HyperStack

1.0

Generated by Doxygen 1.6.3

Mon Nov 22 09:26:19 2010

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 HyperADSR Struct Reference

Linear ADSR Envelope generator class.

`#include <HyperADSR.h>`

Public Member Functions

- HyperADSR ()

  Constructor.

- HyperADSR (const HyperADSR &v)

  Copy constructor.

- ~HyperADSR ()

  Destructor.

- void operator= (const HyperADSR &v)

  Copy operator.

- void update ()

  Updates actual ADSR values. Must be called for every update.

- void setA (double a)
- void setD (double d)
- void setS (double s)
- void setR (double r)
- void reset ()

  Re-starts this envelope, makes the current value 0.

- bool done ()

Data Fields

- double **A**
- double **D**
- double **S**
- double **R**

3.1.1 Detailed Description

Linear ADSR Envelope generator class.

3.1.2 Member Function Documentation

3.1.2.1 bool done ()

Checks if this envelope is completely done, i.e. release fade time is over.

Returns

false if envelope is not finished yet, true if envelope is finished.

3.1.2.2 void setA (double *a*)

Sets the attack time.

Parameters

a The attack time factor, from 0.0 to 1.0;

Returns

nothing

3.1.2.3 void setD (double *d*)

Sets the decay time.

Parameters

d The decay time factor, from 0.0 to 1.0;

Returns

nothing

3.1.2.4 void setR (double *r*)

Sets the release time.

Parameters

r The release level factor, from 0.0 to 1.0;

Returns

nothing

3.1.2.5 void setS (double *s*)

Sets the sustain level.

Parameters

s The sustain level factor, from 0.0 to 1.0;

Returns

nothing

The documentation for this struct was generated from the following files:

- HyperADSR.h
- HyperADSR.cpp

3.2 HyperFilter Class Reference

Low-pass filter class.

`#include <HyperFilter.h>`

Public Member Functions

- HyperFilter ()
  Constructor.
- ~HyperFilter ()
  Destructor.
- void setSampleRate (float sampleRate)
- void setCutoff (float c)
- void setCutoffFactor (float f)
- void **setResonance** (float r)
- void set (float c, float r)
- float getCutoff ()
- float filter (float input)

3.2.1 Detailed Description

Low-pass filter class.

3.2.2 Member Function Documentation

3.2.2.1 float filter (float *input*)

Filters a single sound signal. NOT stateless, keeps history.

Parameters

input The next value in the signal.

Warning

This function is NOT stateless, and can only process one signal.

Returns

The filtered next value of the signal.

3.2.2.2 float getCutoff ()

Gets the current cutoff frequency

Returns

Current cutoff frequency in Hz

3.2.2.3 void set (float *c*, float *r*)

Sets both the cutoff and the resonance.

Parameters

c The cutoff frequency in Hz

r The resonance factor

Returns

nothing

3.2.2.4 void setCutoff (float *c*)

Sets the cutoff frequency.

Parameters

c The cutoff frequency in Hz

Returns

nothing

3.2.2.5 void setCutoffFactor (float *f*)

Sets the cutoff frequency by a 0.0 - 1.0 factor. Automatically sets the cutoff freq in Hz via an exponential function: cutoff = ((pow(64.0f, x) - 1.0f) / 63.0f) ∗ nyquistF;

Parameters

f The cutoff frequency factor 0.0 - 1.0

Returns

nothing

3.2.2.6 void setSampleRate (float *sampleRate*)

Sets the sample rate. Called automatically by the constructor on the default sample rate.

Parameters

sampleRate The sample rate.

Warning

Most be called at least once before using!

Returns

nothing

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

- HyperFilter.h
- HyperFilter.cpp

3.3 HyperGUI Class Reference

GUI class, implements the VST GUI editor interface.

`#include <HyperGUI.h>`

Public Member Functions

- HyperGUI (AudioEffect *ae)

  Constructor.

- virtual ~HyperGUI ()

  Destructor.

- virtual long open (void *ptr)
- virtual void close ()

  Closes the UI editor interface.

- virtual void setParameter (long index, float value)
- void updateKnob (long index, float value)
- virtual void valueChanged (CDrawContext *context, CControl *control)

Data Fields

- CBitmap * **hBackground**
- bool **bOpened**
- HyperKnob **oscKnobs** [GUI_NUM_KNOBS]
- AudioEffect * **oscEffect**

3.3.1 Detailed Description

GUI class, implements the VST GUI editor interface.

3.3.2 Member Function Documentation

3.3.2.1 long open (void * *ptr*) `[virtual]`

Creates and initialises the UI editor interface.

Parameters

ptr Data pointer. Passed on and used internally by the VST architecture.

Returns

true if success, false otherwise

3.3.2.2 void setParameter (long *index*, float *value*) [virtual]

Updates an indexed parameter. Also updates value of the parent effect class.

Parameters

index The parameter index.

value The parameter value.

Returns

nothing

3.3.2.3 void updateKnob (long *index*, float *value*)

Updates an indexed parameter. Does NOT update value of the parent effect class.

Parameters

index The parameter index.

value The parameter value.

Returns

nothing

3.3.2.4 void valueChanged (CDrawContext * *context*, CControl * *control*) [virtual]

Callback function called when the user or host system changes a parameter.

Parameters

context The rendering context.

control Handle of the updated control.

Returns

nothing

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

- HyperGUI.h
- HyperGUI.cpp

3.4 HyperKnob Struct Reference

Structure to store and manage an interface knob.

`#include <HyperGUI.h>`

Public Member Functions

- HyperKnob ()

  Constructor.

- void set (int xc, int yc, int xl, int yl, const char *lab)
- void update (AudioEffect *oscEffect, long index, float value)

Data Fields

- int **x**
- int **y**
- int **xLabel**
- int **yLabel**
- char **label** [128]
- CAnimKnob * **handle**
- CLabel * **handleLabel**

3.4.1 Detailed Description

Structure to store and manage an interface knob.

3.4.2 Member Function Documentation

3.4.2.1 void set (int *xc*, int *yc*, int *xl*, int *yl*, const char * *lab*)

Sets all the locational parameters of this UI element at once.

Parameters

xc The X value (in pixels) of the center position of the knob.

yc The Y value (in pixels) of the center position of the knob.

xl The Y value (in pixels) of the center position of the label. (0 means no label)

yl The Y value (in pixels) of the center position of the label. (0 means no label)

lab Initial label caption.

Returns

nothing

3.4.2.2 void update (AudioEffect * *oscEffect*, long *index*, float *value*)

Updates the parameter value of this knob, and also updates the corresponding text in the label, if there is a label.

Parameters

oscEffect Handle to the parent effect class

index The parameter index.

value The parameter value.

Returns

nothing

The documentation for this struct was generated from the following files:

- HyperGUI.h
- HyperGUI.cpp

3.5 HyperLFO Class Reference

LFO generator class.

`#include <HyperLFO.h>`

Public Member Functions

- HyperLFO ()

  Destructor.

- HyperLFO (const HyperLFO &l)

  Copy constructor.

- ~HyperLFO ()

  Constructor.

- void setSampleRate (float sampleRate)
- void tick (float frequency, float frequency2, float *output, int size)
- float tick (float frequency, float frequency2)
- void reset ()
- void operator= (const HyperLFO &l)

  Copy operator.

Data Fields

- float LFOVolume

  Volume of the LFO.

- float LFOVolumeNoise

  Volume of the Perlin noise.

3.5.1 Detailed Description

LFO generator class.

3.5.2 Member Function Documentation

3.5.2.1 void reset ()

Resets the LFO phase. Also picks a new random function seed for the Perlin noise generator.

3.5.2.2 void setSampleRate (float *sampleRate*)

Sets the sample rate for the LFO

Parameters

sampleRate The new sample rate

Returns

nothing

3.5.2.3 float tick (float *frequency*, float *frequency2*)

Calculates the next value of the LFO signal.

Parameters

frequency The LFO frequency in Hz

frequency2 The Perlin noise frequency

Returns

The next value of the LFO signal.

3.5.2.4 void tick (float *frequency*, float *frequency2*, float * *output*, int *size*)

Fills out a buffer with the LFO signal.

Parameters

frequency The LFO frequency in Hz

frequency2 The Perlin noise frequency

output The destination buffer

size The size of the destination buffer to fill

Returns

nothing

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

- HyperLFO.h
- HyperLFO.cpp

3.6 HyperOsc Class Reference

Oscillator class.

`#include <HyperOSC.h>`

Public Member Functions

- void setSampleRate (float sampleRate)
- void setQuality (int quality)
- void setShape (int shape)
- HyperOsc ()

  Constructor.

- HyperOsc (const HyperOsc &h)

  Copy constructor.

- ~HyperOsc ()

  Destructor.

- void TRIVtick (float frequency, float *output, int size)
- void BLITtick (float frequency, float *output, int size)
- void tick (float frequency, float *output, int size)
- float tick (float frequency)
- void reset ()

  Resets the oscillator phase (unused).

- void operator= (const HyperOsc &h)

  Copy operator.

Data Fields

- char **oscShape**
- char **oscQuality**
- float **oscSampleRate**
- float **oscVolume**

3.6.1 Detailed Description

Oscillator class.

3.6.2 Member Function Documentation

3.6.2.1 void BLITtick (float *frequency*, float * *output*, int *size*)

Fills the buffer with the next oscillator signal values, using the band-limited alias-free STK BLIT algorithm.

Parameters

frequency The oscillator frequency in Hz

output The destination buffer

size The size of the destination buffer

Returns

nothing

3.6.2.2 void setQuality (int *quality*)

Sets the the quality of the oscillator to HIGH or LOW. HIGH uses the Stk alias-free baldn-limited oscillators, while LOW uses a fast trivial wavetable algorithm.

Parameters

quality The quality setting of this oscillator

Returns

nothing

3.6.2.3 void setSampleRate (float *sampleRate*)

Sets the sample rate. Called automatically by the constructor on the default sample rate.

Parameters

sampleRate The sample rate.

Warning

Most be called at least once before using!

Returns

nothing

3.6.2.4 void setShape (int *shape*)

Sets the the waveshape of this oscillator to SAW or SQUARE.

Parameters

shape The wave shape setting of this oscillator

Returns

nothing

3.6.2.5 float tick (float *frequency*)

Calculates the next oscillator signal value.

Parameters

frequency The oscillator frequency in Hz

Returns

The next oscillator signal value.

3.6.2.6 void tick (float *frequency*, float * *output*, int *size*)

Fills the buffer with the next oscillator signal values.

Parameters

frequency The oscillator frequency in Hz

output The destination buffer

size The size of the destination buffer

Returns

nothing

3.6.2.7 void TRIVtick (float *frequency*, float * *output*, int *size*)

Fills the buffer with the next oscillator signal values, using the fast trivial wavetable algorithm.

Parameters

frequency The oscillator frequency in Hz

output The destination buffer

size The size of the destination buffer

Returns

nothing

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

- HyperOSC.h
- HyperOSC.cpp

3.7 HyperOscParam Struct Reference

Oscillator single parameter structure.

`#include <HyperOSC.h>`

Public Member Functions

- HyperOscParam ()
  Constructor.
- HyperOscParam (const HyperOscParam &p)
  Copy constructor.
- ~HyperOscParam ()
  Destructor.
- void reset ()
- operator float ()
- void operator= (const HyperOscParam &p)
  Copy operator.

Data Fields

- float **mean**
- float **var**
- float **value**

3.7.1 Detailed Description

Oscillator single parameter structure.

3.7.2 Member Function Documentation

3.7.2.1 operator float ()

Array operator

Returns

The current parameter value.

3.7.2.2 void reset ()

Resets the oscilator parameter, assigns a new value depending on the deviation parameter.

The documentation for this struct was generated from the following files:

- HyperOSC.h
- HyperOSC.cpp

3.8 HyperOscParams Struct Reference

Oscillator parameter manager structure.

`#include <HyperOSC.h>`

Public Member Functions

- HyperOscParams ()

  Constructor.

- HyperOscParams (const HyperOscParams &p)

  Copy constructor.

- ~HyperOscParams ()

  Destructor.

- void reset ()
- float operator[ ] (int index)
- void operator= (const HyperOscParams &p)

  Copy operator.

Data Fields

- HyperOscParam **param** [NUM_OSC_PARAMS]

3.8.1 Detailed Description

Oscillator parameter manager structure.

3.8.2 Member Function Documentation

3.8.2.1 float operator[ ] (int *index*)

Array operator

Parameters

index The parameter index

Returns

The nth parameter in the array

3.8.2.2 void reset ()

Resets this param array, by resetting every parameter in the array.\ and giving them new values, based on their deviation.

The documentation for this struct was generated from the following files:

- HyperOSC.h
- HyperOSC.cpp

3.9 HyperOscStack Struct Reference

Oscillator stack/array manager structure.

`#include <HyperOSC.h>`

Public Member Functions

- HyperOscStack ()

  Constructor.

- ~HyperOscStack ()

  Destructor.

- int getNumOSC ()
- void setNumOSC (int num)
- void updateBlueprint (int type)
- HyperOsc * bp ()

  Returns the oscillator blueprint for this array to be based upon.

- HyperOscParams * bpp ()

  Returns the parameter blueprint for this array to be based upon.

- HyperLFO * bpLFO ()

  Returns the LFO blueprint for this array to be based upon.

- HyperADSR * bpADSR ()

  Returns the ADSR envelope blueprint for this array to be based upon.

- HyperOsc * operator[ ] (int index)
- HyperOscParams * p (int index)
- HyperLFO * LFO (int index)
- HyperADSR * ADSR (int index)
- void reset ()
- void setSampleRate (float sampleRate)
- void setQuality (int quality)
- void setShape (int shape)
- void keyOff ()

  MIDI Key off event - sets all ADSR envelopes to the RELEASE state.

Data Fields

- HyperOsc **oscStack** [OSC_MAX]
- HyperOscParams **oscParams** [OSC_MAX]
- HyperLFO **oscLFO** [OSC_MAX]
- HyperADSR **oscADSR** [OSC_MAX]
- int **oscStackSize**
- HyperOsc **oscBlueprint**
- HyperOscParams **oscParamsBlueprint**

- HyperLFO **oscLFOBlueprint**
- HyperADSR **oscADSRBlueprint**
- HyperFilter **oscFilter**
- HyperADSR **oscFilterADSR**
- HyperADSR **oscLFOADSR**

3.9.1 Detailed Description

Oscillator stack/array manager structure.

3.9.2 Member Function Documentation

3.9.2.1 HyperADSR * ADSR (int *index*)

Array operator function for ADSR envelopes

Parameters

index The envelope index

Returns

the i-th envelope

3.9.2.2 int getNumOSC ()

Gives the current number of oscillators in this array

Returns

The current number of oscillators in this array

3.9.2.3 HyperLFO * LFO (int *index*)

Array operator function for LFOs

Parameters

index The LFO index

Returns

the i-th LFO

3.9.2.4 HyperOsc * operator[ ] (int *index*)

Array operator

Parameters

index The oscillator index

Returns

the i-th oscillator

3.9.2.5 HyperOscParams * p (int *index*)

Array operator function for parameters

Parameters

index The parameter manager index

Returns

the i-th parameter manager

3.9.2.6 void reset ()

Resets everything, phase-offsets all oscillators randomly, get ready for a new note to be played.

3.9.2.7 void setNumOSC (int *num*)

Sets the current number of oscillators in this array

Parameters

num The new number of oscillators this stack will have

Returns

nothing

3.9.2.8 void setQuality (int *quality*)

Sets the quality setting for every oscillator in the array.

Parameters

quality The new oscillator quality setting

Returns

nothing

3.9.2.9 void setSampleRate (float *sampleRate*)

Sets the sample rate for everything.

Parameters

sampleRate The new sample rate

Returns

nothing

3.9.2.10 void setShape (int *shape*)

Sets the wave shape setting for every oscillator in the array.

Parameters

shape The new oscillator wave shape setting

Returns

nothing

3.9.2.11 void updateBlueprint (int *type*)

Copies from the blueprints to each module in the arrays.

Parameters

type -1 = update everything, 0 only update oscillator stack, 1 only update parameter array, 2 only update LFO array, 3 only update ADSR envelope array.

Returns

nothing

The documentation for this struct was generated from the following files:

- HyperOSC.h
- HyperOSC.cpp

3.10 HyperPoly Struct Reference

Polyphony manager - array of voices and oscillator stacks.

`#include <HyperPoly.h>`

Public Member Functions

- int nextVoice (long note)
- int getMaxVoices ()
- HyperOscStack * operator[ ] (int index)

Data Fields

- HyperVoice **voice** [MAX_VOICES]
- HyperOscStack **stack** [MAX_VOICES]

3.10.1 Detailed Description

Polyphony manager - array of voices and oscillator stacks.

3.10.2 Member Function Documentation

3.10.2.1 int getMaxVoices ()

Gives the maximum supported number of voices

Returns

The maximum supported number of voices

3.10.2.2 int nextVoice (long *note*)

Finds and gives the best slot for the next note to go into.

Parameters

note The MIDI note index.

Returns

The index of best slot for the next note to go into.

3.10.2.3 HyperOscStack * operator[ ] (int *index*)

Array operator

Parameters

index The voice index.

Returns

The corresponding i-th voice oscillator stack.

The documentation for this struct was generated from the following files:

- HyperPoly.h
- HyperPoly.cpp

3.11 HyperProgram Struct Reference

Structure to store and manage a preset's parameter array.

`#include <HyperStack.h>`

Public Member Functions

- void set (int index, float value)
- HyperProgram ()

  Constructor.

- ~HyperProgram ()

  Destructor.

Data Fields

- char **name** [PROGRAM_NAME_LENGTH]
- float **fVolume**
- float **fNumOSC**
- float **fQuality**
- float **fWaveShape**
- float **fSemi**
- float **fFine**
- float **fAttack**
- float **fSustain**
- float **fDecay**
- float **fRelease**
- float **fAttackDev**
- float **fSustainDev**
- float **fDecayDev**
- float **fReleaseDev**
- float **fDetuneDev**
- float **fCutoff**
- float **fResonance**
- float **fPLFOGain**
- float **fPLFOFrequency**
- float **fPLFOFreqDev**
- float **fPLFOAmount**
- float **fPLFONoise**
- float **fPLFONoiseFreq**
- float **fPLFOAttack**
- float **fPLFOSustain**
- float **fPLFODecay**
- float **fPLFORelease**
- float **fFilterAttack**
- float **fFilterSustain**
- float **fFilterDecay**
- float **fFilterRelease**
- float **fFilterAmount**

3.11.1 Detailed Description

Structure to store and manage a preset's parameter array.

3.11.2 Member Function Documentation

3.11.2.1 void set (int *index*, float *value*)

Sets parameter value given its index

Parameters

index The parameter index

value The parameter value

Returns

nothing

The documentation for this struct was generated from the following files:

- HyperStack.h
- HyperParams.cpp
- HyperStack.cpp

3.12 HyperStack Class Reference

Main hyperSTACK VST effect class.

`#include <HyperStack.h>`

Public Member Functions

- HyperStack (audioMasterCallback audioMaster)

  Constructor - Initialises presets, makes new editor, tells host parameters...etc.

- ~HyperStack ()

  Destructor.

- virtual void process (float **inputs, float **outputs, long sampleFrames)
- virtual void processReplacing (float **inputs, float **outputs, long sampleFrames)
- virtual long processEvents (VstEvents *events)
- virtual void setProgram (long index)
- virtual long getProgram ()
- virtual void setProgramName (char *name)
- virtual void getProgramName (char *name)
- virtual bool getProgramNameIndexed (long category, long index, char *text)
- virtual bool copyProgram (long dest)
- virtual void setParameter (long index, float value)
- virtual float getParameter (long index)
- virtual void getParameterLabel (long index, char *label)
- virtual void getParameterDisplay (long index, char *text)
- virtual void getParameterName (long index, char *text)
- virtual void setSampleRate (float sampleRate)
- virtual void setBlockSize (long blockSize)
- virtual long canDo (char *text)
- virtual void resume ()

  Tell the VST host that we are ready to recieve MIDI events.

- virtual bool getOutputProperties (long index, VstPinProperties *properties)
- virtual bool getEffectName (char *name)
- virtual bool getVendorString (char *text)
- virtual bool getProductString (char *text)
- virtual long getVendorVersion ()
- virtual long getMidiProgramName (long channel, MidiProgramName *midiProgramName)
- virtual long getCurrentMidiProgram (long channel, MidiProgramName *currentProgram)
- virtual long getMidiProgramCategory (long channel, MidiProgramCategory *category)
- virtual bool hasMidiProgramsChanged (long channel)

  Notifies host if a MIDI program has changed.

- virtual bool getMidiKeyName (long channel, MidiKeyName *keyName)

3.12.1 Detailed Description

Main hyperSTACK VST effect class.

3.12.2 Member Function Documentation

3.12.2.1 long canDo (char * *text*) [virtual]

Tells host if the given VST feature is supported.

Parameters

text The VST feature.

Returns

1 if this feature is supported, -1 otherwise.

3.12.2.2 bool copyProgram (long *dest*) [virtual]

Copies the parameters of current activated preset into a given destination preset.

Parameters

dest The preset index of the destination.

Returns

true if successful, false otherwise

3.12.2.3 long getCurrentMidiProgram (long *channel*, MidiProgramName * *currentProgram*) [virtual]

Tells host the MIDI program for given MIDI channel.

Parameters

channel The MIDI channel program index

currentProgram The destination program structure to fill out

Returns

the MIDI channel index

3.12.2.4 bool getEffectName (char * *name*) [virtual]

Tells host name of this VST plugin

Parameters

name The destination char buffer to write the VST name into.

Returns

true if successful, false otherwise

3.12.2.5 bool getMidiKeyName (long *channel*, MidiKeyName * *keyName*) [virtual]

Tells host MIDI hey name for given channel

Parameters

channel The MIDI channel index

keyName The destination keyname structure to fill out

Returns

true sucessful, false otherwise.

3.12.2.6 long getMidiProgramCategory (long *channel*, MidiProgramCategory * *category*) [virtual]

Tells host the category of a midi program.

Parameters

channel The MIDI channel index

category The destination category structure to fill out

Returns

The number of categories.

3.12.2.7 long getMidiProgramName (long *channel*, MidiProgramName * *midiProgramName*) [virtual]

Tells host the MIDI program name for given MIDI channel.

Parameters

channel The MIDI channel index

midiProgramName The destination program name structure to fill out

Returns

0 if successful, 1 if this channel is the drum channel, 128 if error.

3.12.2.8 bool getOutputProperties (long *index*, VstPinProperties * *properties*) [virtual]

Tells host of the names of the audio output channels.

Parameters

index The audio output index

properties The destination properties array to write into

Returns

true if successful, false otherwise

3.12.2.9 float getParameter (long *index*) [virtual]

Gets VST parameter value given its index

Parameters

index The VST parameter index

Returns

The corresponding VST parameter value

3.12.2.10 void getParameterDisplay (long *index*, char * *text*) [virtual]

Gets VST parameter value display text given its index

Parameters

index The VST parameter index

text The destination buffer for the corresponding parameter display text

Returns

nothing

3.12.2.11 void getParameterLabel (long *index*, char * *label*) [virtual]

Gets VST parameter value label given its index

Parameters

index The VST parameter index

label The destination buffer for the corresponding VST parameter label

Returns

nothing

3.12.2.12 void getParameterName (long *index*, char * *label*) [virtual]

Gets VST parameter name display text given its index

Parameters

index The VST parameter index

label The destination buffer for the corresponding parameter display name

Returns

nothing

3.12.2.13 bool getProductString (char * *text*) [virtual]

Tells product name of this VST plugin

Parameters

text The destination char buffer to write the product name into.

Returns

true if successful, false otherwise

3.12.2.14 long getProgram () [virtual]

Gives the current preset.

Returns

The current activated preset.

3.12.2.15 void getProgramName (char * *name*) [virtual]

Gives the name of the current preset.

Parameters

name The destination char buffer for the name of the current preset

Returns

nothing

3.12.2.16 bool getProgramNameIndexed (long *category*, long *index*, char * *text*) [virtual]

Gives the name of the preset at the given index.

Parameters

category The preset category

index The preset index

text The destination char buffer for the name of the indexed preset

Returns

true if successful, false otherwise

3.12.2.17 bool getVendorString (char * *text*) [virtual]

Tells vendor name of this VST plugin

Parameters

text The destination char buffer to write the vendor name into.

Returns

true if successful, false otherwise

3.12.2.18 long getVendorVersion () [virtual]

Tells product version of this VST plugin

Returns

The product version of this VST plugin

3.12.2.19 bool hasMidiProgramsChanged (long *channel*) [virtual]

Notifies host if a MIDI program has changed.

Notifies host if a MIDI program has changed

Returns

true if program has changed, false otherwise.

3.12.2.20 void process (float ** *inputs*, float ** *outputs*, long *sampleFrames*) [virtual]

The main VST audio output function. Does most of the wiring, processing, filtering and calculations.

Parameters

inputs The next audio input signal buffer (unused)

outputs The destination buffer for following audio output signal

sampleFrames The size of the destination buffer

Returns

nothing

3.12.2.21 long processEvents (VstEvents * *events*) [virtual]

The main VST MIDI event parsing function.

Parameters

events The array of VST events.

Returns

0 if error, 1 if successful.

3.12.2.22 void processReplacing (float ** *inputs*, float ** *outputs*, long *sampleFrames*) [virtual]

The main VST audio output function. Does most of the wiring, processing, filtering and calculations. (A destructive, overwriting variant of process().)

Parameters

inputs The next audio input signal buffer (unused)

outputs The destination buffer for following audio output signal

sampleFrames The size of the destination buffer

Returns

nothing

3.12.2.23 void setBlockSize (long *blockSize*) [virtual]

Sets the effect's block size.

Parameters

sampleRate The new block size.

Returns

nothing

3.12.2.24 void setParameter (long *index*, float *value*) [virtual]

Sets VST parameter value given its index, and notify the necessary modules about the parameter change.

Parameters

index The VST parameter index

value The VST parameter value

Returns

nothing

3.12.2.25 void setProgram (long *index*) [virtual]

Changes the current preset.

Parameters

index The index of the new preset

Returns

nothing

3.12.2.26 void setProgramName (char * *name*) [virtual]

Sets the name of the current preset.

Parameters

name The char buffer with the new name for the current preset

Returns

nothing

3.12.2.27 void setSampleRate (float *sampleRate*) [virtual]

Sets the effect's sample rate. This is the function the host notifies. Passes on the sample rate to each osc stack, which passes it down to each individual module.

Parameters

sampleRate The new sample rate.

Returns

nothing

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

- HyperStack.h
- HyperParams.cpp
- HyperProcess.cpp
- HyperStack.cpp

3.13 HyperTuner Class Reference

Static class providing functions for frequency calculation.

`#include <HyperTuner.h>`

Public Member Functions

- float getNoteFrequency (int note, float detune)
- float getNoteFrequency (float note)

3.13.1 Detailed Description

Static class providing functions for frequency calculation.

3.13.2 Member Function Documentation

3.13.2.1 float getNoteFrequency (float *note*)

Calculates MIDI frequency given note (can take in floating point half-notes)

Parameters

note The MIDI note value

Returns

The corresponding note frequency in Hz

3.13.2.2 float getNoteFrequency (int *note*, float *detune*)

Calculates MIDI frequency given integer note and detune

Parameters

note The MIDI integer note value

detune The note detune in cents

Returns

The corresponding note frequency in Hz

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

- HyperTuner.h
- HyperTuner.cpp

3.14 HyperVoice Struct Reference

Structure to store a note.

`#include <HyperPoly.h>`

Public Member Functions

- HyperVoice ()
  Constructor.
- bool isActive ()

Data Fields

- long **note**
- long **velocity**
- long **delta**
- bool **active**
- int **index**

3.14.1 Detailed Description

Structure to store a note.

3.14.2 Member Function Documentation

3.14.2.1 bool isActive ()

Gives whether this voice is currently producing sound or inactive.

Returns

true if this voice is currently active and producing sound, false otherwise

The documentation for this struct was generated from the following files:

- HyperPoly.h
- HyperPoly.cpp

Chapter 4

File Documentation

4.1 HyperADSR.cpp File Reference

ADSR generator.

4.1.1 Detailed Description

ADSR generator.

ADSR Envelope Generator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.2 HyperADSR.h File Reference

ADSR generator.

Data Structures

- struct HyperADSR

  Linear ADSR Envelope generator class.

4.2.1 Detailed Description

ADSR generator.

ADSR Envelope Generator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.3 HyperFilter.cpp File Reference

Butterworth low-pass filter.

4.3.1 Detailed Description

Butterworth low-pass filter.

Butterworth low-pass filter class

Author

Xi Chen hypernewbie@gmail.com
neotec

Date

21/11/2010

4.4 HyperFilter.h File Reference

Butterworth low-pass filter.

Data Structures

- class HyperFilter

  Low-pass filter class.

4.4.1 Detailed Description

Butterworth low-pass filter.

Butterworth low-pass filter class

Author

Xi Chen hypernewbie@gmail.com
neotec

Date

21/11/2010

4.5 HyperGUI.cpp File Reference

VST GUI class interface.

Functions

- void oscGUISetupUITable (HyperKnob *oscKnobs)

4.5.1 Detailed Description

VST GUI class interface.

The VST GUI class interface implementation

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.5.2 Function Documentation

4.5.2.1 void oscGUISetupUITable (HyperKnob * *oscKnobs*)

Fills out the hardcoded locations of the UI elements into a table.

Parameters

oscKnobs The destination table.

Returns

nothing

4.6 HyperGUI.h File Reference

VST GUI class interface.

Data Structures

- struct HyperKnob

  Structure to store and manage an interface knob.

- class HyperGUI

  GUI class, implements the VST GUI editor interface.

Enumerations

- enum enum_resource_ids { **kBackgroundID** = 128, **kKnobID** }

  Resource ID numbers for images.

4.6.1 Detailed Description

VST GUI class interface.

The VST GUI class interface implementation

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.7 HyperLFO.cpp File Reference

LFO Generator.

Functions

- float LFOFactorFrequency (float factor)

  Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

4.7.1 Detailed Description

LFO Generator.

Low Frequency Oscillator generator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.7.2 Function Documentation

4.7.2.1 float LFOFactorFrequency (float *factor*)

Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

Parameters

factor The factor 0.0 - 1.0.

Returns

The corresponding LFO frequency in Hz.

4.8 HyperLFO.h File Reference

LFO Generator.

Data Structures

- class HyperLFO

  LFO generator class.

Functions

- float LFOFactorFrequency (float factor)

  Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

4.8.1 Detailed Description

LFO Generator.

Low Frequency Oscillator generator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.8.2 Function Documentation

4.8.2.1 float LFOFactorFrequency (float *factor*)

Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

Converts a LFO 0.0 - 1.0 factor to a Hz frequency.

Parameters

factor The factor 0.0 - 1.0.

Returns

The corresponding LFO frequency in Hz.

4.9 HyperOSC.cpp File Reference

Oscillator class.

4.9.1 Detailed Description

Oscillator class.

Band-Limited Oscillator class

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.10 HyperOSC.h File Reference

Oscillator class.

Data Structures

- class HyperOsc

  Oscillator class.

- struct HyperOscParam

  Oscillator single parameter structure.

- struct HyperOscParams

  Oscillator parameter manager structure.

- struct HyperOscStack

  Oscillator stack/array manager structure.

Enumerations

- enum osc_params_enumeration {

  OSC_PARAM_ATTACK_DEV, **OSC_PARAM_SUSTAIN_DEV**, **OSC_PARAM_DECAY_-DEV**, **OSC_PARAM_RELEASE_DEV**,

  OSC_PARAM_PLFO_FREQ, **NUM_OSC_PARAMS** }

  Individual oscillator params.

4.10.1 Detailed Description

Oscillator class.

Band-Limited Oscillator class

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.11 HyperParams.cpp File Reference

VST Parameter Functions.

4.11.1 Detailed Description

VST Parameter Functions.

VST Parameter Functions

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.12 HyperPoly.cpp File Reference

Polyphonic voice manager.

4.12.1 Detailed Description

Polyphonic voice manager.

Polyphonic voice manager

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.13 HyperPresets.cpp File Reference

VST Factory Presets Functions.

Functions

- void initStackPresets (HyperProgram *hyperPresets)

4.13.1 Detailed Description

VST Factory Presets Functions.

VST Factory Presets Functions

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.13.2 Function Documentation

4.13.2.1 void initStackPresets (HyperProgram * *hyperPresets*)

Sets up the hardcoded factory-programmed presets table into the given preset array.

Parameters

hyperPresets The destination preset array to write into.

Returns

nothing

4.14 HyperProcess.cpp File Reference

VST Audio Processing Functions.

4.14.1 Detailed Description

VST Audio Processing Functions.

VST Audio Processing Functions - main output module of the algorithm

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.15 HyperRandom.cpp File Reference

Random utilities.

Functions

- unsigned long _timeseed (void)
  Hashes time - for use as seeding for RNGs.
- void _srand_linc (unsigned long seed)
  Seeds linear congruential random number generator - works like c srand().
- unsigned long _rand_linc_custom (unsigned long A, unsigned long C)
  Linear congruential generator, given option to customise multiplier A and constant C.
- unsigned long _rand_linc (void)
  Linear congruential generator, using GCC constants.
- unsigned long _rand_linc_msg (void)
  Linear congruential generator, using Mininal Standard Generator constants.
- unsigned long _rand_linc_borlandc (void)
  Linear congruential generator, using Borland C/C++ constants.
- unsigned long _rand_linc_msvc (void)
  Linear congruential generator, using Microsoft Visual C++ constants.
- unsigned long _rand_xor (void)
  XORShift Random Number Generator.
- void _srand (unsigned long seed)
  Seeds Mersenne twister matrix - works like cstdlib srand().
- void _gen (void)
  Used interally by Mersenne twister to fill matrix.
- unsigned long _rand (void)
  Gives a (very) random number - works like cstdlib rand().
- float _randfloat (float min, float max)
  Gives a (very) random float, in a given range.
- unsigned long _randrange (unsigned long mod)
  Gives a (very) random number, in a given range.
- float _noise1 (int x)
  Noise function 1 - From Huge Elias' page.
- float _noise2 (int x)

Noise function 2 - Variation of noise1.

- float _noise2D (int x, int y)

  Noise function 2D - 2-Dimensional noise function.

- float _noiseSmooth (int x, int seed)

  Smoothed noise by weighted avg - Used internally (mostly).

- float _interpolateCosine (float a, float b, float frac)

  Does cosine interpolation between two values - Used internally (mostly).

- float _interpolateNoise (float x, int seed)

  Calculates smooth interpolated noise - Used internally (mostly).

- float _perlinNoise (float x, int seed, float persistence, int octaves)

  Perlin Noise Generator : 1-Dimensional.

- unsigned int **RSHash** (char *str, unsigned int len)
- unsigned int **JSHash** (char *str, unsigned int len)
- unsigned int **PJWHash** (char *str, unsigned int len)
- unsigned int **ELFHash** (char *str, unsigned int len)
- unsigned int **BKDRHash** (char *str, unsigned int len)
- unsigned int **SDBMHash** (char *str, unsigned int len)
- unsigned int **DJBHash** (char *str, unsigned int len)
- unsigned int **DEKHash** (char *str, unsigned int len)
- unsigned int **BPHash** (char *str, unsigned int len)
- unsigned int **FNVHash** (char *str, unsigned int len)
- unsigned int **APHash** (char *str, unsigned int len)
- unsigned int JenkinsHash (unsigned char *k, unsigned int len, unsigned int seed)

  Jenkins Hash - Pretty good, apparently.

- float _fabs (float f)

  Fast float absolute value.

- float _fneg (float f)

  Fast float negation.

- int _fsign (float f)

  Gives back +1 for 0 or positive numbers, -1 for negative numbers.

- float _invsqrt (float x)

  iD Software's famous inverse square root approximation

- float _sqrt (float x)

  Fast sqrt using iD Software inverse sqrt.

- int _floor_log2 (float f)
- int _log2 (float f)

  Fast approximation of log2.

- float fastexp3 (float x)
- float **fastexp4** (float x)
- float **fastexp5** (float x)
- float **fastexp6** (float x)
- float **fastexp7** (float x)
- float **fastexp8** (float x)
- float **fastexp9** (float x)
- float _flog2 (float val)
- float _fpow (float f, int n)

  Fast pow() f^n approximation - rough estimate!

- float _froot (float f, int n)

  Fast approximation for the n-th root of f.

- float _sinfast (float fAngle)

  Extra fast sin() approximation : [0, pi/2].

- float _sin (float fAngle)

  Fast sin() approximation : [0, pi/2].

- float _cosfast (float fAngle)

  Extra fast cos() approximation : [0, pi/2].

- float _cos (float fAngle)

  Fast cos() approximation : [0, pi/2].

- float _tanfast (float fAngle)

  Extra fast tan() approximation : [0,pi/4].

- float _tan (float fAngle)

  Fast tan() approximation : [0,pi/4].

- float _asin (float fValue)

  Fast inverse sin() approximation : [0, 1].

- float _acos (float fValue)

  Fast inverse cos() approximation : [0, 1].

- float _atanfast (float fValue)

  Extra fast inverse tan() approximation : [-1, 1].

- float _atan (float fValue)

  Fast inverse tan() approximation : [-1, 1].

- int _binomial_coefficient (int n, int m)

  Calculates nCm using dynamics programming. n, m < 100.

- int gcd (int a, int b)

  Calculates gcd using Euclid's algorithm.

- int _abs (int v)

  Branchless integer abs().

- int _min (int x, int y)

  Branchless min().

- int _max (int x, int y)

  Branchless max().

- unsigned int _nextpowerof2 (unsigned int v)

  Round up integer to next highest power of 2.

- float _gaussianNoise ()

4.15.1 Detailed Description

Random utilities.

Random utilities

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.15.2 Function Documentation

4.15.2.1 float _flog2 (float *val*)

Fast log^2 approximation assert (val > 0);

4.15.2.2 int _floor_log2 (float *f*)

Fast calculation of floor of log2 assert(f > 0.); assert(sizeof(f) == sizeof(int)); assert(sizeof(f) == 4);

4.15.2.3 float _gaussianNoise ()

Gaussian noise random number generator - range 1 to -1. Uses Mersenne twister, so seed using _srand().

4.15.2.4 int _log2 (float *f*)

Fast approximation of log2.

Fast approximation of log2 assert(f > 0.); assert(sizeof(f) == sizeof(int)); assert(sizeof(f) == 4);

4.15.2.5 float fastexp3 (float *x*)

Fast exp() approximation in the -3.14..3.14 range. The increasing numbers in the name of the function means increasing precision.

4.16 HyperRandom.h File Reference

Random utilities.

Functions

- unsigned long _timeseed (void)
  Hashes time - for use as seeding for RNGs.
- void _srand_linc (unsigned long seed=_timeseed())
  Seeds linear congruential random number generator - works like c srand().
- unsigned long _rand_linc_custom (unsigned long A, unsigned long C)
  Linear congruential generator, given option to customise multiplier A and constant C.
- unsigned long _rand_linc (void)
  Linear congruential generator, using GCC constants.
- unsigned long _rand_linc_msg (void)
  Linear congruential generator, using Mininal Standard Generator constants.
- unsigned long _rand_linc_borlandc (void)
  Linear congruential generator, using Borland C/C++ constants.
- unsigned long _rand_linc_msvc (void)
  Linear congruential generator, using Microsoft Visual C++ constants.
- unsigned long _rand_xor (void)
  XORShift Random Number Generator.
- void _srand (unsigned long seed=_timeseed())
  Seeds Mersenne twister matrix - works like cstdlib srand().
- void _gen (void)
  Used interally by Mersenne twister to fill matrix.
- unsigned long _rand (void)
  Gives a (very) random number - works like cstdlib rand().
- float _randfloat (float min, float max)
  Gives a (very) random float, in a given range.
- unsigned long _randrange (unsigned long mod)
  Gives a (very) random number, in a given range.
- float _noise1 (int x)
  Noise function 1 - From Huge Elias' page.
- float _noise2 (int x)

Noise function 2 - Variation of noise1.

- float _noise2D (int x, int y)

  Noise function 2D - 2-Dimensional noise function.

- float _noiseSmooth (int x, int seed)

  Smoothed noise by weighted avg - Used internally (mostly).

- float _interpolateCosine (float a, float b, float frac)

  Does cosine interpolation between two values - Used internally (mostly).

- float _interpolateNoise (float x, int seed)

  Calculates smooth interpolated noise - Used internally (mostly).

- float _perlinNoise (float x, int seed=0, float persistence=0.5f, int octaves=4)

  Perlin Noise Generator : 1-Dimensional.

- unsigned int **RSHash** (char *str, unsigned int len)
- unsigned int **JSHash** (char *str, unsigned int len)
- unsigned int **PJWHash** (char *str, unsigned int len)
- unsigned int **ELFHash** (char *str, unsigned int len)
- unsigned int **BKDRHash** (char *str, unsigned int len)
- unsigned int **SDBMHash** (char *str, unsigned int len)
- unsigned int **DJBHash** (char *str, unsigned int len)
- unsigned int **DEKHash** (char *str, unsigned int len)
- unsigned int **BPHash** (char *str, unsigned int len)
- unsigned int **FNVHash** (char *str, unsigned int len)
- unsigned int **APHash** (char *str, unsigned int len)
- unsigned int JenkinsHash (unsigned char *k, unsigned int len, unsigned int seed)

  Jenkins Hash - Pretty good, apparently.

- float _fabs (float f)

  Fast float absolute value.

- float _fneg (float f)

  Fast float negation.

- int _fsign (float f)

  Gives back +1 for 0 or positive numbers, -1 for negative numbers.

- float _invsqrt (float x)

  iD Software's famous inverse square root approximation

- float _sqrt (float x)

  Fast sqrt using iD Software inverse sqrt.

- int _floor_log2 (float f)
- int _log2 (float f)

  Fast approximation of log2.

- float fastexp3 (float x)
- float **fastexp4** (float x)
- float **fastexp5** (float x)
- float **fastexp6** (float x)
- float **fastexp7** (float x)
- float **fastexp8** (float x)
- float **fastexp9** (float x)
- float _flog2 (float val)
- float _fpow (float f, int n)

  Fast pow() f^n approximation - rough estimate!

- float _froot (float f, int n)

  Fast approximation for the n-th root of f.

- float _sinfast (float fAngle)

  Extra fast sin() approximation : [0, pi/2].

- float _sin (float fAngle)

  Fast sin() approximation : [0, pi/2].

- float _cosfast (float fAngle)

  Extra fast cos() approximation : [0, pi/2].

- float _cos (float fAngle)

  Fast cos() approximation : [0, pi/2].

- float _tanfast (float fAngle)

  Extra fast tan() approximation : [0,pi/4].

- float _tan (float fAngle)

  Fast tan() approximation : [0,pi/4].

- float _asin (float fValue)

  Fast inverse sin() approximation : [0, 1].

- float _acos (float fValue)

  Fast inverse cos() approximation : [0, 1].

- float _atanfast (float fValue)

  Extra fast inverse tan() approximation : [-1, 1].

- float _atan (float fValue)

  Fast inverse tan() approximation : [-1, 1].

- int _binomial_coefficient (int n, int m)

  Calculates nCm using dynamics programming. n, m < 100.

- int gcd (int a, int b)

  Calculates gcd using Euclid's algorithm.

- int _abs (int v)

  Branchless integer abs().

- int _min (int x, int y)

  Branchless min().

- int _max (int x, int y)

  Branchless max().

- unsigned int _nextpowerof2 (unsigned int v)

  Round up integer to next highest power of 2.

- float _gaussianNoise ()

4.16.1 Detailed Description

Random utilities.

Random utilities

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.16.2 Function Documentation

4.16.2.1 float _flog2 (float *val*)

Fast log^2 approximation assert (val > 0);

4.16.2.2 int _floor_log2 (float *f*)

Fast calculation of floor of log2 assert(f > 0.); assert(sizeof(f) == sizeof(int)); assert(sizeof(f) == 4);

4.16.2.3 float _gaussianNoise ()

Gaussian noise random number generator - range 1 to -1. Uses Mersenne twister, so seed using _srand().

4.16.2.4 int _log2 (float *f*)

Fast approximation of log2.

Fast approximation of log2 assert(f > 0.); assert(sizeof(f) == sizeof(int)); assert(sizeof(f) == 4);

4.16.2.5 float fastexp3 (float *x*)

Fast exp() approximation in the -3.14..3.14 range. The increasing numbers in the name of the function means increasing precision.

4.17 HyperStack.cpp File Reference

Main VST Effect Class.

4.17.1 Detailed Description

Main VST Effect Class.

Main VST Effect Class

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.18 HyperStack.h File Reference

Main VST Effect Class.

Data Structures

- struct HyperProgram

  Structure to store and manage a preset's parameter array.

- class HyperStack

  Main hyperSTACK VST effect class.

Enumerations

- enum enum_parameter_indexes {
  F_VOLUME, **F_NUMOSC**, **F_QUALITY**, **F_WAVESHAPE**,
  F_SEMI, **F_FINE**, **F_ATTACK**, **F_SUSTAIN**,
  F_DECAY, **F_RELEASE**, **F_ATTACK_DEV**, **F_SUSTAIN_DEV**,
  F_DECAY_DEV, **F_RELEASE_DEV**, **F_DETUNE_DEV**, **F_CUTOFF**,
  F_RESONANCE, **F_PLFO_GAIN**, **F_PLFO_FREQUENCY**, **F_PLFO_FREQ_DEV**,
  F_PLFO_AMOUNT, **F_PLFO_NOISE**, **F_PLFO_NOISEFREQ**, **F_PLFO_ATTACK**,
  F_PLFO_SUSTAIN, **F_PLFO_DECAY**, **F_PLFO_RELEASE**, **F_FILTER_ATTACK**,
  F_FILTER_SUSTAIN, **F_FILTER_DECAY**, **F_FILTER_RELEASE**, **F_FILTER_AMOUNT**
  }

  Parameter index constants.

Functions

- void initStackPresets (HyperProgram *hyperPresets)

4.18.1 Detailed Description

Main VST Effect Class.

Main VST Effect Class

Copyright (C) 2010 Xi Chen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.18.2 Function Documentation

4.18.2.1 void initStackPresets (HyperProgram * *hyperPresets*)

Sets up the hardcoded factory-programmed presets table into the given preset array.

Parameters

hyperPresets The destination preset array to write into.

Returns

nothing

4.19 HyperStackMain.cpp File Reference

VST Effect Plugin Entry Point.

Functions

- AEffect * VSTPluginMain (audioMasterCallback audioMaster)
  The main VST entry point.
- BOOL WINAPI DllMain (HINSTANCE hInst, DWORD dwReason, LPVOID lpvReserved)
  WIN32 Platform DLL Entry Point.

Variables

- bool **OOME** = false
- void * **hInstance**

4.19.1 Detailed Description

VST Effect Plugin Entry Point.

VST Effect Plugin Entry Point

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.20 HyperTuner.cpp File Reference

MIDI Note Frequency Calculator.

4.20.1 Detailed Description

MIDI Note Frequency Calculator.

MIDI Note Frequency Calculator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.21 HyperTuner.h File Reference

MIDI Note Frequency Calculator.

Data Structures

- class HyperTuner

  Static class providing functions for frequency calculation.

4.21.1 Detailed Description

MIDI Note Frequency Calculator.

MIDI Note Frequency Calculator

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

4.22 HyperUtil.h File Reference

Small Inline Utility Functions.

4.22.1 Detailed Description

Small Inline Utility Functions.

Small Inline Utility Functions

Author

Xi Chen hypernewbie@gmail.com

Date

21/11/2010

Index