graphics

COLLABORATORS			
	TITLE : graphics		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 23, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

graphics

1.1 graphics.doc

AddAnimOb()
AddBob()
AddFont()
AddVSprite()
AllocBitMap()
AllocDBufInfo()
AllocRaster()
AllocSpriteDataA()
AndRectRegion()
AndRegionRegion()
Animate()
AreaCircle()
AreaDraw()
AreaEllipse()
AreaEnd()
AreaMove()
AskFont()
AskSoftStyle()
AttachPalExtra()
AttemptLockLayerRom()
BestModeIDA()
BitMapScale()
BltBitMap()
BltBitMapRastPort()
BltClear()
BltMaskBitMapRastPort()
BltPattern()
BltTemplate()
CalcIVG()
CBump()
CEND
ChangeExtSpriteA()
ChangeSprite()
ChangeVPBitMap()
CINIT
ClearEOL()
ClearRectRegion()
ClearRegion()

```
ClearScreen()
ClipBlit()
CloseFont()
CloseMonitor()
CMOVE
CoerceMode()
CopySBitMap()
CWAIT
DisownBlitter()
DisposeRegion()
DoCollision()
Draw()
DrawEllipse()
DrawGList()
EraseRect()
ExtendFont()
FindColor()
FindDisplayInfo()
Flood()
FontExtent()
FreeBitMap()
FreeColorMap()
FreeCopList()
FreeCprList()
FreeDBufInfo()
FreeGBuffers()
FreeRaster()
FreeSprite()
FreeSpriteData()
FreeVPortCopLists()
GetAPen()
GetBitMapAttr()
GetBPen()
GetColorMap()
GetDisplayInfoData()
GetDrMd()
GetExtSpriteA()
GetGBuffers()
GetOPen()
GetRGB32()
GetRGB4()
GetRPAttrA()
GetSprite()
GetVPModeID()
GfxAssociate()
GfxFree()
GfxLookUP()
GfxNew()
InitArea()
InitBitMap()
InitGels()
InitGMasks()
InitMasks()
InitRastPort()
InitTmpRas()
InitView()
InitVPort()
```

```
LoadRGB32()
LoadRGB4()
LoadView()
LockLayerRom()
MakeVPort()
ModeNotAvailable()
Move()
MoveSprite()
MrgCop()
NewRegion()
NextDisplayInfo()
ObtainBestPenA()
ObtainPen()
OpenFont()
OpenMonitor()
OrRectRegion()
OrRegionRegion()
OwnBlitter()
PolyDraw()
QBlit()
QBSBlit()
ReadPixel()
ReadPixelArray8()
ReadPixelLine8()
RectFill()
ReleasePen()
RemBob()
RemFont()
RemIBob()
RemVSprite()
ScalerDiv()
ScrollRaster()
ScrollRasterBF()
ScrollVPort()
SetABPenDrMd()
SetAPen()
SetBPen()
SetChipRev()
SetCollision()
SetDrMd()
SetFont()
SetMaxPen()
SetOPen()
SetOutlinePen()
SetRast()
SetRGB32()
SetRGB32CM()
SetRGB4()
SetRGB4CM()
SetRPAttrA()
SetSoftStyle()
SetWriteMask()
SortGList()
StripFont()
SyncSBitMap()
Text()
TextExtent()
```

```
   TextFit()
   TextLength()
   UnlockLayerRom()
   VBeamPos()
   VideoControl()
   WaitBlit()
   WaitBOVP()
   WaitTOF()
   WriteChunkyPixels()
   WritePixel()
   WritePixelArray8()
   WritePixelLine8()
   XorRectRegion()
   XorRegionRegion()
```

1.2 graphics.library/AddAnimOb

```
 NAME
AddAnimOb  --  Add an AnimOb to the linked list of AnimObs.

 SYNOPSIS
AddAnimOb(anOb, anKey, rp)
          A0    A1     A2

void AddAnimOb(struct AnimOb *,struct AnimOb **, struct RastPort *);

 FUNCTION
Links this AnimOb into the current list pointed to by animKey.
Initializes all the Timers of the AnimOb's components.
Calls AddBob with each component's Bob.
rp->GelsInfo must point to an initialized GelsInfo structure.

 INPUTS
anOb  = pointer to the AnimOb structure to be added to the list
anKey = address of a pointer to the first AnimOb in the list
        (anKey = NULL if there are no AnimObs in the list so far)
rp    = pointer to a valid RastPort

 RESULT

 BUGS

 SEE ALSO
Animate() graphics/rastport.h graphics/gels.h
```

1.3 graphics.library/AddBob

```
 NAME
AddBob -- Adds a Bob to current gel list.

 SYNOPSIS
AddBob(Bob, rp)
```

```
       A0   A1

void AddBob(struct Bob *, struct RastPort *);

 FUNCTION
Sets up the system Bob flags, then links this gel into the list
via AddVSprite.

 INPUTS
Bob = pointer to the Bob structure to be added to the gel list
rp  = pointer to a RastPort structure

 RESULT

 BUGS

 SEE ALSO
InitGels()  AddVSprite()  graphics/gels.h  graphics/rastport.h
```

1.4 graphics.library/AddFont

```
 NAME
AddFont -- add a font to the system list

 SYNOPSIS
AddFont(textFont)
        A1

void AddFont(struct TextFont *);

 FUNCTION
This function adds the text font to the system, making it
available for use by any application.  The font added must be
in public memory, and remain until successfully removed.

 INPUTS
textFont - a TextFont structure in public ram.

 RESULT

 NOTES
This function will set the tf_Accessors to 0.

 BUGS

 SEE ALSO
SetFont()  RemFont()  graphics/text.h
```

1.5 graphics.library/AddVSprite

```
 NAME
AddVSprite -- Add a VSprite to the current gel list.
```

```
 SYNOPSIS
AddVSprite(vs, rp)
           A0  A1

void AddVSprite(struct VSprite *, struct RastPort *);

 FUNCTION
Sets up the system VSprite flags
Links this VSprite into the current gel list using its Y,X

 INPUTS
vs = pointer to the VSprite structure to be added to the gel list
rp = pointer to a RastPort structure

 RESULT

 BUGS

 SEE ALSO
InitGels()  graphics/rastport.h  graphics/gels.h
```

1.6 graphics.library/AllocBitMap

```
 NAME
AllocBitMap -- Allocate a bitmap and attach bitplanes to it. (V39)


 SYNOPSIS
bitmap=AllocBitMap(sizex,sizey,depth, flags, friend_bitmap)
                   d0    d1    d2     d3       a0

struct BitMap *AllocBitMap(ULONG,ULONG,ULONG,ULONG, struct BitMap *);

 FUNCTION
Allocates and initializes a bitmap structure. Allocates and initializes
bitplane data, and sets the bitmap's planes to point to it.

 INPUTS
sizex = the width (in pixels) desired for the bitmap data.

sizey = the height (in pixels) desired.

depth = the number of bitplanes deep for the allocation.
  Pixels with AT LEAST this many bits will be allocated.

flags = BMF_CLEAR to specify that the allocated raster should be
        filled with color 0.

        BMF_DISPLAYABLE to specify that this bitmap data should
        be allocated in such a manner that it can be displayed.
        Displayable data has more severe alignment restrictions
        than non-displayable data in some systems.

        BMF_INTERLEAVED tells graphics that you would like your
```

```
        bitmap to be allocated with one large chunk of display
        memory for all bitplanes. This minimizes color flashing
        on deep displays. If there is not enough contiguous RAM
  for an interleaved bitmap, graphics.library will fall
  back to a non-interleaved one.

        BMF_MINPLANES causes graphics to only allocate enough space
        in the bitmap structure for "depth" plane pointers. This
  is for system use and should not be used by applications use
  as it is inefficient, and may waste memory.

friend_bitmap = pointer to another bitmap, or NULL. If this pointer
        is passed, then the bitmap data will be allocated in
        the most efficient form for blitting to friend_bitmap.

 BUGS

 NOTES
When allocating using a friend bitmap, it is not safe to assume
anything about the structure of the bitmap data if that friend
BitMap might not be a standard amiga bitmap (for instance, if
the workbench is running on a non-amiga display device, its
Screen->RastPort->BitMap won't be in standard amiga format.
The only safe operations to perform on a non-standard BitMap are:

        - blitting it to another bitmap, which must be either a standard
          Amiga bitmap, or a friend of this bitmap.

        - blitting from this bitmap to a friend bitmap or to a standard
          Amiga bitmap.

        - attaching it to a rastport and making rendering calls.

Good arguments to pass for the friend_bitmap are your window's
RPort->BitMap, and your screen's RastPort->BitMap.
Do NOT pass &(screenptr->BitMap)!

BitMaps not allocated with BMF_DISPLAYABLE may not be used as
Intuition Custom BitMaps or as RasInfo->BitMaps.  They may be blitted
to a BMF_DISPLAYABLE BitMap, using one of the BltBitMap() family of
functions.

 SEE ALSO
FreeBitMap()
```

1.7 graphics.library/AllocDBufInfo

```
 NAME
     AllocDBufInfo -- Allocate structure for multi-buffered animation (V39)

 SYNOPSIS
     AllocDBufInfo(vp)
        a0

struct DBufInfo * AllocDBufInfo(struct ViewPort *)
```

```
   FUNCTION
  Allocates a structure which is used by the ChangeVPBitMap()
  routine.

   INPUTS
       vp  =  A pointer to a ViewPort structure.

   BUGS

   NOTES
  Returns 0 if there is no memory available or if the display mode
  of the viewport does not support double-buffering.

  The only fields of the DBufInfo structure which can be used by application
  programs are the dbi_SafeMessage, dbi_DispMessage, dbi_UserData1 and
  dbi_UserData2 fields.

  dbi_SafeMessage and dbi_DispMessage are standard exec message structures
  which may be used for synchronizing your animation with the screen update.

  dbi_SafeMessage is a message which is replied to when it is safe to write to
  the old BitMap (the one which was installed when you called ChangeVPBitMap).

  dbi_DispMessage is replied to when it is safe to call ChangeVPBitMap again
  and be certain that the new frame has been seen at least once.

  The dbi_UserData1 and dbi_UserData2 fields, which are stored after each
  message, are for your application to stuff any data into that it may need
  to examine when looking at the reply coming into the ReplyPort for either
  of the embedded Message structures.

  DBufInfo structures MUST be allocated with this function. The size of
  the structure will grow in future releases.

  The following fragment shows proper double buffering synchronization:

  int SafeToChange=TRUE, SafeToWrite=TRUE, CurBuffer=1;
  struct MsgPort *ports[2];    /* reply ports for DispMessage and SafeMessage
*/
  struct BitMap *BmPtrs[2];
  struct DBufInfo *myDBI;

  ... allocate bitmap pointers, DBufInfo, set up viewports, etc.

  myDBI->dbi_SafeMessage.mn_ReplyPort=ports[0];
  myDBI->dbi_DispMessage.mn_ReplyPort=ports[1];
  while (! done)
  {
      if (! SafeToWrite)
    while(! GetMsg(ports[0])) Wait(1l<<(ports[0]->mp_SigBit));
      SafeToWrite=TRUE;

      ... render to bitmap # CurBuffer.

      if (! SafeToChange)
    while(! GetMsg(ports[1])) Wait(1l<<(ports[1]->mp_SigBit));
```

```
    SafeToChange=TRUE;
    WaitBlit();         /* be sure rendering has finished */
    ChangeVPBitMap(vp,BmPtrs[CurBuffer],myDBI);
    SafeToChange=FALSE;
    SafeToWrite=FALSE;
    CurBuffer ^=1;  /* toggle current buffer */
}
     if (! SafeToChange)  /* cleanup pending messages */
    while(! GetMsg(ports[1])) Wait(1l<<(ports[1]->mp_SigBit));
     if (! SafeToWrite) /* cleanup */
    while(! GetMsg(ports[0])) Wait(1l<<(ports[0]->mp_SigBit));

 SEE ALSO
FreeDBufInfo() ChangeVPBitMap()
```

1.8 graphics.library/AllocRaster

```
 NAME
AllocRaster -- Allocate space for a bitplane.

 SYNOPSIS
planeptr = AllocRaster( width, height )
   d0                    d0     d1

PLANEPTR AllocRaster(ULONG,ULONG);

 FUNCTION
This function calls the memory allocation routines
to allocate memory space for a bitplane "width" bits
wide and "height" bits high.

 INPUTS
width - number of columns in bitplane
height  - number of rows in bitplane

 RESULT
planeptr - pointer to first word in bitplane, or NULL if
     it was not possible to allocate the desired
     amount of memory.

 NOTES
In order to assure proper alignment of display memory, the
AllocBitMap() function should be used instead of AllocRaster
when you wish to allocate display memory (rasters which are
attached to a ViewPort or Screen).

 BUGS

 SEE ALSO
FreeRaster() graphics/gfx.h
```

1.9 graphics.library/AllocSpriteDataA

```
 NAME
AllocSpriteDataA -- allocate sprite data and convert from a bitmap. (V39)
AllocSpriteData -- varargs stub for AllocSpriteData(). (V39)

 SYNOPSIS
SpritePtr | 0 = AllocSpriteDataA(bitmap,taglist)
 d0                   a2      a1

struct ExtSprite *AllocSpriteDataA( struct BitMap *, struct TagItem * );

extsprite=AllocSpriteData(bitmap,tags,...TAG_END)

  FUNCTION
Allocate memory to hold a sprite image, and convert the passed-in
bitmap data to the appropriate format. The tags allow specification
of width, scaling, and other options.

 INPUTS
bitmap - ptr to a bitmap. This bitmap provides the source data for the
  sprite image.

tags -
  SPRITEA_Width specifies how many pixels wide you desire
  the sprite to be. Specifying a width wider than the hardware
  can handle will cause the function to return failure. If the
  bitmap passed in is narrower than the width asked for, then
  it will be padded on the right with transparent pixels.
  Defaults to 16.

  SPRITEA_XReplication controls the horizontal pixel replication factor
  used when converting the bitmap data. Valid values are:
    0 - perform a 1 to 1 conversion
    1 - each pixel from the source is replicated twice
        in the output.
    2 - each pixel is replicated 4 times.
         -1 - skip every other pixel in the source bitmap
         -2 - only include every fourth pixel from the source.

    This tag is useful for converting data from one resolution
  to another. For instance, hi-res bitmap data can be correctly
  converted for a lo-res sprite by using an x replication factor
  of -1. Defaults to 0.

  SPRITEA_YReplication controls the vertical pixel replication factor
  in the same manner as SPRITEA_XReplication controls the horizontal.

  SPRITEA_OutputHeight specifies how tall the resulting sprite
  should be. Defaults to the bitmap height. The bitmap MUST be at
  least as tall as the output height.

  SPRITEA_Attached tells the function that you wish to convert
  the data for the second sprite in an attached sprite pair.
  This will cause AllocSpriteData() to take its data from the
  3rd and 4th bitplanes of the passed in bitmap.
```

```
Bitplane data is not required to be in chip ram for this function.


 RESULTS
SpritePtr = a pointer to a ExtSprite structure, or 0 if there is
a failure. You should pass this pointer to FreeSpriteData() when finished
with the sprite.

 BUGS
  Under V39, the appropriate attach bits would not be set in the sprite
data.
  The work-around is to set the bits manually. Bit 7 of the second
word should be set. On a 32 bit sprite, bit 7 of the 3rd word should
also be set. For a 64 bit sprite, bit 7 of the 5th word should also be
set. This should NOT be done under V40, as the bug is fixed.

 SEE ALSO
FreeSpriteData() FreeSprite() ChangeSprite() MoveSprite() GetExtSpriteA()
AllocBitMap() graphics/sprite.h
```

1.10 graphics.library/AndRectRegion

```
 NAME
AndRectRegion -- Perform 2d AND operation of rectangle
     with region, leaving result in region.

 SYNOPSIS
AndRectRegion(region,rectangle)
    a0  a1

void AndRectRegion( struct Region *, struct Rectangle * );

 FUNCTION
Clip away any portion of the region that exists outside
of the rectangle. Leave the result in region.

 INPUTS
region - pointer to Region structure
rectangle - pointer to Rectangle structure

 NOTES
Unlike the other rect-region primitives, AndRectRegion() cannot
fail.

 BUGS

 SEE ALSO
AndRegionRegion() OrRectRegion() graphics/regions.h
```

1.11 graphics.library/AndRegionRegion

```
 NAME
     AndRegionRegion -- Perform 2d AND operation of one region
                     with second region, leaving result in second region.

 SYNOPSIS
     status = AndRegionRegion(region1,region2)
        d0                         a0      a1

BOOL AndregionRegion(struct Region *, struct Region * );

 FUNCTION
     Remove any portion of region2 that is not in region1.

 INPUTS
     region1 - pointer to Region structure
     region2 - pointer to Region structure to use and for result

 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS

 SEE ALSO
OrRegionRegion() AndRectRegion() graphics/regions.h
```

1.12 graphics.library/Animate

```
 NAME
Animate  --  Processes every AnimOb in the current animation list.

 SYNOPSIS
Animate(anKey, rp)
        A0     A1

void Animate(struct AnimOb **, struct RastPort *);

 FUNCTION
For every AnimOb in the list
    - update its location and velocities
    - call the AnimOb's special routine if one is supplied
    - for each component of the AnimOb
        - if this sequence times out, switch to the new one
        - call this component's special routine if one is supplied
        - set the sequence's VSprite's y,x coordinates based
          on whatever these routines cause

 INPUTS
ankey = address of the variable that points to the head AnimOb
rp    = pointer to the RastPort structure

 RESULT

 BUGS
```

```
 SEE ALSO
AddAnimOb() graphics/gels.h graphics/rastport.h
```

1.13 graphics.library/AreaCircle

```
  NAME
AreaCircle -- add a circle to areainfo list for areafill.


  SYNOPSIS
error = (int) AreaCircle( rp,  cx,  cy, radius)
D0        A1   D0   D1  D2

ULONG AreaCircle(struct RastPort *, WORD, WORD, UWORD);

  FUNCTION
Add circle to the vector buffer. It will be drawn to the rastport when
AreaEnd is executed.

  INPUTS
rp   - pointer to a RastPort structure

cx, cy   - the coordinates of the center of the desired circle.

radius   - is the radius of the circle to draw around the centerpoint.

  RESULTS
0 if no error
-1 if no space left in vector list

  NOTES
This function is actually a macro which calls
    AreaEllipse(rp,cx,cy,radius,radius).

  SEE ALSO
AreaMove() AreaDraw() AreaCircle() InitArea() AreaEnd()
graphics/rastport.h graphics/gfxmacros.h
```

1.14 graphics.library/AreaDraw

```
 NAME
AreaDraw -- Add a point to a list of end points for areafill.


 SYNOPSIS
error = AreaDraw( rp,  x,     y)
  d0            A1 D0:16 D1:16

ULONG AreaDraw( struct RastPort *, SHORT, SHORT);

 FUNCTION
```

```
Add point to the vector buffer.


 INPUTS
rp  - points to a RastPort structure.
x,y - are coordinates of a point in the raster.

 RESULT
error - zero for success, else -1 if no there was no space
    left in the vector list.

 BUGS

 SEE ALSO
AreaMove() InitArea() AreaEnd() graphics/rastport.h
```

1.15 graphics.library/AreaEllipse

```
  NAME
AreaEllipse -- add a ellipse to areainfo list for areafill.


  SYNOPSIS
error = AreaEllipse( rp, cx,   cy,   a,    b    )
d0         a1  d0:16 d1:16 d2:16 d3:16

LONG AreaEllipse( struct RastPort *, SHORT, SHORT, SHORT, SHORT)

  FUNCTION
Add an ellipse to the vector buffer. It will be draw when AreaEnd() is
called.

  INPUTS
rp - pointer to a RastPort structure
cx - x coordinate of the centerpoint relative to the rastport.
cy - y coordinate of the centerpoint relative to the rastport.
a  - the horizontal radius of the ellipse (note: a must be > 0)
b  - the vertical radius of the ellipse (note: b must be > 0)

  RESULT
error - zero for success, or -1 if there is no space left in the
  vector list

  SEE ALSO
AreaMove() AreaDraw() AreaCircle() InitArea() AreaEnd()
graphics/rastport.h
```

1.16 graphics.library/AreaEnd

```
 NAME
AreaEnd -- Process table of vectors and ellipses and produce areafill.
```

```
 SYNOPSIS
error = AreaEnd(rp)
  d0            A1

LONG AreaEnd( struct RastPort * );

 FUNCTION
Trigger the filling operation.
Process the vector buffer and generate required
fill into the raster planes. After the fill is complete, reinitialize
for the next AreaMove or AreaEllipse. Use the raster set up by
InitTmpRas when generating an areafill mask.

 RESULT
error - zero for success, or -1 if an error occurred anywhere.

 INPUTS
rp - pointer to a RastPort structure which specifies where the filled
     regions will be rendered to.

 BUGS

 SEE ALSO
InitArea() AreaMove() AreaDraw() AreaEllipse()  InitTmpRas()
graphics/rastport.h
```

1.17 graphics.library/AreaMove

```
 NAME
AreaMove -- Define a new starting point for a new
            shape in the vector list.


 SYNOPSIS
error =  AreaMove( rp,   x,     y)
 d0               a1  d0:16  d1:16

LONG AreaMove( struct RastPort *, SHORT, SHORT );

 FUNCTION
Close  the last polygon and start another polygon
at  (x,y). Add the necessary  points  to  vector
buffer. Closing a polygon may result in the generation
of another AreaDraw() to close previous polygon.
Remember to have an initialized AreaInfo structure attached
to the RastPort.

 INPUTS
rp  - points to a RastPort structure
x,y - positions in the raster

 RETURNS
error - zero for success, or -1 if there is no space left in the
```

```
vector list

 BUGS

 SEE ALSO
InitArea() AreaDraw() AreaEllipse() AreaEnd() graphics/rastport.h
```

1.18 graphics.library/AskFont

```
 NAME
AskFont -- get the text attributes of the current font

 SYNOPSIS
AskFont(rp, textAttr)
        A1  A0

void AskFont(struct RastPort *, struct TextAttr *);

 FUNCTION
This function fills the text attributes structure with the
attributes of the current font in the RastPort.

 INPUTS
rp       - the RastPort from which the text attributes are
           extracted
textAttr - the TextAttr structure to be filled.  Note that
           there is no support for a TTextAttr.

 RESULT
The textAttr structure is filled with the RastPort's text
attributes.

 BUGS

 SEE ALSO
graphics/text.h
```

1.19 graphics.library/AskSoftStyle

```
 NAME
AskSoftStyle -- Get the soft style bits of the current font.

 SYNOPSIS
enable = AskSoftStyle(rp)
D0                    A1

ULONG AskSoftStyle(struct RastPort *);

 FUNCTION
This function returns those style bits of the current font
that are not intrinsic in the font itself, but
```

```
algorithmically generated.  These are the bits that are
valid to set in the enable mask for SetSoftStyle().

 INPUTS
rp - the RastPort from which the font and style are extracted.

 RESULTS
enable - those bits in the style algorithmically generated.
         Style bits that are not defined are also set.

 BUGS

 SEE ALSO
SetSoftStyle()  graphics/text.h
```

1.20 graphics.library/AttachPalExtra

```
 NAME
     AttachPalExtra -- Allocate and attach a palette sharing structure to a

                  colormap. (V39)


 SYNOPSIS
     status=AttachPalExtra( cm, vp)
                       a0  a1

LONG AttachPalExtra( Struct ColorMap *, struct ViewPort *);

 FUNCTION
Allocates and attaches a PalExtra structure to a ColorMap.
This is necessary for color palette sharing to work. The
PalExtra structure will be freed by FreeColorMap().
The set of available colors will be determined by the mode
and depth of the viewport.

 INPUTS
cm  =  A pointer to a color map created by GetColorMap().

vp   = A pointer to the viewport structure associated with
       the ColorMap.

 RESULTS
status - 0 if sucessful, else an error number. The only currently
         defined error number is out of memory (1).

 BUGS

 NOTES
This function is for use with custom ViewPorts and custom ColorMaps,
as Intuition attaches a PalExtra to all of its Screens.
If there is already a PalExtra associated with the ColorMap, then
this function will do nothing.

 SEE ALSO
```

```
GetColorMap() FreeColorMap() ObtainPen() ObtainBestPenA()
```

1.21 graphics.library/AttemptLockLayerRom

```
                          *
 NAME
AttemptLockLayerRom -- Attempt to Lock Layer structure
         by ROM(gfx lib) code

 SYNOPSIS
gotit = AttemptLockLayerRom( layer )
 d0           a5

BOOL AttempLockLayerRom( struct Layer * );

 FUNCTION
Query the current state of the lock on this Layer. If it is
already locked then return FALSE, could not lock. If the
Layer was not locked then lock it and return TRUE.
This call does not destroy any registers.
This call nests so that callers in this chain will not lock
themselves out.

 INPUTS
layer - pointer to Layer structure

 RESULT
gotit - TRUE or FALSE depending on whether the Layer was
  successfully locked by the caller.

 SEE ALSO
LockLayerRom() UnlockLayerRom()
```

1.22 graphics.library/BestModeIDA

```
 NAME
BestModeIDA -- calculate the best ModeID with given parameters (V39)
BestModeID  -- varargs stub for BestModeIDA()

 SYNOPSIS
ID = BestModeIDA(TagItems)
d0               a0

ULONG BestModeIDA(struct TagItem *);

ID = BestModeID(Tag1, ...)

ULONG BestModeID(ULONG, ...);

 FUNCTION
To determine the best ModeID to fit the parameters set in the TagList.
```

```
 INPUTS
TagItems - A pointer to an array of TagItems.

 TAGS
BIDTAG_DIPFMustHave (ULONG) - Mask of DIPF_ flags
       (from DisplayInfo->PropertyFlags) that the returned ModeID
       must have.
       Default - NULL

BIDTAG_DIPFMustNotHave (ULONG) - Mask of DIPF_ flags that the
       returned ModeID must not have.
       Default - SPECIAL_FLAGS

BIDTAG_ViewPort (struct ViewPort *) - ViewPort for which a best-fit
       ModeID is sought.
       Default - NULL

BIDTAG_NominalWidth (UWORD),
BIDTAG_NominalHeight (UWORD) - together make the aspect ratio.
       These values override the vp->DWidth and vp->DHeight values
       in the given ViewPort.
       Default - SourceID NominalDimensionInfo if BIDTAG_SourceID is
       passed, or vp->DWidth and vp->DHeight if BIDTAG_ViewPort is
       passed, or 640 x 200.

BIDTAG_DesiredWidth (UWORD),
BIDTAG_DesiredHeight (UWORD) - Used to distinguish between two
       mode IDs with identical aspect ratios.
       Default - same values as NominalWidth and NominalHeight.

BIDTAG_Depth (UBYTE) - minimum the returned ModeID must support.
       Default - vp->RasInfo->BitMap->Depth if BIDTAG_ViewPort is
       passed, else 1.

BIDTAG_MonitorID (ULONG) - returned ModeID must use this monitor.
       Default - will not restrict the search to any particular monitor

BIDTAG_SourceID (ULONG) - Use this ModeID instead of a ViewPort.
       If specified, the DIPFMustHave mask is made up of the
       ((DisplayInfo->PropertyFlags of this ID & SPECIAL_FLAGS) |
        DIPFMustHave flags).
       Default - VPModeID(vp) if BIDTAG_ViewPort was passed, else the
       DIPFMustHave and DIPFMustNotHave masks are left unchanged.

BIDTAG_RedBits (UBYTE),
BIDTAG_BlueBits (UBYTE),
BIDTAG_Greenits (UBYTE) - Minimum bits per gun the resultant
       ModeID must support.
       Default - 4 bits per gun.

 RESULTS
ID      - ID of the best mode to use, or INVALID_ID if a match could
          not be found.

 NOTES
This function takes into account the Compatability of the Monitor
being matched to, and the source ViewPort or ModeID.
```

```
Incompatibilitys will cause a result of INVALID_ID.

BIDTAG_NominalWidth, BIDTAG_NominalHeight,
BIDTAG_DesiredWidth, BIDTAG_DesiredHeight, must all be non-0.

The comparisons are made against the DimensionInfo->Nominal values.
ie, this will not return a best fit against overscan dimensions.

 EXAMPLE
IFF Display Program with a HAM image, to be displayed in the same
monitor type as the Workbench ViewPort.

ID = BestModeID(BIDTAG_NominalWidth, IFFImage->Width,
                BIDTAG_NominalHeight, IFFImage->Height,
                BIDTAG_Depth, IFFImage->Depth,
                BIDTAG_DIPFMustHave, DIPF_IS_HAM,
                BIDTAG_MonitorID, (GetVPModeID(WbVP) & MONITOR_ID_MASK),
                TAG_END);

To make an interlace version of a ViewPort:

ID = BestModeID(BIDTAG_ViewPort, ThisViewPort,
                BIDTAG_MustHave, DIFP_IS_LACE,
                TAG_END);

 SEE ALSO
<graphics/modeid.h> <graphics/displayinfo.h>
```

1.23 graphics.library/BitMapScale

```
 NAME
BitMapScale -- Perform raster scaling on a bit map. (V36)

 SYNOPSIS
BitMapScale(bitScaleArgs)
            A0

void BitMapScale(struct BitScaleArgs *);

 FUNCTION
Scale a source bit map to a non-overlapping destination
bit map.

 INPUTS
bitScaleArgs - structure of parameters describing scale:
    bsa_SrcX, bsa_SrcY - origin of the source bits.
    bsa_SrcWidth, bsa_SrcHeight - number of bits to scale from in x
    and y.
    bsa_DestX, bsa_DestY - origin of the destination.
    bsa_DestWidth, bsa_DestHeight - resulting number of bits in x
    and y.  NOTE: these values are set by this function.
    bsa_XSrcFactor:bsa_XDestFactor - equivalent to the ratio
        srcWidth:destWidth, but not necessarily the same
        numbers.  Each must be in the range 1..16383.
    bsa_YSrcFactor:bsa_YDestFactor - equivalent to the ratio
```

```
        srcHeight:destHeight, but not necessarily the same
        numbers.  Each must be in the range 1..16383.
    bsa_SrcBitMap - source of the bits to scale.
    bsa_DestBitMap - destination for the bits to scale.  This had
        better be big enough!
    bsa_Flags - future scaling options.  Set it to zero!
    bsa_XDDA, bsa_YDDA - for future use.  Need not be set by user.
    bsa_Reserved1, bsa_Reserved2 - for future use.  Need not be set.

 RESULT
The destWidth, destHeight fields are set by this function as
described above.

 NOTES
o   This function may use the blitter.
o   Overlapping source and destination bit maps are not
    supported.
o   No check is made to ensure destBitMap is big enough: use
    ScalerDiv to calculate a destination dimension.

 BUGS
o   This function does not use the HighRes Agnus 'Big Blit'
    facility. You should not use XSrcFactor == XDestFactor,
    where SrcWidth or DestWidth > 1024.

o   Also, the blitter is used when expanding in the Y direction.
    You should not expand in the Y direction if
    ((DestX & 0xf) + DestWidth) >= 1024 pixels. (Up to 1008 pixels
    is always safe).

 SEE ALSO
ScalerDiv()  graphics/scale.h
```

1.24 graphics.library/BltBitMap

```
 NAME
BltBitMap -- Move a rectangular region of bits in a BitMap.

 SYNOPSIS
planecnt = BltBitMap(SrcBitMap, SrcX, SrcY, DstBitMap,
D0                   A0         D0:16 D1:16 A1
    DstX, DstY, SizeX, SizeY, Minterm, Mask [, TempA])
    D2:16 D3:16 D4:16  D5:16  D6:8     D7:8   [A2]

ULONG BltBitMap(struct BitMap *, WORD, WORD, struct BitMap *,
    WORD, WORD, WORD, WORD, UBYTE, UBYTE, UWORD *);

 FUNCTION
Perform non-destructive blits to move a rectangle from one
area in a BitMap to another area, which can be on a different
BitMap.
This blit is assumed to be friendly: no error conditions (e.g.
a rectangle outside the BitMap bounds) are tested or reported.

 INPUTS
```

```
SrcBitMap, DstBitMap - the BitMap(s) containing the
      rectangles
    - the planes copied from the source to the destination are
      only those whose plane numbers are identical and less
      than the minimum Depth of either BitMap and whose Mask
      bit for that plane is non-zero.
    - as a special case, if a plane pointer in the SrcBitMap
      is zero, it acts as a pointer to a plane of all zeros, and
      if the plane pointer is 0xffffffff, it acts as a pointer
      to a plane of all ones.  (Note: new for V36)
    - SrcBitMap and DstBitMap can be identical if they point
      to actual planes.
SrcX, SrcY - the x and y coordinates of the upper left corner
    of the source rectangle.  Valid range is positive
    signed integer such that the raster word's offset
    0..(32767-Size)
DstX, DstY - the x and y coordinates of the upper left
    corner of the destination for the rectangle.  Valid
    range is as for Src.
SizeX, SizeY - the size of the rectangle to be moved.  Valid
    range is (X: 1..976; Y: 1..1023 such that final raster
    word's offset is 0..32767)
Minterm - the logic function to apply to the rectangle when
    A is non-zero (i.e. within the rectangle).  B is the
    source rectangle and C, D is the destination for the
    rectangle.
    - $0C0 is a vanilla copy
    - $030 inverts the source before the copy
    - $050 ignores the source and inverts the destination
    - see the hardware reference manual for other combinations
Mask - the write mask to apply to this operation.  Bits set
    indicate the corresponding planes (if not greater than
    the minimum plane count) are to participate in the
    operation.  Typically this is set to 0xff.
TempA - If the copy overlaps exactly to the left or right
    (i.e. the scan line addresses overlap), and TempA is
    non-zero, it points to enough chip accessible memory
    to hold a line of A source for the blit (ie CHIP RAM).
    BltBitMap will allocate (and free) the needed TempA if
    none is provided and one is needed.  Blit overlap is
    determined from the relation of the first non-masked
    planes in the source and destination bit maps.

 RESULTS
planecnt - the number of planes actually involved in the blit.

 NOTES
o   This function may use the blitter.

 SEE ALSO
ClipBlit()  graphics/gfx.h  hardware/blit.h
```

1.25 graphics.library/BltBitMapRastPort

```
 NAME
BltBitMapRastPort -- Blit from source bitmap to destination rastport.

 SYNOPSIS
error = BltBitMapRastPort
        (srcbm, srcx, srcy, destrp, destX, destY, sizeX, sizeY, minterm)
 D0      A0     D0    D1    A1      D2     D3     D4     D5     D6

BOOL BltBitMapRastPort
     (struct BitMap *, WORD, WORD, struct RastPort *, WORD, WORD,
      WORD, WORD, UBYTE);

 FUNCTION
Blits from source bitmap to position specified in destination rastport
using minterm.

 INPUTS
srcbm   - a pointer to the source bitmap
srcx    - x offset into source bitmap
srcy    - y offset into source bitmap
destrp  - a pointer to the destination rastport
destX   - x offset into dest rastport
destY   - y offset into dest rastport
sizeX   - width of blit in pixels
sizeY   - height of blit in rows
minterm - minterm to use for this blit

 RESULT
TRUE

 BUGS

 SEE ALSO
BltMaskBitMapRastPort() graphics/gfx.h graphics/rastport.h
```

1.26 graphics.library/BltClear

```
 NAME

BltClear - Clear a block of memory words to zero.

 SYNOPSIS
BltClear( memBlock, bytecount, flags )
      a1         d0       d1

void BltClear( void *, ULONG, ULONG );

 FUNCTION
For memory that is local and blitter accessible, the most
efficient way to clear a range of memory locations is
     to use the system's most efficient data mover, the blitter.
     This command accepts the starting location and count and clears
     that block to zeros.
```

```
 INPUTS
memBloc - pointer to local memory to be cleared
    memBlock is assumed to be even.
flags - set bit 0 to force function to wait until
    the blit is done.
    set bit 1 to use row/bytesperrow.

bytecount - if (flags & 2) == 0 then
      even number of bytes to clear.
    else
      low 16 bits is taken as number of bytes
      per row and upper 16 bits taken as
      number of rows.

This function is somewhat hardware dependent. In the rows/bytesperrow
mode (with the pre-ECS blitter) rows must be <- 1024. In bytecount mode
multiple runs of the blitter may be used to clear all the memory.

Set bit 2 to use the upper 16 bits of the Flags as the data to fill
memory with instead of 0 (V36).

 RESULT
The block of memory is initialized.

 BUGS

 SEE ALSO
```

1.27 graphics.library/BltMaskBitMapRastPort

```
 NAME
BltMaskBitMapRastPort -- blit from source bitmap to destination rastport
with masking of source image.

 SYNOPSIS
BltMaskBitMapRastPort
    (srcbm, srcx, srcy, destrp, destX, destY, sizeX, sizeY,
     A0     D0    D1    A1      D2     D3     D4     D5
     minterm, bltmask)
     D6       A2

void BltMaskBitMapRastPort
     (struct BitMap *, WORD, WORD, struct RastPort *, WORD, WORD,
      WORD, WORD, UBYTE, APTR);

 FUNCTION
Blits from source bitmap to position specified in destination rastport
using bltmask to determine where source overlays destination, and
minterm to determine whether to copy the source image "as is" or
to "invert" the sense of the source image when copying. In either
case, blit only occurs where the mask is non-zero.

 INPUTS
srcbm   - a pointer to the source bitmap
srcx    - x offset into source bitmap
```

```
srcy    - y offset into source bitmap
destrp  - a pointer to the destination rastport
destX   - x offset into dest rastport
destY   - y offset into dest rastport
sizeX   - width of blit in pixels
sizeY   - height of blit in rows
minterm - either (ABC|ABNC|ANBC) if copy source and blit thru mask
          or     (ANBC)          if invert source and blit thru mask
bltmask - pointer to the single bit-plane mask, which must be the
          same size and dimensions as the planes of the
          source bitmap.

 RESULT

 BUGS

 SEE ALSO
BltBitMapRastPort() graphics/gfx.h graphics/rastport.h
```

1.28 graphics.library/BltPattern

```
 NAME
BltPattern --  Using standard drawing rules for areafill,
         blit through a mask.

 SYNOPSIS
     BltPattern(rp, mask, xl, yl, maxx, maxy, bytecnt)
                a1,  a0   d0  d1   d2   d3     d4

void BltPattern
   (struct RastPort *, void *, SHORT, SHORT, SHORT, SHORT, SHORT);

 FUNCTION
     Blit using drawmode,areafill pattern, and mask
     at position rectangle (xl,yl) (maxx,maxy).

 INPUTS
     rp    -  points to the destination RastPort for the blit.
     mask  -  points to 2 dimensional mask if needed
              if mask == NULL then use a rectangle.
     xl,yl -  coordinates of upper left of rectangular region in RastPort
     maxx,maxy - points to lower right of rectangular region in RastPort
     bytecnt - BytesPerRow for mask

 RESULT

 SEE ALSO
AreaEnd()
```

1.29 graphics.library/BltTemplate

```
 NAME
BltTemplate -- Cookie cut a shape in a rectangle to the RastPort.

 SYNOPSIS
BltTemplate(SrcTemplate, SrcX, SrcMod, rp,
            A0           D0:16  D1:16  A1
    DstX,  DstY, SizeX, SizeY)
    D2:16  D3:16 D4:16  D5:16

void BltTemplate(UWORD *, WORD, WORD, struct RastPort *,
     WORD, WORD, WORD, WORD);

 FUNCTION
This function draws the image in the template into the
RastPort in the current color and drawing mode at the
specified position.  The template is assumed not to overlap
the destination.
If the template falls outside the RastPort boundary, it is
truncated to that boundary.

Note: the SrcTemplate pointer should point to the "nearest" word
   (rounded down) of the template mask. Fine alignment of the mask
   is achieved by setting the SrcX bit offseet within the range
   of 0 to 15 decimal.

 INPUTS
SrcTemplate  - pointer to the first (nearest) word of the template mask.
SrcX         - x bit offset into the template mask (range 0..15).
SrcMod       - number of bytes per row in template mask.
rp           - pointer to destination RastPort.
DstX, DstY   - x and y coordinates of the upper left
               corner of the destination for the blit.
SizeX, SizeY - size of the rectangle to be used as the
               template.

 NOTES
o   This function may use the blitter.

 SEE ALSO
BltBitMap()  graphics/rastport.h
```

1.30 graphics.library/CalcIVG

```
 NAME
CalcIVG -- Calculate the number of blank lines above a ViewPort (V39)

 SYNOPSIS
count = CalcIVG(View, ViewPort)
 d0.w           a0    a1

UWORD CalcIVG(struct View *, struct ViewPort *);

 FUNCTION
To calculate the maximum number of blank lines above a viewport needed to
```

```
load all the copper instructions, after accounting for the viewport
bandwidth and size.

 INPUTS
View       - pointer to the View
ViewPort   - pointer to the ViewPort you are interested in.

 RESULT
count      - the number of ViewPort resolution scan lines needed to
             execute all the copper instructions for ViewPort,
             or 0 if any error.

 NOTES
The number of copper instructions comes from the vp->vp_DspIns list.
Although there may be other copper instructions in the final list (from
UCopIns, SprIns and ClrIns) they are currently ignored for this
function. This also means that if the ViewPort has never been made
(for example, the ViewPort of an intuition screen was opened behind)
then vp->vp_DspIns is NULL.

Although CalcIVG() returns the true number of lines needed by the
copper, intuition still maintains an inter-screen gap of 3 non-laced
lines (6 interlaced). Therefore, for intuition screens use:
MAX(CalcIVG(v, vp), (islaced ? 6 : 3))


 SEE ALSO
GfxNew()  VideoControl()  graphics/view.h
```

1.31 graphics.library/CBump

```
 NAME
CBump -  increment user copper list pointer (bump to next position in list).

 SYNOPSIS
CBump( c )
       a1

void CBump( struct UCopList * );

 FUNCTION
Increment pointer to space for next instruction in user copper list.

 INPUTS
c - pointer to UCopList structure

 RESULTS
User copper list pointer is incremented to next position.
Pointer is repositioned to next user copperlist instruction block
if the current block is full.

    Note: CBump is usually invoked for the programmer as part of the
          macro definitions CWAIT or CMOVE.

 BUGS
```

```
 SEE ALSO
CINIT() CWAIT() CMOVE() CEND() graphics/copper.h
```

1.32 graphics.library/CEND

```
 NAME
CEND -- Terminate user copper list.

 SYNOPSIS
CEND( c )

struct UCopList *c;

 FUNCTION
Add instruction to terminate user copper list.

 INPUTS
c - pointer to UCopList structure

 RESULTS
This is actually a macro that calls the macro CWAIT(c,10000,255)
10000 is a magical number that the graphics.library uses.
I hope display technology doesn't catch up too fast!

 BUGS

 SEE ALSO
CINIT() CWAIT() CMOVE() graphics/copper.h
```

1.33 graphics.library/ChangeExtSpriteA

```
 NAME
     ChangeExtSpriteA -- Change the sprite image pointer. (V39)

 SYNOPSIS
     ChangeExtSpriteA( vp, oldsprite, newsprite, tags)
                       a0  a1     a2     a3

success=ChangeExtSpriteA(struct ViewPort *, struct ExtSprite *,
    struct ExtSprite *, struct TagList *);

success=ChangeExtSprite(vp,old_sp,new_sp,tag,....);

 FUNCTION
Attempt to change which sprite is displayed for a given
sprite engine.

 INPUTS
     vp - pointer to ViewPort structure that this sprite is
    relative to,  or 0 if relative only top of View
oldsprite - pointer the old ExtSprite structure
```

```
newsprite - pointer to the new ExtSprite structure.

 RESULTS
success - 0 if there was an error.
 BUGS

 SEE ALSO
FreeSprite() ChangeSprite() MoveSprite() AllocSpriteDataA()
graphics/sprite.h
```

1.34 graphics.library/ChangeSprite

```
 NAME
     ChangeSprite -- Change the sprite image pointer.

 SYNOPSIS
     ChangeSprite( vp, s, newdata)
                   a0  a1   a2

void ChangeSprite(struct ViewPort *, struct SimpleSprite *, void * )

 FUNCTION
The sprite image is changed to use the data starting at newdata

 INPUTS
     vp - pointer to ViewPort structure that this sprite is
    relative to,  or 0 if relative only top of View
s - pointer to SimpleSprite structure
newdata - pointer to data structure of the following form.
  struct spriteimage
  {
      UWORD    posctl[2]; /* used by simple sprite machine*/
      UWORD    data[height][2];   /* actual sprite image */
      UWORD    reserved[2]; /* initialized to */
                            /*  0x0,0x0 */
  };
The programmer must initialize reserved[2].  Spriteimage must be
in CHIP memory. The height subfield of the SimpleSprite structure
must be set to reflect the height of the new spriteimage BEFORE
calling ChangeSprite(). The programmer may allocate two sprites to
handle a single attached sprite.  After GetSprite(), ChangeSprite(),
the programmer can set the SPRITE_ATTACHED bit in posctl[1] of the
odd numbered sprite.
If you need more than 8 sprites, look up VSprites in the
graphics documentation.

 RESULTS

 BUGS

 SEE ALSO
FreeSprite() ChangeSprite() MoveSprite() AddVSprite() graphics/sprite.h
```

1.35 graphics.library/ChangeVPBitMap

```
 NAME
     ChangeVPBitMap -- change display memory address for multi-buffered
      animation (V39)

 SYNOPSIS
     ChangeVPBitMap(vp,bm,db)
                    a0 a1 a2

void ChangeVPBitMap(struct ViewPort *, struct BitMap *, struct DBufInfo *);

 FUNCTION
Changes the area of display memory which will be displayed in a
viewport. This can be used to implement double (or triple)
buffering, a method of achieving smooth animation.

 INPUTS
vp  =  a pointer to a viewport
     bm  = a pointer to a BitMap structure. This BitMap structure must be
      of the same layout as the one attached to the viewport (same
      depth, alignment, and BytesPerRow).
db  =  A pointer to a DBufInfo.

 BUGS

 NOTES
This will set the vp->RasInfo->BitMap field to the bm pointer which is
passed.

When using the synchronization features, you MUST carefully insure that
all messages have been replied to before calling FreeDBufInfo or
calling ChangeVPBitMap with the same DBufInfo.


 SEE ALSO
AllocDBufInfo() AllocBitMap()
```

1.36 graphics.library/CINIT

```
 NAME
CINIT -- Initialize user copperlist to accept intermediate
   user copper instructions.

 SYNOPSIS
cl = CINIT( ucl , n )

cl = UCopperListInit( ucl , n )
          a0    d0

struct CopList *UCopperListInit( struct UCopList *, UWORD );

 FUNCTION
Allocates and/or initialize copperlist structures/buffers
```

```
internal to a UCopList structure.

This is a macro that calls UCopListInit. You must pass a
(non-initialized) UCopList to CINIT (CINIT will NOT allocate
a new UCopList if ucl==0 ). If (ucl != 0) it will initialize the
intermediate data buffers internal to a UCopList.

The maximum number of intermediate copper list instructions
that these internal CopList data buffers contain is specified
as the parameter n.

 INPUTS
ucl - pointer to UCopList structure
n - number of instructions buffer must be able to hold

 RESULTS
cl- a pointer to a buffer which will accept n intermediate copper
    instructions.

NOTE: this is NOT a UCopList pointer, rather a pointer to the
      UCopList's->FirstCopList sub-structure.

 BUGS
CINIT will not actually allocate a new UCopList if ucl==0.
Instead you must allocate a block MEMF_PUBLIC|MEMF_CLEAR, the
sizeof(struct UCopList) and pass it to this function.

The system's FreeVPortCopLists function will take care of
deallocating it if they are called.

Prior to release V36 the  CINIT macro had { } braces surrounding
the definition, preventing the proper return of the result value.
These braces have been removed for the V36 include definitions.

 SEE ALSO
CINIT() CMOVE() CEND() graphics/copper.h
```

1.37 graphics.library/ClearEOL

```
 NAME
ClearEOL -- Clear from current position to end of line.

 SYNOPSIS
ClearEOL(rp)
         A1

void ClearEOL(struct RastPort *);

 FUNCTION
Clear a rectangular swath from the current position to the
right edge of the rastPort.  The height of the swath is taken
from that of the current text font, and the vertical
positioning of the swath is adjusted by the text baseline,
such that text output at this position would lie wholly on
this newly cleared area.
```

```
Clearing consists of setting the color of the swath to zero,
or, if the DrawMode is 2, to the BgPen.

 INPUTS
rp - pointer to RastPort structure

 RESULT

 NOTES
o   This function may use the blitter.

 SEE ALSO
Text()  ClearScreen()  SetRast()
graphics/text.h  graphics/rastport.h
```

1.38 graphics.library/ClearRectRegion

```
 NAME
ClearRectRegion -- Perform 2d CLEAR operation of rectangle
    with region, leaving result in region.

 SYNOPSIS
status = ClearRectRegion(region,rectangle)
 d0       a0    a1

BOOL ClearRectRegion(struct Region *, struct Rectangle * );

 FUNCTION
Clip away any portion of the region that exists inside
of the rectangle. Leave the result in region.

 INPUTS
region - pointer to Region structure
rectangle - pointer to Rectangle structure

 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS

 SEE ALSO
AndRectRegion() graphics/regions.h
```

1.39 graphics.library/ClearRegion

```
 NAME
     ClearRegion -- Remove all rectangles from region.

 SYNOPSIS
     ClearRegion(region)
                 a0
```

```
viod ClearRegion( struct Region * );

 FUNCTION
     Clip away all rectangles in the region leaving nothing.

 INPUTS
     region - pointer to Region structure

 BUGS

 SEE ALSO
NewRegion() graphics/regions.h
```

1.40 graphics.library/ClearScreen

```
 NAME
ClearScreen -- Clear from current position to end of RastPort.

 SYNOPSIS
ClearScreen(rp)
            A1

void ClearScreen(struct RastPort *);

 FUNCTION
Clear a rectangular swath from the current position to the
right edge of the rastPort with ClearEOL, then clear the rest
of the screen from just beneath the swath to the bottom of
the rastPort.
Clearing consists of setting the color of the swath to zero,
or, if the DrawMode is 2, to the BgPen.

 INPUTS
rp - pointer to RastPort structure

 NOTES
o   This function may use the blitter.

 SEE ALSO
ClearEOL()  Text()  SetRast()
graphics/text.h  graphics/rastport.h
```

1.41 graphics.library/ClipBlit

```
 NAME
ClipBlit  --  Calls BltBitMap() after accounting for windows

 SYNOPSIS
ClipBlit(Src, SrcX, SrcY, Dest, DestX, DestY, XSize, YSize, Minterm)
         A0   D0    D1    A1    D2     D3     D4     D5     D6
```

```
void ClipBlit
     (struct RastPort *, WORD, WORD, struct RastPort *, WORD, WORD,
      WORD, WORD, UBYTE);

 FUNCTION
Performs the same function as BltBitMap(), except that it
takes into account the Layers and ClipRects of the layer library,
all of which are (and should be) transparent to you.  So, whereas
BltBitMap() requires pointers to BitMaps, ClipBlit requires pointers to
the RastPorts that contain the Bitmaps, Layers, etcetera.

If you are going to blit blocks of data around via the RastPort of your
Intuition Window, you must call this routine (rather than BltBitMap()).

Either the Src RastPort, the Dest RastPort, both, or neither, can have
Layers. This routine takes care of all cases.

See BltBitMap() for a thorough explanation.

 INPUTS
Src          = pointer to the RastPort of the source for your blit
SrcX, SrcY   = the topleft offset into Src for your data
Dest         = pointer to the RastPort to receive the blitted data
DestX, DestY = the topleft offset into the destination RastPort
XSize        = the width of the blit (must be ta least 1)
YSize        = the height of the blit (must be at least 1)
Minterm      = the boolean blitter function, where SRCB is associated
               with the Src RastPort and SRCC goes to the Dest RastPort

 RESULT

 BUGS

 SEE ALSO
BltBitMap()
```

1.42 graphics.library/CloseFont

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

 SYNOPSIS
CloseFont(font)
          A1

void CloseFont(struct TextFont *);

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

 INPUTS
font -  a font pointer as returned by OpenFont() or OpenDiskFont()
```

```
 RESULT

 BUGS

 SEE ALSO
OpenFont()  diskfont.library/OpenDiskFont  graphics/text.h
```

1.43 graphics.library/CloseMonitor

```
 NAME
CloseMonitor -- close a MonitorSpec (V36)

 SYNOPSIS
error = CloseMonitor( monitor_spec )
d0                    a0

LONG CloseMonitor( struct MonitorSpec * );

 FUNCTION
Relinquish access to a MonitorSpec.

 INPUTS
monitor_spec - a pointer to a MonitorSpec opened via OpenMonitor(),
or NULL.

 RESULTS
error - FALSE if MonitorSpec closed uneventfully.
        TRUE if MonitorSpec could not be closed.

 BUGS

 SEE ALSO
OpenMonitor()
```

1.44 graphics.library/CMOVE

```
 NAME
CMOVE -- append copper move instruction to user copper list.

 SYNOPSIS
CMOVE( c , a , v )

CMove( c , a , v )
      a1  d0  d1
CBump( c )
      a1

void CMove( struct UCopList *, void *, WORD );

 FUNCTION
Add instruction to move value v to hardware register a.
```

```
 INPUTS
c - pointer to UCopList structure
a - hardware register
v - 16 bit value to be written

 RESULTS
This is actually a macro that calls CMove(c,&a,v)
and then calls CBump(c) to bump the local pointer
to the next instruction. Watch out for macro side affects.

 BUGS

 SEE ALSO
CINIT() CWAIT() CEND() graphics/copper.h
```

1.45 graphics.library/CoerceMode

```
 NAME
CoerceMode -- calculate ViewPort mode coercion (V39)

 SYNOPSIS
ID = CoerceMode(RealViewPort, MonitorID, Flags);
d0              a0            d0         d1

ULONG CoerceMode(struct ViewPort *, ULONG, ULONG);

 FUNCTION
To determine the best mode in the MonitorID to coerce RealViewPort to,
given the restrictions set in Flags.

 INPUTS
RealViewPort - ViewPort to coerce
MonitorID    - Montor number to coerce to (ie a mode masked with
               MONITOR_ID_MASK).
Flags        - PRESERVE_COLORS - keep the number of bitplanes
               in the ViewPort.
               AVOID_FLICKER - do not coerce to an interlace mode

 RESULTS
ID      - ID of the best mode to coerce to, or INVALID_ID if could not
          coerce (see NOTES).

 NOTES
This function takes into account the compatibility of the Monitor
being coerced to, and the ViewPort that is being coerced.
Incompatibilities will cause a result of INVALID_ID.

 EXAMPLE
newmode = CoerceMode(vp, VGA_MONITOR_ID, PRESERVE_COLORS);

 SEE ALSO
<graphics/coerce.h> <graphics/displayinfo.h>
```

1.46 graphics.library/CopySBitMap

```
 NAME
     CopySBitMap -- Syncronize Layer window with contents of
          Super BitMap

 SYNOPSIS
     CopySBitMap( layer )
                   a0

void CopySBitMap(struct Layer *);

 FUNCTION
This is the inverse of SyncSBitMap.
     Copy all bits from SuperBitMap to Layer bounds.
This is used for those functions that do not
want to deal with the ClipRect structures but do want
to be able to work with a SuperBitMap Layer.

 INPUTS
layer - pointer to a SuperBitMap Layer
    The Layer must already be locked by the caller.

 BUGS

 SEE ALSO
LockLayerRom() SyncSBitMap()
```

1.47 graphics.library/CWAIT

```
 NAME
CWAIT -- Append copper wait instruction to user copper list.

 SYNOPSIS
CWAIT( c , v , h )

CWait( c , v , h )
       a1  d0  d1
CBump( c )
      a1

void CWait( struct UCopList *, WORD, WORD)

 FUNCTION
Add instruction to wait for vertical beam position v and
horizontal position h to this intermediate copper list.

 INPUTS
c - pointer to UCopList structure
v - vertical beam position (relative to top of viewport)
h - horizontal beam position

 RESULTS
this is actually a macro that calls CWait(c,v,h)
```

```
and then calls CBump(c) to bump the local pointer
to the next instruction.

 BUGS
User waiting for horizontal values of greater than 222 decimal
is illegal.

 SEE ALSO
CINIT() CMOVE() CEND() graphics/copper.h
```

1.48 graphics.library/DisownBlitter

```
 NAME
     DisownBlitter -- return blitter to free state.


 SYNOPSIS
     DisownBlitter()

void DisownBlitter( void );

 FUNCTION
Free blitter up for use by other blitter users.

 INPUTS

 RETURNS

 SEE ALSO
     OwnBlitter()
```

1.49 graphics.library/DisposeRegion

```
 NAME
     DisposeRegion -- Return all space for this region to free
     memory pool.

 SYNOPSIS
     DisposeRegion(region)
                   a0

void DisposeRegion( struct Region * );

 FUNCTION
     Free all RegionRectangles for this Region then
free the Region itself.

 INPUTS
     region - pointer to Region structure

 BUGS
```

```
 SEE ALSO
NewRegion() graphics/regions.h
```

1.50 graphics.library/DoCollision

```
 NAME
DoCollision -- Test every gel in gel list for collisions.

 SYNOPSIS
DoCollision(rp)
            A1

void DoCollision(struct RastPort *);

 FUNCTION
Tests each gel in gel list for boundary and gel-to-gel collisions.
On detecting one of these collisions, the appropriate collision-
handling routine is called. See the documentation for a thorough
description of which collision routine is called. This routine
expects to find the gel list correctly sorted in Y,X order.
The system routine SortGList performs this function for the user.

 INPUTS
rp = pointer to a RastPort

 RESULT

 BUGS

 SEE ALSO
InitGels()  SortGList()  graphics/gels.h  graphics/gels.h
```

1.51 graphics.library/Draw

```
 NAME
     Draw -- Draw a line between the current pen position
                    and the new x,y position.

 SYNOPSIS
     Draw( rp,   x,     y)
           a1  d0:16  d1:16

void Draw( struct RastPort *, SHORT, SHORT);

 FUNCTION
     Draw a line from the current pen position to (x,y).

 INPUTS

rp - pointer to the destination RastPort
x,y - coordinates of where in the RastPort to end the line.
```

```
 BUGS

 SEE ALSO
Move() graphics/rastport.h
```

1.52 graphics.library/DrawEllipse

```
  NAME
DrawEllipse -- Draw an ellipse centered at cx,cy with vertical
   and horizontal radii of a,b respectively.

  SYNOPSIS
DrawEllipse( rp, cx, cy, a, b )
       a1  d0  d1  d2 d3

void DrawEllipse( struct RastPort *, SHORT, SHORT, SHORT, SHORT);

  FUNCTION
     Creates an elliptical outline within the rectangular region
specified by the parameters, using the current foreground pen color.

  INPUTS
rp - pointer to the RastPort into which the ellipse will be drawn.
cx - x coordinate of the centerpoint relative to the rastport.
cy - y coordinate of the centerpoint relative to the rastport.
a - the horizontal radius of the ellipse (note: a must be > 0)
b - the vertical radius of the ellipse (note: b must be > 0)

  BUGS

  NOTES
this routine does not clip the ellipse to a non-layered rastport.

  SEE ALSO
DrawCircle(), graphics/rastport.h
```

1.53 graphics.library/DrawGList

```
 NAME
DrawGList -- Process the gel list, queueing VSprites, drawing Bobs.

 SYNOPSIS
DrawGList(rp, vp)
          A1  A0

void DrawGList(struct RastPort *, struct ViewPort *);

 FUNCTION
Performs one pass of the current gel list.
   - If nextLine and lastColor are defined, these are
     initialized for each gel.
```

```
        - If it's a VSprite, build it into the copper list.
        - If it's a Bob, draw it into the current raster.
        - Copy the save values into the "old" variables,
     double-buffering if required.

 INPUTS
rp = pointer to the RastPort where Bobs will be drawn
vp = pointer to the ViewPort for which VSprites will be created

 RESULT

 BUGS
MUSTDRAW isn't implemented yet.

 SEE ALSO
InitGels()  graphics/gels.h graphics/rastport.h  graphics/view.h
```

1.54 graphics.library/EraseRect

```
 NAME

     EraseRect -- Fill a defined rectangular area using the current
        BackFill hook. (V36)

 SYNOPSIS
EraseRect( rp, xmin, ymin, xmax, ymax)
                a1  d0:16 d1:16 d2:16 d3:16

void EraseRect(struct RastPort *, SHORT, SHORT, SHORT, SHORT);

 FUNCTION
Fill the rectangular region specified by the parameters with the
BackFill hook. If non-layered, the rectangular region specified by
the parameters is cleared. If layered the Layer->BackFill Hook is used.

 INPUTS
rp  - pointer to a RastPort structure
xmin  - x coordinate of the upper left corner of the region to fill.
ymin  - y coordinate of the upper left corner of the region to fill.
xmax  - x coordinate of the lower right corner of the region to fill.
ymax  - y coordinate of the lower right corner of the region to fill.

 BUGS

 NOTES
The following relation MUST be true:
(xmax >= xmin) and (ymax >= ymin)

 SEE ALSO
graphics/rastport.h graphics/clip.h
```

1.55 graphics.library/ExtendFont

```
 NAME
ExtendFont -- ensure tf_Extension has been built for a font (V36)

 SYNOPSIS
success = ExtendFont(font, fontTags)
D0                   A0    A1

ULONG ExtendFont(struct TextFont *, struct TagItem *);

success = ExtendFontTags(font, Tag1, ...)  (V39)

ULONG ExtendFontTags(struct TextFont *, ULONG, ...);

 FUNCTION
To extend a TextFont structure.

 INPUTS
font - The font to extend.
fontTags - An optional taglist.  If NULL, then a default is used.
           Currently, the only tag defined is TA_DeviceDPI.

 RESULT
success - 1 if the TextFont was properly extended, else 0.

 NOTES
The varargs stub was missing from amiga.lib until V39.

 SEE ALSO
graphics/text.h
```

1.56 graphics.library/FindColor

```
 NAME
FindColor -- Find the closest matching color in a colormap. (V39)

 SYNOPSIS
color = FindColor(  cm,  R,   G,    B , maxpen)
                  a3   d1   d2    d3   d4

ULONG FindColor( struct ColorMap *, ULONG, ULONG, ULONG,LONG);

 INPUTS
cm = colormap
R = red level   (32 bit left justified fraction)
G = green level (32 bit left justified fraction)
B = blue level  (32 bit left justified fraction)
MaxPen = the maximum entry in the color table to search. A value of
        -1 will limt the search to only those pens which could be
        rendered in (for instance, it will not examine the sprite
        colors on a 4 color screen).


 RESULT
The system will attempt to find the color in the passed colormap
```

```
which most closely matches the RGB values passed. No new pens will
be allocated, and you should not ReleasePen() the returned pen.
This function is not sensitive to palette sharing issues. Its
intended use is for:

        (a) programs which pop up on public screens when those
            screens are not using palette sharing. You might
            use this function as a fallback when ObtainBestPenA()
            says that there are no sharable pens.

        (b) Internal color matching by an application which is
            either running on a non-public screen, or which
            wants to match colors to an internal color table
            which may not be associated with any displayed screen.

 BUGS

 NOTES
In order to use the MaxPen=-1 feature, you must have initialized
palette sharing via AttachPalExtra() (all intuition screens do this).
Otherwise, MaxPen=-1 will search all colors in the colormap.


 SEE ALSO
ObtainBestPenA() GetColorMap() ObtainPen() ReleasePen()
```

1.57 graphics.library/FindDisplayInfo

```
 NAME
FindDisplayInfo -- search for a record identified by a specific key (V36)

 SYNOPSIS
handle = FindDisplayInfo(ID)
D0                       D0

DisplayInfoHandle FindDisplayInfo(ULONG);

 FUNCTION
Given a 32-bit Mode Key, return a handle to a valid DisplayInfoRecord
found in the graphics database.  Using this handle, you can obtain
information about this Mode, including its default dimensions,
properties, and whether it is currently available for use.

 INPUTS
ID     - unsigned long identifier

 RESULT
handle - handle to a displayinfo Record with that key
         or NULL if no match.

 BUGS

 SEE ALSO
graphics/displayinfo.h
```

1.58 graphics.library/Flood

```
 NAME
Flood -- Flood rastport like areafill.

 SYNOPSIS
error = Flood( rp, mode, x, y)
      d0            a1   d2  d0  d1

BOOL Flood(struct RastPort *, ULONG, SHORT, SHORT);

 FUNCTION
Search the BitMap starting at (x,y).
Fill all adjacent pixels if they are:
    Mode 0: not the same color as AOLPen
    Mode 1: the same color as the pixel at (x,y)

When actually doing the fill use the modes that apply to
standard areafill routine such as drawmodes and patterns.

 INPUTS
rp - pointer to RastPort
(x,y) - coordinate in BitMap to start the flood fill at.
mode -  0 fill all adjacent pixels searching for border.
  1 fill all adjacent pixels that have same pen number
  as the one at (x,y).

 NOTES
In order to use Flood, the destination RastPort must
have a valid TmpRas raster whose size is as large as
that of the RastPort.

 SEE ALSO
AreaEnd() InitTmpRas() graphics/rastport.h
```

1.59 graphics.library/FontExtent

```
 NAME
FontExtent -- get the font attributes of the current font (V36)

 SYNOPSIS
FontExtent(font, fontExtent)
           A0    A1

void FontExtent(struct TextFont *, struct TextExtent *);

 FUNCTION
This function fills the text extent structure with a bounding
(i.e. maximum) extent for the characters in the specified font.

 INPUTS
font       - the TextFont from which the font metrics are extracted.
fontExtent - the TextExtent structure to be filled.
```

```
 RESULT
fontExtent is filled.

 NOTES
The TextFont, not the RastPort, is specified -- unlike
TextExtent(), effect of algorithmic enhancements is not
included, nor does te_Width include any effect of
rp_TxSpacing.  The returned te_Width will be negative only
when FPF_REVPATH is set in the tf_Flags of the font -- the
effect of left-moving characters is ignored for the width of
a normal font, and the effect of right-moving characters is
ignored if a REVPATH font.  These characters will, however,
be reflected in the bounding extent.

 SEE ALSO
TextExtent()  graphics/text.h
```

1.60 graphics.library/FreeBitMap

```
 NAME
FreeBitMap -- free a bitmap created by AllocBitMap (V39)

 SYNOPSIS
FreeBitMap(bm)
           a0

VOID FreeBitMap(struct BitMap *)

 FUNCTION
Frees bitmap and all associated bitplanes

 INPUTS
bm  =  A pointer to a BitMap structure. Passing a NULL-pointer
       (meaning "do nothing") is OK.

 BUGS

 NOTES
Be careful to insure that any rendering done to the bitmap has
completed (by calling WaitBlit()) before you call this function.

 SEE ALSO
AllocBitMap()
```

1.61 graphics.library/FreeColorMap

```
 NAME
     FreeColorMap -- Free the ColorMap structure and return memory
          to free memory pool.

 SYNOPSIS
     FreeColorMap( colormap )
```

```
                      a0

void FreeColorMap(struct ColorMap *);

 FUNCTION
Return the memory to the free memory pool that was allocated
with GetColorMap.

 INPUTS
colormap - pointer to ColorMap allocated with GetColorMap.

    Passing a NULL pointer (meaning "do nothing") is
    acceptable (V39).

 RESULT
The space is made available for others to use.

 BUGS

 SEE ALSO
     SetRGB4() GetColorMap() graphics/view.h
```

1.62 graphics.library/FreeCopList

```
 NAME
FreeCopList -- deallocate intermediate copper list

 SYNOPSIS
     FreeCopList(coplist)
        a0

void FreeCopList( struct CopList *);

 FUNCTION
Deallocate all memory associated with this copper list.

 INPUTS
     coplist  - pointer to structure CopList

 RESULTS
memory returned to memory manager

 BUGS

 SEE ALSO
graphics/copper.h
```

1.63 graphics.library/FreeCprList

```
 NAME
     FreeCprList -- deallocate hardware copper list
```

```
 SYNOPSIS
     FreeCprList(cprlist)
        a0

void FreeCprList(struct cprlist *);

 FUNCTION
     return cprlist to free memory pool

 INPUTS
     cprlist - pointer to cprlist structure

 RESULTS
memory returned and made available to other tasks

 BUGS

 SEE ALSO
graphics/copper.h
```

1.64 graphics.library/FreeDBufInfo

```
 NAME
     FreeDBufInfo -- free information for multi-buffered animation (V39)

 SYNOPSIS
     FreeDBufInfo(db)
              a1

void FreeDBufInfo(struct DBufInfo *)

 FUNCTION
Frees a structure obtained from AllocDBufInfo

 INPUTS
     db  =  A pointer to a DBufInfo.

 BUGS

 NOTES
FreeDBufInfo(NULL) is a no-op.

 SEE ALSO
AllocDBufInfo() ChangeVPBitMap()
```

1.65 graphics.library/FreeGBuffers

```
 NAME
FreeGBuffers -- Deallocate memory obtained by GetGBufers.

 SYNOPSIS
FreeGBuffers(anOb, rp, db)
```

```
              A0    A1  D0

  void FreeGBuffers(struct AnimOb *, struct RastPort *, BOOL);

   FUNCTION
  For each sequence of each component of the AnimOb,
  deallocate memory for:
      SaveBuffer
      BorderLine
      CollMask and ImageShadow (point to same buffer)
      if db is set (user had used double-buffering) deallocate:
          DBufPacket
          BufBuffer

   INPUTS
  anOb = pointer to the AnimOb structure
  rp   = pointer to the current RastPort
  db   = double-buffer indicator (set TRUE for double-buffering)

   RESULT

   BUGS

   SEE ALSO
  GetGBuffers()  graphics/gels.h  graphics/rastport.h
```

1.66 graphics.library/FreeRaster

```
   NAME
       FreeRaster -- Release an allocated area to the system free memory pool
.


   SYNOPSIS
       FreeRaster( p, width, height)
       a0   d0:16  d1:16

  void FreeRaster( PLANEPTR, USHORT, USHORT);

   FUNCTION
  Return the memory associated with this PLANEPTR of size
  width and height to the MEMF_CHIP memory pool.

   INPUTS
       p  =  a pointer to a memory space  returned  as  a
             result of a call to AllocRaster.

  width - the width in bits of the bitplane.
  height - number of rows in bitplane.

   BUGS

   NOTES
       Width and height should be the same values with which you
       called AllocRaster in the first place.
```

```
 SEE ALSO
AllocRaster() graphics/gfx.h
```

1.67 graphics.library/FreeSprite

```
 NAME
     FreeSprite -- Return sprite for use by others and virtual
          sprite machine.

 SYNOPSIS
     FreeSprite( pick )
                  d0

void FreeSprite( WORD );

 FUNCTION
Mark sprite as available for others to use.
     These sprite routines are provided to ease sharing of sprite
hardware and to handle simple cases of sprite usage and
movement.  It is assumed the programs that use these routines
do want to be good citizens in their hearts. ie: they will
not FreeSprite unless they actually own the sprite.
The Virtual Sprite machine may ignore the simple sprite machine.

 INPUTS
     pick - number in range of 0-7

 RESULTS
sprite made available for subsequent callers of GetSprite
as well as use by Virtual Sprite Machine.

 BUGS

 SEE ALSO
     GetSprite() ChangeSprite() MoveSprite() graphics/sprite.h
```

1.68 graphics.library/FreeSpriteData

```
 NAME
FreeSpriteData -- free sprite data allocated by AllocSpriteData() (V39)

 SYNOPSIS
FreeSpriteData(extsp)
    a2

void FreeSpriteData(struct ExtSprite *);


 FUNCTION

 INPUTS
```

```
extsp - The extended sprite structure to be freed. Passing NULL is a
NO-OP.

 SEE ALSO
FreeSpriteData() FreeSprite() ChangeSprite() MoveSprite() GetExtSprite()
AllocBitMap() graphics/sprite.h
```

1.69 graphics.library/FreeVPortCopLists

```
 NAME
     FreeVPortCopLists -- deallocate all intermediate copper lists and
     their headers from a viewport

 SYNOPSIS
     FreeVPortCopLists(vp)
                       a0

void FreeVPortCopLists(struct ViewPort *);

 FUNCTION
     Search display, color, sprite, and user copper
     lists and call FreeMem() to deallocate them from memory

 INPUTS
     vp - pointer to ViewPort structure

 RESULTS
     The memory allocated to the various copper lists will be returned
to the system's free memory pool, and the following fields in
the viewport structure will be set to NULL:

  DspIns, Sprins, ClrIns, UCopIns

 BUGS
     none known

 SEE ALSO
graphics/view.h
```

1.70 graphics.library/GetAPen

```
 NAME
     GetAPen -- Get the A Pen value for a RastPort (V39).


 SYNOPSIS
     pen = GetAPen  ( rp )
 d0    a0

ULONG GetAPen(struct RastPort *)

 FUNCTION
```

```
Return the current value of the A pen for the rastport. This function
should be used instead of peeking the structure directly, because future
graphics devices may store it differently, for instance, using more bits.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.

 BUGS

 NOTES

 SEE ALSO
SetAPen() graphics/gfx.h
```

1.71 graphics.library/GetBitMapAttr

```
 NAME
GetBitMapAttr -- Returns information about a bitmap (V39)


 SYNOPSIS
value=GetBitMapAttr(bitmap,attribute_number);
  d0                    a0       d1

ULONG GetBitMapAttr(struct BitMap *,ULONG);

 FUNCTION
Determines information about a bitmap. This function should be used
instead of reading the bitmap structure fields directly. This will
provide future compatibility.

 INPUTS
bm  =  A pointer to a BitMap structure.

attribute_number = A number telling graphics which attribute
                   of the bitmap should be returned:

                BMA_HEIGHT returns the height in pixels
                BMA_WIDTH  returns the width in pixels.
                BMA_DEPTH  returns the depth. This is the number of
                        bits which are required to store the information
                        for one pixel in the bitmap.
                BMA_FLAGS  returns a longword bitfield describing
                        various attributes which the bitmap may have.
                        Currently defined flags are BMF_DISPLAYABLE,
                        BMF_INTERLEAVED (see AllocBitMap()). The flag
                        BMF_STANDARD returns will be set if the
                        bitmap is represented as planar data in Amiga
                        Chip RAM.

 BUGS

 NOTES

Unknown attributes are reserved for future use, and return zero.
```

```
BMF_DISPLAYABLE will only be set if the source bitmap meets all of the
required alignment restrictions. A bitmap which does not meet these
restrictions may still be displayable at some loss of efficiency.

Size values returned by this function may not exactly match the values
which were passed to AllocBitMap(), due to alignment restrictions.

 SEE ALSO
AllocBitMap()
```

1.72 graphics.library/GetBPen

```
 NAME
     GetBPen -- Get the B Pen value for a RastPort (V39).


 SYNOPSIS
     pen = GetBPen  ( rp )
d0        a0

ULONG GetBPen(struct RastPort *)

 FUNCTION
Return the current value of the B pen for the rastport. This function
should be used instead of peeking the structure directly, because future
graphics devices may store it differently, using more bits.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.

 BUGS

 NOTES

 SEE ALSO
SetBPen() graphics/gfx.h
```

1.73 graphics.library/GetColorMap

```
 NAME
     GetColorMap -- allocate and initialize Colormap


 SYNOPSIS
     cm = GetColorMap( entries )
     d0        d0

struct ColorMap *GetColorMap( ULONG);

 FUNCTION
     Allocates, initializes and returns a pointer to a ColorMap
```

```
     data structure, later enabling calls to SetRGB4
     and LoadRGB4 to load colors for a view port. The ColorTable
pointer in the ColorMap structure points to a hardware
specific colormap data structure. You should not count on
it being anything you can understand. Use GetRGB4() to
query it or SetRGB4CM to set it directly.

 INPUTS
entries - number of entries for this colormap

  RESULT
The pointer value returned by this routine, if nonzero,
     may be stored into the ViewPort.ColorMap pointer.
     If a value of 0 is returned, the system was unable
     to allocate enough memory space for the required
     data structures.

 BUGS

 SEE ALSO
     SetRGB4() FreeColorMap()
```

1.74 graphics.library/GetDisplayInfoData

```
 NAME
GetDisplayInfoData -- query DisplayInfo Record parameters (V36)

 SYNOPSIS
result = GetDisplayInfoData(handle, buf, size, tagID, [ID])
D0                          A0      A1   D0    D1     [D2]

ULONG GetDisplayInfoData(DisplayInfoHandle, UBYTE *, ULONG, ULONG, ULONG);

 FUNCTION
GetDisplayInfoData() fills a buffer with data meaningful to the
DisplayInfoRecord pointed at by your valid handle. The data type
that you are interested in is indicated by a tagID for that chunk.
The types of tagged information that may be available include:

DTAG_DISP: (DisplayInfo)   - properties and availability information.
DTAG_DIMS: (DimensionInfo) - default dimensions and overscan info.
DTAG_MNTR: (MonitorInfo)   - type, position, scanrate, and compatibility
DTAG_NAME: (NameInfo)      - a user friendly way to refer to this mode.

 INPUTS
handle - displayinfo handle
buf    - pointer to destination buffer
size   - buffer size in bytes
tagID  - data chunk type
ID     - displayinfo identifier, optionally used if handle is NULL

 RESULT
result - if positive, number of bytes actually transferred
         if zero, no information for ID was available
```

```
 BUGS

 SEE ALSO
FindDisplayInfo(), NextDisplayInfo()
graphics/displayinfo.h
```

1.75 graphics.library/GetDrMd

```
 NAME
     GetDrMd -- Get the draw mode value for a RastPort (V39).


 SYNOPSIS
     mode = GetDrMd  ( rp )
d0                   a0

ULONG GetDrMd(struct RastPort *)

 FUNCTION
Return the current value of the draw mode for the rastport. This function
should be used instead of peeking the structure directly, because future
graphics devices may store it differently.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.

 BUGS

 NOTES

 SEE ALSO
SetDrMd() graphics/gfx.h
```

1.76 graphics.library/GetExtSpriteA

```
 NAME
GetExtSpriteA -- Attempt to get a sprite for the extended sprite
         manager. (V39)
GetExtSprite -- varargs stub for GetExtSpriteA. (V39)

 SYNOPSIS
Sprite_Number = GetExtSpriteA( sprite, tags )  (V39)
    d0                 a2      a1

LONG GetExtSpriteA( struct ExtSprite *, struct TagItem * );

spritenum=GetExtSprite(sprite,tags,...);

 FUNCTION
Attempt to allocate one of the eight sprites for private use
with the extended sprite manager.
```

```
 INPUTS
sprite - ptr to programmer's ExtSprite (from AllocSpriteData()).
tags - a standard tag list:

  GSTAG_SPRITE_NUM  specifies a specific sprite to get by number.

  GSTAG_ATTACHED specifies that you wish to get a sprite pair.
    the tag data field points to a ExtSprite structure
    for the second sprite. You must free both sprites.


 RESULTS
Sprite_number = a sprite number or -1 for an error.
  This call will fail if no sprites could be allocated, or
  if you try to allocate a sprite which would require
  a mode change when there are other sprites of incompatible
  modes in use.

 BUGS

GSTAG_ATTACHED does not work in version 39. When running under V39,
you should attach the second sprite with a separate GetExtSprite call.

 SEE ALSO
FreeSprite() ChangeSprite() MoveSprite() GetSprite() graphics/sprite.h
```

1.77 graphics.library/GetGBuffers

```
 NAME
GetGBuffers -- Attempt to allocate ALL buffers of an entire AnimOb.

 SYNOPSIS
status = GetGBuffers(anOb, rp, db)
D0                   A0    A1  D0

BOOL GetGBuffers(struct AnimOb *, struct RastPort *, BOOL);

 FUNCTION
For each sequence of each component of the AnimOb, allocate memory for:
    SaveBuffer
    BorderLine
    CollMask and ImageShadow (point to same buffer)
    if db is set TRUE (user wants double-buffering) allocate:
        DBufPacket
        BufBuffer

 INPUTS
anOb = pointer to the AnimOb structure
rp   = pointer to the current RastPort
db   = double-buffer indicator (set TRUE for double-buffering)

 RESULT
status = TRUE if the memory allocations were all successful, else FALSE

 BUGS
```

```
If any of the memory allocations fail it does not free the partial
allocations that did succeed.

 SEE ALSO
FreeGBuffers() graphics/gels.h
```

1.78 graphics.library/GetOPen

```
 NAME
     GetOPen -- Get the O Pen value for a RastPort (V39).


 SYNOPSIS
     pen = GetOPen  ( rp )
d0     a0

ULONG GetOPen(struct RastPort *)

 FUNCTION
Return the current value of the O pen for the rastport. This function
should be used instead of peeking the structure directly, because future
graphics devices may store it differently, for instance, using more bits.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.

 BUGS

 NOTES

 SEE ALSO
SetOutlinePen() graphics/gfx.h
```

1.79 graphics.library/GetRGB32

```
 NAME
GetRGB32 -- Set a series of color registers for this Viewport. (V39)

 SYNOPSIS
GetRGB32(  cm,  firstcolor, ncolors, table )
           a0   d0      d1    a1

     void GetRGB32( struct ColorMap *, ULONG, ULONG, ULONG *);

 INPUTS
cm = colormap
firstcolor = the first color register to get
ncolors = the number of color registers to set.
table=a pointer to a series of 32-bit RGB triplets.

 RESULT
The ULONG data pointed to by 'table' will be filled with the 32 bit
```

```
fractional RGB values from the colormap.
 BUGS

 NOTES
'Table' should point to at least ncolors*3 longwords.

 SEE ALSO
LoadRGB4() GetColorMap() LoadRGB32() SetRGB32CM() graphics/view.h
```

1.80 graphics.library/GetRGB4

```
 NAME
     GetRGB4 -- Inquire value of entry in ColorMap.

 SYNOPSIS
     value = GetRGB4( colormap, entry )
        d0              a0       d0

ULONG GetRGB4(struct ColorMap *, LONG);

 FUNCTION
Read and format a value from the ColorMap.

 INPUTS
colormap - pointer to ColorMap structure
entry - index into colormap

 RESULT
returns -1 if no valid entry
return UWORD RGB value 4 bits per gun right justified

 NOTE
Intuition's DisplayBeep() changes color 0. Reading Color 0 during a
DisplayBeep() will lead to incorrect results.

 BUGS

 SEE ALSO
     SetRGB4() LoadRGB4() GetColorMap() FreeColorMap() graphics/view.h
```

1.81 graphics.library/GetRPAttrA

```
 NAME
     GetRPAttrA -- examine rastport settings via a tag list
GetRPAttrs  -- varargs stub for GetRPAttrA

 SYNOPSIS
     GetRPAttrA(rp,tags)
                a0   a1

void GetRPAttrA(struct RastPort *, struct TagItem *);
```

```
GetRPAttrs(rp,attr1,&result1,...);

 FUNCTION
     Read the settings of a rastport into variables. The
ti_Tag field of the TagItem specifies which attribute
should be read, and the ti_Data field points at the
location where the result hsould be stored. All current
tags store the return data as LONGs (32 bits).

currently available tags are:

  RPTAG_Font    Font for Text()
  RPTAG_SoftStyle   style for text (see graphics/text.h)
  RPTAG_APen    Primary rendering pen
  RPTAG_BPen    Secondary rendering pen
  RPTAG_DrMd    Drawing mode (see graphics/rastport.h)
  RPTAG_OutLinePen  Area Outline pen
  RPTAG_WriteMask   Bit Mask for writing.
  RPTAG_MaxPen    Maximum pen to render (see SetMaxPen())
  RPTAG_DrawBounds  Determine the area that will be rendered
        into by rendering commands. Can be used
        to optimize window refresh. Pass a pointer
        to a rectangle in the tag data. On return,
        the rectangle's MinX will be greater than
        its MaxX if there are no active cliprects.

 INPUTS
rp - pointer to the RastPort to examine.
tags - a standard tag list specifying the attributes to be read,
  and where to store their values.

 RESULT

 BUGS

 SEE ALSO
GetAPen() GetBPen() GetDrMd() GetOutLinePen()
GetWriteMask() SetRPAttrA() graphics/rpattr.h
```

1.82 graphics.library/GetSprite

```
 NAME
GetSprite -- Attempt to get a sprite for the simple sprite manager.

 SYNOPSIS
Sprite_Number = GetSprite( sprite, pick )
    d0          a0      d0

WORD GetSprite( struct SimpleSprite *, WORD );

 FUNCTION
Attempt to allocate one of the eight sprites for private use
with the simple sprite manager. This must be done before using
further calls to the simple sprite machine. If the programmer
wants to use 15 color sprites, they must allocate both sprites
```

```
and set the 'SPRITE_ATTACHED' bit in the odd sprite's posctldata
array.

 INPUTS
sprite - ptr to programmers SimpleSprite structure.
pick - number in the range of 0-7 or
  -1 if programmer just wants the next one.

 RESULTS
If pick is 0-7 attempt to allocate the sprite. If the sprite
is already allocated then return -1.
If pick -1 allocate the next sprite starting search at 0.
If no sprites are available return -1 and fill -1 in num entry
of SimpleSprite structure.
If the sprite is available for allocation, mark it allocated
and fill in the 'num' entry of the SimpleSprite structure.
If successful return the sprite number.

 BUGS

 SEE ALSO
FreeSprite() ChangeSprite() MoveSprite() GetSprite() graphics/sprite.h
```

1.83 graphics.library/GetVPModeID

```
 NAME
GetVPModeID -- get the 32 bit DisplayID from a ViewPort. (V36)

 SYNOPSIS
modeID =  GetVPModeID( vp )
d0           a0

ULONG GetVPModeID( struct ViewPort *);

 FUNCTION
returns the normal display modeID, if one is currently  associated
with this ViewPort.

 INPUTS
vp -- pointer to a ViewPort structure.

 RESULT

modeID -- a 32 bit DisplayInfoRecord identifier associated with
    this ViewPort, or INVALID_ID.

 NOTES
Test the return value of this function against INVALID_ID, not NULL.
(INVALID_ID is defined in graphics/displayinfo.h).

 BUGS

 SEE ALSO
graphics/displayinfo.h, ModeNotAvailable()
```

1.84 graphics.library/GfxAssociate

```
 NAME
GfxAssociate -- associate a graphics extended node with a given pointer
                (V36)

 SYNOPSIS
     GfxAssociate(pointer, node);
                  A0       A1

void GfxAssociate(VOID *, struct ExtendedNode *);

 FUNCTION
Associate a special graphics extended data structure (each of which
begins with an ExtendedNode structure)  with another structure via
the other structure's pointer. Later, when you call GfxLookUp()
with the other structure's pointer you may retrieve a pointer
to this special graphics extended data structure, if it is
available.

 INPUTS
pointer = a pointer to a data structure.
node = an ExtendedNode structure to associate with the pointer

 RESULT
an association is created between the pointer and the node such
that given the pointer the node can be retrieved via GfxLookUp().

 BUGS

 SEE ALSO
graphics/gfxnodes.h GfxNew() GfxFree() GfxLookUp()
```

1.85 graphics.library/GfxFree

```
 NAME
     GfxFree -- free a graphics extended data structure (V36)

 SYNOPSIS
     GfxFree( node );
             a0

void GfxFree(struct ExtendedNode *);

 FUNCTION
Free a special graphics extended data structure (each of which
begins with an ExtendedNode structure).

 INPUTS
node = pointer to a graphics extended data structure obtained via
       GfxNew().

 RESULT
the node is deallocated from memory. graphics will disassociate
```

```
this special graphics extended node from any associated data
structures, if necessary, before freeing it (see GfxAssociate()).

 BUGS
an Alert() will be called if you attempt to free any structure
other than a graphics extended data structure obtained via GfxFree().

 SEE ALSO
graphics/gfxnodes.h GfxNew() GfxAssociate() GfxLookUp()
```

1.86 graphics.library/GfxLookUP

```
 NAME
    GfxLookUp -- find a graphics extended node associated with a
       given pointer (V36)

 SYNOPSIS
     result = GfxLookUp( pointer );
     d0       a0

struct ExtendedNode *GfxLookUp( void *);

 FUNCTION
Finds a special graphics extended data structure (if any) associated
with the pointer to a data structure (eg: ViewExtra associated with
a View structure).

 INPUTS
pointer = a pointer to a data structure which may have an
    ExtendedNode associated with it (typically a View ).

 RESULT
result = a pointer to the ExtendedNode that has previously been
   associated with the pointer.

 BUGS

 SEE ALSO
graphics/gfxnodes.h GfxNew() GfxFree() GfxAssociate()
```

1.87 graphics.library/GfxNew

```
 NAME
     GfxNew -- allocate a graphics extended data structure (V36)

 SYNOPSIS
result = GfxNew( node_type );
d0     d0

struct ExtendedNode *GfxNew( ULONG);

 FUNCTION
```

```
Allocate a special graphics extended data structure (each of which
begins with an ExtendedNode structure).  The type of structure to
be allocated is specified by the node_type identifier.

 INPUTS
node_type = which type of graphics extended data structure to allocate.
      (see gfxnodes.h for identifier definitions.)

 RESULT
result = a pointer to the allocated graphics node or NULL if the
   allocation failed.

 BUGS

 SEE ALSO
graphics/gfxnodes.h GfxFree() GfxAssociate() GfxLookUp()
```

1.88 graphics.library/InitArea

```
 NAME

InitArea -- Initialize vector collection matrix

 SYNOPSIS

  InitArea( areainfo, buffer, maxvectors )
      a0          a1      d0

void InitArea(struct AreaInfo *, void *, SHORT);

 FUNCTION
This function provides initialization for the vector collection matrix
such that it has a size of (max vectors ).  The size of the region
pointed to by buffer (short pointer) should be five (5) times as large
as maxvectors. This size is in bytes.  Areafills done by using AreaMove,
AreaDraw, and AreaEnd must have enough space allocated in this table to
store all the points of the largest fill. AreaEllipse takes up two
vectors for every call. If AreaMove/Draw/Ellipse detect too many
vectors going into the buffer they will return -1.

 INPUTS
areainfo - pointer to AreaInfo structure
buffer - pointer to chunk of memory to collect vertices
maxvectors - max number of vectors this buffer can hold

 RESULT
Pointers are set up to begin storage of vectors done by
AreaMove, AreaDraw, and AreaEllipse.

 BUGS

 SEE ALSO
AreaEnd() AreaMove() AreaDraw() AreaEllipse() graphics/rastport.h
```

1.89 graphics.library/InitBitMap

```
 NAME

  InitBitMap -- Initialize bit map structure with input values.

 SYNOPSIS
InitBitMap( bm, depth, width, height )
      a0   d0     d1      d2

void InitBitMap( struct BitMap *, BYTE, UWORD, UWORD );

 FUNCTION
Initialize various elements in the BitMap structure to
correctly reflect depth, width, and height.
Must be used before use of BitMap in other graphics calls.
The Planes[8] are not initialized and need to be set up
by the caller.  The Planes table was put at the end of the
structure so that it may be truncated to conserve space,
as well as extended. All routines that use BitMap should
only depend on existence of depth number of bitplanes.
The Flagsh and pad fields are reserved for future use and
should not be used by application programs.

 INPUTS
bm - pointer to a BitMap structure (gfx.h)
depth - number of bitplanes that this bitmap will have
width - number of bits (columns) wide for this BitMap
height- number of bits (rows) tall for this BitMap

 BUGS

 SEE ALSO
graphics/gfx.h
```

1.90 graphics.library/InitGels

```
 NAME
InitGels -- initialize a gel list; must be called before using gels.

 SYNOPSIS
InitGels(head, tail, GInfo)
         A0    A1    A2

void InitGels(struct VSprite *, struct VSprite *, struct GelsInfo *);

 FUNCTION
Assigns the VSprites as the head and tail of the gel list in GfxBase.
Links these two gels together as the keystones of the list.
If the collHandler vector points to some memory array, sets
the BORDERHIT vector to NULL.

 INPUTS
head  = pointer to the VSprite structure to be used as the gel list head
```

```
tail  = pointer to the VSprite structure to be used as the gel list tail
GInfo = pointer to the GelsInfo structure to be initialized

 RESULT

 BUGS

 SEE ALSO
graphics/gels.h  graphics/rastport.h
```

1.91 graphics.library/InitGMasks

```
 NAME
InitGMasks -- Initialize all of the masks of an AnimOb.

 SYNOPSIS
InitGMasks(anOb)
           A0

void InitGMasks(struct AnimOb *);

 FUNCTION
For every sequence of every component call InitMasks.

 INPUTS
anOb = pointer to the AnimOb

 BUGS

 SEE ALSO
InitMasks() graphics/gels.h
```

1.92 graphics.library/InitMasks

```
 NAME
InitMasks -- Initialize the BorderLine and CollMask masks of a VSprite.

 SYNOPSIS
InitMasks(vs)
          A0

void InitMasks(struct VSprite *);

 FUNCTION
Creates the appropriate BorderLine and CollMask masks of the VSprite.
Correctly detects if the VSprite is actually a Bob definition, handles
the image data accordingly.

 INPUTS
vs = pointer to the VSprite structure

 RESULT
```

```
 BUGS

 SEE ALSO
InitGels()  graphics/gels.h
```

1.93 graphics.library/InitRastPort

```
 NAME
InitRastPort -- Initialize raster port structure

 SYNOPSIS
  InitRastPort( rp )
        a1

void InitRastPort(struct RastPort *rp);

 FUNCTION
     Initialize a RastPort structure to standard values.

 INPUTS
rp  = pointer to a RastPort structure.

 RESULT
all entries in RastPort get zeroed out, with the following exceptions:

    Mask, FgPen, AOLPen, and LinePtrn are set to -1.
    The DrawMode is set to JAM2
    The font is set to the standard system font

 NOTES
The struct Rastport describes a control structure
     for a write-able raster. The RastPort structure
     describes how a complete single playfield display
     will be written into. A RastPort structure is
     referenced whenever any drawing or filling
     operations are to be performed on a section of
     memory.

     The section of memory which is being used in this
     way may or may not be presently a part of the
     current actual onscreen display memory. The name
     of the actual memory section which is linked to
     the RastPort is referred to here as a "raster" or
     as a bitmap.

     NOTE: Calling the routine InitRastPort only
     establishes various defaults. It does NOT
     establish where, in memory, the rasters are
     located. To do graphics with this RastPort the user
must set up the BitMap pointer in the RastPort.

 BUGS

 SEE ALSO
```

```
   graphics/rastport.h
```

1.94 graphics.library/InitTmpRas

```
 NAME
InitTmpRas -- Initialize area of local memory for usage by
    areafill, floodfill, text.

 SYNOPSIS
  InitTmpRas(tmpras, buffer, size)
                  a0       a1     d0

void InitTmpRas( struct TmpRas *, void *, ULONG );

 FUNCTION
The area of memory pointed to by buffer is set up to be used
by RastPort routines that may need to get some memory for
intermediate operations in preparation to putting the graphics
into the final BitMap.
Tmpras is used to control the usage of buffer.

 INPUTS
tmpras - pointer to a TmpRas structure to be linked into
  a RastPort
buffer - pointer to a contiguous piece of chip memory.
size - size in bytes of buffer

 RESULT
makes buffer available for users of RastPort

 BUGS
Would be nice if RastPorts could share one TmpRas.

 SEE ALSO
AreaEnd() Flood() Text() graphics/rastport.h
```

1.95 graphics.library/InitView

```
 NAME
 InitView - Initialize View structure.

 SYNOPSIS
InitView( view )
     a1

void InitView( struct View * );

 FUNCTION
Initialize View structure to default values.

 INPUTS
view - pointer to a View structure
```

```
 RESULT
View structure set to all 0's. (1.0,1.1.1.2)
Then values are put in DxOffset,DyOffset to properly position
default display about .5 inches from top and left on monitor.
InitView pays no attention to previous contents of view.

 BUGS

 SEE ALSO
MakeVPort graphics/view.h
```

1.96 graphics.library/InitVPort

```
 NAME
InitVPort - Initialize ViewPort structure.

 SYNOPSIS
InitVPort( vp )
     a0

void InitViewPort( struct ViewPort * );

 FUNCTION
Initialize ViewPort structure to default values.

 INPUTS
vp - pointer to a ViewPort structure

 RESULT
ViewPort structure set to all 0's. (1.0,1.1)
     New field added SpritePriorities, initialized to 0x24 (1.2)

 BUGS

 SEE ALSO
MakeVPort() graphics/view.h
```

1.97 graphics.library/LoadRGB32

```
 NAME
LoadRGB32 -- Set a series of color registers for this Viewport. (V39)

 SYNOPSIS
LoadRGB32(  vp,  table )
            a0   a1

void LoadRGB32( struct ViewPort *, ULONG *);

 INPUTS
vp = viewport
table = a pointer to a series of records which describe which colors to
```

```
        modify.
 RESULT
The selected color registers are changed to match your specs.
 BUGS

 NOTES

Passing a NULL "table" is ignored.
The format of the table passed to this function is a series of records,
each with the following format:

        1 Word with the number of colors to load
        1 Word with the first color to be loaded.
        3 longwords representing a left justified 32 bit rgb triplet.
        The list is terminated by a count value of 0.

   examples:
        ULONG table[]={1l<<16+0,0xffffffff,0,0,0} loads color register
                0 with 100% red.
        ULONG table[]={256l<<16+0,r1,g1,b1,r2,g2,b2,.....0} can be used
                to load an entire 256 color palette.

Lower order bits of the palette specification will be discarded,
depending on the color palette resolution of the target graphics
device. Use 0xffffffff for the full value, 0x7fffffff for 50%,
etc. You can find out the palette range for your screen by
querying the graphics data base.

LoadRGB32 is faster than SetRGB32, even for one color.

 SEE ALSO
LoadRGB4() GetColorMap() GetRGB32() SetRGB32CM() graphics/view.h
```

1.98 graphics.library/LoadRGB4

```
 NAME
LoadRGB4 -- Load RGB color values from table.

 SYNOPSIS
LoadRGB4( vp, colors , count )
               a0   a1     d0:16

void LoadRGB4( struct ViewPort *, UWORD *, WORD);

 FUNCTION
  load the count words of the colormap from table starting at
entry 0.

 INPUTS
vp - pointer to ViewPort, whose colors you wish to change
colors - pointer to table of RGB values set up as an array
         of USHORTS
    background--  0x0RGB
    color1    --  0x0RGB
    color2    --  0x0RGB
```

```
     etc.         UWORD per value.
  The colors are interpreted as 15 = maximum intensity.
                                0 = minimum intensity.
count = number of UWORDs in the table to load into the
  colormap starting at color 0(background) and proceeding
  to the next higher color number

 RESULTS
The ViewPort should have a pointer to a valid ColorMap to store
the colors in.
Updates the hardware copperlist to reflect the new colors.
Updates the intermediate copperlist with the new colors.

 BUGS

NOTE: Under V36 and up, it is not safe to call this function
from an interrupt, due to semaphore protection of graphics
copper lists.

 SEE ALSO
SetRGB4() GetRGB4() GetColorMap() graphics/view.h
```

1.99 graphics.library/LoadView

```
 NAME
     LoadView -- Use a (possibly freshly created) coprocessor instruction
                 list to create the current display.

 SYNOPSIS
     LoadView( View )
               A1

void LoadView( struct View * );

 FUNCTION
Install a new view to be displayed during the next display
refresh pass.
     Coprocessor instruction list has been created by
     InitVPort(), MakeVPort(), and MrgCop().

 INPUTS
     View - a pointer to the View structure which contains the
     pointer to the constructed coprocessor instructions list, or NULL.

 RESULT
If the View pointer is non-NULL, the new View is displayed,
according to your instructions.  The vertical blank routine
will pick this pointer up and direct the copper to start
displaying this View.

If the View pointer is NULL, no View is displayed.

 NOTE
Even though a LoadView(NULL) is performed, display DMA will still be
active.  Sprites will continue to be displayed after a LoadView(NULL)
```

```
unless an OFF_SPRITE is subsequently performed.

 BUGS

 SEE ALSO
     InitVPort() MakeVPort() MrgCop() intuition/RethinkDisplay()
graphics/view.h
```

1.100 graphics.library/LockLayerRom

```
 NAME
LockLayerRom -- Lock Layer structure by ROM(gfx lib) code.

 SYNOPSIS
LockLayerRom( layer )
         a5

void LockLayerRom( struct Layer * );

 FUNCTION
Return when the layer is locked and no other task may
alter the ClipRect structure in the Layer structure.
This call does not destroy any registers.
This call nests so that callers in this chain will not lock
themselves out.
Do not have the Layer locked during a call to intuition.
There is a potential deadlock problem here, if intuition
needs to get other locks as well.
Having the layer locked prevents other tasks from using the
layer library functions, most notably intuition itself. So
be brief.
layers.library's LockLayer is identical to LockLayerRom.

 INPUTS
layer - pointer to Layer structure

 RESULTS
The layer is locked and the task can render assuming the
ClipRects will not change out from underneath it until
an UnlockLayerRom is called.

 SEE ALSO
UnlockLayerRom() layers.library/LockLayer() graphics/clip.h
```

1.101 graphics.library/MakeVPort

```
 NAME
MakeVPort -- generate display copper list for a viewport.

 SYNOPSIS
error =  MakeVPort( view, viewport )
 d0                 a0     a1
```

```
ULONG MakeVPort( struct View *, struct ViewPort * );

 FUNCTION
Uses information in the View, ViewPort, ViewPort->RasInfo to
construct and intermediate copper list for this ViewPort.

 INPUTS
view - pointer to a View structure
viewport - pointer to a ViewPort structure
   The viewport must have valid pointer to a RasInfo.

 RESULTS
constructs intermediate copper list and puts pointers in
viewport.DspIns
If the ColorMap ptr in ViewPort is NULL then it uses colors
from the default color table.
If DUALPF in Modes then there must be a second RasInfo pointed
to by the first RasInfo

From V39, MakeVPort can return a ULONG error value (previous versions
returned void), to indicate that either not enough memory could be
allocated for MakeVPort's use, or that the ViewPort mode
and bitplane alignments are incorrect for the bitplane's depth.

You should check for these error values - they are defined in
<graphics/view.h>.


 BUGS
In V37 and earlier, narrow Viewports (whose righthand edge is
less than 3/4 of the way across the display) do not work properly.

 SEE ALSO
InitVPort() MrgCop() graphics/view.h intuition.library/MakeScreen()
intuition.library/RemakeDisplay() intuition.library/RethinkDisplay()
```

1.102 graphics.library/ModeNotAvailable

```
 NAME
ModeNotAvailable -- check to see if a DisplayID isn't available. (V36)

 SYNOPSIS
error =  ModeNotAvailable( modeID )
d0               d0

ULONG ModeNotAvailable( ULONG);

 FUNCTION
returns an error code, indicating why this modeID is not available,
or NULL if there is no reason known why this mode should not be there.

 INPUTS
modeID -- a 32 bit DisplayInfoRecord identifier.
```

```
 RESULT
error -- a general indication of why this modeID is not available,
   or NULL if there is no reason why it shouldn't be available.

 NOTE
ULONG return values from this function are a proper superset of the
DisplayInfo.NotAvailable field (defined in graphics/displayinfo.h).

 BUGS

 SEE ALSO
graphics/displayinfo.h, GetVPModeID()
```

1.103 graphics.library/Move

```
 NAME
Move -- Move graphics pen position.

 SYNOPSIS
Move( rp,   x,    y)
      a1  d0:16 d1:16

void Move( struct RastPort *, SHORT, SHORT );

 FUNCTION
Move graphics pen position to (x,y) relative to upper left (0,0)
of RastPort. This sets the starting point for subsequent Draw()
and Text() calls.

 INPUTS
rp - pointer to a RastPort structure
x,y - point in the RastPort

 RESULTS

 BUGS

 SEE ALSO
Draw() graphics/rastport.h
```

1.104 graphics.library/MoveSprite

```
 NAME
MoveSprite -- Move sprite to a point relative to top of viewport.

 SYNOPSIS
MoveSprite(vp, sprite, x, y)
           A0  A1      D0 D1

void MoveSprite(struct ViewPort *,struct SimpleSprite *, WORD, WORD);

 FUNCTION
```

```
Move sprite image to new place on display.

 INPUTS
vp - pointer to ViewPort structure
     if vp = 0, sprite is positioned relative to View.
sprite - pointer to SimpleSprite structure
(x,y)  - new position relative to top of viewport or view.

 RESULTS
Calculate the hardware information for the sprite and
place it in the posctldata array. During next video display
the sprite will appear in new position.

 BUGS
Sprites really appear one pixel to the left of the position you specify.
This bug affects the apparent display position of the sprite on the
screen, but does not affect the numeric position relative to the
viewport or view. This behaviour only applies to SimpleSprites,
not to ExtSprites.


 SEE ALSO
FreeSprite()  ChangeSprite()  GetSprite()  graphics/sprite.h
```

1.105 graphics.library/MrgCop

```
 NAME
     MrgCop -- Merge together coprocessor instructions.

 SYNOPSIS
     error = MrgCop( View )
     d0              A1

ULONG MrgCop( struct View * );

 FUNCTION
     Merge together the display, color, sprite and user coprocessor
     instructions into a single coprocessor instruction stream.  This
     essentially creates a per-display-frame program for the coprocessor.
     This function MrgCop is used, for example, by the graphics animation
     routines which effectively add information into an essentially
     static background display.  This changes some of the user
     or sprite instructions, but not those which have formed the
     basic display in the first place.  When all forms of coprocessor
     instructions are merged together, you will have a complete per-
     frame instruction list for the coprocessor.

     Restrictions:  Each of the coprocessor instruction lists MUST be
     internally sorted in min to max Y-X order.  The merge routines
     depend on this! Each list must be terminated using CEND(copperlist).

 INPUTS
     View - a pointer to the view structure whose coprocessor
            instructions are to be merged.
```

```
 RESULT
     The view structure will now contain a complete, sorted/merged
     list of instructions for the coprocessor, ready to be used by
     the display processor.  The display processor is told to use
     this new instruction stream through the instruction LoadView().

From V39, MrgCop() can return a ULONG error value (previous versions
returned void), to indicate that either there was insufficient memory
to build the system copper lists, or that MrgCop() had no work to do
if, for example, there were no ViewPorts in the list.

You should check for these error values - they are defined in
<graphics/view.h>.

 BUGS

 SEE ALSO
     InitVPort() MakeVPort() LoadView() graphics/view.h
intuition.library/RethinkDisplay()
```

1.106 graphics.library/NewRegion

```
 NAME
     NewRegion -- Get an empty region.

 SYNOPSIS
     region = NewRegion()
d0

struct Region *NewRegion();

 FUNCTION
Create a Region structure, initialize it to empty, and return
a pointer it.

 RESULTS
region - pointer to initialized region. If it could not allocate
  required memory region = NULL.

 INPUTS
none

 BUGS

 SEE ALSO
graphics/regions.h
```

1.107 graphics.library/NextDisplayInfo

```
 NAME
NextDisplayInfo -- iterate current displayinfo identifiers (V36)
```

```
 SYNOPSIS
next_ID = NextDisplayInfo(last_ID)
D0                        D0

ULONG NextDisplayInfo(ULONG);

 FUNCTION
The basic iteration function with which to find all records in the
graphics database.  Using each ID in succession, you can then call
FindDisplayInfo() to obtain the handle associated with each ID.
Each ID is a 32-bit Key which uniquely identifies one record.
The INVALID_ID is special, and indicates the end-of-list.

 INPUTS
last_ID - previous displayinfo identifier
          or INVALID_ID if beginning iteration.

 RESULT
next_ID - subsequent displayinfo identifier
          or INVALID_ID if no more records.

 BUGS

 SEE ALSO
FindDisplayInfo(), GetDisplayInfoData()
graphics/displayinfo.h
```

1.108 graphics.library/ObtainBestPenA

```
 NAME
ObtainBestPenA --- Search for the closest color match, or allocate a
                   new one. (V39)
ObtainBestPen  --- varargs stub for ObtainBestPenA

 SYNOPSIS
color | -1 =ObtainBestPenA(  cm,  R,   G,    B,    taglist)
                             a0   d1   d2    d3       a1

LONG ObtainBestPenA( struct ColorMap *, ULONG, ULONG,
    ULONG, struct TagItem *);

color = ObtainBestPen(cm,r,g,b,tags....);

 INPUTS
cm = colormap
R = red level   (32 bit left justified fraction)
G = green level (32 bit left justified fraction)
B = blue level  (32 bit left justified fraction)
taglist = a pointer to a standard tag list specifying the color
          matching settings desired:

        OBP_Precision - specifies the desired precision for the
                match. Should be PRECISION_GUI, PRECISION_ICON, or
                PRECISION_IMAGE or PRECISION_EXACT.
                Defaults to PRECISION_IMAGE.
```

```
        OBP_FailIfBad - specifies that you want ObtainBestPen to return
                a failure value if there is not a color within the
                given tolerance, instead of returning the closest color.
                With OBP_FailIfBad==FALSE, ObtainBestPen will only fail
                if the ViewPort contains no sharable colors.
                Defaults to FALSE.


 FUNCTION
This function can be used by applications to figure out
what pen to use to represent a given color.

The system will attempt to find the color in your viewport closest
to the specified color. If there is no color within your tolerance,
then a new one will be allocated, if available. If none is available,
then the closest one found will be returned.

 RESULT
The correct pen value, or -1 if no sharable palette entries are available.


 BUGS

 NOTES
If this call succceeds, then you must call ReleasePen() when you are
done with the color.

The error metric used for ObtainBestPen() is based on the magnitude
squared between the two RGB values, scaled by the percentage of free
entries.

 SEE ALSO
GetColorMap() ObtainPen() ReleasePen()
```

1.109 graphics.library/ObtainPen

```
 NAME
ObtainPen -- Obtain a free palette entry for use by your program. (V39)


 SYNOPSIS
n = ObtainPen( cm, n, r, g, b, flags)
d0         a0 d0  d1 d2 d3  d4

LONG ObtainPen(struct ColorMap *,ULONG,ULONG,ULONG,ULONG,ULONG);

 FUNCTION
Attempt to allocate an entry in the colormap for use by the application.
If successful, you should ReleasePen() this entry after you have finished
with it.

Applications needing exclusive use of a color register (say for color
cycling) will typically call this function with n=-1. Applications needing
only the shared use of a color will typically use ObtainBestPenA() instead.
```

```
Other uses of this function are rare.

 INPUTS
cm  =  A pointer to a color map created by GetColorMap().
n   =  The index of the desired entry, or -1 if any one is acceptable
rgb =  The RGB values (32 bit left justified fractions) to set the new
       palette entry to.
flags= PEN_EXCLUSIVE - tells the system that you want exclusive
       (non-shared) use of this pen value. Default is shared access.

       PEN_NO_SETCOLOR - tells the system to not change the rgb values
       for the selected pen. Really only makes sense for exclusive pens.


 RESULTS

n   =  The allocated pen. -1 will be returned if there is no pen available
       for you.

 BUGS

 NOTES
When you allocate a palette entry in non-exclusive mode, you
should not change it (via SetRGB32), because other programs on the
same screen may be using it. With PEN_EXCLUSIVE mode, you can
change the returned entry at will.

To avoid visual artifacts, you should not free up a palette
entry until you are sure that your application is not displaying
any pixels in that color at the time you free it. Otherwise, another
task could allocate and set that color index, thus changing the colors
of your pixels.

Generally, for shared access, you should use ObtainBestPenA()
instead, since it will not allocate a new color if there is one
"close enough" to the one you want already.
If there is no Palextra attached to the colormap, then this
routine will always fail.

 SEE ALSO
GetColorMap() ReleasePen() AttachPalExtra() ObtainBestPenA()
```

1.110 graphics.library/OpenFont

```
 NAME
OpenFont -- Get a pointer to a system font.

 SYNOPSIS
font = OpenFont(textAttr)
D0              A0

struct TextFont *OpenFont(struct TextAttr *);

 FUNCTION
This function searches the system font space for the graphics
```

```
text font that best matches the attributes specified.  The
pointer to the font returned can be used in subsequent
SetFont and CloseFont calls.  It is important to match this
call with a corresponding CloseFont call for effective
management of ram fonts.

 INPUTS
textAttr - a TextAttr or TTextAttr structure that describes the
           text font attributes desired.

 RESULT
font is zero if the desired font cannot be found.  If the named
font is found, but the size and style specified are not
available, a font with the nearest attributes is returned.

 BUGS
Prior to V39 this function would return a TextFont pointer
for any font which matched exactly in Y size, regardless of
differences in DPI, or DotSize.

As part of fixing this bug it is REQUIRED that you use pass the
same TextAttr (or TTextAttr) to this function that was used when
OpenDiskFont() was called.

OpenFont(), and OpenDiskFont() use WeighTAMatch() to measure
how well two fonts match.  WeightTAMatch() was a public function
in graphics.library V36-V37; it is now a system PRIVATE function
as of V39.

 SEE ALSO
CloseFont()  SetFont()
diskfont.library/OpenDiskFont  graphics/text.h
intuition/intuition.h
```

1.111 graphics.library/OpenMonitor

```
 NAME
     OpenMonitor -- open a named MonitorSpec (V36)

 SYNOPSIS
     mspc = OpenMonitor( monitor_name , display_id)
     d0                  a1      d0

     struct MonitorSpec *OpenMonitor( char *, ULONG );

 FUNCTION
     Locate and open a named MonitorSpec.

 INPUTS
     monitor_name - a pointer to a null terminated string.
     display_id - an optional 32 bit monitor/mode identifier

 RESULTS
     mspc - a pointer to an open MonitorSpec structure.
            NULL if MonitorSpec could not  be opened.
```

```
 NOTE
if monitor_name is non-NULL, the monitor will be opened by name.
if monitor_name is NULL the monitor will be opened by optional ID.
if both monitor_name and display_id are NULL returns default monitor.

 BUGS

 SEE ALSO
     CloseMonitor() graphics/monitor.h
```

1.112 graphics.library/OrRectRegion

```
 NAME
     OrRectRegion -- Perform 2d OR operation of rectangle
                     with region, leaving result in region.

 SYNOPSIS
     status = OrRectRegion(region,rectangle)
       d0                  a0      a1

BOOL OrRectRegion( struct Region *, struct Rectangle * );

 FUNCTION
     If any portion of rectangle is not in the region then add
     that portion to the region.

 INPUTS
     region - pointer to Region structure
     rectangle - pointer to Rectangle structure

 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS

 SEE ALSO
AndRectRegion() OrRegionRegion() graphics/regions.h
```

1.113 graphics.library/OrRegionRegion

```
 NAME
     OrRegionRegion -- Perform 2d OR operation of one region
                     with second region, leaving result in second region

 SYNOPSIS
     status = OrRegionRegion(region1,region2)
       d0                    a0      a1

BOOL OrRegionRegion( struct Region *, struct Region * );
```

```
 FUNCTION
     If any portion of region1  is not in the region then add
     that portion to the region2

 INPUTS
     region1 - pointer to Region structure
     region2 - pointer to Region structure

 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS

 SEE ALSO
OrRectRegion() graphics/regions.h
```

1.114 graphics.library/OwnBlitter

```
 NAME
     OwnBlitter -- get the blitter for private usage

 SYNOPSIS
     OwnBlitter()

void OwnBlitter( void );

 FUNCTION
If blitter is available return immediately with the blitter
locked for your exclusive use. If the blitter is not available
put task to sleep. It will be awakened as soon as the blitter
is available. When the task first owns the blitter the blitter
may still be finishing up a blit for the previous owner. You
must do a WaitBlit before actually using the blitter registers.

Calls to OwnBlitter() do not nest. If a task that owns the
blitter calls OwnBlitter() again, a lockup will result.
(Same situation if the task calls a system function
that tries to own the blitter).

 INPUTS
NONE

 RETURNS
NONE

 SEE ALSO
DisownBlitter() WaitBlit()
```

1.115 graphics.library/PolyDraw

```
 NAME
PolyDraw -- Draw lines from table of (x,y) values.

 SYNOPSIS
PolyDraw( rp, count , array )
    a1   d0      a0

void PolyDraw( struct RastPort *, WORD, WORD * );

 FUNCTION
starting with the first pair in the array, draw connected lines to
it and every successive pair.

 INPUTS
rp - pointer to RastPort structure
count -  number of (x,y) pairs in the array
array - pointer to first (x,y) pair

 BUGS

 SEE ALSO
Draw() Move() graphics/rastport.h
```

1.116 graphics.library/QBlit

```
 NAME

QBlit -- Queue up a request for blitter usage

 SYNOPSIS
QBlit( bp )
       a1

void QBlit( struct bltnode * );

 FUNCTION
Link a request for the use of the blitter to the end of the
     current blitter queue.  The pointer bp points to a blit structure
     containing, among other things, the link information, and the
     address of your routine which is to be called when the blitter
     queue finally gets around to this specific request.  When your
     routine is called, you are in control of the blitter ... it is
     not busy with anyone else's requests.  This means that you can
     directly specify the register contents and start the blitter.
     See the description of the blit structure and the uses of QBlit
     in the section titled Graphics Support in the OS Kernel Manual.
     Your code must be written to run either in supervisor or user
     mode on the 68000.

 INPUTS
bp - pointer to a blit structure

 RESULT
Your routine is called when the blitter is ready for you.
```

```
In general requests for blitter usage through this channel are
put in front of those who use the blitter via OwnBlitter and
DisownBlitter. However for small blits there is more overhead
using the queuer than Own/Disown Blitter.

 NOTES
Code which uses QBlit(), or QBSBlit() should make use of
the pointer to a cleanup routine in the bltnode structure.
The cleanup routine may be called on the context of an
interrupt, therefore the routine may set a flag, and signal
a task, but it may not call FreeMem() directly.  Use of
the cleanup routine is the only safe way to signal that
your bltnode has completed.

 BUGS
QBlit(), and QBSBlit() have been rewritten for V39 due to
various long standing bugs in earlier versions of this code.

 SEE ALSO
QBSBlit() hardware/blit.h
```

1.117 graphics.library/QBSBlit

```
 NAME

QBSBlit -- Synchronize the blitter request with the video beam.

 SYNOPSIS

QBSBlit( bsp )
   a1

void QBSBlit( struct bltnode * );

 FUNCTION
Call a user routine for use of the blitter, enqueued separately from
     the QBlit queue.  Calls the user routine contained in the blit
     structure when the video beam is located at a specified position
     onscreen.   Useful when you are trying to blit into a visible part
     of the screen and wish to perform the data move while the beam is
     not trying to display that same area.  (prevents showing part of
     an old display and part of a new display simultaneously).  Blitter
     requests on the QBSBlit queue take precedence over those on the
     regular blitter queue. The beam position is specified the blitnode.

 INPUTS
bsp - pointer to a blit structure.  See description in the
           Graphics Support section of the manual for more info.

 RESULT
     User routine is called when the QBSBlit queue reaches this
     request AND the video beam is in the specified position.
     If there are lots of blits going on and the video beam
     has wrapped around back to the top it will call all the
     remaining bltnodes as fast as it can to try and catch up.
```

```
 NOTES
     QBlit(), and QBSBlit() have been rewritten for V39.  Queued
     blits are now handled in FIFO order.  Tasks trying to
     OwnBlitter() are now given a fair share of the total
     blitter time available.  QBSBlit() are no longer queued
     separately from nodes added by QBlit().  This fixes the
     ordering dependencies listed under BUGS in prior autodoc
     notes.

 BUGS

 SEE ALSO
QBlit() hardware/blit.h
```

1.118 graphics.library/ReadPixel

```
 NAME
     ReadPixel -- read the pen number value of the pixel at a
                  specified x,y location within a certain RastPort.

 SYNOPSIS
     penno = ReadPixel( rp,    x,    y )
       d0               a1  d0:16 d1:16

LONG ReadPixel( struct RastPort *, SHORT, SHORT );

 FUNCTION
     Combine the bits from each of the bit-planes used to describe
     a particular RastPort into the pen number selector which that
     bit combination normally forms for the system hardware selection
     of pixel color.

 INPUTS
     rp -  pointer to a RastPort structure
     (x,y) a point in the RastPort

 RESULT
     penno - the pen number of the pixel at (x,y) is returned.
  -1 is returned if the pixel cannot be read for some reason.

 BUGS

 SEE ALSO
     WritePixel() graphics/rastport.h
```

1.119 graphics.library/ReadPixelArray8

```
 NAME
ReadPixelArray8 -- read the pen number value of a rectangular array
of pixels starting at a specified x,y location and continuing
through to another x,y location within a certain RastPort. (V36)
```

```
 SYNOPSIS
count = ReadPixelArray8(rp,xstart,ystart,xstop,ystop,array,temprp)
D0                      A0 D0:16  D1:16  D2:16 D3:16 A2    A1

   LONG ReadPixelArray8(struct  RastPort *, UWORD, UWORD, UWORD, UWORD,
   UBYTE *, struct RastPort *);

 FUNCTION
For each pixel in a rectangular region, combine the bits from each
of the bit-planes used to describe a particular RastPort into the pen
number selector which that bit combination normally forms for the
system hardware selection of pixel color.

 INPUTS
rp    -  pointer to a RastPort structure
(xstart,ystart) - starting point in the RastPort
(xstop,ystop)   - stopping point in the RastPort
array - pointer to an array of UBYTEs from which to fetch the pixel data
        allocate at least ((((width+15)>>4)<<4)*(ystop-ystart+1)) bytes.
temprp - temporary rastport (copy of rp with Layer set == NULL,
         temporary memory allocated for
         temprp->BitMap with Rows set == 1,
         temprp->BytesPerRow == (((width+15)>>4)<<1),
         and temporary memory allocated for
         temprp->BitMap->Planes[])

 RESULT
For each pixel in the array:
    Pen - (0..255) number at that position is returned
count   - the number of pixels read.

 NOTE
xstop must be >= xstart
ystop must be >= ystart

 BUGS

 SEE ALSO
ReadPixel()  ReadPixelLine8()  graphics/rastport.h
```

1.120 graphics.library/ReadPixelLine8

```
 NAME
ReadPixelLine8 -- read the pen number value of a horizontal line
of pixels starting at a specified x,y location and continuing
right for count pixels. (V36)

 SYNOPSIS
count = ReadPixelLine8(rp,xstart,ystart,width,array,temprp)
D0                     A0 D0:16  D1:16  D2    A2    A1

LONG ReadPixelLine8(struct RastPort *, UWORD, UWORD, UWORD,
     UBYTE *, struct RastPort * );
```

```
 FUNCTION
For each pixel in a rectangular region, combine the bits from each
of the bit-planes used to describe a particular RastPort into the pen
number selector which that bit combination normally forms for the
system hardware selection of pixel color.

 INPUTS
rp    - pointer to a RastPort structure
(x,y) - a point in the RastPort
width - count of horizontal pixels to read
array - pointer to an array of UBYTEs from which to fetch the pixel data
        allocate at least (((width+15)>>4)<<4) bytes.
temprp - temporary rastport (copy of rp with Layer set == NULL,
         temporary memory allocated for
         temprp->BitMap with Rows set == 1,
         temprp->BytesPerRow == (((width+15)>>4)<<1),
         and temporary memory allocated for
         temprp->BitMap->Planes[])

 RESULT
For each pixel in the array:
    Pen - (0..255) number at that position is returned
count   - the number of pixels read.

 NOTE
width must be non negative

 BUGS

 SEE ALSO
ReadPixel()  graphics/rastport.h
```

1.121 graphics.library/RectFill

```
 NAME
     RectFill -- Fill a rectangular region in a RastPort.

 SYNOPSIS

RectFill( rp, xmin, ymin, xmax, ymax)
               a1  d0:16 d1:16 d2:16 d3:16

void RectFill( struct RastPort *, SHORT, SHORT, SHORT, SHORT );

 FUNCTION
Fills  the  rectangular  region  specified  by  the
parameters  with the chosen pen  colors,  areafill
pattern, and drawing mode. If no areafill pattern is
specified, fill the rectangular region with the FgPen
color, taking into account the drawing mode.

 INPUTS
rp - pointer to a RastPort structure
(xmin,ymin) (xmax,ymax) are the coordinates of the upper
  left corner and the lower right corner, respectively, of the
```

```
        rectangle.
 NOTE

The following relation MUST be true:
  (xmax >= xmin) and (ymax >= ymin)

 BUGS
Complement mode with FgPen complements all bitplanes.

 SEE ALSO
AreaEnd() graphics/rastport.h
```

1.122 graphics.library/ReleasePen

```
 NAME
ReleasePen -- Release an allocated palette entry to the free pool. (V39)


 SYNOPSIS
ReleasePen( cm, n)
            a0 d0

void ReleasePen( Struct ColorMap *, ULONG);

 FUNCTION
Return the palette entry for use by other applications.
If the reference count for this palette entry goes to zero,
then it may be reset to another RGB value.

 INPUTS
     cm  =  A pointer to a color map created by GetColorMap().

n    =  A palette index obtained via any of the palette allocation
        functions. Passing a -1 will result in this call doing
        nothing.

 BUGS

 NOTES
This function works for both shared and exclusive palette entries.

 SEE ALSO
GetColorMap() ObtainPen() ObtainBestPenA()
```

1.123 graphics.library/RemBob

```
 NAME
RemBob -- Macro to remove a Bob from the gel list.

 SYNOPSIS
RemBob(bob)
```

```
RemBob(struct Bob *);

 FUNCTION
Marks a Bob as no-longer-required.  The gels internal code then
removes the Bob from the list of active gels the next time
DrawGList is executed. This is implemented as a macro.
If the user is double-buffering the Bob, it could take two
calls to DrawGList before the Bob actually disappears from
the RastPort.

 INPUTS
Bob = pointer to the Bob to be removed

 RESULT

 BUGS

 SEE ALSO
RemIBob()  DrawGList()  graphics/gels.h  graphics/gfxmacros.h
```

1.124 graphics.library/RemFont

```
 NAME
RemFont -- Remove a font from the system list.

 SYNOPSIS
RemFont(textFont)
        A1

void RemFont(struct TextFont *);

 FUNCTION
This function removes a font from the system, ensuring that
access to it is restricted to those applications that
currently have an active pointer to it: i.e. no new SetFont
requests to this font are satisfied.

 INPUTS
textFont - the TextFont structure to remove.

 RESULT

 BUGS

 SEE ALSO
SetFont()  AddFont()  graphics/text.h
```

1.125 graphics.library/RemIBob

```
 NAME
RemIBob -- Immediately remove a Bob from the gel list and the RastPort.
```

```
 SYNOPSIS
RemIBob(bob, rp, vp)
        A0   A1  A2

void RemIBob(struct Bob *, struct RastPort *, struct ViewPort *);

 FUNCTION
Removes a Bob immediately by uncoupling it from the gel list and
erases it from the RastPort.

 INPUTS
bob = pointer to the Bob to be removed
rp  = pointer to the RastPort if the Bob is to be erased
vp  = pointer to the ViewPort for beam-synchronizing

 RESULT

 BUGS

 SEE ALSO
InitGels()  RemVSprite()  graphics/gels.h
```

1.126 graphics.library/RemVSprite

```
 NAME
RemVSprite -- Remove a VSprite from the current gel list.

 SYNOPSIS
RemVSprite(vs)
           A0

void RemVSprite(struct VSprite *);

 FUNCTION
Unlinks the VSprite from the current gel list.

 INPUTS
vs = pointer to the VSprite structure to be removed from the gel list

 RESULT

 BUGS

 SEE ALSO
InitGels()  RemIBob()  graphics/gels.h
```

1.127 graphics.library/ScalerDiv

```
 NAME
ScalerDiv -- Get the scaling result that BitMapScale would. (V36)

 SYNOPSIS
```

```
result = ScalerDiv(factor, numerator, denominator)
D0                 D0      D1         D2

UWORD ScalerDiv(UWORD, UWORD, UWORD);

 FUNCTION
Calculate the expression (factor*numerator/denominator) such
that the result is the same as the width of the destination
result of BitMapScale when the factor here is the width of
the source, and the numerator and denominator are the
XDestFactor and XSrcFactor for BitMapScale.

 INPUTS
factor                 - a number in the range 0..16383
numerator, denominator - numbers in the range 1..16383

 RESULT
this returns factor*numerator/denominator
```

1.128 graphics.library/ScrollRaster

```
 NAME
 ScrollRaster -- Push bits in rectangle in raster around by
                 dx,dy towards 0,0 inside rectangle.
 SYNOPSIS
 ScrollRaster(rp, dx, dy, xmin, ymin, xmax, ymax)
              A1  D0  D1  D2    D3    D4    D5

 void ScrollRaster
      (struct RastPort *, WORD, WORD, WORD, WORD, WORD, WORD);

 FUNCTION
 Move the bits in the raster by (dx,dy) towards (0,0)
 The space vacated is RectFilled with BGPen.
 Limit the scroll operation to the rectangle defined
 by (xmin,ymin)(xmax,ymax). Bits outside will not be
 affected. If xmax,ymax is outside the rastport then use
 the lower right corner of the rastport.
 If you are dealing with a SimpleRefresh layered RastPort you
 should check rp->Layer->Flags & LAYERREFRESH to see if
 there is any damage in the damage list.  If there is you should
 call the appropriate BeginRefresh(Intuition) or BeginUpdate(graphics)
 routine sequence.

 INPUTS
 rp - pointer to a RastPort structure
 dx,dy are integers that may be positive, zero, or negative
 xmin,ymin - upper left of bounding rectangle
 xmax,ymax - lower right of bounding rectangle

 EXAMPLE
 ScrollRaster(rp,0,1,minx,miny,maxx,maxy) /* shift raster up by one row */
 ScrollRaster(rp,-1,-1,minx,miny,maxx,maxy)
    /* shift raster down and to the right by 1 pixel
```

```
BUGS
In 1.2/V1.3 if you ScrollRaster a SUPERBITMAP exactly left or
right, and there is no TmpRas attached to the RastPort, the system
will allocate one for you, but will never free it or record its
location. This bug has been fixed for V36.  The workaround for
1.2/1.3 is to attach a valid TmpRas of size at least
MAXBYTESPERROW to the RastPort before the call.

Beginning with V36 ScrollRaster adds the shifted areas into the
damage list for SIMPLE_REFRESH windows. Due to unacceptable
system overhead, the decision was made NOT to propagate this
shifted area damage for SMART_REFRESH windows.

SEE ALSO
ScrollRasterBF() graphics/rastport.h
```

1.129 graphics.library/ScrollRasterBF

```
NAME
ScrollRasterBF -- Push bits in rectangle in raster around by
                dx,dy towards 0,0 inside rectangle. Newly empty areas
   will be filled via EraseRect(). (V39)

SYNOPSIS
ScrollRasterBF(rp, dx, dy, xmin, ymin, xmax, ymax)
             A1  D0  D1  D2    D3    D4    D5

void ScrollRasterBF
     (struct RastPort *, WORD, WORD, WORD, WORD, WORD, WORD);

FUNCTION
Move the bits in the raster by (dx,dy) towards (0,0)
The space vacated is filled by calling EraseRect().
Limit the scroll operation to the rectangle defined
by (xmin,ymin)(xmax,ymax). Bits outside will not be
affected. If xmax,ymax is outside the rastport then use
the lower right corner of the rastport.
If you are dealing with a SimpleRefresh layered RastPort you
should check rp->Layer->Flags & LAYERREFRESH to see if
there is any damage in the damage list.  If there is you should
call the appropriate BeginRefresh(Intuition) or BeginUpdate(graphics)
routine sequence.

INPUTS
rp - pointer to a RastPort structure
dx,dy are integers that may be positive, zero, or negative
xmin,ymin - upper left of bounding rectangle
xmax,ymax - lower right of bounding rectangle

NOTES

This call is exactly the same as ScrollRaster, except that it calls
EraseRect() instead of RectFill() when clearing the newly exposed
area. This allows use of a custom layer backfill hook.
```

```
 BUGS

 SEE ALSO
 ScrollRaster() EraseRect() intuition.library/ScrollWindowRaster()
 graphics/rastport.h
```

1.130 graphics.library/ScrollVPort

```
 NAME
ScrollVPort -- Reinterpret RasInfo information in ViewPort to reflect
    the current Offset values.

 SYNOPSIS
ScrollVPort( vp )
       a0

void ScrollVPort(struct ViewPort *vp);

 FUNCTION
After the programmer has adjusted the Offset values in
the RasInfo structures of ViewPort, change the
the copper lists to reflect the the Scroll positions.
Changing the BitMap ptr in RasInfo and not changing the
the Offsets will effect a double buffering affect.

 INPUTS
     vp - pointer to a ViewPort structure
     that is currently be displayed.
 RESULTS
modifies hardware and intermediate copperlists to reflect
new RasInfo

 BUGS
     pokes not fast enough to avoid some visible hashing of display (V37)
This function was re-written in V39 and is ~10 times faster than
before.

 SEE ALSO
MakeVPort() MrgCop() LoadView()  graphics/view.h
```

1.131 graphics.library/SetABPenDrMd

```
 NAME
SetABPenDrMd -- Set pen colors and draw mode for a RastPort. (V39)

 SYNOPSIS
SetABPenDrMd( rp, apen, bpen, mode )
        a1  d0     d1    d2

void SetABPenDrMd( struct RastPort *, ULONG, ULONG, ULONG );

 FUNCTION
```

```
Set the pen values and drawing mode for lines, fills and text.
Get the bit definitions from rastport.h

 INPUTS
rp - pointer to RastPort structure.
apen - primary pen value
bpen - secondary pen value
mode - 0-255, some combinations may not make much sense.

 RESULT
The mode set is dependent on the bits selected.
Changes minterms to reflect new drawing mode and colors.
Sets line drawer to restart pattern.

NOTES
This call is essentially the same as a sequence of
SetAPen()/SetBPen()/SetDrMD() calls, except that it is
significantly faster. The minterms will only be generated
once, or not at all if nothing changed (warning to illegal
RastPort pokers!).

 BUGS

 SEE ALSO
SetAPen() SetBPen() SetDrMd() graphics/rastport.h
```

1.132 graphics.library/SetAPen

```
 NAME
SetAPen -- Set the primary pen for a RastPort.

 SYNOPSIS
SetAPen( rp, pen )
   a1  d0

void SetAPen( struct RastPort *, UBYTE );

 FUNCTION
Set the primary drawing pen for lines, fills, and text.

 INPUTS
rp - pointer to RastPort structure.
pen - (0-255)

 RESULT
Changes the minterms in the RastPort to reflect new primary pen.
Sets line drawer to restart pattern.

 BUGS

 SEE ALSO
SetBPen() graphics/rastport.h
```

1.133 graphics.library/SetBPen

```
 NAME
SetBPen -- Set secondary pen for a RastPort

 SYNOPSIS
SetBPen( rp, pen )
   a1  d0

void SetBPen( struct RastPort *, UBYTE );

 FUNCTION
Set the secondary drawing pen for lines, fills, and text.

 INPUTS
rp - pointer to RastPort structure.
pen - (0-255)

 RESULT
Changes the minterms in the RastPort to reflect new secondary pen.
Sets line drawer to restart pattern.

 BUGS

  SEE ALSO
SetAPen() graphics/rastport.h
```

1.134 graphics.library/SetChipRev

```
 NAME
SetChipRev -- turns on the features of a Chip Set  (V39)

 SYNOPSIS
chiprevbits = SetChipRev(ChipRev)
                         d0

ULONG SetChipRev(ULONG);

 FUNCTION
Enables the features of the requested Chip Set if available,
and updates the graphics database accordingly.

 INPUTS
ChipRev - Chip Rev that you would like to be enabled.

 RESULT
chiprevbits - Actual bits set in GfxBase->ChipRevBits0.

 NOTES
This routine should only be called once. It will be called by the system
in the startup-sequence, but is included in the autodocs for authors
of bootblock-games that wish to take advantage of post-ECS features.

 SEE ALSO
```

```
<graphics/gfxbase.h>
```

1.135 graphics.library/SetCollision

```
 NAME
SetCollision -- Set a pointer to a user collision routine.

 SYNOPSIS
SetCollision(num, routine, GInfo)
             D0   A0       A1

void SetCollision(ULONG, VOID (*)(), struct GelsInfo *);

 FUNCTION
Sets a specified entry (num) in the user's collision vectors table
equal to the address of the specified collision routine.

 INPUTS
num     = collision vector number
routine = pointer to the user's collision routine
GInfo   = pointer to a GelsInfo structure

 RESULT

 BUGS

 SEE ALSO
InitGels()  graphics/gels.h  graphics/rastport.h
```

1.136 graphics.library/SetDrMd

```
 NAME
SetDrMd -- Set drawing mode for a RastPort

 SYNOPSIS
SetDrMd( rp, mode )
   a1  d0:8

void SetDrMd( struct RastPort *, UBYTE );

 FUNCTION
Set the drawing mode for lines, fills and text.
Get the bit definitions from rastport.h

 INPUTS
rp - pointer to RastPort structure.
mode - 0-255, some combinations may not make much sense.

 RESULT
The mode set is dependent on the bits selected.
Changes minterms to reflect new drawing mode.
Sets line drawer to restart pattern.
```

```
 BUGS

 SEE ALSO
SetAPen() SetBPen() graphics/rastport.h
```

1.137 graphics.library/SetFont

```
 NAME
SetFont -- Set the text font and attributes in a RastPort.

 SYNOPSIS
SetFont(rp, font)
        A1   A0

void SetFont(struct RastPort *, struct TextFont *);

 FUNCTION
This function sets the font in the RastPort to that described
by font, and updates the text attributes to reflect that
change.  This function clears the effect of any previous
soft styles.

 INPUTS
rp   - the RastPort in which the text attributes are to be changed
font - pointer to a TextFont structure returned from OpenFont()
       or OpenDiskFont()

 RESULT

 NOTES
This function had previously been documented that it would
accept a null font.  This practice is discouraged.
o   Use of a RastPort with a null font with text routines has
    always been incorrect and risked the guru.
o   Keeping an obsolete font pointer in the RastPort is no more
    dangerous than keeping a zero one there.
o   SetFont(rp, 0) causes spurious low memory accesses under
    some system software releases.

As of V36, the following Amiga font variants are no longer
directly supported:
    fonts with NULL tf_CharSpace and non-NULL tf_CharKern.
    fonts with non-NULL tf_CharSpace and NULL tf_CharKern.
    fonts with NULL tf_CharSpace and NULL tf_CharKern with
  a tf_CharLoc size component greater than tf_XSize.
Attempts to SetFont these one of these font variants will
cause the system to modify your font to make it acceptable.

 BUGS
Calling SetFont() on in-code TextFonts (ie fonts not
OpenFont()ed) will result in a loss of 24 bytes from
the system as of V36.
This can be resolved by calling StripFont().
```

```
 SEE ALSO
OpenFont()  StripFont()
diskfont.library/OpenDiskFont()  graphics/text.h
```

1.138 graphics.library/SetMaxPen

```
 NAME
     SetMaxPen -- set maximum pen value for a rastport (V39).


 SYNOPSIS
     SetMaxPen ( rp, maxpen)
        a0   d0

void SetMaxPen(struct RastPort *,ULONG)

 FUNCTION
This will instruct the graphics library that the owner of the rastport
will not be rendering in any colors whose index is >maxpen. If there
are any speed optimizations which the graphics device can make based
on this fact (for instance, setting the pixel write mask), they will
be done.

Basically this call sets the rastport mask, if this would improve
speed. On devices where masking would slow things down (like with
chunky pixels), it will be a no-op.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.
maxpen =  a longword pen value.

 BUGS

 NOTES
The maximum pen value passed must take into account not only which
colors you intend to render in the future, but what colors you will
be rendering on top of.
SetMaxPen(rp,0) doesn't make much sense.

 SEE ALSO
SetWriteMask()
```

1.139 graphics.library/SetOPen

```
 NAME
SetOPen -- Change the Area OutLine pen and turn on Outline
    mode for areafills.

 SYNOPSIS
SetOPen(rp, pen)

void SetOPen( struct RastPort *, UBYTE );
```

```
 FUNCTION
This is implemented as a c-macro.
Pen is the pen number that will be used to draw a border
around an areafill during AreaEnd().

 INPUTS
rp = pointer to RastPort structure
pen = number  between 0-255

 BUGS

 SEE ALSO
AreaEnd() graphics/gfxmacros.h graphics/rastport.h
```

1.140 graphics.library/SetOutlinePen

```
 NAME
     SetOutlinePen -- Set the Outline Pen value for a RastPort (V39).


 SYNOPSIS
     old_pen=SetOutlinePen  ( rp, pen )
  d0                    a0   d0

ULONG SetOutlinePen(struct RastPort *,ULONG)

 FUNCTION
Set the current value of the O pen for the rastport and turn on area outline
mode. This function should be used instead of poking the structure directly,
because future graphics devices may store it differently, for instance,
using more bits.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.
pen =  a longword pen number

returns the previous outline pen
 BUGS

 NOTES

 SEE ALSO
GetOPen() graphics/gfxmacros.h
```

1.141 graphics.library/SetRast

```
 NAME
     SetRast - Set an entire drawing area to a specified color.

 SYNOPSIS
     SetRast( rp, pen )
```

```
              a1  d0

void SetRast( struct RastPort *, UBYTE );

 FUNCTION
     Set the entire contents of the specified RastPort to the
     specified pen.

 INPUTS
     rp - pointer to RastPort structure
     pen - the pen number (0-255) to jam into bitmap

 RESULT
     All pixels within the drawing area are set to the
selected pen number.

 BUGS

 SEE ALSO
RectFill() graphics/rastport.h
```

1.142 graphics.library/SetRGB32

```
 NAME
SetRGB32 -- Set one color register for this Viewport. (V39)

 SYNOPSIS
SetRGB32(  vp,  n,   r,    g,    b)
           a0  d0   d1    d2    d3

void SetRGB32( struct ViewPort *, ULONG, ULONG, ULONG, ULONG );

 INPUTS
vp = viewport
n = the number of the color register to set.
r = red level   (32 bit left justified fraction)
g = green level (32 bit left justified fraction)
b = blue level  (32 bit left justified fraction)

 RESULT
If there is a ColorMap for this viewport, then the value will
be stored in the ColorMap.
The selected color register is changed to match your specs.
If the color value is unused then nothing will happen.

 BUGS

 NOTES
Lower order bits of the palette specification will be discarded,
depending on the color palette resolution of the target graphics
device. Use 0xffffffff for the full value, 0x7fffffff for 50%,
etc. You can find out the palette range for your screen by
querying the graphics data base.

 SEE ALSO
```

```
  GetColorMap() GetRGB32() SetRGB32CM() LoadRGB32() graphics/view.h
```

1.143 graphics.library/SetRGB32CM

```
   NAME
       SetRGB32CM -- Set one color register for this ColorMap. (V39)

   SYNOPSIS
       SetRGB32CM(  cm,  n,   r,    g,    b)
                    a0  d0   d1    d2    d3

       void SetRGB4CM( struct ColorMap *, ULONG, ULONG, ULONG , ULONG);

   INPUTS
  cm = colormap
       n = the number of the color register to set. Must not exceed the numbe
r of colors
      allocated for the colormap.
       r = red level (32 bit unsigned left justified fraction)
       g = green level
       b = blue level

   RESULT
  Store the (r,g,b) triplet at index n of the ColorMap structure.
       This function can be used to set up a ColorMap before before
  linking it into a viewport.

   BUGS

   SEE ALSO
       GetColorMap() GetRGB32() SetRGB32() SetRGB4CM() graphics/view.h
```

1.144 graphics.library/SetRGB4

```
    NAME
       SetRGB4 -- Set one color register for this viewport.

    SYNOPSIS
       SetRGB4(  vp, n,   r,    g,    b)
                 a0  d0  d1:4  d2:4  d3:4

  void SetRGB4( struct ViewPort *, SHORT, UBYTE, UBYTE, UBYTE );

    FUNCTION
  Change the color look up table so that this viewport displays
  the color (r,g,b) for pen number n.

    INPUTS
  vp - pointer to  viewport structure
       n - the color number (range from 0 to 31)
       r - red level (0-15)
       g - green level (0-15)
```

```
     b - blue level (0-15)

  RESULT
If there is a ColorMap for this viewport, then the value will
be stored in the ColorMap.
     The selected color register is changed to match your specs.
If the color value is unused then nothing will happen.

  BUGS

NOTE: Under V36 and up, it is not safe to call this function
from an interrupt, due to semaphore protection of graphics
copper lists.

  SEE ALSO
     LoadRGB4() GetRGB4() graphics/view.h
```

1.145 graphics.library/SetRGB4CM

```
 NAME
     SetRGB4CM -- Set one color register for this ColorMap.

 SYNOPSIS
     SetRGB4CM(  cm,  n,   r,    g,    b)
                 a0  d0  d1:4  d2:4  d3:4

     void SetRGB4CM( struct ColorMap *, SHORT, UBYTE, UBYTE, UBYTE );

 INPUTS
cm = colormap
     n = the number of the color register to set. Ranges from 0 to 31
    on current Amiga displays.
     r = red level (0-15)
     g = green level (0-15)
     b = blue level (0-15)

 RESULT
Store the (r,g,b) triplet at index n of the ColorMap structure.
     This function can be used to set up a ColorMap before before
linking it into a viewport.

 BUGS

 SEE ALSO
     GetColorMap() GetRGB4() SetRGB4() graphics/view.h
```

1.146 graphics.library/SetRPAttrA

```
 NAME
     SetRPAttrA -- modify rastport settings via a tag list
SetRPAttrs  -- varargs stub for SetRPAttrA
```

```
 SYNOPSIS
     SetRPAttrA(rp,tags)
                a0   a1

void SetRPAttrA(struct RastPort *, struct TagItem *);

SetRPAttrs(rp,tag,...);

 FUNCTION
     Modify settings of a rastport, based on the taglist passed.
currently available tags are:

  RPTAG_Font    Font for Text()
  RPTAG_SoftStyle   style for text (see graphics/text.h)
  RPTAG_APen    Primary rendering pen
  RPTAG_BPen    Secondary rendering pen
  RPTAG_DrMd    Drawing mode (see graphics/rastport.h)
  RPTAG_OutLinePen  Area Outline pen
  RPTAG_WriteMask   Bit Mask for writing.
  RPTAG_MaxPen    Maximum pen to render (see SetMaxPen())

 INPUTS
rp - pointer to the RastPort to modify.
tags - a standard tag list

 RESULT

 BUGS

 SEE ALSO
SetFont() SetSoftStyle() SetAPen() SetBPen() SetDrMd() SetOutLinePen()
SetWriteMask() SetMaxPen() GetRPAttrA() graphics/rpattr.h
```

1.147 graphics.library/SetSoftStyle

```
 NAME
SetSoftStyle -- Set the soft style of the current font.

 SYNOPSIS
newStyle = SetSoftStyle(rp, style, enable)
D0                      A1  D0     D1

ULONG SetSoftStyle(struct RastPort *, ULONG, ULONG);

 FUNCTION
This function alters the soft style of the current font.  Only
those bits that are also set in enable are affected.  The
resulting style is returned, since some style request changes
will not be honored when the implicit style of the font
precludes changing them.

 INPUTS
rp     - the RastPort from which the font and style
         are extracted.
style  - the new font style to set, subject to enable.
```

```
enable - those bits in style to be changed.  Any set bits here
         that would not be set as a result of AskSoftStyle will
         be ignored, and the newStyle result will not be as
         expected.

 RESULTS
newStyle - the resulting style, both as a result of previous
           soft style selection, the effect of this function,
           and the style inherent in the set font.

 BUGS

 SEE ALSO
AskSoftStyle()  graphics/text.h
```

1.148 graphics.library/SetWriteMask

```
 NAME
     SetWriteMask -- Set the pixel write mask value for a RastPort (V39).


 SYNOPSIS
     success=SetWriteMask ( rp, msk )
  d0                 a0   d0

ULONG SetWriteMask(struct RastPort *,ULONG)

 FUNCTION
Set the current value of the bit write mask for the rastport.
bits of the pixel with zeros in their mask will not be modified by
subsequent drawing operations.

 INPUTS
     rp  =  a pointer to a valid RastPort structure.
msk =  a longword mask value.

Graphics devices which do not support per-bit masking will
return 0 (failure).
 BUGS

 NOTES

 SEE ALSO
graphics/gfxmacros.h
```

1.149 graphics.library/SortGList

```
 NAME
SortGList -- Sort the current gel list, ordering its y,x coordinates.

 SYNOPSIS
SortGList(rp)
```

```
          A1

void SortGList(struct RastPort *);

 FUNCTION
Sorts the current gel list according to the gels' y,x coordinates.
This sorting is essential before calls to DrawGList or DoCollision.

 INPUTS
rp = pointer to the RastPort structure containing the GelsInfo

 RESULT

 BUGS

 SEE ALSO
InitGels()  DoCollision()  DrawGList()  graphics/rastport.h
```

1.150 graphics.library/StripFont

```
 NAME
StripFont -- remove the tf_Extension from a font (V36)

 SYNOPSIS
StripFont(font)
          A0

VOID StripFont(struct TextFont *);
```

1.151 graphics.library/SyncSBitMap

```
 NAME
     SyncSBitMap -- Syncronize Super BitMap with whatever is
    in the standard Layer bounds.

 SYNOPSIS
     SyncSBitMap( layer )
                  a0

void SyncSBitMap( struct Layer * );

 FUNCTION
     Copy all bits from ClipRects in Layer into Super BitMap
BitMap.  This is used for those functions that do not
want to deal with the ClipRect structures but do want
to be able to work with a SuperBitMap Layer.

 INPUTS
layer - pointer to a Layer that has a SuperBitMap
  The Layer should already be locked by the caller.

 RESULT
```

```
After calling this function, the programmer can manipulate
the bits in the superbitmap associated with the layer.
Afterwards, the programmer should call CopySBitMap to
copy the bits back into the onscreen layer.

 BUGS

 SEE ALSO
CopySBitMap() graphics/clip.h
```

1.152 graphics.library/Text

```
 NAME
Text -- Write text characters (no formatting).

 SYNOPSIS
Text(rp, string, length)
     A1  A0      D0-0:16

void Text(struct RastPort *, STRPTR, WORD);

 FUNCTION
This graphics function writes printable text characters to the
specified RastPort at the current position.  No control meaning
is applied to any of the characters, thus only text on the
current line is output.

The current position in the RastPort is updated to the next
character position.
If the characters displayed run past the RastPort boundary,
the current position is truncated to the boundary, and
thus does not equal the old position plus the text length.

 INPUTS
rp     - a pointer to the RastPort which describes where the
         text is to be output
string - the address of string to output
length - the number of characters in the string.
         If zero, there are no characters to be output.

 NOTES
o   This function may use the blitter.
o   Changing the text direction with RastPort->TxSpacing is
    not supported.

 BUGS
For V34 and earlier:
o   The maximum string length (in pixels) is limited to
    (1024 - 16 = 1008) pixels wide.
o   A text string whose last character(s) have a
    tf_CharLoc size component that extends to the right of
    the rightmost of the initial and final CP positions
    will be (inappropriately) clipped.

 SEE ALSO
```

```
Move()  TextLength()  graphics/text.h  graphics/rastport.h
```

1.153 graphics.library/TextExtent

```
 NAME
TextExtent -- Determine raster extent of text data. (V36)

 SYNOPSIS
TextExtent(rp, string, count, textExtent)
           A1  A0      D0:16  A2

void textExtent(struct RastPort *, STRPTR, WORD,
     struct TextExtent *);

 FUNCTION
This function determines a more complete metric of the space
that a text string would render into than the TextLength()
function.

 INPUTS
rp     - a pointer to the RastPort which describes where the
         text attributes reside.
string - the address of the string to determine the length of.
count  - the number of characters in the string.
              If zero, there are no characters in the string.
textExtent - a structure to hold the result.

 RESULTS
textExtent is filled in as follows:
    te_Width  - same as TextLength() result: the rp_cp_x
                advance that rendering this text would cause.
    te_Height - same as tf_YSize.  The height of the
                font.
    te_Extent.MinX - the offset to the left side of the
                rectangle this would render into.  Often zero.
    te_Extent.MinY - same as -tf_Baseline.  The offset
                from the baseline to the top of the rectangle
                this would render into.
    te_Extent.MaxX - the offset of the left side of the
                rectangle this would render into.  Often the
                same as te_Width-1.
    te_Extent.MaxY - same as tf_YSize-tf_Baseline-1.
                The offset from the baseline to the bottom of
                the rectangle this would render into.

 SEE ALSO
TextLength()  Text()  TextFit()
graphics/text.h  graphics/rastport.h
```

1.154 graphics.library/TextFit

```
 NAME
TextFit - count characters that will fit in a given extent (V36)

 SYNOPSIS
chars = TextFit(rastport, string, strLen, textExtent,
D0              A1        A0      D0      A2
        constrainingExtent, strDirection,
        A3                  D1
        constrainingBitWidth, constrainingBitHeight)
        D2                    D3

ULONG TextFit(struct RastPort *, STRPTR, UWORD,
    struct TextExtent *, struct TextExtent *, WORD, UWORD, UWORD);

 FUNCTION
This function determines how many of the characters of the
provided string will fit into the space described by the
constraining parameters.  It also returns the extent of
that number of characters.

 INPUTS
rp     - a pointer to the RastPort which describes where the
         text attributes reside.
string - the address of string to determine the constraint of
strLen - The number of characters in the string.
         If zero, there are no characters in the string.
textExtent - a structure to hold the extent result.
constrainingExtent - the extent that the text must fit in.
    This can be NULL, indicating only the constrainingBit
    dimensions will describe the constraint.
strDirection - the offset to add to the string pointer to
    get to the next character in the string.  Usually 1.
    Set to -1 and the string to the end of the string to
    perform a TextFit() anchored at the end.  No other value
    is valid.
constrainingBitWidth - an alternative way to specify the
    rendering box constraint width that is independent of
    the rendering origin.  Range 0..32767.
constrainingBitHeight - an alternative way to specify the
    rendering box constraint height that is independent of
    the rendering origin.  Range 0..32767.

 RESULTS
chars - the number of characters from the origin of the
        given string that will fit in both the constraining
        extent (which specifies a CP bound and a rendering
        box relative to the origin) and in the rendering width
        and height specified.

 NOTES
The result is zero chars and an empty textExtent when the fit
cannot be performed.  This occurs not only when no text will
fit in the provided constraints, but also when:
-   the RastPort rp's rp_TxSpacing sign and magnitude is so
    great it reverses the path of the text.
-   the constrainingExtent does not include x = 0.
```

```
 BUGS
Under V37, TextFit() would return one too few characters if the
font was proportional. This can be worked around by passing
(constrainingBitWidth + 1) for proportional fonts. This is fixed
for V39.

 SEE ALSO
TextExtent()  TextLength()  Text()
graphics/text.h  graphics/rastport.h
```

1.155 graphics.library/TextLength

```
 NAME
TextLength -- Determine raster length of text data.

 SYNOPSIS
length = TextLength(rp, string, count)
D0                  A1  A0      D0:16

WORD TextLength(struct RastPort *, STRPTR, WORD);

 FUNCTION
This graphics function determines the length that text data
would occupy if output to the specified RastPort with the
current attributes.  The length is specified as the number of
raster dots: to determine what the current position would be
after a Text() using this string, add the length to cp_x
(cp_y is unchanged by Text()).  Use the newer TextExtent() to
get more information.

 INPUTS
rp     - a pointer to the RastPort which describes where the
         text attributes reside.
string - the address of string to determine the length of
count  - the string length.  If zero, there are no characters
         in the string.

 RESULTS
length - the number of pixels in x this text would occupy, not
         including any negative kerning that may take place at
         the beginning of the text string, nor taking into
         account the effects of any clipping that may take
         place.

 NOTES
Prior to V36, the result length occupied only the low word of
d0 and was not sign extended into the high word.

 BUGS
A length that would overflow single word arithmetic is not
calculated correctly.

 SEE ALSO
```

```
TextExtent()  Text()  TextFit()
graphics/text.h  graphics/rastport.h
```

1.156 graphics.library/UnlockLayerRom

```
 NAME
UnlockLayerRom -- Unlock Layer structure by ROM(gfx lib) code.

 SYNOPSIS
UnlockLayerRom( layer )
     a5

void UnlockLayerRom( struct Layer * );

 FUNCTION
Release the lock on this layer. If the same task has called
LockLayerRom more than once than the same number of calls to
UnlockLayerRom must happen before the layer is actually freed
so that other tasks may use it.
This call does destroy scratch registers.
This call is identical to UnlockLayer (layers.library).

 INPUTS
layer - pointer to Layer structure

 BUGS

 SEE ALSO
LockLayerRom() layers.library/UnlockLayer() graphics/clip.h
```

1.157 graphics.library/VBeamPos

```
 NAME
VBeamPos -- Get vertical beam position at this instant.

 SYNOPSIS
pos = VBeamPos()
 d0

LONG VBeamPos( void );

 FUNCTION
Get the vertical beam position from the hardware.

 INPUTS
none

 RESULT
interrogates hardware for beam position and returns value.
valid results in are the range of 0-511.
Because of multitasking, the actual value returned may have
no use. If you are the highest priority task then the value
```

```
  returned should be close, within 1 line.

   BUGS

   SEE ALSO
```

1.158 graphics.library/VideoControl

```
   NAME
  VideoControl -- Modify the operation of a ViewPort's ColorMap (V36)
  VideoControlTags -- varargs stub for VideoControl (V36)
   SYNOPSIS
  error = VideoControl( cm , tags )
  d0                    a0   a1

  ULONG VideoControl( struct ColorMap *, struct TagItem * );

  error= VideoControlTags(cm, tags,...);


   FUNCTION
  Process the commands in the VideoControl command TagItem buffer
  using cm as the target, with respect to its "attached" ViewPort.

  viewport commands:

  VTAG_ATTACH_CM     [_SET        | _GET] -- set/get attached viewport
  VTAG_VIEWPORTEXTRA [_SET        | _GET] -- set/get attached vp_extra
  VTAG_NORMAL_DISP   [_SET        | _GET] -- set/get DisplayInfoHandle
                                           (natural mode)
  VTAG_COERCE_DISP   [_SET        | _GET] -- set/get DisplayInfoHandle
                                           (coerced mode)
  VTAG_PF1_BASE      [_SET        | _GET] -- set/get color base for
                                           first playfield. (V39)
  VTAG_PF2_BASE      [_SET        | _GET] -- set/get color base for
                                           second playfield. (V39)
  VTAG_SPODD_BASE    [_SET        | _GET] -- set/get color base for odd
                                           sprites. (V39)
  VTAG_SPEVEN_BASE   [_SET        | _GET] -- set/get color base for even
                                           sprites. (V39)
  VTAG_BORDERSPRITE  [_SET        | _GET] -- on/off/inquire sprites in
                                           borders. (V39)
  VTAG_SPRITERESN    [_SET        | _GET] -- set/get sprite resolution
                (legal values are SPRITERESN_ECS/_140NS/_70NS/_35NS.
                 see graphics/view.h) (V39)
  VTAG_PF1_TO_SPRITEPRI [_SET     | _GET] -- set/get playfield1 priority
                                                with respect to sprites (V3
9)
  VTAG_PF2_TO_SPRITEPRI [_SET     | _GET] -- set/get playfield2 priority
                                                with respect to sprites (V3
9)
   (These two require that the ColorMap is attached to a ViewPort to be
    effective).
```

```
genlock commands:

VTAG_BORDERBLANK   [_SET | _CLR | _GET] -- on/off/inquire blanking
VTAG_BORDERNOTRANS [_SET | _CLR | _GET] -- on/off/inquire notransparency
VTAG_CHROMAKEY     [_SET | _CLR | _GET] -- on/off/inquire chroma mode
VTAG_BITPLANEKEY   [_SET | _CLR | _GET] -- on/off/inquire bitplane mode
VTAG_CHROMA_PEN    [_SET | _CLR | _GET] -- set/clr/get chromakey pen #
VTAG_CHROMA_PLANE  [_SET |      | _GET] -- set/get bitplanekey plane #

control commands:

VTAG_IMMEDIATE  - normally, VideoControl changes do not occur until the
        next MakeVPort. Using this tag, some changes can be made to
        happen immediately. The tag data is a pointer to a longword
        flag variable which will be cleared if all changes happened
        immediately. See the example. (V39)

VTAG_FULLPALETTE   [_SET | _CLR | _GET] -- enable/disable loading of all
        colors in the copper list.
        Normally, graphics will only load the color which are necessary
        for the viewport, based upon the screen depth and mode. In order
        to use the color palette banking features, you may need to use
        this tag to tell graphics to load ALL colors, regardless of
        screen depth. (V39)

VC_IntermediateCLUpdate
VC_IntermediateCLUpdate_Query
         When set, graphics will update the intermediate copper
   lists on colour changes. When FALSE, graphics won't update
         the intermediate copperlists, so ScrollVPort(),
         ChangeVPBitMap() and colour loading functions will be faster.
         This value is TRUE by default. (V40)

VC_NoColorPaletteLoad
VC_NoColorPaletteLoad_Query
         When set, only colour 0 will be loaded for this ViewPort,
         hence the inter-ViewPort gap will be smaller. The colours for
         this ViewPort are inherited from the next higher ViewPort. The
   results are undefined if this is the first or only ViewPort in
         the display, and undefined when used in conjunction with
         VTAG_FULLPALETTE (!?!).
         This value is FALSE by default. (V40)

VC_DUALPF_Disable
VC_DUALPF_Disable_Query
  When set, disables the setting of the dual-playfield
  bit in bplcon0. When used with a dual-playfield mode
  screen, this allows using separate scroll and bitmaps
  for the odd and even bitplanes, without going through
  the normal dual-playfield priority and palette selection.
  With appropriate palette setup, this can be used for
  transparency effects, etc.

copper commands

VTAG_USERCLIP      [_SET | _CLR | _GET] -- on/off/inquire clipping of
                                           UserCopperList at bottom
```

```
                                          edge of ColorMap->cm_vp
                                          (defaults to off)

buffer commands:

VTAG_NEXTBUF_CM                         -- link to more VTAG commands
VTAG_END_CM                             -- terminate command buffer

batch mode commands:

(if you want your videocontrol taglist to be processed in "batch"
 mode, that is, at the next MakeVPort() for the ColorMap->cm_vp;
 you may install a static list of videocontrol TagItems into the
 ColorMap with the BATCH_ITEMS_SET command; and then enable/disable
 batch mode processing of those items via the BATCH_CM control
 command)

VTAG_BATCH_CM      [_SET | _CLR | _GET] -- on/off/inquire batch mode
VTAG_BATCH_ITEMS   [_SET | _ADD | _GET] -- set/add/get batched TagLists

private commands (used internally by intuition -- do not call):

VTAG_VPMODEID      [_SET | _CLR | _GET] -- force GetVPModeID() return


 INPUTS
cm   = pointer to struct ColorMap obtained via GetColorMap().
tags = pointer to a table of videocontrol tagitems.

 RESULT
error = NULL if no error occurred in the control operation.
(non-NULL if bad colormap pointer, no tagitems or bad tag)

The operating characteristics of the ColorMap and its attached
ViewPort are modified. The result will be incorporated into the
ViewPort when its copper lists are reassembled via MakeVPort().

Note that you must NOT change colors in the viewport (via SetRGB4(),
LoadRGB4(), SetRGB4(), etc.) after changing any of the color palette
offsets (VTAG_PF1_BASE, etc), without first remaking the ViewPort.

 NOTES
Sprite resolutions is controlled by two sets of tags, SPRITERESN
and DEFSPRITERESN. If you don't set the sprite resolution, it will
follow the intuition-controlled "default" sprite resolution. Setting
the sprite resolution to one of the SPRITERESN_ values will allow the
application to override intuition's control of it.

This function will modify the contents of the TagList you pass to it by
changing _GET tags to the corresponding _SET or _CLR tag. The
exceptions to this rule are documented as such above (such as
VTAG_IMMEDIATE).

The new tags added for V40 have the prefix VC_ instead of VTAG_. These
tags work in the same manner as all other tags in the system, and will
not be modified by VideoControl().
```

```
 EXAMPLE
must_remake=-1;
error=VideoControl(myvp->ColorMap,VTAG_BORDERBLANK_SET,-1,
                   (GFXBase->lib_Version>=39)?VTAG_IMMEDIATE:TAG_IGNORE,
          &must_remake);
if (must_remake) { MakeVPort(myview,myvp); MrgCop(myview); }

 EXAMPLE
struct TagItem VCTags[] =
{
    {VTAG_BORDERBLANK_GET, NULL},
    {VTAG_SPRITERESN_SET, SPRITERESN_35NS},
    {TAG_DONE, NULL},
};
BOOL bblank = FALSE;

if (VideoControl(cm, VCTags) == NULL)
{
    bblank = (VCTags[0].ti_Tag == VTAG_BORDERBLANK_SET);
}

 EXAMPLE
struct TagItem VCTags[] =
{
    {VC_NoColorPaletteLoad_Query, NULL},
    {TAG_DONE},
};
ULONG query;

VCTags[0].ti_Data = (ULONG)&query;
if (VideoControl(cm, VCTags) == NULL)
{
    printf("Palette loading is %s\n", (query ? "off" : "on"));
}

 BUGS

 SEE ALSO
graphics/videocontrol.h, GetColorMap(), FreeColorMap()
```

1.159 graphics.library/WaitBlit

```
 NAME
     WaitBlit -- Wait for the blitter to be finished before proceeding
                 with anything else.

 SYNOPSIS
     WaitBlit()

void WaitBlit( void );

 FUNCTION
WaitBlit returns when the blitter is idle. This function should
normally only be used when dealing with the blitter in a
synchronous manner, such as when using OwnBlitter and DisownBlitter.
```

```
WaitBlit does not wait for all blits queued up using QBlit or
QBSBlit. You should call WaitBlit if you are just about to modify or
free some memory that the blitter may be using.

 INPUTS
     none

 RESULT
     Your program waits until the blitter is finished.
This routine does not use any the CPU registers.
do/d1/a0/a1 are preserved by this routine.
It may change the condition codes though.

 BUGS
When examining bits with the CPU right after a blit, or when freeing
temporary memory used by the blitter, a WaitBlit() may be required.

Note that many graphics calls fire up the blitter, and let it run.
The CPU does not need to wait for the blitter to finish before
returning.

Because of a bug in Agnus (prior to all revisions of fat Agnus)
this code may return too soon when the blitter has, in fact, not
started the blit yet, even though BltSize has been written.

This most often occurs in a heavily loaded system with extended memory,
HIRES, and 4 bitplanes.

WaitBlit currently tries to avoid this Agnus problem by testing
the BUSY bit multiple times to make sure the blitter has started.
If the blitter is BUSY at first check, this function busy waits.

This initial hardware bug was fixed as of the first "Fat Agnus" chip,
as used in all A500 and A2000 computers.

Because of a different bug in Agnus (currently all revisions thru ECS)
this code may return too soon when the blitter has, in fact, not
stopped the blit yet, even though blitter busy has been cleared.

This most often occurs in a heavily loaded system with extended memory,
in PRODUCTIVITY mode, and 2 bitplanes.

WaitBlit currently tries to avoid this Agnus problem by testing
the BUSY bit multiple times to make sure the blitter has really
written its final word of destination data.

 SEE ALSO
OwnBlitter() DisownBlitter() hardware/blit.h
```

1.160 graphics.library/WaitBOVP

```
 NAME
WaitBOVP -- Wait till vertical beam reached bottom of
      this viewport.
```

```
 SYNOPSIS
WaitBOVP( vp )
    a0

void WaitBOVP( struct ViewPort * );

 FUNCTION
Returns when the vertical beam has reached the bottom of this viewport

 INPUTS
vp - pointer to ViewPort structure

 RESULT
This function will return sometime after the beam gets beyond
the bottom of the viewport.  Depending on the multitasking load
of the system, the actual beam position may be different than
what would be expected in a lightly loaded system.

 BUGS
Horrors! This function currently busy waits waiting for the
beam to get to the right place.  It should use the copper
interrupt to trigger and send signals like WaitTOF does.

 SEE ALSO
WaitTOF() VBeamPos()
```

1.161 graphics.library/WaitTOF

```
 NAME
    WaitTOF -- Wait for the top of the next video frame.

 SYNOPSIS
    WaitTOF()

void WaitTOF( void );

 FUNCTION
    Wait  for vertical blank to occur and all vertical blank
    interrupt routines to complete before returning to caller.

 INPUTS
    none

 RESULT
Places this task on the TOF wait queue. When the vertical blank
interrupt comes around, the interrupt service routine will fire off
signals to all the tasks doing WaitTOF. The highest priority task
ready will get to run then.

 BUGS

 SEE ALSO
exec.library/Wait() exec.library/Signal()
```

1.162 graphics.library/WriteChunkyPixels

```
 NAME
WriteChunkyPixels -- write the pen number value of a rectangular array
of pixels starting at a specified x,y location and continuing
through to another x,y location within a certain RastPort. (V40)

 SYNOPSIS
WriteChunkyPixels(rp,xstart,ystart,xstop,ystop,array,bytesperrow)
                  A0 D0     D1     D2    D3    A2     D4

VOID WriteChunkyPixels(struct  RastPort *, LONG, LONG,
     LONG, LONG, UBYTE *, LONG);

 FUNCTION
For each pixel in a rectangular region, decode the pen number selector
from a linear array of pen numbers into the bit-planes used to describe
a particular rastport.

 INPUTS
rp     -  pointer to a RastPort structure
(xstart,ystart) -  starting point in the RastPort
(xstop,ystop)   -  stopping point in the RastPort
array  - pointer to an array of UBYTEs from which to fetch the
         pixel data.
bytesperrow - The number of bytes per row in the source array.
  This should be at least as large as the number of pixels
  being written per line.

 RESULT

 NOTE
xstop must be >= xstart
ystop must be >= ystart
The source array can be in fast RAM.

 ===chunky-to-planar conversion HW:

 GfxBase->ChunkyToPlanarPtr is either NULL, or a pointer to a HW
 register used to aid in the process of converting 8-bit chunky
 pixel data into the bit-plane format used by the Amiga custom
 display chips. If NULL, then such hardware is not present.

 If an expansion device provides hardware which operates compatibly,
 than it can install the HW address into this pointer at boot time,
 and the system will use it.

 This pointer may be used for direct access to the chunky-to-planar
 conversion HW, if more is desired than the straight chunky-pixel
 copy that is performed by WriteChunkyPixels().

 If using the hardware directly, it should only be accessed when
 the task using it has control of the blitter (via OwnBlitter()),
 since this is the locking used to arbitrate usage of this device.

 The hardware may be viewed as a device which accepts 32 8-bit
```

```
 chunky pixels and outputs 8 longwords of bitplane data.

 For proper operation, exactly 8 longwords (containing 32 pixels)
 of chunky data should be written to *(GfxBase->ChunkyToPlanarPtr).
 After the data is written, bitplane data (starting with plane 0)
 can be read back a longword at a time. There is no need to read
 back all 8 longwords if the high-order bitplanes are not needed.

 Since WriteChunkyPixels is not (currently) particularly fast on
 systems without the chunky-to-planar hardware, time critical
 applications (games, etc) may want to use their own custom conversion
 routine if GfxBase->ChunkyToPlanarPtr is NULL, and call
 WriteChunkyPixels() otherwise.

 This pointer is only present in GfxBase in versions of graphics.library
 >= 40, so this should be checked before the pointer is read.

 BUGS
Not very fast on systems without chunky-to-planar conversion
hardware.

 SEE ALSO
WritePixel()  graphics/rastport.h
```

1.163 graphics.library/WritePixel

```
 NAME
     WritePixel -- Change the pen num of one specific pixel in a
                   specified RastPort.

 SYNOPSIS
     error = WritePixel(  rp, x,  y)
       d0                 a1 D0  D1

LONG WritePixel( struct RastPort *, SHORT, SHORT );

 FUNCTION
     Changes the pen number of the selected pixel in the specified
     RastPort to that currently specified by PenA, the primary
     drawing pen. Obeys minterms in RastPort.

 INPUTS
     rp - a pointer to the RastPort structure
     (x,y) - point within the RastPort at which the selected
         pixel is located.

 RESULT
     error = 0 if pixel succesfully changed
      = -1 if (x,y) is outside the RastPort

 BUGS

 SEE ALSO
     ReadPixel() graphics/rastport.h
```

1.164 graphics.library/WritePixelArray8

```
 NAME
WritePixelArray8 -- write the pen number value of a rectangular array
of pixels starting at a specified x,y location and continuing
through to another x,y location within a certain RastPort. (V36)

 SYNOPSIS
count = WritePixelArray8(rp,xstart,ystart,xstop,ystop,array,temprp)
D0                       A0 D0:16  D1:16  D2:16 D3:16  A2   A1

LONG WritePixelArray8(struct  RastPort *, UWORD, UWORD,
     UWORD, UWORD, UBYTE *, struct  RastPort *);

 FUNCTION
For each pixel in a rectangular region, decode the pen number selector
from a linear array of pen numbers into the bit-planes used to describe
a particular rastport.

 INPUTS
rp     -  pointer to a RastPort structure
(xstart,ystart) -  starting point in the RastPort
(xstop,ystop)   -  stopping point in the RastPort
array  - pointer to an array of UBYTEs from which to fetch the
         pixel data. Allocate at least
         ((((width+15)>>4)<<4)*(ystop-ystart+1)) bytes.
temprp - temporary rastport (copy of rp with Layer set == NULL,
         temporary memory allocated for
         temprp->BitMap with Rows set == 1,
         temprp->BytesPerRow == (((width+15)>>4)<<1),
         and temporary memory allocated for
         temprp->BitMap->Planes[])

 RESULT
count will be set to the number of pixels plotted.

 NOTE
xstop must be >= xstart
ystop must be >= ystart

 BUGS

 SEE ALSO
WritePixel()  graphics/rastport.h
```

1.165 graphics.library/WritePixelLine8

```
 NAME
WritePixelLine8 -- write the pen number value of a horizontal line
of pixels starting at a specified x,y location and continuing
right for count pixels. (V36)

 SYNOPSIS
count = WritePixelLine8(rp,xstart,ystart,width,array,temprp)
```

```
D0                      A0 D0:16  D1:16  D2    A2    A1

LONG WritePixelLine8(struct RastPort *, UWORD, UWORD,
     UWORD, UBYTE *, struct RastPort *);

 FUNCTION
For each pixel in a horizontal region, decode the pen number selector
from a linear array of pen numbers into the bit-planes used to describe
a particular rastport.

 INPUTS
rp    -  pointer to a RastPort structure
(x,y) - a point in the RastPort
width - count of horizontal pixels to write
array - pointer to an array of UBYTEs from which to fetch the pixel data
        allocate at least (((width+15)>>4)<<4) bytes.
temprp - temporary rastport (copy of rp with Layer set == NULL,
         temporary memory allocated for
         temprp->BitMap with Rows set == 1,
         temprp->BytesPerRow == (((width+15)>>4)<<1),
         and temporary memory allocated for
         temprp->BitMap->Planes[])

 RESULT
Count will be set to the number of pixels plotted

 NOTE
width must be non negative

 BUGS

 SEE ALSO
WritePixel()  graphics/rastport.h
```

1.166 graphics.library/XorRectRegion

```
 NAME
     XorRectRegion -- Perform 2d XOR operation of rectangle
                      with region, leaving result in region

 SYNOPSIS
     status = XorRectRegion(region,rectangle)
       d0                    a0      a1

BOOL XorRectRegion( struct Region *, struct Rectangle * );

 FUNCTION
Add portions of rectangle to region if they are not in
the region.
Remove portions of rectangle from region if they are
in the region.

 INPUTS
     region - pointer to Region structure
     rectangle - pointer to Rectangle structure
```

```
 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS

 SEE ALSO
OrRegionRegion() AndRegionRegion() graphics/regions.h
```

1.167 graphics.library/XorRegionRegion

```
 NAME
     XorRegionRegion -- Perform 2d XOR operation of one region
                     with second region, leaving result in second region

 SYNOPSIS
     status = XorRegionRegion(region1,region2)
       d0                         a0      a1

BOOL XorRegionRegion( struct Region *, struct Region * );

 FUNCTION
Join the regions together. If any part of region1 overlaps
region2 then remove that from the new region.

 INPUTS
     region1      = pointer to Region structure
     region2      = pointer to Region structure

 RESULTS
status - return TRUE if successful operation
   return FALSE if ran out of memory

 BUGS
```