GEN Routines

CSOUND

F-Statements

A GEN routine is declared in the score file with an f-statement:

F p1 p2 p3 p4 p5…

P1 = f-table identification number
P2 = Initialization time
P3 = f-table size
P4 = GEN routine number used by the f-table

Example:
F1 0 1024 10 1

Specifies an f-table using GEN10 used to generate a sine wave

F-Statements

A GEN routine is declared in the score file with an f-statement:

F p1 p2 p3 p4 p5…

P1 = f-table identification number
P2 = Initialization time
P3 = f-table size
P4 = GEN routine number used by the f-table

Example:
F1 0 1024 10 1

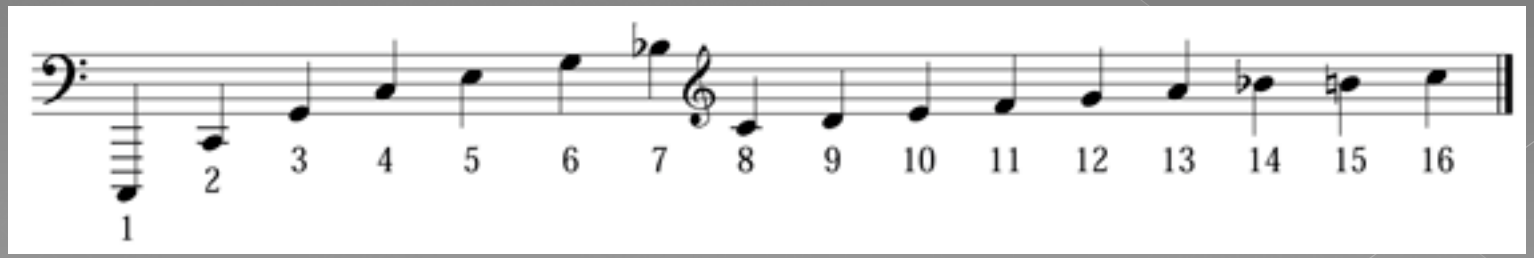
Specifies an f-table using GEN10 used to generate a sine wave

GEN X

- GEN routines are look up tables
- They hold periodic waveforms, random values, polynomials etc…
- These values are stored in f-tables, and reside in RAM during the synthesis process
- F-tables are usually declared in the score file and are typically restricted in length to a power of 2 or a power of 2+1
- F-tables are often accessed from the orc file. For instance the oscil opcode

GEN 10

- Although Gen 10 s often used to generate sinewaves, it is capable of more complex waveforms as well, by adding additional parameters
- Each additional parameters represents the amplitude of the 1st, 2nd, 3rd etc… harmonics, effectively acting as an additive synthesizer



GEN 10

- A sawtooth wave whose harmonics have an amplitude of 1/f can thus be approximated:

```
f10 0 16384 10 1 .5 .333333 .25 .2 .166667 .142857 .
   125 .111111 .1 .090909 .083333 .076923 ; Saw
```

- A square wave whose odd harmonics have an amplitude of 1/f can thus be approximated:

```
f3 0 16384 10 1 0  .333333 0 .2 0 .142857 0 .111111
   0  .090909 0 .076923 ; Square
```

GEN 1

- Gen 1 writes a soundfile into a f-table
- **f#** time size 1 filcod skiptime format channel
- The table size has to be a power of 2 or power of 2 + 1
- It is possible to set the table size to 0, which means csound will allocate the actual number of points in the table (usually ot a power of 2). In this case the table will not be useable by an OSCIL opcode, but a LOSCIL will read it fine.
- Skiptime: begin reading at skiptime seconds into the file
- Channel: number of chanels to read. 0 reads all channels
- If format = 0, the info is taken from the file's header

GEN 7, 5

- Gen 7 generates functions by connecting straight line segments
- The odd fields represent the point that are to be connected by a straight line
- The even fields indicate the number of values it takes to connect the two points. A zero value will specify a discontinous waveform.
- A sawtooth wave can be represented as

```
f4 0 16384 7 1 16385 -1
```

- A square wave can be represented as

```
f5 0 16384 7 1 8192 1 0 -1 8192 -1
```

- Note: These wavefors aca end to sound harsh, or buzzy due to the large vertical steps generated by the straight lines. (aliasing)
- Gen 5 operates on the same principle but generates exponential curves

  (0 values in odd fields not allowed)

Other GEN functions

- Gen 6: generate a function comprised of segments of cubic polynomials, spanning specified points just three at a time.
- Gen 8: generate a piecewise cubic spline curve, the smoothest possible through all specified points.
- Gen 3, 13,14,15: generate polynomial functions that can be used in waveshaping instruments

- http://www.tonalsoft.com/pub/pitch-bend/pitch.2005-10-03.17-01.aspx

Envelopes

- The most basic opcodes for envelope generation in csound are **line** and **expon**
- **Line**: Creates a straight line between 2 points over a specified period of time
- `kstraight line 0, p3, p5`
- `Where 0 = start value, p3= duration, p5= end value`
- `Linear values are not ideal for amplitude, as they tend to sound abrupt and unnatural. For that purpose a beter solution is expon`
- **`Expon:`** `creates exponential segments (as a ratio) between two values`
- `Zeros are illegal when using the expon opcode`
- `kexpon expon 0, p3, p5`

- `Both` **`Line`** `and` **`expon`** `can run at both k and a rate. Not I rate.`

Envelopes

Linseg and Expseg

Output values kr or ar trace a straight line (exponential curve) or a series of line segments (exponential segments) between specified points.

```
ar    expon      ia, idur1, ib
Kr    linseg     ia, idur1, ib[, idur2, ic[...]]
```

Linseg and Expseg take an odd number of arguments

Envelopes

- **adsr** — Calculates the classical ADSR envelope using linear segments
- **linen** — Applies a straight line rise and decay pattern to an input amp signal.
- **envlpx** — Applies an envelope consisting of 3 segments.

Basic Oscillators

Oscil: Relies on a ftable stored waveform to generate sound

For that reason the oscil opcode is able to generate many different waveforms

Oscili: Oscillator with linear interpolation

```
ares oscili xamp, xcps, ifn [, iphs]
Ifn requires a wrap around guard point
```

```
Loscil: reads sample sounds from a table
Oscili: performs linear interpolation between
        values in the wavetable. Requires a
        table size of power of 2 + 1
```

Basic Oscillators

See also:

oscil3 -- A simple oscillator with cubic interpolation
oscils -- A simple, fast sine oscillator
poscil -- High precision oscillator
poscil3 -- High precision oscillator with cubic interpolation.

Other Oscillators

oscilikt — A linearly interpolated oscillator that allows changing the table number at k-rate.

osciliktp — A linearly interpolated oscillator that allows allows phase modulation.

oscilikts — A linearly interpolated oscillator with sync status that allows changing the table number at k-rate.

buzz — Output is a set of harmonically related sine partials.

mpulse — Generates a set of impulses.

vco — Implementation of a band limited, analog modeled oscillator.

LFOs

lfo -- A low frequency oscillator of various shapes.

Description

A low frequency oscillator of various shapes.

Syntax

kr **lfo** kamp, kcps [, itype]ar **lfo** kamp, kcps [, itype]

Initialization*itype* (optional, default=0) -- determine the waveform of the oscillator.

Default is 0.•*itype* = 0 - sine•*itype* = 1 - triangles•*itype* = 2 - square (bipolar)•*itype* = 3 - square (unipolar)•*itype* = 4 - saw-tooth•*itype* = 5 - saw-tooth(down)

Oscils and Envelopes

It is possible to use an oscil as an envelope generator by setting its Frequency to 1/idur in order for a single cycle of the wave to be generated ver the duration of the notek

```
kenv oscili 1, 1/dur, 1
out asig * kenv
```

Using an interpolating oscillator in this context makes the most sense In order to keep the table size down and improve overall sound Quality.

ftgen

Generate a score function table from within the orchestra.

```
gir ftgen ifn, itime, isize, igen, iarga
[, iargb ] [...]
```

Pitch Conversion

Value converters allow the user to express frequency and amplitude in a much more intuitive manner.
Cpspch converts from the pitch class (octave point pitch) to cps (cycles per seconds). Using values in between the ones in the table below will allow us to work microtonally.

NOTE #	Hertz (Hz)	CPSPCH	MIDI NOTE NUMBER
C4	261.626	8.00	60
C#4	277.183	8.01	61
D4	293.665	8.02	62
D#4	311.127	8.03	63
E4	329.628	8.04	64
F4	349.228	8.05	65
F#4	369.994	8.06	66
G4	391.955	8.07	67
G#4	415.305	8.08	68
A4	440.000	8.09	69
A#4	466.164	8.10	70
B4	493.883	8.11	71
C5	523.251	9.00	72

Amplitude Conversion

Ampdb reads a decibel value and converts it into a raw amplitude value.
Note: While the logarithmic dB scale is linear in perception, ampdb performs a direct conversion with no scaling.

```
ampdb(42)       =       125
ampdb(48)       =       250
ampdb(54)       =       500
ampdb(60)       =       1000
ampdb(66)       =       2000
ampdb(72)       =       4000
ampdb(78)       =       8000
ampdb(84)       =       16000
ampdb(90)       =       32000
ampdb(96)       =       64000       ; WARNING: SAMPLES OUT OF RANGE!!!
```