CDP SNDINFO Functions

(with Command Line Usage)

Functions to get Information about soundfiles

NB: Beware of conflict with *Csound* function with the same name.

(Names in brackets mean that these are separate programs. The others are sub-modules of EXTEND.)

CHANDIFF
Compare channels in a stereo soundfile

DIFF
Compare two sound, analysis, pitch, transposition, envelope or formant files

FINDHOLE
Find largest low level hole in a soundfile

LEN
Display duration of a soundfiling-system file

LENS
List durations of several soundfiling- system files

LOUDCHAN
Find loudest channel in a stereo soundfile

MAXI
List levels of several soundfiles

MAXSAMP
Find maximum sample in a soundfile or binary data file

MAXSAMP2
Find maximum sample within a specified timerange in a soundfile or binary data file

[PEAKFIND]
Find times of loudness peaks in a sound and output as a datafile

PRNTSND
Print sound sample data to a textfile

PROPS
Display properties of soundfiling-system file

SMPTIME
Convert sample count to time in soundfile

SUMLEN
Sum durations of several soundfiling- system files

TIMEDIFF
Find difference in duration of two soundfiles

TIMESMP
Convert time to sample count in soundfile

UNITS
Convert between different units

ZCROSS
Display fraction of zero-crossings in a soundfile

Also see:

RETIME Mode 12

Find the start of the sound in the file (the 1st non-zero sample)

RMSINFO

Scan file and report RMS and average power level statistics

SFPROPS

Display soundfile details, with WAVE-EX speaker positions

SNDINFO CHANDIFF – Compare channels in a stereo soundfile

Usage

sndinfo chandiff *infile* [**-t***threshold*] [**-n***count*]

Parameters

infile – input stereo soundfile
-t*threshold* – maximum permissible difference in data values (Range: 0.0 to 1.0)
-n*count* – maximum number of differences to accept (Default: 1)

Understanding the SNDINFO CHANDIFF Function

This function is a simple utility to compare the two channels of a stereo soundfile sample by sample.

Musical Applications

The presence of differences between the channels will indicate that it is a true stereo file: i.e., some differences in data. If the two channels are identical – report no differences – it will really be a mono source which has been 'spread' across two channels.

End of SNDINFO CHANDIFF

SNDINFO DIFF – Compare two sound, analysis, pitch, transposition, envelope or formant files

Usage

sndinfo diff *infile1 infile2* [**-t***threshold*] [**-n***count*] [**-l**] [**-c**]

Parameters

infile1 – name of first file to check (binary non-text files only)
infile2 – name of file to compare with the first
-t*threshold* – maximum permissible difference in data values (Range: 0.0 to 1.0)
-n*count* – maximum number of differences to accept (Default: 1)
-l – continue, even if the files are not the same length
-c – continue, even if the (sound)files do not have the same number of channels

Understanding the SNDINFO DIFF Function

SNDINFO DIFF compares the data in the two input files and reports any differences. The value given for *threshold* will be determined by the type of files being compared: e.g., amplitude in a 0 to 1 range.

Musical Applications

Sometimes we make copies of soundfiles and give the copy a new name. Later, we forget whether or not the two files are really the same. SNDINFO DIFF provides the answer.

Note that SNDINFO DIFF can compare a variety of files in a binary, non-text format: soundfiles, analysis files, pitch data files (from **REPITCH GETPITCH**) etc.

End of SNDINFO DIFF

SNDINFO FINDHOLE – Find largest low level hole in a soundfile

Usage

sndinfo findhole *infile* **-t***threshold*

Parameters

infile – input soundfile to search
-t*threshold* – amplitude value: consider there to be a hole only if the level falls and stays below *threshold* (Range: floating point value: 0.0 < *threshold* <= 1.0) (Default: 0.0)

Understanding the SNDINFO FINDHOLE Function

Reports the length of the hole and the time at which it begins. If it displays the message 'Maximum holesize 0.0 at time 0.0', try raising the threshold.

Musical Applications

This utility enables us to cut out areas of low signal in a soundfile without having to access a soundfile viewer. It helps to make the command line environment more practical. The presence of the basic editing facilities (cut, paste (insert), splice and mix in command line form also supports our partially-sighted users.

End of SNDINFO FINDHOLE

SNDINFO LEN – Display duration of a soundfiling-system file

Usage

sndinfo len *infile*

Parameters

infile – sound, analysis or binary file input

> **NB:** You need to enter the file extension because of the various extensions which are 'legal'. The main part of the name may be the same, but the extension differ, such as 'afile.wav' and 'afile.ana' and 'afile.frq'. SNDRPT LEN needs to know which one you want information about. However, PVPLAY will happily find and play the soundfile among these three, even without specifying the extension as long as CDP_SOUND_EXT is set.

Understanding the SNDINFO LEN Function

Displays the type of file, the duration in seconds, the number samples and bytes for a soundfile, the number of windows, floats and bytes for an analysis or analysis-derived file.

Musical Applications

SNDINFO LEN provides a quick check on length for a specific soundfile. I have used this most often when preparing breakpoint files or mixing (Ed.).

DIRSF will provide the same information and is useful when a little unsure about the name: you can get close with wildcards (e.g., 'cr*.wav' to see all .wav files beginning with 'cr') and then spot the one you want in the list displayed.

End of SNDINFO LEN

SNDINFO LENS – List durations of several soundfiling-system files

Usage

sndinfo lens *infile* [*infile2 ...*]

Parameters

infile , *infile2 ...* – soundfiles to display length of

Unlike SNDINFO LEN, SNDINFO LENS only seems to handle soundfiles.

Understanding the SNDINFO LENS Function

Here you see a display of the soundfile name and its duration (only).

Musical Applications

You are able to check on several (sound)files at once.

End of SNDINFO LENS

SNDINFO LOUDCHAN – Find loudest channel in a stereo soundfile

Usage

sndinfo loudchan *infile*

Parameters

infile – stereo soundfile to check

Understanding the SNDINFO LOUDCHAN Function

Displays which channel is loudest.

Musical Applications

Knowing which channel is loudest can help in setting levels in the mixfile. If stereo files are present and there is some slight overload, it may be useful to check which specific channel might be causing the problem.

End of SNDINFO LOUDCHAN

SNDINFO MAXI – List levels of several soundfiles

Usage

sndinfo maxi *infile infile2* [*infile3 ...*] *outtextfile*

Parameters

infile infile2, [*infile3 ...*] – find the amplitude levels of two soundfiles (minimum) or more
outtextfile – write the soundfile names and their levels to a text file (not optional)

Understanding the SNDINFO MAXI Function

SNDINFO MAXI lists levels within the normalised range of 0 to 1, e.g., 0.421491 and 0.717514, i.e., in reference to 1 as full volume (0 dB). It displays both the soundfile names and their levels on the screen and writes this information to a (mandatory) text file.

You can convert these values to dB using Mode 24 of **SNDINFO UNITS**. E.g., 0.421491 is -7.504dB and 0.717514 is -2.896dB. Note that .7 is closer to 1 than 0.4 and therefore a higher level. Also see the **Gain-to-dB Chart**.

Musical Applications

You are able to check on several soundfiles at once. This can be useful for example to compare a soundfile to which gain has been applied and the original.

End of SNDINFO MAXI

SNDINFO MAXSAMP – Find maximum sample in a soundfile or binary data file

Usage

sndinfo maxsamp *infile* [**-f**]

Parameters

infile – input soundfile file or binary data file to check
-f – force file to be scanned (ignoring any information in the header about max sample which may already be present)

Understanding the SNDINFO MAXSAMP Function

SNDINFO MAXSAMP provides a good deal of useful information. Here is a typical output and what it all means:

SNDINFO MAXSAMP Display, with Commentary

Info labels	Info values	Commentary
maximum abs value:	32380	Value belongs to the 0-32767 amplitude range
maximum level:	-0.1032dB	0 dB is full volume, so this will be just under
at:	1758 samples	Sample location of the sample with the highest level
time:	0.0399 sec	Corresponding time location of the sample with the highest level
repeated:	1 times	Number of times this highest value occurs in the file
max possible gain:	1.0120	Gain factor which could be used to 'normalise' the sound, i.e., to bring it to full volume – values less than 1 reduce volume
max dB gain:	0.1032	You might want to use a dB value in a mixfile

This information is also written to the header of a CDP soundfile, for possible future use by the system. When using SNDINFO MAXSAMP again on the same file, it will just read the header unless you force another look at the sound data in the file by using **-f**.

Musical Applications

Sometimes you just want the sound to be a bit louder (or softer). Or the Phase Vocoder may report a bit of overload that needs to be reduced. Or some process has introduced overload and you need to start with a slightly softer input. Use **MODIFY LOUDNESS** to make the changes.

It is wise to apply gain to a (digital) soundfile with discretion and only as needed. Being too heavy-handed, such as applying a large amount of gain to a rather soft sound, can introduce distortion artefacts.

End of SNDINFO MAXSAMP

SNDINFO MAXSAMP2 – Find maximum sample within a specified timerange in a soundfile

Usage

sndinfo maxsamp2 *infile start end*

Parameters

infile – input soundfile file to check
start – start time of search in file
end – end time of search in file

Understanding the SNDINFO MAXSAMP2 Function

This function returns information about maximum amplitude and recommended gain factor as does **SNDINFO MAXSAMP** above, but enables you to specify a time period in the soundfile that you would like to check.

End of SNDINFO MAXSAMP2

PEAKFIND – Find times of loudness peaks in a sound and output as a datafile

Usage

peakfind peakfind *infile outtextfile windowsize* [**-t***threshold*]

Example command line to create a peaks datafile :

```
peakfind peakfind count countpeaks.txt 200
```

Parameters

infile – input soundfile
outfile – output text data file
windowsize – size, in milliseconds, of the window used for locating succesive peaks (Range: 1 to 500)
-t*threshold* – the level below which peaks are ignored in any window (Range 0-1).

Understanding the PEAKFIND Function

This function was originally developed to track the syllables in speech.

It outputs a textfile, with a list of the times of the peaks found. Only the times of the peaks are shown.

A very large *windowsize* will miss detail in the sound. A very small *windowsize* will confuse the waveform's amplitude variation with the loudness envelope and produce an excessive number of 'peak' times. In the example command line above, the input soundfile, *count.wav* is 8.065986 seconds long and consists of a spoken count from 1 to 10. A fairly large *windowsize* in order to focus on the the main peaks in each counted number. This was the output:

```
0.165465
1.066667
2.001655
2.672562
3.530340
4.650567
4.816236
5.458549
5.666145
6.135329
6.549093
7.035646
7.765986
```

The default of the *threshold* parameter is to ignore peaks whose level is less than one-fifth of the **local** maximum level. The local maximum is measured over 10 windows.

Trevor Wishart explains:

> "In a complex signal there can be lots of 'peaks' i.e., localised moments where the signal is slightly louder than the preceding and following windows, but we're really interested in events which stand out in loudness, and are therefore heard as rhythmic 'accents' or 'downbeats' in the event. The algorithm therefore picks out the local maximum peak, then ignores local fluctuations in level (producing localised maxima) which are close by and much lower in level than this peak.
>
> If the threshold is > 0.0, the algorithm works in the same way, but if any of the the local peaks it finds lie below some absolute level (i.e., the *threshold* which you have set), they are ignored."

I may have focused even more on the peaks in the numbers with *windowsize* = 100 and *threshold* = 0.4. This is the output:

```
0.165465
1.066667
2.001655
2.672562
3.530340
4.650567
5.458549
6.135329
7.035646
7.765986
```

Musical Applications

PEAKFIND is useful for obtaining rhythmic information about musical materials, especially for use with the **RETIME** functions. The data produced can also be analysed, for example to find tempo, in the **TABLE EDITOR** in *Sound Loom*.

End of PEAKFIND

SNDINFO PRNTSND – Print sound sample data to a textfile

Usage

sndinfo prntsnd *infile outtextfile starttime endtime*

Parameters

infile – input soundfile
outtextfile – textfile generated – **CARE!!! large amounts of data**
starttime – time in *infile* at which to begin printing data
endtime – time in *infile* at which to end printing data

Understanding the SNDINFO PRNTSND Function

The textfile generated by the function contains data for every sample within the time specified in the form *[sample_count] amplitude level*. Each value is displayed on a separate line.

You will therefore have sample_rate number of samples for every second of sound. At a sample rate of 44100, even 1/100th of a second will contain 441 values per channel.

Musical Applications

I think more than anything else, this little utility shows what can be done. Anyone wanting to develop this towards a more particular application or type of display is free to have the source code!

End of SNDINFO PRNTSND

SNDINFO PROPS – Display properties of a CDP system file

Usage

sndinfo props *infile*

Parameters

infile – input soundfile, or other CDP system file.

Understanding the SNDINFO PROPS Function

SNDINFO PROPS displays 'header' information for several CDP file types:

- Soundfiles (**.wav**, **.aif**)
- Analysis files (**.ana**)
- Binary pitch files (**.frq**)
- Binary transposition files (**.trn**)
- Binary formant files (**.for**)
- Binary envelope files (**.evl**)

The 'properties' are the key defining features of a soundfile. They usually include:

- its size in bytes and samples,
- the type of number used to store the sample data (SHORT or FLOAT),
- the sample rate (number of samples per second),
- and the number of channels.

A soundfile sample is typically stored as a 16-bit integer or as a 32-bit floating-point number. A byte is 8-bits. This is why there are twice as many bytes as there are samples in a 16-bit soundfile. An analysis file stores its data floating-point numbers, with 4 bytes per sample.

This information is written in the 'header' of the file, a text portion that precedes the data. This information can be altered with **HOUSEKEEP RESPEC**. A CDP header may also have some other information, as we have seen with **SNDINFO MAXSAMP**, but the structure and use of the header conforms to the .wav standard.

Many CDP functions 'interrogate' the header and make use of the information it contains. This for example, is how **SUBMIX DUMMY** puts the correct number of channels in the file it creates.

The following is a typical display, with commentary, for an analysis file (the display for a soundfile is like the list of key features above):

SNDINFO PROPS Analysis File Display, with Commentary

Info labels	Info values	Commentary
bytes:	2860488	Number of bytes at 4 bytes per sample
samples:	715122	Sample rate * number of seconds
sample rate:	344	Really the frame rate – see the PVOC Reference Manual
channels:	1026	The default value for **-N** (*points*) + 2 extra floats for the zero Hz channel in each window
original sample type:	16bit SHORTS	The sample type of the source soundfile
original sample rate:	44100	Sample rate of the source soundfile
analysis rate:	344.531250	The precise frame rate (see above)
analysis window len:	1024	Number of soundfile samples analysed, per window
decimation factor:	128	Number of samples to step from start of one analysis window to start of next (Default: 1/8 window-length)

Musical Applications

SNDINFO PROPS provides a quick confirmation of the header information in a soundfile. A typical use for SNDINFO PROPS would be before and after HOUSEKEEP RESPEC.

ALSO SEE the complementary function **SFPROPS**.

End of SNDINFO PROPS

SEARCH SIGSTART– Find earliest time at which there is signal in two or more soundfiles.

Usage

search sigstart *infile1 infile2 [infile3...]*

Parameters

infile1, infile2 [infile3...] – input soundfiles to search
Files must have same channel-count.

Understanding the SEARCH SIGSTART Function

SIGSTART compares the infiles and reports (to the console) the earliest start-time in the set, when there is some signal. The report does not say which of the sounds starts first.

Musical Applications

When mixing files or creating a mixfile, it may be helpful to know when some signal first appears, in order to discard any initial silence before or after mixing, or adjust the starting point of some or all of the files, for example, by prefixing further silence. Note that **RETIME Mode 12** finds the start of the sound in a single soundfile, or it can be run a number of times with different soundfiles to add this information to the same report file.

End of SEARCH SIGSTART

SNDINFO SMPTIME – Convert sample count to time in soundfile

Usage

sndinfo smptime *infile sampcount* [**-g**]

Parameters

infile – input soundfile
sampcount – number of samples (an integer)
-g – the sample count is taken as a count of **multi-channel** samples. For example, in a stereo file, sample-**pairs** are counted.

Understanding the SNDINFO SMPTIME Function

There may be occasions when you would like to know how much time a given number of samples comprises. This function provides the answer in seconds, to 6 decimal places of precision.

You need to specify an *infile* because the number of samples used for a given length of time differs according to sample rate and number of channels. The function looks at the header of *infile* to get the information it needs to make the calculation – saving you a bit of time.

If your value for *sampcount* is greater than the number of samples in the file, an 'out of range' error message is issued, showing as the maximum value the number of samples in the file.

The reverse operation, from time to samples, is handled by SNDINFO TIMESMP.

Musical Applications

Most software which displays soundfiles will show time both in seconds and as a sample count. But if you come across one which just uses sample count, this will convert to the equivalent time in seconds.

End of SNDINFO SMPTIME

SNDINFO SUMLEN – Sum durations of several soundfiling-system files

Usage

sndinfo sumlen *infile infile2* [*infile3 ...*] **-s***splicelen*

Parameters

infile infile2 – at least two soundfiles. **They must have the same sample rate and the same number of channels**.
infile3 ... – possible additional soundfiles
-s*splicelen* – length of splice in milliseconds. (Default: 15ms).

Understanding the SNDINFO SUMLEN Function

Unless using butt edits exclusively, splicing soundfiles together involves overlapping the files by the length of the splices. Thus the total duration of the spliced files is not the same as the total duration of all the files taken individually.

SNDINFO SUMLEN will tell you what the total duration of the spliced files is, taking into account the splice overlaps. If a *splice* length is not specified, it will use the Default 15ms. You can enter a value of 0 for *splice* and get a total duration with no overlaps: i.e., for all the files taken individually

Musical Applications

When splicing and mixing, you sometimes want to have an idea how long a certain combinations of soundfiles will become. This function will tell you that without your having to look up each file individually.

One restriction is that files with different numbers of channels can be mixed, but their durations cannot be summed by SNDINFO SUMLEN.

End of SNDINFO SUMLEN

SNDINFO TIMEDIFF – Find difference in duration of two soundfiles

Usage

sndinfo timediff *infile1 infile2*

Parameters

infile1 infile2 – the two input soundfiles

Understanding the SNDINFO TIMEDIFF Function

The inputs must be soundfiles. The function displays the amount of time difference there is between their respective durations. It does this calculation based on information it finds in the headers of the soundfiles.

Musical Applications

Suppose you wanted to offset a file in a mix, so that it ended at the same time as another file. SNDINFO TIMEDIFF would tell you the size of the offset. It might also be relevant when using the *stagger* option of **MORPH MORPH**.

End of SNDINFO TIMEDIFF

SNDINFO TIMESMP – Convert time to sample count in a soundfile

Usage

sndinfo timesmp *infile time* [**-g**]

Parameters

infile – input soundfile t
time – (floating point) time value in seconds
-g – the sample count displayed is taken as a count of **multi-channel** samples. For example, in a stereo file, sample-**pairs** are counted.

Understanding the SNDINFO TIMESMP Function

There may be occasions when you would like to know how many samples are contained in a given length of time in a soundfile. This function provides the answer.

You need to specify an *infile* because the number of samples used for a given length of time differs according to sample rate and number of channels. The function looks at the header of *infile* to get the information it needs to make the calculation – saving you a bit of time.

A stereo soundfile has twice as many samples per unit time as does a mono soundfile. For example 1.52 seconds in a mono soundfile at a sample rate of 44100 comprises 67032 samples, but 1.52 seconds in a stereo soundfile at the same sample rate comprises 134064 samples. With the **-g** flag, the two channels are 'grouped' (treated as a single pair), so SNDINFO TIMESMP will display 67032.

The reverse operation, from samples to time, is handled by **SNDINFO SMPTIME**.

Musical Applications

Most software which displays soundfiles will show time both in seconds and as a sample count. But if you come across one which just uses time, and for some reason you need to know the sample count equivalent to this duration, SNDINFO TIMESMP will display the number of samples.

End of SNDINFO TIMESMP

SNDINFO UNITS – Convert between different units

Usage

sndinfo units mode *value*

Modes

The 25 Modes of SNDINFO UNITS Grouped into Categories

PITCH	INTERVAL	SPEED	LOUDNESS
1 MIDI to FRQ	**7** FRQ RATIO to SEMITONES	**16** FRQ RATIO to TIME RATIO	**24** GAIN FACTOR to DBGAIN
2 FRQ to MIDI	**8** FRQ RATIO to INTERVAL	**17** SEMITONES to TIME RATIO	**25** DBGAIN to GAIN FACTOR
3 NOTE to FRQ	**9** INTERVAL to FRQ RATIO	**18** OCTAVES to TIME RATIO	
4 NOTE to MIDI	**10**SEMITONES to FRQ RATIO	**19**INTERVAL to TIME RATIO	
5 FRQ to NOTE	**11**OCTAVES to FRQ RATIO	**20**TIME RATIO to FRQ RATIO	
6 MIDI to NOTE	**12** OCTAVES to SEMITONES	**21** TIME RATIO to SEMITONES	
	13 FRQ RATIO to OCTAVES	**22** TIME RATIO to OCTAVES	
	14 SEMITONES to OCTAVES	**23** TIME RATIO to INTERVAL	
	15 SEMITONES to INTERVAL		

Parameters

value – expression representing the unit to be converted. The conventions for representing *value* are shown in the table below.

Valid ways to represent the *values* given to SNDINFO UNITS

Type of Value	Representation Conventions
NOTE	A0 = A, in Octave 0 (octave of Middle-C); A1 = A, in Octave 1 (octave of High-C); A-1 = A, in Octave -1 (octave of Low-C); Ebu4 = E-flat + Up a ¼-tone, in Octave 4 F#d-2 = F-sharp - Down a ¼-tone, in Octave -2

Type of Value	Representation Conventions
INTERVAL	3 = a (major) third **up** -m3 = a minor third **down** m3u = a minor third plus Up a ¼-tone #4d = a tritone minus Down a ¼-tone 15 = a 15th (largest interval permitted)
MIDI	Integer representing a MIDI note-value
FRQ	A frequency in Hz
FRQ RATIO	A fractional multiplier used to transpose a frequency (see chart below)
SEMITONES	Number of semitones and fractions of semitones
OCTAVES	Number of octaves and fractions of octaves
TIME RATIO	Timestretch/compression = reciprocal of the frequency ratio: period = 1/frq_ratio E.g., twice the frequency will be ½ the duration: 1/2 = 0.5
GAIN FACTOR	Amplitude multiplier: each integer above zero represents an increase of ca. 3dB Fractions reduce amplitude: 0.7 is down ca. 3dB, 0.5 is down ca. 6dB etc.
DB GAIN	Amplitude expressed in dB, the logarithmic scale in which every 6dB represents a doubling or halving of the amplitude – the range of hearing is enormous!

Understanding the SNDINFO UNITS Function

References to octaves can be confusing. Note that here 'Octave 0' is the middle octave on the keyboard (Middle-C) and that Octave -1 means the octave below this. When entering 'Notes' in the form A-1, for example, this would be the A below Middle-C. The Middle-C octave is represented by 8.xx in *Csound*, and one also comes across Middle-C referred to as C3, probably because it's the 3rd octave on most piano keyboards. I've made up a handwritten chart for myself which gathers together these octave differences, together with the note name and MIDI note number into boxes for each note over the several octaves I use most of the time. (Ed.)

The precision of the units may sometimes seem extreme, but are usually more significant than they at first appear. Frequencies and amplitudes cover a very wide numerical range, and rounding errors usually do need to be minimised. You will notice that even with 6 decimal places, corresponding conversions don't come out exactly the same.

Musical Applications

Exploring some of the conversions, such as Gain Factor & dB Gain, can be very instructive. There are a couple of charts below for quick reference in areas I tend to use frequently. You can use the program to work out your own charts for values you find you often require. We really ought to have a book of charts of various kinds at our fingertips, just as mathematicians used to have log tables before the advent of the calculator, e.g.,:

- the notes chart described above
- a table of frequencies corresponding to equal-(or other)-tempered scales
- a chart showing how (the logarithmic scale of) dB levels relates to sounds in the environment
- a chart of transposition ratios (start of one is shown below)
- a chart showing how gain factors relate to dB (start of one is shown below)

We can save a lot of time by preparing some tools for working in the electroacoustic medium. SNDINFO UNITS gives us answers for many of the specific questions which will inevitably come up.

Transposition Ratios Down/Up for 24 Semitones

Semitones Down	Transposition Ratio	Semitones Up	Transposition Ratio	Name of Interval
-1	0.94387431	**1**	1.05946309	Semitone [**(-)m2**]
-2	0.89089872	**2**	1.12246205	Whole Tone [**(-)2**]
-3	0.84089642	**3**	1.18920712	Minor Third [**(-)m3**]
-4	0.79380053	**4**	1.25992105	Major Third [**(-)3**]
-5	0.74915354	**5**	1.33483985	Perfect Fourth [**(-)4**]
-6	0.70710678	**6**	1.41421356	Tritone [**(-)#4**]
-7	0.66741993	**7**	1.49830708	Perfect Fifth [**5**]
-8	0.629996052	**8**	1.58740105	Minor Sixth [**(-)m6**]
-9	0.59460356	**9**	1.68179283	Major Sixth [**(-)6**]
-10	0.56123102	**10**	1.78179744	Minor Seventh [**(-)m7**]
-11	0.52973155	**11**	1.88774863	Major Seventh [**(-)7**]
-12	0.50000000	**12**	2.0	Octave [**(-)8**]
-13	0.47193716	**13**	2.11892619	Minor Ninth [**(-)m9**]
-14	0.44544936	**14**	2.24492410	Ninth [**(-)9**]
-15	0.42044821	**15**	2.37841423	Minor Tenth [**(-)m10**]
-16	0.39685026	**16**	2.51984210	Tenth [**(-)10**]
-17	0.37457677	**17**	2.66968971	Eleventh [**(-)11**]

Semitones Down	Transposition Ratio	Semitones Up	Transposition Ratio	Name of Interval
-18	0.35355339	**18**	2.82842712	Octave + Tritone [**(-)#11**]
-19	0.33370996	**19**	2.99661415 (3.0)	Twelfth [**(-)12**]
-20	0.31498026	**20**	3.17480210	Minor Thirteenth [**(-)m13**]
-21	0.29730178	**21**	3.36358566	Thirteenth [**(-)13**]
-22	0.28061551	**22**	3.56359487	Minor Fourteenth [**(-)m14**]
-23	0.26486577	**23**	3.77549725	Fourteenth [**(-)14**]
-24	0.25	**24**	4.0	Fifteenth (2 Octaves) [**(-)15**]

And here is a table relating gain factors, dB and the 0 to 32767 amplitude scale:

Getting a Handle on dB values

Gain Increase Factor	dB Alteration	Gain Reduction Factor	dB Alteration	Reduction on 32767 scale
1.0	**0.0**dB	**1.0**	**0.0**dB	32767
1.5	**3.5**dB	**0.7**	**-3.1**dB	22936
2.0	**6.0**dB	**0.5**	**-6.0**dB	16383
2.5	**8.0**dB	**0.4**	**-8.0**dB	13106
3.0	**9.5**dB	**0.35**	**-9.1**dB	11468
3.5	**10.9**dB	**0.3**	**-10.5**dB	9830
4.0	**12.0**dB	**0.25**	**-12.0**dB	8191
8.0	**18.1**dB	**0.125**	**-18.1**dB	4096
16	**24.1**dB	**0.0625**	**-24.1**dB	2047
32	**30.1**dB	**0.03125**	**-30.1**dB	1024
64	**36.1**dB	**0.015626**	**-36.1**dB	512
128	**42.1**dB	**0.0078125**	**-42.1**dB	256
0.0	**-96**dB (i.e., zero level)			0

The gain factor is a ratio, an increase or decrease relative to 1 (full volume). The 0 to 32767 scale also covers the range between zero level and full volume. (Most or all CDP levels are now expressed on a scale of 0-1). One can't go higher than this without causing distortion. The above chart tells us that a gain factor of 0.5 is equivalent to -6dB, which halves the volume. Applied to the 32767 scale, this is 16363.5, so this gives some idea what the numbers along the 0 to 32767 scale really mean. The 5th column in the table above lists the amplitude *reductions* along this scale.

As a rule of thumb, given_level/32767 = the gain factor being applied. SNDINFO UNITS can convert this into dB for you. For example, if you set a 0 to 32767 level at 20000, the gain factor is 20000/ 32767 = 0.61037, which is a reduction of -4.288dB below full volume. **However, remember that all this is relative:** the gain factor can be applied to any 0 to 32767 level. Thus a gain factor of 1.5 (3dB) applied to 20000 = 30000. A gain factor of 0.5 (-6dB) applied to 20000 = 10000, etc.

End of SNDINFO UNITS

SNDINFO ZCROSS – Display fraction of zero-crossings in a soundfile

Usage

sndinfo zcross *infile starttime endtime*

Parameters

infile – input stereo soundfile
starttime – time in *infile* at which to begin the count
endtime – time in *infile* at which to end the count

Understanding the SNDINFO ZCROSS Function

SNDINFO ZCROSS counts the number of zero-crossings in your file. **Zero-crossings** occur when the waveform crosses the zero amplitude centre-line, which represents the rest position of the speaker cone between positive and negative motions. But what do we mean here by 'fraction of' and what use can we make of this information?

For a pure sinewave you will have 2 zero-crossings for every wavecycle. For a 441Hz wave, at a sample rate of 44100 samples per second, you would expect to find 100 wavecycles per second, and therefore 200 zero-crossings per second. The zero-crossing ratio is therefore defined here as 1:200 (= 0.005).

With more complex sounds, the wave may cross the zero amplitude line more than twice in each wavecycle, and with noise, the signal crosses the zero line very frequently. **Thus, more complex or noisy signals have a higher zero-crossing ratio, e.g. 1:3 (= 0.333). So the ratio gives some indication of how complex or noisy your sound is: the higher the ratio, the noisier the sound.**

4 Examples, specifying 1 second segments within the sound:

1. A fairly complex chord (with several adjacent semitones) made with SYNTH CHORD – the fraction of zero-crossings is 0.024442, or approximatgely 1:32.
2. Text delivered by a male voice, speaking normally – the fraction of zero-crossings is 0.044444, or approximately 3:64.
3. A riff played on a drum-kit – the fraction of zero-crossings is 0.176098, or approximately 11:64.
4. Noise generated by SYNTH NOISE – the fraction of zero-crossings is 0.502540, or approximately 1:2.

Musical Applications

SNDINFO ZCROSS is therefore a diagnostic tool for use with programs which use zero-crossing detection to modify a sound (e.g. the **DISTORT** programs). It gives you a numerical indication of the relative complexity of the sound and therefore the degree of distortion that may result from manipulating the 'pseudo-wavecycles' between zero-crossings.

End of SNDINFO ZCROSS