SECTION IX
Formatted Output

The PRINT USING and IMAGE statements of BASIC/3000 give the user explicit and exact control over the format of his program output. The PRINT # USING statement allows the output to be directed to a specified ASCII file. All types of numbers can be printed: integer, fixed-point, floating-point, and complex. The exact position of plus and minus signs can be specified. String values can be printed in specified fields; literal strings and blanks can be inserted wherever needed. Carriage return and line feed are under explicit control and lines longer than 72 characters can be printed.

Format strings are used to specify the output format. These strings are explicitly included in the PRINT USING statement, or they may be specified in IMAGE statements whose labels are referenced in the PRINT USING statements.

PRINT USING Statement

The PRINT USING statement allows the user to output a list of items according to a customized format.

Form

The forms of PRINT USING are:

PRINT USING label;print using list

PRINT USING label

PRINT USING string expression;print using list

PRINT USING string expression

The *print using list* is an optional list of expressions and functions from which items are printed. It is like a *print list* (see PRINT Statement, Section II) except that semicolons and trailing punctuation are not allowed.

Either a *label* or *string expression* must be specified following USING. If a *label* is used, it identifies an IMAGE statement containing the format string. If a *string expression* is used instead, the expression itself is the format string. In the case when the format string is of the form "literal specifications", outer quote marks must be represented by the ASCII character code for the quote symbol ('34).

The *print using list* can be omitted when the format consists entirely of literal specifications.

Explanation

A format string describes the form in which items in the *print using list* are to be printed. The full description of format strings is contained under Format Strings, this section.

Any commas in the print using list are separators only; they have no formatting function as they do in PRINT.

When PRINT USING is executed, each specification in the associated format string is extracted and examined. If the specification calls for a string or numeric value, the print using list is examined for a corresponding expression. Each expression is output according to its corresponding specification in the format string. Any print functions in the list are executed as they are encountered.

If the expression and the specification do not match because one is a string and the other is numeric, the program terminates with an error message. If the value of a numeric expression is greater than can be printed with the format, the number is printed without format control, and preceded by two asterisks, on a separate line. The program continues. An integer specification prints a real number with any fraction rounded to the nearest integer.

If the format specification is a literal specification, it is printed without examining the print using list. When the format string contains only literal specifications, the print using list, if present, is ignored.

If the end of the format string is reached before the end of the print using list, processing returns to the beginning of the string. The next expression in the list is matched to the first specification in the format string. When all expressions in the list have been printed, a carriage return and linefeed are generated unless a carriage control character specified in the format string suppresses the final carriage return and/or linefeed.

Examples

In the following example, three ways of specifying the format with PRINT USING statements are shown. In line 30, numeric variables A and B are printed according to a format specified in a string expression. In line 40, the PRINT USING statement references the IMAGE statement of line 50, so that A and B are printed according to the format string given there. In line 60, the print using list is omitted and outer quote marks are given by the ASCII character code '34:

```
  10 DIM A$[10]
  20 LET A=100000,B=999999
  30 PRINT USING "10D";A,B
  40 PRINT USING 50;A,TAB(23),B
  50 IMAGE $DXDDDXDD
  60 PRINT USING '34"ABCDEF"'34
>RUN
    100000    999999
$1 000 00                $9 999 99
ABCDEF
```

The exact meaning of the format strings used in these examples is described below under Format Strings. Note here that:

integers are replicators causing repetition of the specification that follows

X causes a blank to be printed

D prints a decimal digit from the print using list

$ causes a dollar sign to be printed

PRINT # USING Statement

The PRINT # USING statement allows the user to output a list of items to an ASCII file according to a customized format.

Form

The forms of PRINT # USING are:

PRINT #*filenumber*[;] USING *label*;*print list*
PRINT #*filenumber*[;] USING *string expression*;*print list*
PRINT #*filenumber*[;] USING *label*
PRINT #*filenumber*[;] USING *string expression*
PRINT #*filenumber*,*recordnumber*[;] USING *label*;*print list*
PRINT #*filenumber*,*recordnumber*[;] USING *string expression; print list*
PRINT #*filenumber*,*recordnumber*[;] USING *label*
PRINT #*filenumber*,*recordnumber*[;] USING *string expression*

The *filenumber* identifies an ASCII file, which has been specified in a FILES or ASSIGN statement (see Section VIII). If the *filenumber* is zero, the standard listing device (your terminal in an interactive session, the line printer in a batch job) is referenced.

The optional *recordnumber* specifies the record into which the items in the *print list* are to be printed. A *recordnumber* may only be specified for an ASCII file with direct access capability, such as a disc file. It is not allowed for a file which has only serial access capability (such as the line printer) and will generate the error DIRECT ACCESS ILLEGAL IN LINE *line number* if attempted in such a case. For this reason, a *recordnumber* is not allowed if the *filenumber* is zero.

The optional *print list* is one of the following: *print using list* (as described for PRINT USING), END, or *print using list*,END. It may be omitted when the format consists entirely of literal specifications.

The format string is in an IMAGE statement identified by the *label* or it is contained in the *string expression.* The form for the *label* or *string expression* is the same as described for PRINT USING.

The semicolon before USING is optional. BASIC will accept it, but it will not appear when the line is listed.

Explanation

PRINT # USING to ASCII files basically acts in the same manner as PRINT USING except that output is directed to the ASCII file specified by the filenumber. If the filenumber is zero, this statement is exactly equivalent to the PRINT USING statement.

The format specifications for printing to an ASCII file describe the form in which items in the print list are to be printed. They are provided and evaluated in the same manner as for PRINT USING.

When a print list is specified, END may be used after or instead of the print using list to cause an end-of-file to be written to the specified file. END is not allowed if the filenumber is zero.

Both the filenumber and recordnumber may be numeric constants, variables, or expressions.

Note that the ASCII code equivalents for carriage return and line feed are not written to ASCII files, but cause adjustment of the record pointer. This also applies to the action of the LIN and CTL functions. See the example below.

Example

```
10 FILES AFILE
20 LET A=123,B=45678
30 PRINT #1 USING "7D";A,B
40 PRINT #1 USING '34"SOME MORE EXAMPLES"'34
50 PRINT #1,4 USING 60;A,B
60 IMAGE $DXDDXDD3X
70 PRINT #1,6 USING '34"ABC"'10"DEF"'34
80 PRINT #1 USING "5D";A,LIN(-3),B
```

AFILE now contains the following:

```
    123  45678
SOME MORE EXAMPLES

$   1 23   $4 56 78

ABC
   DEF
  123


     45678
```

The recordnumbers given in lines 50 and 70 caused output to be directed to those specific records of the ASCII file AFILE. The ASCII equivalent of line feed ('10) in line 70 and the LIN function in the print list of line 80 caused adjustment of the record pointer.

MAT PRINT USING Statement

The MAT PRINT USING statement allows the printing of one or more complete arrays according to a specified format.

Form

The form of MAT PRINT USING is:

MAT PRINT USING label

MAT PRINT USING label;mat print using list

MAT PRINT USING string expression

MAT PRINT USING string expression;mat print using list

The *mat print using list* is a list of arrays and print functions.

The format string is in an IMAGE statement identified by *label*, or it is contained in the *string expression.*

Explanation

The format specifications for printing the arrays in the list are provided and evaluated in the same manner as for PRINT USING. The arrays are printed row by row as in the MAT PRINT statement, but all spacing is provided by the format string associated with the mat print using list.

Examples

```
  10 DIM A[3,7],B[3,5]
  20 MAT READ A,B
  30 MAT PRINT USING 100;B
  40 MAT PRINT USING "7(3D2X)/";A
  50 DATA 100,110,120,130,140,150,160,170,180,190,200
  60 DATA 210,220,230,240,250,260,270,280,290,300
  70 DATA 400,410,420,430,440,450,460,470,480,490
  80 DATA 500,510,520,530,540
 100 IMAGE 5(DXDDXX)/
>RUN
4 00   4 10   4 20   4 30   4 40
4 50   4 60   4 70   4 80   4 90
5 00   5 10   5 20   5 30   5 40

100   110   120   130   140   150   160
170   180   190   200   210   220   230
240   250   260   270   280   290   300
```

Array A is printed according to the string expression in line 40; array B is printed according to the format in the IMAGE statement in line 100, which inserts a blank after the first digit of each number.

MAT PRINT # USING

The MAT PRINT # USING statement allows the printing of one or more complete arrays to an ASCII file according to a customized format.

Form

The forms of MAT PRINT # USING are:

MAT PRINT *#filenumber*[;] USING *label;mat print list*
MAT PRINT *#filenumber*[;] USING *string expression;mat print list*
MAT PRINT *#filenumber*[;] USING *label*
MAT PRINT *#filenumber*[;] USING *string expression*
MAT PRINT *#filenumber,recordnumber*[;] USING *label;mat print list*
MAT PRINT *#filenumber,recordnumber*[;] USING *string expression;mat print list*
MAT PRINT *#filenumber,recordnumber*[;] USING *label*
MAT PRINT *#filenumber,recordnumber*[;] USING *string expression*

The *filenumber* identifies an ASCII file. If the *filenumber* is zero, the standard listing device (your terminal or the line printer) is referenced.

The optional *recordnumber* specifies the record into which the items in the *mat print list* are to be printed. As for the PRINT # USING statement, a *recordnumber* may only be specified for an ASCII file having direct access capability.

The optional *mat print list* is one of the following: *mat print using list* (as described for MAT PRINT USING), END, or *mat print using list*, END. It may be omitted when the format consists entirely of literal specifications.

The format string is in an IMAGE statement identified by *label*, or it is contained in the *string expression.*

As for PRINT # USING, the semicolon before USING is optional.

Explanation

MAT PRINT # USING to ASCII files acts in the same manner as MAT PRINT USING except that output is directed to the ASCII file specified by the filenumber When the filenumber is zero, this statement is equivalent to the MAT PRINT USING statement.

The format specifications for printing the mat print list are provided and evaluated in the same manner as for MAT PRINT USING. END may be used after or instead of a mat print using list to cause an end-of-file to be written to the specified ASCII file and is not allowed if the filenumber is zero.

As in PRINT # USING, note that the ASCII code equivalents for carriage return and line feed cause adjustment of the record pointer and that this affects the action of the LIN and CTL functions.

Examples

```
   10 FILES AFILE
   20 DIM A[10],B[3,5]
   30 MAT READ A,B
   40 MAT PRINT #1 USING "10(DDX)";A
   50 MAT PRINT #1 USING 100;LIN(1),B
   60 MAT PRINT #1,8 USING "5(DDDXXX)/";B,END
   70 DATA 10,11,12,13,14,15,16,17,18,19
   80 DATA 111,222,333,444,555,666,777,888,999
   90 DATA 101,202,303,404,505,606
   100 IMAGE 2($DXDDX),3(3D)/
>RUN
```

AFILE now contains the following:

```
10 11 12 13 14 15 16 17 18 19

$1 11 $2 22 333444555
$6 66 $7 77 888999101
$2 02 $3 03 404505606


111   222   333   444   555
666   777   888   999   101
202   303   404   505   606
```

IMAGE Statement

The IMAGE statement specifies a format string to be used in a PRINT USING statement. It is a declarative statement and is not executed.

Form

The form of the IMAGE statement is

IMAGE format string

The *format string* is not quoted. It may be used by PRINT USING, MAT PRINT USING, PRINT # USING or MAT PRINT # USING statements

Format string specification is described in this section under Format Strings.

Explanation

The format string specified in an IMAGE statement is associated through the label of the IMAGE statement with one or more formatted print statements. By specifying a format in an IMAGE statement, many PRINT USING and MAT PRINT USING statements can use the same format.

The legitimacy of the format string in the IMAGE statement is not checked until execution of the PRINT USING statement referencing it.

Examples

```
  10 DIM A$[20]
  20 A$="1,2,3,4,5,6,7,8,9,0,"
  30 A=123456,B=-789100,C=12
  40 X=1234.56
  50 IMAGE 6D2X,"LITERAL",S2DXX,$4D.2D
  60 IMAGE S4DXX.DD/20AX,S7D
  70 PRINT USING 50;A,C,X
  80 PRINT USING 60;X,A$,B
>RUN
123456  LITERAL+12  $1234.56
+1234  .56
1,2,3,4,5,6,7,8,9,0,  -789100
```

The decimal point in the format string indicates that a decimal point should be printed and, by its position, specifies the number of digits to the left and right of the decimal. The slash causes a carriage return and linefeed.

Format Strings

The format string consists of an optional carriage control character and comma followed by one or more format specifications separated by one or more commas and/or slashes. Each format specification consists of orderly combinations of format symbols.

Format Symbols

Format specifications are composed of permutations of the following symbols (the replicator *n* is an integer between 1 and 255 inclusive):

A	Causes the printing of a single character. *n*A causes *n* characters to be printed. A is legal only for string data items.
D	Causes a decimal digit to be printed. *n*D causes *n* decimal digits to be printed.
X	Causes a blank to be printed. *n*X causes *n* blanks to be printed.
.	Indicates placement of decimal point.
S	Indicates placement of "+" and "-" signs.
M	Indicates placement of "-" only.
E	Causes numbers to be printed in E-type float form.
C	Indicates a complex format follows.
K	Indicates compressed numeric formatting.
I	Causes the character "I" to be printed; *n*I causes *n* I's to be printed.
$	Causes the character "$" to be printed; *n*$ causes *n* dollar signs to be printed.
+ or -	Separate the real from the imaginary part in a complex number and indicate placement of the sign for the imaginary part.
/	Separates specifications and generates a carriage return and linefeed.
( )	Enclose a group field specification so that a replicator can be used to signify the multiple occurrence of this group of specifications. Also enclose complex specification preceded by C.
,	Separates specifications
" "	Enclose literal specifications.
'	Apostrophe is used for special string characters (e.g., '40)

The format symbols are combined according to certain rules to form the following types of specifications:

literal specification

string specification

numeric specification

The symbols X, I, $, and literal strings are considered editing symbols and may be interspersed freely in any specification.

Literal Specification

This specification contains only the editing symbols X, I, $, and literal strings. When a literal string is encountered in a format string, the characters between quotes are printed. For instance,

"THIS IS A LITERAL STRING"

When the characters I and $ are encountered in a format string, the literal characters "I" and "$" respectively are printed.

Each X in a format string causes a single blank character to be printed.

Replicators may precede an X, I, or $ in a format string:

2I	equivalent to II
3X	equivalent to XXX
2$5X	equivalent to $$XXXXX

String Specification

In a string specification, any combination of editing symbols and A's (with or without replicators) is permitted. At least one A must be present to signify a string specification. The matching item in the print using list must be a string value; it is printed left-justified with blank fill on the right. If necessary, the string value is truncated on the right to fit in the field. For example,

3A	(prints 3 characters adjacent)
AXAXA	(prints 3 characters separated by blanks)

Examples

```
  10 DIM A$[10],B$[3]
  20 LET A$="ABCDEFGHIJ",B$="ABC"
  30 PRINT USING "3X3A3X,AXAXAX";B$,"UVWX"
  40 PRINT USING 50;A$
  50 IMAGE "ALPHA STRING",3X10A
>RUN
   ABC   U V W
ALPHA STRING   ABCDEFGHIJ
```

Numeric Specification

The specification for a numeric field other than complex consists of the symbols S, M, D, ".", and E. S and M may be used to position the sign, D's are used to indicate the positions of the digits, the "." is used to position the decimal point, and E may be used to specify that an exponent field is to be printed. The editing symbols may be freely inserted into a numeric field. The specification for a complex field usually consists of two numeric specifications combined in a special way that is described in detail later.

Generally, numbers are rounded when being converted for formatted output.

In all numeric specifications (except the imaginary part of a complex specification) printing of signs is handled by the characters S and M, according to the following rules:

1. If an S appears *before* all D's and any decimal point, the sign (whether + or -) is printed immediately preceding either the most significant digit or the decimal point, whichever is further to the left.

2. If an S appears after any D's or to the right of the decimal point, the sign is always printed in that particular position (i.e., a fixed sign).

3. If an M appears anywhere in a specification, either a blank is printed in that position (if the number is positive or zero) or a "-" is printed in that position (if the number is negative).

4. If no S or M appears anywhere in the specification, the number is printed as specified (if the number is zero or positive). If the number is negative, the specification is treated exactly as if the first D were replaced by an S, provided at least one D precedes the decimal point.

5. Only one S or M is allowed per specification.

If the field specified by any numeric specification is not large enough to hold the most significant digits of a number to be printed, the number is printed alone on the next line preceded by two asterisks. It is printed in a standard format according to type as if generated by a PRINT statement.

INTEGER SPECIFICATION. Any combination of editing symbols and D's (with or without replicators) is allowed in an integer specification, but at least one D must be present. One S or M can appear. The number is printed right-justified, one digit per D with leading zeros suppressed and the fractional part (if any) rounded to the nearest integer.

Examples

```
 10 PRINT USING 20;-1,1,5400,-19.301,-74,103.65
 20 IMAGE XDXDXD,DM2DX,S4D,3X5D,DDDDM,X$4D
>RUN
  - 1   1 +5400     -19  74- $ 104
```

Note that the fractional part of decimal numbers are dropped after rounding.

FIXED SPECIFICATION. Any combination of editing symbols and D'S (with or without replicators) is allowed in a fixed specification, but at least one D and only one period must be present. One S or M can appear. The number is printed anchored around the position of the decimal point (which is printed) with leading zeros suppressed. If necessary, the fractional part of the number is rounded or filled with zeros before being printed to the right of the decimal point. One digit or blank is printed per D. Trailing zeros are printed.

Examples

```
 10 PRINT USING 20;7032,-4.29374,21,-.001,470.32,1.9
 20 IMAGE 3XDS3D.5D,MDD.D,3D.X10D,DDD.DD,XMDDD.4D,DD
>RUN
   7+032.00000- 4.3 21. 0000000000  -.00  470.3200 2
```

FLOATING SPECIFICATION. A floating specification is an integer or fixed specification followed by E. Editing symbols may be interspersed freely. The most significant digits are printed from left to right, and an exponent is printed as E±dd. If a negative number is to be printed in floating point format, there must be an S or M in the specification.

Examples

```
  10 PRINT USING 20;74.92,-400000,(32.1,4),1E12,1.5L21
  20 IMAGE DDEX,M.DDEXX,4DEX,2(5D.EX)
>RUN
75E+00 -.40E+06  3210E-02 10000.E+08 15000.E+17
```

Note that the imaginary part of the complex number is not printed.

COMPLEX SPECIFICATION. The real part of a complex number can be printed using an integer, fixed, or floating specification. The imaginary specification is preceded by a + or - and consists of an integer, fixed, or floating specification *in which there is no S or M.* The only difference is in the treatment of the sign. The + is treated exactly like an S; the - is treated as an X if the number is not negative and as an S if the number is negative.

To print both the real and imaginary parts of a complex number, the real specification precedes the imaginary specification in the same format with no intervening comma.

An I is not printed automatically to indicate imaginary, but if desired, can be included in the format string as a literal or by using the I symbol.

Examples

```
  10 PRINT USING 30;(3,2),(0,1),(.003,-4),(.003,4),21
  30 IMAGE 3D+DDX,3D.D-.DE,2(X.3D-D.D,I),XXXDD
>RUN
  3 +2     .0 .1E+01 .003-4.0I .003 4.0I   21

  10 PRINT USING "XXXDD,+DD";(21,0),(3,-10)
>RUN
   21-10
```

The I in the format specification is printed as "I"; it may be used to indicate that the imaginary part of the number is being printed. The second example prints the real part of the first number, and the imaginary part of the second.

The C specification is a second method of formatting complex numbers for output; this results in formatted output in a form similar to the standard PRINT format (i.e., parentheses enclosing two numbers separated by a comma). The C specification is the letter C followed by two noncomplex numeric fields (integer, fixed, or floating) which are separated by a comma and enclosed in parentheses. The first field defines the format of the real part of the number and the second defines the format of the imaginary part. Both fields must be present. If a noncomplex number is printed using this field, the imaginary part is assumed to be zero.

Examples

```
  10 PRINT USING 20;(3,2),(0,1),(.003,-4),(3,-4),21
  20 IMAGE C(3D,SDD),C(3D.D,M.DE),C(DD,MD.D),C(D.E,MDI),C(DD,SXXDD)
>RUN
(  3, +2)(    .0,  .1E+01)( 0,-4.0)(3.E+00,-4I)(21,    +0)
```

In these examples, the same expressions printed in the first example are printed with the C specification. With this specification, the imaginary part may have an M or an S.

COMPRESSED FORMAT SPECIFICATION. The compressed specification consists of one K and any editing symbols. It prints a number using only as many characters as are required. The resulting format is identical to that used in the simple PRINT statement (Section II), except that no blanks appear and trailing zeroes and decimal points are deleted. This specification is useful when numeric data is to appear within text.

Examples

```
  10 PRINT USING 20;25
  20 IMAGE "I HAVE",XKX,"BANANAS"
>RUN
I HAVE 25 BANANAS
```

Separators

BASIC/3000 format strings have two separators: the comma (,) and the slash (/). A separator is required between two adjoining specifications. The slash, in addition to separating specifications, also generates a carriage return and linefeed. Two separators can be adjacent. The meaning of adjacent commas is equivalent to a single comma.

Examples

```
  10 PRINT USING 20;25,10,-5,(1,250),2575.5,2.5
  20 IMAGE DDX.E/4D,/,SDDX,D+3D.E//"HIWAY"/4DX,,SD.E
  30 PRINT USING "5A";"ABCDE"
>RUN
25  .E+00
  10
 -5  1+250.E+00

HIWAY
2575  +3.E+00
ABCDE
```

Each slash in the format string causes a carriage return and linefeed.

Grouping

BASIC/3000 format strings allow one or more field specifications to be enclosed by parentheses and preceded by a replicator to indicate the repetition of the entire enclosed set of specifications. This newly grouped field must be separated from other specifications of the format string by a separator. Grouped specifications can be nested indefinitely.

Examples

```
   5 LET A$="A",B$="B",C$="C"
  10 PRINT USING 20;3000,300,A$,400,B$,500,C$,1.5,2.4,3.3,6.6
  20 IMAGE 4D4X,3(3D.D/5X,3A)/2(2(D.DX)/)
>RUN
3000     300.0
      A  400.0
      B  500.0
      C
1.5  2.4
3.3  6.6
```

The IMAGE statement in the example above is exactly equivalent to the following IMAGE statement:

```
  5 LET A$="A",B$="B",C$="C"
 10 PRINT USING 20;3000,300,A$,400,B$,500,C$,1.5,2.4,3.3,6.6
 20 IMAGE 4D4X,3D.D/5X,3A,3D.D/5X,3A,3D.D/5X,3A/D.DX,D.DX/D.DX,D.DX/
>RUN
3000     300.0
     A   400.0
     B   500.0
     C
1.5 2.4
3.3 6.6
```

Carriage Control Characters

The carriage control characters are

+	suppress linefeed
-	suppress carriage return
#	suppress linefeed and carriage return

These characters specify action to be taken *following* execution of the PRINT USING statement. They specify whether a carriage return and linefeed are generated after the last item is printed. If no carriage control characters are present, a carriage return and linefeed will be provided automatically following each line. This is the only automatic line control in PRINT USING; within a format specification, the user is responsible for supplying carriage returns and linefeeds with slashes.

Carriage control characters are executed after all other formatted output, but they must be specified *first* in the format string.

Example

```
 10 PRINT USING "#,S3D.2D,3XS6D";125,625
 20 PRINT USING "X3D.DEX,10A";125,"E STRING"
>RUN
+125.00      +625 125.0E+00 E STRING
```

The carriage control character "#" in line 10 suppresses the automatic carriage return and linefeed following that PRINT USING statement, and causes the next PRINT USING statement to print on the same line.