VIMOS SOFTWARE FAMILY

VIMOS Programming

5 July 2004

Thank you for your interest in our Vision Inspection and Optical Measurement System (VIMOS). This manual describes how to program the VIMOS system.

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

Atto-Systems Ltd.
Sofia, Bulgaria

Wolf Systeme AG
Karlsbad, Germany

TRADEMARKS

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 About"	*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 to user programs

VIMOS is an opened user-programmable system for machine-vision applications. It does not support a fixed number of hard-coded algorithms, but interprets a sequence of relatively simple commands called tools. To create a VIMOS application you need to combine a group of tools into a data set called "user-program".

Tools have input arguments and results. The basic types of arguments and results are integer numbers, floating point numbers, points (pairs of x/y coordinates), angles and text strings. Tool arguments can be linked to tool results, which means that results are stored into arguments of other tools, executed next in the user-program.

The creation of a user-program includes the following stages:

- Put tools in the user-program.
- Specify order of tool execution.
- Configure tools – set tool arguments and link tool results to input arguments of other tools. The results of a tool may be linked to arguments of tools, which are executed later.
- Save the user-program to file.

***INFORMATION**. The system is continuously upgraded with new tools (currently more than 100).*

Use the following system components to create and modify user-programs:

- **Editor.** The Editor is a PC program, which provides Windows-based graphical interface (menus, dialogs, icons, etc.) for creation of user-programs. Select tools from tool-list menus or tool-bars, place tool icons in the editor window and configure tools by context menus and dialogs. Specify order of tool execution by visual means. Put comments into the program.
- **Simulator.** The simulator is a PC program, which simulates operation of VIMOS Kernel on camera and provides some other functions - file transfer between camera and PC, usage of PC mouse for camera control, etc. It provides similar to Editor but more simplified single-color graphical interface for creation of user-programs.
- **VIMOS Kernel on camera.** This is the target platform, which executes user programs. It has everything to create a complete user-program. The graphical interfaces of VIMOS kernel on PC simulator and on camera are identical.

The Editor is the recommended tool for creation of user-programs. Use the simulator and the camera VIMOS for online changes in the user-program.

This manual describes in details how to program VIMOS. It emphasizes on the creation user-programs from both functional viewpoint (tool sequence, tool interaction and system resources) and technical viewpoint (how to use the graphical interface). Refer to manuals "*Using the Editor*" and "*Using the VIMOS kernel*" to find more information about Editor and Kernel GUI.

Section "2*. Main system loop*" describes the way user-program programs are executed by the VIMOS kernel.

Section "*3. Editing user-programs*" describes how to create and modify user-programs.

2. Main system loop

VIMOS executes the user-program repeatedly in a loop, called the "***main system loop***". Each loop pass takes a new image from the video-input (automatically or by a “Take image” tool in the user-program), executes the user-program, and displays results in the overlay. The tools are executed sequentially according to their order in the user-program. In other words the main loop applies the image-processing algorithm, coded in the user-program, over current camera image.

Each loop pass calls the following system modules:

- Serial communication module
- Graphical interface module
- Image acquisition module
- User-program execution module
- User-program drawing module

The VIMOS kernel has two modes of operation - run and edit. User programs are executed in run mode. The edit mode is used to create and modify user-programs and to configure the system. Depending on the operation mode and other system settings, different modules may be called in the main loop.

2.1. Serial communication

The serial communication module checks for commands, received through the serial port of the camera. Usually these are mouse commands, processed by the graphical interface module (see “*2.1.1. Overview of Kernel graphical interface*” and “*2.1.2. Overview of mouse commands*”). The commands, which concern execution of user programs, are:

- **Run mode: Stop execution of user program.** Right mouse click opens the run main menu (run mode is not stopped). The menu will be displayed after finishing the current loop cycle. Left mouse click on “Stop run-mode” option terminates user-program execution, closes the menu and enters edit mode. Note that you may have to wait some time before the system reacts to your commands, if the VIMOS kernel executes a complex and time consuming user program.
- **Edit mode: Start execution of user program.** Left mouse click on “Start run-mode” option of the edit main menu starts execution of current user program.

Mouse commands can interact with tools in run mode. If a point argument of a tool is linked to mouse, you can move the tool icon by the mouse during the execution of the user-program. Thus you will enter manually run-time data into the program. For example, you may link the point argument of a “Marker” tool to mouse and move the marker to different objects in the screen to measure distances. Refer to chapter “Tool interaction” for more details about argument linkage.

You may stop/resume user-program execution in run mode. Insert a “Pause” tool in the user program and configure the tool to wait click. Now the program will be stopped when the pause tool is reached (you will see a “pause” message displayed on the screen). Resume execution of the user-program by left click. The right click is reserved to open the run main menu.

2.1.1. Overview of Kernel graphical interface

The Kernel’s graphical interface represents a tree of menus and dialogs. The menus are composed of lines, called menu ***options***. The dialogs are composed of ***controls*** – buttons, toggle controls, spin controls and static text. The menu options and dialog controls usually open other menus and dialogs or execute some VIMOS Kernel functions.

Due to the embedded system environment, the mouse operation differs from what you have used to see in most DOS and Windows programs. There is no mouse cursor. One menu line or dialog control is always selected (highlighted). Use vertical mouse movements to navigate through menus/dialogs and to highlight next/previous item. Single button clicks activate (execute) the highlighted item. Double clicks are not supported. See manual "*Using the VIMOS Kernel*" for more details.

2.1.2. Overview of mouse commands

This section presents a brief overview of mouse commands used to work with VIMOS Kernel graphical interface:

- **Left button click.** Activate (execute) the selected menu option. Change state of selected dialog toggle or spin control and press dialog buttons.
- **Right button click.** Cancel selected option and/or close current menu/dialog. Set normal or fast operation of spin controls (fast increment: ++, fast decrement: --).
- **Vertical mouse movement**. Move mouse down/up without pressed button to select (highlight) next/previous menu option, dialog control or user-program tool when the user-program browser is open.
- **Arbitrary mouse movement.** Move tool icons on the screen (used for tool configuration).

See manual "*Using the VIMOS Kernel*" for more details about mouse operation.

2.2. Graphical interface module

This module displays menus/dialogs and interprets mouse commands, received by the serial communication module. In edit mode the module processes user commands for navigation through menus/dialogs. In run mode the module opens or closes the run main menu (the only menu which is used in run mode).

2.3. Image acquisition

Let us first discuss some aspects of camera hardware, which concern image acquisition by VIMOS Kernel. The camera sensor takes continuously images with specified shutter speed. Sensor images are transmitted to the following destinations:

- *Camera video-output*. The video-output image is shown on the camera monitor.
- *Camera frame buffer*. This is a DRAM buffer where sensor images are stored on request. The process of storing sensor image into the frame buffer is called **image acquisition** or **image taking**. VIMOS image-processing tools work on pixels in the frame buffer. Don't forget to take image in each system cycle, otherwise your user program will use indeterminate or invalid data in the frame buffer.

The video-output of the camera has two data sources:

- *Sensor data*. Sensor images are continuously sent to video-output and we see ***live picture*** on the camera monitor.
- *Frame buffer*. Frame buffer image is sent to video-output and we see ***still picture*** on the camera monitor.

ADSP cameras have separate paths from sensor to video-output and to frame buffer, so we may see live picture on the monitor, while sensor images are not saved in the frame buffer.

TI cameras have no such data path. Sensor data is always stored in the frame buffer and the picture shown on the monitor is read from the frame buffer. Continuous storing of sensor images into frame buffer and display of frame buffer on monitor provides the live picture on TI cameras.

Video display modes affect performance of ADSP and TI cameras:

- *Good ADSP camera performance* - when live sensor picture is shown on the monitor.
- *Bad ADSP camera performance* - when still picture is shown on the monitor.
- *Good TI camera performance* - when still picture is shown on the monitor (sensor images are not taken and stored into frame buffer).
- *Bad TI camera performance* - when live picture is shown on the monitor (sensor images are continuously taken and stored into frame buffer).

Take images in each main loop pass by one of the following methods:

- By automatic image acquisition (built-in function of VIMOS kernel). This is done before the execution of the user program.
- By a “Take image“ tool in the user program.

Both options may be used simultaneously, but you will receive better performance if you disable the first option when using the second one.

A system parameter called “run-mode type” controls automatic image acquisition. Enter edit mode and open the “**Set run-mode type**” dialog from **Edit main menu > Configuration > Set run-mode type…** Click on “Run-mode” toggle to select one of the modes, described in the table below. The last two table columns specify whether the corresponding mode takes automatically images and how it affects performance of ADSP and TI cameras. Note that different modes should be used on different camera models to achieve same functionality or good performance.

Mode	Description	ADSP camera	TI camera
Live	Show ***live picture*** on the monitor. Intended for manual inspection applications.	No image taking Good performance	Image taking Bad performance
Live & Shoot	Show ***live picture*** on the monitor. Intended for image-processing applications where you don’t need to see the processed image.	Image taking Good performance	Image taking Bad performance
Shoot & Show	Show ***still picture*** on the monitor. Intended for image-processing applications where you need to see the processed image (testing).	Image taking Bad performance	Image taking Good performance
Show	Show ***still picture*** on the monitor. Intended for image-processing applications where you take images with a “Take image” tool.	No image taking Bad performance	No image taking Good performance

The “Take image” tool has the following advantages, compared to automatic image acquisition. You are able to:

- Specify dynamically shutter speed (link shutter argument to result of a previous tool).
- Specify video-mode, which should be set after the image taking operation (live or still).
- Get images, synchronized with external trigger.
- Take and process several images in one user-program pass.
- Use conditional image taking to achieve better system performance.

ATTENTION**. The automatic image acquisition, controlled by the "run-mode type" parameter, is valid for VIMOS kernel running on camera. The PC simulator of the VIMOS kernel uses images, read from image files. Open the* **"Image Source" dialog from* **Camera > Image Source…** *to specify image files. Refer to manual "Using the Simulator" for more details.*

2.4. User-program execution module

This module interprets the user program. Tools are executed according to their order in the user-program. Jumps and conditional branches may be specified by GOTO and IF tools. The module performs one pass of the user-program, which begins from the first tool and ends with the last tool.

You can enable or disable tool execution in Editor and VIMOS Kernel (default: all enabled). See manuals "*Using the Editor*", section "*2.3.3. Enable/disable a tool*" and "*Using the VIMOS Kernel*", sections "*6.2.1. User-program menu*" and "*6.2.1.3. User-program browser*".

2.5. Drawing module

All tool results, generated by the user-program execution module, are saved into internal buffer of VIMOS Kernel. In run mode the module uses the results of user-program execution stage to draw tool icons in the overlay screen. In edit mode the module draws menus and dialogs.

You can hide or show individual tools in Editor and VIMOS Kernel (default: all shown). See manuals "*Using the Editor*", section "*2.3.4. Hide/show a tool*" and "*Using the VIMOS Kernel*", sections "*6.2.1. User-program menu*" and "*6.2.1.3. User-program browser*".

You can totally disable execution of drawing module in run mode by setting Draw = OFF in the "Set run-mode type" dialog (see manual "*Using the VIMOS Kernel*", section "*6.2.2.1. Set run-mode type dialog*").

3. Editing user-programs

You can create and modify user-programs on two platforms – Editor or VIMOS Kernel (on Simulator and on camera). Next sections describe how to edit user-programs on both platforms. Remember that Editor is the recommended tool for creation and maintenance of user-programs. Use VIMOS Kernel on Simulator to correct user-programs during the test phase and update changes back in the Editor. It is possible to create a user-program entirely by VIMOS Kernel and export it to the Editor.

VIMOS software components support different formats of user-programs files. Refer to *manual "Using the Simulator*", sections "*5.2. User-program file formats*" and "*5.3. Overview of user-program exchange operations*" for details about user-program file formats and export/import operations.

***ATTENTION**. You can edit multiple user-programs in different Editor windows, but one user-program at a time in VIMOS Kernel. Remember to save changes in current Kernel program (**Edit main menu > Save user-program to file…**) before you begin editing a new or load existing user-program. Remember that editing of user-programs is possible in "edit" system mode.*

Next sections describe how to edit user programs:

Section "*3.1. Opening a new or existing user-program*" describes how to begin creation of a new user-program and how to open an existing user-program.

Section "*3.2. Adding/removing user-program tools*" describes how to add, insert or delete tools in the user-program.

Section "3.3. *Tool ordering*" describes how to specify order of tool execution in the user-program.

Section "3.4. *Tool configuration and interaction*" describes how to configure tools – setting tool arguments and tool interaction (linking tool results to input arguments of other tools).

Section "*3.5. Saving user-program to file*" describes how to save Editor and Kernel user-program to file.

Section "*3.6. User-program browser in VIMOS Kernel*" describes how to list entire user-program in VIMOS Kernel.

3.1. Opening a new or existing user-program

This section describes how to begin creation of a new user-program and how to open an existing user-program.

3.1.1. Create a new user-program

3.1.1.1. New user-program in Editor

Begin creation of a new user-program by one of the following methods:

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

You will see the "Project Properties" dialog. Specify a camera model and press OK button. A new diagram window is opened with a single *Begin* icon. Now you can start adding tools into the new user-program. The program must end with an *End* icon.

Refer to manual "*Using the Editor*", section "*2.1.2. Create a new user-program*" for more details.

3.1.1.2. New user-program in VIMOS Kernel

Begin creation of a new VIMOS Kernel user-program by the following operations:

- Enter edit mode.
- Select **Edit main menu > New user-program…** This will open "System Message" dialog. Press "Yes" button to begin creation of a new program.

Use **Edit main menu > Save user-program to file…** to save current Kernel program (if changed) before you start editing of new program. Refer to manual "*Using the VIMOS Kernel*", section "*6.2. Edit main menu*" for more details.

3.1.2. Open an existing user-program

3.1.2.1. Open user-program in Editor

Open an existing user-program file by one of the following methods:

- Press **Open** button  in the main toolbar.
- Select **File > Open** menu.
- Press **`Ctrl+O`** key combination.
- Double-click the desired diagram file (`.aef`) in Windows Explorer (you should have installed VIMOS Editor).

Refer to manual "*Using the Editor*", section "*2.1.3. Open an existing user-program*" for more details.

3.1.2.2. Open user-program in VIMOS Kernel

Open an existing VIMOS Kernel user-program by the following operations:

- Enter edit mode.
- Select **Edit main menu > Load user-program from file…** and open "Load user-program from file" dialog. Select user-program file by the "File name" toggle and press "OK' button. If the file exists, the Kernel will load the program and you will see the user-program tools displayed on the screen.

Use **Edit main menu > Save user-program to file…** to save current Kernel program (if changed) before you load an existing program. Refer to manual "*Using the VIMOS Kernel*", section "*6.2. Edit main menu*" for more details.

3.2. Adding/removing user-program tools

This section describes how to add and remove tools in the user-program.

3.2.1. Add/remove tools in Editor

Add a tool icon in the user-program diagram by one of the following methods:

- Press a tool icon button in the toolbar. There is a toolbar button for each icon type. If you cannot recognize the icon by the button picture, hold the mouse cursor over the button to see a hint with icon name.
- Choose a tool from the **Tools** menu.

The mouse pointer will change to a cross. Click on the diagram where you want to place an instance of the tool icon. You can place as many instances as you want. Right-click or hit **`Esc`** key to restore the mouse pointer and return to normal Editor operation.

Refer to manual "*Using the Editor*", section "*2.2.3. Add icons*" for more information. Note that order of tool execution is not specified automatically when you put tools in Editor user-program (see "*3.3. Tool ordering*").

***ATTENTION**. Insert a "Counters" pseudo-tool at the beginning of the user-program if you are going to add tools, which work with statistics counters. Refer to section "3.1. Pseudotools" in the manual "Using the Editor" for more information about pseudo-tools.*

Remove a tool icon from the user-program diagram by one of the following methods:

- Open a tool icon context menu by right-click and select **Delete** option. If the icon belongs to a group of selected icons (see manual "*Using the Editor*", section "*2.2.4. Select icons*"), all icons in the group will be deleted.
- Hit **`Del`** key to delete all selected icons at once.
- Press **Delete** button  in the main toolbar. The mouse pointer will change to a small hollow square. Click on each icon you want to delete. Right-click or press **Esc** key to restore mouse pointer and return to normal Editor mode.

You cannot delete the *Begin* icon. Refer to manual "*Using the Editor*", section "*2.2.6. Delete icons*" for more information.

3.2.2. Add/remove tools in VIMOS Kernel

Add a tool in Kernel user-program by the following operations:

- Enter edit mode.
- Select **Edit main menu > Edit user-program > Add program-element** and open "Select element group" menu. Select an element group and then select a tool from the group (all selections done by left clicks). The tool will be added in the user-program and you will see the tool's graphical representation (icon) and the "Configure program-element" menu displayed on the screen.

The tool is appended at the end of the user-program. Move or configure the added tool (see "3.4. *Tool configuration and interaction*"). Note that order of tool execution is specified by tool order in the Kernel user-program.

Insert a tool in Kernel user-program by the following operations:

- Enter edit mode.
- Select **Edit main menu > Edit user-program > Insert program-element** to open the user-program browser and select a tool. The new tool will be inserted in front of the selected tool. Cancel the operation by right click. Accept the selection by left click, which opens the "Select element group" menu. Follow the instructions for adding a tool in the user-program.

Delete a tool from Kernel user-program by the following operations:

- Enter edit mode.
- Select **Edit main menu > Edit user-program > Delete program-element**, open the user-program browser and select a tool, which should be deleted. Cancel the operation by right click. Delete selected tool by left click.

Refer to "*6.2.1. User-program menu*" in the manual "*Using the VIMOS Kernel*" for more information about adding and removing tools in Kernel user-program.

3.3. Tool ordering

This section describes how to specify order of tool execution in the user-program. Ordinary tools are executed sequentially. GOTO/LABEL and IF/ELSE/ENDIF tools may specify jumps and conditional branches inside the user-program.

3.3.1. Tool ordering in Editor

The order of tool execution in an Editor user-program is shown by arrows, which connect tool icons. Each tool should be included in the execution path, which stars from the *Begin* icon and ends at the *End* icon. This means that each tool should have one incoming and one outgoing arrow. Each condition icon should have one incoming and two outgoing arrows.

***ATTENTION**. A tool can use results from tools, which precede it in the execution path, i.e. you can link arguments to results of preceding tools (or link tool results to arguments of succeeding tools). Don't mix up **connections**, showing execution order, with **links** made between arguments and results. In normal Editor mode you will see tool connections only, which show execution order. Section "2.3.6. Tool links" in "Using the Editor" describes how to view and make links between arguments and results.*

Connect two tools by:

1. Right-click first tool's icon to open its context menu.
2. Choose **Create connection** option. You will see an arrow going from icon's center to the mouse pointer.
3. Click on tool icon that should be executed next. The two icons are connected by arrow, starting from the first tool.

Connect condition icons in the same way. Remember that you have to make two outgoing arrows for each condition. Open condition context menu and use **Create TRUE connection** and **Create FALSE connection** options. The TRUE arrow is green and the FALSE arrow is red.

To remove existing connections between tool icons press **Delete** button  in the main toolbar. The mouse cursor becomes a small square. Click on arrows you want to delete. Right-click or press **Esc** key to restore mouse pointer and return to normal Editor mode.

More information about tool connections and links in Editor can be found in the manual "*Using the Editor*", section "*2.3. Managing user-program tools*".

3.3.2. Tool ordering in VIMOS Kernel

Tool execution order in VIMOS Kernel user-program is specified by the sequence of added tools when creating the program. You can see the execution order in the user-program browser. If an Editor program has been exported to Kernel, you will see the execution path (see previous section).

You have the following possibilities to change the natural order of execution:

- Delete a tool, insert the tool at a new position in the user-program
- Use GOTO and LABEL tools to make unconditional jumps in the user-program.
- Use IF, ELSE and ENDIF tools to make conditional branches in the user-program.

3.4. Tool configuration

This section describes how to configure user-program tools. The configuration includes setting tool arguments and specifying tool interaction - linking tool results to input arguments of other tools.

When a tool is added in the user program, its arguments are initialized with default values. Usually you need to change default arguments i.e. to configure the tool. When you add a tool with graphical representation (icon) for example, you will see that it is usually placed at the center of the screen. The tool needs to be moved to other position, probably over some object in the gray picture or just to get rid of tool icon overlapping. Moving a tool modifies its point arguments – and this is one of the methods to configure a tool.

***INFORMATION**. There is not a strict sequence of tool adding and tool configuration operations. You may add some tools in the user-program, configure the tools and then add more tools.*

3.4.1. Tool interaction

The functionality of the user program is realized by loading results of executed tools into input arguments of other tools, executed next. This is done by the so-called “links” between arguments and results. In general, arguments are linked to results of the same type, but some tools have arguments, which can be linked to results of different types (“Text box”, “IF”, “Tolerance”). By default all tool arguments, which may be linked to results, are in unlinked state. Next sections describe how to link arguments to results In Editor and VIMOS Kernel and how to view argument’s link states.

***PREMISE**. Learn more about tool arguments and results from section “Resources”.*

The following types of argument links are possible:

- **Unlinked**. The argument is set to a constant value, which doesn’t change in run-mode, but could be changed in edit mode.
- **Linked to mouse**. Applicable only for point arguments. The argument follows the relative mouse movement in run-mode. In edit-mode it stays in its initial position.
- **Linked**. The argument is linked to a result and receives its value.
- **Steered**. Applicable only for points and angles. The argument is not loaded directly with a result value, but is changed relative to one or two result values. Steering means that the difference between argument and result values is constant, while the values are different. When the argument is a point, it is possible to steer its X and Y-coordinates independently and link them to different result points.

3.4.1.1. Angle-steering example

Consider the case, when the angle argument Ang1 is steered to the result A2. The initial values of the angles in edit mode are Ang1 = 30° and A2 = 60°. When A2 is changed to 70° in run mode (increment of 10), then Ang1 will change to 40°. When A2 is changed to 55° (increment of –5), then Ang1 becomes 25° and so on.

3.4.1.2. Point-steering example

Points are linked similarly, but their X and Y components are steered separately. If X and Y are steered to the same result point, the vector between the argument and result point will remain constant when the result point moves in run mode. It is possible to steer one component only of the point argument. For example if X is steered and Y is constant, the argument point will move on a horizontal line in such a way, that the X-distance between the argument and respective result point remains constant. Another possibility is to steer the X component to one result point and the Y component to another. In this case the argument point will move horizontally together with the first result point and vertically together with the second one.

ATTENTION.

When a tool is deleted from the user program, its results remain with constant values, received from the last tool execution. Be careful to remove any existing links to results of deleted tool.

Avoid linking of arguments to results of tools, which follow the current tool. An attempt to make such link opens in VIMOS Kernel a warning message dialog after which the system reopens the "Link" dialog to make a proper link. You can't make invalid links in Editor.

When an argument is linked to invalid (error) result, the tool is not drawn in run mode. This is applied for all tools except "Best line" and "Best circle", which ignore all invalid arguments and attempt to calculate their results from the remaining valid arguments.

3.4.2. Tool configuration in Editor

3.4.2.1. Setting tool arguments in Editor

See manual "*Using the Editor*" – sections "*2.3.1. Tool properties*" and "*2.1.7.2. Edit user-program in the Simulator*". Editing user-programs in Simulator is especially convenient when you set point arguments by moving tool icons on the simulation screen.

3.4.2.2. Linking arguments to results in Editor

See manual "*Using the Editor*" – section "*2.3.6. Tool links*".

3.4.3. Tool configuration in VIMOS Kernel

This section describes tool configuration in VIMOS Kernel. To configure a tool (called also user-program element) you should open the "Configure program-element" menu. There are two paths to this menu:

- The menu is automatically opened when a tool is added or inserted in the user-program (see "*3.2.2. Add/remove tools in VIMOS Kernel*").
- To configure existing tool you should open the "User-program menu" from **Edit main menu > Edit user-program**. Select the "Configure program element" option and open the user-program browser. Select the tool, which should be configured. Cancel the operation and return to parent menu by right click. Accept the selection by left click, which opens the "Configure program-element" menu.

INFORMATION*. Right clicks usually close current menu. If parent menu is not opened automatically use right click to open it. Remember that tool configuration is possible in edit mode of VIMOS Kernel.*

Now you are ready to configure the selected tool. The menu has two options:

- **Move**. Move the tool icon on the screen by arbitrary mouse movements without a pressed button. When you return to parent menu by right click, the position of the tool is saved in the user-program. Note that tools without graphical representation can't be moved.
- **Configure…** Configure the tool. Open a corresponding configuration dialog by left click and configure the tool (see "*3.4.3.1.1. Configuration dialog*"). Different tools have different configuration dialogs. Set argument values by the dialog controls. Press "**OK**" or "**Cancel**" buttons to close the dialog and return to parent menu. Tools, which don't have configuration dialog, can't be configured.

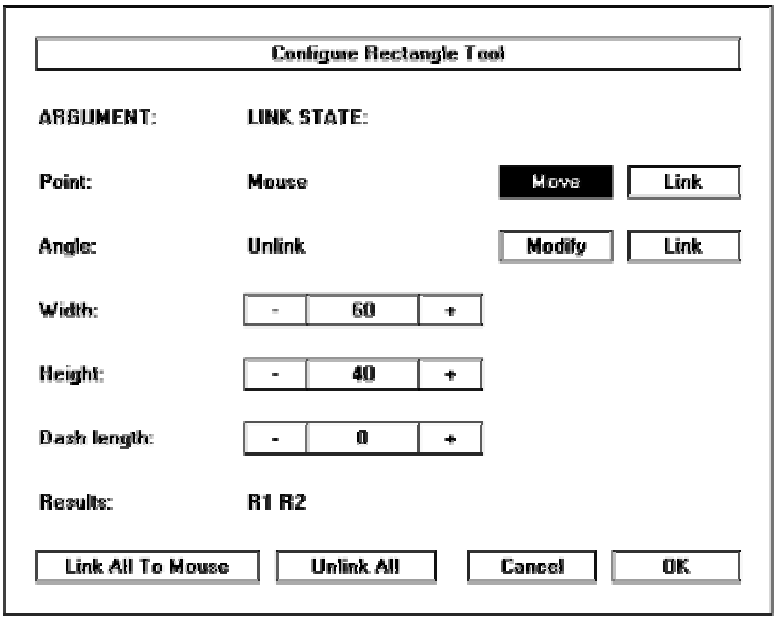
3.4.3.1. Setting tool arguments in VIMOS Kernel

3.4.3.1.1. Configuration dialog

Select “Configure…” option in “Configure program element” menu to open a configuration dialog. Use buttons, toggle and spin controls to set tool arguments. Enter this dialog to link tool arguments to results of other tools (described in “*3.4.3.2. Linking arguments to results in VIMOS Kernel*”).

The dialog format depends on the type of the selected tool. The general format of the dialog is described below. This format is valid for tools with graphical representations (icons) on the screen. Some tools have configuration dialogs in different format. Detailed information about tool-configuration dialogs can be found in the manual “*Tool Description*”.

Configuration dialog example:



The dialog **title** occupies the top dialog area. The title contains the tool name.

Tool **arguments** occupy separate screen lines. Argument lines have different formats depending on the argument type. There are two basic groups of arguments:

- Arguments, which may be linked to results (points, angles, floating-point numbers and strings). The argument line contains “Link” button.
- Arguments, which can’t be linked to results (integers, floating-point numbers, strings). They usually specify some constant tool parameter, for example line type (dash length), size, etc.

An optional **result** line under all argument lines displays tool results (if any).

Several **buttons** occupy the bottom dialog area:

- **Link All To Mouse**. Link all point arguments to mouse.
- **Unlink All.** Unlink all linked arguments.
- **Cancel.** Close the dialog and ignore any changes with one exception - links, set by the “**Link**” button, which opens the link dialog, are not ignored.
- **OK.** Close the dialog and save all argument modifications in the user program.

3.4.3.1.2. Setting point arguments

The configuration dialog contains 4 controls for each point argument. A label shows the argument name. A static text shows current link-state of the argument (unlinked, linked to mouse, linked to result or steered). The last 2 controls are buttons:

- Press **Move** button to close the configuration dialog and move the point by the mouse. Click a mouse button to terminate the move operation is and reopen the dialog. Points previously linked to results are unlinked by the move operation. States of steered and linked-to-mouse points remain unchanged. This operation resembles the move operation in the “Configure program element” menu, but the menu command moves all tool points simultaneously.
- Press **Link** button to open the “Link” dialog and change the argument’s link-state (see “*3.4.3.2. Linking arguments to results in VIMOS Kernel*”). Point arguments can be linked to point results only.

3.4.3.1.3. Setting angle arguments

The configuration dialog contains 4 controls for each angle argument. A label shows the argument name. A static text shows current link-state of the argument (unlinked, linked to result or steered). The last 2 controls are buttons:

- Press **Modify** button to close the dialog and change the angle by vertical mouse movements (down movement increases the angle, up movement decreases the angle). Usually the angle argument specifies an angle between to lines and you will see a line rotating on the screen. Click a mouse button to terminate angle modification and reopen the configuration dialog. Angles previously linked to results are unlinked by the modify operation. States of steered angles remain unchanged.
- Press **Link** button to open the “Link” dialog and change the argument’s link-state (see “*3.4.3.2. Linking arguments to results in VIMOS Kernel*”). Angle arguments can be linked to angle results only.

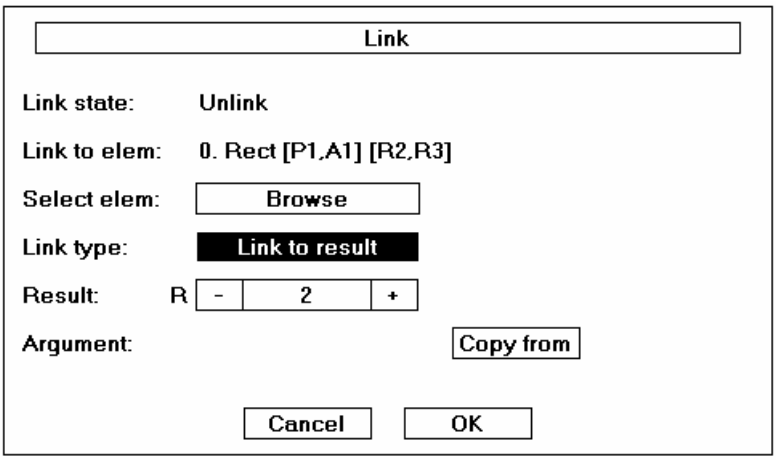
3.4.3.2. Linking arguments to results in VIMOS Kernel

3.4.3.2.1. Link dialog

Use this dialog to link an argument (called ***selected*** argument) to a result of other tool, which is executed before the current tool. The “Link” dialog is opened when you press argument’s **Link** button in the configuration dialog.

***ATTENTION**. You can’t link arguments to results of tools, which are executed after the current tool. Use the “Link” dialog to remove links as well.*

Link dialog:



The functions of the dialog box controls are described below. On entry, the controls show the current link-state of the argument:

- **Link state**. Shows the link-state of the selected argument when the “Link” dialog is opened. It is not changed by the “Copy from” button.
- **Link to elem.** Shows current user-program tool, the arguments and results of which can be used in the dialog (it is called the ***referenced*** tool).
- **Select elem.** Press **Browse** button to open the user-program browser and select another referenced element.
- **Link type**. Use the toggle to link selected argument to result. You cam make different links depending on the argument type:

Argument type	Possible link types
Angle	Unlink, Link to result, Angle-steer
Point	Unlink, Link to mouse, Link to result, X-steer, Y-steer
Float, string	Unlink, Link to result

- **Result**. Use this combination of a toggle and a spin control to specify a result, the selected argument is linked to. The toggle control specifies one of the following result types: **R** (float), **P** (point), **A** (angle) and **C** (counter). The result type is fixed and cannot be changed for angle or point arguments. The result type may be changed if the selected argument may be linked to an arbitrary type of result (see “*3.4.3.2.4. Linking arguments to arbitrary results*”). The spin control specifies a result identifier. It has a minimum and maximum value, derived from the range of available results of the referenced tool. A zero value in the spin control means that there are no results with type, specified by the “Result type” toggle (all results have identifiers greater than zero).
- **Argument**. Use the combination of toggle control and “Copy from” button to copy other tool argument into selected argument. The toggle selects argument from the referenced tool. The option is available if the selected argument is a point or an angle. An empty toggle indicates that there are no such arguments in the referenced tool. Press the “Copy from” button to copy the argument. If the copied argument is not linked, its value is copied into the selected argument. If the copied argument is linked, its link-state is copied into the link-state of the selected argument. The “Link type” and “Result” controls are updated by the “Copy from” button to reflect the new argument’s state. The “Link state” text is not updated.
- **Cancel**. Press this button to ignore changes and exit the dialog.
- **OK**. Press this button to accept changes and exit the dialog.

Notes:

1. Open the “Link” dialog twice to establish both X and Y steering for a point argument.
2. A warning message is displayed if an argument is linked to an incorrect result number.

3.4.3.2.2. Linking point arguments

Press the respective Link button in the configuration dialog. Use the “Link” dialog to change the link-state of the point. Point arguments can be linked to point results only. After closing the “Link” dialog, the configuration dialog is reopened and the link-state text reflects the modified state of the point. Note that links, made by the “Link” dialog, are not cancelled when you press the “Cancel” button of the configuration dialog.

3.4.3.2.3. Linking angle arguments

Press the respective Link button in the configuration dialog. Use the “Link” dialog to change the link-state of the angle. Angle arguments can be linked to angle results only. After closing the “Link” dialog, the configuration dialog is reopened and the link-state text reflects the modified state of the angle. Note that links, made by the “Link” dialog, are not cancelled when you press the “Cancel” button of the configuration dialog.

3.4.3.2.4. Linking arguments to arbitrary results

Some tools have arguments, which can be linked to results of arbitrary type. The argument line in the configuration dialog usually consists of 3 controls - argument name, link-state text and **Link** button. The link-state has two options (unlinked, linked to result). The “Link” dialog contains toggle, used to select type of result, the argument is linked to. Typical tools from this group are:

Tool	Argument linked to arbitrary result
Text box	“Text 2” argument (string)
IF command	“Result” argument (floating-point)
ANDIF command	“Result” argument (floating-point)
ORIF command	“Result” argument (floating-point)
Tolerance tool	“Result” argument (floating-point)

3.5. Saving user-program to file

3.5.1. Save user-program in Editor

Save user-program diagram to AEF file by one of the following medthods:

- 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 for saving it.

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

3.5.2. Save user-program in VIMOS Kernel

Save current user-program in VIMOS Kernel file by the following operations:

- Enter edit mode.
- Select **Edit main menu > Save user-program to file…** to save user-program to file with one of the predefined names ***up0.vm, up1.vm***,… If the file exists, you have the choice to cancel the save operation.

3.6. User-program browser in VIMOS Kernel

Each time you insert, delete or configure user-program tool in VIMOS Kernel, you will enter the user-program browser to list the entire user-program. You can also use the browser to enable/disable tool execution or to hide/show tool drawings in run mode.

The user-program browser looks like a menu but you can scroll up/down menu lines and see hidden previous/next tools. The number of browser lines is set in the “General-purpose configuration” dialog, opened by the “Configuration” menu. The browser wraps from the last user-program tool to the first one and vice versa.

Vertical mouse movements select consecutive tools. The graphical representation (if any) of the selected tool is marked on the screen. Left click usually accepts selected operation and closes the

browser. Right click usually cancels selected operation, closes the browser and returns to a parent menu or dialog.

Example:

Browser lines have the following format:

n. *name* [*argument 1, argument 2,* …] [*result 1, result 2,* …]

where:

n = number of the tool in the user program: 0, 1, 2, …

name = name (type) of tool.

An exclamation mark (**!**) is displayed before the name of a disabled element. An asterisk (*****) is displayed before the name of a hidden element. The format of the argument list depends on the link-state of each argument:

- Unlinked – an argument value is shown, for example: 5, 90, 2.74 for float, (120,110) for point, “text” for string argument, etc.
- Linked to mouse - the string “ML” is displayed.
- Linked – name (identifier) of result, the argument is linked to. Example: R3, P10, A2, C6, etc.
- Steered – name of result, the argument is linked to, followed by “+” for angles, “.x+” and ”.y+” for X/Y steering of points. Example: A5+, (P2.x+,P3.y+), (120,P15.x+), etc.

The result list contains result identifiers, for example: R7, P9, A4, etc.

4. Resources

4.1. Camera resources

4.1.1. Introduction to VC cameras

The cameras, manufactured by Vision Components GmbH, are compact, lightweight black-and-white or color video cameras with video memory and an image processor. They integrate high-resolution CCD sensor and fast image-processing signal processor (DSP processor). A dynamic RAM is used to store data and video images. Interfaces allow communication with the outside world. The cameras set standards for performance and integration density.

These cameras are built for industrial applications. High goals were set regarding frame resolution, sturdiness of casing and electromagnetic compatibility. The cameras are insensitive to vibrations and shocks, while permitting precise measurements and tests. They are ideally suited as OEM cameras for mechanical engineering applications.

One supply voltage is required to for camera operation (12 or 24 volts). An image processing system or a PC with a frame grabber board is no necessary. Simple control problems can be implemented with built-in camera interfaces. For more complex control tasks, the cameras can be connected with additional I/O hardware.

VIMOS Kernel runs on two basic types of VC cameras depending on the DSP processor being used:

- **ADSP cameras** - based on ADSP-21XX processor family from Analog Devices. There are two types of ADSP cameras depending on the presence of video-output hardware:
  - Cameras with video-output - live and overlay pictures are displayed on a monitor connected to camera (VC38, VC61).
  - Cheap cameras without video-output, called sensors (VCM40, VCM50).
- **TI cameras** - based on TMS62C11 processor from Texas Instruments (VC2038, VC2065).

Hardware configurations of some camera models:

Model	CPU	Sensor	Video-output	Flash	MMC*	DRAM
VC38	ADSP2183 50.35 MHz	640 x 480 (1/3”)	Yes	2MB	No	8MB
VC61	ADSP2181 40 MHz	752 x 582 (1/3”)	Yes	2MB	No	8MB
VCM40	ADSP2185 75 MHz	640 x 480 (1/3”)	No	512K	No	8MB
VCM50	ADSP2185 75 MHz	640 x 480 (1/3”)	No	512K	No	8MB
VC2038	TMS62C11 50 MHz	640 x 480 (1/3”)	Yes	2MB	Yes	16MB
VC2065	TMS62C11 50 MHz	782 x 582 (1/2”)	Yes	2MB	Yes	16MB

*) Optional multi-media card with size 8 Mbytes.

Next sections describe some basic blocks of the camera hardware.

4.1.2. CCD sensor

A high-resolution CCD sensor forms the image. Together with the appropriate control chips, it is comparable with a conventional video camera. Sensor image is stored into DRAM memory upon request and processed by the DSP processor. Different camera models have different sensor sizes

(see table in "*4.1.1. Introduction to VC cameras*"). Exposure time is controlled by shutter hardware. Shutter speed may vary from 1/10000 sec to 20 sec.

4.1.3. Video-output

The video-output hardware is used to connect a PC-compatible monitor to camera. The live picture, received by the camera sensor is displayed on a monitor. The camera generates a second "overlay" picture, which is drawn over the live picture on the monitor. ADSP cameras have single-color overlay. TI cameras have multi-color overlay. Note that cheap sensor cameras based on ADSP processor have no video-output.

4.1.4. Flash EPROM

The flash PROM of the camera plays the role of the hard disk of the PC. It is a non-volatile memory, which preserves data when the camera is switched off. The flash accommodates files in special camera format. For example camera programs are kept as executable files on flash and started by sending the file name to the shell of the camera operating system.

The flash memory is organized in sectors of 64K bytes. Sectors in erased state are usually filled with byte 0xFF. To overwrite one byte in the sector, it must be in erased state. If not, the whole sector of 64K bytes should be erased in advance.

The file system of the camera writes each new file into the free area above the existing files. The flash area occupied by deleted files can't be used. The unused space with deleted files is freed when a pack operation is performed by the "pk" shell command.

4.1.4.1. Multi-media card on TI camera

Cameras based on TI DSP processor with VCRT 5.00 and above have optional multimedia flash card (MMC) with size 8 Mbytes. The file system on the multimedia card does not require packing operations to free memory occupied by deleted files. The normal flash is accessed as **FD:** device. The MMC card is accessed as **MD:** device.

4.1.5. DRAM memory

VC cameras are equipped with DRAM (dynamic RAM) for storage of large amounts of data. The size of the DRAM memory varies from 2 Mbytes to 16 Mbytes. The DRAM is used to store any data, but is basically used for video data. The overlay picture is generated from data contained in DRAM as well. The DRAM memory is volatile – data is lost when camera power is switched off.

4.1.6. Serial interface

The camera communicates with the external world by the serial interface hardware. The camera has one serial port, which may be connected to a PC or other device. Connect the camera to a PC and use the PC software to develop VIMOS applications. Use the Simulator to transfer VIMOS system files, user-programs and other files between the PC and the camera. Use the Simulator to start VIMOS Kernel on camera and to simulate the operation of an external mouse.

By connecting a mouse to the serial port you can run VIMOS Kernel in stand-alone mode. Other external devices can be connected as well (V&C IO-box for example).

Supported baud rates – up to 115200.

4.1.7. PLC lines

The camera is equipped with additional digital I/O lines used to communicate with external devices. Most cameras have 4 PLC input and 4 PLC output lines. Sensor ADSP cameras have 2 input and 4 output lines.

4.1.8. Operating system shell

The shell of the camera's operating system is used to communicate with the user via the serial interface. A terminal like PROCOMM or HyperTerminal is usually used for this purpose. Like most operating systems, commands can be entered (with or without parameters) and are interpreted by the shell. Sending to shell names of executable flash files starts camera programs. It is possible to execute batch files in text format. A batch file with name ***autoexec*** is started automatically after power on.

4.2. VIMOS Kernel resources

4.2.1. Data types

The VIMOS Kernel supports the following data types for tool arguments:

- **Point**. A pair of X/Y pixel coordinates with additional sub-pixel fields (accuracy of 1/1000 pixel). The upper left screen corner has coordinates (0,0).
- **Angle**. Integer value in degrees in the range [0,360°]. Here 0° is at 12 o'clock and 90° is at 3 o'clock. For example an infinite line at angle 0° is vertical and a line at angle 90° is horizontal. Some tools accept angles in radians (PI/1000).
- **Floating-point number**.
- **16-bit integer number**.
- **32-bit integer number**.
- **Counter**. 32-bit Integer used for statistics calculations.
- **String**.

When a tool is added to the user program, all tool arguments receive default values. These values are changed when the tool is configured (in Editor or VIMOS Kernel).

4.2.2. Results

There is a global pool of tool results. Results have names (identifiers), specified by the data type and the number of the result:

- Results in floating-point format: **R1, R2, R3, …**
- Point results (screen points, X/Y pairs): **P1, P2, P3, …**
- Angle results: **A1, A2, A3, …**
- Statistic counters (32-bit integers): **C1, C2, C3, …** Although not results, they are used in the same way as ordinary results are used.

Each result from the pool is assigned to a certain tool when the tool is added to the user program. Each tool receives unique result identifiers. For example if you begin creation of new user-program and add two marker tools, the first one will have results P1, P2 and the second one – results P3 and P4.

Each result consists of two parts: result value and error code. When a tool produces a valid result value, it sets the result's error code to 0. When a tool fails to produce a valid result, it doesn't change its value, but sets the error code to appropriate number (> 0). Thus a result always contains the last valid value.

***ATTENTION**. When a tool is deleted from the user program, its results are not assigned to other tools. These results remain with constant values, received from the last tool execution. The user-program is still functional, but we recommend removing all argument links to results of deleted tools.*

Currently the number of results of each type is limited to 1023.

4.2.3. Frame buffer and freeze buffer

The frame buffer is a DRAM buffer where sensor images are stored on request. The size of the frame buffer is specified by the size of the sensor image (640 x 480 for example). VIMOS image-processing tools work on pixels in the frame buffer. Image acquisition fills the frame buffer with sensor data (see "*2.3. Image acquisition*").

The freeze buffer is a second DRAM buffer with the size of the frame buffer, which is used to save frame images when necessary.

4.2.4. Point-list buffer

The point-list buffer is a global DRAM buffer, which is accessed by VIMOS tools. Each point-list item consists of a point (X/Y-coordinate pair) and 8 additional floating-point parameters. Currently the size of the point-list is 25000 items (points). Point indexes begin from 0 and to 24999.

4.2.5. String buffer

The point-list buffer is a global DRAM buffer, used by the system to keep arguments and tool results in text format. Currently the size of the string buffer is 10K symbols. The user-program is responsible to distribute buffer space between different tools. Text arguments are specified by position (offset in the string buffer) and text length. Tools, which produce text results, usually receive as input argument start position of result string and return length of result string.

4.2.6. FAT buffer

The FAT buffer is a global DRAM buffer, used by the system to keep images and other data. The buffer provides fast access to DRAM data instead of reading data from flash files. Currently the following tools use the FAT buffer:

- Correlation Init tool
- Correlation Exec tool
- Free pattern tool

5. Simple programs

Sorry, not ready yet.

6. Complete solutions of certain problems

Sorry, not ready yet.