fx::pack SDK 1.2.0

by zplane.development

September 22, 2016

Contents

1 fx::pack SDK Documentation

1.1 Introduction

The fx::pack SDK offers a range of realtime audio filters and effects. These include

- Filters: Lowpass (6dB/12dB), Highpass (6dB/12dB), Shelving (1st/2nd order), Bandpass (2nd order), Notch (2nd order), Peak, (2nd order)
- Special Filters: Resonance Lowpass (four pole)
- Delay, Chorus, Flanger, Phaser
- Dynamics Processing: Compressor, Expander, Limiter, Gate
- Other: Bit Crusher

The API of the effects is designed to give the SDK user maximum control over different parameters. It should be considered not to make all available parameters available (or combine them) when designing a user interface.

The SDK is available on Win32/64 platforms as well as on MacOsX (UB)

1.2 SDK Content

1.2.1 Folder Structure

1.2.1.1 Documentation

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

1.2.1.2 Project Files

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

1.2.1.3 Source Files

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

1.2.1.4 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/incl**. The main interface header of the SDK can be found in **./inc**.

1.2.1.5 Resource Files

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

1.2.1.6 Library Files

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

1.2.1.7 Binary Files

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

1.2.1.8 Temporary Files

The directory **./tmp** is for all temporary files while building the projects, structured into project and configuration names.

1.2.2 Project Structure

The project structure is as following:

- **libfxpack**: The actual fxpack-library. The following files interface to this library-:
  - ChorusFlangerIf.h: interface for chorus/flanger-like effects
  - DelayIf.h: interface for delay effect
  - DynamicsIf.h: interface for dynamics processing
  - ParamEQIf.h: interface for parametric filters
  - PhaserIf.h: interface for phaser effect
  - ResonanceLpIf.h: interface for resonance lowpass filter
  - BitCrushIf.h: interface for a bit crusher effect
- **fxpackTestCL**: Application using the SDK. Consists of the following files:
  - fxpackTestCLMain.cpp: example code showing how to integrate the various SDK interfaces.

  The project output is an executable binary (EXE).
- **libzplAudioFile**: internal library for audio IO.
  - various files

  The project output is a Static Library (Lib).

1.2.3 Project Configurations

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

1.3 API Documentation

The interface of the SDK is based on the push principle: succeeding blocks of input audio frames are pushed into the process function. Internal memory cannot be accessed from outside. The process function allows inplace and out-of-place processing of the audio samples.

The SDK is capable of running multiple instances at the same time, but the API is not threadsafe, meaning especially that the parameters should not be updated during process calls.

1.3.1 Naming Conventions

Function names are basically set together from two words "Action" + "Target", e.g. SetParam, CreateInstance, GetBypass, etc. The only exception is the Process function itself.

When talking about **frames**, the number of audio samples per channel is meant. I.e. 512 stereo frames correspond to 1024 float values (samples). If the sample size is 32bit float, one sample has a memory usage of 4 byte.

1.3.2 Effect Parameters

Every effect provides a "SetParam" function that allows you to adjust all internal floating point parameters by calling it with the appropriate index. Other parameters have a dedicated function that may be called separately.

1.3.2.1 General Parameters

The basic underlying structure of the effects Delay, Chorus, Flanger, Phaser is shown in the following figure.

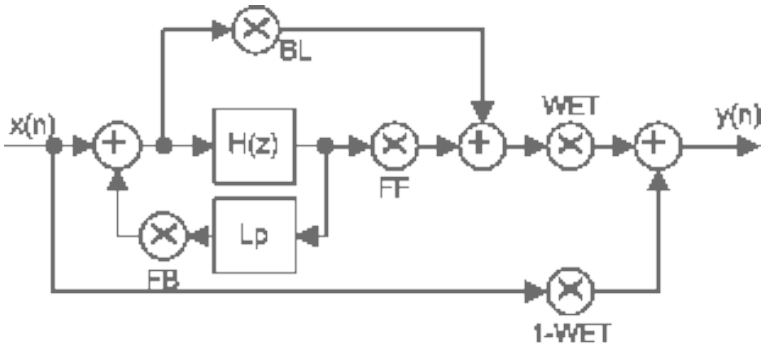

Figure 1: General Delay-Based Effect Structure

The following effect parameters can be directly mapped to this Graph:

- *ParamFeedbackRel* (FB): relative amount of feedback in the effect, meaning the amount of processed audio added to the effect input
- *ParamFeedForwardRel* (FF):
- *ParamBlendRel* (BL): relative amount of input signal in the output
- *ParamWetnessRel* (WET): relative amount of dry signal in the amount; this parameter is closely related to the combination of Blend, FeedForward, and -Feedback.
- *kChFlParamLpInFeedbackRel:* relative amount of low-pass filter in the feedback path

1.3.2.2 Modulation Parameters

Modulation Parameters are available for the following effects: Delay, Chorus, Flanger, Phaser and Resonance Lowpass. In each case, there is one oscillator (LFO) per audio channel.

- *ParamLfoDepth:* this controls the amplitude of the modulation signal. Depending on the property to be modulated and the effect, this is either a relative amount (see CDelayIf::DelParameter_t, CPhaserIf::PhParameter_t), an absolute value in seconds (see CChorusFlangerIf::ChFlParameter_t), or an absolute value in Hz (see CResonanceLpIf::RlpParameter_t)
- *ParamLfoFreqInHz:* modulation frequency

The following modulation parameters cannot be accessed via the SetParam function because the parameters do not have floating point format.

- *LfoType* (can be set with function SetLfoType): modulation waveform, as defined in zfxLfoType_t
- *PhaseOffsetBetweenChannels* (can be set with function SetLfoPhaseBetweenChannels): phase between succeeding channels, as defined in zfxPhaseOffsetBetweenChannels_t

1.3.2.3 Special Delay Parameters

Besides the parameters CDelayIf::kDelParamLfoDepthRel, CDelayIf::kDelParamLfoFreqInHz, CDelayIf::kDelParamFeedbackRel, CDelayIf::kDelParamBlendRel, CDelayIf::kDelParamFeedForwardRel, CDelayIf::kDelParamLpInFeedbackRel, CDelayIf::kDelParamWetnessRel that are explained above, the Delay effect offers two additional parameters:

- CDelayIf::kDelParamDelayInS: average delay time
- CDelayIf::kDelParamStereoFadeRel: relative amount of interchannel/cross feedback

1.3.2.4 Special Chorus and Flanger Parameters

Besides the parameters CChorusFlangerIf::kChFlParamLfoDepthInS, CChorusFlangerIf::kChFlParamLfoFreqInHz, CChorusFlangerIf::kChFlParamFeedbackRel, CChorusFlangerIf::kChFlParamBlendRel, CChorusFlangerIf::kChFlParamFeedForwardRel, CChorusFlangerIf::kChFlParamLpInFeedbackRel, CChorusFlangerIf::kChFlParamWetnessRel that are explained above, the Chorus family of effects offers two additional parameters:

- CChorusFlangerIf::kChFlParamDelayInS: average delay time
- CChorusFlangerIf::kChFlParamStereoFadeRel: relative amount of interchannel/cross feedback The interface additionally supports the setting of preset as defined in CChorusFlangerIf::ChFlPreset_t via the function CChorusFlangerIf::SetPreset.

1.3.2.5 Special Phaser Parameters

Besides the parameters CPhaserIf::kPhParamLfoDepthRel, CPhaserIf::kPhParamLfoFreqInHz, CPhaserIf::kPhParamFeedbackRel, CPhaserIf::kPhParamBlendRel, CPhaserIf::kPhParamFeedForwardRel, CPhaserIf::kPhParamLpInFeedbackRel, CPhaserIf::kPhParamWetnessRel that are explained above, the phaser effect offers two additional parameters:

- CPhaserIf::kPhParamStereoFadeRel: relative amount of interchannel/cross feedforward

An additional parameter, not being set over the CPhaserIf::SetParam function is the number of modulating stages of the phaser. The number of stages (between 1 and 10) can be set with the function CPhaserIf::SetNumOfStages.

1.3.2.6 Special Resonance Lowpass Parameters

Besides the parameters CResonanceLpIf::kRlpParamLfoDepthInHz, CResonanceLpIf::kRlpParamLfoFreqInHz that are explained above, this filter offers two additional parameters:

- CResonanceLpIf::kRlpParamFrequencyInHz: cut-off frequency of the filter
- CResonanceLpIf::kRlpParamResonance: filter resonance at cutoff

1.3.2.7 Special Parametric Equalizer Parameters

The parameters for the parametric Equalizer are:

- CParametricEqIf::kEqParamFrequency: the cutoff or mid frequency for the filter
- CParametricEqIf::kEqParamQ: the Q of the filter, if applicable; in the case of a low-pass/high pass, the Q factor deforms the transfer function at the cutoff frequency with a peak, the default Q value is

  $$\frac{1}{\sqrt{2}}$$

  In the case of a band pass filter, Q defines the bandwidth

- CParametricEqIf::kEqParamGain: the gain of the filter, if applicable (e.g. for shelv and peak filters)

The filter shape or type is specified by calling CParametricEqIf::SetType, using the types defined in CParametricEqIf::EqType_t.

1.3.2.8 Special Dynamics Processing Parameters

The parameters for the parametric Equalizer are:

- CDynamicsIf::kDynParamThreshold: the threshold for the dynamic processor
- CDynamicsIf::kDynParamRatio: the ratio (1..99) of compression or expansion
- CDynamicsIf::kDynParamAttack: the attack time in sec (0.00001..0.1)
- CDynamicsIf::kDynParamRelease: the release time in sec (0.001..4.0)
- CDynamicsIf::kDynParamGain: the gain of the dynamics processor
- CDynamicsIf::kDynParamLookAhead: look ahead in sec (0.0..0.01)

The mode of dynamic processing is specified by calling CDynamicsIf::SetMode(), using the types defined in CDynamicsIf::DynMode_t.

1.3.3 Instance Control Functions

The following functions can be used to control instances of the fx::pack library:

- **zfxError_t CreateInstance()**

  Creates a new instance of the required effect. The first parameter is the handle to be written, the second parameter fSampleRate gives the sample rate in Hz, and the third parameter iNumberOfChannels is the number of channels.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::CreateInstance
  - CDelayIf::CreateInstance
  - CParametricEqIf::CreateInstance
  - CPhaserIf::CreateInstance
  - CResonanceLpIf::CreateInstance
  - CDynamicsIf::CreateInstance

- **zfxError_t DestroyInstance()**

  Destroys an instance of the effect. The handle to the effect which is the first parameter is set to NULL.

  The function returns kNoError if no error occured.

  References:

- CChorusFlangerIf::DestroyInstance
- CDelayIf::DestroyInstance
- CParametricEqIf::DestroyInstance
- CPhaserIf::DestroyInstance
- CResonanceLpIf::DestroyInstance
- CDynamicsIf::DestroyInstance

1.3.4 Parameter Setting Functions

- **zfxError_t SetParam()**

  Sets an (floating point) effect parameter. The first parameter e∗ParameterIdx is the parameter index, the second parameter fParamValue is the value to be set.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::SetParam, CChorusFlangerIf::ChFlParameter_t
  - CDelayIf::SetParam, CDelayIf::DelParameter_t
  - CParametricEqIf::SetParam, CParametricEqIf::EqParameter_t
  - CPhaserIf::SetParam, CPhaserIf::PhParameter_t
  - CResonanceLpIf::SetParam, CResonanceLpIf::RlpParameter_t
  - CDynamicsIf::SetParam, CDynamicsIf::DynParameter_t

- **zfxError_t SetLfoType (zfxLfoType_t eLfoType)**

  Sets the waveform of the modulation signal. The parameter eLfoType gives the waveform shape as defined in zfxLfoType_t. This function is only available in CChorusFlangerIf, CDelayIf, CPhaserIf, and ::CResonanceLpIf.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::SetLfoType
  - CDelayIf::SetLfoType
  - CPhaserIf::SetLfoType
  - CResonanceLpIf::SetLfoType

- **zfxError_t SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t ePhase)**

  Sets the phase difference of the modulation signal between succeeding channels. The parameter ePhase gives the phase difference as defined in zfxPhaseOffsetBetweenChannels_t. This function is only available in CChorusFlangerIf, CDelayIf, CPhaserIf, and ::CResonanceLpIf.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::SetLfoPhaseBetweenChannels

  - CDelayIf::SetLfoPhaseBetweenChannels
  - CPhaserIf::SetLfoPhaseBetweenChannels
  - CResonanceLpIf::SetLfoPhaseBetweenChannels

- **zfxError_t SetPreset (ChFlPreset_t eChFlPreset)**

  Sets a parameter preset for the Chorus/Flanger effect. The parameter eChFlPreset gives the preset as defined in CChorusFlangerIf::ChFlPreset_t. This function is only available in CChorusFlangerIf.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::SetPreset, CChorusFlangerIf::ChFlPreset_t

- **zfxError_t SetType (EqType_t eEqType)**

  Sets the filter type/shape for the parametric filter. The parameter eEqType gives the type as defined in CParametricEqIf::EqType_t. This function is only available in CParametricEqIf.

  The function returns kNoError if no error occured.

  References:

  - CParametricEqIf::SetType, CParametricEqIf::EqType_t

- **zfxError_t SetNumOfStages (int iNumOfStages)**

  Sets the number of processing stages for the phaser effect. The parameter iNumOfStages gives number of stages. This function is only available in CPhaserIf.

  The function returns kNoError if no error occured.

  References:

  - CPhaserIf::SetNumOfStages

- **zfxError_t SetMode (DynMode_t eDynMode)**

  Sets the operation mode of the dynamic processor. The modes are defined in DynMode_t.

  References:

  - CDynamicsIf::SetMode

- **zfxError_t SetChannelLink (bool bLinkChannels)**

  Sets the channel link option for the dynamic processor. If channels are linked dynamic processing is the same for all channels.

  References:

  - CDynamicsIf::SetChannelLink

- **zfxError_t SetChannelLink (bool bLinkChannels)**

  Sets the channel link option for the dynamic processor. If channels are linked dynamic processing is the same for all channels.

  References:

  - CDynamicsIf::SetChannelLink

1.3.5 Process Function

- **zfxError_t Process (float **ppfInputBuffer, float **ppfOutputBuffer, int iNumberOfFrames)**

  Processes incoming audio data. The first parameter ppfInputBuffer contains the input audio data (of dimension [channels][frames]), the parameter ppfOutputBuffer points to the resulting audio data of the same dimension as the input data, and the parameter iNumberOfFrames gives the number of frames to be processed. ppfOutputBuffer may be equal to ppfInputBuffer for inplace processing.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::Process
  - CDelayIf::Process
  - CParametricEqIf::Process
  - CPhaserIf::Process
  - CResonanceLpIf::Process

- **zfxError_t Process (float **ppfInputBuffer, float **ppfSideChainBuffer, float **ppfOutputBuffer, int iNumberOfFrames)**

  Processes incoming audio data. The first parameter ppfInputBuffer contains the input audio data (of dimension [channels][frames]), the parameter ppfSideChainBuffer is optional (NULL is not required) and defines a sidechain input buffer (dimensions must be the same as ppfInputBuffer) the parameter ppfOutputBuffer points to the resulting audio data of the same dimension as the input data, and the parameter iNumberOfFrames gives the number of frames to be processed. ppfOutputBuffer may be equal to ppfInputBuffer for inplace processing.

  The function returns kNoError if no error occured.

  References:

  - CDynamicsIf::Process

1.3.6 Parameter Retrieving Functions

- **float GetParam()**

  Returns an (floating point) effect parameter. The first parameter e*ParameterIdx is the parameter index that is to be retrieved

  References:

  - CChorusFlangerIf::GetParam, CChorusFlangerIf::ChFlParameter_t
  - CDelayIf::GetParam, CDelayIf::DelParameter_t
  - CParametricEqIf::GetParam, CParametricEqIf::EqParameter_t
  - CPhaserIf::GetParam, CPhaserIf::PhParameter_t
  - CResonanceLpIf::GetParam, CResonanceLpIf::RlpParameter_t

- CDynamicsIf::GetParam, CDynamicsIf::DynParameter_t

- **zfxLfoType_t GetLfoType ()**

  Returns the waveform of the modulation signal as defined in zfxLfoType_t. - This function is only available in CChorusFlangerIf, CDelayIf, CPhaserIf, and ::CResonanceLpIf.

  References:

  - CChorusFlangerIf::GetLfoType
  - CDelayIf::GetLfoType
  - CPhaserIf::GetLfoType
  - CResonanceLpIf::GetLfoType

- **zfxPhaseOffsetBetweenChannels_t GetLfoPhaseBetweenChannels ()**

  Returns the phase difference of the modulation signal between succeeding channels as defined in zfxPhaseOffsetBetweenChannels_t. This function is only available in CChorusFlangerIf, CDelayIf, CPhaserIf, and ::CResonanceLpIf.

  References:

  - CChorusFlangerIf::GetLfoPhaseBetweenChannels
  - CDelayIf::GetLfoPhaseBetweenChannels
  - CPhaserIf::GetLfoPhaseBetweenChannels
  - CResonanceLpIf::GetLfoPhaseBetweenChannels

- **ChFlPreset_t GetPreset ()**

  Returns the current preset for the Chorus/Flanger effect as defined in CChorusFlangerIf::ChFlPreset_t. This function is only available in CChorusFlangerIf.

  References:

  - CChorusFlangerIf::GetPreset, CChorusFlangerIf::ChFlPreset_t

- **EqType_t GetType ( )**

  Returns the filter type/shape for the parametric filter as defined in CParametricEqIf::EqType_t. This function is only available in CParametricEqIf.

  References:

  - CParametricEqIf::GetType, CParametricEqIf::EqType_t

- **int GetNumOfStages ( )**

  Returns the number of processing stages for the phaser effect. This function is only available in CPhaserIf.

  References:

  - CPhaserIf::GetNumOfStages

- **DynMode_t GetMode ( )**

  Returns the current dynamics operation mode. References:

  - CDynamicsIf::GetType, CDynamicsIf::DynMode_t

- **bool GetChannelLink ( )**

  Returns if the channels are currently linked. References:

  – CDynamicsIf::GetChannelLink

- **zfxError_t GetCurrentLevels (int iChannelIdx, float& fReduction, float& fLevel)**

  Returns for each audio channel the current peak rms level and the current reduction/expansion in dB. The level is calculated as rms and converted to a sine peak value.

  References:

  – CDynamicsIf::GetCurrentLevels

- **float GetMagnitudeInDb (float fFrequencyInHz)**

  Returns the value of the magnitude response in dB for the specific frequency given in parameter fFrequencyInHz.

  References:

  – CParametricEqIf::GetMagnitudeInDb

1.3.6.1 Utilitary Functions

- **zfxError_t SetAddDenormalNoise (bool bAddNoise = true)**

  Switching the internal addition of a small amplitude noise on or off. The parameter bAddNoise can be set to true or false.

  The function returns kNoError if no error occured.

  References:

  – CChorusFlangerIf::SetAddDenormalNoise
  – CDelayIf::SetAddDenormalNoise
  – CParametricEqIf::SetAddDenormalNoise
  – CPhaserIf::SetAddDenormalNoise
  – CResonanceLpIf::SetAddDenormalNoise

- **bool GetAddDenormalNoise ()**

  Return current status of denormal noise additional, the result can be true or false.

  References:

  – CChorusFlangerIf::GetAddDenormalNoise
  – CDelayIf::GetAddDenormalNoise
  – CParametricEqIf::GetAddDenormalNoise
  – CPhaserIf::GetAddDenormalNoise
  – CResonanceLpIf::GetAddDenormalNoise

- **zfxError_t SetBypass (bool bBypass = false)**

  Switching between bypass mode or normal processing mode. The parameter bBypass can be set to true or false.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::SetBypass
  - CDelayIf::SetBypass
  - CParametricEqIf::SetBypass
  - CPhaserIf::SetBypass
  - CResonanceLpIf::SetBypass
  - CDynamicsIf::SetBypass

- **bool GetBypass ()**

  Return current status of bypass, the result can be true or false.

  References:

  - CChorusFlangerIf::GetBypass
  - CDelayIf::GetBypass
  - CParametricEqIf::GetBypass
  - CPhaserIf::GetBypass
  - CResonanceLpIf::GetBypass
  - CDynamicsIf::GetBypass

- **zfxError_t Reset ()**

  Reset all internal buffers.

  The function returns kNoError if no error occured.

  References:

  - CChorusFlangerIf::Reset
  - CDelayIf::Reset
  - CParametricEqIf::Reset
  - CPhaserIf::Reset
  - CResonanceLpIf::Reset
  - CDynamicsIf::Reset

1.3.6.2 Other Functions

- **char* GetVersionString ()**

  Returns the library version in a string. In the current SDK, this will return the same version number for all effects.

  References:

  - CChorusFlangerIf::GetVersionString

  - CDelayIf::GetVersionString
  - CParametricEqIf::GetVersionString
  - CPhaserIf::GetVersionString
  - CResonanceLpIf::GetVersionString

- **char∗ GetBuildDateString ()**

  Returns the build date in a string. In the current SDK, this will return the same build date for all effects.

  References:

  - CChorusFlangerIf::GetBuildDateString
  - CDelayIf::GetBuildDateString
  - CParametricEqIf::GetBuildDateString
  - CPhaserIf::GetBuildDateString
  - CResonanceLpIf::GetBuildDateString

1.3.7 Usage Example

The complete code can be found in the example source file fxpackTestCLMain.cpp. For audio file IO, an internal (C++) library is used.

In the first step, a pointer to each fx::pack instance has to be declared:

In this command line example, we define which of the effects should be applied to the input audio by setting the following variable

We use a wrapped open source library sndlib for file IO parsing here, so the input file is opened with

The following example makes use of the parametric EQ, the code for other interface classes has been omitted for the sake of simplicity.

First, we have to create a new instance

After successful instanciation, we are able to set the available options and parameters for the parametric EQ. In this case, we select a 12dB per octave lowpass with a cut-off frequency of 500Hz, with no amplitude modification at the cutoff frequency.

To retrieve the current status, the following function calls may be used

In the processing loop, we read succeeding blocks of audio data

and feed them to the processing function

After successful processing, the instance can be destroyed:

The above code snippets demonstrated the basic functionality of the fx::pack library. Most of the additional functions can be used similar to the given code examples. The exact functionality of the functions is described above.

1.3.8 Error Codes

Possible error codes are defined in zfxError_t.

1.4 Third Party Libraries

The command line example application uses `sndlib` for audio file IO. Please ensure license compliance if you intend to use it in your application as well.

1.5 Support

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

`zplane.development`

grunewaldstr.83

d-10823 berlin

germany

fon: +49.30.854 09 15.0

fax: +49.30.854 09 15.5

@: `info@zplane.de`

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 CBitCrushIf Class Reference

```
#include <BitCrushIf.h>
```

Public Types

- enum BitCrushParameter_t { kBitCrushOutputBits, kBitCrushDownsampleFactor, kBitCrushClippingThreshold, kBitCrushWrapAround, kNumOfBitCrushParameters }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer=0, float **ppfOutputBuffer=0, int iNumberOfFrames=0)=0
- virtual zfxError_t SetParam (BitCrushParameter_t eBitCrushParameterIdx, float fParamValue)=0
- virtual float GetParam (BitCrushParameter_t eBitCrushParameterIdx)=0
- virtual zfxError_t SetBypass (bool bBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CBitCrushIf *&pCBitCrushIf, float fSampleRate, int iNumberOfChannels)
- static zfxError_t DestroyInstance (CBitCrushIf *&pCBitCrushIf)

4.1.1 Detailed Description

Definition at line 40 of file BitCrushIf.h.

4.1.2 Member Enumeration Documentation

4.1.2.1 enum CBitCrushIf::BitCrushParameter_t

available BitCrush parameters

Enumerator:

kBitCrushOutputBits Output bit depth as int value.

kBitCrushDownsampleFactor Downsample factor as int value.

kBitCrushClippingThreshold Clipping threshold in dB as float.

kBitCrushWrapAround Enable overflow of values above clipping threshold.

kNumOfBitCrushParameters

Definition at line 45 of file BitCrushIf.h.

```
{
    kBitCrushOutputBits,
    kBitCrushDownsampleFactor,
    kBitCrushClippingThreshold,
```

```
    kBitCrushWrapAround,
                                        //   false: hard clipping
                                        //   true:  wrap around clipping

    kNumOfBitCrushParameters
};
```

4.1.3 Member Function Documentation

4.1.3.1 static zfxError_t CBitCrushIf::CreateInstance (CBitCrushIf *& *pCBitCrushIf,* float *fSampleRate,* int *iNumberOfChannels*) [static]

create new instance of one filter

Parameters

pCBitCrush-If	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data

Returns

zfxError_t : kNoError if no error

4.1.3.2 static zfxError_t CBitCrushIf::DestroyInstance (CBitCrushIf *& *pCBitCrushIf*) [static]

destroy an instance

Parameters

pCBitCrush-If	: handle to instance to be destroyed

Returns

zfxError_t : kNoError if no error

4.1.3.3 virtual bool CBitCrushIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t : kNoError if no error

4.1.3.4 virtual float CBitCrushIf::GetParam (BitCrushParameter_t *eBitCrushParameterIdx*) [pure virtual]

return current filter parameter

Parameters

eBitCrush-Parameter-Idx	: parameter index as defined in CBitCrushIf::BitCrushParameter_t

Returns

float : corresponding parameter value

4.1.3.5 virtual zfxError_t CBitCrushIf::Process (float ** *ppfInputBuffer* = 0, float ** *ppfOutputBuffer* = 0, int *iNumberOfFrames* = 0) [pure virtual]

processing function

Parameters

ppfInput-Buffer	: input audio buffer of dimension [channels][frames]
ppfOutput-Buffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOf-Frames	: number of frames (=samples per channel)

Returns

zfxError_t : kNoError if no error

4.1.3.6 virtual zfxError_t CBitCrushIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t : kNoError if no error

4.1.3.7 virtual zfxError_t CBitCrushIf::SetBypass (bool *bBypass* = false) [pure virtual]

set effect to bypass

Returns

zfxError_t : kNoError if no error

4.1.3.8 virtual zfxError_t CBitCrushIf::SetParam (BitCrushParameter_t *eBitCrushParameterIdx*, float *fParamValue*) [pure virtual]

set filter parameter (parameters are updated once per process call)

Parameters

eBitCrush-Parameter-Idx	: parameter index as defined in CBitCrushIf::BitCrushParameter_t
fParamValue	: parameter value

Returns

zfxError_t : kNoError if no error

The documentation for this class was generated from the following file:

- BitCrushIf.h

4.2 CChorusFlangerIf Class Reference

`#include <ChorusFlangerIf.h>`

Public Types

- enum ChFlPreset_t { kChorus, kWhiteChorus, kFlanger1, kFlanger2, kVibrato, kNumOfChFlPresets }
- enum ChFlParameter_t { kChFlParamLfoDepthInS, kChFlParamLfoFreqInHz, kChFlParamDelayInS, kChFlParamFeedbackRel, kChFlParamFeedForwardRel, kChFlParamBlendRel, kChFlParamLpInFeedbackRel, kChFlParamStereoFadeRel, kChFlParamWetnessRel, kNumOfChFlParameters }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer, float **ppfOutputBuffer, int iNumberOfFrames)=0
- virtual zfxError_t SetPreset (ChFlPreset_t eChFlPreset)=0
- virtual ChFlPreset_t GetPreset ()=0
- virtual zfxError_t SetLfoType (zfxLfoType_t eLfoType)=0
- virtual zfxLfoType_t GetLfoType ()=0
- virtual zfxError_t SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t ePhase)=0
- virtual zfxPhaseOffsetBetweenChannels_t GetLfoPhaseBetweenChannels ()=0
- virtual zfxError_t SetParam (ChFlParameter_t eChFlParameterIdx, float fParamValue)=0
- virtual float GetParam (ChFlParameter_t eChFlParameterIdx)=0

- virtual zfxError_t SetAddDenormalNoise (bool bAddNoise=true)=0
- virtual bool GetAddDenormalNoise ()=0
- virtual zfxError_t SetBypass (bool bBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CChorusFlangerIf *&pCChorusFlanger, float fSampleRate, int iNumberOfChannels)
- static zfxError_t DestroyInstance (CChorusFlangerIf *&pCChorusFlanger)

4.2.1 Detailed Description

Definition at line 40 of file ChorusFlangerIf.h.

4.2.2 Member Enumeration Documentation

4.2.2.1 enum CChorusFlangerIf::ChFlParameter_t

available ChFl parameters

Enumerator:

kChFlParamLfoDepthInS modulation range peak to peak that is added to delay (0...0.1) leads to (delaytime...delaytime+LfoDepth)

kChFlParamLfoFreqInHz modulation frequency (0...20)

kChFlParamDelayInS delay (0...0.150)

kChFlParamFeedbackRel amount of feedback between (0...0.9999)

kChFlParamFeedForwardRel amount of feedforward (modulation output) (0...1)

kChFlParamBlendRel amount of blend (modulation input) (0...1)

kChFlParamLpInFeedbackRel amount of low pass filtering in feedback (0...0.9999F)

kChFlParamStereoFadeRel amount of cross feedback between channels (0...1)

kChFlParamWetnessRel amount of effect (0...1) (this is closely related to blend)

kNumOfChFlParameters

Definition at line 56 of file ChorusFlangerIf.h.

```
{
    kChFlParamLfoDepthInS,
    kChFlParamLfoFreqInHz,
    kChFlParamDelayInS,
    kChFlParamFeedbackRel,
```

```
    kChFlParamFeedForwardRel,
    kChFlParamBlendRel,
    kChFlParamLpInFeedbackRel,
    kChFlParamStereoFadeRel,
    kChFlParamWetnessRel,

    kNumOfChFlParameters
};
```

4.2.2.2 enum CChorusFlangerIf::ChFlPreset_t

provided presets = parameter initialization

Enumerator:

kChorus default chorus

kWhiteChorus white chorus (feedback)

kFlanger1 flanger

kFlanger2 flanger (feedback)

kVibrato vibrato

kNumOfChFlPresets

Definition at line 44 of file ChorusFlangerIf.h.

```
{
    kChorus,
    kWhiteChorus,
    kFlanger1,
    kFlanger2,
    kVibrato,

    kNumOfChFlPresets
};
```

4.2.3 Member Function Documentation

4.2.3.1 static zfxError_t CChorusFlangerIf::CreateInstance (CChorusFlangerIf *& *pCChorusFlanger,* float *fSampleRate,* int *iNumberOfChannels*) [static]

create new instance

Parameters

pCChorus-Flanger	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data

Returns

zfxError_t : kNoError if no error

4.2.3.2 static zfxError_t CChorusFlangerIf::DestroyInstance (CChorusFlangerIf *& *pCChorusFlanger*) [static]

destroy an instance

Parameters

pCChorus-Flanger	: handle to instance to be destroyed

Returns

zfxError_t : kNoError if no error

4.2.3.3 virtual bool CChorusFlangerIf::GetAddDenormalNoise ( ) [pure virtual]

return if noise addition to avoid denormals is avtivated or not

Returns

bool : true if activated

4.2.3.4 virtual bool CChorusFlangerIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t : kNoError if no error

4.2.3.5 virtual zfxPhaseOffsetBetweenChannels_t CChorus-FlangerIf::GetLfoPhaseBetweenChannels ( ) [pure virtual]

return current phase difference between channels

Returns

zfxPhaseOffsetBetweenChannels_t : phase between succeeding channels

4.2.3.6 virtual zfxLfoType_t CChorusFlangerIf::GetLfoType ( ) [pure virtual]

return current lfo waveform

Returns

zfxLfoType_t : lfo type

4.2.3.7 virtual float CChorusFlangerIf::GetParam (ChFlParameter_t *eChFlParameterIdx*) [pure virtual]

return current parameter

Parameters

eChFlParameterIdx	: parameter index as defined in CChorusFlangerIf::ChFlParameter_t

Returns

float : corresponding parameter value

4.2.3.8 virtual ChFlPreset_t CChorusFlangerIf::GetPreset ( ) [pure virtual]

return current preset (is also returned when modified)

Returns

ChFlPreset_t : current filter type as defined in CChorusFlangerIf::ChFlPreset_t

4.2.3.9 virtual zfxError_t CChorusFlangerIf::Process (float ** *ppfInputBuffer,* float ** *ppfOutputBuffer,* int *iNumberOfFrames*) [pure virtual]

processing function

Parameters

ppfInputBuffer	: input audio buffer of dimension [channels][frames]
ppfOutputBuffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOfFrames	: number of frames (=samples per channel)

Returns

zfxError_t : kNoError if no error

4.2.3.10 virtual zfxError_t CChorusFlangerIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t : kNoError if no error

4.2.3.11 virtual zfxError_t CChorusFlangerIf::SetAddDenormalNoise (bool *bAddNoise* = `true`) [pure virtual]

set optional noise addition to avoid denormals

Parameters

bAddNoise	: true if noise should be added

Returns

EqError_t : kNoError if no error

4.2.3.12 virtual zfxError_t CChorusFlangerIf::SetBypass (bool *bBypass* = `false`) [pure virtual]

set effect to bypass mode

Returns

zfxError_t : kNoError if no error

4.2.3.13 virtual zfxError_t CChorusFlangerIf::SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t *ePhase*) [pure virtual]

set phase difference between succeeding channels

Parameters

ePhase	: filter type as defined in zfxPhaseOffsetBetweenChannels_t

Returns

zfxError_t : kNoError if no error

4.2.3.14 virtual zfxError_t CChorusFlangerIf::SetLfoType (zfxLfoType_t *eLfoType*) [pure virtual]

set lfo waveform

Parameters

eLfoType	: filter type as defined in zfxLfoType_t

Returns

zfxError_t : kNoError if no error

4.2.3.15 virtual zfxError_t CChorusFlangerIf::SetParam (ChFlParameter_t *eChFlParameterIdx*, float *fParamValue*) [pure virtual]

set parameter

Parameters

eChFl-Parameter-Idx	: parameter index as defined in CChorusFlangerIf::ChFlParameter_t
fParamValue	: parameter value

Returns

zfxError_t : kNoError if no error

4.2.3.16 virtual zfxError_t CChorusFlangerIf::SetPreset (ChFlPreset_t *eChFlPreset*) [pure virtual]

set preset (different parameter initializations)

Parameters

eChFlPreset	: filter type as defined in CChorusFlangerIf::ChFlPreset_t

Returns

zfxError_t : kNoError if no error

The documentation for this class was generated from the following file:

- ChorusFlangerIf.h

4.3 CDelayIf Class Reference

```
#include <DelayIf.h>
```

Public Types

- enum DelParameter_t { kDelParamLfoDepthRel, kDelParamLfoFreqInHz, kDelParamDelayInS, kDelParamFeedbackRel, kDelParamBlendRel, kDelParamFeedForwardRel, kDelParamLpInFeedbackRel, kDelParamStereoFadeRel, kDelParamWetnessRel, kNumOfDelParameters }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer, float **ppfOutputBuffer, int iNumberOfFrames)=0
- virtual zfxError_t SetLfoType (zfxLfoType_t eLfoType)=0
- virtual zfxLfoType_t GetLfoType ()=0
- virtual zfxError_t SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t ePhase)=0
- virtual zfxPhaseOffsetBetweenChannels_t GetLfoPhaseBetweenChannels ()=0
- virtual zfxError_t SetParam (DelParameter_t eDelParameterIdx, float fParamValue)=0
- virtual float GetParam (DelParameter_t eDelParameterIdx)=0
- virtual zfxError_t SetAddDenormalNoise (bool bAddNoise=true)=0
- virtual bool GetAddDenormalNoise ()=0
- virtual zfxError_t SetBypass (bool bBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CDelayIf *&pCDelay, float fSampleRate, int iNumberOfChannels, float fMaxDelayInS=-1.F, float fMaxLfoDepthRel=-1.F)
- static zfxError_t DestroyInstance (CDelayIf *&pCDelay)

4.3.1 Detailed Description

Definition at line 40 of file DelayIf.h.

4.3.2 Member Enumeration Documentation

4.3.2.1 enum CDelayIf::DelParameter_t

available Delay parameters

Enumerator:

kDelParamLfoDepthRel modulation range in relation to current delay, (0...1) leads to (0s...2*delaytime)

kDelParamLfoFreqInHz modulation frequency

kDelParamDelayInS mean delay (0...fMaxDelay in CreateInstance)

kDelParamFeedbackRel amount of feedback (0...0.9999)

kDelParamBlendRel amount of dry (i.e. input + feedback) path (0...1)

kDelParamFeedForwardRel amount of wet path (0...1)

kDelParamLpInFeedbackRel low pass amount in feedback path (0...0.9999)

kDelParamStereoFadeRel amount of cross feedback between channels (0...1)

kDelParamWetnessRel amount of wetness (0...1), (this is closely related to blend)

kNumOfDelParameters

Definition at line 45 of file DelayIf.h.

```
{
    kDelParamLfoDepthRel,
    kDelParamLfoFreqInHz,
    kDelParamDelayInS,
    kDelParamFeedbackRel,
    kDelParamBlendRel,
    kDelParamFeedForwardRel,
    kDelParamLpInFeedbackRel,
    kDelParamStereoFadeRel,
    kDelParamWetnessRel,

    kNumOfDelParameters
};
```

4.3.3 Member Function Documentation

4.3.3.1 static zfxError_t CDelayIf::CreateInstance (CDelayIf *& *pCDelay*, float *fSampleRate*, int *iNumberOfChannels*, float *fMaxDelayInS* = -1.F, float *fMaxLfoDepthRel* = -1.F) [static]

create new instance

Parameters

pCDelay	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data
fMaxDelay-InS	: maximum length of delay in seconds (sample number is rounded to a power of two)
fMaxLfo-DepthRel	: maximum modulation amplitude wrt fMaxDelayInS (see kDelParam-LfoDepthRel)

Returns

zfxError_t : kNoError if no error

4.3.3.2 static zfxError_t CDelayIf::DestroyInstance (CDelayIf *& *pCDelay*) [static]

destroy an instance

Parameters

pCDelay	: handle to instance to be destroyed

Returns

zfxError_t : kNoError if no error

4.3.3.3 virtual bool CDelayIf::GetAddDenormalNoise ( ) [pure virtual]

return if noise addition to avoid denormals is avtivated or not

Returns

bool : true if activated

4.3.3.4 virtual bool CDelayIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t : kNoError if no error

4.3.3.5 virtual zfxPhaseOffsetBetweenChannels_t CDelayIf::GetLfoPhaseBetweenChannels ( ) [pure virtual]

return current phase difference between channels

Returns

zfxPhaseOffsetBetweenChannels_t : phase between succeeding channels

4.3.3.6 virtual zfxLfoType_t CDelayIf::GetLfoType ( ) [pure virtual]

return current lfo waveform

Returns

zfxLfoType_t : lfo type

4.3.3.7 virtual float CDelayIf::GetParam (DelParameter_t *eDelParameterIdx*) [pure virtual]

return current parameter

Parameters

eDelParameterIdx	: parameter index as defined in CDelayIf::DelParameter_t

Returns

float : corresponding parameter value

4.3.3.8 virtual zfxError_t CDelayIf::Process (float ** *ppfInputBuffer,* float ** *ppfOutputBuffer,* int *iNumberOfFrames*) [pure virtual]

processing function

Parameters

ppfInputBuffer	: input audio buffer of dimension [channels][frames]
ppfOutputBuffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOfFrames	: number of frames (=samples per channel)

Returns

zfxError_t : kNoError if no error

4.3.3.9 virtual zfxError_t CDelayIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t : kNoError if no error

4.3.3.10 virtual zfxError_t CDelayIf::SetAddDenormalNoise (bool *bAddNoise* = true) [pure virtual]

set optional noise addition to avoid denormals

Parameters

bAddNoise	: true if noise should be added

Returns

EqError_t : kNoError if no error

4.3.3.11 virtual zfxError_t CDelayIf::SetBypass (bool *bBypass* = false) [pure virtual]

set effect to bypass mode

Returns

zfxError_t : kNoError if no error

4.3.3.12 virtual zfxError_t CDelayIf::SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t *ePhase*) [pure virtual]

set phase difference between succeeding channels

Parameters

ePhase	: filter type as defined in zfxPhaseOffsetBetweenChannels_t

Returns

zfxError_t : kNoError if no error

4.3.3.13 virtual zfxError_t CDelayIf::SetLfoType (zfxLfoType_t *eLfoType*) [pure virtual]

set lfo waveform

Parameters

eLfoType	: filter type as defined in zfxLfoType_t

Returns

zfxError_t : kNoError if no error

4.3.3.14 virtual zfxError_t CDelayIf::SetParam (DelParameter_t *eDelParameterIdx*, float *fParamValue*) [pure virtual]

set parameter

Parameters

eDelParameterIdx	: parameter index as defined in CDelayIf::DelParameter_t
fParamValue	: parameter value

Returns

zfxError_t : kNoError if no error

The documentation for this class was generated from the following file:

- DelayIf.h

4.4 CDynamicsIf Class Reference

`#include <DynamicsIf.h>`

Public Types

- enum DynParameter_t { kDynParamThreshold, kDynParamRatio, kDynParamAttack, kDynParamRelease, kDynParamGain, kDynParamLookAhead, kNumOfDynParameters }
- enum DynMode_t { kDynModeCompressor, kDynModeLimiter, kDynModeExpander, kDynModeGate, kNumOfDynModes }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer, float **ppfSideChainBuffer, float **ppfOutputBuffer, int iNumberOfFrames)=0
- virtual zfxError_t GetCurrentLevels (int iChannelIdx, float &fReduction, float &fLevel)=0
- virtual zfxError_t SetMode (DynMode_t eDynMode)=0
- virtual DynMode_t GetMode ()=0
- virtual zfxError_t SetChannelLink (bool bLinkChannels)=0
- virtual bool GetChannelLink ()=0
- virtual zfxError_t SetParam (DynParameter_t eDynParameterIdx, float fParamValue)=0
- virtual float GetParam (DynParameter_t eDynParameterIdx)=0
- virtual zfxError_t SetBypass (bool eBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CDynamicsIf *&pCDynamics, float fSampleRate, int iNumberOfChannels)
- static zfxError_t DestroyInstance (CDynamicsIf *&pCDynamics)

4.4.1 Detailed Description

Definition at line 40 of file DynamicsIf.h.

4.4.2 Member Enumeration Documentation

4.4.2.1 enum CDynamicsIf::DynMode_t

available Dyn modes

Enumerator:

kDynModeCompressor compressor mode (all parameters)

kDynModeLimiter limiter mode (ratio parameter is not available)

kDynModeExpander expander mode (all parameters)

kDynModeGate noise gate mode (ratio parameter is not available)

kNumOfDynModes

Definition at line 58 of file DynamicsIf.h.

```
{
    kDynModeCompressor,
    kDynModeLimiter,
    kDynModeExpander,
    kDynModeGate,

    kNumOfDynModes
};
```

4.4.2.2 enum CDynamicsIf::DynParameter_t

available Dyn parameters

Enumerator:

kDynParamThreshold the operation threshold in dB (0..-120)

kDynParamRatio the ration of dynamic processing, only available for the compressor and expander modes (1..100), in compressor mode it is 1:r ratio and in expander mode r:1.

kDynParamAttack the attack time of the dynamic processor in sec (0.00001..0.-1), the lowest value is recommended for limiter operation

kDynParamRelease the release time of the dynamic processor in sec (0.001..4.0)

kDynParamGain the post processing gain in dB (-80..80)

kDynParamLookAhead the look ahead time in sec (0.0..0.01)

kNumOfDynParameters

Definition at line 45 of file DynamicsIf.h.

```
{
    kDynParamThreshold,
    kDynParamRatio,
    kDynParamAttack,
    kDynParamRelease,
    kDynParamGain,
    kDynParamLookAhead,

    kNumOfDynParameters
};
```

4.4.3 Member Function Documentation

4.4.3.1 static zfxError_t CDynamicsIf::CreateInstance (CDynamicsIf *& *pCDynamics*, float *fSampleRate*, int *iNumberOfChannels*) [static]

create new instance

Parameters

pCDynamics	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data

Returns

zfxError_t : kNoError if no error

4.4.3.2 static zfxError_t CDynamicsIf::DestroyInstance (CDynamicsIf *& *pCDynamics*) [static]

destroy an instance

Parameters

pCDynamics	: handle to instance to be destroyed

Returns

zfxError_t : kNoError if no error

4.4.3.3 virtual bool CDynamicsIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t : kNoError if no error

4.4.3.4 virtual bool CDynamicsIf::GetChannelLink ( ) [pure virtual]

return channel link state

Returns

zfxError_t : kNoError if no error

4.4.3.5 virtual zfxError_t CDynamicsIf::GetCurrentLevels (int *iChannelIdx*, float & *fReduction*, float & *fLevel*) [pure virtual]

returns max. level and max reduction for requested channel for each block

Parameters

iChannelIdx	: index of channel of interest
fReduction	: current reduction in dB
fLevel	: current level in dB

Returns

zfxError_t : kNoError if no error

4.4.3.6 virtual DynMode_t CDynamicsIf::GetMode ( ) [pure virtual]

return current operation mode

Returns

zfxError_t : kNoError if no error

4.4.3.7 virtual float CDynamicsIf::GetParam (DynParameter_t *eDynParameterIdx*) [pure virtual]

return current parameter

Parameters

eDynParameterIdx	: parameter index as defined in CDynamicsIf::DynParameter_t

Returns

float : corresponding parameter value

4.4.3.8 virtual zfxError_t CDynamicsIf::Process (float ** *ppfInputBuffer*, float ** *ppfSideChainBuffer*, float ** *ppfOutputBuffer*, int *iNumberOfFrames*) [pure virtual]

processing function

Parameters

ppfInputBuffer	: input audio buffer of dimension [channels][frames]
ppfSideChainBuffer	: side chain audio buffer of dimension [channels][frames], set to NULL if there is no side chain
ppfOutputBuffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOfFrames	: number of frames (=samples per channel)

Returns

zfxError_t : kNoError if no error

4.4.3.9 virtual zfxError_t CDynamicsIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t : kNoError if no error

4.4.3.10 virtual zfxError_t CDynamicsIf::SetBypass (bool *eBypass* = false) [pure virtual]

set effect to bypass mode

Returns

zfxError_t : kNoError if no error

4.4.3.11 virtual zfxError_t CDynamicsIf::SetChannelLink (bool *bLinkChannels*) [pure virtual]

enable/disable channel link

Parameters

bLinkChannels	: enable/disable

Returns

zfxError_t : kNoError if no error

4.4.3.12 virtual zfxError_t CDynamicsIf::SetMode (DynMode_t *eDynMode*) [pure virtual]

set operation mode

Parameters

eDynMode	: operation mode

Returns

zfxError_t : kNoError if no error

4.4.3.13 virtual zfxError_t CDynamicsIf::SetParam (DynParameter_t *eDynParameterIdx*, float *fParamValue*) [pure virtual]

set parameter

Parameters

eDyn-Parameter-Idx	: parameter index as defined in CDynamicsIf::DynParameter_t
fParamValue	: parameter value

Returns

zfxError_t : kNoError if no error

The documentation for this class was generated from the following file:

- DynamicsIf.h

4.5 CParametricEqIf Class Reference

`#include <ParamEQIf.h>`

Public Types

- enum EqType_t { kLP6, kLP12, kHP6, kHP12, kBP12, kLShelv6, kLShelv12, kHShelv6, kHShelv12, kPeak12, kNotch12, kAP6, kAP12, kNumOfEqTypes }
- enum EqParameter_t { kEqParamFrequency, kEqParamQ, kEqParamGain, kNumOfEqParameters }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer=0, float **ppfOutputBuffer=0, int iNumberOfFrames=0)=0
- virtual zfxError_t SetType (EqType_t eEqType)=0
- virtual EqType_t GetType ()=0
- virtual zfxError_t SetParam (EqParameter_t eEqParameterIdx, float fParamValue)=0
- virtual float GetParam (EqParameter_t eEqParameterIdx)=0
- virtual float GetMagnitudeInDb (float fFrequencyInHz)=0
- virtual zfxError_t SetAddDenormalNoise (bool bAddNoise=true)=0
- virtual bool GetAddDenormalNoise ()=0

- virtual zfxError_t SetBypass (bool bBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CParametricEqIf *&pCParamEQ, float fSampleRate, int iNumberOfChannels)
- static zfxError_t DestroyInstance (CParametricEqIf *&pCParamEQ)

4.5.1 Detailed Description

Definition at line 40 of file ParamEQIf.h.

4.5.2 Member Enumeration Documentation

4.5.2.1 enum CParametricEqIf::EqParameter_t

available EQ parameters, not all are always available, see CParametricEqIf::EqTypes_t

Enumerator:

kEqParamFrequency Frequency in Hz, CutOff or Mid (20...20kHz)

kEqParamQ Q (0.1...24)

kEqParamGain amplification in dB (-24...24)

kNumOfEqParameters

Definition at line 64 of file ParamEQIf.h.

```
{
    kEqParamFrequency,
    kEqParamQ,
    kEqParamGain,

    kNumOfEqParameters
};
```

4.5.2.2 enum CParametricEqIf::EqType_t

provided filter shapes and orders

Enumerator:

kLP6 first order low pass (6dB/oct), parameters: f_c

kLP12 second order low pass (12dB/oct), parameters: f_c, Q (Resonance)

kHP6 first order high pass (6dB/oct), parameters: f_c

kHP12 second order high pass (12dB/oct), parameters: f_c, Q (Resonance)

kBP12 second order band pass, parameters: f_c, Q

kLShelv6 first order low shelving, parameters: f_c, Gain

kLShelv12 second order low shelving, parameters: f_c, Q, Gain

kHShelv6 first order high shelving, parameters: f_c, Gain

kHShelv12 second order high shelving, parameters: f_c, Q, Gain

kPeak12 second order peak, parameters: f_c, Q, Gain

kNotch12 second order notch, parameters: f_c, Q

kAP6 first order all pass, parameters: f_c

kAP12 second order all pass, parameters: f_c, Q

kNumOfEqTypes

Definition at line 44 of file ParamEQIf.h.

```
{
    kLP6,
    kLP12,
    kHP6,
    kHP12,
    kBP12,
    kLShelv6,
    kLShelv12,
    kHShelv6,
    kHShelv12,
    kPeak12,
    kNotch12,
    kAP6,
    kAP12,

    kNumOfEqTypes
};
```

4.5.3 Member Function Documentation

4.5.3.1 static zfxError_t CParametricEqIf::CreateInstance (CParametricEqIf *& *pCParamEQ,* float *fSampleRate,* int *iNumberOfChannels*) [static]

create new instance of one filter

Parameters

pCParamEQ	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data

Returns

zfxError_t : kNoError if no error

4.5.3.2 static zfxError_t CParametricEqIf::DestroyInstance (CParametricEqIf *& *pCParamEQ*) [static]

destroy an instance

Parameters

pCParamEQ	: handle to instance to be destroyed

Returns

zfxError_t : kNoError if no error

4.5.3.3 virtual bool CParametricEqIf::GetAddDenormalNoise ( ) [pure virtual]

return if noise addition to avoid denormals is avtivated or not

Returns

bool : true if activated

4.5.3.4 virtual bool CParametricEqIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t : kNoError if no error

4.5.3.5 virtual float CParametricEqIf::GetMagnitudeInDb (float *fFrequencyInHz*) [pure virtual]

return magnitude response at specific frequency

Parameters

fFrequency-InHz	: frequency in Hz to be evaluated

Returns

float : magnitude response in dB

4.5.3.6 virtual float CParametricEqIf::GetParam (EqParameter_t *eEqParameterIdx*) [pure virtual]

return current filter parameter

Parameters

eEqParameterIdx	: parameter index as defined in CParametricEqIf::EqParameter_t

Returns

float : corresponding parameter value

4.5.3.7 virtual EqType_t CParametricEqIf::GetType ( ) [pure virtual]

return current filter type

Returns

EqType_t : current filter type as defined in CParametricEqIf::EqType_t

4.5.3.8 virtual zfxError_t CParametricEqIf::Process (float ** *ppfInputBuffer* = 0, float ** *ppfOutputBuffer* = 0, int *iNumberOfFrames* = 0) [pure virtual]

processing function, if called with no input arguments, only the coefficients are updated internally (if necessary)

Parameters

ppfInputBuffer	: input audio buffer of dimension [channels][frames]
ppfOutputBuffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOfFrames	: number of frames (=samples per channel)

Returns

zfxError_t : kNoError if no error

4.5.3.9 virtual zfxError_t CParametricEqIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t : kNoError if no error

4.5.3.10 virtual zfxError_t CParametricEqIf::SetAddDenormalNoise (bool *bAddNoise* = true) [pure virtual]

set optional noise addition to avoid denormals

Parameters

bAddNoise	: true if noise should be added

Returns

zfxError_t : kNoError if no error

4.5.3.11 virtual zfxError_t CParametricEqIf::SetBypass (bool *bBypass* = `false`) `[pure virtual]`

set effect to bypass

Returns

zfxError_t : kNoError if no error

4.5.3.12 virtual zfxError_t CParametricEqIf::SetParam (EqParameter_t *eEqParameterIdx*, float *fParamValue*) `[pure virtual]`

set filter parameter (parameters are updated once per process call)

Parameters

eEqParameterIdx	: parameter index as defined in CParametricEqIf::EqParameter_t
fParamValue	: parameter value

Returns

zfxError_t : kNoError if no error

4.5.3.13 virtual zfxError_t CParametricEqIf::SetType (EqType_t *eEqType*) `[pure virtual]`

set filter type

Parameters

eEqType	: filter type as defined in CParametricEqIf::EqType_t

Returns

zfxError_t : kNoError if no error

The documentation for this class was generated from the following file:

- ParamEQIf.h

4.6 CPhaserIf Class Reference

`#include <PhaserIf.h>`

Public Types

- enum PhParameter_t { kPhParamLfoDepthRel, kPhParamLfoFreqInHz, kPhParamFeedbackRel, kPhParamFeedForwardRel, kPhParamBlendRel, kPhParamLpInFeedbackRel, kPhParamStereoFadeRel, kPhParamWetnessRel, kNumOfPhParameters }

Public Member Functions

- virtual zfxError_t Process (float **ppfInputBuffer, float **ppfOutputBuffer, int iNumberOfFrames)=0
- virtual zfxError_t SetLfoType (zfxLfoType_t eLfoType)=0
- virtual zfxLfoType_t GetLfoType ()=0
- virtual zfxError_t SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t ePhase)=0
- virtual zfxPhaseOffsetBetweenChannels_t GetLfoPhaseBetweenChannels ()=0
- virtual zfxError_t SetParam (PhParameter_t ePhParameterIdx, float fParamValue)=0
- virtual float GetParam (PhParameter_t ePhParameterIdx)=0
- virtual zfxError_t SetNumOfStages (int iNumOfStages=5)=0
- virtual int GetNumOfStages ()=0
- virtual zfxError_t SetAddDenormalNoise (bool bAddNoise=true)=0
- virtual bool GetAddDenormalNoise ()=0
- virtual zfxError_t SetBypass (bool bBypass=false)=0
- virtual bool GetBypass ()=0
- virtual zfxError_t Reset ()=0

Static Public Member Functions

- static zfxError_t CreateInstance (CPhaserIf *&pCPhaser, float fSampleRate, int iNumberOfChannels)
- static zfxError_t DestroyInstance (CPhaserIf *&pCPhaser)

4.6.1 Detailed Description

Definition at line 40 of file PhaserIf.h.

4.6.2 Member Enumeration Documentation

4.6.2.1 enum CPhaserIf::PhParameter_t

available Ph parameters

Enumerator:

kPhParamLfoDepthRel modulation range (0...1)

kPhParamLfoFreqInHz modulation frequency (0...20)

kPhParamFeedbackRel amount of feedback (0...0.9999)

kPhParamFeedForwardRel amount of feedforward (modulation output) (0...1)

kPhParamBlendRel amount of blend (modulation input) (0...1)

kPhParamLpInFeedbackRel low pass amount in feedback path (0...0.9999)

kPhParamStereoFadeRel crosschannel amount of processed samples (0...1)

kPhParamWetnessRel amount of effect between (0...1)

kNumOfPhParameters

Definition at line 45 of file PhaserIf.h.

```
{
    kPhParamLfoDepthRel,
    kPhParamLfoFreqInHz,
    kPhParamFeedbackRel,
    kPhParamFeedForwardRel,
    kPhParamBlendRel,
    kPhParamLpInFeedbackRel,
    kPhParamStereoFadeRel,
    kPhParamWetnessRel,

    kNumOfPhParameters
};
```

4.6.3 Member Function Documentation

4.6.3.1 static zfxError_t CPhaserIf::CreateInstance (CPhaserIf *& *pCPhaser*, float *fSampleRate*, int *iNumberOfChannels*) [static]

create new instance

Parameters

pCPhaser	: handle to new instance
fSampleRate	: sample rate of input/output audio data
iNumberOf-Channels	: number of channels of input/output audio data

Returns

zfxError_t: kNoError if no error

4.6.3.2 static zfxError_t CPhaserIf::DestroyInstance (CPhaserIf *& *pCPhaser*) [static]

destroy an instance

Parameters

pCPhaser	: handle to instance to be destroyed

Returns

zfxError_t: kNoError if no error

4.6.3.3 virtual bool CPhaserIf::GetAddDenormalNoise ( ) [pure virtual]

return if noise addition to avoid denormals is avtivated or not

Returns

bool : true if activated

4.6.3.4 virtual bool CPhaserIf::GetBypass ( ) [pure virtual]

return bypass state

Returns

zfxError_t: kNoError if no error

4.6.3.5 virtual zfxPhaseOffsetBetweenChannels_t CPhaserIf::GetLfoPhaseBetweenChannels ( ) [pure virtual]

return current phase difference between channels

Returns

zfxPhaseOffsetBetweenChannels_t : phase between succeeding channels

4.6.3.6 virtual zfxLfoType_t CPhaserIf::GetLfoType ( ) [pure virtual]

return current lfo waveform

Returns

zfxLfoType_t : lfo type

4.6.3.7 virtual int CPhaserIf::GetNumOfStages ( ) [pure virtual]

return current number of stages

Returns

int : number of stages

4.6.3.8 virtual float CPhaserIf::GetParam (PhParameter_t *ePhParameterIdx*) [pure virtual]

return current parameter

Parameters

ePhParameterIdx	: parameter index as defined in CPhaserIf::PhParameter_t

Returns

float : corresponding parameter value

4.6.3.9 virtual zfxError_t CPhaserIf::Process (float ** *ppfInputBuffer,* float ** *ppfOutputBuffer,* int *iNumberOfFrames*) [pure virtual]

processing function

Parameters

ppfInputBuffer	: input audio buffer of dimension [channels][frames]
ppfOutputBuffer	: output audio buffer of dimension [channels][frames], may be identical with ppfInputBuffer for inplace processing
iNumberOfFrames	: number of frames (=samples per channel)

Returns

zfxError_t: kNoError if no error

4.6.3.10 virtual zfxError_t CPhaserIf::Reset ( ) [pure virtual]

reset internal buffers

Returns

zfxError_t: kNoError if no error

4.6.3.11 virtual zfxError_t CPhaserIf::SetAddDenormalNoise (bool *bAddNoise* = true) [pure virtual]

set optional noise addition to avoid denormals

Parameters

bAddNoise	: true if noise should be added

Returns

EqError_t : kNoError if no error

4.6.3.12 virtual zfxError_t CPhaserIf::SetBypass (bool *bBypass* = `false`) [pure virtual]

set effect to bypass mode

Returns

zfxError_t: kNoError if no error

4.6.3.13 virtual zfxError_t CPhaserIf::SetLfoPhaseBetweenChannels (zfxPhaseOffsetBetweenChannels_t *ePhase*) [pure virtual]

set phase difference between succeeding channels

Parameters

ePhase	: filter type as defined in zfxPhaseOffsetBetweenChannels_t

Returns

zfxError_t: kNoError if no error

4.6.3.14 virtual zfxError_t CPhaserIf::SetLfoType (zfxLfoType_t *eLfoType*) [pure virtual]

set lfo waveform

Parameters

eLfoType	: filter type as defined in zfxLfoType_t

Returns

zfxError_t: kNoError if no error

4.6.3.15 virtual zfxError_t CPhaserIf::SetNumOfStages (int *iNumOfStages* = `5`) [pure virtual]

set number of phaser stages (between 1...10)

Parameters

iNumOf-Stages	: number of stages

Returns

zfxError_t: kNoError if no error

4.6.3.16 virtual zfxError_t CPhaserIf::SetParam (PhParameter_t *ePhParameterIdx*, float *fParamValue*) [pure virtual]

set parameter

Parameters

ePhParameterIdx	: parameter index as defined in CPhaserIf::PhParameter_t
fParamValue	: parameter value

Returns

zfxError_t: kNoError if no error

The documentation for this class was generated from the following file:

- PhaserIf.h

5 File Documentation

5.1 BitCrushIf.h File Reference

interface of the BitCrushIf class.

`#include "fxpack.h"` Include dependency graph for BitCrushIf.h:

Classes

- class CBitCrushIf

5.1.1 Detailed Description

interface of the BitCrushIf class. :

Definition in file BitCrushIf.h.

5.2 ChorusFlangerIf.h File Reference

interface of the CChorusFlangerIf class.

`#include "fxpack.h"` Include dependency graph for ChorusFlangerIf.h:

Classes

- class CChorusFlangerIf

5.2.1 Detailed Description

interface of the CChorusFlangerIf class. :

Definition in file ChorusFlangerIf.h.

5.3 DelayIf.h File Reference

interface of the CDelayIf class.

`#include "fxpack.h"` Include dependency graph for DelayIf.h:

Classes

- class CDelayIf

5.3.1 Detailed Description

interface of the CDelayIf class. :

Definition in file DelayIf.h.

5.4 docugen.txt File Reference

5.4.1 Detailed Description

source documentation main file

Definition in file docugen.txt.

5.5 DynamicsIf.h File Reference

interface of the CDynamicsIf class.

`#include "fxpack.h"` Include dependency graph for DynamicsIf.h:

Classes

- class CDynamicsIf

5.5.1 Detailed Description

interface of the CDynamicsIf class. :

Definition in file DynamicsIf.h.

5.6 fxpack.h File Reference

global enums and functions for fxpack

This graph shows which files directly or indirectly include this file:

Enumerations

- enum zfxLfoType_t { kSine, kSaw, kTriang, kNoLfo, kNumOfLfoTypes }
- enum zfxPhaseOffsetBetweenChannels_t { k0Degree, k90Degree, k180Degree, k270Degree, kNumOfPhaseOffsets }
- enum zfxError_t { kNoError, kMemError, kInvalidFunctionParamError, kUnknownError, kNumOfErrors }
- enum zfxVersion_t { kzfxVersionMajor, kzfxVersionMinor, kzfxVersionPatch, kzfxVersionBuild, kNumzfxVersionInts }

Functions

- const int zfxGetVersion (const zfxVersion_t eVersionIdx)
- const char * zfxGetBuildDate ()

5.6.1 Detailed Description

global enums and functions for fxpack :

Definition in file fxpack.h.

5.6.2 Enumeration Type Documentation

5.6.2.1 enum zfxError_t

error return values

Enumerator:

kNoError no error occurred

kMemError memory allocation failed

kInvalidFunctionParamError one or more function parameters are not valid

kUnknownError unknown error occurred

kNumOfErrors

Definition at line 61 of file fxpack.h.

```
{
    kNoError,
    kMemError,
    kInvalidFunctionParamError,
    kUnknownError,

    kNumOfErrors
};
```

5.6.2.2 enum zfxLfoType_t

lfo specification for modulated effects

Enumerator:

kSine sinusoidal oscillator

kSaw saw oscillator

kTriang triangular oscillator

kNoLfo no oscillator

kNumOfLfoTypes

Definition at line 39 of file fxpack.h.

```
{
    kSine,
    kSaw,
    kTriang,
    kNoLfo,

    kNumOfLfoTypes
};
```

5.6.2.3 enum zfxPhaseOffsetBetweenChannels_t

phase offset between channels for modulated effects

Enumerator:

k0Degree no phase difference between channels

k90Degree phase difference = pi/2

k180Degree phase difference = pi

k270Degree phase difference = 3pi/2

kNumOfPhaseOffsets

Definition at line 50 of file fxpack.h.

```
{
    k0Degree,
```

```
    k90Degree,
    k180Degree,
    k270Degree,

    kNumOfPhaseOffsets
};
```

5.6.2.4 enum zfxVersion_t

Enumerator:

kzfxVersionMajor

kzfxVersionMinor

kzfxVersionPatch

kzfxVersionBuild

kNumzfxVersionInts

Definition at line 71 of file fxpack.h.

```
{
    kzfxVersionMajor,
    kzfxVersionMinor,
    kzfxVersionPatch,
    kzfxVersionBuild,

    kNumzfxVersionInts
};
```

5.6.3 Function Documentation

5.6.3.1 const char* zfxGetBuildDate ( )

5.6.3.2 const int zfxGetVersion (const zfxVersion_t *eVersionIdx*)

5.7 ParamEQIf.h File Reference

interface of the CParametricEqIf class.

`#include "fxpack.h"` Include dependency graph for ParamEQIf.h:

Classes

- class CParametricEqIf

5.7.1 Detailed Description

interface of the CParametricEqIf class. :

Definition in file ParamEQIf.h.

5.8 PhaserIf.h File Reference

interface of the CPhaserIf class.

`#include "fxpack.h"` Include dependency graph for PhaserIf.h:

Classes

- class CPhaserIf

5.8.1 Detailed Description

interface of the CPhaserIf class. :

Definition in file PhaserIf.h.