

cybergraphics

COLLABORATORS

	TITLE : cybergraphics		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		December 2, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	cybergraphics	1
1.1	cybergraphics.doc	1
1.2	cybergraphics.library/--background--	1
1.3	cybergraphics.lib/AllocCModelListTagList	2
1.4	cybergraphics.library/BestCModelIDTagList	3
1.5	cybergraphics.library/CModeRequestTagList	4
1.6	cybergraphics.library/DoCDrawMethodTagList	5
1.7	cybergraphics.lib/FreeCModelList	7
1.8	cybergraphics.library/GetCyberMapAttr	7
1.9	cybergraphics.library/GetCyberIDAttr	8
1.10	cybergraphics.library/IsCyberModelID	8
1.11	cybergraphics.library/FillPixelArray	9
1.12	cybergraphics.library/InvertPixelArray	10
1.13	cybergraphics.library/MovePixelArray	10
1.14	cybergraphics.library/ReadPixelArray	11
1.15	cybergraphics.library/ReadRGBPixel	12
1.16	cybergraphics.library/ScalePixelArray	12
1.17	cybergraphics.library/SwapPixelArray	14
1.18	cybergraphics.library/WritePixelArray	15
1.19	cybergraphics.library/WriteRGBPixel	16

Chapter 1

cybergraphics

1.1 cybergraphics.doc

```
--background--
AllocCModeListTagList ()
BestCModeIDTagList ()
CModeRequestTagList ()
DoCDDrawMethodTagList ()
FillPixelArray ()
FreeCModeList ()
GetCyberMapAttr ()
GetCyberIDAttr ()
IsCyberModeID ()
MovePixelArray ()
ReadPixelArray ()
ReadRGBPixel ()
ScalePixelArray ()
SwapPixelArray ()
WritePixelArray ()
WriteRGBPixel ()
```

1.2 cybergraphics.library/--background--

PURPOSE

cybergraphics.library is meant to be a gfx board extension for the native graphics.library. Because C= never developed a rtg compliant graphics.library (up to os3.1 there are no possibilities to add custom boards into the display database in a system friendly fashion), it patches some of the original graphics functions in order to integrate custom graphics boards into the system.

It also features some new functions which are not available in the standard graphics.library. Currently graphics.library v39 and v40 is supported.

Additionally it is possible to open screens deeper than 256 colours which are completely intuition compatible. Nearly every function may be used on such screen except some special blits which rely on planar bitmap graphics. Also this screens are 8bit backwards compatible. Nearly every standard 8bit application will work on this 15/16/24 bit

screen. But it only can use the additional features, if it knows of cybergraphics, of course.

Maybe in a future release of graphics, cybergraphics will be obsolete. But until then, cybergraphics.library is the only way to add truecolour features to your application under an intuition environment with minimum effort.

So, cybergraphics is not meant to be a replacement for the original graphics/intuition system (just like EGS is for example), e.g. there is no ECS/AGA native chipset cybergraphics driver, because all the functionality of the original chipset is already given in the original graphics.library.

Because of the different goal cybergraphics spots in comparison to EGS for example, it still may lack some features other systems have already but it is perhaps the most system friendly graphics extension the amiga computer family has seen so far.

OVERVIEW

This part only describes how to use the extended features of cybergraphics. Anything else about programming graphics can be found in the original graphics.library documentation.

1.3 cybergraphics.lib/AllocCModeListTagList

NAME

AllocCModeListTagList -- get an exec list with requested modes.

SYNOPSIS

```
result = CModeRequestTagList (ModeListTags);
D0                                     A1
```

```
APTR AllocCModeListTagList (struct TagItem *);
```

FUNCTION

Allocates a list structure which contains all requested modes (All nodes are of type CyberModeNode). See defines for more information about the structure of this nodes.

INPUTS

tags - pointer to an optional tag list which may be used to control the number of returned modes.

TAGS

Tags available are:

CYBRMREQ_MinWidth (ULONG) - The minimum display width to let the user choose. Default is 320.

CYBRMREQ_MaxWidth (ULONG) - The maximum display width to let the user choose. Default is 1600.

CYBRMREQ_MinHeight (ULONG) - The minimum display height to let the user

choose. Default is 240.

CYBRMREQ_MaxHeight (ULONG) - The maximum display height to let the user choose. Default is 1200.

CYBRMREQ_MinDepth (UWORD) - The minimum display depth to let the user choose. Default is 8.

CYBRMREQ_MaxDepth (UWORD) - The maximum display depth to let the user choose. Default is 32.

CYBRMREQ_CModelArray (UWORD *) - Array of color models which should be available for screenmode selection. Currently supported colormodels are:

```
PIXFMT_LUT8
PIXFMT_RGB15
PIXFMT_BGR15
PIXFMT_RGB15PC
PIXFMT_BGR15PC
PIXFMT_RGB16
PIXFMT_BGR16
PIXFMT_RGB16PC
PIXFMT_BGR16PC
PIXFMT_RGB24
PIXFMT_BGR24
PIXFMT_ARGB32
PIXFMT_BGRA32
```

default is all colormodels available, nothing filtered

RESULT

result - 0 if no modes are available, a pointer to a exec list if there are modes which fit your needs.

1.4 cybergraphics.library/BestCModelIDTagList

NAME

BestCModelIDTagList -- calculate the best ModeID with given parameters

SYNOPSIS

```
ID = BestCModelIDTagList(TagItems)
d0                                a0
```

```
ID = BestCModelIDTags(Tag1, ...)
```

FUNCTION

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

INPUTS

TagItems - A pointer to an array of TagItems.

TAGS

CYBRBIDTG_Depth (ULONG) - depth the returned ModeID must support
Default is 8

CYBRBIDTG_NominalWidth (UWORD),
CYBRBIDTAG_NominalHeight (UWORD) - desired width and height the ModeID
should have

CYBRBIDTG_MonitorID (ULONG) - if multiple graphics boards are
installed in the system, you can choose
the desired one with this tag

Currently supported boards are:

CYBERVISION
PICCOLO
PICASSO
SPECTRUM
DOMINO
RETINAZ3
PICCOSD64

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

1.5 cybergraphics.library/CModeRequestTagList

NAME
CModeRequestTagList -- get screenmode from user using a requester.

SYNOPSIS
result = CModeRequestTagList(requester, tags);
D0 A0 A1

LONG CModeRequestTagList(APTR, struct TagItem *);

FUNCTION
Prompts the user for input by showing all available cybergraphics
screenmodes in a requester.
If the user cancels or the system aborts the request, FALSE is returned,
otherwise the displaymode id of the selected screenmode.

INPUTS
requester - not used currently. you have to set it to 0 !
tags - pointer to an optional tag list which may be used to
control features of the requester.

TAGS

Tags used for the screen mode requester

CYBRMREQ_Screen (struct Screen *) - Screen on which to open the requester.
default locale will be used.

CYBRMREQ_WinTitle (STRPTR) - Title to use for the requesting window.

CYBRMREQ_OKText (STRPTR) - Label of the positive gadget in the requester. English default is "OK".

CYBRMREQ_CancelText (STRPTR) - Label of the negative gadget in the requester. English default is "Cancel".

CYBRMREQ_MinWidth (ULONG) - The minimum display width to let the user choose. Default is 320.

CYBRMREQ_MaxWidth (ULONG) - The maximum display width to let the user choose. Default is 1600.

CYBRMREQ_MinHeight (ULONG) - The minimum display height to let the user choose. Default is 240.

CYBRMREQ_MaxHeight (ULONG) - The maximum display height to let the user choose. Default is 1200.

CYBRMREQ_MinDepth (UWORD) - The minimum display depth to let the user choose. Default is 8.

CYBRMREQ_MaxDepth (UWORD) - The maximum display depth to let the user choose. Default is 32.

CYBRMREQ_CModelArray (UWORD *) - Array of color models which should be available for screenmode selection. Currently supported colormodels are:

PIXFMT_LUT8
PIXFMT_RGB15
PIXFMT_BGR15
PIXFMT_RGB15PC
PIXFMT_BGR15PC
PIXFMT_RGB16
PIXFMT_BGR16
PIXFMT_RGB16PC
PIXFMT_BGR16PC
PIXFMT_RGB24
PIXFMT_BGR24
PIXFMT_ARGB32
PIXFMT_BGRA32

default is all colormodels available, nothing filtered

RESULT

result - 0 if the user cancelled the requester or if something prevented the requester from opening. If != 0 the displaymode id of the selected screenmode is returned.

BUGS

The requester structure is not supported in (v40)

1.6 cybergraphics.library/DoCDrawMethodTagList

NAME

DoCDrawMethodTagList - Do the given hook for the supplied rastport

SYNOPSIS

```
DoCDrawMethodTagList(hook, rport, taglist)
                    a0    a1    a2
```

```
void DoCDrawMethodTagList(struct Hook *, struct RastPort *,
                          struct TagItem *)
```

FUNCTION

This function will call the given hook for the given rastport. Is is mainly used to do direct bitmap modifications in a locked graphics environment. You have to support ALL known color models, so only use this call if you really need it !!

INPUTS

hook - pointer to callback hook which will be called
 with object == (struct RastPort *)
 and message == [(APTR) memptr,
 (ULONG) offsetx, (ULONG) offsey,
 (ULONG) xsize, (ULONG) ysize,
 (ULONG) bytesperrow, (UWORD) bytesperpix,
 (UWORD) colormodel]

Where colormodel is one of the following:

```
PIXFMT_LUT8
PIXFMT_RGB15
PIXFMT_BGR15
PIXFMT_RGB15PC
PIXFMT_BGR15PC
PIXFMT_RGB16
PIXFMT_BGR16
PIXFMT_RGB16PC
PIXFMT_BGR16PC
PIXFMT_RGB24
PIXFMT_BGR24
PIXFMT_ARGB32
PIXFMT_BGRA32
```

rport- A pointer to a cybergraphics RastPort

tags - optional taglist, currently not used. Set it to NULL

NOTES

Use this call only if you want high speed. Remember that you have to handle all color models ! Do not use ANY os functions in your hook. They would cause unpredictable results.

BUGS

1.7 cybergraphics.lib/FreeCModelist

NAME

FreeCModelist -- frees a previously allocated Modelist
(AllocCModelistTagList)

SYNOPSIS

```
FreeCModelist (ModelistTags);
               A0
```

```
void FreeCModelist (struct List *);
```

FUNCTION

Frees all data which was previously allocated by AllocCModelistTagList

INPUTS

ModelistTags - a list structure which contains all the mode data.

RESULT

none

1.8 cybergraphics.library/GetCyberMapAttr

NAME

GetCyberMapAttr -- Returns information about a cybergraphics bitmap

SYNOPSIS

```
value=GetCyberMapAttr(bitmap,attribute_number);
d0                                a0      d1
```

```
ULONG GetCyberMapAttr(struct BitMap *,ULONG);
```

FUNCTION

Determines information about a extended cybergraphics bitmap. This function should be used instead of making any assumptions about fields in the bitmap. This will provide future compatibility.

INPUTS

bitmap - pointer to a cybergraphics bitmap structure

attribute_number - A number telling cybergraphics which attribute of the bitmap should be returned:

CYBRMATTR_XMOD returns BytesPerRow of the supplied bitmap

CYBRMATTR_BPPIX returns number of bytes per pixel

CYBRMATTR_PIXFMT return the pixel format of the bitmap

CYBRMATTR_WIDTH return width of the bitmap in pixels

CYBRMATTR_HEIGHT return the height in lines

CYBRMATTR_DEPTH returns bits per pixel

BUGS

NOTES

Unknown attributes are reserved for future use, and return (-1L).

You should know what you are doing if you call this function !
Don't apply it on a non cybergraphics bitmap !

1.9 cybergraphics.library/GetCyberIDAttr

NAME

GetCyberIDAttr -- Returns information about a cybergraphics id

SYNOPSIS

```
value=GetCyberIDAttr(CyberIDAttr,CyberDisplayModeID);
d0                      d0                      d1
```

```
ULONG GetCyberIDAttr(ULONG,ULONG);
```

FUNCTION

Determines information about a specified displaymode id.

INPUTS

CyberDisplayModeID - cybergraphics mode id

attribute_number - A number telling cybergraphics which attribute
of the displaymode id should be returned:

CYBRIDATTR_PIXFMT return the pixel format of the supplied
screenmode id

CYBRIDATTR_WIDTH returns visible width in pixels
CYBRIDATTR_HEIGHT returns visible height in lines
CYBRIDATTR_DEPTH returns bits per pixel
CYBRIDATTR_BPPIX should return BytesPerPixel

BUGS

NOTES

Unknown attributes are reserved for future use, and return (-1L).

You should know what you are doing if you call this function !
Don't apply it on a non cybergraphics displaymode !

1.10 cybergraphics.library/IsCyberModeID

NAME

IsCyberModeID -- returns whether supplied ModeID is a cybergraphics id

```

SYNOPSIS
result = IsCyberModeID(modeID)
D0                                     D0

BOOL IsCyberModeID(ULONG)

FUNCTION
Returns whether the supplied ModeID is a cybergraphics.library mode
identifier.

INPUTS
modeID -- a 32 bit display identifier.

RESULT
result - Flag to indicate if modeID is a cybergraphics ID

```

1.11 cybergraphics.library/FillPixelArray

```

NAME
    FillPixelArray -- fill a rectangular area with the supplied ARGB value
    starting at a specified x,y location and continuing through to another
    x,y location within a certain RastPort

```

```

SYNOPSIS
count = FillPixelArray(RastPort, DestX, DestY, SizeX, SizeY, ARGB)
D0      A1      D0:16 D1:16 D2:16 D3:16 D4:32

```

```

LONG FillPixelArray(struct RastPort *,UWORD,UWORD,UWORD,UWORD,ULONG)

```

```

FUNCTION
For each pixel in a rectangular region, write the supplied color value
into the bitmap used to describe a particular rastport.

```

```

INPUTS
RastPort - pointer to a RastPort structure
(DestX, DestY) - starting point in the RastPort
(SizeX, SizeY) - size of the rectangle that should be transferred
    ARGB - the desired color in AARRGGBB format. Every component
    allocates 8 bits of the returned longword. The coding is as
    follows:

```

```

        AA - 8-bit alpha channel component
        (set it to 00 if you do not use it !)
        RR - 8-bit red component of the pixel
        GG - 8-bit green component
        BB - 8-bit blue component

```

```

RESULT
count will be set to the number of pixels plotted

```

```

NOTES
This function should only be used on screens depths > 8 bits.

```

BUGS

1.12 cybergraphics.library/InvertPixelArray

NAME

InvertPixelArray -- invert a rectangular area tarting at a specified x,y location and continuing through to another x,y location within a certain RastPort

SYNOPSIS

count = InvertPixelArray(RastPort, DestX, DestY, SizeX, SizeY)

D0 A1 D0:16 D1:16 D2:16 D3:16

LONG InvertPixelArray(struct RastPort *,UWORD,UWORD,UWORD,UWORD)

FUNCTION

Invert each pixel in a rectangular region.

INPUTS

RastPort - pointer to a RastPort structure

(DestX, DestY) - starting point in the RastPort

(SizeX, SizeY) - size of the rectangle that should be transfered

RESULT

count will be set to the number of pixels plotted

NOTES

This function should only be used on screens depths > 8 bits.

BUGS

1.13 cybergraphics.library/MovePixelArray

NAME

MovePixelArray -- move the color values of a rectangular area of pixels starting at a specified x,y location and continuing through to another x,y location within a certain RastPort

SYNOPSIS

count = MovePixelArray(SrcX, SrcY, RastPort, SrcX , SrcY ,SizeX, SizeY)

D0 D0:16 D1:16 A1 D2:16 D3:16 D4:16 D5:16

LONG MovePixelArray(UWORD,UWORD,struct RastPort *,UWORD,UWORD,UWORD,UWORD)

FUNCTION

For each pixel in a rectangular region, move the pixel value from a specified source to a specified destination

INPUTS

(SrcX, SrcY) - starting point in the destination rectangle

RastPort - pointer to a RastPort structure
 (DestX, DestY) - starting point in the destination rectangle
 (SizeX, SizeY) - size of the rectangle that should be transferred

RESULT
 count will be set to the number of pixels moved

NOTES
 This function should only be used on screens depths > 8 bits.
 The blitter can be used to move the data if the bitmap is in display memory. This is way you should use this call.

BUGS

1.14 cybergraphics.library/ReadPixelFormat

NAME

ReadPixelFormat -- Read the color values 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

SYNOPSIS

```
count = ReadPixelFormat(destRect, DestX, DestY, DestMod, RastPort, SrcX ,
D0      A0      D0:16 D1:16 D2:16      A1      D3:16
      SrcY , SizeX, SizeY, DestFormat)
      D4:16 D5:16 D6:16      D7
```

```
LONG ReadPixelFormat(APTR, UWORD, UWORD, UWORD, struct RastPort *, UWORD,
      UWORD, UWORD, UWORD, UBYTE)
```

FUNCTION

For each pixel in a rectangular region, write the color value to a linear array of color values from the bitmap used to describe a particular rastport.

INPUTS

destRect - pointer to an array of pixels where to write the pixel data to. The pixel format is specified in DestFormat
 (DestX, DestY) - starting point in the destination rectangle
 DestMod - The number of bytes per row in the destination rectangle.
 RastPort - pointer to a RastPort structure
 (SrcX, SrcY) - starting point in the RastPort
 (SizeX, SizeY) - size of the rectangle that should be transferred
 DestFormat - pixel format in the destination rectangle
 Currently supported formats are:

RECTFMT_RGB 3 bytes per pixel, one byte red, one blue
 and one byte green component

RECTFMT_RGBA 4 bytes per pixel, one byte red, one blue,
 one byte green component and the last
 byte is alpha channel information which
 is 0 if the board does not support alpha
 channel

RECTFMT_ARGB 4 bytes per pixel, one byte red, one blue,
 one byte green component and the first
 byte is alpha channel information. If the
 board does not support alpha channel a
 0 is returned for alpha channel information

RESULT
 count will be set to the number of pixels read

NOTES
 This function should only be used on screens depths > 8 bits.

BUGS

1.15 cybergraphics.library/ReadRGBPixel

NAME

ReadRGBPixel -- Reads a pixel from a specified location

SYNOPSIS

```
color = ReadRGBPixel(RastPort,x ,y )
D0                      A0      D0 D1
```

```
ULONG ReadRGBPixel(struct RastPort *,UWORD,UWORD);
```

FUNCTION

Read the Alpha,Red,Green & Blue 8-bit color value of the pixel at a
 specified x,y location within a certain RastPort

INPUTS

rp - pointer to a RastPort structure
 x,y - the coordinates of the pixel

RESULT

color - the desired color in AARRGGBB format. Every component
 allocates 8 bits of the returned longword. The coding is as
 follows:

```
AA - 8-bit alpha channel component
(board which do not have an alpha channel return 00)
RR - 8-bit red component of the pixel
GG - 8-bit green component
BB - 8-bit blue component
```

NOTES

This function should only be used on screens depths > 8 bits. Use
 ReadPixel() on 8 bit screens !

BUGS

1.16 cybergraphics.library/ScalePixelArray

NAME

ScalePixelFormatArray -- Scale the colors values 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 (V41)

SYNOPSIS

```
count = ScalePixelFormatArray(srcRect,SrcW,SrcH ,SrcMod,RastPort,DestX,
D0      A0      D0:16 D1:16 D2:16      A1      D3:16
      DestY,DestW,DestH,SrcFormat)
      D4:16 D5:16 D6:16      D7
```

```
LONG ScalePixelFormatArray(APTR,UWORD,UWORD,UWORD,struct RastPort *,UWORD,
      UWORD,UWORD,UWORD,UBYTE)
```

FUNCTION

For each pixel in a rectangular region, scale the color values from a linear array of color values into the bitmap used to describe a particular rastport.

INPUTS

srcRect - pointer to an array of pixels from which to fetch the pixel data. The pixel format is specified in SrcFormat
 (SrcW,SrcH) - Width and Height of the source rectangle
 SrcMod - The number of bytes per row in the source rectangle.
 RastPort - pointer to a RastPort structure
 (DestX,DestY) - starting point in the RastPort
 (DestW,DestH) - size of the destination area
 SrcFormat - pixel format in the source rectangle
 Currently supported formats are:

```
RECTFMT_RGB 3 bytes per pixel, one byte red, one blue
              and one byte green component
```

```
RECTFMT_RGBA 4 bytes per pixel, one byte red, one blue,
              one byte green component and the last
              byte is alpha channel information. If you
              do not use alpha channel set this byte to
0 !!!
```

```
RECTFMT_ARGB 4 bytes per pixel, one byte red, one blue,
              one byte green component and the first
              byte is alpha channel information. If you
              do not use alpha channel set this byte to
0 !!!
```

RESULT

count will be set to the number of pixels plotted

NOTES

This function should only be used on screens depths > 8 bits.

BUGS

1.17 cybergraphics.library/SwapPixelFormat

NAME

SwapPixelFormat -- Swap the color values 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

SYNOPSIS

```
count = SwapPixelFormat(MemRect,MemX ,MemY,MemMod,RastPort, RastX ,
D0      A0      D0:16 D1:16 D2:16      A1      D3:16
      RastY ,SizeX,SizeY,MemFormat)
      D4:16 D5:16 D6:16      D7
```

```
LONG SwapPixelFormat(APTR,UWORD,UWORD,UWORD,struct RastPort *,UWORD,
      UWORD,UWORD,UWORD,UBYTE)
```

FUNCTION

For each pixel in a rectangular region, swap the color values of a linear array of color values and a bitmap used to describe a particular rastport.

INPUTS

MemRect - pointer to an array of pixels where to write the pixel data of the bitmapto. The pixel format is specified in

MemFormat

(MemX,MemY) - starting point in the memory rectangle

MemMod - The number of bytes per row in the memory rectangle.

RastPort - pointer to a RastPort structure

(RastX,RastY) - starting point in the RastPort rectangle

(SizeX,SizeY) - size of the rectangle that should be transfered

MemFormat - pixel format in the memory rectangle

Currently supported formats are:

RECTFMT_RGB 3 bytes per pixel, one byte red, one blue and one byte green component

RECTFMT_RGBA 4 bytes per pixel, one byte red, one blue, one byte green component and the last byte is alpha channel information which is 0 if the board does not support alpha channel

RECTFMT_ARGB 4 bytes per pixel, one byte red, one blue, one byte green component and the first byte is alpha channel information. If the board does not support alpha channel a 0 is returned for alpha channel information

RESULT

count will be set to the number of pixels swapped

NOTES

This function should only be used on screens depths > 8 bits.

DON'T USE THIS FUNCTION WITH V40 OF CYBERGRAPHICS

BUGS

THIS FUNCTION IS BROKEN IN V40 ... DON'T USE IT

1.18 cybergraphics.library/WritePixelFormat

NAME

WritePixelFormat -- write the color 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

SYNOPSIS

```
count = WritePixelFormat(srcRect,SrcX ,SrcY ,SrcMod,RastPort,DestX,
D0      A0   D0:16 D1:16 D2:16      A1   D3:16
      DestY,SizeX,SizeY,SrcFormat)
      D4:16 D5:16 D6:16      D7
```

```
LONG WritePixelFormat(APTR,UWORD,UWORD,UWORD,struct RastPort *,UWORD,
      UWORD,UWORD,UWORD,UBYTE)
```

FUNCTION

For each pixel in a rectangular region, write the color value from a linear array of color values into the bitmap used to describe a particular rastport.

INPUTS

srcRect - pointer to an array of pixels from which to fetch the pixel data. The pixel format is specified in SrcFormat
 (SrcX,SrcY) - starting point in the source rectangle
 SrcMod - The number of bytes per row in the source rectangle.
 RastPort - pointer to a RastPort structure
 (DestX,DestY) - starting point in the RastPort
 (SizeX,SizeY) - size of the rectangle that should be transferred
 SrcFormat - pixel format in the source rectangle
 Currently supported formats are:

```
RECTFMT_RGB  3 bytes per pixel, one byte red, one blue
              and one byte green component

RECTFMT_RGBA 4 bytes per pixel, one byte red, one blue,
              one byte green component and the last
              byte is alpha channel information. If you
              do not use alpha channel set this byte to
0 !!!

RECTFMT_ARGB 4 bytes per pixel, one byte red, one blue,
              one byte green component and the first
              byte is alpha channel information. If you
              do not use alpha channel set this byte to
0 !!!
```

RESULT

count will be set to the number of pixels plotted

NOTES

This function should only be used on screens depths > 8 bits. Use WritePixelFormat8() on 8 bit screens !

BUGS

1.19 cybergraphics.library/WriteRGBPixel

NAME

WriteRGBPixel -- Writes a pixel to a specified location

SYNOPSIS

```
error = WriteRGBPixel(RastPort,x ,y ,color)
```

```
D0                      A0      D0 D1 d2
```

```
ULONG WriteRGBPixel(struct RastPort *,UWORD,UWORD,ULONG);
```

FUNCTION

Write the Alpha,Red,Green & Blue 8-bit color value of the given color to a specified x,y location within a certain RastPort

INPUTS

rp - pointer to a RastPort structure
 x,y - the coordinates of the pixel
 color - the desired color in AARRGGBB format. Every component allocates 8 bits of the returned longword. The coding is as follows:

```
    AA - 8-bit alpha channel component
(set it to 00 if you dont want to use it !)
    RR - 8-bit red component of the pixel
    GG - 8-bit green component
    BB - 8-bit blue component
```

RESULT

error = 0 if pixel succesfully changed
 = -1 if (x,y) is outside the rastport

NOTES

This function should only be used on screens depths > 8 bits. Use WritePixel() on 8 bit screens !

BUGS