MuMove4K

Thomas Richter

COLLABORATORS			
	TITLE : MuMove4K		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas Richter	January 13, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MuMove4K

1.1 MuMove4K Guide

´### ,#. ,#### ,### ####’ #### ,#### ,###’ ####’ #### _____ ,#### ,###’ || ####’ #### | ___ ___ | ___ ,#### ,###’ ---- | | | | | | |___ ####’ #### | | | | | | | ,#####. ,###’ . ||___||___||_ ___| ####’##. ,#### ,## ,#### ########’ # ,##’ ####’ `####’ `###’ ,#### ##### © 1999-2001 THOR - Software, ####’ Thomas Richter `##’

MuMove4K Guide

Guide Version 1.07 MuMove4K Version 40.22

The Licence : Legal restrictions

MuTools : What is this all about, and what’s the MMU library?

What is it : Overview

Installation : How to install MuMove4K

Synopsis : The command line options and tool types

History : What happened before

© THOR-Software

Thomas Richter

Rühmkorffstraße 10A

12209 Berlin

Germany

EMail: thor@math.tu-berlin.de

1.2 The THOR-Software Licence

The THOR-Software Licence (v2, 24th June 1998)

This License applies to the computer programs known as "MuMove4K" and the "MuMove4K.guide". The "Program", below, refers to such program. The "Archive" refers to the package of distribution, as prepared by the author of the Program, Thomas Richter. Each licensee is addressed as "you".

The Program and the data in the archive are freely distributable under the restrictions stated below, but are also Copyright (c) Thomas Richter.

Distribution of the Program, the Archive and the data in the Archive by a commercial organization without written permission from the author to any third party is prohibited if any payment is made in connection with such distribution, whether directly (as in payment for a copy of the Program) or indirectly (as in payment for some service related to the Program, or payment for some product or service that includes a copy of the Program "without charge"; these are only examples, and not an exhaustive enumeration of prohibited activities).

However, the following methods of distribution involving payment shall not in and of themselves be a violation of this restriction:

(i) Posting the Program on a public access information storage and retrieval service for which a fee is received for retrieving information (such as an on-line service), provided that the fee is not content-dependent (i.e., the fee would be the same for retrieving the same volume of information consisting of random data).

(ii) Distributing the Program on a CD-ROM, provided that

a) the Archive is reproduced entirely and verbatim on such CD-ROM, including especially this licence agreement;

b) the CD-ROM is made available to the public for a nominal fee only,

c) a copy of the CD is made available to the author for free except for shipment costs, and

d) provided further that all information on such CD-ROM is re-distributable for non-commercial purposes without charge.

Redistribution of a modified version of the Archive, the Program or the contents of the Archive is prohibited in any way, by any organization, regardless whether commercial or non-commercial. Everything must be kept together, in original and unmodified form.

Limitations.

THE PROGRAM IS PROVIDED TO YOU "AS IS", WITHOUT WARRANTY. THERE IS NO WARRANTY FOR THE PROGRAM, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IF YOU DO NOT ACCEPT THIS LICENCE, YOU MUST DELETE THE PROGRAM, THE ARCHIVE AND ALL DATA OF THIS ARCHIVE FROM YOUR STORAGE SYSTEM. YOU ACCEPT THIS LICENCE BY USING OR REDISTRIBUTING THE PROGRAM.

Thomas Richter

1.3 What's the MMU.library?

All "modern" Amiga computers come with a special hardware component called the "MMU" for short, "Memory Management Unit". The MMU is a very powerful piece of hardware that can be seen as a translator between the CPU that carries out the actual calculation, and the surrounding hardware: Memory and IO devices. Each external access of the CPU is filtered by the MMU, checked whether the memory region is available, write protected, can be hold in the CPU internal cache and more. The MMU can be told to translate the addresses as seen from the CPU to different addresses, hence it can be used to "re-map" parts of the memory without actually touching the memory itself.

A series of programs is available that make use of the MMU: First of all, it's needed by the operating system to tell the CPU not to hold "chip memory", used by the Amiga custom chips, in its cache; second, several tools re-map the Kickstart ROM to faster 32Bit RAM by using the MMU to translate the ROM addresses - as seen from the CPU - to the RAM addresses where the image of the ROM is kept. Third, a number of debugging tools make use of it to detect accesses to physically unavailable memory regions, and hence to find bugs in programs; amongst them is the "Enforcer" by Michael Sinz. Fourth, the MMU can be used to create the illusion of "almost infinite memory", with so-called "virtual memory systems". Last but not least, a number of miscellaneous applications have been found for the MMU as well, for example for display drivers of emulators.

Unfortunately, the Amiga Os does not provide ANY interface to the MMU, everything boils down to hardware hacking and every program hacks the MMU table as it wishes. Needless to say this prevents program A from working nicely together with program B, Enforcer with FastROM or VMM, and other combinations have been impossible up to now.

THIS HAS TO CHANGE! There has to be a documented interface to the MMU that makes accesses transparent, easy and compatible. This is the goal of the "mmu.library". In one word, COMPATIBILITY.

Unfortunately, old programs won't use this library automatically, so things have to be rewritten. The "MuTools" are a collection of programs that take over the job of older applications that hit the hardware directly. The result are programs that operate hardware independent, without any CPU or MMU specific parts, no matter what kind of MMU is available, and programs that nicely co-exist with each other.

I hope other program authors choose to make use of the library in the future and provide powerful tools without the compatibility headache. The MuTools are just a tiny start, more has to follow.

1.4 What's the job of MuMove4K?

MuMove4K is a memory layout preparation tool to be used with MuFastZero, and with the "ShapeShifter" Macintosh® emulator. MuMove4K is special amongst the MuTools in the sense that it doesn't require the mmu.library to run.

MuMove4K is required in the startup sequence in case you want to re-map two important system libraries from slow chip memory to faster memory, namely the "exec.library" providing the core of the Os and and "expansion.library". The actual re-mapping process is performed by MuFastZero and its "FastExec" option, and you should consult the guide of this program to see how this is done. However, MuMove4K must be run first to ensure no graphics data is "in the way" and to allow re-mapping the libraries without trying to re-map parts of the graphics engine.

Additionally, MuMove4K may perform the same job as the "PrepareEmul" program of the ShapeShifter distribution and move these system libraries even out of the second 4K block, reserving the low memory area for the MacOs. MuMove4K is not compatible to PrepareEmul, but it replaces it in a compatible way, hence allows you to run MuFastZero and the ShapeShifter at the same time. This allows you to speed up memory access to the crucial low memory vectors of the MacOs, too.

Warning: MuMove4K will reboot your system the first time you call it. This is desired! This means, for you, that you should finish all disk activities before calling MuMove4K or you might invalidate your HD. MuMove4K tries to be as friendly as possible, but might fail to keep the HD safe. The use of a tool like "DiskSafe" is recommended in case you start the program by hand instead running it early in the startup-sequence.

More technical details

1.5 Technical Details

What is precisely the job of MuMove4K and MuFastZero?

Some TurboBoards come with non-auto-configurable memory, i.e. memory that is not located in the Zorro-II, or Zorro-III configuration area. Since this kind of memory is not located at the time the expansion.library is build, the exec and expansion library will end up in Chip mem. To be precise, the following happens on a system startup:

o) The startup code looks for "native" Amiga memory. This is chip mem for first, and the so-called "ranger memory" of the early A2000's in the memory region 0x00c00000 and above, as well as the native motherboard memory of the A4000 and A3000. The system library, exec.library, is then build in this memory as a first step, and this native memory is added to the memory free list of exec.

o) Exec initialized the expansion library, used to mount the hardware expansions. Since only native memory is available at that time, expansion will end up in "slow" memory as well.

o) The expansion library will scan the hardware for auto-configuring boards. Every board found is added to the expansion library ConfigDev's, and memory boards are added to the system memory list.

o) In a fourth step, exec checks if now fast memory is available. If it is, it removes its copy from the slow memory and rebuilds itself in fast memory. Hence, in systems with auto-configuring memory available, the exec library is actually *build twice*.

o) The exec library runs the diag.init. This is the place where additional expansions might "hook in", as for example non-auto-configuring memory. However, since execbase is already constructed at this stage of the startup process, it will be no longer relocated to faster memory if that is made available now.

Hence, on systems without auto-configuring memory, execbase will end up in slow memory, in worst case in ChipMem.

MuMove4K hooks now in at reset time, right below the diag init. It scans the library and device list for entries that reside in chip memory, and builds a private memory pool keeping these memory areas. This is to avoid the graphics library allocating

more memory from that area and using it for the Amiga custom chips. However, this memory still remains marked as ChipMem, so that the 68040 or 68060.library marks this area correctly as "non-cacheable". Setting this domain to cacheable won't cause conflicts with DMA operations, though, because of the MuMove4K program, but the CPU would try burst accesses into that memory domain the chip memory can't handle correctly.

If the PREPAREEMUL option is given, the program installs another reset resident segment which relocates the chip memory start address to a higher address to make room for the Macintosh ROM globals. The implementation is clearly a "hack", as this is not supported by the AmigaOs.

If MuFastZero is loaded, it detects the private memory pool build by MuMove4K and relocates this memory to faster memory, using the MMU. Since no graphics buffers have been allocated there, this can be done safely without any risk. The memory attributes get now "private, non-chip", to avoid allocations from this pool.

1.6 Installation of MuMove4K

Installation is pretty simple:

- First, install the "mmu.library": Copy this library to your LIBS: drawer if you haven't installed it yet. It's contained in this archive. Even though MuMove4K doesn't require this library, it is most useful with it available.

- Copy "MuMove4K" wherever you want. The canonical place is the C: directory.

- Remove all other zero page remappers from your startup-sequence and add the following line IN FRONT OF THE SETPATCH COMMAND:

MuMove4K

to get the graphics library used out of the graphics memory. In case you plan to use the "ShapeShifter" as well, use the following line instead:

MuMove4K PREPAREEMUL

and remove the "PrepareEmul" program from the script. It is no longer required, MuMove4K takes over.

In case the above line does not work, try the following instead:

MuMove4K PREPAREEMUL A1200

This uses an alternative way of implementing the "hack".

Be warned! The "PREPAREEMUL" option is really a hack. This doesn't mean its worse than the original "PREPAREEMUL" - this is a hack as well. It *might* fail on certain boards, please let me know in case this happens on your machine.

More technical details

In case it still doesn't work - there are more compatibility options you might want to try.

That's all.

1.7 Command line options and tooltypes

MuFastZero can be started either from the workbench or from the shell. In the first case, it reads its arguments from the "tooltypes" of its icon; you may alter these settings by selecting the "MuMove4k" icon and choosing "Information..." from the workbench "Icon" menu. In the second case, the arguments are taken from the command line. No matter how the program is run, the arguments are identically. Without any option, it will install itself, without the "PrepareEmul" function.

Warning: MuMove4K will reboot your system the first time you call it. This is desired! This means, for you, that you should finish all disk activities before calling MuMove4K or you might invalidate your HD. MuMove4K tries to be as friendly as possible, but might fail to keep the HD safe. The use of a tool like "DiskSafe" is recommended in case you start the program by hand instead running it early in the startup-sequence.

MuMove4K OFF/S,PREPAREEMUL/S,A1200/S,NOREBOOT/S,OFF/S, PREPAREEMUL/S,A1200/S,NOREBOOT/S,IGNOREVER
INCHIPMEM/S:

OFF

If given, MuMove4K removes itself from the memory and reboots the system. This option replaces the "NoMuMove4K" program.

PREPAREEMUL

Another switch. If this switch is present, MuMove4K will push the beginning of the chip memory upwards to the 8K boundary in order to make room for the MacOs globals. It will reboot your system for the first time you invoke it like this.

A1200

This implements a different mechanism for the chip memory move. In case the "PREPAREEMUL" option does not work as expected, add this option and try again. It will implement a different "hack" which might work on boards the first mechanism fails on.

NOREBOOT

Tells MuMove4K not to reboot the computer, even if it found the patches not installed. This is useful if a second program run after MuMove4K is going to reboot the system anyhow, for example the V44 SetPatch. Note that the MuMove4K function will not be available unless the computer gets rebooted after installation, hence don't use this option if you DO NOT run a rebooting program afterwards.

IGNOREVERIFY

MuMove4K comes with a build-in security mechanism that skips the reboot in case it finds any connected devices in "verify" state. This happens for example if you rebooted the computer while it was accessing the harddisk. This security mechanism ensures that MuMove4K doesn't damage the system even more by rebooting while verification is active.

Some devices might, however, not come up fast enough and might therefore prevent MuMove4K from rebooting altogether. If this is a problem, specify IGNOREVERIFY on the command line and MuMove4K will reboot the system regardless of devices verifying or not.

REVERSE

Makes MuMove4K allocating the memory for the "reset resident" part in reverse order "top down" from the end of the memory instead of taking it from the beginning of the memory. It might be that it is simply "in the way" for other reset-resident programs - if this is the case, REVERSE will propably move it out of the way.

LOWPRI

Specifies a different - smaller - priority for the "reset resident" part of MuMove4K, directly in front of the graphics.library. This may move MuMove4K out of the way for some other reset-resident parts of the system.

This option does not affect the "PREPAREEMUL" function, but only the ordinary "Move4K" feature.

INCHIPMEM

Enforces that MuMove4K places its reset resident code in chip memory rather than in - what it thinks - optimal memory type. This might be required in case some of your expansion board memory is erraneously specified as "reset-resident" even though it isn't, in fact. Various hacks exist that modify the memory properties inaccurately for moving the RAD ram disk out of chip memory, but will then break MuMove4K as a side effect. INCHIPMEM can be used to work around these tools.

This option does not change any memory type for remapping, ExecBase placement or has any impact on system performance.

1.8 History

Release 40.1:

This is the first official release.

Release 40.2:

MuMove4K moves now actually 32K instead of 4K. This ensures first that the low memory remains aligned even in worst case, for a 68030 or 68851 with 32K pages, and that a possible future Oxypatcher utility might make use of the additional 32K of low memory, for example to use short absolute sub- routine calls.

Release 40.10:

Added PREPAREEMUL and OFF option, making NoMuMove4K obsolete. MuMove4K greatly enhanced, replaces now PREPAREEMUL as well.

Release 40.13:

Fixed the shutdown code which seems to fail for certain devices. Added the A1200 option. Made some minor modifications to the original hack.

Release 40.14:

The "PREPAREEMUL" option disabled the CPU instruction cache, or more precisely, the ROM did not re-enable it as expected. Fixed.

Release 40.15:

Added the "NOREBOOT" option which is useful if another tool reboots the computer anyhow, to avoid unnecessary reboots.

Release 40.17:

Added much more error messages in case something goes wrong. MuMove4K will no longer fail "silently", but it will always try to tell you what's wrong.

Added more compatibility options, "REVERSE", "LOWPRI" and "IGNOREVERIFY".

Release 40.18:

MuMove4K still reserves 32K of memory, as ever, but PREPAREEMUL moves the chip memory lower bound to the 16K line instead of just 8K. This might make it more compatible to Fusion. Thanks, Pavel.

Release 40.19:

MuMove4K contains now a checksum and some other sanity tests. In case something should have damaged MuMove4K, it will disable all reset resident programs and will show a red screen, waiting for a reboot.

Release 40.21:

Fixed an important bug in MuMove4K, namely it forgot to bypass a Kickstart ROM checksum deep in its logic. This might have caused a mess on various systems.

Release 40.22:

The pre-computed memory type for the PREPAREEMUL option was, in fact, broken and caused mis-function of MuMove4K on boards with true autoconfig memory. This bug can be worked around with INCHIPMEM for the 40.21, but it got fixed in the 40.21 anyhow.