STACK

```
STACK [TOP] { COMMAND operand1 [operand2 [(parameter)]] ...
            { [DATA] [FORMATTED] {operand2 [(parameter)]} ...
```

This chapter covers the following topics:

- Function
- Syntax Description
- Example

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

Related Statements: `INPUT` | `RELEASE`

Function

The `STACK` statement is used to place any of the following into the Natural stack:

- the name of a Natural program or Natural system command to be executed;
- data to be used during the execution of an `INPUT` statement.

For further information on the stack, see *Further Programming Aspects*, *Stack* in the *Programming Guide*.

Syntax Description

Operand Definition Table:

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

Syntax Element Description:

Syntax Element	Description
TOP	**TOP Option:** If you specify TOP, the data/program/command will be placed at the top of the Natural stack. Otherwise, they are placed at the bottom of the stack. Example: The following statement causes the content of the variable #FIELDA to be placed as data on top of the stack: `STACK TOP #FIELDA`

DATA	**DATA Option:**
	
	This option, which is also the default, causes data to be placed in the stack which are to be used as input data for an INPUT statement.
	
	Delimiter characters or input assign characters contained within the data values will be processed as delimiters. For details on how data from the stack are processed by an INPUT statement, refer to *Processing Data from the Natural Stack* (in the description of the INPUT statement).
	
	Example: The following statements cause the contents of the variables #FIELD1 and #FIELD2 to be placed in the stack:
	
	`MOVE 'ABC' TO #FIELD1`
	`MOVE 'XYZ' TO #FIELD2`
	`STACK #FIELD1 #FIELD2`
	
	These variables will be passed as data to the next INPUT statement in the Natural program, using delimiter mode:
	
	`INPUT #FIELD1  #FIELD2`
	
	Note:
	If *operand2* is a time variable (Format T), only the time component of the variable content is placed in the stack, but not the date component.

Syntax Element	Description

FORMATTED	**FORMATTED Option:**
	
	This option causes all data to be passed on a field-by-field basis to the next INPUT statement; no key assignments or delimiter characters will be interpreted.
	
	Examples:
	
	The following statements cause ABC,DEF to be placed in #FIELD1 and XYZ in #FIELD2:
	
	`MOVE 'ABC,DEF' TO #FIELD1`
	`MOVE 'XYZ'     TO #FIELD2`
	`STACK TOP DATA FORMATTED #FIELD1 #FIELD2`
	`...`
	`INPUT #FIELD1 #FIELD2`
	
	Assuming the input delimiter character to be the comma (profile/session parameter ID=,), the following statements - without the keyword FORMATTED - cause ABC to be placed in #FIELD1 and DEF in #FIELD2:
	
	`MOVE 'ABC,DEF' TO #FIELD1`
	`STACK TOP DATA #FIELD1`
	`...`
	`INPUT #FIELD1 #FIELD2`
	
	Note:
	The FORMATTED option should be used if the data to be passed contains delimiter, control or DBCS characters to avoid unintentional interpretation of these characters.

COMMAND *operand1*	**COMMAND Option:** To place a command (or program name) in the stack, you specify the keyword COMMAND followed by the command specified in *operand1*. Natural will execute the command instead of displaying the NEXT prompt and prompting the user for input. Example: The following statement causes the command RUN to be placed at the top of the stack. Natural will execute this command at the point where the NEXT prompt would normally be issued. `STACK TOP COMMAND 'RUN'`

Syntax Element	Description
COMMAND *operand1 operand2* ...	**COMMAND with Data Option:** Together with a command (*operand1*), you may also place data (*operand2*) in the stack. These data will then be processed by the next INPUT statement after the command has been executed. Data stacked with a command are always stacked unformatted. **Note:** If the data to be stacked include empty alphanumeric fields (that is, blanks), these blanks will be interpreted as delimiters between values and thus not processed correctly by the corresponding INPUT statement. Therefore, if you wish to stack empty alphanumeric fields as data with a command, you have to use two STACK statements: one STACK DATA *operand2* ... to stack the data, and one STACK COMMAND *operand1* to stack the command.
parameter	**Date Format:** If *operand2* is a date variable, you can specify the session parameter DF as a parameter for this variable.

Example

```
** Example 'STKEX1': STACK
************************************************************************
DEFINE DATA LOCAL
1 #CODE (A1)
END-DEFINE
*
INPUT //
  10X 'PLEASE SELECT COMMAND' //
  10X 'LIST VIEW      (V)' /
  10X 'LIST PROGRAM * (P)' /
  10X 'TECH INFO      (T)' /
  10X 'STOP           (.)' //
  20X 'CODE:' #CODE
*
*
DECIDE ON FIRST #CODE
  VALUE 'V'
    STACK TOP DATA    'VIEW'
    STACK TOP COMMAND 'LIST'
  VALUE 'P'
    STACK TOP COMMAND 'LIST PROGRAM *'
  VALUE 'T'
    STACK TOP COMMAND 'LAST *'
    STACK TOP COMMAND 'TECH'
    STACK TOP COMMAND 'SYSPROD'
  VALUE '.'
    STOP
```

```
  NONE
    REINPUT 'PLEASE ENTER VALID CODE'
END-DECIDE
*
*
END
```

Output of Program STKEX1:

```
PLEASE SELECT COMMAND

LIST VIEW      (V)
LIST PROGRAM * (P)
TECH INFO      (T)
STOP           (.)

          CODE:P
```

After entering and confirming code:

```
16:46:28               ***** NATURAL LIST COMMAND *****              2005-01-19
User HTR                 - LIST Objects in a Library -         Library SYSEXSYN

Cmd  Name       Type         S/C  SM Version  User ID    Date        Time
---  *________  P__________  *__  *  *______  *________  *__________ *________
__   ACREX1     Program      S/C  S  4.1.03   RKE        2004-11-11  16:32:37
__   ACREX2     Program      S/C  S  4.1.03   RKE        2005-01-05  10:29:51
__   ADDEX1     Program      S/C  S  4.1.03   RKE        2004-11-11  16:36:49
__   AEDEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  16:40:34
__   AEDEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  16:39:57
__   AEPEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  16:41:57
__   AEPEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  16:42:31
__   AEPEX2     Program      S/C  S  4.1.03   RKE        2004-11-11  16:43:37
__   ASDEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:00:21
__   ASDEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:00:50
__   ASGEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:02:01
__   ASGEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:02:08
__   ATBEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:03:18
__   ATBEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:03:05
                                                                 14 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```