Screen.hyper

COLLABORATORS

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Screen.hyper

1.1 Screens and Windows (Tue Jul 14 16:08:11 1992)

Contents:

1.2 Screens and Windows : Commands used in this tutorial

active	Set active logical window
awin	Open the 'Rexx' logical window
color	Set RGB colors for PowerVisor screen
colrow	Set the number of columns and rows for a logical window
current	Make another logical window the current one
dwin	Open the 'Debug' logical window
fit	Fit the logical window to the visible size
home	Go to the home position of the logical window
list	Show a list (tasks, libraries, message ports, ...)
log	Log output to a file
memory	List memory
mode	Set PowerVisor preferences
move	Move a physical window
on	Execute command on other logical window
owin	Open the 'PPrint' logical window
prefs	Set preferences
refresh	Refresh a command on the 'Refresh' logical window
rwin	Open the 'Refresh' logical window
screen	Set PowerVisor on another screen
setfont	Set another font on a logical window
size	Size a physical window
swin	Open the 'Source' logical window
to	Redirect output to a file
xwin	Open the 'Extra' logical window

1.3 Screens and Windows : Functions used in this tutorial

cols	Ask the number of columns on a logical window
getactive	Ask the active logical window
getlwin	Ask the current logical window
getcol	Ask the prefered number of columns on a logical window
getrow	Ask the prefered number of rows on a logical window
lines	Ask the number of rows on a logical window

1.4 Screens and Windows : Introduction

The screen and window of PowerVisor are very customizable.
Interlace, pal, ntsc, vga, even the A2024 monitor, colors, fonts, ...
All these characteristics and more are customizable. Read this tutorial
for more information about these options.

```
All the examples in this chapter assume that you have an NTSC monitor.
If this is not the case replace 'NTSC' with 'PAL' and 'PAL' with 'NTSC'
(in your mind) when you read this chapter. I also assume that you have
started a vanilla PowerVisor. This means that there was no
s:PowerVisor-config and a minimal s:PowerVisor-startup when you started
PowerVisor.
```

1.5 Screens and Windows : The PowerVisor screen

```
                Normally PowerVisor uses an Intuition screen for output.
The default screen has the following characteristics :

  - 2 bitplanes (four colours)
  - non interlaced
  - PAL, NTSC, VGA or A2024 depending on your preferences settings
    if you use AmigaDOS 2.0.
    If you use AmigaDOS 1.2 or 1.3 PowerVisor will open a screen as big as
    the workbench screen would be if it is not interlaced.
  - Size is inherited from the preferences settings. PowerVisor uses
    overscan if you set overscan in preferences (only AmigaDOS 2.0).
  - Colours are inherited from the Workbench.
  - Topaz.font 8 is uses for all text (default).
  - The screen is a public screen if you have AmigaDOS 2.0 (with the
    name 'PowerVisorScreen').

The PowerVisor screen is partitioned in logical windows (see

                 The PowerVisor window system
                PowerVisor gives you the option to open its window on another ↩
                    screen, to
change the colours, switch to interlace, switch to another monitor (2.0)
or to use more bitplanes.

Note that in this tutorial we constantly talk about three different
things :

   - The PowerVisor screen : This is the Intuition screen. It is possible
     that this screen does not exist. In that case, PowerVisor resides
     on another screen.
   - The PowerVisor window : This is the Intuition window that normally
     lives on the screen. When we talk about the PowerVisor window we
     are talking about the physical window 'Main'. PowerVisor can have
     more physical windows.
   - Physical windows : These correspond directly with Intuition windows.
   - Logical Windows : Each physical window is partitioned in logical
     windows, do not confuse a logical window with an Intuition window.
```

1.6 Screens and Windows : Interlace and monitors

```
                If the PowerVisor screen is open (default when you start ↩
                    PowerVisor),
you can switch to interlace or to another monitor using the  mode
command (This command is also used for other settings, see the
appropriate documentation). The 'mode' command with screen arguments
has no effect when PowerVisor resides on another screen.

The following 'mode' arguments have something to do with screens or
windows :

   - lace         switch to interlace
   - nolace       switch to non interlace (default)

   - pal          pal monitor (AmigaDOS 2.0)       640x256 or 640x512
   - ntsc         ntsc monitor (AmigaDOS 2.0)      640x200 or 640x400
   - vga          vga monitor (AmigaDOS 2.0)       640x480 or 640x960
   - viking       a2024 monitor (AmigaDOS 2.0)     1024x1008

   - fancy        use two bitplanes (default)
   - nofancy      use only one bitplane

   - sbottom      sizegadget is included in bottom border (default)
                  this option is only useful if there are more physical
                  windows, or if the 'Main' physical window is a
                  non-backdrop window.
   - nosbottom    sizegadget is included in right border

Here are some examples :

< mode lace nofancy <enter>

You will now get a one bitplane interlace screen (if you have enough
memory).

Note that PowerVisor only uses half the interlaced screen for output
(try some commands with a lot of output to test this: 'help commands',...).
This is normal. We will see later how you can make the logical window
full size again (in the
                 The PowerVisor window system
                 section).

Back to normal with :

< mode fancy nolace <enter>
```

1.7 Screens and Windows : Setting the colours

```
Try :

< color 0 0 0 7 <enter>
< color 1 15 15 15 <enter>

This will install a blue background and a white foreground. (Note that
this command only works if PowerVisor is on its own screen (see later for
more info)).
```

1.8 Screens and Windows : PowerVisor on other screen

You can open the PowerVisor window on each screen available in the system. However, when you do this you must be very careful NOT to close the screen where PowerVisor resides.

Example :

List all screens :

```
< list scrs <enter>
> Screen name         : Address  Left  Top Width Height FirstWindow
> ----------------------------------------------------------------------------
> PowerVisor   (V1.00/: 07EA28F0    0    0   692    442 07EA67B8
> Workbench Screen    : 07E280D0    0 -582   692   1024 07E1F0F8
```

For example, let's open on the Workbench screen :

```
< screen Workbench <enter>
```

You can now change the size of the PowerVisor window :

```
< size main 600 150 <enter>
```

This command sizes the specified physical window ('Main' in this case). The size command only works on non-backdrop windows.

You can also move the PowerVisor window with :

```
< move main 10 10 <enter>
```

You can also resize the PowerVisor window using the size gadget.
The screen command normally opens a window with the same size as the previous size. If this is too big for the new screen, PowerVisor will make the window as big as possible.

When PowerVisor is on another screen, you cannot use the following commands :

- color
- mode with one of the following arguments :
  fancy,nofancy,pal,ntsc,vga,viking,lace,nolace

If you are configuring PowerVisor in a special way you can also make the 'Main' physical window a non-backdrop window on the PowerVisor screen :

```
< screen 0 <enter>
```

Now you can size and move the PowerVisor window.

Go back to the PowerVisor intuition screen :

```
< screen <enter>
```

Note that the 'screen' commands moves all physical windows (see later) present.

1.9 Screens and Windows : The PowerVisor window system

The PowerVisor window system is fairly complex. At the hart of the system you have the physical windows. These correspond directly with Intuition windows. Normally there is only one physical window called 'Main'. This window also contains the stringgadget. You can open more physical windows if you want (see later).

Each physical window has a tree of boxes. A box is some space that can later be used by logical windows. By default there is only one box on a physical window. This box is called the masterbox and is always present (you can't remove this box). When you open more logical windows on a physical windows the appropriate boxes are made automatically, so you generally need not concern yourselves with these objects.

A box is not always used by a logical window. Sometimes it is used by two other boxes. For example: if you want three logical windows, two above each other and the third one right of the previous two. You now seem to have three boxes :

```
                  +---------+-------+
                  |         |       |
                  |         |       |
                  |    1    |       |
                  |         |       |
                  |         |       |
                  +---------+   3   |
                  |         |       |
                  |         |       |
                  |    2    |       |
                  |         |       |
                  |         |       |
                  +---------+-------+
```

In this example there are three boxes containing a logical window. But there are in fact five boxes on the physical window.
First the masterbox :

```
                  +-----------------+
                  |                 |
                  |                 |
                  |                 |
                  |                 |
                  |                 |
                  |        4        |
                  |                 |
                  |                 |
                  |                 |
                  |                 |
                  |                 |
                  +-----------------+
```

The masterbox contains two boxes :

```
                  +---------+-------+
                  |         |       |
                   |         |       |
                   |         |       |
                   |         |       |
                   |         |       |
                   |    5    |   3   |
                   |         |       |
                   |         |       |
                   |         |       |
                   |         |       |
                   |         |       |
                   +---------+-------+
```

Box 5 contains two other boxes and box 3 contains a logical window.

It is easy to see the tree structure for the boxes. Box 4 is the masterbox
and only the leaves of the tree contain logical windows.

```
                            +---+
                            | 4 |
                            +---+
                             / \
                            /   \
                           /     \
                        +---+   +---+
                        | 5 |   | 3 |
                        +---+   +---+
                         / \
                        /   \
                       /     \
                    +---+   +---+
                    | 1 |   | 2 |
                    +---+   +---+
```

All boxes containing two other boxes automatically draw a size bar.
By dragging this size bar you can change the size of the two child
boxes. For example, dragging the vertical size bar between box 5
and box 3 (thus managed by box 4) changes the size of box 5,3,1 and 2.
Dragging the horizontal size bar between box 1 and 2 (managed by
box 5) only changes the size of box 1 and 2. Note that each parent
box remembers the size for the two children with one number: this number
is the percentage (x10) that child A may use of the parent box. This
means that if you change the size of a box, all children are resized
proportionally.

When you add a logical window using the appropriate commands (see later),
PowerVisor will add two boxes. For example, if you want to add a logical
window below all boxes already visible, the tree will look as follows
(box 6 and 7 are new) :

```
                            +---+
                            | 6 |
                            +---+
                             / \
                            /   \
                           /     \
                        +---+   +---+
                        | 7 |   | 4 |
                        +---+   +---+
                                 / \
                                /   \
                               /     \
                            +---+   +---+
                            | 5 |   | 3 |
                            +---+   +---+
                             / \
                            /   \
                           /     \
                        +---+   +---+
                        | 1 |   | 2 |
                        +---+   +---+
```

(Note that the masterbox has changed)
The physical window changes to :

```
                  +---------+-------+
                  |         |       |
                  |    1    |       |
                  |         |       |
                  +---------+   3   |
                  |         |       |
                  |    2    |       |
                  |         |       |
                  +---------+-------+
                  |                 |
                  |        7        |
                  |                 |
                  +-----------------+
```

A logical window is the object you are probably going to use most. It can contain text. To know where the text must appear in the physical window, the logical window uses a box. The standard logical window for output is the 'Main' logical window. This logical window always resides on the 'Main' physical window (but there can be more logical windows on the 'Main' physical window). There are seven predefined logical windows :

- Main	for normal output
- Extra	for output
- Refresh	if present, all output from the refresh command goes to this logical window. Otherwise the output goes to the current logical window
- Debug	for the fullscreen debugger
- Rexx	if present, all output from a rexx command goes to this logical window. Otherwise the output goes to the current logical window
- PPrint	if present, all PortPrint messages go to this logical window. Otherwise the output goes to the current logical window
- Source	The source logical window used by the sourcelevel debugger

You can add your own logical windows.

All physical windows currently present can be found in the 'pwin' list.

All logical window can be found in the 'lwin' list.

1.10 Screens and Windows : The size of a logical window

A logical window contains text. It has a certain number of rows and a certain number of columns. Normally this number is independent of the visible size of the logical window. What this means is that the number of columns and rows remains the same even if you switch to interlace or if you change the size of a logical window (by dragging the size bar).
This is the reason that PowerVisor only uses half the screen when you switch from non-interlaced to interlaced. There are several solutions to this problem :

- If you want the number of columns and rows automatically fit for the visible size you can make the logical window autoscalable. You can do this with the colrow command (see later). The disadvantage of this is that the logical window will be cleared everytime the visible size changes. If you want you can also make the logical window autoscalable for the number of rows only, or for the number of columns only.

- If interlace is your default screen type (defined with the mode and the saveconfig commands) the 'Main' logical window will be bigger. If you switch to non-interlace, the number of columns and rows will remain the same. This means that you can scroll in the logical window (see later).

- You can explicitelly tell PowerVisor to fit the 'Main' logical window to the visible size with the fit command (see later). This is not the same as the first method since the window will have a fixed size (not autoscalable).

- You can explicitelly set the size of the 'Main' logical window using the colrow command. If this size is too big for the visible size, you will be able to scroll in the logical window.

The first, second and fourth methods can be made permanent with the mode , prefs and saveconfig commands
(see Installing PowerVisor).

Note that this discussion is also valid for other logical windows as well.
All the standard logical windows behave a bit different (you can change this behaviour with the 'prefs' command) :

- Main	number of columns and rows is set to a fixed value. This value is the maximum number of columns and rows at the time the logical window is created.
- Extra	like 'Main'
- Refresh	the number of columns is set to a fixed value. This value is the maximum number of columns at the time the logical window is created. The number of rows is fixed and always set to 50.
- Debug	the number of columns is fixed and set to 90. The number of rows is fixed and set to 42.
- Rexx	Like 'Refresh'
- PPrint	Like 'Refresh'

```
   - Source    Like 'Debug' except that the number of columns and rows
               are autoscalable
```

1.11 Screens and Windows : Standard behaviour for logical windows

```
All logical windows can behave different. Here are the possible behaviours
for logical windows :

   -MORE- enabled or disabled
               When enabled, PowerVisor will pause when there is a full
               window of output.
   Interrupt/Pause enabled or disabled
               When enabled you can use the <esc> and <right-alt>+<help>
               keys to interrupt or pause PowerVisor when PowerVisor is
               sending output to the logical window.
   Home position is top-visible or real-top
               PowerVisor maintains a home position for each logical
               window. This position is either the real top of the logical
               window or the first line that is visible (starting from
               above) of the bottom visible half of the logical window.
               When a top-visible logical window is cleared, PowerVisor
               will scroll to the bottom part of the logical window and set
               the current cursor position to the first line of this
               visible part.
               When a real-top logical window is cleared, PowerVisor will
               scroll to the top part of the logical window and set the
               current cursor position to the first line.
               There is yet another difference between top-visible and
               real-top logical windows. When the visible size of a logical
               window changes (this does not always imply a change of the
               number of rows and columns), PowerVisor will try to keep
               the top visible line (for real-top logical windows) or the
               bottom visible line (for top-visible logical windows) on the
               same visible position.
   Status line on/off
               The statusline is the bar at the top of a logical window.
   Auto Output Snap on/off
               When enabled, PowerVisor will automatically scroll the
               logical window to the position of the appearing output. This
               means that you will always see all new output on the logical
               window. When disabled, PowerVisor will not scroll and output
               may appear off screen. Note that this flag is only useful
               when the logical window is bigger than the current visible
               size.

The standard logical windows have the following behaviour :

   - Main      -MORE- enabled/disabled depending on the setting of
               the  mode  command (see later)
               Interrupt/Pause enabled
               Home position is top-visible
               Status line on
               Auto Output Snap is on
   - Extra     -MORE- disabled
               Interrupt/Pause enabled
               Home position is top-visible
               Status line on
               Auto Output Snap is on
   - Refresh   -MORE- disabled
               Interrupt/Pause disabled
               Home position is real-top
               Status line on
               Auto Output Snap is off
   - Debug     -MORE- disabled
               Interrupt/Pause disabled
               Home position is real-top
               Status line on
               Auto Output Snap is off
   - Rexx      -MORE- disabled
               Interrupt/Pause disabled
               Home position is top-visible
               Status line on
               Auto Output Snap is off
   - PPrint    -MORE- disabled
               Interrupt/Pause disabled
               Home position is top-visible
               Status line on
               Auto Output Snap is off
   - Source    -MORE- disabled
               Interrupt/Pause disabled
               Home position is real-top
               Status line on
               Auto Output Snap is off
```

All other logical windows have the following default behaviour :

```
   -MORE- disabled
   Interrupt/Pause enabled
   Home position is top-visible
   Status line on
   Auto Output Snap is off
```

You can change the default behaviour for the standard logical windows with the prefs command. You can change the behaviour for each logical window with the setflags command (see later).

1.12 Screens and Windows : Opening standard logical windows

There are predefined commands to open the standard logical windows :

```
   -  xwin    Extra
   -  dwin    Debug
   -  rwin    Refresh
   -  awin    Rexx
   -  owin    PPrint
   -  swin    Source
```

These commands open/close the specified logical window at the top. This means that they split the masterbox and add a new box above all other logical windows. This box gets 30 % of the total physical window height.

```
The logical windows in the remaining 70 % are shrinked accordingly. These
predefined commands always open the window on the 'Main' physical window.

These commands also have an optional argument to indicate the number of
lines you want in the logical window.

If the 'intui' mode flag (set with the  mode  command) is set, then these
commands will also create a physical window with the same name as the
standard logical window. The logical window is opened in this new
physical window. The optional argument is ignored if this is the case.


Some examples :

< xwin <enter>

Now we have two logical windows on our 'Main' physical window. These two
logical windows are called 'Main' and 'Extra'. You can change the size
of the windows by dragging the horizontal bar.

You can now use this window instead of 'Main' for output
(with the  current  command) :

< current extra <enter>

Type some command :

< list task <enter>
> Task node name      : Node     Pri StackPtr  StackS Stat Command         Acc
> ---------------------------------------------------------------------------
> Background Process  : 07E28330 00  07E2D500    4096 Wait iprefs     (02) -
> PowerSnap 1.0 by Nic: 07E51228 05  07E51A72    2000 Wait            PROC -
> Background Process  : 07E5B3E8 00  07E5AD92    4096 Wait addtools   (06) -
> SYS:System/CLI      : 07E5CAA8 00  07E5D9E6    4096 Wait            (00) -
> * Blanker           : 07E605D8 00  07E615E4    4000 Wait            PROC -
> RexxMaster          : 07E4AD58 04  07E4B59A    2048 Wait            (00) -
> ...

The output appears on 'Extra'.

< current main <enter>

Now all following output will appear on 'Main'.

You can also use the  on  command :

< on extra list task <enter>
> Task node name      : Node     Pri StackPtr  StackS Stat Command         Acc
> ---------------------------------------------------------------------------
> Background Process  : 07E28330 00  07E2D500    4096 Wait iprefs     (02) -
> PowerSnap 1.0 by Nic: 07E51228 05  07E51A72    2000 Wait            PROC -
> Background Process  : 07E5B3E8 00  07E5AD92    4096 Wait addtools   (06) -
> SYS:System/CLI      : 07E5CAA8 00  07E5D9E6    4096 Wait            (00) -
> * Blanker           : 07E605D8 00  07E615E4    4000 Wait            PROC -
> RexxMaster          : 07E4AD58 04  07E4B59A    2048 Wait            (00) -
> ...
```

This command temporarily sets the current logical window to the parameter supplied. It then executes the following command ('list task' in this example).

To see all logical windows you can list them :

```
< list lwin <enter>
> Logical Window      : Node
> -----------------------------------------------------------------------------
> Extra               : 07EBCB20
> Main                : 07E25A60
```

Close 'Extra' with :

```
< xwin <enter>
```

1.13 Screens and Windows : Opening logical windows in general

Instead of using the predefined commands to open the standard logical windows you can also use the more powerful openlw and closelw commands. These commands can also be used to open other logical windows.

Some examples :

To open the 'Extra' logical window right from the 'Main' logical window (instead of above the 'Main' logical window) you can use :

```
< openlw main extra 80 40 main r <enter>
```

The first argument is the physical window where we want to open the new logical window. The second argument is the name of the logical window. The two following arguments are the number of columns and rows. If you want an autoscale logical window you can use -1 for one or both of these arguments. The two last arguments specify where you want to open the 'Extra' logical window. In this case we opened it at the right ('r') of the 'Main' logical window.

Do not close this logical window yet. We will first open a third :

```
< openlw main testwindow 100 -1 extra u <enter>
```

This window is autoscalable for the height. This means that when you change the horizontal visible size nothing will happen, but if you change the visible vertical size the window will be cleared and the number of columns will change.
Now we have three logical windows on the 'Main' physical window.

Suppose we wanted to open a logical window at the bottom of the physical window. This can be done with :

```
< openlw main bottomwindow 80 40 main pd <enter>
```

The 'pd' argument means that we first take the parent and then go down. You will probably understand how this works when you think how PowerVisor manages logical windows and boxes. When you said something like 'extra u' some commands ago, PowerVisor interpreted this as : take the box containing the ’extra’ logical window. Split this box (make a new parent instead of the old box and make the existing ’extra’ box a child of this new parent, also create a new box as the brother of the ’extra’ box) and put the new logical window above the ’extra’ logical window. When you say something like ’main pd’ in the previous command, PowerVisor interpretes this as : take the box containing the ’main’ logical window. Take the parent of this box (the ’p’ stands for parent) and perform the same action as described before on this parent box.

You can use as many p’s in front of the direction argument as you wish. You can use u (up), d (down), r (right) or l (left) for direction arguments.

To close all logical windows use :

```
< closelw extra <enter>
or
< xwin <enter>

< closelw testwindow <enter>
< closelw bottomwindow <enter>
```

Now we are back with only one logical window: ’Main’.

1.14 Screens and Windows : Opening physical windows

You are not limited to the default physical window ’Main’. You can open five additional physical windows (This limitation stems from the fact that PowerVisor only has 5 remaining signals. In a later release of PowerVisor this limitation will probably be removed). These physical windows can contain as many logical windows as memory permits.

For example, open a physical windows and two logical windows in it (with the openpw command) :

```
< openpw test 0 0 400 150 <enter>
```

You now have an extra window on position (0,0), width 400 and height 150.

```
< openlw test leftwin 80 40 <enter>
```

Note that for the first logical window on a physical window you need not give positional arguments.

```
< openlw test rightwin 80 40 leftwin r <enter>
```

Some tests :

```
< on leftwin list wins <enter>
> ...

< on rightwin list scrs <enter>
> ...
```

You need not close the logical windows. If you close the physical window
the logical windows are closed automatically
(use the closepw command) :

```
< closepw test <enter>
```

1.15 Screens and Windows : Active versus current

The current logical window is simply the window receiving all ↩
output from
the commands you type on the commandline. The default current logical
window is 'Main'. With the current command you can change this
permanently or you can use the on command to execute one command (or
more using groups) on another current logical window.

The active logical window is the logical window with the full size bar
(blue if you have the standard AmigaDOS 2.0 palette and if you use 2
bitplanes). You can scroll the active logical window
(see
Scrolling in logical windows
).
The most important distinction (note that this is new for version V1.10
of PowerVisor) is the fact that input is not directed to the current
logical window but to the active one. For normal commandline mode this
makes no difference because all logical windows will accept the input,
execute the command and send the output to the current logical window. But
what happens if a certain command executing with a certain current logical
window asks for input? PowerVisor will block all other logical windows and
only accept input if the active logical window is the one waiting for
input.

An example :

Assume that you are running an ARexx script and you have the 'Rexx'
logical window open. At some intervals this script sends output to
the 'Rexx' window. You are simply typing commands and looking at the
output on the 'Main' logical window (the current logical window).
Note that when ARexx executes a command, the current logical window
is temporarily set to 'Rexx'.

The 'Main' logical window is also the active one.

At some moment the ARexx script needs input. At once you will not be
able to type. The commandline will be locked and the current list
indicator will have changed to '------'. If you see this you know that
some other logical window is waiting for input. With the <tab> key you
make the 'Rexx' logical window active. The current list indicator
changes to '-HALT-' or '????' or some other string depending on the type
of input the ARexx program wants. The input is unlocked and you can type
the input needed for the ARexx program.

In summary the following things are valid for a current logical window :

- all output appears there (note that some commands executed in special

```
     environments like ARexx have other current logical windows)
   - when a command executing on a certain current logical window wants
     input, all other logical windows are locked (the current list
     indicator changes to '------' if one of these logical windows is
     active)

The following things are valid for an active logical window :

   - you can scroll the active logical window
   - you can interrupt the command (with <esc>) when the active logical
     window is equal to the current logical window for that command
   - you can pause the output (with <right-alt>+<help>) when the active
     logical window is equal to the current logical window for that
     command
   - the active logical window also determines the state of the current
     list indicator and the possibility to use input for that logical
     window


The following current list indicators are used in PowerVisor :

   -MORE-   the command has just printed a full page of output, press space
            to continue the output (see
                 More
                )
   -HALT-   you have paused the command with <right-alt>+<help> or the
            command is waiting for a key (with the  key()  function)
   -BUSY-   you can't use input in any logical window, PowerVisor is busy
            with something (executing a command for example)
   ------   the logical window is locked for input. This means that there
            is another logical window waiting for input (cycle with <tab>
            until you find that other logical window)
   ????>    some command is waiting for a full line of input (with the
             scan  command)
   Task>    'Task', 'Libs', ... or some other current list simply indicates
            that you are in normal command executing mode. All logical
            window accept input in this mode but output is always directed
            to the current logical window
   ?        something else. This is the same as with '????>' but you can
            change the current list indicator with the 'scan' command


Note. There is a bug in PowerVisor for AmigaDOS 1.2/1.3. I have not been
able to find a way to unactivate (unselect) a stringgadget. So you will
sometimes have to unselect the gadget yourselves (press enter or click
on the window) before you can press the key. I'm sorry for this.
In AmigaDOS 2.0 everything is fine.
```

1.16 Screens and Windows : Scrolling in logical windows

```
                As was mentioned before, the number of columns and rows in a  ←
                   logical window
can be greater than the visible number of columns and rows. If this is
the case you can scroll in the logical window.
```

```
The window that will scroll when you use the appropriate keys (see below)
is the one that is active (see
                 Active versus current
                ).

The following keys are defined (numeric keypad) :

   - <l-alt>+<home>   (7)  go to the top left position in the logical
                           window
   - <l-alt>+<end>    (1)  go to the bottom left position
   - <l-alt>+<pgup>   (9)  scroll 5 lines up
   - <l-alt>+<pgdn>   (3)  scroll 5 lines down
   - <l-alt>+<arrows> (2,4,6,8)
                           scroll one line/column in the right direction
   - <l-alt>+<cntr>   (5)  go to the complete right
   - <tab>                 make the next logical window active


To let you experiment with all keys try the following :

< colrow main 100 80 <enter>

 colrow  sets the number of columns and rows to 100 and 80
resp. Now put some output on the screen (with  memory  or  list )
and scroll in all directions.

Note that the little box in the statusline changes when you scroll.
This box is an indicator of where you are in the logical window. If you
can't scroll (because the number of columns and rows is less or equal than
the number of columns and rows visible) the box will be full. Otherwise
it represents the position of the visible size of the logical window.
```

1.17 Screens and Windows : Setting the font

```
You can install a different non-proportional font for each logical window.
The default font is always 'topaz 8'.

Open the 'Debug' window :

< dwin <enter>

(Ignore the 'task not loaded' message, you will need this later when you
start debugging).
Use  setfont  to set the font :

< setfont debug topaz.font 9 <enter>

You will see that the size of the letters change.
The fonts you want to use must be either memory resident or available in
the 'fonts:' directory.

< setfont main courier.font 13 <enter>

(You must have a non-proportional courier.font in your 'fonts:' directory
to do this).
```

To restore everything type :

```
< dwin <enter>
< setfont main topaz.font 8 <enter>
```

1.18 Screens and Windows : The snap feature

(Also see Snapping away).

If you click anywhere on the PowerVisor window (except on size bars), PowerVisor will ’snap’ the word under the mousepointer to the commandline. If there is no word under the mousepointer nothing happens.

This feature works on all logical windows regardless of their size, font, number of columns, ...

Note that the snap will not happen if the window is just made active. You must click twice if the window is not active and you want to snap something.

The ’snapping’ feature can behave in different ways. You can use the mode command to set the behaviour you like most.

1.19 Screens and Windows : Refreshing

The ’Refresh’ logical window can be used together with the refresh command.

Open the ’Refresh’ window and make the window big enough :

```
< rwin <enter>
```

Start the refresh of the current list :

```
< refresh 10 {home;list} <enter>
```

(See the Expressions chapter for more info about the grouping operator ’{}’).

This ’refresh’ command will execute home and list one time each second and send the output to the ’Refresh’ logical window. Using the ’tab’ key and the numeric keypad keys (with left-alt) you can now scroll in this refresh display.

To disable the refresh use :

```
< refresh 0 <enter>
```

and close ’Refresh’ with :

```
< rwin <enter>
```

1.20 Screens and Windows : Redirection

If you want to redirect all output of a logical window to a file you can
use the log command :

```
< log main file <enter>
```

Now all output that appears on 'Main' is also sent to the file 'file'.

You can stop the redirection with :

```
< log <enter>
```

or 'log' with another logical window, since there can only be one log
file active at the same time.

If you are logging output to a file it can be useful to have no output on
the PowerVisor screen. You can accomplish this with :

```
< -list <enter>
```

When you precede a line with '-', PowerVisor will send the output from the
following command to void. Except when you have logging enabled.

If you want to temporarily discard the feedback (the reprint of the
executed command on the screen) you can type :

```
< ~list <enter>
> ...
```

This '~' operator is very useful if you want to attach a command to a key.
If you precede the command in this attachment with a '~', PowerVisor will
execute the command without showing it on the screen.

You can also disable this feedback for all commands you type with
the mode command :

```
< mode nofb <enter>
```

To enable it type (this is default) :

```
< mode fb <enter>
```

If you want to combine these two operators you must use the following order
(See Technical Information for more information about commandline
parsing) :

```
< ~-list <enter>
```

If you only want the output from one command in a file you can use the
to command :

```
< to ram:MyOutputFile list task <enter>
> ...
```

The output will still appear on the current logical window. This command temporarily works like the ’log’ command. The real log file is restored after this command exits.
If you only want output in a file you can use :

```
< -to ram:MyOutputFile list task <enter>
```

or

```
< to ram:MyOutputFile -list task <enter>
```

You can also combine the ’to’ and the on command :

First open the ’Extra’ window (if it is not already open) :

```
< xwin <enter>

< to ram:MyOutputFile on extra list task <enter>
> ...
```

This command will list all tasks on the ’Extra’ logical window. No output will be written to the file since the ’to’ command only redirects the output from the current logical window. However :

```
< on extra to ram:MyOutputFile list task <enter>
> ...
```

will also list all task on the extra window. The difference is that this time there will be output in the file since the ’to’ command redirects the output from the current logical window. At the time the ’to’ command is executed, this logical window is equal to ’Extra’.

1.21 Screens and Windows : Pens

You can change the color pens used for various drawing elements with the ’prefs pens’ command (prefs). See Installing PowerVisor for more information.

1.22 Screens and Windows : More ...

If the output of a specific command is too big, PowerVisor will wait and display a prompt. Try this :

Make the number of columns and rows for ’Main’ just big enough with the fit command :

```
< fit main <enter>
```

List a lot of memory :

```
< memory 0 10000 <enter>
> 00000000: 00000000  07E007CC  00F80834  00F80B16          ...........4....
> 00000010: 00F80ADA  00F80ADC  00F80ADE  00F80AE0          ................
> 00000020: 00F80C00  00F80AE4  00F80AE7  00F80AE8          ................
> 00000030: 00F80AEA  00F80AEC  00F80AEE  00F80AF0          ................
> 00000040: 00F80AF2  00F80AF4  00F80AF6  00F80AF8          ................
> 00000050: 00F80AFA  00F80AFC  00F80AFE  00F80B00          ................
> 00000060: 00F80B02  00F810F4  00F81152  00F81188          ...........R....
> 00000070: 00F811E6  00F8127C  00F812C6  00F81310          .......|........
> 00000080: 00F80B70  00F80B72  00F80B74  00F80B76          ...p...r...t...v
> 00000090: 00F80B78  00F80B7A  00F80B7C  00F80B7E          ...x...z...|...~
> 000000A0: 00F80B80  00F80B82  00F80B84  00F80B86          ................
> ...
```

After each page of output PowerVisor will display a prompt. Press any key to continue the output. Press <esc> abort the output.

```
< <esc>
> Break...
```

Note that PowerVisor will display this prompt after each page. A page is defined as the number of lines in the logical window. All these lines do not have to be visible. For example, if the 'Main' logical window is bigger than the visible size on screen, PowerVisor will only display a prompt after the TOTAL number of lines has passed. You can always scroll back to view the rest of the output if you want.

You can disable this prompt with :

```
< mode nomore <enter>
```

And enable it with :

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

The 'Main' logical window is the only window with a '-MORE-' prompt. All the other logical windows simply scroll until you interrupt them in one way or another. (Note that you can change this behaviour if you want to with the prefs command (see Installing PowerVisor)).