CDP GROUP LISTS

LIST OF PROCESSES IN EACH FUNCTION GROUP

BLUR – blur spectral data to diffuse clarity

Some BLUR processes operate across a number of time-windows, for example time-averaging the spectrum, shuffling or moving along the windows.
Others operate on spectral-channel amplitudes: thinning or averaging the spectral energy across a number of adjacent windows, or randomising amplitudes and frequencies.

BLUR **AVRG**	Average spectral energy over *N* adjacent channels
BLUR **BLUR**	Blur the spectral data over time
CALTRAIN	Time-blur the upper spectral channels
BLUR **CHORUS**	Atd random variation to amplitude or frequency in analysis channels
BLUR **DRUNK**	Modify sound by a drunken walk along analysis windows
BLUR **NOISE**	Add noise to spectrum
BLUR **SCATTER**	Randomly thin the spectrum
SELFSIM	Replace spectral windows with the most similar, louder window(s)
BLUR **SHUFFLE**	Shuffle analysis windows according to a specific scheme
BLUR **SPREAD**	Spread spectral peaks
BLUR **SUPPRESS**	Suppress the most prominent channel data
SUPPRESS PARTIALS	Suppress the most prominent partials in the frequency band indicated
WEAVE	Weave amongst the analysis windows in a specified pattern

COMBINE – combine frequency analysis data from two or more files

This group combines two or more spectra, for example by interleaving them or producing the sum, difference, maximum or mean value of the spectra.

Some processes combine characteristics of one spectral envelope with another (**CROSS, SPECROSS, SPECTWIN** and **SPECSPHINX**).

MAKE and **MAKE2** are key tools for combining **PITCH DATA** files with **FORMANT** files to produce spectra that can be re-synthesized back to sound.

COMBINE **CROSS**	Replace spectral amplitudes of 1st file with those of 2nd
COMBINE **DIFF**	Subtract one spectrum from another
COMBINE **INTERLEAVE**	Interleave (groups of) windows of several spectra
COMBINE **MAKE**	Generate spectrum from pitch & formant data
COMBINE **MAKE2**	Generate spectrum from pitch, formant & envelope data
COMBINE **MAX**	In each analysis channel, in each window, take the maximum value amongst the input files
COMBINE **MEAN**	Generate the spectral 'mean' of 2 sounds
SPECROSS	Interpolate partials of pitched *inanalfile1* towards those of pitched *inanalfile2*
SPECSPHINX	Impose the channel amplitudes of inanalfile2 onto the channel frequencies of inanalfile1
SPECTWIN	Combine the formant and/or total spectral envelopes of two spectra
COMBINE **SUM**	Find the sum of two spectra

DISTORT – distortion based on wavesets (pseudo-wavecycles)

The DISTORT functions operate on (groups of) pseudo-wavecycles called ***wavesets***, found in zero-crossings. Most, but not all, of the processes do introduce doestortion and many produce unpredictable results.

DISTORT **AVERAGE**	Average the waveshape over *N* 'wavecycles'
CLIP	Clip a signal
DISTORT **CYCLECNT**	Count 'wavecycles' in soundfile
DISTORT **DELETE**	Time-contract soundfile by deleting 'wavecycles'
DISTCUT	Cut sound into elements with falling envelope
DISTMARK	Interpolate between waveset-groups at marked points
DISTMORE BRIGHT	Reorder sound segments in order of average zero-crossing rate
DISTMORE DOUBLE	Double (quadruple etc.) frequency of each waveset
DISTMORE SEGSBKWD	Reverse certain (sets of) segments
DISTMORE SEGZIG	Zigzag across tail segments or across whole soundfile
DISTORTT	Repeat wavesets within given duration
DISTREP	Timestretch soundfile by repeating wavesets
DISTSHIFT	Time-shift or swap wavecycles
DISTWARP	Warp wavecycles by a multiplier
DISTORT **DIVIDE**	Distortion by dividing 'wavecycle' frequency
DISTORT **ENVEL**	Impose envelope over each group of *cyclecnt* 'wavecycles'
DISTORT **FILTER**	Time-contract a sound by filtering out 'wavecycles'
DISTORT **FRACTAL**	Superimpose miniature copies of source 'wavecycles' onto themselves
FRACTAL WAVE	Fractally distort an input sound or wavecyle
DISTORT **HARMONIC**	Harmonic distortion by superimposing 'harmonics' onto 'wavecycles'
DISTORT **INTERACT**	Time-domain interaction of two sounds
DISTORT **INTERPOLATE**	Time-stretch file by repeating 'wavecycles' and interpolating between them
DISTORT **MULTIPLY**	Distortion by multiplying 'wavecycle' frequency
DISTORT **OMIT**	Omit *A* out of every *B* 'wavecycles', replacing them with silence
DISTORT **OVERLOAD**	Clip the signal with noise or a (possibly timevarying) waveform
DISTORT **PITCH**	Pitchwarp 'wavecycles' of sound
DISTORT **PULSED**	Impose regular pulsations on a sound
QUIRK	Distort signal by raising sample values to a power
DISTORT **REFORM**	Modify the shape of 'wavecycles'
DISTORT **REPEAT**	Timestretch soundfile by repeating 'wavecycles'
DISTORT **REPEAT2**	Repeat 'wavecycles' without time-stretching
DISTORT **REPLACE**	The strongest 'wavecycle' in a *cyclecnt* group replaces the others
DISTORT **REPLIM**	Timestretch by repeating 'wavecycles' (below a specified frequency)
DISTORT **REVERSE**	Cycle-reversal distortion in which the 'wavecycles' are reversed in groups
SCRAMBLE	Scramble waveset order randomly or by size and level
DISTORT **SHUFFLE**	Distortion by shuffling 'wavecycles'
SPLINTER	Create splinters by repeating & shrinking selected waveset-group
DISTORT **TELESCOPE**	Time-contract sound by telescoping *N* wavecycles into 1

ENVEL – alterations to the amplitude envelope

CDP can extract an envelope (either as a binary or breakpoint text file) and edit or transform it,alter the dynamic shape, creating fades and swells, warping it in many ways, or totally re-drawing it, creating new envelopes. The envelope of one sound can be imposed on or replace that of another.

ENVEL **ATTACK**	Emphasize the attack of a sound
ENVEL **BRKTOENV**	Convert a (text) breakpoint envelope to a binary envelope file
ENVEL **CREATE**	Create an envelope
ENVEL **CURTAIL**	Curtail a soundfile by fading to zero at some time within it
ENVEL **CYCLIC**	Create a sequence of repeated envelopes, in a binary envelope file
ENVEL **DBTOENV**	Convert a (text) breakpoint file with dB values to a binary envelope file
ENVEL **DBTOGAIN**	Convert a (text) breakpoint file with dB values to gain values (0 to 1)
ENVEL **DOVETAIL**	Create fade ins and outs in a soundfile by enveloping its beginning and end
ENVEL **ENVTOBRK**	Convert a binary envelope file to a (text) breakpoint envelope file
ENVEL **ENVTODB**	Convert a binary envelope file to a (text) breakpoint envelope file with dB values
ENVEL **EXTRACT**	Extract the amplitude envelope from an input soundfile
FLATTEN	Equalise level of sound elements
ENVEL **GAINTODB**	Convert a (text) breakpoint file with gain (0 to 1) values to dB values
ENVEL **IMPOSE**	Impose an envelope on an input soundfile
ENVEL **PLUCK**	Pluck the start of a sound (Mono files only)
ENVEL **REPLACE**	Replace the existing envelope of a soundfile with a different envelope
ENVEL **RESHAPE**	Warp the envelope in a binary envelope file
ENVEL **REPLOT**	Warp the envelope in a (text) breakpoint envelope file
ENVEL **SCALED**	Impose an envelope on an input soundfile, scaling it timewise to the sound's duration
SPIKE	Envelope the sound to spike at the peak
ENVEL **SWELL**	Cause sound to fade in to and out from a peak moment
ENVEL **TIMEGRID**	Partition a soundfile into a sequence of 'windows' separated by silence
TOPANTAIL2	Fade beginning & end of a sound
TREMENV	Tremolo a sound, with witdh narrowed after peak
ENVEL **TREMOLO**	Tremolo a sound
TREMOLO	Witdh-controlled tremolo
ENVEL **WARP**	Warp the envelope of a soundfile

ENVNU – specialised operations on envelopes

EXPDECAY	Produce a true exponential decay to zero on a sound
PEAKCHOP	Isolate peaks in a source and either play back at a specified tempo (Mode **1**) OR gate the sound around peaks (Mode **2**)

EXTEND – processes which extend and/or segment a sound in time

The EXTEND group is concerned with repetitions of various sorts, which prolong the sound. The iterations tend not to repeat the whole file, but only segments of it (e.g. **LOOP**, **ZIGZAG**, **DRUNK**). **SEQUENCE** and **SEQUENCE 2** are quite versatile sample-playing functions, which can play a sequence of sounds at given times, pitch, loudness and duration. Many newer standalone programs have been attached to the group, which explore different aspects of repetition.

EXTEND **BAKTOBAK**	Join a time-reversed copy of the sound to a normal copy, in that order
BOUNCE	'Bounce' a sound: accelerating repeats, decaying in level
CERACU	Repeat the source sound in several cycles that synchronise after specified counts
EXTEND **DOUBLETS**	Divide a sound into segments that repeat, and splice them together
EXTEND **DRUNK**	Drunken walk through source file (chosen segments read forwards)
DVDWIND	Shorten a sound by read, skip, read, skip procedure
SFECHO **ECHO**	Repeat a sound with timing and level adjustments between repeats
ENVSPEAK	Process speech 'syllables'
EXTEND **FREEZE**	Freeze a portion of sound by iteration
HOVER	Move through a file, zig-zag reading it at a given frequency
HOVER2	Move through a file, zig-zag reading it at a given frequency, with inverted copies
EXTEND **ITERATE**	Repeat sound with subtle variations
ITERLINE	Iterate an input sound, following a transposition line
ITERLINEF	Iterate an input sound set, following a transposition line
EXTEND **LOOP**	Loop (repeat [advancing] segments) inside soundfile
MADRID	Spatially syncopate repetitions of the source soundfile(s)
MOTOR	Create faster pulse-stream within slower pulsed envelope
PULSER	Iterate a sound to create a stream of enveloped & pitched sound-packets
PULSER MULTI	Iterate a number of sounds, randomly permutated, to create a stream of enveloped and pitched sound-packets
REPEATER	Play source, with specified elements repeating
EXTEND **REPETITIONS**	Repeat source at given times
ROTOR	Generate note-sets that grow and shrink in pitch-range and speed
EXTEND **SCRAMBLE**	Scramble soundfile and write to any given length
EXTEND **SEQUENCE**	Produce a sequence from an input sound played at specified transpositions and times
EXTEND **SEQUENCE2**	Produce a sequence from several sounds played at transpositions and times specified
SHIFTER	Generate simultaneous repetition streams, shifting rhythmic pulse from one to another
SHRINK	Repeat a sound, shortening it on each repetition
SORTER	Chop sound into elements, then reorganise by loudness or duration
STUTTER	Randomly repeat segments cut from elements
TESSELATE	Create repeating patterns with shift in space and time
EXTEND **ZIGZAG**	Read soundfile backwards and forwards, as you specify

FILTER – techniques to filter or focus on frequency bands

Filtering changes the tone-colour of a sound, attenuating some of the harmonics and emphasizing others. CDP's filter functions include all the classic types plus particularly effective Filter Banks, which can be used to "tune" the sound by strongly emphasising specified frequencies. There are also spectrally based filters (see **HILITE**).

FILTER **BANK**	Bank of filters, with time-varying *Q*
FILTER **BANKFRQS**	Generate a bank of frequencies for use as a filterbank
FILTRAGE	Generate randomised VARIBANK filterbank files
FILTER **FIXED**	Boost or Cut: above, below or around a given frequency
FILTER **ITERATED**	Iterate a sound, with cumulative filtering by a filterbank
FILTER **LOHI**	Fixed low pass or high pass filter
FILTER **PHASING**	Phase shift sound, or produce phasing effect
PHASOR	Introduce phasing into (mono) signal
FILTER **SWEEPING**	Filter whose focus-frequency sweeps over a range of frequencies
FILTER **USERBANK**	User-defined filterbank, with time-varying *Q*
FILTER **VARIABLE**	Lo-pass, High-pass, Band-pass or Notch filter with time-varying frequency
FILTER **VARIBANK/2**	User-defined time-varying filterbank, with time-varying *Q*
FILTER **VFILTERS**	Make datafiles for fixed-pitch FILTER VARIBANK filters

FOCUS – focus or sustain spectral amplitude data

The FOCUS functions complement those in HILITE and BLUR by emphasising certain aspects of the spectral envelope, or operating across a number of time-windows: freezing, holding or sustaining each spectral band. **FOLD** and **SPECFOLD** radically alter spectral frequencies.

FOCUS **ACCU**	Sustain each spectral band, until louder data appears in that band
FOCUS **EXAG**	Exaggerate the spectral (formant) contour
FOCUS **FOCUS**	Focus spectral energy onto the peaks in the spectrum
FOCUS **FOLD**	Octave-transpose spectral components into a specified frequency range
FOCUS **FREEZE**	Freeze the spectral characteristics in a sound, at given times, for specified durations
FOCUS **HOLD**	Hold sound spectrum, at given times
SPECFOLD	Fold, invert or randomise the spectrum
FOCUS **STEP**	Step-frame through a sound by freezing the spectrum at regular time intervals
SUPERACCU	Sustain each spectral band until louder data appears in that band

FORMANTS – extract and impose the spectral envelope (formants)

Formant files extract the time-varying spectral envelope. This can then be imposed on a different sound or combined with a pitch file to create a new sound. The group includes a spectral version of the classic Vocoder, in which the spectral envelope of one sound is imposed on another.

Related groups are the **ONEFORM Group**, which operates with single formants and **SPECFNU**, a group of twenty-three formant-manipulating processes.

FORMANTS **GET**	Extract evolving formant envelope from an analysis file
FORMANTS **GETSEE**	Get formant data from an analysis file to a pseudo-soundfile to view in VIEWSF
FORMANTS **PUT**	Impose formants in a formant data file on data in an analysis file
FORMANTS **SEE**	Convert formant data in formant data file to a pseudo soundfile to view in VIEWSF
SPECENV	Extract the spectral envelope of file 2 and apply it to file 1
FORMANTS **VOCODE**	Impose spectral envelope of one sound on another
SEE ALSO:	
ONEFORM	Operations with single formants
SPECFNU	Modify spectral shape in relation to formant peaks

ONEFORM – operations with single formants

The ONEFORM group extracts and works with a single spectral envelope at a specific time. It complements the **FORMANTS** functions, and **SPECFNU**, which deals with various transformations while preserving the formants.

ONEFORM **GET**	Extract formant-envelope at a specific time in an existing formant file
ONEFORM **PUT**	Impose the formant-envelope in a single-moment-formants datafile onto the sound in an analysis file
ONEFORM **COMBINE**	Generate a new sound from pitch information and a single-moment formant

GRAIN – granulate and manipulate grains

The main CDP granular programs are **MODIFY BRASSAGE** and its variants **WRAPPAGE** and **NEWTEX.** Most functions in the GRAIN group manipulate the grains in a "grainy" sound, normally one with gaps (possibly created by **MODIFY BRASSAGE** Mode 5).
Grains can be treated in many ways, such as being duplicated, omitted, repitched, repositioned, shuffled and reversed in order.

GRAIN **ALIGN**	Synchronise grain onsets in 2nd grainy sound with those in the 1st
GRAIN **ASSESS**	Estimate best gate value for grain extraction
GRAIN **COUNT**	Count grains found in a sound (at given *gate* and *minhole* values)
GRAIN **DUPLICATE**	Duplicate grains in a grainy sound
GRAINEX	Find grains in a sound and extend the area that contains them
GRAIN **GREV**	Find and manipulate 'grains', using envelope troughs and zero-crossings
NEWTEX	Generate a texture of grains made from a source sound or sounds
GRAIN **NOISE_EXTEND**	Find and time-stretch noise components in a sound
GRAIN **FIND**	Locate timings of grain-onsets in a grainy sound
GRAIN **OMIT**	Omit a proportion of grains from a grainy sound
GRAIN **REMOTIF**	Change pitch and rhythm of grains in a grainy sound
GRAIN **REORDER**	Reorder grains in a grainy sound
GRAIN **REPITCH**	Repitch grains in a grainy sound
GRAIN **REPOSITION**	Reposition grain onsets in a grainy sound
GRAIN **RERHYTHM**	Change rhythm of grains in a grainy sound
GRAIN **REVERSE**	Reverse order of grains in a grainy sound without reversing the grains themselves
GRAIN **R_EXTEND**	'Time-stretch' natural sounds like the rolled 'rrr' in speech
GRAIN **TIMEWARP**	Stretch (or shrink) the duration of a grainy sound without stretching the grains themselves
WRAPPAGE	Granular reconstitution of one or more soundfiles over multi-channel space

HILITE – emphasise spectral amplitude, including spectral filtering

The spectral filtering functions are probably the most powerful of the HILITE group. **TRACE** is related to other thinning functions in the **BLUR** group.

HILITE **ARPEG**	Arpeggiate the spectrum
HILITE **BAND**	Split spectrum into bands and process these individually
HILITE **BLTR**	Time-average and TRACE the spectrum
HILITE **FILTER**	Filter the spectrum
GLISTEN	Randomly partition the spectrum into bins and play back in order
HILITE **GREQ**	Graphic EQ type filter on the spectrum
HILITE **PLUCK**	Emphasise spectral changes
HILITE **TRACE**	Highlight *n* loudest partials, at each moment (window) in time
HILITE **VOWELS**	Impose vowels on a sound

HOUSEKEEP – basic soundfile housekeeping utilities

Some HOUSEKEEP functions are sound utilities, providing basic channel and gating functions, or tidying-up facilities.
The rest of the group consists of file-handling utilities.

HOUSEKEEP **BAKUP**	Concatenate soundfiles in one backup file, with silences inbetween
HOUSEKEEP **BATCHEXPAND**	Expand an existing batch file
HOUSEKEEP **BUNDLE**	List files into a textfile, for sorting, backup or creating a dummy mixfile
CHANPHASE	Invert the phase of a specified channel of an input sound
HOUSEKEEP **CHANS**	Extract channels or change channel format of a soundfile
HOUSEKEEP **COPY**	Make and delete exact copies of a sound
HOUSEKEEP **DEGLITCH**	Attempt to deglitch a sound
HOUSEKEEP **DISK**	Show available space on disk
HOUSEKEEP **ENDCLICKS**	Remove clicks from the start or end of a soundfile
HOUSEKEEP **EXTRACT**	Extract significant sound from a recording, top & tail, remove glitches, etc.
HOUSEKEEP **GATE**	Chop at zeros
GATE	Remove low-level sound from signal
PAIREX	Extract any pair of channels from a multichannel sound
HOUSEKEEP **REMOVE**	Remove existing copies of a file
REPAIR	Join a list of mono sounds into stereo or multi-channel outputs
HOUSEKEEP **RESPEC**	Change sample rate, format or properties of a soundfile (CARE!!)
HOUSEKEEP **SORT**	Sort files listed in a textfile
TOSTEREO	Diverge from mono to stereo, in a stereo file

MODIFY –Wide variety of effects, including granular synthesis, transposition, loudness, pan, delay and ring-modulation

The **MODIFY** group has several fundamental time-based functions as well as more radical traditional techniques. **BRASSAGE** implements granular reconstruction.

MODIFY **BRASSAGE**	Granular reconstitution of soundfile
MODIFY **CONVOLVE**	Convolve the first soundfile with the second
DSHIFT	Atds Doppler shift to panning
MODIFY **FINDPAN**	Find stereo-pan position of a sound in a stereo file
MODIFY **LOUDNESS**	Alter loudness or balance of sound
NEWDELAY	Delay with pitch-defined output sound
PHASE	Invert phase or enhance stereo separation of a sound
MODIFY **RADICAL**	Reverse, shred, scrub, low-res, ring/cross mod, quantise resn.
MODIFY **REVECHO**	Atd reverberation or echo to the sound
MODIFY **SAUSAGE**	Brassage on several sources
MODIFY **SCALEDPAN**	Distribute sound in stereo space, scaling pan data to soundfile duration
MODIFY **SHUtdER**	Shutder a stereo soundfile
MODIFY **SPACE**	Spatialise, or alter the spatialisation of a soundfile
MODIFY **SPACEFORM**	Create a sinusoidal spatial distribution data file
MODIFY **SPEED**	Change speed (& pitch) of sound
MODIFY **STACK**	Create a mix that stacks transposed versions of source on top of one another
VERGES	Play source, with specified brief moments glissing up or down

MORPH – Create smooth transitions between sounds, using amplitude and frequency analysis data

Morphing interpolates between the spectra of two frequency-analysis files, creating a gradual transition between them. **GLIDE** makes a frequency transition between two single spectra over any time; a spectral envelope from a suitable sound can then be imposed on the result.

MORPH **BRIDGE**	Interpolate between a specified window in one file, and another window specified in another file
MORPH **GLIDE**	Interpolate between two single window spectra
MORPH **MORPH**	Morph between one spectrum and another, where spectra may be time-varying
NEWMORPH	Morph between dissimilar spectra
NEWMORPH2	Morph frequencies of spectral peaks

MULTICHANNEL – multi-channel diffusion and processing

The processes in the MULTI-CHANNEL group are all standalone programs which have been brought together purely for reference purposes.

Several are multi-channel equivalents of earlier mono/stereo functions. A second group manipulate channels or disperse sounds in multi-channel space. A third group is concerned with multi-channel mixing. See also the **MULTI-CHANNEL TOOLKIT** for further multichannel functions.

BROWNIAN	Generate texture of sampled elements following brownian motion in pitch and space
CASCADE	Successive segments are repeat-echoed, and the echosets are superimposed on the source
CRUMBLE	Project segments spatially over progressively smaller groups of channels
CRYSTAL	Generate sound-events based on the position of vertices of a crystal, then rotate the crystal in 3-D space
FLUTTER	Atd multi-channel-distributed tremolo to a multi-channel file
FRACTURE	Disperse a mono signal into fragments spread over *N*-channel space
FRAME SHIFT	Reorient or rotate a multi-channel file
MCHANPAN	Pan sounds around a multi-channel space
MCHANREV	Create multi-channel echoes or reverb
MCHITER	Iterate the input sound in a fluid manner, scattering around a multi-channel space
MCHSHRED	Cut sounds into random segments and reassemble in random order within original duration
MCHSTEREO	Combine stereo files in a multi-channel output
MCHZIG	Read back and forth in the soundfile, panning to a new channel at each 'zig' or 'zag'
MTON	Create a multi-channel equivalent of a mono soundfile
MULTIMIX CREATE	Create a multi-channel mixfile
NEWMIX	Mix from a multi-channel mixfile to give a multi-channel soundfile output
PANORAMA	Distribute *N* source files in a panorama across a specified angle of a sound-surround loudspeaker array
SPIN STEREO	Spin a wide stereo image across stereo / multichannel space, with possible doppler-shift
SPIN QUAD	Spin two wide stereo-images across a 5-channel-wide sound image, with possible doppler-shift
STRANS MULTI	Change the speed or pitch of a multi-channel sound, or atd vibrato
TANGENT SUBGROUP:	
TANGENT ONEFILE	Play repeats of a mono soundfile along a tangent path
TANGENT TWOFILES	Play repeats of two synchronised mono soundfiles along a tangent path
TANGENT SEQUENCE	Play a sequence of mono soundfiles along a tangent path
TANGENT LIST	Play a sequence of mono soundfiles as listed in a textfile along a tangent path
TEXMCHAN	Create textures over a multi-channel frame

TRANSIT SUBGROUP:

TRANSIT SIMPLE	Place repetitions of a mono soundfile on a path *into* and *across* an 8-channel array
TRANSIT FILTERED	Place filtered repetitions of a mono soundfile on a path *into* and *across* an 8-channel array
TRANSIT DOPPLER	Place pitch-shifted repetitions of a mono soundfile on a path *into* and *across* an 8-channel array, suggesting a doppler shift
TRANSIT DOPLFILT	Doppler effect on a path *into* and *across* an 8-channel array with filtering, to suggest greater distance
TRANSIT SEQUENCE	Position a sequence of mono sounds (at least 3) on a path *into* and *across* an 8-channel array
TRANSIT LIST	Position a sequence of mono sounds (at least 3), as listed in a textfile, on a path *into* and *across* an 8-channel array

Other multi-channel processes:

NEWTEX	Generate a texture of grains made from a source sound or sounds
WRAPPAGE	*Brassage* on one or more sounds, with (moving) multi-channel output

MULTI-CHANNEL TOOLKIT – multi-channel file handling functions

The Multi-Channel Toolkit is a group of versatile tools for handling multi-channel files, including a special emphasis on ambisonics and the **WAVE_EX** file format. **COPYSFX** can copy/convert between a very wide range of soundfile formats.

ABFPAN	Apply a fixed or orbiting 1st order ambisonic B-Format pan to mono soundfile
ABFPAN2	Apply a fixed or orbiting 2nd order ambisonic B-Format pan to mono soundfile
CHANNELX	Extract all or selected channels from a multi-channel soundfile
CHORDER	Reorder soundfile channels in multi-channel soundfile
CHXFORMAT	Modify WAVE_EX header to change GUID and/or speaker positions
COPYSFX	Copy soundfiles / convert from one format to another
FMDCODE	Decode 1st or 2nd order ambisonic B-Format soundfile to a choice of speaker layouts
INTERLX	Interleave mono or stereo files into a multi-channel file
NJOIN	Concatenate multiple soundfiles into a single file, with optional CUE list for CD burning
NMIX	Simple mix of two multi-channel soundfiles, with optional offset
PAPLAY	Playback of multi-channel soundfiles
RMSINFO	Scan file and report RMS and average power level statistics
SFPROPS	Display soundfile details, with WAVE_EX speaker positions

PITCH – operations based on partials

The PITCH spectral group consists of functions which directly alter frequency content

PITCH **ALTHARMS**	Delete alternate harmonics
PITCH **CHORD**	Transposed versions of the sound are superimposed onto the original
PITCH **CHORDF**	Transposed versions of the spectrum are superimposed within the existing spectral envelope
PITCH **OCTMOVE**	Octave transpose without formant shift
PITCH **PICK**	Only retain channels which might hold the partials specified
SPECTUNE	Find most prominent pitch and transpose file to it
PITCH **TRANSP**	Shift pitch of (part of) the spectrum
PITCH **TUNE**	Replace spectral frequencies by harmonics of specified pitch(es)
TUNEVARY	Replace spectral frequencies by harmonics of specified pitch(es), varying over time

PITCHINFO – information about partials or extracted pitch traces

PCHINFO functions are for binary pitch files (**.frq**), extracted from spectral files by **REPITCH GETPITCH**.

PITCHINFO **CONVERT**	Convert a binary pitch data file to a *time frequency* breakpoint text file
PITCHINFO **HEAR**	Convert binary pitchfile to analysis test tone file (resynthesise to hear pitch)
PITCHINFO **INFO**	Display information about pitch data in a (binary) pitchfile
PITCHINFO **SEE**	Convert binary pitchfile or transposition file to a pseudo-soundfile, for viewing
PITCHINFO **ZEROS**	Show whether a pitch data file contains uninterpolated zeros (unpitched windows)

PSOW – manipulate pitch-synchronous grains (FOFs)

PSOW is a set of experimental grain processes for vocal sounds. It attempts to find and then manipulate FOFs (formant grains), used to synthesise the singing voice. PSOW processes require a pitch-trace taken from a spectral analysis file (see PTOBRK). The programs allow you to alter formants independently of pitch, often with unexpected results.

PSOW **CHOP**	Chop sound into sections between specified grain (chunks) OR: Chop away sections of soundfile that you DON'T want to manipulate with PSOW functions.
PSOW **CUTATGRAIN**	Cut at exact grain time
PSOW **DELETE**	Time shrink sound by deleting a proportion of the pitch-synchronised grains
PSOW **DUPL**	Timestretch/transpose a sound by duplicating the pitch-synchronised grains
PSOW **FEATURES**	Impose new features on vocal-type sound, preserving or modifying FOF-grains
FOFEX EXTRACT	Extract FOFs to a file or to separate soundfiles
FOFEX CONSTRUCT	Superimpose FOFs to make output FOF
PSOW **GRAB**	Grab a pitch-synchronised grain from a file, and use it to create a new sound
PSOW **IMPOSE**	Impose vocal FOFs in 1st sound onto the 2nd sound
PSOW **INTERLEAVE**	Interleave FOFs from two different files
PSOW **INTERP**	Interpolate between 2 pitch-synchronised grains, to produce a new sound
PSOW **LOCATE**	Locate exact start time of nearest FOF-grain
PTOBRK	Convert pitch trace from binary .frq to text breakpoint file (.txt or .brk) for PSOW
PSOW **REINFORCE**	Reinforce harmonics in a vocal-type FOF-grain file
PSOW **REPLACE**	Combine FOFs of 1st sound with the pitch of the 2nd sound
PSOW **SPACE**	Distribute the alternate FOFs in the sound over a stereo space
PSOW **SPLIT**	Split vocal FOFs into subharmonic and upwardly transposed pitch
PSOW **STRETCH**	Timestretch/transpose a sound by repositioning the pitch-synchronised grains.
PSOW **STRTRANS**	Timestretch/transpose a sound by repositioning the pitch-synchronised grains
PSOW **SUSTAIN**	Sustain a pitch-synchronised FOF within a sound
PSOW **SUSTAIN2**	Sustain an explicitly specific FOF within a sound
PSOW **SYNTH**	Impose vocal FOFs on a stream of synthesised sound
TWEET	Replace FOFs in vocal sound by synthetic tweets or noise

PVOC – FFT analysis and resynthesis

The PHASE VOCODER (**PVOC**) converts between soundfiles in the time-domain and spectral analysis files (.ana or .pvx) in the frequency domain – and back again.

ANA2PVX	Convert CDP analysis file (**.ana**) to PVOC-EX file (**.pvx**)
PVOC **ANAL**	Convert soundfile to spectral file
PVOC **EXTRACT**	Analyse, then resynthesise sound with various options
FTURANAL ANAL	Extract spectral features from an analysis file and output to a textfile
FTURANAL SYNTH	Use spectral features data to reassemble MONO source file
PVOC **SYNTH**	Convert spectral file to soundfile
PVOCEX2	Stereo phase vocoder based on CARL pvoc (Mark Dolson)
SEE ALSO:	
PVPLAY	Direct playback of PVOC analysis files

REPITCH – Extract and alter pitch-related frequency analysis data

The REPITCH programs mostly extract pitch from frequency analysis files and process the resultant pitch files. To re-synthesise sound, **COMBINE MAKE** combines the pitch file with a formant file (.for), which represents the time-varying spectral envelope.

REPITCH **ANALENV**	Extract the window-loudness envelope of an analysis file
REPITCH **APPROX**	Make an approximate copy of a binary pitch data file
BRKTOPI	Convert a breakpoint pitch data file to a binary pitch data file
REPITCH **COMBINE**	Generate transposition data from 2 sets of pitch data, OR transpose pitch data with transposition data, OR combine 2 sets of transposition data to form new transposition data, producing a binary data file output
REPITCH **COMBINEB**	Generate transposition data from 2 sets of pitch data, OR transpose pitch data with transposition data, OR combine 2 sets of transposition data to form new transposition data, producing a *time value* breakpoint file output
REPITCH **CUT**	Cut out and keep a segment of a binary pitch data file
REPITCH **EXAG**	Exaggerate the contour of a pitch data file
REPITCH **FIX**	Massage pitch data in a pitch data file
REPITCH **GENERATE**	Create a binary pitch data file from a textfile of *time midi* value pairs
REPITCH **GETPITCH**	Extract pitch from spectrum to a pitch data file
REPITCH **INSERTSIL**	Mark areas as silent in a binary pitch data file
REPITCH **INSERTZEROS**	Mark areas as unpitched in a pitch data file
REPITCH **INTERP**	Replace noise or silence by pitch interpolated from existing pitches
REPITCH **INVERT**	Invert pitch contour of a pitch data file
REPITCH **NOISETOSIL**	Replace unpitched windows by silence in a pitch data file
REPITCH **PCHSHIFT**	Transpose pitches in a pitch data file by a constant number of semitones

REPITCH **PCHTOTEXT**	Convert binary pitch data to text
REPITCH **PITCHTOSIL**	Replace pitched windows by silence
REPITCH **QUANTISE**	Quantise pitches in a pitch data file
REPITCH **RANDOMISE**	Randomise pitch line in a pitch data file
REPITCH **SMOOTH**	Smooth pitch contour in a pitch data file
REPITCH **SYNTH**	Create the spectrum of a sound following the pitch contour in a pitch data file
REPITCH **TRANSPOSE**	Transpose spectrum (spectral envelope also moves)
REPITCH **TRANSPOSEF**	Transpose spectrum: but retain original spectral envelope
REPITCH **VIBRATO**	Atd vibrato to pitch in a pitch data file
REPITCH **VOWELS**	Create spectrum of vowel sound(s) following pitch contour in pitch file
SEE ALSO:	
COMBINE MAKE	Generate spectrum from pitch & formant data
COMBINE MAKE2	Generate spectrum from pitch, formant & envelope data only
PITCHINFO CONVERT	Convert a binary pitch data file to a time-frequency breakpoint text file
PTOBRK	Convert pitch trace from binary file (.frq) to text breakpoint file (with zeros) for PSOW

RETIME – rearrange and retime events in a soundfile

The **RETIME** program has 14 modes dealing with the retiming of events in a soundfile. There are three divisions: functions that create or retime silences, those that retime amplitude peaks, and those that retime silence-separated events. RETIME is documented within the SFEDIT (soundfile editing) group.

1: PULSED PEAKS	Output user-specified peaks at a regular pulse at the given tempo
2: SYNCHRONISE PEAKS	Reposition specified peaks to specified times at a given tempo
3: SHORTEN EVENTS	Shorten existing silence-separated events
4: PULSED	Find existing silence-separated events and output them at a regular pulse
5: SPEED	Find existing silence-separated events and change their speed
6: REPOSITION AT BEATS	Find existing silence-separated events and position them at specified beats in the output
7: REPOSITION AT TIMES	Find existing silence-separated events and position them at specified times in the output
8: REPEAT EVENT(S)	Repeat one (or a group of) silence-separated event(s) at a specified tempo
9: MASK EVENTS	Replace some silence-separated events by silence in a specified pattern
10: ACCENTS	Adjust levels of silence-separated events to be more equal, or accented
11: FIND DURATIONS	Find the durations of the shortest and longest silence-separated events
12: FIND START	Find first non-zero sample in soundfile
13: MOVE FOUND PEAK	Find sound peak and move whole sound so peak goes to specified time
14: MOVE SPECIFIED PEAK	Specify peak position, then move whole sound so peak goes to specified time

REVERB – functions to reverberate soundfiles

The REVERB group's two main programs are **REVERB** and **ROOMVERB**, comprehensive reverberation functions incorporating such features as reverberation time, dry/wet mix, absorption and early reflections. The programs use built-in sets for small, medium and large rooms, or they can use a datafile created by the separate **ROOMRESP** program. There is also optional filtering and pre-delay (giving the effect of a bigger room). ROOMRESP's data can also be used by **TAPDELAY**, a tapped delay line.

Also in the group is **FASTCONV**, an FFT-based convolution program, which can simulate a wide range of reverberation types using a sampled impulse response of a building or other responsive space. More experimentally, the impulse-response input can be any soundfile. Convolution can also implement an FIR linear-phase filter.

FASTCONV	Multi-channel FFT-based convolution
REVERB	Multi-channel reverberation (classic Schroeder)
ROOMRESP	Create early reflections data file for REVERB, ROOMVERB and TAPDELAY
ROOMVERB	Multi-channel reverberation with room simulation
TAPDELAY	Stereo multi-tapped delay line with feedback

SFEDIT – soundfile editing

SFEDIT has fundamental cutting and splicing functions, plus creative edits not usually found elsewhere, e.g. **MASKS**, **ISOLATE** and **PARTITION**, **RANDCHUNKS** and **RANDCUTS**, **TWIXT** and **SPHINX**, **PACKET** and **WAVEFORM**, **JOINSEQ** and **JOINDYN**.)

CANTOR	Cut holes in a sound in the manner of a cantor set (holes within holes within holes)
CONSTRICT	Shorten the durations of any zero-level sections in a sound
SFEDIT **CUT**	Cut and keep a segment of a sound
SFEDIT **CUTEND**	Cut out and keep the end part of a soundfile
SFEDIT **CUTMANY**	Cut and keep several segments of a sound
ENVCUT	Cut sound into elements with falling envelope
SFEDIT **EXCISE**	Remove a segment from a soundfile and close up the gap
SFEDIT **EXCISES**	Remove segments of a soundfile and close up the gaps
SFEDIT **INSERT**	Insert 2nd sound into 1st (overwriting or spreading first sound)
SFEDIT **INSIL**	Insert silence into a sound (overwriting or spreading the sound apart)
ISOLATE	Disjunct portions of soundfile are specified by textfile or dB loudness and saved to separate files
SFEDIT **JOIN**	Join files together, one after another
SFEDIT **JOINDYN**	Join in loudness-patterned sequence
SFEDIT **JOINSEQ**	Join in patterned sequence
MANYSIL	Insert many silences into a soundfile
SFEDIT **MASKS**	Mask specified chunks of a sound, with silence
SFEDIT **NOISECUT**	Suppress noise in a (mono) sound file, replacing with silence
PACKET	Isolate or generate a sound packet
PARTITION	Partition a mono soundfile into disjunct files in blocks defined by groups of wavesets
PREFIX SILENCE	Atd silence to the beginning of a soundfile
SFEDIT **RANDCHUNKS**	Cut chunks from a soundfile, randomly
SFEDIT **RANDCUTS**	Cut soundfile into pieces with cuts at random times
SFEDIT **REPLACE**	Insert a 2nd sound into an existing sound, replacing part of the original sound
REJOIN	Remix segment-files originating in ISOLATE process
RETIME	Rearrange and retime events within a soundfile
SILEND	Atd silence to the end of a soundfile
SFEDIT **SPHINX**	Switch between several files, with different switch times, to make new sound
SFEDIT **SUBTRACT**	Subtract one file from another
SFEDIT **SYLLABLES**	Separate out vocal syllables
SFEDIT **TWIXT**	Switch between several files, to make a new sound
WAVEFORM	Generate a wavetable from existing sound
SFEDIT **ZCUT**	Cut and keep a segment of soundfile, cutting at zero crossings
SFEDIT **ZCUTS**	Cut and keep segments of a MONO soundfile, cutting at zero crossings (no splice)
SEE ALSO: **DISTCUT**	Cut sound into elements with falling envelope

SNDINFO – basic soundfile information

SNDINFO functions either report to the terminal or produce textfile reports. Some functions require two or more infiles, to compare them. **UNITS** and **TIMESMP** offer a number of useful unit conversions.

Other reporting functions specific to particular Function Groups are found in that group. (These include DISTORT **cycle count** , GRAIN **grain count**, MODIFY **find pan** and STRETCHA: **stretch factor** for **STRETCH TIME**.)

SNDINFO **CHANDIFF**	Compare channels in a stereo soundfile
SNDINFO **DIFF**	Compare two sound, analysis, pitch, transposition, envelope or formant files
SNDINFO **FINDHOLE**	Find largest low level hole in a soundfile
SNDINFO **LEN**	Display duration of a soundfiling-system file
SNDINFO **LENS**	List durations of several soundfiling-system files
SNDINFO **LOUDCHAN**	Find loudest channel in a stereo soundfile
SNDINFO **MAXI**	List levels of several soundfiles
SNDINFO **MAXSAMP**	Find maximum sample in a soundfile or binary data file
SNDINFO **MAXSAMP2**	Find maximum sample within a specified timerange in a soundfile or binary data file
ONSET	Return the succession of sound-onsets in each channel of a multichannel file
SNDINFO **PEAKFIND**	Find times of loudness peaks in a sound and output as a datafile
SNDINFO **PRNTSND**	Print sound sample data to a textfile
SNDINFO **PROPS**	Display properties of soundfiling-system file
SEARCH **SIGSTART**	Find earliest time at which there is signal in two or more soundfiles.
SNDINFO **SMPTIME**	Convert sample count to time in soundfile
SNDINFO **SUMLEN**	Sum durations of several soundfiling-system files
SNDINFO **TIMEDIFF**	Find difference in duration of two soundfiles
SNDINFO **TIMESMP**	Convert time to sample count in soundfile
SNDINFO **UNITS**	Convert between different units
SNDINFO **ZCROSS**	Display fraction of zero-crossings in a soundfile
SEE ALSO:	
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

SPEC – gain and editing utilities in the spectral domain

A collection of spectral utilities, some of which are spectral counterparts of time-domain functions. **CLEAN** has been superseded by **SPECNU CLEAN**; both require a sample of noise for comparison, so that the noise frequencies may be spectrally removed.

ANALJOIN	Join analysis files together
SPEC **BARE**	Zero the data in channels which do not contain harmonics
SPEC **CLEAN**	Remove noise from spectral analysis file
SPEC **CUT**	Cut a section out of an analysis file, between *starttime* and *entdime* (seconds)
SPEC **GAIN**	Amplify or attenuate the spectrum
SPEC **GATE**	Zero all channels (in all windows) whose amplitude lies below the threshold
SPEC **GRAB**	Grab a single analysis window at the point specified
SPEC **MAGNIFY**	Expand (in duration) a single analysis window at time *time* to duration *dur*

SPECFNU – Modify spectral shape in relation to formant peaks

SPECFNU is a single program of 23 modes, modifying the spectral shape while aiming to retain the existing formants. Many of the functions are equivalents of existing ones, but with the addition of preserving formants. Other modes manipulate the formants themselves.

1 NARROW FORMANTS	Steepen skirts of formant peaks by power factor
2 SQUEEZE SPECTRUM	Squeeze spectrum around specified formant
3 INVERT FORMANTS	Formant peaks become troughs, and troughs peaks
4 ROTATE FORMANTS	Formant peaks & freqs move up (or down) spectrum, re-appearing at foot (or top) on reaching formants' edge
5 SPECTRAL NEGATIVE	Spectral values inverted for each channel
6 SUPPRESS FORMANTS	Suppress the selected formant(s)
7 GENERATE FILTER(S) FROM FORMANTS	Output Varibank filter data based on formant peaks
8 MOVE FORMANTS BY	Displace individual formants by a Hz value
9 MOVE FORMANTS TO	Displace individual formants to specified frequencies
10 ARPEGGIATE	Arppegiate partials of sound, under formants
11 OCTAVE-SHIFT	Octave-shift pitch of sound, under formants
12 TRANSPOSE	Transpose pitch of sound, under formants
13 FREQ-SHIFT	Frequency shift partials of source sound, under formants
14 RESPACE PARTIALS	Respace partials in source spectrum, retaining formants.
15 PITCH-INVERT	Invert pitch of sound, under formants
16 PITCH-EXAGG/SMOOTH	Exaggerate/Smooth pitch line, under formants
17 PITCH-QUANTISE	Force pitch onto pitch field, under formants
18 PITCH-RANDOMISE	Randomise pitch of source, under formants
19 RANDOMISE PARTIALS	Random-shift partials of sound, under formants
20 SEE SPEC ENVELOPES	Outputs viewable (not playable) soundfile
21 SEE SPEC PEAKS/TROUGHS	Print textfile of frequencies of peaks & troughs per window
22 GET LOUDNESS TROUGHS	Print textfile of times-of-troughs between syllables
23 SINE SPEECH	Convert formant frequencies to sine tones. A single sine wave represents each formant.

SPECINFO – information about spectral data

SPECINFO functions either report to the terminal or produce textfile reports. Various functions report on the time-varying spectral peaks in the sound or extract the amplitudes of partials. (**NEWMORPH2** in the MORPH function group lists the frequencies of the most prominent spectral peaks.) **PRINT** prints (part of) the analysis data to a textfile. This produces huge files and is best used for tiny portions.

SPECINFO **CHANNEL**	Returns PVOC channel number corresponding to frequency given
SPECINFO **FREQUENCY**	Returns centre frequency of the PVOC channel specified
GET_PARTIALS HARMONIC	Extract relative amplitudes of partials in a pitched source
SPECINFO **LEVEL**	Convert (varying) level of analysis file to a pseudo-soundfile, for viewing (1 window -> 1 sample)
SPECINFO **OCTVU**	Text display of the time varying amplitude of the spectrum, within octave bands
SPECINFO **PEAK**	Locate time varying energy centre of spectrum (text display)
PEAK EXTRACT	Extract spectral peaks from analysis file and write to a text file
SPECINFO **PRINT**	Print data in an analysis file as text to file
SPECINFO **REPORT**	Text report on location of frequency peaks in the evolving spectrum
SPECINFO **WINDOWCNT**	Returns the number of windows in the *infile*

SPECNU – various functions, mainly to clean up frequency analysis files

Three functions in this group are concerned with cleaning up files. **SLICE** partitions the spectrum into frequency bands and outputs these as separate files, an important resource for processing of these files separately prior to a possible remix.

FRACTAL SPECTRUM	Fractally distort spectrum by transposition
SPECNU **CLEAN**	Eliminate any persisting signal that falls below a threshold (defined by the noise file)
MATRIX	Matrix manipulation of spectrum of sound
SPECNU **RAND**	Randomise the order of spectral windows
SPECNU **REMOVE**	Remove a pitched component from the spectrum of a sound
SPECNU **SLICE**	Divide an analysis file into individual frequency bands, saving each as a separate analysis file OR: Invert spectral frquencies around a given frequency
SPECGRIDS	Partition a spectrum into parts, over a grid (=SLICE Mode 1)
SPECULATE	Generate versions of source with channel data progressively permutated
SPECNU **SQUEEZE**	Squeeze the spectrum into a frequency range, around a specified frequency
SPECNU **SUBTRACT**	Eliminate any persisting signal that falls below a threshold and subtract the amplitude of the noise in the *noisefile* from any source file signal that is passed

STRANGE – some unusual and unpredictable effects

The "strangeness" of this Function Group is perhaps subjective. **WAVER**'s detuning can be subtle or extreme; if minimal, it can be used to give a chorus effect.
The frequency-shift function **SHIFT** is a standard way of turning harmonic sounds into inharmonic ones.

STRANGE **GLIS**	Create glissandi inside the (changing) spectral envelope of the original sound
STRANGE **INVERT**	Invert the spectrum
STRANGE **SHIFT**	Linear frequency shift of (part of) the spectrum
STRANGE **WAVER**	Oscillate between harmonic and inharmonic state

STRETCH – stretch time or frequency data

The two original processes **STRETCH SPECTRUM** (c.f. STRANGE SHIFT) and **STRETCH TIME** are complemented by a revised time-stretcher **SPECTSTR** and the utility program **STRETCHA**.

SPECTSTR	Time-stretch analysis file, suppressing artefacts when stretch > 1
STRETCH **SPECTRUM**	Stretch/compress the frequency components of a sound in an inharmonic way
STRETCH **TIME**	Stretch/compress a sound in time without changing the pitch
STRETCHA	Utility to calculate *timestretch* factor relating to beats and tempo for use with STRETCH TIME

SUBMIX – mixing and related processes

The centrepiece of the SUBMIX group is **MIX**, which mixes the sounds listed in a text ***mixfile***. (See also the multi-channel version: **NEWMIX**). There are also support functions which manipulate the data in mixfiles.

Other mixing functions do not use mixfiles, including **MERGE**, **MERGEMANY**, **BALANCE**, **CROSSFADE** and **FADERS** – plus the two **INBETWEEN** functions. An important workhorse is **INTERLEAVE**, which combines mono soundfiles into a stereo or multi-channel file. (See also **INTERLX** in the Multi-Channel Toolkit.)

SUBMIX **ADDTOMIX**	Atd soundfiles (at maximum level and time zero) to an existing mixfile
SUBMIX **ATSTEP**	Convert a list of soundfiles to a mixfile (fixed time-step)
SUBMIX **ATTENUATE**	Alter the overall level of a mixfile
SUBMIX **BALANCE**	Mix between two soundfiles, using a balance function
SUBMIX **CROSSFADE**	Quick crossfade between 2 soundfiles (with same number of channels)
SUBMIX **DUMMY**	Convert list of sound names to a basic mixfile (for editing)
SUBMIX **FADERS**	Mix several mono or stereo files using a time-changing balance function
SUBMIX **FILEFORMAT**	Display format of a mixfile
SUBMIX **GETLEVEL**	Test maximum level of a mix, defined in a mixfile
SUBMIX **INBETWEEN**	Generate a set of sounds inbetween the 2 input sounds, through weighted mixes of the input sounds, from mostly sound1 to mostly sound2
SUBMIX **INBETWEEN2**	Generate a set of sounds in-between the 2 input sounds by interpolation pegged to zero-crossings
SUBMIX **INTERLEAVE**	Interleave mono files to make a single multichannel outfile
SUBMIX **MERGE**	Quick mix of 2 soundfiles (with the same number of channels)
SUBMIX **MERGEMANY**	Quick mix of several soundfiles (with the same number of channels)
SUBMIX **MIX**	Mix sounds as instructed in a mixfile
SUBMIX **MODEL**	Replace soundfiles in an existing mixfile
SUBMIX **ONGRID**	Convert listed soundfiles to a basic mixfile on a timed grid (for editing)
SUBMIX **PAN**	Pan sound positions in a mixfile
SUBMIX **SHUFFLE**	Shuffle the data in a mixfile
SUBMIX **SPACEWARP**	Alter the spatial distribution of a mixfile
SUBMIX **SYNC**	Synchronise soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles
SUBMIX **SYNCATTACK**	Synchronise the attacks of soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles
SUBMIX **TEST**	Test the syntax of a mixfile
SUBMIX **TIMEWARP**	Timewarp the data in a mixfile

SYNTH – functions to synthesise sounds

Only a small number of synthesis functions are provided within CDP, which is predominantly a sound-processing system. Apart from basic waveforms, there are additive-synthesis possibilities in **NEWSYNTH**, while **SYNFILT** and **SYNSPLINE** can also produce a richly varying timbre and **TS OSCIL** sonifies an arbitrary series of numbers.

SYNTH **CHORD**	Generate a chord with a simple waveform
CLICKNEW	Make clicktrack using times listed in textfile
SYNTH **CLICKS**	Create a click track from tempo, meter and barring data
IMPULSE	Create a stream of impulses
MULTIOSC	Nested FM-style oscillations
MULTISYNTH	Synthesize several sound-streams from a score
NEWSYNTH	Synthesisze complex spectra
NEWSCALES	Synthesise a series of short tones with defined frequency and timbre
SYNTH **NOISE**	Generate noise
PULSER SYNTH	Iterate synthesized wave-packets defined by partials data
SYNTH **SILENCE**	Make silent soundfile
SYNTH **SPECTRA**	Generate both channels of a stereo spectral band
SYNFILT	Noise filtered by time-varying filterbank, with time-variable Q
SYNSPLINE	Synthesise waveforms by smoothly joining randomly generated points
TS OSCIL	Create sound from time-series text data
TS TRACE	Create sound from time-series data treated as a pitch-trace
TSCONVERT	Convert input data to specified range and format
SYNTH **WAVE**	Generate simple waveforms

SYSUTILS – system utilities

A group of standalone support programs. Note especially **COPYSFX**, which converts between soundfile formats, and **COLUMNS** manipulates or generates data files in over 100 ways.

ASCIIGET	Display the contents of a text file as a list of characters with ASCII decimal code
(ALIAS)	Create a shortcut to a soundfile (PC only) – withdrawn
COLUMNS	Manipulate or generate columns of numbers
COPYSFX	Copy/convert a soundfile
DIRSF	Soundfile directory listing
GETCOL	Extract a column of numbers from a textfile
LISTAUDEVS	List audio devices
PAPLAY	Playback of multi-channel soundfiles
PUTCOL	Place a column of numbers into a textfile
PVPLAY	Play back (audition) a PVOC analysis file or soundfile
RECSF	Record, creating a soundfile
VECTORS	Numerical operations between two columns of figures

SEE ALSO:

TSCONVERT	Convert input data to specified range and format

TEXTURE – create textures of sounds

The TEXTURE processes repeat and transpose the input sound(s) in various ways to create a texture of note-events. The note-events are whole sounds, which may or may not be played to the end.

Note-events can be treated as simple repetitions, or repeated in groups, or with a timed rhythm, or as transposed ornaments or fully-defined motifs (timed and transposed). Key parameter values such as pitch, duration, gain (level), spatialisation and sound number are generally chosen randomly from within a specified time-variable range (which can also be a fixed value).

In each of the eight main functions, repetitions can be pitched at random within the defined pitch range, or restricted to a user-defined ***pitch-set*** or ***harmonic-field*** (which octave-transposes the pitches); the set/field can be time-varying. More than one input sound can optionally be used, and the range of sounds chosen can be time-varied.

The Texture programs are best explored at first using a simple note sample or similar short sound. CDP supplies two sets of tutorial examples for this purpose.

TEXTURE **SIMPLE**	Create textures from single events
TEXTURE **GROUPED**	Create textures from groups of events
TEXTURE **DECORATED**	Create a texture with decorations
TEXTURE **MOTIFS**	Create a texture with motifs
TEXTURE **MOTIFSIN**	Create a texture with motifs forced onto a harmonic field
TEXTURE **ORNATE**	Create a texture with ornaments
TEXTURE **POSTDECOR**	Create a texture with decorations following events
TEXTURE **POSTORNATE**	Create a texture with ornaments following events
TEXTURE **PREDECOR**	Create a texture with decorations preceding events
TEXTURE **PREORNATE**	Create a texture with ornaments preceding events
TEXTURE **TIMED**	Create a texture with timed single events
TEXTURE **TGROUPED**	Create a texture with timed event groups
TEXTURE **TMOTIFS**	Create a texture with timed motifs
TEXTURE **TMOTIFSIN**	Create a texture with timed motifs forced onto a harmonic field