

REINPUT

<pre> REINPUT [FULL] [(statement-parameters)] { USING HELP } [MARK-option] [WITH-TEXT-option] [ALARM-option] </pre>
--

Related Statement: INPUT

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.

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 a control variable used in the corresponding INPUT statement.

REINPUT FULL

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.

statement-parameters

Parameters specified in a REINPUT statement will be applied to all fields specified in the statement.

Any parameter specified at field level (see MARK option) will override any corresponding parameter at statement level.

If AD=P is specified as a statement parameter, all fields - except those used in the MARK option - are protected.

For information on the individual parameters see the section Session Parameters in the Natural Reference documentation.

USING HELP

USING HELP causes the helproutine defined for the INPUT map to be invoked.

USING HELP used in combination with the MARK option (see below) 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] { *operand1  
operand2 } [(attributes)] [, operand3]...7
```

Operand	Possible Structure		Possible Formats											Referencing Permitted	Dynamic Definition					
Operand1	C	S						N	P	I		B							yes	no
Operand2	C	S					A											yes	no	
Operand3	C	S					A	N	P	I	F	B	D	T	L			yes	no	

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.

Message Text from Natural Message File - *operand1

Operand1 represents the number of the message text as stored in the Natural message file. The value provided is used to access the Natural message file in order to retrieve a message text.

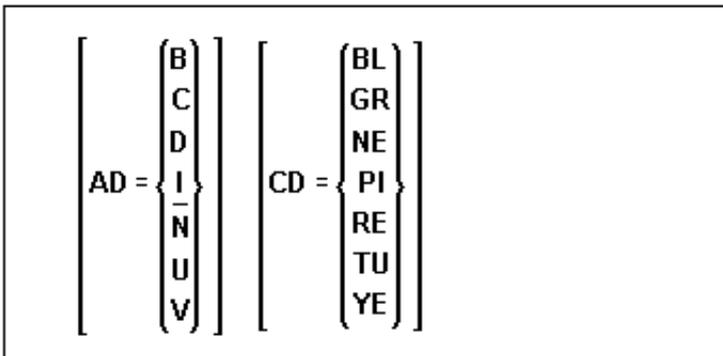
Messages are stored in the Natural message file by library. A maximum of 9999 messages may be stored per library.

See Generating Message Files in the Natural SYSERR Utility documentation for more information.

Message Text - operand2 and Attributes - attributes

Operand2 represents the message to be placed in the message line.

Attributes may be used to assign display and color attributes for *operand1/2*. The following attributes may be specified:



With AD= you specify a display attribute; with CD= you specify a color attribute (see also the session parameters AD and CD in the Natural Reference documentation).

Dynamic Replacement of Message Text - operand3

Operand3 represents a numeric or text constant or the name of a variable.

The values provided are used to replace parts of the message text.

The notation ":n:" is used within the message text as a reference to *operand3* contents, where "n" represents the occurrence (1 - 7) of *operand3*.

Example:

```

...
MOVE 'MESSAGE-1' TO #FIELD
...
REINPUT 'THE ERROR IS :1:',#FIELD
...

```

As a result, the following message will be output:

THE ERROR IS MESSAGE-1

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).

Insignificant zeros or blanks will be removed from the field value before it is displayed in a message.

MARK-option

```

MARK [POSITION operand4 [IN]] [FIELD] { { operand5 } { *fieldname } [[attributes]] } ...
    
```

Operand	Possible Structure		Possible Formats												Referencing Permitted	Dynamic Definition			
Operand4	C	S																yes	no
Operand5	C	S	A															yes	no

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.

Field to be Marked - operand5

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.

```

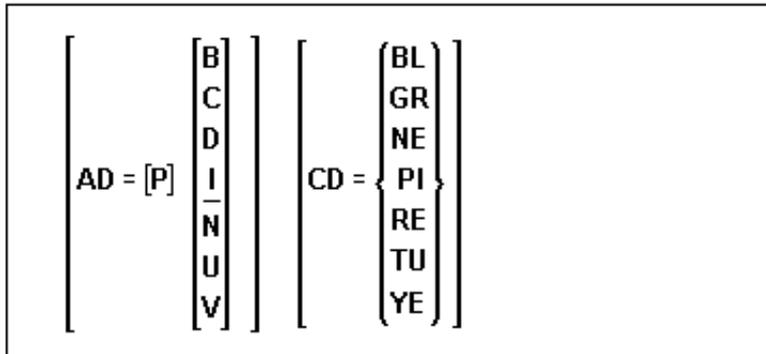
RESET #X(N2/1:2)
INPUT #ARRAY ...
...
REINPUT (AD=P) 'TEXT' MARK #X (1:2)
    
```

MARK POSITION

With MARK POSITION, you can have the cursor placed at a specific position - as specified with *operand4* - within a field.

Operand4 must not contain decimal digits.

attributes



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

Note:

It is not possible via an attribute to make output-only fields (AD=O) available for input.

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

Moreover, you can change display and color attributes of fields. For information on these attributes, see the session parameters AD and CD in the Natural Reference documentation.

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.

Example 1

```

/* EXAMPLE 'REIEX1': REINPUT
/*****
/* IF FUNCTION = A AND PARM = X
/*   ROUTINE-A IS TO BE EXECUTED.
/* IF FUNCTION = B AND PARM = X
/*   ROUTINE-B IS TO BE EXECUTED.
/* IF FUNCTION = C THRU D
/*   ROUTINE-CD IS TO BE EXECUTED.
/* FOR ALL OTHER CASES,
/*   REINPUT STATEMENT IS TO BE EXECUTED.
/*****
DEFINE DATA LOCAL
1 #FUNCTION (A1)
1 #PARM (A1)
END-DEFINE
/*****
INPUT #FUNCTION #PARM
/*****
DECIDE FOR FIRST CONDITION
    WHEN #FUNCTION = 'A' AND #PARM = 'X'
        PERFORM ROUTINE-A
    WHEN #FUNCTION = 'B' AND #PARM = 'X'
        PERFORM ROUTINE-B
    WHEN #FUNCTION = 'C' THRU 'D'
        PERFORM ROUTINE-CD
    WHEN NONE
        REINPUT 'PLEASE ENTER A VALID FUNCTION'
        MARK **FUNCTION
END-DECIDE
/*****
END

```

```
#FUNCTION a #PARM y
```

```
PLEASE ENTER A VALID FUNCTION
#FUNCTION a #PARM y
```

Example 2

```

/* EXAMPLE 'REIEX2': REINPUT WITH ATTRIBUTE ASSIGNMENT
/*****
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)
          *#B (AD=U CD=PI)
END-IF
/*****
END

```

Example 3

```

/* EXAMPLE 'REIEX3': REINPUT FULL WITH POSITION
/*****
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

```

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