VCS Quick Reference(V1.0, March 6th, 2001)

Functions

Canvas, x=vcs.init()

open/clear/close	*()*	
plot	*(slab,[grid=grd],[graphic method],[template object],...)*	
mode	*= 1/0*	*automatic/manual update*
update	*()*	*updates the changes to vcs canvas*
flush	*()*	*flushes the X server*
portrait/landscape	*()*	
page	*()*	*toggles between portrait/landscape*
geometry	*(width,height,xoffsetleft,yoffsettop)*	*resize the window*
set	*('method',['name'])*	*set the default plotting graphics method*
setcolormap	*('name')*	*set the colormap*
setcolorcell	*(icell,r,g,b)*	*0 <= r,g,b <= 100*
getcolormap	*('name')*	*get the colormap 'name'*
createcolormap	*('name')*	*create a colormap named 'name'*
getcolormapname	*()*	*get the name of the colormap used*
getcolorcell	*(icell)*	*return the [r,g,b] values for color icell*
colormapgui	*()*	*pops up the colormap GUI*
animate	*()*	*pops up the animation GUI*
grid	*(dim10,dim11,...,dimn0,dimn1)*	*sets the dims(1..n) values for "out of graphics method" dimensions*
resetgrid	*()*	*undo grid()*

Output

printer	*(printername,['p'/'l'])*	*p/l: portrait/landscape*
postscript	*(filename[.ps],['p'/'l'])*	*p/l: portrait/landscape*
eps	*(filename[.ps],['p'/'l'])*	*p/l: portrait/landscape*
gif	*(filename[.gif],['a'/'r'],['p'/'l'])*	*a/r:append/replace,p/l:portrait/landscape*
cgm	*(filename[.cgm],['a'/'r'])*	*a/r: append/replace*
raster	*(filename[.ras],['a'/'r'])*	*a/r: append/replace*
pstogif	*(filename[.ps],['l'/'p'])*	*p/l: portrait/landscape*

Querying (x=vcs.init(), gm is a graphics method)

gm.list	*()*	*list the option for a vcs object and the values they're set to*
vcs.graphicsmethodtype(gm)		*returns graphic method type (boxfill,...)*
x.isgraphicsmethod	(gm)	*1 if it is a graphicsmethod, 0 if not*
x.is"vcsobjecttype"	(gm)	*1/0 "vcsobjecttype" can be any the graphics method ('boxfill', 'isofill',...) or secondary methods ('marker', 'texttable',...), or 'template'*
x.show	*('vcsobjecttype')*	*show all the vcs objects of type "vcsobjecttype" available*
x.islandscape	*()*	*1/0*
x.isportrait	*()*	*1/0*

Help (x=vcs.init(), gm is a graphics method)

x.objecthelp	*(vcs object)*	*print the help for vcs object*
vcs.help	*('function')*	*print the help for the vcs function*
gm.list	*()*	*list the option for a vcs object and the values they're set to.*
anypyobject.__doc__		*Any python object built in documentation*

Scripting

scriptrun	*('name of vcs script')*	*reads a vcs script*
scriptsave	*('name',['a/w'])*	*save the state of the system a/w: append (def)/replace*
scriptobject	*(vcs object,'name[.scr]','a/w')*	*save the object as a python script unless .scr specified, then saved as a vcs script, a/w: append (def)/replace*
saveinitialfile	*()*	*resave the VCS initial.attribute file*
vcsobject.script	*('file[.scr]','a/w')*	*save the vcs object to file, a/w: append (def)/replace*

Graphics Methods (x=vcs.init())

Creating/Retrieving/Deleting

x..create"graphicsmethod"	*('name','name to copy from')*
x.get"graphicsmethod"	*('name')*
x.removeobject	*(object)*

Common Attributes

all	
name	*= 'name'*
projection	*= 'linear', 'mollweide', 'robinson', or 'polar'*
x[y]ticlabels1[2]	*= [list]or {dic}*
x[y]mtics1[2]	*= [list] or {dic}*
datawc_x[y]1[2]	*= value # World coordinates*
x[y]axisconvert	*= 'linear', 'log10', 'ln', 'exp', or 'area_wt'*

Specific Attributes

boxfill	
level_1	*= value # Plot data greater than level_1*
level_2	*= value # Plot data lower than level_2*
color_1	*= 0 <= value <= 255 # First color in palette*
color_2	*= 0 <= value <= 255 # last color in palette*
legend	*= [list], (tuple) or {dictionary}*
ext_1	*= 'n' or 'y' # Left arrow (off/on)*
ext_2	*= 'n' or 'y' # Right arrow (off/on)*
missing	*= 0 <= value <= 255 # color of missing value*
isofill	
levels	*= [list] or (tuple) # ex [10,20,30] or [(20,30),(50,60)]*
fillareacolors	*= [list] or (tuple) 0 <= values <= 255*
fillareastyle	*= 'solid', 'hatch', or 'pattern'*
fillareaindices	*= 1 <= value <= 17 or fillarea obects*
ext_1	*= 'n' or 'y' # Left arrow (off/on)*
ext_2	*= 'n' or 'y' # Right arrow (off/on)*
missing	*= 0 <= value <= 255 # color of missing value*
isoline	
level	*= [list] or (tuple) # ex [10,20,30] or [(20,30),(50,60)]*
label	*= 'n','y' or 0,1*
line	*= line object or value in line.type (0<=value<=4)*
linecolors	*=[list] or (tuple) 0 <= values <= 255*
text	*= 0<=values<=9 or texttable or textorientation object*
textcolors	*= [values] or from the objects passed to text*
vector	
line	*= line object or value in line.type (0<=value<=4)*
linecolor	*=[list] or (tuple) 0 <= values <= 255*
scale	*= value*
alignement	*= "head", "center", "tail" (or 0, 1, 2)*
type	*= "arrows", "barbs", "solidarrows" (or 0, 1, 2)*
reference	*= value*
continents	
line	*= line object or value in line.type (0<=value<=4)*
linecolor	*=[list] or (tuple) 0 <= values <= 255*
type	*= 0 <= value <= 7 (or more if you added coninents outline)*
scatter (=xvsy with linecolor=whitecolor)	
marker	*= None or values in marker.type (1<=value<=17)*
markercolor	*=[list] or (tuple) 0 <= values <= 255*
markersize	*= 1 <= value <= 300*
xvsy/xyvsy/yxvsx	
line	*= line object or value in line.type (0<=value<=4)*
linecolors	*=[list] or (tuple) 0 <= values <= 255*
marker	*= None or values in marker.type (1<=value<=17)*
markercolor	*=[list] or (tuple) 0 <= values <= 255*
markersize	*= 1 <= value <= 300*
outfill	
fillareacolor	*= [list] or (tuple) 0 <= values <= 255*
fillareastyle	*= 'solid', 'hatch', or 'pattern'*
fillareaindex	*= 1 <= value <= 17 or fillarea obects*
outfill	*= values*
outline	
line	*= line object or value in line.type (0<=value<=4)*
linecolor	*=[list] or (tuple) 0 <= values <= 255*
outline	*= values*

Template

name	*= `name'*
file	
priority	*= [value]*
x	*= [value]*
y	*= [value]*
texttable	*= [string or texttable obj]*
textorientation	*= [string or texorientation obj]*
source/title/units	*...same...*
mean/min/max	*...same...*
comment1[2/3/4]	*...same...*
function/logicalmask/dataname	*...same...*
crdate/crtime/transformation	*...same...*
x[y/z/t]name[units/value/ x[/y][min/max]tic1[/2]	*...same...*
priority	*= [value]*
y[/x]1	*= [value]*
y[/x]2	*= [value]*
line	*= [string or line obj]*
x[/y]label1[/2]	
priority	*= [value]*
y[/x]	*= [value]*
line	*= [string or line obj]*
data	
priority	*= [value]*
x1	*= [value]*
y1	*= [value]*
x2	*= [value]*
y2	*= [value]*
line	*= [string or line obj]*
legend	*...same...*
line1[/2/3/4]	*...same...*
box1[/2/3/4]	*...same...*

Secondary Methods

textorientation (To)	
name	*= 'name'*
height	*= 1 <= value <= 100*
angle	*= 0 <= value<= 360*
path	*= 'right','left','up','down' or (0,1,2,3)*
halign	*= 'left','center','right' or (0,1,2)*
valign	*= 'top','cap','half','base','bottom' or (0,1,2*
texttable (Tt)	
name	*= 'name'*
font	*= 1 <= value <= 9*
spacing	*= -50 <= value <= 50*
expansion	*= 50 <= value <= 150*
color	*= 0 <= value <= 255*
line (Tl)	
name	*= 'name'*
type	*= 'solid', 'dash', 'dot', 'dash-dot', 'long-das or (0, 1, 2, 3, 4)*
width	*= 1 <= value <= 300*
color	*= 0 <= value <= 255*
marker (Tm)	
name	*= 'name'*
type	*= 'dot', 'plus', 'star', 'circle', 'cross', 'diam 'triangle_up', 'triangle_down','triangle_ 'triangle_right', 'square','diamond_fill', 'triangle_up_fill', 'triangle_down_fill','triangle_left_fill', 'triangle_right_fill', 'square_fill', or 0 <= 17*
width	*= 1 <= value <= 300*
color	*= 0 <= value <= 255*
fillarea(Tf)	
name	*='name'*
color	*= [list] or (tuple) 0 <= values <= 255*
style	*= 'solid', 'hatch', or 'pattern'*
index	*= 1 <= value <= 17*