SET KEY

This chapter covers the following topics:

Function

The `SET KEY` statement is used to assign functions to the following types of keys:

- video terminal PA (program attention) keys,
- PF (program function) keys,
- CLEAR key.

When a `SET KEY` statement is executed, Natural receives control of the keys during program execution and uses the values assigned to the keys.

The Natural system variable `*PF-KEY` identifies which key was pressed last.

Note:
If a user presses a key to which no function is assigned, either a warning message will be issued prompting the user to press a valid key, or the value `ENTR` will be placed into the Natural system variable `*PF-KEY`; that is, Natural will react as if the ENTER key had been pressed (this depends on the Natural profile parameter `IKEY` as set by the Natural administrator). Processing of PA and PF keys is also affected by the Natural profile parameter `KEY` as set by the Natural administrator.

See also *Processing Based on Function Keys* (in the *Programming Guide*).

Application Programming Interface: USR4005N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

Syntax Description

Several structures are possible for this statement.

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

Syntax 1 - Affecting All Keys:

```
SET KEY { ALL                  }
        { ON                   }
        { OFF                  }
        {                      }
        { COMMAND { ON  }      }
        {         { OFF }      }
        {                      }
        { NAMED OFF            }
```

Syntax 2 - Affecting Individual Keys:

```
        { { PAn               } [ = { ON                 } ] }
        { { PFn               } [   { OFF                } ] }
SET KEY { { CLR               } [   { DISABLED           } ] }
        { { DYNAMIC operand1  } [   {                    } ] }
        {                       [   { COMMAND { ON  }    } ] }
        {                       [   {         { OFF }    } ] }...
```

Syntax 3 - Affecting Individual Keys:

```
SET KEY {  { PAn                 } [ = { PGM                } ] [ NAMED { operand4 } ] }
           { PFn                 }     { operand2           }               { OFF      }
           { CLR                 }     { HELP               }
           { DYNAMIC operand1    }     { DATA operand3      }

           ENTR [ NAMED                          { operand4 } ]                          } ...
                                                 { OFF      }
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
operand1		S				A													yes	no
operand2	C	S				A	U												yes	no
operand3	C	S				A	U												yes	no
operand4	C	S				A	U												yes	no

Making Keys Program-Sensitive and Deactivating Keys

Making a key program-sensitive means that the key will be available for interrogation by the currently active program. If a key is made program-sensitive, pressing the key has the same effect as pressing ENTER. All data that have been entered on the screen are transferred to the program.

Note:
PA keys and the CLEAR key, when made program-sensitive, do not cause any data to be transferred from the screen.

The program-sensitivity remains in effect only for the execution of the current program. See also the section *SET KEY Statements on Different Program Levels*.

Examples:

SET KEY ALL	This statement causes all keys to be made program-sensitive. All function assignments to any keys are overwritten.

SET KEY PF2
SET KEY PF2=PGM
	Each of these statements causes PF2 to be made program-sensitive.

SET KEY OFF	This statement de-activates all key settings. The Natural system variable `*PF-KEY` contains `ENTR` after `SET KEY OFF` has been executed.
SET KEY ON	This statement re-activates the functions assigned to all keys that had an assignment and re-activates the program-sensitivity of keys that were made program-sensitive before they were de-activated.
SET KEY PF2=OFF	This statement de-activates PF2. After execution of `SET KEY PF2=OFF`, the Natural system variable `*PF-KEY` contains `ENTR` if it contained `PF2` before.
SET KEY PF2=ON	This statement re-activates the function assigned to PF2 before it was de-activated or made program-sensitive. If no function had been assigned to PF2, it will be made program-sensitive again.

Key Program-Sensitivity and Contents of *PF-KEY

The following example shows the relation between the program-sensitivity of a key and the contents of the system variable `*PF-KEY`.

Assume that PF2 has been made program-sensitive by means of `SET KEY PF2=PGM` and an `INPUT` statement is executed afterwards. The table below shows how user actions and executed Natural statements influence the contents of `*PF-KEY`.

Sequence	Natural Statement Executed / User Action	Contents of *PF-KEY
1	User presses PF2.	`PF2`
2	`SET KEY OFF`	`ENTR`
3	`SET KEY ON`	`PF2`
4	`SET KEY PF2=OFF`	`ENTR`
5	`SET KEY PF2=ON`	`PF2`
6	`SET KEY PF3=OFF`	`PF2`

Assigning Commands/Programs

You can assign a command or program name to a key. When the key is pressed, the current program is terminated and the command/program assigned to the key is invoked via the Natural stack. When assigning a command/program, you can also pass parameters to the command/program (see third example below).

You can also assign a terminal command to a key. When the key is pressed, the terminal command assigned to the key is executed.

When *operand2* is specified as a constant, it must be enclosed within apostrophes.

Examples:

SET KEY PF4='SAVE'	The command SAVE is assigned to PF4.
SET KEY PF4=#XYX	The value contained in the variable #XYZ is assigned to PF4.
SET KEY PF6='LIST MAP *'	The command LIST, including the LIST parameters MAP and *, is assigned to PF6.
SET KEY PF2='%%'	The terminal command %% is assigned to PF2.

The assignment remains in effect until it is overwritten by another SET KEY statement, until the user logs on to another application, or until the end of the Natural session. See also the section *SET KEY Statements on Different Program Levels*.

Note:
Before a program invoked via a key is executed, Natural internally issues a BACKOUT TRANSACTION statement.

Assigning Input DATA

You can assign a data string (*operand3*) to a key. When the key is pressed, the data string is placed into the input field in which the cursor is currently positioned, and the data are transferred to the executing program (as if ENTER had been pressed).

When *operand3* is specified as a constant, it must be enclosed within apostrophes.

Example:

```
SET KEY PF12=DATA 'YES'
```

For the validity of a DATA assignment, the same applies as for a command assignment, that is, it remains in effect until it is overwritten by another SET KEY statement, until the user logs on to another application, or until the end of the Natural session. See also the section *SET KEY Statements on Different Program Levels*.

COMMAND OFF/ON

With COMMAND OFF, you can temporarily de-activate any function (command, program, or data) assigned to a key. If the key had been program-sensitive before the function was assigned, COMMAND OFF will make it program-sensitive again.

With a subsequent COMMAND ON, you can re-activate the assigned function again.

Examples:

`SET KEY PF4=COMMAND OFF`	The function that has been assigned to PF4 is temporarily de-activated; if PF4 had been program-sensitive before the function was assigned, it is now made program-sensitive again.
`SET KEY PF4=COMMAND ON`	The function assigned to PF4 is re-activated again.
`SET KEY COMMAND OFF`	All functions assigned to all keys are temporarily de-activated; those keys which had been program-sensitive before functions were assigned to them, are now made program-sensitive again.
`SET KEY COMMAND ON`	All functions assigned to all keys are re-activated again.

With `SET KEY PF`*`nn`*`=''` you can delete the function assigned to a key and at the same time deactivate the program sensitivity of the key.

Assigning HELP

You can assign `HELP` to a key. When the key is pressed, the helproutine assigned to the field in which the cursor is currently positioned will be invoked.

The effect is the same as when entering the help character in the field to invoke help. (The help character - default is a question mark (?) - is determined by the Natural profile parameter `HI` as set by the Natural administrator.)

Example:

```
SET KEY PF1=HELP
```

For the validity of a `HELP` assignment, the same applies as for program-sensitivity, that is, it remains in effect only for the execution of the current program. See also the section *SET KEY Statements on Different Program Levels*.

DYNAMIC Option

Instead of specifying a specific key with the `SET KEY` statement, you can use the `DYNAMIC` option with a variable (*operand1*), and assign a value (PF*n*, PA*n*, CLR) to this variable in the program. This allows you to specify a function and make it dependent on the program logic which key this function is assigned to.

Example:

```
...
IF ...
   MOVE 'PF4' TO #KEY
ELSE
   MOVE 'PF7' TO #KEY
END-IF
...
SET KEY DYNAMIC #KEY = 'SAVE'
...
```

DISABLED Option

Graphical user interface (GUI) elements, such as push buttons, menus, and bitmaps, are implemented as PF keys. With the `DISABLED` option, you can disable the use the of a GUI element assigned to a PF key. Push buttons and menu items will then be displayed grey.

To cancel the effect of `SET KEY PF`*`nn`*`=DISABLED`, you use `SET KEY PF`*`nn`*`=ON`.

Example:

`SET KEY PF10=DISABLED`	Disables the use of the GUI element assigned to PF10.

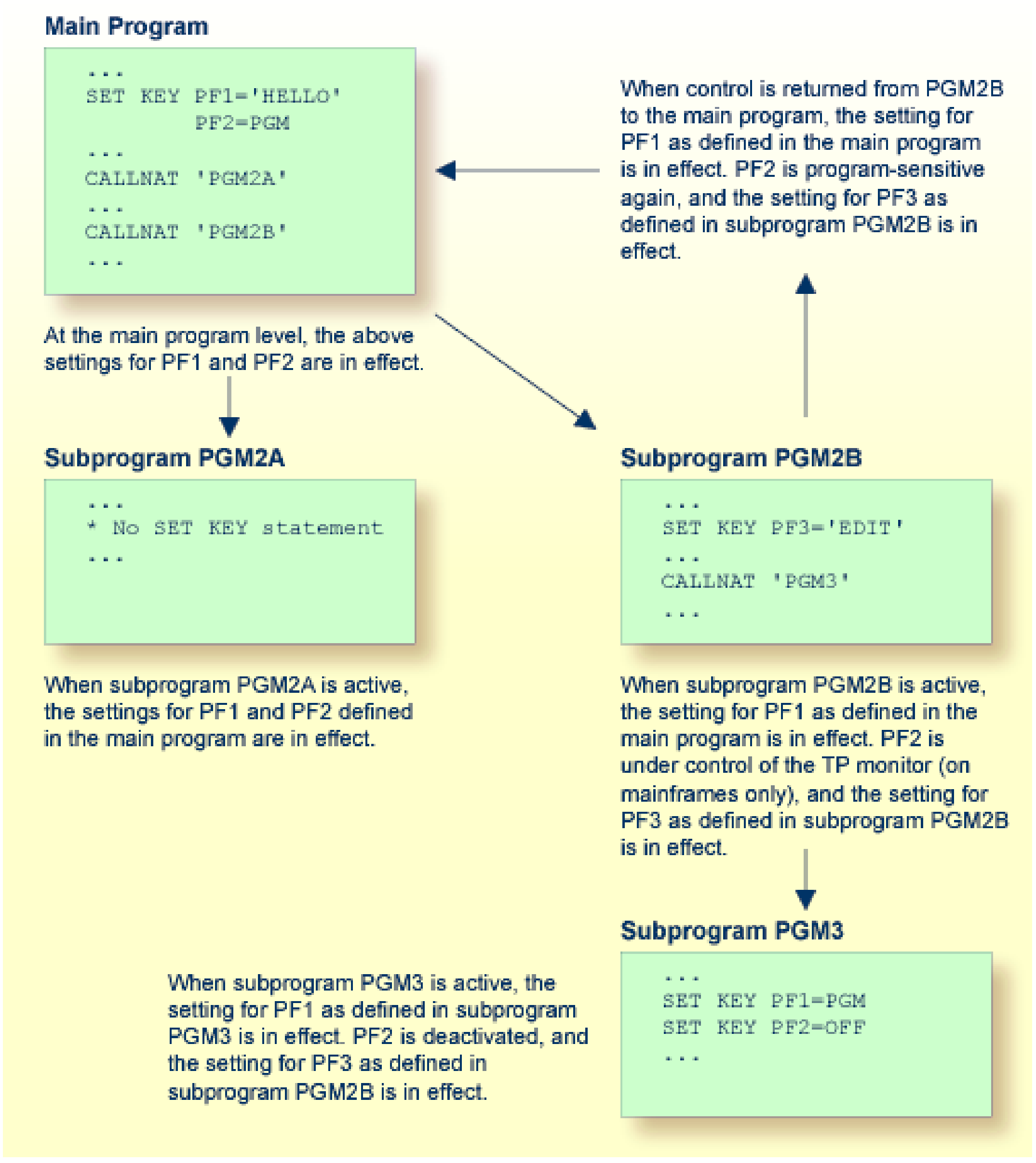
The `DISABLED` option can only be used within a processing rule.

SET KEY Statements on Different Program Levels

When an application contains `SET KEY` statements at different levels, the following applies:

- When keys are made program-sensitive, the program-sensitivity also applies to all lower level (called) programs, unless these programs contain further `SET KEY` statements. When control is returned to a higher level program, the `SET KEY` assignments made at the higher level come into effect again.
- For keys which are defined as `HELP` keys, the same applies as for keys which are program-sensitive.
- When a function (program, command, terminal command, or data string) is assigned to a key, this assignment is valid at all higher and lower levels - regardless of the level at what the assignment is made - until another function is assigned to the key or it is made program-sensitive, or until the user logs on to another application or the Natural session is terminated.

Example of SET KEY Statements on Different Program Levels


Main Program
...
SET KEY PF1='HELLO'
PF2=PGM
...
CALLNAT 'PGM2A'
...
CALLNAT 'PGM2B'
...
At the main program level, the above settings for PF1 and PF2 are in effect.
When control is returned from PGM2B to the main program, the setting for PF1 as defined in the main program is in effect. PF2 is program-sensitive again, and the setting for PF3 as defined in subprogram PGM2B is in effect.
Subprogram PGM2A
...
* No SET KEY statement
...
When subprogram PGM2A is active, the settings for PF1 and PF2 defined in the main program are in effect.
Subprogram PGM2B
...
SET KEY PF3='EDIT'
...
CALLNAT 'PGM3'
...
When subprogram PGM2B is active, the setting for PF1 as defined in the main program is in effect. PF2 is under control of the TP monitor (on mainframes only), and the setting for PF3 as defined in subprogram PGM2B is in effect.
Subprogram PGM3
...
SET KEY PF1=PGM
SET KEY PF2=OFF
...
When subprogram PGM3 is active, the setting for PF1 as defined in subprogram PGM3 is in effect. PF2 is deactivated, and the setting for PF3 as defined in subprogram PGM2B is in effect.

Assigning Names

With the NAMED clause, you can assign a name (*operand4*) to a key. The name will then be displayed in the PF-key lines on the screen; this allows the users to identify the functions assigned to the keys:

```
                   ?   Help
                 .   Exit
                ----  ---------------------------------------------------------
      Code ..: ?   Library ..: *_______
                   Object ...: *_______________________________________
                   DBID .....: 0__     FILENR ...: 0__


  Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Last        Flip                                Canc
```

The display of the PF-key lines is activated with the session parameter `KD` (see the *Parameter Reference*). You can control the way in which the PF-key lines are displayed by using the terminal command `%Y` (see the *Terminal Commands* documentation).

The maximum length of a name to be assigned to a key is 10 characters. In normal tabular PF-key line format (`%YN`), only the first 5 characters are displayed.

When *operand4* is specified as a constant, it must be enclosed within apostrophes (see examples below).

You cannot assign a name to a key without assigning a function to it or making it program-sensitive. To the ENTER key, however, you can only assign a name, but no function.

With `NAMED OFF`, you delete the name assigned to a program-sensitive key.

Examples:

SET KEY ENTR NAMED ’EXEC’	The name `EXEC` is assigned to the ENTER key.
SET KEY PF3 NAMED ’EXIT’	PF3 is made program-sensitive, and the name `EXIT` is assigned to PF3.
SET KEY PF3 NAMED OFF	PF3 is made program-sensitive, and the name that has been assigned to PF3 is deleted.
SET KEY NAMED OFF	All names that have been assigned to any program-sensitive keys are deleted.
SET KEY PF4=’AP1’ NAMED ’APPL1’	The program `AP1` and the name `APPL1` are assigned to PF4.

When you use normal tabular PF-key line format (`%YN`), the following applies:

- If you omit the `NAMED` clause when assigning a command/program to a key, the command/program name will be displayed in the PF-key line; if the command/program name is longer than 5 characters, `CMND` will be displayed.
- If you omit the `NAMED` clause when assigning input data to a key, `DATA` will be displayed in the PF-key line.
- If you assign (with the NAMED clause) a name in Unicode format to a PF-key, the name might not be correctly positioned under the respective headers. This problem, however, may occur only when you are using the *Natural Web I/O Interface* and only for "wide" characters. In this case, the sequential PF-key line format (`%YS` or `%YP`) is recommended.

When you use sequential PF-key line format (`%YS` or `%YP`), only those keys to which names have been assigned will be displayed in the PF-key line; that is, if you omit the `NAMED` clause when assigning a command/program/data to a key, the key will not be displayed in the PF-key line.

See also *Processing Based on Function Key Names* (in the *Programming Guide*).

Example

```
** Example 'SKYEX1': SET KEY
************************************************************************
DEFINE DATA LOCAL
1 #PF4 (A56)
END-DEFINE
*
MOVE 'LIST VIEW' TO #PF4
*
SET KEY PF1 PF2
SET KEY PF3 = 'MENU'
        PF4 = #PF4
        PF5 = 'LIST VIEW EMPLOYEES' NAMED 'Empl'
*
FORMAT KD=ON
INPUT ////
      10X 'The following function keys are assigned:' //
      10X 'PF1: Function for PF1      ' /
      10X 'PF2: Function for PF2      ' /
      10X 'PF3: Return to MENU program' /
      10X 'PF4: LIST VIEW             ' /
      10X 'PF5: LIST VIEW EMPLOYEES   ' ///
*
IF *PF-KEY = 'PF1'
  WRITE 'Function for PF1 executed.'
END-IF
IF *PF-KEY = 'PF2'
  WRITE 'Function for PF2 executed.'
END-IF
*
END
```

Output of Program SKYEX1:

```
The following function keys are assigned:

PF1: Function for PF1
PF2: Function for PF2
PF3: Return to MENU program
PF4: LIST VIEW
PF5: LIST VIEW EMPLOYEES
```