SECTION IV
Variable Types

In addition to the floating-point real numbers used so far in this manual, BASIC/3000 allows three additional representations of data: integer, long real, and complex. Including real, these four number types apply to variables, arrays, constants, expressions, assignments, functions, input and output.

Type Statements

The type statements allocate space for variables and arrays and assign them a specific data type. Any particular variable or array can appear only in one type statement or DIM statement.

Form

The form of the type statement is

type typespec list

where *type* is either INTEGER, REAL, LONG, or COMPLEX. The *typespec list* includes variables and arrays to be assigned the data type of *type*. Arrays are defined in the same form as in the DIM statement (Section III).

Explanation

A simple variable or array which does not appear in a type statement is automatically type REAL. The explicit typing of variables in a REAL statement is, therefore, redundant, except within a function body, where all local variables must be declared in order to distinguish them from variables of the same name outside the function. Real numbers are represented as 32-bit quantities consisting of a sign, exponent, and fraction. The range of real numbers is $\pm (10^{-77}, 10^{77})$ with approximately 6 to 7 digits of precision.

Variables which appear in an INTEGER statement hold integers. The range of integers is -32767 to 32767.

Variables which appear in a LONG statement hold long numbers. Long representation is a 64-bit quantity with sign, exponent, and fraction. The range is identical to real, but long has a precision of 16 to 17 digits.

Variables which appear in a COMPLEX statement hold numbers in complex form. Complex representation is a 64-bit quantity consisting of two real numbers, one for the real part of the complex number and one for the imaginary part.

NUMERIC CONSTANT FORMS

When constants are used in an expression, DATA statement, or during execution of an INPUT or ENTER statement, they are represented in one of five forms: integer, fixed-point, floating-point, complex, or long. Fixed and floating-point numbers are *type* REAL.

Integer Form

An integer is a series of digits without a decimal point. A number in integer form is represented externally (e.g., on the list device) as *type* INTEGER, but internally as *type* REAL. Examples of integer form:

```
 10 INTEGER A,B,C,D
 20 A=10,B=150,C=5903,D=5
 30 PRINT A,B,C,D
>RUN
 10             150            5903           5
```

When arithmetic operations are performed on expressions containing an integer constant, the results are real numbers. However, when both operands are *type* INTEGER, the result is truncated to the nearest integer. For instance,

```
 10 INTEGER I,J
 20 LET I=3,J=5
 30 PRINT 3/5,3/J,I/5,I/J
>RUN
 .6             .6             .6             0
```

Fixed-Point Form

A fixed-point number is a series of digits with a decimal point. A number in fixed-point form is represented internally as *type* REAL. For example:

```
 10 REAL A,B,C,D
 20 A=73,B=5.5,C=.000567,D=153.97
 30 PRINT A,B,C,D
>RUN
 73             5.5            .000567        153.97
```

Floating-Point Form

A floating-point number is a fixed or integer form number followed by the letter E and an optionally signed exponent. The exponent represents the power of 10 by which the number is multiplied. For example 3E-11 equals 3×10^{-11}. Numbers in floating-point form are represented internally as *type* REAL. Examples of floating-point numbers:

```
 10 REAL A,B,C,D
 20 A=3E-11,B=.4723E-4,C=1.1E4,D=1.1E10
 30 PRINT A,B,C,D
>RUN
 3.00000E-11    4.72300E-05    11000          1.10000E+10
```

A fixed or floating point real number that has an integer value between -999999 and 999999 is printed as an integer.

Complex Form

A complex number consists of two numbers in integer, fixed-point, or floating-point form, separated by a comma and enclosed in parentheses. The first number is the real part, the second represents the imaginary part. Complex numbers are represented internally as *type* COMPLEX. Examples of complex numbers:

```
 10 COMPLEX A,B,C,D
 20 A=(3,5),B=(3.2E-9,0),C=(0,-47),D=(0,0)
 30 PRINT A,B,C,D
>RUN
( 3.00000E+00, 5.00000E+00)    ( 3.20000E-09, 0.00000E+00)
( 0.00000E+00,-4.70000E+01)    ( 0.00000E+00, 0.00000E+00)
```

Long Form

Numbers in long form are identical to numbers in real form, except that the letter E is replaced by the letter L. Long numbers have almost double the precision of real numbers. Long numbers are represented internally as *type* LONG. Examples of long numbers:

```
  10 LONG A,B,C,D
  20 A=3L-11,B=4.751259L-6,C=-1.1L5,D=1.1L-15
  30 PRINT A,B,C,D
>RUN
 3.00000000000000000L-11          4.75125900000000000L-06
-1.10000000000000000L+05          1.10000000000000000L-15
```

Examples of Type Statements

This example assigns values to and prints two integer variables and an integer array:

```
  10 INTEGER A,B1,N[5,5]
  20 LET A=5,B1=10
  30 MAT N=ZER
  40 PRINT A,B1
  50 MAT PRINT N;
>RUN
 5                10
 0      0      0      0      0

 0      0      0      0      0

 0      0      0      0      0

 0      0      0      0      0

 0      0      0      0      0
```

Note that the *type* statement is used instead of a DIM statement to define the dimensions of array N.

This example assigns values to and prints two real variables; one is printed as floating-point and the other as fixed-point:

```
 10 REAL I,J
 20 LET I=2795348.6,J=2.79E-3
 30 PRINT I,J
>RUN
 2.79535E+06     .00279
```

I is printed as a floating-point number because its magnitude is greater than 999999.5; J is printed as fixed-point because its magnitude is less than 999999.5 (see Numeric Output Formats in the PRINT statement description, Section II). Note that the printed value of I is rounded.

The following example inputs values to the *type* LONG variable P, then doubles each value and prints it:

```
  10 LONG P
  20 INPUT P
  30 LET P=P+P
  40 PRINT P
>RUN
?2.7L+10
 5.4000000000000000L+10

>RUN
?2.5L+12
 5.0000000000000000L+12

>RUN
?2.0L+11
 4.0000000000000000L+11
```

The example below reads data into two complex variables and one complex array, and then prints the variable and array values:

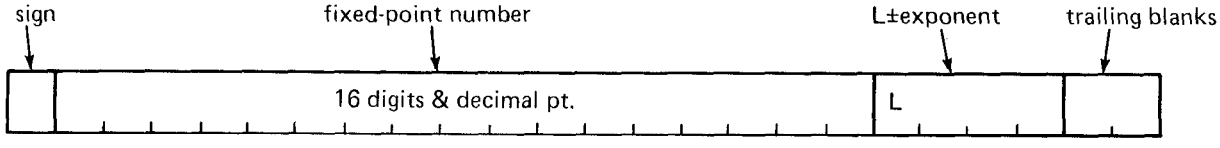
```
  10 COMPLEX C9,Q8,M[15]
  20 READ C9,Q8
  30 MAT READ M[5]
  40 PRINT C9,LIN(1),Q8,LIN(1)
  50 MAT PRINT M;
  90 DATA (4.5E-6,1.2E-9),4.23E6
 100 DATA (3,9),(4.5,-30),(4.5E-6,1.2E-9),(25.3,30.2),(0,0)
>RUN
( 4.50000E-06, 1.20000E-09)
( 4.23000E+06, 0.00000E+00)

( 3.00000E+00, 9.00000E+00)    ( 4.50000E+00,-3.00000E+01)
( 4.50000E-06, 1.20000E-09)    ( 2.53000E+01, 3.02000E+01)
( 0.00000E+00, 0.00000E+00)
```

PRINTING LONG AND COMPLEX DATA

Numbers of all data types can be output with the PRINT statement. All numeric quantities, regardless of type, are printed left-justified in a field whose width is always a multiple of 3. At least one blank is always printed on the right side of the field, unless it is the last item on the line.

The output form for values of type INTEGER and REAL is described under Numeric Output Formats in the PRINT statement description, Section II.

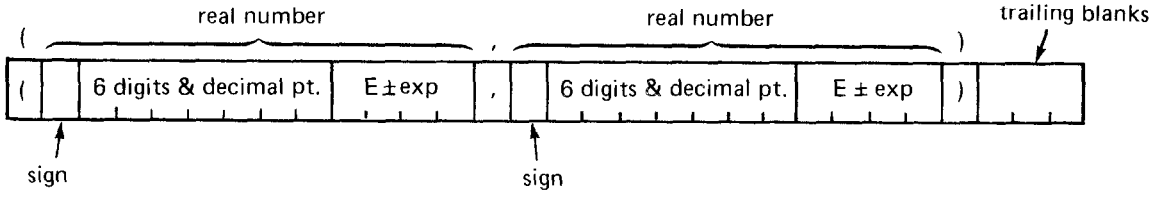
The output form for long quantities is an 16 digit fixed-point number followed by an exponent and two trailing blanks. The total required is 24 print positions.



For example:

```
  10 LONG A,B,C,D,E
  20 A=7.3215L9,B=4.32L-8
  30 C=4.3214978L-8,D=2.173L2
  40 E=2.173L6
  50 PRINT A;B;C;LIN(1),D;E
>RUN
 7.321500000000000L+09    4.320000000000000L-08    4.321497800000000L-08
 2.173000000000000L+02    2.173000000000000L+06
```

The output form for complex numbers is two real numbers separated by a comma and enclosed in parentheses (i.e., this is the same form as a complex constant). Each part of the number is printed as a separate 6-digit fixed-point number, followed by an exponent. The total required is 30 print positions including 3 trailing blanks:



For example:

```
  10 COMPLEX A,B
  20 LET A=(1.2E8,1.39E-6)
  30 LET B=(12.5,1.56E6)
  40 PRINT A;B
>RUN
( 1.20000E+08,  1.39000E-06)    ( 1.25000E+01,  1.56000E+06)
```

NUMERIC EXPRESSIONS

Variables of all data types and numbers of all data forms can be used in numeric expressions. BASIC/3000 provides the arithmetic operations for all four data types as well as automatic conversion when two operands are not of the same type. The following table summarizes the results of combining arithmetic elements with any operator (except AND, OR, NOT, and relationals):

First Element Data Type	Second Element Data Type: INTEGER	REAL	LONG	COMPLEX
INTEGER	INTEGER	REAL	LONG	COMPLEX
REAL	REAL	REAL	LONG	COMPLEX
LONG	LONG	LONG	LONG	COMPLEX
COMPLEX	COMPLEX	COMPLEX	COMPLEX	COMPLEX

When the operators AND, OR, NOT, =, <, >, <=, >=, and <> are used, the result is always type REAL (0 for false, 1 for true). When relations are performed on complex numbers, the real parts are compared first; the imaginary parts are compared only if the real parts are equal.

Examples

An integer combined with a real type in an expression results in a real number; two integers result in an integer:

```
  10 INTEGER I,I1
  20 REAL R
  30 LET I=25,I1=50,R=2.75
  40 PRINT I+I1
  50 PRINT I+R
>RUN
 75
 27.75
```

A real type combined with a long results in a long type number; a long type combined with a complex results in a complex type number:

```
  10 REAL R
  20 LONG L
  30 LET L=-5.25L2,R=2.75
  40 PRINT L+R
  50 COMPLEX C
  60 C=(2.75,-1.25)
  70 PRINT L+C
>RUN
-5.22500000000000L+02
(-5.22500E+02,-1.25000E+00)
```

CONDITIONAL STATEMENT

The numeric expression used to make a branching decision in a conditional statement (Section II) can contain, or result in, any numeric data type. The expression is considered false if equal to 0, true otherwise.

NUMERIC ASSIGNMENT

When the result of a numeric expression is assigned to a variable, it is converted to the type of that variable. In a LET statement, the same result can be assigned to several variables in turn, from right to left (A=B=C=5+D7). These variables need not be of the same type. If they are not, a conversion is performed at each step in the assignment.

The method of conversion used in assigning values to variables of differing data types is summarized in this table:

Variable Type	Value Type	Conversion Method
INTEGER	REAL	Round.
INTEGER	LONG	Round.
INTEGER	COMPLEX	Round real part; drop imaginary part.
REAL	INTEGER	Float.
REAL	LONG	Truncate to real precision.
REAL	COMPLEX	Drop imaginary part.
LONG	INTEGER	Float to long precision.
LONG	REAL	Extend mantissa with zeroes.
LONG	COMPLEX	Extend mantissa of real part with zeroes; drop imaginary part.
COMPLEX	INTEGER	Float for real part; imaginary part equals zero.
COMPLEX	REAL	Imaginary part equals zero.
COMPLEX	LONG	Truncate to real precision for real part; imaginary part equals zero.

Note that this table applies wherever values are assigned to variables (INPUT, READ, etc.).

An example of multiple assignment with type conversion is:

```
 10 INTEGER I
 20 REAL R
 30 COMPLEX C
 40 LONG L
 50 LET R=C=I=L=1.5L0
 60 PRINT R,C,I,L
>RUN
 2                ( 2.00000E+00,  0.00000E+00)      2
 1.5000000000000000L+00
```

Note that the long number is rounded up to 2 when it is converted to the integer variable I. R and C also equal 2 since they are assigned after I. If line 50 is changed so that C is assigned before I, the rounding does not affect C:

```
 10 INTEGER I
 20 REAL R
 30 COMPLEX C
 40 LONG L
 50 LET R=I=C=L=1.5L0
 60 PRINT R,C,I,L
>RUN
 2                ( 1.50000E+00,  0.00000E+00)      2
 1.5000000000000000L+00
```

If the constant 1.4 is assigned instead of 1.5, the number is rounded to 1 when it is converted to an integer and assigned to I. As a result of this integer conversion, R is also set equal to 1:

```
 10 INTEGER I
 20 REAL R
 30 COMPLEX C
 40 LONG L
 50 LET R=I=C=L=1.4L0
 60 PRINT R,C,I,L
>RUN
 1                ( 1.40000E+00,  0.00000E+00)      1
 1.4000000000000000L+00
```

ENTERING NUMERIC DATA

Constants of all data forms can be entered using READ, INPUT, and ENTER statements. Once entered they are converted to the type of the receiving variable according to the table under "Numeric Assignment."

OTHER USES OF DATA TYPES

Numbers of all data types can be output with controlled format with the PRINT USING statement (see Section IX). Numbers of all data types can also be written onto and read from mass storage data files. This process is described fully in Section VIII.

NUMERIC ARRAYS

Arrays can be of all data types. Each element of the array is a variable of the specified type. The type statement effectively provides the dimensions of an array. All of the MAT statements dealing with arrays (see Section III) apply equally to integer, real, long, and complex arrays, except that integer arrays cannot be inverted with the MAT Inverse statement. Arrays of different types cannot be mixed in a MAT statement.

Examples

```
  10 INTEGER I[3,5]
  20 LONG L[2,2]
  30 REAL R[2,2]
  40 COMPLEX C[2,2]
  50 MAT I=ZER
  60 MAT R=CON
  70 MAT C=IDN
  80 MAT L=IDN
  90 MAT L=(25**2)*L
 100 MAT PRINT I;LIN(1),R;LIN(1),C;LIN(1),L;
>RUN
 0      0      0      0      0

 0      0      0      0      0

 0      0      0      0      0


 1      1

 1      1


( 1.00000E+00,  0.00000E+00)    ( 0.00000E+00,  0.00000E+00)

( 0.00000E+00,  0.00000E+00)    ( 1.00000E+00,  0.00000E+00)


 6.2500000000000000L+02    0.0000000000000000L+00

 0.0000000000000000L+00    6.2500000000000000L+02
```

FUNCTION CLASS

Numeric built-in functions are divided into four classes according to the nature of their result.

The following table defines the four classes of function. The type of result they return is shown for the different argument types:

		Type of Argument		
		INTEGER/REAL	LONG	COMPLEX
Class	1	REAL	LONG	REAL
	2	REAL	LONG	COMPLEX
	3	COMPLEX	COMPLEX	COMPLEX
	4	REAL	REAL	REAL

The numeric functions are listed below according to their class. A complete list of these functions with their meaning is contained in Appendix E.

Class 1 Functions

ABS(x)	Absolute value of x.
ATN(x)	Arctangent x.
INT(x)	Largest integer less than or equal to x.
CEI(x)	Smallest integer greater than or equal to x.

Class 2 Functions

EXP(x)	e^x
LOG(x)	$\log_e x$
SQR(x)	Square root of x.
SIN(x)	Sine x.
COS(x)	Cosine x.

TAN(x)	Tangent x.
SNH(x)	Hyperbolic sine x.
CSH(x)	Hyperbolic cosine x.
TNH(x)	Hyperbolic tangent x.
PIX(x)	π * x

Class 3 Functions

CNJ(x)	Complex conjugate of x.
CPX(x,y)	Complex number x+y*i*

Class 4 Functions

RND(x)	Random number.
REA(x)	Real part of x.
IMG(x)	Imaginary part of x.