NAME

out123 - send raw PCM audio or a waveform pattern to an output device

SYNOPSIS

cat audio.raw | **out123** [-] [*options*]

out123 [*options*] *filename* [*filename* ...]

out123 --wave-freq *freq1*[,*freq2*,...] [*options*]

out123 --source geiger [*options*]

DESCRIPTION

out123 reads raw PCM data (in host byte order) from standard input and plays it on the audio device specified by given options. Alternatively, it can generate periodic or random signals for playback itself.

OPTIONS

out123 options may be either the traditional POSIX one letter options, or the GNU style long options. POSIX style options start with a single '-', while GNU long options start with '--'. Option arguments (if needed) follow separated by whitespace (not '='). Note that some options can be absent from your installation when disabled in the build process.

--name *name*

Set the name of this instance, possibly used in various places. This sets the client name for JACK output.

-o *module*, **--output** *module*

Select audio output module. You can provide a comma-separated list to use the first one that works. Also see **-a**.

--list-modules

List the available modules.

--list-devices

List the available output devices for given output module. If there is no functionality to list devices in the chosen module, an error will be printed and out123 will exit with a non-zero code.

-a *dev*, **--audiodevice** *dev*

Specify the audio device to use. The default as well as the possible values depend on the active output. For the JACK output, a comma-separated list of ports to connect to (for each channel) can be specified.

-s, **--stdout**
The audio samples are written to standard output, instead of playing them through the audio device. The output format is the same as the input ... so in this mode, **out123** acts similar the standard tool **cat**, possibly with some conversions involved. This shortcut is equivalent to '-o raw -a -'.

-S, **--STDOUT**
This variant additionally writes the data to stdout, while still playing it on the output device. So it is more like some flavour of **tee** than a **cat**.

-O *file*, **--outfile**
Write raw output into a file (instead of simply redirecting standard output to a file with the shell). This shortcut is equivalent to '-o raw -a *file*'.

-w *file*, **--wav**
Write output as WAV file *file* , or standard output if - is or the empty string used as file name. You can also use *--au* and *--cdr* for AU and CDR format, respectively. Note that WAV/AU writing to non-seekable files or redirected stdout needs some thought. The header is written with the first actual data. The result of decoding nothing to WAV/AU is a file consisting just of the header when it is seekable and really nothing when not (not even a header). Correctly writing data with prophetic headers to stdout is no easy business. This shortcut is equivalent to '-o wav -a *file*'.

--au *file*
Write to *file* in SUN audio format. If - or the empty string is used as the filename, the AU file is written to stdout. See paragraph about WAV writing for header fun with non-seekable streams. This shortcut is equivalent to '-o au -a *file*'.

--cdr *file*
Write to *file* as a CDR (CD-ROM audio, more correctly CDDA for Compact Disc Digital Audio). If - is or the empty string used as the filename, the CDR file is written to stdout. This shortcut is equivalent to '-o cdr -a *file*'.

-r *rate*, **--rate** *rate*
Set sample rate in Hz (default: 44100). If this does not match the actual input sampling rate, you get changed pitch. Might be intentional;-)

-R *rate*, **--inputrate** *rate*
Set input sample rate to a different value. This triggers resampling if the output rate is indeed different. See **--resample**.

--speed *factor*
Speed up/down playback by that factor using resampling. See **--resample**.

--resample *method*
This chooses the method for resampling between differing sampling rates or to apply a change in tempo. You can choose between two variants of the syn123 resampler: fine (the default) and dirty. The fine one features 108 dB dynamic range and at worst-case 84% bandwidth. The dirty one uses a bit less CPU time (not that much, though) by reducing the dynamic range to 72 dB with worst-case bandwidth of 85%. The exact properties vary with the sampling rate ratio, as there is interpolation of filter coefficients involved.

-c *count*, **--channels** *count*
Set channel count to given value.

-C *count*, **--inputch** *count*
Set input channel count to a differnt value than for output. This probably means you want some remixing. Also see **--mix**.

-e *enc*, **--encoding** *enc*
Choose output sample encoding. Possible values look like f32 (32-bit floating point), s32 (32-bit signed integer), u32 (32-bit unsigned integer) and the variants with different numbers of bits (s24, u24, s16, u16, s8, u8) and also special variants like ulaw and alaw 8-bit. See the output of **out123**'s longhelp for actually available encodings. Default is s16.

--endian *choice*
Select output endianess (byte order). Choice is big, little, or native, which is the default. The processing can only work in native mode, so you need to specify input or output byte order if that does not match your machine. This also sets the input endianess if that is not set separately. See also **--inputend** and **--byteswap**.

-E *enc*, **--inputenc** *enc*
Specify input encoding different from output encoding for conversion.

--inputend *choice*
Select input endianess (byte order). By default it is the same as output byte order. See **--endian**.

--byteswap
A switch to trigger swapping of byte order just before output, after any other transformations. This works on top of any endianess you specify with

-m, **--mono**
Set for single-channel audio (default is two channels, stereo).

--stereo
Select stereo output (2 channels, default).

--list-encodings
List known encoding short and long names to standard output.

--mix *matrix*
Specify a mixing matrix between input and output channels as linear factors, comma separated list for the input channel factors for output channel 1, then output channel 2, and so forth. The default is a unit matrix if channel counts match, so for 3 channels the equivalent of both channels with halved amplitude, so '--mix 0.5,0.5'. For splitting mono to stereo, it is '--mix 1,1' top keep the symmetry.

--filter *coeff*
Apply digital filter(s) before pre-amplification (see **--preamp**) with the coefficient list coeff as
b_0,...,b_N,a_0,...,a_N
where a_0=1 is mandatory and perhaps helps orientation a bit. Multiple filters are separated by ':'.

-P *dbvalue* **--preamp** *dbvalue*
Enable a pre-amplification stage that amplifies the signal with the given value in dB before output.

--offset *value*
Apply a PCM offset (floating point value scaled in [-1:1] in the pre-amplification stage. Normally, you would do that to correct a known DC offset in a recording.

--clip *mode*
Select clipping mode: 'soft' or 'hard' for forced clipping also for floating point output, 'implicit' (default) for implied hard clipping during conversion where necessary.

--dither
Enable dithering for conversions to integer. If you insist. This is just some un-spectacular TPDF dither. For some people, that is not fancy enough. Most people cannot be bothered that way or the other.

--test-format
Check if given format is supported by given driver and device (in command line before encountering this), silently returning 0 as exit value if it is the case.

--test-encodings
Print out the short names of encodings supported with the current setup.

--query-format
If the selected driver and device communicate some default accepted format, print out a command line fragment for **out123** setting that format, always in that order: --rate <r> --channels <c> --encoding <e>

-o h, **--headphones**
Direct audio output to the headphone connector (some hardware only; AIX, HP, SUN).

-o s, **--speaker**
Direct audio output to the speaker (some hardware only; AIX, HP, SUN).

-o l, **--lineout**
Direct audio output to the line-out connector (some hardware only; AIX, HP, SUN).

-b *size*, **--buffer** *size*
Use an audio output buffer of *size* Kbytes. This is useful to bypass short periods of heavy system activity, which would normally cause the audio output to be interrupted. You should specify a buffer size of at least 1024 (i.e. 1 Mb, which equals about 6 seconds of usual audio data) or more; less than about 300 does not make much sense. The default is 0, which turns buffering off.

--preload *fraction*
Wait for the buffer to be filled to *fraction* before starting playback (fraction between 0 and 1). You can tune this prebuffering to either get sound faster to your ears or safer uninterrupted web radio. Default is 0.2 (changed from 1 since version 1.23).

--devbuffer *seconds*
Set device buffer in seconds; <= 0 means default value. This is the small buffer between the application and the audio backend, possibly directly related to hardware buffers.

--timelimit *samples*
Set playback time limit in PCM samples if set to a value greater than zero. **out123** will stop reading from stdin or playing from the generated wave table after reaching that number of samples.

--seconds *seconds*
Set time limit in seconds instead.

--source *name*
Choose the signal source: ’file’ (default) for playback of the given file(s) on the command line or standard input if there are none, or one of the generators ’wave’ (see **--wave-freq**), geiger (see **--geiger-activity**), or just ’white’ for some white noise.

--wave-freq *frequencies*
Set wave generator frequency or list of those with comma separation for enabling a generated test signal instead of standard input. Empty values repeat the previous one.

--wave-pat *patterns*
Set the waveform patterns of the generated waves as comma-separated list. Choices include sine, square, triangle, sawtooth, gauss, pulse, and shot. Empty values repeat the previous one.

--wave-phase *phases*
Set waveform phase shift(s) as comma-separated list, negative values inverting the pattern in time and empty value repeating the previous. There is also **--wave-direction** overriding the negative bit.

--wave-direction
Set wave direction explicitly (the sign counts).

--wave-sweep *frequency*
Sweep a generated wave to the given frequency, from first one specified for **--wave-freq**, using the first wave pattern and direction, too.

--sweep-time *seconds*
Set frequency sweep duration in seconds if > 0. This defaults to the configured time limit if set, otherwise one second, as endless sweeps are not sensible.

--sweep-count *count*
Set timelimit to exactly produce that many (smooth) sweeps

--sweep-type *type*
Set sweep type: lin(ear) for linear, qua(d) (default) for quadratic, or exp(onential) for an exponential change of frequency with time.

--sweep-hard
Disable post-sweep smoothing for periodicity.

--genbuffer *bytes*
Set the buffer size (limit) for signal generators, if > 0 (default), this enforces a periodic buffer also for non-periodic signals, benefit: less runtime CPU overhead, as everything is precomputed as enforced periodic signal.

--wave-limit *samples*
This is an alias for **--genbuffer**.

--pink-rows *number*
Activate pink noise source and choose rows for the algorithm (<1 chooses default). The generator follows code provided by Phil Burk (http://softsynth.com) and uses the Gardner method.

--geiger-activity *number*
This configures the simulation of a Geiger-Mueller counter as source, with the given numer as average events per second. Play with it. It’s fun!

-t, **--test**
Test mode. The audio stream is read, but no output occurs.

-v, **--verbose**
Increase the verbosity level.

-q, **--quiet**
Quiet. Suppress diagnostic messages.

--aggressive
Tries to get higher priority

-T, **--realtime**
Tries to gain realtime priority. This option usually requires root privileges to have any effect.

-?, **--help**
Shows short usage instructions.

--longhelp
Shows long usage instructions.

--version
Print the version string.

AUTHORS

Maintainer:

Thomas Orgis <maintainer@mpg123.org>, <thomas@orgis.org>

Creator (ancestry of code inside mpg123):

Michael Hipp

Uses code or ideas from various people, see the AUTHORS file accompanying the source code.

LICENSE

WEBSITE

http://www.mpg123.org

http://sourceforge.net/projects/mpg123