hz at 48 kHz sample rate for sensitivity. An additional scaling will be necessary using the sensor sensitivity to calculate engineering units.

Typical analog accelerometers output a voltage proportional to the acceleration they undergo. Conventional systems use some variety of separate analog to digital conversion hardware to read the analog sensor. With the digital sensors, the conversion to digital occurs within the sensor itself so the only outputs are digital samples. Because of this, a voltage / acceleration sensitivity value is not the most accurate.

The most accurate units are Digital Counts / (m/s^2). This provides calibration of the analog to digital conversion as well as the acceleration sensing element. The Digital Counts is based on a 24 bit resolution sample. It can be scaled to 16 bit samples by dividing by 256 (bit shifting right by 8).

The following figure illustrates conceptually how the digital accelerometer sensitivity is determined.

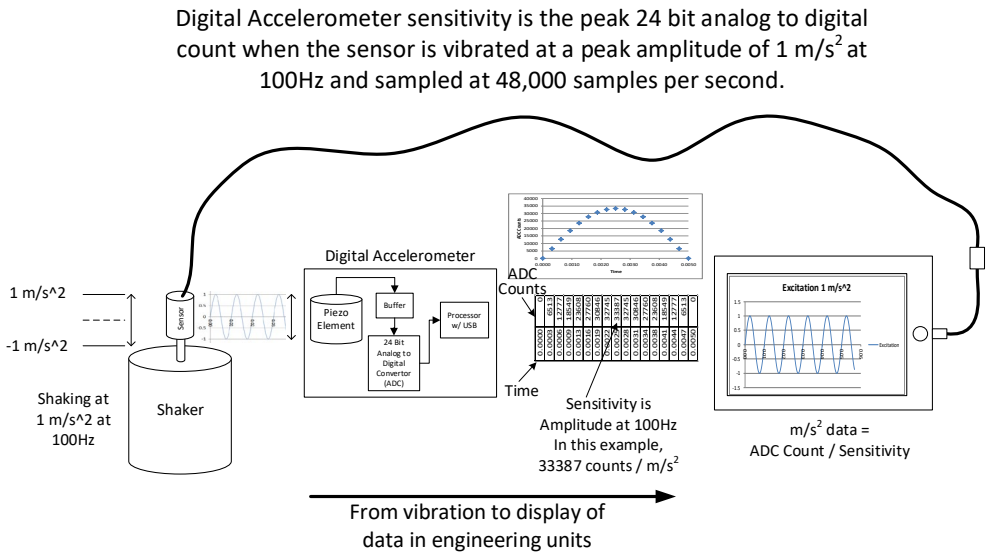

Figure 1 – Digital Accelerometer Sensitivity

The Digital ICP® - USB Signal Conditioner Model 485B39 converts ICP sensor signals from voltage to digital samples. Instead of being limited to only a single axis of acceleration data, the digital signal conditioners interface to the wide family of ICP (IEPE) sensors. Sensor types include accelerometers, force, microphone, and tachometer sensors. They also support 2 channels of phase coherent data. The implementation and processing is similar to the digital accelerometers. The applicable units are Digital Counts / Vpeak.

3. SOURCES OF SENSITIVITY

Device sensitivity data is available from a variety of sources. The recommended source is the data embedded in the device itself to optimize the user experience. The software applications can access this information and use it to scale the acquired data into calibrated units. This is defined in the following sections.

A calibration sticker is typically attached to the device packaging. A full calibration certificate is typically available upon request from where you purchased the device.

There is also a recommended format to record sensitivity information into a WAV file and to extract the sensitivity from the WAV file.

It is possible to manually decode the sensitivity based on the device name.

We also provide a Windows based Excel spreadsheet that scans for attached devices, reads the calibration data, and calculates a variety of typical units. This can also be useful tool as software is developed.

We also provide some examples in a few languages available from GitHub.

3.1 Supported Model Numbers

MODEL	DEVICE TYPE	NOTES
333D01	Digital accelerometer	Single axis of vibration.
333D02	Digital accelerometer	Industrialized package with electronics separated from sensor.
'MB63_'	Digital accelerometer	Private label – support is optional. Model name is padded by 2 spaces.
633A01	Digital accelerometer	Alternate labeling. Support is strongly recommended.
485B39	Digital Signal Conditioner	ICP sensors to digital data. Also includes V485B39 (voltage input) and IV485B39 (ICP and voltage inputs).
SDC011	Digital Signal Conditioner	ICP sensors to digital data. In an industrial housing

Table 1 – Supported Model Numbers

3.2 Embedded Sensitivity

The USB format describes a number of descriptor fields that allow text based information to be defined. These devices utilize those fields for to encode sensitivity information as well other useful information. Typically, the model field is used for the device name in the operating system. The encoded information is straightforward to decode.

Prototype 333D01 units did not have the calibration feature implemented. A nominal calibration value of 33000 should be used for channel A (or 1) and 65000 for channel B (or 2) if the calibration data is not present.

For general ease of use on Windows 7 and later, the more recent formats are providing the calibration data in both the model field and the serial number field. We have added the date of calibration as well. The serial number string up to the version identifier has remained the same so applications can accommodate multiple versions.

We recommend software applications look for specific model numbers, confirm the version letter, and that the full data field is defined to recognize a device adhering to this protocol.

It is not recommended to utilize the 'Digiducer.com' manufacturer name because some devices will be private labeled. We also accidentally changed the spacing on the manufacturer field part way though product deployment.

The V485B39 is a voltage only input version of the 485B39 i.e. there is no ICP current. The IV485B39 has ICP on channel 1 and voltage only on channel. The programmed model number still is 485B39 because as far as software and calibration is concerned, it behaves the same as the 485B39.

3.2.1 Version 0 Format – Acceleration (Obsolete)

This format was used in a limited number of beta units and these will be upgraded to version 1. This data was only encoded in the USB serial number descriptor. We recommend applications that have already implemented this version maintain it for backward compatibility and use the version character to determine the version. There is no need for new applications to support this format.

The version 0 format of the information is a follows:

NNNNNNssss0sAAAAAAsBBBBBB

Where

NNNNNN is 6 digit serial number.
s is space.
0 is single character format definition. This indicates format 0 being used.
AAAAAA is 6 digit sensitivity value in counts / (m/s^2).
BBBBBB is 6 digit sensitivity value in counts / (m/s^2).

3.2.2 Version 1 Format – Acceleration

This is the format used by the digital accelerometer products.

The version 1 format serial number field:
NNNNNNssss1NNNNNNAAAAABBBBBYYMMDD

The version 1 format model number field:
MMMMMMs1NNNNNNAAAAABBBBBYYMMDD

Where
MMMMMM is 6 digit model number (333D01, 333D02, 'MB63 ' is padded with 2 spaces, etc.).
s is space.
NNNNNN is 6 digit serial number.
1 is single character format definition. This indicates format 1 being used.
AAAAA is 5 digit sensitivity value in counts / (m/s^2).
BBBBB is 5 digit sensitivity value counts / (m/s^2).
YY is the last two digits of the year (00-99).
MM is the month number (1-12).
DD is the day (1-31).

3.2.3 Version 2 Format – Voltage

This is the format used by the digital signal conditioner products.

The version 2 format serial number field:
NNNNNNssss2NNNNNNAAAAAAABBBBBBBYYMMDD

The version 2 format model number field:
MMMMMMs2NNNNNNAAAAAAABBBBBBBYYMMDD

Where
MMMMMM is 6 digit model number (485B39, etc.).
s is space.
NNNNNN is 6 digit serial number.
2 is single character format definition. This indicates format 2 being used.
AAAAAAA is 7 digit sensitivity value in counts / Vpeak.
BBBBBBB is 7 digit sensitivity value counts / Vpeak.
YY is the last two digits of the year (00-99).
MM is the month number (1-12).
DD is the day (1-31).

3.2.4 Version 3 Format – Low Range Voltage

This is the format used by the digital signal conditioner products. For increased sensitivity, there is a version of the digital signal conditioner that will have a reduced full scale voltage input range that may be below 1 Vpeak. This requires a calibration standard below 1 Vpeak. This format is the similar to Version 2 except against a 50 mV reference.

The version 3 format serial number field:

NNNNNNssss2NNNNNNAAAAAAABBBBBBBYYMMDD

The version 3 format model number field:

MMMMMMs2NNNNNNAAAAAAABBBBBBBYYMMDD

Where

MMMMMM is 6 digit model number (485B39, etc.).
s is space.
NNNNNN is 6 digit serial number.
2 is single character format definition. This indicates format 2 being used.
AAAAAAA is 7 digit sensitivity value in counts / 50 mVpeak.
BBBBBBB is 7 digit sensitivity value counts / 50 mVpeak.
YY is the last two digits of the year (00-99).
MM is the month number (1-12).
DD is the day (1-31).

3.3 WAV File Sensitivity

The WAV file format supports the addition of arbitrary “chunks” of custom defined data. We have defined the following chunk to retain the calibration information with data recorded from these devices. Recorder applications should define this chunk. Analysis applications should interpret this chunk and scale data correctly for analysis.

Applications that include both recording and analysis can utilize their own formats to retain scaling information.

For general WAV file information, search Google for WAV file formats or take a look at this link:
http://soundfile.sapp.org/doc/WaveFormat/

Applications that do not recognize a specific chunk ID should just skip that chunk.

The chunk will have the following format:

CAL1LLLLMMMMMMss<Serial Number Descriptor><padding spaces>

Where:

CAL1 is the chunk identifier per the WAV specification. It is a 4 byte item

LLLL is the chunk length per the WAV specification. It is a 4 byte integer and defines the length of the rest of the chunk.

MMMMMM is the model number. It will be set to 6 character model number for the device. A decoding application should confirm the model number data is a Digiducer device because the chunk identifiers are uncontrolled. As far as we are aware, ‘CAL1’ is unused but this is an important cross check to avoid invalid data.

s Spaces to pad the model number to an even multiple of 4 bytes.

<The Serial Number Descriptor> formatted as defined in the previous sections.

Note: This format can also be created from the Model number information with some slight rearranging of the data if that is the most convenient to implement.

<padding spaces> Add the necessary spaces so the chunk is a multiple of 4 bytes in length.

3.4 Sample Rate Sensitivity Adjustment

The devices are calibrated at a sample rate of 48,000 samples per second. The analog to digital converter used in these devices slightly varies the amplitude (around 3%) in a linear fashion based on sample rate. For improved accuracy, the following adjustment to the sensitivity is recommended when sample rates other than 48,000 are selected.

Certain Microsoft Windows audio interfaces such as Wavin or DirectAudio are problematic because the actual sample rate is not actually controlled by audio function calls. These audio interfaces present the programmer with parameters to select the sample rate. The returned data is at the selected sample rate. However, the Windows audio function resamples the data without regard to proper signal processing creating the potential for aliasing. The actual sample rate is controlled by the Recorder interface in the advanced options. We advise users to set the sample rate to 48,000 and resolution to 24 bits. We advise when using these Windows audio interfaces to not adjust the sensitivity.

Other audio interfaces such as Windows Kernel Streaming or ASIO do not have this issue. We are not aware of other operating systems having this issue.

The adjusted sensitivity can be calculated using the following formula:

Sensitivity = (Slope * Sample_rate + Offset) * Sensitivity_48kHz

Where:

Sensitivity is the adjusted sensitivity
Slope is -7.9915858E-07
Sample_rate is the actual rate the device is sampling data
Offset is 1.03853340739
Sensitivity_48kHz is the device sensitivity as determined in the previous section

It also can be adjusted using the factors defined in the following table:

Sample Rate	Sensitivity Adjustment Factor
8000	1.03214014
11025	1.02972268
16000	1.02574687
22050	1.02091196
32000	1.01296033
44100	1.00329051
48000	1.00000000

Table 2: Sensitivity Adjustment Factors

The application would perform a table look up to determine the Sensitivity Adjustment Factor.

Sensitivity = Sensitivity_Adjustment_Factor * Sensitivity_48kHz

Where:

Sensitivity is the adjusted sensitivity
Sensitivity_48kHz is the device sensitivity as determined in the previous section

4. TYPICAL SENSITIVITY CONVERSIONS

Different software environments provide audio data in a variety of formats. Matlab® for example provides data scaled from -1.0 to 1.0 as a floating point values. Other environments provided data as signed integers.

4.1 Conversion to Acceleration

The table below has typical conversions from sensitivity to the scale factor to calculate either g's or m/s² from the digital samples. This sensitivity is obtained from format 1 defined above. **Sensitivity** is the value obtained using one of the methods previously defined.

From	Factor	Decimal	To
Digital Sample -2^{23} to $(2^{23}-1)$	$\frac{1}{\mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right) * 9.80665 \left(\frac{\frac{m}{s^2}}{g}\right)}$	$\frac{0.10197}{\mathbf{sensitivity}}$	g
Digital Sample -2^{23} to $(2^{23}-1)$	$\frac{1}{\mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right)}$	$\frac{1}{sensitivity}$	m/s²
Normalized -1.0 to 1.0 Matlab, M+P Smart Office	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{\mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right) * 9.80665 \left(\frac{\frac{m}{s^2}}{g}\right)}$	$\frac{855400}{\mathbf{sensitivity}}$	g
Normalized -1.0 to 1.0 Matlab, M+P Smart Office	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{\mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right)}$	$\frac{8388608}{\mathbf{sensitivity}}$	m/s²
%FSV -100% to 100%	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{100\left(\frac{\%}{full\ scale}\right) * \mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right) * 9.80665 \left(\frac{\frac{m}{s^2}}{g}\right)}$	$\frac{8553.99958}{\mathbf{sensitivity}}$	g
%FSV -100% to 100%	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{100\left(\frac{\%}{full\ scale}\right) * \mathbf{sensitivity}\left(\frac{counts}{\frac{m}{s^2}}\right)}$	$\frac{83886.08000}{\mathbf{sensitivity}}$	m/s²

Table 3 – Sensitivity to Acceleration

4.2 Conversion to Voltage

The table below has typical conversions from sensitivity to the scale factor to calculate voltage from the digital samples. This sensitivity is obtained from format 2 defined above. **Sensitivity** is the value obtained using one of the methods previously defined.

From	Factor	Decimal	To
Digital Sample -2^{23} to $(2^{23}-1)$	$\frac{1}{\mathbf{sensitivity}\left(\frac{counts}{1\,Vpeak}\right)}$	$\frac{1}{\mathbf{sensitivity}}$	V
Normalized -1.0 to 1.0 Matlab, M+P Smart Office	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{\mathbf{sensitivity}\left(\frac{counts}{1\,Vpeak}\right)}$	$\frac{8388608}{\mathbf{sensitivity}}$	V
%FSV -100% to 100%	$\frac{2^{23}\left(\frac{counts}{signed\ full\ scale}\right)}{100\left(\frac{\%}{full\ scale}\right) * \mathbf{sensitivity}\left(\frac{counts}{1\,Vpeak}\right)}$	$\frac{83886.08000}{\mathbf{sensitivity}}$	V

Table 4 – Sensitivity to Voltage

5. IMPLEMENTATION CONSIDERATIONS

There currently is no reliable way to get consistent phase synchronization between multiple USB devices. There is a random, linear phase shift between devices. It seems dependent on when the operating systems starts a data acquisition on each device.

Simultaneous connection of multiple devices does work on most of the supported operating systems (iOS is only known exception) but this is only suitable for amplitude only measurement.

5.1 Digital Accelerometers

The current digital accelerometers are all single axis devices. However, there is different scaling into a 2 channel A/D converter that results in one channel having a nominal 10g range and the other channel a nominal 20g range. Other than the different maximum range, there really is little difference in noise floor or other results.

Application developers can opt to only use the 20g maximum range channel if that works best for their applications. They also can provide both channels if the application is already multi-channel focused.

5.2 Digital Signal Conditioners

The Digital ICP® - USB Signal Conditioner Model 485B39 is a two channel device that supports independent channels. These channel will have a good phase match so are suitable for multi-channel analysis such as transfer functions. The same caveat previously mentioned applies. Only the channels on the same device will be synchronized. Multiple 485B39 will have a random linear phase shift between them.

There are variants V485B39 and IV485B39. V485B39 has ICP disabled on both channels and is useful for sensors that output voltage. The IV485B39 has ICP (IEPE) on channel 1 and ICP disabled on channel 2 for voltage output sensors. They will collectively be referred to as 485B39.

The SDC011 is a 485B39 variant assembled in an industrial housing that is rack mountable supporting a sealed USB cable. Unless otherwise specified, all references to the 485B39 are applicable to the SDC011.

The 485B39 will disable the ICP boost voltage and other circuitry when in USB suspend mode to reduce quiescent power consumption. This reduces the power draw by around 95% and is necessary to comply with the USB specification.

Suspend mode is indicated by a LED on the production units going off. The LED is illuminated when the USB device is active. Some of the earlier functional prototypes do not have the LED.

There will be an approximately 2 second settling time after the 485B39 exits suspend mode. This time is based on the ramp up time of the ICP power supply and predominantly the AC coupling capacitors settling.

This will show up as a slow drift in the DC bias of the acquired data. In the frequency domain, this is sometimes called ‘ski slope’ as the drift in the DC bias shows up as a very prominent peak around 0 Hz. It can dominate real frequency peaks of interest.

The USB suspend mode behavior depends on the operating system and configuration of the operating system. Windows 7 typically does not suspend USB devices in our experience. Windows 10 can especially in laptops. iOS tends to not suspend devices. Android depends on the device and version of the operating system. It can be very aggressive in suspending devices.

It also depends on how the acquisition is structured in the application. If it opens the audio port and keeps it open, we expect the USB device would not be suspended. There might be some initial settling time required for the best results.

If the application opens the audio port, acquires data, and then closes the port, the application may need to account for the settling time by disregarding the first seconds of data to allow the bias to settle. While this is dependent on the operating system and device, we recommend applications structured like this include settling time.

The 485B39 sensitivity information only gets the data to calibrated voltage. A secondary sensor sensitivity is necessary to provide engineering units to the user.

These recommendations are how to provide the best user experience based on our interaction with users. Developers are obviously free to select an implementation approach that you feel best.

We have a worldwide sales network that will be promoting our devices and providing customers with the compatible software options. Customers will gravitate to the applications they find the easiest to use.

For sensor sensitivity, we suggest providing two sensor sensitivity options. These would be supported for each channel.

The first option is manual entry of the sensitivity. It is standard to provide a calibrated sensitivity with a sensor. The user should be able to enter this number with no external calculations into your software.

For acceleration, standard units are volts / acceleration often in mv / g. For example, a common accelerometer is 100mV / g. There may be metric equivalents.

Other common dynamic sensors are microphones that measure sound pressure with units of mV / Pascal.

Force sensors with mV / Newton.

A voltage option would be useful to just display the raw sensor voltage.

Consider supporting custom units where the text label is associated with a sensitivity factor. We know of one customer interfacing to a current probe.

We offer a LASER tachometer that produces pulses proportional to RPM. The default for this would be voltage. More sophisticated application can calculate RPM from the data stream.

The second recommended approach is to support calibrating to a specific sensor. The user would provide the excitation level and sensor type. The application would acquire data with the sensor on a calibration device and calculate the sensitivity.

These comments on sensor sensitivity are in no way mandatory but rather reflect our understanding of our users and existing applications.

6. OPERATING SYSTEM CONSIDERATIONS

6.1 Microsoft® Windows®

The Windows® sound application programming interfaces (API) represent an evolving history of Windows with legacy interfaces remaining. The below figure is an attempt to capture the interrelationship of the various APIs.

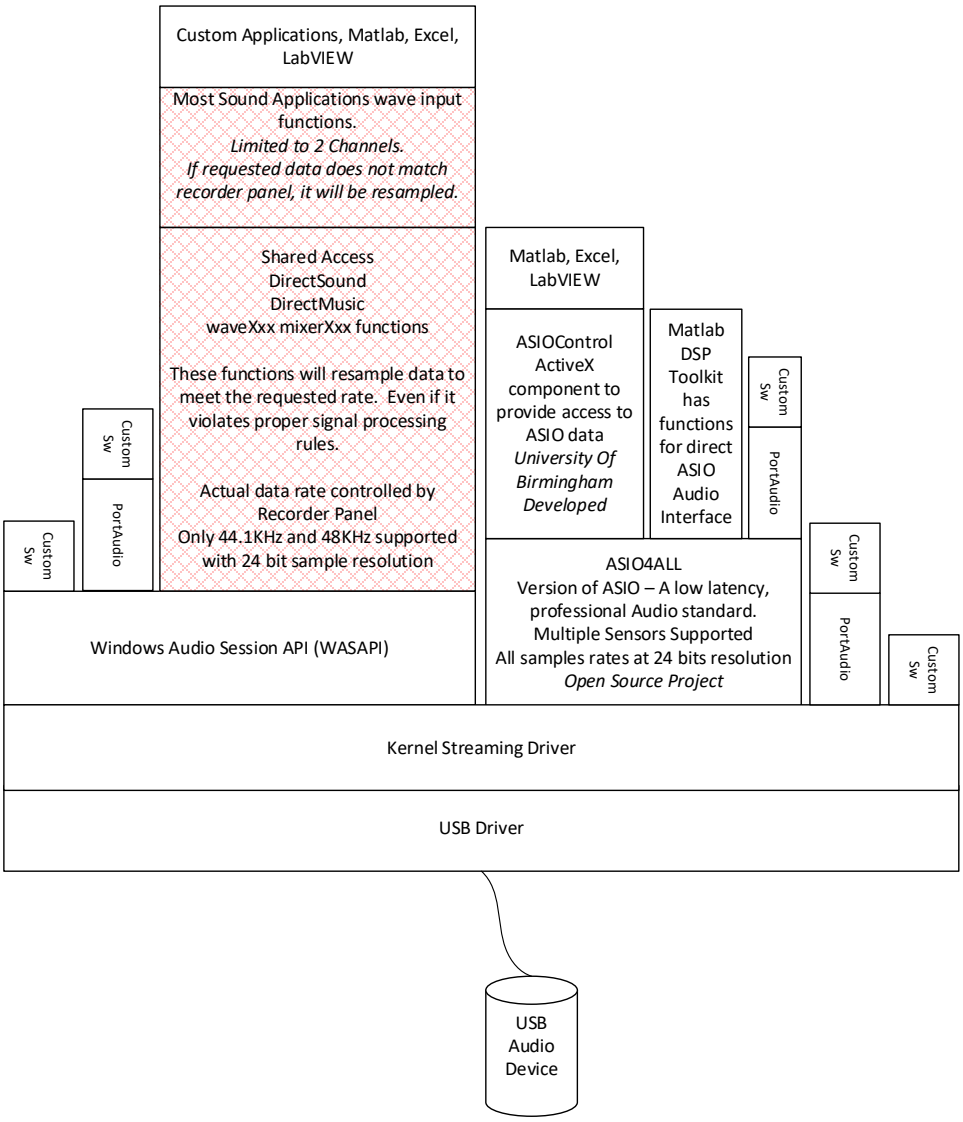

Figure 2 – Windows API Frameworks

For the simplest software interface, we recommend WaveIn with some caveats. The WaveIn interface allows the sample rate and data resolution to be specified. Unfortunately, if these settings do not match what is set up in the Windows recorder control panel, the interface will resample the data and often violate proper signal processing. For example, the recorder interface is configured to 48,000 samples per second and the WaveIn function requests 24,000 samples per second. Data will be returned at 24,000 samples per second but is subject to aliasing. For WaveIn usage, we recommend 48,000 samples per second and 24 bit resolution.

There are also restrictions on the resolution at lower sample rates. Windows recorder is focused on legacy audio rates so drops the resolution to 16 bits at samples below 44,100.

PortAudio© is an Open Source / Cross Platform (Windows/Linux/MacOS) Audio interface library available from www.portaudio.com. We have used it in examples and for internal usage. It works well and simplifies the interface to the Windows Kernel Streaming interface. Windows Kernel Streaming provides an interface to 24 bit resolution at all sample rates. More advanced programmers can access kernel streaming directly.

Matlab® uses PortAudio for its audio functions.

ASIO® is a low latency audio standard by controlled by the Steinberg company at http://www.steinberg.net/ .

ASIO4ALL© is a free application that allows general sound cards to support ASIO. We have found this hard to configure sometimes and not the most robust.

6.2 Linux®

Linux® is well supported with standard applications like arecord or Audacity®. These access the 485B39 like a basic audio device with no optimizations.

There are C based examples using PortAudio to simplify the interface to the Linux Audio application programming interface.

6.3 Android™

There is a lot of fragmentation in Android™ devices and Android operating systems. It was only a few versions back that Android added standard support for USB Audio devices. Prior to that, there were a few third party developed USB Audio devices.

We have found at least one Motorola® phone to be very quick to suspend the 485B39 at the conclusion of an acquisition. Special consideration is required to ensure the sensor bias has settled sufficiently.

Some versions of Android apply filters to the default audio input that can affect the signal validity in ways such as a higher frequency high pass filter or modifying the amplitude. Android implementations should be validated against know reference signals.

6.4 iOS®

iOS® only supports a single USB Audio device even if a hub is used.

6.5 MacOS®

MacOS® has the option to aggregate separate audio devices so they show up as one, multi-channel device. We were hopeful that this would provide synchronization between the devices but our testing a few years ago found the same device to device phase shifts.

The Modal Shop, Inc.

Phone: 513.351.9919
Fax: 513.458.2172
www.modalshop.com
info@modalshop.com