BlitzBasic 2 Library Commands V1.2

Jurgen Valks.

COLLABORATORS			
	TITLE : BlitzBasic 2 Library Commands V1.2		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jurgen Valks.	January 19, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

BlitzBasic 2 Library Commands V1.2

1.1 BlitzBasic 2 Library Commands

```
                                    BLITZ BASIC 2 LIBRARY GUIDE V1.3

                          Last updated on: 16-10-1994
                            written by Jurgen Valks
                       mail me for futher updates: j.valks@hsbos.nl

                NOW INCLUDING THE BUM MAGAZINE COMMANDS!!: Thanks Simon!

                                  Make a choice


                All PD/Update commands

                See all the PD libraries included

                BUM Magazines 1-6 / Commands

                BUM 7

                INFO
```

1.2 libraryindex

```
                    This file contains all the commands of the following  ↩
                       libraries:


                CIA-TRACKER library

                COMMODITIES library

                ELMORE library

                FX library
```

FNS library

FUNC library

GFX library

PACK library

PCF library

REQ library

RIANIM library

TOOLTYPES library

TRACKDISK library

WB library

ZONE-JOY library

1.3 trackmain

```
                ==========================================
=    T R A C K D I S K    L I B R A R Y    =
==========================================
```

(C)1994 Reflective Images

Written by Steve Matty.

You can do whatever the hell you like to this library but must still
give me some credit!

Command List :

CloseDisk

MotorOn

MotorOff

=FormatTrack

=OpenDisk

=ReadSector

=WriteSector

=WriteBoot

1.4 track_opendisk

```
Command : OpenDisk
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=OpenDisk(unit#)

This attempts to open unit 'unit#' of the trackdisk.device, for use with
the other commands in this library. A return value of 0 indicates
failure, -1 indicates success.
```

1.5 track_motoron

```
Statement : MotorOn
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : MotorOn unit#

This attempts to switch the drive motor on of the previously opened
trackdisk unit (called with OpenDisk). You must call this command
before attempting to ReadSector/WriteSector/FormatTrack/WriteBoot
```

1.6 track_motoroff

```
Statement : MotorOff
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : MotorOff unit#

This turns the drive motor of 'unit#' off.
```

1.7 track_rs

```
Command : ReadSector
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : [success=]ReadSector(unit#,sector#,buffer[,numsectors])

This attempts to read 'numsectors' sectors from a trackdisk device which
has been opened with OpenDisk and has its Motor On. If numsectors is
omitted then 1 sector is read. The data is read into the memory location
pointed to by 'buffer'.

WARNING! Please MAKE SURE the MOTOR is _ON_ otherwise, all hell will break
loose!!!
```

1.8 track_ws

```
Command : WriteSector
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : [success=]WriteSector(unit#,sector#,buffer[,numsectors])

This is the same as ReadSector except.......... it writes! (and no, I
am not being lazy by not typing any decent docs)
```

1.9 track_ft

```
Command : FormatTrack
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : [success=]FormatTrack(unit#,track#,buffer[,numtracks])

This does a TD_FORMAT on the specified track number. Buffer should point
to the area of memory which the track should be formatted with. I don't
know why this command exists - but hey, it might come in useful.
```

1.10 track_closedisk

```
Statement : CloseDisk
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : CloseDisk unit#

This closes the trackdisk.device of the specified unit#. The Motor is
automatically switched off if it is already on.
```

1.11 track_wb

```
Command : WriteBoot
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : [success=]WriteBoot(unit#[,buffer])

This writes 1kilobyte of data to the bootblock of the specified disk unit.
The optional buffer parameter should point to an area of memory with which
to write the bootblock.
```

1.12 rianimmain

```
                RIAnim Library v1.0
==================

By Stephen McNamara
(c)1994 Reflective Images


                 RIANIM COMMANDS
                This library enables the playback of both Anim5 and Anim7 format
animations.  It allows you to playback animations at any co-ordinate in a
bitmap and supports different palettes for frames of the animation.  It
also allows you to playback animations from FAST ram, thus you can now play
massive animations that can only fit in FAST ram.

When playing back animations you must make sure that your display is
double-buffered.  Please refer to the Blitz manual for information about
how anims can be played back properly - or look at the example program
included with this file.

Note: there may still be a few bugs in the animation playback routines - if
you have any problems or spot any bugs then please contact us at the
address given in the main file of this archive.
```

1.13 rianimindex

```
                     These are the RIANIM library commands:


                AnimLoop

                RIAnimInit

                RINextAnimFrame

                =RIAnimInit

                =RINextAnimFrame
```

1.14 rianim_init

```
Statement/Function: RIAnimInit
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: [suc=]RIAnimInit(address,bitmap#,palette#[,xy_offset])

This command attempts to take an animation held in memory (CHIP or FAST)
and identify it as a supported animation format.  If it identifies it
okay it will set up the animation by unpacking frame 1 of the anim onto
the specified bitmap and copying the palette to the specified palette
object.
```

```
You must ensure that the bitmap is big and deep enough to actually hold
the animation.  At the moment there is no checking of the bitmap size.
The palette object you give is automatically resized to the size of the
palette in the animation.

The optional parameter allows you to play an animation at an offset into
a bitmap.  Thus you could center a half screen animation on a bitmap.
The offset is given as a byte offset from the start of each bitplane.
It is calculated like this:

      offset=(X/8)+(Y*(pixel_width/8))

      where: X and Y are your co-ordinates
             pixel_width is the width of your bitmap.

If used as a function, this command returns true for a successful
initialise or false for failure.
```

1.15 rianim_nextf

```
Statement/Function: RINextAnimFrame
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: [suc=]RINextAnimFrame bitmap#

This command attempts to unpack the next frame of a previously
initialised animation onto the specified bitmap.  It returns true or
false to say whether it succeeded or not.
```

1.16 rianim_loop

```
Statement: AnimLoop
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: AnimLoop ON|OFF

  This command allows you to control the looping mode of the animation.
With animloop off, playback of an animation will stop at the last frame
of it. Any attempt to draw another frame will fail.  With it on,
though, the animation will loop around.

Note: you must ensure that your animation has loop frames at the end of
it if you want to loop the animation around.  The reverse of this is
true for animloop off - the animation must not have loop frames if you
don't want it to loop around.  If you select animloop off but have
looping frames in your anim then the animation will end by displaying a
copy of frame 2 of the animation.
```

1.17 commoditiesmain

```
                ------------------------------------------------------------------------

====          Reflective Images Commodities Library V0.9 (C)1994      ====
-------------------------------------------------------------------------


                 COMMODITIES COMMANDS
                Introduction
============

This library allows the easy use of Commodities. It requires Kickstart 2
or higher.
```

1.18 commoditiesindex

```
                          The COMMODITIES library commands:


                 ExchangeAppear

                 ExchangeDisAppear

                 ExchangeEnable

                 ExchangeDisAble

                 ExchangeKill

                 ExchangeChangeList

                 ExchangeUnique

                 MakeCommodity

                 SetStatus

                 SetHotKey

                 =HotKeyHit

                 =CommodityEvent

                 =ExchangeMessage

                 =CxAppear

                 =CxDisAppear

                 =CxEnable

                 =CxDisable

                 =CxKill
```

```
                =CxChangeList

                =CxUnique
```

1.19 comm_make

```
Function : MakeCommodity
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=MakeCommodity(name$,title$,description$)

This command attempts to add your Commodity to the list of commodities.
A return value of -1 indicates success, 0 means failure. (not enough
memory)

name$ refers to the name of the Commodity and it should be unique. This
is the name that appears when running the Commodity Exchange program.
title$ is the title of your program, e.g. "My Screen Blanker".
description$ is a brief description of your program.

The Commodity Exchange program will then have 'name$' in its list of
Commodities and when a user clicks on your commodity, it will display
the title$ and description$.
```

1.20 comm_setkey

```
Function : SetHotKey
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=SetHotKey(hotkey#,hotkeydescription$)

This will add a hotkey event to your commodity so that after a hotkey
has been pressed you can find out which one.

e.g.    success=SetHotKey(0,"lalt lshift a")
```

1.21 comm_hit

```
Function : HotKeyHit
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : hitkeynum=HotKeyHit

This will return the number of the hot key which has been hit since the
last 'CommodityEvent' was called, or -1 if no such hotkey has been
activated.
```

1.22 comm_event

```
Function : CommodityEvent
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : anyevent=CommodityEvent

This looks to see if either
  a) A hotkey has been pressed
  b) A message from Exchange has been received

and returns -1 if such an event occurred, of 0 is nothing has yet
happened. This should be inside a Repeat-Until loop, e.g.

  Repeat
    VWait
    ev.l=Event
    ce.l=CommodityEvent
    hk.l=HotKeyHit    ; This must be used after
  Until ev or ce or hk            ; CommodityEvent
```

1.23 comm_setstatus

```
Statement : SetStatus
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : SetStatus on|off

This sets the status of your Commodity to either Active (on) or Inactive
(off) - this can be seen by running the Commodities Exchange program.
```

1.24 comm_exmessage

```
Function : ExchangeMessage
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : messnum.l=ExchangeMessage

This looks to see if the Commodities Exchange has issued you with as
message, e.g. Hide Interface, Show Interface. It returns the message ID
of the incoming message or 0 for no message.
```

1.25 comm_cx

```
Functions: CxAppear/CxDisAppear/CxEnable/CxDisable
                       CxKill/CxChangeList/CxUnique
------------------------------------------------------------------------------
Modes : Amiga
```

```
These are to be used in conjunction with ExchangeMessage, ie

  em.l=ExchangeMessage
  Select em
    Case CxAppear
      Gosub _appear
    Case CxDisAppear
            Gosub _disappear
        End Select

The functions merely return the ID value associated with that particular
Commodities Exchange message.
```

1.26 comm_ex

```
Functions: ExchangeAppear/ExchangeDisAppear/ExchangeEnable/
ExchangeDisable/ExchangeKill/ExchangeChangeList/ExchangeUnique
------------------------------------------------------------------------

Modes : Amiga

To be used in conjunction with ExchangeMessage, ie

  em.l=ExchangeMessage
  If em
    If ExchangeAppear then Gosub _appear
    If ExchangeDisAppear then Gosub _dispappear
  EndIf

This is intended as an alternative way of acting upon Exchange Messages.
```

1.27 wbmain

```
                ;------------------------------
;- WB library version 0.9     -
;- ©1994 Reflective Images    -
;------------------------------


                 WB COMMANDS
                This small library provides quick and easy to use commands for  ↩
                   accessing
AppWindows, AppIcons and AppMenus.

* PLEASE NOTE *
This library must have at least V37+ of Workbench/DOS/Icon libraries

This version of the library only enables you to read the FIRST file
dragged to an AppWindow/AppIcon or selected from an AppMenu - future
versions will have additional commands AppWindowArg/AppIconArg/
AppMenuArg which returns the filename of the specified arg. E.g.
f$=AppIconArg(1)
```

1.28 wbindex

```
                                                The WB commands:


                AppEvent

                AppWindowEvent

                AppIconEvent

                AppMenuEvent

                AddAppWindow

                AddAppIcon

                AddAppMenu

                DelAppWindow

                DelAppIcon

                DelAppMenu

                AppWindowFile

                AppIconFile

                AppIconHit

                AppMenuFile

                AppMenuHit
```

1.29 wb_appevent

```
Function : AppEvent
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppEvent

This command checks the msg ports of any open AppIcons/AppWindows/
AppMenus and if an event has been passed, returns -1. 0 indicates no
event has occurred.

e.g.
  Repeat
    VWait
  Until AppEvent
```

1.30 wb_appwindowevent

```
Function : AppWindowEvent
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppWindowEvent

This command checks the msg ports of any open AppWindows and if an event
has been passed, returns -1. 0 indicates no event has occurred.

e.g.
  Repeat
    VWait
  Until AppWindowEvent
```

1.31 wb_appiconevent

```
Function : AppIconEvent
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppIconEvent

This command checks the msg ports of any AppIcons and if an event has
been passed, returns -1. 0 indicates no event has occurred.

e.g.
  Repeat
    VWait
  Until AppIconEvent
```

1.32 wb_appmenuevent

```
Function : AppMenuEvent
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppMenuEvent

This command checks the msg ports of any AppMenus and if an event has
been passed, returns -1. 0 indicates no event has occurred.

e.g.
  Repeat
    VWait
  Until AppMenuEvent
```

1.33 wb_addappwindow

```
Function : AddAppWindow
--------------------------------------------------------------------------------
```

```
Modes  : Amiga
Syntax : success=AddAppWindow(windownumber)

This command attempts to make the window specified by 'windownumber' to
become an AppWindow. -1 means success, 0 means failure. There is a
currently limit of 4 AppWindows.
```

1.34 wb_addappicon

```
Function : AddAppIcon
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=AddAppIcon(id,text$,iconname$)

This command attempts to place an AppIcon onto the Workbench desktop.
ID is a unique identification number. Text$ is text to display
underneath the AppIcon and Iconname$ is the name of the file to use the
Icon imagery. -1 means success, 0 means failure.

e.g.
  suc=AddAppIcon(0,"Test","Work:Test")
  If suc=0 Then End
```

1.35 wb_addappmenu

```
Function : AddAppMenu
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=AddAppMenu(id,text$)

This command tries to add 'text$' to the Tools menu of Workbench.
ID is a unique identification number. Returns -1 for success, 0 for
failure.

e.g.
  suc=AddAppMenu(0,"Blitz2")
  If suc=0 Then End
```

1.36 wb_appwindowfile

```
Function : AppWindowFile
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : filename$=AppWindowFile(windownumber)

This command returns the complete path of the file which was dragged to
the AppWindow. If the file was in fact a directory a '/' is appended.
An empty string signifies nothing was Dragged.
```

1.37 wb_appiconfile

```
Function : AppIconFile
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : filename$=AppIconFile(id)

This command returns the complete path of the file which was dragged to
the AppIcon. If the file was in fact a directory a '/' is appended.
An empty string signifies nothing was Dragged.
```

1.38 wb_appmenufile

```
Function : AppMenuFile
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : filename$=AppMenuFile(id)

This command returns the complete path of the file which was selected
when the AppMenu was hit. If the file was in fact a directory a '/' is
appended. An empty string signifies nothing was selected.
```

1.39 wb_appiconhit

```
Function : AppIconHit
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppIconHit(id)
   idnumber=AppIconHit

This command returns the status of the AppIcon <id>. -1 = The icon was
doubleclicked, 0 = nothing has happened.

If no argument is supplied, the function returns the number of the
doubleclicked icon, or -1 for none.
```

1.40 wb_appmenuhit

```
Function : AppMenuHit
--------------------------------------------------------------------------------
Modes  : Amiga
Syntax : status=AppMenuHit(id)
   idnumber=AppMenuHit

This returns the status of the AppMenu item <id>. -1 = This menu
item was selected, 0 = This menu item was not selected.
If no argument is given, the function returns the numbe of the
selected menu item, or -1 for none.
```

1.41 wb_delapp

```
Function : DelAppWindow/DelAppIcon/DelAppMenu
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : success=DelAppWindow[(number)]
   success=DelAppIcon[(id)]
   success=DelAppMenu[(id)]

These commands will remove the AppWindow/AppIcon/AppMenu from the system
and free up the associated message ports.

*** IMPORTANT *** You must call DelAppWindow BEFORE closing a window,
or your machine will GURU!
```

1.42 toolmain

```
                    Reflective Images Tooltypes Library
      ===================================

                                  Release #2


                 TOOLTYPES COMMANDS
                By Stephen McNamara, inspired by the collection of tooltype  ↩
                   functions by
Mark Tiffany.

(c)1994 Reflective Images

This library contains commands to allow the reading, comparing and
setting of tooltypes in a .info file.  All tooltype names are case
insignificant but as a general sort of rule they should really be
completely uppercase.

This library attempts to open the system Icon.library, if the opening of
this library fails ALL commands in this library will be unusable.
Almost every function in this library relies on the Icon.library
completely.

Changed commands:

FindToolValue - now returns "" if the tooltype was found but did not
                have a value (e.g. DONOTWAIT).  You should now use
                FindToolType to check for the existance of a tooltype
                and then use FindToolValue to get its value.

PutIconObject - now has an optional parameter that lets you set the type
                of the file.  See SetIconType for more information
                about possible values for this command.
```

1.43 toolindex

```
Command list:

 GetIconObject

 PutIconObject

 FreeIconObject

 FindToolValue

 FindToolNumber

 IconDefaultTool

 IconRender

 MatchToolValue

 SetIconHit

 SetIconType

 SetToolValue

 ShapeToIcon

 NewToolType

 ClearToolTypes

 =FindToolType
```

1.44 tool_seticonhit

```
Statement: SetIconHit
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : SetIconHit width#,height#

This command sets the size of the 'hit-box' around the image in the
currently loaded .info file.  This is only of use if your info file has
an image associated with it.  You should note that the hit box should
never be smaller, horizontally or vertically, than the actual size of
the image.

When Workbench renders an image for a file onto a window, it
automatically puts a 3d box border around it. The size of the hit box
determines the size of this border.  Your image will always be located
in the top left border of the hit box.
```

1.45 tool_shapetoicon

```
Statement: ShapeToIcon
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : ShapeToIcon shape#[,shape#]

This command lets you change the images associated with the currently
loaded .info file.  What it does is to set up the .info file in memory
so that when it is saved out next, the images you give are saved out
with it.

Using this command does not actually copy any shape data around memory,
all it does it place a pointer in the .info to the shape data. You
should therefore not delete a shape WITHOUT first saving the .info file
to disk (that is of course if you want to keep your changes).

When you use this command, the hit box area for the .info file is
automatically set to the size of the first shape given.  It is
important, therefore, that the second shape is not larger than the
first. When you give a second shape, this shape is set up to be the
'alternate render' image, this means that this is the second image
associated with the .info file (remember the two windows in the
IconEditor?)
```

1.46 tool_seticontype

```
Statement: SetIconType
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : SetIconType type#

This command lets you specify the type of the file associated with the
currently loaded .info file.  The type describes whether or not the file
is a tool or project etc...., and can take the following values:

      1 Disk
      2 Drawer
      3 Tool
      4 Project
      5 Trashcan

This command is identical to the menu in the IconEditor 'Type'.
```

1.47 tool_iconrender

```
Statement: IconRender
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : IconRender mode#

This command lets you specify what Workbench should do to the icons
```

```
image when the user clicks on it.  It lets you choose whether a separate
image should be displayed or whether the current image should just be
modified.  Mode# is made up of several different values that should be
added together to create different effects, these are:

      0 Complement the select box
      1 Draw a box around the image
      2 Draw the alternate image
      3 Don't highlight
      4 Double image icon

Thus if you wanted an icon to change to a second image when selected,
and the icon has a second image, you would set the render to 6 (4+2).
This would mean that you had a second image (4) and that you wanted it
to be displayed when you select the icon (2).

Note: when you use ShapeToIcon with two shape numbers the IconRender is
automatically set to 6.
```

1.48 tool_icondefaulttool

```
Statement: IconDefaultTool
------------------------------------------------------------------------
Modes  : Amiga
Syntax : IconDefaultTool tool$

This command lets you set the default tool for the current .info file.
The default tool only applies for project files (see SetIconType) and is
the program that is run when you double click the icon file (e.g. all
Blitz2 source code files saved out with icons have the default tool
'Blitz2:Blitz2').

This command can be used to make a file saved out by your program
double-clickable.  I have used it myself to make map files saved out
from my editor automatically load the editor when selected.
```

1.49 tool_findtooltype

```
Statement: FindToolType
------------------------------------------------------------------------
Modes  : Amiga
Syntax : bool=FindToolType (tool$)

This command simply returns true or false to say whether or not the
given tooltype was found in the currently loaded .info file.
```

1.50 tool_giobject

```
Statement/Function: GetIconObject
------------------------------------------------------------------------
Modes  : Amiga
Syntax : GetIconObject filename$
   suc.l=GetIconObject (filename$)

This command reads in a .info file from disk.  The filename given will
have '.info' added to the end of it and will be loaded into memory (chip
or fast depending on what is available for allocation) as a diskobject.
Please refer to the Amiga hardware includes for information about the
diskobject structure (or see your Blitz Basic Amigalibs resident file).

If used as a function, this command will return either FALSE for failure
or the address of the allocated diskobject in memory.
```

1.51 tool_piobject

```
Statement/Function: PutIconObject
------------------------------------------------------------------------
Modes  : Amiga
Syntax : PutIconObject filename$
   suc.l=PutIconObject (filename$)

This command takes a diskobject structure reserved and initialised by
GetIconObject and saves it out to disk as a .info file for the specified
file.

All current tooltypes and values will be saved with the file. The
optional parameter allows you to set the type of the file associated
with the .info file.  See SetIconType for possible values for this
parameter.  Note that if you leave out this parameter the icontype will
not be changed.
```

1.52 tool_fiobject

```
Statement/Function: FreeIconObject
------------------------------------------------------------------------
Modes  : Amiga
Syntax : FreeIconObject
   suc.l=FreeIconObject

This command will free up the diskobject that is currently being used.
It will not save out any tooltype changes and will free up the memory
without ANY changes being made to the .info file loaded from disk.
```

1.53 tool_ftvalue

```
Function: FindToolValue
------------------------------------------------------------------------
```

```
Modes  : Amiga
Syntax : toolval$=FindToolValue(tooltype$)

This function returns the value of the selected tooltype.  The return
value is a string, and is the part of the tooltype string after the "="
in the tooltype entry.  The tooltype$ string that you pass can be in
either lower case or uppercase since all testing in done in uppercase,
although as a general rule, all tooltypes should be in uppercase.

This function will return a null string if the named tooltype was not
found in the list of tooltypes for the file.  If the selected tooltype
did not have an actual value (e.g. DONOTWAIT) then this function will
return the string "!!".
```

1.54 tool_ftnumber

```
Function: FindToolNumber
--------------------------------------------------------------------------
Modes  : Amiga
Syntax : toolval$=FindToolNumber(tooltype$)

This command will return the FULL tooltype string in the selected
tooltype position.  If the tooltype number does not exist then "" will
be returned.

  Example:  tooltypes: "DONOTWAIT"
         "CLOCKX=157"

  FindToolNumber(0) will return "DONOTWAIT"
  FindToolNumber(1) will return "CLOCKX"
  FindToolNumber(49) will return ""
```

1.55 tool_mtvalue

```
Function: MatchToolValue
--------------------------------------------------------------------------
Modes  : Amiga
Syntax : suc.l=MatchToolValue(tooltype$,value$)

This command searchs the current list of tooltypes for the selected
tooltype and, if found, attempts to match the values of it with the
given value.  This command uses the operating system call
MatchToolType(), it is able to cope with a tool having more than one
value,

      e.g. LANGUAGE=ENGLISH|FRENCH
      (the | is used to show OR, thus this tooltype
       means that LANGUAGE equals ENGLISH or FRECH)
    When using match toolvalue with this tooltype, TRUE will
                be returned when you use value$="ENGLISH" or "FRENCH"
                but not (I think) both.
```

```
You should note that for this command, the case of VALUE$ is
insignificant.
```

1.56 tool_stvalue

```
Statement/Function: SetToolValue
------------------------------------------------------------------------
Modes  : Amiga
Syntax : SetToolValue tooltype$,value$
   suc.l=SetToolValue (tooltype$,value$)

This command will attempt to set a tooltype that is currently defined to
the specified value.  When used as a function, this command will return
TRUE for success or FALSE for failure, possible failures include: no
icon file loaded and tooltype not found.  When used, this command
attempts to allocate memory to store the new tooltype information in,
it does not attempt to free up the old memory allocated to the tooltype.

This means that you should keep alterations of tooltypes to a minimum.
The best way to manage tooltypes is:

    1. Open the icon
    2. Read the tooltypes
    3. Close the icon
    4. ... do your program ...
    5. Open the icon
    6. Alter the tooltypes
    7. Save the icon

Using this series of events, you'll keep memory usage (which will be
fairly small anyway...) to the very minimum.
```

1.57 tool_nttype

```
Statement/Function: NewToolType
------------------------------------------------------------------------
Modes  : Amiga
Syntax : NewToolType tooltype$,value$
   suc.l=NewToolType (tooltype$,value$)

This command allocates a new tooltype in the currently loaded .info file
and sets its value.  No check is done to see is the tooltype already
exists and the new tooltype is added to the end of the current list of
tooltypes.
```

1.58 tool_cttypes

```
Statement: ClearToolTypes
------------------------------------------------------------------------
Modes  : Amiga
```

```
Syntax : ClearToolTypes

This command is used to clear all the tooltype information from the
currently loaded .info file.  It does not attempt, though, to free up
all the memory reserved to store tooltype names and values, you should
therefore not used this command too many times in a row.  Once you have
used this command, any attempt to read tooltype values will fail.
```

1.59 reqmain

```
                ;------------------------------
;- ReqLib.library version 0.9 -
;- ©1994 Reflective Images    -
;------------------------------


                 REQ COMMANDS
                The well known Req.Library for the Amiga is one of the best file
requesters around, so I wrote this small lib to enable Blitz users to
have Req requesters in their programs with the minimum of hassle.

* PLEASE NOTE * That this library must have at least v2.2 of the
Req.Library available.
```

1.60 reqindex

```
                Command List:

                 ReqOutput

                 ReqFileRequest

                 ReqFileLoc

                 REQ flags, structure
```

1.61 req_output

```
Statement: ReqOutput
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : ReqOutput windownumber

This command sets the ReqLib.library to put all requesters onto the
window specified by <windownumber>. If this command is not called
then the requesters will appear on the Default Public Screen.
```

1.62 req_filerequest

```
Function: ReqFileRequest
------------------------------------------------------------------------------
Modes :  Amiga Syntax :  pathname$=ReqFileRequest([title$[,flags]])

This opens up the standard file requester. If <title$> is given then
the text will appear on the requester title bar.
The optional <flags> parameter specifies a flag setting (see below)
for use. If this is omitted then the last flag setting is used.
```

1.63 req_fileloc

```
Function: ReqFileLoc
--------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : memorylocation.l=ReqFileLoc

This simply returns the address in memory where the Req.Library file
requester stucture is located.
```

1.64 req_flags

```
FLAGS
=====

Below is a list of possible flag settings and a brief description of each.


#FRQSHOWINFOB     = %1   ;Set to show .info files. Default is not.
#FRQEXTSELECTB    = %10   ;Extended select.  Default is not.
#FRQCACHINGB      = %100   ;Directory caching.  Default is not.
#FRQGETFONTSB     = %1000   ;Font requester rather than a file requester.
#FRQINFOGADGETB   = %10000   ;Hide-info files gadget.
#FRQHIDEWILDSB    = %100000   ;DON'T want 'show' and 'hide' string gadgets.
#FRQABSOLUTEXYB   = %1000000   ;Use absolute x,y positions rather than centering  ↩
   on mouse.
#FRQCACHEPURGEB   = %10000000   ;Purge the cache whenever the directory date stamp ↩
    changes if this is set.
#FRQNOHALFCACHEB  = %100000000   ;Don't cache a directory unless it is completely  ↩
   read in when this is set.
#FRQNOSORTB       = %1000000000   ;DON'T want sorted directories.
#FRQNODRAGB       = %10000000000  ;DON'T want a drag bar and depth gadgets.
#FRQSAVINGB       = %100000000000  ;Are selecting a file to save to.
#FRQLOADINGB      = %1000000000000  ;Are selecting a file(s) to load from.
#FRQDIRONLYB      = %10000000000000  ;Allow the user to select a directory, rather ↩
    than a file.


STRUCTURE
=========
```

Below is a description of the Req.Library file requester structure.

```
STRUCTURE AFileRequester,0
  UWORD frq_VersionNumber   ;MUST BE REQVERSION!!!!!!!!!!!!!!!!!

  ;You will probably want to initialize these three variables.
  APTR  frq_Title   ; Hailing text
  APTR  frq_Dir     ; Directory array (must be DSIZE+1 characters long)
  APTR  frq_File    ; Filename array (must be FCHARS+1 characters long)
          ; If you initialize this variable then the file requester will place the ↩
              complete path name in here on exit.
  APTR  frq_PathName    ; Complete path name array - (must be DSIZE+FCHARS+2 long)
          ; If you want the file requester to pop up on your custom screen, put  ↩
              one of your window pointers here.
          ; Or better yet, you can leave this field zeroed and put a pointer to  ↩
              one of your windows in the
          ; pr_WindowPtr field in your process structure.
  APTR  frq_Window    ; Window requesting or NULL
          ; Initialize these to the number of lines and columns you want to appear ↩
              in the inner window that
          ; displays the file names.  If you leave these set to zero then default  ↩
              values will be used.
  UWORD frq_MaxExtendedSelect ; Zero implies a maximum of 65535, as long as  ↩
     FRQEXTSELECT is set.
  UWORD frq_numlines    ; Number of lines in file window.
  UWORD frq_numcolumns    ; Number of columns in file window.
  UWORD frq_devcolumns    ; Number of columns in device window.
  ULONG frq_Flags   ; Various - umm - flags.  See above for more info.
  UWORD frq_dirnamescolor ;These five colors will all default
  UWORD frq_filenamescolor  ;to color one if you don't specify
  UWORD frq_devicenamescolor  ;a color (ie; if you specify color zero).
  UWORD frq_fontnamescolor  ;If you want color zero to be used, specify
  UWORD frq_fontsizescolor  ;color 32, or some other too large number
          ;which mods down to zero.

  UWORD frq_detailcolor   ;If both of these colors are specified as
  UWORD frq_blockcolor    ;zero then the block pen will be set to one.

  UWORD frq_gadgettextcolor ;The color for the text of the five boolean gadgets.   ↩
     Defaults to 1.
  UWORD frq_textmessagecolor  ;The color for the message at the screen top.   ↩
     Defaults to 1.
  UWORD frq_stringnamecolor ;The color for the words Drawer, File, Hide and Show.  ↩
      Defaults to 3.
  UWORD frq_stringgadgetcolor ;The color for the borders of the string gadgets.   ↩
     Defaults to 3.
          ;Unfortunately it is not possible to specify
          ;the color of the actual text in an Intuition
          ;string gadget.
  UWORD frq_boxbordercolor  ;The color for the boxes around the file and directory ↩
      areas.  Defaults to 3.
  UWORD frq_gadgetboxcolor  ;The color for the boxes around the five boolean  ↩
     gadgets.  Defaults to 3.

  STRUCT  frq_RFU_Stuff,36  ;This area, which is reserved for
          ;future use, should all be zero.
```

```
  STRUCT  frq_DirDateStamp,ds_SIZEOF  ; A copy of the cached directories date ↩
     stamp.
            ; There should never be any need to change this.

  UWORD frq_WindowLeftEdge; ;These two fields are only used when the
  UWORD frq_WindowTopEdge;  ;FRQABSOLUTEXY flag is set.  They specify
          ;the location of the upper left hand
          ;corner of the window.

  UWORD frq_FontYSize   ;These fields are used to return the selected
  UWORD frq_FontStyle   ;font size and style, only applicable when the
          ;font bit is set.

          ;If you set the extended select bit and the user extended selects, the ↩
             list of filenames will start from here.
  APTR  frq_ExtendedSelect  ; Linked list of ESStructures if more than one ↩
     filename is chosen.
          ;All of the following variables you shouldn't need to touch.  They ↩
             contain fields that the file
          ;requester sets and likes to preserve over calls, just to make life ↩
             easier for the user.
  STRUCT  frq_Hide,WILDLENGTH+2 ; Wildcards for files to hide.
  STRUCT  frq_Show,WILDLENGTH+2 ; Wildcards for files to show.
  WORD  frq_FileBufferPos ; Cursor's  position  and first
  WORD  frq_FileDispPos   ; displayed character number in
  WORD  frq_DirBufferPos  ; the three string gadgets.  No
  WORD  frq_DirDispPos    ; need  to initialized these if
  WORD  frq_HideBufferPos ; you don't want to.
  WORD  frq_HideDispPos
  WORD  frq_ShowBufferPos
  WORD  frq_ShowDispPos

; The  following  fields are PRIVATE!  Don't go messing with them or
; wierd  things may/will happen.  If this isn't enough of a warning, go read
; the one in intuition.h, that should scare you off.

  APTR  frq_Memory      ; Memory allocated for dir entries.
  APTR  frq_Memory2     ; Used for currently hidden files.
  APTR  frq_Lock      ; Contains lock on directories being read across calls.
  STRUCT  frq_PrivateDirBuffer,DSIZE+2  ; Used for keeping a record of which
            ; directory we have file names for.
  APTR  frq_FileInfoBlock
  WORD  frq_NumEntries
  WORD  frq_NumHiddenEntries
  WORD  frq_filestartnumber
  WORD  frq_devicestartnumber
  LABEL frq_SIZEOF

Enjoy!

Steve.
```

1.65 pcfmain

```
                    PCF Library - Picture Crunch Format
      ==================================

            -Brought to you by FUNdamental-


                 PCF COMMANDS
                About This Archive
------------------
```

  This archive contains:

    o A new library of commands for Blitz Basic 2
    o A compiled blitz program to generate PCF files
      from IFF files
    o A Blitz Basic and ASCII version of the same demo
                  program, to show use of the commands
          o A pre-converted image
    o This file 8)

All coding was written by Nigel Hughes, with a thank you to Steve from Reflective Images for his help with AllocDosObject and addressing objects in libraries. Not to mention the excellent RIB libraries.

About PCF Format

On the Blitz mailing list, there was a call for a method of protecting graphics from the "general public" I responded by saying,

  "Use my library"

And then disappeared to prepare for my finals! Well the finals are over and so here is version one of the PCF Library, version 2 will be out soon, more details later.

PCF is more compact graphics file format, that cannot be read by any general release paint package. There are commands within the library to cache these pictures and decompress to a bitmap only when you need them. Later versions will enable a coder to add his own personal tag so only he/she can decrypt the file.

Making a PCF Picture

In order to turn a IFF ILBM picture into a PCF file one need only use the picture_crunch program supplied in the archieve. Click on the "Load N Crunch" button to load an IFF and convert it to a PCF file. You will be asked if you wish to generate a V 1.0 file or the latest format. Please only select the "Latest Format" option as V1.0 is reserved for my use only and is protected and cannot be decrypted by any one else!

One can only use the Display gadgets once a IFF picture has been crunched, this is a bug that will be fixed in later versions. Sorry.

The Library

```
The library can be installed either by copying "PCF_Lib.obj" to your
BlitzLibs:Userlibs directory and then selecting "RELOAD ALL LIBS" in the
COMPILER menu in BB2, or by using the MakeDefLibs program after copying
"PCF_LIb.obj" to your BlitzLibs:Userlibs directory.
```

1.66 pcfindex

```
                        PCF Commands:


                    CachePCF

                    FreePCFCache

                    LoadPCF

                    UnpackPCF

                    PCFDepth

                    PCFHeight

                    PCFInfo

                    PCFVersion

                    PCFWidth

                    Other info
```

1.67 pcf_cachepcf

```
                  Function : CachePCF
------------------------------------------------------------------------------
Modes:  Amiga
Syntax: cache_ptr.l=CachePCF (Filename$,Memory Type,Cache Length)

The function loads a PCF file into memory, returning the pointer to
the cache. The Cache Length variable will contain the length of the
cache and is needed in order to use the FreePCFCache command. This
command does not cause the PCF image to be displayed.

If anything goes wrong during the loading of the file, no memory
will be allocated and 0 will be returned.

See Also:
                    FreePCFCache
                   ,
                    LoadPCF
```

,
UnpackPCF

1.68 pcf_freepcfcache

```
                Statement: FreePCFCache
------------------------------------------------------------------------------
Modes:  Amiga/Blitz
Syntax: FreePCFCache cache_ptr,cache_length

Frees the memory used by the PCF cache.

See Also:
                 CachePCF
```

1.69 pcf_unpackpcf

```
                Statement: UnpackPCF
------------------------------------------------------------------------------
Modes:  Amiga/Blitz
Syntax: UnpackPCF Bitmap#,Palette#,cache_ptr

Decompresses a PCF cache to a bitmap and palette. Both objects must
already exist. The PCF library currently makes no attempt to check the
bitmap or the palette are deep enough. If the bitmap is too large then
the image WILL be corrupt. The statement checks the version of PCF Cache
to ensure that it can decompress it!

See Also:  CachePCF ,
                 LoadPCF
```

1.70 pcf_loadpcf

```
                Statement: LoadPCF
------------------------------------------------------------------------------
Modes:  Amiga
Syntax: LoadPCF Bitmap#,Palette#,cache_ptr

Loads and decompresses a PCF image straight into the bitmap and palette.
The image is NOT cached afterwards. The same restriction apply to this
command as to UnpackPCF

See Also:
                 CachePCF
                ,
                 UnpackPCF
```

1.71 pcf_pcfinfo

```
                Statement: PCFInfo
------------------------------------------------------------------------------
Modes: Amiga/Blitz
Syntax: PCFInfo cache_ptr

Enables the use of PCFWidth, PCFHeight, PCFDepth, PCFVersion. These
commands will all return the relevant details about the cache pointed to
by cache_ptr. This allows a programmer to ensure that the destination
bitmap and palette are of the correct dimensions.

See Also:
                 PCFWidth
                ,
                 PCFHeight
                ,
                 PCFDepth
                ,
                 PCFVersion
```

1.72 pcf_pcfversion

```
Function: PCFVersion
------------------------------------------------------------------------------
Modes: Amiga/Blitz
Sytax: v.l=PCFVersion

Returns the version of the last cache interogated by PCFInfo.
```

1.73 pcf_pcfwidth

```
Function: PCFWidth
------------------------------------------------------------------------------
Modes: Amiga/Blitz
Sytax: v.l=PCFWidth

Returns the width of the last cache interogated by PCFInfo.
```

1.74 pcf_pcfheight

```
Function: PCFHeight
------------------------------------------------------------------------------
Modes: Amiga/Blitz
Sytax: v.l=PCFHeight

Returns the height of the last cache interogated by PCFInfo.
```

1.75 pcf_pcfdepth

```
Function: PCFDepth
------------------------------------------------------------------------------
Modes: Amiga/Blitz
Sytax: v.l=PCFDepth

Returns the number of bitplanes of the last cache interogated by
PCFInfo.
```

1.76 pcf_other

```
                Performence
-----------

The UnpackPCF command can decompress a 320x256 by 256 colour image
in under 10/50 of a second. For an image of 5 bitplanes or lower,
the command can often decompress in under a frame.

The PCF file format itself is usually about 1k smaller than the
related IFF file. This ratio will be much improved in the next
version.

The Futre
---------

I have a HUGE list of things to do, I just really wanted to get this
out to the general public so people can tell me what they think (wince).
But futre enhancements will include...

  o A PackIFF type command
  o Improved Compression rate
  o Unique key for decompression
  o Multiple file types, including Shapes
  o Multiple files in one PCF file.

Any bugs etc please contact FUNdamental at

  Nigel Hughes
  2 Slimmons Drive
  St. Albans
  Herts
  AL4 9AS

Until the 23rd Of June at nlh1@uk.ac.aber, and after
that via Mike Richards at mhr@uk.ac.aber.

Right, it is 1:05 AM and I am going to bed...

Nigel Hughes.


                 Back to main
```

1.77 packmain

```
                    PACK Library v0.1
        =================

    By Stephen McNamara with a little help from Steve Matty
    (c)1994 Reflective Images


                 PACK COMMANDS
                This library contains commands for the unpacking of ILBM's (IFF
pictures) and the grabbing of their palettes (CMAP chunks).  Nearly all
the commands in this library can be used as either STATEMENTS or
FUNCTIONS.

Usage is identical in both cases but if used as a function then the
command will return:

    FALSE for failure
    TRUE for success

Please feel free to critisise (or praise!) this library, send me
anything you want to say about it at:

      Stephen McNamara,
        17 Mayles Road,
              Southsea,
            Portsmouth,
             Hampshire,
               England.
               PO4 8NP.
  Telephone: (England) 0705 781507.

Or send us anything you've written........
```

1.78 packindex

```
                   These are all the PACK library commands:


                DeIce

                ILBMGrab

                ILBMPalette

                LoadIFF

                =LoadIFF

                UnpackIFF

                =ChunkHeader
```

```
                =DeIce

                =ILBMPalette

                =UnpackIFF
```

1.79 pack_unpackiff

```
Statement/Function: UnpackIFF
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: UnpackIFF address.l,bitmap#[,lines]
  suc=UnpackIFF (address.l,bitmap#[,lines])

This command is used to unpack an IFF picture file from memory onto a
bitmap.  Address.l should point to the START of the iff file header in
memory (either CHIP or FAST mem can be used), bitmap should be the
number of a previously initialised bitmap.  The optional lines parameter
allows you to specify the number of lines to unpack from the IFF file.

This command checks the size of the bitmap against the size of the IFF
before it unpacks the IFF onto it.  Checks are made for width, height
and depth of the bitmap and the IFF and the following is done:

(size=WIDTH, HEIGHT and DEPTH)

    BITMAP 'size' < IFF 'size' : unpack aborted
    BITMAP 'size' = IFF 'size' : pic is unpacked
    BITMAP 'size' > IFF 'size' : pic is unpacked

  Extra aborts can be caused by:
    - not using a previously installed bitmap
    - given the optional lines parameter as 0 or less
    - not giving ADDRESS.l as a pointer to a valid IFF ILBM
      header

When using the optional parameter, you should note that if you try to
unpack more lines than the IFF has, the unpack routine will
automatically stop at the last line of the IFF.  It will not reject the
UnpackIFF command.

NOTE: you should save your IFF pictures with the STENCIL OFF because at
the moment this routine does not check to see if STENCIL data is present
in the IFF file.
```

1.80 pack_ilbmpalette

```
Statement/Function: ILBMPalette
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ILBMPalette address.l,palette#
```

```
  suc=ILBMPalette (address.l,palette#)

This command is used to grab the palette from a IFF picture file held in
memory (CHIP or FAST mem).  Address.l should be given as the address of
either an IFF file in memory or a CMAP chunk in memory.  When you use
the SAVE PALETTE command from inside an art program (e.g. DPaint) or
from inside Blitz2, the program saves out a CMAP chunk which gives
details about the palette.  The CMAP chunk is also saved with IFF
picture files to give the palette of the picture.

This command will look at the address you gave and try and find a CMAP
chunk from the address given to address+5120.  If it finds a chunk it
will grab the palette into the given palette object.  If the palette
object already contains palette information then this information is
deleted. This routine looks in the CMAP chunk and reserves the palette
object to have the same number of colour entries.

This command will fail if it doesn't find a CMAP chunk.
```

1.81 pack_ilbmgrab

```
                Statment: ILBMGrab
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ILBMGrab address.l,bitmap#,palette#

This command lets you grab both the palette and the graphics from an IFF
picture file with just one command.  It returns to success parameter to
say whether or not it succeeded in grabbing the data, so if you need to
know if the grabbing was successful you'll have to use the separate
commands for grabbing palettes and graphics.

  NOTE: this command essentially just calls both
                UnpackIFF
                and

                ILBMPalette
                 so everything said about these commands is relevent for ILBMGrab ↩
                    .
```

1.82 pack_loadiff

```
                Statment/Function: LoadIFF
------------------------------------------------------------------------
Modes : Amiga
Syntax: LoadIFF filename$,bitmap#[,palette#]
  suc=LoadIFF (filename$,bitmap#[,palette#])

This command is a direct replacement for Blitz2's LoadBitmap. It is a
lot faster than Blitz's command since it loads the file into memory and
then unpacks it from there.  Thus you need to ensure that you have
enough free memory to load the IFF into before trying to use this
```

command.

This command is also more stable than Blitz's since it checks for the existence of the file before trying to load it in.

The optional parameter allows you to load in the palette of the IFF picture. Refer to UnpackIFF and ILBMPalette for more information about unpacking the graphics and grabbing the palettes.

IMPORTANT NOTE: to use this command you must have
our
FUNC
library installed in your copy of Blitz2.
Use of this command without this library will probably lead to a bad crash of your Amiga!

1.83 pack_deice

Statement/Function: DeIce

Modes : Amiga
Syntax: DeIce source_address,dest_address
suc=DeIce (source_address,dest_address)

This is a command from my (Stephen McNamara) past.
It is used to unpack data files packed by my favourite Atari ST packer - PACK ICE v2.40. I've put it into Blitz because still have loads of files that I've packed with it. To use it, source_address should (obviously) contain the address of the data, dest_address should be where to unpack the data to. In the function form, this command returns either 0 for unpack failed or -1 for success.

Note: The size of the data unpacked is the long
word at source_address+8 (I think, or is it 4?) if anybody is interested......

1.84 pack_chunkheader

Function: ChunkHeader

Modes : Amiga
Syntax: val.l=ChunkHeader (A$)

This command was put in by me (Stephen McNamara) before I realised Blitz already had a command that does exactly the same. I've left it in just because I want to. It is useful when looking through IFF files for chunks (e.g. ILBM, CMAP, etc.) as it gives you a longword value to look for in memory to find the chunk. The string should be a four character string (e.g. CMAP), you'll be returned the longword value of the string. This command does the job of the following bit of Blitz2 code:

```
a$="CMAP"
```

```
    val.l=Peek.l(&a$)
```

1.85 gfxmain

```
                    GFX Library v0.1
       ================

      By Stephen McNamara and Steve Matty
          (c)1994 Reflective Images


                GFX Commands
               This library contains commands for the control of palette objects ↩
                  inside
Blitz2.  These are just simple commands that allow either interrogation
of the palette objects are modifications to the colour values contained
in them.  After changing the palette with these commands, you'll have to
do either a USE PALETTE or DISPLAYPALETTE (whichever is applicable to
what you're doing) to make the changes come into effect on your screen.

Please feel free to critisise (or praise!) this library, send me
anything you want to say about it at:

  SIS3149@SISVAX.PORT.AC.UK
    or
  SIS3147@SISVAX.PORT.AC.UK

Or send us anything you've written........
```

1.86 gfxindex

```
                   These are all the GFX library commands:


                AGAFillPalette

                FillPalette

                PalAdjust

                PaletteInfo

                =PalRed

                =PalGreen

                =PalBlue

                =AGAPalRed

                =AGAPalGreen
```

```
                =AGAPalBlue
```

1.87 gfx_paletteinfo

```
Statement: PaletteInfo
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: PaletteInfo Palette#

This command is used to specify the palette object that all palette
interrogations should look at.  The majority of the commands use this
palette object as the source for their data, e.g. PalRed(1) will look at
the red value of colour 1 of the palette last used in a PaletteInfo
command.
```

1.88 @{fg

```
------------------------------------------------------------------------------
Modes :  Amiga/Blitz Syntax:  r.w=PalRed (Colour#)

This command is used to get the red value of colour number Colour#. You
should use the PaletteInfo command to specify what palette this command
takes its information from.

The value returned will be from 0 to 15
```

1.89 gfx_palgreen

```
Function: PalGreen
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: r.w=PalGreen (Colour#)

This command is used to get the green value of colour number Colour#.
You should use the PaletteInfo command to specify what palette this
command takes its information from.

The value returned will be from 0 to 15
```

1.90 gfx_palblue

```
Function: PalBlue
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: b.w=PalBlue (Colour#)
```

```
This command is used to get the blue value of colour number Colour#. You
should use the PaletteInfo command to specify what palette this command
takes its information from.

The value returned will be from 0 to 15
```

1.91 gfx_agapalred

```
Function: AGAPalRed
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: r.w=AGAPalRed (Colour#)

This command is used to get the red value of colour number Colour#. You
should use the PaletteInfo command to specify what palette this command
takes its information from.

The value returned will be from 0 to 255, this number of shades, though,
can only be displayed on an AGA machine.
```

1.92 gfx_agapalgreen

```
Function: AGAPalGreen
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: g.w=AGAPalGreen (Colour#)

This command is used to get the green value of colour number Colour#.
You should use the PaletteInfo command to specify what palette this
command takes its information from.

The value returned will be from 0 to 255, this number of shades, though,
can only be displayed on an AGA machine.
```

1.93 gfx_agapalblue

```
Function: AGAPalBlue
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: b.w=AGAPalBlue (Colour#)

This command is used to get the blue value of colour number Colour#. You
should use the PaletteInfo command to specify what palette this command
takes its information from.

The value returned will be from 0 to 255, this number of shades, though,
can only be displayed on an AGA machine.
```

1.94 gfx_paladjust

```
Statement: PalAdjust
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: PalAdjust dest_palette#,ration.q[,start_col,end_col]

This command is used to multiple all the colours, or a range of colours,
in a palette object, by a ratio.  The dest_palette# arguement is used to
give a destination for the adjusted colour information.  This
destination should be a pre-reserved palette and should be AT LEAST as
big and the source palette.  The source palette is taken as being the
palette last used in the PaletteInfo command.

The ratio should be given as either a quick value or a float and should
be below one for a fade or above to lighten a palette.  If you give a
ratio of 1 then a palette copy will occur.

The optional start and end parameters let you specify the range of
colours to adjust.  Only this range of colours, though, will be adjusted
and stored in the destination palette.
```

1.95 gfx_fillpalette

```
Statement: FillPalette
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FillPalette palette#,r,g,b[start_col,end_col]

This command lets you fill a given palette object with specific r,g,b
values.  The values given should be between 0 to and 15.  Optionally,
you can give start and end colour numbers to set a range for the fill.
You should be careful, though, because when you specify a range, no
checking is done (at the moment) to make sure that you don't exceed
the colour limit of the palette.

You should note that this command does not work on the palette last
PaletteInfo'ed.
```

1.96 gfx_agafillpalette

```
                Statement: AGAFillPalette
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: AGAFillPalette palette#,r,g,b[start_col,end_col]

This command is identical to
                 FillPalette
                 except that it lets
you specify AGA shade values for the r,g,b parameters.

See
```

FillPalette
for more information.

1.97 fnsmain

FNS Library v0.992
==========

By Stephen McNamara
(c)1994 Reflective Images

FNS COMMANDS
This Blitz2 library prints proportional fonts in either Amiga or Blitz
mode. It uses my own (rather primitive) font file format, details of which can be found at the end of this text file. Fonts can be upto 64 pixels wide and any height (although the font editor is limited to 64 pixels at the present moment). Fonts can be output in upto 256 colours (AGA!) and in the following ways: bold, centred, underlined, right-aligned or just standard left-aligned.

Note: a default font (PERSONAL.8) is built into this library and can be used by simply using font number 0. You do not have to install this font, it is automatically available for your use. A second point is to make is that the library is set up with a clipping rectangle of 0,0 to 0,0. Thus you have to use either FNSClip, FNSClipOutput or FNSOutput (with the optional clip parameter) to set the clipping rectangle before you try to print anything.

Please feel free to critisise (or praise!) this library, send me anything you want to say about it at:

SIS3149@SISVAX.SIS.PORT.AC.UK

Or send me anything you've written.....

1.98 fnsindex

These are all the FNS library commands:

InstallFNS

=FNSHeight

RemoveFNS

=FNSLength

```
                FNSClip

                =FNSLoad

                FNSClipOutput

                =FNSSlot

                FNSSetTab

                =FNSUnderline

                FNSInk

                =FNSVersion

                FNSOrigin

                =FNSWidth

                FNSOutput

                FNSPrefs

                FNSPrint

                FNSUnLoad

                FNS font format
               Note: All return values will be words except when using InstallFNS ↩
                   and
     FNSVersion.
```

1.99 fns_format

```
FNS Font file format:
=====================

Header: 256 bytes.
  0-3   : 'FNS.' - file identifier - looked for by InstallFNS
  4-5   : height of font (#word)
  6-7   : width of font in multiples of 16 (#word)
  8-9   : underline position (offset from top of font, #word)
  10-11 : size of data for each font character
    [ (WIDTH/8) * height ]
  32-255: byte giving widths of each character in the font.
    These bytes doesn't really hold the width, rather
    they hold the value to add to the X position of the
    character to get to the position to print the next
    character at (!).

  256-EOF:character data starting at ASCII 32 (space)
```

1.100 fns_settab

```
Statement: FNSSetTab
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSSetTab tab_width

Description:
Use this command to set the tab spacing used when printing. The value
given should be the spacing IN pixels.
```

1.101 fns_load

```
                Function: FNSLoad
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: suc=FNSLoad (filename$,font#)

Description:
This command is used to load a font from disk and automatically install
it for use by the FNS commands.  Filename$ should be the full name of
the file to load (path$+file$) and font# should be 0<= and >=15.  This
command returns a value of -1 for failure or the font number the font
was installed as (see InstallFNS).  A failure could either be a load
error or an installation error.

You should make sure that the file you load IS an FNS font file.

IMPORTANT NOTE: to use this command, you must have
our
                 FUNC
                library installed on your copy of Blitz2.
Running it without this library could, and probably will, cause a major
crash of your computer.

Also note that if you do an ERASEALL (this is a
                 FUNC
                library command for
erasing banks), you will DELETE your font from memory!
```

1.102 fns_unload

```
                Statement: FNSUnLoad
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSUnLoad font#

This command is used to remove a font installed with the FNSLoad
command. When this command runs it automatically removes the font
entry in the FNS commands and deletes the memory that the font file is
held in. There is no need to do this at the end of a program as the
```

```
                FUNC
               library automatically frees up all allocated
memory.
```

1.103 fns_slot

```
Function: FNSSlot
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: address.l=FNSSlot

Steve: this command was not in the doc file.
```

1.104 fns_installfns

```
                Function: InstallFNS
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: font_num.b=InstallFNS(font_num.b,address.l)

This is used to install a font so that it is available for use by
the output routines.  Font_num should be a number >=0 and <=15,
address should be the address in memory of the FNS font file.
This function will check that the address given does contain a FNS
font (it will look for the header 'FNS.'), if it cannot find the font
or something else goes wrong it will return a 0 to you, otherwise it
will return the number the font was installed as.

Note: The font number you give is automatically ANDED with $F when you
      call this function, thus if you supply a number greater that 15
      you could actually overwrite a previously installed font.

See:
                RemoveFNS
```

1.105 fns_removefns

```
                Statement: RemoveFNS
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: RemoveFNS font#

This command simply removes an installed font from the list of font
held internally by the FNS routines.  There is no real need to remove
fonts as installing fonts takes up no memory, except of course the
actual font data.  You do not need to remove FNS fonts before ending a
program.

See:
```

InstallFNS

1.106 fns_print

```
                Statement: FNSPrint
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSPrint font_num.b,x.w,y.w,a$/string_address
        [,preferences,colour]

This command prints the string a$ in an FNS font at the position X,Y.
Font_num is the number of a previously installed FNS font, the output
of this command is sent to the current FNS bitmap (see FNSOutput). You
can setting a drawing rectangle on the currently used bitmap to limit
the output of the font - see FNSClip for more info.

Instead of a string, though, you can give the address of a null
terminated string in memory.  Also, you can change the colour that text
is being output in in the current string by putting the character ASCII 1
followed by a byte value from 0-255 specifying the colour to change to.

The optional parameters are for controlling how the text is output.
They automatically overide the default setting but are not permanent,
i.e. the default output style and colour are restored after the line
has been output.  Use FNSInk and FNSPrefs to set the default font
output mode.

  See:
                FNSOuput
               ,
                FNSInk
               ,
                FNSPrefs
               ,
                FNSOrigin
               , FNSClip
```

1.107 fns_output

```
                Statement: FNSOutput
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSOutput bitmap#[,clip_update]

This command selects a bitmap for use by the FNS routines, the bitmap
must be a previously reserved Blitz 2 bitmap object.  After this
command all FNS font printing will occur on the selected bitmap.  The
optional parameter allows you to update the clipping rectangle for
output at the same time as setting the output bitmap.  Setting
clip_update to a non-zero value will cause the clipping area to
automatically be set to the dimensions of the selected bitmap.
```

```
NOTE:
-----
This command MUST be used before you attempt to use FNSPrint.
The maximum depth of the bitmap for printing is 8 bitplanes since this
is all Blitz 2 currently supports.

See:
                 FNSClip
                ,
                 FNSClipOutput
```

1.108 fns_ink

```
                Statement: FNSInk
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSInk colour#

This sets the output colour for the FNS font drawing routines. The
number range is dependant on the depth of the destination bitmap, the
max posible range, though, is limited to 0 to 255 colours.  The FNS
output routines will attempt to draw in all the bitplanes of the
selected bitmap, any extra bits in the ink colour will be ignored.

See:
                 FNSPrefs
```

1.109 fns_prefs

```
                Statement: FNSPrefs
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSInk preferences[,colour#]

This sets the output prefs for the FNS font drawing routines but at
the same time also sets the colour for the FNS routines (optional).
At the moment the following options are available, the bits of the
preferences byte are used to select the different options:

                            bit 0: Centred text
                            bit 1: Bold text
          bit 2: Underline
          bit 3: Right aligned

See:
                 FNSInk
                ,
                 FNSPrint
                , FNSLength
```

1.110 fns_height

```
                Function: FNSHeight
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: height.w=FNSHeight(font_num)

This routine returns the height of a previously installed FNS font.
Font_num should be >=0 and <=15.

See:
                 FNSUnderline
                ,
                 FNSWidth
```

1.111 fns_underline

```
                Function: FNSUnderline
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: under_pos=FNSUnderline(font_num)

This routine returns the underline position of the selected FNS font.
Font_num should be >=0 and <=15.

See:
                 FNSHeight
                ,
                 FNSWidth
```

1.112 fns_width

```
                Function: FNSWidth
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: width.w=FNSWidth(font_num)

This routine returns the width in multiples of 16 of the selected FNS
font.  Font_num should be >=0 and <=15.

See:
                 FNSHeight
                ,
                 FNSUnderline
```

1.113 fns_clip

```
                Statement: FNSClip
------------------------------------------------------------------------
```

```
Modes : Amiga/Blitz
Syntax: FNSClip x1,y1,x2,y2

This command is used to limit the output of the FNSPrint command.  The
co-ordinates given should describe a rectangle that is to be used to
clip the output.  This rectangle can be thought of as a window on the
bitmap - no printing can occur outside of the window.
X1,Y1 are the top left corner of the clipping rectangle and X2,Y2 are
the bottom right corner.  Please note that both X co-ordinates should be
multiples of 16 and that X2 should be the heightest multiple of 16 that
you do not wish output to occur at.  Thus if your bitmap is 320x256 then
you would use the following to set the clipping rectangle to the full
bitmap:
    FNSClip 0,0,320,256

See:
                 FNSClipOutput
                ,
                 FNSOutput
```

1.114 fns_clipoutput

```
                Statement: FNSClipOutput
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSClipOutput

This command is used to quickly set the clipping rectangle for the FNS
commands to the full size of a bitmap.

See:
                 FNSClip
                ,
                 FNSOutput
```

1.115 fns_origin

```
Statement: FNSOrigin
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FNSOrigin [x,y]

This command is used to set an origin co-ordinate for printing output.
Whenever you use FNSPrint, the origin co-ordinates are added (as words)
to the co-ordinates you give for output.  I.e. setting the origin at
100,0 and printing at co-ordinates 0,0 will cause the output to be at
100,0.

Using this command without any parameters will cause the origin to
be reset to the position 0,0.
Note: This command does not affect the use of the FNSClip command.
```

1.116 fns_lenght

```
                Function: FNSLength
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: a=FNSLength (font#,a$[,prefs])

This command is equivalent of the basic command a=len(a$) except that
it returns the x size, in pixels, of the string if it were to be printed
in the font font#.  The optional preferences parameter allows you to
adjust the output of the string, if you specify no preferences then this
function will use the previously selected preferences to calculate the
string length.  Using preferences allows you to account for things like
bold text output.

See:
                 FNSPrefs
```

1.117 fns_version

```
Function: FNSVersion
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: a.q=FNSVersion

This command allows you to test the version number of the FNS library
that your program is being compiled with.  It returns a quick float
value and so you should use a quick float variable for the answer.  This
doc file was written for version 0.991 of the library.


FNS Font file format:
=====================

Header: 256 bytes.
  0-3   : 'FNS.' - file identifier - looked for by InstallFNS
  4-5   : height of font (#word)
  6-7   : width of font in multiples of 16 (#word)
  8-9   : underline position (offset from top of font, #word)
  10-11 : size of data for each font character
    [ (WIDTH/8) * height ]
  32-255: byte giving widths of each character in the font.
    These bytes doesn't really hold the width, rather
    they hold the value to add to the X position of the
    character to get to the position to print the next
    character at (!).

  256-EOF:character data starting at ASCII 32 (space)
```

1.118 funcmain

```
                  Func/AMOS Library v1.0
=================

By Steven Matty
   ©1994 Reflective Images


               FUNC COMMANDS
              This library was written primarily to emulate the functions that ↩
                 were
present in AM*S but not in Blitz Basic 2. It began life as a load of
Blitz Statements but was then converted to high speed 680x0. The library
will continually be expanded upon and free updates will be sent on
request. If you decide to use any of the function please give me a
little cred, not a lot, just something. Anyway, enough of this
baloney....on with the command list.
```

1.119 funcindex

These are all the FUNC library commands:

CacheOff

=KeyCode

CopyByte

=Length

CopyWord

=Lisa

CopyLong

=MakeDir

Erase

=Max

EraseAll

=MemFree

FillMem

=Min

NextBank

=PLoad

```
                Reboot

                =Rename

                ResetTimer

                =Reserve

                =BLoad

                =CludgeShapes

                =BSave

                =CludgeSound

                =FileSize

                =Start

                =XOR

                =Timer
               ********************************* NOTE  ↩
                  ************************************
* VALID BANKS RANGE FROM 0-49 INCLUSIVE. DO NOT USE A VALUE GREATER THAN 49 *
* OR IT WILL BE INTERPRETED AS AN ADDRESS RATHER THAN A BANKNUMBER          *
*****************************************************************************
```

1.120 func_resettimer

```
Statement: ResetTimer
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : ResetTimer

This will recent the CIA timer to 0.
```

1.121 func_cludgeshapes

```
Statement/Function : CludgeShapes
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : [success]=CludgeShapes(shape#,numshapes,address)

This allows the creation of shapes through INCBIN statements. It
allocates chip memory for each shape and copies the data into this.
It does the same as LoadShapes except it grabs shapes from memory.
```

1.122 func_cludgesound

```
Statement/Function : CludgeSound
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : [success]=CludgeSound(sound#,address)

This does that same for CludgeShapes but works on only 1 sound at a time

NOTE: Looped sounds are not currently supported! The sound must be a valid
8SVX sample.
```

1.123 func_reserve

```
Function: Reserve
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : success=Reserve(banknumber,length)

  This will attempt to reserve <length> bytes of memory. If succesfull,
it will return the address of the bank. If unsuccessfull, 0 is returned.
Banks are limited by the Compiler Options Menu.
```

1.124 func_erase

```
Statement: Erase
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : Erase(banknumber)

The Erase command will erase the specified memory bank.
```

1.125 func_eraseall

```
Statement: EraseAll
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : EraseAll

This command will erase ALL allocated memory banks.
```

1.126 func_bload

```
Function: BLoad
------------------------------------------------------------------------------
Mode   : Amiga
Syntax : success=BLoad(filename$,bank/address[,length,offset])
```

```
If bank is specified, then the file is loaded into that bank. If address
is specified then it is loaded to the address. Valid banks are 0-49.
If the bank does not exist, Blitz will reserve a bank for you.
If the bank does exist, Blitz will erase the bank from memory, and
allocate a new one.

The return result is -1 for success, or 0 for failure  (not enough RAM,
file not exist). If offset is specified, then <length> bytes will be
read from the specified offset position in the file.
```

1.127 func_pload

```
Function: PLoad
------------------------------------------------------------------------
Mode   : Amiga
Syntax : success=PLoad(filename$,bank/address)

This will attempt to load the executable file to the specified address.
-1 is success, 0 is failure.
```

1.128 func_bsave

```
Function: BSave
------------------------------------------------------------------------
Mode   : Amiga
Syntax : success=BSave(filename$,bank/address,length)

This will save <length> bytes at bank/address to the file. Return result
is -1 for success, 0 for failure. If length > bank length then the
length of the bank is saved instead. If 0 is specified, the entire bank
is saved.
```

1.129 func_start

```
Function: Start
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : start_address.l=Start(banknumber.b)

This will return the start address of the specified bank. (0=no bank)
```

1.130 func_length

```
Function: Length
------------------------------------------------------------------------
Mode   : Amiga/Blitz
```

```
Syntax : length_of_bank.l=Length(banknumber.b)

This will return the length of the specified bank in bytes. (0=No bank)
```

1.131 func_memfree

```
Function: MemFree
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : bytes.l=MemFree

This will return the total amount of Public Free RAM available to the
system.
```

1.132 func_nextbank

```
Function: NextBank
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : bank.b=NextBank

This will return the number of the first available bank (-1 if none
free).
```

1.133 func_fillmem

```
Statement: FillMem
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : FillMem(address.l,length.l,value.b)

This will fill 'length' bytes starting from the specified address with
'value'.
```

1.134 func_copybyte

```
Statement: CopyByte
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : CopyByte(source.l,dest.l,num.l)

This will copy <num> bytes from <source> to <dest>
```

1.135 func_copyword

```
Statement: CopyWord
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : CopyByte(source.l,dest.l,num.l)

This will copy <num> words from <source> to <dest>
```

1.136 func_copylong

```
Statement: CopyLong
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : CopyByte(source.l,dest.l,num.l)

This will copy <num> longwords from <source> to <dest>
```

1.137 func_makedir

```
Function: MakeDir
------------------------------------------------------------------------------
Mode   : Amiga
Syntax : success=MakeDir(name$)

This function attempts to create a directory called <name$>
If it is unsuccessfull, 0 is returned else -1 is returned.
```

1.138 func_rename

```
Function: Rename
------------------------------------------------------------------------------
Mode   : Amiga
Syntax : success=Rename(source$,dest$)

This attempts to rename the file <source$> to <dest$>

NOTE: It is not possible to rename across devices. -1 is returned if
successfull, else 0.
```

1.139 func_timer

```
Function: Timer
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : t.l=Timer

This will return the number of 50ths of a second since startup.
```

1.140 func_lisa

```
Function: Lisa
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : chipver=Lisa

   This will return the current Lisa chip version :

    $00 for OCS Denise
    $F7 for ECS Denise
    $F8 for AGA Lisa
```

1.141 func_reboot

```
Statement: Reboot
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : Reboot

This will perform a cold reboot
```

1.142 func_filesize

```
Function: FileSize
------------------------------------------------------------------------------
Mode   : Amiga
Syntax : size.l=FileSize(filename$)

This return the length (in bytes) of the file.
```

1.143 func_cacheoff

```
Statement: CacheOff
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : Cache Off

This will turn off the instruction cache of the CPU.
```

1.144 func_xor

```
Function: XOR
------------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : x.l=XOR(x.l,y.l)
```

```
This will perform an Exclusive-Or operation between X and Y and put the
result back into X
e.g

    x=XOR(%101,%100)

Will place %001 into X (%101 XOR %100 = %001)
```

1.145 func_max

```
Function: Max/Min
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : value=Max(first_var,second_var)
         value=Min(first_var,second_var)

This will compare both values and return either the Higher of the values
(Max) or the Lower (Min). This currently supports INTEGERs only.
```

1.146 func_keycode

```
Function: KeyCode
------------------------------------------------------------------------
Mode   : Amiga/Blitz
Syntax : keycode=KeyCode

This will return the status of the keyboard in the form of a keycode.
You will need to experiment to find out the desired keycode for
a particular key.

This merely peeks address $bfec01 and returns the value found.
```

1.147 fxmain

```
                    Reflective Images Effects Library
      ================================

    By Stephen McNamara, with help from Steve Matty
         (c)1994 Reflective Images

                 FX COMMANDS
                Note: The library has had a lot of the commands inside it expanded ↩
                    so
that they work on any size bitmap.  At the moment the following, though,
will only work on lorez bitmaps: ZoomX8, Derez and ZoomXY

None of the commands in this library use the blitter chip, any blitter
source code that anybody has please send to me.
```

1.148 fxindex

```
                    Here are all the FX commands:

Command list:

                ChunkyToPlanar (Slow)

                ClearBitmap

                Derez

                FadeInBitmap

                InitZoomXY

                PlanarToChunky (Slow)

                ReduceX2

                ZoomXY

                ZoomX2

                ZoomX4

                ZoomX8

                =ADDValue(bitmap#,x,y)
               No instructions for the planar<>chunky commands since their not  ↩
                  really
that useful at the moment.  If anybody has some working code thats good
then.....................
```

1.149 fx_planar

```
No instructions for the planar<>chunky commands since their not really
that useful at the moment.  What I'm going to try and do is put some
faster conversion routines in this library to do the jobs of these
commands.
```

1.150 fx_fadeinbm

```
                Statement: FadeInBitmap
------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: FadeInBitmap source#,dest#,delay[,offset1,offset2,height]

This is used to make a low rez, any height, bitmap appear on another
one in a nice way.  Source# and dest# should be bitmap object numbers
and delay is the 'slow-down' value for the fade.  This is necessary
because this routine works very fast - at full speed it looks just like
```

a slow screen copy. You should note that the delay is taken as being a word, thus don't pass 0 or you'll actually get a delay of 65535. This routine will adjust itself to take into account the depth of the bitmap,

WARNING: the depth of the destination bitmap should be AT LEAST as big as the depth of the source# bitmap because the depth of the fade is taken from the source# bitmap.

The optional parameters in this command allow you to set respectively: the source bitmap y offset, the destination bitmap y offset and the height of the fade (in pixels). If these parameters are left out then the fade automatically occurs across the full size of the bitmap.

See:
ClearBitmap

1.151 fx_clearbm

Statement: ClearBitmap

Modes : Amiga/Blitz
Syntax: ClearBitmap source#,delay[,offset,height]

This is used to clear a low res, any height, bitmap in a very pleasant way. The parameters are the same as for FadeInBitmap except that only one bitmap is needed. The delay parameter i used for the same reason as in FadeInBitmap - to slow down the effect. The optional parameters allow you to set a y start value for the clear and the height (in pixels) of the clear.

See:
FadeInBitmap

1.152 fx_zoom2

Statement: ZoomX2

Modes : Amiga/Blitz
Syntax: ZoomX2 source#,dest#,add_source,add_dest,width,height

This command does a very fast X2 zoom. It works with two bitmaps - one source and one dest (note: these can be the same bitmap but you should be careful that the zoom is not done over the source data). The two parameters add_source and add_dest allow you to specify the position of the start of the zoom, they specified as byte offsets from the top left corner of the bitmaps (byte 0). These values can be calculated by the following method:

add_source=(Y x BITMAP_WIDTH (in bytes) + (X / 8)

or by using the built in command ADDValue. Width and height are both specified in pixels.

```
NOTE: There is no clipping on this command - be careful not to zoom off
      the edges of bitmaps. you can zoom from a bitmap to a different
      size bitmap BUT the destination bitmap must be as deep as the
      source and big enough to hold the zoomed data.

See:
                 ZoomX4
                ,
                 ZoomX8
                ,
                 ADDValue
```

1.153 fx_zoom4

```
                Statement: ZoomX4
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ZoomX4 source#,dest#,add_source,add_dest,width,height

This is exactly the same as ZoomX2 except that a times 4 zoom is done
by this command.

Note: You can zoom from a bitmap to a different size bitmap BUT the
        destination bitmap must be as deep as the source and big enough
  to hold the zoomed data.

See:
                 ZoomX2
                ,
                 ADDValue
```

1.154 fx_zoom8

```
                Statement: ZoomX8
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ZoomX8 source#,dest#,add_source,add_dest,width,height

This is exactly the same as ZoomX2 except that a times 8 zoom is done
by this command

See:
                 ZoomX2
                ,
                 ADDValue
```

1.155 fx_addvalue

```
                Function: ADDValue
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: addval.w=ADDValue(bitmap#,x,y)

This function can be used the calculate the add_source and add_dest
values used in all the zoom commands.  Just give the bitmap number, x
co-ordinate and the y co-ordinate and you'll get an answer back that can
be used straight in the ZoomXn commands.

See:
                 ZoomX2
                ,
                 ZoomX4
                ,
                 ZoomX8
                ,
                 ZoomXY
```

1.156 fx_initzoomxy

```
                Statement: InitZoomXY
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: InitZoomXY source#,dest#,add_source,add_dest

This command initialises the ZoomXY routine to the bitmaps you want it
to work on.  You MUST use this routine before calling ZoomXY.  The
parameters are the same as the first four parameter for the ZoomXn
commands - source and dest bitmaps and add_source/dest values.

See:
                 ZoomXY
```

1.157 fx_zoomxy

```
                Statement: ZoomXY
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ZoomXY xzoom_value,yzoom_value,height

This command does a zoom based on the values you give it.  You should
note, though, that zoom values should be integer values (no fractional
part).  The height is the height in pixels that the source data should
be zomed to.  Please note that this command is different to the other
zoom commands in that the output of it is clipped to fit inside 320
pixels.

This command should only be used after InitZoomXY has been called.
This routine has an extra feature in that if you give both zoom values
as 1 then a bitmap copy is done from the source to the dest using the
```

offsets given and the height.

See:
InitZoomXY

1.158 fx_derez

Statement: Derez

Modes : Amiga/Blitz
Syntax: Derez source#,dest#,add_source,add_dest,derez_value,height

This command is used to derez a low resolution bitmap onto another one.
The bitmaps are source# and dest#, add_source and add_dest are used to
control the start position of the derez (see ZoomX2 and ADDValue to see
how these are calculated). The derez value if obviously the amount that
each pixel will be derezed to in both the x and y directions, the height
is the height of the derez - the derez is clipped to fit inside this in
the y direction and inside 320 pixels in the x direction.
This routine has an extra feature in that if you give derez_value as 1
then a bitmap copy is done from the source to the dest using the offsets
given and the height.

1.159 fx_reducex2

Statement: ReduceX2

Modes : Amiga/Blitz
Syntax: ReduceX2 source#,dest#,add_source,add_dest,width,height

Description:
This command halves the given rectangle of one bitmap and pastes it onto
the destination bitmap. Width should be a multiple of 16, width and
height should describe a rectangular area that will be reduced (these
values should be in pixels).

See
ZoomX2
and other commands for more information about the
syntax of this command.

1.160 zj_main

Reflective Images Zone-Joystick Library v1.2
==========

By Stephen McNamara, original Joy Library by Steve Matty
(c)1994 Reflective Images

This library contains commands for setting up zones and testing the status

```
of the joysticks attached to the Amiga.

 ZONE-JOY COMMANDS
```

1.161 zonejoymain

```
                        Zone-Joy commands:

Command list:

                 FreeZoneTable
                         NewZoneTable

                 Setzone

                 UseZoneTable

                 ZoneInit

                 =AllFire

                 =JFire

                 =JHoriz

                 =JVert

                 =ZoneTableSize

                 =Zone

                 =ZoneTest

                 =ZoneTable
```

1.162 zj_ztsize

```
Function: ZoneTableSize
------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : size.l=ZoneTableSize

This function returns the size, in zones, of the current zonetable.  It
is equivalent of doing: size.l=peek.l(ZoneTable).
```

1.163 zj_uztable

```
Statement/Function: UseZoneTable
------------------------------------------------------------------------
Modes  : Amiga/Blitz
```

```
Syntax : UseZoneTable table#

This command is used to change the current zonetable to the selected
one. If used as a function, it will return TRUE for success or FALSE
for failure.

Valid zonetable numbers range from 0 to 15.
```

1.164 zj_nztable

```
Statement/Function: NewZoneTable
------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : NewZoneTable table#,size

This command will attempt to allocate a new zonetable with the given
table number. If the table already exists it will be deleted.  The
maximum size for a zonetable is 65536 zones. If used as a function, this
command will return FALSE for failure or TRUE for success.  You should
note that all zones are automatically reset in the new table and that
creating a table does not make it the current table, this must be done
with UseZoneTable.

Valid zonetable numbers range from 0 to 15.

IMPORTANT NOTE: you cannot define the size of zonetable 0.  You cannot
use this command to alter it in any way.
```

1.165 zj_fztable

```
Statement/Function: FreeZoneTable
------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : FreeZoneTable table#

This command is used to free a zonetable from memory.  If used as a
function, it will return TRUE or FALSE.  When successfully called, this
command will free the zonetable and change the currently used zonetable
to table number 0.

Valid zonetable numbers range from 0 to 15.

IMPORTANT NOTE: you cannot free zone table 0.
```

1.166 zj_ztable

```
Function: ZoneTable
------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : ad.l=ZoneTable
```

```
This function returns the address in memory of the zone information
storage area for the current zonetable. The zones are stored one after
the other, with each zone taking up 8 words (16 bytes) in the data area,
making a total size of 2048 bytes.  They are stored in the following
way:

    Rectangular:    +0: x1
        +2: y1
        +4: x2
        +6: y2

    Circular: +0: x1
        +2: y1
        +4: radius of zone
        +6: -1 <-- this is set to show that the
                   zone is circular.

    Undefined zone: +0: -1
        +2: -1
        +4: -1
        +6: -1

The first longword (4 bytes) of the zonetable is used to hold the size,
in zones, of the table (thus the true size of the zonetable is 4+number
of zones*8).
```

1.167 zj_zoneinit

```
Statement: ZoneInit
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : ZoneInit [zone_num]|[start_zone,end_zone]

This command is used to clear any zones currently set.  The optional
parameters allow you to select either a single zone or a range of zones
to reset.
```

1.168 zj_setzone

```
Statement: Setzone
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : Setzone zone#,x1,y1,radius
         Setzone zone#,x1,y1,x2,y2

This command lets you set up zones for testing.  The first version is
used when you want to set up a circular zone and the second when you
want a rectangular one.  With rectangular zones, x1,y1 should be the top
left corner of the rectangle and x2,y2 should be the bottom left.

Note: The max zone number is 255.
```

```
When you use this command, the zone number you give is ANDed with
256 so you should ensure that you give a number lower than 256 so
that previously defined zones don't get corrupted.
Zones can be defined in any order.
Circular zones are used in exactly the same way as rectangular
ones.
```

1.169 zj_zone

```
Function: Zone
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : a.w=Zone(x,y)

This command takes the co-ordinates x,y and checks to see if they are
inside any of the defined zones.  The zones are searched in order,
starting at 0 and going through to 255.  This command will return the
first zone that the co-ordinates were found to be inside, you should
note that both types of zones are tested (rectangular and circular).

This command returns either -1 for not inside a zone or the zone number.
```

1.170 zj_zonetest

```
Function: ZoneTest
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : a.w=ZoneTest(start_num[,end_num],x,y)

This command is the same as the Zone command except that it allows you
to select either one individual zone to test or a range of zones.  You
should, though, ensure that end_num if greater than start_num.

This command returns either -1 for not inside a zone or the zone number.
```

1.171 zj_zonetable

```
Function: ZoneTable
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : ad.l=ZoneTable

This function returns the address in memory of the zone information
storage area. The zones are stored one after the other, with each zone
taking up 8 words (16 bytes) in the data area, making a total size of
2048 bytes.  They are stored in the following way:

    Rectangular:    +0: x1
        +2: y1
        +4: x2
```

```
        +6: y2

    Circular: +0: x1
        +2: y1
        +4: radius of zone
        +6: -1 <-- this is set to show that the
                   zone is circular.

    Undefined zone: +0: -1
        +2: -1
        +4: -1
        +6: -1
```

1.172 zj_jfire

```
Function: JFire
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : jf.b=JFire(joy#)

This command tests the fire button status of the joystick joy#, where
joy# is between 1 and 4.  You should note that, as with all the joystick
commmands, joy#=1 refers to the Amiga's joystick port, joy#=2 refers to
the mouse port, and joy#=3 or joy#=4 refer to the four player adapter
 ports.

This command returns 0 for fire button not pressed or -1 for pressed
```

1.173 zj_jhoriz

```
Function: JHoriz
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : jh.b=JHoriz(joy#)

This command is used to test the horizontal direction of the selected
joystick.  It returns:

    0: No horizontal direction
          -1: Joystick left
          1: Joystick right
```

1.174 zj_jvert

```
Function: JVert
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : jv.b=JVert(joy#)

This command is used to test the vertical direction of the selected
```

```
joystick.  It returns:

    0: No vertical direction
       -1: Joystick up
       1: Joystick down
```

1.175 zj_allfire

```
Function: AllFire
------------------------------------------------------------------------------
Modes  : Amiga/Blitz
Syntax : af.b=AllFire [(bit_pattern)]

This command is used to test the fire button status of all four
joysticks.  It returns a byte with the first four bits giving the
joystick status, false=fire button not pressed, true=fire button
pressed. The following bits belong to joysticks:

    bit 0: joystick 1 (joystick port)
    bit 1: joystick 2 (mouse port)
    bit 2: joystick 3 (four player adaptor)
    bit 3: joystick 4 (four player adaptor)

The optional bit pattern can be used to restrict the testing of the fire
buttons.  If a bit in the pattern is clear (false) then the joystick it
belongs to will not have its fire button tested,

e.g. AllFire (%0011) will test joysticks 1 and 2 and return the
     result.  It will return false for joysticks 3 and 4.
```

1.176 ciatrackermain

```
                Library: neilsciatrackerlib #56
------------------------------------------------------------------------------
Author: Neil O'Rourke, 6 Victoria St, TAMWORTH, NSW 2340, AUSTRALIA

Overview:
Many thanks to Neil, from what I have seen on the net there are already
many BlitzUsers using this library to great success. I'm trying to fit the
example code on the disk as I type...


                 Quick Usage

                 Author's Doc
                                        The CIA Tracker commands:


                 BuildNoteTable

                 GetSongPosition
```

CheckTrackerEvent

GetTrackerEvent

CheckTrackerModuleID

GetTrackerInstrument

FreeTrackerModule

GetTrackerLocation

GetPatternPosition

GetTrackerName

GetSampleLength

GetTrackerNote

GetSampleLocation

GetTrackerNoteNumber

GetSampleName

GetTrackerSize

GetSongLength

GetTrackerVolume

InitTracker

SetTrackerModule

LoadTrackerModule

SetTrackerTempo

OldGetTrackerNoteNumber

StartTracker

PauseTracker

StartTrackerPatPos

PlayTrackerSample

StopTracker

RestartTracker

WaitTrackerEvent

SetDMAWait

SetSongPatternPosition

SetTrackerMask
Notes:
~~~~~~
Quite a number of these commands extract their data from the playroutine in real time; that is, around fifty times a second (depending upon the tempo). Therefore, the value your program receives could well be very different from what is actually happening in the song.

Disclaimer:
~~~~~~~~~~~
By installing this software on your system, you are agreeing that I have no liability as to the outcome of such use. If, for example, you use a command as documented and a floppy disk is ejected from your disk drive with such force that it severs your head from your neck, tough. Next time, duck.

1.177 cia_author

Author's Documentation: CIATracker.lib Documentation

Neil O'Rourke

Version 1.6 (24/6/94)

Introduction
~~~~~~~~~~~~
The standard soundtracker replay routines supplied with Blitz Basic 2 have many faults, which this library attempts to overcome. Some of the features are:

- Plays all ST/NT/PT songs that utilise either the VBLANK timing or the more recent CIA based timings
- Plays back correctly on 50/60Hz systems, running either PAL or NTSC
- Contains more specialised functions for advanced programmers
- Enables the programmer to syncronise graphics with their music

Credits:
~~~~~~~~
Original ProTracker playroutine by Amiga Freelancers, converted and enhanced for Blitz by Neil O'Rourke. Naggings from Roy, Jeff and Richard.

The 1.6 upgrade
~~~~~~~~~~~~~~~
This is a maintenance upgrade, with some subtle (and not so subtle) bugs fixed or noted.

LoadTrackerModule no longer crashes the machine if the name was invalid.

SetTrackerMask has been removed for the moment (this was causing the TrackerEvent system to foul up)
~~~~~~~~~~~~~~~

WaitTrackerEvent has a nasty tendancy to lock the machine up. Don't call this command, use While NOT CheckTrackerEvent:Wend to wait for an event if you must. WaitTrackerEvent currently sits on the VBLANK interrupt, however I think the problem is due to the sheer bulk of ciaTrackerLib getting in the way of checking. I think.

GetTrackerNoteNumber was found to be chewing up CPU time, and has been replaced by a new version that chews up 2K of ram extra.

I've found that if you have run errors enabled to bring up the requester, your module won't start sometimes. Don't know what to do about this, as I don't know what causes it.

1.178 cia_quickusage

Quick Usage:
~~~~~~~~~~~~
First you must set the DMAWait time with the SetDMAWait command. Then, enable all the channels with SetTrackerMask. Load the module you want with the LoadTrackerModule command, and then either StartTrackerModule it, or InitTracker/RestartTracker later on.

1.179 cia_ltmodule

Function: LoadTrackerModule

Syntax : success=LoadTrackerModule(TrackerModule#,FileName$)

Description:
Loads the named module into chip ram, ready for playing. This command can only be called in Amiga mode. success is a boolean return code (true). If the load fails for any reason, success returns the AmigaDOS error code.

Note that there is an implicit call to FreeTrackerModule for whatever module you are trying to load. However, if you want to load another module, don't try to load it on top of the existing one that is playing. Use another TrackerModule# (you have from 0 to 8). The results are unpredictable, and range from nothing to a system crash. We can't call StopTracker, because this will stop everything.

1.180 cia_stracker

Function: StartTracker

Syntax : success=StartTracker(TrackerModule#)

Description:
Starts to play the requested module, stopping any modules already playing, or restarts the current module, and returns true. Returns false if the
~~~~~~~~~~~~

module couldn't be started for some reason (like it isn't loaded).

1.181 cia_stoptracker

```
Statement: StopTracker
------------------------------------------------------------------------

Description:
Stops the current module
```

1.182 cia_sdwait

```
Statement: SetDMAWait
------------------------------------------------------------------------
Syntax   : SetDMAWait value

Description:
This sets the DMA Wait for your machine.  On a standard 7.14MHz 68000
based machine, the value is the default (300).  However, faster machines
can cause the replay routine to skip notes.  On a 25MHz 68030 machine, the
suggested value is 900.  Set this as low as possible so that you still
hear all the notes.  A future upgrade *may* do this automatically, but I
have no intention of implementing it at this stage, as I don't know what
DMAWait to set for different speed processors and version motherboards.

DMA wait is important.  Technically, when the replay routine loads the
chip registers with the information about the current note (location,
volume, pitch), a delay is needed to ensure that the chips actually get
the data, which happens on the next DMA slot.  Since the CPU can be
clocked independantly of the motherboard, we can't just delay by a set
amount.  How this problem has been solved is a busy wait that simply
loops around the number of times as specified by the DMAWait value.
A low value therefore lessens the load on the CPU but increases the
chances of missing notes while playing a song.  Too high a value can bog
the CPU down, and slow the song down as interrupts are missed.
```

1.183 cia_ftmodule

```
Statement: FreeTrackerModule
------------------------------------------------------------------------
Syntax   : FreeTrackerModule TrackerModule#

Description:
This frees a module loaded with LoadTrackerModule.  You cannot free a
module that has been set up with SetTrackerModule (see below), but there
is nothing to stop you trying.
```

1.184 cia_stmodule

```
Statement: SetTrackerModule
------------------------------------------------------------------------------
Syntax   : SetTrackerModule TrackerModule#,ModuleAddress

Description:
This sets an arbitary area of memory as a tracker module, useful if you
have BLoaded a file and want to hear if it is a module. Caution: a
non-module may crash the Amiga.
```

1.185 cia_gtsize

```
Functions: GetTrackerSize & GetTrackerLocation
------------------------------------------------------------------------------
Syntax   : trackerlength=GetTrackerSize(TrackerModule#)
           GetTrackerLocation (TrackerModule#)

Description:
Both these functions return information about the module that has been
loaded with LoadTrackerModule.  There should be no need to use this
information, and these commands are only included because they served a
purpose in debugging a long time ago, and to remove them would cause
problems with the Blitz tokens
```

1.186 cia_gtevent

```
Function: GetTrackerEvent
------------------------------------------------------------------------------
Syntax  : trackerevent=GetTrackerEvent

Description:
This command is a customised extension to the ProTracker replay routine.
A "TrackerEvent" occurs when the replay routine comes across a $8xx
command. This command is not defined in the command list, and many demos
(eg Jesus on E's) use it to trigger effects.  This command gets the most
recent TrackerEvent, so any program looking at this will have to compare
the current value to the value that triggered the current effect.
```

1.187 cia_ctevent

```
Function: CheckTrackerEvent
------------------------------------------------------------------------------
Syntax  : success=CheckTrackerEvent

Description:
This routine checks to see if a TrackerEvent has occured since the last
time the routine was called, and returns True if it has.  Use
GetTrackerEvent to determine what data the $8xx command had.
```

1.188 cia_wtevent

```
Statement: WaitTrackerEvent
------------------------------------------------------------------------------

** V1.6: DO NOT USE THIS COMMAND! **
```

1.189 cia_ctmid

```
Function: CheckTrackerModuleID
------------------------------------------------------------------------------
Syntax  : success=CheckTrackerModuleID(TrackerModule#)

Description:
This checks the module for the standard Pro/Noise/SoundTracker ID string
"M.K." (or "M!K!" in the case of a 100 pattern PT module), and returns
True if one of them is found.  This means that you can safely call
StartTracker.

Note that there is no 100% guarenteed way of determining what is a module
and what isn't.  Bit Arts, for example, remove the M.K. identifier to make
it harder to rip modules, so if you're writing a module ripping program,
you have to take this result with a grain of salt.
```

1.190 cia_gtvolume

```
Function: GetTrackerVolume
------------------------------------------------------------------------------
Syntax  : volume=GetTrackerVolume(TrackerChannel#)

Description:
Returns the last volume set by a $Cxx command for the named channel, which
are numbered from 0 to 3.  This is not the "real" volume of the sample that
is currently playing.
```

1.191 cia_gtnote

```
Function: GetTrackerNote
------------------------------------------------------------------------------
Syntax  : note=GetTrackerNote(TrackerChannel#)

Description:
Returns the note that the play routine has just played in the named
channel.  This command is really only useful for graphic bars or simple
syncronisation of graphics to the music, but for that purpose the
TrackerEvent commands are far more flexable.  Note that the value returned
is the period of the note.  You have to look up the note in a period table
to find out what was actually being played.
```

1.192 cia_sttempo

```
Statement: SetTrackerTempo
--------------------------------------------------------------------------------
Syntax   : SetTrackerTempo Tempo

Description:
Sets the tempo of the current song.  Note that a tempo command ($Fxx) will
override any value set by this command.  This command is really a stub to
the actual $Fxx command in the playroutine, and has all the features
associated with it.  Check your tracker docs for more details.
```

1.193 cia_gtinstrument

```
Function: GetTrackerInstrument
--------------------------------------------------------------------------------
Syntax  : instrument=GetTrackerInstrument(TrackerChannel#)

Description:
Gets the instrument that is playing in the channel.
```

1.194 cia_gpposition

```
Function: GetPatternPosition
--------------------------------------------------------------------------------
Syntax  : PatPos=GetPatternPosition

Description:
This returns the current position in the current pattern.
```

1.195 cia_gsongposition

```
Function: GetSongPosition
--------------------------------------------------------------------------------
Syntax  : SongPos=GetSongPosition

Description:
This returns the current pattern that is playing in the song
```

1.196 cia_sspposition

```
Statement: SetSongPatternPosition
--------------------------------------------------------------------------------
Syntax   : SetSongPatternPosition Pattern#,Position#

Description:
This command sets what pattern to play, and from what position.  Use this
```

while a song is playing to jump to another pattern (eg. a game over
music). Call StartTrackerPatPos() to start a module from scratch.

1.197 cia_gsonglength

```
Function: GetSongLength
------------------------------------------------------------------------
Syntax  : NumPatterns=GetSongLength

Description:
Returns the number of patterns in the current module.  Useful for displays
like in IntuiTracker, where the title bar of the window gives a display
that can be done like:

  NPrint GetSongLength,":",GetSongPosition
```

1.198 cia_stmask

```
Statement: SetTrackerMask
------------------------------------------------------------------------
Syntax   : SetTrackerMask Mask

** REMOVED IN V1.6 **
```

1.199 cia_ogtnnumber

```
Function: OldGetTrackerNoteNumber
------------------------------------------------------------------------
Syntax  : notenumber=OldGetTrackerNoteNumber(Channel#)

Description:
This returns the number of the note played on the specified channel, with
C-1 being note 1.  Of use really in creating "equalizer bars".

V1.6: This command has turned out to be a CPU-hog!  The new implementation
will consume a lot of memory but will be much faster.  When you load your
old programs, GetTracker... will be replaced by OldGetTracker..., so your
code will continue to work.
```

1.200 cia_stppos

```
Function: StartTrackerPatPos
------------------------------------------------------------------------
Syntax  : ret.l=StartTrackerPatPos(TrackerModule#,Pattern#,Position#)

This starts the named module at the requested pattern and position.  In
all other respects it is the same as StartTracker.
```

1.201 cia_prtracker

```
Statements: PauseTracker &  RestartTracker
------------------------------------------------------------------------------

Description:
These commands allow you to stop a tracker module are restart it at a later
time.
```

1.202 cia_ptsample

```
Statement: PlayTrackerSample
------------------------------------------------------------------------------
Syntax   : PlayTrackerSample Sample#,Period,Volume,Channel

Description:
Plays a sample through the channel.  The module must not be running.
```

1.203 cia_itracker

```
Statement: InitTracker
------------------------------------------------------------------------------
Syntax   : success=InitTracker(TrackerModule#)

Description:
Identical to StartTracker, except that the module doesn't start, but is
initialised.  Of use with the commands that use the current tracker
module. Use ReStartTracker to start playing.
```

1.204 cia_gslocation

```
Function: GetSampleLocation
------------------------------------------------------------------------------
Syntax  : location=GetSampleLocation(Sample#)

Description:
Returns the address in memory of the named sample in the current module.
```

1.205 cia_gslength

```
Function: GetSampleLength
------------------------------------------------------------------------------
Syntax  : length=GetSampleLength(Sample#)

Description:
Returns the length in words of the named sample in the current module.
Multiply by two to get the byte length.
```

1.206 cia_gsname

```
Function: GetSampleName
------------------------------------------------------------------------------
Syntax  : name$=GetSampleName(Sample#)

Description:
Returns the name of the sample in name$.
```

1.207 cia_gtname

```
Function: GetTrackerName
------------------------------------------------------------------------------
Syntax  : name$=GetTrackerName(TrackerModule#)

Description:
Returns the name of the module in name$
```

1.208 cia_bntable

```
Statement: BuildNoteTable
------------------------------------------------------------------------------

Description:
This command builds a note table for use with GetTrackerNoteNumber.  It
consumes 2K of memory for the look-up table.
```

1.209 cia_gtnnumber

```
Function: GetTrackerNoteNumber
------------------------------------------------------------------------------
Syntax  : notenumber=GetTrackerNoteNumber(Channel#)

Description:
This returns the number of the note played on the specified channel, with
C-1 being note 1.  Of use really in creating "equalizer bars".

For speed purposes, no error checking (like, has the note table been
built?) is done.
```

1.210 elmoremain

```
                There's not much to tell about the Elmore library, it's GREAT!

      Library name: ELMORELIB
         Written by: Richard T. Elmore
       Copyright: 1994 HeadSoft Software
```

```
Library number: 111


          Hardware Programming

          Math/Numeric functions

          Array functions

          Intuition Programming

          String Handling

          Library Programming
```

1.211 elmore_hardwareindex

```
                                        ELMORE HARDWARE LIBRARY
------------------------------------------------------------------------------


          ClickMouse

          ForceNTSC

          ForcePAL

          Freq

          Quiet

          ResetTimer

          VwaitPos

          =CheckAGA

          =ChipFree

          =Depth

          =FastFree

          =JoyC

          =LargestFree

          =Peekto$

          =Ticks
```

1.212 elmore_mathindex

```
                                                ELMORE MATH LIBRARY
------------------------------------------------------------------------------

                Rrandomize

                =Avg

                =Avg.L

                =Avg.Q

                =Largest

                =Largest.l

                =Largest.q

                =Rrnd

                =Smallest

                =Smallest.l

                =Smallest.q

                =Xor
```

1.213 elmore_intuitionindex

```
                      INTUITION PROGRAMMING
------------------------------------------------------------------------------

                Request

                ShowRequestors

                WaitFor

                =Activescreen

                =ActiveWindow

                =ScreenHeight

                =ScreenWidth
```

1.214 elmore_stringindex

```
                        STRING HANDELING
------------------------------------------------------------------------------

                    =Bin#

                    =CharCount

                    =Checksum

                    =Cipher$

                    =Hex#

                    =Null

                    =Repeats

                    =SearchBegin

                    =SearchBegin

                    =Space$
```

1.215 elmore_libraryindex

```
                        LIBRARY PROGRAMMING
------------------------------------------------------------------------------
These functions will return the base address of their
respective libraries, for advanced system programming.  Note that
register A6 will also be loaded with this address, to make programming
a bit easier for assembly routines.

                    CommoditieBase

                    DiskfontBase

                    DosBase

                    FFPBase

                    GraphicsBase

                    IconBase

                    IntuitionBase

                    RexxsysBase
```

1.216 elm_quiet

```
Statement: QUIET
------------------------------------------------------------------------------
Syntax: Quiet ChannelMask
Modes: Amiga or Blitz

This command will silence the sound channels specified by ChannelMask.
See the description for "Envelope" for more information on channelmasks.
```

1.217 elm_freq

```
Statement: FREQ
------------------------------------------------------------------------------
Syntax: Freq Channelmask,period
Modes: Amiga or Blitz

This command allows you to change the period, or pitch, of the currently
playing sound effect. Note that the lower the period, the higher the
frequency; Thus, a period of 100 would be very high-pitched, whereas a
period of 30000 would be low-pitched.
```

1.218 elm_ticks

```
                Function: TICKS
------------------------------------------------------------------------------
Syntax: Ticks
Modes: Amiga or Blitz

This function returns the number of "ticks" since the Amiga was switched
on, or since the last "RESETTIMER" command.  The unit of measurement is
1/60 of a second for NTSC machines, and 1/50 of a second for PAL
machines.

See Also:
                 ResetTimer
```

1.219 elm_resettimer

```
                Statement: RESETTIMER
------------------------------------------------------------------------------
Syntax: ResetTimer
Modes: Amiga or Blitz

Resets the Amiga's hardware timer to zero "ticks." Read the description
for
                 TICKS
                 for more information.
```

1.220 elm_joyc

```
Function: JOYC
------------------------------------------------------------------------------
Syntax: JoyC (Port)
Modes: Amiga or Blitz

This function works similarly to the JoyB() function, however it allows
you to read the second fire button on two-button joysticks.  It will
return a 1 if the normal fire button is pressed, a 2 if the second
button is pressed, or 3 if both buttons are pressed.  Otherwise, it will
return a zero (no buttons pressed.)
```

1.221 elm_vwaitpos

```
Statement: VWAITPOS
------------------------------------------------------------------------------
Syntax: VWaitPos RasterLine
Modes: Amiga or Blitz

This command is similar to VWAIT, except it allows you to wait for any
raster position, not just the top of the display.  This is useful for
interesting graphics effects.
```

1.222 elm_checkaga

```
Function: CHECKAGA
------------------------------------------------------------------------------
Syntax: CheckAGA
Modes: Amiga or Blitz

Returns 'TRUE' for AGA machines, otherwise returns 'FALSE.' Using
ExecVersion alone will not detect an AGA machine.  Kickstart version 39
can and does run on pre-AGA machines, such as the A3000, etc.
Therefore, this function is provided to allow you to accurately
determine if the AGA chipset is present.
```

1.223 elm_peekto

```
Function: PEEKTO$
------------------------------------------------------------------------------
Syntax: PeekTo$ (Address,byte)
Modes: Amiga or Blitz

PeekTo$() is similar to the Peek$() function, except you can specify
what terminator byte to use.  With Peek$() the terminator will always
be zero, but PeekTo$() will accept any byte value as a terminator.
```

1.224 elm_forcepal

```
Statement: FORCEPAL
------------------------------------------------------------------------------
Syntax: ForcePAL
Modes: Amiga or Blitz

This command switches the current screen from NTSC to PAL.
```

1.225 elm_forcentsc

```
Statement: FORCENTSC
------------------------------------------------------------------------------
Syntax: ForceNTSC
Modes: Amiga or Blitz

This command switches the current screen from PAL to NTSC.
```

1.226 elm_depth

```
Function: DEPTH
------------------------------------------------------------------------------
Syntax: Depth (Bitmap#)
Modes: Amiga or Blitz

This function returns the depth of the specified Blitz2 bitmap object.
```

1.227 elm_clickmouse

```
Statement: CLICKMOUSE
------------------------------------------------------------------------------
Syntax: ClickMouse
Modes: Amiga or Blitz

Similar to Mousewait, this command halts program execution until the
user clicks the mouse.  There must must be a separate mouseclick for
each CLICKMOUSE command, unlike Mousewait, which will continue through
without pausing if the left mouse button was already being pressed.

NOTE:  Avoid using this command in Amiga mode, as it seriously degrades
       multitasking.
```

1.228 elm_chipfree

```
                Function: CHIPFREE
------------------------------------------------------------------------
Syntax: ChipFree
Modes: Amiga or Blitz

This function will return the size, in bytes, of the largest block of
free CHIP memory in your system.

See Also:
                 FastFree
                ,
                 LargestFree
```

1.229 elm_fastfree

```
Function: FASTFREE
------------------------------------------------------------------------
Syntax: FastFree
Modes: Amiga or Blitz

This function returns the size of the largest block of FAST memory.
```

1.230 elm_largestfree

```
Function: LARGESTFREE
------------------------------------------------------------------------
Syntax: LargestFree
Modes: Amiga or Blitz

This function will return the size of the largest chunk of memory
available. This memory may be FAST or CHIP, depending on your system.
```

1.231 elm_xor

```
Function: XOR
------------------------------------------------------------------------
Syntax: Xor (expression,expression)
Modes: Amiga or Blitz

Returns Exclusive OR of two expressions This function returns the
"exclusive-OR" or the two supplied arguments. For example, Xor(255,170)
will return 85, and Xor(-1) will return 0.
```

1.232 elm_largestl

```
Function: LARGEST.L
------------------------------------------------------------------------------
Syntax: Largest.l (Long Integer1,Long Integer2)
Modes: Amiga or Blitz

This function will return the larger of the two supplied long integers.
For example, Largest.l(255,20045) would return 20045.
```

1.233 elm_smallestl

```
Function: SMALLEST.L
------------------------------------------------------------------------------
Syntax: Smallest.l (Long Integer1,Long Integer2)
Modes: Amiga or Blitz

This function will return the smaller of two supplied long integers.
For example, Smallest.l(-999,5) would return -999.
```

1.234 elm_largestq

```
                Function: LARGEST.Q
------------------------------------------------------------------------------
Syntax: Largest.q (Quick1,Quick2)
Modes: Amiga or Blitz

Identical to the function
                 Largest.l
                 except that
it accepts quick-type variables or expressions.
```

1.235 elm_smallestq

```
                Function: SMALLEST.Q
------------------------------------------------------------------------------
Syntax: Smallest.q (Quick1,Quick2)
Modes: Amiga or Blitz

Identical to
                 Smallest
                 but uses quick-types.
```

1.236 elm_largest

```
                Function: LARGEST
------------------------------------------------------------------------------
Syntax: Largest (Integer1,Integer2)
Modes: Amiga or Blitz
```

```
This is the fastest "Largest()" function.  Note that if passed floats
or quick-types, the fraction will be cut off.  See description for

                 Largest.l
                 and  Largest.q
LARGESTQ}.
```

1.237 elm_smallest

```
                Function: SMALLEST
------------------------------------------------------------------------
Syntax: Smallest (Integer1,Integer2)
Modes: Amiga or Blitz

Like
                 Smallest.l
                 and  Smallest.q
SMALLESTQ}, with less accuracy, but faster than the long-integer and
quick-type versions.
```

1.238 elm_avgl

```
Function: AVG.L
------------------------------------------------------------------------
Syntax: Avg.l (Long Integer 1,Long Integer 2)
Modes: Amiga or Blitz

This function will return the average of two long-integers (although
the fraction is cut off.)  Thus, Avg.l(5,15)=10, and Avg.l(1,2)=1.
(Since fractions will be cut off with this function, you may wish to
use the quick-type version of this function for more accuracy.)
```

1.239 elm_avgq

```
                Function: AVG.Q
------------------------------------------------------------------------
Syntax: Avg.q (Quick1,Quick2)
Modes: Amiga or Blitz

See the description for
                 Avg.l
                .
```

1.240 elm_avg

```
                Function: AVG
------------------------------------------------------------------------------
Syntax: Avg (Integer1,Integer2)
Modes: Amiga or Blitz

See the description for
                 Avg.l
                .
This version is the fastest Avg() function available.
```

1.241 elm_rrandomize

```
                Statement: RRANDOMIZE
------------------------------------------------------------------------------
Syntax: RRandomize Seed
Modes: Amiga or Blitz

Given a float-type expression or variable, RRandomize will "seed" the
reproducible random number generator.  The sequence of pseudo-random
numbers produced by
                 RRND
                 will be the same for each
seed given it. If you require trully random numbers, try "RRandomize
Ticks."
```

1.242 elm_rrnd

```
                Function: RRND
------------------------------------------------------------------------------
Syntax: RRnd (Low,High)
Modes: Amiga or Blitz

Given a range such as (1,6) this function will return a random number
based on the seed given it by
                 RRandomize
                These sets of "random" numbers can be repeated if you provide the  ↩
                   same
seed.  This can be useful in games, etc. so that using "RRandomize
Level#" and then using the RRnd() function to randomly draw the screen,
each time the player returns to that particular level, it will be the
same.
```

1.243 elmore_arrayindex

```
                              ELMORE ARRAY LIBRARY


Function: INDEX
------------------------------------------------------------------------------
```

```
Syntax: Index List()
Modes: Amiga or Blitz

Returns index from top of LIST This function will return the current
index number of the supplied List() array passed to it.  For example,
if the list pointer is currently at item 10 in the list, Index would
return 10.
```

1.244 elm_request

```
Statement or Function: REQUEST
------------------------------------------------------------------------
Syntax: Request (Title$,Text$,GadgetText$)
Modes: Amiga


This command is 2.0-specific.  If you're still using 1.3,
this command will be unavailable to you.

"Request" can be used as both a command or a function.  You may
provide an optional title (or "" for default window title) a string
of text (separated by pipes "|" for each line) and a string containing
text for gadgets within the requester.  (Separate with "|" if you
need more than one.)
Used as a command, it merely displays the requester on the current
screen and waits for the user to click a gadget.  As a function, it
will also return a number corresponding to the gadget selected.
The gadget on the right should be reserved for negative responses
such as "CANCEL" or "NO" and will always return zero.  Other gadgets
will return values in the order that they appear, beginning with
1 for the first gadget, 2 for the next, etc.
```

1.245 elm_activescreen

```
Function: ACTIVESCREEN
------------------------------------------------------------------------
Syntax: ActiveScreen
Modes: Amiga

This function returns ADDRESS of current Intuition screen.  This is
useful with many Intuition library commands, or to find out information
about the currently active screen.
```

1.246 elm_screenwidth

```
Function: SCREENWIDTH
------------------------------------------------------------------------
Syntax: ScreenWidth
Modes: Amiga
```

```
This function returns the pixelwidth of the currently active screen.
```

1.247 elm_screenheight

```
Function: SCREENHEIGHT
------------------------------------------------------------------------
Syntax: ScreenHeight
Modes: Amiga

This function returns the pixelheight of the active screen
```

1.248 elm_activewindow

```
Function: ACTIVEWINDOW
------------------------------------------------------------------------
Syntax: ActiveWindow
Modes: Amiga

This function returns the address of the current window.
This address is mainly used in conjunction with Intuition library
commands.
```

1.249 elm_waitfor

```
Statement or Function: WAITFOR
------------------------------------------------------------------------
Syntax: WaitFor (IDCMP Code)
Modes: Amiga

Similar to WaitEvent, WAITFOR puts the Amiga to "sleep" until a
specified IDCMP code wakes it up.  For example, WaitFor $400 would wait
until the user strikes a key, and WaitFor $8 would wait until the
"close" gadget of the current window was clicked on.  These IDCMP codes
are additive, so WaitFor $408 would wait until either the "close" gadget
was selected, or a key was pressed.  Refer to the section on "windows"
in the Blitz2 Reference Manual for more information on IDCMP codes.
```

1.250 elm_showreq

```
Statement: SHOWREQUESTERS
------------------------------------------------------------------------
Syntax: ShowRequesters OPTION>
Modes: Amiga or Blitz

OPTIONS:  0=Cancel all requesters
          1=Show requesters on Workbench Screen
          2=Direct requesters to current window
```

```
This command allows you to force system requesters like "Please insert
volume Foo in any drive" etc. to either be turned off, directed to the
workbench, or directed to the current window.  When requesters are
turned off, the system will behave as if the "CANCEL" gadget was
selected for each requester that would otherwise have been displayed.
Be sure to re-activate requesters before exiting your program!
```

1.251 elm_checksum

```
Function: CHECKSUM
----------------------------------------------------------------------------
Syntax: Checksum (String$)
Modes: Amiga or Blitz

Given a string, Checksum() will return a unique 32-bit integer as
a checksum, useful in situations such as serial transfers, etc. to
ensure both parties have the same data.
```

1.252 elm_charcount

```
Function: CHARCOUNT
----------------------------------------------------------------------------
Syntax: CharCount (String$,byte)
Modes: Amiga or Blitz

This function will return the number of occurances of a given byte
within a string.  For example, CharCount(text$,32) will count the
number of spaces in text$.
```

1.253 elm_searchbegin

```
Function: SEARCHBEGIN
----------------------------------------------------------------------------
Syntax: SearchBegin (String$,byte,# from Begin)
Modes: Amiga or Blitz

Similar to Instr(), SearchBegin will search the given string
for the specified byte.  For example, SearchBegin(a$,32,1)
will return the character position of the first space in a$,
while SearchBegin(a$,32,3) will return the position of the
third space.  If the byte is not found in the string, SearchBegin
will return a zero.
```

1.254 elm_searchend

```
Function: SEARCHEND
--------------------------------------------------------------------------------
Syntax: SearchEnd (String$,byte,# from End)
Modes: Amiga or Blitz

Like SearchBegin() (above) except it searches from the end of
the string to the front.  For example, SearchBegin(a$,asc("A"),2)
will return the character position of the second-from-last letter
"A" in the string 'a$.'
```

1.255 elm_cipher$

```
Function: CIPHER$
--------------------------------------------------------------------------------
Syntax: Cipher$ (String$)
Modes: Amiga or Blitz

The Cipher$() function will encrypt or decrypt a string passed to it.
This is especially handy if you don't want users "zapping" your
executeable or data files to read it's contents.  Note that Cipher$()
can only decrypt strings previously created with Cipher$().
```

1.256 elm_null

```
Function: NULL
--------------------------------------------------------------------------------
Syntax: Null (String$)
Modes: Amiga or Blitz

Many Amiga shared libraries (like the DOS library) require addresses
of null-terminated strings as arguments.  This function will return
a long-integer address of a null-terminated string in memory for such
commands.
```

1.257 elm_repeats

```
Function: REPEATS
--------------------------------------------------------------------------------
Syntax: Repeats (String$)
Modes: Amiga or Blitz

This function will return the number of repeated bytes at the
beginning of your string.  Thus, Repeats("...Test") would return 3,
while Repeats("Example") would return 1.  If the string is null,
Repeats() will return zero.
```

1.258 elm_space$

```
Function: SPACE$
--------------------------------------------------------------------------------
Syntax: SPACE$ (number of spaces)
Modes: Amiga or Blitz

This function is identical to the Space$ function in many other dialects
of BASIC.  It will return a string containing the desired number of
spaces, making it easier to align tables etc. to the screen or printer.
```

1.259 elm_bin#

```
Function: BIN#
--------------------------------------------------------------------------------
Syntax: Bin# (BinString$)
Modes: Amiga or Blitz

This function accepts binary value stored in a string and returns the
decimal value.
```

1.260 elm_hex#

```
Function: HEX#
--------------------------------------------------------------------------------
Syntax: Hex# (HexString$)
Modes: Amiga or Blitz

This function accepts hexadecimal value stored in a string and returns
the decimal value.
```

1.261 elm_intuibase

```
Function: INTUITIONBASE
--------------------------------------------------------------------------------
Syntax: IntuitionBase
Modes: Amiga or Blitz

Returns Intuition Library base
```

1.262 elm_dosbase

```
Function: DOSBASE
--------------------------------------------------------------------------------
Syntax: DosBase
Modes: Amiga or Blitz

Returns DOS Library base
```

1.263 elm_graphicsbase

```
Function: GRAPHICSBASE
--------------------------------------------------------------------------------
Syntax: GraphicsBase
Modes: Amiga or Blitz

Returns Graphics Library base
```

1.264 elm_ffpbase

```
Function: FFPBASE
--------------------------------------------------------------------------------
Syntax: FFPBase
Modes: Amiga or Blitz

Returns FFP Math Library base
```

1.265 elm_diskfontbase

```
Function: DISKFONTBASE
--------------------------------------------------------------------------------
Syntax: DiskFontBase
Modes: Amiga or Blitz

Returns DiskFont Library base
```

1.266 elm_commo

```
Function: COMMODITIESBASE
--------------------------------------------------------------------------------
Syntax: CommoditiesBase
Modes: Amiga or Blitz

Returns Commodities Library base
```

1.267 elm_iconbase

```
Function: ICONBASE
--------------------------------------------------------------------------------
Syntax: IconBase
Modes: Amiga or Blitz

Returns Icon Library base
```

1.268 elm_rexxbase

```
Function: REXXSYSBASE
------------------------------------------------------------------------------
Syntax: RexxSysBase
Modes: Amiga or Blitz

Returns RexxSys Library base
```

1.269 info

```
                              Blitz2 Guidefile Info
------------------------------------------------------------------------------

                    BlitzBasic is copyrighted to Acid Software
                    ==========================================

I created this file because I was sick of loading everytime the doc
files in the background. So I started to do some typing and block
cutting with the original doc files delivered with the libs.

I'm glad that these library's are created because they make Blitz2 a lot
better and more usefull.

            You can contact me on internet if needed: j.valks@hsbos.nl

Special thanks are going to Simon Armstrong of Acid Software for giving
me the BUM ascii files. Thanks!

                                    Written by:

                              Jurgen Valks (BlitzUser 418)
                                     Kerkeind 8a
                                5293 AB  Gemonde (NB)
                                   The Netherlands

This file is to big now! I gonna make a other file with commands and
syntax only...
```

1.270 bummain

```
                                                Choose a item:


                    Amiga Support library     (BUM 6)

                    Palette library Additions (BUM 5)

                    Anim library              (BUM 2)

                    Screen  library Additions (BUM 5)
```

```
Arexx library             (BUM 2)

Window  library Additions (BUM 5)

Console library           (BUM 6)

New Screen Flags          (BUM 4)

Crunch library            (BUM 6)

Elmore library            (BUM 6)

AGA Palette Handling      (BUM 4)

Locale library            (BUM 6)

3.0 Bitmap  Handling      (BUM 4)

Med library               (BUM 2)

New Gadget  Handling      (BUM 4)

Printer libary            (BUM 6)

SerialPort library        (BUM 2)

Date & Time Commands      (BUM 4)

The New Display library   (BUM 5)

Environment Commands      (BUM 4)

The New ASL library       (BUM 5)

New Drawing Commands      (BUM 4)

The GadTools library      (BUM 5)

Misc Additions            (BUM X)
```

1.271 bum_misc

```
                                        MISC ADDITIONS

    These are misc commands added in several BUM magazines:

        BlitColl

        Block

        Exists

        ILBMViewMode
```

```
LoadFont

LoadShape

ReMap

Runerrsoff

Runerrson

SetBPLCON0

ShapeGadget

ShowBitmap

SortList

SpriteMode

Vpos
```

1.272 bum_sortlist

```
Statement: SortList
------------------------------------------------------------------------------
Syntax: SortList Arrayname()

The SortList command is used to rearrange the order of elements in a
Blitz2 linked list. The order in which the items are sorted depends on
the first field of the linked list type which must be a single integer
word. Sorting criteria will be extended in future releases.
```

1.273 bum_loadfont

```
Statement: LoadFont
------------------------------------------------------------------------------
Syntax: LoadFont IntuiFont#,Fontname.font$,Y size [,style]

The LoadFont command has been extended with an optional style
parameter. The following constants may be combined:

#underlined=1
#bold=2
#italic=4
#extended=8 ;wider than normal
#colour=64  ;hmm use colour version I suppose
```

1.274 bum_spritemode

```
Statement: SpriteMode
------------------------------------------------------------------------------
Syntax: SpriteMode mode

For use with the capabilities of the new Display library SpriteMode is
used to define the width of sprites to be used in the program. The mode
values 0, 1 and 2 correspong to the widths 16, 32 and 64.
```

1.275 bum_exists

```
Function: Exists
------------------------------------------------------------------------------
Syntax: Exists (FileName$)

Exists actually returns the length of the file, if 0 the file either
does not exist or is empty or is perhaps not a file at all! Hmmm,
anyway the following poke turns off the "Please Insert Volume Blah:"
requester so you can use Exists to wait for disk changes:

     Poke.l Peek.l(Peek.l(4)+276)+184,-1
```

1.276 bum_runerrson

```
Statements: Runerrson & Runerrsoff
------------------------------------------------------------------------------
Syntax: Runerrson & Runerrsoff

These two new compiler directives are for enabling and disabling error
checking in different parts of the program, they override the settings
in Compiler Options.
```

1.277 bum_block

```
Statement: Block
------------------------------------------------------------------------------
Syntax: Block Shape#,X,Y
Modes: Amiga/Blitz

Description:
Block is an extremely fast version of the Blit command with some
restrictions. Block should only be used with shapes that are 16,32,48,
64... pixels wide and that are being blitted to an x position of 0,16,
32,48,64...

Note that the height and y destination of the shape are not limited by
the Block command. Block is intended for use with map type displays.
```

1.278 unnamed.1

```
Statement: LoadFont
------------------------------------------------------------------------------
Syntax: LoadFont IntuiFont#,Fontname.font$,Y Size
Modes: Amiga

Description:
LoadFont is used to load a font from the fonts: directory. Unlike
BlitzFonts any size IntuiFont can be used. The command WindowFont is
used to set text output to a certain IntuiFont in a particular Window.
```

1.279 bum_vpos

```
Function: VPos     (add to chapter 5)
------------------------------------------------------------------------------
Syntax: VPos
Modes: Amiga/Blitz

Description:
VPos returns the video's beam vertical position. Useful in both high-
speed animation where screen update may need to be synced to a certain
video beam position (not just the top of frame as with VWait) and for a
fast random nember generator in non frame-synced applications.
```

1.280 bum_animlib

```
                The Anim.lib

The following 4 commands allow the display of Animations in Blitz BASIC.
The Animation must be compatible with the DPaint 3 format, this method
uses long delta (type 2) compression and does not include any palette
changes.
                                The Anim Commands:


                 LoadAnim

                 InitAnim

                 NextFrame

                 Frames
                Anims in nature use a double buffered display, with the addition ↩
                   of the
ShowBitMap command to Blitz we can now display (play) Anims in both
Blitz and Amiga modes. An Anim consists of an initial frame which needs
to be displayed (rendered) using the InitAnim command, subsequent frames
are then played by using the NextFrame command. The Frames() function
returns the number of frames of an Anim. We have also extended the
LoadShape command to support Anim brushes. The following example loads
and plays an Anim on a standard Amiga (Intuition) Screen.
```

```
;
;play anim example
;
;anim file name could use f$=par$(1) to play anim from cli
f$="test.anim"
;open screen same resolution as animation

ILBMInfo f$
Screen 0,0,0,ILBMWidth,ILBMHeight,ILBMDepth,ILBMViewMode,"",1,2
ScreensBitMap 0,0

;an extra bitmap same size as screensbitmap for double buffering

BitMap 1,ILBMWidth,ILBMHeight,ILBMDepth

;load anim and set screen colours to same as animation

LoadAnim 0,f$,0:Use Palette 0

;draws first frame to current bitmap (1) and bitmap #0

InitAnim 0,0
While Joyb(0)=0
    ShowBitMap db        ;tell intuition which bitmap to display
    VWait                ;wait for top of frame
    db=1-db              ;swap current bitmap
    Use BitMap db
    NextFrame 0          ;and draw next frame
Wend
```

1.281 bum_loadanim

```
Statement: LoadAnim
------------------------------------------------------------------------
Syntax: LoadAnim Anim#,FileName$[,Palette#]
Modes: Amiga

Description:
The LoadAnim command will create an Anim object and load a DPaint
compatible animation. The ILBMInfo command can be used to find the
correct screensize and resolution for the anim file. The optional
Palette# parameter can be used to load a palette with the anims correct
colours.

Notes:
unlike more advanced anim formats DPaint anims use a single static
palette for the entire animation. Like all other Blitz commands that
access files the command must be executed in Amiga mode.
```

1.282 bum_initanim

```
Statement: InitAnim
------------------------------------------------------------------------------
Syntax: InitAnim Anim#[,Bitmap#]
Modes: Amiga/Blitz

Description:
InitAnim renders the first two frames of the Anim onto the current
BitMap and the BitMap specified by the second parameter. The second
BitMap# parameter is optional, this is to support Anims that are not in
a double-buffered format (each frame is a delta of the last frame not
from two frames ago). However, the two parameter double buffered form of
InitAnim should always be used. (hmmm don't ask me O.K.!)
```

1.283 bum_nextframe

```
Statement: NextFrame
------------------------------------------------------------------------------
Syntax: NextFrame Anim#
Modes: Amiga/Blitz

Description:
NextFrame renders the nextframe of an Anim to the current BitMap. If
the last frame of an Anim has been rendered NextFrame will loop back to
the start of the Animation.
```

1.284 bum_frames

```
Function: Frames
------------------------------------------------------------------------------
Syntax: Frames (Anim#)

Description:
The Frames() function returns the number of frames in the specified
Anim.
```

1.285 bum_showbitmap

```
Statement: ShowBitMap
------------------------------------------------------------------------------
Syntax: ShowBitMap [BitMap#]
Modes: Amiga
Library: ScreensLib

Description:
The ShowBitMap command is the Amiga-mode version of the Show command. It
enables you to change a Screens bitmap allowing double buffered (flicker
free) animation to happen on a standard Intuition Screen.

Unlike Blitz mode it is better to do ShowBitMap then VWait to sync up
```

```
with the Amiga's display, this will make sure the new bitmap is being
displayed before modifying the previous BitMap.
```

1.286 bum_blitcoll

```
Function: BlitColl
------------------------------------------------------------------------
Syntax: BlitColl (Shape#,x,y)
Modes: Amiga/Blitz

Description:
BlitColl is a fast way of collision detection when blitting shapes.
BlitColl returns -1 if a collision occurs, 0 if no collision. A
collision occurs if any pixel on the current BitMap is non zero where
your shape would have been blitted.

ShapesHit is faster but less accurate as it checks only the rectangular
area of each shape, where as BlitColl takes into account the shape of
the shape and of course1bcan not tell you what shapeyou have collided
with.

Note: make sure only things that you want to
collide with have been drawn on the BitMap e.g. don't Blit your ship and
then try BlitColl!
```

1.287 bum_ilbmviewmode

```
Statement: ILBMViewMode
------------------------------------------------------------------------
Syntax: ILBMViewMode
Modes: Amiga/Blitz
Library: ILBMIFFLib

Description:
ILBMViewMode returns the viewmode of the file that was processed by
ILBMInfo. This is useful for opening a screen in the right mode before
using LoadScreen etc. The different values of ViewMode are as follows
(add/or them for different combinations):

32768 ($8000) hires
2048  ($0800) ham
128   ($0080) halfbright
4     ($0004) interlace
0     ($0000) lores

See Also:  ILBMInfo

Example:

;
;ilbminfo example
;
```

```
;iff file name could use f$=par$(1) to use cli argument
f$="test.iff"
;get ilbm information
ILBMInfo f$
;open screen with correct parameters
Screen 0,0,0,ILBMWidth,ILBMHeight,ILBMDepth,ILBMViewMode,"",1,2
;load the iff onto the screens
LoadScreen 0,f$,0
;set the palette
Use Palette 0
MouseWait
```

1.288 bum_loadshape

```
Statement: LoadShape
------------------------------------------------------------------------
Syntax: LoadShape Shape#,Filename$[,Palette#]
Modes: Amiga

Description:
The LoadShape command has now been extended to support anim brushes, if
the file is an anim brush the shapes are loaded into consecutive shapes
starting with the Shape# provided.
```

1.289 bum_remap

```
Statement: ReMap
------------------------------------------------------------------------
Syntax: ReMap colour#0,colour#1[,Bitmap]
Modes: Amiga/Blitz
Library: Sis2dLib

Description:
ReMap is used to change all the pixels on a BitMap in one colour to
another colour. The optional BitMap parameter will copy all the pixels
in Colour#0 to their new colour on the new bitmap.
```

1.290 bum_shapegadget

```
Statement: ShapeGadget
------------------------------------------------------------------------
Syntax: ShapeGadget GadgetList#,X,Y,Flags,Id,Shape#[,Shape#]
Mode: Amiga

Description:
The ShapeGadget command allows you to create gadgets with graphic
imagery. The Shape# parameter refers to a shape object containing the
graphics you wish the gadget to contain.

The ShapeGadget command has been extended to allow an alternative image
```

```
to be displayed when the gadget is selected. All other parameters are
identical to those in TextGadget.

Example:

;
;ShapeGadget example
;
Screen 0,3
ScreensBitMap 0,0
;generate 2 shapes for our shape gadget
Cls:Circlef 15,15,15,2:Circlef 8,8,9,5,3:Circlef 24,8,9,2,3
GetaShape 1,0,0,32,32:Circlef 24,8,9,5,3:GetaShape 0,0,0,32,32
;
ShapeGadget 0,148,50,0,1,0,1
TextGadget 0,140,180,0,2,"EXIT"
Window 0,0,0,320,200,$100f,"ClickMe",1,2,0

Repeat
Until WaitEvent=64 AND GadgetHit=2
```

1.291 bum_setbplcon0

```
Statement: SetBPLCON0
------------------------------------------------------------------------
Syntax: SetBPLCON0 Default
Modes: Amiga/Blitz

Description:
The SetBPLCON0 command has been added for advanced control of Slice
display modes. The BPLCON0 hardware register is on page A4-1 of the
reference manual (appendix 4). The bits of interest are as follows:

bit#1-ERSY external sync (for genlock enabling)
bit#2-LACE interlace mode
bit#3-LPEN light pen enable

Example:

;
; Blitz Interlaced Slice Example using BPLCON0
;
BitMap 0,640,512,4
; use SetBPLCON0 4 to set the lace bit on when slice is created
SetBPLCON0 4                                ;set lace bit

BLITZ
;bitmap width=1280 so slice's bitmap modulos miss each 2nd line
Slice 0,44,640,256,$fffb,4,8,8,1280,1280   ;cludge the modulo
;every vertical blank either show odd lines or even lines
;depending on the long frame bit of VPOSR hardware register
SetInt 5
  If Peek($dff004)<0 Show 0,0,0 Else Show 0,0,1
End SetInt
;draw lines to prove it
```

```
For i=1 To 1000
  Line Rnd(640),Rnd(512),Rnd(640),Rnd(512),Rnd(16)
Next
;
MouseWait
```

1.292 bum_speakcommands

```
                Speak Commands


The Amiga speech synthesiser can be activated using the following
commands. The narrator.device has been upgraded in Workbench2.0
increasing the quality of the speech. With a bit of messing around you
can have a lot of fun with the Amiga's 'voice', Also note that these are
compatible with the commands used in BlitzUser1's speech program.

Speak Commands:


                 Speak

                 SetVoice

                 Translate$

                 PhoneticSpeak

                 VoiceLoc
```

1.293 bum_speak

```
                Statement: Speak
------------------------------------------------------------------------------
Syntax: Speak string$
Modes: Amiga

Description:
The Speak command will first convert the given string to phonetics and
then pass it to the Narrator.Device. Depending on the settings of the
Narrator device (see
                 SetVoice
                ) the Amiga will
"speak" the string you have sent in the familiar Amiga synthetic voice.

Example:

NPrint "Type something and hit return..."
NPrint "(just return to exit)"
Repeat
  a$=Edit$(80)
  Speak a$
```

```
Until a$=""
```

1.294 bum_setvoice

```
Statement: SetVoice
------------------------------------------------------------------------------
Syntax: SetVoice rate,pitch,expression,sex,volume,frequency
Modes: Amiga

Description:
SetVoice alters the sound of the Amiga's speech synthsiser by changing:

Rate:       measured in words per minute. Default 150, range 40-400.
Pitch:      the BaseLine pitch in Hz. Default 110, range 65-320
Expression: 0=robot 1=natural 2=manual
Sex:        0=male 1=female
Volume:     0 to 64
Frequency:  samples per second (22200)

As the following example shows you could very well rename the Speak
command the Sing command!
;
; sing the praises of Blitz BASIC!
;
While Joyb(0)=0
  Pitch=65+Rnd(255)
  rate=100+Rnd(200)
  SetVoice rate,pitch,1,1,64,22200
  Speak "BLITZ BASIC"
Wend
```

1.295 bum_translate$

```
Function: Translate$
------------------------------------------------------------------------------
Syntax: Translate$(string$)
Modes: Amiga

Description:
Translate$() returns the phonetic equivalent of the string for use with
the Translate

Example:

Print "Enter a Sentence ":a$=Edit$(80)
NPrint "Phonetic=",Translate$(a$)
MouseWait
```

1.296 bum_phoneticspeak

```
Statement: PhoneticSpeak
------------------------------------------------------------------------------
Syntax: PhoneticSpeak phonetic$
Modes: Amiga

Description:
PhoneticSpeak is similar to the Speak command but should only be passed
strings containing legal phonemes such as that produced by the
Translate$() function.
```

1.297 bum_voiceloc

```
Function: VoiceLoc
------------------------------------------------------------------------------
Syntax: VoiceLoc
Modes: Amiga

Description:
VoiceLoc returns a pointer to the internal variables in the speech
synthesiser that enable the user to access new parameters added to the
V37 Narrator Device. Formants as referred to in the descriptions are the
major vocal tracts and are separated into the parts of speech that
produce the bass, medium and trebly sounds.

The new paramters are as listed

\flags                : set to 1 if using extended commands
\f0enthusiasm         : amount of pitch difference on accents default=32
\f0perturb            : amount of "wurble" ie random shake default=0
\f1adj,\f2adj,\f3adj  : pitch adjust for low medium and high frequency
                        formants. 0=default
\a1adj,\a2adj,\a3adj  : amplitude adjust for low medium and high
                        frequency formants 0=default
\articulate           : speed of articulation 100=default
\centralize           : amount of the centphon vowel in other vowels
                        0=default
\centphon             : a vowel to which all others are adjusted by the
\centralize           : variable,
                        (limited to IY,IH,EH,AE,AA,AH,AO,OW,UH,ER and
                         UW)
\AVbias,\AFbias       : amount of bias added to voiced and unvoiced
                        speech sounds, (y,r,w,m vs st,sh,f).
\priority             : task priority when speaking 100=default

Example:

;
; voiceloc() example
;

NEWTYPE .voicepars          ;new V37 parameters available
    flags.b
    f0enthusiasm:f0perturb
    f1adj:f2adj:f3adj
```

```
    a1adj:a2adj:a3adj
    articulate:centralize:centphon$
    avbias.b:afbias:priority:pad1
End NEWTYPE

*v.voicepars=VoiceLoc

*v\flags=1
*v\f0enthusiasm=82,90 ;old aged highly excited voice
*v\f1adj=0,0,0    ;these are fun to mess with
*v\a1adj=0,0,0
*v\centralize=50,"AO" ;no effect
*v\articulate=90
*v\avbias=20,20

Speak "COME ON EVERYBODY, DANCE? boom boom !"
End
```

1.298 bum_medlib

The MED commands:

LoadMedModule

SetMedVolume

StartMedModule

GetMedVolume

PlayMed

GetMedNote

StopMed

GetMedInstr

JumpMed

SetMedMask

1.299 bum_loadmedmodule

```
Statement: LoadMedModule
------------------------------------------------------------------------------
Syntax: LoadMedModule MedModule# Name
Modes: Amiga

Description:
The LoadMedModule command loads any version 4 channel Octamed module.
```

The following routines support upto and including version 3 of the Amiganut's Med standard.

The number of MedModules loaded in memory at one time is only limited by the MedModules maximum set in the Blitz2 Options requester. Like any Blitz commands that access files LoadMedModule can only be used in AmigaMode.

1.300 bum_startmedmodule

Statement: StartMedModule

Syntax: StartMedModule MedModule#
Modes: Amiga/Blitz

Description:
StartMedModule is responsible for initialising the module including linking after it is loaded from disk using the LoadMedModule command. It can also be used to restart a module from the beginning.

1.301 bum_playmed

Statement: PlayMed

Modes: Amiga/Blitz

Description:
PlayMed is responsible for playing the current MedModule, it must be called every 50th of a second either on an interupt (#5) or after a VWait in a program loop.

1.302 bum_stopmed

Statement: StopMed

Syntax: StopMed
Modes: Amiga/Blitz

Description:
StopMed will cause any med module to stop playing. This not only means that PlayMed will have no affect until the next StartMedModule but silences the audio channels so they are not left ringing as is the effect when PlayMed is not called every vertical blank.

1.303 bum_jumpmed

```
Statement: JumpMed
------------------------------------------------------------------------------
Syntax: JumpMed Pattern#
Modes: Amiga/Blitz

Description:
JumpMed will change the pattern being played in the current module.
```

1.304 bum_setmedvolume

```
Statement: SetMedVolume
------------------------------------------------------------------------------
Syntax: SetMedVolume Volume
Modes: Amiga/Blitz

Description:
SetMedVolume changes the overall volume that the Med Library plays the
module, all the audio channels are affected. This is most useful for
fading out music by slowly decreasing the volume from 64 to 0.
```

1.305 bum_getmedvolume

```
Function: GetMedVolume
------------------------------------------------------------------------------
Syntax: GetMedVolume Channel#
Modes: Amiga/Blitz

Description:
GetMedVolume returns the current volume setting of the specified audio
channel. This is useful for graphic effects that you may wish to sync
to certain channels of the music playing.
```

1.306 bum_getmednote

```
Function: GetMedNote
------------------------------------------------------------------------------
Syntax: GetMedNote Channel#
Modes: Amiga/Blitz

Description:
GetMedNote returns the current note playing from the specified channel.
As with GetMedVolume this is useful for producing graphics effects
synced to the music the Med Library is playing.
```

1.307 bum_getmedinstr

```
Function: GetMedInstr
------------------------------------------------------------------------------
Syntax: GetMedInstr Channel
Modes: Amiga/Blitz

Description:
GetMedInstr returns the current instrument playing through the specified
audio channel.
```

1.308 bum_setmedmask

```
Statement: SetMedMask
------------------------------------------------------------------------------
Syntax: SetMedMask Channel Mask
Modes: Amiga/Blitz

Description:
SetMedMask allows the user to mask out audio channels needed by sound
effects stopping the Med Library using them.
```

1.309 bum_serialport

```
                Serial Port Commands

The following are a set of commands to drive both the single RS232
serial port on an Amiga as well as supporting multiserial port cards
such as the A2232 card. The unit# in the following commands should be
set to 0 for the standard RS232 port, unit 1 refers to the default
serial port set by the advanced serial preferences program and unit 2
on refer to any extra serial ports available.

                            Serial Port Commands:


                 OpenSerial

                 WriteSerial

                 WriteSerialString

                 ReadSerial

                 ReadSerialString

                 CloseSerial

                 SetSerialBuffer

                 SetSerialLens

                 SetSerialParams
```

```
                SerialEvent
```

1.310 bum_openserial

```
Function: OpenSerial
------------------------------------------------------------------------------
Syntax: OpenSerial device$,unit#,baud,io_serflags
Modes: Amiga

Description:
OpenSerial is used to configure a Serial Port for use. As with OpenFile,
OpenSerial is a function and returns zero if it fails. If it succeeds
advanced users may note the return result is the location of the IOExtSer
structure.

Use "serial.device" for device$.

The baud rate should be in the range of 110-292,000. The io_serflags
parameter includes the following flags:

bit7: #serf_xdisabled=128 ; disable xon/xoff
bit6: #serf_eofmode=64    ; enable eof checking
bit5: #serf_shared=32     ; set if you don't need exclusive use of port
bit4: #serf_rad_boogie=16 ; high speed mode
bit3: #serf_queuedbrk=8   ; if set, a break command waits for buffer
                            empty
bit2: #serf_7wire=4       ; if set, use 7 wire RS232
bit1: #serf_parity_odd=2  ; select odd parity (even if not set)
bit0: #serf_parity_on=1   ; enable parity checking
```

1.311 bum_writeserial

```
Statement: WriteSerial
------------------------------------------------------------------------------
Syntax: WriteSerial unit#,byte
Modes: Amiga

Description:
WriteSerial sends one byte to the serial port. Unit# defines which
serial port is used. If you are sending characters use the Asc()
function to convert the character to a byte e.g. WriteSerial 0,asc("b").
```

1.312 bum_writeserialstring

```
Statement WriteSerialString
------------------------------------------------------------------------------
Syntax: WriteSerialString unit#,string
Modes: Amiga
```

```
Description:
WriteSerialString is similar to WriteSerial but sends a complete string
to the serial port.
```

1.313 bum_readserial

```
Function: ReadSerial
------------------------------------------------------------------------
Syntax: ReadSerial (unit#)  returns -1 if nothing waiting
Modes: Amiga

Description:
ReadSerial returns the next byte waiting in the serial port's read
buffer. If the buffer is empty it returns a -1. It is best to use a word
type (var.w=ReadSerial(0)) as a byte will not be able to differentiate
between -1 and 255.
```

1.314 bum_readserialstring

```
Function: ReadSerialString
------------------------------------------------------------------------
Syntax: ReadSerialString (unit#)
Modes: Amiga

Description:
ReadSerialString puts the serial port's read buffer into a string, if
the buffer is empty the function will return a null string (length=0).
```

1.315 bum_closeserial

```
Statement: CloseSerial
------------------------------------------------------------------------
Syntax: CloseSerial unit#
Modes: Amiga

Description:
The CloseSerial command will close the port, enabling other programs to
use it.

Note: Blitz will automatically close all ports that are opened when a program
ends.
```

1.316 bum_setserialbuffer

```
Statement SetSerialBuffer
------------------------------------------------------------------------
Modes: Amiga
```

```
Description:
SetSerialBuffer changes the size of the ports read buffer. This may be
useful if your program is not always handling serial port data or is
receiving and processing large chunks of data. The smallest size for
the internal serial port (unit#0) is 64 bytes. The bufferlength variable
is in bytes.
```

1.317 bum_setseriallens

```
Statement: SetSerialLens
------------------------------------------------------------------------
Syntax: SetSerialLens unit#,readlen,writelen,stopbits
Modes: Amiga

Description:
SetSerialLens allows you to change the size of characters read and
written by the serial device. Generally readlen=writelen and should be
set to either 7 or 8, stopbits should be set to 1 or 2.

Default values are 8,8,1.
```

1.318 bum_setserialparams

```
Statement: SetSerialParams
------------------------------------------------------------------------
Syntax: SetSerialParams unit#
Modes: Amiga

Description:
For advanced users, SetSerialParams tells the serial port when
parameters are changed. This would only be necesary if they were changed
by poking offsets from IOExtSer which is returned by the OpenSerial
command.
```

1.319 bum_serialevent

```
Function: SerialEvent
------------------------------------------------------------------------
Syntax: SerialEvent (unit#)
Modes: Amiga

Description:
SerialEvent is used when your program is handling events from more than
1 source, Windows, ARexx etc.

This command is currently not implemented
```

1.320 bum_arexxcommands

Here's a overview of the Arexx Commands:

CreateMsgPort

ReplyRexxMsg

DeleteMsgPort

GetRexxResult

CreateRexxMsg

GetRexxCommand

DeleteRexxMsg

GetResultString

ClearRexxMsg

Wait

FillRexxMsg

RexxEvent

CreateArgString

IsRexxMsg

DeleteArgString

RexxError

SendRexxCommand

1.321 bum_createmsgport

```
Function: CreateMsgPort()
------------------------------------------------------------------------
Syntax: PortAddress.l = CreateMsgPort("Name")
MODES: Amiga

Description:
CreateMsgPort is a general Function and not specific to ARexx.

CreateMsgPort opens an intuition PUBLIC message port of the name
supplied as the only argument. If all is well the address of the port
created will be returned to you as a LONGWORD so the variable that you
assign it to should be of type long.
```

```
If you do not supply a name then a private MsgPort will be opened for
you.

  Port.l=CreateMsgPort("PortName")

It is important that you check you actually succeeded in opening a port
in your program. The following code or something similar will suffice.

  Port.l=CreateMsgPort("Name")
  IF Port=0 THEN Error_Routine{}

The name you give your port will be the name that Arexx looks for as the
HOST address,(and is case sensitive) so take this into consideration
when you open your port.  NOTE IT MUST BE A UNIQUE NAME AND SHOULD NOT
INCLUDE SPACES.
```

1.322 bum_deletemsgport

```
Statement: DeleteMsgPort()
------------------------------------------------------------------------
Syntax: DeleteMsgPort Port
Modes: Amiga

Desription:
DeleteMsgPort deletes a MessagePort previously allocated with
CreateMsgPort(). The only argument taken by DeleteMsgPort is the address
returned by CreateMsgPort(). If the Port was a public port then it will
be removed from the public port list.

  Port.l=CreateMsgPort("Name")
  IF Port=0 Then End
  DeleteMsgPort Port

Error checking is not critical as if this fails we have SERIOUS PROBLEMS.

YOU MUST WAIT FOR ALL MESSAGES FROM AREXX TO BE RECEIVED BEFORE YOU
DELETE THE MSGPORT. IF YOU NEGLECT TO DELETE A MSGPORT BLITZ2 WILL
DO IT FOR YOU AUTOMATICALLY ON PROGRAM EXIT.
```

1.323 bum_createrexxmsg

```
Function: CreateRexxMsg()
------------------------------------------------------------------------
Syntax: msg.l=CreateRexxMsg(ReplyPort,"exten","HOST")
Modes: Amiga

Description:
CreateRexxMsg() allocates a special Message structure used to
communicate with Arexx.  If all is successful it returns the LONGWORD
address of this rexxmsg structure.

The arguments are ReplyPort which is the long address returned by
```

```
CreateMsgPort(). This is the Port that ARexx will reply to after it has
finished with the message.

EXTEN which is the exten name used by any ARexx script you are wishing
to run. i.e. if you are attempting to run the ARexx script test.rexx you
would use an EXTEN of "rexx".

HOST is the name string of the HOST port.  Your program is usually the
HOST and so this equates to the name you gave your port in
CreateMsgPort(). REMEMBER IT IS CASE SENSITIVE.

As we are allocating resources error checking is important and can be
achieved with the following code:

  msg.l=CreateRexxMsg(Port,"rexx","HostName")
  IF msg=0 THEN Error_Routine{}
```

1.324 bum_deleterexxmsg

```
Statement: DeleteRexxMsg
------------------------------------------------------------------------
Syntax: DeleteRexxMsg rexxmsg
Modes: Amiga

Description:
DeleteRexxMsg simply deletes a RexxMsg Structure previously allocated by
CreateRexxMsg().  It takes a single argument which is the long address
of a RexxMsg structure such as returned by CreateRexxMsg().

  msg.l=CreateRexxMsg(Port,"rexx","HostName")
  IF msg=0 THEN Error_Routine{}
  DeleteRexxMsg msg

Again if you neglect to delete the RexxMsg structure Blitz2 will do this
for you on exit of the program.
```

1.325 bum_clearrexxmsg

```
Statement: ClearRexxMsg
------------------------------------------------------------------------
Syntax: ClearRexxMsg1k
Modes: Amiga

Description:
ClearRexxMsg is used to delete and clear an ArgString from one ormore of
the Argument slots in a RexxMsg Structure.  This is most useful for the
more advanced programmer wishing to take advantage of the Arexx #RXFUNC
abilities.

The arguments are a LONGWORD address of a RexxMsg structure.
ClearRexxMsg will always work from slot number 1 forward to 16.
```

```
  Port.l=CreateMsgPort("TestPort")
  If Port = NULL Then End
  msg.l=CreateRexxMsg(Port,"vc","TestPort")
  If msg=NULL Then End
  SendRexxCommand msg,"open",#RXCOMM|#RXFF_RESULT
  wait:WHILE GetMsg_(Port) <> msg:Wend ;Wait for reply to come
  ClearRexxMsg msg                     ;Delete the Command string
                                              we sent

NOTE: ClearRexxMsg() is called automatically by RexxEvent()
so the need to call this yourself is removed unless you have not sent
the RexxMsg to Arexx.
```

1.326 bum_fillrexxmsg

```
Statement: FillRexxMsg()
------------------------------------------------------------------------------
Syntax: FillRexxMsg rexxmsg,&FillStruct
Modes: Amiga

Description:
FillRexxMsg allows you to fill all 16 ARGSlots if necessary with either
ArgStrings or numerical values depending on your requirement.

FillRexxMsg will only be used by those programmers wishing to do more
advanced things with Arexx, including adding libraries to the ARexx
library list, adding Hosts,Value Tokens etc.  It is also needed to
access Arexx using the #RXFUNC flag.

The arguments are a LONG Pointer to a rexxmsg.

The LONG address of a FillStruct NEWTYPE structure. This structure is
defined in the Arexx.res and has the following form.

NEWTYPE.FillStruct
   Flags.w             ;Flag block
   Args0.l             ; argument block (ARG0-ARG15)
   Args1.l             ; argument block (ARG0-ARG15)
   Args2.l             ; argument block (ARG0-ARG15)
   Args3.l             ; argument block (ARG0-ARG15)
   Args4.l             ; argument block (ARG0-ARG15)
   Args5.l             ; argument block (ARG0-ARG15)
   Args6.l             ; argument block (ARG0-ARG15)
   Args7.l             ; argument block (ARG0-ARG15)
   Args8.l             ; argument block (ARG0-ARG15)
   Args9.l             ; argument block (ARG0-ARG15)
   Args10.l            ; argument block (ARG0-ARG15)
   Args11.l            ; argument block (ARG0-ARG15)
   Args12.l            ; argument block (ARG0-ARG15)
   Args13.l            ; argument block (ARG0-ARG15)
   Args14.l            ; argument block (ARG0-ARG15)
   Args15.l            ; argument block (ARG0-ARG15)
   EndMark.l           ;End of the FillStruct
End NEWTYPE
```

The Args?.l are the 16 slots that can possibly be filled ready for converting into the RexxMsg structure. The Flags.w is a WORD value representing the type of LONG word you are supplying for each ARGSLOT (Arg?.l).

Each bit in the Flags WORD is representative of a single Args?.l, where a set bit represents a numerical value to be passed and a clear bit represents a string argument to be converted into a ArgString before installing in the RexxMsg. The Flags Value is easiest to supply as a binary number to make the bits visible and would look like this.

```
%0000000000000000  ;This represents that all Arguments are Strings.

%0110000000000000  ;This represent the second and third as being
                    integers.
```

FillRexxMsg expects to find the address of any strings in the Args?.l slots so it is important to remember when filling a FillStruct that you must pass the string address and not the name of the string. This is acomplished using the '&' address of operand.

So to use FillRexxMsg we must do the following things in our program:

1. Allocate a FillStruct
2. Set the flags in the FillStruct\Flags.w
3. Fill the FillStruct with either integer values or the addresses of our string arguments.
4. Call FillRexxMsg with the LONG address of our rexxmsg and the LONG address of our FillStruct.

To accomplish this takes the following code:

```
  ;Allocate our FillStruct (called F)

  DEFTYPE.FillStruct F

  ;assign some string arguments

  T$="open":T1$="0123456789"

  ;Fill in our FillStruct with flags and (&) addresses of our
         strings

  F\Flags= %0010000000000000,&T$,&T1$,4

  ;Third argument here is an integer (4).

  Port.l=CreateMsgPort("host")
  msg.l=CreateRexxMsg(Port,"vc","host")

  FillRexxMsg msg,&F

  ;<-3 args see #RXFUNC

  SendRexxCommand msg,"",#RXFUNC|#RXFF_RESULT|3
```

1.327 bum_createargstring

```
Function: CreateArgString()
------------------------------------------------------------------------------
Syntax: ArgString.l=CreateArgString("this is a string")
Modes: Amiga

Description:
CreateArgString() builds an ARexx compatible ArgString structure around
the provided string.  All strings sent to, or received from Arexx are in
the form of ArgStrings. See the TYPE RexxARG.

If all is well the return will be a LONG address of the ArgString
structure. The pointer will actually point to the NULL terminated String
with the remainder of the structure available at negative offsets.

  arg.l=CreateArgString("this is a string")
    IF arg=0 THEN Error_Routine{}:ENDIF
  DeleteArgString arg

NOTE: An ArgString maybe used as a normal BB2 string variable
by simple conversion using PEEK$

i.e. msg$=PEEK$(arg) or perhaps NPRINT PEEK$(arg)

NOTE: Most of the BB2 Arexx Functions call this themselves and
there will be only limited need for you to access this function.
```

1.328 bum_deleteargstring

```
Statement: DeleteArgString
------------------------------------------------------------------------------
Syntax:DeleteArgString ArgString
Modes: Amiga

Description:
DeleteArgString is designed to Delete ArgStrings allocated by either
Blitz2 or ARexx in a system friendly way. It takes only one argument the
LONGWORD address of an ArgString as returned by CreateArgString().

  arg.l=CreateArgString("this is a string")
    IF arg=0 THEN Error_Routine{}:ENDIF
  DeleteArgString arg

NOTE: This function is also called automatically by most of
the BB2 Arexx Functions that need it so you should only need to call this on
rare occations.
```

1.329 bum_sendrexxcommand

```
Statement: SendRexxCommand
------------------------------------------------------------------------------
```

```
SendRexxCommand rexxmsg,"commandstring",#RXCOMM|#RXFF_RESULT
Modes: Amiga

Description:
SendRexxCommand is designed to fill and send a RexxMsg structure to
ARexx inorder to get ARexx to do something on your behalf.

The arguments are as follows;

Rexxmsg is the LONGWORD address of a RexxMsg structure as returned by
CreateRexxMsg().

Commandstring is the command string you wish to send to ARexx. This is a
string as in "this is a string" and will vary depending on what you wish
to do with ARexx. Normally this will be the name of an ARexx script file
you wish to execute. ARexx will then look for the script by the name as
well as the name with the exten added.(this is the exten you used when
you created the RexxMsg structure using CreateRexxMsg()). This could
also be a string file. That is a complete ARexx script in a single line.

ActionCodes are the flag values you use to tell ARexx what you want it
to do with the commandstring you have supplied. The possible flags are
as follows;

COMMAND (ACTION) CODES

The command codes that are currently implemented in the resident process
are described below. Commands are listed by their mnemonic codes,
followed by the valid modifier flags. The final code value is always the
logical OR of the code value and all of the modifier flags selected. The
command code is installed in the rm_Action field of the message packet.

USAGE: RXADDCON

This code specifies an entry to be added to the Clip List. Parameter
slot ARGO points to the name string,slot ARG1 points to the value
string,and slot ARG2 contains the length of the value string.

The name and value arguments do not need to be argstrings,but can be
just pointers to storage areas. The name should be a null-terminated
string,but the value can contain arbitrary data including nulls.

USAGE: RXADDFH

This action code specifies a function host to be added to the Library
List. Parameter slot ARGO points to the (null-terminated) host name
string,and slot ARG1 holds the search priority for the node. The search
 priority should be an integer between 100 and -100 inclusive;the
remaining priority ranges are reserved for future extensions. If a node
already exists with the same name,the packet is returned with a warning
level error code.

Note that no test is made at this time as to whether the host port
exists.

USAGE:RXADDLIB
```

This code specifies an entry to be added to the Library List. Parameter slot ARGO points to a null-terminated name string referring either to a function library or a function host. Slot ARG1 is the priority for the node and should be an integer between 100 and -100 inclusive;the remaining priority ranges are reserved for future extensions. Slot ARG2 contains the entry Point offset and slot ARG3 is the library version number. If a node already exists with the same name,the packet is returned with a warning level error code. Otherwise,a new entry is added and the library or host becomes available to ARexx programs. Note that no test is made at this time as to whether the library exists and can be opened.

USAGE:RXCOMM [RXFF_TOKEN] [RXFF_STRING] [RXFF_RESULT] [RXFF_NOIO]

Specifies a command-mode invocation of an ARexx program. Parameter slot ARGO must contain an argstring Pointer to the command string. The RXFB_TOKEN flag specifies that the command line is to be tokenized before being passed to the invoked program. The RXFB_STRING flag bit indicates that the command string is a "string file." Command invocations do not normally return result strings,but the RXFB_RESULT flag can be set if the caller is prepared to handle the cleanup associated with a returned string. The RXFB_NOIO modifier suppresses the inheritance of the host’s input and output streams.

USAGE:RXFUNC [RXFF_RESULT] [RXFF_STRING] [RXFF_NOIO] argcount

This command code specifies a function invoction. Parameter slot ARGO contains a pointer to the function name string,and slots ARG1 through ARG15 point to the argument strings,all of which must be passed as argstrings. The lower byte of the command code is the argument count; this count excludes the function name string itself. Function calls normally set the RXFB_RESULT flag,but this is not mandatory. The RXFB_STRING modifier indicates that the function name string is actually a "string file". The RXFB_NOIO modifier suppresses the inheritance of the host’s input and output streams.

USAGE:RXREMCON

This code requests that an entry be removed from the Clip List. Parameter slot ARGO points to the null-terminated name to be removed. The Clip List is searched for a node matching the supplied name,and if a match is found the list node is removed and recycled. If no match is found the packet is returned with a warning error code.

USAGE:RXREMLIB

This command removes a Library List entry. Parameter slot ARGO points to the null terminated string specifying the library to be removed. The Library List is searched for a node matching the library name,and if a match is found the node is removed and released. If no match is found the packet is returned with a warning error code. The libary node will not be removed if the library is currently being used by an ARexx program.

USAGE:RXTCCLS

This code requests that the global tracing console be closed. The

console window will be closed immediately unless one or more ARexx programs are waiting for input from the console. In this event,the window will be closed as soon as the active programs are no longer using it.

USAGE:RXTCOPN

This command requests that the global tracing console be opened. Once the console is open,all active ARexx programs will divert their tracing output to the console. Tracing input(for interactive debugging)will also be diverted to the new console. Only one console can be opened; subsequent RXTCOPN requests will be returned with a warning error message.

MODIFIER FLAGS

Command codes may include modifier flags to select various processing options. Modifier flags are specific to certain commands,and are ignored otherwise.

RXFF_NOIO.

This modifier is used with the RXCOMM and RXFUNC command codes to suppress the automatic inheritance of the host’s input and output streams.

RXFF_NONRET.

Specifies that the message packet is to be recycled by the resident process rather than being returned to the sender. This implies tht the sender doesn’t care about whether the requested action succeeded, since the returned packet provides the only means of acknowledgement.

(RXFF_NONRET MUST NOT BE USED AT ANY TIME)

RXFF_RESULT.

This modifer is valid with the RXCOMM and RXFUNC commands,and requests that the called program return a result string. If the program EXITs(or RETURNs)with an expression,the expression result is returned to the caller as an argstring. This ArgString then becomes the callers responsibility to release. This is automatically accomplished by using GetResultString(). It is therefore imperitive that if you use RXFF_RESULT then you must use GetResultString() when the message packet is returned to you or you will incure a memory loss equal to the size of the ArgString Structure.

RXFF_STRING.

This modifer is valid with the RXCOMM and RXFUNC command codes. It indicates that the command or function argument(in slot ARGO)is a "string file" rather than a file name.

RXFF_TOKEN.

This flag is used with the RXCOMM code to request that the command string be completely tokenized before being passed to the invoked

```
program. Programs invoked as commands normally have only a single
argument string. The tokenization process uses "white space" to
separate the tokens,except within quoted strings. Quoted strings can use
either single or double quotes,and the end of the command string(a null
character) is considered as an implicit closing quote.

EXAMPLES:

  Port.l=OpenRexxPort("TestPort")
      If Port = NULL End:EndIf
  msg.l=CreateRexxMsg(Port,"vc","TestPort")
      If msg=NULL End:EndIf
  SendRexxCommand msg,"open",#RXCOMM|#RXFF_RESULT
```

1.330 bum_replyrexxmsg

```
Statement: ReplyRexxMsg
------------------------------------------------------------------------
Syntax: ReplyRexxMsg rexxmsg,Result1,Result2,"ResultString"
Modes: Amiga

Description:
When ARexx sends you a RexxMsg (Other than a reply to yours i.e.
sending yours back to you with results) you must repl to the message
before ARexx will continue or free that memory associated with that
RexxMsg.  ReplyRexxMsg accomplishes this for you. ReplyRexxMsg also will
only reply to message that requires a reply so you do not have to
include message checking routines in your source simply call
ReplyRexxMsg on every message you receive wether it is a command or not.

The arguments are;

rexxmsg is the LONGWORD address of the RexxMsg Arexx sent you as
returned by GetMsg_(Port).

Result1 is 0 or a severity value if there was an error.

Result2 is 0 or an Arexx error number if there was an error processing
the command that was contained in the message.

ResultString is the result string to be sent back to Arexx. This will
only be sent if Arexx requested one and Result1 and 2 are 0.

ReplyRexxMsg rexxmsg,0,0,"THE RETURNED MESSAGE"
```

1.331 bum_getrexxresult

```
Function: GetRexxResult()
------------------------------------------------------------------------
Syntax: Result.l=GetRexxResult(rexxmsg,ResultNum)
Modes: Amiga
```

```
Description:
GetRexxResult extracts either of the two result numbers from the RexxMsg
structure.  Care must be taken with this Function to ascertain wether
you are dealing with error codes or a ResultString address.  Basically
if result 1 is zero then result 2 will either be zero or contain a
ArgString pointer to the ResultString. This should then be obtained
using GetResultString().

The arguments to GetRexxResult are;

rexxmsg is the LONGWORD address of a RexxMsg structure returned from
ARexx.

ResultNum is either 1 or 2 depending on wether you wish to check result
1 or result 2.

  ;print the severity code if there was an error

  NPrint GetRexxResult(msg,1)

  ;check for ResultString and get it if there is one

  IF GetRexxResult(msg,1)=0
  IF GetRexxResult(msg,2) THEN GetResultString(msg)
  ENDIF
```

1.332 bum_getrexxcommand

```
Function: GetRexxCommand()
------------------------------------------------------------------------
Syntax: String$=GetRexxCommand(msg,1)
Modes: Amiga

Description:
GetRexxCommand allows you access to all 16 ArgString slots in the given
RexxMsg. Slot 1 contains the command string sent by ARexx in a command
message so this allows you to extract the Command.

Arguments are:

rexxmsg is a LONGWORD address of the RexxMsg structure as returned by
 RexxEvent()

ARGNum is an integer from 1 to 16 specifying the ArgString Slot you wish
to get an ArgString from.

BEWARE YOU MUST KNOW THAT THERE IS AN ARGSTRING THERE.
```

1.333 bum_getresultstring

```
Function: GetResultString()
------------------------------------------------------------------------
```

```
Syntax: String$=GetResultString(rexxmsg)
Modes: Amiga

Description:
GetResultString allows you to extract the result string returned to you
by ARexx after it has completed the action you requested. ARexx will
only send back a result string if you asked for one (using the
ActionCodes) and the requested action was successful.

  ;check for ResultString and get it if there is one

  IF GetRexxResult(msg,1)=0
    IF GetRexxResult(msg,2) THEN GetResultString(msg)
  ENDIF

NOTE: Do not attempt to DeleteArgString the result
string returned by this function as the return is a string and not an
ArgString pointer. BB2 will automatically delete this argstring for you.
```

1.334 bum_wait

```
Statement: Wait
------------------------------------------------------------------------
Syntax: Wait
Modes: Amiga

Description:
Wait halts all program execution until an event occurs that the program
is interested in.  Any intuition event such as clicking on a gadget in
a window will start program execution again.

A message arriving at a MsgPort will also start program execution again.
So you may use Wait to wait for input from any source including messages
from ARexx to your program.

Wait should always be paired with EVENT if you need to consider
intuition events in your event handler loop.

  Repeat
      Wait:rmsg.l=REXXEVENT(Port):ev.l=EVENT
      IF IsRexxMsg(Rmsg) Process_Rexx_Messages{}:ENDIF
      ;
      ;
      ;Rest of normal intuition event loop statements case etc
      ;
  Until ev =$200
```

1.335 bum_rexxevent

```
                Function: RexxEvent()
------------------------------------------------------------------------
Syntax: Rmsg.l=RexxEvent(Port)
```

```
Modes: Amiga

Description:
RexxEvent is our Arexx Equivalent of EVENT().  It's purpose is to check
the given Port to see if there is a message waiting there for us.

It should be called after a WAIT and will either return a NULL to us if
there was no message or the LONG address of a RexxMsg Structure if
there was a message waiting.

Multiple Arexx MsgPorts can be handled using separate calls to
RexxEvent():

Wait:Rmsg1.l=RexxEvent(Port1):Rmsg2.l=RexxEvent(Port2):etc

RexxEvent also takes care of automatically clearing the rexxmsg if it is
our message being returned to us.

The argument is the LONG address of a MsgPort as returned by
CreateMsgPort().

EXAMPLES:

  Repeat
      Wait:Rmsg.l=REXXEVENT(Port):ev.l=EVENT
      IF IsRexxMsg(Rmsg) Process_Rexx_Messages{}:ENDIF
      ;
      ;
      ;Rest of normal intuition event loop statements case etc
  Until ev =$200

SEE ALSO:
                 Wait()
                ,
                 CreateMsgPort()
```

1.336 bum_isrexxmsg

```
                Function: IsRexxMsg()
------------------------------------------------------------------------
Syntax: IsRexxMsg(rexxmsg)
Modes: Amiga

Description:
IsRexxMsg tests the argument (a LONGWORD pointer hopefully to a message
packet) to see if it is a RexxMsg Packet. If it is TRUE is returned (1)
or FALSE if it is not (0).

  Repeat
      Wait:Rmsg.l=REXXEVENT:ev.l=EVENT
      IF IsRexxMsg(Rmsg) Process_Rexx_Messages{}:ENDIF
      ;
      ;
    ;Rest of normal intuition event loop statements case etc
  Until ev =$200
```

```
As the test is non destructive and extensive passing a NULL value or a
LONGWORD that does not point to a Message structure (Intuition or Arexx)
will safely return as FALSE.

SEE ALSO:
                 CreateRexxMsg()
                , GetMsg_()
```

1.337 bum_rexxerror

```
                Function: RexxError()
------------------------------------------------------------------------
Syntax: ErrorString$=RexxError(ErrorCode)
Modes: Amiga

Description:
RexxError converts a numerical error code such as you would get from
GetRexxResult(msg,2) into an understandable string error message.  If
the ErrorCode is not known to ARexx a string stating so is returned
this ensures that this function will always succeed.

  NPRINT RexxError(5)

SEE ALSO:
                 GetRexxResult()
```

1.338 bum_agahandling

```
                                              AGA PALETTE HANDLING

Blitz 2's palette object has (again) changed. Palette objects are now
capable of containing AGA compatible 24 bit colours.

                                  AGA Commands:


                 AGARGB

                 AGAGreen

                 AGAPalRGB

                 AGABlue

                 AGARed

                 NEW SCREENFLAGS

                 3.0 BITMAP HANDLING

                 NEW GADGET HANDLING
```

```
                The new palette objects look like this:

NEWTYPE.rgbcomp
  _red.l   ;left justified red component.
  _green.l ;left justified green component.
  _blue.l  ;left justified blue component.
End NEWTYPE

NEWTYPE.palettedata
  _numcols.w          ; same as palette/_numcols.
  _zero.w             ; for compatibility with graphics lib
                      ; LoadRGB32.
  _rgbs.rgbcomp[256]  ;256 is the max the amount will actually
                      ;depend upon the highest palette entry.
  _zero2.l            ;for graphics lib too.
End NEWTYPE

This is the actual object return by Addr Palette(n):

NEWTYPE.palette
  _*data.palettedata ; 00: NULL if no palette present
                     ;     else a pointer to  palettedata.
  _numcols.w         ; 04: num cols present in palettedata.
                     ;     below is colour cycling info.
  _lowcol.w          ; 06: low colour for cycle range.
  _hicol.w           ; 08: high colour for cycle range.
  _speed.w           ; 10: speed of cycle : 16384 = max speed
                     ;     sign indicates cycling direction.
  _var.w             ; 12: cvariable speed is added to.
                     ;
                     ; more possible cycling entries....
                     ;
                     ; 128: sizeof.
End NEWTYPE

Now for the new AGA functions added to Blitz 2...these will all generate
a runtime error if used on a non-AGA Amiga....
```

1.339 bum_agargb

```
Statement: AGARGB
------------------------------------------------------------------------------
Syntax: AGARGB Colour Register,Red,Green,Blue
Modes: Amiga

Description:
The AGARGB command is the AGA equivalent of the RGB command. The 'Red',
'Green' and 'Blue' parameters must be in the range 0 through 255, while
'Colour Register' is limited to the number of colours available on the
currently used screen.

Example:

  ;
  ; AGA test
```

```
  ;

  Screen 0,0,0,1280,512,8,$8024,"SUPER HIRES 256 COLORS",1,2

  ScreensBitMap 0,0

  For i=0 To 255
    AGARGB i,i/2,i/3,i       ;shades of purple
      Circle 640,256,i*2,i,i   ;big SMOOTH circles
  Next

  MouseWait
```

1.340 bum_agapalrgb

```
Statement: AGAPalRGB
--------------------------------------------------------------------------------
Syntax: AGAPalRGB Palette#,Colour Register,Red,Green,Blue
Modes: Amiga

Description:
The AGAPalRGB command is the AGA equivalent of the PalRGB command.
AGAPalRGB allows you to set an individual colour register within a
palette object. This command only sets up an entry in a palette object,
and will not alter the actual screen palette until a 'Use Palette' is
executed.
```

1.341 bum_agared

```
Function: AGARed
--------------------------------------------------------------------------------
Syntax: AGARed(colour register)
Modes: Amiga

Description:
The AGARed function returns the red component of the specified colour
register within the currently used screen. The returned value will be
within the range 0 (being no red) through 255 (being full red).
```

1.342 bum_agagreen

```
Function: AGAGreen
--------------------------------------------------------------------------------
Syntax: AGAGreen(colour register)
Modes: Amiga

Description:
The AGAGreen function returns the green component of the specified
colour register within the currently used screen. The returned value
will be within the range 0 (being no green) through 255 (being full
```

green).

1.343 bum_agablue

```
Function: AGABlue
------------------------------------------------------------------------
Syntax: AGABlue(colour register)
Modes: Amiga

Description:
The AGABlue function returns the blue component of the specified colour
register within the currently used screen. The returned value will be
within the range 0 (being no blue) through 255 (being full blue).
```

1.344 bum_newscreenflags

```
                                NEW SCREEN FLAGS

The superhires viewmode flag $20 is now acceptable, but should always be
used in conjunction with the standard hires flag of $8000.

The depth of a screen may now be specified up to 8 bitplanes (256
colours) deep (if you've got an AGA machine!). Here's how you would go
about opening a super-hires, 256 colour screen:

Screen 0,0,0,1280,256,8,$8020,"MyScreen",1,0
```

1.345 bum_30bitmaphandling

```
3.0 BITMAP HANDLING

Blitz 2's Bitmap object has been upgraded to allow for interleaved
bitmaps:

NewType.Bitmap
_   ebwidth[0]     ;00: for compatability.
_   linemod.w      ;00: value to get from one scanline to next.
_   height.w       ;02: currently pixel height - but open to commodore
                   ;    'enhancement'.
_   depth.w        ;04: number of bitplanes.
_   pad.b[2]       ;06: nothing.
_   data.l[8]      ;08: actual bitplane pointers.
_   pad2.b[12]     ;40: zilch.
_   flags.w        ;0=normal bitmap, <0=interleaved.
_   bitplanemod.w  ;value to get from one bitplane to next. MAY BE 0!
_   xclip.w        ;pixel width for render clipping
_   yclip.w        ;pixel height for render clipping
_   cclip.w        ;number of colours available on bitmap ( = 2^_depth)
_   isreal.w       ;0=no bitmap here, <0=blitz created bitmap,
                    >0=borrowed
```

```
             ;64: sizeof
End NEWTYPE
```

Also, many Blitz2 bitmap related commands have been altered to take this new object into account.

1.346 bum_newgadgethandling

NEW GADGET HANDLING

A new bit, bit 9, in the 'Flags' parameter of the 'TextGadget' and 'ShapeGadget' commands allow you to create mutually exclusive radio button type gadgets. These gadgets DO NOT require Kickstart 2.0 to operate!

Here is an example of setting up some radio button style text gadgets:

```
TextGadget 0,16,16,512,1,"OPTION 1":Toggle 0,1,On
TextGadget 0,16,32,512,2,"OPTION 2"
TextGadget 0,16,48,512,3,"OPTION 3"
```

The new 'ButtonGroup' command allows you to specify which 'group' a series of button gadgets belong to. See 'ButtonGadget' below.

Note that if you are using button gadgets, you SHOULD really toggle ONE of the gadgets 'On' before giving the gadgetlist to a window - as in the example above.

Text Gadgets may now be used to create 'cycling' gadgets. Again, these gadgets DO NOT require kickstart 2.0 to work.

If you create a text gadget which contains the '|' character in the gadget's text, Blitz 2 will recognize this as a 'cycling' gadget, using the '|' character to separate the options - like this:

```
  TextGadget 0,16,16,0,1," HELLO |GOODBYE| SEEYA |"
```

Now, each time this gadget is clicked on, the gadgets text will cycle through 'Hello', 'GOODBYE' and 'SEEYA'. Note that each option is spaced out to be of equal length. This feature should not be used with a GadgetJam mode of 0.

NEW GADGETS EXAMPLE
NEW GADGET COMMANDS:

GagetStatus

ButtonGroup

ButtonId

Enable

```
                Disable

                SetGadgetStatus
```

1.347 bum_gadgetstatus

```
Function: GadgetStatus
------------------------------------------------------------------------
Syntax: GadgetStatus(GadgetList#,Id)
Modes: Amiga

Description:
GadgetStatus may be used to determine the status of the specified
gadget. In the case of 'toggle' type gadget, GadgetStatus will return
true (-1) if the gadget is currently on, or false (0) if the gadget is
currently off.

In the case of a cycling text gadget, GadgetStatus will return a value
of 1 or greater representing the currently displayed text within the
gadget.
```

1.348 bum_buttongroup

```
Statement: ButtonGroup
------------------------------------------------------------------------
Syntax: ButtonGroup Group
Modes: Amiga

Description:
ButtonGroup allows you to determine which 'group' a number of button
type gadgets belong to. Following the execution of ButtonGroup, any
button gadgets created will be identified as belonging to the specified
group. The upshot of all this is that button gadgets are only mutually
exclusive to other button gadgets within the same group.

'Group' must be a positive number greater than 0. Any button gadgets
created before a 'ButtonGroup' command is executed will belong to group
1.
```

1.349 bum_buttonid

```
Function: ButtonId
------------------------------------------------------------------------
Syntax: ButtonId(GadgetList#,ButtonGroup)
Modes: Amiga

Description:
ButtonId may be used to determine which gadget within a group of button
type gadgets is currently selected. The value returned will be the
GadgetId of the button gadget currently selected.
```

1.350 bum_enabledisable

```
Statements: Enable & Disable
------------------------------------------------------------------------------
Syntax: Enable GadgetList#,Id  & Disable GadgetList#,Id
Modes: Amiga

Description:
A gadget when disabled is covered by a "mesh" and can not be accessed
by the user. The commands Enable and Disable allow the programmer to
access this feature of Intuition.
```

1.351 bum_setgadgetstatus

```
Statement: SetGadgetStatus
------------------------------------------------------------------------------
Syntax: SetGadgetStatus GadgetList#,Id,Value
Modes: Amiga

Description:
SetGadgetStatus is used to set a cycling text gadget to a particular
value, once set ReDraw should be used to refresh the gadget to reflect
it's new value.
```

1.352 bum_newgadgetsexample

```
NEW GADGETS EXAMPLE:

  ;
  ; new gadget types
  ;
  WBStartup:FindScreen 0     ;open on workbench
  TextGadget 0,32,14,0,0,"CYCLE 1|CYCLE 2|CYCLE 3"

  ButtonGroup 1  ;first group of radio buttons follows
  For i=1 To 5
    TextGadget 0,32,14+i*14,512,i,"CHANNEL #"+Str$(i)
  Next

  ButtonGroup 2  ;second group of radio buttons follows
  For i=6 To 10
    TextGadget 0,32,14+i*14,512,i,"BAND #"+Str$(i)
  Next
  Window 0,20,20,160,180,$1008,"GADGET TEST",1,2,0

  Repeat               ;wait until close window gadget hit
    ev.l=WaitEvent
  Until ev=$200
```

1.353 bum_datetimecommands

Available commands:

SystemDate

Date$

NumDays

DateFormat

Days

Months

Years

WeekDay

Hours

Mins

Secs

1.354 bum_systemdate

```
Function: SystemDate
--------------------------------------------------------------------------------
Syntax: SystemDate
Modes: Amiga

Description:
SystemDate returns the system date as the number of days passed since
1/1/1978.

Example:

;
; date/time test
;

Dim d$(6):Restore daynames:For i=0 To 6:Read d$(i):Next
Dim m$(12):Restore monthnames:For i=1 To 12:Read m$(i):Next

NPrint Date$(SystemDate)
NPrint d$(WeekDay)," ",Days," ",m$(Months)," ",Years
NPrint Hours,":",Mins,":",Secs
NPrint "press mouse to quit"
MouseWait

daynames:
```

```
  Data$ SUNDAY,MONDAY,TUESDAY,WEDNESDAY
  Data$ THURSDAY,FRIDAY,SATURDAY
monthnames:
  Data$ JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

1.355 bum_date$

```
Function: Date$
--------------------------------------------------------------------------------
Syntax: Date$ (days)
Modes: Amiga

Description:
Date$ converts the format returned by SystemDate (days passed since
1/1/1978) into a string format of dd/mm/yyyy or mm/dd/yyyy depending on
the dateformat (defaults to 0).
```

1.356 bum_numdays

```
Function: NumDays
--------------------------------------------------------------------------------
Syntax: NumDays (date$)
Modes: Amiga

Description:
Numdays converts a Date$ in the above format to the day count format,
where numdays is the number of days since 1/1/1978.
```

1.357 bum_dateformat

```
Statement: DateFormat
--------------------------------------------------------------------------------
Syntax: DateFormat format# ; 0 or 1
Modes: Amiga

Description:
DateFormat configures the way both date$ and numdays treat a string
representation of the date: 0=dd/mm/yyyy and 1=mm/dd/yyyy
```

1.358 bum_days

```
Functions: Days Months Years & WeekDay
--------------------------------------------------------------------------------
Syntax: Days Months Years & WeekDay
Modes: Amiga

Description:
```

Days Months and Years each return the particular value relevant to the
last call to SystemDate. They are most useful for when the program needs
to format the output of the date other than that produced by date$.
WeekDay returns which day of the week it is with Sunday=0 through to
Saturday=6.

1.359 bum_hoursminssecs

Functions: Hours Mins & Secs

Syntax: Hours Mins & Secs
Modes: Amiga

Description:
Hours, Mins and Secs return the time of day when SystemDate was last
called.

1.360 bum_environments

New Environment commands:

WBWidth

WBHeight

WBDepth

WBViewMode

Processor

ExecVersion

Added in BUM #4

1.361 bum_wbwidth

Functions: WBWidth Height Depth & ViewMode

Syntax: WBWidth, WBHeight, WBDepth & WBViewMode
Modes: Amiga

Description:
The functions WBWidth, WBHeight, WBDepth & WBViewMode return the width,
height,depth & viewmode of the current WorkBench screen as configured
by preferences.

1.362 bum_processor

```
Functions: Processor & ExecVersion
------------------------------------------------------------------------------
Syntax: Processor & ExecVersion
Modes: Amiga

Description:
The two functions Processor & ExecVersion return the relevant
information about the system the program is running on. The values
returned are as follows:

  ExecVersion   OS Release    Processor   Part#
        ------------+------------   -----------+---------------
        33          |  1.2          0          | 68000
        34?         |  1.3          1          | 68010
        36          |  2.0          2          | 68020
        39          |  3.0          3          | 68030
              |               4          | 68040
```

1.363 bum_newdrawingcommands

```
                                  New drawing commands included in BUM #4:


                Poly

                Polyf

                BitPlanesBitMap

                ClipBlit
```

1.364 bum_polypolyf

```
Statement: Poly & Polyf
------------------------------------------------------------------------------
Syntax: Poly  numpoints,*coords.w,color
        Polyf numpoints,*coords.w,color[,color2]
Modes: Amiga/Blitz

Description:
Poly & Polyf are bitmap based commands such as Box and Line. They draw
polygons (unfilled and filled respectively) using coordinates from an
array or newtype of words. Polyf has an optional parameter color2, if
used this colour will be used if the coordinates are listed in anti-
clockwise order, useful for 3D type applications. If color2= -1 then the
polygon is not drawn if the verticies are listed in anti-clockwise
order.

Example:
```

```
  NEWTYPE .tri:x0.w:y0:x1:y1:x2:y2:End NEWTYPE
  BLITZ
  BitMap 0,320,256,3
  Slice 0,44,3:Show 0
  While Joyb(0)=0
        a.tri\x0=Rnd(320),Rnd(256),Rnd(320),Rnd(256),Rnd(320),Rnd(256)
    Polyf 3,a,1+Rnd(7)
  Wend
```

1.365 bum_bitplanesbitmap

```
Statement: BitPlanesBitMap
------------------------------------------------------------------------
Syntax: BitPlanesBitMap SrcBitMap, DestBitMap, PlanePick
Modes: Amiga/Blitz

Description:
BitPlanesBitMap creates a 'dummy' bitmap from the SrcBitMap with only
the bitplanes specified by the PlanePick mask. This is useful for shadow
effects etc. where blitting speed can be speed up because of the fewer
bitplanes involved
```

1.366 bum_clipblit

```
Statement: ClipBlit
------------------------------------------------------------------------
Syntax: ClipBlit Shape#,X,Y
Modes: Amiga/Blitz

Description:
ClipBlit is the same as the Blit command except ClipBlit will clip the
shape to the inside of the used bitmap, all blit commands in Blitz2 are
due to be expanded with this feature.
```

1.367 bum_windowlibadd

```
                                        Window Library Additions

                                The Commands:


                    Window

                    PositionSuperBitmap

                    GetSuperBitmap

                    PutSuperBitmap

                    WTitle
```

```
CloseWindow

WPrintScroll

WBlit

BitMapToWindow

EventCode

EventQualifier
```

1.368 bum_window

```
Statement: Window
------------------------------------------------------------------------
Syntax: Window Window#,x,y,width,height,flags,title$,dpen,bpen[,gadgetlist#
        [,bitmap#]]

The Window library has been extended to handle super bitmap windows.
Super-BitMap windows allow the window to have it's own bitmap which can
actually be larger than the window. The two main benefits of this
feature are the window's ability to refresh itself and the ability to
scroll around a large area "inside" the bitmap.

To attach a BitMap to a Window set the SuperBitMap flag in the flags
field and include the BitMap# to be attached.
```

1.369 bum_positionsuperbitmap

```
Statement: PositionSuperBitMap
------------------------------------------------------------------------
Syntax: PositionSuperBitMap x,y

PositionSuperBitMap is used to display a certain area of the bitmap in a
super bitmap window.

Example:

     ;
     ; super bitmap example
     ;

     ;create large bimtap for our superbitmap window

     width=320:height=200
     BitMap 0,width,height,2
     Circlef 160,100,160,100,1 : Box 0,0,width-1,height-1,3

     FindScreen 0
```

```
;two sliders for the borders (see new gadget flags next page)

PropGadget 0,3,-8,$18000+4+8+64,1,-20,8
PropGadget 0,-14,10,$11000+2+16+128,2,12,-20

;reporting of mousemoves means we can track the propgadget as it is
 moved

AddIDCMP $10
SizeLimits 32,32,width+22,height+20
Window 0,0,0,100,100,$1489,"HELLO",1,2,0,0
Gosub drawsuper
Repeat
  ev.l=WaitEvent
  If ev=2 Then Gosub dosize
  If ev=$20 Then Gosub domove
Until ev=$200
End

dosize:
  SetHProp 0,1,posx/width,InnerWidth/width
  SetVProp 0,2,posy/height,InnerHeight/height
  Redraw 0,1:Redraw 0,2:Goto drawsuper

domove:
  Repeat:Gosub drawsuper:Until WaitEvent<>$10:Return

drawsuper:
  ww=width-InnerWidth:hh=height-InnerHeight
  posx=QLimit(HPropPot(0,1)*(ww+1),0,ww)
  posy=QLimit(VPropPot(0,2)*(hh+1),0,hh)
  PositionSuperBitMap posx,posy
Return
```

1.370 bum_getputsuperbitmap

```
Statement: GetSuperBitMap & PutSuperBitMap
------------------------------------------------------------------------------
Syntax: GetSuperBitMap & PutSuperBitMap

After rendering changes to a superbitmap window thebitmap attached can
also be updated with the GetSuperBitMap. After rendering changes to a
bitmap the superbitmap window can be refreshed with the PutSuperBitMap
command. Both commands work with the currently used window.
```

1.371 bum_wtitle

```
Statement: WTitle
------------------------------------------------------------------------------
Syntax: WTitle windowtitle$,screentitle$

WTitle is used to alter both the current window's title bar and it's
```

screens title bar. Useful for displaying important stats such as program status etc.

1.372 bum_closewindow

```
Statement: CloseWindow
------------------------------------------------------------------------
Syntax: CloseWindow Window#
```

CloseWindow has been added for convenience. Same as Free Window but a little more intuitive (added for those that have complained about such matters).

1.373 bum_wprintscroll

```
Statement: WPrintScroll
------------------------------------------------------------------------
Syntax: WPrintScroll
```

WPrintScroll will scroll the current window upwards if the text cursor is below the bottom of the window and adjust the cursor accordingly. Presently WPrintScroll only works with windows opened with the gimme00 flag set (#gimmezerozero=$400).

1.374 bum_wblit

```
Statement: WBlit
------------------------------------------------------------------------
Syntax: WBlit Shape#,x,y
```

WBlit can be used to blit any shape to the current window. Completely system friendly this command will completely clip the shape to fit inside the visible part of the window. Use GimmeZeroZero windows for clean clipping when the window has title/sizing gadgets.

1.375 bum_bitmaptowindow

```
Statement: BitMaptoWindow
------------------------------------------------------------------------
Syntax: BitMaptoWindow Bitmap#,Window#[,srcx,srcy,destx,desty,wid,height]
```

BitMaptoWindow will copy a bitmap to a window in an operating system friendly manner (what do you expect). The main use of such a command is for programs which use the raw bitmap commands such as the 2D and Blit libraries for rendering bitmaps quickly but require a windowing environment for the user inyerface.

1.376 bum_eventcq

```
Functions: EventCode & EventQualifier
------------------------------------------------------------------------------
Syntax: EventCode & EventQualifier

EventCode returns the actual code of the last Event received by your
program, EventQualifier returns the contents of the Qualifier field. Of
use with the new GadTools library and some other low level event
handling requirements.
```

1.377 bum_gadgetadd

```
Gadget Library Additions
------------------------------------------------------------------------------
Five new flags have been added when defining gadgets in Blitz2. The
first four are for attaching the gadget to one of the windows borders,
the GZZGADGET flag is for attaching the gadget to the "outer" rastport/
layer of a gimme zero zero window.

#RIGHTBORDER    $1000
#LEFTBORDER     $2000
#TOPBORDER      $4000
#BOTTOMBORDER   $8000
#GZZGADGET      $10000

PropGadgets have been upgraded to take advantage of the 2.0 "newlook"
when/if available.
```

1.378 bum_toggle

```
Statement: Toggle
------------------------------------------------------------------------------
Syntax: Toggle GadgetList#,Id [,On|Off]

The Togggle command in the gadget library has been extended so it will
actually toggle a gadgets status if the no On|Off parameter is missing.
```

1.379 bum_screenlibadd

```
                                        Screen Library Additions

                                New commands:


                    CloseScreen

                    HideScreen
```

BeepScreen

MoveScreen

ScreenTags

1.380 bum_closescreen

```
Statement: CloseScreen
------------------------------------------------------------------------------
Syntax: CloseScreen Screen#

CloseScreen has been added for convenience. Same as Free Screen but a
little more intuitive (especially for those that have complained about
such matters (yes we care)).
```

1.381 bum_hidescreen

```
Statement: HideScreen
------------------------------------------------------------------------------
Syntax: HideScreen Screen#

Move Screen to back of all Screens open in the system.
```

1.382 bum_beepscreen

```
Statement: BeepScreen
------------------------------------------------------------------------------
Syntax: BeepScreen Screen#

Flash specified screen.
```

1.383 bum_movescreen

```
Statement: MoveScreen
------------------------------------------------------------------------------
Syntax: MoveScreen Screen#,deltax,deltay

Move specified screen by specified amount. Good for system friendly
special effects.
```

1.384 bum_screentags

```
Statement: ScreenTags
------------------------------------------------------------------------------
Syntax: ScreenTags Screen#,Title$ [&TagList] or [[,Tag,Data]...]

Full access to all the Amiga's new display resoutions is now available
in Amiga mode by use of the Screen Tags command. The following tags are
of most interest to Blitz2 programmers: (see autodocs)

#Left=$80000021
#Top=$80000022
#Width=$80000023
#Height=$80000024
#Depth=$80000025
#DetailPen=$80000026
#_BlockPen=$80000027
#Title=$80000028
#Colors=$80000029
#ErrorCode=$8000002A
#Font=$8000002B
#SysFont=$8000002C
#Type=$8000002D
#BitMap=$8000002E
#PubName=$8000002F
#PubSig=$80000030
#PubTask=$80000031
#DisplayID=$80000032
#DClip=$80000033
#Overscan=$80000034
#Obsolete1=$80000035

#ShowTitle=$80000036
#Behind=$80000037
#_Quiet=$80000038
#AutoScroll=$80000039
#Pens=$8000003A
#FullPalette=$8000003B
#ColorMapEntries=$8000003C
#Parent=$8000003D
#Draggable=$8000003E
#Exclusive=$8000003F

#SharePens=$80000040
#BackFill=$80000041
#_Interleaved=$80000042
#Colors32=$80000043
#VideoControl=$80000044
#FrontChild=$80000045
#BackChild=$80000046
#LikeWorkbench=$80000047
#Reserved=$80000048

;
; open super wide screen with overscan set for smooth horizontal scroll
; for 2.0 and above with amigalibs.res in resident
;
```

```
#_BitMap=$8000002E:#_Overscan=$80000034:#_Width=$80000023:
#_Height=$80000024

BitMap 0,1280,512,2:Circlef 320,256,256,1

ScreenTags 0,"TEST",#_BitMap,Addr BitMap(0),#_Overscan,1,#_Width,640,
                    #_Height,512

*vp.ViewPort=ViewPort(0)

While Joyb(0)=0
     VWait
     *vp\DxOffset=-SMouseX,-SMouseY
     ScrollVPort_ *vp
Wend
```

1.385 bum_palettelibadd

Palette Library Additions@{fg text}

The Palette library has been modified in BUM5 for two reasons. Firstly, it was impossible to perform custom fades using two palettes as the Use Palette command affected the current Slice or Screen. Also with the advent of the Display library the extra properties of the Use Palette command (copy colors to current Slice or Screen) became unwanted.

New commands:

ShowPalette

NewPaletteMode

The ShowPalette command has been added to replace the above functionality removed from the Use Palette command. Also, for compatability reasons NewPaletteMode On is used for enabling the above modifications (default is off).

1.386 bum_showpalette

```
Statement: ShowPalette
------------------------------------------------------------------------------
Syntax: ShowPalette Palette#
```

ShowPalette replaces Use Palette for copying a palette's colours to the current Screen or Slice.

1.387 bum_newpalettemode

```
Statement: NewPaletteMode
------------------------------------------------------------------------------
Syntax: NewPaletteMode On|Off

The NewPaletteMode flag has been added for compatibility with older
Blitz2 programs. By setting NewPaletteMode to On the Use Palette command
merely makes the specified palette the current object and does not try
to copy the colour information to the current Screen or Slice.
```

1.388 bum_newdisplaylibrary

```
                                    The New Display Library (#displaylib=143)

The new display library is an alternative to the slice library.
Instead of extending the slice library for AGA support a completely new
display library has been developed.

Besides support for extended sprites, super hires scrolling and 8
bitplane displays a more modular method of creating displays has been
implemented with the use of CopLists. CopLists need only be initialised
once at the start of the program. Displays can then be created using any
combination of CopLists and most importantly the CreateDisplay command
does not allocate any memory avoiding any memory fragmenting problems.
The new display library is for non-AGA displays also.

                            Display Library Commands:


                  InitCopList

                  DisplayPalette

                  CreateDisplay

                  DisplayControls

                  DisplayBitmap

                  DisplayAdjust

                  DisplaySprite
```

1.389 bum_initcoplist

```
Statement: InitCopList
------------------------------------------------------------------------------
Syntax: InitCopList CopList#,ypos,height,type,sprites,colors,
        customs[,widthadjust]

InitCopList is used to create a CopList for use with the CreateDisplay
```

```
command. The ypos, height parameters define the section of screen.
Sprites, colors and customs will allocate instructions for that many
sprites (always=8!) colors (yes, as many as 256!) and custom copper
instructions (to be used by the new DisplayFX library currently in
devlopment).

The widthadjust parameter is currently not implemented, for display
widths other than standard see the DisplayAdjust command. The following
constants make up the type parameter, add the number of bitplanes to the
total to make up the type parameter.

#smoothscroll=$10  #dualplayfield=$20 #extrahalfbrite=$40  #ham=$80
#lores=$000        #hires=$100        #super=$200
#loressprites=$400 #hiressprites=$800 #supersprites=$c00
#fmode0=$0000      #fmode1=$1000      #fmode2=$2000         #fmode3=$3000

For displays on non-AGA machines only #fmode0 and #loressprites are
allowed. More documentation, examples and fixes will be published soon
for creating displays.
```

1.390 bum_createdisplay

```
Statement: CreateDisplay
------------------------------------------------------------------------
Syntax: CreateDisplay CopList#[,CopList#..]

CreateDisplay is used to setup a new screen display with the new display
library. Any number of CopLists can be passed to CreateDisplay although
at present they must be in order of vertical position and not overlap.
CreateDisplay then links the CopLists together using internal pointers,
bitmaps, colours and sprites attached to coplists are not affected.
```

1.391 bum_displaybitmap

```
Statement: DisplayBitMap
------------------------------------------------------------------------
Syntax: DisplayBitMap CopList#,bmap[,x,y] [,bmap[,x,y]]

The DisplayBitMap command is similar in  usage to the slice libraries'
show commands. Instead of different commands for front and back
playfields and smooth scroll options there is only the one DisplayBitMap
command with various parameter options. With AGA machines, the x
positioning of lores and hires coplists uses the fractional part of the
x parameter for super smooth scrolling.The CopList must be initialised
with the smooth scrolling flag set if the x,y parameters are used, same
goes for dualplayfield.
```

1.392 bum_displaysprite

```
                Statement: DisplaySprite
------------------------------------------------------------------------
Syntax: DisplaySprite CopList#,Sprite#,X,Y,Sprite Channel

DisplaySprite is similar to the slice libraries ShowSprite command with
the added advantage of super hires positioning and extra wide sprite
handling.

See also
                 SpriteMode
                .
```

1.393 bum_displaypalette

```
Statement: DisplayPalette
------------------------------------------------------------------------
Syntax: DisplayPalette CopList#,Palette# [,coloroffset]

DisplayPalette copies colour information from a Palette to the CopList
specified.
```

1.394 bum_displaycontrols

```
Statement: DisplayControls
------------------------------------------------------------------------
Syntax: DisplayControls CopList#,BPLCON2,BPLCON3,BPLCON4

DisplayControls allows access to the more remote options available in
the Amiga's display system. The following are the most important bits
from these registers (still unpublished by Commodore!*()@GYU&^)

Bit| BPLCON2                    | BPLCON3                | BPLCON4
---+----------------------------+------------------------+-----------------
15 | *                          | BANK2 * active colour  | BPLAM7 xor with
   |                            |         bank           |        bitplans
---+----------------------------+------------------------+-----------------
14 | ZDBPSEL2  which bitplane   | BANK1 *                | BPLAM6 DMA for
   |           for ZD           |                        |        altering
---+----------------------------+------------------------+-----------------
13 | ZDBPSEL1                   | BANK0 *                | BPLAM5 effective
   |                            |                        |        colour
---+----------------------------+------------------------+-----------------
12 | ZDBPSEL0                   | PF2OF2  col-offset for | BPLAM4 look up
   |                            |         playfield 2    |
---+----------------------------+------------------------+-----------------
11 | ZDBPEN    makes above bp   | PF2OF1                 | BPLAM3
   |           hit ZD           |                        |
---+----------------------------+------------------------+-----------------
10 | ZDCTEN    ZD is bit#15 of  | PF2OF0                 | BPLAM2
   |           colour           |                        |
---+----------------------------+------------------------+-----------------
```

```
09 | KILLEHB   *                | LOCT * palette hi/lo   | BPLAM1
   |                            |        nibble mode     |
---+----------------------------+------------------------+-----------------
08 | RDRAM=0   *                |                        | BPLAM0
---+----------------------------+------------------------+-----------------
07 | SOGEN     ! sync on green  + SPRES1 * sprites-      | ESPRM7 high order
   |                            |          resolution    |        color
---+----------------------------+------------------------+-----------------
06 | PF2PRI    H playfield 1/2  | SPRES0 *               | ESPRM6 offset for
   |           priority         |                        |        even
---+----------------------------+------------------------+-----------------
05 | PF2P2     H playfield/     | BRDRBLANK border is    | ESPRM5 sprites
   |           sprite priority  |           black        |
---+----------------------------+------------------------+-----------------
04 | PF2P1                      | BRDNTRAN  border       | ESPRM4
   |                            |           hits ZD      |
---+----------------------------+------------------------+-----------------
03 | PF1P0                      |                        | OSPRM7 high order
   |                            |                        |        color
---+----------------------------+------------------------+-----------------
02 | PF1P2                      | ZDCLCKEN  ZD=14Mhz     | OSPRM6 offset for
   |                            |           clock        |        odd
---+----------------------------+------------------------+-----------------
01 | PF1P1                      | BRDSPRT   sprites in   | OSPRM5 sprites
   |                            |           borders!     |
---+----------------------------+------------------------+-----------------
00 | PF1P0                      | EXTBLKEN  wo blank     | OSPRM4
   |                            |           outputl      |
---+----------------------------+------------------------+-----------------

!  - Don't touch
H  - See standard hardware reference manual
*  - controlled by display library
ZD - any reference to ZD is only a guess (just sold my genlock)
```

1.395 bum_displayadjust

```
Statement: DisplayAdjust
------------------------------------------------------------------------------
Syntax: DisplayAdjust CopList#,fetchwid,ddfstrt,ddfstop,diwstrt,diwstop

Temporary control of display registers until I get the widthadjust
parameter working with InitCopList. Currently only standard width
displays are available but you can modify the width manually (just stick
a screwdriver in the back of your 1084) or with some knowledge of
Commodores AGA circuitry.

Anyway, before I start going on about why they couldn't just give us
byte per pixel instead of 8 darn bitplanes (CD32 to the rescue!) see
the cover disk for more information...
```

1.396 bum_newasllibrary

The New ASL Library (#myasllib=80)

Our policy until now has been that we would only place emphasis on 1.3 compatible commands unless of course they had to do with AGA. Then again I don't even have a LoadWB in my startup-sequence! So instead of complaining I spent an uncomfortable week adding the following 2.0 above specific commands to Blitz2.

And as for those with 1.3 and want new ROMS? BURN BABY BURN...

The commands:

ASLFileRequest

ASLFontRequest

ASLScreenRequest$

1.397 bum_aslfilerequest$

```
Function: ASLFileRequest$
--------------------------------------------------------------------------
Syntax: ASLFileRequest$ (Title$,Pathname$,Filename$ [,Pattern$]
                         [,x,y,w,h] )
```

The ASL File Requester is nice. Except for the highlight bar being invisible on directories you get to use keyboard for everything, stick in a pattern$ to hide certain files and of course you get what ever size you want. I made it call the Blitz2 file requester if the program is running under 1.3 (isn't that nice!). There is a fix that patches the ReqTools file requester but that doesn't have the date field.

I couldn't get the Save-Only tag or the "Create Directory" option working maybe next upgrade.

EXAMPLE:

```
MaxLen pa$=192
MaxLen fi$=192

FindScreen 0

f$=ASLFileRequest$("test",pa$,fi$,"#?.bb",0,0,640,256)

If f$
     NPrint f$
Else
     NPrint "failed"
EndIf

MouseWait
```

1.398 bum_aslfontrequest

```
Function: ASLFontRequest
------------------------------------------------------------------------
Syntax: ASLFontRequest (enable_flags)

The ASL Font Requester is also pretty useful. The flags parameter
enables the user to modify the following options:

#pen=1:#bckgrnd=2:#style=4:#drawmode=8:#fixsize=16

It doesn't seem to handle colour fonts, no keyboard shortcuts so perhaps
patching ReqTools is an option for this one. The following code
illustrates how a .fontinfo structure is created by a call to
ASLFontRequest (just like programming in a high level language man!).

EXAMPLE:

NEWTYPE .fontinfo
     name.s
     ysize.w
     style.b:flags.b
     pen1.b:pen2:drawmode:pad
End NEWTYPE

FindScreen 0

*f.fontinfo=ASLFontRequest(15)

If *f
     NPrint *f\name
     NPrint *f\ysize
     NPrint *f\pen1
     NPrint *f\pen2
     NPrint *f\drawmode
Else
     NPrint "cancelled"
EndIf

MouseWait
```

1.399 bum_aslscreenrequest

```
Function: ASLScreenRequest
------------------------------------------------------------------------
Syntax: ASLScreenRequest (enable_flags)

Those who are just getting to grips with 2.0 and above will find this
command makes your programs look really good, however I haven't got time
to explain the difficulties of developing programs that work in all
screen resolutions (what are ya?).

EXAMPLE:
```

```
#width=1:#height=2:#depth=4:#overscan=8:#scroll=16

NEWTYPE .screeninfo
     id.l
     width.l
     height.l
     depth.w
     overscan.w
     autoscroll.w
     bmapwidth.l
     bmapheight.l
End NEWTYPE

FindScreen 0

*sc.screeninfo=ASLScreenRequest(31)

If *sc
     NPrint *sc\width," ",*sc\height," ",*sc\depth
Else
     NPrint "cancelled"
EndIf
MouseWait
```

1.400 bum_newgadtoolslibrary

The New GadTools Library (# mygadtoolslib=141)

GadTools is a 2.0 and greater extension to the operating system that gives the Amiga programmer a few extra enhancements to create juicy user interfaces with. Instead of listing each as a separate command this issue I'll just add a brief description and a relevant taglist to each of the 12 gadgets.

The Commands:

AttachGTList

GTBevelBox

GTTags

GTChangeList

GTGadPtr

GTSetAttrs

You are allowed both standard gadgets and GadTools ones in the same window, of course id clashes must be avoided and unlike standard gadgets, gadtools gadgets are attached to the Window after it is open with the AttachGTList command.

GTButton GTList#,id,x,y,w,h,Text$,flags

Same as Blitz2's TextGadget but with the added flexibility of placing the label Text$ above, below to the left or right of the button (see flags).

GTCheckBox GTList#,id,x,y,w,h,Text$,flags

A box with a check mark that toggles on and off, best used for options that are either enabled or disabled.

GTCycle GTList#,id,x,y,w,h,Text$,flags,Options$

Used for offering the user a range of options, the options string should be a list of options separated by the | character eg. "HIRES } LORES } SUPERHIRES"

GTInteger GTList#,id,x,y,w,h,Text$,flags,default

A string gadget that allows only numbers to be entered by the user.

GTListView GTList#,id,x,y,w,h,Text$,flags,list()

The ListView gadget enaables the user to scroll through a list of options. These options must be contained in a string field of a Blitz2 linked list. Currently this string field must be the second field, the first being a word type.

GTMX GTList#,id,x,y,w,h,Text$,flags,Options$

GTMX is an exclusive selection gadget , the Options$ is the same as GTCycle in format, GadTools then displays all the options in a vertical list each with a hi-light beside them.

GTNumber GTList#,id,x,y,w,h,Text$,flags,value

This is a readonly gadget (user cannot interact with it) used to display numbers.

GTPalette GTList#,id,x,y,w,h,Text$,flags,depth

Creates a number of coloured boxes relating to a colour palette,

GTScroller GTList#,id,x,y,w,h,Text$,flags,Visible,Total

A prop type gadget for the user to control an amount or level, is accompanied by a set of arrow gadgets.

GTSlider GTList#,id,x,y,w,h,Text$,flags,Min,Max

Same as Scroller but for controlling the position of display inside a larger view.

GTString GTList#,id,x,y,w,h,Text$,flags,MaxChars

A standard string type gadget

```
GTText GTList#,id,x,y,w,h,Text$,flags,Display$

A read only gadget (see GTNumber) for displaying text messages.

The parameters x,y,w,h refer to the gadgets position and size, the Text$
is the label as referred to above. The flags field is made up of the
following fields:

#_LEFT=1  ;positioning of the optional gadget label Text$
#_RIGHT=2
#_ABOVE=4
#_BELOW=8
#_IN=$10
#_High=$20       ;highlight
#_Disable=$40    ;turned off
#_Immediate=$80  ;activate on gadgetdown
#_BoolValue=$100 ;checkbox on
#_Scaled=$200    ;scale arrows for slider
#_Vertical=$400  ;make slider/scroller vertical
```

1.401 bum_attachgtlist

```
Statement: AttachGTList
------------------------------------------------------------------------------
Syntax: AttachGTList GTList#,Window#

The AttchGTList command is used to attach a set of GadTools gadgets to a
Window after it has been opened.
```

1.402 bum_gttags

```
Statement: GTTags
------------------------------------------------------------------------------
Syntax: GTTags Tag,Value [,Tag,Value...]

The GTTags command can be used prior to initialisation of any of the 12
gadtools gadgets to preset any relevant Tag fields. The following are
some useful Tags that can be used with GTTags:

#tag=$80080000
#GTCB_Checked=#tag+4      ; State of checkbox
#GTLV_Top=#tag+5          ; Top visible item in listview
#GTLV_ReadOnly=#tag+7     ; Set TRUE if listview is to be ReadOnly
#GTMX_Active=#tag+10      ; Active one in mx gadget
#GTTX_Text=#tag+11        ; Text to display
#GTNM_Number=#tag+13      ; Number to display
#GTCY_Active=#tag+15      ; The active one in the cycle gad
#GTPA_Color=#tag+17       ; Palette color
#GTPA_ColorOffset=#tag+18 ; First color to use in palette
#GTSC_Top=#tag+21         ; Top visible in scroller
#GTSC_Total=#tag+22       ; Total in scroller area
```

```
#GTSC_Visible=#tag+23     ; Number visible in scroller
#GTSL_Level=#tag+40       ; Slider level
#GTSL_MaxLevelLen=#tag+41 ; Max length of printed level
#GTSL_LevelFormat=#tag+42 ;* Format string for level
#GTSL_LevelPlace=#tag+43  ;* Where level should be placed
#GTLV_Selected=#tag+54    ; Set ordinal number of selected
#GTMX_Spacing=#tag+61     ;* Added to font height to
```

All of the above except for those marked * can be set after initialisation of the Gadget using the GTSetAttrs command. The following is an example of creating a slider gadget with a numeric display:

```
f$="%2ld" : GTTags #GTSLLevelFormat, &f$, #GTSLMaxLevelLen, 4
GTSlider 2,10,320,120,200,20,"GTSLIDER",2,0,10
```

1.403 bum_gtgadptr

```
Function: GTGadPtr
------------------------------------------------------------------------
Syntax: GTGadPtr (GTList#,id)
```

GTGadPtr returns the actual location of the specified GadTools gadget in memory.

1.404 bum_gtbevelbox

```
Statement: GTBevelBox
------------------------------------------------------------------------
Syntax: GTBevelBox GTList#,x,y,w,h,flags
```

GTBevelBox is the GadTools library equivalent of the Borders command and can be used to render frames and boxes in the currently used Window.

1.405 bum_gtchangelist

```
Statement: GTChangeList
------------------------------------------------------------------------
Syntax: GTChangeList GTList#,id [ ,List() ]
```

GTChangeList must be used whenever a List attached to a GTListView needs to be modified. Call GTChangeList without the List() parameter to free the List, modify it then reattache it with another call to GTChangeList this time using the List() parameter.

1.406 bum_gtsetattrs

```
Statement: GTSetAttrs
------------------------------------------------------------------------------
Syntax: GTSetAttrs GTList#,id [,Tag,Value...]

GTSetAttrs can be used to modify the status of certain GadTools gadgets
with the relevant Tags. See GTTags for more information on the use of
Tags with the GadTools library.
```

1.407 bum_printerlib

```
                                                PRINTER LIBRARY

This is a library for using the printer.device!!! There are only four
commands included, but I think these are the most wanted ones...

                                    The Commands:


                CheckPrt

                PrtCommand

                PrtText

                HardCopy
```

1.408 bum_checkprt

```
Statement: CheckPrt
------------------------------------------------------------------------------
Syntax: status.b=CheckPrt

Modes : AMIGA/BLITZ

Description:
Checks the state of the Printer and return it.

status: -1 = Printer Off
        -3 = Printer Offline
        -4 = Printer On
Bugs:

I had exculded this Routine, because it doesn' t worked 100%...I have
now reincluded it and compiled with the newest version of the BB2
compiler...It seems that it now works 100%...
```

1.409 bum_prtcommand

```
Statement: PrtCommand
------------------------------------------------------------------------------
Syntax: PrtCommand Command,Para1,Para2,Para3,Para4

Modes : AMIGA

Description:
Send a ESC-Sequence to the printer.

Command: Escape-Sequence

Para1 - Para 4: Parameters for ESC-Sequence

Beispiel: PrtCommand 6,0,0,0,0 ;Kursiv on
          PrtCommand 7,0,0,0,0 ;Kursiv off
```

1.410 bum_prttext

```
Statement: PrtText
------------------------------------------------------------------------------
Syntax: PrtText <STRING>

Mode  : AMIGA

Description:

PrtText: Prints the text 'STRING' at your printer...
         It' s the same like WriteFile(0,"PRT:"), but why have the AMIGA
         a own printer.device...
         So I think we should use it, right?
```

1.411 bum_hardcopy

```
Statement: Hardcopy
------------------------------------------------------------------------------
Syntax: Hardcopy ScreenPointer,X,Y,Width,Height,PrtWidth,PrtHeight,Flags

Mode  : AMIGA

Description:
Prints the screen or a part of it at your printer.

ScreenPointer: Adress of the screen (Addr Screen(x))
X............: \ Are the corners of the screen where we
Y............: / want to start printing it to paper...
Width........: Width of the screen part you want to print
Height.......: Height of the screen part you want to print
PrtWidth.....: Width of the print(on the paper)
PrtHeight....: Height of the print(on the paper)
Flags........: Printerflags...
               Have a look to a documantation aout it, the most
```

```
               importent(?) ones:

               $40  - Centre graphic
               $10  - Weidth= Maximum
               $20  - Height = Maximum
               $100,$200,$300,$400 - Printing quality($400 = Heighest)

Bugs:

Hm, it seems that the flags are not 100% taken...I haven' t find out
why, but might be you do...I still working on it! But however, it
works..

BTW: This library is copyright 1992/93 by Andre Bergmann.
     Use it on your own risk, I don' t take the responsibility for using
     it! This source is PD, feel free to update it!
     Please send me updates done by you and Bug Reports!

Andre Bergmann
Am Pannesbusch 39a
42281 Wuppertal 2
Germany
Tel: 0049/0202/702606
```

1.412 bum_consolelib

```
                                                  CONSOLE LIBRARY
------------------------------------------------------------------------------

                        ConsoleLib (a little buggy yet):


                  OpenConsole

                  PrintCon

                  NPrintCon

                  CloseConsole
```

1.413 bum_openconsole

```
Command: OpenConsole
------------------------------------------------------------------------------
Syntax : OpenConsole Window#,Console#

Description:
Open a CON: Port for the window, so ANSI output is possible.
```

1.414 bum_printcon

```
Command: PrintCon
--------------------------------------------------------------------------------
Syntax:  PrintCon Console#,Expression

Description:
Write text to window-console.
```

1.415 bum_nprintcon

```
Command: NprintCon
--------------------------------------------------------------------------------
Syntax:  NPrintCon Console#,Expression

Description:
Write text to window-console, and add a Linefeed.
```

1.416 bum_closeconsole

```
Command: CloseConsole
--------------------------------------------------------------------------------
Syntax:  CloseConsole Console#

Description:
Close the CON: Port, but NOT the window itself!

Note:
These command work if you use them only for ONE window...If you wanna
use the CON's for more windows, so that object are use it doesn' t work
anymore... Ya see, I need ya help, please...
```

1.417 bum_crunchlib

```
                                                CRUNCH LIBRARY

                                Available Commands:


                        Implode

                        Deplode

                        CrMDecrunch

                        PPDecrunch
```

1.418 bum_implode

```
Function: Implode
------------------------------------------------------------------------------
Syntax:   Implode Error/CrunchedLen=Bufferadr,DataLen,?Callback

Description:
Crunch a buffer using the Imploder algorythm. The ?Callback could be a 0
for no own routine or a pointer to an own routine for display or abort
checking. If the Imploder command jump to the callback in register d0
the current crunch position is presend. The callback itself have to
return True for cont crunching or False for a break!

The command return the crunched buffer len or of course a break or an
error. If a <0 is returned the callback returned a userbreak, a return
of 0 means an error happens. Else the new buffer len is returned.
```

1.419 bum_deplode

```
Function: Deplode
------------------------------------------------------------------------------
Syntax:   Deplode Success=Startadr

Description:
Decrunch a Imploder-Crunched buffer. There is no need to give the
crunched buffer len to the command, imploder handle this by itself. But
be careful, if the alloceted buffer hasn' t enought space the program
crash.

    The buffer has to have a header like this:

    Type  Offset  Contents                Function
    LONG  0       "IMP!"                  To recongnize crunched files
    LONG  4       Original Len            Datalen before packing
    LONG  8       Crunched Len-$32         Datalen after packing

    So $a is the start of the datas...

    The decrunch routine NEED this header to decrunch!!!
    Do memoryallocation for the buffer using a allocate for the
    Startadr+$4 size.
```

1.420 bum_crmdecrunch

```
Command: CrMDecrunch
------------------------------------------------------------------------------
Syntax:  CrMDecrunch Bufferadr [,Destinationadr]

Description:
If the only parameter is the Bufferadr this routine works like the
Deplode command, but decrunch a CrunchMania crunched file. If you use it
with to parameters the decrunch will be done from Bufferadr to
Destinationadr, so 2 buffers have to been allocated.
```

```
    The header for CrunchMania files have to look like this:

    Type  Offset  Contents                    Function
    ------------------------------------------------------------------------
    LONG  0       "CrM!"/"CrM2"               To recongnize crunched
                                              files.

    WORD  4       Minimum Security Distance   To savely decrunch Data
                                              when Source AND Dest is
                                              in the same Memoryblock.

    LONG  6       Original Len                Datalen before packing

    LONG  10 ($a) Crunched Len                Datalen after packing
                                              without header.

    So at $d is the data startadress.

    This header is NEEDED for decrunching!
```

1.421 bum_ppdecrunch

```
Command: PPDecrunch
------------------------------------------------------------------------------
Syntax: PPDecrunch Bufferadr,Bufferend,Destinationadr

Description:
This command decrunch a PowerPacker crunched file. PowerPacker need two
buffers for decrunching. Also the lenght of the buffer must be given!

Sorry, I can' t find my PowerPacker archive where the header is
descripted...Like Imploder and CrunchMania PowerPacker also support his
own file header.

Please have a look at the PowerPacker(.library) documentation!
```

1.422 bum_localelib

```
                                                    LOCALE LIBRARY
------------------------------------------------------------------------------

                                Available commands:


                    IsLocale

                    UseCatalog

                    FreeCatalog

                    GetLocaleStr
```

1.423 bum_islocale

```
Function: IsLocale
------------------------------------------------------------------------------
Syntax:   result=IsLocale

Description:
There is now way to check the ROM-Version of the Kickstart for locale
presents. Both, OS 2.0 and 2.1 have the ROM-Version 37.175! So the
command returns if the locale.library exists on the system.
```

1.424 bum_usecatalog

```
Command: UseCatalog
------------------------------------------------------------------------------
Syntax: UseCatalog Catalogname

Description:
Opens the catalog for your programm. Might by your program is called
BB2Program the catalog should be called BB2Program.catalog. But you're
also able to open a catalog from an other program, like Term.catalog.

The catalog files are in an IFF format!!! Read the Commodore
documentation of it...
```

1.425 bum_freecatalog

```
Command: FreeCatalog
------------------------------------------------------------------------------
Syntax:  FreeCatalog

Description:
Removes the catalog that you opened with UseCatalog.
```

1.426 bum_getlocalestr

```
Function: GetLocaleStr
------------------------------------------------------------------------------
Syntax: GetLocaleStr #StringNumber,DefaultString$

Description:
Read a string from the catalog that you opened with UseCatalog. You have
to give a defaultstring to that command. If the asked string could be
find in the catalog the default string will be returned. Else the string
from the catalog will be returned.
```

1.427 bum_requesterlibrary

```
                                REQUESTER LIBRARY
------------------------------------------------------------------------------

                                Available Commands:

Function: EasyRequest
------------------------------------------------------------------------------
Syntax:   EasyRequest Result=EasyRequest([#Window,]title$,body$,gtext$)

Description:
A intuition system requester will be open. Optional you could give a
window number. The title$ is the displayed string in the top. body$ is
the displayed text in the requester, a Chr$(10) means a linefeed.

gtext$ is the text for the gadgets. Every gadgettext will be added by a
'|'.

Examples:  "Ok" will only display one gadget in the requester.
           "Ok|Cancel" add two gadgets to it.
           "1|2|3|4|5" five gadgets are displayed.
```

1.428 bum_amigasupportlib

```
                                            AMIGA SUPPORT LIBRARY
------------------------------------------------------------------------------

                                Available Commands:


                AllocMem

                FreeMem

                IsEven

                SearchString
```

1.429 bum_allocmem

```
Function: AllocMem
------------------------------------------------------------------------------
Syntax:   MemoryBlock=AllocMem(Size,Type)

Description:
Unlike calling Exec's AllocMem_ command directly Blitz2 will automatically
free any allocated memory when the program ends. Programmers are advised
to use the InitBank command.

Flags that can be used with the memory type parameter are:
```

```
1=public    ;fast is present
2=chipmem
65536=clear ;clears all memory allocated with 0's
```

1.430 bum_freemem

```
Command: FreeMem
--------------------------------------------------------------------------
Syntax:  FreeMem MemoryBlock,ByteSize
         (long)    (long)

Description:
Deallocates memory obtained with AllocMem
```

1.431 bum_iseven

```
Function: IsEven
--------------------------------------------------------------------------
Syntax:   Result=IsEven(Expression)
          (bool)  (byte,word,long)

Description:
Returns true if Expression is even.  Of use when requesting a value from
a user that MUST be even.
```

1.432 bum_searchstring

```
Function: SearchString
--------------------------------------------------------------------------
Syntax:   result=SearchString(StringPointer,StartAddress,BlockLength)
          (long)                (&string)    (long)        (long)

Description:
Finds a string in the given memory block and returns its address.
Returns False otherwise.
```

1.433 bum_elmorelib

```
                    BUM #6 contains almost all the PD commands of Elmore and ↩
                       some more.


                DOS Elmore Library

                Hardware Elmore Library

                Math Elmore Library
```

```
                Array Elmore Library

                Sys Elmore Library

                String Elmore Library

                Library Programming
                                                New in BUM7 :


                Include Library
                                             (For the uninitiated:)

                 NOTE ON FUNCTIONS, STATEMENTS and COMMANDS:
                 ------------------------------------------

"FUNCTIONS" are Blitz2 tokens that require parameters in parentheses,
and return a value: n=ABS(m)

"STATEMENTS" are Blitz2 tokens that only perform an action but do not
return a value.  Their arguments do not require parentheses:
PRINT "HELLO!"

"COMMANDS" are Blitz2 tokens that can be used as either a FUNCTION or
a STATEMENT, depending upon whether the arguments were in parentheses or
not.

[Function form:]
n=REQUEST("TITLE","SELECT YES OR NO","YES|NO")

[Statement form:]
REQUEST "TITLE","SELECT OK TO CONTINUE","OK"
```

1.434 bum_elmoredos

```
                                                    DOS.ELMORE LIBRARY
------------------------------------------------------------------------------


                ChDir

                EntryHour

                PathLock

                EntryMins

                CopyFile

                EntrySecs

                SetCopyBuffer

                EntryComment$
```

NameFile

AnalyzeDisk

MakeDir

DiskUnit

MoreEntries

DiskErrs

EntryName$

DiskCapacity

EntryDir

DiskUsed

EntryBit$

DiskFree

EntrySize

DiskBlocks

EntryDate

DIRECTORY EXAMPLE

1.435 bum_chdir

```
Command: CHDIR
--------------------------------------------------------------------------------
Syntax: CHDIR "Path:"  -or-  IF CHDIR("Path:") Then...

This command will change the current working directory for ALL disk-
related commands.  Used as a function, a value of TRUE will be returned
if the directory change was successful, or FALSE if it was unsuccessful.
```

1.436 bum_pathlock

```
Function: PATHLOCK
--------------------------------------------------------------------------------
Syntax: Lock.l=PATHLOCK

This function will return the BCPL pointer to the lock of the current
directory.  You should NEVER "Unlock_" this lock, but it is useful to
```

```
use command "NameFromLock_" with it to determine the full pathname of
the current directory, for example.  (NOTE: NameFromLock_ requires 2.0
and above!)
```

1.437 bum_copyfile

```
Command: COPYFILE
------------------------------------------------------------------------
Syntax: COPYFILE "First","SECOND"  -or-  IF COPYFILE("FIRST","SECOND") Then...

This command will copy files, much like the CLI command "Copy."  In the
function form, it will return TRUE for success, and FALSE for failure.
Note that the speed at which it copies can be increased by increasing the
"CopyBuffer," which defaults to 8192 bytes.  (See below)
```

1.438 bum_setcopybuffer

```
Statement: SetCopyBuffer
------------------------------------------------------------------------
Syntax: SetCopyBuffer BUFFERSIZE

This statement is used to set the size of the COPYFILE command's memory
buffer.  The default size is 8192 bytes, but this can be adjusted from
256 bytes to nearly all your free memory.  A larger buffer will normally
increase the speed at which the COPYFILE command operates, but only up to
the size of the largest file you're copying.  For example, if the largest
file you need to copy is 25000 bytes, then it will be useless to set the
COPYBUFFER above 25000.
```

1.439 bum_namefile

```
Command: NAMEFILE
------------------------------------------------------------------------
Syntax: NAMEFILE "Oldname","Newname"  -or-
        IF NAMEFILE("Oldname","Newname") Then...

This command returns FALSE for failure, TRUE for success:
The file "oldname" is renamed to "newname," if possible, and may be moved
to other directories within the same volume.  It is not yet possible to
use NAMEFILE to move a file from one volume to another, however.
```

1.440 bum_makedir

```
Command: MAKEDIR
------------------------------------------------------------------------
Syntax: NAMEFILE "Path:Dir"  -or-  If NAMEFILE("Path:Dir") Then...
```

This command will attempt to create a new directory with the given pathname. It is only possible to create one level at a time, however. For example, MAKEDIR will fail if you attempt to MAKEDIR "RAM:New/Data" if the directory "RAM:New" does not yet exist. Used as a function, MAKEDIR returns TRUE for success, and FALSE for failure.

1.441 bum_moreentries

Command: MOREENTRIES

Syntax: MOREENTRIES -or- If MOREENTRIES Then...

This command will read the next entry in the current directory for inspection with other "ENTRY" commands. Used within a loop, it is easy to read an entire directory with these commands, similar to the "DIR" or "LIST" commands of AmigaDOS. (See below. An example follows)

1.442 bum_entryname$

Function: ENTRYNAME$

Syntax: n$=ENTRYNAME$

This function returns the name of the current directory entry. If used before the fist "MOREENTRIES" command, it will return the name of the current directory. (Just the current directory's name, not the full path name)

1.443 bum_entrydir

Function: ENTRYDIR

Syntax: If ENTRYDIR Then...

This function returns TRUE if the current entry is a sub-directory, or FALSE if it is a file.

1.444 bum_entrybit$

Function: ENTRYBITS$

Syntax: n$=ENTRYBITS$

This function returns a string containing the protection-bits status of the current file or directory. An example may be "----RWED" the same format as given by the AmigaDOS "LIST" command. Possible bit settings are HSARWED: H=HIDDEN, S=SCRIPT, A=ARCHIVED, R=READABLE, W=WRITEABLE,

```
E=EXECUTEABLE, D=DELETEABLE.
Any bits that are not set will have the "-" character in their place.
```

1.445 bum_entrysize

```
Function: ENTRYSIZE
--------------------------------------------------------------------------------
Syntax: n.l=ENTRYSIZE

This function returns the size in bytes of the current directory entry.
Note that sub-directories return a size of zero whether they are empty
or not.
```

1.446 bum_entrydate

```
Function: ENTRYDATE
--------------------------------------------------------------------------------
Syntax: d$=DATE$(ENTRYDATE)

This function returns the date the current entry was last modified, in
the same format as SYSTEMDATE uses.  (The number of days since 1/1/1978)
Thus, you may use the DATE$ and DATEFORMAT commands to translate it into
a string with a more human-readable string.
```

1.447 bum_entryhour

```
Function: ENTRYHOUR, ENTRYMINS, ENTRYSECS
--------------------------------------------------------------------------------
Syntax: h=ENTRYHOUR:m=ENTRYMINS:s=ENTRYSECS

ENTRYHOUR:
This function is related to ENTRYDATE, above, but returns the hour of the
day (0-23) at which the entry was last modified.

ENTRYMINS:
Returns the minute (0-59) of the time at which the entry was modified.

ENTRYSECS:
Returns the second (0-59) of the time at which the entry was modified.
```

1.448 bum_entrycomment$

```
Function: ENTRYCOMMENT$
--------------------------------------------------------------------------------
Syntax: c$=ENTRYCOMMENT$

This function will return the string containing the filenote for the
current directory entry, or "" if there is none.
```

1.449 bum_elmoredosexample

```
*********************
* DIRECTORY EXAMPLE *
*********************

This example will list the entries in RAM: in a format very similar
to the AmigaDOS "LIST" command.  Note that you need to "ChDir" to
a directory in order to read it from the first entry again.


ChDir "RAM:"

While MoreEntries
  Print LSet$(EntryName$,30)
  If EntryDIR then Print "Dir   " Else Print LSet$(Str$(EntrySize),6)
  Print EntryBits$," ",Date$(EntryDate)," "
  Print EntryHour,":",Right$("0"+Str$(EntryMins),2),":"
  NPrint Right$("0"+Str$(EntrySecs),2)
Wend
MouseWait
```

1.450 bum_analyzedisk

```
Command: ANALYZEDISK
------------------------------------------------------------------------------
Syntax: ANALYZEDISK "DRIVE:"  -or-  If ANALYZEDISK "DRIVE:" Then...

This command returns FALSE if the specified device or pathname was not
valid.  If successful, details about the specified drive can be read with
the following "DISK" functions.  The values for these functions will not
change until ANALYZEDISK is executed again, either on the same drive or
another one.
Note:  If given a full pathname, such as "DF0:System/Utilities" this
command will still know enough to analyze the disk "DF0:"
```

1.451 bum_diskunit

```
Function: DISKUNIT
------------------------------------------------------------------------------
Syntax: n=DISKUNIT

This function will return the unit number of the most recently analyzed
disk.  DF0: for example, would return zero, while DF1: would return 1.
```

1.452 bum_diskerrs

```
Function: DISKERRS
------------------------------------------------------------------------------
```

```
Syntax: n=DISKERRS

This function will return the number of soft errors DOS knows about on
the last analyzed disk.  This should normally be zero.
```

1.453 bum_diskcapacity

```
Function: DISKCAPACITY
------------------------------------------------------------------------------
Syntax: n=DISKCAPACITY

This function returns the capacity in bytes of the last analyzed drive.
For example, a fastfilesystem-formatted disk's max capacity is 837K, so
DISKCAPACITY would return 857904, which divided by 1024 is 837.
```

1.454 bum_diskused

```
Function: DISKUSED
------------------------------------------------------------------------------
Syntax: n=DISKUSED

This function returns the number of bytes actually in-use on the last
analyzed drive.
```

1.455 bum_diskfree

```
Function: DISKFREE
------------------------------------------------------------------------------
Syntax: n=DISKFREE

The opposite of DISKUSED, DISKFREE returns the number of bytes free
on the disk.  This function would be very useful, for example, in a
program that needed to save information to disk.  You would be able
to first determine if the specified SAVE disk had sufficient space.
```

1.456 bum_diskblocks

```
Function: DISKBLOCKS
------------------------------------------------------------------------------
Syntax: n=DISKBLOCKS

This function returns the number of bytes each block on a disk uses,
making it possible to convert the byte-values of the above functions
to number of blocks.
```

1.457 bum7main

BUM7 MAIN DOC

Updates and Fixes to Blitz2 v1.9

NEW COMMANDS

NEW LIBRARY'S INCLUDED

Stability

Several improvements have been made to the stability of Blitz2 programs. First up all string commands have been fixed to both work properly with the null-termination system introduced in v.18 (our apologies here) and error checking has been added. No longer will system crashes be caused with illegal size parameters in mid$() etc.

Also, the ASMEND command has been added. Using assembler in statements and functions use to require the use of UNLK A4 and RTS. This system did not work properly when runtime errors were enabled. A fullproof method is now available, simply use the ASMEND command in place of any RTS commands. Blitz2 will look after the unlinking of A4, allow for runtime errors and then do an RTS. Finally my darts demo runs with runtime errors enabels (yipeeee!).

And finally, runtime error checking has been added for square bracket arrays. Yup, out of range checking has been incorporated for those of us whose first guess at why our programs were crashing was to go through and check such usage manually. This with the new string checking and the sexy new debugger should return a few people to using Blitz2's runtime debugging features. Thanks to all those and their abuse for helping us get these problems resolved.

Debugging

The debugger is now a separate program that is launched by Blitz2 when a prgram is run (runtime errors enabled of course).

The gadgets in the window allow the programmer access to the standard debugging features. CtrlAltC can still be used to halt programs, especially those using Slices and Displays in Blitz mode.

By increasing the size of the window the program listing can be viewed.

A PANIC! button has also been introduced once a program is launched from the editor. Yup, programs are now launched not run so those into weird system crashes may be able to return to Ted leaving their programs disabled in memory. A REBOOT button may have been more useful...

The source code for the default debugger is included in the acidlibsrc directory of the libsdev archive. It is extremely well documented by Mark so those wanting to extend the functionality of the system are most welcome. Serial port support for using a remote terminal would be very nice.

Interupts and BlitzKeys

BlitzKeys, BlitzKeys, BlitzKeys. A common profanity used by those of us use to keyboard lock ups in keyboard based Blitz games (especially SkidMarks). Well no more!

Blitz now leaves Amiga interupts enabled in Blitz mode. This means that not only is the system keyboard interupt still running (thank the lord) but any SetInts initiated in Amiga mode will continue.

Other advantages are that Blitz mode is now more acceptable to the CD32 environment and RawStatus can be used in Amiga mode for keyboard games not wanting to run in Windows (yuck).

Blitzkeys On now does a bit of a "BlitzkeysInput" for one character inputs only, any other inputs use the previously defined Input channel.

Blitzkeys Off no longer exists. BlitzRepeat has gone (no repeating keys).

Serial Stuff

Peter Tavinor has upgraded the Serial Library. ReadSerial now return a word (read unsigned byte) so chr$(255) is acceptable. WriteSerialString includes flags for DoIO and True String (not null terminated). ReadSerial has a new flag "WaitForChar"

GadTools

The GTPalette has had several default tags removed as they crashed under 2.0 (yeh, great, just what tags are suppose to avoid). AttachGTList had a minor problem in some situations (now fixed).

Another bug that has been found in GadTools under 2.0 is that GTLists actually allocate gadget id's for internal use. Besides being completely unethical (and fixed in 3.0) it means that programmers should use id values of greater than 50 to avoid this system bug. Adding GTLists last in your list should also work although their id's should be more than the number of lines of text they should display (no I am not going to explain further).

ScreensLib

The Screen command now rounds the width up to the nearest multiple of 16 rather than causing the error "Screen Width Must be a multiple of 16". Common sense I think.

ValLib

Val() now accepts hex and binary strings (preceeded by "$" and "%" of course.) Because Val() returns a float it should not be used to evaluate 32 bit integers (longs).

Display Library

A quick version of the InitCoplist command has been included which calculates the number of colours, sprites and size depending on just the

type parameter.

As promised the Display library now sports new commands for palette effects and so forth. There are two varieties of copper based commands, the first allows the user to insert a new palette or copperstring at a certain line of the display, the other allows control of each and every line of the display.

For line based effects a negative value should be used in combination with the numcustoms parameter of the InitCopList command. Color splits, bitmap scrolling, scan doubling/trebling/quadrupling and custom copper strings can now be acheived on a line by line basis.

Palette Library.

A number of commands have been added to the Palette library for use mainly with the display library. Fades and Colour cycling can now be performed on palette objects themselves (rather than on screens and slices) and hence can be used in conjunction with the DisplayPalette command.

Banks and Decoding.

Decode commands have been added to allow programmers to both include shapes, sounds, palettes, music and ILBM's (IFF bitmaps) in their programs or from preloaded files (mainly using the LoadBank command or unpacking type commands).

To include such files in the program the incbin command is used. Typically a list of included files will be situated at the bottom of the listing (with and End statement just above to be safe). Each IncBin will be preceeded by a label and the ?label syntax would be used to pass the location of each included file to the appropriate Decode command at the top of the program.

Those unhappy with the slow but memory unhungry LoadBitMap command can take advantage of the fast but memory hungry method of loading iff/ilbm files with the code listed in the DecodeILBM command description.

Argslib fixes

This library processes arguments passed to it. A few fixes have mainly been made over the old one.

1) Quoted arguments count as one argument. EG "One arg" will give your program both words as one argument, not 2.

2) Mulitple workbench arguments are allowed now.

If you are to use workbench arg handling, you MUST have WBSTARTUP at the top of your program!!

1.458 bum7_newlibs

Here are all the new library's included with BUM7:

AaronsIconLib

RICommoditiesLib

ElmoreLibs

RIEncryptLib

ElmoreIncLib

RIFxLib

FuzziesReqLib

RIGfxLib

NeilsCIATrackerLib

RIPackLib

NeilsProgressLib

RIReqLib

NeilsReqToolsLib

RIToolTypesLib

RIAmosFuncLib

RITrackDiskLib

RIAnimLib

RIZoneJoyLib

RIAppLib/WBlib

RomulusLibs

1.459 romulusmain

Look for these commands in:
CRUNCHLIB

PRINTERLIB

REQUESTERLIB

```
                LOCALE LIB

                CONSOLE LIB
```

1.460 riencrypt

```
Library Name: riencryptlib #55
Authors     : ReflectiveImages, 17 Mayles Road, Southsea, Portsmouth,
              Hampshire, UK PO4 8NP
OverView    : Another Reflective Images Library, good for war games?


                                Authors Docs:

                               RIEncryptLibrary

                             Date sent: 26-AUG-1994

------------------------------------------------------------------------
You can have this little library if you like.  Sorry there are no full
docs in the archive - the lib was done in a hurry for someone.  It
performs Enigma compression and is very cute ;-).
The archive contains the source code for the library.  I'm sure it
wouldn't take you a second to have a look at it ;-).

Commands in the library:

  Encrypt memadr,len[,wheel1,wheel2,wheel3]

This will encrypt a block of memory starting at the address and running
through to addresslength-1.  The optional wheel parameters allow you to
specify the start positions of the three wheels.  If you leave these out
then the wheels' start positions will be randomised.

  GetWheel n

This will tell you the position that wheel n stopped at after encrypting
a file.  n can range from 1 to 3 - YOU MUST REMEMBER THESE POSITIONS
IF YOU WANT TO DECRYPT THE FILE (at the moment at least).

  Decrypt memadr,len,wheel1,wheel2,wheel3

Same Encrypt except that it does the opposite and the wheel positions ARE
NOT OPTIONAL.  The positions should be the ones you wrote down after
encrypting the file.  I may, in future, change it so that you can also
specify the start wheel positions instead of the end ones (shouldn't be
tooooo hard ;-)).

Oh well, have a look and tell us what you think.............
```

1.461 reqtoolsmain

```
               Library:  neilsreqtoolslib #54
Author:   Neil O'Rourke, 6 Victoria St, TAMWORTH, NSW 2340, AUSTRALIA
Overview: Access to the ReqTools library.
```

RTEZRequest

RTEZGetString

RTEZFlagsRequest

RTLockWindow

RTEZFontRequest

RTUnlockWindow

RTEZScreenModeRequest

RTVersion

RTEZPaletteRequest

RTRevision

RTEZLoadFile

IsReqtoolsActive

RTEZSaveFile

RTASyncRequest

RTEZPathRequest

RTCheckASyncRequest

RTEZMultiLoadFile

RTEndASyncRequest

RTEZRNextPathEntry

RTASyncPaletteRequest

RTEZSetDefaultDirectory

RTCheckASyncPaletteRequest

RTEZSetPattern

RTEndASyncPaletteRequest

RTEZFreePattern

RTRequest

RTEZGetLong

RTFileRequest

RTEZGetLongRange

Author's Documentation

ReqToolslib V1.70b

Neil O'Rourke

** BETA FOUR **

This is an implementation of Nico Franco's ReqTools library.

There are two different implementations of each function, a simple one (denoted by EZ (pronounced E-Zee) in the command name), and a complex one. The simple implementation has bog standard requesters that the programmer has little (if any) control over. The purpose of these is to get your programs working fast (or it could be that you don't need all the fancy options that are available), with a minimum of setup for the requesters. That isn't to say the requesters aren't powerful; on the contrary, ReqTools requesters leave ASL requesters in the dust when it comes down to sheer power.

The more complex implementation requires you to build a TagList and supply it to the requester. This shouldn't really be needed, as all the EZ requesters have a resonable set of defaults and options to avoid this.

All the ReqTools requesters attach themselves to the window that DOS errors are. To do this, simply make your window the current window with Use Window WindowNum, then CatchDosErrs. By default, the requesters will go to the Workbench.

Compatibility
~~~~~~~~~~~~~
All ReqTools requesters, with the exception of the ScreenMode request, are compatible with KickStart 1.3.

DO NOT CALL RTEZSCREENMODEREQUEST IF YOU ARE RUNNING 1.3!! THIS IS YOUR RESPONSIBILITY!

Future Directions
~~~~~~~~~~~~~~~~~
The recent releases of ReqTools have included a preferences program to control the behaviour of requesters. This does not sit well with the pre-programmed options that my interface code uses. It could be that future releases of ReqToolsLib will not set these, but this is early days and only time and user feed-back will indicate the path to go.

1.462 progressmain

```
The doc file was so much damaged that I could not convert it....If
somebody has a good file, please send or mail it to me!
```

1.463 nreq1

```
Function: RTEZRequest
------------------------------------------------------------------------
Syntax  : result=RTEZRequest(Title$,BodyText$,GadgetText$ [,ReqPosition
                [, DefaultResponse, Flags]])

Description:
Opens a simple requester in the center of your screen.  You can have
multiple gadgets in Gadget$, seperate the by a bar (|).

To have multiple lines in your gadget, seperate them by a Chr$(10).

Title$ can be whatever you want.  If it is left blank, the title of the
calling window will be used.

The requester auto-adjusts its size to the length of the body text.

Also, the requester will block any input to the calling window, and if the
user selects that window he will see the usual wait pointer.

The requester returns the number of the gadget selected, gadget zero is
the extreme right hand gadget (usually Cancel), and numbered from one
starting from the left hand side of the requester.

The optional parameter ReqPos allows relative positioning of the
requester. You can have the requester open up in the center of the screen
(the default position), or the center of the window, or the TopLeft corner
of the screen or window.  The valid flags are:

#REQPOS_POINTER      =0  Relative to MousePointer
#REQPOS_CENTERWIN    =1  Center of window
#REQPOS_CENTERSCR    =2  Center of screen (default)
#REQPOS_TOPLEFTWIN   =3  TopLeft of the window
#REQPOS_TOPLEFTSCR   =4  TopLeft of the screen (Amiga default)

There are two further options:

DefaultResponse allows you to change what gadget is selected when the
Return key is hit, and this is by default the left hand gadget (1)

Flags controls a few other items in the requester.
#EZREQB_NORETURNKEY  =1   Turns off the return key as positive response
#EZREQB_LAMIGAQUAL   =2   Keyboard shortcuts are limited to LA-V and LA-B
#EZREQB_CENTERTEXT   =4   Centers the text in the requester.

You can make keyboard shortcuts for the gadgets by placing an underscore
character '_' before the character you wish to have as the shortcut, for
example your "Ok" gadget could be defined as "_Ok", and Right Amiga-O
would then satisfy the requester.
```

1.464 nreq2

```
Function: RTEZFlagsRequest
--------------------------------------------------------------------------
Syntax  : result=RTEZFlagsRequest(Title$,BodyText$,GadgetText$,
                IDCMPFlags[,ReqPos])


Description:
This requester is similar to the standard RTEZRequest, but it can also be
satisfied by an IDCMP flag (eg DiskInserted).  Either the gadget number or
the IDCMP flag will be returned in result.

This requester also supports no gadgets, by supplying "" as the Gadget
Text$. Since the window is locked, the user cannot proceed until the
request is satisfied.  Use this at your own peril!  This requester can
force the user to take an action he may not want to, if you don't supply
any gadgets.  Think, have second thoughts, and then think some more.
With enough thought, you *will* come to the conclusion that the user
needs at least one gadget.

The ReqPosition flag is also available for this requester, as is the
keyboard shortcuts.
```

1.465 nreq3

```
Function: RTEZFontRequest
--------------------------------------------------------------------------
Syntax  : *MyFont.TextAttr=RTEZFontRequest(Title$)

Description:
Brings up the Font requester, and returns a pointer to a TextAttr
structure.

The Font requester has had a total rewrite for the V1.7 release of
ReqToolsLib.  Using it is just the same, but it now returns a saner
structure.

The structure is defined:

NewType.TA
  Name.s
  YSize.w
  Style.b
  Flags.b
End Newtype
```

1.466 nreq4

```
Function: RTEZScreenModeRequest
--------------------------------------------------------------------------
Syntax  : *MyScreenMode =RTEZScreenModeRequest(Title$ [,DisplayFlags])
```

```
Description:
Returns a pointer to the following structure:

NEWTYPE.MyScreenMode
  DisplayID.l
  DisplayWidth.w
  DisplayHeight.w
  DisplayDepth.w
  OverscanType.w
  AutoScroll.l
End NEWTYPE

The DisplayFlags field allows you to have control over what options you
offer the user.  By default, the requester has a resonable set of options,
but you may wish to add too (or subtract from) these.

Allowable flags are:
#SCREQF_OVERSCANGAD  -  Add an overscan cycle gadget to the requester.
      After the requester returns you may read the
      overscan type in '\OverscanType'  If this is 0 no
      overscan is selected
      (Regular Size), if non-zero it holds one of the
      OSCAN_... values defined in the include file
      'intuition /screens.[h|i]'.
#SCREQF_AUTOSCROLLGAD-  Add an autoscroll checkbox gadget to the requester.
      After the requester returns read '\AutoScroll' to
      see if the user prefers autoscroll to be on or off.
#SCREQF_SIZEGADS     -  Add width and height gadgets to the requester.  If
      you do not add these gadgets the width and height
      returned will be the default width and height for
      the selected overscan type.
#SCREQF_DEPTHGAD     -  Add a depth slider gadget to the requester.  If you
      do not add a depth gadget, the depth returned will
      be the maximum depth this mode can be opened in.
#SCREQF_NONSTDMODES  -  Include all modes. Unless this flag is set
      RTEZScreenModeRequest() will exclude nonstandard
      modes. Nonstandard modes are presently HAM and EHB
      (ExtraHalfBrite). So unless you are picking a mode
      to do some rendering in leave this flag unset.
      Without this flag set the mode returned will be a
      normal bitplaned mode.
#SCREQF_GUIMODES     -  Set this flag if you are getting a screen mode to
      open a user interface screen in.  The modes shown
      will be standard modes with a high enough
      resolution (minumum 640 pixels).  If this flag is
      set the SCREQF_NONSTDMODES flag is ignored.

Do not attempt to call this requester under WB1.3.
```

1.467 nreq5

```
Function: RTEZPaletteRequest
------------------------------------------------------------------------
Syntax  : SelectedColour.w=RTEZPaletteRequest(Title$,FirstColour)
```

Description:
Brings up the Palette requester. Returns the last colour the user selected, or -1 if the user hit cancel. If the user changed the colours, they are reflected in the viewport that the window is attached to.

1.468 nreq6

Function: RTEZLoadFile

Syntax : name$=RTEZLoadFile(Title$,FileName$)

Description:
This brings up the standard file requester. The directories are buffered, so it doesn't have to reload the directory each time it is called. Note that FileName$ must be at least 108 characters long (use the MaxLen function of this).

Note that by default, pattern matching is not enabled. If you want to match a particular pattern, use the RTEZSetPattern command described below.

Also, the file name isn't copied to FileName$. You can have a default file name by writing to FileName$, but this will be cleared after the call finishes. This is also true of the SaveFile requester.

1.469 nreq7

Function: RTEZSaveFile

Syntax : name$=RTEZSaveFile(Title$,FileName$)

Description:
A seperate requester, the SaveFile requester is different from the LoadFile requester in a number of ways. First, it has a seperate buffer from the LoadFile requester. Second, the OK text is changed to Save. Third, the user cannot double-click a file to select it, to prevent accidental deletions. Finally, if the user types in a non-existent directory, he will be asked if he would like that directory created.

FileName$ must be at least 108 bytes long as well.

Note that by default, pattern matching is not enabled. If you want to match a particular pattern, use the RTEZSetPattern command described below.

1.470 nreq8

Function: RTEZPathRequest

Syntax : name$=RTEZPathRequest(Title$)

```
Description:
Prompts the user to select a path.  This is also a seperate requester to
the LoadFile and SaveFile requesters, and maintains its own directory list.
```

1.471 nreq9

```
Function: RTEZMultiLoadFile
------------------------------------------------------------------------
Syntax  : ret.l=RTEZMultiLoadFile(Title$)

Description:
Allows the user to select multiple files for loading (this makes no sense
for saving).  ret is either True for a list having been selected, or False
if the user cancelled.

If you call RTEZMultiLoadFile, any previous FileList that was loaded is
deleted, even if the user cancels the requester.
```

1.472 nreq10

```
Function: RTNextPathEntry
------------------------------------------------------------------------
Syntax  : name$=RTNextPathEntry

Description:
This function returns the next file from a RTEZMultiLoadFile call, or a
null string is there is no entry, so you can safely loop about until an
empty string is returned.
```

1.473 nreq11

```
Statement: RTEZSetDefaultDirectory
------------------------------------------------------------------------
Syntax   : RTEZSetDefaultDirectory Requester#,Directory$

Description:
This can be used to set a default directory for the user. Directory$ is
the default path, and Requester# is one of the following:
0 - EZLoadFile
1 - EZSaveFile
2 - EZPathRequest
3 - EZMultiLoadFile
```

1.474 nreq12

```
Statement: RTEZSetPattern
------------------------------------------------------------------------
Syntax   : RTEZSetPattern Requester#,Pattern$

Description:
Enables and sets the pattern matching in LoadFile and SaveFile requesters.
Valid requesters are:
0 - EZLoadFile
1 - EZSaveFile
3 - EZMultiLoadFile
```

1.475 nreq13

```
Statement: RTEZFreePattern
------------------------------------------------------------------------
Syntax   : RTEZFreePattern Requester#

Description:
Turns off pattern matching in the requester.  Valid requester numbers are:

0 - EZLoadFile
1 - EZSaveFile
3 - EZMultiLoadFile
```

1.476 nreq14

```
Function: RTEZGetLong
------------------------------------------------------------------------
Syntax  : result.l=RTEZGetLong(Title$,BodyText$ [,DefaultValue])

Description:
This prompts the user for a number.  BodyText$ can be formatted with
chr$(10) if needed.  DefaultValue can be supplied to suggest a value to
the user.
```

1.477 nreq15

```
Function: RTEZGetLongRange
------------------------------------------------------------------------
Syntax  : result.l=RTEZGetLongRange(Title$,BodyText$,Min.l,Max.l
                  [,DefaultValue])

Description:
Like RTEZGetLong, but this imposes an inclusive minimum and maximum on the
number entered.  Again, DefaultValue can be used to suggest a value to the
user.
```

1.478 nreq16

```
Function: RTEZGetString
------------------------------------------------------------------------
Syntax  : returned$=RTEZGetString(Title$,BodyText$,MaxChars
          [,DefaultString])

Description:
Prompts the user to enter a string (which can be up to MaxChars in length).
As usual, you can format the BodyText$ with chr$(10) and supply a default
string.  If you do supply one, then make sure that the length of the string
is less than MaxChars, otherwise you could corrupt innocent memory.
```

1.479 nreq17

```
Function: RTLockWindow
------------------------------------------------------------------------
Syntax  : WinLock=RTLockWindow(Window#)

Description:
This locks the numbered window, blocks all input to that window except
depth arranging (and the Zip gadget, under 2.0), and put up the standard
wait pointer.  If you have some utility to make the hands spin or something
like that, then that happens as well.  WinLock must be saved!
```

1.480 nreq18

```
Statement: RTUnlockWindow
------------------------------------------------------------------------
Syntax   : RTUnlockWindow Window#,WinLock

Description:
Unlocks the window that you locked with RTLockWindow.
```

1.481 nreq19

```
Functions: RTVersion and RTRevision
------------------------------------------------------------------------
Syntax   : ver=RTVersion and ver=RTRevision

Description:
Both these functions return the version number and revision number of the
ReqTools library that this code interfaces to.

Of no real use at the moment, but future developments in ReqTools may
require a minimum library version to work.  ReqToolsLib will always open
whatever ReqTools are available.
```

1.482 nreq20

```
Function: IsReqToolsActive
------------------------------------------------------------------------
Syntax  : result=IsReqToolsActive

Description:
Returns True if ReqTools was able to initialise, and False if it wasn't (eg
not available).
```

1.483 nreq21

```
Function: RTASyncRequest
------------------------------------------------------------------------
Syntax  : ret.l=RTASyncRequest(Title$,BodyText$,GadgetText$)

Description:
This function puts up a request, locks the window and returns immediately.
If the requester couldn't be put up, ret is False.  The program is now free
to continue, but the user can have the option of aborting a lengthy
operation if required.

Important Note: Do not attempt to have two asyncronous requesters up.

Note: As of V1.41b, RTASyncRequest uses the current window for the
requester.
```

1.484 nreq22

```
Function: RTCheckASyncRequest
------------------------------------------------------------------------
Syntax  : ret.l=RTCheckASyncRequest

Description:
Checks the status of the asyncronous requester, and returns True if it is
still up.
```

1.485 nreq23

```
Statement: RTEndASyncRequest
------------------------------------------------------------------------
Syntax   : RTEndASyncRequest

Description:
Ends the asyncronous request, under program control, and unlocks the
calling window.

NOTE:  Do not call this finction if the user has hit the gadget in the
request!  The requester automatically frees its self.
```

```
These three commands require a demonstration to illustrate:

NoCli:WBStartup
WbToScreen 0
Window 0,10,10,100,100,$8|$1000|2|4,"RTTest window",2,1
CatchDosErrs
ret.l=RTASyncRequest("Hi There!","Please Wait...","Cancel")

If ret  ;The requester opened OK
  For x.w=10 To 1 Step -1
    WLocate 0,10
    NPrint "Seconds:",x
    VWait(50)
    ret1.l=RTCheckASyncRequest ;Is the requester still up?
    If NOT ret1 ;No, so end this processing
      Pop If:Pop For:Pop If
      Goto cancelled
    EndIf
  Next x
  RTEndASyncRequest ;Normal finish
EndIf
End

cancelled:
a.l=RTEZRequest("Oi!","You cancelled!?!","Sure Did")
End
```

1.486 nreq24

```
Function: RTASyncPaletteRequest
--------------------------------------------------------------------------
Syntax  : ret.l=RTASyncPaletteRequest(Title$,FirstColour)

Description:
Similar to RTEZPaletteRequest, this command puts up a palette requester and
returns immediatly.  Note, however, that the calling window is NOT locked,
unlike all other ReqTools requesters.  This allows you to launch a seperate
palette requester and continue processing.
```

1.487 nreq25

```
Function: RTCheckASyncPaletteRequest
--------------------------------------------------------------------------
Syntax  : ret.l=RTCheckASyncPaletteRequest

Description:
Returns True if the requester is still up, False if the user hit Ok or
Cancel.  NOTE: There is no way to detect exactly how the user exited the
command.
```

1.488 nreq26

```
Statement: RTEndASyncPaletteRequest
------------------------------------------------------------------------------
Syntax   : RTEndASyncPaletteRequest

Description:
Closes the requester.

A short demonstration program to illustrate:

WbToScreen 0
Window 0,0,0,200,100,$40,"Hi there",2,1
CatchDosErrs

ret.l=RTASyncPaletteRequest("Play with these",1)

count.l=0
If ret
  While count<100
    count+1
    WLocate 0,0
    NPrint "Seconds:",count
    If NOT RTCheckASyncPaletteRequest Then Goto quit
    Delay_ 60
  Wend
  RTEndASyncPaletteRequest
EndIf

quit:
Free Window 0
```

1.489 nreq27

```
Function: RTRequest
------------------------------------------------------------------------------
Syntax  : ret=RTRequest(BodyText$,GadgetText$,TagList)

Description:
This is the standard form of the ReqTools Requester.  You must supply the
tag list to control the requester.  The requester title, if not specified
in the tag list, will be "Information" if you have only one response
gadget, or "Request" if you have two or more responses.

If you don't supply a tag list, ReqTools will use its own defaults.

It is *your* responsibility to ensure the TagList is correctly set up.

Most of the tags of interest are included in RTEZRequest and
RTEZFlagsRequest as standard.

Acceptable tags are:
   #RT_Window       - *name.Window
                      Window that will be used to find the screen to put
```

```
                    the requester on.
                    You *MUST* supply this if you are a task calling
                    this function and not a process! This is because
                    tasks don't have a pr_WindowPtr.
#RT_IDCMPFlags    - (LONG)
                    Extra idcmp flags to return on.  If one these IDCMP
                    flags causes the requester to abort the return code
                    will equal the flag in question.
#RT_ReqPos        - (LONG)
                    One of the following:
                    #REQPOS_POINTER    - requester appears where the
                                         mouse pointer is (default).
                    #REQPOS_CENTERSCR  - requester is centered on the
                                         screen.
                    #REQPOS_CENTERWIN  - requester is centered in the
                                         window (only works if the
                                         pr_WindowPtr of your process is
                                         valid or if you use RT_Window).
                                         If RT_Window is NULL the
                                         requester will be centered on
                                         the screen.
                    #REQPOS_TOPLEFTSCR - requester appears at the top
                                         left of the screen.
                    #REQPOS_TOPLEFTWIN - requester appears at the top
                                         left of the window (only works
                                         if the pr_WindowPtr of your
                                         process is valid or if you use
                                         RT_Window).
                    The requester will always remain in the visible part
                    of the screen, so if you use the Workbench 2.0
                    ScreenMode preferences editor to enlarge your
                    Workbench screen and you scroll around, the
                    requester will always appear in the part you can see.
                    REQPOS_CENTERSCR and REQPOS_TOPLEFTSCR also apply to
                    the visible part of the screen. So if you use one of
                    these the requester will be appear in the center or
                    the top left off what you can see of the screen as
                    opposed to the entire screen.
                    REQPOS_CENTERWIN and REQPOS_TOPLEFTWIN fall back to
                    REQPOS_CENTERSCR or REQPOS_TOPLEFTSCR respectively
                    when there is no parent window.  So you can safely
                    use these without worrying about the existence of a
                    window.
#RT_LeftOffset    - (LONG)
                    Offset of left edge of requester relative to
                    position specified with RT_ReqPos (does not offset
                    the requester when RT_ReqPos is REQPOS_POINTER).
#RT_TopOffset     - (LONG)
                    Offset of top edge of requester relative to position
                    specified with RT_ReqPos (does not offset the
                    requester when RT_ReqPos is REQPOS_POINTER).
#RT_PubScrName    - (*string)
                    Name of public screen requester should appear on.
                    When this tag is used the RT_Window tag will be
                    ignored.
                    If the public screen is not found the requester will
                    open on the default public screen.
```

```
                    Only works on Kickstart 2.0!  reqtools.library does
                    not check this, it is up to you *NOT* to use this
                    tag on Kickstart 1.3 or below!
                    Note that the 1.3 version of reqtools.library also
                    understands and supports this tag (on 2.0).
  #RT_Screen        - (*name.Screen)
                    Address of screen to put requester on.  You should
                    never use this, use RT_Window or RT_PubScrName.
  #RT_ReqHandler    - (struct rtHandlerInfo **)
                    Using this tag you can start an "asynchronous"
                    requester. ti_TagData of the tag must hold the
                    address of a pointer variable to a rtHandlerInfo
                    structure.
                    The requester will initialize this pointer and will
                    return immediately after its normal initialization.
                    The return code will not be what you would normally
                    expect.  If the return code is _not_ equal to
                    CALL_HANDLER an error occurred and you should take
                    appropriate steps. If the return code was
                    CALL_HANDLER everything went ok and the requester
                    will still be up!
                    See the explanation for rtReqHandlerA() below for
                    the following steps you have to take.
  #RT_WaitPointer   - (BOOL)
                    If this is TRUE the window calling the requester
                    will get a standard wait pointer set while the
                    requester is up. This will happen if you used the
                    RT_Window tag or if your process's pr_WindowPtr is
                    valid.  Note that after the requester has finished
                    your window will be ClearPointer()-ed.  If you used
                    a custom pointer in your window you will have to
                    re-set it, or not use the RT_WaitPointer tag and put
                    up a wait pointer yourself.
                    If your program requires ReqTools V38 it is advised
                    you use RT_LockWindow instead.  Defaults to FALSE.
  #RT_LockWindow    - (BOOL) [V38]
                    If this is TRUE the window calling the requester
                    will get locked.  It will no longer accept any user
                    input and it will get standard wait pointer set.
                    This will happen only if you used the RT_Window tag
                    or if your process's pr_WindowPtr is valid.
                    RT_LockWindow will restore a custom pointer if you
                    have used one (unlike RT_WaitPointer).  So you do
                    not have to worry about having to restore it
                    yourself.  It is advised you use this tag as much as
                    possible. Defaults to FALSE.
  #RT_ScreenToFront - (BOOL) [V38]
                    Boolean indicating whether to pop the screen the
                    requester will appear on to the front. Default is
                    TRUE.
  #RT_ShareIDCMP    - (BOOL) [V38]
                    Boolean indicating whether to share the IDCMP port
                    of the parent window. Use this tag together with the
                    RT_Window tag to indicate the window to share IDCMP
                    with. Sharing the IDCMP port produces less overhead,
                    so it is advised you use this tag. Defaults to
                    FALSE.
```

```
#RT_Locale       - (struct Locale *) [V38]
                   Locale to determine what language to use for the
                   requester text.  If this tag is not used or its data
                   is NULL, the system's current default locale will be
                   used. Default NULL.
#RT_IntuiMsgFunc - (struct Hook *) [V38]
                   The requester will call this hook for each IDCMP
                   message it gets that doesn't belong to its window.
                   Only applies if you used the RT_ShareIDCMP tag to
                   share the IDCMP port with the parent window.
                   Parameters are as follows:
                     A0 - (struct Hook *) your hook
                     A2 - (struct rtReqInfo *) your requester info
                     A1 - (struct IntuiMessage *) the message
                   After you have finished examining the message and
                   your hook returns, ReqTools will reply the message.
                   So do not reply the message yourself!
#RT_Underscore   - (char) [V38]
                   Indicates the symbol that precedes the character in
                   the gadget label to be underscored.  This is to
                   define a keyboard shortcut for this gadget.
                   Example: to define the key 'Q' as a keyboard
                   shortcut for "Quit" and 'N' for "Oh, No!" you would
                   use the tag RT_Underscore, '_' and pass as gadfmt
                   "_Quit|Oh, _No!".  Do not use the symbol '%' as it
                   is used for string formatting.  The
                   usual character to use is '_' like in the example.
                   IMPORTANT: the shortcuts defined using RT_Underscore
                   take precedence of the default shortcuts!  It is for
                   example not wise to use a 'N' for a positive
                   response! Pick your shortcuts carefully!
#RT_TextAttr     - (struct TextAttr *) [V38]
                   Use this font for the requester.  Default is to use
                   the screen font.  Note that the font must already be
                   opened by you.  ReqTools will call OpenFont() on
                   this TextAttr, _not_ OpenDiskFont()!  If the font
                   cannot be opened using OpenFont() the default screen
                   font will be used.
#RTEZ_ReqTitle   - (char *)
                   Title of requester window, default is "Request"
                   unless the requester has less than 2 responses, then
                   the default title is "Information".
#RTEZ_Flags      - (ULONG)
                   Flags for rtEZRequestA():
                   #EZREQF_NORETURNKEY - turn off the RETURN key as
                                         shortcut for positive response.
                   #EZREQF_LAMIGAQUAL  - keyboard shortcuts are limited
                                         to Left Amiga 'V' and 'B', ESC
                                         and RETURN.
                   #EZREQF_CENTERTEXT  - centers each line of body text
                                         in the requester window. Useful
                                         for about requesters.
#RTEZ_DefaultResponse - (ULONG)
                   Response value that will be returned when the user
                   presses the return key.  Will be ignored if the
                   EZREQF_NORETURNKEY flag is set.  The text for this
```

response will be printed in bold. Default is 1.

1.490 nreq28

```
Function: RTFileRequest
------------------------------------------------------------------------------
Syntax  : name$=RTFileRequest(Title$,FileName$,TagList)

Description:
This is the standard ReqTools requester, and is seperate from the
LoadFile, SaveFile and PathRequest requesters.  No setup is done, but the
file name etc is returned as per the above requesters.

Most of the tags that you would set normally are included as standard in
the RTREZxFile requesters.

It is *your* responsibility to ensure that the TagList is correctly set
up.

Acceptable tags are:

   #RT_Window          - see rtEZRequestA()
   #RT_ReqPos          - see rtEZRequestA()
   #RT_LeftOffset      - see rtEZRequestA()
   #RT_TopOffset       - see rtEZRequestA()
   #RT_PubScrName      - see rtEZRequestA()
   #RT_Screen          - see rtEZRequestA()
   #RT_ReqHandler      - see rtEZRequestA()
   #RT_WaitPointer     - see rtEZRequestA()
   #RT_LockWindow      - [V38] see rtEZRequestA()
   #RT_ScreenToFront   - [V38] see rtEZRequestA()
   #RT_ShareIDCMP      - [V38] see rtEZRequestA()
   #RT_Locale          - [V38] see rtEZRequestA()
   #RT_IntuiMsgFunc    - (struct Hook *) [V38]
                         The requester will call this hook for each IDCMP
                         message it gets that doesn't belong to its window.
                         Only applies if you used the RT_ShareIDCMP tag to
                         share the IDCMP port with the parent window.
                         Parameters are as follows:
                           A0 - (struct Hook *) your hook
                           A2 - (struct rtFileRequester *) your requester
                           A1 - (struct IntuiMessage *) the message
                         After you have finished examining the message and
                         your hook returns, ReqTools will reply the
                         message. So do not reply the message yourself!
   #RT_Underscore      - (char) [V38]
                         Indicates the symbol that precedes the character
                         in a gadget's label to be underscored.  This will
                         also define the keyboard shortcut for this gadget.
                         Currently only needed for RTFI_OkText.  Usually
                         set to '_'.
   #RT_DefaultFont     - (struct TextFont *)
                         This tag allows you to specify the font to be
                         used in the requester when the screen font is
                         proportional. Default is GfxBase->DefaultFont.
```

```
#RT_TextAttr        - (struct TextAttr *) [V38]
                      Use this font for the requester.  Must be a fixed
                      width font, _not_ a proportional one.  Default is
                      to use the screen font or the default font (if
                      the screen font is proportional).  Note that the
                      font must already be opened by you.  ReqTools
                      will call OpenFont() on this TextAttr, _not_
                      OpenDiskFont()!
                      If the font cannot be opened using OpenFont() or
                      if the font is proportional the default screen
                      font will be used (or the font set with
                      RT_DefaultFont).
#RTFI_Flags         - (ULONG)
                      Several flags:
                      #FREQF_NOBUFFER    - do _not_ use a buffer to
                                           remember directory contents
                                           for the next time the file
                                           requester is used.
                      #FREQF_MULTISELECT - allow multiple files to be
                                           selected. rtFileRequest()
                                           will return a pointer to an
                                           rtFileList structure which
                                           will contain all selected
                                           files.  Use rtFreeFileList()
                                           to free the memory used by
                                           this file list.
                      #FREQF_SELECTDIRS  - set this flag if you wish to
                                           enable the selecting of dirs
                                           as well as files.  You *must*
                                           also set FREQF_MULTISELECT.
                                           Directories will be returned
                                           together with files in
                                           rtFileList, but with StrLen
                                           equal to -1.  If you need the
                                           length of the directory's
                                           name use strlen().
                      #FREQF_SAVE        - Set this if you are using the
                                           requester to save or delete
                                           something.  Double-clicking
                                           will be disabled so it is
                                           harder to make a mistake and
                                           select the wrong file.  If
                                           the user enters a non-
                                           existent directory in the
                                           drawer string
                                           gadget, a requester will
                                           appear asking if the
                                           directory should be created.
                      #FREQF_NOFILES     - Set this if you want to use
                                           the requester to allow the
                                           user to select a directory
                                           rather than a file.  Ideal
                                           for getting a destination
                                           dir. May be used with FREQF_
                                           MULTISELECT
                                           and FREQF_SELECTDIRS.
                      #FREQF_PATGAD      - When this is set a pattern
```

```
                                        gadget will be added to the
                                        requester.
#RTFI_Height        - (ULONG)
                      Suggested height of file requester window.
#RTFI_OkText        - (char *)
                      Replacement text for "Ok" gadget, max 6 chars
                      long.
#RTFI_VolumeRequest - (ULONG) [V38]
                      The presence of this tag turns the file requester
                      into a volume/assign disk requester.  This
                      requester can be used to get a device name
                      ("DF0:", "DH1:",..) or an assign ("C:", "FONTS:",
                      ...) from the user.
                      The result of this requester can be found in the
                      filereq->Dir field.  The volume can also be
                      changed with rtChangeReqAttrA() and the RTFI_Dir
                      tag.
                      Note:
                      that the user may edit the disk/assign names, or
                      enter a new one.  Note also that the real device
                      name is returned, not the name of the volume in
                      the device.  For example "DH1:", not "Hard1:".
                      The tag data (ULONG) is used to set following
                      flags:
                       #VREQF_NOASSIGNS - Do not include the assigns in
                                          the list, only the real
                                          devices.
                       #VREQF_NODISKS   - Do not include devices, just
                                          show the assigns.
                       #VREQF_ALLDISKS  - Show _all_ devices.  Default
                                          behavior is to show only those
                                          devices which have valid disks
                                          inserted into them.  So if you
                                          have no disk in drive DF0: it
                                          will not show up.  Set this
                                          flag if you do want these
                                          devices included.
                     NOTE: Do *NOT* use { RTFI_VolumeRequest, TRUE }!
                           You are then setting the VREQF_NOASSIGNS
                           flag! Use { RTFI_VolumeRequest, 0 } for a
                           normal volume requester.
                     NOTE: If you use the RTFI_FilterFunc described
                           below the third parameter will be a pointer
                           to a rtVolumeEntry structure rather than a
                           pointer to a FileInfoBlock structure!
                           Tech note: the DOS device list has been
                           unlocked, so it is safe to e.g. Lock() this
                           device and call Info() on this lock.
                     NOTE: A file requester structure allocated with
                           rtAllocRequest() should not be used for both
                           a file and a volume requester. Allocate two
                           requester structures if you need both a file
                           and a volume requester in your program!
#RTFI_FilterFunc    - (struct Hook *) [V38]
                      Call this hook for each file in the directory
                      being read (or for each entry in the volume
                      requester).
```

```
                        Parameters are as follows:
                          A0 - (struct Hook *) your hook
                          A2 - (struct rtFileRequester *) your filereq
                          A1 - (struct FileInfoBlock *) fib of file OR
                               (struct rtVolumeEntry *) device or assign
                               in case of a volume requester.
                        If your hook returns TRUE the file will be
                        accepted. If it returns FALSE the file will be
                        skipped and will not appear in the requester.
                        IMPORTANT NOTE: If you change your hook's
                                        behavior you _MUST_ purge the
                                        requester's buffer (using rtFree
                                        ReqBuffer())!
                        IMPORTANT NOTE: When this callback hook is called
                                        from a volume requester the
                                        pr_WindowPtr of your process will
                                        be set to -1 so *no* DOS
                                        requesters will appear when an
                                        error occurs!
   #RTFI_AllowEmpty    - (BOOL) [V38]
                          If RTFI_AllowEmpty is TRUE an empty file string
                          will also be accepted and returned.  Defaults to
                          FALSE, meaning that if the user enters no
                          filename the requester will be canceled.  You
                          should use this tag as little as possible!
```

1.491 bum7_fuzziesreqlib

```
                Library Name: fuzziesreqlib #53

Author: Peter Tavinor, 22 Tuhangi St, Kamo, Whangarei, New Zealand

Commands:


                 ColourRequest

                 Con_Base

                 Dos_Base

                 FileFilter

                 FileReqSize

                 FileStructure

                 GetString$

                 Gfx_Base

                 Int_Base

                 MaxSelect$
```

```
                NextFile$

                ReqColours

                ReqFileRequest$

                ReqFontSize

                Req_Base

                Rex_Base

                TextRequest

                TextTimeout
               OverView:
  Not only has Peter Tavinor (King Fuzzy) kindly fixed up and added
  stuff to our own libraries (seriallib especially) he has also sent
  us this library which is one of three that take advantage of the
  ReqLibrary.
  Fuzzy has also cludged up Ted with hotkeys and stuff, if you
  want to get hold of this version I suggest writing to him
  at the above address. Anyway, sorry I couldn't fit more of
  his stuff in this issue.

Authors Docs:

 Req   Library  Ver 1.1 By King Fuzzy  No. 201

 $VER: Req Library extention Docs version 1.1 by King Fuzzy

The following commands are in the Reqlib library and they require the req.library
To get the Requests on a custom window use CatchDosErrs (see Reference Manual)
```

1.492 bum7_colourrequest

```
Function: ColourRequest
------------------------------------------------------------------------------
Syntax  : Colour=ColourRequest(Colour)

Description:
This function brings up a handy little palette and allows the user to
select a colour using Colour as the default.

Example:
  c=ColourRequest(1)
```

1.493 bum7_conbase

```
Function: Con_Base
------------------------------------------------------------------------------
Syntax  : cl.l=Con_Base
```

```
This Returns pointer to Console.device. Used for jsr calls mainly rawkey
to cookedkey
```

1.494 bum7_dosbase

```
Function: Dos_Base
------------------------------------------------------------------------
Syntax  : dl.l=Dos_Base

This Returns pointer to Dos.Library used in jsr calls
```

1.495 bum7_filefilter

```
Statement: FileFilter
------------------------------------------------------------------------
Syntax   : FileFilter Show$,Hide$

This sets the Hide and Show filters in the req file requester

FileFilter "Req.*","*.Bak"

This will show all files starting with 'Req.' but not ones ending with
'.Bak'
```

1.496 bum7_filereqsize

```
Statement: FileReqSize
------------------------------------------------------------------------
Syntax   : FileReqSize Lines High,File Length,Device Length

Description:
This sets the size of the Req File Requester. The defaults are 8,16,10

FileReqSize 20,25,12
```

1.497 bum7_filestructure

```
Function: FileStructure
------------------------------------------------------------------------
Syntax  : Fs.l=FileStructure

Description:
Returns a pointer to the req file requester structure
```

1.498 bum7_getstring$

```
Function: GetString$
--------------------------------------------------------------------------
Syntax  : String$=GetString$(Title$,Default$,Visable size,Max length)

Description:
This brings up a string requester allowing the user to enter a string
The maximum length and the visable length are set with Max length and
Visable size The default string must have a maximum length of at least
Max length Returns a null string if cancel is selected

MaxLen d$=40
s$=GetString$("Type something",d$,30,40)
```

1.499 bum7_gfxbase

```
Function: Gfx_Base
--------------------------------------------------------------------------
Syntax  : gl.l=Gfx_Base

Description:
This Returns pointer to Graphics.Library used in jsr calls
```

1.500 bum7_intbase

```
Function: Int_Base
--------------------------------------------------------------------------
Syntax  : il.l=Int_Base

Description:
This Returns pointer to Intuition.Library used in jsr calls
```

1.501 bum7_maxselect$

```
Statement: MaxSelect$
--------------------------------------------------------------------------
Syntax   : Path$=MaxSelect$(Title$,File$,Path$,Flags,number of files)

Select the number of files that can be selected with the extended selector
see ReqFile Requestor for more information
```

1.502 bum7_nextfile$

```
Function: NextFile$
------------------------------------------------------------------------------
Syntax  : f$=NextFile$

Description:
Returns the next file in the extended file structure
```

1.503 bum7_reqcolours$

```
Statement: ReqColours
------------------------------------------------------------------------------
Syntax   : ReqColours Text,Detail,Block[,File,Dir,Device[,GadText,GadBox,
                      StringName,StringBox,Frame]]

Description:
Text,Detail and Block are for both the req file requester and the text
requester File,Dir,Device,GadText,GadBox,StringName,StringBox and Frame
are only for the req file requester

Try changing the colours one at a time to see what they change

ReqColours 1,2,3
ReqColours 1,2,3,3,2,1
ReqColours 1,2,3,3,2,1,1,2,3,4,5
```

1.504 bum7_reqfilerequest$

```
                Function: ReqFileRequest$
------------------------------------------------------------------------------
Syntax:  selectedfile$=ReqFileRequest$(Title$,File$,Path$,Flags)

Yes it's another file/font requester, No it won't bring up the blitz
requester if it fails the maximum length of File$ must be 32 and Path$
must be 132 The Flags are as follows (Clipped from reqbase.i)

FRQSHOWINFOB      EQU 0  =   1      ;Set this in Flags if you want .info files to  ↩
   show.  They default to hidden.
FRQEXTSELECTB     EQU 1  =   2      ;Set this in Flags if you want extended select ↩
   .  Default is not.
FRQCACHINGB       EQU 2  =   4      ;Set this in Flags if you want directory  ↩
   caching.  Default is not.
FRQGETFONTSB      EQU 3  =   8      ;Set this in Flags if you want a font  ↩
   requester rather than a file requester.
FRQINFOGADGETB    EQU 4  =  16      ;Set this in Flags if you want a hide-info  ↩
   files gadget.
FRQHIDEWILDSB     EQU 5  =  32      ;Set this in Flags if you DON'T want 'show'  ↩
   and 'hide' string gadgets.
FRQABSOLUTEXYB    EQU 6  =  64      ;Use absolute x,y positions rather than  ↩
   centering on mouse.
FRQCACHEPURGEB    EQU 7  = 128      ;Purge the cache whenever the directory date  ↩
   stamp changes if this is set.
```

```
FRQNOHALFCACHEB   EQU 8  = 256      ;Don't cache a directory unless it is ↩
   completely read in when this is set.
FRQNOSORTB        EQU 9  = 512      ;Set this in Flags if you DON'T want sorted ↩
   directories.
FRQNODRAGB        EQU 10 =1024      ;Set this in Flags if you DON'T want a drag ↩
   bar and depth gadgets.
FRQSAVINGB        EQU 11 =2048      ;Set this bit if you are selecting a file to ↩
   save to.
FRQLOADINGB       EQU 12 =4096      ;Set this bit if you are selecting a file(s) ↩
   to load from.
                                    ;These two bits (save and load) aren't ↩
                                       currently used for
                                    ;anything, but they may be in the future, so ↩
                                       you should
                                    ;remember to set them.  Also, these bits make ↩
                                       it easier if
                                    ;somebody wants to customize the file ↩
                                       requester for their
                                    ;machine.  They can make it behave differently ↩
                                        for loading
                                    ;vs saving.
FRQDIRONLYB       EQU 13 =8192      ;Allow the user to select a directory, rather ↩
   than a file.
```

Just add together what you want and use it. EG. 6 is Caching and extended select

Also see:

FileReqSize

FileFilter

ReqColours$

FileStructure

MaxSelect$

NextFile$

```
MaxLen fl$=32 : MaxLen dr$=132
f$=ReqFileRequest$("Select a file",fl$,dr$,4)
```

FRQEXTSELECTB is not used

Returns a null string if user aborts

1.505 bum7_reqfontsize

```
Function: ReqFontSize
------------------------------------------------------------------------
Syntax  : sz=ReqFontSize
```

This Returns the size of the last font selected with the font requester

1.506 bum7_reqbase

```
Function: Req_Base
--------------------------------------------------------------------------------
Syntax  : rl.l=Req_Base

This Returns pointer to Req.Library used in jsr calls
```

1.507 bum7_rexbase

```
Function: Rex_Base
--------------------------------------------------------------------------------
Syntax  : rxl.l=Rex_Base

This Returns pointer to RexxSys.Library I think, It says RexxSysBase in
the req library docs
```

1.508 bum7_textrequest

```
Function: TextRequest
--------------------------------------------------------------------------------
Syntax  : Button=TextRequest(Text,Title,Left Text[,[Middle Text,]Right Text])

This brings up a text requester with Text as the message and Title in the
titlebar It can have 1, 2 or 3 buttons to select from The requester's
colours can be changed: See  ReqColours

Left button
Left and Right buttons
Left, Middle and Right buttons

The text in the buttons is determined by Left Text,Middle Text and Right
Text

the value returned is
1  left button
2  middle button
0  right button

Button=TextRequest("I am a simple requester","Blitz Prog","Left","Middle","Right")
```

1.509 bum7_texttimeout

```
Statement: TextTimeout
--------------------------------------------------------------------------------
Syntax   : TextTimeout Timeout Seconds

Its surpose to set the timeout for the text requester but it don't work
yet
```

1.510 bum7_elmoreinclib

```
                Library Name: elmoreinclib #111
-------------------------------------------------------------------------
Author: Richard T Elmore, HeadSoft, 126 STATE ST. #20,
        SPEARFISH, SD 57783, USA

OverView:
This is a crippled version of a library which Richard has put a whole
heap of time into. It basically allows you to include compressed object
data into your programs which you can unpack at your leisure. Please see
the registration material at the bottom of this file if you like what
you see. Demo, util and lh.library are in userlibprogs/elmore.

Commands:


                 IncSound

                 IncNextShape

                 FreeIncData

                 IncBitmap

                 IncText$

                 SaveIncData

                 IncMod
                      IncData

                 IncMed

                 IncSize

                 IncShape

                 IncDataAbs

                 Using the Include-Util program
                                                Author's Docs:

                           ABOUT INCLUDE.ELMORELIB
                          =========================


The Include library by Richard T. Elmore of HeadSoft Software enables
Blitz2 programmers to make stand-alone files that don't require special
directories, external IFF files, etc. to run.  To achieve this in the most
efficient manner possible, the incredible efficiency and speed of the
LH.Library is used (in the INCLUDE-UTIL tool, which is freely
distributeable.)

At present, the Include library supports Bitmaps, (up to 8 bitplanes!)
sound effects, MED music modules, IFF brushes for Blitz2 SHAPES, Blitz2
```

SHAPES-FILES for multiple shapes, (as created by the SAVESHAPES command or with the SHAPESMAKER utility) entire ASCII text files, or raw binary data which gives the advanced programmer the ability to include other object types or other data such as executeable programs, variable arrays for cosine tables, etc.

The library was designed with Blitz2's INCBIN compiler directive, but may also be used with data loaded with READMEM or similar commands, to conserve disk space when you don't mind having external files, not to mention they will be made next to impossible to "rip" by anyone without the Include Library!

NOTE: Your executeables do NOT require the "lh.library" to run... They will have their own self-contained decrunching routine (which is much faster than the crunching routine in lh.library!)

1.511 elmore_includeutil

USING THE INCLUDE-UTIL PROGRAM
==========

In order to include the above-mentioned data in your Blitz2 executeables, the original data files must be converted and packed so that the resulting runtime program consumes the least memory possible. This also allows the data to be stored in public or "fast" memory, not just CHIP ram. The INCLUDE-UTIL program is supplied with the library to convert the data for you.

First, ensure the "LH.Library" file resides in your LIBS: directory. The INCLUDE-UTIL program will crash if it isn't availble.

In order to run the program, just click on it's icon, and a custom screen will appear containing icons representing the types of data you wish to convert to includeable data. Note that the ST/NT Module button is ghosted, as this data type is not supported in the public domain release of the library.

You may note there is a gadget to "quit" even though there is also a "close" gadget in the top-left of the main window. Clicking the close-gadget will NOT quit the program, it will merely close the window and screen, then an icon will appear on the Workbench screen. Simply click the icon to reactivate the program. (While in this idle mode, INCLUDE-UTIL uses no processor time and consumes less memory.)

Upon clicking one of the other icons, a file requester will appear prompting you to select a source file (the IFF, text, MOD, whatever) to convert. Note that you may load either IFF brushes *OR* BB2 shapes-files in the SHAPES filerequester, the appropriate INCLUDE-OBJECT type will be created.

After the file has been converted to it's INCLUDE-OBJECT form, you will be given a filerequester to SAVE the object with. Note that an INCLUDE-OBJECT extension will be appended to the filename to help you more easily

recognise the object types in a directory listing. They are:

```
  .ISFX     - Sound effects
  .IBMP     - Bitmaps
  .ISHP     - Single shapes
  .ISHPS    - Multiple shapes
  .IMED     - MED modules
  .IBIN     - Either binary or text (IncText$, IncDataABS, IncData, etc.)
```

The INCLUDE-UTIL program accepts tooltypes for default paths. Then whenever you click on an appropriate gadget the file requester will use the path you prefer. The following keywords identify the paths:

```
SOUNDS=pathname
BITMAPS=pathname
SHAPES=pathname
MODULES=pathname
BINARIES=pathname
SAVE=pathname     (This is the same path for saving all object types)
```

You should keep the name of this utility "INCLUDE-UTIL" or the tooltype preferences will not be available.

A few features are available under OS2.0 and above only: Notably, when you iconify INCLUDE-UTIL, it uses a real appicon, so you can drag objects onto it to have them automatically identified and loaded. You can also simply double-click the icon without dragging anything if you just want to wake the program up.

Now that you have your INCLUDE-OBJECTs, how do you go about making them part of your Blitz2 executeables? It is relatively simple, but you must be careful to follow these guidelines unless you know EXACTLY what you're doing or you'll crash your Amiga!

1. Place a copy of Include.ElmoreLib in your Blitzlibs:Userlibs directory, then optionally create a new DEFLIBS file.

2. Write and debug your program using normal loading routines until you're satisfied with it. No need to use INCLUDE-OBJECTs during debugging, as it will only slow down development. (Before being made executeable, Blitz2 will both load the objects from disk AND decrunch them.)

3. Go to the end of your sourcecode (usually the safest place) and select a different program label for EACH INCLUDE-OBJECT to be decrunched at run-time. Directly following the label, enter INCBIN "filename" which should reference the name you saved the INCLUDE-OBJECT as from INCLUDE-UTIL. See page 8-3 of your Blitz2 reference manual for details on INCBIN and INCDIR commands if you're not familiar with them.

4. Ensure you have an "END" statement or some such before your fist INCLUDE-OBJECT's label. If program flow continues into the data, you will almost surely have a crash.

```
5.  Replace your Blitz2 DOS-based loading commands with the approriate
    Include library versions.  It's wise to check the results of those
    commands that return "success" or "failure" (TRUE or FALSE) so that
    your program can exit gracefully if there isn't enough memory, etc.
    when its run.

6.  That's all!  You should be able to run your program normally, and
    executeables you create will run fine with graphics, sounds, whatever
    you want, and *NO* external files needed!!  Of course since all the
    data is included in the size of the executeable, it will be much
    larger than usual. (Size of INCLUDE-OBJECTs+normal executeable size)
    You may have some success crunching the entire executeable with
    PowerPacker or similar programs, but if the percentage of
    INCLUDE-OBJECT data in the executeable is very high, most crunchers
    will choke on it, since most of the program is already crunched
    by INCLUDE-UTIL.
```

1.512 bum7_incsound

```
Function/Statement: Incsound
------------------------------------------------------------------------
Syntax: success=IncSound(Sound#, ?Label)
        IncSound Sound#, ?Label

Description:
Ensure you put the question mark before the label name or you'll have
errors!  The actual include-object should be INCBINed directly after the
label, and be careful to put an END statement somewhere above your INCBIN
data or you'll crash!

Example:

If IncSound(0,?Mysound)=False Then End  ;Unpack the sound or end!

Sound 0,15  ;Play it back!
Mousewait   ;pause for the user....
End

Mysound:
IncBin "RAM:SoundEffect.isfx"

NOTE:  In the above example the FUNCTION version was used so you could
test it with an IF/THEN statement to see if it was successful...  If
you don't think you'll need to be so careful, the STATEMENT version would
be:

Example:

IncSound 0,?Mysound  ;Unpack the sound (Note no parentheses for
                                        statement!)
Sound 0,15  ;Play it back!
Mousewait   ;pause for the user....
End
```

1.513 bum7_incbitmap

```
Function/Statement: IncBitmap
------------------------------------------------------------------------------
Syntax: success=IncBitmap(bitmap#,?Label)
        IncBitmap bitmap#,?Label

Description:
Nearly identical in useage to IncSound (above) Note that if the bitmap
already exists, it should be the same dimensions such as 640x256x4 or
IncBitmap will return FALSE (for failure) if you don't know for sure, you
can let IncBitmap create the bitmap exactly like it was Included by doing
FREE BITMAP 0 or some such before you INCBITMAP it.

Example:

Blitz
If IncBitmap(0,?Mypicture)=0 Then End
Slice 0,32,4
Show 0
Mousewait:end

Mypicture:
IncBin "RAM:Picture.ibmp"
```

1.514 bum7_incmod

```
Function/Statement: IncMod
------------------------------------------------------------------------------
Syntax: success=IncMod(ST-NT Module#,?Label)
        IncMod Module#,?Label

Description:
Like the above examples, only for music mods. You could then use
StartModule etc. just as if you had loaded it from disk.

---------- NOTE: This function is a bit buggy, so it has
            been temporarily disabled with this release of
            the library!  (Sorry!) --------------------------------
```

1.515 bum7_incmed

```
Function/Statement: IncMed
------------------------------------------------------------------------------
Syntax: success=IncMed(MEDModule#,?Label)
        IncMed MedModule#,?Label

Description:
For including MED modules.  Usage is otherwise the same as IncMod.
```

1.516 bum7_incshape

```
Function/Statement: IncShape
------------------------------------------------------------------------------
Syntax: success=IncShape(Shape#,?Label[,Cookiecut?])
                IncShape Shape#,?Label[,Cookiecut?]
                                          ^^^^
                                    OPTIONAL DUMMY VALUE

Description:
The one you've been waiting for!  Will retrieve the shape# for BLITTING...
Along with the command below, IncNextShape, you can even include several
shapes in one step!  I will be adding multiple shapes as an option in the
INCLUDE-UTIL program as well... For now, just INCBIN as many shapes as you
want (only need label for the first one)

Note: If you don't need a cookiecut for the shape, you can leave out the
CookieCut parameter.  Any number placed in the cookiecut parameter will
cause a cookiecut to be made for the shape when it is made however.  If
you plan to do BLITs etc. you should always use the cookiecut.
```

1.517 bum7_incnextshape

```
Function: IncNextShape
------------------------------------------------------------------------------
Syntax  : success=IncNextShape [CookieCut?]
                               ^^^^^^^^^^^^
                          OPTIONAL DUMMY VALUE

Example:

;Include shape #0
 IncShape(0,?Shapes,1)
;Will loop 3 times from shape #1 to #3 in this case
While IncNextShape(1):Wend
;(Do your blitting stuff etc. here)
End

Shapes:
IncBin "Shape0.ISHP"
IncBin "Shape1.ISHP"
IncBin "Shape2.ISHP"
IncBin "Shape3.ISHP"

;(Note this is a FUNCTION only, no statement version)
```

1.518 bum7_inctext$

```
Function: IncText$
------------------------------------------------------------------------------
Syntax  : string$=IncText$(?Label[,optional length])
```

```
Description:
Used with BINARY include types....  You can then put whole text files
into strings.  The optional LENGTH will limit then string length to
whatever number you want, similar to the LEFT$ function.
It will only return a null-string "" in case of failure.

Example:

a$=IncText$(?text)
Nprint a$
b$=IncText$(?text2,32)
Nprint b$
Mousewait:End

text:
IncBin "Message.IBIN"

text2:
IncBin "Greetz.IBIN"
```

1.519 bum7_saveincdata

```
Function/Statement: SaveIncData
------------------------------------------------------------------------
Syntax: success=SaveIncData(Filename$,?Label)
        SaveIncData Filename$,?Label

Description:
This will write to disk the unpacked version of whatever BINARY
include-object you specify.  One good use of this is to save
programs to RAM and then EXECUTE them, and DELETE them again.

Of course there are hundreds of uses....

  If SaveIncData("Ram:newfile",?executeable)=0 Then End
  Execute_ "Ram:Newfile"
  KillFile "Ram:Newfile"
  End

  executeable:
  IncBin "myprog.ibin"
```

1.520 bum7_incdata

```
Function: IncData
------------------------------------------------------------------------
Syntax  : Address=IncData(?Label,memory type)

Description:
This allows you to include BINARY data for any number of uses that
aren't provided with the other functions.  Memory types are:
```

```
0-  Any kind of memory (preferably FAST RAM)
2-  CHIP RAM ONLY!

This function will return the address of the binary data in
memory, or 0 for failure....


Example of including a pure ASM routine object file for execution:

asmloc.l=IncData(?mlroutine,0)
Call asmloc
End

mlroutine:
IncBin "Ram:MLRoutine.IBIN"
```

1.521 bum7_incsize

```
Function: IncSize
------------------------------------------------------------------------
Syntax  : size.l=IncSize(?Label)

Description:
Returns the size in bytes of the BINARY object at the specified label
Among other uses, you need it if you want to FREE the uncrunched
binary data.  (It will automatically be freed when your program ends)
```

1.522 bum7_freeincdata

```
Statement: FreeIncData
------------------------------------------------------------------------
Syntax   : FreeIncData Size,Address

Description:
If you wanted to free up the memory allocated by the above IncData
function, here is how you'd do it:

  FreeIncData IncSize(?mlroutine),asmloc
```

1.523 bum7_incdataabs

```
Function: IncDataAbs
------------------------------------------------------------------------
Syntax  : bytesize=IncDataAbs(?Label,Destination Address)

Description:

*********************************************************
**                                                     **
**  A D V A N C E D :  Use with caution!               **
```

```
**                                                          **
************************************************************

This command will unpack the binary data directly to the area of memory
you specify, so if you're not sure what you'll probably
GURU the Amiga!  However, it is very useful to fill arrays, uncrunch data
directly to Banks, or whatever.  Just be careful!

It will return zero for failure, or the number of bytes in the binary
object.  I'll provide a better example of making use of this function
later....  (Filling up array variables, etc.)

InitBank 0,1000,0
size.l=IncDataAbs(?Binary,BankLoc(0))
Mousewait:End

Binary:
IncBin "binarydata.IBIN"
```

1.524 bum7_aaronsiconlib

```
                Library Name: aaronsiconlib #62

Author      : Aaron Koolen, Vision Software,
              15 Day Street,
              Newton, Auckland, NZ

OverView:
  Not only has Aaron kindly fixed up passing of argumens in our
  cliargslib but has also donated this library which similar to
  the Reflective Images version allows access to information
  from the programs workbench icon.

Commands:


                 GetIconInfo

                 IconTool$

                 IconSubTool$

                 IconType

                 IconStack

                 IconDefTool$
                Authors Documentation:

AaronsIconLib
=============

This library is for processing the icon (.info) files.  It only provides
routines for reading the data from icons, not for writing or creating new
icons, which may be added later.  It is most useful when used in
```

conjunction with the ArgsLib. You can set the maximum number of allowed
icon info's in the options. Also to free an IconInfo object, after a
GetIconInfo use Free IconInfo #

1.525 aaron_geticoninfo

```
Function: GetIconInfo
------------------------------------------------------------------------
Syntax: boolean.w=GetIconInfo(icon#,iconname$)

Description:
This examines a .info file so you can get information about it.
'iconname$' is the name of the icon without the .info suffix and icon# is
the number of the IconInfo object you want to put the data under.  It will
return FALSE (0) if it failed, or TRUE (-1) if it succeeded.
```

1.526 aaron_icontool$

```
Function: IconTool$
------------------------------------------------------------------------
Syntax: tool$=IconTool$(icon#,toolname$)

Description:
Returns the respective data of the tooltype specified by 'toolname$' of
IconInfo object icon#.

EG

If IconTool$(0,"CX_POPUP")="YES" Then Gosub PopUpWindow
```

1.527 aaron_iconsubtool$

```
Function: IconSubTool$
------------------------------------------------------------------------
Syntax: boolean.w=IconSubTool$(toolname$,subtool$)

Description:
Returns TRUE (-1) or FALSE (0) if the sub tool type 'subtool$' exists
within the tool 'toolname$'

EG

  If IconSubTool$(0,IconTool$(0,"FILETYPE"),"ILBM") Then file type of
  file was ILBM.

A Subtool (My word) is one that resides in a tool type but is separated by
bars (|).  EG

FILETYPE=PaintProgram|ILBM    (PaintProgram and ILBM are "Sub Tools")
```

```
IconTool$ will return the PaintProgram|ILBM part and you can then use
IconSubtool$ to see if things like ILBM or PaintProgram exist in that
string.

NOTE: This does not require the passing if an IconInfo object, it simply
requires 2 strings, so you can use it for other things too.
```

1.528 aaron_icontype

```
Function: IconType
--------------------------------------------------------------------------
Syntax: type.w=IconType(icon#)

Description:
Returns the type of IconInfo object icon#.

EG  NPrint IconType(0)

'type' is one of the list from workbench/workbench.bb2.
```

1.529 aaron_iconstack

```
Function: IconStack
--------------------------------------------------------------------------
Syntax: stackSize.l=IconStack(icon#)

Description:
Returns the stack sie setting of the icon.
```

1.530 aaron_icondeftool$

```
Function: IconDefTool$
--------------------------------------------------------------------------
Syntax: deftool$.w=IconDefTool$(icon#)

Description:
Returns the default tool of the icon.

EG  NPrint IconDefTool$(icon#)

May print something like "blitz2:blitz2" if icon# references a Blitz2
source program.
```

1.531 bum7_newcommands

AllocMem

FreeMem

Bank

FromCLI

BlockScroll

GameB

ClipBlitMode

GTArrowSize

CustomColors

GTStatus

CustomString

InitPalette

CyclePalette

InitShape

DecodeILBM

LoadBank

DecodeMedModule

NumPars

DecodePalette

PaletteRange

DecodeShapes

Par$

DecodeSound

ParPath$

DisplayDblScan

PopInput

DisplayRainbow

PopOutput

```
DisplayRGB

ReadSerialMem

DisplayScroll

SavePalette

DisplayUser

SetPeriod

DuplicatePalette

WriteSerialMem

FadePalette
```

1.532 bum7_bank

```
Function: Bank
--------------------------------------------------------------------------
Syntax  : Bank(Bank#)

Returns the memory location of the given memory Bank, replaces the older
and more stupidly named BankLoc command.
```

1.533 bum7_blockscroll

```
Statement: BlockScroll
--------------------------------------------------------------------------
Syntax   : BlockScroll X1,Y1,Width,Height,X2,Y2[,BitMap#]

library: scrolllib

Description:
Same as the Scroll command except that BlockScroll is much faster but
only works with 16 bit aligned areas. This means that X1, X2 and Width
must all be multiples of 16. Useful for block scrolling routines that
render the same blocks to both sides of the display, the programmer
can now choose to render just one set and then copy the result to the
other side with the BlockScroll command.
```

1.534 bum7_clipblitmode

```
Statement: ClipBlitMode
--------------------------------------------------------------------------
Syntax   : ClipBlitMode BPLCON0
Library  : 2dlib
```

```
Description:
Same as BlitMode except applies to the ClipBlit command. Another oversight
now fixed.
```

1.535 bum7_customcolors

```
Statement: CustomColors
------------------------------------------------------------------------
Syntax   : CustomColors CopList#,CCOffset,YPos,Palette,startcol,numcols
Library  : displaylib

Using the custom copper space in a display, CustomColors will alter the
displays palette at the given YPos. The number of customcops required is
either 2+numcols for ecs displays and 2+n+n+n/16 for aga displays. In aga,
numcols must be a multiple of 32.

Note that large AGA palette changes may take several lines of the display
to be complete.
```

1.536 bum7_customstring

```
Statement: CustomString
------------------------------------------------------------------------
Syntax   : CustomString CopList#,CCOffset,YPos,Copper$
Library  : displaylib

Description:
CustomString allows the user to insert their own copper commands
(contained in a string) into the display's copper list at a given vertical
position. The amount of space required is equal to the number of copper
instructions in the Copper$ (length of string divide by 4) plus 2 which of
course have to be allocated with InitCopList before CustomString is used.
```

1.537 bum7_cyclepalette

```
Statement: CyclePalette
------------------------------------------------------------------------
Syntax   : CyclePalette Palette#
Library  : palettelib

Description:
CyclePalette uses the standard color cycling parameters in the palette
object to cycle the colors. Unlike the Cycle command which copied the
resulting palette to the current screen the CyclePalette command just
modifies the palette object and can hence be used with the DisplayBitmap
command in the new Display library.
```

1.538 bum7_decodeilbm

```
Statement: DecodeILBM
------------------------------------------------------------------------
Syntax   : DecodeILBM BitMap#,MemoryLocation
Library  : ilbmifflib

Description:
A very fast method of unpacking standard iffilbm data to a bitmap. Not
only does this command allow a faster method of loading standard IFF
files but allows the programmer to "incbin" iff pictures in their programs.
See the discussion above for using DecodeILBM on both files and included
memory.
```

1.539 bum7_decodemedmodule

```
Statement: DecodeMedModule
------------------------------------------------------------------------
Syntax   : DecodeMedModule MedModule#,MemoryLocation
Library  : medlib

Description:
DecodeMedModule replaces the cludgemedmodule, as med modules are not
packed but used raw, DecodeMedModule simply checks to see the
memorylocation passed is in ChipMem (if not it copies the data to chip)
and points the Blitz2 MedModule object to that memory.
```

1.540 bum7_decodepalette

```
Statement: DecodePalette
------------------------------------------------------------------------
Syntax   : DecodePalette Palette#,MemoryLocation[,Palette Offset]
Library  : palettelib

Description:
DecodePalette allows the programmer to unpack included iff palette
information to Blitz2 palette objects.
```

1.541 bum7_decodeshapes

```
Statament: DecodeShapes
------------------------------------------------------------------------
Syntax   : DecodeShapes Shape#[,Shape#],MemoryLocation
Library  : shapeslib

Description:
DecodeShapes, similar to DecodeMedModule ensures the data is in chip and
then configures the Shape object(s) to point to the data.
```

1.542 bum7_decodesound

```
Statement: DecodeSound
------------------------------------------------------------------------
Syntax   : DecodeSound Sound#,MemoryLocation
Library  : audiolib

Description:
DecodeSound similar to the other new Decode commands allows the programmer
to include sound files within their program's object code.
```

1.543 bum7_displaydblscan

```
Statement: DisplayDblScan
------------------------------------------------------------------------
Syntax   : DisplayDblScan CopList#,Mode[,copoffset]
Library  : displaylib

Description:
DisplayDblScan is used to divide the vertical resolution of the display by
2,4,8 or 16 using Modes 1,2,3 and 4. This is most useful for fast bitmap
based zooms. A Mode of 0 will return the display to 100% magnification.

As with the DisplayRainbow, DisplayRGB, DisplayUser and DisplayScroll
commands DisplayDblScan uses the new line by line copper control of the
display library. To initialise this mode a negative parameter is used
in the CustomCops parameter of the InitCopList command. DisplayDblScan
requires 2 copper instructions per line (make CustomCops=-2).
```

1.544 bum7_displayrainbow

```
Statement: DisplayRainbow
------------------------------------------------------------------------
Syntax   : DisplayRainbow CopList#,Register,Palette[,copoffset]
Library  : displaylib

Description:
DisplayRainbow is used to alter a certain colour register vertically down
a display. It simple maps each colour in a palette to the coresponding
vertical position of the display. ECS displays require one copper
instruction per line while AGA displays require 4.
```

1.545 bum7_displayrgb

```
Statement: DisplayRGB
------------------------------------------------------------------------
Syntax   : DisplayRGB CopList#,Register,line,r,g,b[,copoffset]
Library  : displaylib
```

```
Description:
DisplayRGB is a single line version of DisplayRainbow allowing the
programmer to alter any register of any particular line. As with
DisplayRainbow ECS displays require 1 copper instruction while AGA
requires 4.
```

1.546 bum7_displayscroll

```
Statement: DisplayScroll
------------------------------------------------------------------------
Statement: DisplayScroll CopList#,&xpos.q(n),&xpos.q(n)[,Offset]
Library  : displaylib

Description:
DisplayScroll allows the program to dynamically display any part of a
bitmap on any line of the display. DisplayScroll should always follow the
DisplayBitMap command. The parameters are two arrays holding a list of
xoffsets that represent the difference in horizontal position from the line
above. AGA machines are able to use the fractional part of each entry for
super hiresolution positioning of the bitmap. Three instructions per line
are required for the DisplayScroll command.
```

1.547 bum7_displayuser

```
Statement: DisplayUser
------------------------------------------------------------------------
Syntax   : DisplayUser CopList#,Line,String[,Offset]
Library  : displaylib

Description:
DisplayUser allows the programmer to use their own Copper$ at any line of
the display. Of course copper instructions have to be allocated with the
number of copper instructions in the InitCoplist multiplied by -1.
```

1.548 bum7_duplicatepalette

```
Statement: DuplicatePalette
------------------------------------------------------------------------
Syntax   : DuplicatePalette SrcPalette#,DestPalette#
Library  : palettelib

Description:
DuplicatePalette simply creates a new Palette which exactly matches the
SrcPalette.
```

1.549 bum7_fadepalette

```
Statement: FadePalette
------------------------------------------------------------------------------
Syntax   : FadePalette SrcPalette#,DestPalette#,Brightness.q
Library  : palettelib

Description:
FadePalette multiplies all colours in a Palette by the Brightness argument
and places the result in the DestPalette.
```

1.550 bum7_freemem

```
Statement: FreeMem
------------------------------------------------------------------------------
Syntax   : FreeMem location,size
Library  : banklib

Description:
Used to free any memory allocated with the AllocMem command.
```

1.551 bum7_fromcli

```
Function: FromCLI
------------------------------------------------------------------------------
Function: FromCLI
Library : cliargslib

Description:
Returns TRUE (-1) if your program was run from the CLI, or FALSE (0) if run
from the WorkBench.
```

1.552 bum7_gameb

```
Function: GameB
------------------------------------------------------------------------------
Syntax  : GameB(por#)
Library : gameiolib

Description:
Returns button state of cd32 style game controllers - values returned are:

1  = play/pause
2  = reverse
4  = forward
8  = green
16 = yellow
32 = red
64 = blue

If more than one button is held down, values are added together. For
```

```
example, a value of 6 means both the forward (4) and reverse (2) buttons
are held down. Use an 'and' to isolate the status of a single button, like
this -

;check RED button on port 1...
;
if gameb(1) & 32
  ;
  ;RED button is down...
  ;
else
  ;
  ;RED button is NOT down...
  ;
endif
```

1.553 bum7_gtarrowsize

```
Statement: GTArrowSize
------------------------------------------------------------------------------
Syntax   : GTArrowSize size
Library  : bbgtlib

Description:
Allows the size of GTScroller arrows to be preset. Default size is 16.
```

1.554 bum7_gtstatus

```
Function: GTStatus
------------------------------------------------------------------------------
Syntax  : GTStatus(GTList#,Id)
Library : bbgtlib

Description:
GTStatus returns the status of and gadtools toggle gadgets, a value
of 1 means the the gadget is selected, 0 deselected.
```

1.555 bum7_initpalette

```
Statement: InitPalette
------------------------------------------------------------------------------
Syntax   : InitPalette Palette#,NumColors
Library  : palettelib

Description:
InitPalette simply initialises a palette object to hold NumColors. All
colors will be set to black.
```

1.556 bum7_initshape

```
Statement: InitShape
------------------------------------------------------------------------
Syntax   : InitShape Shape#,Width,Height,Depth
Library  : shapeslib

Description:
InitShape has been added to simple create blank shape objects. Programmers
who make a habit of using ShapesBitMap to render graphics to a shape
object will appreciate this one for sure.
```

1.557 bum7_loadbank

```
Statement: LoadBank
------------------------------------------------------------------------
Syntax   : LoadBank Bank#,FileName$[,MemType]

Description:
The LoadBank command has been modified, instead of having to initialise
the bank before loading a file, LoadBank will now initialise the bank
to the size of the file if it is not already large enough or has not been
initialised at all.
```

1.558 bum7_numpars

```
Function: NumPars
------------------------------------------------------------------------
Syntax  : NumPars
Library : cliargslib

Description:
Returns the number of parameters passed to your program.
```

1.559 bum7_paletterange

```
Statement: PaletteRange
------------------------------------------------------------------------
Syntax   : PaletteRange Palette#,StartCol,EndCol,r0,g0,b0,r1,g1,b1
Library  : palettelib

Description:
PaletteRange creates a spread of colors within a palette. Similar to
DPaint's spread function PaletteRange takes a start and end colour and
creates the color tweens between them.
```

1.560 bum7_par$

```
Function: Par$
------------------------------------------------------------------------------
Syntax  : Par$(parameter#)
Library : cliargslib

Description:
Returns the string value of a parameter.

NOTE: If the parameter asked for is a directory/device/volume etc
(IE NOT A FILE) then Par$(#) will return an empty string.  This is a one
way you can check to see if a file was passed or not.
```

1.561 bum7_parpath$

```
Function: ParPath$
------------------------------------------------------------------------------
Syntax  : ParPath$ (parameter,type)
Library : cliargslib

Description:
This returns the path that this parameter resides in. 'type' specifies
how you want the path returned.

0 You want only the directory of the parameter returned.
1 You want the directory along with the parameter name returned.

EG:

If you passed the parameter "FRED" to your program from WorkBench, and
FRED resides in the directory "work:mystuff/myprograms" then ParPath$(0,0)
will return "work:mystuff/myprograms", but ParPath$(0,1) will return
"work:mystuff/myprograms/FRED".

CAVEAT
The way WB handles argument passing of directories is different to that
of files.  When a directory is passed as an argument, ArgsLib gets an
empty string for the name, and the directory string holds the path to the
passed directory AND the directory name itself.  EG

Passing the blitz2 directory to a program will result in:

Par$(x)     Being an empty string.
ParPath$(x,0)   Being something like work:Basic/blitz2.
ParPath$(x,1) Being work:Basic/blitz2/

YES!  The / is appended!  This is because to keep things simpler, and more
uniform ParPath$(x,1) Is the concatenation of

1)  The directory string passed by Workbench

AND
```

```
2)  A / followed by the name given by WorkBench.

So you can see why the / followed by the empty string occurs.

The easy way around this is simply to check Par$(x), if it is empty, then
use ParPath$(x,0), if it isn't (IE a file was passed) use ParPath$(x,1)
and you will have the entire pathname of the file OR directory.

See the demo program, which handles both cases.

NOTE 2: Is only useable from WorkBench, you will get an error if your
program was run from the CLI and you try to call ParPath$.
```

1.562 bum7_popinput

```
Statement: PopInput & PopOutput
------------------------------------------------------------------------------
Library  : inputoutputlib

Description:
After input or output has been re-directed (eg using windowoutput/
fileoutput), these two commands may be used to return the channel to it's
previous condition.
```

1.563 bum7_readserialmem

```
Statement: ReadSerialMem
------------------------------------------------------------------------------
Syntax   : ReadSerialMem Unit#,Address,Length
Library  : seriallib

Description:
ReadSerialMem will fill the given memory space with data from the
given serial port.
```

1.564 bum7_savepalette

```
Statement: SavePalette
------------------------------------------------------------------------------
Syntax   : SavePalette Palette#,FileName$
Library  : iffmakelib

Description:
Creates a standard IFF "CMAP" file using the given Palette's colors.
```

1.565 bum7_setperiod

```
Statement: SetPeriod
------------------------------------------------------------------------------
Syntax   : SetPeriod Sound#,Period
Library  : audiolib

Description:
Hmmm, not sure why we never included this command in the original
audiolib, SetPeriod simply allows the user to override the frequence
information (period) of the sound object after it has been loaded. To
alter a sound's pitch while playing programmers should hit the audio
hardware direct (hardware locations are listed at the back of the
reference manual).
```

1.566 bum7_writeserialmem

```
Statement: WriteSerialMem
------------------------------------------------------------------------------
Syntax   : WriteSerialMem Unit#,Address,Length
Library  : seriallib

Description:
WriteSerialMem send the given memory space out the given serial port.
```

1.567 allcommands

```
                        All the commands are now sorted, few that was a lot of ↩
                           work!
       But what the hell, it's finished now (until a new BUM is released)

                                Choose a letter:


                     A

                     N

                     B

                     O

                     C

                     P

                     D

                     Q

                     E

                     R
```

F

S

G

T

H

U

I

V

J

W

K

X

L

Y

M

Z

1.568 ind_a

- A -

Activescreen

AnimLoop

ActiveWindow

AppEvent

AddAppIcon

AppIconEvent

AddAppMenu

AppIconFile

AddAppWindow

AppIconHit

ADDValue

AppMenuEvent

AGABlue

AppMenuFile

AGAFillPalette

AppMenuHit

AGAGreen

AppWindowEvent

AGAPalBlue

AppWindowFile

AGAPalGreen

ASLFileRequest

AGAPalRed

ASLFontRequest

AGAPalRGB

ASLScreenRequest$

AGARed

AttachGTList

AGARGB

Avg

AllFire

Avg.L

AllocMem

Avg.Q

AnalyzeDisk

1.569 ind_b

- B -

Bank

BeepScreen

Bin#

BitMapToWindow

BitPlanesBitMap

BlitColl

BLoad

Block

BlockScroll

BSave

ButtonGroup

ButtonId

1.570 ind_c

- C -

CacheOff
CloseDisk
CrMDecrunch

CachePCF

CloseScreen

CustomColors

CharCount

CloseSerial

CustomString

ChDir

CloseWindow

CxAppear

CheckAGA

CludgeShapes

CxChangeList

CheckPrt

CludgeSound

CxDisable

Checksum

ColourRequest

CxDisAppear

ChipFree

CommoditieBase

CxEnable

ChunkHeader

CommodityEvent

CxKill

ChunkyToPlanar

Con_Base

CxUnique

Cipher$

CopyByte

CyclePalette

ClearBitmap

CopyFile

ClearRexxMsg

CopyLong

ClearToolTypes

CopyWord

ClickMouse

CreateArgString

ClipBlit

CreateDisplay

ClipBlitMode

CreateMsgPort

CloseConsole

CreateRexxMsg

1.571 ind_d

- D -

Date$

Derez

DisplayUser

DateFormat

Disable

DosBase

Days

DiskBlocks

Dos_Base

DecodeILBM

DiskCapacity

DuplicatePalette

DecodeMedModule

DiskErrs

DecodePalette

DiskfontBase

DecodeShapes

DiskFree

DecodeSound

DiskUnit

Decrypt

DiskUsed

DeIce

DisplayAdjust

DelAppIcon

DisplayBitmap

DelAppMenu

DisplayControls

DelAppWindow

DisplayDblScan

DeleteArgString

DisplayPalette

DeleteMsgPort

DisplayRainbow

DeleteRexxMsg

DisplayRGB

Deplode

DisplayScroll

Depth

DisplaySprite

1.572 ind_e

- E -

EasyRequest

ExchangeDisAble

Enable

ExchangeDisAppear

Encrypt

ExchangeEnable

EntryBit$

ExchangeKill

EntryComment$

ExchangeMessage

EntryDate

ExchangeUnique

EntryDir

ExecVersion

EntryHour

Exists

EntryMins

EntryName$

EntrySecs

EntrySize

Erase

EraseAll

EventCode

EventQualifier

ExchangeAppear

ExchangeChangeList

1.573 ind_f

- F -

FadeInBitmap

FNSLength

FreeIncData

FadePalette

FNSLoad

FreeMem

FastFree

FNSOrigin

FreePCFCache

FFPBase

FNSOutput

FreeZoneTable

FileFilter

FNSPrefs

Freq

FileReqSize

FNSPrint

FromCLI

FileSize

FNSSetTab

FileStructure

FNSSlot

FillMem

FNSUnderline

FillPalette

FNSUnLoad

FillRexxMsg

FNSVersion

FindToolNumber

FNSWidth

FindToolType

ForceNTSC

FindToolValue

ForcePAL

FNSClip

FormatTrack

FNSClipOutput

Frames

FNSHeight

FreeCatalog

FNSInk

FreeIconObject

1.574 ind_g

- G -

GagetStatus

GTChangeList

GameB

GTGadPtr

GetIconInfo

GTSetAttrs

GetIconObject

GTStatus

GetLocaleStr

GTTags

GetMedInstr

GetMedNote

GetMedVolume

GetResultString

GetRexxCommand

GetRexxResult

GetString$

GetSuperBitmap

GetWheel

Gfx_Base

GraphicsBase

GTArrowSize

GTBevelBox

1.575 ind_h

- H -

HardCopy

Hex#

HideScreen

HotKeyHit

Hours

1.576 ind_i

- I -

IconBase

IncShape

IconDefaultTool

IncSize

IconDefTool$

IncSound

IconRender

IncText$

IconStack

InitAnim

IconSubTool$

InitCopList

IconTool$

InitPalette

IconType

InitShape

ILBMGrab

InitZoomXY

ILBMPalette

InstallFNS

ILBMViewMode

IntuitionBase

Implode

Int_Base

IncBitmap

IsEven
IncData
IsLocale

IncDataAbs

IsReqtoolsActive

IncMed

IsRexxMsg

IncMod

IncNextShape

1.577 ind_j

```
- J -

 JFire

 JHoriz

 JoyC

 JumpMed

 JVert
```

1.578 ind_k

```
- K -

 KeyCode
```

1.579 ind_l

```
- L -

 Largest

 Largest.l

 Largest.q

 LargestFree

 Length

 Lisa

 LoadAnim

 LoadFont

 LoadIFF

 LoadIFF

 LoadMedModule

 LoadPCF

 LoadShape
```

1.580 ind_m

```
- M -


MakeCommodity

MakeDir

MatchToolValue

Max

MemFree

Min

Mins

Months

MoreEntries

MotorOff

MotorOn

MoveScreen
```

1.581 ind_n

```
- N -


NameFile

NewPaletteMode

NewToolType
NewZoneTable

NextBank

NextFile$

NextFrame

NPrintCon

Null

NumDays

NumPars
```

1.582 ind_o

```
- O -

OpenConsole
OpenDisk
OpenSerial
```

1.583 ind_p

```
- P -

PalAdjust
PLoad
PalBlue
Poly
PaletteInfo
Polyf
PaletteRange
PopInput
PalGreen
PopOutput
PalRed
PositionSuperBitmap
Par$
PPDecrunch
ParPath$
PrintCon
PathLock
Processor
PCFDepth
```

PrtCommand

PCFHeight

PrtText

PCFInfo

PutIconObject

PCFVersion

PutSuperBitmap

PCFWidth

Peekto$

PhoneticSpeak

PlanarToChunky

PlayMed

1.584 ind_q

- Q -

Quiet

1.585 ind_r

- R -

ReadSector

Request

RTEZFlagsRequest

RTRequest

ReadSerial

Req_Base

RTEZFontRequest

RTRevision

ReadSerialMem

Reserve

RTEZFreePattern

RTUnlockWindow

ReadSerialMem

ResetTimer

RTEZGetLong

RTVersion

ReadSerialString

RexxError

RTEZGetLongRange

Runerrsoff

Reboot

RexxEvent

RTEZGetString

Runerrson

ReduceX2

RexxsysBase

RTEZLoadFile

ReMap

Rex_Base

RTEZMultiLoadFile

RemoveFNS

RIAnimInit

RTEZPaletteRequest

Rename

RINextAnimFrame

RTEZPathRequest

Repeats

Rrandomize

RTEZRequest

ReplyRexxMsg

Rrnd

RTEZRNextPathEntry

ReqColours

RTASyncPaletteRequest

RTEZSaveFile

ReqFileLoc

RTASyncRequest

RTEZScreenModeRequest

ReqFileRequest

RTCheckASyncPaletteRequest

RTEZSetDefaultDirectory

ReqFileRequest$

RTCheckASyncRequest

RTEZSetPattern

ReqFontSize

RTEndASyncPaletteRequest

RTFileRequest

ReqOutput

RTEndASyncRequest

RTLockWindow

1.586 ind_s

- S -

SaveIncData

SetMedVolume

Space$

SavePalette

SetPeriod

Speak

ScreenHeight

SetSerialBuffer

SpriteMode

ScreenTags

SetSerialLens

Start

ScreenWidth

SetSerialParams

StartMedModule

SearchBegin

SetStatus

StopMed

SearchEnd

SetToolValue

SystemDate

SearchString

SetVoice

Secs

Setzone

SendRexxCommand

ShapeGadget

SerialEvent

ShapeToIcon

SetBPLCON0

ShowBitmap

SetCopyBuffer

ShowPalette

SetGadgetStatus

ShowRequestors

SetHotKey

Smallest

SetIconHit

Smallest.l

SetIconType

Smallest.q

SetMedMask

SortList

1.587 ind_t

- T -

TextRequest

TextTimeout

Ticks

Timer

Translate$

1.588 ind_u

- U -

UnpackIFF

UnpackPCF

UseCatalog

```
UseZoneTable
```

1.589 ind_v

```
- V -

VoiceLoc
Vpos
VwaitPos
```

1.590 ind_w

```
- W -

Wait
WaitFor
WBDepth
WBHeight
WBlit
WBViewMode
WBWidth
WeekDay
Window
WPrintScroll
WriteBoot
WriteSector
WriteSerial
WriteSerialMem
WriteSerialMem
WriteSerialString
WTitle
```

1.591 ind_x

```
- X -


Xor

XOR
```

1.592 ind_y

```
- Y -


Years
```

1.593 ind_z

```
- Z -


Zone

ZoneInit

ZoneTable

ZoneTableSize

ZoneTest

ZoomX2

ZoomX4

ZoomX8

ZoomXY
```