Chapter 4 MFBASIC FUNCTIONS

This chapter describes the intrinsic functions provided by MFBASIC. These functions may be called at any time, either from a program or in the direct mode; there is no need for definition on the part of the user.

The arguments of functions are always enclosed in parentheses. In the explanations which follow, the following abbreviations are used for the arguments.

X or Y	Represent any numeric expressions
I or J	Represent integer expressions
X$ or Y$ ...	Represent string expressions

If a floating point value is specified when the function requires an integer, MFBASIC automatically rounds the number to the nearest integer value.

ABS

Format ABS(X)

Purpose Returns the absolute value of expression X.

Remarks Any numeric expression may be specified for X.

Example

```
10 PRINT ABS(50)
20 PRINT ABS(-50)
30 END
Ok
RUN
 50
 50
Ok
```

ASC

Format ASC(X$)

Purpose Returns the numeric value which is the ASCII code for the first character of string X$. (See Appendix H for the ASCII codes.)

Remarks X$ must be a string expression. An "Illegal function call" error will occur if X$ is a null string.

See also CHR$

Example

```
10 X=65
20 FOR I=0 TO 10
30 A$(I)=CHR$(X)
40 PRINT A$(I),
50 X=X+1
60 NEXT I
70 PRINT
80 FOR I=0 TO 10
90 PRINT ASC(A$(I)),
100 NEXT I
```

```
Ok
RUN
A         B         C         D         E
F         G         H         I         J
K
 65        66        67        68        69
 70        71        72        73        74
 75
Ok
```

ATN

Format ATN(X)

Purpose Returns the arc tangent in radians for X.

Remarks This function returns an angle in radians for expression X as a value from -π/2 to π/2. The angle will be returned as a double precision number if X is a double precision number, and as a single precision number if X is a single precision number or an integer. To obtain the angle in degrees, multiply the result by 90/1.57080 (for single precision) or 90/1.570796326794897 (for double precision).

Example

```
10 INPUT"ENTER ARBITRARY NUMBER";X
20 Y=ATN(X)*90/1.5708
30 PRINT"ARCTANGENT OF";X;"IS";Y;"DEGREES"
40 GOTO 10
```

```
Ok
RUN
ENTER ARBITRARY NUMBER? .1
ARCTANGENT OF .1 IS 5.71058 DEGREES
ENTER ARBITRARY NUMBER? 1
ARCTANGENT OF 1 IS 44.9999 DEGREES
ENTER ARBITRARY NUMBER? 5
ARCTANGENT OF 5 IS 78.6899 DEGREES
ENTER ARBITRARY NUMBER? 9
ARCTANGENT OF 9 IS 83.6596 DEGREES
ENTER ARBITRARY NUMBER? 100
ARCTANGENT OF 100 IS 89.4269 DEGREES
ENTER ARBITRARY NUMBER?
```

ATTR$

Format

ATTR$ <file descriptor>)

Purpose

Returns the setting of the write protect attribute for the specified disk file.

Remarks

This function returns the setting of the write protect attribute for the disk file specified in <file descriptor>. (The file specified in <file descriptor> must be a disk file.) If the write protect attribute is set for the specified file, the ATTR$ function returns the 1-byte character string "P". If the write protect attribute is not set, the function returns a null character string.

Example

```
10 SET "A:FILE1",P
20 GOSUB 60
30 SET "A:FILE1"
40 GOSUB 60
50 END
60 IF ATTR$("A:FILE1")="P" THEN PRINT "FILE IS PROTECTED"
ELSE PRINT "FILE IS NOT PROTECTED"
70 RETURN

Ok
RUN
FILE IS PROTECTED
FILE IS NOT PROTECTED
Ok
```

CDBL

Format CDBL(X)

Purpose Converts numeric expression X to a double precision number.

Remarks This function converts the values of integer or single precision numeric expressions to double precision numbers. Significant decimal places added to converted numbers are padded with zeros.

Example

```
10 INPUT "X,Y";X,Y
20 PRINT CDBL(X*Y)
30 PRINT CSNG(X*Y)
40 END
```

```
Ok
RUN
X,Y? 123456,1234
 152344704
 1.52345E+08
Ok
```

CHR$

Format CHR$(I)

Purpose Returns the 1-byte character string whose ASCII code equals the value of integer expression I. (See Appendix H for ASCII codes.)

Remarks The CHR$ function is frequently used to send special characters to terminal equipment such as the console or printer. For example, executing PRINT CHR$(12) clears the display screen and returns the cursor to the home position; executing PRINT CHR$(7) causes the tone generator to beep; and executing PRINT CHR$(11) moves the cursor to the home position without clearing the screen. See the description of the ASC function for conversion of ASCII characters to numeric values.

Example

```
5 PRINT "ASCII","CHARACTER"
10 FOR I=65 TO 75
20 PRINT I,CHR$(I)
30 NEXT I

Ok
RUN
ASCII        CHARACTER
 65          A
 66          B
 67          C
 68          D
 69          E
 70          F
 71          G
 72          H
 73          I
 74          J
 75          K
Ok
```

CINT

Format CINT(X)

Purpose Rounds the decimal portion of numeric expression X to the nearest whole number and returns the equivalent integer value.

Remarks X must be a numeric expression which, when rounded, is within the range from -32768 to +32767; otherwise, an "Overflow" error will occur.

See the descriptions of the FIX and INT functions for other methods of converting numbers to integers.

See the descriptions of the CDBL and CSNG functions for conversion of numeric expressions to double and single precision numbers.

Example

```
5 PRINT "X","CINT(X)"
10 FOR I=1 TO 4
20 READ A(I)
30 A=A(I)
40 PRINT A(I),CINT(A)
50 NEXT I
60 DATA 3,5.45,1.5,99.8
70 END

Ok
RUN
X          CINT(X)
 3          3
 5.45       5
 1.5        2
 99.8       100
Ok
```

NOTE:
Differences between the CINT, FIX, and INT functions are as follows.

CINT (X)	Rounds the decimal portion of X to the nearest integer value.
FIX (X)	Discards the decimal portion of X.
INT (X)	Returns the integer value with is less than or equal to X.

- For X = 1.2
  CINT (X) = 1
  FIX (X) = 1
  INT (X) = 1
- For X = 1.2
  CINT (X) = 2
  FIX (X) = 1
  INT (X) = 1

- For X = −1.2
  CINT (X) = −1
  FIX (X) = −1
  INT (X) = −2
- For X = −1.2
  CINT (X) = −2
  FIX (X) = −1
  INT (X) = −2

COS

Format COS(X)

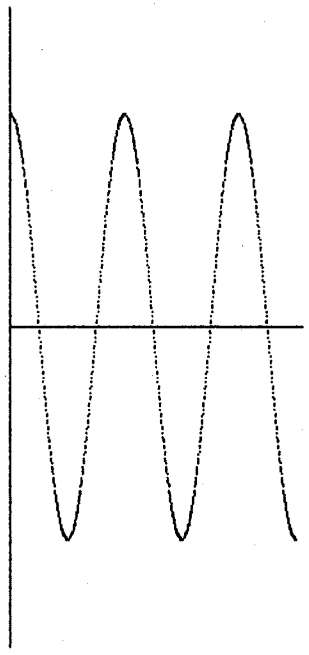
Purpose Returns the cosine of angle X, where X is in radians.

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

Example

```
10 CLS
20 PI=3.14159
30 D=PI/180
40 LINE (50,200)-(280,200)
50 LINE (50,50)-(50,350)
60 FOR I=0 TO 900
70 X=50+I/4
80 Y=200-COS(D*I)*100
90 PSET (X,Y)
100 NEXT I
110 END

Ok
```



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.

CSNG

Format CSNG(X)

Purpose Returns the single precision number obtained by conversion of the value of numeric expression X.

Remarks See the descriptions of the CDBL and CINT functions for conversion of numeric values to double precision or integer type numbers.

Example

```
10 A#=1234567.123456789#
20 PRINT A#
30 PRINT CSNG(A#)
40 END

Ok
RUN
 1234567.123456789
 1.23457E+06
Ok
```

CSRLIN

Format CSRLIN

Purpose Returns the number corresponding to the current vertical position of the cursor on the display screen.

Remarks The lines of the display screen are numbered from top to bottom, starting with 1 at the top and ending with 20 at the bottom. The CSRLIN function returns the value which corresponds to the cursor's current vertical position on the screen. (The POS function is used to obtain the current horizontal position of the cursor.)

Example

```
10 CLS
20 FOR I=1 TO 10
30 LOCATE I,I
40 PRINT "CSRLIN=";CSRLIN
50 NEXT I
```

```
CSRLIN= 1
 CSRLIN= 2
  CSRLIN= 3
   CSRLIN= 4
    CSRLIN= 5
     CSRLIN= 6
      CSRLIN= 7
       CSRLIN= 8
        CSRLIN= 9
         CSRLIN= 10
Ok
```

CVI/CVS/CVD

Format CVI(<2-byte string>)
CVS(<4-byte string>)
CVD(<8-byte string>)

Purpose These functions are used to convert string values into numeric values.

Remarks Numeric values are converted to string values for storage in random access files with the MKI$, MKS$, or MKD$ functions; when they are read back in from the file, they must be converted back into numeric values. This is done using the CVI, CVS, and CVD functions.

CVI returns an integer for a 2-byte string, CVS returns a single precision number for a 4-byte string, and CVD returns a double precision number for an 8-byte string.

See also MKI$/MKS$/MKD$

Example

```
10 OPEN"R",#1,"A:CVISD.DAT"
20 FIELD #1,2 AS I$,4 AS S$,8 AS D$
30 I%=13523:S!=344799!:D#=7080606095.1#
40 RSET I$=MKI$(I%)
50 RSET S$=MKS$(S!)
60 RSET D$=MKD$(D#)
70 PUT #1,1
80 CLOSE #1
85 PRINT "STRING",,,"VALUE"
90 OPEN"R",#1,"A:CVISD.DAT"
100 FIELD #1,2 AS I$,4 AS S$,8 AS D$
110 GET#1
115 PRINT ASC(I$);ASC(MID$(I$,2,1)),,,
120 PRINT CVI(I$)
125 PRINT ASC(S$);ASC(MID$(S$,2,1));ASC(MID$(S$,3,
2));ASC(MID$(S$,4,1)),,
130 PRINT CVS(S$)
135 PRINT ASC(D$);ASC(MID$(D$,2,1));ASC(MID$(D$,3,
2));ASC(MID$(D$,4,1));ASC(MID$(D$,5,1));ASC(MID$(D
$,6,1));ASC(MID$(D$,7,1));ASC(MID$(D$,8,1)),
140 PRINT CVD(D$)
150 CLOSE#1
160 END
200 FIELD #1,2 AS I$,4 AS S$,8 AS D$

Ok
RUN
STRING                                  VALUE
 211  52                                 13523
 224  91  40  147                        344799
 204  204  140  199  188  4  83  161     7080606095.1
Ok
```

DATE/DATE$

Format

DATE
DATE$

Purpose

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

Remarks

DATE returns the number of days which have passed since the beginning of the current calendar year. Values returned range from 1 (for January 1) to 365 or 366 (for December 31), depending on whether the current year is a leap year.

DATE$ returns the date of the built-in clock in "MM/DD/YY" format. MM is a number from 01 to 12 which indicates the month, DD is a number from 01 to 31 which indicates the day, and YY is a number from 00 to 99 which indicates the year.

DATE$ is a system variable, and can also be set by executing DATE$="MM/DD/YY".

Example

```
10 PRINT "TODAY'S DATE IS: ";DATE$
20 PRINT "TODAY IS DAY";DATE;"OF THE YEAR"
```

```
Ok
RUN
TODAY'S DATE IS: 02/02/83
TODAY IS DAY 33 OF THE YEAR
Ok
```

DAY

Format DAY

Purpose Returns the day of the week from the QX-10's built-in clock.

Remarks This function returns the day of the week from the built-in clock as a number from 0 to 6. Normally, "0" is used for Sunday and "6" is used for Saturday. However, DAY is a system variable which can be set independently of the clock, so any value from 0 to 6 can be assigned to the current day of the week by executing DAY= <value>.

Example

```
10 FOR I=0 TO 6
20 DAY=I
30 PRINT DAY,MID$("SUNMONTUEWEDTHUFRISAT",DAY*3+1,
3)
40 NEXT I

Ok
RUN
 0            SUN
 1            MON
 2            TUE
 3            WED
 4            THU
 5            FRI
 6            SAT
Ok
```

DSKF

Format DSKF (<drive name>)

Purpose Returns the amount of free area on the disk in the specified drive.

Remarks This function returns the amount of unused area on the disk in the drive specified by <drive name>. The value returned is an integer which indicates the amount of free area in 1K-byte units. <drive name> must be specified as a character expression whose value corresponds to "A:", "B:", or "E:".

Example

```
10 PRINT"DRIVE","FREE AREA"
20 FOR I=1 TO 3
30 READ A$
40 PRINT A$,DSKF(A$);"KILOBYTES"
50 NEXT I
60 END
70 DATA "A:","B:","E:"

Ok
RUN
DRIVE         FREE AREA
A:             16 KILOBYTES
B:             212 KILOBYTES
E:             25 KILOBYTES
Ok
```

EOF

Format EOF (<file number>)

Purpose Returns a value indicating whether the end of a sequential file has been reached during sequential input.

Remarks During input from a sequential file, an "Input past end" error will occur if INPUT # statements are executed against that file after the end of the file has been reached. This can be prevented by testing whether the end of file has been reached with the EOF function.

<file number> is the number under which the file was opened. The function will return "false" (0) if the end of file has not been reached, and "true" (-1) if the end of file has been reached.

Example

```
10 OPEN"O",#1,"A:FILE1"
20 FOR I=1 TO 25
30 PRINT #1,STR$(I)
40 NEXT
50 CLOSE
60 OPEN"I",#1,"A:FILE1"
70 INPUT #1,A$
80 PRINT A$;" ";
90 IF EOF(1) THEN GOTO 110
100 GOTO 70
110 CLOSE

Ok
RUN
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
Ok
```

ERL

Format ERL

Purpose Returns the line number of the program line at which an error occurred during command or statement execution.

Remarks The ERL function returns the line number of the command/statement causing an error during program execution. If an error occurs during execution of a command or statement in the direct mode, this function returns the number 65535 as the line number. As with the ERR function, ERL is normally used in the IF...THEN statements of an error processing routine to control the flow of program execution.

See also ON ERROR GOTO, RESUME, ERR

Example

```
10 ON ERROR GOTO 80
20 CLS
30 INPUT "INPUT A NUMBER FROM 1 TO 9";A
40 IF (A<1) OR (A>9) THEN ERROR 200
50 PRINT A
60 ERROR 210
70 END
80 IF ERR=200 THEN PRINT "TRY AGAIN":RESUME 30
90 IF ERR=210 THEN PRINT "ERROR CODE";ERR;"ON LINE";ERL
100 END
```

```
INPUT A NUMBER FROM 1 TO 9? 0
TRY AGAIN
INPUT A NUMBER FROM 1 TO 9? 10
TRY AGAIN
INPUT A NUMBER FROM 1 TO 9? 5
 5
ERROR CODE 210 ON LINE 60
Ok
```

NOTE:
If the ERL function is used in a relational expression, it must be included to the left of an equal sign ("="); otherwise, the number on the right side of the expression will not be updated when the RENUM command is executed.

ERR

Format ERR

Purpose Returns the error code of errors occurring during command or statement execution.

Remarks The ERR function returns the error code of errors occurring during command or statement execution, and is normally used in the IF...THEN statements of an error processing routine to control the flow of program execution.

For example, including the following statement in an error processing routine causes program execution to resume at the program line specified in <line number> if the error code returned by ERR is that specified in <error code>.

```
IF ERR=<error code> THEN RESUME <line number>
```

See also ON ERROR GOTO, RESUME, ERL

Example See the example given in the explanation of the ERL function.

EXP

Format EXP(X)

Purpose Returns the value of the natural base *e* to the power of X.

Remarks The value specified for X must not be greater than 87.3365; otherwise, an "Overflow" error will occur.

Example

```
10 FOR I=0 TO 80 STEP 10
20 PRINT "e^";I;"=";EXP(I)
30 NEXT
```

```
Ok
RUN
e^ 0 = 1
e^ 10 = 22026.5
e^ 20 = 4.85165E+08
e^ 30 = 1.06865E+13
e^ 40 = 2.35385E+17
e^ 50 = 5.1847E+21
e^ 60 = 1.142E+26
e^ 70 = 2.51544E+30
e^ 80 = 5.54063E+34
Ok
```

FIX

Format FIX(X)

Purpose Returns the integer portion of numeric expression X.

Remarks The value returned by FIX(X) is equal to the sign of X times the integer portion of the absolute value of X. Thus, -1 is returned for -1.5, -2 is returned for -2.33333, and so forth. (Compare with the INT function, which returns the largest integer which is less than or equal to X.)

See also CINT, INT

Example

```
10 PRINT "FOR FOR/NEXT LOOP",,"FOR DATA STATEMENT"
20 PRINT "I","FIX(I)",,"A","FIX(A)"
30 FOR I=-5 TO 5 STEP .8
40 READ A
50 PRINT I,FIX(I),,A,FIX(A)
60 NEXT I
70 DATA -5,-4.2,-3.4,-2.6,-1.8,-1,-.2,.6,1.4,2.2,3,3.8,
4.6
```

```
Ok
RUN
FOR FOR/NEXT LOOP             FOR DATA STATEMENT
I          FIX(I)             A          FIX(A)
-5         -5                 -5         -5
-4.2       -4                 -4.2       -4
-3.4       -3                 -3.4       -3
-2.6       -2                 -2.6       -2
-1.8       -1                 -1.8       -1
-1          0                 -1         -1
-.2         0                 -.2         0
 .6         0                  .6         0
 1.4        1                  1.4        1
 2.2        2                  2.2        2
 3          3                  3          3
 3.8        3                  3.8        3
 4.6        4                  4.6        4
Ok
```

FONT

Format FONT (<X$>)

Purpose Returns the font number of the first character in string expression X$.

Remarks This function returns the font number of the first character in the string specified by X$ as a number from 0 to 16. "0" is returned for 1-byte characters, and the font number (from 1 to 16) corresponding to the applicable character font is returned for MultiFonts characters. (See Appendix J.)

Example

```
10 A$="EPSON"
20 B$="QX-10"
30 C$="MFBASIC"
40 PRINT A$,"FONT IS";FONT(A$)
50 PRINT B$,"FONT IS";FONT(B$)
60 PRINT C$,"FONT IS";FONT(C$)

Ok
RUN
EPSON     FONT IS 1
QX-10     FONT IS 13
MFBASIC   FONT IS 0
Ok
```

FRE

Format

FRE(0)
FRE(X$)

Purpose

Returns the number of unused bytes of memory in the BASIC text area or the BASIC string area.

Remarks

If a string expression is specified in the function's argument, the value returned is the number of unused bytes of memory in the BASIC string area; if a numeric expression is specified, the value returned is the number of unused bytes of memory in the BASIC text area. If a variable name is specified, the contents of that variable are not affected.

Note that the value returned includes a work area which is used by MFBASIC during program execution, and thus does not provide a direct indication of the number of bytes available for string or numeric variables.

Example

```
10 PRINT FRE(X$)
20 A$="EPSON QX-10 MFBASIC"
30 PRINT FRE(X$)

Ok
RUN
 56180
 56173
Ok
```

HEX$

Format HEX$(X)

Purpose Returns a character string which represents the hexadecimal value of X.

Remarks The value of the numeric expression specified in the argument must be a number in the range from -32768 to 65535. If the value of the expression includes a decimal fraction, it is rounded to the nearest integer before the string representing the hexadecimal value is returned.

See also OCT$

Example

```
10 FOR I=10 TO 100 STEP 11
20 PRINT I;"DECIMAL IS ";HEX$(I);" HEXADECIMAL"
30 NEXT I
```

```
Ok
RUN
 10 DECIMAL IS A HEXADECIMAL
 21 DECIMAL IS 15 HEXADECIMAL
 32 DECIMAL IS 20 HEXADECIMAL
 43 DECIMAL IS 2B HEXADECIMAL
 54 DECIMAL IS 36 HEXADECIMAL
 65 DECIMAL IS 41 HEXADECIMAL
 76 DECIMAL IS 4C HEXADECIMAL
 87 DECIMAL IS 57 HEXADECIMAL
 98 DECIMAL IS 62 HEXADECIMAL
Ok
```

INKEY$

Format INKEY$

Purpose This function checks the keyboard buffer during program execution and returns one character if any of the keys have been pressed, or a null string if no key has been pressed.

Remarks INKEY$ returns a null string if the keyboard buffer is empty. If any key whose code is included in the ASCII code table has been pressed, INKEY$ reads that character from the keyboard buffer and returns it as a one-character string. Characters read from the keyboard buffer by INKEY$ are not displayed on the screen.

Example

```
10 CLS
20 DIM TXT$(5)
30 A$=INKEY$:IF LEN(A$)=0 THEN 30
40 PRINT A$;:B$=B$+A$
50 IF A$=" " THEN GOSUB 120
60 IF I>4 THEN GOTO 80
70 GOTO 30
80 PRINT:FOR I=0 TO 5
90 PRINT TXT$(I)
100 NEXT
110 END
120 IF LEN(C$+B$)<20 THEN C$=C$+B$:B$="":RETURN:ELSE
TXT$(I)=C$:C$=B$:B$=""
130 I=I+1
140 RETURN
```

```
Now is the time for all good men to come to the aid of their party. The quick br
own fox
Now is the time
for all good men
to come to the aid
of their party.
The quick brown

Ok
```

INP

Format INP(I)

Purpose Returns one byte of data from machine port I.

Remarks The machine port number must be specified in I as an integer expression in the range from 0 to 255.

Example

```
Ok
PRINT INP(11)
 46
Ok
```

INPUT$

Format INPUT$(X[,[#]Y])

Purpose Reads a string of X characters from the keyboard buffer or the file opened under file number Y.

Remarks INPUT$(X) reads the number of characters specified by X from the keyboard buffer and returns a string consisting of those characters to the program; if the keyboard buffer does not contain the specified number of characters, INKEY$ reads those characters which are present and waits for other keys to be pressed. Characters read are not displayed on the screen. Unlike the INPUT and LINE INPUT statements, INPUT$ allows characters such as RETURN (character code 13) to be passed to the program.

INPUT (X,[#]Y) reads the number of characters specified by X from a sequential file opened under file number Y. As with the first format, characters which would be recognized as delimiters between items by the INPUT # or LINE INPUT # statements are returned as part of the character string. Execution of the INPUT$ function can be terminated by pressing the BREAK key.

Example 1

```
10 'LIST THE CONTENTS OF SEQUENTIAL FILE IN HEXADECIMAL
20 OPEN"I",#1,"A:FILE1"
30 IF EOF(1) THEN 60
40 PRINT HEX$(ASC(INPUT$(1,#1)));" ";
50 GOTO 30
60 PRINT
70 END
```

```
RUN
20 31 D A 20 32 D A 20 33 D A 20 34 D A 20 35 D A 20 36 D A 20 37 D A 20 38 D A
20 39 D A 20 31 30 D A 20 31 31 D A 20 31 32 D A 20 31 33 D A 20 31 34 D A 20 31
 35 D A 20 31 36 D A 20 31 37 D A 20 31 38 D A 20 31 39 D A 20 32 30 D A 20 32
31 D A 20 32 32 D A 20 32 33 D A 20 32 34 D A 20 32 35 D A
Ok
```

Example 2

```
10 A$=INPUT$(5)
20 PRINT A$
30 GOTO 10
```

```
RUN
ABCDE
FGHIJ
KLMNO
PQRST
```

INSTR

Format INSTR([I,]X$,Y$)

Purpose This function searches for the first occurrence of string Y$ in string X$ and returns the position at which a match is found.

Remarks If I is specified, the search for string Y$ begins at position I in string X$. I must be specified as an integer expression in the range from 1 to 255. If a null string is specified for Y$, INSTR returns the value which is equal to that specified for I. "0" is returned if I>LEN(X$), if X$ is a null string, or if Y$ cannot be found. Both X$ and Y$ may be specified as string variables, string expressions, or string literals.

Example

```
10 X$="ABC.EIBIDKJKLNJDSKLNN"
20 Y$="KL"
30 PRINT INSTR(X$,Y$),
40 PRINT INSTR(13,X$,Y$)

Ok
RUN
 12            18
Ok
```

NOTE:
An "Illegal function call" error will occur if 0 or a number greater than 255 is specified for I.

INT

Format INT(X)

Purpose Returns the largest integer which is less than or equal to X.

Remarks Any numeric expression may be specified for X.

See also CINT, FIX

Example

```
10 PRINT "I","INT(I)"
20 FOR I=-5 TO 5 STEP .8
30 PRINT I,INT(I)
40 NEXT I

Ok
RUN
I             INT(I)
-5            -5
-4.2          -5
-3.4          -4
-2.6          -3
-1.8          -2
-1            -1
-.2           -1
 .6            0
 1.4           1
 2.2           2
 3             3
 3.8           3
 4.6           4
Ok
```

LEFT$

Format LEFT$(X$,I)

Purpose Returns a string composed of the I characters making up the left 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 MID$, RIGHT$

Example

```
10 A$="MFBASIC"
20 FOR I=1 TO 7
30 PRINT LEFT$(A$,I)
40 NEXT
```

```
Ok
RUN
M
MF
MFB
MFBA
MFBAS
MFBASI
MFBASIC
Ok
```