

DaisySP

Generated by Doxygen 1.8.18

1 Main Page	1
1.0.0.1 Getting Started	1
1.0.0.2 Contributing	1
1.0.0.3 License	1
2 Todo List	3
3 Class Index	5
3.1 Class List	5
4 Class Documentation	7
4.1 daisysp::AdEnv Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Function Documentation	7
4.1.2.1 GetCurrentSegment()	8
4.1.2.2 GetValue()	8
4.1.2.3 Init()	8
4.1.2.4 IsRunning()	8
4.1.2.5 Process()	8
4.1.2.6 SetCurve()	9
4.1.2.7 SetMax()	9
4.1.2.8 SetMin()	9
4.1.2.9 SetTime()	9
4.1.2.10 Trigger()	9
4.2 daisysp::Adsr Class Reference	9
4.2.1 Detailed Description	10
4.2.2 Member Function Documentation	10
4.2.2.1 GetCurrentSegment()	10
4.2.2.2 Init()	10
4.2.2.3 IsRunning()	11
4.2.2.4 Process()	11
4.2.2.5 SetSustainLevel()	11
4.2.2.6 SetTime()	11
4.3 daisysp::ATone Class Reference	12
4.3.1 Detailed Description	12
4.3.2 Member Function Documentation	12
4.3.2.1 GetFreq()	12
4.3.2.2 Init()	12
4.3.2.3 Process()	13
4.3.2.4 SetFreq()	13
4.4 daisysp::Autowah Class Reference	13
4.4.1 Detailed Description	14
4.4.2 Member Function Documentation	14

4.4.2.1 Init()	14
4.4.2.2 Process()	14
4.4.2.3 SetDryWet()	14
4.4.2.4 SetLevel()	15
4.4.2.5 SetWah()	15
4.5 daisysp::Balance Class Reference	15
4.5.1 Detailed Description	16
4.5.2 Member Function Documentation	16
4.5.2.1 Init()	16
4.5.2.2 Process()	16
4.5.2.3 SetCutoff()	16
4.6 daisysp::Biquad Class Reference	17
4.6.1 Detailed Description	17
4.6.2 Member Function Documentation	17
4.6.2.1 Init()	17
4.6.2.2 Process()	18
4.6.2.3 SetCutoff()	18
4.6.2.4 SetRes()	18
4.7 daisysp::Bitcrush Class Reference	18
4.7.1 Detailed Description	19
4.7.2 Member Function Documentation	19
4.7.2.1 Init()	19
4.7.2.2 Process()	19
4.7.2.3 SetBitDepth()	19
4.7.2.4 SetCrushRate()	20
4.8 daisysp::BIOsc Class Reference	20
4.8.1 Detailed Description	20
4.8.2 Member Enumeration Documentation	21
4.8.2.1 Waveforms	21
4.8.3 Member Function Documentation	21
4.8.3.1 Init()	21
4.8.3.2 Process()	21
4.8.3.3 SetAmp()	21
4.8.3.4 SetFreq()	21
4.8.3.5 SetPw()	22
4.8.3.6 SetWaveform()	22
4.9 daisysp::Comb Class Reference	22
4.9.1 Detailed Description	22
4.9.2 Member Function Documentation	22
4.9.2.1 Init()	22
4.9.2.2 Process()	23
4.9.2.3 SetFreq()	23

4.9.2.4 SetRevTime()	23
4.10 daisysp::Compressor Class Reference	23
4.10.1 Detailed Description	24
4.10.2 Member Function Documentation	24
4.10.2.1 Init()	24
4.10.2.2 Process() [1/2]	24
4.10.2.3 Process() [2/2]	25
4.10.2.4 SetAttack()	25
4.10.2.5 SetRatio()	25
4.10.2.6 SetRelease()	25
4.10.2.7 SetThreshold()	25
4.11 daisysp::CrossFade Class Reference	26
4.11.1 Detailed Description	26
4.11.2 Member Function Documentation	26
4.11.2.1 GetCurve()	26
4.11.2.2 GetPos()	26
4.11.2.3 Init() [1/2]	26
4.11.2.4 Init() [2/2]	27
4.11.2.5 Process()	27
4.11.2.6 SetCurve()	27
4.11.2.7 SetPos()	27
4.12 daisysp::DcBlock Class Reference	27
4.12.1 Detailed Description	28
4.12.2 Member Function Documentation	28
4.12.2.1 Init()	28
4.12.2.2 Process()	28
4.13 daisysp::Decimator Class Reference	28
4.13.1 Detailed Description	28
4.13.2 Member Function Documentation	29
4.13.2.1 GetBitcrushFactor()	29
4.13.2.2 GetDownsampleFactor()	29
4.13.2.3 Init()	29
4.13.2.4 Process()	29
4.13.2.5 SetBitcrushFactor()	29
4.13.2.6 SetBitsToCrush()	29
4.13.2.7 SetDownsampleFactor()	30
4.14 daisysp::DelayLine< T, max_size > Class Template Reference	30
4.14.1 Detailed Description	30
4.14.2 Member Function Documentation	30
4.14.2.1 Init()	31
4.14.2.2 Read()	31
4.14.2.3 Reset()	31

4.14.2.4 SetDelay() [1/2]	31
4.14.2.5 SetDelay() [2/2]	31
4.14.2.6 Write()	31
4.15 daisysp::Fold Class Reference	32
4.15.1 Detailed Description	32
4.15.2 Member Function Documentation	32
4.15.2.1 Init()	32
4.15.2.2 Process()	32
4.15.2.3 SetIncrement()	32
4.16 daisysp::Limiter Class Reference	33
4.16.1 Detailed Description	33
4.16.2 Member Function Documentation	33
4.16.2.1 Init()	33
4.16.2.2 ProcessBlock()	33
4.17 daisysp::Line Class Reference	34
4.17.1 Detailed Description	34
4.17.2 Member Function Documentation	34
4.17.2.1 Init()	34
4.17.2.2 Process()	34
4.17.2.3 Start()	34
4.18 daisysp::Maytrig Class Reference	35
4.18.1 Detailed Description	35
4.18.2 Member Function Documentation	35
4.18.2.1 Process()	35
4.19 daisysp::Metro Class Reference	36
4.19.1 Detailed Description	36
4.19.2 Member Function Documentation	36
4.19.2.1 GetFreq()	36
4.19.2.2 Init()	36
4.19.2.3 Process()	37
4.19.2.4 Reset()	37
4.19.2.5 SetFreq()	37
4.20 daisysp::Mode Class Reference	37
4.20.1 Detailed Description	37
4.20.2 Member Function Documentation	38
4.20.2.1 Clear()	38
4.20.2.2 Init()	38
4.20.2.3 Process()	38
4.20.2.4 SetFreq()	38
4.20.2.5 SetQ()	38
4.21 daisysp::MoogLadder Class Reference	38
4.21.1 Detailed Description	39

4.21.2 Member Function Documentation	39
4.21.2.1 Init()	39
4.21.2.2 Process()	39
4.21.2.3 SetFreq()	39
4.21.2.4 SetRes()	40
4.22 daisysp::NIFilt Class Reference	40
4.22.1 Detailed Description	40
4.22.2 Member Function Documentation	41
4.22.2.1 Init()	41
4.22.2.2 ProcessBlock()	41
4.22.2.3 SetA()	41
4.22.2.4 SetB()	41
4.22.2.5 SetC()	41
4.22.2.6 SetCoefficients()	41
4.22.2.7 SetD()	42
4.22.2.8 SetL()	42
4.23 daisysp::Oscillator Class Reference	42
4.23.1 Detailed Description	42
4.23.2 Member Enumeration Documentation	42
4.23.2.1 anonymous enum	43
4.23.3 Member Function Documentation	43
4.23.3.1 Init()	43
4.23.3.2 PhaseAdd()	43
4.23.3.3 Process()	43
4.23.3.4 Reset()	44
4.23.3.5 SetAmp()	44
4.23.3.6 SetFreq()	44
4.23.3.7 SetWaveform()	44
4.24 daisysp::Phasor Class Reference	44
4.24.1 Detailed Description	45
4.24.2 Member Function Documentation	45
4.24.2.1 GetFreq()	45
4.24.2.2 Init() [1/3]	45
4.24.2.3 Init() [2/3]	45
4.24.2.4 Init() [3/3]	45
4.24.2.5 Process()	46
4.24.2.6 SetFreq()	46
4.25 daisysp::PitchShifter Class Reference	46
4.25.1 Detailed Description	46
4.25.2 Member Function Documentation	47
4.25.2.1 Init()	47
4.25.2.2 Process()	47

4.25.2.3 SetDelSize()	47
4.25.2.4 SetFun()	47
4.25.2.5 SetTransposition()	47
4.26 daisysp::Pluck Class Reference	47
4.26.1 Detailed Description	48
4.26.2 Member Function Documentation	48
4.26.2.1 GetAmp()	48
4.26.2.2 GetDamp()	48
4.26.2.3 GetDecay()	48
4.26.2.4 GetFreq()	49
4.26.2.5 GetMode()	49
4.26.2.6 Init()	49
4.26.2.7 Process()	49
4.26.2.8 SetAmp()	49
4.26.2.9 SetDamp()	49
4.26.2.10 SetDecay()	50
4.26.2.11 SetFreq()	50
4.26.2.12 SetMode()	50
4.27 daisysp::PolyPluck< num_voices > Class Template Reference	50
4.27.1 Detailed Description	50
4.27.2 Member Function Documentation	51
4.27.2.1 Init()	51
4.27.2.2 Process()	51
4.27.2.3 SetDecay()	51
4.28 daisysp::Port Class Reference	52
4.28.1 Detailed Description	52
4.28.2 Member Function Documentation	52
4.28.2.1 GetHtime()	52
4.28.2.2 Init()	52
4.28.2.3 Process()	53
4.28.2.4 SetHtime()	53
4.29 daisysp::ReverbSc Class Reference	53
4.29.1 Detailed Description	54
4.29.2 Member Function Documentation	54
4.29.2.1 Init()	54
4.29.2.2 Process()	54
4.29.2.3 SetFeedback()	54
4.29.2.4 SetLpFreq()	55
4.30 daisysp::ReverbScDI Struct Reference	55
4.30.1 Detailed Description	55
4.30.2 Member Data Documentation	55
4.30.2.1 buf	55

4.30.2.2 buffer_size	56
4.30.2.3 dummy	56
4.30.2.4 filter_state	56
4.30.2.5 rand_line_cnt	56
4.30.2.6 read_pos	56
4.30.2.7 read_pos_frac	56
4.30.2.8 read_pos_frac_inc	56
4.30.2.9 seed_val	56
4.30.2.10 write_pos	57
4.31 daisysp::Svf Class Reference	57
4.31.1 Detailed Description	57
4.31.2 Member Function Documentation	57
4.31.2.1 Band()	58
4.31.2.2 High()	58
4.31.2.3 Init()	58
4.31.2.4 Low()	58
4.31.2.5 Notch()	59
4.31.2.6 Peak()	59
4.31.2.7 Process()	59
4.31.2.8 SetDrive()	59
4.31.2.9 SetFreq()	59
4.31.2.10 SetRes()	60
4.32 daisysp::Tone Class Reference	60
4.32.1 Detailed Description	60
4.32.2 Member Function Documentation	60
4.32.2.1 GetFreq()	60
4.32.2.2 Init()	60
4.32.2.3 Process()	61
4.32.2.4 SetFreq()	61
4.33 daisysp::WhiteNoise Class Reference	61
4.33.1 Detailed Description	61
4.33.2 Member Function Documentation	61
4.33.2.1 Init()	62
4.33.2.2 Process()	62
4.33.2.3 SetAmp()	62

Chapter 1

Main Page

DSP Library for the Daisy product family...and elsewhere!

DaisySP is an open source DSP library written in C++ and specifically tailored to embedded audio applications.

1.0.0.1 Getting Started

- Browse the reference documentation at `/doc/`
- Check out our [How to Build](#) Wiki page.
- Make some sound!

1.0.0.2 Contributing

We'd love to have you become a contributor!

Here are ways that you can get involved:

- Make new DSP modules. See issues labeled "feature".
- Port existing DSP modules from other open source projects (MIT). See issues labeled "port".
- Fix problems with existing modules. See issues labeled "bug" and/or "polish".
- Test existing functionality and make issues.

Before working on code, please check out our [Contribution Guidelines](#) and `/doc/StyleGuide.pdf`

1.0.0.3 License

DaisySP is licensed with the permissive MIT open source license.

This allows for modification and reuse in both commercial and personal projects. It does not provide a warranty of any kind.

For the full license, read the [LICENSE](#) file in the root directory.

Chapter 2

Todo List

Class `daisysp::AdEnv`

- Add Cycling
- Implement Curve (its only linear for now).
- Maybe make this an `ADsr_` that has `AD/AR/Asr_` modes.

Class `daisysp::Compressor`

With fixed controls this is relatively quick, but changing controls now costs a lot more

Still pretty expensive

Add soft/hard knee settings

Maybe make stereo possible? (needing two for stereo is a bit silly, and their gain shouldn't be totally unique.

Class `daisysp::NIFilt`

make this work on a single sample instead of just on blocks at a time.

Class `daisysp::Phasor`

Selecting which channels should be initialized/included in the sequence conversion.

Setup a similar start function for an external mux, but that seems outside the scope of this file.

Class `daisysp::PitchShifter`

- move `hash_xs32` and `myrand` to `dsp.h` and give appropriate names

Chapter 3

Class Index

3.1 Class List

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

daisysp::AdEnv	7
daisysp::Adsr	9
daisysp::ATone	12
daisysp::Autowah	13
daisysp::Balance	15
daisysp::Biquad	17
daisysp::Bitcrush	18
daisysp::BIOsc	20
daisysp::Comb	22
daisysp::Compressor	23
daisysp::CrossFade	26
daisysp::DcBlock	27
daisysp::Decimator	28
daisysp::DelayLine< T, max_size >	30
daisysp::Fold	32
daisysp::Limiter	33
daisysp::Line	34
daisysp::Maytrig	35
daisysp::Metro	36
daisysp::Mode	37
daisysp::MoogLadder	38
daisysp::NIFilt	40
daisysp::Oscillator	42
daisysp::Phasor	44
daisysp::PitchShifter	46
daisysp::Pluck	47
daisysp::PolyPluck< num_voices >	50
daisysp::Port	52
daisysp::ReverbSc	53
daisysp::ReverbScDI	55
daisysp::Svf	57
daisysp::Tone	60
daisysp::WhiteNoise	61

Chapter 4

Class Documentation

4.1 daisysp::AdEnv Class Reference

```
#include <adenv.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) ()
- void [Trigger](#) ()
- void [SetTime](#) (uint8_t seg, float time)
- void [SetCurve](#) (float scalar)
- void [SetMin](#) (float min)
- void [SetMax](#) (float max)
- float [GetValue](#) () const
- uint8_t [GetCurrentSegment](#) ()
- bool [IsRunning](#) () const

4.1.1 Detailed Description

Trigger-able envelope with adjustable min/max, and independent per-segment time control.

Author

shensley

- Todo**
- Add Cycling
 - Implement Curve (its only linear for now).
 - Maybe make this an ADsr_ that has AD/AR/Asr_ modes.

4.1.2 Member Function Documentation

4.1.2.1 GetCurrentSegment()

```
uint8_t daisysp::AdEnv::GetCurrentSegment ( ) [inline]
```

Returns the segment of the envelope that the phase is currently located in.

4.1.2.2 GetValue()

```
float daisysp::AdEnv::GetValue ( ) const [inline]
```

Returns the current output value without processing the next sample

4.1.2.3 Init()

```
void AdEnv::Init (
    float sample_rate )
```

Initializes the ad envelope.

Defaults:

- current segment = idle
- curve = linear
- phase = 0
- min = 0
- max = 1

Parameters

<i>sample_rate</i>	sample rate of the audio engine being run
--------------------	---

4.1.2.4 IsRunning()

```
bool daisysp::AdEnv::IsRunning ( ) const [inline]
```

Returns true if the envelope is currently in any stage apart from idle.

4.1.2.5 Process()

```
float AdEnv::Process ( )
```

Processes the current sample of the envelope. This should be called once per sample period.

Returns

the current envelope value.

4.1.2.6 SetCurve()

```
void daisysp::AdEnv::SetCurve (
    float scalar ) [inline]
```

Sets the amount of curve applied. A positive value will create a log curve. Input range: -100 to 100. (or more)

4.1.2.7 SetMax()

```
void daisysp::AdEnv::SetMax (
    float max ) [inline]
```

Sets the maximum value of the envelope output. Input range: -FLTmax_, to FLTmax_

4.1.2.8 SetMin()

```
void daisysp::AdEnv::SetMin (
    float min ) [inline]
```

Sets the minimum value of the envelope output. Input range: -FLTmax_, to FLTmax_

4.1.2.9 SetTime()

```
void daisysp::AdEnv::SetTime (
    uint8_t seg,
    float time ) [inline]
```

Sets the length of time (in seconds) for a specific segment.

4.1.2.10 Trigger()

```
void daisysp::AdEnv::Trigger ( ) [inline]
```

Starts or retriggers the envelope.

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

- modules/adenv.h
- modules/adenv.cpp

4.2 daisysp::Adsr Class Reference

```
#include <adsr.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (bool gate)
- void [SetTime](#) (int seg, float time)
- void [SetSustainLevel](#) (float sus_level)
- uint8_t [GetCurrentSegment](#) ()
- bool [IsRunning](#) () const

4.2.1 Detailed Description

adsr envelope module

Original author(s) : Paul Batchelor

Ported from Soundpipe by Ben Sergentanis, May 2020

4.2.2 Member Function Documentation

4.2.2.1 GetCurrentSegment()

```
uint8_t daisysp::Adsr::GetCurrentSegment ( ) [inline]
```

get the current envelope segment

Returns

the segment of the envelope that the phase is currently located in.

4.2.2.2 Init()

```
void Adsr::Init (
    float sample_rate )
```

Initializes the [ATone](#) module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.2.2.3 IsRunning()

```
bool daisysp::Adsr::IsRunning ( ) const [inline]
```

Tells whether envelope is active

Returns

true if the envelope is currently in any stage apart from idle.

4.2.2.4 Process()

```
float Adsr::Process (
    bool gate )
```

Processes one sample through the filter and returns one sample.

Parameters

<i>gate</i>	- trigger the envelope, hold it to sustain
-------------	--

4.2.2.5 SetSustainLevel()

```
void daisysp::Adsr::SetSustainLevel (
    float sus_level ) [inline]
```

Sustain level

Parameters

<i>sus_level</i>	- sets sustain level
------------------	----------------------

4.2.2.6 SetTime()

```
void daisysp::Adsr::SetTime (
    int seg,
    float time ) [inline]
```

Sets time Set time per segment in seconds

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

- modules/adsr.h
- modules/adsr.cpp

4.3 daisysp::ATone Class Reference

```
#include <atone.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float &in)
- void [SetFreq](#) (float &freq)
- float [GetFreq](#) ()

4.3.1 Detailed Description

A first-order recursive high-pass filter with variable frequency response. Original Author(s): Barry Vercoe, John Ffitch, Gabriel Maldonado

Year: 1991

Original Location: Csound – OOps/ugens5.c

Ported from soundpipe by Ben Sergentanis, May 2020

4.3.2 Member Function Documentation

4.3.2.1 GetFreq()

```
float daisysp::ATone::GetFreq ( ) [inline]
```

get current frequency

Returns

the current value for the cutoff frequency or half-way point of the filter.

4.3.2.2 Init()

```
void ATone::Init (
    float sample_rate )
```

Initializes the [ATone](#) module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.3.2.3 Process()

```
float ATone::Process (
    float & in )
```

Processes one sample through the filter and returns one sample.

Parameters

<i>in</i>	- input signal
-----------	----------------

4.3.2.4 SetFreq()

```
void daisysp::ATone::SetFreq (
    float & freq ) [inline]
```

Sets the cutoff frequency or half-way point of the filter.

Parameters

<i>freq</i>	- frequency value in Hz. Range: Any positive value.
-------------	---

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

- modules/atone.h
- modules/atone.cpp

4.4 daisysp::Autowah Class Reference

```
#include <autowah.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)
- void [SetWah](#) (float wah)
- void [SetDryWet](#) (float drywet)
- void [SetLevel](#) (float level)

4.4.1 Detailed Description

[Autowah](#) module

Original author(s) :

Ported from soundpipe by Ben Sergentanis, May 2020

4.4.2 Member Function Documentation

4.4.2.1 Init()

```
void Autowah::Init (
    float sample_rate )
```

Initializes the [Autowah](#) module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.4.2.2 Process()

```
float Autowah::Process (
    float in )
```

Initializes the [Autowah](#) module.

Parameters

<i>in</i>	- input signal to be wah'd
-----------	----------------------------

4.4.2.3 SetDryWet()

```
void daisysp::Autowah::SetDryWet (
    float drywet ) [inline]
```

sets mix amount

Parameters

<i>drywet</i>	: set effect dry/wet
---------------	----------------------

4.4.2.4 SetLevel()

```
void daisysp::Autowah::SetLevel (
    float level ) [inline]
```

sets wah level

Parameters

<i>level</i>	: set wah level
--------------	-----------------

4.4.2.5 SetWah()

```
void daisysp::Autowah::SetWah (
    float wah ) [inline]
```

sets wah

Parameters

<i>wah</i>	: set wah amount
------------	------------------

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

- modules/autowah.h
- modules/autowah.cpp

4.5 daisysp::Balance Class Reference

```
#include <balance.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float sig, float comp)
- void [SetCutoff](#) (float cutoff)

4.5.1 Detailed Description

Balances two sound sources. Sig is boosted to the level of comp.

Original author(s) : Barry Vercoe, john ffitch, Gabriel Maldonado

Year: 1991

Ported from soundpipe by Ben Sergentanis, May 2020

4.5.2 Member Function Documentation

4.5.2.1 Init()

```
void Balance::Init (
    float sample_rate )
```

Initializes the balance module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.5.2.2 Process()

```
float Balance::Process (
    float sig,
    float comp )
```

adjust sig level to level of comp

4.5.2.3 SetCutoff()

```
void daisysp::Balance::SetCutoff (
    float cutoff ) [inline]
```

adjusts the rate at which level compensation happens

Parameters

<i>cutoff</i>	: Sets half power point of special internal cutoff filter.
---------------	--

defaults to 10

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

- modules/balance.h
- modules/balance.cpp

4.6 daisysp::Biquad Class Reference

```
#include <biquad.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)
- void [SetRes](#) (float res)
- void [SetCutoff](#) (float cutoff)

4.6.1 Detailed Description

Two pole recursive filter

Original author(s) : Hans Mikelson

Year: 1998

Ported from soundpipe by Ben Sergentanis, May 2020

4.6.2 Member Function Documentation

4.6.2.1 Init()

```
void Biquad::Init (  
    float sample_rate )
```

Initializes the biquad module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.6.2.2 Process()

```
float Biquad::Process (
    float in )
```

Filters the input signal

Returns

filtered output

4.6.2.3 SetCutoff()

```
void daisysp::Biquad::SetCutoff (
    float cutoff ) [inline]
```

Sets filter cutoff in Hz

Parameters

<i>cutoff</i>	: Set filter cutoff.
---------------	----------------------

4.6.2.4 SetRes()

```
void daisysp::Biquad::SetRes (
    float res ) [inline]
```

Sets resonance amount

Parameters

<i>res</i>	: Set filter resonance.
------------	-------------------------

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

- modules/biquad.h
- modules/biquad.cpp

4.7 daisysp::Bitcrush Class Reference

```
#include <bitcrush.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)
- void [SetBitDepth](#) (int bitdepth)
- void [SetCrushRate](#) (float crushrate)

4.7.1 Detailed Description

bitcrush module

Original author(s) : Paul Batchelor,

Ported from soundpipe by Ben Sergentanis, May 2020

4.7.2 Member Function Documentation

4.7.2.1 Init()

```
void Bitcrush::Init (
    float sample_rate )
```

Initializes the bitcrush module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
--------------------	--

4.7.2.2 Process()

```
float Bitcrush::Process (
    float in )
```

bit crushes and downsamples the input

4.7.2.3 SetBitDepth()

```
void daisysp::Bitcrush::SetBitDepth (
    int bitdepth ) [inline]
```

adjusts bitdepth

Parameters

<i>bitdepth</i>	: Sets bit depth.
-----------------	-------------------

4.7.2.4 SetCrushRate()

```
void daisysp::Bitcrush::SetCrushRate (
    float crushrate ) [inline]
```

adjusts the downsampling frequency

Parameters

<i>crushrate</i>	: Sets rate to downsample to.
------------------	-------------------------------

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

- modules/bitcrush.h
- modules/bitcrush.cpp

4.8 daisysp::BIOsc Class Reference

```
#include <blosc.h>
```

Public Types

- enum [Waveforms](#) { [WAVE_TRIANGLE](#), [WAVE_SAW](#), [WAVE_SQUARE](#), [WAVE_OFF](#) }

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) ()
- void [SetFreq](#) (float freq)
- void [SetAmp](#) (float amp)
- void [SetPw](#) (float pw)
- void [SetWaveform](#) (uint8_t waveform)

4.8.1 Detailed Description

Band Limited [Oscillator](#)

Based on bltriangle, blsaw, blsquare from soundpipe

Original Author(s): Paul Batchelor, saw2 Faust by Julius Smith

Ported by Ben Sergentanis, May 2020

4.8.2 Member Enumeration Documentation

4.8.2.1 Waveforms

```
enum daisysp::BlOsc::Waveforms
```

Bl Waveforms

4.8.3 Member Function Documentation

4.8.3.1 Init()

```
void BlOsc::Init (  
    float sample_rate )
```

-Initialize oscillator. -Defaults to: 440Hz, .5 amplitude, .5 pw, Triangle.

4.8.3.2 Process()

```
float BlOsc::Process ( )
```

- Get next floating point oscillator sample.

4.8.3.3 SetAmp()

```
void daisysp::BlOsc::SetAmp (  
    float amp ) [inline]
```

- Float amp: Set oscillator amplitude, 0 to 1.

4.8.3.4 SetFreq()

```
void daisysp::BlOsc::SetFreq (  
    float freq ) [inline]
```

- Float freq: Set oscillator frequency in Hz.

4.8.3.5 SetPw()

```
void daisysp::BlOsc::SetPw (
    float pw ) [inline]
```

- Float pw: Set square osc pulsewidth, 0 to 1. (no thru 0 at the moment)

4.8.3.6 SetWaveform()

```
void daisysp::BlOsc::SetWaveform (
    uint8_t waveform ) [inline]
```

- uint8_t waveform: select between waveforms from enum above.
- i.e. SetWaveform(BL_WAVEFORM_SAW); to set waveform to saw

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

- modules/blosc.h
- modules/blosc.cpp

4.9 daisysp::Comb Class Reference

```
#include <comb.h>
```

Public Member Functions

- void [Init](#) (float sample_rate, float *buff, size_t size)
- float [Process](#) (float in)
- void [SetFreq](#) (float looptime)
- void [SetRevTime](#) (float revtime)

4.9.1 Detailed Description

[Comb](#) filter module

Original author(s) :

Ported from soundpipe by Ben Sergentanis, May 2020

4.9.2 Member Function Documentation

4.9.2.1 Init()

```
void Comb::Init (
    float sample_rate,
    float * buff,
    size_t size )
```

Initializes the [Comb](#) module.

Parameters

<i>sample_rate</i>	- The sample rate of the audio engine being run.
<i>buff</i>	- input buffer, kept in either main() or global space
<i>size</i>	- size of buff

4.9.2.2 Process()

```
float Comb::Process (
    float in )
```

processes the comb filter

4.9.2.3 SetFreq()

```
void Comb::SetFreq (
    float looptime )
```

Sets the frequency of the comb filter

4.9.2.4 SetRevTime()

```
void daisysp::Comb::SetRevTime (
    float revtime ) [inline]
```

Sets the decay time of the comb filter

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

- modules/comb.h
- modules/comb.cpp

4.10 daisysp::Compressor Class Reference

```
#include <compressor.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in, float key)
- float [Process](#) (float in)
- void [SetRatio](#) (const float &ratio)
- void [SetThreshold](#) (const float &thresh)
- void [SetAttack](#) (const float &atk)
- void [SetRelease](#) (const float &rel)

4.10.1 Detailed Description

dynamics compressor

influenced by compressor in soundpipe (from faust).

Modifications made to do:

- Less calculations during each process loop (coefficients recalculated on parameter change).
- C++-ified
- added sidechain support

by: shensley

Todo With fixed controls this is relatively quick, but changing controls now costs a lot more

Still pretty expensive

Add soft/hard knee settings

Maybe make stereo possible? (needing two for stereo is a bit silly, and their gain shouldn't be totally unique.

4.10.2 Member Function Documentation

4.10.2.1 Init()

```
void Compressor::Init (  
    float sample_rate )
```

Initializes compressor sample_rate - rate at which samples will be produced by the audio engine.

4.10.2.2 Process() [1/2]

```
float Compressor::Process (  
    float in )
```

compresses the audio input signal

Parameters

<i>in</i>	- audio input signal (to be compressed)
-----------	---

4.10.2.3 Process() [2/2]

```
float Compressor::Process (
    float in,
    float key )
```

compresses the audio input signal, keyed by a secondary input.

Parameters

<i>in</i>	- audio input signal (to be compressed)
<i>key</i>	- audio input that will be used to side-chain the compressor.

4.10.2.4 SetAttack()

```
void daisysp::Compressor::SetAttack (
    const float & atk ) [inline]
```

envelope time for onset of compression for signals above the threshold. Expects 0.001 -> 10

4.10.2.5 SetRatio()

```
void daisysp::Compressor::SetRatio (
    const float & ratio ) [inline]
```

amount of gain reduction applied to compressed signals Expects 1.0 -> 40. (untested with values < 1.0)

4.10.2.6 SetRelease()

```
void daisysp::Compressor::SetRelease (
    const float & rel ) [inline]
```

envelope time for release of compression as input signal falls below threshold. Expects 0.001 -> 10

4.10.2.7 SetThreshold()

```
void daisysp::Compressor::SetThreshold (
    const float & thresh ) [inline]
```

threshold in dB at which compression will be applied Expects 0.0 -> -80.

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

- modules/compressor.h
- modules/compressor.cpp

4.11 daisysp::CrossFade Class Reference

```
#include <crossfade.h>
```

Public Member Functions

- void [Init](#) (int curve)
- void [Init](#) ()
- float [Process](#) (float &in1, float &in2)
- void [SetPos](#) (float pos)
- void [SetCurve](#) (uint8_t curve)
- float [GetPos](#) (float pos)
- uint8_t [GetCurve](#) (uint8_t curve)

4.11.1 Detailed Description

Performs a [CrossFade](#) between two signals

Original author: Paul Batchelor

Ported from Soundpipe by Andrew Ikenberry

added curve option for constant power, etc.

4.11.2 Member Function Documentation

4.11.2.1 [GetCurve\(\)](#)

```
uint8_t daisysp::CrossFade::GetCurve (  
    uint8_t curve ) [inline]
```

Returns current curve

4.11.2.2 [GetPos\(\)](#)

```
float daisysp::CrossFade::GetPos (  
    float pos ) [inline]
```

Returns current position

4.11.2.3 [Init\(\)](#) [1/2]

```
void daisysp::CrossFade::Init ( ) [inline]
```

Initialize with default linear curve

4.11.2.4 Init() [2/2]

```
void daisysp::CrossFade::Init (
    int curve ) [inline]
```

Initializes [CrossFade](#) module Defaults

- current position = .5
- curve = linear

4.11.2.5 Process()

```
float CrossFade::Process (
    float & in1,
    float & in2 )
```

processes [CrossFade](#) and returns single sample

4.11.2.6 SetCurve()

```
void daisysp::CrossFade::SetCurve (
    uint8_t curve ) [inline]
```

Sets current curve applied to [CrossFade](#) Expected input: See [Curve Options](#)

4.11.2.7 SetPos()

```
void daisysp::CrossFade::SetPos (
    float pos ) [inline]
```

Sets position of [CrossFade](#) between two input signals Input range: 0 to 1

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

- modules/crossfade.h
- modules/crossfade.cpp

4.12 daisysp::DcBlock Class Reference

```
#include <dcblock.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)

4.12.1 Detailed Description

Removes DC component of a signal

4.12.2 Member Function Documentation

4.12.2.1 Init()

```
void DcBlock::Init (
    float sample_rate )
```

Initializes [DcBlock](#) module

4.12.2.2 Process()

```
float DcBlock::Process (
    float in )
```

performs [DcBlock](#) Process

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

- modules/dcblock.h
- modules/dcblock.cpp

4.13 daisysp::Decimator Class Reference

```
#include <decimator.h>
```

Public Member Functions

- void [Init](#) ()
- float [Process](#) (float input)
- void [SetDownsampleFactor](#) (float downsample_factor)
- void [SetBitcrushFactor](#) (float bitcrush_factor)
- void [SetBitsToCrush](#) (const uint8_t &bits)
- float [GetDownsampleFactor](#) ()
- float [GetBitcrushFactor](#) ()

4.13.1 Detailed Description

Performs downsampling and bitcrush effects

4.13.2 Member Function Documentation

4.13.2.1 GetBitcrushFactor()

```
float daisysp::Decimator::GetBitcrushFactor ( ) [inline]
```

Returns current setting of bitcrush

4.13.2.2 GetDownsampleFactor()

```
float daisysp::Decimator::GetDownsampleFactor ( ) [inline]
```

Returns current setting of downsample

4.13.2.3 Init()

```
void Decimator::Init ( )
```

Initializes downsample module

4.13.2.4 Process()

```
float Decimator::Process (
    float input )
```

Applies downsample and bitcrush effects to input signal.

Returns

one sample. This should be called once per sample period.

4.13.2.5 SetBitcrushFactor()

```
void daisysp::Decimator::SetBitcrushFactor (
    float bitcrush_factor ) [inline]
```

Sets amount of bitcrushing Input range:

4.13.2.6 SetBitsToCrush()

```
void daisysp::Decimator::SetBitsToCrush (
    const uint8_t & bits ) [inline]
```

Sets the exact number of bits to crush 0-16 bits

4.13.2.7 SetDownsampleFactor()

```
void daisysp::Decimator::SetDownsampleFactor (
    float downsample_factor ) [inline]
```

Sets amount of downsample Input range:

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

- modules/decimator.h
- modules/decimator.cpp

4.14 daisysp::DelayLine< T, max_size > Class Template Reference

```
#include <delayline.h>
```

Public Member Functions

- void [Init](#) ()
- void [Reset](#) ()
- void [SetDelay](#) (size_t delay)
- void [SetDelay](#) (float delay)
- void [Write](#) (const T sample)
- const T [Read](#) () const

4.14.1 Detailed Description

```
template<typename T, size_t max_size>
class daisysp::DelayLine< T, max_size >
```

Simple Delay line. November 2019

Converted to Template December 2019

declaration example: (1 second of floats)

```
DelayLine<float, SAMPLE_RATE> del;
```

By: shensley

4.14.2 Member Function Documentation

4.14.2.1 Init()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Init ( ) [inline]
```

initializes the delay line by clearing the values within, and setting delay to 1 sample.

4.14.2.2 Read()

```
template<typename T , size_t max_size>
const T daisysp::DelayLine< T, max_size >::Read ( ) const [inline]
```

returns the next sample of type T in the delay line, interpolated if necessary.

4.14.2.3 Reset()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Reset ( ) [inline]
```

clears buffer, sets write ptr to 0, and delay to 1 sample.

4.14.2.4 SetDelay() [1/2]

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::SetDelay (
    float delay ) [inline]
```

sets the delay time in samples If a float is passed in, a fractional component will be calculated for interpolating the delay line.

4.14.2.5 SetDelay() [2/2]

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::SetDelay (
    size_t delay ) [inline]
```

sets the delay time in samples If a float is passed in, a fractional component will be calculated for interpolating the delay line.

4.14.2.6 Write()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Write (
    const T sample ) [inline]
```

writes the sample of type T to the delay line, and advances the write ptr

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

- modules/delayline.h

4.15 daisysp::Fold Class Reference

```
#include <fold.h>
```

Public Member Functions

- void [Init](#) ()
- float [Process](#) (float in)
- void [SetIncrement](#) (float incr)

4.15.1 Detailed Description

fold module

Original author(s) : John FFitch, Gabriel Maldonado

Year : 1998

Ported from soundpipe by Ben Sergentanis, May 2020

4.15.2 Member Function Documentation

4.15.2.1 Init()

```
void Fold::Init ( )
```

Initializes the fold module.

4.15.2.2 Process()

```
float Fold::Process (
    float in )
```

applies foldvoer distortion to input

4.15.2.3 SetIncrement()

```
void daisysp::Fold::SetIncrement (
    float incr ) [inline]
```

Parameters

<i>incr</i>	: set fold increment
-------------	----------------------

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

- modules/fold.h
- modules/fold.cpp

4.16 daisysp::Limiter Class Reference

```
#include <limiter.h>
```

Public Member Functions

- void [Init](#) ()
- void [ProcessBlock](#) (float *in, size_t size, float pre_gain)

4.16.1 Detailed Description

Simple Peak [Limiter](#)

This was extracted from pichenettes/stmlib.

Credit to pichenettes/Mutable Instruments

4.16.2 Member Function Documentation

4.16.2.1 Init()

```
void Limiter::Init ( )
```

Initializes the [Limiter](#) instance.

4.16.2.2 ProcessBlock()

```
void Limiter::ProcessBlock (
    float * in,
    size_t size,
    float pre_gain )
```

Processes a block of audio through the limiter.

Parameters

<i>in</i>	- pointer to a block of audio samples to be processed. The buffer is operated on directly.
<i>size</i>	- size of the buffer "in"
<i>pre_gain</i>	- amount of <i>pre_gain</i> applied to the signal.

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

- modules/limiter.h
- modules/limiter.cpp

4.17 daisysp::Line Class Reference

```
#include <line.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (uint8_t *finished)
- void [Start](#) (float start, float end, float dur)

4.17.1 Detailed Description

creates a [Line](#) segment signal

4.17.2 Member Function Documentation

4.17.2.1 Init()

```
void Line::Init (  
    float sample_rate )
```

Initializes [Line](#) module.

4.17.2.2 Process()

```
float Line::Process (  
    uint8_t * finished )
```

Processes [Line](#) segment. Returns one sample. value of finished will be updated to a 1, upon completion of the [Line](#)'s trajectory.

4.17.2.3 Start()

```
void Line::Start (  
    float start,  
    float end,  
    float dur )
```

Begin creation of [Line](#).

Parameters

<i>start</i>	- beginning value
<i>end</i>	- ending value
<i>dur</i>	- duration in seconds of Line segment

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

- modules/line.h
- modules/line.cpp

4.18 daisysp::Maytrig Class Reference

```
#include <maytrig.h>
```

Public Member Functions

- float [Process](#) (float prob)

4.18.1 Detailed Description

Probabilistic trigger module

Original author(s) : Paul Batchelor

Ported from soundpipe by Ben Sergentanis, May 2020

4.18.2 Member Function Documentation

4.18.2.1 Process()

```
float daisysp::Maytrig::Process (
    float prob ) [inline]
```

probabilistically generates triggers

Parameters

<i>prob</i>	(1 always returns true, 0 always false)
-------------	---

Returns

given a probability 0 to 1, returns true or false.

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

- modules/maytrig.h

4.19 daisysp::Metro Class Reference

```
#include <metro.h>
```

Public Member Functions

- void [Init](#) (float freq, float sample_rate)
- uint8_t [Process](#) ()
- void [Reset](#) ()
- void [SetFreq](#) (float freq)
- float [GetFreq](#) ()

4.19.1 Detailed Description

Creates a clock signal at a specific frequency.

4.19.2 Member Function Documentation

4.19.2.1 GetFreq()

```
float daisysp::Metro::GetFreq ( ) [inline]
```

Returns current value for frequency.

4.19.2.2 Init()

```
void Metro::Init (
    float freq,
    float sample_rate )
```

Initializes [Metro](#) module. Arguments:

- freq: frequency at which new clock signals will be generated Input Range:
- sample_rate: sample rate of audio engine Input range:

4.19.2.3 Process()

```
uint8_t Metro::Process ( )
```

checks current state of [Metro](#) object and updates state if necessary.

4.19.2.4 Reset()

```
void daisysp::Metro::Reset ( ) [inline]
```

resets phase to 0

4.19.2.5 SetFreq()

```
void Metro::SetFreq (
    float freq )
```

Sets frequency at which [Metro](#) module will run at.

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

- modules/metro.h
- modules/metro.cpp

4.20 daisysp::Mode Class Reference

```
#include <mode.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)
- void [Clear](#) ()
- void [SetFreq](#) (float freq)
- void [SetQ](#) (float q)

4.20.1 Detailed Description

Resonant Modal Filter

Extracted from soundpipe to work as a Daisy Module,
originally extracted from csound by Paul Batchelor.

Original Author(s): Francois Blanc, Steven Yi

Year: 2001

Location: Opcodes/biquad.c (csound)

4.20.2 Member Function Documentation

4.20.2.1 Clear()

```
void Mode::Clear ( )
```

Clears the filter, returning the output to 0.0

4.20.2.2 Init()

```
void Mode::Init (
    float sample_rate )
```

Initializes the instance of the module. sample_rate: frequency of the audio engine in Hz

4.20.2.3 Process()

```
float Mode::Process (
    float in )
```

Processes one input sample through the filter, and returns the output.

4.20.2.4 SetFreq()

```
void daisysp::Mode::SetFreq (
    float freq ) [inline]
```

Sets the resonant frequency of the modal filter. Range: Any frequency such that $\text{sample_rate} / \text{freq} < \text{PI}$ (about 15.2kHz at 48kHz)

4.20.2.5 SetQ()

```
void daisysp::Mode::SetQ (
    float q ) [inline]
```

Sets the quality factor of the filter. Range: Positive Numbers (Good values range from 70 to 1400)

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

- modules/mode.h
- modules/mode.cpp

4.21 daisysp::MoogLadder Class Reference

```
#include <moogladder.h>
```


Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float in)
- void [SetFreq](#) (float freq)
- void [SetRes](#) (float res)

4.21.1 Detailed Description

Moog ladder filter module

Ported from soundpipe

Original author(s) : Victor Lazzarini, John ffitch (fast tanh), Bob Moog

4.21.2 Member Function Documentation

4.21.2.1 Init()

```
void MoogLadder::Init (  
    float sample_rate )
```

Initializes the [MoogLadder](#) module. sample_rate - The sample rate of the audio engine being run.

4.21.2.2 Process()

```
float MoogLadder::Process (  
    float in )
```

Processes the lowpass filter

4.21.2.3 SetFreq()

```
void daisysp::MoogLadder::SetFreq (  
    float freq ) [inline]
```

Sets the cutoff frequency or half-way point of the filter. Arguments

- freq - frequency value in Hz. Range: Any positive value.

4.21.2.4 SetRes()

```
void daisysp::Moogladder::SetRes (
    float res ) [inline]
```

Sets the resonance of the filter.

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

- modules/moogladder.h
- modules/moogladder.cpp

4.22 daisysp::NlFilt Class Reference

```
#include <nlfilt.h>
```

Public Member Functions

- void [Init](#) ()
- void [ProcessBlock](#) (float *in, float *out, size_t size)
- void [SetCoefficients](#) (float a, float b, float d, float C, float L)
- void [SetA](#) (float a)
- void [SetB](#) (float b)
- void [SetD](#) (float d)
- void [SetC](#) (float C)
- void [SetL](#) (float L)

4.22.1 Detailed Description

Non-linear filter

port by: Stephen Hensley, December 2019

The four 5-coefficients: a, b, d, C, and L are used to configure different filter types.

Structure for Dobson/Fitch nonlinear filter

Revised Formula from Risto Holopainen 12 Mar 2004

$$Y\{n\} = \tanh(a \cdot Y\{n-1\} + b \cdot Y\{n-2\} + d \cdot Y^2\{n-L\} + X\{n\} - C)$$

Though traditional filter types can be made, the effect will always respond differently to different input.

This Source is a heavily modified version of the original source from Csound.

Todo make this work on a single sample instead of just on blocks at a time.

4.22.2 Member Function Documentation

4.22.2.1 Init()

```
void NlFilt::Init ( )
```

Initializes the [NlFilt](#) object.

4.22.2.2 ProcessBlock()

```
void NlFilt::ProcessBlock (
    float * in,
    float * out,
    size_t size )
```

Process the array pointed to by *in and updates the output to *out; This works on a block of audio at once, the size of which is set with the size.

4.22.2.3 SetA()

```
void daisysp::NlFilt::SetA (
    float a ) [inline]
```

Set Coefficient a

4.22.2.4 SetB()

```
void daisysp::NlFilt::SetB (
    float b ) [inline]
```

Set Coefficient b

4.22.2.5 SetC()

```
void daisysp::NlFilt::SetC (
    float C ) [inline]
```

Set Coefficient C

4.22.2.6 SetCoefficients()

```
void daisysp::NlFilt::SetCoefficients (
    float a,
    float b,
    float d,
    float C,
    float L ) [inline]
```

inputs these are the five coefficients for the filter.

4.22.2.7 SetD()

```
void daisysp::NlFilt::SetD (  
    float d ) [inline]
```

Set Coefficient d

4.22.2.8 SetL()

```
void daisysp::NlFilt::SetL (  
    float L ) [inline]
```

Set Coefficient L

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

- modules/nlfilt.h
- modules/nlfilt.cpp

4.23 daisysp::Oscillator Class Reference

```
#include <oscillator.h>
```

Public Types

- enum {
 WAVE_SIN, WAVE_TRI, WAVE_SAW, WAVE_RAMP,
 WAVE_SQUARE, WAVE_POLYBLEP_TRI, WAVE_POLYBLEP_SAW, WAVE_POLYBLEP_SQUARE,
 WAVE_LAST }

Public Member Functions

- void [Init](#) (float sample_rate)
- void [SetFreq](#) (const float f)
- void [SetAmp](#) (const float a)
- void [SetWaveform](#) (const uint8_t wf)
- float [Process](#) ()
- void [PhaseAdd](#) (float _phase)
- void [Reset](#) (float _phase=0.0f)

4.23.1 Detailed Description

Synthesis of several waveforms, including polyBLEP bandlimited waveforms.

4.23.2 Member Enumeration Documentation

4.23.2.1 anonymous enum

anonymous enum

Choices for output waveforms, POLYBLEP are appropriately labeled. Others are naive forms.

4.23.3 Member Function Documentation

4.23.3.1 Init()

```
void daisysp::Oscillator::Init (
    float sample_rate ) [inline]
```

Initializes the [Oscillator](#)

Parameters

<i>sample_rate</i>	- sample rate of the audio engine being run, and the frequency that the Process function will be called.
--------------------	--

Defaults:

- `freq_` = 100 Hz
- `amp_` = 0.5
- `waveform_` = sine wave.

4.23.3.2 PhaseAdd()

```
void daisysp::Oscillator::PhaseAdd (
    float _phase ) [inline]
```

Adds a value 0.0-1.0 (mapped to 0.0-TWO_PI) to the current phase. Useful for PM and "FM" synthesis.

4.23.3.3 Process()

```
float Oscillator::Process ( )
```

Processes the waveform to be generated, returning one sample. This should be called once per sample period.

4.23.3.4 Reset()

```
void daisysp::Oscillator::Reset (
    float _phase = 0.0f ) [inline]
```

Resets the phase to the input argument. If no argument is present, it will reset phase to 0.0;

4.23.3.5 SetAmp()

```
void daisysp::Oscillator::SetAmp (
    const float a ) [inline]
```

Sets the amplitude of the waveform.

4.23.3.6 SetFreq()

```
void daisysp::Oscillator::SetFreq (
    const float f ) [inline]
```

Changes the frequency of the [Oscillator](#), and recalculates phase increment.

4.23.3.7 SetWaveform()

```
void daisysp::Oscillator::SetWaveform (
    const uint8_t wf ) [inline]
```

Sets the waveform to be synthesized by the [Process\(\)](#) function.

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

- modules/oscillator.h
- modules/oscillator.cpp

4.24 daisysp::Phasor Class Reference

```
#include <phasor.h>
```

Public Member Functions

- void [Init](#) (float sample_rate, float freq, float initial_phase)
- void [Init](#) (float sample_rate, float freq)
- void [Init](#) (float sample_rate)
- float [Process](#) ()
- void [SetFreq](#) (float freq)
- float [GetFreq](#) ()

4.24.1 Detailed Description

Generates a normalized signal moving from 0-1 at the specified frequency.

Todo Selecting which channels should be initialized/included in the sequence conversion.

Setup a similar start function for an external mux, but that seems outside the scope of this file.

4.24.2 Member Function Documentation

4.24.2.1 GetFreq()

```
float daisysp::Phasor::GetFreq ( ) [inline]
```

Returns current frequency value in Hz

4.24.2.2 Init() [1/3]

```
void daisysp::Phasor::Init (
    float sample_rate ) [inline]
```

Initialize phasor with samplerate

4.24.2.3 Init() [2/3]

```
void daisysp::Phasor::Init (
    float sample_rate,
    float freq ) [inline]
```

Initialize phasor with samplerate and freq

4.24.2.4 Init() [3/3]

```
void daisysp::Phasor::Init (
    float sample_rate,
    float freq,
    float initial_phase ) [inline]
```

Initializes the [Phasor](#) module sample rate, and freq are in Hz initial phase is in radians Additional Init functions have defaults when arg is not specified:

- phs = 0.0f
- freq = 1.0f

4.24.2.5 Process()

```
float Phasor::Process ( )
```

processes [Phasor](#) and returns current value

4.24.2.6 SetFreq()

```
void Phasor::SetFreq (
    float freq )
```

Sets frequency of the [Phasor](#) in Hz

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

- modules/phasor.h
- modules/phasor.cpp

4.25 daisysp::PitchShifter Class Reference

```
#include <pitchshifter.h>
```

Public Member Functions

- void [Init](#) (float sr)
- float [Process](#) (float &in)
- void [SetTransposition](#) (const float &transpose)
- void [SetDelSize](#) (uint32_t size)
- void [SetFun](#) (float f)

4.25.1 Detailed Description

time-domain pitchshifter

Author: shensley

Based on "Pitch Shifting" from ucsd.edu

$$t = 1 - ((s * f) / R)$$

where: s is the size of the delay f is the frequency of the lfo r is the sample_rate

solving for t = 12.0 f = (12 - 1) * 48000 / SHIFT_BUFFER_SIZE;

Todo • move hash_xs32 and myrand to [dsp.h](#) and give appropriate names

4.25.2 Member Function Documentation

4.25.2.1 Init()

```
void daisysp::PitchShifter::Init (
    float sr ) [inline]
```

Initialize pitch shifter

4.25.2.2 Process()

```
float daisysp::PitchShifter::Process (
    float & in ) [inline]
```

process pitch shifter

4.25.2.3 SetDelSize()

```
void daisysp::PitchShifter::SetDelSize (
    uint32_t size ) [inline]
```

sets delay size changing the timbre of the pitchshifting

4.25.2.4 SetFun()

```
void daisysp::PitchShifter::SetFun (
    float f ) [inline]
```

sets an amount of internal random modulation, kind of sounds like tape-flutter

4.25.2.5 SetTransposition()

```
void daisysp::PitchShifter::SetTransposition (
    const float & transpose ) [inline]
```

sets transposition in semitones

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

- modules/pitchshifter.h

4.26 daisysp::Pluck Class Reference

```
#include <pluck.h>
```

Public Member Functions

- void [Init](#) (float sample_rate, float *buf, int32_t npt, int32_t mode)
- float [Process](#) (float &trig)
- void [SetAmp](#) (float amp)
- void [SetFreq](#) (float freq)
- void [SetDecay](#) (float decay)
- void [SetDamp](#) (float damp)
- void [SetMode](#) (int32_t mode)
- float [GetAmp](#) ()
- float [GetFreq](#) ()
- float [GetDecay](#) ()
- float [GetDamp](#) ()
- int32_t [GetMode](#) ()

4.26.1 Detailed Description

Produces a naturally decaying plucked string or drum sound based on the Karplus-Strong algorithms.

Ported from soundpipe to DaisySP

This code was originally extracted from the Csound opcode "pluck"

Original Author(s): Barry Vercoe, John fitch Year: 1991

Location: OOps/ugens4.c

4.26.2 Member Function Documentation

4.26.2.1 GetAmp()

```
float daisysp::Pluck::GetAmp ( ) [inline]
```

Returns the current value for amp.

4.26.2.2 GetDamp()

```
float daisysp::Pluck::GetDamp ( ) [inline]
```

Returns the current value for damp.

4.26.2.3 GetDecay()

```
float daisysp::Pluck::GetDecay ( ) [inline]
```

Returns the current value for decay.

4.26.2.4 GetFreq()

```
float daisysp::Pluck::GetFreq ( ) [inline]
```

Returns the current value for freq.

4.26.2.5 GetMode()

```
int32_t daisysp::Pluck::GetMode ( ) [inline]
```

Returns the current value for mode.

4.26.2.6 Init()

```
void Pluck::Init (
    float sample_rate,
    float * buf,
    int32_t npt,
    int32_t mode )
```

Initializes the [Pluck](#) module.

```
\param sample_rate: Sample rate of the audio engine being run.
\param buf: buffer used as an impulse when triggering the Pluck algorithm
\param npt: number of elements in buf.
\param mode: Sets the mode of the algorithm.
```

4.26.2.7 Process()

```
float Pluck::Process (
    float & trig )
```

Processes the waveform to be generated, returning one sample. This should be called once per sample period.

4.26.2.8 SetAmp()

```
void daisysp::Pluck::SetAmp (
    float amp ) [inline]
```

Sets the amplitude of the output signal. Input range: 0-1?

4.26.2.9 SetDamp()

```
void daisysp::Pluck::SetDamp (
    float damp ) [inline]
```

Sets the dampening factor applied by the filter (based on PLUCK_MODE) Input range: 0-1

4.26.2.10 SetDecay()

```
void daisysp::Pluck::SetDecay (
    float decay ) [inline]
```

Sets the time it takes for a triggered note to end in seconds. Input range: 0-1

4.26.2.11 SetFreq()

```
void daisysp::Pluck::SetFreq (
    float freq ) [inline]
```

Sets the frequency of the output signal in Hz. Input range: Any positive value

4.26.2.12 SetMode()

```
void daisysp::Pluck::SetMode (
    int32_t mode ) [inline]
```

Sets the mode of the algorithm.

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

- modules/pluck.h
- modules/pluck.cpp

4.27 daisysp::PolyPluck< num_voices > Class Template Reference

```
#include <PolyPluck.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float &trig, float note)
- void [SetDecay](#) (float p)

4.27.1 Detailed Description

```
template<size_t num_voices>
class daisysp::PolyPluck< num_voices >
```

Simplified Pseudo-Polyphonic [Pluck](#) Voice

Template Based [Pluck](#) Voice, with configurable number of voices and simple pseudo-polyphony.

DC Blocking included to prevent biases from causing unwanted saturation distortion.

Author**: shensley

Date Added**: March 2020

4.27.2 Member Function Documentation

4.27.2.1 Init()

```
template<size_t num_voices>
void daisysp::PolyPluck< num_voices >::Init (
    float sample_rate ) [inline]
```

Initializes the [PolyPluck](#) instance.

Parameters

<i>sample_rate</i>	rate in Hz that the Process() function will be called.
--------------------	--

4.27.2.2 Process()

```
template<size_t num_voices>
float daisysp::PolyPluck< num_voices >::Process (
    float & trig,
    float note ) [inline]
```

Process function, synthesizes and sums the output of all voices, triggering a new voice with frequency of MIDI note number when trig > 0.

Parameters

<i>trig</i>	value by reference of trig. When trig > 0 a the next voice will be triggered, and trig will be set to 0.
<i>note</i>	MIDI note number for the active_voice.

4.27.2.3 SetDecay()

```
template<size_t num_voices>
void daisysp::PolyPluck< num_voices >::SetDecay (
    float p ) [inline]
```

Sets the decay coefficients of the pluck voices.

Parameters

<i>p</i>	expects 0.0-1.0 input.
----------	------------------------

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

- modules/PolyPluck.h

4.28 daisysp::Port Class Reference

```
#include <port.h>
```

Public Member Functions

- void [Init](#) (float sample_rate, float htime)
- float [Process](#) (float in)
- void [SetHtime](#) (float htime)
- float [GetHtime](#) ()

4.28.1 Detailed Description

Applies portamento to an input signal.

At each new step value, the input is low-pass filtered to move towards that value at a rate determined by ihtim. ihtim is the half-time of the function (in seconds), during which the curve will traverse half the distance towards the new value, then half as much again, etc., theoretically never reaching its asymptote.

This code has been ported from Soundpipe to DaisySP by Paul Batchelor.

The Soundpipe module was extracted from the Csound opcode "portk".

Original Author(s): Robbin Whittle, John fitch

Year: 1995, 1998

Location: Opcodes/biquad.c

4.28.2 Member Function Documentation

4.28.2.1 GetHtime()

```
float daisysp::Port::GetHtime ( ) [inline]
```

returns current value of htime

4.28.2.2 Init()

```
void Port::Init (
    float sample_rate,
    float htime )
```

Initializes [Port](#) module

Parameters

<i>sample_rate</i>	sample rate of audio engine
<i>htime</i>	half-time of the function, in seconds.

4.28.2.3 Process()

```
float Port::Process (  
    float in )
```

Applies portamento to input signal and returns processed signal.

Returns

slewed output signal

4.28.2.4 SetHtime()

```
void daisysp::Port::SetHtime (  
    float htime ) [inline]
```

Sets htime

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

- modules/port.h
- modules/port.cpp

4.29 daisysp::ReverbSc Class Reference

```
#include <reverbsc.h>
```

Public Member Functions

- int [Init](#) (float sample_rate)
- int [Process](#) (const float &in1, const float &in2, float *out1, float *out2)
- void [SetFeedback](#) (const float &fb)
- void [SetLpFreq](#) (const float &freq)

4.29.1 Detailed Description

Stereo Reverb

Reverb SC: Ported from csound/soundpipe

Original author(s): Sean Costello, Istvan Varga

Year: 1999, 2005

Ported to soundpipe by: Paul Batchelor

Ported by: Stephen Hensley

4.29.2 Member Function Documentation

4.29.2.1 Init()

```
int ReverbSc::Init (
    float sample_rate )
```

Initializes the reverb module, and sets the `sample_rate` at which the `Process` function will be called. Returns 0 if all good, or 1 if it runs out of delay times exceed maximum allowed.

4.29.2.2 Process()

```
int ReverbSc::Process (
    const float & in1,
    const float & in2,
    float * out1,
    float * out2 )
```

Process the input through the reverb, and updates values of `out1`, and `out2` with the new processed signal.

4.29.2.3 SetFeedback()

```
void daisysp::ReverbSc::SetFeedback (
    const float & fb ) [inline]
```

controls the reverb time. reverb tail becomes infinite when set to 1.0

Parameters

<i>fb</i>	- sets reverb time. range: 0.0 to 1.0
-----------	---------------------------------------

4.29.2.4 SetLpFreq()

```
void daisysp::ReverbSc::SetLpFreq (
    const float & freq ) [inline]
```

controls the internal dampening filter's cutoff frequency.

Parameters

<i>freq</i>	- low pass frequency. range: 0.0 to sample_rate / 2
-------------	---

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

- modules/reverbSc.h
- modules/reverbSc.cpp

4.30 daisysp::ReverbScDI Struct Reference

```
#include <reverbSc.h>
```

Public Attributes

- int [write_pos](#)
- int [buffer_size](#)
- int [read_pos](#)
- int [read_pos_frac](#)
- int [read_pos_frac_inc](#)
- int [dummy](#)
- int [seed_val](#)
- int [rand_line_cnt](#)
- float [filter_state](#)
- float * [buf](#)

4.30.1 Detailed Description

Delay line for internal reverb use

4.30.2 Member Data Documentation

4.30.2.1 buf

```
float* daisysp::ReverbScDI::buf
```

buffer ptr

4.30.2.2 `buffer_size`

```
int daisysp::ReverbScDl::buffer_size
```

buffer size

4.30.2.3 `dummy`

```
int daisysp::ReverbScDl::dummy
```

dummy var

4.30.2.4 `filter_state`

```
float daisysp::ReverbScDl::filter_state
```

state of filter

4.30.2.5 `rand_line_cnt`

```
int daisysp::ReverbScDl::rand_line_cnt
```

number of random lines

4.30.2.6 `read_pos`

```
int daisysp::ReverbScDl::read_pos
```

read position

4.30.2.7 `read_pos_frac`

```
int daisysp::ReverbScDl::read_pos_frac
```

fractional component of read pos

4.30.2.8 `read_pos_frac_inc`

```
int daisysp::ReverbScDl::read_pos_frac_inc
```

increment for fractional

4.30.2.9 `seed_val`

```
int daisysp::ReverbScDl::seed_val
```

randseed

4.30.2.10 write_pos

```
int daisysp::ReverbScDl::write_pos
```

write position

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

- modules/reverbSc.h

4.31 daisysp::Svf Class Reference

```
#include <svf.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- void [Process](#) (float in)
- void [SetFreq](#) (float f)
- void [SetRes](#) (float r)
- void [SetDrive](#) (float d)
- float [Low](#) ()
- float [High](#) ()
- float [Band](#) ()
- float [Notch](#) ()
- float [Peak](#) ()

4.31.1 Detailed Description

Double Sampled, Stable State Variable Filter

Credit to Andrew Simper from musicdsp.org

This is his "State Variable Filter (Double Sampled, Stable)"

Additional thanks to Laurent de Soras for stability limit, and Stefan Diedrichsen for the correct notch output

Ported by: Stephen Hensley

4.31.2 Member Function Documentation

4.31.2.1 Band()

```
float daisysp::Svf::Band ( ) [inline]
```

bandpass output

Returns

band pass output of the filter

4.31.2.2 High()

```
float daisysp::Svf::High ( ) [inline]
```

highpass output

Returns

high pass output of the filter

4.31.2.3 Init()

```
void Svf::Init (
    float sample_rate )
```

Initializes the filter float sample_rate - sample rate of the audio engine being run, and the frequency that the Process function will be called.

4.31.2.4 Low()

```
float daisysp::Svf::Low ( ) [inline]
```

lowpass output

Returns

low pass output of the filter

4.31.2.5 Notch()

```
float daisysp::Svf::Notch ( ) [inline]
```

notchpass output

Returns

notch pass output of the filter

4.31.2.6 Peak()

```
float daisysp::Svf::Peak ( ) [inline]
```

peak output

Returns

peak output of the filter

4.31.2.7 Process()

```
void Svf::Process (
    float in )
```

Process the input signal, updating all of the outputs.

4.31.2.8 SetDrive()

```
void daisysp::Svf::SetDrive (
    float d ) [inline]
```

sets the drive of the filter affects the response of the resonance of the filter

4.31.2.9 SetFreq()

```
void Svf::SetFreq (
    float f )
```

sets the frequency of the cutoff frequency. f must be between 0.0 and sample_rate / 2

4.31.2.10 SetRes()

```
void Svf::SetRes (
    float r )
```

sets the resonance of the filter. Must be between 0.0 and 1.0 to ensure stability.

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

- modules/svf.h
- modules/svf.cpp

4.32 daisysp::Tone Class Reference

```
#include <tone.h>
```

Public Member Functions

- void [Init](#) (float sample_rate)
- float [Process](#) (float &in)
- void [SetFreq](#) (float &freq)
- float [GetFreq](#) ()

4.32.1 Detailed Description

A first-order recursive low-pass filter with variable frequency response.

4.32.2 Member Function Documentation

4.32.2.1 GetFreq()

```
float daisysp::Tone::GetFreq ( ) [inline]
```

Returns

the current value for the cutoff frequency or half-way point of the filter.

4.32.2.2 Init()

```
void Tone::Init (
    float sample_rate )
```

Initializes the [Tone](#) module. sample_rate - The sample rate of the audio engine being run.

4.32.2.3 Process()

```
float Tone::Process (
    float & in )
```

Processes one sample through the filter and returns one sample. in - input signal

4.32.2.4 SetFreq()

```
void daisysp::Tone::SetFreq (
    float & freq ) [inline]
```

Sets the cutoff frequency or half-way point of the filter.

Parameters

<i>freq</i>	- frequency value in Hz. Range: Any positive value.
-------------	---

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

- modules/tone.h
- modules/tone.cpp

4.33 daisysp::WhiteNoise Class Reference

```
#include <whitenoise.h>
```

Public Member Functions

- void [Init](#) ()
- void [SetAmp](#) (float a)
- float [Process](#) ()

4.33.1 Detailed Description

fast white noise generator

I think this came from musicdsp.org at some point

4.33.2 Member Function Documentation

4.33.2.1 Init()

```
void daisysp::WhiteNoise::Init ( ) [inline]
```

Initializes the [WhiteNoise](#) object

4.33.2.2 Process()

```
float daisysp::WhiteNoise::Process ( ) [inline]
```

returns a new sample of noise in the range of -amp_ to amp_

4.33.2.3 SetAmp()

```
void daisysp::WhiteNoise::SetAmp (
    float a ) [inline]
```

sets the amplitude of the noise output

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

- modules/whitenoise.h

Index

- Band
 - daisysp::Svf, [57](#)
- buf
 - daisysp::ReverbScDI, [55](#)
- buffer_size
 - daisysp::ReverbScDI, [55](#)
- Clear
 - daisysp::Mode, [38](#)
- daisysp::AdEnv, [7](#)
 - GetCurrentSegment, [7](#)
 - GetValue, [8](#)
 - Init, [8](#)
 - IsRunning, [8](#)
 - Process, [8](#)
 - SetCurve, [8](#)
 - SetMax, [9](#)
 - SetMin, [9](#)
 - SetTime, [9](#)
 - Trigger, [9](#)
- daisysp::Adsr, [9](#)
 - GetCurrentSegment, [10](#)
 - Init, [10](#)
 - IsRunning, [10](#)
 - Process, [11](#)
 - SetSustainLevel, [11](#)
 - SetTime, [11](#)
- daisysp::ATone, [12](#)
 - GetFreq, [12](#)
 - Init, [12](#)
 - Process, [13](#)
 - SetFreq, [13](#)
- daisysp::Autowah, [13](#)
 - Init, [14](#)
 - Process, [14](#)
 - SetDryWet, [14](#)
 - SetLevel, [15](#)
 - SetWah, [15](#)
- daisysp::Balance, [15](#)
 - Init, [16](#)
 - Process, [16](#)
 - SetCutoff, [16](#)
- daisysp::Biquad, [17](#)
 - Init, [17](#)
 - Process, [17](#)
 - SetCutoff, [18](#)
 - SetRes, [18](#)
- daisysp::Bitcrush, [18](#)
 - Init, [19](#)
 - Process, [19](#)
 - SetBitDepth, [19](#)
 - SetCrushRate, [20](#)
- daisysp::BIOsc, [20](#)
 - Init, [21](#)
 - Process, [21](#)
 - SetAmp, [21](#)
 - SetFreq, [21](#)
 - SetPw, [21](#)
 - SetWaveform, [22](#)
 - Waveforms, [21](#)
- daisysp::Comb, [22](#)
 - Init, [22](#)
 - Process, [23](#)
 - SetFreq, [23](#)
 - SetRevTime, [23](#)
- daisysp::Compressor, [23](#)
 - Init, [24](#)
 - Process, [24](#)
 - SetAttack, [25](#)
 - SetRatio, [25](#)
 - SetRelease, [25](#)
 - SetThreshold, [25](#)
- daisysp::CrossFade, [26](#)
 - GetCurve, [26](#)
 - GetPos, [26](#)
 - Init, [26](#)
 - Process, [27](#)
 - SetCurve, [27](#)
 - SetPos, [27](#)
- daisysp::DcBlock, [27](#)
 - Init, [28](#)
 - Process, [28](#)
- daisysp::Decimator, [28](#)
 - GetBitcrushFactor, [29](#)
 - GetDownsampleFactor, [29](#)
 - Init, [29](#)
 - Process, [29](#)
 - SetBitcrushFactor, [29](#)
 - SetBitsToCrush, [29](#)
 - SetDownsampleFactor, [29](#)
- daisysp::DelayLine< T, max_size >, [30](#)
 - Init, [30](#)
 - Read, [31](#)
 - Reset, [31](#)
 - SetDelay, [31](#)
 - Write, [31](#)
- daisysp::Fold, [32](#)
 - Init, [32](#)

- Process, 32
- SetIncrement, 32
- daisysp::Limiter, 33
 - Init, 33
 - ProcessBlock, 33
- daisysp::Line, 34
 - Init, 34
 - Process, 34
 - Start, 34
- daisysp::Maytrig, 35
 - Process, 35
- daisysp::Metro, 36
 - GetFreq, 36
 - Init, 36
 - Process, 36
 - Reset, 37
 - SetFreq, 37
- daisysp::Mode, 37
 - Clear, 38
 - Init, 38
 - Process, 38
 - SetFreq, 38
 - SetQ, 38
- daisysp::MoogLadder, 38
 - Init, 39
 - Process, 39
 - SetFreq, 39
 - SetRes, 39
- daisysp::NIFilt, 40
 - Init, 41
 - ProcessBlock, 41
 - SetA, 41
 - SetB, 41
 - SetC, 41
 - SetCoefficients, 41
 - SetD, 41
 - SetL, 42
- daisysp::Oscillator, 42
 - Init, 43
 - PhaseAdd, 43
 - Process, 43
 - Reset, 43
 - SetAmp, 44
 - SetFreq, 44
 - SetWaveform, 44
- daisysp::Phasor, 44
 - GetFreq, 45
 - Init, 45
 - Process, 45
 - SetFreq, 46
- daisysp::PitchShifter, 46
 - Init, 47
 - Process, 47
 - SetDelSize, 47
 - SetFun, 47
 - SetTransposition, 47
- daisysp::Pluck, 47
 - GetAmp, 48
 - GetDamp, 48
 - GetDecay, 48
 - GetFreq, 48
 - GetMode, 49
 - Init, 49
 - Process, 49
 - SetAmp, 49
 - SetDamp, 49
 - SetDecay, 49
 - SetFreq, 50
 - SetMode, 50
- daisysp::PolyPluck< num_voices >, 50
 - Init, 51
 - Process, 51
 - SetDecay, 51
- daisysp::Port, 52
 - GetHtime, 52
 - Init, 52
 - Process, 53
 - SetHtime, 53
- daisysp::ReverbSc, 53
 - Init, 54
 - Process, 54
 - SetFeedback, 54
 - SetLpFreq, 54
- daisysp::ReverbScDI, 55
 - buf, 55
 - buffer_size, 55
 - dummy, 56
 - filter_state, 56
 - rand_line_cnt, 56
 - read_pos, 56
 - read_pos_frac, 56
 - read_pos_frac_inc, 56
 - seed_val, 56
 - write_pos, 56
- daisysp::Svf, 57
 - Band, 57
 - High, 58
 - Init, 58
 - Low, 58
 - Notch, 58
 - Peak, 59
 - Process, 59
 - SetDrive, 59
 - SetFreq, 59
 - SetRes, 59
- daisysp::Tone, 60
 - GetFreq, 60
 - Init, 60
 - Process, 60
 - SetFreq, 61
- daisysp::WhiteNoise, 61
 - Init, 61
 - Process, 62
 - SetAmp, 62
- dummy
 - daisysp::ReverbScDI, 56

- filter_state
 - daisysp::ReverbScDI, 56
- GetAmp
 - daisysp::Pluck, 48
- GetBitcrushFactor
 - daisysp::Decimator, 29
- GetCurrentSegment
 - daisysp::AdEnv, 7
 - daisysp::Adsr, 10
- GetCurve
 - daisysp::CrossFade, 26
- GetDamp
 - daisysp::Pluck, 48
- GetDecay
 - daisysp::Pluck, 48
- GetDownsampleFactor
 - daisysp::Decimator, 29
- GetFreq
 - daisysp::ATone, 12
 - daisysp::Metro, 36
 - daisysp::Phasor, 45
 - daisysp::Pluck, 48
 - daisysp::Tone, 60
- GetHtime
 - daisysp::Port, 52
- GetMode
 - daisysp::Pluck, 49
- GetPos
 - daisysp::CrossFade, 26
- GetValue
 - daisysp::AdEnv, 8
- High
 - daisysp::Svf, 58
- Init
 - daisysp::AdEnv, 8
 - daisysp::Adsr, 10
 - daisysp::ATone, 12
 - daisysp::Autowah, 14
 - daisysp::Balance, 16
 - daisysp::Biquad, 17
 - daisysp::Bitcrush, 19
 - daisysp::BIOsc, 21
 - daisysp::Comb, 22
 - daisysp::Compressor, 24
 - daisysp::CrossFade, 26
 - daisysp::DcBlock, 28
 - daisysp::Decimator, 29
 - daisysp::DelayLine< T, max_size >, 30
 - daisysp::Fold, 32
 - daisysp::Limiter, 33
 - daisysp::Line, 34
 - daisysp::Metro, 36
 - daisysp::Mode, 38
 - daisysp::MoogLadder, 39
 - daisysp::NIFilt, 41
 - daisysp::Oscillator, 43
 - daisysp::Phasor, 45
 - daisysp::PitchShifter, 47
 - daisysp::Pluck, 49
 - daisysp::PolyPluck< num_voices >, 51
 - daisysp::Port, 52
 - daisysp::ReverbSc, 54
 - daisysp::Svf, 58
 - daisysp::Tone, 60
 - daisysp::WhiteNoise, 61
- IsRunning
 - daisysp::AdEnv, 8
 - daisysp::Adsr, 10
- Low
 - daisysp::Svf, 58
- Notch
 - daisysp::Svf, 58
- Peak
 - daisysp::Svf, 59
- PhaseAdd
 - daisysp::Oscillator, 43
- Process
 - daisysp::AdEnv, 8
 - daisysp::Adsr, 11
 - daisysp::ATone, 13
 - daisysp::Autowah, 14
 - daisysp::Balance, 16
 - daisysp::Biquad, 17
 - daisysp::Bitcrush, 19
 - daisysp::BIOsc, 21
 - daisysp::Comb, 23
 - daisysp::Compressor, 24
 - daisysp::CrossFade, 27
 - daisysp::DcBlock, 28
 - daisysp::Decimator, 29
 - daisysp::Fold, 32
 - daisysp::Line, 34
 - daisysp::Maytrig, 35
 - daisysp::Metro, 36
 - daisysp::Mode, 38
 - daisysp::MoogLadder, 39
 - daisysp::Oscillator, 43
 - daisysp::Phasor, 45
 - daisysp::PitchShifter, 47
 - daisysp::Pluck, 49
 - daisysp::PolyPluck< num_voices >, 51
 - daisysp::Port, 53
 - daisysp::ReverbSc, 54
 - daisysp::Svf, 59
 - daisysp::Tone, 60
 - daisysp::WhiteNoise, 62
- ProcessBlock
 - daisysp::Limiter, 33
 - daisysp::NIFilt, 41
- rand_line_cnt
 - daisysp::ReverbScDI, 56

Read
 daisysp::DelayLine< T, max_size >, 31
 read_pos
 daisysp::ReverbScDI, 56
 read_pos_frac
 daisysp::ReverbScDI, 56
 read_pos_frac_inc
 daisysp::ReverbScDI, 56
 Reset
 daisysp::DelayLine< T, max_size >, 31
 daisysp::Metro, 37
 daisysp::Oscillator, 43

 seed_val
 daisysp::ReverbScDI, 56
 SetA
 daisysp::NIFilt, 41
 SetAmp
 daisysp::BIOsc, 21
 daisysp::Oscillator, 44
 daisysp::Pluck, 49
 daisysp::WhiteNoise, 62
 SetAttack
 daisysp::Compressor, 25
 SetB
 daisysp::NIFilt, 41
 SetBitcrushFactor
 daisysp::Decimator, 29
 SetBitDepth
 daisysp::Bitcrush, 19
 SetBitsToCrush
 daisysp::Decimator, 29
 SetC
 daisysp::NIFilt, 41
 SetCoefficients
 daisysp::NIFilt, 41
 SetCrushRate
 daisysp::Bitcrush, 20
 SetCurve
 daisysp::AdEnv, 8
 daisysp::CrossFade, 27
 SetCutoff
 daisysp::Balance, 16
 daisysp::Biquad, 18
 SetD
 daisysp::NIFilt, 41
 SetDamp
 daisysp::Pluck, 49
 SetDecay
 daisysp::Pluck, 49
 daisysp::PolyPluck< num_voices >, 51
 SetDelay
 daisysp::DelayLine< T, max_size >, 31
 SetDelSize
 daisysp::PitchShifter, 47
 SetDownsampleFactor
 daisysp::Decimator, 29
 SetDrive
 daisysp::Svf, 59

 SetDryWet
 daisysp::Autowah, 14
 SetFeedback
 daisysp::ReverbSc, 54
 SetFreq
 daisysp::ATone, 13
 daisysp::BIOsc, 21
 daisysp::Comb, 23
 daisysp::Metro, 37
 daisysp::Mode, 38
 daisysp::MoogLadder, 39
 daisysp::Oscillator, 44
 daisysp::Phasor, 46
 daisysp::Pluck, 50
 daisysp::Svf, 59
 daisysp::Tone, 61
 SetFun
 daisysp::PitchShifter, 47
 SetHtime
 daisysp::Port, 53
 SetIncrement
 daisysp::Fold, 32
 SetL
 daisysp::NIFilt, 42
 SetLevel
 daisysp::Autowah, 15
 SetLpFreq
 daisysp::ReverbSc, 54
 SetMax
 daisysp::AdEnv, 9
 SetMin
 daisysp::AdEnv, 9
 SetMode
 daisysp::Pluck, 50
 SetPos
 daisysp::CrossFade, 27
 SetPw
 daisysp::BIOsc, 21
 SetQ
 daisysp::Mode, 38
 SetRatio
 daisysp::Compressor, 25
 SetRelease
 daisysp::Compressor, 25
 SetRes
 daisysp::Biquad, 18
 daisysp::MoogLadder, 39
 daisysp::Svf, 59
 SetRevTime
 daisysp::Comb, 23
 SetSustainLevel
 daisysp::Adsr, 11
 SetThreshold
 daisysp::Compressor, 25
 SetTime
 daisysp::AdEnv, 9
 daisysp::Adsr, 11
 SetTransposition

- daisysp::PitchShifter, [47](#)
- SetWah
 - daisysp::Autowah, [15](#)
- SetWaveform
 - daisysp::BIOsc, [22](#)
 - daisysp::Oscillator, [44](#)
- Start
 - daisysp::Line, [34](#)
- Trigger
 - daisysp::AdEnv, [9](#)
- Waveforms
 - daisysp::BIOsc, [21](#)
- Write
 - daisysp::DelayLine< T, max_size >, [31](#)
- write_pos
 - daisysp::ReverbScDI, [56](#)