PDBView DOCUMENT

Stewart A. Brown

Last Updated: 06/18/2003

Disclaimer

Introduction

PDBView Commands

o indicates functions that are heavily used
o indicates functions for more optional use

Opening, Creating, Closing, and Controlling PDB Files

o CD	Change directory
o CF	Change file
o CLOSE	Close file
o CLT	Close text file
o FILE	Print current file info
o LF	List files
o MKDIR	Make a directory
o N-ENTRIES	# file variables
o OT	Open text file
o PWD	Print working directory

Opening, Closing, and Controlling Windows

o CLV	Close viewport
o CLW	Close window

o CV	Create viewport
o CW	Create window
o CREATE-PALETTE	Create palette
o FONT	Set font
o HCV	Hardcopy viewport
o HC	Hardcopy window
o LIST-PALETTES	List all palettes
o LIST-WINDOWS	List active graphics devices
o LOAD-PALETTE	Load a palette
o NXM	Create multi-viewport window
o SAVE-PALETTE	Save a palette
o SHOW-PALETTES	Show available palettes
o VIEWPORT-AREA	Set viewport
o WU	Update window

Plotting Data

o ANIMATE	Animate a rotation
o ANNOT	Annotate a window
o AUTOPLOT	Set/get autoplot flag
o COPY-MAP	Copy display mapping
o DATA-ID	Set/get data-id flag
o DATA-REFERENCE	Mapping from index
o VR	Set default rendering
o DL	Delete a mapping
o DISPLAY-DOMAIN	Print the domain
o DR	Set mapping rendering
o DOMM	Set mapping domain
o DOMV	Set viewport domain
o GET_DOMAIN	Get domain limits
o GET-DOMAIN-NUMBER-POINTS	Get # points in domain
o GET_LABEL	Get mapping label
o GET-RANGE	Get range limits
o GET-RANGE-NUMBER-POINTS	Get # points in range
o HISTOGRAM	Histogram rendering
o LABEL-DRAWABLE	Set mapping label
o LEVELS	Set contour levels

o LIGHT-SOURCE	Set light source
o HIDE	Suppress plot
o LNCOLOR	Set line color
o LNSTYLE	Set line style
o LNWIDTH	Set line width
o LOGICAL	Logical rendering
o LSV	List viewport mappings
o MARKER-SCALE	Set marker size
o MENU	Print menu
o MOVE	Move mapping
o OVERLAY	Set/get overlay flag
o PLANE	Create hyper-plane
o PL	Plot a menu item
o PLOT	Construct mapping and plot
o PALETTE	Set default palette
o RANM	Set mapping range
o RANV	Set viewport range
o REFMESH	Set/get refmesh flag
o REFMESHCOLOR	Set/get refmesh line color
o SCATTER	Scatter plot
o SET-VECTOR-ATT	Set vector rendering
o SPAN	Create line
o UNHIDE	Undo HIDE
o VA	Set view angle

Working with Data

o NDS	Make set from array
o CHANGE	Change data values
o CHANGE-DIMENSION	Change dimensions
o CMC	Copy mapping as curve
o COPY	Copy variable(s)
o COPY-DIR	Copy directory contents
o CM	Copy mapping to file
o DEF	Define variable to workspace
o DEFINE-FILE-VARIABLE	Define new file variable

o DESC	Describe variable
o FIND	Find array values
o LS	List file variables
o LS-ATTR	List variable attributes
o LST	List workspace
o AC	Make an AC set
o CPS	Make Cartesian product set
o LR	Make LR mapping
o LRS	Synthesize LR mapping
o MINMAX	Print mim/max of variable
o PDB-COPY	Make PDB copy of file
o PRINT	Print file variable
o PM	Print mapping
o SM	Save mapping to file
o SCALE-FILE-VARIABLE	Scale file variable
o STRUCT	Display struct definition
o TABLE	Write table to text file
o TYPES	List file data types

Data Analysis

o DERIVATIVE	Derivative of 1D mapping
o INTEGRATE	Integrate mapping
o NORM	Norm of mapping
o DOMAIN FUNCTIONS	Unary operators on domain
o RANGE FUNCTIONS	Unary operators on range
o BINARY OPERATORS	Binary operators on mappings
o MATH OPERATORS	Functions on mappings
o SHIFT/SCALE OPERATORS	Shift or scale mappings

Working with PDBView

o COMMAND-LOG	Control command logging
o END	Exit PDBView
o FORMAT	Control data print formats
o HELP	Get on-line help

- o LD — Load user functions
- o MODE — Struct print control
- o SET — Set printing parameters
- o SYNONYM — Define synonyms or aliases

Disclaimer

Part of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Introduction

The PACT system provides a number of tools for working with PDB files in a fairly generic fashion. In particular, PDBView lets you interactively browse a PDB file displaying the contents graphically or textually.

PDBView is an SX program. SX is the SCHEME dialect of the LISP programming language with PACT extensions. The extensions provide functionality for graphics, binary data handling, and other areas of functionality.

PDBView has a **help** command which provides information about available commands.

PDBView Syntax

PDBView uses a slightly different data description syntax than PDBLib. With PDBLib variable names and names of members of structures cannot contain the characters: ".", "(", ")", "[", and "]". The characters "(", ")", "[", and "]" are used in array reference and dimension definition expressions. In PDBView, "(" and ")" CANNOT be used in variable reference expressions because they are special characters for PDBView. For example,

```
print a(2)
```

is illegal and results in an error. The legal expression is:

print a[2]

Other examples of legal expressions are:

print a.b[3,2]

print a.b[3][2]

print a[3].b[2][5].c

print a[3].b[2][3].c[3:4,5:50:10]

The first two forms are not identical. In the first form an element of a two dimensional array is being referenced. In the second form, the third element of the fourth array of arrays is being referenced (assuming zero based indexing in the definition of the b member of a).

For completeness and clarity in the following discussion, an index expression is defined as:

```
index expression := [index list]

index list := index | index, index list

index := simple index |

              index-min : index-max |                 (*)

              index-min : index-max : increment       (*)

simple index := integer

index-min := integer

index-max := integer

increment := integer
```

If an index expression uses either of the two starred forms it is said to be a hyper-index expression. A hyper-index expression implies more than one data item. Only the terminating index expression in a data reference may be a hyper-index expression. An index expression is said to dereference an indirection (or pointer or array). For each level of indirection a suitable index expression dereferences the desired data.

For example, this means that a variable defined as:

char **s

is said to have two levels of indirection and can have parts accessed as follows:

```
print s                        prints the entire item

print s[2]                     prints the third character array of s
```

```
print s[3][10]                prints the eleventh character
                              of the fourth character array of s
```

In the above example a zero based index is assumed.

When referring to part of a variable, especially a structured variable, the terminal node must be of primitive type or a structure containing no indirections and whose descendant members contain no indirections. Furthermore, the path to the desired part must contain one array reference for each level of indirection traversed.

Starting PDBView

On UNIX systems a shell script called **pdbview** is provided; it starts up SX which then loads the PDBView SCHEME forms. You must add a line to your **.cshrc** or **.profile** file that defines the environment variable SCHEME. This variable tells SX where to find the PDBView SCHEME files. Consult your system administrator or some other knowledgeable source to find out where these files are located on your system. If, for example, the directory **/usr/localscheme** contains the PDBView SCHEME files, add the line:

setenv SCHEME /usr/local/scheme

to your **.cshrc** or **.profile** file.

Usage: pdbview [-d] [-h] [-l command-file] [-p n] [data-file]
Options:

```
d - dump the formatted contents of the data file in ASCII to stdout

h - print execute line information to stdout

l - load SCHEME forms (e.g. PDBView commands) from a file

      (not available with -d option)

p - number of digits to use in displaying floating point numbers

      (only available with -d option)
```

Automatic or Implicit Plotting

PDBView can be used in several different contexts and in these contexts the behaviour of it with respect to refreshing the current window or viewport when new mappings are created can be quite different. Consider the following cases: (1) an interactive session doing visualization or data analysis; and (2) an interactive session that is not doing visualization but does create new mappings.

In the first case, it is very convenient for users if new data is plotted as it is available. This includes data read from file for display or data created via any of number of PDBView commands.

The autoplot flag is the important means of controlling PDBView's behaviour. By default it is on and data is automatically plotted in the current viewport of the current window as it becomes available. You can exert explicit control of what gets plotted where, but we are speaking about implicit or default behaviour here.

If you want to assert explicit control of all plotting then set the autoplot flag to off. This point can be confusing, but a little experimentation will help to familiarize you with the behavior of PDBView in these different contexts.

Demonstration Script

In order to help make sense of the sometimes bewildering variety of commands and capabilties in PDBView. We have included a PDBView demonstration script. It can be run by starting PDBView as follows:

```
pdbview -l pdbvdemo.scm
```

The demo proceeds in small sections. Before each section starts it describes in broad terms what it is about to do. It then prompts you to go ahead with the commands in that section. Each command is printed out as you would do yourself and then it is executed so that you can see its effect. In this way, you get a chance to go through the demo at your own pace and study each section without losing your way in a blizzard of output.

The demo will open several plotting windows so you may want to run it with a relatively uncluttered desktop.

PDBView Commands

In this section the PDBView commands are listed alphabetically. The command name is followed by synonyms, if any. Each command is given with a brief description of its function and its usage. The examples are intended to illustrate the various ways of invoking each command. Optional arguments are enclosed in brackets. Alternate argument forms are separated by '|'. Formal arguments in italic font are replaced by actual arguments. Command and argument keywords are in regular non-italic font.

The file most recently referred to in a **cf (change-file)** command is the current file. Most commands implicitly refer to the current file or its contents. Some commands permit another file to be explicitly specified. A file may be referred to by name or by the alias assigned when the file was opened. Commands that operate on the contents of more than one file (e.g. **copy, copy-mapping**), require that the target file be specified, while the current file is assumed to be the source.

For the sake of brevity all menu items including mappings, images, and ULTRA curves are collectively referred to as mappings. All commands that accept mappings as arguments, accept images and curves as well.

Opening, Creating, Closing, and Controlling PDB Files

A large part of PDBView has to do with browsing binary data files. It was specifically designed for PDB files, but it is extensible and the user can write functions which permit PDBView to examine many kinds of binary data files.

CD

Change the current file directory. The new directory may be specified by either a relative path or a full path.

Usage: cd [directory]

Examples:

    cd /zoo/mammals
    cd ../reptiles
    cd

CHANGE-FILE / CF

Change the current file. If it is not already open, open the file. Many commands refer to the current file or its contents by default. Any number of files may be open - only one is the current file. The mode may be: **r**, open file read only; **a**, open file read/write (default); or **w**, overwrite any existing file. If no alias is specified, **fd** is assigned, where d is an increasing decimal integer. The options available for type are determined by the output spokes you have installed. The default type is **pdb**. Files are maintained in a circular list and, if given no argument, CF changes to the next file in the list.

Procedure version of this command is change-file*

Usage: change-file [filename [mode [alias [type]]]]

Examples:

    change-file foo
    cf foo a
    cf foo w bar pdb
    cf

CLOSE

Close a file. The default is to close the current file.

Usage: close [file]

Examples:

    close
    close foo
    close f1

CLOSE-TEXTFILE / CLT

Close the opened text file.

Usage: close-textfile

Examples:

```
close-textfile
clt
```

FILE

Describe the current file. Information returned is: name; type; creation date; version of PDB that created the file; default offset; major order; types that will require conversion when read on the current host platform; header address; symbol table address; and structure char address.

Optionally, the file type alone may be requested.

Usage: file [t | type]

Example:

```
file type
```

LIST-FILES / LF

List the open files. The current file is indicated by an asterisk.

Usage: list-files

Examples:

```
list-files
lf
```

MKDIR

Create a new directory in the current file

Usage: mkdir [directory-name]

Examples: mkdir /foo/bar mkdir ../baz mkdir mydir

N-ENTRIES

Print the total number of variables, links, and directories in the current file.

Usage: n-entries

OPEN-TEXTFILE / OT

Open a text file. These are used to dump out tabular ASCII versions of entries in binary files.

Usage: open-textfile filename

Example: open-textfile foo

PWD

Print the current file directory.

Usage: pwd

Opening, Closing, and Controlling Windows

An important aspect of handling data is visualization. The foundation of visualization is the display. PDBView gives the user many options for displaying data. It supports: multiple screen windows; multiple viewports in a window; and multiple other output devices - PS, CGM, JPEG, and MPEG.

CLOSE-VIEWPORT / CLV

Close a viewport. If no viewport is specified, close the current viewport.

Usage: close-viewport [viewport [window]]

Examples:

```
close-viewport "ABC"
clv
```

CLOSE-WINDOW / CLW

Close a graphics window. If no window is specified, close the current window.

Usage: close-window [window]

Examples:

```
close-window "ABC"
clw
```

CHANGE-VIEWPORT / CV

Change the current viewport. When the NXM command has been used to make a multi-viewport window, CV is used to move the focus to a new viewport in such a window. Many commands refer to the current viewport or its contents by default. Viewports are maintained in a circular list and, if given no argument, CV changes to the next viewport in the list.

Usage: change-viewport [name [x y dx dy]]

Examples:

```
change-viewport
change-viewport "B" 0.01 0.01 0.98 0.48
cv "V2"
```

CHANGE-WINDOW / CW

Change the current window. If it is not already open, open the window. Many commands refer to the current window or its contents by default. Any number of windows may be open - only one is the current window. Windows are maintained in a circular list and, if given no argument, CW changes to the next window in the list.

Usage: change-window [title [mode type x y dx dy]]

Examples:

```
change-window
change-window "B" "COLOR" "WINDOW" 0.5 0.1 0.4 0.4
cw "bar"
cw
```

CREATE-PALETTE

Create a new palette. See the SAVE-PALETTE command for saving to disk.

Usage: create-palette name [ndims nx ny]

Examples:

```
create-palette new_palette
create-palette new_palette 2 24 8
```

FONT

Set/get the current font information.

Usage: font [face point-size style]

face - helvetica | times | courier point-size - 8 | 10 | 12 | 14 ... style - medium | bold | italic

Examples:

```
font
font helvetica 12 medium
```

HARDCOPY-VIEWPORT

Send the current graph to all open hardcopy devices. Color output is indicated by an optional argument on the first call for a given device. The optional resolution scale factor is an integer factor by which the resolution will be decreased below the full resolution of the device.

Usage: hardcopy-viewport [color | monochrome] [portrait | landscape] [resolution-scale-factor]]

Examples:

```
hardcopy-viewport
hardcopy-viewport color
hardcopy-viewport color 8
```

HARDCOPY-WINDOW / HC

Draw the contents of the current window to all open hardcopy devices. Color output is indicated by the optional keyword, color, on the first call for a given device. The resolution is decreased below full device resolution by an integer resolution scale factor, if present. Hardcopy files remain open until explicitly closed or until PDBView is terminated.

Usage: hardcopy-window [color] [resolution-scale-factor]

Examples:

```
hardcopy-window
hc color 8
```

LIST-PALETTES

List the available palettes.

Usage: list-palettes [window]

Examples:

```
list-palettes "A"
list-palettes
```

LIST-WINDOWS

List the open windows.

Usage: list-windows

Examples:

```
list-windows
```

LOAD-PALETTE

Load a palette from a file.

Usage: load-palette fname

Example:

```
load-palette foo.pal
```

NXM

If there is a current window, this command changes it to a multi-viewport window which are tiled n x m (hence the name). If there is no open window such a multi-viewport window will be created. See CW for further details.

Usage: nxm <# viewport in x> <# viewport in y> []+

Examples:

```
nxm 2 1 "A" "B"
nxm 3 2
```

SAVE_PALETTE

Write palette pname out to disk file fname.

Usage: save-palette pname fname

Example:

```
save-palette foo foo.pal
```

SHOW_PALETTES

Show the available palettes and make selection current.

Usage: show-palettes [window]

Examples:

```
show-palettes
show-palettes "B"
```

VIEWPORT-AREA

Set drawing area, in NDC, of the current viewport.

Usage: viewport-area x-min x-max y-min ymax

UPDATE-WINDOW / WU

Update all of the viewports of the specified windows. Sometimes plotting operations are done without a window update. User defined functions very commonly do this. WU lets you explicitly do the update. It also marks the completion of a picture for a non-screen device window. After an update a new frame or picture will be started.

Just typing a carriage return will implicity do a WU command.

Usage: update-window [window]+

Examples:

```
update-window "2x2"
wu
```

Plotting Data

PDBView can be used to plot data. The most common usage is to plot data from files. It has a variety of commands to let users gather diverse kinds of data from files and assemble them into plots. It also possesses the capability of creating data from scratch to be plotted.

Plotted datasets are kept in a list associated with a viewport. Each dataset or mapping is referred to by an integer index. In many of the following commands and elsewhere you will see mappings referred to by a number. The meaning is the mapping whose index is specified in the viewport implicitly or explicitly specified. The LSV command is used to examine the list of mappings in a viewport.

Mappings in files are also specified by an integer index. These can be examined by using the MENU command. There can be some confusion as to whether an index refers to a file or a viewport. Try to be mindful of this. This behaviour differs from ULTRA where the displayed curves are referred to by letters. Unfortunately, this limited one to having a maximum of 26 curves displayed at once. PDBView removes this limitation at the cost of using integer indeces in both contexts.

ANIMATE

Rotate the specified mapping thru 360 degress

Usage: animate [theta] [phi] [chi]

Examples:

```
animate 30 20 0
```

ANNOT

Add an annotation to the current window.

Usage: annot label color xmin xmax ymin ymax

Example:

```
annot "look here" blue 0.4 0.6 0.95 0.98
```

AUTOPLOT

Get/set the autoplot flag.

Usage: autoplot [off | on]

Example:

```
autoplot
autoplot off
```

COPY-MAP

Copy the specified mappings

Usage: copy-map [mapping]*

Example:

```
copy-map 12
```

DATA-ID

DATA-ID - Turn on/off the plotting of data ids.

Usage: data-id [on | off]

DATA-REFERENCE

Refer to the nth mapping in the optionally specified file. This is primarily for the benefit of those who want to write their own functions.

The procedure version is data-reference*.

Usage: data-reference n [file-name|file-alias|file-type]

VR

Specify a default rendering mode for a viewport.

Usage: default-viewport-rendering [viewport [window]] info

```
for 1d domain - 1d range mappings
info - cartesian | polar | insel

for 2d domain - 1d range mappings
info - contour [n-levels] |
       image |
       dvb |
       fill-poly |
```

```
              shaded |
              points |
              wire-frame |
              mesh

       for 2d domain - 2d range mappings
       info - vector |
              shaded |
              mesh

       for 3d domain
       info - mesh |
              shaded
```

Examples:

```
default-viewport-rendering "A" "A" polar
default-viewport-rendering contour
default-viewport-rendering contour 15
default-viewport-rendering image
default-viewport-rendering fill-poly
vr wire-frame
vr shaded
vr vector
```

DL

Delete mappings from a viewport.

Usage: delete-mapping [vname [wname]] mapping*

Examples:

```
delete-mapping 2 4 5
```

DISPLAY-DOMAIN

Plot a domain mesh in a viewport.

Usage: display-domain [viewport [window]] domain*

Example:

```
display-domain 18
```

DR

Specify the rendering mode for a drawable. This permits the user to render different datasets differently in the same viewport.

Usage: drawable-rendering drawable info

info - [spec]* spec - (render render) | - (view-angle theta phi chi) | - (change-palette palette) for 1d domain - 1d range mappings render - cartesian | polar | insel for 2d domain - 1d range mappings render - contour [n-levels] | image | dvb | fill-poly | shaded | points | wire-frame | mesh for 2d domain - 2d range mappings render - vector | shaded for 3d domain render - mesh | shaded Examples:

```
drawable-rendering 3 (render polar)
drawable-rendering 1 (render contour)
drawable-rendering 1 (render contour 15)
drawable-rendering 12 (render image) (change-palette spectrum)
drawable-rendering 18 (render fill-poly)
drawable-rendering 4 (render wire-frame) (view-angle 60 45 0)
dr 1 (render shaded) (change-palette cyans)
dr 1 (render vector)
```

DOMM

Select plotting limits for the domain of a mapping. If invoked with no limit values the domain will default to that implied by the data set.

Usage: domm [viewport [window]] mapping x1_min x1_max ...

Examples:

```
domm 1 -5.0 10.0 10 20
domm 1
```

DOMV

Select plotting limits for the domain of a viewport. If invoked with no limit values the domain will default to that implied by the data sets.

Usage: domv [viewport [window]] x1_min x1_max ...

Examples:

```
domv -5.0 10.0 10 20
domv
```

GET-DOMAIN

Get the domain of the given mapping.

Usage: get-domain mapping

Examples:

```
get-domain 2
```

GET-DOMAIN-NUMBER-POINTS

Get the domain of the given mapping.

Usage: get-domain-number-points mapping

Examples:

```
get-domain-number-points 2
```

GET-LABEL

Get the domain of the given mapping.

Usage: get-domain-number-points mapping

Examples:

```
get-domain-number-points 2
```

GET-RANGE

Get the range of the given mapping.

Usage: get-range mapping

Examples:

```
get-range 2
```

GET-RANGE-NUMBER-POINTS

Get the range of the given mapping.

Usage: get-range-number-points mapping

Examples:

```
get-range-number-points 2
```

HISTOGRAM

Display given mappings from current viewport as histograms.

Usage: histogram on | off | left | right | center [integer]+

Examples:

```
histogram on 1 2
```

```
histogram left 3
```

LABEL-DRAWABLE

Label a drawable.

Usage: label-drawable [vname [wname]] mapping

Examples:

```
label-drawable 2
```

LEVELS

Set the contour levels

Usage: levels [clev]*

Examples:

```
levels 1.0 1.2 1.5 1.9 2.5 5.8 10.3
```

LIGHT-SOURCE

Set the theta and phi angles for the light source. This is used in the rendering of surface of 3D renderings.

Usage: light-source theta phi mapping*

Examples:

```
light-source 30 -20 1
```

HIDE

Don’t plot the specified mappings. They remain on the list for the viewport and can be made visible with subsequent unhide commands.

Usage: hide [integer]+

Examples:

```
hide 1 3 19
```

LNCOLOR

Set the line color used in drawing the specified mappings.

Usage: lncolor color [integer]+

Examples:

```
lncolor blue 1 3 19
lncolor red 2 5 7
```

LNSTYLE

Set the line style used in drawing the specified mappings.

Usage: lnstyle style [integer]+

Examples:

```
lnstyle dotted 1 3 19
lnstyle dashed 2 5 7
```

LNWIDTH

Set the line width used in drawing the specified mappings.

Usage: lnwidth width [integer]+

Examples:

```
lnwidth 1.5 1 3 19
lnwidth 2.0 2 5 7
```

LOGICAL

Display given mappings from current viewport versus index.

Usage: logical on | off [integer]+

Examples:

```
logical on 1 3 19
logical off 2 5 7
```

LSV

List the mappings or images associated with a viewport.

Usage: list-mappings-in-viewport [vname [wname]]

Examples:

```
lsv
```

```
lsv "V2"
lsv "V2" "2x1"
```

MARKER-SCALE

Set the marker scale for the specified mappings.

Usage: marker-scale scale [integer]+

Examples:

```
marker-scale 1.8 1 3 19
```

MENU

List the labels of mappings in the current file directory. The ordinals preceding the labels are used to reference the mappings in other commands. An optional selection pattern may be specified. Each '?' in the pattern matches any single character in a label. Each '*' in the pattern matches any string of zero or more characters. Note that the pattern, if supplied, must match the entire label. Liberal use of "*"

Usage: menu [pattern]

Examples:

```
menu
menu ?*d
menu *foo*
```

MOVE

Move items from the current viewport and window to the named viewport and window.

Usage: move vname wname item*

Example:

```
move "V0" "B" 1 2 3
move "V2" A 3
```

OVERLAY

Get/set the overlay flag. If on, the overlay flag plots all mappings in the current window viewport in a common graphical viewport. Otherwise the mappings will have their own graphical viewport and won't in general register with one another unless they have the same rendering mode.

Usage: overlay [off | on]

Example:

```
overlay
overlay off
```

PLANE

Create and draw a hyper plane.

The proceure version is plane*.

Usage: plane c0 (c1 x1min x1max npts1) ...

Examples:

```
plane 1.0 (2.0 -5.0 5.0 100)
plane 0.0 (2.0 -5.0 5.0 10) (0.5 -5.0 5.0 10)
```

DISPLAY-MAPPING / PL

Plot the specified mappings. Mappings are referenced by the numbers displayed by the menu command or by the names of variables containing mapping information.

Usage: display-mapping mapping-list

Examples:

```
display-mapping 5
pl dir1/Mapping8
pl 5 8
```

PLOT

Plot one variable or part of a variable (range) against another (domain). Plot versus index (logical) if no domain given.

Procedure version is plot*.

Usage: plot range [domain]

Examples:

```
plot y x
plot z
plot dir1/ddd[0:60, 0:15, 20]
plot log10 y x
plot log10 z (cps log10 x log10 y)
plot (nds x 10 15)
```

CHANGE-PALETTE / PALETTE

Set the current color palette.

Usage: change-palette standard | spectrum | rainbow | bw

Example:

```
palette rainbow
```

RANM

Select plotting limits for the range of a mapping. If invoked with no limit values the range will default to that implied by the data set.

Usage: ranm [viewport [window]] mapping x1_min x1_max ...

Examples:

```
ranm 1 -5.0 10.0 10 20
ranm 1
```

RANV

Select plotting limits for the range of a viewport. If invoked with no limit values the range will default to that implied by the data sets.

Usage: ranv [viewport [window]] x1_min x1_max ...

Examples:

```
ranv -5.0 10.0 10 20
ranv
```

REFMESH

Overlay the plot of a mapping with its mesh iff on.

Usage: refmesh on | off

REFMESHCOLOR

Set the line color for the reference mesh.

Usage: refmeshcolor color

SCATTER

Display given mappings from current viewport as scatter plots.

Usage: scatter on | off | plus | star | triangle [integer]+

Examples:

```
scatter on 1 3 19
scatter star 2 5 7
```

SET-VECTOR-ATT

Set vector plotting attributes.

Usage: set-vector-att scale | headsize | color value

Examples:

```
set-vector-att 0.1 0.2 red
```

SPAN

Create a line with slope 1 and intercept 0. This a restricted special case of the PLANE command.

Usage: span xmin xmax

Example:

```
span 0 10
```

UNHIDE

Do plot the specified mappings. This undoes what the HIDE command does.

Usage: unhide [integer]+

Examples:

```
unhide 1 3 19
```

VIEW-ANGLE / VA

Set the three Euler viewing angles. The view angle is a rotation by phi about the z axis starting from the negative y axis counterclockwise followed by a rotation by theta about the x' axis counterclockwise and followed by a rotation by chi about the z'' axis again starting from the negative y'' axis counterclockwise. Angles are measured in degrees.

Usage: view-angle theta phi chi

Examples:

```
va 20 30 0
```

Working with Data

PDBView allows the user to manipulate various kinds of data in non-graphical ways. Data in files can be perused, manufactured, or modified. Data types can also be defined and be used to add new data to a file.

ARRAY-ND-SET

Construct a set from an array which has specific dimensions.

Usage: array-nd-set var [dim]*

Example:

```
array-nd-set x 10 20
```

CHANGE

Reset a single value in a variable or structure member. Note that the command keyword may be omitted. To change an array element, qualify the name with index expressions. (See the PDBView Syntax section for a discussion of index expressions).

This command has an implicit form in which the "change" can be omitted. The implicit form can be used at the command prompt when the file variable name does not conflict with a PDBView command name.

Usage: [change] variable | structure-member value

Examples:

```
change a[10,15] 3.5
change time 0.0
x[1:2] 1 2
a[5,10,3] 5.7e4
dir1/jkl.k 2
```

CHANGE-DIMENSION / CHDIM

Change the dimensions of a variable in memory.

Usage: change-dimension name dimension_list

Examples:

```
change-dimension foo 20
change-dimension bar 10 10
change-dimension foobar (2 . 10) (3 . 5)
```

CMC

Copy 1d mappings from the current file to another file as ULTRA curves. Mappings are referenced by the numbers displayed by the menu command. If the mapping list is *, copy all 1d mappings in the current directory.

Usage: cmc file mapping-list

Examples:

```
cmc foo *
cmc bar 1 5
```

COPY

Copy variables from the current file to another file. If variable list is *, copy all variables in current directory.

Usage: copy file variable-list

Procedure version is copy*

Examples:

```
copy foo *
copy f3 bird cat dog
```

COPY-DIR

Copy the contents of the specified directory from the source file to the destination file. The directory name will be the same in both files.

Usage: copy-dir dir src-file dst-file

Example:

```
copy-dir "/a/b" f0 f1
```

COPY-MAPPING / CM

Copy mappings from the current file to another file. Mappings are referenced by the numbers displayed by the **menu** command. If the mapping list is *, copy all mappings in the current directory.

Usage: copy-mapping file mapping-list

Examples:

```
copy-mapping foo *
copy-mapping bar 1 5
```

DEF

Install an object in the workspace under the given name.

Usage: def name object

Example:

```
def foo m3
```

DEFINE-FILE-VARIABLE

Create a new variable in a PDB file

Procedure version is define-file-variable*.

Usage: define-file-variable type name [values]*

Examples:

```
define-file-variable double foo 3.42
define-file-variable "char *" bar "Hi" "there"
define-file-variable "int" baz 1
```

DESC

Describe variables or structure members in the current file directory. To get a description of part of a variable or structure member, qualify the name with index expressions. (See the PDBView Syntax section for a discussion of index expressions.) If a selection pattern is specified in place of a variable name, all variables matching the pattern will be described. Each '?' in the pattern matches any single character. Each '*' in the pattern matches any string of zero or more characters. An optional type qualifier may also be specified in order restrict the list to a given type.

Procedure version is desc*.

Usage: desc variable | structure-member | pattern [type]

Examples:

```
desc Mapping1
desc dir1/a.b.c[12:14]
desc * double
desc var? integer
```

FIND

Find and return the indeces in an array of numbers which meet the criteria specified.

This command has an implicit form in which the "find" can be omitted. The implicit form can be used at the command prompt when the file variable name does not conflict with a PDBView command name.

Usage: [find] arr [predicate # [conjunction]]*

arr := a pm-array of values
predicate := = | != | <= | < | >= | >
conjuntion := and | or

Examples:

```
find x < 1 and > 0.5
find x > 1 or < 0.5
x > 0.1 and < 0.75 or > 1.25 and < 1.75
x < 0.15 or > 0.75 and < 1.25 or > 1.75
```

LS

List the names of variables, links, and directories in the current file directory. Directories have a terminal slash. An optional selection pattern may be specified. Each '?' in the pattern matches any single character. Each '*' in the pattern matches any string of zero or more characters. An optional type qualifier may also be specified in order restrict the list to a given type.

Usage: ls [pattern [type]]

Examples:

```
ls dir1/curve*
ls var? integer
ls * Directory
```

LS-ATTR

List the attributes in the current file.

Usage: ls-attr

Examples:

```
ls-attr foo
```

LST

List the labels of mappings in the work space. A selection pattern may be specified.

Usage: lst [pattern]

Examples:

```
lst
lst ?*d
lst *foo*
```

MAKE-AC-MAPPING-DIRECT / AC

Construct an arbitrarily connect mapping from component sets.

Usage: make-ac-mapping-direct ran dom [centering [label]]

Example:

```
ac ran dom
ac ran dom ncent "my label"
```

MAKE-CP-SET / CPS

Construct a set which is the Cartesian product of a list of sets or arrays. Unary operations can optionally be applied to the components.

Usage: make-cp-set [item]* item := var | oper var

Example:

```
cps x y z
cps log10 x y exp z
```

MAKE-LR-MAPPING-DIRECT / LR

Construct a logically rectangular mapping from component sets.

Usage: make-lr-mapping-direct ran dom [centering [existence-map [label]]]

Example:

```
lr ran dom zcent nil "some data"
```

MAKE-LR-MAPPING-SYNTH / LRS

Construct a logically rectangular mapping by synthesis from lower dimensional datasets.

Usage: make-lr-mapping-synth ran dom ...

Example:

```
lrs '("rng" 0 (td23 td24 td25 td26 td27 td28 td29
               td30 td31 td32))
    '("dmn" 1 (td0 td0 td0 td0 td0 td0 td0
               td0 td0 td0))
```

MINMAX

Print the minimum and maximum values and their offsets from the beginning of an array in a file.

Usage: minmax name

Example:

```
minmax foo
```

PDB-COPY

Make a PDB file whose contents match the current file which may not be a PDB file.

Usage: pdb-copy file

Example:

```
pdb-copy file.cdf
```

PRINT

Print out all or part of the specified variable or structure member. Note that the command keyword may be omitted. To print part of a variable or structure member, qualify the name with index expressions. (See the PDBView Syntax section for a discussion of index expressions).

This command has an implicit form in which the "print" can be omitted. The implicit form can be used at the command prompt when the file variable name does not conflict with a PDBView command name.

Usage: [print] variable | structure-member

Examples:

```
print Mapping4.domain.elements
print dir1/Mapping2.range.elements[1]
```

```
print a.b[3].c[5,10:20,1:8:3]
Mapping2
a[5,10:20,1:8:3]
```

PRINT-MAPPING / PM

Print out the specified mappings. Mappings are referenced by the numbers displayed by the menu command.

Usage: print-mapping mapping-list

Example:

```
print-mapping 3 4
```

SAVE-MAPPING / SM

Save mappings from the current viewport to a file. Mappings are referenced by the numbers displayed by the lsv command. If the mapping list is *, save all mappings in the current viewport.

Usage: save-mapping file mapping-list

Examples:

```
save-mapping foo *
save-mapping bar 1 5
```

SCALE-FILE-VARIABLE

Scale variable name by scale_factor and write it into the current file with name aliasname.

Usage: scale-file-variable name aliasname scale_factor

Examples:

```
scale-file-variable foo two_foo 2.0
scale-file-variable bar -bar -1.0
```

STRUCT

Describe the specified data type in the current file.

Usage: struct data-type

Examples:

```
struct double
struct PM_mapping
```

TABLE

Write out all or part of a variable or member structure to an opened textfile To write part of a variable or member qualify the name with index expressions whose parts are in one of the three forms:

index

index-min:index-max

index-min:index-max:increment

Only the first form may be used to qualify variables or terminal structure members with embedded pointers and non-terminal members.

Procedure version is table*.

Usage: table variable | structure-member

Examples:

```
table Mapping2
table Mapping4.domain.elements
table Mapping2.range.elements[1]
table a[5,10:20,1:8:3]
table a.b[3].c[5,10:20,1:8:3]
```

TYPES

List the data types in the current file.

Usage: types

Data Analysis

In a more graphical context, PDBView offers the user many tools for data analysis. These include algebraic manipulations, calculus, and a wide variety of elementary functions (e.g. sin, exp, j0).

DERIVATIVE / DER

Take simple derivative of scalar function of one variable.

Usage: derivative mapping*

Examples:

```
der 2
der 4 6:9
```

INTEGRATE / INT

Integrate the specified mapping.

Usage: integrate mapping*

Examples:

```
integrate 2
integrate 4 6:9
```

NORM

Take the Euclidean norm of the range values of a mapping.

Usage: norm mapping*

Examples:

```
norm 2
norm 4 6:9
```

DOMAIN FUNCTIONS

The following functions are appplied to all components of the domain values of the given mappings:

```
Absolute value              absx
Logarithm and Expontional   lnx, log10x, expx
Trigonometry                cosx, sinx, tanx
                            acosx, asinx, atanx
Hyperbolic Functions        coshx, sinhx, tanhx
Bessel Functions            j0x, j1x
                            y0x, y1x
Others                      sqrx, sqrtx, recipx
```

Usage: fnc mapping*

Examples:

```
sqrx 2
lnx (j1x 3)
```

RANGE FUNCTIONS

The following functions are appplied to all components of the range values of the given mappings:

```
Absolute value              abs
Logarithm and Expontional   ln,log10,exp
Trigonometry                cos, sin, tan
                            acos, asin, atan
Hyperbolic Functions        cosh, sinh, tanh
Bessel Functions            j0, j1
                            y0, y1
Tchebyshev Functions        tchn
Others                      sqr, sqrt, recip, random
```

Usage: fnc mapping*

Examples:

```
sin 2
cos (log10 3)
```

BINARY OPERATORS

The following binary functions combine mappings two at a time:

```
m+      Sum
m*      Product
m-      Difference
m/      Quotient
hypot   sqrt(a^2 + b^2)
```

The result is a new mapping.

Usage: oper [mapping]*

Examples:

```
m+ 1 3 5 9
m/ 2 8 19
hypot 3 17
```

MATH OPERATORS

The following functions operate on mappings using auxilliary information:

```
powrx, powx   Raise domain values to a power x = x^a
powax         Raise a to the power of the x value x = a^x

powr, pow     Raise range values to a power y = y^a
powa          Raise a to the power of the y value y = a^y

jn            Nth order Bessel function of the first kind on range values
yn            Nth order Bessel function of the second kind on range values
```

```
smooth        Smooth mapping using n point integral to average
xmm           Excerpt part of a mapping
```

These do not result in a new mapping.

Usage: oper [info] [mapping]*

Examples:

```
powax 3.0 8
jn 2 2 10 56
smooth 3  1 18
xmm '(1.0 3.0 4.0 18.0) 25
```

SHIFT/SCALE OPERATORS

The following functions shift or scale the specified mappings

```
dx      Shift domain values of curve by a constant
dy      Shift range values of curve by a constant
mx      Scale domain values of curve by a constant
my      Scale range values of curve by a constant
divx    Divide domain values by a constant
divy    Divide range values by a constant
```

These do not result in a new mapping.

Usage: oper [info] [mapping]*

Examples:

```
dx -2.5 1 5 9
divy 3.14 18
```

Working with PDBView

In a more graphical context, PDBView offers the user many tools for data analysis. These include algebraic manipulations, calculus, and a wide variety of elementary functions (e.g. sin, exp, j0).

COMMAND-LOG

Turn logging of typed commands on or off. A log file name may be specified in place of the keyword **on**. The default log file name is **pdbview.log**. If the log file already exists, it is appended to. Logging is off by default.

Usage: command-log [on | off | filename]

Examples:

```
command-log on
command-log mylog2
```

END / QUIT

End the session of PDBView.

Usage: end

FORMAT

Set the printing format for a specified data type. If the argument specifying the type has '1' or '2' appended, then the new format is applied only to arrays shorter than **array-length** or to arrays longer than or equal to **array-length**, respectively. Otherwise, the format applies to both. Invoking the format command with the single argument, default, causes the formats for all types to be reset to their default values. The format argument must be a Standard C I/O Library format string. Double quotes are only necessary if the format string contains embedded blanks. See the set command for more information about the **array-length** variable. This command overrides the settings of the decimal-precision and bits-precision display control parameters.

Usage: format integer[1 | 2] | long[1 | 2] | float[1 | 2] |

```
double[1 | 2] | short[1 | 2] | char[1 | 2] format
```

Usage: format default

Examples:

```
format double %12.5e
format double2 %10.2e
format char "%s "
```

HELP

Print a list of commands or documentation for an optionally specified command.

Usage: help [command]

Examples:

```
help
help menu
```

LD

Read SCHEME forms (e.g. PDBView commands) from the specified ASCII disk file. The -l execute line option can be used to cause PDBView to read a file of SCHEME forms at start-up.

Usage: ld filename

Example:

```
ld script.scm
```

MODE

Set the print mode for structures.

Display modes are:

```
full-path - the full path name is printed at each branch, e.g. foo.ba
indent - indent 4 spaces at each branch (default)
tree - display as a tree (lines connecting branches)
```

Type display is controlled by:

```
no-type - turns off the display of types (default)
type - displays the type of each item and branch
```

Display of recursive structures is controlled by:

```
recursive - indent each level of recursive structures
iterative - number each level of recursive structures (default)
```

Usage: mode full-path | indent | tree | no-type | type | recursive | iterative

Example:

```
mode full-path
```

SET

Set the value of some array display parameters. Parameters:

```
line-length - the number of array elements per line
array-length - for arrays shorter than value, label each element indi
bits-precision - number of mantissa bits of precision
decimal-precision - number of digits of precision
```

Usage: set line-length | array-length | bits-precision | decimal-precision value

Examples:

```
set line-length 3
set decimal-precision 6
```

SYNONYM

Define synonyms for the given function.

Usage: synonym func [synonym]*

Examples:

```
synonym change-file cf
```

Default State

The state in which PDBView starts out is:

- mode iterative
- mode indent
- mode no-type
- set array-length 6
- set line-length 3
- set decimal-precision 4
- palette spectrum
- command-log off

Other PACT Documents

Interested readers may wish to refer to the other PACT documents which describe the data structures and functionality underlying the more common PDB files upon which PDBView operates. The PANACEA, PGS, and PDBLib manuals are of special interest to people who wish to generate and view data files with PACT.

The list of PACT Documents is:

PACT User's Guide	UCRL-MA-112087
SCORE User's Manual	UCRL-MA-108976 Rev.1
PPC User's Manual	UCRL-MA-108964 Rev.1
PML User's Manual	UCRL-MA-108965 Rev.1
PDBLib User's Manual	M-270 Rev.2
PGS User's Manual	UCRL-MA-108966 Rev.1

PANACEA User’s Manual	M-276 Rev.2	
ULTRA II User’s Manual	UCRL-MA-108967 Rev.1	
PDBDiff User’s Manual	UCRL-MA-108975 Rev.1	
PDBView User’s Manual	UCRL-MA-108968 Rev.1	Current Document
SX User’s Manual	UCRL-MA-112315	

Send questions and comments to rodrigues2@llnl.gov