

wizard

COLLABORATORS

	<i>TITLE :</i> wizard		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 19, 2024	

REVISION HISTORY

<i>NUMBER</i>	<i>DATE</i>	<i>DESCRIPTION</i>	<i>NAME</i>

Contents

1	wizard	1
1.1	WIZARD.DOC	1
1.2	wizard.library/WZ_AllocWindowHandleA	1
1.3	wizard.library/WZ_CloseSurface	3
1.4	wizard.library/WZ_CloseWindow	4
1.5	wizard.library/WZ_CreateWindowObjA	4
1.6	wizard.library/WZ_DrawVImageA	5
1.7	wizard.library/WZ_EasyRequestArgs	12
1.8	wizard.library/WZ_FreeWindowHandle	12
1.9	wizard.library/WZ_GadgetConfig	13
1.10	wizard.library/WZ_GadgetHelp	13
1.11	wizard.library/WZ_GadgetHelpMsg	14
1.12	wizard.library/WZ_GadgetKey	15
1.13	wizard.library/WZ_GetNode	16
1.14	wizard.library/WZ_InitEasyStruct	16
1.15	wizard.library/WZ_InitNodeA	17
1.16	wizard.library/WZ_InitNodeEntryA	17
1.17	wizard.library/WZ_ListCount	18
1.18	wizard.library/WZ_LockWindow	19
1.19	wizard.library/WZ_LockWindows	19
1.20	wizard.library/WZ_MenuConfig	20
1.21	wizard.library/WZ_MenuHelp	20
1.22	wizard.library/WZ_NewObjectA	21
1.23	wizard.library/WZ_ObjectID	32
1.24	wizard.library/WZ_OpenSurfaceA	33
1.25	wizard.library/WZ_OpenWindowA()	34
1.26	wizard.library/WZ_SnapShotA()	34
1.27	wizard.library/WZ_UnlockWindow	35
1.28	wizard.library/WZ_UnlockWindows	36

Chapter 1

wizard

1.1 WIZARD.DOC

Referenz der ANSI-C und C++ Bibliothek zu StormC

© 1996 by HAAGE & PARTNER Computer GmbH

```
WZ_AllocWindowHandleA ()
WZ_CloseSurface ()
WZ_CloseWindow ()
WZ_CreateWindowObjA ()
WZ_DrawVImageA ()
WZ_EasyRequestArgs ()
WZ_FreeWindowHandle ()
WZ_GadgetConfig ()
WZ_GadgetHelp ()
WZ_GadgetHelpMsg ()
WZ_GadgetKey ()
WZ_GetNode ()
WZ_InitEasyStruct ()
WZ_InitNodeA ()
WZ_InitNodeEntryA ()
WZ_ListCount ()
WZ_LockWindow ()
WZ_LockWindows ()
WZ_MenuConfig ()
WZ_MenuHelp ()
WZ_NewObjectA ()
WZ_ObjectID ()
WZ_OpenSurfaceA ()
WZ_OpenWindowA ()
WZ_SnapShotA ()
WZ_UnlockWindow ()
WZ_UnlockWindows ()
```

1.2 wizard.library/WZ_AllocWindowHandleA

NAME

WZ_AllocWindowHandleA -- WizardWindowHandle anlegen
 WZ_AllocWindowHandle -- variable Parameterübergabe für Hochsprachen

SYNOPSIS

```
winhandle = WZ_AllocWindowHandleA(screen, user_sizeof, surface, tags )
D0                D0          D1          A0          A1
```

```
struct WizardWindowHandle *WZ_AllocWindowHandleA(
  struct Screen *, ULONG, APTR, struct TagItem *);
```

```
winhandle = WZ_AllocWindowHandle(
  screen, user_sizeof, surface, firstTag, ... )
```

```
struct WizardWindowHandle *WZ_AllocWindowHandle(
  struct Screen *, ULONG, APTR, Tag, ...);
```

FUNCTION

Belegt den Speicher für einen WizardWindowHandle. Gleichzeitig wird dieser auch initialisiert.

WARNING

Diese Struktur ist intern noch mit zusätzlichen Feldern definiert.

INPUTS

screen - ein Zeiger auf die Screen-Struktur, auf dem das Fenster später erscheinen soll

user_sizeof - die Größe in Bytes die eine private Struktur haben soll, mit der man eigene Daten zum Fenster verwalten will, siehe WZ_WindowUserStruct().

surface - der Returnwert von WZ_OpenSurface()

tags

WWH_StackSize, ULONG

Größe des Stacks, der für den Layoutvorgang bereitgestellt werden soll. (Vorgabe 8192 Bytes).

RESULT

winhandle - Zeiger auf eine WizardWindowHandle-Struktur oder Null im Fehlerfall.

Window - ein Zeiger auf die Fensterstruktur von Intuition, falls das Fenster geöffnet ist, ansonsten eine Null.

MenuStrip - Zeiger auf den MenuStrip dieses Fensters oder Null, wenn kein Menu existiert.

DrawInfo - wird beim Anlegen der Struktur von der Library ausgefüllt.

VisualInfo - wird beim Anlegen dieser Struktur von der Library ausgefüllt.

ScreenTitle - in diesem Feld sollte der Screenshot des Fensters eingetragen sein, welcher beim Öffnen des Fensters mittels WZ_OpenWindowTags() benutzt wird. Dieses Feld wird beim Anlegen der Struktur mit -1 ausgefüllt. Der Aufruf von WZ_CreateWindowObj() setzt diesen Wert auf einen sinnvollen Wert.

Objects - Eine Minlist-Struktur, in der BOOPSI-Objekte verkettet

sind, welche beim Freigeben dieser Struktur mittels `intuition.library/DisposeObject()` ebenfalls entfernt werden sollen. Das Eintragen eigener Objekte ist möglich.

`RootGadget` - ein Zeiger auf das Gruppengadget, welches innerhalb des Fensters seine Mitglieder plaziert und für diese auch den Layoutvorgang initialisiert.
Ein `RootGadget` existiert IMMER, wenn alle Objekte die zu einem Fenster gehören, angelegt wurden, Gleichzeitig ist dieses Gadget der Anfang der gesamten Gadgetliste!

`RootTopGadget`

`RootLeftGadget`

`RootBottomGadget`

`RootRightGadget`

Diese Felder werden erst in späteren Versionen der Wizard-Library eine Bedeutung bekommen. Im Moment sind diese Felder auf Null gesetzt.

`UserStruct` - ein Zeiger auf eine private Struktur, welche mit Nullwerten vorinitialisiert ist. War der Funktionsaufruf mit dem Parameter `user_sizeof = Null`, dann ist dieser Zeiger mit Null initialisiert und darf nicht als solcher benutzt werden.
Das verwenden dieses Feldes zum Selbsteintragen einer Struktur oder eines anderen Wertes ist in einem solchen Fall aber gestattet !

SEE ALSO

`WZ_CreateWindowObjA()`, `WZ_OpenWindowA()`, `WZ_CloseWindow()`,
`WZ_FreeWindowHandle()`

1.3 wizard.library/WZ_CloseSurface

NAME

`WZ_CloseSurface` -- Oberflächenbeschreibung abmelden

SYNOPSIS

`WZ_CloseSurface(surface)`
A0

`VOID WZ_CloseSurface(APTR);`

FUNCTION

Diese Funktion meldet eine Oberflächenbeschreibung ab und gibt den belegten Speicher wieder frei.

WARNING

Sind noch `WindowHandles` oder geöffnete Fenster vorhanden, werden diese geschlossen und entfernt. Alle Ressourcen sind nicht mehr ansprechbar.

INPUTS

`surface` - der Returnwert von `WZ_OpenSurface()`

SEE ALSO

```
WZ_OpenSurfaceA(),WZ_SnapShotA()
```

1.4 wizard.library/WZ_CloseWindow

NAME

```
WZ_CloseWindow -- Fenster schliessen
```

SYNOPSIS

```
WZ_CloseWindow(winhandle)
                A0
```

```
VOID WZ_CloseWindow(struct WizardWindowHandle *);
```

FUNCTION

schliesst ein Fenster ähnlich dem Intuition-Aufruf.

WARNING

Das Benutzen der Intuition-Funktion ist nicht erlaubt.

INPUTS

winhandle - WizardWindowHandle von WZ_AllocWindowHandle()

SEE ALSO

WZ_OpenWindowA()

1.5 wizard.library/WZ_CreateWindowObjA

NAME

```
WZ_CreateWindowObjA -- Fenster-Objecte anlegen
WZ_CreateWindowObj  -- variable Parameterübergabe für Hochsprachen-
                    programmierer
```

SYNOPSIS

```
newwin = WZ_CreateWindowObjA(winhandle, id, tags)
D0                      A0                      D0  A1
```

```
struct NewWindow *WZ_CreateWindowObjA
    (ULONG, struct WizardWindowHandle *,struct TagItem *);
```

```
newwin = WZ_CreateWindowObj(winhandle, id, firstTag, ... )
```

```
struct NewWindow *WZ_CreateWindowObj
    (ULONG, struct WizardWindowHandle *,Tag, ... );
```

FUNCTION

Mit diesem Funktionsaufruf werden alle für das Fenster wichtigen Objecte (Gadgets, Menu, Notifyobjecte) angelegt.

WARNING

Die Beschreibung der Objecte kann NICHT kontrolliert werden.

INPUTS

id - Nummer des anzulegenden Windows
 winhandle - Zeiger auf einen von WZ-AllowWindowHandle()
 besorgten WizardWindowHandle
 tags - TagItems, um z.B. das GadgetArray zu übergeben

 WWH_GadgetArray - ein Zeiger auf ein Array, in dem die erzeugten
 Gadgets nach der GadgetID abgelegt werden.
 Diese Tag wird immer verlangt !
 WWH_GadgetArraySize - Größe des Arrays in Bytes zur Sicherheit.
 dieses Tag darf wegelassen werden.
 WWH_PreviousGadget - Gadget, hinter dem die zu erzeugenden Gadgets
 eingehangen werden sollen
 WWH_StringHook - Zeiger auf eine StringHook-Struktur, welche an
 alle Stringgadgets in dem Fenster übergeben
 werden soll. Genauere Information finden Sie
 bei der Beschreibung der "strgclass" !

RESULT

newwin - Zeiger auf eine initialisierte NewWindow-Struktur
 oder Null im Fehlerfall. Diese ist wie folgt
 vorinitialisiert:

 LeftEdge,
 TopEdge,
 Width,
 Height - Werte die in der Oberflächenbeschreibung abge-
 speichert sind. Diese lassen sich wieder mit
 WZ_SnapShotA() fixieren.

 DetailPen,
 BlockPen - mit ~0 vorbelegt
 IDCMPFlags - stammen aus der Oberflächenbeschreibung
 Flags - ebenfalls aus der Oberflächenbeschreibung
 FirstGadget - RootGadget des Fensters
 CheckMark - Null
 Title - String der in StormWizard festgelegt wurde.
 Screen - Screenstruktur, welche bei WZ_AllocWindowHandle()
 übergeben wurde.

 BitMap - Null
 MinWidth,
 MinHeight - minimale Dimension, die das Fenster haben
 darf, diese Werte werden jedesmal aufs Neue
 berechnet.

 MaxWidth,
 MaxHeight - diese Werte werden mit ~0 initialisiert.
 Type - mit dem Wert von CUSTONSCREEN.

SEE ALSO

WZ_AllocWindowHandle(), WZ_FreeWindowHandle(), WZ_OpenWindow(),
 WZ_CloseWindow()

1.6 wizard.library/WZ_DrawVImageA

NAME

WZ_DrawVImageA -- Vektorgrafik zeichnen
 WZ_DrawVImage -- für Hochsprachenprogrammierer mit variabler
 Parameterübergabe

SYNOPSIS

```

success = WZ_DrawVImageA(vimage, x, y, w, h, type, rp, drinfo, tags)
D0                A0                D0 D1 D2 D3 D4                D5 D6                A1

```

```

BOOL WZ_DrawVImageA(
    struct WizardVImage *,
    WORD, WORD, WORD, WORD, UWORD, struct RastPort *,
    struct DrawInfo *, struct TagItem *);

```

```

success = WZ_DrawVImage
    (vimage, x, y, w, h, type, rp, drinfo, firstTag, ... )

```

```

BOOL WZ_DrawVImage(
    struct WizardVImage *,
    WORD, WORD, WORD, WORD, UWORD, struct RastPort *,
    struct DrawInfo *, Tag, ... );

```

FUNCTION

Diese Funktion zeichnet eine Vektorgrafik mit den angegebenen Parametern. Es sind damit Turtle-Graphics realisierbar.

WARNING

Teilweise sind in der Definition einer Vektorgrafik Verweise auf andere Vektorgrafiken vorhanden. Dabei ruft sich diese Funktion selbst auf. Der Stack wird dabei gnadenlos genutzt !

INPUTS

vimage - Zeiger die WizardVImage-Struktur die wie folgt aufgebaut ist :

flags

- WVIF_MinWidth - wenn die zum Zeichnen übergebene Breite kleiner ist als die in der WizardVImage-Struktur übergebene, dann wird im Normalfall nicht gezeichnet. Dieses Flag zwingt in einem solchen Fall die übergeben Breite auf diesen Mindestwert, dabei wird die X-Position zur ← Hälfte der Differenz nach links korrigiert.
- WVIF_MinHeight - wenn die übergebene Höhe mindestens den Wert aus WizardVImage->MinHeight haben soll. Eine Korrektur nach oben findet in einem solchen Fall ebenfalls statt. Sollte dieses Flag nicht gesetzt sein und die Höhe wird unterschritten, dann wird die Funktion WZ_DrawVImage() nicht durchgeführt.
- WVIF_AreaInit - dieses Flag besitzt zur Zeit noch keine Bedeutung.
- WVIF_Recursion - zeigt an, das diese Vektorgrafik sich selbst aufruft und dabei der Wert, der in VImage->Counter steht, bei jedem Aufruf dieser Grafik um eins ← erniedrigt wird. Wird der Wert Null erreicht, dann wird die Rekursion abgebrochen. Dieses Flag darf nur verwendet werden, wenn die Vektorgrafik nur einem einzigen Task zur Verfügung steht.

Counter - sollte das Flag WVIF_Recursion gesetzt sein, dann wird hiermit die maximale Tiefe bei einer Rekursion ← angegeben.

- Möchten Sie das die Grafik nach dem Funktionsaufruf von WZ_DrawVImage() vom Task aus noch ein einziges mal aufgerufen wird, dann geben Sie hier eine 2 an.
- MinWidth - Geben Sie hier die Mindestbreit an, mit der ein Funktionsaufruf von WZ_DrawVImage() erfolgen muß. siehe Flag WVIF_MinWidth
- MinHeight - Geben Sie die Mindesthöhe für einen Aufruf von WZ_DrawVImage() an. siehe Flag WVIF_MinHeight
- RelCoords - Möchten Sie eigene Bezugspunkte angeben, dann geben Sie hier einen Zeiger auf eine UWORD-Tabelle an. dessen 1.Word dem relativen X und dessen 2.Word dem relativen Y entspricht. Terminieren Sie diese Tabelle mit zwei aufeinanderfolgenden Nullen. Der 1.Eintrag in dieser Tabelle entspricht dem ↔ Bezugspunkt mit der Nummer 4 ! Die einzutragenden X und Y Werte müssen sich im Bereich von 0 bis 65535 befinden. Wobei eine Null links bzw. oben entspricht. Vordefinierte Bezugspunkte sind : 0 - links und oben, 1 - rechts und oben, 2 - rechts und unten sowie 3 - links und unten. Wenn Sie diesen Zeiger mit Null ausfüllen, dann dürfen Sie nur die Bezugspunkte 0 bis 3 nutzen !
- Images - hier können beliebig viele aufeinander folgende Zeiger eintragen werden. Der übergebene type bestimmt dann, welcher Zeiger und damit welche Zeichendefinitionen benutzt wird. Wurde type mit 0 angegeben, dann wird direkt der Zeiger aus WizardVImage->Images genommen, ansonsten einer der darauf folgenden. Gleichzeitig ist damit bedingt, das die Länge der Struktur NICHT festgelegt werden kann.
- x - linke Position der zu zeichnenden Grafik
- y - obere Position
- w - Breite
- h - Höhe
- type - Type der zu zeichnenden Grafik
- rp - Zeiger auf den zu verwendenden RastPort
- drinfo - Zeiger auf eine DrawInfo-Struktur, wenn DrawInfo-Farben benutzt werden sollen oder Textfunktionen im Image vorkommen.
- tags - folgende Tags sind definiert:
- WVIA_Text (V37), STRPTR
Adresse eines mit NullByte oder Return abgeschlossenen Strings, für Text-Kommandos.
- WVIA_TextFont (V37), struct TextFont *
TextFont-Struktur, der bei Textfunktionen genommen ↔ werden soll. Vorgabe ist der DrawInfo-Zeichensatz.
- WVIA_TextPlace (V37), ULONG
beschreibt die Textausgabe mittels der Konstanten: WZRDPLACE_LEFT, WZRDPLACE_CENTER, WZRDPLACE_RIGHT für links, mittig, rechts. (WZRDPLACE_CENTER)
- WVIA_TextPen (V37), UWORD
dieses Tag hat zur Zeit noch keine Funktion, setzen Sie die Farbe vorher mit einem entsprechendem Kommando ↔ vorher.

WVIA_TextStyles (V37), ULONG
 dieses Tag hat zur Zeit noch keine Funktion, setzen Sie den Style mit einem entsprechendem Kommando vorher.

WVIA_TextHighlights (V37), ULONG
 dieses Tag hat zur Zeit noch keine Funktion.

WVIA_TextImages (V37), BOOL
 TRUE, um das Zeichnen von Images mit der Dimension von einer TextAusgabe zuzulassen. FALSE, wenn diese \leftrightarrow ImageArt ausgeschlossen werden soll. (TRUE)

WVIA_TagImage (V37), struct WizardVImage *
 Ermöglicht die Angabe einer variablen Vektorimage-Definition.

WVIA_TagImageCode (V37), UWORD
 Beschreibt den Type, mit der das in WVIA_TagImage angegebene VImage gezeichnet werden soll. (0)

WVIA_ImageCode (V37), UWORD
 Falls in einem festen Verweis auf ein anderes VImage der zu zeichnende Type mit -1 festgelegt wurde, dann kann hier der richtige Type für diese VImage angegeben werden. Fehlt trotzdem diese Tag, wird der Imageverweis nicht bearbeitet. (-1)

WVIA_Color0-7 (V37), UWORD
 Diese Tags enthalten Farben, die mittels des Kommandos WVICMD_TAGCOLOR angesprochen werden können. Es können auch DrawInfo-Farben gewählt werden.

WVIA_TPoint0-7 (V37), struct TPoint
 Hier können von außen berechnete Positionen in die VImage-Definition eingebracht werden. (X<<16+Y)

WVIA_AreaPtrn (V37), APTR
 Ein Zeiger auf ein zu benutzendes Raster für alle \leftrightarrow folgenden flächenfüllenden Kommandos (WVICMD_RECTFILL). Die Höhe wird direkt beim Kommandoaufruf WVICMD_SETAFPT angegeben.

WVIA_TmpRas (V37), struct TmpRas *
 Hier muß eine initialisierte TmpRas-Struktur angegeben werden, um mittels WVICMD_AREAINIT im RastPort fixiert \leftrightarrow zu werden.

WVIA_BitMapWidth (V37), UWORD
 Die Breite der in speziellen Tags angegeben Bitmap. (0 \leftrightarrow)

WVIA_BitMapHeight (V37), UWORD
 Die Höhe der in speziellen Tags angegeben Bitmap. (0)

WVIA_BitMap0-7 (V37), struct BitMap *
 Bitmaps für DrawVImage-Kommandos.

WVIA_PureText (V37), BOOL
 TRUE, wenn ein Unterstrich das Unterstreichen des \leftrightarrow folgenden Buchstabens kennzeichnen soll, sonst FALSE. (Vorgabe TRUE).

Kommandos für VImage-Definitionen:

Alle Kommandos sind vom Type LONG !

Auch der Platzverbrauch aller folgenden Parameter ist identisch mit der Größe LONG !!!

WVICMD_END :
Signalisiert das Ende einer Definition.

WVICMD_COLOR : Pen;
Setzt den APen des Rastports auf den folgenden Pen, dabei sind DrawInfo-Farben zugelassen. Bereich 0 - 255 oder WZRD_... -Farben

WVICMD_COLOR2 : Pen(;
Setzt den BPen des Rastports auf den folgenden Pen, dabei sind DrawInfo-Farben zugelassen. Wird nur selten benötigt.

WVICMD_MOVE : BPkt, XOffset, YOffset;
Bewegt den Grafikkursor auf den Bezugspunkt, um die in XOffset und YOffset angegebenen Werte verschoben.

WVICMD_DRAW : BPkt, XOffset, YOffset, Mask;
Zieht eine Linie von der aktuellen Grafikkursorposition zum angegebenen Bezugspunkt, dabei werden XOffset und YOffset zum Bezugspunkt dazu addiert. Mit Maske wird das Linienmuster definiert. Der eingestellte APen wird benutzt.

WVICMD_RECTFILL : BPkt, XOffset, YOffset;
Zeichnet von der aktuellen Position des Cursors zum angegebenen Bezugspunkt eine ausgefüllte Fläche mit dem APen. Dabei muß der Cursor bereits auf der linken oberen Ecke des zu zeichnenden Rechtecks stehen. Der durch den Bezugspunkt definierte Grafikkursor muß die rechte untere Ecke kennzeichnen.

WVICMD_WRITEPIXEL : BPkt, XOffset, YOffset;
Zeichnet an den um XOffset und YOffset verschobenen Bezugspunkt einen farbigen Punkt, in der APen-Farbe.

WVICMD_IMAGE : BPkt, XOffset, YOffset, Type, VImage;
Führt recursiv einen VImage-Aufruf durch. Dabei muß die linke obere Ecke vorher mit dem Kommando WVICMD_MOVE festgelegt worden sein. Die rechte untere Ecke bestimmt sich aus dem Bezugspunkt und dem X- sowie YOffset. Type gibt für den Aufruf den zu benutzenden Type an, der eine -1 enthalten muß, falls der eigentliche Type dem Tag WVIA_ImageCode entommen werden soll. VImage ist ein Zeiger auf die Vektorgrafik, die hier gezeichnet werden soll.

WVICMD_TEXT :
Geben Sie für dieses Kommando immer mindestens das Tag WVIA_Text an ! Mit dieser Funktion wird der angegebene Text an die aktuelle Position gezeichnet. Setzen Sie vorher also Farbe, Schnitt und Position.

WVICMD_SETDRMD :
Hiermit wird der DrawMode des RastPort umgestellt. Bei Beginn des DrawImageaufrufes ist dieser immer RP_JAM1 !

WVICMD_TEXTIMAGE : HBorder, VBorder, Type, VImage;

Wenn ein Text ausgegeben wird, dann ist dieser an einer Position und hat eine bestimmte Ausdehnung. Hiermit wird diese Position für einen Imageaufruf genommen. Dabei wird aber nach links und rechts die Ausdehnung um den Wert HBorder vergrößert. Dies gilt auch für die vertikale Ausdehnung durch \leftrightarrow VBorder.

Der Imageaufruf erfolgt mit dem angegebenen Type und dem angegebenen Vektorimage.

Diese Art von Images kann mit dem Tag WVIA_TextImages \leftrightarrow für das aktuelle Vektorimage abgeschaltet werden.

WVICMD_TEXTMOVE : FontSizeFak, BaseFak, LengthFak, XOffset, YOffset;
 Diese Funktion verschiebt den Cursor von der aktuellen Position um einen bestimmten Wert. Dabei wird die Länge der möglichen Textausgabe durch LengthFak dividiert und zur aktuellen X-Position hinzuaddiert. Den Wert TextFont->Baseline dividiert er durch BasekFak und verschiebt den Cursor in der Y-Richtung. Die Zeichensatzhöhe dividiert er durch FontSizeFak und verschiebt den Cursor ebenfalls in der Y-Richtung. Bei allen Parametern dieses Kommandos kann eine Null angegeben werden, um ihn wirkungslos zu machen.

WVICMD_TAGCOLOR : Col;
 Col enthält die Nummer des WVIA_Color - Tags, dessen Farbe gesetzt werden soll. Um die Farbe des Tags WVIA_Color1 setzten zu können muß dieser Parameter eine Eins enthalten.

WVICMD_TEXTPLACE: BPKT, XOffset, YOffset;
 Um komfortabel TextPositionen angeben zu können, wurde diese Funktion geschrieben. Hier wird der TextCursor so plaziert, das der Text vertikal zentriert in einem definierten Rechteck ausgegeben wird. Mit dem Tag WVIA_TextPlace kann der Text links, mittig oder rechts innerhalb eines Rechtecks angegeben werden. Als linke obere Ecke des unsichtbaren Rechtecks gilt die aktuelle Grafikkursorposition. Mittels Bezugspunkt und den beiden Offsets wird die untere rechte Ecke angegeben. Eine Ausgabe in den RastPort erfolgt nicht.

WVICMD_SETAFPT : AreaPtSz, AreaPtrn;
 Um für Flächenfüllkommandos beliebig Raster verwenden zu können, wird diese Funktion eingesetzt. Wurde der Wert AreaPtrn mit einer -1L angegeben, dann muß das Tag WVIA_AreaPtrn die richtige Rasteradresse enthalten. Die Dimension des Raster errechnet sich aus Zwei hoch AreaPtSz (2^{AreaPtSz}). Jetzt steht dieses Raster allen Kommandos zu Verfügung.

WVICMD_SNAPCURSOR : BPkt;
 Schreibt die aktuelle Grafikkursorposition an den angegebenen Bezugspunkt zurück. Möglich sind diese im Bereich von (0 - 15).

WVICMD_SNAPX : BPkt;
 Schreibt die aktuelle X-Position des Grafikkursor in den Bezugspunkt zurück.

WVICMD_SNAPY : BPkt;
 Schreibt die aktuelle Y-Position des Grafikkursor in den Bezugspunkt zurück.

WVICMD_TAGMOVE : TPoint;
Setzt den Grafikkursor auf eine Position, deren Werte aus einem WVIA_TPoint-Tag entnommen werden. Dabei bestimmt TPoint die Nummer des Tags.

WVICMD_TAGIMAGE : BPKT, XOffset, YOffset, Type, VImageTags;
Position und Dimension errechnen sich wie bei WVICMD_IMAGE. Sehen Sie also dort noch einmal nach. Als Imageadresse wird aber der im Tag WVIA_TagImage angegebene Wert genommen. VImageTags enthält dabei die Tags für den WZ_DrawVImage() - Aufruf. Wird dieser Wert mit -1L angegeben, dann werden die aktuellen Tags dafür genommen. Der Type für den Funktionsaufruf wird dem Tag WVIA_TagImageCode entnommen.

WVICMD_BITMAP_TO_RP : BPKT, XOffset, YOffset, Map;
Damit lassen sich Bitmaps in den RastPort blitzen. Bpkt, XOffset und YOffset ergibt die linke obere Ecke, an die diese Bitmap kopiert wird. Map enthält die Nummer des Tags WVIA_BitMap. Die Ausdehnung der Bitmap muß mit den Tags WVIA_BitMapWidth und WVIA_BitMapHeight angegeben werden.

WVICMD_FILLBORDER : BPKT, XOffset, YOffset, HBorder, VBorder;
Diese Funktion zeichnet einen Rahmen mit der aktuellen APen. Gezeichnet wird von der linken oberen Ecke der gesamten Vektorgrafik bis zu deren unteren rechten Ecke. Dabei wird ein Bereich von der aktuellen Grafikkursorposition bis zu dem angegebenen Bezugspunkt nicht überzeichnet. Bezugspunkt, XOffset und YOffset werden zur Bestimmung der rechten unteren Ecke genommen, deren Inhalt NICHT überzeichnet werden soll. Mit HBorder und VBorder kann die Dimension der Vektorgrafikdimension innerhalb dieser Routine verringert werden.

WVICMD_Beep :
ruft die Funktion intuition.library/DisplayBeep mit dem Parameter Screen = NULL auf.

WVICMD_AREAINIT :
Initialisiert den RastPort, um Areafunktionen nutzen zu können. Das Tag WVIA_TmpRas muß einen Zeiger auf eine initialisierte TmpRas-Struktur enthalten.

WVICMD_AREAMOVE : BPKT, XOffset, YOffset;
Damit wird der Grafikkursor für Areafunktionen auf die Position des Bezugspunktes gesetzt. Dabei werden XOffset und YOffset zur Position addiert. Hiernach dürfen 3 Aufrufe des Kommandos WVICMD_AREADRAW folgen.

WVICMD_AREADRAW : BPKT, XOffset, YOffset;
Zieht eine Begrenzungslinie zum angegebenen Bezugspunkt. Dabei werden XOffset und YOffset zur Position addiert. Diese Funktion darf maximal 3mal ↔ hintereinander aufgerufen werden !

WVICMD_AREAEND :
Zeichnet ein ausgefülltes Image in den RastPort. Dabei werden APen und das aktuelle Raster ↔ berücksichtigt. Danach dürfen AreaMove und AreaDraw wieder erfolgen.

RESULT
success - TRUE in jedem Falle

1.7 wizard.library/WZ_EasyRequestArgs

NAME
WZ_EasyRequestArgs -- einfachen Requester handhaben

SYNOPSIS
result = WZ_EasyRequestArgs(surface, window, id, args)
D0 A0 A1 D0 A2

LONG WZ_EasyRequestArgs(APTR, struct Window *, ULONG, void *);

FUNCTION
Ähnlich der EasyRequestArgsA-Funktion der intuition.library.
Dabei wird automatisch die Tastatur überwacht.

WARNING
Die übergebenen Argumente können nicht überprüft werden.
Nur eine Gadget-Anwahl mit der Maus oder der
Tastatur kann diesen Requester beenden. Der Userport wird nicht
gegen Eingaben gesperrt, also Locken Sie das Fenster irgendwie
selber!

INPUTS
surface - Returnwert von WZ_OpenSurface()
window - Adresse der Window-Struktur, dessen Fenster gesperrt
 werden soll
id - Nr des Requesters aus dem StormWizard
args - Adresse der Argumentenliste

RESULT
result - Nummer des Gadgets, der zum verlassen des Requesters
 geführt hat oder -1, wenn der Requester aus
 Speicherplatzgründen nicht dargestellt werden konnte.

SEE ALSO
WZ_InitEasyStruct ()

1.8 wizard.library/WZ_FreeWindowHandle

NAME
WZ_FreeWindowHandle -- WizardWindowHandle freigeben

SYNOPSIS
WZ_FreeWindowHandle(winhandle)

VOID WZ_FreeWindow(struct WizardWindowHandle *);

FUNCTION

Gibt den Speicher und damit alle Objecte die zu einem Fenster gehören frei. Ein geöffnetes Fenster wird mit diesem Funktionsaufruf gleichzeitig geschlossen.

INPUTS

winhandle - WizardWindowHandle von WZ_AllocWindowHandle

SEE ALSO

WZ_AllocWindowHandleA()

1.9 wizard.library/WZ_GadgetConfig

NAME

WZ_GadgetConfig -- Gadgetconfigstring holen

SYNOPSIS

```
string = WZ_GadgetConfig(winhandle, gadget)
D0                      A0          A1
```

```
STRPTR ,WZ_GadgetConfig( struct WizardWindowHandle *, struct Gadget *);
```

FUNCTION

Der Configurationsstring zu einem Gadget kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizard eingegeben.

INPUTS

winhandle - Zeiger auf ein WizardWindowHandle
gadget - Zeiger auf ein Wizard-Gadget

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

WZ_MenuConfig(), WZ_GadgetHelp()

1.10 wizard.library/WZ_GadgetHelp

NAME

WZ_GadgetHelp -- Gadgethilfestring holen

SYNOPSIS

```
string = WZ_GadgetHelp(winhandle, iaddress)
D0                      A0          A1
```

```
STRPTR WZ_GadgetHelp(struct WizardWindowHandle *, APTR);
```

FUNCTION

Der Hilfestring zu einem Gadget oder Fenster kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizard eingegeben.

Wurde die Oberflächenbeschreibung mit dem Tag SFH_Catalog

bereitgestellt, dann wird der übersetzte String geliefert.

WARNING

Ist das übergebene Gadget nicht als Window im WizardWindowHandle eingetragen, wird ohne Überprüfung davon ausgegangen, daß es sich um ein Gadget der Wizard.library handelt.

INPUTS

winhandle - Zeiger ein WizardWindowHandle
iaddress - Zeiger auf ein Wizardgadget oder ein geöffnetes Fenster

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

WZ_MenuHelp(), WZ_GadgetConfig()

1.11 wizard.library/WZ_GadgetHelpMsg

NAME

WZ_GadgetHelpMsg -- Gadgethelpmessage holen

SYNOPSIS

```

succes = WZ_GadgetHelpMsg(winhandle, winaddress, iaddress, mousex, mousey ←
, flags)
D0          A0          A1          A2          D0          D1 ←
          D2

```

```

BOOL WZ_GadgetHelpMsg(struct WizardWindowHandle *,
struct WizardWindowHandle **,
APTR *, WORD, WORD, UWORD);

```

FUNCTION

Um unter V37 und V38 des Amiga-OS auch GadgetHelpMessages empfangen zu können, wurde diese Funktion geschrieben. Wahlweise kann diese Funktion ab V39 zu einer Dummyfunktion werden, um diese Nachrichten von Intuition zu bekommen.

WARNING

In späteren Version kann IAddress auch Objecte liefern, die zwar unter der Maus liegen und auch zur Surface gehören, nicht aber im aktiven Fenster liegen. Das schliesst Fenster natürlich selbst auch ein.

INPUTS

winhandle - Zeiger ein WizardWindowHandle, von dem ein MausMove gemeldet wurde
winaddress - Zeiger auf einen 4Byte großen Speicherbereich, der von der Funktion genutzt wird, um einen 2.Wert zurückzuliefern.
iaddress - Zeiger auf einen 4Byte großen Speicherbereich, der von der Funktion zur Rückgabe eines 3.Wertes genutzt wird.
mousex - X-Pos. der Maus

mousey - Y-Pos. der Maus
 flags - folgende sind definiert

 WGHF_IgnoreOS - um auch unter 3.0 und höher diese Funktion nutzen
 zu können
 WGHF_FullControl - um auch Nachrichten von anderen Fenstern
 empfangen zu können. Dieses Flag funktioniert
 unter V37.0 noch nicht !

RESULT

success - hat sich die Position der Maus geändert und damit
 auch das darunterliegende Object, dann ist dieser
 Wert TRUE, andernfalls FALSE.
 winaddress - Wenn success gleich TRUE ist, dann steht hier ein
 Zeiger auf das WizardWindowHandle, welches
 sich unter der Maus befindet.
 iaddress - Wenn success gleich TRUE ist, dann steht hier ein
 Zeiger auf das darunterliegende Object.
 Er muß dann genauso wie das entsprechende Feld
 einer IntuitMessage bei einer IDCMP_GADGETHELP-Msg
 des Betriebssystems bewertet werden.

SEE ALSO

WZ_GadgetHelp(), WZ_MenuHelp(),

1.12 wizard.library/WZ_GadgetKey

NAME

WZ_GadgetKeyA -- Gadgets über Tastendruck informieren
 WZ_GadgetKey -- Gadgets über Tastendruck informieren

SYNOPSIS

```

success = WZ_GadgetKeyA(winhandle, code, qualifier, tags)
D0                    A0            D0    D1            A1
  
```

```

BOOL WZ_GadgetKeyA(
    struct WizardWindowHandle *, ULONG, ULONG,
    struct TagItem *);
  
```

```

success = WZ_GadgetKey(
    winhandle, code, qualifier, firstTag, ... )
  
```

```

BOOL WZ_GadgetKey(
    struct WizardWindowHandle *, ULONG, ULONG, APTR, Tag, ...);
  
```

FUNCTION

Um die Gadgets systemkonform über einen Tastendruck zu
 informieren, sollte man diese Funktion benutzen.
 Ist ein Stringgadget oder ein Integergadget für diese Tasten-
 kombination zuständig, dann wird dieses hier automatisch aktiviert.
 Sollte ein anderes Gadget für die Tastenkombination vorbelegt sein,
 dann sendet diese eine neue IDCMP_IDCMPUPDATE - Message an das
 Fenster, dadurch werden Notifys automatisch berücksichtigt !

WARNING

Wenn mehrere Gadgets für diese Tastenkombination verantwortlich sind und gleichzeitig sichtbar sind, dann ist nicht vorhersagbar welches Gadget die Message auslöst.

INPUTS

winhandle - WizardWindowHandle, in dessen Fenster diese Taste gedrückt wurde
 key - ASCII-Tastencode der gedrückten Taste
 qualifier - Qualifercode der bei der Taste gegolten hat

RESULT

success - war ein Gadget für diese Kombination zuständig dann TRUE, ansonsten FALSE

1.13 wizard.library/WZ_GetNode

NAME

WZ_GetNode -- Node besorgen

SYNOPSIS

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

```
struct WizardNode *WZ_GetNode(struct MinList *, UWORD);
```

FUNCTION

Diese Funktion ist für die Hierarchy-Klasse geschrieben worden. Mit ihr kann die sichtbare Node des Baumes geholt werden, die sich unter dem Cursor befindet. Rufen Sie diese Funktion also mit dem Wert aus dem Tag WHIERARCHYA_Selected auf.

INPUTS

list - Adresse der MinList-Struktur
 number - Nummer der sichtbaren Node, deren Adresse geholt werden soll

RESULT

node - Adresse der WizardNode oder NULL im Fehlerfall

SEE ALSO

WZ_ListCount ()

1.14 wizard.library/WZ_InitEasyStruct

NAME

WZ_InitEasyStruct -- Easystruct initialisieren

SYNOPSIS

```
easy = WZ_InitEasyStruct(surface, easystruct, id, size)
D0          A0      A1          D0 D1
```

```
struct EasyStruct *WZ_InitEasyStruct(
    APTR, struct EasyStruct *, ULONG *, ULONG, ULONG);
```

FUNCTION

Initialisiert die angegebene EasyStruct-Struktur.

INPUTS

```
surface    - der Returnwert von WZ_OpenSurface()
easystruct - Zeiger auf eine EasyStruct der intuition.library
id         - Nummer des Requesters aus dem StormWizard
size      - Größe dieser EasyStruct-Struktur
```

RESULT

```
easy      - Adresse der übergebenen EasyStruct-Struktur
           oder im Fehlerfall NULL
```

SEE ALSO

```
WZ_EasyRequestArgs()
```

1.15 wizard.library/WZ_InitNodeA

NAME

```
WZ_InitNodeA -- WizardNode initialisieren
```

SYNOPSIS

```
WZ_InitNodeA(wizardnode, entrys, tags)
             A0          D0          A1
```

```
void WZ_InitNodeA(struct WizardNode *, ULONG, struct TagItem *);
```

```
WZ_InitNode(winhandle, entrys, firstTag, ... )
```

```
void WZ_InitNode(struct WizardNode *, ULONG, Tag, ...);
```

FUNCTION

Initialisiert der angegebenen WizardNode.

INPUTS

```
wizardnode - Node die initialisiert werden soll
entrys     - Anzahl der Entrys, die diese Node besitzt
tags       - folgendende sind definiert:
```

```
WNDOEAE_Flags -- (V37), UBYTE
               Flags, die für diese Node gelten
```

SEE ALSO

```
WZ_InitNodeEntry()
```

1.16 wizard.library/WZ_InitNodeEntryA

NAME

```
WZ_InitNodeEntryA -- WizardNode initialisieren
```

SYNOPSIS

```
WZ_InitNodeEntryA(wizardnode, entry, tags)
                  A0          D0      A1
```

```
void WZ_InitNodeEntryA( struct WizardNode *, ULONG, struct TagItem *);
```

```
WZ_InitNodeEntry( winhandle, entry, firstTag, ... )
```

```
void WZ_InitNodeEntry( struct WizardNode *, ULONG, Tag, ...);
```

FUNCTION

Initialisiert den angegebenen Eintrag der WizardNode.

INPUTS

wizardnode - Node deren Eintrag initialisiert werden soll
 entry - Nummer des Eintrages, der initialisiert werden soll
 tags - folgendende sind definiert:

```
WENTRYA_Type -- (V37), UBYTE
               Type, der dieser Eintrag angehören soll
WENTRYA_Pen -- (V37), UWORD
               Farbe dieses Eintrages
WENTRYA_SPen -- (V37), UWORD
               Farbe dieses Eintrages im sel.Zustand
WENTRYA_Style -- (V37), UBYTE
               Zeichensatzstyle dieses Eintrages
WENTRYA_SStyle -- (V37), UBYTE
               Zeichensatzstyle dieses Eintrages im sel.Zustand
WENTRYA_ParentNode -- (V37), struct WizardNode *
               übergeordnete WizardNode
WENTRYA_Childs -- (V37), struct MinList *
               falls von dieser Node weitere ausgehen, ein Zeiger
               auf die MinListStruktur
```

SEE ALSO

WZ_InitNode()

1.17 wizard.library/WZ_ListCount

NAME

WZ_ListCount -- Anzahl der sichtbaren Nodes holen

SYNOPSIS

```
count = WZ_ListCount(list)
D0          A0
```

```
ULONG WZ_ListCount(struct MinList *);
```

FUNCTION

Diese Funktion ist für die Hierarchy-Klasse geschrieben worden.
 Mit ihr kann die Anzahl der sichtbaren Nodes dieses Baumes
 geholt werden.

WARNING

Es muß sich um eine Liste, in der WizardTreeNodees verkettet sind.

INPUTS

list - Adresse der List-Struktur, mit WizardNodes

RESULT

count - Anzahl der sichtbaren Nodes

SEE ALSO

WZ_GetNode()

1.18 wizard.library/WZ_LockWindow

NAME

WZ_LockWindow -- Fenster sperren

SYNOPSIS

```
WZ_LockWindow(winhandle)
                A0
```

```
VOID WZ_LockWindow(struct WizardWindowHandle *);
```

FUNCTION

Diese Funktion sperrt das Fenster für alle möglichen Formen der Benutzereingabe. Gleichzeitig wird der Mauszeiger umgestellt.

INPUTS

winhandle - WizardWindowHandle-Struktur, dessen Fenster gesperrt werden soll

SEE ALSO

WZ_UnlockWindow()

1.19 wizard.library/WZ_LockWindows

NAME

WZ_LockWindows -- alle Fenster sperren

SYNOPSIS

```
WZ_LockWindows(surface)
                A0
```

```
VOID WZ_LockWindows(APTR);
```

FUNCTION

Diese Funktion führt für alle Fenster ein WZ_LockWindow() durch, die zu einer Oberflächenbeschreibung gehören.

INPUTS

surface - Returnwert von WZ_OpenSurface()

SEE ALSO
WZ_UnlockWindows()

1.20 wizard.library/WZ_MenuConfig

NAME
WZ_MenuConfig -- Menüconfigstring holen

SYNOPSIS
string = WZ_MenuConfig(winhandle, code)
D0 A0 D0

STRPTR WZ_MenuConfig(struct WizardWindowHandle *, ULONG);

FUNCTION
Der Configurationsstring zu einem Menü kann mittels dieser Funktion geholt werden. Dieser wurde im StormWiard festgelegt.

INPUTS
winhandle - Zeiger auf ein WizardWindowHandle
code - MenuCode des angewählten MenuTitles,-item
 oder -subitem

RESULT
string - Stringadresse oder NULL in Fehlerfall

SEE ALSO
WZ_GadgetConfig(), WZ_MenuHelp()

1.21 wizard.library/WZ_MenuHelp

NAME
WZ_MenuHelp -- Menühilfestring holen

SYNOPSIS
string = WZ_MenuHelp(winhandle, code)
D0 A0 D0

STRPTR WZ_MenuHelp(struct WizardWindowHandle *, ULONG);

FUNCTION
Der Hilfestring zu einem Menüpunkt kann mittels dieser Funktion geholt werden. Dieser wurde im StormWiard festgelegt. Wurde die Oberflächenbeschreibung mit dem Tag SFH_Catalog bereitgestellt, dann wird der übersetzte String geliefert.

INPUTS
winhandle - Zeiger auf ein WizardWindowHandle
code - MenuCode des angewählten MenuTitles,-item
 oder -subitem

RESULT
 string - Stringadresse oder NULL in Fehlerfall

SEE ALSO
 WZ_GadgetHelp(), WZ_MenuConfig()

1.22 wizard.library/WZ_NewObjectA

NAME

WZ_NewObjectA -- Object anlegen
 WZ_NewObject -- variable Parameterübergabe für Hochsprachen-
 programmierer

SYNOPSIS

```
obj = WZ_NewObjectA(class, tags)
D0          D0          A0
```

```
APTR WZ_NewObjectA(ULONG, struct TagItem *);
```

```
obj = WZ_NewObject(class, firstTag, ... )
```

```
APTR WZ_NewObject(ULONG, Tag, ... );
```

FUNCTION

Legt ein Object einer angegebenen Klasse an.

INPUTS

class - Klasse von der ein Object angelegt werden soll
 tags - folgende Tags sind definiert:

```
GA_DrawInfo, struct DrawInfo *
  Geben Sie hier IMMER einen Zeiger auf einen DrawInfo an.
WGA_Label -- (V37) [C...], STRPTR
  Übergibt einen Zeiger auf einen String, der für das
  Anlegen des Objects wichtig ist. Es handelt sich hierbei
  um ein Universal-Tag.
WGA_Label2 -- (V37) [C...], STRPTR
  Dient dazu einen zweiten Zeiger für das Anlegen eines
  Objektes übergeben zu können. Universal-Tag !
WGA_TextFont -- (V37) [C...], struct TextFont *
  Mit diesem Tag kann der von DrawInfo vorgegebene
  Zeichensatz überschrieben werden. Damit ist es möglich
  jedem Object einen eigenen Zeichensatz zu geben.
  Universal-Tag ! (Vorgabe: DrawInfo->Font)
WGA_Flags -- (V37) [C...], UWORD
  Hier können Sie Flags für das zu erzeugende Object angeben.
  folgenden allgemeine Flags sind definiert :
  WGF_GadgetHelp - falls dieses Gadget eine GadgetHelp
  Message senden soll, wenn es von der Maus
  überfahren wird.
  WGF_Disabled - falls dieses Gadget nicht anwählbar
  sein soll
WGA_Priority -- (V37) [C.G.], UBYTE
  Da es sinnvoll ist, jedem Object einen eigene Priorität
  zu geben, wurde dieses Tag ins Leben gerufen. Geben Sie
```

hier einen Wert im Bereich von 0 bis 255 an. (Vorgabe 0).

WGA_RelHeight -- (V37) [C...], UBYTE
Geben Sie hier den oberen und unteren Rand eines Objectes von seiner TextDimension aus an. (Vorgabe 2).

WGA_MinWidth -- (V37) [C.G.], UWORD
Dieses Attribut kann von jedem Object abgefragt werden. Hiermit kann für anzulegende Objecte die minimale Breite angegeben werden. Dieses Tag ist Klassenabhängig.

WGA_MinHeight -- (V37) [C.G.], UWORD
Dieses Attribut kann von jedem Object abgefragt werden. Hiermit kann für anzulegende Objecte die minimale Höhe angegeben werden. Dieses Tag ist stark Klassenabhängig.

WGA_Link -- (V37) [..S.], struct Gadget *
Um Kommunikationen zwischen Gadgets zu erlauben, deren Inhalt völlig unwichtig für den Programmierer ist, sollte dieses Tag verwendet werden.

WGA_LinkData -- (V37) [C...], UBYTE
Wenn Links zwischen Gadgets gelegt sind, dann ist es klassenabhängig, ob das Object (welches den Link bekommt) auch eine Zeilenangabe erhalten muß. Geben Sie ganz einfach die Zeilennummer an, wenn in Richtung Labelgadget gelinkt wird. Für die Tastatursteuerung ist dieses Tag unerlässlich.

WGA_HelpText -- (V37) [C...], STRPTR
Hilfetexte, die sich mittels WZ_GadgetHelp() von einem Wizardgadget abfragen lassen sollen.

WGA_Config -- (V37) [C...], STRPTR
Die Function WZ_GadgetConfig() gibt diesen String zurück. Geben Sie hier also diesen String mit einem NullByte abgeschlossen an.

WGA_NewImage -- (V37) [C...], struct WizardNewImage *
Damit dieses Object sein Image bekommt, das es für die Darstellung und seine Dimensionsberechnung benötigt, ist dieses Tag definiert worden.

WGA_SelNewImage -- (V37) [C...], struct WizardNewImage *
Geben Sie hier eine WizardNewImage-Beschreibung an, damit sich das Object im selektierten Zustand richtig zeichnen kann.

WGA_Group -- (V37) [C...], struct Gadget *
Damit sich das anzulegende Object in die richtige Gruppe einhängen kann, ist dieses Tag notwendig.
Übergeben Sie die Gadget-Struktur des Gruppengadgets.

WGA_GroupPage -- (V37) [C...], UWORD
Falls das Gruppengadget mehrere Seiten besitzt, dann müssen Sie hier die Seite angeben, in die unser anzulegendes Objekt eingehangen werden soll. (Vorgabe 0)

WGA_Locale -- (V37) [C...], struct Locale *
Bis jetzt ist dieses Tag nur bei Erzeugen eines Datumsgadgets sinnvoll, um die Tageskürzel in der richtigen Sprache zu erhalten. Dieses Tag muß angegeben werden.

WCLASS_LAYOUT:

Sie starten und regeln den Layoutvorgang

WLAYOUTA_RootGadget - (V37) [C...], struct Gadget *

Rootgruppe, deren Layoutvorgang gestartet werden soll

WLAYOUTA_Type - (V37) [C...], UWORD
 Art des Rootgadgets, noch nicht unterstützt

WLAYOUTA_BorderLeft - (V37) [C...], WORD
 WLAYOUTA_BorderRight - (V37) [C...], WORD
 WLAYOUTA_BorderTop - (V37) [C...], WORD
 WLAYOUTA_BorderBottom - (V37) [C...], WORD
 Borderwerte, die beim Layoutvorgang freigehalten werden sollen. (Vorgabe 0)

WLAYOUTA_StackSwap -- (V37) [C...], struct StackSwapStruct *
 Gleichnamige Struktur, in der der Stack für den Layoutvorgang festgehalten ist. Dieses Tag muß angegeben werden.

WCLASS_HGROUP und WCLASS_VGROUP:

- machen Paging (max. 32)
- berechnen Dimension und Position ihrer Mitglieder
- Flags: WGRPF_EqualSize, um allen Mitgliedern die selbe Mindestdimension zu verordnen und WGRPF_DockMode, um es in den Dockmodus umzuschalten.

WGROUPE_ActivePage -- (V37) [CSGN], UWORD
 Dieses Tag beschreibt, welche Seite dieser Gruppe sichtbar ist.

WGROUPE_MaxPage -- (V37) [C...], UWORD
 Gibt die maximale Seitennummer an, welche von diesem Gadget verwaltet werden soll. Maximalwert ist 31. Dieses Tag wird überschrieben, falls der Gruppentitel mehrzeilig ist.

WGROUPE_HBorder -- (V37) [C...], WORD
 Dieser Wert bestimmt den unsichtbaren linken und rechten Rand, welcher freigelassen werden soll.

WGROUPE_VBorder -- (V37) [C...], WORD
 Ähnlich dem HBorder wird hiermit der obere und untere Rand angegeben. Falls ein einzeiliger Title übergeben wurde, dann wird zum oberen Rand noch die Zeichensatzhöhe hinzuaddiert.

WGROUPE_BHOffset -- (V37) [C...], WORD
 Bestimmt den Offset vom inneren Bereich nach links und rechts weg, mit dem ein angegebenes Frame, gezeichnet wird.

WGROUPE_BVOffset -- (V37) [C...], WORD
 Hiermit legt man den Offset vom inneren Bereich nach oben und unten fest, mit dem ein eventuelles Frame gezeichnet werden soll.

WGROUPE_Space -- (V37) [C...], UWORD
 Legt den festen Mindestabstand fest, der zwischen den Mitgliedern dieser Gruppe vorhanden sein soll. Besitzt diese Gruppe kein oder nur ein Mitglied, dann ist dieses Tag wirkungslos.

WGROUPE_VarSpace -- (V37) [C...], UWORD
 Bestimmt das Verhältnis, mit dem der Platz auf die Mitglieder und deren Zwischenräume verteilt werden soll. Maximalwert ist 65535 und bedeutet, daß der Platz voll an die Zwischenräume verteilt werden soll. Alle Objecte dieser Gruppe besitzen dann ihre minimale Dimension.

Ein Wert von 32768 veranlasst das Gruppengadget das Verhältnis zwischen genutzten Platz und Spaceraum etwa gleich zu berechnen. Dies ist natürlich nicht immer möglich.

WGROUPE_FrameType -- (V37) [C...], UWORD
Bestimmt das zu zeichnende Frame im Bereich von 0 bis 8.
(Vorgabe 0).

WCLASS_BUTTON

- einfache Knöpfe

WBUTTONA_Label -- (V37) [CS..], STRPTR
Dient der Angabe eines Labels für ein Buttongadget, es überschreibt das Universaltag WGA_Label !

WCLASS_STRING

- wenn es mit Taste aktivierbar sein soll, muß es mit einem Label gelinkt werden
- Flag: WGF_KeyControl für TABCYCLE. Muß gesetzt sein !

WSTRINGA_String -- (V37) [CSGN], STRPTR
Übergibt die Adresse eines mit NullByte terminierten Strings. Dieses Tag überschreibt beim Anlegen eines Objektes das Universaltag WGA_Label.

WSTRINGA_MaxChars -- (V37) [C...], UWORD
Bestimmt die maximale Länge eines bearbeitbaren Strings ohne NullByte. Maximalwert ist 255.

WSTRINGA_Justification -- (V37) [C...], ULONG
Justierung (GACT_...). (Vorgabe Links).

WCLASS_CHECKBOX

- muß mit Labelgadget gelinkt werden.

WCHECKBOXA_Checked -- (V37) [CSGN], BOOL
Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_MX

- muß mit Labelgadget gelinkt werden.

WMXA_Active -- (V37) [CSGN], UWORD
Beschreibt den ausgewählten Knopf innerhalb dieses Objekts.

WCLASS_LABEL

- häufig als Zielobjekt für Links
- Texte können mehrzeilig sein
- sendet keine Notifys

WLABELA_FrameType -- (V37) [C...], UWORD
Kennzeichnet das Frame, das um dieses Objekt herum gezeichnet werden soll.

WLABELA_Space -- (V37) [C...], UWORD

Abstand in Bildpunkten zwischen den Textzeilen.

WLABELA_BGPen -- (V37) [C...], UWORD
Dieses beschreibt die Hintergrundfarbe. (Vorgabe 0)

WLABELA_TextPlace -- (V37) [C...], UWORD
Ausrichtung der Textzeilen. Geben Sie hier folgendes an:
WZRDPLACE_LEFT, WZRDPLACE_CENTER oder WZRDPLACE_RIGHT.

WLABELA_HighLights -- (V37) [C...], ULONG
Beschreibt, welche Zeilen bei der Darstellung in heller
Textfarbe dargestellt werden sollen. um Zeile 0 und
Zeile 2 hell darzustellen, muß eine $2^0 + 2^2 = 5$
angegeben werden.

WLABELA_Styles -- (V37) [C...], ULONG
Beschreibt, welche Zeilen bei der Darstellung in mit
dem TextStyle Fett dargestellt werden soll.
Um Zeile 0 und Zeile 2 fett darzustellen, muß eine 5
angegeben werden.

WLABELA_Label -- (V37) [C...], STRPTR
Adresse des darzustellenden Textes, dieses Tag über-
schreibt WGA_Label.

WLABELA_Lines -- (V37) [..G.], UWORD
Anzahl der Textzeilen dieses Objects.

WCLASS_INTEGER

- ermöglicht Eingabe von Zahlen in 3 Zahlensystemen
- Flag: WGF_KeyControl für TABCYCLE. Muß gesetzt sein !

WINTERGERA_Long -- (V37) [CSGN], LONG
vorzeichenbehafteter Zahlenwert der vom Benutzer verändert
werden kann.

WINTERGERA_MinLong -- (V37) [C...], LONG
WINTERGERA_MinLong -- (V37) [C...], LONG
Begrenzung des Gadgetwertes.

WINTERGERA_Justification -- (V37) [C...], ULONG
Justierung (GACT...). (Vorgabe Links).

WCLASS_HSCROLLER und WCLASS_VSCROLLER

- Schieberegler für das Scrollen von Listen.
- Links möglich in Richtung:
ListView, MultiListView oder Hierarchy
- Flag WSCF_NewLook, um den Regler im neuen Look zu sehen.

WSCROLLERA_Top -- (V37) [CSGN], LONG
Zahlenwert, der sich im Normalfall im Bereich von
0 bis Total-1 bewegt.
(Vorgabe 0)

WSCROLLERA_Visible (V37) [CSG.], ULONG
Bestimmt die Anzahl der sichtbaren Einträge innerhalb
einer möglichen Liste.

WSCROLLERA_Total -- (V37) [CSG.], ULONG
Anzahl der Einträge einer möglichen Liste.
(Vorgabe 10).

WCLASS_ARROW

- Pfeilgadgets

- Links in Richtung Proportionalgadgets möglich, dann senden die Proportionalgadgets eine Message

WARROWA_Type -- (V37) [C...], UWORD
Pfeilart im Bereich von 0 bis 3. (Links, Rechts, Hoch und Runter). (Vorgabe 0 - Links)

WARROWA_Step -- (V37) [CS..], UWORD
Impulsgröße, bei Wiederholung eines Steps infolge eines Ticks

WCLASS_LISTVIEW und WCLASS_MULTILISTVIEW

- stellt eine Liste mit WizardNodes dar
- Flag WLVF_ReadOnly, um den Rahmen eingedrückt erscheinen zu lassen und Flag WLVF_DoubleClicks, um deren Nachrichten zu erhalten. s.u.

WLISTVIEWEA_Top -- (V37) [CSG.], ULONG
Nummer der ersten sichtbaren Node.

WLISTVIEWEA_Selected -- (V37) [CSGN], LONG
Nummer des selektierten Eintrages in der Liste.
Bei MultiListView handelt es sich um den Eintrag, bei dem sich die Selektierung ändert.

WLISTVIEWEA_List -- (V37) [CS..], struct MinList *
Adresse der darzustellenden Liste, in der Wizard-ListNodes verkettet sind. Dabei können Sie den Strings eine Vektorgrafik vorangestellt. Geben Sie in einem solchen Falle die Adresse in der Node an. Ist eine Node nicht selektiert, dann erfolgt der WZ_DrawVImage() - Aufruf dem type = 0, ansonsten mit eins. Bei einem MultiListView müssen Sie dem Flag WLNF_Selected entnehmen, ob diese Node selektiert wurde bzw. ist. Das Flag LNF_Selected darf bei einem ListView nicht verwendet werden !

WLISTVIEWEA_DoubleClick -- (V37) [...N], BOOL
Wurde eine Message von einem ListView abgeschickt und dieses Tag enthält TRUE, dann handelte es sich um einen Doppelklick als Auslöser.

WLISTVIEWEA_Visible -- (V37) [.S..], UWORD
Geben Sie hier die Nummer einer Node an, die in jedem Fall sichtbar sein soll. Diese Tag kommt in Konflikt mit dem Tag WLISTVIEWEA_Top, da es, wenn die Node nicht sichtbar ist, diesen Wert modifiziert.

WLISTVIEWEA_HeaderSpace

WLISTVIEWEA_HeaderString

WLISTVIEWEA_HeaderStyle

Diese Tags bekommen erst bei späteren Versionen der Library eine Bedeutung.

WCLASS_TOGGLE

- ähnlich den CheckBoxgadgets
- Flag WTGF_SimpleMode, wenn sich das Object nur selektieren lassen soll, das Deselektieren ist dann nicht mehr möglich

WTOGGLEA_Checked -- (V37) [CSGN], BOOL
Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_LINE

- stellen Linien dar
- sendet keine Notifys

WLINEA_Type -- (V37) [C...], UWORD
Beschreibt, wie das Linegadget eine Linie zu zeichnen hat.
Diese sind horizontal(0), vertikal(1), selektiert horizontal(2) und selektiert vertikal(2). (Vorgabe 0)

WLINEA_Label -- (V37) [C...], STRPTR
Übergibt die Adresse eines mit NullByte terminierten Strings, der bei horizontalem Type mit in die Linie hineingeschrieben wird. Dieses Tag überschreibt das UniversalTag WGA_Label.

WCLASS_COLORFIELD

- kann Farben für eine Legende darstellen
- muß mit einem Labelgadget gelinkt werden
- sendet keine Notifys

WCOLORFIELDA_Pen -- (V37) [CS..], UWORD
Nummer des Farbregisters, deren Farbe in dem umrahmten Kästchen erscheinen soll. Es sind die WZRD_... - Farben zugelassen ! (Vorgabe 0)

WCLASS_ARGS

- kann bis zu 10 verschiedene Argumente darstellen
- sendet keine Notifys

WARGSA_Format -- (V37) [.S..], STRPTR
Formatstring für RawDoFmt.

WARGSA_TextPlace -- (V37) [C...], UWORD
Konstante die beschreibt, wie der formatierte Text innerhalb der grafischen Abmessungen justiert werden soll.

WARGSA_FrameType -- (V37) [C...], UWORD
Eine Zahl, die den zu zeichnenden FrameType beschreibt.

WARGSA_TextPen -- (V37) [CS..], UWORD
Farbe, in der die Textausgabe erfolgt.

WARGSA_Arg0-9 -- (V37) [CS..], ULONG
Geben Sie hier ihr Argument an, welches dann nach dem übergebenen Formatierungsstring formatiert wird.
ACHTUNG: Wird die Adresse eines Strings übergeben, kann dieser nicht in einen eigenen Puffer kopiert werden !

WCLASS_GAUGE

- zum Darstellen von zeitlichen Abläufen.
- sendet keine Notifys

WGAUGE_Total -- (V37) [CS..], UWORD
obere Grenze des Füllstandes

WGAUGEA_Current -- (V37) [CS..], UWORD
momentane Position des Füllstandes, dieser Wert darf
sich im Bereich von 0 bis Total bewegen.

WGAUGEA_Format -- (V37) [CS..], STRPTR
Dieses Tag überschreibt WGA_Label und dient als
Formatierungsstring für die Funktion der zusammen mit
den Argumenten an die exec.library/RawDoFmt() übergeben
wird. Dieses Tag überschreibt das UniversalTag WGA_Label.

WCLASS_CYCLE

- können für die Tastaturkontrolle mit einem Label gelinkt werden

WCYCLEA_Active -- (V37) [CSGN], UWORD
beschreibt den aktiven Eintrag.

WCYCLEA_Labels -- (V37) [CS..], STRPTR
Adresse mehrerer Strings, die mit einem Return ("\n")
getrennt sind. Dieses Tag überschreibt das Tag WGA_Label.

WCLASS_VECTORBUTTON

- Verhalten sich wie Objecte der Buttonklasse

WVECTORBUTTONA_Type -- (V37) [C...], UWORD
Nummer des Vektorimages, das für die Darstellung genommen
werden soll: File(0), Drawer(1) oder Popup(2).

WCLASS_DATE

- Verhalten sich wie Kalenderblätter

WDATEA_Day -- (V37) [CSGN], UWORD
der selektierte Tag, der mindestens vom Wert Eins sein
muß. (Vorgabe 1)

WDATEA_Month -- (V37) [CS..], UWORD
der Monat der für die Darstellung des Kalenderblattes
wichtig ist. Mindestens eine Eins für den Januar !
(Vorgabe 1)

WDATEA_Year -- (V37) [CS..], UWORD
das Jahr, das mindestens 1978 betragen muß.
(Vorgabe 1978)

WCLASS_SPACE

- dienen als Platzhalter und Flächenfüller

WSPACEA_Pen -- (V37) [C...], UWORD
Farbe, die mit einem Raster gezeichnet werden soll.
Als zweite Farbe dient die Hintergrundfarbe.

WCLASS_IMAGE

- stellen Bitmaps im Fenster dar
- senden keine Notifys

WIMAGEA_BGPen -- (V37) [C...], UWORD
Farbe, mit der der eventuell vorhandene Rand dargestellt

werden soll.

WIMAGEA_FrameType -- (V37) [C...], UWORD
 Beschreibt, mit welchem Frame die Dimension des Objektes
 gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,
 WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
 WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGEA_HBorder -- (V37) [C...], UWORD
 Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGEA_VBorder -- (V37) [C...], UWORD
 Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGEA_NewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das zu zeichnende
 Image beschreibt. Dieses Tag überschreibt WGA_NewImage.

WCLASS_IMAGEBUTTON

- verhalten sich wie Buttons, sehen aber aus wie Images

WIMAGEBUTTONA_BGPen -- (V37) [C...], UWORD
 Farbe, mit der der vorhandene Rand gezeichnet werden soll,
 wenn das Image im Normalzustand ist.

WIMAGEBUTTONA_SelBGPen -- (V37) [C...], UWORD
 Farbe, mit der der vorhandene Rand gezeichnet werden soll,
 wenn das Image im selektierten Zustand ist.

WIMAGEBUTTONA_FrameType -- (V37) [C...], UWORD
 Beschreibt, mit welchem Frame die Dimension des Objektes
 gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,
 WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
 WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGEBUTTONA_HBorder -- (V37) [C...], UWORD
 Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGEBUTTONA_VBorder -- (V37) [C...], UWORD
 Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGEBUTTONA_NewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das zu zeichnende
 Image beschreibt. Dieses Tag muß angegeben werden.

WIMAGEBUTTONA_SelNewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das selektierte
 Image beschreibt.

WCLASS_IMAGETOGGLE

- verhalten sich wie Toggles, sehen aber aus wie Images
 - Flag WITF_SimpleMode, wenn sich das Object nur selektieren lassen
 soll, das Deselektieren ist dann nicht mehr möglich

WIMAGETOGGLEA_BGPen -- (V37) [C...], UWORD
 Farbe, mit der der vorhandene Rand gezeichnet werden soll,
 wenn das Image im Normalzustand ist.

WIMAGETOGGLEA_SelBGPen -- (V37) [C...], UWORD
 Farbe, mit der der vorhandene Rand gezeichnet werden soll,
 wenn das Image im selektierten Zustand ist.

WIMAGETOGGLEA_FrameType -- (V37) [C...], UWORD
 Beschreibt, mit welchem Frame die Dimension des Objektes
 gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,
 WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
 WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGETOGGLEA_HBorder -- (V37) [C...], UWORD
 Mindestabstand zum linken und rechten Rand. (Vorgabe 0)
 WIMAGETOGGLEA_VBorder -- (V37) [C...], UWORD
 Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)
 WIMAGETOGGLEA_NewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das zu zeichnende
 Image beschreibt. Dieses Tag muß angegeben werden.
 WIMAGETOGGLEA_SelNewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das selektierte
 Image beschreibt.
 WIMAGETOGGLEA_Checked -- (V37) [CSGN], BOOL
 Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_IMAGEPOPUP

 - Images mit Textpopup-Effekt
 - Flag WIPF_NewLook, um den Popup in den 3D-Look zu schalten

 WIMAGEPOPUPA_BGPen -- (V37) [C...], UWORD
 Farbe, mit der der vorhandene Rand gezeichnet werden soll,
 wenn das Image im Normalzustand ist.
 WIMAGEPOPUPA_FrameType -- (V37) [C...], UWORD
 Beschreibt, mit welchem Frame die Dimension des Objektes
 gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,
 WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
 WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)
 WIMAGEPOPUPA_HBorder -- (V37) [C...], UWORD
 Mindestabstand zum linken und rechten Rand. (Vorgabe 0)
 WIMAGEPOPUPA_VBorder -- (V37) [C...], UWORD
 Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)
 WIMAGEPOPUPA_NewImage -- (V37) [C...], struct WizardNewImage *
 Adresse der WizardNewImage-Struktur, die das zu zeichnende
 Image beschreibt. Dieses Tag muß angegeben werden.
 WIMAGEPOPUPA_TextPlace -- (V37) [C...], UWORD
 Justierung für die Textausgabe während eines Popups.
 WZRDPLACE_LEFT, ...
 WIMAGEPOPUPA_Labels -- (V37) [C...], STRPTR
 Adresse der durch Return ("\n") getrennten Text für den
 Popup-Dialog. Dieses Tag überschreibt das Tag WGA_Label.
 WIMAGEPOPUPA_Selected -- (V37) [...N], UWORD
 Nummer des Selektierten Texteintrages

WCLASS_TEXTPOPUP

 - sehen aus wie Buttons, aber mit Popup-Effekt
 - Flag WTPF_NewLook, um den Popup in den 3D-Look zu schalten

 WTEXTPOPUPA_TextPlace -- (V37) [C...], UWORD
 Justierung der Textausgabe bei einem Popup-Effekt
 WTEXTPOPUPA_Labels -- (V37) [C...], STRPTR
 Adresse der durch Return ("\n") getrennten Texte,
 für den Popup-Dialog. Dieses Tag überschreibt das Tag
 WGA_Label2 und muß angegeben werden.
 WTEXTPOPUPA_Name -- (V37) [C...], STRPTR
 Adresse des Textes, der in dem Button-Frame erscheinen
 tut. Dieses Tag überschreibt WGA_Label !
 WTEXTPOPUPA_Selectd -- (V37) [...N], UWORD

Nummer des selektierten Textes, bei einem Popup-Effekt

WCLASS_PALETTE

- ermöglichen Auswahl einer Farbe aus einer Farbpalette

WPALETTEA_Colors -- (V37) [C...], UWORD
 Anzahl der Farben, die zur Auswahl stehen sollen oder
 eine -1, um die maximale Farbanzahl zu nutzen.
 Maximal darf dieser Wert 255 betragen.
 WPALETTEA_Selected -- (V37) [CSGN], WORD
 Nummer der selektierten Farbe. (Vorgabe -1)
 WPALETTEA_Offset -- (V37) [C...], UWORD
 Nummer des ersten zu Wahl stehenden Farbregisters.

WCLASS_VECTORPOPUP

- sehen aus wie Vectorbuttons, aber mit Popup-Effekt
 - Flag WVPF_NewLook, um den Popup in den 3D-Look zu schalten

WVECTORPOPUPA_Type -- (V37) [C...], UWORD
 Nummer des Vektorimages, das für die Darstellung genommen
 werden soll: File(0), Drawer(1) oder Popup(2).
 WVECTORPOPUPA_Labels -- (V37) [C...], STRPTR
 Adresse der durch Return ("\n") getrennten Texte,
 für den Popup-Dialog. Dieses Tag überschreibt das Tag
 WGA_Label und muß angegeben werden.
 WVECTORPOPUPA_TextPlace -- (V37) [C...], UWORD
 Justierung der Textausgabe bei einem Popup-Effekt
 WVECTORPOPUPA_Selected -- (V37) [...N]
 Nummer des selektierten Textes, bei einem Popup-Effekt

WCLASS_HIERARCHY

- ähnlich den ListViews
 - können hierarchische Listen darstellen
 - Flag WHRF_DoubleClicks erlaubt das Abschicken einer Nachricht
 bei einem Doppelklick !

WHIERARCHYA_ImageType -- (V37) [C...], UWORD
 wählt ein bestehendes Image vor. Es kann sein:
 Keins(0), Baum (1), Dreieck (2).
 WHIERARCHYA_ImageWidth -- (V37) [CS..], UWORD
 Breite der zu zeichnenden Images, die auch als
 Schalter fungieren.
 WHIERARCHYA_Top -- (V37) [CSG.], ULONG
 Nummer der ersten sichtbaren Node.
 WHIERARCHYA_List -- (V37) [CS..], struct MinList *
 Adresse der Liste, deren Inhalt dargestellt werden soll.
 Dabei muß es sich um eine Liste mit WizardTreeNodees
 handeln und deren Felder ParentNode und Childs müssen
 korrekt initialisiert sein. Die Struktur kann und sollte
 in Form eines Baumes verkettet sein. Alle direkt in der
 MinList-Struktur verketteten Nodes müssen eine Null
 im Feld ParentNode erhalten, da diese nicht existiert.
 WHIERARCHYA_Visible -- (V37) [.S..], UWORD
 Geben Sie hier die Nummer einer sichtbaren Node an,

die in jedem Fall im Fenster dargestellt werden soll.
Dabei zählen nur Nodes, deren Darstellung aufgrund der
offenen ElternNode möglich ist.

WHIERARCHYA_Selected -- (V37) [CSGN], LONG

Nummer des selektierten Eintrages in der Liste.
Bei MultiListView handelt es sich um den Eintrag, bei
dem sich die Selektierung ändert.

WHIERARCHYA_DoubleClick -- (V37) [...N], BOOL

Wurde eine Message von einem Object abgeschickt
und dieses Tag enthält TRUE, dann handelte es sich
um einen Doppelklick als Auslöser.

WCLASS_HSLIDER und WCLASS_VSLIDER

- Schieberegler für alle möglichen Einstellung.
- Flag WSLF_NewLook, um den Regler im neuen Look zu sehen.

WSLIDERA_Level -- (V37) [CSGN], WORD

Zahlenwert, der sich im Bereich der Werte von Min bis
Max bewegen kann.
(Vorgabe 0)

WSLIDERA_Min (V37) [CSG.], WORD

Bestimmt die untere Grenze, die vom Level-Wert nicht unter-
schritten werden darf. (Vorgabe -10)

WSLIDERA_Max -- (V37) [CSG.], WORD

Bestimmt die obere Grenze, die der Level-Wert nicht
überschreiten darf. (Vorgabe 10)

RESULT

obj - Zeiger auf das erzeugte Object.

SEE ALSO

intuition.library/DisposeObject()

1.23 wizard.library/WZ_ObjectID

NAME

WZ_ObjectID -- ObjectID ermitteln

SYNOPSIS

```
success = WZ_ObjectID(surface, id, objectname)
D0                A0                A2 A1
```

FUNCTION

Diese Funktion liefert die ObjectID eines Objektes, dessen Name
übergeben wurde.

INPUTS

surface - Returnwert von WZ_OpenSurface()
id - Zeiger auf einen 4Byte großen Speicherbereich, aus
dem bei Erfolg die ID entnommen werden kann
objectname - Name des Objektes, dessen interne ID ermittelt werden
soll

```

RESULT
    sucess      - wurde die ID gefunden, dann TRUE sonst FALSE

SEE ALSO
    WZ_OpenSurface()

```

1.24 wizard.library/WZ_OpenSurfaceA

```

NAME
    WZ_OpenSurfaceA -- Oberflächenbeschreibung verfügbar machen
    WZ_OpenSurface  -- variable Parameterübergabe für Hochsprachen-
                    programmierer

SYNOPSIS
    Surface = WZ_OpenSurfaceA(name, memaddr, tags)
    D0          A0      A1      A2

    APTR WZ_OpenSurfaceA(STRPTR, APTR, struct TagItem *);

    Surface = WZ_OpenSurface(name, memaddr, firsttag, ... )

    APTR WZ_OpenSurface(STRPTR, APTR, Tag, ... );

FUNCTION
    Diese Funktion wird benutzt um eine Oberflächenbeschreibung
    benutzen zu können. Befindet sich die Datei bereits an einer
    Adresse im Speicher, kann der name mit NULL übergeben werden
    und memaddr enthält die Adresse im Speicher.
    Im Normalfall sollte der name den Zeiger auf einen Namen der
    Datei enthalten und memaddr gleich NULL sein.

WARNING
    Wird eine Adresse mittels memaddr übergeben, muß sichergestellt
    werden, das diese nur einmal bei einem WZ_OpenSurface - Aufruf
    initialisiert wird. D.h. Programme die von dieser Möglichkeit
    Gebrauch machen, dürfen nicht resident gemacht werden !

INPUTS
    name      - ein Zeiger auf den Namen der Datei (NullByte !)
    memaddr   - wenn name nicht angegeben wird, ein Zeiger auf eine
                Adresse, die die Beschreibung der Oberfläche enthält
    tags      - folgende Tags sind definiert:

                SFH_Locale - (V37) , struct Locale *
                    ein Zeiger von locale.library/OpenLocale(),
                    falls die Standardeinstellung des Betriebssystems
                    nicht verwendet werden soll.
                SFH_Catalog - (V37), struct Catalog *
                    ein Zeiger auf den geöffnete Catalog,
                    siehe locale.library/OpenCatalog().
                SFH_AutoInit - (V37), BOOL
                    soll die Oberflächenbeschreibung von einer vorgegebenen
                    Speicheradresse geöffnet werden, dann kann das initiali-
                    sieren hiermit abgeschaltet werden, falls bereits eine
                    Initialisierung erfolgt ist. ( Vorgabe TRUE )

```

RESULT

Surface - undefinierter Zeiger oder Null im Fehlerfall

SEE ALSO

WZ_CloseSurface(), WZ_SnapShotA()

1.25 wizard.library/WZ_OpenWindowA()

NAME

WZ_OpenWindowA -- Fenster öffnen
 WZ_OpenWindow -- variable Parameterübergabe für Hochsprachen-
 programmierer

SYNOPSIS

```
window = WZ_OpenWindowA(winhandle, newwin, tags)
D0          A0          A1          A2
```

```
struct Window *WZ_OpenWindowA(struct WizardWindowHandle *,
    struct NewWindow *, struct TagItem *);
```

```
window = WZ_OpenWindow(winhandle, newwin, firstTag, ... )
```

```
struct Window *WZ_OpenWindow(struct WizardWindowHandle *,
    struct NewWindow *, Tag, ... );
```

FUNCTION

Öffnet ein Fenster ähnlich dem Intuitionsaufruf.

WARNING

Das Benutzen der Intuition-Funktion führt zu Fehlern im Layout-
 vorgang.

INPUTS

winhandle - von WZ_AllocWindowHandle initialisiertes
 WizardWindowHandle
 newwin - NewWindowStruct or NULL
 z.b. von WZ_CreateWindowObj()
 tags - Tags (z.b. WA_AutoAdjust)

RESULT

window - Zeiger auf eine Window-Struktur
 oder NULL im Fehlerfall

SEE ALSO

WZ_CloseWindow()

1.26 wizard.library/WZ_SnapShotA()

NAME

WZ_SnapShotA -- Fensterpositionen und -dimensionen fixieren

SYNOPSIS

```

success = WZ_SnapShotA(surface, tags)
D0                      A0

BOOL WZ_SnapShot(APTR, struct TagItem *);

success = WZ_SnapShot(surface, firstTag, ... )
D0                      A0

BOOL WZ_SnapShot(APTR, Tag, ... );

```

FUNCTION

Eine Funktion zum speichern der Fensterpositionen und -dimensionen in die .wizard - Datei zurück. Noch offene Fenster werden nicht berücksichtigt. Im allgemeinen sollte man diese Funktion am Ende des Program's aufrufen.

WARNING

War die Funktion WZ_OpenSurface() mit dem memadr - Parameter aufgerufen worden, dann ist das fixieren nicht möglich.

INPUTS

```

surface - Returnwert von WZ_OpenSurface()
tags    - zur Zeit keine definiert.

```

RESULT

```

success - TRUE -> wenn alles glatt ging,
          FALSE -> bei einem Fehler

```

SEE ALSO

```

WZ_OpenSurface(), WZ_CloseSurface()

```

1.27 wizard.library/WZ_UnlockWindow

NAME

```

WZ_UnlockWindow -- Fenstereingaben wieder erlauben

```

SYNOPSIS

```

result = WZ_UnlockWindow(winhandle)
D0                      A0

ULONG WZ_UnlockWindow(struct WizardWindowHandle *);

```

FUNCTION

Benutzereingaben werden mit dieser Funktion wieder erlaubt.

WARNING

Ist das Fenster nicht gesperrt, dann kommt es zu schweren Systemfehlern. Wurde das Fenster mittels WZ_LockWindow() mehrmals gesperrt, dann muß es auch genauso oft mittels WZ_UnlockWindow wieder erlaubt werden. D.h. ein Aufruf muß nicht unbedingt tatsächlich Benutzereingaben wieder zulassen.

INPUTS

```

winhandle - WizardWindowHandler, dessen Fenster Eingaben

```

wieder erlauben soll

RESULT

result - enthält, wie oft das Fenster noch mittels WZ_UnlockWindow() für Benutzereingaben zugänglich gemacht werden muß, dieser Wert ist gleich NULL wenn das Fenster wieder Benutzereingaben empfängt bei -1 ist ein schwerer Fehler aufgetreten.

SEE ALSO

WZ_LockWindow()

1.28 wizard.library/WZ_UnlockWindows

NAME

WZ_UnlockWindows -- Fenstereingaben wieder erlauben

SYNOPSIS

```
WZ_UnlockWindows(surface)
                A0
```

```
VOID WZ_UnlockWindows(APTR);
```

FUNCTION

Diese Funktion führt für alle Fenster ein WZ_UnlockWindow() durch, die zu einer Oberflächenbeschreibung gehören.

INPUTS

surface - Returnwert von WZ_OpenSurface()

SEE ALSO

WZ_LockWindows()