The Surf-Hippo Reference Manual
Version 3.0 β

Lyle J Graham*
Unite de Neurosciences Integratives et Computationnelles
Institut Federatif de Neurobiologie Alfred Fessard, CNRS
91198 Gif-sur-Yvette, France

March, 2002

Abstract

This document comprises the reference manual for the Surf-Hippo Neuron Simulation System, a public domain circuit simulation package written in Lisp for Unix workstations and PCs that is used to investigate morphometrically and biophysically detailed models of single neurons and networks of neurons.

Companion volume to the Surf-Hippo User Manual.

http://www.cnrs-gif.fr/iaf/iaf9/surf-hippo.html

*lyle@cogni.iaf.cnrs-gif.fr, Tel: (331) 69 82 34 13, Fax: (331) 69 82 34 27

Contents

1 Introduction 5

2 SYS Source File: declare.lisp 6

3 SYS Source File: biophysics-declare.lisp 25

4 SYS Source File: structures.lisp 28

5 SYS Source File: structure-macros.lisp 33

6 SYS Source File: models.lisp 34

7 SYS Source File: create-models.lisp 42

8 SYS Source File: declare-2.lisp 42

9 SYS Source File: element-functions-0.lisp 42

10 SYS Source File: element-functions-1.lisp 42

11 SYS Source File: element-functions-2.lisp 48

12 SYS Source File: math.lisp 54

13 SYS Source File: statistics.lisp 55

14 SYS Source File: fft.lisp 56

15 SYS Source File: randoms.lisp 56

16 SYS Source File: renewal-process.lisp 58

17 SYS Source File: waveforms.lisp 58

18 SYS Source File: misc.lisp 65

19 SYS Source File: debug.lisp 66

20 SYS Source File: pump-preliminaries.lisp 67

21 SYS Source File: conc-int.lisp 67

22 SYS Source File: biophysics.lisp 69

23 SYS Source File: sim.lisp 71

24 SYS Source File: circuit-input.lisp 71

25 SYS Source File: node.lisp 72

26 SYS Source File: soma.lisp 73

27 SYS Source File: segment.lisp 74

28 SYS Source File: source.lisp 75

29 SYS Source File: isource.lisp 78

30 SYS Source File: vsource.lisp 78

31 SYS Source File: electrode.lisp 78

32 SYS Source File: general-membrane-elements.lisp 79

33 SYS Source File: channel.lisp 80

34 SYS Source File: particle.lisp 81

35 SYS Source File: markov-particle.lisp 81

36 SYS Source File: conc-part.lisp 82

37 SYS Source File: synapse.lisp 82

38 SYS Source File: light-synapse-functions.lisp 84

39 SYS Source File: synapse-events.lisp 84

40 SYS Source File: buffer.lisp 85

41 SYS Source File: pump.lisp 85

42 SYS Source File: axon.lisp 85

43 SYS Source File: event-generators.lisp 86

44 SYS Source File: cell.lisp 86

45 SYS Source File: cable-functions.lisp 88

46 SYS Source File: trees.lisp 91

47 SYS Source File: print.lisp 93

48 SYS Source File: analysis.lisp 94

49 SYS Source File: store-plot-data.lisp 94

50 SYS Source File: histology-hack.lisp 95

51 SYS Source File: user-cell-graphics.lisp 95

52 SYS Source File: sparse-data.lisp 96

53 SYS Source File: colorizing.lisp 96

54 SYS Source File: plot.lisp 97

55 SYS Source File: 3dplot.lisp 98

56 SYS Source File: calc-lte-ratio.lisp 98

57 SYS Source File: init.lisp 98

58 SYS Source File: step.lisp 98

59 SYS Source File: hacks.lisp 99

60 SYS Source File: raster-plot.lisp **100**

61 SYS Source File: protocols.lisp **100**

62 SYS Source File: sample-cells.lisp **102**

63 SYS Source File: ntscable.lisp **102**

64 LOADERS Source File: main-loader.lisp **103**

65 GUI Source File: macros.lisp **104**

66 GUI Source File: math.lisp **105**

67 GUI Source File: strings.lisp **107**

68 GUI Source File: sequences.lisp **108**

69 GUI Source File: windows-hack.lisp **110**

70 GUI Source File: print-windows.lisp **113**

71 GUI Source File: files.lisp **115**

72 GUI Source File: show-image.lisp **116**

73 GUI Source File: plot-hack.lisp **116**

74 GUI Source File: plot-hack-top.lisp **118**

Index **120**

1 Introduction

This document covers most of the Surf-Hippo functions, global variables and macros that will be required by the user. Specifically, this includes a subset of those forms which include documentation strings. This reference is organized according to source file.

Some functions described in the User Manual are not included here.

SYS Source Files

2 SYS Source File: declare.lisp

temperature *300.0* [*Variable*]

Temperature of the simulation in degrees Kelvin. In general this variable should not be set by the user to set the temperature – set *TEMP−CELCIUS* instead.

temp-celcius *27.0* [*Variable*]

Temperature of the simulation in degrees Celcius. This is the global variable that should be changed by the user for setting the temperature.

circuit *""* [*Variable*]

The name of the circuit function or file (string) (w/o file extension).

circuit-function *""* [*Variable*]

The name of the just loaded circuit function.

initialize-before-next-circuit *t* [*Variable*]

Initialize and clear loaded circuit before next circuit. If NIL, then new circuit will be added to existing one.

multiple-source-circuit *nil* [*Variable*]

When more than one file or function defined the current circuit.

circuit-loaded *nil* [*Variable*]

As soon as any circuit is loaded, this is T.

circuit-drawn *nil* [*Variable*]

As soon as any circuit is drawn, this is T.

simulation-name *""* [*Variable*]

Automatically generated name for the current simulation.

simulation-initialized *nil* [*Variable*]

T when circuit has been initialized for simulation.

simulation-in-progress *nil* [*Variable*]

T while a simulation is running.

simulation-started *nil* [*Variable*]

NIL until the first simulation of the loaded circuit has started.

simulation-finished *nil* [*Variable*]

T as soon as a new circuit simulation is finished.

session-name *""* [*Variable*]

A session name can be used to delinate a series of experiments.

surf-interactive *t* [*Variable*]

True if the program is being run interactively.

write-log-file *nil* [*Variable*]

Keep a running log file of all output to the Lisp window.

kill-extra-messages *nil* [*Variable*]

For text−wise silent simulations.

suppress-element-creation-messages *nil* [*Variable*]

Suppresses various non−fatal messages generated during element creation.

kill-all-output *nil* [*Variable*]

For suppressing all output.

beep-after-surf *t* [*Variable*]

Beep after every simulation.

scrub-and-gc-on-global-initialization *nil* [*Variable*]

When simulator is initialized for a new circuit definition, run the function SCRUB−AND−GC.
** NOT VERIFIED **

log-gc-to-file *nil* [*Variable*]

When true [default NIL], GC messages will be written to a text file.

beep-after-gc *t* [*Variable*]

When true [default], GC will beep when done. Useful signal for long simulations to verify machine is breathing.

show-time-remaining *t* [*Variable*]

Display a simulation timer window. Best not used when running series of fast simulations.

show-time-remaining-update-time *1* [*Variable*]

Time (actual) in seconds between time window updates [integer>0].

always-clear-models *nil* [*Variable*]

Wipe out the previous list of models whenever a circuit is read in.

always-clear-types *nil* [*Variable*]

Wipe out the previous element type hash tables − synapse, channel, particle, etc. − whenever a circuit is read in.

input-is-function *t* [*Variable*]

Whether the circuit description is a compiled function, or the name of file. Normally set automatically.

default-cell-type-name *nil* [*Variable*]

If there is no specified cell type for a newly created cell, for example when loading an anatomy file−based cell, reference this type.

cell-name-suffix *nil* [*Variable*]

When non−NIL, is automatically added as a suffix to the name of a cell created by CREATE−CELL.

next-cell-name *nil* [*Variable*]

If non−NIL, then this is used for the name of the next created cell, overriding any other specfication. This is cleared after CREATE−CELL is called.

add-cell-name-to-segs *nil* [*Variable*]

Add cell name to segment name, unless *USE−SIMPLE−NAMES* is T.

circuit-file-type *:lisp* [*Variable*]

Pertaining to the circuit definition file. Values include :NEUROLUCIDA and :LISP.

circuit-catalog-functions *nil* [*Variable*]

A list of circuit functions that can be selected from the input menus instead of typing them in.

circuit-parts *nil* [*Variable*]

If the loaded circuit is composed of more than one function and/or file then this list includes the names of the components.

loaded-circuit-parts *nil* [*Variable*]

These are the circuits that have actually been loaded.

circuit-source *:catalog_function* [*Variable*]

A symbol which says how the circuit was loaded into the system. Possible values include :Catalog_Function, :Function, or :File.

use-simple-names *nil* [*Variable*]

When true, cells, membrane element and segment names are just integers.

prompt-for-alternate-element-names *t* [*Variable*]

Prompt for alternate element names of duplicate elements.

allow-duplicate-synaptic-connections *t* [*Variable*]

Allow duplicate synaptic connections.

allow-duplicate-elements *t* [*Variable*]

Unless T, only one synapse, channel or other membrane element of a given type can be added to the same cell element.

minimal-capacitance *10.0e−7* [*Variable*]

Temporary capacitance value used for cell elements during low capacitance steady−state determination [nF].

user-save-data-functions *'()* [*Variable*]

This is a list of user−defined functions, each of which is called after plot data is saved.

user-output-data-functions *'()* [*Variable*]

This is a list of user−defined functions, each of which is called by SIM−OUTPUT.

enable-channels *t* [*Variable*]

Enables evaluation of all channels, pumps, and concentration integrators.

only-load-passive *nil* [*Variable*]

When T, both channel and synapse creation is blocked when loading a new circuit.

enable-reorder-elements *t* [*Variable*]

Some types of elements (e.g. channels, synapses) must be reordered if instances are created or destroyed. When T all elements are reordered whenever create−element is called. Reordering is always made, if necessary, at the start of every simulation.

circuit-directory *""* [*Variable*]

Default is given by value of *SURF−USER−DIR*.

add-simulation-to-filenames *nil* [*Variable*]

For files written by the simulator.

data-directory *""* [*Variable*]

Default given by concatenation of *SURF−USER−DIR* and /data/.

plot-directory *""* [*Variable*]

Default given by concatenation of *SURF−USER−DIR* and /plot/.

last-simulation-file-path *nil* [*Variable*]

The name of the last file written (w/o type).

minimum-source-transition-time *0.001* [*Variable*]

Minimum pulse transition time in milliseconds for PWL sources. If this is too small then source waveforms can be distorted.

consider-isource-drop *nil* [*Variable*]

For the recorded voltage on somas and segments that have current sources, consider the IR drop across the source. Used in the function RECORDED−NODE−VOLTAGE.

lte-node-criterium *:all* [*Variable*]

Determines which circuit nodes will be considered for the LTE estimate. Options include :ALL (default), :SOMAS, :CHANNELS, :SYNAPSES, :VSOURCES, :ISOURCES, :AXONS, or a list of circuit elements that may or may not include the afore−mentioned keywords. If :ALL, then include all circuit nodes with externally−driven elements (e.g. sources or synapses or channels). If :SOMAS, then include only somas. If :CHANNELS, :SYNAPSES, :VSOURCES, :ISOURCES or :AXONS, include only those nodes with the appropriate elements.

num-nodes *0* [*Variable*]

The number of electrical circuit nodes in the circuit. Not set by user.

archive-variable-list *'()* [*Variable*]

This is a list of sublists, each of which reference variable symbol names from specific simulations that have been loaded from a data file. The format is:

```
(....
 (circuit-and-simulation-name time-variable-symbol
  (element-data-variable-symbol data-type)
  ...
  (element-data-variable-symbol data-type))
 ....
)
```

archive-session-results *'()* [*Variable*]

Set by results files (see analysis.doc). The contents of this list is typically results of analysis done on circuit data (waveforms) at the end of a simulation, rather than the raw data.

file-output-variable-list *'()* [*Variable*]

This is a list of the variables and their properties. Each entry has the following format:

```
(VAR-SYMB CIRCUIT-ELEMENT DATA-SLOT)
```

simulation-print-detail *:terse* [*Variable*]

Amount of detail for PRINT−CIRCUIT at the start of every simulation, including :NONE :TERSE :MEDIUM :FULL and :FULL_WITH_SEGMENTS.

save-simulation-data *t* [*Variable*]

Enables saving of simulation data.

save-simulation-data-to-file *nil* [*Variable*]

Enables saving of simulation data to file.

documented-user-variables *nil* [*Variable*]

An explicit list of global variables that will printed out by PRINT−CIRCUIT. See also *DOCUMENT−ALL−NEW−VA

document-all-new-variables *nil* [*Variable*]

When T, PRINT−CIRCUIT will print out any bound variables that were either defined after initialization (in SURF package), or in *DOCUMENTED−USER−VARIABLES*.

enable-print-documented-user-variables *t* [*Variable*]

Enable printing of *DOCUMENTED−USER−VARIABLES*.

colorize-simulation *nil* [*Variable*]

Enable colorization of simulation in some or all histology windows.

enable-colorize-time *nil* [*Variable*]

Enable time display in colorized histology windows.

enable-colorize-scale *nil* [*Variable*]

Enable color scale display in colorized histology windows.

enable-sparse-data *nil* [*Variable*]

When non−NIL, data from all the circuit elements are stored on a (typically sparse) time grid stored in *SPARSE−DATA−TIMES*.

sparse-data-step *0.5* [*Variable*]

Time step target for *SPARSE−DATA−TIMES*.

include-channel-type-comment-in-particle-plots *nil* [*Variable*]

Include channel type Gbar and E−rev in particle type plots.

motion-snapshots *5* [*Variable*]

Number of moving stimulus snapshots for histology graphics.

label-stimulus-times *t* [*Variable*]

If synapse light stimulus drawn, label times of stimulus snapshots.

hard-copy-screen *nil* [*Variable*]

Hardcopy screen after each simulation.

create-new-simulation-plots *nil* [*Variable*]

Create a new set of plot windows for each simulation.

massage-element-plot-labels *t* [*Variable*]

When non−NIL, elements with simple integers as names are referenced in data plots with elaboration.

plot-standard-windows *t* [*Variable*]

Enable the standard plotting. Does not affect simulation data storage.

hide-plot-windows *nil* [*Variable*]

Hide simulation plots, even if they're created.

save-conductances-normalized *nil* [*Variable*]

Save element conductances in normalized form.

plot-channels-by-major-ion *t* [*Variable*]

Plot channels by major ion.

group-plot-data-by-cell *t* [*Variable*]

For a given type of data (e.g. voltage, conductance), all traces associated with each cell in the circuit are displayed in their own window or windows.

group-plot-data-by-cell-type *t* [*Variable*]

For a given type of data (e.g. voltage, conductance), all traces associated with each cell type in the circuit are displayed in their own window or windows.

use-simulation-name-for-simulation-plot-titles *t* [*Variable*]

Base simulation plot titles on the *SIMULATION−NAME*, which changes with every simulation, or if NIL then on *CIRCUIT*, which is normally constant for a given loaded circuit.

simulation-plot-window-comment *nil* [*Variable*]

When a string, comment added to all simulation plot windows.

simulation-plot-window-comment-position *:lower−right* [*Variable*]

Position for *SIMULATION−PLOT−WINDOW−COMMENT* that is added to all simulation plot windows.

traces-per-plot *6* [*Variable*]

Unless equal to 0, constrains the number of traces per plot window.

max-num-traces-for-plot-trace-labels *6* [*Variable*]

When non−nil, if the number of traces in a simulation plot is more than this number, then the plot trace labels will be suppressed.

save-data-step *2* [*Variable*]

Data and time are saved onto the appropriate lists every *SAVE−DATA−STEP* time steps.

save-data-called-p *nil* [*Variable*]

Set T after the first time element data is saved during a simulation.

plot-total-concs-separately *t* [*Variable*]

For plotting out concentration integrator total concentrations.

plot-node-elements *'()* [*Variable*]

A reference or list of reference, for each everything on the associated cell node will be plotted.

plot-total-conductances-p *nil* [*Variable*]

Whether to collect and plot total conductances as defined in *PLOT−TOTAL−CONDUCTANCES*.

plot-total-conductances *nil* [*Variable*]

Summed conductances over cells and/or membrane element types are plotted when this is non−NIL. The format is a list of atoms or lists, as follows:

```
:ALL
TYPE
CELL
(CELL TYPE TYPE ...)
(CELL :ALL)
```

where TYPE refers to a synapse or channel type and CELL refers to a cell or cell type. If a cell or cell type is indicated, then the total linear membrane conductance is saved for that cell or for all cells of the cell type, as appropriate. If a cell or cell type is in a list followed by synapse or channel types, then the linear membrane conductance is summed with the conductance of all the synapses or channels of the indicated types. If a cell or cell type is in a list followed by :ALL, then the summation is taken over all synapse and channel types in a given cell or cells of a cell type. In general, use the function SETUP−PLOT−TOTAL−CONDUCTANCES to set up this variable. If the only entry is :ALL, then the total conductance of all cells in the circuit will be plotted.

interpolate-particle-arrays *nil* [*Variable*]

For the evalation of two−state gating particles, interpolate values derived from the voltage−dependent kinetic look−up arrays.

default-waveform-step *1.0* [*Variable*]

Default value for waveform time steps for sources and synapses, in milliseconds.

pwl-isource-di-dt *100.0* [*Variable*]

The transition slope for pwl isources (nA/msec).

isource-electrode-resistance *0.0* [*Variable*]

Mohms

vclamp-default-magnitude *−70.0* [*Variable*]

mV

pwl-vsource-dv-dt *1000.0* [*Variable*]

The transition slope for pwl vsources (mV/msec).

vsource-resistance *0.001* [*Variable*]

Series resistance, in Mohms, for the non−ideal vclamp. Must be >0.

user-stop-time *1.0* [*Variable*]

The time to end the simulation, in milliseconds.

user-max-step *5.0* [*Variable*]

The maximum time step allowed, in milliseconds. When 0, then *MAX−STEP* is bound by the simulation duration.

user-min-step *0.0* [*Variable*]

The minimum time step allowed, in milliseconds. When 0, then *MIN−STEP* is *MIN−STEP−MRTS*.

sim-time-n+1 *0* [*Variable*]

The time for the step currently being computed in units of *mrt*. t(n+1)

sim-time-n *0* [*Variable*]

The time for the step already computed in units of *mrt*. t(n)

sim-time-n-1 *0* [*Variable*]

The time for one step back in units of *mrt*. t(n−1)

sim-time-n-2 *0* [*Variable*]

The time for two steps back in units of *mrt*. t(n−2)

time-step *1* [*Variable*]

The current time step in units of *mrt*.

last-time-step *1* [*Variable*]

The last time step in units of *mrt*.

t[n+1] [*Macro*]

Time during simulation (msec), corresponding to the prediction time t_(n+1).

real-time *0.0* [*Variable*]

Time during simulation (msec), corresponding to the prediction time t_(n+1).

simulation-max-time *0.0* [*Variable*]

Maximum time reached during simulation (msec).

fractional-time [*Macro*]

The fractional part of (*t[n+1]*).

input-time [*Macro*]

The time reference for inputs (msec)

t[n] [*Macro*]

Time during simulation (msec), corresponding to the current time t_n.

last-real-time *0.0* [*Variable*]

Time during simulation corresponding to the current time (msec)

t-prime[n+1] [*Macro*]

Time during simulation (msec) of the staggered grid, corresponding to the prediction time t−prime_(n+1).

t-prime[n] [*Macro*]

Time during simulation (msec) of the staggered grid, corresponding to the current time t−prime_n.

t-prime[n-prime] [*Macro*]

Time during simulation (msec) of the staggered grid, halfway between (*t−prime[n]*) and (*t−prime[n+1]*).

t-prime[n-prime-1] [*Macro*]

Time during simulation (msec) of the staggered grid, halfway between (*t−prime[n−1]*) and (*t−prime[n]*).

t[n]-t-prime[n-prime] [*Macro*]

Difference between (*t[n]*) and (*t−prime[n−prime]*) in msec.

delta-t[n] [*Macro*]

The current time step [t_(n+1) − t_n], in msec.

delta-t[n]-squared [*Macro*]

msec2

delta-t[n-1] [*Macro*]

The last time step [t_n − t_(n−1)], in msec.

delta-t-prime[n] [*Macro*]

The current time step for the staggered grid, [t−prime_(n+1) − t−prime_n], in msec.

delta-t-prime[n]-squared [*Macro*]

msec2

delta-t-prime[n-1] [*Macro*]

The last time step for the staggered grid, [t−prime_(n) − t−prime_(n−1)], in msec.

2/delta-t[n] [*Macro*]

The constant to multiply capacitances by for the trapezoidal rule.

integer-time *0* [*Variable*]

The integer part of real−time

use-fixed-step *nil* [*Variable*]

Unless *USE−TIME−LIST* is T and valid, when T, use a fixed step integration, referencing *USER−STEP*. Otherwise use variable time step.

use-time-list *nil* [*Variable*]

When T, take integration time points from *LAST−SIM−REVERSE−TIME−STEP−LIST*. If there was no previous simulation, this is ignored.

user-step *0.01* [*Variable*]

Fixed time step [ms] for fixed step integration.

sim-reverse-plot-time-list *'()* [*Variable*]

The list of plotted time points for the simulation in the reverse order. Updated during the simulation.

sim-plot-time-list *'()* [*Variable*]

After a simulation is complete, the list of plotted time points in the correct order.

sim-reverse-time-step-list *'()* [*Variable*]

All the time steps in the simulation in reverse time order. Updated during the simulation.

last-sim-reverse-time-step-list *'()* [*Variable*]

All the time steps in the last simulation (if saved) in reverse time order.

sim-reverse-time-list *'()* [*Variable*]

All the time points in the simulation in reverse time order. Updated during the simulation.

last-sim-reverse-time-list *'()* [*Variable*]

All the time points in the last simulation (if saved) in reverse time order.

auto-refresh-last-sim-reverse-time-list *nil* [*Variable*]

Set *LAST−SIM−REVERSE−TIME−LIST* to the last simulation's time list.

use-node-voltage-initializations *nil* [*Variable*]

Use *NODE−VOLTAGE−INITIALIZATIONS* for setting node voltages.

node-voltage-initializations *'()* [*Variable*]

A list of dotted pairs, where for each pair the CAR is a node and the CDR is that node's initial value in mV.

use-conc-int-initializations *nil* [*Variable*]

Use *CONC−INT−INITIALIZATIONS* to set initial concentrations.

use-buffer-initializations *nil* [*Variable*]

Use *BUFFER−INITIALIZATIONS* to set initial buffer states.

use-pump-initializations *nil* [*Variable*]

Use *PUMP−INITIALIZATIONS* to set initial pump states.

conc-int-initializations *'()* [*Variable*]

A list of dotted pairs, where for each pair the CAR is a concentration integrator and the CDR is that integrator's initial value in mM.

buffer-initializations *'()* [*Variable*]

A list of dotted pairs, where for each pair the CAR is a buffer and the CDR is that buffer's initial value in mM.

pump-initializations *'()* [*Variable*]

A list of dotted pairs, where for each pair the CAR is a pump and the CDR is that pump's initial value in mM.

user-breakpoint-list *'()* [*Variable*]

A list of break points that is specified by the user, and then added to the points automatically collected into *BREAKPOINT−LIST*.

enable-dynamic-breakpoint-generation *t* [*Variable*]

Allow breakpoints to be dynamically generated during a simulation by event−based elements such as axons and voltage−dependent synapses.

total-num-iterations *0* [*Variable*]

The total number of iterations over all time.

total-num-time-points *0* [*Variable*]

The total number of time points taken for the simulation.

use-max-iterations *nil* [*Variable*]

Use *MAX−ITERATIONS* to constrain time steps.

max-iterations *1* [*Variable*]

Useful for debugging.

rectify-synapse-conductances *t* [*Variable*]

When T halfwave rectify, threshold at 0, all synaptic conductances during evaluation. Applies to VOLTAGE and EVENT synapse types.

absolute-voltage-error *0.05* [*Variable*]

A reasonable range: 0.01 − 0.1 (mV).

absolute-particle-error *0.001* [*Variable*]

In terms of particle state [0−1].

absolute-conc-int-error *0.001* [*Variable*]

mM

full-error-step-change *nil* [*Variable*]

Full documentation of error step changes.

consider-particle-error *t* [*Variable*]

Consider LTE for particle states.

calculate-particle-error *t* [*Variable*]

Actually calculate LTE for particle states.

calculate-markov-particle-error *t* [*Variable*]

If *CALCULATE−PARTICLE−ERROR* is T, then also calculate LTE for Markov particle states.

consider-conc-particle-error *t* [*Variable*]

Consider LTE for particle states.

consider-conc-int-error *nil* [*Variable*]

Consider LTE for concentration ints as well, using the factor above for concentrations.

calculate-conc-int-error *nil* [*Variable*]

Actually calculate it.

pick-time-step-fudge *0.8* [*Variable*]

Coefficient for choosing new time step based on lte estimate. Less than one to speed up time step reduction when lte−ratio = 1

soma-shunt *1.0e+30* [*Variable*]

Default value [ohms] for non−specific soma shunt.

default-conc-int-type-diffusion-distance *10.0* [*Variable*]

microns

enable-axons *t* [*Variable*]

When nil, all axons are blocked.

include-events-in-element-documentation-code *nil* [*Variable*]

Enables storing of event times in element documentation code, for example for event synapses.

enable-event-generators *t* [*Variable*]

Event generators reduce evaluations for axons and synapses [VOLTAGE, LIGHT and LIGHT−EVENT] whose control parameters are identical for a given simulation.

setup-event-generators-and-followers *t* [*Variable*]

Enables the automatic assignment of event element sets at the beginning of every simulation, as long as *USER−SPECIFIED−EVENT−ELEMENT−SETS* is NIL. This variable may be set to NIL after a simulation for more efficiency in subsequent simulations, or may always be NIL as long as the function SETUP−ALL−EVENT−GENERATORS−AND−FOLLOWERS is explicitly called when the circuit is setup or changed.

user-specified-event-element-sets *nil* [*Variable*]

If this flag is T, then the user has the responsibility to setup event generators and followers, e.g. with calls to USER−SETUP−EVENT−GENERATORS−AND−FOLLOWERS or SETUP−ALL−EVENT−GENERAT before a simulation.

maximum-synapse-printed-events *5* [*Variable*]

Maximum number of event times or delays that will be explicitly printed when printing out a synapse's information.

enable-synapses *t* [*Variable*]

When nil, all synapses are blocked.

adjust-breakpoints-for-event-synapses *t* [*Variable*]

Before each simulation, add event synapse event times to the *BREAKPOINT−LIST* to ensure catching the events.

convert-light-response-to-events-for-each-synapse *nil* [*Variable*]

When event generators are used for light related synapses, this flag causes the light−response −> event conversion to be done individually for each synapse.

enable-light-event-update *t* [*Variable*]

When T, renew :EVENT−TIMES slot for LIGHT−EVENT synapses.

constant-light-input-from-negative-infinity *t* [*Variable*]

Whatever the light input is calculated to be at time 0, assume that this is the initial conditions (otherwise, intial light conditions are taken as zero state).

enable-light *t* [*Variable*]

Let there be light.

light-input-offset-distance *0.0* [*Variable*]

um

light-input-offset-angle *0.0* [*Variable*]

radians

light-input-delay *0.0* [*Variable*]

Light input delay between light event at offset location and ynaptic response, in milliseconds.

light-speed *0.0* [*Variable*]

Microns per millisecond

bar-width *0.0* [*Variable*]

Microns

bar-length *0.0* [*Variable*]

Microns

light-theta *1.5707964* [*Variable*]

Radians

light-direction *t* [*Variable*]

T / nil => movement is in the direction of / opposite to *light−theta*.

light-start-position-x *0.0* [*Variable*]

Point of center of stimulus at *motion−start−time* in microns

light-start-position-y *0.0* [*Variable*]

Point of center of stimulus at *motion−start−time* in microns

grating-temporal-period *1000000.0* [*Variable*]

Milliseconds

grating-spatial-period *1000000.0* [*Variable*]

Microns

use-aperture *nil* [*Variable*]

Consider aperture.

aperture-radius *300.0* [*Variable*]

Microns

aperture-center-x *0.0* [*Variable*]

Microns

aperture-center-y *0.0* [*Variable*]

Microns

bar-a-width *0.0* [*Variable*]

microns

bar-a-length *0.0* [*Variable*]

microns

bar-a-start-time *0.0* [*Variable*]

milliseconds

bar-a-stop-time *0.0* [*Variable*]

milliseconds

bar-a-position-x *0.0* [*Variable*]

microns

bar-a-position-y *0.0* [*Variable*]

microns

bar-b-width *0.0* [*Variable*]

microns

bar-b-length *0.0* [*Variable*]

microns

bar-b-start-time *0.0* [*Variable*]

milliseconds

bar-b-stop-time *0.0* [*Variable*]

milliseconds

bar-b-position-x *0.0* [*Variable*]

microns

bar-b-position-y *0.0* [*Variable*]

microns

light-stimulus-types *'(moving−spot annulus on−spot off−spot on−moving−bar off−moving−bar apparent−motion on−bar off−bar moving−bar)* [*Variable*]

:MOVING−BAR is equivalent to :ON−MOVING−BAR

light-stimulus *nil* [*Variable*]

Can take a value out of *LIGHT−STIMULUS−TYPES*.

light-stimulus-plane *:xy* [*Variable*]

:XY for retina, :XZ for radial mount cortical cells.

count-active-and-triggered-synapses *t* [*Variable*]

When non−NIL, the function COUNT−ACTIVE−SYNAPSES, which normally prints out info at the end of each simulation, also prints the number of synapses actually fired.

always-intialize-random-gen *nil* [*Variable*]

Forces a call to GET−REFERENCE−RANDOM−STATE at various places.

3 SYS Source File: biophysics-declare.lisp

faraday *96480.0* [*Constant*]

Faraday's constant – Coulombs/mole

boltzmanns-constant *1.3806221e−23* [*Constant*]

Joules/degree Kelvin

gasconstant *8.31434* [*Constant*]

Gas Constant

foverr *(/ faraday gasconstant)* [*Constant*]

Faraday / GasConstant.

plancks-constant *6.626196e−34* [*Constant*]

Joules second

d_na *1.33e−5* [*Variable*]

Diffusion constant of Na+ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_li *1.0299999e−5* [*Variable*]

Diffusion constant of Li+ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_k *1.96e−5* [*Variable*]

Diffusion constant of K+ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_cs *2.0599999e−5* [*Variable*]

Diffusion constant of Cs+ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_cl *2.03e−5* [*Variable*]

Diffusion constant of Cl− in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_br *2.0799999e−5* [*Variable*]

Diffusion constant of Br+ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_tea *8.7e−6* [*Variable*]

Diffusion constant of TEA in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_mg *7.1e−6* [*Variable*]

Diffusion constant of Mg++ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

d_ca *7.9e−6* [*Variable*]

Diffusion constant of Ca++ in aqueous solutions at 25deg C (Hille, B., Ionic Channels of Excitable Membranes, 1984).

v-leak *−70.0* [*Variable*]

Default membrane leak reversal potential, mV.

v-leak-dendrite *−70.0* [*Variable*]

Default dendritic membrane leak reversal potential, mV.

na-conc-extra *145.0* [*Variable*]

mM

na-conc-intra *12.0* [*Variable*]

mM

e-na *64.0* [*Variable*]

mV

fix-e-na *t* [*Variable*]

Fix the global *E−NA* − otherwise, *E−NA* is updated as with the Nernst equation using *TEMPERATURE* and the global *NA−CONC−INTRA* and *NA−CONC−EXTRA*.

k-conc-extra *4.0* [*Variable*]

mM

k-conc-intra *155.0* [*Variable*]

mM

e-k *−95.0* [*Variable*]

mV

fix-e-k *t* [*Variable*]

Fix the global *E−K* − otherwise, *E−K* is updated as with the Nernst equation using *TEMPERATURE* and the global *K−CONC−INTRA* and *K−CONC−EXTRA*.

e-cl *−90.0* [*Variable*]

mV

cl-conc-extra *120.0* [*Variable*]

mM

cl-conc-intra *4.0* [*Variable*]

mM

fix-e-cl *t* [*Variable*]

Fix the global *E−Cl* − otherwise, *E−Cl* is updated as with the Nernst equation using *TEMPERATURE* and the global *Cl−CONC−INTRA* and *Cl−CONC−EXTRA*.

e-ca *110.0* [*Variable*]

mV

ca-conc-extra *1.8* [*Variable*]

mM

ca-conc-intra *5.0e−5* [*Variable*]

mM

fix-e-ca *t* [*Variable*]

Fix the global *E−Ca* − otherwise, *E−Ca* is updated as with the Nernst equation using *TEMPERATURE* and the global *Ca−CONC−INTRA* and *Ca−CONC−EXTRA*.

mg-conc-extra *1.5* [*Variable*]

mM

mg-conc-intra *1.5* [*Variable*]

mM

e-mg *0.0* [*Variable*]

mV

rm *40000.0* [*Variable*]

Default value of dendrite membrane resistivity (ohms cm2)

ri *200.0* [*Variable*]

Default value of segment axial resistivity (ohms cm)

rm-soma *40000.0* [*Variable*]

Default value of soma membrane resistivity (ohms cm2)

cm *0.7* [*Variable*]

Default value of membrane capacitance (uF/cm2)

cm-dendrite *0.7* [*Variable*]

Default value of membrane capacitance (uF/cm2)

r-extracellular *200.0* [*Variable*]

Default value of extracellular resistivity (ohms cm)

4 SYS Source File: structures.lisp

model *name hash–table parameter–type–library output–data–structure–variables output–data–enabled data–types–and–access–info child–structure–type parent–structure–type top–pointer–symbol save–output–data–routine edit–routine eval–routine print–routine short–print–routine document–routine create–routine* [*Structure*]

dummy-floats *nuthin* [*Structure*]

dummy-fixnums *nuthin* [*Structure*]

dummy-double-floats *nuthin* [*Structure*]

node-double-floats *voltage−n+1 voltage−n voltage−n−1 jacobian const−jacobian alpha−charge current const−current dvdt−n dvdt−n−1* [*Structure*]

node-fixnums *prt−v−index* [*Structure*]

node *name cell is−physical−cell−node relative−location absolute−location elements has−v−dep−element index prt−v−index−rem has−ideal−voltage−source parameters fixnums double−floats* [*Structure*]

soma-guts *capacitance g−leak*v−leak g−leak* [*Structure*]

soma-capacitance *soma* [*Macro*]

Membrane capacitance in nF.

soma-g-leak *soma* [*Macro*]

Membrane leak conductance in uS.

soma *name node inherit−parameters−from−type diameter include−shunt g−shunt current parameters guts* [*Structure*]

segment-guts *capacitance g−leak*v−leak g−axial g−leak* [*Structure*]

segment-capacitance *segment* [*Macro*]

Membrane capacitance in nF.

segment-g-axial *segment* [*Macro*]

Axial conductance in uS.

segment-g-leak *segment* [*Macro*]

Membrane leak conductance in uS.

segment *name node−1 node−2 dummy−proximal−node−location length diameter theta phi distance−to−soma inherit−parameters−from−type branch−node−index mat−12−point mat−21−point parameters guts* [*Structure*]

isource-type *name class activation−function rf−function first−element last−element parameters* [*Structure*]

isource-floats *current* [*Structure*]

isource *name type cell−element control−element next−element node−1 node−2 blocked use−pulse−list resistance pwl−list waveform−array waveform−length waveform−time−interval−inverse waveform−time−interval−mrt period delay node−1−pointp node−2−pointp parameters floats* [*Structure*]

vsource-type *name class activation−function rf−function first−element last−element parameters* [*Structure*]

vsource *name type cell−element control−element next−element node blocked use−pulse−list pwl−list waveform−array waveform−time−interval−inverse waveform−time−interval−mrt resistance function−list period delay current last−voltage voltage adjacent−nodes−and−g−axials parameters* [*Structure*]

system-of-differential-equations *number−of−states coefficients−array* [*Structure*]

membrane-element-type-iv *relation source reference density inherit−parameters−from−type use−defined−e−rev e−rev variable−e−rev ion−permeabilities conc−int−type−params blocked q10 reference−temp* [*Structure*]

membrane-element-iv-values *conductance gbar e−rev current gbar/perm−reference−value* [*Structure*]

channel-type *name iv−parameters first−element last−element particle−types−and−powers conc−particle−types−and−powers parameters* [*Structure*]

channel *name type cell−element next−element blocked conc−ints−params pre−synaptic−element inherit−parameters−from−type particles conc−particles parameters iv−values* [*Structure*]

particle-type *name class q10 reference−temp q10−rate−factor number−of−states state−transition−array open−state−array tau−array inf−array valence gamma base−rate v−half tau−0 ignore−tau−voltage−dependence alpha−function beta−function tau−coefficient tau−function ss−function first−element last−element evaluation−function parameters* [*Structure*]

particle-double-floats *state−n+1 state−n dsdt−n dsdt−n−1* [*Structure*]

particle *name type conc−particle channel vnode−point next−element state−arrays parameters double−floats* [*Structure*]

Model for a gating particle

conc-particle-type *name class alpha beta tau−0 conc−power k1 k4 alpha−function beta−function tau−function ss−function conc−dependence shell conc−int−type base−concentration concentration−coefficient q10 reference−temp q10−rate−factor first−element last−element evaluation−function parameters* [*Structure*]

conc-particle *name type channel next−element cnode−point conc−int parameters double−floats* [*Structure*]

synapse-type *name iv−parameters control waveform−time−interval−inverse waveform−time−interval−mrt input−threshold refractory−period supra−threshold−duration−min evaluation−function first−element last−element parameters* [*Structure*]

synapse *name type cell−element next−element blocked conc−ints−params pre−synaptic−element channel inherit−parameters−from−type event−times event−generator delay fixnum−delay transformed−events sub−threshold−time wave−ref parameters iv−values* [*Structure*]

conc-int-type *name class species valence blocked enabled−for−this−simulation intra−p system−of−differential−equations shell−2−p shell−3−p core−p volumes membrane−areas diffusion−areas diffusion−distances juxtamembrane−shell−thickness alpha−s inner−shell−thickness diffusion−coefficient interdigitation−coefficient pump−type−params instantaneous−buffer−enabled shell−1−instantaneous−buffer−ratio+1 shell−2−instantaneous−buffer−ratio+1 shell−3−instantaneous−buffer−ratio+1 global−instantaneous−buffer−ratio+1 core−conc core−conc−double transmembrane−conc transmembrane−conc−double q10 reference−temp q10−rate−factor conc−ints parameters* [*Structure*]

conc-int-double-floats *shell−1−conc−n+1 shell−1−conc−n shell−1−dcdt−n shell−1−dcdt−n−1 shell−2−conc−n+1 shell−2−conc−n shell−2−dcdt−n shell−2−dcdt−n−1 shell−3−conc−n+1 shell−3−conc−n total−conc−n total−conc−n+1 e−rev−shell−1 e−rev−shell−2* [*Structure*]

conc-int *name type cell−element blocked enabled−for−this−simulation evaluate−total−concentration shell−1−volume shell−2−volume shell−3−volume core−volume total−volume transmembrane−integrator shell−1−pores shell−2−pores shell−1−pumps shell−2−pumps shell−3−pumps shell−1−buffers shell−2−buffers shell−3−buffers state−arrays beta−2−1 beta−1−2 beta−1−3 beta−3−1 beta−2−3 beta−3−2 beta−core−3 beta−core−1 beta−core−2 beta−current−1 beta−current−2 parameters double−floats* [*Structure*]

axon-type *name propagation−velocity input−threshold refractory−period supra−threshold−duration−min output−waveform blocked q10 reference−temp axons waveform−time−interval−inverse waveform−time−interval−mrt parameters* [*Structure*]

axon-floats *voltage sub−threshold−time* [*Structure*]

axon *name type node proximal−node target−synapse cell−element blocked length delay propagation−delay spike−times inherit−parameters−from−type event−generator parameters floats* [*Structure*]

cell-type *name notes v−leak−soma v−leak−dendrite rm−soma soma−shunt rm−dendrite ri cm−soma cm−dendrite na−conc−intra na−conc−extra na−conc−extra−dependence e−na−dependence e−na k−conc−intra k−conc−extra k−conc−extra−dependence e−k−dependence e−k ca−conc−intra ca−conc−extra ca−conc−extra−dependence e−ca−dependence e−ca cl−conc−intra cl−conc−extra cl−conc−extra−dependence e−cl−dependence e−cl inherit−parameters−from−type global−membrane−conductance−factor cells parameters* [*Structure*]

cell *name type origin soma segments max−x min−x max−y min−y max−z min−z max−g−in z−tree−discrete−in−cell z−tree−cable−in−cell z−discrete−in−cell z−cable−in−cell parameters* [*Structure*]

buffer-type *name class species blocked total−conc total−conc−double system−of−differential−equations k−forward k−backward q10 reference−temp buffers parameters* [*Structure*]

buffer-double-floats *conc−n+1 conc−n* [*Structure*]

buffer *name type blocked enabled−for−this−simulation cell−element conc−int conc−int−compartment parameters double−floats* [*Structure*]

pump-type *name class species blocked q10 reference−temp equilibrium−conc tau tau−array v−max kd system−of−differential−equations total−density k−1 k−2 k−3 k−4 parameters pumps* [*Structure*]

pump-double-floats *density−n+1 density−n current* [*Structure*]

pump *name type blocked enabled−for−this−simulation area basal−rate mm−coefficent conc−int conc−int−compartment parameters double−floats* [*Structure*]

extracellular-electrode *name absolute−location parameters* [*Structure*]

5 SYS Source File: structure-macros.lisp

channel-reference-temp *ch* [*Macro*]

Degrees C

channel-conductance *ch* [*Macro*]

uS or cm3/s

channel-gbar *ch* [*Macro*]

uS or cm3/s

channel-e-rev *ch* [*Macro*]

mV

channel-current *ch* [*Macro*]

nA

channel-gbar/perm-reference-value *ch* [*Macro*]

uS or cm3/s

channel-iv-reference-value *syn* [*Macro*]

uS or cm3/s

synapse-reference-temp *syn* [*Macro*]

Degrees C

synapse-conductance *syn* [*Macro*]

uS or cm3/s

synapse-gbar *syn* [*Macro*]

uS or cm3/s

synapse-e-rev *syn* [*Macro*]

mV

synapse-current *syn* [*Macro*]

nA

synapse-gbar/perm-reference-value *syn* [*Macro*]

uS or cm3/s

synapse-iv-reference-value *syn* [*Macro*]

uS or cm3/s

6 SYS Source File: models.lisp

cell-type-def body [*Macro*]

Parameter wrapper for cell type definitions. Units of typical parameters include the following:
Required:

```
RM              ohms-cm2
RI              ohms-cm
CM              uF/cm2
V-LEAK          mV
```

Distinct values for somatic versus dendritic segment membrane may be specified using:

```
RM-SOMA or RM-DENRITE           ohms-cm2
CM-SOMA or CM-DENDRITE          uF/cm2
V-LEAK-SOMA or V-LEAK-DENDRITE  mV
```

Optional:

```
E-NA                    mV
E-K                     mV
E-CA                    mV
E-CL                    mV
```

channel-type-def body [*Macro*]

Parameter wrapper for channel type definitions. Units and formats of typical parameters include the following:

One of the following four parameters are required to specify gbar or pbar in terms of absolute value or density. GBAR or GBAR-DENSITY parameters will apply the :OHMIC iv-relation model, and permeability or permeability-density parameters will apply the :CONSTANT-FIELD iv-relation model:

```
GBAR                    uS for conductance
GBAR-DENSITY            pS/um2 for conductance
PERMEABILITY            cm3/s for permeability
PERMEABILITY-DENSITY    1.0e-6cm3/s/um2 for permeability
```

Required:

```
E-REV               mV
V-PARTICLES         List of lists, each specifying particle type
                    and order, e.g. ((N-HH 2)(Y-HH 1))
```

Required for :CONSTANT-FIELD model, optional for :OHMIC model:

```
ION-PERMEABILITIES  List of lists, each specifying ion type and
                    permeability (total must add to unity), e.g. ((K 0.8)(NA 0.2))
```

Optional parameters:

```
QTEN              For conductance or permeability value, default 1
REFERENCE-TEMP    Degrees C, for QTEN
GBAR-MODULATION   Optional. Applies to permeability model also.

CONDUCTANCE-FUNCTION Function name with a single channel argument returning single float numeri
                      that is multiplied by standard channel conductance value.

STATIC-VOLTAGE-DEPENDENCE-FUNCTION  Function name or specification which returns a numeric sequ
                                sequence, where the sequence is used as a look up table of voltage
                                corresponding to the values given by *PARTICLE-LOOK-UP-TABLE-MIN-V
                                *PARTICLE-LOOK-UP-TABLE-MAX-VOLTAGE* and *PARTICLE-LOOK-UP-TABLE-P
                                whose value is then multiplied by standard channel conductance val
                                must be single float.
```

synapse-type-def body [*Macro*]

Parameter wrapper for synapse type definitions. Units and formats of typical parameters the include following:

```
CONTROL           :EVENT, :LIGHT, :CHANNEL, :VOLTAGE, :LIGHT-EVENT, :TONIC, and other
```

One of the following four parameters are required to specify gbar or pbar in terms of absolute value or density. GBAR or GBAR-DENSITY parameters will apply the :OHMIC iv-relation model, and permeability or permeability-density parameters will apply the :CONSTANT-FIELD iv-relation model:

```
GBAR                    uS for conductance
GBAR-DENSITY            pS/um2 for conductance
PERMEABILITY            cm3/s for permeability
PERMEABILITY-DENSITY    1.0e-6cm3/s/um2 for permeability

WAVEFORM-SPEC          Either numeric sequence, function symbol (if no required args), or full
WAVEFORM-TIME-INTERVAL In milliseconds, otherwise reference *DEFAULT-WAVEFORM-STEP*

E-REV              mV
```

Optional:

```
SPECIFIED-WAVEFORM-BREAKPOINTS  List of times in milliseconds
REFERENCE-TEMP      degC
ION-PERMEABILITIES  List of lists, each specifying ion type and
                    permeability (total must add to unity), e.g. ((K 0.8)(NA 0.2))

CONDUCTANCE-FUNCTION Function name with a single synapse argument returning single float numeri
                     that is multiplied by standard synapse conductance value.

STATIC-VOLTAGE-DEPENDENCE    Function specification which returns a numeric sequence, or an exp
                             sequence, where the sequence is used as a look up table of voltage
                             corresponding to the values given by *PARTICLE-LOOK-UP-TABLE-MIN-V
                             *PARTICLE-LOOK-UP-TABLE-MAX-VOLTAGE* and *PARTICLE-LOOK-UP-TABLE-P
                             must be single float.ISION*,
                             whose value is then multiplied by standard synapse conductance val
```

For user defined synapse type control:

```
 EVALUATION-FUNCTION   This function should take one required argument, the synapse type, and a
                       argument, a flag indicating first iteration. The function should loop ov
                       synapses of the type and apply the FINISH-SYN-EVAL macro to each.
```

particle-type-def body [*Macro*]

Parameter wrapper for particle type definitions. Units and formats of typical parameters the include following:

```
CLASS                         :HH, :HH-EXT, :MARKOV, and other
CONCENTRATION-PARTICLE-TYPE Type symbol
QTEN                          For kinetics, default 1.0
REFERENCE-TEMP                Degrees C, for QTEN
```

For :HH-EXT class of particle types:

```
VALENCE       particle valence, dimensionless
GAMMA         dimensionless - between 0 and 1
BASE-RATE     1/ms
V-HALF        mV
TAU-0         ms
ALPHA_0       V-indep forward rate constant [1/ms]
BETA_0        V-indep backward rate constant [1/ms]
```

For :MARKOV class of particle types:

```
STATES
OPEN-STATES
STATE-TRANSITIONS
```

For :HH or :HH-EXT class of particle types:

```
ALPHA      Lambda form or function name with single voltage [mV] argument,
           that return value in 1/ms, or number [1/ms].
BETA       Same as for ALPHA
```

or

```
SS         Lambda form or function name with single voltage [mV] argument,
           that return value between 0 and 1 inclusive, or number.
TAU        Lambda form or function name with single voltage [mV] argument,
           that return value in milliseconds, or number [ms].
```

Explicit specifications of either SS and TAU will override :HH-EXT parameter derived, ALPHA or BETA for the SS and Tau curves, respectively.

For either :HH-EXT or :HH class of particle types:

```
LINEAR-MARKOV     (Linear-Markov-N-Value Linear-Markov-M-Value)
FIXED-BOLTZMANN-REFERENCE-TEMPERATURE   degrees C
IGNORE-TAU-VOLTAGE-DEPENDENCE           T or NIL [default]
TAU-COEFFICIENT                         Number
```

For user defined particle type classes:

```
EVALUATION-FUNCTION   This function should take one required argument, the particle type, and
                      argument, a flag indicating initial state. The function should loop over
                      particles of the type and set the :STATE-N+1-DOUBLE slot of each.
```

conc-int-type-def body [*Macro*]

Parameter wrapper for concentration integrator type definitions. Units and formats of typical parameters include the following:

```
CLASS                    :MULTI-SHELL [default], :GENERAL, :FIRST-ORDER, :GENERIC, and other
SPECIES       'CA, 'K, 'NA etc.
VALENCE                  number
INTRA-P                  Concentration integration for the intra (T [default])
                         or extracellular space (NIL)
```

For :MULTI-SHELL conc-ints:

```
SHELL-2-P                    [default nil]
SHELL-3-P                    [default nil]
CORE-P                       [default nil]
INTERDIGITATION-COEFFICIENT  1/microns [between shells 1 and 2]
```

Lists of functions or explicit values for determining compartment parameters of :MULTI-SHELL conc-ints:

```
DIFFUSION-COEFFICIENT     cm2/sec
VOLUMES                   um3
MEMBRANE-AREAS            um2
DIFFUSION-AREAS           um2
DIFFUSION-DISTANCES       um
```

If either VOLUMES, DIFFUSION-AREAS or MEMBRANE-AREAS are NIL, then the following parameters are used:

```
JUXTAMEMBRANE-SHELL-THICKNESS   microns
INNER-SHELL-THICKNESS           Thickness of shell 3, microns
ALPHA-S                         Proportion of juxta-membrane shell assigned to shell 1.

INSTANTANEOUS-BUFFER-ENABLED    Enables the instantaneous shell buffers.
SHELLS-W-INSTANTANEOUS-BUFFER
INSTANTANEOUS-BUFFER-RATIO

QTEN                        For diffusion kinetics, default 1.0
REFERENCE-TEMP              Degrees C, for QTEN
```

Concentration reference values, in mM:

```
TRANSMEMBRANE-CONCENTRATION
RESTING-FREE-CONC
```

For :FIRST-ORDER types:

```
TAU                         milliseconds
```

For :GENERIC conc-ints, one of the following functions should be included, each of which must take a single conc-int argument:

```
CONC-FUNCTION       This must set the :SHELL-1-CONC-N+1 slot of the conc-int
C-N+1-FUNCTION      Return value in mM that will be used to set the :SHELL-1-CONC-N+1 slot of t
DCDT-FUNCTION Return value in mM/ms (forward euler integration)
```

pump-type-def body [*Macro*]

Parameter wrapper for pump type definitions. Units and formats of typical parameters the include following:

```
CLASS              :MM, :MM-ZADOR, :FIRST-ORDER, :FIRST-ORDER-TAU-V

V-MAX              For :MM
K-D                For :MM, :MM-ZADOR
K-MAX, DENSITY     For :MM-ZADOR
EQUILIBRIUM-CONC   For :FIRST-ORDER, :FIRST-ORDER-TAU-V

TAU          For :FIRST-ORDER-TAU-V, Lambda form or function name
             with single voltage [mV] argument, with return units of ms
             For :FIRST-ORDER, number [ms]

QTEN               For kinetics, default 1.0
REFERENCE-TEMP     Degrees C, for QTEN
SPECIES            Ionic species, e.g. NA, K, CA
```

segments &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all segments associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all segments in circuit.

somas &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all somas associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all somas in circuit.

cell-types &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all cell-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all cell-types in circuit.

cells &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all cells associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all cells in circuit.

channel-types &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all channel-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all channel-types in circuit.

channels &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all channels associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all channels in circuit.

synapse-types &optional *(element nil element–supplied-p)* [*Function*]

Returns a list of all synapse-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all synapse-types in circuit.

synapses &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all synapses associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all synapses in circuit.

particle-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all particle-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all particle-types in circuit.

particles &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all particles associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all particles in circuit.

conc-particle-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all conc-particle-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all conc-particle-types in circuit.

conc-particles &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all conc-particles associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all conc-particles in circuit.

conc-int-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all conc-int-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all conc-int-types in circuit.

conc-ints &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all conc-ints associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all conc-ints in circuit.

isource-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all isource-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all isource-types in circuit.

isources &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all isources associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all isources in circuit.

vsource-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all vsource-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all vsource-types in circuit.

vsources &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all vsources associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all vsources in circuit.

axon-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all axon-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all axon-types in circuit.

axons &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all axons associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all axons in circuit.

buffer-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all buffer-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all buffer-types in circuit.

buffers &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all buffers associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all buffers in circuit.

pump-types &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all pump-types associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all pump-types in circuit.

pumps &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all pumps associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all pumps in circuit.

extracellular-electrodes &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all extracellular-electrodes associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all extracellular-electrodes in circuit.

nodes &optional *(element nil element−supplied-p)* [*Function*]

Returns a list of all nodes associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all nodes in circuit.

enable-element-save-data *element* &optional *data−type model−type* [*Function*]

Enable saving of DATA−TYPE (as in ELEMENT−DATA) of elements in ELEMENT of MODEL−TYPE. If ELEMENT is an element type, then all elements of that type are affected. For elements that can generate more than one type of simulation data, setting DATA−TYPE to :ALL will enable saving of all data types (except for events). DATA−TYPE may also be a list of data types, with the deffault given by the function DEFAULT−DATA−TYPE. For saving element data that will also be used for plotting, use ENABLE−ELEMENT−PLOT.

disable-element-save-data *element* &optional *data−type* *(abort−disable−if−plotted t)* *model−type* [*Function*]

Disables saving of DATA−TYPE, in opposition to ENABLE−ELEMENT−SAVE−DATA. If ABORT−DISABLE−IF−PLOTTED is T [default], and if the data type for this element is currently earmarked for plotting, the disabling of saving data is aborted.

7 SYS Source File: create-models.lisp

cell-types &optional *element* [*Function*]

Returns a list of all cell types associated with the cell elements referenced by ELEMENT. If there are no such cell elements, or ELEMENT is NIL, then a list of all cell−types in circuit is returned.

8 SYS Source File: declare-2.lisp

circuit-object-type nil *common−lisp::backq−cons 'or *model−names** [*Type*]

The union of all circuit object types.

9 SYS Source File: element-functions-0.lisp

element *elt−reference* &optional *model−type fast* [*Function*]

Return either the single structure object (as an atom) or objects (as a list) that are associated, if any, with ELT−REFERENCE. If none, then ELEMENT returns NIL. ELT−REFERENCE is either a structure object, a structure name, or a (mixed) list of same. In the case of a name (which may be either a string, a symbol, or an integer), since two or more structure objects of different model−types may have the same name the search priority of structure model−types is given by *MODEL−HASH−TABLES*.

10 SYS Source File: element-functions-1.lisp

cell-element-elements *element* &optional *(types :all)* [*Function*]

Return a list of all elements whose model type (e.g. 'SYNAPSE or 'CONC−INT) or element type (e.g. specific synapse or conc−int type) is contained in TYPES, and that are associated with all the circuit nodes implied by ELEMENT. If TYPES is :ALL, which is the default, then return all elements including the cell element directly associated with the circuit nodes, excluding any proximally connected cell elements.

cell-elements &optional *cell* [*Function*]

Returns a list of all somas and segments associated with the cell or cells referenced by CELL (can be a single cell or a list) [default all cells in circuit].

pre-synaptic-element *element* [*Function*]

Return the pre−synaptic cell element of ELEMENT.

reorder-elements-of-type *type* [*Function*]

Reorders all the elements of TYPE for the iterator constructs (applies to synapses, channels, particles and conc−particles at the moment). See also *ENABLE−REORDER−ELEMENTS*.

element-type-p *element* &optional *type* [*Function*]

Predicate whether ELEMENT of TYPE is a circuit element type, for example BUFFER−TYPE, PUMP−TYPE, CHANNEL−TYPE, CONC−INT−TYPE, PARTICLE−TYPE, CONC−PARTICLE−TYPE, SYNAPSE−TYPE, AXON−TYPE, or CELL−TYPE.

library-catalog *element* &optional *verbose* [*Function*]

Returns a list of all type symbols in the type library referenced by ELEMENT. Thus, if ELEMENT is a channel, then the symbols of all library channel types are returned. ELEMENT can also be a symbol for the model class, e.g. 'CHANNEL or 'CHANNEL−TYPE. If VERBOSE then print out the contents of the TYPE−DEF forms as well.

element-node *element* &optional *model−type* [*Function*]

Return the soma or segment circuit node(s) associated with ELEMENT of MODEL−TYPE.

type-instances-in-cell *thing* &optional *model−type* [*Function*]

Given a name or instance of THING of MODEL−TYPE, returns a list of instances of the same model−type.

elements-of-type *element−reference* &optional *cell−elements−reference* [*Function*]

If ELEMENT−REFERENCE is a model type symbol [e.g. 'channel or 'channel−type], then returns all instances of the model [e.g. all channels or all channel types]. Otherwise, if ELEMENT−REFERENCE refers to a specific instance of an element parent type (synapse type, channel type, etc.), returns all the child instances (synapses of that synapse type, or channels of that channel type). If CELL−ELEMENTS−REFERENCE is included, returned elements are restricted to cell elements associated with CELL−ELEMENTS−REFERENCE; otherwise all elements are returned.

create-element *thing* &rest *others* [*Function*]

Generic create function for elements. Takes any number of arguments, and considers all atoms in these arguments as members of a flat list ARGS. Given any atom in ARGS which references an element type, CREATE−ELEMENT adds an element of that type to all the cell elements referenced in ARGS. If any member of ARGS refers to a cell, then that cell's soma is processed as a cell element. Any element types referenced in ARGS are created if they do not already exist. Returns either a single object (as an atom) or objects (as a list) of all created circuit elements, if any; otherwise then all referenced element types are returned, whether or not created during the current invocation of CREATE−ELEMENT. For example:

```
* (create-element 'NA-HH)
<Channel Type NA-HH>

* (create-element 'NA-HH *soma*)
<Channel Hippo-soma-NA-HH: type NA-HH>

* (create-element 'NA-HH *soma* 'DR-HH)
(<Channel Hippo-soma-NA-HH: type NA-HH>
 <Channel Hippo-soma-DR-HH: type DR-HH>)
```

If the keyword :NO−DUPLICATES is included in the arguments, then no duplicate elements (for example the same synapse type on the same cell element) will be created. Otherwise duplicates may be generated with or without use interaction depending on the values of *USE−SIMPLE−NAMES*, *ALLOW−DUPLICATE−ELEMENTS* and *PROMPT−FOR−ALTERNATE−ELEMENT−NAMES*. The arguments 'SEGMENT, 'SOMA and 'CELL−ELEMENT will reference all the segments, somas, and both segments and somas of the circuit, respectively. See also the global variable *ENABLE−REORDER−ELEMENTS*.

random-segment &optional *(element *cell*)* [*Function*]

Returns a randomly selected (using RANDOM−NTH) segment from the cell or cells associated with ELEMENT (default *CELL*).

element-of-ion-type-p *element ion−type* [*Function*]

Given ELEMENT, returns T if it is associated with ION−TYPE (e.g. 'NA, 'K, 'CA, 'CL, etc.).

element-capacitance *element* &optional *(value nil value−supplied−p)* [*Function*]

Sets the capacitance of the soma or segment associated with ELEMENT to VALUE, in nF, if non−nil, Otherwise returns the current value in nF.

element-current *element* &optional *conc−in conc−out valence* [*Function*]

ELEMENT must refer to a channel, synapse, isource, vsource or pump. Returns value in nA.

element-reversal-potential *element* [*Function*]

Returns the reversal potential in mV for the channel or synapse associated with ELEMENT.

element-conductance *element* [*Function*]

Returns the conductance in uS for the channel or synapse associated with ELEMENT.

element-voltage *element* [*Function*]

Returns the voltage in mV for the node associated with ELEMENT.

element-dvdt *element* [*Function*]

Either NODE−DVDT−N or GET−NODE−DVDT (V_n+1 − V_n / delta−t[n]). Value in mV/ms.

element-capacitance-current *element* [*Function*]

Return the membrane capacitance current in nA for the node associated with ELEMENT.

element-g-leak *element* &optional *(value nil value−supplied−p)* [*Function*]

Sets the g−leak of the soma or segment associated with ELEMENT to VALUE, in uS, if non−nil, Otherwise returns the current value in uS.

element-leak-current *element* [*Function*]

Return the membrane leak current in nA for the node associated with ELEMENT.

element-parameter *element parameter* &optional *(value nil value−supplied−p) update* [*Function*]

Returns the value or values associated with PARAMETER for elements in ELEMENT, where PARAMETER is stored as part of an element's :PARAMETERS slot (an a−list). For accessing an element's explicitly defined slots, use the function ELEMENT−SLOT. If VALUE is supplied, the parameter is set to this new value. For some types of elements and parameters, the UPDATE flag will cause the parameter to be fully processed.

element-parameter-fast *element parameter* &optional *(value nil value−supplied−p) update* [*Function*]

As ELEMENT−PARAMETER, but ELEMENT must be an atom and an element pointer.

update-type-from-definition *element* [*Function*]

Updates the ELEMENT type from the most recently loaded library definition. Note that change in an ELEMENT type does not necessarily propagate to elements of that type.

convert-iv-relations-to-densities *model−type* &optional *(reference−area (element−area *soma*))* [*Function*]

Convert all channel or synapse types, depending on whether MODEL−TYPE is 'CHANNEL−TYPE or 'SYNAPSE−TYPE, from absolute gbars or permeabilities to densities, where REFERENCE−AREA is in um2.

set-element-absolute-iv-reference *element iv−reference* [*Function*]

Set the gbar or permeability for the synapse or channel ELEMENT to an absolute value IV−REFERENCE (uS or cm3/s). To get the current value, use ELEMENT−GBAR.

element-iv-density *element* [*Function*]

Returns the effective gbar or permeability density in pS/um2 or 1.0e−6cm3/s/um2 for ELEMENT.

element-conductance *element* [*Function*]

Returns the conductance in uS for ELEMENT.

element-relative-conductance *element* [*Function*]

Returns the actual conductance divided by the GBAR for ELEMENT.

element-gbar *element* &optional *cell−type always−update element−type* [*Function*]

Returns the total effective GBAR of all the elements associated with ELEMENT in uS.

set-element-membrane-parameters *element* &optional *ignore−membrane−elements* [*Function*]

Use when ELEMENT dimensions or gbar ref change.

element-value *element* &key *(target−time *real−time*) (time−list (current−sim−plot−time−list)) data−type dt data−list* [*Function*]

Returns the value of ELEMENT of DATA−TYPE [default given by the function DEFAULT−DATA−TYPE] associated with TARGET−TIME [ms, default *REAL−TIME*]. If TARGET−TIME is not equal to the value of *REAL−TIME*. then the DATA−TYPE for ELEMENT must have already been specified for saving, e.g. by an ENABLE−ELEMENT−PLOT. Element data time base is given by TIME−LIST [default given by CURRENT−SIM−PLOT−TIME−LIST]. Data values for times between simulation time points are linear interpolations. Original data can be explicitly resampled by included the arg DT [ms].

distals-without *distal−border* &optional *(model−type 'segment) cell* [*Function*]

Returns all elements of MODEL−TYPE associated with CELL (if specified, if not then all in circuit) that are further from the soma than DISTAL−BORDER (microns).

proximals-within *proximal−border* &optional *(model−type 'segment) cell* [*Function*]

Returns all elements of MODEL−TYPE associated with CELL (if specified, if not then all in circuit) that are closer to the soma than PROXIMAL−BORDER (microns).

neighbors *target radius* &optional *restrict−to−cell−of−target* [*Function*]

Returns list of all elements of the same type as TARGE, when TARGET is a membrane element, or the same model type if it is a cell element, which lie at most RADIUS microns away. If RESTRICT−TO−CELL−OF−TARGET is T, then only consider elements that are part of the same cell as TARGET.

erase-element-type *elt* [*Function*]

Specifically for removing all elements of a given type, where ELT is either a instance of a type or points to the type itself.

element-type *element* &optional *model−type create* [*Function*]

Returns as values a list (if more than one) or atom (if just one) of all element types associated with ELEMENT of MODEL−TYPE, and also a flag if any new types are created (when CREATE is non−NIL). If CREATE is non−NIL [default NIL], then element types associated with symbols in ELEMENT will be created according to their definitions in the parameter library.

cell-element *element* &optional *model−type* [*Function*]

Returns a list (if more than one) or atom (if just one) of all cell elements associated with ELEMENT of MODEL−TYPE.

element-soma *element* &optional *model−type* [*Function*]

Returns a list (if more than one) or atom (if just one) of all soma associated with the elements associated with ELEMENT of MODEL−TYPE.

element-cell *element* &optional *model−type* [*Function*]

Returns a list (if more than one) or atom (if just one) of all cells associated with the elements associated with ELEMENT of MODEL−TYPE.

erase-element *element* &optional *model−type (remove−segment−from−cell t) just−erase−top−element* [*Function*]

Erase ELEMENT, if it is singular, or the members of ELEMENT, if it is a list, with the qualification that all erased elements are of MODEL−TYPE, if this arg is included. If a segment is to be erased, then be sure to remove it from its cell when REMOVE−SEGMENT−FROM−CELL is non−nil. For erasing all segments of a cell, it is more efficient to remove the segments from the cell's :SEGMENTS slot separately. If JUST−ERASE−TOP−ELEMENT is nil [default], then remove all elements which are components of an erased element (i.e. particles from a channel). ELEMENT can also be a symbol such as 'SYNAPSE or 'CHANNEL, in which case all instances of that type of element will be erased.

print-element *element* &optional *model−type (stream *standard−output*)* [*Function*]

Print documentation apropos for ELEMENT.

disable-element *element* &optional *model−type* [*Function*]

Generic disable (blocking) for ELEMENT.

enable-element *element* &optional *model−type* [*Function*]

Generic enable (unblocking) for ELEMENT.

document-element *element* &optional *model−type* [*Function*]

Generates loadable description of ELEMENT.

edit-element *element* &optional *model−type* [*Function*]

Edit menu for properties of ELEMENT.

element-name *element* &optional *model−type* [*Function*]

The printed name for ELEMENT.

element-control-waveform *element* [*Function*]

Return the controlling sequence (list or array) associated with ELEMENT, if it exists. Examples include conductance waveforms for event synapses or synapse types, explicit current or voltage waveforms for current or voltage sources, respectively.

element-control-waveform-timestep *element* [*Function*]

Return the timestep associated with the controlling sequence associated with ELEMENT as given by the function ELEMENT−CONTROL−WAVEFORM, if this sequencee exists. Otherwise return NIL.

element-distribution *type total−number targets distribution−function* &rest *distribution−function−args* [*Function*]

Add TOTAL−NUMBER elements of TYPE to cell elements associated with TARGETS with a probability given by DISTRIBUTION−FUNCTION and the optional DISTRIBUTION−FUNCTION−ARGS. DISTRIBUTION−FUNCTION may be either a function or the keyword :FLAT. In the first case, the function is applied to the distance to the soma of each cell element. The first argument of DISTRIBUTION−FUNCTION is the cell element, with possible additional arguments given by DISTRIBUTION−FUNCTION−ARGS, and DISTRIBUTION−FUNCTION must return a single−float. If DISTRIBUTION−FUNCTION is set to :FLAT, then the targets that receive an element of TYPE are chosen with an equal probability.

cumulative-pdf-element-distribution *type total−number targets distribution−function* &rest *distribution−function−args* [*Function*]

Add TOTAL−NUMBER elements of TYPE to cell elements associated with TARGETS with a probability given by DISTRIBUTION−FUNCTION and its DISTRIBUTION−FUNCTION−ARGS, applied to each target. It is assumed that the first argument of DISTRIBUTION−FUNCTION is the target, with possible additional arguments given by DISTRIBUTION−FUNCTION−ARGS. DISTRIBUTION−FUNCTION must return a single−float. TOTAL−NUMBER must be a fixnum. The cumulative probability distribution function for DISTRIBUTION−FUNCTION is integrated over all the TARGETS, with a final value of TOTAL−NUMBER, and each target is assigned the value of the cpdf after taking into account their contribution to the integral (given by applying DISTRIBUTION−FUNCTION to the given target). Elements of TYPE are added one at a time, by generating a random number (range given by TOTAL−NUMBER) THIS−ROLL and finding the target which is associated with THIS−ROLL in the cumulative PDf.

gaussian-element-distribution *type targets total−number mean−distance sd* [*Function*]

Add TOTAL−NUMBER elements of TYPE to cell elements associated with TARGETS with a probability given by a gaussian of the difference between MEAN−DISTANCE and the distance to the soma of each cell element, with a standard deviation of SD.

11 SYS Source File: element-functions-2.lisp

element-data *element* &optional *data−type model−type state−index* [*Function*]

Given ELEMENT or elements of type MODEL−TYPE, returns the plot data list [in correct time order, according to the list of times in CURRENT−SIM−PLOT−TIME−LIST. or a list of lists [for more than one element] of type given by DATA−TYPE. The possible DATA−TYPE for the different element model−types are:

```
Element Model-Type       Data Type [first is default, given by the
                              function DEFAULT-DATA-TYPE]
------------              ------------------------------
SOMA                      'VOLTAGE, 'DVDT, 'DENDRITE-CURRENT
SEGMENT                   'VOLTAGE, 'DVDT
```

```
EXTRACELLULAR-ELECTRODE   'FIELD-POTENTIAL
AXON                      'VOLTAGE
CHANNEL, SYNAPSE          'CURRENT, 'REVERSAL-POTENTIAL,
                          'CONDUCTANCE
ISOURCE                   'CURRENT
VSOURCE                   'CURRENT, 'VOLTAGE
PARTICLE                  'STATE, 'MARKOV-STATE
CONC-PARTICLE             'STATE
CONC-INT                  'TOTAL, 1, 2, 3 (numbers refer to shells or compartments)
BUFFER                    'CONCENTRATION
PUMP                      'CURRENT
```

The STATE−INDEX argument is used when retrieving state data of Markov gating particles.

element-data-dted *element* &optional *(dt 1.0) data−type model−type (time−base (current−sim−plot−time−list)) state−index* [*Function*]

Given an element or elements in ELEMENT or element type MODEL−TYPE, returns a simulation data list (or lists for more than one element) of type DATA−TYPE [as is ELEMENT−DATA] sampled on an even time base as given by the optional DT [milliseconds, default 1.0]. The time base for the original data is taken from TIME−BASE, which is either a list of numbers or a single number, in which case this is the even time grid of the original data. [Bug] Note that the original time base must include steps on the order of DT in order for proper sampling during simulation periods of long time steps.

element-location *element* &optional *model−type* [*Function*]

Returns the XYZ coordinates [microns] of the cell element node associated with ELEMENT of MODEL−TYPE.

where *element* &optional *model−type* [*Function*]

Returns the XYZ coordinates [microns] of the cell element node associated with ELEMENT of MODEL−TYPE.

where-x *element* &optional *model−type* [*Function*]

Returns the X coordinate [microns] of the cell element node associated with ELEMENT of MODEL−TYPE.

where-y *element* &optional *model−type* [*Function*]

Returns the Y coordinate [microns] of the cell element node associated with ELEMENT of MODEL−TYPE.

where-z *element* &optional *model−type* [*Function*]

Returns the Z coordinate [microns] of the cell element node associated with ELEMENT of MODEL−TYPE.

min-max-circuit-coordinates [*Function*]

Return as values the minimum and maximum coordinates of the curent circuit cell elements:

```
(min-x max-x min-y max-y min-z max-z)
```

as-the-crow-flies *location−1 location−2* [*Function*]

Returns the straight line distance between LOCATION−1 and LOCATION−2, where the arguments can either be references to circuit elements or explicit location lists (X Y Z).

element-cloud *reference−element cloud−radius* &optional *restrict−to−reference−element−cell returned−model−type* [*Function*]

Return a list of elements of RETURNED−MODEL−TYPE [somas and segments if this is NIL, the default] that are within CLOUD−RADIUS [microns] of REFERENCE−ELEMENT. Candidate returned elements are restricted to the cell associated with REFERENCE−ELEMENT when RESTRICT−TO−REFERENCE−ELEMENT−CELL is non−nil.

closest-element *element* &key *exclude−these−elements proximal−measure candidates just−somas* [*Function*]

Returns the closest soma or segment to ELEMENT, calculated with AS−THE−CROW−FLIES, taken from cell elements in CANDIDATES and excluding those in EXCLUDE−THESE−ELEMENTS. ELEMENT may also be an explicit location list (x y z), with each value in microns. The second returned value is the distance between ELEMENT and the closest cell element in microns. If CANDIDATES is not supplied then all segments and the soma of the cell associated with ELEMENT are used, unless ELEMENT is a location, then all cell elements in the circuit are tested. If PROXIMAL−MEASURE is non−NIL, then the proximal location of ELEMENT is used, otherwise the distal location is used to calculate the distance metric. If JUST−SOMAS is T, then candidate elements are restricted to somas.

distance-to-soma *element* [*Function*]

Given an ELEMENT (name or object), returns the distance along the tree to the soma in microns. Faster way is to reference the :SEGMENT−DISTANCE−TO−SOMA slot of the segment which is set when the cell anatomy is first processed.

electrotonic-distance-to-soma *element* [*Function*]

Given an ELEMENT (name or object), returns the electrotonic distance along the tree to the soma in microns.

tree-radius &optional *(cell *cell*) (defined−as :max)* [*Function*]

Given the optional CELL (default *CELL*), each of the distal tip segments are compared, and depending on the setting of DEFINED−AS (default :MAX) the maximum (DEFINED−AS equals :MAX), the minimum (DEFINED−AS equals :MIN), or the average (DEFINED−AS equals T) of the distances to the soma of the segments is returned.

distal-tips &optional *cell* [*Function*]

Return a list of all distal tip segments associated with CELL, if supplied, otherwise, all distal tips in circuit.

segments-out *element* &optional *(segment−skip 0) previous−segs* [*Function*]

Starting with the cell element associated with ELEMENT, returns a list of all the segments moving distally, skipping by SEGMENT−SKIP [default 0].

segments-in *element* &optional *(segment−skip 0)* [*Function*]

Returns an inclusive list of all the segments starting from the segment associated with ELEMENT on the path to the soma, skipping by SEGMENT−SKIP [default 0].

trunk-segment *element* [*Function*]

Returns the trunk segment associated with the dendritic branch that includes ELEMENT.

trunk-segments &optional *element* [*Function*]

Return a list of trunk segments for the cell associated with ELEMENT, if supplied, otherwise all in circuit.

primary-segs &optional *element* [*Function*]

Returns a list of all segments of the cell of ELEMENT, if supplied, otherwise all in circuit, that are proximal to the first branch point.

soma-segments &optional *target* [*Function*]

Returns a list of segments which are conceptually assigned to the actual cell soma.

cell-distal-segments &optional *(cell *cell*)* [*Function*]

Returns a list of all the distal segments of CELL [default *CELL*].

element-param-distribution *model−type parameter* &key *parameter−function cell note param−max param−min type−for−title x−label y−label x−min x−max x−inc (x−axis−tick−skip 0) x−are−fns y−min y−max y−inc (y−axis−tick−skip 0) y−are−fns bin−width include−simulation−name (width 350) (height 300) font title−position create−new−window* [*Function*]

Plots histogram of properties of all elements associated with MODEL−TYPE (associated with CELL, if supplied, otherwise all in the circuit). MODEL−TYPE either refers to a class of elements (e.g. 'SEGMENT, 'CHANNEL, 'SYNAPSE) or a specific element type (e.g. a particular channel or synapse type). PARAMETER must be consistent with MODEL−TYPE, and for a given ELT of MODEL−TYPE can include:

```
'AREA         =>  (element-area elt)
'DISTANCE     =>  (distance-to-soma elt)
'DIAMETER     =>  (element-diameter elt)
'CAPACITANCE  =>  (element-capacitance elt)
'GBAR         =>  (element-gbar elt)
```

If ELT is on a segment, then:

```
'LENGTH       =>  (segment-length (element-cell-element elt))
```

membrane-area-distribution &optional *(cell *cell*)* &key *x−axis−tick−skip plot−pdf−histogram (distance−increment 10) histogram−x−max y−inc return−distribution (pdf−title "distribution of membrane area vs. distance to soma")* [*Function*]

Construct the distribution of membrane area as a function of distance from the soma for all cell elements referenced by CELL

`[default *CELL*]. Resulting distribution is binned with increments given by DISTANCE-INCREMENT, in microns [default 10], and is`

plotted as a histogram with PDF−TITLE when PLOT−PDF−HISTOGRAM is T. Y−INC and X−AXIS−TICK−SKIP apply to the plotted histogram, as specified in the function PLOT−HISTOGRAM. The distribution is returned as a list of (Distances Areas) when RETURN−DISTRIBUTION is T.

element-diameter *element* &optional *new−diameter* [*Function*]

For the the cell element of ELEMENT, return the diameter of the segment or soma in microns. If NEW−DIAMETER is a number, then the cell element diameter will be changed to this value, and *CIRCUIT−PROCESSED* will be set to NIL.

element-length *element* &optional *new−length* [*Function*]

When the cell element of ELEMENT is a segment, return the length of the segment in microns. If NEW−LENGTH is a number, then the segment length will be changed to this value, and *CIRCUIT−PROCESSED* will be set to NIL.

soma-area *cell−element* &optional *consider−virtual−elements* [*Function*]

Return the membrane area of the soma associated with CELL−ELEMENT, in um2. The area of any virtual soma segments is included when CONSIDER−VIRTUAL−ELEMENTS is T.

segment-area *cell−element* [*Function*]

Return the membrane area of the segment associated with CELL−ELEMENT, in um2.

element-area *element* &optional *consider−virtual−elements model−type* [*Function*]

The total area of somas and segments associated with ELEMENT, in square microns (single float). Segment areas do not include the cylinder ends (only the lateral areas are considered). If somas have 'ADJUST−AREA−FOR−TRUNKS parameter, then their area is adjusted for the areas of the faces of any abutting segments. If CONSIDER−VIRTUAL−ELEMENTS, any virtual soma segments will be included in somatic area calculations.

element-volume *element* &optional *consider−virtual−elements model−type* [*Function*]

Total volume of cell elements associated with ELEMENT in cubic microns (single−float). If a cell is given by ELEMENT, then the total cell volume is considered.

element-sv-ratio *element* &optional *consider−virtual−elements model−type* [*Function*]

Ratio of area divided by volume of the cell elements associated with ELEMENT, in 1/microns (single−float).

element-concentration-volume *element* &optional *consider−virtual−elements model−type* [*Function*]

Returns the volume in um3 of the cell element associated with ELEMENT, minus the volume of any nucleus associated with the cell element, as indicated by the element parameter 'nucleus−diameter in microns.

all-data-types *element* &optional *model−type* [*Function*]

Returns a list of symbols corresponding to all possible types of plot data appropriate for ELEMENT or for the associated child type.

default-data-type *element* &optional *model−type model* [*Function*]

Returns a symbol corresponding the default type of plot data appropriate for ELEMENT or for the associated child type. Apropos for element types with more than one type of data.

disable-all-element-plot [*Function*]

Disables plotting of all circuit elements.

plotted-elements [*Function*]

Return a list of all element names and the data types for which data is currently saved.

enable-element-plot *element* &optional *data−type model−type* [*Function*]

Enable plot of DATA−TYPE (as in ELEMENT−DATA, but also including 'EVENT for axons and synapses) of elements in ELEMENT of MODEL−TYPE. If ELEMENT is an element type, then all elements of that type are affected. For elements that can generate more than one type of simulation data, setting DATA−TYPE to :ALL will enable all plotting of all data types (except for events). DATA−TYPE may also be a list of data types.

disable-element-plot *element* &optional *data−type model−type* [*Function*]

Disables plot of DATA−TYPE (as in ELEMENT−DATA, but also including 'EVENT for axons and synapses) of elements in ELEMENT of MODEL−TYPE. If ELEMENT is an element type, then all elements of that type are affected. For elements that can generate more than one type of simulation data, setting DATA−TYPE to :ALL will disable all plotting of all data types. Setting ELEMENT to :ALL will disable all plotting, period. DATA−TYPE may also be a list of data types.

setup-plot-total-conductances *spec−list* [*Function*]

Set up and enable plotting of total conductances with a SPEC−LIST whose format is described in the documentation for *PLOT−TOTAL−CONDUCTANCES*. To plot the total conductance of all cells in the circuit:

```
(SETUP-PLOT-TOTAL-CONDUCTANCES :ALL)
```

clear-plot-total-conductances [*Function*]

Clear and disable plotting of total conductances.

plot-element *element* [*Function*]

Takes a single element or a list of elements for ELEMENT, and plots the intrinsic characteristics [not simulation data] of the associated element type.

plot-segments-to-soma *element* &optional *(segment−skip 0) clear−first* [*Function*]

Enables plotting on a separate window all the segments on the path from the ELEMENT to the soma, skipping path segments by SEGMENT−SKIP [default 0]. If CLEAR−FIRST [default NIL] is T, then any segments previously including in such a plot are cleared first.

12 SYS Source File: math.lisp

sphere-diameter-from-capacitance *capacitance* &optional *(specific−capacitance 1.0)* [*Function*]

Returns the diameter in microns of a sphere with CAPACITANCE in pF, assuming a SPECIFIC−CAPACITANCE in uF/cm2 [default 1].

sphere-area *radius−microns* [*Function*]

Sphere surface area is in um2 − RADIUS−MICRONS is in micrometers.

sphere-area-cm2 *radius−microns* [*Function*]

Sphere surface area is in cm2 − RADIUS−MICRONS is in micrometers.

sphere-area-from-diameter *diameter* [*Function*]

Sphere surface area is in cm2 − DIAMETER is in micrometers.

sphere-diameter-from-area *area* [*Function*]

Returns the diameter in microns of a sphere with AREA in um2.

sphere-volume *radius* [*Function*]

Returns volume in um3 of sphere with RADIUS (single float) in um (single float).

sphere-volume-from-diameter *diameter* [*Function*]

Returns volume in um3 of sphere with DIAMETER (single float) in um (single float).

cylinder-volume *length diameter* [*Function*]

Returns volume in um3 of cylinder with LENGTH and DIAMETER (both single floats) in um (single float).

fast-fractional-part *number* [*Function*]

Returns second result of TRUNCATE, where NUMBER is double float.

real-from-mag-phase *mag phase* [*Function*]

Return the single value real part of the complex number described by the single floats MAG and PHASE.

imag-from-mag-phase *mag phase* [*Function*]

Return the single value imaginary part of the complex number described by the single floats MAG and PHASE.

float-mod *number divisor* [*Function*]

Returns second result of FLOOR. NUMBER and DIVISOR must be single floats.

nonlinearity *input* &optional *nonlinearity (parameter 0.0)* [*Function*]

Pass the single float INPUT through a nonlinearity and return the single float result. Default for PARAMETER is 0.

```
:NONLINEARITY           :PARAMETER              COMMENT
    NIL                    n/a                  linear
:THRESHOLD              Threshold    (if INPUT >= threshold then INPUT, else 0.0)
:NEGATE-THRESHOLD       Threshold    (if INPUT >= threshold then (- INPUT), else 0.0)
:BELOW-THRESHOLD        Threshold    (if INPUT <= threshold then INPUT, else 0.0)
:NEGATE-BELOW-THRESHOLD Threshold    (if INPUT <= threshold then (- INPUT), else 0.0)
:RECTIFY                   n/a          full wave, i.e. x>0 -> x, x<0 -> -x
```

notify-exp-limit *nil* [*Variable*]

When T print out message when one of EXP−W−LIMITS functions punts.

13 SYS Source File: statistics.lisp

sample-s *data−list* [*Function*]

Returns the sample sigma of the values in DATA−LIST.

14 SYS Source File: fft.lisp

element-data-dft *element* &key *data−type type state−index (delta−t 1.0) (reference−time−list (current−sim−plot−time−list)) (dc−offset 0.0)* [*Function*]

Plot magnitude and phase of the DFT of ELEMENT data, resampled on a regular grid given by DELTA−T [milliseconds, default 1.0]. DC−OFFSET [default 0.0] is subtracted from the data before the DFT. Remaining arguments are as for ELEMENT−DATA−DTED. DFT processing done by DFT−STRETCH−WAVE, which may also resample the data.

15 SYS Source File: randoms.lisp

poisson-interval *lambda* [*Function*]

Returns an time interval from a poisson distribution with rate LAMBDA [single float].

poisson-events *lambda start stop* &optional *(min−interval−value 0.0)* [*Function*]

Returns a list of times generated by a Poisson process with rate constant LAMBDA [1/ms], starting at START and ending with STOP [ms]. Poisson intervals are taken at a minimum value of MIN−INTERVAL−VALUE [ms].

modulated-poisson-events *lambda−spec start stop* &key *(step 1.0) (time−offset 0.0) (min−interval−value 0.0) (lambda−coefficient 1.0)* [*Function*]

Returns a list of event times generated by a Poisson process with a modulated rate constant lambda over a total interval given by START and STOP [ms]. Calculation of lambda at any given time depends on LAMBDA−SPEC. When LAMBDA−SPEC is a function, lambda is given by a funcall of LAMBDA−SPEC with the time as the argument (this function should take a single numeric argument and return a single−float). When LAMBDA−SPEC is a sequence lambda is the interpolated value of LAMBDA−SPEC appropriate for the current time. Finally, if LAMBDA−SPEC is a number this gives directly the constant value of lambda. In all cases, the evaluation of LAMBDA−SPEC is taken to be in 1/ms. When LAMBDA−SPEC is a function, STEP [ms] should be a value for which the function is relatively constant. When LAMBDA−SPEC is a sequence, then STEP is the time base. If LAMBDA−SPEC is a number, then the list of events is generated by the function POISSON−EVENTS, with the START and STOP arguments adjusted by the value of TIME−OFFSET [ms, default 0.0]. Otherwise, the events are generated by repeated calls to POISSON−INTERVAL, with the initial time given by START, and until time reaches STOP. At every time increment if POISSON−INTERVAL gives a value greater than STEP, then the time is incremented by STEP and the process repeated. If the returned poisson interval is less than STEP, then the time is incremented by the interval, and this value is pushed onto the result list. All times in the returned list are adjusted by the addition of TIME−OFFSET [ms]. Poisson intervals are taken at a minimum value of MIN−INTERVAL−VALUE [ms]. All numeric arguments must be single floats.

exponential-pdf *lambda* [*Function*]

Returns a random sample (single float) out of an exponential PDF with characteristic decay constant LAMBDA (must be a single float).

shuffled-indices *length* [*Function*]

Return of list of integers ranging from 0 to (LENGTH − 1), in random order.

shuffled-list *list* [*Function*]

Return of scrambled version of LIST.

random-phase-sequence *duration* &key *(low−cutoff 0) high−cutoff (delta−t 1.0) (mag−function :flat) (verbose−plot−titles t) (rms 1) (rms−on−infinite−sequence t) (ac−coupled t) plot−mag−phase plot−result plot−dft−sequences return−list* [*Function*]

Generate a single float list of length [maximum (expt 2 16)] given by the ratio of DURATION and DELTA−T, the latter in units of ms [default 1], whose frequency spectrum magnitude is given by MAG−FUNCTION − :FLAT [default], :1/F or :1/F−SQUARED − between LOW−CUTOFF [default 0] and HIGH−CUTOFF [default NIL, both in hz when numeric], otherwise 0, and whose phase is given by a random deviate with flat distribution between −Pi and Pi. When HIGH−CUTOFF is NIL then there is no low pass characteristic. The DC component is included only if AC−COUPLED is NIL [default T]. The list is returned if RETURN−LIST is T [default NIL], and the output plotted if PLOT−RESULT is T [default NIL]. Intermediate results are plotted if PLOT−DFT−SEQUENCES is T [default NIL]. The average power in the output is given by RMS. This constraint is applied on the corresponding infinite output sequence if RMS−ON−INFINITE−SEQUENCE is T [default], otherwise on the actual finite length output sequence. For reasonable reproduction of the highest frequency components in the output list, the product of DELTA−T and the maximum frequency with significant power in the passband should be at most 100. VERBOSE−PLOT−TITLES [default T] enables spectrum information added to non−DFT plot window titles.

exponential-random-number [*Function*]

Returns a double−float random number taken from an exponential distribution with lambda of 1.

normal-random-number [*Function*]

Return a double−float random number taken from a normal distribution with mean of 0 and variance of 0.5.

sf-random-not-zero *x* [*Function*]

Returns a random value, of type single−float, between 0.0 and x. The returned value is guaranteed to be different from 0.0 and x. Argument x must be of type single−float.

df-random-not-zero *x* [*Function*]

Returns a random value, of type double−float, between 0.0 and x. The returned value is guaranteed to be different from 0.0 and x. Argument x must be of type double−float.

random-not-zero *x* [*Function*]

Returns a random value, of type double−float or single−float depending on x, between 0.0 and x. The returned value is guaranteed to be different from 0.0 and x. The properly optimized function is called.

random-nth *list* &optional *(total 1)* [*Function*]

Returns TOTAL [default 1] elements picked randomly from LIST, without replacement.

random-nth-fraction *list* &optional *(fraction 1.0)* [*Function*]

Returns a FRACTION [default 1.0] of the elements in LIST, picked randomly without replacement.

random-subseq *liste number* [*Function*]

Returns a random sub−list of a list. Non−destructive.

16 SYS Source File: renewal-process.lisp

sf-find-poisson-waiting-time *rate* [*Function*]

Returns a realization of an interval, in a Poisson process of the given 'rate'. if rate is in spikes/sec, the interval will be in seconds.

sf-find-gamma-waiting-time *lambda order* [*Function*]

Returns a realization of a random variable distributed following a gamma distribution of parameter 'lambda' and of given 'order'. order must be an integer. For t>=0,

$$g(t) = P(t<T<t+dt) = \lambda \frac{(\lambda t)^{(order-1)}}{(r-1)!} \exp(-\lambda t)$$

IMPORTANT: If lambda is given in spikes/sec, the interval generated is in milliseconds.

default-thinning-step-acons *'((11.0 . 1) (75.0 . 5) (1000.0 . 50))* [*Variable*]

this acons holds the stepwise function relating the thinning−factor to the current frequency.

correlated-unit-rate-poisson-processes *correlation start stop* [*Function*]

correlation is proportion of correlation, between 0 and 1.

test-correlated-processes *correlation* [*Function*]

Tests the above routine. This should return a float close to correlation.

serial-coeff *interval−seq order* [*Function*]

'interval−seq' is a list of *INTERVALS*. 'order' is the distance between intervals.

17 SYS Source File: waveforms.lisp

nth-derivative *data time−base n* [*Function*]

Returns as values the Nth derivative of DATA, and the associated time list, derived from TIME−BASE, which can be a list of single floats or a single number, in which case it is taken as dt. Recursively calls DIFFERENTIATE−WAVE.

differentiate-wave *wave* &optional *(time−spec 1.0)* [*Function*]

Given a n−valued sequence WAVE with values

```
[x0 x1 x2 ... xn-1]
```

returns an (n−1)−valued list with values

```
[(x1-x0)/TIME-SPEC, (x2-x1)/TIME-SPEC, ... (x(i)-x(i-1))/TIME-SPEC, ... (x(n-1)-x(n-2))/TIME-SPEC]
```

if TIME−SPEC is a number (corresponding to delta−T). Otherwise, if TIME−SPEC is a sequence, then returns

```
[(x1-x0)/(T1-T0), (x2-x1)/(T1 - T2), ... (x(n-1)-x(n-2))/(T(n-1)-T(n-2)]
```

where Tn = (NTH n TIME−SPEC).

differentiate-float-wave *wave* &optional *(time−spec 1.0) array−output* [*Function*]

As for DIFFERENTIATE−WAVE, but assumes single float values.

midpoints *wave* [*Function*]

Given a n−valued sequence WAVE with values

```
[x0 x1 x2 ... xn-1]
```

returns an (n−1)−valued list with values

```
[(x0 + x1)/2, (x1 + x2)/2, ... (xn-2 + xn-1)/2]
```

element-data-window *element start stop* &key *(dt 0.1) data−list data−type model−type (time−base (current−sim−plot−time−list)) state−index* [*Function*]

Returns element data for the time window defined between START and STOP, in milliseconds. Remaining arguments are as for ELEMENT−DATA−DTED. If DATA−LIST is supplied, the sampled data is taken directly from this list, which is assumed to be on a time grid of DT, and ELEMENT is ignored.

data-window *data−list start stop dt* [*Function*]

Return a sub−list of a timed sequence DATA−LIST, sampled at DT starting at time=0, from START to STOP.

expand-time-reference *time−base length* &optional *(start−time 0.0)* [*Function*]

If TIME−BASE is a list, return that list. Otherwise return the list made by (LIST−OF−NUMS LENGTH START−TIME TIME−BASE).

element-spike-times *element* &key *(spike−threshold −20.0) (sub−threshold−time 0.5) (supra−threshold−duration−min 0.1) model−type data−list start−time (time−base (current−sim−plot−time−list))* [*Function*]

Returns a list of postive threshold crossings from the voltage of the soma or segment associated with ELEMENT of MODEL−TYPE, according to the SPIKE−THRESHOLD [mV], SUPRA−THRESHOLD−DURATION−MIN, SUB−THRESHOLD−TIME. All times are in milliseconds, and are referenced from the time that the voltage last went above SPIKE−THRESHOLD. If DATA−LIST is supplied, the sampled data is taken directly from this list, and the ELEMENT argument is ignored. The time base is given by TIME−BASE [default given by the function CURRENT−SIM−PLOT−TIME−LIST], after processing by the function EXPAND−TIME−REFERENCE.

element-spike-heights *element* &key *(spike−threshold −20.0) (supra−threshold−duration−min 0.1) (sub−threshold−time 0.5) model−type data−list element−spike−times (time−base (current−sim−plot−time−list))* [*Function*]

Finds subsequent maximum voltages [when dV/dT = 0] referenced from spikes as generated by calling the function ELEMENT−SPIKE−TIMES. Otherwise, the spike times can be supplied by an explicit list of ELEMENT−SPIKE−TIMES. Remaining arguments are as for the function ELEMENT−SPIKE−TIMES. Returns as values a list of the max voltages and a list of times for the maximum voltages.

element-spike-thresholds *element* &key *(spike−threshold −20.0) (supra−threshold−duration−min 0.1) (sub−threshold−time 0.5) model−type data−list element−spike−times (minimum−d2vdt2 1.0) plot−d2v−dt2 (time−base (current−sim−plot−time−list))* [*Function*]

Returns a list of the precendent threshold values [maximum d2V/dT2 greater than MINIMUM−MAX−D2VDT2, mV/ms2, default 1.0e3] referenced from spikes as generated by calling the function ELEMENT−SPIKE−TIMES. Otherwise, the spike times can be supplied by an explicit list of ELEMENT−SPIKE−TIMES. Remaining arguments are as for the function ELEMENT−SPIKE−TIMES. Returns as values a list of the threshold voltages and a list of times for the thresholds.

element-firing-frequency *element* &key *(spike−threshold −20.0) (supra−threshold−duration−min 0.1) (sub−threshold−time 0.5) model−type data−list (time−base (current−sim−plot−time−list)) (start−time 0) (end−time *user−stop−time*)* [*Function*]

Returns the firing frequency in Hz from spikes detected from the voltage of the soma or segment associated with ELEMENT of MODEL−TYPE, between START−TIME [ms] and END−TIME. Keyword arguments for spike detection as used by ELEMENT−SPIKE−TIMES.

element-extreme *element* &key *data−type (min−time 0.0) (max−time *user−stop−time*) dt maxp (what :value) data−list model−type (time−list (current−sim−plot−time−list))* [*Function*]

For data in DATA−LIST, if supplied, otherwise from data of DATA−TYPE of ELEMENT of MODEL−TYPE, call DATA−EXTREME with remaining arguments.

data-extreme &key *(min−time 0.0) (max−time *user−stop−time*) dt maxp (what :value) data−list (time−list (current−sim−plot−time−list))* [*Function*]

Analysis of DATA−LIST, considered with respect to a time base of step DT [ms], if supplied, otherwise from times in TIME−LIST. The maximum [respectively minimum], depending on MAXP of WHAT [:SLOPE, 1ST−DERIVATIVE (same as :SLOPE), :2ND−DERIVATIVE, :VALUE (default)], within a time window between MIN−TIME [ms] and MAX−TIME [ms]. Returns as values the extreme value and the time for which that value was detected. If no extreme was detected, then returns as values 0.0 and MIN−TIME. Data units are as appropriate for the type of data.

element-amplitude *element* &key *data−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list model−type negative−p base−level* [*Function*]

Returns the amplitude in units appropriate for the type of data in DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. The reference level for the rise time is given by BASE−LEVEL [assumed to be in the units corresponding to that of the data] if supplied, otherwise the reference is taken as the minimum (respectively maximum) when NEGATIVE−P is NIL, (respectively T). The measured event amplitude is either the maximum or minimum value thereafter, again depending on NEGATIVE−P. Additional arguments are as for ELEMENT−EXTREME.

data-amplitude &key *(min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list negative−p base−level* [*Function*]

Same as ELEMENT−AMPLITUDE, except that DATA−LIST must be supplied.

element-10-90-rise-time *element* &key *data−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list model−type negative−p base−level* [*Function*]

Returns the time in milliseconds for the 10% to 90% rise time applied to DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. Remaining arguments are as for ELEMENT−AMPLITUDE and ELEMENT−EXTREME.

element-10-90-slope *element* &key *data−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list model−type negative−p base−level* [*Function*]

Returns the slope in units/ms for the 10% to 90% rise time applied to the DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. Remaining arguments are as for ELEMENT−AMPLITUDE and ELEMENT−EXTREME.

element-max-slope *element* &key *data−type model−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Returns the maximum slope in units/ms applied to the DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. Remaining arguments are as for ELEMENT−EXTREME.

data-max-slope &key *(min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Same as ELEMENT−MAX−SLOPE, except that DATA−LIST must be supplied.

element-min-slope *element* &key *data−type model−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Returns the minimum slope in units/ms applied to the DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. Remaining arguments are as for ELEMENT−EXTREME.

data-min-slope &key *(min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Same as ELEMENT−MIN−SLOPE, except that DATA−LIST must be supplied.

element-max *element* &key *data−type model−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Returns the maximum applied to the DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT. Remaining arguments are as for ELEMENT−EXTREME.

data-max &key *(min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Same as ELEMENT−MAX, except that DATA−LIST must be supplied.

element-min *element* &key *data−type model−type (min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Returns the maximum applied to the DATA−LIST, if supplied, otherwise to the data of DATA−TYPE of ELEMENT of MODEL−TYPE. Remaining arguments are as for ELEMENT−EXTREME.

data-min &key *(min−time 0.0) (max−time *user−stop−time*) dt (time−list (current−sim−plot−time−list)) data−list* [*Function*]

Same as ELEMENT−MIN, except that DATA−LIST must be supplied.

element-integrated-data *element* &optional *data−type model−type* [*Function*]

According to the plot data and time points of the last simulation, returns the sum of the integrals of the data of type DATA−TYPE of each element in ELEMENT, of element type MODEL−TYPE, where the default DATA−TYPE is given in the documentation for the ELEMENT−DATA function. ELEMENT can either be a single element or a list of elements.

hwhh *data time−base* &optional *max−reference−time* [*Function*]

Simple estimate of the half width at half height of the numeric sequence DATA, indexed by TIME−BASE, which may either be a value for dt or a numeric sequence. Assumes unimodal data, and is sensitive to noise. Maximum of DATA is determined internally, unless a value is given for MAX−REFERENCE−TIME. Returns as values the hwhh and the maximum value of data.

integrate-wave *wave* &optional *(delta−t 1.0) (x−0 0.0)* [*Function*]

Given WAVE, an array or list of numbers assumed to be spaced evenly by DELTA−T with respect to the independent variable, returns a list which is the cumulative integral of WAVE, with the initial conditions given by the optional argument X−0.

list-mins *wave* &optional *(delta−t 1.0) (min 0.0) (min−min−time 0.0)* [*Function*]

For list in WAVE, with time steps DELTA−T, returns two lists as values. These lists are the MIN negative−going and positive−going crossing times, respectively, whenever the duration framing a particular negative−positive pair of MIN crossings is greater than MIN−MIN−TIME.

list-maxs *wave* &optional *(delta−t 1.0) (max 0.0) (min−max−time 0.0)* [*Function*]

For list in WAVE, with time steps DELTA−T, returns two lists as values. These lists are the MAX positive−going and negative−going crossing times, respectively, whenever the duration framing a particular positive−negative pair of MAX crossings is greater than MIN−MAX−TIME.

frame-min-maxs *wave max−min−wave* &optional *(delta−t 1.0) (max 0.0) (min 0.0) (min−min−max−time 0.0) messages* [*Function*]

Strip epochs in WAVE (time step of DELTA−T) according to analysis applied to MAX−MIN−WAVE. Epochs are detected by applying LIST−MINS and LIST−MAXS to WAVE, using MAX and MIN, respectively, and MIN−MIN−MAX−TIME as for the MIN−MIN−TIME and MIN−MAX−TIME arguments, respectively. Returns the processed wave.

find-zero-crossings *wave* &optional *(delta−t 1.0) (min−difference−from−0 0.0)* [*Function*]

For data in WAVE with time step DELTA−T, return a list of the zero−crossing times. True zero crossings are detected when they are framed by alternating polarity amplitudes of at least MIN−DIFFERENCE−FROM−0 [default 0.0].

array-vol *array grid−side* &optional *other−grid−side* [*Function*]

Given 2D ARRAY, with sides GRID−SIDE X OTHER−GRID−SIDE (when optional OTHER−GRID−SIDE supplied), GRID−SIDE X GRID−SIDE otherwise, returns volume of array.

add-delay-to-waveform *waveform delay* &optional *(waveform−time−step 1.0) (delay−value 0.0)* [*Function*]

Adds a series of numbers, given by DELAY−VALUE [default 0.0] to the head of the sequence WAVEFORM. The length of this series is given by DELAY divided by WAVEFORM−TIME−STEP [default 1.0]. The returned sequence is of the same type [cons or array] as the original WAVEFORM.

sinewave &optional *(amplitude 1.0) (duration *user−stop−time*) (frequency 1.0)* &key *(phase 0.0) (offset 0.0) (step 0.2) (start 0.0) zero−before−start* [*Function*]

FREQUENCY is in cycles per unit time, as given by STEP [default 0.2]. PHASE is in degrees. Returns a single−float array. Function times less than START [default 0.0] return 0.0 when ZERO−BEFORE−START is T, otherwise OFFSET. Time argument given to the sin function is relative to START:

```
sin [{2pi * FREQUENCY * (TIME - START)} + PHASE]
```

pulse *delay pulse−duration amplitude total−duration step* [*Function*]

Return a single float pulse waveform [list] of length given by TOTAL−DURATION divided by STEP, which is 0.0 until DELAY, then AMPLITUDE for PULSE−DURATION, then 0.0 for the rest of the list.

wave-cutoff *(exp −6)* [*Variable*]

Relative max cut off value for various waveform creation functions, e.g. ALPHA−ARRAY.

exponential-array-unit-area &optional *(tau 1.0) (step 1.0) length* [*Function*]

Returns an array filled with a decaying exponential, whose amplitude is adjusted so that its area is 1.0 and with time base increment given by STEP [default 1.0]. The length of array is given by LENGTH, if given, otherwise when the amplitude is less than *WAVE−CUTOFF* times the AMPLITUDE.

exponential &optional *(tau 1.0) (step 1.0) (length 0) (offset 0.0) (amplitude 1.0) (start 0.0)* [*Function*]

Returns an array filled with a decaying exponential, of AMPLITUDE [default 1.0] and time base increment given by STEP [default 1.0], and OFFSET [0.0]. The length of array is given by LENGTH, if positive, otherwise when the amplitude without the OFFSET is less than *WAVE−CUTOFF* times the AMPLITUDE. Values before the START [default 0.0] of waveform are given by OFFSET. Function time argument is referenced from START [default 0.0].

double-exponential &optional *(tau−rise 1.0) (tau−fall 1.0)* &key *(amplitude 1.0) normalize (step 1.0) (length 0) (offset 0.0) (start 0.0)* [*Function*]

Returns an array with the difference of two decaying exponentials:

```
AMPLITUDE * [Exp(-t/TAU-FALL) - Exp(-t/TAU-RISE)] + OFFSET
```

AMPLITUDE has a default value of 1.0. Array time base increment is given by STEP [default 1.0]. OFFSET has a default value of 0.0. If NORMALIZE is non−NIL, then the waveform is adjusted and the peak given by (AMPLITUDE + OFFSET). The length of array is given by LENGTH, if positive [default 0], otherwise when the larger of the two exponential terms is less than *WAVE−CUTOFF*. Function time argument is referenced from START [default 0.0].

alpha &optional *(tau 1.0)* &key *(time−exponent 1) (adjustment :normalize) (step 1.0) (duration 0.0) (offset 0.0) (amplitude 1.0) (delay 0.0)* [*Function*]

Returns an array of an alpha function (K * timeA * e(−time/tau)) with time constant TAU [ms], starting at time = DELAY (value prior to DELAY is OFFSET [default 0.0]). The exponent for the leading time coefficient, A, is given by TIME−EXPONENT [default 1] ADJUSTMENT [default :NORMALIZE] determines the value of K as follows:

```
:NORMALIZE  -  K set so that function amplitude is given by AMPLITUDE
:UNIT-AREA  -  K set so that function area is given by AMPLITUDE
ELSE        -  K = 1
```

STEP [ms, default 1.0] gives the time step of the array. The array length is given by DURATION [ms, default 0.0] if positive, otherwise the length is set when the function value is less than *WAVE−CUTOFF* times the maximum. OFFSET adds an offset to the returned array, after the above constraints have been met.

alpha-list &optional *(tau 1.0)* &key *(adjustment :normalize) (step 1.0) (duration 0) (offset 0.0) (amplitude 1.0) (delay 0.0)* [*Function*]

As in ALPHA−ARRAY, but returns a list.

double-alpha &optional *(tau1 1.0) (tau2 1.0) (alpha−proportion 1.0)* &key *(offset 0.0) (step 1.0) (tau1−alpha−area 1.0) (start 0.0)* [*Function*]

Returns an array with the difference of two alpha functions, defined by TAU1 and TAU2 [ms] respectively. The area of the first alpha function is defined with TAU1−ALPHA−AREA [default 1.0], with the relative area of the second given by ALPHA−PROPORTION, with a default of 1.0. Thus the total integral is equal to (TAU1−ALPHA−AREA * (1 − ALPHA−PROPORTION)). The length of the array is determined when the value of the component with the longest time constant is less than than *WAVE−CUTOFF* times its maximum. A correction term is added to the waveform in order to give the proper integral despite the truncated length. The value of OFFSET

`[default 0.0] is added to the final waveform. Function time argument is referenced from START [default 0.0].`

18 SYS Source File: misc.lisp

clear-user-variables &optional *variables−to−keep* [*Function*]

Unintern global variables defined during the current session, other than those given by the symbol or list of symbols in VARIABLES−TO−KEEP [default NIL].

format-time-smallest-unit *:second* [*Variable*]

One of :second, :minute, :hour, :day, :month, :year

format-time-style *:string* [*Variable*]

One of :string, :s−expression, nil. Nil means no formatting.

format-time-include-date-p *t* [*Variable*]

Whether the date is included.

format-time-long-date-p *nil* [*Variable*]

t => February 22, 1958; nil => 2/22/58

load-and-compile-user-source *candidates* &key *src−dir bin−dir* [*Function*]

For the file names in CANDIDATES (namestrings w/o extensions), look in the SRC−DIR (if not supplied, then the "circuits" directory under *SURF−USER−DIR*), compile file and write binary to the BIN−DIR (if not supplied, same directory as above), and load binary.

get-surf-data-directory [*Function*]

Create a directory based on the value of *SURF−USER−DIR*, and return its namestring. If *MAKE−CIRCUIT−SUBDIR* if T, incoporate the current value of *CIRCUIT*:

```
*SURF-USER-DIR*/data/*CIRCUIT*/M_D_Y/
```

where "M_D_Y" is the date. If *MAKE−CIRCUIT−SUBDIR* if NIL:

```
*SURF-USER-DIR*/data/
```

get-surf-plot-directory [*Function*]

Create a directory based on the value of *SURF−USER−DIR*, and return its namestring. If *MAKE−CIRCUIT−SUBDIR* if T, incoporate the current value of *CIRCUIT*:

```
*SURF-USER-DIR*/plot/*CIRCUIT*/M_D_Y/
```

where "M_D_Y" is the date. If *MAKE−CIRCUIT−SUBDIR* if NIL:

```
*SURF-USER-DIR*/plot/
```

load-surf-user-file *filename* [*Function*]

Loads FILENAME which must be in the Surf−Hippo user directory (as specified by *SURF−USER−HOME*.

load-surf-home-file *filename* [*Function*]

Loads FILENAME which must be in the Surf−Hippo home directory (as specified by *SURF−HOME*).

19 SYS Source File: debug.lisp

debug-time-trace *nil* [*Variable*]

When t, prints one line of info at each time step.

debug-at-time-steps *nil* [*Variable*]

When t, prints out the node voltages at every node.

debug-all-iterations *nil* [*Variable*]

When t, prints out the node voltages at every iteration.

print-matrix *nil* [*Variable*]

When t, prints out the matrix at each iteration.

profile-all [*Function*]

Run a simulation while profiling most of the major functions active during integration.

time-innards *iterations* &body *body* [*Macro*]

Prints total execution time for n ITERATIONS of BODY, with local bindings of:

```
*KILL-ALL-OUTPUT* T            (for last n-1 iterations)
*KILL-EXTRA-MESSAGES* T        (for last n-1 iterations)
*SHOW-TIME-REMAINING* NIL
*BEEP-AFTER-GC* NIL
*BEEP-AFTER-SURF* NIL
```

Writes all results to a .timing file in surf−hippo/logs/.

print-obj-size *obj* [*Function*]

Finds the space taken by an object.

20 SYS Source File: pump-preliminaries.lisp

pump-conc-int-compartment-volume *pump* [*Function*]

PUMP compartment volume in cm3.

pump-concentration-current *pump* &optional *compartment−volume* [*Function*]

Returns mM/ms. COMPARTMENT−VOLUME is in cm3.

21 SYS Source File: conc-int.lisp

concentration-clamp *element* &optional *concentration* [*Function*]

Turn off all concentration integrators associated with ELEMENT. If CONCENTRATION is a number [mM, default NIL], then set steady−state value of the associated integrator types to this value. If CONCENTRATION is :FIX, then set the steady−state value of integrator type to the current value of the concentration integrator. Returns steady−state concentration(s). To turn concentration integrators back on, use CONCENTRATION−CLAMP−OFF.

concentration-clamp-off *element* &optional *concentration* [*Function*]

Turn on all concentration integrators associated with ELEMENT. Otherwise identical to CONCENTRATION−CLAMP-

default-diffusion-coefficient *type* [*Function*]

The default diffusion coefficient for the ionic species specified for the concentration integrator type associated with TYPE, given by global variables such as *D_CA*, [cm2 sec−1]. If ion not associated with a diffusion coefficient global variable, returns 0.0.

conc-int-shell-1-free-conc-n *cint* [*Function*]

Concentration [mM] of free ion in shell 1 of CINT at time n.

conc-int-shell-1-free-conc-n+1 *cint* [*Function*]

Concentration [mM] of free ion in shell 1 of CINT at time n+1.

conc-int-shell-2-free-conc-n *cint* [*Function*]

Concentration [mM] of free ion in shell 2 of CINT at time n.

conc-int-shell-2-free-conc-n+1 *cint* [*Function*]

Concentration [mM] of free ion in shell 2 of CINT at time n+1.

conc-int-shell-3-free-conc-n *cint* [*Function*]

Concentration [mM] of free ion in shell 3 of CINT at time n.

conc-int-shell-3-free-conc-n+1 *cint* [*Function*]

Concentration [mM] of free ion in shell 3 of CINT at time n+1.

conc-int-core-free-conc *cint* [*Function*]

Concentration [mM] of free ion in core compartment of CINT.

conc-int-active-p *conc−int* [*Function*]

Only true when the CONC−INT will actually be evaluated − requires that not only the CONC−INT and it's type not be blocked, but also that the associated channel is active.

conc-int-membrane-current-component *cint* &optional *(shell 1)* [*Function*]

Returns the concentration derivative [mM/ms], for non :GENERIC integrators, or the current [nA] for :GENERIC integrators, for the compartment SHELL of CINT that is due to that compartment's associated :SHELL−PORES and pumps. The coefficient :BETA−CURRENT−sh, specific for a given CINT, converts channel current (nA) to d[x]/dt (mM/ms). :BETA−CURRENT−sh = 1 for :GENERIC integrators.

core-volume *cint* &optional *element−conc−volume element−area* [*Function*]

Returns the core volume of CINT in um3. Optional ELEMENT−CONC−VOLUME is also in um3.

conc-int-shell-membrane-area *cint shell* &optional *element−area* [*Function*]

Returns the surface area of one face of SHELL of CINT in um2. Optional ELEMENT−AREA is area of the associated cell element in um2.

interdigitation-area *cint* &optional *element−area* [*Function*]

Returns the diffusion membrane area between shells 1 and 2 of CINT in um2 for the :MULTI−SHELL class. Optional ELEMENT−AREA is area of the associated cell element in um2.

conc-int-diff-area *cint shell−x shell−y* &optional *element−area* [*Function*]

Returns the double−float diffusional area between SHELL−X and SHELL−Y of CINT in cm2. Optional ELEMENT−AREA is area of the associated cell element in um2.

22 SYS Source File: biophysics.lisp

nernst-potential *inside−conc outside−conc* &optional *(valence 1) (temperature−celcius *temp−celcius*)* [*Function*]

Returns potential (outside − inside) in mV. INSIDE−CONC and OUTSIDE−CONC are in mM. Default for optional VALENCE is 1, and for TEMPERATURE−CELCIUS, in degrees centigrade, is the value of the global variable *TEMP−CELCIUS*.

ghk-potential *inside−activities outside−activities permeabilities* &optional *(temperature *temp−celcius*)* [*Function*]

Returns potential (outside − inside) in mV. For monovalent species, the list of activities in INSIDE−ACTIVITIES and OUTSIDE−ACTIVITIES [mM] should be ordered according to specific species. The inside and outside activity of monovalent anions [e.g. Cl−] should be included in the OUTSIDE−ACTIVITIES and INSIDE−ACTIVITIES, respectively. The relative PERMEABILITIES of the ions should be in the same order as that for the other args. TEMPERATURE is in degrees celcius.

default-ion-reversal-potential *species* &optional *value* [*Function*]

Set the default (fixed) reversal potential for ion SPECIES ('NA, 'K, 'CL, 'CA, 'MG) if VALUE [mV] supplied. Returns the current value.

valence-from-k *k* &optional *(temperature *temperature*)* [*Function*]

Derive the valence of the gating particle given the slope of the Boltzmann fit (typically this is referred to as "K" in the literature). TEMPERATURE is in degrees kelvin.

boltzmann-equation *voltage v−half k* &optional *(power 1)* [*Function*]

Note that 1 divided by 4 times K is the slope of the Boltzmann expression at the midpoint.

constant-field-equation-exponential-term *voltage valence* [*Macro*]

Voltage is in V. All arguments are single−floats. This form is that used by Mcc−Hug−92.

constant-field-equation-exponential-term-double *voltage valence* [*Macro*]

Voltage (double−float) is in V. valence is single−float. This form is that used by Mcc−Hug−92.

constant-field-equation *voltage conc−in conc−out permeability valence* &optional *(gating−term 1.0)* [*Function*]

VOLTAGE is in mV, CONC−IN and CONC−OUT are in mM, PERMEABILITY in cm3/s, GATING−TERM and VALENCE are dimensionless. Returns current in nA. All arguments and returned value are single−floats. This form is that used by Mcc−Hug−92.

constant-field-equation-double *voltage conc−in conc−out permeability valence gating−term* [*Function*]

Double−float version of CONSTANT−FIELD−EQUATION (except for VALENCE, which is a single−float). Returns current in nA (double−float).

q10-tau-factor *reference−temp temp q10* &optional *(ignore−q10 *ignore−q10*)* [*Function*]

Returns the q10 factor for time constants (as temperature goes up, tau goes down).

q10-rate-factor *reference−temp temp q10* &optional *(ignore−q10 *ignore−q10*)* [*Function*]

This calculates the q10 factor for rate constants (as temperature goes up, so does rate).

q10-factor *reference−temp temp q10* &optional *(ignore−q10 *ignore−q10*)* [*Function*]

This calculates the q10 factor for rate constants (as temperature goes up, so does rate).

scaled-sigmoid-rate *voltage* &key *(scale 1.0) (steepness 1.0) (v−half 0.0) (base−rate 0.0)* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE and SCALE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope for VOLTAGE = V−HALF is [(0.25 * SCALE) / STEEPNESS].

scaled-exponential-rate *voltage* &key *(scale 1.0) (steepness 1.0) (v−half 0.0) (base−rate 0.0) (max−rate −1.0)* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE, SCALE, MAX−RATE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope for VOLTAGE = V−HALF is SCALE / STEEPNESS.

scaled-exponential-soft-rate *voltage* &key *(scale 1.0) (steepness 1.0) (v−half 0.0) (base−rate 0.0) max−rate* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE, SCALE, MAX−RATE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope for VOLTAGE = V−HALF is SCALE / STEEPNESS. If MAX−RATE is nil, then it is ignored.

squeezed-exponential *voltage* &key *(v−half 0.0) (k 1.0) (tau−max −1.0) (tau−min 0.0)* [*Function*]

Exponential rate function with minimum and maximum rates given by the reciprocal of TAU−MAX [default −1.0] and TAU−MIN [default 0.0], respectively. When TAU−MAX is NIL or non−positive, then the minimum rate is 0.0. V−HALF [default 0.0] and the inverse steepness K [default 1.0] are assumed to be in the same units as VOLTAGE, typically in mV. Returns single float value.

scaled-exponential-rate-double *voltage* &key *(scale 1.0d0) (steepness 1.0d0) (v−half 0.0d0) (base−rate 0.0d0) (max−rate −1.0d0)* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE, SCALE, MAX−RATE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope for VOLTAGE = V−HALF is SCALE / STEEPNESS.

scaled-exponential-rate-double *voltage* &key *(scale 1.0d0) (steepness 1.0d0) (v−half 0.0d0) (base−rate 0.0d0) (max−rate −1.0d0)* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE, SCALE, MAX−RATE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope for VOLTAGE = V−HALF is SCALE / STEEPNESS.

scaled-slanted-step-rate *voltage* &key *(scale 1.0) (steepness 1.0) (v−half 0.0) (base−rate 0.0)* [*Function*]

VOLTAGE, STEEPNESS and V−HALF are in mV. BASE−RATE and SCALE are in 1/ms. The minimum/maximum values are given by BASE−RATE and (BASE−RATE + SCALE), respectively. Note that the slope at VOLTAGE = V−HALF is [SCALE / STEEPNESS].

23 SYS Source File: sim.lisp

surf &optional *circuit (automatic *automatic−run*) (load−only *load−only*) (keep−track−of−time−for−auto−run nil)* [*Function*]

The main simulation function of Surf−Hippo − launches the GUI loop. If there is an optional CIRCUIT, then it is loaded first (even if the current circuit is the same). See also GOFERIT.

goferit &optional *(stop−time *user−stop−time*)* [*Function*]

Simulate the loaded circuit for STOP−TIME [default *USER−STOP−TIME*] milliseconds.

gotimed &optional *(stop−time *user−stop−time*)* [*Function*]

As GOFERIT, but keeping track of the simulation run time.

goquiet &optional *(stop−time *user−stop−time*)* [*Function*]

As GOFERIT, but suppressing text output to Lisp window.

set-celcius-temperature *new−value* [*Function*]

Set the temperature of the simulation to NEW−VALUE degrees celcius, and propagate this value to all the temperature−dependent elements in the circuit. Returns the new value of *TEMP−CELCIUS* as a single−float.

queue-breakpoint-time *time* [*Function*]

For variable step integration, puts TIME [milliseconds] on the queue of break points so that the simulation can be sure to step there.

24 SYS Source File: circuit-input.lisp

read-in-circuit &optional *circuit* &key *nts−cell−name (origin−offset '(0.0 0.0 0.0))* [*Function*]

If CIRCUIT is supplied, then it is loaded. Otherwise, the current state of *CIRCUIT−SOURCE* is referenced and, as appropriate, *CIRCUIT−FUNCTION* or *CIRCUIT−FILENAME* is loaded. NTS−CELL−NAME is valid when CIRCUIT names an ntscable created file − if not supplied, cell name is taken from the specification in the file. ORIGIN−OFFSET applies to all cells in the circuit created by the current evaluation of READ−IN−CIRCUIT.

toplоad &body *body* [*Macro*]

Load circuit definition expressed by BODY. If not called recursively, first clears all circuits when *INITIALIZE−BEFORE−NEXT−CIRCUIT* is T. BODY may be a (function) symbol, filename, or a series of Lisp forms. Unless *INITIALIZE−BEFORE−NEXT−CIRCUIT* is T, any loaded circuit is cleared by calling INITIALIZE−GLOBALS−FOR−CIRCUIT, even if BODY is not included or is NIL.

25 SYS Source File: node.lisp

print-node-states [*Function*]

Prints the vector of node voltages and delta−v's. Mainly for debugging.

print-node-dv-states [*Function*]

Prints the vector of node voltages and delta−v's. Mainly for debugging.

core-off-diag *point−diag lower* [*Structure*]

declare-ground *gnd−name* [*Function*]

Creates the ground node.

reorder-circuit [*Function*]

Collects nodes to be evaluated in to *CORE−NODE−ARRAY*, ordered by the segment *BRANCH−LIST* according to the Hines ordering. Contructs the 3 (tridiagonal) matrix arrays (upper, lower, diagonal) and the right hand side and output array.

set-*node-voltage-initializations* [*Function*]

Set *NODE−VOLTAGE−INITIALIZATIONS* to a list with all the circuit nodes and their voltages (node−voltage−n+1).

element-holding-potential *element* &optional *(value nil value−supplied−p)* [*Function*]

If VALUE (in mV) is supplied, then sets the 'HOLDING−POTENTIAL parameter of the circuit node(s) associated with ELEMENT and returns VALUE (converted to double−float). If VALUE is supplied, and NIL, then the 'HOLDING−POTENTIAL is cleared for the node. Otherwise, returns the current value of the 'HOLDING−POTENTIAL parameter for the node, if that value has been set previously.

element-constant-current *element* [*Function*]

Returns constant current term [nA] if it exists to the node associated with ELEMENT, otherwise nil.

add-constant-current *element current* [*Function*]

Adds a constant CURRENT [nA] to the cell elements associated with ELEMENT. This is equivalent to including a current source at the element with a fixed DC value.

clear-constant-currents [*Function*]

Removes any constant current terms from the circuit nodes

element-resting-potential *element* [*Function*]

Return the membrane resistance leak potential of the cell element associated with ELEMENT.

26 SYS Source File: soma.lisp

soma-membrane-resistivity *soma* [*Function*]

The membrane resistivity of SOMA, in ohms−cm2.

soma-specific-capacitance *soma* [*Function*]

The membrane specific capacitance of SOMA, in uF/cm2.

soma-v-leak *soma* [*Function*]

The reversal potential of the leak resistance of SOMA, in mV.

create-soma &key *cell cell−type name (location '(0.0 0.0 0.0)) length soma−cylinder−diameter (diameter *default−soma−diameter*) parameters adjust−area−for−trunks shunt (enable−automatic−cell−names *enable−automatic−cell−names*) (automatic−name−fixing *prompt−for−alternate−element−names*)* [*Function*]

DIAMETER is in microns. CELL refers to either a cell structure or the name of one − if not supplied, a new cell is created of CELL−TYPE. SHUNT [ohms, default NIL], when non−NIL, is a non−specific somatic shunt. LOCATION gives the xyz coordinates of the SOMA in microns. When ADJUST−AREA−FOR−TRUNKS is T [default nil], then the soma area [as returned by the ELEMENT−AREA and ELEMENT−AREA−CM2 functions] is adjusted for the areas of the faces of any abutting segments.

set-soma-absolute-parameters *soma capacitance g−leak* [*Function*]

Set linear membrane properties of SOMA to the absolute values of CAPACITANCE [nF] and G−LEAK [uS]. Sets :INHERIT−PARAMETERS−FROM−TYPE of SOMA to NIL. If any of the soma parameter arguments are NIL, then the original value is retained.

set-soma-parameter *soma parameter value* [*Function*]

Set a PARAMETER distinct from the associated cell type for somas associated with SOMA, for example 'RM, 'CM, or 'V−LEAK. Sets :INHERIT−PARAMETERS−FROM−TYPE for SOMA to NIL.

soma-voltage &optional *(soma *soma*)* [*Function*]

Retrun the voltage of optional SOMA [default *soma*].

27 SYS Source File: segment.lisp

segment-rm *segment* [*Function*]

The membrane resistivity of SEGMENT, in ohms−cm2.

segment-cm *segment* [*Function*]

The membrane specific capacitance of SEGMENT, in uF/cm2.

segment-ri *segment* [*Function*]

The cytoplasmic resistivity of SEGMENT, in ohms−cm.

segment-v-leak *segment* [*Function*]

The reversal potential of the leak resistance of SEGMENT, in mV.

segment-ri-coefficient *segment* [*Function*]

The dimensionless coefficient for the cytoplasmic resistivity of SEGMENT.

create-segment *name proximal−element* &optional *cell* &key *(diameter 0.0) (length 0.0) (theta 0.0) (phi (* −0.5 pi−single)) (relative−location '(0.0 0.0 0.0)) relative−location−is−float absolute−location absolute−location−is−float dummy−proximal−element−location dummy−proximal−element−location−is−float (ri−coefficient 1.0) parameter−a−list* [*Function*]

Returns a segment with NAME that is attached to the soma, segment or node PROXIMAL−ELEMENT. DIAMETER and LENGTH are in microns. The location of the distal node relative to the soma of CELL [if not given, derived from PROXIMAL−ELEMENT] is given by the XYZ values [microns] in the list RELATIVE−LOCATION. Alternatively, if PROXIMAL−ELEMENT is a segment, the location can be defined relative to the orientation of the proximal segment by THETA and PHI, each in radians.

create-segment-fast *name proximal−element* &optional *cell* &key *(diameter 0.0) (length 0.0) (theta 0.0) (phi (* −0.5 pi−single)) (relative−location '(0.0 0.0 0.0)) absolute−location dummy−proximal−element−location parameter−a−list (ri−coefficient 1.0)* [*Function*]

An optimized version of CREATE−SEGMENT. All numeric arguments are assumed to be single floats, except if NAME is a number, in which case it must be an integer.

set-segments-inherit-parameters-from-type &optional *cell* [*Function*]

Makes all segments in CELL (if supplied) or in the circuit (else) inherit their properties from the associated CELL−TYPE.

set-segment-absolute-parameters *seg capacitance g−axial g−leak* [*Function*]

Set cable properties of SEG to the absolute values of CAPACITANCE [nF], G−AXIAL and G−LEAK [uS]. Sets :INHERIT−PARAMETERS−FROM−TYPE of SEG to NIL. If any of the segment parameter arguments are NIL, then the original value is retained.

set-segment-parameter *seg parameter value* [*Function*]

Set a PARAMETER distinct from the associated cell type for segments associated with SEG, for example including 'RM, 'CM, 'V−LEAK, 'RI, or 'RI−COEFFICIENT. Sets :INHERIT−PARAMETERS−FROM−TYP for SEG to NIL.

rename-segments-simple &optional *segments* [*Function*]

Rename SEGMENTS [default all segments in circuit] with simple integer names.

distal-tip-p *element* [*Function*]

Predicate for whether cell element associated with ELEMENT is located on a distal tip of the dendritic tree.

distal-segments *element* &optional *include−electrodes* [*Function*]

Returns a list of all the segments directly attached to the distal node of segment associated with ELEMENT, or the trunk segments if ELEMENT is associated with the soma. Electrode segments are not included, unless INCLUDE−ELECTRODES is T [default NIL].

segs-until-bifurcation *seg* [*Function*]

Given a segment, returns all distal segments, including this one, before the next branch point in the tree.

proximal-cell-element *elt* [*Function*]

Returns the proximal cell element (segment or soma) associated with the cell element of ELT. If ELT is on the soma, then the soma is returned.

proximal-segment *elt* [*Function*]

Returns the proximal segment associated with the cell element of ELT, if there is one.

attached-to-soma-p *element* [*Function*]

True if ELEMENT is either a soma, a membrane element of a soma, or a trunk segment.

28 SYS Source File: source.lisp

sources &optional *cell* [*Function*]

Return a list of all current and voltage sources.

add-isource &optional *(element *soma*)* &key *name pulse−list (type 'autonomous)* [*Function*]

Adds current source to cell elements associated with optional ELEMENT [default the value of *SOMA*]. Source is called NAME, if supplied, or is given by EltName−isrc, where EltName is the name of cell−element. Creates new source only if one does not exist of the derived name. Optional PULSE−LIST may also be supplied. Returns the source(s). If ELEMENT refers to a cell, then an isource is added to that cell's soma. The default 'AUTONOMOUS type is the generic :AUTONOMOUS current source driven by a waveform or pulse specification (here given by PULSE−LIST − see the function PULSE−LIST for the format of this list).

add-vsource &optional *(element *soma*)* &key *name (ideal t) pulse−list default−magnitude* [*Function*]

Adds voltage source to cell elements associated with the optional ELEMENT [default the value of *SOMA*], if no voltage source is already there. Source is called NAME, if supplied, or is given by EltName−vsrc, where EltName is the name of cell−element. If IDEAL then ideal voltage sources are created. An optional PULSE−LIST may also be supplied. Returns the source(s). When DEFAULT−MAGNITUDE is a number [mV, default NIL], this value is used as the default magnitude for the source. If ELEMENT refers to a cell, then a vsource is added to that cell's soma.

cell-isource *element* [*Function*]

Returns a current source, from the soma preferably, associated with the cell associated with ELEMENT, if there are any such sources.

cell-vsource *element* [*Function*]

Returns a voltage source, from the soma preferably, associated with the cell associated with ELEMENT, if there are any such sources.

edit-source *source* [*Function*]

For editing all SOURCE parameters.

pulse-list *source* &optional *(pulse−list nil pulse−list−supplied)* [*Function*]

For adding a PULSE−LIST to SOURCE, where the format of PULSE−LIST is either:

```
(pulse-1 pulse-2 ...)
```

or for just a single pulse:

```
pulse
```

The format of each pulse is as follows:

```
(start-time stop-time amplitude)
```

The time parameters are in milliseconds, and amplitude is either nA or mV for current and voltage sources, respectively. For example a pulse defined with '(4 6 .1) applied to a current source defines a 0.1nA pulse from 4 to 6 milliseconds. This function will also set the :USE−PULSE−LIST slot for the source. If called with only the SOURCE arg, the pulse−list currently assigned to the source will be returned. If there is an explicit NIL PULSE−LIST arg any pulse−list assigned to SOURCE will be cleared. Pulse must be separated by a minimum time which is a function of the transition speed, and thus they cannot overlap in time. Typically the minimum separation is on the order of 0.005 milliseconds. The SOURCE argument is initially processed by ELEMENT.

pulse-train *source* &optional *(start nil start−supplied) stop delay duration period amplitude* [*Function*]

When START, STOP, DELAY, DURATION, PERIOD, AMPLITUDE are numbers, assign the corresonding pulse train specification to SOURCE and enable the pulse train. All time args in milliseconds. AMPLITUDE is in nA or mV depending on whether SOURCE refers to a current or voltage source, respectively. If called with only the SOURCE arg, the list of pulse train specs [same order as PULSE−TRAIN args] currently assigned to the source will be returned. If there is an explicit NIL START arg any pulse train assigned to SOURCE will be cleared, and pulse train will be disabled for this SOURCE.

enable-pulse-train *source* [*Function*]

Enables pulse train generation by SOURCE.

disable-pulse-train *source* [*Function*]

Disables pulse train generation by SOURCE.

enable-individual-pulses *source* [*Function*]

Enables individual pulse generation by SOURCE.

disable-individual-pulses *source* [*Function*]

Disables individual pulse generation by SOURCE.

add-waveform-to-element-control-waveform *element−reference added−waveform* &key *element−reference−timestep added−waveform−timestep* [*Function*]

Add successive values of the numerical sequence ADDED−WAVEFORM to the successive values of ELEMENT−REFERENCE, if ELEMENT−REFERENCE is a sequence, else the sequence returned by applying ELEMENT−CONTROL−WAVEFORM to ELEMENT−REFERENCE. If the length of ADDED−WAVEFORM is less than the sequence associated with ELEMENT−REFERENCE, then the subsequent added values are taken to be 0. The returned sequence is the same type (array or list) and length as that of the sequence associated with ELEMENT−REFERENCE. If ADDED−WAVEFORM−TIMESTEP is supplied, and is inconsistent with the timestep of the sequence associated with ELEMENT−REFERENCE, then the ADDED−WAVEFORM is resampled as appropriate. The latter timestep is either given explicitly by ELEMENT−REFERENCE−TIMESTEP or determined by the function ELEMENT−CONTROL−WAVEFORM−TIMESTEP

add-waveform *destination* &key *waveform−spec (waveform−time−interval *default−waveform−step*) delay use−menu float−input* [*Function*]

Add a waveform to DESTINATION, which can refer to either current or voltage sources, synapse or synapse types. WAVEFORM−SPEC is either a sequence of numbers or a function specification (lambda list) which returns a number sequence. If not included, or if WAVEFORM−SPEC is a function spec and USE−MENU is T, the function WAVEFORM−MENU is called. WAVEFORM−TIME−INTERVAL [ms, default *DEFAULT−WAVEFORM−STEP*] is the time base for WAVEFORM−SPEC. DELAY, when not NIL [default] is in milliseconds, and sets the destination :DELAY slot of current or voltage sources, or adds a delay to the actual waveform used in synapse types. DESTINATION may also be an electrode, in which case the actual destination is extracted with the function ELECTRODE−SOURCE. If FLOAT−INPUT is T, then the waveform associated with WAVEFORM−SPEC is a single float numeric sequence.

29 SYS Source File: isource.lisp

rename-isources-simple &optional *(isources (isources))* [*Function*]

Rename ISOURCES [default all isources in circuit] with simple integer names.

30 SYS Source File: vsource.lisp

rename-vsources-simple &optional *(vsources (vsources))* [*Function*]

Rename VSOURCES [default all vsources in circuit] with simple integer names.

ideal-vsource *vsource* [*Function*]

Make VSOURCE ideal.

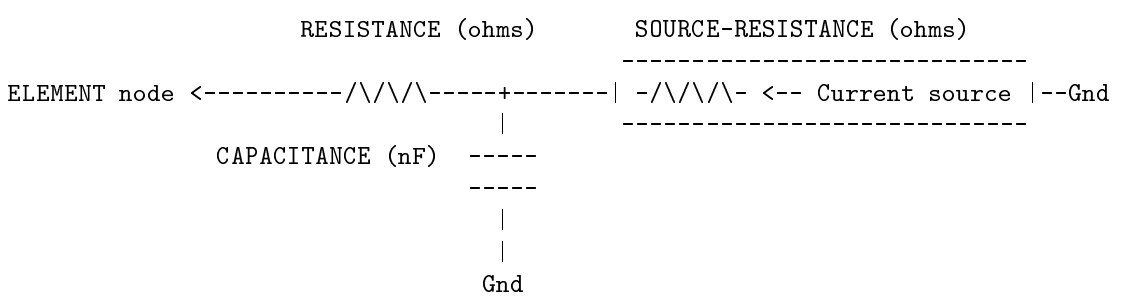
non-ideal-vsource *vsource* &optional *resistance* [*Function*]

Make VSOURCE non−ideal. When RESISTANCE is included, set the internal resistance of VSOURCE to this value [in Mohms]. If successful, returns the internal resistance of VSOURCE.

31 SYS Source File: electrode.lisp

add-ielectrode *element* &key *(capacitance 0.001) (resistance 1.0e+7) (source−resistance 0.0) (leak−resistance 1.0d+16) name* [*Function*]

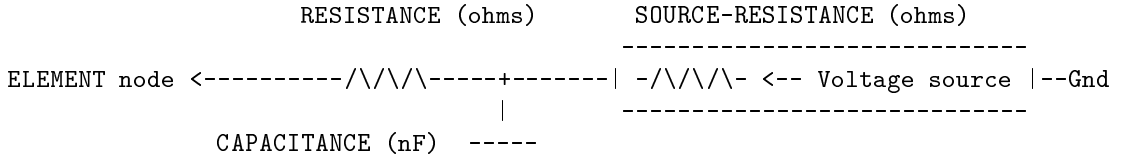
Adds an electrode model with current source to the cell element associated with ELEMENT −



Default values are CAPACITANCE 1.0e−3 (nF), RESISTANCE 10e6 (ohms), SOURCE−RESISTANCE 0 (ohms). A LEAK−RESISTANCE (ohms, default 10e16) is also included in parallel to the electrode capacitance. RESISTANCE must be greater than 0.

add-velectrode *element* &key *(capacitance 0.001) (resistance 1.0e+7) (source−resistance 0.0) (leak−resistance 1.0d+16) name* [*Function*]

Adds an electrode model with voltage source to the cell element associated with ELEMENT −



Default values are CAPACITANCE 1.0e−3 (nF), RESISTANCE 10e6 (ohms), SOURCE−RESISTANCE 0 (ohms). A LEAK−RESISTANCE (ohms, default 10e16) is also included in parallel to the electrode capacitance. RESISTANCE must be greater than 0.

electrodes &optional *(element nil element−supplied−p)* [*Function*]

Returns a list of all electrodes associated with the cell elements referenced by ELEMENT, if supplied; otherwise, all electrodes in circuit.

electrode-source *electrode* [*Function*]

Returns any source(s) associated with ELECTRODE.

electrode-resistance *electrode* [*Function*]

Return the axial resistance of ELECTRODE in Mohms.

electrode-leak-resistance *electrode* [*Function*]

Return the axial leak−resistance of ELECTRODE in Mohms.

electrode-capacitance *electrode* [*Function*]

Return the capacitance of ELECTRODE in pF.

set-electrode-capacitance *c−electrode* &optional *(electrode *electrode*)* [*Function*]

Sets the capacitance of ELECTRODE to C−ELECTRODE (nF).

set-electrode-resistance *r−electrode* &optional *(electrode *electrode*)* [*Function*]

Sets the resistance of ELECTRODE to R−ELECTRODE (Mohms − must be greater than 0).

32 SYS Source File: general-membrane-elements.lisp

iv-type-parameter *element param* &optional *(value nil value−supplied−p) (update t)* [*Function*]

For examining/setting specific structure parameters of the synapse or channel type associated with ELEMENT. PARAM can be:

```
:IV-SOURCE (e.g. :ABSOLUTE or :DENSITY)
:IV-REFERENCE [for :ABSOLUTE gbar (uS) or permeability (cm3/sec)]
:IV-DENSITY [pS/um2 (0.1mS per square cm) for gbar, 1.0e-6 cm3/sec/um2 for permeability]
:IV-MODULATION [applied to all type children, regardless of inheritance]
:E-REV [mV]
:BLOCKED [T or NIL]
```

If no new VALUE follows the PARAM, then the current value of the slot corresponding to PARAM is returned. Supplying a non−nil value for UPDATE will cause the change to propagate to the appropriate elements of the type associated with ELEMENT.

pore-blocked-p *pore* [*Function*]

Predicate for the block of a channel or synapse PORE, either due to an individual block or block of the associated type. If PORE is a channel or synapse type, then return T if that type is blocked.

document-element-type-def-decimals *2* [*Variable*]

For automatic element TYPE−DEF form generation by the DOCUMENT−ELEMENT function, this is the number of decimal values used for numeric parameters.

effective-reversal-potential *ion−perms* &optional *element* [*Function*]

Calculate reversal potential based on the list ION−PERMS, which has the format:

```
'((ion permeability) (ion permeability) ...)
```

where ION is one of the symbols used by the function DEFAULT−ION−REVERSAL−POTENTIAL, and PERMEABILITY is the [single float] relative permeability. The reversal potentials for each ION references the cell−type associated with ELEMENT, or the DEFAULT−ION−REVERAL−POTENTIAL.

33 SYS Source File: channel.lisp

channel-active-p *channel* &optional *fast* [*Function*]

T when a CHANNEL satisfies the condition for being ACTIVE: the channel or type is not blocked, and the channel conductance is not 0.0. If CHANNEL is a pointer to a channel, then the FAST flag may be used.

nb-active-chs-of-type *type* [*Function*]

Returns the number of channels of the associated with TYPE and satisfying the conditions for being active.

channel-particle-power *particle* &optional *channel−type* [*Function*]

Returns the number of phenomenological particles of type associated with PARTICLE for the channel type associated with CHANNEL−TYPE, if supplied, or with PARTICLE, if PARTICLE refers to a specific particle instance.

create-channel-type *type−symbol* &optional *actual−type−symbol update−parameters* [*Function*]

TYPE−SYMBOL is a symbol or a channel type; in the former case it must match the CAR of one of the lists contained in channel type model parameter library. Returns the channel type structure, whether is was already made or not. If the type was already made, and UPDATE−PARAMETERS is T, the type will be updated according to the current description loaded in parameter library. This will create particle types (v−dep and conc) according the entries in the V−PARTICLES and CONC−PARTICLES a−list entries.

rename-channels-simple &optional *(channels (channels))* [*Function*]

Rename CHANNELS [default all channels in circuit] with simple integer names.

create-channel *element type* &key *pre−synaptic−element conc−int−delta* [*Function*]

Create a channel of TYPE on the cell element associated with ELEMENT. CONC−INT−DELTA applies to channels associated with concentration integrators, and specifies the fraction between 0 and 1 of the total channel current which will source the appropriate integrator. When supplied, PRE−SYNAPTIC−ELEMENT specifies the cell element which controls the channel, otherwise taken as the cell element associated with ELEMENT.

channels-of-type *type* &optional *cell−element* [*Function*]

Return a list of channels of TYPE that are associated with members of CELL−ELEMENT [atom or list]. Members of CELL−ELEMENT may refer explicitly to a cell type or specific cell, or may be associated with a cell element. If CELL−ELEMENT is NIL, then all channels of TYPE are returned.

34 SYS Source File: particle.lisp

rename-particles-simple &optional *(particles (particles))* [*Function*]

Rename PARTICLES [default all particles in circuit] with simple integer names.

set-particle-type-tau *type tau−form* [*Function*]

Set the :TAU−FUNCTION slot of particle TYPE to TAU−FORM, and update the appropriate arrays. TAU−FORM can be either a number, representing a fixed time constant in milliseconds, or a function name or lambda form, either with a single voltage argument in millivolts, that return a tau value in milliseconds.

set-particle-type-ss *type ss−form* [*Function*]

Set the :SS−FUNCTION slot of particle TYPE to SS−FORM, and update the appropriate arrays. SS−FORM can be either a number, representing a fixed steady−state value between 0 and 1, or a function name or lambda form, either with a single voltage argument in millivolts, that return a ss value between 0 and 1.

v-function-array *function−or−form* &key *(min−voltage* [*Function*]
**particle−look−up−table−min−voltage*) (voltage−increment*
**particle−look−up−table−precision*) (array−length*
**particle−look−up−table−length*)*

Return a double−float array whose values are derived from evaluating FUNCTION−OR−FORM on a numeric sequence of ARRAY−LENGTH values starting with MIN−VOLTAGE and incrementing by VOLTAGE−INCREMENT. FUNCTION−OR−FORM may be either a function with a single argument, a list whose CAR is a function, whose CADR is a dummy variable that will be taken from the numeric sequence and the remainder corresponding to other function arguments, or it can be a constant, which will be used to filled the returned array. Default values of all arguments for V−FUNCTION−ARRAY reflect its usage in generating voltage−dependent particle type lookup tables.

35 SYS Source File: markov-particle.lisp

return-markov-rate *val* [*Macro*]

This macro wraps around the code for any state transition function with the particle as its arg, where VAL is the double−float result of the function. The purpose of this macro is to pass the double−float rate to the markov particle evaluation code via the global *MARKOV−RATE−ARRAY*.

36 SYS Source File: conc-part.lisp

rename-conc-particles-simple &optional *(conc−particles (conc−particles))* [*Function*]

Rename CONC−PARTICLES [default all conc−particles in circuit] with simple integer names.

conc-particle-concentration-arg *concentration type* [*Function*]

All concentration dependent particles [conc−particles] are evaluated with the actual CONCENTRATION [mM] passed through this function. The returned value is given by

```
[CONCENTRATION-COEFFICIENT * Th(CONCENTRATION - BASE-CONCENTRATION)]^CONC-POWER
```

where CONCENTRATION−COEFFICIENT, BASE−CONCENTRATION, and CONC−POWER are given by the corresponding slots of the concentration particle TYPE. 'Th' is the rectifying function.

37 SYS Source File: synapse.lisp

synapse-active-p *synapse* &optional *fast* [*Function*]

T when a SYNAPSE satisfies the condition for being ACTIVE: the synapse or type is not blocked, the synapse conductance is not 0.0, and for a light synapse, it is within the aperture. If SYNAPSE is a pointer to a synapse, then the FAST flag may be used.

nb-active-chs-of-type *type* [*Function*]

Returns the number of synapses of the associated with TYPE and satisfying the conditions for being active.

active-synapses *type* [*Function*]

List of all active synapses of the given TYPE.

create-synapse-type *type−symbol* &optional *actual−type−symbol update−parameters* [*Function*]

TYPE−SYMBOL is a symbol or synapse type; in the former case it must match the CAR of one of the lists contained in synapse type model parameter library. Returns the synapse type structure, whether is was already made or not. If the type was already made, and UPDATE−PARAMETERS is T, the type will be updated according to the current description loaded in parameter library.

rename-synapses-simple &optional *(synapses (synapses))* [*Function*]

Rename SYNAPSES [default all synapses in circuit] with simple integer names.

create-synapse *post−synaptic−element type* &optional *pre−synaptic−element (add−pre−synaptic−element−name−to−name t) name* [*Function*]

Returns a synapse of TYPE, installed on the cell element associated with POST−SYNAPTIC−ELEMENT. Synapse types that are controlled by the voltage of another node must include a PRE−SYNAPTIC−ELEMENT (element or name associated with a soma, segment, or axon). If the POST−SYNAPTIC−ELEMENT (can be NIL) already has a synapse of the same type, and the PRE−SYNAPTIC−ELEMENT is either different or not required for TYPE, then an alternate name will be created from the addition of a number at the end of the standard synapse name. If the global variable *PROMPT−FOR−ALTERNATE−ELEMENT−NAMES* is T, then the user is prompted before the additional synapse is created, otherwise, the synapse is created. The standard synapse name is either an integer, if *USE−SIMPLE−NAMES* is T (generated by GET−SYNAPSE−SIMPLE−NAME), or given by NAME, if non−NIL, or composed from the TYPE and the post−synaptic element, including also the name of the PRE−SYNAPTIC−ELEMENT if there is one and if ADD−PRE−SYNAPTIC−ELEMENT−NAME−TO−NAME is T.

driven-synapses *element* [*Function*]

Return a list of synapses whose pre−synaptic cell element is that associated directly with ELEMENT.

synaptic-targets *element* [*Function*]

Return a list of cell elements who are post−synaptic to the cell element associated directly with ELEMENT.

impinging-synapses *element* [*Function*]

Return a list of synapses whose post−synaptic cell element is that associated directly with ELEMENT.

synapses-of-type *type* &optional *cell−element* [*Function*]

Return a list of synapses of TYPE that are associated with members of CELL−ELEMENT [atom or list]. Members of CELL−ELEMENT may refer explicitly to a cell type or specific cell, or may be associated with a cell element. If CELL−ELEMENT is NIL, then all synapses of TYPE are returned.

nb-event-syn-bps-w-ideal-vsource *10* [*Variable*]

If even synapse node has an ideal voltage source, add this number of breakpoints over the duration of the synapse conductance waveform to try to sample properly. Must be a fixnum.

queue-event-synapse-breakpoint-time *syn conductance−waveform−event−breakpoints* [*Function*]

Queue not only the event start times, but also the midpoint of the associated event waveform and any user−specified breakpoints, in order to better catch event−driven changes in the circuit. If node has an ideal voltage source, add *NB−EVENT−SYN−BPS−W−IDEAL−VSOURCE* points over the duration of the synapse conductance waveform to try to sample properly. EVENT−SYNAPSE−CONDUC must be a single−float.

count-active-and-triggered-synapses *t* [*Variable*]

When non−NIL, the function COUNT−ACTIVE−SYNAPSES, which normally prints out info at the end of each simulation, also prints the number of synapses actually fired.

38 SYS Source File: light-synapse-functions.lisp

light-controlled-p *element* [*Function*]

Returns T if ELEMENT is a LIGHT or LIGHT−EVENT controlled synapse or synapse type.

39 SYS Source File: synapse-events.lisp

clear-events &optional *syns−or−types* [*Function*]

Clear all :EVENT−TIMES slots in the synapses associated with the atom or list SYNS−OR−TYPES. If SYNS−OR−TYPES is not supplied, then this is done for all synapses.

events &optional *(syns−or−types 'synapse) (event−times nil event−times−supplied−p)* &rest *more−events* [*Function*]

Return the event times associated with the synapses associated with SYNS−OR−TYPES (atom or list). If EVENT−TIMES (arbitrary number of lists of numbers or single numbers, in milliseconds) is included, these times are added to the existing events. If EVENT−TIMES is set explictly to NIL, then all events are cleared.

remove-events *syns−or−types event−times* [*Function*]

Remove EVENT−TIMES (list of numbers or single number) from existing events for all synapses associated with SYNS−OR−TYPES (atom or list).

add-poisson-events *syns−or−types lambda−spec start stop* &key *(step 1.0) (time−offset 0.0) (lambda−coefficient 1.0) (min−interval−value 1.0) clear−events* [*Function*]

Adds events derived from a poisson process from START to STOP [ms] to synapses associated with SYNS−OR−TYPES (atom or list). Poisson processes are generated with MODULATED−POISSON−EVENTS: if LAMBDA−SPEC is a number, then this is the mean lambda in 1/ms; if it is a function or a sequence, then LAMBDA−SPEC determines the lambda as a function of time. In the latter case, the value of STEP [ms] is passed to MODULATED−POISSON−EVENTS. LAMBDA−COEFFICIENT is applied to the lambda value at all times. If an element of SYNS−OR−TYPES refers specifically to a synapse type, then events are added to all synapses of that type. All times in the returned list are adjusted by the addition of TIME−OFFSET[ms]. Poisson intervals have a minimum value of MIN−INTERVAL−VALUE [ms]. When CLEAR−EVENTS is true, then the existing :EVENT−TIMES are cleared before adding the new poisson events to each synapse.

histogram-synapse-events &key *(bins 10) (min−time 0) type syn (max−time *user−stop−time*)* [*Function*]

Plot a histogram of the event times assigned to either synapses associated with SYN or TYPE − if neither is supplied then plot events of all EVENT synapses. Individual plots are generated for all types of referenced synapses.

plot-scatter-synapse-distances-event-times &key *(synapse−types (synapse−types))* [*Function*]
(white−is−maximum−p t) (x−are−fns t)
(y−are−fns t) (width 400) (height 400)
plot−type min−plot−time−given−by−events
(plot−events−prior−to−0 t)
title (3dplot−scale 30.0) (3dplot−aspect 0.2)
*(3dplot−time−bins (/ *user−stop−time**
100)) (3dplot−distance−bins 20)
(minimum−distance 0) maximum−distance
*(maximum−time *user−stop−time*)*
*(minimum−time 0.0) (cell *cell*)*
distance−plot−increment

Plot event times for all enabled event synapses in CELL of types given by the atom or list SYNAPSE−TYPES versus distances to soma of the synapses.

40 SYS Source File: buffer.lisp

rename-buffers-simple &optional *(buffers (buffers))* [*Function*]

Rename BUFFERS [default all buffers in circuit] with simple integer names.

41 SYS Source File: pump.lisp

rename-pumps-simple &optional *(pumps (pumps))* [*Function*]

Rename PUMPS [default all pumps in circuit] with simple integer names.

42 SYS Source File: axon.lisp

rename-axons-simple &optional *(axons (axons))* [*Function*]

Rename AXONS [default all axons in circuit] with simple integer names.

create-axon *proximal−element axon−type−symbol* &key *(length 0.0) (delay 0.0) target* [*Function*]
(distance−coeff 1.0) (consider−mid−points−for−length t) mid−points

An axon of AXON−TYPE−SYMBOL is created, controlled by the voltage of the cell element associated with PROXIMAL−ELEMENT. Axons do have nodes, so that accessing their voltage is the same as for segments and somas, but these nodes are not considered part of the core circuit [the :IS−PHYSICAL−CELL−NODE slot is NIL for these nodes] since they are not part of the circuit equations. If included, MID−POINTS is a list of XYZ coordinates that define a series of points on the axon, that is considered in the graphical representation of the axon, and as well for the actual length if CONSIDER−MID−POINTS−FOR−LENGTH is T unless LENGTH is non−NIL. Otherwise, the axon length is derived from the straight line distance between PROXIMAL−ELEMENT and the TARGET, multiplied by DISTANCE−COEFF. All distance arguments are in microns. TARGET is only used for calculating the axon length. Any synapse that is controlled by the created axon must be created after the axon. In this case the length of the axon will be re−evaluated using the location of the target synapse as a reference.

create-axon-type *type−symbol* &optional *actual−type−symbol update−parameters* [*Function*]

TYPE−SYMBOL is a symbol or an axon type; in the former case it must match the CAR of one of the lists contained in axon type parameter library. Returns the axon type structure, whether is was already made or not.

43 SYS Source File: event-generators.lisp

event-generator *event−element* &optional *only−slot−value* [*Function*]

Returns the value of the :EVENT−GENERATOR slot of EVENT−ELEMENT. If NIL, and ONLY−SLOT−VALUE is NIL, then returns EVENT−ELEMENT.

user-setup-event-generators-and-followers *event−generator event−followers* [*Function*]

Sets the :EVENT−GENERATOR slot of EVENT−GENERATOR and the EVENT−FOLLOWERS to EVENT−GENERATOR, and assigns the list of EVENT−FOLLOWERS to EVENT−GENERATOR. Ensures that EVENT−GENERATOR is also a member of the actual event−followers.

44 SYS Source File: cell.lisp

cell-type-parameter *element param* &optional *(value nil value−supplied−p) (update t)* [*Function*]

For examining/setting specific structure parameters of the cell type associated with ELEMENT. PARAM can be:

```
:RI [ohms-cm]
```

These set both the soma and the dendritic values −

```
:RM [ohms-cm2]
:V-LEAK [mV]
:CM [uF/cm2]
```

These set only the dendritic values−

```
:RM-DENDRITE [ohms-cm2]
:V-LEAK-DENDRITE [mV]
:CM-DENDRITE [uF/cm2]
```

These set only the somatic values−

```
:RM-SOMA [ohms-cm2]
:V-LEAK-SOMA [mV]
:CM-SOMA [uF/cm2]
:SOMA-SHUNT [ohms]
```

If :CM is specified, this value is assigned to both the somatic and dendritic slots. Likewise, if :V−LEAK is specified, then this value is assigned to both the :V−LEAK−SOMA and :V−LEAK−DENDRITE slots. Note that the cell type parameters will not be propagated to the segments and soma until SET−CIRCUIT−ELEMENTS−PARAMETERS is called. When UPDATE is non−NIL then propagate parameter values to type cells. If no new VALUE follows the PARAM, then the current value of the slot corresponding to PARAM is returned.

create-cell-type &optional *type−symbol actual−type−symbol update−parameters* [*Function*]

TYPE−SYMBOL is a symbol or a cell type; in the former case it must match the CAR of one of the lists contained in cell type model parameter library. Returns the cell type structure, whether is was already made or not. If the type was already made, and UPDATE−PARAMETERS is T, the type will be updated according to the current description loaded in parameter library. If TYPE−SYMBOL does not correspond to an entry in the parameter library, then the cell type parameters will be taken from various global variables, including *RM*, *RI*, *CM*, *CM−DENDRITE*, *SOMA−SHUNT*, *V−LEAK*, *V−LEAK−DENDRITE*, *E−NA*, *E−K*, *E−CA*, *E−CL*, in addition to default specifications for reversal potentials (:FIXED) and concentrations (:FOLLOWS−GLOBAL). The TYPE−SYMBOL that is actually used for the type is an uppercase symbol.

create-cell-type-w-params &optional *name* &key *(rm *rm*) (ri *ri*) rm−soma (soma−shunt *soma−shunt*) cm−soma cm−dendrite (cm *cm−dendrite*) spcap (v−leak *v−leak−dendrite*) v−leak−dendrite v−leak−soma (e−na *e−na*) (e−k *e−k*) (e−ca *e−ca*) (e−cl *e−cl*) (e−na−dependence :fixed) (na−conc−extra−dependence :follows−global) (e−k−dependence :fixed) (k−conc−extra−dependence :follows−global) (e−ca−dependence :fixed) (ca−conc−extra−dependence :follows−global) (e−cl−dependence :fixed) (cl−conc−extra−dependence :follows−global) (notes "")* [*Function*]

Creates and returns a new cell type with name NAME and parameters given by the key arguments, if not already defined. Alternative to using the function CREATE−CELL−TYPE, which references the cell−type parameter library. For the dendritic/somatic parameter assignments if the dendritic or somatic argument is not supplied, then the general parameter is used. For example, :RM−SOMA will supersede :RM for the soma. Likewise, if V−LEAK is specified, then this value is assigned to both the :V−LEAK−SOMA and :V−LEAK−DENDRITE slots. If CM is specified, this value is assigned to both the somatic and dendritic slots.

cell-r-in &optional *(cell *cell*)* [*Function*]

Input resistance of CELL, in Mohms [references CELL−Z−DISCRETE−IN−CELL].

rename-cells-simple &optional *(cells (cells))* [*Function*]

Rename CELLS [default all cells in circuit] with simple integer names.

rename-cell-elements-simple &optional *(cell−elements (cells))* [*Function*]

Rename all the somas and segments in CELL−ELEMENTS [default all cells in circuit] with simple integer names.

create-cell *cell−name* &key *cell−type tree−list soma−diameter (segment−diameter 0.5) (origin '(0.0 0.0 0.0)) (name−suffix *cell−name−suffix*) (enable−automatic−cell−names *enable−automatic−cell−names*) (automatic−name−fixing *prompt−for−alternate−element−names*)* [*Function*]

Creates a new cell, if not already defined, of CELL−TYPE. Returns the cell. When creating a new cell, a soma will also be created if SOMA−DIAMETER [microns] is some number [default NIL]. If both SOMA−DIAMETER and TREE−LIST are non−NIL, TREE−LIST is used in a call to CREATE−TREE, with a :DEFAULT−DIAMETER argument given by SEGMENT−DIAMETER [microns, default 0.5]. If the global variable *NEXT−CELL−NAME* is non−NIL, then this will be used instead of CELL−NAME. Always sets *NEXT−CELL−NAME* to NIL on exit. If NAME−SUFFIX is non−NIL [default given by global variable *CELL−NAME−SUFFIX*], it is automatically added as a suffix to the name of a cell, even if this is supplied by *NEXT−CELL−NAME*. CELL−NAME can be a number.

move-cell *cell new−origin* [*Function*]

Moves the absolute location of CELL to the XYZ coordinates [microns] given in the numeric list NEW−ORIGIN.

shift-cell *cell* &key *(x−shift 0.0) (y−shift 0.0) (z−shift 0.0)* [*Function*]

Moves the relative location of CELL according to X−SHIFT, Y−SHIFT and Z−SHIFT [microns, default 0].

cell-element-p *element* [*Function*]

True if ELEMENT is a soma or segment.

45 SYS Source File: cable-functions.lisp

g-element *element g−density* [*Function*]

G−DENSITY is in pS per square micron. Conductance returned is in uS.

cable-lambda *rm ri diameter* [*Function*]

RM in ohms−cm2, RI in ohms−cm, DIAMETER in microns, result in microns.

capacitance-mem *length diameter cm* [*Function*]

Returns membrane capacitance in nF of cable with dimensions LENGTH and DIAMETER [both in microns]. CM is in units of microfarads/cm2.

g-leak-mem *length diameter rm* [*Function*]

Returns membrane leak conductance in uS of cable with dimensions LENGTH and DIAMETER [both in microns]. RM is in units of ohms−cm2.

g-axial *length diameter r−cyto* [*Function*]

Returns axial conductance in uS of cable with dimensions LENGTH and DIAMETER (both in microns). R−CYTO is in units of ohms−cm.

g-soma *radius resistivity* [*Function*]

Return conductance in uS of a spherical membrane with RADIUS in microns, and specific RESISTIVITY in ohms−cm2.

cap-soma *radius capacitance* [*Function*]

Return capacitance in nF of a spherical membrane with RADIUS in microns, and specific CAPACITANCE in microfarads/cm2.

r-in-soma-short-cable *ri rm a−um l−um a−soma−um rm−soma* [*Function*]

Returns somatic input resistance (ohms) to soma−short−cable structure with soma radius A−SOMA−UM in microns, soma membrane resistivity RM−SOMA and cable membrane resistivity RM in ohm−cm2. Intracellular resistivity RI is in ohm−cm, and cable radius A−UM is in microns.

max-g-in &optional *(cell *cell*) (exclude−electrodes t)* [*Function*]

Returns the linear somatic input conductance (uS) of CELL, with the cytoplasmic resistivity set to zero.

cell-cap &optional *(cell *cell*) (exclude−electrodes t)* [*Function*]

Returns the total capacitance of the CELL in nF.

lambda-cable *ri rm a−um* [*Function*]

Cable electrotonic space constant in cm. Intracellular resistivity RI is in ohm−cm, membrane resistivity RM is in ohm−cm2, and cable radius A−UM is in microns.

length-from-lambda *ri rm a−um l* [*Function*]

Returns cable length in um given intracellular resistivity RI [ohm−cm], membrane resistivity RM [ohm−cm2], cable radius A−UM [microns], and electrotonic length L [dimensionless!].

segment-electrotonic-length *segment* [*Function*]

Returns electrotonic length of segment SEGMENT.

electrotonic-length *length diameter cell−type−ri ri−coefficient cell−type−rm* [*Function*]

Returns electrotonic length of segment given explicit parameters LENGTH (uM), DIAMETER (uM), CELL−TYPE−RI (ohms−cm), RI−COEFFICIENT (dimensionless), and CELL−TYPE−RM (ohms−cm2).

g-inf-in *ri rm a−um* &optional *lambda−cable* [*Function*]

Input conductance of semi−infinite cable, in uS. Intracellular resistivity RI is in ohm−cm, membrane resistivity RM is in ohm−cm2, and cable radius A−UM is in microns.

z-cable-in *ri rm a−um l−um* &optional *(g−end 0.0)* [*Function*]

Returns input resistance (Mohms) to sealed−end (open circuit) cable of length L−UM in microns. Intracellular resistivity RI is in ohm−cm, membrane resistivity RM is in ohm−cm2, and cable radius A−UM is in microns. Optional G−END is in uS.

z-cable-in-seg *segment* &key *store−segment−z−cable−in* [*Function*]

Returns input resistance (Mohms) of SEGMENT, taking into account the tree distal to the segment, using the cable parameters.

r-in &optional *(cell *cell*)* [*Function*]

Returns input resistance (Mohms) of CELL, using the cable parameters for the dendritic tree if there is one.

z-cable-in-cell &optional *(cell *cell*) z−tree* [*Function*]

Returns input resistance (Mohms) of CELL, using the cable parameters for the dendritic tree if Z−TREE is not supplied. Otherwise, the input resistance is calculated from the soma resistance and the Z−TREE argument (Mohms).

z-tree-cable-in-cell &optional *(cell *cell*) include−virtual−soma* [*Function*]

Returns input resistance (Mohms) of dendritic tree of CELL, using the cable parameters. If no tree, returns NIL. If INCLUDE−VIRTUAL−SOMA is T, include any segments assigned to the soma.

z-discrete-in-cell &optional *(cell *cell*) z−tree* [*Function*]

Returns somatic input resistance (Mohms) of CELL, using the compartmental network parameters.

z-tree-discrete-in-cell &optional *(cell *cell*) include−virtual−soma* [*Function*]

Returns input resistance (Mohms) of dendritic tree of CELL, using the compartmental network parameters. If no tree, returns NIL. If INCLUDE−VIRTUAL−SOMA is T, include any segments assigned to the soma.

rho &optional *(cell *cell*)* [*Function*]

Tree/soma conductance ratio of cell associated with CELL.

cell-area &optional *(cell *cell*)* [*Function*]

Returns the total membrane area of cell associated with CELL in square microns.

tree-area &optional *(cell *cell*)* [*Function*]

Returns the area in square microns of the dendritic (and axonal) tree attached to the soma of associated with CELL

trunk-3/2s-info &optional *(cell (cells))* [*Function*]

Prints out the 2/3 root of the sum of the dendritic trunk diameters raised to the 3/2.

tree-length &optional *(cell *cell*)* [*Function*]

Returns the total length in microns of the dendritic (and axonal) tree attached to the soma of CELL.

46 SYS Source File: trees.lisp

create-tree *cell−reference segment−list* &key *(xy−factor 1.0) (z−factor 1.0) (add−cell−name−to−segs *add−cell−name−to−segs* add−cell−name−to−segs−supplied−p) (default−soma−diameter *default−soma−diameter*) (default−diameter 0.5)* [*Function*]

Creates a segment tree according to SEGMENT−LIST, adding the tree to the soma associated with CELL−REFERENCE. The associated cell is returned. SEGMENT−LIST is a list of lists, where the sublist format is as follows:

```
(prox-elt-name seg-name x y z &optional diameter extras)
```

The PROX−ELT−NAME refers to the proximal segment or soma, the SEG−NAME is for the segment to be created, and X, Y, and Z refer to the coordinates of the distal node of the segment to be created. EXTRAS is a list of lists for adding channels or synapses to a segment. XY−FACTOR and Z−FACTOR are scaling factors for node coordinates, which may be useful when translating histological renderings into the sublists. The PROX−ELT−NAME of the first sublist will refer to the soma, which has been created already with CREATE−SOMA. For example, the segment sublist:

```
(soma 1a 7 -1 -5 1.2)
```

specifies a segment named 1a whose proximal end connects to the node named soma, whose distal node has coordinates (7*xy−factor, −1*xy−factor, −5*z−factor), and whose diameter is 1.2 microns. Likewise, the segment sublist:

```
(1a 1b 12 -3 -7 0.6 '(KA-HPC))
```

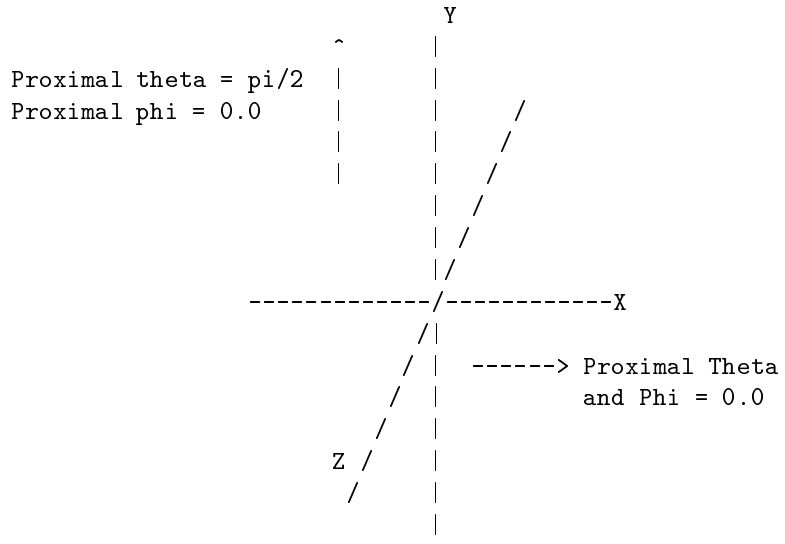
specifies a segment named 1b whose proximal end connects to the distal node of segment 1a, whose distal node has coordinates (12*xy−factor, −3*xy−factor, −7*z−factor), and whose diameter is 0.6 microns. In addition, an KA−HPC type channel is included at the segment's distal node. If the PROX−ELT−NAME and SEG−NAME are the same, then this is the soma, and the diameter is the soma diameter (which overrides the previous diameter). This entry will be used to reference the coordinates of the segments, so that they are created in relative coordinates. The soma origin is set elsewhere if it is to be other than (0 0 0). If the ADD−CELL−NAME−TO−SEGS keyword is T and the global *USE−SIMPLE−NAMES* is nil, then the cell name is prepended to the segment names specified in the segment sublists.

segment-chain *proximal−cell−element chain−name total−segs seg−length seg−diam* &key *(proximal−phi 0.0) (proximal−theta 0.0)* [*Function*]

Adds a straight chain of segments of the same dimensions to the PROXIMAL−CELL−ELEMENT, returning the last (distal) segment in the created chain. If CHAIN−NAME is nil, then the segment names are derived from the cell−name. The PROXIMAL−PHI and PROXIMAL−THETA arguments specify the angle of the branch chain with respect to the PROXIMAL−CELL−ELEMENT, in radians. The default values of 0.0 for PROXIMAL−PHI and PROXIMAL−THETA generate a chain that extends from the PROXIMAL−CELL−ELEMENT in the positive X direction. For a chain of segments that extends in the positive Y direction, include the key argument:

```
:PROXIMAL-THETA (* -0.5 pi)
```

The orientations of the PROXIMAL−PHI and PROXIMAL−THETA arguments are as follows:



tree-control *parent−element depth branch−depth* &key *(length 50.0) (diameter 1.0) (branch−diameter−decrement 0.8) (branch−angle−decrement 0.02)* [*Function*]

Generates a binary tree structure recursively with number of bifurcations equal to DEPTH. Segments are named by their distal node. Each branch of the tree has BRANCH−DEPTH segments, each of LENGTH [microns]. The diameter of a segment at a given branch level is given by BRANCH−DIAMETER−DECREMENT] times the diameter of the previous level (given the DIAMETER [microns] of the first level branch). The bifurcation angle at a given branch level is given by:

```
2 x PI x BRANCH-ANGLE-DECREMENT x DEPTH
```

where DEPTH is the number of bifurcations distal to the present one.

consolidate-cell-tree &key *(cell *cell*) single−step starting−segment (maximum−electrotonic−length *maximum−electrotonic−length*) (use−strict−lambda−criterium *use−strict−lambda−criterium*)* [*Function*]

Consolidate the dendritic tree of CELL, according to the values of MAXIMUM−ELECTROTONIC−LENGTH and USE−STRICT−LAMBDA−CRITERIUM. If SINGLE−STEP is T, then only one pair of segments are consolidated, otherwise successive segment pairs are consolidated moving distally and starting from STARTING−SEGMENT, if supplied, otherwise starting from the trunk segments of the CELL soma. The circuit is processed with PROCESS−CIRCUIT−STRUCTURE at the end of the consolidation.

47 SYS Source File: print.lisp

get-surf-filename *pathname−directory extension* &optional *(trailer "")* [*Function*]

Generate a full pathname of the form:

```
PATHNAME-DIRECTORY/filename.EXTENSION
```

where "filename" is either the current value of *SIMULATION−NAME*, when *ADD−SIMULATION−TO−FILENAM
is T, otherwise the current value of *TIME−STAMP*, in either case postpended with TRAILER.
Any repeated "/" will be replaced with a single "/".

sim-output [*Function*]

Overall simulation output function, normally called automatically at the end of each simulation. Prepares output lists, prints simulation information, writes data files, and plots results, as appropriate. SIM−OUTPUT may also be called during a simulation, where it will process all output data up to the current simulation time.

print-numerical-details [*Function*]

Print out a number of numerical details relevant to the last or current simulation.

print-simulation-stats &optional *complete* [*Function*]

Print out the progress of the last or current simulation and a number of numerical details.

write-element-data &optional *elements−and−slots* &key *(output−format :lisp) filename suppress−comments* [*Function*]

Writes ELEMENT−DATA from all elements referenced in ELEMENTS−AND−SLOTS. ELEMENTS−AND−SLOTS may be a list whose members are either element references or sublists of element references and data types. If ELEMENTS−AND−SLOTS is :MENU, then a menu will be generated. If ELEMENTS−AND−SLOTS is NIL or not supplied, then all data saved from last simulation will be written to file. Remaining args are as for GRAB−AND−STORE−PLOT−DATA. For :LISP OUTPUT−FORMAT, variable symbols are created from the simulation, element and data type name. These symbols are used when the file is loaded into Lisp. For :COLUMNS OUTPUT−FORMATseparate files are written, whose names are constructed as the case for the variable symbols in the :LISP OUTPUT−FORMAT case. In both cases the created variable symbols are listed in the global variable *FILE−OUTPUT−VARIABLE−LIST*. If FILENAME is not supplied then the full pathname is obtained with the GET−SURF−FILENAME function (directory obtained with the GET−SURF−DATA−DIRECTORY function), with file extension "dat".

dump-documented-user-variables-file [*Function*]

Write a loadable file of all documented user variables

dump-elements-file &optional *elements−or−select−each−element* [*Function*]

Write a loadable file with TYPE−DEF forms for selected (loaded) elements which are element types, and CREATE forms for selected elements such as channels, synapses, or sources. Selected elements are determined by the ELEMENT−OR−SELECT−EACH−ELEMENTS argument − this arg can be either a single element, a list of elements, non−NIL (generating a selection menu, or NIL which will select all loaded elements.

print-circuit &optional *(description−level *simulation−print−detail*)* [*Function*]

Print out circuit details. The level of detail is given by the optional DESCRIPTION−LEVEL [default given by global variable *SIMULATION−PRINT−DETAIL* − see this for more explanation].

48 SYS Source File: analysis.lisp

fit-exp *data−list* [*Function*]
start stop &key *(delta−t 1.0) plot−normalize−fit (max−amp−proportion 0.5) (min−amp−proportion 0.05) negative−p return−fit*

Fit single exponential to evenly sampled DATA−LIST points between START and STOP, with grid DELTA−T.

convert-data-dt-lists *data−list old−time−base new−time−base* &optional [*Function*]
(output−order−data−time t) include−time−list new−time−base−is−length

As CONVERT−DATA−TIME−LISTS, with that function's TIME−LIST arg set to a sequence whose length is given by the length of DATA−LIST, starting with 0 and incremented by OLD−TIME−BASE.

convert-data-time-lists *data−list time−base new−time−base* &optional [*Function*]
(output−order−data−time t) include−time−list new−time−base−is−length

Given a list of time points in TIME−BASE, which may not be evenly spaced, and a sequence of data points in DATA−LIST that refer to these time points, generate a data list that is sampled evenly [linear interpolation] at intervals derived from NEW−TIME−BASE. If TIME−BASE is a single number, this is taken as the fixed time step of the input data. If NEW−TIME−BASE−IS−LENGTH is NIL [default NIL] then NEW−TIME−BASE is the new time step, whose units are the same assumed in TIME−BASE. Otherwise, the new time step is chosen so that the resampled data list has length given by NEW−TIME−BASE. For making evenly sampled versions of Surf−Hippo simulations, the current simulation time list is found with CURRENT−SIM−PLOT−TIME−LIST. When the optional INCLUDE−TIME−LIST arg is T, depending on the optional argument OUTPUT−ORDER−DATA−TIME [default T] the function returns:

```
(list new-data-list new-time-list)  <=  OUTPUT-ORDER-DATA-TIME = T
(list new-time-list new-data-list)  <=  OUTPUT-ORDER-DATA-TIME = NIL
```

Otherwise the function returns just the new−data−list.

49 SYS Source File: store-plot-data.lisp

length-element-output-data [*Function*]

Return the length of all the simulation data lists of all the circuit elements.

clear-element-output-data [*Function*]

Clears all simulation data stored by all the circuit elements.

50 SYS Source File: histology-hack.lisp

default-histology-window-background-color *'black* [*Variable*]

Default background color for histology windows (e.g. 'BLACK 'RED 'GREEN 'BLUE 'ORANGE 'CYAN 'PURPLE 'YELLOW 'WHITE).

51 SYS Source File: user-cell-graphics.lisp

just-draw &key *(win (opal−obj−exists *standard−graphics−output*)) mark−elements mark−all−nodes draw−synapse−stimulus (motion−snapshots *motion−snapshots*) (colorize (when win (g−value win :colorize))) (background−color (extract−color (when win (g−value win :background−color)) *default−histology−window−background−color*)) (draw−all−synapse−rfs (when win (g−value win :draw−all−synapse−rfs))) (draw−synapse−rfs (when win (g−value win :draw−synapse−rfs))) (scale (if win (g−value win :scale) 3.0)) (phi−deg (if (and win (g−value win :phi−deg)) (g−value win :phi−deg) 0.0)) (theta−deg (if (and win (g−value win :theta−deg)) (g−value win :theta−deg) 0.0)) (draw−axons t) (draw−synapse−cxns t)* [*Function*]

Draw the current circuit without invoking the Histology Menu. Arguments are the same as those for SET−MISC−HISTO−SLOTS.

histo [*Function*]

Invoke the Histology Menu for drawing the circuit.

mark-elements &optional *marked−elements* &key *(win *standard−graphics−output*) (key−diameter 15)* [*Function*]

MARKED−ELEMENTS can be an atom or list, including 'CHANNEL, 'SYNAPSE, :ALL or specific cell elements, membrane elements or element types. If a cell element is referenced, then all channel or synapses on that cell element will be marked. KEY−DIAMETER is in pixels (default 15). Previously marked elements in the histology WIN will be erased.

set-misc-histo-slots &key *(win *standard−graphics−output*) (scale (if win (g−value win :scale) 3.0)) (colorize (when win (g−value win :colorize))) (background−color (if win (g−value win :background−color) opal:white)) (mark−all−synapses (when win (g−value win :mark−all−synapses))) (enable−marked−synapses (when win (g−value win :enable−marked−synapses))) (draw−all−synapse−rfs (when win (g−value win :draw−all−synapse−rfs))) (draw−synapse−rfs (when win (g−value win :draw−synapse−rfs))) (phi−deg (if (and win (g−value win :phi−deg)) (g−value win :phi−deg) 0.0)) (theta−deg (if (and win (g−value win :theta−deg)) (g−value win :theta−deg) 0.0)) (draw−axons t) (draw−synapse−cxns t)* [*Function*]

Set some basic graphics parameters without using the menus. e.g.

```
(SET-MISC-HISTO-SLOTS :SCALE 3 :PHI-DEG 90 :DRAW-AXONS NIL)
```

Angle args PHI−DEG and THETA−DEG are in degrees, and SCALE arg is in microns/pixel.

label-element *element* &optional *(win *standard−graphics−output*) (update t) label−agg* [*Function*]

Labels the cell element associated with ELEMENT in the histology WIN [default *STANDARD−GRAPHICS−OUTPUT

mark-segments-and-somas *elements* &key *window−menu (win *standard−graphics−output*) (type 'marked−nodes) label−agg mark−agg (marker−diameter (or (and win (g−value win :marker−diameter)) 10)) (mark−somas t) (mark−fill opal:black−fill) (update t)* [*Function*]

Add marker to the cell elements associated with ELEMENTS − in all cases, a marker and/or label will be drawn referring to the underlying segment or soma. Markers are added to the histology in WIN, if supplied, otherwise if WINDOW−MENU is non−nil, then a window selection menu is invoked, otherwise, the current value of *STANDARD−GRAPHICS−OUTPUT* is used.

52 SYS Source File: sparse-data.lisp

read-element-sparse-data &key *(data−type 'voltage) data−filename* [*Function*]

Read sparse data of DATA−TYPE associated with ELEMENTS from file DATA−FILENAME.

write-element-sparse-data &key *(elements (cell−elements)) (data−type 'voltage) data−filename* [*Function*]

Write sparse data of DATA−TYPE associated with ELEMENTS to file DATA−FILENAME.

plot-element-sparse-data *elements* &key *(data−type 'voltage) (y−label "mv")* [*Function*]

Plot sparse data of DATA−TYPE associated with ELEMENTS.

element-sparse-data *element* &optional *data−type* [*Function*]

Return a list of sparse data of DATA−TYPE for ELEMENT.

53 SYS Source File: colorizing.lisp

replay-colorized-simulation &key *(start−time 0) (stop−time *user−stop−time*) (time−step 0.1) win (repetitions 1) (display−time t) (elements (cell−elements)) (data−type 'voltage)* [*Function*]

Runs colorized animation of stored values of DATA−TYPE for ELEMENTS in WIN [if NIL, will run in all histology windows].

show-sparse-data &key *(target−time *real−time*) (data−type 'voltage) window (lock−color t)* [*Function*]

Update colorization in histology WINDOW with DATA−TYPE at TARGET−TIME [ms, default *REAL−TIME*]. If WINDOW is not included, any windows referenced by the variable *COLORIZED−WINDOWS* will be used, otherwise, if none, then all current histology windows. When LOCK−COLOR is T (default) the colorizing is locked.

set-color-map-menu [*Function*]

Menu for reseting the color mapping used in the colorizing code. Assumes voltage scale.

set-color-map &key *(map *default−color−map*) (variable−color−min *colorizing−arrays−color−min*) (variable−color−max *colorizing−arrays−color−max*) (variable−color−step *colorizing−arrays−color−step*) (colors−array *colorizing−colors*) (color−fill−styles−array *cell−fill−styles*) (color−line−styles−array *cell−color−line−styles*) [*Function*]

Resets the color mapping used in the colorizing code. MAP can be any symbol given by *COLOR−MAP−FUNCTIONS*, e.g.

```
'(HOT-JET-RGB-MAP SH-ORIGINAL-RGB-MAP HOT-RGB-MAP JET-RGB-MAP GRAY-RGB-MAP PINK-RGB-MAP
HOT-COLD-RGB-MAP)
```

Note that if too many color maps are used (for SunOs, more than 3), an XLIB:ALLOC−ERROR will occur. This will necessitate restarting Lisp.

54 SYS Source File: plot.lisp

collect-extracted-events &key *element data−type model−type data−dted (dt 0.1) return−traces (pre−event−time 0.0) (post−event−time 0.0) event−times (event−type :spike) (threshold −20) title title−prefix plot draw−grid y−min y−max y−inc x−inc (y−label "mv") (x−label *default−surf−plot−x−label*) (time−base (current−sim−plot−time−list)) state−index* [*Function*]

Extract events from DATA−DTED, which may be a list or a list of lists, if supplied, otherwise from the data of DATA−TYPE associated with ELEMENT of MODEL−TYPE. The time base for event extraction is given by TIME−BASE [default (CURRENT−SIM−PLOT−TIME−LIST)], which is resampled at DT [default 0.1]. The reference times for the extracted events are either explicitly given by EVENT−TIMES, if supplied, otherwise these times are derived by event detection from either DATA−DTED [which in this case must be a flat list], or else from the ELEMENT data as mentioned above. In the case of where events are detected, the detection algorithm is provided by the function ELEMENT−SPIKE−TIMES, searching for events of EVENT−TYPE [default :SPIKE, using the value of THRESHOLD]. The duration of extracted events is defined by PRE−EVENT−TIME and POST−EVENT−TIME, prior to and after the event time, respectively. If PLOT is T, then plot the overlaid events with window TITLE [default 'Superimposed' followed by the EVENT−TYPE], concatenated with TITLE−PREFIX. If RETURN−TRACES returns event traces as

```
((event-1-timebase event-1)
 (event-2-timebase event-2)
  ...
 (event-n-timebase event-n))
```

where each event timebase is a list of times referenced to the associated event time.

phase-plots *element−pairs* &key *title y−label x−label x−min y−min x−max y−max (prompt−for−overlay t)* [*Function*]

The argument ELEMENT−PAIRS is either a list of two elements:

```
(element-1 element-2) -> The data types for each element are defaults from DEFAULT-DATA-TYPE
```

or a list of element pairs (with optional data types), e.g.:

```
(element-pair-1 element-pair-2 ...)
```

where each element−pair−X may specify a data−type for one, both or neither element of the pair:

```
((element-1 data-type) element-2)
(element-1 (element-2 data-type))
((element-1 data-type) (element-2 data-type))
```

55 SYS Source File: 3dplot.lisp

3dplot *array* &key *(theta 0.0) (phi 0.0) (scale 3.0) (gain 1.0) (aspect 1.0) (x−offset 0.0) (y−offset 0.0) (grid−size 100.0) grid−size−x grid−size−y (width 500) (height 500) comment (title "3d plot") win (color 'black) filled (halftone−percent 20)* [*Function*]

Plot a 2 dimensional ARRAY of single floats. Viewing angles THETA and PHI are in degrees.

56 SYS Source File: calc-lte-ratio.lisp

plot-error-step-rasters &key *(include−time−steps t) include−only−used−time−steps font event−height include−particle−steps−<−min−step title* [*Function*]

When LTE adaptive time step used, generates a raster plot of simulation time points [when INCLUDE−TIME−STEPS, default NIL] and those times in which various LTE criteria set the time step, depending on the values of *CALCULATE−PARTICLE−ERROR* and *CALCULATE−CONC−INT−ERRO When INCLUDE−ONLY−USED−TIME−STEPS [default NIL] only those LTE time steps that are finally used will be plottted.

57 SYS Source File: init.lisp

scrub-and-gc &key *verbose (full t) (gen 0) show−dynamic−space* [*Function*]

This function will clean up dead pointers more thouroughly than the basic garbage collection. Explicit calls during large simulations (temporally or spatially) may avoid excessive disk thrashing or memory faults. ** NOT VERIFIED **

initialize-globals-for-circuit [*Function*]

Initialize simulator to accept a new circuit definition.

58 SYS Source File: step.lisp

current-sim-plot-time-list [*Function*]

If current simulation is finished, then return *SIM−PLOT−TIME−LIST*, otherwise, return the reverse of the current value of *SIM−REVERSE−PLOT−TIME−LIST*.

current-sparse-data-times [*Function*]

If current simulation is finished, then return *SPARSE−DATA−TIMES*, otherwise, return the reverse of the current value of *REVERSE−SPARSE−DATA−TIMES*.

59 SYS Source File: hacks.lisp

type-on-the-path-p *type target* [*Function*]

From the point of view of the cell element associated with TARGET, is an element of TYPE on the path to the soma.

branch *element* &optional *type* [*Function*]

Returns branch associated with ELEMENT.

look-at-branchs [*Function*]

Print out branch structure of circuit.

branch-ends *element* [*Function*]

Returns the proximal and distal end segments of the branch associated with ELEMENT.

branch-elements *branch−element element−type* &optional *total−segments ends* [*Function*]

Returns a list of elements of type ELEMENT−TYPE on branch associated with BRANCH−ELEMENT. If TOTAL−SEGMENTS is a number, then segments of the branch are chosen mod TOTAL−SEGMENTS. If ENDS is non−NIL, then only the ends of the branch are considered. Otherwise, all segments of a branch are examined. A branch is defined as a set of singly connected segments whose proximal and distal ends are nodes with more than 2 segments, or a termination (soma or distal tip) point.

branch-synapses-of-type *branch−element type* &optional *total−segments ends* [*Function*]

Returns a list of synapses of TYPE that are associated with BRANCH−ELEMENT (an element on a branch), using BRANCH−ELEMENTS.

branch-channels-of-type *branch−element type* &optional *total−segments ends* [*Function*]

Returns a list of channels of TYPE that are associated with BRANCH−ELEMENT (an element on a branch), using BRANCH−ELEMENTS.

simulation-trial-message *trial* &optional *num−trials message* [*Function*]

Useful for printing out iteration numbers. Insert this in the iteration loops in user defined simulation functions.

log-gc-to-file *nil* [*Variable*]

When true [default NIL], GC messages will be written to a text file.

60 SYS Source File: raster-plot.lisp

default-raster-spacing *20* [*Variable*]

Default raster spacing in pixels as used by RASTER−PLOTS.

default-raster-event-width *1* [*Variable*]

Default raster event width in pixels as used by RASTER−PLOTS.

default-raster-event-height *4* [*Variable*]

Default raster event height in pixels as used by RASTER−PLOTS.

61 SYS Source File: protocols.lisp

std-setup &optional *(element *soma*)* [*Function*]

Plot voltage and add an isource to the somas associated with ELEMENT [default *SOMA*].

vclamp-soma-conductance *clamp−potential−1 clamp−potential−2* &key *(dt 0.1) new−plot (plot−data t) (plot−uncorrected−wave t) comment (cell *cell*) (r−electrode 0.0) (ideal−vsource t) (timeit t) return−g−waves pause−between−clamps (correction−coeff 1.0) (disable−vsources t) interclamp−function run−reg−cap−for−ss* [*Function*]

Calculates the input soma conductance as a function of time using voltage clamp at two holding potentials CLAMP−POTENTIAL−1 and CLAMP−POTENTIAL−2 [mV]. An electrode resistance may be included with R−ELECTRODE [Mohms]. The voltage clamp (ideal if IDEAL−VSOURCE is T) is applied to the soma. Any other voltage sources in the circuit are disabled during the simulation when DISABLE−VSOURCES is T.

clamp-soma-conductance *clamp−command−1 clamp−command−2* &key *(dt 0.1) new−plot (plot−data t) (plot−uncorrected−wave t) (timeit t) pause−between−clamps (vclamp−p t) (cell *cell*) (r−electrode 0.0) (ideal−vsource t) return−g−waves (correction−coeff 1.0) (disable−sources t) interclamp−function run−reg−cap−for−ss* [*Function*]

Calculates the input soma conductance as a function of time using voltage clamp or current clamp protocols, depending on VCLAMP−P, at two clamp command levels CLAMP−COMMAND−1 and CLAMP−COMMAND−2 [mV or nA, as appropriate]. An electrode resistance may be included with R−ELECTRODE [Mohms]. The voltage clamp (ideal if IDEAL−VSOURCE is T) is applied to the soma. Any other voltage sources in the circuit are disabled during the simulation when DISABLE−SOURCES is T.

steady-state-vclamp *v−holding* &key *(vsource *vsource*)* [*Function*]

Run a steady−state voltage clamp simulation with the optional VSOURCE set to V−HOLDING [mV].

f/i &key *step start−stimulus* [*Function*]
*stop−stimulus (isource *isource*) (spike−element *soma*) (spike−threshold −20.0) (supra−threshold−duration−min 0.1) (sub−threshold−time 0.5) stimulus−function (stimulus−units "na") (prompt−for−overlay t) title comment steps−to−plot plot−gains (add−step−to−plot−comment t) (kill−all−output *kill−all−output*) individual−plots (announce−stimulus−step t) (plot−standard−windows *plot−standard−windows*) (lock−f/i−windows t) (stimulus−start−time 10.0) (stimulus−stop−time (* 0.9 *user−stop−time*))*

Generate f/I characteristics, with stimulus magnitudes defined either by STEP if it is a list of values, otherwise by START−STIMULUS, STOP−STIMULUS, and STEP. Stimulus timing is set by STIMULUS−START−TIME [default 10.0ms] and STIMULUS−STOP−TIME

`[default (* 0.9 *USER-STOP-TIME*)]. When STIMULUS-FUNCTION is NIL stimulus [nA] is applied via ISOURCE [default`

ISOURCE]. Otherwise STIMULUS−FUNCTION is called for each magnitude of the stimulus:

```
(funcall STIMULUS-FUNCTION current-stimulus-value START-STIMULUS STOP-STIMULUS)
```

Spikes are detected by application of the function ELEMENT−SPIKE−TIMES applied to SPIKE−ELEMENT [default *SOMA*], with the arguments SPIKE−THRESHOLD [default −20.0mV], SUPRA−THRESHOLD−DURATIO [default 0.1ms] and SUB−THRESHOLD−TIME

`[default 0.5ms]. Plot windows can include the standard simulation plots as setup prior to calling F/I, when`

PLOT−STANDARD−WINDOWS is T or when the stimulus magnitude is a member of STEPS−TO−PLOT. Instantaneous gain in Hz/nA is also plotted when PLOT−GAINS is T. When ADD−STEP−TO−PLOT−COMMENT is T, then the stimulus step value is appended to *SIMULATION−PLOT−WINDOW−COMMENT* for output to the standard simulation plots. If plotting is enabled, then all traces will go to a single plot when INDIVIDUAL−PLOTS is NIL, otherwise each plot will go to a new window.

clamp-steps &key *start−clamp stop−clamp step (source (or *isource* *vsource*)) (clamp−start−time 10.0) (clamp−stop−time (* 0.9 *user−stop−time*)) (holding−potential −70) individual−plots lock−plots (show−plot−windows t) comment include−comment (extra−comment "") timeit return−source−data* [*Function*]

Run a series of clamp simulations where SOURCE [default either *ISOURCE* or *VSOURCE*] has clamp steps taken from STEP, if it is a list, otherwise ranging from START−CLAMP to STOP−CLAMP by STEP [nA or mV, as appropriate]. Each clamp step begins at CLAMP−START−TIME and ends at CLAMP−STOP−TIME, both in milliseconds. When INDIVIDUAL−PLOTS is nil, otherwise output is overlaid on a single plot. Plot window(s) can have the string EXTRA−COMMENT added, and they will be locked if LOCK−PLOTS is set. A non−nil value of INCLUDE−COMMENT adds a comment describing the clamp source values to the plot output, unless a non−nil string COMMENT which will overrule all other comments. SHOW−PLOT−WINDOWS enables plot window updating and refreshing after each simulation − for dense plots, it is more efficient to do this only at the end of the runs. When run with a voltage source, the function INIT−WITH−STEADY−STATE−LINEA is called for every step, using HOLDING−POTENTIAL [mV, default −70]. When RETURN−SOURCE−DATA the function returns as values the source data, the source cell−element voltage, and the time for each step.

find-rm *cell rin* [*Function*]

Finds an approximate value for membrane resistivity, under the constraint that the input resistance of CELL is ideally RIN [Mohms]. Shrinking ranges of Rm are explored, with exponential increments (argument of 10) ranging from 1 to .001 (by decade). Rin associated with returned value is less than than RIN.

62 SYS Source File: sample-cells.lisp

ball-and-sticks &key *name (origin '(0 0 0)) (cell−type 'cortical) (soma−diameter 35) (include−apical t) (apical−dendrite−diameter 12) (apical−dendrite−length 1200) include−basal (basal−dendrite−diameter 12) (basal−dendrite−length 200) ri rm rm−soma cm cm−soma (apical−total−segs 5) (basal−total−segs 5)* [*Function*]

Create and returns a soma/short−cable cell model of CELL−TYPE (default 'CORTICAL) at ORIGIN (default '(0 0 0)), with an "apical" dendrite (when INCLUDE−APICAL is T, the default) with APICAL−DENDRITE−DIAMETER (default 12) and APICAL−DENDRITE−LENGTH in microns (default 1200) and comprised of APICAL−TOTAL−SEGS (default 5), and a "basal" dendrite (when INCLUDE−BASAL is T, default NIL), with similar arguments. Soma is specified with SOMA−DIAMETER in microns (default 35). The cell type parameters RI

`(ohm-cm), RM, RM-SOMA (ohm-cm2), CM, CM-SOMA (uF/cm2) may be explicitly specified if non-NIL (default NIL).`

63 SYS Source File: ntscable.lisp

process-ntscable-list &optional *(cell−name *nts−cell−name*)* [*Function*]

Main function for processing ntscable lisp files.

LOADERS Source Files

64 LOADERS Source File: main-loader.lisp

surf-home *nil* [*Variable*]

The pathname for the top−level Surf−Hippo home directory. Set by the SURFHOME environment variable, if exists, otherwise by the HOME environment variable.

surf-user-home *""* [*Variable*]

The pathname for the user home directory. Set by the HOME environment variable.

surf-user-dir *nil* [*Variable*]

This is the top level Surf−Hippo directory for the user, set by the SURFUSERHOME environment variable, otherwise by $HOME/surf−hippo/. Directory for simulation data, etc.

GUI Source Files

65 GUI Source File: macros.lisp

abort-on-sim-error *t* [*Variable*]

When T SIM−ERROR aborts to top level; otherwise sim−error stays in debugger.

sim-error &optional *(message "") (abort−on−error *abort−on−sim−error*)* [*Function*]

A fatal error occurred, print MESSAGE and abort if ABORT−ON−ERROR [default *ABORT−ON−SIM−ERROR*]; otherwise stay in debugger.

mapcar-return-no-nils *target* &body *body* [*Macro*]

Loop through the result of applying FLATTEN−LIST to TARGET, via the local variable VALUE, collect the results of BODY if non−nil and return them.

d-flt *arg* [*Macro*]

Coerce number ARG to a double−float.

d-flt-list *arg* [*Macro*]

Coerce numeric sequence ARG to a list of double−floats.

d-flt-array *arg* [*Macro*]

Coerce numeric sequence ARG to an array of double−floats.

s-flt *arg* [*Macro*]

Coerce number ARG to a single−float.

s-flt-list *arg* [*Macro*]

Coerce numeric sequence ARG to a list of single−floats.

s-flt-array *arg* [*Macro*]

Coerce numeric sequence ARG to an array of single−floats.

fix *arg* [*Macro*]

Coerce number ARG to a fixnum.

fix-list *arg* [*Macro*]

Coerce numeric sequence ARG to a list of fixnums.

fix-array *arg* [*Macro*]

Coerce numeric sequence ARG to an array of fixnums.

yes-or-no-p-default-no &optional *format−string* &rest *arguments* &key *default* [*Function*]

Clears the input, beeps, prints the message, if any, and reads characters from *QUERY−IO* until the user enters YES (case insensitive) as an affirmative. If user only RETURNs or types something else, then returns DEFAULT [modification from cmucl 17c query.lisp].

y-or-n-p-default-no &optional *format−string* &rest *arguments* &key *default* [*Function*]

Clears the input, beeps, prints the message, if any, and reads characters from *QUERY−IO* until the user enters Y (case insensitive) as an affirmative. If user only RETURNs or types something else, then returns DEFAULT [modification from cmucl 17c query.lisp].

announce-shell-exec *nil* [*Variable*]

Announce execution of shell command by SHELL−EXEC.

shell-exec *command* &optional *(show−result *show−csh−result*)* [*Function*]

Execute the COMMAND string in the shell, announcing the process when *ANNOUNCE−SHELL−EXEC* is T, and announcing the result if SHOW−RESULT is T [default *SHOW−CSH−RESULT*].

software-version [*Function*]

Returns a string describing version of the supporting software.

sim-warning *message* &rest *args* [*Function*]

A warning has occurred, print MESSAGE and continue.

time-form *form* &optional *(repetitions 20)* [*Macro*]

Runs FORM N times, printing avg execution time and returning FORM's value

66 GUI Source File: math.lisp

round-up-to-power-of-2 *number* [*Function*]

Return nearest value greater than or equel to NUMBER that is a integer power of 2.

closest-n-1-vals *list* [*Function*]

Takes a LIST of N numbers and returns a sorted list of the N−1 closest values.

average-closest-n-1-vals *list* [*Function*]

Takes a LIST of N single−float numbers and returns the single float average of the N−1 closest values.

rad-to-deg *angle−in−radians* [*Function*]

Return the angle in degrees corresponding to ANGLE−IN−RADIANS.

deg-to-rad *angle−in−degrees* [*Function*]

Return the angle in radians corresponding to ANGLE−IN−DEGREES.

cartesian-distance-float *x1 y1* &optional *(x2 0.0) (y2 0.0)* [*Function*]

Return as a single float value the cartesian distance between the two single float points [X1 Y1] and the optional [X2 Y2]. X2 and Y2 default to the origin.

cartesian-distance *x1 y1* &optional *(x2 0.0) (y2 0.0)* [*Function*]

Return the cartesian distance between the two points [X1 Y1] and the optional [X2 Y2]. X2 and Y2 default to the origin.

cartesian-vector-distance *x1 x2* [*Function*]

Return the cartesian distance between the two lists of numbers, X1 and X2, of the same arbitrary length. Returns a single float value.

cartesian-vector-distance-float *x1 x2* [*Function*]

Return the cartesian distance between the two lists of single float numbers, X1 and X2, of the same arbitrary length. Returns a single float value.

t-or-nil-p *thing* [*Function*]

Predicate for the explicit symbols T or NIL.

r *listx listy* [*Function*]

Return the correlation coefficient of the number lists.

lin-reg *xylists* &optional *print−results* [*Function*]

Calculates linear regression values for XYLISTS, which has the format ((X−LIST) (Y−LIST)). Internally calculations are done in double precision, but returned number types are either single or double float depending on the types of the arguments. When PRINT−RESULTS is T then linear regression info is printed to the standard output. Returns values as (SLOPE INTERCEPT CORRELATION−COEFF). SLOPE may be a number, :UNDEFINED or :INFINITE. The INTERCEPT [on the X axis] may be a number or :UNDEFINED.

mean *sequence* [*Function*]

Returns single−float mean of SEQUENCE.

mean-sf *sequence* [*Function*]

Returns single−float mean of single−float SEQUENCE, which may be a list or simple array.

mean-df *sequence* [*Function*]

Returns double−float mean of single−float SEQUENCE, which may be a list or simple array.

ss *sequence* [*Function*]

Returns single−float sum of squares of SEQUENCE.

ss-sf *sequence* [*Function*]

Returns single−float sum of squares of the single−float SEQUENCE, which may be a list or simple array.

rms-sf *sequence* [*Function*]

Returns single−float root mean square of the single−float SEQUENCE, which may be a list or simple array.

rms *sequence* [*Function*]

Returns root mean square of SEQUENCE.

median *list* [*Function*]

Returns the single−float median of LIST − when the length of LIST is even, then the median is the average of the values flanking the true median. Also returns a copy of LIST, converted to single floats and sorted.

67 GUI Source File: strings.lisp

print-spaces *stream indent* [*Function*]

Print INDENT spaces to STREAM. If INDENT is not a number, but T, then print 1 space.

substring-found *substring parentstring* [*Function*]

Look for substring in parentstring, if found, returns start position of substring in parentstring, otherwise return NIL.

dollars *stream amount colon−p atsign−p* &optional *(width 0) padchar commachar* [*Function*]

Print an amount of dollars for humans to read. The full form is width,padchar,commachar:@/dollars/, where width is the minimum width of the field (default 0), padchar is the character used to pad to the width on the left end, and commachar is the separator between each group of three digits. The @ modifier controls printing of the sign of positive amounts. If omitted, a positive value prints without a sign. Otherwise, a positive amount has an explicit + sign. The : modifier controls the position of the sign and the padding. If omitted, the dollar sign is printed in the leftmost position, then any intervening pad characters, then the signed value. Otherwise, the sign occupies the leftmost position, and the dollar sign follows any intervening padchars. Copyright 1997 by Erik Naggum. Any use is permitted provided that this copyright notice is retained and that all changes are duly identified.

parse-float *string* [*Function*]

Return a float read from string, and the index to the remainder of string.

68 GUI Source File: sequences.lisp

find-closest-list-value *numeric−list number* [*Function*]

Return as values the value in NUMERIC−LIST that is arithmetically closest to NUMBER, and the index of that value in NUMERIC−LIST.

negate *thing* [*Function*]

Given a number or numeric sequence THING, return the negative version.

negate-single-float *thing* [*Function*]

Given a single float number or single float numeric sequence THING [list or simple array], return the negative version.

normalize-sequence *sequence* &optional *(min−is−zero−p t)* [*Function*]

Return a list of normalized values derived from the numbers in the 1d SEQUENCE. Minimum value is 0 when MIN−IS−ZERO−P is true

`[default], otherwise taken as minimum of sequence.`

add-val-to-float-list *val list* [*Function*]

Return a single float list whose members are given by the sums of VAL and each member of LIST, all of which must be single floats.

scale-float-list *scale list* [*Function*]

Return a single float list whose members are given by the products of SCALE and each member of LIST, all of which must be single floats.

scale-and-offset-float-list *scale offset list* [*Function*]

Return a single float list whose members are given by the products of SCALE and the sum of each member of LIST and OFFSET, all of which must be single floats.

scale-float-array *scale array* [*Function*]

Return a single float array whose members are given by the products of SCALE and each member of ARRAY, all of which must be single floats.

scale-and-offset-float-array *scale offset array* [*Function*]

Return a single float array whose members are given by the products of SCALE and the sum of each member of ARRAY and OFFSET, all of which must be single floats.

atomize-list *thing* [*Function*]

If THING is an atom, returns THING; if THING is a list with one element, returns that element, otherwise returns THING.

list-of-nums *length* &optional *(start 0.0) (increment 1.0)* [*Function*]

Return a list of LENGTH of increasing numbers (type determined by START [default 0.0] and INCREMENT [default 1.0]. The number type will be determined by the type of START and INCREMENT.

list-of-sf-nums *length* &optional *(start 0.0) (increment 1.0)* [*Function*]

As LIST−OF−NUMS with returned list of single floats.

list-of-ints *length* &optional *(start 0) (increment 1)* [*Function*]

As LIST−OF−NUMS with returned list of integers.

array-of-nums *length* &optional *(start 0.0) (increment 1.0)* [*Function*]

Return an array of LENGTH increasing numbers (type determined by START [default 0.0] and INCREMENT [default 1.0].

complement-sequence *seq* [*Function*]

Returns a BOOLEAN complement of the elements in SEQ.

clean-up-list *list* [*Function*]

Remove all NILs in list and delete duplicates.

coerce-to-list *stuff* &rest *rest* [*Function*]

If STUF is an atom, return (LIST STUFF). If REST, then (LIST STUFF REST).

collect-to-array *thing* &rest *rest* [*Function*]

If THING is an atom, return #(THING). If REST, then #(THING REST).

insert-after *newelt list index* [*Function*]

Insert NEWELT in LIST after the INDEXth cell.

`Returns LIST.`

insert-after *newelt list index* [*Function*]

Insert NEWELT in LIST after the INDEXth cell.

`Returns LIST.`

69 GUI Source File: windows-hack.lisp

automatic-run *nil* [*Variable*]

When T suppress GUI, e.g. menus are disabled.

default-plot-window-background-color *'white* [*Variable*]

Default background color for plot windows (e.g. 'BLACK 'RED 'GREEN 'BLUE 'ORANGE 'CYAN 'PURPLE 'YELLOW 'WHITE).

default-comment-position *:upper−right* [*Variable*]

Default comment/title position for graphics windows.

displayed-host-name *""* [*Variable*]

For annotating windows − defaults to actual host.

always-add-host-name-to-windows *nil* [*Variable*]

Always add host name to windows.

add-host-name-to-windows *t* [*Variable*]

Add host name to windows if different than the display server name.

lock-all-windows *nil* [*Variable*]

When T, any new graphics windows are locked.

screen-width *0* [*Variable*]

Slightly adjusted value from OPAL:*SCREEN−WIDTH*.

screen-height *0* [*Variable*]

Slightly adjusted value from OPAL:*SCREEN−height*.

standard-graphics-width *0* [*Variable*]

Default width in pixels for windows initialized with INITIALIZE−GRAPHICS−WINDOW. Initialized by INITIALIZE−WINDOW−SYSTEM−VARIABLES.

standard-graphics-height *0* [*Variable*]

Default height in pixels for windows initialized with INITIALIZE−GRAPHICS−WINDOW. Initialized by INITIALIZE−WINDOW−SYSTEM−VARIABLES.

raise-output-windows *t* [*Variable*]

New content to output windows will raise them.

hide-output-windows *nil* [*Variable*]

Hide all output windows. Overrides *SHOW−OUTPUT−WINDOWS*, *RAISE−OUTPUT−WINDOWS*.

global-window-title-suffix *nil* [*Variable*]

When a non−zero length string, this is added to the end of all window titles, preceded by ': '.

window-tile-x-gap *0* [*Variable*]

Sets the horizontal spacing in pixels used by ARRANGE−WINDOWS.

window-tile-y-gap *0* [*Variable*]

Sets the vertical spacing in pixels used by ARRANGE−WINDOWS.

host-is-display-server [*Function*]

Return T if host machine is display server.

3x2-plot-tiling-square [*Function*]

Set *STANDARD−GRAPHICS−WIDTH* and *STANDARD−GRAPHICS−HEIGHT* for 3x2 tiling, with squares.

3x2-plot-tiling [*Function*]

Set *STANDARD−GRAPHICS−WIDTH* and *STANDARD−GRAPHICS−HEIGHT* for 3x2 tiling, with windows filling screen.

match-win-dimensions &optional *target match (matched−dimensions :width_&_height)* [*Function*]

Resize windows referenced by TARGET to the dimensions of the MATCH window as specified by MATCHED−DIMENSIONS [default :WIDTH_&_HEIGHT, otherwise :WIDTH or :HEIGHT].

add-comment *window comment* &key *font (update t) append−to−old−comment (position *default−comment−position*) (borderp t)* [*Function*]

Add a comment to WINDOW, which may be either a single window or a list of windows, and may be either the pointers to the windows or window titles.

add-window-title-prefix *prefix* &optional *wins* [*Function*]

For each window referenced in WINS [atom or list, default NIL], add PREFIX and a space to the beginning of the title. If WINS not supplied, the windows are selected from a menu of all output windows.

add-window-title-suffix *suffix* &optional *wins* [*Function*]

For each window referenced in WINS [atom or list, default NIL], add a space and SUFFIX to the beginning of the title. If WINS not supplied, the windows are selected from a menu of all output windows.

clear-window *window* &optional *always* [*Function*]

Destroy WINDOW if unlocked, or if ALWAYS is T. Updates *OUTPUT−WINDOWS*. WINDOW may be a list of windows.

unstick-windows [*Function*]

If windows don't respond, maybe they're stuck.

caows [*Function*]

Clear all output windows. Does not kill menus (use MDW).

caulws [*Function*]

Clear all unlocked output windows. Does not kill menus (use MDW).

lock-window &optional *(win *twin*)* [*Function*]

Make sure that WIN (can be a list) is not overwritten. If :ALL then all windows locked.

lock-windows &optional *(wins :all)* [*Function*]

Make sure that none of the output windows are overwritten.

unlock-window &optional *(win *twin*)* [*Function*]

Allow overwriting WIN (can be a list). If :ALL then all windows unlocked.

unlock-all-windows [*Function*]

If windows don't respond, maybe they're stuck. Also unlock them all.

win-menu &optional *(title "select windows") (windows (clean−up−*output−windows*)) pre−selected−window−list only−one* [*Function*]

Returns a list of output windows selected with a menu.

mdw [*Function*]

Destroy any window selected by the mouse.

id-win &optional *(prompt−string "click or type on any object or window...")* [*Function*]

Identifies any window selected by the mouse.

reassociate-windows *nil* [*Variable*]

When T, retiling of windows changes the window list order using REASSOCIATE−WINDOWS.

reassociate-windows-sublist-length *0* [*Variable*]

Association length for REASSOCIATE−WINDOWS.

arrange-windows &key *(windows−per−row *arrange−windows−per−row*) wins use−menu (reassociate−windows *reassociate−windows*) (reassociate−windows−sublist−length *reassociate−windows−sublist−length*) (window−tile−x−gap *window−tile−x−gap*) (window−tile−y−gap *window−tile−y−gap*)* [*Function*]

Retiles the WINS, according to WINDOWS−PER−ROW. If WINS is NIL, or if USE−MENU is T, then a window menu will choose the retiled windows. If WINDOWS−PER−ROW is nil, then there will be a menu for this as well. If WINS is :ALL, then all *OUTPUT−WINDOWS* will be arranged. Tiling references the largest width and height over all windows.

resize-windows *width height* &optional *windows font add−titles* [*Function*]

Resize with WIDTH and HEIGHT [pixels] all WINDOWS, if supplied, otherwise as selected from menu.

70 GUI Source File: print-windows.lisp

delete-delay-after-print *2* [*Variable*]

Delay erasing PS files after printing, in seconds. Needs to be empirically determined.

ps-filename-suffix *""* [*Variable*]

Added to .ps filenames

ps-landscape-p *nil* [*Variable*]

T or NIL, to rotate 90 degrees or portrait.

ps-borders-p *t* [*Variable*]

T, NIL, :GENERIC, or :MOTIF, frames to print around windows.

ps-color-p *t* [*Variable*]

Enable color information in PS file.

ps-left-margin *72* [*Variable*]

Distance in points.

ps-right-margin *72* [*Variable*]

Distance in points.

ps-top-margin *72* [*Variable*]

Distance in points.

ps-bottom-margin *72* [*Variable*]

Distance in points.

lpr-paper-size *:a4* [*Variable*]

:LETTER, :A4, or (WIDTH HEIGHT) in points specifies page size.

ps-position-x *:center* [*Variable*]

:LEFT, :CENTER, or :RIGHT.

ps-position-y *:center* [*Variable*]

:TOP, :CENTER, or :BOTTOM.

ps-scale-x *nil* [*Variable*]

Scale factor for image. Default is NIL, which means the image will be automatically scaled to fit on the page.

ps-scale-y *nil* [*Variable*]

Scale factor for image. Default is NIL, which means the image will be automatically scaled to fit on the page.

ps-file-page-comment *""* [*Variable*]

A string added to the lower left corner of all PS files.

kill-ps-margins *t* [*Variable*]

When T minimize all margins for PS files.

include-filename-and-date-in-ps-files *t* [*Variable*]

Include filename and date in lower right corner of PS files.

print-windows *windows* &key *(landscape *ps−landscape−p*) (include−title *print−windows−include−title*) (what−to−do *print−windows−what−to−do*) (printer *printer*) (print−together *print−together*) directory (filename−suffix *ps−filename−suffix*) arrange erase−files hard−copy−screen use−menu (exclusion−list *print−windows−exclusion−list*) (inclusion−list *print−windows−inclusion−list*)* [*Function*]

Generates and prints PS files of selected WINDOWS on PRINTER [default given by *PRINTER*]. WHAT−TO−DO options [default given by *PRINT−WINDOWS−WHAT−TO−DO*] include:

```
:PRINT-NOW       => Send PS files to PRINTER.
:PRINT-&-DESTROY => Destroy windows after writing and printing PS files.
:JUST-WRITE      => Only write PS files.
:JUST-WRITE-&-DESTROY => Only write PS files, then destroy windows.
:NO_FILE/PRINT   => No files are generated. Used to access auxilary
                    functions such as arranging the selected windows, etc.
```

DIRECTORY applies to the PS files. If not supplied, the file directory will be determined according to the global variable *USE−*PLOT−DIRECTORY*: when T the directory is taken from *PLOT−DIRECTORY*, otherwise from a call to GET−PLOT−DIRECTORY. When WINDOWS is :ALL, then all visible windows are selected for printing according to the other options. Additional options include:

```
ERASE-FILES      => PS files will be erased after printing [default NIL].
                    BUG ALERT: This may erase file before printing occurs.
HARD-COPY-SCREEN => A PS file including all visible windows is printed [default NIL].
PRINT-TOGETHER   => All selected windows are printed in one file [default
                    given by *PRINT-TOGETHER*]. If set to a filename string, this is
                    used to name the PS file. Otherwise an ad-hoc filename is
                    constructed from the selected windows.
ARRANGE          => Arrange selected windows so that they don't overlap [default NIL].
                    If this is a number, then ARRANGE specifies the number of windows
                    per row, used by ARRANGE-WINDOWS.
FILENAME-SUFFIX  => If non-NIL, this will be added to the end of the filename for all
                    created .ps files. The default value is given by *PS-FILENAME-SUFFIX*.
                    Ignored if PRINT-TOGETHER specifies an explicit filename.
```

71 GUI Source File: files.lisp

read-number-file *filename* &key *max−length* [*Function*]

Reads sequential numbers from FILENAME, returns these numbers as a list. File is read until end, unless MAX−LENGTH is a number in which case this [rounded] value sets the number of numbers read.

full-pathname-p *pathname* [*Function*]

Given a PATHNAME string or pathname, returns T if this refers to a full pathname including an existing directory.

simple-format-list *thing* &optional *(stream t) (count 0)* [*Function*]

Print out list THING to STREAM [default T].

dump-lists-2-column *list1* [*Function*]
list2 &key *(filename *results−filename*) (pathname−directory "") announce−write indent comment supersede*

Writes the values in LIST1 and LIST2 in a 2 column format to FILENAME under PATHNAME−DIRECTORY. If PATHNAME−DIRECTORY not supplied, then derive one under the surf−hippo data directory. If included, COMMENT is written to file first, followed by the data.

write-lists-multi-column *lists* &key *(filename *results−filename*) (pathname−directory "") announce−write indent comment supersede (column−width 20) quit−on−first−null* [*Function*]

Writes the values in the sublists of LISTS in a multi−column format to FILENAME under PATHNAME−DIRECTORY. If PATHNAME−DIRECTORY not supplied, then derive one under the surf−hippo data directory. If included, COMMENT is written to file first, followed by the data. Columns are tabbed by COLUMN−WIDTH spaces [default 20]. If a given sublist is empty while running through all the sublists of list, then a space is output for the corresponding column entry, unless QUIT−ON−FIRST−NULL is T [default NIL], in which case the file is closed.

72 GUI Source File: show-image.lisp

show-image *image* &key *(win nil window−supplied) title left top height width (update t) crop image−left image−top (horizontal−side−border 0) (vertical−side−border 0)* [*Function*]

Returns a window, default WIN, that shows the bitmap IMAGE image. IMAGE can be either a Garnet bitmap schema or a filename. If IMAGE is nil then browse. Image size is adjusted by HORIZONTAL−SIDE−BORDER and VERTICAL−SIDE−BORDER. Window dimensions are given by HEIGHT or WIDTH when non−nil, or by the adjusted image size if either CROP is non−nil or WIN is nil. Image placement is given by IMAGE−LEFT and IMAGE−TOP when either is non−nil, otherwise is centered in window. IMAGE−LEFT and IMAGE−TOP can either be specified in pixel coordinates, or by the keywords :LEFT or :RIGHT, or :TOP or :BOTTOM, respectively, thus justifying the image position with respect to the window as indicated. Place window at LEFT and TOP position in the screen, if included. Window TITLE defaults to IMAGE, unless a supplied WIN already has a non−nil :TITLE.

73 GUI Source File: plot-hack.lisp

force-plot-x-fixnums *nil* [*Variable*]

When T, plots will be forced to display X values as fixnums.

force-plot-y-fixnums *nil* [*Variable*]

When T, plots will be forced to display Y values as fixnums.

label-plot-traces *t* [*Variable*]

Default enable for trace labels in plots.

x-axis-tick-mark-length *−5* [*Variable*]

In pixels. If negative, X axis tick marks will point away from tick labels.

y-axis-tick-mark-length *−5* [*Variable*]

In pixels. If negative, Y axis tick marks will point away from tick labels.

accomodate-all-overlays *nil* [*Variable*]

When T the layout of existing windows adapt to any new overlaid data.

preserve-plot-layout *nil* [*Variable*]

When T new plots to existing windows will not change the layout.

overlay-all-plots *nil* [*Variable*]

When T encourage overlays when plotting data to existing windows.

default-x-plot-top-gap-extra-waterfall *20* [*Variable*]

In pixels

default-y-plot-top-gap-extra-waterfall *20* [*Variable*]

In pixels

x-trace-offset *0.0* [*Variable*]

In data units

x-plot-left-gap-waterfall *30* [*Variable*]

In pixels

mark-plot-wins-at-time *time* &key *(win (standard−plot−windows))* *(line−style thick−black−line)* [*Function*]

Add a vertical LINE−STYLE line at TIME, from :y−min to :y−max, to each plot window referenced by WIN.

restore-plots &optional *(windows :all)* *pre−restore−function* [*Function*]

Restore standard plot windows as referenced by WINDOWS [default :ALL], calling PRE−RESTORE−FUNCTION on each window prior to the restore, if supplied.

grab-and-store-plot-data &key *filename win force−menu (output−format :lisp) suppress−comments* [*Function*]

Writes plot data from WIN into FILENAME. Prompts for non−specfied args. OUTPUT−FORMAT specifies how the data are arranged in the output file, as follows:

```
   :OUTPUT-FORMAT key    File format
-------------------------------------------
     :LISP [default]   A list of lists - ((x1 x2 ... xn)(y1 y2 ... yn))
     :COLUMNS          Two columns -
                          x1   y1
                          x2   y2
                             .
                             .
                             .
                          xn   yn
```

For :LISP format, if more than one trace from the plot is selected, the additional trace data are appended to the output file. For :COLUMNS format, multiple traces are written to separate files, where each filename is appended with the trace label. Comments about the plot window and the trace labels are written to the output file, preceded by a ;, unless SUPPRESS−COMMENTS is T [default NIL].

74 GUI Source File: plot-hack-top.lisp

global-plot-comment *nil* [*Variable*]

When set to a string, this comment is added to the window produced by PLOT−XY−DATA, PLOT−TIMED−DATA, PLOT−POINTS, PLOT−SCATTER, PLOT−HISTOGRAM, PLOT−HISTOGRAM−LIST, PLOT−POLAR−DATA, PLOT−POLAR−SIMPLE−ARRAY and PLOT−POLAR−VECTORS, at the position given by *GLOBAL−PLOT−COMMENT−POSITION* [default *DEFAULT−COMMENT−POSITION If this position is the same as the comment position argument of these plotting functions, then any comment supplied to the plotting function is added to the global plot comment.

default-plot-grid-p *nil* [*Variable*]

When T functions such as PLOT−TIMED−DATA will draw a grid by default.

add-trace *new−trace new−label* &optional *win−or−title time−base* [*Function*]

This function adds NEW−TRACE with reference NEW−LABEL to an existing :STANDARD−PLOT plot window, using the time base of the window unless a TIME−BASE is supplied. WIN−OR−TITLE can be the (string) title of an existing window, a window, or NIL. If NIL, then a menu is provided.

dump-plot-to-lisp-file &optional *wins filename (directory *plot−code−directory*) force* [*Function*]

Given optional WINS (list or atom), FILENAME (string), and DIRECTORY [default *PLOT−CODE−DIRECTORY*] write lisp files that recreate a plot window. Menus prompt for missing arguments. Already existing files with same name will be overwritten. FORCE disables all prompts except initial window menu if needed.

add-trace-analysis-to-plot &optional *win* &key *append−to−existing−comment return−strings trace (y−base 0.0) (position :upper−right) (overlay−index 0)* [*Function*]

Adds analysis result of traces in plotting WIN [if NIL then a menu is given] at POSITION [default :UPPER−RIGHT], including integral relative to Y−BASE [default 0], maximum and minimum. Chosen traces are determined by the TRACE [default NIL] and OVERLAY−INDEX [default 0] arguments, as described for the function EXTRACT−PLOT−WINDOW−DATA. If RETURN−STRINGS is T then a list of all generated strings is returned.

extract-plot-window-data &optional *win (trace :menu) (overlay−index 0)* [*Function*]

Extract one or more plot data lists from WIN. Prompts for non−specfied WIN. For overlayed plots, retrieves the overlay according to OVERLAY−INDEX, default 0, referenced from the last overlay. Returns as values DATA and LABELS. DATA is of the form:

```
(((x-list) (y-list)) ... ((x-list) (y-list)))
```

If TRACE is nil then the first trace is returned. Otherwise, if TRACE is an integer or a list of integers, then the traces corresponding to those numbers [starting from 1] are returned. If TRACE is :ALL, then all data lists are included. If TRACE is :MENU, the default, then a menu for the traces is given.

density-plot *array* &key *win (title "2d plot") (overlay nil) (element−aspect−ratio 1.0) (color nil) (x−inc 1) (y−inc 1) (width 400) (height 400) border (vertical−border 50) (side−border 50) left−border right−border (x−axis−tick−skip 0) (x−axis−tick−mark−skip 0) (y−axis−tick−skip 0) (y−axis−tick−mark−skip 0) (x−axis−p t) (y−axis−p t) x−are−fns y−are−fns x−min x−max y−min y−max (x−label "") (y−label "") z−max z−min invert* [*Function*]

BORDER, WIDTH, HEIGHT are in pixels.

Index

2/DELTA-T[N] 17
ABORT-ON-SIM-ERROR 104
ABSOLUTE-CONC-INT-ERROR 20
ABSOLUTE-PARTICLE-ERROR 20
ABSOLUTE-VOLTAGE-ERROR 19
ACCOMODATE-ALL-OVERLAYS 117
ADD-CELL-NAME-TO-SEGS 8
ADD-HOST-NAME-TO-WINDOWS 110
ADD-SIMULATION-TO-FILENAMES 10
ADJUST-BREAKPOINTS-FOR-EVENT-SYNAPSES 21
ALLOW-DUPLICATE-ELEMENTS 9
ALLOW-DUPLICATE-SYNAPTIC-CONNECTIONS 9
ALWAYS-ADD-HOST-NAME-TO-WINDOWS 110
ALWAYS-CLEAR-MODELS 8
ALWAYS-CLEAR-TYPES 8
ALWAYS-INTIALIZE-RANDOM-GEN 24
ANNOUNCE-SHELL-EXEC 105
APERTURE-CENTER-X 23
APERTURE-CENTER-Y 23
APERTURE-RADIUS 23
ARCHIVE-SESSION-RESULTS 11
ARCHIVE-VARIABLE-LIST 11
AUTO-REFRESH-LAST-SIM-REVERSE-TIME-LIST 18
AUTOMATIC-RUN 110
BAR-A-LENGTH 23
BAR-A-POSITION-X 23
BAR-A-POSITION-Y 23
BAR-A-START-TIME 23
BAR-A-STOP-TIME 23
BAR-A-WIDTH 23
BAR-B-LENGTH 24
BAR-B-POSITION-X 24
BAR-B-POSITION-Y 24
BAR-B-START-TIME 24
BAR-B-STOP-TIME 24
BAR-B-WIDTH 24
BAR-LENGTH 22
BAR-WIDTH 22
BEEP-AFTER-GC 7
BEEP-AFTER-SURF 7
BUFFER-INITIALIZATIONS 19
CA-CONC-EXTRA 27
CA-CONC-INTRA 27
CALCULATE-CONC-INT-ERROR 20
CALCULATE-MARKOV-PARTICLE-ERROR 20
CALCULATE-PARTICLE-ERROR 20
CELL-NAME-SUFFIX 8
CIRCUIT 6
CIRCUIT-CATALOG-FUNCTIONS 8
CIRCUIT-DIRECTORY 10
CIRCUIT-DRAWN 6
CIRCUIT-FILE-TYPE 8
CIRCUIT-FUNCTION 6
CIRCUIT-LOADED 6
CIRCUIT-PARTS 8
CIRCUIT-SOURCE 9
CL-CONC-EXTRA 27
CL-CONC-INTRA 27
CM 28
CM-DENDRITE 28
COLORIZE-SIMULATION 12
CONC-INT-INITIALIZATIONS 19
CONSIDER-CONC-INT-ERROR 20
CONSIDER-CONC-PARTICLE-ERROR 20
CONSIDER-ISOURCE-DROP 10
CONSIDER-PARTICLE-ERROR 20
CONSTANT-LIGHT-INPUT-FROM-NEGATIVE-INFINITY 22
CONVERT-LIGHT-RESPONSE-TO-EVENTS-FOR-EACH-SYNAPSE 21
COUNT-ACTIVE-AND-TRIGGERED-SYNAPSES .24
COUNT-ACTIVE-AND-TRIGGERED-SYNAPSES .83
CREATE-NEW-SIMULATION-PLOTS 12
DATA-DIRECTORY 10
DEBUG-ALL-ITERATIONS 66
DEBUG-AT-TIME-STEPS 66
DEBUG-TIME-TRACE 66
DEFAULT-CELL-TYPE-NAME 8
DEFAULT-COMMENT-POSITION 110
DEFAULT-CONC-INT-TYPE-DIFFUSION-DISTANCE 20
DEFAULT-HISTOLOGY-WINDOW-BACKGROUND-COLOR 95
DEFAULT-PLOT-GRID-P 118
DEFAULT-PLOT-WINDOW-BACKGROUND-COLOR 110
DEFAULT-RASTER-EVENT-HEIGHT 100
DEFAULT-RASTER-EVENT-WIDTH 100
DEFAULT-RASTER-SPACING 100
DEFAULT-THINNING-STEP-ACONS 58
DEFAULT-WAVEFORM-STEP 14
DEFAULT-X-PLOT-TOP-GAP-EXTRA-WATERFALL 117
DEFAULT-Y-PLOT-TOP-GAP-EXTRA-WATERFALL 117
DELETE-DELAY-AFTER-PRINT 113
DELTA-T-PRIME[N-1] 17
DELTA-T-PRIME[N] 17
DELTA-T-PRIME[N]-SQUARED 17
DELTA-T[N-1] 17
DELTA-T[N] 17
DELTA-T[N]-SQUARED 17
DISPLAYED-HOST-NAME 110
DOCUMENT-ALL-NEW-VARIABLES 11
DOCUMENT-ELEMENT-TYPE-DEF-DECIMALS ..80
DOCUMENTED-USER-VARIABLES 11
D_BR 25
D_CA 26
D_CL 25

D_CS 25
D_K 25
D_LI 25
D_MG 26
D_NA 25
D_TEA 25
E-CA 27
E-CL 27
E-K 26
E-MG 27
E-NA 26
ENABLE-AXONS 21
ENABLE-CHANNELS 9
ENABLE-COLORIZE-SCALE 12
ENABLE-COLORIZE-TIME 12
ENABLE-DYNAMIC-BREAKPOINT-GENERATION 19
ENABLE-EVENT-GENERATORS 21
ENABLE-LIGHT 22
ENABLE-LIGHT-EVENT-UPDATE 21
ENABLE-PRINT-DOCUMENTED-USER-VARIABLES 11
ENABLE-REORDER-ELEMENTS 10
ENABLE-SPARSE-DATA 12
ENABLE-SYNAPSES 21
FILE-OUTPUT-VARIABLE-LIST 11
FIX-E-CA 27
FIX-E-CL 27
FIX-E-K 27
FIX-E-NA 26
FORCE-PLOT-X-FIXNUMS 116
FORCE-PLOT-Y-FIXNUMS 116
FORMAT-TIME-INCLUDE-DATE-P 65
FORMAT-TIME-LONG-DATE-P 65
FORMAT-TIME-SMALLEST-UNIT 65
FORMAT-TIME-STYLE 65
FRACTIONAL-TIME 16
FULL-ERROR-STEP-CHANGE 20
GLOBAL-PLOT-COMMENT 118
GLOBAL-WINDOW-TITLE-SUFFIX 111
GRATING-SPATIAL-PERIOD 23
GRATING-TEMPORAL-PERIOD 23
GROUP-PLOT-DATA-BY-CELL 13
GROUP-PLOT-DATA-BY-CELL-TYPE 13
HARD-COPY-SCREEN 12
HIDE-OUTPUT-WINDOWS 111
HIDE-PLOT-WINDOWS 13
INCLUDE-CHANNEL-TYPE-COMMENT-IN-PARTICLE-PLOTS 112
INCLUDE-EVENTS-IN-ELEMENT-DOCUMENTATION-CODE 24
INCLUDE-FILENAME-AND-DATE-IN-PS-FILES 114
INITIALIZE-BEFORE-NEXT-CIRCUIT 6
INPUT-IS-FUNCTION 8
INPUT-TIME 16
INTEGER-TIME 17
INTERPOLATE-PARTICLE-ARRAYS 14
ISOURCE-ELECTRODE-RESISTANCE 15
K-CONC-EXTRA 26
K-CONC-INTRA 26
KILL-ALL-OUTPUT 7
KILL-EXTRA-MESSAGES 7
KILL-PS-MARGINS 114
LABEL-PLOT-TRACES 116
LABEL-STIMULUS-TIMES 12
LAST-REAL-TIME 16
LAST-SIM-REVERSE-TIME-LIST 18
LAST-SIM-REVERSE-TIME-STEP-LIST 18
LAST-SIMULATION-FILE-PATH 10
LAST-TIME-STEP 16
LIGHT-DIRECTION 22
LIGHT-INPUT-DELAY 22
LIGHT-INPUT-OFFSET-ANGLE 22
LIGHT-INPUT-OFFSET-DISTANCE 22
LIGHT-SPEED 22
LIGHT-START-POSITION-X 22
LIGHT-START-POSITION-Y 22
LIGHT-STIMULUS 24
LIGHT-STIMULUS-PLANE 24
LIGHT-STIMULUS-TYPES 24
LIGHT-THETA 22
LOADED-CIRCUIT-PARTS 9
LOCK-ALL-WINDOWS 110
LOG-GC-TO-FILE 7
LOG-GC-TO-FILE 99
LPR-PAPER-SIZE 114
LTE-NODE-CRITERIUM 10
MASSAGE-ELEMENT-PLOT-LABELS 12
MAX-ITERATIONS 19
MAX-NUM-TRACES-FOR-PLOT-TRACE-LABELS 13
MAXIMUM-SYNAPSE-PRINTED-EVENTS 21
MG-CONC-EXTRA 27
MG-CONC-INTRA 27
MINIMAL-CAPACITANCE 9
MINIMUM-SOURCE-TRANSITION-TIME 10
MOTION-SNAPSHOTS 12
MULTIPLE-SOURCE-CIRCUIT 6
NA-CONC-EXTRA 26
NA-CONC-INTRA 26
NB-EVENT-SYN-BPS-W-IDEAL-VSOURCE 83
NEXT-CELL-NAME 8
NODE-VOLTAGE-INITIALIZATIONS 18
NOTIFY-EXP-LIMIT 55
NUM-NODES 10
ONLY-LOAD-PASSIVE 9
OVERLAY-ALL-PLOTS 117
PICK-TIME-STEP-FUDGE 20
PLOT-CHANNELS-BY-MAJOR-ION 13
PLOT-DIRECTORY 10
PLOT-NODE-ELEMENTS 14
PLOT-STANDARD-WINDOWS 12
PLOT-TOTAL-CONCS-SEPARATELY 14

PLOT-TOTAL-CONDUCTANCES 14
PLOT-TOTAL-CONDUCTANCES-P 14
PRESERVE-PLOT-LAYOUT 117
PRINT-MATRIX 66
PROMPT-FOR-ALTERNATE-ELEMENT-NAMES ... 9
PS-BORDERS-P 113
PS-BOTTOM-MARGIN 114
PS-COLOR-P 113
PS-FILE-PAGE-COMMENT 114
PS-FILENAME-SUFFIX 113
PS-LANDSCAPE-P 113
PS-LEFT-MARGIN 113
PS-POSITION-X 114
PS-POSITION-Y 114
PS-RIGHT-MARGIN 114
PS-SCALE-X 114
PS-SCALE-Y 114
PS-TOP-MARGIN 114
PUMP-INITIALIZATIONS 19
PWL-ISOURCE-DI-DT 14
PWL-VSOURCE-DV-DT 15
R-EXTRACELLULAR 28
RAISE-OUTPUT-WINDOWS 110
REAL-TIME 16
REASSOCIATE-WINDOWS 113
REASSOCIATE-WINDOWS-SUBLIST-LENGTH . 113
RECTIFY-SYNAPSE-CONDUCTANCES 19
RI 28
RM 28
RM-SOMA 28
SAVE-CONDUCTANCES-NORMALIZED 13
SAVE-DATA-CALLED-P 14
SAVE-DATA-STEP 13
SAVE-SIMULATION-DATA 11
SAVE-SIMULATION-DATA-TO-FILE 11
SCREEN-HEIGHT 110
SCREEN-WIDTH 110
SCRUB-AND-GC-ON-GLOBAL-INITIALIZATION 7
SESSION-NAME 7
SETUP-EVENT-GENERATORS-AND-FOLLOWERS 21
SHOW-TIME-REMAINING 7
SHOW-TIME-REMAINING-UPDATE-TIME 8
SIM-PLOT-TIME-LIST 18
SIM-REVERSE-PLOT-TIME-LIST 18
SIM-REVERSE-TIME-LIST 18
SIM-REVERSE-TIME-STEP-LIST 18
SIM-TIME-N 15
SIM-TIME-N+1 15
SIM-TIME-N-1 15
SIM-TIME-N-2 15
SIMULATION-FINISHED 7
SIMULATION-IN-PROGRESS 6
SIMULATION-INITIALIZED 6
SIMULATION-MAX-TIME 16
SIMULATION-NAME 6
SIMULATION-PLOT-WINDOW-COMMENT 13
SIMULATION-PLOT-WINDOW-COMMENT-POSITION 13
SIMULATION-PRINT-DETAIL 11
SIMULATION-STARTED 6
SOMA-SHUNT 20
SPARSE-DATA-STEP 12
STANDARD-GRAPHICS-HEIGHT 110
STANDARD-GRAPHICS-WIDTH 110
SUPPRESS-ELEMENT-CREATION-MESSAGES ... 7
SURF-HOME 103
SURF-INTERACTIVE 7
SURF-USER-DIR 103
SURF-USER-HOME 103
T-PRIME[N+1] 16
T-PRIME[N-PRIME-1] 16
T-PRIME[N-PRIME] 16
T-PRIME[N] 16
TEMP-CELCIUS 6
TEMPERATURE 6
TIME-STEP 15
TOTAL-NUM-ITERATIONS 19
TOTAL-NUM-TIME-POINTS 19
TRACES-PER-PLOT 13
T[N+1] 16
T[N] 16
T[N]-T-PRIME[N-PRIME] 17
USE-APERTURE 23
USE-BUFFER-INITIALIZATIONS 18
USE-CONC-INT-INITIALIZATIONS 18
USE-FIXED-STEP 17
USE-MAX-ITERATIONS 19
USE-NODE-VOLTAGE-INITIALIZATIONS 18
USE-PUMP-INITIALIZATIONS 18
USE-SIMPLE-NAMES 9
USE-SIMULATION-NAME-FOR-SIMULATION-PLOT-TITLES 13
USE-TIME-LIST 17
USER-BREAKPOINT-LIST 19
USER-MAX-STEP 15
USER-MIN-STEP 15
USER-OUTPUT-DATA-FUNCTIONS 9
USER-SAVE-DATA-FUNCTIONS 9
USER-SPECIFIED-EVENT-ELEMENT-SETS ... 21
USER-STEP 17
USER-STOP-TIME 15
V-LEAK 26
V-LEAK-DENDRITE 26
VCLAMP-DEFAULT-MAGNITUDE 15
VSOURCE-RESISTANCE 15
WAVE-CUTOFF 64
WINDOW-TILE-X-GAP 111
WINDOW-TILE-Y-GAP 111
WRITE-LOG-FILE 7
X-AXIS-TICK-MARK-LENGTH 116

X-PLOT-LEFT-GAP-WATERFALL 117
X-TRACE-OFFSET 117
Y-AXIS-TICK-MARK-LENGTH 116
3DPLOT 98
3X2-PLOT-TILING 111
3X2-PLOT-TILING-SQUARE 111
ACTIVE-SYNAPSES 82
ADD-COMMENT 111
ADD-CONSTANT-CURRENT 72
ADD-DELAY-TO-WAVEFORM 63
ADD-IELECTRODE 78
ADD-ISOURCE 75
ADD-POISSON-EVENTS 84
ADD-TRACE 118
ADD-TRACE-ANALYSIS-TO-PLOT 118
ADD-VAL-TO-FLOAT-LIST 108
ADD-VELECTRODE 78
ADD-VSOURCE 76
ADD-WAVEFORM 77
ADD-WAVEFORM-TO-ELEMENT-CONTROL-WAVEFORM 77
ADD-WINDOW-TITLE-PREFIX 111
ADD-WINDOW-TITLE-SUFFIX 111
ALL-DATA-TYPES 53
ALPHA 64
ALPHA-LIST 64
ARRANGE-WINDOWS 113
ARRAY-OF-NUMS 109
ARRAY-VOL 63
AS-THE-CROW-FLIES 50
ATOMIZE-LIST 108
ATTACHED-TO-SOMA-P 75
AVERAGE-CLOSEST-N-1-VALS 105
AXON 32
AXON-FLOATS 32
AXON-TYPE 32
AXON-TYPES 41
AXONS 41
BALL-AND-STICKS 102
BOLTZMANN-EQUATION 69
BOLTZMANNS-CONSTANT 25
BRANCH 99
BRANCH-CHANNELS-OF-TYPE 99
BRANCH-ELEMENTS 99
BRANCH-ENDS 99
BRANCH-SYNAPSES-OF-TYPE 99
BUFFER 33
BUFFER-DOUBLE-FLOATS 33
BUFFER-TYPE 32
BUFFER-TYPES 41
BUFFERS 41
CABLE-LAMBDA 88
CAOWS 112
CAP-SOMA 89
CAPACITANCE-MEM 88
CARTESIAN-DISTANCE 106
CARTESIAN-DISTANCE-FLOAT 106
CARTESIAN-VECTOR-DISTANCE 106
CARTESIAN-VECTOR-DISTANCE-FLOAT 106
CAULWS 112
CELL 32
CELL-AREA 90
CELL-CAP 89
CELL-DISTAL-SEGMENTS 51
CELL-ELEMENT 46
CELL-ELEMENT-ELEMENTS 42
CELL-ELEMENT-P 88
CELL-ELEMENTS 42
CELL-ISOURCE 76
CELL-R-IN 87
CELL-TYPE 32
CELL-TYPE-DEF 34
CELL-TYPE-PARAMETER 86
CELL-TYPES 39
CELL-TYPES 42
CELL-VSOURCE 76
CELLS 39
CHANNEL 30
CHANNEL-ACTIVE-P 80
CHANNEL-CONDUCTANCE 33
CHANNEL-CURRENT 33
CHANNEL-E-REV 33
CHANNEL-GBAR 33
CHANNEL-GBAR/PERM-REFERENCE-VALUE 34
CHANNEL-IV-REFERENCE-VALUE 34
CHANNEL-PARTICLE-POWER 80
CHANNEL-REFERENCE-TEMP 33
CHANNEL-TYPE 30
CHANNEL-TYPE-DEF 35
CHANNEL-TYPES 39
CHANNELS 39
CHANNELS-OF-TYPE 81
CIRCUIT-OBJECT-TYPE ..... NIL 42
CLAMP-SOMA-CONDUCTANCE 100
CLAMP-STEPS 101
CLEAN-UP-LIST 109
CLEAR-CONSTANT-CURRENTS 73
CLEAR-ELEMENT-OUTPUT-DATA 94
CLEAR-EVENTS 84
CLEAR-PLOT-TOTAL-CONDUCTANCES 54
CLEAR-USER-VARIABLES 65
CLEAR-WINDOW 112
CLOSEST-ELEMENT 50
CLOSEST-N-1-VALS 105
COERCE-TO-LIST 109
COLLECT-EXTRACTED-EVENTS 97
COLLECT-TO-ARRAY 109
COMPLEMENT-SEQUENCE 109
CONC-INT 32

CONC-INT-ACTIVE-P 68
CONC-INT-CORE-FREE-CONC 68
CONC-INT-DIFF-AREA 68
CONC-INT-DOUBLE-FLOATS 31
CONC-INT-MEMBRANE-CURRENT-COMPONENT ... 68
CONC-INT-SHELL-1-FREE-CONC-N 67
CONC-INT-SHELL-1-FREE-CONC-N+1 67
CONC-INT-SHELL-2-FREE-CONC-N 68
CONC-INT-SHELL-2-FREE-CONC-N+1 68
CONC-INT-SHELL-3-FREE-CONC-N 68
CONC-INT-SHELL-3-FREE-CONC-N+1 68
CONC-INT-SHELL-MEMBRANE-AREA 68
CONC-INT-TYPE 31
CONC-INT-TYPE-DEF 37
CONC-INT-TYPES 40
CONC-INTS 40
CONC-PARTICLE 31
CONC-PARTICLE-CONCENTRATION-ARG 82
CONC-PARTICLE-TYPE 31
CONC-PARTICLE-TYPES 40
CONC-PARTICLES 40
CONCENTRATION-CLAMP 67
CONCENTRATION-CLAMP-OFF 67
CONSOLIDATE-CELL-TREE 92
CONSTANT-FIELD-EQUATION 69
CONSTANT-FIELD-EQUATION-DOUBLE 69
CONSTANT-FIELD-EQUATION-EXPONENTIAL-TERM 69
CONSTANT-FIELD-EQUATION-EXPONENTIAL-TERM-DOUBLE 69
CONVERT-DATA-DT-LISTS 94
CONVERT-DATA-TIME-LISTS 94
CONVERT-IV-RELATIONS-TO-DENSITIES 45
CORE-OFF-DIAG 72
CORE-VOLUME 68
CORRELATED-UNIT-RATE-POISSON-PROCESSES 58
CREATE-AXON 85
CREATE-AXON-TYPE 85
CREATE-CELL 87
CREATE-CELL-TYPE 86
CREATE-CELL-TYPE-W-PARAMS 87
CREATE-CHANNEL 80
CREATE-CHANNEL-TYPE 80
CREATE-ELEMENT 43
CREATE-SEGMENT 74
CREATE-SEGMENT-FAST 74
CREATE-SOMA 73
CREATE-SYNAPSE 82
CREATE-SYNAPSE-TYPE 82
CREATE-TREE 91
CUMULATIVE-PDF-ELEMENT-DISTRIBUTION ... 48
CURRENT-SIM-PLOT-TIME-LIST 98
CURRENT-SPARSE-DATA-TIMES 98
CYLINDER-VOLUME 55
D-FLT 104
D-FLT-ARRAY 104
D-FLT-LIST 104
DATA-AMPLITUDE 61
DATA-EXTREME 60
DATA-MAX 62
DATA-MAX-SLOPE 61
DATA-MIN 62
DATA-MIN-SLOPE 62
DATA-WINDOW 59
DECLARE-GROUND 72
DEFAULT-DATA-TYPE 53
DEFAULT-DIFFUSION-COEFFICIENT 67
DEFAULT-ION-REVERSAL-POTENTIAL 69
DEG-TO-RAD 106
DENSITY-PLOT 119
DF-RANDOM-NOT-ZERO 57
DIFFERENTIATE-FLOAT-WAVE 59
DIFFERENTIATE-WAVE 58
DISABLE-ALL-ELEMENT-PLOT 53
DISABLE-ELEMENT 47
DISABLE-ELEMENT-PLOT 53
DISABLE-ELEMENT-SAVE-DATA 41
DISABLE-INDIVIDUAL-PULSES 77
DISABLE-PULSE-TRAIN 77
DISTAL-SEGMENTS 75
DISTAL-TIP-P 75
DISTAL-TIPS 50
DISTALS-WITHOUT 46
DISTANCE-TO-SOMA 50
DOCUMENT-ELEMENT 47
DOLLARS 107
DOUBLE-ALPHA 65
DOUBLE-EXPONENTIAL 64
DRIVEN-SYNAPSES 83
DUMMY-DOUBLE-FLOATS 28
DUMMY-FIXNUMS 28
DUMMY-FLOATS 28
DUMP-DOCUMENTED-USER-VARIABLES-FILE ... 93
DUMP-ELEMENTS-FILE 93
DUMP-LISTS-2-COLUMN 115
DUMP-PLOT-TO-LISP-FILE 118
EDIT-ELEMENT 47
EDIT-SOURCE 76
EFFECTIVE-REVERSAL-POTENTIAL 80
ELECTRODE-CAPACITANCE 79
ELECTRODE-LEAK-RESISTANCE 79
ELECTRODE-RESISTANCE 79
ELECTRODE-SOURCE 79
ELECTRODES 79
ELECTROTONIC-DISTANCE-TO-SOMA 50
ELECTROTONIC-LENGTH 89
ELEMENT 42
ELEMENT-10-90-RISE-TIME 61
ELEMENT-10-90-SLOPE 61
ELEMENT-AMPLITUDE 61

ELEMENT-AREA 52
ELEMENT-CAPACITANCE 44
ELEMENT-CAPACITANCE-CURRENT 44
ELEMENT-CELL 47
ELEMENT-CLOUD 50
ELEMENT-CONCENTRATION-VOLUME 53
ELEMENT-CONDUCTANCE 44
ELEMENT-CONDUCTANCE 45
ELEMENT-CONSTANT-CURRENT 72
ELEMENT-CONTROL-WAVEFORM 47
ELEMENT-CONTROL-WAVEFORM-TIMESTEP 47
ELEMENT-CURRENT 44
ELEMENT-DATA 48
ELEMENT-DATA-DFT 56
ELEMENT-DATA-DTED 49
ELEMENT-DATA-WINDOW 59
ELEMENT-DIAMETER 52
ELEMENT-DISTRIBUTION 48
ELEMENT-DVDT 44
ELEMENT-EXTREME 60
ELEMENT-FIRING-FREQUENCY 60
ELEMENT-G-LEAK 44
ELEMENT-GBAR 45
ELEMENT-HOLDING-POTENTIAL 72
ELEMENT-INTEGRATED-DATA 62
ELEMENT-IV-DENSITY 45
ELEMENT-LEAK-CURRENT 44
ELEMENT-LENGTH 52
ELEMENT-LOCATION 49
ELEMENT-MAX 62
ELEMENT-MAX-SLOPE 61
ELEMENT-MIN 62
ELEMENT-MIN-SLOPE 62
ELEMENT-NAME 47
ELEMENT-NODE 43
ELEMENT-OF-ION-TYPE-P 44
ELEMENT-PARAM-DISTRIBUTION 51
ELEMENT-PARAMETER 45
ELEMENT-PARAMETER-FAST 45
ELEMENT-RELATIVE-CONDUCTANCE 45
ELEMENT-RESTING-POTENTIAL 73
ELEMENT-REVERSAL-POTENTIAL 44
ELEMENT-SOMA 46
ELEMENT-SPARSE-DATA 96
ELEMENT-SPIKE-HEIGHTS 60
ELEMENT-SPIKE-THRESHOLDS 60
ELEMENT-SPIKE-TIMES 59
ELEMENT-SV-RATIO 53
ELEMENT-TYPE 46
ELEMENT-TYPE-P 42
ELEMENT-VALUE 46
ELEMENT-VOLTAGE 44
ELEMENT-VOLUME 52
ELEMENTS-OF-TYPE 43
ENABLE-ELEMENT 47
ENABLE-ELEMENT-PLOT 53
ENABLE-ELEMENT-SAVE-DATA 41
ENABLE-INDIVIDUAL-PULSES 77
ENABLE-PULSE-TRAIN 77
ERASE-ELEMENT 47
ERASE-ELEMENT-TYPE 46
EVENT-GENERATOR 86
EVENTS 84
EXPAND-TIME-REFERENCE 59
EXPONENTIAL 64
EXPONENTIAL-ARRAY-UNIT-AREA 64
EXPONENTIAL-PDF 56
EXPONENTIAL-RANDOM-NUMBER 57
EXTRACELLULAR-ELECTRODE 33
EXTRACELLULAR-ELECTRODES 41
EXTRACT-PLOT-WINDOW-DATA 119
F/I 100
FARADAY 25
FAST-FRACTIONAL-PART 55
FIND-CLOSEST-LIST-VALUE 108
FIND-RM 101
FIND-ZERO-CROSSINGS 63
FIT-EXP 94
FIX 104
FIX-ARRAY 104
FIX-LIST 104
FLOAT-MOD 55
FOVERR 25
FRAME-MIN-MAXS 63
FULL-PATHNAME-P 115
G-AXIAL 88
G-ELEMENT 88
G-INF-IN 89
G-LEAK-MEM 88
G-SOMA 88
GASCONSTANT 25
GAUSSIAN-ELEMENT-DISTRIBUTION 48
GET-SURF-DATA-DIRECTORY 65
GET-SURF-FILENAME 93
GET-SURF-PLOT-DIRECTORY 66
GHK-POTENTIAL 69
GOFERIT 71
GOQUIET 71
GOTIMED 71
GRAB-AND-STORE-PLOT-DATA 117
HISTO 95
HISTOGRAM-SYNAPSE-EVENTS 84
HOST-IS-DISPLAY-SERVER 111
HWHH 62
ID-WIN 112
IDEAL-VSOURCE 78
IMAG-FROM-MAG-PHASE 55
IMPINGING-SYNAPSES 83

INITIALIZE-GLOBALS-FOR-CIRCUIT 98
INSERT-AFTER 109
INSERT-AFTER 109
INTEGRATE-WAVE 62
INTERDIGITATION-AREA 68
ISOURCE 30
ISOURCE-FLOATS 30
ISOURCE-TYPE 29
ISOURCE-TYPES 40
ISOURCES 40
IV-TYPE-PARAMETER 79
JUST-DRAW 95
LABEL-ELEMENT 95
LAMBDA-CABLE 89
LENGTH-ELEMENT-OUTPUT-DATA 94
LENGTH-FROM-LAMBDA 89
LIBRARY-CATALOG 43
LIGHT-CONTROLLED-P 84
LIN-REG 106
LIST-MAXS 63
LIST-MINS 63
LIST-OF-INTS 109
LIST-OF-NUMS 109
LIST-OF-SF-NUMS 109
LOAD-AND-COMPILE-USER-SOURCE 65
LOAD-SURF-HOME-FILE 66
LOAD-SURF-USER-FILE 66
LOCK-WINDOW 112
LOCK-WINDOWS 112
LOOK-AT-BRANCHS 99
MAPCAR-RETURN-NO-NILS 104
MARK-ELEMENTS 95
MARK-PLOT-WINS-AT-TIME 117
MARK-SEGMENTS-AND-SOMAS 96
MATCH-WIN-DIMENSIONS 111
MAX-G-IN 89
MDW 112
MEAN 106
MEAN-DF 107
MEAN-SF 106
MEDIAN 107
MEMBRANE-AREA-DISTRIBUTION 52
MEMBRANE-ELEMENT-IV-VALUES 30
MEMBRANE-ELEMENT-TYPE-IV 30
MIDPOINTS 59
MIN-MAX-CIRCUIT-COORDINATES 49
MODEL 28
MODULATED-POISSON-EVENTS 56
MOVE-CELL 88
NB-ACTIVE-CHS-OF-TYPE 80
NB-ACTIVE-CHS-OF-TYPE 82
NEGATE 108
NEGATE-SINGLE-FLOAT 108
NEIGHBORS 46
NERNST-POTENTIAL 69
NODE 29
NODE-DOUBLE-FLOATS 28
NODE-FIXNUMS 29
NODES 41
NON-IDEAL-VSOURCE 78
NONLINEARITY 55
NORMAL-RANDOM-NUMBER 57
NORMALIZE-SEQUENCE 108
NTH-DERIVATIVE 58
PARSE-FLOAT 107
PARTICLE 31
PARTICLE-DOUBLE-FLOATS 31
PARTICLE-TYPE 30
PARTICLE-TYPE-DEF 36
PARTICLE-TYPES 40
PARTICLES 40
PHASE-PLOTS 97
PLANCKS-CONSTANT 25
PLOT-ELEMENT 54
PLOT-ELEMENT-SPARSE-DATA 96
PLOT-ERROR-STEP-RASTERS 98
PLOT-SCATTER-SYNAPSE-DISTANCES-EVENT-TIMES 84
PLOT-SEGMENTS-TO-SOMA 54
PLOTTED-ELEMENTS 53
POISSON-EVENTS 56
POISSON-INTERVAL 56
PORE-BLOCKED-P 79
PRE-SYNAPTIC-ELEMENT 42
PRIMARY-SEGS 51
PRINT-CIRCUIT 93
PRINT-ELEMENT 47
PRINT-NODE-DV-STATES 72
PRINT-NODE-STATES 72
PRINT-NUMERICAL-DETAILS 93
PRINT-OBJ-SIZE 67
PRINT-SIMULATION-STATS 93
PRINT-SPACES 107
PRINT-WINDOWS 114
PROCESS-NTSCABLE-LIST 102
PROFILE-ALL 66
PROXIMAL-CELL-ELEMENT 75
PROXIMAL-SEGMENT 75
PROXIMALS-WITHIN 46
PULSE 63
PULSE-LIST 76
PULSE-TRAIN 76
PUMP 33
PUMP-CONC-INT-COMPARTMENT-VOLUME 67
PUMP-CONCENTRATION-CURRENT 67
PUMP-DOUBLE-FLOATS 33
PUMP-TYPE 33
PUMP-TYPE-DEF 38
PUMP-TYPES 41

PUMPS 41
Q10-FACTOR 70
Q10-RATE-FACTOR 70
Q10-TAU-FACTOR 70
QUEUE-BREAKPOINT-TIME 71
QUEUE-EVENT-SYNAPSE-BREAKPOINT-TIME ... 83
R 106
R-IN 90
R-IN-SOMA-SHORT-CABLE 89
RAD-TO-DEG 105
RANDOM-NOT-ZERO 57
RANDOM-NTH 57
RANDOM-NTH-FRACTION 58
RANDOM-PHASE-SEQUENCE 57
RANDOM-SEGMENT 44
RANDOM-SUBSEQ 58
READ-ELEMENT-SPARSE-DATA 96
READ-IN-CIRCUIT 71
READ-NUMBER-FILE 115
REAL-FROM-MAG-PHASE 55
REMOVE-EVENTS 84
RENAME-AXONS-SIMPLE 85
RENAME-BUFFERS-SIMPLE 85
RENAME-CELL-ELEMENTS-SIMPLE 87
RENAME-CELLS-SIMPLE 87
RENAME-CHANNELS-SIMPLE 80
RENAME-CONC-PARTICLES-SIMPLE 82
RENAME-ISOURCES-SIMPLE 78
RENAME-PARTICLES-SIMPLE 81
RENAME-PUMPS-SIMPLE 85
RENAME-SEGMENTS-SIMPLE 75
RENAME-SYNAPSES-SIMPLE 82
RENAME-VSOURCES-SIMPLE 78
REORDER-CIRCUIT 72
REORDER-ELEMENTS-OF-TYPE 42
REPLAY-COLORIZED-SIMULATION 96
RESIZE-WINDOWS 113
RESTORE-PLOTS 117
RETURN-MARKOV-RATE 81
RHO 90
RMS 107
RMS-SF 107
ROUND-UP-TO-POWER-OF-2 105
S-FLT 104
S-FLT-ARRAY 104
S-FLT-LIST 104
SAMPLE-S 55
SCALE-AND-OFFSET-FLOAT-ARRAY 108
SCALE-AND-OFFSET-FLOAT-LIST 108
SCALE-FLOAT-ARRAY 108
SCALE-FLOAT-LIST 108
SCALED-EXPONENTIAL-RATE 70
SCALED-EXPONENTIAL-RATE-DOUBLE 70
SCALED-EXPONENTIAL-RATE-DOUBLE 70
SCALED-EXPONENTIAL-SOFT-RATE 70
SCALED-SIGMOID-RATE 70
SCALED-SLANTED-STEP-RATE 71
SCRUB-AND-GC 98
SEGMENT 29
SEGMENT-AREA 52
SEGMENT-CAPACITANCE 29
SEGMENT-CHAIN 91
SEGMENT-CM 74
SEGMENT-ELECTROTONIC-LENGTH 89
SEGMENT-G-AXIAL 29
SEGMENT-G-LEAK 29
SEGMENT-GUTS 29
SEGMENT-RI 74
SEGMENT-RI-COEFFICIENT 74
SEGMENT-RM 74
SEGMENT-V-LEAK 74
SEGMENTS 39
SEGMENTS-IN 51
SEGMENTS-OUT 51
SEGS-UNTIL-BIFURCATION 75
SERIAL-COEFF 58
SET-*NODE-VOLTAGE-INITIALIZATIONS* 72
SET-CELCIUS-TEMPERATURE 71
SET-COLOR-MAP 97
SET-COLOR-MAP-MENU 96
SET-ELECTRODE-CAPACITANCE 79
SET-ELECTRODE-RESISTANCE 79
SET-ELEMENT-ABSOLUTE-IV-REFERENCE 45
SET-ELEMENT-MEMBRANE-PARAMETERS 45
SET-MISC-HISTO-SLOTS 95
SET-PARTICLE-TYPE-SS 81
SET-PARTICLE-TYPE-TAU 81
SET-SEGMENT-ABSOLUTE-PARAMETERS 74
SET-SEGMENT-PARAMETER 75
SET-SEGMENTS-INHERIT-PARAMETERS-FROM-TYPE 74
SET-SOMA-ABSOLUTE-PARAMETERS 73
SET-SOMA-PARAMETER 73
SETUP-PLOT-TOTAL-CONDUCTANCES 53
SF-FIND-GAMMA-WAITING-TIME 58
SF-FIND-POISSON-WAITING-TIME 58
SF-RANDOM-NOT-ZERO 57
SHELL-EXEC 105
SHIFT-CELL 88
SHOW-IMAGE 116
SHOW-SPARSE-DATA 96
SHUFFLED-INDICES 56
SHUFFLED-LIST 56
SIM-ERROR 104
SIM-OUTPUT 93
SIM-WARNING 105
SIMPLE-FORMAT-LIST 115
SIMULATION-TRIAL-MESSAGE 99
SINEWAVE 63

SOFTWARE-VERSION 105
SOMA 29
SOMA-AREA 52
SOMA-CAPACITANCE 29
SOMA-G-LEAK 29
SOMA-GUTS 29
SOMA-MEMBRANE-RESISTIVITY 73
SOMA-SEGMENTS 51
SOMA-SPECIFIC-CAPACITANCE 73
SOMA-V-LEAK 73
SOMA-VOLTAGE 73
SOMAS 39
SOURCES 75
SPHERE-AREA 54
SPHERE-AREA-CM2 54
SPHERE-AREA-FROM-DIAMETER 54
SPHERE-DIAMETER-FROM-AREA 54
SPHERE-DIAMETER-FROM-CAPACITANCE 54
SPHERE-VOLUME 54
SPHERE-VOLUME-FROM-DIAMETER 54
SQUEEZED-EXPONENTIAL 70
SS 107
SS-SF 107
STD-SETUP 100
STEADY-STATE-VCLAMP 100
SUBSTRING-FOUND 107
SURF 71
SYNAPSE 31
SYNAPSE-ACTIVE-P 82
SYNAPSE-CONDUCTANCE 34
SYNAPSE-CURRENT 34
SYNAPSE-E-REV 34
SYNAPSE-GBAR 34
SYNAPSE-GBAR/PERM-REFERENCE-VALUE 34
SYNAPSE-IV-REFERENCE-VALUE 34
SYNAPSE-REFERENCE-TEMP 34
SYNAPSE-TYPE 31
SYNAPSE-TYPE-DEF 35
SYNAPSE-TYPES 39
SYNAPSES 39
SYNAPSES-OF-TYPE 83
SYNAPTIC-TARGETS 83
SYSTEM-OF-DIFFERENTIAL-EQUATIONS 30
T-OR-NIL-P 106
TEST-CORRELATED-PROCESSES 58
TIME-FORM 105
TIME-INNARDS 66
TOPLOAD 71
TREE-AREA 90
TREE-CONTROL 92
TREE-LENGTH 91
TREE-RADIUS 50
TRUNK-3/2S-INFO 90
TRUNK-SEGMENT 51
TRUNK-SEGMENTS 51
TYPE-INSTANCES-IN-CELL 43
TYPE-ON-THE-PATH-P 99
UNLOCK-ALL-WINDOWS 112
UNLOCK-WINDOW 112
UNSTICK-WINDOWS 112
UPDATE-TYPE-FROM-DEFINITION 45
USER-SETUP-EVENT-GENERATORS-AND-FOLLOWERS 86
V-FUNCTION-ARRAY 81
VALENCE-FROM-K 69
VCLAMP-SOMA-CONDUCTANCE 100
VSOURCE 30
VSOURCE-TYPE 30
VSOURCE-TYPES 40
VSOURCES 40
WHERE 49
WHERE-X 49
WHERE-Y 49
WHERE-Z 49
WIN-MENU 112
WRITE-ELEMENT-DATA 93
WRITE-ELEMENT-SPARSE-DATA 96
WRITE-LISTS-MULTI-COLUMN 116
Y-OR-N-P-DEFAULT-NO 105
YES-OR-NO-P-DEFAULT-NO 105
Z-CABLE-IN 89
Z-CABLE-IN-CELL 90
Z-CABLE-IN-SEG 90
Z-DISCRETE-IN-CELL 90
Z-TREE-CABLE-IN-CELL 90
Z-TREE-DISCRETE-IN-CELL 90