GadUtil

COLLABORATORS			
	TITLE : GadUtil		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 23, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GadUtil

1.1 GadUtil.doc

GU_AddTail()

GU_AttachList()

GU_BeginRefresh()

GU_BlockInput()

GU_ChangeStr()

GU_CheckVersion()

GU_ClearList()

GU_ClearWindow()

GU_CloseCatalog()

GU_CloseFont()

GU_CoordsInGadBox()

GU_CountNodes()

GU_CreateContext()

GU_CreateGadgetA()

GU_CreateLocMenuA()

GU_CreateMenusA()

GU_DetachList()

GU_DisableGadget()

GU_DrawBevelBoxA()

GU_EndRefresh()

GU_FilterIMsg()

GU_FindNode()

GU_FreeGadgets()

GU_FreeInput()

GU_FreeLayoutGadgets()

GU_FreeMenus()

GU_FreeVisualInfo()

GU_GadgetArrayIndex()

GU_GetGadgetAttrsA()

GU_GetGadgetPtr()

GU_GetIMsg()

GU_GetLocaleStr()

GU_GetVisualInfoA()

GU_LayoutGadgetsA()

GU_LayoutMenuItemsA()

GU_LayoutMenusA()

GU_NewList()

GU_NodeDown()

GU_NodeUp()

GU_OpenCatalog()

GU_OpenFont()

GU_PostFilterIMsg()

GU_RefreshBoxes()

GU_RefreshWindow()

GU_ReplyIMsg()

GU_SetGadgetAttrsA()

GU_SetGUGadAttrsA()

GU_SetToggle()

GU_SizeWindow()

GU_SortList()

GU_TextWidth()

GU_UpdateProgress()

1.2 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, SR (Z)        D0   A0      A1

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

 FUNCTION
Add a node to the end of a list linked to a gadget (eg. a ListView).

Normally this is used to add a node to a listview list, but it can
also be used if you eg have another list that that contains
information about the items in the listview. This list will then
also be deallocated at the same time as the listview's own list.
If you do this, be sure to clear the list header of this other list
when you want to deallocate the list (you can use the GU_NewList
function).

 INPUTS
gad    - Gadget to keep the nodes in (ie the ListView gadget).

string - String to put in the nodes' LN_NAME field (visible entry).

list   - List to add the node to.

 RESULT
node - Struct Node for success, FALSE for error.
SR(Z)- 0 for success, 1 for error.

 NOTES
The SR(Z) is probably most usable for assembly programmers.

 BUGS
none known

 SEE ALSO

              GU_CountNodes()
              ,
              GU_NewList()
              ,
```

```
              GU_ClearList()
              ,
              GU_DetachList()
              ,

              GU_AttachList()
              ,
              GU_FindNode()
              ,
              GU_NodeUp()
              ,
              GU_NodeDown()
              ,
              GU_SortList()
```

1.3 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
Attach a new (or changed) list to a listview. The new list will be
shown immediately.

 INPUTS
gad  - Gadget to change.
win  - Window that the gadget is located in.
list - List to connect to the listview gadget.

 RESULT
none

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_NewList()

              GU_FindNode()
              ,
              GU_NodeUp()
              ,
```

```
               GU_NodeDown()
               ,
               GU_CountNodes()
               ,
               GU_SortList()
```

1.4 gadutil.library/GU_BeginRefresh

```
                  NAME
GU_BeginRefresh -- Begin refreshing friendly to GadTools.

 SYNOPSIS
GU_BeginRefresh(win)
                A0

VOID GU_BeginRefresh(struct Window *);

 FUNCTION

Invokes the intuition.library/BeginRefresh() function in a manner
friendly to the Gadget Toolkit. This function call permits the
GadTools gadgets to refresh themselves at the correct time.
Call
               GU_EndRefresh()
                function when done.

 INPUTS
win - pointer to Window structure for which a IDCMP_REFRESHWINDOW
      IDCMP event was received.

 NOTES
See gadtools/GT_BeginRefresh() for more information.

 SEE ALSO

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

1.5 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()
```

1.6 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
Change the string in a string gadget.

 INPUTS
gad    - Gadget to change
string - New string
win    - Window that the gadget is located in.

 RESULT
none

 BUGS
none known

 SEE ALSO
```

1.7 gadutil.library/GU_CheckVersion

```
 NAME
GU_CheckVersion -- Check the version.revision of a library
```

```
 SYNOPSIS
success = GU_CheckVersion(library, version, revision)
D0, SR(Z)                 A0       D0       D1

BOOL GU_CheckVersion(struct Library *, UWORD, UWORD);

 FUNCTION
Compares a library's version and revision with a required one.

 INPUTS
library  - a pointer to an opened library/device
version  - the version of the library that is required
revision - the revision of the library that is required

 RESULT
success - TRUE if the required version is older or equal to the
    opened one. FALSE otherwise.
SR(Z)   - 0 if function returns TRUE, 1 otherwise

 BUGS
none known

 SEE ALSO
```

1.8 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
Clear a listview gadget and deallocate all nodes.

 INPUTS
gad  - Gadget To Change
win  - Window that the gadget is located in.
list - List structure to deallocate nodes from.

 RESULT
none

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_NewList()
              ,
```

```
GU_DetachList()
,
GU_AttachList()

GU_FindNode()
,
GU_NodeUp()
,
GU_NodeDown()
,
GU_CountNodes()
,
GU_SortList()
```

1.9 gadutil.library/GU_ClearWindow

```
 NAME
GU_ClearWindow -- Fill the inside of a window with selected color.

 SYNOPSIS
GU_ClearWindow(window, color)
               A0      D0

VOID GU_ClearWindow(struct Window *, UWORD);

 FUNCTION
Fills the inner area of a window with given color.

 INPUTS
window - pointer to the window to be filled

color - the color to use. Use color 0 to clear the window.

 BUGS
none known

 SEE ALSO
```

1.10 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()
```

1.11 gadutil.library/GU_CloseFont

```
                  NAME
GU_CloseFont -- Release a pointer to a system font.

 SYNOPSIS
GU_CloseFont(font)
             A0

VOID GU_CloseFont(struct TextFont *);

 FUNCTION
This function indicates that the font specified is no longer
in use. It is used to close a font opened by GU_OpenFont, so
that fonts that are no longer in use do not consume system
resources.

 INPUTS
font - a font pointer as returned by
              GU_OpenFont()
                 RESULT

 BUGS
none known

 SEE ALSO

              GU_OpenFont()
```

1.12 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 */
  }
```

1.13 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.
           Note that count starts from one and not from zero!

 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.

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()

              GU_FindNode()
              ,
              GU_NodeUp()
              ,
              GU_NodeDown()
              ,
              GU_NewList()
              ,
              GU_SortList()
```

1.14 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
This function is a replacement for the GadTools version. Use this
instead. This will make your program to take advantage of future
enhancements of gadutil.

Creates a place for GadTools to store any context data it might
need for your window.  In reality, an unselectable invisible
gadget is created, with room for the context data.
This function also establishes the linkage from a glist type
pointer to the individual gadget pointers.  Call this function
before any of the other gadget creation calls.

 INPUTS

glistptr - Address of a pointer to a Gadget, which was previously
           set to NULL.  When all the gadget creation is done, you may
           use that pointer as your NewWindow.FirstGadget, or
           in intuition.library/AddGList(),
           intuition.library/RefreshGList(), FreeGadgets(), etc.

 RESULT
gad - pointer to context gadget, or NULL if failure.

 NOTES
Look in gadtools/CreateContext() for more information.

 SEE ALSO
gadtools/CreateContext()
```

1.15 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>
```

1.16 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_FreeMenus()
              , gadtools/CreateMenusA(), gadtools/LayoutMenusA(),
gadtools/FreeMenus()
```

1.17 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
CreateMenusA() allocates and initializes a complete menu
structure based on the supplied array of NewMenu structures.
Optionally, CreateMenusA() can allocate and initialize a complete
set of menu items and sub-items for a single menu title.  This
is dictated by the contents of the array of NewMenus.

These GadTools routines are only here to make it simpler for someone
who wants to use them. This for example, is of no use if you use the
GadUtil function
              GU_CreateLocMenuA()
               to define localized (optional)
menus with automatic hotkey handlimg etc.

 INPUTS
newmenu - pointer to an array of initialized struct NewMenus.
tagList - pointer to an array of tags providing optional extra
    parameters, or NULL.

 TAGS
See gadtools/CreateMenusA() for available tags.

 RESULT
```

```
menu - pointer to the resulting initialized menu structure (or
            the resulting FirstItem), with all the links for menu items
            and subitems in place.
            The result will be NULL if CreateMenusA() could not allocate
            memory for the menus, or if the NewMenu array had an
            illegal arrangement (eg. NM_SUB following NM_TITLE).
            (see also the GTMN_SecondaryError tag above).

 NOTES
See gadtools/CreateMenusA() for more information.

 SEE ALSO

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

1.18 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
Detach a listview's list (without visual effect).
This has to be done before you can change anything in the list,
eg add a node, delete a node etc.

Use
              GU_AttachList()
               to put the list back to the gadget.

 INPUTS
gad - Gadget to detach list from.
win - Window that the gadget is located in.

 RESULT
none

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_NewList()
```

```
                ,
                GU_AttachList()

                GU_FindNode()
                ,
                GU_NodeUp()
                ,
                GU_NodeDown()
                ,
                GU_CountNodes()
                ,
                GU_SortList()
```

1.19 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
Disables or enables a gadget (that supports this).

 INPUTS
status - The new status for the gadget, TRUE for disabled, FALSE for
         enabled.
gadget - A pointer to the gadget to change.
window - Pointer to the window that the gadget is placed in.

 RESULT
none

 BUGS
none known

 SEE ALSO
```

1.20 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
This function renders a bevelled box of specified dimensions
and type into the supplied RastPort.

 INPUTS
rport - RastPort into which the box is to be drawn.
left - left edge of the box.
top - top edge of the box.
width - width of the box.
height - height of the box.
tagList - pointer to an array of tags providing extra parameters

 RESULT
none

 NOTES
See gadtools/DrawBevelBoxA() for more information.

 SEE ALSO

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

1.21 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

Invokes the intuition.library/EndRefresh() function in a manner
friendly to the Gadget Toolkit.  This function call permits
GadTools gadgets to refresh themselves at the correct time.
Call this function to EndRefresh() when you have used

              GU_BeginRefresh()
              .

 INPUTS
win - pointer to Window structure for which a IDCMP_REFRESHWINDOW
      IDCMP event was received.
complete - TRUE when done with refreshing.

 NOTES
See gadtools/GT_EndRefresh for more information.

 SEE ALSO
```

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

1.22 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
NOTE WELL:  Extremely few programs will actually need this function.
You almost certainly should be using GT_GetIMsg() and GT_ReplyIMsg()
only, and not GT_FilterIMsg() and GT_PostFilterIMsg().

GT_FilterIMsg() takes the supplied IntuiMessage and asks the
Gadget Toolkit to consider and possibly act on it.  Returns
NULL if the message was only of significance to a GadTools gadget
(i.e. not to you), else returns a pointer to a modified IDCMP
message, which may contain additional information.

You should examine the Class, Code, and IAddress fields of
the returned message to learn what happened.  Do not make
interpretations based on the original imsg.

You should use GT_PostFilterIMsg() to revert to the original
IntuiMessage once you are done with the modified one.

 INPUTS
imsg - an IntuiMessage you obtained from a Window's UserPort.

 RESULT
modimsg - a modified IntuiMessage, possibly with extra information
          from GadTools, or NULL. When NULL, the message passed in to
    the function should be sent back to Intuition via ReplyMsg().

 NOTES
See gadtools/GT_FilterIMsg() for more information.

 SEE ALSO

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

1.23 gadutil.library/GU_FindNode

```
                  NAME
GU_FindNode -- Find the node structure of a given node number

 SYNOPSIS
node = GU_FindNode(list, number)
D0,A0,SR(Z)        A0    D0

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

 FUNCTION
Finds a specified node in a list.

 INPUTS
list   - Struct List. The list in where to search.
number - Number of the node to find (counting from 0).

 RESULT
node  - Address to the node structure or NULL if the node wasn't
        existing.

SR(Z) - 0 for success, 1 for failure.

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()

              GU_NodeUp()
              ,
              GU_NodeDown()
              ,
              GU_CountNodes()
              ,
              GU_NewList()
              ,
              GU_SortList()
```

1.24 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()
               , gadtools/FreeGadgets().
```

1.25 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()
```

1.26 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
Frees gadgets laid out with LayoutGadgetsA().

 INPUTS
gad_info - The pointer returned by LayoutGadgetsA().

 RESULT
none

 SEE ALSO
```

1.27 gadutil.library/GU_FreeMenus

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

 SYNOPSIS
GU_FreeMenus(menu)
             A0

VOID GU_FreeMenus(struct Menu *);

 FUNCTION
Frees the menus allocated by
              GU_CreateMenusA()
              .  It is safe to
call this function with a NULL parameter.

 INPUTS
menu - pointer to menu structure (or first MenuItem) obtained
       from
              GU_CreateMenusA()
              .

 NOTES
See gadtools/FreeMenus() for more information.

 SEE ALSO
```

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

1.28 gadutil.library/GU_FreeVisualInfo

```
                  NAME
GU_FreeVisualInfo -- Return any resources taken by GU_GetVisualInfoA

 SYNOPSIS
GU_FreeVisualInfo(vi)
                   A0

VOID GU_FreeVisualInfo(APTR);

 FUNCTION
FreeVisualInfo() returns any memory or other resources that
were allocated by GetVisualInfoA(). You should only call this
function once you are done with using the gadgets (i.e. after
      CloseWindow()), but while the screen is still valid (i.e. before
      CloseScreen() or UnlockPubScreen()).

 INPUTS
vi - pointer that was obtained by calling GetVisualInfoA(). This
     value may be NULL.

 NOTES
See gadtools/FreeVisualInfo() for more information.

 SEE ALSO

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

1.29 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
Get a gadget's index in the Layoutgadget structure.
 INPUTS
id      - The ID of the gadget you want to find.
gadgets - The LayoutGadget array that this gadget is defined in.

 RESULT
index - The index into the LayoutGadget array of the entry with the
```

```
        gadget ID you asked for. Returns -1 for failure.

SR(Z) - Set for failure. Cleared otherwise. Probably only usable for
        assembly language programmers.

 SEE ALSO
```

1.30 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
Retrieve the attributes of the specified gadget, according to the
attributes chosen in the tag list.  For each entry in the tag list,
ti_Tag identifies the attribute, and ti_Data is a pointer to
the long variable where you wish the result to be stored.

 INPUTS
gad     - Pointer to the gadget in question, may be NULL.
win     - Pointer to the window containing the gadget
req     - Reserved for future use, should always be NULL
taglist - Pointer to a TagItem list

 RESULT
numProcessed - The number of attributes successfully filled in.

 NOTES
See gadtools/GT_GetGadgetAttrsA() for tags and examples.

Requires kickstart 3.0 (V39).

 SEE ALSO

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

1.31 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
```

```
APTR 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(a6) ; 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.
```

1.32 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()
```

1.33 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()
```

1.34 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

Get a pointer to a (private) block of data containing various bits
of information that GadTools needs to ensure the best quality
visuals.  Use the result in the NewGadget structure of any gadget
you create, or as a parameter to the various menu calls.  Once the
gadgets/menus are no longer needed (after the last CloseWindow()),
call FreeVisualInfo().

 INPUTS
screen - pointer to the screen you will be opening on. This parameter
   may be NULL, in which case this function fails.
tagList - pointer to an array of tags providing optional extra
    parameters, or NULL.

 RESULT
vi - pointer to private data, or NULL for failure
```

```
 NOTES
See gadtools/GetVisualInfoA() for more information.

 SEE ALSO

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

1.35 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 or a LABEL_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 an array of tags providing optional extra
    parameters, or NULL.

                These tags can be used here:

                  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 overridden 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.
```

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.

TAGS

Tags for the gadgets lg_LayoutTags taglist. The other tags can be found in the autodoc to gadtools.library/CreateGadgetA().

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.

The function
GU_RefreshBoxes()
can be used to redraw all
bevelboxes.

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)
Flags for text placement, text shadowing and 3D text:

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-

Combined flags
~~~~~~~~~~~~~~
   BB_TEXT_ABOVE_CENTER - BB_TEXT_ABOVE + BB_TEXT_CENTER
   BB_TEXT_ABOVE_LEFT   - BB_TEXT_ABOVE + BB_TEXT_LEFT
   BB_TEXT_ABOVE_RIGHT  - BB_TEXT_ABOVE + BB_TEXT_RIGHT

   BB_TEXT_IN_CENTER    - BB_TEXT_IN + BB_TEXT_CENTER
   BB_TEXT_IN_LEFT      - BB_TEXT_IN + BB_TEXT_LEFT
   BB_TEXT_IN_RIGHT     - BB_TEXT_IN + BB_TEXT_RIGHT

   BB_TEXT_BELOW_CENTER - BB_TEXT_BELOW + BB_TEXT_CENTER
   BB_TEXT_BELOW_LEFT   - BB_TEXT_BELOW + BB_TEXT_LEFT
   BB_TEXT_BELOW_RIGHT  - BB_TEXT_BELOW + BB_TEXT_RIGHT

Default is 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.

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 is BB_SHADOW_DR (BB_SUNAT_UL).

   BB_3DTEXT    - This flag can be used in place of the tag
                  GUBB_3DText, TRUE
```

```
            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.
               Another way to enable 3D text is to set the flag BB_3DTEXT
               in the GUBB_Flags tag.

            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.

    LABEL_KIND
      A text label. Use this to avoid the use of absolute placement
      of text that you print into the window. Supports the most of
      the text placement and shadow flags for the BEVELBOX_KIND.

      The function
           GU_RefreshBoxes()
            can be used to redraw all
      text created by LABEL_KIND gadgets.

      Extra tags for LABEL_KIND:

         GULB_TextColor (ULONG)
            Selects which color to print the text in. Defaults to the
            color of the TEXTPEN.

         GULB_TextPen (ULONG)
            Selects which pen to print the text in. Defaults to TEXTPEN.
            This tag overrides the GULB_TextColor tag.

         GULB_Flags (ULONG)
            Flags for text placement, text shadowing and 3D text:

            ___1_____2_____3___
            |_____|_____|_____| A
            |_____|_____|_____| B
            |_____|_____|_____| C

            Y-pos flags
            ~~~~~~~~~~~
               LB_TEXT_TOP    - Places the topmost point of the text below
                                the upper border of the box (row A)

               LB_TEXT_MIDDLE - Places the text centered in the box, not
                                counting in the part of the text that is
                                below the font's baseline (row B)

               LB_TEXT_BOTTOM - Places the text at the bottom of the box.
                                Any part of the text that is below the
                                baseline will be below the box (row C)

            X-pos flags
            ~~~~~~~~~~~
               LB_TEXT_CENTER - Places the text centered on the width of
                                the box (column 2).

               LB_TEXT_LEFT   - Places the text left adjusted in the box
                                (column 1)

               LB_TEXT_RIGHT  - Places the text right adjusted in the box
```

(column 3)

Combined flags
~~~~~~~~~~~~~~

LB_TEXT_TOP_CENTER - LB_TEXT_TOP + LB_TEXT_CENTER
LB_TEXT_TOP_LEFT - LB_TEXT_TOP + LB_TEXT_LEFT
LB_TEXT_TOP_RIGHT - LB_TEXT_TOP + LB_TEXT_RIGHT

LB_TEXT_MIDDLE_CENTER - LB_TEXT_MIDDLE + LB_TEXT_CENTER
LB_TEXT_MIDDLE_LEFT - LB_TEXT_MIDDLE + LB_TEXT_LEFT
LB_TEXT_MIDDLE_RIGHT - LB_TEXT_MIDDLE + LB_TEXT_RIGHT

LB_TEXT_BOTTOM_CENTER - LB_TEXT_BOTTOM + LB_TEXT_CENTER
LB_TEXT_BOTTOM_LEFT - LB_TEXT_BOTTOM + LB_TEXT_LEFT
LB_TEXT_BOTTOM_RIGHT - LB_TEXT_BOTTOM + LB_TEXT_RIGHT

Default is LB_TEXT_TOP|LB_TEXT_CENTER. Combine the x and y position flags by OR:ing them together. Don't combine two X or two Y flags together.

Shadow placement flags
~~~~~~~~~~~~~~~~~~~~~~

LB_SHADOW_DR - Places the bevel box text shadow one pixel below and to the right of the text.

LB_SHADOW_UR - Places the bevel box text shadow one pixel above and to the right of the text.

LB_SHADOW_DL - Places the bevel box text shadow one pixel below and to the left of the text.

LB_SHADOW_UL - Places the bevel box text shadow one pixel above and to the left of the text.

LB_SUNAT_UL - Another name for LB_SHADOW_DR
LB_SUNAT_DL - Another name for LB_SHADOW_UR
LB_SUNAT_UR - Another name for LB_SHADOW_DL
LB_SUNAT_DR - Another name for LB_SHADOW_UL

Default is LB_SHADOW_DR (LB_SUNAT_UL).

LB_3DTEXT - This flag can be used in place of the tag GULB_3DText, TRUE

GULB_3DText (BOOL)
Enables the shadow on the text. This tag must be used if GULB_ShadowColor or GULB_ShadowPen isn't used. Another way to enable 3D text is to set the flag LB_3DTEXT in the GULB_Flags tag.

GULB_ShadowColor (ULONG)
Selects which color to print the shadow text in. Defaults to the color of the SHADOWPEN.

GULB_ShadowPen (ULONG)
Selects which pen to print the shadow text in. Defaults to SHADOWPEN. This tag overrides the GULB_ShadowColor tag.

```
      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.

   GU_FractWidth (LONG parts)
Divide or multiply the gadget's width with ti_Data. A positive
value divides the gadget's width by the ti_Data, a negative ti_Data
multiplies the gadget's width with ti_Data.

 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).
```

```
   GU_FractHeight (LONG parts)
Divide or multiply the gadget's height with ti_Data. A positive
value divides the gadget's height by the ti_Data, a negative ti_Data
multiplies the gadget's height with ti_Data.

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.

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.

      Also note that you do not have to specify any tags that do not
      change from gadget to gadget. Just be sure that you know in which
order the gadgets are processed (eg. relatives etc).

Another thing to note, is that we have tried to make the processing
of position and width /~height tags as usable as possible, what I
mean with this, is that if you eg first define the left edge and
then define the right edge, the width will change. BUT, there are
special cases when this isn't true. This is because we have tried
out this and decided that this was the best way to do it.

Here comes some examples of the special cases;

dc.l  GU_AlignLeft, GAD_1 ; Left edge aligned with GAD_1's left.
dc.l  GU_AlignRight, GAD_2  ; This stretches the gadget, so that
        ; both the left and right edges are
        ; positioned as defined.
; Then, if we want to move the right edge 2 pixels right, and the left
; edge two pixels right, we might try this:

dc.l  GU_AddLeft, -2    ; This works as we want, it moves the
        ; left edge to the right place, but it
        ; also moves the whole gadget two
        ; pixels left..

dc.l  GU_AddRight, 2    ; <- This is a common mistake. This
        ; moves the whole gadget to the right.
        ; Ie. it is moved back to the old
        ; position, not as we wanted...

; But if we replace the previous line with the following;

dc.l  GU_AddWith,4    ; This works just as we wanted it to.
        ; Now the gadget should be 4 pixels
        ; wider, two to the left and two to
        ; the right.

The same goes for GU_AddHeight and GU_AddBottom etc.
```

```
This is actually a feature. Sometimes you might want to move the whole
button, so we made it work this way.


 SEE ALSO

              GU_FreeLayoutGadgets()
              ,
              GU_CreateGadgetA()
              , gadtools/CreateGadgetA()

              GU_RefreshWindow()
              ,
              GU_RefreshBoxes()
              ,
              GU_UpdateProgress()
```

1.36 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
Lays out all the menu items and sub-items according to
the supplied visual information and tag parameters. You would use this
if you used CreateMenusA() to make a single menu-pane (with sub-items,
if any), instead of a whole menu strip.
This routine attempts to columnize and/or shift the MenuItems in
the event that a menu would be too tall or too wide.

 INPUTS
menuitem - Pointer to the first MenuItem in a linked list of items.
vi       - Pointer returned by
               GU_GetVisualInfoA()
               .
tags     - Pointer to an array of tags providing optional extra
           information.

 TAGS
See gadtools/LayoutMenuItemsA() for tags.

 RESULT
success  - TRUE if successfull, FALSE otherwise.

 SEE ALSO

              GU_CreateMenusA()
              ,
              GU_GetVisualInfoA()
```

```
              , gadtools/LayoutMenuItemsA()
```

1.37 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
Lays out all the menus, menu items and sub-items in the supplied
menu according to the supplied visual information and tag parameters.
This routine attempts to columnize and/or shift the MenuItems in
the event that a menu would be too tall or too wide.

These GadTools routines are only here to make it simpler for someone
who wants to use them. This for example, is of no use if you use the
GadUtil function
              GU_CreateLocMenuA()
               to define localized (optional)
menus with automatic hotkey handlimg etc.

 INPUTS
menu    - Pointer to menu obtained from
              GU_CreateMenusA()
               or

              GU_CreateLocMenuA()
              .
vi      - Pointer returned by GU_GetVisualInfoA.
taglist - Pointer to an array of tags providing optional extra
          parameters.

 TAGS
See gadtools/LayoutMenusA() for tags.

 RESULT
success - TRUE if successfull, FALSE otherwise.

 NOTES
See gadtools/LayoutMenusA() for more information.

 SEE ALSO

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

1.38 gadutil.library/GU_NewList

```
                  NAME
GU_NewList -- Initialize a list header for use.

 SYNOPSIS
GU_NewList(list)
           A0

VOID GU_NewList(struct List *);

 FUNCTION
This initializes a list header for use. Much easier than to do it
by hand.

 INPUTS
list - Struct List.

 RESULT
none

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()

              GU_FindNode()
              ,
              GU_NodeUp()
              ,
              GU_NodeDown()
              ,
              GU_CountNodes()
              ,
              GU_SortList()
```

1.39 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,SR(Z)              A0    A1

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

```
 FUNCTION
Move a node one step downwards in a list. To do the opposite, see

              GU_NodeUp()
              .

 INPUTS
node - The node to move.
list - The list that the node is a part of.

 RESULT
success - TRUE if the node could be moved, else FALSE.
SR (Z)  - 0 if node could be moved, else 1.

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()

              GU_FindNode()
              ,
              GU_NodeUp()
              ,
              GU_CountNodes()
              ,
              GU_NewList()
              ,
              GU_SortList()
```

1.40 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,SR(Z)            A0    A1

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

 FUNCTION
Move a node one step up. You can also use the function
              GU_NodeDown()
                for moving downwards.

 INPUTS
```

```
node - The node to move.
list - The list that the node is a part of.

 RESULT
success - TRUE if the node could be moved, else FALSE
SR(Z)   - 0 if the node could be moved, else 1.

 BUGS
none known

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()
              ,

              GU_FindNode()
              ,
              GU_NodeDown()
              ,
              GU_CountNodes()
              ,
              GU_NewList()
              ,
              GU_SortList()
```

1.41 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()
```

1.42 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
```

```
Open a disk or ROM based font. Uses diskfont library if available.
Much easier to use than the standard OpenFont functions.

 INPUTS
textAttr - This is a pointer to a TextAttr structure.

 RESULT
font - Struct TextFont ot NULL for failure.

 BUGS
none known

 SEE ALSO

              GU_CloseFont()
```

1.43 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
NOTE WELL:  Extremely few programs will actually need this function.
You almost certainly should be using GT_GetIMsg() and GT_ReplyIMsg()
only, and not GT_FilterIMsg() and GT_PostFilterIMsg().

Performs any clean-up necessitated by a previous call to
GT_FilterIMsg().  The original IntuiMessage is now yours to handle.
Do not interpret the fields of the original IntuiMessage, but
rather use only the one you got from GT_FilterIMsg().  You
may only do message related things at this point, such as queueing
it up or replying it.  Since you got the message with
exec.library/GetMsg(), your responsibilities do include replying
it with exec.library/ReplyMsg(). This function may be safely
called with a NULL parameter.

 INPUTS
modimsg - A modified IntuiMessage obtained with
              GU_FilterIMsg()
              ,
          or NULL.

 RESULT
imsg - a pointer to the original IntuiMessage, if GT_FilterIMsg()
       returned non-NULL.
```

```
 NOTES
See gadtools/GT_PostFilterIMsg for more information.

 SEE ALSO

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

1.44 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
Redraw all bevel boxes and PROGRESS_KIND gadgets in a window.
Also refreshes all text in the window that is created by the
LABEL_KIND gadget. This function is basically the same as
GU_RefreshWindow, without a call to the GT_RefreshWindow
function.

 INPUTS
window - Window to be refreshed.

gad_info - The value returned by
              GU_LayoutGadgetsA()
                 RESULT
none

 BUGS
no known

 SEE ALSO

              GU_UpdateProgress()
              ,
              GU_RefreshWindow()
```

1.45 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_RefreshBoxes()
              ,
              GU_UpdateProgress()
```

1.46 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()
```

1.47 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()
```

1.48 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
Change attributes of a GadUtil gadget.

 INPUTS
gad_info - The value returned by LayoutGadgetsA().
gad      - Pointer to the gadget in question.
win      - Pointer to the window containing the gadget.
taglist  - Pointer to a TagItem list.
 TAGS
See
             GU_LayoutGadgetsA()
              for tags.

 RESULT
none

 SEE ALSO
```

1.49 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
This function selects or unselects a toggle gadget, that means a
gadget with the tag GU_ToggleSelect set to TRUE.
```

```
For some more information about the tags, look in <libraries/gadutil.h>
or in the autodoc for the
              GU_LayoutGadgetsA()
               function.

 INPUTS
status - New status for the gadget. TRUE for selected, FALSE for
         unselected.
gadget - The gadget to change.
window - Window that the gadget is located in.

 RESULT
none

 BUGS
none known

 SEE ALSO

              GU_LayoutGadgetsA()
              , <libraries/gatutil.h>, <libraries/gadutil.i>
```

1.50 gadutil.library/GU_SizeWindow

```
 NAME
GU_SizeWindow -- Resize a window, and moves it if necessary

 SYNOPSIS
success = GU_SizeWindow(window, deltax, deltay)
D0,SR(Z)                A0      D0      D1

BOOL GU_SizeWindow(struct Window *, WORD, WORD);

 FUNCTION
This function sends a reuqest to Intuition asking to size the window
the specified amounts. The delta arguments describes how much to
size the window along the respective axes.

This function works like the Intuition function SizeWindow(), and the
parameters are also the same. This function also moves the window
automatically if it could not be sized at the current position.

Remember that this function does not change the window immediately.
Just like the Intuition functions SizeWindow(), MoveWindow() and
ChangeWindowBox(), you have to wait for a IDCMP_CHANGEWINDOW IDCMP
message to arrive.

An extra feature of GU_SizeWindow is that it sends back a result,
telling you if the window could be resized or not.

 INPUTS
window - pointer to the structure of the window to be sized

deltax - signed value describing how much to size the window
         on the x-axis
```

```
deltay - signed value describing how much to size the window
         on the y-axis

 RESULT
success - TRUE if the window could be sized the specified amount

SR(Z)   - 0 if function returns TRUE, 1 otherwise

 BUGS
none known

 SEE ALSO
intuition/ChangeWindowBox(), intuition/SizeWindow()
intuition/MoveWindow()
```

1.51 gadutil.library/GU_SortList

```
                 NAME
GU_SortList -- Sorts all nodes in a list.

 SYNOPSIS
GU_SortNode(list,slavelist)
            A0   A1

VOID GU_SortList(struct List *, struct List *);

 FUNCTION
Sorts all items an a list and optionally in a slave list.

The slave list option can be used eg. if you have a ListView
gadget that shows a number of filenames and another list
with the path-names to the files in the other list.

Note that the slavelist pointer must be NULL if not used.

 INPUTS
list      - A pointer to the list to sort
slavelist - A pointer to an optional slave list (or NULL).

 NOTES
Written in assembler using Shell-sort, so it is quite fast..

 SEE ALSO

              GU_AddTail()
              ,
              GU_ClearList()
              ,
              GU_DetachList()
              ,
              GU_AttachList()

              GU_FindNode()
              ,
```

```
GU_NodeUp()
,
GU_NodeDown()
,
GU_NewList()
,
GU_CountNodes()
```

1.52 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.

 RESULT
textwidth - pixel length of the text
```

1.53 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()