WRITE WORK FILE

WRITE WORK [**FILE**] *work-file-number* [**VARIABLE**] *operand1* **...**

This chapter covers the following topics:

- Function
- Syntax Description
- External Representation of Fields
- Handling of Large and Dynamic Variables
- Example

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: DEFINE WORK FILE | READ WORK FILE | CLOSE WORK FILE | DOWNLOAD PC FILE

Belongs to Function Group: *Control of Work Files / PC Files*

Function

The `WRITE WORK FILE` statement is used to write records to a physical sequential work file.

This statement can only be used within a program to be executed under Com-plete, CICS, CMS, TSO or TIAM, or in batch mode. Appropriate JCL or system commands must be executed to allocate the work file. For further information, see the *Operations* documentation. For information on work file assignments, see profile parameter `WORK` in the *Parameter Reference*.

It is possible to create a work file in one program or processing loop and to read the same file in a subsequent independent processing loop or in a subsequent program using the `READ WORK FILE` statement.

Note:
For Unicode and code page support, see *Work Files and Print Files on Mainframe Platforms* in the *Unicode and Code Page Support* documentation.

Syntax Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
operand1	C	S	A	G		A	U	N	P	I	F	B	D	T	L	C	G		yes	no

Note:
Neither Format C nor Format G is valid for Natural Connection.

Syntax Element Description:

work-file-number	**Work File Number:** The work file number (as defined to Natural) to be used.
VARIABLE	**Variable Entry:** It is possible to write records with different fields to the same work file with different `WRITE WORK FILE` statements. In this case, the `VARIABLE` entry must be specified in all `WRITE WORK FILE` statements. The records on the external file will be written in variable format. Natural will write all output files as variable-blocked (unless you specify a record format and block size in the execution JCL). When the operand list includes a dynamic variable (that could change in size for different executions of the `WRITE WORK FILE` statement), the `VARIABLE` entry must be specified in all `WRITE WORK FILE` statements. **Variable Index Range:** When writing an array to a work file, you can specify a variable index range for the array. For example: `WRITE WORK FILE work-file-number VARIABLE #ARRAY (I:J)`
operand1	**Fields:** With *operand1* you specify the fields to be written to the work file. These fields may be database fields, user-defined variables, and/or fields read from another work file using the `READ WORK FILE` statement. A database array may be referenced with one single range of indices which indicates the occurrences that are to be written to the work file. Groups from database files may be referenced using the group name. All fields belonging to that group will be written to the work file individually.

External Representation of Fields

Fields written with a `WRITE WORK FILE` statement are represented in the external file according to their internal definition. No editing is performed on the field values.

For fields of format A and B, the number of bytes in the external file is the same as the internal length definition as defined in the Natural program. No editing is performed and a decimal point is not represented in the value.

For fields of format N, the number of bytes on the external file is the sum of internal positions before and after the decimal point. The decimal point is not represented on the external file.

For fields of format P, the number of bytes on the external file is the sum of positions before and after the decimal point, plus 1 for the sign, divided by 2, rounded upward to a full byte.

Note:
No format conversion is performed for fields that are written to a work file.

Examples of Field Representation:

Field Definition	Output Record
#FIELD1 (A10)	10 bytes
#FIELD2 (B15)	15 bytes
#FIELD3 (N1.3)	4 bytes
#FIELD4 (N0.7)	7 bytes
#FIELD5 (P1.2)	2 bytes
#FIELD6 (P6.0)	4 bytes

Note:
When the Natural system functions AVER, NAVER, SUM or TOTAL for numeric fields (format N or P) are written to a work file, the internal length of these fields is increased by one digit (for example, SUM of a field of format P3 is increased to P4). This has to be taken into consideration when reading the work file.

Handling of Large and Dynamic Variables

Work File Type	Handling
UNFORMATTED	Work file type UNFORMATTED can be used to write variables whose size exceeds the maximum record length. See also *Work File Access With Large and Dynamic Variables*.
FORMATTED	A dynamic variable is written in its currently defined length (including length 0).

Example

```
** Example 'WWFEX1': WRITE WORK FILE
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
END-DEFINE
*
FIND EMPLOY-VIEW WITH CITY = 'LONDON'
  WRITE WORK FILE 1
        PERSONNEL-ID NAME
END-FIND
*
END
```