New Technical Notes

Macintosh

Developer Support

OS 510 -Math Function Q&As

OS Utilities

Revised by: Developer Support Center September 1993
Written by: Developer Support Center March 1993

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

Different Macintosh SANE engines generate different NANs
Date Written: 9/10/92
Last reviewed: 11/6/92

Our program generates NAN(030) on a 68040 but generates NAN(255) on a 68030 with a 68881 FPU, causing it to work differently on a Macintosh Quadra than on a Macintosh IIfx. Is this a bug or are there supposed to be differences in numerical results between the emulated code and the hardware code?

—

The different behavior you're encountering is caused by the different SANE engine. In the past, a hardware-generated NAN was always coded 255; now SANE tries to let you know which operation caused the NAN. For example, 34 means an invalid argument to an inverse trig function *(Apple Numerics Manual,* 2nd ed., page 41).

What can be a bit puzzling is why you'd get such a return if you're compiling with the 68881 flag. The reason is that sometimes you invoke the SANE engine indirectly. If you eliminate WriteLn calls before calling ArcCos, for example, the NAN code returned is 255. The deeper reason is that the SANE engine installs exception handlers when called and leaves them on for the duration so that subsequent operations cause the handlers to be called

if an exception occurs. The differences you see are caused by the operand error handler.

If this a problem for you (though normally you'd handle all NANs in a similar manner), a possible solution is to bracket calls that may cause problems with GetEnvironment and SetEnvironment calls. This will prevent the exception handling and you should get the result

you expect, NAN(255). Note that the compiler may be doing you a favor so that you don’t have to make direct calls for this to happen.