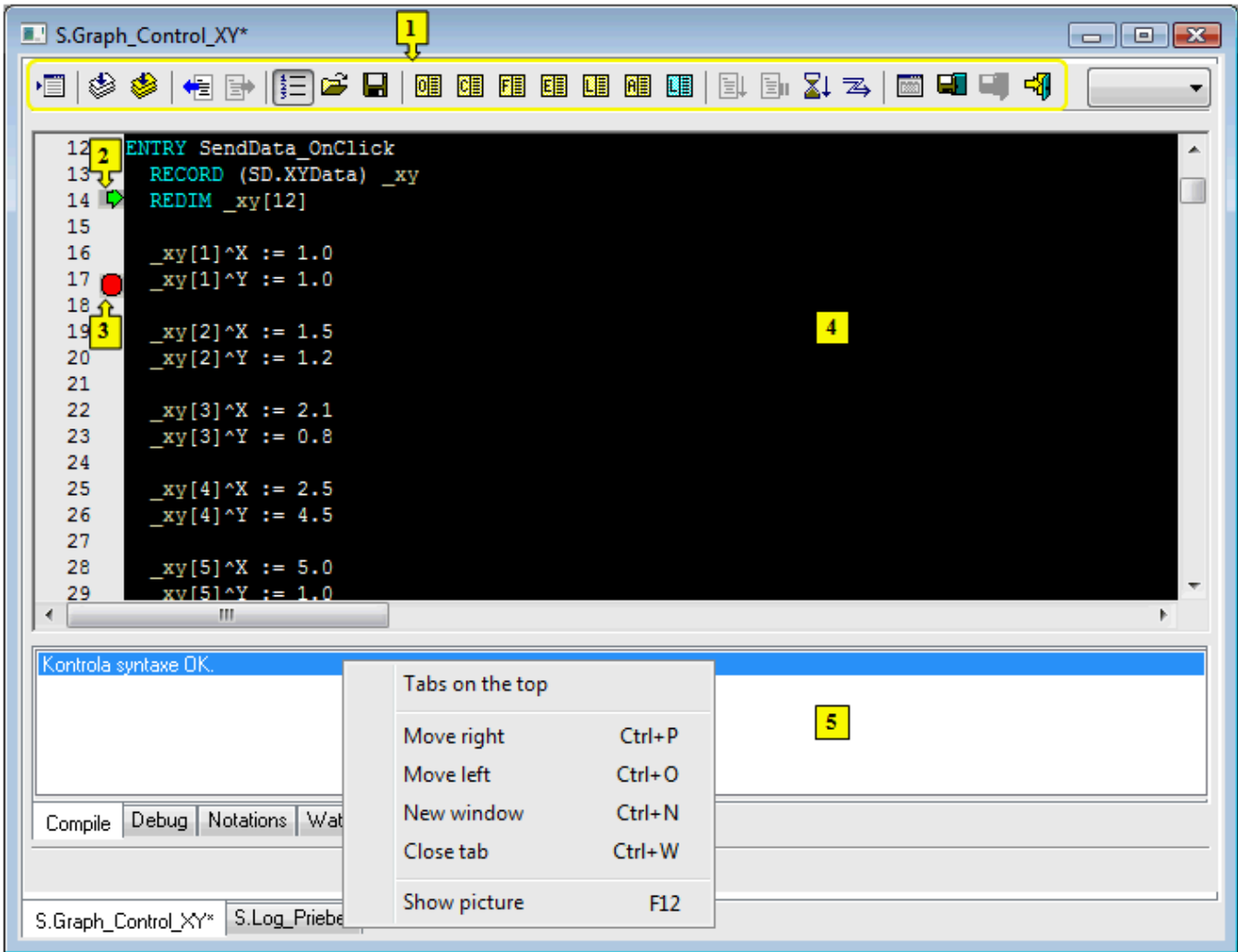




Script Editor

Script editor

The figure below shows the ESL script editor window. Each script opens in own tab.



1	Toolbar.
2	The current action of the debugging process .
3	The break point set for debugging purpose  .
4	The source text of the script - individual actions that form the script.
5	<div>The part of the script editor that contains the following displaying:<ul style="list-style-type: none">• <i>Compile</i> - errors that occurred during the compilation (syntactic, semantic check),• <i>Debug</i> - debugging logs,• <i>Notations</i> - comments existing in the script,• <i>Watch</i> - values of local variables</div> <div>To resize this part and the area for editing the script text point the mouse cursor to the margin between the parts and when the mouse cursor changes its shape to , then press the left mouse button and drag the border (left mouse button has to be still pressed) to the desired position.</div>

Clicking on the tab the following popup menu occurs:

Tabs on the top	
Move right	Ctrl+P
Move left	Ctrl+O
New window	Ctrl+N
Close tab	Ctrl+W
Show picture	F12

- **Tabs on the top** - if the option is checked, the tabs occur at the top, otherwise the tabs are at the bottom.
- **Move right** - moves the tab one position to the right.
- **Move left** - moves the tab one position to the left.
- **New window** - opens the tab in a separate window.
- **Close tab** - closes the tab over which the popup menu was displayed (Ctrl+W closes the current tab).
- **Show picture** - opens the edited picture and minimizes the ESL script. The option is available only for the script of active picture, for the script of event is unavailable.

Notes:

A title bar of tab window and CNF (if the window is maximized) contain the name of active tab. If the tab window is minimized or hidden and the new script is opened the tab window will be displayed again. The insertion point is on the area for editing of the current tab.

When the script editor is closed all the tabs will be closed and the window will hide.

The rules of the tab closing:

- script of the active picture (opened in [D2000 GrEditor](#)) does not display the question whether to save the script providing that there was made some changes (script is occupied by the edited picture),
- script of the event (opened in [D2000 CNF](#) or [D2000 GrEditor](#)) displays the question whether to save the script providing that there was made some changes

Keyboard shortcuts:

Shortcut	Action
CTRL+N	Opens the tab in a separate window.
CTRL+O	Moves the tab one position to the left.
CTRL+P	Moves the tab one position to the right.
CTRL+W	Closes the actual tab.
F12	Opens the edited picture and minimizes the ESL script.

ESL editor features:

- Automatic color differentiation of key words: **END**.
- Automatic color differentiation of texts: correctly written: **"Text"**, incorrectly written: .
- After a successful **Syntax check**:
 - Typing the symbol '^' after the name of an object of **Structured variable** type or after the name of a local variable of **RECORD** type shows the list of all structure columns with the column number and description. The columns are listed in alphabetical order.

```
RECORD (SD.FIND_FILES) _test
REDIM _test[1]
_test[1]^
```

AccessTime	[3]	Čas, kedy bolo k súb...
bFile	[1]	@TRUE, ak ide o súbo...
CreateTime	[2]	Čas, kedy bol súbor ...
FileName	[6]	Meno súboru.
FileSize	[5]	Veľkosť súboru v baj...
FullAttrib	[7]	Atribúty súboru (adr...
LastWriteTime	[4]	Čas, kedy bolo do sú...

- Pointing the mouse cursor to a local variable, or a structured variable item displays information about its type and the place where it is declared (in the debugging mode, the value of the variable is shown).
 - Right-mouse click above identifier (local variable, procedure name) opens the local popup menu containing the only item - **Go to definition** - after clicking the item the cursor is automatically moved to the declaration of the identifier.
- After an unsuccessful **Syntax check** or **Compilation**, double-clicking an error from the lists of errors (5) moves the mouse cursor to where the error occurred.
- Pressing **F1** (help) when the cursor is pointed to the name of a function / action opens the help document for the given function/action/picture event.
- Automatic display of function description and types of parameters. If user enters the function name and the first bracket in script editor the window will appear where the function and its parameters are described. Parameter is highlighted in dark blue color when you proceed to the new parameter.

Example:

```
%SetItemValue{
END
SetItemValue

Function sets item value given by row and column index

Declaration :

%SetItemValue(HBJ, INT, INT, NONE)
```

- Automatic display of action and its parameters. If user enters the action name and presses the button SPACE the window with action's declarations will be shown in editor.

Examples:

```
DB READ BLOB handleIdent Int, BlobColNameIdent str, blobFileNameIdent Str,
DB_READ_BLOB handleIdent_Int , blobColNameIdent_str , blobFileNameIdent_Str , retCodeIdent_Int , idKeyIdent
DB_READ_BLOB handleIdent Int , blobColNameIdent_str , blobFileNameIdent_Str , retCodeIdent_Int [ WHERE strExpression Str ]

DB READ BLOB handleIdent Int, BlobColNameIdent str, blobFileNameIdent Str, retCodeIdent_Int ,
DB_READ_BLOB handleIdent_Int , blobColNameIdent_str , blobFileNameIdent_Str , retCodeIdent_Int , idKeyIdent

DB READ BLOB handleIdent Int, BlobColNameIdent str, blobFileNameIdent Str, retCodeIdent_Int WHERE
DB_READ_BLOB handleIdent_Int , blobColNameIdent_str , blobFileNameIdent_Str , retCodeIdent_Int WHERE strExpression Str
```

- When you write an object identifier (picture object identifier or system server event identifier) and the symbol "]" after it the list of RPC procedures and interface, defined for this object, will display. The selected definition of procedure or interface will be added to script editor.

Example:

```
TEXT _msg
_msg := " ... "
CALL [E.MSGServer]
I.MsgServer*RegisterClient(IN TEXT _clientName, IN INT _clientHOBj, IN INT _clientProcessHOBj, INT _clientUID)
I.MsgServer*UnRegisterClient(IN INT _clientUID)
I.MsgServer*GetClientList(RECORD NOALIAS (SD.prva) _clients)
I.MsgServer*SendMessage(IN INT _srcClientUID, IN INT _dstClientUID, IN TEXT _msg, INT _retCode)
GetClientNumber(IN INT _ahoj)
SetTime(IN TIME _time)
```

- When entering the symbol "^" after the interface name, the list of procedures, defined for this interface, displays. The selected definition of procedure will be added to script editor.

Example:

```
TEXT _msg
_msg := " ... "
I.MsgServer^
RegisterClient(IN TEXT _clientName, IN INT _clientHOBj, IN INT _clientProcessHOBj, INT _clientUID)
UnRegisterClient(IN INT _clientUID)
GetClientList(RECORD NOALIAS (SD.prva) _clients)
SendMessage(IN INT _srcClientUID, IN INT _dstClientUID, IN TEXT _msg, INT _retCode)
```

- View of RPC procedure description called in ESL script. This property allows to write the description of RPC procedures which is visible in a different script. The comments and parameters of RPC procedure are shown in ESL script. The description can consist of one or more rows, the empty row is not allowed. The declaration of procedure have to follow the description (without the empty rows). The description starts with a character ";", like the comments.

Example of description:

```

12
13 ;a procedure returns the number of clients
14 RPC PROCEDURE GetClientNumber (INT _nrClient)
15 END GetClientNumber
16
17 ;procedure sets the time
18 ;the value must be valid
19 RPC PROCEDURE SetTime (IN TIME _time)
20 END SetTime
21

```

Example of display:

```

60
61 CALL [E.MSGServer] SetTime(_time) ON _FROM_HIP
62
63 CALL [E.MSGServer] GetClientNumber(_a) ON _FROM_HIP
64
65
66
67
68
69
70
71

```

[E.MSGServer] GetClientNumber

a procedure returns the number of clients

Parameters :

_nrClient : INT

- Automatic indentation
When writing the ESL script and moving on the new line, ESL editor automatically indents the current line (according to the first non-zero line) and move the cursor on the given position.

Features of automatic indentation:

- When moving on the new line after the action (e.g. RPC, PROCEDURE, PUBLIC, FOR, DO_LOOP, IF...) in ESL editor, the text is automatically indented by 2 characters to the right from the previous line.
- When moving on the new line after the action (e.g. END_LOOP, ENDIF...) in ESL editor, the text is automatically indented by 2 characters to the left from the previous line.
- When pressing Ctrl+i, the selected text is formatted according to the previous settings. For this action is important the indentation of the first line within the selected lines.

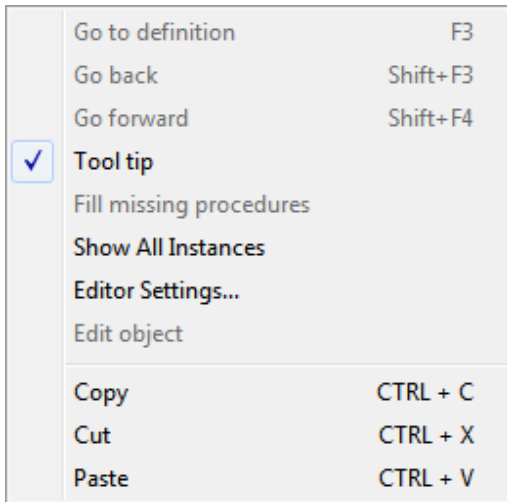
The automatic indentation may be enabled / disabled in the dialog box [Settings](#) under the menu **System**.

- Keyboard shortcuts:

Shortcut	Action
CTRL+K	Causes the current line, or current selection of lines to be commented.
CTRL+SHIFT+K	Removes the comment of current line, or current selection of lines.
CTRL+F	Displays the dialog box to search or to replace a text in the script.
CTRL+G	Goes to given line in the text.
CTRL+i	Indents the selected rows in the script. The indentation of the first row of the selected part of script is the decisive.
CTRL+S	For active picture script - saving the picture. For script of an object of Event type - saving.
CTRL+1	Opens a list of D2000 system objects .
CTRL+2	Opens a list of predefined constants .
CTRL+3	Opens a list of functions .
CTRL+4	Opens a list of actions .
CTRL+5	Opens a list of local variables .
CTRL+6	Opens a list of value attributes of object or local variable.
CTRL+~	ESL script editor will complete automatically a written word according to next suitable one (from cursor position) and later according to its type (function, constant, action, ...). To add other adequate words press the button repeatedly.
CTRL+>	Shows tool tip in ESL script editor even if the automatic tool tip is switch off.

Popup menu

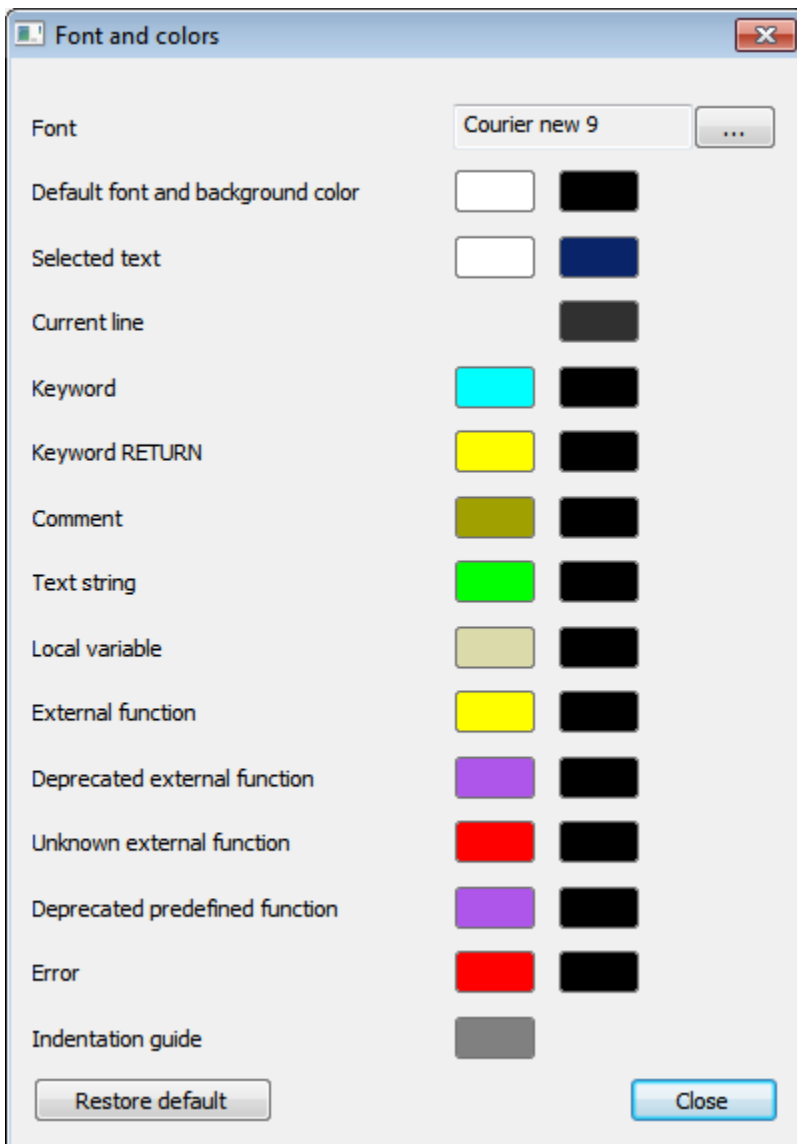
The popup menu containing these items can be displayed over the script source text (part 4 on the picture). Click by right mouse button or push **Menu key** on the keyboard.



- **Go to definition** - cursor is automatically moved to the declaration of the identifier. When using "Go to definition" on the remote procedure ([RPC/UBLIC](#)), ESL editor automatically opens the script, which contains the definition of procedure, and moves the cursor on the definition.
- **Go back** - cursor is moved back on the place where "Go to definition" had been called from. It keeps the last 50 callbacks "Go to definition".
- **Go forward** - cursor is moved on the place where it was before "Go back" calling. "Go back" and "Go forward" are supported even between the different ESL scripts. ESL editor automatically chooses / opens the ESL scripts
- **Tool tip** - enables / disables displaying of a tool tip.
- **Fill missing procedures** - adds the missing procedures (with an empty body) defined in ESL Interface which have been declared in ESL script. If menu opens by right-click over declaration of used ESL Interface, only its procedures will be added. Otherwise, the procedures of all ESL Interfaces declared in ESL script will be added.
- **Show All Instances** - shows the list of all running instances of the edited [ESL script](#).
- **Editor Settings...** - shows ESL editor settings dialog (fonts and colors).
- **Edit object** - if the text under the mouse cursor is a valid object name, the option enables to open it for editing.
- **Copy** (CTRL+C) - copy selected text into clipboard,
- **Cut** (CTRL+X) - delete selected text,
- **Paste** (CTRL+V) - paste selected text from clipboard.

Colors in ESL script

For better orientation in ESL script, editor uses different coloring scheme for each type of the individual text tokens. Default coloring parameters can be modified using ESL editor settings dialog.



ESL Editor within *String* detects the references to [dictionary](#). If some reference to a dictionary (key), which has not defined yet, is identified, it will be colored as *Error* (see the dialog above).

Example:

The key `D2_ActAlarm` exists in the dictionary, but `D2_ActAlarmAAAAA` does not exist.

```

16  _t := "D2_ActAlarm"
17  _t := "... {!D2_ActAlarm} . text."
18  _t := "... {!D2_ActAlarmAAAAA} . text."

```