SPBAudiolib - Version 1.b1r4 - last revised: 08/16/2000

This RB module is open source. I am now entering the beta testing stage of development. Please send me your bug reports. Please indicate whether the problem you are having seems to be with the demo project or the module itself. Bugs in the module will be addressed in a higher priority than bugs in the demo project. Several people have been a great deal of help to my efforts, but I would like to offer special thanks to the guys working on TBFinder (who've made this project much simpler because of that great tool) and Doug Holton who has answered questions and helped tweak some of the methods. It was Doug's AIFFlib (another open source effort) that gave me the background to get this started.

If you want to contact me, my name is Benjamin J. Schneider (Ben) and my email is schneidb@ohio.edu. Feel free to drop me a post to let me know if you've improved on the module so I can incorporate your changes into the module that will be posted on my website. Also, let me know of any ideas you may have that should be incorporated that you may not know how to do. Whatever, feel free to contact me period. ;)

SPBAudiolib - What is it?

This module allows RB developers to make toolbox calls to the SoundInputManager. The goal was to create a complete audio recording solution. Unlike all of the other RB solutions available, SPBAudioLib allows you to select an input device, select an input source, adjust the available parameters of that device and source and record to ram or disk asyncrronously. As of a4, I have added compiler conditional statements to call CarbonLib for Carbon builds and InterfaceLib for Classic builds. If you want to use this module for OSX apps, please wait until OSX beta is released and you've tested with it. With a5, I have added a constant to the module called "TargetCarbon." This was added so the module can be used with RB v 2.X. If you are using RB v3.X, remove this constant.

NOTE: This module does not support playback at all. Doug Holton's AIFFlib does. I am using the MoviePlayer control for audio playback so I don't have a need to make these toolbox calls. If I choose to pursue playback, I'll most likely do it in another class or module so this one can be kept pure to recording.

Conventions:

Most of these methods return integer values. With methods that return an integer value, a value returned that is less than 0 usually means an error has occurred and the error returned will be documented in Inside Macintosh. So, if we are trying to set a device parameter, the only thing the return value represents is the error code. 0 usually means no error has occurred. If we are trying to retrieve data from a device, a negative value will still represent an error while a nil (0) value or a positive value will be a result we can use. If I needed to return a string of characters, I usually called a msgBox within the method to report the error instead of returning the error code from the method. As far as the exceptions to this go, they are noted in each function's description.

Instructions:

(Function) - SndSoundManagerVersion() As String - Returns the short version. It returns a string value to allow for the "**X.X.X"** version format. You should probably call this first to make sure all the below functions will be available to you. The current version is 3.6.5 which comes as part of QuickTime 4.1.2. (added in a4)

(Function) - SPBGetDeviceList() As String - This is probably the first call you should make after determining the version of the Sound Manager. It is the only device related method in this lib that doesn't require that you've already opened a device. When you call this method, it will return a comma delimeted list of all the audio devices on your Mac that are supported by the Sound Manager. (added in a1)

(Function) - SPBOpenDevice(Name as String) As Integer - You will need to call this method before attempting to call any of the others (with the exception of the above). This will open the device that you pass to it, ie "Built-in." Leave the device open as long as you need to use it, but be sure to close the device when you are done with it. When you call this method, it will return the input device reference number. Once you've made this call, you should store this number so you can pass it to the rest of the calls. (added in a1)

NOTE: Most of the below calls require that you pass the InputDevice Reference number that was returned from the SPBOpenDevice call. The reason for this is to allow you to open and use as many Input Devices as your Mac will support. The idea is if you have a Digidesign AudioMedia III card and the built in audio on your Mac, you can open them both and record from them simultaniously (this has not yet been tried or

tested, but it will be eventually).

(Sub) - SPBCloseDevice(InDevNum as Integer) - When you are done using a device, you should call this method and pass the InputDevice Reference number that you got from the SPBOpenDevice method. (added in a1)

(Function) - SPBGetDeviceName(InDevNum as Integer) As String - This method will return the name of the device number passed to it. (added in a5)

(Function) - SPBGetDeviceConnected(InDevNum as Integer) As Integer - This method will return 1 if the device is connected, 0 if the device is not connected and -1 if the SoundInputManager doesn't know whether or not the device is connected. It will generally return -1 for all devices that are not removable (as in the built in audio). This was added for USB audio devices and has not yet been tested. If you are using usb audio devices, you should call this function before trying to open the device. (added in a5)

(Function) - SPBGetDeviceAsync(InDevNum as Integer) As Integer - This method will return 1 if the device supports asynchronous recording and 0 if it does not. (added in a5)

(Function) - SPBGetDeviceContinuous(InDevNum as Integer) As Integer - This method will return 1 if the device is set for continuous recording and 0 if it is not. This method will only be valid if the device supports asynchronous recording. (added in a5)

(Function) - SPBSetDeviceContinuous(InDevNum as Integer, State as Integer) As Integer - This method should only be called if the device supports asynchronous recording. See Inside Macintosh SoundInputManager for more detailed information. (added in a5)

(Function) - SPBGetDeviceInputSource(InDevNum as Integer) As Integer - This method will return the current input source number. This is the older way of getting the input source. To get the source by name, use the SPBGetCurrentSource function. (added in a5)

(Function) - SPBGetCurrentInputSource(InDevNum as Integer) As String - This method will tell you what the current input source is for the device you have just opened (ie Sound In or CD Rom). Keep in mind, however, that Apple uses the OSType (four character string) for this so you should check out Inside Macintosh for all the possibilities. Some examples are "sinj" for Sound in, "cd " for CD Rom, "emic" for External Microphone. (added in a1)

(Function) - SPBGetInputSourceList(InDevNum as Integer) As String - This method will return a comma delimeted list of all the available input sources for the device. As with SPBGetCurrentInputSource, these are returned as an OSType. (added in a1)

(Function) - SPBSetDeviceInputSource(InDevNum as Integer, Source as Integer) As Integer- Use this method to set the input source by it's number. This is the older way of setting the input source. To set the source by name, use the SPBSetInputSource function. (added in a5)

(Function) - SPBSetInputSource(SName as String,InDevNum as Integer) As Integer - This method will set the device's input source to the source that you pass. Remember, these are OSType's (four character strings). (added in a1)

(Function) - SPBGetCurrentDeviceCompression(InDevNum as Integer) As String - This method will return the compression type currently used by the device. This is also an OSType. The default is "NONE." Refer to Inside Macintosh for other types. (added in a1)

(Function) - SPBGetDeviceCompressionList(InDevNum as Integer) As String - This method will return a comma delimeted list of all the available compression types the device supports. As with SPBGetCurrentDeviceCompression, these are returned as an OSType. (added in a1)

(Function) - SPBGetDeviceCompressionFactor(InDevNum as Integer) As Integer - This call will return the compression factor for the current device. This means that if you are using "MAC3," it will return 3 and if you are using

"NONE," it'll return 1. I have now called this method from within the SPBRecordToFile method to determine the number of bytes to be recorded, so please don't remove this method unless you want the SPBRecordToFile method to stop working. (added in a3)

(Function) - SPBSetDeviceCompression(InDevNum as Integer,CompType as String) As Integer - Pass the four character string representing the Compression type to change the compression type that the device will use. (added in a1)

(Function) - SPBGetDeviceSampleRate(InDevNum as Integer) As Integer - This method will return the device's current samplerate (most likely 44100). See the note at the end of the ReadMe for non-Apple Devices. (added in a1)

(Function) - SPBGetDeviceSampleRateList(InDevNum as Integer) As String - This method will return a comma delimeted list of samplerates. If the device only supports fixed samplerates, it will return a list of the samplerates that are supported. If the device supports variable samplerates, it will return a list of two values: The minimum and the maximum rate the device supports. If this is confusing to you, refer to Inside Macintosh. See the note at the end of the ReadMe for non-Apple Devices. (added in a1)

(Function) - SPBSetDeviceSampleRate(InDevNum as Integer,SRate as Integer) As Integer - Pass the desired samplerate to change the device to that samplerate. (added in a1)

(Function) - SPBGetDeviceSampleSize(InDevNum as Integer) As Integer - This method will return the device's current samplesize, ie 8bit ,16 bit, 24bit etc. (added in a1)

(Function) - SPBGetDeviceSampleSizeList(InDevNum as Integer) As String - This method will return a comma delimeted list of samplesizes that are supported by the device. (added in a5)

(Function) - SPBSetDeviceSampleSize(InDevNum as Integer,SSize as Integer) As Integer - Pass the desired samplesize to change the device to that samplesize. (added in a1)

(Function) - SPBGetDeviceNumberOfChannels(InDevNum as Integer) As Integer - This method will return the number of input channels that are being used and the format of the recorded file. (1 for mono, 2 for stereo). (added in a1)

(Function) - SPBGetDeviceChannelsAvailable(InDevNum as Integer) As Integer - This function will return the maximum number of channels that a device can support. Apple's built in audio only support 2 channels (stereo) but some 3rd party devices support up to 8 channels of audio. This has not yet been thoroughly tested. (added in a5)

(Function) - SPBGetDeviceCurrentChannels(InDevNum as Integer) As Integer - This function will return the number of channels that are currently being used by the device. This may seem redundant with the SPBGetDeviceNumberOfChannels, but it is not. Refer to Inside Macintosh SoundInputManager for more information on this. (added in a5)

(Function) - SPBSetDeviceNumberOfChannels(InDevNum as Integer,Num as Integer) As Integer - Pass the desired number of channels (1 for mono, 2 for stereo). (added in a1)

(Function) - SPBGetActiveNumberOfChannels(InDevNum as Integer) As Integer - The device will return 1 if it is set to pass audio only from the left channel. The device will return 2 if it is set to pass audio only from the right channel. If the device number of channels is 1 (mono) then a return of 3 means that the left and right channels will be mixed so their sum will be recorded as a mono soundfile. If the device number of channels is 2 (stereo) then a return of 3 means that the left and right input channels will be interleaved into a stereo file. The default is 3, so you may not even need to use this. (added in a2)

(Function) - SPBSetActiveNumberOfChannels(InDevNum as Integer,Num as Integer) As Integer - Just as with SPBGetActiveNumberOfChannels, you can set the mode with this method. 3 is the default and is what is recommended by Apple See above for the other two options (1 for left only, 2 for right only). (added in a2)

(Function) - SPBSetLevelMeter(InDevNum as Integer,State as Integer) As Integer - Pass 1 to this method to turn on the level meter. Pass 0 to turn off the level meter. (added in a1)

(Sub) - SPBGetActiveLevels(InDevNum as Integer,ByRef LCh as Integer,ByRef RCh as Integer) - This subroutine will set the LCh and RCh parameters to the current left and right audio levels of the device. To use this for a VU Meter, you'll have to call this over and over again very quickly, either in a thread with a loop or with a timer with a vey short period. The range of values that LCh and RCh will be changed to is from 0 to 255. I have not yet broken it down into what value represents what dBv (and it may be different from device to device), so you'll have to play around to scale things properly. If you're unsure how to use ByRef parameters, see the project file or refer to the RealBasic documentation. (added in a2)

(Function) - SPBGetLevel(InDevNum as Integer) As Integer - This method will return the mono (or sum) audio level at the time it is called. To use this for a VU Meter, you'll have to call this over and over again very quickly, either in a thread with a loop or with a timer with a vey short period. The range of values returned is from 0 to 255. I have not yet broken it down into what value represents what dBv (and it may be different from device to device), so you'll have to play around to scale things properly. (added in a1)

(Function) - SPBSetPlayThru(InDevNum as Integer,State as Integer) As Integer - Pass 1 to this method to enable Playthru. Pass 0 todisable Playthru. Playthru means that the audio coming into the device is routed to the output of the device. Different devices and input sources behave differently. (added in a1)

(Function) - SPBGetPlayThru(InDevNum as Integer) As Integer - This method will return 1 if playthrough is turned on and 0 if playthrough is turned off. Playthru means that the audio coming into the device is routed to the output of the device. (added in a1)

(Function) - SPBGetTwosComplement(InDevNum as Integer) As Integer - This method will return 1 if TwosComplement is turned on and 0 if TwosComplement is turned off. When the samplesize is set to 16 bits, TwosComplement is automatically turned on. When the samplesize is set to 8 bits, TwosComplement is automatically turned off. TwosComplement means that if it is on, the sampledata will be stored as TwosComplement values (meaning -128 to 127). If it is off the sampledata will be stored as offset binary values (0 to 255). In most cases, you'll want TwosComplement to be on. Otherwise, your audio will be distorted. (added in a1)

(Function) - SPBSetTwosComplement(InDevNum as Integer,State as Integer) As Integer - This method will set set the device's TwosComplement state to On or Off. You will pass 1 for on and 0 for off. (added in a1)

(Function) - SPBGetOptionsDialogAvailable(InDevNum as Integer) As Integer - This method will return 1 if the device driver for the current device has it's own options dialog box and 0 if it does not. See the note at the end of the ReadMe for non-Apple Devices. (added in a1)

(Function) - SPBShowOptionsDialog(InDevNum as Integer) As Integer - If the device driver has it's own options dialog box, calling this method will show it. See the note at the end of the ReadMe for non-Apple Devices. (added in a1)

(Function) - SPBGetDeviceAGC(InDevNum as Integer) As Integer - This method will return 0 if the AGC is off and 1 if it is on. AGC stands for automatic gain control. Not all devices support this. (added in a2)

(Function) - SPBSetDeviceAGC(InDevNum as Integer,State as Integer) As Integer - Pass 1 to this method to turn the AGC on. Pass 0 to this method to turn the AGC off. AGC stands for automatic gain control. Not all devices support this. (added in a1)

(Function) - SPBGetDeviceGain(InDevNum as Integer) As Integer - This method will return an integer value between 0 and 127 representing the amount of input gain used on the current device. (added in a2)

(Function) - SPBSetDeviceGain(InDevNum as Integer,Val as Integer) As Integer - This method will set the level of input gain of the current device. Use a low value for line level devices and a hight value for mic level devices. The range for the gain is 0 - 127. (added in a1)

(Function) - SPBRecordToFile(FPath as String,InDevNum as Integer,RecSeconds as Integer,NumOfChannels as Integer,SRate as Integer,SSize as Integer,CompType as String,ASync as boolean) As Integer - This is the method that actually records the file. The parameters are as follows:
FPath as String - You first need to create the file of type AIFF using the binarystream class. Then, you pass that folderitem's path to this parameter.
InDevNum as Integer - As with all of the calls, this is the Input Device Reference Number that is returned from the SPBOpenDevice function.
RecSeconds as Integer - This is the number of seconds you want to record. If you want to record an undetermined amount of time, set this number to a greater value than the maximum number of seconds you want to be able to record and then call the SPBStopRecording method to end the recording process.
NumOfChannels as Integer - The number of channels to be recorded. 1 for Mono, 2 for Stereo
SRate as Integer - The samplerate to be used (ie 44100).
SSize as Integer - The samplesize to be used (ie 16 bit).
CompType as String - The compression type. "NONE" if no compression type is to be used. As of now, only "NONE," MAC3" and "MAC6" are supported.
ASync as boolean - This allows you to choose between asynchronous recording and synchronous recording. True for Asynchronous, False for Synchronous. (added in a1, modified in a3 to calculate compression factor and to fix memory leak)

(Function) - SPBRecord(InDevNum as Integer, RecSeconds as Integer, NumOfChannels as Integer, SRate as Integer, SSize as Integer, CompType as String, Async as Boolean) As Integer - This method records to ram. The parameters are as follows:
InDevNum as Integer - As with all of the calls, this is the Input Device Reference Number that is returned from the SPBOpenDevice function.
RecSeconds as Integer - This is the number of seconds you want to record. If you want to record an undetermined amount of time, set this number to a greater value than the maximum number of seconds you want to be able to record and then call the SPBStopRecording method to end the recording process.
NumOfChannels as Integer - The number of channels to be recorded. 1 for Mono, 2 for Stereo
SRate as Integer - The samplerate to be used (ie 44100).
SSize as Integer - The samplesize to be used (ie 16 bit).
CompType as String - The compression type. "NONE" if no compression type is to be used. As of now, only "NONE," MAC3" and "MAC6" are supported.
ASync as boolean - This allows you to choose between asynchronous recording and synchronous recording. True for Asynchronous, False for Synchronous.

Once you've recorded into ram, it is up to you how to use the recorded data. The samples are stored in a memoryblock called "SoundBlock." SoundBlock is a global property, so once you've recorded to it, you'll have access to it from anywhere in your application. (added in b1)

(Function) - WriteSoundBlockIntoFile(FPath as String, RecSeconds as Integer, NumOfChannels as Integer, SRate as Integer, SSize as Integer, CompType as String, CompFactor as Integer) As Integer - This method will take the samples that are recorded into the "SoundBlock" memoryblock using the SPBRecord function and write them to disk as an aiff file. The reason I provided this method is so you can manipulate the sound data and then save it to disk. The parameters are as follows:
RecSeconds as Integer - This is the number of seconds you've recorded. If you are unsure of the seconds recorded, you should pass the same value that you passed to the SPBRecord function. This method will search for the end of the sound data and only write the sampled sound to disk so you aren't left with a huge audio file that is silent at the end.
NumOfChannels as Integer - The number of channels that were recorded. 1 for Mono, 2 for Stereo
SRate as Integer - The samplerate that was used (ie 44100).
SSize as Integer - The samplesize that was used (ie 16 bit).
CompType as String - The compression type. "NONE" if no compression type was used. As of now, only "NONE," MAC3" and "MAC6" are supported. (added in b1)

NOTE: I would strongly recommend that you make the "SPBSet..." calls to set the device to the desired parameters, then make the "SPBGet..." calls to determine the device's current parameters instead of just passing what you want to the record method for if you pass values that don't match what the device is set to, the recording will not be what you expect.

(Function) - SPBPauseRecording(InDevNum as Integer) As Integer - Pretty self explanatory, wouldn't you say? (added in a1)

(Function) - SPBResumeRecording(InDevNum as Integer) As Integer - Pretty self explanatory, wouldn't you say? (added in a1)

(Function) - SPBGetRecordingStatus(InDevNum as Integer) As Boolean - Returns True if the device is recording or paused. Returns False if the device is not recording and when the device finishes recording. (a5 fixed a bug that where this would return true even if it should've returned false) (added in a1 as a subroutine, changed to a function in a2)

(Function) - SPBStopRecording(InDevNum as Integer) As Integer - This is to allow the user to stop the recording process. If you are using the RecSeconds in the SPBRecordToFile method to determine how long the device will record, it is not necessary to make this call. (added in a1)

(Function) - SPBVersion() As String - Returns the short version. It returns a string value to allow for the "**X.X.X**" version format. You'll probably never need to call this. It was easy to add, so I did. ;) (added in a4)

NOTE ABOUT NON-APPLE DEVICES: Most Non-Apple Devices have their own options dialog box and you should use it to set the parameters for those devices rather than making some of these calls. I have not yet tested and debugged enough to determine how well this module will work with non-Apple devices. I can tell you I've done a bit of testing with Digidesign's 888I/O and had mixed results. Basically, things work fine at 48KHz, but do not work properly at 44.1KHz as Digi's gear uses a "SampleRate" property as well as a "ClockRate" property. I have not tested any other third party devices.

Enjoy,

Ben