terak
CORPORATION

SMATH
STRING ARITHMETIC EXTENSIONS
FOR
MUBASIC / RT-11

TERAK Publication Number 60-0018-001

REVISED

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

The information in this document is subject to change without notice and should not be construed as a commitment by TERAK Corporation. TERAK Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

TERAK Corporation assumes no responsibility for the use or reliability of its' software on equipment that is not supplied by TERAK.

The following are trademarks of TERAK Corporation:

TERAK

The following are trademarks of Digital Equipment Corporation:

RT-11

The following are trademarks of the TRUSTEES OF DARTMOUNT COLLEGE:

BASIC

First Release 5 Nov 76
Revised 26 Jun 79

SOFTWARE SPECIFICATION

STRING ARITHMETIC PACKAGE

1. PRELIMINARY

MUBASIC, or BASIC-11, currently carries 16 bits of accuracy *in* integer computation (restricted to + and -) and 24 bits of accuracy in the mantissa of all floating point calculations (all *, /, and +, - when result overflows). All transcendental functions are performed in floating point. These accuracies are equivalent to 5-1/2 digits (integer) and 6-1/2 digits floating. Accounting functions require further precision: at least 12 digits, but do not generally require floating point. An additional restriction is that an integer variable can only be printed to six digits by the BASIC print routines; to print more digits, concatenated prints or strings must be used. Since most accounting data will be carried as strings to avoid trucation to 6 digits, the requirement exists for four function arithmetic which operates directly on strings.

2. CALL SYNTAX

Five subroutines are provided in the SMATH package: ADD$, SUB$, ADR$, MUL$ and DIV$. These may be called implicitly (by reference only) or explicitly (by the CALL statement). The former is recommended as it is more efficient. The operations are as follows:

IMPLICIT CALL	FUNCTION
ADD$(B$,C$,A$)	A$ = B$ + C$
SUB$(B$,C$,A$)	A$ = B$ - C$
MUL$(B$,C$,A$)	A$ = B$ * C$
DIV$(B$,C$,A$)	A$ = B$ / C$
ADR$(B$,C$,A$)	A$ = B$ + C$ WITH A$ TRUNCATED AT LEAST SIGNIFICANT NON-ZERO DIGIT OF B$ PLUS 1

Each call statement must reference three arguments, separated by commas (as shown above). The A$ (target) argument must be a string variable. The B$ or C$ (operand) arguments may be string variables, literal strings, or string expressions. Violation of these rules will produce a ?ARG or ?SYN error stop.

EXAMP1.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO COMPARE TWO ARITHMETIC STRINGS
2 REM AND BRANCH IF VALUE (B$) < VALUE (C$)
3 REM
10 REM ENTER WITH B$ & C$ LOADED WITH VALID NUMERIC STRINGS
15 REM
20 SUB$(B$,C$,A$) \ REM      A$= B$ - C$
30 IF SEG$(A$,1,1)="-" THEN 1000
35 REM
40 REM IF RESULT IS NEGATIVE, BRANCH TO 1000
```

EXAMP2.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO COMPUTE    .015 * X$ + Y$
2 REM AND STORE RESULT INTO Z$
3 REM
10 MUL$(X$,".015",Z$)
20 ADD$(Y$,Z$,Z$)
```

EXAMP3.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO DIVIDE TWO NUMBERS, CHECK FOR ERRORS,
2 REM AND ROUND THE ABSOLUTE VALUE OF THE RESULT TO CENTS
3 REM
10 DIV$(B$,C$,Q$) \ REM       Q$ = B$ / C$
15 IF Q$="?" THEN PRINT "?DIV ERR?" \ STOP \ REM       CHECK FOR DIVISION ERR
20 ADR$(SEG$(Q$,2,255),".005",A$) \ REM       REMOVE SIGN CHARACTER  AND ROUND
```

3. OPERAND STRING SYNTAX

In general, strings which are valid arguments for the VAL function are valid for the SMATH functions. The following rules apply to ADR$, ADD$, SUB$, MUL$ operands (2nd and 3rd arguments). DIV$ places a few additional restrictions upon its operands. Operands which violate the following rules will cause a one character string: "?" to be returned in A$, and the program will continue running. Since "?" is an illegal operand string, subsequent computation on A$, using SMATH will propogate "?" results.

B$ and C$ must be strings of zero to 255 digits, period, minus sign, plus sign, spaces, and the characters "I" thru "R" (IBM sign). Any other character will cause a "?" returned in A$. A null string will be interpreted as zero.

SYNTAX rules are:

- Spaces are ignored except in length of string
- No more than one period
- Period, if present, must follow any + or -
- Only one of + or - may be present
- If none of + or - are present, + is assumed
- + or - may not occur after first digit (incl. Ø)
- One of the characters "I" thru "R" may appear once only. It will be interpreted as "Ø" thru "9" and the value represented will be negative, as if a minus sign were present. This "IBM sign" will terminate the string.
- A "-" may not appear in the same string with an IBM sign.
- Leading & trailing zeroes are ignored except:
- If no period is present, a period is assumed at the end of the string.
- Strings are considered terminated on right by: Last character or 16th character, or IBM sign.

DIV$ places the following restrictions upon its operands. C$ in DIV$(B$,C$,A$) must not represent zero. A "?" will be returned if division by zero is attempted. The result from division must be representable to sixteen digits of significance, with sixteen decades of magnitude, i.e. the result must be between

9999999999999999000000000000000

and

.00000000000000001000000000000000

or a "?" return will occur, avoiding a loss in significance, and implying the error lies in the operands.

(Note that though the trailing zeroes in the second number will be truncated, 16 digits of significance is implied.)

The following are examples of valid and invalid operand strings. △ represents a space.

Example	Interpretation
△△ △△1 (16 spaces)	VALID...truncated at 16 characters = null string = Ø.
△△1△△2△△3△	VALID...123.Ø
△ -12ØØ△△	VALID...-12ØØ.Ø
+128.E2	INVALID...illegal char
△△△△△	VALID.....all spaces = null string = Ø.
12ØI%#2	VALID...-12ØØ.Ø (IBM sign terminates string)
6R	VALID...-69 (IBM sign)

4. DESTINATION STRING SYNTAX

In general, the destination string will conform to the rules for operand strings except in length. The result of ADD$, SUB$, MUL$ or DIV$ will be a string of from two to thirty-three characters. All leading zeroes left of the point and trailing zeroes right of the point will be removed, and the number will be left justified with no space characters. A decimal point will be inserted unless it would be the last character.

Zero will always be represented as the string " Ø", when using +- signs or "Ø" when using IBM signs.
Because of point alignment, addition or subtraction can produce a destination string of up to 33 characters, e.g. a 33 character string will occur in

```
SUB$(".123412341234123","1234123412341234",A$)
```

Multiplication can produce a destination string which is smaller and/or larger than that allowed by division. This is because no loss in significance occurs in

```
MUL$(".0000000000000001",".0000000000000001",A$)
```

or in

```
MUL$("9876987698769876","9876987698769876",A$)
```

Multiplication will produce a destination string of from 2 to 33 characters.

Division will always produce a quotient with 16 digits of significance. Depending upon point alignment, the destination string will be from 2 to 33 characters. Prior to truncation, trailing zeroes are considered significant. The destination magnitude (absolute value) will be between

```
9999999999999999000000000000000
```

and

```
.00000000000000001000000000000000
```

i.e., 16 digits of significance and ±16 decades. If quotient is outside this range, overflow or underflow occurred, and "?" is returned in A$.

No remainder is generated.

Signs will be represented in one of two ways, depending upon the type of sign last encounted as a negative operand. If "-" was last used, all results will have a leading space or "-" character. If an IBM sign was last used, all results will not have a leading space

or "-", and the last character of the string will be returned as "I" thru "R" to indicate a negative result. Upon loading MUBASIC, the space and "-" convention is assumed, but may be changed by passing an IBM sign in one of the operands. See Appendix.

The resultant string from the ADR$ call is an exception to some of the above rules.

ADR$ (B$,C$,A$) performs an addition (like ADD$), then truncates A$ at the digit position corresponding to the least significant non-zero digit position of C$, plus 1. Thus, if C$ = ".ØØ5", B$ will be rounded to nearest .Ø1 and placed in A$. (or if C$ = "5Ø", B$ will be rounded to hundreds and placed into A$). ADR$ will override the normal suppression of trailing zeroes, if necessary. If C$ is a null string, A$ will be truncated to the units position.

Examples:

Call	Result (A$)
ADR$("1234.5678",".05",A$)	1234.6
ADR$("1234",".005",A$)	1234.00
ADR$("1234.56","50",A$)	1200
ADR$("123.3","-50",A$)	0

5. ALGORITHMS

All calculations are made in excess - 60_8 (ASCII) BCD. Addition or subtraction will product a true result with carry (borrow) correction required. Data is manipulated one digit per byte. Multiplication of individual digits is accomplished in 2's compliment binary. Division is accomplished by the nonrestoring method. The remainder will be uncorrected (sign may be different from quotient sign) and is <u>not</u> returned.

6. ERRORS

Since up to 32 characters (sans sign character) are allowed for output, no arithmetic error can occur with addition, subtraction, or multiplication. Two errors can occur under division. First, attempted division by zero will produce the "?" return.

Second, if the quotient cannot be correctly represented in 16 digits significance and 16 decades of magnitude, the "?" error return will occur.

Any of the syntax errors in the call statement, or failure to supply the correct number or type of arguments will result in the ?SYN or ?ARG error and the program will stop.

7. DELIVERY

SMATH is delivered on floppy disk as a set of eight object modules, as follows:

	FUNCTIONS SUPPORTED						
MODULE NAME	ADD$	SUB$	ADR$	MUL$	DIV$	IBM SIGN	SIZE WORDS$_{10}$
SMATH.OBJ	X	X	X	X	X	X	696
SMATHØ.OBJ	X	X				X	407
SMATH1.OBJ	X	X	X			X	442
SMATH2.OBJ	X	X		X	X	X	661
SMATH3.OBJ	X	X	X	X	X		671
SMATH4.OBJ	X	X					382
SMATH5.OBJ	X	X	X				417
SMATH6.OBJ	X	X		X	X		636

The user should select one of these to provide the required features. In addition, two call interface files are provided:

MUBISA.OBJ
MUBISA.MAC

The selected SMATH module and MUBISA.OBJ, substituted for MUBI, are linked into MUBASIC. In addition, MUBC (call support) should be substituted for MUBNC (no call support). Linked overhead for MUBISA.OBJ (SMATH related) is 15_{10} words. The source file is provided for users who may have other call functions supported. In this case, MUBISA.MAC should be merged with the users call interface file, using the editor, and then assembled to an object file to be used when linking MUBASIC.

APPENDIX

A.1 Removal of IBM sign feature

A global symbol is included in the SMATH object modules (SMATH thru SMATH2) to locate the IBM sign option word. The symbol is "IBMOPT" and will appear at link time, in the link map, under the SMATH CSECT. (1) The IBM sign support may be defeated by patching this location to a NOP. For example, assuming MUBASIC has been linked with SMATH in the root section:

```
.R PATCH
*FILE NAME -- MUBASA/O <CR>
*?NOT IN PROGRAM BOUNDS?     (2)
*400;B
*IBMOPT/101445 240 <CR>
*E
```

(1) The IBMOBT global will not appear in the SMATH object modules without IBM SIGN.

(2) The error message and setting of the "B" register is peculiar only to files linked with other than 1000 as the bottom address.

A.2 Use of SMATH for conversion of formats

SMATH is useful to format strings by inserting decimal points or trailing zeroes or by converting sign formats. The following examples will illustrate these functions.

EXAMP4.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO TAKE A REAL VARIABLE, CONVERT TO STRING, AND
  REM INSERT A DECIMAL POINT. INTERPRETING THE NUMBER AS CENTS
3 REM
10 MUL$(STR$(INT(X)),".01",A$) \ REM INTEGERIZE, STRING-IZE, AND MPY BY .01
20 ADR$(A$,".005",A$) \ REM FORCE TWO DECIMAL POSITIONS (INCL. ZEROES)
30 A$="$ "&A$ \ REM ADD A DOLLAR SIGN (NOTE...POSSIBLE SIGN HASNOT BEEN REMOVED
```

EXAMP5.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO INPUT A NUMBER FROM THE OPERATOR, AND CHECK IT FOR
2 REM BOTH SYNTAX, AND RANGE
3 REM
5 PRINT "ENTER VALUE TWIXT $1 AND $1000";
10 INPUT #0:X$
20 SUB$(X$,"1",Z$) \ REM VALUE ENTERED MUST BE GREATER THAN $1.00
30 IF Z$="?" THEN PRINT Z$ \ GO TO 5 \ REM "?" -> ILLEGAL SYNTAX IN X$
40 IF SEG$(Z$,1,1)="-" THEN PRINT "?TOO SMALL?" \ GO TO 5
50 SUB$(X$,"1000",Z$) \ REM VALUE ENTERED MUST BE LESS THAN $1000.00
60 IF SEG$(Z$,1,1)=" " THEN PRINT "?TOO BIG?" \ GO TO 5
70 REM
80 REM LINES 40,50,60 COULD ALSO HAVE USED THE VAL FUNCTION FOR COMPARISON
```

EXAMP6.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO CONVERT 100 STRINGS IN VIRTUAL FILE VF1$(99)
2 REM FROM SPACE/MINUS SIGN NOTATION TO IBM SIGN NOTATION
3 REM AND STORE INTO VIRTUAL FILE VF7$(99)
10 FOR I=0 TO 99
20 ADD$("I",VF1(I),X$) \ REM LAST SIGN TYPE SCANNED BY SMATH WILL BE IBM "-0"
30 REM ZERO IS ADDED TO X$ ( = X$ ) AND X$ IS IBM SIGNED
40 VF7(I)=X$ \ REM VIRTUAL FILE TARGET VARIABLE CAN ONLY APPEAR
50 REM                ON LEFT OF AN ASSIGNMENT STATEMENT, NOT AS ARGUMENT
60 NEXT I
```

EXAMP7.BAS 30-DEC-76 PAGE 1

```
1 REM ROUTINE TO DO THE OPPOSITE OF ABOVE ROUTINE (EXAMP6)
10 FOR I=0 TO 99
20 ADD$("-0",VF1(I),X$) \ REM LAST SIGN TYPE SCANNED BY SMATH WILL BE "-"
30 VF7(I)=X$
40 NEXT I
```