Documentation for the HawkV9Utils support module.

This document describes the 9 SWI calls provided by the module 'HawkV9Utils': This is the support module which forms part of the !HawkV9 application from Computer Concepts for the dithered version of the HawkV9 Mark II colour digitiser.

Version 1:00
18 October 1993

SWI HawkV9Utils_DigitiseFrame

Entry:	r0 = reserved (write 0)
	r1 = y offset
	r2 = flags
	r3 = log2 of no: of frames to sample

Exit:	r0 = pointer to result word
Use:	Digitise a frame into the framestore in the background.

Flags are

bits 0-1	0 non interlaced, grab any field 1 non interlaced, grab odd field 2 non interlaced, grab even field 3 interlaced
bit 2	0 grab 512 pixels per line 1 reserved
bit 3	0 sample at full pclk rate 1 reserved
bit 4	0 don't swap fields 1 swap fields over

The result word will remain at 0 while the frame is digitising, and will change to -1 to indicate the frame has been correctly digitised, or change to a +ve error number if an error occurs while digitising:

This makes the result word ideal for a wimp front end to use, using poll word non-zero on risc os 3, and polling the word on nulls on risc-os 2

Result word meaning:

DR_Done	=-1
DR_Digitising	=0
DR_FIQclaimfailed	=1
DR_IRQoverrun	=2
DR_NoVideo	=3
DR_BadVideo	=4

The format of the frame in the frame buffer varies according to what flags were set: The lines are always in the order:

Left half of sample 0 of line 0
If necessary right half of sample 0 of line 0
This repeats up to sample n if necessary, then next line follows:

SWI HawkV9Utils_AbortDigitise

Entry: -

Exit: -

Use: Stops any background digitising process that may be going on instantly:

SWI HawkV9Utils_InitRegion

Entry: r2 = pointer to block (word aligned)
r3 = size of block (ignored at moment)

Exit: r2 = pointer to block initalised as null region

Use: Initialises a block with a null region

SWI HawkV9Utils_AddRectangleToRegion

Entry: r2 = pointer to region
r3 = rectangle x min (inclusive)
r4 = rectangle y min (inclusive)
r5 = rectangle x max (exclusive)
r6 = rectangle y max (exclusive)

Exit: -

Use: Adds a rectangle to the region.

Rectangle coordinates should be with +ve x coordinates going right, -ve y coordinates going down, and the origin should be above, and left of the top left of the rectangle. IE all x coordinates +ve, all y coordinates -ve, as is the case for wimp window work areas.
Coordinates are in os units.

NB Currently the rectangle MUST NOT overlap with any existing part of the region.

SWI HawkV9Utils_DisplayFrame

Entry:

r0 = x coordinate (os units) of top left of where framestore is to appear (NB the clipping region may mean that nothing is plotted here):
r1 = y coordinate (os units)
r2 = pointer to clipping region
r3 = pointer to display context
r4 = scale factor (1=normal,2=half size)
r5 = pointer to palette lut

Exit: -

Use:

Display a frame direct from the framestore into a display: If the display context is set up to point at the screen this call can plot the framestore directly onto the screen: Alternatively this call can be used to read the framestore into a sprite or any other form of bitmap:

Currently only 4,8 and 16bpp displays are handled. For 4bpp displays the palette lut is used. The first 8 bytes of this are the colour numbers used when plotting black, red, green, yellow, blue, meganta, cyan and white pixels respectively: The next 16K of table is made up of words containing in the bottom 16 bits all possible combinations of 4 pixels of colour:

For 8bpp displays the palette lut is not currently used, the palette is assumed to be the standard acorn 8bpp palette:
For 16bpp displays the pixel format is 5 bits raw BGR:

The format of a display context is

Offset	Contents
+0	pointer to bitmap
+4	log 2 of bpp
+8	x eig factor
+12	y eig factor
+16	line length (bytes)
+20	x window limit (pixels across -1)
+24	y window limit (pixels down -1)

to plot things on screen the display context may easily be read using OS_ReadVduVariables to read the appropritate mode and vdu vars.

SWI HawkV9Utils_StartGrab

Entry:
r0 = grab type to perform
r1 = x origin in pixels (origin is top left)
r2 = y origin in pixels (+ve y is DOWN)
r3 = width in pixels
r4 = depth in pixels

Exit:
r0 = corrupt
r1 = pointer to status word, non 0 when more data avaliable

Use:
Start a quick or high quality frame grab

All grabs are avaliable in either 2x1 or 1x1 aspect ratio:

Grab types are
bit 31 = 0 for monochrome, 1 for colour
bit 30 = 0 for 2x1 pixels (max res 512x256), 1 for 1x1 pixels (512x512 res)
bit 29 = exchange fields
bits 0,1,2 = log2 of amount of time sampling, ie
0 for direct image (16bpp colour/6bpp mono)
1 for 2x averaging (19bpp colour/7bpp mono)
2 for 4x averaging (22bpp colour/8bpp mono)
3 for 8x averaging (25bpp colour/9bpp mono)

Once a grab has been started normal digitising is prevented, until all the scanlines have been read using the GrabScanline SWI, or the grab is aborted:

SWI HawkV9Utils_GrabScanline

Entry:
r0 = pointer to buffer for scanline

Exit:
r1 = pointer to status word, non 0 when more data avaliable

Use:
If the routine is called when data is not yet ready it returns carry set, otherwise if it is returning valid data it returns carry clear:

Format of grab scan line output is 3 words per pixel, blue, green then red, in 16:16 fixed point form. The highest value that can be returned is &ffff, the lowest &0000:

SWI HawkV9Utils_AbortGrab

Entry: -

Exit: -

SWI HawkV9Utils_PALDecoder

Entry: r0 = brightness (0-63 to set, -1 to read)
r1 = contrast (0-63 to set, -1 to read)
r2 = saturation (0-63 to set, -1 to read)

Exit: r0 = previous brightness
r1 = previous contrast
r2 = previous saturation

Use: Sets the pal decoder controls