Sine Underflow Or Roundoff Errors

```
1 REM TEST FOR SIN FUNCTION ROUNDOFF OR UNDERFLOW ERRORS
10 LET X = 10/6
20 PRINT "ITERATION","  VAL OF X","  VAL OF SIN(X)"
30 FOR I = 1 TO 40
40 LET X = X / 10
50 PRINT I,X,SIN(X)
60 NEXT I
70 END
```

ITERATION	VAL OF X	VAL OF SIN(X)
1	.166667	.165896
2	.0166667	.0166659
3	1.66667E-03	1.66667E-03
4	1.66667E-04	1.66667E-04
5	1.66667E-05	1.66667E-05
6	1.66667E-06	1.66667E-06
7	1.66667E-07	1.66667E-07
8	1.66667E-08	1.66667E-08
9	1.66667E-09	1.66667E-09
10	1.66667E-10	1.66667E-10
11	1.66667E-11	1.66667E-11
12	1.66667E-12	1.66667E-12
13	1.66667E-13	1.66667E-13
14	1.66667E-14	1.66667E-14
15	1.66667E-15	1.66667E-15
16	1.66667E-16	1.66667E-16
17	1.66667E-17	1.66667E-17
18	1.66667E-18	1.66667E-18
19	1.66667E-19	1.66667E-19
20	1.66667E-20	1.66667E-20
21	1.66667E-21	1.66667E-21
22	1.66667E-22	1.66667E-22
23	1.66667E-23	1.66667E-23
24	1.66667E-24	1.66667E-24
25	1.66667E-25	1.66667E-25
26	1.66667E-26	1.66667E-26
27	1.66667E-27	1.66667E-27
28	1.66667E-28	1.66667E-28
29	1.66667E-29	1.66667E-29
30	1.66667E-30	1.66667E-30
31	1.66667E-31	1.66667E-31
32	1.66667E-32	1.66667E-32
33	1.66667E-33	1.66667E-33
34	1.66667E-34	1.66667E-34
35	1.66667E-35	1.66667E-35
36	1.66667E-36	1.66667E-36
37	1.66667E-37	1.66667E-37
38	1.66667E-38	1.66667E-38
39	0	0
40	0	0

A run of this program using Microsoft's MBASIC is given. This particular BASIC interpreter handles the SIN of small numbers quite well and doesn't cause trouble until the value of θ is less than 1.0E-38 (a decimal point followed by 37 zeros).

The program given depends on an adequate dynamic range in BASIC's handling of floating point arithmetical operations. It is well to remember that before you can use any mathematical operation in BASIC with confidence, you need to be aware of the range of numbers it can handle accurately.

Remember that a variable name alone, such as X or TREND will automatically be *single precision* (i.e. capable of storing no more than seven digits). Alternatively, variables can be specified as, or changed to, single precision by appending an exclamation mark, as in X! or TREND!. *Double precision* variables (which can store 17 digits) are specified by appending a hash sign, as in X# or TREND#. *Integer* variables (which can store only whole numbers) are specified in many versions of BASIC by appending a per cent sign, as in X% or TREND%.

We end this article with a short program that lets you test how many digits can be stored in a variable in your version of BASIC, together with a print-out of the program when run using Microsoft BASIC. There are two versions, one for testing small numbers and one for large ones. The print-out for small numbers shows that as the numbers become very small (less that 3.3 × 1OE-38) BASIC rounds the numbers off to zero. For large numbers (greater than 3.3 × 1OE37) an overflow occurs and the results are unreliable. If you need to work with very large or very small numbers you may need to write special arithmetic routines to overcome these limitations.

Small Numbers In BASIC

```
 1 REM  TESTS HANDLING OF SMALL NUMBERS IN BASIC
10 LET X# =.000333333333334#
20 PRINT "ITERATION","  DBL PREC","  ","  SNGL PREC"
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# / 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERATION	DBL PREC	SNGL PREC
1	.0000033333333333	3.33333E-06
2	.0000003333333333	3.33333E-07
3	3.333333333D-08	3.33333E-08
4	3.333333333D-09	3.33333E-09
5	3.333333333D-10	3.33333E-10
6	3.333333333D-11	3.33333E-11
7	3.333333333D-12	3.33333E-12
8	3.333333333D-13	3.33333E-13
9	3.333333333D-14	3.33333E-14
10	3.333333333D-15	3.33333E-15
11	3.333333333D-16	3.33333E-16
12	3.333333333D-17	3.33333E-17
13	3.333333333D-18	3.33333E-18
14	3.333333333D-19	3.33333E-19
15	3.333333333D-20	3.33333E-20
16	3.333333333D-21	3.33333E-21
17	3.333333333D-22	3.33333E-22
18	3.333333333D-23	3.33333E-23
19	3.333333333D-24	3.33333E-24
20	3.333333333D-25	3.33333E-25
21	3.333333333D-26	3.33333E-26
22	3.333333333D-27	3.33333E-27
23	3.333333333D-28	3.33333E-28
24	3.333333333D-29	3.33333E-29
25	3.333333333D-30	3.33333E-30
26	3.333333333D-31	3.33333E-31
27	3.333333333D-32	3.33333E-32
28	3.333333333D-33	3.33333E-33
29	3.333333333D-34	3.33333E-34
30	3.333333333D-35	3.33333E-35
31	3.333333333D-36	3.33333E-36
32	3.333333333D-37	3.33333E-37
33	3.333333333D-38	3.33334E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

Large Numbers In BASIC

```
 1 REM  TESTS HANDLING OF LARGE NUMBERS IN BASIC
10 LET X# = 3.3333333333334#
20 PRINT "ITERATION","  DBL PREC","  ","  SNGL PREC"
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# * 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERATION	DBL PREC	SNGL PREC
1	33.333333333334	33.3333
2	333.33333333334	333.333
3	3333.3333333334	3333.33
4	33333.333333334	33333.3
5	333333.33333334	333333
6	3333333.3333334	3.33333E+06
7	33333333.333334	3.33333E+07
8	333333333.33334	3.33333E+08
9	3333333333.3334	3.33333E+09
10	33333333333.334	3.33333E+10
11	333333333333.34	3.33333E+11
12	3333333333333.4	3.33333E+12
13	33333333333334	3.33333E+13
14	333333333333334.	3.33333E+14
15	3333333333333340	3.33333E+15
16	3.3333333333333340D+16	3.33333E+16
17	3.3333333333333340D+17	3.33333E+17
18	3.3333333333333340D+18	3.33333E+18
19	3.3333333333333340D+19	3.33333E+19
20	3.3333333333333340D+20	3.33333E+20
21	3.3333333333333340D+21	3.33333E+21
22	3.3333333333333340D+22	3.33333E+22
23	3.3333333333333340D+23	3.33333E+23
24	3.3333333333333340D+24	3.33333E+24
25	3.3333333333333340D+25	3.33333E+25
26	3.3333333333333340D+26	3.33333E+26
27	3.3333333333333340D+27	3.33333E+27
28	3.3333333333333340D+28	3.33333E+28
29	3.3333333333333340D+29	3.33333E+29
30	3.3333333333333340D+30	3.33333E+30
31	3.3333333333333350D+31	3.33333E+31
32	3.3333333333333350D+32	3.33333E+32
33	3.3333333333333340D+33	3.33333E+33
34	3.3333333333333350D+34	3.33333E+34
35	3.3333333333333350D+35	3.33333E+35
36	3.3333333333333350D+36	3.33333E+36
37	3.3333333333333350D+37	3.33333E+37
38	1.70141183460469 3D+38	1.70141E+38
39	1.70141183460469 3D+38	1.70141E+38
40	1.70141183460469 3D+38	1.70141E+38