INPUT Syntax 1 - Dynamic Screen Layout Specification

This form of the `INPUT` statement is used to create a layout of an `INPUT` screen, or to create an `INPUT` data layout which is to be read (on mainframe computers) in batch mode from a sequential input file.

```
INPUT [WINDOW='window-name'] [NO ERASE]
      [(statement-parameters)]
      [WITH-TEXT-option]
      [MARK-option]
      [ALARM-option]
  ⎧          ⎧ ⎧ 'text'  ⎫ [(attributes)]                                 ⎫ ⎫
  ⎪  ⎡    ⎤  ⎪ ⎩ 'c' (n) ⎭                                                ⎪ ⎪
  ⎪  ⎢ nX ⎥  ⎪                                                           ⎪ ⎪
  ⎪  ⎢ nT ⎥  ⎪            ⎡        ⎤                                      ⎪ ⎪
  ⎨  ⎢ x/y⎥  ⎨            ⎢ *IN    ⎥                                      ⎬ ⎬
  ⎪  ⎢ /  ⎥  ⎪  ⎡ '_' ⎤   ⎣ *OUT   ⎦ {operand1 [(parameter)]}             ⎪ ⎪
  ⎪  ⎣    ⎦  ⎪  ⎣ '=' ⎦     *OUTIN                                        ⎪ ⎪
  ⎩      ... ⎩                                                           ⎭ ⎭ ...
```

This chapter covers the following topics:

- INPUT Syntax 1 - Description
- Examples - Syntax 1

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

INPUT Syntax 1 - Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
operand1		S	A	G	N	A	U	N	P	I	F	B	D	T	L		G		yes	yes

Syntax Element Description:

INPUT WINDOW=’*window-name*’

With the option WINDOW=’*window-name*’, you indicate that the INPUT statement is to be executed for the specified window. The specified window must be defined in a DEFINE WINDOW statement. The specified window is only active for the duration of that INPUT statement, and is automatically deactivated when the INPUT statement has been executed.

See also the statements DEFINE WINDOW and SET WINDOW.

NO ERASE

NO ERASE causes a screen map of an INPUT statement to be overlaid onto an existing screen without erasing the screen contents.

Screen as used here refers to a logical screen rather than a physical screen.

All unprotected fields that existed on the screen are converted to protected (display only) fields. The old data remain on the screen until the new layout is displayed. If a field from the new screen content partially overlays an existing field, the one character before the new field and the next character in the existing field will be replaced by a blank.

statement-parameters

One or more parameters, enclosed within parentheses, may be specified immediately after the INPUT statement or an element being displayed.

For a list of parameters that can be specified with the INPUT statement, refer to the section *Statement Parameters* below.

Each parameter specified in this manner will override any previous parameter specified in a GLOBALS command, SET GLOBALS or FORMAT statement. If more than one parameter is specified, one or more blanks must be present between each entry. An entry may not be split between two statement lines.

The parameter settings applied here will only be regarded for variable fields, but they have no effect on text-constants. If you would like to set field attributes for a text-constant, they have to be set explicitly for this element.

Example:

```
DEFINE DATA LOCAL
1 VARI (A4)     INIT <’1234’>                  /*    Displays
END-DEFINE                                     /*       as
FORMAT AD=M                                    /*    ---------
INPUT           ’Text’           VARI          /*    Text 1234
INPUT (PM=I)    ’Text’           VARI          /*    Text 4321
INPUT           ’Text’ (PM=I)    VARI (PM=I)   /*    txeT 4321
INPUT           ’Text’ (PM=I)    VARI          /*    txeT 1234
END
```

Examples of using parameters at the statement and element level are provided below.

WITH TEXT-option

WITH TEXT is used to provide text which is to be displayed in the message line (see the corresponding section below).

MARK-option

See the section *MARK Option* below).

ALARM-option

See the section *Alarm Option* below.

Other syntax elements (*n*X, *n*T, *x/y*, *operand1*, etc.)	See the section *Field Positioning, Text Specification, Attribute Assignment* below.

Statement Parameters

Parameters that can be specified with the INPUT statement		**Specification (S = at statement level, E = at element level)**
AD	Attribute Definition	SE
AL	Alphanumeric Length for Output	SE
BX	Box Definition	SE
CD	Color Definition	SE
CV	Control Variable	SE
DF	Date Format	SE
DL	Display Length for Output	SE
DY	Dynamic Attributes	SE
EM	Edit Mask	SE
FL	Floating Point Mantissa Length	SE
HE	Helproutine	SE
IP	Input Prompting Text	SE
LS	Line Size	S
MC	Multiple-Value Field Count	S
MS	Manual Skip	S
NL	Numeric Length for Output	SE
PC	Periodic Group Count	S
PM	Print Mode	SE
PS	Page Size *	S
SB	Selection Box	E
SG	Sign Position	SE
ZP	Zero Printing	SE

* If the number of occurrences of an array exceeds the PS value, a NAT0303 error will be output.

The individual session parameters are described in the *Parameter Reference*.

WITH TEXT Option

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

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1	C	S					N	P	I		B*							yes	yes
operand2	C	S				A												yes	yes
operand3	C	S				A	N	P	I	F	B	D	T	L				yes	yes

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

`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.

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* in the description of the `REINPUT` statement. 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* in the description of the `REINPUT` statement.
attributes	It is possible to assign various output attributes for *operand1/2*. 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 *operand3* occurrence (1 - 7). See also *Example 4 - WITH TEXT Options* in the description of the `REINPUT` statement. **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), because the slash has a different meaning in the `INPUT` statement syntax. 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:

```
⎧ ⎡ AD=AD-value ...  ⎤      ⎫
⎪ ⎢ BX=BX-value ...  ⎥      ⎪
⎨ ⎨ CD=CD-value ...  ⎬      ⎬
⎪ ⎢ PM=PM-value ...  ⎥      ⎪
⎪ ⎣                  ⎦...   ⎪
⎩ ⎧ AD-value ...     ⎫      ⎭
  ⎩ CD-value ...     ⎭...
```

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*
- *BX - Box Definition*
- *PM - Print Mode*

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 cause the cursor to be placed at any non-protected field on screen. In addition, you can specify the position of the cursor within that field. By default, that is, when the `MARK` option is omitted, the cursor is placed at the beginning of the first non-protected field.

```
MARK [POSITION operand4 [IN]] [FIELD] ⎧ operand1   ⎫
                                      ⎩ *fieldname ⎭
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand4	C	S					N	P	I									yes	yes
operand1	C	S	A				N	P	I									yes	yes

Syntax Element Description:

operand1	**Field Reference Number:** *operand1* specifies the number of the field where the cursor is to be positioned in. Each field attribute AD=A or AD=M (that is, non-protected field) specified in an INPUT statement is assigned a field reference number, beginning with 1.
****fieldname***	**Field Name for Referencing:** Instead of the field reference number, the field name may be used to position to a field, using the **fieldname* notation.
operand4	**Cursor Position within Referenced Field:** With MARK POSITION, you can have the cursor placed at a specific position - as specified with *operand4* - within a field specified with *operand1* or **fieldname*. *operand4* must not contain decimal digits.

Examples:

```
MARK #NUMBER                /* Field number
MARK 3                      /* Third map field
MARK *#FIELD1               /* Map field
MARK POSITION 3 IN #NUMBER /* Third character in field number
```

See also *Example 3 - INPUT Statement with MARK POSITION Option* at the end of this section.

ALARM Option

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

[[**AND**] [**SOUND**] **ALARM**]

Default Prompting Text

Unless the session parameter IP (input prompting) is set to IP=OFF, the field name of the field used in an INPUT statement will be displayed preceding the field value (forms mode) or as a prompting keyword to select the field (keyword/delimiter mode). This default field name may be overridden by specifying either a *'text'* element (which replaces the default name) or '-' (which suppresses the display of the default field name) immediately preceding the field name.

Field Positioning, Text Specification, Attribute Assignment

```
         ┌ 'text'   [(attributes)] ┐
┌ nX  ┐  │ 'c' (n) [(attributes)]  │ ┌ *IN    ┐ {operand1 [(parameter(s))]}
│ nT  │  │         '-'             │ │ *OUT   │
└ x/y ┘  │         '='             │ └ *OUTIN ┘
         └         /...            ┘
```

Several notations are available for field positioning, attribute assignment, and text creation.

***n*X**	Causes *n* spaces to be inserted between fields. **Note:** This notation inserts *n* spaces between columns. *n* must not be zero.
***n*T**	Causes positioning (tabulation) to print position *n*.
x/y	Places the next element on line *x*, beginning in column *y*. *y* must not be zero. Backward positioning in the same line is not permitted.
'*text*'	Causes *text* to be displayed write protected; see also *Text Notation*, *Defining a Text to Be Used with a Statement*.
'*c*' (*n*)	Identical to `'text'`, except that the character *c* is displayed *n* times. *n* must be 1 - 132; see also *Text Notation*, *Defining a Character to Be Displayed n Times before a Field Value*.
attributes	Indicates the attributes to be used for display. See *Attributes* below.
'-'	**Minus Sign:** When placed before a field, `'-'` suppresses the generation of a field name as prompting text. **Note:** Any text string before a field will replace the field name as prompting text.
'='	**Equal Sign:** When placed before a field, `'='` results in the display of the field heading followed by the field contents.

’/’	**Slash Sign:** When placed between fields or text elements, `'/'` causes positioning to the beginning of the next print line. The contents of fields may be specified for input, output only, and output for modification using the attribute settings `AD=A`, `AD=O`, and `AD=M` respectively. The default is `AD=A`. All fields specified with `AD=A` (input only) or `AD=M` (output for modification) will create unprotected fields on the screen. A value for such a field may be entered by the user. For TTY devices, output for modification fields will occupy twice the size of the field (one for output, one for input) so that a new value may be entered. An input field (`AD=A/M`) specified as non-displayable will always start on a new line on a TTY device. Example: `INPUT #A (AD=A) #B (AD=O) #C (AD=M)` `#A` is an input field which is unprotected, i.e., a value is to be entered for the field. `#B` is a field which is to be displayed write-protected, i.e., no value may be entered for the field. `#C` is a field whose current value is to be displayed, and the value may be modified by entering a new value for the field.
***IN, *OUT and *OUTIN**	Equivalent to the attributes `AD=A`, `AD=O`, `AD=M` respectively. **Note:** If a non-modifiable system variable is used in an `INPUT` statement, the value will be displayed as an output-only field `AD=O` or `*OUT` attribute.

operand1	**Field Specification:** *operand1* represents the field to be used. Database fields or user-defined variables may be specified. Natural directly maps the content of each field from the data area to the `INPUT` statement, no move operation is necessary. When the content of a database field is modified as a result of `INPUT` processing, only the value as contained in the data area is modified. Appropriate database `UPDATE` / `STORE` statements must be used to change the content of the database. When the name of a group of database fields is referenced in an `INPUT` statement, all fields belonging to that group will be individually used as input fields. When reference is made to a range of occurrences within an array, all occurrences are individually processed as input fields, but no prompting text will be created for each individual occurrence, only for the first one. On mainframe computers, arrays with ranges that allow to vary the number of occurrences at execution time may not be specified.
parameter(s)	One or more parameters, enclosed within parentheses, may be specified immediately after *operand1* (see table and example below). Each parameter specified will override any previous parameter specified in a `GLOBALS` command, `SET GLOBALS` (in Reporting Mode) or `FORMAT` statement. If more than one parameter is specified, they must be separated by one or more blanks from one another. Each parameter specification must not be split between two statement lines. The parameter settings applied here will only be regarded for variable fields, but they have no effect on text-constants. If you would like to set field attributes for a text-constant, they have to be set explicitly for this element. For information on the individual parameters, see the table in the section *Statement Parameters*. **Note:** The session parameter `EM` will be referenced dynamically in the DDM if an edit mask is defined for a database field. Edit masks may be specified for output and input fields. When an edit mask is defined for an input field, the data for the field must be entered according to the edit mask specification.

Attributes

The following attributes may be used:

```
[      { B } ]  [      { BL } ]  [      { T   } ]  [      {   } ]
[      { C } ]  [      { GR } ]  [      { B   } ]  [      { C } ]
[      { D } ]  [      { NE } ]  [      { L   } ]  [      { D } ]
[ [AD=]{ I } ]  [ [CD=]{ PI } ]  [ [BX=]{ R   } ]  [ [PM=]{ I } ]
[      { N } ]  [      { RE } ]  [      { ON  } ]  [      { N } ]
[      { U } ]  [      { TU } ]  [      { OFF } ]  [      {   } ]
[      { V } ]  [      { YE } ]  [      {     } ]  [      {   } ]
         1               2                3                4
```

1. Display attributes; see the session parameter AD (in the *Parameter Reference*).
2. Color attributes; see the session parameter CD (in the *Parameter Reference*).
3. Outlining attributes; see the session parameter BX (in the *Parameter Reference*).
4. Print mode attributes; see the session parameter PM (in the *Parameter Reference*).

Examples - Syntax 1

- Example 1 - INPUT Statement
- Example 2 - INPUT Statement with DEFINE WINDOW Statement
- Example 3 - INPUT Statement with MARK POSITION Option

Example 1 - INPUT Statement

```
** Example 'IPTEX1': INPUT
************************************************************************
DEFINE DATA LOCAL
1 #FNC  (A1)
END-DEFINE
*
INPUT 10X 'SELECTION MENU FOR EMPLOYEES SYSTEM' /
      10X '-' (35) //
      10X 'ADD      (A)' /
      10X 'UPDATE   (U)' /
      10X 'DELETE   (D)' /
      10X 'STOP     (.)' //
      10X 'PLEASE ENTER FUNCTION: ' #FNC
*
DECIDE ON EVERY VALUE OF #FNC
  VALUE 'A'   /* invoke the object containing the add function here
    WRITE 'Add function selected.'
  VALUE 'U'   /* invoke the object containing the update function here
```

```
    WRITE 'Update function selected.'
  VALUE 'D'   /* invoke the object containing the delete function here
    WRITE 'Delete function selected.'
  VALUE '.'
    STOP
  NONE
    REINPUT 'Please enter a valid function.' MARK *#FNC
END-DECIDE
*
END
```

Output of Program IPTEX1:

```
SELECTION MENU FOR EMPLOYEES SYSTEM
-----------------------------------

ADD      (A)
UPDATE   (U)
DELETE   (D)
STOP     (.)

PLEASE ENTER FUNCTION:
```

Example 2 - INPUT Statement with DEFINE WINDOW Statement

```
** Example 'INPEX1': INPUT (with DEFINE WINDOW statement)
************************************************************************
DEFINE DATA LOCAL
1 #STRING (A15)
END-DEFINE
*
DEFINE WINDOW WIND1
  SIZE 10 * 40
  BASE 5 / 10
  FRAMED ON POSITION TEXT
*
INPUT WINDOW='WIND1'
      'PLEASE ENTER HERE:' / #STRING
*
END
```

Output of Program INPEX1:

```
+-----------------------------------Top+
! PLEASE ENTER HERE:                    !
! #STRING                               !
!                                       !
!                                       !
!                                       !
!                                       !
!                                       !
!                                       !
+--------------------------------Bottom+
```

Example 3 - INPUT Statement with MARK POSITION Option

```
** Example 'INPEX2': INPUT (with POSITION)
************************************************************************
DEFINE DATA LOCAL
1 #START (A30)
END-DEFINE
```

```
*
ASSIGN #START = 'EXAM_'
*
INPUT (AD=M) MARK POSITION 5 IN *#START
     / 'PLEASE COMPLETE START VALUE FOR SEARCH'
     / 5X #START
END
```

Output of Program INPEX2:

```
PLEASE COMPLETE START VALUE FOR SEARCH
       #START EXAM[]
```