

LEN

Format

LEN(X\$)

Purpose

Returns the number of characters in string X\$.

Remarks

The number returned by this function also indicates any blanks or non-printable characters included in the string (such as the return and cursor control codes).

Example

```
10 FOR I=1 TO 8
20 A$=INPUT$(I)
30 PRINT A$:PRINT "LENGTH OF A$ IS";LEN(A$)
40 NEXT
```

```
RUN
A
LENGTH OF A$ IS 1
AB
LENGTH OF A$ IS 2
ABC
LENGTH OF A$ IS 3
ABCD
LENGTH OF A$ IS 4
ABCDE
LENGTH OF A$ IS 5
ABCDEF
LENGTH OF A$ IS 6
ABCDEFG
LENGTH OF A$ IS 7
ABCDEFGH
LENGTH OF A$ IS 8
OK
```

LOC

Format

LOC (< file number >)

Purpose

With random access files, returns the record number which will be used by the next GET or PUT statement which is executed without specifying a record number; with sequential files, returns the number of file sectors (128-byte blocks) which have been read or written since the file was opened.

Remarks

This function can be used to control the flow of program execution according to the number of records or file sectors which have been accessed by a program since the file was opened.

Example

```
10 ON ERROR GOTO 200
20 OPEN "R", #1, "A:LOCTEST", 5
30 PRINT "OUTPUT"
40 FIELD #1, 5 AS A#
50 FOR A=1 TO 100
60 LSET A#=STR$(A)
70 PRINT STR$(A);
80 PUT#1, A
90 NEXT
100 PRINT
110 CLOSE
120 OPEN "R", #1, "A:LOCTEST", 5
130 PRINT "INPUT"
140 FIELD #1, 5 AS A#
150 IF LOC(1) > 50 THEN 190
160 GET#1
170 PRINT A#;
180 GOTO 150
190 ERROR 230
200 IF ERR=230 THEN PRINT:PRINT "INPUT PAST LIMIT"
210 END
```

RUN

OUTPUT

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

INPUT

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51
```

INPUT PAST LIMIT

Ok

LOF

Format LOF (<file number>)

Purpose Returns the size of a file.

Remarks When the file specified in <file number> is a disk file, the LOF function returns the size of that file. If the file has been opened in the "R" mode, the size of the file is calculated using the highest record number of that file and the record size under which the file was opened. If the file was opened in the sequential mode, the size is calculated based on the number of records and a record size of 128 bytes.

When the specified file is the QX-10's CMOS RAM, the LOF function returns the size of the CMOS RAM file in bytes.

Example

```
10 OPEN"0",#1,"A:LOFTEST"  
20 FOR I=1 TO 50  
30 PRINT#1,I  
40 NEXT  
50 PRINT "NUMBER OF SECTORS=";LOF(1)  
60 FOR I=51 TO 100  
70 PRINT#1,I  
80 NEXT  
90 PRINT "NUMBER OF SECTORS=";LOF(1)  
100 FOR I=101 TO 200  
110 PRINT #1,I  
120 NEXT  
130 PRINT "NUMBER OF SECTORS=";LOF(1)  
140 CLOSE
```

```
RUN  
NUMBER OF SECTORS= 2  
NUMBER OF SECTORS= 4  
NUMBER OF SECTORS= 10  
OK
```

NOTE:

When a disk file is specified in the LOF function, the disk containing that file is accessed in order to determine its size. Therefore, execution time may be greatly increased if this function is constantly used in a program which searches for records in a random file. This can be avoided by assigning the value returned by LOF to a variable before entering the search loop, then referring to that variable within the loop instead of to the LOF function itself.

Under CP/M, the size of files is returned in units of 128 bytes. In consequence of this, it should be noted that the value returned by the LOF function for a random file will be larger than the actual file size if the file has a record length of less than 128 bytes.

LOG

Format

LOG(X)

Purpose

Returns the natural logarithm of X.

Remarks

The value specified for X must be greater than zero. LOG(X) is calculated to the precision of the numeric type of expression X.

Example

```
10 PRINT "X"; "LOG(X)"
20 FOR I=1 TO 100 STEP 10
30 PRINT; LOG(I)
40 NEXT
```

RUN

X LOG(X)

0

2.3979

3.04452

3.43399

3.71357

3.93183

4.11087

4.26268

4.39445

4.51086

OK

LPOS

Format

LPOS(X)

Purpose

Returns the current position of the buffer pointer in the printer output buffer.

Remarks

The maximum value returned by LPOS is determined by the line width which has been set by the WIDTH LPRINT statement, and does not necessarily correspond to the physical position of the print head. X is a dummy argument, and may be specified as any numeric expression.

Example

```
10 LPRINT "0";  
20 IF LPOS(X)>10 THEN 40  
30 GOTO 10  
40 LPRINT "END"  
50 END  
0000000000END
```

MID\$

Format

MID\$(X\$,I[,J])

Purpose

Returns the character string from the middle of string expression X\$ which consists of the J characters beginning with the Ith character.

Remarks

I and J must be integer expressions in the range from 1 to 255. If J is omitted, or there are fewer than J characters to the right of the Ith character, the string returned consists of all characters to the right of the Ith character. If the value specified for I is greater than the number of characters in X\$, MID\$ returns a null string.

See also

LEFT\$, RIGHT\$

Example

```
10 A$="ABCDEFGHijklmnop"  
20 FOR I=1 TO 11 STEP 2  
30 PRINT "4 CHARACTERS BEGINNING WITH CHARACTER"; I  
40 PRINT MID$(A$, I, 4)  
50 NEXT I
```

```
RUN  
4 CHARACTERS BEGINNING WITH CHARACTER 1  
ABCD  
4 CHARACTERS BEGINNING WITH CHARACTER 3  
CDEF  
4 CHARACTERS BEGINNING WITH CHARACTER 5  
EFGH  
4 CHARACTERS BEGINNING WITH CHARACTER 7  
GHIJ  
4 CHARACTERS BEGINNING WITH CHARACTER 9  
IJKL  
4 CHARACTERS BEGINNING WITH CHARACTER 11  
KLMN  
OK
```

MKI\$/MKSS\$/MKD\$

Format

MKI\$(<integer expression>)
MKSS\$(<single precision expression>)
MKD\$(<double precision expression>)

Purpose

Converts numeric values to string values for storage in random access disk files.

Remarks

Numeric values must be converted to string values before they can be placed in a random file buffer by a LSET or RSET statement for storage with a PUT statement. MKI\$ converts integers to 2-byte strings, MKSS\$ converts single precision numbers to 4-byte strings, and MKD\$ converts double precision numbers to 8-byte strings.

Unlike the STR\$ function, which produces an ASCII character string representation of the numeric value specified in its argument, these functions convert numeric values to characters whose ASCII codes correspond to the binary coded decimal values with which the numbers would be stored in memory. Less disk space is required for storage of numbers which are converted to strings with the MKI\$/MKSS\$/MKD\$ functions.

See also

CVI/CVS/CVD, STR\$

Example

```
10 OPEN"R",#1,"A:MKITEST",8
20 FIELD#1,8 AS A$
30 FOR I=1 TO 4
40 LSET A$=MKI$(I)
50 PUT#1,I
60 NEXT
70 FOR I=5 TO 8
80 LSET A$=MKS$(I)
90 PUT#1,I
100 NEXT
110 FOR I=9 TO 12
120 LSET A$=MKD$(I)
130 PUT#1,I
140 NEXT
150 CLOSE
160 OPEN"R",#1,"A:MKITEST",8
170 FIELD#1,8 AS A$
180 PRINT "INTEGER VALUES"
190 FOR R=1 TO 4
200 GET#1,R
210 PRINT CVI(A$);
220 NEXT
230 PRINT
240 PRINT "SINGLE PRECISION VALUES"
250 FOR R=5 TO 8
260 GET#1,R
270 PRINT CVS(A$);
280 NEXT
290 PRINT
300 PRINT "DOUBLE PRECISION VALUES"
310 FOR R=9 TO 12
320 GET#1,R
330 PRINT CVD(A$);
340 NEXT
350 CLOSE
```

RUN

INTEGER VALUES

1 2 3 4

SINGLE PRECISION VALUES

5 6 7 8

DOUBLE PRECISION VALUES

9 10 11 12

OK

OCT\$

Format OCT\$(X)

Purpose Returns a string which represents the octal value of X.

Remarks The numeric expression specified in the argument is rounded to the nearest integer value before it is evaluated.

Example

```
10 PRINT " X","OCT(X)"," X","OCT(X)"
20 FOR X=1 TO 6 STEP .5
30 PRINT X,OCT$(X),X+.6,OCT$(X+.6)
40 NEXT
```

```
RUN
X      OCT(X)  X      OCT(X)
1      1      7      7
1.5    2      7.5    10
2      2      8      10
2.5    3      8.5    11
3      3      9      11
3.5    4      9.5    12
4      4      10     12
4.5    5      10.5   13
5      5      11     13
5.5    6      11.5   14
6      6      12     14
Ok
```

PEEK

Format

PEEK(I)

Purpose

Returns the byte of data contained in memory location I as a decimal integer in the range from 0 to 255.

Remarks

The integer value specified for I must be in the range from 0 to 65535. This function is complementary to the POKE statement described in Chapter 3.

Example

```
10 FOR X=2624 TO 2643
20 POKE X, I
30 I=I+1
40 NEXT
50 FOR X=2624 TO 2640 STEP 4
60 PRINT PEEK(X), PEEK(X+1), PEEK(X+2), PEEK(X+3)
70 NEXT
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

Ok

PEN

Format PEN (<function>)

Purpose Returns information regarding the current state of the light pen.

Remarks A number from 0 to 4 is specified for <function>; these numbers have the following meanings.

PEN(0) - Trigger sense

This function checks whether the light pen has been pressed since the last time PEN(0) was executed. If the light pen has been pressed, -1 is returned; otherwise, 0 is returned.

Note that this function is used to determine whether the light pen has been pressed, rather than whether it is currently being pressed. Also, note that the trigger sense status is reset once it has been read by executing PEN(0).

The screen coordinates returned by PEN(1) to PEN(4) are also read in when PEN(0) is executed and -1 (TRUE) is returned; therefore, it is necessary to confirm that PEN(0) returns TRUE before executing any of PEN(1) to PEN(4).

PEN(1)

This function returns the horizontal graphic screen coordinate at which the light pen was pressed as a value from 0 to 624. The value returned will be a multiple of 16 (because the light pen's resolution in the horizontal direction is 16 dots).

PEN(2)

This function returns the vertical graphic screen coordinate at which the light pen was pressed as a value from 0 to 399.

PEN(3)

This function returns the horizontal character coordinate at which the light pen was pressed. The value returned is a multiple of 2 (because the light pen has a horizontal resolution of two characters), and will be in the range from 1 to 40 (when the screen is in the WIDTH 40 mode) or 1 to 80 (when the display is in the WIDTH 80 mode).

PEN(4)

This function returns the vertical character coordinate at which the light pen was pressed as a number from 1 to 20.

Note that the PEN ON statement must be executed before the PEN functions can be used; otherwise, an "Illegal function call" error will occur. Also, it is recommended that a PEN OFF statement be executed when light pen processing has been discontinued; this is to disable light pen interrupts.

Example

```
10 CLS:FOR Y=0 TO 399:LINE (0,Y)-(639,Y)
20 NEXT
30 PEN ON
40 IF PEN(0)<>-1 THEN 40
50 X=PEN(1):Y=PEN(2)
60 FOR I=X TO X+16:FOR J=Y TO Y+16:PRESET (I,J):NEXT J,I
70 GOTO 40
```

NOTE:

After PEN ON has been executed and TRUE has been returned by PEN(0), no further graphic coordinates are read in when the light pen is pressed until after PEN(0) is accessed again.

POINT

Format

POINT (horizontal coordinate,vertical coordinate)

Purpose

Returns the color of the display dot at the specified graphic screen coordinates.

Remarks

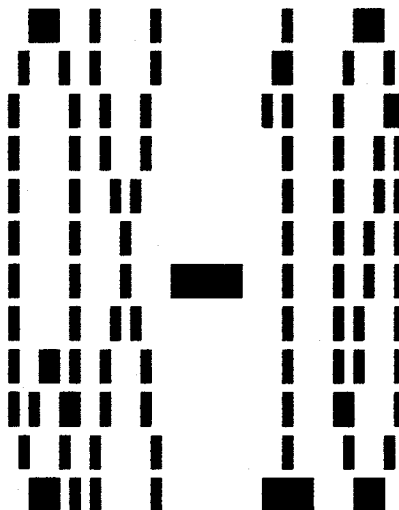
This function returns the color code corresponding to the color of the display dot at the specified graphic coordinates.

In the color mode, this is returned as a value from 0 to 7; in the black and white mode, 0 is returned for black and 7 is returned for white.

-1 will be returned if the coordinates specified are outside the screen area.

Example

```
10 CLS:DEFINT A-Z
20 DIM P(40,15)
30 PRINT "OX-10"
40 FOR Y=2 TO 15
50 FOR X=0 TO 38
60 P(X,Y)=POINT(X,Y)
70 NEXT X,Y
80 CLS
90 FOR Y=2 TO 15
100 LOCATE 19,Y+1:FOR X=0 TO 38
110 IF P(X,Y) THEN A$="■" ELSE A$=" "
120 PRINT A$;
130 NEXT X,Y
```



Ok

POS

Format

POS (< file no. >)

Purpose

Returns the current horizontal position of the cursor or of the buffer pointer in the file output buffer.

Remarks

This function returns the horizontal position of the cursor when "0" is specified for < file no. >. The value returned will be in the range from 1 to 40 (for the WIDTH 40 mode) or 1 to 80 (for the WIDTH 80 mode).

When < file no. > is other than "0", this function returns the current output column position of the specified file. This file must previously have been opened for sequential output or random access; the value returned for a file which has been opened for sequential input is meaningless.

When < file no. > is other than "0", the value returned by this function is a number in the range from 1 to 255; 1 is returned immediately after the file is opened or a carriage return is output.

When the file opened is "LPT0:", this function returns the same value as the LPOS function.

Example

```
10 *PRESS CURSOR CONTROL KEYS
20 CLS
30 LOCATE 40,10
40 A#=INPUT$(1)
50 IF ASC(A#)>=28 OR ASC(A#)<=31 THEN PRINT A#;"*";
CHR$(29);:ELSE 40
60 LOCATE POS(X),CSRLIN
70 GOTO 40
```

```
XXXXXXXXXXXXXXXXX
*                *
*  XXXXXXXX      *
* *      *      *
* *  XXX  *      *
* *      *      *
* XXXXXXXXXXXXXXX
*
*
```

RIGHT\$

Format RIGHT\$(X\$,I)

Purpose Returns a string composed of the I characters making up the right end of string X\$.

Remarks The value specified for I must be in the range from 0 to 255. If I is greater than LEN(X\$), the entire string will be returned. If I=0, a null string of zero length will be returned.

See also LEFT\$, MID\$,

Example

```
10 A$="QX-10 MFBASIC"  
20 FOR I=1 TO 13  
30 PRINT RIGHT$(A$,I)  
40 NEXT
```

```
C  
IC  
SIC  
ASIC  
BASIC  
FBASIC  
MFBASIC  
 MFBASIC  
0 MFBASIC  
10 MFBASIC  
-10 MFBASIC  
X-10 MFBASIC  
QX-10 MFBASIC  
Ok
```

RND

Format

RND[(X)]

Purpose

Returns a random number between 0 and 1.

Remarks

Unless the RANDOMIZE statement is executed, this function generates the same sequence of random numbers each time the program is executed. If X is omitted or the number specified for X is greater than 0, the next random number in the sequence is generated each time RND is executed. If 0 is specified for X, RND repeats the last random number generated. If the number specified for X is less than 0, RND starts a new sequence whose initial value is determined by the value specified for X.

See also

RANDOMIZE

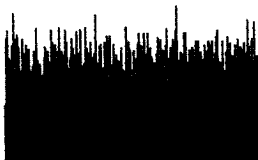
Example 1

```
10 CLS
15 DIM A(200)
16 LOCATE 20,1,0:PRINT "START:"TIME#
17 FOR I=0 TO 10000
18 LOCATE 1,1,0:PRINT I
40 X=RND*200
50 A(X)=(A(X)+1):PSET (X,200-A(X))
60 NEXT
70 LOCATE 40,1,0:PRINT "END:"TIME#
80 END
```

10000

START:12:23:52

END:12:31:07



SCREEN

Format

SCREEN (<horizontal position>, <vertical position> [, <function>])

Purpose

This function reads a character code from the specified position on the screen.

Remarks

The SCREEN function returns the code or type of the character currently being displayed at the character coordinates specified by <horizontal position>, <vertical position>. <function> is specified as a numeric expression in the range from 0 to 255. An "Illegal function call" error will result if a value outside this range is specified.

When <function> is "0", the SCREEN function returns the character code of the character displayed in the specified position. If the character in that position is a 2-byte character, the first or second byte of that character is returned (depending on whether the first or second half of the character is displayed in the specified position).

When <function> is other than "0", the SCREEN function returns the type of the character being displayed at the specified position. In this case, "0" is returned for 1-byte characters; for 2-byte characters, 1 is returned for the first byte and 2 is returned for the second byte.

Example

```
10 CLS
20 FOR I=65 TO 122
30 PRINT CHR$(I);
40 NEXT
50 PRINT:PRINT:PRINT
60 FOR I=58 TO 1 STEP -1
70 A=SCREEN(I,1)
80 PRINT CHR$(A);
90 NEXT
```

```
ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_ 'abcdefghijklmnopqrstu vwxyz
```

```
zyxwutsrqponmlkjihgfedcba'^_\ZYXWUTSRQPONMLKJIHGFEDCBA
```

```
Ok
```

SGN

Format

SGN(X)

Purpose

Returns the sign of numeric expression X.

Remarks

If $X > 0$, SGN(X) returns 1. If $X = 0$, SGN(X) returns 0. If $X < 0$, SGN(X) returns -1. Any numeric expression can be specified for X.

Example

```
10 CLS
20 G=1:F=1:X=1:Y=1
30 A=RND-.5
40 B=RND-.5
50 IF SGN(A)=-1 AND X>2 THEN X=X-1 ELSE IF SGN(A)=1
   AND X<78 THEN X=X+1
60 IF SGN(B)=-1 AND Y>2 THEN Y=Y-1 ELSE IF SGN(B)=1
   AND Y<18 THEN Y=Y+1
70 LOCATE G,F:PRINT " ":G=X:F=Y:LOCATE X,Y:PRINT "*"
80 FOR T=1 TO 100:NEXT
90 GOTO 30
```

SIN

Format SIN(X)

Purpose Returns the sine of X, where X is an angle in radians.

Remarks The sine of angle X is calculated to the precision of the type of numeric expression specified for X.

Example 1

```
10 CLS
20 INPUT "ENTER ANGLE IN DEGREES";A
30 PI=3.1416
40 D=PI/180
50 PRINT "SIN(";A;")=";SIN(A*D)
60 GOTO 20
```

```
ENTER ANGLE IN DEGREES? 0
SIN( 0 )= 0
ENTER ANGLE IN DEGREES? 30
SIN( 30 )= .500001
ENTER ANGLE IN DEGREES? 45
SIN( 45 )= .707108
ENTER ANGLE IN DEGREES? 60
SIN( 60 )= .866026
ENTER ANGLE IN DEGREES? 90
SIN( 90 )= 1
ENTER ANGLE IN DEGREES?
```

NOTE:

The value returned by this function will not be correct if (1) X is a single precision value which is greater than or equal to 2.7E7, or (2) X is a double precision value which is greater than or equal to 1.2D17.

SPACE\$

Format

SPACE\$(X)

Purpose

Returns a string of spaces whose length is determined by the value specified for X.

Remarks

The value of X is rounded to the nearest integer, and must be in the range from 0 to 255.

See also

SPC

Example

```
10 FOR I=1 TO 10
20 PRINT "*" ; SPACE$(I*2) ; "*"
30 NEXT
```

RUN

```
* *
*  *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
*           *
*            *
*             *
*              *
*               *
*                *
*                 *
*                  *
*                   *
*                    *
*                     *
*                      *
*                       *
*                        *
*                         *
*                          *
*                           *
*                            *
*                             *
*                              *
*                               *
*                                *
*                                 *
*                                  *
*                                   *
*                                    *
*                                     *
*                                      *
*                                       *
*                                        *
*                                         *
*                                          *
*                                           *
*                                            *
*                                             *
*                                              *
*                                               *
*                                                *
*                                                 *
*                                                  *
*                                                   *
*                                                    *
*                                                     *
*                                                      *
*                                                       *
*                                                        *
*                                                         *
*                                                          *
*                                                           *
*                                                            *
*                                                             *
*                                                              *
*                                                               *
*                                                                *
*                                                                 *
*                                                                  *
*                                                                   *
*                                                                    *
*                                                                     *
*                                                                      *
*                                                                       *
*                                                                        *
*                                                                         *
*                                                                          *
*                                                                           *
*                                                                            *
*                                                                             *
*                                                                              *
*                                                                               *
*                                                                                *
*                                                                                 *
*                                                                                  *
*                                                                                   *
*                                                                                    *
*                                                                                     *
*                                                                                      *
*                                                                                       *
*                                                                                        *
*                                                                                         *
*                                                                                          *
*                                                                                           *
*                                                                                            *
*                                                                                             *
*                                                                                              *
*                                                                                               *
*                                                                                                *
*                                                                                                 *
*                                                                                                  *
*                                                                                                   *
*                                                                                                    *
*                                                                                                     *
*                                                                                                      *
*                                                                                                       *
*                                                                                                        *
*                                                                                                         *
*                                                                                                          *
*                                                                                                           *
*                                                                                                            *
*                                                                                                             *
*                                                                                                              *
*                                                                                                               *
*                                                                                                                *
*                                                                                                                 *
*                                                                                                                  *
*                                                                                                                   *
*                                                                                                                    *
*                                                                                                                     *
*                                                                                                                      *
*                                                                                                                       *
*                                                                                                                        *
*                                                                                                                         *
*                                                                                                                          *
*                                                                                                                           *
*                                                                                                                            *
*                                                                                                                             *
*                                                                                                                              *
*                                                                                                                               *
*                                                                                                                                *
*                                                                                                             *
```

Ok

SPC

Format

SPC(I)

Purpose

Returns a string of I spaces for output to the display or printer.

Remarks

Unlike the SPACE\$ function, which can be used with the LET statement to assign spaces to variables, the SPC function can only be used with output statements such as PRINT and LPRINT. The value specified for I must be in the range from 0 to 255.

Example

```
5 CLS
10 FOR I=0 TO 16 STEP 4
15 LOCATE 20-I/2,CSRLIN
20 PRINT "*" ; SPC(I) ; "*"
30 NEXT
40 FOR I=16 TO 0 STEP -4
45 LOCATE 20-I/2,CSRLIN
50 PRINT "*" ; SPC(I) ; "*"
60 NEXT
```

```
      **
     *  *
    *   *
   *   *
  *   *
 *   *
*   *
*   *
 *   *
  *   *
   *   *
    *   *
     *  *
      **
```

Ok

SQR

Format

SQR(X)

Purpose

Returns the square root of X.

Remarks

The value specified for X must be greater than or equal to 0.

Example

```
10 PRINT "X", "SQR(X)"
20 FOR X=0 TO 100 STEP 10
30 PRINT X, SQR(X)
40 NEXT
```

RUN

X	SQR(X)
0	0
10	3.16228
20	4.47214
30	5.47723
40	6.32456
50	7.07107
60	7.74597
70	8.3666
80	8.94427
90	9.48683
100	10

OK

STR\$

Format	STR\$(X)
Purpose	Returns a string of ASCII characters representing the value of X.
Remarks	X must be a numeric expression.
See also	VAL
Example	

```
10 FOR X=1 TO 10
20 PRINT X;
30 NEXT
40 PRINT
50 FOR X=1 TO 10
60 A$=STR$(X)
70 PRINT A$;
80 NEXT
```

RUN

```
1 2 3 4 5 6 7 8 9 10
```

```
12345678910
```

OK

STRING\$

Format

STRING\$(I,J)
STRING\$(I,X\$)

Purpose

Returns a string of characters whose length is specified by I.

Remarks

The length of the character string returned by this function is determined by the value of I. If J is specified, this function returns a string of I characters whose ASCII code corresponds to the value of J (if a non-integer numeric expression is specified for J, its value is rounded to the nearest integer before the string of characters is returned).

If X\$ is specified, this function returns a string of I characters made up of the first character of the specified string.

Example

```
10 FOR I=1 TO 5
20 A$=STRING$(I,65)
30 PRINT A$
40 NEXT I
50 END
```

```
OK
RUN
A
AA
AAA
AAAA
AAAAA
OK
```

STYLE\$

Format STYLE\$(<X\$ > , ,)

Purpose This function returns a character string X\$ with the font changed as specified by and .

Remarks In the character string returned by this function, those characters in X\$ which are in are converted to and other characters are unaffected. and are specified as numbers from 0 to 16, where 0 specifies 1-byte characters. Numbers from 1 to 16 specify 2-byte characters of the corresponding character font.

Example

```
10 A$="EPSON QX-10"  
20 FOR I=2 TO 15  
30 PRINT STYLE$(A$,I,I)  
40 NEXT
```

```
RUN  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
EPSON QX-10  
OK
```

TAB

Format

TAB(I)

Purpose

Returns the number of spaces required to move the cursor or print head to character position I on the display screen or printer. If the cursor/print head is already past character position I on the current line, it is moved to that position on the next line.

The character position on the far left of the display screen/printer is specified as 0, and that on the right is specified as the device width minus one. I must be a value in the range from 0 to 255. If the value of I is greater than the device width, I MOD n is assumed (where n is the device width).

The TAB function can only be used with the PRINT and LPRINT statements.

Example

```
10 PRINT 1;2;3;4;5;6;7
20 PRINT 1;TAB(4);2;TAB(9);3;TAB(15);4;TAB(22);5;TAB(30)
```

RUN

```
1 2 3 4 5 6 7
1 2 3 4 5
```

Ok

TAN

Format TAN(X)

Purpose Returns the tangent of X, where X is an angle in radians.

Remarks The tangent of angle X is calculated to the precision of the type of numeric expression specified for X.

Example

```
10 INPUT"ENTER ANGLE IN DEGREES";A
20 PRINT "TANGENT";A;"DEGREES IS";TAN(A*3.14159/180)
50 GOTO 10
```

RUN

```
ENTER ANGLE IN DEGREES? 30
TANGENT 30 DEGREES IS .57735
ENTER ANGLE IN DEGREES? 45
TANGENT 45 DEGREES IS .999999
ENTER ANGLE IN DEGREES? 60
TANGENT 60 DEGREES IS 1.73205
ENTER ANGLE IN DEGREES? 90
TANGENT 90 DEGREES IS 762908
ENTER ANGLE IN DEGREES?
```

NOTE:

The value returned by this function will not be correct if (1) X is a single precision value which is greater than or equal to 2.7E7, or (2) X is a double precision value which is greater than or equal to 1.2D17.

TIME/TIME\$

Format

TIME
TIME\$

Purpose

Returns the time of the QX-10's built-in clock.

Remarks

TIME returns the number of seconds which have passed since 00:00:00 as a number in the range from 0 to 86399.

TIME\$ returns the time of the built-in clock in "HH:MM:SS" format. HH is a number from 00 to 23 which indicates the hour, MM is a number from 00 to 59 which indicates the minute, and SS is a number from 00 to 59 which indicates the second.

TIME\$ is a system variable, and can also be set by executing TIME\$="HH:MM:SS".

Example

```
10 CLS
20 LOCATE 36,8,0
30 PRINT TIME$
40 LOCATE 36,9,0
50 PRINT DATE$
60 GOTO 20
```

16:31:19

02/02/83

USR

Format USR[<digit>](argument)

Purpose Passes (argument) to a user-written machine language routine and returns the result of that routine.

Remarks <digit> is an integer from 0 to 9 which corresponds to the digit specified in the DEF USR statement for the machine language routine.

If <digit> is omitted, USR0 is assumed.

A string or numeric expression must be specified for (argument); this argument is passed to the machine language routine as described in Appendix E.

Example See the example given for the DEF USR statement.

VAL

Format

VAL(X\$)

Purpose

Returns the numeric value of a string composed of numeric ASCII characters.

Remarks

The first character of string expression X\$ must be "+", "-", ".", "&", or a numeric character; otherwise, VAL(X\$)=0. See the description of the STR\$ function for conversion of numeric values to strings.

The VAL function makes it possible to convert strings consisting of numeric characters into decimal numbers.

- (1) VAL(X\$) — Returns the number corresponding to the character string representation of a decimal number; complementary to the STR\$ function.
- (2) VAL("&H" + X\$) — Returns the number corresponding to the character string representation of a hexadecimal number; complementary to the HEX\$ function.
- (3) VAL("&O" + X\$) — Returns the number corresponding to the character string representation of an octal number; complementary to the OCT\$ function.

Example

```
10 A$="12345"  
20 B$="67890"  
30 PRINT A$+B$  
40 PRINT VAL(A$)+VAL(B$)  
50 END
```

RUN

1234567890

800235

Ok

VARPTR

Format VARPTR(<variable name>)
 VARPTR(# <file number>)

Purpose The first format returns the address in memory of the first data byte of the variable specified in <variable name>.

The second format returns the starting address of the disk I/O buffer assigned to the file opened under <file number>.

Remarks With the first format, a value must be assigned to <variable name> before executing VARPTR; otherwise, an "Illegal function call" error will result. Any type of variable name (numeric, string, or array) may be specified, and the address returned will be an integer in the range from -32768 to 32767. If a negative number is returned, add it to 65536 to obtain the actual address.

The second format returns the address of the first byte of the I/O buffer assigned to the file opened under <file number>.

This function is generally used to obtain the address of a variable prior to passing it to a machine language program. In the case of array variables, the format VARPTR(A(0)) is generally used so that the address returned is that of the lowest-numbered element of the array.

Example

```
10 INPUT "ENTER SINGLE PRECISION NUMBER";A!  
20 PRINT "VARPTR(A)=";VARPTR(A!)  
30 IF SGN(VARPTR(A!))=-1 THEN I=VARPTR(A!)+65536!  
   ELSE I=VARPTR(A!)  
40 PRINT "ADDRESS A!=";I;"(";"HEX$(I);" IN HEXADECI  
   MAL)"  
50 PRINT "CONTENTS OF A! ARE:";PEEK(I);PEEK(I+1);P  
   EEK(I+2);PEEK(I+3)  
60 PRINT "CONTENTS OF A! WHEN CONVERTED USING CVS  
   FUNCTION:";CVS(CHR$(PEEK(I))+CHR$(PEEK(I+1))+CHR$(  
   PEEK(I+2))+CHR$(PEEK(I+3)))  
70 GOTO 10
```

RUN

```
ENTER SINGLE PRECISION NUMBER? 1  
VARPTR(A)=-25678  
ADDRESS A!= 39858 (9BB2 IN HEXADECEIMAL)  
CONTENTS OF A! ARE: 0 0 0 129  
CONTENTS OF A! WHEN CONVERTED USING CVS FUNCTION: 1  
ENTER SINGLE PRECISION NUMBER? 123  
VARPTR(A)=-25678  
ADDRESS A!= 39858 (9BB2 IN HEXADECEIMAL)  
CONTENTS OF A! ARE: 0 0 118 135  
CONTENTS OF A! WHEN CONVERTED USING CVS FUNCTION: 123  
ENTER SINGLE PRECISION NUMBER? 123.123  
VARPTR(A)=-25678  
ADDRESS A!= 39858 (9BB2 IN HEXADECEIMAL)  
CONTENTS OF A! ARE: 250 62 118 135  
CONTENTS OF A! WHEN CONVERTED USING CVS FUNCTION: 123.123  
ENTER SINGLE PRECISION NUMBER?
```

NOTE:

The addresses of array variables change whenever a value is assigned to a new simple variable; therefore, all simple variable assignments should be made before calling VARPTR for an array.