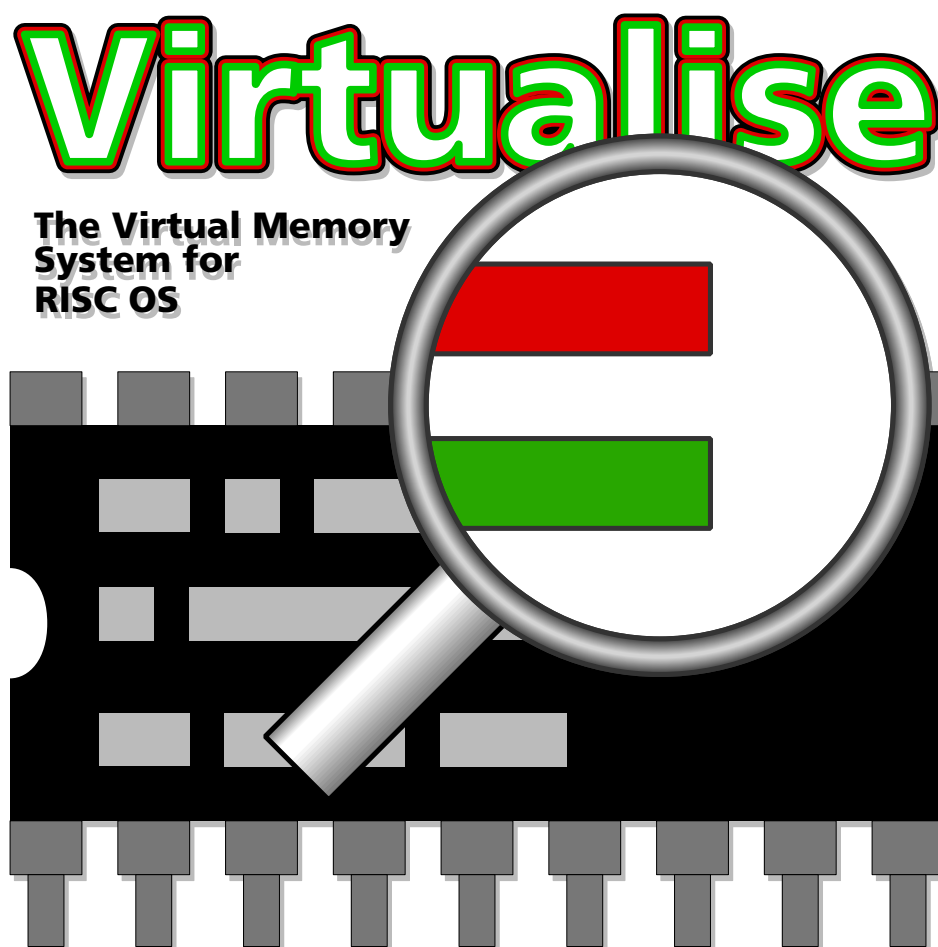




Virtualise

Virtual Memory for RISC OS 3 and RISC OS 4

ProAction

3QD
DEVELOPMENTS LTD

APDL

Virtualise

The "Virtualise" module and "Virtualis" front-end are
both © Alexander Thoukydides 2026

The marketing rights are owned by APDL.
Virtualise may be freely used by the single user it is licensed to.

The "Ensure" utility included in the front-end is copyright Alexander Thoukydides and has been released as
FreeWare. See the "Ensure" file for more details.

Copyright Alexander Thoukydides & APDL 2026
3QD Developments Ltd 2026
All rights reserved.

No part of this product may be reproduced in whole or part by any means without written permission of the
publisher. Unauthorised hiring, renting, lending, public performance or broadcasting of this product or its parts
is prohibited.

While every care is taken, the publisher cannot be held responsible for any errors in this product, or for loss of
data or consequential effects from the use of this package.

Software and Documentation supplied by
APDL
a part of
3QD Developments Ltd
Single User version downloaded from the free of charge software section of
www.riscos.com

***Please note that any free of charge downloads from RISCOS.com are supplied without any technical support
package as per the licence agreement.***

This document last edited on 29/09/26



If you have found this application and this manual useful please consider making
a donation to help cover the costs of cataloguing, documenting, updating and
hosting the APDL applications. Scanning the QR code will take you to our
donations page.

Contents

Virtualise	5
Introduction	5
Installing the Software	5
Starting the Front End	5
Configuring Virtual Memory	5
Enabling Virtual Memory for an Application	6
Modifying an Application to use Virtual Memory by default	7
Quitting the front end	7
Other Features	7
 The Desktop FrontEnd	 8
Introduction	8
Starting the Program	8
The Icon Bar	8
Dynamic Area List Window	9
Modify Application Dialogue	11
Page Status Window	12
Configuration Window	12
Page Faults Window	14
Automatic Virtual Memory Window	15
Task Swapper Window	16
 The software SWIs	 17
Introduction	17
Changes to OS SWIs	17
SWIs provided by Virtualise	18
 Appendices	 25
Glossary of Terms	25
Version History	26
Acknowledgments	27

Virtualise

Introduction

This introduction provides a straightforward introduction to using the Virtualise system to provide virtual memory for existing applications. A reference guide to all aspects of the program are included in the "Desktop FrontEnd" section of this manual.

If this is the first time that it has been used then it is first necessary to install the software as described in the next section, otherwise skip to the section entitled "Starting the Front End".

It is assumed that you know how to use the mouse and windows environment on your computer. If this is not the case then please read the Welcome guide which came with your computer before continuing.

Installing the Software

Virtualise is suitable for use on all versions of RISC OS from 3.5 to 4.39 (RISC OS Adjust). Virtualise is not suitable for use on RISC OS 6, or 32bit versions of RISC OS.

RISC OS 3.5 Installation

Before the program will run on RISC OS 3.5 it is necessary to merge the supplied !System folder (in the RISCOS350 directory) with the one on your RiscPC's hard disc. See the text file called "Important" for the instructions on how to do this.

Harddisc Installation

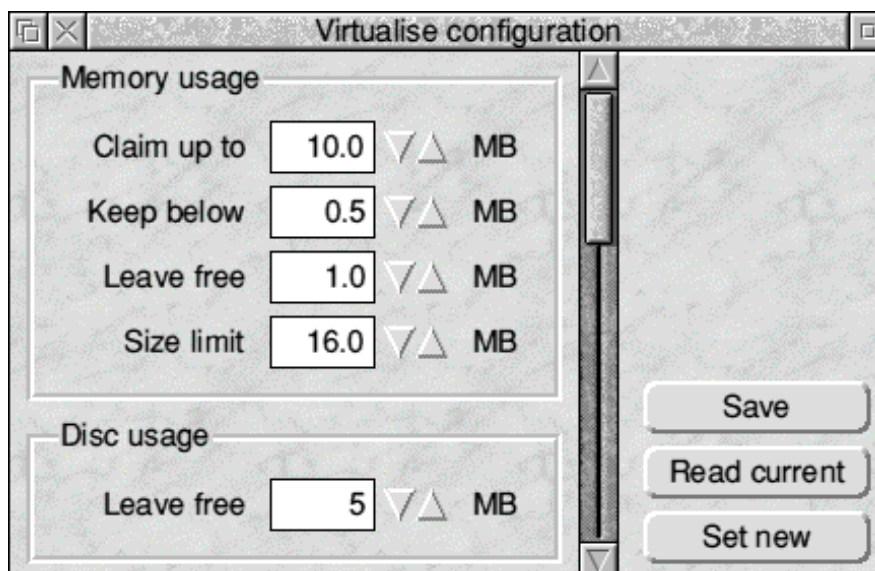
The application may be installed on the computer's hard disc in the standard RISC OS manner by simply dragging the "!Virtualis" icon to a suitable directory viewer.

Starting the Front End

The front-end supports interactive help; use Acorn's !Help to obtain information about the item under the mouse pointer. After a brief delay the application icon will appear on the icon-bar.

Configuring Virtual Memory

Click ADJUST on the !Virtualis icon on the icon-bar. This will open the "Virtualise configuration window". The most important settings are the three numbers at the top of the window which control memory usage of the virtual memory system:



These three numbers are:

Claim up to:

The amount of real memory to use. Larger values (up to the total memory in the computer) speed up most programs that use virtual memory by reducing the number of disc accesses required.

Leave free: The minimum amount of memory to leave free. When memory is required for another task the virtual memory system will attempt to change its usage to ensure that the specified amount is left unused.

Size limit:

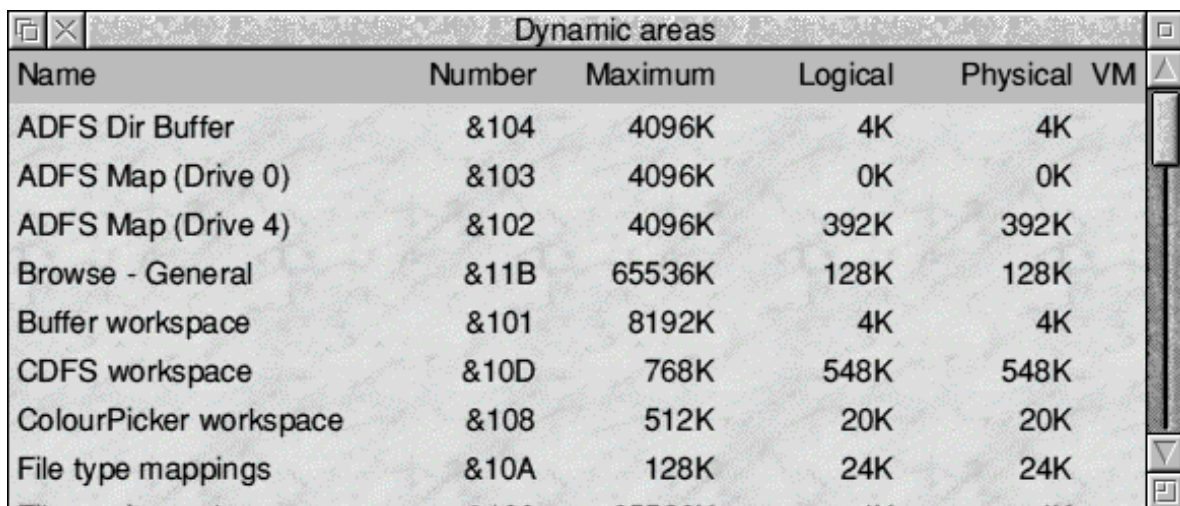
Many programs do not specify a maximum size for dynamic areas they create. In such cases the operating system normally sets the maximum size to be the same as the amount of memory in the computer. If virtual memory is going to be used this becomes very restrictive since the maximum size can not be extended after the area has been created. This option allows a different maximum size of be used.

Note that these values can only be specified to the nearest 0.1MB. The other options within the window are described in the "FrontEnd" file.

To ensure that these values are used the next time !Virtualis is started click SELECT on the "Save icon" at the bottom of the window. Then click SELECT on the "Set new" icon at the bottom of the window to use the values shown.

Enabling Virtual Memory for an Application

Click SELECT on the !Virtualis icon to open a window listing all of the dynamic areas in the computer. There is quite a lot of information in the window, but for now only the names of the dynamic areas down the left-hand side are important. All of the values are explained in the "FrontEnd" file.



Name	Number	Maximum	Logical	Physical	VM
ADFS Dir Buffer	&104	4096K	4K	4K	
ADFS Map (Drive 0)	&103	4096K	0K	0K	
ADFS Map (Drive 4)	&102	4096K	392K	392K	
Browse - General	&11B	65536K	128K	128K	
Buffer workspace	&101	8192K	4K	4K	
CDFS workspace	&10D	768K	548K	548K	
ColourPicker workspace	&108	512K	20K	20K	
File type mappings	&10A	128K	24K	24K	

Click on the toggle size icon in the top-right-hand corner to expand the window to its full size.

Next load (in the usual way) the application which you wish to use virtual memory with, e.g. !ChangeFSI which is supplied with the computer.

Find the entries for the dynamic areas created by the application you have started, and select them by clicking SELECT on the first one, and ADJUST for each subsequent dynamic area.

If you are using !ChangeFSI then the relevant areas are only created when a file is loaded. Drag a file to the !ChangeFSI icon, and after the window has opened there will be a "ChangeFSI sprite" dynamic area created. When all of the required dynamic areas have been selected click MENU within the window.

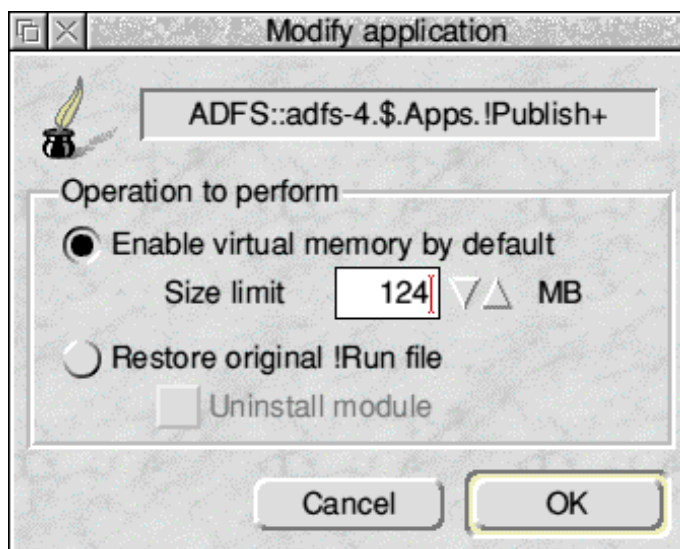
From the menu that appears select the "Virtualise" option to enable virtual memory. If an error box is produced then it may not be possible to enable virtual memory for one or more of the selected areas. If all goes well, a tick will appear in the right-hand column of the table for the selected areas.

That is all that is required to use virtual memory. However, for some applications it is even easier; see the "Products" file for more details.

Modifying an Application to use Virtual Memory by default

To remove the need to enable virtual memory for an application by hand every time it is used it is possible to modify the application to use virtual memory automatically. A backup copy should be made of the application before doing this.

Ensure that the required dynamic areas are selected within the list window, and drag the application to modify from its directory viewer into the list window. This will open the "Modify application" window. Click on "Enable virtual memory by default" and adjust the "Size limit" to the required maximum dynamic area size. Finally click on OK to perform the modify the application. The change will take effect the next time the application is run.



Note that modifying an application also installs the module into the computer's boot sequence so that it is loaded every time the computer is started or reset.

To remove the modification drag the application icon into the list window again, and select the "Restore original !Run file option". This will only take effect after the computer has next been reset. It is also possible to choose whether the module should be removed from the boot sequence by selecting the "Uninstall module option".

Quitting the front end

When it is no longer required the front-end can be quit by selecting the "Quit" option from the icon-bar menu. This does not affect the module which provides the virtual memory, so it is safe to do this even if virtual memory is being used by another application.

Other Features

This tutorial has only skimmed the surface of what the Virtualise system can do. For more details please consult the further chapters in this manual.

The Desktop FrontEnd

Introduction

This chapter is intended as a reference to all of the features included in the "!Virtualis" front-end. For a simple introduction read Chapter 1.

The front-end supports interactive help; use Acorn's !Help application to obtain information about the item under the mouse pointer.

Starting the Program

It is necessary to register the software before using it for the first time. See the "Tutorial" chapter for details of how to do this.

Start the front-end by simply double-clicking SELECT on the "!Virtualis" icon in a directory viewer. After a short delay the application icon will appear on the icon-bar.

The Icon Bar

Clicking MENU on the icon which appears on the iconbar opens the main program menu which offers the following options:

Info

Moving over the sub-menu arrow displays a standard program information window. The version number should be the same as the version of the Virtualise module being used.

List areas...

This opens up a window giving details of all the dynamic areas in the machine, and allow various operations to be performed.

Page faults...

This opens up a window showing the number of page faults which have been serviced classified according to the type of instruction which generated the exception.

Configure...

This opens up a dialogue box which allows the Virtualise module to be configured.

Automatic...

This opens up a window listing the dynamic area names for which virtual memory will automatically be enabled when they are next created.

Tasks...

This opens up a window listing the current WIMP tasks, allowing specified tasks to be suspended and swapped to disc.

Quit

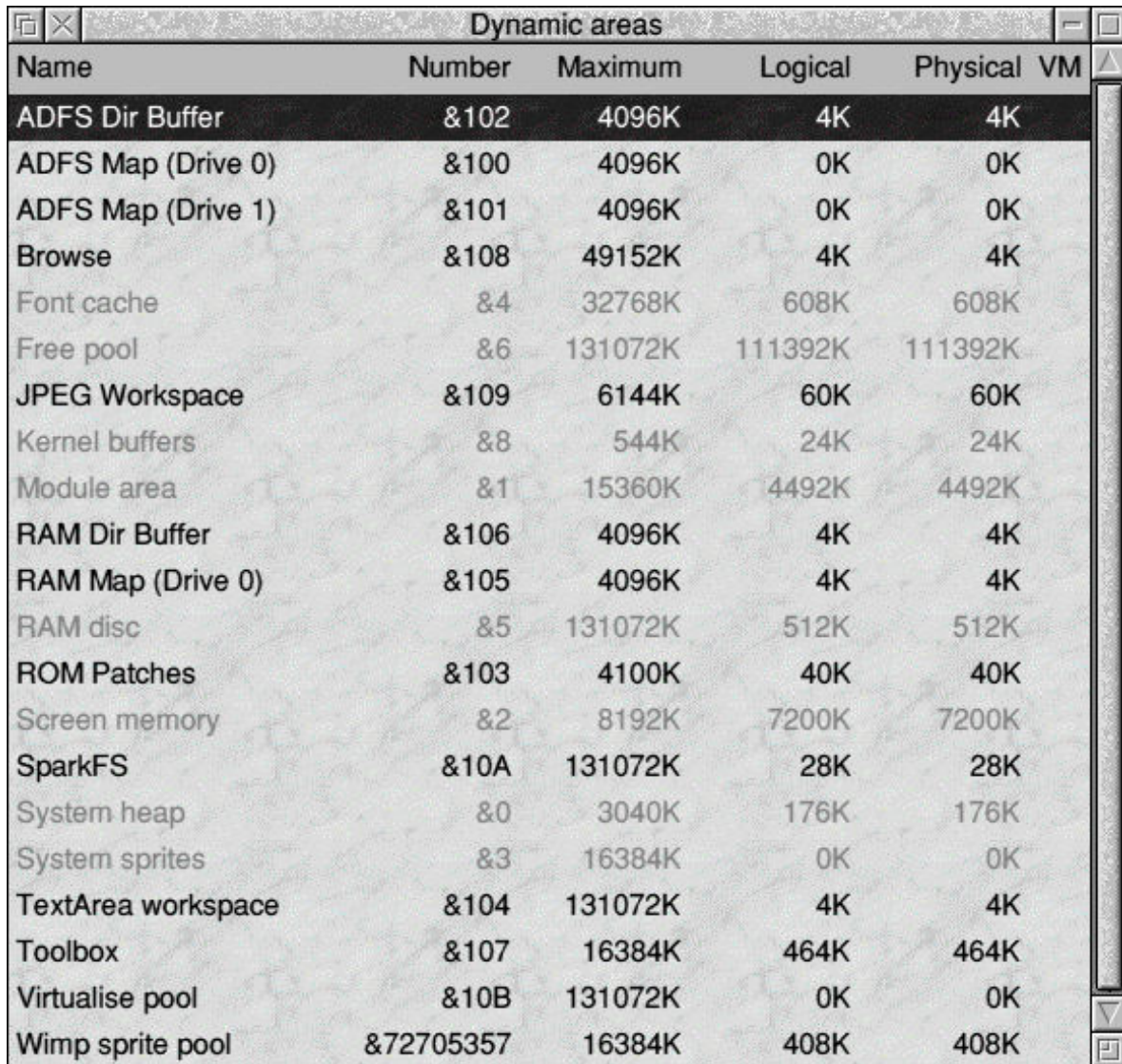
Quits the front-end program. The module is not affected.

These options can be assigned to SELECT and ADJUST clicks on the iconbar icon.



Dynamic Area List Window

The dynamic area list window can be opened by either selecting the "List areas..." option from the icon-bar menu.



Name	Number	Maximum	Logical	Physical	VM
ADFS Dir Buffer	&102	4096K	4K	4K	
ADFS Map (Drive 0)	&100	4096K	0K	0K	
ADFS Map (Drive 1)	&101	4096K	0K	0K	
Browse	&108	49152K	4K	4K	
Font cache	&4	32768K	608K	608K	
Free pool	&6	131072K	111392K	111392K	
JPEG Workspace	&109	6144K	60K	60K	
Kernel buffers	&8	544K	24K	24K	
Module area	&1	15360K	4492K	4492K	
RAM Dir Buffer	&106	4096K	4K	4K	
RAM Map (Drive 0)	&105	4096K	4K	4K	
RAM disc	&5	131072K	512K	512K	
ROM Patches	&103	4100K	40K	40K	
Screen memory	&2	8192K	7200K	7200K	
SparkFS	&10A	131072K	28K	28K	
System heap	&0	3040K	176K	176K	
System sprites	&3	16384K	0K	0K	
TextArea workspace	&104	131072K	4K	4K	
Toolbox	&107	16384K	464K	464K	
Virtualise pool	&10B	131072K	0K	0K	
Wimp sprite pool	&72705357	16384K	408K	408K	

The window which is opened displays details of all the dynamic areas which exist in the computer. This list is updated once per second on WIMP null-polls. Areas with numbers under 256 (those which are not listed as dynamic areas in the Taskmanager) are shaded. The columns of details are:

Name

The name of the dynamic area as displayed by the TaskManager or returned by OS_DynamicArea 2.

Number

The dynamic area number as used with OS_DynamicArea, OS_ReadDynamicArea and OS_ChangeDynamicArea.

Maximum

The maximum size (reserved logical address space) of the dynamic area.

Logical

The current logical size of the area, i.e. how much memory the user of the area thinks is being used.

Physical

The current physical size of the area, i.e. how much real memory is being used to support accesses to area. If the area has not got virtual memory enabled then it will be the same as the logical size.

Virtualise

VM

An indicator of whether virtual memory is enabled for the area.

Unshaded areas may be selected by clicking with SELECT and ADJUST in the usual manner. Clicking Menu opens a menu that allows various operations to be performed. The options are:

Virtualise

Either enable or disable virtual memory for all of the selected dynamic areas. If any of the selected areas have virtual memory already enabled then a tick appears next to this item. This option is shaded if no areas are selected.

Delete area...

Remove the selected dynamic areas. This option is shaded if no areas are selected. A dialogue box is opened to confirm the deletion.

Show map...

This opens a window for each selecting area showing the status of all the pages of memory within the areas.

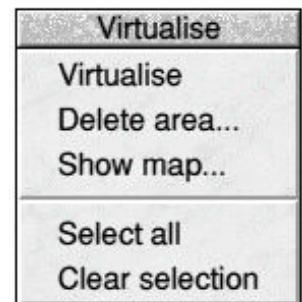
Double-clicking SELECT on a dynamic area entry has the same effect.

Select all

This selects all of the dynamic areas.

Clear selection

This deselects any areas which are selected.



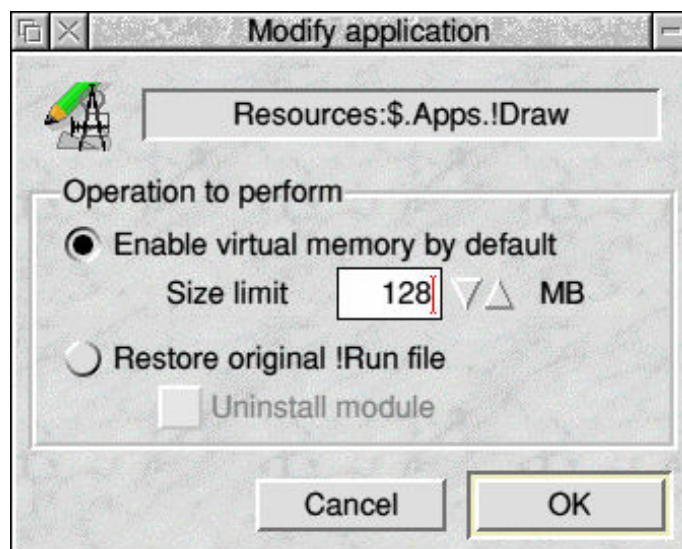
The Virtualise and Remove options should obviously be used with care; it is very easy to crash the computer if these are used carelessly.

If an application is dragged into the list window the "Modify application" dialogue box is opened. This is described in the next section.

Modify Application Dialogue

This allows the application to be modified so that it always uses virtual memory when it is run. A backup of the application should be made before using this option.

The top of the window shows which application will be altered. The centre of the window controls the operation to perform:



Enable virtual memory by default

This option modifies the !Run file of the application to automatically enable virtual memory for the dynamic areas which are selected within the dynamic area list window.

These should be the dynamic areas that are used by the application being modified. This will take effect the next time that the application is started. This option also copies the Virtualise module into the "!Boot.Choices.Boot.PreDesk" directory to ensure that it is always loaded.

Size limit - This is the maximum size that will be set for the selected dynamic areas.

Restore original !Run file

This attempts to restore the application to the state it was in before it was modified. Due to the way in which this operates it may not take effect until the next time the computer is reset.

Uninstall module

Selecting this option removes the Virtualise module from the boot sequence if it was previously installed by the enable virtual memory option.

At the bottom of the window are two action buttons:

Cancel

Clicking on this icon aborts the modification, leaving the application unchanged. Pressing Escape has the same effect.

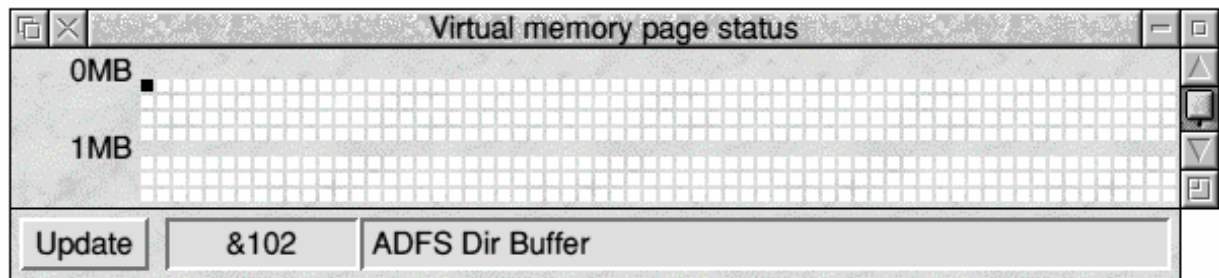
OK

Clicking on this icon performs the selected modification.

It is also possible to press Return to have the same effect.

Page Status Window

Any number of page status windows may be opened. The main part of the window at the top contains a square for each page of memory within the dynamic area.



The colour of the square indicates the state of that page. Clicking MENU within the window opens a menu:

Key...

A window showing the meaning of the different colours.

Zoom

A list of the possible scales at which the status display can be shown.

The lower part of the window has an Update button which takes a new snapshot of the state of the dynamic area. Also shown is the dynamic area number and name.

Note that the access count is only meaningful if the NFU page replacement policy is configured.

Configuration Window

The configuration window is split into two areas. The scrollable area on the left displays all of the options that may be controlled. The area on the right contains three buttons:

Save

Save the values shown. These values will be used by both the Virtualise module and the "!Virtualis" front-end each time they are started in the future.

Read current

Reads the current settings from the Virtualise module.

The numerical values are rounded down to 0.1MB multiples before being displayed.

Set new

Sets the values shown in the window. The settings are then read back and window updated. If the Virtualise module is not loaded the last two buttons will produce an error message.

The section of the scrollable area entitled "Memory usage" shows each of the values which can be set using Virtualise_Configure, and one extra that can be set using Virtualise_UserConfigure. Note that these values can only be specified to the nearest 0.1MB due to the way in which the Toolbox system handles number range gadgets. The items are:

Claim up to

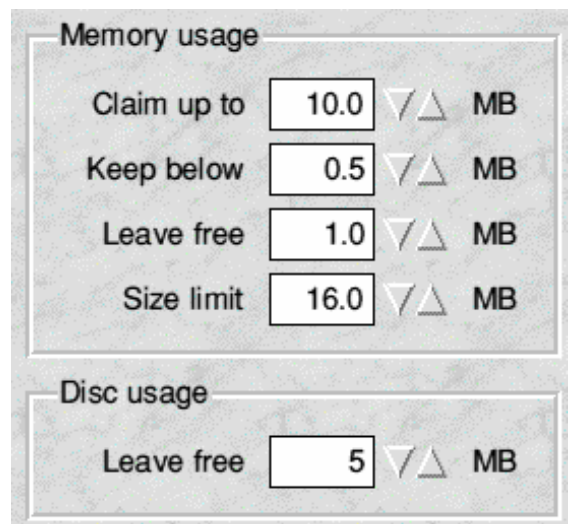
The amount of real memory to claim automatically for the virtual memory pool if it could be useful.

Keep below

The amount of real memory to keep even if other applications require more memory than is available.

Leave free

The minimum amount of memory to leave free when the size of the virtual memory pool is adjusted.



Size limit

The amount of logical address space reserved for dynamic areas if -1 is specified as the maximum size when they are created.

The second section of the scrollable area entitled "Disc usage" controls the usage of disc space for swap files. The current version only has a single item:

Leave free

The amount of disc space to leave free when a swap file is either created or enlarged. This values may sometimes be exceeded due to the way in which disc space is allocated.

Swap file directory

The next section of the scrollable area allows the directory used for storing swap files to be set. This sets the system variable Virtualise\$SwapDir as a macro, so system variables (such as <Wimp\$ScrapDir>) may be included. There are three ways of setting the directory: dragging the directory icon to a directory window, dragging a file or directory icon into the configuration window, or by typing the required pathname into the writable icon. Note that a change to the swap file directory only affects swap files created after the change has been made;

Existing swap files are not moved.

This section of the window also contains an option button. If the option is selected then the swap file directory will be purged each time the Virtualise module is loaded. This will delete any swap files that have been left from previous uses, and can prevent the disc being filled with unused files.

Page replacement policy

Sets the algorithm used to select pages to swap to disc. The options are:

NFU Not frequently used.

This chooses the page that has been accessed the least frequently.

FIFO First-in, first-out.

Pages are swapped out in the order they were originally swapped page.

Random

A page is chosen at random, irrespective of how recently or frequently it has been used.

There are also two option buttons which control whether pages are read and written individually or grouped into larger file operations (which should be faster).

Load module

Controls when the Virtualise module is loaded.

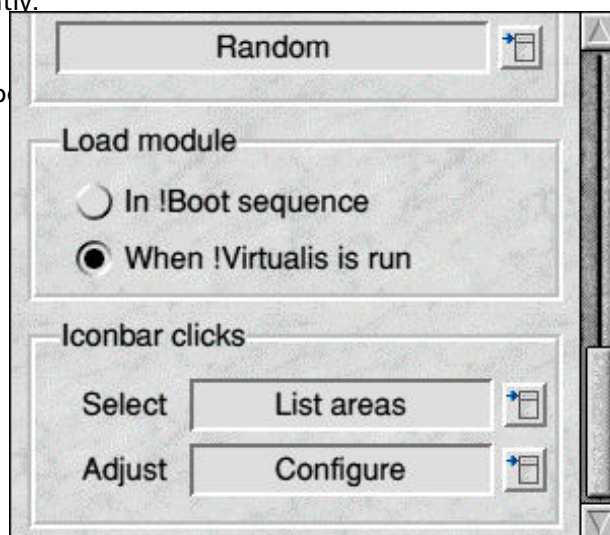
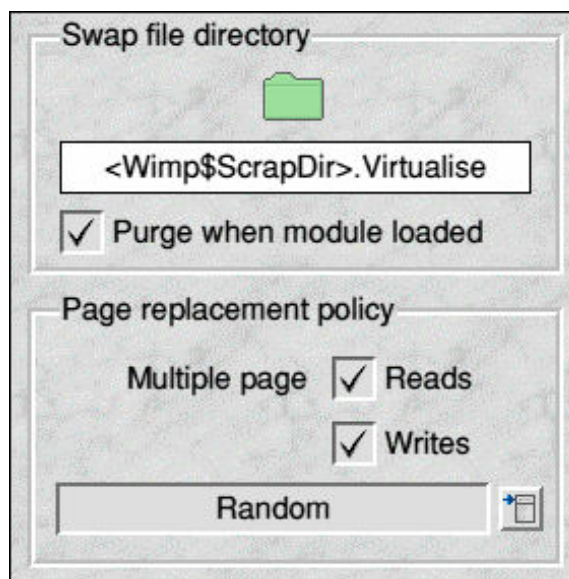
In !Boot sequence

When this option is selected, the Virtualise module will be placed into the "!Boot.Choices.Boot.PreDesk" directory to ensure that it is always loaded.

When !Virtualis is run

When this option is selected, any copy of Virtualise in the "!Boot.Choices.Boot.PreDesk" directory will be deleted. The module will then only be loaded when the !Virtualis front-end is run.

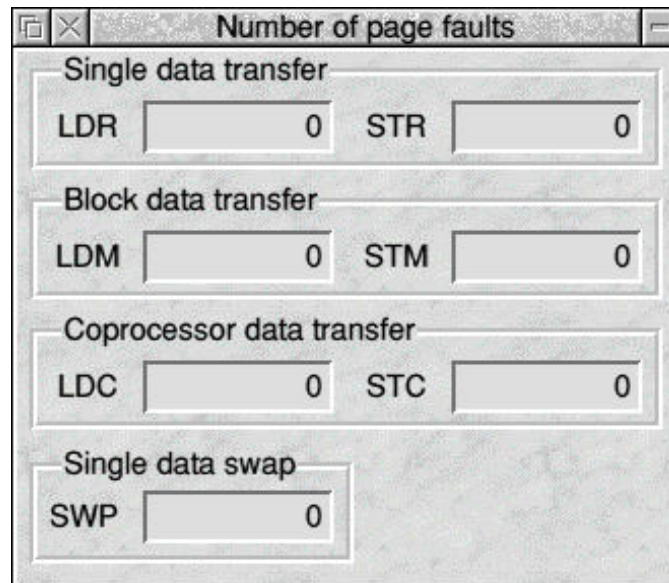
The final section of the scrollable area entitled "Iconbar clicks" allows actions to be assigned to SELECT and ADJUST clicks on the iconbar icon. The options are the same as those available from the iconbar icon menu.



Page Faults Window

The "Number of page faults" window is opened by selecting the "Page faults..." option from the icon-bar menu. The window displays the numbers of virtual memory page faults which were generated by different classes of instruction since the module was loaded.

Note that page faults generated by floating point processor instructions will only be listed under "Coprocessor data transfer" if a real floating point unit is being used. If Acorn's Floating Point Emulator is being used then it will be an instruction within the emulator code which actually generates the exception.



The screenshot shows a window titled "Number of page faults" with a standard Mac OS-style title bar (minimize, maximize, close buttons). The window contains four sections, each with a title and two input fields (spinners) showing the count of page faults:

- Single data transfer**: LDR (0) and STR (0)
- Block data transfer**: LDM (0) and STM (0)
- Coprocessor data transfer**: LDC (0) and STC (0)
- Single data swap**: SWP (0)

Instruction Class	Instruction	Count
Single data transfer	LDR	0
	STR	0
Block data transfer	LDM	0
	STM	0
Coprocessor data transfer	LDC	0
	STC	0
Single data swap	SWP	0

Automatic Virtual Memory Window

The automatic virtual memory list window is opened by selecting the "Automatic..." option from the icon-bar menu.

The window which is opened lists the names of the dynamic areas for which virtual memory will be enabled by default. This operates by setting system variables as described in the "Module" file. This list is updated once per second on WIMP null-polls. The columns of details are:

Name

The name of the dynamic area as displayed by the TaskManager or returned by OS_DynamicArea 2. If a system variable corresponding to an unknown dynamic area then the capitalisation and special characters may not be exactly the same as the actual dynamic area name.

Size limit

The maximum size (reserved logical address space) that should be allocated for the dynamic area if no limit is specified by the program that creates it.

Dynamic area names may be selected by clicking with SELECT and ADJUST in the usual manner. Clicking Menu opens a menu that allows various operations to be performed. The options offered are:

Size limit

The maximum size (reserved logical address space) that should be allocated for the dynamic area if no limit is specified by the program that creates it.

Add area

The sub-menu lists all the names of dynamic areas that have been seen by the "!Virtualis" front-end. Select the name of a dynamic area to add it to the list. It is initially added with a size limit of 256MB. The list of known dynamic areas is saved between sessions.

Remove area

Remove the selected dynamic area names from the list. This also unsets the corresponding system variables.

Save

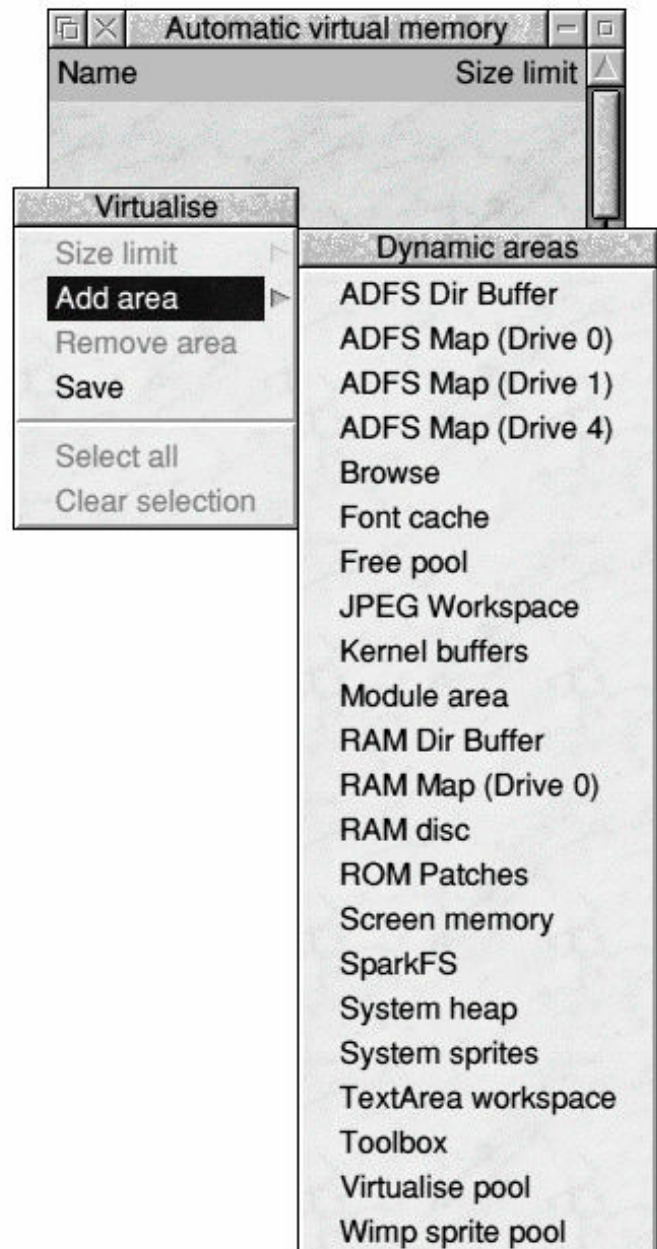
Save a list of the current settings. This option also copies the Virtualise module into the "!Boot.Choices.Boot.PreDesk" directory to ensure that it is always loaded. The file of dynamic areas is read by the Virtualise module each time it is loaded. Hence once the use of virtual memory has been configured, it is not necessary to use the "!Virtualis" front-end.

Select all

This selects all of the dynamic areas.

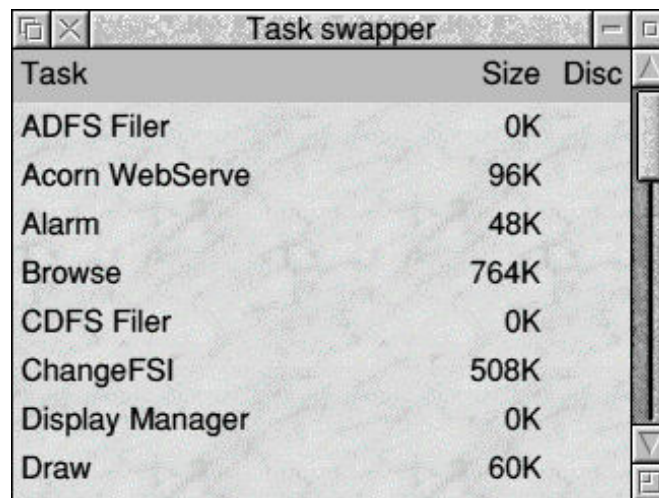
Clear selection

This deselects any areas which are selected.



Task Swapper Window

The task swapper window is opened by selecting the "Tasks..." option from the icon-bar menu. Note that **Not all tasks can be safely swapped out**. If in doubt, save any important data (in all applications) before trying to suspend the task. Programs with filters applied are likely to cause problems.



The screenshot shows a window titled "Task swapper" with a table of tasks. The table has three columns: "Task", "Size", and "Disc". The "Disc" column is currently empty. The tasks listed are: ADFS Filer (0K), Acorn WebServe (96K), Alarm (48K), Browse (764K), CDFS Filer (0K), ChangeFSI (508K), Display Manager (0K), and Draw (60K). A vertical scrollbar is visible on the right side of the table.

Task	Size	Disc
ADFS Filer	0K	
Acorn WebServe	96K	
Alarm	48K	
Browse	764K	
CDFS Filer	0K	
ChangeFSI	508K	
Display Manager	0K	
Draw	60K	

The window which is opened lists the names of the WIMP tasks running within the computer. The columns of details are:

Task

The name of the task, as displayed in the TaskManager window.

Size

The size of the task, either in memory or on disc. For active tasks this is the same as the figure displayed by the TaskManager.

Disc

An indicator of whether the task has been swapped to disc.

Clicking on a task within the window attempts to suspend the task, and swap it to disc. Clicking on a suspended task attempts to reload it and resume execution. If ADJUST is used, instead of SELECT, then the window is also closed. When a task is suspended, any open windows belonging to the task are closed, and any icons on the icon-bar are replaced. The task may be resumed by clicking on any of the replacement icon-bar icons.

If it is not possible to restore a task, for example if there is insufficient memory, then an error is displayed. This offers the following options:

Cancel

Close the error window, and leave the task suspended.

Kill Task

Stop the task immediately. This does not exit the task normally, so should only be used as a last resort. If an attempt is made to exit the desktop while a task is swapped out, an error dialogue is displayed, and the following options are offered:

Restore

Attempt to swap the task back. The shutdown is aborted.

Abort

Abandon the shutdown, and leave the task suspended.

Shutdown

Kill the task, and resume the shutdown. The task is not exited normally, so this should only be used as a last resort.

The software SWIs

Introduction

The Virtualise module can be controlled directly via a number of SWIs (SoftWare Interrupts). These are described in detail below. However, they are really only required by programs which offer control over how virtual memory is used; just using the module to provide virtual memory for other author's software does not require use or knowledge of these SWIs.

The interface is logically split into two parts: existing RISC OS calls whose behaviour has been modified, and new SWIs provided by the module.

Changes to OS SWIs

SWI "OS_DynamicArea" (&66)

This SWI has been extended to allow dynamic areas larger than physical RAM to be created, and to return the logical size of the area rather than the physical size.

An extra area flag has been defined:

Bit 31 - Virtual memory is active for dynamic area. If this bit is set when the area is created then virtual memory is activated immediately. Note that no error is produced if it is not possible to start virtual memory for the new dynamic area.

Note that reading this flag bit only gives meaningful results if the Virtualise module is loaded at the time. This is because the OS is perfectly happy to maintain and return this bit itself.

The area size may be specified to be larger than the amount of RAM in the computer, up to a limit of 256MB. However, if -1 is specified for the maximum size of the area then the maximum size is set to the limit set using Virtualise_Configure (by default the amount of physical RAM (as at present)). The maximum may be specified larger than the RAM size even if virtual memory is not being started initially. If Virtualise_Start is later used then the virtual memory will be able to utilise all the of the allocated address space.

One important point to note is that there is only 3.5GB of address space available in the machine. This can soon get used up if all applications ask for the maximum possible of 256MB. This problem is made worse by a bug in Filecore which fails with top bit set addresses (over 2GB). Other applications, such as !ChangeFSI, sometimes fall over at even lower addresses.

SWI "OS_ChangeDynamicArea" (&2A)

If used on an area that has virtual memory active this SWI adjusts the logical size of the area, otherwise the physical size is adjusted. In practice this means that this call can be used by the application without worrying whether virtual memory is active.

SWI "OS_ReadDynamicArea" (&5C)

If used on an area that has virtual memory active this SWI reads the logical size of the area, otherwise the behaviour is the same as before. In practice this means that this call can be used by the application without worrying whether virtual memory is active.

SWIs provided by Virtualise

The SWI chunk starting at &4B6C0 has been officially allocated by Acorn for use by this module.

SWI "Virtualise_Configure" (&4B6C0)

Configure logical and physical memory usage.

On entry:

- **R0**
Set the amount of logical address space, in bytes, allocated for dynamic areas when -1 is passed to OS_DynamicArea 0, or -1 to read current value.
- **R1**
Amount of memory to claim automatically, in bytes, or -1 to read current value.
- **R2**
Amount of memory to leave free, in bytes, or -1 to read current value.

On exit:

- **R0**
The size of address space to allocate.
- **R1**
The amount of memory to claim, in bytes.
- **R2**
The amount of memory to leave free, in bytes.

Interrupts: Interrupt status is undefined. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

This call provides control over how much real memory to claim, and how much to leave free. These values are only treated as guidelines. In particular the user may drag the bar in the Task Manager or the specified values may lie outside the range of required or useful memory.

Do not rely on any particular default values. They may be altered by another application, the configuration specified by the user, or they could change in a future release of this module.

The recommended way of using this call (if it is used at all) is to include an option in the application configuration window for minimum amount of memory to claim and minimum memory to leave free. When the window is opened this call should be used to read the current settings. When the user selects OK or Save this call should be used to set the values. If the application saves a configuration then it should call this SWI when it starts to set the required values. The application should not normally set these values any more frequently- it is likely to result in confusing behaviour for the user if more than one application is using the virtual memory system.

It is also possible to control how much address space is allocated for dynamic area which are created with -1 passed for the maximum size. This allows badly written programs to be used on machines with a large memory without using ridiculously large sections of the logical address space. This option should not normally be used by applications themselves - it is intended for use by the user to control wastage of logical address space by programs which are not aware of virtual memory.

Under RISC OS 4 and above it is recommended that OS_DynamicArea 8 is used to clamp the size of dynamic areas.

SWI "Virtualise_Start" (&4B6C1)

Start virtual memory in an existing dynamic area.

On entry:

- **R0**
The number of the dynamic area to virtualise.
- **R1**
Maximum size of area (reserved logical address space), or -1 to use maximum size of the dynamic area.
- **R2**
Pointer to name to use for swap file. If this is a null pointer then a name is chosen automatically.

On exit:

All registers preserved.

Interrupts: Interrupts are enabled. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

After a dynamic area has been created using `OS_DynamicArea 0` (as usual) use this SWI to enable virtual memory.

The initial size of the virtual memory is the same as the actual size of the dynamic area; any data already in the area is copied into the virtual memory. If there is no data in the area when virtual memory is enabled then it would be more efficient to shrink the area to 0 size before using this SWI, and to grow the virtual memory with `OS_ChangeDynamicArea` afterwards.

Normally virtual memory is restricted to being within the logical address space of the dynamic area. However, if more contiguous logical address space has been reserved it is possible to set R1 to this value to allow its use. It should never be necessary to use this option since the Virtualise module is capable of creating dynamic areas with maximum sizes greater than the RAM size of the computer.

The swap file will normally be placed in the automatically created directory . This may be changed by specifying a full pathname in R2. The variable defaults to Virtualise.

Note that the following flags must be set:

- **bit 4 0 - Area is bufferable**
- **bit 5 0 - Area is cacheable**

If the dynamic area has a handler routine then virtual memory can only be enabled for it if this module was loaded when the area was created.

SWI "Virtualise_End" (&4B6C2)

Disable virtual memory in a dynamic area.

On entry:

- R0
The number of the dynamic area to un-virtualise.

On exit:

All registers preserved.

Interrupts: Interrupts are enabled. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

Use this SWI to disable virtual memory for a dynamic area. If virtual memory is not active for the specified dynamic area then the call returns without error.

If any virtual memory is associated with the area then an attempt is made to convert it into physical memory. If it is not possible to claim sufficient memory then the SWI will return an error, and the virtual memory will remain active.

If the area contains pages locked using Virtualise_Lock then an error will be produced, and the area will remain with virtual memory active.

If the dynamic area is removed without virtual memory being first disabled then this SWI is implicitly called, but no check is made for locked pages. It is better to call this SWI explicitly to allow more elaborate checks to be performed.

SWI "Virtualise_Lock" (&4B6C3)

Lock pages of a dynamic area for which virtual memory has been enabled.

On entry:

- **R0**
Start of range of addresses to lock.
- **R1**
End of range of addresses to lock.

On exit:

All registers preserved.

Interrupts: Interrupt status is undefined. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

This SWI allows an area of virtual memory to be locked in physical memory. This enables the use of that area for interrupt routines, and other cases which could otherwise lead to reentrancy of either the virtual memory manager or the file system.

All pages that are included in the range R0 to (R1-1) are locked. This range must lie fully within the current size of a dynamic area for which virtual memory has been enabled. A count of how many times a page has been locked is maintained, and a page will only be unlocked when the count reaches zero.

Care should be taken when using this SWI to ensure that the minimum possible number of pages are locked at any time, otherwise there may not be sufficient physical memory available to provide the rest of the virtual memory. It is advised that pages are locked for the shortest possible period.

It is possible to use this call to improve performance of the virtual memory system by forcing particular pages to remain in memory. However, this approach should only be used in rare circumstances where the performance increase would be substantial. If possible, locking pages in this manner should be a user configurable option.

Virtualise

SWI "Virtualise_Unlock" (&4B6C4)

Unlock pages of a dynamic area for which virtual memory has been enabled.

On entry:

- **R0**
Start of range of addresses to unlock.
- **R1**
End of range of addresses to unlock.

On exit:

All registers preserved.

Interrupts: Interrupt status is undefined. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

This SWI unlocks pages that were previously locked with `virtualise_Lock`. Exactly the same values of R0 and R1 should be used when unlocking pages as were used when they were originally locked; unpredictable results might otherwise occur.

SWI "Virtualise_MiscOp" (&4B6C5)

This is for the internal use of the "Virtualis" front-end, and should not be used.

SWI "Virtualise_UserConfigure" (&4B6C6)

Configure the memory and disc usage.

On entry:

- R0
Set the fraction of the logical virtual memory size to claim as physical memory, or -1 to read the current value. This is a fixed point value, with (1 R1
Lower limit of memory to claim, below which all of the required memory is claimed, or -1 to read the current value.
- R2
Upper limit of memory to claim, above which no more memory will be claimed, or -1 to read the current value.
- R3
Amount of memory to keep even when another task is attempting to claim memory, or -1 to read the current value.
- R4
Amount of disc space to leave free, or -1 to read the current value.

On exit:

- R0
Fraction of required memory to claim.
- R1
Minimum memory to claim, in bytes.
- R2
Maximum memory to claim, in bytes.
- R3
Memory to keep, in bytes.
- R4
Disc space to leave free, in bytes.

Interrupts:

Interrupt status is undefined. Fast interrupts are enabled.

Processor mode: Processor is in SVC mode.

Re-entrancy: SWI is not re-entrant.

Use:

This call provides extra control over how much real memory and disc space to claim. It is provided as a supplement to Virtualise_Configure to provide extra configuration options while ensuring backwards compatibility. Some of the values are adjusted by the user when the bar in the Task Manager is dragged. These values are only guidelines. More or less memory may be claimed depending upon other constraints.

Do not rely on any particular default values. They may be altered by another application, by the user dragging the bar in the TaskManager, or they could change in a future release of this module.

The amount of memory claimed is (approximately) chosen as follows:

The amount of physical memory that could usefully be used to cache virtual memory is calculated.

This value is scaled by the specified fraction to obtain the initial allocation.

If the value is above the specified upper limit, then the upper limit is used instead.

If the value is below the specified lower limit, then the lower limit is used instead.

If the value is below the minimum amount to claim, then the required minimum amount to claim is used instead.

If the value does not leave the specified amount of memory free, then the claimed amount is reduced as required.

If the value is below the minimum memory to keep, then the amount is increased to the minimum to keep.

Other constraints, such as the number of locked pages and the maximum memory that can be used, are then enforced.

The amount of memory to keep is also used to decide whether to reduce the memory claimed when another task attempts to allocate memory.

The disc space to leave free value allows disc space to be reserved for other purposes, and to avoid problems with FileCore that can result in corruption of the disc.

Be careful using this call. Inappropriate use could make virtual memory use very inefficient.

Appendices

Glossary of Terms

Binary

A two-state system which can be represented by the digits 0 and 1. Digital computers perform information processing, store data and transmit information as binary digits.

Bit

Abbreviation for Binary digit. It is a unit of storage which can take the values 0 or 1.

Byte

A group of 8 bits. This is generally the amount of memory required to store a single character.

Data

Synonymous with information but generally used for the representation and processing of information by a computer system.

Dynamic area

A dynamic area is a named area of memory created by an application. The Tasks window obtained by clicking SELECT on the Acorn icon on the icon-bar lists the dynamic areas at the bottom of the window.

Hexadecimal

Numbers in base 16. The digits are 0123456789ABCDEF, where "A" to "F" represent the values 10 to 15.

Interrupt

When the control unit stops processing to handle some external need, such as performing an input/output task.

Logical address

This is the address that the programs running in the computer use to access memory. The hardware in the computer converts the logical address into a physical address when required. When virtual memory is being used the mapping between logical and physical memory is frequently updated as different areas of memory are accessed.

kB

Abbreviation for kilobyte. This is equivalent to $1024 (2^{10})$ bytes.

MB

Abbreviation for megabyte. This is equivalent to 1,048,576 ($1024 * 1024 = 2^{20}$) bytes. This is sometimes used more approximately to refer to 1 million bytes.

Operating system

Essential systems software which coordinates and controls the use of the system's resources. In the Risc PC this is called RISC OS.

Page fault

An access to a page of memory which is not present at that time results in an abort on data transfer exception being raised. This is termed a page fault.

Physical address

This is the address that is actually used to access a particular section of the memory in the hardware of the computer. A physical address always references the same part of the same memory chip.

RAM

Random Access Memory. This is the main memory on a chip which allows data to be stored and modified; read/write memory is a more apt description.

Virtual memory

This is a system which enables disc space to be used to simulate extra RAM. It is usually provided as a feature of the operating system.

VM

Abbreviation for Virtual Memory.

WIMP

Abbreviation for Windows Icons Menus Pointer. This is a generic term for the graphical interface to the computer. On the RiscPC it is sometimes called the Desktop.

Version History

1.00 (21-Oct-95)

First official release version.

1.01 (03-Dec-95)

Fixed error handling when out of disc space. Handling of errors during creation of area handled better. No longer locks pages used by OS_FSControl 4. Front-end handles unexpected errors better. Redraw of page status window speeded up. Page status display is scalable.

1.02 (08-Jan-96)

Robust handling of flags on entry to file system intercepts.

1.03 (31-Jan-96)

Modified !Run files pass on original parameters.

1.04 (07-Mar-96)

Configuration file moved to .Virtualise. Added editor for automatic virtual memory system variables. Module reads configuration without front-end being run.

1.05 (10-Apr-96)

Added ability to swap complete tasks to disc. Changed Yes/No indicators to use ticks.

1.06 (14-May-96)

Fixed handling of dismounts.

1.10 (08-Aug-96)

Made StrongARM compatible. A dialogue box is used to check deleting a dynamic area. Added Virtual check command to verify module consistency. Improved handling of OS_Heap operations in virtual memory. New look title window.

1.11 (12-Sep-96)

Added Virtualise_UserConfigure SWI. New look configuration window. Enhanced memory claim options, and new defaults. Action taken on iconbar icon clicks now configurable. Option added to purge the swap file directory when loaded.

1.12 (25-Jan-97)

Prevent !Virtualis front-end being swapped out. System variable size limit only used if none from SWI. Configuration file correctly terminated with a newline. Corrected automatic processor recognition.

1.13 (01-Apr-97)

Kludge around bug in revisions less than 3 of SA110.

1.14 (15-Oct-97)

The "keep below" value is used when adjusting the pool size. Free disk space option is ignored when used with networks.

1.15 (27-May-99)

Fixed the free disk space check with large (>2GB) drives. Cured spurious crashes when virtual memory pool shrunk. Fixed rare corruption of the relocatable module area. Corrected order of loading of Toolbox modules. More efficient service call handling under Ursula. Corrected creation of non-empty areas with handlers. Fixed incorrect stack handling in configuration SWI veneer.

1.16 (23-Aug-99)

Added support for RISC OS 4. Corrected behaviour if no configuration file found. Corrected validation of address ranges that wrap.

1.17 (14-Dec-99)

Processor flags preserved across filesystem SWI calls.

Acknowledgments

The author would like to thank the following people and organisations for their contributions to the development of this application:

Acorn Computers - The RiscPC, RISC OS and the ObjAsm assembler.

Tim Armes - Hints about how Straylight approached the problems.

Richard Barlow - Patience while an obscure bug was fixed.

Dave Clare - The idea and incentive behind this module.

Aidan Corey - Volunteering to test a potentially dangerous program.

Jonathan Coxhead - For maintaining OSLib even though not at Acorn now.

Robert Davison - The stunning Compo, useful bug reports and comments.

David Jackson - Many bug reports and more ideas for the interface.

Dominic Symes - The brilliant !Zap editor; get a copy immediately.

Robin Watts - !Larger, the greatest backdrop program ever.

Here are just a few of the other products you will be able to
download from RISCOS.com free of charge
(You can make a donation to help support future product releases but this is entirely voluntary)



This Documentation is © 3QD Developments Ltd 2026
Some of the products shown above might not be available immediately.
Not all products will run on all versions of RISC OS, check the product information page for details.

Other Products list last updated 29/2/26