

CLAIMING PROCESSOR VECTORS

The new SWI OS_ClaimProcessorVector has been added to allow low-level code to claim the ARM's hardware vectors which are called when interrupts and aborts occur. On entry, R0 contains the vector number:

- 0 Branch through zero
- 1 Undefined Instruction
- 2 SWI executed
- 3 Prefetch abort
- 4 Data abort
- 6 IRQ

Bit 8 of R0 should be set to claim the vector, and clear to release it. For claiming, R1 should be set to the address to call, and for release R2 to the address used when the vector was claimed. The current address is returned in R2 for a claim operation.

This is one of those calls that you should only use if you know exactly what you're doing as RISC OS provides no checking or support on its use!

KILLING SPRITES

The new command:

`*WimpKillSprite <sprite name>`

will remove the named sprite from the Wimp's RAM sprite pool. This can be particularly useful when an older application has loaded its own copy of one of the window tools, wiping over the newer 3D version.

TILED WINDOWS

Provided that window tiling has been enabled using the !Boot application, the Wimp will replace any window background with a colour of 1 by a pattern formed by tiling a sprite called `tile_1`. A default sprite is included in the Wimp's pool, but an application can include a different one in its own sprite file, provided the window definition points to the application's sprite area. This means that each application could use a different texture if desired.

DON'T USE VDU

With the ever increasing complexity of the video system, many of the older VDU calls to change colour and the like are now very restrictive in

Risc PC Hints and Tips

David Spencer offers another potpourri of RISC OS 3.5 features.

the way they can operate. Acorn's recommendation is that VDU calls are only used for actual printing of characters and cursor positioning, with calls such as SWI OS_Plot and SWI OS_SetColour being used for other graphics operations.

RESIZING ICONS

A new Wimp call is SWI Wimp_ResizeIcon. This allows an existing icon to be resized, and optionally moved. On entry, R0 and R1 contain the window and icon handles respectively, and R2 to R5 contain the new bounding box. The Wimp will coordinate all the necessary redrawing operations automatically.

CHANGING SCREEN MODE

A new call, SWI OS_ScreenMode, handles the changing of screen modes. To change mode, R0 should be set to 0, and R1 either to an old-style mode number, or to a pointer to a mode specifier block (see *New Modes for Future Machines* in RISC User 7:5).

To read the current mode, issue the call with R0 set to 1. On exit R1 will contain either a mode number or a pointer to a mode specifier block.

HANDLING COLOURS

With the advent of 32-thousand and 16-million colour modes, it is important that programs retain R, G and B values as eight-bit quantities. In the past, many programs have treated them as four-bit values and set the lower nibble to either zero, or the same as the upper nibble.

