pLayerD

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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

pLayerD

1.1 pLayerD.doc

pLayer.library

```
pOS_AddLayerHook()            pOS_AddLayerInfoHook()
pOS_AndRectRegion()           pOS_AndRegionRegion()
pOS_AndRegionRegionLst()      pOS_BeginLayerUpdate()
pOS_BehindLayer()             pOS_CheckVisibleRect()
pOS_ClearRegion()             pOS_CloneRegion()
pOS_ConstructLayer()          pOS_ConstructLayerInfo()
pOS_ConstructRegion()         pOS_CopyLayerToSup()
pOS_CopySupToLayer()          pOS_CreateBehindLayer()
pOS_CreateChildLayer()        pOS_CreateLayerInfo()
pOS_CreateRegion()            pOS_CreateUpfrontLayer()
pOS_CreateViClipListFrom()    pOS_DamageLayerInfo()
pOS_DamageLayerInfoList()     pOS_DeleteLayer()
pOS_DeleteLayerInfo()         pOS_DeleteRegion()
pOS_DeleteViClipList()        pOS_DestructLayer()
pOS_DestructLayerInfo()       pOS_DestructRegion()
pOS_EndLayerUpdate()          pOS_InstallClipRegion()
pOS_LockClipLayer()           pOS_LockHideLayer()
pOS_LockLayer()               pOS_LockLayerInfo()
pOS_LockLayers()              pOS_MoveLayerBehindOf()
pOS_MoveRegion()              pOS_MoveRegionLst()
pOS_MoveSizeLayer()           pOS_NAndRectRegion()
pOS_NAndRegionRegion()        pOS_NewSizeRegion()
pOS_OrRectRegion()            pOS_OrRectRegionILst()
pOS_OrRegionRegion()          pOS_RemLayerHook()
pOS_RemLayerInfoHook()        pOS_ScrollLayer()
pOS_SortRegionLst()           pOS_UnlockClipLayer()
pOS_UnlockHideLayer()         pOS_UnlockLayer()
pOS_UnlockLayerInfo()         pOS_UnlockLayers()
pOS_UpfrontLayer()            pOS_WhichLayer()
pOS_XOrRectRegion()
```

1.2 pLayer.library/pOS_ConstructRegion()

```
NAME
    pOS_ConstructRegion -- initialisiert eine Struktur zur Verwaltung
                           von Flächen

SYNOPSIS
     pOS_ConstructRegion(region);
                         _R_A0

     VOID pOS_ConstructRegion(struct pOS_Region*);

FUNCTION
     Initialisiert die Verwaltungs-Struktur für Flächenstücken.
     In der Region können beliebig viele Flächenstücke verwaltet
     werden, die sich NIE überschneiden.
     Am Ende muß pOS_DestructRegion() aufgerufen werden, damit
     der gesamte Speicherplatz freigegeben wird.

INPUTS
     region - zu initialisierende Struktur

RESULT

WARNING
     Die Region-Daten sind 'Read-Only' und dürfen nur über die
     Library modifiziert werden.

SEE ALSO
     pOS_DestructRegion, pOS_CreateRegion
```

1.3 pLayer.library/pOS_DestructRegion()

```
NAME
    pOS_DestructRegion -- Verwaltung auflösen, Daten freigeben

SYNOPSIS
     pOS_DestructRegion(region);
                        _R_A0

     VOID pOS_DestructRegion(struct pOS_Region*);

FUNCTION
     Alle dynamisch erzeugen Daten wieder freigeben.
     'region' wird nicht freigegeben.

INPUTS
     region - aufzulösende Struktur

RESULT

SEE ALSO
     pOS_ConstructRegion, pOS_DeleteRegion
```

1.4 pLayer.library/pOS_CreateRegion()

```
NAME
    pOS_CreateRegion -- Erzeugen einer Region-Struktur

SYNOPSIS
     region = pOS_CreateRegion();
      _R_D0

     __ARID__ struct pOS_Region* pOS_CreateRegion();

FUNCTION
     Erzeugen und initialisieren einer Region-Struktur.

INPUTS

RESULT
     region - erzeuge Struktur oder NULL

SEE ALSO
     pOS_ConstructRegion, pOS_DeleteRegion
```

1.5 pLayer.library/pOS_DeleteRegion()

```
NAME
    pOS_DeleteRegion -- Löschen einer Region-Struktur

SYNOPSIS
     pOS_DeleteRegion(region);
                      _R_A0

     VOID pOS_DeleteRegion(__ARID__ struct pOS_Region*);

FUNCTION
     Die Region wird mit allen darin verwalteten Daten freigegeben.

INPUTS
     region - zu löschende Struktur

RESULT

SEE ALSO
     pOS_DestructRegion, pOS_CreateRegion
```

1.6 pLayer.library/pOS_OrRectRegionILst()

```
NAME
    pOS_OrRectRegionILst -- Rectangle zur Region addieren
```

```
SYNOPSIS
     BOOL pOS_OrRectRegionILst(region,rectangle,list);
     _R_D0                    _R_A0     _R_A1    _R_A2

     BOOL = pOS_OrRectRegionILst(struct pOS_Region*,
                                 const struct pOS_Rectangle*,
                                 struct pOS_List*);

FUNCTION
     Addiert eine Fläche zur Region. Alle neuen Flächenstücke werden
     in der Liste 'list' abgelegt. Die Funktion erzeugt nie überlappende
     Flächenstücke.

INPUTS
     region    - Struktur
     rectangle - zu addierende Fläche
     list      - Result, neue pOS_RegionRectangle

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.7 pLayer.library/pOS_OrRectRegion()

```
NAME
    pOS_OrRectRegion -- Rectangle zur Region addieren

SYNOPSIS
     BOOL pOS_OrRectRegion(region,rectangle);
     _R_D0                 _R_A0    _R_A1

     BOOL = pOS_OrRectRegion(struct pOS_Region*,const struct pOS_Rectangle*);

FUNCTION
     Addiert eine Fläche zur Region.

INPUTS
     region    - Struktur
     rectangle - zu addierende Fläche

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
     pOS_OrRectRegionILst,
```

1.8 pLayer.library/pOS_OrRegionRegion()

```
NAME
```

```
    pOS_OrRegionRegion -- Region auf Region addieren

SYNOPSIS
     BOOL pOS_OrRegionRegion(srcRegion,dstRegion);
     _R_D0                   _R_A0    _R_A1

     BOOL = pOS_OrRegionRegion(const struct pOS_Region*,struct pOS_Region*);

FUNCTION
     Addiert alle Flächenstücke von 'srcRegion' zu 'dstRegion'.

INPUTS
     srcRegion - Quell-Region, die addiert werden soll
     dstRegion - Ziel, diese Region wird erweitert

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
     pOS_OrRectRegionILst,
```

1.9 pLayer.library/pOS_AndRectRegion()

```
NAME
    pOS_AndRectRegion -- Bildet Schnittmenge

SYNOPSIS
     BOOL pOS_AndRectRegion(region,rectangle);
     _R_D0                  _R_A0    _R_A1

     BOOL = pOS_AndRectRegion(struct pOS_Region*,const struct pOS_Rectangle*);

FUNCTION
     Alle Flächenstückte, die außerhalb von 'rectangle' liegen, werden
     aus der Region 'region' herausgeschnitten.

INPUTS
     region    - Struktur
     rectangle - nur diese Fläche innerhalb der Region bleibt erhalten

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.10 pLayer.library/pOS_AndRegionRegion()

```
NAME
    pOS_AndRegionRegion -- Schnittmenge zweier Regionen bilden
```

```
SYNOPSIS
     BOOL pOS_AndRegionRegion(XRegion,region);
     _R_D0                  _R_A0    _R_A1

     BOOL = pOS_AndRegionRegion(const struct pOS_Region*,struct pOS_Region*);

FUNCTION
     Es werden alle Flächenstücke aus 'region' herausgeschnitten, die
     nicht ebenfalls in 'XRegion' enthalten sind.

INPUTS
     XRegion - Menge der zu vergleichenden Flächen
     region  - Schnittmenge

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.11 pLayer.library/pOS_NAndRectRegion()

```
NAME
    pOS_NAndRectRegion -- Fenster aus eine Region herausschneiden

SYNOPSIS
     BOOL pOS_NAndRectRegion(region,rectangle);
     _R_D0                  _R_A0    _R_A1

     BOOL = pOS_NAndRectRegion(struct pOS_Region*,const struct pOS_Rectangle*);

FUNCTION
     Die Fläche 'rectangle' wird aus der Region 'region' herausgeschnitten.

INPUTS
     region    - Struktur
     rectangle - herauszuschneidendes 'Fenster'

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.12 pLayer.library/pOS_MoveRegion()

```
NAME
    pOS_MoveRegion -- Region verschieben

SYNOPSIS
     pOS_MoveRegion(region, dx,   dy);
                    _R_A0  _R_D0 _R_D1
```

```
     pOS_MoveRegion(struct pOS_Region*,SLONG dx,SLONG dy);

FUNCTION
     Verschiebt alle Flächenstücke der Region.

INPUTS
     region - Struktur
     dx     - Verschiebung in horizontaler Richtung
     dy     - Verschiebung in vertikaler Richtung

RESULT

SEE ALSO
```

1.13 pLayer.library/pOS_MoveRegionLst()

```
NAME
    pOS_MoveRegionLst -- Verschieben von pOS_RegionRectangle

SYNOPSIS
     pOS_MoveRegionLst(list,dx,   dy);
                    _R_A0  _R_D0 _R_D1

     pOS_MoveRegionLst(struct pOS_List*,SLONG dx,SLONG dy);

FUNCTION
     Verschiebt alle Nodes der Liste.

INPUTS
     list - Liste vom Typ (struct pOS_RegionRectangle*)
     dx   - Verschiebung in horizontaler Richtung
     dy   - Verschiebung in vertikaler Richtung

RESULT

SEE ALSO
```

1.14 pLayer.library/pOS_NAndRegionRegion()

```
NAME
     pOS_NAndRegionRegion -- Region aus einer Region herausschneiden

SYNOPSIS
     BOOL pOS_NAndRegionRegion(XRegion,region);
     _R_D0                     _R_A0   _R_A1

     BOOL = pOS_NAndRegionRegion(const struct pOS_Region*,struct pOS_Region*);

FUNCTION
```

```
     Aus der Region 'region' werden alle Flächenstücke herausgeschnitten,
     die ebenfalls in 'XRegion' enthalten sind.

INPUTS
     XRegion - herauszuschneidene Region
     region  - zu verändernde Region

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.15 pLayer.library/pOS_XOrRectRegion()

```
NAME
     pOS_XOrRectRegion -- Invertiert eine Fläche.

SYNOPSIS
     BOOL pOS_XOrRectRegion(region,rectangle);
     _R_D0                  _R_A0  _R_A1

     BOOL = pOS_XOrRectRegion(struct pOS_Region*,const struct pOS_Rectangle*);

FUNCTION
     Die gesamte Fläche inner halb von 'rectangle' wird in 'region'
     invertiert. Vorgandene Flächenstücke werden entfernt und
     Lücken werden gefüllt.

INPUTS
     region    - Struktur
     rectangle - diese Fläche wird innerhalb von 'region' invertiert.

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
     pOS_OrRectRegionILst,
```

1.16 pLayer.library/pOS_ClearRegion()

```
NAME
     pOS_ClearRegion -- Alle Flächenstücke entfernen

SYNOPSIS
     pOS_ClearRegion(region,list);
                     _R_A0  _R_A1

     pOS_ClearRegion(struct pOS_Region*,struct pOS_List*);

FUNCTION
```

```
     Alle verwalteten Flächenstücke werden aus der Liste 'list'
     (Memeber of struct pOS_RegionRectangle*) gelöscht.
     Der Memory-Pool wird über 'region' referenziert.

INPUTS
     region - indirekte Referenz auf Memory-Pool
     list   - Liste von Nodes of (pOS_RegionRectangle*)

RESULT

SEE ALSO
```

1.17 pLayer.library/pOS_AndRegionRegionLst()

```
NAME
     pOS_AndRegionRegionLst -- Schnittmenge zweier Regionen bilden

SYNOPSIS
     BOOL pOS_AndRegionRegionLst(aList,bList,cList,pool);
     _R_D0                       _R_A0  _R_A1 _R_A2 _R_A3

     BOOL = pOS_AndRegionRegionLst(
              const struct pOS_List*, const struct pOS_List*,
              struct pOS_List*, struct pOS_MemPool*);

FUNCTION
     Es werden neue Nodes (struct pOS_RegionRectangle*) erzeugt, deren
     Fläche in 'aList' und 'bList' enthalten ist.

INPUTS
     aList - erste Menge der zu vergleichenden Flächen
             Node : (const struct pOS_RegionRectangle*)
     bList - zweite Menge der zu vergleichenden Flächen
             Node : (const struct pOS_RegionRectangle*)
     cList - Initialisierte leere Liste, die das Ergebnis aufnimmt.
     pool  - Memory-Pool für die neuen Flächenstücke (cList)
               (struct pOS_RegionRectangle*)

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
     pOS_OrRectRegionILst,
```

1.18 pLayer.library/pOS_NewSizeRegion()

```
NAME
     pOS_NewSizeRegion -- Berechnet die Region->Bounds neu

SYNOPSIS
```

```
     pOS_NewSizeRegion(region);
                       _R_A0

     pOS_NewSizeRegion(struct pOS_Region*);

FUNCTION
     In der Region wird die gesamte Fläche in rg_Bounds abgebildet.

INPUTS
     region - Struktur

RESULT

SEE ALSO
```

1.19 pLayer.library/pOS_SortRegionLst()

```
NAME
     pOS_SortRegionLst -- Flächenstücke sortieren

SYNOPSIS
     pOS_SortRegionLst(list,  dx,  dy);
                      _R_A0 _R_D0 _R_D1

     pOS_SortRegionLst(struct pOS_List*,SLONG dx,SLONG dy);

FUNCTION
     Sortiert alle Flächenstückte je nach 'dx' und 'dy'.

INPUTS
     list   - Liste der (struct pOS_RegionRectangle*) Nodes
     dx,dy  - Sortiervorschrift

RESULT

SEE ALSO
```

1.20 pLayer.library/pOS_CloneRegion()

```
NAME
     pOS_CloneRegion -- Flächenstücke kopieren

SYNOPSIS
     BOOL pOS_CloneRegion(list,region);
     _R_D0               _R_A0 _R_A1

     BOOL = pOS_CloneRegion(const struct pOS_List*,struct pOS_Region*);

FUNCTION
     Alle Flächenstücke aus der List 'list' werden kopiert und zu
```

```
     'region' addiert. Hierbei wird nicht geprüft, ob Überlappungen
     vorliegen. ACHTUNG: Wenn sich Flächenstücke überlappen ist die
     Region zerstört. Die Funktion macht nur Sinn, wenn die Region leer
     ist.

INPUTS
     list   - Liste der (const struct pOS_RegionRectangle*) Nodes
     region - Region für die neuen Nodes und MemPool zur Verwaltung

RESULT
     TRUE - Ok,  FALSE - Speichermangel

SEE ALSO
```

1.21 pLayer.library/pOS_ConstructLayerInfo()

```
NAME
     pOS_ConstructLayerInfo -- Initialisierung

SYNOPSIS
     VOID pOS_ConstructLayerInfo(LayerInfo);
                                 _R_A0

     pOS_ConstructLayerInfo(struct pOS_LayerInfo*);

FUNCTION
     Internal

INPUTS

RESULT

SEE ALSO
```

1.22 pLayer.library/pOS_DestructLayerInfo()

```
NAME
     pOS_DestructLayerInfo -- Daten freigeben

SYNOPSIS
     pOS_DestructLayerInfo(LayerInfo);
                           _R_A0

     pOS_DestructLayerInfo(struct pOS_LayerInfo*);

FUNCTION
     Internal

INPUTS
```

```
RESULT

SEE ALSO
```

1.23 pLayer.library/pOS_CreateLayerInfo()

```
NAME
     pOS_CreateLayerInfo -- LayerInfo erzeugen

SYNOPSIS
     LayerInfo = pOS_CreateLayerInfo();
      _R_D0

     __ARID__ struct pOS_LayerInfo* pOS_CreateLayerInfo();

FUNCTION
     Erzeugt eine LayerInfo-Struktur.

INPUTS

RESULT
     LayerInfo-Struktur oder NULL für Speichermangel

SEE ALSO
```

1.24 pLayer.library/pOS_DeleteLayerInfo()

```
NAME
     pOS_DeleteLayerInfo -- Löscht eine LayerInfo

SYNOPSIS
     pOS_DeleteLayerInfo(LayerInfo);
                         _R_A0

     VOID pOS_DeleteLayerInfo(__ARID__ struct pOS_LayerInfo*);

FUNCTION
     Die LayerInfo wird gelöscht. Zuvor muß sichergestellt werden,
     daß keine Layers mehr offen sind.

INPUTS
     LayerInfo - zu löschende Struktur

RESULT

SEE ALSO
```

1.25 pLayer.library/pOS_ConstructLayer()

```
NAME
     pOS_ConstructLayer -- Initalisierung

SYNOPSIS
     VOID pOS_ConstructLayer(Layer);
                             _R_A0

     VOID pOS_ConstructLayer(struct pOS_Layer*);

FUNCTION
     Internal

INPUTS

RESULT

SEE ALSO
```

1.26 pLayer.library/pOS_DestructLayer()

```
NAME
     pOS_DestructLayer -- Freigeben

SYNOPSIS
     VOID pOS_DestructLayer(Layer);
                             _R_A0

     VOID pOS_DestructLayer(struct pOS_Layer*);

FUNCTION
     Internal

INPUTS

RESULT

SEE ALSO
```

1.27 pLayer.library/pOS_CreateBehindLayer()

```
NAME
     pOS_CreateBehindLayer -- untenliegenden Layer erzeugen

SYNOPSIS
     Layer = pOS_CreateBehindLayer(LayerInfo,GfxMap,rectangle,flags,superGfx);
      _R_D0                         _R_A0     _R_A1  _R_A2     _R_D0   _R_A3

    __ARID__ struct pOS_Layer* pOS_CreateBehindLayer(
```

```
                  struct pOS_LayerInfo*, struct pOS_GfxMap*,
                  const struct pOS_Rectangle*, ULONG, struct pOS_GfxMap*);

FUNCTION
     Neuen Layer in 'LayerInfo' erzeugen, der ganz unten liegen soll.
     Der neue Layer zeichnet in 'GfxMap' und hat die Größe 'rectangle'.
     Wird 'superGfx' angegeben, so werden alle unsichtbaren Flächenteile
     in dieser Map zwischengespeichert. Die Applikation erhält keine
     IDCMP_RefreshWindow - Message.

INPUTS
     LayerInfo - Layer wird Member von LayerInfo
     GfxMap    - Der Layer soll in diese Map zeichnen
     rectangle - Startgröße des Layers
     flags     - (enum pOS_LayerFlags) Eigenschaften
     superGfx  - NULL oder eine eigene GfxMap

RESULT
     Layer Struktur oder NULL bei Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.28 pLayer.library/pOS_CreateUpfrontLayer()

```
NAME
     pOS_CreateUpfrontLayer -- obenliegenden Layer erzeugen

SYNOPSIS
     Layer = pOS_CreateUpfrontLayer(LayerInfo,GfxMap,rectangle,flags,superGfx);
      _R_D0                          _R_A0     _R_A1  _R_A2     _R_D0   _R_A3

    __ARID__ struct pOS_Layer* pOS_CreateUpfrontLayer(
                  struct pOS_LayerInfo*, struct pOS_GfxMap*,
                  const struct pOS_Rectangle*, ULONG, struct pOS_GfxMap*);

FUNCTION
     Neuen Layer in 'LayerInfo' erzeugen, der ganz oben liegen soll.
     Der neue Layer zeichnet in 'GfxMap' und hat die Größe 'rectangle'.
     Wird 'superGfx' angegeben, so werden alle unsichtbaren Flächenteile
     in dieser Map zwischengespeichert. Die Applikation erhält keine
     IDCMP_RefreshWindow - Message.

INPUTS
     LayerInfo - Layer wird Member von LayerInfo
     GfxMap    - Der Layer soll in diese Map zeichnen
     rectangle - Startgröße des Layers
     flags     - (enum pOS_LayerFlags) Eigenschaften
     superGfx  - NULL oder eine eigene GfxMap

RESULT
     Layer Struktur oder NULL bei Speichermangel
```

```
NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.29 pLayer.library/pOS_DeleteLayer()

```
NAME
     pOS_DeleteLayer -- Layer entfernen und Daten freigebene

SYNOPSIS
     BOOL pOS_DeleteLayer(Layer);
     _R_D0                _R_A0

     BOOL pOS_DeleteLayer(__ARID__ struct pOS_Layer*);

FUNCTION
     Ein zuvor mit pOS_CreateBehindLayer() oder pOS_CreateUpfrontLayer()
     erzeugten Layer wieder entfernen.

INPUTS
     Layer - zu löschenden Layer

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.30 pLayer.library/pOS_LockLayer()

```
NAME
     pOS_LockLayer -- Layer-Gfx-Daten locken

SYNOPSIS
     pOS_LockLayer(Layer);
                   _R_A0

     VOID pOS_LockLayer(struct pOS_Layer*);

FUNCTION
     Jeder Zugriff auf die Gfx-Layer-Daten muß mit
     dem Lock auf diesen Layer beginnen.
     Der Lock darf jedoch nur kurze Zeit bestehen.

INPUTS
     Layer - zu lockender Layer
```

```
RESULT

SEE ALSO
```

1.31 pLayer.library/pOS_UnlockLayer()

```
NAME
     pOS_UnlockLayer -- Layer-Gfx-Daten un-locken

SYNOPSIS
     pOS_UnlockLayer(Layer);
                     _R_A0

     VOID pOS_UnlockLayer(struct pOS_Layer*);

FUNCTION
     Ein zuvor gelockten Layer wieder freigeben.

INPUTS
     Layer - Struktur

RESULT

SEE ALSO
```

1.32 pLayer.library/pOS_LockLayerInfo()

```
NAME
     pOS_LockLayerInfo -- Layer-Verwaltung locken

SYNOPSIS
     pOS_LockLayerInfo(LayerInfo);
                       _R_A0

     VOID pOS_LockLayerInfo(struct pOS_LayerInfo*);

FUNCTION
     Vor dem Umbau eines oder mehrerer Layers muß die Layer-Verwaltung
     (LayerInfo) gelockt werden.
     Dieser Lock verhindert nicht, daß ein anderer Task in einen Layer
     zeichnen darf.

INPUTS
     LayerInfo - Struktur

RESULT

SEE ALSO
```

1.33 pLayer.library/pOS_UnlockLayerInfo()

```
NAME
     pOS_UnlockLayerInfo -- Layer-Verwaltung un-locken

SYNOPSIS
     pOS_UnlockLayerInfo(LayerInfo);
                         _R_A0

     VOID pOS_UnlockLayerInfo(struct pOS_LayerInfo*);

FUNCTION
     Eine zuvor gelockten LayerInfo wieder freigen.

INPUTS
     LayerInfo - Struktur

RESULT

SEE ALSO
```

1.34 pLayer.library/pOS_DamageLayerInfo()

```
NAME
     pOS_DamageLayerInfo -- Fläche als ungültig markieren

SYNOPSIS
     BOOL pOS_DamageLayerInfo(LayerInfo,rectangle,lyFirst,lyLast);
     _R_D0                    _R_A0     _R_A1     _R_A2    _R_A3

     BOOL pOS_DamageLayerInfo(
                struct pOS_LayerInfo*,const struct pOS_Rectangle*,
                const struct pOS_Layer*,const struct pOS_Layer*);

FUNCTION
     Alle Layers, die ab 'lyFirst' beginnen und 'rectangle' berühren
     werden an den Schnittflächen ungültig.
     Ein Simple-Refresh-Layer setzt das LAYERF_Refresh-Bit und
     wird u.U. die Damage-Liste erweitern.
     Beim Super-Layer wird sofort der ungültige Grafikbereich aus
     der SuperGfxMap umkopiert.

INPUTS
     LayerInfo - Struktur
     rectangle - zu markierende Fläche
     lyFirst   - ab diesem Layer mit dem Markieren beginnen
                 (NULL => nichts bearbeiten)
     lylast    - Abbruchbedingung, dieser Layer wird nicht mehr
                 bearbeitet (NULL => bis zum untersten Layer bearbeitet)

RESULT
     TRUE - Ok,  FALSE - Speichermangel
```

```
NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )

SEE ALSO
```

1.35 pLayer.library/pOS_DamageLayerInfoList()

```
NAME
     pOS_DamageLayerInfoList --

SYNOPSIS
     BOOL pOS_DamageLayerInfoList(LayerInfo,list, lyFirst,lyLast);
     _R_D0                        _R_A0     _R_A1  _R_A2    _R_A3


     BOOL pOS_DamageLayerInfoList(
                struct pOS_LayerInfo*,const struct pOS_List*,
                const struct pOS_Layer*,const struct pOS_Layer*);

FUNCTION
  Siehe pOS_DamageLayerInfo()

INPUTS
     LayerInfo - Struktur
     list      - zu markierende Flächen, Nodes (const pOS_RegionRectangle*)
     lyFirst   - ab diesem Layer mit dem Markieren beginnen
                 (NULL => nichts bearbeiten)
     lylast    - Abbruchbedingung, dieser Layer wird nicht mehr
                 bearbeitet (NULL => bis zum untersten Layer bearbeitet)

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )

SEE ALSO
```

1.36 pLayer.library/pOS_MoveSizeLayer()

```
NAME
     pOS_MoveSizeLayer -- Verschiebt und vergrößert den Layer

SYNOPSIS
     BOOL pOS_MoveSizeLayer(Layer,  dx,  dy,   dw,   dh);
     _R_D0                  _R_A0 _R_D0 _R_D1 _R_D2 _R_D3
```

```
     BOOL pOS_MoveSizeLayer(struct pOS_Layer*,SLONG,SLONG,SLONG,SLONG);

FUNCTION
     Verschiebt und vergrößert den Layer.

INPUTS
     Layer - zu ändernden Layer
     dx    - horizontale Verschiebung
     dy    - vertikale Verschiebung
     dw    - Breitenänderung
     dh    - Höhenänderung

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.37 pLayer.library/pOS_WhichLayer()

```
NAME
     pOS_WhichLayer -- Ermittelt Layer an Position

SYNOPSIS
     Layer = pOS_WhichLayer(LayerInfo, x,   y);
     _R_D0                  _R_A0    _R_D0 _R_D1

     struct pOS_Layer* pOS_WhichLayer(struct pOS_LayerInfo*,SLONG,SLONG);

FUNCTION
     Ermittelt den Layer, an Position (x,y) sichtbar ist.

INPUTS
     LayerInfo - Struktur
     x,y       - zu ermittelnde Position

RESULT
     Layer an der Position (x,y) oder NULL

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.38 pLayer.library/pOS_LockLayers()

```
NAME
```

```
     pOS_LockLayers -- Alles Locken

SYNOPSIS
     pOS_LockLayers(LayerInfo);
                    _R_A0

     VOID pOS_LockLayers(struct pOS_LayerInfo*);

FUNCTION
     Alle Layers werden gelockt, keiner kann Ausgaben oder
     Veränderungen vornehmen.
     Dies wird in einigen Fällen benötigt, wie Layer verschieben,
      Layer-Tiefe ändern, Layer erzeugen, Layer löschen.

INPUTS
     LayerInfo - Struktur

RESULT

SEE ALSO
```

1.39 pLayer.library/pOS_UnlockLayers()

```
NAME
     pOS_UnlockLayers -- Alles un-locken

SYNOPSIS
     pOS_UnlockLayers(LayerInfo);
                      _R_A0

     VOID pOS_UnlockLayers(struct pOS_LayerInfo*);

FUNCTION
     Zuvor gelockte Layers ( pOS_LockLayers ) wieder un-locken.

INPUTS
     LayerInfo - Struktur

RESULT

SEE ALSO
```

1.40 pLayer.library/pOS_BehindLayer()

```
NAME
     pOS_BehindLayer -- Layer in den Hintergrund

SYNOPSIS
     BOOL pOS_BehindLayer(Layer);
     _R_D0                _R_A0
```

```
     BOOL pOS_BehindLayer(struct pOS_Layer*);

FUNCTION
     Layer ganz nach unten verschieben.

INPUTS
     Layer - nach unten zu bringenden Layer

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.41 pLayer.library/pOS_UpfrontLayer()

```
NAME
     pOS_UpfrontLayer -- Layer in den Vordergrund

SYNOPSIS
     BOOL pOS_UpfrontLayer(Layer);
     _R_D0                 _R_A0

     BOOL pOS_UpfrontLayer(struct pOS_Layer*);

FUNCTION
     Der Layer wird ganz nach vorne gebracht.

INPUTS
     Layer - in den Vordergrund zu bringenden Layer

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.42 pLayer.library/pOS_AddLayerHook()

```
NAME
     pOS_AddLayerHook -- neuen Callback aktivieren

SYNOPSIS
     pOS_AddLayerHook(Layer,Callback,drawMode);
                      _R_A0   _R_A1    _R_D0
```

```
     VOID pOS_AddLayerHook(struct pOS_Layer*,struct pOS_Callback*,ULONG);

FUNCTION

INPUTS
     Layer    - betreffenden Layer
     Callback - Funktionsbeschreibung
     drawMode - 0 == ohne Refresh, 1 == Callback wird sofort gezeichnet

RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.43 pLayer.library/pOS_RemLayerHook()

```
NAME
     pOS_RemLayerHook -- Callback entfernen

SYNOPSIS
     pOS_RemLayerHook(Layer,Callback);
                      _R_A0   _R_A1

     VOID pOS_RemLayerHook(struct pOS_Layer*,struct pOS_Callback*);

FUNCTION

INPUTS
     Layer    - betreffenden Layer
     Callback - Funktionsbeschreibung
     drawMode - 0 == ohne Refresh, 1 == Layer wird sofort gezeichnet

RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.44 pLayer.library/pOS_BeginLayerUpdate()

```
NAME
     pOS_BeginLayerUpdate -- Berechnet den zu Refreshenden Bereich

SYNOPSIS
     BOOL pOS_BeginLayerUpdate(Layer,list);
     _R_D0                     _R_A0   _R_A1
```

```
     BOOL pOS_BeginLayerUpdate(struct pOS_Layer*,const struct pOS_List*);

FUNCTION
     Aus der Damage-Liste wird berechnet, welche Layer-Fläche erneuert
     werden muß. Damit nur diese Fläche gezeichnet wird, wird zwischen
     pOS_BeginLayerUpdate() und pOS_EndLayerUpdate() ein spezielles
     Clipping eingesetzt.
     Durch zusätzliche Angabe einer ClipListe 'list' wird der zu
     zeichnende Bereich mit der Liste AND - Verknüpft.

INPUTS
     Layer - Refresh Layer
     list  - NULL oder Nodes (const struct pOS_RegionRectangle*)

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.45 pLayer.library/pOS_EndLayerUpdate()

```
NAME
     pOS_EndLayerUpdate -- Normale Darstellung aktivieren

SYNOPSIS
     BOOL pOS_EndLayerUpdate(Layer,done);
     _R_D0                   _R_A0   _R_D0

     BOOL pOS_EndLayerUpdate(struct pOS_Layer*,ULONG);

FUNCTION
     Beendet den Refresh-Vorgang.
     Z.B.
      pOS_BeginLayerUpdate(ly,NULL)
        ... Draw
      pOS_EndLayerUpdate(ly,TRUE)

     oder

      pOS_BeginLayerUpdate(ly,NULL)
        ... Draw - 1
      pOS_EndLayerUpdate(ly,FALSE)

      pOS_BeginLayerUpdate(ly,List)
        ... Draw - 2
      pOS_EndLayerUpdate(ly,TRUE)

INPUTS
     Layer -
     done  - TRUE:  Refresh-Vorgang ist fertig, alle Damage-Daten werden
```

```
                    gelöscht.
               FALSE: Die Damage-Daten bleiben erhalten.
RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.46 pLayer.library/pOS_CreateViClipListFrom()

```
NAME
     pOS_CreateViClipListFrom -- Berechnung der View-Clips

SYNOPSIS
     BOOL pOS_CreateViClipListFrom(LayerInfo,firstLy,breakMode,refRect);
     _R_D0                         _R_A0   _R_A1      _R_D0    _R_A2

     BOOL pOS_CreateViClipListFrom(
                struct pOS_LayerInfo*,const struct pOS_Layer*,
                BOOL,const struct pOS_Rectangle*);

FUNCTION

INPUTS
     LayerInfo - Struktur
     firstLy   - ab diesem Layer beginnen (darf nicht null sein)
     breakMode - TRUE: nur den firstLayer bearbeiten
     refRect   - NULL oder Filtervorschrift

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )

SEE ALSO
```

1.47 pLayer.library/pOS_DeleteViClipList()

```
NAME
     pOS_DeleteViClipList -- Löschen von View-Clips

SYNOPSIS
     pOS_DeleteViClipList(LayerInfo,firstLy,breakMode);
                          _R_A0   _R_A1     _R_D0
```

```
     VOID pOS_DeleteViClipList(
                struct pOS_LayerInfo*,const struct pOS_Layer*,BOOL);

FUNCTION

INPUTS
     LayerInfo - Struktur
     firstLy   - ab diesem Layer beginnen (darf nicht null sein)
     breakMode - TRUE: nur den firstLayer bearbeiten

RESULT

NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )

SEE ALSO
```

1.48 pLayer.library/pOS_InstallClipRegion()

```
NAME
     pOS_InstallClipRegion -- Layer zusätzlich Clippen

SYNOPSIS
     region = pOS_InstallClipRegion(Layer,region);
      _R_D0                         _R_A0   _R_A1

    struct pOS_Region *pOS_InstallClipRegion(
                struct pOS_Layer*,const struct pOS_Region*);

FUNCTION

INPUTS
     Layer  - zu bearbeitender Layer
     region - neue Clip-Region oder NULL => kein Clipping

RESULT
     Zuvor gesetzte Region, kann NULL sein

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.49 pLayer.library/pOS_CreateChildLayer()

```
NAME
     pOS_CreateChildLayer --

SYNOPSIS
```

```
     Layer = pOS_CreateChildLayer(GfxMap,rectangle,flags,layer,superGfx);
      _R_D0                      _R_A0   _R_A1     _R_D0 _R_A2  _R_A3

     __ARID__ struct pOS_Layer* pOS_CreateChildLayer(
                struct pOS_GfxMap*,const struct pOS_Rectangle*,
                ULONG,struct pOS_Layer*,struct pOS_GfxMap*);

FUNCTION
     Siehe pOS_CreateUpfrontLayer

INPUTS
     GfxMap    - Der Layer soll in diese Map zeichnen
     rectangle - Startgröße des Layers. Tochter-Layer MUSS innerhalb
                 der Mutter liegen.
     flags     - (enum pOS_LayerFlags) Eigenschaften
     layer     - Mutter-Layer
     superGfx  - NULL oder eine eigene GfxMap

RESULT
     neuer Layer oder NULL bei Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.50 pLayer.library/pOS_MoveLayerBehindOf()

```
NAME
     pOS_MoveLayerBehindOf -- Layer-Tiefe gezielt ändern

SYNOPSIS
     BOOL pOS_MoveLayerBehindOf(layer,refLayer);
      _R_D0                     _R_A0   _R_A1

     BOOL pOS_MoveLayerBehindOf(struct pOS_Layer*,const struct pOS_Layer*);

FUNCTION
     Der Layer wird in der Tiefe hinter 'refLayer' gelegt.

INPUTS
     layer    - zu verschiebender Layer
     refLayer - Referenz-Layer

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.51 pLayer.library/pOS_ScrollLayer()

```
NAME
     pOS_ScrollLayer -- Layer-Grafik wird verschoben

SYNOPSIS
     pOS_ScrollLayer(Layer, dx,  dy);
                     _R_A0 _R_D0 _R_D1

     VOID pOS_ScrollLayer(struct pOS_Layer*,SLONG dx,SLONG dy);

FUNCTION
     Die Grafik wird innerhalb vom Layer verschoben. Der Verschiebe-
     Offset kann aus (ly_ScrollX, ly_ScrollY) ausgelesen werden.
     Nie schreibend auf ly_Scroll zugreifen. Sämtliche Grafikoperationen
     werten die Layer-Verschiebung aus und sorgen somit für eine echte
     Verschiebung der Grafik. Ein Mausklick wird ebenfalls über die
     Offsets berechnet.

INPUTS
     Layer - zu bearbeitenden Layer
     dx,dy - Pixel-Verschiebung

RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.52 pLayer.library/pOS_CopySupToLayer()

```
NAME
     pOS_CopySupToLayer -- Grafik von SuperMap in Layer kopieren

SYNOPSIS
     pOS_CopySupToLayer(Layer);
                        _R_A0

     VOID pOS_CopySupToLayer(struct pOS_Layer*);

FUNCTION

INPUTS
     Layer - Struktur

RESULT

NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )
```

```
SEE ALSO
```

1.53 pLayer.library/pOS_CopyLayerToSup()

```
NAME
     pOS_CopyLayerToSup -- Grafik von Layer in SuperMap kopieren

SYNOPSIS
     pOS_CopyLayerToSup(Layer);
                        _R_A0

     VOID pOS_CopyLayerToSup(struct pOS_Layer*);

FUNCTION

INPUTS
     Layer - Struktur

RESULT

NOTE
     Beim Funktionsaufruf müssen alle Layers und die LayerInfo
     gelockt sein. ( pOS_LockLayers() )

SEE ALSO
```

1.54 pLayer.library/pOS_LockHideLayer()

```
NAME
     pOS_LockHideLayer -- Layer locken und verbergen

SYNOPSIS
     BOOL pOS_LockHideLayer(Layer,LayerHide,mode);
     _R_D0                  _R_A0  _R_A1    _R_D0

     BOOL pOS_LockHideLayer(struct pOS_Layer*,struct pOS_LayerHide*,ULONG);

FUNCTION

INPUTS
     Layer     - Struktur
     LayerHide - Zwischenspeicher
     mode      -

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.
```

```
SEE ALSO
```

1.55 pLayer.library/pOS_UnlockHideLayer()

```
NAME
     pOS_UnlockHideLayer -- Layer wieder herstellen

SYNOPSIS
     VOID pOS_UnlockHideLayer(Layer,LayerHide);
                              _R_A0  _R_A1

     VOID pOS_UnlockHideLayer(struct pOS_Layer*,struct pOS_LayerHide*);

FUNCTION

INPUTS
     Layer     - Struktur
     LayerHide - Zwischenspeicher

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.56 pLayer.library/pOS_CheckVisibleRect()

```
NAME
     pOS_CheckVisibleRect -- Ermittelt, ob Fläche sichtbar ist

SYNOPSIS
     BOOL pOS_CheckVisibleRect(layer,rectangle);
     _R_D0                     _R_A0  _R_A1

     BOOL pOS_CheckVisibleRect(
                const struct pOS_Layer*,const struct pOS_Rectangle*);

FUNCTION
     Ermittelt, ob die Fläche 'rectangle' ganz oder teilweise
     sichtbar ist. Liefert die Funktion FALSE, so hat eine Zeichen-
     Operation innerhalb von 'rectangle' keine visuelle Wirkung.

INPUTS
     layer     - zu prüfender Layer
     rectangle - zu prüfende Fläche

RESULT
     TRUE sichtbar, FALSE vollständig unsichtbar
```

```
NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.57 pLayer.library/pOS_AddLayerInfoHook()

```
NAME
     pOS_AddLayerInfoHook -- LayerInfo Callback einfügen

SYNOPSIS
     pOS_AddLayerInfoHook(LayerInfo,Callback,drawMode);
                          _R_A0      _R_A1    _R_D0

     VOID pOS_AddLayerInfoHook(
                struct pOS_LayerInfo*,struct pOS_Callback*,ULONG);

FUNCTION

INPUTS
     LayerInfo - Struktur
     Callback  - Funktionsbeschreibung
     drawMode  - 0 == nicht zeichnen, 1 == sofort zeichnen

RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.58 pLayer.library/pOS_RemLayerInfoHook()

```
NAME
     pOS_RemLayerInfoHook -- LayerInfo Callback entfernen

SYNOPSIS
     pOS_RemLayerInfoHook(LayerInfo,Callback,drawMode);
                          _R_A0      _R_A1    _R_D0

     VOID pOS_RemLayerInfoHook(
                struct pOS_LayerInfo*,struct pOS_Callback*,ULONG);

FUNCTION

INPUTS
     LayerInfo - Struktur
     Callback  - Funktionsbeschreibung
     drawMode  - 0 == nicht zeichnen, 1 == sofort zeichnen
```

```
RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.59 pLayer.library/pOS_LockClipLayer()

```
NAME
     pOS_LockClipLayer -- Layer schnell Clippen

SYNOPSIS
     BOOL pOS_LockClipLayer(Layer,LayerClip,List,  mode);
     _R_D0                  _R_A0   _R_A1   _R_A2  _R_D0

     BOOL pOS_LockClipLayer(
                struct pOS_Layer*,struct pOS_LayerClip*,
                const struct pOS_List*,ULONG);

FUNCTION

INPUTS
     Layer     - zu clippenden Layer
     LayerClip - Hilfsdaten
     List      - Liste mit den neuen (const pOS_RegionRectangle*)
     mode      - 0

RESULT
     TRUE - Ok,  FALSE - Speichermangel

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```

1.60 pLayer.library/pOS_UnlockClipLayer()

```
NAME
     pOS_UnlockClipLayer -- schnell geclippten Layer wiederherstellen

SYNOPSIS
     pOS_UnlockClipLayer(Layer,LayerClip);
                         _R_A0   _R_A1

     VOID pOS_UnlockClipLayer(struct pOS_Layer*,struct pOS_LayerClip*);

FUNCTION
```

```
INPUTS
     Layer     - zu clippenden Layer
     LayerClip - Hilfsdaten

RESULT

NOTE
     Sämtliche Semaphoren werden von dieser Funktion beachtet.

SEE ALSO
```