

VIMOS SOFTWARE FAMILY

Using the Editor

28 December 2005



Thank you for your interest in our Vision Inspection and Optical Measurement System (VIMOS). In this manual you will find information about the Editor – our visual development environment, which speeds-up and improves the process of developing and testing a new VIMOS user-program.

Before going on reading the manual, we kindly ask you to read the following

DISCLAIMER

This documentation is provided for reference purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, this documentation is provided “as is” without any warranty whatsoever and to the maximum extent permitted, atto-Systems Ltd. and Wolf Systeme AG disclaim all implied warranties, including without limitation the implied warranties of merchantability, non-infringement and fitness for a particular purpose, with respect to the same. Neither atto-Systems Ltd. nor Wolf Systeme AG shall be responsible for any damages, including without limitation, direct, indirect, consequential or incidental damages, arising out of the use of, or otherwise related to, this documentation or any other documentation. Notwithstanding anything to the contrary, nothing contained in this documentation or any other documentation is intended to, nor shall have the effect of, creating any warranties or representations from atto-Systems Ltd., Wolf Systeme AG or any of their suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of this software.

The described software is provided 'as is', without any warranty expressed or implied. No guaranty is given that the software is suitable for any given purpose.

COPYRIGHT

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of atto-Systems Ltd or Wolf Systeme AG, except in the manner described in the documentation or the applicable licensing agreement governing the use of the software. All rights are reserved. Do not reverse-engineer. Do not modify or distribute without all of the documentation.

© Copyright atto-Systems Ltd. and Wolf Systeme AG

Atto-Systems Ltd.
Sofia, Bulgaria

Wolf Systeme AG
Karlsbad, Germany

All rights reserved.

TRADEMARKS

All trademarks and copyrights mentioned within the documentation are respected. They are the property of their respective owners.

CONVENTIONS USED IN THIS MANUAL



INFORMATION. This sign marks section in the manual, which is for information only. You can decide to read or skip this section.



ATTENTION. This sign marks section of the manual, which is particularly important for the general understanding of VIMOS. Please, make sure to read this section before proceeding with reading the manual.



TIPS & TRICKS. This sign marks a Tips & Tricks section. Here you can find some practical advises on using the system or get a more detailed explanation of some features. Reading this section may help you in solving a particular problem or give you some ideas but is not vital for understanding VIMOS.



PREMISE. This sign marks a section, which requires you to do something before proceeding with reading the manual. Usually this is a demo program you have to run or something similar.

File *Menu item*

File > Open *Sub-menu item*

1.1. *Section name. If the section is within the current manual no manual name is specified. When the section is within external manual the name of the respective manual is also included.*

Ctrl+E *Hot-key combination. The first part of the combination specifies which system key to use. Possible values are: Ctrl, Alt, Shift. The second part specifies the normal key to be used in the combination.*

CONTENTS

1. INTRODUCTION.....	6
1.1. EDITOR WINDOW	6
1.2. USER-PROGRAM DIAGRAM	7
1.3. RUN THE DEMO PROGRAMS	7
1.4. REGISTER THE EDITOR	7
2. EDITOR FUNCTIONS	9
2.1. WORKING WITH USER-PROGRAMS	9
2.1.1. <i>User-program properties</i>	9
2.1.2. <i>Create a new user-program</i>	10
2.1.3. <i>Open an existing user-program</i>	10
2.1.4. <i>Save changes</i>	11
2.1.5. <i>Undo/redo</i>	11
2.1.6. <i>Refresh the diagram</i>	11
2.1.7. <i>Integration with VIMOS Simulator</i>	12
2.1.7.1. Run user-program in the Simulator	12
2.1.7.2. Edit user-program in the Simulator	13
2.1.7.3. Manual user-program import.....	13
2.1.7.4. Manual user-program export	13
2.2. MANAGING DIAGRAMS.....	13
2.2.1. <i>Zoom</i>	13
2.2.2. <i>The grid</i>	14
2.2.3. <i>Add icons</i>	14
2.2.4. <i>Select icons</i>	14
2.2.5. <i>Arrange icons</i>	14
2.2.6. <i>Delete icons</i>	15
2.2.7. <i>Copy/Paste</i>	15
2.2.8. <i>Duplicate icons</i>	15
2.2.9. <i>User comments</i>	15
2.2.9.1. <i>Comment dialog box</i>	16
2.3. MANAGING USER-PROGRAM TOOLS	17
2.3.1. <i>Tool properties</i>	18
2.3.2. <i>Conditional branches</i>	19
2.3.3. <i>Enable/disable a tool</i>	21
2.3.4. <i>Hide/show a tool</i>	21
2.3.5. <i>Connecting tools</i>	21
2.3.6. <i>Tool links</i>	21
2.3.6.1. View links	21
2.3.6.2. Link arguments and results.....	22
2.3.6.3. Delete links	22
2.4. SUBROUTINES	22
2.4.1. <i>Public arguments/results</i>	23
2.4.2. <i>Calling subroutines</i>	24
2.4.3. <i>Listing subroutines used</i>	24
2.4.4. <i>Collecting subroutines</i>	25
2.4.5. <i>Reloading definitions</i>	25
2.5. EDITOR OPTIONS	25
3. EDITOR SPECIAL FEATURES	27
3.1. PSEUDO-TOOLS	27
3.1.1. <i>Mouse tool</i>	27
3.1.2. <i>Counters tool</i>	27
3.1.3. <i>Calculator 2</i>	28
3.2. TOOL ARGUMENTS AND RESULTS	28

3.2.1. <i>Steering</i>	28
3.2.1.1. Angle steering	28
3.2.1.2. Point steering.....	28
3.2.2. <i>Variable type arguments</i>	29
3.2.3. <i>Results' error codes</i>	29
3.2.4. <i>Example</i>	29
4. USER INTERFACE REFERENCE.....	31
4.1. MAIN MENU	31
4.1.1. <i>File</i>	31
4.1.2. <i>Edit</i>	31
4.1.3. <i>View</i>	31
4.1.4. <i>Tools</i>	32
4.1.5. <i>Subroutines</i>	32
4.1.6. <i>Window</i>	32
4.1.7. <i>Help</i>	32
4.2. MAIN TOOLBAR.....	32
4.3. ICON CONTEXT MENU	33
4.4. OUTPUT WINDOW CONTEXT MENU	34
4.5. SHORTCUT KEYS	34
5. MESSAGE REFERENCE.....	35
5.1. WARNING MESSAGES	35
5.2. ERROR MESSAGES	35

1. Introduction

The Editor is part of the VIMOS software family. It is designed to help you create and edit VIMOS user-programs. The Editor can exchange these programs with the Simulator and the camera.



INFORMATION. See *Getting Started* for an introduction to VIMOS.

Of course you can edit user-programs with the Simulator or directly on the camera, but the Editor is more convenient for this task. It presents the program as a diagram, which reflects its structure and content in a more natural way. The Editor also provides visual means to manipulate the user-program. Many operations, that require complex navigation in the limited user interface on the camera, are achieved with a couple of mouse clicks in the Editor. This is possible because the Editor takes full advantage of Windows GUI.

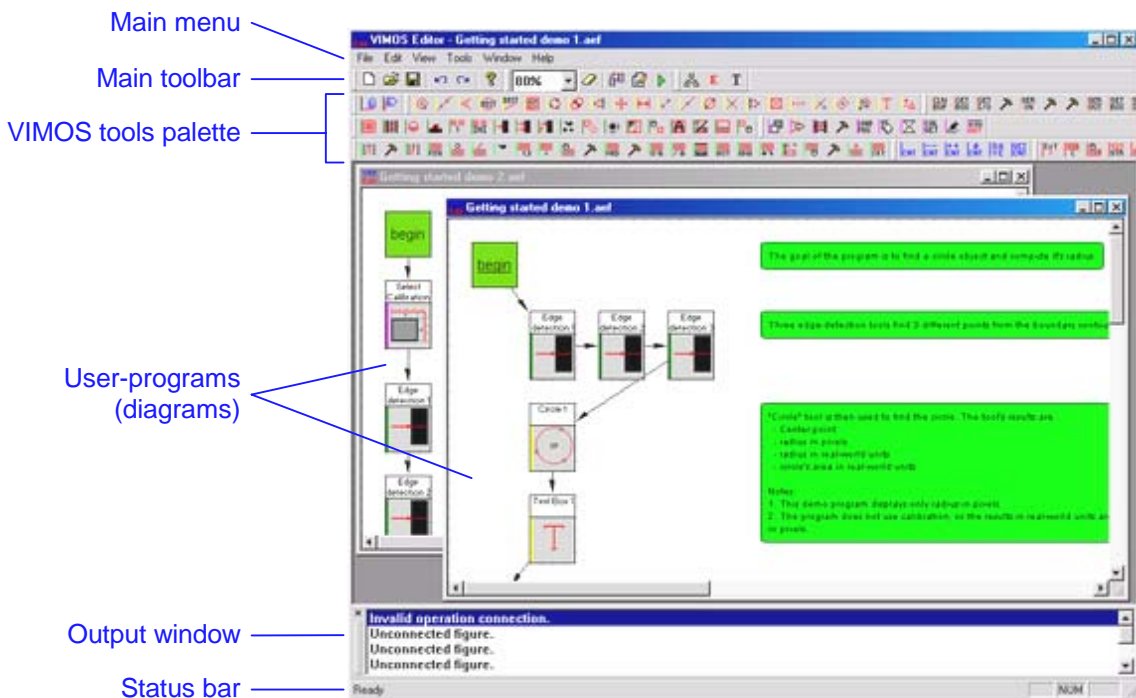
1.1. Editor window

The Editor employs the standard GUI features, seen in almost any other Windows application. So, if you have used Windows before, the Editor should be familiar to you.

These are the major components of the Editor window:

- **Main menu** – provides access to all commands
- **Main toolbar** – buttons for quick access to frequently used operations
- **VIMOS tools palette** – toolbars containing VIMOS tools for use in the user-program
- **User-program windows** – currently open user-programs, each in a separate window
- **Output window** – displays status and warning messages
- **Status bar** – displays current status or selected command's description

The following figure shows the major components of the Editor window:

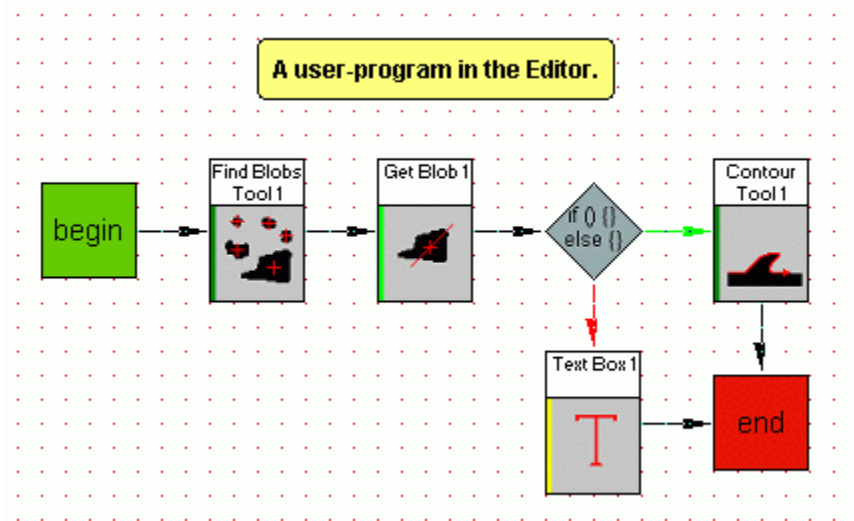


You can rearrange window components or selectively hide/show any one of them. To do this, use **View** and **Window** menus.

1.2. User-program diagram

The Editor presents each user-program in a visual way that allows you to grasp quickly the logic of your program. The Editor employs a diagram notation that is widely used to describe algorithms.

Generally a diagram consists of icons, some of them being connected with arrows. Rectangles depict user-program tools and diamonds – conditional branches. Arrows show the path of execution. It always starts at the *Begin* icon and terminates at an *End* icon (could be more than one). Rectangles with rounded corners represent user comments. The little dots scattered on the workspace are the grid, which is used to align icons.



In section “2.2. Managing diagrams” you will find out how to manipulate the diagram.

1.3. Run the demo programs

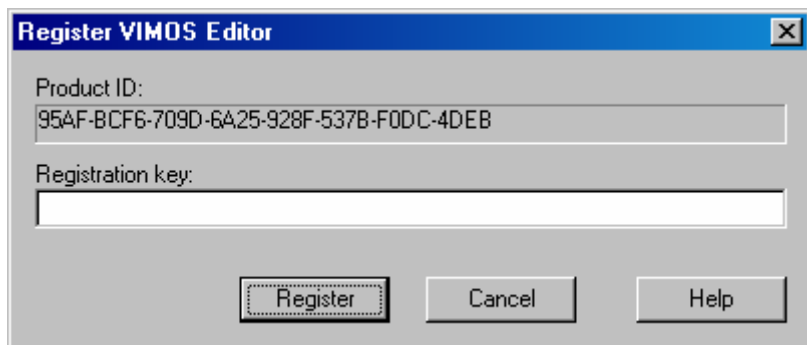
See the “Getting started” manual for detailed instructions how to run the demo programs provided with VIMOS software family.

1.4. Register the Editor

This software is usually distributed as a 30-days trial version for free. After the trial period has expired, you will not be able to use it. When you buy the software this limit is removed.

Here is the registration procedure:

1. Choose **Help > Register** menu option. Copy the **Product ID** from the registration dialog box. Send this ID together with details about your payment to the software supplier.
2. The software supplier will send you back the registration key. Enter it in the **Registration key** filed within the registration dialog. Press **Register**. The Editor will inform you of the result of the registration.



2. Editor functions

2.1. Working with user-programs

The Editor uses multiple-document interface (MDI), like MS Word and many other Windows programs. This allows you to have more than one user-program open at the same time. Each one of them is displayed in a separate window inside the main application window. You can arrange user-program windows with the commands from **Window** menu.

The Editor stores each user-program in a file with `.aef` extension. This file uses a private format, which is recognized only by the Editor. That is why this file cannot be directly used in other parts of the VIMOS software family. The Editor provides other means to exchange user-programs with the rest of the VIMOS (see section “2.1.7. Integration with VIMOS Simulator”).



ATTENTION. Although the Editor files (`.aef`) are not recognized and cannot be used by the Simulator and the VIMOS-Kernel directly, there is a simple way to exchange your user-programs between the members of the VIMOS software family. See section “2.1.7. Integration with VIMOS Simulator” to find out how to exchange user-programs between the Editor and the Simulator.

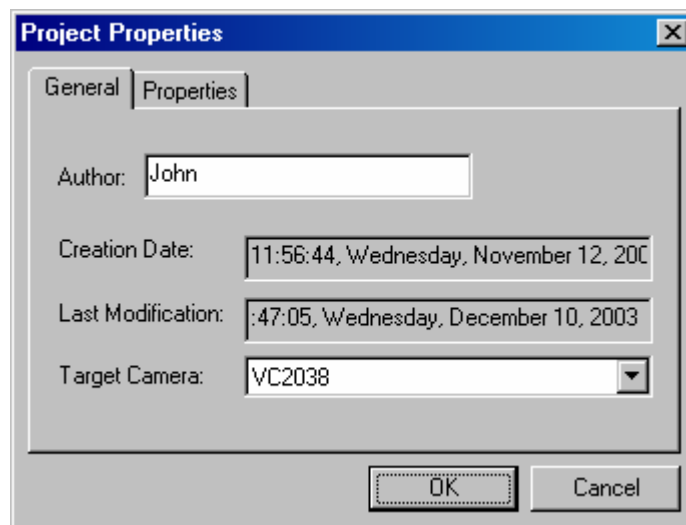
2.1.1. User-program properties

The Editor maintains a set of properties for each user-program. They are stored in the `.aef` file.

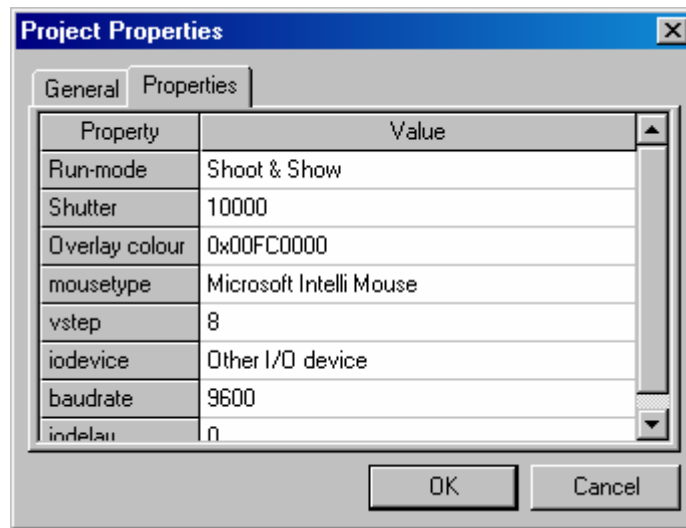
Currently the following general properties are supported:

- **Author** – this should be the name of the person who has created the user-program. Actually, this can be any text string with no restrictions.
- **Creation date** – date and time when the user-program was created
- **Last modification** – date and time when the user-program was last modified
- **Target camera** – Camera model for which the user-program is designed. This property affects the set of available user-program tools. Some tools are not available on some camera models. So, depending on the value of this property some tools could be disabled. To find out what tools are available on what camera models, refer to the “Tools’ Description” manual.

To review or modify user-program properties select **File > Properties** menu option. This will open the **Project Properties** dialog.




Also a number of additional properties are stored in each user-program. They affect the Vimos-Kernel so look in *“Using the Vimos-Kernel”* for more details.



2.1.2. Create a new user-program

Since the Editor uses MDI, you don't have to close any open user-program before creating a new one. To create a new user-program use one of the following alternatives:

- Press **New** button  in the main toolbar
- Select **File > New** menu option
- Press **Ctrl+N** key combination


This will present you the **Project Properties** dialog. Here you enter author name and target camera model. See section *“2.1.1. User-program properties”* for details on this dialog. Press **OK** to continue. You can change the properties anytime you wish as described in section *“2.1.1. User-program properties”*.

A new window is opened. It contains an empty diagram with a single *Begin* icon. Now you can start composing your new user-program.

2.1.3. Open an existing user-program

Since the Editor uses MDI, you don't have to close any open user-program before opening another one.

To open an existing user-program, created by the Editor, use one of the following alternatives:

- Press **Open** button  in the main toolbar
- Select **File > Open** menu option
- Press **Ctrl+O** key combination
- Double-click the desired diagram file (.aef) in Windows Explorer. You should have installed VIMOS Editor successfully, for this to work.



INFORMATION. *The Editor is backward compatible. This means that it can open diagrams created by older Editor versions, but it cannot open files created by newer versions.*


In order to get more power and flexibility, tools' format can change in new VIMOS versions. This means that tools' arguments and results can change. Usually newer versions add some new arguments or results to a couple of tools. This change is safe because it does not affect old user-program data.

If some tools in your user-program use an older format, they will be automatically converted to the new format. The Editor will display warning messages in the **Output** window for each tool it had to convert. You can find message descriptions in section “5. Message reference”. You should check each warning message to verify that the behavior of your user-program is not altered.

If the Editor has loaded the file successfully, it displays the diagram in a new window.

2.1.4. Save changes

To keep the changes you have made to a diagram, you should save it to file (.aef). To do this use one of the following alternatives:

- Press **Save** button  in the main toolbar
- Select **File > Save** or **File > Save As** menu option
- Press **Ctrl+S** key combination

When closing an unsaved diagram the Editor will prompt you to save it.




INFORMATION. We strongly recommend you to save your diagram (.aef file) in the same folder, where the rest of the files for your user-program, such as VIMOS Kernel program(s), images, etc, are stored. It is good practice to keep each user-program in a separate folder (workspace).

For more information see section “2.1.7. Integration with VIMOS Simulator”.

2.1.5. Undo/redo


The Editor is forgiving. If you accidentally make a mistake or change your mind, you can undo the changes you have made to the diagram. This actually restores a previous state of the diagram.

These are the alternative ways to undo changes:

- Press **Undo** button  in the main toolbar
- Select **Edit > Undo** menu option
- Press **Ctrl+Z** key combination

As you undo changes, they are not irretrievably lost. If you change your mind, you can redo them again. However, you cannot redo changes, if you have made any modifications after the undo operation.


Use one of these alternatives to redo changes:

- Press **Redo** button  in the main toolbar
- Select **Edit > Redo** menu option
- Press **Ctrl+Y** key combination

What is even better, the Editor supports multiple levels of undo/redo. This allows you to trace back and forth the recent history of your diagram. Look in section “2.5. Editor options” to find out how to adjust the levels of undo/redo.

2.1.6. Refresh the diagram

You can tell the Editor to redraw the whole diagram. To refresh the active diagram use one of these alternatives:

- Press **Refresh** button  in the main toolbar
- Select **View > Refresh** menu option
- Press **F5** key

The diagram will blink. Now it reflects exactly the underlying data.

2.1.7. Integration with VIMOS Simulator

Most program-editing tasks can be done in both Editor and Simulator, but there are some operations that can be accomplished only with the Simulator. For example to position a tool visually on an image, you need to see that image. Since images are not available in the Editor, you have to use the Simulator for this job.

The Editor and the Simulator provide means to exchange user-programs. This section will show you how to do that.



INFORMATION. *Since Editor and Simulator cannot share user-program files directly, some sort of conversion has to be done. An intermediate metaprogram file (.mpr) is used for this purpose. Both Editor and Simulator can import/export user-programs from/to .mpr files. Usually this conversion happens automatically and you do not see it.*

Metaprograms are actually text files and in rear occasions it could be useful to view or edit them.

Metaprograms have limited capabilities and do not support all Editor's features. So when exporting a diagram to metaprogram, some information (like comments) could be lost.



INFORMATION. *The cooperation between Editor and Simulator will work only if you have installed VIMOS software family successfully on your PC.*

Whenever the Editor needs the Simulator, it will launch it automatically. If the Simulator is already running, the Editor will use it instead of starting a new instance. If the Simulator is busy with some other task, you will get an error message. For details see section "5.2. Error messages".



ATTENTION. *Any unsaved changes to the current user-program in the Simulator are lost when the Editor transfers a new one.*

If your user-program is not complete or has errors, it will not be transferred to the Simulator. In this case you will get error messages in the **Output** window. Select a message to make the Editor highlight the icon causing the error. For description of error messages go to section "5. Message reference".


Note: Whenever the Editor transfers a user-program directly into the Simulator the working folder (workspace) in the Simulator will be set to the directory of the transferred diagram (.aef file). This is important because the current workspace determines a number of Simulator settings, like the image source. For more information about Simulator workspace see "Using the Simulator" manual.



ATTENTION. *A user-program in the Editor (a diagram) is not associated with any image source. This is done in the Simulator. So, to have your program working on the proper image we strongly recommend you to save your diagram (.aef file) in the folder of the respective Simulator workspace.*

2.1.7.1. Run user-program in the Simulator





You will often need to test your user-program in the Simulator. You can do this very easily from the Editor; just choose one of these alternatives:

- Press **Run** button  in the main toolbar
- Select **File > Run in Simulator** menu option
- Press **F9** key

This will transfer the active user-program to the Simulator and start it. You will see your user-program running and observe its results. You may need to switch manually to the Simulator application.

If there is no open diagram, this command will not be available.

2.1.7.2. Edit user-program in the Simulator

To configure a specific tool in the Simulator, right-click its icon on the diagram and select **Edit in Simulator** from the context menu. This will transfer the user-program to the Simulator and select the chosen tool for configuration. You may need to switch manually to the Simulator application. Editor and simulator versions 2.80 and higher support interactive move/resize/rotate operations by the mouse using the buttons ,  and  respectively. The  button is used to move tools with 2 or more point arguments. Terminate editing of current tool and return to parent “Configure program element” menu by right mouse click.

To update the user-program in the Editor with changes you have made in the Simulator, press **F6** key or select **File > Update in Editor** menu in Simulator.

Some tools in Editor cannot be edited in Simulator. For Details see section “3.1. Pseudo-tools”.

2.1.7.3. Manual user-program import

As described above in this section, the Editor can import metaprograms exported by the Simulator or by the Editor itself.

To import an `.mpr` file in Editor choose menu **File > Import**. Next, browse for the desired file. The Editor will import the chosen file and open a new window containing the rendered diagram. The new window will become active.

If there are any errors in the import process, you will see them in the **Output** window.

This operation does not affect the currently active diagram.

2.1.7.4. Manual user-program export

To export the active diagram, choose menu **File > Export** or press **Export** button . You can select an export format from:

- **Meta-program file (*.mpr)** - Editor will convert the active diagram to metaprogram and save it in the chosen file. If export is successful, you can import that file in the Simulator or back in the Editor.
- **Enhanced metafile (*.emf)** – the diagram will be saved as scalable graphics. This standard file format can be used with many popular programs including MS Word and Excel.
- **Text file (*.txt)** – the user-program will be saved as simple text. This is intended only for advanced users. This file cannot be imported back in the Editor or Simulator, it is just text.

If there is no open diagram, this command will not be available.

2.2. Managing diagrams

2.2.1. Zoom

Zooming is a very useful feature, present in almost all drawing applications. With its help you can balance the extent and the detail of the view (the visible portion of the diagram). When you zoom out, you see more of your diagram but in less detail and vice versa.

There are two ways to change the zoom level:

- **Zoom** dropdown in the main toolbar. It also displays the current zoom level.
- **View > Zoom** menu

The Editor defines zoom level in percents, 100% being the default. The scale range is from 20% to 200%.

2.2.2. The grid

This concept is very popular among drawing applications, so it is very likely that you are already familiar with it.

The grid is displayed as little dots scattered on the workspace. It helps you align your icons in good-looking rows and columns.

The grid is optional. You can turn it on/off from **View > Grid** menu. It is on by default.

When the grid is on and you move an icon it snaps to the nearest grid point.

2.2.3. Add icons

Adding new icon to your diagram is a breeze. Just press the icon's button in the toolbar or choose the icon from the **Tools** menu. The mouse pointer will change to a cross. Then click on the diagram where you want to place instances of that icon. You can place as many instances as you want. Right-click or hit *Esc* key to restore mouse pointer and return to normal operation.

As you can see there is a toolbar button for each icon type. If you cannot recognize the icon by the picture on the button, hold the mouse over the button and you will see a hint with the icon name.

2.2.4. Select icons

You can select several icons and work with them as a group. Selected icons appear with dashed outline.

You can select a number of neighboring icons with a single mouse action. Just drag the mouse, starting at an empty place on the diagram. As you drag you will see a dashed rectangle expand/shrink. This is the selection rectangle. When you release the mouse, all icons that fall inside or intersect the selection rectangle will be selected. Any previous selection will be cleared.

To select icons outside the current view, drag the mouse past the edge of the window. This will make the diagram scroll in that direction. Release the mouse or drag it back inside the window and the diagram will stop scrolling.

You can toggle an individual icon selection by clicking it while holding down **Ctrl** key. This way you can add/remove icons to/from selected group.

To clear the selection, just click on an empty place on the diagram or hit *Esc* key.

2.2.5. Arrange icons

Icon disposition does not affect the user-program in any way. It is used only for aesthetic reasons and to make the diagram more readable.

You can move icons by dragging them with the mouse. If the grid is on, the icons will snap to the nearest grid point when you release the mouse. If you want to place the icon at arbitrary position, turn the grid off. For details about the grid see section "2.2.2. *The grid*".


To move an icon outside the view, drag it past the edge of the window. Then the diagram will start scrolling in that direction. Release the mouse or drag it back inside the window and the diagram will stop scrolling.

While you can expand your diagram to the right and to the bottom indefinitely, you cannot move an icon past the left or top border.

If you drag any of the selected icons, all selected icons will move together. This way you can easily move a portion or the whole diagram to new location.

2.2.6. Delete icons

There are several ways to remove icons from the diagram:

- Choose **Delete** from icon's context menu (brought up with right-click). If the icon is part of a selection, this action will delete all selected icons.
- Hit *Del* key to delete all selected icons at once.
- Press **Delete** button  in the main toolbar. The mouse pointer will change to delete icon. Click each icon you want to delete. Right-click or press **Esc** key to restore the mouse pointer and return to normal mode of operation.

You cannot delete the *Begin* icon.


2.2.7. Copy/Paste

You can copy selected icons to the clipboard and then paste them to a new location, another user-program or another application (e.g. MS Word).


Note: when pasting icons back in Editor, only links and connections among participating icons are preserved. For more information about tool connections and links see sections "2.3.5. Connecting tools"; "2.3.6. Tool links".

Note: when pasting icons to another application only their graphical representation is preserved. Enhanced metafile is used for this operation so the resulting graphic can be scaled in accurately in the receiving application.

As with most Windows applications you can copy the selection with:


- **Edit > Copy** menu
- **Ctrl+C** shortcut
- **Copy** button 

Then you can paste it in Editor with:

- **Edit > Paste** menu
- **Ctrl+V** shortcut
- **Paste** button 

2.2.8. Duplicate icons

Often you will need to copy an icon or a group within the same diagram and preserve argument links with remaining icons. To do this select the desired icons and duplicate them with:

- **Edit > Duplicate** menu
- **Ctrl+D** shortcut
- **Duplicate** button 

Note: links between results of selected figures and arguments of unselected figures will not be duplicated because an argument cannot be linked to several results.

For more information about tool connections and links see sections "2.3.5. Connecting tools" and "2.3.6. Tool links".

2.2.9. User comments


A user comment is arbitrary text appearing inside a rectangular icon with rounded corners. You can place as many comments as you wish.

Generally, comments are used to document a user-program. In comments you usually write program description, explain the function of a tricky part of the program, record author's name and copyright holder information, etc.

Beside the text, each comment also has a style. Comment style includes:

- Text font and color
- Text alignment within the icon
- Icon background color

Follow this procedure to add new comment to the diagram:

1. Press **Comment** button  in the main toolbar or choose **Tools > Comment** menu. This will change the mouse pointer into a cross.
2. Click on the diagram where you want to place the comment icon. The **Comment** dialog box will pop up.
3. Enter comment text and style (see section "2.2.9.1. Comment dialog box").
4. After you close the dialog, you will see the new comment on the diagram.
 - a. To create another comment, proceed with step 2, or
 - b. To restore the mouse pointer and return to normal operation, right-click or press Esc key.

To change a comment, double-click its icon. This will open again the **Comment** dialog box. Here you can change comment text and style.

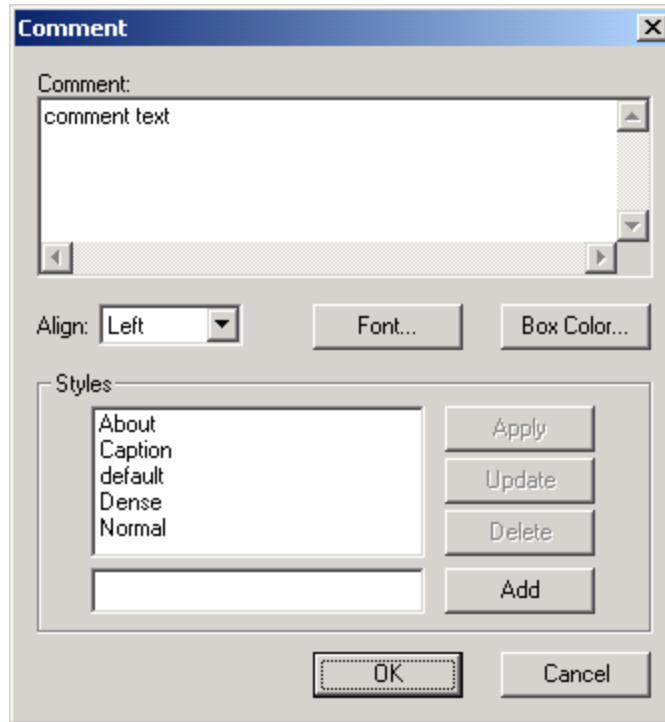


INFORMATION. *Metaprograms (.mpr) do not support user comments. So comments are not exported with the rest of the diagram. If you export a diagram and then import it back in the Editor, you will lose your comments.*

2.2.9.1. Comment dialog box

This dialog box allows you to review and change the properties of a comment. Here is the description of its controls:

- **Comment** text box – enter here your comment text
- **Align** – text alignment within the icon (left, center or right)
- **Font** – press this button to select text font and color from a standard Windows dialog
- **Box Color** – press this button to select icon background color from a standard Windows dialog
- Styles list – list of available styles. The Editor comes with some handy predefined styles.
- Styles text box – you can enter here a name for a new style (see *Add* button)
- **Apply** – push this button to apply the style selected from the list and close the dialog
- **Update** – updates the currently selected style in the list with current settings
- **Delete** – removes the selected style from the list
- **Add** – adds a new style with current settings and name – from the text box



Note that changes you make to the style list are permanent and are not revoked if you press **Cancel**.

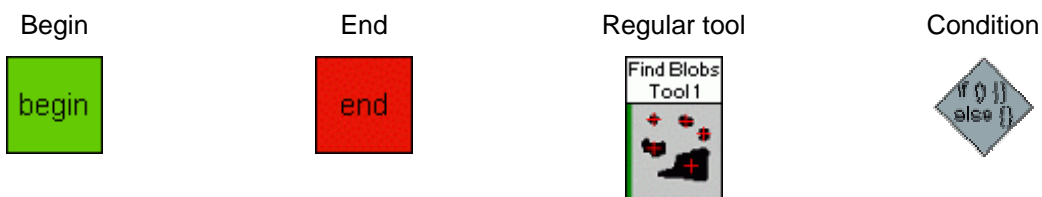
The list of comment styles is global to the Editor and not bound to any diagram. This list is not saved in the .aeF file.

Also note that changes you make to a style do not affect comments that use this style. You have to reapply the style if you want to update them.

2.3. Managing user-program tools

Tools are the building blocks of a user-program. Each tool is specialized in performing a specific task. By combining the proper tools, you can develop a user-program that solves your problem.

There are four kinds of tool icons:



Begin and *End* icons are not real tools. They do not perform any task. Their only use is to mark the start and end of the user-program.

Each diagram has exactly one *Begin* icon. You can neither remove it nor add more of it. This is where your program starts.

You can have one or more *End* icons in your diagram. This is where your program completes.



INFORMATION. In run-mode, VIMOS executes the user-program in a loop. So, after reaching an *End*, the execution goes back to *Begin*. This goes on until run-mode is stopped.

For more information refer to the “VIMOS Programming” manual.

To add *End* icon use **End** button or **Tools > End** menu. For details see section “2.2.3. Add icons”.

The condition introduces a branch in the path of execution. Two arrows, going out of condition icon, represent the possible ways to continue execution. Executions will take the green arrow, if condition is true and the read arrow, if false. For more information see sections “2.3.5. Connecting tools” and “2.3.2. Conditional branches”.

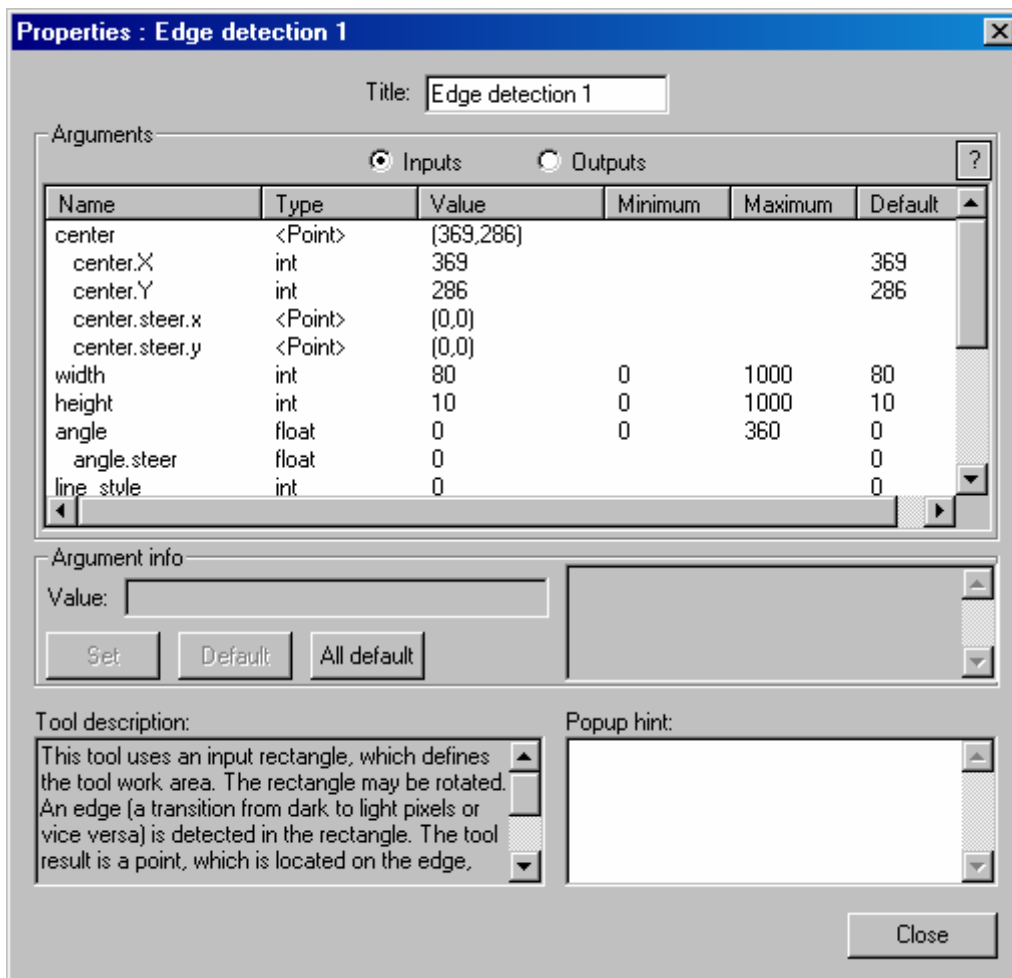
For complete description of all tools refer to the “*Tools Description*” manual.

For an in-depth discussion of VIMOS user-program read the “*VIMOS Programming*” manual.

2.3.1. Tool properties

This section is about properties of regular tools. For condition’s properties, see section “2.3.2. Conditional branches”.

Double-click a tool’s icon to see its properties. This will open the **Properties** dialog for this tool. Here you can view/change the tool’s properties.



Properties dialog controls:

- **Title** – here you can change this figure’s title
- **Inputs** – select this radio button to show tool’s arguments in the list
- **Outputs** - select this radio button to show tool’s results in the list
- **List** – here you can see all arguments/results of this tool
- **Argument info** – information about the currently selected argument/result in the list above
 - **Value** – currently selected argument value. Here you can change the value. This text box is disabled for results.
 - **Unlink** – this button is displayed in place of **Value** edit when the selected argument is linked. Press it to unlink the argument.

- **Set** – push this button to update selected argument's value with the text from *Value* box
- **Default** – resets selected argument to its default value (see column *Default* in the list)
- **All default** – resets all argument to their default values (see column *Default* in the list)
- Read-only text box – displays short description for selected argument/result
- **Tool description** – short description for this tool
- **Popup hint** – you can enter here the popup hint to be displayed when you hold the mouse pointer over this tool's icon. Leave it empty to display the default popup hint.

So, to change an argument's value, you have to select it in the list of arguments. Then you will see its value in the **Value** box. Edit the value and push **Set** or hit **Enter** to update it. You should see the argument's value change in the list.

You cannot change the value of a linked argument. To do so you have to unlink it first using the **Unlink** button. See section "2.3.6. *Tool links*" for details.

When you add a new tool to the diagram, all its arguments are set to their default values.

To understand why tool arguments and results appear different in the Editor than in VIMOS-Kernel, read section "3.1.2. *Tool arguments and results*".

2.3.2. Conditional branches

The condition statement used in a branch consists of one or more comparisons combined with Boolean AND and OR operators. Each comparison matches a tool result against a constant value.

Here is a sample condition statement:

```
Circle 1.radius1.err == 0
&& Circle 1.radius1 > 100
|| Edge detection 1.point.X <= 420
```

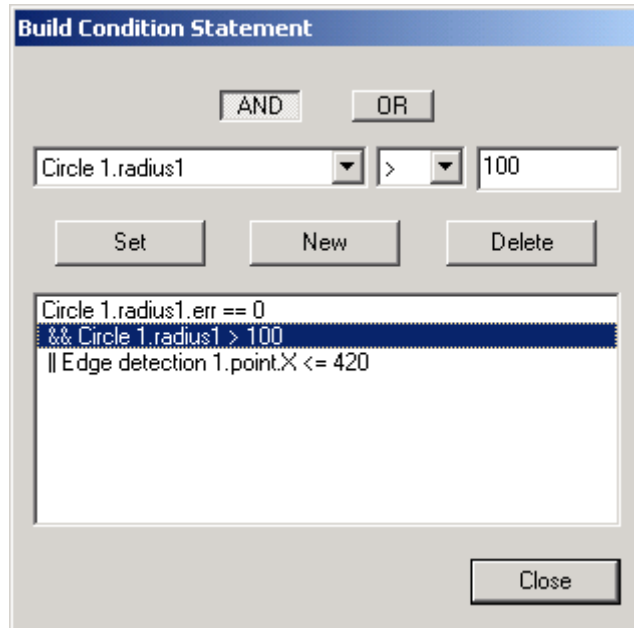
Here && is AND operator and || is OR operator.



INFORMATION. AND operators take precedence over OR operators.

When you hold the mouse pointer over a condition icon, you will see its statement in a popup hint.

To modify a condition's statement, double-click its icon. You will see the **Build Condition Statement** dialog box.



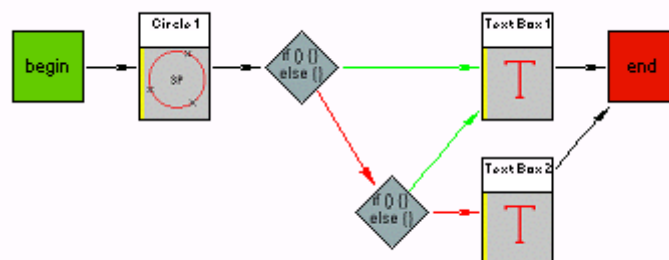
Build Condition Statement dialog controls:

- **AND** – when pressed, the current comparison is combined with the preceding using the AND operator
- **OR** – when pressed, the current comparison is combined with the preceding using the OR operator
- Tool result dropdown – select a tool result to compare. You can choose from all the results you have linked to this Condition.
- Operator dropdown – choose comparison operator
- Value box – enter a constant value to compare against the selected result
- **Set** – push this button to update the selected comparison in the list with the settings from controls above
- **New** – adds new comparison to the statement using the settings from controls above
- **Delete** – removes the selected comparison from the statement
- List – displays the full condition statement, one comparison per line
- **Close** – press to apply changes and close the dialog



ATTENTION. You should not connect conditional branches arbitrarily. They should be properly nested. This means that nested branches should merge before their parent branches. This restriction is caused by VIMOS Kernel where the user-program is presented linearly with IF-ELSE constructs.

Here is an example of mixed conditional branches, i.e. they do not nest properly.



2.3.3. Enable/disable a tool

VIMOS maintains enabled/disabled state for each tool in a user-program. A disabled tool is neither executed nor drawn in run-mode. Disabled tools are skipped during user-program execution. Tool results preserve their values from last tool execution.

Icons of disabled tools appear embossed.

To enable/disable a tool in Editor, right-click on its icon. This will show its context menu. Choose **Disabled** item to toggle tool state.

2.3.4. Hide/show a tool

Larger programs can easily clutter the screen with graphics. To avoid this you can hide some tools during run-mode.



INFORMATION. Hidden tools are executed like ordinary tools. They are not drawn in VIMOS-Kernel and their drawings are not seen in run-mode. In the Editor their icons are still visible.

Icons of hidden tools appear dim.

Check/uncheck **Hidden** item from icon's context menu to hide/show the respective tool. This menu item is not available when the tool is disabled.

2.3.5. Connecting tools

Arrows, connecting icons on a diagram, represent the order of tool execution. Each tool or condition should be in a path of execution starting from *Begin* and ending at an *End* icon. That means that each tool should have one incoming and one outgoing arrow. Each condition should have one incoming and two outgoing arrows.

Note: A tool can use results only from tools that go before it in the execution path. This means that you can link arguments only to preceding tools and results – only to succeeding tools.

To connect two tools:

1. Right-click first tool's icon to open its context menu.
2. Choose **Create connection** command. Now, as you move the mouse you see an arrow going from icon's center to the mouse pointer.
3. Click on tool's/condition's icon that should be executed next. This will connect the two icons with an arrow.

You connect conditions the same way except that you have to make two outgoing arrows for each condition. Use **Create TRUE connection** and **Create FALSE connection** commands from condition's context menu to do this. The TRUE arrow is drawn in green and the FALSE one – in red. For more information see section "2.3.2. Conditional branches".

2.3.6. Tool links


Tool cooperation is the key to making powerful user-programs. This is achieved by exchanging data among different tools. One tool can use as an argument the result of another tool. We call this linking tool results to arguments.

You can link one result to several arguments. The opposite is impossible – you cannot link one argument to several results.

2.3.6.1. View links

Normally, links among tools are not displayed on the diagram. However you can select to view all links of a particular tool.

To do this:

1. Press **Links** button  in the main toolbar, hit **F8** key or choose menu option **View > Links**.

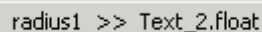
Mouse cursor will change to a pointing finger .

2. Click an icon to see all its links. Incoming links are displayed as dashed red arrows, outgoing – as dashed blue arrows. Repeat this for each icon whose links you want to see.
3. To restore the mouse pointer and return to normal operation, right-click, hit **ESC** or **F8** key, press **Links** button or choose menu option **View > Links**.

While you see the link arrows for an icon, click any one of them to see the names of linked argument and result in a popup menu.

You can also see linked tool arguments in its **Properties** dialog. The value of each linked argument is displayed as the full name of linked result.

Here is an example:



radius1 >> Text_2.float

It shows you that `radius1` result is linked to `Text_2.float` argument. It is possible to select this menu item. In this case the Editor will ask you if you want to delete this link. The link will be deleted after your confirmation.

If you want to view all the links in the diagram select menu **View > All Links**. This will show the links among all icons.

2.3.6.2. Link arguments and results

Note that one tool can use results only of a preceding tool, i.e. the source tool should be executed before the target one.

To link a tool's result to another tool's argument:

1. Right-click source tool's icon to open its context menu.
2. Choose the desired output from **Link output** submenu. Now, as you move the mouse you see a dashed arrow going from icon's center to the mouse pointer.
3. Click on the target tool/condition icon.
 - a. If the target is a condition, the link will be complete.
 - b. If the target is a tool, you will see a popup menu with all its arguments. Only arguments of matching types will be available. Choose the desired argument and the link will be complete.

You can create the same link in reverse order. First you select the target argument, then – the source result. It is the same link, no matter which way you do it.

2.3.6.3. Delete links

One way to delete a link is to show the links of one of the participating tools, click on the desired link and select the corresponding context menu item. See section "2.3.6.1. View links" for details.

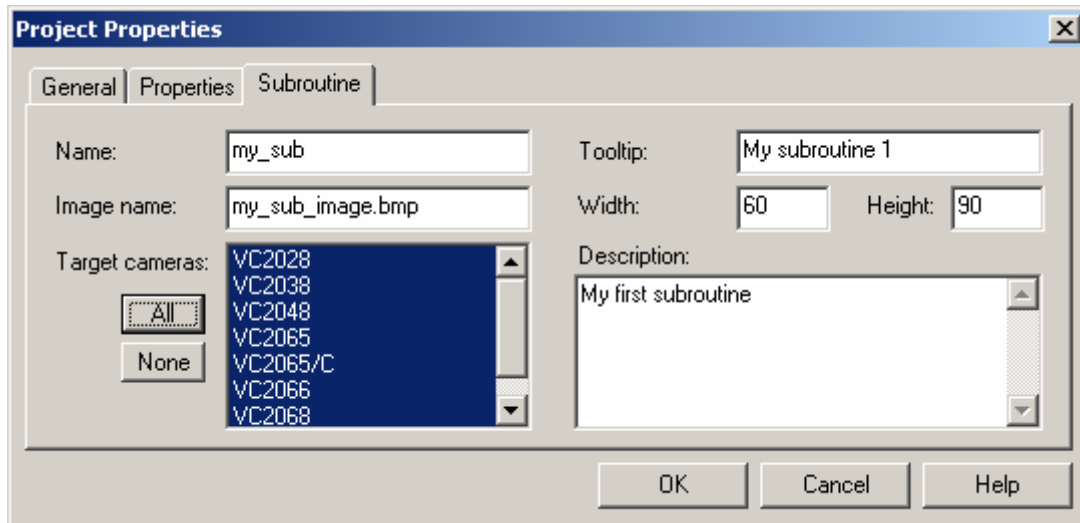
An alternative approach is to display the receiving tool **Properties**, select the desired argument and press **Unlink** button.

2.4. Subroutines

Subroutines (subprograms) are very useful when constructing large user-programs or when reusing existing solutions. A subroutine is like a usual user-program but you can declare some of its arguments and results as public. Then you can insert the subroutine in another user-program and use it just like an ordinary tool, i.e. it is displayed in the diagram by a single icon. The arguments and results of the new icon are the public arguments and results of the subroutine. They can be linked to

other tools in the diagram as usual. During user-program execution the subroutine will be executed as if it body was part of the calling user-program.

To make a new subroutine first you have to create a new user-program (section 2.1.2) or open an existing one (section 2.1.3). Then select menu item **File > Save As Subroutine** and choose new file name where the subroutine will be saved. This will bring the **Subroutine** properties dialog.



Here you have to give you subroutine a **Name** and select **Target camera** models.



ATTENTION. Due to lack of resources you can't use subroutines on ADSP cameras (VC38, VC61) and sensor cameras (VCM40, VCM50).

You can also customize the **Tooltip**, **Image** and **Description** associated with your subroutine. If you specify **Image name**, it should be a BMP file. If relative path is used, the directory containing the subroutine is used as base.

You can always open this dialog from menu **File > Properties**.


A subroutine is stored in two files:

- **.mcr** file – the main file where user-program data is stored
- **.ini** file – contains description of subroutine interface and other data used by Editor. Created automatically with the same base name as the **.mcr** file.

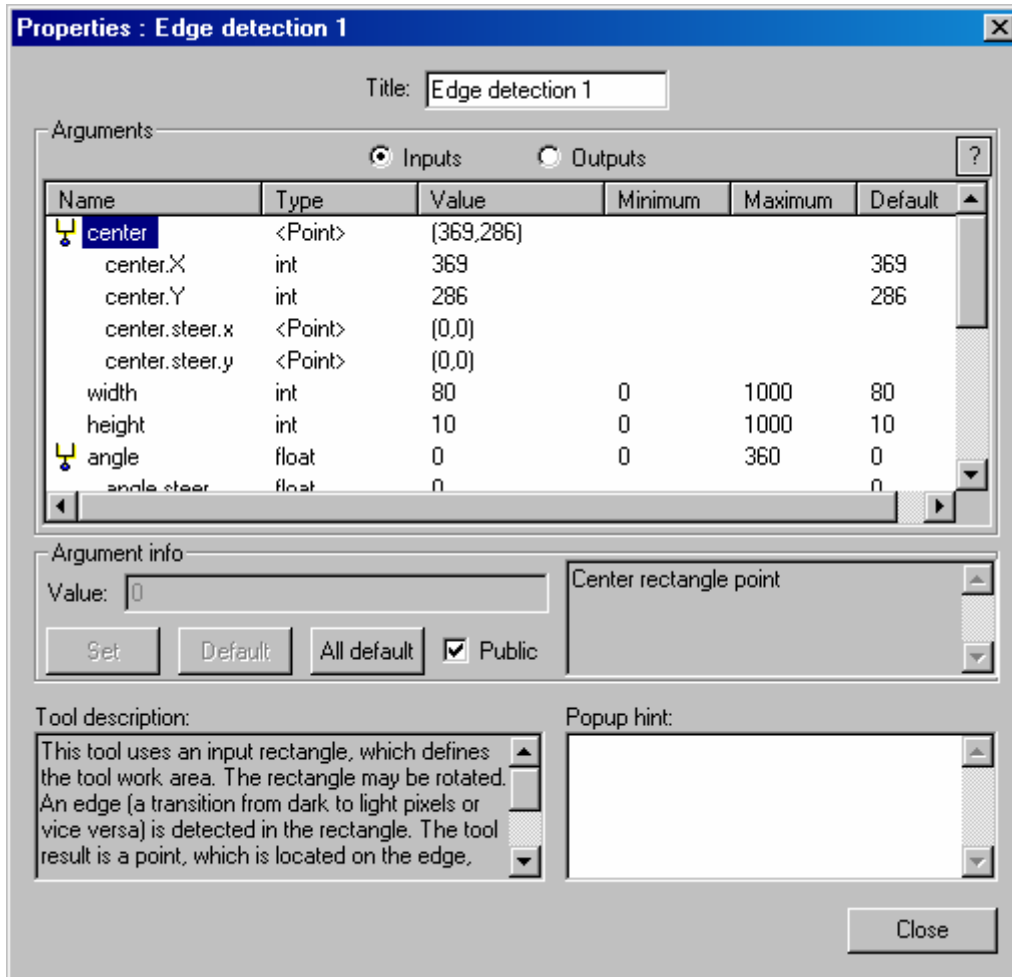
Note: with the use of subroutines you could easily create enormous user-programs that can consume the limited resources available on the camera, so use them judiciously.

2.4.1. Public arguments/results


You have to specify which arguments and results can be used from other user-programs (including other subroutines). These are called subroutine's public arguments and results. A subroutine receives input from its public arguments and emits output through its public results.

To change public argument and results open the desired tool **Properties** dialog. Public arguments and results are marked with the public icon . To change the *public* state of an argument or result, select it in the list and toggle the **Public** checkbox.

Note: public arguments are not allowed to link because they are reserved to link to the calling program.




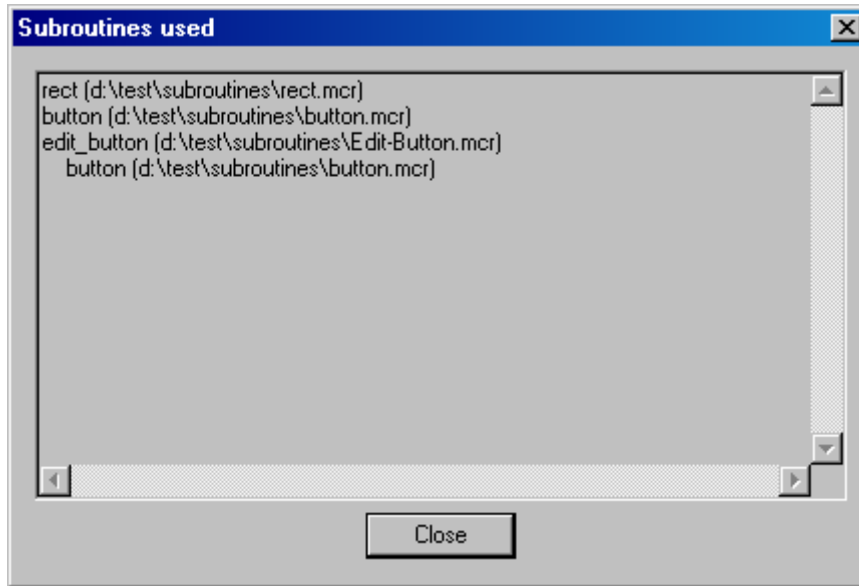
2.4.2. Calling subroutines

You can add subroutines to your user-program via menu **Subroutines > Add** or **Subroutine** button . The mouse pointer will change to a cross. Then click on the diagram where you want to place instances of that subroutine. You can place as many instances as you want. Right-click or hit *Esc* key to restore mouse pointer and return to normal operation. You connect and link the subroutine to other tools as usual (see sections 2.3.5, 2.3.6).

When you add a subroutine the Editor doesn't copy its contents into your user-program, but stores only the path to subroutine definition (.ini) file. To see this path, open the **Properties** dialog of the subroutine figure. At the bottom of the dialog you can see the path to subroutine definition file. You can change it through the browse button next to it.

2.4.3. Listing subroutines used

If you want to find out all subroutines used directly or indirectly (through other subroutines) by your user-program press the **List Subroutines** button  or choose menu **Subroutines > List**. This will show the **Subroutines used** list. If a subroutine calls other subroutines, they are listed below it and indented. Ellipses (...) in this list indicate places where a subroutine is called recursively.




2.4.4. Collecting subroutines

Sometimes you will need to copy your user-program to another computer. To be sure you will get all necessary files, it is best to collect subroutines first. You do this with **Subroutines > Collect** menu.

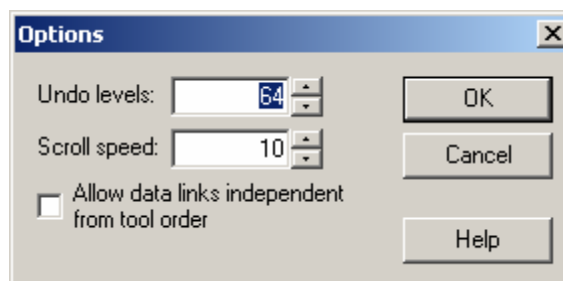
This operation will gather all subroutines that your user-program depends on in **subroutines** subdirectory where your user-program resides. This directory will be created automatically if necessary.

2.4.5. Reloading definitions

Since the multi-document interface (MDI) allows you open several diagrams, you could change a subroutine that is used by an open user-program. To reflect the changes you have to update the calling user-program. You do this by pressing the **Reload definitions** button . This is necessary only when you make changes to public arguments and results or the appearance of the subroutine (e.g. its icon).

2.5. Editor options

You can adjust some global Editor options. To do this, select menu **Edit > Options**. This will open the **Options** dialog:



The available options are:

- **Undo levels** – levels of undo/redo maintained by the Editor. Note that each undo level requires enough memory to copy the whole user-program. If you work with large diagrams and your computer has little memory, decrease this value. Change in this option will not affect currently open documents.

- **Scroll speed** – a number between 1 and 100, which defines the scrolling speed of the Editor. Greater numbers specify greater speeds.
- **Allow data links independent from tool order** – this check box enables linking of tool results to tools, which are placed before the tool in the execution sequence (i.e. the connection sequence).

3. Editor special features

3.1. Pseudo-tools

Not all tools available in the Editor are real *VIMOS-Kernel* tools. Some of them exist only in the Editor to make diagram construction more convenient. Their job in the Kernel is done by the system itself.

This is the current set of pseudo-tools:

- Mouse tool
- Counters tool
- Calculator 2

Since pseudo-tools do not exist in the Kernel, the following operations are not applicable to them:

- Enable/disable
- Hide/show
- Edit in the Simulator



INFORMATION. When the Simulator exports a user-program, it generates mouse and counters pseudo-tools (if necessary) and places these pseudo-tools in the beginning of the generated metaprogram. So when you update a user-program from the Simulator back in the Editor, you will see that the pseudo-tools are connected after the *Begin*. Therefore, we recommend connecting these tools after the *Begin* icon when you create your program

The Simulator exports series of calculator tools into one calculator 2 tool to Editor.

3.1.1. Mouse tool

As you already know, there is no Mouse tool in *VIMOS-Kernel*. Nevertheless you can link point arguments to the mouse. This is actually steering to the mouse, because the point argument retains its relative position on the screen. (Were it direct link, all points linked to the mouse would snap to the same point on the screen.)

To make this possible in the Editor, a pseudo-tool has been introduced – the Mouse tool. Its result `point` provides the current mouse position.



ATTENTION. You should not link the Mouse's result directly to any tool's argument. Instead, you should link it to the corresponding steering arguments (`.steer.x` and `.steer.y`).

3.1.2. Counters tool

Place this tool at the beginning of the diagram if you are going to use counters in your program. The tool retrieves the values of several statistics counters. Each argument value specifies number of counter, the value of which is returned by the respective result. Currently there are 1023 counters numbered from 1 to 1023.

3.1.3. Calculator 2

This pseudo-tool generates a sequence of ordinary calculator tools in the user-program, which implement more complex calculation formula. The calculations are organized in a tree of binary or unary operations.

Build desired diagram with all necessary connections between tools. First you should link to the calculator 2 tool all tool results, which participate in the calculations. You can do this by opening tool context menus by right clicks and then linking tool outputs to calculator 2.

Now you are ready to program the calculator 2 itself. Open the "Properties" dialog by left double click. Select a root calculator operation by the **Operator** combo-box. Select left and right operands (left operand only for unary operations). Each operand can be a constant, a link to one of the selected already results or a formula, which adds next entry to the tree. Click on **Constant** (if enabled), **Link** or **Formula** radio-button to specify current left and right operand. Select a desired tool result by the **Link** combo-box. Selecting **Formula** will open a child window, which defines next level in the operation tree in a similar manner. The calculations are done as if operands and operation from a given tree level are closed in brackets.

3.2. Tool arguments and results

You will notice some difference in tool arguments and results between the Editor and VIMOS-Kernel. This is caused by some special features of VIMOS-Kernel.

3.2.1. Steering

This feature allows you to specify an argument value relative to some result. In the Editor this is achieved by adding some extra arguments. In effect, a steered argument's value changes together with referenced result's value, i.e. their difference remains constant.

3.2.1.1. Angle steering

For each angle argument in VIMOS-Kernel an extra steering argument is added in the Editor.

Steering argument name: `<base name>.steer`

Here `<base name>` is the name of the original angle argument.

If the steering argument is not linked to a result, it will be ignored.

If it is linked, effective argument value = `<base value> + <steering value>`.

Here `<base value>` is the value of the original angle argument and `<steering value>` is the value of the result linked to the steering argument.

3.2.1.2. Point steering

For each point argument in VIMOS-Kernel two extra steering arguments are added in the Editor – X-steering and Y-steering. This is done to allow independent steering in horizontal and vertical directions.

X-steering argument name: `<base name>.steer.x`

Y-steering argument name: `<base name>.steer.y`

Here `<base name>` is the name of the original point argument.

Any steering argument that is not linked to a result will be ignored.

Effective argument value:

$X = \text{<base value's X> + <X-steering value's X>}$

$Y = \text{<base value's Y> + <Y-steering value's Y>}$

Here <base value> is the value of the original point argument, <X-steering value> is the value of the result linked to the X-steering argument and <Y-steering value> is the value of the result linked to the Y-steering argument.

3.2.2. Variable type arguments

Some arguments in VIMOS-Kernel can hold values of different types. Such arguments are called *variable type arguments*.

In the Editor each argument has its type and it cannot be changed. To overcome this difference, each variable-type argument from the Kernel is represented in the Editor by a group of arguments – one for each type accepted.

Each one of these arguments has a special name: <base name> .<type>

Here <base name> is the name of the original VIMOS-Kernel argument and <type> is the type name (int, float, string or point).

Here is how the effective argument value is determined. If any argument in the group is linked, the value of the referenced result will be used. Otherwise, the value of the first argument in the group will be used.



INFORMATION. Variable type arguments cannot be steered.

3.2.3. Results' error codes

In VIMOS-Kernel each tool result consists of two fields: result value and associated error code. Error code 0 indicates success.



INFORMATION. For description of result error codes see “Using the VIMOS-Kernel” manual.

To reflect this in the Editor, an extra result is added for each VIMOS-Kernel result. Thus in the Editor each tool has twice as many results as in VIMOS-Kernel.

Each extra result has a special name: <base name> .err

Here <base name> is the name of the original VIMOS-Kernel argument.

This extra result provides the error code of the corresponding result.

3.2.4. Example

As an example let’s take a look at arguments and results of *Line Across Circle* tool.

Arguments		Results	
Editor	VIMOS-Kernel	Editor	VIMOS-Kernel
Angle	Angle	point1	point1
Angle.steer		point1.err	
Center	Center	point1rw	point1rw
Center.steer.x		point1rw.err	
Center.steer.y		point2	
Point	Point	point2.err	point2
Point.steer.x		point2rw	
Point.steer.y		point2rw.err	point2rw

Radius.float	Radius
Radius.int	

You can see that `Angle` argument uses angle steering.

`Center` and `Point` arguments use point steering.

`Radius` is a variable type argument. It accepts both floating-point and integer values.

Each result has a companion `.err` result for the error code.

4. User interface reference

4.1. Main menu

4.1.1. File

Menu item	Description	See section
New	Create new user-program.	"2.1.2. Create a new user-program"
Open program...	Open existing user-program in a new window.	"2.1.3. Open an existing user-program"
Close	Close the active user-program.	
Save	Save the active user-program.	"2.1.4. Save changes"
Save As...	Save the active user-program under a different file name.	"2.1.4. Save changes"
Save As Subroutine...	Save the active user-program as s subroutine to new file.	"2.4. Subroutines"
Properties...	View/change user-program properties.	"2.1.1. User-program properties"
Run in Simulator	Run active user-program in the Simulator.	"2.1.7.1. Run user-program in the Simulator"
Import...	Create new user-program from an .mpr file.	"2.1.7.3. Manual user-program import"
Export...	Saves the active user-program as an .mpr file.	"2.1.7.4. Manual user-program export"
<recent files>	List of recently opened files.	
Exit	Close all user-programs and exit the Editor.	

4.1.2. Edit

Menu item	Description	See section
Program components...	Display information about all tools.	
Options...	Display Editor Options dialog	"2.5. Editor options"
Redo	Redo changes.	"2.1.5. Undo/redo"
Undo	Undo changes.	"2.1.5. Undo/redo"
Duplicate	Duplicate selected figures	"2.2.8. Duplicate icons"
Cut	Copy selection to clipboard and remove it from the diagram	"2.2.7. Copy/Paste"
Copy	Copy selection to clipboard	"2.2.7. Copy/Paste"
Paste	Paste clipboard contents to diagram	"2.2.7. Copy/Paste"
Delete	Delete selected figures	"2.2.6. Delete icons"
Select All	Select all figures in the diagram	"2.2.4. Select icons"

4.1.3. View

Menu item	Description	See section
Toolbars	Hide/show main toolbar	"1.1. Editor window"
Status Bar	Hide/show status bar	"1.1. Editor window"
Grid	Hide/show the grid on active diagram.	"2.2.2. The grid"

Output Window	Hide/show the output window.	"1.1. Editor window"
Zoom	Change zoom level for active diagram.	"2.2.1. Zoom"
Refresh	Redraw active diagram.	"2.1.6. Refresh the diagram"
Links	View links of a particular icon.	"2.3.6.1. View links"
All Links	View links among all icons	"2.3.6.1. View links"

4.1.4. Tools

Menu item	Description	See section
Condition	Add a conditional branch to active user-program.	"2.3.2. Conditional branches"
End	Add <i>End</i> icon to active diagram.	"2.3. Managing user-program tools"
Comment	Add <i>Comment</i> icon to active diagram.	"2.2.9. User comments"
<tool groups>	Add a tool to active user-program. Related tools are grouped together.	"2.2.3. Add icons"

4.1.5. Subroutines

Menu item	Description	See section
Add...	Insert subroutine call in the diagram.	"2.4.2. Calling subroutines"
List...	List all subroutines used	"2.4.3. Listing subroutines used"
Collect	Collect all subroutines used in one folder	"2.4.4. Collecting subroutines"

4.1.6. Window

Menu item	Description	See section
New Window	Open a new window with the active diagram.	
Cascade	Arrange windows so they overlap.	
Tile	Arrange windows as non-overlapping tiles.	
Arrange Icons	Arrange icons at the bottom of the main window.	
<open windows>	List of all open diagram windows. Use it to switch the active window.	


















4.1.7. Help

Menu item	Description	See section
Register...	Register the Editor.	"1.4. Register the Editor"
About VIMOS Editor...	Display information about the Editor, such as version, copyright, license text etc.	

4.2. Main toolbar

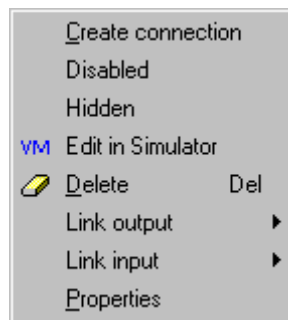


Toolbar item	Description	See section
New	Create new user-program.	"2.1.2. Create a new user-program"
Open	Open existing user-program in a new window.	"2.1.3. Open an existing user-program"
Save	Save the active user-program.	"2.1.4. Save changes"

 Export	Export user-program to another format.	"2.1.7.4. Manual user-program export"
 Cut	Copy selection to clipboard and remove it from the diagram	"2.2.7. Copy/Paste"
 Copy	Copy selection to clipboard	"2.2.7. Copy/Paste"
 Paste	Paste clipboard contents to diagram	"2.2.7. Copy/Paste"
 Duplicate	Duplicate selected figures	"2.2.8. Duplicate icons"
 Undo	Undo changes.	"2.1.5. Undo/redo"
 Redo	Redo changes.	"2.1.5. Undo/redo"
 About	Display information about the Editor, such as version, copyright, license text etc.	
 Zoom	Change zoom level for active diagram.	"2.2.1. Zoom"
 Delete	Delete icons by clicking them.	"2.2.6. Delete icons"
 Links	View links of a particular icon.	"2.3.6.1. View links"
 Refresh	Redraw active diagram.	"2.1.6. Refresh the diagram"
 Reload definitions	Reload icon definitions	"2.4.5. Reloading definitions"
 Run	Run active user-program in the Simulator.	"2.1.7.1. Run user-program in the Simulator"
 List subroutines	List all subroutines used	"2.4.3. Listing subroutines used"
 Subroutine	Insert subroutine call in the diagram.	"2.4.2. Calling subroutines"
 Comment	Add <i>Comment</i> icon to active diagram.	"2.2.9. User comments"

4.3. Icon context menu

When you right-click an icon, you will see its context menu.



Context menu items depend on the type of its icon. Here is a list of all possible items you could see in this menu.

Menu item	Description	See section
Create connection	Connect the icon.	"2.3.5. Connecting tools"
Create TRUE connection	Connect True branch of the condition icon.	"2.3.5. Connecting tools"
Create FALSE connection	Connect False branch of the condition icon.	"2.3.5. Connecting tools"
Disabled	Toggle icon's disabled state.	"2.3.3. Enable/disable a tool"

Hidden	Toggle icon's hidden state.	"2.3.4. Hide/show a tool"
Edit in Simulator	Edit this tool in the Simulator.	"2.1.7.2. Edit user-program in the Simulator"
Delete	Delete selected icons.	"2.2.6. Delete icons"
Link output	Choose a tool result to link.	"2.3.6.2. Link arguments and results"
Link input	Choose a tool argument to link.	"2.3.6.2. Link arguments and results"
Properties	Display icon properties dialog.	"2.3.1. Tool properties"

4.4. Output window context menu

When you right-click inside the **Output** window, you will see its context menu.

Menu item	Description	See section
Clear	Clear all messages.	
Hide	Hide <i>Output</i> window.	

4.5. Shortcut keys

You can use these shortcut keys with the active diagram.

Menu item	Description	See section
Del	Delete selected figures.	"2.2.6. Delete icons"
F5	Redraw active diagram.	"2.1.6. Refresh the diagram"
F8	View links of a particular icon.	"2.3.6.1. View links"
F9	Run active user-program in the Simulator.	"2.1.7.1. Run user-program in the Simulator"
Ctrl+A	Select all figures in the diagram	"2.2.4. Select icons"
Ctrl+C	Copy selection to clipboard	"2.2.7. Copy/Paste"
Ctrl+D	Duplicate selected figures	"2.2.8. Duplicate icons"
Ctrl+N	Create new user-program.	"2.1.2. Create a new user-program"
Ctrl+O	Open existing user-program in a new window.	"2.1.3. Open an existing user-program"
Ctrl+S	Save the active user-program.	"2.1.4. Save changes"
Ctrl+V	Paste clipboard contents to diagram	"2.2.7. Copy/Paste"
Ctrl+X	Copy selection to clipboard and remove it from the diagram	"2.2.7. Copy/Paste"
Ctrl+Y	Redo changes.	"2.1.5. Undo/redo"
Ctrl+Z	Undo changes.	"2.1.5. Undo/redo"

5. Message reference

This section describes warning and error messages that the Editor displays. You will see them in a message box or in the **Output** window.

5.1. Warning messages

You can see some of these messages when you open a file saved by an older Editor version. This happens when the format (arguments and result) of some tools has been changed since that version. See section "2.1.3. Open an existing user-program".

These messages are displayed in the **Output** window.

"<N> input(s) removed: <list of arguments>"

Some tool arguments have been removed. The message lists their names.

"<N> new input(s) added: <list of arguments>"

Some new tool arguments have been added. They have been set to their default values.

The message lists the names of the new arguments.

"Type mismatch in input <old name>(<old type>) -> <new name>(<new type>)"

The type of an argument has been changed. You can see its old name and type and its current name and type.

"<N> output(s) removed: <list of results>"

Some tool results have been removed. The message lists their names.

"<N> new output(s) added: <list of results>"

Some new tool results have been added. The message lists their names.

"Type mismatch in output <old name>(<old type>) -> <new name>(<new type>)"

The type of a result has been changed. You can see its old name and type and its current name and type.

5.2. Error messages

"Simulator is busy"

The Editor shows this message when it tries to connect to the Simulator but it is busy with some other task.

The Simulator is considered busy when:

- User-program is running, i.e. it is in run-mode
Solution: switch to edit-mode or turn off simulation
- There is a menu or dialog open (Windows or simulated)
Solution: close the menu/dialog

"Simulation is off"

Simulation mode not activated in the Simulator. Normally, this should happen automatically.

This is an internal error.

Solution: try again or activate simulation mode manually in the Simulator.

"Unconnected figure."

There is no path from *Begin* icon to this one.

Solution: connect this icon in your user-program. See section "2.3.5. Connecting tools".

"No conditions specified."

Empty condition statement.

Solution: specify a condition statement. See section "2.3.2. Conditional branches".

"Unconnected condition."

Both True and False branches of this condition icon should be connected. See section "2.3.5. Connecting tools".

"Empty input in figure."

There is an argument with no value.

Solution: specify an immediate value for this argument or link it. See sections "2.3.1. Tool properties" and "2.3.6.2. Link arguments and results".

"Input can be linked only to a preceding figure."

A tool can use results only from tools that go before it in the execution path.

Solution: change the execution order of the tools or link the argument to another tool.

"No operation found."

Empty user-program.

Solution: place at least one tool in your user-program.

"Mixed conditional branches."

Conditional branch do not nest properly. See section "2.3.2. Conditional branches".

Solution: connect properly the tools in conditional branches.

"Loop back detected."

The tool is connected to a preceding tool in the execution path. Loops are not allowed.

Solution: change tool connections to avoid loops.

"Invalid operation connection."

This tool has no outgoing connection.

Solution: connect this tool to the next one in the execution path.

"Unexpected ELSEIF."

Unexpected ELSEIF encountered in .mpr file during import. The .mpr file is invalid.

Solution: regenerate the .mpr file or fix it with a text editor.

"Unexpected ELSE."

Unexpected ELSE encountered in .mpr file during import. The .mpr file is invalid.

Solution: regenerate the .mpr file or fix it with a text editor.

"Unexpected ENDIF."

Unexpected ENDIF encountered in .mpr file during import. The .mpr file is invalid.

Solution: regenerate the .mpr file or fix it with a text editor.

"Missing ENDIF."

Missing ENDIF in .mpr file during import. The .mpr file is invalid.

Solution: regenerate the .mpr file or fix it with a text editor.

"Missing definition <path to .ini file>"

Definition (.ini) file not found for this figure.

Solution: restore the definition file. If it is a subroutine, open and save it again. If it is an ordinary tool, reinstall Vimos Editor. See section "2.4.2. Calling subroutines".

"Missing subroutine <path to .mcr file>"

Subroutine file not found or corrupt subroutine definition file.

Solution: restore subroutine file and/or definition file. See section "2.4.2. Calling subroutines".

"Failed to load document <path>"

Editor file is corrupt.

Solution: restore from backup.

"Failed to save document to <path>"

Editor was unable to save the document.

Solution: make sure the file is not read-only and that you have necessary permissions.