rtgmaster

COLLABORATORS

	TITLE : rtgmaster	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 rtgmaster **1**

1.1 Rtgmaster.library V13.0 . . . 1
1.2 Rtgmaster.library V13.0 . . . 1
1.3 Rtgmaster.library V13.0 . . . 3
1.4 Rtgmaster.library V13.0 . . . 4
1.5 Rtgmaster.library V13.0 . . . 4
1.6 Rtgmaster.library V13.0 . . . 4
1.7 Rtgmaster.library V13.0 . . . 5
1.8 Rtgmaster.library V13.0 . . . 7
1.9 Rtgmaster.library V13.0 . . . 7
1.10 Rtgmaster.library V13.0 . . . 9
1.11 Rtgmaster.library V13.0 . . . 11
1.12 Rtgmaster.library V13.0 . . . 12
1.13 Rtgmaster.library V13.0 . . . 13
1.14 Rtgmaster.library V13.0 . . . 14
1.15 Rtgmaster.library V13.0 . . . 14

Chapter 1

rtgmaster

1.1 Rtgmaster.library V13.0

The RtgMaster Library - EASY GFX BOARD ADAPTION FOR GAMES AND DEMOS

Table of contents

Part A : User Documentation

Part B : Developper Documentation

Part C : Buglist

Part D : Benchmarks

Part E : History

Part F : Authors and Copyright

Part G : TCP/IP for Newcomers

1.2 Rtgmaster.library V13.0

The RtgMaster Library - EASY GFX BOARD ADAPTION FOR GAMES AND DEMOS

1. Purpose

==========

Sometimes it is a bit difficult to code an Amiga games that supports all systems,
ECS, AGA and all sorts of Graphics Boards. Rtgmaster Library is a system that
makes it easier for programmers to support every existing Amiga Graphics Hardware.
You, as user, simply will have to run the Installer Script that is provided with
this package, to run software using rtgmaster.library.
Also, not many developpers for games know how to code TCP/IP Support, especially
on Amiga (TCP/IP is nearly the same for Amiga, UNIX and OS/2, btw... only TCP for
Windows/Windows95 is different... Bill Gates never could stand standards by
OTHER people than himself...). The functions provided by rtgmaster.library
are NO standard TCP/IP functions, though, they are a much
simplyfied version especially intended for game developpement.

Also, contrary to the standard, TCP/IP Support is now possible in ASM,
and not only in C... and you do not need the AmiTCP Includes to do it.
Only the Rtgmaster Includes. TCP/IP Support is, contrary to the rest
of rtgmaster.library, coded in C, as i had to use the AmiTCP stuff,
that only supports interfaces for C.
The TCP/IP Stuff runs for sure with the AmiTCP 4.0 Demo version of
Aminet, i do not know, if it runs with older versions, too. I doubt
that it will run with the Amiga Technologies Internet Package, and
i can’t fix that as i do not know where to get Includes for it from.

2. Usage

========

After you have installed rtgmaster.library, simply run a program using it (try
the demo "flame" that is provided with this Package). You will be able to choose
a Screenmode from a Screenmoderequester. This need not be ALL Screenmodes available
on your system, as the programmer of the game/demo might have filtered out only
the interesting Screenmodes. If rtgmaster.library says that it did not find any
fitting Screenmodes, you might have to create a fitting Screenmode with your
Screenmode-Utility (cvmode for CyberGraphX for example...). On the right side of
the Screenmode, you will see some information on the selected Screenmode.
You are able to choose a c2p module out of libs:rtgc2p and a preferred c2p mode.
And you are able to get some information about the c2p algorithms that are installed.
You should use the algorithm that is best optimized for your system. With this package,
there are some c2p algorithms provided :

- One 040 optimized cpu-only 320x200 resolution only, AGA+ECS ("040")
- a cpu-only 020 optimized c2p for ALL resolutions, AGA+ECS ("GD")
GD up to now is the only rtgmaster c2p that can handle smaller
than Fullscreen resolutions (but the GFX has still to start at
(0,0), Offsets are not yet supported... Y Offset might work,
but X Offset won’t...
- Two more 020 optimized c2p.
- Some more or less working ports of other c2p of Aminet and other sources
the c2p’s in c2p-garbage can’t handle yet ECS, the ones in strange_c2p
can do, but are buggy...

(Sorry, i am no c2p coder... people who want to convert their own c2p
algortihms to the format of my library, are invited to do so and to email
the algorithms to me... i need some more algorithms, especially ECS ones
and ones that are not fixed-resolution-ones... and asynchronce c2p algorithms)
If you have a fast system you should use 1x1 c2p Mode, if you have a slower system
you should use a lower resolution one. Some c2p algorithms only support 1x1.

Then there are the buttons SAVE, USE, CANCEL, that work like expected. If you
already saved a Screenmode/c2p algorithm, the Screenmoderequester won't appear
on the next start of the program for which you chose a Screenmode. It will appear
again, if you press SHIFT while starting the program.
Additional there are some menus providing an "About" and a possibility to load/save
the configuration of the rtgmaster-using-program that you started just recently.
And the possibility to switch back to the last selection.
Rtgmaster.library runs starting with OS2.0 and 68020. Starting with OS2.1 it supports
localization, but you do not need OS2.1 to get rtgmaster.library running. You can use
the cursor keys to choose a Screenmode and s,u,c for SAVE, USE, CANCEL, too...
3. Other Stuff
==========
- Do NOT use Screenmodes that might damage your monitor. rtgmaster.library does not
check if a Screenmode is OK for your monitor. You better should not install such
Screenmodes AT ALL.
- If a program needs 2 MB Video RAM, but your graphics Board only has 1 MB, rtgmaster.library
might give you a software failure.
- Do not use the utility DosWedge. It causes rtgmaster.library to bring you a Software Failure.
- Screenmodes whose resolution and name differ (a 832x600 mode called PiccoSD64: 8Bit 800x600
for example) might confuse rtgmaster.library and the graphics will look strange with such
modes.
- rtgmaster.library is FREEWARE, may be ABSOLUTELY free copied, and might even included
in commercial games for free. I would like a free copy of the game then, though... of course,
if a programmer/publisher is too AVARICIOUS for this, he may include it anyway for free...
even if i do not LIKE it that way...
- As always, the developpers of rtgmaster are in no way responsible to damage done to your
system (but there should not be such damage, i wrote this done, only to be sure...)
- If you have an OLD VERSION of rtgmaster.library installed, replace it with THIS version.
As to this version of rtgmaster.library, some things changed... for example, the older
versions of the c2p algorithms did not support ECS... also, older versions than V2.30
of rtgCGX.library tended to crash on the Merlin GFX Board (a bug in the Merlin software,
created a workaround for it... as to my Betatesters, that workaround works ...)

1.3 Rtgmaster.library V13.0

The RtgMaster Library - EASY GFX BOARD ADAPTION FOR GAMES AND DEMOS
Table of contents (Developpers)
I. Introduction
II. Installation
III. Short description
IV. Remarks to some Sublibs
V. Changes to the Beta
VI. Some remarks about ECS

1.4 Rtgmaster.library V13.0

Well, ECS is old stuff, like we all know. But if it is EASY to support : WHY NOT ? :)
There are three levels of ECS support for rtgmaster :
1. "I don’t waste my time for ECS"
Simply put -1 to smr_PlanarSupport, so that choosing a EHB mode is allowed for the user. rtgmaster will try its best to run the program under ECS, but the result won’t look that good (very dark... as it uses a 256 color palette on a 64 color screen...
2. "I waste SOME time for ECS"
Put -1 to PlanarSupport, and find out how many colours the opened screen has, using the GetRtgScreenData call. If you got 256 colors, use a 256 color palette, else (EHB mode for example) a 64 color palette. The result looks much better, good enough for most games/demos. But some things still might look strange.
3. "I want full ECS Support"
Well, i don’t think, anybody will still do this at these times... but well... IF you want full ECS Support : Put -1 to PlanarSupport, provide 256 color palette for AGA, 64 color palette for ECS (ECS can use EHB mode only with Lowres Screens, BTW... so 320x256 or 320x512 are the highest possible ECS resolutions for 64 colors). AND do two versions of your GFX files... one with color values from 0-255 (AGA), one with color values between 0 and 63 (maybe with a converter). Well, for myself, that’s too much work :)
Most people have at least one of the two : CyberGraphX,AGA.

1.5 Rtgmaster.library V13.0

This library makes it possible for programmers to easily do demos/games that run on ECS/AGA *and* Graphics Boards. It is Freeware, so you can use it in your project without having to pay for it. It would be fine, though, if you gave me a free copy of your game, then :)
rtgmaster.library COMPLETELY replaces GFXStuff141.lha on Aminet, that contained some code out of rtgmaster.library. This is no Beta release anymore, it is working stuff and could be used in your projects RIGHT NOW.

1.6 Rtgmaster.library V13.0

Some Benchmarks
I did some tests using the "Flame" Demo that is provided with source as example with this package. "Flame" is a "Copper style" Plasma demo, running in 320x200 (or in 160x120, if small is specified as parameter).
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

A2000/060 50 MHz GVP Spectrum CyberGr. 20 fps 63 fps

NOTE: 040 optimized c2p did not work on 060, i

used the 020 optimized c2p for 060 + AGA then...

Anoter NOTE: If you have the Merlin or Merlin II Board, use at

least Probench3.0 and rtgCGX.library V2.30. Due to a bug in

Probench, older versions won’t work (i created a workaround

for the bug, that is working, according to my Betatesters :))

1.7 Rtgmaster.library V13.0

Introduction

In the meantime, AGA probably is not anymore the HIT as to graphics speed.

And there are a lot of people who do not have an AGA system, but who have

a GFX Board. So the question is : Why can’t they run decent games or demos on their systems ?

The answer : Most Amiga Workbench Emulations for Graphics Boards had only applications in mind, and some games/demos related stuff is a bit complicated to do with them. Also, there is no real standard... we have CyberGraphX, EGS, Picasso WB-Emu, Retina WB-Emu, HRG-System ... which one to support ?

rtgmaster.library is a Library that provides an interface to all existing WB-Emulations (well, up to now only CyberGraphX, EGS and Picasso II, but more will come :)) and even to not-Graphics-Board Display Hardware (AGA, ECS, and in the future probably Graffiti and AGX). Also, the library provides some special functions that simplify a lot of things that are a bit difficult in most WB Emulations (Double/Triple Buffering, direct access on the Video RAM,...). The library is about 90% ASM coded (well, the Screenmoderequester is in C :)).

With rtgmaster.library you won’t have to bother any longer about what GFX Hardware you got, you simple call (as an example) OpenRtgScreen, and if there is for example a Cybervision board, rtgmaster.library calls the rtgCGX.library and performs an OpenRtgScreen call specially optimized for the Cybervision. If, contrary, it finds an unexpanded A1200 instead of a Cybervision, rtgAMI.library is called and a OpenRtgScreen for AGA and ECS systems will be called. The Assembly source of all these calls is done in a way, that it is VERY FAST.

Of course, for ECS and AGA you will still need to put a c2p algorithm between your code and the library, where the library supports a standard format for c2p algorithms. The user can choose the c2p algorithm in the Screenmoderequester, and with a library function the chosen c2p algorithm will be called then.

So the game/demo will run on optimal speed on ANY system, even if there comes a new/faster c2p out sometimes... the c2p algorithms sit in libs:rtgc2p. Two are already provided in this package.

As you probably understand, rtgmaster.library has features to examine an opened screen, to look, how its video RAM is organized, so you are able to use your own very fast "Write to the Screen" functions. Later we will add some high-level-calls, but the feature to do everything oneself (what often is the fastest way to do it) will stay with the library. Later versions will be compatible to this version (only some new functions...).

This first Aminet release of rtgmaster.library MIGHT not be any longer compatible to early Beta-Versions that were spread the last half year. Read the docs carefully, and to go sure, maybe reassemble/recompile your stuff...

rtgmaster.library has NOTHING to do with "RTG" from Commodore or with a workbench emulation. It provides stuff to do GFX Board (and AGA/ECS) compatible games and demos,

and to do this very FAST. You need at least an 68020 and OS2.0 to use it. For the
CyberGraphX sublibrary you need OS3.0 (as CyberGraphX needs this) and at least
cybergraphics.library V40.65.
Rtgmaster.library is Freeware (But even if you may copy it for FREE, it is copyright
to Steffen Haeuser, John Hendrikx, Wolfram Schenk, and Hans-Joerg and Thomas Frieden).
It maybe copied with commercial products that use rtgmaster.library, and it might
be put on PD/Shareware CDs.
The original idea was from John Hendrikx (Textdemo57-coder), and most parts of
rtgmaster.library and rtgAMI.library are done by him. Rest of rtgmaster.library
and rtgAMI.library are done by me, Steffen Haeuser. rtgEGS.library and rtgCGX.library
are done by me, too (in the meanwhile i overtook the developpement of rtgmaster.library
and rtgAMI.library completely). rtgPICA.library is done by Hans-Joerg Frieden. The
new Screenmoderequester (the old one in the Beta Version was dropped, because it was
too buggy) is done by Wolfram Schenk.
If you want to do a sublibrary for Merlin, Retina, Graffiti,... feel free to contact
me. Also, if you have some more demos, that i could include with this package... :)

1.8 Rtgmaster.library V13.0

Simply run the installer script. (Well, if you already have an older
version installed, better get sure, it REALLY installed the new version...
sometimes this installer script sucks... probably i should do a rework
of it soon... but if you never installed it before, it will work for sure...)
Rtgmaster.library consists of :
- rtgmaster.library in libs:
- the sublibraries in libs:rtg
- the c2p algorithms (for ECS/AGA) in libs:rtgc2p
- Includes and documentation
- demos

1.9 Rtgmaster.library V13.0

rtgAMI.library :
The thought came to me, one could include a
"Pseudo-Screenmode" called FAST:Lowres that in FACT does a 320x256 display WITHOUT
an intuition screen, doing ALL THAT SILLY HARDWARE HACKS that are possible, using
a 2x2 display. But i do not know enough about that hardware hacking, to do such
a thing myself... anyone else ? :)
THIS LIBRARY IS FOR AGA AND ECS.

rtgEGS.library : The Blitter functions do wait till the Board blitter has finished
its job ALL the time, even if you do not do "WaitRtgBlit". That is EGS's fault, i can't
do anything about it... but as the BlitRtg function is not implemented up to now anyway,
that shouldn't bother you up to now :)
While doing RtgBlit for rtgEGS.library i noticed that EB_BitBlt did not work right
on 24 Bit Screens. I took EB_CopyBitMap for the minterm $C0, as this takes the blitter
too... but as i do not see myself as EGS Betatester, i did not try, if all other
combinations would work... so if you want EGS compatibility with minterms other than
$C0 ... good luck :)
THIS LIBRARY IS FOR EGS GRAPHICS BOARDS WITH INSTALLED EGS SYSTEM. IT SUPPORTS :
PICCOLO, PICCOLO SD64, EGS110, EGS SPECTRUM, RAINBOW 3, MAYBE RAINBOW 2,
MAYBE EGS GRAFFITY, IF THIS BOARD WAS EVER BUILT. DOES NOT RUN ON THE A2410.
rtgCGX.library : Better do not implement gadgets at the top few lines of the Screen. As to
the way CyberGraphX is implemented there is the Screen bar and if you press a mouse button
insise a CyberGraphX screen, EVEN if there is a big borderless Window in it, like it is in
rtgmaster.library, mouse movement STOPS when pressing the left mouse button inside the Screen
bar. Needs at least CyberGraphics.library V40.65BETA (or higher). V2.30 is the first
version of rtgCGX.library that runs stable under Probench3.0 CyberGraphX Emulation
for the Merlin/MerlinII GFX Boards.
THIS SUBLIBRARY DOES NOT SUPPORT A2410.
Some 24 Bit Modes (ARGB) seem to crash the system sometimes... it is definitely
CyberGraphX' fault, not that of rtgmaster.library... often, you can cause it to crash
by simply switching screens very often... could be a bug in the V40.65 CyberGraphX Beta.
This bug does not appear on 8/15/16 Bit Screens.
THIS LIBRARY SUPPORTS ALL GRAPHICS BOARDS WITH INSTALLED CYBERGRAPHX SYSTEM.
rtgPica.library :
THIS LIBRARY SUPPORTS PICASSO II BOARDS WITH INSTALLED PICASSO II WB EMU (NOT WITH
CYBERGRAPHX, FOR THAT USE RTGCGX.LIBRARY !!!)
Note: Some calls do not work on the Picasso (Text, Mousepointer stuff). I hope
the programmer of the sublibrary will fix this, as far as it is possible.
All Libraries : Later, when Graffiti and AGX versions will be there, it will be easier to
support them using the High level calls... of course writing directly to the Video RAM is
faster... some of the High level functions even use OS functions... and do color conversions...
for really fast Video RAM access CopyRtgPixelArray should be used, because it directly
accesses the Video RAM (on the other hand, it does no color conversions, so 15/16/24 Bit
stuff will be a lot of work... but color conversions slow things down...)
Use the call GetRtgScreenData to determine if you got a chunky display, a Bitmap display,
or something else.
Introduction

1.10 Rtgmaster.library V13.0

Short description

Up to now rtgmaster.library supports :

- ECS

- AGA

(for ECS/AGA only WriteRtgPixelArray/CopyRtgPixelArray

are still unimplemented)

- EGS (RtgSetPointer up to now only clears the pointer)

- Picasso II (Text/pointer calls unimplemented

- CyberGFX

- Merlin/MerlinII under Probench3.0 CyberGraphX Emulation

(since rtgCGX.library this runs stable, due to a workaround

around a Probench Bug)

The functions of rtgmaster :

- Double Buffering :

SwitchScreens is used to switch between buffers. No main CPU time is consumed,

but you may have to wait for the next frame to see the change happen. Up to now, only

two Buffers per screen (Front and Back) are supported, but that may change in the future.

You can get the video memory base Adress of a buffer using GetBufAdr.

- Opening Screens :

You use OpenRtgScreen for this job. You are able to specify minimum and maximum

values for resolution, depths and if you need Double Buffering. All boards at least

support 8 Bit Chunky and 24 Bit Chunky Screens. By the way,

using RtgScreenModeReq and FreeRtgScreenModeReq you are provided

with a screenmode requester interface to the Screenmodes. This function only exists in

the rtgmaster.library (no need for sublibrary programmers to code this... :)).

- Closing Screens :

You close Screens using CloseRtgScreen.

- Locking/Unlocking Screens :

To provide maximum system conformity, rtg.library provides LockRtgScreen and

UnlockRtgScreen.

- Changing the color map :

For 1 Bit and 8 Bit screens (for some boards for other types too...) you are able to

manipulate the color map using LoadRGBRtg.

- Segmented Memory :

Boards using segmented memory are supported using SetSegment and GetSegment.

For other boards these calls simply do nothing.

- Writing to the video memory :

You are able to get video RAM base addresses using GetBufAdr and write
to the memory yourself.
- Getting data about an already opened screen :
With GetRtgScreenData you get information about Width, Height, structure of the
video memory (RTGx or xRTG or ...), and if the screen supports Double Buffering and
to what Bussystem (Custom chips, Zorro Bus...) your GFX Board HW is connected.
- Finding out, if the RTG screen is in front :
This is done using RtgScreenAtFront.
- RtgWaitTOF and WaitRtgSwitch wait for TOF/for Doublebuffering to be
happened... might do nothing at all on some Boards, though...
- CallRtgC2P is implemented for c2p'ing. (on GFX Boards it does plain copy.
But direct write to Video RAM is faster for GFX Boards than write to buffer and
additional Write to Video RAM (from Buffer).
- WriteRtgPixel, WriteRtgPixelRGB, WriteRtgPixelArray, WriteRtgPixelRGBArray are
implemented (those MIGHT use OS calls), CopyRtgPixelArray is implemented (uses
direct Video RAM access for sure, and therefor should be preferred to the
WritePixel calls), RtgBlit, WaitRtgBlit and RtgBltClear
are implemented. WaitRtgBlit might do nothing at all for some Boards.
- DrawRtgLine, DrawRtgLineRGB, FillRtgRect and FillRtgRectRGB
are some more high level calls. These calls will use the GFX Board Blitter, if possible, otherwise they will
use Bresenham and Direct Video RAM access.
- RtgSetPointer and RtgClearPointer are mousepointer support functions.
They only work for CyberGraphX, ECS and AGA up to now. They will soon work for EGS
too. They probably NEVER will work for rtgPICA.library (currently, rtgPICA.library
will crash with those functions... soon it will simply do nothing...)
- RtgOpenFont, RtgCloseFont, RtgSetFont, RtgSetTextMode,
RtgSetTextModeRGB and RtgText are for Text/Font Support. Same
problems with rtgPICA.library like with the mousepointer stuff.
- RtgInitRDCMP, RtgWaitRDCMP, RtgGetMsg and RtgReplyMsg
are the support functions for the RDCMP (rtgmaster direct communication message port), that
handles inputevents much the same way like IDCMP does for Intuition. There probably will
be an update for the RtgGadgets package that uses RDCMP in the future. As to my personal
experience, the InputHandler method lost some mousekey events, if they came very fast.
Therefore i created RDCMP for rtgmaster V12, that fixed the problems (in providing
a way to store old input events, that might not yet been handled). The RDCMP supports
both WAITING for an event and POLLING (busy-looping without Wait). it reacts on keyboard
and mousebutton events.
More information :
- The RtgGadgets.lha Source does not work with ECS/AGA up to now.

Anyways, it is quite obsolete with the RDCMP added to rtgmaster...
better use RDCMP (there might be a RDCMP-using RtgGadgets.lha in
the future...)
- CopyRtgPixelArray and WriteRtgPixelArray do not work with ECS/AGA yet
- The New Parameters (SrcX/SrcY) of CopyRtgPixelArray are not implemented
up to now.

1.11 Rtgmaster.library V13.0

Bugliste/Unimplemented Features
- WriteRtgPixelArray, CopyRtgPixelArray are not implemented for ECS/AGA
- rtgPICA.library version number did NEVER change, even as
new features were included...
- RtgGadgets sources (at least the Gadget part of it) does not work
on ECS/AGA. Input-Handler part works, though.
- Versionnumber-check of rtgmaster.library (main library)
in installer script does not behave correctly... i managed
to fix it for the sublibs, though...
- On some systems, the Screenmoderequester crashed, if smr_PlanarSupport
is 0 !!!
- SrcWidth/SrcHeight of CopyRtgPixelArray not implemented yet. Set them to 0 currently.
- Mousepointer/Text-stuff crashes with rtgPICA.library, works okay on
Screens under CyberGraphX,EGS, AGA/ECS, though.
- RtgSetPointer does strange stuff under EGS, instead of setting it.
- RtgSetTextModeRGB does not work correctly on 15/16/24 Bit screens of the
rtgCGX.library (due to a bug/missing feature in CyberGraphX).
- On SD64 Boards, rtgmaster claims 16 Bit modes being 24 Bit Modes.
Only a bug in the Screenmoderequester... the programs will behave correctly...
(16 Bit)
Some things that aren’t nice, but probably won’t change :
- If you try to open a Screen that does not fit to the Video Ram of your
Board (1280x1024 24 Bit 3 Buffers for example :)), your system might
crash.
- The rtgmaster.library does not test, if your MONITOR can display a
choosen Screenmode. So you should look, that no Screenmode that might
damage your monitor is installed with the WB Emulation of your computer.
- If you want to use rtgmaster.library, do NOT use the utility DosWedge.
It causes rtgmaster.library to *CRASH*.
- When testing i found a "CyberGraphX Screenmode that LIED...". It claimed
to be a 800x600 mode, but was IN FACT a 832x600 mode. Some functions of
rtgmaster.library do not like LYING BASTARDS :) ...

1.12 Rtgmaster.library V13.0

Authors and Copyright

Rtgmaster.library is FREEWARE (but up to THIS release you only may copy this package
to another person, if you have the okay from one of the authors... this will
change soon of course, and it will get TRUE freeware).

If you use rtgmaster.library in own project you should mention this in the documentation
of your software (of course you could give the authors of rtgmaster.library a free copy,
but do not NEED to do so, if you do not want to ... :))

Send bug reports, critics and other remarks to me, Steffen Haeuser. If
you want a QUICK answer, do not use my old haeuser@tick.informatik.uni-stuttgart.de
email address, but use MagicSN@birdland.es.bawue.de

EMail : John.Hendrikx@grafix.xs4all.nl
Snail Mail : John Hendrikx
Figarostraat 36
3208 PD SPIJKENISSE
NETHERLANDS
(Till Feb. 96 probably)
(Former Main Programmer)

EMail : MagicSN@birdland.es.bawue.de or haeuser@tick.informatik.uni-stuttgart.de
Snail Mail : Steffen P. Häuser
Limburgstraße 127
73265 Dettingen/Teck
GERMANY
(Main Programmer, rtgmaster.library and
CyberGraphX, EGS, and AMI Sublibraries, Documentation)

EMail : schenkwm@tick.informatik.uni-stuttgart.de
Snail Mail : Wolfram Schenk
Neckartenzlinger Strasse 16
72658 Bempflingen
GERMANY
(Author of the Screenmoderequester)

EMail : tfrieden@fax.uni-trier.de,hfrieden@fax.uni-trier.de
Snail Mail : Thomas & Hans-Joerg Frieden
Schlossstr. 176
54293 Trier
(Picasso II Sublibrary and Demo Sources)

EMail : O.Asholz@btx.dtag.de
Snail Mail : ???
(Moon Demo Program)

1.13 Rtgmaster.library V13.0

History

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 throwed
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 gdecs 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 CopyRtgPixelArray
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 willdo strange things when blitting to
Buffer 2" Bug
Removed d6/d7 Parameters of CopyRtgPixelArray

(this time it is final !!!), Recompiling/Assemblong
needed,sorry...
Fixed Installer Script and some minor Bugs
OffX/OffY parameters of CopyRtgPixelArray
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...

1.14 Rtgmaster.library V13.0

Changes to the Beta
Things people who used the Beta Version of rtgmaster.library
have to be careful about :
The Tags for the Screenmode Requester have changed a bit. Some Tags were added,
some deleted. Look at the includes and recompile your stuff :)
... and the ChunkySupport Bits changed a bit... sorry about that ... :(
... so, if you used one of the Beta-Versions of rtgmaster, better
read the docs very CAREFUL... and recompile/reassemble all your stuff done with it :)
... Such things won't happen anymore in the future... future versions will
be compatible, as this is the first public release.
NOTE: CopyRtgPixelArray changed a LOT with some new parameters...
but it is backward compatible (if the new parameters are set to 0,
it will behave as before, so old executables will still run)

1.15 Rtgmaster.library V13.0

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.
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 exits 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 guarantie

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 migh 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...)