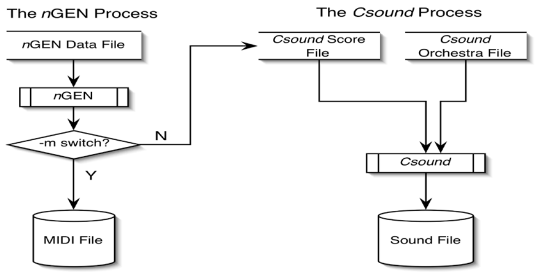
The *n*GEN Quick Reference Card

(Version 2.1.1)



Running *n*Gen from the Command Line*:

Switch	Required Arguments	Comments
-m	none	Output file will be a MIDI file (format 1).
-l	file name	Make log of screen output and put it in “file name” (text file).
-t	none	Write verbose information about tempo changes to console.
-x	file name	Put macro expansions in “file name” (text file).

Example:

```
ngen -m -t -l log.txt -x exp.txt ex1a.gen ex1a.mid
```

will run the program using input from “ex1a.gen” creating MIDI output in “ex1a.mid”, screen output will be logged in “log.txt”, and macro expansion can be viewed in “exp.txt.” Tempo information will be written to the console.

*N.B. The command line version of *n*Gen is available for DOS/Windows, Linux, and Mac OS X.

Instrument Blocks (I-Blocks)

I-Blocks are the guts of the *n*Gen input file and create a *Csound* score-file instrument block (currently MIDI output can only have one I-block). I-blocks contain a *header* and a *body*.

I-Block Header
Each i-block must contain a header in the following format:

```
i<number> = <# of p-fields> <start time> <X>
 {
 ...p-field data...
 }
```

N.B. If the instrument number is negative, the entire i-block will be ignored. (This can be useful for debugging, etc.)

MIDI

If you are going to be creating MIDI files, only 4 p-fields apply:
p2: always the start time of the event (as usual).
p3: always the duration of the event (as usual).
p4: For MIDI this should always be amplitude (0-32767). The value in p4 will be scaled to the range 0-127 automatically. If you happen to use 0-127 it will still be scaled, so stick with 0-32767 always. While this may seem strange, it exists so that Csound files will always convert to MIDI.
p5: For MIDI this should always be pitch. Again these values will be converted to MIDI code automatically. For example, C4 (stored internally by the program as 60), will convert to 48.
Important: All p-fields above p5 will be ignored when creating MIDI files.

where: <number> = the instrument number, <# of p-fields> = the number of p-fields used in the instrument. This is identical to *Csound* where P1 is the instrument number. The highest P# used in your instrument will be the number of p-fields in your i-block. <start time> = the global start time for the instrument (in beats). <X> = the duration of the instrument. If X is a positive real number, it denotes the number of beats for the duration of the i-block’s material. If X is a negative integer, it denotes the number of events that will be calculated.

IMPORTANT: If the data in any p-field is less than that specified in the instrument header’s duration (total time or number of events), the last value in the p-field will be repeated until the end of the note list. If this is not desired see the <> delimiters. If you have included too much data in a particular p-field for the duration of the i-block, it will be truncated (a warning message will be printed when this occurs).

I-Block Body
Each i-block must also contain a body, after the header; the body is enclosed in {}s. There are only two obligatory p-fields in the i-block’s body: P2 (start times) and P3 (durations). The current limit on the number of p-fields that can be contained within a single i-block is 256 (but this may be significantly more than *Csound* can realistically accept). The only limitation on the size of an expanded i-block or the number of i-blocks in a file is machine specific memory.

*n*Gen’s p3 Codes: (p3 specifies duration; the “codes” allow for several special duration possibilites)

p3 Code	Function
0-199.999	When p3 is in this range, the duration of the event is multiplied by the given scaling factor (e.g., 1 leaves all durations as continuing until the start of the next event – i.e., 100% scaling, .5 will make all durations last 50% of the time between both events.) To make events legato, consider setting P3 to 1.05...
200-299.999	Adds a constant amount, x - 200, to the temporal interval (default duration). For example, if x is 200.1, the duration will be the temporal interval between two events plus .1.
300-399.999	Subtracts a constant amount, x - 300, from the temporal interval (default duration). For example, if x is 300.1, the duration will be the temporal interval between two events minus .1. If the value negates the length of the temporal interval, the event turns into a rest (in this case a warning message is printed).
400+ / 1000+	Force x - 400 to be the literal value of the duration (in beats). / Force x - 1000 to be the literal value of the duration (in seconds).

Dynamic Data Functions (DDFs)

Dynamic data functions (DDFs), such as **mo**, will create a stream of changing values over a specified time or number of events, based on a particular algorithm. The data will be generated for a the period found in the “time” field (first field in all DDFs) – negative numbers will create a specific number of events of generated data, while positive number will create a stream of data lasting for a specific number of beats (similar to the i-block header’s duration field).

DDF	Description	Example
ex	Extract previous p-fields; these can optionally be included in an equation (left to right precedence).	`ex(T, 1. [2000 - p4 * .01])`
mo	Moves between two values (v1 and v2) or two ranges (v1a to v1b and v2a to v2b). Additionally, the type of interpolation – linear, exponential, or logarithmic, can be specified and there can be several ”moves” nested in the same time period.	`mo(-10 1.E 1 [5 10])` time % dist v1 v2 ... OR time % dist [range1] [range2] ...
ms	A combination of the **mo** (move) and **se** (sets) commands that allows for the interpolation between two “sets” of values.	`ms(T,1. L [1 2 11][10 20 30])` time % dist [set1] [set2] ...
se / se2	A special case of the random (ra) command. It randomly chooses a succession of elements taken from a “set” of listed materials (this is similar to using **ra** to specify equal weighted distributions of selected values). **se2** will allow fused data as sets (i.e., “chords”).	`no se(10 .9 [c4, cs3, fs4, g5] .1 [df2/bfff2])` time % [set1] % [set2] ...
ra	Used to create a stream of random values over the specified number of beats or events. Each value (or range of values) must be preceded by a percentage value specifying the weighting of that value (or range) and all percentage values must sum to 1.	`ra(T, .5 [g -1 -2] .5 [x 1 2])` time % v1 % v2 OR time % dist [range1] % dist [range2]
rw	The **rw** command is useful for creating sequences of random numbers where each successive number lies within a constrained distance from the previous (similar to 1/f noise).	`rw(T, 1. [0 32767], .25 16000)` time % [range] win sz % start val
ho	The **ho** command is used to “hold” or continue the same value for a specified period.	`ho(45.25, 54.2)` time value

Linear Distribution Flags

Flag	Distribution
e, l, f	exponential, logarithmic, flat (lin)
s,c	sine rise (270-360), sine fall
V*n*	*n*: 1=flat, 2=exp, .5=log, 3=steep exp., .25=steep log

Random Distribution Flags

Flag	Distribution
f	flat (normal)
l	low (near low anchor)
h	high (near high anchor)
g	Gaussian (bell)
x	bilateral exponential (center)
b	beta (near sides)
t	triangular (center)

Commands

Command	Description	Command	Description	Command	Description	Command	Description
#define	Define a macro.	**< ... >**	Put ... in data queue.	**op, opx**	Octave.pch in/out filter.	**te()**	Tempo function (static or dynamic).
#undef	Undefine a macro.	**db**	Decibel in/out filter.	**pf(X)**	P-field scaling factor of X (1=same).	**tr**	Transposition ratio output filter.
#include	Read in a file.	**dv(X)**	Random deviation func. (p-field)	**rd(X)**	Random deviation function (global).	**xx**	Scratch p-field output filter (no output).
$	Call a macro.	**in**	Integer output filter.	**re[X]**	Floating-point output filter.	**/, x**	Repeat last datum.
T	Grab time/events from header.	**no, nox**	Note input filter.	**rh**	Reciprocal duration code input filter.	**|, “,”**	Data separators (optional).
>	Output rest of line to file.	**od**	Octave.decimal out filter.	**rs(X)**	Reset random number seed.	**z**	Use last generated value.

Sample *n*Gen Input File

```
/*
An example nGen file.  This must be saved as ASCII text. You can create this file using a
simple text-editor (NotePad, Vi, Emacs, TextWrangler, etc.).
*/

>f1 0 8192 10 1                                          ;sine wave function
>f2 0 1025  -5 4 1025 .01                                ;index envelope
>f10 0 4097 5 .001 197 1 3997 .001                       ;envelope (sharp attack)
>f11 0 4097 7 0 100 1 3897 1 100 0                       ;(quick ramp)
>f12 0 4097 5 .001 2048 1 2049 .001                      ;(quick ramp)

/* Orchestra Parameters:
p4 = amplitude (0-32767)
p5 = Herz
p6 = C:M ratio
p7 = Spatial position 0-1
p8 = envelope function #
p9 = amount of wet signal (reverb) */

#define X #ds2:e3:g#     ;a macro

te(60)   ;"quarter-note" = 60b.p.m. (the default)

i1 = 9 0 15 {
  ;rhythms
  p2 rh   16x4/4/12x2/24x2/4/12x2/24x2   ;measure 1
          24//12/24//20x5/4              ;measure 2
          16/8./10./20//4/20///10        ;measure 3
          24///8/8./16/2                 ;measure 4

  p3 4      ;each duration is 4x its "intervalic duration"
  p4 10000  ;amp

  p5(hz)   no  ef5/r/d6:f/rx4/en5/rx5/df6                  ;m1
               c2/r/r/c2:b:af3/r//bf4:a5/r/e3:g/ef2/r      ;m2
               d3:cs4:gf5:f6/r//$X/r///fs1:fn2/b2:gs3/r    ;m3
               c5/r/d2:cs6/r//bf4:a5/r                     ;m4

  p6 se(15 1. [1, 2, 1.5, 3, 1.414]

  p7(re2) ra(15 .5 .5 .5 [b 0 1])  ;spatial placement (1 = right)
  p8(in)  10  ;envelope function #
  p9 .1  ;reverb amount
}

>i2 0 20
```

These lines (preceded w/ >) will be copied directly to the score (output) file and will be stripped of the ">".

This is a macro named "X" that contains a chord (the macro definition is contained within the #s).

This line specifies the start of the instrument block (i.e., instrument 1 starts at beat 0 and last for 15 beats). A negative number in the last parameter specifies the # of events to generate.

Macros can be accessed by preceding their name with a "$".

The **hz** output filter will convert internal values (48 = C4) to Hertz values. (Here the **no** input filter has been used to convert note names-octave numbers into the internal values.)

This line is stripped of the > and sent to the *Csound* score-file. It controls the global reverb instrument.

Comments can use the /* */ style for multiple lines or the semicolen ";" for single ine comments.

The current instrument block must end with "}". There may be many i-blocks in a single file.

p2 uses the **rh** input mode to specify reciprocal duration codes (rhythms). The /s are used to separate values but are optional. The data could have been listed in one huge chunk but, instead, I have separated each measure by a line and have been careful to comment. The xs are used to repeat the previous value (slashes may also occur more than once denoting a repeat – as in the SCORE input language). (This example was taken from the piano part from Mikel Kuehn's composition *Between the Lynes* © 1994.)

The **no** input filter allows for note-name octave number input. Rests are indicated by R and will constitute a period of silence attached to the corresponding start times in p2. Formatting is similar to p2.

p6 is the carrier to modulator ratio. Here we use SE to list some "set" values to choose from at random.

In p7 the RE output filter is used to send floating point numbers to the output file (the "2" is a field width specifier). The RA command is used here to generate random spatial values: 50% of the time a .5 (middle) and 50% of the time a random number between 0 (left) and 1 (right). The "b" specifies a beta distribution.

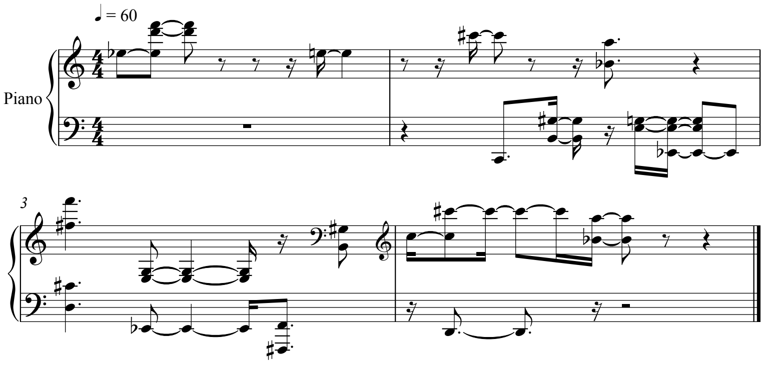
*n*Gen Output File (*Csound* Score File)

p1, p2, p3,p4, p5, p6, p7, p8, p9

```
f1 0 8192 10 1                          ;sine wave function
f2 0 1025  -5 4 1025 .01                ;index envelope
f10 0 4097 5 .001 197 1 3997 .001       ;envelope (sharp attack)
f11 0 4097 7 0 100 1 3897 1 100 0       ;(quick ramp)
f12 0 4097 5 .001 2048 1 2049 .001      ;(quick ramp)
;I-block #1 (i1):
i1    0.000  1.000 10000.000   622.25397     3.000     0.50    10     0.100
i1    0.500  1.000 10000.000  1174.65894     1.500     0.44    10     0.100
i1    0.500  1.000 10000.000  1396.91272     1.000     0.50    10     0.100
i1    2.667  0.667 10000.000   659.25519     1.000     0.00    10     0.100
i1    4.833  0.667 10000.000  1108.73059     3.000     0.02    10     0.100
i1    5.000  0.667 10000.000    65.40639     3.000     0.08    10     0.100
i1    5.667  0.667 10000.000    65.40639     3.000     0.95    10     0.100
i1    5.667  0.667 10000.000   123.47083     1.500     0.50    10     0.100
i1    5.667  0.667 10000.000   207.65236     1.500     0.50    10     0.100
i1    6.200  0.800 10000.000   466.16382     1.000     0.50    10     0.100
i1    6.200  0.800 10000.000   880.00000     3.000     0.50    10     0.100
i1    6.600  0.800 10000.000   164.81377     1.000     0.50    10     0.100
i1    6.600  0.800 10000.000   195.99771     1.414     0.50    10     0.100
i1    6.800  0.800 10000.000    77.78175     1.000     0.05    10     0.100
i1    8.000  1.000 10000.000   146.83240     3.000     0.50    10     0.100
i1    8.000  1.000 10000.000   277.18265     1.414     0.00    10     0.100
i1    8.000  1.000 10000.000   739.98883     3.000     0.37    10     0.100
i1    8.000  1.000 10000.000  1396.91272     1.500     0.20    10     0.100
i1    9.600  0.800 10000.000    77.78175     3.000     0.50    10     0.100
i1    9.600  0.800 10000.000   164.81377     1.414     0.50    10     0.100
i1    9.600  0.800 10000.000   195.99771     3.000     0.13    10     0.100
i1   11.200  0.800 10000.000    46.24930     2.000     0.89    10     0.100
i1   11.200  0.800 10000.000    87.30706     1.000     0.50    10     0.100
i1   11.400  0.800 10000.000   123.47083     1.000     0.50    10     0.100
i1   11.400  0.800 10000.000   207.65236     1.414     0.96    10     0.100
i1   12.000  0.667 10000.000   523.25116     3.000     0.01    10     0.100
i1   12.333  0.667 10000.000    73.41620     1.500     1.00    10     0.100
i1   12.333  0.667 10000.000  1108.73059     2.000     0.17    10     0.100
i1   13.750  1.000 10000.000   466.16382     1.500     0.70    10     0.100
i1   13.750  1.000 10000.000   880.00000     1.414     0.50    10     0.100

;I-block #2 (i1):
i2 0 20
e
```

MIDI File Output (imported to *Sibelius*)



For More Information:
See the *n*GEN HTML Manual
(available online)

The *n*GEN Home Page
http://mikelkuehn.com/index.php/ng

Mikel Kuehn, Author
mikelkuehn.com

Download *n*Gen for FREE!