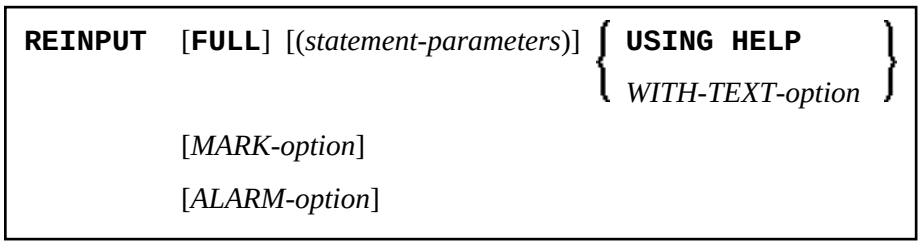
REINPUT



This chapter covers the following topics:

- Function
- Syntax Description
- Examples

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

Related Statements: `DEFINE WINDOW` | `INPUT` | `SET WINDOW`

Belongs to Function Group: *Screen Generation for Interactive Processing*

Function

The `REINPUT` statement is used to return to and re-execute an `INPUT` statement. It is generally used to display a message indicating that the data input as a result of the previous `INPUT` statement were invalid. See *Example 1*.

No `WRITE` or `DISPLAY` statements may be executed between an `INPUT` statement and its corresponding `REINPUT` statement. The `REINPUT` statement is not valid in batch mode.

The `REINPUT` statement, when executed, repositions the program status regarding subroutine, special condition and loop processing as it existed when the `INPUT` statement was executed (as long as the status of the `INPUT` statement is still active). If the loop was initiated after the execution of the `INPUT` statement and the `REINPUT` statement is within this loop, the loop will be discontinued and then restarted after the `INPUT` statement has been reprocessed as a result of `REINPUT`.

If a hierarchy of subroutines was invoked after the execution of the `INPUT` statement, and the `REINPUT` is performed within a subroutine, Natural will trace back all subroutines automatically and reposition the program status to that of the `INPUT` statement.

It is not possible, however, to have an `INPUT` statement positioned within a loop, a subroutine or a special condition block, and then execute the `REINPUT` statement when the status under which the `INPUT` statement was executed has already been terminated. An error message will be produced and program execution terminated when this error condition is detected.

Note:
The execution of a `REINPUT` statement (without `FULL` option) does not reset the `MODIFIED` status of an attribute control variable used in the corresponding `INPUT` statement. To check if an attribute control

variable has been assigned the status MODIFIED, use the MODIFIED option.

Syntax Description

Syntax Element	Description
REINPUT FULL	**FULL Option:** If you specify the FULL option in a REINPUT statement, the corresponding INPUT statement will be re-executed fully: • With an ordinary REINPUT statement (without FULL option), the contents of variables that were changed between the INPUT and REINPUT statement will not be displayed; that is, all variables on the screen will show the contents they had when the INPUT statement was originally executed. • With a REINPUT FULL statement, all changes that have been made after the initial execution of the INPUT statement will be applied to the INPUT statement when it is re-executed; that is, all variables on the screen contain the values they had when the REINPUT statement was executed. **Note:** The contents of input-only fields (AD=A) will be deleted again by REINPUT FULL. Another characteristic of the REINPUT FULL statement is that the status of attribute control variables is reset to NOT MODIFIED. This is not done with the ordinary REINPUT statement. To check if an attribute control variable has been assigned the status MODIFIED, use the *MODIFIED option*. See also *Example 3 - REINPUT FULL WITH MARK POSITION*.

Syntax Element	Description
statement-parameters	**Parameters:** Parameters specified in a REINPUT statement will be applied to all fields specified in the statement. Any parameter specified at element (field) level (see *MARK Option*) will override any corresponding parameter at statement level. Parameters that can be specified with the REINPUT statement: — Specification (S = at statement level, E = at element level) AD — Attribute Definition* — SE CD — Color Definition — S * If AD=P is specified at statement level, all fields - except those used in the MARK option - are protected. The individual session parameters are described in the *Parameter Reference*.
USING HELP	**USING HELP Option:** This option causes the helproutine defined for the INPUT map to be invoked. USING HELP used in combination with the MARK option causes the helproutine defined for the first field specified in the MARK option to be invoked. If no helproutine is defined for that field, the helproutine for the map will be invoked. Example: `REINPUT USING HELP MARK 3` As a result, the helproutine defined for the third field in the INPUT map will be invoked.
WITH-TEXT-option	**WITH TEXT Option:** The WITH TEXT option is used to provide text which is to be displayed in the message line. See *WITH TEXT Option* below.

Syntax Element	Description
MARK-option	**MARK Option** With the MARK option, you can mark a specific field, that is, specify a field in which the cursor is to be placed when the REINPUT statement is executed. See *MARK Option* below.
ALARM-option	**ALARM Option:** This option causes the sound alarm feature of the terminal to be activated when the REINPUT statement is executed. See *ALARM Option* below.

WITH TEXT Option

WITH TEXT is used to provide text which is to be displayed in the message line. This is usually a message indicating what action should be taken to process the screen or to correct an error.

[**WITH**] [**TEXT**] { ***** *operand1* / *operand2* } [(*attributes*)] [,*operand3*]**...** *7*

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1	C	S						N	P	I		B *						yes	no
operand2	C	S				A	U											yes	no
operand3	C	S				A	U	N	P	I	F	B	D	T	L			yes	no

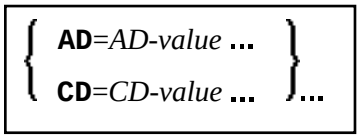
* Format B of *operand1* may be used only with a length of less than or equal to 4.

Syntax Element Description:

Syntax Element	Description
operand1	**Message Text from Natural Message File:** *operand1* represents the number of a message text that is to be retrieved from a Natural message file. You can retrieve either user-defined messages or Natural system messages: • If you specify a positive value of up to four digits (for example: 954), you will retrieve user-defined messages. • If you specify a negative value of up to four digits (for example: -954), you will retrieve Natural system messages. See also *Example 4 - WITH TEXT Options*. Natural message files are created and maintained with the SYSERR utility as described in the relevant documentation.
operand2	**Message Text:** *operand2* represents the message to be placed in the message line. See also *Example 4 - WITH TEXT Options*.
attributes	It is possible to assign various output attributes for *operand1*/*operand2*. These attributes and the syntax that may be used are described in the section *Output Attributes* below.
operand3	**Dynamic Replacement of Message Text:** *operand3* represents a numeric or text constant or the name of a variable. The values provided are used to replace parts of a message text that are either specified with *operand1* or *operand2*. The notation :*n*: is used within the message text as a reference to *operand3* contents, where *n* represents the occurrence (1 - 7) of *operand3*. See also *Example 4 - WITH TEXT Options*. **Note:** Multiple specifications of *operand3* must be separated from each other by a comma. If the comma is used as a decimal character (as defined with the session parameter DC) and numeric constants are specified as *operand3*, put blanks before and after the comma so that it cannot be misinterpreted as a decimal character. Alternatively, multiple specifications of *operand3* can be separated by the input delimiter character (as defined with the session parameter ID); however, this is not possible in the case of ID=/ (slash). Leading zeros or trailing blanks will be removed from the field value before it is displayed in a message.

Output Attributes

attributes indicates the output attributes to be used for text display. Attributes may be:



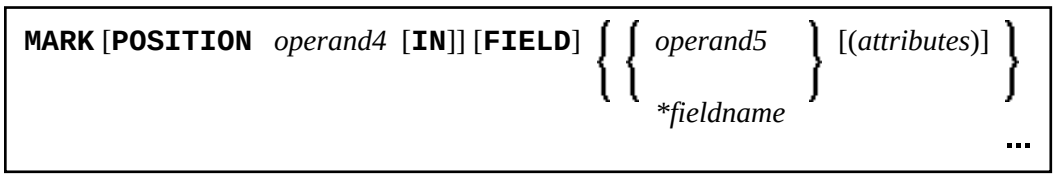
For the possible session parameter values, refer to the corresponding sections in the *Parameter Reference* documentation:

- *AD - Attribute Definition*, section *Field Representation*
- *CD - Color Definition*

Note:
The compiler actually accepts more than one attribute value for an output field. For example, you may specify: `AD=BDI`. In such a case, however, only the last value applies. In the given example, only the value `I` will become effective and the output field will be displayed intensified.

MARK Option

With the `MARK` option, you can mark a specific field, that is, specify a field in which the cursor is to be placed when the `REINPUT` statement is executed. You can also mark a specific position within a field. Moreover, you can make fields input-protected, and change their display and color attributes.



Operand Definition Table:

Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
operand4	C	S						N	P	I								yes	no
operand5	C	S	A					N	P	I								yes	no

Syntax Element Description:

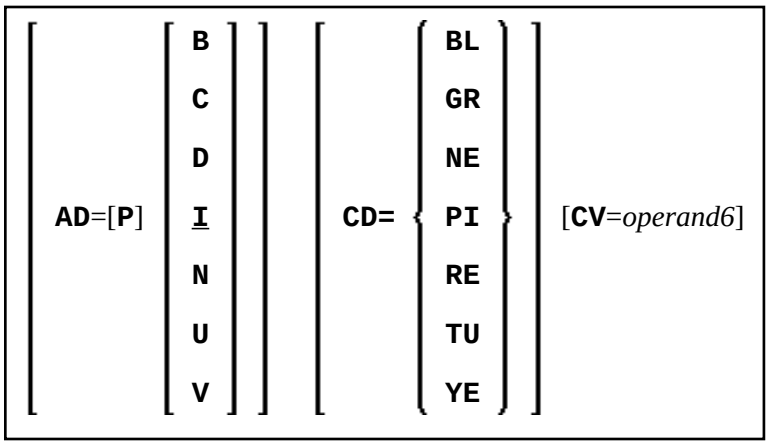
Syntax Element	Description

operand5	**Field to be Marked:**
	
	All AD=A or AD=M (that is, non-protected) fields specified in an INPUT statement are sequentially numbered (beginning with 1) by Natural. *operand5* represents the number of the field in which the cursor is to be positioned.
	
	The **fieldname* notation is used to position to a field (as used in the INPUT statement) using the name of the field as a reference.
	
	If the corresponding INPUT field is an array, a unique index or an index range may be used to reference one or more occurrences of the array.
	
	`INPUT #ARRAY (A1/1:5)`
	`...`
	`REINPUT (AD=P) 'TEXT' MARK *#ARRAY (2:3)`
	
	If *operand5* is also an array, the values in *operand5* are used as field numbers for the INPUT array.
	
	`RESET #X(N2/1:2)`
	`INPUT #ARRAY ...`
	`...`
	`REINPUT (AD=P) 'TEXT' MARK #X (1:2)`

MARK POSITION	**MARK POSITION Option:** With this option, you can have the cursor placed at a specific position - as specified with *operand4* - within a field. See also *Example 3 - REINPUT FULL WITH MARK POSITION*.
operand4	**Cursor Position:** *operand4* specifies the cursor position. *operand4* must not contain decimal digits.
attributes	**Attribute Assignments:** See *Attribute Assignments* below.

Attribute Assignments

With explicit attributes, you can define the display presentation and color of the WITH TEXT message and also the layout of the MARK field (which is positioned by the REINPUT statement).



Operand Definition Table:

Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
operand6	S	C	no	no

With the attribute AD=P, you can make an input field (AD=A or AD=M) input-protected.

Note:
You cannot use an attribute to make output-only fields (AD=O) available for input.

For information on the attributes AD, CD and CV, refer to the *Parameter Reference*.

The attributes for the WITH TEXT and MARK fields need not be specified in a fixed manner, but can also be assigned dynamically by means of a control variable which is referenced in a (CV=) clause. If both an AD and a CV option are specified for the same field, the attributes from the AD option are completely ignored, except (AD=P) which remains in effect.

If a CD and a CV option are specified for the same field, the color from the CV option is used. If the CV variable contains no color specification, the color from the CD option is applied to that field.

If AD=P is specified at statement level, all fields except those specified in the MARK option are input-protected.

ALARM Option

This option causes the sound alarm feature of the terminal to be activated when the REINPUT statement is executed. The appropriate hardware must be available to be able to use this feature.

Examples

- Example 1 - REINPUT Statement

- Example 2 - REINPUT with Attribute Assignment
- Example 3 - REINPUT FULL with MARK POSITION
- Example 4 - WITH TEXT Options
- Example 5 - REINPUT with Attribute Assignment Using a Control Variable

Example 1 - REINPUT Statement

```
** Example 'REIEX1': REINPUT
************************************************************************
DEFINE DATA LOCAL
1 #FUNCTION (A1)
1 #PARM     (A1)
END-DEFINE
*
INPUT #FUNCTION #PARM
*
DECIDE FOR FIRST CONDITION
  WHEN #FUNCTION = 'A' AND #PARM = 'X'
    REINPUT 'Function A with parameter X selected.'
            MARK *#PARM
  WHEN #FUNCTION = 'C' THRU 'D'
    REINPUT 'Function C or D selected.'
  WHEN #FUNCTION = 'X'
    STOP
  WHEN NONE
    REINPUT 'Please enter a valid function.'
            MARK *#FUNCTION
END-DECIDE
*
END
```

Output of Program REIEX1:

```
#FUNCTION A #PARM Y
```

And after pressing ENTER:

```
PLEASE ENTER A VALID FUNCTION
#FUNCTION A #PARM Y
```

Example 2 - REINPUT with Attribute Assignment

```
** Example 'REIEX2': REINPUT  (with attributes)
************************************************************************
DEFINE DATA LOCAL
1 #A (A20)
1 #B (N7.2)
1 #C (A5)
1 #D (N3)
END-DEFINE
*
INPUT (AD=A) #A  #B  #C  #D
*
IF #A = ' ' OR #B = 0
  REINPUT (AD=P) 'RETYPE VALUES'
                 MARK *#A (AD=I CD=RE)    /* put cursor on first field
```

```
                      *#B (AD=U CD=PI)    /* and change colours
END-IF
*
END
```

Example 3 - REINPUT FULL with MARK POSITION

```
** Example 'REIEX3': REINPUT  (with FULL and POSITION option)
************************************************************************
DEFINE DATA LOCAL
1 #A (A20)
1 #B (N7.2)
1 #C (A5)
1 #D (N3)
END-DEFINE
*
INPUT (AD=M) #A  #B  #C  #D
*
IF #A = ' '
  COMPUTE #B = #B + #D
  RESET #D
END-IF
*
IF #A = SCAN 'TEST' OR = ' '
REINPUT FULL 'RETYPE VALUES' MARK POSITION 5 IN *#A
END-IF
*
END
```

Output of Program REIEX3:

```
RETYPE VALUES
#A                     #B        0.00 #C       #D    0
```

Example 4 - WITH TEXT Options

```
** Example 'REIEX4': REINPUT (with TEXT option)
************************************************************************
DEFINE DATA LOCAL
01 #NAME   (A8)
01 #TEXT   (A20)
END-DEFINE
*
*
INPUT WITH TEXT 'Enter a program name.' 'Program name:' #NAME
*
IF #NAME = ' '
  REINPUT WITH TEXT 'Input missing. Enter a name.'
END-IF
*
IF #NAME NE MASK (A)
  MOVE 'Invalid input.' TO #TEXT
  REINPUT WITH TEXT ':1: Name must start with a letter.',#TEXT
ELSE
  /* Using Natural error message 7600 for demonstration
  COMPRESS *INIT-USER 'on' *DAT4I INTO #TEXT
  INPUT WITH TEXT *-7600,#NAME,#TEXT 'Input accepted.'
END-IF
END
```

Example 5 - REINPUT with Attribute Assignment Using a Control Variable

```
DEFINE DATA LOCAL
1 #HELLO (A5)  INIT <’HELO’>
1 #VAR   (A20) INIT <’Enter "HELLO"’>
1 #CV (C)
END-DEFINE
*
INPUT (IP=OFF) #HELLO (AD=M)
*
IF #HELLO NE ’HELLO’ THEN
  MOVE (AD=U CD=RE) TO #CV
  REINPUT FULL WITH TEXT #VAR (CD=YE)
               MARK *#HELLO   (CV=#CV)
END-IF
END
```