1. Identification of the proposed change

1.1. Title: Mathematics Errors

1.2. MDC Proposer and Sponsor

Alan Frank	SC13/TG5
35 Gardner St.	Mathematics
Arlington, MA 02174	Chair: Ed de Moel
Phone: 617 647-4884x3280 (at Dynamic)	800 Nelson St.
Fax: 617 648-2824	Rockville, MD 20850
alf@matchups.com	demoel@radix.net

1.3. Motion: That SC13 present this document to the MDC for acceptance as a type A, superseding X11/SC13/97-1.

1.4. History:

August, 1997: X11/SC13/97-9: This document. No changes from previous document except in sections 1.4 and 7.
January, 1997: X11/SC13/97-1: No changes from previous document except in sections 1.4 and 7. Passed as subcommittee type A. 12:0:5.
September, 1996: X11/SC13/TG5/96-7: Editorial corrections made. Exact locations in standard specified in formalization. Passed as replacement type
B by SC13. 20:0:5.
August, 1996: X11/SC13/TG5/96-6: Strikes error condition on underflow (both in formalization and in examples and discussion)
based on amendment in subcommittee. Note: this obsoletes error code M93.
March, 1996: X11/SC13/TG5/96-3: Changes per SC13 straw poll. Fills in specific M codes. Passed as amended by SC13, 17:0:2.
October, 1995: X11/SC13/TG5/95-11 considered by TG. passed 2:0:2 Straw poll of SC13 supported adding underflow and complex results.
June, 1995: A straw poll of SC13 showed interest in proceeding in this direction.

1.5. Dependencies: None

2. Justification of proposed change

2.1. Needs

The current language standard is unclear in some cases as to how an implementation should proceed when it is unable to evaluate a mathematical operation.

2.2. Existing practice:

I have checked three implementations. All give an error on overflow; none give errors on underflow. One returns 0 for 0**anything. One gives an error on 0**(n≤0). The other does not support exponentiation. Both of the ones that support exponentiation give errors on -3**.5. However, they also give the same error on -3**.2, which should be valid (it's approximately -1.24573093961552). None distinguishes among exponentiation errors.

3. Description of the proposed change

3.1. General description

Errors would occur in the case of overflow on any mathematical operation, 0**(≤0), or exponentiation resulting in a complex (non-real) value.

3.2. Annotated example of use

```
SET N=1 FOR  SET N=N*2 ;will eventually get error M92
SET N=0**-1 ;will get error M9
SET N=0**0 ;will get error M94
SET N=-3**.5 ;will get error M95
```

3.3. Formalization:

Add in the standard:

- **Section II, clause 2.6**
  If the result of any mathematical operation is too large (either positive or negative) for the implementation to represent it to the accuracy specified earlier in this clause, an error will occur, with ecode equal to M92.
- **Section I, clause 7.2.1.2**
  On an attempt to compute 0**(a negative number), an error will occur, with ecode equal to M9.
- **Section I, clause 7.2.1.2**
  On an attempt to compute 0**0, an error will occur, with ecode equal to M94.
- **Section I, clause 7.2.1.2**
  On an attempt to compute the result of an exponentiation, the true value of which is a complex number with a nonzero imaginary part, an error will occur, with ecode equal to M95.

4. Implementation effects

4.1. Impact on existing user practices and investments

Programs which contain only valid mathematical calculations will continue to run without change. Programs which contain invalid mathematical calculations which were not trapped by their implementation will now crash. Note that in some cases, a program could be valid overall, yet still contain an invalid computation. For example:

```
READ "Length: ",L
READ "Width: ",W
IF L*W>1000 WRITE !,"Too big" QUIT
```

4.2. Impact on existing vendor practices and investments

Implementors necessarily already special-case these conditions, so triggering an error shouldn't be hard. One vendor wrote to me: "The level of effort is relatively minor (mostly tedious coding)."

4.3. Techniques and costs for compliance verification

Check that the examples in section 3.2 work as given.

4.4. Legal considerations: None known

5. Closely related standards activities

5.1. Other X11 proposals under consideration:

The task group requested an additional proposal in which a flag would be set on overflow instead of an error being triggered.

5.2. Other related standards efforts:

The task group has seen from other standards bodies' documents on arithmetic that there are some holes in our specifications. There are many approaches taken by other bodies, including errors, flags, and special values. We have been given direction not to take the "special value" approach.

5.3. Recommendations for coordinating liaison:

We should continue to look at documents from other standards bodies.

6. Associated Documents: None

7. Issues, Pros and Cons, and Discussion:

Recent Pros and Cons:

Pro:	Clarifies unspecified issues
	Better than previous proposals
	Much clearer editorial correction
Con:	*none*

Pros and Cons from the Subcommittee vote in March, 1996:

Pro:	1. Addresses issues [6]	
Con:	1. Incomplete [1]	*The document editor will be pleased to fill in any missing items; however, none were noted in the minutes of the subcommittee or task group.*
	2. Underflow should not be an error [1]	*Underflow wasn't an error in the document voted on!*