

GettingStarted.hyper

COLLABORATORS

	<i>TITLE :</i> GettingStarted.hyper		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		December 6, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	GettingStarted.hyper	1
1.1	Getting Started (Mon Nov 2 20:06:37 1992)	1
1.2	Getting Started : Commands used in this tutorial	1
1.3	Getting Started : Functions used in this tutorial	2
1.4	Getting Started : How to start PowerVisor	2
1.5	Getting Started : Starting PowerVisor on faulty 68020 boards	3
1.6	Getting Started : Commandline options	3
1.7	Getting Started : Current list and some basic commands	3
1.8	Getting Started : Snapping away	7
1.9	Getting Started : The PowerVisor hot key	8
1.10	Getting Started : Errors	8
1.11	Getting Started : Templates	9
1.12	Getting Started : Interrupting PowerVisor	10
1.13	Getting Started : The history buffer	12
1.14	Getting Started : Making a Config file for PowerVisor	13
1.15	Getting Started : Standard PowerVisor keys	14
1.16	Getting Started : Special files	14
1.17	Getting Started : A summary	15

Chapter 1

GettingStarted.hyper

1.1 Getting Started (Mon Nov 2 20:06:37 1992)

Contents:

- How to start PowerVisor
- Starting PV on faulty 68020 boards
- Commandline options
- Current list and some basic commands
- Snapping away
- The PowerVisor hot key
- Errors
- Templates
- Interrupting PowerVisor
- The history buffer
- Making a Config file for PowerVisor
- Standard PowerVisor keys
- Special files
- A summary

Various:

- Commands used in this tutorial
- Functions used in this tutorial
- Back to main contents

1.2 Getting Started : Commands used in this tutorial

disp	Display integer
help	Ask for help
hold	Go to 'hold' mode (PowerVisor closes screen and window)
info	Give information on element in list
kill	Kill a task or process
list	Show a list (tasks, libraries, message ports, ...)
memory	List memory
mode	Set PowerVisor preferences
port	Go to the message port list
saveconfig	Save config file
task	Go to the task list
why	Ask more information about an error

prefs Set/Get preferences for PowerVisor

1.3 Getting Started : Functions used in this tutorial

None

1.4 Getting Started : How to start PowerVisor

To start PowerVisor, you can simply type (Before you continue, note that I assume that the s:PowerVisor-config file does not exist and that the s:PowerVisor-startup file is minimal (like on the original disk). If this is not the case some output may not be the same as in this tutorial. If you want to be absolutely sure, delete s:PowerVisor-config and s:PowerVisor-startup (or rename them)) :

```
< pv <enter>
```

or

```
< run pv <enter>
```

Normally PowerVisor will now open a screen. If this does not work it is possible that you do not have enough memory. Quit some programs and try again. PowerVisor is not very memory consuming. Also note that you need the powervisior.library installed in your libs: directory.

If everything is fine you will see the PowerVisor screen. This screen is very sober. The bottom line of the screen is the stringgadget where you must type all PowerVisor commands. This stringgadget is automatically activated whenever the PowerVisor window becomes active.

(1.3 : due to a bug in AmigaDOS 1.2/1.3 you can get problems when you use a sunmouse like program. If the PowerVisor stringgadget is active, no other window can be activated. If this happens you must activate the window by clicking on it.)

The stringgadget buffer is 400 bytes long by default. This is also the largest command that you can execute (you can change this value with the prefs command).

The rest of the screen is dedicated for PowerVisor output. After startup this screen is normally empty (except for the copyright message).

Because the PowerVisor commandline is a stringgadget you can use all editing facilities provided for stringgadgets.

You will also see a blue bar (this color may be different if you use other color preferences or if you use AmigaDOS 1.3) just below the screen bar containing the word 'Main' at the left and a strange box at the right. These things will be explained in the Screens and Windows tutorial chapter.

If you start more than one PowerVisor at the same time, all later instances of PowerVisor are called slaves. The first instance is the master. Slave instances can't debug and only the master instance traps

crashes. You can use the 'pv' constant to see which instance you are (0 is the master, 1 is the first slave, ...) or you can use the `pubscreen()` and `arexport()` functions to get the name of the public screen and the name of the ARexx port respectively for this instance of PowerVisor. This is useful if you are writing ARexx scripts for PowerVisor. Normally you need not concern yourselves with the difference between instances (except for the debugging of course). You can see in which instance of PowerVisor you are running by looking at the titlebar of the screen. All slave instances have '[<num>]' in the titlebar.

1.5 Getting Started : Starting PowerVisor on faulty 68020 boards

There are some 68020/30 boards that give incorrect information when asked if a MMU is present. Therefore PowerVisor is not able to check if the MMU is present and will even hang on those systems. To solve this problem I have added a commandline option '-m'. With this commandline option you can force a certain MMU or no MMU. When you use this option PowerVisor will not use the internal MMU test routine.

For example:

- use -m0 if you have a 68020 but NO MMU
- use -m68851 if you have a 68020 WITH a MMU (68851)
- use -m68030 or -m68040 if you have these processors

You should not use this commandline option unless you have a faulty board (like some CSA 68020 boards and maybe others)

1.6 Getting Started : Commandline options

PowerVisor has the following commandline options :

- m<processor> To specify the processor or MMU you have. Few people will need this option.
See the Starting PowerVisor on faulty 68020 boards section for more information.
- c Don't use the PowerVisor-config file. All default values are used instead.
- s Don't use the PowerVisor-startup file.
- d Do not detach from the CLI. Note that PowerVisor only detaches if you have AmigaDOS 2.0

1.7 Getting Started : Current list and some basic commands

Left from the stringgadget is the current list indicator. Default is 'task'. The current list is the list that is used for string parsing (see later) and the `list` command. Try this :

```
< list <enter>
```

And you get something like this :

```
> Task node name      : Node      Pri   StackU   StackS Stat Command      Acc
```

```

> -----
> ConClip Process      : 07E60410 00      242      4000 Wait sys:c/ConCl(02) -
> REXXMaster          : 07E6AA48 04      162      2048 Wait                (00) -
> « IPrefs »         : 07E59568 00      862      3500 Wait                PROC -
> ClickToFront       : 07E75210 15      398      4096 Wait                PROC -
> CpuBlit            : 07E7BA18 00      266      2048 Wait                PROC -
> DefaultIcon        : 07EA0FE8 00      118      4000 Wait                PROC -
> Stacker            : 07E80DE0 00      102      8192 Wait                PROC -
> LacePointer.Deamon : 07E72470 05        82      1024 Wait                TASK -
> GMC-Thread         : 07E7DAD0 05      122      2000 Wait                PROC -
> ToolsDaemon        : 07E9BF08 02      126      4000 Wait ToolsDaemon(05) -
> Star_Blanker       : 07E7F848 00      310      4096 Wait                PROC -
> ramlib             : 07E29048 00      230      2048 Wait                PROC -
> Shell Process      : 07E9ED00 00      826      3200 Wait                (04) -
> GMC                : 07E7C308 05      170      2500 Wait                PROC -
> console.device     : 07E0DEE8 05        90      4096 Wait                TASK -
> PowerSnap 1.1 by Nic: 07E9B630 05      102      2000 Wait                PROC -
> AltMenuCom         : 07E8FDC0 00      198      3000 Wait                PROC -
> SCSI handler       : 07E0A318 0C        98      1000 Wait                TASK -
> scsi.device        : 07E096E0 0B        94      1000 Wait                TASK -
> WB_2.x             : 07E0FF38 0A      130      2400 Wait                PROC -
> DF0                : 07E15910 0A      130      2400 Wait                PROC -
> Backup            : 07E17DD0 0A      130      2400 Wait                PROC -
> DH1                : 07E1DB88 0A      130      2400 Wait                PROC -
> DH2                : 07E23770 0A      130      2400 Wait                PROC -
> Workbench          : 07E6C340 01      166      6000 Wait Workbench (03) -
> PowerVisor.task    : 07E7B4F0 00        82      4096 Wait                TASK -
> trackdisk.device   : 07E0E714 05        98        512 Wait                TASK -
> RAM                : 07E31220 0A      678      1200 Wait                PROC -
> input.device       : 07E07F12 14        86      4096 Wait                TASK -
> pv                 : 07F62FC0 04      808     16000 Run pv          (01) -

```

You can also get this list by typing :

```
< list task <enter>
```

But since 'task' is the current list (at this moment) this is not necessary.

In this list you see all the tasks currently in the system. 'Node' is the address in memory, 'Pri' is the task priority (in hexadecimal), 'StackU' is the current stack usage, 'StackS' is the size of the stack, 'Stat' is the state of the task ('Rdy' for ready, 'Wait' for waiting and 'Run' for running) and 'Command' is the executing command (this is only for cli processes). After the command name you can see 'TASK' for tasks, 'PROC' for a process and '(xx)' for a cli. 'Acc' is for accounting information, it is disabled at this moment.

All hexadecimal numbers are padded with zeroes.

You can also go to another current list by typing the list name :

```
< p <enter>
```

(Note how the current list indicator has changed)

```
< l <enter>
```

Note that we used abbreviations for the commands. 'l' is the same as 'list' and 'p' is equivalent to 'port'. Most commands can be abbreviated. You can type 'help commands' to see all commands with their shortcuts (the capital letters represent the required bit of the command, all other characters are optional, this convention is used everywhere in this documentation where appropriate).

You get something like this :

```
> MsgPort node name      : Node      Pri SigBit SigTask
> -----
> REXX                   : 07E2B13C 00          31 07E39BA8
> AREXX                  : 07E3A438 00          30 07E39BA8
> PowerVisor1.0.port     : 07E81996 00           1 00000000
> IPrefs.rendezvous      : 07E23800 E2          31 07E26BA8
> SetPatch Port          : 07E23060 9C           0 00000000
```

Now we go back to the task list :

< t <enter>

or

< task <enter>

and ask a list :

< list <enter>

```
> Task node name        : Node      Pri   StackU   StacksS Stat Command           Acc
> -----
> ConClip Process      : 07E60410 00       242     4000 Wait sys:c/ConCl(02) -
> REXXMaster           : 07E6AA48 04       162     2048 Wait                (00) -
> « IPrefs »           : 07E59568 00       862     3500 Wait                PROC -
> ClickToFront         : 07E75210 15       398     4096 Wait                PROC -
> CpuBlit              : 07E7BA18 00       266     2048 Wait                PROC -
> DefaultIcon          : 07EA0FE8 00       118     4000 Wait                PROC -
> Stacker              : 07E80DE0 00       102     8192 Wait                PROC -
> LacePointer.Daemon   : 07E72470 05        82     1024 Wait                TASK -
> GMC-Thread           : 07E7DAD0 05       122     2000 Wait                PROC -
> ToolsDaemon          : 07E9BF08 02       126     4000 Wait ToolsDaemon(05) -
> Star_Blanker         : 07E7F848 00       310     4096 Wait                PROC -
> ramlib               : 07E29048 00       230     2048 Wait                PROC -
> Shell Process        : 07E9ED00 00       826     3200 Wait                (04) -
> GMC                  : 07E7C308 05       170     2500 Wait                PROC -
> console.device       : 07E0DEE8 05        90     4096 Wait                TASK -
> PowerSnap 1.1 by Nic: 07E9B630 05       102     2000 Wait                PROC -
> AltMenuCom           : 07E8FDC0 00       198     3000 Wait                PROC -
> SCSI handler         : 07E0A318 0C        98     1000 Wait                TASK -
> scsi.device          : 07E096E0 0B        94     1000 Wait                TASK -
> WB_2.x               : 07E0FF38 0A       130     2400 Wait                PROC -
> DF0                  : 07E15910 0A       130     2400 Wait                PROC -
> Backup               : 07E17DD0 0A       130     2400 Wait                PROC -
> DH1                  : 07E1DB88 0A       130     2400 Wait                PROC -
> DH2                  : 07E23770 0A       130     2400 Wait                PROC -
> Workbench            : 07E6C340 01       166     6000 Wait Workbench (03) -
```

```

> PowerVisor.task      : 07E7B4F0 00          82      4096 Wait          TASK -
> trackdisk.device    : 07E0E714 05          98       512 Wait          TASK -
> RAM                  : 07E31220 0A         678     1200 Wait          PROC -
> input.device        : 07E07F12 14          86     4096 Wait          TASK -
> pv                   : 07F62FC0 04         808    16000 Run   pv      (01) -

```

Now we are going to ask some information about a task with the `info` command :

```
< info powervisor <enter>
```

and we get something like this :

```

> Task node name      : Node      Pri   StackU   StackS Stat Command          Acc
> -----
> PowerVisor.task    : 07E7B4F0 00          82      4096 Wait          TASK -
>
> IDNestCnt          : FF          | TDNestCnt      : 00          | SigAlloc       : C000FFFF |
> SigWait            : C0000000    | SigRecvd       : 00000004    | SigExcept      : 00000000 |
> TrapAlloc          : 8000        | TrapAble       : 0000         | ExceptData     : 00000000 |
> ExceptCode         : 00F83068    | TrapData       : 00000000    | TrapCode       : 07F4F3B6 |
> SpLower            : 07E9FFE8    | SpUpper        : 07EA0FE8    | SpReg          : 07EA0F96 |
> MemEntry           : 07E7B53A    | Switch         : 00000000    | Launch        : 00000000 |
> Flags              : 00          | UserData       : 00000000    |

```

This is the listing of the task structure. If you ask `info` about a process you get more information. If you ask `info` about a cli process you get even more information.

(1.3 : The amount of information is a bit less in 1.3 because there are some new fields in AmigaDOS 2.0).

Because the task list was the current list we can use the names present in this list instead of the pointer to the task. This name is not case sensitive and need not be the complete name. All the following commands would be equivalent (see the `Expressions` chapter for more info) :

```

< info powerv <enter>

< info 07E8CE60 <enter>

< info 07e8ce60 <enter>

< info 'PoWerVisor.tASK' <enter>

< info task:powerv task <enter>

```

Look at the last command. We would have needed this notation if our current list wasn't the task list. The '`<list>:`' notation can be used everywhere. (Do not forget the extra '`task`' argument, otherwise PowerVisor can crash)

You see that it can be handy to set the current list right.

Some commands (like `kill` , `freeze` , ...) do not need the current list. They automatically assume the list which is right in most cases for that particular command (this feature is called '`autodefault`'). For example (Don't type this, because it can crash your Amiga !) :

Go to another list :

```
< port <enter>
```

Try to kill the PowerVisor task :

```
< kill powervisor <enter>
```

Although the current list is 'port', you need not precede 'powervisor' with 'task:'. The 'kill' command automatically assumes the 'task' list. If you still want to 'kill' a port you can always type :

```
< kill port:someport <enter>
```

The `info` command (see above) does not set it's own current list because it can be used on all lists.

1.8 Getting Started : Snapping away

It is not always possible to use names. If you have two tasks with the same name it would be ambiguous. In that case you must use the address of the task. Try the following :

List the screens with `list` :

```
< list scrs <enter>
```

and you will get something like this :

```
> Screen name      : Address  Left  Top Width Height FirstWindow
> -----
> PowerVisor      (V1.00 : 07E8EB20   0   0   692   452 07E8F818
> Other screen    : 07E748E0   0   0   704   456 07E753F8
> Workbench Screen : 07E3AF88   0 -572  692  1024 07E507C8
```

Type (Don't press enter)

```
< info <space>
```

and position the mouse on the address 07E8EB20 :

```
< <click left mouse button> scrs
```

On the commandline there should now be something like 'info 07E8EB20 scrs'.

```
< <enter>
```

and we get :

```
> Screen name      : Address  Left  Top Width Height FirstWindow
> -----
> PowerVisor      (V1.00 : 07E8EB20   0   0   692   452 07E8F818
>
> Flags           : 021F      | Font           : 07E8ECA6 | ViewPort      : 07E8EB4C
> RastPort        : 07E8EB74 | BitMap         : 07E8EBD8 | FirstGadget   : 07E7D17C
```

```

> DefaultTitle : PowerVisor (V1.00 beta)
> DetailPen    : 00          | BlockPen     : 01          | ExtData      : 00000000
> UserData     : 00000000 | BarHeight    : 0A          | BarVBorder   : 01
> BarHBorder   : 05          | MenuVBorder  : 02          | MenuHBorder  : 04
> WBorTop      : 02          | WBorLeft     : 04          | WBorRight    : 04
> WBorBottom  : 02          | LayerInfo    : 07E8EC00 | BarLayer     : 07E8F650
>
> Flags: CUSTOMSCREEN SHOWTITLE SCREENHIRES

```

This feature of PowerVisor is called 'snapping'. The left mouse button copies the word under the mousepointer to the stringgadget. If you snap a word PowerVisor will automatically add a space to the commandline below. If you do not like this you can disable this feature with the mode command.

Also note the special form of the info command. The second argument to 'info' is optional and is a name of a list. You MUST use the second argument if you want info about something that is not in the current list. Otherwise 'info' will try to interpret a screen as a task or something else. This could crash the Amiga !

1.9 Getting Started : The PowerVisor hot key

Go to the Workbench screen :

```
< <left amiga> n
```

If you want PowerVisor back you can use the hotkey (if you have QWERTY):

```
< <right alt>+<right shift>+ /
```

For any other keyboard you must press the key left from the <right shift> key.

The PowerVisor screen snaps back to life.

You can also use this key combination if PowerVisor is in hold-mode.

```
< hold <enter>
```

The PowerVisor screen disappears.

```
< <right alt>+<right shift>+ /
```

The PowerVisor screen is opened again.

This keycombination can be redefined with the prefs command. Also see the Standard PowerVisor keys section.

1.10 Getting Started : Errors

Everybody makes mistakes. Therefore PowerVisor needs some sort of error handling. Make sure that you have the following files installed in the S: directory or if you have AmigaDOS 2.0 you can put the files in the directory where the PowerVisor executable can be found :

PowerVisor-errors, PowerVisor-help and PowerVisor-ctrl.

Try this :

```
< task <enter>
< list <enter>
```

```
> Task node name      : Node      Pri   StackU   StackS Stat Command      Acc
> -----
> ConClip Process    : 07E60410 00      242     4000 Wait sys:c/ConCl(02) -
> REXXMaster         : 07E6AA48 04      162     2048 Wait                (00) -
> « IPrefs »        : 07E59568 00      862     3500 Wait                PROC -
> ...
> trackdisk.device  : 07E0E714 05        98       512 Wait                TASK -
> PowerVisor.task   : 07E7B4F0 00        82     4096 Wait                TASK -
> input.device      : 07E07F12 14        86     4096 Wait                TASK -
> RAM               : 07E31220 0A       678     1200 Wait                PROC -
> pv                : 07F62FC0 04       438    16000 Run   pv                (01) -
```

```
< info07F62FC0 <enter>
```

```
> Syntax Error !
```

Now you have done something wrong ! But what ? Well we should ask PowerVisor :

```
< why <enter>
```

```
> Syntax Error !
```

```
> You typed a command that powervisor did not understand, or some
> of your arguments are badly formed.
```

The `why` command (in fact it is an alias defined in `s:PowerVisor-startup`) can be very handy sometimes. You now notice that you forgot a `<space>` between `'info'` and `'07F62FC0'`.

1.11 Getting Started : Templates

If you forgot the syntax for some command you can ask the command template.

```
< memory? <enter>
or
< memory ? <enter>
```

```
> Memory [<start> [<bytes>]]
```

You can read the following information from this output :

- You can use `'m'` as an abbreviation for `'memory'` (the uppercase part of the command `'Memory'`)
- `<start>` is an optional argument

- <bytes> is also optional but you must supply a <start> value if you want to supply a <bytes> value.

The templates for these commands are located in the 'PowerVisor-help' file. You can also show the template using the help command :

```
< help memory_tmp <enter>
> Memory [<start> [<bytes>]]
```

This is the only way to ask a template for an ARexx command that has no equivalent on the PowerVisor commandline :

```
< help assign_tmp <enter>
> ASSIGN <assignment string>
```

(Note that 'ASSIGN' is completely in uppercase. This is normal because in ARexx you can't use abbreviations for commands)

1.12 Getting Started : Interrupting PowerVisor

Try the following :

```
< memory <enter>
```

or

```
< m <enter>
```

and you get something like :

```
> 00000000: 00000000 07E007E4 00F807FA 00F80ADE .....
> 00000010: 00F80AA2 00F80AA4 00F80AA6 00F80AA8 .....
> 00000020: 00F80BC8 00F80AAC 00F80AAF 00F80AB0 .....
> 00000030: 00F80AB2 00F80AB4 00F80AB6 00F80AB8 .....
> 00000040: 00F80ABA 00F80ABC 00F80ABE 00F80AC0 .....
> 00000050: 00F80AC2 00F80AC4 00F80AC6 00F80AC8 .....
> 00000060: 00F80ACA 00F810BC 00F8111A 00F81150 .....P
> 00000070: 00F811AE 00F81244 00F8128E 00F812D8 .....D.....
> 00000080: 00F80B38 00F80B3A 00F80B3C 00F80B3E ...8...:<...>
> 00000090: 00F80B40 00F80B42 00F80B44 00F80B46 ...@...B...D...F
> 000000A0: 00F80B48 00F80B4A 00F80B4C 00F80B4E ...H...J...L...N
> 000000B0: 00F80B50 00F80B52 FFFFFFFF 00F80B56 ...P...R.....V
> 000000C0: 00F80B58 00F80B5A 00F80B5C 00F80B5E ...X...Z...\...^
> 000000D0: 00F80B60 00F80B62 00F80B64 00F80B66 ...`...b...d...f
> 000000E0: 00F80B68 00F80B6A 00F80B6C 00F80B6E ...h...j...l...n
> 000000F0: 00F80B70 00F80B72 00F80B74 00F80B76 ...p...r...t...v
> 00000100: 66FFE6FC 66D7FE08 66FFAEF7 00000000 f...f...f.....
> 00000110: 66FF6EFA 66FBE67F 66FFAE7F 66FF66F1 f.n.f..f.f.f.
> 00000120: 66FFF6FB 66F7E6DF 66FFE67F 66FF6EF7 f...f...f..f.n.
> 00000130: 66FF66F9 66FF67FF 66FF6EFF 66DFB6E7 f.f.f.g.f.n.f...
```

This is a memory listing.

Now try this :

```
< memory 0 100000 <enter>
```

PowerVisor will now list 100000 bytes beginning at 0 :

```
> 00000000: 00000000 07E007E4 00F807FA 00F80ADE .....
> 00000010: 00F80AA2 00F80AA4 00F80AA6 00F80AA8 .....
> 00000020: 00F80BC8 00F80AAC 00F80AAF 00F80AB0 .....
> 00000030: 00F80AB2 00F80AB4 00F80AB6 00F80AB8 .....
> 00000040: 00F80ABA 00F80ABC 00F80ABE 00F80AC0 .....
> 00000050: 00F80AC2 00F80AC4 00F80AC6 00F80AC8 .....
> 00000060: 00F80ACA 00F810BC 00F8111A 00F81150 .....P
> 00000070: 00F811AE 00F81244 00F8128E 00F812D8 .....D.....
> 00000080: 00F80B38 00F80B3A 00F80B3C 00F80B3E ...8...:<...>
> 00000090: 00F80B40 00F80B42 00F80B44 00F80B46 ...@...B...D...F
> 000000A0: 00F80B48 00F80B4A 00F80B4C 00F80B4E ...H...J...L...N
> 000000B0: 00F80B50 00F80B52 FFFFFFFF 00F80B56 ...P...R.....V
> ...
```

If the page is full PowerVisor will wait for you. The current list indicator changes to '-MORE-'. Press <space> everytime you want to proceed to the next page. If you want to stop this listing you can press <esc>.

You can use the <esc> key to interrupt PowerVisor whenever you want. Or you can use the <right-alt>+<help> key to pause the PowerVisor output (the current list indicator will change to '-HALT-')

Note that if you do not want PowerVisor to stop at each page you can use the mode command to disable this feature :

```
< mode no more <enter>
```

```
< memory 0 100000 <enter>
```

```
> 00000000: 00000000 07E007E4 00F807FA 00F80ADE .....
> 00000010: 00F80AA2 00F80AA4 00F80AA6 00F80AA8 .....
> 00000020: 00F80BC8 00F80AAC 00F80AAF 00F80AB0 .....
> 00000030: 00F80AB2 00F80AB4 00F80AB6 00F80AB8 .....
> 00000040: 00F80ABA 00F80ABC 00F80ABE 00F80AC0 .....
> 00000050: 00F80AC2 00F80AC4 00F80AC6 00F80AC8 .....
> 00000060: 00F80ACA 00F810BC 00F8111A 00F81150 .....P
> 00000070: 00F811AE 00F81244 00F8128E 00F812D8 .....D.....
> 00000080: 00F80B38 00F80B3A 00F80B3C 00F80B3E ...8...:<...>
> 00000090: 00F80B40 00F80B42 00F80B44 00F80B46 ...@...B...D...F
> 000000A0: 00F80B48 00F80B4A 00F80B4C 00F80B4E ...H...J...L...N
> 000000B0: 00F80B50 00F80B52 FFFFFFFF 00F80B56 ...P...R.....V
> ...
```

```
< <esc>
```

```
< mode more <enter>
```

The interrupt and pause keys can be redefined with the prefs command. (See Screens and Windows for more info about 'more') Also see the Standard PowerVisor keys section.

1.13 Getting Started : The history buffer

PowerVisor has a history buffer so you can easily retrieve previous commands.

This can be handy if you want to repeat a command a few times, or if you want to correct an error in a commandline.

Starting with PowerVisor V1.10 this history buffer works exactly like the history buffer in the AmigaDOS 2.0 shell. There is one exception : PowerVisor does not support the standard <shift>+<up> key to search in the history. However, you can use this feature if you use the standard s:PowerVisor-startup file (the one provided with this release of PowerVisor). This startup script installs this feature for you (it uses the s:pv/SearchHist ML-script for that purpose).

For example type :

```
< help <enter>
> PowerVisor help (1.10 Beta)   Sun Sep 22 13:10:59 1991
> -----
> You can type one of the following for more information on a specific
> item:
>
>     help                for this screen
>     help general        for general information
>
>     help commands      for a list of all available commands
>     help functions     for a list of all available functions
>     help syslists      for a list of all available lists
>     help <command>_cmd  gives help for a specific command
>     help <command>_tmp  gives a command template (or <command> ?)
>     help <function>_func gives help for a specific function
>     help <list>_list    gives help for a specific list
>
>     help cmdline       shows all the commandline options available
>     help arguments     gives help for all possible argument types
>     help libfuncs      information about library functions
>     help bugs          for all bugs in the current version
>     help debugging     for general debugging help
>
> You can use abbreviations ('h gen' instead of 'help general')
> Note that you could get the wrong help when you do this.
> ('help li' will probably not give what you wanted: list, libfuncs, ...)
```

< <arrow up>

On the commandline appears 'help', this is the previous command.

Use the stringgadget key to go to the end of the line :

```
< <shift>+<arrow right>
```

```
< <space> general <enter>
```

Now you have executed the command 'help general'.

```

> Help general
> -----
> You can type one of the following for more information on a specific
> item:
>
>     help snap           for the screen snap feature
>     help keys          for information about keys
>     help input         for info about the input editing possibilities
>     help redirection   for redirection to a file
>     help files         for all the files PowerVisor uses
>     help historybuf    the history buffer
>     help portprint     for the portprint facility
>     help autodefauult  for the automatic default feature
>     help templates    for the template feature

```

You can use the <arrow up> and <arrow down> keys to scroll in the history buffer. Normally only 20 lines are remembered in the history buffer.

Try this :

```

< prefs history <enter>
> 00000014 , 20

```

```

< prefs history 100 <enter>

```

```

< prefs history <enter>
> 00000064 , 100

```

You see that you can change the maximum number of lines in the history with the `prefs` command. Note that setting another number clears the history buffer.

1.14 Getting Started : Making a Config file for PowerVisor

PowerVisor uses the `s:PowerVisor-config` file (if present) to set default preferences. This file contains, for example, the value of the 'mode' variable. If you have installed PowerVisor as you wish (with the `mode` command) you can use `saveconfig` to save the config file.

The `s:PowerVisor-config` file contains the following information :

- All things you can set with the `mode` command
- All things you can set with the `prefs` command

Some examples :

- The maximum length for a commandline
- Some default values for logical windows
- Some keydefinitions
- historybuffer length
- ...

I recommend that you work with the default values until you now more about PowerVisor. Especially the logical window preferences are difficult to set right if you are a first time user. If you think you can cope you can read `Installing PowerVisor` .

Note that 'saveconfig' also saves the state and position of all standard logical windows. See the 'mode intui' feature and the screen command for more info.

Warning! If an update for PowerVisor arrives it is possible that the format for the PowerVisor-config file is not valid anymore. Because of this it is probably better to delete the PowerVisor-config file when you have a new version and do the configuration again.

1.15 Getting Started : Standard PowerVisor keys

Interrupt PowerVisor at any time by pressing <esc>.

Press <right alt>+<right shift>+ / to bring the PowerVisor screen to the front (or reopen it if PowerVisor was in hold mode).

Using <left-alt> in combination with the numeric keypad you can scroll in the active logical window. With the <tab> key you can select the active logical window.

Using the <ctrl> key in combination with the numeric keypad you can scroll in the debug logical window.

To pause the output of PowerVisor use the <right-alt>+<help> key combination.

Note that almost all these keys can be reconfigured with the prefs command.

Also see The PowerVisor hot key and Interrupting PowerVisor .

1.16 Getting Started : Special files

PowerVisor uses the following files:

```
s:PowerVisor-startup
    script file with all the initialization commands you find
    useful. This file is not necessary
PowerVisor-menus (*)
    This file is only used by the AmigaDOS 2.0 version of
    PowerVisor. It contains the description of all menus
    used in PowerVisor. This file is not really needed, but
    you won't have any menus if you don't have this file
PowerVisor-help (*)
    This is the help file. If this file does not exist, you have no
    online help (with the help command. The ahelp ,
    cmdhelp , funhelp and index aliases are not affected.
    You can change the help file if you like. After you have changed
    it you must type (in the Cli or shell) :
        'makehelp PowerVisor-help PowerVisor-ctrl word 2'
    to update the PowerVisor-ctrl file
```

PowerVisor-ctrl (*)
This is the control file for the help file. Without this file you have no online help. See the QuickHelp manual for more details about this help format

PowerVisor-errors (*)
This file contains all error messages. When this file is not available PowerVisor will print error numbers rather than messages. You can change this file. Please make sure that each line in this file is 70 bytes long (return included)

s:PowerVisor-config
This file contains config information for PowerVisor. All things you can install with the mode and prefs commands are in this file. If this file is not present, default values are used

libs:powervisor.library
This is the portprint library. PowerVisor needs this library

s:pv/
This subdirectory is the preferred subdirectory for scripts and structure definition files

pv:docs/
This subdirectory is the directory that AmigaGuide uses when reading the PowerVisor hypertext manual. If you want to use AmigaGuide you must assign pv: to the root of the PowerVisor directory (containing the 'docs' subdirectory)

PowerVisor will search all filenames ending with (*) in the following subdirectories :

PROGDIR: (only for AmigaDOS 2.0) This is the subdirectory where the PowerVisor executable is located. If you have AmigaDOS 2.0 this is the recommended place for the (*) files

S: If you have AmigaDOS 1.3 or earlier, this is the recommended place for the (*) files

current If the above failed, PowerVisor will look in the current directory

1.17 Getting Started : A summary

In this section I mention the most useful commands of PowerVisor. These are the commands that you are probably going to use most. To get more information about these commands refer to Command Reference or look in the corresponding tutorial section. Note that all commands described here begin with some letters in uppercase and the rest in lowercase. All uppercase is the required part of the command. You may abbreviate the command until only the uppercase part is left. Note that this uppercase/lowercase convention is also used in the online help and in the 'Command Reference'.

General commands :

Quit
Help <subject>

Setting preferences :

```
MOde
PREfs
SAVEConfig
```

Looking at things :

```
Disp <expression>
List [<list name>]
Info <list element> [<list name>]
Memory [<start> [<bytes>]]
Unasm [<start> [<instructions>]]
```

Searching, clearing and copying memory :

```
Search <start> <bytes> <string>
Next
Fill <destination> <bytes> <fill with string>
Copy <source> <destination> <bytes>
```

Debugging :

```
DEBug <command> [<argument>]
TRace <command> [<argument>]
Break <command> [<argument>]
SYmbol <command> [<argument>]
SOurce <command> [<argument>]
```

Some useful aliases defined in s/PowerVisor-startup (See
Alias Reference) :

```
WHY Give a short help about the last error
LFD <library name> Load a fd-file for a library
FDEBUG Initialize the fullscreen debugger
```
