Chapter

6

The Text Editor

The Text Editor is the graphical tool that is used to create, edit, display and print text file documents.

The Text Editor can also be used as a state matrix viewer. One or several state-signal matrices can be presented for each process in an SDL system. The state matrices for an SDL system are presented as one read-only text file. As in any other SDL viewer tool, it is possible to navigate from an entity in a state matrix to the corresponding entity in the SDL system.

This chapter contains a reference manual to the Text Editor; the functionality it provides, its menus and windows.

Text Files

The Text Editor works with ASCII text files. The text editor can be used to view, edit and print text files. In addition to providing typical text editor functions, the Text Editor provides special support for embedding link endpoints into the text. For more information on links and endpoints, see chapter 10, *Implinks and Endpoints*.

Note:

The Text Editor is only available if the Organizer is started with the preference SDT*TextEditor set to “SDT”.

Endpoint Support

In the Text Editor, endpoints are user-defined, non-overlapping, contiguous ranges of characters, possibly spanning several lines of text, that are made visible to other tools, allowing you to define links between fragments in your text and endpoints in other documents.

Endpoints are displayed in the text editor by underlining the range of characters that are part of the endpoint.

Note that when you define an endpoint, you do not impose any particular restrictions on the text in the endpoint, meaning that you can still edit the text in the endpoint using the normal text editing functions to modify the endpoint’s contents. The text editor dynamically notifies other tools the changes you make to your endpoints as you make your changes, to make sure that the view of the endpoints in your file is always current.

Considerations when Using Text Endpoints

The requirement that endpoints must be non-overlapping means that you cannot create an endpoint inside another endpoint. Also, while an endpoint is allowed to span several lines, it is generally best to restrict endpoints to small, well-defined fragments of text on a single line, used as external reference points to the information in your files.

Also note that while the Text Editor visualizes endpoints by simply underlining the text in the endpoints, the endpoints are stored in your actual text files using special keywords before and after the characters in the endpoint. This means that it is not advisable to view or edit text documents with endpoints in a text editor that is not aware of the special significance of these keywords. If you do so, you should be careful so as to not destroy information about the endpoints in the file, as this could cause links attached to these endpoints to break.

Another consequence of how the Text Editor stores the endpoint information is that if you are editing text documents that are intended as input to other tools (such as a C or C++ compiler) you should take care to place endpoints only within commented regions of text. This makes sure that the extra text inserted when storing the endpoints does not conflict with the allowed grammar of the type of file you are editing.

Examples

In the examples below you will find typical placements of text endpoints. These endpoints could then be the basis for creation of links into requirements, analysis, design, and specification documents, to name only a few possibilities.

Example 20 ––

Improperly commented C header with endpoints:

```
/* Primary protocol packet */
struct Packet { ... };
```

Properly commented C header with endpoints:

```
/* Primary protocol packet */
struct Packet { ... };
```

Example 21 ––

Improperly commented C++ header with endpoints:

```
class Base {

virtual char *name() const = 0;
        virtual int size() const = 0;
                        ...
}
```

Properly commented C++ header with endpoints:

```
// class Base
class Base {

virtual char *name() const = 0;  // get name
        virtual int size() const = 0;    // get size
                        ...
}
```

––

State Overview Files and State Matrices

The Organizer can produce a state overview file, see “State Overview” on page 129 in chapter 2, *The Organizer*. The Text Editor recognizes a state overview file by the extension `.ins`. If such a file is opened in the Text Editor, the Text Editor extracts information from the file and presents the information as a read-only text file containing state matrices. State overview files cannot be edited or saved in the Text Editor.

A state matrix is a textual table with the states in a process or procedure along one dimension and the signals in the same process or procedure along the other dimension. In the table, you will find the transitions for the process or procedure.

A state matrix may look like this: (“-” means no signal)

```
Process Main: nextstates

States
a Start state
b Game_Off
c Game_On
            |  STATES
SIGNALS     |  a    b    c
            |-------------
-           |  b
Newgame     |       c
Endgame     |            b
```

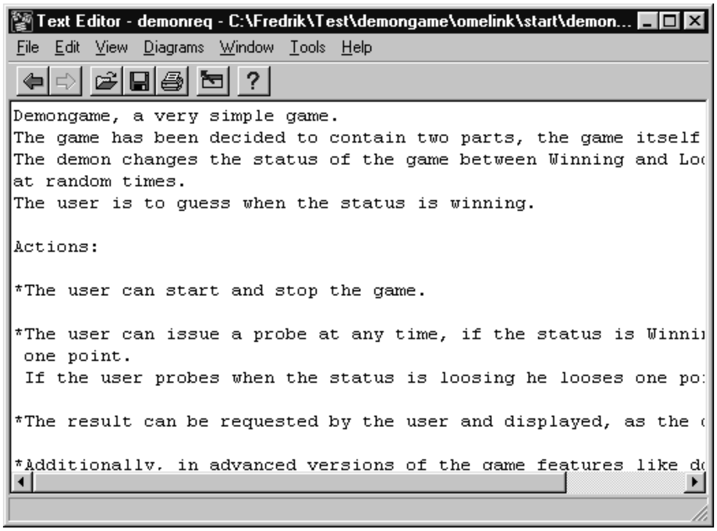
The Text Editor keeps track of how entities in the state matrix is related to the SDL system. This makes it possible to navigate back to the SDL system. To do so, place the text cursor close to an entity in a state matrix and bring up the pop-up menu for that entity and invoke one of the available menu choices.

A state overview file can be saved as a normal text file. This might be useful if you want to edit a state matrix. But by doing so, you will lose all the extra information associated with the state matrices that is not visible when viewing a state overview file in the text editor. You will no longer be able to:

- Navigate back to SDL.
- Get paper page references to SDL pages for transitions in the state matrix (read more about how this is done in “Advanced Print Facilities” on page 348 in chapter 5, *Printing Documents and Diagrams*).
- Decide the kind of information that should be presented for a transition in the state matrix.

Text Editor User Interface

The Text Editor window is depicted below. For a general description of the user interface, see chapter 1, *User Interface and Basic Operations*.


Text Editor - demonreq - C:\Fredrik\Test\demongame\omelink\start\demon...
File Edit View Diagrams Window Tools Help

```
Demongame, a very simple game.
The game has been decided to contain two parts, the game itself
The demon changes the status of the game between Winning and Lo
at random times.
The user is to guess when the status is winning.

Actions:

*The user can start and stop the game.

*The user can issue a probe at any time, if the status is Winni
 one point.
 If the user probes when the status is loosing he looses one po

*The result can be requested by the user and displayed, as the 

*Additionally, in advanced versions of the game features like d
```

Figure 121: The Text Editor window

Text Area

Text in the Text Editor is displayed and printed using the non-proportional font face *Courier*, which is suitable for displaying ASCII texts such as source code.

While the screen font is fixed, you can change the font face used use for printing by editing the SDT preference SDT*PrintFontFamily.

Text Editing Functions

The following text editing functions are provided in the text window:

Text Selection

To select text you can either:

- Drag the pointer over the text.
- Place the cursor in the beginning or end of the text to be selected and then `<Shift>`-click where you want the selection to end.

Word Selection

Double-click a single word, delimited by spaces, to select it.

Line Selection

Triple-click a line of text to select it.

Text Editing

When you edit text, you can:

- Position the cursor by clicking or by using the arrow keys.
- Insert characters after the current position of the text pointer.
- Delete one character backward by pressing `<backspace>`.
- Delete one character forward by pressing `<delete>`.
- Replace selected text by typing.
- Delete selected text by pressing `<backspace>` or `<delete>`.

Undoing Text Modifications

The *Undo* command reverts the previous editing command. Several undo operations can be performed in sequence to undo sequences of editing commands.

Undo can negate the effects of all commands that alter the text in the Text Editor; however, the effect of user operations on endpoints and links cannot be undone by the undo command.

Depending on the nature of the last performed operation, the effects of the last text editing operation is undone, on a per editing operation. For undo purposes, the text editor recognizes the following operations:

- Typing a character
- Inserting a character or a fragment of text
- Deleting a character or a fragment of text

While having to undo each typed character individually may be somewhat inconvenient, a keyboard accelerator (`<Ctrl+Z>`, see “Keyboard Accelerators” on page 398) can speed up the process.

Note that since the text editor supports unlimited undo, it is not possible to use the undo operation to undo the last undo operation. Instead there is a keyboard accelerator (`<Ctrl+Y>`, see “Keyboard Accelerators” on page 398) which can be used repeatedly to undo the effects of a sequence of undo operations.

Editing Text Containing Endpoints

It is possible to edit text using normal operations even after endpoints have been embedded in your text. To unambiguously determine the effects of editing operations on endpoints, the Text Editor uses the character just before the position of your insertion cursor or your selection (the initial character) to determine if your editing affects an endpoint:

- If the initial character belongs to an endpoint (e.g. the insertion cursor is positioned just after an endpoint), inserted text will become part of the endpoint.

- If the initial character belongs to non-endpoint text (e.g. the insertion cursor is positioned just before an endpoint), the inserted text will not be included in an endpoint.

Note:

When deleting or overwriting text, you should be careful not to delete or overwrite all characters in an endpoint since this will remove the endpoint and the associated links.

Dealing with Consecutive Endpoints

When working with consecutive endpoints (i.e. two endpoints follow each other without any intervening non-endpoint text), it is impossible to insert extra characters between these endpoints using the normal text editing operations, since any text inserted between the two endpoints will be considered a part of the first endpoint.

If this particular case arises, you should use *Clear Endpoint* (see "Link > Clear Endpoint" on page 447 in chapter 10, *Implinks and Endpoints*) to remove some characters from one of the endpoints so that some non-endpoint characters become available between your endpoints.

Also note that while the text editor has no difficulty in dealing with consecutive endpoints, visual limitations will make it difficult to distinguish between the two endpoints. You should therefore avoid using consecutive endpoints if possible.

Menu Bar

The menu bar provides commands available in menus and menu choices for editing diagrams and pages of diagrams. Most the functionality that the Text Editor offers is contained within the commands from the menu bar. The Text Editor's menu bar provides the following menus:

- *File Menu*
- *Edit Menu*
- *View Menu*
- *Diagrams Menu*
- *Window Menu*
- *Tools Menu*
- *Help Menu*
  (see "Help Menu" on page 15 in chapter 1, *User Interface and Basic Operations*)

File Menu

The *File* menu contains the following menu choices:

- *New*
- *Open*
- *Save*
- *Save As*
- *Save a Copy As*
- *Save All*
- *Close Diagram*
- *Revert Diagram*
- *Print*
- *Exit*

The menu choices are described in "File Menu" on page 8 in chapter 1, *User Interface and Basic Operations*, except *Print* which is described in "The Print Dialogs in the SDL Suite and in the Organizer" on page 308 in chapter 5, *Printing Documents and Diagrams*.

Edit Menu

The *Edit* menu provides the following choices, which may or may not be dimmed depending upon whether or not some text is selected:

- *Undo*
- *Cut*
- *Copy*
- *Paste*
- *Paste As*
- *Clear*
- *Select All*

Undo

This command restores the last text editing operation. The following operations can be undone:

- *Typing, insertions and deletions.*
- *Cut*, *Paste*, *Paste As* and *Clear*

Note:

Undo does not restore endpoints or links, even though typing, insertions and deletions can destroy both endpoints and links

For more information on the operation of the *Undo* command, see "Undoing Text Modifications" on page 383.

Cut

Cut removes the selected portion of text from the text window and saves it in the clipboard buffer just as if a *Copy* has been made. *Cut* is only available if a portion of text has been selected. *Cut* is not available for state overview files (`*.ins`), because they are not editable.

Also see "Deleting an Object" on page 461.

Copy

Copy makes a copy of the selected portion of text in the clipboard buffer. The content of the text window is not affected. *Copy* is only valid if a portion of text has been selected.

When copying text to the clipboard buffer it becomes possible to paste into three different contexts:

- As simple text (without endpoint and link information) into applications that normally support cut/copy of text, such as terminal windows and other text editors.
- As text with endpoint and link information when pasting into one of the Text Editor windows, which means that endpoints and links are preserved.
- As a Paste As operation in all the Editors (including the Text Editor itself), provided that the copied text contained no endpoints or exactly matched an endpoint. See "Pasting a Text Fragment" on page 460 for more information.

Paste

Paste inserts the content of the clipboard buffer into the text window. The text in the clipboard buffer will be appended immediately following the cursor position, or replacing the selected text. *Paste* is only available if a portion of text has been cut or copied into clipboard buffer. *Paste* is not available for state overview files.

Also see "Pasting an Object" on page 461.

Paste As

Pastes the currently copied object (from the OM or Text Editor) as a text in the text window. The object is transformed and a link is optionally created between the copied object and the pasted text.

The Paste As dialog is opened. See "The Paste As Command" on page 448 for more information. *Paste As* is not available for state overview files (`*.ins`), because these files are not editable.

Clear

Clear removes the selected portion of text from the text window. The content of the clipboard buffer is not affected. *Clear* is only available if a portion of text has been selected. *Clear* is not available for state overview files (`*.ins`), because these files are not editable.

Also see "Deleting an Object" on page 461.

Select All

Select All selects all of the text contained in the text window.

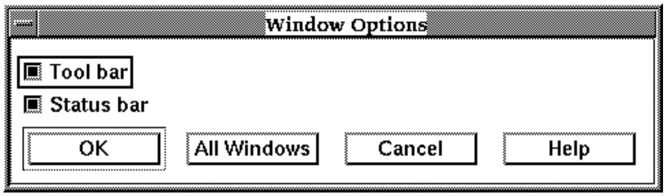
View Menu

The *View* menu provides rescaling functions and access to various options that affect the behavior of the Text Editor. The following menu choices are available:

- *Window Options*
- *Editor Options*
- *State Matrix > Filter Processes*
- *State Matrix > Matrix Options*

Window Options

This menu choice issues a dialog where the presence of the tool and status bars can be set:



[✓ Tool bar]
[✓ Status bar]

Figure 122: The Window Options dialog

- Clicking *OK* applies the options as defined in the dialog to the current window only.
- Clicking *All Windows* applies the options as defined in the dialog to all windows opened by the Text Editor.

Editor Options

This menu choice issues a dialog where the behavior of the Text Editor can be customized.

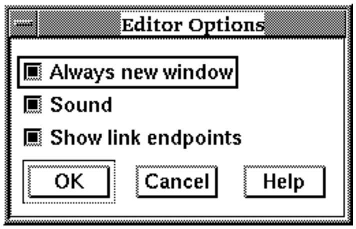


Figure 123: The Editor Options dialog

The options are controlled by toggle buttons. They are:

- *Always new Window*

  This option indicates whether or not a new window should be opened whenever the *New* or *Open* command or any command from the *File* menu is operated.

  The default behavior is not to open a new window.

- *Sound*

  This option indicates whether or not improper actions in the Text Editor, such as attempting to overlap symbols, should be brought to the user's attention by producing an alert sound.

  The default value for this option is on.

- *Show Link Endpoints*

  This option indicates whether link endpoints should be displayed by underlining the endpoint text.

Note:

Regardless of the setting of this option, link endpoints are never shown when printed.

The default value for this option is controlled by the TE*ShowLinks preference.

State Matrix > Filter Processes

This menu choice is only available for state overview files (`*.ins`).

This menu choice brings up a dialog that decides the group of processes that should be visible as state matrices. As default, all processes in the SDL system are visible. The dialog allows you to show selected processes, hide selected processes or show all processes.

State Matrix > Matrix Options

This menu choice is only available for state overview files (`*.ins`).

This menu choice brings up a dialog with options for state matrices. The available options are listed below.

Sort states

The states in the state matrices will be sorted in alphabetical order.

Sort signals

The signals in the state matrices will be sorted in alphabetical order.

Show page reference matrix

The page references refer to the printed page where the transition in question starts, i.e. where the SDL input symbol is placed. ‘-’ means no signal.

Page references are normally not visible, they are replaced with ‘*’. The only chance you have to see page references instead of ‘*’ is to print the SDL process together with the state overview file (`*.ins`) from the Organizer.

```
Page reference matrix for process Demon:

States
a Start state
b Generate

        |  STATES
SIGNALS |  a    b
        |--------
-       |  *
T       |       *
```

Show nextstate matrix

For each transition, the possible nextstates are listed. '-' means no signal.

```
Nextstate matrix for process Demon:

States
a Start state
b Generate

        |  STATES
SIGNALS |  a    b
        |--------
-       |  b
T       |       b
```

Show signal sending matrix

For each transition, signals that might be sent from SDL output symbols during the transition are listed. '-' means no signal or no output.

```
Signal sending matrix for Process Demon:

States
a Start state
b Generate

        |  STATES
SIGNALS |  a    b
        |--------
-       |  -
T       |       1

1 = Bump
```

Show procedure call matrix

For each transition, procedures that might be called during the transition are listed. '-' means no signal or no procedure call.

```
Process Demon: procedure calls

States
a Start state
b Generate

        |  STATES
SIGNALS |  a    b
        |--------
-       |  -
T       |       -
```

Diagrams Menu

The *Diagrams* menu records all diagrams and pages that are opened by the Text Editor. The available menu choices are:

- *Back*
- *Forward*
- *<Diagram Name>*
- *List All*

Back

Select this menu choice to browse back to the document that was previously displayed in the window.

Forward

Select this menu choice to browse forward to the document that was displayed in the window before you selected *Back*.

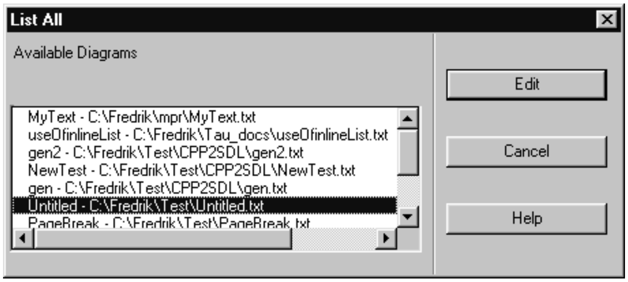
<Diagram Name>

The last edited page always goes to the top of the list, and subsequently moves the other diagrams and pages down a position. A maximum of 9 open pages can be shown. A tenth one will be put at the top of the list, but any subsequent opening of a diagram or page will only show the last 9 that have been opened. Another option – *List All* (at the bottom of the list) – is available to list all the open diagrams in the Text Editor.

Each item in the menu provides information about the diagram name, possibly followed by the file it is stored on (the file information is missing if the diagram has never been saved). A diagram that is preceded by an asterisk (‘*’) denotes that it has been modified during the Text Editor session.

List All

This menu choice becomes available when a maximum of 9 open pages has been surpassed. When *List All* is selected, it provides a dialog containing all diagrams and pages that are currently open in the Text Editor:



MyText - C:\Fredrik\mpr\MyText.txt
useOfinlineList - C:\Fredrik\Tau_docs\useOfinlineList.txt
gen2 - C:\Fredrik\Test\CPP2SDL\gen2.txt
NewTest - C:\Fredrik\Test\CPP2SDL\NewTest.txt
gen - C:\Fredrik\Test\CPP2SDL\gen.txt
Untitled - C:\Fredrik\Test\Untitled.txt
PageBreak - C:\Fredrik\Test\PageBreak.txt

Figure 124: The List All dialog

Window Menu

The *Window* menu contains the following menu choices:

- *Close Window*
- *Entity Dictionary*

Close Window

This option closes the open window, **but**, not necessarily the diagram. If more than one editor window is opened, only the current window is closed and not the diagram. If the last open editor window is closed, the Text Editor will act as if *Exit* has been chosen, possibly in conjunction with a save of information (see “Close Diagram” on page 14 in chapter 1, *User Interface and Basic Operations*).

Entity Dictionary

Opens the Entity Dictionary window. See “The Entity Dictionary” on page 434 for more information.

When working with the Entity Dictionary in the Text Editor, note that the *Link > Create* command will not automatically create an endpoint on the current selection as is the case in the graphical Editors.

Tools Menu

The *Tools* menu contains the following menu choices:

- *Show Organizer*
- *Link > Create*
- *Link > Create Endpoint*
- *Link > Traverse*
- *Link > Link Manager*
- *Link > Clear*
- *Link > Clear Endpoint*

  (The *Link* commands are described in "Link Commands in the Tools Menus" on page 442 in chapter 10, *Implinks and Endpoints*.)

- *Go To Line*
- *Search*
- *Go To Source*
- *Show Transition*
- *Show Nextstate*
- *Show Output*
- *Show Call*
- *Show State*
- *Show Signal*

Go To Line

This menu choice moves the cursor to a line specified by the user. When selected, a dialog will appear querying the user where to move.

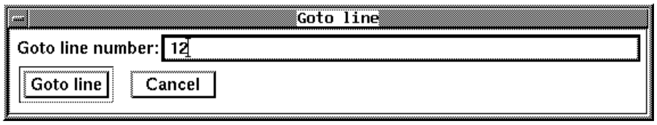


Figure 125: The Go To Line dialog

- *Goto line number*

  In this text field the user can specify a line number.

- *Goto line*

  Moves the cursor to the specified line and closes the dialog. This button will be dimmed if the *Goto line number* text field does not contain a valid line number. If you have entered a line number that is larger than the number of lines in the document, the cursor is moved to the end of the document.

Search

This menu choice opens a dialog where you may search through the document for a text string and replace it with some other piece of text.

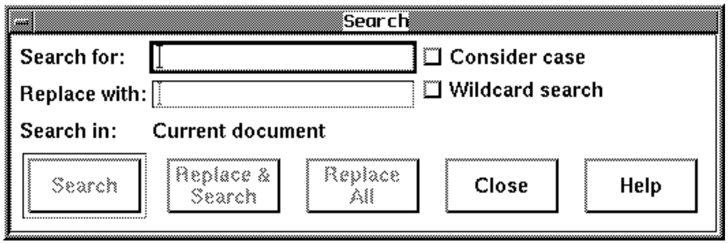


Figure 126: The Search dialog

Dialog Fields and Options

The *Search* dialog contains the following fields and options. Values from the previous invocation is used for settings when the dialog is used again.

- *Search for*

  The text to search for.

- *Replace with*

  Optionally, you can specify a text to replace the text searched for.

- *Search in*

  A read-only field showing the current status of the search. This field will display the text “Current document” until the search has ended, when it will show the text “End of document”.

- *Consider case*

  If this option is set, search is case sensitive.

- *Wildcard search*

  If this option is set, wildcard matching will be used when searching. In wildcard search, the asterisk character (’*’) matches a sequence of zero or more characters of any kind. The backslash character (’\’) escapes the character following it, which is useful if you want to search for asterisks.

Dialog Buttons

When the dialog is first opened, all buttons except *Close* are disabled. Buttons will be enabled when appropriate, for instance the *Search/Restart Search* button will be enabled when a string is entered in the *Search for* text field.

When the dialog opens, searches always start at the top of the document.

- *Search/Restart Search*

  This button changed its name depending on the current search status. When it displays *Search*, it finds the next match in the text. When it displays *Restart Search* it will restart the search from the top of the document when pressed.

- *Replace & Search*

  Replaces the current match with the replace string and searches for the next match.

- *Replace All*

  Replaces all subsequent occurrences of the search string with the replace string.

Go To Source

This command takes the currently selected text fragment and attempts to interpret it as an SDT Reference and display it in the appropriate editor. The selection must include the selected reference exactly. This command corresponds to *Go To Source* in the Organizer.

The syntax of the graphical references used in the Text Editor is described in chapter 19, *SDT References*.

Show Transition

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a transition in a state matrix.

An SDL Editor window is popped up with the SDL input symbol corresponding to the start of the transition currently in focus in the Text Editor.

Show Nextstate

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a transition in a state matrix.

Show Nextstate is a sub menu. The menu choices in the sub menu have the names of the states found in all nextstate symbols that ends the transition currently in focus in the Text Editor. When a nextstate name is selected, an SDL Editor window pops up, showing the nextstate symbol.

Show Output

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a transition in a state matrix.

Show Output is a sub menu. The menu choices in the sub menu have the names of all signals that might be sent from an SDL output symbol during the transition currently in focus in the Text Editor. When a signal is selected, an SDL Editor window pops up, showing the output symbol with the signal sending.

Show Call

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a transition in a state matrix.

Show Call is a sub menu. The menu choices in the sub menu have the names of all procedures that might be called during the transition currently in focus in the Text Editor. When a procedure call is selected, an SDL Editor window pops up, showing the procedure call symbol.

Show State

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a state or transition in a state matrix.

Show State is a sub menu. The menu choices in the sub menu represents all the places where SDL state symbols can be found for the state currently in focus in the Text Editor. If a transition is currently in focus in the Text Editor, the *Show State* operation operates on the start state for the transition. When a menu choice is selected, an SDL Editor window pops up, showing the corresponding state symbol.

Show Signal

This operation is only available for state overview files (`*.ins`), and only when the text cursor is placed close to a signal or transition in a state matrix.

When this operation is invoked, an SDL Editor window is popped up with the signal declaration corresponding to the signal in focus in the Text Editor. If a transition is in focus in the Text Editor, the operation operates on the signal that triggers the transition.

Pop-Up Menu

Open the pop-up menu by right-clicking or pressing `<F2>`. The pop-up menu remains opened until you click the mouse or press `<Esc>`.

Pop-Up Menu in Text Documents

In text documents, the pop-up menu contains some of the menu choices in the *Edit* and *Tools* menus.

In state overview files (`*.ins`), the pop-up menu contains the state overview menu choices in the *Tools* menu. The menu choices are different depending on if a state, signal or transition is closest to the cursor position.

Keyboard Accelerators

In addition to the standard keyboard accelerators, described in "Keyboard Accelerators" on page 35 in chapter 1, *User Interface and Basic Operations*, the Text Editor includes the following:

Accelerator	Command / functionality
`Ctrl+L`	Redraw window.
`Ctrl+Y`	Redo (i.e. undo last undo)
`Ctrl+1`	Show Organizer
`<Delete>`	Clear