MuFastChip

Thomas Richter

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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MuFastChip

1.1 MuFastChip Guide

´### ,#. ,#### ,### ####’ #### ,#### ,###’ ####’ #### _____ ,#### ,###’ || ####’ #### | ___ ___ | ___ ,#### ,###’ ---- | | | | | | |___ ####’ #### | | | | | | | ,#####. ,###’ . ||___||___||_ ___| ####’##. ,#### ,## ,#### ########’ # ,##’ ####’ `####’ `###’ ,#### #####

MuFastChip Guide

Guide Version 1.02 MuFastChip Version 40.3

The Licence : Legal restrictions

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

What is it : Overview

Installation : How to install MuFastChip

Synopsis : The command line options and tool types

History : What happened before

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 "MuFastChip" and the "MuFastChip.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 MuFastChip?

MuFastChip optimizes the access rules of the 68040 and 68060 processors to speed up chip memory accesses in a MMU library compatible way. It does this by changing the cache mode for chip memory.

The MMU library will already build its own MMU tables in an optimized way in case the MMU is inactive when it gets loaded, but in case you've to use the mmu.library "on top" of a third-party 68040 or 68060 library, this tool might improve the MMU table layout as setup by these third-party libraries.

"MuFastChip" is so to speak a "special" version of the MuSetCacheMode program, just the same can be archived "by hand" either by using MuSetCacheMode, or by editing the "ENVARC:MMU-Configuration" file by hand. For simplicity and convenience, this program is included in the MuTools anyhow.

This tool can be run if a 68030 or 68020/68851 MMU is in use, but it won't do very much. These older MMUs do not support the faster access mechanism of the 68040 or 68060.

1.5 What's a cache?

All members of the Motorola family, starting with the 68020, come with one or two build in data buffers, the so-called "caches". A cache is a small buffer within the CPU itself that keeps frequently accessed memory locations; hence, it avoids unnecessary bus accesses and memory reads and writes of data recently accessed and therefore speeds up the CPU operation. However, special care must be taken in case the "memory location" is in fact a hardware component, e.g. an I/O port that might change its contents without the knowledge of the CPU. If this port would be buffered, a program trying to read from the port would access the internal CPU caches instead of trying to re-read the up-to-date data from the port. It is therefore important to tell the CPU not to buffer I/O ports. Since the chip-memory is accessed by the blitter bypassing the CPU, it must not be buffered as well.

What MuFastChip now does is that it sets the cache mode to "IMPRECISE NONSERIALIZED".

"IMPRECISE" is a flag that is understood by the 68060 only, it is ignored by the 68040 MMU code. It means that the 68060 is allowed to be a bit "sloppy" on access and is allowed to report "bus errors" in a way that is not always "fix-able". This doesn't hurt for chip memory access because the hardware never generates bus errors for chip memory anyhow.

"NONSERIALIZED" is only understood by the 68040 MMU driver. This flag allows the 68040 to re-order reads and writes to memory to allow optimal performance. This means especially that the actual writes to the hardware need not to appear in the order they have been programmed. However, this is not a problem at all for memory accesses and can be tolerated by chip memory writes to optimize the access speed.

Both flags are ignored by either the 68030 and the 68020/68851. Hence, running "MuFastChip" on these machines doesn't improve the performance at all, but it doesn't hurt either.

1.6 Installation of MuFastChip

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.

- Copy "MuFastChip" wherever you want.

- Run it in the startup-sequence or in the WBStartup drawer.

1.7 Command line options and tooltypes

MuFastChip 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 "MuFastRom" 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.

MuFastChip ON/S,OFF/S

ON/S

Enables the MuFastChip function. This is also the default option.

OFF/S

Disables MuFastChip and re-enables the slower chip access protocol.

When started from the workbench, MuSetCacheMode knows one additional tooltype, namely:

WINDOW=<path>

where <path> is a file name path where the program should print its output. This should be a console window specification, i.e. something like

CON:0/0/640/100/MuSetCacheMode/AUTO/CLOSE/WAIT

This argument defaults to NIL:, i.e. all output will be thrown away.

1.8 History

Release 40.1:

This is the first official release.

Release 40.2:

The shutdown code did not unload the program if run from the workbench. Fixed. The code is now a bit more error tolerant.

Release 40.3:

Added a check whether the program is really required, and prints a warning in case it is not.