Glossary.hyper

COLLABORATORS

	TITLE : Glossary.hyper		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Glossary.hyper

1.1 Glossary (Wed Jul 15 15:32:26 1992)

Contents:

contents operator

crash node

current debug node

current list

current list indicator

current logical window

current tag list

debug nodes

debug tasks

double quotes

dummy debug task

expression

fancy mode

FD-files

feedback mode

fullscreen debugger

function definitions

function monitor

functions

group operator

history buffer

hold mode

home position

hot key

interrupt key

IntuiTick

key attacements

key code

led monitor

linenumber operator

list

list element

list operator

log file

logical window

LW

machinelanguage scripts

macro

masterbox

ML-scripts

MMU tree

monitor functions

MORE checking

names

nofancy mode

normal breakpoints

output log

pause key

pen

physical window

PortPrint

PowerVisor device

PowerVisor screen

PowerVisor scripts

PowerVisor startup file

PowerVisor window

prefix operators

private breakpoints

profiler breakpoint

prompt

PVDevice

PVSD file

PW

qualifier

quote operator

quotes

real-top windows

recursive aliases

resident breakpoints

resident commands

resident ML-scripts

scripts

single quotes

singlestep mode

size bar

snapping

special operator

special variables

stack checking

stack fail level

standard logical window

startup file

string expansion

string pointers

strings

structure definition

strong quote

symbols

tag

tag file

tag list

task accounting

Task Control Block (or TCB)

task list

TCB

templates

temporary breakpoint

timeout breakpoints

top-visible windows

variables

Various:

Back to main contents

1.2 Glossary : abbreviations

PowerVisor allows you to abbreviate several things. You can abbreviate command names and names for list elements

Tutor chapters : Expresions Getting Started

1.3 Glossary : active logical window

The active logical window is the logical window where you can ↩ scroll
with the keyboard. You can see if a logical window is active by looking
at the titlebar. A full (blue in AmigaDOS 2.0) titlebar indicates that
the logical window is active. Note that the active logical window is NOT
the same as the current logical window.
Starting at V1.10, the active logical window is also used for pending
input (see the Screens and windows chapter for more info)

Commands : scroll active
Tutor chapters : Screens and windows
Related terminology :

```
                current logical window

                logical window
```

1.4 Glossary : address operator

```
                  The address operator ('&') can be used to ask the address of an ↩
                      element
   in the current list. You can only use this operator for the  exec ,
    graf  and  intb  lists. An address operator always precedes a
   list operator

      Tutor chapters :  Expressions
      Related terminology :
                contents operator

                list operator
```

1.5 Glossary : aliases

```
                  An alias in its simpliest form is another name for a command.  ↩
                     PowerVisor
   aliases are a bit more powerful than normal aliases. You can actually
   construct whole new commands with them

      Commands :  alias   unalias
      Tutor chapters :  Installing PowerVisor
      Related terminology :
                alias string

                recursive aliases
```

1.6 Glossary : alias string

```
                  The alias string is the string that is used instead of the  ↩
                     commandline
   when an alias command is entered. It is in fact the definition of the
   alias

      Commands :  alias   unalias
      Tutor chapters :  Installing PowerVisor
      Related terminology :
                aliases
```

1.7 Glossary : ARexx port

```
                   The ARexx port is used by ARexx to send commands to. The name  ↩
                      of
   the PowerVisor ARexx port is REXX_POWERVISOR

      Commands :  rx
      Tutor chapters :  Scripts
      Related terminology :
                 ARexx scripts
```

1.8 Glossary : ARexx scripts

```
                   An ARexx script is a ASCII script file containing ARexx  ↩
                      commands.
   ARexx is a versatile script language for the Amiga. With ARexx you
   can interface PowerVisor to any other program supporting ARexx or
   you can write powerful scripts making life easier for you and other
   users of PowerVisor. ARexx scripts must begin with a comment
   (/* */)

      Commands :  rx
      Tutor chapters :  Scripts
      Related terminology :
                 PowerVisor scripts

                 scripts

                 ML-scripts

                 ARexx port
```

1.9 Glossary : auto output snap

```
                   This feature is an optional setting for logical windows. When ' ↩
                      auto
   output snap' is on, the logical window will automatically scroll to
   the place where output appears. 'auto output snap' is on by default
   for the 'Main' logical window. You can change this behaviour with
   the  setflags  or  prefs  commands

      Commands :  setflags   prefs
      Tutor chapters :  Screens and windows
      Related terminology :  logical windows
                 MORE checking

                 home position
```

1.10 Glossary : autodefault

```
                Autodefault is a method provided to make life easier for the ←↩
                    user.
Normally when PowerVisor parses some sequence of characters, several
steps occur. In one of these steps PowerVisor tests if the sequence
of characters could be an abbreviation for a name of a 'list element'
in the current list.
But some commands are only useful for specific list elements. For
example : the  freeze  command to freeze a task is only useful for
tasks. Therefor 'freeze' uses 'autodefault' to the 'task' list. This
means that the parsing of the sequence of characters is not done for
the current list but for the task list. In practice this means that
you can simply use the name of the task even if the  task  list is
not the current list. Many commands use this feature (see the
 Command Reference  chapter if you want to know if a certain command
uses 'autodefault'). Some functions also use 'autodefault'

   Related terminology :
              list element

              current list
```

1.11 Glossary : autoscalable

```
                When a logical window is autoscalable for one or both  ←↩
                    directions
(vertical or horizontal) the visible size will always be equal to
the real size (in that direction) (see 'logical window' for more info).
This means that when you change the visible size (by opening or closing
another logical window or by dragging the size bar) the logical window
will be cleared and the real size will be recomputed. The 'Main' logical
window is NOT autoscalable by default

   Commands :  fit   colrow
   Tutor chapters :  Screens and windows
   Related terminology :
              logical window

              box
```

1.12 Glossary : box

```
                A box is used by the physical window to manage the space for ←↩
                    logical
windows. A box has a parent (unless it is the root box for the
physical window) and two children (unless it is a leaf of the box
tree). A box with no children (a leaf box) corresponds with a
logical window.

   Tutor chapters :  Screens and windows
   Related terminology :
              logical window
```

physical window

masterbox

1.13 Glossary : breakpoint node

A breakpoint node is the internal data structure used by ↩
PowerVisor
to hold information about a breakpoint. Each breakpoint has its
own breakpoint node. You can use info for a debug node to see
a list of all breakpoint nodes

Commands : info debug
Tutor chapters : Debugging
Related terminology :
breakpoints

debug nodes

1.14 Glossary : breakpoints

A breakpoint is a location in memory where a debug node should ↩
stop
(sometimes depending on some condition). They are implemented using
'ILLEGAL' instructions (so don't use them in shared memory)

Commands : break trace debug
Tutor chapters : Debugging
Related terminology :
debug nodes

normal breakpoints

timeout breakpoints

conditional breakpoints
temporary breakpoints profile ↩
breakpoints

resident breakpoints

private breakpoints

1.15 Glossary : code

See
key code

1.16 Glossary : commandline

```
                The commandline of PowerVisor is a stringgadget. This means ←
                   that you
can use all stringgadget editing facilities supported by the
operating system. You can also use some extra facilities provided
by PowerVisor like the 'history buffer'.
By default the commandline is 400 bytes long but you may decrease
or increase this value with the  prefs  command

   Commands :  prefs
   Tutor chapters :  Getting Started
   Related terminology :
              history buffer
```

1.17 Glossary : conditional breakpoints

```
                Conditional breakpoints only break when a certain condition ( ←
                   expression)
is true

   Commands :  break   trace   debug
   Tutor chapters :  Debugging
   Related terminology :
              breakpoints
               debug node
```

1.18 Glossary : conditional expressions

```
                Using the  if()  function you can make conditional expressions  ←
                   like in
C

   Tutor chapters :  Expressions
   Related terminology :
              expression
```

1.19 Glossary : constants

```
                A constant is just like a PowerVisor variable. The only  ←
                   difference
is that you can't change the value (obvious :-)
'version' is the only constant in the current version of PowerVisor

   Commands :  vars
   Tutor chapters :  Expressions
   Related terminology :
              variables
```

```
                functions

                special variables
```

1.20 Glossary : contents operator

```
                The contents operator ('*') can be used to read from memory ↩
                   locations.
You can use it in expressions or before the assignment operator

   Tutor chapters :  Expressions
   Related terminology :
              address operator
```

1.21 Glossary : crash node

```
A crash node (in the  crsh  list) corresponds with a crashed task.
When a task crashes and PowerVisor traps the crash, PowerVisor will
create a crash node and halt the task. This crash node contains
some extra information about the crash
```

1.22 Glossary : current debug node

```
The current debug node is the debug node that you are currently
debugging. All trace and breakpoint commands use the current debug
node. You can have more than one debug node in memory at the same
time

   Commands :  with   duse
   Tutor chapters :  Debugging
   Related terminology :  debug node
```

1.23 Glossary : current list

```
                When PowerVisor parses some sequence of characters, several
steps occur. In one of these steps PowerVisor tests if the sequence
of characters could be an abbreviation for a name of a 'list element'
in a certain list. This list is the current list. In the bottom
left corner of the PowerVisor window you can see the name of the
current list. If you want to change the current list to some other
list, just type the name of the other list ( task ,  lock ,
 wins , ...). Note that some commands do not use the current list for
parsing sequences of characters (see 'autodefault')

   Commands :  task   lock   wins  ...
   Tutor chapters :  List Reference
   Related terminology :
```

```
              list

              list element

              autodefault

              current list indicator
```

1.24 Glossary : current list indicator

```
                The current list indicator is located in the left-bottom corner ↩
                    of
  the 'Main' physical window (the PowerVisor window). It displays
  the name of the current list ('Task', ...).
  Right from the current list indicator is the prompt

     Related terminology :
              current list

              prompt
```

1.25 Glossary : current logical window

```
                The current logical window is the logical window that receives  ↩
                   all
  output from most commands. This is 'Main' by default. Note that the
  current logical window is NOT the same as the active logical window

     Commands :  current   on
     Tutor chapters :  Screens and windows
     Related terminology :
              active logical window

              logical window
```

1.26 Glossary : current tag list

```
                The current tag list is the tag list that is used by all the
  tag commands. There are sixteen possible tag lists (0..15)

     Commands :  tg   usetag
     Tutor chapters :  Looking at things
     Related terminology :
              tag list

              tag
```

1.27 Glossary : debug nodes

See
debug tasks

1.28 Glossary : debug tasks

(or debug nodes) A debug task is a task you are debugging. When it
is called a debug task, we are talking about the real task. When
it is called a debug node (which is logically the same) we are also
talking about the task in most cases, but sometimes the term debug
node is used for the internal structure needed by PowerVisor to
debug the task (this last meaning is actually more exact than the
first one). All debug nodes are in the dbug list

Commands : dbug debug
Tutor chapters : Debugging

1.29 Glossary : double quotes

See
quotes

1.30 Glossary : dummy debug task

With a dummy debug task you can make symbols without having to ↩
create
a real debug task. This makes it more easy to disassemble programs.
You can't trace or set breakpoints with a dummy debug task, but you
can create and show symbols

Commands : debug symbol
Tutor chapters : Debugging
Related terminology :
debug nodes

1.31 Glossary : expression

A sequence of characters corresponding with some algebraic or other
operations on integers

Tutor chapters : Expressions

1.32 Glossary : fancy mode

```
                Fancy mode is another name for two-bitplane mode. PowerVisor is
in fancy mode when the PowerVisor screen uses two bitplanes (default).
Otherwise PowerVisor is in 'nofancy' mode

   Commands :  mode
   Tutor chapters :  Screens and windows
   Related terminology :
              nofancy mode
```

1.33 Glossary : FD-files

```
                A fd-file (function definition file) contains definitions for ↩
                    the
functions defined in a shared library (you can find these files
on the Extras 1.3 disk). PowerVisor uses these files for several
purposes (to make the disassembly more readable when you are
debugging for example, the  addfunc  command also uses FD-files).
All loaded FD-files are in the  fdfi  list

   Commands :  loadfd   unloafd   fdfi
   Related terminology :
              function definitions
```

1.34 Glossary : feedback mode

```
When feedback mode is on (default) PowerVisor will first print each
command on the PowerVisor window before executing it. That way you
have an easy way to know which command caused which output

   Commands :  mode
   Tutor chapters :  Getting Started
```

1.35 Glossary : fullscreen debugger

```
Normally debugging in PowerVisor is commandline based. Using the
 fdebug  alias (or the 'db' script) you can install a fullscreen
debugger. This fullscreen debugger uses the 'Debug' logical window
for output

   Commands :  debug   fdebug   dwin
   Tutor chapters :  Debugging
```

1.36 Glossary : function definitions

```
                A function definition is a definition for a library function
(the name, registers, ...). Function definitions are part of
an FD-file

   Commands :  loadfd   libinfo   libfunc
   Related terminology :
              FD-files
```

1.37 Glossary : function monitor

```
                The function monitor is the device in PowerVisor that monitors
library functions (monitor functions)

   Commands :  addfunc
   Related terminology :
              monitor functions
```

1.38 Glossary : functions

```
                A function is a routine you may include in expressions. It has
some arguments (or none) and most often a result. Note that PowerVisor
always expects brackets after the function name even if there are no
arguments (like in C). Functions live in the same list as variables.
This is why you can't have a variable with the same name as a function

   Commands :  vars
   Tutor chapters :  Expressions
   Related terminology :
              variables

              constants

              special variables
```

1.39 Glossary : group operator

```
                The group operator ('{' ... '}') can be used to group several  ←
                   commands
together. You can do this because you want to execute several commands
at once or because you are interested in the return code of the last
executed command in the list. Group operators can be used as stand
alone command or in expressions. The commands in the group are seperated
by ';'.
The group operator is also useful if you want to create recursive
aliases

   Tutor chapters :  Expressions
   Related terminology :
              recursive aliases
```

1.40 Glossary : history buffer

```
                To make editing easier PowerVisor supports a history buffer. ←-
                   Using
the arrow up/down keys you can move in this history buffer and make
changes to previous commands.
You can change the maximum number of lines in the history buffer with
the  prefs  command

   Commands :  prefs
   Tutor chapters :  Getting Started
   Related terminology :
              commandline
```

1.41 Glossary : hold mode

```
                When PowerVisor is in hold mode, the screens and windows of  ←-
                   PowerVisor
are all closed. This is useful to preserve memory. You can reopen
PowerVisor with the hot key

   Commands :  hold
   Related terminology :
              hot key
```

1.42 Glossary : home position

```
                The home position for a logical window depends on whether the  ←-
                   logical
window is a top-visible window or a real-top window

   Tutor chapters :  Screens and windows
   Related terminology :
              top-visible windows

              real-top windows
```

1.43 Glossary : hot key

```
                The PowerVisor hot key is the key combination used to bring the
PowerVisor screen to the front. Normally the hot key is
<right-shift>+<right-alt>+'?' but you can redefine it to any other
key you want with the  prefs  command

   Commands :  prefs
   Tutor chapters :  Getting Started
   Related terminology :
              hold mode
```

```
                interrupt key

                pause key
```

1.44 Glossary : interrupt key

```
                  The interrupt key (<esc> by default) can be used to interrupt a
   PowerVisor command. You can use any key you want for the interrupt
   key with the  prefs  command

      Commands :  prefs
      Tutor chapters :  Installing PowerVisor
      Related terminology :
                 hot key

                 pause key
```

1.45 Glossary : IntuiTick

```
   An IntuiTick is one tenth of a second. It is used by the  refresh
   command to measure the refresh rate

      Commands :  refresh
```

1.46 Glossary : key attacements

```
                  See
                macro
```

1.47 Glossary : key code

```
                  A key code is some quantity used by the Amiga operating system ←
                     to
   distinguish between different keys on the keyboard. PowerVisor uses
   key codes in several cases (in conjunction with qualifiers) in order
   to install some commands on keys or other things

      Commands :  attach   prefs
      Tutor chapters :  Installing PowerVisor
      Related terminology :
                 code

                 qualifier
```

1.48 Glossary : led monitor

```
                This is a (older) name for the function monitor

   Related terminology :
              function monitor
```

1.49 Glossary : linenumber operator

```
The linenumber operator ('#') is useful when debugging. Directly after
the operator follows a linenumber in the currently loaded source. The
result is the address in memory for that linenumber

   Tutor chapters :  Expressions   Debugging
   Releated terminology : 'special operator'
```

1.50 Glossary : list

```
                A list contains some list elements (structures or nodes). You ↩
                    can look
at a list with the  list  command. You can ask more information
about a list element in a list with the  info  command.
For example the  task  list contains all processes and tasks. The list
is called 'task' and each list element in the list is either a process
or a task.

   Commands :  list   info
   Tutor chapters :  List Reference   Looking at things
   Related terminology :
              list element

              current list
```

1.51 Glossary : list element

```
                See
              list
```

1.52 Glossary : list operator

```
                The list operator (':') is a binary operator with both ↩
                    arguments for
the operator optional. You can use it to search some item in a list

   Tutor chapters :  Expressions   List Reference
   Related terminology :
              list
```

```
                list element

                address operator
```

1.53 Glossary : log file

```
                  A log file is a file used to store all output appearing in a  ←
                     logical
   window. You can only have one log file at a time and only for one
   logical window

      Commands :  log   to
      Tutor chapters :  Screens and windows
      Related terminology :  logical windows
                 output log
```

1.54 Glossary : logical window

```
                  A logical window is the primary output device used by  ←
                     PowerVisor. It
   corresponds with a visible rectangle on a physical window (or
   Intuition window). You can have more than one logical window on
   each physical window. A logical window has an internal size and
   a visible size. The internal size is the number of columns and
   rows that PowerVisor really remembers for that logical window.
   The visible size is the part of the logical window that you can see.
   Using commands or keys you can scroll the visible logical window
   region in the real logical window region.
   The most important logical window is the 'Main' logical window. This
   logical window is always open and receives all output (by default)
   for most commands. All logical windows can be found in the  lwin
   list

      Commands :  openlw   closelw   rwin   dwin   awin   owin   xwin
      Tutor chapters :  Screens and windows
      Related terminology :
                 physical window

                 LW

                 standard logical window
```

1.55 Glossary : LW

```
                  An abbreviation for
                 logical window
                      Related terminology :
                 logical window
```

1.56 Glossary : machinelanguage scripts

```
                 See
               ML-scripts
```

1.57 Glossary : macro

```
A macro is another (and better) name for a key attachement. I hope to
remove the term 'key attachement' from all documentation and use
'macro' instead. A macro is a command that is attached to a key.
When you press that key the command is executed.

   Commands :  attach   remattach
   Tutor chapters :  Installing PowerVisor
   Related terminology :  key attachement
```

1.58 Glossary : masterbox

```
                 The masterbox is the box that is always present in a physical ←-
                    window.
It is the root of the box tree. The masterbox is the only box that
can have no children while containing no logical window

   Tutor chapters :  Screens and windows
   Related terminology :
              box
               logical windows   physical windows
```

1.59 Glossary : ML-scripts

```
                 ML-scripts (or machinelanguage scripts) are scripts written in ←-
                    an
external language (like C or machinelanguage). They are useful
for more specialized tasks. Some examples can be found in the
's/pv' subdirectory

   Commands :  script   pvcall
   Tutor chapters :  Scripts   The wizard corner
   Related terminology :
              ARexx scripts

              PowerVisor scripts

              scripts
```

1.60 Glossary : MMU tree

```
The MMU tree (or Memory Management Unit tree) is used by the 68851,
68030 or 68040 for memory management. On the Amiga the MMU is only
marginally used. In future PowerVisor may be able to protect memory
using the MMU. At this moment you can list the MMU tables and the
MMU registers

   Commands :  mmutree   mmuregs   specregs
   Tutor chapters :  Looking at things
```

1.61 Glossary : monitor functions

```
Monitor functions are library functions you are monitoring with the
 addfunc  command. Using this command provides a powerful way to
debug some programs

   Commands :  addfunc
```

1.62 Glossary : MORE checking

```
                MORE checking is an optional setting for logical windows. When
MORE checking is on, PowerVisor will wait for a key press after
a full page of output has appeared (a full page is measured by
the real size of the logical window and not by the visible size).
MORE checking is on by default for the 'Main' logical window

   Commands :  mode
   Tutor chapters :  Screens and windows
   Related terminology :
              auto output snap
               logical windows

              home position
```

1.63 Glossary : names

```
                Names are actually strings but without the single quotes. There ←
                    is
no other significant difference. Note however, that names are NOT
always interpreted in the same way as a normal string (with single
quotes). For example, you cannot use single quotes when you want
the name to use as a variable

   Tutor chapters :  Expressions
   Related terminology :
              strings

              single quotes
```

1.64 Glossary : nofancy mode

```
                  See
                fancy mode
```

1.65 Glossary : normal breakpoints

```
                  See
                breakpoints
```

1.66 Glossary : output log

```
                  See
                log file
```

1.67 Glossary : pause key

```
                  The pause key ('<right-alt>+<help> by default) can be used to  ↩
                     pause
   the output of a command

      Commands :  prefs
      Tutor chapters :  Installing PowerVisor
      Related terminology :
                hot key

                interrupt key
```

1.68 Glossary : pen

```
   A pen is a graphical object. A pen in PowerVisor has a name and a
   value. For all graphic operations PowerVisor uses a pen. This means
   that you can customize all colors you see on the PowerVisor screen
   (and not only with RGB values)

      Commands :  prefs
      Tutor chapters :  Installing PowerVisor   Screens and windows
```

1.69 Glossary : physical window

```
                  A physical window directly corresponds with a normal Intuition
   window. A physical window can contain one or more logical windows.
   The visible size for logical windows is managed by the physical
   windows using the 'Box' concept. The most important physical
```

window is the 'Main' physical window. This physical window contains
the 'Main' logical window. All physical windows can be found in
the pwin list

Commands : openpw closepw
Tutor chapters : Screens and windows
Related terminology :
logical window

box

PW

1.70 Glossary : PortPrint

PowerVisor supports a PortPrint feature. This means that you can
print debug information (using the powervisor.library) on the
PowerVisor screen. This is useful for tasks for example because
they normally can't easily print output. The name PortPrint is
derived from the way this feature works. A message is send to
the PowerVisor message port containing the output string (or
some other data because PowerVisor supports more types of output).
Note that the output of the portprint commands appears on the
'PPrint' logical window if this window is open

Commands : owin

1.71 Glossary : PowerVisor device

See
PVDevice

1.72 Glossary : PowerVisor screen

This is the screen where all PowerVisor windows (physical ↩
windows)
live. It is possible that this screen does not exist. This happens
when you have opened PowerVisor on the WorkBench screen or on any
other screen in the system

Commands : screen
Tutor chapters : Screens and windows
Related terminology : physical windows
PowerVisor window

1.73 Glossary : PowerVisor scripts

```
                PowerVisor scripts are simple script containing only sequential
PowerVisor commands. They are useful for installing things. Some
examples scripts can be found in the 's/pv' subdirectory. The
PowerVisor startup script (s/PowerVisor-startup) is also a
PowerVisor script. For more complex scripts you should use
ARexx

   Commands :  script
   Tutor chapters :  Scripts
   Related terminology :
              ARexx scripts

              scripts

              ML-scripts
```

1.74 Glossary : PowerVisor startup file

```
                See
              startup file
```

1.75 Glossary : PowerVisor window

```
                The PowerVisor window is another name for the 'Main' physical ↩
                   window.
This physical window contains at least the 'Main' logical window for
standard output

   Tutor chapters :  Screens and windows
   Related terminology :
              PowerVisor screen
               physical windows
                          logical windows
```

1.76 Glossary : prefix operators

```
Prefix operators are operators you can put in front off the commandline
before you execute it (press enter). These prefix operators have
some effect on the output of the command or on other things

   Tutor chapters :  Screens and windows   Technical information
```

1.77 Glossary : private breakpoints

```
                Private breakpoints are used by PowerVisor to skip an  ←
                   instruction or
for other purposes

   Commands :  break   debug   trace
   Tutor chapters :  Debugging
   Related terminology :
              debug nodes

              breakpoints
```

1.78 Glossary : profiler breakpoint

```
                A profiler breakpoint never breaks, but only increments a  ←
                   counter
everytime the breakpoint is passed. This is useful to gather usage
statistics

   Commands :  break   trace   debug
   Tutor chapters :  Debugging
   Related terminology :
              breakpoints

              debug nodes
```

1.79 Glossary : prompt

```
                The prompt is the '>' symbol right from the current list  ←
                   indicator.
It indicates the stringgadget or commandline where you can type
PowerVisor commands

   Related terminology :
              current list indicator
```

1.80 Glossary : PVDevice

```
(or PowerVisor device) A PVDevice is a data structure used by some
commands. With a PVDevice you can open any device in the system and
send commands to it. This is useful to test selfmade devices or
to learn about other devices

   Commands :  opendev   devcmd
```

1.81 Glossary : PVSD file

```
                A pvsd file (PowerVisor Structure Definition file) contains  ↩
                   some
structures. 'pvsd' files are made by the 'MStruct' utility

   Commands :  addstruct   interprete
   Tutor chapters :  Looking at things
   Related terminology :
              structure definition
```

1.82 Glossary : PW

```
                An abbreviation for
              physical window
                   Related terminology :
              physical window
```

1.83 Glossary : qualifier

```
                A qualifier is used together with a key code to distinguish  ↩
                   between
different key presses. A qualifier says something about some special
keys pressed at the same time with the key (shift, alt, ...)

   Commands :  attach   prefs
   Tutor chapters :  Installing PowerVisor
   Related terminology :
              key code
```

1.84 Glossary : quote operator

```
                The quote operator (or backslash '\') can be used to put  ↩
                   integers,
characters or other strings in one way or another in a string or
string pointer. The quote operator is actually quiet powerful

   Tutor chapters :  Expressions
   Related terminology :
              strings

              string pointers

              quotes
```

1.85 Glossary : quotes

```
                Quotes are used to define a string or string pointer. A single ↩
                   quote
is used for real strings and a double quote is used for string pointers.
For commands expecting a string there is no difference between using
the single quote or the double quote. But if a command expects an
integer as an argument there is a difference. A double quoted string (or
string pointer) is in fact a pointer to that string while a single
quoted string will be parsed according to several steps (variable, list
element, symbol, function, ...)

   Tutor chapters :  Expressions
   Related terminology :
              strong quote

              strings

              string pointers
```

1.86 Glossary : real-top windows

```
                A real-top window is a logical window with the home position ↩
                   set
to location (0,0). This means that when the logical window is cleared
the current cursor position is automatically set to that position and
the logical window is scrolled to the top-left visible corner in
the real region of the logical window. The 'Refresh' and 'Debug'
logical windows are real-top windows by default. See
'top-visible windows' for the other way to set the home position

   Tutor chapters :  Screens and windows
   Related terminology :
              top-visible windows

              logical window

              home position
```

1.87 Glossary : recursive aliases

```
                Using the group operator you can make recursive aliases. This ↩
                   is
because alias expansion is done again in a new group

   Tutor chapters :  Installing PowerVisor
   Related terminology :
              aliases

              group operator
```

1.88 Glossary : resident breakpoints

```
                A resident breakpoint is a breakpoint that you can put in your ↩
                   program
even before PowerVisor is running. You put it in your program before
compiling or assembling it. Resident breakpoints (like all breakpoints)
are simply 'ILLEGAL' instructions

   Commands :  debug
   Tutor chapters :  Debugging
   Related terminology :
              debug nodes

              breakpoints
```

1.89 Glossary : resident commands

```
                See
              resident ML-scripts
```

1.90 Glossary : resident ML-scripts

```
                (or resident commands) For faster execution you can make ML- ↩
                   scripts
resident. Note that they must be reentrant

   Commands :  resident   unresident
   Tutor chapters :  Scripts
   Related terminology :
              resident commands

              ML-scripts
```

1.91 Glossary : scripts

```
                See
              ARexx scripts
             ,
              PowerVisor scripts
              or
              ML-scripts
```

1.92 Glossary : single quotes

```
                See
              quotes
```

1.93 Glossary : singlestep mode

```
                When you are tracing a program (a debug node), PowerVisor can ↩
                    use
two modes : 'singlestep mode' or 'execute mode'. In singlestep mode
each instruction is executed step by step. After each instruction
an exception handler is called and some action is taken (you can
control this action with the  trace  command). In execute mode
the program is running at full speed. The program only stops when
a breakpoint or other exception occurs

   Commands :  trace   break   debug
   Tutor chapters :  Debugging
   Related terminology :
              debug nodes
```

1.94 Glossary : size bar

```
The size bar is the (mostly horizontal) bar between two logical windows.
You can use this bar to resize the logical windows

   Tutor chapters :  Screens and windows
   Related terminology :  logical windows
```

1.95 Glossary : snapping

```
Snapping is the process of moving the mouse to a position in a logical
window and clicking on the word under the mouse pointer. The word
will be copied to the stringgadget

   Tutor chapters :  Getting Started
```

1.96 Glossary : special operator

```
The special operator ('@') is useful when debugging. It returns the
value of the registername directly after the operator character

   Tutor chapters :  Expressions   Debugging
   Releated terminology : 'linenumber operator'
```

1.97 Glossary : special variables

```
                Special variables are a bit special :-) Special variables ↩
                    behave
like normal variables in that you can assign values to them. But
when you assign something to a special variable, a certain routine
is called. The 'mode' variable is an example of a special variable.
```

When you change something in the mode variable PowerVisor will
automatically adapt all internal settings to the new settings provided
in the assignment

Commands : vars mode
Tutor chapters : Looking at things
Related terminology :
variables

constants

functions

1.98 Glossary : stack checking

PowerVisor has two stack checkers (not counting the internal ↩
stack
checker for PowerVisor). These stack checkers check if a certain
task (with the stack command) or all tasks (with the account
command) have enough room left on the stack. The minimum amount of room
allowed on the stack is called the ’stack fail level’

Commands : stack account
Related terminology :
stack fail level

1.99 Glossary : stack fail level

The stack fail level is the minimum size of the stack that ↩
PowerVisor
allows before it will halt a task. It is used both by the account
and the stack commands

Commands : stack account prefs
Related terminology :
stack checking

1.100 Glossary : standard logical window

A standard logical window is a logical window with a predefined
meaning for PowerVisor. In the current version there are seven
standard logical windows : Main, Extra, Debug, Refresh, Rexx, PPrint
and Source

Commands : rwin awin dwin xwin owin swin
Tutor chapters : Screens and windows
Related terminology :
logical window

1.101 Glossary : startup file

```
                (or 'PowerVisor startup file') The startup file or s/PowerVisor ↩
                   -startup
   file is equivalent to the startup-sequence file. It is a PowerVisor
   script containing initialization commands. It is executed when
   PowerVisor starts

      Related terminology :
                 scripts

                 PowerVisor scripts
```

1.102 Glossary : string expansion

```
                String expansion is sometimes used to refer to the process of  ↩
                   parsing
   a string (a sequence of characters) while assigning special meanings
   to some characters (like the quote operator and strong quote operator)

      Tutor chapters :  Expressions
      Related terminology :
                 strings

                 quote operator

                 strong quote
```

1.103 Glossary : string pointers

```
                A string pointer (defined with double quotes) is a pointer to
   a sequence of characters. It is actually an integer is and is
   used as such by all commands expecting integers as an argument.
   This means that arithmetic on string pointers is perfectly valid
   and is equivalent to C pointer arithmetic

      Tutor chapters :  Expressions
      Related terminology :
                 strings

                 quotes
```

1.104 Glossary : strings

```
                A string (defined with or without single quotes) is a sequence  ↩
                   of
   characters. Normally strings are surrounded by single quotes (or
   without quotes) but if a command expects a string as an argument
   double quotes will do as well. Note that this is NOT the case for
```

```
a command expecting an integer as an argument. Strings (with
single quotes) will be parsed according to some steps (variable,
function, symbol, list element, ...) while a string pointer
(with double quotes) simply corresponds to the pointer to the
string

   Tutor chapters :  Expressions
   Related terminology :
              string pointers

              quotes
```

1.105 Glossary : structure definition

```
                A structure definition corresponds with a structure (like in C  ↩
                   or
assembler) or a record (like in Pascal). With the external utility
'MStruct' you can make structure definitions to be used by
PowerVisor. A structure definition contains a list of names (for
the structure fields) and their corresponding types (APTR, BPTR,
BSTR, CSTR, BYTE, WORD, LONG, ...). You can interprete a range
of memory as a structure or you can use tags to permanently define
a region of memory as a structure

   Commands :  addstruct   interprete
   Tutor chapters :  Looking at things
   Related terminology :
              pvsd file

              tag
```

1.106 Glossary : strong quote

```
                The strong quote '·' (or <alt>+8 on the keyboard) is normally  ↩
                   not
used very often. Using the strong quotes you can easily put all
characters in a string expect one. This is the character directly
after the string quote. This character is used to end the
strong quote region

   Tutor chapters :  Expressions
   Related terminology :
              strings

              string pointers

              quotes
```

1.107 Glossary : symbols

Symbols are names for labels and addresses used in programs. Most assemblers and compilers can output symbols in the program hunks. PowerVisor supports these symbols when you are debugging programs

```
   Commands :  symbol   debug
   Tutor chapters :  Debugging
```

1.108 Glossary : tag

A tag is a definition for a region of memory. There are 16 tag ↩ lists.
Each tag list can contain an arbitrary number of tags. One tag contains a pointer to the start of a memory block, a size in bytes and a type (Byte/Ascii, Code, Structure, ...)

```
   Commands :  addtag   remtag   view
   Tutor chapters :  Looking at things
   Related terminology :
              tag list

              tag file

              current tag list
```

1.109 Glossary : tag file

A tag file contains some tags saved with the savetags command ↩ .

```
   Commands :  savetags   loadtags
   Tutor chapters :  Looking at things
   Related terminology :
              tag

              tag list
```

1.110 Glossary : tag list

See
tag

1.111 Glossary : task accounting

When you enable task accounting (with the account command) PowerVisor counts the number of task switches for each task. This gives a rough indication of the cpu time a task uses. You can see this accounting information in the task list

```
      Commands :  account   list
```

1.112 Glossary : Task Control Block (or TCB)

```
                   The Task Control Block is another name for the task structure.

      Related terminolgy :
                 TCB

                 task list
```

1.113 Glossary : task list

```
                   The  task  list contains all processes and tasks currently in  ←
                      the
   system.

      Commands :  task   list
      Tutor chapters :  List Reference
      Related terminology :
                 list
```

1.114 Glossary : TCB

```
                   See
                 Task Control Block (or TCB)
```

1.115 Glossary : templates

```
   A template is a syntaxical description of a command. If you have the
   online help files installed (PowerVisor-help and PowerVisor-ctrl)
   you can get command templates by using '?' as the first argument (just
   like CLI commands)
```

1.116 Glossary : temporary breakpoint

```
                   A temporary breakpoint only breaks once. After the breakpoint  ←
                      has done
   its work it will automatically disappear

      Commands :  break   trace   debug
      Tutor chapters :  Debugging
      Related terminology :
                 breakpoints
```

```
              debug nodes
```

1.117 Glossary : timeout breakpoints

```
                A timeout breakpoint only breaks after a specified number of ↩
                   times

     Commands :  break   trace   debug
     Tutor chapters :  Debugging
     Related terminology :
              breakpoints

              debug nodes
```

1.118 Glossary : top-visible windows

```
                A top-visible logical window is a logical window with the home ↩
                   position
   set to the top-left position of the bottom-left visible region of the
   real region of the logical window. This means that when such a window
   is cleared, the current cursor position is set to that position and
   the logical window is scrolled to the bottom visible region.
   The 'Main' logical window is top-visible by default. See
   'real-top windows' for the other way to set the home position

     Tutor chapters :  Screens and windows
     Related terminology :
              real-top windows
               logical windows

              home position
```

1.119 Glossary : variables

```
                A variable can be used to remember some value. PowerVisor only ↩
                   has
   integer type variables (although a variable may point to a string,
   this is in fact a C string). There is no limitation (except memory)
   on the length of the variable name. A variable name must start with
   a letter or an underscore but may contain digits in the rest of the
   name.
   Note that variables, constants, special variables and functions all
   live in the same internal list

     Commands :  vars   remvar   assign
     Tutor chapters :  Expressions
     Related terminology :
              constants
```

```
functions

special variables
```