

GadUtil

COLLABORATORS

	<i>TITLE :</i> GadUtil		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	GadUtil	1
1.1	GadUtil.doc	1
1.2	gadutil.library/GU_LayoutGadgetsA	3
1.3	gadutil.library/GU_FreeLayoutGadgets	14
1.4	gadutil.library/GU_CreateGadgetA	14
1.5	gadutil.library/GU_SetGadgetAttrsA	16
1.6	gadutil.library/GU_GetIMsg	17
1.7	gadutil.library/GU_CountNodes	18
1.8	gadutil.library/GU_GadgetArrayIndex	18
1.9	gadutil.library/GU_BlockInput	19
1.10	gadutil.library/GU_FreeInput	19
1.11	gadutil.library/GU_FreeGadgets	20
1.12	gadutil.library/GU_SetGUGadAttrsA	21
1.13	gadutil.library/GU_CoordsInGadBox	21
1.14	gadutil.library/GU_GetGadgetPtr	22
1.15	gadutil.library/GU_TextWidth	24
1.16	gadutil.library/GU_GetLocaleStr	25
1.17	gadutil.library/GU_CreateLocMenuA	25
1.18	gadutil.library/GU_OpenCatalog	26
1.19	gadutil.library/GU_CloseCatalog	28
1.20	gadutil.library/GU_DisableGadget	28
1.21	gadutil.library/GU_SetToggle	29
1.22	gadutil.library/GU_RefreshBoxes	29
1.23	gadutil.library/GU_RefreshWindow	30
1.24	gadutil.library/GU_OpenFont	31
1.25	gadutil.library/GU_NewList	31
1.26	gadutil.library/GU_ClearList	32
1.27	gadutil.library/GU_DetachList	32
1.28	gadutil.library/GU_AttachList	33
1.29	gadutil.library/GU_AddTail	33

1.30	gadutil.library/GU_ChangeStr	34
1.31	gadutil.library/GU_CreateContext	34
1.32	gadutil.library/GU_GetGadgetAttrsA	35
1.33	gadutil.library/GU_CreateMenusA	36
1.34	gadutil.library/GU_FreeMenus	36
1.35	gadutil.library/GU_LayoutMenuItemsA	37
1.36	gadutil.library/GU_LayoutMenusA	38
1.37	gadutil.library/GU_GetVisualInfoA	38
1.38	gadutil.library/GU_FreeVisualInfo	39
1.39	gadutil.library/GU_BeginRefresh	39
1.40	gadutil.library/GU_EndRefresh	40
1.41	gadutil.library/GU_FilterIMsg	41
1.42	gadutil.library/GU_PostFilterIMsg	41
1.43	gadutil.library/GU_ReplyIMsg	42
1.44	gadutil.library/GU_DrawBevelBoxA	43
1.45	gadutil.library/GU_FindNode	43
1.46	gadutil.library/GU_NodeUp	44
1.47	gadutil.library/GU_NodeDown	44
1.48	gadutil.library/GU_UpdateProgress	45

Chapter 1

GadUtil

1.1 GadUtil.doc

GU_LayoutGadgetsA()
GU_FreeLayoutGadgets()
GU_CreateGadgetA()
GU_SetGadgetAttrsA()
GU_GetIMsg()
GU_CountNodes()
GU_GadgetArrayIndex()
GU_BlockInput()
GU_FreeInput()
GU_FreeGadgets()
GU_SetGUGadAttrsA()
GU_CoordsInGadBox()
GU_GetGadgetPtr()
GU_TextWidth()
GU_GetLocaleStr()
GU_CreateLocMenuA()
GU_OpenCatalog()
GU_CloseCatalog()
GU_DisableGadget()

GU_SetToggle ()
GU_RefreshBoxes ()
GU_RefreshWindow ()
GU_OpenFont ()
GU_NewList ()
GU_ClearList ()
GU_DetachList ()
GU_AttachList ()
GU_AddTail ()
GU_ChangeStr ()
GU_CreateContext ()
GU_GetGadgetAttrsA ()
GU_CreateMenusA ()
GU_FreeMenus ()
GU_LayoutMenuItemsA ()
GU_LayoutMenusA ()
GU_GetVisualInfoA ()
GU_FreeVisualInfo ()
GU_BeginRefresh ()
GU_EndRefresh ()
GU_FilterIMsg ()
GU_PostFilterIMsg ()
GU_ReplyIMsg ()
GU_DrawBevelBoxA ()
GU_FindNode ()
GU_NodeUp ()
GU_NodeDown ()

1.2 gadutil.library/GU_LayoutGadgetsA

NAME

GU_LayoutGadgetsA -- Formats an array of GadTools gadgets.

SYNOPSIS

```
gad_info = GU_LayoutGadgetsA(gad_list, gadgets, screen, taglist)
D0,A0                A0          A1          A2          A3
```

```
APTR GU_LayoutGadgetsA(struct Gadget **, struct LayoutGadget *,
    struct Screen *, struct TagItem *);
```

FUNCTION

Creates a laid-out gadget list from a LayoutGadget array, which describes each gadget you want to create. Gadgets you create can be any of the gadget kinds supported by GadTools, as well as any of the extended gadget kinds provided by GadUtil. The gadgets created by this routine, can easily be defined so that they adjust their sizes and positions to accomodate fonts of any size, and also adapt to different locale strings.

INPUTS

gad_list - a pointer to the gadget list pointer. This will be ready to pass to OpenWindowTagList() or AddGList().

gadgets - an array of LayoutGadget structures. Each element in the array describes one of the gadgets that you will be creating. Each LayoutGadget structure in the array should be initialized as follows:

lg_GadgetID - the ID for this gadget. An ID of -1 terminates the array.

lg_LayoutTags - tags that describes each gadget to create. These tags is used to calculate positions, sizes and other attributes of the created gadgets.

lg_GadToolsTags - additional tags for GadTools gadgets. This would be the same set of tags that you might pass to CreateGadgetA() if you were using GadTools directly.

lg_Gadget - the pointer to the Gadget structure created for this gadget will be placed here. You should initialize this field to NULL. The gadget structure created should be considered READ ONLY! This field will contain a pointer to a struct BBoxData, if the created gadget kind is a BEVELBOX_KIND.

Assembly language programmers can use the macro GADGET:

```
GADGET GadgetID, Gad_LayoutTags, Gad_GadToolsTags
```

screen - a pointer to the screen that the gadgets will be created

for. This is required, so that the layout routines can get display info about the screen where the rendering will be done. Use `LockPubScreen()` to use a public screen, or `OpenScreenTagList()`, if you want to use your own screen.

`taglist` - pointer to a taglist (see in a later release of this library for allowed tags, or [click here](#) to jump to the PRELIMINARY DOCS for this function).

RESULT

`gad_info` - a pointer to a private structure. You must keep this value and pass it to

```
GU_FreeLayoutGadgets()
    later on
```

in order to free up all resources used by your gadgets. This pointer is also used in a lot of other functions in this library.

NOTES

You must be careful with the `taglist` in the `lg_LayoutTags` field. Tags are processed sequentially in the order you give them in, and if a tag references another gadget (eg. the `GL_TopRel` tag), then processing of the current gadget halts while the referenced gadget is processed (if it has not already been processed). Problems can occur if this gadget refers back to the original gadget that referenced it, if it is referring to a field that has not yet been processed in that gadget.

SEE ALSO

```
GU_FreeLayoutGadgets()
,
GU_CreateGadgetA()
, gadtools/CreateGadgetA()
    GadUtil tags
    ~~~~~
```

Tags to use for the yet undocumented function `GU_LayoutGadgetsA()`.

Tags to use in the tag lists for each gadget to create. Note that you do not have to specify any tags that not change from gadget to gadget.

GU_GadgetKind (ULONG)

Can be any of the standard GadTools gadget kinds, or one of the extensions provided by GadUtil. Currently extended types are:

IMAGE_KIND

A gadget that uses an Intuition Image structure for its contents. Selected and unselected states can use different images. The images are centered automatically.

Extra tags for `IMAGE_KIND`:

```
GUIM_Image (struct Image *)
    Image for the gadget in its unselected state. This is
    the only required (extra) tag for IMAGE_KIND gadgets.
```

GUIM_SelectImg (struct Image *)
Image for the gadget in its selected state. If this tag is omitted, the selected image will be the same as the unselected, and only the border and the background color will change (depending on the GUIM_BOOPSIlook tag).

GUIM_ReadOnly (BOOL)
TRUE to create a read-only image gadget.

GUIM_BOOPSIlook (BOOL)
This tag will allow the programmer to select how the secondary image should be shown, if only one image is used for the gadget. Defaults to TRUE, which means that the background color will change when the user selects the gadget.

DRAWER_KIND

A "select drawer" image button. This can be used to select a path, but is often used to select files.

FILE_KIND

A "select file" image button. This can be used to allow the user to select a file. Most programs uses the DRAWER_KIND for both file and path selection.

BEVELBOX_KIND

A GadTools bevelbox. Use this to avoid the use of absolute sizing of bevelboxes. All bevel box kinds from OS3.0 is supported, even if the computer only has OS2.0.

Extra tags for BEVELBOX_KIND:

GUBB_Recessed (BOOL)
Create a recessed ("pushed in") bevel box. Differs from the GadTools tag GTBB_Recessed, in that it works with both TRUE and FALSE as parameter. GadTools creates a recessed box independent from the given value.

Defaults to FALSE.

GUBB_FrameType (ULONG)
Determines what kind of box this function renders. The current available alternatives are:

BFT_BUTTON - Generates a box like what is used around a GadTools BUTTON_KIND gadget.

BFT_RIDGE - Generates a box like what is used around a GadTools STRING_KIND gadget.

BFT_DROPBOX - Generates a box suitable for a standard icon drop box imagery.

BFT_HORIZBAR - Generates a horizontal shadowed line. Can also be used to draw a normal line, using 1 for the line's height.

or two Y flags together.

Shadow placement flags

~~~~~

BB\_SHADOW\_DR - Places the bevel box text shadow one pixel below and to the right of the text.

BB\_SHADOW\_UR - Places the bevel box text shadow one pixel above and to the right of the text.

BB\_SHADOW\_DL - Places the bevel box text shadow one pixel below and to the left of the text.

BB\_SHADOW\_UL - Places the bevel box text shadow one pixel above and to the left of the text.

BB\_SUNAT\_UL - Another name for BB\_SHADOW\_DR

BB\_SUNAT\_DL - Another name for BB\_SHADOW\_UR

BB\_SUNAT\_UR - Another name for BB\_SHADOW\_DL

BB\_SUNAT\_DR - Another name for BB\_SHADOW\_UL

Default if BB\_SHADOW\_DR (BB\_SUNAT\_UL).

GUBB\_3DText (BOOL)

Enables the shadow on the bevel box text. This tag must be used if GUBB\_ShadowColor or GUBB\_ShadowPen isn't used.

GUBB\_ShadowColor (ULONG)

Selects which color to print the shadow text in. Only useful for a bevelbox with a title. Defaults to the color of the SHADOWPEN.

GUBB\_ShadowPen (ULONG)

Selects which pen to print the shadow text in. Only useful for a bevelbox with a title. Defaults to SHADOWPEN. This tag overrides the GUBB\_ShadowColor tag.

PROGRESS\_KIND:

Gadget used to display a value out of a total. Can be used to display the progress of a search, a diskcopy or anything else.

Extra tags for PROGRESS\_KIND:

GUPR\_FillColor (ULONG)

Selects which color to use to paint the current value of the progress requester with. Defaults to the color of the FILLPEN.

GUPR\_FillPen (ULONG)

Selects which pen to use to paint the current value of the progress requester with. Defaults to the FILLPEN. This tag overrides the GUPR\_FillColor tag.

GUPR\_BackColor (ULONG)

Selects which color to fill the background of the progress requester with. Defaults to the color of the BACKGROUNDPEN.

**GUPR\_BackPen (ULONG)**

Selects which pen to fill the background of the progress requester with. Defaults to the BACKGROUNDPEN.  
This tag overrides the GUPR\_BackColor tag.

**GUPR\_Current (ULONG)**

The initial current value of the progress requester. The gadget's current value may be changed later by directly modifying the pg\_Current field of the ProgressGad structure. Use GU\_UpdateProgress to redraw the progress requester with the new value. The pg\_Current field must not be larger than 4.294.967.295 / the width of the progress gadget. A "normal" width of 410 pixels allows a current value of 10.737.418.

Defaults to 0.

**GUPR\_Total (ULONG)**

The initial total value of the progress requester. The gadget's total value may be changed later by directly modifying the pg\_Total field of the ProgressGad structure. Use GU\_UpdateProgress to redraw the progress requester with the new value. The total value can be as large as a longword allows (4.294.967.295), but you can't display a current value larger than 4.294.967.295 / the width of the progress gadget.

Defaults to 100.

Changed tags, and additions to GadTools:

**LISTVIEW\_KIND:****GTLV\_ShowSelected (UWORD id)**

This tag was changed, so that you don't have to create the string gadget before all other gadgets. The difference from this tag in GadTools, is that we now have to give the ID of the string gadget to use to show the selected item.

An example of a valid (and probably most useful) gadget to use for GTLV\_ShowSelected:

**ShowSelGad:**

```
dc.l  GU_GadgetKind,  STRING_KIND,  GU_AutoHeight,  4
dc.l  GU_DupeWidth,   GAD_LISTVIEW, GU_GadgetText,  NULL
dc.l  TAG_DONE
```

This gadget MUST be before the LISTVIEW gadget in the LayoutGadget array.

**Special:**

ti\_Data = -1 Creates a read-only gadget below the listview, same as for GTLV\_ShowSelected, 0 for GadTools.

ti\_Data = x Gadget ID for the gadget that the selected item should be displayed in. Same as GadTools reaction on a gadget pointer in ti\_Data.

This gadget's ti\_Data field will be changed during the creation

of the gadget, but will be changed back before `GU_LayoutGadgets` returns.

#### MX\_KIND:

The `gng_GadgetText` field in the `NewGadget` structure can be used even with `MX_KIND` gadgets. This should have been included in `GadTools`. The gadget text will always be placed ABOVE the gadget, on the same side as the other texts for the gadget. Positions are checked against `WBPattern` & `SerialPrefs` to get them "right". The `GU_GadgetText` and `GU_LocaleText` tags are used to access this field.

Tags for all gadget kinds:

Gadget width control:

`GU_Width` (UWORD wid)

Absolute width of the gadget. Not recommended to use for other gadgets than `IMAGE_KIND`, `DRAWER_KIND` and `FILE_KIND`.

`GU_DupeWidth` (UWORD id)

Duplicate the width of another gadget.

`GU_AutoWidth` (WORD add)

Width = length of text label + `ti_Data`. For `CYCLE_KIND` gadgets, the gadget width will be calculated by checking the length of all alternatives and using the one that is widest + 26 as width.

`GU_Columns` (UWORD numcols)

Set the gadget width so that approximately `ti_Data` columns of text will fit.

`GU_AddWidth` (WORD add)

Add `ti_Data` to the total width calculation.

`GU_MinWidth` (UWORD wid)

Make the gadget at least `ti_Data` pixels wide.

`GU_MaxWidth` (UWORD wid)

Make the gadget at most `ti_Data` pixels wide.

`GU_AddWidChar` (WORD chars)

Add the length of `ti_Data` characters to the total width calculation.

Gadget height control:

`GU_Height` (UWORD hei)

Absolute height of the gadget. Not recommended to use for other gadgets than `IMAGE_KIND`, `DRAWER_KIND` and `FILE_KIND`.

`GU_DupeHeight` (UWORD id)

Duplicate the height of another gadget.

`GU_AutoHeight` (WORD add)

Height = height of the gadget's font + `ti_Data`. This tag doesn't work as it should with `MX_KIND` gadgets. Will be fixed later. Use `GU_HeightFactor` or `GU_Height` for `MX_KIND` until this is fixed.

---

GU\_HeightFactor (UWORD numlines)  
Set the gadget height to approximately ti\_Data lines.

GU\_AddHeight (WORD add)  
Add ti\_Data to the total height calculation.

GU\_MinHeight (UWORD wid)  
Make the gadget at least ti\_Data pixels high.

GU\_MaxHeight (UWORD wid)  
Make the gadget at most ti\_Data pixels high.

GU\_AddHeiLines (WORD numlines)  
Add the height of ti\_Data/2 lines to the final height calculation.  
The numlines argument is given in units of 1/2 lines to get  
better resolution (ti\_Data of 4 means that the height of 2  
lines should be added).

#### Gadget top edge control:

GU\_Top, GU\_TopRel and GU\_AlignTop locks the top edge of the gadget, and allows any bottom edge control tag to adjust the height, so that both top and bottom edges will be correct.

GU\_Top (UWORD ypos)  
Absolute top edge of the gadget. Not recommended to use for other gadgets than the top-most gadgets.

GU\_TopRel (UWORD id)  
Make the top edge relative to another gadgets bottom edge. This gadget will be placed BELOW the given gadget.

GU\_AddTop (WORD add)  
Add ti\_Data to the final top edge calculation.

GU\_AlignTop (UWORD id)  
Align the top edge of the gadget with another gadgets top edge.

GU\_AdjustTop (WORD add)  
Add the height of the text font + ti\_Data to the top edge.

GU\_AddTopLines (WORD numlines)  
Add the height of ti\_Data/2 lines to the final top edge. The numlines argument is given in units of 1/2 lines to get better resolution (ti\_Data of 4 means that the height of 2 lines should be added).

#### Gadget bottom edge control:

GU\_Bottom, GU\_BottomRel and GU\_AlignBottom locks the bottom edge of the gadget, and allows any top edge control tag to adjust the height, so that both top and bottom edges will be correct.

GU\_Bottom (UWORD ypos)  
Absolute bottom edge of the gadget. Not recommended to use for other gadgets than the bottom-most gadgets. Should not be necessary to use

---

at all.

GU\_BottomRel (UWORD id)

Make the bottom edge relative to another gadgets top edge. This gadget will be placed ABOVE the given gadget.

GU\_AddBottom (WORD add)

Add ti\_Data to the final bottom edge calculation.

GU\_AlignBottom (UWORD id)

Align the bottom edge of the gadget with another gadgets bottom edge.

GU\_AdjustBottom (WORD add)

Subtract the height of the gadget's font + ti\_Data from the top edge. This will move the gadget UPWARDS (ti\_Data + font height) pixels.

Gadget left edge control:

GU\_Left, GU\_LeftRel and GU\_AlignLeft locks the left edge of the gadget, and allows any right edge control tag to adjust the width, so that both left and right edges will be correct.

GU\_Left (UWORD xpos)

Absoulute left edge of the gadget. Not recommended to use for other gadgets than the left-most gadgets.

GU\_LeftRel (UWORD id)

Make the left edge relative to another gadgets right edge. This gadget will be placed TO THE RIGHT of the given gadget.

GU\_AddLeft (WORD add)

Add ti\_Data to the final left edge calculation.

GU\_AlignLeft (UWORD id)

Align the left edge of the gadget with another gadgets left edge.

GU\_AdjustLeft (WORD add)

Add the width of the gadget label + ti\_Data to the left edge.

GU\_AddLeftChar (WORD chars)

Add the length of ti\_Data characters to the left edge.

Gadget right edge control:

GU\_Right, GU\_RightRel and GU\_AlignRight locks the right edge of the gadget, and allows any left edge control tag to adjust the width, so that both left and right edges will be correct.

GU\_Right (UWORD xpos)

Absoulute right edge of the gadget. Not recommended to use for other gadgets than the right-most gadgets. Should not be necessary to use at all.

GU\_RightRel (UWORD id)

Make the right edge relative to another gadgets left edge. This gadget will be placed TO THE LEFT of the given gadget.

---

GU\_AddRight (WORD add)  
Add ti\_Data to the final right edge calculation.

GU\_AlignRight (UWORD id)  
Align the right edge of the gadget with another gadgets right edge.

GU\_AdjustRight (WORD add)  
Add the width of the gadget label + ti\_Data to the left edge.

Other tags:

GU\_ToggleSelect (BOOL)  
Create a toggle select gadget. Works with BUTTON\_KIND and IMAGE\_KIND gadgets.

GU\_Selected (BOOL)  
Set the initial value of a toggle select gadget.

GU\_Hotkey (CHAR)  
Hotkey that should simulate a press (release) of a gadget.

GU\_HotkeyCase (BOOL)  
Make the hotkey case-sensitive. Default is not case sensitive.

GU\_LabelHotkey (BOOL)  
Get the hotkey directly from the gadget's label. The hotkey can be case-sensitive, but not for CYCLE, LISTVIEW and MX gadgets.

GU\_RawKey (BYTE)  
Use a rawkey as a gadget hotkey. May not be case-sensitive.

Tags that gives access to other fields in the NewGadget structure:

GU\_GadgetText (UBYTE \*)  
A pointer to the gadget's label. Will be copied directly into the gng\_GadgetText field of the NewGadget structure.

GU\_TextAttr (struct TextAttr \*)  
A pointer to an initialized TextAttr structure (to select the font). Will be copied directly into the gng\_TextAttr field of the NewGadget structure.

GU\_Flags (ULONG)  
Gadget flags. Currently available flags are as for GadTools, but here is a short list of them:

PLACETEXT\_LEFT - Place the gadget label right aligned on the left side of the gadget.

PLACETEXT\_RIGHT - Place the gadget label left aligned on the right side of the gadget.

PLACETEXT\_ABOVE - Place the gadget label centered above the gadget.

PLACETEXT\_BELOW - Place the gadget label centered below the gadget.

PLACETEXT\_IN - Place the gadget label centered inside the gadget.

---

NG\_HIGHLABEL - Highlight the label (render it using SHINEPEN).

GU\_UserData (APTR)

Storage for your own data. Will probably be removed, since GadUtil uses this field for an internal structure (with some external available fields).

GU\_LocaleText (ULONG stringid)

Get gadget label from a catalog. This allows easy localization of all new programs.

Tags that should be passed directly to GU\_LayoutGadgetsA():

GU\_RightExtreme (ULONG \*)

A pointer to a longword that is used to store the rightmost point that a gadget will exist in.

GU\_LowerExtreme (ULONG \*)

A pointer to a longword that is used to store the lowermost point that a gadget will exist in.

GU\_Catalog (struct Catalog \*)

A pointer to the programs translation catalog. NULL indicates that the program should use the internal strings. You must open the catalog by yourself (use  
GU\_OpenCatalog()  
or locale/OpenCatalog).

The GU\_AppStrings tag MUST be used together with this tag.

GU\_DefTextAttr (struct TextAttr \*)

Specifies the default font to use with all gadgets. Can be overridden with GU\_TextAttr tag for each gadget.

GU\_AppStrings (struct AppString \*)

A pointer to an array of AppString structures. These structures contains the programs internal strings.  
This tag must be used together with the GU\_Catalog tag.

GU\_BorderLeft (ULONG)

Size of the window's left border.

GU\_BorderTop (ULONG)

Size of the window's top border.

GU\_NoCreate (BOOL)

Don't create any gadgets. Useful to determine if the window will fit on the screen etc.

GU\_MinimumIDCMP (ULONG \*)

A pointer to a longword that is used to store the minimum required IDCMP flags, so that all gadgets will work. The longword can already be initialized, and any new needed IDCMP flags will be appended to that longword.

---

### 1.3 gadutil.library/GU\_FreeLayoutGadgets

NAME

GU\_FreeLayoutGadgets -- Frees gadgets laid out with  
GU\_LayoutGadgetsA()

.

SYNOPSIS

```
GU_FreeLayoutGadgets(gad_info)
                    A0
```

```
VOID GU_FreeLayoutGadgets(APTR);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

### 1.4 gadutil.library/GU\_CreateGadgetA

NAME

GU\_CreateGadgetA -- Create a gadget with built-in hotkey support.

SYNOPSIS

```
gad = CreateGadgetA(kind, prevgad, newgad, taglist)
D0,A0          D0    A0    A1    A2
```

```
struct Gadget *GU_CreateGadgetA(ULONG, struct Gadget *,
                                struct NewGadget *, struct TagItem *);
```

FUNCTION

GU\_CreateGadgetA() allocates and initializes a new gadget of the specified kind, and attaches it to the previous gadget. The gadget is created based on the supplied kind, NewGadget structure, and tags.

This function differs from the GadTools equivalent by supporting some extra tags to allow the gadget to be selected using the keyboard.

INPUTS

---

kind - kind of gadget to create. One of the XXX\_KIND values defined in <libraries/gadtools.h>.

prevgad - pointer to the previous gadget that this new gadget should be attached to. This function will fail if this value is NULL.

newgad - a filled in NewGadget structure describing the desired gadget's size, position, label, etc.

taglist - pointer to an array of tags providing optional extra parameters, or NULL.

#### TAGS

All kinds:

GT\_Underscore - Indicates the symbol that precedes the character in the gadget label to be underscored. This can be to indicate keyboard equivalents for gadgets. GadUtil has the ability to process the keyboard equivalents if some other tags are used.

GU\_Hotkey - This tag selects the hotkey to be used as an automatic keyboard command for the gadget. The ti\_Data field should contain the ASCII or RAWKEY code for the hotkey. This tag may be used together with the GU\_HotkeyCase and GU\_LabelHotkey. The GU\_RawKey tag requires this tag.

GU\_HotkeyCase - This tag makes the keyboard command case-sensitive. This tag may not be used with GU\_RawKey.

GU\_LabelHotkey - This tag picks the hotkey from the gadget text field in the NewGadget structure when the gadget is created. If no key is marked with the underscore, the GU\_HotKey code will be used.

This tag supports and requires the GT\_Underscore tag. The character that was given in the GT\_Underscore tag will be used to find the code for the hotkey.

GU\_RawKey - This tag makes the hotkey code to a RAWKEY code. All keys on the keyboard has one rawkey code that matches the key.

If your program also gets VANILLAKEY events, you must be careful when selecting the rawkey code to be used as a keyboard shortcut for this gadget. RAWKEY events will only be sent for special keys (such as the HELP, Control, Alt and function keys) that don't map to a single character. All keys that maps to a single character will be processed as a VANILLAKEY event.

This tag may not be used with (and will also disable) the GU\_HotkeyCase and GU\_LabelHotkey tags.

Kind specific tags:

See the GadTools function CreateGadgetA for all other tags.

---

## RESULT

gad - pointer to the new gadget, or NULL if the allocation failed or if prevgad was NULL.

## NOTES

Note that the ng\_VisualInfo and ng\_TextAttr fields of the NewGadget structure must be set to valid VisualInfo and TextAttr pointers, or this function will fail.

## SEE ALSO

```

    GU_FreeGadgets()
    ,
    GU_SetGadgetAttrsA()
    ,
    GU_GetVisualInfoA()
    ,
    GU_GetIMsg, gadtools/CreateGadgetA, <libraries/gadtools.h>

```

## 1.5 gadutil.library/GU\_SetGadgetAttrsA

## NAME

GU\_SetGadgetAttrsA -- Change the attributes of a GadTools gadget.

## SYNOPSIS

```

GU_SetGadgetAttrsA(gad, win, req, taglist)
    A0    A1  A2  A3

```

```

VOID GU_SetGadgetAttrsA(struct Gadget *, struct Window *,
    struct Requester *, struct TagItem *);

```

## FUNCTION

Change the attributes of the specified gadget, according to the attributes chosen in the tag list. If an attribute is not provided in the tag list, its value remains the unchanged. This function also stores some information for the hotkey part of the library.

Use this in place of the gadtools function GT\_SetGadgetAttrsA().

## INPUTS

gad - pointer to the gadget in question. This address may be NULL, in which case this function does nothing.

win - pointer to the window containing the gadget. Starting with V39 (of the OS), this value may be NULL, in which case the internal attributes of the gadgets are altered but no rendering occurs.

req - reserved for future use, should always be NULL.

taglist - pointer to an array of tags providing optional extra parameters, or NULL.

## TAGS

See the GadTools function `GT_SetGadgetAttrsA()` for all tags, since this is an extended version of that routine.

#### NOTES

This function may not be called inside of a `GU_BeginRefresh()` /

```

    GU_EndRefresh()
    session. (as always, restrict yourself to simple
rendering functions).
```

#### SEE ALSO

```

gadtools/GT_SetGadgetAttrsA(),
    GU_GetGadgetAttrsA()
```

## 1.6 gadutil.library/GU\_GetIMsg

#### NAME

`GU_GetIMsg` -- Get an `IntuiMessage`, process GadTools & Hotkey events.

#### SYNOPSIS

```

img = GU_GetIMsg(intuiport)
D0,A0,SR(Z)      A0
```

```

struct IntuiMessage *GU_GetIMsg(struct MsgPort *);
```

#### FUNCTION

Use `GU_GetIMsg()` in place of the usual `exec.library/GetMsg()` when reading `IntuiMessages` from your window's `UserPort`. If needed, the GadTools dispatcher will be invoked, and suitable processing will be done for gadget actions.

If the message is an `IDCMP_VANILLAKEY` or an `IDCMP_RAWKEY`, this routine will search through all gadgets for that key, and if it is found, the message will change to the type of message that gadget is supposed to send. If the key is not used as a hotkey, the message will not change.

If there are no messages (or if the only messages are meaningful only to GadTools/GadUtil), `NULL` will be returned.

#### INPUTS

`intuiport` - the `Window->UserPort` of a window that is using the GadUtil library.

#### RESULT

`img` - pointer to modified `IntuiMessage`, or `NULL` if there are no applicable messages.

`SR (Z)` - the zero flag will be set if there was no message. This is probably only useful for assembly language programmers.

#### NOTES

Be sure to use

```

    GU_ReplyIMsg()
    and not exec.library/ReplyMsg() on
messages obtained with GU_GetIMsg().
If you intend to do more with the resulting message than read its
```

fields, act on it, and reply it, you may find  
 GU\_FilterIMsg()  
 more appropriate.

Starting with V39 (of the OS), this function actually returns a pointer to an ExtIntuiMessage structure, but the prototype was not changed for source code compatibility with older software.

SEE ALSO

```
GU_ReplyIMsg()
'
GU_FilterIMsg()
```

## 1.7 gadutil.library/GU\_CountNodes

NAME

GU\_CountNodes -- Count number of nodes in a list.

SYNOPSIS

```
numnodes = GU_CountNodes(list)
D0                      A0
```

```
ULONG GU_CountNodes(struct List *);
```

FUNCTION

This function will count the number of nodes attached to a list.

INPUTS

list - a pointer to the list to get the number of nodes in

RESULT

numnodes - number of nodes that was in the list for the moment.

NOTES

Use Forbid() and Permit() around a call to this function if you are using it on a list that can change at any time (e.g. a list that wasn't created by yourself). This function may not be accurate when you are using it on a system list.

## 1.8 gadutil.library/GU\_GadgetArrayIndex

NAME

GU\_GadgetArrayIndex -- Get a gadget's index in the LayoutGadget array.

SYNOPSIS

```
index = GU_GadgetArrayIndex(id, gadgets)
D0,D1,SR(Z)                      D0  A0
```

```
WORD GU_GadgetArrayIndex(WORD, struct LayoutGadget *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.9 gadutil.library/GU\_BlockInput

NAME

GU\_BlockInput -- Block all input to a window.

SYNOPSIS

```
GU_BlockInput(window)
    A0
```

```
VOID GU_BlockInput(struct Window *);
```

FUNCTION

Changes the window's pointer to the standard wait pointer (OS 2.0) or the preferred wait pointer (OS 3.0+) and opens a requester to block the user input of the given window. The requester that is opened is not visible.

INPUTS

window - the parent window for the requester. This is the window that will be blocked for user input.

EXAMPLE

```
BlockInput(myWin);
About();
FreeInput();
```

Will block the parent window for user input while displaying the About requester of a program.

SEE ALSO

GU\_FreeInput()

## 1.10 gadutil.library/GU\_FreeInput

---

## NAME

GU\_FreeInput -- Unblock input to a blocked window.

## SYNOPSIS

```
GU_FreeInput(window)
           A0
```

```
VOID GU_FreeInput(struct Window *);
```

## FUNCTION

Unblock a window's user input. Call this function after blocking the input with

```
GU_BlockInput()
```

.

## INPUTS

window - the window that was blocked with

```
GU_BlockInput()
```

.

## EXAMPLE

```
BlockInput(myWin);
About();
FreeInput();
```

Will block the parent window for user input while displaying the About requester of a program.

## SEE ALSO

```
GU_BlockInput()
```

## 1.11 gadutil.library/GU\_FreeGadgets

## NAME

GU\_FreeGadgets -- Free a linked list of gadgets.

## SYNOPSIS

```
GU_FreeGadgets(glist)
           A0
```

```
VOID GU_FreeGadgets(struct Gadget *);
```

## FUNCTION

Frees all gadgets found on the linked list of gadgets beginning with the specified one. Frees all the memory that was allocated by

```
GU_CreateGadgetA()
```

. This function will return safely with no action if it receives a NULL parameter.

Use this function in place of gadtools/FreeGadgets().

---

INPUTS  
glist - pointer to the first gadget in list to be freed

SEE ALSO

GU\_CreateGadgetA()

## 1.12 gadutil.library/GU\_SetGUGadAttrsA

NAME  
GU\_SetGUGadAttrsA -- Change the attributes of a GadUtil gadget.

SYNOPSIS  
GU\_SetGUGadAttrsA(gad\_info, gad, win, taglist)  
                  A0          A1   A2   A3

VOID GU\_SetGadAttrsA(APTR, struct Gadget \*, struct Window \*,  
          struct TagItem \*);

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.13 gadutil.library/GU\_CoordsInGadBox

NAME  
GU\_CoordsInGadBox -- Check if a coordinate pair is within a gadget.

SYNOPSIS  
IsInBox = GU\_CoordsInGadBox(coords, gad)  
D0, SR(Z)                  D0      A0

BOOL GU\_CoordsInGadBox(ULONG, struct Gadget \*);

FUNCTION

Check if a coordinate pair is within a gadget's border. This function may be used to make coordinate sensitive AppWindows (allows the user to drop a file on a string gadget etc.).

---

To use this function, you must save the coordinates from the received message (AppMessage, IntuiMessage) to have something to compare against.

#### INPUTS

coords - a combined LONG of both the X and Y coordinates to compare against. The X coordinate should be in the upper word of the parameter.

gad - the gadget to check the coordinates against.

#### RESULT

IsInBox - TRUE if both given coordinates was within the gadget's outer box (X coord is between gadx and gadx+gadw, Y coord is between gady and gady+gadh). Otherwise this function will return FALSE.

#### EXAMPLES

Assembly language:

```

move.l  am_MouseX(a0),d0  ; Get X and Y coordinates
move.l  mystrgad(pc),a0
move.l  GadUtilBase(pc),a6
jsr  _LVOGU_CoordsInGadBox(a6)
beq.b  .notinbox  ; Not in gadget box

; Do what you want to do if the coordinates are
; within the gadget box
notinbox:
; Here, you may want to check for some other gadgets

```

C:

```

long coords;
coords = (LONG)appmsg->MouseX << 16 | appmsg->MouseY;
if (CoordsInGadBox(coords,mystrgad) = TRUE)
{
    /* Do what you want to do if the coordinates
    are within the gadget box */
}
else
{
    /* Here, you may want to check for some other
    gadgets */
}

```

## 1.14 gadutil.library/GU\_GetGadgetPtr

#### NAME

GU\_GetGadgetPtr -- Get a pointer to a gadget's gadget structure.

#### SYNOPSIS

```

gad = GU_GetGadgetPtr(id, gadgets)
D0,SR(Z)                D0  A0

```

```

struct Gadget *GU_GetGadgetPtr(UWORD, struct LayoutGadget *);

```

## FUNCTION

Find a gadget's gadget structure by giving its ID. The gadget pointer is always found last in the LayoutGadget structure. You can use this function to get that pointer.

It is also possible to get the gadget structure by taking it directly from the LayoutGadget structure array, but then you must know exactly in which structure it is located. Assembly language programmers can use a PC relative pointer to get the gadget pointer.

## INPUTS

id - the ID of the gadget to search for

gadgets - a pointer to the array of LayoutGadget structures.

## RESULT

gad - pointer to the requested gadget. For bevelboxes, this function will return a BBoxData structure.

## EXAMPLE

Some of the LayoutGadget structures from BetterTest.c:

```
struct LayoutGadget gadgets[] = {
{ MSG_NEXTDRIVE, NextDriveGad, StdGTTags,  NULL },
{ MSG_PREVDRIVE, PrevDriveGad, StdGTTags,  NULL },
{ MSG_DRIVE,    DriveGad,      DriveGTTags, NULL },
{ MSG_REQUESTER, ReqGad,      StdGTTags,  NULL },
{ MSG_CHECKME,  CheckBoxGad,  StdGTTags,  NULL },
{ MSG_FILENAME, FileNameGad,  StdGTTags,  NULL },
{ -1,          NULL,          NULL,        NULL }
};
```

The examples should get the gadget structure of the Requester gadget (not assuming that ID's begin with 0).

There is two methods you can use to get a gadgets structure:

1. Use the pointer in the array directly:

```
thegadget = gadgets[3].lg_Gadget
```

This will only work if you know in which LayoutGadget structure the gadget is in.

2. Use this library function:

```
thegadget = GU_GetGadgetPtr(MSG_REQUESTER, gadgets);
```

This will always work if all gadgets have a unique ID.

Some of the LayoutGadget structures from BetterTest.s:

```
gadgets:
GADGET MSG_NEXTDRIVE, NextDriveGad, StdGTTags
GADGET MSG_PREVDRIVE, PrevDriveGad, StdGTTags
```

```
GADGET MSG_DRIVE,      DriveGad,      DriveGTags
GADGET MSG_REQUESTER, ReqGad,        StdGTags
GADGET MSG_CHECKME,   CheckBoxGad, StdGTags
GADGET MSG_FILENAME,  FileNameGad,  StdGTags
GADGET -1,NULL,NULL
```

Assembly language programmers can use three methods to get the pointer:

1. Use the pointer in the array directly:

```
lea.l gadgets(pc),a0 ; Get array
moveq.l #lg_SIZEOF,d0 ; Get the size of the LayoutGadget
mulu #3,d0 ; struct and calculate offset to
; the right structure in the array
move.l lg_Gadget(a0,d0.l),d0 ; Get the gadget pointer
```

This will only work if you know in which LayoutGadget structure the gadget is in.

2. Use this library function:

```
lea.l gadgets(pc),a0 ; Get array
moveq.l #MSG_REQUESTER,d0 ; Gadget to search for
jsr _LVOGU_GetGadgetPtr ; Get gadget, result in D0
```

This will always work if all gadgets have a unique ID.

3. Use an extra label for each gadget that should be easy to access:

```
gadgets:
GADGET MSG_NEXTDRIVE, NextDriveGad, StdGTags
GADGET MSG_PREVDRIVE, PrevDriveGad, StdGTags
GADGET MSG_DRIVE,     DriveGad,     DriveGTags
GADGET MSG_REQUESTER, ReqGad,      StdGTags
reqgad: equ *-4
GADGET MSG_CHECKME,   CheckBoxGad, StdGTags
GADGET MSG_FILENAME,  FileNameGad,  StdGTags
filenamegad: equ *-4
GADGET -1,NULL,NULL
```

```
move.l reqgad(pc),d0 ; Get the gadget
```

This will always work.

## 1.15 gadutil.library/GU\_TextWidth

NAME

GU\_TextWidth -- Calculate the pixel length of a text string.

SYNOPSIS

```
textwidth = GU_TextWidth(string, textattr)
```

```
D0                                A0      A1
```

```
ULONG GU_TextWidth(STRPTR, struct TextAttr *);
```

#### FUNCTION

Calculate the length of the text, using the specified font. This function will open the required font, if it isn't opened before.

#### INPUTS

string - NULL terminated text to calculate width of.

textattr - a filled in TextAttr structure. Only the IText and ITextFont fields of the structure have to be filled in.

#### RESULT

textwidth - pixel length of the text

## 1.16 gadutil.library/GU\_GetLocaleStr

#### NAME

GU\_GetLocaleStr -- Get a localized string from a catalog.

#### SYNOPSIS

```
string = GU_GetLocaleStr(stringID, catalog, defstrings)
D0,A0                D0        A0        A1
```

```
STRPTR GU_GetLocaleStr(ULONG, struct Catalog *, struct AppString *);
```

#### FUNCTION

Get a localized string, or the default string, from a catalog or from the programs built-in strings.

#### INPUTS

stringID - the ID of the string to get

catalog - the opened catalog for the program

defstrings - an array of AppString structures, the programs built-in strings and ID's.

#### RESULT

string - the address of the localized string, or if the catalog was not available (or if the selected language was the programs built-ins), a pointer to the default string with the given ID.

#### SEE ALSO

```
GU_OpenCatalog()
,
GU_CloseCatalog()
```

## 1.17 gadutil.library/GU\_CreateLocMenuA

## NAME

GU\_CreateLocMenuA -- Create a menu with localized items.

## SYNOPSIS

```
menu = GU_CreateLocMenuA(newmenu, gad_info, createtags, layouttags)
D0                               A0           A1           A2           A3
```

```
struct Menu *GU_GetLocaleStr(struct NewMenu *, APTR,
                             struct TagItem *, struct TagItem *);
```

## FUNCTION

Create and layout a localized menu. This function replaces both the gadtools/CreateMenusA and gadtools/LayoutMenusA functions. See those functions for a more in-depth description of this function.

## INPUTS

newmenu - pointer to an array of initialized struct NewMenus. This differs from the GadTools function in that, instead of giving pointers to strings in the nm\_Label and nm\_CommKey fields of the structure, you should put a string ID in the nm\_Label field of the structure. The string must be in the format "A\x00About...". The nm\_CommKey field should be left empty. ^^^^^^

```
| |
| | - nm_Label field for the NewMenu structure
|
```

```
| - nm_CommKey replacement. If no keyboard shortcut,
  this field MUST CONTAIN A SPACE CHAR.
  The \x00 is a NULL character.
```

gad\_info - the value returned from GU\_LayoutGadgetsA

createtags - tags for the "CreateMenusA" part of this routine. All gadtools tags are supported (directly passed to GT).

layouttags - tags for the "LayoutMenusA" part of this routine. All gadtools tags are supported (directly passed to GT).

## TAGS

See the gadtools functions CreateMenusA and LayoutMenusA.

## RESULT

menu - pointer to the resulting initialized and laid out menu structure, ready to pass to the intuition function SetMenuStrip, or NULL for failure.

## SEE ALSO

GU\_FreeMenusA(), gadtools/CreateMenusA(), gadtools/LayoutMenusA(), gadtools/FreeMenus()

## 1.18 gadutil.library/GU\_OpenCatalog

## NAME

GU\_OpenCatalog -- Open a message catalog.

## SYNOPSIS

```
catalog = GU_OpenCatalog(name, version)
D0                A0        D0
```

```
struct Catalog *GU_OpenCatalog(STRPTR, ULONG);
```

## FUNCTION

This function opens a message catalog. Catalogs contain all the text strings that an application uses. These strings can easily be replaced by strings in a different language, which causes the application to magically start operation in that new language.

Catalogs originally come from disk files. This function searches for them in the following places:

```
PROGDIR:Catalogs/languageName/name
LOCALE:Catalogs/languageName/name
```

where languageName is the name of the language associated with the locale parameter.

## INPUTS

catalogname - the NULL terminated name of the catalog to open (just the name, not the complete path to it).

version - required version of the catalog to open. Passign 0 as version number means that the program will accept any found version of the catalog. Other values than 0 means exactly that version.

## RESULT

catalog - A message catalog to use with GU\_GetLocaleStr or any of the Locale library functions or NULL. NULL is returned on error or if the application can use its built-in strings instead of loading a catalog from disk.

## EXAMPLE

```
GU_OpenCatalog("myprogram.catalog",0);
```

will open any version of the catalog file "myprogram.catalog" found in either PROGDIR:Catalogs/languageName/ (where the program was started from), or LOCALE:Catalogs/languageName/.

```
GU_OpenCatalog("myprogram.catalog",5);
```

will open version 5 of the catalog file. If v5 is not available, the program will use its internal strings.

## NOTES

If you want to specify other tags than the version tag, you must use the Locale library OpenCatalog(). This function is generally a shortcut to that function. By using this routine, you may not need to open Locale library at all.

This routine assumes that the built-in language of the program is english. If you write your programs in another language, you must open the catalog by yourself.

---

SEE ALSO

```
GU_CloseCatalog()  
, locale/OpenCatalog()
```

## 1.19 gadutil.library/GU\_CloseCatalog

NAME

GU\_CloseCatalog -- Close a message catalog.

SYNOPSIS

```
GU_CloseCatalog(catalog)  
                A0
```

```
VOID GU_CloseCatalog(struct Catalog *);
```

FUNCTION

Concludes access to a message catalog. The usage count of the catalog is decremented. When this count reaches 0, the catalog can be expunged from system memory whenever a memory panic occurs.

INPUTS

catalog - the message catalog to close. A NULL catalog is a valid parameter and is simply ignored.

NOTES

This function is a shortcut to the locale/CloseCatalog() function.

SEE ALSO

```
GU_OpenCatalog()  
,  
GU_GetLocaleStr()
```

## 1.20 gadutil.library/GU\_DisableGadget

NAME

GU\_DisableGadget -- Disable / Enable a gadget.

SYNOPSIS

```
GU_DisableGadget(status, gadget, window)  
                D0,      A0,   A1
```

```
VOID GU_DisableGadget(BOOL, struct Gadget *, struct Window *);
```

FUNCTION

INPUTS

TAGS

---

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.21 gadutil.library/GU\_SetToggle

NAME

GU\_SetToggle -- Change status of a toggle-select gadget.

SYNOPSIS

```
GU_SetToggle(status, gadget, window)
              D0,      A0,    A1
```

```
VOID GU_SetToggle(BOOL, struct Gadget *, struct Window *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.22 gadutil.library/GU\_RefreshBoxes

NAME

GU\_RefreshBoxes -- Redraw all bevel boxes in a window.

SYNOPSIS

```
GU_RefreshBoxes(window, gad_info)
                A0      A1
```

```
VOID GU_RefreshBoxes(struct Window *, APTR);
```

---

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.23 gadutil.library/GU\_RefreshWindow

NAME

GU\_RefreshWindow -- Redraw bevel boxes and gadgets in a window.

SYNOPSIS

```
GU_RefreshWindow(window, gad_info)
                  A0      A1
```

```
VOID GU_RefreshWindow(struct Window *, APTR);
```

FUNCTION

Perform the initial refresh of all the GadTools gadgets you have created. After you have opened your window, you must call this function. Or, if you have opened your window without gadgets, you add the gadgets with `intuition/AddGList()`, refresh them using `intuition/RefreshGList()`, then call this function. You should not need this function at other times.

This function differs from the `gadtools/GT_RefreshWindow()`, in that it also renders all bevelbox kind gadgets. If `NULL` is given in `gad_info`, no boxes will be rendered, and this function will work exactly as the `GT_RefreshWindow()`.

INPUTS

`window` - pointer to the window containing GadTools gadgets.

`gad_info` - the value returned from `GU_LayoutGadgetsA()`, or `NULL`.

SEE ALSO

`GU_BeginRefresh()`

---

## 1.24 gadutil.library/GU\_OpenFont

### NAME

GU\_OpenFont -- Load and get a pointer to a disk or system font.

### SYNOPSIS

```
font = GU_OpenFont(textAttr)
```

```
D0          A0
```

```
struct TextFont *GU_OpenFont(struct TextAttr *);
```

### FUNCTION

### INPUTS

### TAGS

### RESULT

### EXAMPLE

### NOTES

### WARNING

### BUGS

### SEE ALSO

## 1.25 gadutil.library/GU\_NewList

### NAME

GU\_NewList -- Initialize a list header for use.

### SYNOPSIS

```
GU_NewList(list)
```

```
A0
```

```
VOID GU_NewList(struct List *);
```

### FUNCTION

### INPUTS

### TAGS

### RESULT

### EXAMPLE

### NOTES

### WARNING

BUGS

SEE ALSO

## 1.26 gadutil.library/GU\_ClearList

NAME

GU\_ClearList -- Clear a listview gadget and deallocate all its nodes.

SYNOPSIS

```
GU_ClearList(gad, win, list)
             D0  A0  A1
```

```
VOID GU_ClearList(struct Gadget *, struct Window *, struct List *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.27 gadutil.library/GU\_DetachList

NAME

GU\_DetachList -- Disconnect the list from a listview gadget.

SYNOPSIS

```
GU_DetachList(gad, win)
             D0  A0
```

```
VOID GU_DetachList(struct Gadget *, struct Window *);
```

FUNCTION

INPUTS

TAGS

RESULT

---

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.28 gadutil.library/GU\_AttachList

NAME

GU\_AttachList -- Change a listview's current list.

SYNOPSIS

```
GU_AttachList(gad, win, list)
               D0   A0   A1
```

```
VOID GU_AttachList(struct Gadget *, struct Window *, struct List *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.29 gadutil.library/GU\_AddTail

NAME

GU\_AddTail -- Add a node to the end of a listview's list.

SYNOPSIS

```
node = GU_AddTail(gad, string, list)
D0           D0   A0   A1
```

```
struct Node *GU_AddTail(struct Gadget *, STRPTR, struct List *);
```

FUNCTION

---

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

### 1.30 gadutil.library/GU\_ChangeStr

NAME

GU\_ChangeStr -- Change the contents of string gadget.

SYNOPSIS

```
GU_ChangeStr(gad, string, win)
              D0   A0     A1
```

```
VOID GU_ChangeStr(struct Gadget *, struct Window *, STRPTR);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

### 1.31 gadutil.library/GU\_CreateContext

NAME

GU\_CreateContext -- Create a space for GadTools context data.

SYNOPSIS

```
gad = GU_CreateContext(glistptr)
```



```
, gadtools/GT_GetGadgetAttrsA()
```

### 1.33 gadutil.library/GU\_CreateMenuA

NAME

GU\_CreateMenuA -- Allocate and fill out a menu structure.

SYNOPSIS

```
menu = GU_CreateMenuA(newmenu, taglist)
D0                A0        A1
```

```
struct Menu *GU_CreateMenuA(struct NewMenu *, struct TagItem *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_LayoutMenuA()
,
GU_FreeMenu()
, gadtools/CreateMenuA()
```

### 1.34 gadutil.library/GU\_FreeMenu

NAME

GU\_FreeMenu -- Frees memory allocated by  
GU\_CreateMenuA()

.

SYNOPSIS

```
GU_FreeMenu(menu)
A0
```

```
VOID GU_FreeMenu(struct Menu *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_CreateMenusA()  
, gadtools/FreeMenus()
```

### 1.35 gadutil.library/GU\_LayoutMenuItemsA

NAME

GU\_LayoutMenuItemsA -- Position all the menu items.

SYNOPSIS

```
success = GU_LayoutMenuItemsA(menuitem, vi, tags)  
D0          A0          A1  A2
```

```
BOOL GU_LayoutMenuItemsA(struct MenuItem *, APTR, struct TagItem *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_CreateMenusA()  
,  
GU_GetVisualInfoA()  
, gadtools/LayoutMenuItemsA()
```

## 1.36 gadutil.library/GU\_LayoutMenusA

NAME

GU\_LayoutMenusA -- Position all the menus and menu items.

SYNOPSIS

```
success = GU_LayoutMenusA(menu, vi, taglist)
```

```
D0                A0    A1  A2
```

```
BOOL GU_LayoutMenusA(struct Menu *, APTR, struct TagItem *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_CreateMenusA()  
,  
GU_GetVisualInfoA()  
, gadtools/LayoutMenusA()
```

## 1.37 gadutil.library/GU\_GetVisualInfoA

NAME

GU\_GetVisualInfoA -- Get information GadTools needs for visuals.

SYNOPSIS

```
vi = GU_GetVisualInfoA(screen, taglist)
```

```
D0                A0    A1
```

```
APTR GU_GetVisualInfoA(struct Screen *, struct TagItem *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

---

NOTES

WARNING

BUGS

SEE ALSO

```
GU_FreeVisualInfo()  
    , gadtools/FreeVisualInfo(), intuition(/LockPubScreen(),  
intuition/UnlockPubScreen())
```

### 1.38 gadutil.library/GU\_FreeVisualInfo

NAME

GU\_FreeVisualInfo -- Return any resources taken by GU\_GetVisualInfoA

SYNOPSIS

```
GU_FreeVisualInfo(vi)  
    A0
```

```
VOID GU_FreeVisualInfo(APTR);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_GetVisualInfoA()  
    , gadtools/FreeVisualInfo()
```

### 1.39 gadutil.library/GU\_BeginRefresh

NAME

GU\_BeginRefresh -- Begin refreshing friendly to GadTools.

SYNOPSIS

```
GU_BeginRefresh(win)
```

A0

```
VOID GU_BeginRefresh(struct Window *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_EndRefresh()  
, gadtools/GT_BeginRefresh, intuition/BeginRefresh()
```

## 1.40 gadutil.library/GU\_EndRefresh

NAME

GU\_EndRefresh -- End refreshing friendly to GadTools.

SYNOPSIS

```
GU_EndRefresh(win, complete)  
A0 D0
```

```
VOID GU_EndRefresh(struct Window *, BOOL);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_BeginRefresh()
```

---

```
, gadtools/GT_EndRefresh(), intuition/EndRefresh()
```

## 1.41 gadutil.library/GU\_FilterIMsg

NAME

GU\_FilterIMsg -- Filter an IntuiMessage through GadTools.

SYNOPSIS

```
modimsg = GU_FilterIMsg(imsg)
D0          A1
```

```
struct IntuiMessage *GU_FilterIMsg(struct IntuiMessage *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_GetIMsg()
,
GU_PostFilterIMsg()
, gadtools/GT_FilterIMsg()
```

## 1.42 gadutil.library/GU\_PostFilterIMsg

NAME

GU\_PostFilterIMsg -- Return the unfiltered message after

```
GU_FilterIMsg()
was called, and clean up.
```

SYNOPSIS

```
imsg = GU_PostFilterIMsg(modimsg)
D0          A1
```

```
struct IntuiMessage *GU_PostFilterIMsg(struct IntuiMessage *);
```

FUNCTION

---

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_FilterIMsg()
, gadtools/GT_PostFilterIMsg()
```

## 1.43 gadutil.library/GU\_ReplyIMsg

NAME

GU\_ReplyIMsg -- Reply a message obtained with  
GU\_GetIMsg()

.

SYNOPSIS

```
GU_ReplyIMsg(imsg)
A1
```

```
VOID GU_ReplyIMsg(struct IntuiMessage *);
```

FUNCTION

Return a modified IntuiMessage obtained with  
GU\_GetIMsg()

. If you

use

```
GU_GetIMsg()
```

, use this function where you would normally have used exec/ReplyIMsg() or gadtools/GT\_ReplyIMsg(). You may safely call this function with a NULL pointer (nothing will be done).

INPUTS

imsg - a modified IntuiMessage obtained with GT\_GetIMsg(), or NULL in which case this function does nothing.

NOTES

When using GadUtil, you MUST explicitly GU\_ReplyIMsg() all messages you receive. You cannot depend on CloseWindow() to handle messages you have not replied.

Starting with V39, this function actually expects a pointer to an ExtIntuiMessage structure, but the prototype was not changed for source code compatibility with older software.

SEE ALSO

GU\_GetIMsg()

## 1.44 gadutil.library/GU\_DrawBevelBoxA

NAME

GU\_DrawBevelBoxA -- Draw a bevelled box.

SYNOPSIS

```
GU_DrawBevelBoxA(rport, left, top, width, height, taglist)
                  A0      D0      D1      D2      D3      A1
```

```
VOID GU_DrawBevelBoxA(struct RastPort *, WORD, WORD, WORD, WORD,
                      struct TagItem *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

```
GU_GetVisualInfoA()
, gadtools/DrawBevelBoxA()
```

## 1.45 gadutil.library/GU\_FindNode

NAME

GU\_FindNode -- Find the node structure of a given node number

SYNOPSIS

```
node = GU_FindNode(list, number)
D0      A0      D0
```

```
struct Node *GU_FindNode(struct List *, WORD);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.46 gadutil.library/GU\_NodeUp

NAME

GU\_NodeUp -- Move a node one step towards the top of the list

SYNOPSIS

```
success = GU_NodeUp(node, list)
D0                A0    A1
```

```
BOOL GU_NodeUp(struct Node *, struct List *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.47 gadutil.library/GU\_NodeDown

NAME

GU\_NodeDown -- Move a node one step towards the end of the list

SYNOPSIS

```
success = GU_NodeDown(node, list)
D0                A0    A1
```

```
BOOL GU_NodeDown(struct Node *, struct List *);
```

FUNCTION

INPUTS

TAGS

RESULT

EXAMPLE

NOTES

WARNING

BUGS

SEE ALSO

## 1.48 gadutil.library/GU\_UpdateProgress

NAME

GU\_UpdateProgress -- Redraw all or specified progress gadget(s).

SYNOPSIS

```
GU_UpdateProgress(window, gad_info, gadget)
                   A0      A1      A2
```

```
VOID GU_UpdateProgress(struct Window *, APTR, struct ProgressGad *);
```

FUNCTION

Redraws all or one specified PROGRESS\_KIND gadget in a window.

INPUTS

window - the window that the progress gadget is in.

gad\_info - the value returned from

```
GU_LayoutGadgetsA()
```

gadget - NULL, or a pointer to a specified progress gadget ↔  
to

redraw. If NULL is given, all progress gadgets will be redrawn. Use this function to redraw the progress gad after changing the total or current value.

This function is called automatically from GU\_RefreshBoxes.

SEE ALSO

```
GU_RefreshBoxes()
```

```
,
```

```
GU_RefreshWindow()
```