NUTMEG

NAME
nutmeg spice post-processor

SYNOPSIS
nutmeg [-] [-n] [-t term] [datafile ...]

DESCRIPTION
Nutmeg is a post processor for SPICE it takes the raw output file created by spice -r and plots the data on a graphics terminal or a workstation display. Note that the raw output file is different from the data that SPICE writes to the standard output.

Arguments are:

- Don't try to load the default data file ("rawspice") if no other files are given.

-n (or **-N**)
Don't try to source the file ".spiceinit" upon startup. Normally **nutmeg** tries to find the file in the current directory, and if it is not found then in the user's home directory.

-t term (or **-T term**)
The program is being run on a terminal with *mfb* name **term**.

Further arguments are taken to be data files in binary or ascii format (see **sconvert**(1)) which are loaded into nutmeg. If the file is in binary format, it may be only partially completed (useful for examining SPICE ouput before the simulation is finished). One file may contain any number of data sets from different analyses.

Nutmeg data is in the form of vectors: time, voltage, etc. Each vector has a type, and vectors can be operated on and combined algebraicly in ways consistent with their types. Vectors are normally created when a data file is read in (see the load command below), and when the initial datafile is loaded. They can also be created with the let command.

An expression is an algebraic formula involving vectors and scalars (a scalar is a vector of length 1), and the following operations:

+, -, *, %, /, ^, and ,.

% is the modulo operator, and the comma operator has two meanings: if it is present in the argument list of a user-definable function, it serves to seperate the arguments. Otherwise, the term **x , y** is synonymous with **x + j(y)**.

Also available are the logical operations & (and), | (or), ! (not), and the relational operations <, >, >=, <=, =, and <> (not equal). If used in an algebraic expression they work like they would in C, producing values of 0 or 1. The relational operators have the following synonyms: **"gt"** is >, **"lt"** is <, **"ge"** is >=, **"le"** is <=, **"ne"** is <>, **"eq"** is =, **"and"** is &, **"or"** is |, and **"not"** is !. These are useful when < and > might be confused with IO redirection (which is almost always).

The following functions are available:

mag(vector) The magnitude of vector.

ph(vector) The phase of vector.

j(vector) *i* (sqrt(-1)) times vector.

real(vector) The real component of vector.

imag(vector) The imaginary part of vector.

db(vector) 20 * log10(mag(vector)).

log(vector) The logarithm (base 10) of the vector.

ln(vector) The natural logarithm (base e) of vector.

exp(vector) e to the vector power.

abs(vector) The absolute value of vector.

sqrt(vector) The square root of vector.

sin(vector) The sin of vector.

cos(vector) The cosine of vector.

tan(vector) The tangent of vector.

atan(vector) The inverse tangent of vector.

norm(vector) The **vector** normalized to 1 (i.e, the largest magnitude of any component will be 1).

rnd(vector) A vector with each component a random integer between 0 and the absolute value of the vectors's corresponding component.

mean(vector) The result is a scalar (a length 1 vector) that is the mean of the elements of **vector**.

vector(number) The result is a vector of length **number**, with elements 0, 1, ... **number - 1**. If **number** is a vector then just the first element is taken, and if it isn't an integer then the floor of the magnitude is used.

length(vector) The length of **vector**.

interpolate(plot.vector) The result of interpolating the named vector onto the scale of the current plot. This function uses the variable **polydegree** to determine the degree of interpolation.

A vector may be either the name of a vector already defined, a floating- point number (a scalar), or a list like **[elt1 elt2 ... eltn]**, which is a vector of length n. A number may be written in any format acceptable to SPICE, such as **14.6MEG** or **-1.231E-4**. Note that you can either use scientific notation or one of the abbreviations like *MEG* or *G*, but not both. As with SPICE, a number may have trailing alphabetic characters after it.

The notation **expr [lower upper]**, where **lower** and **upper** are numbers, denotes the range of elements from **expr** between **lower** and **upper**. The notation **expr [num]** denotes the **num**'th element of **expr**. If **upper** is lower than **lower**, the order of the elements in the vector is reversed. In all other cases, **[** and **]** serve to surround literal vectors as described above. (You may have to use a lot of parentheses to make sure that you get what you want. For instance, you have to type **print (foo) ([1 2])** to print the two vectors. Otherwise it will be interpreted as a function call or a vector with an index.) Note that the expression **foo[10 20][5]** will *not* yield the 15th element of **foo**, but rather the 5th. In general only the last index suffix on an expression will take effect.

To reference vectors in a plot that is not the *current plot* (see the **setplot** command, below), the notation **plotname.vecname** can be used.

Either a plotname or a vector name may be the wildcard **all**. If the plotname is **all**, matching vectors from all plots are specified, and if the vector name is **all**, all vectors in the specified plots are referenced. Note that you may not use binary operations on expressions involving wildcards it is not obvious what **all + all** should denote, for instance.

Thus some (contrived) examples of expressions are:

cos(TIME) + db(v(3))

sin(cos(log([1 2 3 4 5 6 7 8 9 10])))

TIME * rnd(v(9)) - 15 * cos(vin#branch) ^ [7.9e5 8]

not ((ac3.FREQ[32] & tran1.TIME[10]) gt 3)

Nutmeg commands are as follows:
plot exprs [ylimit ylo yhi] [xlimit xlo xhi] [xindices xilo xihi]
[xcompress comp] [xdelta xdel] [ydelta ydel] [xlog] [ylog]
[vs xname] [xlabel word] [ylabel word] [title word] [samep]
Plot the given **exprs** on the screen (if you are on a graphics terminal). The **xlimit** and **ylimit** arguments determine the high and low x- and y-limits of the axes, respectively. The **xindices** arguments determine what range of points are to be plotted everything between the **xilo**'th point and the **xihi**'th point is plotted. The **xcompress** argument specifies that only one out of every **comp** points should be plotted. If an **xdelta** or a **ydelta** parameter is present, it specifies the spacing between grid lines on the X- and Y-axis. These parameter names may be abbreviated to **xl**, **yl**, **xind**, **xcomp**, **xdel** and **ydel** respectively. The **xname** argument is an expression to use as the scale on the x-axis. If **xlog** or **ylog** are present, the X or Y scale respectively will be logarithmic. The **xlabel** and **ylabel** arguments cause the specified labels to be used for the X and Y axes, respectively. If **samep** is given, the values of the other parameters (other than **xname**) from the previous **plot, hardcopy,** or **asciiplot** command will be used unless re-defined on the command line. Finally, the **title** argument will be used in the place of the plot name at the bottom of the graph.
hardcopy file *plotargs*
Just like **plot**, except creates a file called **file** containing the plot. The file is an image in *plot(5)* format, and can be printed by either the **plot(1)** program or **lpr** with the **-g** flag.

asciiplot *plotargs*
Produce a line printer plot of the vectors. The plot is sent to the standard output, so you can put it into a file with *asciiplot args* ... > *file*. The **set** options **width, height,** and **nobreak** determine the width and height of the plot, and whether there are page breaks, respectively. Note that you will have problems if you try to **asciiplot** something with an X-scale that isn't monotonic (i.e, something like *sin(TIME)*), because **asciiplot** uses a simple-minded sort of linear interpolation.

define function(arg1, arg2, ...) expression
Define the *user-definable function* with the name *function* and arguments *arg1, arg2,* ... to be *expression,* which may involve the arguments. When the function is later used, the arguments it

is given are substituted for the formal arguments when it is parsed. If *expression* is not present, any definition for *function* is printed, and if there are no arguments to *define* then all currently active definitions are printed. Note that you may have different functions defined with the same name but different arities. Some useful definitions are:
define max(x,y) (x > y) * x + (x <= y) * y
define min(x,y) (x < y) * x + (x >= y) * y

undefine function ...
Definitions for the named user-defined functions are deleted.

let name = expr
Creates a new vector called **name** with the value specified by **expr**, an expression as described above. If **expr** is [] (a zero-length vector) then the vector becomes undefined. If there are no arguments, **let** is the same as **display**.

print [col] [line] expr ...
Prints the vector described by the expression expr. If the col argument is present, print the vectors named side by side. If **line** is given, the vectors are printed horizontally. **col** is the default, unless all the vectors named have a length of one, in which case **line** is the default. The options **width, length,** and **nobreak** are effective for this command (see **asciiplot**). If the expression is **all**, all of the vectors available are printed. Thus **print col all > file** will print everything in the file in SPICE2 format. The scale vector (time, frequency) will always be in the first column unless the variable **noprintscale** is true.

load [filename] ...
Loads the raw data in either binary or ascii format from the files named. The default filename is **rawspice**, or the argument to the **-r** flag if there was one.

source filename
Reads commands from the file filename. Lines beginning with the character ***** are considered comments and ignored.

help [all] [command ...]
Prints help. If the argument **all** is given, a short description of everything you could possibly type is printed. If **command**s are given, descriptions of those commands are printed. Otherwise help for only a few major commands is printed.

display [varname ...]
Prints a summary of currently defined vectors, or of the names specified. The vectors are sorted by name unless the variable **nosort** is set. The information given is the name of the vector, the length, the type of the vector, and whether it is real or complex data. Additionally, one vector will be labeled **[scale]**. When a command such as *plot* is given without a *vs* argument, this scale is used for the X-axis. It is always the first vector in a rawfile, or the first vector defined in a new plot. If you undefine the scale (i.e, *let TIME = []*), a random remaining vector will become the scale.

setplot [plotname]
Set the **current plot** to the plot with the given name, or if no name is given, prompt the user with a menu. (Note that the plots are named as they are loaded, with names like **tran1** or **op2**. These names are shown by the **setplot** and **display** commands and are used by **diff**, below.) If the "New plot" item is selected, the current plot will become one with no vectors defined. Note that here the word "plot" refers to a group of vectors that are the result of one SPICE run. When more than

one file is loaded in, or more than one plot is present in one file, **nutmeg** keeps them seperate and only shows you the vectors in the current plot.

settype type vector ...
Change the type of the named vectors to **type**. Type names can be found in the manual page for **sconvert**.

diff plot1 plot2 [vec ...]
Compare all the vectors in the specified *plots*, or only the named vectors if any are given. There are different vectors in the two plots, or any values in the vectors differ significantly the difference is reported. The variables **abstol, reltol,** and **vntol** are used to determine what "significantly" means (see the SPICE3 User's Manual).

quit Quit nutmeg.

bug Send a bug report. (If you have defined BUGADDR, the mail will go there.)

write [file] [exprs]
Writes out the expr's to file. First vectors are grouped together by plots, and written out as such. (I.e, if the expression list contained three vectors from one plot and two from another, then two plots will be written, one with three vectors and one with two.) Additionally, if the scale for a vector isn't present, it is automatically written out as well. The default format is ascii, but this can be changed with the **set filetype** command. The default filename is **rawspice**, or the argument to the **-r** flag on the command line, if there was one, and the default expression list is **all**.

shell [args ...]
Fork a shell, or execute the arguments as a command to the shell.

alias [word] [text ...]
Causes **word** to be aliased to **text**. History substitutions may be used, as in C-shell aliases.

unalias [word ...]
Removes any aliases present for the **word**s.

history [number]
Print out the history, or the last **number** commands typed at the keyboard. *Note:* in \\*S version 3a7 and earlier, all commands (including ones read from files) were saved.

set [word] [word = value] ...
Set the value of **word** to be **value**, if it is present. You can set any word to be any value, numeric or string. If no value is given then the value is the boolean 'true'. The value of *word* may be inserted into a command by writing *$word*. If a variable is set to a list of values that are enclosed in parentheses (which **must** be seperated from their values by white space), the value of the variable is the list. The variables meaningful to **nutmeg** (of which there are too many) are:

abstol The absolute tolerance used by the **diff** command.

appendwrite
Append to the file when a **write** command is issued, if one already exists.

color*N* These variables determine the colors used, if **X** is being run on a color display. *N* may be between 0 and 15. Color 0 is the background, color 1 is the grid and text color, and colors 2 through 15 are used in order for vectors plotted. The value of the **color** variables should be names of colors, which may be found in the file **/usr/lib/rgb.txt**.

combplot
Plot vectors by drawing a vertical line from each point to the X-axis, as opposed to joining the points. Note that this option is subsumed in the *plottype* option, below.

cpdebug
Print *cshpar* debugging information. (Must be complied with the -DCPDEBUG flag.)

debug If set then a lot of debugging information is printed. (Must be compiled with the -DFTEDEBUG flag.)

device The name (/dev/tty??) of the graphics device. If this variable isn't set then the user's terminal is used. To do plotting on another monitor you will probably have to set both the **device** and **term** variables. (If **device** is set to the name of a file, **nutmeg** will dump the graphics control codes into this file -- this is useful for saving plots.)

echo Print out each command before it is executed.

filetype This can be either ascii or binary, and determines what the format of rawfiles will be. The default is ascii.

fourgridsize
How many points to use for interpolating into when doing fourier analysis.

gridsize If this variable is set to an integer, this number will be used as the number of equally spaced points to use for the Y-axis when plotting. Otherwise the current scale will be used (which may not have equally spaced points). If the current scale isn't strictly monotonic, then this option will have no effect.

hcopydev
If this is set, when the **hardcopy** command is run the resulting file is automatically printed on the printer named **hcopydev** with the command *lpr -P***hcopydev** *-g* **file**.

hcopydevtype
This variable specifies the type of the printer output to use in the **hardcopy** command. If hcopydevtype is not set, plot (5) format is assumed. The standard distribution currently recognizes **postscript** as an alternative output format. When used in conjunction with **hcopydev**, **hcopydevtype** should specify a format supported by the printer.

height The length of the page for **asciiplot** and **print col**.

history The number of events to save in the history list.

nfreqs The number of frequencies to compute in the fourier command. (Defaults to 10.)

nobreak
Don't have **asciiplot** and **print col** break between pages.

noasciiplotvalue
Don't print the first vector plotted to the left when doing an **asciiplot**.

noclobber
Don't overwrite existing files when doing IO redirection.

noglob Don't expand the global characters `*', `?', `[', and `]'. This is the default.

nogrid Don't plot a grid when graphing curves (but do label the axes).

nomoremode
If **nomoremode** is not set, whenever a large amount of data is being printed to the screen (e.g, the **print** or **asciiplot** commands), the output will be stopped every screenful and will continue

when a carriage return is typed. If **nomoremode** is set then data will scroll off the screen without hesitation.

nonomatch
If **noglob** is unset and a global expression cannot be matched, use the global characters literally instead of complaining.

nosort Don't have **display** sort the variable names.

noprintscale
Don't print the scale in the leftmost column when a **print col** command is given.

numdgt The number of digits to print when printing tables of data (**fourier, print col**). The default precision is 6 digits. On the VAX, approximately 16 decimal digits are available using double precision, so **numdgt** should not be more than 16. If the number is negative, one fewer digit is printed to ensure constant widths in tables.

plottype
This should be one of *normal*, *comb*, or *point:***chars**. *normal*, the default, causes points to be plotted as parts of connected lines. *comb* causes a comb plot to be done (see the description of the *combplot* variable above). *point* causes each point to be plotted seperately the **chars** are a list of characters that will be used for each vector plotted. If they are omitted then a default set is used.

polydegree
The degree of the polynomial that the **plot** command should fit to the data. If *polydegree* is N, then **nutmeg** will fit a degree N polynomial to every set of N points and draw 10 intermediate points in between each endpoint. If the points aren't monotonic, then it will try rotating the curve and reducing the degree until a fit is achieved.

polysteps
The number of points to interpolate between every pair of points available when doing curve fitting. The default is 10. (This should really be done automatically.)

program The name of the current program (*argv[0]*).

prompt The prompt, with the character `!' replaced by the current event number.

rawfile The default name for rawfiles created.

reltol The relative tolerance used by the **diff** command.

rhost The machine to use for remote SPICE-3 runs, instead of the default one. (See the description of the **rspice** command, below.)

rprogram
The name of the remote program to use in the **rspice** command.

slowplot
Stop between each graph plotted and wait for the user to type return before continuing.

sourcepath
A list of the directories to search when a **source** command is given. The default is the current directory and the standard spice library (*/usr/local/lib/spice*, or whatever **LIBPATH** is #defined to in the \\*S source.

spicepath
The program to use for the **aspice** command. The default is /cad/bin/spice.

term The *mfb* name of the current terminal.

units If this is **degrees**, then all the trig functions will use degrees instead of radians.

unixcom If a command isn't defined, try to execute it as a UNIX command. Setting this option has the effect of giving a **rehash** command, below. This is useful for people who want to use **nutmeg** as a login shell.

verbose Be verbose. This is midway between **echo** and **debug / cpdebug**.

vnto The absolute voltage tolerance used by the **diff** command.

width The width of the page for **asciiplot** and **print col**.

xbrushheight
The height of the brush to use if **X** is being run.

xbrushwidth
The width of the brush to use if **X** is being run.

xfont The name of the X font to use when plotting data and entering labels. The plot may not look entirely great if this is a variable-width font.

unset [word] ...
Unset the variables **word**.

shift [varname] [number]
If *varname* is the name of a list variable, it is shifted to the left by *number* elements. (I.e, the *number* leftmost elements are removed.) The default *varname* is **argv**, and the default *number* is 1.

rusage [resource ...] Print resource usage statistics. If any **resource**s are given, just print the usage of that resource. Currently valid **resource**s are:

elapsed The amount of time elapsed since the last **rusage elaped** call.

faults Number of page faults and context switches (BSD only).

space Data space used.

time CPU time used so far.

everything
All of the above.

cd [directory]
Change the current working directory to **directory**, or to the user's home directory if none is given.

aspice [output-file]
Start a SPICE-3 run, and when it is finished load the data. The raw data is kept in a temporary file. If *output-file* is specified then the diagnostic output is directed into that file, otherwise it is thrown away.

jobs Report on the asynchronous SPICE-3 jobs currently running. **Nutmeg** checks to see if the jobs are finished every time you execute a command. If it is done then the data is loaded and becomes available.

rspice [input file]
Runs a SPICE-3 remotely taking the **input file** as a SPICE-3 input deck, or the current circuit if no argument is given. **Nutmeg** waits for the job to complete, and passes output from the remote

job to the user's standard output. When the job is finished the data is loaded in as with aspice. If the variable *rhost* is set, **nutmeg** will connect to this host instead of the default remote SPICE-3 server machine. Note that this command will only work if your system administrator is running a SPICE-3 daemon on the remote host. If the variable *rprogram* is set, then **rspice** will use this as the pathname to the program to run.

echo [stuff...]
Echos the arguments.

fourier fundamental_frequency [value ...]
Does a fourier analysis of each of the given values, using the first 10 multiples of the fundamental frequency (or the first *nfreqs*, if that variable is set see below). The output is like that of the **.four** \\*S card. The values may be any valid expression. The values are interpolated onto a fixed-space grid with the number of points given by the **fourgridsize** variable, or 200 if it is not set. The interpolation will be of degree **polydegree** if that variable is set, or 1. If **polydegree** is 0, then no interpolation will be done. This is likely to give erroneous results if the time scale is not monotonic, though.

version [version id]
Print out the version of **nutmeg** that is running. If there are arguments, it checks to make sure that the arguments match the current version of SPICE. (This is mainly used as a **Command:** line in rawfiles.)

rehash Recalculate the internal hash tables used when looking up UNIX commands, and make all UNIX commands in the user's PATH available for command completion. This is useless unless you have **set unixcom** first (see above).

The following control structures are available:
while *condition*
statement

...

end

While *condition*, an arbitrary algebraic expression, is true, execute the statements.
repeat *[number]*
statement

...

end

Execute the statements *number* times, or forever if no argument is given.
dowhile *condition*
statement

...

end

The same as **while**, except that the *condition* is tested after the statements are executed.
foreach *var value ...*
statement

...

end

The statements are executed once for each of the *value*s, each time with the variable *var* set to the current one. (*var* can be accessed by the $*var* notation see below).
if *condition*
statement

...

else

statement

...

end

If the *condition* is non-zero then the first set of statements are executed, otherwise the second set. The **else** and the second set of statements may be omitted.
label *word*
If a statement of the form **goto** *word* is encountered, control is transfered to this point, otherwise this is a no-op.
goto *word*
If a statement of the form **label** *word* is present in the block or an enclosing block, control is transfered there. Note that if the label is at the top level, it *must* be before the **goto** statement (i.e, a forward **goto** may occur only within a block).
continue
If there is a **while, dowhile,** or **foreach** block enclosing this statement, control passes to the test, or in the case of **foreach**, the next value is taken. Otherwise an error results.
break
If there is a **while, dowhile,** or **foreach** block enclosing this statement, control passes out of the block. Otherwise an error results.

Of course, control structures may be nested. When a block is entered and the input is the terminal, the prompt becomes a number of >'s equalling the number of blocks the user has entered. The current control structures may be examined with the debugging command cdump.

If a word is typed as a command, and there is no built-in command with that name, the directories in the *sourcepath* list are searched in order for the file. If it is found, it is read in as a command file (as if it were **source**d). Before it is read, however, the variables *argc* and *argv* are set to the number of words following the filename on the command line, and a list of those words respectively. After the file is finished, these variables are **unset**. Note that if a command file calls another, it must save its *argv* and *argc* since they will get altered. Also, command files may not be re-entrant since there are no local variables. (Of course, the procedures may explicitly manipulate a stack...) This way one can write scripts analogous to shell scripts for **nutmeg** and \\*S. Note that for the script to work with \\*S, it **must** begin with a blank line (or whatever you like, since it will be thrown away) and then a line with **.control** on it. This is an unfortunate result of the **source** command being used for both circuit input and command file execution. Note also that this allows the user to merely type the name of a circuit file as a command, and it will be automatically run.

There are various command scripts installed in */usr/local/lib/spice/scripts* (or whatever the path is on your machine), and the default *sourcepath* includes this directory, so you can use these command files (almost) like builtin commands.

Nutmeg will use either **X** or **MFB**, depending on whether it finds the variable **DISPLAY** in the environment. If you are using **X** on a workstation, it should already be present, but if you want to display graphics on a different machine than the one you are running **nutmeg** on, **DISPLAY** should be of the form *machine*:0.

If **X** is being used, the cursor may be positioned at any point on the screen when the window is up and characters typed at the keyboard will be added to the window at that point. The window may then be sent to a printer using the **xpr(1)** program.

There are a number of pre-defined constants in **nutmeg**. They are:

pi	pi
e	The base of natural logarithms
c	The speed of light
i	The square root of -1
kelvin	Absolute 0 in Centigrade
echarge	The charge on an electron
boltz	Boltzman's constant
planck	Planck's constant (h)

These are all in MKS units. If you have another variable with a name that conflicts with one of these then it takes precedence.

Nutmeg occasionally checks to see if it is getting close to running out of space, and warns the user if this is the case. (This is more likely to be useful with the SPICE front end.)

C-shell type quoting with "" and '', and backquote substitution may be used. Within single quotes, no further substitution (like history substitution) is done, and within double quotes, the words are kept together but further substitution is done. Any text between backquotes is replaced by the result of executing the text as a command to the shell.

Tenex-style ('set filec' in the 4.3 C-shell) command, filename, and keyword completion is possible: If EOF (control-D) is typed after the first character on the line, a list of the commands or possible arguments is printed. (If it is alone on the line it will exit **nutmeg**.) If escape is typed, then **nutmeg** will try to complete what the user has already typed. To get a list of all commands, the user should type &ltspace> ^D.

The values of variables may be used in commands by writing **$varname** where the value of the variable is to appear. The special variables *$$* and *$<* refer to the process ID of the program and a line of input which is read from the terminal when the variable is evaluated, respectively. If a variable has a name of the form **$&word**, then **word** is considered a vector (see above), and its value is taken to be the value of the variable. If *$foo* is a valid variable, and is of type **list**, then the expression *$foo[low-high]* represents a range of elements. Either the upper index or the lower may be left out, and the reverse of a list may be obtained with *$foo[len-0]*. Also, the notation *$?foo* evaluates to 1 if the variable *foo* is defined, 0 otherwise, and *$#foo* evaluates to the number of elements in *foo* if it is a list, 1 if it is a number or string, and 0 if it is a boolean variable.

History substitutions, similar to C-shell history substitutions, are also available see the C-shell manual page for all of the details.

The characters ~, {, and } have the same effects as they do in the C-Shell, i.e., home directory and alternative expansion. It is possible to use the wildcard characters *, ?, [, and] also, but only if you **unset noglob** first. This makes them rather useless for typing algebraic expressions, so you should **set noglob** again after you are done with wildcard expansion. Note that the pattern **[^abc]** will match all characters *except* **a, b,** and **c.**

IO redirection is available the symbols **>, >>, >&, >>&,** and **<** have the same effects as in the C-shell.

You may type multiple commands on one line, seperated by semicolons.

If you want to use a different **mfbcap** file than the default (usually **~cad/lib/mfbcap**), you have to set the environment variable **MFBCAP** before you start **nutmeg**. The **-m** option and the **mfbcap** variable no longer work.

SEE ALSO

sconvert(1), spice(1), mfb(3), writedata(3)

AUTHOR

Wayne Christopher (faustus@cad.berkeley.edu)

BUGS

The label entry facilities are very primitive after all, **nutmeg** isn't a graphics editor (yet). You must be careful to type very slowly when entering labels -- **nutmeg** checks the **X** event queue once every second, and can get very confused if characters arrive faster than that.

If you redefine colors after creating a plot window with X, and then cause the window to be redrawn, it will not to the right thing.

When defining aliases like

alias pdb plot db('!:1' - '!:2')

you must be careful to quote the argument list substitutions in this manner. If you quote the whole argument it might not work properly.

In a user-defined function, the arguments cannot be part of a name that uses the *plot.vec* syntax. I.e,

define poke(duck) cos(tran1.duck)

won't do the right thing.

If you type **plot all all**, or otherwise use a wildcard reference for one plot twice in a command, bad things will happen.

The **asciiplot** command doesn't deal with log scales or the **delta** keywords.

There are probably some features that **nutmeg** doesn't have yet.

CAVEATS

Often the names of terminals recognised by **MFB** are different from those in /etc/termcap. Thus you may have to reset your terminal type with the command

set term = termname

where **termname** is the name in the **mfbcap** file.

The **hardcopy** command is useless on VMS and other systems without the **plot** command, unless the user has a program that understands *plot(5)* format.

VMS NOTES

Nutmeg can be run under VAX/VMS. Some features like command, etc completion, expansion of *, ?, and [], backquote substitution, the shell command, and so forth do not work. (In fact command completion only works on 4.2 or 4.3 BSD.)

Nutmeg will look for start-up commands in the file *spice.rc* in the current directory.

The standard suffix for rawspice files in VMS is ".raw".

You will have to respond to the -*more*- prompt during plot with a carriage return instead of any key as you can do on UNIX.

Parent Directory