GTLib

COLLABORATORS			
	TITLE : GTLib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 13, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GTLib

1.1 Blitz Basic 2 GadTools Library

The maintainer accepts no responsibility for any damage caused to your system
from the use of this library, or any of the other files in this archive.

Some of the information in this guide might be wrong, so don't blame me, just
contact me so I can get it fixed.

1.2 Introduction

The bbgtlib is a library for AmiBlitz2/Blitz Basic 2 on the Amiga which allows you
to create Intuition gadgets using the gadtools.library (the standard user interface
creation library from OS2 to OS3.1).

Please report any bugs in the library or this guide to the
maintainer
.

1.3 Installing the bbgtlib

If you have AmiBlitz and are using a "decompiled" acidlibs then you simply need to copy the bbgtlib.obj file to whatever directory it currently resides in.

If you have an "acidlibs" file version of Blitz2 then use the supplied script (UpdateAcidlibs). If you have separate library files then copy bbgtlib.obj to blitzlibs:basic/ and then run makedeflibs or use BlitzLibMan to remake your deflibs file (if you have a deflibs file).

1.4 Commands available in the library

If you intend on using this library in your programs, and are looking for any kind of use more advanced than the most basic, I would suggest you get a copy of the Amiga OS Includes & Autodocs, as this will explain many of the tags and options available in these commands.

Most of these descriptions will not use values, but will use the constants which are defined in amigalibs.res. Make sure you have "blitzlibs:amigalibs.res" in the Resident list of the Compiler Options window (found in the Compiler menu).

AFAIK, all the gadgets should produce #IDCMP_GADGETHELP events, assuming you set that in the IDCMP flags for your window to include gadget help events and you are using OS3+.

AttachGTList

DetachGTList

GTActivateGadget

GTArrowSize

GTBevelBox

GTButton

GTChangeCycle

GTChangeList

GTCheckBox

GTCycle

GTDisable

GTEnable

GTEventMicros

GTEventSeconds

GTFreeGadget

GTGadPtr

GTGetAttrs

GTGetInteger

GTGetInternal

GTGetString

GTGZZPosition

GTInteger

GTList

GTListAddress

GTListView

GTMX

GTNewLookProp

GTNumber

GTPalette

GTScroller

GTSetAttrs

GTSetHighlight

GTSetInteger

GTSetString

GTShape

GTSlider

GTStatus

```
                GTString

                GTTags

                GTText

                GTToggle

                GTUnderscore

                GTUserData
               (These are sorted alphabetically (duh!) NOT by token number!)
```

1.5 Example source for the BBGTLib

```
 This archive contains the following examples. You can load the examples
into Blitz2 directly from this guide by clicking on the "Load" buttons in
this page - but you must have the ShowExample script installed correctly
(part of AmiBlitz2 and BSS).

File            Description
gtbutton        Shows the various ways to create and handle GTButton gadgets  Load
gtcheckbox      As above, but for checkbox gadgets (toggle select)  Load
gtcycle         As above, but for cycle gadget (1-of-n select)  Load
gtinteger       As above, but for integer gadgets (numeric entry)  Load
gtlistview      As above, but for listview gadgets (displays scrollable list of  ↩
   text)  Load
gtmx            As above, but for MX gadgets (1-of-n select, all options visible)  ↩
    Load
gtnumber        As above, but for number gadgets (read only numeric display)  Load
gtpalette       As above, but for palette gadgets (colour selection)  Load
gtscroller      As above, but for scroller gadgets (range between limits)  Load
gtshape         As above, but for shape gadgets (any kind of image)  Load
gtslider        As above, but for slider gadgets (single value between limits)   ↩
   Load
gtstring        As above, but for string gadgets (text entry)  Load
gttext          As above, but for text gadgets (read only text display)  Load

gtbevelbox      Shows the different styles of bevel boxes that can be drawn  Load
gtchangecycle   Demo of how to use the GTChangeCycle command   Load
gtchangelist    Demo of the GTChangeList command and proper usage  Load
gtgetattr       Demo of GTGetAttrs when used with a GTCycle     Load
gtgetinteger    GTGetInteger demo on both GTInteger and GTNumber  Load
gtgetstring     Demo of GTGetString on GTString and GTText gadgets  Load
gtnewlookprop   Shows how to use the 2 different modes of GTNewLookProp  Load
gtsetinteger    Sets the contents of integer and number gadgets  Load
gtsetstring     Similar to above, but sets contents of string and text gadgets   ↩
   Load
gttoggle        Toggles the status of a GTButton gadget (in toggle mode)   Load
gtunderscore    Gadget text underlined in different ways with GTUnderscore  Load
gtuserdata      How to set and get the UserData field of Gadtools gadgets  Load
messagetimes    A proof that the GTEvent#? commands work correctly  Load

 Note that you may need to change some of the fonts in the examples - these
```

were added to check that the font used for the gadget could be loaded
correctly.

 And if you are using AmiBlitz then you will probably need to go into the
"Compiler Options" window (from the "Compiler" menu) and change the
"blitzlibs:amigalibs.res" to "blitzlibs:all.res".

1.6 List of bugs in the BBGTLib

```
                None currently known, but please send reports of any you find ( ↩
                   and source code
if possible) to the
                maintainer
               .
```

1.7 The source for the bbgtlib

 The source file is included for the first time. This is so that the library
can be updated, even if I am uncontactable (although I don't know who'd be
interested in this library or doing updates). Anyway, this source was based
on the source for the library by Acid which they released in the library
developer information archive (and has been available on the web for a long
time, along with a lot of their other libs). I had to re-code the same
functionality that was provided by the RWE update (as it was old source) and
have now got around to adding new features. If you do want to make updates to
this library, feel free, and send me a copy or make it publicly available. I
would also ask you to consider NOT breaking anything in the library or changing
the way commands work (too much) - I know, people fear change :)
NB, my including the source in this archive does not mean that I will not
continue to provide bugfixes or updates when necessary.

 The source is 100% ASM and compiles in Blitz 2. If you want to use some other
assembler, you would need to port the library creation macros from Blitz to
a format your assembler would understand.

 The source file can be found in this archive in the Source/gtlib.bb2 file,
which saved as ASCII so it can be read by anything.

1.8 Improvements that need to be made

 I am not saying these will be done by me, it depends on interest and time
available. Of course, anyone else willing to do these is welcome to. This is
just a list of things that I have noticed that could enhance the library.

  * Add more and better error checking for the debugger

  * More examples would be nice

  * Some more items need to be added to the "See Also" sections of the command

```
    descriptions.
```

1.9 History of the gtlib

```
  Current history:
     13th August 2004 (David McMinn)
      * Fixed bug which caused program to crash when GTChangeCycle was used
        when the GTList was attached to a window

     26th April 2003 (David McMinn)
      * Fixed a sma;ll bug which caused GTToggle list,id,state to not work

     12th April 2003 (Bernd Roesch)
      * Updated guide file so that examples load from help in AmiBlitz2.

     25th June 2002 (David McMinn)
      * Removed debug ouput from library which I had mistakenly compiled into it,  ↩
         making all
        the commands really slow.

     27th February 2002 (David McMinn)
      * Removed code to set GadgetRender and SelectRender fields of GTShapes to 0, ↩
          since it
        seems it would clear it for all gadgets - previous versions of the library ↩
            were
        broken in this respect and that is why it worked :)
      * GTToggle bug where using it before attaching a GTList to a Window would  ↩
         cause
        intuition.library to hang is fixed. Added check for NULL window and skips
        refreshing the gadget if that is the case.
      * GTButton and GTShape now have extra flag to set initial status of gadget  ↩
         if they
        are being used in toggle mode"
      * GTGadPtr was accessing the GTList object from the wrong address register  ↩
         as
        so it sometimes did not work for getting gadget pointers.


     1st February 2002 (David McMinn)
      * Rewrite finally complete (phew) ;)


     31st January 2002 (David McMinn)
      * GTGetString, GTGetInteger, GTSetString and GTSetInteger now check the
        gadget type of the gadget to determine what kind of gadget it is
        rather than assuming other values will/will not be 0.


     30th January 2002 (David McMinn)
      * GTGetString re-implemented. Now uses GT_GetGadgetAttrsA_ on OS3+ machines.
      * Same done for GTGetInteger (assumes Gadget\SpecialInfo=0 for GTNumbers)
        has the added bonus of actually working for GTNumbers now (only OS3+
        since the old code is still used for pre-v39)
      * Fixed enforcer hit in GTGetInteger
      * GTGetAttrs quits silently on pre-v39 machines (no more crashing ;)
```

```
 * GTBevelBox checks for GTList not being attached to a window and exits
   (no more crashing when  debugger is turned off)
 * More guide updates and examples


29th January 2002 (David McMinn)
 * All memory in BBGTLib freed when not needed any more
 * GTUnderscore completed
 * GTGadPtr again has only 2 parameters (GTList#,ID.w)
 * GTGZZPos now influences GTBevelBox's
 * GTChangeCycle completed (silly bugs)


28th January 2002 (David McMinn)
 * GTNumber now has flag for controlling whether gadget has border or not.
 * GTShape flags now work correctly. Also, extra flags for making shape
   gadgets into toggle gadgets and turning off IDCMP_GADGETUP messages
   is possible.


25th January 2002 (David McMinn)
 * Updated this guide some more, to include descriptions of what tags cannot
   be used with commands (use the parameters instead), fixed some of the
   types in the command descriptions, command return values, etc.
 * Examples for all gadget creation commands added.
 * More commands found that do not free memory until program exit
   (in fact, the Text$ parameter for all commands). Fixed.
 * Added option in GTText to remove gadget border.
 * Added option to GTScrollers and GTSliders to stop them sending
   IDCMP_GADGETUP events.
 * Added option of getting IDCMP_GADGETDOWN events from GTShape gadgets
 * GTUnderscore added (not completely functional yet)


26th December 2001 (David McMinn)
 * Fixed some mistakes in this guide (wrong types for command parameters).
 * Gadget creation commands are now commands and can optionally return a
   pointer to the gadget that is created (can also be used as a success
   indicator).
 * GTListAddress, GTUserData, GTChangeCycle and GTGetInternal added.
 * Complete rewrite for many internal bugfixes (e.g. non-creation of gadgets
   cannot screw up an entire list, exact gadget pointers stored internally
   for easy access, etc)
 * Memory for GTMX and GTCycle gadgets freed when gadgets are freed, to
   prevent memory fragmentation and slowdown.

1st September 2001 (David McMinn)
 * Updated docs for GTCheckBox, GTMX and GTSlider, for information on scaling ↩
     the
   checkboxes and MX gadgets and other tags for the slider.

28th November 2000 (David McMinn)
 * Fixed GTBevelBox when using debugger, was checking for errors from the  ↩
     wrong address register

26th November 2000 (David McMinn)
 * Added option for GTBevelBox to specify the frame type (OS3+!)
```

```
    * Fixed the above option :(

   25th November 2000 (David McMinn)
    * GTGadPtr now behaves like the original Acid version, although with an ↩
      optional
      parameter it will behave like the update of 15/8/2000.
    * Added GTNewLookProp
    * Bumped date in GTList help string
    * Would like to make all the gadget creation routines commands
      which return a pointer to the gadget created

  15th August 2000 (David McMinn)
    * Fixed problems with GTGadPtr and GTSetAttrs for GTListviews (and possibly ↩
      palettes?)
      [GTGadPtr now searches for the last item in the GTList with the
      ID we are looking for]
    * Latest build date added to the help text of the GTList dumtoke
    * Really need to go through and make all routines use a standard findgad ↩
      routine
      to get the gadget pointer and check the return result from some things

  19th April 2000 (David McMinn)
    * GTGetStatus renamed to GTStatus
    * highlight shape now displayed correctly, was testing the wrong memory ↩
      location
      when checking for the highlight image flag being set

  3rd April 2000 (David McMinn)
    * added detachgtlist
    * added gtgzzposition
    * added gtsethighlight
    * added gtfreegadget but it needs to be rewritten
    * added gtactivategadget
    * fixed possible enforcer death from hell in internal routine "findgadget"
    * hopefully the free gtlist will now also free the Image structures ↩
      allocated for GTShapes and still work with VP
    * and the same with the GTFreeGadget routine

  2nd April 2000 (David McMinn)
    * Started a much needed update
    * Commenting some routines
    * Fixed free gtlist bug for visual prefs
    * fixed gtshapes having wrong PlanePick values
    * fixed gtshapes having wrong highlight images


Ancient history:

   RWE done some nifty updates and released that version, although the
  source was never available.

   Andre Bergmann added GTShape commands and bugfixed some other stuff.

   In the beginning there was Acid, and when Mark Sibly created the Blitz,
  there was much rejoicing. So it was obviously Mark and Simon Armstrong
  that wrote the first versions of the bbgtlib. The source was made
  available in the library developer archive (on some Blitz related FTP
```

site and from Acid).

1.10 Thank you to...

* Mark Sibly, Simon Armstrong and the other guys at Acid that created Blitz2

* Everyone who updated the library before me (RWE, Andre Bergmnn, ???)

* Jean-Marc Gigandet for the idea for the GTNewLookProp command and bug reporting

* Bernd Roesch for reporting the GTCycle and GTMX problem with not freeing memory until program exit, and for suggesting the additions to the IDCMP message pre-handler.

* Kev Harrison for the GTChangeCycle bug report

* Thilo Kohler for the GTEventMicros and GTEventSeconds command suggestions.

* All the guys on the Blitz list for testing the library, providing feedback and bug reports

1.11 Who to contact

Currently, the only contact for updating and providing bugfixes for the GTLib is David McMinn. If anyone else would like to be listed here, give me an email, and make sure you understand the source :)

David McMinn dave@blitz-2000.co.uk ICQ 16827694

If you don't get me at either of those, email the Blitz list, I should be on that under some address.

1.12 attachgtlist

```
                Command name
  AttachGTList

Template
  AttachGTList GTList#,Window#

Parameters
  GTList# - The number of the GTList object you wish to attach
  Window# - The number of the Window object you wish to attach to

Returns
  Nothing

Description
```

```
  Attaches the specified GTList to the specified window. The gadgets in
  the list will now be in the window and will be operational (user can
  use the gadgets). You can only have 1 GTList attached to a window
  at any time. You can only attach a GTList to 1 window at a time.

  You should define all your gadgets in a GTList while it is not attached
  to a window.

See also

              DetachGTList
```

1.13 detachgtlist

```
                Command name
  DetachGTList

Template
  DetachGTList GTList#

Parameters
  GTList# - The number of the GTList object you wish to remove from a window

Returns
  Nothing

Description
  Removes the specified GTList from the window it is attached to. You can only
  detach a GTList once (unless you attach it again before you call this command
  the next time).

See also

              AttachGTList
```

1.14 gtactivategadget

```
                Command name
  GTActivateGadget

Template
  GTActivateGadget GTList#,id

Parameters
  GTList# - The object number of the GTList in which the gadget can be found
  id(.w)  - The ID number of the gadget to activate

Returns
  Nothing
```

```
Description
  Activates a string type gadget (string or integer) or a custom gadget.
  In the case of the string type, the gadget will be activated and the
  cursor will appear in the gadget, allowing you to type into it. In the
  case of custom gadgets, the behaviour will depend on the gadget.

See also

              GTInteger
              ,
              GTString
```

1.15 gtarrowsize

```
                Command name
  GTArrowSize

Template
  GTArrowSize size

Parameters
  size(.l) - The size you wish to set the arrow gadgets on GTScrollers to

Returns
  Nothing

Description
  Sets the current size of the arrow gadgets on GTScroller gadgets. Any
  GTScroller gadgets you create after calling this command will use the
  size specified. In the case or horizontal scrollers, the size sets the
  width of the arrows. In the case of vertical scrollers, the size sets
  the height of the arrows.

  Defaults to 16.

See also

              GTScroller
```

1.16 gtbevelbox

```
                Command name
  GTBevelBox

Template
  GTBevelBox GTList#,x,y,w,h,flags[,type]

Parameters
  GTList#   - the object number of the GTList to take the information for
              the box from
```

```
  x(.l)     - x position of the top left corner
  y(.l)     - y position of the top left corner
  w(.l)     - width of the box
  h(.l)     - height of the box
  flags(.l) - Flag to set what state of box to draw (0=raised,
              anything else=recessed)
  type(.l)  - Specifies the type of frame to draw (optional parameter,
              only works on OS3+). Available types are:
                #BBFT_BUTTON      - Looks like a standard button (default)
                #BBFT_RIDGE       - Looks like a string gadget
                #BBFT_ICONDROPBOX - Imagery suitable for icon drop box

Returns
  Nothing

Description
  Draws a rectangular box (just the edges, does not draw the inner area) on
  the window which the GTList is currently attached to. This is not a gadget,
  only a piece of drawing, so you will need to redraw this yourself when a
  window needs refreshed.

See also

              AttachGTList
```

1.17 gtbutton

```
                Command name
  GTButton

Template
  [*g.Gadget=]GTButton [() GTList#,id,x,y,w,h,Text$,flags[,UserData.l] ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTButton gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above, below or inside the
                 gadget. The flags (below) control the position of this.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
```

```
                    #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                       default)
                    #PLACETEXT_RIGHT - Text$ is located right of the gadget
                    #PLACETEXT_ABOVE - Text$ is located above the gadget
                    #PLACETEXT_BELOW - Text$ is located below the gadget
                    #PLACETEXT_IN    - Text$ is located inside the gadget
                    #NG_HIGHLABEL    - Text$ will be highlighted
                    $40              - Disable (gadget is ghosted and unusable)
                    $80              - Immediate (gadget produces events when  ↩
                       clicked
                                       down, as well as when released)
                    $100             - If the gadget is in toggle mode (see below ↩
                       )
                                       setting this flag will make the
                                       initial state "pressed".
                    $2000            - Makes the gadget a toggle gadget (the
                                       gadget will stay selected until the
                                       user clicks on the gadget again)
  UserData(.l) - Optional parameter which allows you to set the value of
                 the UserData field of the gadget when it is created.

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  Creates a standard push-button type gadget and adds it to the specified
  GTList object.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  You cannot specify GA_Disabled, GA_Immediate or GT_Underscore tags in
  the user specified taglist - use the flags aboive instead.

See also

              GTGZZPosition
              ,
              GTTags
```

1.18 gtchangecycle

```
                Command name
  GTChangeCycle

Template
  [success.l]=GTChangeCycle [() GTList#,id,option$ [)]

Parameters
  GTList# - The number of the GTList object that the gadget is in
  id(.w)  - The ID number of the gadget - THIS MUST BE A CYCLE GADGET!
  Option$ - The options you want to display in the GTCycle. Each option
```

```
            must be separated by a "|" character, e.g. "foo|bar|snafu"

Returns
  This will return true (-1) for successfully being able to change the
  contents of the GTCycle and false (0) if it failed. If it failed, the
  previous contents of the GTCycle will still be there.

  Also can be used without the return value.

Description
  This command is used to change the options which are being offered in
  a GTCycle gadget. You should only use this on GTCycle gadgets, it
  will produce weird results if used on other types of gadgets.

  When the items in the GTCycle gadget are changed, the first item
  in the string will be the one which is displayed.

See also

              GTCycle
```

1.19 gtchangelist

```
               Command name
  GTChangeList

Template
  GTChangeList GTList#,id [ ,List() ]

Parameters
  GTList# - The number of the GTList obejct which contains the GTListview
            you want to change the display of
  id(.w)  - The ID number of the GTListview
  List()  - List to attach or leave out to detach a list

Returns
  Nothing

Description
  This command can either attach a list array or detach a list array from a
  GTListView gadget. If you want to modify a list while the gadgets are
  attached to a window, you need to detach it by calling this command without
  the List() parameter. You can then do whatever you like to the list. Once
  you want to display the list again, call this command and specify the list
  you want to display.

See also

              GTListView
```

1.20 gtcheckbox

```
                Command name
  GTCheckBox

Template
  [*g.Gadget=]GTCheckBox [() GTList#,id,x,y,w,h,Text$,flags ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTCheckBox to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above, below or inside the
                 gadget. The flags (below) control the position of this.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
                   $100             - Checkbox is ticked by default
                   $200             - Checkbox size is scaled to the width and
                                      height specified (V39+)
Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  The GTCheckBox is a single gadget which can either be ticked or unticked.
  The state of the gadget is kept after clicking on it. This command adds
  a GTCheckBox to the specified GTList object.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: You cannot set the GTCB_Scaled, GA_Disabled, GTCB_Checked or
  GT_Underscore tags using the GTTags command before a GTCheckBox, instead
  you should logically OR the values described above to the flags parameter.

See also
```

```
GTGZZPosition
,
GTStatus
,
GTTags
,
GTToggle
```

1.21 gtcycle

```
                Command name
  GTCycle

Template
  [*g.Gadget=]GTCycle [() GTList#,id,x,y,w,h,Text$,flags,Options$[,active] ())]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTCycle gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
  Option$      - The different options you want to be available in the gadget
                 separated by the "|" character
  active(.w)   - the currently active option in the cycle gadget, starting from ↩
     1

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).
```

```
Description
  Creates a GTCycle gadget and adds it to the specified GTList object. The
  Options$ should be specified like "blah|foo|bar" (each option in the gadget
  separated by a "|" character).

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

See also

              GTGZZPosition
              ,
              GTTags
```

1.22 gtdisable

```
                Command name
  GTDisable

Template
  GTDisable GTList#,id

Parameters
  GTList# - The number of the GTList object in which the gadget you wish to
            disable can be found
  id(.w)  - The ID number of the gadget to disable

Returns
  Nothing

Description
  Disables the gadget with the specified ID number in the specified GTList.
  The gadget will be redrawn ghosted and will not accept any input from the
  user until it is enabled.

  I don't think it's possible to disable GTListView's under OS2.

See also

              GTEnable
```

1.23 gtenable

```
                Command name
  GTEnable

Template
  GTEnable GTList#,id
```

```
Parameters
  GTList# - The number of the GTList object which contains the gadget you
            want to enable
  id(.w)  - The ID number of the gadget you want to enable

Returns
  Nothing

Description
  Enables the specified gadget from the specified GTList. This has the
  opposite effect of GTDisable - it will remove the ghosting on the gadget
  and will allow the user to interact with it again.

See also

               GTDisable
```

1.24 gteventmicros

```
                 Command name
  GTEventMicros

Template
  microseconds.l=GTEventMicros

Parameters
  None

Returns
  A long which represents the microseconds value of the last event.

Description
  Each event you recieve from your window has a time associated with it.
  This time is specified as a seconds and microseconds count.This command
  returns the microseconds value. You can use these times for doing things
  like checking for doubleclicks of mousebuttons.

See also

               GTEventSeconds
               , intuition.library/DoubleClick() from OS Autodocs
```

1.25 gteventseconds

```
                 Command name
  GTEventSeconds

Template
  seconds.l=GTEventSeconds
```

```
Parameters
  None.

Returns
  A long which represents the seconds value of the last event.

Description
  Each event you recieve from your window has a time associated with it.
  This time is specified as a seconds and microseconds count.This command
  returns the seconds value. You can use these times for doing things
  like checking for doubleclicks of mousebuttons.

See also

               GTEventMicros
               , intuition.library/DoubleClick() from OS Autodocs
```

1.26 gtfreegadget

```
                 Command name
  GTFreeGadget

Template
  GTFreeGadget GTList#,id

Parameters
  GTList# - The number of the GTList object in which the gadget is
  id(.w)  - The ID number of the gadget to free

Returns
  The number of gadgets removed from the gadget list (optional).

Description
  Will free a single gadget from a GTList. Not sure if it works properly
  and you may have to detach the GTList from the window before doing this.

See also

               DetachGTList
```

1.27 gtgadptr

```
Command name
  GTGadPtr

Template
  *g.Gadget=GTGadPtr(GTList#,id[,last])

Parameters
  GTList#  - The number of the GTList object in which the gadget is
```

```
  id(.w)   - the ID number of the gadget to get the pointer to
  last(.w) - optional parameter, set this to something other than 0
             and it will return the last gadget in the list with the
             correct ID. Default bahaviour (and if this parameter is
             0) is to return the first gadget with the correct ID.

Returns
  A pointer to a Gadget structure which is the gadget you specified.

Description
  This command is intended for use by advanced programmers. It will
  return a pointer to the gadget as specified by the easier method
  of GTList# and Gadget ID number.

  The command by default (and also when last=0) finds the first
  gadget in the list with the correct ID. When you set the value
  of last to something else, you get the last gadget in the list
  with the matching ID.

  The reason for doing this is because gadgets such as GTListViews,
  GTScrollers, GTSliders, etc, are all made up from a number of
  separate gadgets. You may need to get the first gadget that makes
  up these complex gadgets for some reason (like stepping through
  all the parts) or you may want the last gadget (which is the one you
  normally use with the OS functions as the gadget pointer).

See also
```

1.28 gtgetattrs

```
Command name
  GTGetAttrs

Template
  value.l=GTGetAttrs(GTList#,id,Tag)

Parameters
  GTList# - The number of the GTList object which the gadget is in
  id(.w)  - the ID number of the gadget you want the attributes of
  Tag(.l) - the tag value, which is used to specify which attribute
            to enquire about. See the gadtools.library autodoc for
            a list of all the tags avaialble for all gagdet types.

Returns
  A long which is the value of the attribute. Depending on the tag
  you specified, this value will mean many different things.

Description
  This command only works on OS3+! You can use it to find out
  attributes about a specific gadget.

See also
  gadtools.library autodoc
```

1.29 gtgetinteger

```
                Command name
  GTGetInteger

Template
  value.l=GTGetInteger(GTList#,id)

Parameters
  GTList# - The number of the GTList object that the gadget is in
  id(.w)  - The ID number of the integer gadget to get the value of

Returns
  Long which has the same value as the number displayed in the GTInteger
  gadget.

Description
  This command reads the value in the specified GTInteger gadget and
  returns it.

See also

              GTInteger
```

1.30 gtgetinternal

```
Command name
  GTGetInternal

Template
  *first.gtil=GTGetInternal

Parameters
  None.

Returns
  Pointer to the first item in the internal memory lists, which store
  pointers to gadgets from the GTList objects.

Description
  The structure of the internal memories look like this:

  NEWTYPE.gtinlist
    *Succ.gtinlist    ; Pointer to next gadget
    *Gad.Gadget       ; Pointer to gadget returned by CreateGadget
    *Text.b           ; Gadget text (if required)
    Special.l         ; The special stuff (size.l stored at (-4,Special))
    id.w              ; gadget ID
    pad.w
  End NEWTYPE

  NEWTYPE.gtil
    *Succ.gtil
```

```
    *First.gtinlist
    *gtlst.gtcontext ; GTList pointer
    pad.w
  End NEWTYPE

See also
```

1.31 gtgetstring

```
                Command name
  GTGetString

Template
  str$=GTGetString(GTList#,id)

Parameters
  GTList# - The number of the GTList object that the gadget is in
  id(.w)  - The ID number of the GTString gadget that you want to get the
            contents of

Returns
  String which is the same as is currently being displayed in the GTString
  gadget.

Description
  This command reads the contents of the specified gadget (must be a
  GTString!) and returns it as a string.

See also

              GTString
```

1.32 gtgzzposition

```
Command name
  GTGZZPosition

Template
  GTGZZPosition On/Off

Parameters
  On/Off - Sets the global mode for gzzposition at gadget creation on or off

Returns
  Nothing

Description
  This partially controls where the gadgets are located in your window at
  when they are created. When GTGZZPosition is set to off (the default)
  all gadgets will have the left window border width added to the x position
  and the top window border height added to the y position of your gadget.
```

```
  This allows you to use co-ordinates starting from 0,0 to mean the top-
  left of the inner area of the window. When this is set to on, these
  window border sizes will not be added - which is normally only used with
  windows which have the #WFLG_GIMMEZEROZERO flag set as this causes the
  top-left of the inner area of the window to be at 0,0 anyway, so adding
  the window borders is not required (and probably not desired).

See also
  Window command in manual (description of GIMMEZEROZERO flag)
```

1.33 gtinteger

```
                Command name
  GTInteger

Template
  [*g.Gadget=]GTInteger [() GTList#,x,y,w,h,Text$,flags,default ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTInteger gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
                   $80              - Gadget produces a gadget down event as  ↩
                      soon
                                      as it is entered
  default(.l)  - The initial value to set the contents of the gadget to

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).
```

```
Description
  Adds a GTInteger gadget to the specified GTList object. This type of
  gadget is a string entry type gadget, but it only allows digits (0-9) to
  be entered; no decimal point, floating point, or mathematical symbols.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: you cannot use the GTIN_Number tag to set the value of the
  gadget when it is being created - you must use the "Default" parameter.

See also

              GTGetInteger
              ,
              GTGZZPosition
              ,
              GTSetInteger
              ,
              GTTags
```

1.34 gtlist

```
                Command name
  GTList

Template
  N/A

Parameters
  N/A

Returns
  N/A

Description
  GTList is the name of the object used by the bbgtlib. You use it just
  the same way as every other object in Blitz 2. You cannot "Load" or "Save"
  a GTList object however, and you do not really need to "Use" a GTList
  object, as all commands take the object number as a parameter and do not
  depend on the currently used object.

  To display a GTList in a window and start getting input from it, you
  use AttachGTList. You can also remove the GTList from the window using
  DetachGTList. When you recieve events from the gadgets, it is the same
  procedures you use as with old-style gadgets.

  Starting from these updated version of the bbgtlib, pressing the "Help"
  key over the GTList command will give you the date that the library was
  compiled - you can use this as a quick version check.

  For advanced coders, the structure of a GTList object is:
      NEWTYPE.mygtlist
```

```
          mycontext.l        ;0 points to context
          visualinfo.l       ;4 our visual info from current screen
          current.l          ;8 current gadget
          gtwindow.l         ;12 window gadgetlist is attached to
      End NEWTYPE

See also
  "Objects" and "Gadgets" chapters of the Blitz manual,
             AttachGTList
             ,
             DetachGTList
```

1.35 gtlistaddress

```
                Command name
  GTListAddress

Template
  *address.List=GTListAddress(list())

Parameters
  list() - The name of a Blitz2 list array.

Returns
  Pointer to the head of list structure for the specified Blitz2 list array.

Description
  Finds the list structure (they define the start of every OS structured
  list) for the specified Blitz 2 list array. This can be useful when dealing
  with the list contents manually, using the list with OS commands,
  and also things like GTSetAttrs where you may need to pass pointers
  to lists in taglists.

See also

             GTSetAttrs
```

1.36 gtlistview

```
                Command name
  GTListView

Template
  [*g.Gadget=]GTListView [() GTList#,id,x,y,w,h,Text$,flags,list()[,selected[, ↩
     top]] [)]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTListView gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
```

```
                value for every gadget in a GTList. If your program is
                going to run on OS2, you should be aware that gadtools
                uses some gadget ID's internally (and if they clash, you
                will not get any events from the gadgets with the clashing
                ID values). Starting the ID values at about 51 is
                usually safe enough.
  x(.w)       - x position of top left corner of gadget
  y(.w)       - y position of top left corner of gadget
  w(.w)       - width of gadget
  h(.w)       - height of gadget
  Text$       - String to add to the gadget. This string can be placed
                either to the left, right, above or below the gadget.
                The flags (below) control the position of this text.
  flags(.l)   - The flags control many aspects of the gadget. The flags
                that can be used with this type of gadget (combine them
                using the "or" symbol: "|") are:
                  (you can only use one of the #PLACETEXT flags at any time)
                  #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                     default)
                  #PLACETEXT_RIGHT - Text$ is located right of the gadget
                  #PLACETEXT_ABOVE - Text$ is located above the gadget
                  #PLACETEXT_BELOW - Text$ is located below the gadget
                  #NG_HIGHLABEL    - Text$ will be highlighted
                  $40              - Disable (gadget is ghosted and unusable,
                                     perhaps only works on OS3+)
                  $1000            - GTListView is read-only
  list()      - The list of items to display in the listview
  selected(.l) - The number of the currently selected item in the listview
  top(.l)     - The first item to display, at the top of the listview

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTListView gadget to the specified GTList object. A
  GTListView displays a list of strings and provides a scroller and arrows
  for navigating the list.

  You should do an AddIDCMP #LISTVIEW_IDCMP before you
  open your window so that the listview gadget behaves correctly.

  The list you attach to the GTListView must be a List array, and must have
  the following structure to the type that each item is:
      NEWTYPE.gtlv
          pad.w       ; or any other variable you want, but it must be a .w
          text.s      ; this is the text that gets displayed in the listview
          ...         ; any other fields can come after the first two
      END NEWTYPE

  For calculating the selected and top parameters, the first item displayed
  in the listview is item number 0. To make full use of the selected
  parameter, you also need to specify the #GTLV_ShowSelected tag. If you
  set the value of this tag to 0 you get a string gadget under the listview
  for OS2.04/5 and a highlight bar for OS3+. If you set it to anything else
  than 0, it must be a pointer to a string gadget, which will show the
```

```
  selected item (I think this is the only way to use this tag on OS2.00).
  I also think that the string gadget must be the same width and x position
  as the listview.

  NB: You cannot use the GTLV_Selected, GTLV_Top, GTLV_Labels or GTLV_ReadOnly
  tags when you are creating a GTListView - you must use the abilities of this
  command (the list, selected and top parameters, and the $1000 readonly flag)
  to set those items.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

See also

              GTGZZPosition
              ,
              GTTags
```

1.37 gtmx

```
                Command name
  GTMX

Template
  [*g.Gadget=]GTMX [(] GTList#,id,x,y,w,h,Text$,flags,Option$[,active] [)]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTMX gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
```

```
                    $200             - Scale each MX button to the specified  ↩
                       width and height.
  Option$       - The text for the different options
  active(.l)    - Which option is active by default

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTMX gadget to the specified GTList object. The GTMX
  gadget is a "1 out of n" choice type gadget (one option out of n is always
  selected).

  The width and height are usually ignored, but you can add the #GTMX_Scaled
  tag (and set it to true) under OS3+ and the buttons will be scaled to the
  size specified.

  The #PLACETEXT... flags in the case of the GTMX gadget determines where the
  text for each option lines up, not the Text$. Usually, the Text$ is ignored
  by the OS, but you can add the #GTMX_TitlePlace (OS3+ only!) to add the Text$
  to the gadget.

  The options are specified in a string, with each option separated by the "|"
  character, i.e. "blah|foo|bar" would give you three options: blah, foo and
  bar.

  You will want to add AddIDCMP #MX_IDCMP before you
  open your window and then check for GTMX events by checking for the
  #IDCMP_GADGETDOWN event.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: You cannot use the GTTags command to set the GTMX_Scaled tag - you must do
  that by logically OR'ing the value $200 to the flags. You also cannot use the
  GTMX_Active tag to set the initial active option, you must use the active
  parameter.

See also

              GTGZZPosition
              ,
              GTTags
```

1.38 gtnewlookprop

```
                Command name
  GTNewLookProp

Template
  GTNewLookProp [Mode=On/Off] or [GTList,ID,On/Off]
```

```
Parameters
  Mode - Sets the default mode for when creating gadgets which contain a
         slider element.

  GTList# - The number of the GTList object in which to find the gadget
  ID(.w)  - The ID number of the gadget to set the prop appearance
  On/Off  - Turns on or off the new look for the prop

Returns
  Nothing

Description
  This command works in two modes:

  1) When using a single parameter, this controls the global setting for
     all gadgets with a slider element created after calling this command.
     If this is set to "On", all the gadgets created which have a
     slider element will get that slider set to use the new (OS2+)
     look. When "Off", the appearance of the slider will be the
     standard look. Defaults to "Off".

  2) When using three parameters you can change the look of a single gadget
     that you specify with the GTList# and gadget ID number. You may have
     to manually re-draw the gadget after calling this command if the
     GTList is already attached to a window.

See also

              GTListView
              ,
              GTScroller
              ,
              GTSlider
```

1.39 gtnumber

```
                Command name
  GTNumber

Template
  [*g.Gadget=]GTNumber [() GTList#,id,x,y,w,h,Text$,flags,value ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTNumber gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
```

```
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $8000            - Remove border from text gadget
  value(.l)    - The value to be displayed initially in the gadget

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTNumber gadget to the specified GTList. The GTNumber
  gadget is a read only numeric display gadget, a bit like the GTInteger.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: You cannot use the GTNM_Number tag to set the initial value of the
  gadget, you use the value parameter in the command.

See also

              GTGZZPosition
              ,
              GTInteger
              ,
              GTSetInteger
              ,
              GTTags
```

1.40 gtpalette

```
                Command name
  GTPalette

Template
  [*g.Gadget=]GTPalette [() GTList#,id,x,y,w,h,Text$,flags,depth[,Color] ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTPalette gadget to
```

```
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
  depth(.l)    - The depth of the gadget, effectively the number of colours
                 (number of colours = 2 ^ depth)
  Color(.w)    - The initial colour to be highlighted in the palette

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTPalette gadget to the specified GTList object.
  The GTPalette gadget is a colour selection gadget, and displays
  a set of boxes, each one showing a different colour from the palette
  of the screen that the gadget is displayed on.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: You cannot use the GTPA_Depth or GTPA_Color tags to set those
  values, you must use the parameters which are available in the command.
  You should also use GTPA_IndicatorWidth or GTPA_IndicatorHeight tag
  when you use the Color parameter.

See also

              GTGZZPosition
              ,
              GTTags
```

1.41 gtscroller

```
                Command name
  GTScroller

Template
  [*g.Gadget=]GTScroller [() GTList#,id,x,y,w,h,Text$,flags,Visible,Total[,Top]  ↩
     [)]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTPalette gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above or below the gadget.
                 The flags (below) control the position of this text.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
                   (you can only use one of the #PLACETEXT flags at any time)
                   #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                      default)
                   #PLACETEXT_RIGHT - Text$ is located right of the gadget
                   #PLACETEXT_ABOVE - Text$ is located above the gadget
                   #PLACETEXT_BELOW - Text$ is located below the gadget
                   #NG_HIGHLABEL    - Text$ will be highlighted
                   $40              - Disable (gadget is ghosted and unusable)
                   $80              - Report IDCMP_GADGETDOWN events
                   $400             - Make slider vertical instead of horizontal ↩
                      .
                   $800             - Display arrow gadgets
                   $4000            - Do not report IDCMP_GADGETUP events for  ↩
                      the
                                      scroller (defaults to reporting these  ↩
                                         events,
                                      which is the opposite of what is described ↩
                                          in
                                      the autodocs)
  Visible(.l)  - Size of part of the total range that can be seen
  Total(.l)    - The total range that the scroller represents
  Top(.l)      - The first value that can be seen currently

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).
```

```
Description
  This command adds a GTScroller to the specified GTList object. A GTScroller
  comprises a slider part and two arrow gadgets. If you think the last three
  parameters are confusing, think of drawer windows on the Workbench: there
  are two GTScrollers (one for horizontal and one for vertical). The size of
  the knob is related to the visible size of the window (the Visible
  parameter) and the total size of the window (Total) affects how big the
  knob is. Of course, it does not have to be limited to use for scrolling
  around areas that are larger than the current window, but that is the most
  common use.

  NB: You cannot use the #PGA_Freedom tag to set the direction of the slider,
  you must use the $400 flag. If you want arrow gadgets displayed you must
  make sure to set the $800 flag. You can specify the size of the arrows with
  GTArrowSize (not the GTSC_Arrows tag). You also cannot use the GA_Disabled,
  GA_Relverify, GA_Immediate, GTSC_Top, GTSC_Total and GTSC_Visible tags as
  these can all be controlled by the parameters of the command.

  You will probably want to add AddIDCMP #SCROLLER_IDCMP
  before you open your window to make sure the scroller behaves as it should.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

See also

              GTArrowSize
              ,
              GTGZZPosition
              ,
              GTNewLookProp
              ,
              GTTags
```

1.42 gtsetattrs

```
Command name
  GTSetAttrs

Template
  GTSetAttrs GTList#,id [,Tag,Value...]

Parameters
  GTList#   - The number of the GTList object
  id(.w)    - The ID number for the gadget
  Tag(.l)   - The tag to set the value of
  Value(.l) - The value of the tag

Returns
  Nothing

Description
  This command changes some attributes of a specific gagdet, via the use
```

```
  of tags. You can specify the tag and the value to set for it. You can
  specify multiple tags at the same time, just keep repeating the Tag,Value
  parameters on the end of the command. Tags are specific to the type
  of gadget in question and some can only be set when the gadget is first
  created. You can find a list of all the tags for all the gadgets in the
  gadtools.library autodoc.

See also
  GT_SetGadgetAttrs command in the gadtools.library autodoc
```

1.43 gtsethighlight

```
                Command name
  GTSetHighlight

Template
  GTSetHighlight GTList#,id,value

Parameters
  GTList#   - The number of the GTList object that the gadget is in
  id(.w)    - The ID number of the GTListView you want to highlight an item from
  value(.w) - The item you want to set as highlighted

Returns
  Nothing

Description
  This command will set the highlighted item it in a GTListView gadget.
  The item numbers start from 0 as the first item in the list. Under OS3+
  you can specify -1 to unhighlight an item.

See also

              GTListView
```

1.44 gtsetinteger

```
                Command name
  GTSetInteger

Template
  GTSetInteger GTList#,id,value

Parameters
  GTList#   - The number of the GTList object that the gadget is in
  id(.w)    - The ID number of the GTInteger or GTNumber gadget to set the
              value of
  value(.l) - The value to display in the gadget

Returns
  Nothing
```

```
Description
  This command sets the value you pass as the value which is being displayed
  in either GTInteger or GTNumber gadgets.

See also

              GTInteger
              ,
              GTNumber
```

1.45 gtsetstring

```
                Command name
  GTSetString

Template
  GTSetString GTList#,id,string$

Parameters
  GTList# - The number of the GTList object that the gadget is in
  id(.w)  - The ID number of the GTString or GTText gadget to set the
            string of
  string$ - The string to display in the gadget

Returns
  Nothing

Description
  This command sets the string you pass as the text which is being displayed
  in either GTString or GTText gadgets.

See also

              GTString
              ,
              GTText
```

1.46 gtshape

```
                Command name
  GTShape

Template
  [*g.Gadget=]GTShape [(] GTList#,id,x,y,flags,Shape#[,Shape#] [)]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTShape gadget to
  id(.w)       - The ID number for this gadget. This should be a unique
```

```
                value for every gadget in a GTList. If your program is
                going to run on OS2, you should be aware that gadtools
                uses some gadget ID's internally (and if they clash, you
                will not get any events from the gadgets with the clashing
                ID values). Starting the ID values at about 51 is
                usually safe enough.
  x(.w)       - x position of top left corner of gadget
  y(.w)       - y position of top left corner of gadget
  w(.w)       - width of gadget
  h(.w)       - height of gadget
  flags(.l)   - The flags control many aspects of the gadget. The flags
                that can be used with this type of gadget (combine them
                using the "or" symbol: "|") are:
                  $40   - Disable (gadget is ghosted and unusable)
                  $80   - Immediate (gadget produces events when clicked
                          down, as well as when released)
                  $100  - If the gadget is in toggle mode (see below) setting
                          this flag will make the initial state "pressed".
                  $2000 - Makes the gadget a toggle gadget (the
                          gadget will stay selected until the
                          user clicks on the gadget again)
                  $4000 - Disabled relverify operation of gadget (does not
                          send any IDCMP_GADGETUP events)
  Shape#      - The number of the Shape object to use as the imagery for
                the gadget
  Shape#      - An optional parameter to allow you to specify the number
                of a Shape object that gets displayed when the object is
                highlight (either pressed by the user, or set in software)

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a generic gadget to the specified GTList, which gets
  its imagery from the specified Shape objects.

See also

              GTGZZPosition
              ,
              GTTags
```

1.47 gtslider

```
                Command name
  GTSlider

Template
  [*g.Gadget=]GTSlider [() GTList#,id,x,y,w,h,Text$,flags,Min,Max[,Level] [)]

Parameters
  GTList#     - The number of the GTList object you want to add the new
```

```
                GTSlider gadget to
  id(.w)      - The ID number for this gadget. This should be a unique
                value for every gadget in a GTList. If your program is
                going to run on OS2, you should be aware that gadtools
                uses some gadget ID's internally (and if they clash, you
                will not get any events from the gadgets with the clashing
                ID values). Starting the ID values at about 51 is
                usually safe enough.
  x(.w)       - x position of top left corner of gadget
  y(.w)       - y position of top left corner of gadget
  w(.w)       - width of gadget
  h(.w)       - height of gadget
  Text$       - String to add to the gadget. This string can be placed
                either to the left, right, above or below the gadget.
                The flags (below) control the position of this text.
  flags(.l)   - The flags control many aspects of the gadget. The flags
                that can be used with this type of gadget (combine them
                using the "or" symbol: "|") are:
                  (you can only use one of the #PLACETEXT flags at any time)
                  #PLACETEXT_LEFT  - Text$ is located left of the gadget ( ↩
                     default)
                  #PLACETEXT_RIGHT - Text$ is located right of the gadget
                  #PLACETEXT_ABOVE - Text$ is located above the gadget
                  #PLACETEXT_BELOW - Text$ is located below the gadget
                  #NG_HIGHLABEL    - Text$ will be highlighted
                  $40              - Disable (gadget is ghosted and unusable)
                  $80              - Gadget sends IDCMP_GADGETDOWN events ( ↩
                     immediate)
                  $400             - Make slider vertical instead of horizontal ↩
                     .
                  $4000            - Stop gadget from sending IDCMP_GADGETUP
                                     events (relverify)
  Min(.l)     - The minimum value that the slider can be set to
  Max(.l)     - The maximum value that the slider can be set to
  Level(.l)   - The initial value to set the slider position to (default=0)

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTSlider gadget to the specified GTList object. A
  GTSlider gadget is a gadget which sllows you to position a knob between
  two limits. This gadget is usually used to set a specific value in a range
  rather than set a range of values within a range (like GTScroller). Typical
  uses are a volume control, etc.

  NB: You cannot use the #PGA_Freedom tag to set the direction of the slider,
  you must use the $400 flag. You also cannot use the GA_Disabled, GA_Immediate,
  GTSL_RelVerify, GTSL_Min, GTSL_Max or GTSL_Level tags, as these are all
  controlled by the parameters in this command.

  You will probably want to add AddIDCMP #SLIDER_IDCMP
  before you open your window to make sure the slider behaves as it should.

  The font that the text for the gadget appears in is the currently "Used"
```

```
    Intuifont object.

  See also

                GTGZZPosition
                ,
                GTTags
```

1.48 gtstatus

```
                  Command name
    GTStatus

  Template
    status.b=GTStatus(GTList#,Id)

  Parameters
    GTList# - The number of the GTList object in which the gadget can be found
    Id(.w)  - The ID number of the gadget you want to get the status of

  Returns
    Byte representing the state of the gadget in question, either 0 for
    unselected/unhighlighted, or -1 for selected/highlighted.

  Description
    This command checks the status of boolean gadgets (single gadgets which
    can be pressed or unpressed, such as checkboxes, buttons or shapes).

  See also

                GTButton
                ,
                GTCheckBox
                , GTShape
```

1.49 gtstring

```
                  Command name
    GTString

  Template
    [*g.Gadget=]GTString [() GTList#,id,x,y,w,h,Text$,flags,MaxChars [,default$]  ↩
       [)]

  Parameters
    GTList#      - The number of the GTList object you want to add the new
                   GTString to
    id(.w)       - The ID number for this gadget. This should be a unique
                   value for every gadget in a GTList. If your program is
                   going to run on OS2, you should be aware that gadtools
```

```
                uses some gadget ID's internally (and if they clash, you
                will not get any events from the gadgets with the clashing
                ID values). Starting the ID values at about 51 is
                usually safe enough.
  x(.w)       - x position of top left corner of gadget
  y(.w)       - y position of top left corner of gadget
  w(.w)       - width of gadget
  h(.w)       - height of gadget
  Text$       - String to add to the gadget. This string can be placed
                either to the left, right, above, below or inside the
                gadget. The flags (below) control the position of this.
  flags(.l)   - The flags control many aspects of the gadget. The flags
                that can be used with this type of gadget (combine them
                using the "or" symbol: "|") are:
                  (you can only use one of the #PLACETEXT flags at any time)
                  #PLACETEXT_LEFT  - Text$ is located left of the gadget
                  #PLACETEXT_RIGHT - Text$ is located right of the gadget
                  #PLACETEXT_ABOVE - Text$ is located above the gadget
                  #PLACETEXT_BELOW - Text$ is located below the gadget
                  #NG_HIGHLABEL    - Text$ will be highlighted
                  $40              - Disable (gadget is ghosted and unusable)
                  $80              - Gadget produces an IDCMP_GADGETDOWN event
                                     when entered (immediate) OS3+
  MaxChars(.l) - The maximum number of characters that this gadget can hold
  default$    - The initial string to be displayed in the gadget

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTString gadget to the specified GTList object. A
  GTString gadget is a text entry gadget, where any character can be
  entered.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

  NB: You cannot use the tags GA_Disabled, GA_Immediate, GTST_MaxChars and
  GTST_String when you use this command since they are all controlled by
  the parameters of this command.

See also

              GTGetString
              ,
              GTGZZPosition
              ,
              GTSetString
              ,
              GTTags
```

1.50 gttags

```
Command name
  GTTags

Template
  GTTags Tag,Value [,Tag,Value...]

Parameters
  Tag(.l)   - The tag you want to set
  Value(.l) - the value you want to set the tag to

Returns
  Nothing

Description
  This command is used to set the tags which are used in the creation of the
  next gadget (and ONLY the next gadget, the tags are cleared afterwards).
  You can specify multiple tags, just keep adding them to the command call,
  as long as you specify both the Tag and Value on each occaision.

  You can vary the gadgets a LOT with the tags.

See also
  CreateGadgetA command in gadtools.library autodoc
```

1.51 gttext

```
                Command name
  GTText

Template
  [*g.Gadget=]GTText [() GTList#,id,x,y,w,h,Text$,flags,Display$ ()]

Parameters
  GTList#      - The number of the GTList object you want to add the new
                 GTString to
  id(.w)       - The ID number for this gadget. This should be a unique
                 value for every gadget in a GTList. If your program is
                 going to run on OS2, you should be aware that gadtools
                 uses some gadget ID's internally (and if they clash, you
                 will not get any events from the gadgets with the clashing
                 ID values). Starting the ID values at about 51 is
                 usually safe enough.
  x(.w)        - x position of top left corner of gadget
  y(.w)        - y position of top left corner of gadget
  w(.w)        - width of gadget
  h(.w)        - height of gadget
  Text$        - String to add to the gadget. This string can be placed
                 either to the left, right, above, below or inside the
                 gadget. The flags (below) control the position of this.
  flags(.l)    - The flags control many aspects of the gadget. The flags
                 that can be used with this type of gadget (combine them
                 using the "or" symbol: "|") are:
```

```
                    (you can only use one of the #PLACETEXT flags at any time)
                    #PLACETEXT_LEFT  - Text$ is located left of the gadget
                    #PLACETEXT_RIGHT - Text$ is located right of the gadget
                    #PLACETEXT_ABOVE - Text$ is located above the gadget
                    #PLACETEXT_BELOW - Text$ is located below the gadget
                    #NG_HIGHLABEL    - Text$ will be highlighted
                    $8000            - Remove border from text gadget
  Display$     - The initial string to be displayed in the gadget

Returns
  Pointer to the gadget which has been created. At a simpler level, it can
  be used to show whether the command was a success (non-zero value will be
  returned) or not (returns 0 for failure).

Description
  This command adds a GTText gadget to the specified GTList object. A
  GTText gadget is a read-only text display gadget, where any character
  can be displayed.

  The font that the text for the gadget appears in is the currently "Used"
  Intuifont object.

See also

              GTGZZPosition
              ,
              GTSetString
              ,
              GTTags
```

1.52 gttoggle

```
                Command name
  GTToggle

Template
  GTToggle GTList#,Id [,On|Off]

Parameters
  GTList# - The number of the GTList object in which the gadget can be found
  Id(.w)  - The ID number of the gadget you want to get the status of
  On|Off  - The state which you want to set the gadget to

Returns
  Nothing

Description
  This command sets the state of boolean gadgets (single gadgets which
  can be pressed or unpressed, such as checkboxes, buttons or shapes).
  Setting it to On highlights/selects/presses the gadget, while off does
  the opposite.

  If you leave out the On|Off parameter the effect of this command is to
  toggle the current state, i.e. On->Off, Off->On.
```

```
See also

              GTButton
              ,
              GTCheckBox
              , GTShape
```

1.53 gtunderscore

```
Command name
  GTUnderscore

Template
  GTUnderscore char

Parameters
  char(.w) - The ASCII code of the character to indicate the underscore
             in gadget texts.

Returns
  Nothing

Description
  When using gadtools.library, you can put special characters in the texts
  for the gadgets which will tell gadtools.library to underline the next
  character in the string. The default value is 95, which corresponds to
  the underscore character ("_"). So, for example, a gadget text like
  "_Hello there" will be drawn in your window like "Hello there".

  The easiest way to get the ASCII value for the character you want to use
  is to use something like: GTUnderscore Asc("_")
  But replacing the underscore character with whatever character you want.

See also
```

1.54 gtuserdata

```
Command name
  GTUserData

Template
  GTUserData userdata

Parameters
  userdata(.l) - The userdata value for the next gadget to be created.

Returns
  Nothing

Description
```

```
    This command sets the userdata field of the next gadget to be created.
    The userdata field is completely for use by the programmer and therefore
    whatever value you set here will be available in the userdata field of
    the created gadget. Up to you what you use it for, which can be anything.

  See also
```