
Quick Reference: *NAPA 4.00* instructions

#	<any comment on one line>	NB: <i>not compatible with the MAC macroprocessor</i>
#*	<any comment on one line>	NB: <i>preferred</i>
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...>
interface	<\$node \$variable \$parm...>
cell interface	<\$node \$variable \$parm...>
data interface	<\$variable \$parm...>
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
num_initial	<number_of_initial_samples>
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>	$R \rightarrow I$
algebra	<C_expression>	<i>Chameleonic</i>
alu	<alu_name> <opcode node> <input_node...>	<i>Chameleonic</i>
and	<input_node> <input_node...>	$I \rightarrow I$
average	<[- +]input_node> <[- +]input_node...>	$R \rightarrow R$
bshift	<number> <input_node>	$I \rightarrow I$
	<[- +]shift_variable> <input_node>	$I \rightarrow I$
	<[- +]shift_node> <input_node>	$I \rightarrow I$
btoi	<input_node> <input_node...>	$I \rightarrow I$
buffer	<input_node>	$I \rightarrow I$
bwand	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
bwbuffer	<input_node>	$I \rightarrow I$
bwinv	<input_node>	$I \rightarrow I$
bwnand	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
bwnor	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
bwnot	<input_node>	$I \rightarrow I$
bwor	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
bwxnor	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
bwxor	<hexavalued_mask> <input_node...>	$I \rightarrow I$
	<input_node> <input_node...>	$I \rightarrow I$
cell	<instance_name> <"file_name"> <parameter_list>	<i>N/A</i>
change	<input_node>	$X \rightarrow I$
clip	<[- +]threshold_low> <[- +]threshold_high> <input_node>	<i>Chameleonic</i>
clock	< ["pattern_descriptor_aperiodic".] "pattern_descriptor_periodic">	$\rightarrow I$
comp	<[- +]positive_input_node> <[- +]negative_input_node>	$X \rightarrow I$
	<[- +]positive_input_node> <[- +]variable>	$X \rightarrow I$
	<[- +]variable> <[- +]negative_input_node>	$X \rightarrow I$
	<[- +]positive_input_node> <[- +]number>	$X \rightarrow I$
	<[- +]number> <[- +]negative_input_node>	$X \rightarrow I$

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

inv	<input_node>	<i>I → I</i>
itob	<bit_rank> <input_node>	<i>I → I</i>
itod	<input_node>	<i>I → R</i>
itool	<itool_name> [<parameter_list>]	<i>X → I</i>
iuser	<iuser_name> [<parameter_list>]	<i>X → I</i>
latch	<set_input_node> <reset_input_node>	<i>I → I</i>
lshift	<number> <input_node>	<i>I → I</i>
	<shift_variable> <input_node>	<i>I → I</i>
	<shift_node> <input_node>	<i>I → I</i>
max	<[- +]input_node> <[- +]input_node...>	<i>Chameleonic</i>
merge	<[- +]input_node_from_seg_a> <[- +]input_node_from_seg_b...>	<i>Chameleonic</i>
min	<[- +]input_node> <[- +]input_node...>	<i>Chameleonic</i>
mod	<[- +]input_node> <[- +]input_node>	<i>Chameleonic</i>
	<[- +]input_node> <[- +]variable>	<i>Chameleonic</i>
	<[- +]input_node> <[- +]number>	<i>Chameleonic</i>
muller	<input_node> <input_node...>	<i>I → I</i>
mux	<control_node> <[- +]input_node_0> <[- +]input_node_1...>	<i>Chameleonic</i>
	<control_variable> <[- +]input_node_0> <[- +]input_node_1...>	<i>Chameleonic</i>
nand	<input_node> <input_node...>	<i>I → I</i>
noise	<[- +]DC_level> <noise_density_level>	<i>→ R</i>
nor	<input_node> <input_node...>	<i>I → I</i>
not	<input_node>	<i>I → I</i>
offset	<[- +]number> <input_node>	<i>Chameleonic</i>
	<[- +]offset_variable> <input_node>	<i>Chameleonic</i>
or	<input_node> <input_node...>	<i>I → I</i>
osc	<[- +]offset> <amplitude> <frequency> <[- +]phase>	<i>X → R</i>
poly	<[- +]coeff ₀ > [<[- +]coeff _i > ...] <[- +]input_node>	<i>Chameleonic</i>
prod	<[- +]input_node> <[- +]input_node...>	<i>Chameleonic</i>
quant	<input_node> <[- +]input_node>	<i>Chameleonic</i>
	<variable> <[- +]input_node>	<i>Chameleonic</i>
	<constant> <[- +]input_node>	<i>Chameleonic</i>
ram	<name['addr_node']> <CS_node> <control_node> <W_node>	<i>Declared</i>
ram2	<name['addr_node']> <CS_node> <control_node> <W_node>	<i>Declared</i>
rect	<input_node>	<i>Chameleonic</i>
register	<control_node> <input_node>	<i>Chameleonic</i>
relay	<control_node> <input_node>	<i>Chameleonic</i>

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

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)
NIS(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*

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