

rtgmaster_user

COLLABORATORS

	<i>TITLE :</i> rtgmaster_user		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		June 24, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	rtgmaster_user	1
1.1	Rtgmaster.library V32.0	1
1.2	RtgMaster Library V32.0	1
1.3	RtgMaster Library V32.0	2
1.4	RtgMaster Library V32.0	4
1.5	RtgMaster Library V32.0	6
1.6	RtgMaster Library V32.0	6
1.7	RtgMaster Library V32.0	7
1.8	RtgMaster Library V32.0	7
1.9	RtgMaster Library V32.0	8
1.10	RtgMaster Library V32.0	9
1.11	RtgMaster Library V32.0	11
1.12	RtgMaster Library V32.0	12
1.13	RtgMaster Library V32.0	12
1.14	RtgMaster Library V32.0	25
1.15	Rtgmaster.library V32.0	26
1.16	Rtgmaster.library V32.0	27
1.17	Rtgmaster.library V32.0	27
1.18	Rtgmaster.library V32.0	28
1.19	Rtgmaster.library V32.0	28
1.20	Rtgmaster.library V32.0	29
1.21	Rtgmaster.library V32.0	30
1.22	Rtgmaster.library V32.0	30
1.23	Rtgmaster.library V32.0	31
1.24	Rtgmaster.library V32.0	31
1.25	Rtgmaster.library V32.0	31
1.26	Rtgmaster.library V32.0	32
1.27	Rtgmaster.library V32.0	32
1.28	Rtgmaster.library V32.0	32
1.29	Rtgmaster.library V32.0	33

1.30 Rtgmaster.library V32.0	34
1.31 Rtgmaster.library V32.0	34
1.32 Rtgmaster.library V32.0	35
1.33 Rtgmaster.library V32.0	35
1.34 RtgMaster Library V32.0	39
1.35 RtgMaster Library V32.0	41

Chapter 1

rtgmaster_user

1.1 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

User Documentation

Table of contents

Part A : [About RtgMaster](#)

Part B : [Coder Docs](#)

Part C : [Supported Software](#)

Part D : [About Amiga GFX Boards](#)

Part E : [Using it](#)

Part F : [Benchmarks](#)

Part G : [c2p algorithms](#)

Part H : [History](#)

Part I : [Authors and Copyright](#)

Part J : [The Goodies](#)

Part K : [The CV64 Problem](#)

Part L : [Speaking about AGA-only](#)

Part M : [WB Window Support](#)

Part M : [PPC Support](#)

1.2 RtgMaster Library V32.0

Starting with rtgmaster.library V32, it is possible to render the graphics on a Workbench Window instead of in a screen. This slows down the display a lot, of course, so it should only be used as OPTIONAL feature. It only works on CyberGraphX/Picasso96/Probench 3/ECS/AGA. On all other Display Hardware (which means: on EGS) the rtg_Workbench Tag will simply be ignored and always a Screen is used.

To add Workbench Window Support you currently have to find out, if the user wants to

run the Application on a Window manually, for example using CLI Parameters. In the next version of rtgmaster this will be integrated into the Screenmode-Requester, though.

To open a Window instead of a screen, put the Source Data Format as Tag Value to `rtg_Workbench`, when you call `OpenRtgScreen`. Currently the Workbench Window has several limitations:

- Source Data has to be 1-8 Bit, so always put 512 or 0 to `rtg_Workbench` (512 to enable, 0 to disable). Later maybe support for 15-24 Bit on Workbench Window will be added
- The SrcX/SrcY Parameters of some Chunky-Copy Functions will be ignored in Workbench Window Mode
- The "Masked" Parameter of `CopyRtgBlit` will always be handled as if it had the value 0
- Doublebuffering does not work on Workbench. You should not even write something to the second or third Bitmap, even if it is NOT displayed
- Direct Video RAM Accesses are not allowed (while they are allowed, when you write to a screen)
- Width of Chunky-Copy always has to be divisible by 16
- `CopyRtgPixelFormat` does not Planar->Planar Copy for Workbench Display, but it does the same like `WriteRtgPixelFormat` instead (c2p... and no Source Offsets in WB Windows... remember ?)

And this is how the user starts the program then:

For example if you start flame with parameter window

flame window

(flame supports the Window-Feature), flame opens on a window. The user chooses the Screenmode, as always, but the Screenmode is not used as screenmode, it is used as Window-Size, instead.

Note, that in the next version a `smr_Workbench` will be added, to be able to choose in the Screenmode-Requester, if the program should run on Workbench or not. I hope to have this next version finished next week.

Note: There are floating some Preview-Versions of rtgmaster PPC around (V31) where the WB Support did not work properly. Not that this is a misunderstanding: The people spreading this preview versions HAD my permission to spread them !!!

1.3 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

In the past, there have been some reports of rtgmaster not running on AGA-only systems, which gave some confusion. I want to solve the confusion now. Actually, there were several (now solved) problems.

- Early versions of rtgmaster took too much ramlib-space, so that rtgmaster did not run when tools like MCP were active. I fixed this bug with rtgmaster V24.

Now rtgmaster runs parallel with MCP (Apparently especially many AGA users

use MCP...)

- rtgmaster had a serious bug in the Library Init Code, that made it crash on some (mostly: AGA-)Systems. This was fixed with rtgmaster V25. On all other systems it ran just fine.

- rtgmaster V25/V26 had a small bug, making it crash on AGA-only-Systems, fixed in rtgmaster V27. (Well, before V27 i did not too much testing on AGA-Only Systems. I changed this now)

There is one more thing to discuss:

rtgmaster seriously needs a CORRECT installed AmigaDOS. When you install your system using the Workbench 3.x Installation disks, somewhere in your startup-sequence there will be a

```
IF EXISTS DEVS:Monitors
```

```
IF EXISTS DEVS:Monitors/VGAOnly
```

```
DEVS:Monitors/VGAOnly
```

```
EndIF
```

```
C>List >NIL: DEVS:Monitors/~(#?.info|VGAOnly) TO T:M LFORMAT "DEVS:Monitors/%s"
```

```
Execute T:M
```

```
EndIF
```

or a

```
IF EXISTS DEVS:Monitors
```

```
C>List >NIL: DEVS:Monitors/~(#?.info|VGAOnly) TO T:M LFORMAT "DEVS:Monitors/%s"
```

```
Execute T:M
```

```
EndIF
```

created. These strange lines install the Screenmodes to the system. If they are not present, rtgmaster won't run. Actually, this is not a bug. A correct installed Workbench HAS this two lines in the startup-sequence !!! (They have to be called before the IPREFS command).

If these lines are not present on your startup-sequence, please add them. BTW, a GFX Board will not work without these lines (anyways, what software one uses), because of that such problems only appeared on AGA-Systems. On GFX Board Systems those lines are present by default.

But well... they are installed by the Installer program on your Workbench Disks, so they should be there. Also install a monitor-Driver !!! (PAL or NTSC, for example...)

1.4 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Maybe you did not notice, but: This rtgmaster version is a WarpOS Mixed/Fat Binary, with both code for PPC and 68k !!!

What does this mean ? Well, of course this version of rtgmaster still runs on plain 68k Systems (and future versions will also do...), but it has also code for PPC inside. This code is not activated automatically. In fact, only doing the Screen refresh on PPC and the rest on 68k would not speed up a program, in fact it would slow it down, due to the Contextswitches. And no "Message-System" can help with this. So, if you use stuff like Genetic Species or Amigadoom or Diversia with rtgmaster, even if you have a PPC Board, only the usual 68k rtgmaster functions, well known from earlier versions, will be used.

It would be possible for coders, though, to create games, which run nearly completely on the PPC, by using the extra PPC Functions of rtgmaster PPC (more information is found in the developer docs `rtgmaster_dev.lha`). These functions, of course, will only run on the PPC. You can do Screen refresh and Keyboard/Mouse Handling completely PPC Native then, without Contextswitch.

Some notes about my choice of the WarpOS/WarpUP Software for the PPC Part of rtgmaster:

- WarpOS is very efficient
- WarpOS does not create a task each time a PPC function is called from 68k
- WarpOS is - thanks to StormC - very easy to use, what is NOT the case for the available compilers for `ppc.library`
- WarpOS was created in the beginning as the coder was disappointed with the speed and stability of `ppc.library`
- Boards from other manufacturers which will soon be manufactured, will support WarpOS, but not `ppc.library`
- major Amiga Game firms (and those are the guys that will use rtgmaster PPC) chose WarpOS as the PPC Software to use also
- Fat/Mixed Binary Support
- No Foreign Elf Executable Format needed
- PPC Shared Library Supported

I am absolutely convinced that WarpOS is the "Amiga Style Solution". Please don't mail me, if you think otherwise. And at this place a greeting to the "Temple of Phase 5" in `comp.sys.amiga.games` with High Priest "knoble". And Yes, Mr.Knoble, I KNOW about what i am speaking. Even if i don't tell names in certain things, these are not just rumours. Ever heard the term "Non-Disclosure Agreement" ?

Well, some people still need to wake up...

Ah, in the Goodies-Section i included the old Version of rtgmaster (V27) to support ppc.library guys like knoble... of course V27 only includes the 68k Stuff... and some of the latest features of rtgmaster are not included...

but well... i *had* to select one of the two PPC Softwares... and WarpOS is definitely the better one... and also the one which will soon run on more different Boards (other manufacturers will release WarpOS-based PPC Boards, still ...)

As to how to include PPC functions into rtgmaster:

ASM:

Have a look at the docs of StormPowerASM or of ppctut.lha, about how to use PPC Shared Libraris, i don't repeat it here...

C:

- Do not include the pragma-file, only include the clib-file
 - Now simply use the commands. For each rtgmaster 68k Command, there exists a PPC Command, like for example PPCopyRtgBlit being the PPC version of CopyRtgBlit.

- Some of these commands use Context-Switches, some of them are 100% PPC Native.

With CheckPPCCommand (68k Version) and PPCCheckPPCCommand (PPC Native Version) you can check, if a command is 100% PPC Native. If this is the case, the value 1 is returned, else 0.

CS: Function uses Contextswitch

- : No PPC available, not even Contextswitch

Name given: 100% PPC Native

rtgP96.library rtgCGX.library rtgAMI.library rtgEGS.library

```
=====
PPCCopyRtgBlit PPCopyRtgBlit PPCopyRtgBlit -
PPCLockRtgScreen PPCLockRtgScreen PPCLockRtgScreen -
PPCUnlockRtgScreen PPCUnlockRtgScreen PPCUnlockRtgScreen -
PPCGetBufAdr PPCGetBufAdr PPCGetBufAdr -
PPCCopyRtgPixelFormatArray PPCopyRtgPixelFormatArray - -
PPCRtgSetTextMode PPCRtgSetTextMode PPCRtgSetTextMode -
PPCRtgSetTextModeRGB PPCRtgSetTextModeRGB PPCRtgSetTextModeRGB -
PPCRtgInitRDCMP PPCRtgInitRDCMP PPCRtgInitRDCMP -
PPCCopyRtgBlit PPCopyRtgBlit PPCopyRtgBlit -
PPCRtgScreenAtFront PPCRtgScreenAtFront PPCRtgScreenAtFront -
PPCDrawRtgLine PPCDrawRtgLine - -
PPCWriteRtgPixel PPCWriteRtgPixel - -
PPCRtgWaitRDCMP PPCWaitRDCMP PPCWaitRDCMP -
```

PPCRtgGetMsg PPCRtgGetMsg PPCRtgGetMsg -
PPCRtgReplyMsg PPCRtgReplyMsg PPCRtgReplyMsg -
PPCFillRtgRect PPCFillRtgRect CS -
PPCFillRtgRectRGB PPCFillRtgRectRGB No 24 Bit for AGA -
PPCWriteRtgPixelArrayPPCWriteRtgPixelArray- -
PPCCallRtgC2P PPCCallRtgC2P PPCCallRtgC2P -

All other PPC functions simply perform a contextswitch to 68k.

If there are still questions about PPC Support, mail your questions to
MagicSN@Birdland.es.bawue.de (that is my email...)

1.5 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Normally we assume that

BytesPerRow = Width * BytesPerPixel

This is **WRONG** for the Cybervision 64 (only 64, not 3D !!!)

At least for some Screenmodes...

The LEGAL way to get BytesPerRow is using GetRtgScreenData with
grd_BytesPerRow !!! If you do not do it this way, your program
might not run on the Cybervision64 !!! Please use GetRtgScreenData !!!

1.6 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

rtgmaster Goodies are some extra some to rtgmaster. Currently they are :

- asmconv : A program to convert 86x86 code (PC source) to 680x0 code.

You have to do some stuff manually, though, still. This is useful,
as a lot of PC Demo Sources are on the net. asmconv was written
by Nikolaus Mausz on the PC, who gave his OK for me porting it
to Amiga and including it to the rtgmaster package. NOTE: You have
have to supply the PC Sources to be converted in PC text format.

- rtggadtools : This is an rtgmaster expansion by Hans-Joerg Frieden.
It provides rtgmaster with Gadget-Functionality. It currently only
runs on GFX Boards.

- You might also have a look at the demos directory. There are a lot
of rtgmaster example sources found there.

1.7 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Coder Docs

Well, as described in [About RtgMaster](#) rtgmaster.library

is a system to provide your games/demos with GFX Board Support. This section is an introduction about how to code this.

If you think you can't make it, even with this excellent library, please contact me per email. I might do the GFX Board coding part for you, if needed.

In this Docs i will describe the process of GFX Board Support in C Syntax, but of course it works in ASM too.

1. [Opening Display](#)
2. [GetRtgScreenData](#)
3. [Methods of Screen-Copy](#)
4. [Input Handling](#)
5. [Some Extra-Calls](#)
6. [Support for Network-Gaming](#)

Also, in the Goodies Directory, there is some already compiled source, that provides a driver containing everything you might need. It already opens all needed libraries, so those among you who are not yet familiar to the useful concept of shared libraries, can use this driver.

1.8 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Note: This Doc-File is not quite finished yet. Have also a look at extradocs.txt !!!

Opening Display

The Amiga Display Database is, as you probably know, organized in Screenmodes.

So the first thing you will have to do is opening a Screenmode-Requester.

RtgScreenModeReq

will do this job. You can describe the screenmodes that are interesting for your application to the call. For example, if you want only Chunky (GFX Board) 8 Bit Modes between 320x200 and 640x480, only such modes will be listed in the Screenmode-Requester. Have a look at the examples in the Demo Directory for this. Alternatively, starting with rtgmaster V24, there are a function to allocate a list of ScreenReq structures, and a function to get the BEST fittable Screenmode.

Next, as the user chose a Screenmode, we have to open a Screen with this

Screenmode.

OpenRtgScreen

After the Screen is open, it has to be locked for private usage.

LockRtgScreen

Then you change the colors

LoadRGBRtg

And last, we will have to ask for the Base Address of the Video RAM, so that we can access the Video RAM directly. If we run the application on ECS/AGA, we will have a pointer to the Chip RAM of the Screen instead.

GetBufAdr

This is used to get the Video RAM Base Address.

RtgWaitTOF

With this function you can wait on the Vertical blank. Starting with V24, there is also a RtgCheckVSync(), which does not wait, but on most Boards, it does just also wait (as a lot of Boards do not support this).

CopyRtgPixelArray

This is a function to copy data to the Screen. It always uses the native format. Which means, on AGA it uses Planar Format (the AGA version does currently only work, if the Offsets are divisibly by 8). If you want to display Chunky Data on AGA, you can use CallRtgC2P for using a c2p algorithm.

CopyRtgBlit

CopyRtgPixelArray did not support Source Modulo. So i added this function (CopyRtgPixelArray is mainly for downward compatibility and under AGA for Planar Copy). On AGA CopyRtgBlit uses c2p.

There are lots of other useful calls.

1.9 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

GetRtgScreenData

GetRtgScreenData is a function to examine an opened screen. It will find out, if the Screen is Planar or Chunky, 4 or 8 or 24 Bit and many more stuff. Well, if you only want your application on 8 Bit Chunky and AGA it is enough to test for `grd_Planar` and `grd_Chunky`. Then you only need two display functions, one for AGA, one for GFX Boards. This is described more detailed in the next section. There is also a possibility to find out current mouse position with this call.

1.10 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Methods of Screencopy

As to the methods of Screencopy, there are basically 3 sorts :

A) Using a Fastram Buffer

You render your buffer in Fastram. Then, after you completely rendered a frame, you use a direct Video RAM access function to copy your buffer to Video RAM.

You can use GetBufAdr to find out about the Video RAM Address.

CopyRtgPixelFormatArray

This function uses COPY_MOVE_MOVE, which is a quite fast Copy-Function. CopyRtgPixelFormatArray does not do any clipping checks or something like that, to get the best speed. If you need Source modulo, use CopyRtgBlit, instead (needs at least rtgmaster V24).

It also does a c2p (only synchronous c2p) on AGA.

B) Using Doublebuffering

With Doublebuffering, both buffers are found in Video RAM, using the GetBufAdr call. You then render to the buffers and switch to each other using

SwitchScreens

Comparing A) to B), the Fastram method has a faster rendering, because we do not have to render over the Zorro Bus. On the other hand, Doublebuffering displays much faster. Altogether it seems that on most WB Emulations those functions have nearly the same speed. With future Amiga Clones using PCI Busses, probably Doublebuffering will be the faster one. Currently (of course depending on the engine) mostly the Fastram method is slightly faster.

C) Using the GFX Board Blitter

Most GFX Boards have very fast blitters. So (theoretically) we could render in Video RAM and fake Doublebuffering using those Blitters using RtgBlit.

As we again have to render in Video RAM this is usually the slowest method. But well, it is faster than using AGA, still.

Now, which method should be used ? I would vote for Fastram Buffer, personally, but additional support for the other two, especially DoubleBuffering, might be interesting. It depends much on the hardware, if Fastram Buffer or Doublebuffering is faster.

Now, which Memory-Copy-Functions should be used for Fastram-Buffer ?

A very easy, and also fast solution is COPY_MOVE_MOVE, that is using the rtgmaster call

CopyRtgPixelFormatArray

It bases on

```

move.l (a0)+,d0
move.l (a0)+,d1
move.l (a0)+,d2
move.l (a0)+,d3
move.l d0,(a1)+
move.l d1,(a1)+
move.l d2,(a1)+
move.l d3,(a1)+

```

The usual implementation of this function needs the Width divisible by 64. rtgmaster uses this function, if possible, else it uses simple Longword-Copy.

An alternative would be COPY_MOVEM_MOVEM, which works like :

```

movem.l (a0)+,lotsofregisters
movem.l lotsofregisters,(a1)
movem.l (a0)+,lotsofregisters
movem.l lotsofregisters,$20(a1)
movem.l (a0)+,lotsofregisters
movem.l lotsofregisters,$40(a1)

```

There is a third version. It has some problems, though. It uses the 68040 command move16 and because of that only works on 040/060 machines. Additionally, it has some requirements:

- X Offset has to be divisible by 16
- Bytesperline has to be divisible by 16
- Source and Destination Address have to be divisible by 16

The problem is that the Video RAM Base Address of a screen is only on some GFX Boards divisible by 16. So it will only work with some GFX Boards. If it is supported, it should be only OPTIONAL. rtgmaster does not use this move16 stuff. It would only be slightly faster, anyways (and you still could implement it yourselves, based on the GetBufAdr call);

Of course there exist lots of variants of the three functions.

Now, one more theme : What about AGA Support ?

For AGA, of course, you can't use CopyRtgPixelFormatArray or one of the other listed functions.

One solution is using

```
CallRtgC2P
```

This function calls CopyRtgPixelFormatArray for GFX Boards, for ECS/AGA it calls a c2p algorithm.

The problem is, for handling asynchrone c2p it has to do some signaling stuff, which slows the function down. So, if you want to handle ECS/AGA through rtgmaster (instead of doing native AGA Support), the best would be to check, if there is a Chunky or a Planar Screen, and use CallRtgC2P only for the Planar Screen.

Notice, that rtgAMI.library does not support the following calls correctly :

CopyRtgPixelFormat

WriteRtgPixel

WriteRtgPixelFormat

There was no time to finish it yet. And the GFX Board Part is probably the most interesting, anyways. AGA can be better handled through Native AGA Support. A nice idea, of course, is using the rtgmaster Screenmoderequester to find out if Native AGA Support should be launched or the GFX Board code (simply check with GetRtgScreenData).

1.11 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Using rtgmaster, there are several possible methods to do the Input Handling.

- Accessing the Hardware, of course.
- Using an InputHandler
- Using the RDCMP Port of the RtgScreen

I myself usually don't access the hardware directly, so i created the RDCMP Port (after i did not like the results of MouseEvents created by the Input Handler... they got lost sometimes).

To use the RDCMP Port you have to get some information using

RtgInitRDCMP

This function initializes some stuff and returns you an RDCMPData structure. This structure contains :

A Messageport (you better do not use it).

A 1<<mp_Signal (checking the signal and after this Wait-ing is the fastest method under Intuition to get Input Events, but i found out that under rtgmaster direct polling is much faster. In fact, checking the signal gave me fps drops sometimes).

The fastest RDCMP Message-loop is :

```
while(!end)
{
while (imsg=GetRtgMsg(RtgScreen))
{
if (imsg->Class==IDCMP_RAWKEY)
{
RtgReplyMsg(RtgScreen,imsg);
...
}
else if (imsg->Class==IDCMP_MOUSEBUTTONS)
{
RtgReplyMsg(RtgScreen,imsg);
```

```
...  
}  
}  
...  
}
```

With this loop you won't get fps drops. You have to include intuition/intuition.h (or .i), as i used some of the Intuition constants. rtgmaster only recognizes RAWKEY and MOUSEBUTTON Events using RDCMP.

In the RDCMPData you will find something else interesting : Two variables that contain a POINTER to the current Mouseposition. Not the Mouseposition itself, but a POINTER to it. This is a bit faster than using GetRtgScreenData. Not much though, but this is for the optimizers among you :). Have a look at the Autodocs for more information.

1.12 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Some Extra-Calls

rtgmaster supports additionally to the already mentioned :

- Text/Font Support (slow, though, it does not directly access the hardware)
- Changing Mousepointers (does not work for EGS/Picasso II Native)
- Networking Support
- GFX Board Blitters
- Linedrawing
- and some more minor stuff

1.13 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

TCP/IP for Newcomers

This section is for people who want to add TCP/IP Support to their games using rtgmaster.library, but do not know anything at all (or just not enough) about TCP/IP to do it. I want to add at the beginning, though, that current rtgmaster network support really does not do much checking for package loss. It was never tested with anything else then example programs on local networks. If you want professional network support, you might have a look at the AMarquee.library. This library is better suited for it. rtgmaster, after all, is a GFX library... but for most

cases it should do the network job, also...

Note: The TCP/IP functions were now moved out of rtgmaster.library to rtgextra.library. In future versions more functions will be found in rtgextra.library, for example some new color conversion functions.

1. The Basics - How does it work ?

=====

There are a lot of networks out there... Ethernet, Tokenring, ISDN, modem, Nullmodem,... all of those have different hardware, different package sizes, different protocols. Therefore IP was invented, a protocol, for which a driver exists for probably ALL existing networks.

On the primary protocol IP two new protocols (TCP and UDP) were invented that are better than IP for transferring data.

TCP and UDP use IP, but support some more things (for example errorhandling...).

TCP is a "connection oriented" protocol, you simulate something like a phone connection with it. UDP is a "connectionless" protocol, you simply send the data, and somewhen, they will probably arrive somewhere.

UDP is faster, but you are only allowed to send VERY short packages, and you nearly do not have any errorchecking at all.

This is the first version of rtgmaster.library that supports UDP.

I can tell you, for YOU it is only a different constant (SOCK_DGRAM instead of SOCK_STREAM), for ME it was a lot of work, as UDP really works quite DIFFERENT than TCP... but well, here it is... if you want to use UDP (if you are coding an action game, you SHOULD use UDP, as TCP is a bit too slow for an action game...), be sure to read the chapter about UDP in this document...

Now... we have some systems... on the systems there are running some applications. What do we want to connect? The Systems ? No, we want to connect certain APPLICATIONS on the systems.

An Example :

On the system 194.55.101.26 is running an application called CircleMUD (a TCP/IP based game). On 194.55.101.26 a lot of other stuff is running too, so CircleMUD is given the number 4000.

This number is called a PORT. With the PORT we can specify, which application on a system we want to connect to.

If a player wants to play CircleMUD (i ported version 3.0 of it to Amiga just recently, BTW... :)), he has to connect

to the SERVER 194.55.101.26 PORT 4000. The application running on HIS system is only a CLIENT.

Then all CLIENTS send there data to the SERVER, and the SERVER calculates some things with that data (who hits whom ? :)) and sends back some data to the CLIENTS. So, if you want to build a network with, lets say 12, players playing a game, you run the CLIENT on each of those systems and on one of the (the fastest one, probably...) additional, the SERVER.

If you only want to code a game playable for 2 players (2 connected systems...) there is another possibility. A CLIENT on one system, a SERVER on the other one. The Server sends data to the CLIENT, then the CLIENT sends data to the SERVER. Both display on the screen what the other guy is doing, and want data once more... for only 2 players, we do not need two Clients, as then the server can do Client job too... for more than 2 players that would be too complicated.

2. Enough of the theory - i want to code

=====

Rtgmaster.library up to now only supports the "One Client - One Server" method. The method "One Server - Lots of Clients" will be added to it as soon as possible. The following explanations are mostly needed for both methods, though... I am using C syntax, but of course it works in ASM too... C is just more easy to understand by most people... BTW, a "ready-to-compile" Example is in the Demo Directory of this package... play around a bit with it, before you try it on your own... :) (P2PServ.c and P2PClient.c)

a) How do i implement a Server

=====

- You need the port number at first. In this example, we take 3008 as port-number.
- Open the bsdsocket.library. That could not be done inside rtgmaster.library, because of some structural problems of AmiTCP.

```
struct Library *SocketBase=0;
SocketBase=OpenLibrary("bsdsocket.library",3);
```

It is REALLY called bsdsocket.library, not socket.library, even if the file in libs: might be called socket.library !!!

- Declare : struct RTG_Socket *sock;
- Call :

```
sock = OpenServer(SocketBase,3008,SOCK_STREAM,0);
```

SocketBase is the LibraryBase of bsdsocket.library, that you HAVE to provide. 3008 is the port on which the Server will run (the Server knows its one IP Address, therefor it does not have to be provided). SOCK_STREAM and 0 are just some constants. If you use SOCK_STREAM, you will get a TCP connection. If you use SOCK_DGRAM instead, you will get a (much faster, but unreliable) UDP connection. Be sure to read the special chapter about UDP, if you need a very fast connection.

The returned structure is a "Socket". A Socket is some stuff TCP/IP uses to access a certain application.

Now your Server is ready to wait for a connection.

b) How do i implement a Client

```
=====
```

- Now we need the IP number of the Server *and* the port, on which the game is running. Let's say, it is 194.55.101.26 and 3008. You HAVE to give the same port number for Server and Client, or you might get strange results or even crashes.

- Open the bsdsocket.library like for the server.

It is REALLY called bsdsocket.library, not socket.library, even if the file in libs: might be called socket.library !!!

- Declare : struct RTG_Socket *sock;

- Call :

```
sock = OpenClient(SocketBase,"194.55.101.26",3008,SOCK_STREAM,0);
```

SocketBase is the LibraryBase of bsdsocket.library, that you HAVE to provide. 3008 is the port on which the Server will run (the Server knows its one IP Address, therefor it does not have to be provided). SOCK_STREAM and 0 are just some constants. If you use SOCK_STREAM, you will get a TCP connection. If you use SOCK_DGRAM instead, you will get a (much faster, but unreliable) UDP connection. Be sure to read the special chapter about UDP, if you need a very fast connection.

The returned structure is a "Socket". A Socket is some stuff TCP/IP uses to access a certain application.

Now your Client is attempting to connect to your Server (Yes, it is already trying to... now call of connect() needed, like in AmiTCP :) And no call of socket, no damned sockaddr_in to configure and all that complicated stuff... everything done by those two functions :))

c) How do i initialize the connection between Server and Client

=====

This is the "One Server - One Client scheme", as it is currently the only one supported by rtgmaster.library, anyways. This will work a bit different with the "One Server - Several Clients" scheme, but i will add another paragraph that explains this, as soon as it is finished.

- Now we have to think... do we want "Blocking" or "Nonblocking" data transmission ? If we have take "Blocking", an application (Client or Server) that is currently transmitting data, WAITS until it gets its data, EVEN if the other side does not send the data up to now. Not very useful for a game, especially an action game.

Therefor, nonblocking exists. If a application is "nonblocking" and does not get its data AT ONCE, the Data Transmission call fails (and we will try later again to get the data, after we updated the display and similar stuff...)

Default is "Blocking". If you want to change this, do :

```
long mode=1;
```

```
RtgIoctl(SocketBase,sock,&mode);
```

Here sock is the socket, that from nowon should be "nonblocking". mode=1 specifies "non-blocking".

You can switch back with an additional call of RtgIoctl with mode=0 (0 = "Blocking"), but that does not make much sense in my opinion. Simply call it with mode=1 for "Nonblocking" Mode...

My suggestion : Use "Blocking" for the Server, "Non-Blocking" for the Client, that seems to be the fastest (you have to remember : "Non-Blocking" does Busy-Waiting... of course a third solution would be only check after a certain time interval if there are new messages...)

- IF WE HAVE TCP, we now have to Accept an incoming connection on the server side. We do this with

```
sockclient=RtgAccept(SocketBase,sock);
```

where sock is the socket of the Server. We get the Socket of the Client in return.

You only should do this (call RtgAccept), if this is a TCP connection... if it is a UDP connection use sock as Socket of the Client instead. (Sounds strange, yes, but the main difference between them both is, that every application uses its OWN Socket, not the Socket of the Application where it is connected to...). For UDP, you are "connected" as soon as there is a Server and a Client (well, UDP is CONNECTIONLESS and if a Server and a Client want to tell something to each other, they simply send the message on the net, without wasting time for synchronization... that's why UDP is faster :)

NOTE: This call WAITS, until we are connected (until the Client tries to connect). Now we see, why method c) is of no use for more than one Client... it can't handle data and wait for a new connection at the same time... for only 2 players, one using the Server Application, one the Client Application, that is no problem, though... the RtgAccept is only called at Game Startup...

d) How do i transfer data between Server and Client ?

=====

NOTE: The calls RtgRecv and RtgSend are not longer compatible to rtgmaster.library V6 or below ... sorry, for that... won't happen again... was because of the way how UDP works...

A. Transferring data with "Blocking" Sockets

NOTE : This is for TCP... if you are using UDP, you should be careful about some things described in the special chapter about UDP, additionally...

```
len=RtgSend(SocketBase,sock,buff,0,1024);
```

transfers 1024 Byte between this application and its connected application (anyway, who is the Server and who the Client...). buff is the buffer where the data (which is of the type ASCII String) is found :

```
char buff[1024];
```

sock is the Socket of THIS application.

In return we get, how many Bytes were in FACT transferred. If this is not exactly the same as

the number we want to transmit, we have to re-transmit the rest of it. If we currently can't send ANYTHING, the socket blocks and waits for better weather to send :) Normally, the application should send all the stuff at once... but "shit happens"...

NOTE: Better do not send packages larger than 1024 Bytes. Some networks do not want big packages. I know that 1024 Bytes work for sure, i do not know, where is the limit (test it out ? :))

```
len=RtgRecv(SocketBase,sock,buff,0,1024);
```

works the same, just for RECEIVING DATA.

This time, buff gets filled...

BTW... be sure, to have a buffer with enough Bytes like you specified :))

Some hints :

- Use a standard Package size, i suggest 1024.
- Let the Applications do a Send followed by a Recv each time... so that you do not confuse yourselves...

B. Transferring Data with "Nonblocking" Sockets

NOTE : This is for TCP... if you are using UDP, you should be careful about some things described in the special chapter about UDP, additionally...

```
do { } while(len=RtgSend(SocketBase,sock,buff,0,1024)<=0);
```

transfers 1024 Byte between this application and its connected application (anyway, who is the Server and who the Client...). buff is the buffer where the data (which is of the type ASCII String) is found :

```
char buff[1024];
```

sock is the Socket of THIS application.

In return we get, how many Bytes were in FACT transferred. If this is not exactly the same as the number we want to transmit, we have to re-transmit the rest of it. If we currently can't send ANYTHING, the RtgSend fails and tries again because of the do-loop. (Of course, you then could do something else, before re-entering the do-loop...

```
do { Something(); } while(len=RtgSend(SocketBase,sock,buff,0,1024)<=0);
```

or something like that...

You should use `<=0`, not `<0` or `=0`, to be on the sure side...

Normally the application should send all the stuff at once... but "shit happens"...

NOTE: Better do not send packages larger than 1024 Bytes. Some networks do not want big packages. I know that 1024 Bytes work for sure, i do not know, where is the limit (test it out ? :))

```
do { } while(len=RtgRecv(SocketBase,sock,buff,1024)<=0);
```

works the same, just for RECEIVING DATA.

This time, buff gets filled...

BTW... be sure, to have a buffer with enough Bytes like you specified :)

Some hints :

- Use a standard Package size, i suggest 1024.
- Let the Applications do a Send followed by a Recv each time... so that you do not confuse yourselves...

e) Now, whats the deal with that "One Server, Multiple Clients" ?

=====

Supported starting with rtgmaster.library V5...

- Open rtgmaster.library and bsdsocket.library as usual...
- Set up Server and Client(s) as usual...
- do the RtgIoctl as usual (but do NOT call RtgAccept !!!)

Some Arguments for both modes :

Blocking :

+ If nothing happens, the Server/Client does not consume Calculation Time

- While a Server/Client is waiting for a message you can't do anything else with it

Non-Blocking :

+ You can do other stuff, while the Client/Server is waiting, as it returns immediately

- Servers/Clients, which whom does not happen much currently, do BUSY-WAIT and this way consume a LOT of CPU Time...

- do a loop

```
while(1)
```

```
{
```

```
int maxsock;
```

```

struct RTG_Buff *in_buff;
struct RTG_Buff *out_buff;
...
Init_Output();
...
Calculate_Some_Stuff();
Put_Together_The_Output_To_The_Sockets(out_buffer);
Init_Input();
new=RunServer(SocketBase,sock,in_buffer,out_buffer,maxplayers);
if (new) put_new_socket_in_socket_list(new);
if (new) maxsock++;
HandleInput(in_buffer);
...
}

```

Explanation of that not-too-easy function (but i tell you, the Source of RunServer is even MORE complicated :))

Calculate_Some_Stuff() is a function of yourselves... here you calculate some stuff and other things, that your game might need...

Put_Together_The_Output_To_The_Sockets(out_buffer);
Init_Output sets the out_buff->num[] to -1. This has to be done before the first call of RunServer.

Init_Input() sets the in_buff->num[] to -1. This has to be done before EVERY call of RunServer.

Here you put together all messages that should be sent to the already connected Clients (1-12 Clients at one time are supported by RunServer).

RTG_Buff looks like :

```

struct RTG_Buff
{
char sock[12][1024];
int num[12];
int in_size;
int out_size;
};

```

In sock you can put up to 12 messages of a length of up to 1024 Bytes (you see, RunServer does not support bigger messages than 1024 Bytes...). In num you put the socket->s value (Surprise, surprise... it is a Integer... nothing than an

Integer behind our great Sockets...).

in_Size and out_size are the sizes of input and output

Packages (usual the same...). They should be NOT bigger than 1024...

Maxplayers is the Maximum of Sockets connected at a time to the Server.

Can't be bigger than 12.

put_new_socket_in_socket_list(new) is a function you have to do yourself, where you put up some sort of "Socket Database" of the currently connected Sockets. Remember, if RunServer returns something else than 0, it is a pointer to a RTG_Sock structure of a new connected Client...

After that the number of connected Sockets is raised, and then you can handle the Input. And again the loop...

BTW, if you want to disconnect a Client from your Server, simply call CloseClient inside the loop...

I hope, everybody understood everything, if not, send me a mail ... but it is not THAT difficult, i would say...

NOTE: Use this function with rtgmaster.library V6 at least.

The version of this function in V5 was VERY slow, as i had some problems in coding it. Now i got help after emailing the programmers of AmiTCP ... they showed me how to do it FAST :)

f) My Code still does not work

=====

It simply does not behave like it should

- Did you try to connect more than 12 Clients to a Server ?

(You should NOT do that...)

- Did you set in_buffer.in_size or out_buffer.out_size to a BIGGER value of the message actually transmitted ? (You should NOT do that !!!)

- Did you initialize in_buff->num[] before every call of RunServer and out_buff->num[] before the first call to -1 each ?

- Did you open SocketBase ? (You should do that...)

- Did you install AmiTCP 4.0 on your system ?

- You might deliver data too fast for AmiTCP, if you get messages from AmiTCP, that there would not be enough mbufs available. Try only to do more RtgSends, if all current ones are already handled (The Demo sources i did, do NOT look at this... they are only short hacks...

you only will get such problems, even if you transmit data VERY FAST, with 5+ connections at a time, BTW... a delay between RunServer calls will fix that for sure...)

- Did you call RtgAccept for a UDP connection ? That does not work...

I transmitted the number of a Client through a call of RtgRecv or through RunServer, but at the Server only ZERO arrived

- This is not possible. Simply do NOT transmit that number, but use in_buffer->num[] instead. That is just the number you want :)

My code just is too slow

- Try calling RtgIoctl, shortly after OpenServer/OpenClient.

But only call it ONCE, it is CPU intensive.

- Use 1024 Byte Buffers, nothing smaller.

This does not help

- TCP/IP with a *lots of Clients* is CPU intensive. RunServer just makes the best out of it. If it is still too slow, try timing it and only call it every 10th loop or something like that... if still too slow, use the faster "One Server-One Client" method... but, one note, One Server and up to 4 Clients should be okay...

g) Is there an easier way to support AmiTCP

=====

There is telser.device, that emulates a serial connection through TCP... it is NOT possible to do a "One Server - Multiple Clients" Method with telser.device, only Point-To-Point, also it might be slower than the direct support method... on the other side, to implement it, you just use serial.device code (just use telser.device instead of serial.device, and connect with atdtIP-Number or something like that...). telser.device is on Aminet... but as i said, maybe slower, and only 2-systems-connections, and additional, only support for Modem/Nullmodem-Connections, while rtgmaster.library supports ALL sorts of TCP/IP hardware...

h) Special Info about UDP

=====

UDP is the connectionless protocol of AmiTCP (yeah, now we do AmiUDP :)))). If you use it (you SHOULD use it for action games,

as TCP is a bit too slow for those... for strategy games, probably TCP is fast enough...)

What difference does it do for me, if i use UDP instead of TCP ?

Well, UDP is connectionless and unreliable. That is : If you send package A, then package B, then package C, you have NO guarantee that they will arrive in the same order... data package C might arrive first...

Also, sometimes a package gets lost... UDP does not do anything like errorchecking or "I want this Package again". It simply says "A Package lost? So what !!!" So if you want to be SURE, that a package arrived, check the return code of RtgSend and resend it, if something got wrong (or do it the UDP way : Throw the package away, as it is now old stuff anyways... will probably be true for action games anyways...)

Now : The most important thing to know about UDP, the other (minor) differences after it...

=====
Note : This is only for UDP ... this problem does not appear for TCP !!! (and even for UDP it only happens for very BAD connections...)
There is ABSOLUTELY NO guaranty that RtgSend REALLY will deliver a package of data. Because of a bad connection the package might be dropped, and to ensure fast transmission, the package is not resent, and there are no ACKs... for most data (that probably will come at a very fast rate for an action game...) it does not matter if one package comes or not, but if you have some data that HAS to arrive, here are possible methods to ensure this :

1. Method :

Use TCP. TCP resends packages that did not arrive (you can use TCP *and* UDP, if you want, TCP for some packages, UDP for others...)

2. Method :

1) Send the package

2) On the other side, send an ACK back ("yes, i received it"). Of course the ACK might be dropped, too... if there does not come an ACK in a certain time, resend the package.

3) if the ACK arrived, acknowledge the ACK. (Just for the case, that only the ACK got lost...). If the ACK is not ACK'd in a certain time, we will timeout. If we still receive old packages (because of ACK of an ACK being dropped...) simply tell the other side not to resend anymore with a similar mechanism.

This is exactly, what TCP does...

well, probably a TCP *and* a UDP channel at the same time is better...

=====

And now, after the MOST IMPORTANT difference, the minor stuff...

DO NOT CALL RtgAccept !!! RtgAccept is not needed for UDP (if it detects a UDP connection, it gives you an error code...)

You can use RunServer for multiple Clients - One Server, though... :)

Another NOTE : UDP gives you even more speed, if you send SMALL packages... the maximum size again should be 1024 Bytes... like

for TCP.. (i did not find in my docs for sure, if it is 1024 or 1500... both values were mentioned... but as it is one of them for sure, i took the save way (maybe one is the number for UDP, one for

TCP... anyways, with 1024 Bytes Max., it works... of course, you might use something like 64 Bytes for UDP or such for optimal speed...)

URGENT NOTE if you have problems understanding how UDP works :

- For TCP you always used the Socket of the Application to that you were CONNECTED to (for example RtgSend(Socket,...)).

- For UDP you always use your OWN socket for RtgSend !!!

If you have multiple sockets, you will have to use that new parameter of RtgSend/RtgRecv to say/find out to which socket you are speaking...

What changes in THE CODE ?

- If you have packages that are NOT ALLOWED to be dropped, you will have to send them with TCP, or you have to do some tricks with ACKs and Timeouts (see above...)

- SOCK_DGRAM instead of SOCK_STREAM

- RtgAccept is NOT called. You can transfer data, as soon as OpenServer and OpenClient are called.

- Multiple Clients are possible with only the basic scheme, without using RunServer (in fact, the UDP version of RunServer is not implemented up to now...)

- For receiving you do

```
struct sockaddr_in *si;
```

```
rval=RtgRecv(SocketBase,sock,buf,si,len);
```

instead of what was shown above.

After that you can find out the IP Address of the Sender with

```
char *ip;
```

```
ip=RtgInAdr(SocketBase,si);
```

The ip address will be in the form "194.55.101.26", for example...

With the help of this call, you might implement "One Server - Multiple Clients" through UDP. But Note: It can't differentiate multiple Clients running on the same machine. For those, it does broadcasting...

- For sending you do

```
struct sockaddr_in *si;
```

```
si=GetUDPName(SocketBase,si);
```

```
rval=RtgSend(SocketBase,sock,buf,si,len);
```

instead of what was shown above. You provide your IP Address for the receiver this way.

BTW, GetUDPName only works for UDP. For TCP it returns 0.

- You will have to check RtgRecv/RtgSend return codes FOR SURE, as packages might arrive in different orders or not at all. If

you get back, that not everything was received/sent, resend it !!!

An easier way for "One Server - Multiple Clients" will come later, as soon as I changed the RunServer call to work with UDP (up to now it does not work with UDP).

If you have still questions about UDP (well, it is a bit more complicated than TCP, check out the examples or write me a mail...)

1.14 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Welcome to the rtgmaster.library.

There are a lot of Graphics Systems on the Amiga. There is OCS. There is ECS, there is AGA. There are a lot of Amiga Graphics Board Standards, the most famous of them being CyberGraphX.

The problem of the Amiga in the last years was, that the hardware was ahead of the software, especially as to games. There was powerful hardware to be found, but nobody used it. The guys who created games and demos still stuck with AGA, the guys who created the Boards did not know anything about Games.

RtgMaster is a link between the different systems, but it is more than this.

RtgMaster is a system specifically dedicated to game/demo-coding on GFX Boards. The System includes high-speed GFX Board Support, and also partially ECS/AGA Backward-Compatibility. The system runs parallel with any existing WB Emulation. So you can use it, anyways, if you have installed CyberGraphX, Picasso96, EGS or what else. And it is even more. For developers there exists special support,

so that they can order a rtgmaster driver or rtgmaster adaption of their game, if they do not want to bother with the GFX Board Coding part themselves.

What does this mean for you, the user ? Simply install the rtgmaster libraries and enjoy playing the next generation of Amiga Games. To install it, simply run the installer script. This will install rtgmaster.library to your system, also some sublibraries, for the different display systems.

The following projects support rtgmaster (some of them are still unfinished).

Note, that not all of them support the Original Picasso II Software, as the rtgmaster driver for this WB Emulation is not fully done. This is due to limitations of the Picasso II Software. Under CyberGraphX, a Picasso II works fine, though.

Before you start i suggest you read the sections about supported software and about Amiga GFX Boards, to find out, if your favourite game has a rtgmaster driver (probably no, up to now) and if your GFX Board is supported (mostly yes).

1.15 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Name Firm or Group Genre

=====

Genetic Species Vulcan Software DOOM-Clone

Genetic Species is a very fast-running DOOM-Clone that supports GFX Boards through rtgmaster.library and AGA through native AGA code. There is an old demo on Aminet, that does not yet support GFX Boards. A new Demo should soon appear.

For more information about this game, i suggest having a look at the new Screenshots on the homepage of Vulcan Software.

3D Games Creator Vulcan Software AB3D II Clone

Cold Blood is a yet to appear DOOM-style game that will blow AB3DII away.

It supports GFX Boards through rtgmaster, AGA through native AGA code.

Note: The game will not be released, but a 3D Construction Kit based on the Engine will be released.

WheelsOnFire Proximity 3D Racing Game

Up to now rtgmaster support of this game is not yet done, but i am in talks with the authors. A Demo of this game is on Aminet (only for AGA).

The authors currently enhance the 3D Voxel Engine to a higher resolution, and in this process probably rtgmaster-Support will be added, to support GFX Boards. AGA will as always be done through native AGA code.

Note: The coder now decided not to use rtgmaster, but he decided to use

rtgmaster in his next project.

Phoenix ??? Privateer Clone

GFX Board Support for Phoenix is still being considered, but if it arrives, it will probably be through rtgmaster. AGA Support is native AGA, as always. Privateer is a PC Space Game similar to Wing Commander. Phoenix is a Clone of it. The game has still to arrive. After, what i heard, end of the year...

AB3D II peabrain@usa.net AB3D II GFX Board Port

Well, just that... :) Will be released as public domain, but you will need a full version of the game (commercial) to use the new executable.

Currently in Beta-Testing.

Boulderdaesh geit@DVZ-102.fh-muenster.de Boulderdash Game

Well, just that :)

There are some more projects in progress, mostly Shareware stuff. It even might be possible, that there will appear a rtgmaster driver for one of the AmigaDOS Clones AROS or p-OS in the future, but this is yet to be decided.

PowerPC-enhanced versions of rtgmaster are already planned.

1.16 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

To get information about supported WB Emulations please click on the links.

[CyberGraphX](#)

[CyberGraphX 3](#)

[Probench 3](#)

[EGS](#)

[Picasso II Native](#)

[Picasso96](#)

[Retina WB Emu](#)

[Graffiti](#)

[ECS/AGA](#)

[Other hardware](#)

1.17 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

CyberGraphX 2

CyberGraphX 2 is probably the most used WB Emulation for Amiga GFX Boards.

It is one of the best WB Emulations out there, but it lacks specific support

for games. rtgmaster fills that gap. rtgmaster runs together with all CyberGraphX Boards that run NON-SEGMENTED. You might wonder what that limitation is about. The Commodore A2410 Board for example is a segmented Board. Segmented Boards lack the feature of Direct Access. Most Game Authors use Direct Access in their Games, so supporting those Boards would make no sense. Direct Access is much faster than all other methods, anyways. It might be possible that the Domino Board is also segmented. I do not know this, so i can't tell you, if rtgmaster supports the Domino or not. All other CyberGraphX Boards i know of are non-segmented. CyberGraphX 2 Support needs libs:rtgmaster.library, libs:rtg/rtgCGX.library and your CyberGraphX 2 Software installed. Installation is done by the Installers of RtgMaster and of CyberGraphX.

1.18 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

CyberGraphX 3

CyberGraphX 3 is the most recent version of CyberGraphX, but up to now it does not support any other boards than the Cybervision 64 and the Cybervision/3D.

RtgMaster always supports all NON-SEGMENTED CyberGraphX 3 Boards.

It is one of the best WB Emulations out there, but it lacks specific support for games. rtgmaster fills that gap.

You might wonder what that limitation about NON-SEGMENTED is about.

The Commodore A2410 Board for example is a segmented Board. Segmented Boards lack the feature of Direct Access. Most Game Authors use Direct Access in their Games, so supporting those Boards would make no sense. Direct Access is much faster than all other methods, anyways.

CyberGraphX 3 Support needs libs:rtgmaster.library, libs:rtg/rtgCGX.library and your CyberGraphX 3 Software installed. Installation is done by the Installers of RtgMaster and of CyberGraphX.

1.19 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Probench 3

Probench is probably one of the first WB Emulations that exist at all. It is only used by the Boards Domino, Merlin and Merlin II, so it is quite an exote. Starting with Probench 3, there is some sort of partially-working CyberGraphX Emulation.

RtgMaster runs together with Probench 3, if you have at least the Upgrade Package 7 running. Earlier versions might work or not, but probably not. I do not know, if

rtgmaster runs together with the Domino, as the Domino might still be a segmented GFX Board. Segmented Boards are old technology, that is not compatible with the way most games are coded. Because of that rtgmaster does not support them. If the Domino runs in Non-Segmented Mode, rtgmaster will support it, of course.

rtgmaster does not run with Probench 2 or earlier.

Probench 3 Support needs libs:rtgmaster.library, libs:rtg/rtgCGX.library and your Probench 3 Software installed. Installation is done by the installers of RtgMaster and of Probench.

1.20 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

EGS

EGS was a quite popular WB Emulation at its time. To tell the truth, the whole rtgmaster project started with EGS. rtgmaster needs at least EGS System 6, the most current version done by Viona Development is EGS System 7. EGS was dropped by Viona later, and someone else took it up, releasing a new version called EGS Plus. I never really tested rtgmaster with EGS Plus. I do not know if it works.

If someone knows, tell me, i do no longer own a EGS Board.

rtgmaster.library supports the following EGS Boards (not all of them tested, some of them might be segmented Boards, and rtgmaster never works with segmented Boards) :

Piccolo

Piccolo SD64

EGS Spectrum

Rainbow 3

Rainbow 2

EGS 110 from GVP

bsc Graffiti (do not confuse with Graffiti, this is something else)

You might wonder, why rtgmaster does not support segmented Boards like the A2410.

This is because those Boards lack a special feature needed for game programming.

Segmented Boards are only the very old boards, anyways.

I heard somewhere, that the old Board Visiona also used EGS. rtgmaster might run on it or not. If someone still uses this Board, please contact me.

Under EGS it is possible, that the mousepointer of an rtgmaster application starts looking strange. Don't worry. This is normal, because of an unfinished part in the driver. Asides from that, rtgmaster runs fine under EGS.

EGS Support needs libs:rtgmaster.library, libs:rtg/rtgEGS.library and your EGS Software insralled. Installation is done by the Installers of RtgMaster and EGS.

1.21 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Picasso II Native WB Emulation

The Picasso II Native WB Emulation is one of the first WB Emulations. It is only available on the Picasso II and the Picasso II+ Boards. There is limited rtgmaster Support for this WB Emulation, limited only, because this WB Emulation does not support all rtgmaster features. Especially the Keyboard-Functions of rtgmaster are not supported. So probably most Software won't support this sublibrary.

Please consider updating to CyberGraphX or Picasso96. These WB Emulations have full rtgmaster Support.

Picasso II Native Support needs libs:rtgmaster.library, libs:rtg/rtgPICA.library and your Picasso II/II+ Native Software installed. Installation is done by the installers of RtgMaster and of the Picasso Software.

Note: The rtgPICA.library is discontinued, as the Picasso WB Emulation just is not powerful, to support some of the later features, and also rtgPICA.library seemed to run instable.

1.22 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Picasso96

Picasso96 is a new WB Emulation standard. With the exception of the Cybervision/3D nearly all Boards have a Picasso96 Driver. There is a CV/3D Driver in developpement currently.

Until just recently, Picasso96 did not have a certain feature called "DoubleBuffering".

If you have the latest Picasso96 installed, use rtgmaster.library and libs:rtg/rtgP96.library.

If you do not have the latest version, use rtgCGX.library instead of rtgP96.library.

The difference between the two (as far as Picasso96 is concerned) is that rtgP96.library supports Doublebuffering for Picasso96, if you have the latest Picasso96 Version installed.

Of course, you also need Picasso96 :)

Installation is done by the installers of rtgmaster and Picasso96. If you do not have the latest version of Picasso96, you have to replace rtgP96.library manually by rtgCGX.library.

But it is REALLY recommended to get the latest version. There might appear games, that NEED Doublebuffering.

1.23 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

The Retina WB Emulation

The Retina WB Emulation was used by the Boards Retina BLT Z3, Retina Z2 and by the DraCo. There are now CyberGraphX Drivers for Retina BLT Z3 and the DraCo, though. Retina WB Emulation is not supported by rtgmaster. If someone wants to do support for it, i could give him developer docs to support it, though. (A new rtgmaster sublibrary had to be created, this takes about 1 week of coding, maybe less. Myself i can't do it, as i do not have a Retina that would be needed for testing). rtgmaster in no way supports the Retina Z2, as the Retina Z2 is a segmented Board. Segmented Boards miss some features needed for Game programming. They are quite old boards, anyways.

1.24 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

The Graffiti

There is not yet Graffiti Support for rtgmaster. It is planned, though. My problem is, i do not have a Graffiti, and it would also not run on my system, as i do not have a 15 kHz monitor (i use 15 kHz modes through a internal flickerfixer). If someone wants to help with the Graffiti Support, please contact me.

1.25 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Other Hardware

There are some other very old Boards. For example the Opalvision or the Firecracker. RtgMaster does not support them. Most of these Boards not even have a real WB Emulation, only some special drivers for some programs. A RtgMaster Driver might be possible for some of those Boards, so if you have such a Board and would be interesting in doing a driver, please contact me. Most of those Boards probably are segmented Boards, though, and segmented Boards miss some feature needed for fast Amiga GFX Board Games. In the future there will be some new Boards not yet listed here, that WILL be supported. There will be a special version of the Picasso IV that will run through PCI on a special Turbo-Board that ships with a PCI slot. This is one of the next Boards that will be supported. As this Board probably uses Picasso96, there won't be a problem, probably. Also, after Phase 5 announced a CAIPIRINHA Board for the Cyberstorm PPC, there might be CAIPIRINHA Support for rtgmaster in the future. But as i do not know much about CAIPIRINHA yet, i can't tell.

1.26 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

ECS/AGA

rtgmaster supports also ECS/AGA. The problem is only, that the fastest possible function of rtgmaster, CopyRtgPixelFormat, won't run on ECS/AGA, as it assumes a chunky display. Only the slower function CallRtgC2P will support it. But with some minutes work, a rtgmaster program runs on ECS/AGA with no problems. Most rtgmaster programs feature native AGA Support, that is probably faster, though. About the ECS Support there is a special problem. If a rtgmaster program uses 256 colors, it principally runs on ECS, but 192 of the colors are simply ignored and get Black. So the result might look strange on ECS.

ECS/AGA support needs libs:rtgmaster.library and libs:rtg/rtgAMI.library (installed by the rtgmaster installation script). Also it needs a monitor driver like PAL or NTSC in the devs:monitors directory. Without this you won't get any ECS/AGA Screenmodes.

1.27 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

To use rtgmaster.library, simply start an executable using rtgmaster.library.

You can then choose a Screenmode. If you save it, it will be saved to the local Directory, and the next time you start a rtgmaster program from this directory, the Screenmode-Requester will only popup, if you press SHIFT while starting the program. If you have any problems using rtgmaster, please check out the section about Amiga GFX Boards, to check out, if your GFX Board Software is installed correctly.

1.28 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Benchmarks and Demos

Provided demos are in the Demos Directory.

These are :

flame

Flame is a Plasma effect. It runs on all Systems. It is coded nearly completely in C (the main-Plasma-Loop is in ASM).

flamme

Flamme is another Plasma Effect. It runs on all Systems, but it has to be started from the Shell. It does not work from Workbench. It is coded in ASM.

Mandel

Mandel is a Mandelbrot set. It runs on GFX Board Systems. It is coded in C.

Moon

Moon is a Voxel-Graphics landscape. It runs only on systems with FPU.

It needs a Screenmode of 320x200 or 320x240 to run. It is coded nearly completely in C, so do not expect it to be fast.

Mywolf

Mywolf is a Wolf3D style texturemapping engine. It runs on all Systems.

It is coded completely in C. The only current rtgmaster c2p with that this demo runs is GD. As it is the only c2p, that currently supports c2p'ing "not-Full-Screen".

There are some more demos in the developer archive. But those are mostly not that impressive.

Benchmarks :

Flame

Flamme

Moon

Mywolf

1.29 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Benchmarks of Flame

Machine GFX System 320x200 160x120

```
=====
A4000/030+Cybervision CyberGraphX 24 fps 30 fps
A4000/030 AGA 10 fps 10 fps
A2000/030+Picasso II Picasso WB-Emu 15 fps 24 fps
A4000/040+PiccoloSD64 EGS 34 fps 63 fps
A4000/040+PiccoloSD64 CyberGraphX 34 fps 63 fps
A4000/040 AGA (020opt.c2p) 14 fps 16 fps
A4000/060+Cybervision CyberGraphX 69 fps 136 fps
A4000/060+Retina Z3 CyberGraphX 50 fps 118 fps
A4000/060 AGA 29 fps 31 fps
A4000/040 AGA (040opt.c2p) 18 fps 22 fps
A3000/030+Piccolo EGS 22 fps 30 fps
A3000/030+Piccolo CyberGraphX 22 fps 30 fps
A4000/040+Cybervision CyberGraphX 35 fps 77 fps
A2000/040+Picasso(33MHz) CyberGraphX 19 fps 50 fps
A2000/030+SD64 CyberGraphX 17 fps 26 fps
```

A2000/060+SD64 CyberGraphX 26 fps 77 fps
 A4000/040/40MHz+Cyberv. CyberGraphX 60 fps 105 fps
 68030 25 MHz+RetinaZ3 CyberGraphX 21 fps
 A1200/030 50 MHz AGA 11 fps 19 fps
 A2000/030 50 MHz+Picasso Picasso WB Emu 21 fps
 A4000/040 25 MHz+Ret.Z3 CyberGraphX 29 fps
 A4000/040 40 MHz+Ret.Z3 CyberGraphX 34 fps 90 fps
 A3000T 25 MHz + SD64 CyberGraphX 22 fps
 A4000/030 25 MHz + Merlin II Probench3.0 21 fps
 A4000/060 50 MHz + Merlin II Probench3.0 51 fps
 A3000/040 25 MHz ECS 64 colors 19 fps 23 fps
 A4000/040 40 MHz+CV/3D CyberGraphX 3 37 fps 80 fps
 A2000/030 25 MHz Domino Picasso96 18 fps
 Note: These are values of the old Flame version, the new one is much
 faster on GFX Boards (using a Fastram Buffer).

1.30 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Benchmarks of Flamme

Machine GFX System 320x200

```

=====
A4000/040 40 Mhz CV/3D CyberGraphX 3 74 fps
A4000/040 40 MHz SD64 CyberGraphX 102 fps
A4000/040 40 MHz AGA 23 fps
A4000/040 40 MHz AGA (040opt. c2p) 40 fps
A2000/030 25 MHz Domino Picasso96 23 fps
  
```

1.31 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Benchmarks of Moon

Machine GFX System 320x200

```

=====
A4000/040 40 Mhz CV/3D CyberGraphX 3 8 fps
A4000/040 40 MHz AGA 6 fps
A4000/040 40 MHz AGA (040opt. c2p) 8 fps
A2000/030 25 MHz Domino Picasso96 1 fps
  
```

1.32 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Benchmarks of Mywolf

Machine GFX System 240x200

=====

A4000/040 40 Mhz CV/3D CyberGraphX 3 36 fps

A4000/040 40 MHz AGA 21 fps

A2000/030 25 MHz Domino Picasso96 9 fps

1.33 Rtgmaster.library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

The History of rtgmaster.library

Version Changes

V0.1 First Alpha release (that one with the famous

Screenmode requester that contained more Bugs

than functions :))

V1.0 First Beta Version with the new Screenmode

requester. Only worked with rtgCGX.library.

V2.0 First official Aminet release of rtgmaster.library

Bug in V1.0 about it only loading rtgCGX.library

is fixed in V2.0, many bugfixes, new implemented

stuff and complete rework of the Autodocs !!!

c2p Support, new Screenmodereqeuster...

V3.0 Some Bugs fixed...

V4.0 First version with TCP/IP Support

V5.0 First version with TCP/IP Support with

"One Server - Multiple Clients"

V6.0 A Major Speed enhancement for the function

RunServer. Do not use V5.0 for TCP/IP, use

V6.0... it is really MUCH faster (i threwed

out a LOT of Forbid()/Permit() Stuff, as the

programmers of AmiTCP explained me by email,

how to do this stuff without a single Forbid() :)

V7.0 First version with UDP Support

RunServer does not run with UDP yet, though...

V8.0 UDP Support finished

V9.0 Finally not-fixed-resolution c2p works (gd

and gdecx c2p are provided), due to a bugfix.

V10.0 Text/Font Support added, "Automatic ECS Support" added (every AGA c2p supports ECS, too...)

V11.0 Expanded features of CopyRtgPixelFormat

V11 did not run on some systems with Cybergraphics.library

V40.92 ... V12.0 fixed that, though...

V12.0 Added RDCMP for Input Handling

V13.0 Fixed "RtgBlit will crash under EGS with Minterm

0xc0 and will do strange things when blitting to

Buffer 2" Bug

Removed d6/d7 Parameters of CopyRtgPixelFormat

(this time it is final !!!), Recompiling/Assembling needed, sorry...

Fixed Installer Script and some minor Bugs

OffX/OffY parameters of CopyRtgPixelFormat

implemented for rtgCGX.library and rtgEGS.library.

Sadly, rtgPICA.library still crashes for unimplemented calls (for example RDCMP is not yet implemented for

Picasso II, only for the other three sublibs, but

the rtgPICA.library is currently in question anyways,

as some stuff is NOT POSSIBLE using that WB Emulation !!!)

BTW, does someone know how to access the VGA Registers

of some boards (information about SD64 and CV64 !!!) ?

I really would like a Mode X Support to rtgmaster, if

this is possible...

V14.0 Fixed the "Crash on some system with smr_PlanarSupport or smr_ProgramUsesC2P =0" bug

Fixed the Screen Depth Bug with 16 Bit Piccolo SD64 Screens

libs:rtgc2p is now default for c2p algorithms

SrcWidth/SrcHeight of CopyRtgPixelFormat removed again, and

THIS TIME the function CopyRtgPixelFormat will stay as it is

As many people use rtgmaster to convert some PC stuff to Amiga,

i included the asmconv program, that converts Intel ASM to

Motorola ASM.

V15.0 Changed rtgCGX.library, so that it runs on both CyberGraphX 2 and

CyberGraphX 3 (for Cybervision/3D Support). Still, rtgmaster.library

won't run with cgxsystem.library versions smaller than 41.1 (41.0 had

a serious bug in the LockBitmapTagList call). rtgPICA.library now no

longer crashes, if a pointer, text or RDCMP function was called. It does

not do anything, nevertheless. rtgCGX.library MIGHT run with Picasso96 Software, maybe not, i do not know, up to now, but will test, as soon as possible (maybe someone can tell me ?) Removed rtggadgets sources, added RtgGadTools.library (which replaces the rtggadgets sources). V16.0 Added flickerfree DBuffering to rtgCGX.library (only does its job for CyberGraphX 3, for CyberGraphX 2 it is ignored, as CGX 2 can't REALLY do Flickerfree buffering currently). Added support for mp_sigbit to all sublibs with exception of rtgPICA.library (yet to do), added gamecoderhelp for game coders in goodies directory.

V17.0 Fixed bug concerning new V16.0 features of RtgInitRDCMP Updated Autodocs&Includes. Best: Do not use V16.0 anymore, as programs might use the new feature.

V20.0 Complete rework of the Package to make it more professional, split into User, Driver and Developer Archive.

V21.0 Update of rtgCGX.library (bug in 15/16/24 Bit modes fixed) Completely new Installer script, as it seems that the "Installer Example" lha from some BBS that i used to do the first installer scripts used copyrighted material. I removed the old installer scripts completely. Thanks to Robert C. Reiswig for reporting. (I also offered him to alternatively overtake the development of the scripts, but he did not answer to my mail, so i had to think of something new... so i took a whole afternoon to rewrite the scripts from the beginning... BTW, these new Scripts now finally WORK !!!)

V22.0 Some internal changes, added section about Cybervision 64 Problem and how to fix it, added workaround for CyberGraphX 3 R 55 Bug (Note: Doublebuffering internal implementation in rtgmaster might change again in the future).

Added sources for moon demo, as the author allowed me to include them now.

V23.0 Under OS3.0 SwitchScreens will now run 2000% faster than with the old version (only of ECS/AGA sublibrary). Also, there are some bugs fixed, especially the bugs concerning rtgmaster and MCP. The fixed version of rtgEGS.library is still not tested, so anyone who still has installed EGS, please test and email me, if it runs. rtgPICA.library still does not work with MCP, but as nearly nobody uses this sublibrary anymore, anyways, i moved this sublibrary to the Goodies directory.

Install it on your own risk. Support for BlitzBasic was added.

V24.0 Added RtgAllocSRLList, RtgFreeSRLList, RtgBestSR, CopyRtgBlit, RtgCheckVSync, fixed some enforcer hits, library now also runs, if setpatch or devs:monitors-drivers are not installed, moved TCP/IP Support to rtgextra.library (they are not any longer inside

rtgmaster.library itself), removed support for segmented Boards (GetSegment/SetSegment) which was never really in, anyways, added already partially working CopyRtgPixelFormat Planar Copy for AGA, added Autodocs for upcoming rtgmaster 3D Extensions (currently in Beta testing), removed the BlitzBasic stuff, as experience brought it, that (as it does not support full rtgmaster functions, anyways) i always had to change some things in the code of a special game to make it running, as every coder needed special functions of rtgmaster. Also it had bugs.

If you need BlitzBasic support, please contact me for it.

Updated Includes and Docs. Changed Flame Demo to use a Fastram Buffer (under GFX Board it is now much faster). Added tagcall-version of certain rtgmaster-calls.

Fixed a serious bug in CopyRtgPixelFormat in 15/16/24 Bit Modes.

Added Amigaguide Version of Autodocs (Note: They not yet represent the V24 changes, they are still V23 !!!) done by Rune Espeseth and James McArthur. Thanks to them for doing it :)

V25.0 Internal Version (also given to some developers). Mainly fixes a serious bug in the init code of some of the rtgmaster library. Actually this bug was in since the very beginning of rtgmaster. I did not notice, as the bug only appears on about 5% of Amigas, on all other Amigas (including mine) it works just fine. Well, it is now fixed.

V26.0 Added first version of E Includes of bhar@hem.passagen.se, added a sublibrary that works native on Picasso96 (of course rtgCGX.library works with Picasso96 too, but it does not have Doublebuffering. rtgP96.library also supports Doublebuffering, in fact a faster Doublebuffering than provided by rtgCGX.library).

Fixed some "not so important" functions that were broken in rtgAMI.library since some versions (FillRtgRect and some others). Note that rtgP96.library needs a quite up-to-date version of Picasso96, as Doublebuffering was only just recently implemented into Picasso96. If you do not have yet this version, use rtgCGX.library instead. Only install one of the two (rtgP96.library or rtgCGX.library), else you might get all Screenmodes twice, and Screenmode Save might do unexpected things. I also fixed the "PAL Modes appear twice" Bug, and also now a .info file in libs:rtg does not cause rtgmaster to crash anymore :)

V27.0 Finally fixed it, so that it runs on ALL AGA-only systems, also. Added info about past AGA-only problems (mostly the problem is a incorrect startup-sequence or a missing monitor-driver). Added info about upcoming rtgmaster with additional PPC Support (of course it also runs on a 68k

Amiga !!! Fat Binary !!!)

V32.0 Added PPC Support, WB Window Support, Softsprite Functions,... Docs still uncomplete (Work in Progress), read Prerelease of the Docs-Addition ppcdocs.txt
Definitely have a look at ppcdocs.txt and devdocs/Spritexamp.lha, there is some stuff not yet properly documented (About the docs: Work in Progress) to be found there.

1.34 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

Authors and Copyright

Some basic notes :

the original rtg.library was developed on an idea of John Hendrikx. He later gave developpement to me and i made the rtgmaster.library out of it. Most work (main library, rtgCGX.library, rtgEGS.library, rtgAMI.library, Docs) was done by myself, Steffen P. Haeuser.

- The rtgPICA.library and the rtggad.library were done by Hans-Joerg Frieden
- Some of the examples were done by Hans-Joerg&Thomas Frieden
- The Moon Example was done by Olaf Asholz
- The Screenmode Requester was done by Wolfram Schenk
- The 680x0 to 860x0 ASM converter was done by Nikolaus Mausz

The original installer scripts were done based on some "Installer Examples" found on different BBS systems. I was notified some weeks ago, that those "Examples" in fact used copyrighted work of Robert C. Reiswig. I offered Robert to overtake the developpement of the rtgmaster installer script and to notify him in the docs. As he did not answer to my mail after over a week, i simply rewrote the installer scripts without using those infamous examples. If i had known at the beginning, that those "Examples" used Copyrighted material, i would never have used them. The Uploader of the Examples Archive did not say anything about that.

User Information

rtgmaster.library is done by Steffen P. Häuser (that's me). Some of the sublibraries are done by other persons. I take no guarantee that it works on your system. Also i don't take guarantee that it not damages your system (don't be upset, this is only a standard disclaimer ... of course this software should work on your system, and it won't damage it... but to keep trouble away i included this standard disclaimer.

Basic Developer License

Developpers can include any part of this archive to their programs, as long as the part they include is still functional. They should read the stuff below, and do like it says. Principially, rtgmaster.library is for free, but i would

appreciate some good will to give me at least a free copy of the program using it.

Exact informations for different sorts of software are listed below.

Freeware/PD Developpers

I would appreciate being mentioned in the Credits or Docs of your program.

Asides from that you can use rtgmaster.library for free.

Shareware Developer

I would appreciate being mentioned in the Credits or Docs of your program.

I would appreciate a free copy of your program, if this is ok for you.

Asides from that you can use rtgmaster.library for free.

Commercial Developer

I would appreciate being mentioned in the Credits/Docs and a free version of your program (if this is OK for you/your publisher). I also would appreciate some extra-bucks from your publisher, if this is OK for him. If it is not okay for him, a free game might also be enough. Give my email to your publisher, so that i can clear things up with him. My Email is MagicSN@Birdland.es.bawue.de.

Developer - Extra Support

If you are a Developer of a Commercial Game, want it to run on a GFX Board, but do not want to have to bother with writing the GFX Board Part of the Code yourselves, i can do this for you. Simply mail your specifications for the needed functions to MagicSN@Birdland.es.bawue.de and i will do the job for you. In this case, we should definitely speak about a free Game and some extra Bucks. Maybe \$100 or something like that. But well, this is open to discussion. But think about it, \$100 definitely will come in with the sales for the GFX Board Version. Of course will be cheaper for Shareware programs.

Well, if your publisher does not want to pay me, i probably will make the GFX Board Version anyways, but then he is a skinflint !!!

Additional note about Copyright :

rtgPICA.library and some of the examples are done by Hans-Joerg Frieden, Moon Demo was done by Olaf Asholz, rtggadtools.library was done by Hans-Joerg Frieden. The 86x86 to 680x0 converter was done by Nikolaus Mausz. The Screenmode-Requester was done by Wolfram Schenk.

Those people gave their okay to include their work to my rtgmaster.library Package. All the rest (main library, the other three sublibraries, and the rest as to demos, Docs...) was done by me.

In case you need my address :

Steffen P. Haeuser

Limburgstr. 127

73265 Dettingen/Teck

Germany

Tel. 07021-51787

Co-Sysop of Birdland BBS : 07021-861920/862428/862429 and other numbers

MagicSN@Birdland.es.bawue.de

1.35 RtgMaster Library V32.0

The RtgMaster Library for GFX Board Using Demos/Games

c2p Algorithms

The rtgmaster c2p format is currently not optimal. It probably will change in the future, but that should not bother the user. It is only to the user to know, that the ECS/AGA Support currently is not speed-optimal, because of this problems. But luckily most rtgmaster-using programs also support native AGA Support.

Currently there are this c2p :

Suzy: The fastest rtgmaster c2p. It is optimized for 030, but on 060 it even outperforms the 040 c2p. It only supports Fullscreen c2p.

Games/Demos that do Smaller-Than-Fullscreen-c2p won't run with this c2p.

ecs: As fast as Suzy, but for ECS. It is a true ECS c2p, it does not leave higher colors out, it c2ps to 64 colors. The program still needs an (optional) 64 Color Color-Table to look good, of course.

Supports Smaller-than-Fullscreen and Y Destination Offset.

Leony: This is a 2x1 Pixelsize c2p. It doubles horizontally. You need programs specially coded for 2x1 to be able to use this c2p (try the flamme demo...). Supports Smaller-than-Fullscreen and X and Y Destination Offset.

040 : 040 optimized, needs a 040/060 to run. Only runs in 320x200.

On some 060 Boards this one does not run.

GD : 020 optimized. Runs in all resolution. Currently the only rtgmaster c2p that supports c2p'ing stuff smaller than Fullscreen.

Chunky4 : 030 optimized. On 040 and above, this c2p is the slowest one, only supports 320x200.

Suzy, Ecs and Leony flicker, if they c2p to the front screen without Doublebuffering. As they are the fastest c2p available for rtgmaster, c2p should ALWAYS be coded to c2p to the Backscreen and perform Doublebuffering after this. (More info in the Developer Docs...)

Note: These are 68k c2p. Currently, only one c2p for the PPC exists, a direct port of a c2p very similar to 040, and it is always used on PPC-side. The PPC c2p only supports Fullscreen-c2p, but all resolutions.