# Black Video Software Functional Specification

```
 ------------------------------------------
| Drawing No : 1303/005/FS/3P95             |
|      Issue : 1.04/3P95                    |
|       Date : 22/11/94                     |
|    Authors : Alan Glover                  |
|            : Graham Simms                 |
|     Sheets :                              |
|  Last Issue: None                         |
 ------------------------------------------
```

Contents

## 1. History

1.04 MRC 21/3/95 Updated for Developer pack.

## 2. Outstanding Issues

None.

## 3 Overview

The work discussed here for Black consists of one major change: the adoption of a new SpriteExtend which adds JPEG display capability.

The new SpriteExtend provides a faster plotting engine for PutScaled, and can also handle JPEG images (although their interface is by means of separate SWI calls).

Although the new SpriteExtend is responsible for providing the JPEG support; its API will reinforce that it is separate from the sprite system by using its own SWI prefix. This approach has been chosen to avoid causing further distortion and complication to the range of SpriteOps by introducing another type which only certain operations can manipulate.

The remainder of the work detailed here is concerned with filling out areas of development which were deferred on Medusa, such as extending the implementation of 1bpp masks to all sprite operations and introducing palettes on new format sprites.

## 4 Technical Background - JPEG and JFIF

JPEG is an internationally standardised data format for the lossy compression of photographic data, capable of displaying screen-resolution colour images for about 1.5-2.5 bits per pixel.

JPEG files encode colour pictures as YUV (Y=intensity, U and V are colour) data. Compressing involves the following steps:

Convert RGB data to YUV.
Throw away 3 out of 4 of the U and V pixels
Convert 8*8 tiles of Y, U and V values through a Discrete Cosine
  Transform, into an 8*8 square of frequency coefficients.
Discard coefficients which are zero, or close to zero. This will
  tend to change the visual appearance of the picture very little
Reduce the accuracy with which the remaining coefficients are held
  (known as 'quantisation'). Again, this changes the appearance

very little. The amount by which this is done, controls the compression factor of the image. By now, most of the coefficients will be zero.

Reorder the 64 coefficients in a zig-zag order, which increases the average length of runs of zeros in the coefficient block.

Huffman-encode the resulting stream of values.

Decompression involves reversing these.

JFIF is a subset of JPEG defined by C-Cube and widely used for the simple interchange of JPEG data - JPEG itself allows many bizarre parameters and combinations, which JFIF limits to more reasonable proportions. When people talk about 'JPEG files', they usually mean JFIF files. We support JFIF 1.02.

JPEG - the standard itself - ISO/IEC JTC1/SC2/WG10

The JPEG Still Picture Compression Standard, Gregory K Wallace, CACM April 1991.

JPEG File Interchange Format (JFIF), version 1.02, Eric Hamilton,
C-Cube Microsystems, 1778 McCarthy Blvd, Miltipas, CA 95035
(eric@c3.pla.ca.us)

## 5 User Interface

### 5.1 !Paint

Work on Paint will be fully detailed in a separate FS (1303,009/FS).

### 5.2 !Draw

The provision of a facility in !Draw/Draw for it to support JPEG objects is fully detailed in a separate FS (1303,010/FS).

## 6 Programmer Interface

### 6.1 Support for 1bpp masks

The following SpriteOps which were not included in the Medusa 1bpp mask coding will be done this time round, completing the development.

| | |
|---|---|
| 31 | Insert Row (*) |
| 32 | Delete Row (*) |
| 33 | X axis flip |
| 35 | Append Sprite (#) |
| 45 | Insert Column (*) |
| 46 | Delete Column (*) |
| 47 | Y axis flip |
| 57 | Insert/Delete rows |
| 58 | Insert/Delete columns |

* These routines will handle 1bpp masks by calling SpriteExtend and performing SpriteOp 57/58 as appropriate. Although this is an implementation detail, it will be acceptable to remove the routines entirely and make the whole call a SpriteExtend facility.

# An error will be raised if an append involving one sprite with a 1bpp mask and another with a normal mask is attempted.

### 6.2 Palettes on New format sprites

Palettes will remain prohibited on 16/32bpp sprites. For 8bpp and below a palette will be introduced. This will be of the same format as the palette on old format sprites. Implementing this is therefore chiefly a matter or relaxing the tests introduced in Medusa to prohibit palettes.

This will have a direct effect on these SpriteOps which may create sprites with palettes where they are allowed.

| | |
|---|---|
| 2 | Screen save |
| 15 | Create Sprite |
| 37 | Create/Remove Palette (SpriteExtend) |

### 6.3 Wide Translation Tables to PutSpriteScaled and PlotSpriteTransformed

This affects these SpriteOps

| | |
|---|---|
| 52 | PutSpriteScaled |
| 56 | PlotSpriteTransformed |

R5 bit 5 will be introduced as a new flag on all these calls. When clear the translation table is byte wide, which is the current situation. When set the translation table is byte wide for 8bpp and below, two bytes wide plotting into 16bpp and four bytes wide for plotting into 32bpp. These sizes correspond to the behaviour of ColourTrans_SelectTable with R5 bit 4 set.

### 6.4 SpriteExtend plotting directly from the palette with 8bpp full palette sprites.

This behaviour will be removed; all SpriteExtend plots with R5 bit 4 clear will expect and use a translation table. The action of R5 bit 4, which tells SpriteExtend to go directly to the palette for use in special cases will remain.

Under both OS 3.50 and the Black OS callers wishing to plot from the palette should be advised to set bit 4 of R5 - even if it is an 8bpp full palette sprite.

### 6.5 Printing

The elements of the printing edifice - !Printers and the various modules underneath it - will need to reflect the other changes to the video system discussed here. 1bpp mask operations and palettes are unlikely to cause any significant work.

However, the same cannot be said for JPEG and a feasability study is needed by an expert within the area. The outcome of this will appear in the FS for work on the printing system (1303,007/FS).

### 6.6 T=9 JPEG sprites

Sprite Type 9 was provisionally reserved for JPEG sprites in the

Medusa specifications. It has been decided to revoke this, and to treat JPEG images in their own right using their own existing filetype.

### 6.7 JPEG SpriteExtend: New PutSprite/MaskScaled code

In addition to JPEG support, the new SpriteExtend provides faster plotting for existing sprite images. However, this requires exhaustive development testing and verification. The following areas, at a minimum, should be tested:

For Sprite, Mask and Sprite+Mask plots (with and without translation table):

- Clipped plots - test all possibilities.

A program will be used to move the sprite around the screen, plotting its centre at the screen edge and moving the sprite all around the edges of the screen graphics area.

A program will be used to set various possibilities of the graphic window size to test clipped plots not testable by the above (eg only plotting a central rectangle of the sprite).

A program will be used to detect whether the plotting engine accesses beyond the end of the sprite data when plotting an image (this has been an infamous source of bugs in the past!).

- Screen modes

These checks will be carried out in all display depths, and in single and double-pixel screen modes. Further, its operation will be tested in all permutations of XEIG 0-3 and YEIG 0-3.

- With output switched to a sprite instead of screen

Using sprites for all the screen permutations.

- Comparitive tests

Test plots will be made using the new SpriteExtend, screensaved, and then compared with the same plot produced by the old SpriteExtend. Identical results should be expected since the intention is that the scaling routines remain the same.

### 6.8 JPEG Support

SWI Chunk = &49980

Originally this was implemented by putting the JPEG data within a T=9 new format sprite. However, it has been decided that this makes the sprite system too unwieldy (since quite a number of SpriteOps would have to fault these images - you cannot, for instance, add a row or column).

The approach we will take appears as six SWI calls. Although they have a close relationship with sprite operations, it has been decided not to class these calls as sprite operations since it would be blurring the boundary between sprites and JPEG images.

The six SWI calls are JPEG_Info/_FileInfo, JPEG_PlotScaled/_PlotFileScaled and JPEG_PlotTransformed/_PlotFileTransformed. Their parameters are detailed in the following sections. The SWI prefix was chosen to ensure possible conflicts with external modules is avoided.

Two possible strategies exist for printing JPEG images. The first is that used for SpriteOps - a vector. The second is a rebound method such as the Font Manager uses. The decision on this is left to the specification of the printing support - 1303,007/FS.

A seventh SWI, JPEG_PDriverInterface, is provided purely for use by the printing system so that it can intercept JPEG plots when output is being sent to the printer.

### 6.9 JPEG_Info (&49980)

This is based upon the SpriteOp 65 code in the prototype code, but with changes to the register allocation to reflect R0 becoming usable.

Entry:

```
R0 = flags for desired operation
     b0 set: return dimensions
R1 = pointer to JPEG file in memory
R2 = length of data in memory (bytes)

     all other bits of R0 reserved; set to 0
```

Exit:

```
R0 returned information flags
   b0: set if greyscale image (colour if clear)
   b1: set if transformed plots are not supported
   b2: set if pixel density is a simple ratio
       clear if pixel density is in DPI
R1 preserved
R2 width in pixels
R3 height in pixels
R4 x pixel density
R5 y pixel density
R6 SpriteExtend memory requirements to plot JPEG.
    0: No extra memory required.
```

If the image does not appear to be valid data an error will be returned with VS and R0 pointing to an error block.

This call will not do a full validation of the data, but it checks the header enough to return the width and height. If it returns with VC the caller may proceed on the basis that this is supposed to be JPEG data.

Reserved bits are intended to allow more information to be returned in the future.

### 6.10 JPEG_FileInfo (&49981)

This is based upon the SpriteOp 65 code in the prototype code, but with changes to the register allocation to reflect R0 becoming usable.

Entry:

```
R0 = flags for desired operation
     b0 set: return dimensions
R1 = pointer to control character terminated filename for JPEG image.

     all other bits of R0 reserved; set to 0
```

Exit:

```
R0 returned information flags
   b0: set if greyscale image (colour if clear)
   b1: set if transformed plots are not supported
   b2: set if pixel density is a simple ratio
       clear if pixel density is in DPI
R1 preserved
R2 width in pixels
R3 height in pixels
R4 x pixel density
R5 y pixel density
R6 SpriteExtend memory requirements to plot JPEG.
    0: No extra memory required.
```

If the image does not appear to be valid data an error will be returned with VS and R0 pointing to an error block.

This call will not do a full validation of the data, but it checks the header enough to return the width and height. If it returns with VC the caller may proceed on the basis that this is supposed to be JPEG data.

Reserved bits are intended to allow more information to be returned in the future.

## 6.11 JPEG_PlotScaled (&49982)

```
R0 = pointer to JPEG file loaded in memory
R1 = x coordinate for plot
R2 = y coordinate for plot
R3 = scale factors or 0
R4 = length of data in memory (bytes)
R5 = Flags
      b0 set: dither output when plotting 24 bit JPEG
      at 16bpp or below
      All other bits are reserved. Set to 0
```

All registers preserved on exit

The scale factors pointer in R3 has the same function as PutSpriteScaled; it is a four word block consisting of x multiplier and divisor and y multiplier and divisor.

The x and y co-ordinates have the same meaning as SpriteOp PutSpriteScaled.

This SWI acts as PlotFileScaled (below) except that it works with a file which has already been loaded into memory (with the subtle side effect that the memory allocation for this becomes the caller's responsibility, rather than SpriteExtend's).

The following errors may be raised:

Too little memory to plot this JPEG image - obvious!

Unrecognised JPEG data - multitudinous potential causes; the data is wrong, the data overruns the end of the file, etc.

## 6.12 JPEG_PlotFileScaled (&49983)

```
R0 = pointer to control character terminated filename for JPEG image
R1 = x coordinate for plot
R2 = y coordinate for plot
R3 = scale factors or 0
R4 = Flags
      b0 set: dither output when plotting 24 bit JPEG
      at 16bpp or below
      All other bits are reserved. Set to 0
```

All registers preserved on exit

The scale factors pointer in R3 has the same function as PutSpriteScaled; it is a four word block consisting of x multiplier and divisor and y multiplier and divisor.

The x and y co-ordinates have the same meaning as SpriteOp PutSpriteScaled.

This SWI decompresses and plots the JPEG file on the screen using the same scaling algorithms as PutSpriteScaled. A direct plot is the only form supported (ie plot action 0 of PutSpriteScaled).

The file is not cached, ie memory will be claimed and released by this call.

The following errors may be raised:

Insufficient memory to plot this JPEG image - obvious!

Unrecognised JPEG data - multitudinous potential causes; the data is wrong, the data overruns the end of the file, etc.

## 6.13 Internationalisation

The new SpriteExtend code and errors require internationalising.

## 6.14 SpriteOp PutSpriteGreyScaled

This sprite operation has not been ported to the new SpriteExtend, in the belief that the call is not used.

A search will be made of all RISC OS sources to ensure that this is the case, and if so an announcement will be made to developers via Developer News to indicate that we intend to declare this call obsolescent and that it may disappear in future releases of RISC OS.

Subject to a satisfactory (lack of) response to both these initiatives code will be put in place to raise the error 'Sprite Operation 'PutSpriteGreyScaled' is not supported in this version of RISC OS'.

In the event that the operation is still required by something, and assuming that we decide the reason is important enough to retain the operation, then further porting work to C will need to be done since PutSpriteGreyScaled has not been done yet and is closely interwoven with Put  riteScaled.

## 6.15 SpriteOp CheckSpriteArea

A central facility for checking the validity of a sprite area will be provided. It is expected that applications will call this once after loading a sprite file to reassure themselves about the file. The checks outlined here were originally proposed for incorporation into Paint.

Other SpriteOps will --not-- make calls to this facility; it would slow all the calls down, and be unneccessary providing an application has already checked the whole area.

OS_SpriteOp 17 (&11, SWI &2E)

On Entry

R0: 17 (&11)
R1: pointer to control block of sprite area

On Exit

Either:

R0, R1 preserved, VC

- nothing wrong was detected

R0->Error block, R1 preserved, VS

- a problem was found, the error 'Unrecognised Sprite Data' is returned (error number &720)

The following checks are performed on the sprite area:

```
    The offset to the first sprite must lie within the "used" part of the
        sprite area
    The offset to the free area must lie within the sprite area
    FOR each sprite
    DO  the offset to the next sprite must lie within the "used" part of
            the sprite area
        the first bit used must be in the range [0; 32) (0 for a new
            format sprite)
        the last bit must lie in the range (0; 32]
        the offset to the image must lie within this sprite
        the offset to the mask must lie within this sprite
        the size of an image with this many bpp and this width and heigh
t
            must fit within the space allowed for the image
        the size of a mask with this many bpp (1 for new format) and thi
s
            width and height must fit within the space allowed for the
            mask
    OD
```

All offsets must be word aligned.

If any of these tests fail, an error will be returned. Note that the mode number is not checked: sprites with illegal modes are explicitly allowed, since they can usefully occur in sprite files.

Where an unknown mode number is encoutered the image data checks will be that:

i) there is sufficient space for the image assuming 1bpp data
ii) (if there is a mask) that the image size is >= mask size

These checks do not include facts which are "usually true" for sprites, but are not guaranteed by the sprite file definition: in particular, they allow an extension area, and they allow palettes of any size.

## 6.17 JPEG_PlotTransformed (&49984)

```
R0 = pointer to JPEG file loaded in memory
R1 = flag word
     bit 0 set: R2 is pointer to dest co-ords, else pointer to matrix
     bit 1 set: dither output when plotting 24 bit JPEG at 16bpp or below
     All other bits reserved. Set to 0
R2 = pointer to matrix (as for Draw module), or
     pointer to destination co-ordinate block, depending on R1:b0
R3 = length of data in memory (bytes)
```

All registers preserved on exit.

This call is NOT being implemented fully in Black. All it will do is test the matrix/destination co-ordinate block to determine whether the plot is a simple scaling with no rotation or other transformation involved. If this is true the plot will be performed by the PlotFileScaled code.

In all other cases the error:

Transformed JPEG plotting is not supported by this version of
the SpriteExtend module

will be returned.

## 6.17 JPEG_PlotFileTransformed (&49985)

```
R0 = pointer to control character terminated filename for JPEG image
R1 = flag word
     bit 0 set: R2 is pointer to dest co-ords, else pointer to matrix
     bit 1 set: dither output when plotting 24 bit JPEG at 16bpp or below
     All other bits reserved. Set to 0
R2 = pointer to matrix (as for Draw module), or
     pointer to destination co-ordinate block, depending on R1:b0
```

All registers preserved on exit.

This call is NOT being implemented fully in Black. All it will do is test the matrix/destination co-ordinate block to determine whether the plot is a simple scaling with no rotation or other transformation involved. If this is true the plot will be performed by the PlotFileScaled code.

In all other cases the error:

Transformed JPEG plotting is not supported by this version of
the SpriteExtend module

will be returned.

### 6.18 JPEG_PDriverIntercept (&49986)

On Entry:

```
R0 = b0: Intercept state; 0 = off, 1 = on
     b1: Always use translation tables when plotting sprites.
     b2-31: reserved; Set to 0
```

R0 Returns previous intercept state

All other registers preserved on exit.

This SWI will be called by the printer drivers in order to inform SpriteExte
|nd
whether JPEG plotting calls should be handed on via the SWI_PDriver_JPEGSWI
|(see
below). JPEG calls are handed on when the intercept state is defined as on.

The only JPEG calls which are affected by the intercept state are the plotti
|ng
calls; namely:

```
  JPEG_PlotScaled
  JPEG_PlotFileScaled
  JPEG_PlotTransformed
  JPEG_PlotFileTransformed
```

### 6.19 Support for dithering 'truecolour' images

SpriteExtend will have the facility to support dithering of both JPEG images and Sprites. The relevant JPEG SWIs all have a flag bit to enable dithering to be switched on, dithering will be extended to Sprites when calling PlotSpriteScaled or PlotSpriteTransformed.

The dithering will be off by default, and only enabled when bit 6 of R5 is set.

Dithering will only affect the plotting of 16/32bpp sprites to a reduced depth. If dithering is turned on for any other plots then it will have no effect.

### 6.20 Error diffused dithering of JPEGs

The JPEG plotting routines used within SpriteExtend have an optimised mode when decompressing a JPEG into an 8bpp mode with the standard palette. This uses a limited error diffusion technique which vastly improves the appearance of the image. The real time error diffusion will only work on JPEGs which have been compressed in a certain way, which could lead to a significant difference in display quality between two apparently similar JPEGs.

In order to use the enhanced dithering, JPEGs must have been compressed using an X and Y sample size of 2. This is compatible with the JPEGs produced by the official Independant Group's software and the new version (1.03+) of !ChangeFSI.

## 7 Data Formats

### 7.1 JPEG Image

This is a file of filetype &C85 "JPEG", containing data conformant with the JFIF specification - this is an existing usage of the filetype supported by ChangeFSI in Medusa and earlier platforms. The major change is that it will now have OS support, and the sprite for it should therefore be included in the Wimp's ROM sprite pool. The proposed sprite is the one that has been used in Medusa.

### 7.2 Palettes on new format sprites

Palettes will be allowed on new format sprites of 8bpp and below. This will take the same format as old format sprites; ie for each palette entry there are two words &bbggrr00 representing the two flash states for the colour. Usually they will be identical.

Where an 8bpp sprite is created in a screen mode with a full palette the sprite will have 256 pairs of entries in the palette data.

## 8 Protocols

No new protocols introduced.

## 9 External Dependencies

All development described in this FS may be tested on a Medusa platform.

## 10 Development Test Strategy

Tests during coding will take the form of:

a) SWI level comparisons
b) Stress testing

Where such tests are of lasting usefulness they will be created/ documented to enable future use during the Audit phase of development.

The new SpriteExtend must have thorough acceptance and validation testing - details are contained in section 6.7 above.

### 10.1 SWI level comparisons

Each individual SWI call will be tested with each possible combination of parameters, where possible, or a large number of combinations where not.

Comparitive testing of changes is largely impossible, since all but one of the changes above represent the addition of new features.

The exception is SpriteOp PutSpriteScaled which is presently provided by SpriteExtend and will in future be provided by the JPEG-capable SpriteExtend. Work is needed on the JPEG-capable SpriteExtend before it can be effectively trialled, however it is expected that the performance of the JPEG-capable SpriteExtend will be better than the current SpriteExtend for PutSpriteScaled. Equal or slower performance is unacceptable from the JPEG-capable SpriteExtend.

The SWI test harness will be used for this method of testing wherever possible. This will result in scripts which may easily be used in future, or run intensively for 'bashing' purposes.

For testing user interfaces Mouse Recorder scripts will be used where it can be reasonably sure that the effort of their creation will not be invalidated at some future point.

Where the SWI test harness and Mouse Recorder prove unsuitable specific programs will be written. These will be written and documented in a manner which makes them suitable for future use.

### 10.2 Stress Testing Programs

These will be variants of the test programs outlined above which will trial routines in an intensive manner to provide assurance that performance under worst case or invalid scenarios is acceptable.

### 10.3 Application of test methods

Each video SWI affected by this Functional Spec (see below for list) will be tested with various option combinations. Where the number of permutations is small all permutations will be tested. When the number of permutations grows too large to cover every possible permutation specific attention will be given to:

* Often used combinations
* Stressful combinations
* Invalid combinations

The definitions of the above terms in relation to a specific SWI will be determined by the developer concerned.

All video SWIs will be tested with parameters groups which are:

* Wildly invalid
* Just invalid
* Just valid
* In the middle of valid range

Where there is no invalid range stressful situations will be used instead.

### 10.4 SWIs directly affected by the work in this FS

OS_SpriteOp (all changed reason codes)
JPEG (SpriteExtend) (all reason codes)

### 10.5 Specific tests for SpriteOp CheckSpriteArea

The following test files will be used:

An empty sprite area
One sprite
Multiple sprites, exercising all permutations of depth and mask/palette presence

Using these files the following tests will be carried out:

Is a valid file approved
Is a violation of each of the criteria above detected, when
i) it is on the first sprite in the area
ii) it is on the last sprite in the area
iii) it is on an intermediate sprite in the area
Are multiple violations coped with properly (ie it should stop at the first violation rather than proceeding)

Finally, a sample of sprite files from common applications will be tested using a version of Paint modified to use this call.

## 11 Organisation

The JPEG-SpriteExtend will be in the ROM. Other existing items of code retain their usual position.

### 11.1 Code Size Target

Kernel size change due to work in this specification should be negligible. It is anticipated that the removal of the kernel's Insert/Delete Row/Column routines will finance other areas of growth.

SpriteExtend is a different story however. The inclusion of new JPEG code, plus the growth due to adding 1bpp mask support to Append Sprite and Insert/Delete Rows/Columns all point to a heavy hit on code size. It is likely that the eventual SpriteExtend image will be around 70K.