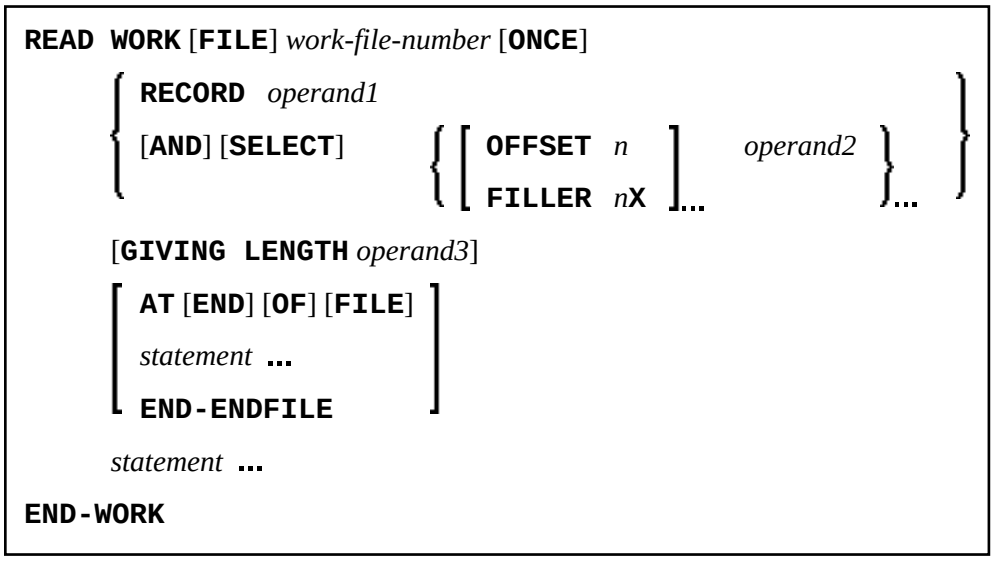
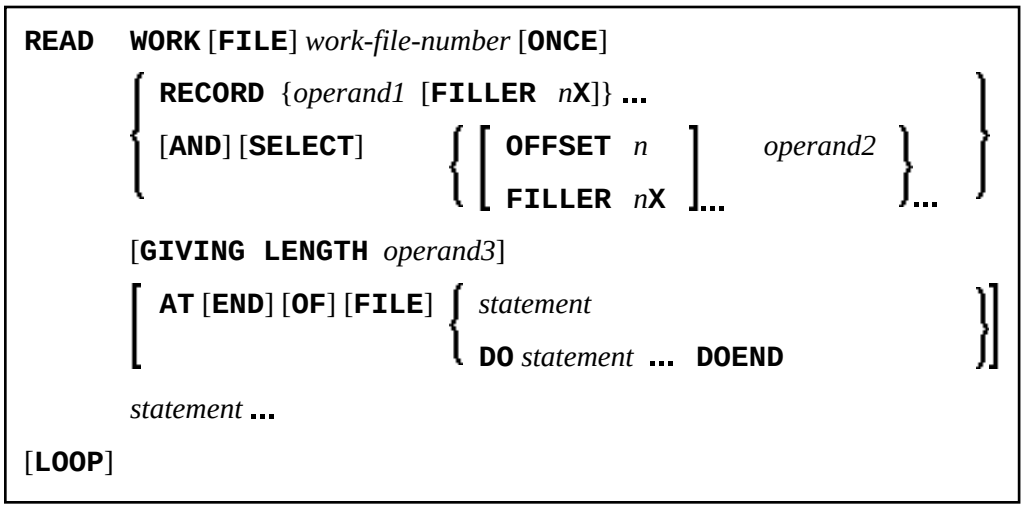
# READ WORK FILE

### Structured Mode Syntax




### Reporting Mode Syntax




This chapter covers the following topics:

- Function
- Syntax Description
- Field Lengths
- Handling of Large and Dynamic Variables
- Example

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

Related Statements: CLOSE WORK FILE | DEFINE WORK FILE | WRITE WORK FILE

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

---

# Function

The `READ WORK FILE` statement is used to read data from a non-Adabas physical sequential work file. The data is read sequentially from the work file. How it is read is independent of how it was written to the 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*.

`READ WORK FILE` initiates and executes a processing loop for reading of all records on the work file. Automatic break processing may be performed within a `READ WORK FILE` loop.

**Notes:**

1. When an end-of-file condition occurs during the execution of a `READ WORK FILE` statement, Natural automatically closes the work file.
2. For Entire Connection: If an Entire Connection work file is read, no I/O statement may be placed within the `READ WORK FILE` processing loop.
3. 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* | | S | A | G | | A | U | N | P | I | F | B | D | T | L | C | G | | yes | yes |
| *operand2* | | S | A | G | | A | U | N | P | I | F | B | D | T | L | C | | | yes | yes |
| *operand3* | | S | | | | | | | | I | | | | | | | | | yes | yes |

Format C is not valid for Natural Connection.

See also *Field Lengths*.

Syntax Element Description:

| | |
|---|---|
| ***work-file-number*** | **Work File Number:**<br><br>The number of the work file (as defined to Natural) to be read.<br><br>**Variable Index Range:**<br><br>When reading an array from a work file, you can specify a variable index range for the array. For example:<br><br>`READ WORK FILE work-file-number #ARRAY (I:J)` |
| **ONCE** | **ONCE Option:**<br><br>`ONCE` is used to indicate that only one record is to be read. No processing loop is initiated (and therefore the loop-closing keyword `END-WORK` or `LOOP` must not be specified). If `ONCE` is specified, the `AT END OF FILE` clause should also be used.<br><br>If a `READ WORK FILE` statement specified with the `ONCE` option is controlled by a user-initiated processing loop, an end-of-file condition may be detected on the work file before the loop ends. All fields read from the work file still contain the values from the last record read. The work file is then repositioned to the first record which will be read upon the next execution of `READ WORK FILE ONCE`. |

| | |
|---|---|
| **RECORD *operand1* FILLER *n*X** | **RECORD Option:**<br><br>If `RECORD` is specified, all fields in each record read are made available for processing. An operand list (*operand1*) corresponding to the layout of the record must be provided.<br><br>A `FILLER nX` entry indicates *n* bytes are to be skipped in the input record. The record as defined in the `RECORD` clause must be in contiguous storage. `FILLER` is not permitted in structured mode.<br><br>In structured mode, or if the record to be used is defined using a `DEFINE DATA` statement, only one field (or group) may be used. `FILLER` is not permitted in this case.<br><br>No checking and no conversion is performed by Natural on the data contained in the record. It is the user's responsibility to describe the record layout correctly in order to avoid program abends caused by non-numeric data in numeric fields. Because no checking is performed by Natural, this option is the fastest way to process records from a sequential file. The record area defined by *operand1* is filled with blanks before the record is read. Thus, an end-of-file condition will return a cleared area. Short records will have blanks appended.<br><br>**The RECORD option cannot be used:**<br><br>● If an Entire Connection work file is read.<br><br>● If any dynamic variables are used.<br><br>If work file type CSV is used, the RECORD option is ignored and the processing switches to SELECT mode. |

<table>
<tr><td>SELECT</td><td colspan="2">SELECT Option (Default):<br><br>If SELECT is specified, only the fields specified in the operand list ( operand2) will be made available. The position of the field in the input record may be indicated with an OFFSET and/or FILLER specification.</td></tr>
<tr><td></td><td>OFFSET n</td><td>OFFSET 0 indicates the first byte of the record. OFFSET cannot be specified for work files defined as TYPE UNFORMATTED.</td></tr>
<tr><td></td><td>FILLER nX</td><td>Indicates that n bytes are to be skipped in the input record.</td></tr>
<tr><td></td><td colspan="2">Natural will assign the selected values to the individual fields and check that numeric fields as selected from the record actually contain valid numeric data according to their definition. Because checking of selected fields is performed by Natural, this option results in more overhead for the processing of a sequential file.<br><br>If a record does not fill all fields specified in the SELECT option, the following applies:<br><br>● For a field which is only partially filled, the section which has not been filled is reset to blanks or zeros.<br><br>● Fields which are not filled at all still have the contents they had before.<br><br>If the file type CSV is read, the OFFSET option are ignored.</td></tr>
<tr><td>GIVING LENGTH operand3</td><td colspan="2">The GIVING LENGTH clause can be used to retrieve the actual length of the record being read. The length (number of bytes) is returned in operand3.<br><br>operand3 must be defined with format/length I4.<br><br>If the work file is defined as TYPE UNFORMATTED, the length returned indicates the number of bytes read from the byte-stream, including bytes skipped using the FILLER operand.<br><br>If the GIVING LENGTH clause is used with work file type CSV, the operand specified with GIVING LENGTH returns the number of fields in the record (not the length of the record).</td></tr>
<tr><td>AT END OF FILE</td><td colspan="2">The AT END OF FILE clause can only be used in conjunction with the ONCE option. If the ONCE option is used, this clause should be specified to indicate the action to be taken when an end-of-file condition is detected.<br><br>If the ONCE option is not used, an end-of-file condition is handled like a normal processing loop termination.</td></tr>
<tr><td>END-WORK</td><td colspan="2">The Natural reserved word END-WORK must be used to end the READ WORK FILE statement.</td></tr>
</table>

# Field Lengths

The field lengths in the Operand Definition Table are determined as follows:

| Format | Length |
|---|---|
| A, B, I, F | The number of bytes in the input record is the same as the internal length definition. |
| N | The number of bytes in the input record is the sum of internal positions before and after the decimal point. The decimal point and sign do not occupy a byte position in the input record. |
| P, D, T | The number of bytes in the input record is the sum of positions before and after the decimal point plus 1 for the sign, divided by 2 rounded upwards. |
| L | 1 byte is used. For C format fields, 2 bytes are used. |

**Examples of Field Lengths:**

| Field Definition | Input 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 |

See also *Format and Length of User-Defined Variables* in the *Programming Guide*.

# Handling of Large and Dynamic Variables

| Work File Type | Handling |
|---|---|
| UNFORMATTED | Reading a dynamic variable from an UNFORMATTED work file puts the complete rest of the file into the variable (from the current position). If the file exceeds 1073741824 bytes, then a maximum of 1073741824 bytes is placed into the variable.<br><br>Format: `UNFORMATTED` |
| FORMATTED | Reading a dynamic variable from a FORMATTED work file fills the variable in its currently defined length (including length 0). If the end-of-file is reached, the remainder of the current field is filled with blanks. The subsequent fields are unchanged. |

# Example

```
** Example 'RWFEX1': READ WORK FILE
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
*
1 #RECORD
  2 #PERS-ID (A8)
  2 #NAME    (A20)
END-DEFINE
*
FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  WRITE WORK FILE 1
        PERSONNEL-ID NAME
END-FIND
*
* ...
*
READ WORK FILE 1 RECORD #RECORD
  DISPLAY NOTITLE #PERS-ID #NAME
END-WORK
*
END
```

### Output of Program RWFEX1:

```
#PERS-ID        #NAME
-------- --------------------

11100328 BERGHAUS
11100329 BARTHEL
11300313 AECKERLE
11300316 KANTE
11500304 KLUGE
11500308 DIETRICH
11500318 GASSNER
11500343 ROEHM
11600303 BERGER
11600320 BLAETTEL
11500336 JASPER
11100330 BUSH
11500328 EGGERT
```