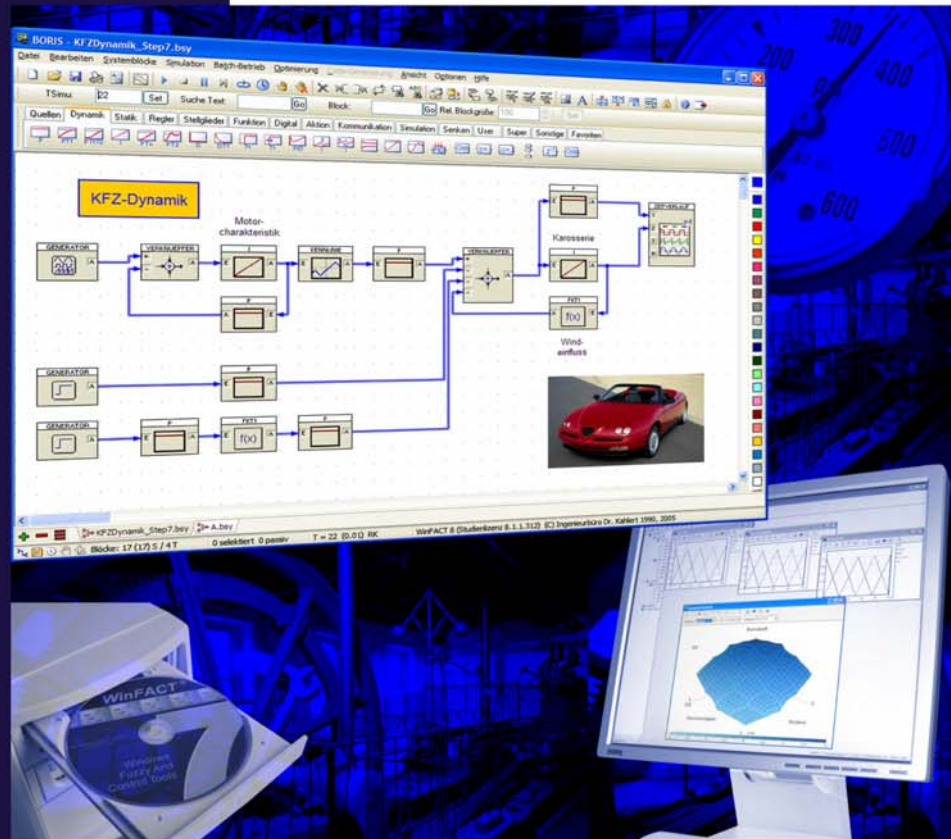


# WinFACT



## WinFACT 7

### User Manual

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45  
D-59065 Hamm

## Contents

---

<b>Contents</b>	<b>2</b>
<b>About this manual</b>	<b>3</b>
<b>Installing WinFACT</b>	<b>3</b>
<b>General instructions</b>	<b>5</b>
WinFACT-File types	5
File operations	5
Printing and exporting graphical results	7
Help options	7
<b>Configuring WinFACT by the WFSETUP-module</b>	<b>8</b>
<b>WinFACT – Program Components</b>	<b>8</b>
<b>System identification by IDA</b>	<b>11</b>
<b>Linear system analysis by LISA</b>	<b>12</b>
<b>Designing linear control systems by RESY</b>	<b>13</b>
<b>Simulation and design systems in state space representation by SUSY</b>	<b>15</b>
<b>The fuzzy shell FLOP</b>	<b>16</b>
Editing linguistic variables	17
Defining the rule base	19
System analysis in the debug mode	21
<b>Generating Fuzzy-C-Code by FALCO</b>	<b>23</b>
<b>Designing Fuzzy-PID-Controllers by FuzzyPID</b>	<b>24</b>
<b>Simulating dynamic systems with BORIS</b>	<b>28</b>
Overview	28
Components of the BORIS main window	28
Inserting and editing system blocks	29
Connecting blocks	33
Text blocks and frames	33
Structure overview	34
Simulation control	35
The BORIS system block library	36
Using superblocs	49
<b>Graphical presentation of results by INGO</b>	<b>51</b>

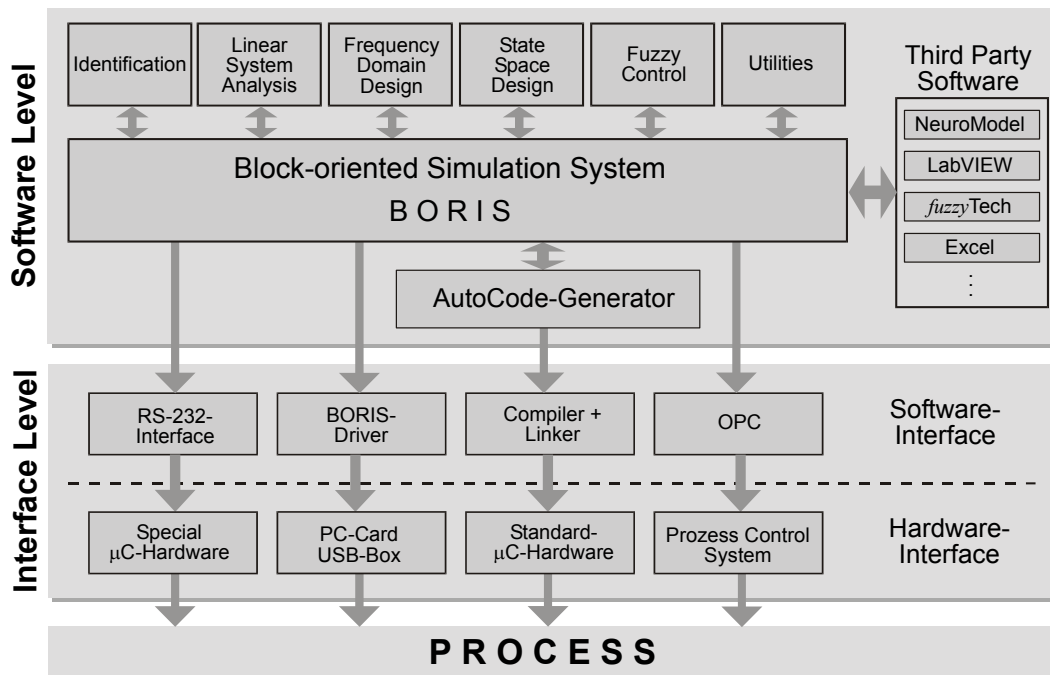
## About this manual

This documentation has primarily been written for WinFACT-users who want to get an overview of WinFACT's possibilities parallel to first experiments with the software itself and to the use of its help documents. For experienced users the study of the original documentation installed as PDF along with WinFACT 7 or the book *Einführung in WinFACT<sup>1</sup>* (German) is recommended.

This documentation mainly consists of two parts:

- A short description of the most important features of WinFACT in general
- An overview of all WinFACT-modules and their specific features and options. Included is the description of sample files delivered with the EXE-files which can be used to test all options of the individual modules.

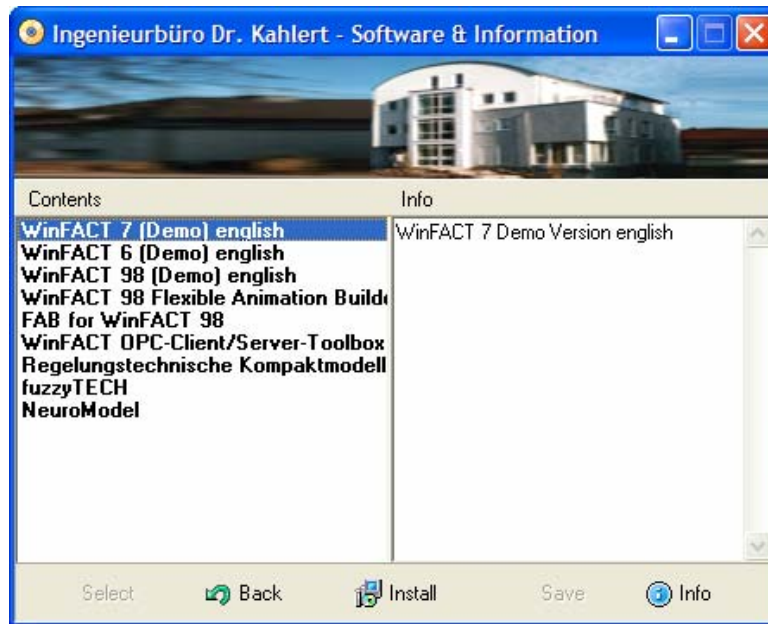
The following diagram shows the interaction of all WinFACT-modules and the different interfaces between software and hardware resp. real processes.



## Installing WinFACT

The installation of WinFACT proceeds self-explanatory. After inserting the software CD-ROM the user interface of the installation program is presented (image 1).

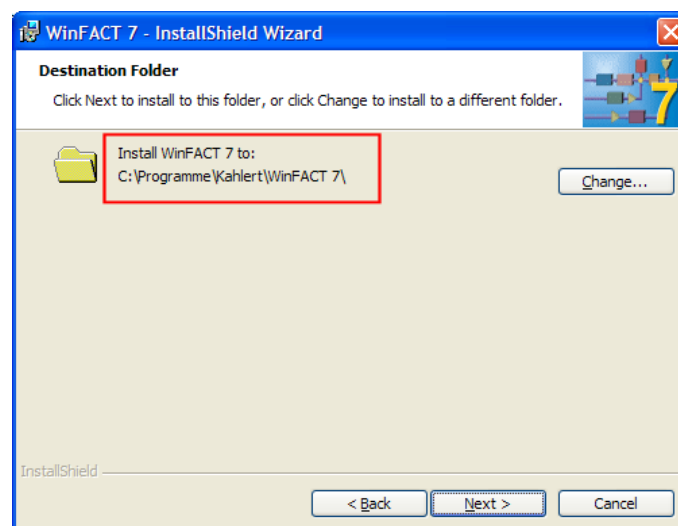
<sup>1</sup> Kahlert, J.: „Einführung in WinFACT“, Hanser-Verlag Juni 2009, ISBN 978-3-446-41960-5



*Image 1. WinFACT installation program*

In the left part of the program window all on the program CD available software components are listed. In the right part of the window a short description of the currently selected software component is shown. Select the software component you want to install and start the installation by pressing the button *Install*.

The actual installation process now proceeds menu-controlled. Please pay attention when choosing the destination folder (image 2). Under an English Windows WinFACT is installed by default into the folder *C:\Program Files\Kahlert\WinFACT 7*. This preset can be changed via the button *Change...* We will refer to the chosen folder also as WinFACT program directory later on.



*Image 2. Choosing the destination folder*

After completed installation you find all WinFACT files in the chosen destination folder resp. corresponding subfolders.

---

## General instructions

---

### WinFACT-File types



All files that are used by WinFACT for the storage of system data are ASCII text files. The main advantage of this file format is that – parallel to the handling within WinFACT – the user has the possibility to view and/or edit all files by a standard text editor. For different purposes and system types WinFACT uses different file types which can be identified by their extensions. The following file types are available:

Extension	File type
UFK	Transfer function
SIM	Simulation results (time responses $y(t)$ )
XY	General pairs of values
MXY	Trajectories stored by module SUSY
BD	Frequency response (Bode plot)
OK	Frequency response (Nyquist plot)
VEK	Vectors
MAT	Matrices
ZRM	State space systems
FWM	Function value matrices (Contour lines resp. 3D-graphics)
FUZ	Fuzzy system files
BSY	BORIS-simulation structures
SBL	BORIS-superblock files

### File operations

#### Loading and Saving of Files

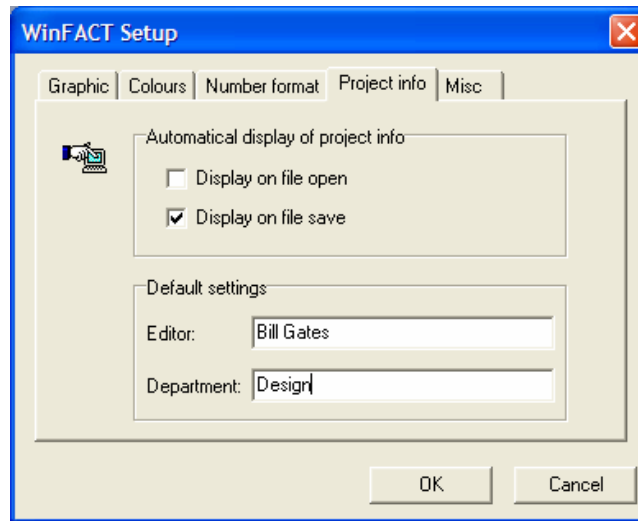
All system files generated by the different WinFACT components are ASCII text files which can be opened with any text editor. It is, however, not recommended to edit WinFACT files this way because otherwise a correct file can become "unreadable" for the corresponding WinFACT program by accidental misentries .

An existing system file is opened in almost every WinFACT component via the menu option *File / Open...*, the button  or the key <F3>. The menu option *File / Save* resp. *File / Save as...*, the button  or the key <F2> can be used for saving files.

#### Project Information

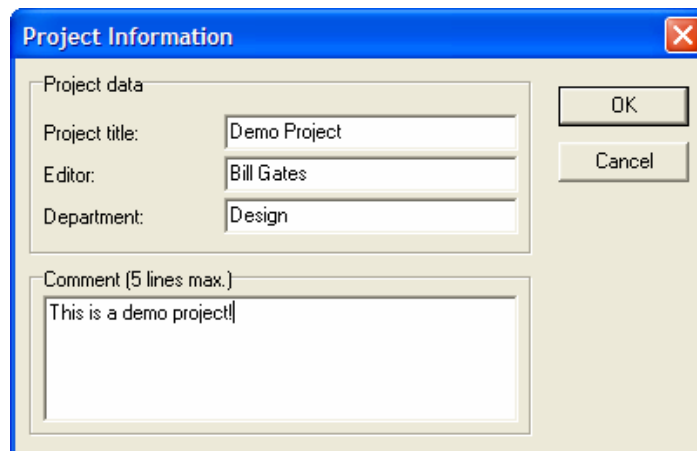
For a detailed description of the file content all file types can include a *project information* with a maximum length of eight lines that is located at the top of the file above the real data. If a file containing a project information is opened the project information is automatically

displayed if this option was activated by the WFSETUP program which you find in the WinFACT program group under *WinFACT Settings*. If a file is saved for the first time the project information is automatically requested if the corresponding option was activated (image 3).



*Image 3. Project info presetsings*

The entries made here for *Editor* and *Department* are used as presetsings in the WinFACT application programs but can be modified there. Image 4 shows the project info dialog presented by the application program.



*Image 4. Project info dialog in the application program*

All project data are already displayed within the WinFACT standard *File open*-dialog before opening the file; thus a fast search for a specific file can be executed using the included project information (image 5). Within the WinFACT application program the project info can anytime be viewed and modified via the menu option *File / Project info....*

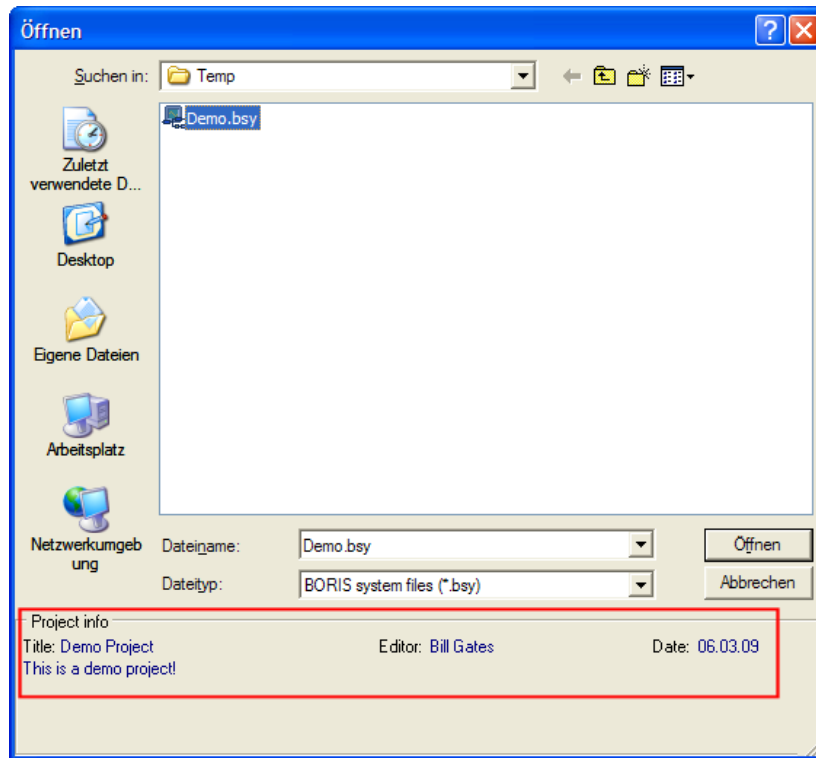


Image 5. Project info in the FILE OPEN dialog (here in BORIS)

## Printing and exporting graphical results

The graphic that is displayed in the active window of each module can be printed and – in most cases – exported at any time the module is running. Alternatively – as usual in standard WINDOWS-applications – the graphic can be copied as a bitmap to the WINDOWS-clipboard by pressing the <Alt> <PrintScreen> resp. the <PrintScreen>-key. From there the bitmap can be pasted into any other WINDOWS-application later.

To print the current graphic use the menu option *File / Print...*, the <Ctrl><P>-key or the toolbar button with the printer symbol. The print is sent to the standard printer that is currently configured. To choose another printer, simply use the menu option *File / Setup printer*. The quality of the print only depends on the printer resolution (not on the resolution of your graphic adapter). To export the graphics in WMF-format (a format that can be imported by nearly any WINDOWS-graphic program, e. g. CorelDraw) use the keys <Ctrl><X> or the toolbar. Before the graphic is saved you have to choose the filename with the WMF-extension.

In the demo version of WinFACT it is *not* possible to print or to export results.

## Help options

WinFACT offers the typical help functions of WINDOWS-applications, which can be started by the menu option *Help / Index*. The following help application is WINDOWS standard so that we don't have to talk about it in detail.

Especially for new users of WinFACT the *toolbar help function* is of great importance. This function can be activated (default) by the module WFSETUP (see later). If it is activated, any time you move the cursor on a toolbar button for more than one second, a little help window



appears which gives you information about the function of this button. By that you are able to learn the operations linked with all toolbar buttons in very short time.

When editing numerical values in parameter dialogs you often have the problem not to know the range that is valid for the variable. To avoid warnings or error messages, WinFACT offers a *numerical range help* for all edit fields in that way, that a message box with the valid range is displayed if you click with the *right* mouse button within the edit field.

---

## Configuring WinFACT by the WFSETUP-module

---

By using the setup-module WFSETUP, which is automatically installed in your WinFACT-program group during installation, you are able to predefine some default settings for all WinFACT-modules. All your changes you have made are saved in the Windows registry.

---

## WinFACT – Program Components

---









After completed installation you find all WinFACT program components in the Windows start menu under *WinFACT 7* (image 6).



Image 6. WinFACT - Program components

The following chapters provide a detailed description of the most important program components. Here they shall first be introduced in a short overview.



Icon	Program name	Short description
	BORIS	Block-oriented simulation of dynamic systems. BORIS is the main module of WinFACT and will be described more detailed later on.
	BODE-Trainer	BODE-Trainer is a program designed especially for training purposes to learn the way how to deal with Bode plots of simple or combined linear systems as well as with the stability analysis using the <i>Nyquist</i> criterion. For this purpose any number of linear standard elements (P-, PT1-, PT2-element etc.) can be combined to a serial connection; afterwards the Bode plots of all elements and their serial connection can be calculated and displayed in various ways. Furthermore various characteristic values can be determined which allow a stability analysis of the corresponding closed-loop system.
	FuzzyPID	The FuzzyPID program allows an interactive design of a Fuzzy-PI-resp. Fuzzy-PD-controller for a standard control loop with a linear plant. The closed-loop system can be simulated representing all time responses graphically. FuzzyPID is especially recommended for first experiments in the field of Fuzzy Control as well as for demonstration and training purposes. More complex control loops should be designed with BORIS. Hints for using FuzzyPID you find in the corresponding online help.
	FRED	The WinFACT module to measure a frequency response is called FRED. It enables you to record an experimentally created frequency response. For this a wobble generator, which generates the different frequencies, is composed by a WinFACT driver and its supported hardware.
	FALCO	The C source code generator FALCO allows the production of ANSI C code for Fuzzy systems, which were developed with the help of the WinFACT Fuzzy Shell FLOP. The C source code is divided into two files. One of them contains the implementation, the other one is the associated header file; it defines the interface for your own application.
	FLOP	The fuzzy shell FLOP (Fuzzy Logic Operating Program) allows the design and the analysis of rule based systems on the basis of fuzzy logic. In this way designed fuzzy systems can later on be integrated into the block-oriented simulation BORIS.
	INGO	The WinFACT-module INGO allows the graphical presentation of all types of WinFACT-graphic files. For the export of graphics INGO offers an export function based on the Windows Metafile Format (WMF files). This format allows a later processing of the graphics with other Windows applications without loss of quality. The printer output of INGO allows the sizing of the axes in mm so that a manual analysis of the printing is easily done.
	RESY	RESY allows the analysis, design and simulation of linear standard control loops. Controller and plant can be built-up stepwise from linear standard elements. For time domain as well as for frequency

domain characteristic values can be determined to get an idea of the dynamic behaviour of the system. The frequency response can be represented as a Bode plot or a Nyquist plot.



#### LISA

LISA allows the analysis of linear systems with input  $u(t)$  and output  $y(t)$  given in form of a transfer function with dead time. Alternatively the system can be configured in form of a block list, i. e. a serial combination of linear standard transfer elements. If the system is specified via block list the corresponding total transfer function is calculated internally. The system analysis covers the calculation of the step response, the representation of the frequency response in form of a Bode plot or a Nyquist plot as well as the calculation of the root locus and of poles and zeros.



#### SIM-Trainer

SIM-Trainer is a program designed especially for training purposes to learn the way how to deal with step responses of simple or combined linear systems as well as with the analysis of their system dynamic. For this purpose any number of linear standard elements (P-, PT1-, PT2-element etc.) can be combined to a serial connection; afterwards the step responses of all elements, their serial connection and the corresponding closed-loop system can be calculated and displayed. Furthermore various characteristic values of the step responses can be determined which allow a judgement of the system dynamic.



#### SUSY

The WinFACT module SUSY allows the handling with single input-/single output-state space systems which can be transformed to a closed-loop system by inserting a linear state-feedback controller. The state-feedback controllers can be designed by Riccati design, pole placement or by hand.



#### IDA

IDA allows the identification of linear systems based on measured input/output data. The input signal can be of any type. For identification the method of multiple integration is used which is especially characterized by its robustness against measurement noise. Numerator order and denominator order can be user-specified or determined automatically by the program. The dead time is also calculated automatically within a lower and upper limit specified by the user.



#### WFSETUP

WFSETUP allows the modification of some cross-program settings (colours etc....).



#### WFINFO

WFINFO provides information about the installed WinFACT-version, e. g. license type, program restrictions (e. g. regarding lite version) or installed add-on components.

---

## System identification by IDA

---

IDA allows the automatic identification of linear systems by measured input/output data. The input signal used for identification can be of any kind. The numerical algorithm used for identification is the *multiple integration method*, which especially shows very great robustness in the case of noisy data. The mathematical model found by IDA is a rational transfer function

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} e^{-T_t s}.$$

The order of the numerator  $m$  and the denominator  $n$  may be set by user before identification or they are determined by IDA automatically. The dead time  $T_t$  also is determined automatically after the lower and upper bound of the dead time have been specified by the user.

The measured data for system input and output have to exist in separate SIM-files. To guarantee correct results it is very important that input and output data were measured at the same *sampling points* so that both files for system input and output contain the same number of data pairs. IDA automatically tests the data files for correctness.

The input file can be opened by the *File / Input signal x(t)...* menu option or the toolbar of the program. The output file is opened by *File / Output signal y(t)* or the toolbar. After reading both files IDA displays the data in its graphic window.

Next you have to specify the control parameters for the identification by the menu option *File / Control parameters...* or IDA's toolbar. The control parameter dialog includes the following data:

- the (maximum) order  $m$  of the numerator of  $G(s)$ ,
- the (maximum) order  $n$  of the denominator of  $G(s)$ ,
- the *identification mode*,
- the *time window* used for identification,
- the control parameters for the calculation of the systems *dead time*.

Normally the identification mode *Single* is selected. In this case the identification is executed for the numerator and denominator order selected in the control parameter dialog. In case of other identification modes the optimal transfer function is calculated for all combinations of numerator resp. denominator orders beginning with 0 up to the order selected in the control parameter dialog. So the optimal orders are found by IDA automatically. This kind of automatic identification can be cancelled by the user after each step.

If a system containing dead time is to be identified, the settings within the *dead time calculation* group box are of importance. These settings determine the discrete values for the dead time  $T_t$  the identification is executed for:  $T_{t \min}$  determines the lower bound,  $T_{t \max}$  the upper bound and *Intermediate steps* the number of intermediate steps.

After selection of control parameters the identification can be started by the *Approximation* menu option or the toolbar. The calculation time for the identification depends on the order of the transfer function, on the number of measurement data and of course on your computers

performance. Normally the results are available after a few seconds. If an identification mode with automatic adjustment of the numerator/denominator or with dead time calculation has been selected, the calculation time rises correspondingly.

After identification is complete the results are displayed graphically. Displayed are

- the measured input data (red dashed line),
- the measured output data (green dashed line),
- the output data based on the identified mathematical model (green line).

### Sample files

CHEN3\_X.SIM, CHEN3\_Y.SIM:

Sample system with  $m=6$  and  $n=8$ , can be approximated very well by a system with  $m=3$ ,  $n=4$

EX1\_X.SIM, EX1\_Y.SIM:

Sample system with  $m=1$  and  $n=2$ , the input signal is an exponential function in this case

N4\_X.SIM, N4\_Y.SIM:

Sample system with all-pass behaviour ( $m=2$  and  $n=4$ ).

RAUSCH\_X.SIM, RAUSCH\_Y.SIM:

Sample system with noisy data ( $m=0$  and  $n=2$ ).

SIN\_X.SIM, SIN\_Y.SIM:

Sample system with a sinus input ( $m=0$  and  $n=2$ ).

---

## Linear system analysis by LISA

---

LISA enables you to analyse linear dynamic systems with input  $u(t)$  and output  $y(t)$  given by a rational transfer function with dead time

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} e^{-T_s}.$$

LISA offers the following features:

- Calculation of the systems step response
- Calculation of the systems frequency response as a Bode plot
- Calculation of the systems frequency response as a Nyquist plot
- Calculation of the systems root locus
- Calculation of the systems zeros and poles (eigen values)

LISA has the so called *multiple document interface* so that the same system can be displayed in different modes at one time, with different scalings etc.

After opening a new display window the display mode *step response* is preset. This mode can be changed by the *Display* menu option or the toolbar. Step responses as well as Bode or Nyquist plots can be saved in specific WinFACT files of SIM, BD or OK type (not available in demo version!). For this purpose choose the *Save* submenu.

All display modes except the Pole/zero output offer a special zoom mode. This mode enables the enlargement of a user specified rectangular section of the graphic by the mouse. Simply click with the left mouse button at the upper left corner of the rectangle you want to zoom and then while keeping the mouse button down zoom up the rectangle. After reaching the lower right corner release the mouse button and the selected area will be displayed again now filling the whole window. Internally this is realized by setting the scaling to manual mode. So to show the whole curves again just set the scaling mode back to automatic or click on the corresponding toolbar button.

### Sample files

LISA1.UFK: System with  $m=1$ ,  $n=2$  and all-pass behaviour  
 LISA2.UFK: Same as LISA1.UFK with additional dead time  
 LISA3.UFK: System with  $m=6$ ,  $n=8$  and all-pass behaviour

---

## Designing linear control systems by RESY

---

RESY enables you to analyse, design and simulate linear control systems with one feedback loop of the following structure:

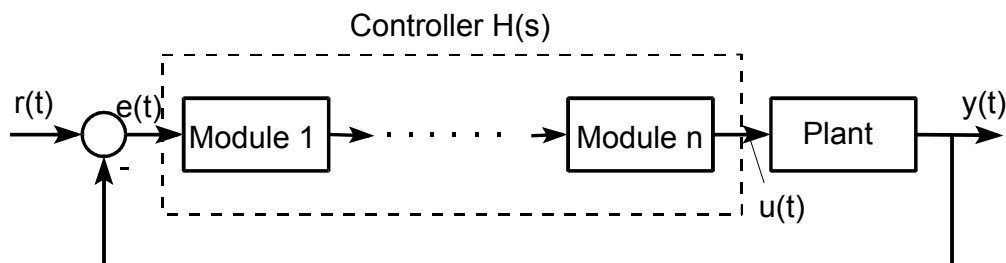


Image 7. Linear control system with one feedback loop

The plant has to be given in form of a rational transfer function  $G(s)$ . The controller can be built up step by step by using the following standard modules:

- P-, I-, PI-, PD- and PID-components with transfer function

$$H_i(s) = K_R \left( 1 + \frac{1}{T_N s} + \frac{T_V s}{1 + T_{VZ} s} \right)$$

- Lead-Lag-components with transfer function

$$H_{i,\text{Lead}}(s) = \frac{1 + \frac{s}{\omega_i}}{1 + \frac{s}{m\omega_i}} \quad \text{resp.} \quad H_{i,\text{Lag}}(s) = \frac{1 + \frac{s}{m\omega_i}}{1 + \frac{s}{\omega_i}}$$

- General rational transfer function with dead time

$$H_i(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} e^{-Ts}$$

Based on these data RESY calculates and displays

- the controllers total transfer function  $H(s)$ ,
- the open loop systems transfer function  $L(s) = G(s)H(s)$ ,
- the closed loop systems transfer function  $T(s) = L(s) / (1 + L(s))$ ,
- the corresponding frequency responses  $G(j\omega)$ ,  $H(j\omega)$ ,  $L(j\omega)$ ,  $T(j\omega)$ ,

and in addition in time range for the case of a step function  $r(t)$  the responses of

- the control error  $e(t)$ ,
- the control output  $u(t)$ ,
- the controlled variable  $y_T(t)$

and the step response  $y_G(t)$  of the plant itself.

In frequency range as well as in time range some characteristic values can be calculated that serve for the characterization of the closed loops dynamics. These values are in time range:

- The control variables overshoot  $M_p$  which gives the maximum value of the control variable during the step response.
- The settling time  $T_a$  corresponding to a 10%-tolerance range around the stationary value of the controlled variable. This time serves as a measure for the systems dynamic.
- The stationary value for the control error  $e(t \rightarrow \infty)$ .
- The maximum of the control output  $u_{\max}$ . This is the maximum value  $u(t)$  taken during the step response.

The open loop criteria in frequency range are:

- The gain crossover frequency  $\omega_c$ . This is the frequency where the gain of the open loop just crosses the 0 dB line.
- The phase margin  $\Phi_r$ . This is the difference between the open loops phase and the  $-180^\circ$  line at the gain crossover frequency. This value serves as a criterium for the closed loop systems stability.
- The amplitude margin  $A_r$ . This is the negative value of the open loops gain at that frequency where the phase crosses the  $-180^\circ$  line.

The closed loop criteria in frequency range are:

- The band width  $\omega_b$ . This is the frequency where the closed loops gain crosses the -3 dB line.
- The frequency overshoot  $M_m$ . This is the maximum value of the closed loop systems gain.

The frequency response can be displayed alternatively as a Bode or Nyquist plot.

To configure the closed loop system first you have to define the plants transfer function. This can be done by manual input or by reading the data from an external UFK-file. Manual input or later modification is selected by the *Plant / Edit plant* menu option or the corresponding toolbar button.

After definition of the plant the controller can be built up step by step. For that the submenu *Controller* is used. It offers the following options:

- Inserting new controller modules,
- Deleting existing controller modules,
- Modification of existing controller modules,
- Listing all existing controller modules.

The selection of the controller modules that has to be deleted resp. modified is made by a corresponding list dialog.

#### Sample file

RESY1.SCL

---

## Simulation and design systems in state space representation by SUSY

---

The WinFACT module SUSY helps you in analysis, design and simulation of linear dynamic systems in state space representation of the form

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{b}u, \quad y = \underline{c}^T \underline{x} + du$$

which can be completed to a closed loop system by an additional linear state controller

$$u = -\underline{k}^T \underline{x} + Vr.$$

SUSY allows the analysis of systems up to 20th order. The main features of SUSY are:

- Comfortable and clearly arranged input and modification of system models
- Display of simulation results by single trajectories, trajectory fields or time responses
- Parallel handling of multiple systems
- Parallel handling of a system with different controllers (Piping)
- Quick change between open and closed loop system
- Calculation of open and closed loops eigenvalues



- Structured system output in matrix form
- Design of state controllers by manual, by pole placement or Riccati design
- ASCII-document generation of all interesting data

The basic structure of the closed loop system handled by SUSY is shown by the following figure.

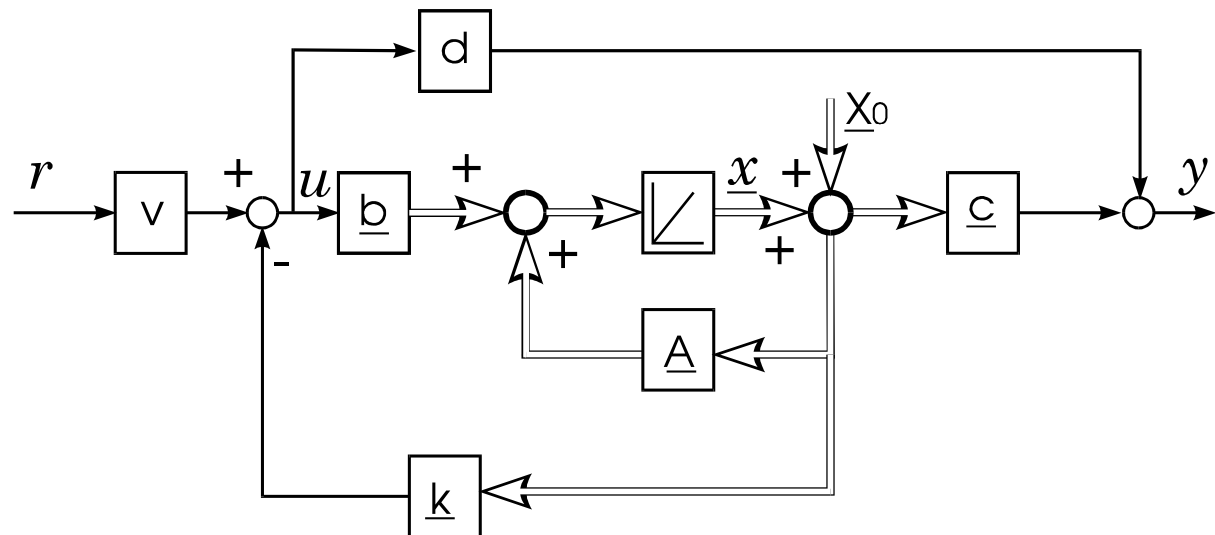


Image 8. Closed loop system with state controller

The variables have the following meanings:

- $\underline{A}$  System matrix of the open loop system
- $\underline{b}$  Input vector
- $\underline{c}$  Output vector
- $d$  Direct influence of the input on the output
- $\underline{x}_0$  Initial values of the state variables
- $V$  Gain (Precompensator) to avoid a stationary control error
- $\underline{k}$  Control vector (state controller)

---

## The fuzzy shell FLOP

---

The fuzzy shell FLOP (Fuzzy Logic Operating Program) enables the design and analysis of rule based systems represented by fuzzy logic. The module offers the following features:

- Definition of linguistic variables and linguistic terms
- Operations on fuzzy sets
- Evaluation of membership values
- Construction of rule bases
- Evaluation of inference rules

- Calculation and graphical display of characteristic curves and 3D-transfer characteristics
- Simulation based on external data
- Programming of microcontroller boards (in combination with specific hardware)
- Design of fuzzy data files for the simulation system BORIS

Different types of operators, inference mechanisms and defuzzification strategies are available for all kinds of operations. In general, the results are presented in a graphical way. For the representation of fuzzy sets three types of membership functions (triangular, trapezoid and singleton) are available. Image 9 shows the user interface of the fuzzy shell FLOP.

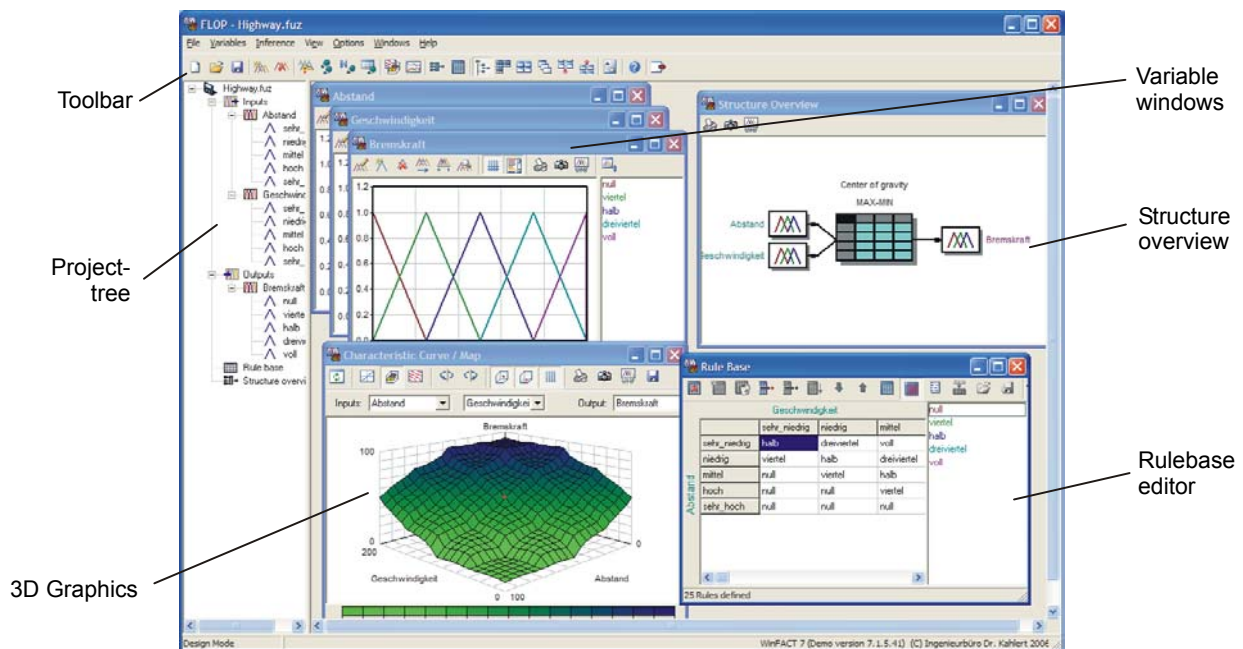


Image 9. User interface of the fuzzy shell FLOP

## Editing linguistic variables

For the clearness of the fuzzy shells user interface it is of great importance that all available data – the currently defined linguistic variables, their fuzzy sets and the rule base – are presented at each moment. Each linguistic variable is displayed in its own window which can be moved, enlarged, reduced, hidden or shown again.

Each linguistic variable is represented by the following items:

- the *type* of the variable (input or output corresponding to premise or conclusion of a rule)
- its name (maximum number of characters: 15)
- its numerical *range*

A new linguistic variable is inserted by the menu option *Variables / New linguistic variable...* or the corresponding toolbar button.

New linguistic terms (fuzzy sets) of a linguistic variable can be inserted by the menu option *Variables / New fuzzy set* or the toolbar. The parameters of all existing fuzzy sets can be modified by the central fuzzy set dialog which can be reached in different ways:

- By a click with the right mouse button on the variable in the project tree and choosing the menu option *Edit variable...*
- By a double click with the left mouse button within the corresponding variable display window. This is the most recommendable way, because the selected variable is preset within the dialog in this case.

Within this dialog the following actions are available:

- Modification of all membership functions of all linguistic variables
- Deleting and renaming of linguistic terms

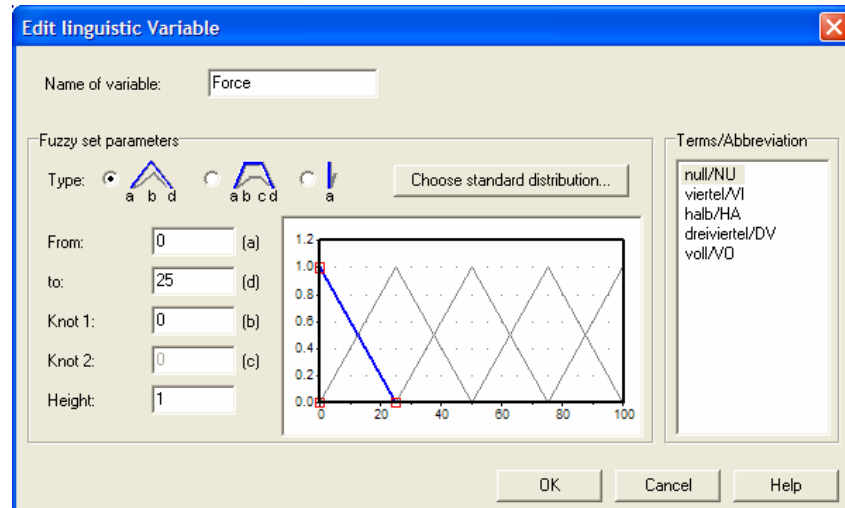


Image 10. Dialog for editing fuzzy sets

All linguistic terms of the current variable are displayed graphically within the corresponding window. The selected membership function is highlighted by its blue colour and red markers at its knots. All changes done by the users are recorded immediately. If the membership function type *Singleton* is selected, only the edit field named *From* has to be filled. The edit field *Knot 2* is only enabled in the case of *trapezoid* membership functions. Changes within the edit field *Height* lead to subnormal fuzzy sets. Normally these types of fuzzy sets should be used with care.

Image 11 gives an overview of the characteristic points of all types of membership functions.

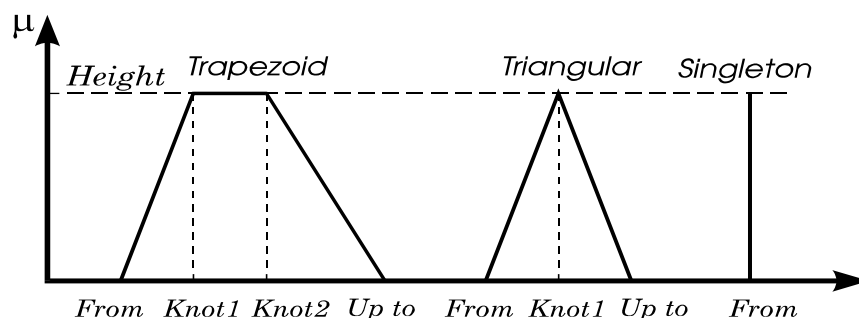


Image 11. Characteristic values of all types of membership functions

As an alternative to the numerical input of the characteristic values a mouse based modification of the fuzzy sets is available. To do so, first click at the desired fuzzy set with the left mouse button so that the fuzzy set will change to blue and red markers will appear at

the characteristic points. Now you can move these points by clicking at them and moving them while keeping the mouse button pressed. The values within the corresponding edit fields are actualized automatically while the markers are moved. Changes of the height of a fuzzy set have always to be made by numerical input.

Another option of great importance is the *Choose standard distribution...* button. Pressing this button leads to a fast standard distribution of the fuzzy sets of the currently selected variable. This standard distribution includes symmetrical triangular fuzzy sets with full overlap.

## Defining the rule base

The menu option *View / Rule base...* leads to the rule base editor. The editor windows as well as the variable windows can be moved, enlarged, reduced or hidden at any time you want. It also is part of the list of all current windows under the *Windows* submenu of FLOP. Image 12 shows the rule base editor immediately after it has been opened for the first time. We see that the rule base is organized in table form where each line represents one rule. Columns for input variables (rule premises) are turquoise, columns with output variables (rule conclusions) are violet.

The definition and modification of rules are realized completely by mouse actions. Therefore the list box at the right margin of the editor contains all linguistic terms of the selected variable. By a double click with the left mouse button on a term this is taken into the rule table. The status line of the editor shows the item currently selected (number of the rule, name of the linguistic variable and the rule weight) and the number of all rules currently defined.

The *Clean rule base* button allows you to delete all invalid rules of the current rule base. Invalid are those rules that do not contain an entry in at least one output variable. In addition to this a cleaning action is carried out automatically always a file is saved.

If the weight of a rule has to be modified, first select the corresponding item within the last column of the rule base. After this at the right margin a scroll bar appears which enables you to modify the current weight to any value between 0 and 1.

In addition to the modification of single entries of the rule base FLOP offers you powerful features for the parallel modification of groups of entries which are especially useful for the handling of complex fuzzy systems with a lot of rules:

- By keeping the <Shift>-key pressed when selecting items of the rule base several items lying one beneath the other can be selected at the same time.
- By keeping the <Ctrl>-key pressed when selecting items of the rule base several items of the same column – which do not have to lie one beneath the other – can be selected at the same time.

In the following all items selected in this way can be deleted, filled by the same linguistic term or weight or moved to another location at the same time. If you want to copy the selected items, simply move them to the target position while keeping the left mouse button pressed. After reaching the target position release the mouse button again.

In the same way complete rules can be selected. For that simply select the corresponding items within the first columns of the rule base editor.

For fuzzy systems with two input variables and one output variable the rule base editor can also be represented in a *matrix form*. This form is especially efficient because you only have to fill the conclusion entry of each rule, not the premise entries. To switch between both table

and matrix form, use the corresponding buttons *Table mode* resp. *Matrix mode* of the toolbar in the rulebase window. Image 13 shows the rule base editor in matrix form filled with all rules.

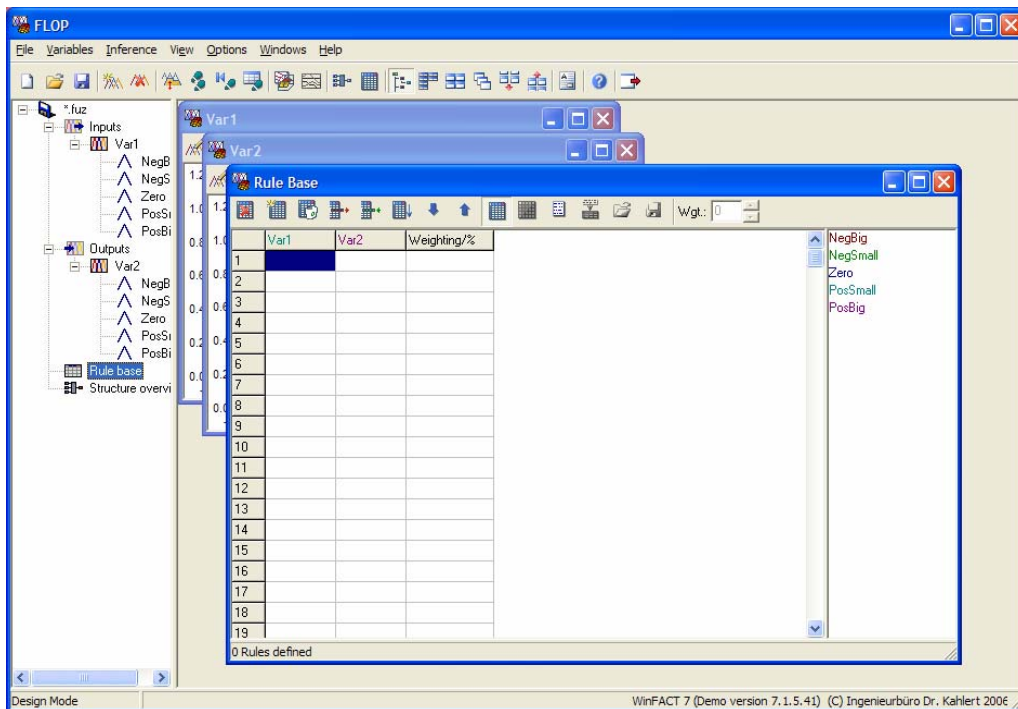


Image 12. Rule base editor immediately after opening

If no specific rule is to define in matrix form, simply let the item empty. The extended editor options (multiple selection of entries, ...) are not available in matrix form.

In both rule base editors a text mode is available which allows to formulate the rules in ASCII text form. The text editor is started by pressing the *Text mode* button of the toolbar.

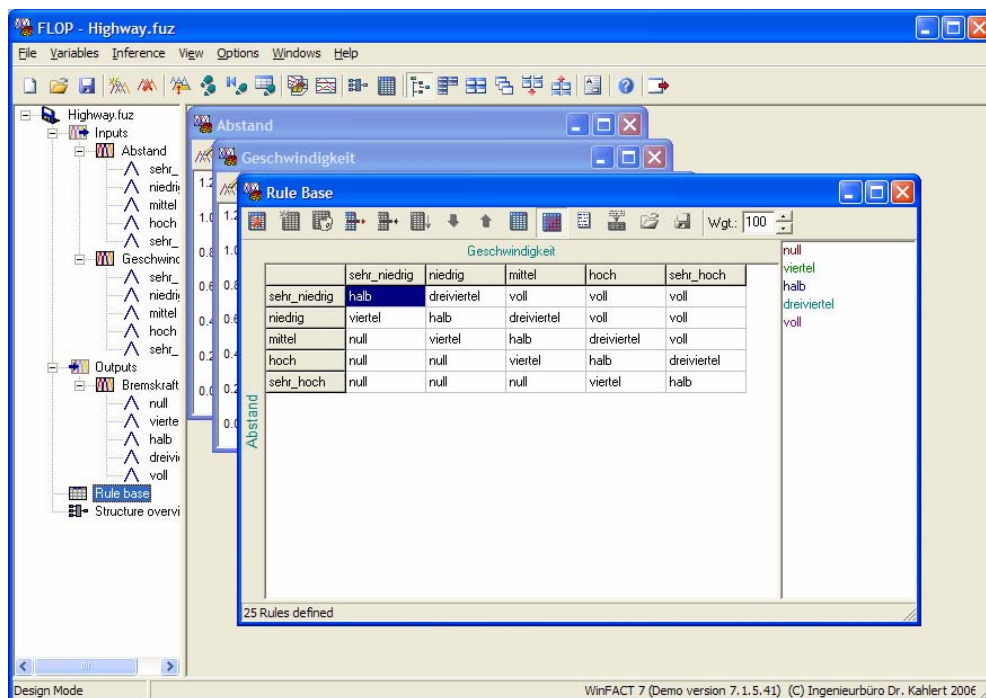


Image 13. Rule base editor in matrix form

## System analysis in the debug mode

After the specification of all linguistic in- and output variables, the rule base, the operation operators as well as the inference mechanism and the defuzzification method the fuzzy system can be tested for its correct functioning. For that various tools for analysis are available which are especially advantageous in the interactive debug mode of FLOP. In the following these tools are described by the example of a fuzzy system which controls the deceleration process on a highway (image 14). We are in the second car which follows the first one with the defined speed  $v$  (1. input variable of the fuzzy system) and the distance  $d$  (2. input variable of the fuzzy system). If the leading car decelerates, we also have to slow down. The brake power  $F$  (output variable of the fuzzy system) should depend on our speed and the distance to the car in front. Tendentially it can be said that the higher the speed and the smaller the distance to the car in front the more powerful the braking has to be. This **causal conjunction** is emulated in the rule base of the sample file.

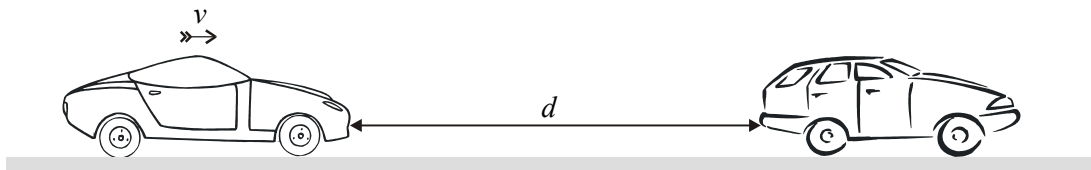




Image 14. Highway example

## Activating the interactive debug mode



For the system analysis the program has to be in the interactive debug mode which can be selected via the menu option *Inference / Interactive debug mode* resp. the  button. The interactive debug mode is confirmed in the status bar of the program by a corresponding message. As long as FLOP is in the debug mode the structure of the fuzzy system cannot be modified (e. g. inserting or deleting of linguistic variables). It is, however, possible to modify e. g. single fuzzy sets and to change the rule base (inserting and deleting rules or changing rules).

## Variable window in the debug mode

The variable windows of the in- and output variables look differently after the debug mode has been activated. Windows of input variables get a scrollbar at the lower border of the window as well as an edit field with button below the term list. Via the scrollbar or the edit field (and a click on the button on the right of the edit field) the current value of the input variable can be set. The default resolution for the scrolling process is 1% of the range of values of the variable. This value can be changed via the  button. In addition the current input value is represented by a bar in the paint area of the window. In the term list of the window the evaluated degree of membership for the current input value is displayed beside every term (image 15).

In the debug mode windows of output variables have a status bar at their lower border which shows the current crisp output value of the variable. At every modification of an input variable of the fuzzy system all output variables are automatically recalculated and the corresponding variable windows are refreshed. In addition the crisp output value is represented by a bar in the paint area of the window. Besides the resulting fuzzy set is colour marked. In the term list, finally, the current matches of degree for all terms of the output variable are listed (image 16).

That way the behaviour of the fuzzy system can be analyzed step by step by modifying single input values and observing the effect of this modification on the output values (single step

analysis). If only the numerical values of in- and output variables are of interest or the system contains many variables it might be useful to minimize some or all variable windows in such a way that only the relevant information is represented. For that the client area of the variable window can be hidden via the  button so that only the scrollbar (input variables) resp. the status bar (output variables) is visible. By selecting the main menu option *Windows / Arrange all variable windows in shut mode* or via the  button of the main window it is possible to minimize and automatically arrange the variable windows all at once.

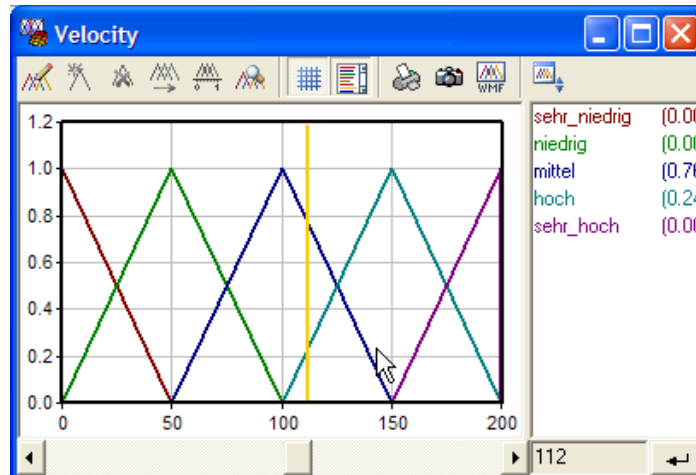


Image 15. Variable window for input variables in debug mode

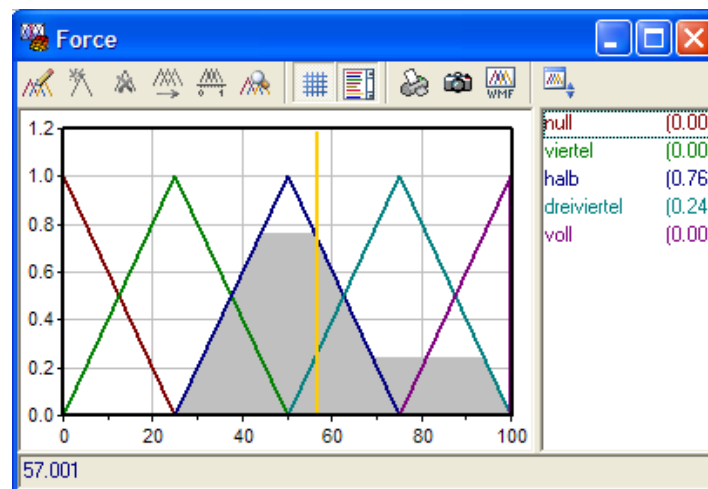


Image 16. Variable window for output variables in debug mode

### Sample files

DEMO1.FUZ

DEMO2.FUZ

DEMO3.FUZ

DEMO4.FUZ (in combination with DEMO4.FSI)

DEMO7.FUZ



---

## Generating Fuzzy-C-Code by FALCO

---

The C-Code-Generator FALCO (**F**uzzy **A**pplication **C**-**C**odegenerator) enables you to generate ANSI-C-Code for fuzzy systems designed by the fuzzy shell FLOP. The C-Code mainly contains a C function which can be included in user written software later or used for the programming of user specific hardware. FALCO offers the following features:

- Comfortable editor for FUZ- and C-files with Multi-Document-Interface containing all standard editor options as copy and paste, search and replace etc.
- Parallel editing of multiple systems
- The code generated by FALCO only consists of one file – no further library is necessary. The code is very compact and contains only the instructions, operations and variables needed for the specific system.
- The C-types for the linguistic variables (inputs and outputs of the fuzzy system) and the membership values can be selected by the user (short int, long int, float, ...)
- On request a main function can be generated for the debugging of the generated code
- Compiling, linking and running of the generated code directly from the FALCO environment; selection between different compilers and/or linkers possible
- Command line calling up FALCO from the FLOP environment

The fuzzy system to be compiled to C code must exist as a FUZ-file like generated by the fuzzy shell FLOP. This file can be opened and loaded into the active editor window by the *File / Open* menu option. A possibly already existing file within this window is overwritten. Shall the file be loaded into a new window, this window first has to be created by the *File / New* menu option. By the *File / Save* command the contents of the active editor window is saved, by the *File / Print...* command it is printed on the standard printer.

After a FUZ-file was loaded the corresponding code can be generated directly from this file. By default the code uses 4-byte floating point numbers. However, via the menu option *Code Generation / Settings...* the data type can be changed to 2-byte floating point or fixed point numbers. Especially it is possible to design your own fuzzy controllers by using templates (code templates) resp. modify the procedure of an existing controller template.

This feature is also very useful for the direct generation of a user defined main-function. You create your own template and leave it to the code generator to complete it for different fuzzy controllers.

The code generation is started via the menu option *Code Generation / Generate code*. The generated code is displayed on several registers within the same document window. The names of the registers are derived from the name of the file (e. g. z.B. Demo4.c, Demo4.h).

The generated C-Code mainly consists of a function the name of which is derived from the name of the FUZ-file. Input parameters of this function are the crisp inputs of the fuzzy system, output parameters the crisp outputs. The function header of the *DEMO4.FUZ* example contains the following lines:

```

void DEMO4_F2_calc(
    const NumTypeF2_t i0,
    const NumTypeF2_t i1,
    NumTypeF2_t *o0)
{
    NumTypeF2_t ai[2];
    NumTypeF2_t ao[1];
    ai[0]=i0;
    ai[1]=i1;
    FCF2_calc(&DEMO4_F2_FC, &DEMO4_F2_FCMem, ai, ao);
    *o0=ao[0];
}

```

In this case *F2* which represents the used data type (2-byte floating point numbers) was added to the identifiers (option: *expand by number format specifier*). Appropriate converting routines are available for the data types so that the function call can be executed without any problem. The function is only a wrapper function. It simplifies the call of the used function `FCF2_calc()` which contains the actual fuzzy controller and is located in the file `fuzzy_tt.c`. *tt* indicates the number format specifier (in the example *tt* is the same as *F2*).

---

## Designing Fuzzy-PID-Controllers by FuzzyPID

---

The module FUZZYPID offers an interactive design of a Fuzzy-PI- or Fuzzy-PD-Controller for a closed loop system with single feedback. The plant must be of linear type. The closed loop system can be simulated with a simultaneous graphical output of all interesting variables. Therefore FuzzyPID is especially suitable for first steps in the area of fuzzy control as well as for demonstrations and representations in education and research. More complex fuzzy control systems should be designed by the fuzzy shell FLOP and the simulation module BORIS.

The concept and handling of FuzzyPID is nearly identical to the fuzzy shell FLOP. In addition the following restrictions have to be kept:

- The linguistic variables have the names  
 $e$  for the control error  $e$ ,  
 $de/dt$  for the derivation  $\dot{e}$  of the control error,  
 $u$  for the control output  $u$  (Fuzzy-PD-Controller),  
 $du/dt$  for the derivation  $\dot{u}$  of the control output (Fuzzy-PI-Controller).  
 These names are declared internally and cannot be changed by the user. Also a deletion of linguistic variables is not allowed.
- The number of linguistic terms per linguistic variable is internally set to five. This number cannot be changed. The terms have the same names for all variables. These are  
 $Negative\_Big$  (Abbreviation --)  
 $Negative\_Small$  (Abbreviation -)  
 $Zero$  (Abbreviation 0)

*Positive\_Small* (Abbreviation +)

*Positive\_Big* (Abbreviation ++).

These names and abbreviations cannot be changed.

- The controller and thus the rule base must have two inputs,  $e$  and  $\dot{e}$ . If a Fuzzy-P-Controller has to be realized, first select the Fuzzy-PD-Controller type and then make the generated output independent of  $\dot{e}$ . This can be achieved by choosing the same linguistic term for each entry in a rule base line.
- All rules have the weight of one. Negated premises are not allowed.

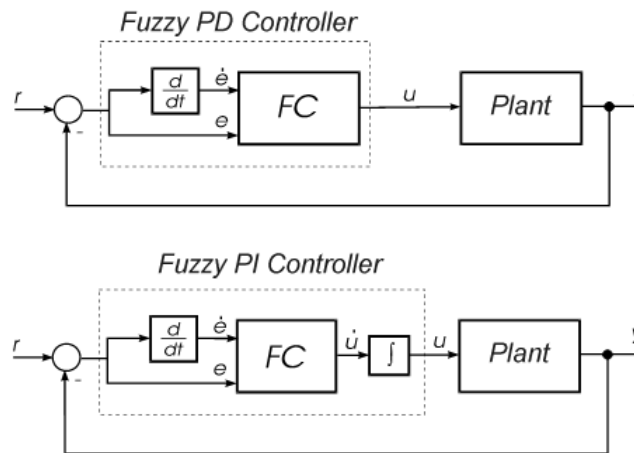


Image 17. Closed loop system with single feedback with Fuzzy-PD-Controller (top) resp. Fuzzy-PI-Controller (bottom)

All relevant information is presented within the FuzzyPID program window simultaneously. These are:

- The structure of the underlying closed loop system,
- the membership function (fuzzy sets) of the controllers inputs and outputs,
- the current rule base,
- the simulation results.

The underlying closed loop system has a single feedback of the controlled variable. For first experiments in the area of fuzzy control the plant model can be selected from five predefined models. The following models are available by the *Closed loop / Plant* command:

- PT<sub>2</sub>-Plant (non oscillating)

This type of plant has a gain of one and two real eigen values. The corresponding time constants are  $T_1 \approx 0.2$  s and  $T_2 \approx 5$  s. The corresponding transfer function of the plant is

$$G(s) = \frac{1}{s^2 + 6s + 1}$$

- PT<sub>2</sub>-Plant (oscillating)

This plant has two complex eigen values and therefore is able to oscillate. The corresponding damping of the plant is  $\zeta = 0.25$ , the eigen frequency is  $\omega_n = 1$ . The corresponding transfer function is

$$G(s) = \frac{1}{s^2 + 0.5s + 1}$$

- PT<sub>1</sub>-I-Plant

This plant is a serial combination of a PT<sub>1</sub>-Element with a time constant of  $T = 10$  s and an integrator. The corresponding transfer function is given by

$$G(s) = \frac{0.2}{s(1 + 10s)}$$

- PT<sub>1</sub>-T<sub>t</sub>-Plant

This plant is a serial combination of a PT<sub>1</sub>-Element and a dead time. The PT<sub>1</sub>-Element has a time constant of  $T = 5$  s. The dead time is  $T_t = 3$  s. Therefore the corresponding transfer function is given by

$$G(s) = \frac{1}{1 + 5s} e^{-3s}$$

- PT<sub>2</sub>- Plant (time variant)

This type of plant has the same structure as the oscillating PT<sub>2</sub>-Plant, but combined with a time variant gain. The transfer function is given by

$$G(s) = \frac{K(t)}{s^2 + 0.5s + 1}$$

For  $0 \leq t \leq 5$  the gain increases linear from 1 up to 2 and then decreases again linear to 1 for  $5 \leq t \leq 10$  :

$$K(t) = \begin{cases} 1 + t/5 & \text{for } 0 \leq t \leq 5 \\ 2 - (t-5)/5 & \text{for } 5 < t \leq 10 \\ 1 & \text{for } t > 10 \end{cases}$$

For all these plants a modification of parameters is not allowed.

In addition to the selection of such a predefined plant the last three options within the *Closed loop / Plant* submenu allow the input of a user defined linear plant in form of a rational transfer function. This transfer function can be read from keyboard or an external file of UFK-type as well as stored in a UFK-file after modification.

Available controller types are Fuzzy-PI- and Fuzzy-PD-Controllers. Inputs of both controller types are the control error  $e$  and its derivative  $\dot{e}$ , the output is the control output  $u$  (Fuzzy-PD-Controller) resp. its derivative  $\dot{u}$  (Fuzzy-PI-Controller). The controller type is selected by the *Closed loop / Controller type PI* resp. *Closed loop / Controller type PD* command. By the *Closed loop / Simulation parameters...* menu option or the <Strg> <P>-keys an input dialog for the simulation parameters can be called up.

After all membership functions and the rule base are read from keyboard or an external file the simulation can be started. For this purpose two different modes of simulation are available. By the *Simulation / Display all* or the <Strg> <S>-keys the first simulation mode is started. In this mode the complete dynamic behaviour of the fuzzy system is represented graphically. The following information is shown online during the simulation:

- Within the diagrams with the membership functions of the controllers in- and outputs (right margin of the program window) the current values are shown by yellow line cursor. The diagram for the controllers output  $u$  resp.  $\dot{u}$  at the top additionally shows the active output fuzzy sets and their degree of match (hatched blue).
- Within the rule base the active rules are changing their colour to yellow.
- The diagram in the lower left corner shows the time responses of all those variables of the control system that were specified by the *Options* command of the programs main menu:
  - The time response of the plant itself, e. g. open loop without controller (dashed violet curve),
  - the progress of the reference variable  $r(t)$  of the closed loop (full green curve),
  - the manipulated variable  $y(t)$  (full violet curve),
  - the error variable  $e(t)$  (full lightred curve),
  - the derivative  $\dot{e}(t)$  of the error variable (full red curve),
  - the control variable  $u(t)$  (full darkblue curve),
  - the derivative  $\dot{u}(t)$  of the control variable (full lightblue curve). This curve can only be shown in the case of an PI-Controller.

By pressing the *Cancel* button during the simulation the simulation can be stopped at any time.

If only the time response of the system is of interest, the second simulation mode is more recommendable because it runs much faster. This mode can be started by the *Simulation / Time response* command or the <Strg> <V>-keys. This dialog offers a larger part for the display of the time responses and the possibility to scale the axes of the diagram individually. The time response of the plant is shown immediately after the dialog is called up; the simulation of the other variables is started by the *Start* button. It can be stopped at any time by pressing the *Cancel* button.

Because there is no time-consuming representation of the fuzzy sets and the rule base in this simulation mode the whole process proceeds much faster.

### Sample files

The sample file collection includes one sample file for each type of plant that works more or less well and can be used as a base for own experiments on analyzing or improving the controller. The files have the following names:

PT2 . FUZ	Fuzzy-PI-Controller for PT <sub>2</sub> -Plant (non oscillating),
PT2S . FUZ	Fuzzy-PI-Controller for PT <sub>2</sub> -Plant (oscillating),
PT1I . FUZ	Fuzzy-PD-Controller for PT <sub>1</sub> -I-Plant,
PT1TT . FUZ	Fuzzy-PI-Controller for PT <sub>1</sub> -T <sub>t</sub> -Plant,
PT2ZV . FUZ	Fuzzy-PI-Controller for PT <sub>2</sub> -Plant (time variant).

The corresponding controller type (PI resp. PD) is recognized and set by FuzzyPID automatically on the base of the variable names found within the file.

---

## Simulating dynamic systems with BORIS

---

### Overview

The **block-oriented** simulation system BORIS is a powerful tool for the simulation of all kinds of dynamic systems with or without fuzzy components. BORIS offers the following features:

- Extensive system block library:
  - Signal generators (Sinus, Pulse, Noise, test functions)
  - Linear standard elements ( $PT_1$ ,  $PT_2$ , ...)
  - Linear systems of higher order
  - Nonlinear characteristics and functions, algebraic functions
  - Linear and nonlinear standard controllers, adaptive controllers
  - Fuzzy Controller with integrated Fuzzy Debugger
  - Virtual instruments (time responses, trajectory plot, analog and digital meter, oscilloscope, status displays)
  - Action blocks (e. g. potentiometer, switch, industry-PID controller)
  - In- and output of signals from files resp. in files
  - Spectral analysis by Fast Fourier Transformation
  - Statistical functions
  - Digital modules (logic gates, Flip Flops)
- Definition of hierarchical macros (Superblocks)
- Placement of system blocks flexible; extended scrollable workplace
- Different numerical integration methods

### Components of the BORIS main window

Image 18 shows the main window of BORIS with its components.

In addition to the WINDOWS standard components the window contains the following elements:

- A first horizontal toolbar (control toolbar) below the menu. This toolbar contains buttons for the most frequently used commands. Choose the menu item *Options/Toolbars/Show system toolbar* to activate or deactivate the toolbar.
- A second horizontal toolbar (system block toolbar) which is subdivided into several palettes. This toolbar allows the direct access to all system blocks. The palette *Favorites* can be configured by the user (see *configuration of the system block toolbar*). Via the menu item *Options/Toolbars/Show system block toolbar* the toolbar can be activated or deactivated.

- A status bar at the bottom of the window. This line contains information about the current number of system blocks and the number of selected resp. set passive blocks. Additionally the simulation parameters are displayed in the following form  $T = T_{\text{Simu}} (\Delta T)$ . In this term  $T_{\text{Simu}}$  is the simulation time and  $\Delta T$  the simulation step size.

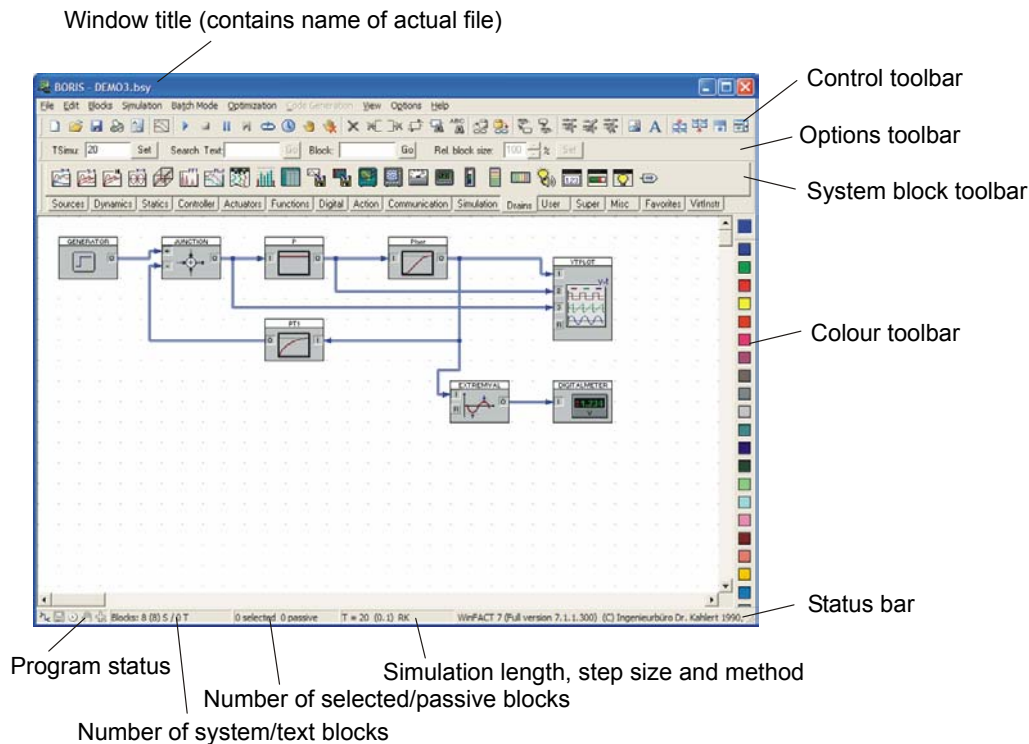







Image 18. Main Window of BORIS

While the simulation is running the progress of the simulation is displayed if this option had not been deactivated. Five icons at the left margin of the status bar inform about the current state of the system:

-  autorouter activated
-  system has not been saved since the last change
-  simulation is running
-  breakpoint activated
-  optimization is running

- The paint window (workplace) for the design of the system structure. This window can be scrolled by its scrollbars in horizontal and vertical direction. After starting BORIS the clipping area is set to its origin. This initial state can be reset at any time by the *Options / View section to origin* command.

The paint window has a grid by default to which all blocks are adjusted with their upper left corner. This automatic adjustment can be deactivated by the *Options / Snap to grid* command. Further the grid itself can be deactivated by the *Options / Show grid* command independent of the status of the block adjustment.

## Inserting and editing system blocks

The following features concerning the block handling are available:



- *Inserting* blocks of the system block library
- *Selecting* single blocks or groups of blocks
- *Moving* single blocks or groups of blocks
- *Deleting* single blocks or groups of blocks
- *Rotating* blocks
- *Copying* and *pasting* of blocks
- Setting the *parameters* of a block
- Changing the *block size*

Most options are available by a context pop-up menu which opens when clicking with the right mouse button.

### **Inserting blocks**

To insert a new block into the current system structure

- click the corresponding button of the system block toolbar or
- select the corresponding menu item of the *Blocks* submenu.

The inserted block appears in the upper left corner of the paint window or - if there is not enough space - a little bit moved to the right or the bottom. Alternatively it can be inserted by Drag & Drop.

### **Selecting blocks**

Before any operation concerning a block or a group of blocks can be executed, the blocks have to be selected first. To select a single block or a group of blocks there are different ways:

- A single block is selected by clicking at it with the left mouse button. If another block is already selected at this time, the selection of that block is deactivated.
- If another block is to select in addition to already selected blocks, click at it with the <Shift>- or <Ctrl>-key pressed. Clicking at an already selected block deselects it.
- Alternatively a group of blocks can be selected by drawing a rectangle with the mouse. This can be realized by moving the mouse within the paint window while keeping the left button pressed. All blocks lying completely within this rectangle are selected.
- By the *Edit / Select all* blocks or the <Strg><A>-key all existing blocks are selected simultaneously.

Clicking at any free location within the paint window with the left mouse button deselects all selected blocks.

### **Moving blocks**

To move a single block or a group of blocks proceed as follows:

1. First select the block resp. the group of blocks to be moved.
2. With the left mouse button click within the block to be moved resp. – if a group of blocks is to move – within any of the blocks of the group and move the mouse while keeping the left button pressed. While moving the cursor takes the form of a cross. If the margin of the paint window is reached the window is scrolled automatically.

After the moving is finished all blocks are adjusted automatically so that no blocks overlap.

## Moving the complete system

During the design of the system structure it often happens that new blocks are to be inserted at the left or top of an already existing structure. Therefore BORIS allows to move the complete system structure (all system and text blocks, all connections) in any direction without selecting all blocks before. The corresponding commands are:

*Options / Move structure right*

*Options / Move structure left*

*Options / Move structure down*

*Options / Move structure up*

Each command moves the system by one grid unit to the corresponding direction.

## Setting blocks passive

Blocks can be set passive. During the simulation process passive blocks are handled as if they were not existent. Passive blocks and connections from/to passive blocks are gray-coloured.

## Deleting blocks

A single block or a group of blocks selected before can be deleted in two ways:

- By the pop-up menu (right mouse button)
- By pressing the corresponding button of the control toolbar.

All input and output connections of the deleted blocks are deleted automatically with their blocks.

## How to rotate blocks

The orientation of a block can be turned by 180° so that the input(s) of the block are on the right side (for example for blocks in a feedback). For this purpose use the *Edit / Rotate block* command. All connections of the block are re-routed automatically.

## Copy and paste

The copy and paste function of BORIS is controlled by a temporary file. All selected blocks, their parameters and the connections between the selected blocks are copied resp. pasted. Two different cases are possible:

If you want to copy a part of the structure *within the same system*, proceed as follows:

- Select the blocks to be copied and then select the *Edit / Copy* menu option or press the corresponding button of the control toolbar. To insert the blocks again use the *Edit / Paste* command or the corresponding toolbar button.

To copy blocks *between different systems* proceed as follows:

- First load the system where you want to copy from, select the blocks and use the *Edit / Copy* command. Now load the system you want to copy to (do not close BORIS before, because BORIS deletes the temporary file if you quit the program!) and use the *Edit / Paste* command. Alternatively of course it is possible to start BORIS twice, load the two systems simultaneously and copy between both BORIS applications.

## How blocks get their parameters

The most simple way to modify the parameters of a block is to double click it with the left mouse button. Alternatively the block can be selected first and then the *Edit / Edit block*

command be executed. Both ways lead to a block specific parameter dialog where the parameter modifications can be made. Image 19 shows as an example the parameter dialog of an oscillating  $PT_2$ -element. By clicking with the right mouse button within an edit field a message box with the valid range is displayed.

For all block types the parameter dialog contains an edit field at its top where the name of the block can be modified. This name is set into the caption of the block window within the paint window. By default this name is identical to the block type name but can be changed by the user. The maximum length of the name is 25 characters. In addition to that all parameter dialogs contain a *Help* button by which a block specific help information can be requested.

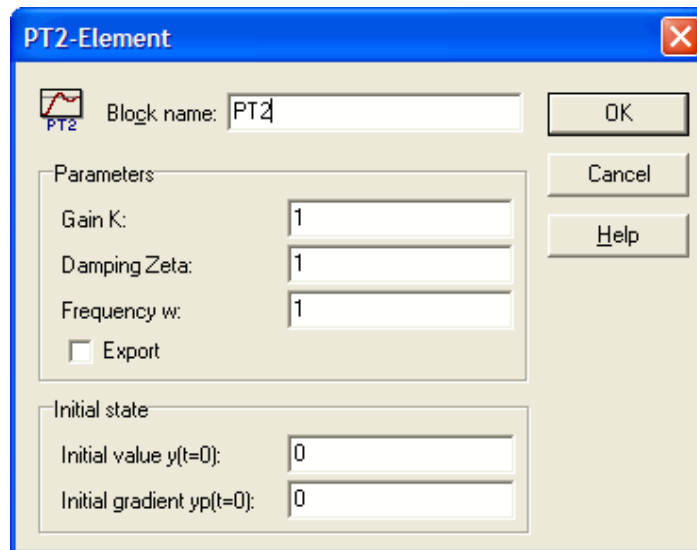


Image 19. Parameter dialog for oscillating  $PT_2$ -element

### Changing the block size

All system blocks can be displayed in different sizes by BORIS. A new inserted block is normally displayed as big as possible (100%). This default setting can be changed by the display dialog (*Options/Customize...*, entry field *Standard block size*).

To change the block-size of a single or of several blocks follow these steps:

1. Click on the menu-item *View/Block size* window to open the block size window.
2. Select the blocks you want to modify.
3. Enter the desired size in the entry field *Relative size* of the block size window and click on the *Apply* button.

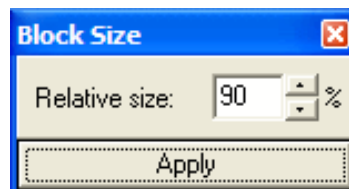


Image20. Block size window

## Connecting blocks

### How to connect two blocks

All connections between blocks are drawn by mouse actions. Because an integrated autorouter automatically routes the connections in a right-angled way and – as far as possible – free of cross points, only the two blocks to be connected have to be selected. In this connection the following principle is to observe:

*Connections are drawn **from output to input!***

So to draw a connection proceed as follows:

1. First select the output square of that block where the connection shall start by clicking at it with the left mouse button. The mouse cursor now changes its symbol to a stylized soldering-iron.
2. Now the input square of the target block can be selected in the same way. If the border of the paint window is reached during the drawing process the window is scrolled automatically. To cancel a connection that you have already begun to draw, simply click at any free position within the paint window.

After a connection was completed correctly, it is drawn automatically by the integrated autorouter including the arrow at its end. The route is calculated in such a way that as far as possible no blocks or other connections are crossed. If nevertheless a crossing occurs, in most cases a small movement of the block can take this back. Connections that belong to the same block output are combined by the autorouter automatically.

Any block output can contain as much connections as you like. Any block input however can naturally have only one input signal. The connections itself can be displayed in different ways (refer to the *Options / Customize...* dialog).

The integrated autorouter can be deactivated by the menu item *Options / Autorouter* so that the connections can manually be drawn.

### Deleting connections

To delete an existing connection, first select the start or target block of the connections. In the following you have two alternatives:

- Output connections of a block are deleted by the *Edit / Connections / Delete output connections* or – more comfortable – the corresponding button of the control toolbar, input connections by the *Edit / Connections / Delete input connections* resp. the toolbar button. In both cases *all* connections of the blocks are deleted simultaneously.
- The second way is to use the popup menu (right mouse button).

## Text blocks and frames

### Using text blocks

Besides the "normal" system blocks BORIS offers text blocks as an opportunity to insert comments into the system structure. These text lines can be of different colours and sizes and may be moved as any other system block. To insert a text block use the *Edit / Insert text block* command or the corresponding button of the control toolbar. Subsequently the default text "Text" appears in the upper left corner of the paint window. By a double click with the left mouse button this text can be modified. By a single click and pressed left mouse button a text block can be moved in the same way as other blocks.

## Using frames

Another important feature for the improved documentation of system structures are *frames*. They represent a way for the graphical combination of system blocks that belong together in any way. Of course frames do not have any influence on the simulation itself.

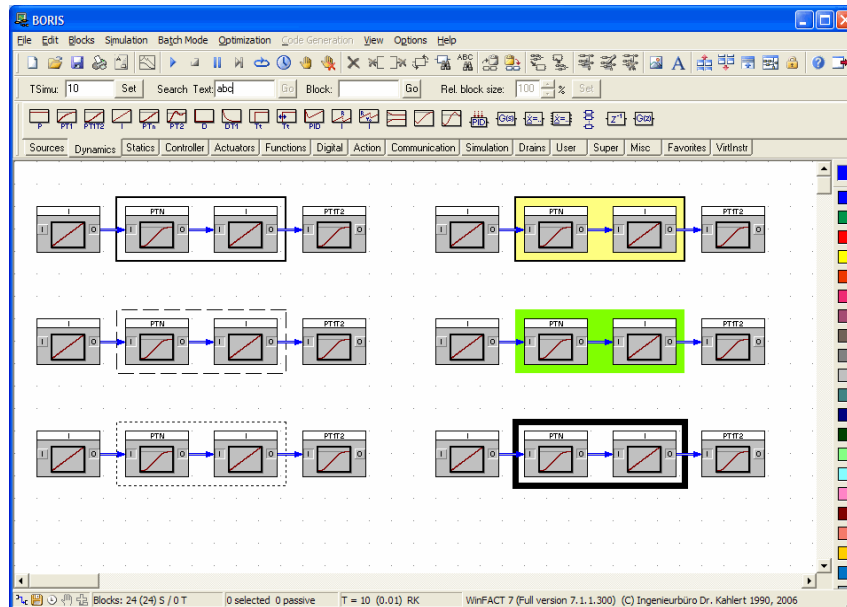


Image 21. Example how to use Frames

To insert a frame...

- first select the blocks to be grouped
- then choose the *Edit / Group frame / Insert frame* menu option or press the corresponding button of the control toolbar.

Attention: Frames are *static* constructs without any link to the blocks within the frame! So they can neither be moved nor are they deleted automatically if the blocks within are deleted.

The inserted frame by default has a distance of eight points to the block surrounding rectangle, a black border with a full line of one point width and no inner colour. These parameters can be modified later. For this purpose proceed as follows:

1. Select at least one block within the frame.
2. Choose the *Edit / Group frame / Edit frame* command of the menu bar.

This command leads to the frames parameter dialog.

To delete a frame proceed as follows:

1. Select at least one block within the frame.
2. Choose the *Edit / Group frame / Delete frame* command of the menu bar.

## Structure overview

When designing complex systems normally not all system blocks are visible within the paint window simultaneously. To get an overview of the designed system without scrolling many times a separate window can be opened that shows all blocks in a zoomed mode so that all blocks are visible. This window is opened by the *View / Structure overview* command or the

corresponding button of the control toolbar. The window can be moved and its size can be reduced or enlarged. As long as the overview window is visible it is actualized after each user action.

Within the overview window the system blocks are drawn in a more simple form without their specific bitmaps. Therefore this window is especially recommendable for the printer output. This output can be realized by the *File / Print...* command. Additionally the system structure can be exported to a WMF- or BMP-file by the *File / Export...* command (not available in the demo version).

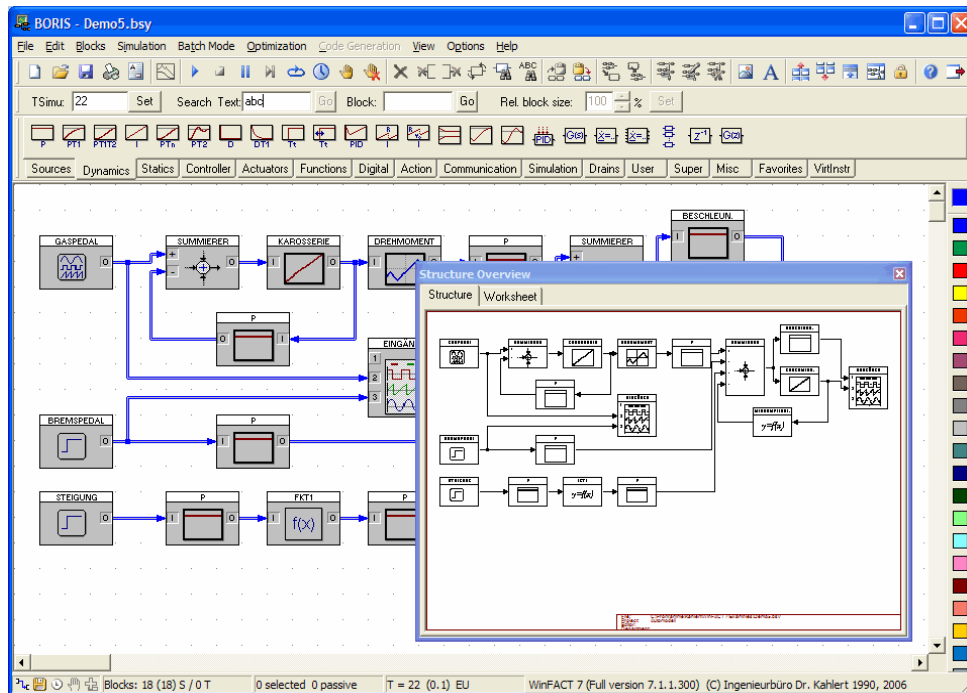


Image 22. Structure overview of a simple system

## Simulation control

### Simulation parameters

Before the simulation can be started, the simulation parameters have to be defined. These are:

- The simulation time  $T_{\text{Simu}}$   
This value determines the time up to which the simulation is executed if it is not cancelled before. The default value is 10.
- The simulation step size  $\Delta T$   
This value determines the discretization step size for the simulation and thus the precision of the simulation results. If  $\Delta T$  is chosen too large, discretization errors occur which in the worst case may lead to numerical instability. Normally  $\Delta T$  should be not greater than 1/10 of the smallest time constant of the simulated system. The default value is 0.01.
- The simulation algorithm  
Determines the choice of the numerical integration method for the simulation of the dynamic system components. Available are the Euler-method, the Runge-Kutta method of 4. order and the matrix exponential method.

- A range check during the simulation

If the option *Check for range overflow* is activated, all block outputs are checked after each simulation step. If at least one of the variables exceeds a value of  $10^{20}$ , the simulation is cancelled and a corresponding warning is displayed. Because this range check needs a certain calculation time, the simulation progresses a little bit slower if this option is activated (this is default).

- Time display in status line









To reach the simulation parameter dialog use the *Simulation / Parameters...* command or the corresponding toolbar button.

All settings concerning real time simulation are only of interest if BORIS is used in combination with a A/D-D/A-card or e. g. a USB-box.




## The BORIS system block library

### Block group *Sources*

This group contains those system block types which produce input signals for the system (e. g. generator, file input).














Icon	Block type	Short description
	GENERATOR	Universal signal generator which can operate in different modes. Besides all kinds of periodic signals (sinus, rectangle, triangle, sawtooth,...) also various test signals (single impulses, step function, ramp,...) as well as noise signals can be generated. In addition to that the desired signal flow can explicitly be specified as formula via the integrated function parser.
	DRIVCURVE	The input block produces a continuous, from 0 to a user-defined amplitude increasing signal, as it is needed, for instance, for the controlled running up of engines.
	DRIVCURVEEXT	This block type represents an extension of the DRIVCURVE block type and can be used for the realization of complex control signals.
	VCO	This block represents a sinus generator ( <i>Voltage Controlled Oscillator</i> ) which frequency is controllable via the block input. The dependence can be linear or exponential. You can use this block type e. g. to design a wobble generator.
	CONST	This block provides a constant output signal.
	FILEINPUT	This block allows the reading of a signal in the form of pairs of values from a file which is then available at the block output. Such a file can be created by various WinFACT-modules or by programs of third-party suppliers (e. g. EXCEL). The corresponding file format is described in the WinFACT-program manual resp. online help.
	TABFILEINPUT	This block type represents an extension of the FILEINPUT block and allows the simultaneous reading of several signals from one table file.
	SIMTIME	The SIMTIME-block delivers the current simulation time, the simulation step size or the total simulation time (simulation length).













	CLOCK	This block represents a real time clock with alarm function and can be used e. g. for time controlled processes.
	RANGEN	This block type allows the generation of uniformly or normally distributed random numbers.
	SOURCE	In combination with the output block type <i>Signal drain</i> this block enables you to realize "wireless" connections between blocks. The signal drain "sends" its input signal which can then be received at any position of the system structure by a signal source of the same name.

### Block group *Dynamics*








To this group belong all linear and nonlinear dynamic systems from the simple P-element up to the transfer functions which can be parameterized in any way and the differential equation systems.






Icon	Block type	Short description
	P	Proportional element (P-element) with adjustable proportional coefficient
	PT1	Delay element of first order (P-T <sub>1</sub> -element)
	PT1T2	Non-oscillating delay element of second order (P-T <sub>2</sub> -element)
	I	Limited integrator (I-element); the limitation can be deactivated if necessary
	PTN	Delay element of <i>n</i> -th order (P-T <sub>n</sub> -element) with <i>n</i> equal time constants
	PT2	Oscillating delay element of second order (P-T <sub>2</sub> -element) with adjustable characteristic angular frequency and damping
	D	Ideal differentiator (D-element)
	DT1	Rational differentiator (delayed differentiator); this block represents the serial connection of an ideal differentiator and a P-T <sub>1</sub> -element
	DEADTIME	Dead time element (T <sub>t</sub> -element) with adjustable dead time; this block type can be used to simulate runtime effects
	VARDELAY	This block type represents an extension of the dead time-element; the dead time can be set via the block input so it can be used for modelling systems with a variable (i. e. state dependent) dead time (e. g. conveyor belts with variable speed).
	PID	PID-controller (see <i>block group Controllers</i> )
	RESI	Limited integrator which can be reset to its initial value via a reset-input; e. g. oscillators can be modelled with this block type by an output feedback to the reset-input.
	RESI2	This block type represents an extension of the block type RESI; the initial value of the integrator can be set via a second block input.

	LEADLAG	Lead-Lag-element with adjustable numerator- and denominator-time constant
	ALLPASS_1	Non oscillating Allpass-element
	ALLPASS_2	Oscillating Allpass-element
	ADAPID	Adaptive PID-Controller (see <i>block group Controllers</i> )
	TRANSFCT	User-defined $s$ -Transfer function for modelling time continuous linear systems
	DEQSYS	User-defined differential equation system (linear or nonlinear)
	SSM	User-defined multi-input multi-output state space model
	BLOCKLIST	User-defined blocklist, i. e. serial connection of linear standard transfer elements (P-T <sub>1</sub> , P-T <sub>2</sub> , ...). The list can be protected via password; so that the internal structure of the list can only be displayed and modified after entering the valid password.
	UNITDELAY	This block represents a sample and hold element with a delay of a sampling period $T$ . Thus it can be used for modelling time discrete systems (sample systems) or digital filters or controllers. Besides it enables the dissolving of algebraic loops.
	ZTRANSFCT	User-defined $z$ -Transfer function for modelling time discrete linear systems

### Block group *Statics*










Basically these are the nonlinear characteristic curve elements resp. 3D-characteristics (e. g. limiter, dead zone).





Icon	Block type	Short description
	LINSCALE	Linear characteristic curve with adjustable gradient and offset; it can be used for scaling variables
	2POINT	Two-point characteristic curve (see <i>block group Controllers</i> )
	2PHYST	Two-point characteristic curve with hysteresis (see <i>block group Controllers</i> )
	3POINT	Three-point characteristic curve (see <i>block group Controllers</i> )
	3PHYST	Three-point characteristic curve with hysteresis (see <i>block group Controllers</i> )
	LIMITER	This block type limits the output variable to a certain range. The signal is cut at the upper resp. lower limit.
	DEADZONE	With this block type a dead zone for input variables can be realized. Within this zone the block operates with a reduced gain (or even a gain = 0).

	<b>PRELOAD</b>	This block type realizes a preload characteristic curve, i. e. a piecewise linear characteristic curve with an offset at the position 0.
	<b>HYST</b>	Hysteresis effects which occur e. g. during magnetization processes or in gears can be simulated with this block type.
	<b>CHARCURVE</b>	Allows the definition of user-defined characteristic curves based on a specified number of base points. Thus any complex characteristic curve can sufficiently accurate be realized.
	<b>QUANTIZER</b>	This block type realizes a quantizer which can work with a fixed resolution or like an $n$ -bit A/D converter. The quantization can be executed by truncating the decimal places or by rounding.
	<b>CHARMAP</b>	This block type allows the specification of characteristic maps by a base point matrix. By default a linear interpolation between the base points is executed.

### Block group *Controllers*



This block group includes various linear and nonlinear controller types as well as continuous and switching controller types.

<b>Icon</b>	<b>Block type</b>	<b>Short description</b>
	<b>PID</b>	PID-controller with adjustable parameters and anti-windup-mode in limited range; the parameters of the controller can independently from each other be activated or deactivated so that P-, I-, PD- and PI-controllers can be realized.
	<b>ADAPID</b>	This controller type corresponds to the standard-PID block but the parameters can be set via additional block inputs so that adaptive control loop structures can be realized.
	<b>INDUPID</b>	This controller type also corresponds to the standard-PID block but it has its own user interface so that controller parameters can be adjusted via corresponding sliders. Besides reference value, manipulated variable and control variable are visualized by corresponding digital displays.
	<b>2POINT</b>	Two-point controller
	<b>2PHYST</b>	Two-point controller with hysteresis
	<b>3POINT</b>	Three-point controller without hysteresis with adjustable switching points
	<b>3PHYST</b>	Three-point controller with hysteresis with adjustable switching points
	<b>ZP-PD</b>	This block type represents a two-point controller with P-T <sub>1</sub> -feedback. By appropriate parameterization it works similar to a PD-controller (quasi-continuous behaviour).
	<b>ZP-PID</b>	This block type is similar to the type ZP-PD but the feedback of the manipulated variable is executed delayed- via a serial connection of a D-T <sub>1</sub> -element and a P-T <sub>1</sub> -element. By appropriate parameterization it works similar to a PID-controller.

	DP-I	This controller type represents a so-called <i>free-run-</i> or <i>limiting value controller</i> . This controller type consists of a symmetric three-point-element with hysteresis and a final limited integrator. Thus the controller can generate a continuous behaviour of the manipulated variable.
	DP-PI	The so-called three-point step controller consists of a symmetric three-point element with hysteresis with a P-T <sub>1</sub> feedback of its output variable and a final optional limited integrator. By appropriate parameterization it has PI-similar behaviour and is often used to control positioning actuators.
	FC	By this block type fuzzy controllers can be integrated into the simulation structure which was designed with WinFACT-Fuzzy-Shell FLOP.
	FCONLINE	This block type corresponds to the block type FC. While the simulation is running the fuzzy shell is automatically started so that the inner workings of the fuzzy controller can be inspected.






### Block group *Actuators*









This class contains blocks which imitate the behaviour of real industrial final controlling elements.

Icon	Block type	Short description
	ACTUATOR_1	This block represents a final controlling element with limited actuating speed. The actuator reacts undelayed if the input variable changes slowly, if it changes too fast the actuating speed will be limited. Besides the block as every other real actuator has a limitation.
	ACTUATOR_2	This type of a final controlling element has a constant actuating speed, i. e. the output variable always changes continuously. This corresponds e. g. to a positioning actuator which can only run forward or reverse with constant speed.

### Block group *Functions*

Block types which are no conventional transfer systems are called function blocks. Especially these are the adder and other junctions of several inputs variables.






Icon	Block type	Short description
	JUNCTION	This block allows the connection of up to 50 input variables by the operations addition, multiplication or division. It is needed in almost every simulation structure.
	FCT1	With this block functions of a variable can be realized. Besides numerous predefined functions a function can be specified by the user via a function parser.
	FCT2	This block type corresponds to the block type FCT1 but functions of two variables can be used.
	FCTN	This block type corresponds to the block type FCT1 but functions of up to 50 variables can be used.
	EXTREMVAL	With this block the minimum resp. maximum of a signal resp. of its absolute value can be determined.














	<b>FREQCNT</b>	This block allows the determination of frequency, angular frequency or periodic time of a periodic signal based on a sequence of zero-crossings (related to a reference value).
	<b>MAXMIN</b>	This block allows the comparison of signals. It either switches the minimum or the maximum of all input signals to the output.
	<b>ANASWITCH</b>	This function block realizes an analog switch (relay). The input variable is switched to the output if High-level resp. Low-level (selectable) occurs at the control input S. Otherwise the output variable will be set to 0.
	<b>ANASWITCH2</b>	This function block realizes an analog change-over switch. Depending on the signal of the control input it switches between the two input signals.
	<b>MULTISWITCH</b>	This function block is also used for switching between signals, but can process up to 49 signals. The signal value at the control input determines the data input which is switched to the output.
	<b>AH</b>	This function block realizes a sample and hold element of 0-th order. The input variable is sampled at a specified sampling time $T_s$ , switched to the output and kept constant up to the next sampling time.
	<b>EXTERNSH</b>	This function block also realizes a sample and hold element. The sampling is triggered by a block input.
	<b>STATISTIC</b>	This block allows the calculation of statistic parameters like mean value, standard deviation, effective value etc.

### Block group *Digital*

This class contains those blocks the output variables of which can only take the binary states *HIGH* (logical 1) resp. *LOW* (logical 0). The corresponding levels can be specified by *Blocks / Digital / Level....*

The value for the threshold defines the level from which the digital input value can be seen as logical 1. By default the value of LOW is 0, the value of HIGH is 5 and the value of the switching value is 2.5 (corresponding to TTL).

Icon	Block type	Short description
	<b>LOGIC1</b>	Logical gate with one input which can operate as inverter or with output always set to HIGH- resp. LOW-level.
	<b>LOGIC2</b>	Logical gate with two inputs and the operating modes AND, OR, NAND, NOR, equivalence and antivalence
	<b>LOGICN</b>	Logical gate with up to 50 inputs and the operating modes AND, NAND, OR and NOR
	<b>RSFLIPFLOP</b>	This block realizes a RS-flip-flop with the two operating modes static or edge triggered and the selectable behaviour set dominant or reset dominant.
	<b>DFLIPFLOP</b>	This block represents a D-flip-flop and can therefore be used as a 1-bit storage unit or for the realization of shift registers or frequency dividers. The input value at the data input is saved and switched to the

	JKFLIPFLOP	output with a positive edge at the clock input C.
	MONOFLOP	This block represents a monostable flip-flop (univibrator). With a positive edge at the input it produces a pulse of a user-definable length. An incoming pulse at the input while the output has HIGH-level is ignored.
	ONDELAY	This block type delays a switch-on event (positive edge at the block input) by a user-defineable time. The switch-off-event is executed undelayed.
	OFFDELAY	This block type delays a switch-off event (negative edge at the block input) by a user-defineable time. The switch-on-event is executed undelayed.
	ONOFFDELAY	This block type represents a combination of switch-on/off delay. It delays the switch-on event as well as the switch-off delay by times which can separately be defined by the user.
	COUNTER	Positive edge-triggered counter with user-definable counting direction (forward/reverse) and reset input.
	INCR	This system block realizes an incremental encoder, i. e. an encoder for measuring position or angle changes. If the block input changes for a value of $\Delta x$ , the block generates a pulse of a definable width at the output.
	PWM	This block type generates a pulse width modulated output signal dependent on an analog input signal. For that a HIGH pulse of constant frequency but with variable pulse width is generated at the block output. The pulse width is proportional to the amplitude of the input signal.
	DISCRIMINATOR	This block realizes a window comparator (window discriminator). It delivers HIGH-level at the output if the input variable for a user-definable minimum time lies in/out of a special range of values. Therefore the block can be used to detect violations of the range.
	ZEROCROSS	This block can be used to detect zero-crossings of signals. At every change of sign of the input variable it produces a pulse at the output. The length of the output pulse corresponds to the simulation step size.
	LOGCHANGE	Logical edge detector; you can choose the detection of positive edges (change from LOW- to HIGH-level) and/or the detection of negative edges (change from HIGH- to LOW-level).
	COMPARATOR	This block compares the two analog input variables and delivers HIGH-level at the output if the condition of the comparison is fulfilled. There are several operators for the comparison to choose from.
	DECODER	This block converts an analog value at the block input to a digital

value in binary code at the block output. The number of binary pre- and post-decimal digits and thus the discretization can be specified as well as the validity of negative input values.



ENCODER

This block represents the counterpart to the digital decoder. It converts a binary coded digital value at the block inputs to an analog output value.



ADC

This block converts an analog input value to a binary coded digital output value. The resolution of the converter can be specified.



DAC

This block converts a binary coded digital input value to an analog output value. The resolution of the converter can be specified.

### Block group *Action*

These are blocks which allow the user an interactive intervention (e. g. switches). There are two different kinds of block types: one kind allows the intervention via a separate window which can be moved on the screen, the other via a corresponding control element which is part of the block symbol itself integrated into the system structure. Labels of the latter begin with BLOCK... (e. g. BLOCKBUTTON).







Icon	Block type	Short description
	BUTTON	Push button/switch
	POTI	Slide potentiometer
	SPINEDIT	Spin edit field
	ROTKNOB	Rotation knob
	INDUPID	Industrial PID-controller (see <i>block group Controllers</i> )
	JOYSTICK	This block allows the use of up to two joysticks at the PC-gameport.
	BLOCKBUTTON	Block push button/switch
	BLOCKPOTI	Block slide potentiometer
	BLOCKSPINEDIT	Block spin edit field

### Block group *Communication*

This group includes blocks for the communication with DDE or the TCP/IP-protocol.






Icon	Block type	Short description
	DDEIN	This block allows the reading of values from other Windows applications (e. g. EXCEL or LabView) by <i>Dynamic Data Exchange</i> (DDE).



	DDEOUT	This block allows the output of values to other Windows applications (e. g. EXCEL or LabView) by <i>Dynamic Data Exchange</i> (DDE).
	TCPINCLIENT	This input block realizes a client which receives data corresponding to the TCP/IP-protocol. The current state of the connection is shown by a separate status window.
	TCPINSERVER	This input block realizes a server which receives data corresponding to the TCP/IP-protocol. The current state of the connection is shown by a separate status window.
	TCPOUTCLIENT	This output block realizes a client which sends data corresponding to the TCP/IP-protocol. The current state of the connection is shown by a separate status window.
	TCPOUTSERVER	This output block realizes a server which sends data corresponding to the TCP/IP-protocol. The current state of the connection is shown by a separate status window.
	EMAIL	This block type can be used for sending an e-mail if a certain event occurs. The block sends the e-mail (if desired with attachment) via an active online connection if it detects a positive edge at the block input. The status of the connection and the sending of the e-mail are displayed in a separate control window.

### Block group *Simulation*



These are all blocks concerning the simulation control itself.





Icon	Block type	Short description
	SIMCANCEL	This block type can be used for a program controlled termination of the simulation resp. switching to the single step mode to continue the simulation run step-by-step.
	SIMDELAY	This block delays the simulation at each simulation step for a user-definable time in msec. It can be used to force a delay of the simulation on very fast computers without switching to the real time mode.
	SIMRESTART	This block can be used to terminate the current simulation run and start a new run if a positive edge at the block input occurs. If desired the new simulation run can be executed based on another system structure than the currently loaded in standard or endless mode.
	STATE2FILE	This block type allows the storage of the current system state in a BSF file. Please refer to the WinFACT program documentation resp. online help for detailed information.
	FILE2STATE	This block type allows the loading of system states from a BSF file and represents the counterpart to the STATE2FILE block. Please refer to the



WinFACT program documentation resp. online help for detailed information.

### Block group *Drains*

This type class includes all blocks which only have one resp. several inputs. Basically these are block types for the visualization or processing of simulation results (e. g. oscilloscope, file output). Besides the system block in the paint window additionally the blocks have a *display window* which shows the simulation results. If you insert such a block, this display window is presented symbolized by a type-specific icon. For the simulation all display windows can be set in normal size using the menu option *View / Show all output windows* or the button . They can be minimized at any time by the menu option *View / Hide all output windows* or the button . In each case the display shows the same caption as the system block itself. The system blocks can be parameterized by a double click at the system block or inside the display. For some of the output blocks the display windows are resizable (e. g. oscilloscope). Besides the blocks with their own display windows there are also blocks with displays integrated into the block symbol itself.

Icon	Block type	Short description
	YTPLOT	This output block allows the simultaneous graphical presentation of the time response of up to three variables. The curves can be drawn in a common or in separate diagrams (with or without grid). The axes of coordinates can be scaled manually or automatically. The amplitude values can be represented linear or logarithmical. With a comfortable measurement function characteristic values can be determined from the diagrams. Because the time responses of the preceded simulation can be "frozen" even tendencies of parameter variations – e. g. of control parameters – can be presented. If necessary the measurement function can be applied to the "frozen" curves. Especially for control engineering applications (e. g. the determination of settling values) a tolerance band with a user-definable width can be inserted into the diagrams. Furthermore PID-controllers can be designed directly from the time response display window according to the so-called rules of thumb.
	MULTILOT2	This block allows the simultaneous display of up to 50 signals within a common or separate coordinate systems.
	MULTILOT	With this block results of entire simulation series can be presented. It allows the recording of curves over any number of simulations and therefore is especially interesting for examinations concerning parameter variations (e. g. time constants) and batch runs. All recorded curves are listed at the right window border. By a right mouse click on a list item a context menu with further options can be opened. In the batch mode the curves are automatically labelled dependent on the current block parameter values that are modified.
	XYPLOT	The trajectory display allows the presentation of two input variables $x$ and $y$ in the $x$ - $y$ -domain with the time $t$ as the curve parameter. A storage function is available for the previous simulation (as you get

it for the time response block).



### 3D PLOT

This block allows the representation of three input variables in the  $x$ - $y$ - $z$ -domain with the time  $t$  as curve parameter. The diagram can be scaled automatically or manually and rotated in two directions.



### FFT

This block allows the calculation of the amplitude spectrum of the input signal by the Fast-Fourier-Transformation (FFT). The time window and the number of base points for the transformation can be chosen by the user. The block can have up to three inputs. For the representation of periodic signals the block can be used in a discrete mode; in this case only the maxima of the determined spectrum are displayed in line form.



### BODE PLOT

This block allows the calculation of the gain (in dB) and phase (in degree) of two sinusoidal signals connected to the block inputs. Both signals must be of the same frequency. In combination with the batch mode this block can be used for the automatical calculation and representation of frequency responses (Bode resp. Nyquist plots).



### RECORDER

This block emulates a  $y$ - $t$ -recorder and is therefore especially suitable for long lasting recordings. The block can handle up to three input channels. Time base, sensitivity and offset of the single channels can be chosen separately.



### HISTO

This block allows the calculation and graphical representation of frequency distributions (histograms).



### TABLE

This block creates a table output of up to 50 input signals. The output interval can be specified as well as the output format for the time column (real time and/or simulation time).



### FILE OUTPUT

The FILE OUTPUT-block is the counterpart to the FILE INPUT-block. It allows the output of a signal in form of pairs of values to a file of type SIM.



### TAB FILE OUTPUT

This block allows the storage of up to 49 time responses in the form of a table. The saved data can be processed directly with spreadsheet applications (e. g. EXCEL) in an easy way.













### SCOPE

This block emulates a four channel oscilloscope on the screen which can be resized in any way. Time base and sensitivity resp. offset of the channels are separately adjustable. All channels can be displayed multicoloured.









### LOGAN













This block can be used for the representation of digital signals. It realizes a logic analyzer with up to 49 data inputs and one reset input. The logic analyzer offers a comfortable measurement mode.

	ANALOGMETER	Analog instrument – optionally with a digital display
	DIGITALMETER	Digital display
	BARGRAPH	Bar graph
	VLEDBAR	Vertical LED bar
	HLEDBAR	Horizontal LED bar
	STATUSDISP	Status display for binary signals
	BLOCKDIGOUT	In-block-digital display
	BLOCKBARGRAPH	In-block-bar graph
	BLOCKSTATUS	In-block-status display
	DRAIN	With this block and the input block type <i>Signal source</i> you can realize "wireless" connections between blocks. The signal drain "sends" out its input signal under a special name (the name of the block). This signal can then be received from a signal source of the same name at any – even several – positions in the system structure.

### Block group *Miscellaneous*

This class includes all blocks which cannot be classified in one of the groups mentioned before.





Icon	Block type	Short description
	PARMOD	This block type allows the modification of any (floating point) export parameter within the simulation. So well-aimed modifications of block parameters (e. g. gains or time constants) can be realized. If the enable input S has HIGH level (or if it is open), the specified export parameter value is set to the value of the data input D. If input S is connected but has LOW level, no modification is executed.
	PARVAL	This block type delivers at its output the current value of any (floating point) export parameter.
	FC	Fuzzy Controller (see <i>block group Controllers</i> )
	FCONLINE	Online-Fuzzy-Controller (see <i>block group Controllers</i> )
	NEUROMODEL	This block type allows the integration of neural nets designed with the program <i>NeuroModel</i> (  <a href="http://www.neuromodel.de">www.neuromodel.de</a> ).

	FTRUN	This block type allows the integration of fuzzy systems designed with the program <i>fuzzyTECH</i> (  <a href="http://www.fuzzytech.de">www.fuzzytech.de</a> ).
	ALASKA	This block type allows the integration of a model designed with the simulation software <i>ALASKA 4</i> .
	LABEL	Blocks of the <i>Label</i> type have no technical function but allow to give a name to signals with which they appear for example in time response blocks or – very important – in superblocks.
	MUX	This block allows the combination of several connections to a single one. Its use is therefore suitable if several connections are to be drawn over a great area at the same time. The output connection of a multiplexer has to be connected to the input of a demultiplexer block.
	DEMUX	This block splits up the multi-connection produced by a multiplexer block. The input connection of a demultiplexer therefore has to start at the output of a multiplexer.
	VIEWDOC	This block can be used to display the content of a document file (e. g. a PDF file) by a mouse click on the button located within the system block. The document is displayed by the application specified for the corresponding file extension in the Windows registry.
	MESSAGE	This block allows the generation of a message resp. a warning if a positive or negative edge appears at its input. Additionally to the graphical display each message can optionally be identified by a WAV-file.
	AUDIOIN	This block type allows the input of audio signals via the microphone input of the sound card. The signal can be visualized within a separate window.
	AUDIOOUT	This block type allows the output of audio signals via the sound card. The signal can be visualized within a separate window.
	HARDWARE	This block is used as an interface to various hardware (e. g. PC-cards, external hardware at the RS-232-Port etc.) and can contain hardware inputs as well as hardware outputs. Further information you get from the documentation delivered with the hardware driver.
	CCODE	This block type is only available if the BORIS AutoCode Generator is installed. The block is used as an interface to various hardware specific I/O-routines. Detailed information you find in the documentation of the AutoCode Generator.

### Block group *VirtInstr*

This block group includes some additional virtual instruments which are realized as User-DLLs. You find the corresponding files in the subdirectory *VirtInstr* of you WinFACT installation (image 23).

Icon	File name	Short description
	MULTISTATUS.DLL	Multi status display

-  MULTIOUTPUT.DLL Multi digital output
-  MULTIINPUT.DLL Multi digital input
-  MULTIBUTTON.DLL Multi switch/button
-  FUNCGEN.DLL General-purpose function generator

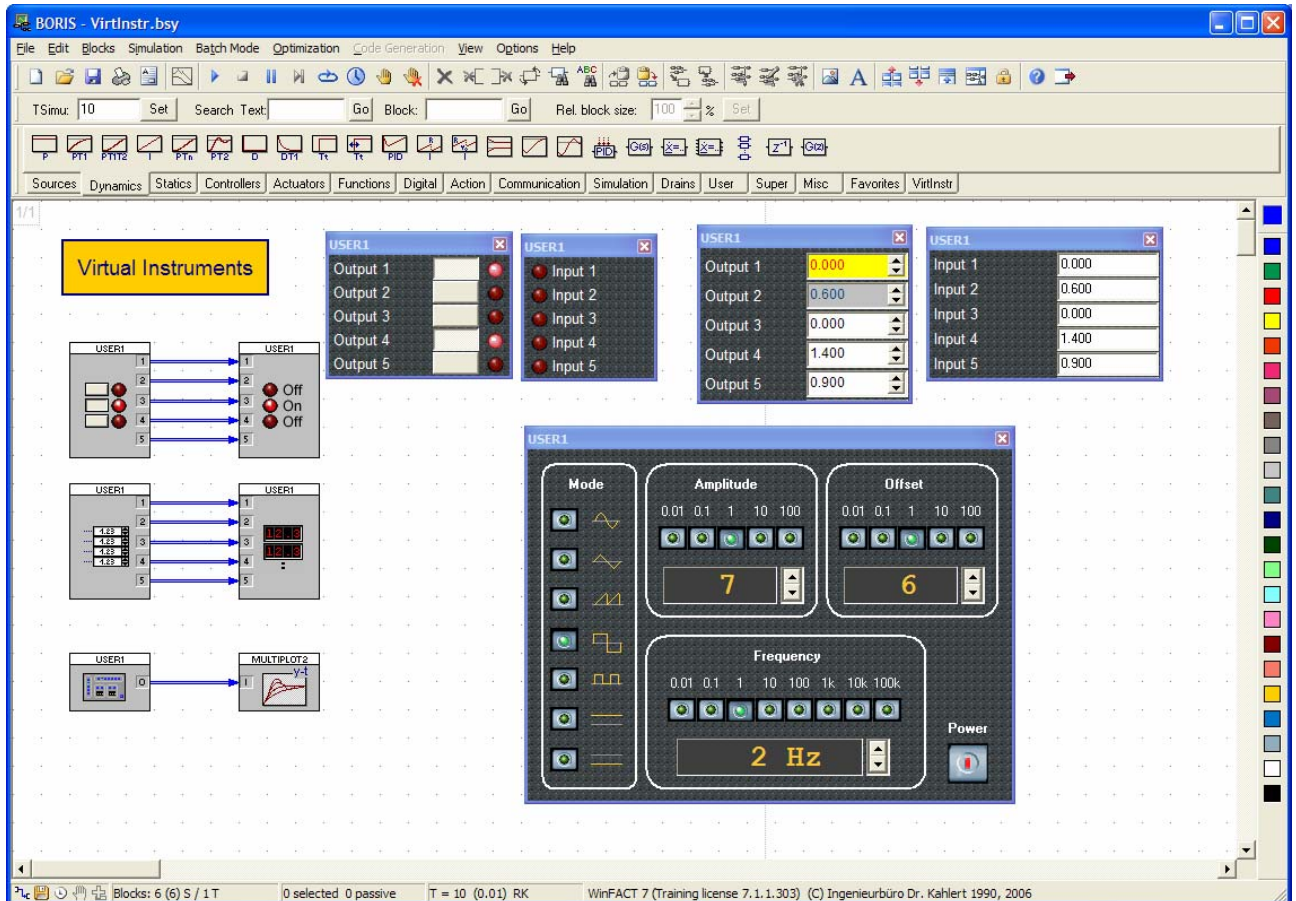


Image 23. Virtual Instruments

## Using superblocks

### What is a superblock?

A superblock is a special system block type that results from the grouping of several system blocks and their connections. Thus a superblock is nothing else than a partial structure – a "black box" – being combined to a new block with in- and outputs. Therefore superblocks are especially recommendable for a clearly arranged structuring of complex systems and for the grouping of frequently used partial structures.

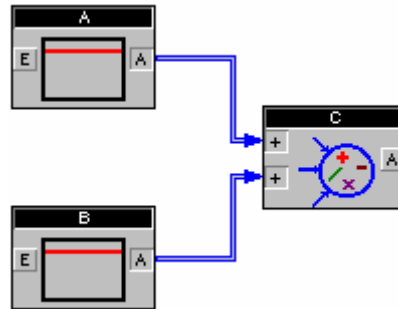
Superblocks are *file referenced*. All informations about the blocks and connections within the superblock are saved in files with the extension SBL. These files are – except an additional file header – identical with "normal" BORIS system files (BSY-files). Therefore superblock files can be saved as normal system files and vice versa, they also can be simulated.



## Inputs and outputs of superblocks

The inputs and outputs of superblocks are automatically defined by BORIS basing upon the following rules:

- All open inputs and outputs of the selected system structure become inputs resp. outputs of the superblock. Example: The following structure has to be grouped to a superblock.



The resulting superblock has two inputs (the inputs of block A resp. B) and one output (the output of block C).

If block outputs that already contain a connection (e. g. a feedback) shall become superblock outputs, the use of label blocks is necessary. This special block type is available by the *Blocks / Miscellaneous / Label* command. By inserting a label and connecting its input to the other blocks output you easily create a new open output that in the following becomes a superblock output. If on the other hand an open input must not become a superblock input, connect it to a constant block with the value 0 (or another proper value).

- The inputs and outputs of the superblock are numbered in the sequence the corresponding blocks were included. If in- or outputs are to be changed, first delete the corresponding blocks and then insert them again in the correct order or – more comfortable – insert label blocks.

## How to create a superblock

There are two ways to create a new superblock:

1. First create the superblock structure separately within the BORIS paint window. After the structure is complete, save it as a superblock file by the *File / Save as...* (choose *BORIS superblock files (\*.sbl)*) command. Later you can load the superblock simply by referring to this filename.
2. If a partial structure of an already existing system shall be grouped to a superblock, first select this partial structure and then create the superblock by the *Edit / Group to superblock* command, the <Ctrl><G>-keys or the corresponding button of the control toolbar. In the following BORIS asks for the filename of the superblock and then inserts the created superblock.

A superblock can be ungrouped at any time by the *Edit / Resolve superblock* command, the <Ctrl><U>-key or the corresponding button of the control toolbar. By this procedure all inner blocks and connections of the superblock appear again within the paint window; thus take care that there is enough free space in the superblocks neighbourhood!

## Sample files:

The sample collection contains a lot of different sample files (Extension BSY).

## Graphical presentation of results by INGO

The WinFACT-module INGO allows the graphical presentation of all types of WinFACT-files:

- Simulation results (SIM-files)
- General data pairs (XY-files)
- Multiple data pairs (MXY-files)
- Bode-diagrams (BD-files)
- Nyquist-diagrams (OK-files)
- Function matrices (FWM-files, Presentation optional as 3D graphic or contour lines)

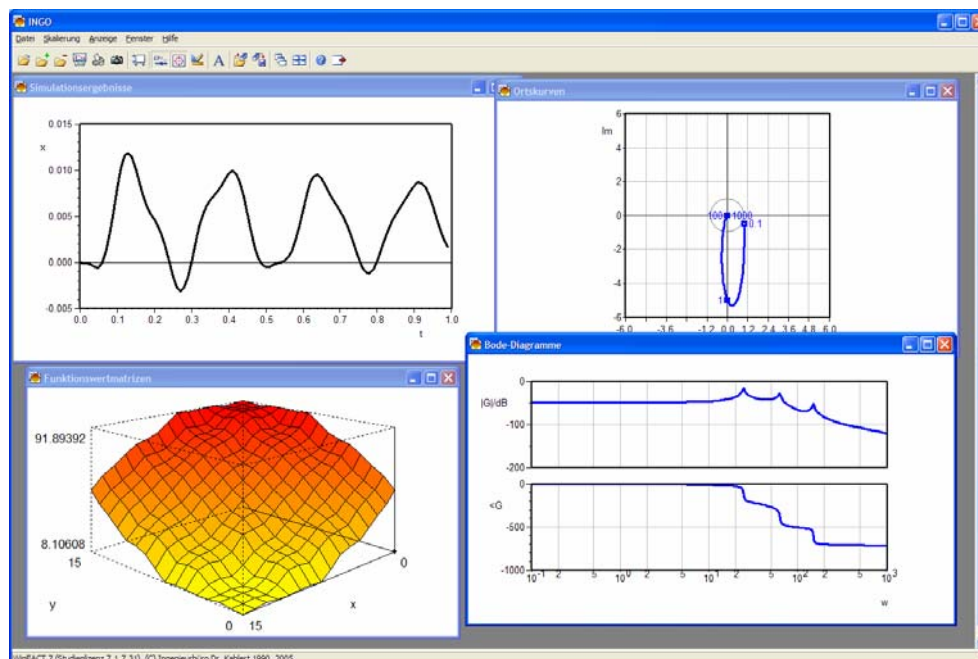


Image 24. User interface of INGO

INGOs user interface is based on the WINDOWS multiple document interface and therefore allows the parallel presentation of several graphic windows (also of different types); each of them can contain several curves. Each curve can be displayed as a line or as marker symbols. Each graphic window can – according to MDI standard – arbitrarily be moved, resized or minimized to symbol.

A new graphic window is opened by the *File / Open...* command or the corresponding button of the toolbar. In the following a file open dialog appears; within this dialog the filename for the first input file has to be defined. The extension of the filename (e. g. SIM) automatically determines the type of the graphic window and thus the curves added later must fit this type.

To add a new curve to the active graphic window, use the *File / Add...* menu option or the corresponding toolbar button. A graphic window of SIM, XY, MXY, BD or OK type can contain any number of curves. FWM-graphics which are presented in contour lines form can display a maximum of 30 contour lines per file. FWM-graphics in 3D-form, however, can display only one file per window.

The scaling of the axes is automatically adjusted in that way, that all curves are displayed completely. A modification of this automatic scaling can be performed by the *Scaling* menu option and the following parameter dialog.

Following the *Display / Options...* command a dialog appears which offers some settings concerning line styles, diagram legend etc.

### **Sample files**

The sample collection contains sample files for all types of WinFACT-graphic files.