SoundFX Plugin 1.0

The Sound Effects Processor For Hollywood

Andreas Falkenhahn

Table of Contents

1 General information ... 1

1.1 Introduction ... 1
1.2 Terms and conditions ... 3
1.3 Requirements ... 4
1.4 Installation ... 4

2 About soundfx.hwp ... 5

2.1 Credits ... 5
2.2 Frequently asked questions ... 5
2.3 Known issues ... 5
2.4 Future ... 5
2.5 History ... 5

3 Usage ... 7

3.1 Using the plugin ... 7
3.2 Effect parameters ... 8
3.3 Clipping ... 9
3.4 Dithering ... 9

4 Function reference ... 11

4.1 soundfx.Generate ... 11
4.2 soundfx.Process ... 11

5 Analysis effects ... 17

5.1 Noiseprof ... 17
5.2 Stat ... 17
5.3 Stats ... 18

6 Editing effects ... 21

6.1 Pad ... 21
6.2 Silence ... 21
6.3 Splice ... 22
6.4 Trim ... 23
6.5 Vad ... 24

7 Low-level signal processing effects ... 27

7.1 Biquad ... 27
7.2 Downsample ... 27
7.3 Fir ... 27
7.4 Upsample ... 28

8 Mastering effects 29
8.1 Dither 29
8.2 Rate 29

9 Miscellaneous effects 33
9.1 Synth 33

10 Mixing effects 35
10.1 Channels 35
10.2 Remix 35
10.3 Swap 36

11 Pitch/tempo effects 37
11.1 Bend 37
11.2 Pitch 37
11.3 Speed 37
11.4 Stretch 38
11.5 Tempo 38

12 Production effects 41
12.1 Chorus 41
12.2 Delay 41
12.3 Echo 42
12.4 Echos 42
12.5 Flanger 43
12.6 Overdrive 43
12.7 Phaser 43
12.8 Repeat 44
12.9 Reverb 44
12.10 Reverse 45
12.11 Tremolo 45

13 Specialised filters/mixers 47
13.1 Deemph 47
13.2 Earwax 47
13.3 Noisered 47
13.4 Oops 48
13.5 Riaa 48

14 Tone/filter effects 49
14.1 Allpass 49
14.2 Band 49
14.3 Bandpass 49
14.4 Bandreject 50
14.5 Bass 50
14.6 Equalizer 50
14.7 Highpass 51
14.8 Hilbert 51
14.9 Lowpass 51
14.10 Sinc 52
14.11 Treble 52

15 Volume/level effects 55
15.1 Compand 55
15.2 Contrast 56
15.3 DCShift 56
15.4 Fade 57
15.5 Gain 57
15.6 Loudness 59
15.7 Mcompand 59
15.8 Norm 59
15.9 Vol 60

Appendix A Licenses 61
A.1 LGPL license 61

Index 69

1 General information

1.1 Introduction

SoundFX is the ultimate sound effects processor plugin for Hollywood. It supports almost 60 different effects and transformations that can easily be applied to Hollywood samples. On top of that, it can also analyze audio data and generate new sounds by synthesizing tones from given text parameters. Finally, the plugin can also be used to join, mix or merge multiple samples into new ones or change the audio format of samples.

The SoundFX plugin features a convenient interface which allows you to easily apply all kinds of effects to existing Hollywood samples using just a single function call. It's also possible to monitor the status of the sound processor by specifying a callback function that will be called from time to time to give your script a status update. The callback function can also be used to log messages from the sound processor.

The following sound effects are currently supported:

Analysis effects

- noiseprof: Produce a DFT profile of the audio (use with noisered)
- stat: Enumerate audio peak & RMS levels, approx. freq., etc.
- stats: Multichannel aware stat

Editing effects

- pad: Pad (usually) the ends of the audio with silence
- silence: Remove portions of silence from the audio
- splice: Perform the equivalent of a cross-faded tape splice
- trim: Cuts portions out of the audio
- vad: Voice activity detector

Low-level signal processing effects

- biquad: 2nd-order IIR filter using externally provided coefficients
- downsample: Reduce sample rate by discarding samples
- fir: FFT convolution FIR filter using externally provided coefficients
- upsample: Increase sample rate by zero stuffing

Mastering effects

- dither: Add dither noise to increase quantisation SNR
- rate: Change audio sampling rate

Miscellaneous effects

- synth: Synthesise/modulate audio tones or noise signals

Mixing effects

- channels: Auto mix or duplicate to change number of channels
- divide+: Divide sample values by those in the 1st channel (W.I.P.)
- remix: Produce arbitrarily mixed output channels
- swap: Swap pairs of channels

Pitch/tempo effects

- bend: Bend pitch at given times without changing tempo
- pitch: Adjust pitch (= key) without changing tempo
- speed: Adjust pitch & tempo together
- stretch: Adjust tempo without changing pitch (simple alg.)
- tempo: Adjust tempo without changing pitch (WSOLA alg.)

Production effects

- chorus: Make a single instrument sound like many
- delay: Delay one or more channels
- echo: Add an echo
- echos: Add a sequence of echos
- flanger: Stereo flanger
- overdrive: Non-linear distortion
- phaser: Phase shifter
- repeat: Loop the audio a number of times
- reverb: Add reverberation
- reverse: Reverse the audio (to search for Satanic messages ;-)
- tremolo: Sinusoidal volume modulation

Specialised filters/mixers

- deemph: ISO 908 CD de-emphasis (shelving) IIR filter
- earwax: Process CD audio to best effect for headphone use
- noisered: Filter out noise from the audio
- oops: Out Of Phase Stereo (or Karaoke) effect
- riaa: RIAA vinyl playback equalisation

Tone/filter effects

- allpass: RBJ all-pass biquad IIR filter
- bandpass: RBJ band-pass biquad IIR filter
- bandreject: RBJ band-reject biquad IIR filter
- band: SPKit resonator band-pass IIR filter
- bass: Tone control: RBJ shelving biquad IIR filter
- equalizer: RBJ peaking equalisation biquad IIR filter

- firfit+: FFT convolution FIR filter using given freq. response (W.I.P.)
- highpass: High-pass filter: Single pole or RBJ biquad IIR
- hilbert: Hilbert transform filter (90 degrees phase shift)
- lowpass: Low-pass filter: single pole or RBJ biquad IIR
- sinc: Sinc-windowed low/high-pass/band-pass/reject FIR
- treble: Tone control: RBJ shelving biquad IIR filter

Volume/level effects

- compand: Signal level compression/expansion/limiting
- contrast: Phase contrast volume enhancement
- dcshift: Apply or remove DC offset
- fade: Apply a fade-in and/or fade-out to the audio
- gain: Apply gain or attenuation; normalise/equalise/balance/headroom
- loudness: Gain control with ISO 226 loudness compensation
- mcompand: Multi-band compression/expansion/limiting
- norm: Normalise to 0dB (or other)
- vol: Adjust audio volume

1.2 Terms and conditions

PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.3 Requirements

- Hollywood 11.0 or better

1.4 Installation

Installing the SoundFX plugin is straightforward and simple: Just copy the file `soundfx.hwp` for the platform of your choice to Hollywood's plugins directory. The location where Hollywood plugins must be installed differs from platform to platform. Here is a breakdown of the different search paths that Hollywood will scan for plugins:

- AmigaOS and compatibles: Plugins can either be stored in a directory named `Plugins` that is in the same directory as the Hollywood interpreter or they can be installed globally in `LIBS:Hollywood`. Hollywood will first look for plugins in the `Plugins` directory in its program directory, then it will scan `LIBS:Hollywood`.
- Android: The Android version of Hollywood also supports Hollywood plugins. You have to copy them to the directory `Hollywood/Plugins` on your SD card. Hollywood will scan this location on every startup and load all plugins from there.
- Linux: Plugins must be stored in a directory named `Plugins` (case sensitive!) that is in the same directory as the Hollywood interpreter.
- macOS: `~/Library/Application Support/AirsoftSoftwair/Hollywood/Plugins` is the directory that should be used for storing plugins. If it doesn't exist, just create it. Alternatively, you can also store plugins inside Hollywood's app bundle in `Hollywood.app/Contents/Resources/Plugins` but this isn't recommended because it requires you to modify the Hollywood distribution and it might also require administrator privileges.
- Windows: `C:\Users\...\AppData\Local\AirsoftSoftwair\Hollywood\Plugins` is the directory that should be used for storing plugins. If it doesn't exist, just create it. Alternatively, you can also store plugins in the `Plugins` directory in Hollywood's installation directory, e.g. in `C:\Program Files\Hollywood\Plugins` but this isn't recommended because it requires you to modify the Hollywood distribution and it might also require administrator privileges to write to Hollywood's installation directory.

On Windows you should also copy the file `SoundFX.chm` to the `Docs` directory which you can find in the same directory as the `Plugins` directory (see above). After you have copied `SoundFX.chm` to the `Docs` directory, you will be able to get online help by pressing F1 when the cursor is over a soundfx.hwp function in the Hollywood IDE.

On macOS copy the `SoundFX` directory that is inside the `Docs` directory of soundfx.hwp's distribution archive to the `Docs` directory which you can find in the same directory as the `Plugins` directory (see above).

On Linux copy the `SoundFX` directory that is inside the `Docs` directory of soundfx.hwp's distribution archive to the `Docs` directory of your Hollywood installation.

2 About soundfx.hwp

2.1 Credits

soundfx.hwp was written by Andreas Falkenhahn. Work on this project was started during the development of Hollywood 11. The plugin was initially designed as a proof-of-concept for the new sound plugin interfaces introduced with Hollywood 11.

If you want to contact me, you can either send an e-mail to `andreas@airsoftsoftwair.de` or use the contact form on `http://www.hollywood-mal.com`.

2.2 Frequently asked questions

This section covers some frequently asked questions. Please read them first before asking on the forum because your problem might have been covered here.

Q: Is there a Hollywood forum where I can get in touch with other users?

A: Yes, please check out the "Community" section of the official Hollywood Portal online at `http://www.hollywood-mal.com`.

Q: Where can I ask for help?

A: There's an active forum at `http://forums.hollywood-mal.com`. You're welcome to join it and ask your question there.

Q: I have found a bug.

A: Please post about it in the "Bugs" section of the forum.

2.3 Known issues

Here is a list of things that soundfx.hwp doesn't support yet or that may be confusing in some way:

- tbd

2.4 Future

Here are some things that are on my to do list:

- add support for Hollywood music objects so that it's possible to apply effects directly to Hollywood music objects

Don't hesitate to contact me if soundfx.hwp lacks a certain feature that is important for your project.

2.5 History

Please see the file `history.txt` for a complete change log of soundfx.hwp.

3 Usage

3.1 Using the plugin

Using the plugin is really simple since there are only two functions that can do all the magic: soundfx.Process() and soundfx.Generate(). soundfx.Process() can be used to apply one or multiple sound effects to a Hollywood sample and soundfx.Generate() can be used to synthesize new audio tones from a given description.

soundfx.Process() basically takes a number of effects in a table and applies them to the Hollywood sample(s) passed to it. Note that it will never modify existing samples but always create new samples that contain the desired modifications. For example, here's how to reduce the volume of sample 1 by 50% and store the result as sample 2:

```
soundfx.Process(1, 2, {{"vol", "0.5"}})
```

It's also possible to pass multiple samples to soundfx.Process(). Here's how to reduce the volume of samples 1 and 2 by 50% and store the result as sample 3:

```
soundfx.Process({{ID=1},{ID=2}}, 3, {{"vol", "0.5"}})
```

The call above will also join samples 1 and 2. If you want to mix samples 1 and 2 instead of joining them, you need to set the `Mode` parameter to `#SOUNDFX_MIX` like this:

```
soundfx.Process({{ID=1},{ID=2}},3,{{"vol","0.5"},Mode = #SOUNDFX_MIX})
```

When passing multiple samples to soundfx.Process() it's also possible to adjust the volume of the individual samples. Here's a call that mixes samples 1 and 2 into a new sample 3. The volume of sample 1 will be reduced to just 20% of the original volume:

```
soundfx.Process({{ID=1, Volume=0.2}, {ID=2}}, 3, Mode = #SOUNDFX_MIX})
```

You can also apply multiple effects to samples. Here's how to apply the flanger and norm effects to sample 1 and store the result in sample 2:

```
soundfx.Process(1, 2, {{"flanger"}, {"norm"}})
```

You can also use soundfx.Process() to convert samples into new audio formats. For example, if sample 1 is in 8-bit mono format you can use this call to convert it to 16-bit stereo and store the result as sample 2:

```
soundfx.Process(1, 2, {Bits = 16, Channels = 2})
```

If you want to synthesize new sound tones, use the soundfx.Generate() function. For example, this call creates a synthesised A minor seventh chord with a pipe-organ sound and stores it as sample 1:

```
soundfx.Generate(1, {{"synth", "sin", "%-12", "sin", "%-9","sin","%-5",
          "sin","%-2"},{"fade", "h", "0.1", "1", "0.1"}, Channels = 1})
```

The code above uses the synth and fade effects to generate and process the audio data. As you can see, different parameters are passed to each effect. See the next chapter to learn how to pass parameters to effects.

3.2 Effect parameters

Most effects supported by the SoundFX plugin accept certain parameters. Those parameters are described in the synopsis section in the documentation of the individual sound effects. For example, the synopsis for the fade effect looks like this:

```
fade [type] fade-in-length [stop-position(=) [fade-out-length]]
```

In such an effect synopsis, brackets [ ] are used to denote parameters that are optional, braces { } to denote those that are both optional and repeatable, and angle brackets < > to denote those that are repeatable but not optional. Where applicable, default values for optional parameters are shown in parenthesis ( ).

The following parameters are used with, and have the same meaning for, several effects:

`center[k]`
: See frequency.

`frequency[k]`
: A frequency in Hz, or, if appended with "k", kHz.

`gain`
: A power gain in dB. Zero gives no gain; less than zero gives an attenuation.

`position`
: A position within the audio stream; the syntax is `[=|+|-]timespec`, where `timespec` is a time specification (see below). The optional first character indicates whether the `timespec` is to be interpreted relative to the start (=) or end (-) of audio, or to the previous position if the effect accepts multiple position arguments (+). The audio length must be known for end-relative locations to work; some effects do accept -0 for end-of-audio, though, even if the length is unknown. Which of =, +, - is the default depends on the effect and is shown in its syntax as, e.g., position(+).

  Examples: =2:00 (two minutes into the audio stream), -100s (one hundred samples before the end of audio), +0:12+10s (twelve seconds and ten samples after the previous position), -0.5+1s (one sample less than half a second before the end of audio).

`width[h|k|o|q]`
: Used to specify the band-width of a filter. A number of different methods to specify the width are available (though not all for every effect). One of the characters shown may be appended to select the desired method as follows: "h" for Hz, "k" for kHz, "o" for Octaves and "q" for Q-factor.

  For each effect that uses this parameter, the default method (i.e. if no character is appended) is the one that it listed first in the first line of the effect's description.

Most effects that expect an audio position or duration in a parameter, i.e. a time specification, accept either of the following two forms:

`[[hours:]minutes:]seconds[.frac][t]`
: A specification of "1:30.5" corresponds to one minute, thirty and a half seconds. The `t` suffix is entirely optional (however, see the silence effect for an exception). Note that the component values do not have to be normalized; e.g., "1:23:45", "83:45", "79:0285", "1:0:1425", "1::1425" and "5025" all are legal and equivalent to each other.

`sampless` Specifies the number of samples directly, as in `"8000s"`. For large sample counts, e notation is supported: `"1.7e6s"` is the same as `"1700000s"`.

Time specifications can also be chained with `+` or `-` into a new time specification where the right part is added to or subtracted from the left, respectively: `"3:00-200s"` means two hundred samples less than three minutes.

3.3 Clipping

Clipping is distortion that occurs when an audio signal level (or "volume") exceeds the range of the chosen representation. In most cases, clipping is undesirable and so should be corrected by adjusting the level prior to the point (in the processing chain) at which it occurs.

In a sound processor, clipping could occur, as you might expect, when using the vol or gain effects to increase the audio volume. Clipping could also occur with many other effects.

For these reasons, it is usual to make sure that a sample's signal level has some "headroom", i.e. it does not exceed a particular level below the maximum possible level for the given representation. Some standards bodies recommend as much as 9dB headroom, but in most cases, 3dB (around 70% linear) is enough. Note that this wisdom seems to have been lost in modern music production; in fact, many CDs, MP3s, etc. are now mastered at levels above 0dBFS i.e. the audio is clipped as delivered.

The stat and stats effects can assist in determining the signal level in a Hollywood sample. The gain or vol effect can be used to prevent clipping, e.g.

```
gain -6 treble +6
```

guarantees that the treble boost will not clip.

If clipping occurs at any point during processing, the plugin will display a warning message to that effect.

3.4 Dithering

Dithering is a technique used to maximise the dynamic range of audio stored at a particular bit-depth. Any distortion introduced by quantisation is decorrelated by adding a small amount of white noise to the signal.

You might want to add TPDF dither if any of the following are true:

- bit-depth reduction has been specified explicitly using the `Bits` option
- an effect has increased effective bit-depth within the internal processing chain

For example, adjusting volume with `"vol 0.25"` requires two additional bits in which to losslessly store its results (since 0.25 decimal equals 0.01 binary). So if the input sample bit-depth is 16, then the plugin's internal representation will utilise 18 bits after processing this volume change. In order to store the output at the same depth as the input, dithering can be used to remove the additional bits.

To apply dithering to the sound processing chain, just add the dither effect as the last effect.

4 Function reference

4.1 soundfx.Generate

NAME

soundfx.Generate – generate sample and apply effect(s)

SYNOPSIS

```
[id] = soundfx.Generate(dst, fx)
```

FUNCTION

This function generates a sample and applies the given effect(s) to it. It will create a new sample using the identifier specified by `dst`. If you pass `Nil` in `dst`, `soundfx.Generate()` will automatically choose an available identifier and return it to you.

Usage is very similar to soundfx.Process() except that `soundfx.Generate()` doesn't require you to pass input samples. Instead, the source sample is generated by this function. This means that the first effect in the `fx` table should be a generative effect like the synth effect. After that effect, you could then add some more effects which process the audio data a generative effect like synth has just produced. See below for an example.

Note that if you don't use a generative effect as the first effect in the `fx` table, this function will run forever because the following effects will wait for audio data which isn't there if you don't use a generative effect first.

You might also want to set the `Channels`, `Bits`, and `Frequency` tags when calling soundfx.Generate() to set the desired format of the sample to be generated.

See the documentation of soundfx.Process() for a detailed description of how to pass effects and options in the `fx` table.

INPUTS

`dst`	identifier for the new sample or `Nil` for auto id selection
`fx`	table containing effect(s) and options to be applied

RESULTS

`id`	optional: identifier of the sample; will only be returned when you pass `Nil` as argument 1 (see above)

EXAMPLE

```
soundfx.Generate(1, {{"synth", "sin", "%-12", "sin", "%-9", "sin", "%-5",
          "sin", "%-2"}, {"fade", "h", "0.1", "1", "0.1"}, Channels = 1})
```

The code above creates a synthesised A minor seventh chord with a pipe-organ sound and stores it as sample 1.

4.2 soundfx.Process

NAME

soundfx.Process – apply effect(s) to sample(s)

SYNOPSIS

```
[id] = soundfx.Process(src, dst[, fx])
```

FUNCTION

This function can be used to apply one or more effects to the sample(s) specified in `src`. The source sample(s) won't be modified directly. Instead, this function will create a new sample using the identifier specified by `dst`. If you pass `Nil` in `dst`, `soundfx.Process()` will automatically choose an available identifier and return it to you. The source and destination samples mustn't be the same.

The `src` argument can either be the identifier of a sample or it can be a table containing a number of samples. If `src` is a table, it must contain a subtable for each sample that you want to pass to `soundfx.Process()`. The subtable can contain the following fields:

`ID` Specifies the ID of the sample. This table element must always be set.

`Volume` Specifies the desired volume of the sample. This can be used to modify the volume before any effects are applied. Values smaller than 1.0 will decrease the volume, whereas values higher than 1.0 will increase the volume. This can be particularly useful when using the `#SOUNDFX_MIX` mode. In that case, it allows you to fine-tune the volume of the individual samples to be mixed together. This tag defaults to 1.0 which means that no changes to the volume will be made.

When passing multiple samples to `soundfx.Process()`, you can use the `Mode` table tag to specify how the samples should be processed. By default, `#MODE_JOIN` is used which means that the samples will be processed one after the other so that once `soundfx.Process()` has finished, all samples will have been joined together. Alternatively, you can also use `#MODE_MIX` which means that the samples will be mixed together. Finally, there's also `#MODE_MERGE` and `#MODE_MULTIPLY`. See below for details.

Note that when passing multiple samples to `soundfx.Process()`, all samples must use the same sampling rate. When using `#MODE_JOIN` (the default), the samples must also use the same channel layout.

The effect(s) that should be applied to the sample must be specified in the `fx` table argument. This must be set to a table of tables, i.e. a table containing an arbitrary number of subtables, each containing a single effect specification that should be applied to the sample. The effects are applied to the sample in the order they appear in the `fx`.

The subtables containing the effect specification must start with the name of the effect followed by the individual effect parameters. These differ from effect to effect. Please see the individual effect descriptions in the following chapters to find out about mandatory and optional effect parameters. Also see the chapter on effect parameters to learn about how the effect syntax is constructed. Also note that some effects like synth are generative which means that they produce their own audio instead of processing given audio data. To use such generative effects you need to use soundfx.Generate() instead of `soundfx.Process()`.

Additionally, the `fx` table also allows you to specify some other options that shall be passed to the sound processor. The following options are currently recognized:

`Mode` This specifies how the sound processor should behave in case multiple samples are passed to it. This can be set to the following constants:

`#SOUNDFX_JOIN`
: This mode will simply concatenate the samples. The frequency and channel layout of the samples to be joined must be the same. This is the default.

`#SOUNDFX_MIX`
: This mode will mix the samples. The frequency of the samples to be mixed must be the same. The channel layout of the samples can be different but in that case some channels in the output sample will not contain audio from every input sample.

`#SOUNDFX_MERGE`
: This mode will merge the samples. The frequency of the samples to be merged must be the same. The channel layout of the samples can be different. A merged sample comprises all of the channels from all of the input samples, e.g. it is possible to merge two mono samples into a stereo one. The first and second mono sample would become the left and right channels of the stereo sample.

`#SOUNDFX_MULTIPLY`
: This mode multiplies the sample values. The frequency of the samples to be multiplied must be the same. The sample values of corresponding channels will be treated as numbers in the interval -1 to +1. If the number of channels in the input samples is not the same, the missing channels are considered to contain all zero.

The default mode is `#SOUNDFX_JOIN`.

`Frequency`
: The new frequency for the sample in Hertz. It's only necessary to set this if the effect alters the sample resolution, e.g. by resampling it, or if you're using soundfx.Generate() and you want to modify the generator effect's default frequency.

`Bits`
: The new bit depth for the sample. Currently only 8 or 16 are supported here. It's only necessary to specify this if you want to change the bit depth of the sample or if you're using soundfx.Generate() and you want to modify the generator effect's default bit depth setting.

`Channels`
: The new number of channels used by the sample. Currently only 1 (mono) or 2 (stereo) are supported here. It's only necessary to specify this if you want to change the channel layout of the sample or if you're using soundfx.Generate() and you want to modify the generator effect's default channel layout setting.

`Verbosity`
: This tag can be used to set how much information the sound processor should output. This defaults to 0 which means that no messages are shown at all. The following verbosity levels are supported:

  `0` No messages are shown at all. This is the default.

`1` Only error messages are shown. These are generated if the plugin cannot apply the requested effects.

`2` Warning messages are also shown. These are generated if the plugin can complete the requested commands, but not exactly according to the requested command parameters, or if clipping occurs.

`3` Descriptions of the processing phases are also shown. Useful for seeing exactly how the plugin is processing your audio.

`4 and above`
Messages to help with debugging are also shown.

By default, messages are printed to the standard output but you can also intercept them by installing a callback function using the `Callback` tag (see below).

`Callback` If you set this tag to a Hollywood function, the SoundFX plugin will call the function periodically so that you can update a status bar or something similar. The callback function will also be called whenever the sound processor has to output a message. When the SoundFX plugin runs your callback, it will pass a table that has the following fields initialized to it:

`Action` This will be set to either `#SOUNDFX_STATUS` if the callback has been run to inform you about a status update or it can be `#SOUNDFX_OUTPUT` if the callback has been run because a message should be output.

`Processed`
This contains the number of PCM frames already processed. This is only set if `Action` is `#SOUNDFX_STATUS`.

`Total` This contains the total number of PCM frames to be processed. This is only set if `Action` is `#SOUNDFX_STATUS`.

`Output` This contains the message string that should be output. This is only set if `Action` is `#SOUNDFX_OUTPUT`.

`UserData` If you use the `UserData` tag (see below) to specify data that should be passed to the callback, this field will be initialized to contain this user data. Otherwise this field won't be set.

`UserData` If you have set the `Callback` tag to a callback function, you can use this tag to specify user data that should be passed to your callback function whenever it is run. The user data you specify here will be passed in the `UserData` element of the table argument that is passed to the callback function.

`Software` If this tag is set to `True`, the destination sample will be created as a software sample. See the Hollywood manual for more information on the difference between software and hardware samples. Defaults to `False`.

`UseTempFile`
Some effects can use a temporary file instead of memory buffers. If you want those sound effects to use a temporary file, set this tag to `True`. Otherwise

everything will be done in memory. Setting this to `True` can be useful when dealing with large samples on low-memory systems. Defaults to `False`.

Note that the `fx` table can also be omitted. In that case, no effects are applied and the source sample is simply copied to the destination one, or, if multiple samples are specified, the source samples are concatenated to form the destination sample.

INPUTS

`src` identifier of the source sample or table containing multiple samples (see above)

`dst` identifier for the new sample or `Nil` for auto id selection

`fx` optional: table containing effect(s) and options to be applied

RESULTS

`id` optional: identifier of the sample; will only be returned when you pass `Nil` as argument 2 (see above)

EXAMPLE

```
soundfx.Process(1, 2, {{"tempo", "0.8"}})
```

The code above reduces the tempo of sample 1 to 80% of its original playback speed and stores the result in sample 2.

```
soundfx.Process(1, 2, {{"phaser"}, {"vol", "0.5"}})
```

The code above first applies the phaser effect to sample 1 and then reduces the volume to 50%. The result will be stored in sample 2.

```
soundfx.Process(1, 2, {{"rate", "48k"}}, {Frequency = 48000})
```

The code above resamples the audio data of sample 1 to 48000hz and stores the result in sample 2. Note that we must pass the new frequency in the table which is passed as the 4th parameter. Passing the 4th parameter is only necessary for effects which modify the sample resolution (i.e. bit depth, frequency, channel layout). See above for more information.

```
soundfx.Process({{ID = 1}, {ID = 2}}, 3)
```

The code above will concatenate samples 1 and 2 and store the result in sample 3.

```
soundfx.Process({{ID = 1, Volume = 0.5}, {ID = 2}}, 3, {Mode =
                #SOUNDFX_MIX})
```

The code above will mix samples 1 and 2 and store the result in sample 3. The volume of sample 1 will be reduced by 50% before mixing.

5 Analysis effects

5.1 Noiseprof

NAME

Noiseprof – produce a DFT profile of the audio (use with noisered)

SYNOPSIS

`noiseprof [profile-file]`

FUNCTION

Calculate a profile of the audio for use in noise reduction. The profile will be written to the file specified by `profile-file`. If that argument is omitted, the profile will be written to the standard output.

See the description of the noisered effect for details.

5.2 Stat

NAME

Stat – enumerate audio peak & RMS levels, approx. freq., etc.

SYNOPSIS

`stat [-s scale] [-rms] [-freq] [-v] [-d]`

FUNCTION

Display time and frequency domain statistical information about the audio. Audio is passed unmodified through the processing chain.

The information is output to the standard output stream and is calculated, where n is the duration of the audio in samples, c is the number of audio channels, r is the audio sample rate, and xk represents the PCM value (in the range -1 to +1 by default) of each successive sample in the audio, as follows:

`Samples read`
: nxc

`Length (seconds)`
: n/r

`Scaled by` See -s below.

`Maximum amplitude`
: max(xk). The maximum sample value in the audio; usually this will be a positive number.

`Minimum amplitude`
: min(xk). The minimum sample value in the audio; usually this will be a negative number.

`Midline amplitude`
: 1/2min(xk)+1/2max(xk).

`Mean norm` 1/nS(|xk|). The average of the absolute value of each sample in the audio.

`Mean amplitude`
: 1/nS(xk). The average of each sample in the audio. If this figure is non-zero, then it indicates the presence of a D.C. offset (which could be removed using the dcshift effect).

`RMS amplitude`
: sqrt(1/nS(xk^2) The level of a D.C. signal that would have the same power as the audio's average power.

`Maximum delta`
: max(|xk-xk-1|)

`Minimum delta`
: min(|xk-xk-1|)

`Mean delta`
: 1/n-1S(|xk-xk-1|)

`RMS delta` sqrt(1/n-1S((xk-xk-1)^2))

`Rough frequency`
: In Hz.

`Volume Adjustment`
: The parameter to the vol effect which would make the audio as loud as possible without clipping. Note: See the discussion on clipping above for reasons why it is rarely a good idea actually to do this.

Note that the delta measurements are not applicable for multichannel audio.

The `-s` option can be used to scale the input data by a given factor. The default value of scale is 2147483647 (i.e. the maximum value of a 32-bit signed integer). Internal effects always work with signed long PCM data and so the value should relate to this fact.

The `-rms` option will convert all output average values to "root mean square" format.

The `-v` option displays only the "Volume Adjustment" value.

The `-freq` option calculates the input's power spectrum (4096 point DFT) instead of the statistics listed above. This should only be used with a single channel audio sample.

The `-d` option displays a hex dump of the 32-bit signed PCM data audio in the sound processor's internal buffer. This is mainly used to help track down endian problems that sometimes occur in cross-platform versions of the sound processor.

See also the stats effect.

5.3 Stats

NAME

Stats – multichannel aware "stat"

SYNOPSIS

```
stats [-b bits|-x bits|-s scale] [-w window-time]
```

FUNCTION

Display time domain statistical information about the audio channels; audio is passed unmodified through the processing chain. Statistics are calculated and displayed for each audio channel and, where applicable, an overall figure is also given.

For example, for a typical well-mastered stereo music file (overall, left, right):

`DC offset` 0.000803 -0.000391 0.000803

`Min level` -0.750977 -0.750977 -0.653412

`Max level` 0.708801 0.708801 0.653534

`Pk lev dB` -2.49 -2.49 -3.69

`RMS lev dB`
-19.41 -19.13 -19.71

`RMS Pk dB` -13.82 -13.82 -14.38

`RMS Tr dB` -85.25 -85.25 -82.66

`Crest factor`
- 6.79 6.32

`Flat factor`
0.00 0.00 0.00

`Pk count` 2 2 2

`Bit-depth`
16/16 16/16 16/16

`Num samples`
7.72M

`Length s` 174.973

`Scale max` 1.000000

`Window s` 0.050

DC offset, Min level, and Max level are shown, by default, in the range +/-1. If the `-b` (bits) options is given, then these three measurements will be scaled to a signed integer with the given number of bits; for example, for 16 bits, the scale would be -32768 to +32767. The `-x` option behaves the same way as `-b` except that the signed integer values are displayed in hexadecimal. The `-s` option scales the three measurements by a given floating-point number.

Pk lev dB and RMS lev dB are standard peak and RMS level measured in dBFS. RMS Pk dB and RMS Tr dB are peak and trough values for RMS level measured over a short window (default 50ms).

Crest factor is the standard ratio of peak to RMS level (note: not in dB).

Flat factor is a measure of the flatness (i.e. consecutive samples with the same value) of the signal at its peak levels (i.e. either Min level, or Max level). Pk count is the number of occasions (not the number of samples) that the signal attained either Min level, or Max level.

The right-hand Bit-depth figure is the standard definition of bit-depth i.e. bits less significant than the given number are fixed at zero. The left-hand figure is the number of most significant bits that are fixed at zero (or one for negative numbers) subtracted from the right-hand figure (the number subtracted is directly related to Pk lev dB).

For multi-channel audio, an overall figure for each of the above measurements is given and derived from the channel figures as follows: DC offset: maximum magnitude; Max level, Pk lev dB, RMS Pk dB, Bit-depth: maximum; Min level, RMS Tr dB: minimum; RMS lev dB, Flat factor, Pk count: average; Crest factor: not applicable.

Length s is the duration in seconds of the audio, and Num samples is equal to the sample-rate multiplied by Length.

Scale Max is the scaling applied to the first three measurements; specifically, it is the maximum value that could apply to Max level. Window s is the length of the window used for the peak and trough RMS measurements.

See also the stat effect.

6 Editing effects

6.1 Pad

NAME

Pad – pad (usually) the ends of the audio with silence

SYNOPSIS

```
pad { length[@position(=)] }
```

FUNCTION

Pad the audio with silence, at the beginning, the end, or any specified points through the audio. `Length` is the amount of silence to insert and `position` the position in the input audio stream at which to insert it. Any number of lengths and positions may be specified, provided that a specified position is not less that the previous one, and any time specification may be used for them. `Position` is optional for the first and last lengths specified and if omitted correspond to the beginning and the end of the audio respectively. For example,

```
pad 1.5 1.5
```

adds 1.5 seconds of silence padding at each end of the audio, whilst

```
pad 4000s@3:00
```

inserts 4000 samples of silence 3 minutes into the audio. If silence is wanted only at the end of the audio, specify either the end position or specify a zero-length pad at the start.

See also delay for an effect that can add silence at the beginning of the audio on a channel-by-channel basis.

6.2 Silence

NAME

Silence – remove portions of silence from the audio

SYNOPSIS

```
silence [-l] above-periods [duration threshold[d|%]
        [below-periods duration threshold[d|%]]
```

FUNCTION

Removes silence from the beginning, middle, or end of the audio. "Silence" is determined by a specified threshold.

The `above-periods` value is used to indicate if audio should be trimmed at the beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying a non-zero above-periods, it trims audio up until it finds nonsilence. Normally, when trimming silence from beginning of audio the `above-periods` will be 1 but it can be increased to higher values to trim all audio up to a specific count of non-silence periods. For example, if you had an audio file with two songs that each contained 2 seconds of silence before the song, you could specify an above-period of 2 to strip out both silence periods and the first song.

When `above-periods` is non-zero, you must also specify a duration and threshold. `Duration` indicates the amount of time that nonsilence must be detected before it stops

trimming audio. By increasing the duration, burst of noise can be treated as silence and trimmed off.

`Threshold` is used to indicate what sample value you should treat as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise.

When optionally trimming silence from the end of the audio, you specify a `below-periods` count. In this case, `below-period` means to remove all audio after silence is detected. Normally, this will be a value 1 of but it can be increased to skip over periods of silence that are wanted. For example, if you have a song with 2 seconds of silence in the middle and 2 second at the end, you could set `below-period` to a value of 2 to skip over the silence in the middle of the audio.

For `below-periods`, `duration` specifies a period of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. For example, if you have a song with an expected 1 second of silence in the middle and 2 seconds of silence at the end, a duration of 2 seconds could be used to skip over the middle silence.

Unfortunately, you must know the length of the silence at the end of your audio file to trim off silence reliably. A workaround is to use the silence effect in combination with the reverse effect. By first reversing the audio, you can use the `above-periods` to reliably trim all audio from what looks like the front of the file. Then reverse the file again to get back to normal.

To remove silence from the middle of a file, specify a `below-periods` that is negative. This value is then treated as a positive value and is also used to indicate that the effect should restart processing as specified by the `above-periods`, making it suitable for removing periods of silence in the middle of the audio.

The option `-l` indicates that `below-periods` duration length of audio should be left intact at the beginning of each period of silence. For example, if you want to remove long pauses between words but do not want to remove the pauses completely.

`Duration` is a time specification with the peculiarity that a bare number is interpreted as a sample count, not as a number of seconds. For specifying seconds, either use the `t` suffix (as in `"2t"`) or specify minutes, too (as in `"0:02"`).

`Threshold` numbers may be suffixed with `d` to indicate the value is in decibels, or % to indicate a percentage of maximum value of the sample value (0% specifies pure digital silence).

6.3 Splice

NAME

Splice – perform the equivalent of a cross-faded tape splice

SYNOPSIS

```
splice [-h|-t|-q] { position(=)[,excess[,leeway]] }
```

FUNCTION

Splice together audio sections. This effect provides two things over simple audio concatenation: a (usually short) cross-fade is applied at the join, and a wave similarity comparison is made to help determine the best place at which to make the join.

One of the options `-h`, `-t`, or `-q` may be given to select the fade envelope as half-cosine wave (the default), triangular (a.k.a. linear), or quarter-cosine wave respectively.

`t` Audio: correlated. Fade level: constant gain. Transitions: abrupt.

`h` Audio: correlated. Fade level: constant gain. Transitions: smooth.

`q` Audio: uncorrelated. Fade level: constant power. Transitions: smooth.

To perform a splice, first use the trim effect to select the audio sections to be joined together. As when performing a tape splice, the end of the section to be spliced onto should be trimmed with a small excess (default 0.005 seconds) of audio after the ideal joining point. The beginning of the audio section to splice on should be trimmed with the same excess (before the ideal joining point), plus an additional leeway (default 0.005 seconds). Any time specification may be used for these parameters. The sound processor should then be invoked with the two audio sections as input samples and the splice effect given with the position at which to perform the splice - this is length of the first audio section (including the excess).

It is also possible to use this effect to perform general crossfades, e.g. to join two songs. In this case, `excess` would typically be a number of seconds, the `-q` option would typically be given (to select an "equal power" cross-fade), and `leeway` should be zero (which is the default if `-q` is given).

6.4 Trim

NAME

Trim – cuts portions out of the audio

SYNOPSIS

```
trim {position(+)}
```

FUNCTION

Cuts portions out of the audio. Any number of positions may be given; audio is not sent to the output until the first position is reached. The effect then alternates between copying and discarding audio at each position. Using a value of 0 for the first `position` parameter allows copying from the beginning of the audio.

For example,

```
trim 0 10
```

will copy the first ten seconds, while

```
trim 12:34 =15:00 -2:00
```

and

```
trim 12:34 2:26 -2:00
```

will both play from 12 minutes 34 seconds into the audio up to 15 minutes into the audio (i.e. 2 minutes and 26 seconds long), then resume playing two minutes before the end of audio.

6.5 Vad

NAME

Vad – voice activity detector

SYNOPSIS

`vad [options]`

FUNCTION

Voice Activity Detector. Attempts to trim silence and quiet background sounds from the ends of (fairly high resolution i.e. 16-bit, 44-48kHz) recordings of speech. The algorithm currently uses a simple cepstral power measurement to detect voice, so may be fooled by other things, especially music. The effect can trim only from the front of the audio, so in order to trim from the back, the reverse effect must also be used. E.g.

```
norm vad
```

to trim from the front,

```
norm reverse vad reverse
```

to trim from the back, and

```
norm vad reverse vad reverse
```

to trim from both ends. The use of the norm effect is recommended, but remember that neither reverse nor norm is suitable for use with streamed audio.

Options: Default values are shown in parenthesis.

`-t num (7)`
: The measurement level used to trigger activity detection. This might need to be changed depending on the noise level, signal level and other charactistics of the input audio.

`-T num (0.25)`
: The time constant (in seconds) used to help ignore short bursts of sound.

`-s num (1)`
: The amount of audio (in seconds) to search for quieter/shorter bursts of audio to include prior to the detected trigger point.

`-g num (0.25)`
: Allowed gap (in seconds) between quieter/shorter bursts of audio to include prior to the detected trigger point.

`-p num (0)`
: The amount of audio (in seconds) to preserve before the trigger point and any found quieter/shorter bursts.

Advanced Options: These allow fine tuning of the algorithm's internal parameters.

`-b num`
: The algorithm (internally) uses adaptive noise estimation/reduction in order to detect the start of the wanted audio. This option sets the time for the initial noise estimate.

`-N num`
: Time constant used by the adaptive noise estimator for when the noise level is increasing.

`-n num` Time constant used by the adaptive noise estimator for when the noise level is decreasing.

`-r num` Amount of noise reduction to use in the detection algorithm (e.g. 0, 0.5, ...).

`-f num` Frequency of the algorithm's processing/measurements.

`-m num` Measurement duration; by default, twice the measurement period; i.e. with overlap.

`-M num` Time constant used to smooth spectral measurements.

`-h num` "Brick-wall" frequency of high-pass filter applied at the input to the detector algorithm.

`-l num` "Brick-wall" frequency of low-pass filter applied at the input to the detector algorithm.

`-H num` "Brick-wall" frequency of high-pass lifter used in the detector algorithm.

`-L num` "Brick-wall" frequency of low-pass lifter used in the detector algorithm.

See also the silence effect.

7 Low-level signal processing effects

7.1 Biquad

NAME

Biquad – 2nd-order IIR filter using externally provided coefficients

SYNOPSIS

`biquad b0 b1 b2 a0 a1 a2`

FUNCTION

Apply a biquad IIR filter with the given coefficients. Where `b` and `a` are the numerator and denominator coefficients respectively. See http://en.wikipedia.org/wiki/Digital_biquad_filter (where a0 = 1).

7.2 Downsample

NAME

Downsample – reduce sample rate by discarding samples

SYNOPSIS

`downsample [factor(2)]`

FUNCTION

Downsample the signal by an integer factor: Only the first out of each factor samples is retained, the others are discarded.

No decimation filter is applied. If the input is not a properly bandlimited baseband signal, aliasing will occur. This may be desirable, e.g., for frequency translation.

For a general resampling effect with anti-aliasing, see rate. See also upsample.

7.3 Fir

NAME

Fir – FFT convolution FIR filter using externally provided coefficients

SYNOPSIS

`fir [coefs-file|coefs]`

FUNCTION

Use the plugin's FFT convolution engine with given FIR filter coefficients. If a single argument is given then this is treated as the name of a file containing the filter coefficients (whitespace separated; may contain "#" comments). Examples:

```
fir 0.0195 -0.082 0.234 0.891 -0.145 0.043
```

```
fir coefs.txt
```

With coefs.txt containing

```
# HP filter
# freq=10000
1.2311233052619888e-01
```

```
-4.4777096106211783e-01
5.1031563346705155e-01
-6.6502926320995331e-02
...
```

7.4 Upsample

NAME

Upsample – increase sample rate by zero stuffing

SYNOPSIS

`upsample [factor]`

FUNCTION

Upsample the signal by an integer factor: factor-1 zero-value samples are inserted between each pair of input samples. As a result, the original spectrum is replicated into the new frequency space (imaging) and attenuated. This attenuation can be compensated for by adding vol factor after any further processing. The upsample effect is typically used in combination with filtering effects.

For a general resampling effect with anti-imaging, see rate. See also downsample.

8 Mastering effects

8.1 Dither

NAME

Dither – add dither noise to increase quantisation SNR

SYNOPSIS

```
dither [-S|-s|-f filter] [-a] [-p precision]
```

FUNCTION

Apply dithering to the audio. Dithering deliberately adds a small amount of noise to the signal in order to mask audible quantization effects that can occur if the output sample size is less than 24 bits. With no options, this effect will add triangular (TPDF) white noise. Noise-shaping (only for certain sample rates) can be selected with `-s`. With the `-f` option, it is possible to select a particular noise-shaping filter from the following list: lipshitz, f-weighted, modified-e-weighted, improved-e-weighted, gesemann, shibata, low-shibata, high-shibata. Note that most filter types are available only with 44100Hz sample rate. The filter types are distinguished by the following properties: audibility of noise, level of (inaudible, but in some circumstances, otherwise problematic) shaped high frequency noise, and processing speed.

The `-S` option selects a slightly "sloped" TPDF, biased towards higher frequencies. It can be used at any sampling rate but below around 22k, plain TPDF is probably better, and above around 37k, noise-shaping (if available) is probably better.

The `-a` option enables a mode where dithering (and noise-shaping if applicable) are automatically enabled only when needed. The most likely use for this is when applying fade in or out to an already dithered sample, so that the redithering applies only to the faded portions. However, auto dithering is not fool-proof, so the fades should be carefully checked for any noise modulation; if this occurs, then either re-dither the whole sample, or use trim, fade, and concatencate.

The `-p` option allows overriding the target precision.

This effect should not be followed by any other effect that effects the audio.

8.2 Rate

NAME

Rate – change audio sampling rate

SYNOPSIS

```
rate [-q|-l|-m|-h|-v] [override-options] RATE[k]
```

FUNCTION

Change the audio sampling rate (i.e. resample the audio) to any given `RATE` using a quality level defined as follows:

Here is an overview of the quality level options:

`-q` Sets quality level to "quick". The band-width is n/a. The Rej dB is around 30 @ Fs/4. The typical use is playback on Fs/4 ancient hardware.

`-l` Sets quality level to "low". The band-width is 80%. The Rej dB is 100. The typical use is playback on old hardware.

`-m` Sets quality level to "medium". The band-width is 95%. The Rej dB is 100. The typical use is audio playback.

`-h` Sets quality level to "high". The band-width is 95%. The Rej dB is 125. The typical use is 16-bit mastering (use with dither).

`-v` Sets quality level to "very high". The band-width is 95%. The Rej dB is 175. The typical use is 24-bit mastering.

Band-width is the percentage of the audio frequency band that is preserved and Rej dB is the level of noise rejection. Increasing levels of resampling quality come at the expense of increasing amounts of time to process the audio. If no quality option is given, the quality level used is "high".

The "quick" algorithm uses cubic interpolation; all others use band-limited interpolation. By default, all algorithms have a "linear" phase response; for "medium", "high" and "very high", the phase response is configurable (see below).

The simple quality selection described above provides settings that satisfy the needs of the vast majority of resampling tasks. Occasionally, however, it may be desirable to fine-tune the resampler's filter response; this can be achieved using override options, as detailed in the following table:

`-M/-I/-L` Phase response = minimum/intermediate/linear

`-s` Steep filter (band-width = 99%)

`-a` Allow aliasing/imaging above the pass-band

`-b 74-99.7`
Any band-width %

`-p 0-100` Any phase response (0 = minimum, 25 = intermediate, 50 = linear, 100 = maximum)

N.B. Override options cannot be used with the "quick" or "low" quality algorithms.

All resamplers use filters that can sometimes create "echo" (a.k.a. "ringing") artefacts with transient signals such as those that occur with "finger snaps" or other highly percussive sounds. Such artefacts are much more noticeable to the human ear if they occur before the transient ("pre-echo") than if they occur after it ("post-echo"). Note that frequency of any such artefacts is related to the smaller of the original and new sampling rates but that if this is at least 44.1kHz, then the artefacts will lie outside the range of human hearing.

A phase response setting may be used to control the distribution of any transient echo between "pre" and "post": with minimum phase, there is no pre-echo but the longest post-echo; with linear phase, pre and post echo are in equal amounts (in signal terms, but not audibility terms); the intermediate phase setting attempts to find the best compromise by selecting a small length (and level) of pre-echo and a medium lengthed post-echo.

Minimum, intermediate, or linear phase response is selected using the `-M`, `-I`, or `-L` option; a custom phase response can be created with the `-p` option. Note that phase responses between "linear" and "maximum" (greater than 50) are rarely useful.

A resampler's band-width setting determines how much of the frequency content of the original signal (w.r.t. the original sample rate when up-sampling, or the new sample rate when down-sampling) is preserved during conversion. The term "pass-band" is used to refer to all frequencies up to the band-width point (e.g. for 44.1kHz sampling rate, and a resampling band-width of 95%, the pass-band represents frequencies from 0Hz (D.C.) to circa 21kHz). Increasing the resampler's band-width results in a slower conversion and can increase transient echo artefacts (and vice versa).

The `-s` "steep filter" option changes resampling band-width from the default 95% (based on the 3dB point), to 99%. The `-b` option allows the band-width to be set to any value in the range 74-99.7 %, but note that band-width values greater than 99% are not recommended for normal use as they can cause excessive transient echo.

If the `-a` option is given, then aliasing/imaging above the passband is allowed. For example, with 44.1kHz sampling rate, and a resampling band-width of 95%, this means that frequency content above 21kHz can be distorted; however, since this is above the pass-band (i.e. above the highest frequency of interest/audibility), this may not be a problem. The benefits of allowing aliasing/imaging are reduced processing time, and reduced (by almost half) transient echo artefacts. Note that if this option is given, then the minimum band-width allowable with `-b` increases to 85%.

Examples:

```
rate -s -a 44100 dither -s
```

This does default (high) quality resampling; overrides: steep filter, allow aliasing; to 44.1kHz sample rate; noise-shaped dither.

```
rate -v -I -b 90 48k
```

This does very high quality resampling; overrides: intermediate phase, band-width 90%; to 48k sample rate.

The pitch and speed effects use the rate effect at their core.

9 Miscellaneous effects

9.1 Synth

NAME

Synth – synthesise/modulate audio tones or noise signals

SYNOPSIS

```
synth [-j KEY] [-n] [len [off [ph [p1 [p2 [p3]]]]]] {[type] [combine]
      [[%]freq[k][:|+|/|-[%]freq2[k]]] [off [ph [p1 [p2 [p3]]]]]}
```

FUNCTION

This effect can be used to generate fixed or swept frequency audio tones with various wave shapes, or to generate wide-band noise of various "colours". Multiple synth effects can be cascaded to produce more complex waveforms; at each stage it is possible to choose whether the generated waveform will be mixed with, or modulated onto the output from the previous stage. Audio for each channel in a multi-channel audio sample can be synthesised independently.

Note that since this effect is generative you need to use soundfx.Generate() instead of soundfx.Process() with this effect. You might also want to set the `Channels`, `Bits`, and `Frequency` tags when calling soundfx.Generate() to set the desired sample format. The length can also be specified as a parameter to `synth` or by another given effect that has an associated length.

For example, the following produces a 3 second audio sample containing a sine-wave swept from 300 to 3300 Hz:

```
synth 3 sine 300-3300
```

Multiple channels can be synthesised by specifying the set of parameters shown between braces multiple times; the following puts the swept tone in the left channel and adds "brown" noise in the right:

```
synth 3 sine 300-3300 brownnoise
```

The following example shows how two synth effects can be cascaded to create a more complex waveform:

```
synth 0.5 sine 200-500 synth 0.5 sine fmod 700-100
```

Frequencies can also be given in "scientific" note notation, or, by prefixing a "%" character, as a number of semitones relative to "middle A" (440 Hz). For example, the following could be used to help tune a guitar's low "E" string:

```
synth 4 pluck %-29
```

N.B.: This effect generates audio at maximum volume (0dBFS), which means that there is a high chance of clipping when using the audio subsequently, so in many cases, you will want to follow this effect with the gain effect to prevent this from happening. (See also Clipping above.) Note that, by default, the `synth` effect incorporates the functionality of `gain -h` (see the gain effect for details); synth's `-n` option may be given to disable this behaviour.

A detailed description of each synth parameter follows:

`len` is the length of audio to synthesise (any time specification); a value of 0 indicated to use the input length, which is also the default.

`type` is one of sine, square, triangle, sawtooth, trapezium, exp, [white]noise, tpdfnoise, pinknoise, brownnoise, pluck; default=sine.

`combine` is one of create, mix, amod (amplitude modulation), fmod (frequency modulation); default=create.

`freq` / `freq2` are the frequencies at the beginning/end of synthesis in Hz or, if preceded with "%", semitones relative to A (440 Hz); alternatively, "scientific" note notation (e.g. E2) may be used. The default frequency is 440Hz. By default, the tuning used with the note notations is "equal temperament"; the `-j` KEY option selects "just intonation", where `KEY` is an integer number of semitones relative to A (so for example, -9 or 3 selects the key of C), or a note in scientific notation.

If `freq2` is given, then len must also have been given and the generated tone will be swept between the given frequencies. The two given frequencies must be separated by one of the characters ":", "+", "/", or "-". This character is used to specify the sweep function as follows:

`:`	Linear: the tone will change by a fixed number of hertz per second.
`+`	Square: a second-order function is used to change the tone.
`/`	Exponential: the tone will change by a fixed number of semitones per second.
`-`	Exponential: as "/", but initial phase always zero, and stepped (less smooth) frequency changes.

Not used for noise.

`off` is the bias (DC-offset) of the signal in percent; default=0.

`ph` is the phase shift in percentage of 1 cycle; default=0. Not used for noise.

`p1` is the percentage of each cycle that is "on" (square), or "rising" (triangle, exp, trapezium); default=50 (square, triangle, exp), default=10 (trapezium), or sustain (pluck); default=40.

`p2` (trapezium): the percentage through each cycle at which "falling" begins; default=50. exp: the amplitude in multiples of 2dB; default=50, or tone-1 (pluck); default=20.

`p3` (trapezium): the percentage through each cycle at which "falling" ends; default=60, or tone-2 (pluck); default=90.

10 Mixing effects

10.1 Channels

NAME

Channels – auto mix or duplicate to change number of channels

SYNOPSIS

```
channels CHANNELS
```

FUNCTION

Invoke a simple algorithm to change the number of channels in the audio signal to the given number `CHANNELS`: mixing if decreasing the number of channels or duplicating if increasing the number of channels.

See also remix for an effect that allows channels to be mixed/selected arbitrarily.

10.2 Remix

NAME

Remix – produce arbitrarily mixed output channels

SYNOPSIS

```
remix [-a|-m|-p] <out-spec>
      out-spec = in-spec{,in-spec} | 0
      in-spec = [in-chan][-[in-chan2]][vol-spec]
      vol-spec = p|i|v[volume]
```

FUNCTION

Select and mix input audio channels into output audio channels. Each output channel is specified, in turn, by a given `out-spec`: a list of contributing input channels and volume specifications.

An `out-spec` contains comma-separated input channel-numbers and hyphen-delimited channel-number ranges; alternatively, 0 may be given to create a silent output channel. For example,

```
remix 6 7 8 0
```

creates an output sample with four channels, where channels 1, 2, and 3 are copies of channels 6, 7, and 8 in the input sample, and channel 4 is silent. Whereas

```
remix 1-3,7 3
```

creates a (somewhat bizarre) stereo output sample where the left channel is a mix-down of input channels 1, 2, 3, and 7, and the right channel is a copy of input channel 3.

Where a range of channels is specified, the channel numbers to the left and right of the hyphen are optional and default to 1 and to the number of input channels respectively. Thus

```
remix
```

performs a mix-down of all input channels to mono.

By default, where an output channel is mixed from multiple (n) input channels, each input channel will be scaled by a factor of 1/n. Custom mixing volumes can be set by

following a given input channel or range of input channels with a `vol-spec` (volume specification). This is one of the letters `p`, `i`, or `v`, followed by a volume number, the meaning of which depends on the given letter and is defined as follows:

`p` Power adjust in dB. 0 = no change.

`i` Power adjust in dB. As `p`, but invert the audio

`v` Voltage multiplier. 1 = no change, 0.5 around 6dB attenuation, 2 around 6dB gain, -1 = invert.

If an `out-spec` includes at least one `vol-spec` then, by default, -1/n scaling is not applied to any other channels in the same `out-spec` (though may be in other out-specs). The `-a` (automatic) option however, can be given to retain the automatic scaling in this case. For example,

```
remix 1,2 3,4v0.8
```

results in channel level multipliers of 0.5,0.5 1,0.8, whereas

```
remix -a 1,2 3,4v0.8
```

results in channel level multipliers of 0.5,0.5 0.5,0.8.

The `-m` (manual) option disables all automatic volume adjustments, so

```
remix -m 1,2 3,4v0.8
```

results in channel level multipliers of 1,1 1,0.8.

The volume number is optional and omitting it corresponds to no volume change; however, the only case in which this is useful is in conjunction with `i`. For example, if the input sample is stereo, then

```
remix 1,2i
```

is a mono equivalent of the oops effect.

If the `-p` option is given, then any automatic -1/n scaling is replaced by -1/sqrt(n) ("power") scaling; this gives a louder mix but one that might occasionally clip.

One use of the remix effect is to split a Hollywood sample into a set of samples, each containing one of the constituent channels (in order to perform subsequent processing on individual audio channels).

See also the swap effect.

10.3 Swap

NAME

Swap – swap pairs of channels

SYNOPSIS

`swap`

FUNCTION

Swap stereo channels. If the input is not stereo, pairs of channels are swapped, and a possible odd last channel passed through. E.g., for seven channels, the output order will be 2, 1, 4, 3, 6, 5, 7.

See also remix for an effect that allows arbitrary channel selection and ordering (and mixing).

11 Pitch/tempo effects

11.1 Bend

NAME

Bend – bend pitch at given times without changing tempo

SYNOPSIS

```
bend [-f frame-rate(25)] [-o over-sample(16)] { start-position(+),cents,end-position(+) }
```

FUNCTION

Changes pitch by specified amounts at specified times. Each given triple: `start-position`, `cents`, `end-position` specifies one bend. `cents` is the number of cents (100 cents = 1 semitone) by which to bend the pitch. The other values specify the points in time at which to start and end bending the pitch, respectively.

The pitch-bending algorithm utilises the Discrete Fourier Transform DFT) at a particular frame rate and over-sampling rate. The `-f` and `-o` parameters may be used to adjust these parameters and thus control the smoothness of the changes in pitch.

See also pitch.

11.2 Pitch

NAME

Pitch – adjust pitch (= key) without changing tempo

SYNOPSIS

```
pitch [-q] shift [segment [search [overlap]]]
```

FUNCTION

Change the audio pitch (but not tempo).

`Shift` gives the pitch shift as positive or negative "cents" (i.e. 100ths of a semitone). See the tempo effect for a description of the other parameters.

See also the bend, speed, and tempo effects.

11.3 Speed

NAME

Speed – adjust pitch & tempo together

SYNOPSIS

```
speed factor[c]
```

FUNCTION

Adjust the audio speed (pitch and tempo together). `factor` is either the ratio of the new speed to the old speed: greater than 1 speeds up, less than 1 slows down, or, if appended with the letter `c`, the number of cents (i.e. 100ths of a semitone) by which the pitch (and tempo) should be adjusted: greater than 0 increases, less than 0 decreases.

Technically, the speed effect only changes the sample rate information, leaving the samples themselves untouched. The rate effect is invoked automatically to resample to the

output sample rate, using its default quality/speed. For higher quality or higher speed resampling, in addition to the speed effect, specify the rate effect with the desired quality option.

See also the bend, pitch, and tempo effects.

11.4 Stretch

NAME

Stretch – adjust tempo without changing pitch (simple alg.)

SYNOPSIS

`stretch factor [window fade shift fading]`

FUNCTION

Change the audio duration (but not its pitch). This effect is broadly equivalent to the tempo effect with (factor inverted and) search set to zero, so in general, its results are comparatively poor; it is retained as it can sometimes out-perform tempo for small factors.

`factor` of stretching: `>`1 lengthen, `<`1 shorten duration. `window` size is in ms. Default is 20ms. The `fade` option, can be "lin". `shift` ratio, in [0 1]. Default depends on stretch factor. 1 to shorten, 0.8 to lengthen. The `fading` ratio, in [0 0.5]. The amount of a fade's default depends on `factor` and `shift`.

See also the tempo effect.

11.5 Tempo

NAME

Tempo – adjust tempo without changing pitch (WSOLA alg.)

SYNOPSIS

`tempo [-q] [-m|-s|-l] factor [segment [search [overlap]]]`

FUNCTION

Change the audio playback speed but not its pitch. This effect uses the WSOLA algorithm. The audio is chopped up into segments which are then shifted in the time domain and overlapped (crossfaded) at points where their waveforms are most similar as determined by measurement of "least squares".

By default, linear searches are used to find the best overlapping points. If the optional `-q` parameter is given, tree searches are used instead. This makes the effect work more quickly, but the result may not sound as good. However, if you must improve the processing speed, this generally reduces the sound quality less than reducing the search or overlap values.

The `-m` option is used to optimize default values of `segment`, `search` and `overlap` for music processing.

The `-s` option is used to optimize default values of `segment`, `search` and `overlap` for speech processing.

The `-l` option is used to optimize default values of `segment`, `search` and `overlap` for "linear" processing that tends to cause more noticeable distortion but may be useful when factor is close to 1.

If `-m`, `-s`, or `-l` is specified, the default value of `segment` will be calculated based on `factor`, while default `search` and `overlap` values are based on `segment`. Any values you provide still override these default values.

`factor` gives the ratio of new tempo to the old tempo, so e.g. 1.1 speeds up the tempo by 10%, and 0.9 slows it down by 10%.

The optional `segment` parameter selects the algorithm's segment size in milliseconds. If no other flags are specified, the default value is 82 and is typically suited to making small changes to the tempo of music. For larger changes (e.g. a factor of 2), 41 ms may give a better result. The `-m`, `-s`, and `-l` flags will cause the `segment` default to be automatically adjusted based on factor. For example using `-s` (for speech) with a tempo of 1.25 will calculate a default `segment` value of 32.

The optional `search` parameter gives the audio length in milliseconds over which the algorithm will search for overlapping points. If no other flags are specified, the default value is 14.68. Larger values use more processing time and may or may not produce better results. A practical maximum is half the value of `segment`. Search can be reduced to cut processing time at the risk of degrading output quality. The `-m`, `-s`, and `-l` flags will cause the `search` default to be automatically adjusted based on `segment`.

The optional `overlap` parameter gives the segment overlap length in milliseconds. Default value is 12, but `-m`, `-s`, or `-l` flags automatically adjust `overlap` based on `segment` size. Increasing overlap increases processing time and may increase quality. A practical maximum for `overlap` is the value of `search`, with `overlap` typically being (at least) a little smaller than `search`.

See also speed for an effect that changes tempo and pitch together, pitch and bend for effects that change pitch only, and stretch for an effect that changes tempo using a different algorithm.

12 Production effects

12.1 Chorus

NAME

Chorus – make a single instrument sound like many

SYNOPSIS

`chorus gain-in gain-out <delay decay speed depth -s|-t>`

FUNCTION

Add a chorus effect to the audio. This can make a single vocal sound like a chorus, but can also be applied to instrumentation.

Chorus resembles an echo effect with a short delay, but whereas with echo the delay is constant, with chorus, it is varied using sinusoidal or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocals are slightly off key.

Each four-tuple parameter `delay`, `decay`, `speed` and `depth` gives the delay in milliseconds and the `decay` (relative to `gain-in`) with a modulation speed in Hz using depth in milliseconds. The modulation is either sinusoidal (`-s`) or triangular (`-t`). `Gain-out` is the volume of the output.

A typical delay is around 40ms to 60ms; the modulation speed is best near 0.25Hz and the modulation depth around 2ms. For example, a single delay:

```
chorus 0.7 0.9 55 0.4 0.25 2 -t
```

Two delays of the original samples:

```
chorus 0.6 0.9 50 0.4 0.25 2 -t 60 0.32 0.4 1.3 -s
```

A fuller sounding chorus (with three additional delays):

```
chorus 0.5 0.9 50 0.4 0.25 2 -t 60 0.32 0.4 2.3 -t 40 0.3 0.3 1.3 -s
```

12.2 Delay

NAME

Delay – delay one or more channels

SYNOPSIS

`delay {position(=)}`

FUNCTION

Delay one or more audio channels such that they start at the given position. For example:

```
delay 1.5 +1 3000s
```

This delays the first channel by 1.5 seconds, the second channel by 2.5 seconds (one second more than the previous channel), the third channel by 3000 samples, and leaves any other channels that may be present un-delayed.

12.3 Echo

NAME

Echo – add an echo

SYNOPSIS

`echo gain-in gain-out <delay decay>`

FUNCTION

Add echoing to the audio. Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the `delay` (time), and the loudness of the reflected signal is the `decay`. Multiple echoes can have different delays and decays.

Each given `delay decay` pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. `Gain-out` is the volume of the output. For example: This will make it sound as if there are twice as many instruments as are actually playing:

```
echo 0.8 0.88 60 0.4
```

If the delay is very short, then it sound like a (metallic) robot playing music:

```
echo 0.8 0.88 6 0.4
```

A longer delay will sound like an open air concert in the mountains:

```
echo 0.8 0.9 1000 0.3
```

One mountain more, and:

```
echo 0.8 0.9 1000 0.3 1800 0.25
```

12.4 Echos

NAME

Echos – add a sequence of echos

SYNOPSIS

`echos gain-in gain-out <delay decay>`

FUNCTION

Add a sequence of echoes to the audio. Each `delay decay` pair gives the delay in milliseconds and the decay (relative to `gain-in`) of that echo. `Gain-out` is the volume of the output.

Like the echo effect, `echos` stand for "ECHO in Sequel", that is the first echos takes the input, the second the input and the first echos, the third the input and the first and the second echos, ... and so on. Care should be taken using many echos; a single echos has the same effect as a single echo.

The sample will be bounced twice in symmetric echos:

```
echos 0.8 0.7 700 0.25 700 0.3
```

The sample will be bounced twice in asymmetric echos:

```
echos 0.8 0.7 700 0.25 900 0.3
```

The sample will sound as if played in a garage:

```
echos 0.8 0.7 40 0.25 63 0.3
```

12.5 Flanger

NAME
Flanger – stereo flanger

SYNOPSIS
`flanger [delay depth regen width speed shape phase interp]`

FUNCTION
Apply a flanging effect to the audio. All parameters are optional (right to left).

`delay` Base delay in milliseconds. The valid range is 0 to 30. Defaults to 0.

`depth` Added swept delay in milliseconds. The valid range is 0 to 10. Defaults to 2.

`regen` Percentage regeneration (delayed signal feedback). The valid range is -95 to 95. Defaults to 0.

`width` Percentage of delayed signal mixed with original. The valid range is 0 to 100. Defaults to 71.

`speed` Sweeps per second (Hz). The valid range is 0.1 to 10. Defaults to 0.5.

`shape` Swept wave shape. This can be either `sine` or `triangle`. Defaults to `sine`.

`phase` Swept wave percentage phase-shift for multi-channel (e.g. stereo). The valid range is 0 to 100. 0 = 100 = same phase on each channel. Defaults to 25.

`interp` Digital delay-line interpolation. This can be either `linear` or `quadratic`. Defaults to `linear`.

12.6 Overdrive

NAME
Overdrive – non-linear distortion

SYNOPSIS
`overdrive [gain(20) [colour(20)]]`

FUNCTION
Non linear distortion. The `colour` parameter controls the amount of even harmonic content in the over-driven output.

12.7 Phaser

NAME
Phaser – phase shifter

SYNOPSIS
`phaser gain-in gain-out delay decay speed [-s|-t]`

FUNCTION
Add a phasing effect to the audio.

`Delay`, `decay`, `speed` gives the delay in milliseconds and the decay (relative to `gain-in`) with a modulation speed in Hz. The modulation is either sinusoidal (`-s`) - preferable

for multiple instruments, or triangular (`-t`) - gives single instruments a sharper phasing effect. The `decay` should be less than 0.5 to avoid feedback, and usually no less than 0.1. `Gain-out` is the volume of the output.

For example:

```
phaser 0.8 0.74 3 0.4 0.5 -t
```

Gentler:

```
phaser 0.9 0.85 4 0.23 1.3 -s
```

A popular sound:

```
phaser 0.89 0.85 1 0.24 2 -t
```

More severe:

```
phaser 0.6 0.66 3 0.6 2 -t
```

12.8 Repeat

NAME

Repeat – loop the audio a number of times

SYNOPSIS

```
repeat [count(1)|-]
```

FUNCTION

Repeat the entire audio count times, or once if `count` is not given. The special value `-` requests infinite repetition. Note that repeating once yields two copies: the original audio and the repeated audio.

12.9 Reverb

NAME

Reverb – add reverberation

SYNOPSIS

```
reverb [-w|--wet-only] [reverberance (50%) [HF-damping (50%)
       [room-scale (100%) [stereo-depth (100%)
       [pre-delay (0ms) [wet-gain (0dB)]]]]]]
```

FUNCTION

Add reverberation to the audio using the "freeverb" algorithm. A reverberation effect is sometimes desirable for concert halls that are too small or contain so many people that the hall's natural reverberance is diminished. Applying a small amount of stereo reverb to a (dry) mono signal will usually make it sound more natural.

Note that this effect increases both the volume and the length of the audio, so to prevent clipping in these domains, a typical invocation might be:

```
gain -3 pad 0 3 reverb
```

The `-w` option can be given to select only the "wet" signal, thus allowing it to be processed further, independently of the "dry" signal.

12.10 Reverse

NAME
Reverse – reverse the audio

SYNOPSIS
`reverse`

FUNCTION
Reverse the audio completely.

12.11 Tremolo

NAME
Tremolo – sinusoidal volume modulation

SYNOPSIS
`tremolo speed [depth]`

FUNCTION
Apply a tremolo (low frequency amplitude modulation) effect to the audio. The tremolo frequency in Hz is given by `speed`, and the `depth` as a percentage by depth (default 40).

13 Specialised filters/mixers

13.1 Deemph

NAME

Deemph – ISO 908 CD de-emphasis (shelving) IIR filter

SYNOPSIS

`deemph`

FUNCTION

Apply Compact Disc (IEC 60908) de-emphasis (a treble attenuation shelving filter).

Pre-emphasis was applied in the mastering of some CDs issued in the early 1980s. These included many classical music albums, as well as now sought-after issues of albums by The Beatles, Pink Floyd and others. Pre-emphasis should be removed at playback time by a de-emphasis filter in the playback device. However, not all modern CD players have this filter, and very few PC CD drives have it; playing pre-emphasised audio without the correct de-emphasis filter results in audio that sounds harsh and is far from what its creators intended.

With the deemph effect, it is possible to apply the necessary de-emphasis to audio that has been extracted from a pre-emphasised CD, and then either burn the de-emphasised audio to a new CD (which will then play correctly on any CD player), or simply play the correctly de-emphasised audio files on the PC.

The de-emphasis filter is implemented as a biquad and requires the input audio sample rate to be either 44.1kHz or 48kHz. Maximum deviation from the ideal response is only 0.06dB (up to 20kHz).

See also the bass and treble shelving equalisation effects.

13.2 Earwax

NAME

Earwax – process CD audio to best effect for headphone use

SYNOPSIS

`earwax`

FUNCTION

Makes audio easier to listen to on headphones. Adds `"cues"` to 44.1kHz stereo (i.e. audio CD format) audio so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers).

13.3 Noisered

NAME

Noisered – filter out noise from the audio

SYNOPSIS

`noisered [profile-file [amount]]`

FUNCTION

Reduce noise in the audio signal by profiling and filtering. This effect is moderately effective at removing consistent background noise such as hiss or hum. To use it, first run the noiseprof effect on a section of audio that ideally would contain silence but in fact contains noise - such sections are typically found at the beginning or the end of a recording. noiseprof will write out a noise profile to `profile-file`, or to the standard output if no `profile-file` or if '-' is given.

To actually remove the noise, run the noisered effect; noisered will reduce noise according to a noise profile (which was generated by noiseprof), from `profile-file`.

How much noise should be removed is specified by `amount` - a number between 0 and 1 with a default of 0.5. Higher numbers will remove more noise but present a greater likelihood of removing wanted components of the audio signal. Before replacing an original recording with a noise-reduced version, experiment with different amount values to find the optimal one for your audio; use headphones to check that you are happy with the results, paying particular attention to quieter sections of the audio.

13.4 Oops

NAME

Oops – Out Of Phase Stereo (or "Karaoke") effect

SYNOPSIS

`oops`

FUNCTION

Mixes stereo to twin-mono where each mono channel contains the difference between the left and right stereo channels. This is sometimes known as the "karaoke" effect as it often has the effect of removing most or all of the vocals from a recording. It is equivalent to `remix 1,2i 1,2i`.

13.5 Riaa

NAME

Riaa – RIAA vinyl playback equalisation

SYNOPSIS

`riaa`

FUNCTION

Apply RIAA vinyl playback equalisation. The sampling rate must be one of: 44.1, 48, 88.2, 96 kHz.

14 Tone/filter effects

14.1 Allpass

NAME

Allpass – RBJ all-pass biquad IIR filter

SYNOPSIS

`allpass frequency[k] width[h|k|o|q]`

FUNCTION

Apply a two-pole all-pass filter with central frequency (in Hz) frequency, and filter-width width. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship. The filter is described in detail in R. Bristow-Johnson, Cookbook formulae for audio EQ biquad filter coefficients, http://musicdsp.org/files/Audio-EQ-Cookbook.txt

14.2 Band

NAME

Band – SPKit resonator band-pass IIR filter

SYNOPSIS

`band [-n] center[k] [width[h|k|o|q]]`

FUNCTION

Apply a band-pass filter. The frequency response drops logarithmically around the center frequency. The `width` parameter gives the slope of the drop. The frequencies at `center + width` and `center - width` will be half of their original amplitudes.

`band` defaults to a mode oriented to pitched audio, i.e. voice, singing, or instrumental music. The `-n` (for noise) option uses the alternate mode for un-pitched audio (e.g. percussion).

Warning: `-n` introduces a power-gain of about 11dB in the filter, so beware of output clipping. `band` introduces noise in the shape of the filter, i.e. peaking at the center frequency and settling around it.

See also sinc for a bandpass filter with steeper shoulders.

14.3 Bandpass

NAME

Bandpass – RBJ band-pass biquad IIR filter

SYNOPSIS

`bandpass [-c] frequency[k] width[h|k|o|q]`

FUNCTION

Apply a two-pole Butterworth band-pass filter with central frequency `frequency`, and (3dB-point) band-width `width`. The `-c` option selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filter rolls off at 6dB per octave (20dB per decade).

See also sinc for a bandpass filter with steeper shoulders.

14.4 Bandreject

NAME

Bandreject – RBJ band-reject biquad IIR filter

SYNOPSIS

`bandreject frequency[k] width[h|k|o|q]`

FUNCTION

Apply a two-pole Butterworth band-reject filter with central frequency `frequency`, and (3dB-point) band-width `width`. The filter rolls off at 6dB per octave (20dB per decade).

See also sinc for a bandpass filter with steeper shoulders.

14.5 Bass

NAME

Bass – tone control: RBJ shelving biquad IIR filter

SYNOPSIS

`bass gain [frequency[k] [width[s|h|k|o|q]]]`

FUNCTION

Boost or cut the bass (lower) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

`gain` gives the gain at 0 Hz. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

If desired, the filter can be fine-tuned using the following optional parameters:

`frequency` sets the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz.

`width` determines how steep is the filter's shelf transition. In addition to the common width specification methods described above, `slope` (the default, or if appended with `s`) may be used. The useful range of `slope` is about 0.3, for a gentle slope, to 1 (the maximum), for a steep slope; the default value is 0.5.

See also equalizer for a peaking equalisation effect.

14.6 Equalizer

NAME

Equalizer – RBJ peaking equalisation biquad IIR filter

SYNOPSIS

`equalizer frequency[k] width[q|o|h|k] gain`

FUNCTION

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike bandpass and bandreject filters) that at all other frequencies is unchanged.

`Frequency` gives the filter's central frequency in Hz, `width`, the band-width, and `gain` the required gain or attenuation in dB. Beware of clipping when using a positive gain.

In order to produce complex equalisation curves, this effect can be given several times, each with a different central frequency.

See also bass and treble for shelving equalisation effects.

14.7 Highpass

NAME

Highpass – high-pass filter: Single pole or RBJ biquad IIR

SYNOPSIS

```
highpass [-1|-2] frequency[k] [width[q|o|h|k]]
```

FUNCTION

Apply a high-pass filter with 3dB point frequency. The filter can be either single-pole (with -1), or double-pole (the default, or with -2). `Width` applies only to double-pole filters; the default is Q = 0.707 and gives a Butterworth response. The filter rolls off at 6dB per pole per octave (20dB per pole per decade).

See also sinc for filters with a steeper roll-off.

14.8 Hilbert

NAME

Hilbert – hilbert transform filter (90 degrees phase shift)

SYNOPSIS

```
hilbert [-n taps]
```

FUNCTION

Apply an odd-tap Hilbert transform filter, phase-shifting the signal by 90 degrees.

This is used in many matrix coding schemes and for analytic signal generation. The process is often written as a multiplication by i (or j), the imaginary unit.

An odd-tap Hilbert transform filter has a bandpass characteristic, attenuating the lowest and highest frequencies. Its band-width can be controlled by the number of filter taps, which can be specified with `-n`. By default, the number of taps is chosen for a cutoff frequency of about 75 Hz.

14.9 Lowpass

NAME

Lowpass – low-pass filter: single pole or RBJ biquad IIR

SYNOPSIS

```
lowpass [-1|-2] frequency[k] [width[q|o|h|k]]
```

FUNCTION

Apply a low-pass filter. See the description of the highpass effect for details.

14.10 Sinc

NAME

Sinc – sinc-windowed low/high-pass/band-pass/reject FIR

SYNOPSIS

```
sinc [-a att|-b beta] [-p phase|-M|-I|-L] [-t tbw|-n taps] [freqHP]
     [-freqLP [-t tbw|-n taps]]
```

FUNCTION

Apply a sinc kaiser-windowed low-pass, high-pass, band-pass, or band-reject filter to the signal. The `freqHP` and `freqLP` parameters give the frequencies of the 6dB points of a high-pass and low-pass filter that may be invoked individually, or together. If both are given, then `freqHP` less than `freqLP` creates a bandpass filter, `freqHP` greater than `freqLP` creates a band-reject filter. For example, the invocations

```
sinc 3k
sinc -4k
sinc 3k-4k
sinc 4k-3k
```

create a high-pass, low-pass, band-pass, and band-reject filter respectively.

The default stop-band attenuation of 120dB can be overridden with `-a`; alternatively, the kaiser-window "beta" parameter can be given directly with `-b`.

The default transition band-width of 5% of the total band can be overridden with `-t` (and tbw in Hertz); alternatively, the number of filter taps can be given directly with `-n`.

If both `freqHP` and `freqLP` are given, then a `-t` or `-n` option given to the left of the frequencies applies to both frequencies; one of these options given to the right of the frequencies applies only to `freqLP`.

The `-p`, `-M`, `-I`, and `-L` options control the filter's phase response; see the rate effect for details.

14.11 Treble

NAME

Treble – tone control: RBJ shelving biquad IIR filter

SYNOPSIS

```
treble gain [frequency[k] [width[s|h|k|o|q]]]
```

FUNCTION

Boost or cut the treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

`gain` gives the gain at whichever is the lower of -22 kHz and the Nyquist frequency. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

If desired, the filter can be fine-tuned using the following optional parameters:

`frequency` sets the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 3 kHz.

`width` determines how steep is the filter's shelf transition. In addition to the common width specification methods described above, `"slope"` (the default, or if appended with `"s"`) may be used. The useful range of `"slope"` is about 0.3, for a gentle slope, to 1 (the maximum), for a steep slope; the default value is 0.5.

See also equalizer for a peaking equalisation effect.

15 Volume/level effects

15.1 Compand

NAME

Compand – signal level compression/expansion/limiting

SYNOPSIS

```
compand attack1,decay1{,attack2,decay2} [soft-knee-dB:]in-dB1
    [,out-dB1]{,in-dB2,out-dB2}[gain [initial-volume-dB [delay]]]
```

FUNCTION

Compand (compress or expand) the dynamic range of the audio.

The `attack` and `decay` parameters (in seconds) determine the time over which the instantaneous level of the input signal is averaged to determine its volume; attacks refer to increases in volume and decays refer to decreases. For most situations, the attack time (response to the music getting louder) should be shorter than the decay time because the human ear is more sensitive to sudden loud music than sudden soft music. Where more than one pair of attack/decay parameters are specified, each input channel is companded separately and the number of pairs must agree with the number of input channels. Typical values are 0.3,0.8 seconds.

The second parameter is a list of points on the compander's transfer function specified in dB relative to the maximum possible signal amplitude. The input values must be in a strictly increasing order but the transfer function does not have to be monotonically rising. If omitted, the value of `out-dB1` defaults to the same value as `in-dB1`; levels below `in-dB1` are not companded (but may have gain applied to them). The point 0,0 is assumed but may be overridden (by 0,out-dBn). If the list is preceded by a `soft-knee-dB` value, then the points at where adjacent line segments on the transfer function meet will be rounded by the amount given. Typical values for the transfer function are 6:-70,-60,-20.

The third (optional) parameter is an additional gain in dB to be applied at all points on the transfer function and allows easy adjustment of the overall gain.

The fourth (optional) parameter is an initial level to be assumed for each channel when companding starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding action has begun to operate: it is quite probable that in such an event, the output would be severely clipped while the compander gain properly adjusts itself. A typical value (for audio which is initially quiet) is -90 dB.

The fifth (optional) parameter is a delay in seconds. The input signal is analysed immediately to control the compander, but it is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the compander to effectively operate in a "predictive" rather than a reactive mode. A typical value is 0.2 seconds.

The following example might be used to make a piece of music with both quiet and loud passages suitable for listening to in a noisy environment such as a moving vehicle:

```
compand 0.3,1 6:-70,-60,-20 -5 -90 0.2
```

The transfer function (6:-70,...) says that very soft sounds (below -70dB) will remain unchanged. This will stop the compander from boosting the volume on "silent" passages such as between movements. However, sounds in the range -60dB to 0dB (maximum volume) will be boosted so that the 60dB dynamic range of the original music will be compressed 3-to-1 into a 20dB range, which is wide enough to enjoy the music but narrow enough to get around the road noise. The "6:" selects 6dB soft-knee companding. The -5 (dB) output gain is needed to avoid clipping (the number is inexact, and was derived by experimentation). The -90 (dB) for the initial volume will work fine for a clip that starts with near silence, and the delay of 0.2 (seconds) has the effect of causing the compander to react a bit more quickly to sudden volume changes.

In the next example, compand is being used as a noise-gate for when the noise is at a lower level than the signal:

```
compand .1,.2 -inf,-50.1,-inf,-50,-50 0 -90 .1
```

Here is another noise-gate, this time for when the noise is at a higher level than the signal (making it, in some ways, similar to squelch):

```
compand .1,.1 -45.1,-45,-inf,0,-inf 45 -90 .1
```

See also mcompand for a multiple-band companding effect.

15.2 Contrast

NAME

Contrast – phase contrast volume enhancement

SYNOPSIS

```
contrast [enhancement-amount(75)]
```

FUNCTION

Comparable with compression, this effect modifies an audio signal to make it sound louder. `enhancement-amount` controls the amount of the enhancement and is a number in the range 0-100. Note that `enhancement-amount` = 0 still gives a significant contrast enhancement.

See also the compand and mcompand effects.

15.3 DCShift

NAME

DCShift – apply or remove DC offset

SYNOPSIS

```
dcshift shift [limitergain]
```

FUNCTION

Apply a DC shift to the audio. This can be useful to remove a DC offset (caused perhaps by a hardware problem in the recording chain) from the audio. The effect of a DC offset is reduced headroom and hence volume. The stat or stats effect can be used to determine if a signal has a DC offset.

The given `shift` value is a floating point number in the range of +/-2 that indicates the amount to shift the audio (which is in the range of +/-1).

An optional `limitergain` can be specified as well. It should have a value much less than 1 (e.g. 0.05 or 0.02) and is used only on peaks to prevent clipping.

An alternative approach to removing a DC offset (albeit with a short delay) is to use the highpass filter effect at a frequency of say 10Hz, as illustrated in the following example:

```
synth 5 sin %0 50
highpass 10
```

15.4 Fade

NAME

Fade – apply a fade-in and/or fade-out to the audio

SYNOPSIS

```
fade [type] fade-in-length [stop-position(=) [fade-out-length]]
```

FUNCTION

Apply a fade effect to the beginning, end, or both of the audio.

An optional `type` can be specified to select the shape of the fade curve: `q` for quarter of a sine wave, `h` for half a sine wave, `t` for linear ("triangular") slope, `l` for logarithmic, and `p` for inverted parabola. The default is logarithmic.

A fade-in starts from the first sample and ramps the signal level from 0 to full volume over the time given as `fade-inlength`. Specify 0 if no fade-in is wanted.

For fade-outs, the audio will be truncated at stop-position and the signal level will be ramped from full volume down to 0 over an interval of `fade-out-length` before the `stop-position`. If `fade-out-length` is not specified, it defaults to the same value as `fade-in-length`. No fade-out is performed if `stop-position` is not specified. If the audio length can be determined from the input file header and any previous effects, then -0 (or, for historical reasons, 0) may be specified for `stop-position` to indicate the usual case of a fade-out that ends at the end of the input audio stream.

Any time specification may be used for `fade-in-length` and `fade-out-length`.

See also the splice effect.

15.5 Gain

NAME

Gain – apply gain or attenuation; normalise/equalise/balance/headroom

SYNOPSIS

```
gain [-e|-B|-b|-r] [-n] [-l|-h] [gain-dB]
```

FUNCTION

Apply amplification or attenuation to the audio signal, or, in some cases, to some of its channels.

Without other options, `gain-dB` is used to adjust the signal power level by the given number of dB: positive amplifies (beware of clipping), negative attenuates. With other options, the `gain-dB` amplification or attenuation is (logically) applied after the processing due to those options.

Given the `-e` option, the levels of the audio channels of a multi-channel sample are "equalised", i.e. gain is applied to all channels other than that with the highest peak level, such that all channels attain the same peak level (but, without also giving `-n`, the audio is not "normalised").

The `-B` (balance) option is similar to `-e`, but with `-B`, the RMS level is used instead of the peak level. `-B` might be used to correct stereo imbalance caused by an imperfect record turntable cartridge. Note that unlike `-e`, `-B` might cause some clipping.

`-b` is similar to `-B` but has clipping protection, i.e. if necessary to prevent clipping whilst balancing, attenuation is applied to all channels. Note, however, that in conjunction with `-n`, `-B` and `-b` are synonymous.

The `-r` option is used in conjunction with a prior invocation of gain with the `-h` option - see below for details.

The `-n` option normalises the audio to 0dB FSD; it is often used in conjunction with a negative `gain-dB` to the effect that the audio is normalised to a given level below 0dB. For example:

```
gain -n
```

normalises to 0dB, and

```
gain -n -3
```

normalises to -3dB.

The `-l` option invokes a simple limiter, e.g.

```
gain -l 6
```

will apply 6dB of gain but never clip. Note that limiting more than a few dBs more than occasionally (in a piece of audio) is not recommended as it can cause audible distortion. See the compand effect for a more capable limiter.

The `-h` option is used to apply gain to provide head-room for subsequent processing. For example, with

```
gain -h bass +6
```

6dB of attenuation will be applied prior to the bass boosting effect thus ensuring that it will not clip. Of course, with bass, it is obvious how much headroom will be needed, but with other effects (e.g. rate, dither) it is not always as clear. Another advantage of using `gain -h` rather than an explicit attenuation, is that if the headroom is not used by subsequent effects, it can be reclaimed with `gain -r`, for example:

```
gain -h bass +6 rate 44100 gain -r
```

The above effects chain guarantees never to clip nor amplify; it attenuates if necessary to prevent clipping, but by only as much as is needed to do so.

Output formatting (dithering and bit-depth reduction) also requires headroom (which cannot be "reclaimed"), e.g.

```
gain -h bass +6 rate 44100 gain -rh dither
```

Here, the second gain invocation, reclaims as much of the headroom as it can from the preceding effects, but retains as much headroom as is needed for subsequent processing.

See also the norm and vol effects.

15.6 Loudness

NAME

Loudness – gain control with ISO 226 loudness compensation

SYNOPSIS

```
loudness [gain [reference]]
```

FUNCTION

Loudness control - similar to the gain effect, but provides equalisation for the human auditory system. See http://en.wikipedia.org/wiki/Loudness for a detailed description of loudness. The gain is adjusted by the given `gain` parameter (usually negative) and the signal equalised according to ISO 226 w.r.t. a reference level of 65dB, though an alternative reference level may be given if the original audio has been equalised for some other optimal level. A default gain of -10dB is used if a gain value is not given.

See also the gain effect.

15.7 Mcompand

NAME

Mcompand – multi-band compression/expansion/limiting

SYNOPSIS

```
mcompand "attack1,decay1{,attack2,decay2} [soft-knee-dB:]in-dB1
    [,out-dB1]{,in-dB2,out-dB2}[gain [initial-volume-dB [delay]]]"
    {crossover-freq[k]"attack1,..."}
```

FUNCTION

The multi-band compander is similar to the single-band compander but the audio is first divided into bands using Linkwitz-Riley cross-over filters and a separately specifiable compander run on each band. See the compand effect for the definition of its parameters. Compand parameters are specified between double quotes and the crossover frequency for that band is given by crossover-freq; these can be repeated to create multiple bands.

For example, the following (one long) command shows how multiband companding is typically used in FM radio:

```
gain -3 sinc 8000- 29 100 mcompand
"0.005,0.1 -47,-40,-34,-34,-17,-33" 100
"0.003,0.05 -47,-40,-34,-34,-17,-33" 400
"0.000625,0.0125 -47,-40,-34,-34,-15,-33" 1600
"0.0001,0.025 -47,-40,-34,-34,-31,-31,-0,-30" 6400
"0,0.025 -38,-31,-28,-28,-0,-25"
gain 15 highpass 22 highpass 22 sinc -n 255 -b 16 -17500
gain 9 lowpass -1 17801
```

See also compand for a single-band companding effect.

15.8 Norm

NAME

Norm – normalise to 0dB (or other)

SYNOPSIS

`norm [dB-level]`

FUNCTION

Normalise the audio. `norm` is just an alias for `gain -n`; see the gain effect for details.

15.9 Vol

NAME

Vol – adjust audio volume

SYNOPSIS

`vol gain [type [limitergain]]`

FUNCTION

Apply an amplification or an attenuation to the audio signal. The amount to change the volume is given by `gain` which is interpreted, according to the given type, as follows: if `type` is amplitude (or is omitted), then `gain` is an amplitude (i.e. voltage or linear) ratio, if power, then a power (i.e. wattage or voltage-squared) ratio, and if dB, then a power change in dB.

When `type` is amplitude or power, a `gain` of 1 leaves the volume unchanged, less than 1 decreases it, and greater than 1 increases it; a negative gain inverts the audio signal in addition to adjusting its volume.

When `type` is dB, a `gain` of 0 leaves the volume unchanged, less than 0 decreases it, and greater than 0 increases it.

Beware of clipping when the increasing the volume.

The `gain` and the `type` parameters can be concatenated if desired, e.g. vol 10dB.

An optional `limitergain` value can be specified and should be a value much less than 1 (e.g. 0.05 or 0.02) and is used only on peaks to prevent clipping. Not specifying this parameter will cause no limiter to be used. In verbose mode, this effect will display the percentage of the audio that needed to be limited.

See also gain for a volume-changing effect with different capabilities, and compand for a dynamic-range compression/ expansion/limiting effect.

Appendix A Licenses

A.1 LGPL license

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent

license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or

any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the

Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

Index

A

Allpass 49

B

Band 49
Bandpass 49
Bandreject 50
Bass 50
Bend 37
Biquad 27

C

Channels 35
Chorus 41
Compand 55
Contrast 56

D

DCShift 56
Deemph 47
Delay 41
Dither 29
Downsample 27

E

Earwax 47
Echo 41
Echos 42
Equalizer 50

F

Fade 57
Fir 27
Flanger 42

G

Gain 57

H

Highpass 51
Hilbert 51

L

Loudness 58
Lowpass 51

M

Mcompand 59

N

Noiseprof 17
Noisered 47
Norm 59

O

Oops 48
Overdrive 43

P

Pad 21
Phaser 43
Pitch 37

R

Rate 29
Remix 35
Repeat 44
Reverb 44
Reverse 44
Riaa 48

S

Silence 21
Sinc 51
soundfx.Generate 11
soundfx.Process 11
Speed 37
Splice 22
Stat 17
Stats 18
Stretch 38
Swap 36
Synth 33

T

Tempo 38
Treble 52
Tremolo 45
Trim 23

U

Upsample 28

V

Vad 23
Vol 60