z.reverb SDK 2.3.0

by zplane.development

July 8, 2016

Contents

1 z.reverb SDK Documentation **2**
- 1.1 Introduction . . . 2
- 1.2 API Documentation . . . 2
  - 1.2.1 Naming Conventions . . . 2
  - 1.2.2 Required Functions . . . 2
  - 1.2.3 Optional Functions . . . 3
  - 1.2.4 Parameter Setting Functions . . . 3
  - 1.2.5 Calling Conventions . . . 8
- 1.3 SDK Project . . . 9
  - 1.3.1 File Structure . . . 9
  - 1.3.2 Project Structure . . . 10
  - 1.3.3 Project Configurations . . . 10
- 1.4 Coding Style minimal overview . . . 10
- 1.5 Support . . . 10

2 File Index **11**
- 2.1 File List . . . 11

3 File Documentation **11**
- 3.1 docugen.txt File Reference . . . 11
  - 3.1.1 Detailed Description . . . 11
- 3.2 ReverbAPI.h File Reference . . . 11
  - 3.2.1 Detailed Description . . . 13
  - 3.2.2 Typedef Documentation . . . 13
  - 3.2.3 Enumeration Type Documentation . . . 13
  - 3.2.4 Function Documentation . . . 15

1 z.reverb SDK Documentation

1.1 Introduction

The z.reverb SDK allows to add a reverberation effect to the incoming audio material.

The first section of this document gives a detailed API documentation with input and output requirements and a detailed function documentation. After that, a short overview over the z.reverb-SDK folder structure is given.

The following chapters contain automatically generated function and parameter description.

1.2 API Documentation

To integrate the reverb in source code, the header file for this effect has to be included. The name of the header file is ReverbAPI.h. Furthermore, the library libReverb.lib has to linked to the executable or library.

The interface allows in-place processing, i.e. the audio data can be replaced by the audio data plus the corresponding effect. Note that inplace processing is only possible if the number of input channels equals the number of output channels.

All needed variable types are either standard C-types or are defined in the API-header files.

In the case of more than one input channel, the input data has to be *interleaved*.

1.2.1 Naming Conventions

All API functions are preceded by an initial *Reverb_* to be easily distinguished from other functions. After that, the function name makes clear what should happen; all functions beginning with *Reverb_Set* do set any properties or parameters of the corresponding effect. All functions ending with *Instance* create or destroy an object/instance of the effect. Examples are Reverb_CreateInstance, Reverb_GetReverbTime, Reverb_Process.

When talking about **frames**, the number of audio samples in one channel is meant. If the sample size is float, one sample has a memory usage of 4 byte. The memory usage of one stereo frame is 8 byte.

1.2.2 Required Functions

The following functions have to be called when using an effect of the z.reverb library:

- **Reverb_CreateInstance (void** ∗∗ **phInstance, int iSampleRate, int iNumOfChannels, Quality_t qQualityMode)**

  This function creates an instance of the required effects. The parameter phInstance is the address of a pointer where the handle to the instance is written, iSampleRate is sample rate of input signal in Hz, iNumOfChannels is the number

of input channels, and qQualityMode defines four different quality levels, as defined in Quality_t. The lowest quality level will give the lowest processing workload and vice versa.

- **Reverb_DestroyInstance (void** ∗∗ **phInstance)**

  This function destroys an instance of the used effect. The parameter phInstance is the address of the instance handle pointer. The instance handle will be set to NULL.

  The return value will equal 0 on success.

- **Reverb_Process (void** ∗ **pCReverbInstance, float** ∗ **pInputBuffer, float** ∗ **pOutputBuffer, int iNumOfFrames)**

  This function destroys an instance of the used effects. Available functions are: : The parameter phInstance is the instance handle pointer, pfInputBuffer is the pointer to interleaved input data, pfOutputBuffer is pointer to buffer for the interleaved output data (may be the same as input data in all cases except the case that the number of output channels does not equal the number of input channels), iNumOfFrames is the number of frames in the input buffer. Note that the number of output channels is always two!

  The return value will equal 0 on success.

1.2.3 Optional Functions

Although the use of the functions listed here is optional, many of these are either required to use this library in a senseful way or are recommended to use.

- **Reverb_Reset ()**

  This function resets the internal buffers of the used effects. It should be called when the same effect instance should process a different input audio stream, or when the the effect was bypassed. The parameter phInstance is the instance handle pointer.

  The return value will equal 0 on success.

1.2.4 Parameter Setting Functions

There are plenty of parameters to control the reverberation effect. However, it does not make sense in every use-case to provide all these parameters in the user interface. From a developers point of view, the parameters could be clustered in different priority classes, i.e. parameters which have to appear in user interface, and parameters with lower priority that do not necessarily have to appear in the user interface. Ths following sorts the available parameters into three priority classes:

High Priority:	Preset, ReverbTime, PreDelay, Wetness (only when used as insert), HighCut
Medium Priority:	LPAmount, ERGain, LRGain, BassBoost
Low Priority:	Liveness, Diffusion, Wetness (when used as send)

The API provides the following parameter functions:

- **Reverb_GetBassBoost (.)**

  Returns the bass gain. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the gain of the bass frequencies in dB. See also function Reverb_SetBassBoost (.).

- **Reverb_GetDiffusion (.)**

  Returns the diffusity. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the diffusity in percent. See also function Reverb_SetDiffusion (.).

- **Reverb_GetERGain (.)**

  Returns the gain of the early reflections. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the gain of the early reflections in dB. See also function - Reverb_SetERGain (.).

- **Reverb_GetHold (.)**

  Returns the status of the hold effect. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the current status of HOLD. See also function Reverb_Set-Hold (.).

- **Reverb_GetLiveness (.)**

  Returns the liveliness. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the liveliness in percent. See also function Reverb_Set-Liveness (.).

- **Reverb_GetLRGain (.)**

  Returns the gain of the late reverberation tail. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

  The return value is the gain of the late reverberation tail in dB. See also function Reverb_SetLRGain (.).

- **Reverb_GetPreDelay (.)**

  Returns the pre delay. The function parameters are:

  – **void∗ pCReverbInstance**: instance handle pointer.

The return value is the pre delay in seconds. See also function Reverb_SetPreDelay (.).

- **Reverb_GetPreset (.)**

  Returns the current preset. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.

  The return value is the current preset. See also function Reverb_SetPreset (.).

- **Reverb_GetReverbTime (.)**

  Returns the reverberation time. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.

  The return value is the reverberation time in seconds. See also function Reverb_SetReverbTime (.).

- **Reverb_GetWetness (.)**

  Returns the wetness. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.

  The return value is the wetness in percent. See also function Reverb_SetWetness (.).

- **Reverb_SetBassBoost (.)**

  Sets the gain for the bass frequencies. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.
  - **float fBassBoost**: gain of the bass frequncies in the reverberation in dB. The allowed range is from -18...18, default value depends on the selected preset. High values will result in a darker room (and in an increased output gain).

  The return value will equal 0 on success.

- **Reverb_SetDiffusion (.)**

  Sets the diffusity of the reverberation. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.
  - **float fDiffusion**: diffusity of the the reverberation in percent. The allowed range is from 0...1, default value depends on the selected preset. High values will result in a larger room impression.

  The return value will equal 0 on success.

- **Reverb_SetERGain (.)**

  Sets the gain for early reflections. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.

- **float fERGainIndB**: gain of the early reflections in the reverberation in dB. The allowed range is from -12...12, default value depends on the selected preset. High values will result in a more narrowed room impression (and in an increased output gain).

The return value will equal 0 on success.

int Reverb_SetERLRBalance (void* , float);

- **Reverb_SetERLRBalance (.)**

Sets the amount of ER/LR gain relative to the current ERGain/LRGain-Settings. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **float fERLRBalance**: value between -1 (maximum ER, minimum LR) and 1 (minimum ER, maximum LR), default value is 0 (changes depend on current ERGain/LRGain settings!)

  on the selected preset. High values will result in a more narrowed room impression (and in an increased output gain).

The return value will equal 0 on success.

- **Reverb_SetHighCut (.)**

Sets the low pass frequency for the reverberation. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **float fLPFreq**: low pass frequency of the reverberation in Hertz. Valid parameter range 100...min (18000, SampleRate/2.2F); the default value depends on selected preset. Low values will result in a darker room (and in a decreased output gain). See also function Reverb_SetLPAmount (.)

The return value will equal 0 on success.

- **Reverb_SetHold (.)**

This function is not motivated by room acoustics, but is an effect function that allows to freeze the reverberation at one state. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **Hold_t eHold**: hold mode can be _HOLD_ON to switch the effect on, _HOLD_OFF to switch the effect off or _HOLD_ACCUMULATE to switch the effect on but also send new input to the reverb. The last option is experimental and will result in clipping, i.e. in instability in most cases, especially in the case of loud input signals.

The return value will equal 0 on success.

- **Reverb_SetLiveness (.)**

Sets the liveliness of of the reverberation tails, i.e. could make the reverberation more interesting. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.

- **float fLiveness**: amount of liveliness of the reverberation in percent. - The allowed range is from 0.05...1.0, default value depends on the selected preset. High values could result in detuning effects especially for tonal input signals.

The return value will equal 0 on success.

- **Reverb_SetLPAmount (.)**

Sets the amount of low pass filtering for the reverberation. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **float fLPAmount**: amount of low pass filtering of the reverberation. - The allowed range is from 0.0005...0.9995, default value depends on the selected preset. High values will result in a darker room (and in a decreased output gain). See also function Reverb_SetHighCut (.)

The return value will equal 0 on success.

- **Reverb_SetLRGain (.)**

Sets the gain for late reverberation. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **float fLRGainIndB**: gain of the early reflections in the reverberation in dB. The allowed range is from -12...12, default value depends on the selected preset. High values will result in a more open room impression (and in an increased output gain).

The return value will equal 0 on success.

- **Reverb_SetPreDelay (.)**

Sets the predelay, i.e. the delay wet output signal, of the effect. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **float fPreDelay**: delay of the wet output signal in seconds. The allowed range is from 0...0.3, default value depends on the selected preset.

The return value will equal 0 on success.

- **Reverb_SetPreset (.)**

Sets all intern parameters acc. to the preset. The function parameters are:

- **void* pCReverbInstance**: instance handle pointer.
- **ReverbPresets_t ePreset**: preset to set acc. the enum ReverbPresets_t. The parameter ePreset can be of the type _MEDIUM_HALL, _LARGE_HALL, _PLATE, _CATHEDRAL or _ROOM. Default value is _MEDIUM_HALL.

The return value will equal 0 on success.

- **Reverb_SetReverbTime (.)**

  Sets the reverberation time (the length of the reverberation tail). The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.
  - **float fReverbTime**: reverberation time in seconds. The allowed range is from 0.4986...1000, default value depends on the selected preset. High values will result in a larger room room impression.

  The return value will equal 0 on success.

- **Reverb_SetWetness (.)**

  Sets the wetness of the effect. The function parameters are:

  - **void* pCReverbInstance**: instance handle pointer.
  - **float fWetness**: amount of the wet signal in percent of the output signal. The allowed range is from 0...1.0, default value is 0.2. The default value corresponds to a dB value of -12dB relative to the direct signal (ratio 0.-2/0.8).

  The return value will equal 0 on success.

1.2.5 Calling Conventions

The usage of the effects is straight forward. After an instance has been created with Reverb_CreateInstance, the parameters can be set with the functions *EffectName*_Set-*ParameterName* (.).

Then the audio signal can be processed with Reverb_Process. Please note that the parameter settings functions must not be called while the processing function is called. However, they may be called between two calls of the processing function.

After successful processing, the instance of the effect can be destroyed with Reverb_-DestroyInstance. The following example can be found in the file TestReverb.cpp.

Declare a new handle of a reverb instance:

```
void                *pReverb;                       // instance handle
```

and create a new instance with the necessary information

```
//create a new instance of reverberation
Reverb_CreateInstance(&pReverb, iSampleRate, iNumOfChannels, _HIGHEST);
```

After that, you are able to set and get parameters, e.g.

```
Reverb_SetPreset (pReverb, _MEDIUM_HALL);
fprintf(stderr, "Reverb Time(s):\t%2.2f\n", Reverb_GetReverbTime (pReverb))
  ;
fprintf(stderr, "EarlyRefl(dB):\t%2.2f\n", Reverb_GetERGain (pReverb));
fprintf(stderr, "LateReverb(dB):\t%2.2f\n", Reverb_GetLRGain (pReverb));
fprintf(stderr, "PreDelay(ms):\t%2.2f\n", 1000.0F*Reverb_GetPreDelay (
  pReverb));
```

The actual processing is done with consecutive blocks of audio data:

```
// now we can begin to process!
while(bReadNextFrame)
{
    iNumSamplesRead = InputFile.Read (apfInputSplit, _BLOCKSIZE);
        for(int j = 0; j < iNumOfChannels; j++)
    // write output data to file
```

After successful processing, the instance can be destroyed:

1.3 SDK Project

1.3.1 File Structure

1.3.1.1 Documentation

This documentation and all other documentation can be found in the directory **./doc**.

1.3.1.2 Project Files

The MS VisualC++-Workspace (.dsw) and all Projectfiles (.dsp) can be found in the directory **./build** and its subfolders, where the subfolders names correspond to the project names.

1.3.1.3 Source Files

All source files are in the directory **./src** and its subfolders, where the subfolder names equally correspond to the project names.

1.3.1.4 Include Files

If include files are project intern, they are in the source directory **./src** of the project itself. If include files are to be included by other projects they can be found in **./src/include**. If a SDK-like library/DLL is built for which a header is needed, such a header can be found in **./inc**.

1.3.1.5 Resource Files

The resource files can be found in the subdirectory **/res** of the corresponding build-directory.

1.3.1.6 Library Files

The directory **./lib** is for used and built libraries.

1.3.1.7 Binary Files

The final executable as well as the distributable Dynamic Link Libraries can be found in the directory **./bin**. In debug-builds, the binary files are in the subfolder **/Debug**.

1.3.1.8 Temporary Files

The directory **./tmp** is for all temporary files while building the projects. In debug-builds, the temporary files can be found in the subfolder **/Debug**.

1.3.2 Project Structure

The project structure is as following:

- **TestReverb**: command line test project for the reverb.

  The project output is an executable binary (EXE).

1.3.3 Project Configurations

For all projects included in the workspace, the default configurations Win32 Release and Win32 Debug are available.

1.4 Coding Style minimal overview

Variable names have a preceding letter indicating their types:

unsigned:	u
pointer:	p
array:	a
class:	C
bool:	b
char:	c
short (int16):	s
int (int32):	i
__int64:	l
float (float32):	f
double (float64):	d
char:	c

For example, a pointer to a buffer of unsigned ints will be named puiBufferName.

1.5 Support

Support for the SDK is - within the limits of the agreement - available from:

zplane.development

katzbachstr. 21

d-10965 berlin

germany

fon: +49.30.854 09 15.0

fax: +49.30.854 09 15.5

@: info@zplane.de

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

ReverbAPI.h
Interface of the zReverb effect **11**

3 File Documentation

3.1 docugen.txt File Reference

3.1.1 Detailed Description

source documentation main file

Definition in file docugen.txt.

3.2 ReverbAPI.h File Reference

interface of the zReverb effect.

Typedefs

- typedef enum ReverbPresets_t_tag ReverbPresets_t
  List of available presets (names explain the preset)
- typedef enum Hold_t_tag Hold_t
  possible states of hold-effect
- typedef enum Quality_t_tag Quality_t
  possible quality modes to adjust workload vs. quality

Enumerations

- enum ReverbPresets_t_tag { _MEDIUM_HALL, _LARGE_HALL, _PLATE, _CATHEDRAL, _ROOM, _ROOM1, _ROOM2 }
  List of available presets (names explain the preset)
- enum Hold_t_tag { _HOLD_ON, _HOLD_OFF, _HOLD_ACCUMULATE }
  possible states of hold-effect
- enum Quality_t_tag { _HIGHEST, _HIGH, _LOW, _LOWEST }

possible quality modes to adjust workload vs. quality

- enum ReverbVersion_t { kReverbMajor, kReverbMinor, kReverbPatch, kReverbBuild, kNumReverbVersionInts }

Functions

- int Reverb_CreateInstance (void **pCReverbInstance, int iSampleRate, int iNumOfChannels, Quality_t qQualityMode)
- int Reverb_DestroyInstance (void **pCReverbInstance)
- const int Reverb_GetVersion (const ReverbVersion_t eVersionIdx)
- const char * Reverb_GetBuildDate ()
- void Reverb_Reset (void *pCReverbInstance)
- int Reverb_Process (void *pCReverbInstance, float *pfInputBuffer, float *pfOutputBuffer, int iNumOfFrames)
- int Reverb_ProcessSplit (void *pCReverb, float **ppfInputBuffer, float **ppfOutputBuffer, int iNumOfFrames)
- int Reverb_SetPreset (void *pCReverbInstance, ReverbPresets_t ePreset)
- int Reverb_SetWetness (void *pCReverbInstance, float fWetness)
- int Reverb_SetPreDelay (void *pCReverbInstance, float fPreDelay)
- int Reverb_SetBassBoost (void *pCReverbInstance, float fBassBoost)
- int Reverb_SetHighCut (void *pCReverbInstance, float fLPFreq)
- int Reverb_SetLPAmount (void *pCReverbInstance, float fLPAmount)
- int Reverb_SetReverbTime (void *pCReverbInstance, float fReverbTime)
- int Reverb_SetERLRBalance (void *pCReverbInstance, float fERLRBalance)
- int Reverb_SetERGain (void *pCReverbInstance, float fERGainIndB)
- int Reverb_SetLRGain (void *pCReverbInstance, float fLRGainIndB)
- int Reverb_SetHold (void *pCReverbInstance, Hold_t eHold)
- int Reverb_SetLiveness (void *pCReverbInstance, float fLiveness)
- int Reverb_SetDiffusion (void *pCReverbInstance, float fDiffusion)
- int Reverb_SetBuildUpSteepness (void *pCReverbInstance, float fBuildUp)
- float Reverb_GetReverbTime (void *pCReverbInstance)
- Hold_t Reverb_GetHold (void *pCReverbInstance)
- float Reverb_GetERLRBalance (void *pCReverbInstance)
- float Reverb_GetERGain (void *pCReverbInstance)
- float Reverb_GetLRGain (void *pCReverbInstance)
- float Reverb_GetPreDelay (void *pCReverbInstance)
- float Reverb_GetBassBoost (void *pCReverbInstance)
- float Reverb_GetWetness (void *pCReverbInstance)
- float Reverb_GetLiveness (void *pCReverbInstance)
- float Reverb_GetDiffusion (void *pCReverbInstance)
- float Reverb_GetBuildUpSteepness (void *pCReverbInstance)
- float Reverb_GetHighCut (void *pCReverbInstance)
- float Reverb_GetLPAmount (void *pCReverbInstance)
- ReverbPresets_t Reverb_GetPreset (void *pCReverbInstance)

3.2.1 Detailed Description

interface of the zReverb effect. :

Definition in file ReverbAPI.h.

3.2.2 Typedef Documentation

3.2.2.1 typedef enum Hold_t_tag Hold_t

possible states of hold-effect

3.2.2.2 typedef enum Quality_t_tag Quality_t

possible quality modes to adjust workload vs. quality

3.2.2.3 typedef enum ReverbPresets_t_tag ReverbPresets_t

List of available presets (names explain the preset)

3.2.3 Enumeration Type Documentation

3.2.3.1 enum Hold_t_tag

possible states of hold-effect

Enumerator:

_HOLD_ON repeat the current reverberation state for infinity

_HOLD_OFF stop repeating the current reverberation state for infinity

_HOLD_ACCUMULATE as _HOLD_ON, but add new samples to the reverb. ∗∗ATTENTION∗∗: this will lead to clipping pretty soon!

Definition at line 74 of file ReverbAPI.h.

```
{
        _HOLD_ON,
        _HOLD_OFF,
        _HOLD_ACCUMULATE
} Hold_t;
```

3.2.3.2 enum Quality_t_tag

possible quality modes to adjust workload vs. quality

Enumerator:

_HIGHEST highest quality mode (enhanced stereo processing, default)

_HIGH high quality mode (previous default v1.x)

_LOW low quality mode (enhanced performance)

_LOWEST lowest quality mode (high performance)

Definition at line 81 of file ReverbAPI.h.

```
{
    _HIGHEST,
    _HIGH,
    _LOW,
    _LOWEST
} Quality_t;
```

3.2.3.3 enum ReverbPresets_t_tag

List of available presets (names explain the preset)

Enumerator:

_MEDIUM_HALL medium sized concert hall

_LARGE_HALL large concert hall

_PLATE rich plate

_CATHEDRAL large cathedral

_ROOM small room

_ROOM1 alternative room 1

_ROOM2 alternative room 2

Definition at line 63 of file ReverbAPI.h.

```
{
        _MEDIUM_HALL,
        _LARGE_HALL,
        _PLATE,
        _CATHEDRAL,
        _ROOM,
        _ROOM1,
        _ROOM2
} ReverbPresets_t;
```

3.2.3.4 enum ReverbVersion_t

Enumerator:

kReverbMajor

kReverbMinor

kReverbPatch

kReverbBuild

kNumReverbVersionInts

Definition at line 114 of file ReverbAPI.h.

```
{
    kReverbMajor,
    kReverbMinor,
    kReverbPatch,
    kReverbBuild,

    kNumReverbVersionInts
};
```

3.2.4 Function Documentation

3.2.4.1 int Reverb_CreateInstance (void ** *pCReverbInstance*, int *iSampleRate*, int *iNumOfChannels*, Quality_t *qQualityMode*)

creates an instance of the reverb

Parameters

pCReverb-Instance	: handle
iSampleRate	: desired samplerate
iNumOf-Channels	: desired number of channels
qQuality-Mode	: _LOW or _HIGH (default), has impact on computing performance vs. quality

Returns

static : 0 if no error occurred

3.2.4.2 int Reverb_DestroyInstance (void ** *pCReverbInstance*)

destroys an instance of the reverb

Parameters

pCReverb-Instance	: handle

Returns

static : 0 if no error occurred

3.2.4.3 float Reverb_GetBassBoost (void * *pCReverbInstance*)

returns the amplify or dampling of bass frequencies

Parameters

pCReverb-Instance	: handle

Returns

float : bass boost in dB

3.2.4.4 const char* Reverb_GetBuildDate ( )

3.2.4.5 float Reverb_GetBuildUpSteepness (void * *pCReverbInstance*)

returns the amount of build up steepness of the reverberation tail (note that this parameter only combines various other parameters)

Parameters

pCReverb-Instance	: handle

Returns

float : steepness in percent

3.2.4.6 float Reverb_GetDiffusion (void * *pCReverbInstance*)

returns the amount of diffusion in the reverberation tail

Parameters

pCReverb-Instance	: handle

Returns

float : diffusion in percent

3.2.4.7 float Reverb_GetERGain (void * *pCReverbInstance*)

returns the amount of early reflections

Parameters

pCReverb-Instance	: handle

Returns

float : amount of early reflections in dB

3.2.4.8 float Reverb_GetERLRBalance (void * *pCReverbInstance*)

returns the amount ER/LR balance

Parameters

pCReverb-Instance	: handle

Returns

float : amount of ER/LR balance between -1 and 1

3.2.4.9 float Reverb_GetHighCut (void * *pCReverbInstance*)

returns the cutoff frequency of a low pass applied to the reverberation

Parameters

pCReverb-Instance	: handle

Returns

float : high cut in Hz

3.2.4.10 Hold_t Reverb_GetHold (void * *pCReverbInstance*)

returns the hold state

Parameters

pCReverb-Instance	: handle

Returns

Hold : hold state (_HOLD_ON,_HOLD_OFF,_HOLD_ACCUMULATE

3.2.4.11 float Reverb_GetLiveness (void * *pCReverbInstance*)

returns the amount of liveness in the reverberation tail

Parameters

pCReverb-Instance	: handle

Returns

float : liveness in percent

3.2.4.12 float Reverb_GetLPAmount (void * *pCReverbInstance*)

returns the amount of low pass filtering

Parameters

pCReverb-Instance	: handle

Returns

float : low pass amount in percent

3.2.4.13 float Reverb_GetLRGain (void * *pCReverbInstance*)

returns the amount of late reverberation

Parameters

pCReverb-Instance	: handle

Returns

float : amount of late reverberation in dB

3.2.4.14 float Reverb_GetPreDelay (void * *pCReverbInstance*)

returns the predelay

Parameters

pCReverb-Instance	: handle

Returns

float : predelay in seconds

3.2.4.15 ReverbPresets_t Reverb_GetPreset (void * *pCReverbInstance*)

returns the current preset

Parameters

pCReverb-Instance	: handle

Returns

ReverbPresets : current preset

3.2.4.16 float Reverb_GetReverbTime (void * *pCReverbInstance*)

returns the reverberation time

Parameters

pCReverb-Instance	: handle

Returns

float : reverberation time in seconds

3.2.4.17 const int Reverb_GetVersion (const ReverbVersion_t *eVersionIdx*)

3.2.4.18 float Reverb_GetWetness (void * *pCReverbInstance*)

returns the amount of wet (reverberated) signal in output

Parameters

pCReverb-Instance	: handle

Returns

float : wetness in percent

3.2.4.19 int Reverb_Process (void * *pCReverbInstance,* float * *pfInputBuffer,* float * *pfOutputBuffer,* int *iNumOfFrames*)

add reverb to the input signal

Parameters

pCReverb-Instance	: handle
pfInput-Buffer	: buffer with dry input samples
pfOutput-Buffer	: buffer for output signal, may be the same as input buffer for inplace processing EXCEPT for the case MONO input / STEREO output. In this case, inplace processing is not allowed
iNumOf-Frames	: number of samples per channel

Returns

void : not _NO_ERROR in case of error (e.g. too large input buffer)

3.2.4.20 int Reverb_ProcessSplit (void * *pCReverb,* float ** *ppfInputBuffer,* float ** *ppfOutputBuffer,* int *iNumOfFrames*)

reverb_process as described above but with split buffer interface instead of interleaved buffers

3.2.4.21 void Reverb_Reset (void * *pCReverbInstance*)

resets all internal buffers

Parameters

pCReverb-Instance	: handle

Returns

void :

3.2.4.22 int Reverb_SetBassBoost (void * *pCReverbInstance,* float *fBassBoost*)

amplify or damp low frequencies in the reverberation. Large rooms often have much longer reverberation time for low frequencies than small ones.

Parameters

pCReverb-Instance	: handle
fBassBoost	: fBassBoost in dB. Valid parameter range -18...18; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fBassBoost outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.23 int Reverb_SetBuildUpSteepness (void * *pCReverbInstance,* float *fBuildUp*)

set build up steepness of the reverberation tail. Large rooms tend to have longer build-ups than small rooms. Note parameter interrelations with diffusion

Parameters

pCReverb-Instance	: handle

fBuildUp	: Build up steepness in percent; valid parameter range 0...1; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.24 int Reverb_SetDiffusion (void * *pCReverbInstance,* float *fDiffusion*)

set diffusion of the reverberation tail. Large rooms tend to have more diffuse reverberation than small rooms

Parameters

pCReverb-Instance	: handle
fDiffusion	: fDiffusion in percent; valid parameter range 0...1; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.25 int Reverb_SetERGain (void * *pCReverbInstance,* float *fERGainIndB*)

set the amount of the early reflections. Early reflections will color the reverberation tail

Parameters

pCReverb-Instance	: handle
fERGainInd-B	: fERGainIndB in dB. no parameter range restrictions, however, +-12dB seems to be useful; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fERGainIndB outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.26 int Reverb_SetERLRBalance (void * *pCReverbInstance,* float *fERLRBalance*)

set the amount of ER/LR gain relative to the current ERGain/LRGain-Settings

Parameters

pCReverb-Instance	: handle
fERLR-Balance	: value between -1 (maximum ER, minimum LR) and 1 (minimum ER, maximum LR), default value is 0 changes are dependent on current ERGain/LRGain settings!

Returns

zERROR : not _NO_ERROR in case of fERLRBalance outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.27 int Reverb_SetHighCut (void * *pCReverbInstance,* float *fLPFreq*)

set the cutoff frequency of a low pass applied to the reverberation. This may be useful for input signals with much high frequency content.

Parameters

pCReverb-Instance	: handle
fLPFreq	: fLPFreq in dB. Valid parameter range 100...min (18000, SampleRate/2.2F); default value depends on selected preset

Returns

int : not _NO_ERROR in case of fLPFreq outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.28 int Reverb_SetHold (void * *pCReverbInstance,* Hold_t *eHold*)

experimental/effect function: if hold is set to _HOLD_ON, the current reverberation tail is repeated until the function is called with parameter _HOLD_OFF. The sound color will change dependent of fLPAmount. If eHold is set to _HOLD_ACCUMULATE, the reverb tail will accumulate with the new input signal (THIS COULD LEAD TO CLIPPING VERY FAST!)

Parameters

pCReverb-Instance	: handle
eHold	: _HOLD_ON, _HOLD_OFF_, _HOLD_ACCUMULATE

Returns

int :

3.2.4.29 int Reverb_SetLiveness (void * *pCReverbInstance,* float *fLiveness*)

set the liveness of the reverberation tail. High values of liveness could lead to detuning effects in reverberation for tonal input signals.

Parameters

pCReverb-Instance	: handle
fLiveness	: fLiveness in percent; valid parameter range 0...1; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.30 int Reverb_SetLPAmount (void * *pCReverbInstance,* float *fLPAmount*)

control the amount of low pass filtering in percent; see also SetHighCut (.)

Parameters

pCReverb-Instance	: handle
fLPAmount	: fLPAmount in Percent. Valid parameter range 0.0005...0.9995; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fLPAmount outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.31 int Reverb_SetLRGain (void * *pCReverbInstance,* float *fLRGainIndB*)

set the amount of the late reverberation tail.

Parameters

pCReverb-Instance	: handle
fLRGainInd-B	: fLRGainIndB in dB. no parameter range restrictions, however, +-12dB seems to be useful; default value is 0dB

Returns

int : not _NO_ERROR in case of fLRGainIndB outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.32 int Reverb_SetPreDelay (void * *pCReverbInstance,* float *fPreDelay*)

set predelay. The Predelay is an delay inserted before the reverberation tail.

Parameters

pCReverb-Instance	: handle
fPreDelay	: fPreDelay in seconds. Valid parameter range 0...0.3; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fPreDelay outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.33 int Reverb_SetPreset (void * *pCReverbInstance,* ReverbPresets_t *ePreset*)

Load a preset

Parameters

pCReverb-Instance	: handle
ePreset	: default value is _MEDIUM_HALL

Returns

int :

3.2.4.34 int Reverb_SetReverbTime (void * *pCReverbInstance,* float *fReverbTime*)

set the reverberation time. Large rooms have a longer reverberation time than small rooms

Parameters

pCReverb-Instance	: handle
fReverbTime	: fReverbTime in seconds, valid parameter range 1e-10...100; default value depends on selected preset

Returns

int : not _NO_ERROR in case of fReverbTime outside parameter range, the parameter is then set to the minimum/maximum value

3.2.4.35 int Reverb_SetWetness (void * *pCReverbInstance,* float *fWetness*)

set the wetness. If the wetness is set to 0, the dry signal is returned; if the wetness is set to 1, the wet signal (i.e. the reverberation signal is returned)

Parameters

pCReverb-Instance	: handle
fWetness	: wetness in percent; valid parameter range 0...1; default value: 0.2

Returns

int : not _NO_ERROR in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value