GADGETSLIB

Conversion program

COLLABORATORS			
	TITLE : GADGETSLIB		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Conversion program	October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GADGETSLIB

1.1 Overview of GADGETSLIB

Overview

An Acid Software Library

Converted to AmigaGuide by

Red When Excited Ltd

Used with the permission of Acid Software

Edited, fixed and cleaned by Toby Zuijdveld 26/02/1999.
mailto:hotcakes@abacus.net.au

1.2 GADGETSLIB

```
Statement: SetPropInfo
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : SetPropInfo GadgetList#,Id,visible,total[,current]

This command allows you to setup proportional gadgets so that they can be
used like gadgets from the gadtools library.  Thus you can set the size of
the pot and body of a propgadget by specifying values like total and
visible - just like the GadTools GTScroller command.

A gadget setup using this command can still be used in the traditional
manner.
```

1.3 GADGETSLIB

```
Statement: GetPropCurrent
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : pos=GetPropCurrent(GadgetList#,Id)

Returns the current position of a propgadgets body.  Note that the
propgadget must have been setup with the SetPropInfo command before you can
call this command on it.  The return value of this command will be the
current position of the body, in the range of 0 to (total-visible).
```

1.4 GADGETSLIB

```
Statement: MoveProp
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : MoveProp GadgetList#,Id,current[,window_redraw]

This command allows you to set a proportional gadgets position.  It should
only be used on gadgets that have been previously initialised with
SetPropInfo.

The optional window_redraw flag allows you to specify whether or not this
gadget should be redraw straight after the change has been made.
```

1.5 GADGETSLIB

```
Statement: TextGadget
------------------------------------------------------------------------------
Modes  :
Syntax : TextGadget GadgetList#,X,Y,Flags,Id,Text$

The TextGadget command adds a text gadget to a gadgetlist. A text
gadget is the simplest type of gadget consisting of a sequence of
characters optionally surrounded by a border.

Flags should be selected from the table at the start of the
chapter.

Boolean gadgets are the simplest type of gadget available. Boolean
gadgets are 'off' until the program user clicks on them with the mouse,
which turns them 'on'. When the mouse button is released, these gadgets
revert back to their 'off' state. Boolean gadgets are most often used
for 'OK' or 'CANCEL' type gadgets.

Toggle gadgets differ in that each time they are clicked on they change
their state between 'on' and 'off'. For example, clicking on a toggle
gadget which is 'on' will cause the gadget to be turned 'off', and vice
versa.
```

X and Y specify where in the window the gadget is to appear. Depending upon the Flags setting, gadgets may be positioned relative to any of the 4 window edges. If a gadget is to be positioned relative to either the right or bottom edge of a window, the appropriate X or Y parameter should be negative.

Id is an identification value to be attached to this gadget. All gadgets in a gadgetlist should have unique Id numbers, allowing you to detect which gadget has been selected. Id may be any positive, non-zero number.

Text$ is the actual text you want the gadget to contain.

1.6 GADGETSLIB

Statement: StringGadget

Modes :
Syntax : StringGadget GadgetList#,X,Y,Flags,Id,Maxlen,Width

StringGadget allows you to create an Intuition style 'text entry' gadget. When clicked on, a string gadget brings up a text cursor, and is ready to accept text entry from the keyboard.

X and Y specifies the gadgets position, relative to the top left of the window it is to appear in.

See the beginning of the chapter for the relevant Flags for a string gadget.

Id is an identification value to be attached to this gadget. All gadgets in a gadgetlist should have unique Id numbers, allowing you to detect which gadget has been selected. Id may be any positive, non-zero number.

Maxlen refers to the maximum number of characters which may appear in this gadgets.

Width refers to how wide, in pixels, the gadget should be. A string gadget may have a width less than the maximum number of characters it may contain, as characters will be scrolled through the gadget when necessary.

You may read the current contents of a string gadget using the StringText function.

1.7 GADGETSLIB

Statement: PropGadget

```
Modes  :
Syntax : PropGadget GadgetList#,X,Y,Flags,Id,Width,Height,[propflags, ↩
   propactivation[,typemask]]
```

The PropGadget command is used to create a 'proportional gadget'. Proportional gadgets present a program user with a 'slider bar', allowing them to adjust the slider to achieve a desired effect. Proportional gadgets are commonly used for the 'R G B' sliders seen in many paint packages.

Proportional gadgets have 2 main qualities - a 'pot' (short for potentiometer) setting, and a 'body' setting.

The pot setting refers to the current position of the slider bar, and is in the range 0 through 1. For example, a proportional gadget which has been moved to 'half way' would have a pot setting of '.5'. The body setting refers to the size of the units the proportional gadget represents, and is again in the range 0 through 1. Again taking the RGB colour sliders as an example, each slider is intended to show a particular value in the range 0 through 15 - giving a unit size, or body setting, of 1/16 or '.0625'.

Put simply, the pot setting describes 'where' the slider bar is, while the body setting describes 'how big' it is.

Proportional gadgets may be be represented as either horizontal slider bars, vertical slider bars, or a combination of both.

See the beginning of the chapter for relevant Flags settings for prop gadgets.

X and Y refer to the gadgets position, relative to the top left of the window it is opened in.

Width and Height refer to the size of the area the slider should be allowed to move in.

Id is a unique, non zero number which allows you to identify when the gadget is manipulated.

Proportional gadgets may be altered using the SetVProp and SetHProp commands, and read using the VPropPot, VPropBody, HPropPot and HPropBody functions.

New Parameters

This new version of the PropGadget command allows you to specify extra flag values for the gadget. When you use this extra values, the propgadget is initialised using the OS flags. That is, all the flags defined in the OS include files can be used, allowing the use of flags like PROPNEWLOOK to make the gadget look like it belongs in OS2+.

The parameter 'flags' is changed when using the extra parameters to represent actual OS include file defined flags.

The extra parameters are:

```
  propflags Define propgadget specific flags (e.g PROPNEWLOOK)
  propactication  Control activation situations for the gadget
  type    Change type of gadget (e.g. make it a GIMMEZEROZERO
      border gadget)
```

1.8 GADGETSLIB

```
Statement: ArrowGadget
------------------------------------------------------------------------------
Modes  :
Syntax : ArrowGadget GadgetList#,Id,direction,x,y,relativity[,special_flags]

This is a new command for this library that allows you to access the OS
provided set of four arrow gadgets.  These arrow gadgets can be seen in the
borders of Workbench windows.  This command allows you to use these arrow
gadgets as normal gadgets inside your GadgetLists.

Relativity is defined as:
#RIGHTREL =%1   Relative to right of window
#BOTTOMREL  =%10    Relative to bottom of window

Special flags is made up of:
#ISGZZGADGET  =%1   Put in border of a GIMMEZEROZERO window
#IMMEDIATE  =%10    Hear #IDCMP_GADGETDOWN events from this
        gadget

Direction is one of:
#LEFTIMAGE  =$0A    Left-arrow gadget image
#UPIMAGE  =$0B    Up-arrow gadget image
#RIGHTIMAGE =$0C    Right-arrow gadget image
#DOWNIMAGE  =$0D    Down-arrow gadget image
```

1.9 GADGETSLIB

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

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

The ShapeGadget command has been extended to allow an alternative image
to be displayed when the gadget is selected.

All other parameters are identical to those in TextGadget.
```

1.10 GADGETSLIB

```
Statement: SetHProp
------------------------------------------------------------------------------
Modes  :
Syntax : SetHProp GadgetList#,Id,Pot,Body

SetHProp is used to alter the horizontal slider qualities of a
proportional gadget. Both Pot and Body should be in the range 0 through
1.

If SetHProp is executed while the specified gadget is already
displayed, execution of the ReDraw command will be necessary to display
the changes.

For a full discussion on proportional gadgets, please refer to the
PropGadget command.
```

1.11 GADGETSLIB

```
Statement: SetVProp
------------------------------------------------------------------------------
Modes  :
Syntax : SetVProp GadgetList#,Id,Pot,Body

SetVProp is used to alter the vertical slider qualities of a
proportional gadget. Both Pot and Body should be in the range 0 through
1.

If SetVProp is executed while the specified gadget is already
displayed, execution of the ReDraw command will be necessary to display
the changes.

For a full discussion on proportional gadgets, please refer to the
PropGadget command.
```

1.12 GADGETSLIB

```
Function: HPropPot
------------------------------------------------------------------------------
Modes  :
Syntax : HPropPot (GadgetList#,Id)

The HPropPot function allows you to determine the current 'pot' setting
of a proportional gadget. HPropPot will return a number from 0 up to,
but not including, 1, reflecting the gadgets current horizontal pot
setting.

Please refer to the PropGadget command for a full discussion on
proportional gadgets.
```

1.13 GADGETSLIB

```
Function: HPropBody
-------------------------------------------------------------------------------
Modes  :
Syntax : HPropBody (GadgetList#,Id)

The HPropBody function allows you to determine the current 'body'
setting of a proportional gadget. HPropBody will return a number from 0
up to, but not including, 1, reflecting the gadgets current horizontal
body setting.

Please refer to the PropGadget command for a full discussion on
proportional gadgets.
```

1.14 GADGETSLIB

```
Function: VPropPot
-------------------------------------------------------------------------------
Modes  :
Syntax : VPropPot (GadgetList#,Id)

The VPropPot function allows you to determine the current 'pot' setting
of a proportional gadget.
VPropPot will return a number from 0 up to, but not including, 1,
reflecting the gadgets current vertical pot setting.

Please refer to the PropGadget command for a full discussion on
proportional gadgets.
```

1.15 GADGETSLIB

```
Function: VPropBody
-------------------------------------------------------------------------------
Modes  :
Syntax : VPropBody (GadgetList#,Id)

The VPropBody function allows you to determine the current 'body'
setting of a proportional gadget.

VPropBody will return a number from 0 up to, but not including, 1,
reflecting the gadgets current vertical body setting.

Please refer to the PropGadget command for a full discussion on
proportional gadgets.
```

1.16 GADGETSLIB

```
Statement: Redraw
--------------------------------------------------------------------------------
Modes  :
Syntax : Redraw Window#[,Id]

ReDraw will redisplay the specified gadget in the specified window.
This command is mainly of use when a proportional gadget has been
altered using SetHProp or SetVProp and needs to be redrawn, or when a
string gadget has been cleared using ClearString, and, likewise, needs
to be redrawn.

If you use this command without an Id parameter, all gadgets attached to
the window will be refreshed.
```

1.17 GADGETSLIB

```
Statement: Gap1
--------------------------------------------------------------------------------
Modes  :
Syntax : Gap1 *Gap2
```

1.18 GADGETSLIB

```
Statement: Gap2
--------------------------------------------------------------------------------
Modes  :
Syntax : Gap2
```

1.19 GADGETSLIB

```
Statement: Toggle
--------------------------------------------------------------------------------
Modes  :
Syntax : Toggle GadgetList#,Id [,On|Off]

The Togggle command in the gadget library has been extended so it will
actually toggle a gadgets status if the no On|Off parameter is
missing.
```

1.20 GADGETSLIB

```
Statement: Gap3
------------------------------------------------------------------------------
Modes  :
Syntax : Gap3 *GadgetBorder

The GadgetBorder command may be used to draw a rectangular border into the
currently used window.

Proportional gadgets and shape gadgets do not have borders automatically
created for them. The GadgetBorder command may be used, once a window is
opened, to render borders around these gadgets.

X,Y, Width and Height refer to the position of the gadget a border is
required around. GadgetBorder will automatically insert sapces between the
gadget and the border. The Borders command may be used to alter the amount
of spacing.

Of course, GadgetBorder may be used to draw a border around any arbitary
area, regardless of whether or not that area contains a gadget.
```

1.21 GADGETSLIB

```
Statement: GadgetBorder
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetBorder X,Y,Width,Height

The GadgetBorder command may be used to draw a rectangular border into
the currently used window.

Proportional gadgets and shape gadgets do not have borders
automatically created for them. The GadgetBorder command may be used,
once a window is opened, to render borders around these gadgets.

X,Y, Width and Height refer to the position of the gadget a border is
required around. GadgetBorder will automatically insert sapces between
the gadget and the border. The Borders command may be used to alter the
amount of spacing.

Of course, GadgetBorder may be used to draw a border around any
arbitary area, regardless of whether or not that area contains a
gadget.
```

1.22 GADGETSLIB

```
Statement: Borders
------------------------------------------------------------------------------
Modes  :
```

```
Syntax : Borders [On|Off]|[Width,Height]

Borders serves 2 purposes. First, Borders may be used to turn on or off
the automatic creation of borders around text and string gadgets.
Borders are created when either a Textgadget or StringGadget command is
executed. If you wish to disable this, Borders Off should be executed
before the appropriate TextGadget or StringGadget command.

Borders may also be used to specify the spacing between a gadget and
it's border, Width referring to the left/right spacing, and Height to
the above/below spacing.
```

1.23 GADGETSLIB

```
Statement: ActivateString
------------------------------------------------------------------------------
Modes  :
Syntax : ActivateString Window#,Id

ActivateString may be used to 'automatically' activate a string gadget.
This is identical to the program user having clicked in the string
gadget themselves, as the string gadget's cursor will appear, and
further keystrokes will be sent to the string gadget.

It is often nice of a program to activate important string gadgets, as
it saves the user the hassle of having to reach for the mouse before
the keyboard.
```

1.24 GADGETSLIB

```
Statement: ResetString
------------------------------------------------------------------------------
Modes  :
Syntax : ResetString GadgetList#,Id

ResetString allows you to 'reset' a string gadget. This will cause the
string gadget's cursor position to be set to the leftmost
position.
```

1.25 GADGETSLIB

```
Function: StringText$
------------------------------------------------------------------------------
Modes  :
Syntax : StringText$ (GadgetList#,Id)

The Stringtext$ function allows you to determine the current contents
```

of a string gadget. StringText$ will return a string of characters
representing the string gadgets contents.

1.26 GADGETSLIB

```
Statement: ClearString
------------------------------------------------------------------------------
Modes  :
Syntax : ClearString GadgetList#,Id

ClearString may be used to clear, or erase, the text in the specified
string gadget. The cursor position will also be moved to the leftmost
position in the string gadget.

If a string gadget is cleared while it is displayed in a window, the
text will not be erased from the actual display. To do this, ReDraw
must be executed.
```

1.27 GADGETSLIB

```
Statement: GadgetList
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetList List of Gadgets
```

1.28 GADGETSLIB

```
Statement: GadgetPens
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetPens Foreground Colour[,Background Colour]

GadgetPens determines the text colours used when text gadgets are
created using the TextGadget command. The default values used for
gadget colours are a foreground colour of 1, and a background colour of
0.
```

1.29 GADGETSLIB

```
Statement: BorderPens
------------------------------------------------------------------------------
Modes  :
```

```
Syntax : BorderPens Highlight Colour,Shadow Colour

BorderPens allows you to control the colours used when gadget borders
are created. Gadget borders may be created by the TextGadget,
StringGadget and GadgetBorder commands.

HighLight Colour refers to the colour of the top and left edges of the
border, while Shadow Colour refers to the right and bottom
edges.

The default value for HighLight Colour is 1. The default value for
Shadow Colour is 2.
```

1.30 GADGETSLIB

```
Statement: GadgetJam
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetJam Jammode

GadgetJam allows you to determine the text rendering method used when
gadgets are created using the TextGadget command. Please refer to the
WJam command in the windows chapter for a full description of jam modes
available.
```

1.31 GADGETSLIB

```
Statement: SelectMode
------------------------------------------------------------------------------
Modes  :
Syntax : SelectMode 1=Box, 0=Inverse

SelectMode is used to predefine how gadget rendering will show a gadget
selection, modes are 1 for box and 0 for inverse. Use prior to creation
of gadgets.
```

1.32 GADGETSLIB

```
Statement: SetString
------------------------------------------------------------------------------
Modes  :
Syntax : SetString GadgetList#,Id,String$

SetString may be used to initialize the contents of a string gadget
created with the StringGadget command. If the string gadget specified
by GadgetList# and Id is already displayed, you will also need to
exeucte ReDraw to display the change.
```

1.33 GADGETSLIB

```
Statement: ButtonGroup
------------------------------------------------------------------------------
Modes  :
Syntax : ButtonGroup Group

ButtonGroup allows you to determine which 'group' a number of button
type gadgets belong to. Following the execution of ButtonGroup, any
button gadgets created will be identified as belonging to the specified
group. The upshot of all this is that button gadgets are only mutually
exclusive to other button gadgets within the same group.

'Group' must be a positive number greater than 0. Any button gadgets
created before a 'ButtonGroup' command is executed will belong to group
1.
```

1.34 GADGETSLIB

```
Function: GadgetStatus
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetStatus (GadgetList#,Id)

GadgetStatus may be used to determine the status of the specified
gadget. In the case of 'toggle' type gadget, GadgetStatus will return
true (-1) if the gadget is currently on, or false (0) if the gadget is
currently off.

In the case of a cycling text gadget, GadgetStatus will return a value
of 1 or greater representing the currently displayed text within the
gadget.
```

1.35 GADGETSLIB

```
Function: ButtonId
------------------------------------------------------------------------------
Modes  :
Syntax : ButtonId (GadgetList#,ButtonGroup)

ButtonId may be used to determine which gadget within a group of button
type gadgets is currently selected. The value returned will be the
GadgetId of the button gadget currently selected.
```

1.36 GADGETSLIB

```
Function: Enable
--------------------------------------------------------------------------------
Modes  :
Syntax : Enable (GadgetList#,Id)

A gadget when disabled is covered by a "mesh" and can not be accessed
by the user. The commands Enable and Disable allow the programmer to
access this feature of Intuition.
```

1.37 GADGETSLIB

```
Function: Disable
--------------------------------------------------------------------------------
Modes  :
Syntax : Disable (GadgetList#,Id)

A gadget when disabled is covered by a "mesh" and can not be accessed
by the user. The commands Enable and Disable allow the programmer to
access this feature of Intuition.
```

1.38 GADGETSLIB

```
Function: SetGadgetStatus
--------------------------------------------------------------------------------
Modes  :
Syntax : SetGadgetStatus (GadgetList#,Id,Value)

SetGadgetStatus is used to set a cycling text gadget to a particular
value, once set ReDraw should be used to refresh the gadget to reflect
it's new value.

NEW GADGETS EXAMPLE:

;
; new gadget types
;
WBStartup:FindScreen 0   ;open on workbench
TextGadget 0,32,14,0,0,"CYCLE 1|CYCLE 2|CYCLE 3"

ButtonGroup 1 ;first group of radio buttons follows
For i=1 To 5
  TextGadget 0,32,14+i*14,512,i,"CHANNEL #"+Str$(i)
Next

ButtonGroup 2 ;second group of radio buttons follows
For i=6 To 10
  TextGadget 0,32,14+i*14,512,i,"BAND #"+Str$(i)
Next
```

```
Window 0,20,20,160,180,$1008,"GADGET TEST",1,2,0

Repeat        ;wait until close window gadget hit
  ev.l=WaitEvent
Until ev=$200
```

1.39 GADGETSLIB

```
                .-------------------------------------------------------------------------
|                                  GADGETSLIB                                           |
`-------------------------------------------------------------------------------------'
```

Overview

Command Index

ActivateString

ArrowGadget

BorderPens

Borders

ButtonGroup

ButtonId

ClearString

Disable

Enable

GadgetJam

GadgetList

GadgetPens

GadgetStatus

Gap1

Gap3

GetPropCurrent

HPropBody

HPropPot

MoveProp

PropGadget

Redraw

ResetString

SelectMode

SetGadgetStatus

SetHProp

SetPropInfo

SetString

SetVProp

ShapeGadget

StringGadget

StringText$

TextGadget

Toggle

VPropBody