Quick Reference: *NAPA 4.00* instructions

#*	<any comment on one line>
alias	<alias_name> <target_node>
	<alias_name> <target_variable>
array	(analog) <array_name> '[‘<array_size>‘]' [<“file_pathname”>]
	(digital) <array_name> '[‘<array_size>‘]' [<“file_pathname”>]
	(pointer) <array_name> '[‘<array_size>‘]' <node_name | variable_name | array_name...>
	(pointer) <array_name> '[ ]' <node_name | variable_name | array_name...>
assert	<“text_message”> <C_Boolean_expression>
call	void <C_function_returning_void>
	<return_value> <C_expression>
command_line	<variable_name...> | fs | ts | void
comment	<”text_message”>
data	<“file_name”> <parameters…>
debug	[<debug_level_number | identifier…>]
decimate	[fs] <decimation_factor> [<decimation_initial_value>]
declare	(analog) <identifier…>
	(digital) <identifier…>
	(char) <identifier…>
	( ) <identifier…>
	(constant) <identifier…>
	(true) <a_function_returning_a_boolean…>
directive	<C_preprocessor_macro_identifier> [<value>]
drop	[fs] <C_Boolean_expression>
dump	<“file_pathname”> [when <C_Boolean_expression_of_events>]
dvar	<variable_name> [<initial_value>] [&update | &constant] [&export]
error	<”text_message”>
event	<event_name> [<C_expression_returning_a_boolean>]
	<event_name> (new) [<C_expression>]
export	<global_variable_name>
	<node_name>
	<variable_name>
format	(analog) <“C_double_output_format”> | S | M | L
	(digital) <“C_long_long_output_format”> | S | M | L
	(string) <“C_string_output_format”> | S | M | L
ganging	<array_name> '[‘<array_size>‘]' <nod_nam | var_nam | number | string | array_nam...>
	<array_name> '[ ]' <nod_nam | var_nam | number | array_nam...>

fs	[<sampling_frequency>]
gateway	[<count_down>]
header	<“file_pathname”> [(noexpand)]
	<“file_pathname”> (expand)
init	void <C_function_returning_void>
	<variable_name> <C_function>
	<node_name> <C_expression>
inject	<node_name> <C_function>
input	<“file_pathname”> <variable_name...>
	“stdin“ <variable_name...>
cell_interface	<$node | $variable | $parm...>
data_interface	<$variable | $parm...>
interlude	<num_of_init_samples>
	<num_of_init_samples min> <num_of_init_samples max>
	<num_of_init_samples min> <num_of_init_samples max> <num_of_init samples res>
interpolate	[fs] <interpolation_factor>
ivar	<variable_name> [<“initial_value”>] [&update | &constant] [&export]
load	<“file_pathname”>
napa_version	<version_id>
node	<node_name> <node_kind> <node_name | variable_name | parameter...>
	void <node_kind> <node_name | variable_name | parameter...>
nominal	fs
opcode	<alu_name> <opcode_number> [<template>]
output	<“path_nam”> <node_nam | var_nam...> [when <boolean_expression_of_events>]
	<string_var_nam> <node_nam | var_nam...> [when <boolean_expression_of_events>]
	“stderr“ <node_nam | var_nam...> [when <boolean_expression_of_events>]
	“stdout“ <node_nam | var_nam...> [when <boolean_expression_of_events>]
ping	[“stderr”]
	<“path_nam”>
	<string_var_nam>
post	[<label>] <function_id> <file_name> [<parameters>]
	<label> void <file_name>
random_seed	<[- | +] seed_number>
restart	
string	<variable_name> [<“initial_value”>]
stuck	<node_name> <C_expression_returning_a_number>
synchronize	(yes) | (no)
terminate	[<C_Boolean_expression>]
title	<“some one-line text”>
tool	<user_defined_itool> <parameters>

ts	[<sampling_period>]
update	<variable_name> [<C_expression>] [when <boolean_expression_of_events>]
void	<file_name>
warning	<”text_message”>

Quick Reference: *NAPA 4.00* nodes

adc	<num_level> <input_node> <reference_node>	*R → I*
algebra	<C_expression>	*Chameleonic*
alu	<alu_name> <opcode node> <input_node...>	*Chameleonic*
and	<input_node> <input_node...>	*I → I*
average	<[- | +]input_node> <[- | +]input_node...>	*R → R*
bshift	<number> <input_node>	*I → I*
	<[- | +]shift_variable> <input_node>	*I → I*
	<[- | +]shift_node> <input_node>	*I → I*
btoi	<input_node> <input_node...>	*I → I*
buffer	<input_node>	*I → I*
bwand	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
bwbuffer	<input_node>	*I → I*
bwinv	<input_node>	*I → I*
bwnand	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
bwnor	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
bwnot	<input_node>	*I → I*
bwor	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
bwxnor	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
bwxor	<hexavalued_mask> <input_node...>	*I → I*
	<input_node> <input_node...>	*I → I*
cell	<instance_name> <“file_name”> <parameter_list>	*N/A*
change	<input_node>	*X → I*
clip	<[- | +]threshold_low> <[- | +]threshold_high> <input_node>	*Chameleonic*
clock	< [“pattern_descriptor_aperiodic”**.**]“pattern_descriptor_periodic”>	*→ I*
comp	<[- | +]positive_input_node> <[- | +]negative_input_node>	*X → I*
	<[- | +]positive_input_node> <[- | +]variable>	*X → I*
	<[- | +]variable> <[- | +]negative_input_node>	*X → I*
	<[- | +]positive_input_node> <[- | +]number>	*X → I*
	<[- | +]number> <[- | +]negative_input_node>	*X → I*

const	[(digital)] <C_expression>	→ *I*
	(analog) <C_expression>	→ *R*
copy	<[- | +]input_node>	*Chameleonic*
cosine	<[- | +]offset> <amplitude> <frequency> <[- | +]phase>	*X* → *R*
dac	<num_level> <input_node> <reference_node>	*I* → *R*
dalgebra	<C_expression_cast_to_real>	*X* → *R*
dc	[(analog)] <C_expression>	→ *R*
	(digital) <C_expression>	→ *I*
delay	<[- | +]input_node>	*Chameleonic*
	<number> <[- | +]input_node>	*Chameleonic*
	<delays_var> <[- | +]input_node>	*Chameleonic*
differentiator	<[- | +]input_node>	*Chameleonic*
div	<[- | +]input_node> <[- | +]input_node>	*Chameleonic*
	<[- | +]input_node> <[- | +]variable>	*Chameleonic*
	<[- | +]input_node> <[- | +]number>	*Chameleonic*
dtoi	<input_node>	*R* → *I*
dtool	<dtool_name> [<parameter_list>]	*X* → *R*
duser	<duser_name> [<parameter_list>]	*X* → *R*
equal	<[- | +]input_node> <[- | +]input_node>	*X* → *I*
	<[- | +]input_node> <[- | +]variable>	*X* → *I*
	<[- | +]variable> <[- | +]input_node>	*X* → *I*
	<[- | +]input_node> <[- | +]number>	*X* → *I*
	<[- | +]number> <[- | +]input_node>	*X* → *I*
fzand	<input_node> <input_node...>	*R* → *R*
fzbuffer	<input_node> <input_node>	*R* → *R*
fzinv	<input_node>	*R* → *R*
fzor	<input_node> <input_node...>	*R* → *R*
fznand	<input_node> <input_node...>	*R* → *R*
fznor	<input_node> <input_node...>	*R* → *R*
fznot	<input_node>	*R* → *R*
gain	<[- | +]number> <input_node>	*Chameleonic*
	<[- | +]gain_variable> <input_node>	*Chameleonic*
generator	<instance_name> <“generator_name”> <parameter_list>	*N/A*
hold	<control_node> <input_node>	*Chameleonic*
ialgebra	<C_expression_cast_to_int>	*X* → *I*
integrator	<[- | +]input_node>	*Chameleonic*

inv	<input_node>	*I → I*
itob	<bit_rank> <input_node>	*I → I*
itod	<input_node>	*I → R*
itool	<itool_name> [<parameter_list>]	*X → I*
iuser	<iuser_name> [<parameter_list>]	*X → I*
latch	<set_input_node> <reset_input_node>	*I → I*
lshift	<number> <input_node>	*I → I*
	<shift_variable> <input_node>	*I → I*
	<shift_node> <input_node>	*I → I*
max	<[- | +]input_node> <[- | +]input_node...>	*Chameleonic*
merge	<[- | +]input_node_from_seg_a> <[- | +]input_node_from_seg_b...>	*Chameleonic*
min	<[- | +]input_node> <[- | +]input_node...>	*Chameleonic*
mod	<[- | +]input_node> <[- | +]input_node>	*Chameleonic*
	<[- | +]input_node> <[- | +]variable>	*Chameleonic*
	<[- | +]input_node> <[- | +]number>	*Chameleonic*
muller	<input_node> <input_node...>	*I → I*
mux	<control_node> <[- | +]input_node_0> <[- | +]input_node_1...>	*Chameleonic*
	<control_variable> <[- | +]input_node_0> <[- | +]input_node_1...>	*Chameleonic*
nand	<input_node> <input_node...>	*I → I*
noise	<[- | +]DC_level> <noise_density_level>	*→ R*
nor	<input_node> <input_node...>	*I → I*
not	<input_node>	*I → I*
offset	<[- | +]number> <input_node>	*Chameleonic*
	<[- | +]offset_variable> <input_node>	*Chameleonic*
or	<input_node> <input_node...>	*I → I*
osc	<[- | +]offset> <amplitude> <frequency> <[- | +]phase>	*X → R*
poly	<[- | +]$coeff_0$> [<[- | +]$coeff_i$> …] <[- | +]input_node>	*Chameleonic*
prod	<[- | +]input_node> <[- | +]input_node...>	*Chameleonic*
quant	<input_node> <[- | +]input_node>	*Chameleonic*
	<variable> <[- | +]input_node>	*Chameleonic*
	<constant> <[- | +]input_node>	*Chameleonic*
ram	<name'['addr_node']'> <CS_node> <control_node> <W_node>	*Declared*
ram2	<name'['addr_node']'> <CS_node> <control_node> <W_node>	*Declared*
rect	<input_node>	*Chameleonic*
register	<control_node> <input_node>	*Chameleonic*
relay	<control_node> <input_node>	*Chameleonic*

	<control_variable > <input_node>	*Chameleonic*
	<control_node> <input_node> < [- | +] setting_variable>	*Chameleonic*
	<control_variable> <input_node> < [- | +] setting_variable>	*Chameleonic*
	<control_node> <input_node> < [- | +] setting_constant>	*Chameleonic*
	<control_variable> <input_node> < [- | +] setting_constant>	*Chameleonic*
rip	<hexavalued_mask> <input_node>	*I → I*
rom	<name'['addr_node']'> <CS_node>	*Declared*
rom2	<name'['addr_node']'> <CS_node>	*Declared*
rshift	<number> <input_node>	*I → I*
	<shift_variable> <input_node>	*I → I*
	<shift_node> <input_node>	*I → I*
rshift1	<number> <input_node>	*I → I*
	<shift_variable> <input_node>	*I → I*
	<shift_node> <input_node>	*I → I*
rshift2	<number> <input_node>	*I → I*
	<shift_variable> <input_node>	*I → I*
	<shift_node> <input_node>	*I → I*
sign	<input_node>	*X → I*
sine	<[- | +]offset> <amplitude> <frequency> <[- | +]phase>	*X → R*
square	<[- | +]offset> <amplitude> <frequency> <delay> [<duty_cycle>]	*→ R*
step	<level1> <level2> <transition_time> [<transition_time>]	*→ R*
sub	<[- | +]input_node> <[- | +]input_node>	*Chameleonic*
	<[- | +]input_node> <[- | +]number>	*Chameleonic*
sum	<[- | +]input_node> <[- | +]input_node...>	*Chameleonic*
toggle	<input_node>	*I → I*
track	<control_node> <input_node>	*Chameleonic*
triangle	<[- | +]offset> <amplitude> <frequency> <delay> [<duty_cycle>]	*→ R*
trig	<number> <input_node> [(dual)]	*X → I*
	<number> <input_node> (positive)	*X → I*
	<number> <input_node> (negative)	*X → I*
uadc	<num_level> <input_node> <reference_node>	*R → I*
udac	<num_level> <input_node> <reference_node>	*I → R*
wsum	<weight> <input_node> ... <weight> <input_node>	*Chameleonic*
xnor	<input_node> <input_node...>	*I → I*
xor	<input_node> <input_node...>	*I → I*
zero	<decimation_factor> <decimation_offset> <input_node>	*Chameleonic*

Quick Reference: *NAPA 4.00* macros functions

ABS(x)
SIGN(x)
MIN(x,y)
MAX(x,y)
CLIP(x,l,h)

ISINSIDE(x,l,h)
ISOUTSIDE(x,l,h)
ISEQUAL(x,y)
ISNOTEQUAL(x,y)
ISSMALL(x)
ISNOTSMALL(x)
ISEVEN(x)
ISODD(x)
ISINTEGER(x)
ISTIME(t)

POWEROF2(n)
MODULO(x,y)
SIN(x)
COS(x)
SQR(x)
SQRT(x)
LOG(x)
POW(x,y)
ROOT(x,y)
LOG10(x)
POW10(y)
D2I(x)
I2D(n)
DB2LIN(x,r)
LIN2DB(x,r)
DB2POW(x,r)
POW2DB(x,r)
RAD2DEG(x)
DEG2RAD(x)
LENGTH(s)

LINDOMAIN(c,b,e)
LOGDOMAIN(c,b,e)
LINSWEEP(c,b,e,n)
LOGSWEEP(c,b,e,n)

RAND_01()
RAND_01_X()

FSS(n)
STS(n)
PS(n)
SEGMENT_CONDITION(n)
TIMER(n)
IO_MANAGER(c,f,n,s,t)
ISOPTION(f,i,o)
ISNOTOPTION(f,i)
ISDELAYED(f,i)

Quick Reference: The *NAPA* file system

Absolute reference	"/ "
Reference to a **generic** library	< >
Reference to the **root** directory	" "
Reference to the **main** directory	"~/"
Reference to the **current cell** directory	"./"

A *generic library* is one of three particular libraries: header, cell and generator libraries

The *root directory* is the working directory from where ***NAPA*** compiler has been called.

The *main directory* is the directory containing the ***NAPA*** main netlist.

The *current cell directory* is the directory containing the cell currently processed.

Quick Reference: *file naming recommendations*

:***NAPA*** main netlist file	xxxx**.nap**	
NAPA netlist of a cell	xxxx**.net**	
MAC output	xxxx**.tmp**	(NAPA preprocessor)
MAXIMA package	xxxx.**mac**	
C code generated by ***NAPA***	xxxx**.c**	
Executable binary code	no suffix	(UNIX platform)
	xxxx**.exe**	(DOS platform)
NAPA header file and user's profile	xxxx**.hdr**	
NAPA data cells	xxxx.**dat**	
Simulation output file	xxxx**.out**	
NAPA log file	xxxx**.log**	
NAPA dump file	xxxx**.dmp**	
NAPA ping file	xxxx**.png**	
NAPA load file	xxxx**.ini**	
NAPA RAM initialization file	xxxx**.ram**	
NAPA ROM description file	xxxx**.rom**	
NAPA generator (executable)	no suffix	(UNIX platform)
	xxxx**.exe**	(DOS platform)
NAPA generator (output cell)	xxxx**.gen**	
Graphics front end directives	xxxx.**plt**	(Gnuplot...)