opal

COLLABORATORS

	TITLE : opal		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 16, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 opal **1**

1.1 opal.doc 1

1.2 opal.library/ActiveScreen24 3

1.3 opal.library/AmigaPriority 4

1.4 opal.library/AppendCopper24 5

1.5 opal.library/AutoSync24 6

1.6 opal.library/BitPlanetoOV 6

1.7 opal.library/ClearDisplayBottom24 7

1.8 opal.library/ClearPFStencil24 7

1.9 opal.library/ClearPRStencil24 8

1.10 opal.library/ClearQuick24 9

1.11 opal.library/ClearScreen24 9

1.12 opal.library/CloseScreen24 10

1.13 opal.library/Config24 11

1.14 opal.library/CreateScreen24 12

1.15 opal.library/DisablePRStencil24 13

1.16 opal.library/DisplayFrame24 13

1.17 opal.library/DisplayThumbnail24 14

1.18 opal.library/DrawEllipse24 15

1.19 opal.library/DrawLine24 16

1.20 opal.library/DualDisplay24 17

1.21 opal.library/DualPlayField2 17

1.22 opal.library/EnablePRStencil24 18

1.23 opal.library/FadeIn24 19

1.24 opal.library/FadeOut24 19

1.25 opal.library/FreeScreen24 20

1.26 opal.library/FreezeFrame24 20

1.27 opal.library/ILBMtoOV 21

1.28 opal.library/LatchDisplay24 22

1.29 opal.library/LoadImage24 22

1.30 opal.library/LowMemUpdate24 24
1.31 opal.library/LowMem2Update24 25
1.32 opal.library/LowMemRGB24 26
1.33 opal.library/OpenScreen24 27
1.34 opal.library/OVPriority 29
1.35 opal.library/OVtoBitPlane 29
1.36 opal.library/OVtoILBM 30
1.37 opal.library/OVtoRGB 31
1.38 opal.library/PaletteMap24 32
1.39 opal.library/ReadPFPixel24 32
1.40 opal.library/ReadPixel24 33
1.41 opal.library/ReadPRPixel24 34
1.42 opal.library/RectFill24 35
1.43 opal.library/Refresh24 35
1.44 opal.library/RegWait24 36
1.45 opal.library/RGBtoOV 37
1.46 opal.library/SaveIFF24 38
1.47 opal.library/SaveJPEG24 39
1.48 opal.library/Scroll24 40
1.49 opal.library/SetControlBit24 41
1.50 opal.library/SetCoPro24 42
1.51 opal.library/SetDisplayBottom24 42
1.52 opal.library/SetHires24 43
1.53 opal.library/SetLoadAddress24 44
1.54 opal.library/SetLores24 44
1.55 opal.library/SetPFStencil24 45
1.56 opal.library/SetPRStencil24 46
1.57 opal.library/SetRGB24 46
1.58 opal.library/SetScreen24 47
1.59 opal.library/SetSprite24 47
1.60 opal.library/SingleDisplay24 49
1.61 opal.library/SinglePlayField24 49
1.62 opal.library/StopUpdate24 50
1.63 opal.library/UpdateAll24 50
1.64 opal.library/UpdateCoPro24 51
1.65 opal.library/UpdateDelay24 52
1.66 opal.library/UpdatePalette24 53
1.67 opal.library/UpdatePFStencil24 53
1.68 opal.library/UpdateRegs24() 54

1.69 opal.library/WriteFrame24 55
1.70 opal.library/WritePFPixel24 56
1.71 opal.library/WritePixel24 56
1.72 opal.library/WritePRPixel24 57
1.73 opal.library/WriteThumbnail24 58
1.74 opalreq.library/OpalRequester 58

Chapter 1

opal

1.1 opal.doc

ActiveScreen24()

AmigaPriority()

AppendCopper24()

AutoSync24()

BitPlanetoOV()

ClearDisplayBottom24()

ClearPFStencil24()

ClearPRStencil24()

ClearQuick24()

ClearScreen24()

CloseScreen24()

Config24()

CreateScreen24()

DisablePRStencil24()

DisplayFrame24()

DisplayThumbnail24()

DrawEllipse24()

DrawLine24()

DualDisplay24()

DualPlayField2()

EnablePRStencil24()

FadeIn24()

FadeOut24()

FreeScreen24()

FreezeFrame24()

ILBMtoOV()

LatchDisplay24()

LoadImage24()

LowMemUpdate24()

LowMem2Update24()

LowMemRGB24()

OpenScreen24()

OVPriority()

OVtoBitPlane()

OVtoILBM()

OVtoRGB()

PaletteMap24()

ReadPFPixel24()

ReadPixel24()

ReadPRPixel24()

RectFill24()

Refresh24()

RegWait24()

RGBtoOV()

SaveIFF24()

SaveJPEG24()

Scroll24()

SetControlBit24()

SetCoPro24()

SetDisplayBottom24()

SetHires24()

SetLoadAddress24()

SetLores24()

SetPFStencil24()

SetPRStencil24()

SetRGB24()

SetScreen24()

SetSprite24()

SingleDisplay24()

SinglePlayField24()

StopUpdate24()

UpdateAll24()

UpdateCoPro24()

UpdateDelay24()

UpdatePalette24()

UpdatePFStencil24()

UpdateRegs24()()

WriteFrame24()

WritePFPixel24()

WritePixel24()

WritePRPixel24()

WriteThumbnail24()

OpalRequester()

1.2 opal.library/ActiveScreen24

```
                NAME
ActiveScreen24 -- Provides a pointer to the currently displayed OpalVision  ←
   screen.

SYNOPSIS
OScrn = ActiveScreen24 (void);
D0

struct OpalScreen *OScrn;

FUNCTION
This function provides a pointer to the currently displayed OpalVision screen.  ←
   If there is no OpalVision
display active then a null value is returned.

This call is useful for writing background colour cycling or coprocessor effects ←
    programs to affect the
currently open screen.

INPUTS
None

RESULT
OScrn -A pointer to the currently open OpalVision screen, or NULL.

CONSIDERATIONS

Caution must be exercised when dealing with screens owned by another task.  ←
   Bitplane access should be
avoided unless running cooperative tasks with mutually exclusive bitplane access ←
   .

SEE ALSO


              OpenScreen24()
```

1.3 opal.library/AmigaPriority

```
                NAME
AmigaPriority -- Gives Amiga graphics priority over OpalVision display.

SYNOPSIS
void AmigaPriority (void);

FUNCTION
This function clears the OVPRI bit of all CoPro instructions which gives Amiga  ←
   graphics priority over
OpalVision graphics. If a dual display has not been set, only Amiga graphics  ←
   will be visible.

INPUTS
None
```

```
RESULT
None

CONSIDERATIONS
If an Opal display bottom has been set, the coprocessor instructions will not be
modified for that region of the display.

SEE ALSO

              OVPriority()

              DualDisplay24()
```

1.4 opal.library/AppendCopper24

```
NAME
AppendCopper24 -- Attaches user copper lists to existing display copper lists.

SYNOPSIS
void AppendCopper24 (CopLists);
      A0

UWORD **CopLists[12];

FUNCTION
Up to 12 different Amiga copper lists are used to update the OpalVision memory. This function allows
user copper lists to be attached to the end of each of the lists to enable split screen 24bit displays and
other copper effects.

Each copper list must be terminated with $FFFFFFFE followed by 30 free bytes for linkage code.

After attaching copper lists, the LastWait field in the OpalScreen structure must be initialised with the
last vertical position wait in the attached copper lists.

The VStart field in the OpalScreen structure contains the scan line of the first displayed line for the
screen. To convert a display 'y' coordinate to a vertical copper wait instruction, use VWait = y + VStart.

INPUTS
CopLists  - Pointer to an array of 12 copper list pointers to be joined to the current display copper lists.

RESULT
None

CONSIDERATIONS
All copper lists must reside in chip ram.

SEE ALSO
```

1.5 opal.library/AutoSync24

```
NAME
AutoSync24 -- Enables auto horizontal synchronisation.

SYNOPSIS
void  AutoSync24 (Sync);
      D0

BOOL Sync;

FUNCTION
Enables the OpalVision's auto synchronisation mode (see "Horizontal  ↩
   Synchronisation").  This mode will
be automatically disabled when frame buffer updates are occurring, and re- ↩
   enabled when they cease.

INPUTS
Sync =  0 = Disable auto syncing,  1 = Enable auto syncing.

RESULT
None

CONSIDERATIONS

SEE ALSO
```

1.6 opal.library/BitPlanetoOV

```
                NAME
BitPlanetoOV -- Converts standard bitplane data to OpalVision format.

SYNOPSIS
void BitPlanetoOV (OScrn, SrcPlanes, SrcWidth, Lines, TopLine, SrcDepth)
       A0   A1       D0        D1     D2       D3

struct OpalScreen *OScrn;
UBYTE **SrcPlanes[];
long SrcWidth;
long Lines;
long TopLine;
long SrcDepth;

FUNCTION

Converts bit plane data from the supplied bitplanes into OpalVision memory  ↩
   format and stores this in the
OpalScreen supplied.

The source data will be clipped if it is wider than the destination screen, or  ↩
   will be padded out if it is
narrower.
```

```
INPUTS
OScrn = Destination OpalScreen.
SrcPlanes = A pointer to an array of pointers to source Bitplanes.
SrcWidth  = Byte width of source planes (must be even).
Lines = Number of lines to convert.
TopLine   = Starting line to place destination data.
SrcDepth  = The number of bitplanes in SrcPlanes.

RESULT
None

CONSIDERATIONS
All bitplanes must start on a word boundary, and SrcWidth must be even.

SEE ALSO

              OVtoBitPlane()
```

1.7 opal.library/ClearDisplayBottom24

```
                NAME
ClearDisplayBottom24 -- Clears the OpalVision display bottom setting.

SYNOPSIS
void ClearDisplayBottom24 (void)

FUNCTION
Remove the OpalVision display bottom previously set with a call to
              SetDisplayBottom24()
              .

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO

              SetDisplayBottom24()
```

1.8 opal.library/ClearPFStencil24

```
                NAME
ClearPFStencil24 -- Clears the PlayField Stencil of the specified screen.

SYNOPSIS
```

```
void ClearPFStencil24 (OScrn);
      A0

struct OpalScreen *OScrn;

FUNCTION
Clears the playfield stencil (least significant bit of green bank 0) of all of  ↩
   the pixels in the specified
screen.

INPUTS
OScrn = OpalScreen structure.

RESULT
None

CONSIDERATIONS
This will only have a visible effect if Dual Playfield mode has been set up  ↩
   using DualPlayField24().

SEE ALSO

              SetPFStencil24()
                DualPlayField24()

              SinglePlayField24()
```

1.9 opal.library/ClearPRStencil24

```
                NAME
ClearPRStencil24 -- Clears the Priority Stencil of the specified Screen.

SYNOPSIS
void ClearPRStencil24 (OScrn);
      A0

struct OpalScreen *OScrn;

FUNCTION
Clears the priority stencil (least significant bit of blue bank 0) of the all of ↩
    the pixels in the specified
screen.

INPUTS
OScrn = OpalScreen structure.

RESULT
None

CONSIDERATIONS
This will only have a visible effect if dual OpalVision/Amiga display mode has  ↩
   been set up using

              DualDisplay24()
              .

SEE ALSO

              SetPRStencil24()

              DualDisplay24()

              SingleDisplay24()
```

1.10 opal.library/ClearQuick24

```
                NAME
ClearQuick24 -- Clears OpalVision frame buffer memory.

SYNOPSIS
void ClearQuick24 (void)

FUNCTION
This function clears the frame buffer memory as quickly as possible by enabling ↩
   a write to all banks of
memory. This function will also zero all bitplanes in memory (see ClearScreen24 ↩
   ()). This operation will
take 1 frame to clear any resolution non-interlaced display, and 2 frames for an ↩
    interlaced display.This
function  acts on the current display screen and cannot be used for virtual  ↩
   screens.

This function is called by OpenScreen24.

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO

              ClearScreen24()
```

1.11 opal.library/ClearScreen24

```
                NAME
ClearScreen24 -- Clears all bitplanes in a screen.


SYNOPSIS
```

```
void ClearScreen24 (OScrn)
        A0

struct OpalScreen *OScrn;

FUNCTION
Clear all bitplanes contained in the OpalScreen structure which may be a virtual ↩
    screen or the display
screen.

This function clears the bitplane memory without updating the frame buffer ( ↩
   unless frame buffer updates
are enabled).

INPUTS
OScrn = Pointer to the Opal screen to be cleared.

RESULT
None

CONSIDERATIONS


SEE ALSO

              ClearQuick24()
```

1.12 opal.library/CloseScreen24

```
                NAME
CloseScreen24 -- Stop current display and free resources.

SYNOPSIS
void CloseScreen24 (void);

FUNCTION
This function closes the current displayed screen if it was opened by the  ↩
   current task.

A screen opened by another task can be closed if it was opened with the  ↩
   CLOSEABLE24 flag. Backdrop
and other low priority programs should use the following procedure to open  ↩
   screen.

  OScrn = OpenScreen24 (CLOSEABLE24);
    .
    .
    .
    .
  if (OScrn!=NULL)
    { WaitPort (OScrn->UserPort);
       CloseScreen24();
       Mesg = GetMsg (OScrn->UserPort);
       ReplyMsg (Mesg);
    }
```

```
The task will be sent a message when another task is trying to open a screen or close down the one
already open. Note that the screen MUST be closed before replying to the message.

An alternative method to create a backdrop is to update the frame buffer, latch the 24 bit display using

              LatchDisplay24()
               and then call CloseScreen24(). The contents of the frame buffer will remain visible
until another task calls
              OpenScreen24()
              .

INPUTS
None

RESULT
None

CONSIDERATIONS

SEE ALSO


              OpenScreen24()
```

1.13 opal.library/Config24

```
NAME
Config24 -- Returns the OpalVision hardware configuration.

SYNOPSIS
Config = Config24 (void);
D0

long Config;

FUNCTION
Returns flags indicating the hardware configuration of the 24bit display card, future flags  will give
details on the existence of OpalVision modules such as the Video Roaster Chip and the frame grabber
genlock module.

Current return flags are:

  OVCF_OPALVISION - Display board is an OpalVision card.
  OVCF_COLORBURST - Display board is a ColorBurst.

INPUTS
```

```
RESULT


CONSIDERATIONS


SEE ALSO
```

1.14 opal.library/CreateScreen24

```
                NAME
CreateScreen24 -- Creates an arbitrarily sized virtual OpalScreen.

SYNOPSIS
OScrn = CreateScreen24 (ScreenModes,Width,Height)
      D0      D1    D2

struct OpalScreen *OScrn;
long Width;
long Height;
long ScreenModes;

FUNCTION
This function can create an arbitrarily sized OpalScreen in Fast Ram. The  ↩
   bitplanes for the screen are
allocated and an OpalScreen structure initialised, this is the virtual screen  ↩
   equivalent of
              OpenScreen24()
              .

Once this screen has been opened, all drawing, file and Memory conversion  ↩
   functions can be applied to
this screen, however it cannot be directly displayed. This allows large super  ↩
   bitmap screens to be
allocated in fast ram for manipulation, or to be partially copied to a primary  ↩
   OpalScreen in chip ram for
display (to allow for scrolling).

NOTE: Virtual screens are now  displayable using the LowMemUpdate() function,  ↩
   virtual screens are
therefore recommended when doing one off frame buffer updates (such as the  ↩
   Show24 command) as it
significantly reduces the chip ram requirements

INPUTS
Width       = Width in pixels of the screen to be opened.
Height        = Height in pixels of the screen to be opened.
ScreenModes = ScreenModes are identical to those of
              OpenScreen24()
              .

RESULT
OScrn = Is a pointer to the new OpalScreen structure or NULL if there is  ↩
   insufficient memory
  to open the screen size specified.
```

```
CONSIDERATIONS
This function allocates memory with no MEMF_ bits set, the program FastMemFirst should be executed
to force all planes to be loaded into fast ram, under AmigaDOS 1.3 or previous.

SEE ALSO

              FreeScreen24()
                OpenScreeen24()
```

1.15 opal.library/DisablePRStencil24

```
                NAME
DisablePRStencil24 -- Disables the use of the priority stencil in dual display mode.

SYNOPSIS
void DisablePRStencil24 (void);

FUNCTION
This function clears the PRISTENCIL bit of all CoPro instructions.

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set using
              SetDisplayBottom24()
              , the CoPro instructions will not be
modified for that part of the display.

The priority stencil will only have an effect when a Dual Display is enabled by calling
              DualDisplay24()
              .

SEE ALSO
EnablePRStencil()

              DualDisplay24()

              SingleDisplay24()
```

1.16 opal.library/DisplayFrame24

```
                NAME
DisplayFrame24 -- Sets the currently displayed frame within the frame buffer  ←
   memory.

SYNOPSIS
void DisplayFrame24 (Frame);
         D0

long Frame;

FUNCTION
Depending on the resolution of the displayed OpalVision screen, a number of  ←
   screens can be stored in the
frame buffer memory. The number of frames available for the screens resolution  ←
   are given in the
MaxFrames field in the OpalScreen structure.

DisplayFrame24() allows each individual frame to be displayed separately where  ←
   Frame is in the range
0...MaxFrames. Using a combination of WriteFrame24 and DisplayFrame24, it is  ←
   possible to store
several images in frame buffer memory and to perform simple page flip animation.

INPUTS
Frame = Frame number to display (0...MaxFrames).

RESULT
None

CONSIDERATIONS
The display frame and the write frame, must reside in the same field area in the ←
    frame buffer memory.
(See "Memory Segment Diagram"). Due to this DisplayFrame24() has the side effect ←
    of changing the
write frame if the new display frame is in a different field.

SEE ALSO

              WriteFrame24()
```

1.17 opal.library/DisplayThumbnail24

```
                NAME
DisplayThumbnail24 -- Displays a file’s thumbnail.

SYNOPSIS
ReturnCode = DisplayThumbnail24 (OScrn, FileName, x, y);
D0         A0 A1    D0 D1

long ReturnCode;
struct OpalScreen *OScrn;
char *FileName;
long x;
long y;

FUNCTION
This function displays the imbedded thumbnail in the file described by FileName ↩
   if it exists.

The x coordinate is rounded down to the nearest multiple of four in low ↩
   resolution mode and to the
nearest multiple of 8 in high resolution mode. The y coordinate is rounded down ↩
   to the nearest even line
in interlaced mode.

INPUTS

OScrn = Pointer to an OpalScreen structure.
FileName= The file name of the picture file with the thumbnail.
x = The x coordinate of the  screen position to display the thumbnail.
y = The y coordinate of the screen position to display the thumbnail.

RESULT
ReturnCode  = OL_ERR Codes described in Opallib.h.
OL_ERR_NOTHUMBNAIL is returned if no thumbnail exists in the file.

CONSIDERATIONS
Thumbnails must always be displayed in low resolution non interlaced mode. For ↩
   an example of
displaying thumbnails in a high resolution and interlaced screen, see the ↩
   example program
"DisplayDir.c"

SEE ALSO

              WriteThumbnail24()

              SaveIFF24()
```

1.18 opal.library/DrawEllipse24

```
NAME
DrawEllipse24 -- Draw an ellipse of given dimensions.

SYNOPSIS
void DrawEllipse24 (OScrn,  cx,  cy,  a,  b)
        A0      D0   D1   D2  D3

struct OpalScreen *OScrn;
long cx;
long cy;
long a;
long b;

FUNCTION
Draws an ellipse in the supplied screen.
```

NOTE: set a=b for circles.

INPUTS

OScrn = Destination OpalScreen.
cx = Centre x-Coordinate of ellipse.
cy = Centre y-Coordinate of ellipse.
a = horizontal radius of ellipse (must be >0).
b = vertical radius of ellipse (must be >0).

RESULT

CONSIDERATIONS
The ellipse will only be rendered in the region specified by the clip region in the screen structure.

SEE ALSO

1.19 opal.library/DrawLine24

NAME
DrawLine24 -- Draws a line into an OpalScreen.

SYNOPSIS

```
void DrawLine24 (OScrn, x1, y1, x2, y2)
     A0 D0  D1  D2  D3

struct OpalScreen *OScrn;
long x1;
long y1;
long x2;
long y2;
```

FUNCTION
Draws a line in the specified screen structure which may be a virtual or display screen.

In 24bit mode the colour of the line is specified by the Pen_R, Pen_G and Pen_B fields in the OpalScreen
structure. In 15bit mode, Pen_R and Pen_G specify the colour of the line, while in 8bit only Pen_R is
used.

INPUTS
OScrn = The OpalScreen structure in which to draw.
x1, y1 = The starting co-ordinates of the line.
x2, y2 = The ending co-ordinates of the line.

RESULT

CONSIDERATIONS
The line will only be rendered in the region specified by the clip region in the screen structure.

```
SEE ALSO
```

1.20 opal.library/DualDisplay24

```
                NAME
DualDisplay24 -- Sets up an Amiga/OpalVision dual display.

SYNOPSIS
void DualDisplay24 (void);

FUNCTION
This function clears the DUALDISPLAY bit of all CoPro instructions, enabling a  ↩
   Dual
Amiga/OpalVision display. The priority of the Amiga/OpalVision graphics can be  ↩
   set with
              OVPriority()
                and
              AmigaPriority()
              .

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set, using the
              SetDisplayBottom24()
               the CoPro instructions will not
be modified for that region of the display.

SEE ALSO

              SingleDisplay24()

              AmigaPriority()

              OVPriority()
```

1.21 opal.library/DualPlayField2

```
                NAME
DualPlayField24 -- Sets up an OpalVision 24bit dual playfield.

SYNOPSIS
void DualPlayField (void)

FUNCTION
This function sets the DUALPLAYFIELD bit of all CoPro instructions, allowing a  ↩
   dual 24 bit overlay
```

```
mode. To determine which bank is displayed for each pixel, the playfield stencil needs to be set
accordingly.

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set using
              SetDisplayBottom24()
              , the coprocessor instructions will
not be modified for that region of the display.

SEE ALSO

              SinglePlayField24()
```

1.22 opal.library/EnablePRStencil24

```
                NAME
EnablePRStencil24 -- Enables the use of the priority stencil in dual display mode.

SYNOPSIS
void EnablePRStencil24 (void);

FUNCTION
This function set the PRISTENCIL bit of all CoPro instructions.

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set using
              SetDisplayBottom24()
              , the CoPro instructions will not be
modified for that part of the display.

The priority stencil will only have an effect when a Dual Display is enabled by calling
              DualDisplay24()
              .

SEE ALSO
DisablePRStencil()

              DualDisplay24()

              SingleDisplay24()
```

1.23 opal.library/FadeIn24

```
                 NAME
  FadeIn24 -- Fades display in from black.

  SYNOPSIS
  void FadeIn24 (Time);
      D0

  long Time;

  FUNCTION
  Fade the current display from black to true colour.

  The Time parameter specifies the amount of time in 1/100 seconds the fade should ←
      take and is
  independent of PAL or NTSC refresh rates.

  INPUTS
  Time = Time in 1/100 seconds to complete fade.

  RESULT
  None

  CONSIDERATIONS
  This function cannot be used in 15bit mode.

  SEE ALSO

               FadeOut24()
```

1.24 opal.library/FadeOut24

```
                 NAME
  FadeOut24 -- Fade display to black.

  SYNOPSIS
  void FadeOut24 (Time)
      D0
  long Time;

  FUNCTION
  Fade the current display from true colour to black.

  The Time parameter specifies the amount of time the fade should take and is  ←
     independent of PAL or
  NTSC machines.

  INPUTS
  Time = Number of 1/100ths seconds in which to complete the fade.
```

```
RESULT
None

CONSIDERATIONS
This function cannot be used in 15bit mode.

SEE ALSO

              FadeIn24()
```

1.25 opal.library/FreeScreen24

```
                NAME
FreeScreen24 -- Frees a virtual OpalScreen.

SYNOPSIS
void FreeScreen24 (OScrn);
       A0

struct OpalScreen *OScrn;

FUNCTION
This function deallocates all memory associated with a virtual screen. This is ↩
   the virtual screen
equivalent of CloseScreen24.

INPUTS
OScrn = A pointer to the virtual screen to be freed

RESULT
None

CONSIDERATIONS


SEE ALSO

              CreateScreen24()

              CloseScreen24()
```

1.26 opal.library/FreezeFrame24

```
NAME
FreezeFrame24 -- Freezes the currently displayed screen.

SYNOPSIS
void FreezeFrame24 (Freeze);
        D0
BOOL Freeze
```

```
FUNCTION
This function freezes the current display. If freeze is TRUE, the display is  ←
   held static. The display is
returned to normal when the value for freeze is FALSE.

Freeze freezes everything on the display including Amiga graphics.

INPUTS
Freeze  = TRUE (1) = Freeze, FALSE (0) = Unfreeze

RESULT
None

CONSIDERATIONS
This functions is available only while a Scan Rate Converter is present.

SEE ALSO
```

1.27 opal.library/ILBMtoOV

```
                NAME
ILBMtoOV -- Converts interleaved bitmap to OpalVision format.

SYNOPSIS
void ILBMtoOV (OScrn, ILBMData, SrcWidth, Lines, TopLine, SrcPlanes)
    A0    A1  D0    D1   D2   D3

struct OpalScreen *OScrn;
UBYTE *ILBMData;
long SrcWidth;
long Lines;
long TopLine;
long SrcPlanes;

FUNCTION
Converts interleaved bitmap memory into OpalVision memory format and stores this ←
    in the OpalScreen
supplied.

The source data will be clipped if it is wider than the destination screen, or  ←
   will be padded out if it is
narrower.

This function is provided to simplify the task of writing a custom IFF loader.

INPUTS
OScrn   = Destination OpalScreen.
ILBMData  = interleaved planes of source data.
SourceWidth = Width of source ILBM data.
Lines   = Number of lines to convert.
TopLine   = Starting line to place destination data.
SrcPlanes =The number of planes contained in the ILBM data.

RESULT
```

```
None

CONSIDERATIONS


SEE ALSO

              OVtoILBM()
```

1.28 opal.library/LatchDisplay24

```
                NAME
LatchDisplay24 -- Locks OpalVision  display

SYNOPSIS
void LatchDisplay24 (Latch)

BOOL Latch;

FUNCTION
LatchDisplay24 sets or clears the Latch bit in the control line register. If  ←
   this bit is set,  the OpalVision
display will remain active regardless of whether there is a valid control line  ←
   in the Amigas' output. If

              CloseScreen24()
               is called after the latch bit is set, all memory and resources  ←
                  will be freed but the display
will still be active, even if the Amiga is reset.

OpalHotKey uses this technique, images are loaded and updated into the buffer,  ←
   latched and then the
screen is closed. If any register information needs to be changed, such as  ←
   changing display priority a
display screen is opened using the CONTROLONLY24 flag which enables registers to ←
    be changed
without effecting the contents of the frame buffer.

INPUTS
Latch = 0 = Free display, 1 = Latch display.

RESULT
None

CONSIDERATIONS


SEE ALSO
```

1.29 opal.library/LoadImage24

NAME

LoadImage24 / LoadIFF24 -- Loads an Image file.

SYNOPSIS

```
ReturnCode = LoadImage24 (OScrn,FileName,Flags)
D0          A0  A1   D0

long ReturnCode;
struct OpalScreen *OScrn;
long Flags;
char *FileName;
```

FUNCTION

Load an IFF or JPEG file.

This is a general purpose image loading routine which will automatically detect and load IFF and JPEG files. As this is a general loader, the name of this function has been renamed to LoadImage24(), which is used as a synonym for the previous function name LoadIFF24() to maintain backward compatibility.

The IFF portion of this loader will load IFF 24bit, Fast Format 24 bit, Palette mapped (up to 256 colours), Hold and Modify and Extra half brite files.

All palette mapped files will be loaded in the 8 bit palette mapped mode unless the CONVERT24 flag is set, in which case they will be converted to a non palette mapped 24 bit display.

There are several different forms in which LoadImage24 can load an image. The way in which it functions is dependant on the specified screen and the flags. If the screen pointer is NULL then LoadImage24 will open a screen itself, the screen it opens will be a display screen unless VIRTUALSCREEN24 is set in which case a virtual screen will be created.

If the passed screen structure is not NULL and the image being loaded is the same resolution, then it will be loaded into that screen. If this is not the case then the screen will be closed and a new screen of the same resolution as the file will be opened. However if KEEPRES24 is set, the file will be loaded into the supplied screen regardless of its resolution.

LoadImage24 returns one of two things. If the files was loaded successfully, a pointer to the screen into which it was loaded is returned. If an error occurred, then an error code will be returned. To determine which of these messages has been returned, the value can be compared to OL_ERR_MAXERR, if it is lower than this value then the result is an error code, if it is greater than this number then it is a screen pointer. If the image is not IFF or JPEG, OL_ERR_FORMATUNKOWN is returned.

```
Flags:
  FORCE24   -  Convert palette mapped files to 24 bit.
  KEEPRES24   -  Keep the same screen resolution.
  CLOSEABLE24   -  Opened screen will be closeable.
  LOADMASK24    -  Load mask plane if present (IFF only).
  VIRTUALSCREEN24 -  Load image into a virtual screen.

The JPEG loader is a baseline loader as specified in the draft standard ISO/IEC  ↩
    Bis 10918-1 it supports
only 8 bit quantization tables and Huffman entropy compression. It can load  ↩
    files with source colour
space of Y Cb Cr, RGB and Grey scale. It does not support non interleaved files, ↩
     progressive, hierarchical
or lossless modes.

INPUTS
FileName  = Filename of image to be display (including path).
Flags   = see above.

RESULT
ReturnCode = > OL_ERR_MAXERR Return code is a pointer to an Opal screen  ↩
    structure.
ReturnCode = < OL_ERR_MAXERR, Return code indicates error.

CONSIDERATIONS
This function only loads an image, it does not update the frame buffer. To do  ↩
    this you must call

              Refresh24()
               or
              LowMemUpdate24()
              .

SEE ALSO

              SaveIFF24()

              SaveJPEG24()

              Refresh24()

              LowMemUpdate24()
```

1.30 opal.library/LowMemUpdate24

```
                NAME
LowMemUpdate24 -- Low chip ram usage OpalVision update.

SYNOPSIS
RetScrn = LowMemUpdate24 (OScrn, Frame);
D0         A0   D0

struct OpalScreen *RetScrn;
struct OpalScreen *OScrn;
long Frame;

FUNCTION
Updates the frame buffer from a virtual screen. This function can update an  ↩
   entire image of any
resolution while only using a small amount of chip ram. This routine uses an 8 ↩
   bit screen to update each
memory segment separately, the CPU is used to copy the bitplane data from the  ↩
   virtual screen to chip
ram. The 8bit plane display screen opened to perform the update is returned, and ↩
    should be subsequently
closed.

The Frame input sets the first memory segment to be updated, this will normally  ↩
   be 0. This can be set to
6 for example to update a lores screen to bank1  instead of bank0.

NOTE:  OScrn must be a pointer to a virtual screen.

INPUTS
OScrn = The virtual OpalScreen to be displayed
Frame = Memory segment to start update (0..11).

RESULT
RetScrn    >= OL_ERR_MAXERR Return code is a pointer to an Opal screen structure ↩
   .
RetScrn   < OL_ERR_MAXERR, Return code indicates error.

CONSIDERATIONS

SEE ALSO

              LowMem2Update24()
```

1.31 opal.library/LowMem2Update24

```
                NAME
LowMem2Update24 -- Low chip ram usage OpalVision update.

SYNOPSIS
RetScrn = LowMem2Update24 (OScrn, Frame);
D0         A0   D0

struct OpalScreen *RetScrn;
struct OpalScreen *OScrn;
long Frame;

FUNCTION
Updates the frame buffer from a virtual screen. This function can update an  ↩
   entire image of any
resolution while only using a small amount of chip ram. This routine uses an 8 ↩
   bit screen to update each
```

```
memory segment separately, the CPU is used to copy the bitplane data from the  ↩
   virtual screen to chip
ram. The 8bit plane display screen opened to perform the update is returned, and ↩
    should be subsequently
closed.

The Frame input sets the first memory segment to be updated, this will normally  ↩
   be 0. This can be set to
6 for example to update a lores screen to bank1  instead of bank0.

This function is similar to
              LowMemUpdate24()
               although it only updates the frame buffer memory, it does
not modify the display modes,  CoPro bits or palette information. This is very  ↩
   useful for performing
transitions  between two images in lores, the first image can be written into  ↩
   bank1 and displayed using
LowMemUpdate24(OScrn,6), the second image is then updated transparently into  ↩
   bank0 using
LowMem2Update24(OScrn,0). The dual display stencil can then be used to perform  ↩
   the transition
between bank1 and bank0. Note that bank0 is written to last , as only bank0  ↩
   contains the dual display
stencil.

NOTE:  OScrn must be a pointer to a virtual screen.

INPUTS
OScrn = The virtual OpalScreen to be displayed
Frame = Memory segment to start update (0..11).

RESULT
RetScrn    >= OL_ERR_MAXERR Return code is a pointer to an Opal screen structure ↩
   .
RetScrn   < OL_ERR_MAXERR, Return code indicates error.

CONSIDERATIONS

SEE ALSO

              LowMemUpdate24()
```

1.32 opal.library/LowMemRGB24

```
NAME
LowMemRGB24 -- Low chip ram usage OpalVision update from an RGB array.

SYNOPSIS
RetScrn = LowMemRGB24 (ScreenModes, Frame, Width, Height, Modulo, RGBPlanes);
D0      D0      D1     D2   D3    D4    A0

struct OpalScreen *RetScrn;
long ScreenModes,Frame,Width,Height,Modulo;
UBYTE *RGBPlanes[3]
```

```
FUNCTION

Updates the frame buffer from RGB byte planes. This function can update an  ↩
   entire image of any
resolution while only using a small amount of chip ram. This routine uses an 8 ↩
   bit screen to update each
memory segment separately, the RGB data is converted into bitplane format one  ↩
   segment at a time and
transferred into the framebuffer. The 8bit plane display screen opened to  ↩
   perform the update is returned,
and should be subsequently closed.

The Modulo parameter allows interleaved RGB data to be updated as well, in this  ↩
   case RGBPlanes would
be initialised 'Width' bytes apart, and modulo would be set to 2*Width.

This function is useful for image processing programs such as ADPro and  ↩
   Imagemaster which store
images in byte planes.

INPUTS
ScreenModes = See OpenScreen24, these flags enable the resolution and format of  ↩
   the
       displayed image to be set.
Frame    = The memory segment to start the update, (0..1)
Width    = Width of the RGB Array (in pixels).
Height    = Height of the RGB Array.
Modulo    = Modulo to be added after each line in the RGB Array.
RGBPlanes = Pointers to the three byte planes required (R,G,B).

RESULT
RetScrn    >= OL_ERR_MAXERR Return code is a pointer to an Opal screen structure ↩
   .
RetScrn   < OL_ERR_MAXERR, Return code indicates error.

CONSIDERATIONS

SEE ALSO
```

1.33 opal.library/OpenScreen24

```
                NAME
OpenScreen24 -- Allocates all resources and displays an OpalVision screen.

SYNOPSIS
OScrn = OpenScreen24 (ScreenModes)
D0          D0

struct OpalScreen *OScrn;
long ScreenModes;

FUNCTION
This function creates a display screen and allocates all the resources required  ↩
   to display the screen. The
```

Frame buffer memory is cleared and updates are disabled to the frame buffer memory.

The screen is positioned according to Amiga preferences, however if the vertical starting position defined
in preferences is too high up, preferences will be modified to set the vertical starting position to the
highest possible, the preferences will be restored when the screen is closed.

The screen will be opened as single playfield, single display mode with OVPriority.

If the CONTROLONLY24 flag is set, the screen will be opened without any bitplanes, this enables the
copro or palette information of a 'latched on' to be modified without losing the contents of the frame
buffer.

```
ScreenModes:
  INTERLACE24 - Open an interlaced screen.
  HIRES24   - Open a Hires screen.
  OVERSCAN24  - Open an overscan screen.
  PLANES15  - 15 bit true colour display.
  PLANES8   - 8 bit true colour/palette mapped display.
  CLOSEABLE24 - Screen can be closed by another task.
  PALMAP24  - Open a palette mapped screen.
  CONTROLONLY24 - Open a bitplaneless screen.
```

Screen Sizes:

Hires	Interlaced	Overscan	PAL	NTSC
No	No	No	320x256	320x200
Yes	No	No	640x256	640x200
No	Yes	No	320x512	320x400
Yes	Yes	No	640x512	640x400

The size of the screen opened will be as specified above unless there is not enough chip ram available, in
which case the maximum amount of lines possible will be displayed. If there is insufficient chip memory,
to hold half of the scan lines, then OpenScreen will be aborted and NULL returned. If the PLANES8 or
PLANES15 flag is not set, a 24 bit screen will be opened.

INPUTS
ScreenModes = See above.

RESULT
OScrn = A pointer to an OpalScreen structure or NULL if unsuccessful.

CONSIDERATIONS

SEE ALSO

CloseScreen24()

```
              LatchDisplay24()
```

1.34 opal.library/OVPriority

```
                NAME
OVPriority -- Give OpalVision graphics priority over Amiga graphics.

SYNOPSIS
void OVPriority (void);

FUNCTION
This function sets the OVPRI bit of all coprocessor instructions which gives  ↩
   OpalVision graphics priority
over Amiga graphics. If a dual display has not been set, only OpalVision  ↩
   graphics will be visible.

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set using
              SetDisplayBottom24()
              , the coprocessor instructions will
not be modified for that region of the display.

SEE ALSO
AmigaPriority24()

              DualDisplay24()
```

1.35 opal.library/OVtoBitPlane

```
NAME
OVtoBitPlane -- Convert OpalVision bit plane data to standard bitplanes.

SYNOPSIS
void OVtoBitPlane (OScrn, BitPlanes, DestWidth, Lines, TopLine)
       A0   A1       D0   D1     D2

struct OpalScreen *OScrn;
UBYTE **BitPlanes[];
long DestWidth;
long Lines;
long TopLine;

FUNCTION
Converts OpalVision bitplane format to standard bitplane data. The destination  ↩
   data will be non-
```

```
interleaved 24, 15 or 8 planes depending on the type of display OScrn is. Note  ↩
   that the 15bit display
mode is actually stored internally as 16 bitplanes which in turn causes this  ↩
   function to return 16 planes
instead of 15.

DestWidth specifies the width of the destination bitplanes, if the width is less ↩
    than the source planes, they
will be clipped. If the destination width is larger, the remaining bytes on each ↩
    scan line will be skipped.

The OpalScreen can be any size, and reside in fast or chip ram.

The array of bitplane pointers passed to this function must contain 8, 16 or 24  ↩
   entries depending on the
screen type.

INPUTS
OScrn   = OpalScreen structure describing source data.
BitPlanes = Array of bitplane pointers to take the destination data.
DestWidth = Width in bytes of destination planes (must be even).
Lines   = Total number of scan lines to convert.
TopLine   = Starting line for conversion within the OpalScreen.

RESULT
None

CONSIDERATIONS
All bitplanes must be on a word boundary and the destination width must be even.

SEE ALSO
BitplanetoOV()
```

1.36 opal.library/OVtoILBM

```
                NAME
OVtoILBM -- Converts OpalVision bit planes to interleaved bitmap format.

SYNOPSIS
void OVtoILBM (OScrn, ILBMData, DestWidth, Lines, TopLine)
    A0    A1  D0     D1   D2

struct OpalScreen *OScrn;
UBYTE *ILBMData;
long DestWidth;
long Lines;
long TopLine;

FUNCTION
Converts bitplane information from the supplied OpalScreen, starting at the scan ↩
    line indicated by
TopLine, into interleaved bitmap format.

If the source OpalScreen is wider than the destination width, the planes will be ↩
    clipped. If the OpalScreen
```

```
is narrower, the extra bytes on each line will be skipped. The OpalScreen can be ↩
    any size, and reside in
fast or chip ram.

The memory pointed to by ILBMData must be large enough to hold
DestWidth * lines * (8 or 16 or 24 depending on screen type) bytes.

INPUTS
OScrn     = OpalScreen structure describing source data.
ILBMData  = Pointer to buffer to hold destination ILBM data.
DestWidth = Width of destination ILBM planes. (must be even)
Lines     = Total number of scan lines to convert.
TopLine   = Starting line for conversion within the OpalScreen.

RESULT
None

CONSIDERATIONS
ILBMData must start on a word boundary, and DestWidth must be even.

SEE ALSO

              ILBMtoOV()
```

1.37 opal.library/OVtoRGB

```
                NAME
OVtoRGB -- Converts OpalVision bitplane data to three planes of RGB.

SYNOPSIS
void OVtoRGB (OScrn, RGBPlanes[], Top, Left, Width, Height)
        A0     A1      D0   D1    D2     D3

struct OpalScreen *OScrn;
UBYTE **RGBPlanes[];
long Top;
long Left;
long Width;
long Height;

FUNCTION
This call converts bitplane data from the OpalScreen into three planes, one  ↩
   containing Red, one Blue and
the last Green, each of these has one byte per pixel. This is useful for making  ↩
   'brush' cut-outs, or for
subsequent scaling of data.

This function is more flexible than the other memory conversion routines in that ↩
    it can convert a
rectangular region of bitplane memory positioned anywhere within the source  ↩
   screen.

The OpalScreen can be any size, and reside in fast or chip ram.
```

```
INPUTS
OScrn   = OpalScreen structure containing source bitplanes.
RGBPlanes = Pointer to an array of 3 plane pointers.
Top   = x coordinate of top left hand corner to start conversion.
Left    = y coordinate of top left hand corner to start conversion.
Width   = Width in pixels of region to cut.
Height    = Number of lines to cut.

RESULT


CONSIDERATIONS
The destination planes must be on a word boundary.

SEE ALSO

              RGBtoOV()
```

1.38 opal.library/PaletteMap24

```
NAME
PaletteMap24 -- Enable/Disable palette mapping.

SYNOPSIS
PaletteMap24 (PaletteMap)
    D0

BOOL PaletteMap;

FUNCTION
If PaletteMap = TRUE, turn on palette mapping, else turn palette mapping off.  ↩
   This function always
operates on the active display screen.

INPUTS
PaletteMap = True to turn palette mapping on, to turn it off.

RESULT
None

CONSIDERATIONS
This function cannot be used in 15bit mode.

SEE ALSO
SetPalette24()
```

1.39 opal.library/ReadPFPixel24

```
                NAME
ReadPFPixel24 -- Returns the state of a give playfield stencil pixel.
```

```
SYNOPSIS
Result = ReadPFPixel24 (OScrn,  x,  y)
D0      A0  D0  D1

long Result;
struct OpalScreen *OScrn;
long x;
long y;

FUNCTION
This function returns 1 if the corresponding playfield stencil pixel is set, or 0 if it is cleared. If the
coordinates are outside of the clip boundary then -1 is returned.

INPUTS
OScrn = OpalScreen to be read.
x = x Coordinate of pixel to read.
y = y Coordinate of pixel to read.

RESULT
Result = 0 if pixel clear , or 1 if pixel set, -1 if pixel is out of range.

CONSIDERATIONS

SEE ALSO

              WritePFPixel24()

              ClearPFStencil24()

              SetPFStencil24()
```

1.40 opal.library/ReadPixel24

```
                NAME
ReadPixel24 -- Returns colour information for a given pixel.

SYNOPSIS
Error = ReadPixel24 (OScrn,  x,  y)
D0         A0      D0  D1

long Error;
struct OpalScreen *OScrn;
long x;
long y;

FUNCTION
Return the colour (bit plane) information for a given pixel. If the OpalScreen is in palette mapped mode,
the actual bit plane data (and not the corresponding palette value) will be returned.

The returned value is placed in Red, Green and Blue in the OpalScreen structure while in 24bit modes. In
```

```
15bit mode, the colour is returned in Red and Green, while in 8bit mode, the colour is returned in Red.
The GetPen macros can be used to extract the components from the returned value.

Use the macros GetCol24(), GetCol15(), GetCol8() or GetCol8P() for 8 bit palette mapped to return the
pixel value.

If the coordinates are outside the screen's clipping region, Error will be -1, else Error = 0.

This function can operate on any sized screens in chip or fast ram.

INPUTS
OScrn = OpalScreen to be read.
x = x Coordinate of pixel to read.
y = y Coordinate of pixel to read.

RESULT
Error = 0 if no error occurred, or -1 if pixel was out of the clipping region.

CONSIDERATIONS


SEE ALSO


              WritePixel24()
```

1.41 opal.library/ReadPRPixel24

```
                NAME
ReadPRPixel24 -- Returns the state of a give priority stencil pixel.

SYNOPSIS
Result = ReadPRPixel24 (OScrn,  x,  y)
D0      A0  D0  D1

long Result;
struct OpalScreen *OScrn;
long x;
long y;

FUNCTION
This function returns 1 if the corresponding priority stencil pixel is set, or 0 if it is cleared. If the
coordinates are outside of the clip boundary then -1 is returned.

INPUTS
OScrn = OpalScreen to be read.
x = x Coordinate of pixel to read.
y = y Coordinate of pixel to read.

RESULT
```

```
Result  = 0 if pixel clear , or 1 if pixel set, -1 if pixel is out of range.

CONSIDERATIONS

SEE ALSO


              WritePRPixel24()

              ClearPRStencil24()

              SetPRStencil24()
```

1.42 opal.library/RectFill24

```
NAME
RectFill24 -- Draws a solid rectangle.

SYNOPSIS
void RectFill24 (OScrn, Left, Top, Bottom, Right)
     A0 D0    D1   D2    D3

struct OpalScreen *OScrn;
long Left;
long Top;
long Bottom;
long Right;

FUNCTION
Draws a solid rectangle with the colour specified by Pen_R,Pen_G & Pen_B in the ←
   OpalScreen structure.

The Rectangle is clipped if all or part of it lies outside the clipping region.

INPUTS
OScrn = OpalScreen to be rendered into.
Left  = x coordinate of top left-hand corner of the rectangle.
Top = y coordinate of top left-hand corner of the rectangle.
Bottom  = x coordinate of the bottom right-hand corner of rectangle.
Right = y coordinate of the bottom right-hand corner of rectangle.

RESULT
None

CONSIDERATIONS


SEE ALSO
```

1.43 opal.library/Refresh24

```
                NAME
Refresh24 -- Refreshes the frame buffer.

SYNOPSIS
void Refresh24 (void)

FUNCTION
Initiates DMA of the currently displayed OpalScreen to the framebuffer. This  ↩
   function will update the
framebuffer in the minimum number of frames required, stop DMA (updates) and  ↩
   return.

This function should be called after any drawing routine, and other routines  ↩
   such as LoadIFF24 which
modify memory, to make the frame buffer (and hence display) consistent with the  ↩
   image in Amiga
memory.

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO

              UpdateDelay24()

              StopUpdate24()

              UpdatePFStencil24()

              UpdateAll24()
```

1.44 opal.library/RegWait24

```
NAME
RegWait24 -- Wait for register update to complete.

SYNOPSIS
void RegWait24 (void);

FUNCTION
This function waits for register information to be updated to the OpalVision  ↩
   before returning, or returns
immediately if no updates are pending.

This function is important for synchronizing your program with the OpalVision's  ↩
   update scheme. After
```

```
any direct modification of OpalVision registers or after a call to a library  ↩
    function which modifies
registers, this function should be called to allow the update to occur, If this  ↩
    function is not called register
data may be lost.

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO
```

1.45 opal.library/RGBtoOV

```
                NAME
RGBtoOV -- Converts three planes of RGB to OpalVision bitplane data.

SYNOPSIS
void RGBtoOV (OScrn, RGBPlanes[], Top, Left, Width, Height)
        A0     A1      D0   D1    D2     D3

struct OpalScreen *OScrn;
UBYTE **RGBPlanes[];
long Top;
long Left;
long Width;
long Height;

FUNCTION
This call converts three source planes, one containing Red, one Blue and the  ↩
    last Green into OpalVision
bitplane format. This function is useful for pasting clipped regions (using  ↩
    OVtoRGB) back into
OpalVision memory, or for pasting back data after scaling.

Unlike the other conversion routines, this function is clipped if it is outside  ↩
    of the clipping region, this
enables it to be used as a drawing function rather than a conversion function.

The OpalScreen can be any size, and reside in fast or chip ram.

INPUTS
OScrn   = OpalScreen structure containing destination bitplanes.
RGBPlanes = Pointer to an array of 3 plane pointers.
Top   = x coordinate of top left hand corner to start conversion.
Left    = y coordinate of top left hand corner to start conversion.
Width   = Width in pixels of region to cut.
Height    = Number of lines to cut.

RESULT
```

```
None

CONSIDERATIONS


SEE ALSO

              OVtoRGB()
```

1.46 opal.library/SaveIFF24

```
                NAME
SaveIFF24 -- Save an OpalScreen as an IFF file.

SYNOPSIS
Error = SaveIFF24 (OScrn, FileName, ChunkFunction, Flags)
D0       A0   A1      A2       D0

long Error;
struct OpalScreen *OScrn;
char *FileName;
long (*ChunkFunction)();
long Flags;

FUNCTION

SaveIFF24 will save any sized OpalScreen in normal IFF file format. The chunks  ↩
   written will include

  CAMG  - Containing the resolution  (Hires/Interlace/Overscan)
  CMAP  - Colour map if in 8bit mode.
  CLUT  - Colour lookup tables if in true colour mode.
  OVTN  - OpalVision 24bit thumb-nail for display in OpalPaint,
      OpalShow and other system software.
  BODY  - Standard 24 bit ILBM data using byte run encoding.

If ChunkFunction is not NULL, the function that it points at will be called  ↩
   after the file has been opened
(and FORM ILBM has been written) and before any other chunks have been written.  ↩
   ChunkFunction is
used to insert your own chunks into the IFF file before any of the above chunks. ↩
    The DOS File Handle for
the open file will be passed to the function on the stack (in standard C calling ↩
    convention) the chunk
function must return 0 or an error code.

Flags:
  OVFASTFORMAT  - Save as OpalVision fast format.
  NOTHUMBNAIL - Inhibit writing thumb-nail chunk.
  SAVEMASK24    - Saves mask plane if one exists.

INPUTS
OScrn   = OpalScreen to be saved.
FileName  = Filename of file to be written (including full path).
```

```
ChunkFunction = Pointer to code to be executed after file is opened.

RESULT
Error = 0 if no error code, >0 if error occurred.

CONSIDERATIONS

SEE ALSO


              LoadImage24()

              SaveJPEG24()
```

1.47 opal.library/SaveJPEG24

```
                NAME
SaveJPEG24 -- Save an OpalScreen as a JPEG JFIF file.

SYNOPSIS
Error = SaveJPEG24 (OScrn, FileName, Flags, Quality)
D0        A0     A1      D0     D1

long Error;
struct OpalScreen *OScrn;
char *FileName;
long Flags;
long Quality;

FUNCTION
SaveJPEG24 will save any sized OpalScreen in the JPEG JFIF file format. JPEG is a compression
standard which enables a large amount of compression to be gained on continuous tone images with
minimum loss in image quality. It should be stressed that this compression method is based on
continuous tone images and compression of images with sharp edges may suffer more degradation. For
more details see the JPEG draft  standard ISO/IEC Dis10918-1.

This generates a base line JPEG file using interleaved components, Huffman entropy compression and 8
bit quatization tables. A thumbnail will also be written into the APP0 marker of the JFIF file unless the
NOTHUMBNAIL flag is set.

The quality factor is a percentage value (0...100) which defines the allowable amount of loss in the
compressed image. A factor of 100 corresponds to a quantization table of all 1's and hence has no
quantization loss. A value of 50 corresponds to the quantization tables suggested by the draft standard as
being acceptable for good image quality. A reasonable default value to use is 75, using this level for
```

```
continuous tone scanned images a compression factor of  between 15:1 and 20:1 is ↩
    typical.

Flags:
  NOTHUMBNAIL - Inhibit writing thumb-nail chunk.

INPUTS
OScrn = OpalScreen to be saved.
FileName= Filename of file to be written (including full path).
Flags = See above.
Quality = (0...100) This determines the amount of loss allowed in the  ↩
   compression of
     the image. 100 % corresponds to minimum loss.

RESULT
Error   = 0 if no error code, >0 if error occurred.

CONSIDERATIONS

SEE ALSO


             LoadImage24()
```

1.48 opal.library/Scroll24

```
                NAME
Scroll24 -- Scrolls currently displayed OpalVision image.

SYNOPSIS
void Scroll24 (DeltaX, DeltaY)
    D0     D1

long DeltaX;
long DeltaY;

FUNCTION
This function scrolls the currently displayed image by DeltaX pixels  ↩
   horizontally, and DeltaY lines
vertically, by modifying the video load address register in the OpalVision.

DeltaX and DeltaY are signed values, to enable scrolling in all directions.

This function also clears the ADDLOAD bit on the first CoPro instruction if it  ↩
   is not already cleared.

INPUTS
DeltaX  = Number of pixels to scroll horizontally.
DeltaY  = Number of lines to scroll vertically.

RESULT
None

CONSIDERATIONS
```

For the Scroll to function correctly, update DMA to the framebuffer must be turned off by calling

StopUpdate24()
.

SEE ALSO

SetLoadAddress24()

1.49 opal.library/SetControlBit24

NAME
SetControlBit24 -- Modifies a bit in the control line register.

SYNOPSIS

```
void SetControlBit24 (FrameNumber, BitNumber, State)
          D0     D1       D2

long FrameNumber;
long BitNumber;
BOOL State;
```

FUNCTION
Sets or clears a bit in the control line register. See "The Opal Control Line Register" for details.

There are 14 different versions of the control line register used to update the maximum of 12 different
memory segments. These differ by the state of the bank and field write enable bits. The frame number
variable specifies which one of these registers should be updated, for bits such as AUTO or COL/CoPro a
global change may be required (i.e. changing all 12 control lines).

Frame Number	Description
0	Red Bank0, Field0 Update
1	Green Bank0, Field0 Update
2	Blue Bank0, Field0 Update
3	Red Bank0, Field1 Update
4	Green Bank0, Field1 Update
5	Blue Bank0, Field1 Update
6	Red Bank1, Field0 Update
7	Green Bank1, Field0 Update
8	Blue Bank1, Field0 Update
9	Red Bank1, Field1 Update
10	Green Bank1, Field1 Update
11	Blue Bank1, Field1 Update
12	Field 0 Display only
13	Field 1 Display only

INPUTS

```
FrameNumber = The OpalVision update frame number to modify. One frame  ↩
    corresponds to
     one bank update (maximum 12 frames, 2 noupdate lists).
BitNumber = Bit number within control line to modify (4...19).
State    = State to be written into bit (Boolean).

RESULT
None

CONSIDERATIONS
These bits should be modified with caution.

SEE ALSO
Control Line Register
```

1.50 opal.library/SetCoPro24

```
                NAME
SetCoPro24 -- Modifies a single instruction in the CoPro list.

SYNOPSIS
void SetCoPro24 (InstructionNumber, Instruction);
     D0       D1

long InstructionNumber;
long Instruction;

FUNCTION
This function modifies a single CoPro instruction and initiates an update to the ↩
     OpalVision CoPro. Note
that this function is much faster than calling
              UpdateCoPro24()
              .

INPUTS
InstructionNumber = The CoPro instruction number (0...289)
Instruction    = 8Bit CoPro instruction. See "The CoPro"

RESULT
None

CONSIDERATIONS
InstructionNumber should be less than LastCoProIns in the OpalScreen structure.

SEE ALSO

              UpdateCoPro24()
```

1.51 opal.library/SetDisplayBottom24

```
NAME
SetDisplayBottom24 -- Sets the lower limit of the OpalVision screen.

SYNOPSIS
Result = SetDisplayBottom24 (BottomLine);
        D0

long BottomLine;
BOOL Result;

FUNCTION
This function specifies the lower limit of the OpalVision screen. Below this point Amiga only graphics
will be displayed. Once a display bottom has been set, the region below that line will always contains
Amiga graphics regardless of whether the frame buffer is being updated or not. This is useful for
displaying Amiga gadgets on the screen.

INPUTS
BottomLine  -Specifies the last line of OpalVision graphics.

RESULT
Result    = 1 if operation successful, 0 if operation failed.

CONSIDERATIONS
This function uses the CoPro to enable Amiga graphics on the bottom section of the screen. To ensure
that the display is not corrupted, only CoPro instructions up to the line specified by LastCoProIns in the
OpalScreen structure should be modified.

SEE ALSO
ClearDisplayBottom24 ()
```

1.52 opal.library/SetHires24

```
NAME
SetHires24 -- Enable a hires display for a section of the screen.

SYNOPSIS
Result = SetHires24 (TopLine, Lines);
         D0       D1
long TopLine;
long Lines;

FUNCTION
Sets the HIRESDISP bits on CoPro instructions starting at TopLine for 'Lines' number of lines. Both
TopLine and Lines must be specified as a non-interlaced scan line (i.e. must be divided by 2 if an
interlaced screen).

INPUTS
```

```
TopLine   -Specifies the first line to start setting HIRESDISP bits.
Lines   -Number of lines to modify.

RESULT
None

CONSIDERATIONS

SEE ALSO
SetLores24 ()
```

1.53 opal.library/SetLoadAddress24

```
                NAME
SetLoadAddress24 -- Updates the OpalVision load address register.

SYNOPSIS
void SetLoadAddress24 (void)

FUNCTION
This function uses the Load Address value in the displayed OpalScreen structure  ←
   to update the load
address register in the OpalVision.

This function is useful for scrolling and distortion effects.

The modulo for a scan line is given in the OpalScreen structure and is  ←
   independent of the display
resolution.

INPUTS
None

RESULT
None

CONSIDERATIONS
Load Address only has an effect when a CoPro instruction having its ADDLOAD bit  ←
   cleared is executed.
Therefore a combination of CoPro instructions and the address load register are  ←
   required to produce the
effect.

SEE ALSO

              Scroll24()
```

1.54 opal.library/SetLores24

```
NAME
SetLores24 -- Enable a Lores display for a section of the screen.
```

```
SYNOPSIS
Result = SetLores24 (TopLine, Lines);
         D0       D1
long TopLine;
long Lines;

FUNCTION
Clears the HIRESDISP bits on CoPro instructions starting at TopLine for 'Lines' ←↩
   number of lines. Both
TopLine and Lines must be specified as a non-interlaced scan line (i.e. must be ←↩
   divided by 2 if an
interlaced screen).

INPUTS
TopLine   -Specifies the first line to start clearing HIRESDISP bits.
Lines   -Number of lines to modify.

RESULT
None

CONSIDERATIONS

SEE ALSO
SetHires24 ()
```

1.55 opal.library/SetPFStencil24

```
                NAME
SetPFStencil24 -- Sets the PlayField Stencil of the specified Screen.

SYNOPSIS
void SetPFStencil24 (OScrn);
         A0

struct OpalScreen *OScrn;

FUNCTION
Sets the playfield stencil (least significant bit of green bank 0) of all of the ←↩
    pixels in the specified screen.

INPUTS
OScrn = OpalScreen structure.

RESULT
None

CONSIDERATIONS
This will only have an effect if Dual Playfield mode has been set up using  ←↩
   DualPlayField24().

SEE ALSO
ClearPFStencil()
DualPlayField24()
```

```
               SinglePlayField24()
```

1.56 opal.library/SetPRStencil24

```
                NAME
SetPRStencil24 -- Sets the Priority Stencil of the specified Screen.

SYNOPSIS
void SetPRStencil24 (OScrn);
         A0

struct OpalScreen *OScrn;

FUNCTION
Sets the priority stencil (least significant bit of blue bank 0) of all pixels ←
   in the specified screen.

INPUTS
OScrn = OpalScreen structure.

RESULT
None

CONSIDERATIONS
This will only have an effect if dual OpalVision/Amiga display mode has been set ←
    up using

             DualDisplay24()
             .

SEE ALSO
ClearPRStencil()

             DualDisplay24()

             SingleDisplay24()
```

1.57 opal.library/SetRGB24

```
NAME
SetRGB24 -- Updates a single palette entry to the OpalVision palette registers.

SYNOPSIS
void SetRGB24 (Entry, Red,  Green,  Blue);
    D0    D1    D2      D3

long Entry;
long Red;
long Green;
long Blue;
```

```
FUNCTION
This function updates a single palette entry in the OpalVision palette registers ↩
    .

INPUTS
Entry - The entry selected for update (0-255).
Red - Red value (0-255).
Green - Green value (0-255).
Blue  - Blue value (0-255).

RESULT
None

CONSIDERATIONS
This function will only have a visible effect when in palette mapped mode.

SEE ALSO
PaletteMap ()
SetPalette ()
```

1.58 opal.library/SetScreen24

```
                NAME
SetScreen24 -- Fills screen with a specified colour.

SYNOPSIS
void SetScreen24 (OScrn)
        A0

struct OpalScreen *OScrn;

FUNCTION
This function is similar to ClearScreen24, but fills the screen with the colour  ↩
    contained in Pen_R,Pen_G
& Pen_B in the OpalScreen structure.

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO

              ClearScreen24()
```

1.59 opal.library/SetSprite24

```
                NAME
SetSprite24 -- Allows Amiga sprites to be displayed over OpalVision graphics.

SYNOPSIS
void SetSprite24 (SpriteData, SpriteNumber);
      A0        D0

USHORT *SpriteData;
long SpriteNumber;

FUNCTION
This function allows Amiga hardware sprites to be displayed in OpalVision  ↩
   graphics.

Sprites are displayed during both display and update cycles and due to this the  ↩
   sprite data is written into
the frame buffer memory along with the video data. This may be undesirable in  ↩
   some cases, so the sprite
may be removed before starting updates using
              Refresh24()
               or
              UpdateDelay24()
               by calling SetSprite24()
with SpriteData = NULL

SpriteData  is a pointer to a data definition of a spite as passed to the  ↩
   SetPointer() function in the
intuition library. The SpriteNumber is the hardware sprite number to be used to  ↩
   display the sprite, this
will normally be 0 to modify the mouse pointer sprite.

N.B.  Passing -1 for the SpriteData will use the currently active Amiga Sprite  ↩
   in the system. For
example SetSprite24((USHORT *) - 1,0) will allow the currently active mouse  ↩
   pointer to be displayed
over the 24 bit image.

INPUTS
SpriteData  - pointer to the sprite data.
SpriteNumber  - Amiga hardware sprite number (0...7).

RESULT
None

CONSIDERATIONS
All sprites other than the mouse pointer sprite should be allocated by the user  ↩
   using GetSprite () as sprite
0 is normally used for the mouse pointer and another sprite is required by the  ↩
   opal library for normal
screen updates.

Sprites will only be visible during display cycles if Amiga priority is set and  ↩
   dual display mode is active.

SEE ALSO
```

1.60 opal.library/SingleDisplay24

```
                NAME
SingleDisplay24 -- Sets up an Amiga/OpalVision single display.

SYNOPSIS
void SingleDisplay24 (void);

FUNCTION
This function sets the DUALDISPLAY bit of all CoPro instructions, allowing an  ↩
   OpalVision or Amiga
only display.

INPUTS
None

RESULT
None

CONSIDERATIONS
If Amiga display bottom has been set, using the
              SetDisplayBottom24()
               the CoPro instructions will not be
modified for that region of the display.

SEE ALSO

              DualDisplay24()
```

1.61 opal.library/SinglePlayField24

```
NAME
SinglePlayField24() -- Sets up an Amiga or OpalVision single playfield.

SYNOPSIS
void SinglePlayField24 (void)

FUNCTION
This function clears the DUALPLAYFIELD bit of all coprocessor instructions,  ↩
   Allowing only one of
OpalVision playfield to be displayed.

INPUTS
None

RESULT
None

CONSIDERATIONS
If an Amiga display bottom has been set, the coprocessor instructions will not  ↩
   be modified for that region
of the display.
```

SEE ALSO
DualPlayField24()

1.62 opal.library/StopUpdate24

```
                NAME
StopUpdate24 -- Stops updates to the frame buffer memory.

SYNOPSIS
void StopUpdate24 (void);

FUNCTION
This function stops updates to the OpalVision frame buffer memory initiated with ←
    a call to

              UpdateDelay24()
              . This allows changes to be made to the Amiga memory without  ←
                  affecting the
OpalVision frame buffer memory thus offering inherent double buffering. Stopping ←
    updates will also
reduce the DMA load on the Amiga.

INPUTS
None

RESULT
None

CONSIDERATIONS
This function may take up to 1 frame as it must wait for the current frame  ←
   update to be completed.

SEE ALSO

              UpdateDelay24()

              Refresh24()
```

1.63 opal.library/UpdateAll24

```
                NAME
UpdateAll24 -- Resets the internal update structure so all required banks are  ←
   updated.

SYNOPSIS
void UpdateAll24 (void);

FUNCTION
Resets the internal update structure so that all required banks are updated.   ←
   This function is useful after a
call to
```

```
              UpdatePFStencil24()
               to reinitialise the internal state of the library so that all the ↩
                   required segments
are updated correctly on subsequent calls to
              Refresh24()
               or
              UpdateDelay24()
              .

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO

              UpdatePFStencil24()
```

1.64 opal.library/UpdateCoPro24

```
                NAME
UpdateCoPro24() -- Writes CoPro list for the current display screen to Video  ↩
   coprocessor

SYNOPSIS
void UpdateCoPro24 (void);

FUNCTION
Encodes the entire CoPro instruction list from the displayed screen structure  ↩
   and initiates a coprocessor
update.

This function also updates the Load Address Register.

INPUTS
None

RESULT
None

CONSIDERATIONS
Modifying the coprocessor list in the screen structure does not have any effect  ↩
   on the display until
UpdateCoPro24 () is called.

The CoPro list will not be updated in the OpalVision until the next vertical  ↩
   blanking period.

SEE ALSO
```

```
              SetCoPro24()

              RegWait24()
```

1.65 opal.library/UpdateDelay24

```
                NAME
  UpdateDelay24 () -- Sets the delay between consecutive frame buffer updates.

  SYNOPSIS
  void UpdateDelay24 (FrameDelay);
        D0

  long FrameDelay;

  FUNCTION
  This function allows a variable frame delay between consecutive frame buffer ↩
     updates. Setting a frame
  delay of zero enables continuous full speed updates.

  This function also initiates continuous updates to the OpalVision frame buffer ↩
     memory which will
  continue until either
              Refresh24()
               or
              StopUpdate24()
               is called.

  Setting a delay increases free bus DMA bandwidth to increase performance of the ↩
     CPU and other DMA
  devices.

  INPUTS
  FrameDelay  = Number of Frames to pause between frame buffer updates.

  RESULT
  None

  CONSIDERATIONS

              UpdateAll24()
               and
              UpdatePFStencil24()
               determine which memory segments will be updated during an
  update sequence.

  SEE ALSO

              StopUpdate24()

              UpdateAll24()

              UpdatePFStencil24()
              Refresh24()
```

1.66 opal.library/UpdatePalette24

```
NAME
UpdatePalette 24 -- Loads all 256 entries of Red, Green and Blue values in the  ↩
   OpalScreen structure onto
    the OpalVision  palette registers.

SYNOPSIS
void UpdatePalette (void);

FUNCTION
Updates the OpalVision palette registers with the palette values in the  ↩
   OpalScreen structure.

This also updates the Pixel Read mask and the Command Register and uses the  ↩
   Palette Load Address as
an offset for the palette update.

INPUTS
None

RESULT
None

CONSIDERATIONS
Updates will have no effect in non palette mapped modes.

SEE ALSO
SetRGB24 ()
PaletteMap24 ()
```

1.67 opal.library/UpdatePFStencil24

```
                NAME
UpdatePFStencil24() -- Updates playfield stencil at highest possible rate.

SYNOPSIS
void UpdatePFStencil24(void);

FUNCTION
Enables updates to only the segments containing the playfield stencil (green  ↩
   segments). The speed of
update is three times that of a normal 24bit update. This enables quick  ↩
   playfield transitions.

This function does not update the playfield stencil as such, but modifies the  ↩
   internal state of the library so
that subsequent calls to
              Refresh24()
```

```
               or
              UpdateDelay24()
               will only update the segments containing the
playfield stencil. The internal state of the library can be returned to normal  ←
   by calling
              UpdateAll24()
              .

To use the playfield stencil in 8bit mode, the green bank contains the stencil  ←
   and therefore must be
updated. The most convenient way to do this is to write the frame for the first  ←
   playfield in the red
segment of bank 0 using WriteFrame24(0) and the second playfield into the red  ←
   segment of bank 1 using
WriteFrame24(3). UpdatePFStencil24() will call WriteFrame24(1) when in 8bit mode ←
    to switch to the
green segment. Placing you playfields in segments other than the green segment  ←
   will give you the full
256 colours rather than 128.

INPUTS
None

RESULT
None

CONSIDERATIONS


SEE ALSO
UpdateDelay24 ()
DualPlayField24()

              SinglePlayField24()

              Refresh24()

              UpdateAll24()
```

1.68 opal.library/UpdateRegs24()

```
                NAME
UpdateRegs24 () -- Updates the hardware registers for the current display screen

SYNOPSIS
void UpdateRegs24 (void)

FUNCTION
Updates the Pixel Read mask, Command register and Palette Load Address registers ←
    in the OpalVision
with the values from the current display screen structure (See "Registers").

INPUTS
None
```

```
RESULT
None

CONSIDERATIONS
Changing register values in the screen structure does not take effect until an ↩
   update has been initiated.

Register updates are not completed until the next vertical blanking period.

SEE ALSO

              RegWait24()
```

1.69 opal.library/WriteFrame24

```
                NAME
WriteFrame24 -- Sets the current frame to be written to within the frame buffer ↩
   memory.

SYNOPSIS
void WriteFrame24 (Frame);
       D0

long Frame;

FUNCTION
Depending on the resolution of the displayed OpalVision screen, a number of ↩
   screens can be stored in the
frame buffer memory. The number of frames available for the screens resolution ↩
   are given in the
MaxFrames variable in the OpalScreen structure.

WriteFrame24() allows each individual frame to be written separately where Frame ↩
    is in the range
0...MaxFrames. Using a combination of WriteFrame24 and DisplayFrame24, it is ↩
   possible to store
several images in frame buffer memory and to perform simple page flip animation.

INPUTS
Frame = Frame number to display (0...MaxFrames).

RESULT
None

CONSIDERATIONS
The display frame and the write frame, must reside in the same field area in the ↩
    frame buffer memory.
(See "Memory Segment Diagram"). Due to this WriteFrame24() has the side effect ↩
   of changing the
display frame if the new write frame is in a different field.

SEE ALSO
```

```
                DisplayFrame24()
```

1.70 opal.library/WritePFPixel24

```
                  NAME
  WritePFPixel24 () -- Set or clear a pixel in the playfield stencil.

  SYNOPSIS
  Result = WritePFPixel24 (OScrn, x, y);
  D0       A0 D0 D1

  struct OpalScreen *OScrn;
  long x;
  long y;
  long Result;

  FUNCTION
  Sets or clears a pixel in the playfield stencil depending on the state of Pen_R  ↩
     in the screen structure. If
  Pen_R is = 0, the pixel will be cleared else it is set. The SetPFPen macro can  ↩
     be used to initialize Pen_R.

  INPUTS
  OScrn = A pointer to an OpalScreen structure.
  x = x coordinate of pixel.
  y = y coordinate of pixel.

  RESULT
  Result  = -1 if the pixel is outside the clip boundary else 0.

  CONSIDERATIONS
  This function only has visible effect in dual playfield mode.

  SEE ALSO

                ReadPFPixel24()
                  UpdatePFStencil()
  DualPlayField24()
  SinglePlayfield24()
```

1.71 opal.library/WritePixel24

```
                  NAME
  WritePixel24 () -- Write a pixel into an OpalScreen.

  SYNOPSIS
  Result = WritePixel24 (OScrn, x, y);
  D0           A0     D0 D1
```

```
struct OpalScreen *OScrn;
long x;
long y;
long Result;

FUNCTION
Writes a pixel in the specified OpalScreen using the current pen value in that structure. The macro
SetPen can be used to initialize pen values correctly.

INPUTS
OScrn = A pointer to an OpalScreen structure.
x = x coordinate of pixel.
y = y coordinate of pixel.

RESULT
Result  = -1 if the pixel is outside the clip boundary else 0.

CONSIDERATIONS


SEE ALSO

              ReadPixel24()
```

1.72 opal.library/WritePRPixel24

```
                NAME
WritePRPixel24 () -- Set or clear a pixel in the priority stencil.

SYNOPSIS
Result = WritePRPixel24 (OScrn, x, y);
D0       A0 D0 D1

struct OpalScreen *OScrn;
long x;
long y;
long Result;

FUNCTION
Sets or clears a pixel in the priority stencil depending on the state of Pen_R in the screen structure. If
Pen_R is = 0, the pixel will be cleared else it is set. The macro SetPRPen can be used to initialize the
state of Pen_R.

INPUTS
OScrn = A pointer to an OpalScreen structure.
x = x coordinate of pixel.
y = y coordinate of pixel.

RESULT
Result  = -1 if the pixel is outside the clip boundary else 0.
```

```
CONSIDERATIONS
This function only has visible effect when the priority stencil is enabled.

SEE ALSO

              ReadPRPixel24()

              EnablePRStencil24()

              DisablePRStencil24()
```

1.73 opal.library/WriteThumbnail24

```
                NAME
WriteThumbnail24 -- Writes an IFF thumb-nail chunk into a file.

SYNOPSIS
ReturnCode = WriteThumbnail24 (OScrn, File);
D0         A0    A1

struct OpalScreen *OScrn;
BPTR File;
long ReturnCode;


FUNCTION
This function generates a 24 bit thumb-nail for the given OpalScreen and writes  ↩
   an IFF OVTN thumb-
nail chunk into the given file.

INPUTS
OScrn = OpalScreen to generate the thumb-nail for.
File  = File Handle of the file to write thumb-nail to.

RESULT
ReturnCode = 0 if all ok, or an OpalVision Error code  if en error occurred.

CONSIDERATIONS


SEE ALSO

              SaveIFF24()
                LoadIFF24()
```

1.74 opalreq.library/OpalRequester

```
NAME
OpalRequester -- The OpalVision file requester.
```

```
SYNOPSIS
ReturnCode = OpalRequester (OReq);
D0          A0

struct OpalReq *OReq;

FUNCTION
This is the entry point for the OpalVision requester. OReq is a pointer to a  ↩
   properly initialised OpalReq
structure defined in the above sections. The requester will be displayed and  ↩
   handled completely by the
library, when the user has selected a file or hit cancel, this function will  ↩
   return the selected file and
directory name in OReq. OKHit will be cleared if the user hit the Cancel gadget  ↩
   and set otherwise.

INPUTS
OReq  = A pointer to a correctly initialised OpalReq structure.

RESULT
ReturnCode = 0 if all ok
     =  OR_ERR_OUTOFMEM if there is not enough memory to display requester
     =  OR_ERR_INUSE if the requester is currently in use.

CONSIDERATIONS
For this release the intuition screen used to display the requester
must be hires interlaced, and have atleast 2 bit planes.

SEE ALSO
```