pGadgetD

COLLABORATORS			
	TITLE : pGadgetD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 29, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 pGadgetD **1**

1.1 pGadgetD.doc . . . 1
1.2 pGadget.library/pOS_GadgetHandleIEvent() . . . 1
1.3 pGadget.library/pOS_CalcGadgetBox() . . . 2
1.4 pGadget.library/pOS_CalcGadgetInnerBox() . . . 2
1.5 pGadget.library/pOS_DrawGadget() . . . 3
1.6 pGadget.library/pOS_NewGObjectA() . . . 3
1.7 pGadget.library/pOS_DisposeGObject() . . . 4
1.8 pGadget.library/pOS_ObtainGRastPort() . . . 4
1.9 pGadget.library/pOS_ReleaseGRastPort() . . . 5
1.10 pGadget.library/pOS_GadgetHitTest() . . . 5
1.11 pGadget.library/pOS_GadgetHitTestC() . . . 6
1.12 pGadget.library/pOS_LinkGadHelpIDs() . . . 6
1.13 pGadget.library/pOS_AddLinkGadHelpID() . . . 7
1.14 pGadget.library/pOS_AddLinkGadHelpAll() . . . 7
1.15 pGadget.library/pOS_CalcRelMousePoint() . . . 8
1.16 pGadget.library/pOS_GadgetDeleteDragList() . . . 8
1.17 pGadget.library/pOS_GadgetDropTest() . . . 9
1.18 pGadget.library/pOS_GadgetDropIEvent() . . . 9
1.19 pGadget.library/pOS_DrawDisableRect() . . . 10
1.20 pGadget.library/pOS_IsGadgetMember() . . . 10

Chapter 1

pGadgetD

1.1 pGadgetD.doc

```
                                pGadget.library

pOS_AddLinkGadHelpAll()      pOS_AddLinkGadHelpID()
pOS_CalcGadgetBox()          pOS_CalcGadgetInnerBox()
pOS_CalcRelMousePoint()      pOS_DisposeGObject()
pOS_DrawDisableRect()        pOS_DrawGadget()
pOS_GadgetDeleteDragList()   pOS_GadgetDropIEvent()
pOS_GadgetDropTest()         pOS_GadgetHandleIEvent()
pOS_GadgetHitTest()          pOS_GadgetHitTestC()
pOS_IsGadgetMember()         pOS_LinkGadHelpIDs()
pOS_NewGObjectA()            pOS_ObtainGRastPort()
pOS_ReleaseGRastPort()
```

1.2 pGadget.library/pOS_GadgetHandleIEvent()

```
   NAME
       pOS_GadgetHandleIEvent -- InputEvent auswerten

   SYNOPSIS
        BOOL pOS_GadgetHandleIEvent(Unit,InputEvent);
        _R_D0                       _R_A0    _R_A1

        BOOL pOS_GadgetHandleIEvent(
                   struct pOS_IntuiUnit*,struct pOS_InputEvent*);

   FUNCTION

   INPUTS
        Unit       - pIntui-Unit
        InputEvent - Event

   RESULT
        TRUE => Event wurde vollständig bearbeitet
```

```
SEE ALSO
```

1.3 pGadget.library/pOS_CalcGadgetBox()

```
NAME
    pOS_CalcGadgetBox -- Gesamte Gadget-Fläche berechnen

SYNOPSIS
     VOID pOS_CalcGadgetBox(window,gadget,rectangle);
                            _R_A0    _R_A1   _R_A2

     VOID pOS_CalcGadgetBox(
                const struct pOS_Window*,const struct pOS_Gadget*,
                struct pOS_Rectangle*);

FUNCTION
     Berechnung der Gadget-Koordinaten im Window. Wird das Gadget
     relativ zur Window-Größe dargestellt, so berechnet
     pOS_CalcGadgetBox() die jetzige absolute Größe.

INPUTS
     window    - Gadget-Window
     gadget    - zu berechnendes Gadget
     rectangle - Ergebnis

RESULT

SEE ALSO
```

1.4 pGadget.library/pOS_CalcGadgetInnerBox()

```
NAME
    pOS_CalcGadgetInnerBox -- Zeichenfläche für Gadget berechnen

SYNOPSIS
     VOID pOS_CalcGadgetInnerBox(window,gadget,rectangle);
                                 _R_A0    _R_A1   _R_A2

     VOID pOS_CalcGadgetInnerBox(
                const struct pOS_Window*,const struct pOS_Gadget*,
                struct pOS_Rectangle*);

FUNCTION
     Siehe pOS_CalcGadgetBox()
     Das Ergebnis dieser Berechnung ist die Fläche, in der sich das
     Gadget zeichnen muß. Der Gadget-Border wird beachtet und aus der
     Gadget-Fläche herausgerechnet.

INPUTS
     window    - Gadget-Window
```

```
     gadget    - zu berechnendes Gadget
     rectangle - Ergebnis

RESULT

SEE ALSO
```

1.5 pGadget.library/pOS_DrawGadget()

```
NAME
    pOS_DrawGadget -- Gadget zeichnen

SYNOPSIS
     VOID pOS_DrawGadget(gadget,method,mode);
                         _R_A0    _R_A1 _R_D0

     VOID pOS_DrawGadget(
                const struct pOS_Gadget*,struct pOS_GadgetMethod*,ULONG);

FUNCTION
     Bevor die Funktion aufgerufen wird, müssen folgende Daten gesetzt
     werden:
       imth_Render.imre_Info
       imth_Render.imre_NewTick
       imth_Render.imre_OldTick
     Das Gadget wird dann mit dem Zeichenmodus 'mode' gezeichnet.

INPUTS
     gadget  - zu zeichnendes Gadget
     methode - gefullte Daten-Struktur
     mode    - (enum pOS_GadgetClassRender)
RESULT

SEE ALSO
```

1.6 pGadget.library/pOS_NewGObjectA()

```
NAME
    pOS_NewGObjectA -- neues Objekt erzeugen

SYNOPSIS
     obj = pOS_NewGObjectA(NClass,name, ver,  tagList);
     _R_D0                 _R_A0  _R_A1 _R_D0  _R_A2

     __ARID__ APTR pOS_NewGObjectA(
                struct pOS_NClass*,const CHAR*,ULONG,const pOS_TagItem*);

FUNCTION

INPUTS
```

```
     NClass  - Zeiger auf die Klasse
     name    - Name der Klasse
     ver     - Version der Klass
     tagList - Parameter

RESULT
     neues Objekt oder NULL

SEE ALSO
```

1.7 pGadget.library/pOS_DisposeGObject()

```
NAME
    pOS_DisposeGObject -- Objekt löschen

SYNOPSIS
     VOID pOS_DisposeGObject(object);
                             _R_A0

     VOID pOS_DisposeGObject(__ARID__ APTR);

FUNCTION
     Ein mit pOS_NewGObjectA() erzeugtes Objekt muß mit
     pOS_DisposeGObject() wieder gelöscht werden.

INPUTS
     objekt - zu löschendes Objekt
RESULT

SEE ALSO
```

1.8 pGadget.library/pOS_ObtainGRastPort()

```
NAME
    pOS_ObtainGRastPort -- Gadget-RastPort ermitteln

SYNOPSIS
     rp = pOS_ObtainGRastPort(gadget,info);
     _R_D0                    _R_A0   _R_A1

     struct pOS_RastPort *pOS_ObtainGRastPort(
                const struct pOS_Gadget*,const struct pOS_IClassInfo*);

FUNCTION
     Ein Gadget darf sich nur in den RastPort zeichnen, der hiermit
     ermittelt wurde. Der RastPort muß wieder mit pOS_ReleaseGRastPort()
     abgemeldet werden.

INPUTS
     gadget - zu zeichnendes Gadget
```

```
     info   - Bezugsdaten

RESULT
     RastPort zum Zeichnen oder NULL

SEE ALSO
```

1.9 pGadget.library/pOS_ReleaseGRastPort()

```
NAME
     pOS_ReleaseGRastPort -- Gadget-RastPort abgelden

SYNOPSIS
     VOID pOS_ReleaseGRastPort(info,rp);
                              _R_A0   _R_A1

     VOID pOS_ReleaseGRastPort(
                const struct pOS_IClassInfo*,struct pOS_RastPort*);

FUNCTION
     Freigeben eines mit pOS_ObtainGRastPort() ermittelten
     RastPorts.

INPUTS
     info - Bezugsdaten
     rp   - RastPort von pOS_ObtainGRastPort()

RESULT

SEE ALSO
```

1.10 pGadget.library/pOS_GadgetHitTest()

```
NAME
    pOS_GadgetHitTest -- Ermittelt Position innerhalb Gadget

SYNOPSIS
     gad = pOS_GadgetHitTest(method,gadget);
     _R_D0                   _R_A0   _R_A1

    struct pOS_Gadget* pOS_GadgetHitTest(
             struct pOS_GadgetMethod*,const struct pOS_Gadget*);

FUNCTION
     Vor dem Funktionsaufruf muß
        imth_HitTest.imht_AMouse und
        imth_HitTest.imht_RMouse
     gestzt werden. Es wird imth_Method mit GCLMTH_HitTest
     überschrieben. Liegt die Position 'imht_AMouse' im Gadget,
     wird das betreffende Gadget zurückgegeben. Es ist nicht
```

```
     sichergestellt, daß das 'gadget' ebenfalls als Ergebnis dient.

INPUTS
     method - gefüllte Struktur
     gadget - zu prüfendes Gadget

RESULT
     Gadget, das an der Position liegt.

SEE ALSO
```

1.11 pGadget.library/pOS_GadgetHitTestC()

```
NAME
    pOS_GadgetHitTestC --

SYNOPSIS
     gad = pOS_GadgetHitTestC(method,gadget,mode);
     _R_D0                   _R_A0   _R_A1  _R_D0

    struct pOS_Gadget* pOS_GadgetHitTestC(
                const struct pOS_GadgetMethod*,const struct pOS_Gadget*,
                ULONG mode);

FUNCTION
     Im Gegensatz zu pOS_GadgetHitTest() wird 'method' nicht verändert.


INPUTS
     method - gefüllt Struktur
     gadget - zu prüfendes Gadget
     mode   - 0

RESULT
     Gadget, das an der Position liegt.

SEE ALSO
```

1.12 pGadget.library/pOS_LinkGadHelpIDs()

```
NAME
    pOS_LinkGadHelpIDs -- Erstellt einen gesamten Help-Path

SYNOPSIS
     str = pOS_LinkGadHelpIDs(screen, point, buffer, bufSize, level);
     _R_D0                   _R_A0   _R_A1  _R_A2    _R_D0    _R_D1

    const CHAR* pOS_LinkGadHelpIDs(
                struct pOS_Screen*,const struct pOS_Point*,
                CHAR*,size_t,ULONG level);
```

```
FUNCTION
     Vollständiger Help-Path an einer Pixelposition erstellen.


INPUTS
     screen  - Screen auf welchen die HelpID berechnet werden soll.
     point   - Pixelposition relativ zum Screen
     buffer  - Arbeitspuffer für den Help-Path
     bufSize - Byte-Länge von 'buffer'
     level   - Arbeits-Level (0,1,2...)

RESULT
     buffer, NULL => kein Help verfügbar

SEE ALSO
```

1.13 pGadget.library/pOS_AddLinkGadHelpID()

```
NAME
    pOS_AddLinkGadHelpID -- Help-String anhängen

SYNOPSIS
     VOID pOS_AddLinkGadHelpID(method,name);
                               _R_A0   _R_A1

    VOID pOS_AddLinkGadHelpID(struct pOS_GadgetMethod*,const CHAR*);

FUNCTION

INPUTS
     method - initialisierte 'imth_Help' Daten
     name   - zu addierender Help-Part

RESULT

SEE ALSO
```

1.14 pGadget.library/pOS_AddLinkGadHelpAll()

```
NAME
    pOS_AddLinkGadHelpAll -- Gadget-Tree durchlaufen und Help-Path bilden

SYNOPSIS
     ULONG pOS_AddLinkGadHelpAll(gadget,method);
      _R_D0                      _R_A0   _R_A1

    ULONG pOS_AddLinkGadHelpAll(
                const struct pOS_Gadget*,struct pOS_GadgetMethod*);
```

```
FUNCTION

INPUTS
     gadget - Gadget, das ein Help erhalten soll
     method - initialisierte 'imth_Help' Daten

RESULT
     aktueller Level

SEE ALSO
```

1.15 pGadget.library/pOS_CalcRelMousePoint()

```
NAME
    pOS_CalcRelMousePoint -- Relative Gadget-Koors berechnen

SYNOPSIS
     pOS_CalcRelMousePoint(gadget, window,abs,  rel);
                           _R_A0  _R_A1  _R_A2  _R_A3

    VOID pOS_CalcRelMousePoint(
                const struct pOS_Gadget*,const struct pOS_Window*,
                const struct pOS_Point*,struct pOS_Point*);

FUNCTION

INPUTS
     gadget - Gadget
     window - Window, in dem 'gadget' dargestellt wird
     abs    - absolute Koordinaten im Window
     rel    - Ergebnis: relative Koordinaten im Gadget

RESULT

SEE ALSO
```

1.16 pGadget.library/pOS_GadgetDeleteDragList()

```
NAME
    pOS_GadgetDeleteDragList -- Drag-Objekt freigeben

SYNOPSIS
     pOS_GadgetDeleteDragList(DragList);
                              _R_A0

     VOID pOS_GadgetDeleteDragList(struct pOS_DragList*);

FUNCTION
     Alle Drag-Objekte werden zum Sende-Gadget geschickt und
     freigegeben.
```

```
INPUTS
     DragList - Liste der Drag-Objekte

RESULT

SEE ALSO
```

1.17 pGadget.library/pOS_GadgetDropTest()

```
NAME
    pOS_GadgetDropTest -- Ermittelt Drop-Position innerhalb Gadget

SYNOPSIS
     gad = pOS_GadgetDropTest(method,gadget);
     _R_D0                    _R_A0   _R_A1

     struct pOS_Gadget* pOS_GadgetDropTest(
                struct pOS_GadgetMethod*,const struct pOS_Gadget*);

FUNCTION
     Vor dem Funktionsaufruf muß
        imth_DropTest.imdt_AMouse und
        imth_DropTest.imdt_RMouse
     gestzt werden. Es wird imth_Method mit GCLMTH_DropTest
     überschrieben. Liegt die Position 'imdt_AMouse' im Gadget, wird
     das betreffende Gadget zurückgegeben. Es ist nicht sichergestellt,
     daß das 'gadget' ebenfalls als Ergebnis dient.

INPUTS
     method - gefüllte Struktur
     gadget - zu prüfendes Gadget

RESULT
     Gadget, das an der Position liegt.

SEE ALSO
```

1.18 pGadget.library/pOS_GadgetDropIEvent()

```
NAME
    pOS_GadgetDropIEvent -- InputEvent während dem Drag&Drop auswerten

SYNOPSIS
     BOOL pOS_GadgetDropIEvent(Unit,  ie,  DragList);
     _R_D0                    _R_A0   _R_A1  _R_A2

     BOOL pOS_GadgetDropIEvent(
                struct pOS_IntuiUnit*,struct pOS_InputEvent*,
                struct pOS_DragList*);
```

```
FUNCTION

INPUTS
     Unit     - pIntui-Unit
     ie       - Event
     DragList - Liste der Drag-Objekte

RESULT

SEE ALSO
```

1.19 pGadget.library/pOS_DrawDisableRect()

```
NAME
    pOS_DrawDisableRect -- Gitter zeichnen

SYNOPSIS
     pOS_DrawDisableRect(rp,x1,y1,x2,y2);
                    _R_A0   _R_D0 - _R_D3

     VOID pOS_DrawDisableRect(
                struct pOS_RastPort*,SLONG,SLONG,SLONG,SLONG);

FUNCTION

INPUTS
     rp          - RastPort
     x1,y1,x2,y2 - Rectange-Koordinaten

RESULT

SEE ALSO
```

1.20 pGadget.library/pOS_IsGadgetMember()

```
NAME
    pOS_IsGadgetMember -- Prüft, ob Gadget Member ist

SYNOPSIS
     BOOL pOS_IsGadgetMember(grpGad,membGad);
     _R_D0                   _R_A0   _R_A1

     BOOL pOS_IsGadgetMember(
                const struct pOS_Gadget*,const struct pOS_Gadget*);

FUNCTION
     Es wird ermittelt, ob 'membGad' innerhalb (direkt oder indirekt)
     in der Gruppe 'grpGad' steht.
```

```
      Grp1
      {
        Grp2
        {
          Gad21
          Gad22
        }
        Gad11
        Gad12
      }

      Gad11 und Gad12 sind Member von Grp1.
      Gad21 und Gad22 sind Member von Grp2 und Grp1.

INPUTS
      grpGad  - Gruppe
      membGad - zu prüfendes Gadget

RESULT
      TRUE => Gadget ist Member von 'grpGad' (direkt oder indirekt)

SEE ALSO
```