Q: In the process of porting my code to the Intel platform I am encountering a problem that I was not previously encountering on Motorola hardware. Evidently when I send a message to a nil object where the sender returns a floating point value, my program experiences problems. Is this a bug?

A: The definition of the Objective-C language states that sending a message to a nil receiver returns a zero value. This may be something that your code relies upon. On Motorola hardware, no matter what the return type, this is dependable behavior. However, the architecture of the Intel platform has forced a change you need to be aware of. The 486 maintains a separate floating point stack that is used to return values not derived from integer/pointer. When confronted with a nil receiver, the runtime system cannot establish the appropriate Class to determine whether it should return a zero on the floating point stack or the integer stack. It returns an integer zero, and so methods returning id, integer, char or short are not affected. In cases where the return type is float, double, or struct, the behavior is unpredictable. The best approach is to write your code such that it is portable between all NEXTSTEP architectures, current and future. To do this, you should treat messaging a nil receiver as an error when the selector's return type is non-integer.

In Release 3.1, a new AppKit default has been added that can help to debug this problem. **NXTrapIllegalFloatingPointOps** will catch invalid comparisons to NaN as well as other illegal IEEE floating point operations. When one occurs, a floating point exception will be

raised and the application will crash helping you to debug the problem. The syntax for this default is:

```
appname.app/appname  -NXTrapIllegalFloatingPointOps   YES
```

See the Release 3.1 AppKit Release Notes for more information on **NXTrapIllegalFloatingPointOps**.

QA890

Valid for 3.1