BigDoc

COLLABORATORS			
	TITLE : BigDoc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 1, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

BigDoc

1.1 gadutil.library

gadutil.library/GU_LayoutGadgetsA()

gadutil.library/GU_FreeLayoutGadgets()

gadutil.library/GU_CreateGadgetA()

gadutil.library/GU_SetGadgetAttrsA()

gadutil.library/GU_GetIMsg()

gadutil.library/GU_CountNodes()

gadutil.library/GU_GadgetArrayIndex()

gadutil.library/GU_BlockInput()

gadutil.library/GU_FreeInput()

gadutil.library/GU_FreeGadgets()

gadutil.library/GU_SetGUGadAttrsA()

gadutil.library/GU_CoordsInGadBox()

gadutil.library/GU_GetGadgetPtr()

gadutil.library/GU_TextWidth()

gadutil.library/GU_GetLocaleStr()

gadutil.library/GU_CreateLocMenuA()

gadutil.library/GU_OpenCatalog()

gadutil.library/GU_CloseCatalog()

gadutil.library/GU_DisableGadget()

gadutil.library/GU_SetToggle()

gadutil.library/GU_RefreshBoxes()

gadutil.library/GU_RefreshWindow()

gadutil.library/GU_OpenFont()

gadutil.library/GU_NewList()

gadutil.library/GU_ClearList()

gadutil.library/GU_DetachList()

gadutil.library/GU_AttachList()

gadutil.library/GU_AddTail()

gadutil.library/GU_ChangeStr()

gadutil.library/GU_CreateContext()

gadutil.library/GU_GetGadgetAttrsA()

gadutil.library/GU_CreateMenusA()

gadutil.library/GU_FreeMenus()

gadutil.library/GU_LayoutMenuItemsA()

gadutil.library/GU_LayoutMenusA()

gadutil.library/GU_GetVisualInfoA()

gadutil.library/GU_FreeVisualInfo()

gadutil.library/GU_BeginRefresh()

gadutil.library/GU_EndRefresh()

gadutil.library/GU_FilterIMsg()

gadutil.library/GU_PostFilterIMsg()

gadutil.library/GU_ReplyIMsg()

gadutil.library/GU_DrawBevelBoxA()

gadutil.library/GU_FindNode()

gadutil.library/GU_NodeUp()

gadutil.library/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 layuot 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 sequentally 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 if, 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.

```
                BFT_VERTBAR - Generates a vertical shadowed line. Can
                              also be used to draw a normal line, using
                              1 for the line's width.

            Defaults to BFT_BUTTON.

        GUBB_TextColor (ULONG)
            Selects which color to print the title text in. Only useful
            for a bevelbox with a title. Defaults to the color of the
            TEXTPEN.

        GUBB_TextPen (ULONG)
            Selects which pen to print the title text in. Only
            useful for a bevelbox with a title. Defaults to TEXTPEN.
            This tag overrides the GUBB_TextColor tag.

        GUBB_Flags (ULONG)
            Currently, only text placement flags are defined. These are:

            Y-pos flags
            ~~~~~~~~~~~
                BB_TEXT_ABOVE - Places the bevel box text above the upper
                                border of the box          ___Example___

                BB_TEXT_IN    - Places the bevel box text at the upper
                                border of the box          ---Example---

                BB_TEXT_BELOW - Places the bevel box text below the upper
                                border of the box          ___      ___
                                                              Example
            X-pos flags
            ~~~~~~~~~~~
                BB_TEXT_CENTER - Places the bevel box text in the middle
                                 of the upper border       ---Example---

                BB_TEXT_LEFT   - Places the bevel box text 8 pixels from
                                 the left edge of the box  -Example-----

                BB_TEXT_RIGHT  - Places the bevel box text 8 pixels from
                                 the right edge of the box -----Example-

            Default if BB_TEXT_ABOVE!BB_TEXT_CENTER. Combine the x and y
            position flags by OR:ing them together. Don't combine two X
            or two Y flags together.

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. This tag doesn't work
        as it should with CYCLE_KIND and MX_KIND gadgets. Will be fixed
        later. Use GU_Columns for those kinds until this is fixed.

    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 filed 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 woll 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 over-
        ridden with GU_TextAttr tag for each gadget.

    GU_AppStrings (struct AppString *)
        A pointer to an array of AppString structures. These strutures
        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.
```

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_BeginRefesh() /

              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
imsg = 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
imsg - 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 recieves 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
recieved 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,       DriveGTTags
 GADGET  MSG_REQUESTER,  ReqGad,         StdGTTags
 GADGET  MSG_CHECKME,    CheckBoxGad,    StdGTTags
 GADGET  MSG_FILENAME,   FileNameGad,    StdGTTags
 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,   StdGTTags
 GADGET  MSG_PREVDRIVE,  PrevDriveGad,   StdGTTags
 GADGET  MSG_DRIVE,      DriveGad,       DriveGTTags
 GADGET  MSG_REQUESTER,  ReqGad,         StdGTTags
reqgad:   equ *-4
 GADGET  MSG_CHECKME,    CheckBoxGad,    StdGTTags
 GADGET  MSG_FILENAME,   FileNameGad,    StdGTTags
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
is 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)
D0                     A0

struct Gadget *GU_CreateContext(struct Gadget **);

 FUNCTION

 INPUTS

 TAGS

 RESULT

 EXAMPLE

 NOTES

 WARNING

 BUGS

 SEE ALSO
gadtools/CreateContext()
```

1.32 gadutil.library/GU_GetGadgetAttrsA()

```
                 NAME
GU_GetGadgetAttrsA -- Request the attributes of a GadTools gadget.

 SYNOPSIS
numProcessed = GU_GetGadgetAttrsA(gad, win, req, taglist)
D0                                A0   A1   A2   A3

LONG *GU_GetGadgetAttrsA(struct Gadget *, struct Window *,
       struct Requester *, struct TagItem *);

 FUNCTION
```

```
 INPUTS

 TAGS

 RESULT

 EXAMPLE

 NOTES

 WARNING

 BUGS

 SEE ALSO

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

1.33 gadutil.library/GU_CreateMenusA()

```
                  NAME
GU_CreateMenusA -- Allocate and fill out a menu structure.

 SYNOPSIS
menu = GU_CreateMenusA(newmenu, taglist)
D0                     A0       A1

struct Menu *GU_CreateMenusA(struct NewMenu *, struct TagItem *);

 FUNCTION

 INPUTS

 TAGS

 RESULT

 EXAMPLE

 NOTES

 WARNING

 BUGS

 SEE ALSO

              GU_LayoutMenusA()
              ,
              GU_FreeMenus()
              , gadtools/CreateMenusA()
```

1.34 gadutil.library/GU_FreeMenus()

```
                  NAME
GU_FreeMenus -- Frees memory allocated by
              GU_CreateMenusA()
              .

 SYNOPSIS
GU_FreeMenus(menu)
             A0

VOID GU_FreeMenus(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
```