

## §DSNAN Deprecate Signaling NaNs

*Copyright © 2005 by the Institute of Electrical and Electronics Engineers, Inc.  
Three Park Avenue  
New York, New York 10016-5997, USA  
All rights reserved.*

*This document is an unapproved draft of a proposed IEEE-SA Standard - USE AT YOUR OWN RISK. As such, this document is subject to change. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only. Prior to submitting this document to another standard development organization for standardization activities, permission must first be obtained from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department. Other entities seeking permission to reproduce portions of this document must obtain the appropriate license from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department. The IEEE is the sole entity that may authorize the use of IEEE owned trademarks, certification marks, or other designations that may indicate compliance with the materials contained herein.*

*IEEE Standards Activities Department  
Standards Licensing and Contracts  
445 Hoes Lane, P.O. Box 1331  
Piscataway, NJ 08855-1331, USA*

### Change Log

§DSNAN Deprecate Signaling NaNs : Deprecate Signaling NaNs because they have no portable use.

### 3.1. Sets of Representable Entities

The values of these parameters for each basic format are given in Table 1a; constraints on these parameters for extended formats are given in Table 1f. Basic formats are named according to the number of bits in their encoding. Within each basic or extended format, the following entities shall be provided:

- Positive and negative zero
- Nonzero numbers of the form  $(-1)^s b^q b^{-p} m$ , where
- $s$  is 0 or 1 [ $s$  is any integer  $0 \leq s \leq 1$ ]
- $q$  is any integer  $emin \leq q+p-1 \leq emax$
- $m$  is any integer  $0 < m < b^p$
- Two infinities,  $+\infty$  and  $-\infty$
- Quiet NaNs
- Signaling NaNs if the implementation supports them

### 3.3 Basic Decimal Format Encodings

If  $G$  is 11111, then  $r$  and  $v$  are NaN regardless of  $S$ . The values of  $F$  and  $T$  distinguish various NaNs. If the implementation supports signaling NaNs and  $F_2$ , the most significant bit of  $F$ , is 1, then the NaN is signaling; otherwise the NaN is quiet.

### 3.4. Extended Formats

An extended format must encompass all the entities defined in one of that implementation's basic formats – signed zeros, signed infinities, quiet and optional signaling NaNs, and numbers - as well as numbers of wider precision and exponent range according to Table 1f.

## 6.2. Operations with NaNs

There are two different kinds of NaN, signaling and quiet. An implementation shall define one or more Quiet NaNs to be supported in all operations. An implementation might [for 754 compatibility] also define one or more Signaling NaNs as well. Signaling NaNs afford representations...

### In Annex 1

Class(x) tells which of the following ten classes x falls into: signaling NaN, quiet NaN,  $-\infty$ .

### In Annex N

6.2: A conforming implementation shall provide no Signaling NaNs generating Invalid exceptions as specified in Section 7. All NaNs are Quiet.

[754's Signaling NaNs feature is deprecated in 754R because the 754 specification was incomplete - e.g. can't return a snan as a function result in IA32 ABI - and there is no portable way to exploit Signaling NaNs.

One way to provide some of the useful features of Signaling NaNs is to provide a language-defined mode to enable a language-defined subset of NaN patterns to be Signaling.

Raising an Invalid Exception on a Signaling NaN operand, as specified by 754, is inconvenient for system and application programmers. An untrapped Signaling NaN operand exception has very limited usefulness and signaling Invalid confuses and complicates handling of other Invalid exceptions.

The deprecation of Signaling NaNs is part of Appendix N because existing 754-conforming implementations would suffer a performance penalty to conform: the hardware's invalid trap would always have to be enabled and distinguished from the invalid trap enable available from the programming environment; every invalid exception would have to trap and system software would have to insure that the invalid exception flag was not set for a Signaling NaN before continuing execution.]