```
-----
```

(: -= BASS_FX v2.4.12 =- :)

-==========

Copyright (c) 2002-2018 JOBnik! [Arthur Aminov, ISRAEL] All rights reserved!

Donate

=====

If you like BASS_FX or use it in a commercial/shareware products, then you may wish to make a donation to support its development (see the docs for info).

Thank you!

Files that you should have found in the BASS_FX "package"

Win32 version

BASS_FX.TXT This file

BASS_FX.DLL The BASS_FX module BASS_FX.CHM BASS_FX documentation

File_ID.Diz BBS description file

x64\

BASS_FX.DLL 64-bit BASS_FX module

 $C\$ C/C++ APIs and samples

BASS_FX.H BASS_FX C/C++ header file BASS_FX.LIB BASS_FX import library

x64\

BASS_FX.LIB 64-bit BASS_FX import library

SAFESEH\

BASS_FX.LIB SAFESEH compatible BASS_FX import library

bpm\ BPM example

bpm.c bpm.h bpm.rc

dsp\ DSP example

dsp.c dsp.h dsp.rc

freeverb\ Freeverb example

freeverb.c freeverb.h freeverb.rc

reverse\ Reverse example

reverse.c reverse.h reverse.rc

tempo\ Tempo example

tempo.c tempo.h tempo.rc

bin\ Precompiled examples

bpm.exe dsp.exe reverse.exe tempo.exe

Delphi\ Delphi APIs and samples
BASS_FX.PAS BASS_FX Delphi APIs Unit

BPM\ BPM example bpm.dfm main form bpm.pas main unit prjBPM.dpr project file

DSP\ DSP example dsp.dfm dsp.pas prjDSP.dpr

Reverse\ Reverse example prjRev.dpr
Reverse.dfm
Reverse.pas

Tempo\ Tempo example prjTempo.dpr tempo.dfm tempo.pas

VB\ Visual Basic APIs and samples BASS_FX.BAS BASS_FX VB APIs Module

BPM\ BPM example

frmBPM.frm main form

modBPM.bas module with some functions

prjBPM.vbp project file

DSP\ DSP example frmDSP.frm prjDSP.vbp

Reverse\ Reverse example frmREVERSE.frm prjREVERSE.vbp

Tempo\ Tempo example frmTempo.frm prjTepmo.vbp

NOTE: To run the sample EXEs, first you'll have to copy BASS_FX.DLL into the same directory as the EXEs. You'll also need BASS.DLL which can be download from the BASS website.

NOTE: To build the examples, you'll need to copy the BASS API into the same directory as the BASS_FX API.

MacOSX version

BASS_FX.TXT This file
LIBBASS_FX.DYLIB The BASS_FX module
BASS_FX.CHM BASS_FX documentation
BASS_FX.H BASS_FX C/C++ header file
MAKEFILE Makefile for all examples
MAKEFILE.IN Makefile helper macros

reverse\ Reverse example

reverse.c makefile reverse.nib

tempo\ Tempo example

tempo.c makefile tempo.nib

NOTE: To view the documentation, you will need a CHM viewer, like CHMOX

which is included in the BASS package.

NOTE: To build the examples, you'll need to copy the BASS API into the same directory as the BASS_FX API.

What's the point?

==============

BASS_FX is an extension to the BASS audio library, providing a complete set of Real-time DSP functions to start developing your own DJ software;) Developed to enable the simple and advanced developers/users to have in their applications sound effects without knowing any DSP at all!

Requirements

=========

BASS 2.4 is required, the BASS_FX module will fail to load if it is not present.

MacOSX version

OSX 10.3.9 or above is recommended. BASS_FX is compatible with both PowerPC and Intel Macs.

Using BASS FX

==========

Win32 version

To use BASS_FX with Borland C++ Builder, you'll first have to create a Borland C++ Builder import library for it. This is done by using the IMPLIB tool that comes with Borland C++ Builder. Simply execute this:

IMPLIB BASS_FXBCB.LIB BASS_FX.DLL

... and then use BASS_FXBCB.LIB in your projects to import BASS_FX.

To use BASS_FX with LCC-Win32, you'll first have to create a compatible import library for it. This is done by using the PEDUMP and BUILDLIB tools that come with LCC-Win32. Run these 2 commands:

PEDUMP /EXP BASS_FX.LIB > BASS_FXLCC.EXP BUILDLIB BASS_FXLCC.EXP BASS_FXLCC.LIB

... and then use BASS_FXLCC.LIB in your projects to import BASS_FX.

TIP: The BASS_FX.CHM file should be put in the same directory as the BASS.CHM file, so that the BASS_FX documentation can be accessed from within the BASS documentation.

MacOSX version

A separate "LIB" file is not required for OSX. Using XCode, you can simply add the DYLIB file to the project. Or using a makefile, you can build your programs like this, for example:

gcc yoursource -L. -lbass -lbass_fx -o yourprog

As with LIBBASS.DYLIB, the LIBBASS_FX.DYLIB file must be put in the same directory as the executable (it can't just be somewhere in the path). See the example makefiles.

LIBBASS_FX.DYLIB is a universal binary, with support for both PowerPC and Intel Macs. If you want PowerPC-only or Intel-only versions, the included makefile can create them for you, by typing "make ppc" or "make i386".

Latest Version

==========

The latest versions of BASS_FX & BASS can be found at these websites:

http://www.un4seen.com
http://www.jobnik.org (the home of BASS)

Copyright, Disclaimer, and all that other jazz

The BASS_FX library is free, so if anyone tries to charge you for it, kick 'em where it hurts.

This software is provided "as is", without warranty of ANY KIND, either expressed or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose. The author shall NOT be held liable for ANY damage to you, your computer, or to anyone or anything else, that may result from its use, or misuse. Basically, you use it at YOUR OWN RISK.

Usage of BASS_FX indicates that you agree to the above conditions.

You may freely distribute the BASS_FX package as long as NO FEE is charged and all the files remain INTACT AND UNMODIFIED.

All trademarks and other registered names contained in the BASS_FX package are the property of their respective owners.

History

======

These are the major (and not so major) changes at each version stage. There are ofcourse bug fixes and other little improvements made along the way too! To make upgrading simpler, all functions affected by a change to the BASS_FX interface are listed.

2.4.12 - 28/03/2018

*BASS FX:

- * Added "Windows Store (Win10-ARM/x86/x64, Win8-x86/x64, Win8-Phone-ARM)" platform support (package bass_fx24-winstore.zip).
- * Android, added "arm64-v8a" architecture support (package bass_fx24-android-arm64.zip).
- * Linux ARM, added "aarch64" architecture support.
- * OSX, removed PowerPC support, since the PowerPC architecture is no longer supported by BASS and Apple.
- * Delphi/Pascal, updated BASS_FX.PAS unit and changed "WIN32" to "MSWINDOWS".

* Tempo:

- * Updated to the latest SoundTouch version 2.0.0
- * Adjusted algorithm parameters to reduce reverberating effect at tempo slowdown.
- * Improved sound quality when using integer processing algorithm.
- * Improved sound quality when BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO == TRUE.

The new quickseek algorithm can find 99% as good results as the default full-scan mode, while the quickseek algorithm is

remarkable less CPU intensive.

* "mixtime" BASS_SYNC_END syncs are delayed until the tempo processing reaches the end (not when the source does), in the

same was as BASS_SYNC_POS syncs.

- * Fixed threading issue.
- * Miscellaneous small fixes and improvements.

* BPM:

* Updated to the latest SoundTouch version 2.0.0

* Reverse:

* Fixed small bug in reverse processing, that BASS_SYNC_END syncs aren't getting triggered on reverse streams that are

playing some files forwards.

* DSP:

- * Fixed fRate parameter in BASS_FX_BFX_AUTOWAH effect.
- * BASS_FX_BFX_PITCHSHIFT effect is also enabled for Android (armeabi-v7a, x86) and Linux ARM (hardfp), that using now

floating-point instead of fixed-point.

2.4.11.1 - 24/12/2014

*BASS FX:

- * Added support for BASS 2.4.11, BASS_DATA_FIXED flag is used in Android, WinCE and Linux ARM platforms.
- * Documentation is updated.
- * iOS, added "x86_64" simulator architecture.
- * Tempo:
- * Fixed "Segmentation fault" on x64 Linux system.
- * DSP:
- * Android, fixed "BASS_FX_BFX_ECHO4" effect as reported in this thread: http://www.un4seen.com/forum/?topic=13225.msg112373#msg112373

2.4.11 - 02/09/2014

* BASS FX:

- * Fixed an issue on OSX (The Xcode project's "Installation Directory" setting set to "@loader_path" instead of "@executable_path").
 - * Tempo/Reverse fixed a thread-safety crash.
 - * Tempo/Reverse added CTYPE info for these streams.
 - * Android, added "x86" architecture support (some effects are buggy, will be fixed soon).
 - * iOS, added "arm64" architecture support.
 - * Fixed Delphi/Pascal unit by changing "FLOAT" to "Single".
 - * Added C/C++ "freeverb" example.
- * Tempo:
 - * Updated to latest SoundTouch library version 1.8.0
- * Fixed a crash when using syncs on tempo, as described in this thread: http://www.un4seen.com/forum/? topic=15708.0
- * Added 3 interpolation algorithms to set using BASS_FX_TEMPO_ALGO_XXX flags (BASS_FX_TempoCreate):
 - * BASS FX TEMPO ALGO LINEAR
 - * BASS_FX_TEMPO_ALGO_CUBIC (default)
 - * BASS_FX_TEMPO_ALGO_SHANNON
- * DSP:
 - * Fixed a bug in BASS_FX_BFX_VOLUME_ENV effect with the "bFollow" option on mobile devices, as described in this thread http://www.un4seen.com/forum/?topic=15866
 - * Added new effects:
 - * BASS_FX_BFX_PITCHSHIFT, that uses FFT for its pitch shifting while maintaining duration.
 - * BASS FX BFX FREEVERB, a reverb effect.

2.4.10.1 - 05/06/2013

- * BPM/Beat:
 - * Fixed a crash when not using BPMPROGRESSPROC callback
- * Added a check for BPMPROC/BPMBEATPROC callbacks, if not available, returns BASS_ERROR_ILLPARAM error code

2.4.10 - 02/06/2013

* BASS FX:

* Please see DSP and BPM sections for decprecated effects/functions.

To not break compatibility with BASS 2.4 version, these effects and functions will still remain in BASS FX,

but are removed from documentations and will be totally removed from BASS_FX in the future.

- * BASS_ERROR_FX_NODECODE error code *removed*, will return BASS_ERROR_DECODE instead (Tempo/Reverse/BPM/Beat)
- * BASS_ERROR_FX_BPMINUSE error code *removed*, will return BASS_ERROR_ALREADY instead (BPM/Beat)
 - * Tempo and BPM functions updated to latest SoundTouch library version 1.7.1
 - * Added BASS_BFX_Linear2dB and BASS_BFX_dB2Linear macros to headers, for convenience.

* Tempo:

- * Multi-channel support is added, but:
 - * No SSE optimizations
- * BASS_ATTRIB_TEMPO_OPTION_USE_AA_FILTER is by default set to FALSE on iOS, Android, WinCE and Linux ARM platforms for lower CPU usage
 - * Not part of SoundTouch library, sources will be sent to Olli Parviainen after BASS_FX release
- * Sound quality improvements
- * Improved output sound stream duration to match better with ideal duration
- * Fixed BASS_ERROR_UNKNOWN issue with Windows 8 x64, posted in this thread: http://www.un4seen.com/forum/?topic=14499.0

* BPM:

- * BASS_FX_BPM_Translate and all of its options, BASS_FX_BPM_TRAN_xxx, are *deprecated*
- * BPMPROCESSPROC *renamed* to BPMPROGRESSPROC
- * BPM example is updated to fit above changes
- * Tuned detection algorithm
- * Fixed detection bug in Android, WinCE & Linux ARM platforms, for returning odd values
- * Fixed percents bug in BPMPROGRESSPROC
- * Changed default min/max BPM window to SoundTouch's 29/200

* Beat:

* Fixed regression since version 2.4.8 in BASS_FX_BPM_BeatDecodeGet function, that would free a "chan" when detection is completed, as described in this thread: http://www.un4seen.com/forum/?topic=2181.msg102805#msg102805

* DSP:

- * Ported all effects to Android, WinCE & Linux ARM platforms
- * BASS FX BFX ROTATE:

```
added new structure "BASS_BFX_ROTATE" with params:
    "fRate" - set the rotation rate/speed in Hz between channels
    "lChannel" - multi-channel support, only for even number of channels
 * BASS FX BFX ECHO4:
    added new effect and structure "BASS_BFX_ECHO4" with params:
    "fDryMix" - unaffected signal mix
    "fWetMix" - affected signal mix
    "fFeedback" - output signal to feed back into input
    "fDelay" - delay seconds
    "bStereo" - even channels are echoed to each other if enabled
    "lChannel" - multi-channel support
 * BASS_FX_BFX_ECHO
                           - *deprecated*, use BASS_FX_BFX_ECHO4
                            - *deprecated*, use BASS FX BFX ECHO4
 * BASS FX BFX ECHO2
 * BASS_FX_BFX_ECHO3
                            - *deprecated*, use BASS_FX_BFX_ECHO4
 * BASS_FX_BFX_REVERB
                             - *deprecated*, use BASS_FX_BFX_ECHO4 with fFeedback enabled
                            - *deprecated*, use BASS FX BFX CHORUS
 * BASS FX BFX FLANGER
 * BASS_FX_BFX_COMPRESSOR - *deprecated*, use BASS_FX_BFX_COMPRESSOR2
                          - *deprecated*, use BASS_FX_BFX_BQF with
 * BASS FX BFX APF
BASS_BFX_BQF_ALLPASS filter
 * BASS FX BFX LPF
                         - *deprecated*, use 2x BASS_FX_BFX_BQF with
BASS BFX BQF LOWPASS filter and appropriate fQ values
* BASS FX:
```

2.4.9 - 16/01/2013

- * WinCE version introduced (package bass_fx24-ce.zip) currently doesn't include most BASS_FX_BFX_xxx effects.
- * Linux ARM version introduced (package bass fx24-linux-arm.zip) currently doesn't include most BASS_FX_BFX_xxx effects.

* DSP:

* Added more effects to Android, WinCE & Linux ARM ports:

BASS_FX_BFX_PEAKEQ

BASS_FX_BFX_MIX

BASS FX BFX VOLUME ENV

- * Tempo and Reverse:
- * According to this request: http://www.un4seen.com/forum/?topic=13910 Added support for DECODETO option.
- * Tempo:
- * iOS, WinCE & Linux ARM: Enabled the BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO option on tempo

streams by default for lower CPU usage. See docs on how to disable it.

* iOS: Added armv7s architecture support.

2.4.8 - 31/07/2012

- *BASS FX:
- * Android version introduced (package bass_fx24-android.zip) currently doesn't include most BASS_FX_BFX_xxx effects.
- * Delphi/Pascal unit: changed "user" param from "DWORD" to "Pointer"

* BPM:

- * Added "user" param to BASS_FX_BPM_DecodeGet and BPMPROCESSPROC (you have to recompile your project).
- * BPM example is updated to fit above changes.
- * According to this request: http://www.un4seen.com/forum/?topic=13319 Added support for BPM detection from the current position with BASS_FX_BPM_DecodeGet (startSec<0).
- * Beat:
- * Couple of little fixes in beat detection.
- * Tempo:
- * Android: Enabled the BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO option on tempo streams by default for lower CPU usage. See docs on how to disable it.

2.4.7.1 - 01/07/2011

- * BASS_FX:
 - * Documentation updated.
- * DSP:
- * Fixed a small issue in BASS_FXGetParameters for BASS_FX_BFX_VOLUME effect, as it would change the lChannel value

when the global volume (lChannel=0) is requested.

- * OSX:
 - * x86_64 architecture support

2.4.7 - 07/04/2011

- * BASS_FX:
- * Delphi unit updated: BASS_BFX_ENV_NODE = record, changed to BASS_BFX_ENV_NODE = packed record
- * DSP:
- * Fixed bug in BASS_FX_BFX_VOLUME_ENV, being applied slightly early when "bFollow" is enabled.
- * BASS_FX_BFX_DAMP made effect parameter setting a bit more convenient. If fGain < 0 then leave the current value.
- * iOS:

- * Added armv7 architecture support.
- * Combined the Device army6/7 architectures and Simulator libraries into a single file.

2.4.6 - 27/07/2010

- * BASS FX:
 - * Tempo and BPM updated to SoundTouch 1.5.1pre
 - * iPhone version introduced (package bass_fx24-iphone.zip)

NOTE: Since iPhone supports only static libraries, it isn't possible for static libraries to declare their dependencies,

so that needs to be done in the app's project instead, eg. by adding "-lstdc++" in the "Other Linker Flags".

- * Tempo:
 - * Updated automatic parameter adjustment logic to reduce tone wobbling at large tempo changes.
- * Retired 3DNow! optimization support as 3DNow! is nowadays obsolete and assembler code is nuisance to maintain.

* BPM:

- * Improved BPM detection.
- * Added automatic cutoff threshold adaptation to beat detection routine to better adapt BPM calculation to different types of music.
 - * Fixed bug in BPMPROCESSPROC percents, when endSec is greater than stream's length.
- * Reverse:
 - * Fixed bug with getting the position when using a large output buffer (BASS_CONFIG_BUFFER).
 - * Fixed bug in BASS ChannelGetPosition, could return an incorrect position.

2.4.5 - 18/12/2009

* DSP: Added new effect, BiQuad filters, BASS FX BFX BQF with filter types:

BASS BFX BQF LOWPASS

BASS BFX BQF HIGHPASS

BASS_BFX_BQF_BANDPASS

BASS_BFX_BQF_BANDPASS_Q

BASS_BFX_BQF_NOTCH

BASS_BFX_BQF_ALLPASS

BASS BFX BQF PEAKINGEQ

BASS_BFX_BQF_LOWSHELF

BASS_BFX_BQF_HIGHSHELF

* Tempo:

- * Tempo processing bypassed when tempo/pitch set to 0
- * Couple of improvements:
- * Added normalization to correlation calculation
- * Heuristic that weights center of the processing window

2.4.4.1 - 29/04/2009

* Tempo: Fixed a bug that could cause a stream to end slightly early.

2.4.4 - 28/03/2009

* BASS_FX:

* Added: Linux x64 build in bass_fx24-linux.zip package.

* Tempo:

- * Updated to most latest SoundTouch library 1.4.1
- * Improved sound quality by automatic calculation of time stretch algorithm processing parameters according to tempo setting.
- * Added new BASS_ATTRIB_TEMPO_OPTION_PREVENT_CLICK, to prevent click when samplerate/pitch crosses the default value during processing.

 Default is FALSE as this involves slight sound quality compromise.

* BPM/Beat:

* Fixed a small bug of internally called SETPOS sync.

2.4.3.1 - 07/01/2009

- * BASS_FX: Delphi/Pascal unit updated to handle both Windows and Linux
- * Tempo: Fixed a small bug in processing with BASS_SAMPLE_LOOP flag

2.4.3 - 24/12/2008

* BASS_FX:

- * Mixtime POS syncs are now triggered when the specified position is rendered in the tempo/reverse stream (rather than when it is decoded from the source).
- * Linux version introduced (package bass_fx24-linux.zip) // examples will follow

* Tempo:

* Corrected BASS ATTRIB TEMPO FREQ min/max values.

* DSP:

- * Added new volume effect: BASS_FX_BFX_VOLUME_ENV, a volume effect using an envelope.
- * BASS_FX_BFX_APF, BASS_FX_BFX_ECHO2, BASS_FX_BFX_ECHO3: fDelay 6 seconds limit removed.

* BPM:

* Improved the peak detection algorithm so that it wouldn't incorrectly report too slow harmonic beats if they are close to the true base beat.

2.4.2 - 16/08/2008

* BASS FX:

* Some processing functions optimized for speed.

* DSP:

* Added new compressor effect BASS_FX_BFX_COMPRESSOR2
For compatibility issues new compressor will replace old one in version 2.5

2.4.1 - 28/06/2008

- * Tempo:
- * Fixed a bug in BASS_ChannelGetPosition, that would return a lower position than it should with a decoding tempo stream.
- * BPM:
- * Multi-channel support.
- * Fixed a bug in BASS_FX_BPM_DecodeGet, that would return 0 if using the same handle and endSec for both Callback and Decode BPM.

2.4.0.2 - 17/04/2008

- * BPM: fixed another bug in BPMPROC
- * Tempo: fixed a bug not allowing changing BASS_ATTRIB_TEMPO_OPTION_xxx

2.4.0.1 - 06/04/2008

* BPM: fixed a bug in BPMPROC

2.4 - 02/04/2008

- * BASS_FX:
- * Updated to BASS 2.4
- * More integrated with BASS plug-in system.
- * Added a function BASS_FX_GetVersion
- * BASS_FX_CONFIG_DSP_RESET *removed*
- * Error codes and names changed
- * To be able to link with BASS_FX, you'll have to call BASS_FX_GetVersion function (or any other function from BASS_FX.DLL) or load it dynamically using LoadLibrary("bass_fx.dll")
- * Win64 version introduced (package bass fx24-x64.zip)

* DSP:

- * Effect names, structure names, flags and chain order are changed e.g: BASS_FX_DSPFX_PHASER -> BASS_FX_BFX_PHASER BASS_FX_DSPPHASER -> BASS_BFX_PHASER
- * All functions integrated with BASS FX functions and the usage is the same as with BASS DX8/DMO effects:

BASS_FX_DSP_Set *removed* (use BASS_ChannelSetFX)

BASS_FX_DSP_SetParameters *removed* (use BASS_FXSetParameters)

BASS FX DSP GetParameters *removed* (use BASS FXGetParameters)

BASS_FX_DSP_Reset *removed* (use BASS_FXReset)

BASS_FX_DSP_Remove *removed* (use BASS_ChannelRemoveFX)

- * It is possible now to apply an effect more than once on the same channel.
- * BASS_FX_BFX_SWAP *removed* (use BASS_FX_BFX_MIX)
- * BASS_FX_BFX_S2M:

Updated to support multi-channel and renamed to BASS_FX_BFX_MIX BASS_FX_BFX_MIX supports Swap, remap and mixing channels together.

- * BASS_FX_BFX_PEAKEQ:
- * fFreq param *removed*

(use 'oldcenter*freq/oldfreq' to update the fCenter after changing the Samplerate)

- * Max fCenter updated from less than 1/3 to 1/2 of info.freq
- * Take a look at DSP example to know how to increase the number of bands
- * BASS FX BFX LPF:
- * fFreq param *removed* (adjust fCutOffFreq param when needed)
- * Tempo:
- * Support for all source sync types
- * Funtions integrated with BASS attribute system (BASS_ChannelSet/GetAttribute):

BASS_FX_TempoSet *removed*

BASS_FX_TempoGet *removed*

BASS_FX_TempoSetOption *removed*

BASS_FX_TempoGetOption *removed*

* Tempo attributes:

BASS_ATTRIB_TEMPO

BASS_ATTRIB_TEMPO_PITCH

BASS_ATTRIB_TEMPO_FREQ

* Option attributes:

BASS_ATTRIB_TEMPO_OPTION_USE_AA_FILTER

BASS ATTRIB TEMPO OPTION AA FILTER LENGTH

BASS ATTRIB TEMPO OPTION USE QUICKALGO

BASS ATTRIB TEMPO OPTION SEQUENCE MS

BASS_ATTRIB_TEMPO_OPTION_SEEKWINDOW_MS

BASS_ATTRIB_TEMPO_OPTION_OVERLAP_MS

* Reverse:

- * Support for all source sync types
- * Funtions integrated with BASS attribute system (BASS_ChannelSet/GetAttribute):

BASS FX ReverseSetDirection *removed*

BASS FX ReverseGetDirection *removed*

* Reverse attribute:

BASS_ATTRIB_REVERSE_DIR

* BPM:

- * Seconds changed from "float" to "double"
- * Callback "user" parameters changed to pointers: BASS_FX_BPM_CallbackSet / BPMPROC
- * Translation names changed:

```
BASS_FX_BPM_X2 -> BASS_FX_BPM_TRAN_X2
BASS_FX_BPM_2FREQ -> BASS_FX_BPM_TRAN_2FREQ
BASS_FX_BPM_FREQ2 -> BASS_FX_BPM_TRAN_FREQ2
BASS_FX_BPM_2PERCENT -> BASS_FX_BPM_TRAN_2PERCENT
BASS_FX_BPM_PERCENT2 -> BASS_FX_BPM_TRAN_PERCENT2
```

* Beat:

- * Multi-channel support
- * Seconds changed from "float" to "double"
- * "cutofffreq" param renamed to "centerfreq"
- * Callback "user" parameters changed to pointers: BASS_FX_BPM_BeatCallbackSet / BPMBEATPROC BASS_FX_BPM_BeatDecodeGet / BPMBEATPROC

2.3.0.4 - 30/10/07

- * DSP:
- * Fixed: a bug in BASS_FX_DSPFX_DAMP to avoid trying to amplify silence data
- * Fixed: a bug in BASS_FX_DSPFX_PEAKEQ to check illegal Center Frequencies (fCenter)
- * Tempo:
- * Fixed: a bug that would prevent using a BASS_SYNC_MESSAGE sync
- * Fixed: a bug that would prevent triggering a BASS_SYNC_END sync
- * Reverse:
- * Fixed: a bug that would prevent using a BASS_SYNC_MESSAGE sync

2.3.0.3 - 08/08/2007

- * BASS_FX:
- * Fixed: a DEP crashing bug.
- * Added: a Config option, to reset DSPs, BPM/Beat Callbacks when position is set. use BASS_FX_CONFIG_DSP_RESET with BASS_Set/GetConfig function.
- * DSP:
- * Fixed: bugs in functions:

BASS_FX_DSP_Set and BASS_FX_DSP_Remove

* Fixed: bugs in DSP effects:

BASS FX DSPFX PEAKEQ

BASS_FX_DSPFX_ECHO

BASS_FX_DSPFX_ECHO2

BASS FX DSPFX REVERB

BASS_FX_DSPFX_VOLUME

BASS_FX_DSPFX_DAMP

* Removed: DSPFX.TXT file, please check the docs for DSP effects/DSP Values

* TEMPO:

- * Fixed: a floating-point bug when calling BASS_FX_TempoCreate
- * Fixed: POS SYNCs to be more accurate

* BPM:

* Fixed: a bug in BASS_FX_BPM_Free and BASS_FX_BPM_BeatFree that would sometimes release the source channnel as well without using BASS_FX_FREESOURCE flag.

* REVERSE:

- * Fixed: a bug that, if you would set the direction to forward before starting playback, the position would keep counting from the end.
- * Updated: When reaching the end of the stream, changing the direction will now reset the stream, so that it can be played again.
- * Fixed: POS SYNCs to be more accurate

2.3.0.2 - 09/04/2007

* BEAT:

* Added Beat position detection for decoded streams BASS FX BPM BeatDecodeGet

* Added new functions:

BASS_FX_BPM_BeatCallbackReset BASS_FX_BPM_BeatSetParameters BASS_FX_BPM_BeatGetParameters

* REVERSE:

- * Multi-channel support
- * Added new feature to change playing direction from Reverse to Forward and vice-versa BASS_FX_ReverseSetDirection BASS_FX ReverseGetDirection

* RDM

* Fixed: one more critical bug in BPM functions

* DSP:

* BASS_FX_DSPFX_PEAKEQ:

Improved effect and reduced CPU usage Fixed: a bug, preventing using fQ if fBandwidth < 0.1f

Changed: fQ min limit to 0.1f

* BASS_FX_DSPFX_ECHO3:

Fixed: a bug in BASS FX DSP GetParameters, that would return a wrong lChannel value

* Error code: BASS_FX_ERROR_STEREO *removed* (replaced with BASS_ERROR_FORMAT)

- * TEMPO:
- * Setting functions name changed:

 $BASS_FX_TempoSetOption$

 $BASS_FX_TempoSettingGet -> BASS_FX_TempoGetOption$

BASS_FX_TEMPO_SETTING_xxx -> BASS_FX_TEMPO_OPTION_xxx

2.3.0.1 - 08/06/2006

- * New in BASS FX:
- * Added a valid parameters check for all functions.
- * BPM:
- * A little improved Beat position trigger.
- * Changing buffer content won't affect the BPM/Beat detection anymore.
- * BASS Stream/MusicFree will free the callback BPM/Beat as well.
- * BASS_FX.CHM:
- * Added a very simple example to BPMBEATPROC callback, showing how to count the BPM with just 2 beats.
- * DSP:
- * Fixed small bugs in BASS_FX_DSPFX_VOLUME effect.

2.3 - 21/05/2006

- * New in BASS_FX:
- * This version has some API changes.
- * You'll have to recompile your application to use this version.
- * =====

If you like BASS_FX or use it in a commercial/shareware products, then you may wish to make a donation to support its development (see the docs for info). =====

- * Tempo:
- * Fixed a bug, that wouldn't clear buffers if a source channel isn't seekable.
- * Removed flags:

BASS_FX_TEMPO_QUICKALGO BASS_FX_TEMPO_NO_AAFILTER

you can set these using a function below, in real-time.

* Added 2 new functions:

BASS_FX_TempoSettingSet BASS_FX_TempoSettingGet

with options (check the docs for more info about using them):

```
BASS FX TEMPO SETTING USE AA FILTER
BASS_FX_TEMPO_SETTING_AA_FILTER_LENGTH
BASS_FX_TEMPO_SETTING_USE_QUICKALGO
BASS_FX_TEMPO_SETTING_SEQUENCE_MS
BASS_FX_TEMPO_SETTING_SEEKWINDOW_MS
BASS_FX_TEMPO_SETTING_OVERLAP_MS
```

* DSP:

* Added new struct: BASS FX DSPSWAP

* Added multi-channel support and a per channel control with flags/macro: each effect with a per channel control has a new "IChannel" param (if you won't set the new param, then the effect will be affected on all channels as by default)

BASS FX DSPFX SWAP -> it's now possible not only swap, but remap as well.

BASS_FX_DSPFX_FLANGER

BASS_FX_DSPFX_VOLUME -> it's now needed to set a global volume, before boosting.

BASS FX DSPFX PEAKEQ

BASS FX DSPFX LPF

BASS FX DSPFX DAMP

BASS FX DSPFX AUTOWAH

BASS_FX_DSPFX_ECHO2

BASS_FX_DSPFX_PHASER

BASS_FX_DSPFX_ECHO3

BASS_FX_DSPFX_CHORUS

BASS FX DSPFX APF

BASS_FX_DSPFX_COMPRESSOR

BASS_FX_DSPFX_DISTORTION

* Channel flags (check the docs for channels order):

BASS_FX_DSP_CHANALL

BASS_FX_DSP_CHANNONE

BASS FX DSP CHAN1.. BASS FX DSP CHAN8

* If you have more than 8 channels (7.1), use this macro.

BASS_FX_DSP_CHANNEL_N(n)

- * Added a DENORMAL check for all effects.
- * BASS FX DSP Reset is updated for all effects.

* BPM:

* Added Real-Time Beat Position Trigger: BASS FX BPM BeatCallbackSet

BASS_FX_BPM_BeatFree

- * BASS_FX_ERROR_BPMX2 error code *removed* and *replaced* with BASS_ERROR_ALREADY
- * Fixed bugs:
- * Serious memory-leak is fixed using both options.
- * A bug that would free resources before the detecting process is finished.
- * A bug that would still continue to detect previous data even if changing file to scan, using a BASS_FX_BPM_BKGRND flag.
- * A bug that would still return BPMs out of MIN/MAX range if using BASS_FX_BPM_MULT2 flag.

2.2.0.1 - 30/11/2005

- * New in BASS_FX:
- * 8-bit support.
- * Added more DSP effect information to BASS_FX.CHM
- * DSP:
- * Automatically free DSP resources when freeing the channel.
- * Multi-channel support started with:
- * BASS FX DSPFX ECHO2
- * BASS_FX_DSPFX_ECHO3
- + more effects will be updated soon! :)
- * Some bugs fixed.
- * BPM:
- * Fixed a bug that would return BPMs out of MIN/MAX range if using BASS_FX_BPM_MULT2 flag.
- * MacOSX:
- * Samples added.

2.2 - 03/10/2005

- * New in BASS_FX:
- * Removed all DSP GPL code.
- * BASS_FX is now fully useable in commercial software, as long as credit is given.
- * BASS_FX_GetVersion() *removed* (won't load if BASS 2.2 isn't present)
- * BASS_FX_ERROR_MODLEN *removed* (replaced with BASS_ERROR_NOTAVAIL)
- * BASS_FX_ERROR_16BIT *removed* (no 16-bit only effects are left)
- * Multi_FX example *removed*
- * MacOSX port introduced
- * DSP:
- * Removed GPL FX:
- * BASS_FX_DSPFX_FLANGER2 & BASS_FX_DSPFX_CUT
- because of that the DSP chain is changed!
- * Reverse:
- * MOD playback is now supported if using BASS_MUSIC_PRESCAN flag.

* MacOSX examples will follow this week.

2.1.0.2 - 07/05/2005

- * DSP:
- * Chorus: fixed a bug, that would convert stereo to mono.
- * Low Pass Filter: fixed a bug, that would convert stereo to mono.
- * DynamicAMP: another bug fix, that would sometimes cause a total silence.
- + Added:
- * A new effect: Distortion
- * DSPFX.TXT a values to use with some effects, to achieve different effect with the same one (not using other effect/s):)

2.1.0.1 - 22/02/2005

- * New in BASS FX:
- + Added:
- * File version info.
- * Documentation file BASS_FX.CHM.
- * DSP:
- * Another DynamicAMP bug fix.
- * Added a new effect: Compressor
- * BPM:
- * Added: "User" param to Callback BPM functions.
- 2.1 27/12/2004 Happy New Year;)

.....

- * New in BASS FX:
- * No more "alpha/beta" releases! :)
- * Updated to BASS 2.1 add-on APIs, coz of that BASS_FX is not compatible with any previous versions. You'll have to make some changes in your project.
- * Full 32-bit floating-point support.
- * Sync support, "Sync & Tempo" example *removed*
- * A lot of functions/error codes removed and integrated with BASS functions/error codes.
- * New flag: BASS_FX_FREESOURCE if you want BASS_FX to free the source handle as well.
- * New error code: BASS_FX_ERROR_16BIT for Flanger 2.
- * BASS_FX_ErrorGetCode *removed* (use BASS_ErrorGetCode)
- * BASS FX Free *removed*
- * Tempo:
- * BASS_FX_TempoGetResampledHandle *removed*
- * BASS_FX_TempoStopAndFlush *removed*
- * BASS_FX_TempoFree *removed* (use BASS_StreamFree for music as well)
- *BASS FX TempoGetApproxSeconds *removed* (use BASS FX TempoGetRateRatio to calculate)
- *BASS FX TempoGetApproxPercents *removed* (use BASS FX TempoGetRateRatio to calculate)
- + New functions:

 $BASS_FX_TempoGetSource \ (get \ the \ source \ handle \ when \ needed)$

BASS_FX_TempoGetRateRatio

+ New in flags:

BASS_FX_TEMPO_QUICKSEEK *renamed* to BASS_FX_TEMPO_QUICKALGO BASS_FX_TEMPO_NO_AAFILTER *added*

* 3DNow! & SSE support.

* Reverse:

- * BASS_FX_ReverseGetReversedHandle *removed*
- * BASS FX ReverseSetPosition *removed* (use BASS ChannelSetPosition)
- * BASS_FX_ReverseFree *removed* (use BASS_StreamFree)
- * BASS_FX_ReverseCreate: "decode" param *removed* (use BASS_STREAM_DECODE flag)
- * New function: BASS FX ReverseGetSource (get the source handle when needed)

* BPM:

- + These functions are combined to one: BASS_FX_BPM_Translate
 - * BASS_FX_BPM_X2 *removed*
 - * BASS FX BPM Frequency2BPM *removed*
 - * BASS_FX_BPM_2Frequency *removed*
 - * BASS FX BPM Percents2BPM *removed*
 - * BASS_FX_BPM_2Percents *removed*
- + Use these translation options with a function above:

BASS FX BPM X2

BASS_FX_BPM_2FREQ

BASS_FX_BPM_FREQ2

BASS FX BPM 2PERCENT

BASS_FX_BPM_PERCENT2

* DSP:

- * Flanger 2 still only 16-bit, will return an error if applied to 32-bit.
- * Some DSP effects bug fixed.
- + Echo 2.1 renamed to Echo 3:
- * BASS_FX_DSPFX_ECHO21 -> BASS_FX_DSPFX_ECHO3
- * BASS FX DSPECHO21 -> BASS FX DSPECHO3
- + New effects added (more will come soon!):
- . Chorus
- . All Pass Filter

2.0 "beta 2" - 28/11/2004

- * New in BASS_FX:
- * Updated to BASS 2.1, just before releasing the official BASS_FX 2.1:)
- * Oops... again BASS_FX is a bit smaller;)

2.0 "beta 2" - 19/10/2004

^{*} DSP:

- + Updated with 32-bit floating-point support:
- . Auto Wah
- + Dynamic Amplification:
- * Fixed bug that would cause a sound mute if there're ~20+ seconds of silence. All parameters changed from "Integer/Long" to "Float/Single" and their names now starts with "f: Float/Single".
- + Not updated, yet [only 16-bit support]:
- . Flanger 2.0!
- + Fixed bugs of:
- * "Echo" & "Reverb" effects that would cause a noise clicks and a sound mute.
- * "Peaking EQ" that would cause a crash with Mono files.
- * Reverse:
- * 32-bit floating-point support.
- * Tempo:
- * Fixed bug that would cause a crash if BASS_FX_Free would be called twice.
- * WARN!NG NOTE:

Not updated effects (Flanger 2/Tempo) must not be used with 32-bit. It will crash your program and could make a very annoying noise!!! Sometimes could even crash your system until RESET!

2.0 "beta 1" - 07/09/2004

- * Really sorry for a long delay with updates!!!
- * New in BASS_FX:
- * Now supports Windows 98/98SE without "msvcp60.dll" ~392KB
- * Some DSP effects updated with 32-bit floating-point support. The updated effects could be used with 16-bit & 32-bit.
- * DSP:
- + Updated with 32-bit floating-point support:
- . Swap Channels
- . Rotate
- . Echo
- . Flanger
- . Volume Amplifier
- . Peaking EQ
- . Reverb
- . Low Pass Filter
- . Volume Cutter
- . Stereo 2 Mono
- . Echo 2.0!
- . Phaser

- . Echo 2.1!
- + Not updated, yet [only 16-bit support]:
- . Flanger 2.0!
- . Dynamic Amplification
- . Auto Wah
- * All parameters that began with "d: Double" changed to "f: Float/Single", as it was forgotten with last update.
- * BPM:
- * Updated with 32-bit floating-point support.
- * WARN!NG NOTE:

Not updated effects (DSP/Tempo/Reverse) must not be used with 32-bit. It will crash your program and could make a very annoying noise!!! Sometimes could even crash your system until RESET!

- * New in Examples:
- * Added: "Sync & Tempo"
- * Removed:
- * BASS_FX_ERROR_BASS20 error code, BASS_FX will show an error message if BASS.DLL version is below 2.0 and won't load.
- 2.0 "alpha" 4/12/2003

- * New in BASS FX:
- * Updated to BASS 2.0!
- * Version jumpted from '1.2 "beta" to '2.0 "alpha", means only BASS v2.x is supported!
- * DSP:
- * Added *priority* param to BASS_FX_DSP_Set(..) func
- * All *Double* types changed to *Float/Single*
- * New in Examples:
- * Added "Multi_FX" C/C++ only.
- * VERSION 2.0 (not "alpha") will support:
- * 32-bit floating-point including in: DSP, Tempo, BPM & Reverse.
- * Multi Channel in some DSPs.
- * Planning to release till the end of this month/year:)
- 1.2 "beta" 30/06/2003

* New in BASS_FX:

- * Not compatible with any previous BASS_FX versions, you'll have to make changes & recompile your application to use with this version!
- + Tempo, Pitch Scaling & Samplerate changers (3 at once;))
- + Functions:
- * BASS_FX_TempoCreate
- * BASS_FX_TempoSet
- * BASS_FX_TempoGet
- * BASS_FX_TempoGetApproxSeconds
- * BASS_FX_TempoGetApproxPercents
- * BASS_FX_TempoGetResampledHandle
- * BASS_FX_TempoStopAndFlush
- * BASS_FX_TempoFree
- + Two BPM Detection options:
- + Option 1 Get BPM from a Decoding channel:
- + Function
- * BASS_FX_BPM_DecodeGet
- + Option 2 Get BPM by period of time of any handle in Real-Time:
- + Functions
- * BASS_FX_BPM_CallbackSet
- * BASS_FX_BPM_CallbackReset
- + Functions to use with both options:
 - * BASS_FX_BPM_X2
 - * BASS_FX_BPM_Frequency2BPM
 - * BASS_FX_BPM_2Frequency
 - * BASS_FX_BPM_Percents2BPM
 - * BASS FX BPM 2Percents
- * BASS_FX_BPM_Free
- * New in DSP:
- * All DSP effects names has changed.
- * The index of 1st DSP effect starts from 0 and not from 1 as it was before.
- + -= DSP FXs =- added:
 - * Dynamic Amplification
 - * Stereo 2 Mono
 - * Auto Wah
 - * Echo v2.0!
 - * Phaser
 - * Echo v2.1!
- + Equalizer:
- * Added *Q* parameter.

- * Fixed some bugs :)
- + Flanger v2.0!
- * fixed bug (crashed with Mono files)
- * New in Functions:
- + Added:
- * BASS_FX_ErrorGetCode
- * Always check for any Function changes.
- * Removed all PITCH functions:
- * BASS_FX_PitchCreate
- * BASS_FX_PitchSet
- * BASS_FX_PitchGet
- * BASS_FX_PitchGetResampledHandle
- * BASS_FX_PitchStopAndFlush
- * BASS_FX_PitchFree
- * New in Examples:
- * Added: C/C++, Delphi & VB

1.1 - 02/10/2002

- * New in BASS FX:
- * Now supports 16/8-Bit Stereo/Mono.
- * Support for Multiple BASS instances.
- * A lot of BUGs fixed =)
- * New in DSP:
- + -= DSP FXs =- added:
 - * Low Pass Filter
 - * Cutter
 - * Flanger v2.0!
- + Equalizer:
- * Algorithm optimized to BiQuad.
- * Added a new parameter *egBandwidth*
- * New in Reverse:
- * Now you can add DX8 effects + change Pitch with BASS_FX_Pitch... (check the *Reverse* example).
- * New in functions:
- + Added:
- * BASS FX DSP Reset
- * BASS_FX_Free
- * Always check for any Function changes.

- * Added:
 - * Delphi APIs + Pitch Example.
 - * C/C++ Examples + corrected BASS_FX.LIB file ;)
- * Switched from MFC to Win32 DLL [MFC42.DLL ~1MB doesn't required]

1.0 - 14/06/2002

* First release

Credits

======

- * Thanks a lot to Ian Luck @ www.un4seen.com for:
 - + BASS Best Available Sound System!
 - + DSP source codes for Echo, Dynamic Amplification, Compressor and Volume Envelope
 - + Reverse playback source code
 - + SoundTouch algorithms implementation for Tempo/BPM
 - + Beat position algorithm fixes
 - + 8/16/32-bit support
 - + Fixed-point support
 - + Multi-channel support
 - + Add-on support
 - + MacOSX support
 - + Android support
- * Ian, you're the best programmer in the whole world!

Credits - API/Sample Contributors

Delphi - Roger Johansson, Alex Graham (bigjim), DJ-Chris BASS_FX.CHM - Thijs van Vliet

Bug reports, Suggestions, Comments, Enquiries, etc...

If you have any of the aforementioned please check the BASS forum (at the website)... If you can't find an answer there, you can email:

bass fx@jobnik.org

System - Desktop/PC

BASS_FX.DLL - Windows - developed and tested using:

System: Intel Core i7 Haswell 4770 3.9GHz 8MB, 16GB DDR3 1600MHz CL9

Intel Core i7 860 2.8GHz 8MB, 4GB DDR3 1600MHz CL7

VMware 10

OS : Microsoft Windows:

x86: 8, 7 Ultimate, Vista Ultimate, XP Pro SP3, 2000 Pro SP4 & 98

x64: 10 Pro, 8.1 Pro, 7 Enterprise/Ultimate, Vista Ultimate SP1, XP Pro SP1

Sound Card: RealTek HD 7.1 (onboard)

Compiler: x86: Microsoft Visual C++ v6.0 SP5 with a Processor Pack

x64: Microsoft Visual C++ 2005 v8.0

DirectX : 12, 11, 10, 9.0c and 7.0

BASS.DLL : 2.4.13.8

LIBBASS_FX.DYLIB - OSX - developed and tested using:

System : VMware 10

OS : Apple Macintosh OS X:

Intel Mac: 10.9, 10.8.2, 10.5.8 and 10.4.10

PowerPC : 10.3.9 and 10.4

Compiler : GCC 4.0.1 IDE : XCode 3.1.4 LIBBASS.DYLIB: 2.4.13.8

LIBBASS_FX.SO - Linux - developed and tested using:

System : VMware 10

OS : Ubuntu Desktop x86 and x64 v8.04

Compiler : x86 and x64: GCC 4.2.4 (g++)

IDE : Code::Blocks v8.02

LIBBASS.SO: 2.4.13.8

System - Mobile/Portable

LIBBASS_FX.A - iOS - developed and tested using:

System: VMware 10

OS : Apple Macintosh OS X: Intel Mac 10.9

Compiler: GCC 4.2 / LLVM

IDE : XCode 3.1.4 / XCode 5.0.2 for armv7s/arm64 architectures

LIBBASS.A: 2.4.13.8

LIBBASS_FX.SO - Android - developed and tested using:

System : Samsung Galaxy S7 Edge

LG G4-H815

Samsung Galaxy S2 GT-I9100

Android Virtual Device

OS : Android Lollipop 5.1

Android JB 4.1.1/2 Android ICS 4.0.3/4 Android GB 2.3.3

Compiler: Android NDK R10: GCC 4.4.8/9

IDE : Eclipse

LIBBASS.SO: 2.4.13.8

BASS_FX.DLL - Windows Store/Phone - developed and tested using:

Compiler : Microsoft Visual C++ 2015 v14.0

BASS.DLL : 2.4.13.8

BASS_FX.DLL - WinCE - developed and tested using:

System: GPS Device with CPU @ 372MHz

Pocket PC 2003 SE Emulator

OS: Windows CE 5

Windows Mobile 2003 SE version 4.21.1088

Compiler: Microsoft Visual C++ 2005 v8.0

BASS.DLL: 2.4.13.8

LIBBASS_FX.SO - Linux ARM - developed and tested using:

System : VMware 8

OS : Ubuntu Desktop x86 v11.10

Compiler : GCC: (crosstool-NG 1.15.2) 4.7.1 20120402 (prerelease)

IDE : Code::Blocks v8.02 LIBBASS.SO : 2.4.13.8

More Credits;)

- * BiQuad filters
- (c) Robert Bristow-JohnsonD
- @ http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt
- * Peaking Equalizer (BiQuad filter)
 The main source is based on Manu Webber's source code.
- http://www.un4seen.com/forum/?topic=1246.msg6484#msg6484
- * Tempo/Pitch/Rate/BPM [SoundTouch v2.0.0]
- (c) Copyright (c) 2002-2017 Olli Parviainen
- @ http://www.surina.net/soundtouch
- L LGPL license
- * Auto Wah, Chorus, Distortion, Echo (some parts from 1st algorithm) and Phaser
- (c) Copyright (c) 2000 Aleksey Smoli
- @ http://st.karelia.ru/~smlalx (offline)
- * Freeverb
- (c) Copyright (c) 2000 Jezar at Dreampoint
- @ http://www.dreampoint.co.uk
- L Public domain
- * Pitch shifting using FFT [smbPitchShift v1.2]
- (c) Copyright (c) 1999-2009 Stephan M. Bernsee <smb [AT] dspdimension [DOT] com>
- http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/
- L WOL license
- * BASS_FX is fully useable in commercial software, as long as credit is given.

* BASS FX.TXT & File ID.Diz are better viewed in DOS mode OR with - Courier - font.

Donate

There is no charge for using BASS_FX, but if you like BASS_FX or are using BASS_FX in Commercial/Shareware products, then you may wish to make a donation to support its development.

Donations can be made quickly and securely via PayPal, by clicking on this button:

Make A Donation

Or to my paypal address: admin@jobnik.org

Thank you!

BASS_FX_GetVersion

Retrieves the version of BASS_FX that is loaded.

DWORD BASS_FX_GetVersion();

Return value

The BASS_FX version. For example, 0x02040103 (hex), would be version 2.4.1.3

Remarks

There is no guarantee that a previous or future version of BASS_FX supports all the BASS or BASS_FX functions that you are using, so you should always use this function to make sure the correct version is loaded. It is safe to assume that future revisions (indicated in the LOWORD) will be fully compatible.

BASS_FX version should be identical to BASS version, only the revision changes on updates.

The BASS API includes a BASSVERSION constant, which can be used to check that the loaded BASS.DLL and BASS_FX.DLL matches the API version used, ignoring revisions.

Example

Check that the correct BASS_FX version is loaded, ignoring the revision.

```
if (HIWORD(BASS_FX_GetVersion())!=BASSVERSION) {
   // incorrect version loaded!
}
```

Check that revision 1.0 (or above) of the correct BASS_FX version is loaded.

if (HIWORD(BASS_FX_GetVersion())!=BASSVERSION // check the main version

```
|| LOWORD(BASS_FX_GetVersion())<0x100) { // check the revision
// incorrect version loaded!
}</pre>
```

${\bf BASS_ChannelSetFX}$

Sets up a DSP effect on a stream, MOD music, or recording channel.

```
HFX BASS_ChannelSetFX(
    DWORD handle,
    DWORD type,
    int priority
);
```

Parameters

handle	The channel handle a HSTREAM, HMUSIC, or HRECORD				
type	The One of the following types of effect:				
	BASS_FX_BFX_ROTATE	A channels volume ping-pong	multi- channel		
	BASS_FX_BFX_VOLUME	Volume	multi- channel		
	BASS_FX_BFX_PEAKEQ	Peaking Equalizer	multi- channel		
	BASS FX BFX MIX	Swap, remap and mix channels	multi- channel		
	BASS FX BFX DAMP	Dynamic Amplification	multi- channel		
	BASS FX BFX AUTOWAH	Auto Wah	multi- channel		
	BASS_FX_BFX_PHASER	Phaser	multi- channel		
	BASS_FX_BFX_CHORUS	Chorus/Flanger	multi- channel		
	BASS_FX_BFX_DISTORTION	Distortion	multi- channel		
	BASS_FX_BFX_COMPRESSOR2	Compressor	multi- channel		
	BASS FX BFX VOLUME ENV	Volume envelope	multi-		

BASS_FX_BFX_BQF	BiQuad filters	channel multi- channel
BASS_FX_BFX_ECHO4	Echo/Reverb	multi- channel
BASS_FX_BFX_PITCHSHIFT	Pitch shift using FFT	multi- channel
BASS_FX_BFX_FREEVERB	Reverb using "Freeverb" algorithm	multi- channel

priority The priority of the new DSP, which determines its position in the DSP chain. DSPs with higher priority are called before those with

lower priority

Return value

If successful, then the new effect's handle is returned, else 0 is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE *handle* is not a valid channel

BASS_ERROR_ILLTYPE type is invalid. Note BASS_FX must be loaded before these effects can be used (call BASS_FX_GetVersion on project load).

BASS_ERROR_FORMAT The selected effect could be applied only on stereo or mono *handle*

BASS ChannelRemoveFX, BASS FXSetParameters, BASS FXGetParameters, BASS FXReset

BASS_ChannelRemoveFX

Removes a DSP effect on a stream, MOD music, or recording channel.

```
BOOL BASS_ChannelRemoveFX(
    DWORD handle,
    HFX fx
);
```

Parameters

handle The channel handle... a HSTREAM, HMUSIC, or HRECORD

fx Handle of the effect to remove from the channel

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE At least one of *handle* and *fx* is not valid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS ChannelSetFX, BASS FXSetParameters, BASS FXGetParameters, BASS FXReset

BASS_FXSetParameters

Sets the parameters of an effect.

```
BOOL BASS_FXSetParameters(
    HFX handle,
    void *params
);
```

Parameters

handle The effect handle.

params Pointer to the parameters structure. One of:

BASS_BFX_ROTATE

BASS BFX VOLUME

BASS_BFX_PEAKEQ

BASS BFX MIX

BASS_BFX_DAMP

BASS BFX AUTOWAH

BASS_BFX_PHASER

BASS BFX CHORUS

BASS_BFX_DISTORTION

BASS BFX COMPRESSOR2

BASS BFX VOLUME ENV

BASS BFX BQF

BASS BFX ECHO4

BASS_FX_BFX_PITCHSHIFT

BASS_FX_BFX_FREEVERB

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE The *handle* is invalid

BASS_ERROR_ILLPARAMOne or more of the parameters are invalid, make sure all the values are within the valid ranges

See also

BASS_ChannelSetFX, BASS_ChannelRemoveFX, BASS_FXGetParameters, BASS_FXReset

BASS_FXGetParameters

Retrieves the parameters of an effect.

```
BOOL BASS_FXGetParameters(
    HFX handle,
    void *params
);
```

Parameters

handle The effect handle.

params Pointer to the parameters structure to fill. Depending on the type of effect, the parameters returned will be one of:

BASS BFX ROTATE

BASS_BFX_VOLUME

BASS BFX PEAKEQ

BASS_BFX_MIX

BASS BFX DAMP

BASS BFX AUTOWAH

BASS BFX PHASER

BASS_BFX_CHORUS

BASS BFX DISTORTION

BASS_BFX_COMPRESSOR2

BASS BFX VOLUME ENV

BASS BFX BQF

BASS_BFX_ECHO4

BASS_FX_BFX_PITCHSHIFT

BASS FX BFX FREEVERB

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE The *handle* is invalid
BASS_ERROR_ILLPARAM One or more of the parameters are invalid, make sure all the values are within the valid ranges

See also

BASS ChannelSetFX, BASS ChannelRemoveFX, BASS FXSetParameters, BASS FXReset

Resets the state of an effect or all effects on a channel.

```
BOOL BASS_FXReset(
   DWORD handle
);
```

Parameters

handle The effect or channel handle... a HFX, HSTREAM, HMUSIC, or HRECORD

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

This function flushes the internal buffers of the effect(s). Effects are automatically reset by <u>BASS_ChannelSetPosition</u>, except when called from a "mixtime" <u>SYNCPROC</u>.

Error codes

BASS_ERROR_HANDLE handle is not valid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS ChannelSetFX, BASS ChannelRemoveFX, BASS FXSetParameters, BASS FXGetParameters

Multi-channel

Multi-channel is supported in next effects:

BASS FX BFX ROTATE

BASS FX BFX VOLUME

BASS FX BFX PEAKEQ

BASS FX BFX MIX

BASS FX BFX DAMP

BASS FX BFX AUTOWAH

BASS FX BFX PHASER

BASS FX BFX CHORUS

BASS FX BFX DISTORTION

BASS FX BFX COMPRESSOR2

BASS_FX_BFX_VOLUME_ENV

BASS_FX_BFX_BQF

BASS FX BFX ECHO4

BASS FX BFX PITCHSHIFT

BASS FX BFX FREEVERB

Multi-channel order of each channel is as follows:

3 channels left-front, right-front, center.

4 channels left-front, right-front, left-rear/side, right-rear/side.

5 channels left-front, right-front, center, left-rear/side, right-rear/side.

6 channels left-front, right-front, center, LFE, left-rear/side, right-rear/side. **(5.1)**

8 channels left-front, right-front, center, LFE, left-rear/side, right-rear/side, left-**(7.1)** rear center, right-rear center.

Usage:

An effect supporting multi-channel has a param "lChannel", using flags:

BASS_BFX_CHANALL -1 all channels at once (as by default)

BASS_BFX_CHANNONE 0 disable an effect for all channels

BASS_BFX_CHAN1 1 left-front channel

BASS_BFX_CHAN2 2 right-front channel

BASS BFX CHAN3 4 see above info

BASS_BFX_CHAN4 8 see above info

BASS_BFX_CHAN5 16 see above info

BASS_BFX_CHAN6 32 see above info

BASS_BFX_CHAN7 64 see above info

BASS_BFX_CHAN8 128 see above info

If you have more than 8 channels (7.1), use this macro: BASS_BFX_CHANNEL_N(n)

See also

BASS_ChannelSeFX, BASS_FXSetParameters, BASS_FXGetParameters

BASS_BFX_ROTATE structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of rotate effect.

```
typedef struct {
    float fRate;
    int lChannel;
} BASS_BFX_ROTATE;
```

Members

fRate Rotation rate/speed in Hz (A negative rate can be used for reverse

direction)

lChannel The affected channels using BASS_BFX_CHANxxx flags

Info

This is a volume rotate effect between even channels, just like 2 channels playing ping-pong between each other.

See also

BASS_BFX_VOLUME structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a volume effect.

```
typedef struct {
  int lChannel;
  float fVolume;
} BASS_BFX_VOLUME;
```

Members

lChannel Affected channels using BASS_BFX_CHANxxx flags or 0 for global

volume control

fVolume [0..1..n] linear

Remarks

lChannel needs to be set before a <u>BASS_FXGetParameters</u> call to tell it what volume level to return.

To set a new global volume, set lChannel = 0

See also

BASS_BFX_PEAKEQ structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a peaking eq effect (BiQuad filter).

```
typedef struct{
  int lBand;
  float fBandwidth;
  float fQ;
  float fCenter;
  float fGain;
  int lChannel;
} BASS_BFX_PEAKEQ;
```

Members

lBand	Band number	[0n]
fBandwidth	In octaves - fQ is not in use (Bandwidth has a priority over fQ)	[0.1<10]
fQ	Quality Factor, the EE kinda definition (linear) (if Bandwidth is not in use)	[01]
fCenter	Center frequency, in Hz	[1Hz <info.freq 2]<="" td=""></info.freq>
fGain	Gain, in dB	[-15dB0+15dB](can be above/below these limits)
lChannel	The affected channels using BASS_BFX_CHANxxx flags	

Info

This is an implementation of BiQuad Peaking Equalizer filter.

A Peaking Equalizer boosts (or reduces) at the set frequency (called the "center frequency") and a boosts band of frequencies around the center frequency by a similar amount.

Good explanation can be read here: http://en.wikiaudio.org/Equalization_filter

Remarks

The "lBand" parameter needs to be set before calling <u>BASS_FXGetParameters</u> function.

See also

BASS ChannelSetFX, BASS FX BFX BQF

BASS_BFX_MIX structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of mix effect.

```
typedef struct {
  int *lChannel;
} BASS_BFX_MIX;
```

Members

lChannel An array of channels to mix using BASS_BFX_CHANxxx flag/s (lChannel[0] is left channel...)

Remarks

By default all lChannel[n] indexes are set to BASS_BFX_CHANxxx channels order, means none of the channels are affected.

Info

This effect is providing the ability to mix, swap or remap channels.

See also

BASS_BFX_DAMP structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a dynamic amplification effect.

```
typedef struct {
    float fTarget;
    float fQuiet;
    float fRate;
    float fGain;
    float fDelay;
    int lChannel;
} BASS_BFX_DAMP;
```

Members

fTarget	Target volume level	[0<1] linear
fQuiet	Quiet volume level	[01] linear
fRate	Amplification adjustment rate	[01] linear
fGain	Amplification level	[01n] linear
fDelay	Delay in seconds before increasing level	[0n] linear
lChannel	The affected channels using BASS_BFX_CHANxxx flags	

See also

BASS_BFX_AUTOWAH structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of an auto wah effect.

```
typedef struct {
    float fDryMix;
    float fWetMix;
    float fFeedback;
    float fRate;
    float fRange;
    float fFreq;
    int lChannel;
} BASS_BFX_AUTOWAH;
```

Members

fDryMix	Dry (unaffected) signal mix	[-2+2]
fWetMix	Wet (affected) signal mix	[-2+2]
fFeedback	Output signal to feed back into input	[-1+1]
fRate	Rate of sweep in cycles per second	[0<<10]
fRange	Sweep range in octaves	[0<<10]
fFreq	Base frequency of sweep range	[0<1000]
lChannel	The affected channels using BASS_BFX_CHANxxx	
	flags	

Info

The effect implements the auto-wah by using 4-stage phaser effect which moves a peak in the frequency response up and down the frequency spectrum by amplitude of input signal.

The fDryMix is the volume of input signal & the fWetMix is the volume of delayed signal. The fFeedback sets feedback of auto wah (phaser). The fRate and fRange control how fast and far the frequency notches move. The fRate is the rate of sweep in cycles per second, fRange is the width of sweep in octaves. And the fFreq is the base frequency of sweep.

Examples

	Dry Wet Feedbac	Dry Wet Feedback Rate Range Free				
Slow Auto Wah	0.5001.5000.5	2.0	4.3	50.0		
Fast Auto Wah	0.5001.5000.5	5.0	5.3	50.0		
Hi Fast Auto Wah	0.5001.5000.5	5.0	4.3	500.0		

See also

BASS_BFX_PHASER structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a phaser effect.

```
typedef struct {
   float fDryMix;
   float fWetMix;
   float fFeedback;
   float fRate;
   float fRange;
   float fRange;
   float fFreq;
   int lChannel;
} BASS_BFX_PHASER;
```

Members

fDryMix	Dry (unaffected) signal mix	[-2+2]
fWetMix	Wet (affected) signal mix	[-2+2]
fFeedback	Output signal to feed back into input	[-1+1]
fRate	Rate of sweep in cycles persecond	[0<<10]
fRange	Sweep range inoctaves	[0<<10]
fFreq	Base frequency of sweep range	[0<1000]
lChannel	The affected channels using BASS_BFX_CHANxxx	
	flags	

Info

Phasers use an internal low frequency oscillator to automatically move notches in the frequency response up and down the frequency spectrum. An important difference between phasing and flanging is that phasers space these notches evenly across the frequency spectrum, while the notches in flanging and chorus are harmonically (musically) related. You don't hear the notches as such (because they are the frequencies that are removed). What you hear is the resulting frequency peaks between these notches. Phasing works by mixing the original signal with one that is phase shifted over the frequency spectrum. For example, a four stage phaser signal (such as this) could be from 0 degrees at

100Hz, shifted to 720 degrees at 5Khz (these extremes are not quite possible practically, but are near enough to explain the effect). This is how the term phase shifter comes about. A 4 stage phaser has 2 notches with bass response, a central peak, and treble response. By using resonance to enhance the central peak, you can get a sound similar to an automatic wah. Using a phaser with lots of stages and setting the resonance high can give a sound similar to flanging, although they are really quite different.

The fDryMix is the volume of input signal & the fWetMix is the volume of delayed signal. The fFeedback sets feedback of phaser. The fRate and fRange control how fast and far the frequency notches move. The fRate is the rate of sweep in cycles per second, fRange is the width of sweep in octaves. And the the fFreq is the base frequency of sweep.

Examples

	Dry	Wet	Feedback	Rate	Range	Freq
Phase shift	0.999	0.999	0.0	1.0	4.0	100.0
Slow invert phase shift with feedback	0.999	0.999	0.6	0.2	6.0	100.0
Basic phase	0.999	0.999	0.0	1.0	4.3	50.0
Phase w/ FB	0.999	0.999	0.6	1.0	4.0	40.0
Med. phase	0.999	0.999	0.0	1.0	7.0	100.0
Fast phase	0.999	0.999	0.0	1.0	7.0	400.0
Invert w/ invert FB	0.999	0.999	-0.2	1.0	7.0	200.0
Tremolo Wah	0.999	0.999	0.6	1.0	4.0	60.0

See also

BASS_BFX_CHORUS structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a chorus/flanger effect.

```
typedef struct {
  float fDryMix;
  float fWetMix;
  float fFeedback;
  float fMinSweep;
  float fMaxSweep;
  float fRate;
  int lChannel;
} BASS_BFX_CHORUS;
```

Members

fDryMix	Dry (unaffected) signal mix	[-2+2]
fWetMix	Wet (affected) signal mix	[-2+2]
fFeedback	Output signal to feed back into input	[-1+1]
fMinSweep	Minimum delay in ms	[0<6000]
fMaxSweep	Maximum delay in ms	[0<6000]
fRate	Rate in ms/s	[0<1000]
lChannel	The affected channels using BASS_BFX_CHANxxx	
	flags	

Info

True vintage chorus works the same way as flanging. It mixes a varying delayed signal with the original to produce a large number of harmonically related notches in the frequency response. Chorus uses a longer delay than flanging, so there is a perception of "spaciousness", although the delay is too short to hear as a distinct slap-back echo. There is also little or no feedback, so the effect is more subtle.

The fDryMix is the volume of input signal & the fWetMix is the volume of delayed signal. The fFeedback sets feedback of chorus. The fRate, fMinSweep

and fMaxSweep control how fast and far the frequency notches move. The fRate is the rate of delay change in millisecs per sec, fMaxSweep-fMinSweep is the range or width of sweep in ms.

Examples

Dry Wet Feedback Min Max Rate Flanger 1.0 0.35 0.5 1.0 5.0 1.0 Exaggerated chorus leads to multiple pitch shifted voices 0.7 0.25 0.5 1.0 200.0 50.0 Motocycle 0.9 0.45 0.5 1.0 100.0 25.0 Devil 0.9 0.35 0.5 1.0 50.0 200.0 Who say that there're not many voices? 0.9 0.35 0.5 1.0 400.0 200.0 Back chipmunk 0.9 -0.2 0.5 1.0 400.0 400.0 Water 0.9 -0.4 0.5 1.0 2.0 1.0 This is the airplane 0.3 0.4 0.5 1.0 10.0 5.0

See also

BASS_BFX_DISTORTION structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a distortion effect.

```
typedef struct {
    float fDrive;
    float fDryMix;
    float fWetMix;
    float fFeedback;
    float fVolume;
    int lChannel;
} BASS_BFX_DISTORTION;
```

Members

fDrive	Distortion drive	[05]
fDryMix	Dry (unaffected) signal mix	[-5+5]
fWetMix	Wet (affected) signal mix	[-5+5]
fFeedback	Output signal to feed back into input	[-1+1]
fVolume	Distortion volume	[0+2]
lChannel	The affected channels using BASS_BEX_CHANxxx flags	

Info

Similar to Pre/Post Gain & Drive controls on amps. They were first introduced as a trick to added color to a guitar's tone. Usually produced back then by turning the amp all the way up, or slightly pulling out a tube from its socket. These tones are now today referred to as Overdrives. Today, there are an almost infinite variety of these effects, and they range in 3 classes: Distortions, Fuzz, and Overdrive. One common feature to mostly all of these types of pedals is a volume and drive (also noted as distortion, fuzz, gain, ...etc.). Overdrives are usually a sustain and volume boosting pedal. Used by more traditional rock and country bands. Next are the Distortions, which range from punk style to death metal screams. They are similar to Overdrives, but have more buzzing quality to them. Finally, there are the Fuzzes which are more distorted than distortions, but are more mellow and compressed sounding. Many pedals can be confused within

these names, and some may be named other than what they are. Towards the end of this era, the back-to-back diode pair became popular as a technique to provide soft clipping (with germanium diodes) and hard clipping (with silicon diodes). Today, overdrive effects usually means soft clipping, where gain is reduced beyond the clipping point, while distortion usually means hard clipping, where the level is fixed beyond the clipping point. Distortion is a little harder sound, good for rock, while overdrive gives a more natural sound.

The fDrive controls the amount of overdrive. The fVolume to balance the effect volume with the bypassed level. It can also be used to boost the signal for solos. The fDryMix is the volume of input signal & the fWetMix is the volume of distorted signal. The fFeedback sets feedback of distortion.

No	ormal Si	<u>n Wave</u>			<u>Wav</u>	<u>e di</u>	storted by algorithm
			*				
			* >	k			
**	***			* *			
*	*		*	* *	:		
*	*		*	* *	*		
*	*		*	* *	*		
*	*		*	* *	*		
*	*		*	*	*		
	*		_			*	
	*	*			*	*	*
	*	*			*	* *	*
	*	*			*	* *	*
	*	*			*	* *	*
	*	*			*	* *	
	**	***				* *	
					* *	k	
					*		

Examples

	Drive	Dry	Wet	Feedback	Volume
Hard Distortion	1.0	0.0	1.0	0.0	1.0
Very Hard Distortion	5.0	0.0	1.0	0.1	1.0
Medium Distortion	0.2	1.0	1.0	0.1	1.0

Soft Distortion 0.0 -2.95-0.05-0.18 0.25

See also

BASS_BFX_COMPRESSOR2 structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a compressor effect.

```
typedef struct {
    float fGain;
    float fThreshold;
    float fRatio;
    float fAttack;
    float fRelease;
    int lChannel;
} BASS_BFX_COMPRESSOR2;
```

Members

fGain	Output gain of signal after compression. The default value is 0 dB	[-60+60] dB
fThreshold	Point at which compression begins. The default value is -15 dB	[-600] dB
fRatio	Compression ratio. The default value is 3, which means 3:1 compression	[1n]
fAttack	Time before compression reaches its full value. The default value is 10 ms	[0.011000] ms
fRelease	Speed at which compression is stopped after input drops below fThreshold. The default value is 200 ms	
lChannel	The affected channels using BASS_BFX_CHANxxx flags	

Info

Compressors are commonly used in recording to control the level, by making loud passages quieter, and quiet passages louder. This is useful in allowing a vocalist to sing quiet and loud for different emphasis, and always be heard clearly in the mix. Compression is generally applied to guitar to give clean sustain, where the start of a note is "squashed" with the gain automatically increased as the not fades away. Compressors take a short time to react to a

picked note, and it can be difficult to find settings that react quickly enough to the volume change without killing the natural attack sound of your guitar.

See also

BASS_BFX_VOLUME_ENV structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a volume envelope effect.

```
typedef struct {
  int lChannel;
  int lNodeCount;
  const struct BASS_BFX_ENV_NODE *pNodes;
  BOOL bFollow;
} BASS_BFX_VOLUME_ENV;
```

Members

lChannel The affected channels using BASS_BFX_CHANxxx flags

INodeCount Number of nodes

pNodes The nodes

bFollow FALSE = process envelope from start to finish, TRUE = follow the

position of the channel

Note

There must be at least 1 node

See also

BASS_BFX_ENV_NODE structure

Used with volume envelope effect.

```
typedef struct {
  double pos;
  float val;
} BASS_BFX_ENV_NODE;
```

Members

pos Node position in seconds (1st envelope node must be at position 0)

val Node value at the position

See also

BASS_BFX_BQF structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a biquad filters effect.

```
typedef struct{
  int lFilter;
  float fCenter;
  float fGain;
  float fBandwidth;
  float fQ;
  float fS;
  int lChannel;
} BASS_BFX_BQF;
```

Members

```
BASS_BFX_BQF_xxx filter One of the following filter types:
lFilter
         types
                                  BASS_BFX_BQF_LOWPASS
                                  BASS_BFX_BQF_HIGHPASS
                                   BASS_BFX_BQF_BANDPASS
                                                              \mathbf{C}
                                                              0
                                                              p
                                                              g
                                                              C
                                 BASS_BFX_BQF_BANDPASS_Qs
                                                              g
                                                             p
                                                             g
(
                                   BASS_BFX_BQF_NOTCH
                                   BASS_BFX_BQF_ALLPASS
                                  BASS_BFX_BQF_PEAKINGEQ
                                   BASS_BFX_BQF_LOWSHELF
```

BASS_BFX_BQF_HIGHSHELF

fCenter Cutoff (central) frequency in [1Hz..<info.freq/2]

Hz

fGain Used only for PEAKINGEQ [-15dB...0...+15dB] (can be above/below th

and Shelving filters in dB limits)

fBandwidth Bandwidth in octaves (fQ is [0.1.....<10]

not in use (fBandwidth has a

priority over fQ))

(between -3 dB frequencies for BANDPASS and NOTCH

or between midpoint

(fGgain/2) gain frequencies

for PEAKINGEQ)

fQ The EE kinda definition [0.1..........1]

(linear) (if fBandwidth is not

in use)

fS A "shelf slope" parameter [0.1.....1]

(linear) (used only with

Shelving filters)

when fS = 1, the shelf slope is as steep as you can get it and

remain monotonically

increasing or decreasing gain

with frequency.

lChannel The affected channels

using BASS_BFX_CHANxxx

flags

Info:

BiQuad filters are second-order recursive linear filters.

BASS_BFX_BQF_LOWPASS:

A **low-pass filter** is a filter that passes low-frequency signals but attenuates (reduces the amplitude of) signals with frequencies higher than the fCenter frequency. The actual amount of attenuation for each frequency varies from filter to filter. It is sometimes called a **high-cut filter**, or **treble cut filter** when used in

audio applications. A low-pass filter is the opposite of a high-pass filter, and a band-pass filter is a combination of a low-pass and a high-pass.

BASS_BFX_BQF_HIGHPASS:

A **high-pass filter** is an LTI filter that passes high frequencies well but attenuates (i.e., reduces the amplitude of) frequencies lower than the fCenter frequency. The actual amount of attenuation for each frequency is a design parameter of the filter. It is sometimes called a **low-cut filter**; the terms **bass-cut filter** or **rumble filter** are also used in audio applications.

BASS_BFX_BQF_BANDPASS:

A **band-pass filter** is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range. An example of an analogue electronic band-pass filter is an RLC circuit (a resistor—inductor—capacitor circuit). These filters can also be created by combining a low-pass filter with a high-pass filter.

BASS_BFX_BQF_NOTCH:

In signal processing, a **band-stop filter** or **band-rejection filter** is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels. It is the opposite of a band-pass filter. A **notch filter** is a band-stop filter with a narrow stopband (high Q factor). Notch filters are used in live sound reproduction (Public Address systems, also known as PA systems) and in instrument amplifier (especially amplifiers or preamplifiers for acoustic instruments such as acoustic guitar, mandolin, bass instrument amplifier, etc.) to reduce or prevent feedback, while having little noticeable effect on the rest of the frequency spectrum. Other names include 'band limit filter', 'T-notch filter', 'band-elimination filter', and 'band-reject filter'.

BASS_BFX_BQF_ALLPASS:

An **all-pass filter** is a signal processing filter that passes all frequencies equally, but changes the phase relationship between various frequencies. It does this by varying its propagation delay with frequency. Generally, the filter is described by the frequency at which the phase shift crosses 90° (i.e., when the input and output signals go into quadrature — when there is a quarter wavelength of delay between them).

BASS_BFX_BQF_PEAKINGEQ:

A **peaking equalizer** raises or lowers a range of frequencies around a central point in a bell shape. A peaking equalizer with controls to adjust the level (fGain), fBandwidth (fQ) and center frequency (Hz) is called a parametric equalizer.

Peaking Equalizer in BASS_FX can be achieved directly and efficiently by BASS_FX_BFX_PEAKEQ

BASS_BFX_BQF_LOWSHELF:

A **low-shelf filter** passes all frequencies, but increasing or reducing frequencies below the fCenter frequency by specified amount.

BASS_BFX_BQF_HIGHSHELF:

A **high-shelf filter** passes all frequencies, but increasing or reducing frequencies above the fCenter frequency by specified amount.

See also

BASS_BFX_ECHO4 structure

Used with BASS FXGetParameters and BASS FXSetParameters to retrieve and set the parameters of an echo/reverb effect.

```
typedef struct {
  float fDryMix;
  float fWetMix;
  float fFeedback;
  float fDelay;
  BOOL bStereo;
  int lChannel;
} BASS_BFX_ECHO4;
```

Members		
fDryMix	Dry (unaffected) signal mix	[-2+2]
fWetMix	Wet (affected) signal mix	[-2+2]
fFeedback	Output signal to feed back into input	[-1+1]
fDelay	Delay in seconds	[0 <n]< td=""></n]<>
bStereo	Only allowed with even number of channels	TRUE/FALSE
lChannel	The affected channels using BASS_BFX_CHANxxx	

Info

flags

This is an echo effect that replays what you have played one or more times after a period of time. It's something like the echoes you might hear shouting against a canyon wall. For reverb effect enable feedback.

The fDryMix is the volume of input signal & the fWetMix is the volume of delayed signal. The fDelay is the delay time in sec. The fFeedback sets how much delay is feed back to the input (for repeating delays). If bStereo is enabled and a stream has an even number of channels then, each even channels will be

echoed to each other.

Examples

	Dry Wet Feedback	Delay
Small Echo	0.9990.9990.0	0.20
Many Echoes	0.9990.9990.7	0.50
Reverse Echoes	0.9990.999-0.7	0.80
Robotic Voice	0.5000.8000.5	0.10

See also

BASS_BFX_PITCHSHIFT structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of pitch-shift using FFT effect.

```
typedef struct {
  float fPitchShift;
  float fSemitones;
  long lFFTsize;
  long lOsamp;
  int lChannel;
} BASS_BFX_PITCHSHIFT;
```

Members

fPitchShift	A factor value which is between 0.5 (one octave down) and 2 (one octave up) (1 won't change the pitch) (fSemitones is not in use, fPitchShift has a priority over fSemitones)	[0.512], def. 1
fSemitones	Semitones (0 won't change the pitch)	def. 0
lFFTsize	Defines the FFT frame size used for the processing. Typical values are 1024, 2048 and 4096 It may be any value <= 8192 but it MUST be a power of 2	[10248192], def. 2048
lOsamp	Is the STFT oversampling factor which also determines the overlap between adjacent STFT frames It should at least be 4 for moderate scaling ratios. A value of 32 is recommended for best quality (better quality = higher CPU usage)	[432], def. 8
lChannel	The affected channels using BASS_BFX_CHANxxx flags	

Info

Changes the sound pitch or key, without affecting the sound tempo or speed using FFT.

More information about Pitch Shifting Using The Fourier Transform: http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

Remarks

The sound quality is average.

On mobile devices, due to expensive CPU and FPU usage, this effect is enabled for iOS, Android (armeabi-v7a, x86) and Linux ARM (hardfp), that using floating-point instead of fixed-point.

See also

BASS_BFX_FREEVERB structure

Used with <u>BASS_FXGetParameters</u> and <u>BASS_FXSetParameters</u> to retrieve and set the parameters of a reverb effect.

```
typedef struct {
    float fDryMix;
    float fWetMix;
    float fRoomSize;
    float fDamp;
    float fWidth;
    DWORD lMode;
    int lChannel;
} BASS_BFX_FREEVERB;
```

Members

fDryMix	Dry (unaffected) signal mix	[01], def. 0
fWetMix	Wet (affected) signal mix	[03], def. 1.0f
fRoomSize	e Room size	[01], def. 0.5f
fDamp	Damping	[01], def. 0.5f
fWidth	Stereo width	[01], def. 1
lMode	0 or BASS_BFX_FREEVERB_MODE_FREEZE	def. 0 (no freeze)
lChannel	The affected channels using BASS_BFX_CHANxxx flags	

Info

Reverb adds reverberation (rapid, modified repetitions blended with the original sound that gives an impression of ambience). The Reverb effect is based on the

original "freeverb" algorithm. Adding reverberation is sometimes desirable for concert halls that are too small or contain so many people that the hall's natural reverberance is diminished. Applying a small amount of stereo reverb to an untreated mono signal duplicated into a two-channel stereo track will usually make it sound more natural.

See also

BASS_FX_TempoCreate

Creates a resampling stream from a decoding channel.

```
HSTREAM BASS_FX_TempoCreate(
    DWORD chan,
    DWORD flags
);
```

Parameters

chan Stream/music/wma/cd/any other supported add-on format using a decoding flags A combination of the following flags:

BASS_SAMPLE_LOOP Looped? Note that only complete sample le

allowed by DirectSound (ie. you can't loop

of a sample)

Force the sample to not use hardware mixi

BASS_SAMPLE_SOFTWARE

BASS_SAMPLE_3D Use 3D functionality. This is ignored if

BASS_DEVICE_3D wasn't specified when BASS_Init. 3D samples must be mono (use

BASS_SAMPLE_MONO)

BASS_SAMPLE_FX
Enable the old implementation of DirectX

requires DirectX 8 or above

See the <u>DX8 effect implementations</u> sectio details. Use <u>BASS_ChannelSetFX</u> to add ϵ

the stream

Automatically free the stream's resources v

BASS_STREAM_AUTOFREE has reached the end, or when BASS_Chan

(or **BASS_Stop**) is called

BASS_STREAM_DECODE Decode the sample data, without outputting

BASS_ChannelGetData to retrieve decode

data.

BASS_SAMPLE_SOFTWARE/3D/FX/AU are all ignored when using this flag, as are

SPEAKER flags

BASS_SPEAKER_xxx Speaker assignment flags

BASS_FX_FREESOURCE Free the source handle as well

Return value

If successful, the tempo stream handle is returned, else 0 is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

Enable Tempo supported flags in BASS_FX_TempoCreate and the others to source handle.

Example

Create a Tempo stream.

HSTREAM chan=BASS_StreamCreateFile(...,BASS_STREAM_DECODE); // create decoded stream if (chan) chan=BASS_FX_TempoCreate(chan,BASS_FX_FREESOURCE); // create a tempo stream

Error codes

BASS_ERROR_HANDLE *chan* is not valid BASS_ERROR_DECODE The *chan* is not a decoding channel. Make sure the

See also

BASS_FX_TempoGetSource, BASS_FX_TempoGetRateRatio, Tempo Attributes

BASS_FX_TempoGetSource

Get the source channel handle.

```
DWORD BASS_FX_TempoGetSource(
    HSTREAM chan
);
```

Parameters

chan Tempo stream (or source channel) handle

Return value

If successful, the source channel handle is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error code

BASS_ERROR_HANDLE chan is not valid

See also

BASS FX TempoCreate, BASS FX TempoGetRateRatio, Tempo Attributes

BASS_FX_TempoGetRateRatio

Get the ratio of the resulting rate and source rate (the resampling ratio).

```
float BASS_FX_TempoGetRateRatio(
    HSTREAM chan
);
```

Parameters

chan Tempo stream (or source channel) handle

Return value

If successful, the resampling ratio is returned, else 0 is returned. Use BASS_ErrorGetCode to get the error code.

Error code

BASS_ERROR_HANDLE chan is not valid

See also

BASS FX TempoCreate, BASS FX TempoGetSource, Tempo Attributes

BASS_ATTRIB_TEMPO attribute

Set tempo of a channel.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO,
    float tempo
);
```

Parameters

handle Tempo stream (or source channel)

handle

tempo Tempo value [-95%..0..+5000%]

percents

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

```
BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified
```

Info

Tempo (time-stretch): Changes the sound to play at faster or slower speed than original, without affecting the sound pitch.

Example

Set Tempo by BPM value:

```
float tempo = (goalBPM / bpmValue - 1.0f) * 100.0f;
BASS_ChannelSetAttribute(chan, BASS_ATTRIB_TEMPO, tempo);
```

See also

BASS_ATTRIB_TEMPO_PITCH attribute

Set pitch of a channel.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_PITCH,
    float pitch
);
```

Parameters

handle Tempo stream (or source channel) handle

pitch Pitch (key) [-60....0....+60] semitones

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified

Info

Pitch (key): Changes the sound pitch or key, without affecting the sound tempo or speed.

See also

BASS_ATTRIB_TEMPO_FREQ attribute

Set sample rate of a channel.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_FREQ,
    float freq
);
```

Parameters

handle Tempo stream (or source channel) handle

freq Samplerate in Hz (must be within 5% to 5000% of the original

sample rate)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified

Info

Sample Rate : Changes both the sound tempo and pitch, as if an LP disc was played at wrong RPM rate.

See also

About Algorithms

BASS_FX provides three seemingly independent effects: tempo, pitch and sample rate control. These three controls are implemented as combination of two primary effects, *sample rate transposing* and *time-stretching*.

Sample rate transposing affects both the audio stream duration and pitch. It's implemented simply by converting the original audio sample stream to the desired duration by interpolating from the original audio samples. In BASS_FX, linear interpolation with anti-alias filtering is used. Theoretically a higher-order interpolation provide better result than 1st order linear interpolation, but in audio application linear interpolation together with anti-alias filtering performs subjectively about as well as higher-order filtering would.

Time-stretching means changing the audio stream duration without affecting it's pitch. BASS_FX uses WSOLA-like time-stretching routines that operate in the time domain. Compared to sample rate transposing, time-stretching is a much heavier operation and also requires a longer processing "window" of sound samples used by the processing algorithm, thus increasing the algorithm input/output latency. Typical i/o latency for the BASS_FX time-stretch algorithm is around 100 ms.

Sample rate transposing and time-stretching are then used together to produce the tempo, pitch and rate controls:

- '**Tempo**' control is implemented purely by time-stretching.
- 'Rate' control is implemented purely by sample rate transposing.
- 'Pitch' control is implemented as a combination of time-stretching and sample rate transposing. For example, to increase pitch the audio stream is first time-stretched to longer duration (without affecting pitch) and then transposed back to original duration by sample rate transposing, which simultaneously reduces duration and increases pitch. The result is original duration but increased pitch.

BASS_FX uses SoundTouch library for its tempo/pitch processing.

See also BASS_FX_TempoCreate, BASS_FX_TempoGetSource,

BASS_FX_TempoGetRateRatio, Tempo Attributes

BASS_ATTRIB_TEMPO_OPTION_USE_AA attribute

Use Anti-Alias Filter on a tempo channel.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_USE_AA_FILTER,
    float use_aa_filter
);
```

Parameters

handle Tempo stream (or source channel) handle

use_aa_filter TRUE / FALSE (default TRUE)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Remarks

iOS, Android, WinCE and Linux ARM: Disabled the BASS_ATTRIB_TEMPO_OPTION_USE_AA_FILTER option on tempo streams with multi-channel by default for lower CPU usage. Doesn't SSE optimized for multi-channel streams.

Error codes

BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS_ATTRIB_TEMPO_OPTION_AA_FIL7 attribute

The AA Filter length.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_AA_FILTER_LENGTH,
    float aa_filter_length
);
```

Parameters

handle Tempo stream (or source channel) handle

aa_filter_length 8 .. 128 taps (default 32)

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE handle is invalid
BASS_ERROR_ILLPARAM An illegal
parameter was
specified

See also

BASS_ATTRIB_TEMPO_OPTION_USE_QU attribute

Use tempo quick algorithm.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO,
    float use_quickalgo
);
```

Parameters

handle Tempo stream (or source channel) handle

use_quickalgo TRUE / FALSE (default FALSE)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Remarks

iOS, Android, WinCE and Linux ARM: Enabled the BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO option on tempo streams by default for lower CPU usage.

Error codes

BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS_ATTRIB_TEMPO_OPTION_SEQUEN attribute

Set tempo sequence in ms.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_SEQUENCE_MS,
    float sequence_ms
);
```

Parameters

handle Tempo stream (or source channel)

handle

sequence_ms [-50%..+100%] of current value (default = 82, automatic

= 0)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE *handle* is invalid BASS_ERROR_ILLPARAM An illegal

parameter was

specified

See also

BASS_ATTRIB_TEMPO_OPTION_SEEKWl attribute

Set tempo seek window in ms.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_SEEKWINDOW_MS,
    float seekwindow_ms
);
```

Parameters

handle Tempo stream (or source channel)

handle

seekwindow_ms [-50%..+100%] of current value (default = 28, automatic

= 0)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE handle is invalid

BASS_ERROR_ILLPARAM An illegal

parameter was specified

See also

BASS_ATTRIB_TEMPO_OPTION_OVERLA attribute

Set tempo overlap in ms.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_OVERLAP_MS,
    float overlap_ms
);
```

Parameters

handle Tempo stream (or source channel)

handle

overlap_ms [-50%..+100%] of current value (default 8)

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE *handle* is invalid BASS_ERROR_ILLPARAM An illegal

parameter was specified

See also

BASS_ATTRIB_TEMPO_OPTION_PREVEN attribute

Prevents a clicking sound when the samplerate/pitch crosses from value <1 to >= 1 or vice versa during processing.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_TEMPO_OPTION_PREVENT_CLICK,
    float prevent_click
);
```

Parameters

handle Tempo stream (or source channel) handle

prevent_click TRUE / FALSE (default FALSE)

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS ErrorGetCode to get the error code.

Remarks

Default is FALSE as this involves slight sound quality compromise.

Error codes

BASS_ERROR_HANDLE handle is invalid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

Tuning the option attributes parameters

Remarks

_MS options are not fully tested, use it carefully as they may crash your application.

Tuning the option attributes parameters

The time-stretch algorithm has few parameters that can be tuned to optimize sound quality for certain application. The current default parameters have been chosen by iterative if-then analysis (read: "trial and error") to obtain best subjective sound quality in pop/rock music processing, but in applications processing different kind of sound the default parameter set may result into a sub-optimal result.

The default attribute parameter values are:

<u>BASS</u>	ATTRIB	<u>TEMPO</u>	OPTION	<u>SEQUENCE MS</u>	AUTOMATIC
BASS	ATTRIB	TEMP0	OPTION	SEEKWINDOW MS	AUTOMATIC
BASS	ATTRIB	TEMP0	OPTION	OVERLAP MS	8

These parameters affect to the time-stretch algorithm as follows:

- BASS_ATTRIB_TEMPO_OPTION_SEQUENCE_MS: This is the default length of a single processing sequence in milliseconds which determines the how the original sound is chopped in the time-stretch algorithm. Larger values mean fewer sequences are used in processing. In principle a larger value sounds better when slowing down the tempo, but worse when increasing the tempo and vice versa.
- BASS_ATTRIB_TEMPO_OPTION_SEEKWINDOW_MS: The seeking window default length in milliseconds is for the algorithm that seeks the best possible overlapping location. This determines from how wide a sample "window" the algorithm can use to find an optimal mixing location when the sound sequences are to be linked back together. By default, this setting value is calculated automatically according to tempo value.

The bigger this window setting is, the higher the possibility to find a better mixing position becomes, but at the same time large values may cause a "drifting" sound artifact because neighboring sequences can be chosen at more uneven intervals. If there's a disturbing artifact that sounds as if a constant frequency was drifting around, try reducing this setting.

• BASS_ATTRIB_TEMPO_OPTION_OVERLAP_MS: Overlap length in milliseconds. When the sound sequences are mixed back together to form again a continuous sound stream, this parameter defines how much the ends of the consecutive sequences will overlap with each other. By default, this setting value is calculated automatically according to tempo value.

This shouldn't be that critical parameter. If you reduce the BASS_ATTRIB_TEMPO_OPTION_SEQUENCE_MS setting by a large amount, you might wish to try a smaller value on this.

The table below summarizes how the parameters can be adjusted for different applications:

Parameter name	Default value magnitude	Larger value affects	Smaller value affects	Music	Speech	Effect in CPU burden
SEQUENCE_MS	Default value is relatively large, chosen for slowing down music tempo	down tempo. Growing	speeding up tempo. Reducing the value accelerates the "echoing"	Default value usually good	A smaller value than default might be better	Increasing the parameter value reduces computati burden

		down the tempo.	tempo			
SEEKWINDOW_MS	Default value is relatively large, chosen for slowing down music tempo	Larger value eases finding a good mixing position, but may cause a "drifting" artifact	the	Default value usually good, unless a "drifting" artifact is disturbing.	value usually good	Increasing the parameter value increases computati burden
OVERLAP_MS	Default value is relatively large, chosen to suit with above parameters.		If you reduce the "sequence ms" setting, you might wish to try a smaller value.			Increasing the parameter value increases computati burden

Performance Optimizations

General optimizations:

The time-stretch routine has a 'quick' mode that substantially speeds up the algorithm but may degrade the sound quality by a small amount. This mode is activated by BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO and value "TRUE", i.e.

```
BASS_ChannelSetAttribute(chan, BASS_ATTRIB_TEMPO_OPTION_USE_QUICKALGO, TRUE);
```

CPU-specific optimizations:

• Intel SSE/SSE2 optimized routines are used with compatible CPUs when floating point sample type is used. Processors compatible with SSE extension are Intel processors starting from Pentium-III, and AMD processors starting from Athlon XP.

See also

BASS FX TempoCreate, BASS FX TempoGetSource, BASS FX TempoGetRateRatio, Tempo Attributes

BASS_FX_ReverseCreate

Creates a reversed stream from a decoding channel.

```
HSTREAM BASS_FX_ReverseCreate(
    DWORD chan,
    float dec_block,
    DWORD flags
);
```

Parameters

chan Stream/music/wma/cd/any other supported add-on format using a decodec_block Length of decoding blocks in seconds. Larger blocks means less seeki larger spikes

flags A combination of the following flags:

BASS_SAMPLE_LOOP Looped? Note that only complete san

allowed by DirectSound (ie. you can'

of a sample)

Force the sample to not use hardware

BASS_SAMPLE_SOFTWARE

BASS_SAMPLE_3D Use 3D functionality. This is ignored

BASS_DEVICE_3D wasn't specified BASS_Init. 3D samples must be mon

BASS_SAMPLE_MONO)

BASS_SAMPLE_FX Enable the old implementation of Dir

See the <u>DX8 effect implementations</u>: details. Use <u>BASS ChannelSetFX</u> to

the stream

Automatically free the stream's resou

BASS_STREAM_AUTOFREE has reached the end, or when BASS_

(or BASS_Stop) is called

BASS_STREAM_DECODE Decode the sample data, without outr

BASS_ChannelGetData to retrieve do

data.

BASS_SAMPLE_SOFTWARE/3D/F

are all ignored when using this flag, a

SPEAKER flags

BASS_SPEAKER_xxx Speaker assignment flags

BASS_FX_FREESOURCE Free the source handle as well

Return value

If successful, the handle of the reversed stream is returned, else 0 is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Remarks

MODs are supported if BASS_MUSIC_PRESCAN flag was applied to a source handle. Enable reverse supported flags in BASS_FX_ReverseCreate and the others to source handle. For better MP3/2/1 reverse playback create the stream using the BASS_STREAM_PRESCAN flag.

By default stream's position will start from the end with the BASS_FX_RVS_REVERSE direction.

Example

Create a Reverse stream.

HSTREAM chan=BASS_StreamCreateFile(...,BASS_STREAM_DECODE); // create decoded stream

if (chan) chan=BASS_FX_ReverseCreate(chan,2,BASS_FX_FREESOURCE); // create reverse stream, 2 secs decoding block

Error codes

BASS_ERROR_HANDLE *chan* is not valid
BASS_ERROR_DECODE The *chan* is not a decoding channel. Make sure the
BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS_FX_ReverseGetSource, Reverse Attribute

BASS_FX_ReverseGetSource

Get the source channel handle of the reversed stream.

```
DWORD BASS_FX_ReverseGetSource(
    HSTREAM chan
);
```

Parameters

chan Reverse stream (or source channel) handle

Return value

If successful, the handle of the source of the reversed stream is returned, else 0 is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error code

BASS ERROR HANDLE chan is not valid

See also

BASS_FX_ReverseCreate, Reverse Attribute

BASS_ATTRIB_REVERSE_DIR attribute

Set playback direction.

```
BOOL BASS_ChannelSetAttribute(
    DWORD handle,
    BASS_ATTRIB_REVERSE_DIR,
    float direction
);
```

Parameters

handle Reverse stream (or source channel) handle

direction Playback direction: BASS_FX_RVS_REVERSE or

BASS_FX_RVS_FORWARD

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE *handle* is invalid BASS_ERROR_ILLPARAM An illegal

parameter was specified

See also

BASS FX ReverseCreate, BASS FX ReverseGetSource

BASS_FX_BPM_DecodeGet

Get the BPM value of a decoding channel.

```
float BASS_FX_BPM_DecodeGet(
   DWORD chan,
   double startSec,
   double endSec,
   DWORD minMaxBPM,
   DWORD flags,
   BPMPROGRESSPROC *proc,
   void *user
);
```

Parameters

chan Stream/music/wma/cd/any other supported add-on format using a

decoding channel

startSec Start detecting position in seconds endSec End detecting position in seconds

minMaxBPM Set min & max bpm, e.g. MAKELONG(LOWORD.HIWORD),

LO=Min, HI=Max. 0 = defaults 29/200

flags BASS_FX_BPM_xxx or BASS_FX_FREESOURCE

proc User defined function to receive the detection progress in

percents, use NULL if not in use

user User instance data to pass to the callback function.

Return value

If successful, the original BPM value is returned, else -1 is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

BASS_FX_BPM_BKGRND flag is supported only in Windows platforms.

Error codes

BASS_ERROR_HANDLE *chan* is not valid
BASS_ERROR_DECODE The *chan* is not a decoding channel. Make sure the
BASS_ERROR_ILLPARAM An illegal parameter was specified
BASS_ERROR_ALREADY BPM detection, for this *chan*, is already being proc

See also

BASS FX BPM CallbackSet, BASS FX BPM CallbackReset, BASS FX BPM Translate, BASS FX BPM Free

BASS_FX_BPM_CallbackSet

Enable getting BPM value after period of time in seconds.

```
BOOL BASS_FX_BPM_CallbackSet(
    DWORD handle,
    <u>BPMPROC</u> *proc,
    double period,
    DWORD minMaxBPM,
    DWORD flags,
    void *user
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format

proc User defined function to receive the bpm value

period Detection period in seconds

minMaxBPM Set min & max bpm, e.g: MAKELONG(LOWORD.HIWORD),

LO=Min, HI=Max. 0 = defaults 29/200

flags Only BASS_FX_BPM_MULT2 flag is used

user User instance data to pass to the callback function.

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE handle is not valid

BASS_ERROR_ILLPARAM An illegal

parameter was

specified

See also

BASS FX BPM DecodeGet, BASS FX BPM CallbackReset, BASS FX BPM Translate, BASS FX BPM Free

BASS_FX_BPM_CallbackReset

Reset the buffers. Call this function after changing position.

```
BOOL BASS_FX_BPM_CallbackReset(
    DWORD handle
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

This function flushes the internal buffers of the BPM callback. BPM callback is automatically reset by <u>BASS_ChannelSetPosition</u>, except when called from a "mixtime" <u>SYNCPROC</u>.

Error code

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM DecodeGet, BASS FX BPM CallbackSet, BASS FX BPM Translate, BASS FX BPM Free

BASS_FX_BPM_Free

Frees all resources used by a given handle (decode or callback bpm).

```
BOOL BASS_FX_BPM_Free(
    DWORD handle
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

If BASS_FX_FREESOURCE flag is used, this will free the source decoding channel as well. You can't set/get this flag with BASS_ChannelFlags/BASS_ChannelGetInfo.

Error code

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM DecodeGet, BASS FX BPM CallbackSet, BASS FX BPM CallbackReset, BASS FX BPM Translate

BPMPROC callback

User defined callback function, to get the BPM after period of time in seconds.

```
void CALLBACK yourBpmProc(
    DWORD chan,
    float bpm,
    void *user
);
```

Parameters

chan Channel handle that the <u>BASS_FX_BPM_CallbackSet</u> applies to

bpm The bpm value

user The user instance data given when <u>BASS_FX_BPM_CallbackSet</u> was

called

See also

BASS FX BPM CallbackSet

BPMPROGRESSPROC callback

User defined callback function, to get the bpm detection progress in percents.

```
void CALLBACK yourBpmProgressProc(
    DWORD chan,
    float percent,
    void *user
);
```

Parameters

chan Channel handle that the <u>BASS_FX_BPM_DecodeGet</u> applies to percent The detection progress in percents [0%..100%]

user The user instance data given when <u>BASS_FX_BPM_DecodeGet</u> was called

See also

BASS FX BPM DecodeGet

BASS_FX_BPM_BeatDecodeGet

Enable getting Beat position in seconds of the decoded channel using the callback function.

```
BOOL BASS_FX_BPM_BeatDecodeGet(
    DWORD chan,
    double startSec,
    double endSec,
    DWORD flags,
    BPMBEATPROC *proc,
    void *user
);
```

Parameters

chan Stream/music/wma/cd/any other supported add-on format using a

decoding channel

startSec Start detecting position in seconds endSec End detecting position in seconds

flags BASS_FX_BPM_BKGRND or BASS_FX_FREESOURCE proc User defined function to receive the beat position in seconds

user User instance data to pass to the callback function

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Remarks

BASS_FX_BPM_BKGRND flag is supported only in Windows platforms.

Error codes

```
BASS_ERROR_HANDLE chan is not valid
BASS_ERROR_DECODE The chan is not a decoding channel. Make sure the
BASS_ERROR_ILLPARAM An illegal parameter was specified
BASS_ERROR_ALREADY Beat detection, for this chan, is already being proce
```

See also

BASS FX BPM BeatCallbackSet, BASS FX BPM BeatCallbackReset,
BASS FX BPM BeatSetParameters, BASS FX BPM BeatGetParameters,
BASS FX BPM BeatFree

BASS_FX_BPM_BeatCallbackSet

Enable getting Beat position in seconds in real-time.

```
BOOL BASS_FX_BPM_BeatCallbackSet(
    DWORD handle,
    <u>BPMBEATPROC</u> *proc,
    void *user
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format proc User defined function to receive the beat position in seconds

user User instance data to pass to the callback function

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error code

BASS_ERROR_HANDLE handle is not valid BASS_ERROR_ILLPARAM An illegal parameter was specified

See also

BASS FX BPM BeatDecodeGet, BASS FX BPM BeatCallbackReset,
BASS FX BPM BeatSetParameters, BASS FX BPM BeatGetParameters,
BASS FX BPM BeatFree

BASS_FX_BPM_BeatCallbackReset

Reset the buffers. Call this function after changing position.

```
BOOL BASS_FX_BPM_BeatCallbackReset(
    DWORD handle
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

This function flushes the internal buffers of the Beat callback. Beat callback is automatically reset by <u>BASS_ChannelSetPosition</u>, except when called from a "mixtime" <u>SYNCPROC</u>.

Error code

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM BeatDecodeGet, BASS FX BPM BeatCallbackSet,
BASS FX BPM BeatSetParameters, BASS FX BPM BeatGetParameters,
BASS FX BPM BeatFree

BASS_FX_BPM_BeatSetParameters

Set new values for beat detection.

```
BOOL BASS_FX_BPM_BeatSetParameters(
    DWORD handle,
    float bandwidth,
    float centerfreq,
    float beat_rtime
);
```

Parameters

handle	Stream/music/wma/cd/any other supported add-on format			
bandwidth	Bandwidth in Hz	[0< <rate 2]<br="">Hz</rate>	-	-1.0f = leave current
centerfreq	Center frequency	[0< <rate 2]<br="">Hz</rate>	-	-1.0f = leave current
beat_rtime	Beat release time in ms	ms	-	-1.0f = leave current

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM BeatDecodeGet, BASS FX BPM BeatCallbackSet,
BASS FX BPM BeatCallbackReset, BASS FX BPM BeatGetParameters,
BASS FX BPM BeatFree

BASS_FX_BPM_BeatGetParameters

Get current beat values.

```
BOOL BASS_FX_BPM_BeatGetParameters(
    DWORD handle,
    float *bandwidth,
    float *centerfreq,
    float *beat_rtime
);
```

Parameters

handle Stream/music/wma/cd/any other supported

add-on format

bandwidth Current bandwidth in Hz NULL = don't

retrieve it

centerfreq Current center frequency NULL = don't

retrieve it

beat rtime Current beat release time in ms NULL = don't

retrieve it

Return value

If successful, TRUE is returned, else FALSE is returned. Use <u>BASS_ErrorGetCode</u> to get the error code.

Error codes

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM BeatDecodeGet, BASS FX BPM BeatCallbackSet, BASS FX BPM BeatCallbackReset, BASS FX BPM BeatSetParameters, BASS FX BPM BeatFree

BASS_FX_BPM_BeatFree

Frees all resources used by a given handle (decode or callback beat).

```
BOOL BASS_FX_BPM_BeatFree(
    DWORD handle
);
```

Parameters

handle Stream/music/wma/cd/any other supported add-on format

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASS_ErrorGetCode to get the error code.

Remarks

If BASS_FX_FREESOURCE flag is used, this will free the source decoding channel as well. You can't set/get this flag with BASS_ChannelFlags/BASS_ChannelGetInfo.

Error code

BASS_ERROR_HANDLE handle is not valid

See also

BASS FX BPM BeatDecodeGet, BASS FX BPM BeatCallbackSet,
BASS FX BPM BeatCallbackReset, BASS FX BPM BeatSetParameters,
BASS FX BPM BeatGetParameters

BPMBEATPROC callback

User defined callback function, to get the Beat position in seconds.

```
void CALLBACK yourBpmBeatProc(
    DWORD chan,
    double beatpos,
    void *user
);
```

Parameters

chan Channel handle that the <u>BASS_FX_BPM_BeatCallbackSet</u> or <u>BASS_FX_BPM_BeatDecodeGet</u> has applied to

beatpos The exact beat position in seconds

user The user instance data given when <u>BASS_FX_BPM_BeatCallbackSet</u> or BASS_FX_BPM_BeatDecodeGet was called

Remarks

To filter out false beat positions, users first will have to find a BPM of a song, using one of the BPM detection functions, e.g: <u>BASS_FX_BPM_DecodeGet</u> When you know the BPM, you can calculate the approximate duration that should be between beat positions, e.g:

```
stream_length = 330 seconds (5 minutes 30 seconds)
stream_bpm = 140 (beats per minute)
duration_between_beats = stream_bpm / stream_length = 140 / 330 = 0.4242 sec
```

If beat position returned by callback functions doesn't fit in **duration_between_beats**, then it's probably some harmonic sound or a false beat, so you can ignore it.

That way you can also detect and map all beats, including the 1st one.

Examples

A very simple way to count the BPM in real-time, using only 2 beats.

```
double prevBeatPos = 0.0f;  // previous beat position in seconds
double bpm = 0.0f;  // the bpm

void CALLBACK BeatProcGetBPM(DWORD handle, double beatpos, void
*user)
{
    if (beatpos != prevBeatPos)
        bpm = 60.0f / (beatpos - prevBeatPos);  // calculate the bpm

    prevBeatPos = beatpos;  // save current beat position
}
```

Get the detection progress in percents:

```
float progress = 100.f * (beatpos - startpos) / (endpos - startpos);
```

See also

BASS FX BPM BeatCallbackSet, BASS FX BPM BeatDecodeGet