Using the Code Frame Editor

A code frame is the basic building block of a model. It provides a rudimentary outline of the code generated by the model. Code frames may contain condition codes to generate blocks of code conditionally. They may also contain subprograms used to generate more complex blocks of code.

This section describes how to access and use the Code Frame editor. The following topics are covered:

- Access the Code Frame Editor
- Features of the Code Frame Editor

Access the Code Frame Editor

There are three methods you can use to access the Code Frame editor. These methods are:

- From the Administration Main Menu
- From the Command Line
- From the Maintain Models Panel

From the Administration Main Menu

To access the Code Frame editor from the Administration main menu:

1. Type "F" in Function.
2. Press Enter.

   The Code Frame menu is displayed. For example:

```
CSMMAIN              N a t u r a l   C o n s t r u c t              CSMMNM0
Jul 05                        Code Frame Menu                        1 of 1

                    Functions
                    ----------------------------------------------
                    E  Edit Code Frame
                    S  Save Code Frame
                    L  List Code Frames
                    P  Purge Code Frame
                    C  Clear Edit Buffer
                    H  Print Saved Code Frame


                    ?  Help
                    .  Return
                    ----------------------------------------------
Function ........... _
Code Frame ......... ________
Description ........ _________________________________________________

Command ............ ____________________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit                                                  main
```

For information about the functions available through this menu, see Code Frame Menu Function.

3. Type "E" in Function.

   Tip:
   To edit an existing code frame, type the name of the code frame in Code Frame before accessing the Code Frame editor.

4. Press Enter.

   The Code Frame editor is displayed. For example:

```
 Code Frame .........                                                SIZE
 Description ........                                                FREE 61361
 >                                          > + ABS X X-Y _ S       L
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.. T C
```

```
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.. T
```

For information about modifying the supplied code frames, see Edit Code Frame.

From the Command Line

You can also access the Code Frame editor from the Natural Next prompt (Direct command box for Unix).

To access the Code Frame editor from the command line:

1. Logon to the SYSCST library.
2. Enter the following command:

```
MENU F E/framename/framedescription
```

From the Maintain Models Panel

You can also access the Code Frame editor from the Maintain Models panel.

To access the Code Frame editor from the Maintain Models panel:

1. Access the Administration main menu.

   For information, see Access the Administration Main Menu.
2. Enter "M" in Function.

   The Maintain Models panel is displayed.

   Note:
   For a description of this panel, see Maintain Models Function.
3. Move the cursor over the code frame you want to edit.
4. Press PF4 (frame).

   The specified code frame is displayed in the Code Frame editor.

   Note:
   For information about editing code frames, see Edit Code Frame.

Features of the Code Frame Editor

The following example shows the CSLC9 code frame in the Code Frame editor:

```
Code Frame ......... CSLC9                                              SIZE 29281
Description ........ Browse-Select* model subroutines                   FREE 29520
>                                              > + ABS X X-Y _ S 408  L 1
Top...+....1....+....2....+....3....+....4....+....5....+....6....+....7.. T C
  *
  * Subroutines (in alphabetical order).
  *
  CHECK-WILD-CHARACTER                                                      1
  ************************************************************************  "
  DEFINE SUBROUTINE CHECK-WILD-CHARACTER                                    "
  ************************************************************************  "
  *                                                                         "
  * Check for wild characters in the input key and                          "
  * reset minimum and maximum values for the key accordingly                "
  RESET #WILD-CHAR #LAST-POS                                                "
  FOR #WINDX = 1 TO 3                                                       "
    EXAMINE #INPUT.#CHAR-ARRAY(*) FOR                                       "
          CDWILDA.#WILD-CARD-CHARS(#WINDX) GIVING INDEX #FIRS-POS(#WINDX)   "
  END-FOR                                                                   "
  /* Find the first wild character                                          "
  FOR #WINDX = 1 TO 2                                                       "
     IF #FIRS-POS(#WINDX) = 1 THRU #FIRS-POS(#WINDX + 1) OR                 "
  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.. T
```

The Code Frame editor supports all generic Natural edit commands except the RUN, CHECK, TEST, STOW, and SAVE commands. This editor has no line numbers, but it does have two extra fields to the right of the edit area: T (Type) and C (Condition). Natural Construct uses these fields to control the generation process for each code frame.

The fields in the Code Frame editor are

Field	Description
Code Frame	Name of the code frame currently in the editor (the name specified in Code Frame on the Code Frame menu).
Description	Brief description of the code frame.
SIZE	Size of the code frame (in bytes).
FREE	Number of bytes currently available in the editor.
>	Command line prompt, at which you can: • Enter "Q", "QUIT", or "." to close the editor. • Issue an edit command (for a list of the edit commands, see Edit Commands.
+	Direction indicator. The plus sign (+) indicates that the ADD, MOVE, COPY, INSERT, and SCAN commands operate in a forward (from top to bottom) direction. To have the commands operate in a backward direction (from bottom to top), type a minus sign (-) over the plus sign. Edit commands use the direction indicator to determine whether to place lines before the first line in the editor or after the last line. For example, using the ADD edit command and a + indicator adds lines after the last line in the editor; using the ADD edit command and a - direction indicator adds lines before the first line in the editor.

Field	Description
ABS	Absolute field, which is used in conjunction with the SCAN and CHANGE edit commands. When this field is marked, the system scans for or changes the specified characters, including those within words. If you specify a blank in this field, the system scans for or changes the specified characters only if they are a separate entity (delimited by blanks or special characters).
X-Y	X and Y delimiters for a block of code. To confine SCAN and CHANGE commands to code within an X-Y delimited range, mark this field. Code outside the X-Y range is not affected.
S	Total number of lines of code currently in the editor.
L	Number of the first line currently displayed in the editor.

Field	Description
T	Editor line type. Valid line types are: • N Indicates that this is a subprogram line and the specified Natural subprogram is invoked during generation. If you specify "N", the line is automatically formatted as follows: `Subprogram: _______  Parameter: ______  N` Type the name of the subprogram in Subprogram. If the subprogram is invoked more than once or in multiple code frames, you can specify a constant in Parameter (the constant is placed in the #PDA-FRAME-PARM field in the CU—PDA parameter data area). The subprogram can test this field to determine where the subprogram is invoked. • F Indicates that this is a secondary (nested) code frame line and the specified code frame is invoked during generation. The names of nested code frames should all end with a question mark (?). This naming convention greatly reduces the time and effort required to modify code frames. • U Indicates insertion points where developers can insert user exit code. (You can specify additional attributes using the .E command after the line is specified.) • * Indicates code frame comments, which are not used by the generated module. • B Indicates that blank lines are valid and will be generated into the source area. This line type is used to explicitly hold blank line positions. Natural Construct will not change the contents of any B type line. If text is entered on a B type line, the text is generated; if a B type line is blank, a blank line is generated. **Note:** Natural code does not require blank lines, whereas many scripting languages use the blank line concept extensively. • X Indicates that the text portion of the line must contain the name of a user exit, and the code in the C field must be a number from 1 to 9. If the user exit exists in the User Exit editor when the program is generated, this line indicates that the condition is True. • blank Indicates that this line is constant text and is inserted directly in the generated program, based on the value in C. Whenever a code frame is updated, Natural Construct compresses blank lines and lines marked with B.

Field	Description
C	Condition level of the corresponding lines. Valid levels are: • *n* (1–9) Indicates a new condition for this level. The conditions are Boolean combinations of the condition constants specified for the generator. If the condition specified on the line is True, all subsequent code with quotation marks (") is included in the generated program. You can nest conditions by specifying a number greater than 1. (For information about setting up conditions for your generators, see Use Code Frame Conditions.) • " Indicates that text on this line is a continuation of the previous block of code and subject to the last condition specified. • blank Indicates that the corresponding line is constant text and is included unconditionally.

This section covers the following topics:

- Use Commands in the Code Frame Editor
- Change the PF-Key Profile for the Current Session
- Save the Contents of the Edit Buffer
- Create GUI Sample Subprograms

Use Commands in the Code Frame Editor

This section describes how to use commands in the Code Frame editor. The following topics are covered:

- Order of Command Execution
- Line Commands
- Edit Commands
- Positional Edit Commands

Order of Command Execution

The Code Frame editor executes commands in the following order:

1. Processes text modifications.
2. Executes line commands.

   These commands are specified in the text area of the editor and are preceded with a period (.E, for example).
3. Executes edit commands.

These commands are specified at the > prompt (ADD, for example).

Line Commands

Within the Code Frame editor, you can issue line commands to copy, move, and delete lines of code. Line commands must be entered in the first column position of a line in the edit area (not at the > prompt) and must begin with a period (.).

Note:
Except for the .L command, you should only issue line commands on modified code after you press Enter.

If the direction indicator is + (indicating from top to bottom), the copied, moved, or inserted lines are placed below the line on which the command is entered. If the direction indicator is - (indicating from bottom to top), the lines are placed above the line on which the command is entered.

Note:
To avoid shifting the T (Type) and C (Condition) fields, the SHIFT, .J, and .S commands are not available in the Code Frame editor.

The line commands applicable in the Code Frame editor are:

Command	Function
.C(*nn*)	Copies the current line *nn* times, where *nn* is the number of times. The default is one time.
.CX(*nn*)	Copies the line marked X *nn* times, where *nn* is the number of times. The default is one time.
.CY(*nn*)	Copies the line marked Y *nn* times, where *nn* is the number of times. The default is one time.
.CX-Y(*nn*)	Copies the block delimited by X and Y *nn* times, where *nn* is the number of times. The default is one time.
.D(*nn*)	Deletes *nn* lines, where *nn* is the number of lines. The default is one line.
.E	Specifies additional attributes for user exits. If the corresponding line is type U (user exit point), you can specify additional attributes for the user exit by issuing the .E command.
.G(*model*, *parameters*)	Invokes the Natural Construct Generation subsystem.
.I(*nn*)	Inserts *nn* lines, where *nn* is the number of lines. The default is 9 lines; the maximum is 9 lines. The Code Frame editor suppresses unused lines unless they are marked with a B line type.
.IF (*code frame name*)	Inserts the specified code frame on the line below the line on which the command is specified. **Note:** The direction indicator has no effect on this command.

Command	Function
.I(*member,startline,number of lines*)	Places a member from the current library onto a specified line in the editor. You can also specify a starting line and the total number of lines to include.
.L	Restores the line on which the command is specified to its previous state. (This command is similar to the LET edit command, except it applies to one line only.)
.MX	If the direction indicator is +, this command moves the line marked X to the line below the one on which .MX is specified. If the indicator is -, this command moves the line marked X to the line above.
.MY	If the direction indicator is +, this command moves the line marked with Y to the line below the one on which .MY is specified. If the direction indicator is -, this command moves the line marked Y to the line above.
.MX-Y	Moves the block of lines delimited by the X and Y markers. If the direction indicator is +, this command moves the block to the line below the one on which .MX-Y is specified. If the direction indicator is -, this command moves the block to the line above.
.N	Marks the line for the POINT edit command (for information on the POINT command, see Positional Edit Commands).
.P	Moves the line on which the command is specified to the top of the panel.
.W(*nn*)	Inserts *nn* blank lines in the editor, where *nn* is the number of lines. The default is 9 lines. Whenever the code frame is updated, Natural Construct suppresses any unused lines unless they are marked as B line types.
.X	Marks a line, or marks the beginning of a block of lines, that ends with a line marked Y.
.Y	Marks a line, or marks the end of a block of lines, that begins with a line marked X.

Edit Commands

Edit commands are specified at the command prompt (>). These commands are:

Command	Function
ADD	Adds 9 blank lines to the editor.

Command	Function
CHANGE	Scans for text and replaces it with the specified value. The syntax is: `CHANGE 'scanvalue'replacevalue'` You can use any special character as a delimiter, as long as you do not use the same character within the command. **Note:** Unless X and Y line commands limit the range, this edit command performs changes to the entire edit buffer.
CLEAR	Clears the current contents of the edit buffer.
DX	Deletes the line marked X.
DY	Deletes the line marked Y.
DX-Y	Deletes the lines between the X and Y markers, inclusively.
END	Ends the edit session and invokes the previous menu.
EX	Deletes all lines before the X marker.
EY	Deletes all lines after the Y marker.
EX-Y	Deletes all the lines before the X marker and after the Y marker.
HELP	Displays help text for the Code Frame editor.
LET	Restores lines to their previous state, should you inadvertently change them. Specify the command before pressing Enter. (This command is similar to the .L line command, but applies to the entire buffer.)
LIST	Lists the current contents of the Main buffer.
PROFILE	Invokes a window in which you can modify PF-key settings and edit specifications for the current edit session (see Change the PF-Key Profile for the Current Session).
QUIT or .	Ends the edit session and invokes the previous menu.
READ *program*	Reads the Natural source for *program* into the edit buffer.
RESET	Clears the X and Y markers.

Command	Function

SCAN	Scans for data in the edit area in the following ways:
	
	`SCAN 'scanvalue`
	
	Scans for text within the delimiters.
	
	`SCAN scan value`
	
	Scans for the entire text after the SCAN keyword, including spaces.
	
	Note:
	You must use delimiters for scan values that begin with a non-alphanumeric character.
	
	If the direction indicator is "+", the scan begins at the first line displayed on the panel and continues to the end of the text. If the indicator is "-", the scan begins at the last line and continues to the beginning. When the scan value is found, "S" is displayed in the left column next to the target line(s).
	
	Note:
	You can also limit the scan range by marking the X-Y field at the top of the Code Frame editor. For a description of this field, see Features of the Code Frame Editor.

UPPER	Invokes a window in which you can specify one or more of the following translation options: • Comments Translates all lower case text in comments (text preceded by *, **, or /*). • Statements Translates all lower case text in statements, including variables. • Quoted strings Translates all lower case text in quoted strings. • Programming Translates text for the programming language specified.
*	Redisplays the last command issued.

Positional Edit Commands

If the code frame in the edit buffer is too large to be displayed in its entirety on the panel, you can issue edit commands at the command prompt (>) to scroll through the code:

Command	Function
+*nnnn* or -*nnnn*	Scrolls forward (+) or backward (-) *nnnn* lines.
+H or -H	Scrolls forward (+) or backward (-) half a panel.
+P or -P	Scrolls forward (+) or backward (-) one panel. **Note:** If the code was not changed, you can press Enter to scroll forward one panel.
BOTTOM or ++	Scrolls forward to end of code frame.
POINT	Scrolls line on which the .N line command is specified to top of panel.
TOP or –	Scrolls backward to top of panel.
X or Y	Scrolls to the line marked X or Y.
nnnn	Scrolls to the *nnnn* line.

Change the PF-Key Profile for the Current Session

You can change the PF- and PA-key settings, the number of updates before an automatic save, and the name of the recovery member. Any changes to the current profile take effect immediately and remain in effect for the duration of the current edit session. These changes do not affect the Natural edit profile.

To change the PF-key profile for the current session:

1. Enter "PROFILE" at the > prompt in the Code Frame editor.

   The Maintain Current PF-Key Profile window is displayed. For example:

```
CS-PROF                  Natural Construct                    CS-PRFM0
Jun 20             Maintain Current PF-Key Profile              1 of 1

 PF1 =  -_______________ PF2 =  T_______________ PF3 =  B_______________
 PF4 =  -H______________ PF5 =  +H______________ PF6 =  +P______________
 PF7 =  N_______________ PF8 =  ________________ PF9 =  Q_______________
 PF10=  ________________ PF11=  ________________ PF12=  ________________
 PF13=  ________________ PF14=  ________________ PF15=  ________________
 PF16=  ________________ PF17=  ________________ PF18=  ________________
 PF19=  ________________ PF20=  ________________ PF21=  ________________
 PF22=  ________________ PF23=  ________________ PF24=  ________________
 PA1 =  ________________ PA2 =  SCAN____________ PA3 =  ________________

 Auto save numbers .............      In member .......... EDITWORK
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11-
      help  retrn
Changes DO NOT affect your edit profile outside Construct
```

   This window displays the various settings in effect for the current edit session. The PF-key settings for the Natural Construct editors are determined in the same manner as those for the Natural editor. If you have a profile that corresponds to your user ID, Natural Construct will use those defaults.

2. Change the settings as desired.

   The fields in this window are:

Field	Description
PF-*nn* or PA-*n*	Functions assigned to the PF- and PA- keys. You can add new functions by typing a command next to the desired key, or modify existing functions by typing a new command over the one displayed.
Auto save numbers	Number of updates allowed before the source is automatically saved. If this field is blank or 0 (zero), Natural Construct does not automatically save work.
In member	Name of the program that is overwritten each time the specified number of updates is exceeded (by default, EDITWORK). To change the name of the program, type a new name over the one displayed. If this field is blank, Natural Construct does not automatically save work.

Save the Contents of the Edit Buffer

The Natural Construct editors can automatically save work in the edit buffer after a certain number of updates. The number specified in Auto save numbers in the Maintain Current PF-Key Profile window determines how often the work is saved. If this field is blank, Natural Construct does not automatically save work. You can also use In member in the Maintain Current PF-Key Profile window to specify the name of the recovery member where you want your work saved.

To recover edits, the value in Auto save numbers must not be blank or 0 (zero) and the value in In member must be specified. For information, see Change the PF-Key Profile for the Current Session.

Tip:
Save your work using a unique recovery member name, such as your user ID. This way, your work will not be overwritten by another user using the same recovery member name in the same library.

To retrieve lost code:

1. Access the Code Frame editor.

   For information, see Access the Code Frame Editor.

2. Read EDITWORK into the edit buffer (or whatever name you specified as your recovery member name in the Maintain Current PF-Key Profile window).

3. Re-specify the description, as it is not saved in the recovery member.

Create GUI Sample Subprograms

Sample subprograms are invoked from a user exit. These subprograms help the developer create user exit code by providing a starting sample. The GUI sample subprogram is a client version of the mainframe sample subprogram — minus the input statements. When Natural Construct generates a model on the client, it bypasses the mainframe sample subprogram and reads the GUI sample subprogram instead.