XDS Family of Products

# XDS Modula-2 IDE

for Microsoft Windows

Version 1.7.0

## User Guide

EXCELSIOR

http://www.excelsior-usa.com

Information in this document is subject to change without notice and does not represent a commitment on the part of Excelsior, LLC.

Excelsior's software and documentation have been tested and reviewed. Nevertheless, Excelsior makes no warranty or representation, either express or implied, with respect to the software and documentation included with Excelsior product. In no event will Excelsior be liable for direct, indirect, special, incidental or consequential damages resulting from any defect in the software or documentation included with this product. In particular, Excelsior shall have no liability for any programs or data used with this product, including the cost of recovering programs or data.

# Contents

# Chapter 1

# Getting started

XDS Modula-2 integrated development environment (**XDS Modula-2 IDE**) is used for the Modula-2 source code editing and interation of various Modula-2 tools and utilites into the single environment.

XDS Modula-2 IDE is implemented as a number of Eclipse platform plugins. IDE itself doesnot includes Modula-2 development tools like compiler, debugger, etc. Instead IDE integrates with already installed *development systems* (SDK).

IDE uses its own project system comprised of: *Workspaces* (see 3.1) (Workspace), *Projects* (see 3.2) (Projects) and *Resources* (see 3.3) (Resources).

Each workspace can contain one or more XDS IDE projects. It is possible to cluster projects into a number of groups and apply different settings to each of these group.

It is possible to create an arbitrary number of workspaces. However, each launched instance of XDS IDE works only with the one workspace at the moment, and locks used worspace making it inaccessible for the other XDS IDE instances. Switching between workspaces restarts the IDE.

## 1.1 IDE installation and start

IDE is distributed as the ZIP-archive. To instal the IDE unpack the archive to the local directory of your PC. Your PC hard drive should have at least 300 MB of free disk space.

No extra actions necessary. ZIP archive contains all required IDE components.

To start IDE start `xds-ide.exe` file from the installation directory. After IDE startup the **XDS Modula-2** *projection* (see 4.2) will open. This projection is configured to support the Modula-2 development.

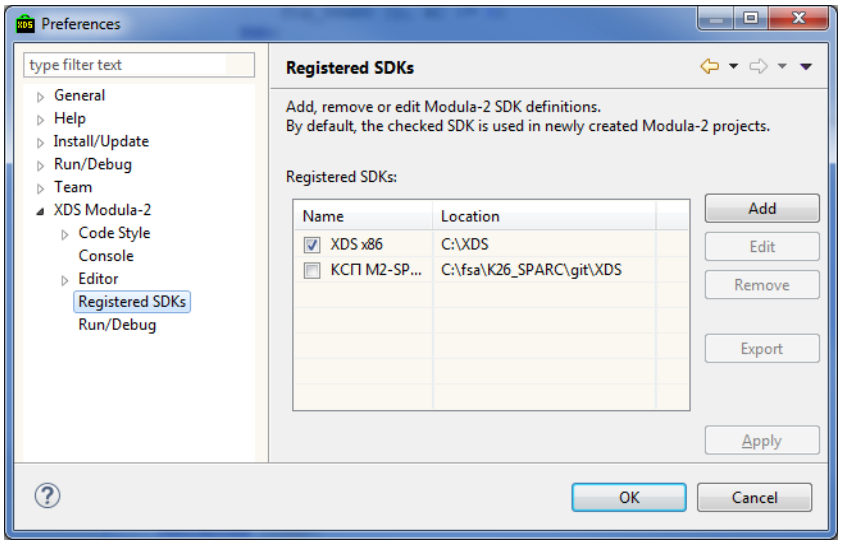
After the first launch workspace is created at the `%USERPROFILE%\xds-workspace` – user home subfolder.

By default subsequent launches will open last opened workspace and perspective.

## 1.2 Add SDK

IDE allows adding and usage of several development systems (SDKs). Development system should be installed locally or to the network drive.

To add SDK open the **Preferences** dialog by clicking **Window** > **Preferences** at the main menu. The select at the dialog left pane **XDS Modula-2** > **Registered SDKs**. After that press the **Add** button at the dialog right and select the SDK folder.




When the selected SDK is configured to be used with the IDE (i.e. SDK root folder contains the `sdk.ini`) then the installed SDK list will show the new added SDK record.

If the `sdk.ini` file is absent then manual SDK configuration wizard will show up. One can always open the SDK configuration wizard by selecting the SDK in the list and pressing the **Edit** button.

After the applying changes by pressing the **OK** button IDE is all set for the Modula-2 development.

IDE allows usage of several SDKs at a time. Each SDK is given a unique name. In the cast of name conflict number suffix is used to resolve it.

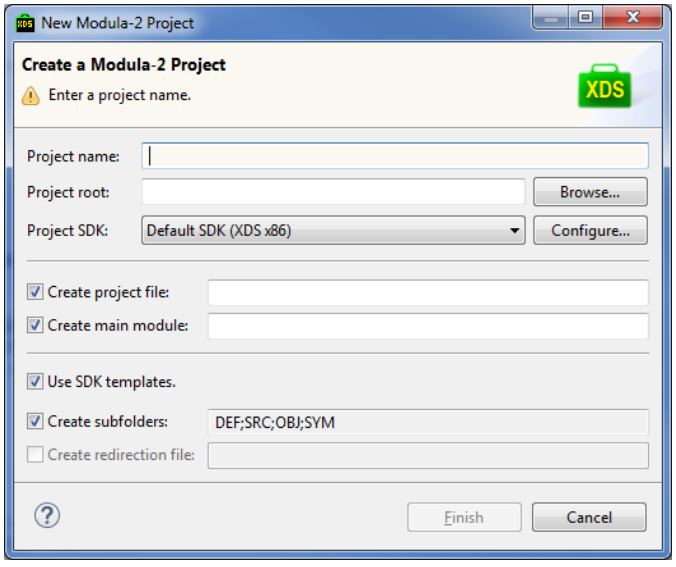
Checkboxed SDK is the default development system. It is used in the case no SDK have been selected for the given project.

# Chapter 2

# First steps

## 2.1 Create new project

To create the new project select **File** > **New** > **Modula-2 Project** at the main menu. In the opened wizard specify new project parameters.




Specify new project name (**Project Name** field).

Then specify project root directory location:(**Project root**).

Select the project SDK (**Project SDK** field): Unless otherwise indicated the default SDK is used. The **Configure...** button will open the SDK settings dialog.

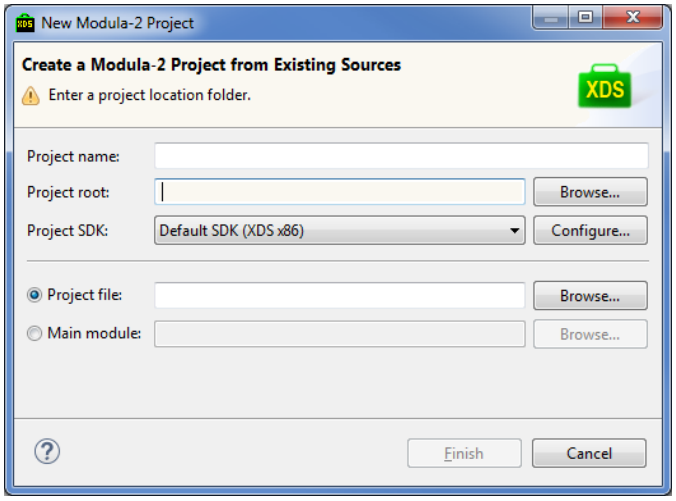
One can create main module or project file, standard directory structure and redirection file. If **Use SDK templates** is checked then SDK templates will be used to create the files. Directory structure and redirection file name are also

taken from the selected SDK settings.

Pressing the **Finish** button will create the new Modula-2 project and add it to the workspace. Newly created files and directories will show up in the Project Explorer.

## 2.2 Create project from existing sources

To create project from existing sources select **File** > **New** > **Modula-2 Project from Existing Sources** in the main menu. Specify project settings in the wizard opened.




You can explicitly specify the new project name or it will be set automatically after the project root directory selection (**Project root** field).

Select the project SDK (**Project SDK** field): Unless otherwise indicated the default SDK is used. The **Configure...** button will open the SDK settings dialog.

Specify whether project file (**Project File** field) or main module (**Main module**) is used for the source build. One can select project file `*.prj` or main module file (`*.mod` or `*.ob2`) by pressing **Browse...** button.

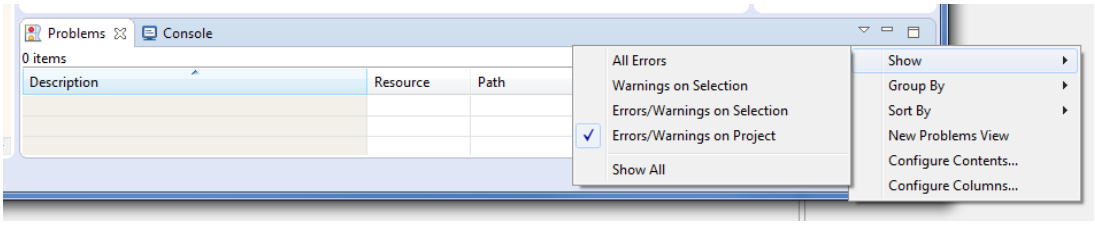
Pressing the **Finish** button will create the new Modula-2 project and add it to the workspace. Newly created files and directories will show up in the Project Explorer.

## 2.3 Build project

To compile project select it in the Project Explorer view and the click **Project** > **Build Project** or **Project** > **Rebuild Project** in the main menu. First compiles the project incrementally while the second rebuilds the project from scratch. These commands also can be invoked via the project element context

menu at the Project Explorer view or by using hot key combinations `Shift + F9` or `Ctrl + Shift + F9` respectfully.

Compiler errors and warnings show up in the **Problems** view. By default markers are shown for all open projects in the IDE. To restrict markers to be shown for the single project select **Show** > **Error/Warnings on Project** item in the toolbar.

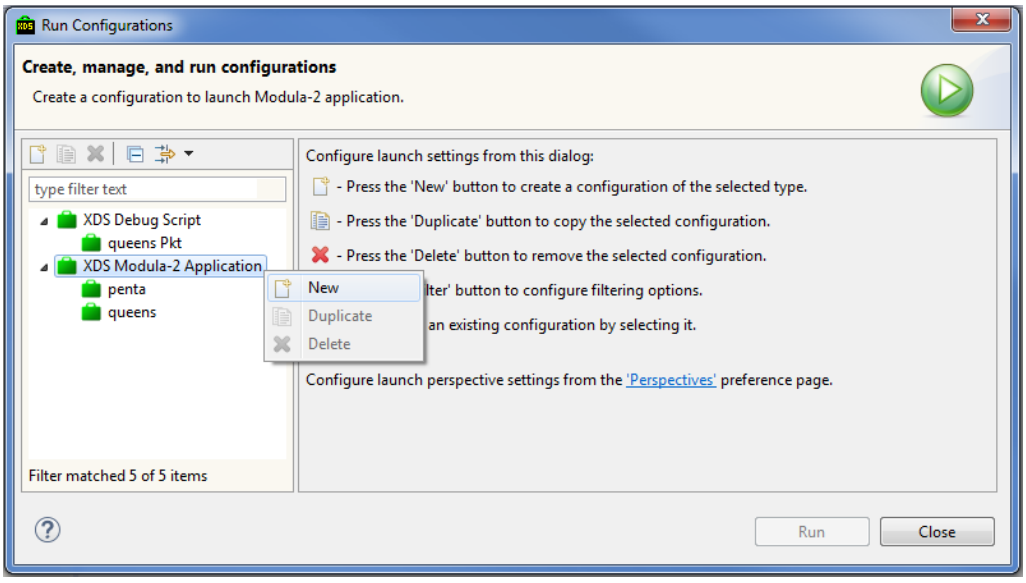



## 2.4 Run program

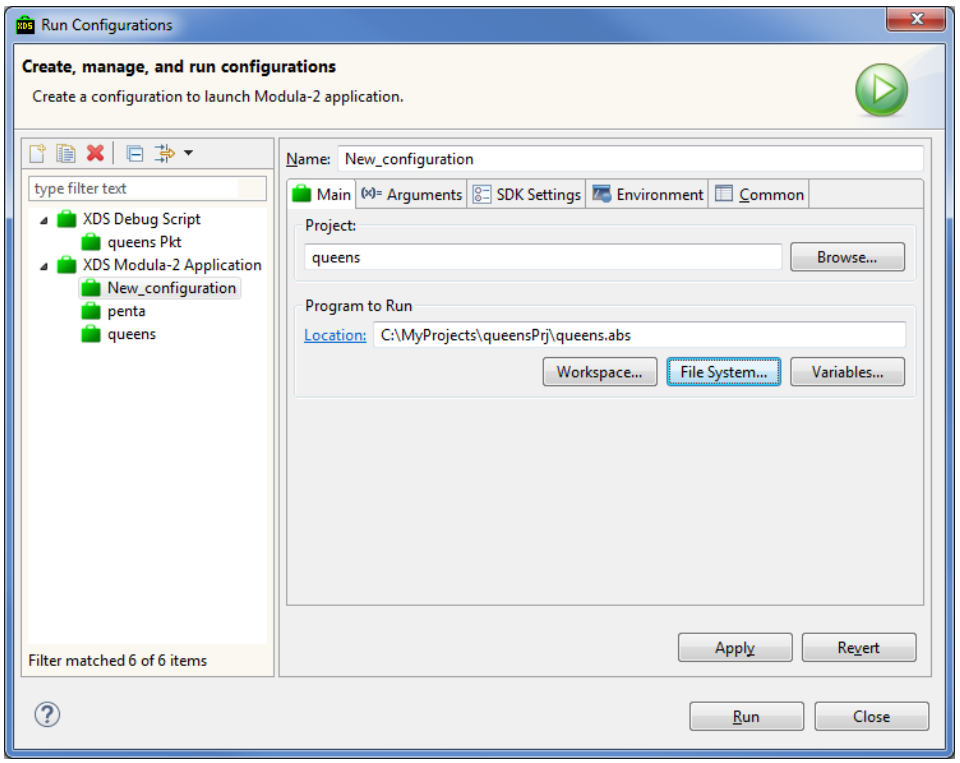
To run the Modula-2 program create the (**Run Configuration**). Run configuration specifies the project, executable file path and optional launch parameters for the program.

To create run configuration select **Run** > **Run Configurations...** in the main menu and specify run configuration parameters.

In the opened dialog select **XDS Modula-2 Application** in the left pane, open context menu with the right mouse click and select **New**.




New run configuration will be created and right pane will show run configuration editor.




Type run configuration name in the **Name** field.

Select target workspace project for this run configuration and executable file for the launch. **Apply** button saves changes for the edited run configuration, **Revert** button reverts run configuration parameters to the last saved state.

Other tabs allow to set additional optional run configuration parameters.

After filling up mandatory fields press the **Run** button to launch the program.

Once created launch configuration can be used again. To do this select it from the Main menu dropdown: **Run** > **Run History**.

The majority of launch actions for the progam is available via the **Run** toolbar button:

This button shows the list of last invoked last configurations and their corresponding edit actions. Progam standard output and standard error will show up in the **Console** view.

## 2.5 Debug program

To launch Modula-2 program in debug mode create (**Debug Configuration**) or use already existent **Run Configuration**.

To create the Debug Configuration select **Run** > **Debug Configurations...** in the main menu. In the opened dialog select **XDS Modula-2 Application** in the left pane, open context menu with the right mouse click and select **New**.

Type run configuration name in the **Name** field.

Other fields can be filled the same way as for the Run Configuration.

After filling up mandatory fields press the **Debug** button to launch the program in the Debug mode.

Once created Debug configuration can be used again. To do this select it from the Main menu dropdown: **Debug** > **Debug History**.

The majority of launch actions for the progam is available via the **Debug** toolbar button:

This button shows the list of last invoked last configurations and their corresponding edit actions.

# Chapter 3

# Basic concepts

The XDS Modula-2 Integrated Development Environment (XDS Modula-2 IDE) is based on the Eclipse platform. Eclipse is the open-source IDE platform simplifying such tasks as resource management, program launching, editing and debugging.

This chapter describes common idioms of the Eclipse platform, leveraged by the XDS Modula-2 IDE.

## 3.1 Worspace

*Worspace* – first of all, this is a project container. Besides, it contains settings used by the IDE components and auxilary data. Workspace is the directory containing project (see 3.2) subdirectories. Special `.metadata` subdirectory is used as the common data and settings storage, used by this workspace projects.

It is possible to have an arbitrary number of workspaces. However only one workspace can be opened in the IDE at the given moment.

On the first launch the workspace is created in the user‘s home directory. To create new workspaces and switch between them select **File** > **Switch Workspace** > **Other...**. Then select the workspace folder. If the folder selected doesnot contain the workspace it will be created there.

On IDE start last opened workspace is loaded by default. It is possible to force workspace selection dialog on IDE start. For this open **Preferences**, by clicking **Window** > **PreferencesPreferences**, then select in the left pane **General** > **Startup and Shutdown** >**Workspaces** and turn on the option **Prompt for workspace on startup**.

When using several workspaces sometimes it is necessary to synchronize their settings. For this please use **File** > **Export...** > **General** > **Preferences** in the source workspace and then **File** > **Import...** > **General** > **Preferences** for the target workspace.

It is possible to explicitly specify workspace on IDE startup by using the fol-

lowing command line:

`-data WorkspacePath`

where `WorkspacePath` is the required argument specifying path to the workspace.

Another command line parameter turns on the workpace path display in the IDE‘s main window title:

`-showlocation [WorkspaceName]`

where `WorkspaceName` is the required argument specifying label to be displayed in the main window title instead of the workspace path displayed by default.

## 3.2 Project

*Project* – is the resource container, containing folders and files, their processing rules and other project properties (such as the compiler settings). Physically it is the directory whose files and folders are the project‘s resources. When creating project it is necessary to specify where to place it on the file system.

In the project‘s directory special file named `.project` is created. This directory also contains `.settings` folder used as project properties storage. Usually, IDE stores other auxilary files and folders with names starting with the `"."` symbol.

There is no one-to-one correspondence between XDS project files (*.prj-files) and IDE projects. XDS project files are just particular kind of the XDS IDE resources. XDS IDE project can contain several prj-files.

It is not necessary to place the project inside the workspace.

It is recommended to place XDS IDE project in the same folder as the Modula-2 program folder. In particular, this simplfies the transfer of the project between workspaces.

In order to import the existing project in to the workspace on can use **File** > **Import...** > **General** > **Existing Projects into Workspace**.

It is assumed by default that all project resources are stored inside the project‘s folder.

To use resources outside the project‘s folder it is necessary to add them via the so called *Linked Resources* mechanism.

## 3.3 Resources

*Resources* – this is general term for projects, folders and files operated by the integrated environment. IDE‘s files and folders corresponds to the usual files and folders of the file system. Folders can contain files and other foldes but not projects. Folders themselves contained in projects and other folders.

Files and folders can be linked to files and folders outside the project‘s directory.

These files and folders are called *linked resources*.

Navigational panels, such as **Project Explorer** (**Project Explorer**) (see 4.4.1) display hierarchical structure of the resources and facilitates resource opening and editing. Besides files and folders navigational panels can display *virtual folders* which are not correspond to any real file system directory.

**See also**

Resource properties (see 6) New element creation (see 7.1) Project (see 3.2)

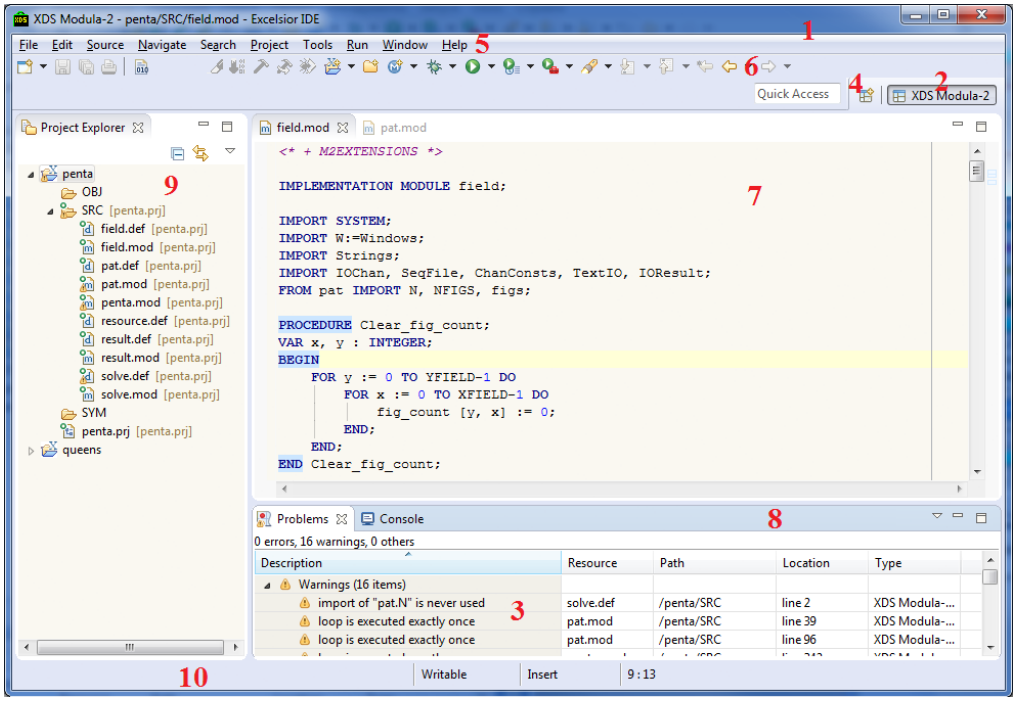
## 3.4 Plugin

The key feature of the Eclipse platform is flexibility and extensibility enabled by the platform modular architecture. Beside the compact runtime core all platform‘s components and Eclipse-based IDE are *Plugins*. These plugins are dynamically loaded independently of each other. Certanly, it is possible to specify for the plugin which plugins are dependendant from it - and thus must be loaded before. Platform runtime core locates and integrates plugins in to the running platform. In this regard standard and newly created plugins are functionaly equivalent.

# Chapter 4

# User interface

Main elements of the user interface are shown in the picture below:




Main elements are:

1. *Workbench* – container containing all other windows. It contains main menu, toolbars and child windows.

2. *Perspective* – visual container for *Views* and *Editors* related to the certain task. The picture shows **XDS Modula-2** perspective - it is shown in the upper left label atop the toolbar. This perspective is used to develop Modula-2 programs.

3. *View* – visual container used to render the current selection. The picture shows the **Problems** view which renders the problems for the current selection.

4. *Short Cut Bar* – shortcuts to access certain perspectives.

5. *Menu Bar* – the set of context-dependent actions invoking certain functionality.

6. *Tool Bar* – the set of context-dependent actions invoking certain functionality. All toolbar actions are also accessible via the Menu.

7. *Editor* – instrument to edit the project resources. Unlike the View the contents of the Editor are defined during the open action, for example, when user double clicks the source file.

8. *Project Explorer* – view which for the management of project and resources. This view is described separately.

9. *Status Bar* – special workbench element which is used to show extra information, such as: document parameters, hints, etc. The picture shows the Status bar for the active Editor.

Editors and views are child windows of the Workbench. Child windows can be active and inactive but only one child windows can be active at the given moment. Active window has its title highlighted. The active window contents are usualy the target for the Copy, Cut and Paste operations. It is the active window on which the Status bar is dependent from. Views can show information from the last active editor, if there no any active editor at the moment.

One can cycle through the workbench windows using the `Ctrl + ~` hotkey.

Lets descrive some main elements in details.

## 4.1 Workbench

*Workbench* – is the main window of the IDE which introduces the framework for managing and navigating workspace (see 3.1) resources.

Workbench can be rearranged using one of the several perspectives (see 4.2), which are govern the layout of views (see 4.4) and editors (see 4.3), organizing them in a way most appropriate for the certain task.

## 4.2 Perspectives

*Perspective* (Perspective) defines the set and the layout of views, editors, menu items and other elements visible to user. Perspective can have an arbitrary number of views and editors. There can be any number of perspectives, but workbench can have only one active perspective at a time. To see all available

perspectives select **Window** > **Open Perspective** > **Other...** in the main menu.

Main purpose of the perspective is to give user the most appropriate set of tools for the particular task, so all necessary tools will be at hand.

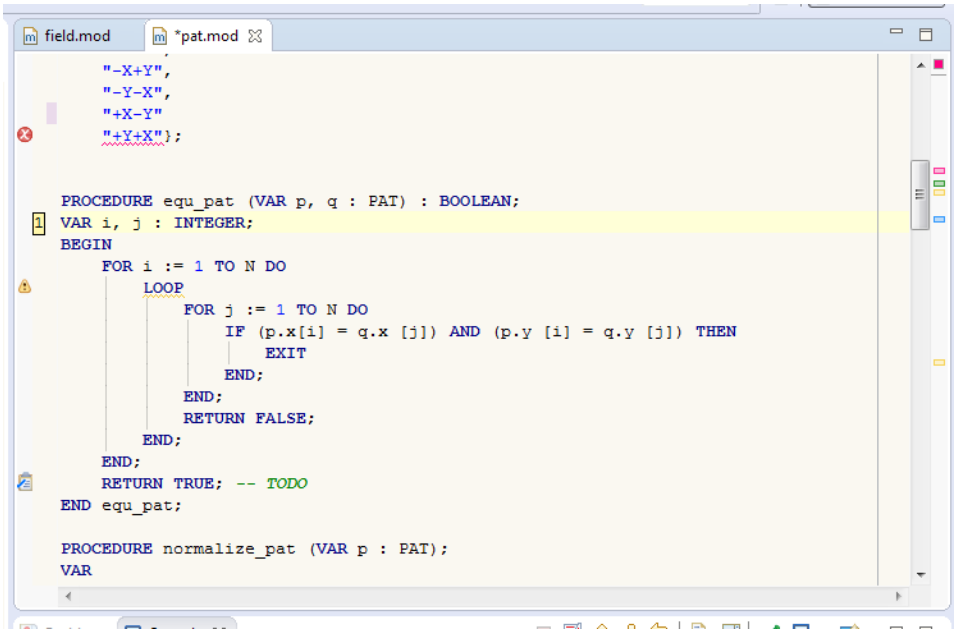
It is possible to restore the initial layout of the perspective by selecting **Window** > **Reset Perspective...** at the main menu.

## 4.3 Editor

*Editor* is used to render and edit the resource (usualy file) contents. For each file the separate instance of editor is created.

Changes done in the editor are not commited to underlying file until the save command is invoked.

The editor's title contans the * symbol in it when the editor contents is modified but is not written to file.




One can cycle through the editors using the `Ctrl + Tab` hotkey.

## 4.4 Views

*Views* usualy provide information about the selected object or element. Some views provide global information, like state of the file system (Project Explorer).

Unlike editors, views immediately apply changes to the object being edited.

For example, **Project Explorer** (**Project Explorer**) (see 4.4.1) lacks the Save command – user action like file copy or rename are immediately affect the corresponding resources. However, few views violate this agreement and require explicit save action from user.

Every view can be closed and then reopened via the **Window** > **Show View** > **Others...** main menu command.

### 4.4.1 Project Explorer

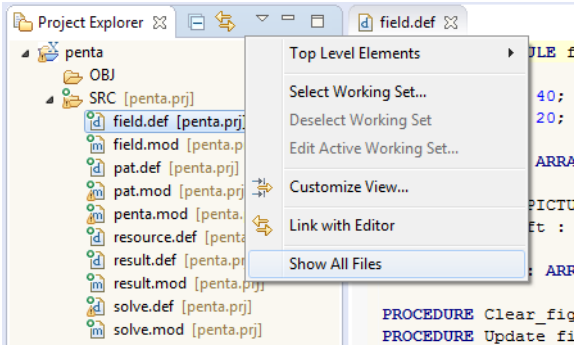
**Project Explorer** renders resources from a project of the current Workspace (see 3.1) as the tree-like structure. This view allows:

- navigate through files and folders of a project,
- open editors for files,
- Create new projects, files, folders,
- delete, rename or move existent files/folders,
- conduct import/export of projects and settings.

Files selected in the **Project Explorer** can immediately affect contents of other IDE views.

Context menu can be opened for any **Project Explorer** element by right clicking this element. Context menu allows to view/edit element properties.

For Modula-2 prjects only Modula-2 source files and packet files are shown – that is, files with the following extensions: `*.mod`, `*.def`, `*.prj`, `*.pkt`, `*.res` and `*.ldp`. Other files are filtered out. To see all files select **Show All Files** action in the Project Explorer action bar.




Modula-2 files which are included in the project compilation set are marked with green dot in the file icon‘s upper right corner. Besides, next to the file name used project file or main module name is shown.

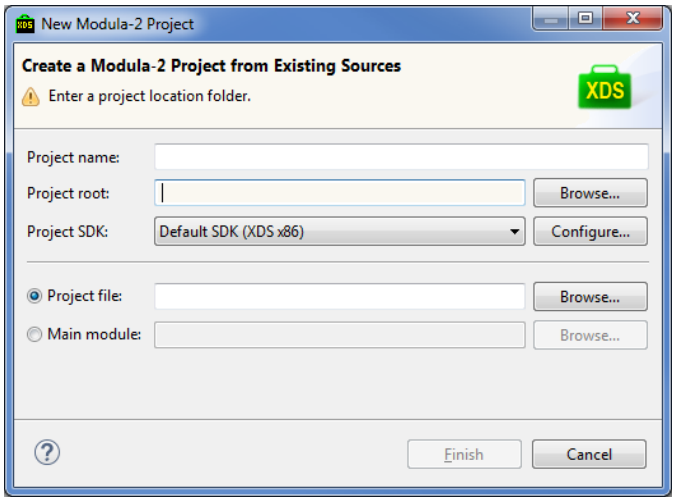
Modula-2 files which are not children of project directory are included in the **External Dependencies** virtual folder.

If SDK is set to show definition modules then **SDK Library** virtual folder will show definition module resources.

## 4.5 Wizard

Wizards are used for the step by step configuration of various parameters.

Typical example of wizard is on the picture below:

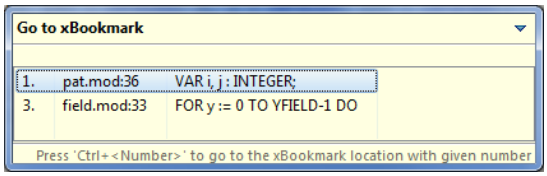



## 4.6 Navigation

### 4.6.1 Numbered Bookmarks

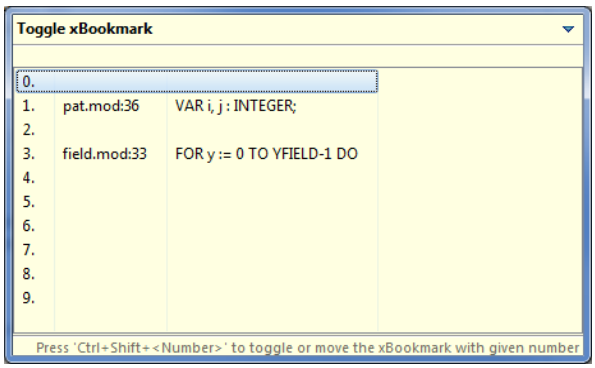
Numbered bookmarks are used for the quick navigation in editors.

To set bookmark in the current line press `Ctrl + Shift + <Number>`, where `<Number>` – is an arbitrary digit from 0 to 9. So it is possible to set maximum 10 numbered bookmarks. When setting the bookmark, if the digit is already assigned to the other bookmark then the digit will be re-assigned to the new bookmark. If press `Ctrl + Shift + <Number>` once again – bookmark will be deleted. Press `Ctrl + <Number>` to jump on the particular numbered bookmark. To see all numbered bookmarks press `Ctrl + =`.




Select bookmark with the cursor key and press `Enter` to open the bookmark location (new editor may open).

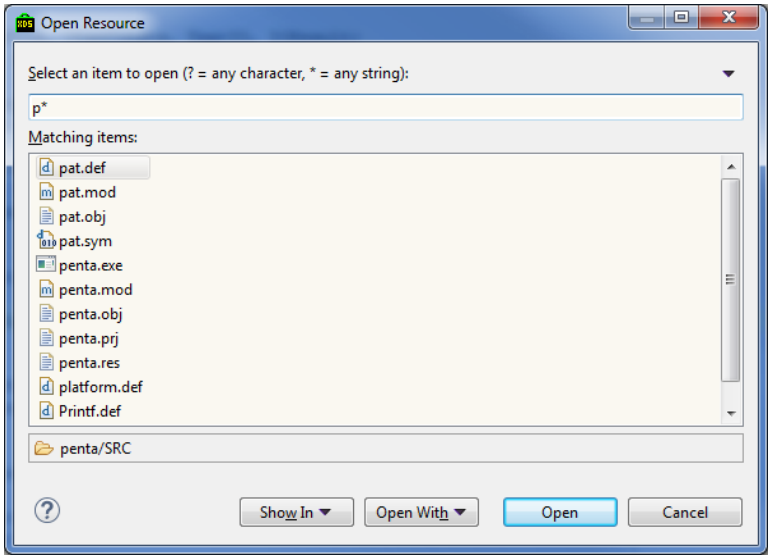
The particular bookmark can reassigned or delete with the `Ctrl + Shift + =` hotkey.




To remove all bookmarks select **Navigate** > **xBookmarks** > **Remove All xBookmarks** in the main menu.

### 4.6.2 Open resource

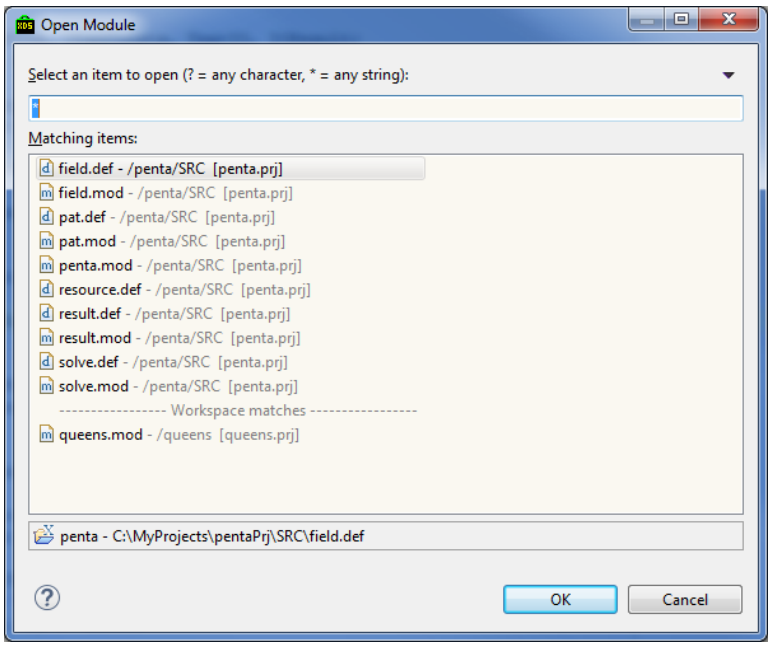
To quickly open workspace resource use the `Ctrl + Shift + R` combination.




Dialog will show all workspace resources (including filtered out in **Project Explorer**).

Text field atop allows to filter resources by name. Wildcards are supported : `*` stands for the arbitrary string, `?` stands for the single character or emptry string.

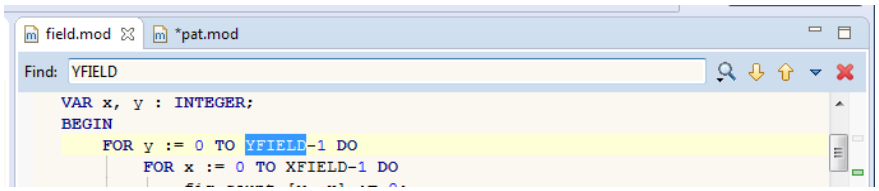
To select the Modula-2 module use `Ctrl + M` hotkey.




### 4.6.3 xFind

To search through the editor contents use **xFind panel**. It can be invoked with `Ctrl + F` hotkey.

Atop the editor text field will appear, input the search pattern there. It is possible to change the panel placement in the settings (see 4.6.3).




Default Eclipse search dialog can be opened with the `Ctrl+Alt+F` combination.

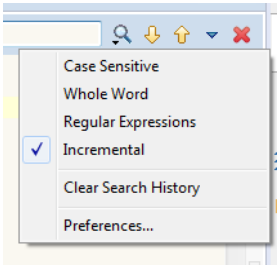
Press `Esc` to close the panel.

For now, **xFind panel** only supports search, to replace use the standard **Find/Replace** dialog, invoked by the `Ctrl + Alt + F` hotkey or via the Eclipse main menu.

After typing the search text use `Up/Down` or `Shift+Enter/Enter` to navigate through the occurences.

If the **xFind panel** has input focus then `Ctrl + Down` will show the search history. `Shift+Down` hotkey will put the next item from the search history in to the text field.

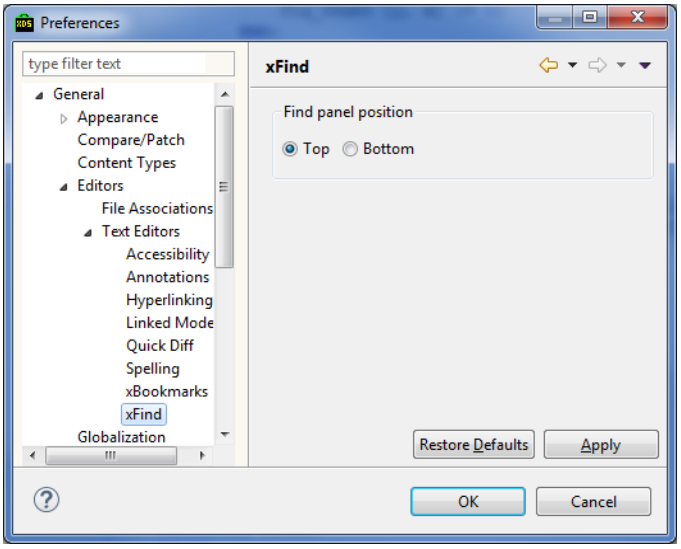
**xFind panel** supports all modes of the standard **Find/Replace** dialog, you can set them in search settings menu:




Besides the **xFind panel** it is possible to quick find the word. For this just place cursor on the word in the editor and press `Ctrl+Up` or `Ctrl+Down`.

### xFind preferences

One can access the **xFind panel** settings using the **Preferences** dialog on the**General** > **Editors** > **Text Editors** > **xFind** page.




Here you can select the xFind panel (see 4.6.3) placement in the editor.

# Chapter 5

# Parameters

IDE preferences management is done via the **Preferences** dialog. Select **Window** > **Preferences** in the main menu to open this dialog.

The great number of preferences is supported for the IDE. Preferences are split into groups which are displayed on separate pages. Preference pages in turn are combined into the hierarchial structure displayed in the left pane of the dialog. In the upper part of the dialog filter text field is located which allows quick search of the particular page.

Press the **Apply** button to apply changes and continue the preferences edit. To aply changes and close dialog press **OK** button. **Restore Defaults** button resets preferences to their defaults.

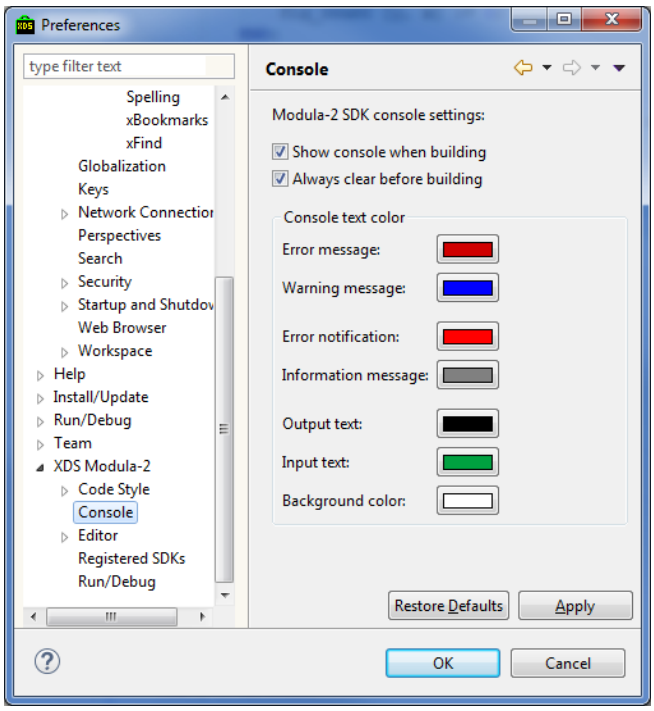
## 5.1 XDS Modula-2

Modula-2 specific preferences are located in the **Preferences** dialog on the **XDS Modula-2** page and on the pages below:

- **Console** (**Console**) (see 5.1.1)
- **Editor** (**Editor**) (see 5.1.2)
- **Registered SDKs** (**Registered SDKs**) (see 5.1.3)

### 5.1.1 Console

Console preferences are located in the **Preferences** dialog on the **XDS Modula-2** > **Console**.




In the upper part of the dialog there are two checkboxes:

- **Show console when building** – whether to show console when building;
- **Always clear before building** – whether to clear console before building.

**Console text color** page specify color key for the various message types, text and background coloring. The color list is as follows:

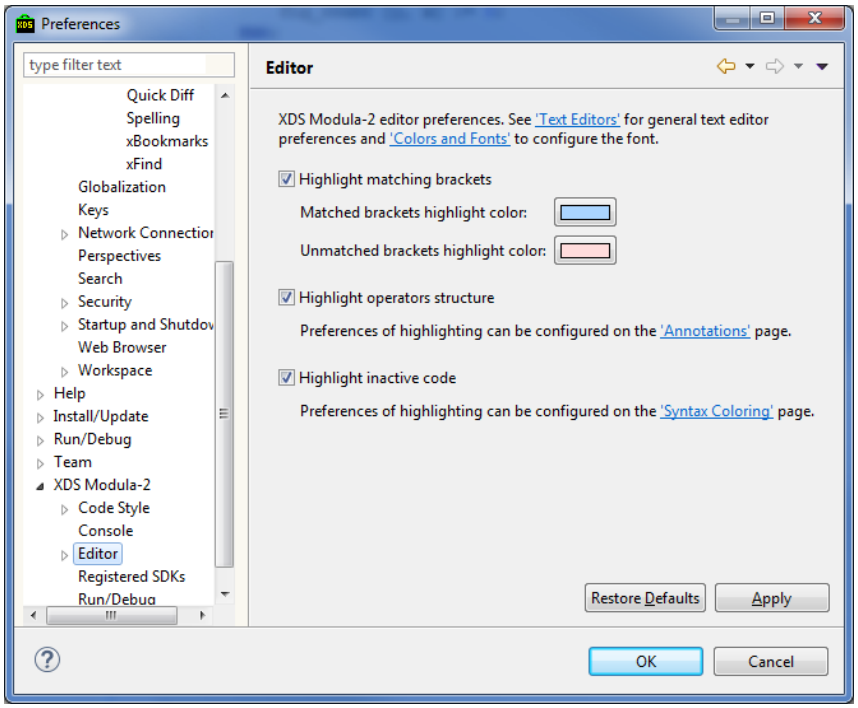
- **Error message** – error message color;
- **Warning message** – warning message color;
- **Error notification** – error notification color;
- **Information message** – ıпformation message color;
- **Output text** – standard output color;
- **Input text** – standard input color;
- **Background color** – background color.

To change color left click on the corresponding rectangle then select preferred color in the standard color selection dialog and press the **OK** color.

### 5.1.2 Editor

Modula-2 editor preferences are located in the **Preferences** dialog on the **XDS Modula-2** > **Editor** and on pages below:

- **Syntax Coloring** (**Syntax Coloring**) (see 5.1.2)
- **Templates** (**Templates**) (see 5.1.2)




By checking **Highlight matching brackets** it is then possible to specify what colors should be used to highlight matched and unmatched brackets (brackets, parenthesis and angle brackets) in the code. The colors being specified:

- for the matched brackets – **Matched brackets highlight color**;
- for the unmatched brackets – **Unmatched brackets highlight color**.

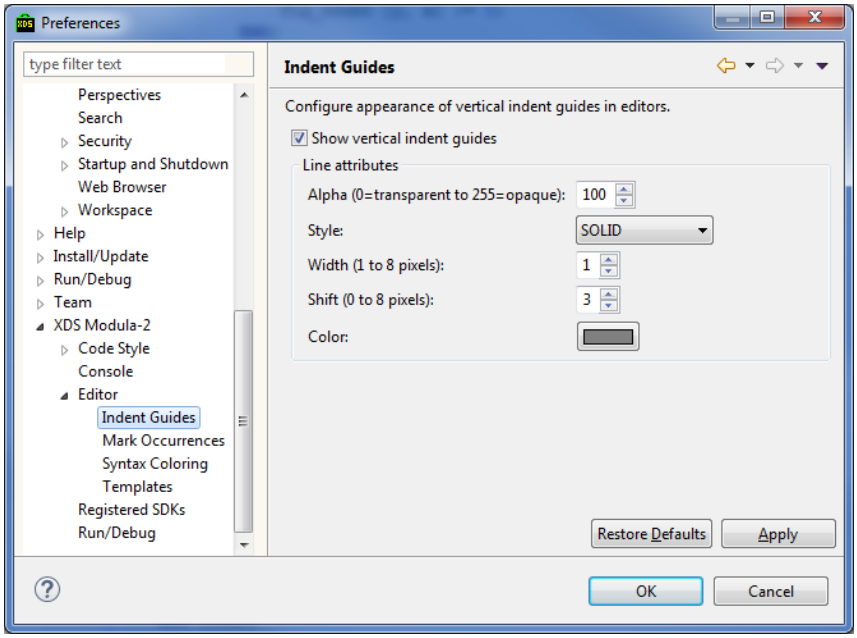
**Highlight operators structure** checkbox turns on token highlighting for the tokens belonging to the same language construct. Highlight color is specified in the **Operators structure highlight color**.

**Highlight inactive code** checkbox turns on highlighting of the code excluded by the compiler pragmas. Highlight color is specified on the **Syntax Coloring** (**Syntax Coloring**) (see 5.1.2) page.

To change the text editor font size go to the page **General** > **Appearance** > **Colors and Fonts** , select **Basic** > **Text Font** and press the **Edit...** button.

### Indent Guides

The page **XDS Modula-2** > **Editor** > **Indent Guides** contains preferences for the indent guide display.




Indent guides are vertical lines displaying the program code structure, see the picture:

```
IF (o.mode IN pc.PROCs) THEN
  IF ~(pc.omark_used_by_code IN o.marks) THEN
    IF ~(tmark_inlined IN o.type.marks) THEN
      WrnObj (o, 303);
    END;
    w.del := TRUE;
  END;
END;
EXCL (o.type.marks, tmark_inlined);
```

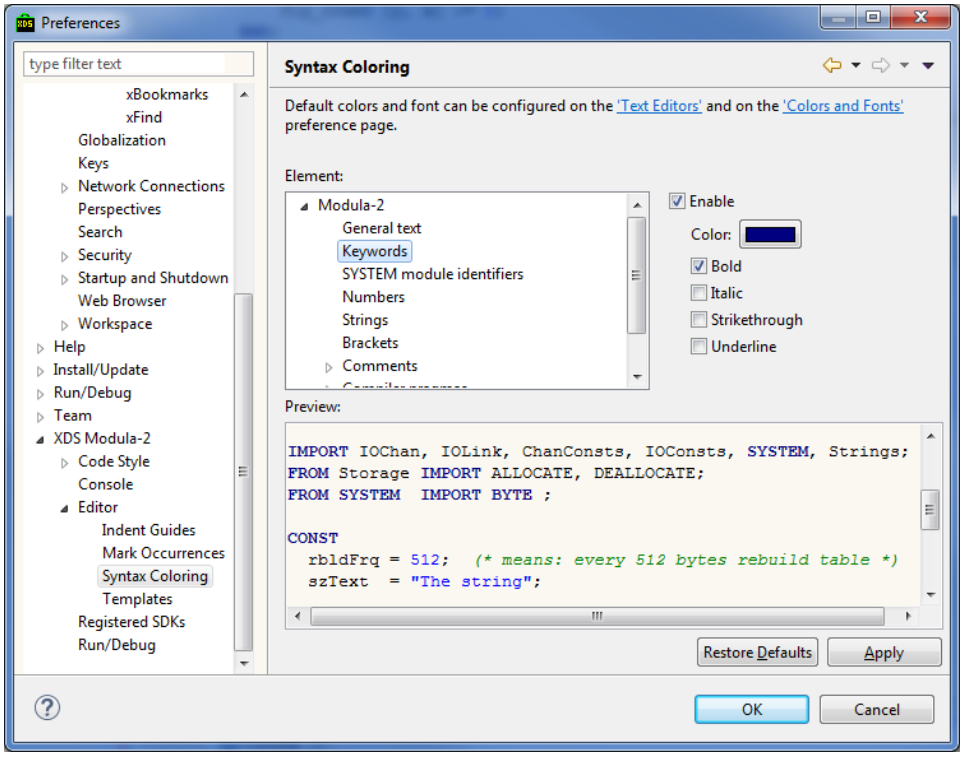
Indent guide display can be turned off by the checkbox **Show vertical indent guides**.

The following parameters can be specified:

- **Alpha** – value in [0..255] - line alpha parameter, where 0 - line is transparent, 255 - line is opaque;
- **Style** – line display style;
- **Width** – line thickness, [1..8] pixels;
- **Shift** – horizontal line displacement, [1..8] pixels;
- **Color** – line color;

### Syntax Coloring

The page **XDS Modula-2 > Editor > Syntax Coloring** allows to specify colors and stylings employed in Modula-2 syntax coloring.



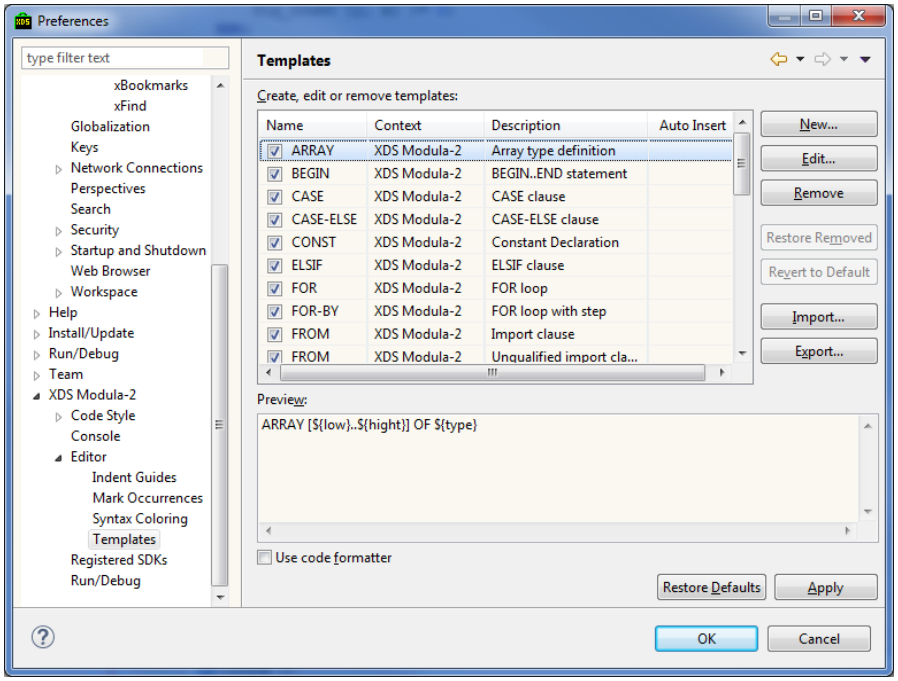

The page **Element** allows to specify the type for which the higlighting is performed. To the right is possible to enable **Enable** highlighting (if is not checked then the given highlighting is saved in the system but is not applied). Also, the following stylings can be specified:

- **Color** – element color;
- **Bold** – element is marked with bold if checked;
- **Italic** – element is marked with italic if checked;
- **Strikethrough** – element is strikethrough if checked;
- **Underline** – element is underlined if checked.

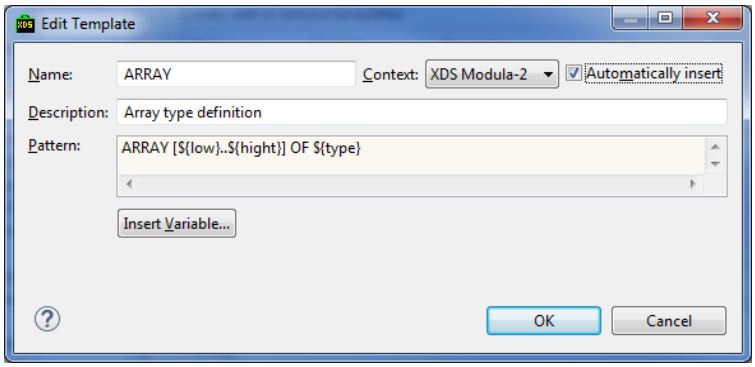
In the page **Preview** it is possible to see the preview of the code corresponding to the parameters specified.

### Templates

The section **XDS Modula-2** > **Editor** > **Templates** allows to setup the code templates.




To add the new template press **New...** button at the dialog right.




In the opened dialog specify the folllowing:

- **Name** – template name;
- **Context** – template usage context, selected in the dropdown list.
- **Automatically insert** – this template will be automatically inserted, when checked;
- **Description** – template description;

- **Pattern** – template pattern.

Last field also allows usage of so called variables. Variable list can be opened by pressing the button **Insert Variable...**.

Template is commited to the list by pressing the **OK** button.

To edit the template press the **Edit...** button. To delete the template press the **Remove** button. To revert the deleted template press the **Restore Removed** button. To reset the template list to defaults press the **Revert to Default** button.

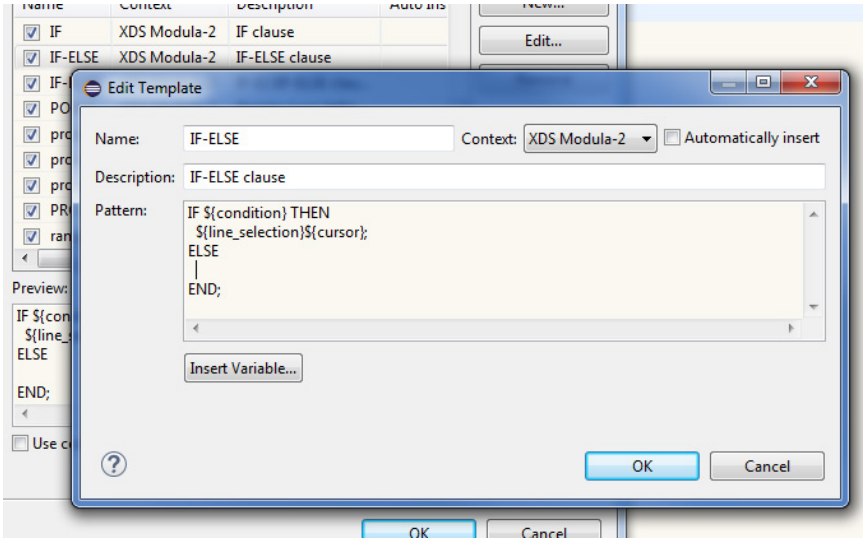
Templates can be imported from the `*.xml` file by pressing the **Import...** button. To export templates press the **Export...** button.

If the template is unchecked then it is saved but not applied (not active).

If the **Use code formatter** checkbox is checked then the template code is formatted after the application.

Lets modify the template using the IF-ELSE template as the example. Select IF-ELSE template and press **Edit...**.

Template editor will pop up:




Autocompletion template itself is the text with so called autocompletion variables. IDE replaces these variables in the moment of template insertion, or such variable can modify the editor behavior (demonstrated in the example below).

There is a number of pre-defined autocompletion variables (to see them press **Insert Variable...** button): `${cursor}`, `${date}`, `$$`, `${line_selection}`, `${time}`, `${user}`, `${word_selection}`, `${year}` .

Each autocompletion variable with the name different from the pre-defined variable name is the edit location (more on this below). When inserting template it is possible to traverse between edit locations using the `Tab` їкї `Shift + Tab` buttons.

Let us consider the ${cursor} variable.

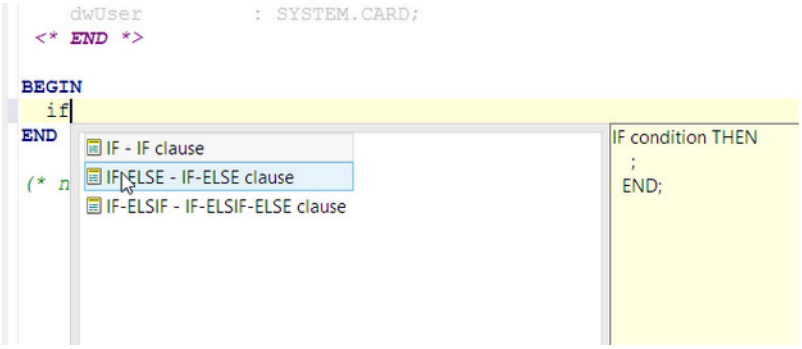
Modify the IF-ELSE template as the following:

```
IF ${condition} THEN
  ${line_selection}
ELSE
  ${cursor};
END;
```

Press **OK**, **Apply** and close the template editor.

Open the Modula-2 editor on some mod-file, go to the procedure body, start typing the `if` and press `Ctrl + Space`.

The following window appears:

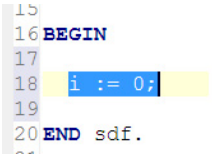



Cursor will be placed near condition, so if type `TRUE`, condition will be replaced with `TRUE`.

Try pressing the `Tab` button - the cursor will traverse between edit locations, in the current example this happens to be lines 17 and 20. Line 20 corresponds to ${cursor} variable edit location.

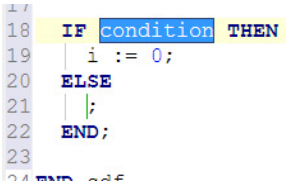
Remove the inserted text.

Now type `i := 0;` and select this text:




Press `Ctrl + Space` and select `IF-ELSE` in the autocompletion list.

The following text will be inserted:



So the current selection was used as the ${line_selection} variable value. Edit the condition location and press `Enter`. Cursor will jump to edit location corresponding to the ${cursor} variable. This variable specifies the position where the cursor should be placed after completion of the autocompletion session.

Now edit IF-ELSE template as the following:

```
IF ${condition} THEN (* ${THEN\_comment} *)
  ${line_selection}
ELSE  (* ${ELSE\_comment} *)
  ${cursor};
END; (* ${END\_comment} *)
```

Go to the source code editor and invoke the `IF-ELSE` template:

```
IF condition THEN (* комментарий для THEN *)

ELSE (* комментарий для ELSE *)
  ;
END; (* комментарий для END *)
```

Since variables ${THEN_comment}, ${ELSE_comment} and ${END_comment} are not built-in they only specify edit locations. During the autocompletion session it is possible to traverse them using the `Tab` and `Shift + Tab` buttons.

Built-in autocompletion variables:

- `${cursor}` – cursor is placed to this location after autcompletion session ends;
- `${date}` – current date;
- `$$` – dollar sign;
- `${line_selection}` – replaced with the current text selection;
- `${time}` – current time;
- `${user}` – user name;
- `${year}` – current year.

### 5.1.3 Registered SDKs

Section **XDS Modula-2** > **Registered SDKs** is used to specify new or edit existing SDKs.

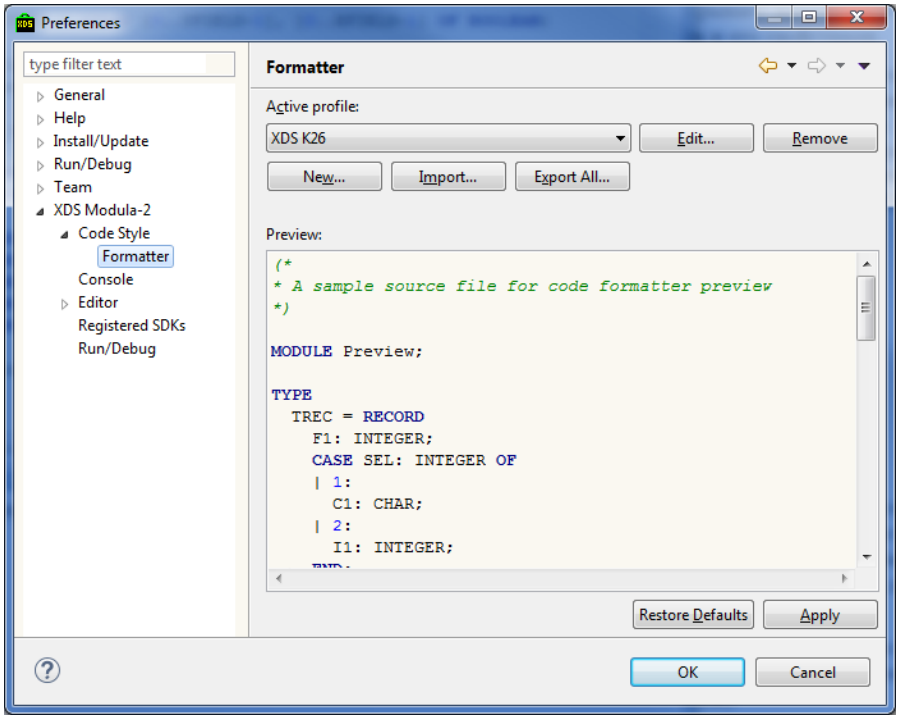
More on SDK setup please see SDK configuration (see 7.2).

### 5.1.4 Code style

At the moment the section **XDS Modula-2** > **Code style** contains only formatting settings.

### 5.1.5 Formatter

The section **XDS Modula-2** > **Code style** > **Formatter** is used to specify source code formatting settings.




All formattings settings are saved in the formatting profile. Initially built-in formatting profile is used - it is possible to inspect its settings but it is not possible to edit them. Listbox atop allows to activate existing profile. The buttons are as follows:

- **Edit** – edit formatting profile. Formatting profile edit dialog (see **??**) will open.
- **Remove** – delete profile.
- **New...** – create new profile. Dialog will open - it allows to specify profile name, also it is possible to select profile to get initial settings from.
- **Import...** – import profile from the file. Select .xml file containing profiles, they will be imported. If some profile names are the same as already presented they will be redefined with the imported.
- **Export All...** – export all profiles into the file.

In the dialog below there is a window **Preview** allowing to preview formatting on the given source code sample.

### Formatting profile edit dialog

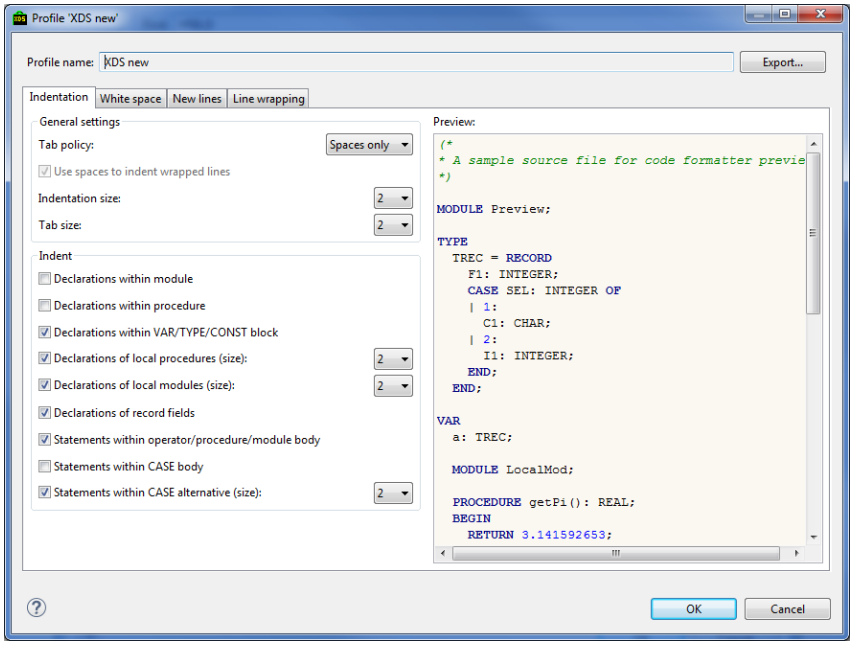
Atop of the dialog there is a **Export...** button allowing export of the profile to the .xml file.

The tabs below specify formatting parameter groups:

- **Indentation** – Indent formatting settings (see **??**)
- **White space** – Whitespace formatting settings (see **??**)
- **New lines** – Newline formatting settings (see **??**)
- **Line wrapping** – Line wrap formatting settings (see **??**)

Each tab contains **Preview** window allowing preview effect of the settings being changed.

### Indent formatting settings



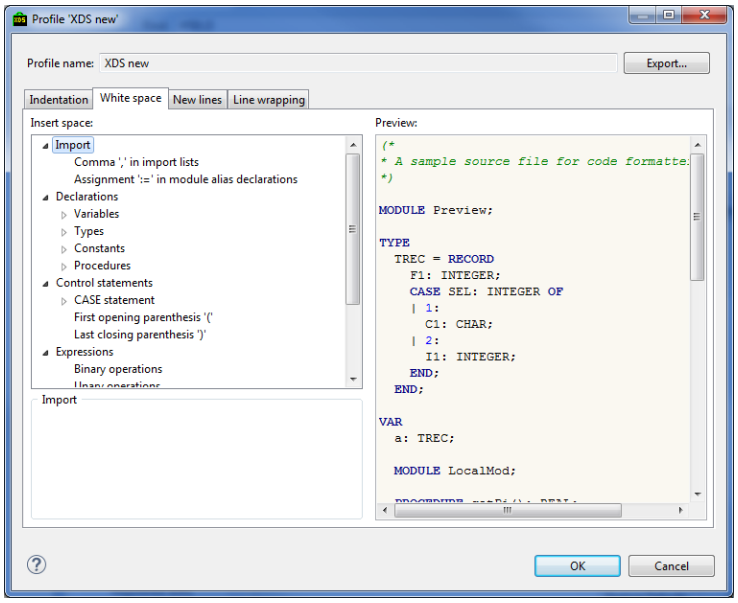

- **Tab policy** - this list specifies tab usage mode.
- **Indentation size** - indentation size in characters.
- **Tab size** - tab width.

The **Indent** section allows to override **Indentation size** for the given elements:

- **Declarations within module**

- **Declarations within procedure**
- **Declarations within VAR/TYPE/CONST block**
- **Declarations of local procedures**
- **Declarations of local modules**
- **Declarations of record fields**
- **Statements within operator/procedure/module body**
- **Statements within CASE body**
- **Statements within CASE alternative**

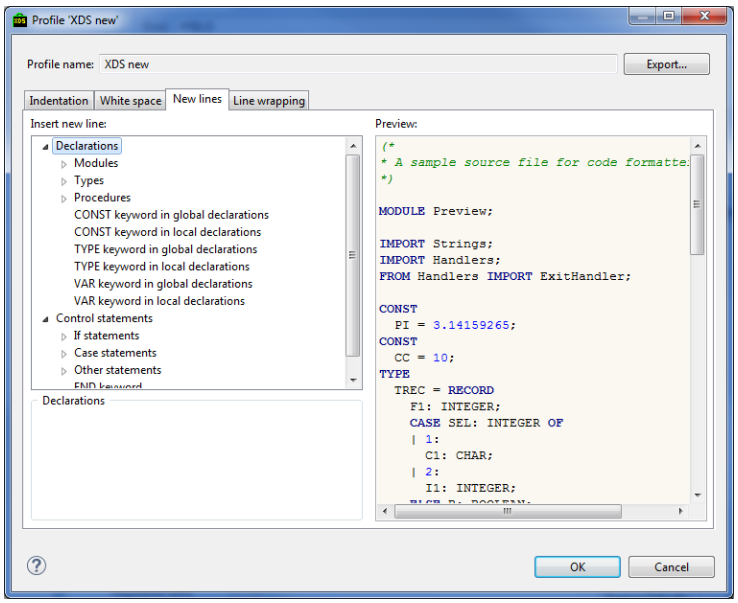
### Whitespace formatting settings




**Insert space** tree allows to select varios language elements. It is possible to specify for the selected element whether to put space after or before this element.

When two elements are adjacent - space will be inserted if either of them confirm this.

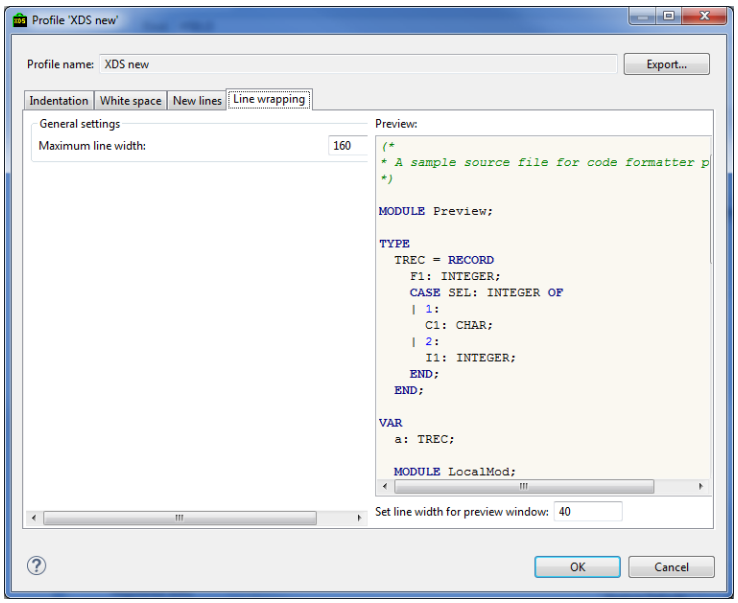
### Newline formatting settings




**Insert new line** tree allows to select varios language elements. It is possible to specify for the selected element whether to put newline after or before this element.

When two elements are adjacent - newline will be inserted if either of them confirm this.

### Line wrap formatting settings




By line wrapping we mean reorganizing long strings in a way that newline or / and whitespace padding is inserted to fit the string in the editor.

One can specify **Maximal line width** value - position in the string which will be used by the formatter to fit the text in. It is not always possible to do this without broking the program syntax so sometimes strings are extended beyond this border.

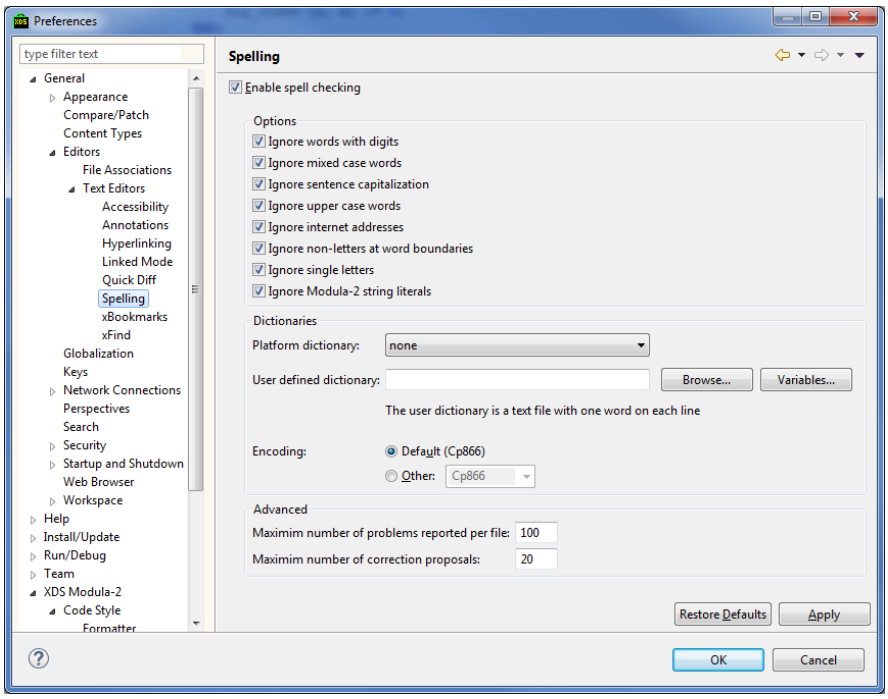
Below the preview window the value **Set line width for preview window** specifies the text wrapping width in the preview window - by changing this value one can guess how the text will appear after the reorganizing.

## 5.2 General

General Eclipse platform settings. XDS Modula-2 IDE uses some of them and besides adds some of its own preferences.

### 5.2.1 Spelling

Spellchecking preferences can be found in the **Preferences** dialog at the **General** > **Editors** > **Text Editors** > **Spelling** page.




**Enable spell checking** checkbox allows to turn on the spell checking.

In the combobox below it is possible to select the currently active spell checking service. Here we describe the **XDS spell checking** settings.

The settings below affect the spell checking:

- **Ignore words with digits**

- **Ignore mixed case words**
- **Ignore sentence capitalization**
- **Ignore upper case words**
- **Ignore internet addresses**
- **Ignore non-letters at word boundaries**
- **Ignore single letters**
- **Ignore Modula-2 string literals**

The section **Dictionaries** allows to select dictionary among the registered or select the user dictionary – in the later case the specially formatted file is expected. The format is as follows : each line contains the single word. IDE should have the permissions for both reading and writing the file. By selecting the file user should also specify the file encoding.

In the dialog bottom is also possible to set maximum number of errors in the single source file and constrain the maximum count of spelling corrections offered to user.

# Chapter 6

# Resource properties

Each IDE resource (see 3.3) is associated with the set of properties. Particular set of properties depends on the corresponding resource type and project type (see **??**) it belongs to.

To view or edit resource properties use the **Properties** dialog. To open it for some resource:

1. Right click some resource in the **Project Explorer** (**Project Explorer**) (see 4.4.1) view.
2. Select the **Properties** item in the context menu.

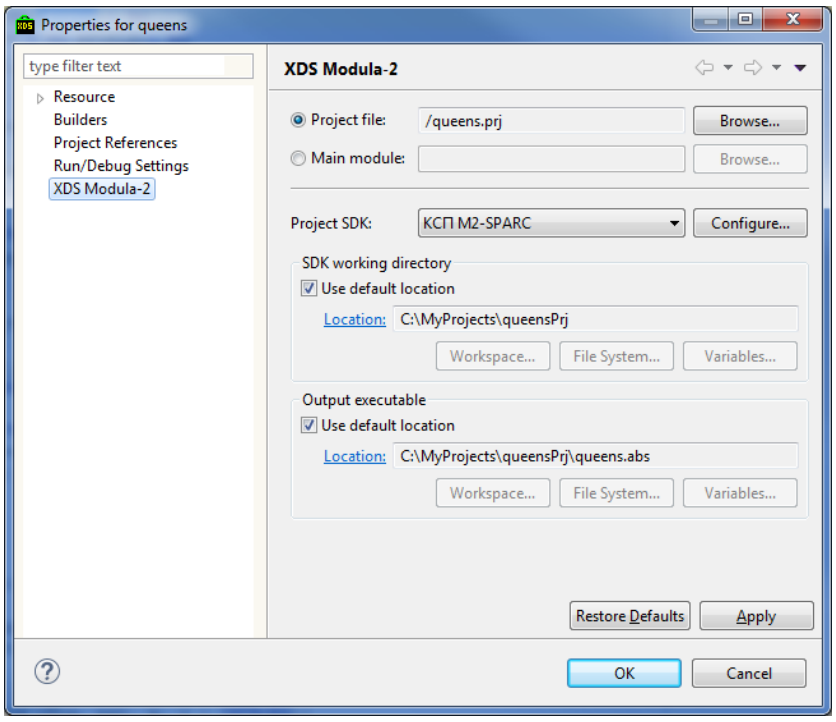
It is also possible to open **Properties** dialog using the `Alt + Enter` hotkey combination.

Properties are divided into groups that are displayed in different dialog pages. This pages are in turn are organized into the hierarchial structure displayed in the left pane of the dialog. Atop the dialog filter field can be used to quickly navigate to the property page.

Select **Window** > **Show View** > **Other...** > **General** > **Properties**, to open the Property view. This view shows main properties of the resource selected in the Navigator view such as the Project Explorer view.

## 6.1 Modula-2 project properties

To view and edit Modula-2 project specific properties use **XDS Modula-2** page of the **Properties** (**Properties**) (see 6) dialog.




You can modify the following Modula-2 project properties:

- SDK used by the project,
- Main module or prj file used for the compilation,
- working directory used for the compilation.

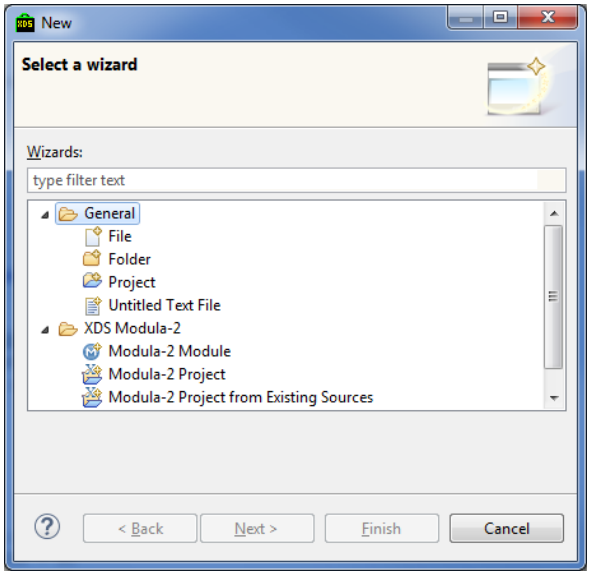
# Chapter 7

# Usage

This chapter describes main usages of the XDS Modula-2 IDE.

## 7.1 Create new elements

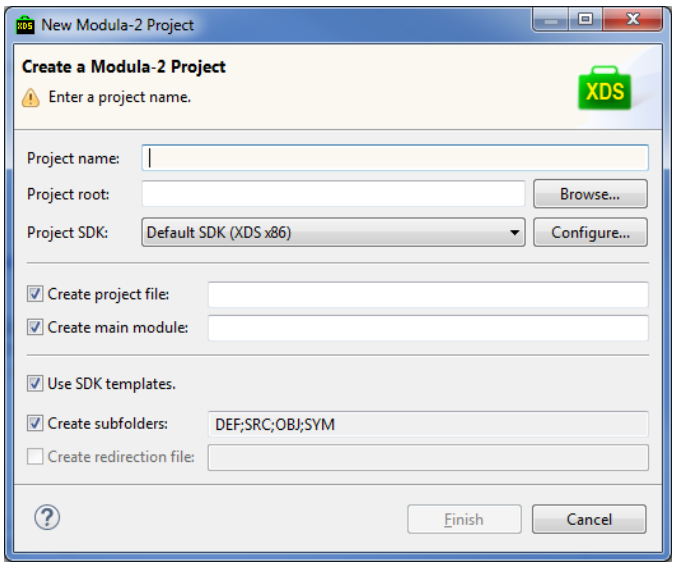
Select **File** > **New** > **Other...** to create new resource. Select the appropriate wizard and press **Next**:




Provide all necessary parameters and **Finish**. The new resource will be created.

### 7.1.1 New project from scratch

Select **File** > **New** > **Modula-2 Project** in the main menu, and provide necessary parameters:




Specify **Project Name**. Next, provide location of the project root directory (**Project root** field).

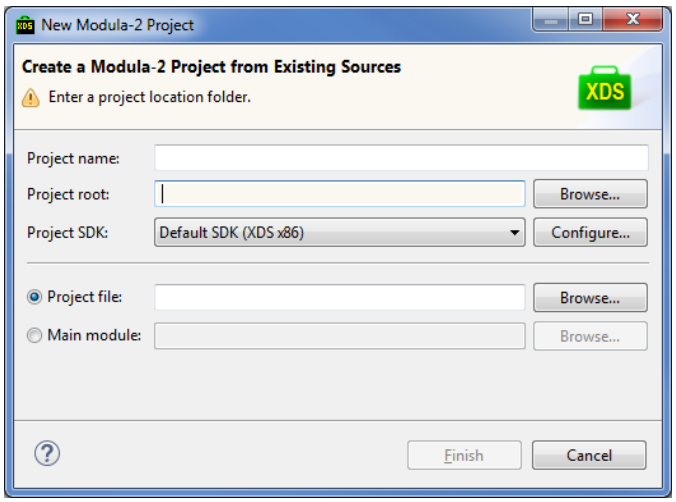
Select project SDK (**Project SDK** field). Use **Configure...** button to edit SDK settings (see 7.2.1).

One can create main module or project file, standard directory structure and redirection file. If **Use SDK templates** is checked then SDK templates will be used to create the files. Directory structure and redirection file name are also taken from the selected SDK settings.

Pressing the **Finish** button will create the new Modula-2 project and add it to the workspace. Newly created files and directories will show up in the Project Explorer.

### 7.1.2 Create project from existing sources

To create project from existing sources select **File** > **New** > **Modula-2 Project from Existing Sources** in the main menu. Specify project settings in the wizard opened.




You can explicitly specify the new project name or it will be set automatically after the project root directory selection (**Project root** field).

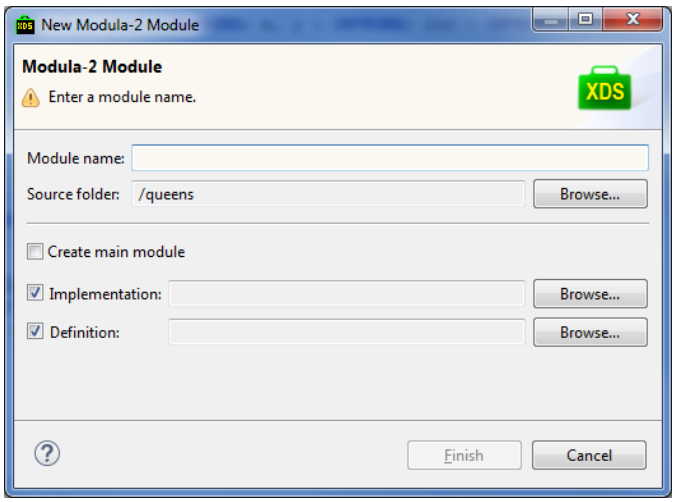
Select the project SDK (**Project SDK** field): Unless otherwise indicated the default SDK is used. The **Configure...** button will open the SDK settings dialog.

Specify whether project file (**Project File** field) or main module (**Main module**) is used for the source build. One can select project file `*.prj` or main module file (`*.mod` or `*.ob2`) by pressing **Browse...** button.

Pressing the **Finish** button will create the new Modula-2 project and add it to the workspace. Newly created files and directories will show up in the Project Explorer.

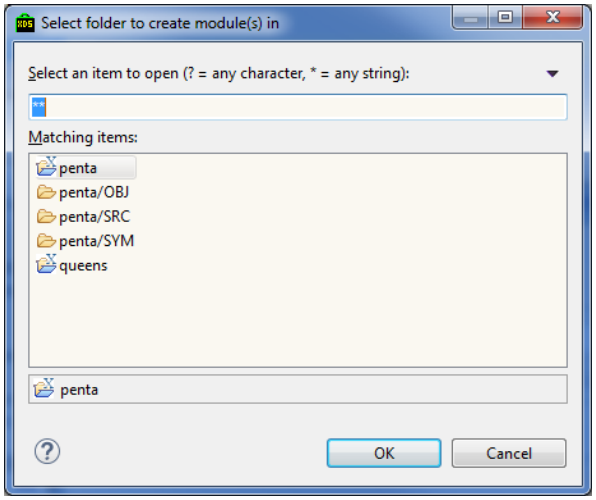
### 7.1.3 New Modula-2 module

To create the new Modula-2 module select **File** > **New** > **Other...** in the main menu or use `Ctrl + N` hotkey. Select **Modula-2 Module** wizard under the **XDS Modula-2** category (you can type `module` in the filter field to narrow the search). In the opened wizard specify module parameters:




Specify new module name (**Module name** field).

Specify folder to store the module that will be created (**Source folder** field). The **Browse...** button allows to select the directory from the available folders in workspace projects.




Text field atop allows to filter resources by name. Wildcards are supported : `*` stands for the arbitrary string, `?` stands for the single character or emptry string.

Check the **Create main module** checkbox to promote the newly created module as the main project module. In this case it is necessary to specify implementation module (**Implementation** field).

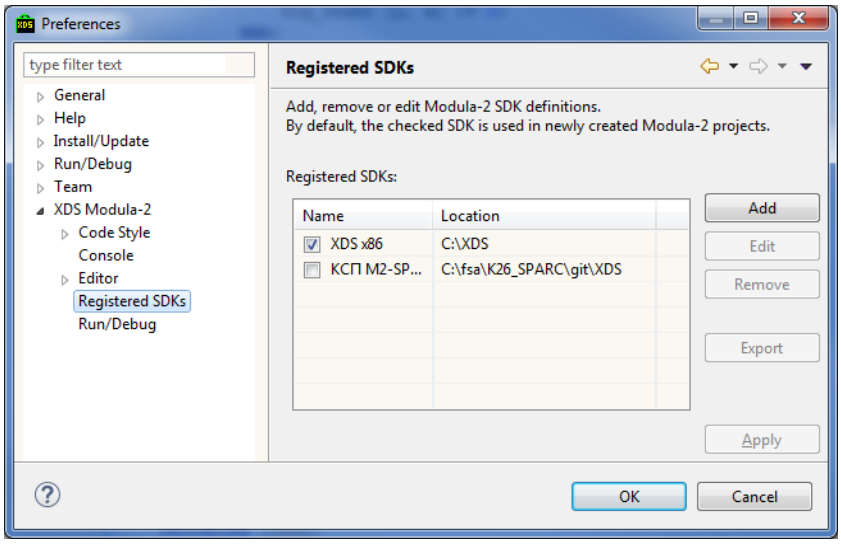
For the ordinary (not main) module one can set up the implementation and definition files – the **Implementation** field and the **Definition** field. The **Browse...** button allows to select module location directory via the standard Eclipse Folder Selection dialog.

Pressing the **Finish** button in the wizard creates the module resource and add

it to the workspace. Newly created files will show up in the Project Explorer.

## 7.2 Edit SDK

Add or edit SDK under the **XDS Modula-2** > **Registered SDKs** preference page.




Use **Add** button to select the folder where SDK is installed to register the SDK.

When the selected SDK is configured to be used with the IDE (i.e. SDK root folder contains the `sdk.ini`) then the installed SDK list will show the new added SDK record.

If the `sdk.ini` file is absent then manual SDK configuration wizard will show up. One can always open the SDK configuration wizard by selecting the SDK in the list and pressing the **Edit** button.

To remove SDK use the **Remove** button.

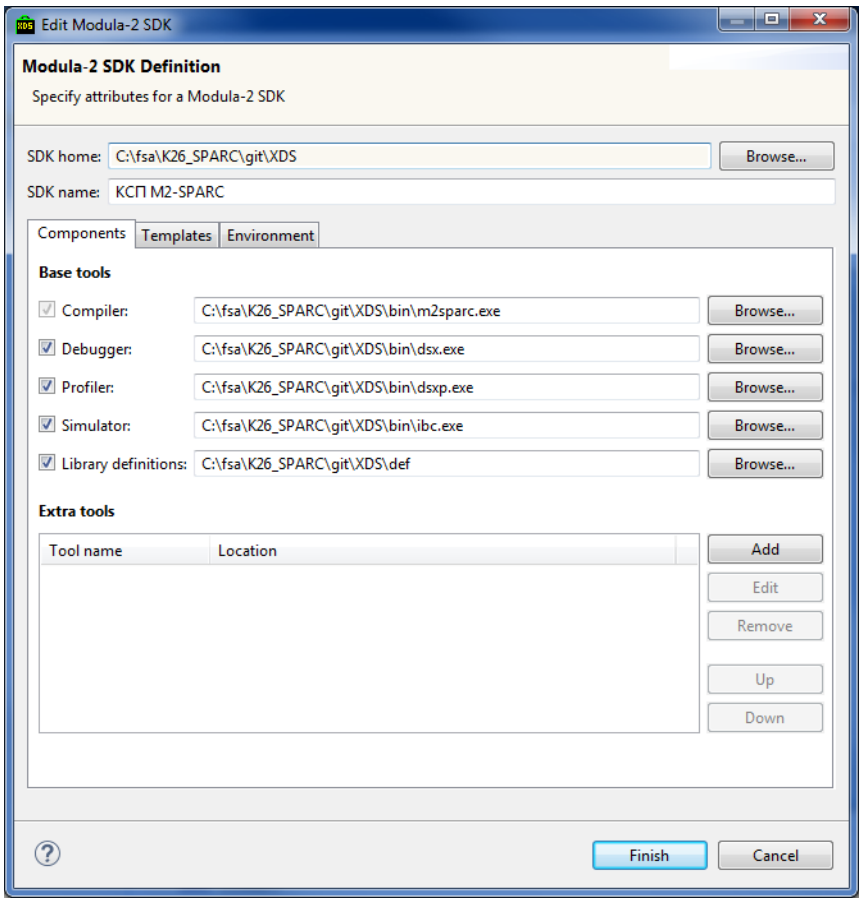
To export selected SDK settings (`*.ini` file) press the **Export** button, select and select folder to export.

To apply and commit SDK changes press the **Apply** button.

To edit the particular SDK settings use the Edit SDK wizard (see 7.2.1).

### 7.2.1 Edit SDK wizard

Edit SDK wizard is depicted on the picture below:




In the top part of the wizard fields **XDS home** and **XDS name** used to set up SDK root folder and SDK name.

Another settings are accessible from the following tabs: **Components**, **Templates**, **Environment**.

### Components

#### Base tools

The **Base tools** section allow to specify main tools, such as:

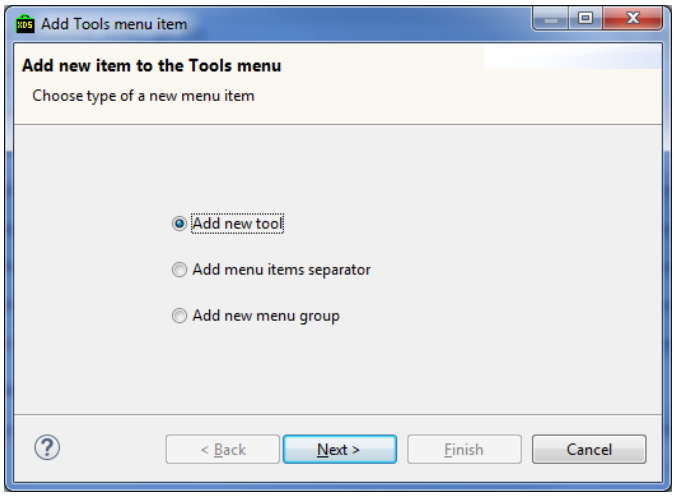
- compiler – **Compiler** field;
- debugger – **Debugger** field;
- simulator – **Simulator** field;
- library definitions – **Library definitions** field.

The first three fields should specify either executable file path (`*.exe` or `*.bat`). Last field should specify path to library definitions folder containing definition

files `*.def`. The only mandatory field of these is the first one, containing compiler path.

### Extra tools

The **Extra tools** section allow to specify extra tools. These extra tools can be invoked via the Project Explorer context menu. It is possible to build the context menu structure.




Here you can specify:

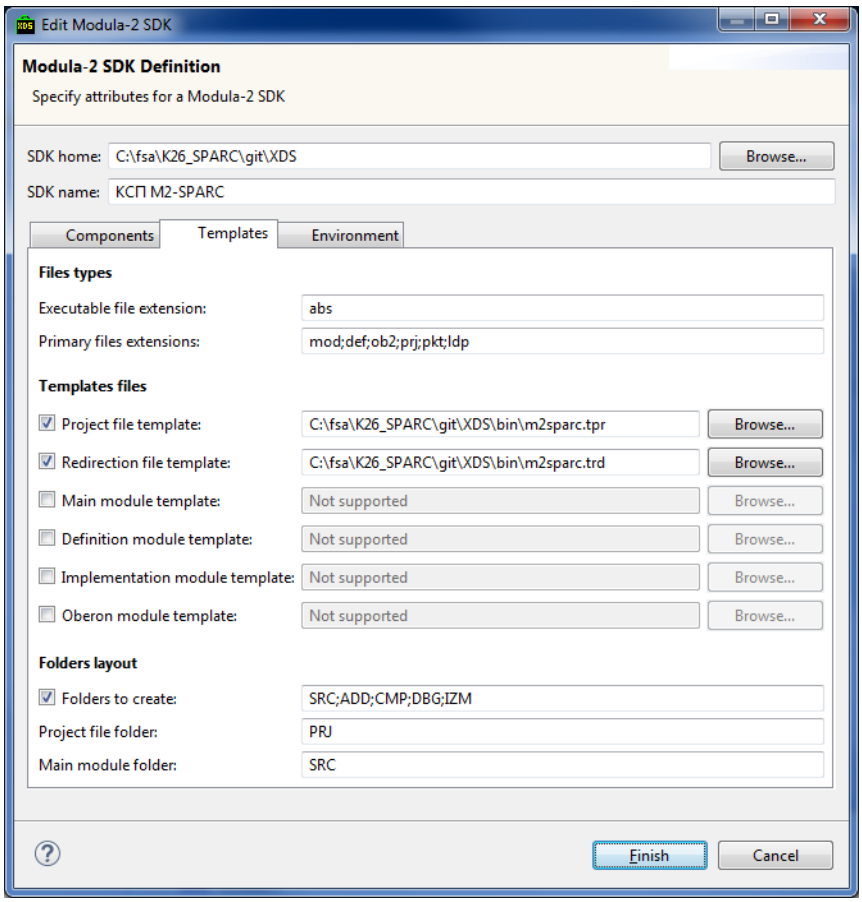
- **Add new Tool** – adds a new tool.
- **Add menu items separator** – adds a horizontal line between menu items.
- **Add new menu group** – adds a menu item group.

To modify the menu items ordering use **Up** and **Down** buttons.

To reassign the instrument to the another group edit this instrument and change its group to another.

### Templates

The **Templates** tab specifies file types and templates to create files to create files of these types.




### Files types

The **Files types** tab specifies two parameters:

- executable file extension – **Executable file extension** field.
- Comma-separated primary files extensions – **Primary files extensions** field.

### Templates files

The **Templates files** section specifies templates to create files.

- Project file template – **Project file template** field, `*.tpr` file;
- Redirection file template – **Redirection file template** field, `*.trd` file;
- Main module template – **Main module template** field, `*.tmd` file;
- Definition module template – **Definition module template** field, `*.tmd` file;

- иґґкнґ hкп дґқкґ ажґкїльґжїї – **Implementation module template** field, `*.tmd` file;
- Oberon module template – **Oberon module template** field, `*.tmd` file.

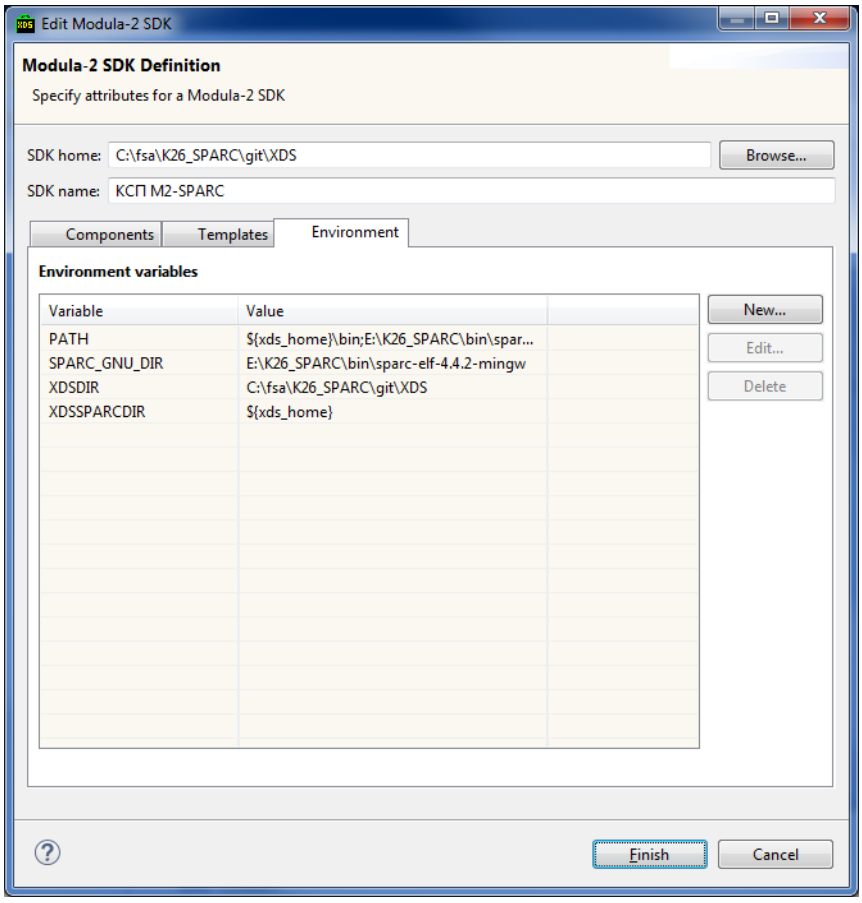
### Folders layout

The **Folders layout** section specifies directory names that will be created for the new project:

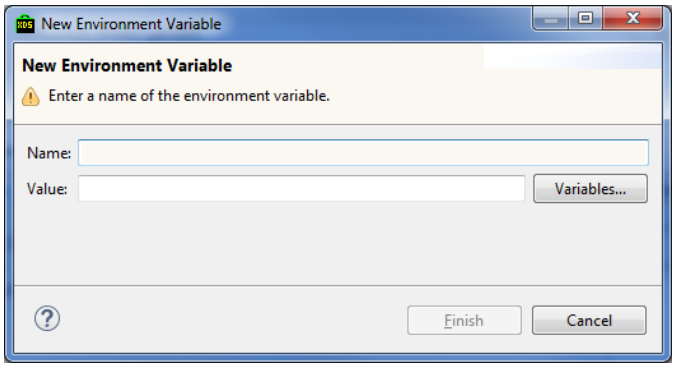
- **Folders to create** - comma-separated list of directories that will be created for the new project;
- **Project file folder** - if set project file will be put into the specified directory;
- **Main module folder** - if set main module will be put into the specified directory;

### Environment

The **Environment** specifies environment variables.



To add the new variable use the **New...** button. In the opened wizard set the variable properties and press the **Finish** button. To edit or delete the variable press the **Edit...** button or the **Delete** button.



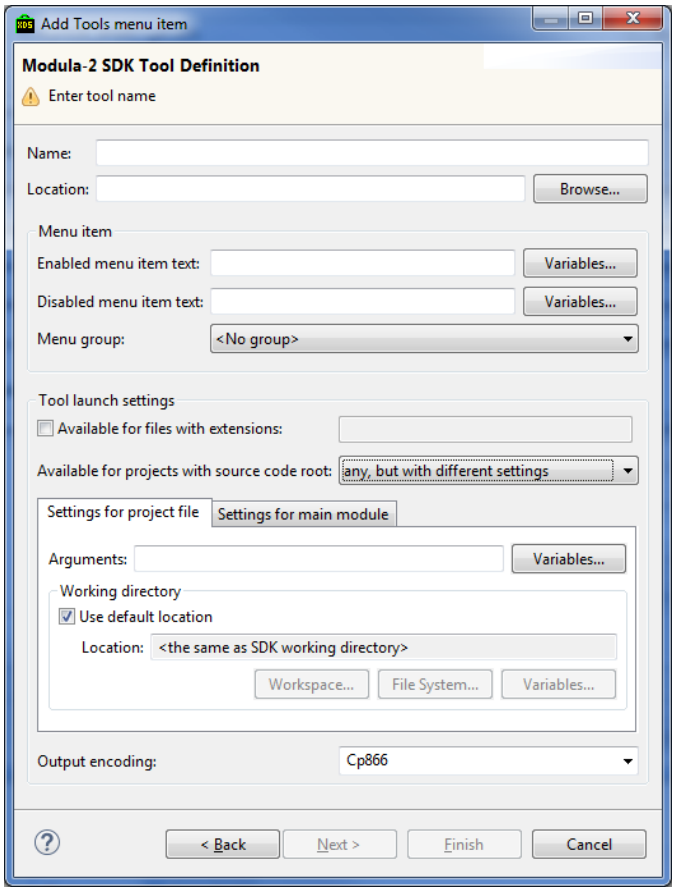

Variable has the following properties:

- variable name – **Name** field;
- variable value – **Value** field.

Only the name field is mandatory. When specifying the variable value one can use predefined IDE variables – these can be selected by pressing the **Variables...** button.

### 7.2.2 Edit tool dialog

#### Tool settings




For the extra tools it is necessary to specify the following settings:

- instrument name– **Name** field;
- instrument location – **Location**, this is the path to (`*.exe` or `*.bat`);

These fields are mandatory.

#### Tool menu

The **Menu item text** section can set up the menu for the Extra tool. It is possible to set the menu name for the enabled menu (**Enabled menu item** field) and the menu name for the disabled menu (**Disabled menu item** field). If the fields are not filled then instument name will be used.

Tool menu is the submenu for the **Tools** menu item in the IDE main menu or context menu.

#### Tool launch settings

The **Tool launch settings** section specifies the tools launch settings.

When **Available for files with extensions** checkbox is set then tool is applied only to the files with the specified extensions: extensions should be separated with comma.

Ъ sнґкж **Available for projects with source code root** ндҗнґеңhїæң ђлґаґвм, hкп ґкґкїе вїsнђ saнжҗквґ hнбвґsжң їңбвагæжңв. Ънљæнžзңл бкжҗhгойїїж ђґаїґңвл:

The field **Available for projects with source code root** specifies project types for which the tool is enabled.

- **any** – any project type;
- **any, but with different settings** – any project type with special settings;
- **project file (*.prj)** – project file based projects `*.prj`;
- **main module** – main module based projects.

After selecting the project type it is possible to specify launch parameters and working directory.

The **Output encoding** field specifies standard output (and standard error) encoding.

#### Arguments

To specify arguments it is possible to use predefined IDE variables – these can be selected by pressing the **Variables...** button. More on this please see Working with variables (see 7.5)

#### Working directory

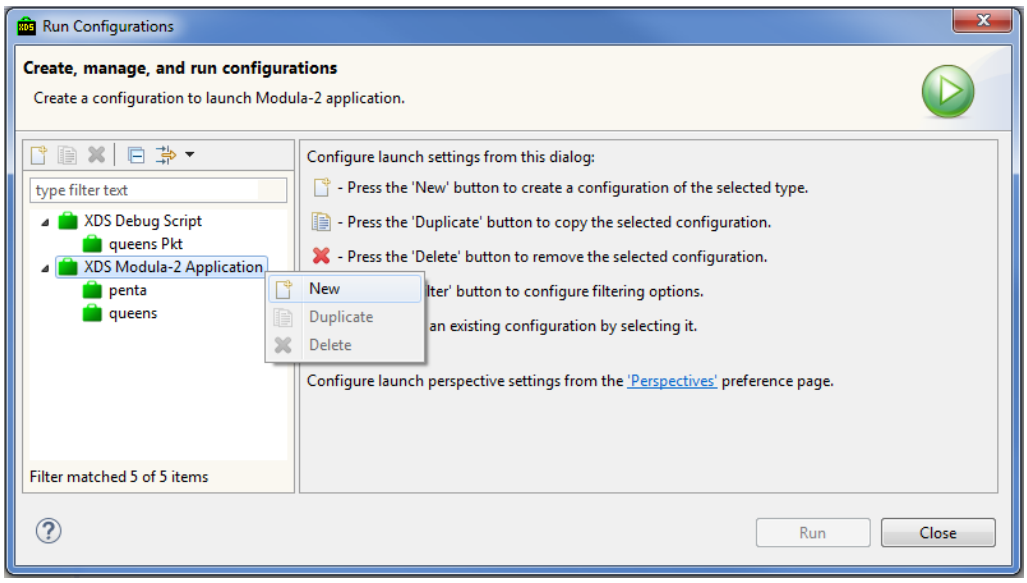
The working directory is set up in the **Working directory** section. To use the SDK working directory – check the **Use default location** checkbox or manualy specify in the **Location** field (see Selecting working directory (see 7.4)).

## 7.3 Launch configurations

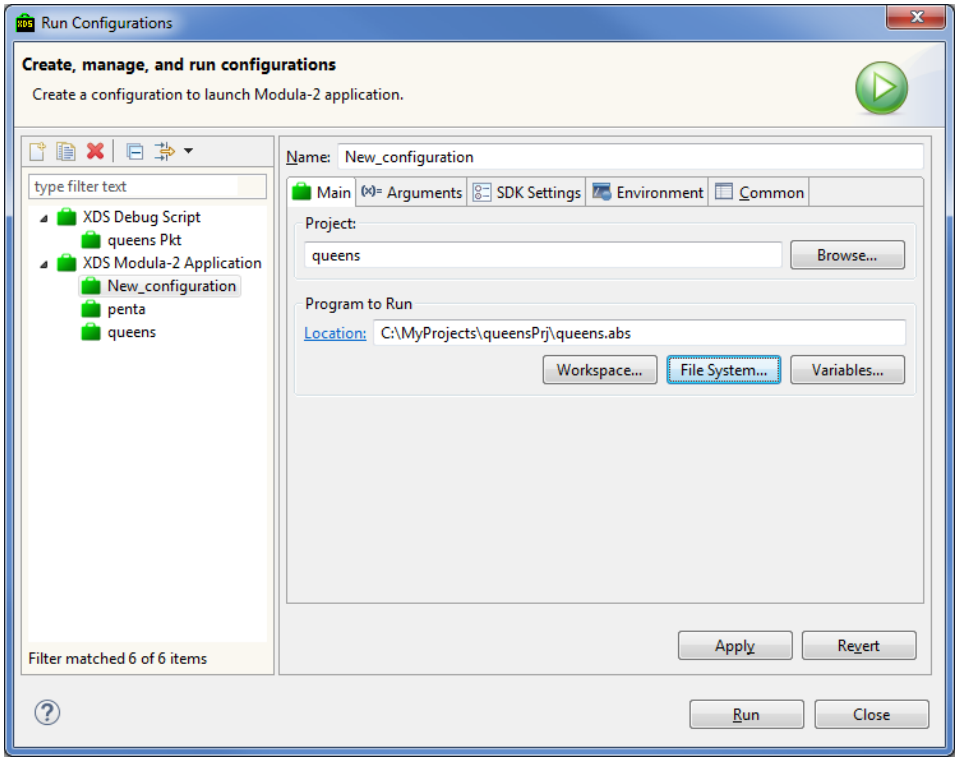
### 7.3.1 Edit launch configurations

To create or edit launch configuration select **Run** > **Run Configurations...** or **Run** > **Debug Configurations...** in the main menu.

In the opened dialog select **XDS Modula-2 Application** in the left pane, open context menu with the right mouse click and select **New**.




To edit the existing launch configuration select it in the left pane - the right pane will display its properties.




Type run configuration name in the **Name** field.

Select target workspace project for this run configuration and executable file

for the launch. **Apply** button saves changes for the edited run configuration, **Revert** button reverts run configuration parameters to the last saved state.

Next we will describe tabs used when creating or editing the run configuration.

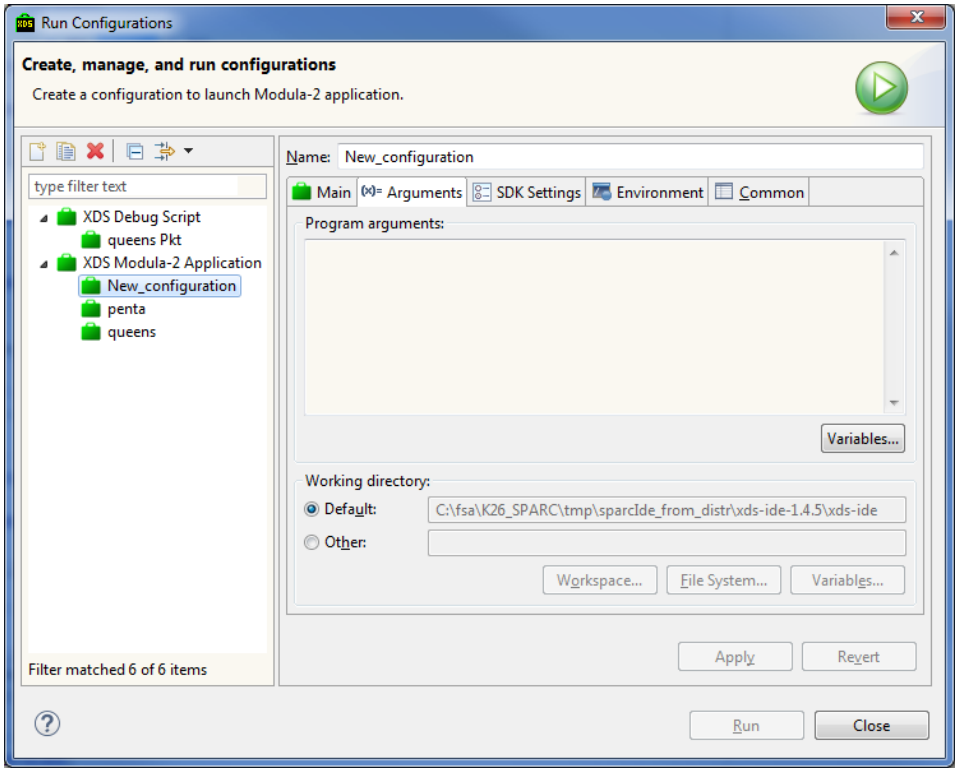
### Main Tab

Select the project for the run configuration on the **Main** tab – fill in the **Project** field. Project is selected from the current workspace it is possible to type the project name manually or to use the project selection dialog.

Next specify the executable file for run – it is done under the **Program to Run** section (see the Select folder or file (see 7.4) section).

### Arguments Tab

The **Arguments** tab allows to specify program arguments and working directory for the launched program.



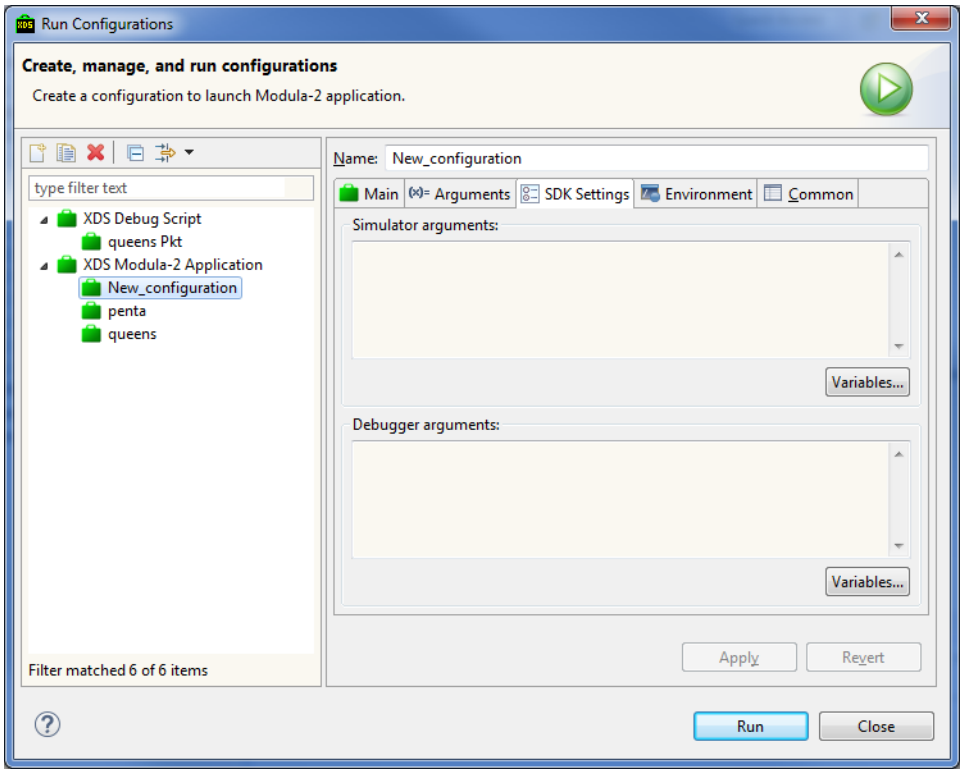

Specify program arguments under the **Program arguments** section. It is possible to use predefined IDE variables to specify arguments. To do this press the **Variables...** button. More on predefined IDE variables see Variables... (see 7.5).

Under the **Working directory** it is possible to specify whether to use project folder – **Default** item, or any other folder – **Other** folder. See (Select file or

folder (see 7.4)).

### SDK Settings Tab

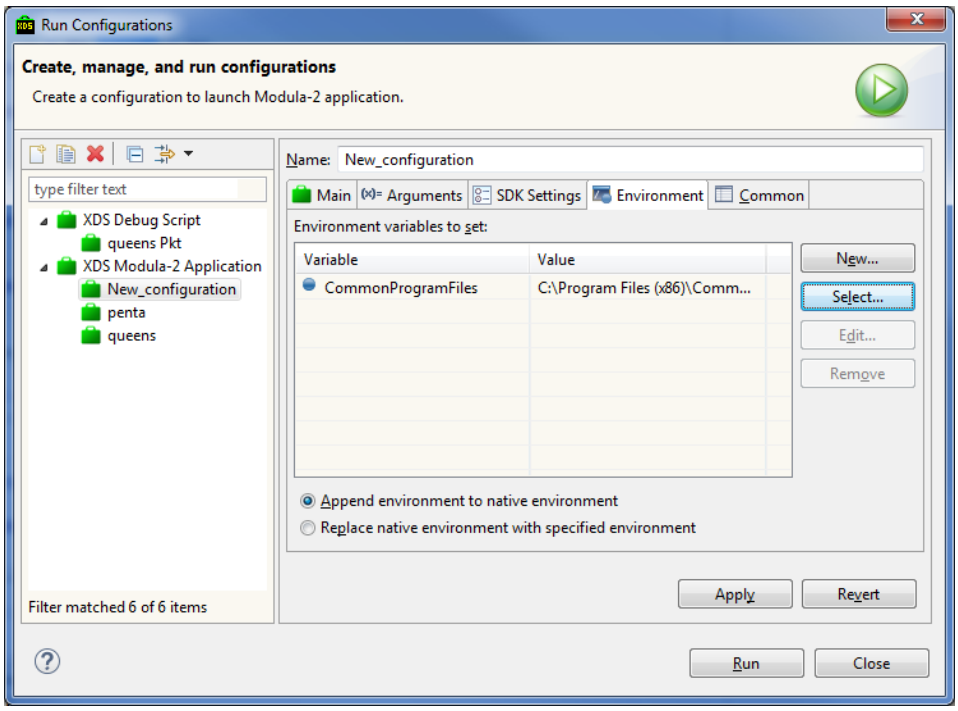
The **SDK Settings** tab allows to specify arguments for simulator and debugger – in the sections **Simulator arguments** and **Debugger arguments** correspondingly.



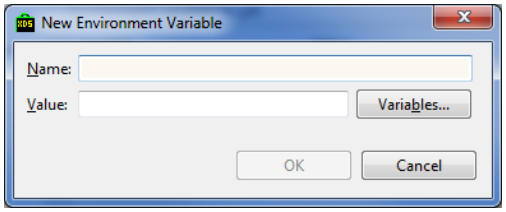

It is possible to use predefined IDE variables. More on this see Working with variables (see 7.5)

### Environment

The **Environment** allow to set environment variables.



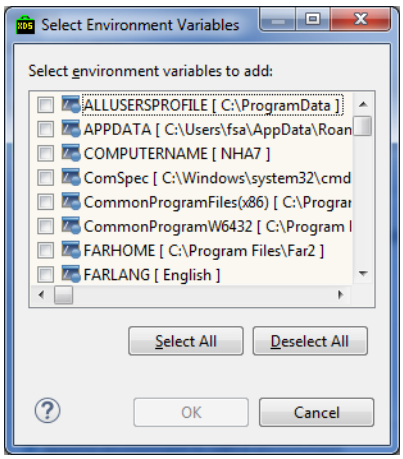

To add new predefined IDE variable press **New...** at the right.




In the opened dialog specify variable name (**Name** field - mandatory) and variable value (**Value** field - optional). Variable value can be typed manually or predefined variables can be used (use **Variables...** button).

Press **Edit...** button to edit the variable.

It is possible to add variable from the list. Press **Select...** button to open this list.




Select required variables in the list. Selected variables will be added to predefined IDE variables.

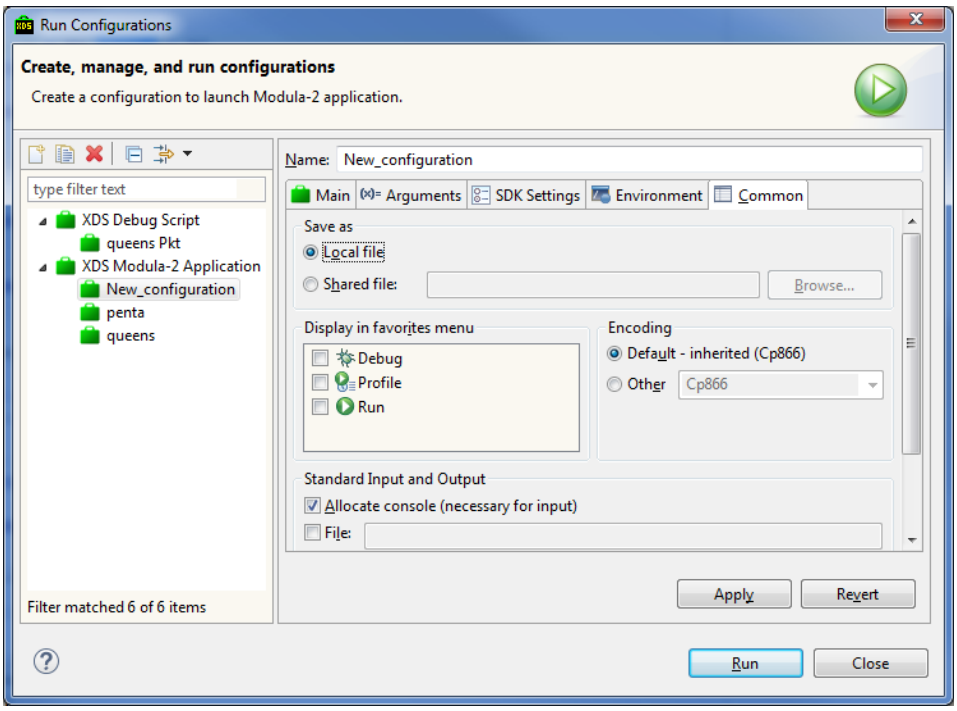
To delete environment variable select it in the list and press **Remove**.

If there is at least one variable choice is enabled:

- **Append environment to native environment** – add selected variables to system variables;
- **Replace native environment with specified environment** – replace system variables with selected variables.

### Common Tab

The **Common** tab allow to specify general launch configuration settings.




The **Save as** section allows to set whether to save launch configuration as local file (**Local file** item) or as shared file (**Shared file** item). In the latter case it is necessary to specify directory to save the launch configuration.

The **Display in favorites menu** section allows to specify whether to show launch configuration in the Favorites dropdown at Run/Debug menu.

Select encoding in the **Encoding** section: default (**Default** item) or specific (**Other** item).

The **Standard Input and Output** section allows to specify input/output integration parameters.

One can use:

- allocated console - **Allocate console**;
- external file - **File** (see Selecting file or folder (see Selecting file or folder7.4)).

The file will be appended if **Append** checkbox is checked.

## 7.4 Select folder or file

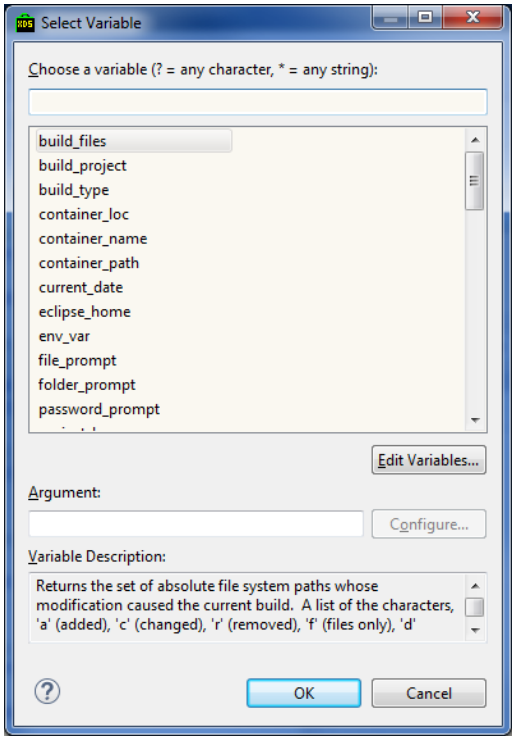
File or folder selection is a common task in various wizards and dialogs. It is possible to fill the **Location** field manually or using buttons:

- **Workspace...** to select from workspace resources;
- **File System...** to select the file or folder from the file system;
- **Variables...** to select the variable (more on variables Working with variables (see 7.5)).

After the field is filled the **Location** field becomes the hyperlink – double click on it will open native operating system explorer.

## 7.5 Working with variables

IDE predefined variables are often used in various wizards and dialogs. To open the variable list click on the **Variables...** button.



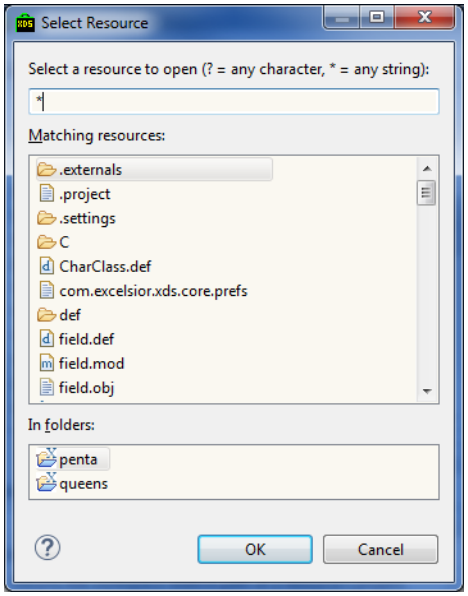

Filter field atop of the dialog alllow to quick filter variables by their name. Wildcards are enabled: `*` is an arbitrary string, `?` is a single character or an empty string.

When the particular variable is selected the **Variable Description** field will show the quick variable description. Some variables can make use of their arguments – in this case **Argument** field will become active – arguments can be configured with the **Configure...** button.

Apply changes and close the dialog with the **OK** button.

### 7.5.1 Configure arguments

Argument configuration dialog looks like the following:

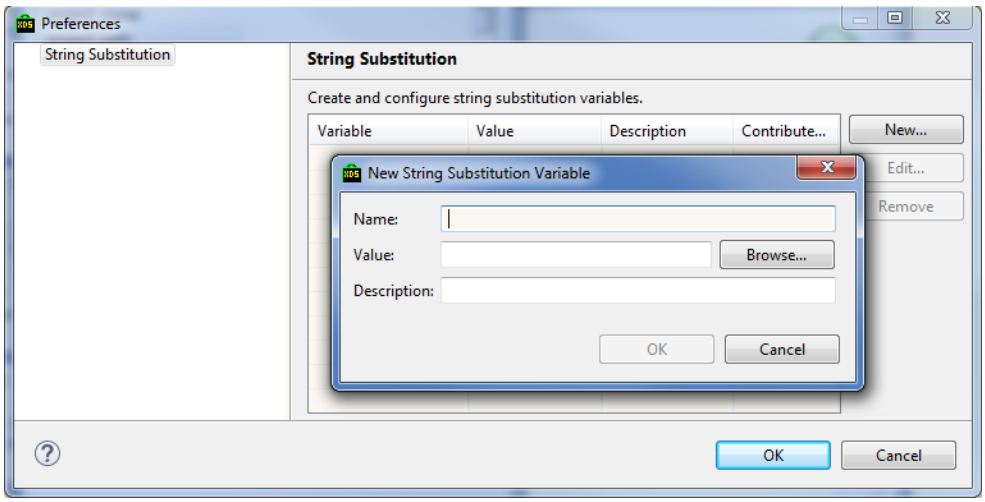



Filter field atop of the dialog alllow to quick filter variables by their name. Wildcards are enabled: `*` is an arbitrary string, `?` is a single character or an empty string. The **Matching resources** field show resources matching the condition.

When the resource is selected the **In folders** field will show its location.

### 7.5.2 Edit variables

One can introduce new variables using the **Edit Variables...** button. To add the variable press the **New...** button on the right.




Variable has the following properties:

- variable name – **Name** field;
- variable value – **Value** field.

Only the name field is mandatory. When specifying the variable value one can use predefined IDE variables – these can be selected by pressing the **Variables...** button.

Press the **OK** button to add the variable to list.

To edit the variable use the **Edit...** button, to delete it use the **Remove** button.

## 7.6 Keyboard map

| Edit | |
|---|---|
| `Ctrl+Z` | Undo last edit |
| `Ctrl+Shift+Z` | Redo last edit |
| `Ctrl+Y` | delete current line |
| `Ctrl+/` | Comment/uncomment lines |
| `Ctrl+Shift+/` | Comment selected |
| `Ctrl+Shift+\` | Uncomment |
| `Ctrl+I` | Indent selected lines |
| `Ctrl+Shift+F` | Format selected text |
| `Alt+Shift+Up` | Select containing construction |
| `Alt+Shift+Down` | Shrink containing construction selection |
| `Ctrl+R` | Rename element |
| Search | |
| `Ctrl+F` | Open xFind panel |
| `Ctrl+ЪЪжае` | xFind: quick find previos occurence |
| `Ctrl+Ъиіљ` | xFind: quick find next occurence |
| `Ctrl+Alt+F` | Search and replace dialog |
| `Ctrl+H` | Æнhгкґ-2 search |
| `Ctrl+G` | Find declarations of the selected element in the workspace |
| `Ctrl+Shift+G` | Find usages of the selected element in the workspace |
| Navigation | |
| `Ctrl+L` | Go to line |
| `F3` | Open declaration of the selected element |
| `Ctrl+M` | Open module from list |
| `Ctrl+P` | Open coupled module |
| `Alt+Shift+O` | Enable/disable mark occurences |
| Ќнæsїкпжїп ї льґsгбќ | |
| `F9` | Compile file |
| `Shift+F9` | Build project |
| `Ctrl+Shift+F9` | Rebuild project from scratch |
| `Ctrl+F11` | Launch last Run configuration |
| `F11` | Launch last Debug configuration |
| Numbered bookmarks | |
| `Ctrl+Shift+<digit>` | Add/remove corresponding bookmark at the current line |
| `Ctrl+<жїдаг>` | Goto corresponding bookmark |
| `Ctrl+=` | Goto bookmark |
| `Ctrl+Shift+=` | Add/remove |
| Sанзжқ | |
| `Ctrl+O` | Show module quick outline |
| `F2` | Show context tooltip |
| `F12` | Show context help |
| `Ctrl+Shift+L` | Show key assist |
| `Alt+F9` | Maximize/restore editor window |
| `Ctrl+Tab` | Next editor |
| `Ctrl+‘` | Next view |