ColourTrans

Introduction

ColourTrans allows a program to select the physical red, green and blue colours that it wishes to use, given a particular output device and palette. ColourTrans then calculates the best colour available to fit the required colour.

Thus, an application doesn't have to be aware of the number of colours available in a given mode.

It can also intelligently handle colour usage with sprites and the font manager, and is the best way to set up colours when printing.

Finally, it supports colour calibration, so that you can make different output devices produce the same colours. (This feature is not supported by RISC OS 2)

Before reading this chapter, you should be familiar with the VDU, sprite and font manager principles.

We also advise that you read the chapter entitled Printing a document from an application. This section gives advice on which ColourTrans calls you should use to set colours when printing. You'll probably find it easiest if you use the same calls for screen output; you should then find that your routines for printer and screen output can share large parts of coding.

Overview

The ColourTrans module is provided on disc in RISC OS 2 as the file System:Modules.Colours, but is in the ROM for later releases of RISC OS. Any application which uses it should ensure it is present using the *RMEnsure command, say from an Obey file. For example:

```
RMEnsure ColourTrans 0.51 RMLoad System:Modules.Colours
RMEnsure ColourTrans 0.51 Error You need ColourTrans 0.51
or later
```

Definition of terms

Here are some terms you should know when using this chapter.

GCOL is like the colour parameter passed to VDU 17. It uses a simple format for 256 colour modes.

Colour number is what is written into screen memory to achieve a given colour in a particular mode.

Palette entry is a word that contains a description of a physical colour in red, green and blue levels. Usually, this term refers to the required colour that is passed to a ColourTrans SWI.

Palette pointer is a pointer to a list of palette entries. The table would have one entry for each logical colour in the requested mode. In 256 colour mode, only 16 entries are needed, as there are only 16 palette registers.

Closest colour is the colour in the palette that most closely matches the palette entry passed. Furthest colour is the one furthest from the colour requested. These terms refer to a least-squares test of closeness.

Finding a colour

There are many SWIs that will find the best fit colour in the palette for a set of parameters. Here is a list of the different kinds of parameters that can return a best fit colour:

- Given palette entry, return nearest or furthest GCOL
- Given palette entry, return nearest or furthest colour number
- Given palette entry, mode and palette pointer, return nearest or furthest GCOL

- Given palette entry, mode and palette pointer, return nearest or furthest colour number

Setting a colour

Some SWIs will set the VDU driver GCOL to the calculated GCOL after finding it.

- Given palette entry, return nearest GCOL, and set that colour
- Given palette entry, return furthest GCOL, and set that colour

Conversion

There is a pair of SWIs to convert GCOLs to and from colour numbers. Note that this only has meaning for 256 colour modes. There are also SWIs to convert between different colour models, such as RGB, CIE, HSV, and CMYK.

Sprites and Fonts

ColourTrans provides full facilities for setting the colours used by sprites and fonts.

Using other palette SWIs

If an application changes the output palette (perhaps by changing the screen colours or by switching output to a sprite), then it has to call a SWI to inform ColourTrans. This is because ColourTrans maintains a cache used for mapping colours. If the palette has independently changed, then it has no way of telling.

If the screen mode has changed there is no need to use this call, since the ColourTrans module detects this itself - but, under RISC OS 2, if output is switched to a sprite (and ColourTrans will be used) then the SWI must also be called.

Wimp

If you are using the Wimp interface, then the ColourTrans calls are fine to use, because they never modify the palette.

Printing

Because ColourTrans allows an application to request an RGB colour rather than a logical colour, it is ideal for use with the printer drivers, where a printer may be able to represent some RGB colours more accurately then the screen.

Colour calibration

There is a major problem in working with colour documents. This is that, if the user selects some colours on the screen, they may well come out as different colours on a printer or other final output device. Colour calibration is a way to get round this problem.

Colour calibration involves calibrating the screen colours with a fixed standard set of colours, and also calibrating the output device colours to the same fixed set of colours. Then, when an application draws to the screen, it does so in standard colours which are converted by the OS to screen colours. If the application draws to the printer it again does so in standard colours, but this time they are converted to printer colours.

So, for the user, calibrating the colours will give constant colour reproduction throughout the system, for the cost of calibrating the devices in the first place.

Colour calibration is not available in RISC OS 2.

Technical Details

Colours

Two different colour systems are used in 256 colour modes. The GCOL form is much easier to use, while the colour number is optimised for the hardware. In all other colour modes, they are identical.

The palette entry used to request a given physical colour is in the same format as that used to set the anti-alias palette in the font manager.

GCOL

The 256 colour modes use a byte that looks like this:

Bit(s)	Meaning
0	Tint bit 0 (red+green+blue bit 0)
1	Tint bit 1 (red+green+blue bit 1)
2	Red bit 2
3	Red bit 3 (high)
4	Green bit 2
5	Green bit 3 (high)
6	Blue bit 2
7	Blue bit 3 (high)

This format is converted into the internal 'colour number' format when stored, because that is what the VIDC hardware recognises.

Colour number

The 256 colour mode in the colour number looks like this:

Bit(s)	Meaning
0	Tint bit 0 (red+green+blue bit 0)
1	Tint bit 1 (red+green+blue bit 1)
2	Red bit 2
3	Blue bit 2
4	Red bit 3 (high)
5	Green bit 2
6	Green bit 3 (high)
7	Blue bit 3 (high)

In fact the bottom 4 bits of the colour number are obtained via the palette, but the default palette in 256 colour modes is set up so that the above settings apply, and this is not normally altered.

Palette entry

The palette entry is a word of the form &BBGGRR00. That is, it consists of four bytes, with the palette value for the blue, green and red gun in the top three bytes. Bright white, for instance would be &FFFFFF00, while half intensity cyan would be &77770000. The current graphics hardware only uses the upper nibbles of these colours, but for upwards compatibility the lower nibble should contain a copy of the upper nibble.

Finding a colour

The SWIs that find the best fit have generally self explanatory names. As shown in the overview, they follow a standard pattern. They are as follows:

SWI ColourTrans_ReturnGCOL (on page 18)
Given palette entry, return nearest GCOL

SWI ColourTrans_ReturnOppGCOL (on page 28)
Given palette entry, return furthest GCOL

SWI ColourTrans_ReturnColourNumber (on page 22)
Given palette entry, return nearest colour number

SWI ColourTrans_ReturnOppColourNumber (on page 32)
Given palette entry, return furthest colour number

SWI ColourTrans_ReturnGCOLForMode (on page 24)
Given palette entry, mode and palette pointer, return nearest GCOL

SWI ColourTrans_ReturnOppGCOLForMode (on page 34)
Given palette entry, mode and palette pointer, return furthest GCOL

SWI ColourTrans_ReturnColourNumberForMode (on page 26)
Given palette entry, mode and palette pointer, return nearest colour number

SWI ColourTrans_ReturnOppColourNumberForMode (on page 36)
Given palette entry, mode and palette pointer, return furthest colour number

Palette pointers

Where a palette pointer is used, certain conventions apply:

- a palette pointer of -1 means the current palette is used
- a palette pointer of 0 means the default palette for the specified mode.

Modes

Similarly, where modes are used:

- mode -1 means the current mode.

Best fit colour

These calls use a simple algorithm to find the colour in the palette that most closely matches the high resolution colour specified in the palette entry. It calculates the distance between the colours, which is a weighted least squares function. If the desired colour is (Rd, Bd, Gd) and a trial colour is (Rt, Bt, Gt), then:

distance = redweight × $(R_t-R_d)^2$ + greenweight × $(G_t-G_d)^2$ + blueweight × $(B_t-B_d)^2$

where redweight = 2, greenweight = 4 and blueweight = 1. These weights are set for the most visually effective solution to this problem. (In RISC OS 2, the weights used were 2, 3 and 1 respectively.)

Setting a colour

SWI ColourTrans_SetGCOL (on page 20) will act like ColourTrans_

ReturnGCOL, except that it will set the graphics system GCOL to be as close to the colour you requested as it can. Note that ECF patterns will not yet be used in monochrome modes to reflect grey shades, as they are with Wimp_SetColour.

Similarly, *SWI ColourTrans_SetOppGCOL (on page 30)* will set the graphics system GCOL with the opposite of the palette entry passed.

Conversion

To convert between the GCOL and colour number format in 256 colour modes, the SWIs *SWI ColourTrans_GCOLToColourNumber (on page 38)* and *SWI ColourTrans_ColourNumberToGCOL (on page 39)* can be used.

Sprites and Fonts

SWI ColourTrans_SelectTable (on page 13) will set up a translation table in the buffer. *SWI ColourTrans_SelectGCOLTable (on page 16)* will set up a list of GCOLs in the buffer. See the chapter entitled Pixel translation table for a definition of these tables (although the latter call does not in fact relate to sprites).

SWI ColourTrans_ReturnFontColours (on page 40) will try and find the best set of logical colours for an anti-alias colour range. *SWI ColourTrans_SetFontColours (on page 42)* also does this, but sets the font manager plotting colours as well. It calls Font_SetFontColours, or Font_SetPalette in 256 colour modes - but it works out which logical colours to use beforehand. See the chapter entitled *Colours (on page 0)* for details of using colours and anti-aliasing colours; see also the descriptions of the relevant commands later in the same chapter, in *SWI Font_SetFontColours (on page 0)* and *SWI Font_SetPalette (on page 0)*.

Using other palette SWIs

If a program has changed the palette, then *SWI ColourTrans_InvalidateCache (on page 44)* must be called. This will reset its internal cache. This applies to Font_SetFontColours or Wimp_SetPalette or VDU 19 or anything like that, but not to mode change, since this is detected automatically.

Under RISC OS 2 you must also call this SWI if output has been switched to a sprite, and ColourTrans is to be called while the output is so redirected. You must then call it again after output is directed back to the screen. Later versions of RISC OS automatically do this for you.

Colour calibration

Colour calibration is performed by ColourTrans using a calibration table that maps from device colours to standard colours.

The palette in RISC OS maps logical colours to device colours (also known as physical colours). When you ask RISC OS to select a colour for you, it takes this palette and uses a calibration table to convert the device colours to standard colours, giving a (transient) palette that maps logical colours to standard colours. It then chooses the closest standard colour to the one that you have specified.

Calibration tables

A calibration table is a one-to-one map that fills the device colour space, but does not necessarily fill the standard colour space. In fact, it consists of three separate mappings: one for each component of the device space (red, green and blue on a monitor, for example). Each mapping consists of a series of device component/ standard colour pairs.

The pairs are stored as 32-bit words, in the form &BBGGRRDD, where DD is the amount of the device component (from 0 to 255), and BBGGRR is the standard colour corresponding to that amount. The two other device components are presumed to be zero.

The format of the table is:

Word	Meaning
0	Number of pairs of component 1 (n1)
1	Number of pairs of component 2 (n2)
2	Number of pairs of component 3 (n3)
3	n1 words giving pairs for component 1
3 + n1	n2 words giving pairs for component 2
3 + n1 + n2	n3 words giving pairs for component 3

The length of the table is therefore 3 + n1 + n2 + n3 words.

Within each of the three sets of mappings, the words must be sorted in ascending order of device component. To fill the device colour space, there must be entries for device components of 0 and 255, so there must be at least two pairs for each component.

As an example, a minimal calibration table might be:

Word	Meaning
&00000002	2 pairs of red component
&00000002	2 pairs of green component
&00000002	2 pairs of blue component
&02010300	Device colour &000000 corresponds to standard colour 020103
&0203FDFF	Device colour &0000FF corresponds to standard colour 0203FD
&02010300	Device colour &000000 corresponds to standard colour 020103
&03FC02FF	Device colour &00FF00 corresponds to standard colour 03FC02
&02010300	Device colour &000000 corresponds to standard colour 020103
&FF0302FF	Device colour &FF0000 corresponds to standard colour FF0302

(Both device and standard colours are given in the format &BBGGRR)

The default mapping for the screen is that device colours and standard colours are the same. This produces the same effect as earlier uncalibrated versions of ColourTrans.

To convert a specific device colour to a standard colour, ColourTrans splits the device colour into its three component parts. Then, for each component, it uses linear interpolation between the two device components 'surrounding' the required device component. The standard colours thus obtained for each component are then summed to give the final calibrated standard colour.

Colour calibration is not available in RISC OS 2.

Service Calls

Service_CalibrationChanged (Service Call &5B)

Screen calibration is changed

On entry

R1 = &5B (reason code)

On exit

R1 = Must be preserved. This service call should not be claimed
RAll preserved

Use

This service is issued by the ColourTrans module when the ColourTrans_SetCalibration SWI has been issued.

It is noticed by the Palette utility in the desktop, which broadcasts a Message_PaletteChange.

This service call is not used by RISC OS 2.

Related SWIs

SWI ColourTrans_SetCalibration (on page 46)

Service_InvalidateCache (Service Call &82)

Broadcast whenever the cache is flushed within ColourTrans

On entry

R1 = &82 (reason code)

On exit

RAll preserved

Use

This service is broadcast whenever the cache is flushed within ColourTrans. You should never claim it.

This service call is not used by RISC OS 2.

Related APIs

None

SWI Calls

ColourTrans_SelectTable (SWI &40740)

Sets up a translation table in a buffer

On entry

R0 = source mode, or -1 for current mode, or (if = 256) pointer to sprite, or (if > 256) pointer to sprite area
R1 = source palette pointer, or -1 for current palette, or (if R0 >= 256) pointer to sprite name/sprite in area pointed to by R0 (as specified by bit 0 of R5)
R2 = destination mode, or -1 for current mode
R3 = destination palette pointer, or -1 for current palette, or 0 for default for the mode
R4 = pointer to buffer, or 0 to return required size of buffer
R5 = flags (used if R0 >= 256):

Bit(s)	Meaning
0	R1 = pointer to sprite; else R1 = pointer to sprite name
1	use current palette if sprite doesn't have one; else use default
2	use R6 and R7 to specify transfer function
24-31	format of table:

Value	Meaning
0	return Pixel translation table
1	return physical palette table
other	reserved

other	Reserved, must be zero

R6 = pointer to workspace for transfer function (if R0 >= 256, and bit 2 of R5 is set)
R7 = pointer to transfer function (if R0 >= 256, and bit 2 of R5 is set)

On exit

R0 - R3 preserved
R4 = required size of buffer (if R4 = 0 on entry), or preserved
R5 - R7 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call sets up a translation table in a buffer - that is, a set of colour numbers as used by scaled sprite plotting. You may specify the source mode palette either directly, or (except in RISC OS 2) by specifying a sprite. See the chapter entitled SWI Pixel translation table for details of such tables.

You should use this call rather than any other to set up translation tables for sprites, as it copes correctly with sprites that have a 256 colour palette.

If bit 2 of the flags word in R5 is set, then R6 and R7 are assumed to specify a transfer routine, which is called to preprocess each palette entry before it is converted. The entry point of the routine (as specified in R7) is called with the palette entry in R0, and the workspace pointer (as specified in R6) in R12. The palette entry must be returned in R0, and all other registers preserved.

In RISC OS 2, R0 must be less than 256, and so R5 - R7 are unused. Consequently, to use a sprite as the source you first have to copy its palette information out from its header. Furthermore, you cannot find the required size of the buffer by setting R4 to 0 on entry.

If R0 is 256 on entry, it is assumed not to point to a sprite area, but R1 is still assumed to point to a sprite. This special value is useful if you need to use sprites that are not held in a sprite area. For example, Draw uses it for sprites that are held in a DrawFile without a preceding sprite area control block.

Related SWIs

SWI ColourTrans_GenerateTable (on page 78)

Related vectors

ColourV

ColourTrans_SelectGCOLTable (SWI &40741)

Sets up a list of GCOLs in a buffer

On entry

R0 = source mode, or -1 for current mode, or (if >= 256) pointer to sprite area
R1 = source palette pointer, or -1 for current palette, or (if R0 >= 256) pointer to sprite name/sprite in area pointed to by R0 (as specified by bit 0 of R5)
R2 = destination mode, or -1 for current mode
R3 = destination palette pointer, or -1 for current palette, or 0 for default for the mode
R4 = pointer to buffer
R5 = flags (used if R0 >= 256):

Bit(s)	Meaning
0	R1 = pointer to sprite; else R1 = pointer to sprite name
1	use current palette if sprite doesn't have one; else use default
2-31	Reserved, must be zero

On exit

R0 - R5 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a source mode and palette (either directly, or - except in RISC OS 2 - from a sprite), a destination mode and palette, and a buffer, sets up a list of GCOLs in the buffer. The values can subsequently be used by passing them to GCOL and Tint.

In RISC OS 2, R0 must be less than 256, and so R5 is unused. Consequently, to use a sprite as the source you first have to copy its palette information out from its header.

Related vectors

ColourV

ColourTrans_ReturnGCOL (SWI &40742)

Gets the closest GCOL for a palette entry

On entry

R0 = palette entry

On exit

R0 = GCOL

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, returns the closest GCOL in the current mode and palette.

It is equivalent to ColourTrans_ReturnGCOLForMode for the given palette entry, with parameters of -1 for both the mode and palette pointer.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_SetGCOL (on page 20)
SWI ColourTrans_ReturnColourNumber (on page 22)
SWI ColourTrans_ReturnGCOLForMode (on page 24)
SWI ColourTrans_ReturnOppGCOL (on page 28)

Related vectors

ColourV

ColourTrans_SetGCOL (SWI &40743)

Sets the closest GCOL for a palette entry

On entry

R0 = palette entry
R3 = flags
R4 = GCOL action

On exit

R0 = GCOL
R2 = log2 of bits-per-pixel for current mode
R3 = initial value AND &80
R4 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, works out the closest GCOL in the current mode and palette, and sets it.

Bit(s)	Meaning
7	Set: set background colour
	Clear: set foreground colour
8	Set: use ECFs to give a better approximation to the colour
	Clear: don't use ECFs

The remaining bits of R3 and the top three bytes of R4 are reserved, and should be set to zero to allow for future expansion. Bit 8 of R3 is ignored in RISC OS 2, which does not support ECF patterns with this call.

Note that if you are using ECF-generating calls, you cannot use the returned GCOL to reselect the pattern; you must instead repeat this call.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnGCOL (on page 18)
SWI ColourTrans_SetOppGCOL (on page 30)

Related vectors

ColourV

ColourTrans_ReturnColourNumber (SWI &40744)

Gets the closest colour for a palette entry

On entry

R0 = palette entry

On exit

R0 = colour number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, returns the closest colour number in the current mode and palette.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnGCOL (on page 18)
SWI ColourTrans_ReturnColourNumberForMode (on page 26)
SWI ColourTrans_ReturnOppColourNumber (on page 32)

Related vectors

ColourV

ColourTrans_ReturnGCOLForMode (SWI &40745)

Gets the closest GCOL for a palette entry

On entry

R0 = palette entry
R1 = destination mode, or -1 for current mode
R2 = palette pointer, or -1 for current palette, or 0 for default for the mode

On exit

R0 = GCOL
R1 preserved
R2 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, a destination mode and palette, returns the closest GCOL.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnGCOL (on page 18)
SWI ColourTrans_SetGCOL (on page 20)
SWI ColourTrans_ReturnColourNumberForMode (on page 26)
SWI ColourTrans_ReturnOppGCOLForMode (on page 34)

Related vectors

ColourV

ColourTrans_ReturnColourNumberForMode (SWI &40746)

Gets the closest colour for a palette entry

On entry

R0 = palette entry
R1 = destination mode, or -1 for current mode
R2 = palette pointer, or -1 for current palette, or 0 for default for the mode

On exit

R0 = colour number
R1 preserved
R2 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, a destination mode and palette, returns the closest colour number.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnColourNumber (on page 22)
SWI ColourTrans_ReturnGCOLForMode (on page 24)
SWI ColourTrans_ReturnOppColourNumberForMode (on page 36)

Related vectors

ColourV

ColourTrans_ReturnOppGCOL (SWI &40747)

Gets the furthest GCOL for a palette entry

On entry

R0 = palette entry

On exit

R0 = GCOL

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, returns the furthest GCOL in the current mode and palette.

It is equivalent to ColourTrans_ReturnOppGCOLForMode for the given palette entry, with parameters of -1 for both the mode and palette pointer.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnGCOL (on page 18)
SWI ColourTrans_SetOppGCOL (on page 30)
SWI ColourTrans_ReturnOppColourNumber (on page 32)
SWI ColourTrans_ReturnOppGCOLForMode (on page 34)

Related vectors

ColourV

ColourTrans_SetOppGCOL (SWI &40748)

Sets the furthest GCOL for a palette entry

On entry

R0 = palette entry
R3 = 0 for foreground, or 128 for background
R4 = GCOL action

On exit

R0 = GCOL
R2 = log2 of bits-per-pixel for current mode
R3 = initial value AND &80
R4 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, works out the furthest GCOL in the current mode and palette, and sets it.

The top three bytes of R3 and R4 should be zero, to allow for future expansion.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_SetGCOL (on page 20)
SWI ColourTrans_ReturnOppGCOL (on page 28)

Related vectors

ColourV

ColourTrans_ReturnOppColourNumber (SWI &40749)

Gets the furthest colour for a palette entry

On entry

R0 = palette entry

On exit

R0 = colour number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, returns the furthest colour number in the current mode and palette.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnColourNumber (on page 22)
SWI ColourTrans_ReturnOppGCOL (on page 28)
SWI ColourTrans_ReturnOppColourNumberForMode (on page 36)

Related vectors

ColourV

ColourTrans_ReturnOppGCOLForMode (SWI &4074A)

Gets the furthest GCOL for a palette entry

On entry

R0 = palette entry
R1 = destination mode or -1 for current mode
R2 = palette pointer, or -1 for current palette, or 0 for default for the mode

On exit

R0 = GCOL
R1 preserved
R2 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, a destination mode and palette, returns the furthest GCOL.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnGCOLForMode (on page 24)
SWI ColourTrans_ReturnOppGCOL (on page 28)
SWI ColourTrans_SetOppGCOL (on page 30)
SWI ColourTrans_ReturnOppColourNumberForMode (on page 36)

Related vectors

ColourV

ColourTrans_ReturnOppColourNumberForMode (SWI &4074B)

Gets the furthest colour for a palette entry

On entry

R0 = palette entry
R1 = destination mode or -1 for current mode
R2 = palette pointer, or -1 for current palette, or 0 for default for the mode

On exit

R0 = colour number
R1 preserved
R2 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a palette entry, a destination mode and palette, returns the furthest colour number.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnColourNumberForMode (on page 26)
SWI ColourTrans_ReturnOppColourNumber (on page 32)
SWI ColourTrans_ReturnOppGCOLForMode (on page 34)

Related vectors

ColourV

ColourTrans_GCOLToColourNumber (SWI &4074C)

Translates a GCOL to a colour number

On entry

R0 = GCOL

On exit

R0 = colour number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call changes the value passed from a GCOL to a colour number.

You should only call this SWI for 256 colour modes; the results will be meaningless for any others.

Related SWIs

SWI ColourTrans_ColourNumberToGCOL (on page 39)

Related vectors

ColourV

ColourTrans_ColourNumberToGCOL (SWI &4074D)

Translates a colour number to a GCOL

On entry

R0 = colour number

On exit

R0 = GCOL

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call changes the value passed from a colour number to a GCOL.

You should only call this SWI for 256 colour modes; the results will be meaningless for any others.

Related SWIs

SWI ColourTrans_GCOLToColourNumber (on page 38)

Related vectors

ColourV

ColourTrans_ReturnFontColours (SWI &4074E)

Finds the best range of anti-alias colours to match a pair of palette entries

On entry

R0 = font handle, or 0 for the current font
R1 = background palette entry
R2 = foreground palette entry
R3 = maximum foreground colour offset (0 - 14)

On exit

R0 preserved
R1 = background logical colour (preserved if in 256 colour mode)
R2 = foreground logical colour
R3 = maximum sensible colour offset (up to R3 on entry)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given background and foreground colours and the number of anti-aliasing colours desired, finds the maximum range of colours that can sensibly be used. So for the given pair of palette entries, it finds the best fit in the current palette, and then inspects the other available colours to deduce the maximum possible amount of anti-aliasing up to the limit in R3.

If anti-aliasing is desirable, you should set R3 = 14 on entry; otherwise set R3 = 0 for monochrome.

The values in R1 - R3 on exit are suitable for passing to Font_SetFontColours. You can also include them in a font string in a control (18) sequence, although we don't recommend this as the printer drivers do not properly support this feature.

Note that in 256 colour modes, you can only set 16 colours before previously returned information becomes invalid. Therefore, if you are using this SWI to obtain information to subsequently pass to the font manager, do not use more than 16 colours.

Also note that in 256 colour modes, the font manager's internal palette will be set, with all 16 entries being cycled through by ColourTrans.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

See *SWI Font_SetFontColours (on page 0)* of the *The Font Manager (on page 0)* for further details of the parameters used in this call.

Related SWIs

SWI ColourTrans_SetFontColours (on page 42)
SWI Font_SetFontColours (on page 0)

Related vectors

ColourV

ColourTrans_SetFontColours (SWI &4074F)

Sets the best range of anti-alias colours to match a pair of palette entries

On entry

R0 = font handle, or 0 for the current font
R1 = background palette entry
R2 = foreground palette entry
R3 = maximum foreground colour offset (0 - 14)

On exit

R0 preserved
R1 = background logical colour (preserved if in 256 colour mode)
R2 = foreground logical colour
R3 = maximum sensible colour offset (up to R3 on entry)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call, given a pair of palette entries, finds the best available range of anti-alias colours in the current palette, and sets the font manager to use these colours. It is the recommended way to set font colours, as the printer drivers properly support this call. A font string control (19) sequence uses this call, and so may also be used when printing.

The colours are not calibrated in RISC OS 2, but are calibrated in later versions.

Related SWIs

SWI ColourTrans_ReturnFontColours (on page 40)

Related vectors

ColourV

ColourTrans_InvalidateCache (SWI &40750)

Informs ColourTrans that the palette has been changed by some other means

On entry

None

On exit

None

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call must be issued whenever the palette has changed since ColourTrans was last called. This forces ColourTrans to update its cache. Note that colour changes due to a mode change are detected; you only need to use this if another of the palette change operations was used.

Under RISC OS 2 you must also call this SWI if output has been switched to a sprite, and ColourTrans is to be called while the output is so redirected. You must then call it again after output is directed back to the screen. For example, the palette utility on the icon bar calls this SWI when you finish dragging one of the RGB slider bars. Later versions of RISC OS automatically do this for you.

Related vectors

ColourV

ColourTrans_SetCalibration (SWI &40751)

Sets the calibration table for the screen

On entry

R0 = pointer to calibration table

On exit

None

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call copies the calibration table pointed to by R0 into the RMA as the new calibration table for the screen. If the call fails due to lack of room in the RMA then the calibration will be set to the default calibration for the screen, and the 'No room in RMA' error will be passed back. Another possible error is 'Bad calibration table', given if the device component pairs do not cover the full range &00 to &FF.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ReadCalibration (on page 48)

Related vectors

ColourV

ColourTrans_ReadCalibration (SWI &40752)

Reads the calibration table for the screen

On entry

R0 = 0 to read required size of table, or pointer to buffer

On exit

R0 preserved
R1 = size of table (if R0 = 0 on entry)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call reads the calibration table for the screen into the buffer pointed to by R0, which should be large enough to contain the complete table. Ideally you should first issue this call with R0=0 to read the size of the table, then allocate space, and then issue this call again to read the table.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_SetCalibration (on page 46)

Related vectors

ColourV

ColourTrans_ConvertDeviceColour (SWI &40753)

Converts a device colour to a standard colour

On entry

R1 = 24-bit device colour (&BBGGRR00 for the screen)
R3 = 0 to use the current screen calibration, or pointer to calibration table to use

On exit

R2 = 24-bit standard colour (&BBGGRR00)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call allows applications to read, say, screen colours, and find the standard colours to which they correspond.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertDevicePalette (on page 51)

Related vectors

ColourV

ColourTrans_ConvertDevicePalette (SWI &40754)

Converts a device palette to standard colours

On entry

R0 = number of colours to convert
R1 = pointer to table of 24-bit device colours
R2 = pointer to table to store standard colours
R3 = 0 to use the current screen calibration, or pointer to calibration table to use

On exit

R0 - R3 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call allows printer drivers to use the same calibration calculation code for their conversions between device and standard colours as the screen does. The printer device palette can be set up and then converted using this call to the standard colours using the printer's calibration table. This call is mainly provided to ease the load on the writers of printer drivers.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertDeviceColour (on page 50)

Related vectors

ColourV

ColourTrans_ConvertRGBToCIE (SWI &40755)

Converts RISC OS RGB colours to industry standard CIE colours

On entry

R0 = red component
R1 = green component
R2 = blue component

On exit

R0 = CIE X tristimulus value
R1 = CIE Y tristimulus value
R2 = CIE Z tristimulus value

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts RISC OS RGB colours to industry standard CIE colours, allowing easy interchange with other systems. The CIE standard that is output is the XYZ tristimulus values.

All parameters are passed as fixed point 32 bit numbers, with 16 bits below the point and 16 bits above the point. We suggest that you use numbers in the range 0 - 1, for compatibility with other conversion SWIs such as ColourTrans_ConvertRGBToCMYK.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertCIEToRGB (on page 55)

Related vectors

ColourV

ColourTrans_ConvertCIEToRGB (SWI &40756)

Converts industry standard CIE colours to RISC OS RGB colours

On entry

R0 = CIE X tristimulus value
R1 = CIE Y tristimulus value
R2 = CIE Z tristimulus value

On exit

R0 = red component
R1 = green component
R2 = blue component

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts industry standard CIE colours to RISC OS RGB colours, allowing easy interchange with other systems. The CIE standard that is accepted is the XYZ tristimulus values.

All parameters are passed as fixed point 32 bit numbers, with 16 bits below the point and 16 bits above the point. We suggest that you use numbers in the range 0 - 1, for compatibility with other conversion SWIs such as ColourTrans_ConvertCMYKToRGB.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertRGBToCIE (on page 53)

Related vectors

ColourV

ColourTrans_WriteCalibrationToFile (SWI &40757)

Saves the current calibration to a file

On entry

R0 = flags
R1 = file handle of file to save calibration to

On exit

R0 corrupted

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call saves the current calibration to a file. It does so by creating a list of * Commands which will recreate the current calibration.

If bit 0 of R0 is clear then the calibration will only be saved if it is not the default calibration. If bit 0 of R0 is set then the calibration will be saved even if it is the default calibration.

This call is not available in RISC OS 2.

Related vectors

ColourV

ColourTrans_ConvertRGBToHSV (SWI &40758)

Converts RISC OS RGB colours into corresponding hue, saturation and value

On entry

R0 = red component
R1 = green component
R2 = blue component

On exit

R0 = hue
R1 = saturation
R2 = value

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts RISC OS RGB colours into corresponding hue, saturation and value.

All parameters are passed as fixed point 32 bit numbers, with 16 bits below the point and 16 bits above the point. Hue ranges from 0 - 360 with no fractional element, whilst the remaining parameters are in the range 0 - 1 and may have fractional elements.

When dealing with achromatic colours, hue is undefined.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertHSVToRGB (on page 60)

Related vectors

ColourV

ColourTrans_ConvertHSVToRGB (SWI &40759)

Converts hue, saturation and value into corresponding RISC OS RGB colours

On entry

R0 = hue
R1 = saturation
R2 = value

On exit

R0 = red component
R1 = green component
R2 = blue component

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts hue, saturation and value into corresponding RISC OS RGB colours.

All parameters are passed as fixed point 32 bit numbers, with 16 bits below the point and 16 bits above the point. Hue ranges from 0 - 360 with no fractional element, whilst the remaining parameters are in the range 0 - 1 and may have fractional elements.

An error is generated if both the hue and saturation are 0; for this reason we recommend that when using this call 0 < hue <= 360.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertRGBToHSV (on page 58)

Related vectors

ColourV

ColourTrans_ConvertRGBToCMYK (SWI &4075A)

Converts RISC OS RGB colours into the CMYK model

On entry

R0 = red component
R1 = green component
R2 = blue component

On exit

R0 = cyan component
R1 = magenta component
R2 = yellow component
R3 = key (black) component

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts RISC OS RGB colours into the CMY (cyan/magenta/yellow) model with a K (key - ie black) additive, allowing easy preparation of colour separations.

All parameters are passed as fixed point 32 bit numbers in the range 0 - 1, with 16 bits below the point and 16 bits above the point. The 'K' acts as a black additive and is a value equally subtracted or added to the given CMY values.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertCMYKToRGB (on page 64)

Related vectors

ColourV

ColourTrans_ConvertCMYKToRGB (SWI &4075B)

Converts from the CMYK model to RISC OS RGB colours

On entry

R0 = cyan component
R1 = magenta component
R2 = yellow component
R3 = key (black) component

On exit

R0 = red component
R1 = green component
R2 = blue component

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call converts from the CMY (cyan/magenta/yellow) model with a K (key - ie black) additive to RISC OS RGB colours, allowing easy conversion from colour separations.

All parameters are passed as fixed point 32 bit numbers in the range 0 - 1, with 16 bits below the point and 16 bits above the point. The 'K' acts as a black additive and is a value equally subtracted or added to the given CMY values.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ConvertRGBToCMYK (on page 62)

Related vectors

ColourV

ColourTrans_ReadPalette (SWI &4075C)

Reads either the screen's palette, or a sprite's palette

On entry

R0 = source mode, or -1 for current mode, or (if >= 256) pointer to sprite area
R1 = source palette pointer, or -1 for current palette, or (if R0 >= 256) pointer to sprite name/sprite in area pointed to by R0 (as specified by bit 0 of R4)
R2 = pointer to buffer, or 0 to return required size in R3
R3 = size of buffer (if R2 <> 0)
R4 = flags (used if R0 >= 256):

Bit(s)	Meaning
0	R1 = pointer to sprite; else R1 = pointer to sprite name
1	Return flashing colours; else don't
2-31	Reserved, must be zero

On exit

R2 = pointer to next free word in buffer
R3 = remaining size of buffer

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call reads either the screen's palette, or a sprite's palette. It is the recommended way of doing so. It provides a way for applications to enquire about the palette and always read the absolute values, no matter what the hardware is capable of.

All palette entries are returned as true 24bit RGB, passing through the calibration if required. In 256 colour modes the palette is returned fully expanded (ie 256 palette entries, rather than the base 16 entries used by VIDC).

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_WritePalette (on page 68)

Related vectors

ColourVPaletteV

ColourTrans_WritePalette (SWI &4075D)

Writes to either the screen's palette, or to a sprite's palette

On entry

R0 = -1 to write current mode's palette, or pointer to sprite area
R1 = -1 to write current palette, else ignored (if R0 = -1); or (if R0 >= 0) pointer to sprite name/sprite in area pointed to by R0 (as specified by R4)
R2 = pointer to palette to write
R3 reserved (must be zero)
R4 = flags (used if R0 >= 0):

Bit(s)	Meaning
0	R1 = pointer to sprite; else R1 = pointer to sprite name
1	flashing colours; else not present
2-31	Reserved, must be zero

On exit

None

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call writes to either the screen's palette, or to a sprite's palette.

256 colour palettes are first compacted to the base 16 entries used by VIDC - but only if the compacted palette expands via the tint mechanism to the

original palette. Otherwise the full 256 colours are written.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ReadPalette (on page 66)

Related vectors

ColourVPaletteV

ColourTrans_SetColour (SWI &4075E)

Changes the foreground or background colour to a GCOL number

On entry

R0 = GCOL number
R3 = flags:

Bit(s)	Meaning
7	set background, else foreground
9	set text colour

R4 = GCOL action

On exit

RAll preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call changes the foreground or background colour to a GCOL number (as returned from ColourTrans_ReturnGCOL). You should only use it for GCOL numbers returned for the current mode.

If bit 9 of R3 is set on entry, then this call sets the text colours rather than the graphics colours.

This call is not available in RISC OS 2.

Related SWIs

SWI ColourTrans_ReturnGCOL (on page 18)

Related vectors

ColourV

ColourTrans_MiscOp (SWI &4075F)

This SWI call is for internal use only. You must not use it in your own code.

ColourTrans_WriteLoadingsToFile (SWI &40760)

Writes a * Command to a file that will set the ColourTrans error loadings

On entry

R1 = file handle

On exit

RAll preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call writes a * Command to the specified file that will set the error loadings within the ColourTrans module. This call is mainly provided to support desktop saving of the loadings.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related vectors

ColourV

ColourTrans_SetTextColour (SWI &40761)

Changes the text foreground or background colour to a GCOL number

On entry

R0 = palette entry
R3 = flags word:

Bit(s)	Meaning
7	set background colour; else set foreground colour
0-6	Reserved, must be zero
8-31	Reserved, must be zero

On exit

R0 = GCOL
R3 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call changes the text foreground or background colour to the GCOL number (as returned from ColourTrans_ReturnGCOL) that is closest to the specified palette entry. You should only use it for GCOL numbers returned for the current mode.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related SWIs

SWI ColourTrans_SetOppTextColour (on page 76)

Related vectors

ColourV

ColourTrans_SetOppTextColour (SWI &40762)

Changes the text foreground or background colour to a GCOL number

On entry

R0 = palette entry
R3 = flags word:

Bit(s)	Meaning
7	set background colour; else set foreground colour
0-6	Reserved, must be zero
8-31	Reserved, must be zero

On exit

R0 = GCOL
R3 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call changes the text foreground or background colour to the GCOL number (as returned from ColourTrans_ReturnGCOL) that is furthest from the specified palette entry. You should only use it for GCOL numbers returned for the current mode.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related SWIs

SWI ColourTrans_SetTextColour (on page 74)

Related vectors

ColourV

ColourTrans_GenerateTable (SWI &40763)

Sets up a translation table in a buffer

On entry

R0 = source mode, or -1 for current mode, or (if = 256) pointer to sprite, or (if > 256) pointer to sprite area
R1 = source palette pointer, or -1 for current palette, or (if R0 >= 256) pointer to sprite name/sprite in area pointed to by R0 (as specified by bit 0 of R5)
R2 = destination mode, or -1 for current mode
R3 = destination palette pointer, or -1 for current palette, or 0 for default for the mode
R4 = pointer to buffer, or 0 to return required size of buffer
R5 = flags:

Bit(s)	Meaning
0	R1 = pointer to sprite; else R1 = pointer to sprite name
1	use current palette if sprite doesn't have one; else use default
2	use R6 and R7 to specify transfer function
24-31	format of table:

Value	Meaning
0	return Pixel translation table
1	return physical palette table
other	reserved

other	Reserved, must be zero

R6 = pointer to workspace for transfer function (if bit 2 of R5 is set)
R7 = pointer to transfer function (if bit 2 of R5 is set)

On exit

R0 - R3 preserved
R4 = required size of buffer (if R4 = 0 on entry), or preserved
R5 - R7 preserved

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call is exactly the same as ColourTrans_SelectTable (see *SWI ColourTrans_SelectTable (on page 13)*), except that it assumes that R5 always contains a valid flags word.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related SWIs

SWI ColourTrans_SelectTable (on page 13)

Related vectors

ColourV

* Commands

*ColourTransLoadings

Sets the red, green and blue weightings used when trying to match colours

Syntax

```
*ColourTransLoadings <redweight> <greenweight>
<blueweight>
```

Parameters

<redweight>	- red weighting used when trying to match colours
<greenweight>	- green weighting used when trying to match colours
<blueweight>	- blue weighting used when trying to match colours

Use

*ColourTransLoadings sets the red, green and blue weightings used when trying to match colours (as described in *Finding a colour (on page 2)*).

The main purpose of this command is to enable the Task Manager to save the calibration when a desktop save is done. You should not use it yourself.

This command is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Examples

```
*ColourTransLoadings &2 &4 &1
```

Related SWIs

SWI ColourTrans_WriteLoadingsToFile (on page 73)

Related vectors

ColourV

*ColourTransMap

Sets up a calibration table from its parameters

Syntax

```
*ColourTransMap <RRGGBBDD> <RRGGBBDD> <RRGGBBDD>
<RRGGBBDD> <etc.>
```

Parameters

<RRGGBBDD> - 8 hex digits, such that &RRGGBBDD is the number to be placed in the calibration table

Use

*ColourTransMap sets up a calibration table from its parameters. The number of parameters passed for each component must have been specified in a previous *ColourTransMapSize command.

The main purpose of this command is to enable the Task Manager to save the calibration when a desktop save is done.

This command is not available in RISC OS 2.

Examples

```
*ColourTransMap 01000000 FF0000FF 00020000 00FE00FF etc.
```

Related * commands

*ColourTransMapSize (on page 82)

Related SWIs

SWI ColourTrans_WriteCalibrationToFile (on page 57)

Related vectors

ColourV

*ColourTransMapSize

Sets how parameters will be passed in the next *ColourTransMap command

Syntax

```
*ColourTransMapSize <n1> <n2> <n3>
```

Parameters

<*n1*> - number of parameters to be passed in *ColourTransMap for component 1

<*n2*> - number of parameters to be passed in *ColourTransMap for component 2

<*n3*> - number of parameters to be passed in *ColourTransMap for component 3

Use

*ColourTransMapSize sets the number of parameters that will be passed in the next *ColourTransMap command for each component. It hence also sets the size of the resultant calibration table, which will be (3 + n1 + n2 + n3) words long. The values n1, n2 and n3 are given in the reverse order to a standard calibration table.

The main purpose of this command is to enable the Task Manager to save the calibration when a desktop save is done.

This command is not available in RISC OS 2.

Examples

```
*ColourTransMapSize 8 10 8
```

Related * commands

*ColourTransMap (on page 81)

Related SWIs

SWI ColourTrans_WriteCalibrationToFile (on page 57)

Related vectors

ColourV

Document information

Maintainer(s): RISCOS Ltd <developer@riscos.com>

History:

Revision	Date	Author	Changes
1		ROL	Initial version