
FORMAT= FREE or BINARY or RB4 or RB8
or DATABANK or EXCEL or LABELS or LOTUS or '(*format text string*)'

Function:

FORMAT is an option used with the READ and WRITE commands. It supplies the format for reading and writing data within a TSP program. This section describes how to construct a FORMAT string. See READ and WRITE for a description of where to use it.

Usage:

FORMAT has several alternatives:

1. FORMAT=FREE specifies that the numbers are to be read in free format, that is, they are delimited by one or more blanks but may be of varying lengths and mixed formats.

2. FORMAT=BINARY specifies that the data are to be read in binary (machine) format, where each variable occupies a single precision floating point word. Binary data may not be mixed with data in other formats, nor can it be moved from one computer type to another. FORMAT=RB4 (Real Binary 4-byte) is the same thing.

3. FORMAT=RB8 is double precision binary (8-byte).

4. FORMAT=DATABANK specifies that the file being used is a TSP databank. This is an alternative to the IN or OUT/KEEP statements.

5. FORMAT=EXCEL reads an Excel spreadsheet file. If the filename ends with .XLS, this is the default.

6. FORMAT=LABELS is for WRITE only -- it means that labels like those of the standard PRINT command are to be used.

7. FORMAT=LOTUS reads Lotus 123 worksheet files (.WK1, .WKS), with column names at the top and optional dates in the first column. This is the default if the filename includes .WK .

8. FORMAT=RB4 is the same as FORMAT=BINARY (single precision binary).

9. FORMAT=RB8 is used for double precision binary.

10. FORMAT='*(format text string)*' specifies a format with which the data will be read. The format text string in TSP is very similar to the format statement in Fortran, since the Fortran format processor is used on it. However, you do not need to know Fortran to construct a simple format string; and consequently, a description of the features you will need is given here.

Note for Fortran Programmers: Since all data in TSP are floating point, do not use integer or alphameric formats (unless you're using OPTIONS CHARID). Also, avoid parenthetical groupings in the format unless you are sure you know what they do, because the results can be unpredictable with different operating systems.

A format string starts and ends with a ' or ", with the parentheses immediately inside these quotes. TSP checks that these parentheses exist and inserts them if they are missing. Note that it is possible to use quotes within the format string -- just use double quotes outside the parentheses and single quotes within them. For example: WRITE(FORMAT="('

SE(beta)=',G12.5)") SEB; . Alternatives for character strings are WRITE (FORMAT=LABELS), the TITLE command, and the H (Hollerith) format type (if you like to count characters).

Format types within the parentheses describe how long numbers are and how many decimal places they have in the data record. The format types useful to you in a TSP program are usually X, F, E, and G. X specifies columns to be skipped on reading; F the format of floating point numbers, and E the format of floating point numbers in exponential (scientific) notation. G is used for output and specifies the most suitable format (F or E) to be used.

Here is a very simple example using the format (F5.2) to read 5 columns of data:

```
Col: 12345

     10000
```

The number read will be 100.00 since the format specified a 5 digit field with 2 digits after the decimal point.

Here is an example of data for the format (F10.5,5X,E10.3,F8.0):

```
     0        1         2         3         4
Col: 1234567890123456789012345678901234567890123456789

        343.5       .98765E-01     200
```

The first format specifies a field of length 10 with 5 digits to the right of the decimal point, but since a decimal point was explicitly included in the data, the number will be read as 343.5 from columns 1 to 10. Then 5X specifies that 5 columns (11 to 15) are to be skipped.

E10.3 reads a number in exponential format, which is used when a number is too big or too small for normal notation. Once again, the 10 specifies a field length of 10 columns (16 to 25) and the 3 that there are 3 digits to the left of the decimal point. Since the decimal point is specifically included, the number is read as .98765 times 10^{-1} or .098765.

The final format is F8.0, which specifies a field length of 8 columns (26 to 33) and no digits to the right of the implied decimal point. In this case the number to be input has no decimal point and will be read as 200.

Formats can be combined in many ways: for example, an integer number prefacing a format specification tells how many such fields should be read. Here are several examples:

(8F10.5)

specifies an 80 column record with eight numbers of ten columns each.

(10X,5E12.6,10X,8F3.1)

specifies a 104 column record with 13 variables, 5 in E-format and 8 in F- format. Columns 1 to 10 and 71 to 80 will be skipped when reading.

(20F5.2/20F5.2)

specifies two records per observation (the / means to skip to a new record), each with 20 variables.

This last example demonstrates an important feature of formatted data loading in TSP. In general, one observation will be read or written with each pass through the format statement. That is, the format statement should allow for exactly as many numbers as there are variables to be loaded. Usually one record will be read for each observation which contains all the variables for that observation, but it is possible to have more than one record per observation by use of the slash (/) format descriptor. The records do not necessarily have to have identical formats, although they will usually be of the same length.