

# WRITE WORK FILE

<code>WRITE WORK [FILE] work-file-number [VARIABLE] operand1 ...</code>
---

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`

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 in batch mode.

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 Windows, UNIX and OpenVMS Platforms* in the *Unicode and Code Page Support* documentation.

## Syntax Description

Operand Definition Table:

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

### Note:

When using the work file types `ENTIRECONNECTION` or `TRANSFER`, *operand1* may neither be of format C, nor G.

Syntax Element Description:

Syntax Element	Description
<i>work-file-number</i>	<p><b>Work File Number:</b></p> <p>The work file number (as defined to Natural) to be used.</p>
VARIABLE	<p><b>Variable Entry:</b></p> <p>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.</p> <p>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.</p> <p><b>Variable Index Range:</b></p> <p>When writing an array to a work file, you can specify a variable index range for the array. For example:</p> <pre>WRITE WORK FILE <i>work-file-number</i> VARIABLE #ARRAY (I:J)</pre>
<i>operand1</i>	<p><b>Fields to Be Written:</b></p> <p>With <i>operand1</i> you specify the fields to be written to the work file. These fields may be database fields, user-defined variables, system variables and/or fields read from another work file using the READ WORK FILE statement.</p> <p>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.</p>

## 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:

<b>Field Definition</b>	<b>Output Record</b>
#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
ASCII ASCII-COMPRESSED SAG (binary)	The work file types ASCII, ASCII-COMPRESSED and SAG (binary) cannot handle dynamic variables and will produce an error. They can, however, handle large variables with a maximum field/record length of 32766 bytes.
ENTIRECONNECTION	The work file type ENTIRECONNECTION cannot handle dynamic variables. It can, however, handle large variables with a maximum field/record length of 1073741824 bytes.
PORTABLE UNFORMATTED	<p>Large and dynamic variables can be written into work files or read from work files using the two work file types PORTABLE and UNFORMATTED. For these types, there is no size restriction for dynamic variables. However, large variables may not exceed a maximum field/record length of 32766 bytes.</p> <p>For the work file type PORTABLE, the field information is stored within the work file. The dynamic variables are resized during READ if the field size in the record is different from the current size.</p> <p>In the WRITE WORK FILE statement, fields are written to the file specified with their byte length. All data types (DYNAMIC or not) are treated the same. No structural information is inserted. Note that Natural uses a buffering mechanism, so you can expect the data to be completely written only after a CLOSE WORK. This is especially important if the file is to be processed with another utility while Natural is running.</p> <p>With the READ WORK FILE statement, fields of fixed length are read with their whole length. If the end-of-file is reached, the remainder of the current field is filled with blanks. The following fields are unchanged. In the case of DYNAMIC data types, all the remainder of the file is read unless it exceeds 1073741824 bytes. If the end of file is reached, the remaining fields (variables) are kept unchanged (normal Natural behavior).</p>
CSV	The maximum field/record length is 32766 bytes for dynamic and large variables. Dynamic variables are supported. X-arrays are not allowed and will result in an error message.

## 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

```