MuEVD

Thomas Richter

COLLABORATORS

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MuEVD

1.1 MuEVD Guide

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

The ShapeShifter tooltype REMAP8K must be off to use this video driver. Use MuFastZero instead to speed up the ShapeShifter.

MuEVD Guide

Guide Version 1.02 MuEVD 40.2

© THOR-Software

Thomas Richter

Rühmkorffstraße 10A

12209 Berlin

Germany

EMail: thor@math.tu-berlin.de

WWW: http://www.math.tu-berlin.de/~thor/thor/index.html

1.2 The THOR-Software Licence

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 MuEVD?

MuEVD is a mmu.library external video driver for the Macintosh "ShapeShifter" emulator which makes use of the MMU to speed up the display refresh. All other video drivers for the Mac are either slow, or they have to hack the MMU themselves, making them incompatible to the MuLib. Since this has to stop , MuEVD was born.

MuEVD supports all graphics modes you may dream of. It supports 1 bit, 2 bit and 4 bit Macintosh screens on all Amiga models with a working MMU, starting at 68020/68851 up to the 68060, which is clearly the benefit of using the mmu.library. It supports eight bit screens on the later AGA driven Amigas, and a HAM8 emulation for 15bit deep Macintosh screens for AGA machines without graphics boards. It also supports both the Picasso96 and the CyberGraphX software and all graphic boards these two packages are able to support, provided the graphics board has "linearly mapped memory". All pixel formats from "Eight Bit Chunky" to "Hi Color" and "True Color" are supported. If the Picasso96 graphics system is used, MuEVD offers "direct access" to the video RAM if possible, to save memory and CPU power. Even this direct access is "MMU-driven", to avoid the usual graphic trash if the ShapeShifter tries to draw directly into the Video RAM of the workbench.

You hardly find any video driver as versatile and flexible as MuEVD; it implements features of the simple P96 driver, the CGfx video driver, TurboEVD and CardTrickEVD.

1.5 MuEVD requirements

MuEVD is for first a MMU driven video driver. This means, of course, that your system must provide a working MMU. The MMU type does not matter, MuEVD - or to be precise - the mmu.library supports all of them. A 68020/68851 would be good enough, even though you won't have much fun with it.

As for the graphics modes: An OCS Amiga is good enough, MuEVD is able to support one to four bit graphics modes on them. A graphics board is clearly preferable, but AGA is also supported up to a 15bit emulation mode using the HAM8 mode.

Memory: Oh yes, please. Lots of! MuEVD is memory hungry, due to its working principle. At least 8MB are required, but don't expect a lot of fun on these systems. You will be able to run it, and ShapeShifter, but not much more. 16MB is maybe considered the bare minimum for a working and useable Mac setup. MuEVD requires at least thrice (that is, yes, three times!) as much memory as the video mode itself requires.

Compatibility: MuEVD is by construction compatible to all MuTools and all programs that make use of the mmu.library to program the MMU. It is therefore incompatible to MMU hacks. This includes some ShapeShifter features as well: The REMAP8K tooltype of the emulator MUST BE SWITCHED OFF. This is not a real loss, the command line

MuFastZero FASTEXEC

will provide the same, but in a more compatible way. Please study the MuFastZero guide for more information on this tool.

Additionally, make sure that no other ShapeShifter MMU hacks are enabled as well: "ROM-write protection" on the "Miscellaneous" control page of the ShapeShifter preferences must be disabled, too.

Additionally, MuEVD is incompatible to the ppc.library. This is not my fault, I don't get the documentation to support it.

1.6 Installation of MuFastZero

- 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.

- Copy "MuEVD" into the "ShapeShifter/Video Drivers" directory where all other video drivers are kept.

- Click on the "ShapeShifter" icon and select "Information..." from the Workbench tools menu. Check the tooltypes. If one of them says "REMAP8K", remove it.

- Start the ShapeShifter, and click on "Graphics....". Select "External" as Screen type and "Video Drivers/MuEVD" as External driver.

- The screen mode and the color depth should now be selected. Both should "fit" together in order to get optimal performance.

On a native Amiga screen, i.e. "PAL, NTSC, Productivity, ...", one bit to four bit screens are supported. Eight bits requires AGA, 15 bit HiColor emulation requires AGA, too. The higher the depth, the slower the system and the more memory is required; One bit screens, i.e. black and white displays, are fastest, but they really do not need a MMU to be that fast anyhow.

On a graphics board, "chunky" "8 bit color lookup modes" fit to the color depth "8 bit" selection. This combination is optimal in speed. Color depth "15 bit" requires either a 15 or a 16 bit "HiColor" screen. If possible, try to select a "15 bit" mode with "non-PC" addressing as this will be pretty optimal. Everything else is slower, but supported. For the color depth "24 bit", a "True Color" screen mode should be selected. If available, a screen mode with "alpha channel" might require more memory, but might be faster here as well. If this is an option, try to find an "RGB" type pixel format and avoid the "BGR" modes. Both of them are supported, but "BGR" is slower. If possible, run Picasso96 and a screen mode that is supported by the Mac directly: These are either 8 bit "chunky" modes, "15 bit RGB" (non-PC) modes or "24 bit RGB with alpha channel" modes. MuEVD will then switch to direct video RAM access, which helps performance and which will cost only one third of the memory. Finally, the four bit planar modes some graphics adapters are able to offer are unsupported. Why use less colors if "8 bit" is optimal?

For the screen size: Due to the internal working of MuEVD, the width must be a multiple of 32. The width should be larger than 400, the height larger than 256. Less might work, but neither MacOs nor you will have fun.

The "Refresh rate" should be set to "1". MuEVD is fast enough for this, even on my 68030/40Mhz machine. MuEVD supports also internal refresh for the native Amiga screen modes and then supports "0" as well, but this must be configured somewhere else . The same goes for the "Black border" checkmark since this flag is unfortunately not forwarded to MuEVD, check the preferences section for details. The "MMU Refresh" should be left off since MuEVD does all this itself, and the MMU refresh is implicit anyhow.

- Close the graphics mode selection window, and click on "Memory". Do NOT give the ShapeShifter all the memory available or there will be no memory left for MuEVD. Except for the graphics buffer itself, MuEVD has to reserve memory for the MMU tables, for the conversion buffers, for an additional delta buffer, and for the Macintosh screen buffer emulation. On a 8MB machine, keep at least 2MB of memory, but that's really tight. On a 16MB machine and a graphics board, 4MB spare memory would work.

- Close this window and click on "Miscellaneous". Make sure that the "Write-protect ROM" check mark is disabled. This is just another MMU hack that should be avoided.

- Save the ShapeShifter settings.

About the 15 bit HAM8 emulation: Due to limitations of the HAM mode, graphics in this mode might look a bit "washed out" and "smuddy". Moreover, the first time you start MuEVD - after installation or after a modification of the preferences file - MuEVD may take up to half a minute to re-compute the optimal palette. It will save a backup of this palette to "ENVARC:MuEVD.colorcache", so this recomputation step is not required the next time and startup will be "fast" again.

That's all, congratulations!

1.7 Customization of MuEVD

MuEVD can be fine-tuned by its preferences file. This is a plain ASCII text file named "ENV:MuEVD.prefs". Just open or create this file with a text editor, and copy it manually to "ENVARC:" when you're done so it persists the next coldstart. There is currently no preferences editor, this might come later.

The first line of the preferences file sets some MuEVD specific options. Currently, the command line template is

BORDERBLANK=BB/T,OWNREFRESH=OR/T,REFRESHRATE=RR/N,EPSILON/N,REFRESHED=REF/T

For example, a command line might look like as follows:

BORDERBLANK yes OWNREFRESH yes REFRESHRATE 0

The meaning of the options is as follows:

BORDERBLANK: If set to "yes", MuEVD will try to blank the screen border instead of displaying it in the screen background color. This requires at least the ECS chip set. Graphics boards blank the border anyhow.

OWNREFRESH: If set to "yes", MuEVD will refresh its screen itself. This refresh method might be smoother than the native ShapeShifter refresh, but it works only for the native Amiga screen modes. This argument is ignored for Picasso96/CyberGraphX screens; due to the way how display arbitration works for both graphics systems, an interrupt driven screen refresh as used by the "OWNREFRESH" option is not available.

REFRESHRATE: This is the refresh rate; the lower the value, the more often the screen is refreshed. Most Amigas would support "1" here as value, all reasonably fast machines would even allow the minimum "0". Even my 68030/40Mhz is "reasonably fast enough" for MuEVD, so give it a try. It shouldn't crash in either case. As for OWNREFRESH, this argument is ignored for Picasso96/CyberGraphX systems. The refresh rate should then be adjusted by the ShapeShifter "Graphics..." control window.

EPSILON: This value is only required for the "15 bit" HAM8 emulation for AGA amigas, it is otherwise ignored. It controls the use of the "HAM" colors. It's effect can be described as follows:

If EPSILON is large, MuEVD will use only few HAM colors. This will reduce the typical HAM artefacts, but MuEVD won't be able to pick the precise colors since there are only 64 non-HAM generated colors. The highest possible value is here 32767. On the other hand, a lower value will give more precise colors, that is the intended colors will match more precisely the colors you see on the screen. The drawback is that this means more HAM artefacts. Higher values should be used for graphics applications where exact representation of images is required, i.e. "PhotoShop" or similar applications, whereas lower values are better suited if fine lines and tiny details should remain visible, as in word processing. A value of 240 is the default and presents a useful compromise.

REFRESHED can be specified if MuEVD should be forced into "refresh" display modes even though direct access is possible. One bit "native" screens are always refreshed since chip memory is so slow, but if possible, MuEVD will enable "direct access" to the video RAM. If this is not desirable because refreshed access is faster - as for some Z-II boards - then specify this option. Note that direct access to graphics boards RAM requires Picasso96 and a Mac supported pixel format.

All following lines can be used to define the palette for the HAM8 emulation mode, one line defining one entry in the palette. The template for the second and all following lines looks as follows:

RED=R/A,GREEN=G/A,BLUE=B/A

For example, the following line would define a palette entry for "dark grey":

R=0x1C G=0x1C B=0x1C

Numbers can be given here in decimal notation, or alternatively in "hex" notiation with a leading "0x" or "$". "R", "G", and "B" values specify then the color components of red, green and blue, and range from "0" = no intensity to "255" = "0xff" for maximum intensity. A sample preferences file can be found in the distribution.

Finally, a word about the "15 bit" HAM8 emulation: If the preferences file is changed, MuEVD has to re-compute its internal palette. This may take up to half a minute; even though MuEVD then seems to "hang", feel asured that it will continue working. It will then create or re-build its color cache file to speed up loading the next time.

1.8 Technical Details

MuEVD is build on top of the mmu.library and its functions. This means especially that MuEVD is by construction compatible to all MMU types and all members of the MC68K family, provided a MMU is available at all. To provide a display buffer for the MacOs, MuEVD allocates this memory itself, and marks the memory region of this display buffer as "MAPP_INDIRECT". This means, MuEVD provides its own MMU descriptors for the display buffer. How these descriptors are made use of depends, however, highly on the environment in which MuEVD is run.

In ECS/AGA mode, these descriptors are checked periodically, either by the "EVD_Refresh" call of the ShapeShifter display refresh task, or, if "OWNREFRESH" is selected, by an interrupt routine. If one of the descriptors is marked as "modified", the corresponding memory region of the display buffer has been touched by the MacOs and has to be re-build. MuEVD uses some advanced chunky-to-planar conversion routines for this purpose.

In CGfx mode, or if the display organization of the graphics card is none of the supported organizations of the MacOs, this refresh technique is used as well, except that its now the matter of a support routine to convert the graphics format used by the MacOs to a format that is understood by your graphics card. The only difference is now that the corresponding graphics card bitmap must be arbitrated from CGfx or P96, an operation which cannot take place in an interrupt routine. Therefore, interrupt driven CGfx/P96 video modes are not available.

In Picasso96 direct access mode, everything's different. The mmu.library is still used to build the descriptors, but they are used in a completely different way: The Mac is told to find the video RAM at the location of the chunky frame buffer in Fast RAM, but whenever the bitmap of the MacOs screen is loaded to the Video RAM of the board, this chunky buffer is remapped by means of the MMU to the real video RAM buffer of the graphics card; hence, all drawing operations go directly to the video card and don't require refresh. Whenever the screen "looses the board", Picasso96 is able to call a user-supplied hook function. This function is made use of to copy the graphics data from the board back to the chunky buffer and to disable the remapping. If this happens, the logical address of the video RAM does not change at all - as seen from the Mac Os. But since the MMU has been reprogrammed, all memory accesses will now go into the physical Fast Ram buffer and no longer into the video RAM. Something similar happens if the Mac screen gains the card: The data from the chunky buffer is copied back into the video RAM and MMU remapping is enabled again. Since up to my knowledge, CGfx does not support these hooks, this direct un-refreshed access mode is currently only available for Picasso96.

1.9 History

Release 40.0: This is the initial release of MuEVD and the first Aminet distribution. Thanks goes to Luca "Hexaae" Longone for continuously "pushing" me, or this project would have never happened. I as a GVP spectrum graphics card owner was never in the need of a MMU driven video driver itself. Big thanks goes to "Aki Laukanen" and his "TurboEVD" sources including the rather genius chunky to planar conversion algorithm. Even though, it turned out to be easier to rewrite everything from scratch, it was very interesting to study the sources and to learn how a ShapeShifter video driver works. Finally, this algorithm was only used for the 15 bit and 8 bit "planar" modes; even though I think I could find something possibly faster for the remaining modes, I highly appreciate the huge amount of brain-power that can be seen thru this algorithm. Excellent work, and I'm really sorry that I haven't been able to come up with something as genius as this one myself. (-:

Release 40.1: MuEVD did not rewrite the color cache file in case the EPSILON value in the preferences changed. Fixed! MuEVD sets now the display refresh type to "NONE" in case it uses its own refresh routine. This may help performance a bit as the "ShapeShifter Refresh" process is then no longer needed.

Release 40.2: Fixed some minor flaws in MuEVD. It didn't release some MMU memory correctly on exit and did not clear the "USED" and "MODIFIED" page attributes as well. Furthermore, due to the MMU address translation cache, it could happen that MuEVD forgot to refresh some parts of the display correctly. This may still happen in this release, but it is much less likely. Thanks to Stephen Brooks for reporting the bugs.