Program Editor

The Natural program editor is used to create and modify the source code of a Natural object of the type program, subprogram, subroutine, helproutine, copycode, text or class.

The *Program Editor* documentation is organized in the following parts:

- Invoking the Program Editor
- Editor Command Line
- Top Information Line
- Editing Area
- Bottom Information Line
- Editor Commands
- Editor Commands for Positioning
- Line Commands
- Special PF-Key Functions
- Cursor-Sensitive Commands
- Saving and Cataloging Sources
- Exit Function

Related Topic:

For information on Unicode and code page support for Natural editors, see *Editors* in the *Unicode and Code Page Support* documentation.

Invoking the Program Editor

To invoke the program editor

- Use the system command `EDIT` as described in the *System Commands* documentation.

  When the program editor is invoked, an editor screen similar to the example below appears:

```
>                                       > +  Program     SAGDEMO  Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** EXAMPLE 'SAGDEMO': DISPLAY
  0020 ************************************************************************
  0030 DEFINE DATA LOCAL
  0040 1 VIEWEMP VIEW OF EMPLOYEES
  0050   2 PERSONNEL-ID
  0060   2 NAME
  0070   2 BIRTH
  0080   2 JOB-TITLE
  0090 END-DEFINE
  0100 *
  0110 READ (3) VIEWEMP BY BIRTH
  0120   DISPLAY PERSONNEL-ID NAME JOB-TITLE
  0130 END-READ
  0140 END
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 14   L 1
```

The editor screen contains the following items (from top to bottom): the editor command line, the top information line, the editing area and the bottom information line. These items are explained in the following sections.

Note:
If Natural ISPF is installed and the editor profile option **ISPF Editor as Program Editor** is set to `Y`, instead of the program editor, either the Natural ISPF main menu (if the `EDIT` command is entered without an object name) or the Natural ISPF editor screen with the specified object is invoked.

Editor Command Line

The editor command line is indicated by the leftmost greater than sign (>) in the top line of the editor screen. In the command line, you can enter one of the following:

- Any Natural system command.

  For example: The system command `CHECK` can be used for checking the syntax of source code and `SAVE` for saving source code (see also *Saving and Cataloging Sources*).

  For other system commands related to maintaining and using object sources, see *Editing and Storing Programming Objects* in the *System Commands* documentation.

- One or more editor commands.

- The name of a Natural program to be executed.

Additionally, the top line can contain the following items (from left to right):

Direction Indicator: + or -	The direction indicator can be set to control the direction of the editor commands ADD and SCAN, and of the line commands .C, .I and .M. The plus sign (+) indicates *after* and the minus sign (-) indicates *before*. The exact interpretation is described with the relevant command description. See also the editor profile option **Direction Indicator** described in *Editor Profile*.
Object Type	The type of object currently in the source work area. If no object type or object name is specified when the program editor is invoked, object type **Program** is displayed by default. The object type can be changed by using the editor command SET TYPE.
Object Name	The name of the object currently in the source work area. No name is displayed if the source work area is empty or if the current source code has not yet been saved as a source object with the SAVE, CATALOG or STOW command.
Modification Indicator: *	An asterisk (*) indicates whether the source code currently in the source work area contains unsaved modifications. The asterisk (*) also appears for new source code that has not yet been saved as a source object. The asterisk (*) is only visible if the editor profile option **Source Status Message** is set to Y (see *Editor Profile*). The asterisk (*) disappears when you execute a successful SAVE or STOW command on the source. See also *Exit Function*.
Lib	The library to which you are currently logged on.

Top Information Line

The top information line of the editor screen is a scale line. It can contain the following:

- A message indicating object modification. This information is only displayed if the editor profile option **Source Status Message** is set to Y (see *Editor Profile*).
- The programming mode (structured or reporting) currently in effect. When a Natural object is read into the source work area, the mode is set to the one which was in effect when the object was saved with the SAVE or STOW command. This information is only displayed if the editor profile option **Source Size Information** is set to Y (see *Editor Profile*).

  For information on the differences between structured and reporting mode, see *Purpose of Programming Modes* in the *Programming Guide*.

Editing Area

The editing area of the editor screen contains the numbered lines where you add or modify source code.

The editing area is either empty or contains source code that was last read into the source work area with the command EDIT or READ as shown in the example of a program in *Invoking the Program Editor*.

When you read in the source of an existing object, the entire source code is loaded into the source work area and is available for editing. However, depending on the size of the source, the editing area may not show all of the lines that belong to the source. In this case, you have to scroll down the source (see *Editor Commands for Positioning*) to go to the line you want to view or modify.

In addition, if you use split-screen mode, the editing area displays fewer lines of source code. See also *Split-Screen Mode*.

To create or edit source code, you can perform multiple functions:

- Type in or update code directly in the relevant source line.
- Use one or more editor commands as described in the relevant section.
- Use one or more line commands as described in the relevant section.

When performing multiple functions, consider the following:

- Only one insert line command (.I) can be performed at a time.
- You can enter multiple commands in the command line of the editor: you can enter more than one editor command, but only the last command entered in the editor command line can be a system command. For example:

```
SC 'MOVE',-2,RENUMBER
```

  Note:
  Natural treats the editor command N like a system command. N corresponds to the system command RENUMBER.

- If you have changed the source code by typing in a modification or by using an editor command, a system command cannot be entered until you press ENTER.

Dynamic Conversion from Lower to Upper Case

You can activate or deactivate dynamic conversion to upper case, by setting the appropriate editor profile options **Editing in Lower Case** and **Dynamic Conversion of Lower Case** to Y (Yes). All source code you enter in the editing area is then converted to upper case, with the following exceptions:

- The contents of a Natural object of the type text remain as entered.
- A text string that is not a hexadecimal constant and is enclosed in apostrophes remains as you enter it.
- A comment indicated by the character string blank-slash-asterisk (/*) remains as you enter it.

Caution:
If the character string slash-asterisk (/*) denotes an executable part of a statement, it must be specified *without* a blank character in front of the string (/*). The string will otherwise be considered a comment.

Bottom Information Line

The bottom information line of the editor screen is a scale line. It can contain the following:

- **Current Source Size**

  The size (number of characters) of the current source. As source lines are stored in variable length in the source work area, trailing blanks within a source line are not counted; leading and embedded blanks are counted. This information is only displayed if the editor profile option **Source Size Information** is set to Y (see *Editor Profile*).

- **Char. Free**

  The number of characters still available in the source work area. This information is only displayed if the editor profile option **Source Size Information** is set to Y (see *Editor Profile*).

- **S**

  The size (number of lines) of the source being edited.

- **L**

  The number of the source line currently displayed as the top line.

Editor Commands

Editor commands are entered in the command line of the program editor. The command parameters must be separated either by the input delimiter character as defined with the Natural session parameter ID (the default delimiter character is comma (,)) or by a blank. When multiple commands are entered, these must also be separated by the delimiter character or by blanks.

The editor commands available are described in the following table and in the section *Editor Commands for Positioning*. For explanations of the syntax symbols used in the editor commands, refer to *System Command Syntax* in the *System Commands* documentation. An underlined portion of a command denotes a valid abbreviation.

Editor Command	Function
ADD[(*n*)]	Adds *n* blank lines. If the direction indicator is set to + (plus sign), the lines are added after the last line of the object being edited; if the direction indicator is set to - (minus sign), the lines are added before the first line of the object. The value for *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 9 lines (4 in split-screen mode) are added by default. With the next ENTER, lines that are still left blank are removed.
CANCEL or . (a period)	Leaves the editor. Any modifications made since the last time the SAVE command was entered are *not* saved.
CATALOG [*object-name*]	Executes the system command CATALOG which checks and catalogs the current source code. You must supply an object name with the command if you catalog new source code or if you want to copy the current source code. See also *Saving and Cataloging Sources*.
CHANGE ['*scan-value*'*replace-value*']	Scans the source code for the character string entered as *scan-value* and replaces each such *scan-value* found with the character string entered as *replace-value*. Each line in which a character string is replaced is marked with an R to the left of the line. Any special character which is not valid within a Natural variable name can be used as the delimiter character. If you enter CHANGE without any parameter, the **SCAN/REPLACE** window is invoked. In addition to the *scan-value* and the *replace-value*, you can specify the following in this window: the null value option (see SET NUL), the absolute scan mode (see SET ABS), and the range mode (see SET RANGE).

Editor Command	Function
CHECK	Executes the system command CHECK which checks the syntax of the current source code. If an error is found, the erroneous line is marked with an E and an appropriate error message appears in the message line. If no errors are found, a message appears indicating successful completion of the check.
CLEAR	Executes the system command CLEAR which clears the source work area (including the object name and the line markers X and Y).
DX or DY	Deletes the X-marked or the Y-marked line. See also the line commands .X and .Y.
DX-Y	Deletes the block of lines delimited by the X and Y markers. See also the line commands .X and .Y.
EX or EY	Deletes lines from the top of the editing area to, but not including, the X-marked line; or from the line following the Y-marked line to the bottom of the editing area. See also the line commands .X and .Y.
EX-Y	Deletes all lines in the editing area excluding the block delimited by X and Y. See also the line commands .X and .Y.
EXIT	Leaves the editor. Any modifications to the source are saved depending on the setting of the editor profile described in *Exit Function*.
LET	Undoes all modifications made to the current screen since the last time ENTER was pressed. In addition, LET ignores all line commands already entered but not yet executed.

Editor Command	Function
N [(*nnnn*)]	This command corresponds to the system command RENUMBER. It renumbers the lines of the source code currently in the source work area. If you only enter N, the lines are numbered in increments of 10; if you enter N (*nnnn*), the lines are renumbered in increments of *nnnn*. If the value specified for *n* is too big, lines are numbered in increments of 5. **Note:** See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.
PROFILE [*name*]	Invokes the **Editor Profile** screen where you can view or change your current editor profile settings. For details, see the section *Editor Profile*.
REN ON|OFF	ON Renumbers the lines of a source whenever the CHECK, RUN, SAVE, CATALOG or STOW command is executed on the source. OFF Indicates that automatic renumbering is not in effect. The default is ON. The REN command corresponds to the editor profile option **Auto Renumber** described in *Editor Profile*. **Note:** See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.
RESET	Deletes the current X and Y line markers and any marker previously set with the line command .N. See also the line commands .X and .Y.

Editor Command	Function
SAVE [*object-name*]	Executes the system command SAVE which saves the current source code. You must supply an object name if you save new source code or if you want to copy the current source code. See also *Saving and Cataloging Sources*.
SCAN [*'scan-value'*]	Scans the source code for a character string (*scan-value*). Each line in which the *scan-value* is found is marked with an S to the left of the line. If the supplied *scan-value* is entered without delimiter characters, for example, SCAN ABC D, the entire character string which follows the keyword SCAN is used as the scan value. **Note:** The SCAN command performs an exact search for the supplied *scan-value*. This should be taken into account when searching for DBCS (Double Byte Character Set) characters. If the direction indicator is set to + (plus sign), the scan is performed from the first line shown on the screen to the last line of the source work area. If the direction indicator is set to - (minus sign), the scan is performed from the last line shown on the screen to the first line of the source work area. If you enter SCAN without any parameter, the **SCAN/REPLACE** window is invoked. In addition to the *scan-value*, you can specify the following in this window: a *replace-value* (see CHANGE), the null value option (see SET NUL), the absolute scan mode (see SET ABS), and the range mode (see SET RANGE). SCAN is a cursor-sensitive command which provides additional options described in *Cursor-Sensitive Commands*.

Editor Command	Function
SCAN=[+|-]	Scans for the next occurrence of *scan-value* specified with the SCAN command. The direction for a given scan command can be explicitly specified by entering SCAN=+ or SCAN=-. The setting of the direction indicator is then ignored. **Note:** The equal sign (=) used with the SCAN command is the default input assign character. If another character has been specified as input assign character (see session parameter IA as described in the *Parameter Reference*), that other character must be used instead.
SET ABS [ON|OFF]	ON The SCAN and CHANGE commands operate in absolute mode, which means that the *scan-value* and the *replace-value* need not be delimited by blanks or special characters. OFF The SCAN and CHANGE commands do not operate in absolute mode, which means that the *scan-value* and the *replace-value* must be delimited by blanks or special characters. The default is OFF. The SET ABS command corresponds to the editor profile option **Absolute Mode for SCAN/CHANGE** described in *Editor Profile*.
SET ESCAPE *character*	The escape character which must precede each line command. The default escape character is the plus sign (.).

Editor Command	Function
SET NUL [ON|OFF]	ON All occurrences of a value scanned with the SCAN command are deleted. After the deletion of the scanned value, the SET NUL command is automatically set to OFF. The default is OFF.
SET RANGE [ON|OFF]	ON The SCAN and CHANGE commands operate in range mode, which means that the value to be scanned/changed must be located within the range of lines delimited by the X and Y line markers. OFF The SCAN and CHANGE commands operate in non-range mode, which means that no range limit is to be in effect. The default is OFF. The SET RANGE command corresponds to the editor profile option **Range Mode for SCAN/CHANGE** described in *Editor Profile*.

Editor Command	Function
SET SEQ [ON|OFF]	OFF If your input is numeric, the first four positions in the editing area are considered as the line number and are moved to the line number position once you press ENTER. This feature is useful, for example, if a statement line is to be referenced by a source code line number in another statement line; when you renumber the source code, the referencing line number is renumbered, too. ON Numeric input in the first four positions remains as entered. Except with object type text, the default is OFF.
SET SIZE [ON|OFF]	ON The size of the source is displayed at the bottom information line of the editor screen and the programming mode is displayed on the scale line. OFF This information is not displayed. The default is OFF. The SET SIZE command corresponds to the editor profile option **Source Size Information** described in *Editor Profile*.

Editor Command	Function
SET STAY [ON|OFF]	ON The current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only. OFF Pressing ENTER positions to the next screen. The default is OFF. The SET STAY command corresponds to the editor profile option **Stay on Current Screen** described in *Editor Profile*.
SET TYPE	This command is used to change the object type (displayed in the editor command line) for the source currently contained in the source work area: SET TYPE PROGRAM SET TYPE SUBROUTINE SET TYPE SUBPROGRAM SET TYPE HELPROUTINE SET TYPE COPYCODE SET TYPE TEXT SET TYPE CLASS The default object type is Program.
SHIFT [-*nn*|+*nn*]	This command shifts each line delimited by the X and Y markers to the left or right. The *nn* parameter represents the number of characters the line is to be shifted. Comment lines are not shifted.
SHIFT --	This command shifts each line delimited by the X and Y markers to the leftmost position. Comment lines are not shifted.
SHIFT ++	This command shifts each line delimited by the X and Y markers to the rightmost position (maximum 99 positions). Comment lines are not shifted.

Editor Command	Function
SPLIT *parameter*	This command splits the editor screen and displays the source of another Natural object in one half of the screen as described in *Split-Screen Mode*. *parameter* represents a parameter that must be specified with the command as described in *Split-Screen Commands* SPLIT is a cursor-sensitive command which provides additional options described in *Cursor-Sensitive Commands*.
STOW [*object-name*]	Executes the system command STOW which saves and catalogs the current source code. You must supply an object name if you STOW new source code or if you want to copy the current source code. Otherwise, an appropriate message appears. See also *Saving and Cataloging Sources*.
STRUCT [DISPLAY]	This command performs structural indentation of source code. The default indentation is in increments of two positions. If DISPLAY is specified, the source code is displayed in compressed form: see the system command STRUCT in the *System Commands* documentation.
*	This command displays the editor command most recently entered.
*=	This command again executes the command most recently entered in the command line.

Editor Commands for Positioning

Editor commands for positioning are entered in the command line of the program editor. The following commands are available for positioning:

Command	Function
ENTER or +P or +	Position forwards one page.
-P or -	Position backwards one page.
+H	Position forwards half a page.
-H	Position backwards half a page.
T or --	Position to top of source.
B or ++	Position to bottom of source.
+*nnnn*	Position forwards *nnnn* lines (maximum 4 digits).
-*nnnn*	Position backwards *nnnn* lines (maximum 4 digits).
nnnn	Position to line number *nnnn*.
X	Position to the line marked with an X.
Y	Position to the line marked with a Y.
POINT	Positions to the line in which the line command .N was entered. See also the line command .P.

Line Commands

Line commands are entered in the first position of a source line. The line commands provided by the program editor are listed below. The notation (*nn*) or (*nnnn*) indicates a repetition factor. The default repetition value is 1 (with the exception of the .I command; see below). For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Note:
You are recommended to enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

Line Command	Function
.C[(*nnnn*)]	Copies the line in which the command was entered.
.CX[(*nnnn*)] or .CY[(*nnnn*)]	Copies the X-marked or the Y-marked line. See also the line commands .X and .Y as well as *Notes for Line Commands*.
.CX-Y[(*nnnn*)]	Copies the block of lines delimited by the X and Y markers. (See also *Notes for Line Commands*.)
.D[(*nnnn*)]	Deletes one or more lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). The default is 1 line.
.I[(*nn*)]	Inserts *nn* empty lines, where *nn* can range from 1 to the total number of lines shown in the editing area minus two. For example, if a total of 28 lines is shown in the editing area, you can insert a maximum of 26 lines. If *nn* is not (or not correctly) specified, 9 lines (4 lines in split-screen mode) are inserted by default. Lines that are left blank are then removed from the source, depending on the setting of the editor profile options **Empty Line Suppression** and **Empty Line Suppression for Text** described in *Editor Profile*. See also *Notes for Line Commands*.
.I(*obj,ssss,nnnn*)	Includes into the source an object contained in the current library or in the steplib (the default steplib is SYSTEM). Depending on the direction indicator, the object is inserted before or after the line in which you enter the command. If you wish to include only part of the object, you specify as *ssss* the first line to be included (for example, 20 means the inclusion will start from the 20th line), and as *nnnn* the number of lines to be included. If you enter multiple commands, this command is always executed after all other line and/or editor commands have been executed. If the object is a map, an INPUT USING MAP statement (see *INPUT Syntax 2 - Using Predefined Map Layout* in the *Statements* documentation) with all defined variables is automatically included in the current line. If the object is a data area, the entire data area is included, except comment lines. Only local and parameter data areas that were saved and cataloged with the STOW command can be included into the source work area; global data areas cannot be included. If the object is an adapter, a PROCESS PAGE USING (see *Syntax 2 - PROCESS PAGE USING* in the *Statements* documentation) with all defined variables is automatically included in the current line.

Line Command	Function
.J	Joins the current line with the next line. If the resulting line exceeds the length of the editor screen line, the line is marked with an L and must be split in two with the .S command (see below) before it can be modified.
.L	Undoes all modifications that have been made to the line since the last time ENTER was pressed.
.MX or .MY	Moves the X-marked or the Y-marked line. See also the line commands .X and .Y as well as *Notes for Line Commands*.
.MX-Y	Moves the block of lines delimited by the X and Y markers (see also *Notes for Line Commands*).
.N	Marks (invisibly) a line to be positioned at the beginning of the source work area by the editor command POINT. The mark is automatically deleted when an error with a line command or editor command occurs, or when the RESET command is executed.
.P	Positions the line marked with this command to the top of the screen.
.S	Splits the line at the position marked with the cursor.
.X	Marks a line with an X (see also *Notes for Line Commands*).
.Y	Marks a line with a Y (see also *Notes for Line Commands*).

Notes for Line Commands:

- If both the commands .X and .Y are applied to one line, it is treated as being marked with an X and with a Y; the line marker actually shown to reflect this status is a Z.
- If the direction indicator is set to + (plus sign), the copied, inserted or moved lines are placed *after* the line in which the corresponding command was entered; if the direction indicator is set to - (minus sign), the copied, inserted or moved lines are placed *before* the line in which the command was entered.

Special PF-Key Functions

The following special functions can also be controlled using PF keys:

Function	Explanation
*CURSOR	A line split function can be combined with the command .I, .CX, .CX-Y, .MX or .MX-Y. This is accomplished by assigning the value *CURSOR to a PF key in the editor profile (see **PF and PA Keys** in *Editor Profile*). If this PF key is then pressed instead of ENTER after a line command has been entered, the line in which the command was entered is first split at the cursor position and then the line command is executed. See also *Example of *CURSOR on PF Key*.
*X or *Y	If a PF key is assigned the value *X or *Y in the editor profile (see **PF and PA Keys** in *Editor Profile*), the cursor position is marked X or Y whenever this PF key is used. These position markers are then used to determine which portion of a line is to be included in the command operation. See also *Example of *X and *Y on PF Keys*.

Example of *CURSOR on PF Key:

The following are instructions for using a PF key to which the value *CURSOR is assigned.

1. In the **PF and PA Keys** section of the **Editor Profile** screen, enter *CURSOR next to PF6.
2. Open the program editor, type in the following text, and press ENTER:

```
>                                      > +  Program              Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 MOVE A TO B
  0020 WRITE A B C
  0030 MOVE C TO B MOVE A TO C
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 3    L 1
```

3. In line 0020, enter the line command .X and press ENTER.

   The line is marked as indicated by the X next to it.

4. In line 0030, enter the line command .CX, place the cursor on the M of the second MOVE, and press PF6.

   The screen then looks similar to the one below:

```
>                                       > +  Program              Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 MOVE A TO B
X 0020 WRITE A B C
  0030 MOVE C TO B
  0020 WRITE A B C
  0030 MOVE A TO C
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0260
       ....+....1....+....2....+....3....+....4....+....5....+... S 5    L 1
```

Line `0030` is split before the cursor position, line `0020` is copied to the line after the line in which you entered the `.CX` command, and the second half of the split line is moved to the last line.

Example of *X and *Y on PF Keys

The following are instructions for using PF keys to which the values `*X` and `*Y` are assigned.

1. In the **PF and PA Keys** section of the **Editor Profile** screen, enter `*X` next to PF4 and `*Y` next to PF5.

2. Open the program editor, type in the following text, and press ENTER:

```
>                                       > +  Text                 Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 THIS PORTION
  0020 OF TEXT IS
  0030 FOR DEMONSTRATION OF
  0040 PF-KEY ASSIGNMENTS.
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 0    L 1
```

3. In line `0010`, place the cursor on the letter `P` and press PF4.

   The position of `P` is marked as shown on the following example screen.

4. In line `0030`, place the cursor on the blank character behind `DEMONSTRATION` and press PF5.

   The screen then looks similar to the one below:

```
>                                      > +  Text                 Lib SAGTEST
All    ....+X...1....+..Y.2....+....3....+....4....+....5....+....6....+....7..
X 0010 THIS PORTION
  0020 OF TEXT IS
Y 0030 FOR DEMONSTRATION OF
  0040 PF-KEY ASSIGNMENTS.
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+X...1....+..Y.2....+....3....+....4....+....5....+... S 4    L 1
```

The positions of the characters (P and blank) are marked as indicated by the X and Y respectively, which appear in the top and bottom information lines and next to the source lines that contain the marked characters.

5. In line 0040, enter the line command .MX-Y and press ENTER.

   The screen then looks similar to the one below:

```
>                                      > +  Text                 Lib SAGTEST
All    X...+....1....+.Y..2....+....3....+....4....+....5....+....6....+....7..
  0010 THIS
  0030 OF
  0040 PF-KEY ASSIGNMENTS.
X 0010 PORTION
  0020 OF TEXT IS
Y 0030 FOR DEMONSTRATION
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0250
       X...+....1....+.Y..2....+....3....+....4....+....5....+... S 6    L 1
```

The block of text starting with P in line 0010 and ending with N in line 0030 is moved to the line below the line in which you entered the command. The moved block of text and the remaining text in line 0010 and 0030 are left-justified.

Cursor-Sensitive Commands

Cursor-sensitive commands are commands where, instead of entering a name in the command line, you can mark the name with the cursor anywhere on the editor screen (except in the command line). You can place the cursor on any word that is not in the command line. It does not matter where on the word the cursor is placed.

The following topics are covered below:

- SCAN Commands
- SPLIT Command
- EDIT and LIST System Commands

SCAN Commands

If the SCAN command is used without any parameter but with the cursor positioned outside the editor command line, this results in a scan operation for the string on which the cursor is placed. (If the cursor is placed on a blank character, however, the **SCAN/REPLACE** window is invoked.)

In split-screen mode, the cursor can be placed on a string in the split-screen area, too.

When using the SPLIT SCAN command, the same applies as for the SCAN command, but the scan operation is performed in the split-screen area only (see also the section *Split-Screen Commands*).

Note:
To benefit from cursor sensitiveness as much as possible, the SCAN or SPLIT SCAN command should be assigned to a PF key (see **PF and PA Keys** in *Editor Profile*).

SPLIT Command

Instead of the commands SPLIT PROGRAM, SPLIT DATA, SPLIT FUNCTION and SPLIT VIEW, which you can use to display a Natural object (including a DDM) in the split-screen area of the editor (see also the section *Split-Screen Commands*), you only have to enter the command SPLIT and place the cursor on the name of the required object. The object must be contained in the current library.

Note:
To benefit from cursor sensitiveness as much as possible, the SPLIT command should be assigned to a PF key (see **PF and PA Keys** in *Editor Profile*).

EDIT and LIST System Commands

The system commands EDIT and LIST are cursor-sensitive, too. Instead of specifying an object name, the cursor can be positioned to a text string of the object currently in the editing area that corresponds to the required object name.

With the EDIT command, the corresponding object is loaded into the editor. If necessary, even a different editor is invoked.

With the LIST command, the corresponding object is listed, even if a view has been referenced.

For more information on EDIT and LIST see the *System Commands* documentation.

Saving and Cataloging Sources

You can save the source code currently in the source work area as a source object and also as a cataloged object, which are stored in a Natural library in a Natural system file.

To save and/or catalog the current source

- Use the system command SAVE, CATALOG or STOW as described in *Saving and Cataloging Objects* in the *Using Natural* documentation.

  Note:
  When you leave the program editor with the EXIT editor command, the current source code is saved automatically if the appropriate editor profile option is set accordingly as described in *Exit Function*.

To keep a copy of the current source

- Use the editor options **Source Save into** and **Auto Save Numbers** as described in *Editor Profile*.

  A copy of the source edited last with any of the Natural editors is then automatically saved as a source object in the current Natural environment.

Exit Function

The effect of the EXIT editor command depends on the setting of the editor profile option **Prompt Window for Exit Function**:

- If set to N, the EXIT command leaves the editor and saves all modifications made to the current source; no prompt window is displayed.
- If set to Y, the EXIT command invokes the **EXIT Function** window whenever you execute the command on a source that contains unsaved modifications (see also Modification Indicator). If no modifications were made to the source, the window does *not* appear and the editor closes without saving the source.

The **EXIT Function** window provides the following options:

Option	Explanation
Save and Exit	Leaves the editor and saves all modifications made to the current source code.
Exit without Saving	Leaves the editor without saving any modifications made to the current source code since it was last saved.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.