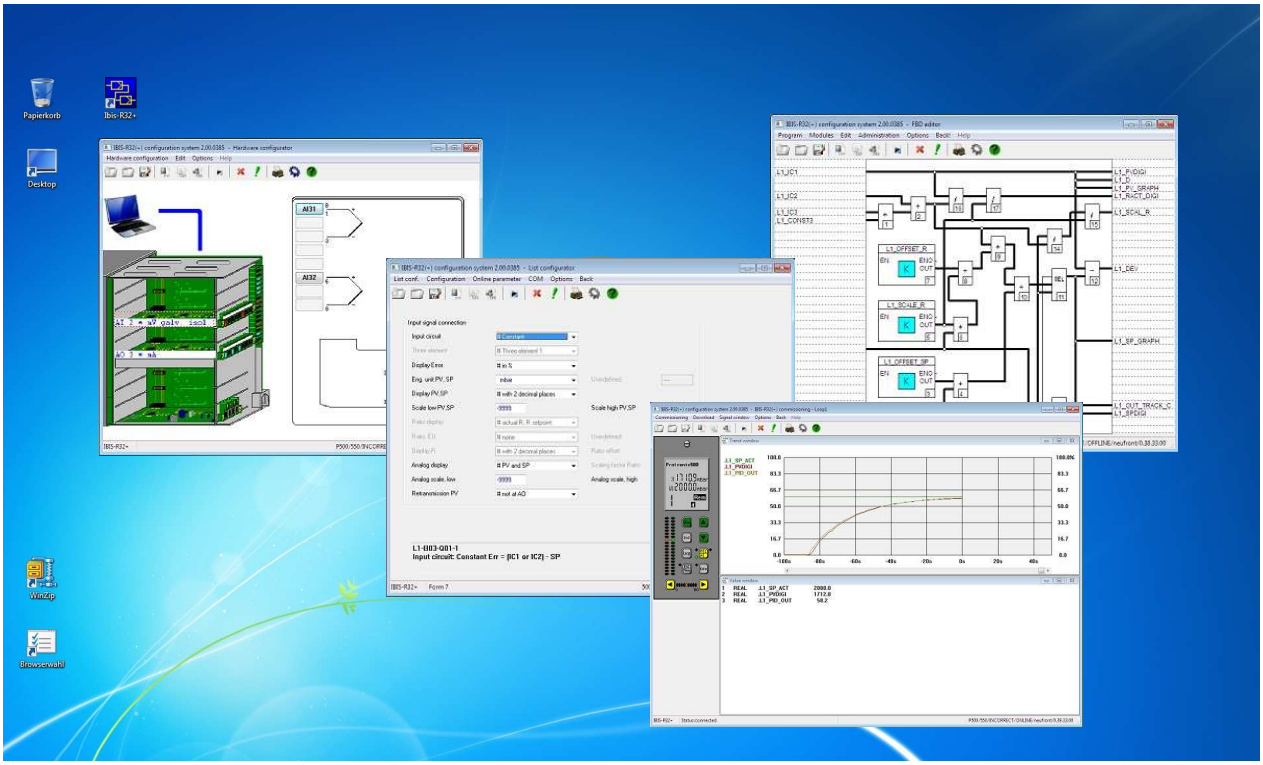
IBIS-R32+

Operating manual

ENA42/62-52030 EN
Rev. 02

Configuration and parameter setting software for D100/D500/D700 and P100/500/550 P700/750

ENA Control



ELECTRONIX
manufacturing & services

Table of contents

Typographical conventions

The formats listed below are used to provide certain items of information:

Menu item Reference to menu item
[Text] Reference to on-screen button
<Key> Reference to key on the keyboard
„Chapter“ Reference to chapter or section in the manual

Operation notes

Operation of IBIS-R32 is based on the Microsoft-Windows standard. Functions provided by Windows are not described separately here. For this reason, you need general knowledge on how to handle Windows. Refer to the Windows user's guide. The formats listed below are used for specific operations:

→Menu item Select the menu item using the mouse (click left mouse button) or using the keyboard commands corresponding to the Windows standard.

→<Key> Press the specified key on the keyboard.

→<Key1+Key2> Press the specified keys simultaneously on the keyboard.

→[Text] Choose (= click) the specified button in the dialog box.

1 Installation

Table of contents

1 Installation

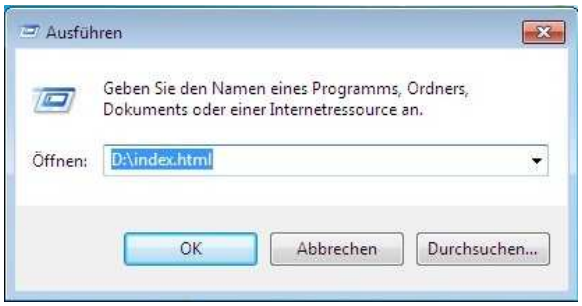
1.1 System requirements

- Processor minimum 80386
- RAM minimum 1 GB.
- Aviailable diskspace minimum 64 MB.
- Monitor and monitor driver minimum VGA standard.
- Windows XP, WIN7 32/64 BIT, WIN8
- A graphics-capable prinzer, supported by a Windows driver, should be used.

Note

IBIS-R32 does not disable mutual file access on networks. For this reason, there are often problems regarding concurrent access to files if this is not managed over the network.

1.2 Install



1. Place program CD in drive D:. If the CD- drive is E:, replace the CD drive name in the work steps below.
2. Load the Windows Program Manager.
3. →File→Run
4. Type D:\Index.html and →[OK]
5. Follow the instructions in the Setup routine to install IBIS-R32
6. Press [Cancel], if you want to interrupt the Setup routine.

The Setup routine generates a new program group called IBIS-R32. This program group stores IBIS-R32 as an application program.

In addition, the IBIS-R32.INI file is stored in the Windows directory containing the basic settings for IBIS-R32. Do not change this file, otherwise the function of IBIS-R32 will be impaired.

1.3 New program versions

Separate installation instructions are provided for installing new program versions of IBIS-R32.
Access to project files in the mass memory is not always guaranteed when a new program version is used.

Files compiled with an older version must be exported from the old and imported into the new one
(refer to “2.3 Exporting from a project” and “2.3 Importing into a project”).

1.4 Starting IBIS-R32

Start the IBIS-R32 program by double-clicking the

IBIS-R icon in the IBIS_R Program Group.

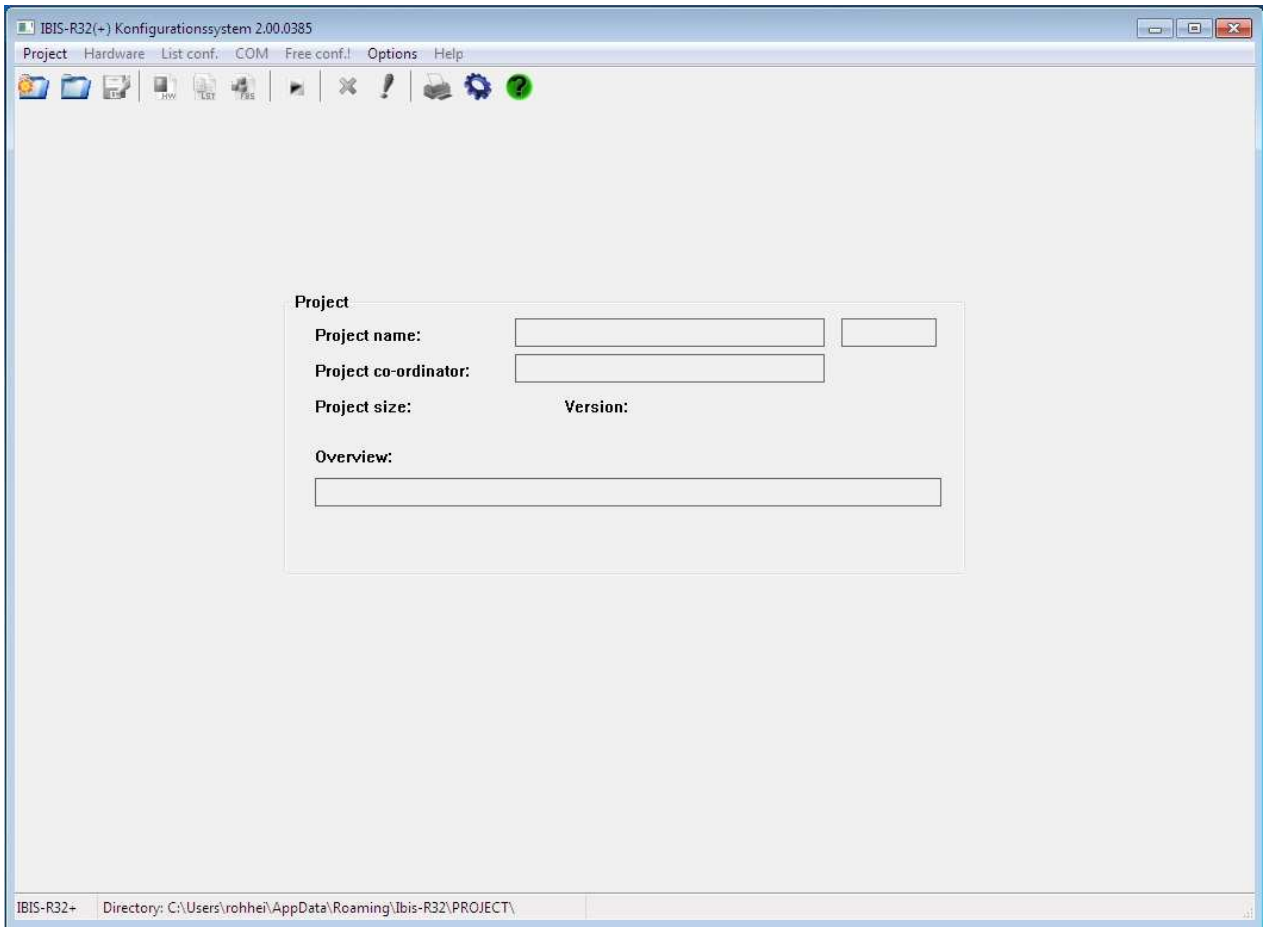
If the requirements listed in the section “1.1 System require

ments” are not fulfilled, an error message is displayed.

If not already available on starting, a file called IBIS_RDB.HBX is created in the Windows directory. Do not change this file.

If memory is not sufficient for IBIS-R32, an error message is displayed. To provide enough memory, all other applications should be closed. To control available memory, use →Programm manager→Help→Info.

1.5 Contents of application window



After starting the program, the application window appears and you can select items in the menu bar which execute various parts of the program and actions.

Additional information is provided in the footer. Its contents are described in detail in the following sections.
Project name with max.32 characters, including 8 characters that are managed by the controller

1.5.1 Project

After program start, you need to select this menu item in order to process a configuration. The menu item is described in detail in the section “2 Project Manager”. Only after processing this menu item program parts for generating or editing a controller configuration can be selected.

1.5.2 Configurators

After you have created a new project or loaded an existing one, you can access the configurators to create a controller configuration. You can access the following configuration routines:

Hardware

The hardware configurator helps you to define the module configuration in the slots for the controller type you selected for the project. You can use this configuration during a session to check its plausibility.

List conf.

The list configurator hepls you to reply to queries and type in values for parameters. This configures the unit, the I/O level and the control loops.

Free conf.!

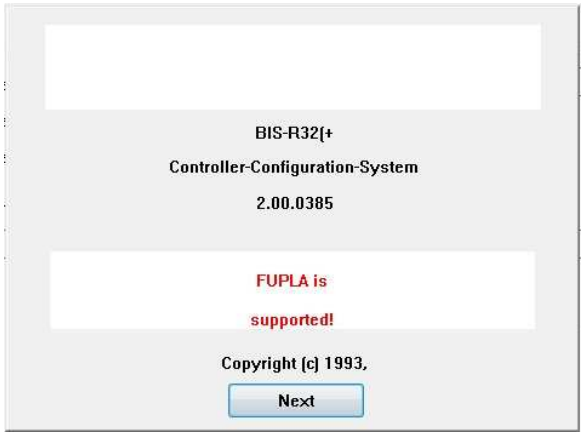
In addition to the list-type configuration for the unit, you can generate a graphic configuration using a funktion-module configurator.

1.5.3 Commissioning/COM

In order to put a configuration into operation, you must load it into the controller. You can use the start-up routine called COM to do this. Start the routine with →COM.

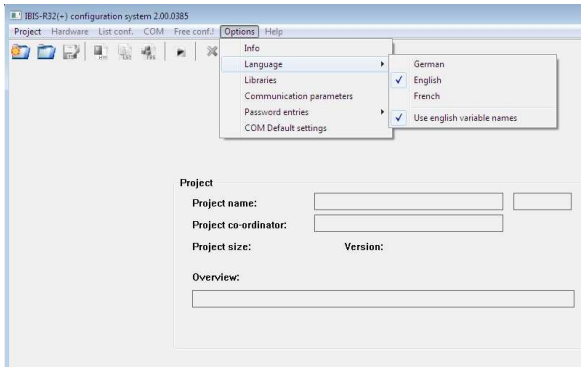
For a detailed description, refer to the section “6 Commisioning”.

1.5.4 Options



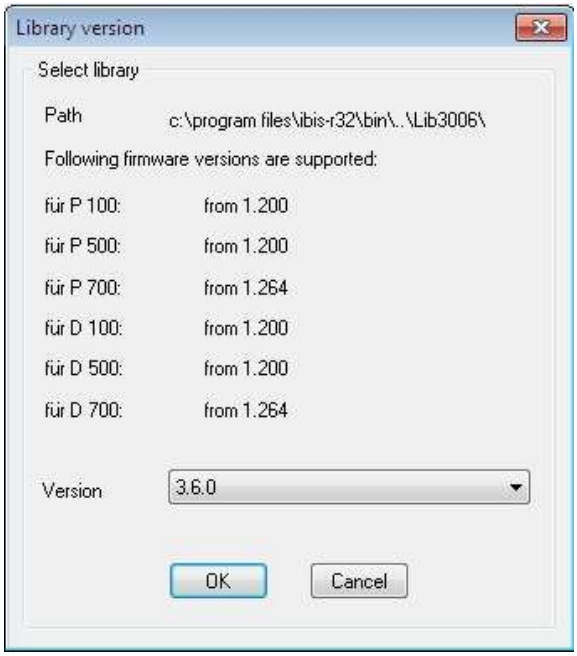
About

provides product information on IBIS-R32: free-style configuration is (not) available.



Language selection

selects the language you want for the user interface. A submenu containing the possible languages appears for your choice. Switchover to a new language takes place immediately. Planned languages which are not yet implemented are displayed dimmed. You should only switch over to a different language if you have a different language version of Windows for this language.



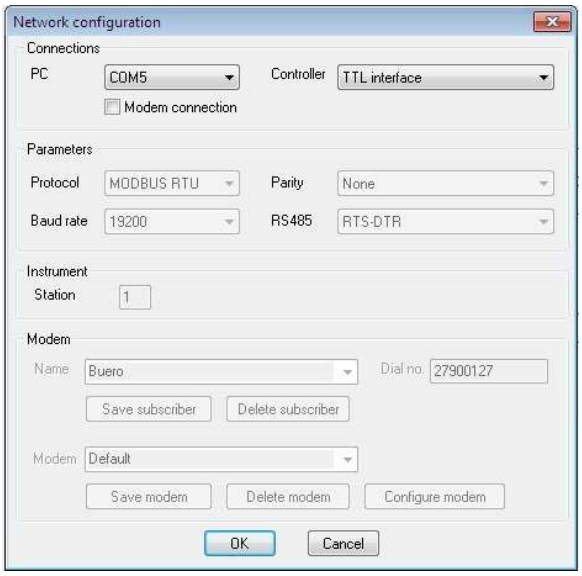
Library management

IBIS-R32 works with the library 3.6.0. A switch to different version of the controllers is not possible.

Libraries in IBIS-R32 contain parts of the hardware and descriptions of the list configuration and functional modules which can be supplied with a controller. Therefore, always forming a part of a firmware version of a controller is a clearly identifiable IBIS-R32 library. One library can however be valid for several firmware versions.

When creating a new configuration (of a new project) with IBIS-R32, it must be predetermined for which firmware version it is being created.

If necessary, the controller via a firmware update be placed on Library 3.6.0
This menu item can also be selected for checking the selected library even when a project has been accessed.



Communication parameters

sets the communication parameter for a serial connection to a unit.

Connections

PC COM1 to COM16
Defines the port conncting zhe computer to the controller

Controller
Front or rear panel
Informs the computer whether the controller is connected via the integrated configuratuin port (with the front panel dismounted) or via the retrofitted RS 485 port module on the rear panel.

Parameters

The parameters are only required for connecting the controller via the rear-panel RS 485 port. If the IBIS-R32 uses the front-panel port, you cannot change these parameters. The values displayed then have no meaning.

Protocol MODBUS RTU equipment bus Enter the protocol configuration for the controller.

Parity even -odd Enter the parity configuration for the controller.

Baud-Rate 600, 1200, ..., 38400
Enter the baud rate configuration for the controller.

RS485 invers RTS-DTR, none, RTS-DTR The controller of the changeover signal for one of the RS-485 converters postconnected to the RS-232 port if IBIS-R32 is connected to the controller by means of the retrofittable RS-485 port module.

Unit

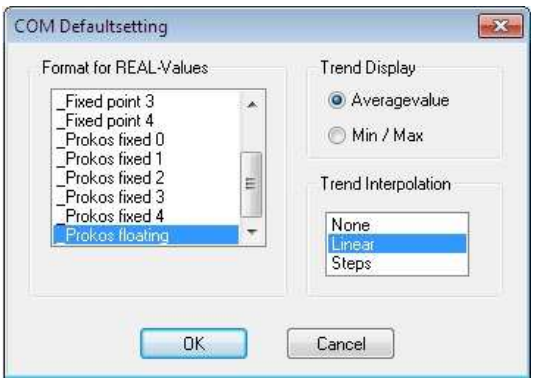
Station 0 -127

If the computer is configured as the master in an RS 485 network, you can use the station number to access the corresponding controller by which the station number is entered.

The settings you have made are stored at the end of your session with IBIS-R32. It is not necessary to reset the environment you want every time.

Password inputs

If the controller connected to the port is protected against unauthorized user access by means of a password, you can only access the controller characteristics by entering a password (list password).

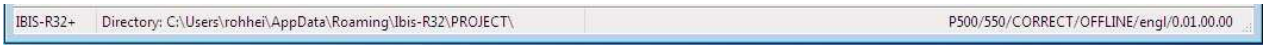


COM Basic settings

ask for the format of the figures in the value window for set point and actual value of the controller for the very first begin of a new project in the commisioning phase, as well as the trend interpolation between two displayed values and the type of display for bigger time intervals. Also refer to “6.8.4 Options”.

These settings are valid for all other new projects till these are remodified.

1.5.5 Footer



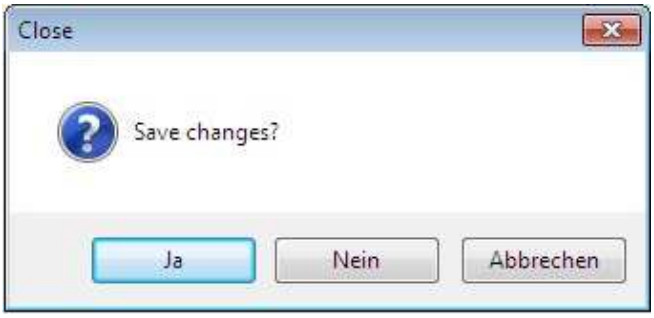
Displays on the left the current directory setting for storing a project located in the memory and its name.

On the right are the digits for the equipment variants, the status of the communication link ONLINE or OFFLINE, the plausibility status CORRECT or INCORRECT, the project name and the project version.

1.5.6 Help

Not supported at present.

1.6 Ending IBIS-R32



You can end your session with IBIS-R32by →Project . →Quit. At this time, you can still reject all the changes you have made since you opened the project or store the changes you have made.

→[Yes] saves all changes.

→[No] rejects all changes and the project status at the time of loading is retained.

→[Cancel] cancels the request to end IBIS-R32 and the current project status is retained and the project is not removed from the desktop.

If the changes are saved, there is a further query as to whether the project should also be exported:

→[Yes] leads to the input mask for exporting. Also refer to “2.5 Exporting a project”.

→[No] ends IBIS-R32, without exporting the project.

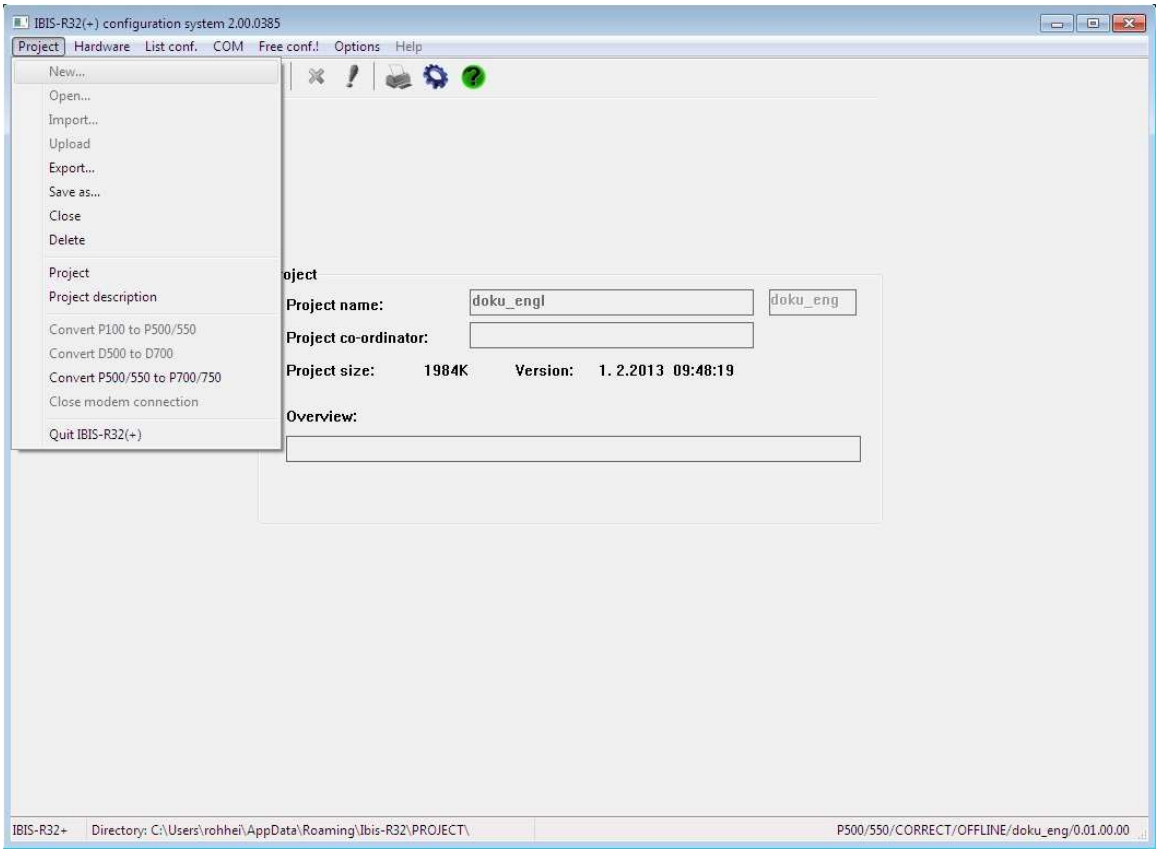
1.7 Error messages

If error messages appear during your session with IBIS-R32, you should save the current project if possible, end WINDOWS and restart.

2 Project manager

Table of contents

3 Project manager

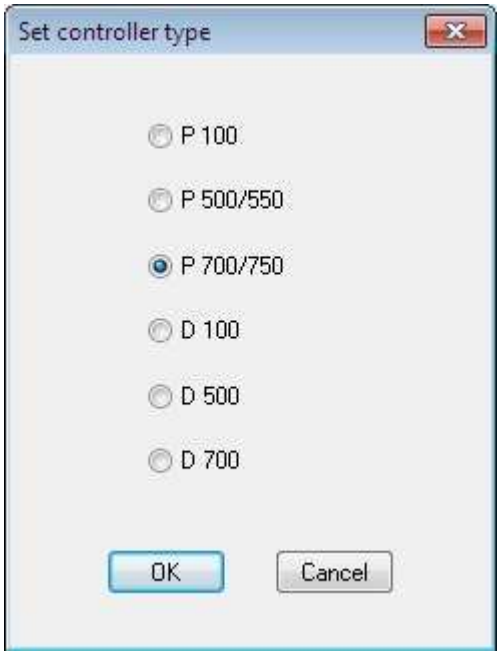


The function of the Project Manager is to manage configurations for the Protrenic controllers on the mass memory. The configurations for the various controller types in IBIS-R32 are called projects.

You can select a number of Project Manager services from the Projectmenu item in the applications window.

In the submenu, you can only select services which are practical in the current situation, otherwise they are displayed dimmed.

2.1 Create a new project

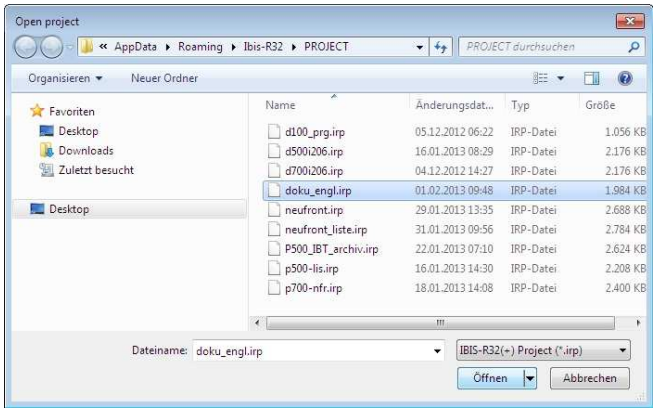


by →Project→New...

For every project IBIS-R32 creates 3 files with the extensions .prx, .hdr und .cmm. All configuration changes during work with IBIS-R32 are saved in the .PRX file.

When working with IBIS-R32, an additional file with the file name extension .bak is created. This file is created from the .PRX file when accessing an existing project with →Project
→Open... This way one can always reaccess the current project status for →Project→Open... despite interim savings.

When creating a new project, you must choose the type of controller:

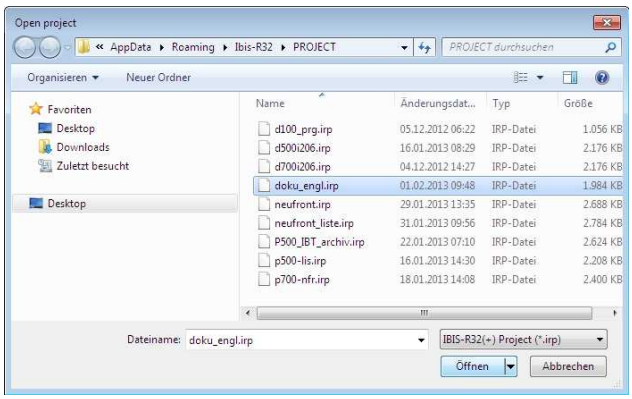


In this dialog box, you can select the drive and the directory in which the files are stored and define the project name. The filename extension .IRP is automatically specified. To convey configurations between various computers, also refer to “Exporting a project”.

You can type in a short comment relating to the project as well as enter a name for the person responsible for the project. The information in the project header can also be edited later using the Projectheader function from the Projectmenu (refer to “2.9 Changig the project header”). Also refer to the Projectcomment function from the Projectmenu for making additional comments on a project (“2.10 Creating and editing a project comment”).

You can now access the configurators for the hardware, lists and the free-style configuration from the menu bar.

2.2 Loading (opening) a project



with →Project→Openproject.

You can use the dialog box to select the project name, drive and directory from which you want to load the project. The system automatically proposes the filename extension .PRX.

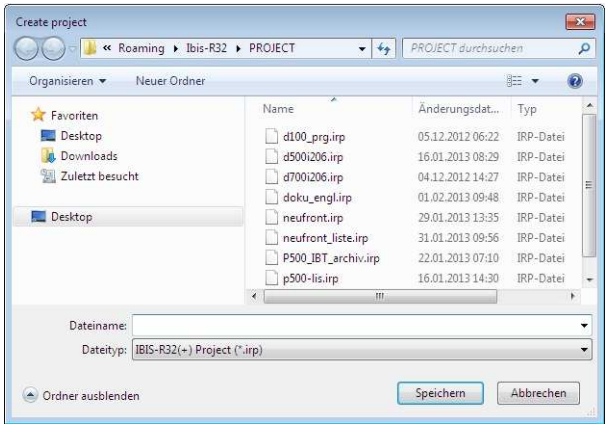
You can now access the configurators for the hardware, lists and the free-style configuration.

The Open projectfunction determines that a file exists with the filename extension .BAK, i.e., during the last project session, no final file backup was executed due to a PC crash (or similar).

You can then access the backup file from the dialog box. The file contains the same project status as when you last used the Openprojectfunction.

→[Yes] uses the last project status saved (.IRP) for the next session.
→[No] uses the backup file (.BAK) for the ext session.
→[Cancel] interrupts →Openproject.

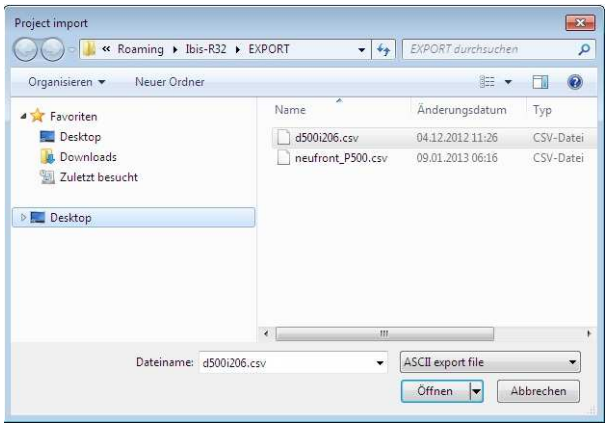
2.3 Importing a project



with →Project→Importing.

You can access a project which was generated on other computers or generated by another or an older version of IBIS-R32 only by →Project→Importing.

To import a project, first create a new project. You can then load a previously exported project into the new project (refer to “2.5 Exporting a project”).



In the dialog box, you can select the project name default, the drive and the subdirectory from which you want to load the project. The system automatically proposes the filename extension .CSV.

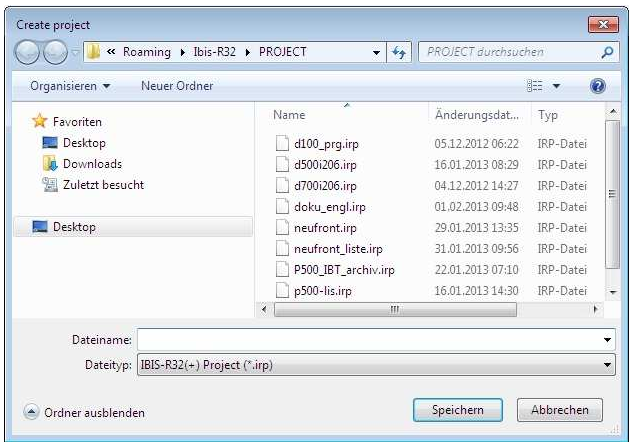
You now have access to the Hardware and List Configurators and the Free-style Configuration function.

2.4 Uploading project from a controller

with →Project→Upload.

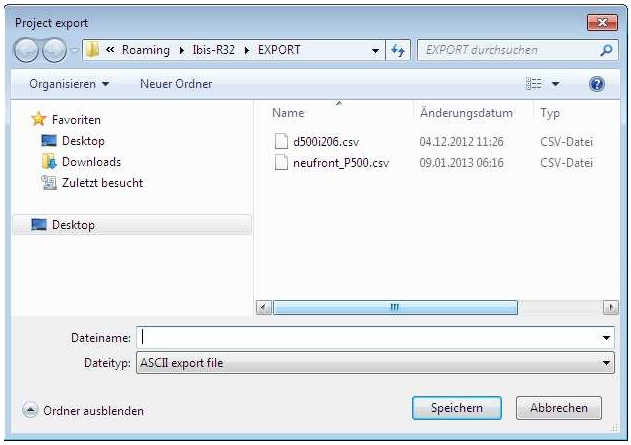
A Configuration in a controller can be uploaded at any time. A Project with Free-style configuration can only be uploaded when when the reverse documentation was created during the plausibility check.

Should there be no connection to unit, two error signals will be displayed.



The configuration in the controller is then available in the form of a project as if it had been loaded from the mass memory. It can then be manipulated.

2.5 Exporting a project

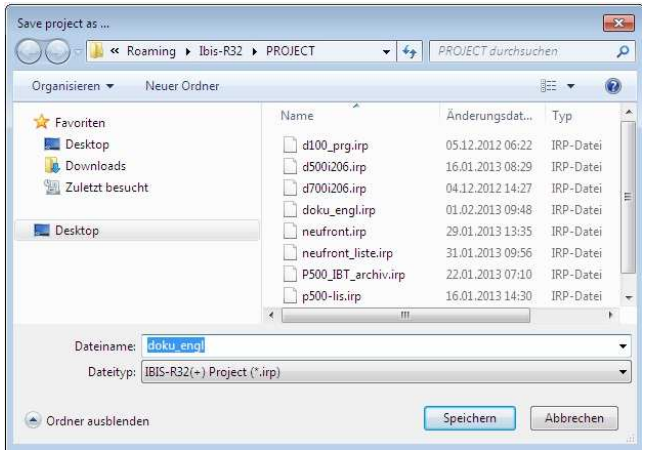


with →Project→Export.

When you use a new version of IBIS-R32 or if you want to use a project created with another library, you cannot access projects directly using the Openprojectfunction. Projects created with another library can be manipulated after exporting with the old library and importing with the new library. Moreover, when you transfer projects to other computers, problems will arise when you handle several files at a time. As a solution to this, you must export a project by →Project→Export.

In the dialog box, select the project name, the drive and the subdirectory in which you want to save the file. The system automatically proposes the filename extension .CSV.

2.6 Saving a project under a new name



with →Project→Saveas.

The project will continue to be available under its old name.

In the dialog box, select the new project name, the drive and the directory in which you want to save the files.

Since a data base backup copy is also saved as a file, it is impossible to save it to floppy disk due to the size of the file.

2.7 Closing a project

with →Project→Close

All project files are saved to mass memory and removed from the desktop on your screen.

When you close files, all the changes you have made since opening the project are saved or rejected:

→[Yes] saves the project and removes it from the desktop.

→[No] removes the project from the desktop without saving.

→[Cancel] interrupts →Close and the project remains on the desktop.

If the modifications are saved, there will be an additional query whether the project should also be exported (Refer to “1.5 Closing IBIS-R32”).

2.8 Deleting a project

with →Project→Delete.

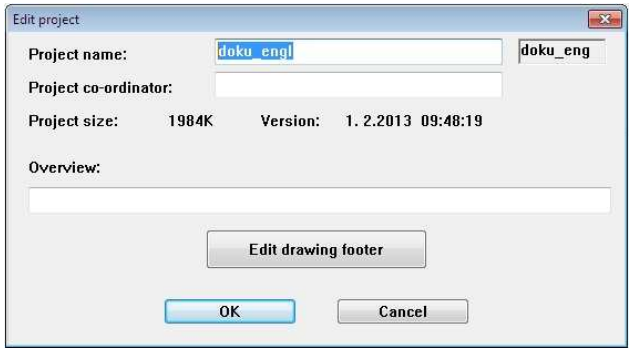
All files belonging to a project are removed from the mass memory and from the desktop. The *.CSV files generated by
→Export are not affected.

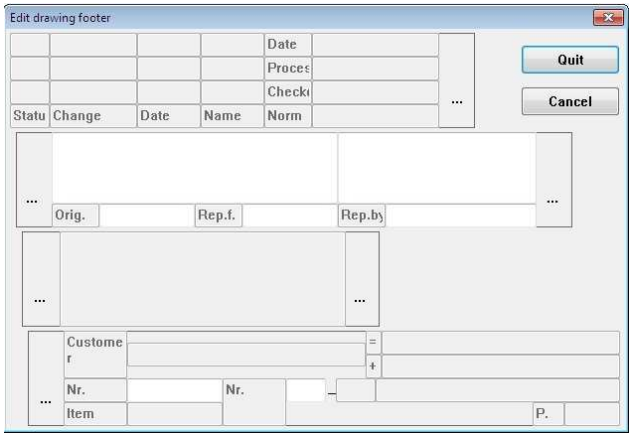
To prevent accidental deletion of a project, a dialog box requests you to confirm the deletion after you select
→Project→Delete:

→[OK] deletes the project and the related files.

→[Cancel] does not delete the files and the project remains on the desktop.

2.9 Changing the project header





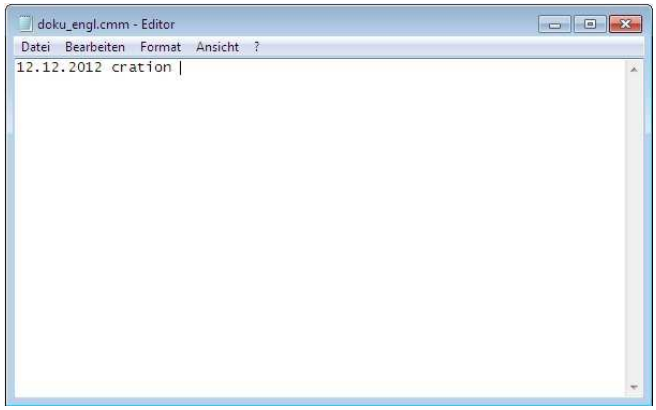
with →Project→Projectheader.

This function lets you change the project data you entered in the project header using →Project→New, with the exception of the project name.

→[Edit drawing footer]	displays a drawing footer to edit.
→[Accept]	accepts all changes made in the project header.
→[Cancel]	rejects all changes in the project header.

When you use →[Edit drawing footer], the drawing footer is displayed to enter data. You can make inputs in the fields highlighted in green. The inputs then become a part of all programs which permit a drawing footer. Program-specific data contained in the non-highlighted fields in the header are taken from the program header of the program.

2.10 Creating and editing a project comment



with →Project→Projectcomment.

In addition to a short comment displayed in the project header, a text file containing additional information can be created and edited.

You need not specify a file name. The file is provided with the filename extension .CMM. The system loads the Windows Editor for you to type in the comments.

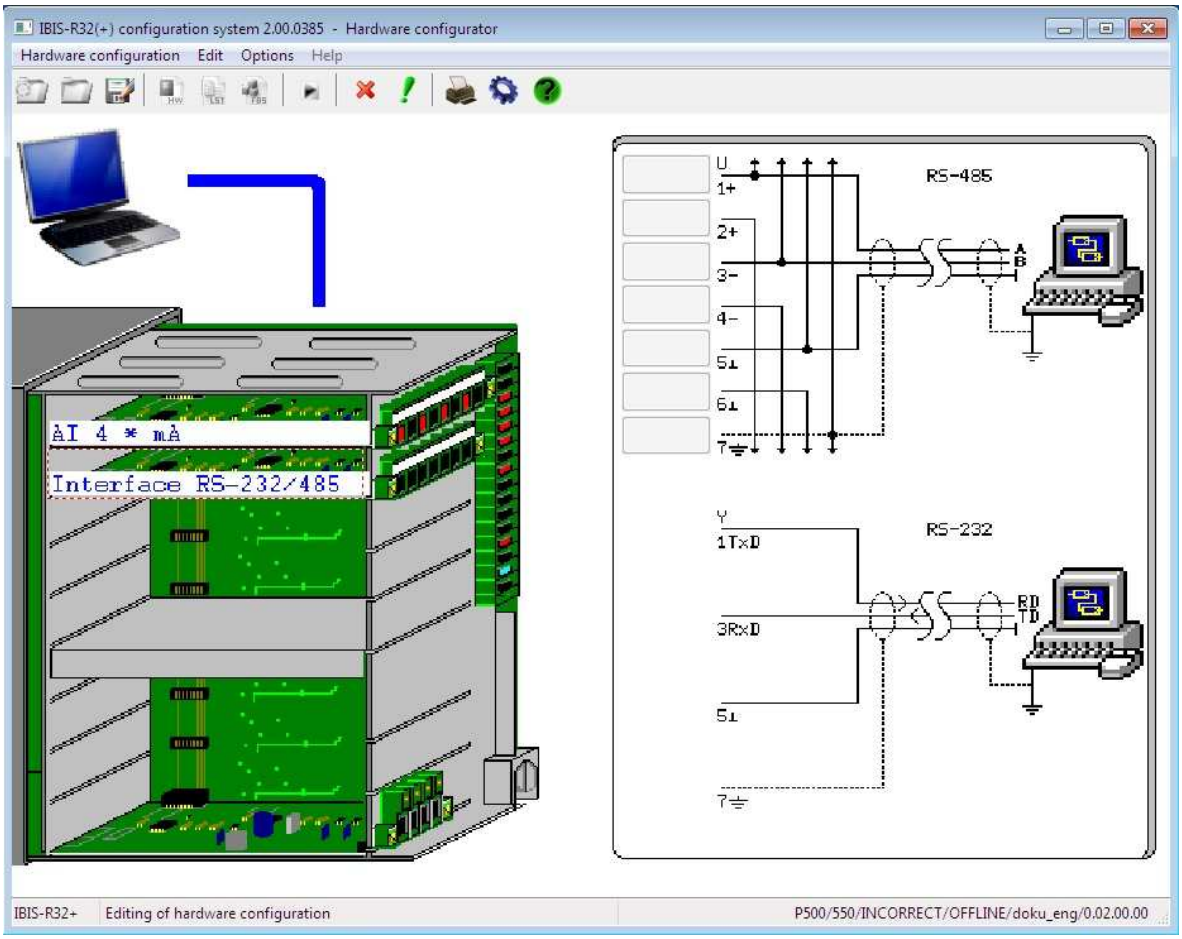
2.11 Converting a project

with →Project→Convert

Before you can use a configuration generated for a Protrenic 500 for a Protrenic 700, you must convert the program. After that you can access the configurators.

3 Hardware configurator Table of contents

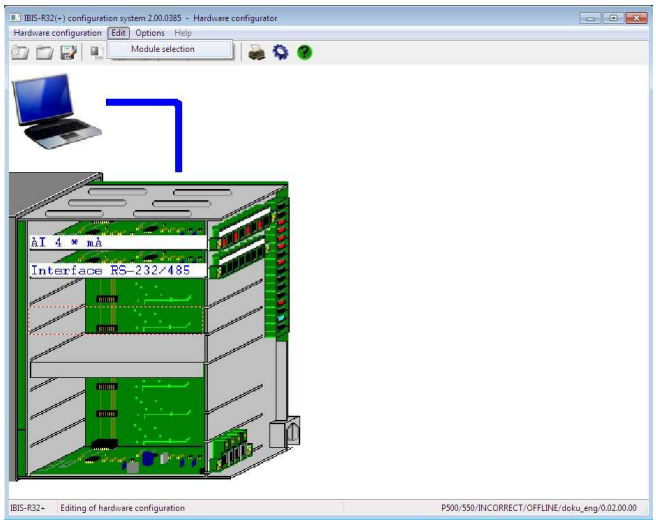
3 Hardware configurator



You can the hardware configurator to wquip a controller with modules for a project by means of computer. You can also access and process the configuration information of a controller connected in series. You can then continue your work using the list configurator to access information on an I/O point which is actually implemented in the hardware.

The computer provides a display which corresponds to the actual hardware. Moreover, when you select a slot, a graphic displays additional information for wiring the inputs and outputs.

3.1 Module configuration



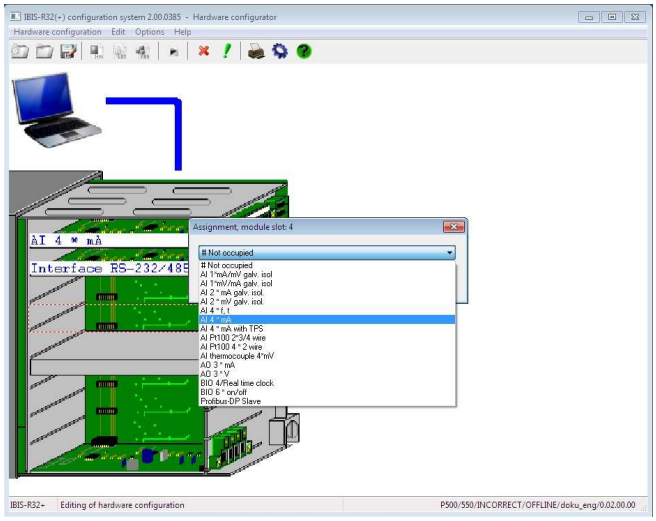
Selecting the slot
double-click a slot area or use →<↑>or →<↓>

Selecting the module list double-click a slot area or use →Edit→Moduleselectionor press the <Spacebar>. The window then refers to the slot you have selected.

Selecting a module

click the arrow button or →<Alt>→<↓> to drop down a list containing selectable modules. Click the module and confirm by choosing →[OK] or select the module by →<↑>or →<↓> and → <Enter> to transfer the configuration.

In the module selector list, the system automatically takes account of whether a module may be inserted in a particular slot. This is accompanied by a check of the total power requirement for the modules. The list only offers modules which do not exceed the total power requirement when mounted.

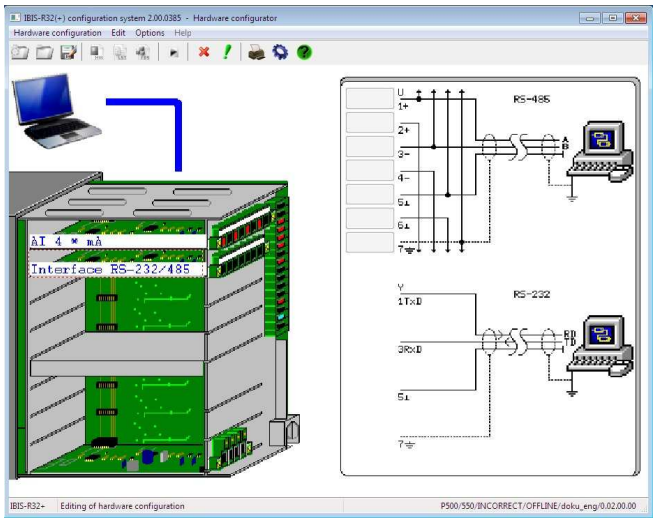


Cancelling module selection
with →[Cancel] or
→<Esc>.

The units terminal strips displayed on screen are colour-coded to indicate whether a terminal is an input or an output:

red	Input terminal
blue	Output terminal
black	Terminal is not used or is an input or output signal feedback.

3.2 I/O point configuration



After selecting or clicking an module equipped in a slot or after clicking the cable connector of the basic unit, a row of buttons, some of which are named, is displayed in the left screen half in addition to the graphic containing additional information. The names in the buttons indicate the signal type (binary (B), analog (A), input (I), output (O), not used (_)) and the numbering scheme within the total configuration.

Clicking a labelled button or →<←>or →<→> and then →<Enter>, a configuration form appears. You can then configure functions for the I/O point. Use the configurator described in the section “4 List configurator”. The section provides a detailed operating description.

As of library 3.3.0 the configured status “unused” of an input/ output point is marked with an underline "_", e.g. A_11. At the same time, all such marked input/output points are displayed in black in the terminal.

3.3 Interface configuration

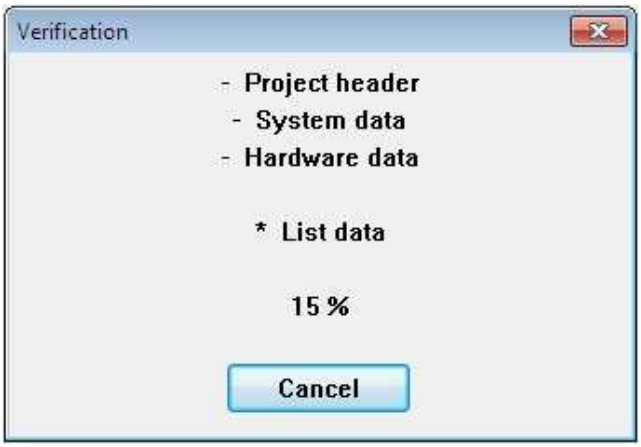
An interface must be configured if you want to upload a module configuration from a controller.
In order to execute the network or interface configuration, click the area representing the computer or →<↑>or →<↓>.
The window is built after selecting the computer by double-clicking or →<Spacebar>.

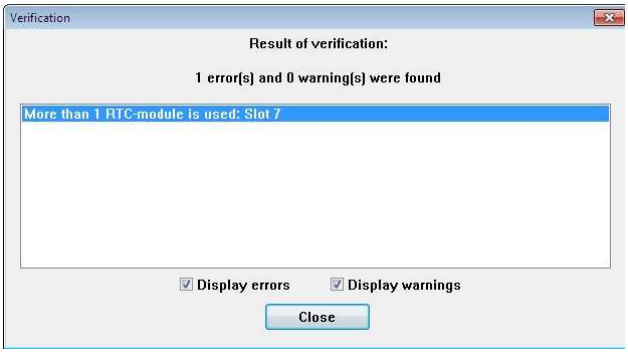
Refer to the section “1.5.4 Communication Parameters”.

3.4 Additional functions of the hardware configurator

Select additional functions or services using the Hardware configurationmenu:

3.4.1 Plausibility





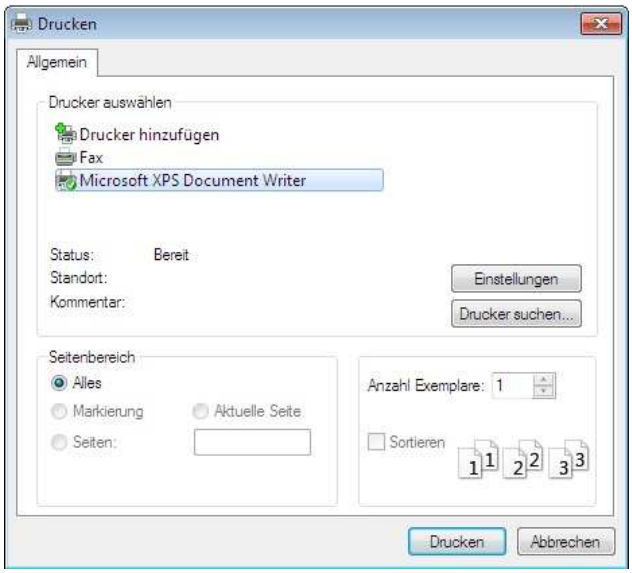
with →Hardwareconfiguration→Plausibility.

The equipment configuration is checked to see whether it can be used on a controller.

If the configuration cannot be used, error messages are displayed. They provide information on how to change the configuration.

This hardware plausibility is also processed in the List configurator for total plausibility in order to guarantee a complete check.

3.4.2 Hardcopy



with →Hardwareconfiguration→Hardcopy.

A graphic screen printout of the module configuration for a Protrenic as seen in the figure in the section “3 Hardware configurator” is obtained (The print routine uses the printer drivers of Windows).

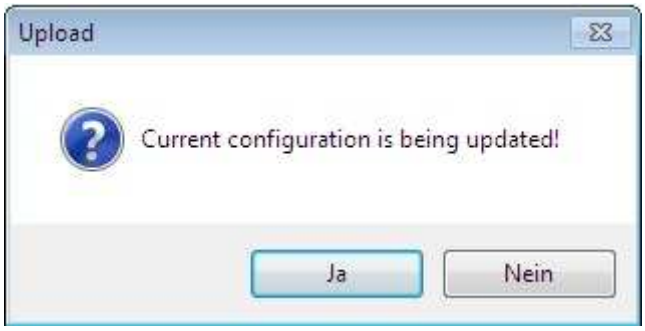
3.4.3 Uploading a module configuration from the controller

with →Hardwareconfiguration→Upload.

The existing controller configuration comprising I/O or interface modules is read out and displayed from a unit connected to the serial port.

Access via the serial port is only possible if this is configured in the transmission parameters (refer to “3.3 Interface configuration” and “1.5.4 Options Communication parameters”).

If there is no connection to the controller, two error messages are displayed.



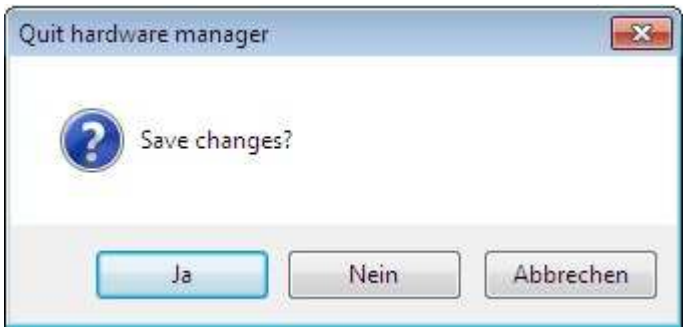
3.4.4 Saving the hardware configuration

with →Hardwareconfiguration→Save.

Saves the module configuration in the project file to mass memory.

→[Yes]	saves the current module configuration to the project file.
→[No]	ends the save operation without writing the configuration to the project file. The hardware configuration just chosen remains unchanged on the desktop.

3.4.5 Ending the hardware configurator



with →Hardwareconfiguration→Quit.

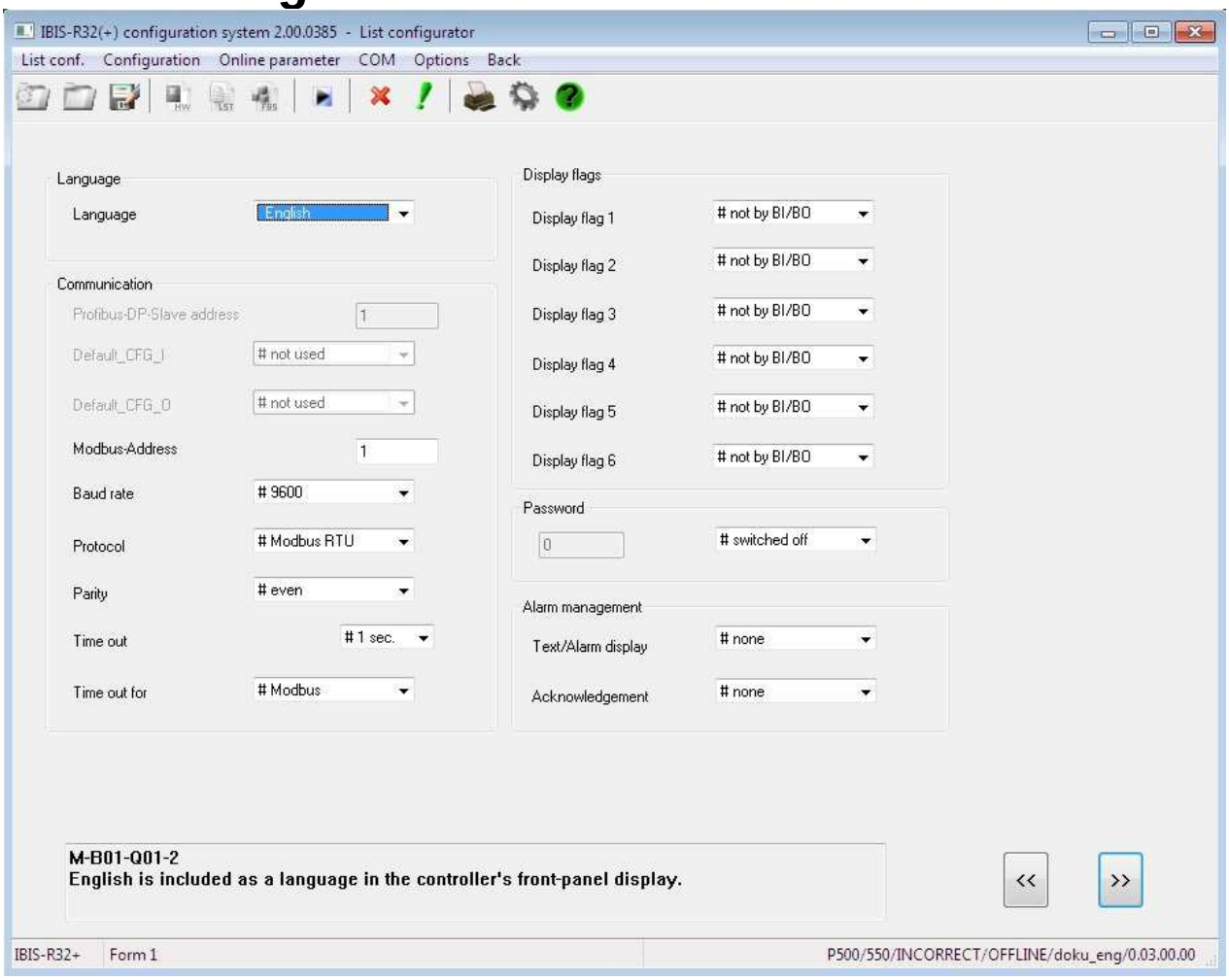
If the module configuration is not yet saved, a dialog box opens.

→[Yes]	saves the current module configuration to the project file and switches to the project manager.
→[No]	switches to the project manager without writing the configuration to the project file.
→[Cancel]	interrupts the end function and remains in the hardware configurator without saving the module configuration.

4 List configurator

Table of contents

4 List configurator



You can use the list configurator to answer queries relating to the controller configuration modules and enter the associated parameters into the forms provided. In the configuration, this takes place by selecting answers to queries or entering values for the parameters. The forms in which one or several devices are displayed can be selected by means of a simple menu system.

Entering a new project, the factory setting for answers to queries and parameter values are provided as defaults in the forms.

In order to help you in handling the list configurator to generate a project, visual cues are used to provide information on how to answer a query or whether an answer to a query is practical. This and other means are described in detail in the sections “4.1 Selecting forms” and “4.2 Completing forms”.

Select the List Configurator using →Listconf.in the menu bar.

If the list configurator is selected, while working on a free-style configuration, the following message is displayed:

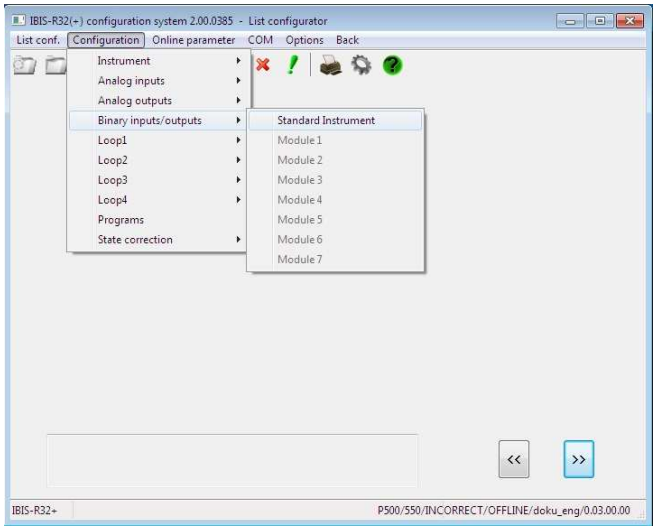
→[Yes] calls the list configurator while keeping the free style configuration.

→[No] calls the list configurator while rejecting the free style configuration completely and continues the project as list configuration.

→[Cancel] does not call the list configurator and leaves the project unchanged.

If the list configurator is called up and the free configuration retained, changes to the list data can be made. These changes are mainly those not covered by elements of the free configuration. For example the processing status of an input/output point. Here, it is only important if the point remains unused or is somehow used. The direction of action for binary inputs or outputs or the type of connected primary detector for analog inputs is not important. This information is also not automatically adopted into the free configuration. The signal processing for analog signals and the direction of action of binary inputs and outputs must be created by the user in the free configuration through functions and function modules.

4.1 Selecting forms



Forms can be selected from the menu bar or by pressing →[<<] or →[>>] to switch to the previous or next form.

Selecting from the menu bar, a hierarchical structure of submenus is offered. A black triangle

▸ next to a command indicates a submenu containing additional commands. If there is no triangle, you can only select the command if you call a form.

Example:
→Configuration→Binaryinputs/outputs→basicmodule

If the command for a form is dimmed, you need not or cannot complete the form. One reason may be that a module slot is not equipped with a specific module type (see modules 1 to 7 in the above figure: a binary input/output module is not plugged into slots 1 to 7).

If the command for a form is black, the form can be displayed if it contains queries or parameters which need to be either answered or entered.

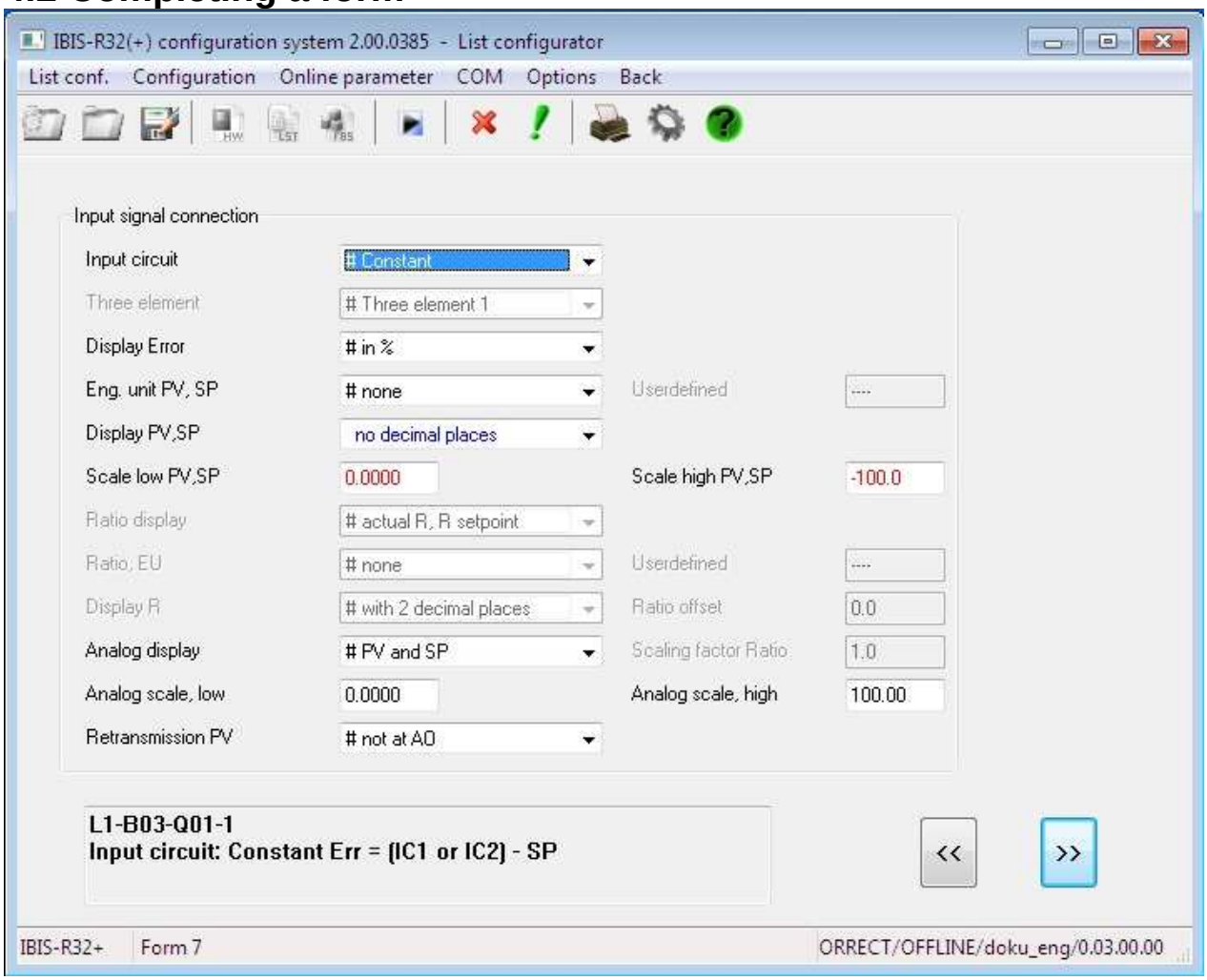
If no new form is displayed, no configuration information in this form is required to continue the session.

Forms are closed when you select forms by pressing →[<<] or
→[>>]. This ensures that only forms which are required for further processing in a project are displayed for completion.

When working in a form, press →<Alt> to switch from the form to the menu bar to make further selections.

To help to understand what a particular selected query or parameter means, a descriptive text is displayed at the bottom left. The text contains a code indicating the group, device/query number or parameter number. With parameters, the value range and the unit are also displayed.

4.2 Completing a form



When a form is displayed, either queries need to be answered or the values of parameters need to be changed. The form contains several fields containing text. List boxes linked to the query text are provided to enter answers to queries. Text boxes linked to parameter names are provided to enter parameters.

As a guide to understanding, different colours and signs are used to display query, answer and parameter texts in the forms. Their meaning is explained below:

black
Queries and parameters which are required for further pro
cessing. The answers to queries or parameter values are factory settings.

grey
Queries and parameter texts which are not necessary for further processing. You cannot select them (neither by mouse nor by keyboard). With parameters, the parameter value is also displayed in grey.

Answer to a selectable (displayed in black) query which is not permitted as an answer and which should be changed.

blue
Answers to a selectable (displayed in black) query and parameter texts which are not factory settings. They are displayed in black when valid. This helps you to view at a glance the changes you have made in a form.

red
Parameter values which are not practical and could lead to problems. These are for example scales in which the initial value is greater than the final value.

#
Answers which correspond to the factory setting of a query.

You can perform the following actions in a form:

Select a query

→<Tab> or →<Shift+Tab>, possibly several times, to select a query.

→Answertextand a list box drops down.

Answer a query with open list box

→<↑>or →<↓> to select the answer. At the same time, the long text switches over to the displayed answer.
→<Alt+Enter> to end input.

Select the answer text you want. At the same time, the long text is displayed for the selected answer. If more than 5 answers are available in a drop-down list box, a scroll bar appears on the left. Use the scroll bar to select the answer. You can reach answers contained above or below the dropdown list box by moving a pixel above or below the list box.

Release the left mouse button to select the answer.

Both selections have in common that you can only select answers displayed in black or blue.

Answer a query without list box open

Only possible using the keyboard. →<↑> bzw. →<↓>to select the answer. The input is closed automatically..

Selecting a parameter

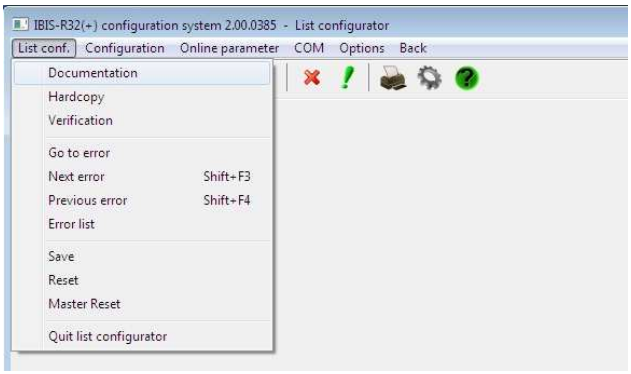
→<Tab> or →<Shift+Tab>, possibly press several times, to select the parameter.
→Parametervalueat the location where you want to change the value.

At the same time, the long text is displayed for the selected parameter.

Enter parameter value

By keyboard input based on Windows standard. The input is automatically checked for the maximum number of possible characters.
→<Tab> or select another parameter or query to close the input. Inputs are then checked for feasibility and incorrect inputs are displayed in an error window.

4.3 Additional actions



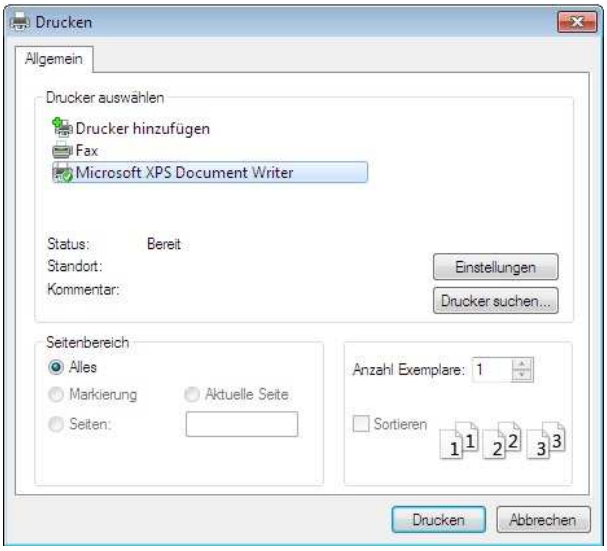
Additional actions can be called with →Listconf.

4.3.1 Creating a document

with →List conf.→Documentation→Documentation→Add new documentations job.

Use this command to print a configuration which the Documentation Manager can access. Refer to the section “7 Documentation Management”.

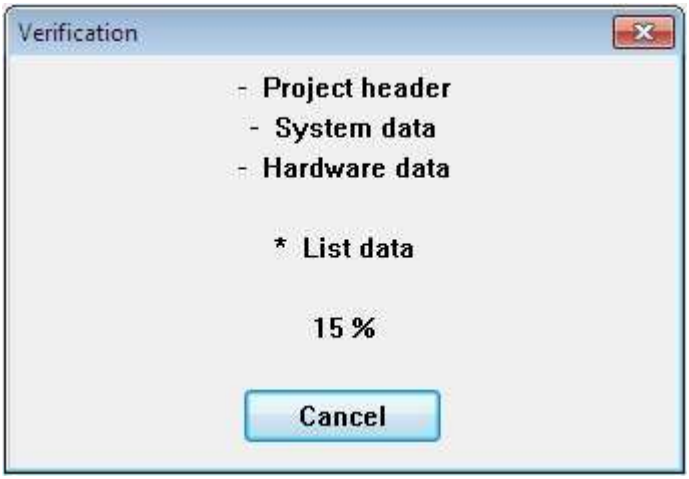
4.3.2 Hardcopy



with →Listenkonf→Hardcopy.

A graphic screen printout of a form is executed by a Windows printer driver.

4.3.3 Plausibility check of a configuration



with →Listconf.→ Plausibility.

The entire configuration is checked for possible error sources. Errors may include invalid answers to queries or incorrect values which are not practical in connection with other values.

During display of the plausibility errors, an error can be selected and the form in question can be called up directly with doubleclick or with →List configurator→Go to error. The query or the required parameter is directly set to the missing or wrong ques-tion/parameter and can thus be corrected.

Also using →Listconfigurator→Nexterrorand →Listconfigurator→Previouserror, the next or previous error within the error list can be selected for correction.

With the aid of →List configurator→Error list, the error list created with the last plausibility check can be displayed.

→[Close] closes the plausibility window.

4.3.4 Saving a configuration

with →List conf.→Save. All previously executed configuration changes are saved to the .PRX file on the hard disk.

4.3.5 Resetting configuration to previous state saved

with →Listconf.→Reset. The entire configuration is reset to the previous configuration status saved using →Listconf.→Save.

4.3.6 Setting configuration to factory setting

with →Listconf.→Master Reset

The entire configuration is reset to the factory setting of the controller type selected for the project.

The configuration state at the time of opening a project can be accessed by leaving IBIS-R32 with →IBIS-R32 beenden→“Save changes?” →[No], the configuration state saved with →List conf.→Save can be accessed with →Listconf.→Reset.

4.3.7 Ending list configurator

with →Listconf.→Quitlistconfigurator. The program switches to the opening menu.

4.3.8 Return

with →Return.

The program switches back to where the list configurator was called from.

4.4 Commisioning (COM)

Open the startup dialog box by selecting →COM. Refer to the section “6 Commisioning”.

5 Free-style configuration

Table of contents

5 Free-style configuration

Free-style configuration enables functional capabilities not contained in the list configurator to be compiled as configuration and loaded into the device.

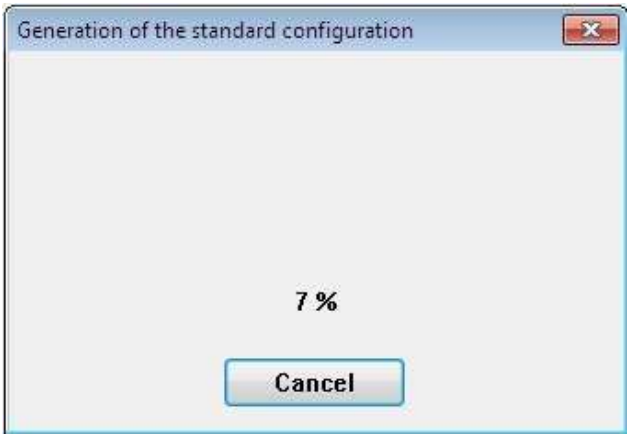
5.1 Dongle

A dongle is necessary for operating IBIS-R32. It is supplied with the software and plugged onto USB-interface of the PC.

An error message is outputted if no dongle is plugged onto USB-interface when free-style configuration is called.

→[Yes] continues the program after the dongle has been plugged on.
→[No] exits free-style configuration.

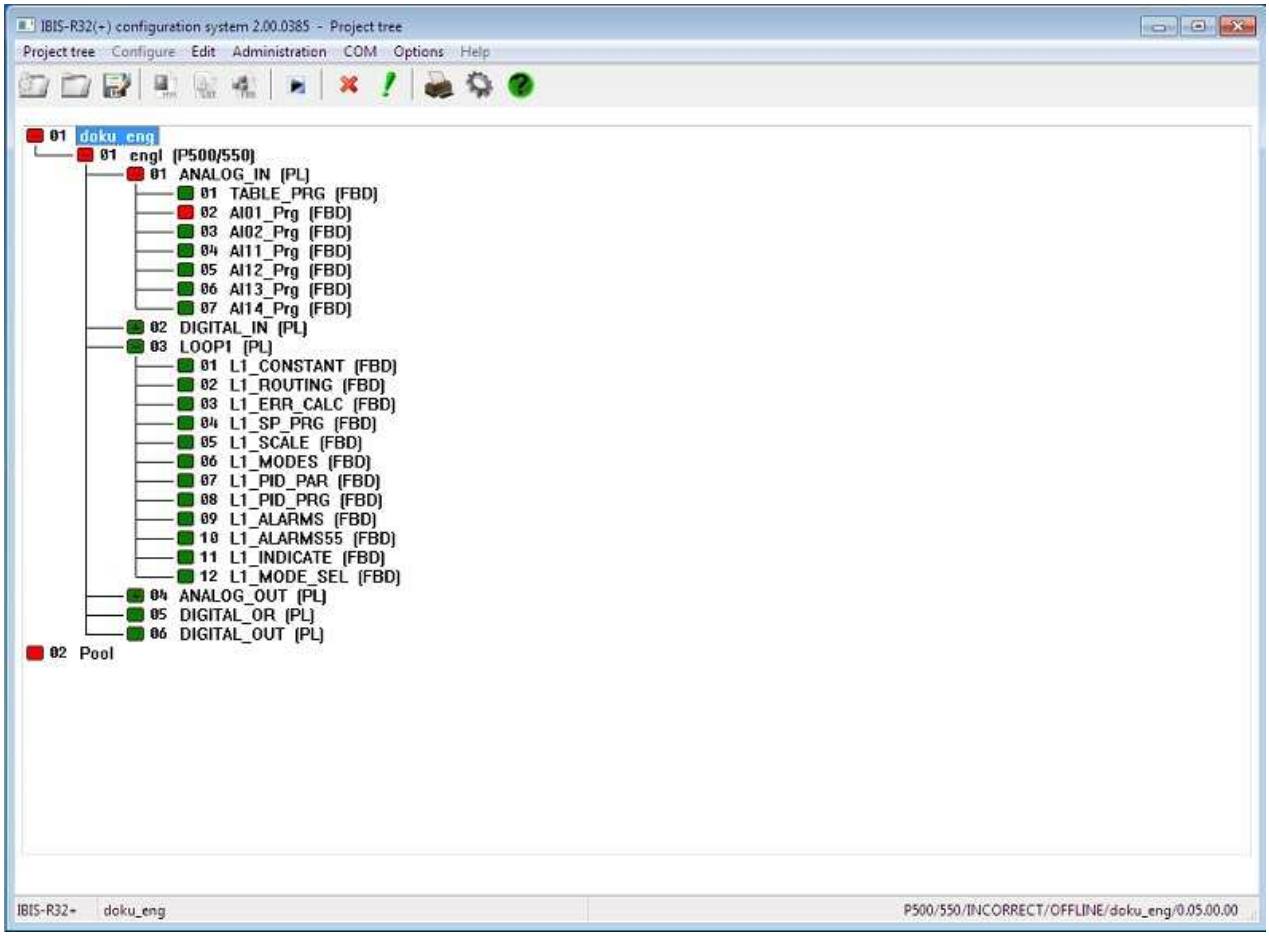
5.2 Standard generation



After entry into free-style configuration, a graphical equivalent for further processing is constructed from the current list configuration. The structure and elements of this configuration are represented by the project tree.

Entry into free-style configuration is only possible with a correct and plausible project of the list configurator. See Section 4.3.3 “Plausibility checking a configuration”.

5.3 Project tree



A project is displayed in tree structure in the project tree. It is used to provide better overview and rapid selection of project tree objects. The individual objects are implemented to IEC 1131-3. Simple guidance can be obtained from the abbreviations of object type set in brackets. The nodes in front of the object indicates processing status and the possibilities of branching.

The project tree is divided into two levels, the process level and the pool:
All functions and programs in the process level can be loaded into the controller.

In contrast, the pool is a “store” of project tree objects which either are not plausible or are no longer required in the execution, but which you may possibly want to bring back to the process level.

Process tree objects can be selected via keyboard or mouse.

5.3.1 Structure of the project tree

← Menu line
← Project tree with nodes

⊞ The path is open: there are further branches.

⊟ The path is closed.
☐ There is no further branch.

■ (red) Object has been changed, there has still been no plausibility checking or the outcome of the plausibility checking is not error-free.

■ (green) The outcome of the plausibility checking is error free.

The colour settings are defined, and can be changed in the project tree by using the colours options.

← Status line

The figure next to the nodes shows the execution sequence of the project tree objects in the corresponding level:

1st line	This contains the project name assigned. The project file is stored under the corresponding name (with the extension ".prx") in the directory chosen during installation.
(P500/550) Ressource	Describes the hardware on which the project is based.
(PL)	Program list. IL and FBD programs are executed according to their execution number.
(IL) IL-Prog.	Program which has been compiled in the language instruction list
(FBD) FBD-Prog.	Program in the function module language.
Pool	"Store" of project tree objects which are either not plausible or no longer required in the execution, but which you may possibly want to bring back to the process level.

The following are displayed in the status line, depending on the interface:
− note on the controller,
− the directory in which the project file is located,
− the object selected in short or long form,
− the program status and
− configuration notes.

5.3.2 Functions of the project tree

A series of menu items have not been described in this section and are therefore provided with an appropriate reference.

Project
- Saving
- Documentation (see “7 Documentation”)
- Hardcopy
- Plausibility checking
- Exporting programs
- Importing programs
- Ending project tree

Configuration
- Processing program selected

Processing
- with sub-menu items

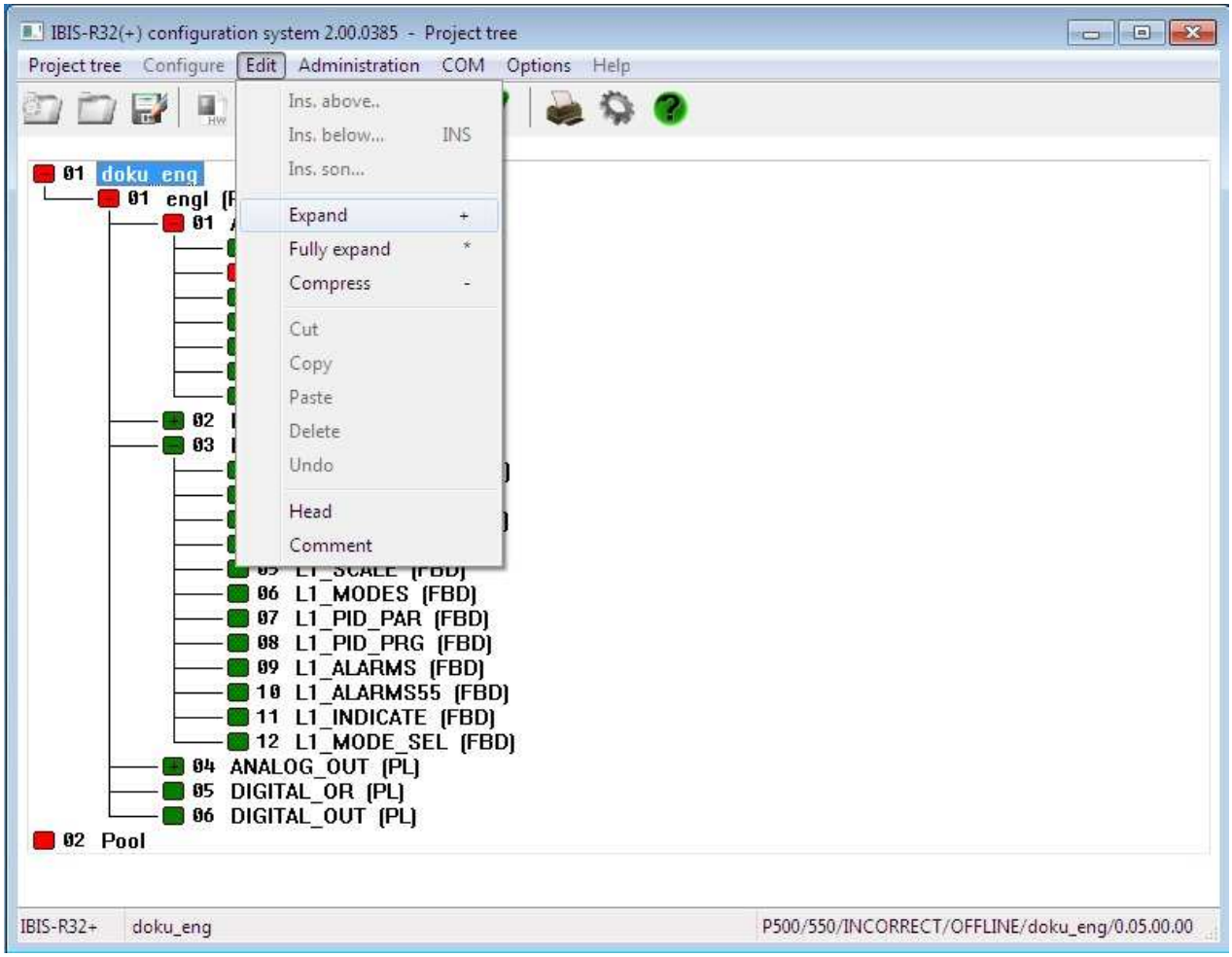
Administration
- Variables administration
  - Loop tag administration

COM
- Calling the commissioning function (see “6 Commissioning”)

Options
- Password configuration Long display

Help
- is not supported at present.

5.3.3 Compiling/processing the project tree



The project tree is used for structuring the project. The structure The project tree can be structured by selecting →InsertAbove, is established for each controller and is displayed as follows: →Insertbelow, →Insertson.

Project→Resource→Program lists→Programs

Contents of the object resource

For the user the header of this object contains important information which are always renewed after a plausibility check.

The size of the memory for self-defined user variables is currently set to 1 KByte. A list of the self-defined user variables containing information on how this is distributed within the internal memory is obtained by pressing [Display ...]. If there is no defined user variable, a corresponding notice will be displayed instead.
Likewise, information of the need of the RAM and flash memories on the controller for the just processed object is output.

In the RAM memory of the controller are stored the entire dynamic data (signals) and information required for processing the functions and function modules. All configuration information is stored with mains-failure protection in the flash memory.

Expanding, fully expanding, compressing

It is possible to open or close individual project parts in order to increase clarity in the project tree.

Expanding:

Only possible where a node is displayed with + opened by one level.

→select node with mouseclick
→Process
→Expand

or via keyboard::
→select position in the project tree <↑>,<↓>
→expand <+>

Fully expanding:

Only possible where a node is displayed with + . The node is completely opened.

→select node with mouseclick
→Process
→Fully expand

or via keyboard:

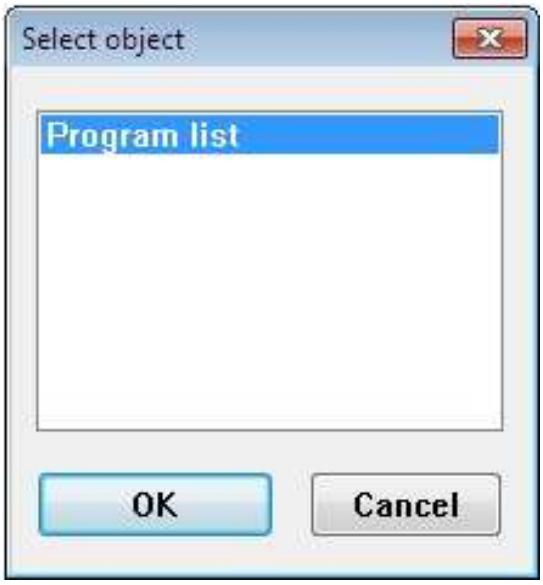
→select the position in the project tree <↑>,<↓>
→fully expand <*>

Compressing:

Only possible if node is not displayed with + or - . The node is reduced to a project tree object.

→select node with mouseclick
→Process
→Compress or via keyboard:
→select the position in the project tree <↑>,<↓>
→compress <−>

Insertion of objects



with →Process

→Ins. above
inserts a new object above the one selected.

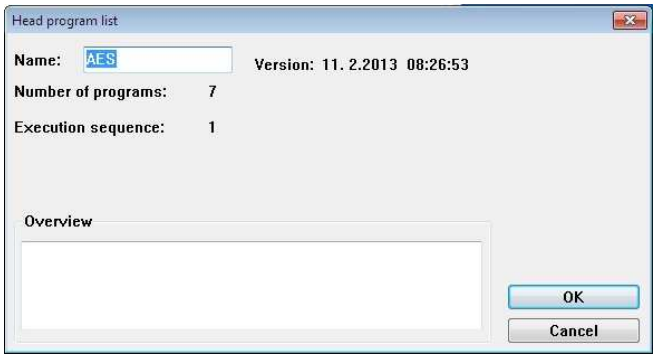
. →Ins. below
inserts a new object below the one selected.

. →Insert son
inserts first object one level lower.

The associated “object selection” window is opened, depending on the object selected in the project tree. Names are assigned to objects in the “header program list” window (see below).

. →select objectz type with mouseclick
. →[OK]

or via keyboard:
→<↑>, <↓>→<Enter>



Enter object names and perhaps short comment in the “header ....” window. Each object must be provided with an unambiguous name. All upper and lower case letters, numerals and “_” (underline) as special character are permissible. The name of an object can be 12 characters (maximum) in length.

The header of each project tree object can be processed by
→Process→Header

The commnet of each project tree object can be processed by
→Process→Comment

Move, delete ... project elements

Using the menu, project tree objects can be cut, copied, inserted or deleted -individually or in blocks -as can entire paths with sub-paths. Exceptions are the project name and the resource created by default. It is possible to move project elements using the mouse, without going via the menu. If an operation is not possible with the object(s), the menu entry is displayed in grey.

1. Individual objects:

→select the desired project element by mouseclick
or via keyboard:
→select the desired project element with <↑>, <↓>

The project element has been selected for further processing (coloured background).

Copying:

→select the desired project element by mouseclick
→Process
→Copy

or via keyboard:

→select the desired project element with <↑>,<↓>
→<Ctrl + Ins>

Cutting:
→select the desired project element by mouseclick
→Process
→Cut

or via keyboard:

→select the desired project element with <↑>,<↓>
→<Shift + Del>

Inserting:

A project element must have been copied or cut beforehand. If the insertion position is not permitted, insertion is displayed light grey in the menu.

→select insertion position
→Process
→Insert

or via keyboard:

→select the desired project element with <↑>,<↓>
→<Shift + Ins>

The “insertion” window, by which the insertion position “above” or “below” must be stipulated, is opened. An unambiguous name must be assigned to each project element which is copied and inserted.

Deleting:

→select the desired project element by mouseclick
→Process
→Delete

or via keyboard:

→select the desired project element with <↑>,<↓>
→<Del>

If the node is displayed thus ???, the “Delete” window containing the question “do you really want to delete object: programs go to the pool?” is opened.

Note
The delete procedure can be reversed using →Process→Cancel. Deletion **after** saving the program and the project is irreversible.

Cut and insert (= move):

→select project element a second time by mouse click and hold mouse key pressed.
→Move mouse to the entry position.
A symbol appears which indicates that insertion is allowed or not possible.
→Release mouse key at the desired position.

The “move” window, by which the insertion position “above” or “below” must be stipulated, appears.

2. Several project elements (= block):

The project elements are so to speak stretched in a frame and are selected for further processing (coloured background). It is possible to proceed with blocks in the same way as with individual project elements (see above).

→select first desired project element with mouseclick and hold that key pressed.
→move mouse to the next (next but one etc.) project element.
→ release the mouse key at the desired position.
or via keyboard:
→select first desired project element with <↑>, <←>, <↓>, <→>
→press and hold down <Shift>
→move to the next (next but one etc.) project element with <↑>, <←>, <↓>, <→>
→ release <Shift> at the desired position.

Cancel
with →Process→Cancel.
Cancels the last action performed.

Configuring header of the project tree objects

with →Select object→Process→Header.

The “configuration ...” window is opened automatically when a new project tree object is filed. The name of the project tree object (and if required a short comment) is then entered in this window. This information can be changed later via the menu.

See “Processing a project tree object comment” and “Saving a projct tree object comment”.

Additional information is displayed, depending on object type, such as:

- Type of the object
- Version (date and time of creation or last change)
- Number of lower-order project tree objects
- Execution sequence

Processing a project tree object comment

with →Select object→Project→Comment

User-defined text (comments, short descriptions) can be compiled and changed using the comment editor. It is called from the project tree and from programs. The comment is assigned to the particular object. A temporary working copy is created when the editor is called, so that the last version created is retained unchanged, against the possibility of the text processing being aborted.

Saving a project tree object comment

with →Save.

The comment is stored in the project file with the extension

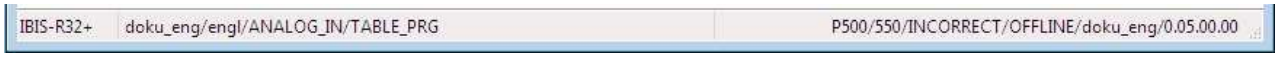
“.log”.

5.3.4 Options of the project tree **Hardcopy**

with →Options→Hardcopy.

The contents of the screen or a block is printed out by a windows printer driver.

Long display



with →Options→Longdisplay.

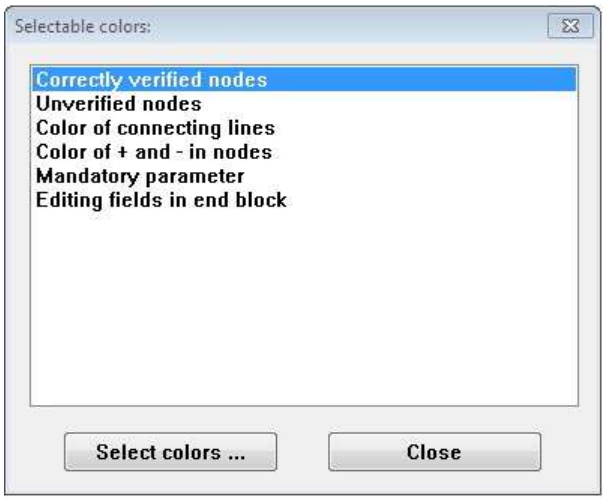
The names of the project tree objects are displayed in the status line.

Short display

with →Options→Shortdisplay.

Short forms for the respective levels are displayed in the status line.

Defining colours



with →Options→Colours.

The colours of lines linking node, connecting lines, “must” parameters and fields in the drawing footer can be changed.

The following colours are preset:

Correctly plausibility checked nodes	green
Nodes not (or not yet) plausibility checked	pink
Colour of connecting lines	black
Colour of + and -in the nodes	black
“Must” parameters	red
Editing fields in the drawing footer	light green
Nodes correct, resource must be used	red

(→[Close] exits colour selection.)

→select entry

→[Select colours ...] opens the colour-selection window.

The colours can be defined in the colour-selection window.

→[Define custom colours >>] opens the colour definition menu.

→[OK] exits colour selection and accepts the selected colours.

→[Abbrechen] exits colour selection without accepting the selected colours.

Password

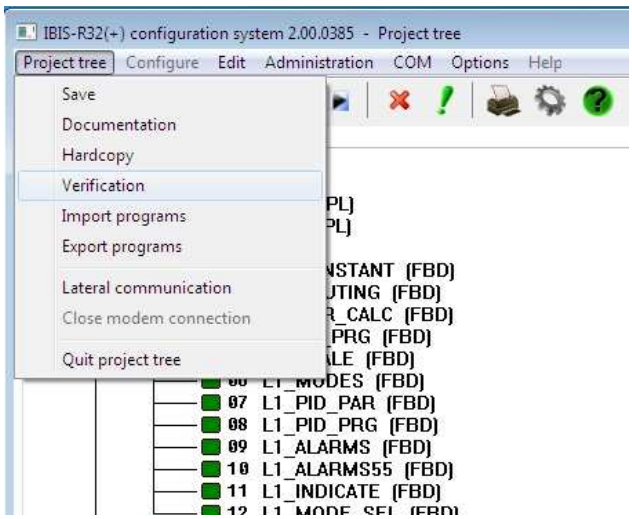
with → Options→ Password configuration

(see “1.5.4 Options”)

The uploading of a user-defined configuration can be disabled against unauthorised access by entry of a Fupla password. Because this password is filed in the memory of the Protrenic, it must be entered during commissioning and before downloading

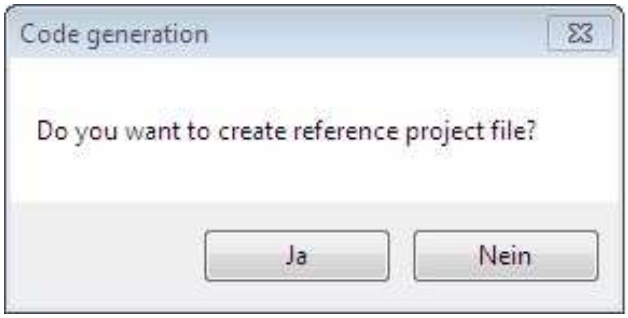
→[Enter]	accepts the word given in the password entry.
→[Cancel]	abortss the current entry without accepting the word.

5.3.5 Plausibility checking



with →Select →Projecttree→ Verification.

The selected project tree objects and all those below are plausibility checked. Plausible project tree objects are displayed in green, non plausible project tree objects in pink.

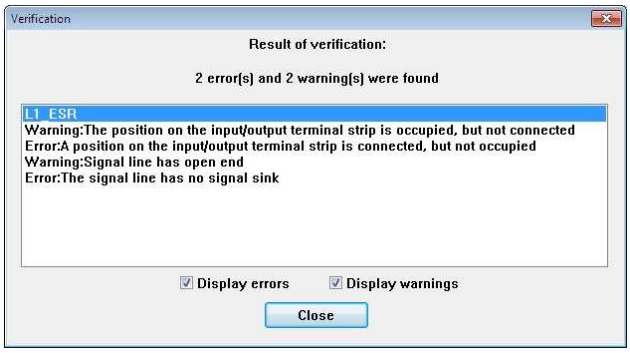


If the entire project is correct after plausibility checking, then reverse documentation can be produced containing the user-de-fined configuration in the form of a graphic.

This can be also be loaded in the controller during commissioning (COM).

→[Yes] generates reverse documentation.

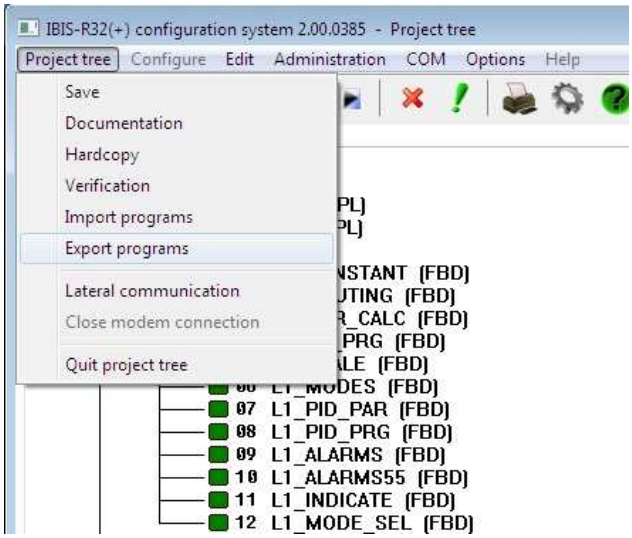
→[No] doesn't generate reverse documentation.



If an error is found during the plausibility checking, this is outputted with information of the project tree object.

→[Close] terminates the display of the plausibility-checking window.

5.3.6 Exporting program



Programs can be saved as files for use in other projects.

→select first program desired with mouseclick
→If several programs are to be exported, keep left mouse key pressed and move mouse to the next program.
→release mouse key at last program to be exported

or via keyboard:

→select first program desired with <↑>, <←>, <↓>, <→>
→If several programs are to be exported, press and hold down <Shift> and move to next (next but one etc.) program using <↑>, <←>, <↓>, <→>.
→release <Shift> at last program to be exported

→Project tree
→Export programm

You can use the dialog box to select the project name, drive and directory from which you want to load the project. The system automatically proposes the filename extension .PRG.

5.3.7 Importing programm

The programs to be imported as file can be imported into the pool.

→Project tree
→Import programm
→Select file

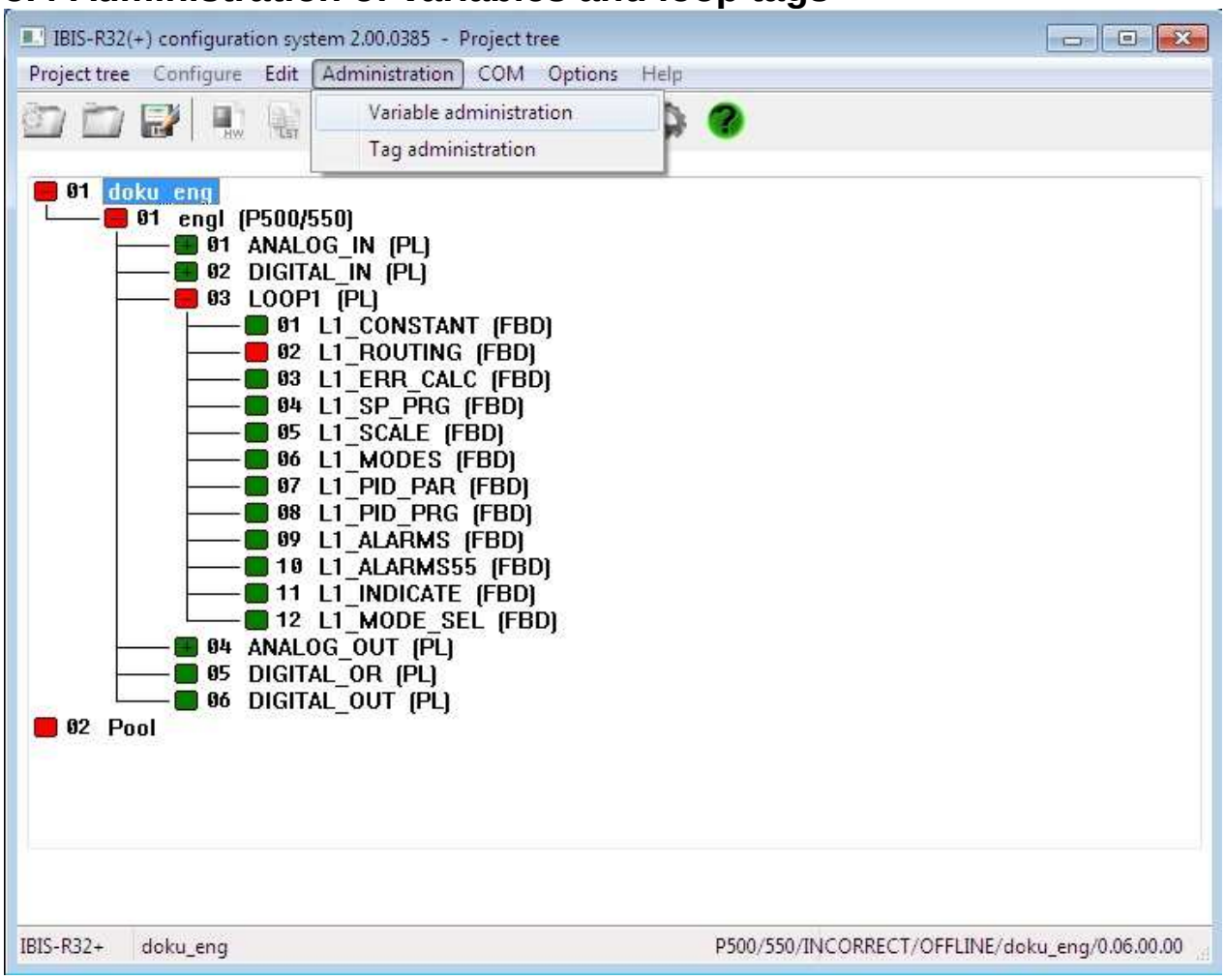
Programs can be copied from the pool to the desired position in the project tree, as described in the Section “5.3.3 Compiling/ processing the project tree”.

5.3.8 Ending project

In order to terminate the user-defined configuration, you must be The program returns to project administration. in the configuration interface of the project tree.

→Project tree→Terminate project tree

5.4 Administration of variables and loop tags



All signals and the nominated modules (measurement and control modules) wich have been cofigured are managed by the system in the form of lists, and are made aviable to the use. These lists are called:

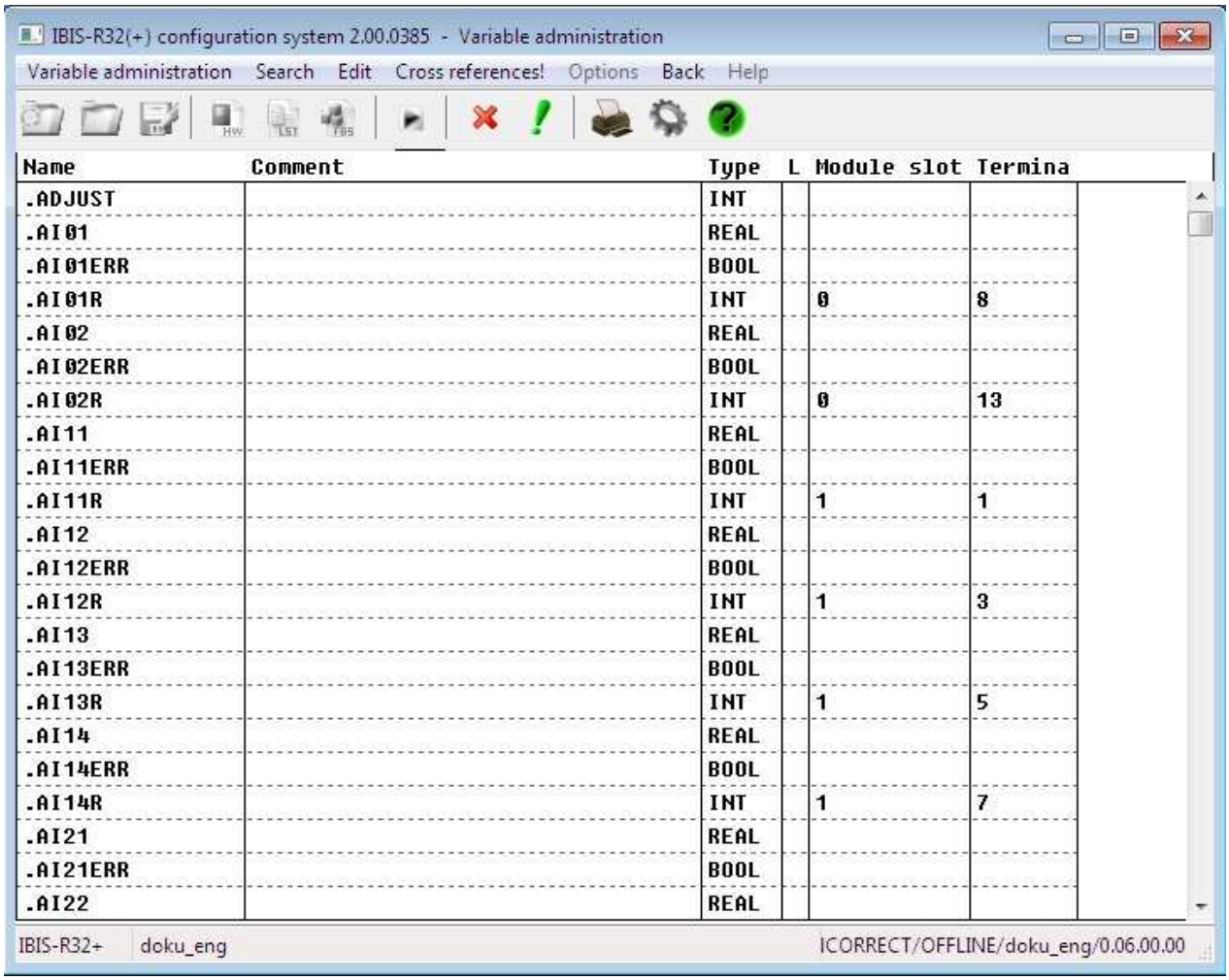
Variables list	used for all signals used.
Loop tag list	used for all modules.

These lists are produkt automatically when a program is configured, although they can be created and corrected directly.

Calling the editors

with → Administration→Variables administration
or
with → Administration→Loop tag administration

5.4.1 Structure of the variables list



Name	Comment	Type	L	Module slot	Termina
.ADJUST		INT			
.AI01		REAL			
.AI01ERR		BOOL			
.AI01R		INT		0	8
.AI02		REAL			
.AI02ERR		BOOL			
.AI02R		INT		0	13
.AI11		REAL			
.AI11ERR		BOOL			
.AI11R		INT		1	1
.AI12		REAL			
.AI12ERR		BOOL			
.AI12R		INT		1	3
.AI13		REAL			
.AI13ERR		BOOL			
.AI13R		INT		1	5
.AI14		REAL			
.AI14ERR		BOOL			
.AI14R		INT		1	7
.AI21		REAL			
.AI21ERR		BOOL			
.AI22		REAL			

The variables list contains all signals contained in the project, together with their attributes:

Name Signal name, max. 16 characters

Comment Comment on the signal

Type Signal type

Slot Card slot names of the module, not changeable within the list

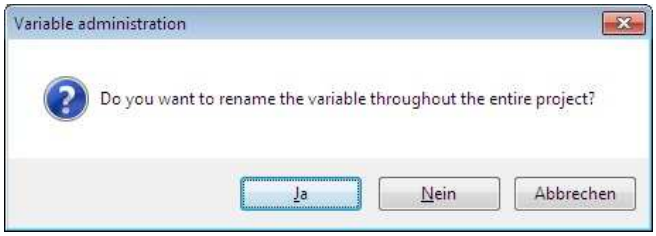
Terminal Terminal number on the module (0...7), not changeable within the list

5.4.2 Changing variables entries

If changes are made with existing variables, then these may affect the various programs. To avoid errors, the programs concerned are listed when such changes are made. It can then be decided in which programs the changes are to be effective.

→select desired field (“Name”, “Comment”, “Type”) with doubleclick

“Name” changes the name of the variable and is terminated with →<Enter>.



If the name is changed, a new window appears:

→[Yes] renames the variable throughout the project.

→[No] shows a new window: see next page.

→[Cancel] cancels the name change.

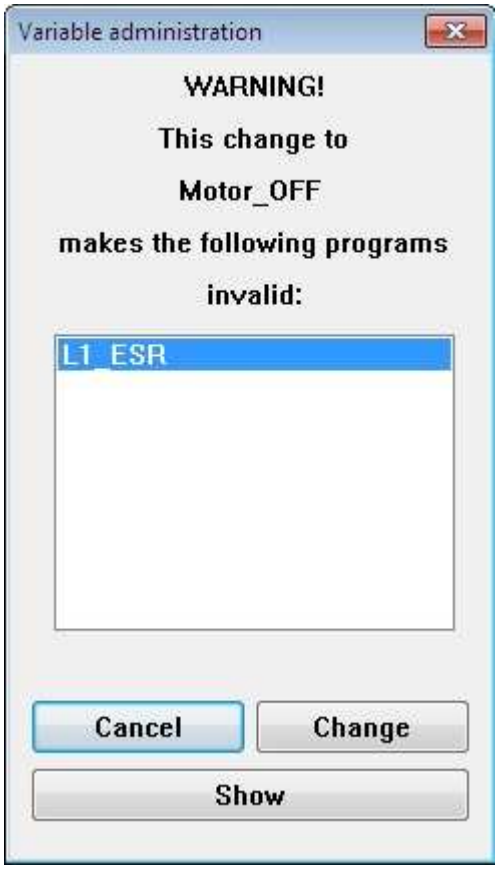
“Comment” changes the comment on variable.

“Type” shows the signal type windows and selects signal type.

→[Ok] (or →<Enter>) accepts selected signal type.

If the signal type is changed, a new window appears: see next page.

→[Cancel] exits the mask without signal type change.
No other entries can be changed within the variables list.



Whenever the query for name change within the entire project is anwered with [No], (see preceding page) or whenever the signal type is changed, a new window will appear with a list of the programs in question.

→[Cancel] exits the mask without signal-type or name change.

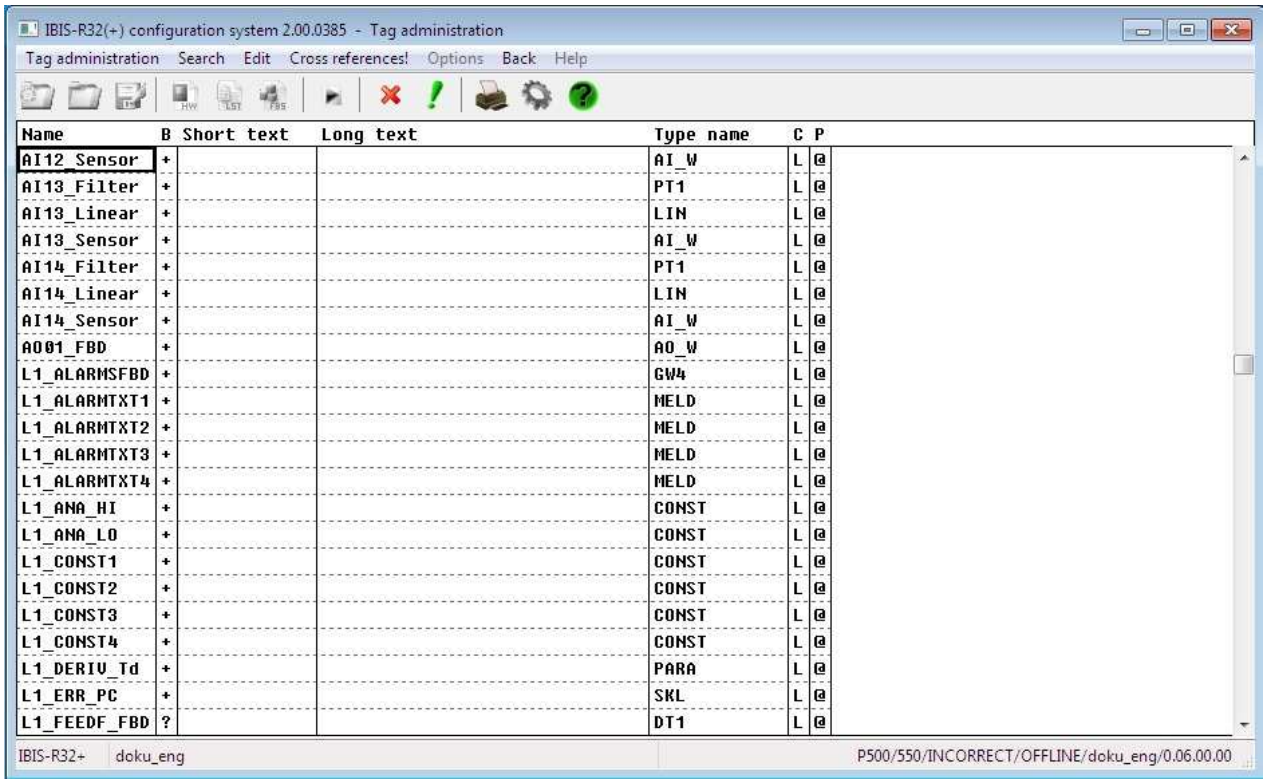
→[Change] performs changes in the program referred to.

→[Show] jumps directly to the selected program.

5.4.3 General predefined global variables

When compiling a project, certain system variables are predefined automatically and made available to the user. They are entered in the variables list and can be interrogated within the projects. They can be further processed if required. The structure of the names is such that a decimal point is placed in front of the names of variables. These variables cannot be deleted!

5.4.4 Structure of the loop tag administration



The loop tag lists includes all function modules contained in the project:

Name — Name of the loop tag, max. 12 characters

B — Processing in the parameters mask
- ? defined via input
- \+ switched on
- \- switched off

Short text — Short text of the loop tag, max. 12 characters

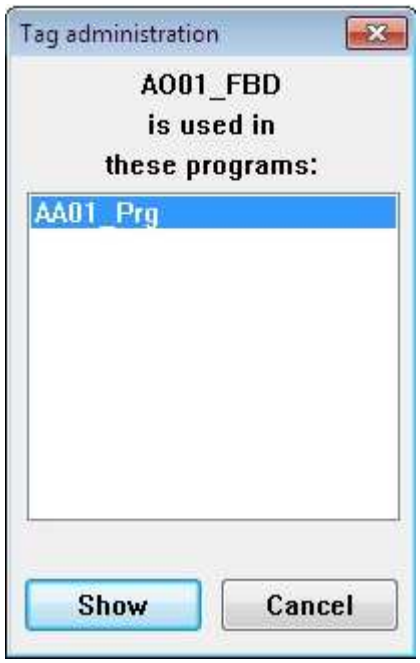
Longt text — Long text of the loop tag, max. 20 characters

Type name — Short names for the funktion modules, such as “M_ANA” for analog monitoring. Changes are possible with the permitted loop tag types via the selection windows

C — Module class

P
- # Module not plausibility checked
- @ Module plausibility checked

5.4.5 Change loop tag entries



If changes are made to existing loop tags, then these affect various programs. The programs concerned are listed when such changes are made, in order to prevent errors. It can then be decided in which programs the changes are to be effective.

→ select desired field with doubleclick (“Name”, “Short text”, “Long text”, “Type name, C”)

“Name” changes the name of the loop tag (conclude with →<Enter>)

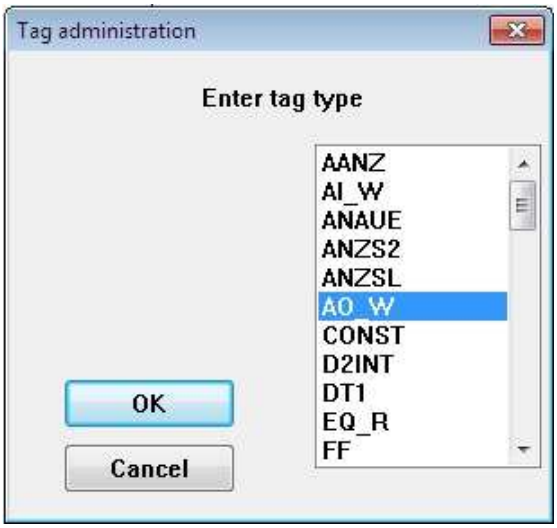
If the name is changed, a new window appears:

→ [Cancel] exits the mask without loop tag changes.
→ [Change] makes the changes in the selected program.
→ [Show] jumps directly to the selected program.
“Short text” changes the short text.
“Long text” changes the long text.
“Type name, C” shows a window with library types and their function modules.
→ [OK] (or →<Enter>) accepts the changed module.

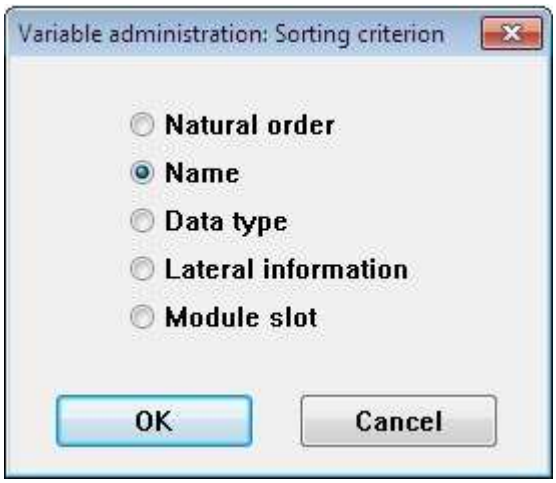
After confirmation, the programs concerned are listed (see above (this page)).

→ [Cancel] leaves without changes.

Note
The entry P cannot be changed within the loop tag list!



5.4.6 Sorting



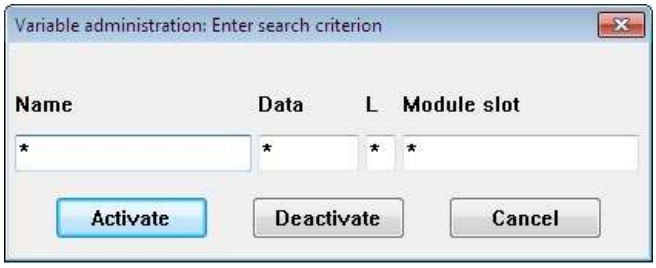
with
→Variables listor
→Loop tag list
→Sort
→select sorting criteria
→[Ok]
→(abort the action with mouseclick on)[Cancel]

The entries in the variables and loop tag lists are outputted on the screen according to the pre-selected sorting criterion.

Sorting criteria

- Natural sequence
  Sorting by entry sequence
- Name
  Sorting by alphabetical names
- Signal type
  Sorting by signal types
- Card slot
  Sorting by card slots (only ariables list)

5.4.7 Searching



with
→Variables list or
→Loop tag list
→Search
→Define
→Combine search criteria in one window
→Search
→Activate

or via keyboard:

→<Alt>
→<U>
→<D>
→Combine search criteria in one window
→<Alt>
→<U>
→<K>

If the menu item “Activate” has been provided with a tick (√), then this is an indication that the outputted list has not been processed completely and with the search criteria. In this case, the complete list is outputted and the tick (√) disappears if “activate” is again selected.

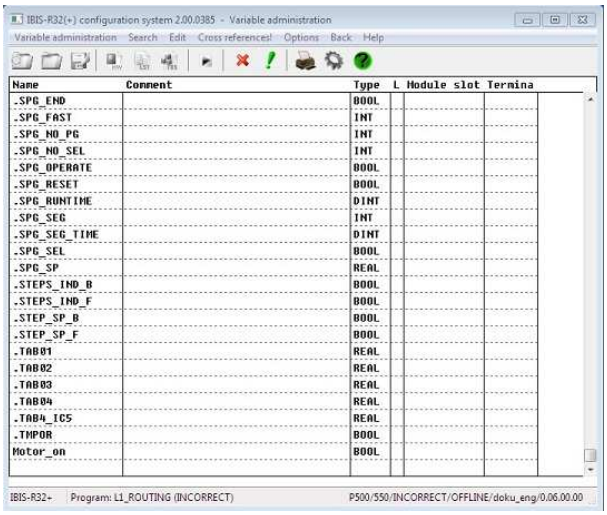
Entries can be searched for in the lists according to defined criteria, and displayed on the screen. A further window appears for this with a mask line which has the same structure as the associated list. Each column in the mask can be filled with a search text. Wild cards such as “*” (for several characters) and “?” (for any character) are permitted.

→ [Aktive] outputs the list according to search criteria.

→ [Deaktive] ignores search criteria, outputs the list completely

→ [Cancel] returns to the variables list.

5.4.8 Process list entries



with
→ Variables list or
→ Loop tag list
→ Process

A series of menu items is available for processing the individual list entries. Hence it is possible to cancel actions, insert new variables/loop tags and delete, cut, copy or insert lines.

Cancel

wiht
→ Process
→ Cancel or → <Shift + C>.

The last change is revoked and the former condition is retained. If no cancellation is possible, the menu item cannot be selected (display in grey).

Insert new variable into list

with
→ Process
→ Insert new variable.

After selection, the program moves to the first free entry in the variables list.

→ enter name
→ <ENTER>

A window then appears in which the data type can be chosen.

A comment can also then be entered.

Insert new loop tag into list

with
→Process
→Insert new loop tag.

If the cursor is on a vacant field (such as the end of the list), then the entry of a new loop tag is made directly into the individual fields of the list line. A window appears if the cursor is on an already-assigned list entry. The chosen name is present in this for information and as new entry. This new entry must then be changed to the desired new name.

Old name information.	The name of the loop tag chosen, as
New name	As default setting, the name of the selected loop tag which must now be re-inserted.
→[Insert]	accepts new loop tag.
→[Skipe]	(no funktion in the entry of a loop tag)

The loop tag entered must be given a new name in “Insert block”. It is possible to jump certain loop tags within the block with → [Skip] and not to include them in the list.

Provided that the cursor is on a free name field at the call-up, the interrogation of the module library (which is described under chapter “Changing loop tag entries, type name”) is performed in both cases via a further window.

The short and long texts are entered directly into the associated list columns.

Editing the field of the list

with
→select desired field with doubleclick (highlighted by field border)
The cursor is at the last entry position.
→click cursor on entry position within the field
→enter changes

The text content of the field selected can be amended. If required, the program will enquire after the change in a further window whether the change is to be effective in the entire project or only in specific programs (see “Changing loop tag list entries, loop tag names”).

A predefined global variable cannot be edited.

Delete field

If an entry of a list line is selected, then it is possible to delete that entry.

Certain entries into fields cannot be unambiguously deleted with this command. In the variables list, these are the fields “Name” and “Type”, and in the loop tags list, the fields “Name”, “B”, “Type name”, “C” and “P”.

→Process
→Deletefield
→select desired entry with <↑>, <←>, <↓>, <→> (highlighted by field border)
→<Del>

Text lines of a list entry are deleted directly with the cursor.

→click field
→position cursor at the beginning of the deletion area
→mark the desired deletion area by holding down the left mouse key
→<Del>

Block processing

Only one block can be defined at a time. It consists of a series of marked lines of the list and can be marked as follows:

→Click cursor on the desired start of block
→pull the cursor along to the end of block with left mouse key pressed
or via keyboard:
→to the desired start of block with <↑>,<←>,<↓>,<→>or<Tab>
→pull over the desired block area with <Shift + ↑>(+ ←>, + ↓>, + →>)

The block so created is labelled and is also retained when the left mouse key or <Shift> is released.

Cutting:

with
→Process
→Cut

or via keyboard:

→<Alt>
→<E>
→<C> or
→<Shift + Del>

A defined text block is removed from the text section and stored in the buffer file. This stored text can be employed again at any location, using the command “Insert”.

Copying:

with
→Process
→Copy

or via keyboard:

→<Alt>
→<E>
→<B> or
→<Ctrl + Ins>

A defined text block is copied and stored in the buffer file. This stored text can be employed again at any points, using the command “Insert”.

Inserting:

with
→Process
→Insert

or via keyboard:

→<Alt>
→<E>
→<I> or
→<Shift + Ins>

A copied or cut text block in the buffer file is inserted at the point indicated by the cursor.

Because the names of variables and loop tags have to be changed, the same window appears as in the menu item “Insert new variable” and “Insert new loop tag”.

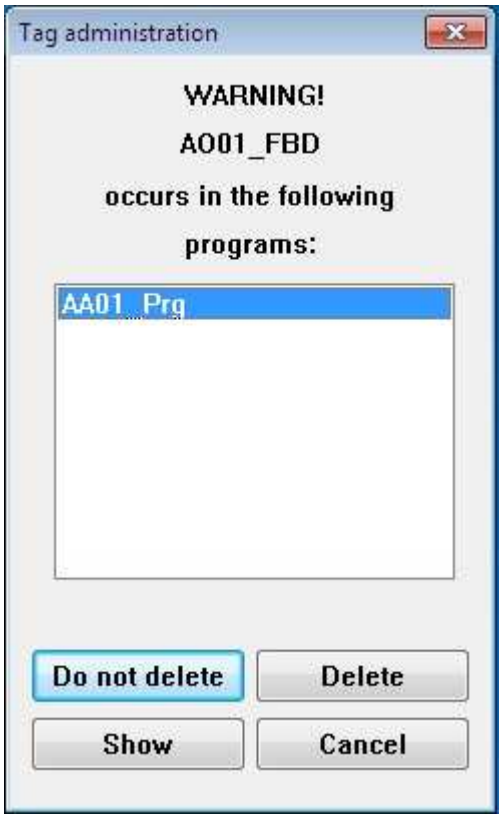
Deleting:

with
→Process
→Delete

or via keyboard:

→<Alt>
→<E>
→<D> or
→<Del>

Within a window, a defined text block is deleted from the text after interrogation.



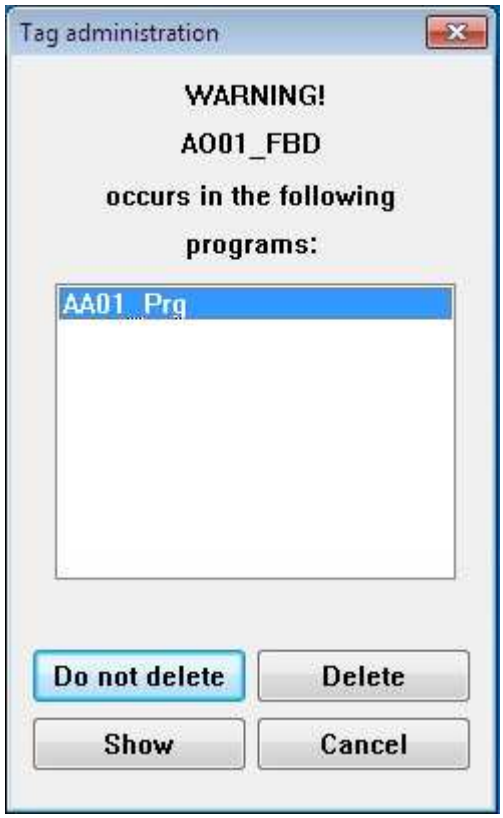
→[Do not delete] does not delete the selected variable/loop tag.

→[Delete] deletes the selected varable/loop tag.

→[Show] jumps to selected program

→[Cancel] returns to the corresponding list.

5.4.9 Cross references



with
→select field
→Crossreferences!

or via keyboard:

→selct field with <↑>,<←>,<↓>,<→>or<Tab>
→<Alt>
→<R>
A window shows the names of the programs concerned.

→select programwith doubleclick

or via keyboard:

→select program with <↑>,<←>,<↓>,<→> and <Enter>

→[Show] jumps into the selected program.

→[Cancel] closes the window in which the cross-references are displayed.

5.4.10 Ending variable/loop tag administration

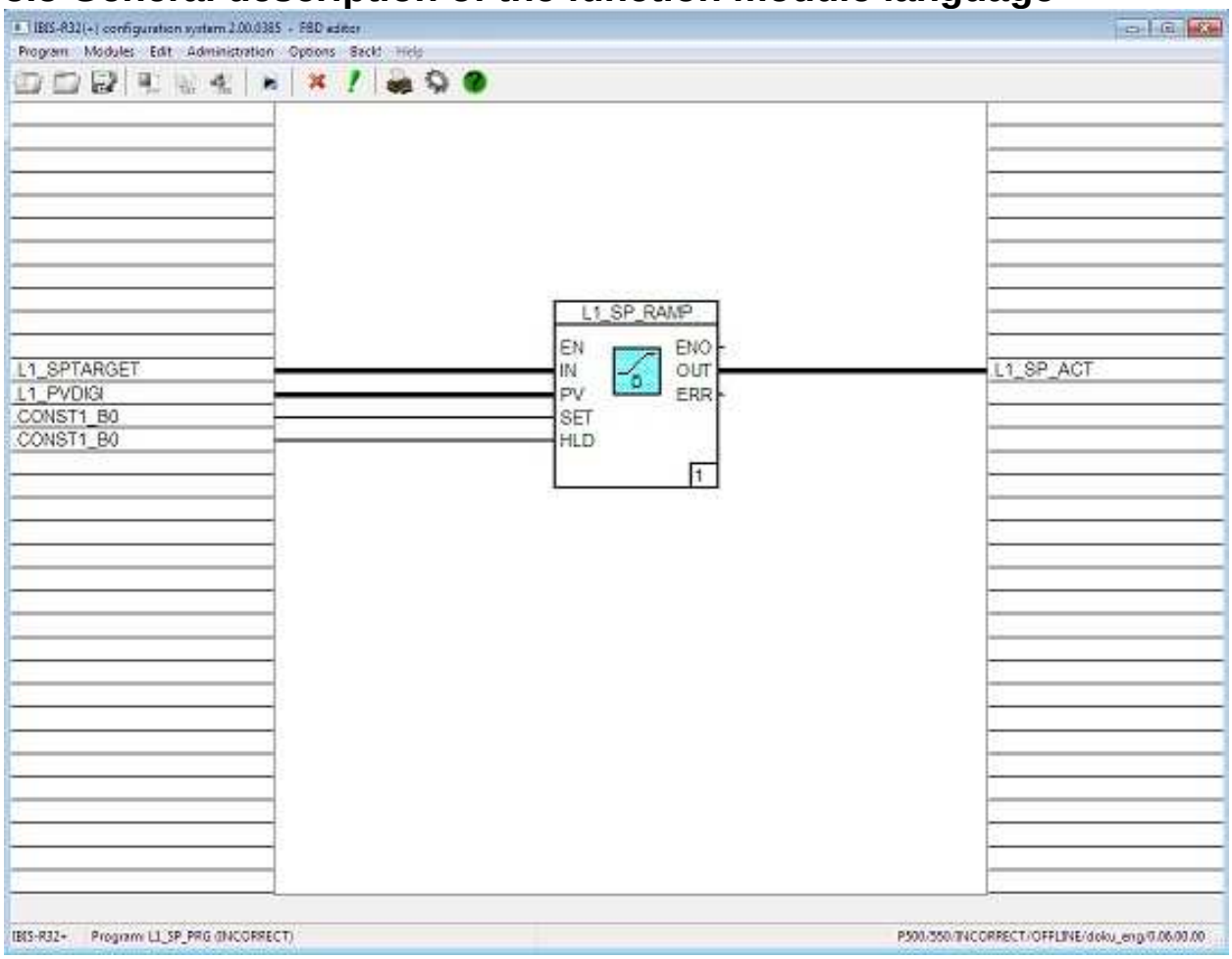
with
→Variableslistor
→Looptaglist
→Terminate The program returns to the project tree.

5.4.11 Return

with →Return

The program returns to where the variable/loop tag administration was called from.

5.5 General description of the function module language



A function module language program is a logical arrangement of elements of the function module language. Functions, function modules and program inputs and outputs are linked in it by signal flow lines.

Each program can also be implemented as an instruction list (IL).

The signal processing needed for the control of a machine or a process by an automated system is made possible by the programs.

The function module language editor is the system tool for creating and changing programs in the function module language (FBD).

Its CAD functional capabilities permit simple placement and connection of modules and variables. The operating range of a program corresponds to one screen page. This makes possible a program documentation which precisely reproduces what can be seen on the surface of the screen.

5.5.1 Definitions

Execution sequence
is defining the sequence of the execution of the individual modules within a function module language program. It is displayed bottom right in the modules.

Connections
are the inputs and outputs of the modules and the variables of the input and output strips.

Output variables
are always in the output strip of the program. They produce the connection to modules of the device, and to the module inputs of other programs.

Moduls
The term “module” is used in the documentation where it has not been stipulated whether “function” and/or “function module” is meant.

Data sources
are the variables of the input strip and the module outputs.

Data sinks
are the variables of the output strip and the module inputs.

Input variables
are always in the input strip of the program. They produce the connection to the modules from the device to the module inputs of other programs.

Functions
deliver exactly one data element (result). Functions do not contain any status information. They do not store any information for the next cyclic processing. A call-up of a function with the same argu-ments (input parameters) therefore always delivers the same result (output parameters).

Function modules
deliver one or several data elements (results). Several copies of a function module can be produced and distinguished by module names. Parameters of each copy can be defined separately and all values of the output variables as well as the necessary internal variables are retained from one proces sing to the next. Each function module therefore contains status information. A call-up of a function module with the same arguments (input parameters) does not therefore always deliver the same result (output parameters).

Function module language
is a graphical programming language for describing control function capabilities. The main elements of the function module language are graphical displays of modules and signals as module diagrams (function modules and functions) and signal flow lines. The language enables all modules to be displayed arbitrarily linked.

Flag variables
are input and output variables which connect inputs and outputs of modules of different programs to one another. Variables become flag variables automatically if they are used for connecting modules of different programs or on a program as input and output variable..

Program elements
are the smallest display units of the program, hence functions and function modules, signal flow lines and sometimes their sections as well as variables of the input and output strips.

Signal flow
is always from the data source to the data sink.

Signal flow lines
are the graphical representation of the signal flow in the program.

5.5.2 Creating a function module language program

A function module language program is created via the project tree (see 5.3.7 “Importing a program”).

with
→Project tree
→select insertion position in the project tree
→Process
→InsertAbove, Insertbelowor Insertson(only if no son available)
→Function module language program from “Object selection”
→assign program names and perhaps short comment

Each new function module language program has vacant input and output strips, an empty graphics area and program comment, the processing status “correct” and the generation date as version identification.

The name of the program list (PL) is preset as program name, which is to be assigned to the program. The same applies to the short comment.

5.5.3 Calling the editor

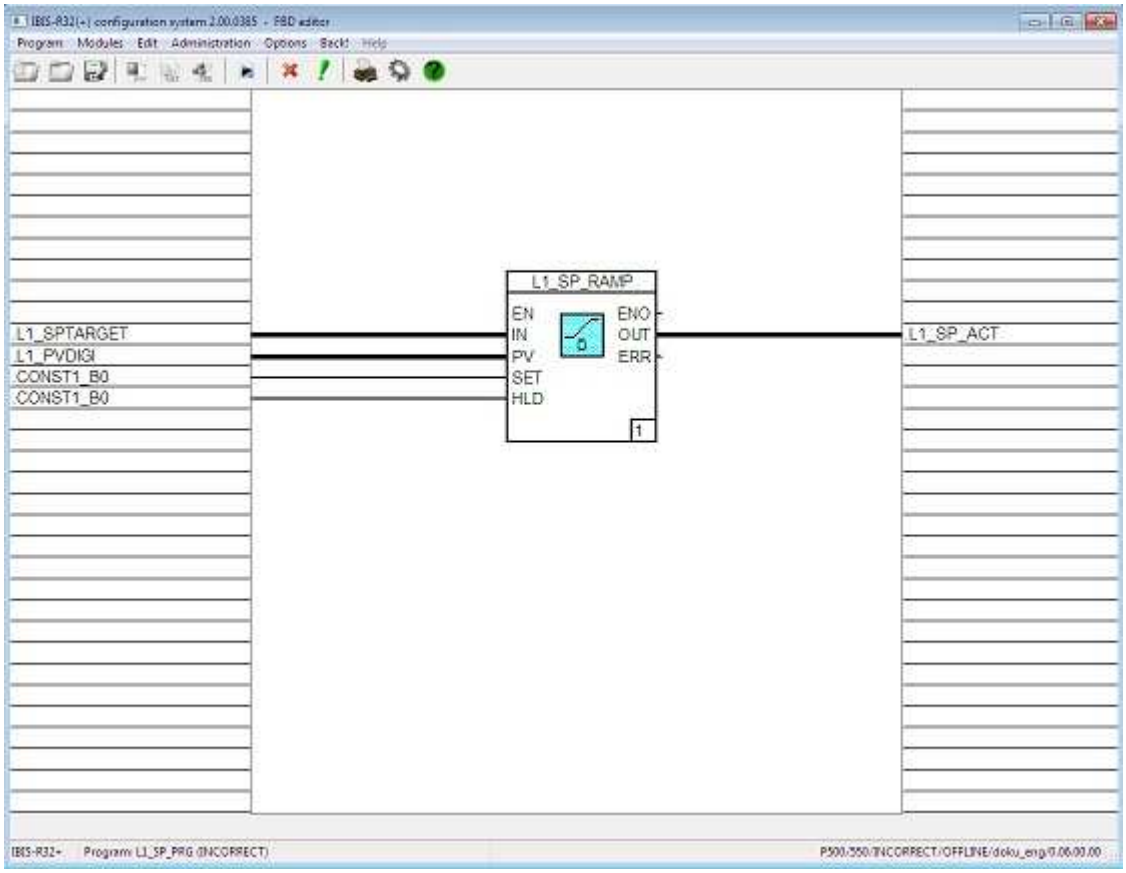
Note
Assumption: there must be a program.

→Project tree
→Process
→Program or →doubleclick on program

The editor is called and the program is selected and displayed with contents (funktions, signsl flow lines...). Its configuration can then be changed.

5.5.4 Interface of the editor

Structure of the configuration interface



The configuration interface of the editor is divided up into the following segments:
− top: menu strip
− bottom: status line
− left: input strip
− centre: graphics area
− right: output strip

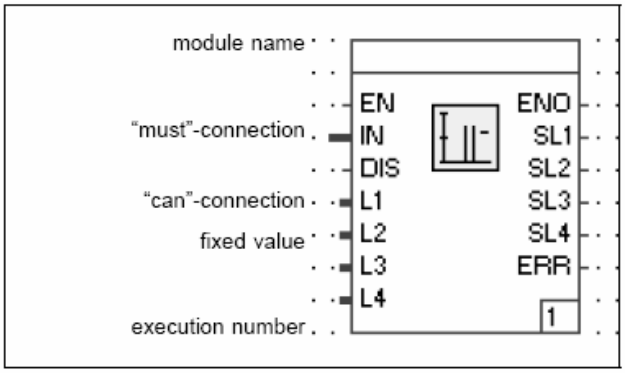
Input strip
contains the input variables of a function module language program.

Output strip
contains the output variables of a function module language program.

Status line
display of the instantaneous program status: correct/incorrect

Graphics area
The modules and signal flow lines are programmed in the graphics area of the function module language program. The graphics area has a grid to permit simple positioning of the elements whilst observing minimum distances. The user is only able to lay corners of modules and signal flow lines in this grid. The connections to the input and output strips and the connections of inputs and outputs of modules are automatically laid in this grid by the editor. The grid can be switched on and off.

Display of the modules



Frame

delimits the selection area of the module. It can be recognized from its colour whether or not the module has been selected or if its parameters have (not) been correctly defined. The colour display for this can be changed (see 5.3.4 "Options of the project tree: colours").

Module name

Unlike functions, all function modules are displayed with a module name (max. 12 characters). It is recommended to chose the loop tag name as module name. All module names are found again in the system-wide loop tag administration. The colour of the lettering of the module name is used to identify the processing status (enable/disable) and can also be set.

Icons

With function modules, the module type is symbolised by an icon, and with functions by a function abbreviation.

Connections

A distinction must be made here between inputs and outputs. Corresponding to the signal flow, inputs are always displayed left and outputs always right. There are also "must" and "can" connections. "Must" connections require supply via a signal flow line, to enable the module to operate correctly, "can" connections do not. To distinguish between them, "can" connections are displayed shorter. Certain "can" connections are omitted entirely due to the parameter assignment of fixed values. As with the signal flow lines, the colour and line width provide information on the required/set signal type.

The connections are displayed in different colours and widths, according to the signal type:

Signal type/ Processing status	Colour	Display
BOOL	black	narrow
DINT	dark green	wide
INT	light green	wide
REAL	black	wide

Connection name

As well as every connection of a function module, an abbreviation also quotes the function of that connection, for example EN for "enable".

Execution number

The code figures bottom right in the modules indicate the execution sequence within the program.

5.5.5 Changing the presettings

Superimposing and blanking the grid in graphics area

with
→Options
→Grid ON/OFF
(→Save!)

The positioning grid in the graphics area is superimposed if it was previously blanked and vice versa.

The change of the setting is retained until another window is opened. If the setting is to remain in force beyond this, the program must be saved after changing the grid.

The saved setting of the last program processed is preset. The grid is superimposed for the first program of a new project to be created.

The grid spacing is not changeable.

Changing the colour display of the editor

with
→Options
→Colours
→select object for which the colour is to be changed (for example the colour of the module background)
→select desired colour
→[OK]

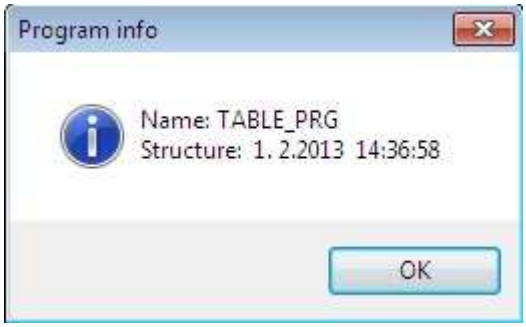
or via keyboard:

→<Alt>
→<O>
→<C>
→ select the display colour to be changed with <↑>,<←>,<↓>,<→>
→<Enter>
→set the desired colour with <↑>,<←>,<↓>,<→>
→<Enter>
→<Esc>

The colour display of the modules, signal flow lines and their statuses are changed.

5.5.6 Displays of program information

Version of the program and assignment to the project



with→Options→INFO.

The program name, the date of the last program change (version) and the assignment of the program to the project, the resource and the program list are displayed.

The program assignment can be displayed as long text or short text.

Status of the program

The status line shows the name of the currently processed program and shows whether it is correct or incorrect.

Correct
The status “correct” is only granted if a plausibility check of the program has been performed and no errors have occurred in it.

Incorrect
Every entry in the program first leads to the status “incorrect”. Each newly-created but not yet processed program is likewise given the status “incorrect”.

5.5.7 Procedure for creating a function module language program

1. Define the insertion position in a program list in the project tree

2. Assign names to the function module language programs, and create program header and comment if required.

3. Call up configuration: function module language (FBD).

4. Select and position module(s) and define the number of inputs of the module if required.

5. Connect modules and variables with signal flow lines, enter tree. variables in the input and output strips (enter directly or select via F2). Change the execution sequence of the modules if required.

6. Define parameters of the module as far as possible, but at least assign module names (loop tag names).

5.6 Modules

5.6.1 General

Data types of the module inputs and outputs

The data types for the module inputs and outputs are preset. The data types can be changed with certain modules. The changing and definition of data types is performed in the configuration dialogue under ???“Change signal type”???. See ???function module language editor???.

Overview of data types:

Data type	Bit	Value range	Explanation	Input formats Examples
REAL	32	±1.17549435E-38 ...±3.4028236E38	Floating-point value IEEE format	0.0, 3.14159, -1.34E-12, -1.2234E-6, 12.6789E10
DINT	32	-2 147 483 647 ...+2 147 483 647	Doubled integer value with sign	355, +23456
INT	16	-32 767 ...+32 767	Integer value with sign	-3, 12345
BOOL	16	FALSE, TRUE, 0, 1	Boolean value	FALSE TRUE 0 1

Tab. 1 Overview of data types

Notes on defining parameters for the modules

Specific parameters (name, start of scale etc.) must or can be entered with different functions and function modules.

Parameter definition for the module is performed in the configuration, using the function module language or instruction list.

The so-called “must” parameters are indicated by color (red) and must be filled in. The mode of operation and the parameters of a function must be defined with the other entries.

All parameter names in these parameter definition masks are displayed in italics. The handling of the functions and function modules is similar, so repetitive handlings, entries and markings are only described here once.

Name

The name is unambiguous within a project. It is absolutely essential to enter it. It can be up to 12 characters long and must start with an alphanumeric character.

Short text

Up to 12 characters, all characters permitted.

Long text

Up to 30 characters, all characters permitted.

Processing

☒ By marking the processing box with a cross, the module after loading is immediately included in the processing, and processed. The module input **EN** (enable, see below) is not given a connection pin.

☐ The function is not processed without the cross. The module input **EN** is not given a connection pin.

■ A filled-in box indicates that the processing is performed according to the input **EN**. The connection **EN** is displayed on the module and must be connected. The module is processed with a TRUE input signal.

Sequence

Define execution sequence (1...99) within the user program. If no input is made then the sequence of execution is that of the module placing. In a subsequent change of the sequence of a function, the normal entries are correspondingly changed as far as necessary.

Input/output pins

Each function module has an input **EN** (enable). With TRUE, the function module is processed; with FALSE, the function module is non-operational (disabled) and all outputs are at 0. An “enable” signal switches the function module back on and initialises it. The module status is made available at the output **ENO** (enable out) 0 = disable, 1 = enable.

Various function modules have the output **ERR**. A TRUE at this output indicates that the function module cannot operate error free in its parameter assignment with the input values given (eg. division by 0). In this case the output values still are only conditionally valid.

The output **STA** delivers a status identification for identifying the error occurring. Without errors, the output is at 0.

Repeating keys

→[Quit] exits the active parameter definition window and saves the parameter status

→[Cancel] exits the active parameter definition window without saving the parameter status. A warning appears if parameter definition data could be lost.

→[Save] saves the instantaneous parameter status and keeps the window active

→[Reset] completlely reset the parameter definition to the values. A previosly saved parameter definition differing from the default can be recalled by aborting and recalling the parameter definition display.

→[Plausib. check] checks plausibility of the function module with the instantaneous but perhaps also unsaved parameter definition

→[<<],→[>>] changes info the previous or nexxt parameter definition display.

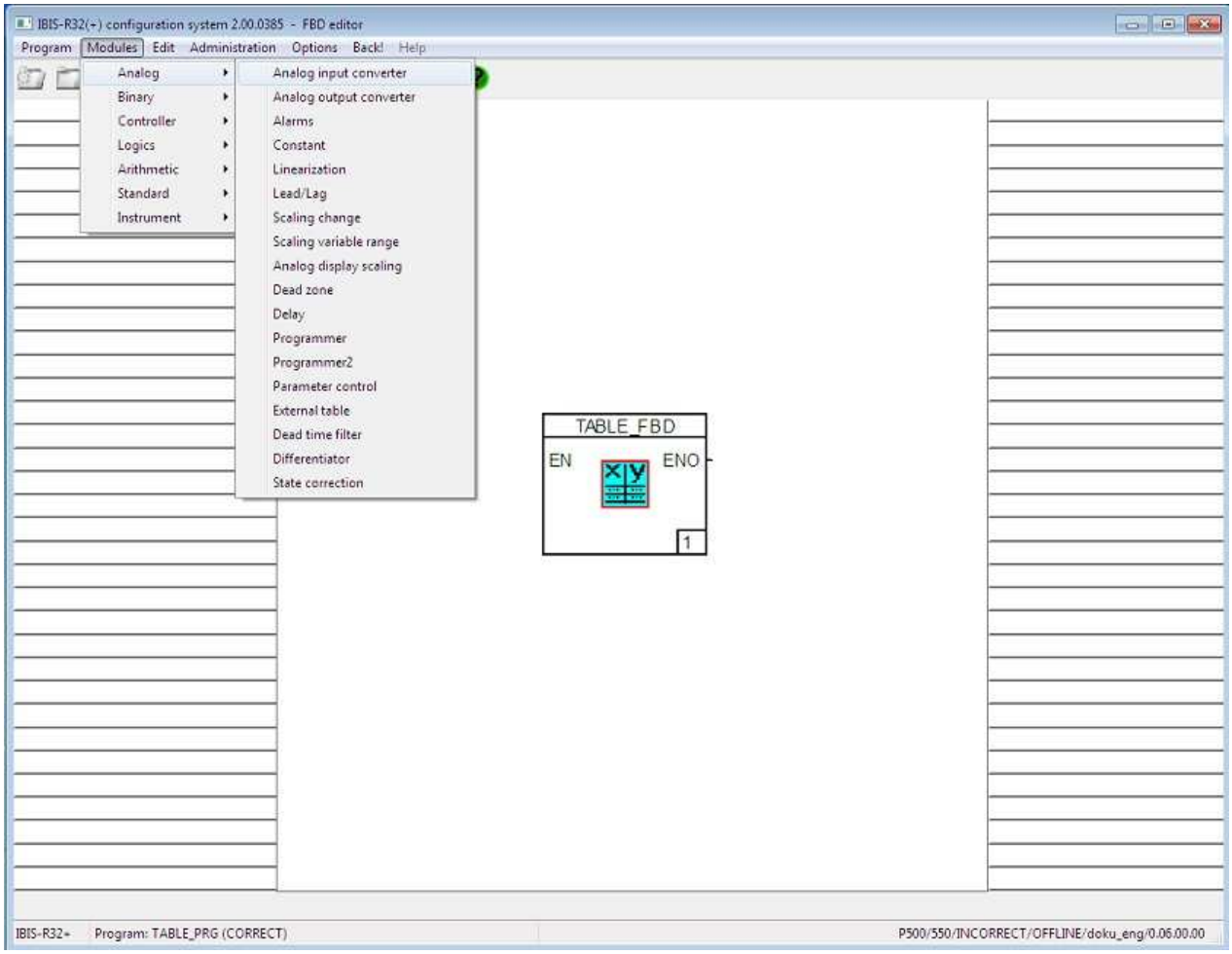
5.6.2 Overview of the function modules

Modules	Funktion/Function module	Short name
Analog	Analog input – converter	AI_W
	Analog output - converter	AO_W
	Analog monitoring	ANAUE
	Constant	COONST
	Lineraization	LIN
	Lead/lag	LD_LG
	Scaling change	SKAL
	Analog display – scaling	AANZ
	Dead zone	TZ
	Delay	PT1
	Programmer	PG
	Paramater control	PARA
	External table	TAB
	Dead-time filter	TOTZ
	Differenziator	DT1
	State correktor	ZKOR
	Alarm values	GW4
Binary	Monoflop	MONOF
	Timer, delayed switch-on	TVE
	Timer, delayed switch-off	TVA
	Timer, time out	TTE
	Forwards/backwards counter	VRZ
	Impulsgeber	PULSE
Logic	Comperator = with tolerance	EQ_R
	Comperator ≥ with differential gap	GE_R
	Comperator > with differential gap	GT_R
	Comperator ≤ with differential gap	LE_R
	Comperator < with differential gap	LT_R
	Comperator <> with differential gap	NE_R
	RS flipflop	FF
	Falling edge detection	FTRIG
	Rising edge detection	RTRIG
	Trigger	TR
Standard logic	And	AND
	Or	OR
	Exclusive or	OR
	Not	NOT
Standard slection	Minimum	MIN
	Maximum	MAX
	Average value	AVG
Standart comperators	Comperator =	EQ
	Comperator ≥	GE
	Comperator >	GT
	Comperator ≤	LT
	Comperator <	LE
	Comperator <>	NE
Standartd switches	Binary criterion	SEL
	Multiplexer	MUX
Standard converters	*_**TO**_*	TO
	DINT_TO_REAL	
	DINT_TO_INT	
	INT_TO_DINT	
	REAL_TO_DINT	
	Trunc	TRUNC
	REAL_TO_DINT	
	REAL_TO_INT	

Modules	Funktion/Funktion module	Short name
Arithmetic: basic aritmetics	Addition Multiplication Subtration Division Modulo Exponentiation	ADD MUL SUB DIV MOD POT
Arithmetic: numerical aritmetics	Absolute value Square root Sign C level	ABS SQRT SGN CPG
Arithmetic: logarithms	Natural logaritm Decimal logaritm Exponent	LN LOG EXP
Arithmetic: modules	Average value with monitoring Average value of three	MW_UE MW_3
Controllers	PID universal controllers Operating mode selectors	PID REGBA
Model	Display loop Keyboard Event	ANZSL KEYB MELD

Tab. 2

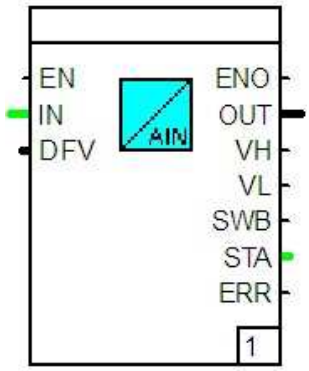
5.6.3 Modules, analog



Overview of modules, analog

AI_W	Analog input converter
AO_W	Analog output converter
ANAUE	Analog monitoring
CONST	Constant
LIN	Linearization
LD_LG	Lead/Lag
SKAL	Scaling change
AANZ	Analog display scaling
TZ	Dead zone
PT1	Delay
PG	Programmer
PARA	Parameter control
TAB	External table
TOTZ	Deadtime filter
DT1	Differentiator
ZKOR	State correction

Analog input converter, AI_W



Function:

The function module calculates an output signal in the range 0....20000 from an AD converter value at the input **IN**. This signal available at the output **OUT** lies in the range 0.0 ... 1.0. The value can only move out of this range under detected erroneous behaviour.

Cable break can be detected with the signals 4..20 mA and 2.. 10 V.

If the analog input signal lies outside of the permitted value range, the function module switches to the default value and the associated alarm signal (**VL**, **VH**) is set.

If a sensor break is detected, the signal **SWB** is set.

With the outputs **VL**, **VH** and **SWB** set in the event of fault, the output **ERR** is also set.

Cable break can not be detected with the signals 0..20 mA und 0..10 V.

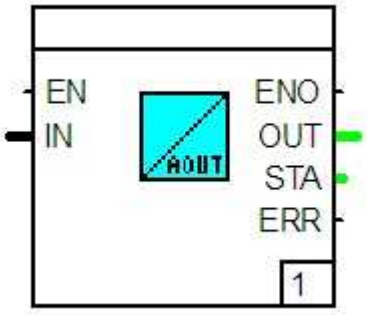
Parameter definition:

Signal type:
0..20mA
A 0..20 mA signal is used as signal source.
4..20mA
A 4..20 mA signal is ...
0..10V
A 0..10 V signal is ...
Pt100
A Pt100 is ...
Thermocouple
A thermoecouple is .... The thermocouple type is set in the function block “Linearization”.
Teletransmitter150 Ω
A resistance teletransmitter of 150 is...
Teletransmitter1500 Ω
A resistance teletransmitter of 1500 Ω is ...

Strategy for sensor faults:
Outputdefaultvalue
With erroneous behaviour in the input, the inputtable value in % is made available as output value. If no value is inputted, the value of the input **DFV** is used.
Retain last value
With erroneous behaviour in the input, the last non-errored value is made available as output value.

Dimension:
A dimension can be assigned to the output signal.

Analog output converter, AO_W



Function:

The function module converts a percentage numerical value from the input **IN** into a signal value for a DA converter.

It is possible to set the output range of the signal **OUT** for the DA converter. The input signal can still be limited by upward and downward limits before the conversion.

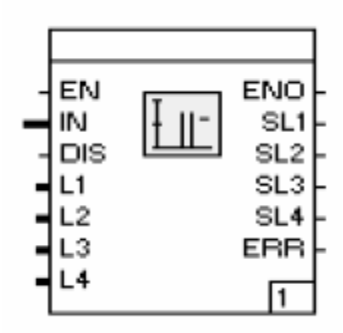
The following error stati are outputted at the output **STA**:
0 No errors.
1 Input value is smaller than minimum output value.
2 Input value is larger than maximum output value.
The output **ERR** is set with an error status not equal to 0.

Parameter definition:

Limiting values:
Minimum output in%
Lower limiting value for the output signal.
Maximum output in %
Upper limiting value for the output signal.

Signal type:
0..20mA
A 0..20 mA signal is produced at the output.
4..20mA
A 4..20 mA signal is ...
0..10V
A 0..10 V signal is ...
10mA
A constant 10 mA or 5 V signal is ...
20mA
A constant 20 mA or 10 V signal is ...

Analog monitoring, ANAUE



not in library 3.6.0

Function:

The function module monitors an analog signal **IN** for up to 4 alarm limits. The 4 alarm values can be entered as input signals **L1** to **L4**, or as constant values. It is also possible to use a percentage part of the input signal as alarm limit. The infringement of an alarm value is available at the outputs **SL1** to **SL4** as TRUE signal.

Alarm limit checking can be switched off via the input **DIS.** The outputs retain their value.

The output **ERR** is not currently set.

Parameter definition:

Alarm limits:

No.
Number of the limit value.

Type
( ) Limit value is not used
MIN Signal is checked for overshooting the limit
MAX ... for undershooting the limit value
XW-MIN Signal is checked for overshooting the limit value as control deviation
XW-MAX ... for undershooting the limit value as control deviation
d/dt[s] Signal is checked for overshooting the limit value in respect of its rate of change.

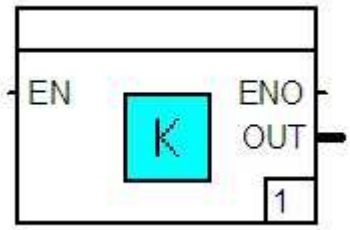
Magnitude |IN |
Before limit value checking, the magnitude of the signal is formed and used for checking.

Alarm limit
Limit value which is to be checked. If a value is defined here, the associated input cannot be used with a signal as limit value. If no value is defined, the corresponding input signal is used as limit value

x%
Percentage weighting of the limit value. The actual limit value is formed from limit value or input signal multiplied by this percentage weighting. This is normally 100.0, so the limit value itself is effective.

Hysteresis[%]of limit value
Associated hysteresis in % of the limit value for the cancel lation of the alarm signalling.

Constant, CONST



Function

The function module makes a parameter-definable value of the type REAL aviable, as signal at the output **OUT**

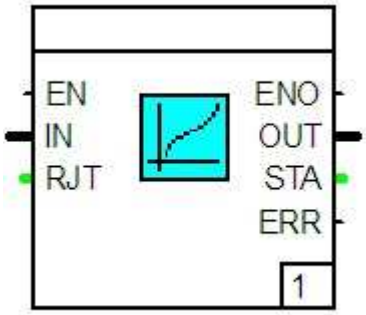
It is possible to switch the output between the value set in parameter definition and 0.0 via the input **EN**

If only one unchangeable constant is requred as signal, this can module language programm

Parameter definition:

Value of constant
Value outputted at the output

Linearization, LIN



Function:

The function module converts an input signal **IN** from the nume rical range 0.0 to 1.0 into an output signal **OUT** in another para-meter-definable numerical range.

Specific non-linear linearization functions can be quoted in this. The reference junction measurement can be injected as direct signal of the AD converter for thermocouple linearizations.

The following error stati are outputted at the output **STA**:
0 No errors.
1 No characteristic defined.
2 Division by 0.0 has occurred.
3 The input value undershoots the lower range-value by more than 2 %.
4 The input value overshoots the upper range-value by more than 2 %.
5 Fewer than 2 coincidence points with Y values are quoted in the module.
6 The module cannot access a table in the internal processing.
The output **ERR** is set if an error status does not equal 0.

Parameter definition:

Measuring range:
Measuring range start
Lower limit of the output scaling.
Measuring range end
Upper limit ...

Characteristic:
Linear
Linearization is performed using a straight-line equation.

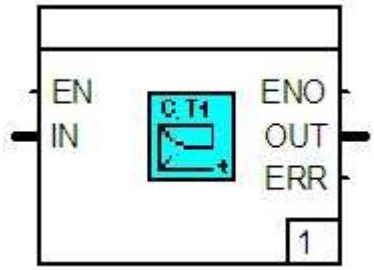
Table int.
Linearization is performed using the parameter-definable table with Y values.
Table ext.1
Linearization is performed using the first table stored in the instrument.
...
Table ext. 4
... the fourth table ...

Squareroot extraction ≥X0
The square root extraction function is used as linearization. The signal is only used from the value X0 (assignable in %).
Squareroot extractionlin.<X0
The square root extraction function is used as linearization. The signal is converted linearly below the value X0 (assign able in %).
Squareroot extractionX0in%
Assignable value in the square root extraction.
Type“L”-200..1000℃
Linearization is performed in the quoted measuring range using the linearization table for thermocouple type “L” stored in the module. The associated temperature values must be entered as lower and upper range values.
Type“J”-200..1200℃
As type “L”, only with regard to thermocouple type “J”.
Type“K”-200..1400℃
As type “L”, only with regard to thermocouple type “K”.
Type“U”-200..600℃
As type “L”, only with regard to thermocouple type “U”.
Type“R”0..1700℃
As type “L”, only with regard to thermocouple type “R”.
Type“S”0..1800℃
As type “L”, only with regard to thermocouple type “S”.
Type“T”-200..400℃
As type “L”, only with regard to thermocouple type “T”.
Type“B”0..1800℃
As type “L”, only with regard to thermocouple type “B”.
Type“D”0..2300℃
As type “L”, only with regard to thermocouple type “D”.
Type“E”-200..1000℃
As type “L”, only with regard to thermocouple type “E”.
Pt100-200..200℃
Linearization is performed in the quoted measuring range for resistance thermometer Pt100 using the linearization table stored in the module. The associated temperature values must be entered as lower and upper limits of the measuring range.
Pt1000..450℃
As Pt100 -200..200℃ only in the range 0..450℃.
Pt100-200..800℃
As Pt100 -200..200℃ only in the range -200..800℃.

Reference junction compensation with thermocouple:
internal
The internal reference junction compensation is used in linearisation for thermocouple.
none
No reference junction compensation is used in linearisation for thermocouple.
ext.0℃
The measured signal has already been compensated externally at 0℃ in linearization for thermocouple.
ext.20℃
... at 20 ℃ ...
ext.50℃
... at 50 ℃ ...
ext.60℃
... at 60 ℃ ...

Y Values:
Referencevalue0 atX=0%
First reference value of the internal table for the linearization of the input value at 0 %. Interpolation between the checkpoints is performed linearly.
...
Referencevalue10atX=0%
Tenth reference value ...

Lead/Lag, LD_LG



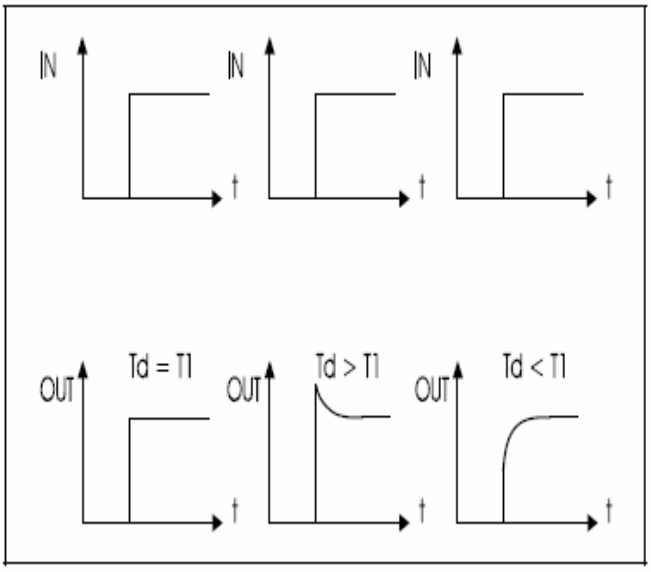
Function:

This function module constitutes a special function of a delay circuit. It is used for example in dynamic disturbance variable feedforward (to improve the process behaviour) or in setpoint value filtering (to prevent an overshoot in sudden setpoint changes).

A different response is obtained to an sudden change at input **IN** depending on the parameters “derivative action time” (Td) and “delay time” (T1). If the same times are chosen for Td and T1, the output responds with a step change of the same level as appeared at the input.

If the parameters differ, then the output signal first changes suddenly to a value which is proportional to the ratio of Td/T1. The output then reaches steady-state condition with a first order transition function, which is of the same level as the inducing sudden change.

The diagram illustrates how this works:



Parameter definition:

The working equation runs:

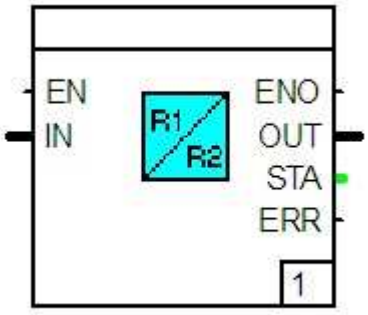
$$AUS_n = AUS_{n-1} + \frac{tz\,(EIN_{n-1} - AUS_{n-1}) + Td\,(EIN_n - EIN_{n-1})}{tz + T1}$$

AUS_n Output at the cycle n
AUS_{n-1} Output at the cycle n-1
EIN_n Input at the cycle n
EIN_{n-1} Input at the cycle n-1
Td Derivative action time
T1 Delay time
tz Cycle time

Derivative action time
in TIME format e.g. T#4m2s.

Delay time
Time constant T1 in TIME format, must be more than 0 seconds.

Scaling change, SKAL



Function:

The function module displays an analog signal **IN** of the REAL type in another numerical range, and makes this available as a signal at the output **OUT**. For this display, one pair of values each must be quoted for the input range and output range. If the input value lies outside the measuring range of the input, it is possible to define whether this value is restricted to the limits or is also to be converted outside. The output value then lies outside of the parameter definable measuring range.

The equation for the conversion runs:

$$Output = \frac{Input - MEE}{MEE - MEA} * (MAE - MAA) + MAE$$

MAE	Upper measuring range limit, output
MAA	Lower measuring range limit, output
MEE	Upper measuring range limit, input
MEA	Lower measuring range limit, input

The lower limits of measuring range at the input and output must be smaller than the upper limits of measuring range.

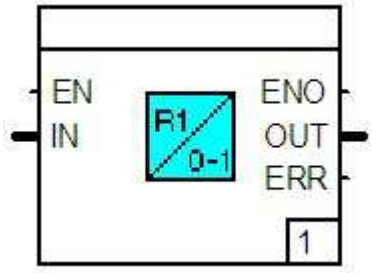
The following error stati are outputted at the output **STA**:
0 No errors.
1 The input value violates the input measuring range.
2 Division by 0.0 has occurred.
The output **ERR** is set if the error status does not equal 0.

Parameter definition:

Measuring range input:
Measuring range start
Lower value of the input signal.
Measuring range end
Upper value ...
Limit
Limiting of the input signal to the measuring range is employed.

Measuring range output:
Measuring range start
Lower value of the output signal.
Measuring range end
Upper value ...

Analog display scaling, AANZ



Function:

The function module displays an analog signal **IN** of the type REAL in the numerical range -100.0 .. 100.0 and makes this value available at the output **OUT** as signal.

A pair of values must be quoted for the positive input range for this display. Only the span (upper to lower range limit) is important in this, not the absolute values.

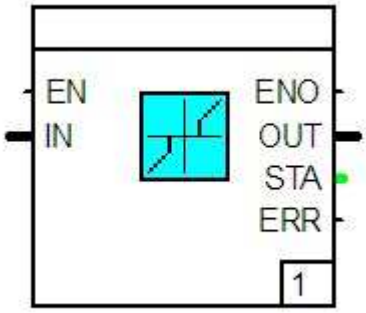
Input values in the range of the negative span up to 0.0 are displayed on the numerical range -100.0 to 0.0, and values in the range of the positive span from 0.0 are displayed on the numerical range 0.0 to 100.0. If the input value lies outside of the range of the span, then this value is limited to -100.0 or 100.0. The output **ERR** is set in this case.

The lower range limit must be greater than the upper range limit at both the input and the output.

Parameter definition:

Measuring range:
Measuring range start
Lower value of the input span.
Measuring range end
Upper value ...

Dead zone, TZ



Function:

The function module cuts out a parameter-definable part area in a bipolar signal at the input **IN.**

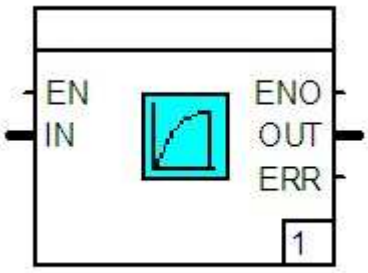
This part area is symmetrical about the Y axis. If the magnitude of the input value is smaller than half of the dead zone, then the value 0.0 is outputted at the output **OUT**, otherwise the value of **IN**.

The outputs **STA** and **ERR** are not currently set.

Parameter definition:

Width of dead zone
Below the half of this value the value 0.0 is outputted instead of the input signal.

Delay, PT1



Function:

The function module provides a first order delay.

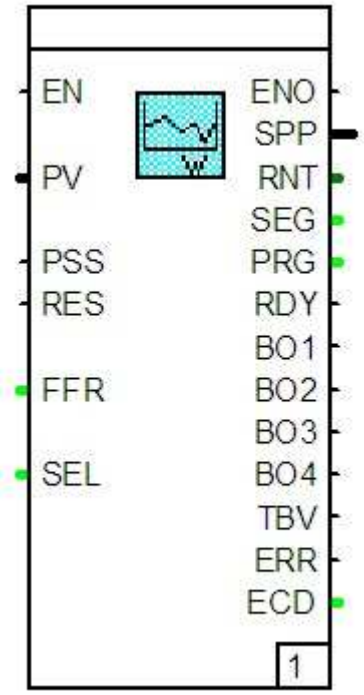
It acts on the input signal as smoothing low-pass filter, the time constant of which can be parameter defined. It is not permitted to enter a delay time of 0.

The output ERR is not currently set.

Parameter definition:

Delay time
in TIME format e.g. T#4m2s.

Programmer, PG



Function:

The function module provides a programmer for definition of setpoint curves (programs).

Up to 10 programs can be defined.

The setpoint is available at the output **SPP**.

The programmer can be started via the input **PSS** by a TRUE signal. The input **SEL** is evaluated at the start. This input specifies the program 1..10 to be used. If a fixed program has been defined in the parameter definition, then this input cannot be interconnected.

Provided that the programmer has been halted, the setpoint curve can be reset to its start via a TRUE signal at **RES**. If the programmer is halted, it can be switched to fast advance by the value 1 at the input **FFR** and with the value 2 to fast return. The setpoint curve remains at the particular point with the value 0.

Because the setpoint ramps can be stopped as a function of the controlled variable, these can be connected to the input **PV**. If this function is activated, this is communicated by setting the output **TBV**.

The total running time of the programmer is outputted in milliseconds at the output **TL** with the number of the selected program at the output **PRG**. The program segment just used is outputted at the output **SEG**.

If a program has been completely executed, this is identified by a TRUE signal at the output **RDY**. A binary channel for up to 4 binary signals is assignable to each program segment. These binary signals are available at the outputs **BO1** to **BO4**.

The outputs **FCD** and **ERR** are not currently set.

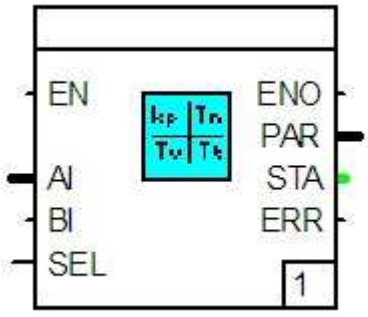
Parameter definition:

Program1
...
Program10
Pressing one of the buttons selects the respective parameter definition mask of the program.

Selected program
Entry of a setpoint curve not definable via input.

The enterable values correspond precisely to the enterable replies in the list configuration and online parameters of the particular program.

Parameter control, PARA



Function:

The function module illustrates a control module for one of the PID controller parameters.

The value provided at the output **PAR** can be outputted as a fixed value, as a controlled value dependent on an analog signal **AI** or binary signal **BI.** It is also possible to change between two different parameter sets via the input **SEL.** If this signal is connected, it is not possible to disable parameter set 2 in the parameter definition.

Transfer function:
Parameter 1fixed
Constant parameter value which is outputted for parameter set 1.
Parameter 1start
Lower range-limit in parameter control for parameter set 1. Parameter 1end
Upper range-limit in parameter control for parameter set 1.
Parameter 2 fixed
Constant parameter value which is outputted for parameter set 2.
Parameter 2 start
Lower range-limit in parameter control for parameter set 1. Parameter 2 end
Upper range-limit in parameter control for parameter set 2.
U start
Value which is used as lower range-value of **AE** for the conversion to parameter 1/2 start.
U end
Value which is used as upper range-value of **AE** for the conversion to parameter 1/2 start.

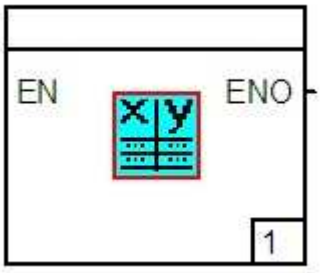
Parameter definition:

Input signal:
Fixed value
Only the parameters 1/2 are used permanently in transfer function.
AE
The parameter control is used via an analog signal. The input signal is converted from the range U lower value and U upper value to the range parameters 1/2 lower and upper values.
|AE|
The parameter control is used via the magnitude of an analog signal. The input signal is converted as with AE.
BE
A parameter switch-over is used via a binary signal. The parameters 1/2 lower value (**BE**=0) and upper value (**BE**=1) are used as outputted values.

Transformation:
Linear
The parameter control of AE and |AE| is performed via a straight line equation.
InternalTab.
The internal table is implemented via the Parameter "para-meter-definable table".
Tableext.1
The parameter control of AE and |AE| is performed via the first table stored in the instrument.
...
Tableext.4
.. the fourth ...
Use parameter set 2
[ ] The parameter set 2 is not used
[X] The parameter set 2 is always used.
[ ] The parameter sets 1 (**SEL**=0) and 2 (**SEL**=1) are used dependently on **SEL**

Parameters:
Ustart0%
First reference value of the internal table for converting the parameters for the value at 0 %. Interpolation between the checkpoint values is performed linearly.
...
Uend100%
10th reference value of the internal table.

External table, TAB



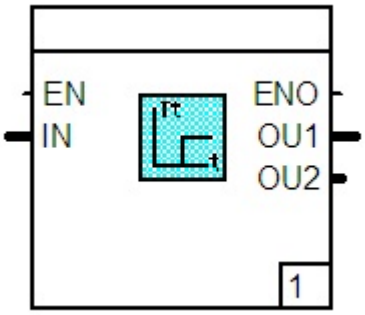
Function:

The function module makes available the entry interface of the four tables stored in the device. These tables are needed for the function modules “linearization” and “parameter” control.

Parameter definition:

Table1 0%
First checkpoint value from table 1 for 0 % of the input variable.
...
Table4 100%
Last checkpoint value from table 4 for 100 % of the input variable.

Dead time filter, TOTZ



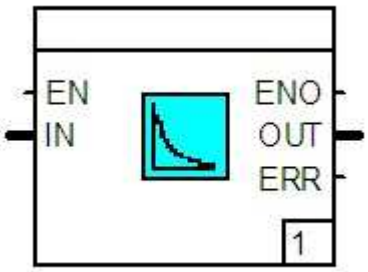
Funktion:

The funktion module can an input signal **IN** , delyed by a parameter definable-dead time, to the outputs **OU1** and **OU2**. Two independently-funktioning dead times are aviable via the two outputs.

Parameter definition:

Dead time 1
First dead time in TIME format e.g. T#4m2s.
Dead time 2
Second dead time in TIME formate e.g. T#4m2s.

Differentiator, DT1



Function:

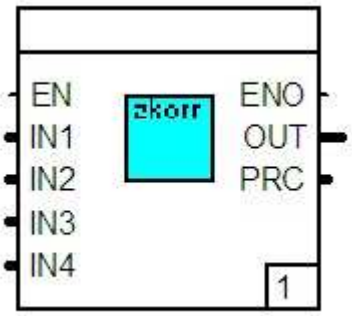
The function module displays a differentiator with parameter definable derivative action time and derivative action gain. The type of differentiation can also be parametrised.

Parameter definition:

Parameters:
Derivative action time
Derivative action time in TIME format e.g. T#4m2s.
Derivative gain
Derivative action gain in REAL format.

Type of differentiation:
Linear
The differentiation is not performed. The output value corresponds to the input value.
diff. bipolar
The differentiation is performed for both positive and negative changes of the input.
diff. unipolar positive
The differentiation is only performed unilaterally for positive changes of the input.
diff. unipolar negative
... negative changes ...

State correction, ZKOR



Function:

The function module provides the functions for various types of state correction.

Each of the signals needed for flow **IN**1, pressure **IN**2, temperature **IN**3 and density **IN**4 are injected at the inputs.

The corrected signal is outputted at the output **OUT** in physical units.

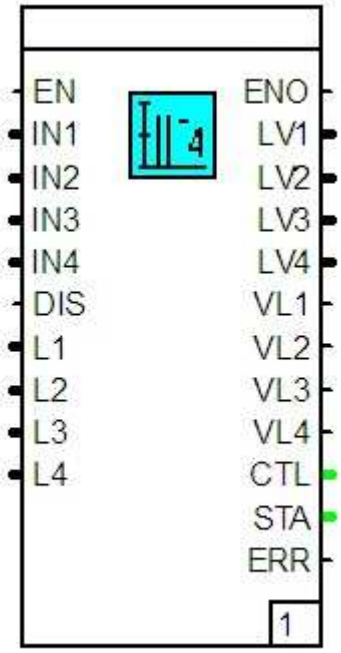
The output **PRC** makes available the corrected signal in the value range 0...100 %. The scaling range for this signal is parameter definable.

Parameter definition:

State correction
Entry of the desired correction task.

The values to be entered for the state correction precisely correspond in their functional capabilities to the parameters of the list configuration of the instrument.

Alarm values, AL4



Funktion:

The functional module independently monitors the alarm value of up to four analog signals at inputs **IN1** to **IN4**. The respective alarm value can be staded as an input signal at inputs **L1** to **L4** or it can also be staded as a constant value. Further, it is also possible to use a percentage part of the input signals as alarm vaue. The alarm values which are adjustablr parameters are stated via outputs **LV1** to **LV4**. The infringement of an alarm value is presented as TRUE signal at outputs **VL1 to VL4**

The alarm value checks can be switched off via input **DIS** The outputs retain their value.

To be able to display and operate the alarm values via the diplay loop of output IND, output **CTL** must be connected to input **CAL** of the function module display loop.

No error stati are currently outputted at output **STA** or **ERR.**

Parameter definition:

Alarm values:
Nr
Number of the alarm value

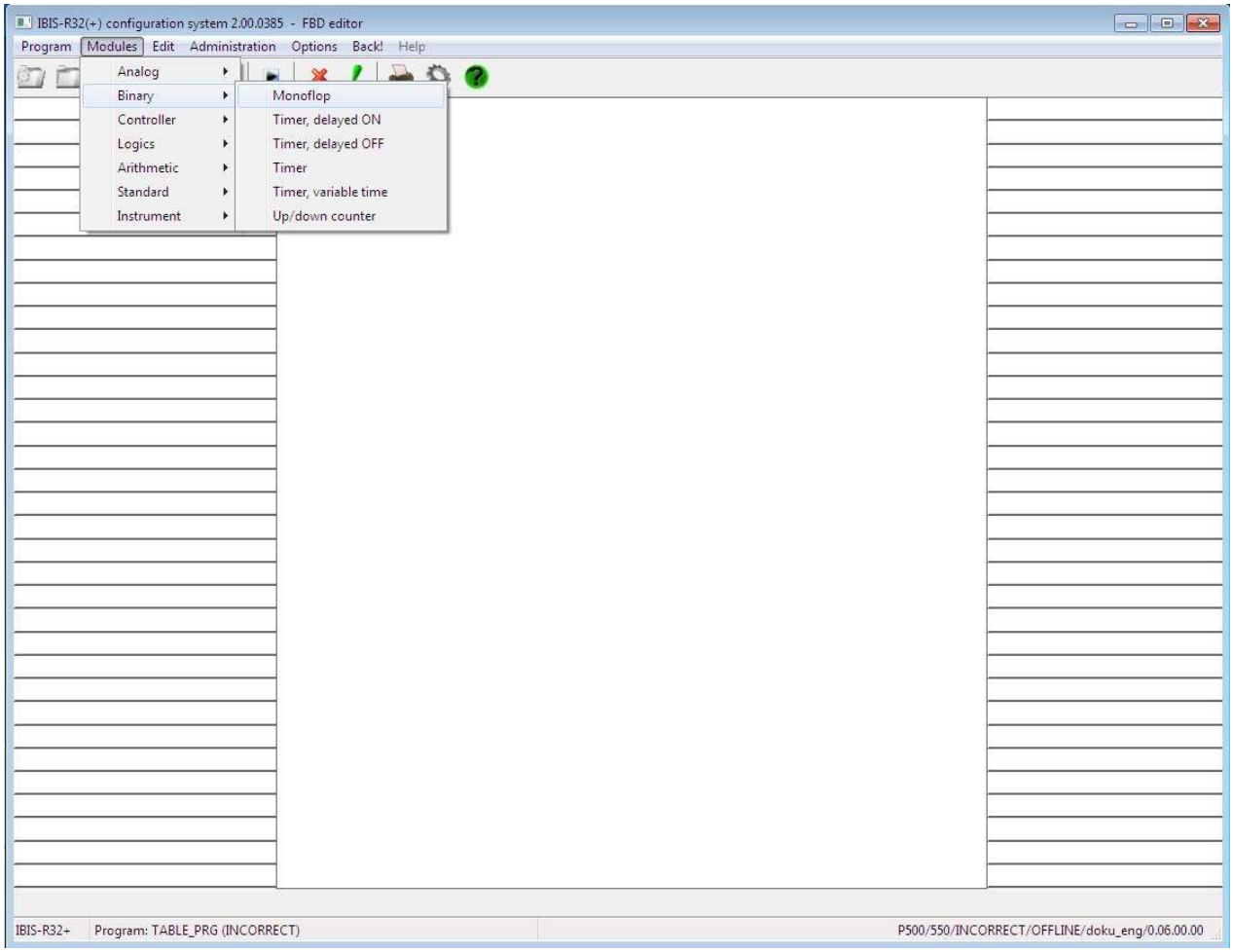
Type

“ ”	Alarm value not used
MIN	Signal is checked for undershooting of the limit alarm
MAX	... for overshooting of the alarm value
XW-MIN	Signal wird als Regelabweichung auf Unterschreitung des Grenzwertes überprüft.
XW-MAX	Upon overshooting ...
d/dt[s]	Upon overshooting the alarm value, the speed of change of the signal is checked. The output of this text is used with the selected time base in seconds.
d/dt[min]	... in minutes.
d/dt[h]	... in hours.
Abs. Value |IN|	Prior to the alarm value check, the absolute avlue of the signal shall be determined and used for the check.

Alarm value

%*Signal	The alarm value to be checked. If the associated input in the signal column is not marked [ ], then this is the direct alarm value. If the column is marked [X], then this value is a percentage weighting of the alarm value. The actual alarm value is then built from the input signal at **L1** to **L4**, multiplied with this percentage weighting.
Signal	[x] The signals **L1** to **L4** multiplied with the parameters (as percentage weighting) stated in “Alarm value % * signal” are used as the limit alarm. [ ] The parameters stated in “Alarm value % * signal” are used directly as physical units.
Hysteresis of Alarm value	The associated hysteresis in the physical unit of signals **IN1** to **IN4** (not in %) for the cancellation of the alarm signal.
Time base	The set time base is used for gradient limit alarm checks.

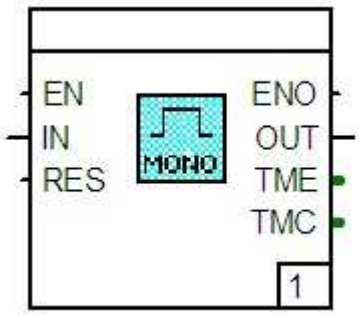
5.6.4 Modules, binary



Overview modules, binary

MONOF Monoflop
TVE Timer, delayed switch-on
TVA Timer, delayed switch-off
TTE Timer, time-out
VRZ Up/down counter

Monoflop, MONO_F

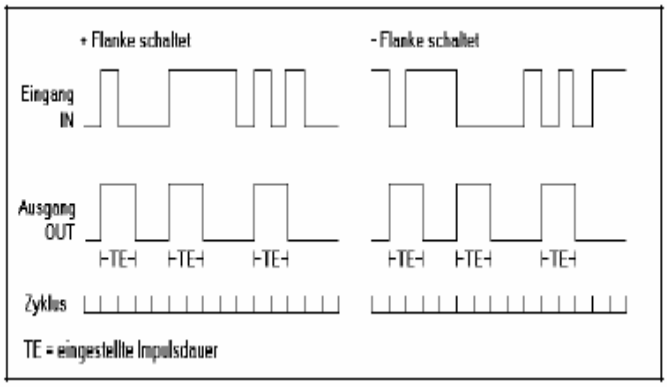


Function:

The function module produces a binary output signal of parameter-definable duration. The positive or negative flank of the binary signal **IN** is used as triggering signal.

The reset signal **RES** enables the output signal to be reset prematurely.

The pulse duration set can be interrogated at the output **TME**, the time already elapsed at the output **TMC** in milliseconds format.



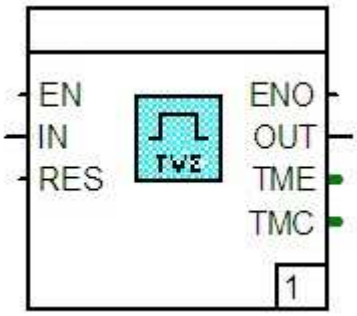
TME Pulse duration

Parameter definition:

Pulse:
Pulseduration
Duration of the output pulse in the TIME format e.g. T#4m2s.

Trigger:
Switch on+flank
Switch at positive flank
Switch on − edge
Switch at negative flank.

Timer, delayed switch-on, TVE



Function:

This module is used for chronological control and monitoring of specific operating conditions. It delays the switch-on by a configurable time.

The switch-off is transmitted undelayed.

The reset signal **RES** enables the output signal to be reset prematurely.

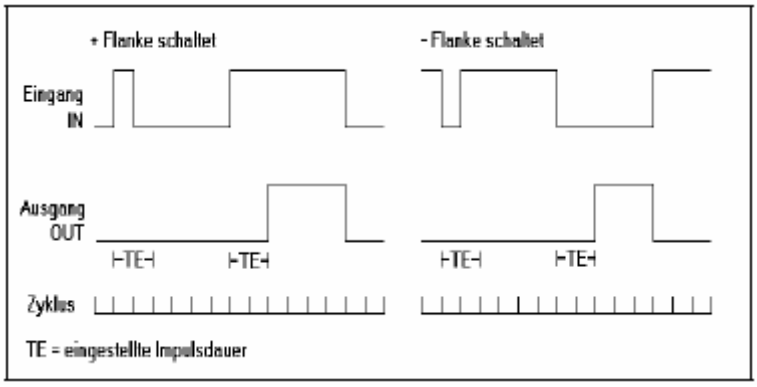
The triggering of positive or negative flank can be parameter defined.

The delay time set can be interrogated at the output **TME**, the time already elapsed at the output **TMC** in milliseconds format.

Parameter definition:

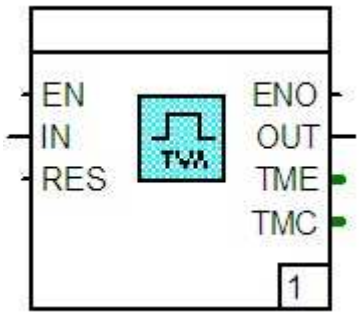
Time:
Delay time
in TIME format e.g. T#4m2s.

Trigger:
Switchon+flank
Switch at positive flank.
Switch on−flank
Switch at negative flank.



TE Pulse duration

Timer, delayed switch-off, TVA



Function:

This module is used for chronological control and monitoring of specific operating conditions. It delays the switch-off by a configurable time.

The input signal change **IN** is transmitted undelayed.

The reset signal **RES** enables the output signal to be reset prematurely.

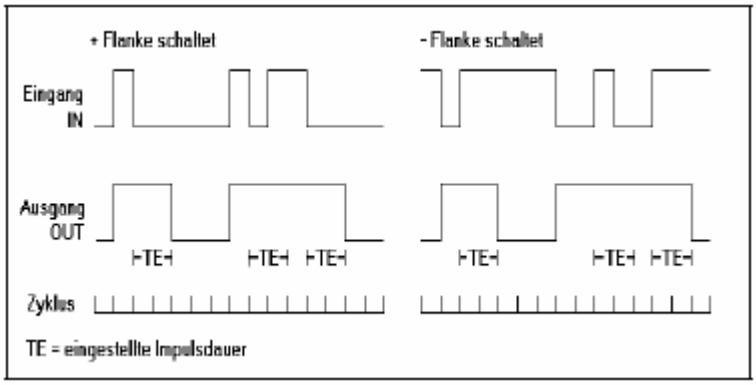
The triggering of positive or negative flank can be parameter defined.

The delay time set can be interrogated at the output **TME**, the time already elapsed at the output **TMC** in milliseconds format.

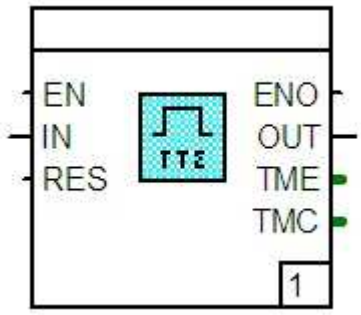
Parameter definition:

Time:
Delay time
in TIME format e.g. T#4m2s.

Trigger:
Switch on+edge
Switch at positive flank
Switch on−edge
Switch at negative flank.



Timer, time-out, TTE



Function:

This module is used for the chronological control and monitoring

of specific operating conditions. It limits the switch-on to a configurable time.

The input signal change is transmitted undelayed.

The reset signal **RES** enables the output signal to be reset prematurely.

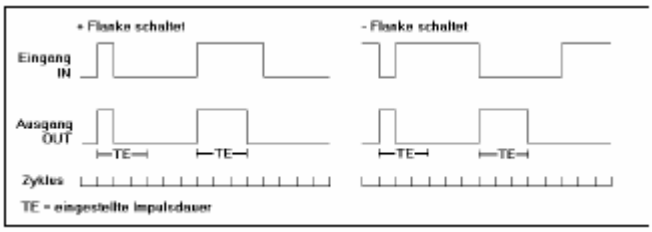
The triggering of positive or negative flank can be parameter defined.

The switch-on time set can be interrogated at the output **TME**, the time already elapsed at the output **TMC** in milliseconds format.

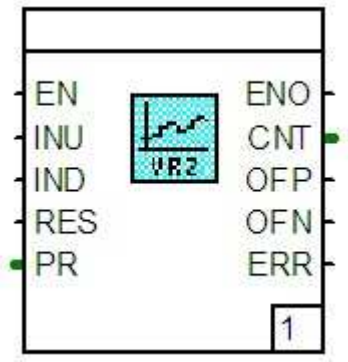
Parameter definition:

Time:
Switch-on time
in TIME format e.g. T#4m2s.

Trigger:
Switch on +flank
Switch at positive flank
Switch on −flank
Switch at negative flank.



Up/down counter, VRZ



Function:

This module enables piece-good processes or quantity measurements to be monitored with pulse generators for intakes and outflows.

An internal signed 32-bit counter contains the balance of the intake and outflow pulses (**INU**, **IND**) since the last reset. This count (weighted with a rank) is routed to the output **CTC.**

The overflow limits in both the positive and negative range are monitored. If these are exceeded, the corresponding overflow output **OFP** or **OFN** is set for a number of program cycles and the counter is reset. A cascading of forward/backward counters can be performed in this way.

After a reset **RES**, the configurable basic value **PR** is accepted as starting value. To make a quantity measurement possible with pulse generators, the module contains a parameter-definable rank. If the input **PR** (basic value) is above or below the permitted range, the module is set as faulty (**ERR** = 1).

Parameter definition:

Output parameter:

Default value

Starting value at beginning of counting or resetting.
Entry only possible if no signal has been connected
at the module input PR.

Rank

Weighting factor for the counter output.

Overflow cascading:

Positive

Limit in the positive range, at which the output **OFP** is
set and the counter reset.

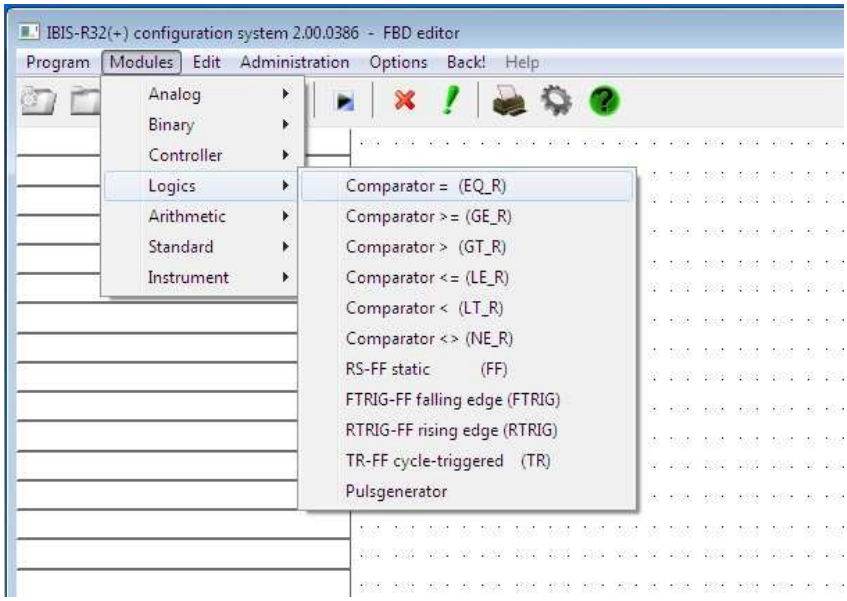
Negative

Limit in the negative range, at which the output **OFN** is
set and the counter reset.

Cykles aktive

Once the positive or negative limit has been reached,
the output **SCP** or **SCN** remains set for the

5.6.5 Modules, logic



Overview modules, logic

R_TRIG	Rising edge detection
F_TRIG	Falling edge detection
FF	RS flipflop
RS	Trigger

EQ_R	Comparator = with tolerance
GE_R	Comparator ≥ with differential gap
GT_R	Comparator > with differential gap
LE_R	Comparator ≤ with differential gap
LT_R	Comparator < with differential gap
NE_R	Comparator <> with tolerance

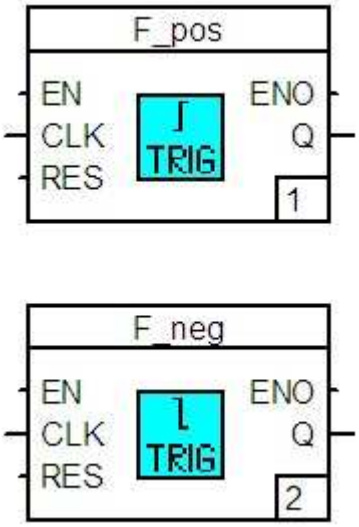
Inputs for function modules, edge detection

Abbr.	Signal designation	Data type
CLK	Input	BOOL
EN	The module is processed with TRUE	BOOL
RES	Reset output Q	BOOL

Outputs for function modules, edge detection

Abbr.	Signal designation	Data type
ENO	Processing status the module is processed with TRUE	BOOL
Q	Output	BOOL

Edge detection RTRIG, FTRIG



Function:

If a positive flank occurs at the input **CLK** with the function RTRIG (with the function FTRIG a negative flank), then the output **Q** is set to TRUE for a configurable number of cycles and then again to FALSE, or continuously to TRUE until the reset.

The data formats of the inputs and outputs of these functions are of the type BOOL.

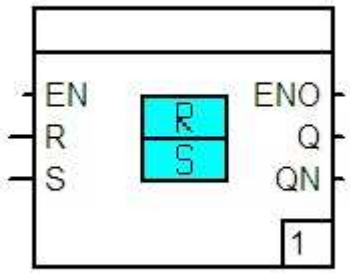
Parameter definition:

Cycles number for output
The output **Q** remains set to TRUE for n (= 1...99) cycles.

Reset output after cycles

- o A TRUE at the output **Q** is only set to FALSE via the reset input **RES**.
- x The output is set to FALSE after n cycles.

RS flipflop, FF



Function:

The flipflop is used for short or long term storage of logical binary states.

A reset or set priority (RS or SR flipflop) can be selected within the parameter definition mask. The logical state is avail-able at the output **Q**: the inverse at the output **QN**.

All inputs and outputs are of the data type BOOL.

The tables below show the method of working:

RS flipflop		
Input		Output
S	R	Q
0	0	x
0	1	0
1	0	1
1	1	0

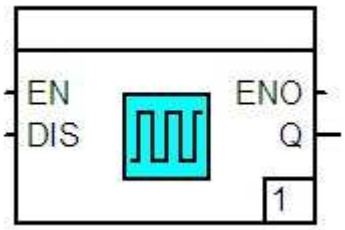
SR flipflop		
Input		Output
S	R	Q
0	0	x
0	1	0
1	0	1
1	1	1

Tab. 3

Parameter definition:

RS flipflop
The reset signal **R** has priority.
SR flipflop
The set signal **S** has priority.

Trigger, TR



Function:

This function module produces a signal with the binary states 0 and 1 at the output **Q**.

The changeover takes place in the processing cycle. No scaling is possible.

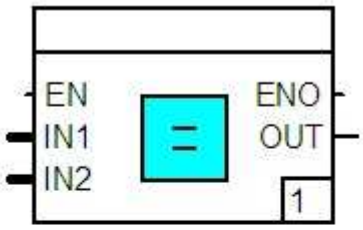
If there is a TRUE at the output **DIS**, then the state of the output **Q** is retained, a FALSE or a TRUE is outputted (this selection is made in the parameter definition mask).

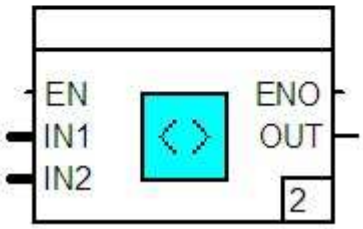
Parameter definition:

Pulse scaling
Scaling factor in the range 1 ... 999.

Behaviour with DISABLE:
Keep old output
The current signal state **Q** is retained.
Output low-signal
FALSE at the output **Q**.
Output high-signal
TRUE at the output **Q**.

Comparator = and <> with tolerance





Function:

The input **IN1** is compared with the input **IN2** for equality or inequality.

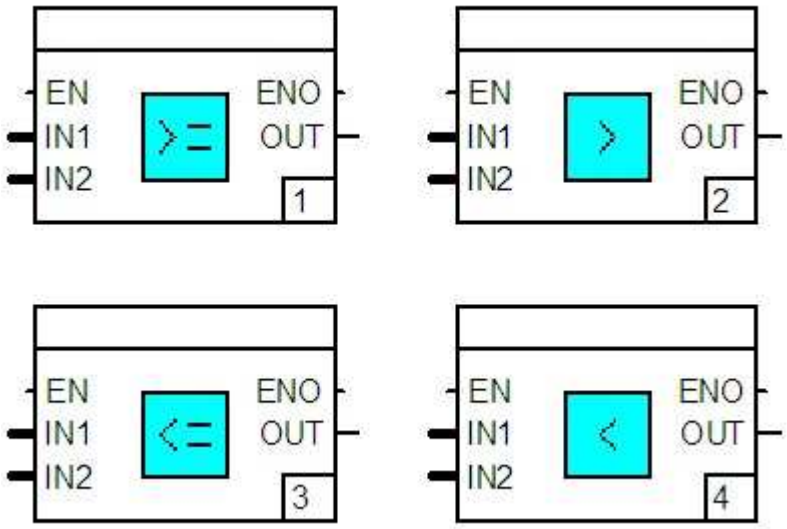
If the condition is satisfied within a declarable tolerance, the output is at TRUE, otherwise at FALSE.

The two input values are of the type REAL.

Parameter definition:

Tolerance value +/-
The magnitude of the difference is used for the comparison

Comparator ≥,>, ≤ and < with differential gap



Function:

The input **IN1** is compared with the input **IN2** in respect of the desired condition.

If the condition is satisfied, the output is at TRUE. It only switches back to FALSE once **IN2** deviates more than the declarable differential gap of **IN1**.

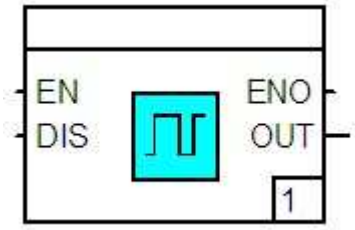
The two input values are of the type REAL.

Parameter definition:

Value for differential gap

The differential gap is used as hysteresis in the derivation of FALSE.

Pulse generator, PULS



Function:

The functional module provides a parameterizable pulse generator.

The changing signal is outputted at output **OUT**.

The output can be switched off via the input **DIS**. If a TRUE signal is available at **DIS**, FALSE will be outputted at output **OUT**. Since the pulse generation takes place independently of **DIS**, a TRUE signal shorter than the parameterized on-time can eventually be outputted after the output has been released.

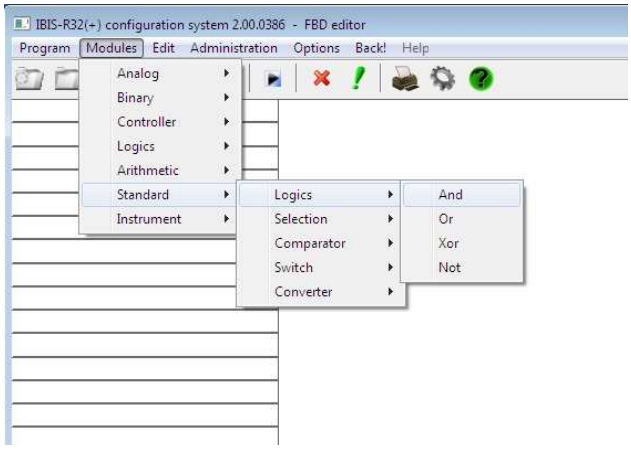
If the processing cycle of the configuration on the controller lasts longer than the set times, pulses could cease coming. The accuracy of the set times of the individual pulses / pulse pauses is only maintained with the grid of the processing cycle.

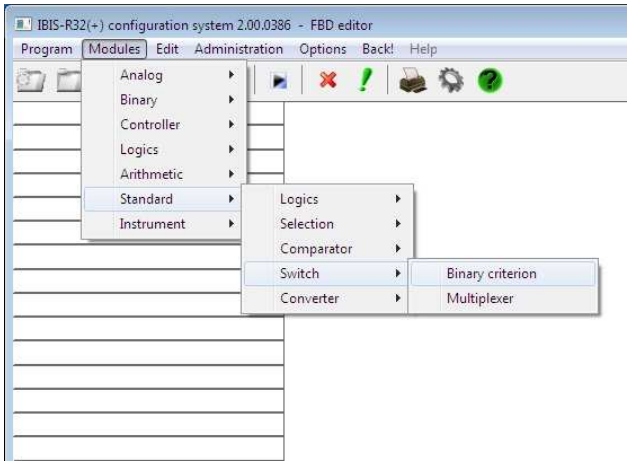
Parameter definition:

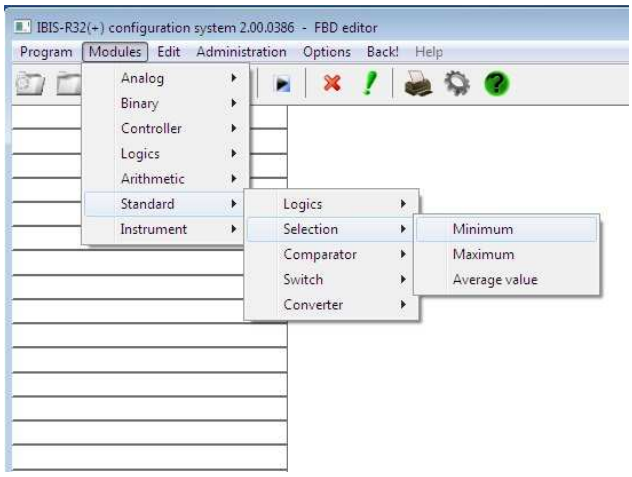
On-time
The time period required for TRUE to be outputted at output **OUT**.

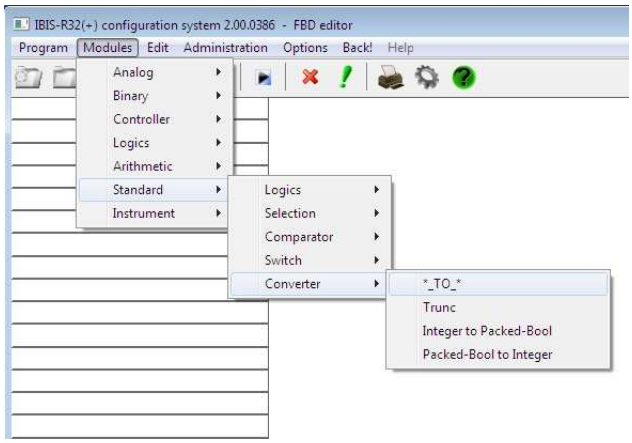
Off-time
The time period required for FALSE to be outputted at output **OUT**.

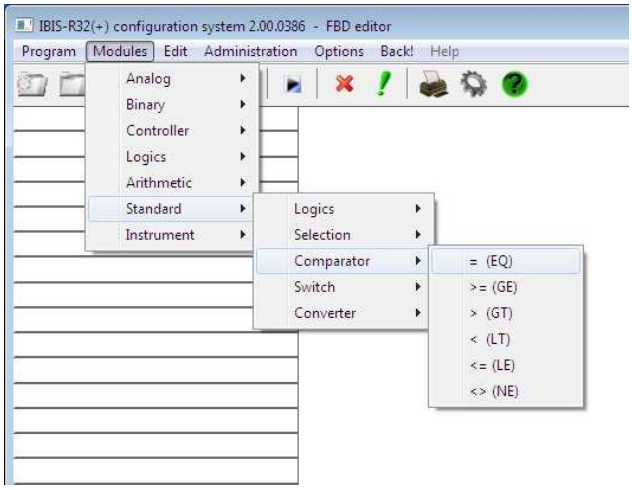
5.6.6 Modules, standard









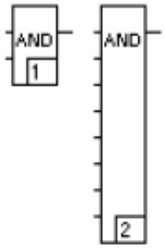


Overview modules, standard –logic

AND Logical AND module
OR Logical OR module
XOR Logical EXOR module
NOT Negation module

Module inputs and outputs are unnamed.

Logical AND module, AND



Function:

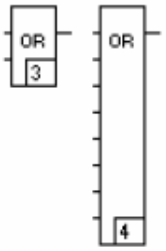
This module performs a bitwise AND combination of the inputs and routes the result to the output. The data types BOOL, INT and DINT are permitted as inputs and outputs.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 30 inputs.

Parameter definition:
None

Logical OR module, OR



Function:

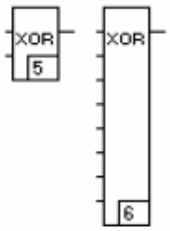
This module performs a bitwise OR combination of the inputs and routes the result to the output. The data types BOOL, INT and DINT are permitted as inputs and outputs.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 30 inputs.

Parameter definition:
None

Logical EXOR module, XOR



Function:

This module performs a bitwise EXCLUSIVE OR combination of the inputs and routes the result to the output. The data types BOOL, INT and DINT are permitted as inputs and outputs.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 30 inputs.

Parameter definition:
None

Negation module, NOT

Function:

The input signal of the data type BOOL, INT or DINT is routed to the output negated.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

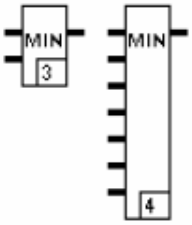
Parameter definition:
None

Overview modules, standard -selection

MIN	Minimum
MAX	Maximum
AVG	Average value

Module inputs and outputs are unnamed.

Minimum, MIN



Function:

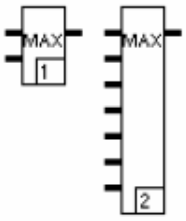
The smallest input value is transmitted to the output. All input and output signals are of the same data type REAL, INT or DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 10 inputs.

Parameter definition:
None

Maximum, MAX



Function:

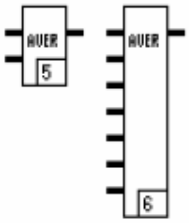
The largest input value is transmitted to the output. All input and output values are of the same data type REAL, INT or DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 10 inputs.

Parameter definition:
None

Average value, AVG



Function:

The average of the input values is determined and transmitted to the output. All input and output values are of the same data type REAL, INT and DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 10 inputs.

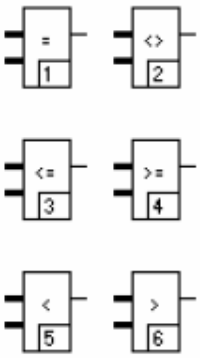
Parameter definition:
None

Overview modules, standard -comperator

EQ Inputs equal =
GE first input is larger than/equal to second input >=
GT first input is larger than second input >
LT first input is smaller than second input <<
LE first input is smaller than/equal to second input <=
NE Inputs unequal <>

Module inputs and outputs are unnamed.

Inputs equal, larger/equal, larger, smaller, smaller/equal, unequal



Function:

The first input is compared with the second input. If the condition is met the output is at TRUE, otherwise at FALSE.

The two input values are of the same data type REAL, INT and DINT.

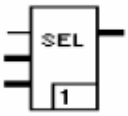
The input and output data types are set via the menu item →*Edit* →*Define signal type*.

Parameter definition:
None

Overview modules, standard -switch

SEL Binary criterion
MUX Multiplexer

Binary criterion, SEL



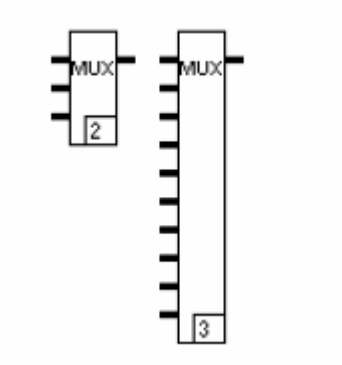
Function:

This function has 3 inputs. The upper input is always of the data type BOOL. It is possible with this to route one of the two inputs below to the output, according to the signal applied, this being the centre input with a FALSE and the lower input with a TRUE. The inputs to be switched and the output are both of the same data type, REAL, INT and DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

Parameter definition:
None

Multiplexer, MUX



Function:

The upper input is always of the data type INT. It is possible with this to route one of the two inputs below to the output, depending on the signal applied. The second input (first switching value) connects through with an integer value of 1 and the last one with
9. The output is not defined outside this range 1 ... 9. The inputs to be switched and the output are of the same data type REAL, INT and DINT.

The input and output data types are set via the menu item →Edit
→Define signal type.

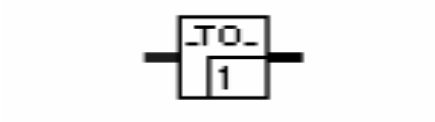
The maximum number of inputs to be switched is 9.

Parameter definition:
None

Overview modules, standard -converter

TO Conversion of data types
TRUNC Conversion of REAL to INTEGER data types

Conversion of data types, *_TO_*



Function:

The functions *_TO_* offer the possibility of converting one data type into another.
The same module is always displayed, although the signal connections from input and output differ from module to module – depending on the conversion.

There are four conversions:
− DINT_TO_REAL
− DINT_TO_INT
− INT_TO_DINT
− REAL_TO_DINT

Value infringements can arise in the DINT_TO_INT conversion. In this case the result is limited to the minimum and/or maximum value to be displayed.

Parameter definition:

None

Conversion of REAL to INTEGER data types, TRUNC

Function:

The function TRUNC converts a floating point value REAL into a fixed-point value (INT, DINT).

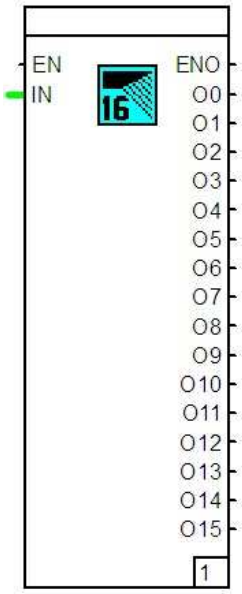
There is no rounding, i.e. a component larger than 0 after the decimal point is cut off. Because the number range of the data type REAL is larger than that of the fixed-point numbers, value violations may occur in the conversion. The behaviour of the TRUNC functions in these cases is described below.

Parameter definition:
None

Output data type	Conversion
INT	Floating-point value in 16-bit value with sign. Alarm cases: − REAL value > INTmax value, then INT value = INTmax value. − REAL value < INTmin value, then INT value = INTmin value
DINT	Floating-point value in 32 bit value with sign Alarm cases: − REAL value > DINTmax value, then DINT value = DINTmax value. − REAL value < DINTmin value, then DINT value = DINTmin value.

I2BP Integer to Packed-Bool

Icon and Functionblock



Library

as of 3.4.0

Function

The module decodes the 16 bits of an integer signal into 16 bool signals.

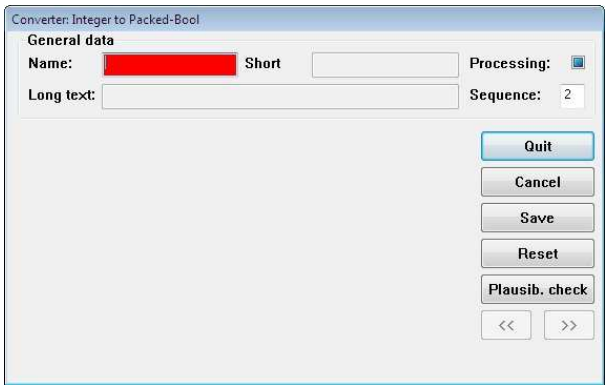
Inputs

EN BOOL According to IEC 1131.
IN INT

Outputs

ENO BOOL According to IEC 1131.
O0 BOOL represents rank 2^0 at the integer input
O1 BOOL represents rank 2^1 at the integer input
O2 BOOL represents rank 2^2 at the integer input
O3 BOOL represents rank 2^3 at the integer input
O4 BOOL represents rank 2^4 at the integer input
O5 BOOL represents rank 2^5 at the integer input
O6 BOOL represents rank 2^6 at the integer input
O7 BOOL represents rank 2^7 at the integer input
O8 BOOL represents rank 2^8 at the integer input
O9 BOOL represents rank 2^9 at the integer input
O10 BOOL represents rank 2^{10} at the integer input
O11 BOOL represents rank 2^{11} at the integer input
O12 BOOL represents rank 2^{12} at the integer input
O13 BOOL represents rank 2^{13} at the integer input
O14 BOOL represents rank 2^{14} at the integer input
O15 BOOL represents rank 2^{15} of sign bit at the integer input

Parameter definition

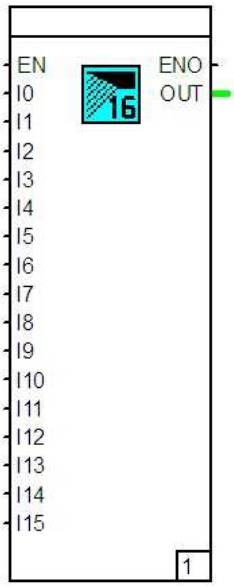


Plausibility checks

none

BP2I
Packed-Bool to Integer

Icon and Functionblock



Library

as of 3.4.0

Function

The module encodes the 0/1 states of up to 16 Bool signals into the 16 bits of an integer signal.

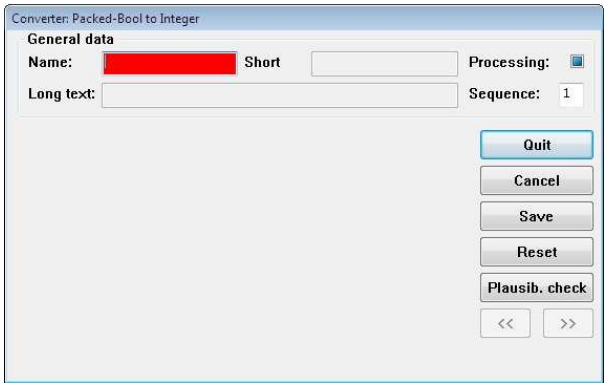
Inputs

EN	BOOL According to IEC 1131.
I0	BOOL represents rank 2^0 at the integer output
I1	BOOL represents rank 2^1 at the integer output
I2	BOOL represents rank 2^2 at the integer output
I3	BOOL represents rank 2^3 at the integer output
I4	BOOL represents rank 2^4 at the integer output
I5	BOOL represents rank2^4 at the integer output
I6	BOOL represents rank 2^6 at the integer output
I7	BOOL represents rank 2^7 at the integer output
I8	BOOL represents rank 2^8 at the integer output
I9	BOOL represents rank 2^9 at the integer output
I10	BOOL represents rank 2^{11} at the integer output
I12	BOOL represents rank 2^{12} at the integer output
I13	BOOL represents rank 2^{13} at the integer output
I14	BOOL represents rank 2^{14} at the integer output
I15	BOOL represents rank $2^{1\%}$ of sign bit at the integer output

Outputs

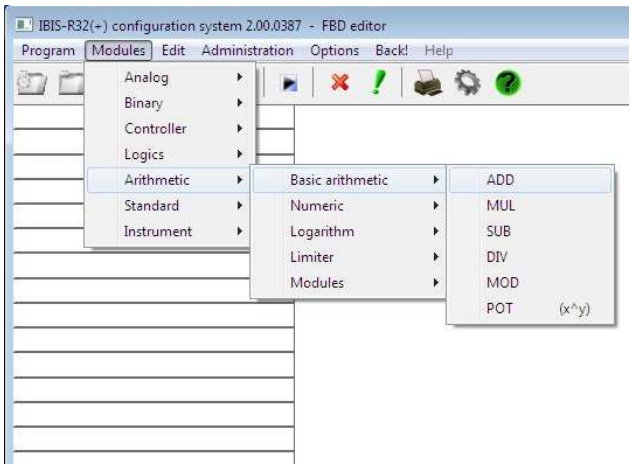
ENO	BOOL According to IEC 1131.
OUT	INT Output for the composite integer signal

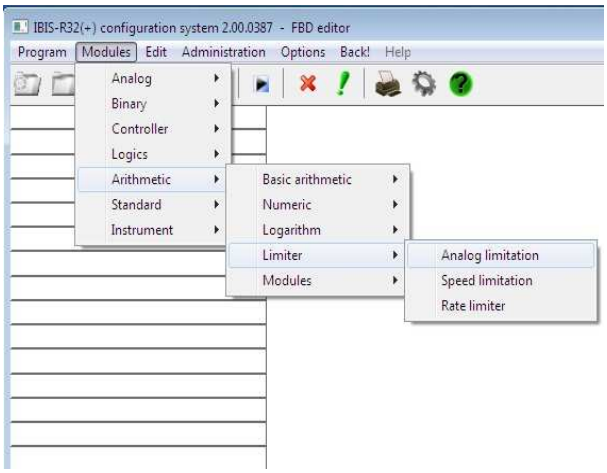
Parameter definition

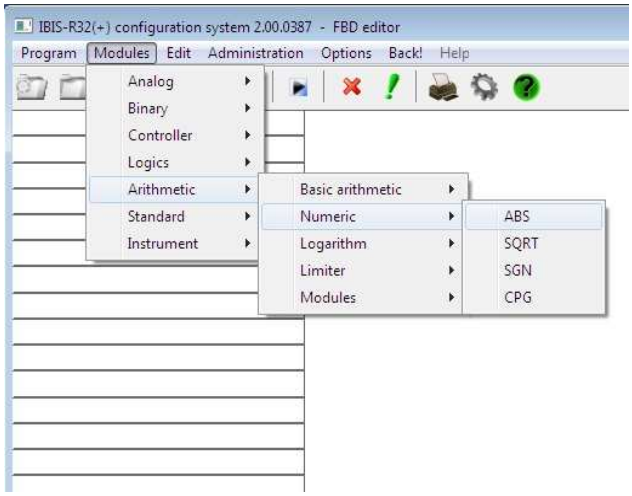


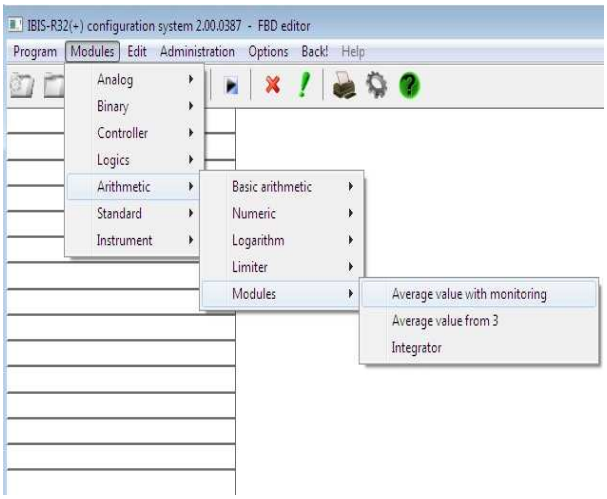
Plausibility checks
none

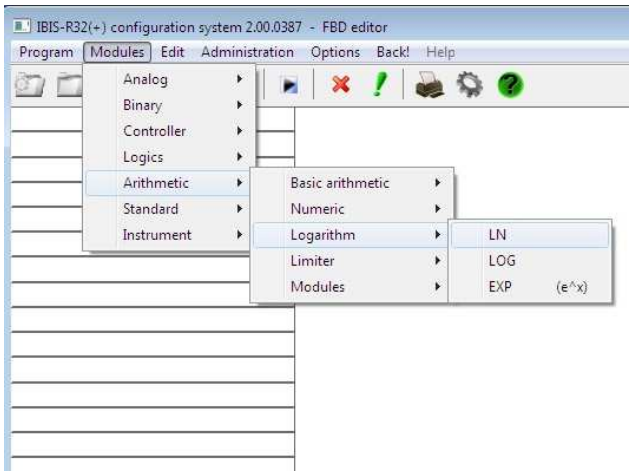
5.6.7 Modules, arithmetic









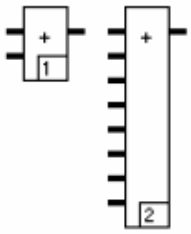


ADD Addition
MUL Multiplication
MOD Modulo

DIV Division
SUB Subtraction
POT Exponentiation

Module inputs and outputs are unnamed.

Addition, ADD



Function:

This function adds the inputs with correct sign and passes the result to the output.

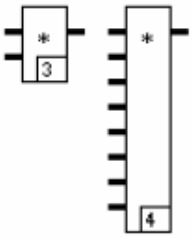
All input and output values are of the same data type REAL, INT or DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 10 inputs.

Parameter definition:
None

Multiplication, MUL



Function:

This function multiplies the inputs with correct sign and passes the result to the output.

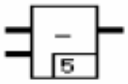
All input and output values are of the same data type REAL, INT or DINT.

The input and output data types are set via the menu item →*Edit* →*Define signal type*.

The module can be enlarged up to 10 inputs.

Parameter definition:
None

Subtraction, SUB



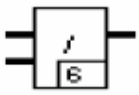
Function:

This function subtracts the second input from the first with correct sign and passes the result to the output.

All input and output values are of the same data type REAL, INT or DINT.

Parameter definition:
None

Division, DIV



Function:

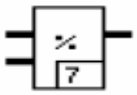
This function divides the first input by the second with correct sign and passes the result to the output.

All input and output values are of the same data type REAL, INT or DINT.

Note
Division by zero must be captured outside of the function. If this is not the case, the output is given an undefined result.

Parameter definition:
None

Modulo, MOD



Function:

This function divides the first input by the second with correct sign and passes the division remainder to the output as result.

All input and output values are of the same data type DINT or INT.

Parameter definition:
None

Exponentiation, POT

Function:

The first input designates x and the second y.

This function calculates from the inputs x^y and passes the result to the output.

The input and output values are of the same data type REAL.

Parameter definition:
None

Overview modules, arithmetic -numerical

ABS Absolute value
SQRT Square root
SGN Sign
CPG C level

Module inputs and outputs are unnamed.

Absolute value, ABS

Function:

The input value is passed to the output as magnitude, i.e. with positive sign.

Input and output values are of the same data type REAL, DINT or INT.

Note
If the negative maximum value is converted by means of the function an error message is produced because the numerical range has been exceeded. In this case the output is undefined.

Parameter definition:
None

Square root, SQRT

Function:

This function calculates the square root from the input signal and passes the result to the output. Input and output values are of the same data type, REAL.

Note
The function is only permissible for positive values. The output is given an undefined result with negative values.

Parameter definition:
None

Sign, SGN

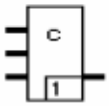
Function:

The sign of the input value is displayed at the output. The input value is of the data type REAL, DINT or INT, the output value always of data type INT.

A "-1" is obtained as output value with negative numbers, with positive numbers a "+1", and a "0" with the number "zero".

Parameter definition:
None

C-Pegel, CPG



Function:

The C level can be calculated with the function CPG from the variables temperature [°C], oxygen content (measured in mV with a zirconium dioxide sensor), and the partial pressure of the CO content.

The temperature is injected at input 1, the oxygen content at input 2 [mV] and the partial pressure at input 3.

The C level is calculated:

$$C = \frac{179{,}45 * Z}{43{,}15 * Z + 1{,}0}$$

Parameter definition:
None

where

$$Z = 10^{\log PCO - \log A - \frac{8725}{T} - 4{,}5604}$$

and where

$$A = (10^{\frac{-E}{0{,}0496 * T}} * 0{,}209)^{0{,}5}$$

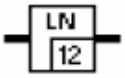
E Measured signal of the zirconium oxide sensor in mV
T Temperature in K
PCO Partial pressure of the CO

Overview modules, arithmetic -logarithmic

LN Natural logarithm
LOG Decimal logarithm
EXP Exponent

Module inputs and outputs are unnamed.

Natural logarithm, LN



Function:

This function determines the natural logarithm to base e (= 2.71828..) as a function of the input and passes the result to the output. The function is only permissible for positive values. Input and output values are of the data type REAL.

Note
The output receives an undefined result with negative numbers.

Parameter definition:
None

Decimal logarithm, LOG

Function:

This function determines the natural logarithm to base 10 as a function of the input and passes the result to the output. The function is only permissible for positive values. Input and output values are of the data type REAL.

Note
The output receives an undefined result with negative numbers.

Parameter definition:
None

Exponent, EXP

Function:

This function calculates the power to base e (= 2.71828..) of the natural logarithm as a function of the input, and passes the result to the output. Input and output values are of the data type REAL.

Parameter definition:
None

Overview modules, arithmetic -limiter

LIM_A Analog limitation
LIM_D Speed limitation
RL Rate limiter

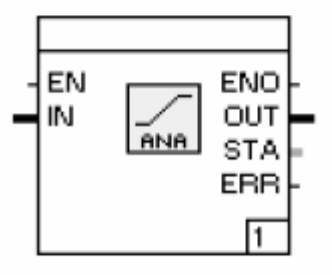
Inputs for arithmetic limiting modules

Abbr.	Signal designation	Data type
EN	the module is processed with TRUE	BOOL
IN	Input signal at LIM_A, LIM_D	REAL

Outputs for arithmetic limiting modules

Abbr.	Signal designation	Data type
ENO	Processing status, the module is processed with TRUE	BOOL
ERR	Error in module with TRUE	BOOL
OUT	Output signal with LIM_A, LIM_D	REAL
STA	Error status with LIM_A	INT

Analog limitation, LIM_A



Function:

The analog input signal **IN** is checked for a parameter-definable upper or lower limit and limited if necessary. The conditions of the limiting are:

	Input IN	Output OUT
upper limit HI	IN > HI IN ≥ HI	IN HI
lower limit LO	IN ≤ LO IN > LO	LO IN
upper and lower limit	IN ≥ HI LO < IN < HI IN ≤ LO	HI IN LO

Parameter definition:

Limitation to:
Lower limit
Checking the lower signal limit.

Upper limit
... upper signal limit.
Both limits
Checking the lower and upper signal limits.

Alarm limits:
Lower limit value
Lower signal limit.
Upper limit value
Upper signal limit.

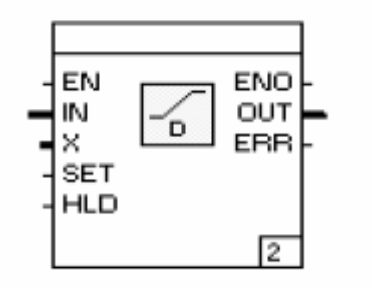
Tab. 4

If the input signal exceeds or undershoots the permitted limits, the error signal **ERR** is set to TRUE.

If the upper limiting is exceeded, the output **STA** is set to 1.

If the lower limiting is undershot, the output **STA** is set to 2.

Speed limitation, LIM_D



Function:

The analog input signal is checked for a parameter-definable upper and/or lower change rate and limited if necessary. The conditions of limiting are:

	Eingang IN	Ausgang OUT
obere Begrenzung HI	$(IN-IN_{ta-1})*ZB/ta \geq HI$ $(IN-IN_{ta-1})*ZB/ta < HI$	$IN_{t-1}+HI \cdot ZB/ta$ IN
untere Begrenzung LO	$(IN-IN_{ta-1})*ZB/ta > LO$ $(IN-IN_{ta-1})*ZB/ta \leq LO$	IN $IN_{t-1}+LO*ZB/ta$
obere und untere Begrenzung	$(IN-IN_{ta-1})*ZB/ta \geq HI$ $LO < (IN-IN_{ta-1}) \cdot ZB/ta < HI$ $(IN-IN_{ta-1})*ZB/ta \leq LO$	$IN_{t-1}+HI*ZB/ta$ IN $IN_{t-1}+LO*ZB/ta$

Tab. 5

The deviation between the inputs **IN** and **X** is simultaneously calculated and monitored via the input **X**. If the deviation is greater than the parameter-definable alarm values, the output is held at the last value until the deviation is once again within the permissible range.

The output is set to the value of the input **X** via a TRUE at the input **SET**.

The output value can be held via a TRUE at the input **HLD**.

If the input signal is above or below the permissible limits, the error signal **ERR** is set to TRUE.

The output can be switched to the value of the signal at the input **X** without delay via a TRUE at the input **SET**.

Parameter definition:

Speed limitation to:
Lower limit
Checking the change rate downwards.
Upper limit
... upwards.
Both limits
... upwards and downwards.
Off
The change rate is not limited.

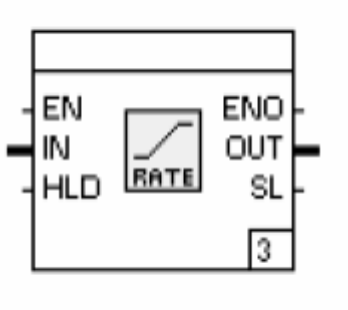
Time base
The change rate is related to second, minute, hour or day.

Speed limit values:
Lower limit
Change rate downwards in REAL format.
Upper limit
... upwards in REAL format.

Delta X, IN limitation to:
Lower limit
Checking whether **X** is smaller than **IN** by this deviation magnitude.
Upper limit
... greater than **IN** by this deviation magnitude.
Both limits
... smaller or greater than **IN** by the deviation magnitudes.
Off
No checking of the deviation between **X** and **IN**.

Delta X, IN limit values:
Lower limit
Deviation magnitude downwards in REAL format.
Upper limit
... upwards in REAL format.

Rate limiter, RL



Function:

The analog input signal **IN** is checked for a parameter-definable upper and lower change rate and limited if necessary.

The conditions must be taken from the change rate module.

If the input signal exceeds or undershoots the permissible limits, the output **SL** is set to TRUE. The output value can be held with a TRUE at the output **HLD.**

Parameter definition:

Limitation to:

Negative changes
Checking the change rate downwards.
Positive changes
... upwards.
Neg. and pos. changes
... upwards and downwards.

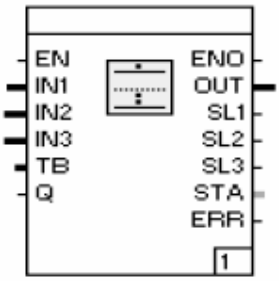
Limits:
Negative change
Change rate downwards in REAL format.
Positive change
... upwards in REAL format.

Time base
The change rate is related to second, minute, hour or day.

Overview modules, arithmetic -modules

MW_UE Average value with monitoring
MW_3 Average value from 3

Average value with monitoring, MW_UE



Function:

The average value can be determined from three variables at the inputs **IN1**, **IN2** and **IN3** with this module.

It is checked at the same time whether these inputs are lying within the tolerance band around the mean value. This tolerance band can either be fed in via the input **TB** or specified as a parameter. If one of the signals falls outside of the tolerance band, it is not used for the average value determination and a TRUE is outputted at the corresponding output **SL1**, **SL2** or **SL3**. These outputs remain TRUE until a TRUE is fed in at the input Q for the acknowledgment.

The average value is available at the output **OUT**.

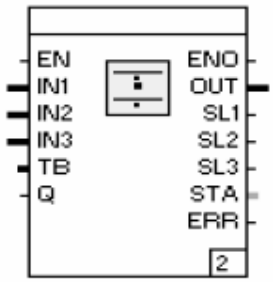
The output **ERR** is set to TRUE if an error has occurred, an error code is outputted at the output **STA**:

0 No errors with average value determination.
-1 The tolerance band is negative.
1 One input falls outside of the average value determination.
2 Two inputs fall outside of the average value determination.

Parameter definition:

Tolerance band
Permanently set value of the tolerance band.

Average value from 3, MW_3



Function:

The average value can be derived from three variables at the inputs **IN1**, **IN2** and **IN3** with this module.

It is checked at the same time whether the two remaining inputs lie within the tolerance band around the mean value. This tolerance band can either be fed in via the input **TB** or specified as a parameter. If one of the signals falls outside of the tolerance band, it is not used for the average value determination and a TRUE is outputted at the corresponding output **SL1**, **SL2** or **SL3**. These outputs remain TRUE until a TRUE is fed in at the input **Q** for the acknowledgment.

The average value is available at the output **OUT**.

The output **ERR** is set to TRUE if an error has occurred, and an error code is outputted at the output **STA**:

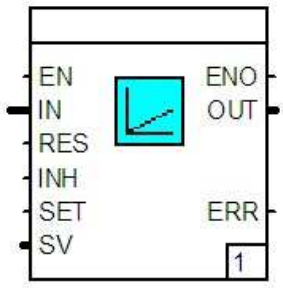
-1 The tolerance band is negative.
0 No signal is lying outside of the tolerance band
1 One signal is lying outside of the tolerance band
2 Two signals are lying outside of the tolerance band.

Parameter definition:

Tolerance band
Permanently set value of the tolerance band.

INTEG
Integrator

Icon and Functionblock



Library

as of 3.4.0

Function

The module sums up the signal at **IN** and outputs it at **OUT**. The time base is set as a parameter in the module. The trapezoidal process is applied as method of integration:

with
T_0 = scan time
A_{j-1} = output value in previous cycle
E_j = momentary value at input
E_{j-1} = input value in previous cycle

Inputs

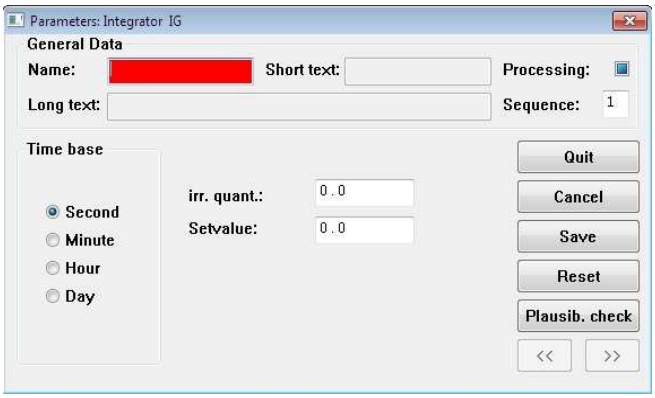
EN	BOOL	Accordung to IEC 1131.
IN	REAL	Input.
RES	BOOL	TRUE sets the output to 0
INH	BOOL	stops the integration process and holds on to the momentary integration result, if SET or RES are not simultaneously TRUE.
SET	BOOL	TRUE sets the output to the set value if RES is not TRUE at the same time.
SV	REAL	The set value of this input exists only if no parameter has been configured. The field “set value” in the parameter input field must be empty.

RES has priority over **SET** and **SET** has priority over **INH**.

Outputs

ENO BOOL According to IEC 1131.
OUT REAL Output.
Err BOOL TRUE = error in the integration.

Parameter definition



Time base
Second, minute, hour or day.

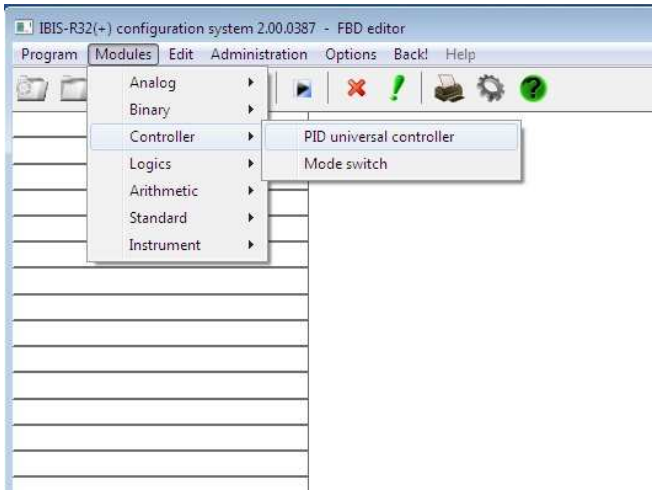
Creep
Quantities which are smaller than the default values. These are neglected during integration.

Set value
RES or **SET** set the output of the default value.
If the input **SV** is wired before the parameter input is called up, the parameter will not long be available for any input. In such case, the value of the input **SV** is valid. Vice versa, the input cannot be wired if a default value (even 0.0) has been entered.

Plausibility checks

none

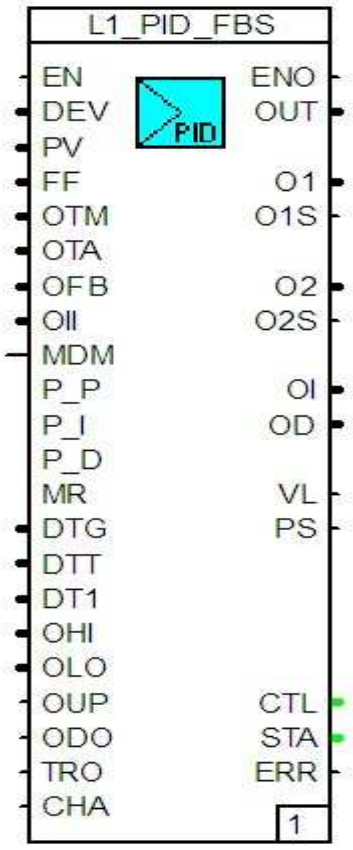
5.6.8 Modules, controller



Overview modules, controller

PID	PID universal controller
REGBA	Operating mode selector

PID universal controller, PID



Function:

This function module provides a universal PID controller with the most commonly encountered controller inputs.

The control deviation to be corrected is switched to the input **DEV** as a percentage signal. For the D component of the controller, the control variable can be switched to the input **PV** as a percentage signal. A percentage signal can be switched to the input **Z** for a disturbance variable feedforward.

In the operating mode MANUAL (M) the value at the input **OTM** is outputted at the outputs **OUT** as well as at **O1** and **O2**. The controller receives the information via the operating mode MANUAL by a TRUE signal at the input **MDM**. If the controller is not in the operating mode MANUAL (M) (**MDM**=FALSE), the value present at the input **OTA** is outputted at the outputs **O12** as well as at **O1** and **O2** provided there is a TRUE signal at the input **TRO**.

If the controller is operated as a step controller, then a position feedback signal can be fed in via the input **OFB**. This is outputted at the output **O12**.

The output value for the I component of the controller can be specified via **OII** at the first processing of the controller.

Provided that they have not been specified in the parameter definition, the controller parameters can be specified individually via the inputs **P_P**, **P_I**, **P_D** and **MR**.

A Smith predictor control can be switched on to control systems susceptible to dead time. Provided that they have not been specified in the parameter definition, the setting values necessary for this control can be specified via the inputs **DTG**, D**TT** and D**T1**.

The limits for the controlled variable within which the controller operates can be specified via the inputs **OHI** and **OLO**.

If the controller is operated as a discontinuous action controller, then the outputs **O1S** and **O2S** can be triggered in the operating mode MANUAL via the inputs **OUP** and **ODO.**

The rising or falling characteristic to be used by the controller for AUTOMATIC operation can be switched over via the input **CHA.** Increasing characteristic is used with **CHA**=FALSE and falling with **CHA**=TRUE.

The computed percentage signal of the controlled variable is outputted at the output **O12**. If only one continuous output is required for the controlled variable, this value is also available at the output **O1**.

If a controller has been configured with a discontinuous action output, then the switching state is outputted at the output **O1S**, with two switching outputs at the outputs **O1S** and **O2S**.

With split-range output configured, the two continuous signals are outputted at the outputs **O1** and **O2**.

The internal controlled variable components for the integral and differential components are outputted at the outputs **OI** and **OD**.

To be able to display and operate alarm values via the IND display loop, output **CTL** must be connected to input **CPI** of the functional module “display loop”.

Maximum change rates which are kept to by the controlled variable can be parameter defined in the controller. If the computed control variable violates these specified values, this is displayed at the output **VL** by a TRUE signal.

When using discontinuous action controllers, the controller can indicate at the output **PS** whether it is performing a positive or negative change of the controlled variable. This information can then be used for a controller parameter transfer.

The following error stati are outputted at the output **STA**:
0 No errors.
1 It is not possible to access the configured controller type during the processing.
2 The value for CP is smaller than 0.001.
3 The value for TR is smaller than 0.001.
4 The value for TD is smaller than 0.001.
5 The cycle time determined is erroneous.
The output **ERR** is set if the error status does not equal 0.

Parameter definition:
Parameters:
CP
Proportional component.
TR
Integral action time in [min].
TD
Derivative action time in [min].
VD1
Derivative action gain for output **Y1** and **Y1S**.
VD2
Derivative action gain for output **Y2** and **Y2S**.
Y0
Operating point in [min].

If a value has been specified in the entry fields, then the corresponding signal input cannot be connected.

Control:
P
The controller is operated as P controller.
PI
... as PI controller.
PD
... as PD controller.
PID
... as PID controller.
PI+Smith-Pr.
... as PI controller with Smith predictor.
PID+Smith-Pr.
... as PID controller with Smith predictor.

Differentiation:
Bipolar
The differentiation at the controller is used bipolarly.
Only positive
Only positive changes of control deviation and controlled variable are differentiated in the processing.
Only negative
Only negative changes of control deviation and controlled variable are differentiated in the processing.

D action of:
X
The D component is computed from the controlled variable.
XW
... from the control deviation.

Automatic characteristic:
direct
The AUTOMATIC characteristic is permanently set to direct (=rising).
inverse
... to inverse (=falling).

BE change over (static):
ACTIVE
The input **CHA** is used for characteristic transfer.
OUT
The automatic characteristic set in the parameter definition is used.

Manual characteristic:
direct The output characteristic for a controller with a controller output is set permanently to direct (rising) in MANUAL operating mode.
inverse The AUTOMATIC characteristic for a controller with a controller output is permanently set to inverse (falling) in the operating mode MANUAL.
inv.-dir. The output characteristic, for a controller with two controller outputs is (in the operating mode MANUAL) permanently set to direct (=rising) for **O1** and inverse (=falling) for **O2**.
dir.-dir. The output characteristic for a controller with two controller outputs is permanently set to direct (rising) for **O1** and **O2** in the operating mode MANUAL.
inv.-inv. The output characteristic for a controller with two controller outputs is permanently set to inverse (falling) for **O1** and **O2** in the operating mode MANUAL.
dir.-inv. The output characteristic for a controller with two controller outputs is (in the operating mode MANUAL) permanently set to inverse (falling) for **O1** and direct (rising) for **O2**.

Controller output:
Continuous The output **O12** is used for the output of the positioning signal.
Step The outputs **O1S** and **O2S** are ... Two-point The output **O1S** is ... Three-pointH-0-C(Z+Z) The outputs **O1S** (H) and **O2S** (C) are ... Three-pointH-0-C(K+Z) The outputs **O1** (H) and **O2S** (C) are ... Split-range(K+K) The outputs **O1** and **O2** are ...

Y limiting: YMIN Input of the lower output variable limit. internal The value set via Y-MIN is used for output variable limiting.
external The value defined via the input **OLO** is used for output variable limiting.
YMAX Input of the upper output variable. internal The value set via Y-MIN is used for output variable limiting.
external The value defined via the input **OHI** is used for output variable limiting.

Action:
Only in automatic
The output limits are only used in the operating mode AUTOMATIC (**MDM**=FALSE).
In manual and automatic
The output limits are always used.
ineffective The output limits are ineffective. Setting for the controller output “step”.

Controller output:
Dead zone[%]
Dead zone in the unit %.
T ON min.[step][s]
Minimum switch-on time of the controller output. Step in the unit “seconds”.
Operations/minZ1
Maximum number of switchings /minute of the output **O1S**.
Operations/minZ2
... the output **O2S**.
Y tracking:
Analog input
ACTIVE
The input **OTA** is used for tracking the output variable. Must be switched together with the binary input ACTIVE.
OFF
The tracking of the output variable is not used. Binary input
ACTIVE
The input **TRO** is used for tracking the control variable. Must be switched together with the analog input ACTIVE.

OFF
The tracking of the output variable is not used.

Y feedback signal:
ACTIVE
The input **OFB** is used as position feedback signal for a step controller.
OFF
The tracking of the position feedback signal is not used.

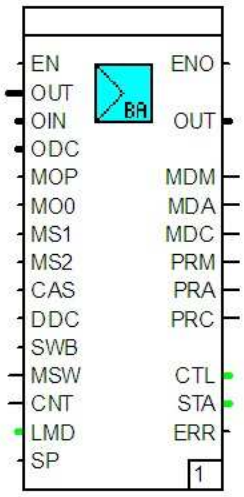
Disturbance variable Z+Y:
ACTIVE
The input **Z** is used for disturbance variable feed forward.
OFF
The disturbance variable feed forward is not used.

Smith predictor:
Controlled system gain
Amplification constant KS.
Dead time[min]
Controlled system dead time TT in [min].
Equivalent time constant[min]
Default time constant T1 in [min].

Rate limiter for controller outputs:
Limitation to negative changes
Only negative changes are checked and observed in regard to the rate of change.
Positive changes
Only positive changes ...
Neg. and pos. changes
Negative and positive changes ...

Limit values:
Negative change
Maximum value for negative output variable changes of the output variable in %/time unit.
Positive change
Maximum value for positive output variable changes ...
Time basis
Second
The alarm values have the time units second.
Minute
... the time unit minute.
Hour
... the time unit hour.
Day
... the time unit day.

Operating mode selector, REG_BA



Function:

The function module performs the operating mode transfers needed for a loop and provides the necessary positioning signal in MANUAL (M) operating mode for the controller at the output **OUT.**
The positioning signal defined from the controller operation must be fed into the input **OIN**.

The controller is switched to and fro between MANUAL operating mode (M) and AUTOMATIC (C) via the input **MOP.** The last output variable is retained as signal at the transfer to MANUAL. This signal is injected via the input **OUT**.

The controller is switched to the MANUAL operating mode via the inputs **MOP** as well as **MS1** and **MS2**. At the transfer to MANUAL, the positioning signal is set to 0 % at **MO0**. When transferring via the inputs **MS1** or **MS2**, the associated parameter definable safety positioning value 1 or 2 is outputted as positioning signal.

The transfer between the AUTOMATIC (A) and CASCADE (C) operating modes is performed via the input **CAS**. For the inputs **MOP**, **MO0**, **MS1**, **MS2** and **CAS** it can be stipulated by parameter definition whether the particular switching function is performed by the level of the input signal (statically) or by the particular rising flank (dynamically).

It is possible with a TRUE signal for the associated loop to be transferred to DDC control via the input **DDC**. During DDC control, the value present at the input **ODC** is outputted as positioning signal.

Any signal can be fed in via the input **SWB**, in order to transfer the controller to MANUAL operating mode. A transfer via the device front or communication to another operating mode is impossible as long as a TRUE signal is present at the input.

The particular operating mode of the loop is displayed via the outputs **MDM** (MANUAL), **MDA** (AUTOMATIC) and **MDC** (CASCADE). Because the operating mode transfer is not performed immediately after the key is pressed, information as to which operating mode the transfer will shortly be made is provided via the outputs **PRM** (MANUAL), **PRA** (AUTOMATIC) and **PRC** (CASCADE). This transfer to the next operating mode is selected by a rising flank at the input **MSW**. The transfer to the final operating mode at the outputs **MDM** to **MDC** is then performed with the next falling flank at the input **CNT**. During this time the future operating mode is displayed at the outputs **PRM** to **PRC**.

The input **LMD** is used to inject the power failure protected operating mode of the loop. Depending on its parameter-defined behaviour, the function module can either inject the operating mode present at the power failure or again the MANUAL operating mode.

The starting of the parameter selftune function is communicated to the module via the input **SP**. This causes automatic transfer to the MANUAL operating mode.

The output **CTL** must be for display and operation via the IND display loop to the input **CMD** of the function block "display loop" connected.

Parameter definition:

Control function:
With out controller
The loop is used without controller.
1-channel controller
... as single-channel controller.
Master controller cascade
... as master controller of a cascade control.
Slave controller cascade
... as slave controller of a cascade control.
Main controller override minimum
... as main controller of an override control in minimum selection.
Main controller override maximum
... as main controller of an override control in maximum selection.
Limit controller override minimum
... as override controller of an override control in minimum selection.
Limit controller override maximum
... as override controller of an override control in maximum selection.
1-channel manual station
... as manual station.
1-channel set point station
... as setpoint station.
1-channel ratio station
... as ratio station.
1-channel positioner
... as positioner.
Ratio station cascade
... as ratio station in a cascade control.

Operating mode:
Manual/automatic
The operating mode can only be transferred between MANUAL and AUTOMATIC.
Manual only
Only MANUAL is possible as operating mode.
Automatic only
Only AUTOMATIC is possible as operating mode.
Only automatic/manual: Ymin
Not yet implemented.
Only automatic/manual: Ymax
Not yet implemented.
Manual/automatic/cascade
The operating mode can be transferred between MANUAL, AUTOMATIC and CASCADE.
Manual/automatic/DDC
The operating mode can be transferred between MANUAL, AUTOMATIC, CASCADE and DDC.

Operating mode after power failure:
Last operating mode
If the device is connected to the power supply, the loop continues to operate in the last operating mode in use at the supply voltage interruption.
Manual: last output
..., the loop continues to operate in MANUAL with the last positioning signal.
Manual: output=0
..., the loop continues to operate in MANUAL with 0 % as positioning signal.
Manual: output=YS1
..., the loop continues to operate in MANUAL with the safety output 1 as positioning signal.
Manual: output=YS2
..., the loop continues to operate in MANUAL with the safety output 2 as positioning signal.

Operating mode after fault at input:
No change
If a fault is indicated at the input **SWB,** the loop continues to operate in the current operating mode.
Manual: last output
..., the loop continues to operate in MANUAL with the l last positioning signal.
Manual: output=0
..., the loop continues to operate in MANUAL with 0 % as positioning signal.
Manual: output=YS1
..., the loop continues to operate in MANUAL with the safety output 1 as positioning signal.
Manual: output=YS2
..., the loop continues to operate in MANUAL with the safety output 2 as positioning signal.

DDC function:
DDCOFF
The DDC control is not used.

Operating mode after computer failure:
Manual: last output
If the supervisory computer fails, the loop operates in MANUAL operating mode with the last output as positioning signal.
Manual: output=0
... in MANUAL operating mode with 0% as positioning signal.
Manual: output=YS1
... in MANUAL operating mode with the safety output 1 as positioning signal.
Manual: output=YS2
... in MANUAL operating mode with the safety output 2 as positioning signal.
Automatic:last output
... in AUTOMATIC operating mode with the last output as first positioning signal.
Cascade
... in CASCADE operating mode.

Binary input for changeover to manual...:

...with last output:

ACTIVE

The transfer via the input **MOP** is used.

OFF

... is not used

static

The required function is implemented by a TRUE signal at the input. A FALSE signal causes switching to AUTOMATIC

dynamic

The required function is implemented by a flank change at the input. The next flank change switches to AUTOMATIC.

...with output = 0

ACTIVE

The transfer via the input **MO0** is used. OFF ... is not used.

Static

The required function is implemented by a TRUE signal at the input. A FALSE signal causes switching back to the previous operating mode.

dynamic

The required function is implemented by a flank change at the input. A transfer back to another operating mode is not implemented.

...with safety output 1:

YS1 Safety output 1 in the unit % in the range 0.0 ... 100.0.

ACTIVE

The transfer via the input **MS1** is used. OFF ... is not used.

static

The required function is implemented by a TRUE signal at the input. A FALSE signal causes switching back to the previous operating mode.

dynamic

The required function is implemented by a flank change at the input. A transfer back to another operating mode is not implemented.

...with safety output 2:

YS2 Safety output 2 in the unit % in the range 0.0 ... 100.0.

ACTIVE

The transfer via the input **MS2** is used. OFF ... is not used.

static

The required function is implemented by a TRUE signal at the input. A FALSE signal causes switching back to the previous operating mode.

dynamic

The required function is implemented by a flank change at the input. A transfer back to another operating mode is not implemented.

Binary input for changeover to cascade:

ACTIVE

The transfer via the input **CAS** is used.

OFF

... is not used.

static

The required function is implemented by a TRUE signal at the input, provided that the controller is not in MANUAL operating mode. A FALSE signal causes it to switch to AUTOMATIC.

dynamic

The required function is implemented by a flank change at the input, provided that the controller is not in MANUAL operating mode. The next flank change switches to AUTOMATIC.

Binary input for computer standby in case of DDC:

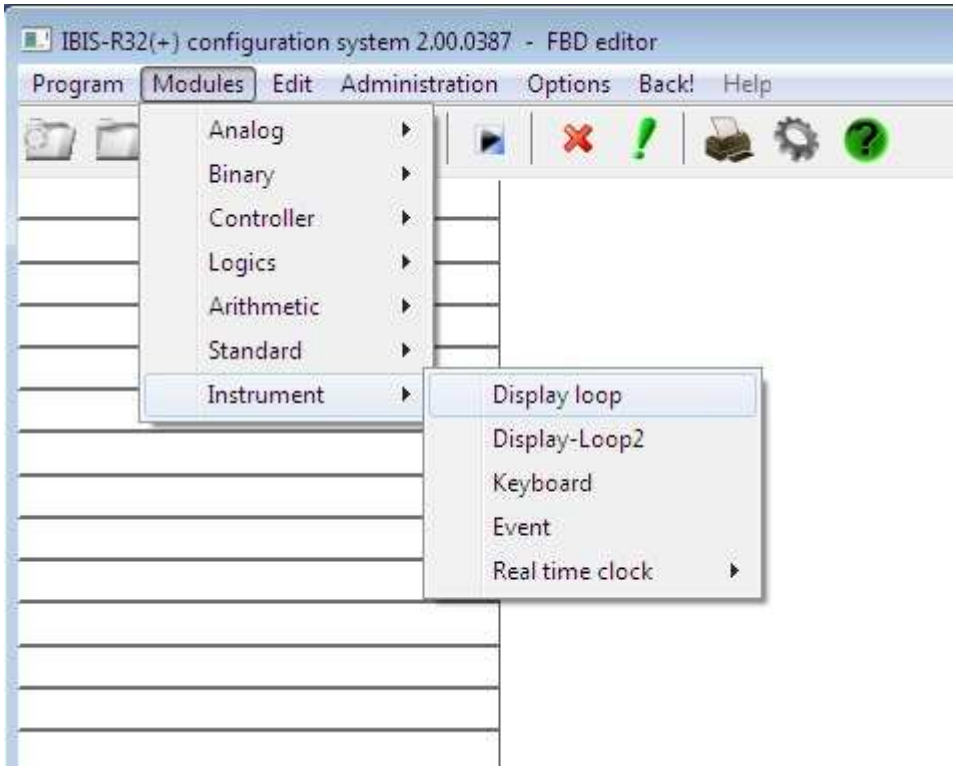
Computer always ready

DDC control is always performed, the input **DDC** is not used.

Computer keep alive

The DDC control is used dependently on the input **DDC**.

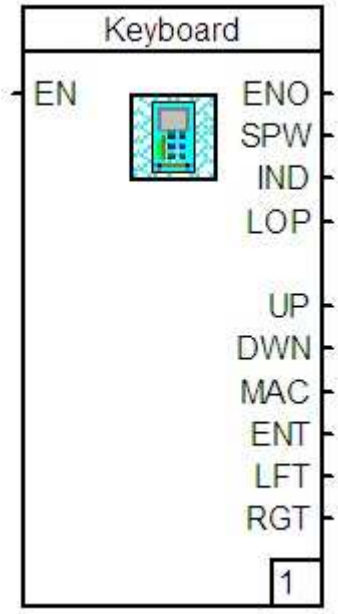
5.6.9 Modules, model



Overview modules model

ANZSL	Display loop
KEYB	Keyboard
MELD	Event

Keyboard, KEYB



Function:

The functional module informs at the outputs **SPW** to **RGT** if keys are pressed on the device front panel.

The use of this functional module has no influence on the meaning of this key during operation.

The output behaviour -TRUE at the output -involving the output pulse can be configured. FALSE means that the associated key has not been pressed.

The outputs are assigned to the following keys (colour is for Protrenic 500/550 700):

Output Button

SPW	<SP-w>	(green)
IND	<Ind>	(grey)
LOP	<Loop>	(grey)
UP	<▲>	(green)
DWN	<▼>	(green)
MAC	<MAC>	(yellow)
ENT	<Enter>	(grey)
LFT	<◄>	(yellow)
RGT	<►>	(yellow)

The key <Esc/Menu> is disabled.

The output behaviour which can be stated by default under “behaviour for continuous signal” is valid for all outputs for which a continuous signal has been configured.

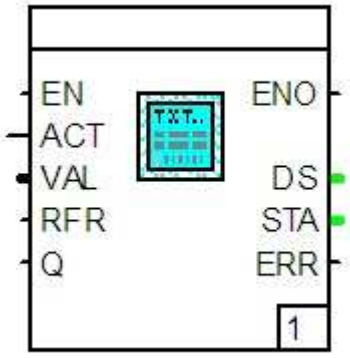
Parameter definition:

Permanentsignal forpressure
- [ ] produces a TRUE signal for a processing cycle at the corresponding output.
- [X] produces a pulse corresponding to the defined type in the parameter definition “behaviour for continuous signal”.

Behaviour for continuous signal
Continuous signals as pulses
- [X] gives a pulse sequence with a frequency of 1 Hz. The duration of the FALSE signal at the output can be input via “pulse pause per s”.
- [ ] produces TRUE at the output for the entire duration of the pressing of the key.

Pulse duration per s
Default statement of the duration of FALSE for the pulse sequence for continuous signal.

Text message, MELD



Function:

The functional module generates a text message in the text line of the controller. The text message, predetermined by means of configuration, is triggered via a TRUE or FALSE signal at input **ACT**. With the text line a value from the data type REAL can be provided. As long as this value has not been parameterized as a constant value, it will be fed to the input **VAL**.

Already acknowledged text messages can be redisplayed through the input **RFR**.

If configured, text messages can be acknowledged or cancelled via the input **Q**.

Status information on the message process is output via **DS**. The following states are possible:

0 No requirement for next message
1 Message is available but not acknowledged
2 Message is available and is acknowledged

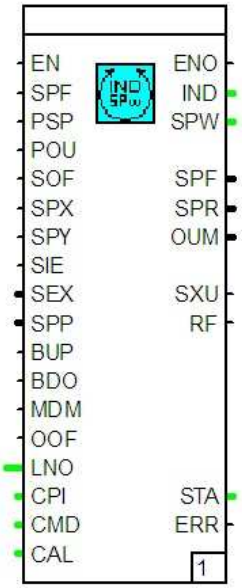
The following error states are output at **STA**:
0 No error
1 Message cannot be transmitted on request

The output **ERR** is set if the error status does not equal 0.

Parameter definition:

Message text
Text to be displayed on the front panel.
Message as alarm
[ ] displays message text as a simple text message.
[X] takes over the message text in the alarm management and treats it like an alarm message.
Name
Three letter short text which stands left of the message value on the front panel during display of the value. If the no input is made, no text message will be output.
Message value
→<Ind> leads to the display of the message value. If no value is input here, the value of the input VAL will be used on the appearance of the text message. If the input VAL is wired, an input in this field will not be possible.
Format
The number of decimal places to be used for the display of the value. The input of the figure 5 corresponds to a sliding point format.
Dimension
4-letter text on the front panel and located to the right of the value during display of the message value. If “USER” is input there, the specifiable text under “Userdim.” is used for display.
Userdim. 4-letter text on the front panel and located to the right of the value during display of the message value.
Message action
Message at logical 1:
The text message is output with a TRUE signal at input **ACT**. Message at logical 0: The text message is output with a FALSE signal at input **ACT**.
Delete action
via binary input: Deletion of the text message is effected via a TRUE signal at input **Q**. not deletable as long as active: Deletion of the text message is effected only after the condition of the message action is no longer fulfilled.
Automatic to [ ]
Deletion of the text message is effected automatically according to the set time if the condition of the message action is no longer fulfilled.

Display loop, ANZSL



Function:

The functional module controls the display and operation of the IND display loop. Further, it provides the processing phase with the value of the adjusted active set point source or the set ratio. Apart from the elements of the IND display loop provided in default, up to 8 free variables of the data type REAL and 2 variables of the data type TIME (type of DINT) can be displayed and operated, if need be, per each control loop.

The next position of the IND display loop is selected via a positive flank at input **SPF**. The position of the valid set point or the output variable can be selected via a TRUE signal at the inputs **PSP** and **POU**.

The possibility of adjusting the set points can be inhibited via TRUE at input **SOF**.

Inputs **SPX** and **SPY** are used for the adjustment of the up to four internal set points. If only **SPX** is used, changeover takes place between **SP1** and **SP2**.

A TRUE signal at input **SIE** changes over from the internal to the external set point. When this happens, the fed value supplied via input **SEX** is used as an external set point. The use of the active set point is displayed by the functional module via TRUE at output **SXU**.

The input **SPP** is used to feed the specified set point of the programmer.

The inputs **BUP** and **BDO** are used (depending on the parameter setting) for the remote adjustment of the set point or output variable. At a constant pulse, the speed amounts to approximately 100 %/min.

The operation mode MANUAL of the control loop is displayed to the functional module via TRUE at input **MDM**.

The adjustment of the output variable can be inhibited via TRUE at input **OOF**.

The control loop currently in the display is switched as a numeral to the input **LNO**.

For display and operation this module requires information from the functional modules PID universal controller (PID), mode selector switch (REGBA) and alarm values (AL4). To gain access to them, a connection must be established to the **CTL** outputs of the modules via the inputs **CPI** (for PID), **CMD** (for REGBA) and **CAL** (for AL4).

The output **IND** shows which input of the IND display loop is displayed on the front panel. The output **SPW** shows the index of the current active set point source (1 = SP1,
2 = SP2/SR1, 3 = SP3/SR2, 4 = SP4, SR3, 5=SPext,
6=SP computer,7=SP program).
The effective set point is displayed at output **SWF**, the effective set point ratio is displayed at output **SPR**.

In the operation mode MANUAL the manual correction value is outputted at output **OUM**.

The following error states are output at **STA**:
0 No error
1 No valid control loop number
2 No new position of the IND display loop found
3 Access to set point information not possible
4 Set input circuit is invalid
5 Invalid value at inputs CPI, CMD or CAL during initialization
6 Invalid value at inputs CPI, CMD or CAL during cyclical processing

The output **ERR** is set if the error status does not equal 0.

Parameter definition:

Inputcircuit
Setting the used input circuit.

LoopNo.
Number of the control loop in which this functional module operates.

Adjustment of alarm values
For each of the 4 alarm values it can be determined if
- the value can be displayed and adjusted during display at the operator level (IND display loop)
- the value can only be displayed at the operator level and adjusted at the parameter level or
- cannot be displayed at the operator level but only at the parameter level and can only be adjusted there.

Dimension SP
4-letter text located on the front to the right of the value during display of the set points. If “USER” is input there, the specifiable text under “USER:” shall be used for display.

Decimal places SP
The number of decimal places to be used during the display of the set points. Apart from the fixed point, the sliding point display can also be selected.

Display Err
To display a control deviation, one can select between a display in % and in physical units (EU).

Dimension R
Choice between no statement of dimenion, % dimension and a user-defined dimension for the display of nominal and actual ratios. During display of the user-defined display of ratios, the specifiable text for display under “USER:” can be used.

Decimal places R
The number of decimal places to be used during display of the ratios. Apart from the fixed point, a sliding point display can also be selected.

Display R
When using ratio control, a differentiation can be made in the digital display between display of the nominal/actual ratio and the norminal/actual value in physical units.

Release of remote adjustment
inhibited:
No remote adjustment can take place. only Y (in manual mode): In the operation mode HAND the output variable can be remotely adjusted via the inputs **BUP**, **BDO**. Only W (all operation modes): In the operation mode AUTOMATIC the output variable is remotely adjusted via the inputs **BUP**, **BDO**. W (in auto mode), Y (in manual mode): In the operation mode MANUAL the output variable and in AUTOMATIC mode the set point can be remotely adjusted via inputs **BUP**, **BDO** respectively.

Set point limits
SP1-min. Lower set point limit.
SP1-max. Upper set point limit.
R min. Lower limit of the nominal value ratio.
R max. Upper limit of the nominal value ratio.

Set point changeover with BI
OFF Changeover of the internal set point via **SPX** and **SPY** is not used.
SP1-SP2
BIx Changeover takes only place between SP1 and SP2 via **SPX**.
SP1-SP4
BIy: Changeover takes place between SP1, SP2, SP3 and SP4 via **SPX** and **SPY**.

SP int/ext with BI
ACTIVE The input **SIE** is used to change over between internal and external set points. **SIE** = TRUE switches to external set point.
OFF The changeover between internal and external set points is not used.

SP-in hi bit with BI
ACTIVE The input **SOF** is used to inhibit set point ad justment. **SOF** = FALSE means release of ad justment.
OFF The inhibit mode of the set point adjustment is not utilized.

SP-Tracking with MANUAL
OFF The effective set point of the control variable is not tracked in the MANUAL operation mode.
ON The effective set point of the control variable is tracked in the MANUAL operation mode.

DDC: Set point in case of computer failure
SP-Actual The adjusted set point is used as effective set point in case of computer failure.
SP-comp. The last computer set point is used as effective set point in case of computer failure.
PV-Actual The current controlled variable is used as effective set point in case of computer failure.

Setpoint1
SP1 Parameter value of the first set point SP1.
OFF Set point SP1 is not used. ON Set point SP1 is used.
Follows act. SP Set point SP1 is used and is tracked to this value by using a different set point source.

Type SP1
No parameter:
The value for set point W1 is not set via the parameter value but via the local operation.
Parameter:
The value for set point W1 is only set via the parameter value.

Setpoint2 / R1
SP2 Parameter value of the second set point SP2 or ratio set point R1
OFF SP2 / R1 is not used.
ON SP2 / R1 is used.
Parameter The value for SP2 / R1 is only set via the parameter value.

Delta Parameter The value is added to the new set point SP1 as delta.
R1 follows act. ratio By using a different source for the set point ratio during ratio control, R1 is tracked to the current ratio

Setpoint3 / R2
SP3 Parameter value of the third set point SP3 or ratio set point R2.
OFF SP3 / R2 is not used.
ON SP3 / R2 is used.
Parameter The value for SP3 / R2 is only set via the parameter value.
Delta Parameter The value is added to the new set point SP1 as delta.

Setpoint4 / R3
SP4 Parameter value of the third set point W4 or ratio set point R3.
OFF SP4 / R3 is not used.
ON SP4 / R3 is used.
Parameter The value for SP4 / R3 is only set via the parameter value.
Delta Parameter The value is added to the new set point SP1 as delta.

SP-external
OFF No external set point is used.
ON An external set point is used.

SP-computer
OFF The set point (via interface) of a higher level computer is not used.
ON The set point (via interface) of a higher level computer is used.

SP-programmer

OFF The programmer is not used as set point source.
ON The programmer is used as set point source.

Globall variables in the display loop

The variables .Lx_R1 to .Lx_R8 and .Lx_T1 and .Lx_T2 can be taken over into the display loop (x corresponds to the number of the control loop) by marking with a cross.

Name

3-letter short text located to the left of the value on the front panel during display of the value.

Dimension

4-letter text on the front panel, located to the right of the value during display of the value. If “USER” has been input there, the specifiable text for display shall be used under “USER:”.

“User”

4-letter text displayed as user-defined dimension, located to the right of the value on the front panel.

K Number of post-decimal points to be used for the display of the value. Stating 5 is equivalent to a sliding point format.

V

[ ] Variable value is only displayed and cannot be processed.
[X] Variable value is displayed and can be processed.

5.6.10 Selecting the modules and positioning in the program

with
→Modules
→select desired module type
→move to the desired point in the graphics area using the mouse
→enter with mouseclick (→with modules having changeable numbers of inputs, now define the size by pulling the mouse vertically→confirm with mouseclick)
→either position the next module of this type or terminate the entry with right-mouseclick
→ending positioning: at any time using <Esc> or right mouseclick

or via keyboard:
→<Alt>
→<M>
→mark the desired module in the menu with <↑>, <←>, <↓>, <→> or via the speed keys
→<Enter>
→position the module in the program with <↑>, <←>, <↓>, <→>
→<Space>

After a module has been selected it is positioned in the graphics area. The module is displayed graphically during this.

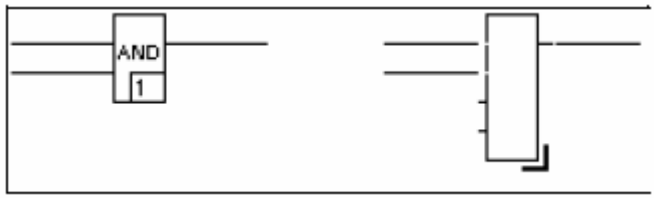
After entering, renewed frame presentation signals that a further module of the same type can now be entered.

Modules with changeable numbers of inputs (such as AND, OR, EXOR) are displayed in minimum size in the positioning. Their size can be changed immediately after placing. Further inputs become visible by pulling the mouse vertically.

The new module has the lowest, not yet assigned execution number within the program. Parameterizable modules have a parameter definition display with presettings, but still no module names.

The module display must not be overlapped with other program elements. It is essential to maintain the minimum distance of three grid points to connections, and two grid points vertically to other modules.

5.6.11 Changing number of inputs



Note
The selected module must be changeable in its number of inputs.

with
→selct module
→Process
→Change number of inputs
or
→doubleclick on the lower demarcation line of the module
→move the mouse up or down until the desired number of inputs is displayed
→[OK]
→ending positioning: at any time using <Esc> or right mouse click

or via keyboard:
→select module
→<Alt>
→<I>
→<N> or
→press <Enter> on the lower demarcation line
→move the cursor up or down with <↑>, <←>, <↓>, <→> far enough for the desired number of inputs to be displayed.
→<Space>

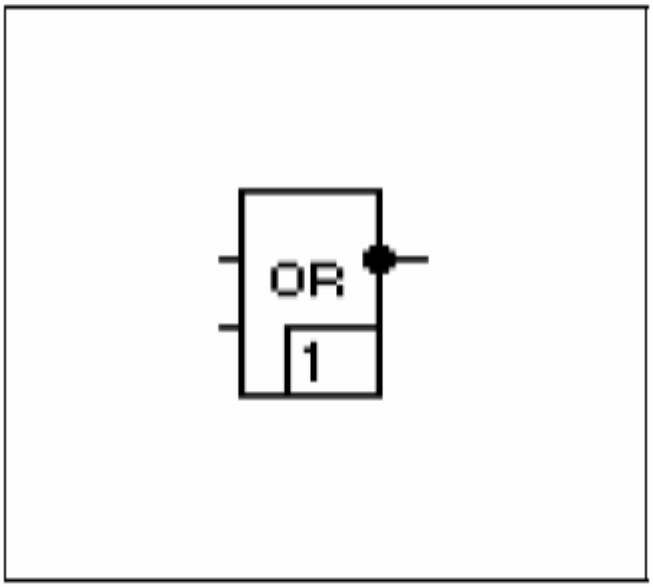
The number of input connections of the function module is changed.

The connecting terminals of the function module already connected are permanently positioned and are not moved when the number of inputs is changed. The number of inputs can thus be changed without affecting connecting terminals already connected.

If the procedure is aborted, the module retains its old status.

If a module is to be reduced by inputs which are already connected but superfluous, the signal flow lines belonging to these inputs must first be disconnected from the module.

5.6.12 Inverting a module connection terminal



(Module with negated connection)

Note
The module connection terminal to be inverted must be of signal type BOOL (binary).

with
→hold <Ctrl> pressed and click the connection terminal of the module to be inverted

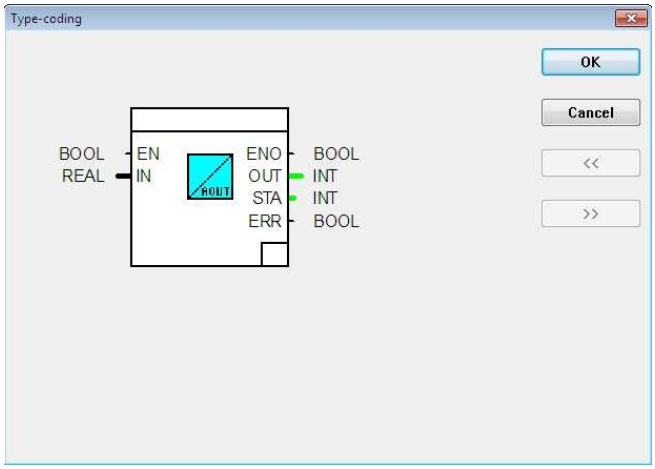
or via the keyboard

→position cursor on the connection terminal of the module to be inverted
→press and hold <Ctrl>
→<Space>
→<Space>

Negation is set or unset for the connection selected. Added inversion markings are treated as a component of the function module.

All modules have non-negated connection terminals preset.

5.6.13 Display and change signal types



wiht
→select module
→Process
→set and accept the desired signal type with [<<] or [>>]

The signal types of the module connection terminals are displayed textually and graphically. In changing, the display is adapted to the new signal types. The display of connected signal flow lines changes correspondingly.

The signal types of the module selected can then only be changed if the module permits other signal types. They can only be changed identically for all connection terminals. Irrespective of this, certain signal types can also be converted by using the converter modules "*_to*" and "Trunc".

5.6.14 Defining parameters of function modules

Parameter types

The items of information which are needed for the processing and display of a module are defined here as parameters. It is necessary to distinguish between the following types:

"Must" Parameter
are necessary items of information such as the module name and (depending on module type) the parameters of particular inputs or outputs. These parameters are displayed in red in the standard colours setting.

"Can" Parameter
are not absolutely essential items of information such as short text, long text, dimension, conductivity and alarm values. These are always occupied with default values when the function module is positioned for the first time.

External parameters
are transferred to a module (and vice versa) by the connection of a signal flow line.

Internal parameters
must be entered within a parameter display. They include information items such as module name and alarm values.

Calling the parameter displays

with
→select function module to have parameters defined
→*Process*
→*Define parameters*
→doubleclick on the function module

It changes into the first parameter display of a function module. All other elements selected are then automatically deselected.

After returning from the parameter definition display, the function module of the changed parameter definition is re-displayed correspondingly.

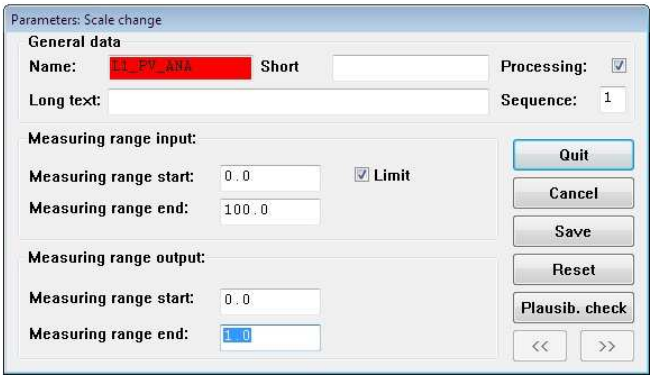
Entering the “must” parameters

To be able to close a function module language program correctly, it is necessary to quote the “must” parameters of the individual function modules of this program. This is normally only the module name (max. 12 characters) of a function module.

All module names for function modules entered are brought together system wide in the loop tag administration.

Alternative entry option for the module name:
→Text field name: select
→<F2>
→select module name from the loop tag administration

Handling the parameter definition displays



There is no unified parameter-setting display, because of the differing parameters of the individual function modules. However, certain components are used equally in all or certain parameter definition displays. With extensive function modules there are also several parameter definition displays which can be edited in any sequence.

The basic components are explained below on the first two parameter definition displays of the function module “L1_PV_ANA”.

Title line

Name, short designation of the module, (number of the parameter definition display in which processing is just taking place).

Group

Certain parameters are have been brought together in groups, such as scaling for measuring range input and output. These parameters are framed, and a group name reproduces the parameter function in the frame top edge.

Colour

Entry field with red background: “must” parameters.
Entry field with blue background: marked for overwriting.

Text field

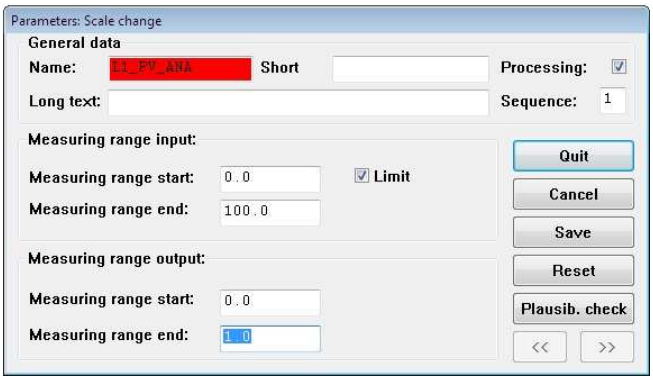
For example for the entry of module name and long text. If the cursor is moved to a text field using <Tab>, then it is marked for overwriting (if using a mouse, mark by double-click).

Alternative entry option for module names:
The module name can also be selected from the loop tag ad ministration via the function key <F2>.

The “can” parameters short and long texts can only be entered after a module name has been assigned.

Data field

For example for entering parameters such as the lower and upper range values. With parameters which can also be defined externally, data entry is only possible provided no signal flow line has been connected to the associated connection terminal. Vice versa, the connection terminal disappears from the module display if a parameter has been entered. The parameter to which this applies must be found from the module description.

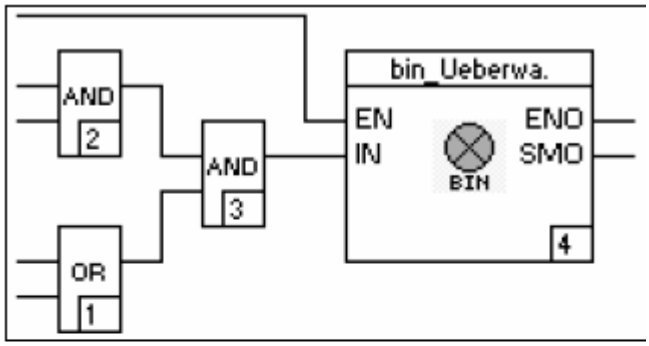


→[Quit]	exits the active parameter definition window and saves the parameter definition status.
→[Cancel]	exits the active parameter definition window without saving the parameter definition status. A warning appears if parameter definition data might get lost.
→[Save]	saves the parameter definition status.
→[Reset]	resets the parameter definition status of the active parameter definition window to the default values. A parameter definition previously saved and differing from the default values can be called back by →[Cancel] and re-calling the parameter definition display.
→[Plausib. check]	checks plausibility of the function module with the current parameter definition.
→[<<], →[>>]	changes into the previous/next definition display

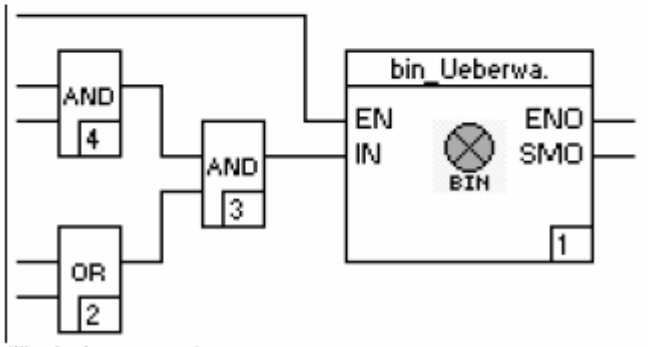
The entry of short and long texts of the module is recommended to increase the clarity, entry only being possible after a name has been assigned to the module.

An input or output of a module which is connected to a signal flow line cannot be assigned internal parameters and vice versa.

5.6.15 Changing execution sequence of the modules



(Logical sequence)



(Illogical sequence)

with
→select module
→Process
→Executionsequence
→enter new execution number in the module (the old one is marked for overwriting)
→<Enter>

The execution sequence can also be changed on modules which have a parameter definition display.

→press and hold <Ctrl>
→click with mouse on the execution number lower right in the module
→<Enter>

or via keyboard

→select module
→<Alt>
→<S>
→enter new execution number
→<Enter>

The unambiguous sequence in which the modules of the program are processed in program execution is changed.

The processed module contains the newly entered execution number. On all other modules of the program, the execution number is corrected in such a way that their sequence to one another is retained, and no gaps occur in the sequence. If a number is entered which exceeds the total number of modules used in the program, then the module processed is given the total number as execution number.

The execution number is assigned automatically in the chronological sequence of the positioning of the modules.

Because the modules are not generally placed in the sequence in the program in which they are intended to be executed in operation, it is wise to check the execution numbers after all modules have been connected, and to change the sequence if necessary.

5.7 Connecting program elements

The signal flow lines between the modules and the variables of the inputs and output strips is produced by entering signal flow lines in the graphics area. Each signal flow line transports data of precisely one signal type. This is derived from the signal types of the connections when inserting the signal flow line.

5.7.1 Display of the signal flow lines

If the signal flow line has the processing status incorrectly or is not connected, this is displayed. Otherwise it shows the signal type transported.

The status or the transported signal type of the signal flow line can be recognised from the line width and colour, the user being able to set any colour (see Section 5.3.4 “Options of the project tree Colours”).

The table below shows the connection between signal type, processing status, line width and the preset colour:

Signal type/ Processing status	Colour	Display
BOOL	black	narrow
DINT	dark green	wide
INT	light green	wide
REAL	black	wide
Error status	red	narrow
Selected	blue	wide
Not connected	black	narrow

5.7.2 Entering a signal flow line into the program

→press and hold <Ctrl>
→position mouse on a connection or an existing signal flow line
→press and hold down left mouse key and guide it to the objective
→release mouse key

If the connection terminals need not be connected by a direct signal flow line, they can be entered in as many part sections as you like. Process as with an individual line. <Ctrl> needs not to be released between the sections.

or via the keyboard

→position cursor on a connection or an existing signal flow line with <↑>, <←>, <↓>, <→>
→press and hold <Ctrl>
→guide cursor to the objective
→release <Ctrl>

When entering part sections, release <Ctrl> between the sections.

A new signal flow line is inserted into the program and displayed according to its status and the signal types connected.

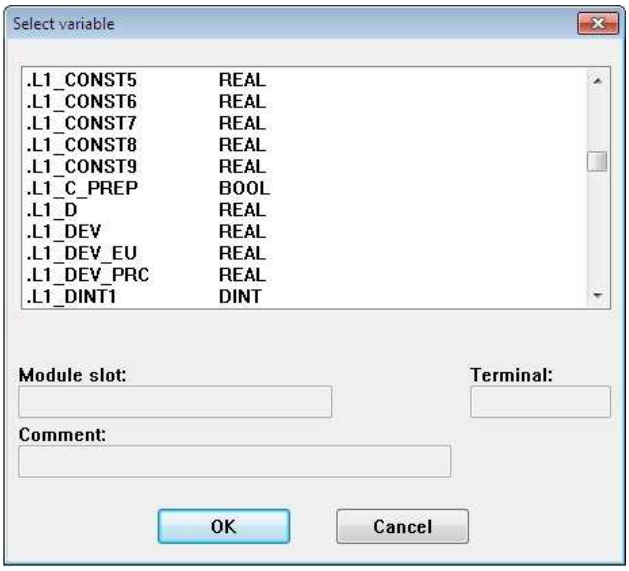
The signal flow line is displayed as incorrect (error status) if the signal types of the connections do not agree, or they connected to one another, or a short circuit is produced.

The signal flow lines may cross one another, but not be superimposed. Superimposed sections are deleted.

Vertical line segments are only possible at a minimum distance of two grid points from input and output strip and module connections.

5.8 Entering and changing variables in the input and output strips

5.8.1 Entering variables



→move to the desired line of the input or output strip
→enter name of variable
→<Enter>
→select the signal type in the “Declare variable” window
→select desired line of the input or output strip
→<F2>
→in the window “Select variable” choose one of the variables previously existing in the project

or via keyboard:

→position the cursor on the desired line of the input or output strip with <Esc>, <Tab> and <↑>,<←>,<↓>,<→>
→enter name
→<Enter>
→select the signal type in the window “Declare variable”.

IIf a variable is being used for the first time in the project, it is transferred automatically into the system wide variables list.

A connection is assigned in the graphics area to each variable which is entered into the input or output strip. It is possible to draw a signal flow line from this connection to further elements in the graphics area of the editor.

Entries are omitted in the “Declare variable” window where the variable already existed in the project.

The multiple entry of the same variable in the output strip produces a warning, although it is permissible.

5.8.2 Change variables

→doubleclick on the variable to be changed
→the variable can be marked for overwriting by another double-click
→change name of variable
→<Enter>
→select the signal type in the “Declare variable” window
Entries are omitted in the window where the variable already existed in the project.
→doubleclick on the variable to be changed
→<F2>
→select one of the variables previously existing in the “Select variable” window

or via keyboard

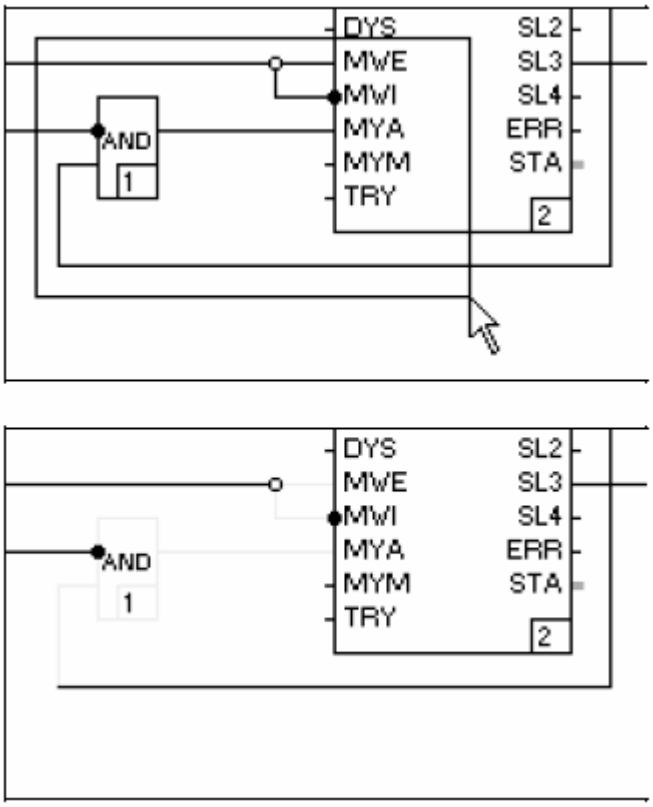
→position the cursor on the variable to be changed with <Tab> and <↑>, <←>, <↓>, <→>
→<Enter>
→the cursor can be marked for overwriting with <Shift + End>
→change name of variable
→<Enter>
→“Declare variable” window

The new name of variable is transferred into the program and into the variables list. The old variable still remains in the variables list.

If the changed variable has been used in several programs of the project, they remain unaffected.

5.9 Processing program elements

5.9.1 Selecting program elements



Select individual program elements

→select desired program element with mouseclick

The entire surface of the program element is regarded as selection field. The line is regarded as selection surface with the variables of the input and output strips.

or via keyboard:

→position the cursor on the program element with <↑>, <←>, <↓>, <→>
→<Space>
(change between graphics area and input and output strips with
→<Tab>)

The program element is selected for further processing and displayed accordingly.

The non-selected status is preset.

Inversions and connection points of signal flow lines are never displayed as selected.

several program elements simultaneously

→draw frame around the elements to be selected (→move mouse to the corner of the frame→press and hold left mouse-key→pull up frame→release mouse key when the frame has the desired size)

or via keyboard

→move the cursor to any corner of the frame to be pulled up with <↑>, <←>, <↓>, <→>
→press and hold <Space>
→move the cursor to the opposite corner with <↑>, <←>, <↓>, <→>
→release <Space> at the desired position

All elements which the frame completely encompasses are selected simultaneously and displayed accordingly. With signal flow lines this applies to all sections which are lying completely in the frame. After making the selection, the desired operations can be performed -as with individual elements. For example: →Process
→Cut.

Select additional program elements
→press and hold <Shift>
→select further element
.
or
.
→move cursor to the element to be selected
→<Shift + Space>: release <Space> before <Shift>!

An element is additionally selected for the selection already existing and is displayed accordingly.

It is also possible to select several elements by <Shift> and stretching the frame.

5.9.2 Deselecting program elements

Deselect all selected program elements

→click on a free point of the graphics areas, or select a non--selected element

or via keyboard
→move the cursor to a free point of the editor with <↑>, <←>, <↓>, <→>
→<Space>

The program elements are deselected and displayed appropriately.

A selection is automatically cancelled by opening another window.

Deselect program elements of a selection

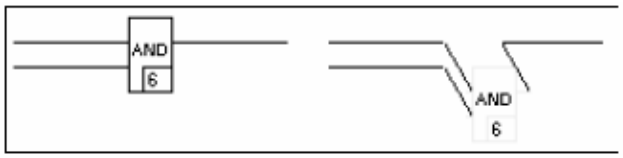
→press and hold <Shift>
→click on element to be deselected

or via keyboard

→move the cursor onto the element to be deselected with <↑>, <←>, <↓>, <→>
→<Shift + Space>

An element of the selection already existing is deselected and displayed accordingly.

5.9.3 Moving program elements



→select elements to be moved
→position cursor on one of the selected elements
→press and hold left mouse key
→pull elements into the desired position
→release mouse key

or via keyboard:
.
→select elements to be moved
→position cursor on one of the selected elements with <↑>, <←>, <↓>, <→>
→<Space>
→move elements to the desired position with <↑>, <←>, <↓>, <→>
→<Space>

The selected elements are moved to a new position. The contours of the elements remain visible during this move. All signal flow lines concerned are interrupted by the moving and must be corrected afterwards! Up to that point they are displayed as “elastic band”. Modules with “must” connections which are now unsupplied are given the status incorrectly. Parameters already configured are retained.

5.9.4 Copying, cutting, inserting, deleting program elements

Copy

→select element(s) to be copied
→Process
→Copy
→Process
→Insert
→press and hold left mouse key
→pull module to the desired position
→release mouse key

If elements are to be copied repeatedly, this is possible by repeating the points from →Processing→Insert.

Alternatively:
→
→<Ctrl + Ins>
→<Shift + Ins>
→press and hold left mouse key
→pull module to the desired position
→release mouse key

or via keyboard:

→select element(s) to be copied
→<Ctrl + Ins>
→<Shift + Ins>
→<Space>
→position element(s)
→<Space>

If elements are to be copied repeatedly, this is possible by repeating the points from <Shift + Insert>.

The program elements selected are duplicated and inserted at any position you like in the program. The contours of the elements remain visible during moving.

Copied modules are given new execution numbers and the status “incorrect”. Their parameter information is copied with them. If several modules are copied simultaneously, their execution sequence amongst themselves is retained.

Cutting

→select program element(s)
→*Process*
→*Cut*

The elements selected are removed from the program and unlike deleted elements they can be accepted back into the program again by insertion.

To insert elements that have been cut, their former position is preset.

Only the last one cut can be re-inserted.

Inserting

A program element must have been copied or cut since calling the editor.

→Process
→Insert
→press and hold left mouse key
→pull element to the desired position
→release mouse key
Alternatively:
→<Shift + Ins>
→press and hold left mouse key
→pull element to the desired position
→release mouse key

or via keyboard:

→<Shift + Ins>
→<Space>
→position
→<Space>

One/several previously copied/cut program elements can be inserted into the program at any point you like.

Delete

→select element to be deleted
→Process
→Delete or via keyboard:
→select element to be deleted
→<Del>

The deleted program elements are finally removed from the program. The program is given the status “incorrect”.

As with all other changes, deletion is no more reversible when the program and the project have been saved.

The status before calling the editor can therefore be re-created. To do so, the editor must be exited without saving and then called again.

5.9.5 Cancelling a working step

with →Process→Cancel

This function makes it possible to cancel the **last action performed.** The status of the program remains “incorrect”, irrespective of this, up to the next plausibility check.

5.10 Entries in the variables and loop tag administration

with
→Administration names in the system.
→Variablesadministrationor
→Loop tag administration
Changing is performed into the variables or loop tag administration (see Section “5.4 Variables and loop tag administration”).

Variables administration contains all inputs and output of the system that are used. A variable can be selected in the list and transferred into the program.

The loop tag administration contains a listing of all loop tag Names in the system.

If loop tag names which had been assigned to the function modules call-ups in the function plan have been deleted in the loop tag administration, then the entries in the corresponding parameter definition and configuration displays are empty after return from the loop tag administration and must be re-entered.

5.11 General processing functions

5.11.1 Saving the program

with →*Program*→*Save*.

The program is saved without exiting the editor.

Even non-plausible programs can be saved and completed at any later date.

If the project is not saved on closing the project or previously in the project tree, the program change is not effective.

5.11.2 Documenting the program

with →Program→Documentation.

It changes from the program to the documentation administration, in which the project documentation is defined and outputted userspecifically (see “7 Documentation”).

5.11.3 Hardcopy

with →Options→Hardcopy

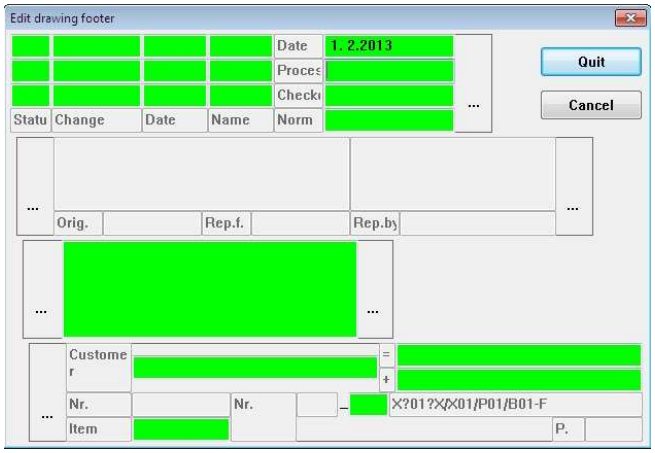
The screen content is outputted to the printer.
On monochrome printers, the frame presentation of a function module differs from the normal display.

5.11.4 Processing program header and drawing footer

with → Program → Program header or →Drawing footer

A program-specific short comment for the header line of the program documentation can be entered and/or processed.

→[Drawing ...] displays the drawing footer to be changed.
→[Accept] accepts all changes to the program header.
→[Cancel] rejects all changes to the program header.



5.11.5 Process program comment

with →Program→Programcomment.

A longer program-specific comment for describing the functional capabilities can be edited here (see “4 List configurator”).

5.11.6 Return

with →Return!.

The function module language program is exited and the application from which the change was last made into the function module language program is called (single-step return).

5.11.7 Ending function module language

with →Program→Quitprogram.

The function module language program is exited and the project tree is called.

5.11.8 Checking plausibility of elements

with
→select program element
→*Program*
→*Plausibility check*

All function relevant inputs are checked for syntactic and contextual correctness. Errors found and warnings are displayed as errors list. The processing status of the program is "incorrect" if errors are discovered in the plausibility checking.

Newly-entered, copied or moved program elements have the processing status "incorrect".

5.11.9 Deleting a function module language program

with
→Project tree
→select program
→*Process*
→*Delete*

The program is deleted from the project.

The variables and loop tag names are retained in other programs and in the variables/loop tag administration, and can be assigned again.

5.11.10 Copying and inserting a function module language program

with
→Project tree
→select program to be copied
→*Process*
→*Copy*
→select position into which the program is to be copied
→*Process*
→*Insert*
→choose "below", "above" or "level", according to position selected
→assign program name

The program is copied and assigned under a new, unambiguous name amongst a program list of the project.

The program is copied in its particular configuration, including program header and program comment. The loop tag names of the modules are not copied with it.

Copying a function module language program has no effect on the declaration of variables or loop tag names.

The copied program is labelled as incorrect. It is given the date and time of the copying operation as version identification.

5.11.11 Connecting programs

Programs are connected to one another via variables or with the input or output cards. These variables are entered into the input or output strips of the program for this (see "5.8 Entering and changing variables in the input and output strips").

5.12 General description of the instruction list

The editor is the system tool for creating and changing programs in the Instruction List language (IL).

With Instruction List, all Digimatik processing operations can be specified in list form. The range of functions covers more than FBD since jumps and program loops can also be programmed. All the function modules and functions configurable in FBD can also be invoked in IL. For modules, a CAL operator and a list of input and output signals are inserted following selection. Signal names should then be allocated to this list by the programmer. Parameters are assigned using the same input masks as in FBD. Functions (= processing elements without “memory”, e.g. SIN) are selected in the same manner and entered in instruction list by the editor as an IL operator. They have neither a parameters display nor further inputs/outputs assigned to them, nor do they require any argument.

The operators are entered in Instruction List as letter symbols
(e.g. SUB for subtraction). IL programs are processed according to the order in the list (from top to bottom). The sequence can only be changed by inserting jump, return and loop operators.

In closely linked structures of simple modules, however, FBD programming offers greater clarity.

When setting up a new program, it is possible to choose between the FBD and IL programming languages when setting up a new program. A task may therefore contain several FBD and IL programs at the same time, which can be processed one after another with the same cycle time. The programming language which is most suitable or which is preferred by the user may thus be used for each part of the configuration. However, only one programming language is valid within a program.

IL programs can be up to 1000 lines in length. As a result, they may contain substantially more processing functions than FBD programs, which are limited to the screen display.

The operators and functions used in IL are based on the standard IEC 1131-3 [previously: IEC 65 B W67].

5.12.1 Creating a new IL program

IL programs can be created from an active program listing or from the program “POOL” and can be called for editing.

An IL program is set up from the Project tree using the following steps:

→Projecttree
→select insert position in project tree
→Edit
→Insertaboveor Insertbelowor Insertson
→“IL program” from “Select object” window
→enter program name and short comment

or via keyboard:

→<↑>, <↓>	selection mark moved over the insert position
→<Alt>	top menu line preselected
→<B>	“Edit” selected as submenu
→<D>	“Insert below” selected
→<Enter>	“IL program” selected. The program name can now be entered/changed in “Head: Program”
→<Tab>	enter the short comment in the comment field
→<Enter>	exit from the menu, the new IL program is inserted into the project tree and is preselected for configuration

Each new IL program has a blank instruction list, the editing status “incorrect” and the creation date as the version identification. The name of the program list is preset as the program name, which be deleted using the <Backspace> key and entered anew.

5.12.2 Calling the editor

As soon as the IL program exists, it can be called for editing:

→*Project tree*
→move cursor to program name and doubleclick

or

→select program by left click
→*Configure*

or via keyboard:

→<↑>, <↓> selection mark is moved to program name
→<Enter> the preselected instruction list is displayed

New lines may now be entered or entries changed in the instruction list displayed.

5.12.3 Editors interface

Owing to the list structure of the editing interface, the operating are steps outlined in the description of the variable and loop tag adminstration apply by analogy, e.g. for selecting fields, marking, deleting, moving or copying blocks and for the selection of variables or loop tags and other column-specific entries (via the <F2> key).

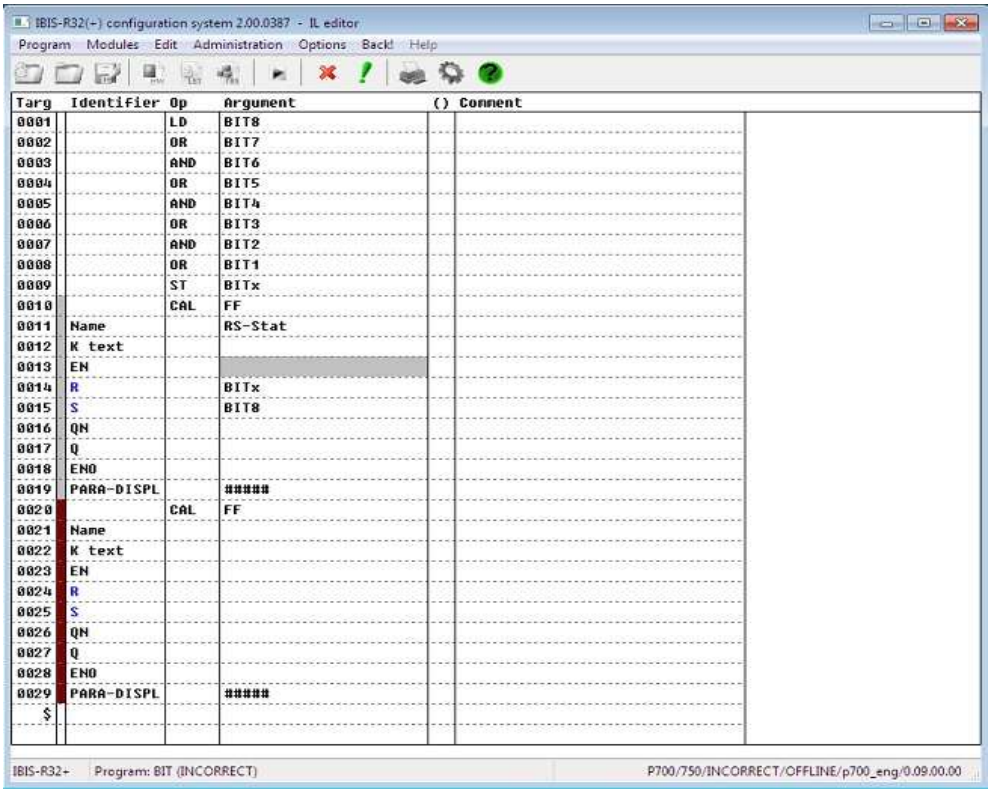
However, peculiarities occur owing to the list columns, which tailored to IL, and the menu line.

Structure of the IL-editor configuration interface:

Menu line→

Column headings→
Report lines→

Status line→



↑ Line no. ↑Label ↑Argument ↑Parenthesis depth
↑Module marking ↑Comment
↑Operator

Line no.

The line number is allocated automatically in consecutive sequence from 1 to 1000. When blank lines or command lines are inserted, the line numbers of subsequent command lines are automatically displaced by the number of lines inserted.

Module marking

All the lines belonging to a function module are marked here in colour unless the “must” parameters contained therein are fully assigned. Once they are fully assigned, these fields become grey.

Label

Jump marks L001 up to L999 (label), which act as transfer addresses for jump operators, are entered in this column. The entry is not tied to any sequence. It is nevertheless recommended to aim for an ascending sequence, but to use only full figures of tens at first, so as to be able to insert further jump marks later in monotone sequence. The monotone sequence makes searching easier in longer program listings.

Operator

Once a field has been selected in this column, the operator can be entered by key input or by selection from a menu, which can be called using →<F2>. Depending on the operator type, a (suitable) argument should then be specified if necessary in the adjacent column.
In the case of function modules, this field is assigned automatically following module selection.

Argument

In the case of jump operators, the jump mark should be entered here, whereas logical operators require a constant or a variable as an argument.

Special conditions apply here also for function modules.

Parenthesis depth

When parenthesizing logical operators, a number 1 ... 8 appears here, which indicates the depth of parenthesis nesting.

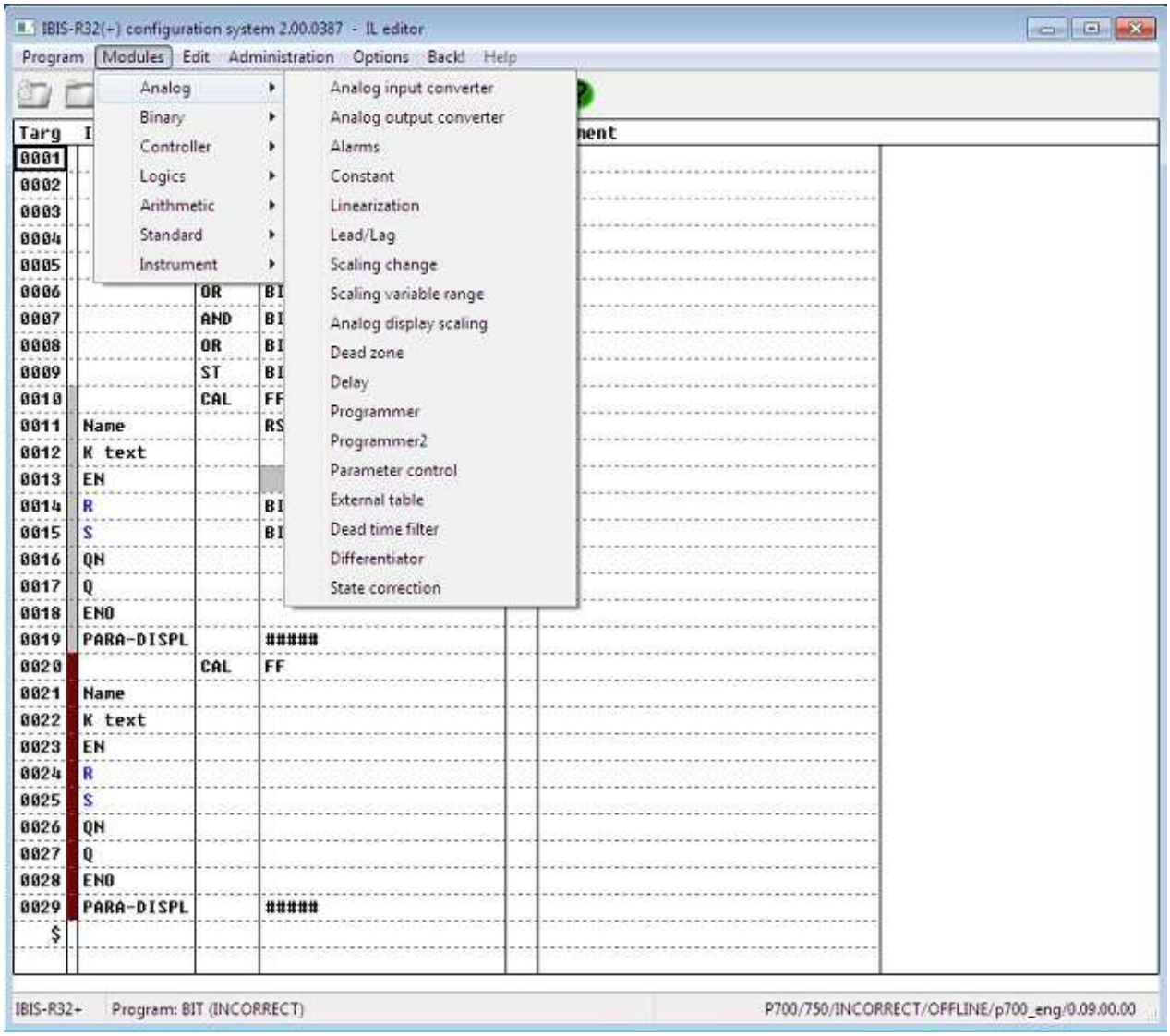
Comment

Explanations can be entered here to aid understanding of the program run, e.g. on the meaning of variables, the function of the program section or the function module called.

Status line

The status line indicates the name of the program which is being edited and the momentary status of the plausibility check. The status “correct” is only allocated if a plausibility check has been carried out on the program and no errors were detected in this process. Each newly created IL program which has not yet been edited is always regarded initially as incorrect.

5.12.4 Operation via the menu bar



For mouse operation or selection by way of letter keys, a variety of actions can be initiated or selections made via the menu bar at the top edge of the display. To display all these menu items, sub-menus can be selected at various levels if required. All those fields in the menu bar which do not branch to any sub-menu are identified by an exclamation mark at the end of the descriptive text. In sub-menus, menu items with a sequential menu are marked by a dark triangle at the right margin.

The menu bar used to edit IL programs contains the following selection fields:

Program
When this field is selected, a sub-menu is opened which contains the following menu items:

Save
stores the program level which has been reached temporarily.

Documentation
branches to documentation management.

Hardcopy
prints out the screen contents.

Check
checks the program for errors.

Program **H**ead
shows the header text of the program and ooffers the possibility to edit.

Program **C**omment
edits the program comment.

Quit program
quits the IL program.

Modules
selects of function modules and functions.

Edit
opens a sub-menu containing the following menu items (Depending on the editing status, not all menu items are effective. Those which are ineffective are identified by grey type.):

Insert line (<Ins>) inserts a blank line ahead of the selected position.

Edit **f**ield
calls the selected field for editing.

Delete fiel**d**
replaces the selected field by a blank field.

Define signal type
changes between input and output data types, e.g. with function AND.

Parameterization
allows input of module parameters.

Number of inputs allows changing of input number for modules with a configurable number of inputs, e.g. “multiplexer”.

Select variable
calls variables list for direct selection.

Cut (<Shift> + <Del>) stores temporarily a marked block and deletes it from its previous location.

Copy (<Shift> + <Ins>) stores temporarily a marked block and leaves it in its previous location.

Insert (<Ins>) inserts the temporarily stored block ahead of the preselected position, but the block remains in temporary storage.

Delete (<Del>)
deletes the marked block.

Undo
reverses the last editing step.

Administration
opens a sub-menu which contains the following menu items:

Variable administration lists all variables currently available with details of data type.

Tag administration lists all controllers, loop controls und similar modules with tags (the lists can each be sorted according to specified criteria).

Options
opens a sub-menu which contains the following menu items:

Info
shows the version text of the program. The creation date appears here initially.

Colours allows to edit and change colours for various text sections and fields of the IL editor

Back! exits from the IL editing display back to where IL was called from. called is displayed.

Help
Is not supported at the moment.

5.12.5 Acceptable data types for IL operators and functions

The 4 data types which are possible are entered in the in the following table as columns. The table provides information in the form of a matrix showing which IL operators can process which data types:

If modules are used in instruction list, the acceptable data types are dictated by the module type. In the case of modules for different data formats (see table below), a menu window is opened in which the data type is selected.

	BOOL	INT	DINT	REAL
LD, ST	✓	✓	✓	✓
LDN, STN	✓	–	–	–
AND, OR, XOR	✓	–	–	–
ANDN, ORN, XORN	✓	–	–	–
NEG	✓	✓	✓	✓
DEC, INC	–	✓	✓	–
EQ, GE, GT, LE, LT, NE	✓	✓	✓	✓
ADD, SUB	–	✓	✓	✓
MUL, DIV, MOD	–	✓	✓	✓

Tab. 1

	BOOL	INT	DINT	REAL
ABS	–	✓	✓	✓
AVG	–	✓	✓	✓
MIN, MAX	–	✓	✓	✓
MUX	✓	✓	✓	✓
SEL	✓	✓	✓	✓
TRUNC	–	✓	✓	✓

Tab. 2 Modules with several data types

5.12.6 Entering constants

Constant numerical values can be input according to data type with or without a sign in binary, octal, decimal or hexadecimal format. Floating-point numbers must always be input with the decimal point, even if accompanied by an exponent.

To differentiate them from decimal numbers, binary, octal and hexadecimal numbers are preceded by a suitable identification character (2#, 8# or 16#, see below).

The input limits which apply are:

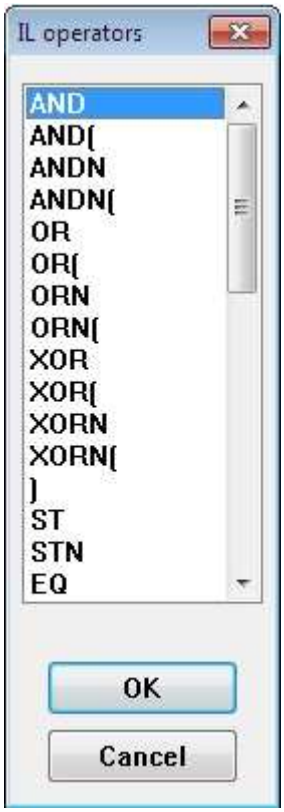
BOOL0, 1, TRUE, FALSE (only capital letters)

INT −32768 ... + 32767, 16#8000 ... +16#7FFF, also binary and octal format

DINT −2147483648 ... +2147483647, 16#8000 0000 ... 16#7FFF FFFF, also binary and octal format

REAL ±5,877472E−39 ... ±3,4028236E38 (IEC format for 32-bit floating-point numbers)

5.12.7 Calling IL operators



The structure of IL programs is adapted to that of assembler programs of simple microprocessors with an accumulator. Constants or variables are loaded into this “accumulator”, combined with other quantities, transformed and stored in a target quantity.

Operators are the basic elements of the instruction set. They can be subdivided into the groups “Logic”, “Basic Arithmetic”, “Comparators”, load, store and other organizational instructions.

Once an operator field has been selected, the list of operator types currently available can be called with →<F2> and the desired operator selected by means of the →<↑>, →<↓> and
→<Enter>. The shorthand symbol for the operator may also be entered directly, bypassing the selection menu
(→<Enter> in operator field and enter letter; completion again by means of
→<Enter>).

To separate program sections from one another by a blank line, the line following the desired point of separation is selected, Edit is called, the menu item Insertlineis selected and confirmed with
→<Enter>.

5.12.8 Loading and storing data

All data and signal types are loaded into the accumulator using the operator **LD**. In the case of Boolean data/signals the operator **LDN** may also be used, which loads the input quantity into the accumulator in inverted form. The corresponding operators for storing the accumulator contents are **ST** or **STN**.

Since a storage operator does not change the accumulator, it can be used several times in succession to distribute the same contents to various outputs. The output variables must be of the same type as the accumulator contents, otherwise an appropriate type corruption message giving the pertinent line number will be generated during the plausibility check.

5.12.9 Logic operations

Boolean and other bit string quantities can be combined with one another using the operators **OR** (or), **AND** (and) und **XOR** (exclusive or) These logical operators can be combined with the supplements “N” (=negated) or/and “(” (=left parenthesis).

A complete list of all IL operators is featured in Section 5.12.15.

The table below provides information on the meaning of the individual logic operations. In-depth treatises on the theory of logic operations are to be found in specialist literature on the subject.

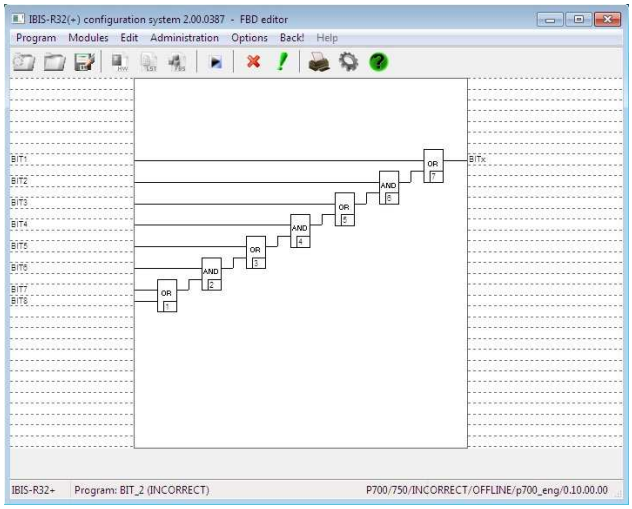
Function	AND		ANDN		OR		ORN		XOR		XORN	
Argument	0	1	0	1	0	1	0	1	0	1	0	1
Accu = 0	0	0	0	0	0	1	1	0	0	1	1	0
Accu =1	0	1	1	0	1	1	1	1	1	0	0	1

Tab. 3

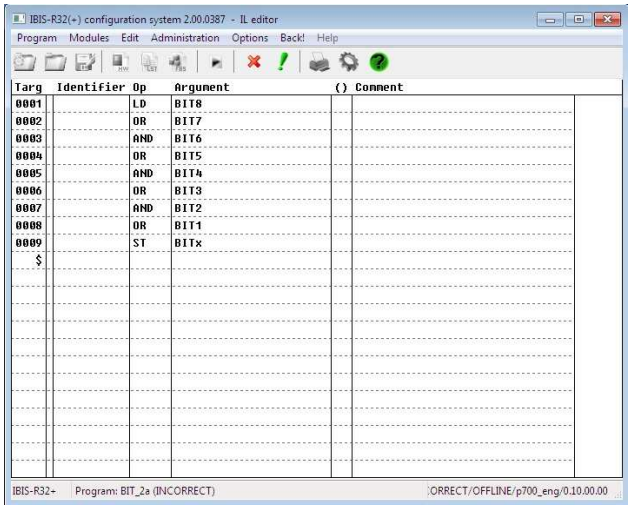
5.12.10 Logical operators with parentheses

The supplement “left parenthesis” named in the previous section, together with the operator “)” (right parenthesis) makes it possible to convert even complex logic operations into corresponding IL line sequences. In principle, all operations can be formulated even without parentheses if intermediate results are filed in flags and reloaded later. However, this calls for more instruction lines, and clarity is reduced.

Nevertheless, the number of lines can be reduced markedly by the skilful use of partial results in the accumulator. It is often possible to avoid storing intermediate quantities simply by resorting the operations, as the following example shows:

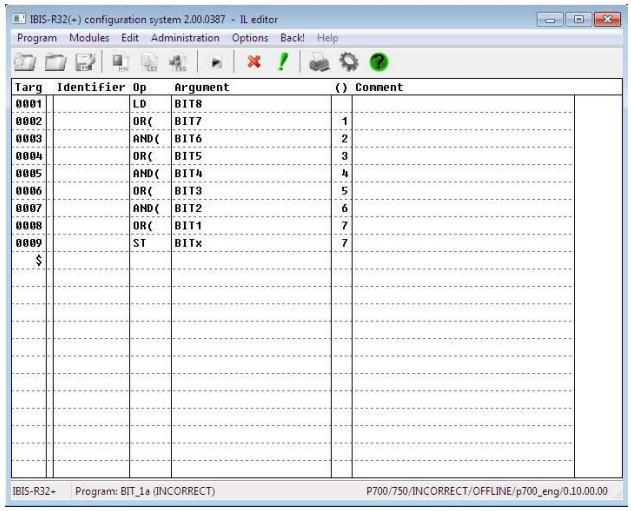


This operation may be implemented by means of the IL program opposite:



In certain structures, however, intermediate quantities and thus unnecessarily long program sections may only be avoided by using parenthesized expressions.

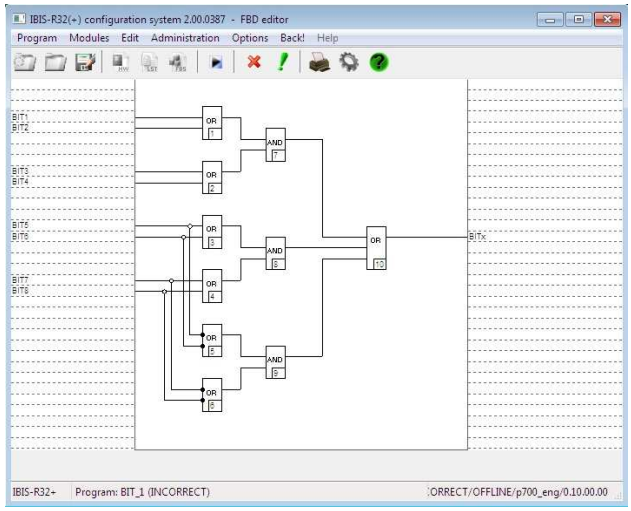
Parentheses may be nested up to a depth of 8 levels. The respective parenthesis depth is shown in instruction list in the 6th column. Red question marks appear in this column if the 8th level is exceeded. The parenthesis depth shown must be brought back down to 0 again in subsequent lines using right parenthesis operators:



Since at present no provision exists for negation following a “left parenthesis” supplement, parenthesized expressions with negated input signals first have to be converted to be able to program them using IL operators. The following conversion rules may be used for this:

(NOT[e1] OR NOT[e2]) = NOT(e1 AND e2)
(NOT[e1] AND NOT[e2]) = NOT(e1 OR e2)
(NOT[e1] XOR NOT[e2]) = (e1 XOR e2)

The following example shows the variants described with and without parentheses:



IL without parentheses	with all intermediate variables	IL without parentheses	intermediate variables reduced	IL with parentheses	[operation converted]
LD	bool1	LD	bool1	LD	bool1
OR	bool2	OR	bool2	OR	bool2
ST	z1	ST	z1	AND(	bool3
LD	bool3	LD	bool3	OR	bool4
OR	bool4	OR	bool4	)	
ST	z2	AND	z1	OR(	bool5
LD	bool5	ST	z7	OR	bool6
OR	bool6	LD	bool5	AND(	bool7
ST	z3	OR	bool6	OR	bool8
LD	bool7	ST	z3	)	
OR	bool8	LD	bool7	)	
ST	z4	OR	bool8	ORN(	bool5
LDN	bool5	AND	z3	AND	bool6
ORN	bool6	ST	z8	OR(	bool7
ST	z5	LDN	bool5	AND	bool8
LDN	bool7	ORN	bool6	)	
ORN	bool8	ST	z5	)	
ST	z6	LDN	bool7	ST	boolX
LD	z1	ORN	bool8		
AND	z2	AND	z5		
ST	z7	OR	z8		
LD	z3	OR	z7		
AND	z4	ST	boolX		
ST	z8				
LD	z5				
AND	z6				
ST	z9				
LD	z7				
OR	z8				
OR	z9				
ST	boolx				

Tab. 4 Example for saving lines because of changes in sequence ...

5.12.11 Relational operators

Two quantities of the same data type (previous accumulator contents and argument) are compared with one another using the relational operators **EQ** ... **LE** and the result stored in the accumulator as a Boolean variable. The relation functions can also be called as modules, but are not distinguished by this from the operators.

5.12.12 Numerical operations

The operators **ADD**, **SUB**, **MUL** and **DIV** can be used to combine two quantities (accumulator and argument) of the same data type. The result is then available in the accumulator for storage or for further operations. Numerical operations can also be called as modules.

However, the addition and multiplication modules are distinguished from the corresponding operators by the fact that they can be used for multiple inputs.

5.12.13 Loop operators

In offering the opportunity to incorporate repeat loops into programs, the IL language differs markedly from FBD. At the start of a loop one of the loop start operators **WLC**, **RPC** or **WLNZ** respectively, followed by the "loop core", which is to be executed several times, consisting of load, processing and storage operators as well as module calls. At the end of this part, the loop terminate operator **LPE** is inserted.

Loop starting instructions have the following meaning:

WLC (**W**hi**L**e **C**ondition)	skips the loop if the accumulator is not TRUE.
RPC (**R**e**P**eat on **C**ondition)	checks the accumulator only at the end of the loop (in the line with LPE). If it is TRUE, the loop is executed once more.
WLNZ (**W**hi**L**e **N**ot **Z**ero)	checks (at the beginning of the loop) a counter defined by "Argument" with DINT format. If it is zero, the loop is aborted, otherwise it is executed.

All three types of loops can degenerate into endless loops as a result of poor programming.

Example of an IL program with loop operator:

The program signals TRUE following TempFlr if at least one of the temperatures Temp1 ... Temp7 is greater than the fixed value 70°C:

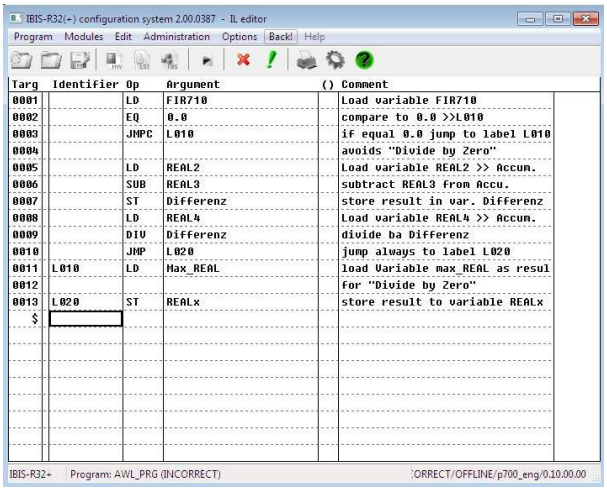
```
      LD      MaxKnl    maximum number of channels
                        to be monitored
      ST      DZLR      store → DZLR
      GT      7         if greater than 7
      RETC    Programm  terminate program
      LD      1         initial value 1
      ST      ZLR       store → ZLR
      WLNZ    DZLR      process loop up to LPE, if
                        DZLR > 0
      LD      ZLR       channel counter as selection
                        criterion for multiplexer
      MUX     Temp1     Channel 1
      ,       Temp2     Channel 2
      ,       Temp3     Channel 3
      ,       Temp4     Channel 4
      ,       Temp5     Channel 5
      ,       Temp6     Channel 6
      ,       Temp7     Channel 7
      GT      70.0      if selected temperature greater
                        than 70.0 °C
      JMPC    L030      then jump (with accumulator =
                        1) → L030
      LD      DZLR
      DEC               decrement the counter DZLR
                        by 1
      ST      DZLR
      LD      ZLR
      INC               increment selection channel
                        ZLR by 1
      ST      ZLR
      LPE               loop end
      LD      FALSE     since no temperature greater
                        than 70°C, load FALSE
L030  ST      TempFlr   store accumulator contents
                        →TempFlr
      RET               program end
```

5.12.14 Jumps and program calls

Der Sprung wird bei **JMP** stets ausgeführt, bei **JMPC** nur, enn der Akku TRUE ist, bei **JMPCN** nur, wenn der Akku FALSE ist.

Using the jump operators **JMP**, **JMPC** or **JMPCN**, the program can be continued at the point named in the argument, the lines lying in between are skipped. The jump destination must lie below the line containing the jump operator. It should be entered in the destination line by means of an identifier in the form L001 ... L999.

The jump is always executed if **JMP** is specified. In the case of **JMPC**, it is only executed if the accumulator is equal TRUE, and for **JPMCN** only if the accumulator is equal FALSE.



5.12.15 Summary of IL operators

AND	Accu AND Operand to accumulator
AND(	Accu AND left parenthesis
ANDN	Accu AND (argument negated)
ANDN(	Accu AND negated, left parenthesis
OR	Accu OR argument to accumulator
OR(	Accu OR left parenthesis
ORN	Accu OR (argument negated)
ORN(	Accu OR NEGATED, left parenthesis
XOR	Accu EXOR argument to accu
XOR(	Accu EXOR left parenthesis
XORN	Accu EXOR (argument negated)
XORN(	Accu EXOR NEGATED, left parenthesis
)	Right parenthesis
ST	Store accumulator to argument
STN	Store accumulator in inverted form to argument
EQ	If accumulator is equal to argument, TRUE to accumulator, otherwise FALSE
NE	If accumulator is not equal to argument, TRUE to accumulator, otherwise FALSE
GT	If accumulator is greater than argument, TRUE to accumulator, otherwise FALSE
GE	If accumulator is greater than or equal to argument, TRUE to accumulator, otherwise FALSE
LT	If accumulator is less than argument, TRUE to accumulator, otherwise FALSE
LE	If accumulator is less than or equal to argument, TRUE to accumulator, otherwise FALSE
DEC	Decrement accumulator (-1)
INC	Increment accumulator (+1)
NEG	Negate accumulator
NOP	No operation
JMP	Jump to mark indicated in argument field unconditionally
JMPC	Jump if accumulator is equal TRUE
JMPCN	Jump if accumulator is equal FALSE

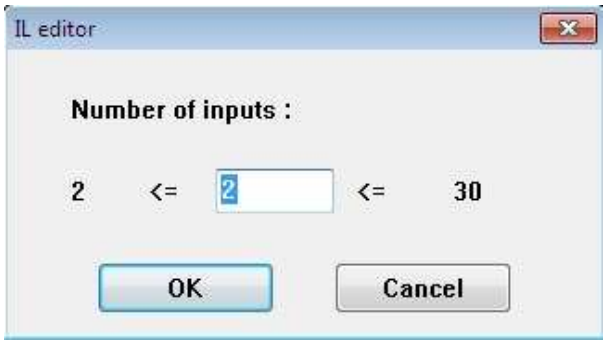
ADD	Accumulator plus argument to accumulator
SUB	Accumulator minus argument to accumulator
MUL	Accumulator times argument to accumulator
DIV	Accumulator divided by argument to accumulator
LD	Load argument to a accumulator
LDN	Load argument in inverted form to accu
RET	Return from program (sub-program) unconditionally, terminates the IL program
RETC	Return if accumulator is equal TRUE
RETCN	Return if accumulator is equal FALSE

WLC	If accumulator is equal TRUE, execute the following lines as far as LPE
RPC	As for WLC, but loop is executed at least once
WLNZ	If the integer counter named by argument is not zero, execute the lines as far as LPE, with every loop the counter in decremented by 1
LPE	End of a loop

5.12.16 Inserting function modules into an IL program

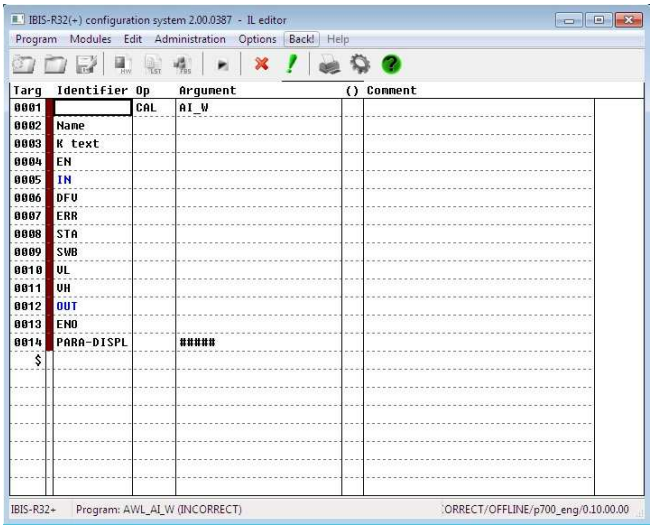
All the function modules available in FBD programming can also be called in IL by way of Modules. Some of these modules have the attribute of a variable number of inputs (AND, OR, XOR, ADD, MUL, MUX). When these modules are called, a selection window is shown, in which the desired number of inputs should be entered within acceptable limits. The smallest reasonable number is preallocated here.

It is also possible to enter the corresponding individual operator into the IL list instead of modules with multiple input and to multiply this by copying, thereby avoiding being restricted to 30 inputs.



A large number of function modules are “named” modules, i.e. they are entered into the instruction list using a CAL operator and receive a name, a comment and a parameters display. When they are called, a fixed block of IL lines is inserted into IL ahead of the selected list position. A line is reserved in each case for all inputs and outputs. All lines of the module except the CAL line contain an identifier text, which identifies the respective signal. The identifiers for necessary inputs/outputs (“must” parameters) are highlighted in colour.

Certain argument fields have a grey background if the relevant input has already been occupied in the parameters display by a constant quantity. Column 2 is marked in colour if all “must” parameters of the module have not yet been properly entered or the module has been taken out of processing, otherwise the colour marking is grey. It is possible to tell from this marking that a module is being used, even in the case of instructions which are available as an operator and as a module.



The parameters display which goes with the named module is selected as follows:

→doubleclick on the field “PARA-DISPL”

or via keyboard:
→position the cursor on the field “PARA-DISPL” (out-lined in black) with <↑>, <↓>
→<Enter>

The parameters displays are the same as for FBD programming. It is possible to quit the parameters display at any time with →<Esc>.

The comment field in the last line of the module initially contains a row of 5 hash signs (#####). This marking indicates that the module has not yet been checked successfully for plausibility. Following the plausibility check these symbols change into „@@@@@“.

5.12.17 Checking the plausibility of the program

If something was missed, forgotten or entered incorrectly during program input, the plausibility check supplies the formal warning and error messages, together with the line number, so that corrections can be made prior to commissioning.

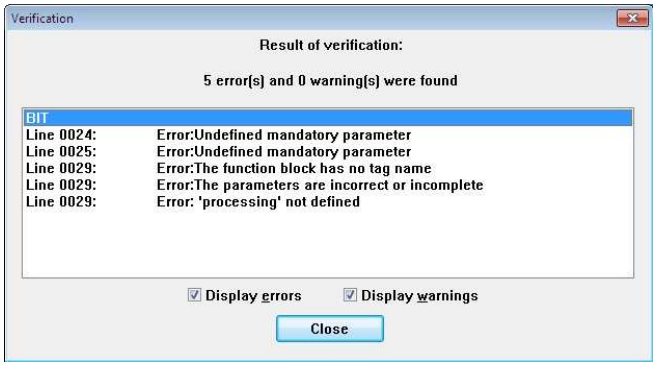
In any case, the program can only be downloaded into the instrument and operated if a plausibility check has been accomplished without any error messages. The plausibility check is called with

→Program→Verification.

or via keyboard:

→<Alt>
→<P>
→<V>

Once the plausibility check has been completed, the following message window appears:

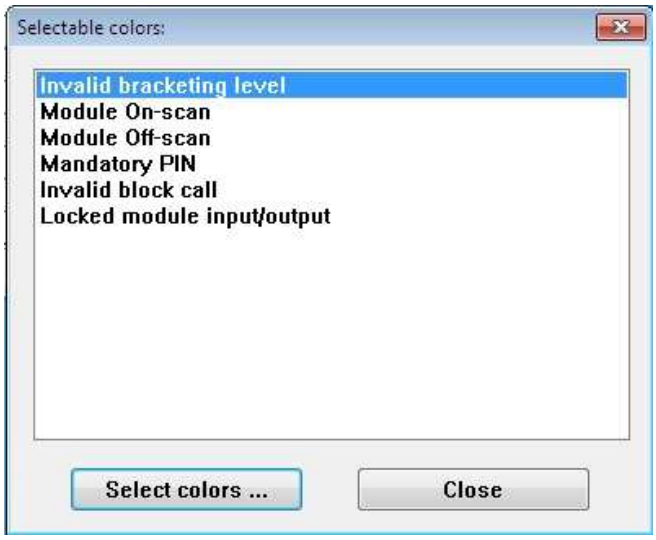


5.12.18 Changing the presettings

Selecting the Options item from the menu bar opens the following menu:

Colours see 5.12.19
Info see 5.12.20

5.12.19 Changing the colour display

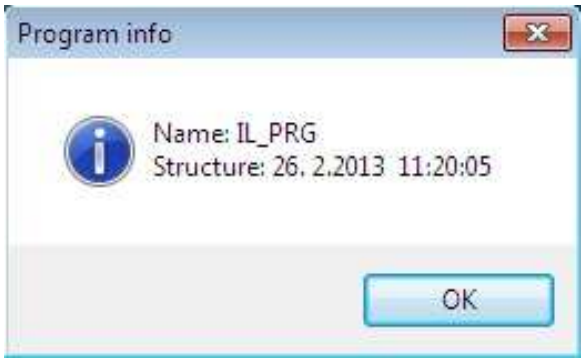


with
→Options
→Colours
→select object of which the colour is to be changed
→select desired colour

or via keyboard:

→<Alt>
→<O>
→<C>
→select the display colour to be changed with <↑>, <↓>
→<Enter>
→set the desired colour with <↑>, <←>, <↓>, <→>
→<Space>
→<Enter>
→<Esc>

5.12.20 Showing the version information



with →Options→INFO.

or via keyboard:

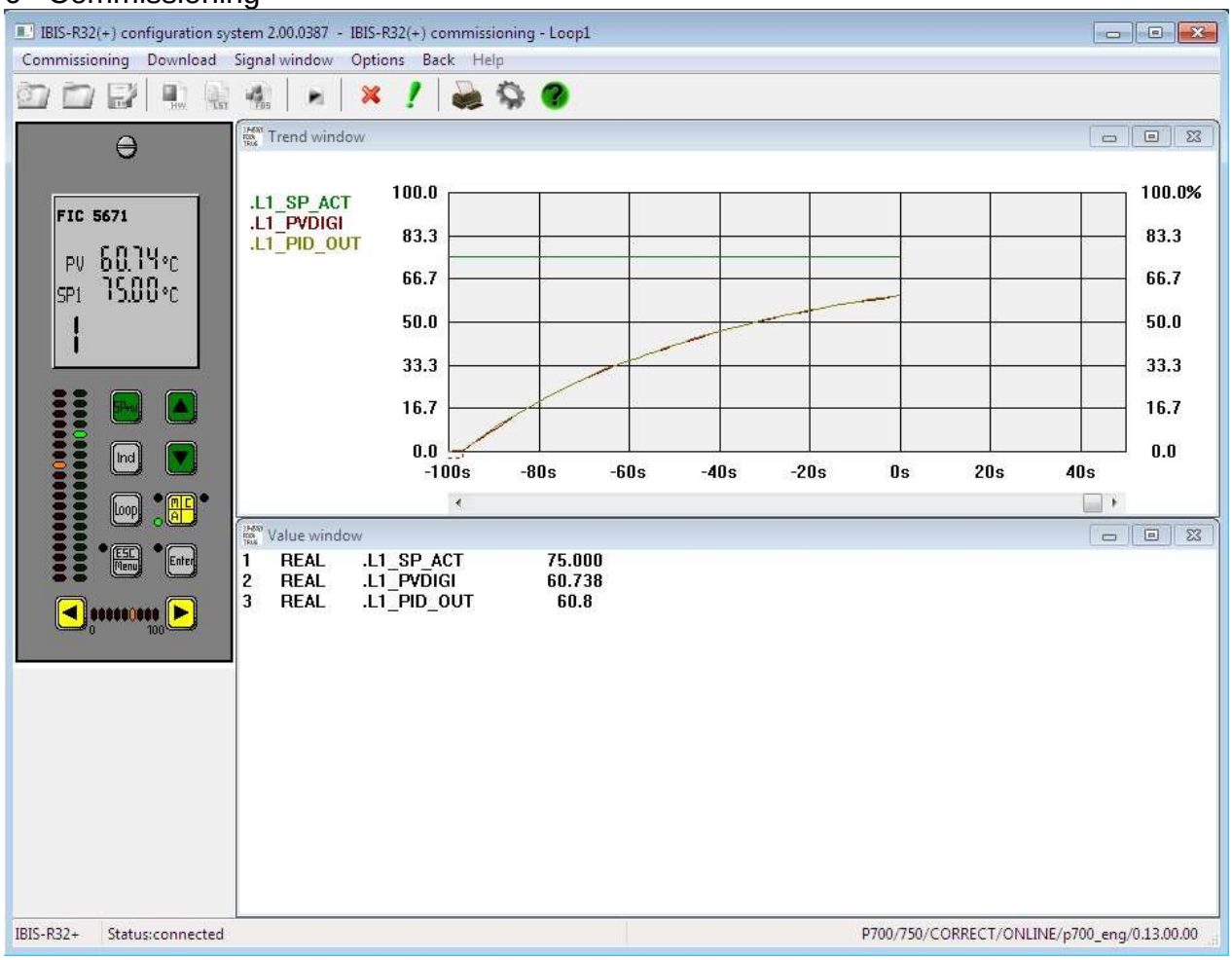
→<Alt>
→<O>
→<I>

The program name, date of last program change (version) and allocation of the program to the project. The program allocation can be shown as long or short text. The setting for this is made in the project tree.

6 Commissioning

Table of contents

6 Commissioning



The function of the commissioning menu is to transfer a configura-tion generated on the computer to a controller or to start up the controller via the computer.

Commissioning offers a number of windows to operate and observe the controller. One feature is a simulation of the controller front panel containing all the unit displays and controls. Another window displays trend curves and a number of controller variables. A special parameter window is specially designed for parameterizing the controller. These windows and their display and data selection are each allocated to a control loop.

If it is determined upon starting the communication between computer and controller that the configured project does not agree with the project being edited on the controller, a message is displayed:

[OK] acknowledges this message.

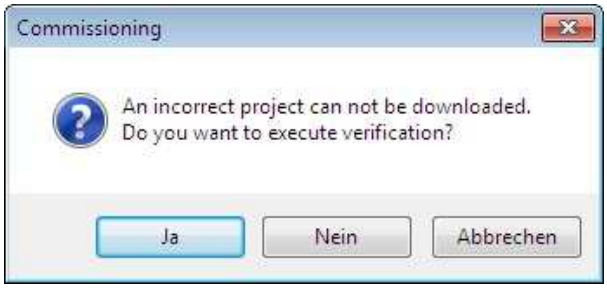
If on-line parameters are changed during the commissioning, these will not be adopted into the project currently running on the computer.

It is only after downloading the project that parameter changes can be introduced into the project.

The nonadoption of parameters into the project is made clear with a texted notice when entering the individual parameters. If a parameter is adopted, this fact is not displayed with a texted notice.

If it is determined upon starting the communication between computer and controller that the library used for the project does not conform with the firmware version of the controller, all dynamic displays will be indicated in red and the text “VERS” will be displayed, instead of the data types in the value window.

If it is determined upon beginning the commissioning that the opened project is not CORRECT, a message shall be output and the plausibility check of the list configurator or the free configuration can be called up.



→[Yes]	calls up the required configurator and starts the plausibility check.
→[No]	calls up the commissioning with the incorrect project. The project can then not be downloaded onto the controller.
→[Cancel]	aborts the start-up of the commissioning and remains in the program from where commissioning was called.

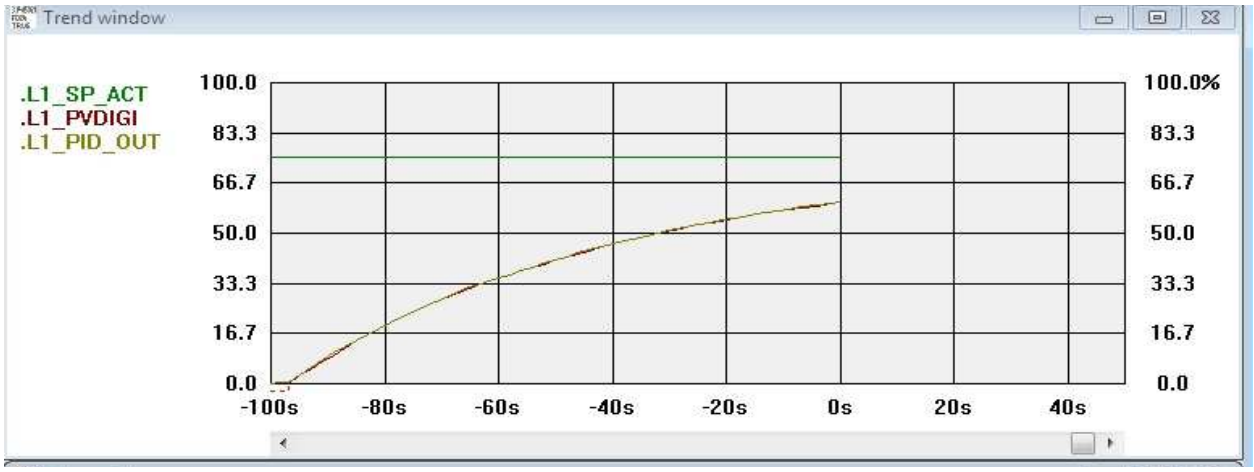
The following description is also valid for Digitrenic 700. It deviates only in the controller front panel.

6.1 Controller front panel

The simulated on-screen controller front panel reproduces the displays which would occur on a real controller. At the same time, operator actions can be performed on the simulated front panel by activating the on-screen buttons via mouse click or keyboard (Switching over the front panel displays using the
→[Loop] command only switches over the front panel display. There is no switchover of other display windows to other control loops.):

▲	<Shift + ↑>	
▼	<Shift + ↓>	
◄	<Shift + ←>	Not Digitrenic 700
►	<Shift + →>	Not Digitrenic 700
SP-w	<Shift + S>	
Ind	<Shift + I>	
Loop	<Shift + L>	
Menu/Esc	<Shift + M>	
Enter	<Shift + Enter>	
MAC	<Shift + A>	

6.2 Trend window



The commissioning function provides a trend window which can be displayed in two sizes. It can be displayed in a small form together with the controller front panel and the value window or it can be displayed covering the full screen. You can switch over between the two displays by →[▲] or →[▼] in the upper right corner or by →Signal window→Trend/value windowfor the small window and by →Signal window→Trend window for the large window.

The variable names displayed in the trend window have different colours. Each variable can be assigned a colour by definition. The colour is used for both displaying the variable name and for displaying the curve. The colours are used as long as the computer and controller are connected via interface. Once the communication is disconnected, the variable names switch over to red and the curve display in the trend window is interrupted. However, the timer for the curve display continues.

A maximum of 6 curves can be displayed for each selectable control loop.

The trend window size can be shifted with →<STRG + →>or <STRG + ←>.

The size of the trend window can be modified with →Options→ Trendtimebaseto various other time intervals. In case of a set time of more than 100 s, not every value read per second is displayed. Therefore it is possible to select different modes of display. Also refer to “6.7.4 Options”.

6.3 Value window

The commissioning function provides a value window which can be displayed in two sizes. It can be displayed in a small form together with the controller front panel and the trend window or it can be displayed covering the full screen. You can switch over between the two displays by → [▲] or →[▼] in the upper right corner or by →Signal window → Trend/value window for the small window and by → Signal window → Value window for the large window.

The variable names displayed in the value window are always in black as long as the computer and controller are connected via interface. Once the communication is disconnected, the variable names switch over to red and the value display is interrupted.

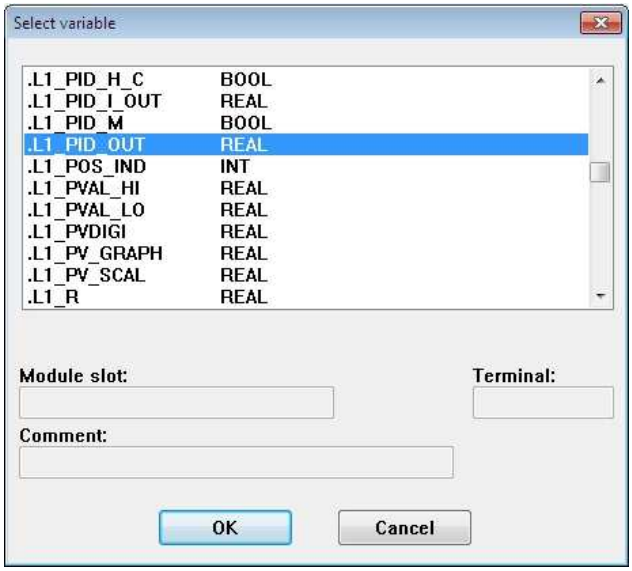
You can define a maximum of 12 values and 20 values can be displayed for each selectable control loop.

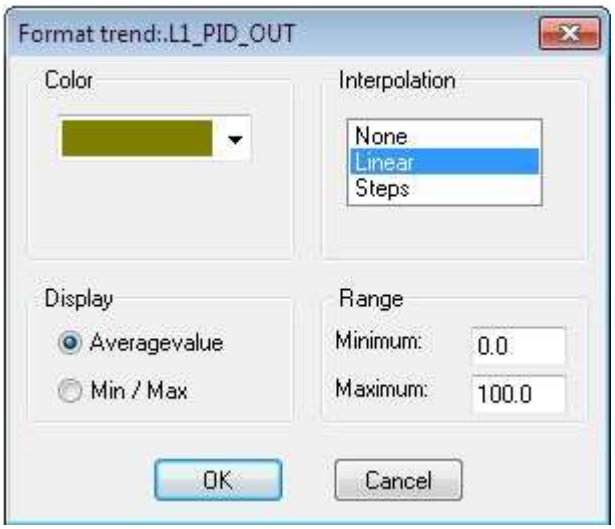
6.4 Variables

Certain variables can be selected for display in the trend and value windows, separately for each window, using →Signalwindow → Definesignals. For the trend and value window it is possible to define certain variables in certain formats for display separately for the windows with →Signalwindow → Signals.

The used variables are displayed in the trend and value windows in the upper left corner.

6.4.1 Choosing variables





with →[New].

Whether a variable should be displayed in the value and/or trend window is indicated in “Type of display”. With mouseclick or →<↑>or →<↓> and →<Spacebar> “Value window” or „Trend window” can be selected. During the first selection of a variable for the trend window, the input window “Format trend” is automatically called up for editing (see also below).

→[New] displays a selection list of the displayed variables. Double-click or →<↑>or →<↓> and choose →[OK] to transfer to the variable list.

→[Delete] deletes the selected variable from the variable list.

→[Up] moves the variable you select by one line up.

→[Down] moves the variable you select by one line down.

→[Apply] activates the entries for a variable in the value and/or trend window(s) without ending the signal window.

→[Reset] undoes the last change to the settings.

→[OK] activates and saves the entries and ends the signal window.

→[Cancel] closes the signal window without saving the changes.

→[Format trend] switches to a further window in which the display of the selected variables in the trend window are determined. These are scale, curve colour, interpolation and display of larger time intervals. If “Average value” is displayed, the average value recorded per second is displayed in the trend window as of the time base 200 s. For “Min/Max” the smallest or biggest value is selected for display respectively.

If trend window check box in the display mode box is not activated before the button is chosen, trend formatting is turned on automatically when you close the function.

6.4.2 Choose display format

The display formats of the variables selected on the left side, in which these variables are displayed in the value window are located in the right upper window. If using in the value window, inputs are either windowed or faded out. It is possible here to select several different formats for one variable.

After selecting a variable with mouseclick or →<↑>or →<↓> and →<Space> the display formats are clicked or selected with
→<Tab> and →<↑>or →<↓> and →<Space>. To delete a format, this is clicked on or selected, as described above and effaced with →<Space>.

The formats offered depend on the data type. The “standard” format in each case corresponds to the most common display format for the data type.

The following formats are available for displaying the REAL data type:

Fixed decimal point 1 Left-flush display with one decimal place
Fixed decimal point 2 Left-flush display with two decimal places
Fixed decimal point 3 Left-flush display with three decimal places Fixed decimal point 4 Left-flush display with four decimal places Floating point Left-flush display in IEC notation

Prokos fixed 1	Display with max. 6 places and one decimal place
Prokos fixed 2	Display with max. 6 places and two decimal places
Prokos fixed 3	Display with max. 6 places and three decimal places
Prokos fixed 4	Display with max. 6 places and four decimal places
ProkosFloat	Display with max. 6 places and floating decimal point

The following formats are available for displaying the INT data type:

Binary	Left-flush display in binary format preceded by 2#
Decimal	Left-flush display in decimal notation
Hexadecimal	Left-flush display in hexadecimal format preceded by 16#
Octal	Left-flush display in octal notation preceded by 8#

The following formats are available for displaying the DINT data type:

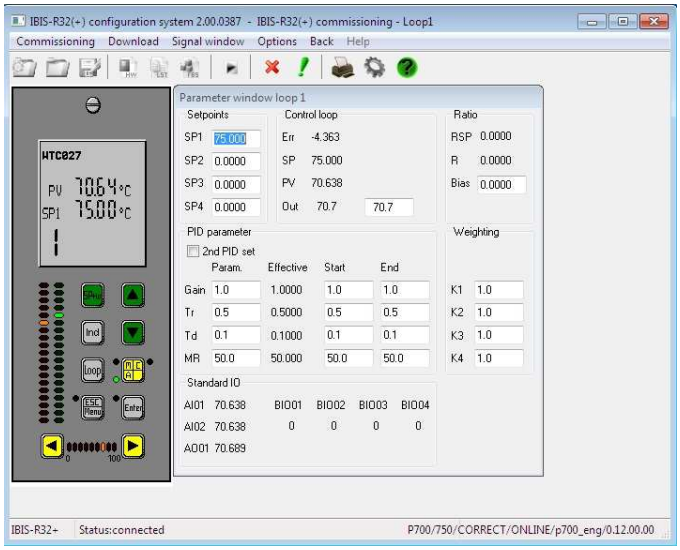
Binary	Left-flush display in binary format preceded by 2#
Decimal	Left-flush display in decimal notation
Hexadecimal	Left-flush display in hexadecimal format preceded by 16#
Octal	Left-flush display in octal notation preceded by 8#
Time	Display in hours-, minutes-, seconds-format preceded by T#

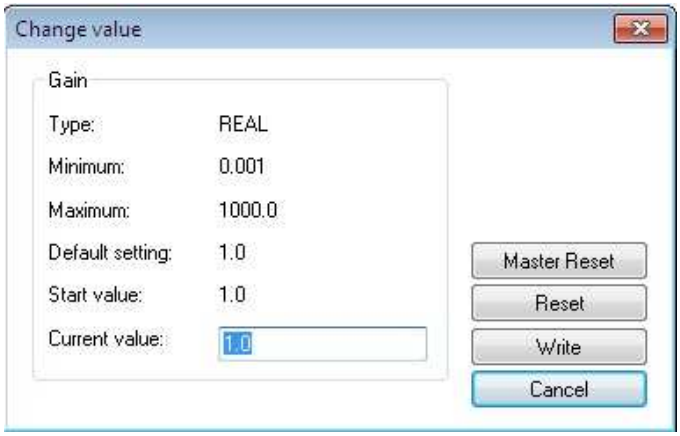
The following formats are available for displaying the BOOL data type:

Bool	Left-flush display of the texts TRUE or FALSE
Decimal	Left-flush display of 0 or 1

Using the value window and trend window check boxes in the display mode box, you can define the window in which you want to display the selected variable. Simultaneous display in both windows is possible.

6.5 Parameter window





with →Signalwindow→Parameterwindow.

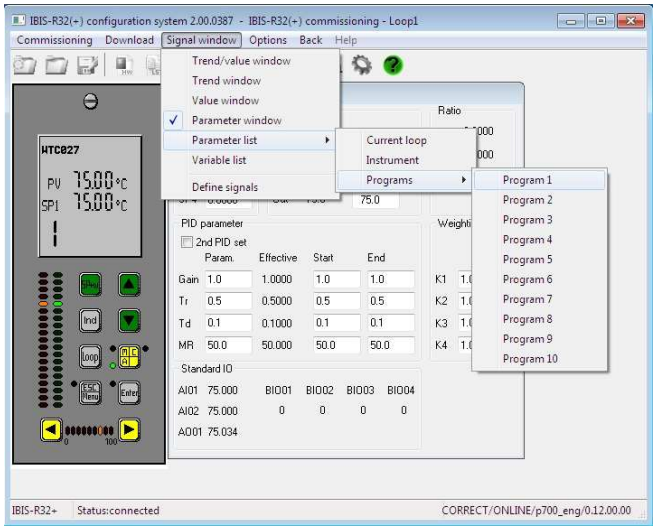
The commissioning function provides a parameter window for dis-playing and editing the main control loop data. The window is displayed for the control loop selected from the commissioning menu.

A displayed entry is selected for editing by double-click or
→<Tab> or →<Shift + Tab> and →<Enter>.
The “Edit value” dialog box that appears contains the name of the selected variable, the data type, the minimum or maximum within which the value can range, and the current value.

→[Factory setting]	sets the current value to the factory setting value.
→[Reset]	sets the current value to the displayed starting value.
→[Write]	writes the value specified under “current value” to the controller variable.
→[Cancel]	closes the box without making any changes.

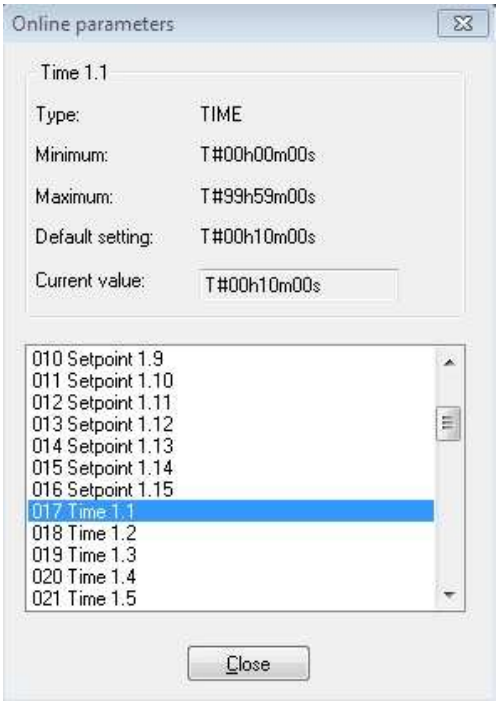
Using the keyboard, <ALT> must be pressed to switch between the parameter window and the menu bar.

6.6 Parameter list



with →Signalwindow→Parameterlist(→Select).

In addition to the parameter window, the commissioning function also provides a list of additional variables which you can change in the controller.



After selecting a variable, its data type, minimum or maximum within which the value can range, and the current value can be displayed. Via mouseclick or →<Tab> you can switch between value input and the window for selecting a new variable.

→[Close] ends inputs in the parameter list.

After selecting the input box for the current value, press <Space> or click the input box to switch over to the “Change value” window. For inputs, see section “6.5 Parameter window”.

Using the keyboard, <ALT> must be pressed to switch between the parameter window and the menu bar.

6.7 Variable list

with →Signalwindow→Variablelist.

Apart from the on-line parameters which can be modified via the parameter windows and the parameter list in the controller, it is also possible to write values on variables which are freely defined (without a leading decimal point) with IBIS-R32.

After selecting a variable, the current value is output in addition to the data type and the factory setting.

It is possible to alternate between the input of the value and the window for selecting a new variable with mouseclick or →<Tab>.

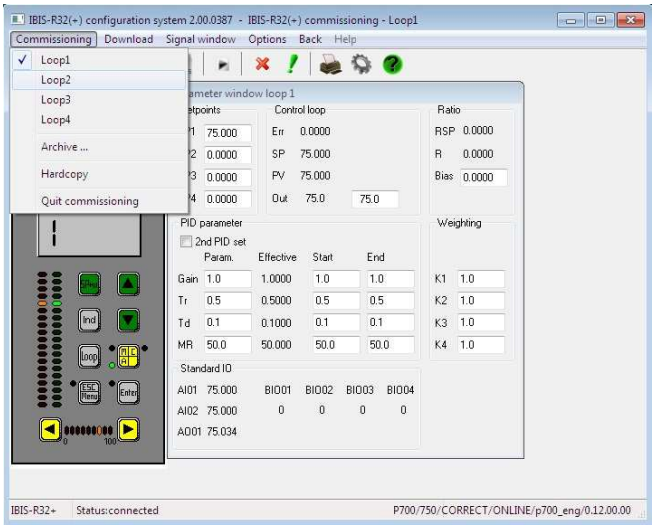
→[Close] terminates inputs into the variable list.

After selecting the input field for the current value, a changeover can be effected with <Space> or by clicking the input field to the window “Change value”. For inputs refer to “6.5 Parameter window”.

6.8 Additional functions

Additional functions can be chosen by →Commissioning and →Download:

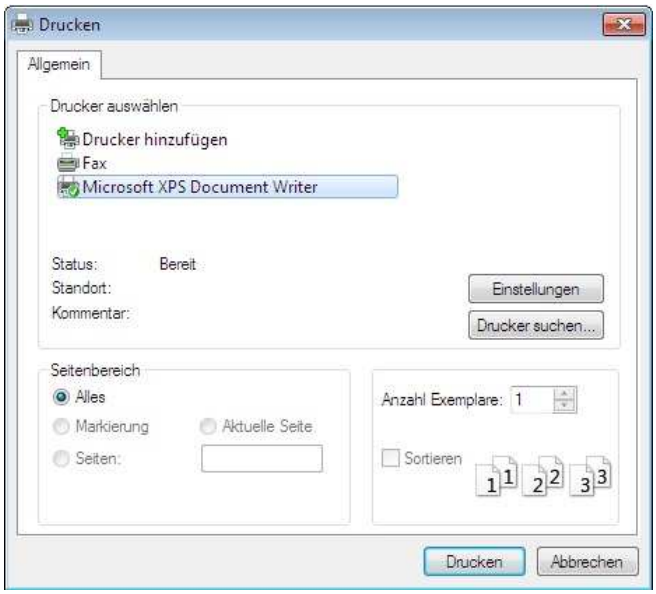
6.8.1 Switching over the displayed control loops



with →Commissioning→Loop1 (to Loop 4).

Value, trend-and parameter window can be switched over to another control loop.

6.8.2 Hardcopy



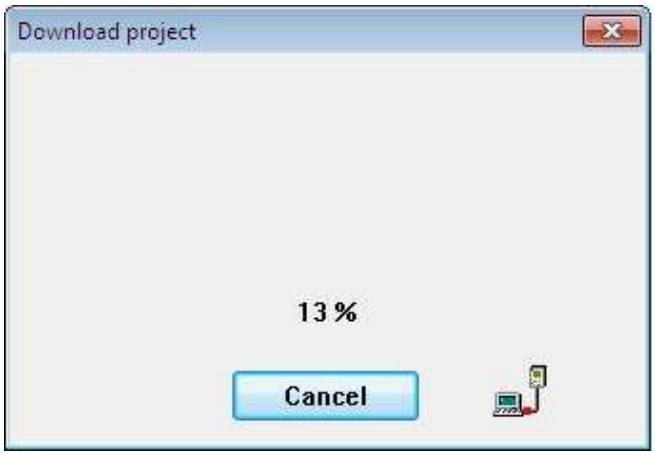
with →Commissioning→Hardcopy.

A hardcopy of a form is printed out using the Windows printer driver.

6.8.3 Loading a configuration into the controller

with →Download→Overallproject

A configuration managed in this project (generated using the hardware configurator, the list configurator or the free-style configuration function) can be downloaded to a controller.



6.8.4 Options

Status
with →Options→Status

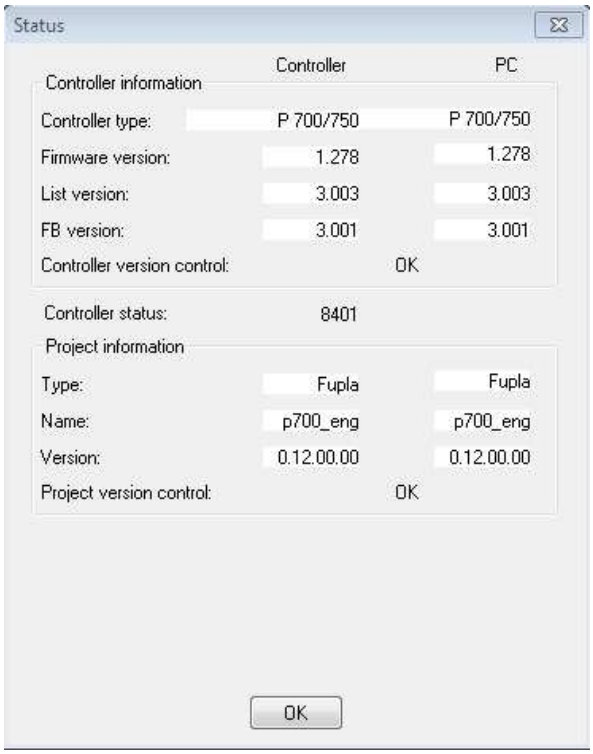
The communication link to the controller can be checked.

This window opens as long as the computer tries to connect to
the controller.

→[Cancel] closes the status query and switches back to the
commissioning window.

This window opens if there is no communication link to a controller.

→[OK] closes the status query and switches back to the commissioning window.



This window opens while a communication link exists.

The status information is provided in two columns for controller and computer. The controller information contains the device firmware with versions for list data and functional modules as well as the project edited with the controller. The computer information contains versions of the list data set with the current version of the library, also including the functional modules and the project just edited.

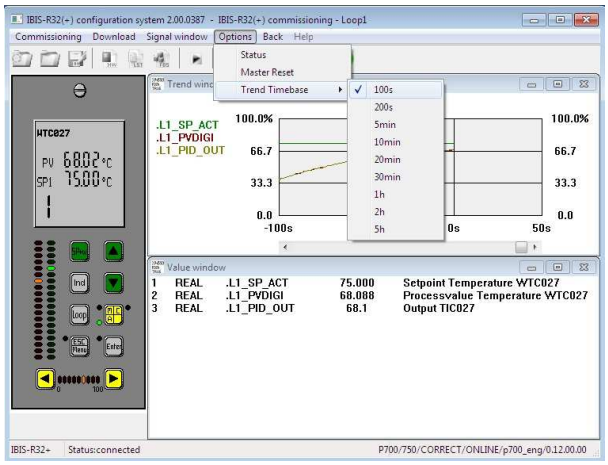
If the control check of the controller version signals an error, then controller firmware and library do not fit together.

If the control check of the version of the project signals an error, the edited project will vary totally from the edited project within the device or the configuration status for the same projects will be different.

→[OK] closes the status query and switches back to the commissioning window.

Basic settings

The basic settings which are possible under →Projectmanagement → COM → Basic settings are also possible here.



Trend time base

The time devoted to the size of the trend window can be changed to different values.

The selected time corresponds to 2/3 of the display size in the trend window, the entire displayable image is about 10 times the size of the time base.

The recording of values for each curve is effected in a second to second rhythm. With the exception of the 100 second time base, however, not every recorded value is displayed. The time period within which a curve value is redisplayed is equal to time base/ 100. The displayed value can be selected separately for each curve between an average value and a min/max value. For min/ max displays the smallest and the biggest interval values are output at any time.

6.8.5 Ending commissioning

with →Commissioning→Quit commissioning.

Switches back from the commissioning window to the project manager.

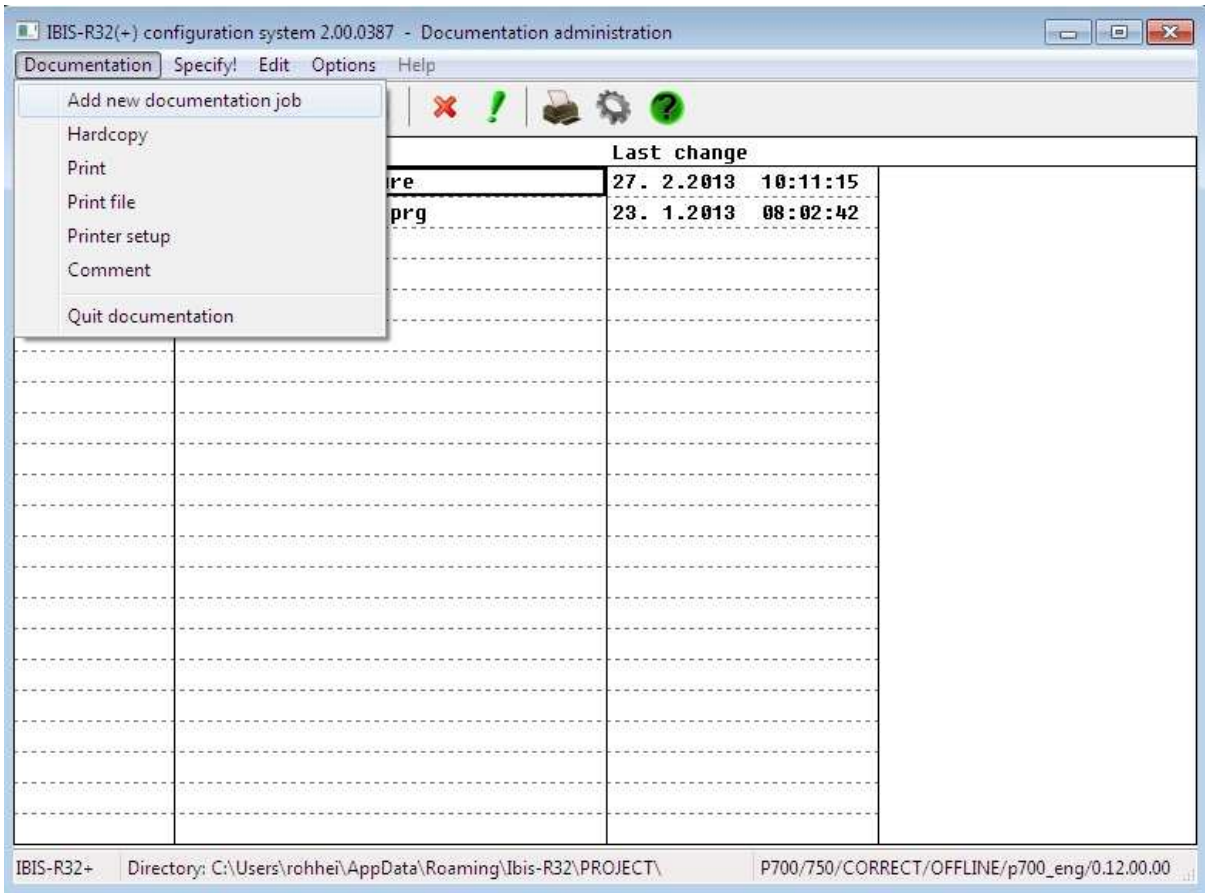
6.8.6 Return

with →Back.

Switches back from the commissioning window to the menu from where commissioning was called.

7 Documentation manager

Table of contents



The Documentation manager provides a routine for generating and managing a complete project in the form of a document comprising module configuration information as well as list and free-style configuration. In addition to printing out a document directly, the document can also be saved to a file.

In order to reduce the volume of information contained in the documentation, specific parts can be chosen and saved.

7.1 Starting the documentation manager

with →Documentation.

You can start the documentation manager from the list configurator, from the project tree or from the FBD Editor routines.

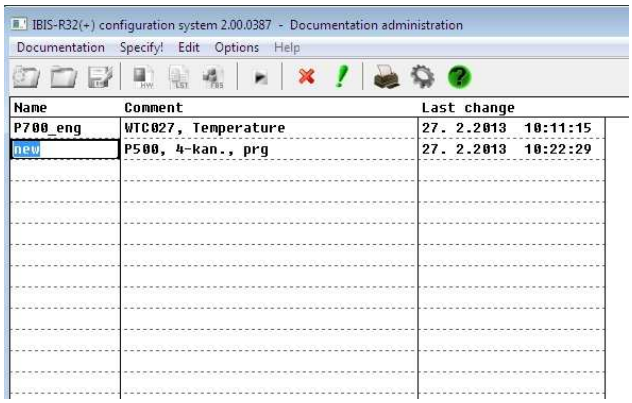
7.2 Documentation job

The printout of documentation is always controlled by a so-called documentation job, i.e., you must always select a job to print out. The job defines the contents of the printout.

To switch between the entry in the window and the menu items <Alt> must be pressed, to move between the entries in the window <Tab> must be pressed.

Every job is valid for all products that are running under IBIS_R32

7.2.1 Generating a documentation job



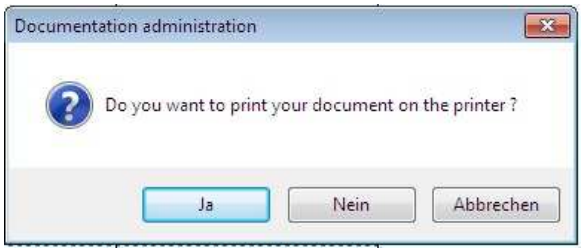
with ⟶ Documentation ⟶ Add new documentation job.

Here a new documentation job can be created. A name can be input which is ended with <ENTER>. Input the job in the first free line. Input is also possible without having to call up the menu item but through direct text input into the first free column “Name”.

If a job has already been selected, its contents and setting can be copied with a new name.

7.2.2 Printing a documentation job

with →Documentation→Hardcopy.



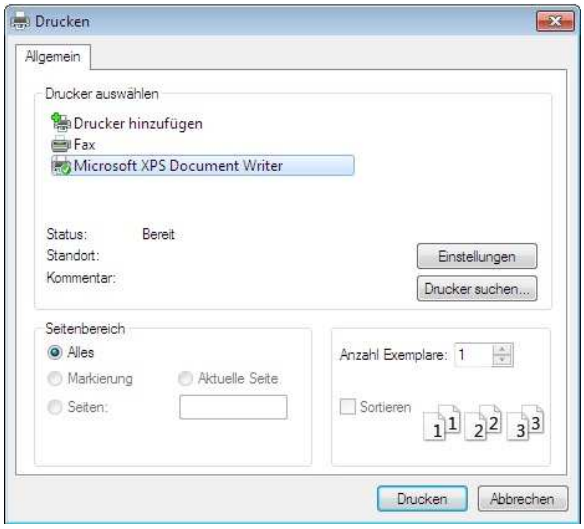
A project is printed out using a selected job. If no job is selected, no action is executed.

After selecting the menu item, you are asked whether you want to print out directly to printer or save to file.

→[Yes] prints out using the printer entered as the standard printer under Windows.

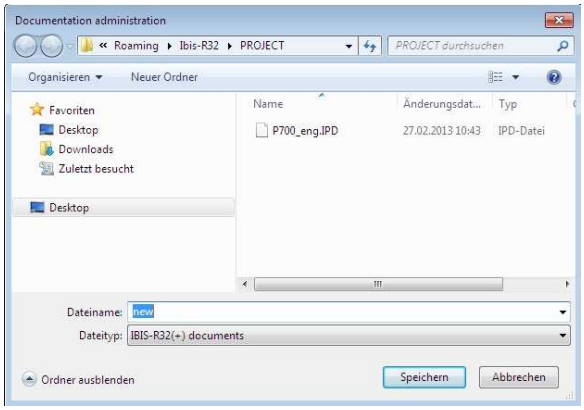
→[No] can save to file. For this reason, a dialog box for entering the necessary information pops up (see below).

→[Cancel] stops the print function.



In this dialog box, the printer type defined for Windows and the printer-specific details, e.g., matching the print quality or selecting the number of copies can be defined.

→[OK] starts the print function.
→[Cancel] stops the print function.
→[Setup] selects and sets up a different printer type than the standard printer.

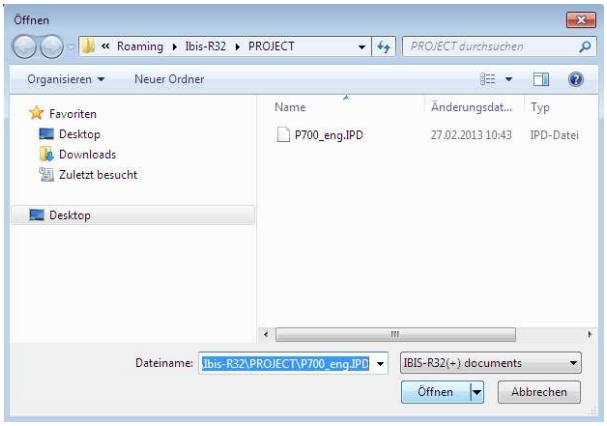


Interrogation for the data required for printing a file is stated in the displayed window.. The default name is the name of a job with the filename extension .IPD and the default path is the working directory entered under Windows. In the dialog box on the left, the file name, the drive and the directory in which the file is saved can be changed.

→[OK] starts saving to file.
→[Cancel] stops the save function.

Since information on the current Windows printer is necessary to save the file, the Windows printer dialog box pops up before the file is generated. After that a message on the actual print job is output.

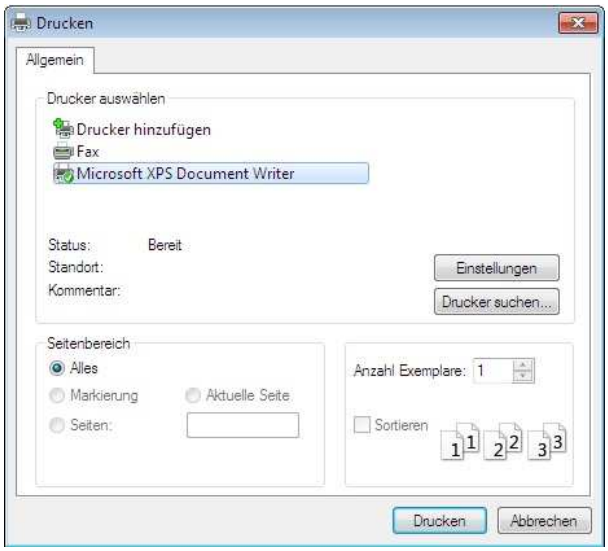
7.2.3 Printing a file



with ⟶ Documentation ⟶ Printfile.

Here a documentation file saved with ⟶Print can be printed out.

Interrogation for the data required for printing a file is stated in the displayed window.. The default name is the name of a job with the filename extension .IPD and the default path is the working directory entered under Windows. In the dialog box on the left, the file name, the drive and the directory in which the file is saved can be changed.



Since information on the current Windows printer is necessary to print the file, the Windows printer dialog box pops up before the file is generated.

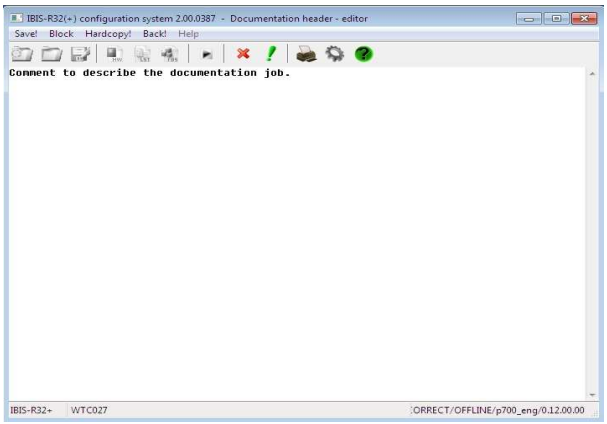
⟶[OK]	starts the print function.
⟶[Cancel]	stops the print function.
⟶[Setup]	selects and sets up a different printer type than the standard printer.

7.2.4 Printer setup

with ⟶ Documentation ⟶ Printer setup.

A printer can be selected and its options can be configured.

7.2.5 Commenting a documentation job



with ⟶ Documentation ⟶ Comment.

A detailed description of a printer macro can be generated and edited.

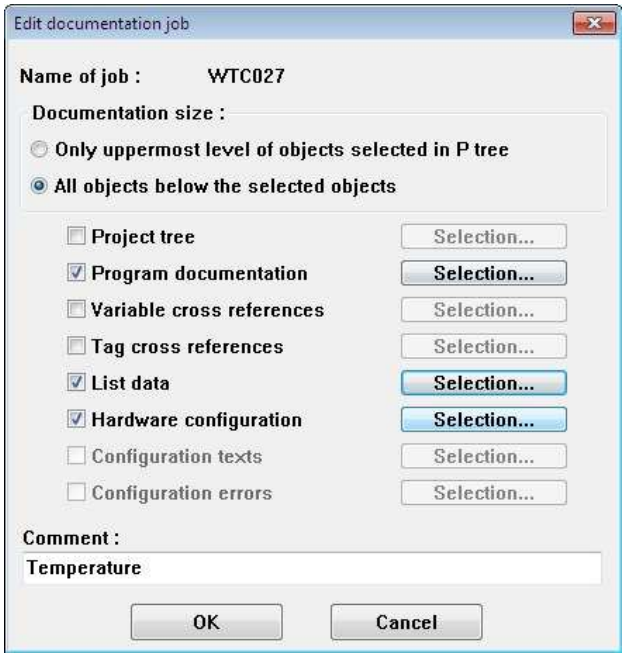
⟶Save! saves the comment.

⟶Block edits the existing text.

⟶Hardcopy! prints the screen contents.

⟶Return! ends the editor.

7.2.6 Detailing a documentation job

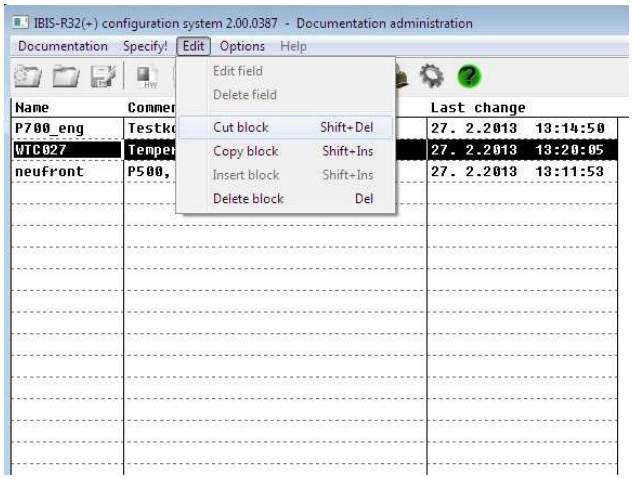


with →Detailling!.

Here a selected job in the “Name” column can be defined for contents. This is to say, the parts to be printed out.

The displayed control boxes besides the black texts are used for switching on/off the parts and for determining the documentation contents. If a control box is selected, a detailed determination of the contents can be made again with the switching field [Selection...], if available.

7.2.7 Editing a documentation job



with →Edit

Texts in the “Name” and “Comment” columns can be edited.

→Field
inputs or changes a text in the selected “Name” or “Comment” columns. The input is terminated by selecting a new field or →<Enter>.

→Delete field

deletes a selected or displayed text in the “Comment” column.

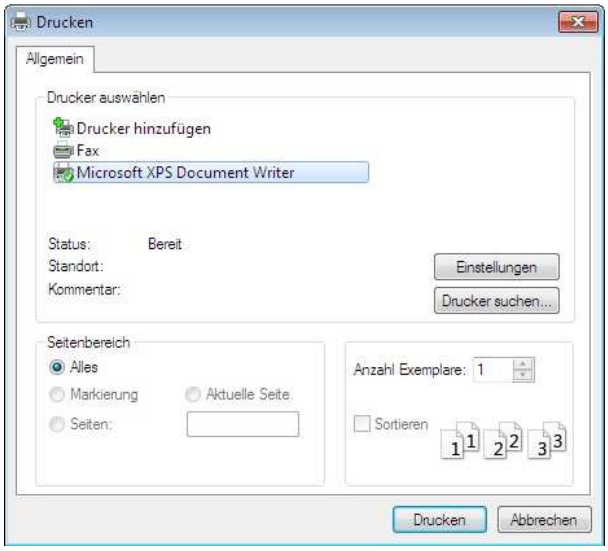
A documentation job can be completely deleted by selecting the said job in the “Name” column and pressing the left mouse button whilst pulling the mouse to the right or by using <Shift + →> to select the line and choosing →Delete field to delete the job.

→Copy
transfers a selected job or a selected block in the list of jobs to a clipboard. The job is not removed from the list.

→Insert
transfers a job or a block in the clipboard to a selected location in the list of jobs.

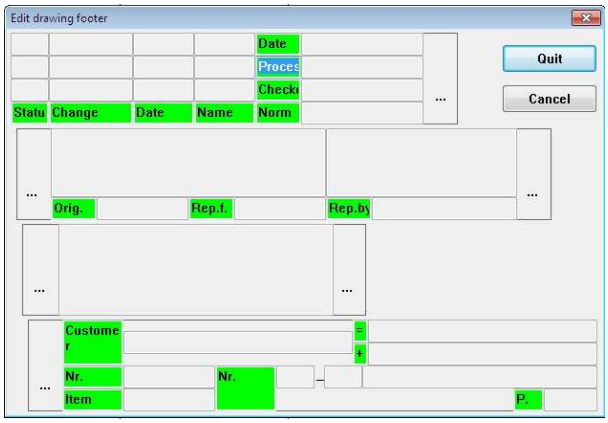
→Delete
deletes a selected job or selected block in the list of jobs from the list of jobs.

7.2.7 Options



Hardcopy

prints out a hardcopy of the screen contents of the documentation manager containing jobs, comments and date of the last change. The printout is executed using a Windows printer driver.



Editing the drawing footer

Changes text entered in the drawing footer, which is highlighted in green.

The texts are valid for the documentation in the entire project.

7.2.8 Return

with →Return.

Switches back to the menu from where the documentation manager was called.

8 Lateral Communication

Table of Contents

8 Lateral Communication
Protrenic 500/550/700, Digitrenic 500/700

8.1 Description

The lateral communication provides a communication system with which signals from several Protrenic and Digitrenic
Controllers can interact with each other without binary and analog input and output connections.

To make this possible, each participating device must be retrofitted with an RS-485 interface via which communication can take place

The communication parameters used here are the online parameters of the communication module which can be adjusted in the list configuration (see also “4 List configuration)”

8.1 Preconditions

8.2.1 Hardware

Each of the participating devices must be equipped with an interface card RS-485 (Cat.No.62619-4-0346257/Suppl. No. 400)

When ordering please also refer to Data Sheet 10/62-6..

For installation pleas refer to the Operating Manuals 42/62-50011 EN (Protrenic) and 42/61-50011 EN (Digitrenic)

These interface card eneble data exchange of up to about 187.500 baud

8.2.2 Software

For Protrenic and Digitrenic lateral communication is possible as of firmware version 1.172 together with free configuration with IBIS-R32 (library version 3.4.0 or higher).

This permits data exchange between a maximum of 6 devices. Each device can transmit up to 64 Bytes to all units participating in the lateral communication.

The access to data transmitted laterally is gained through the variable names, usually used within a project.

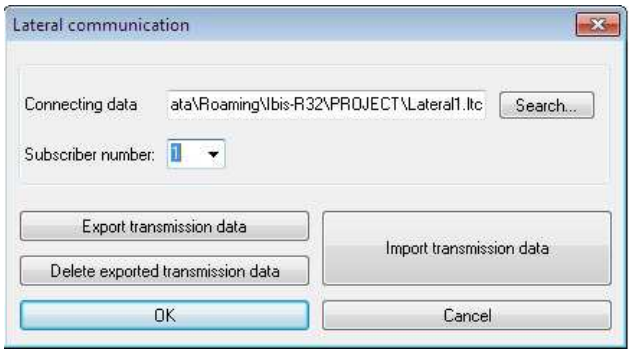
Notice
Lateral communication does not permit vertical communication with higher level systems via the RS-485 interface.

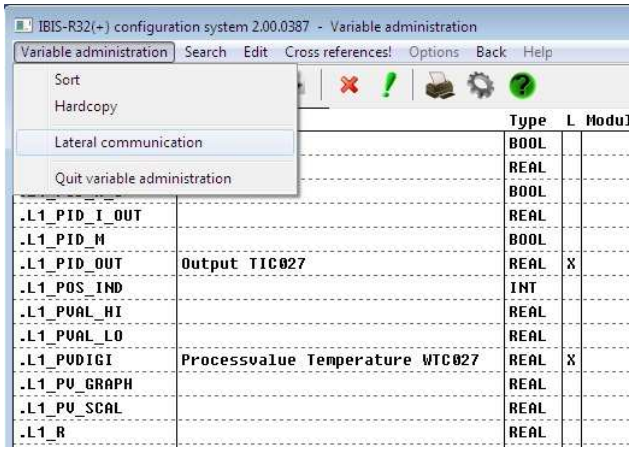
8.2.3 Other information

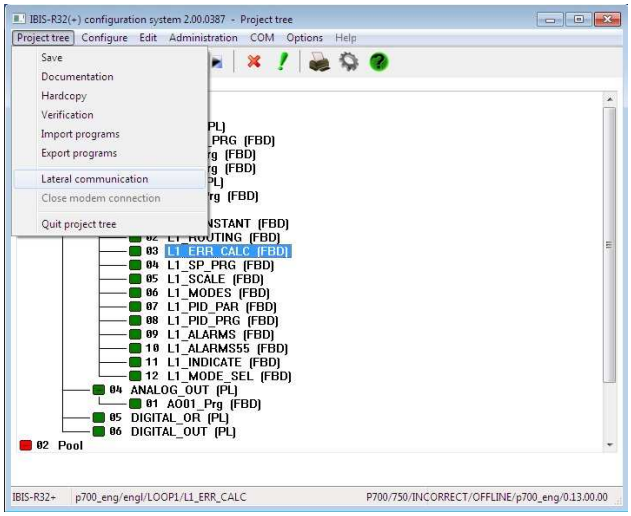
No terminal resistor is required (even for the highest baud rate).

Protrenic, Digitrenic can also be operated in mixed form on a lateral bus.

8.3 Connection file and subscriber number







The central element of the lateral communication is a file in which all variables laterally trasmitted are described. This file is referred to as the **Connection file** and can be provided in default in every project.

To differentiate between up to 6 devices, each device/project must have a specific **Subscriber number** defined for the lateral communication. Permissible are numbers 1 to 6.

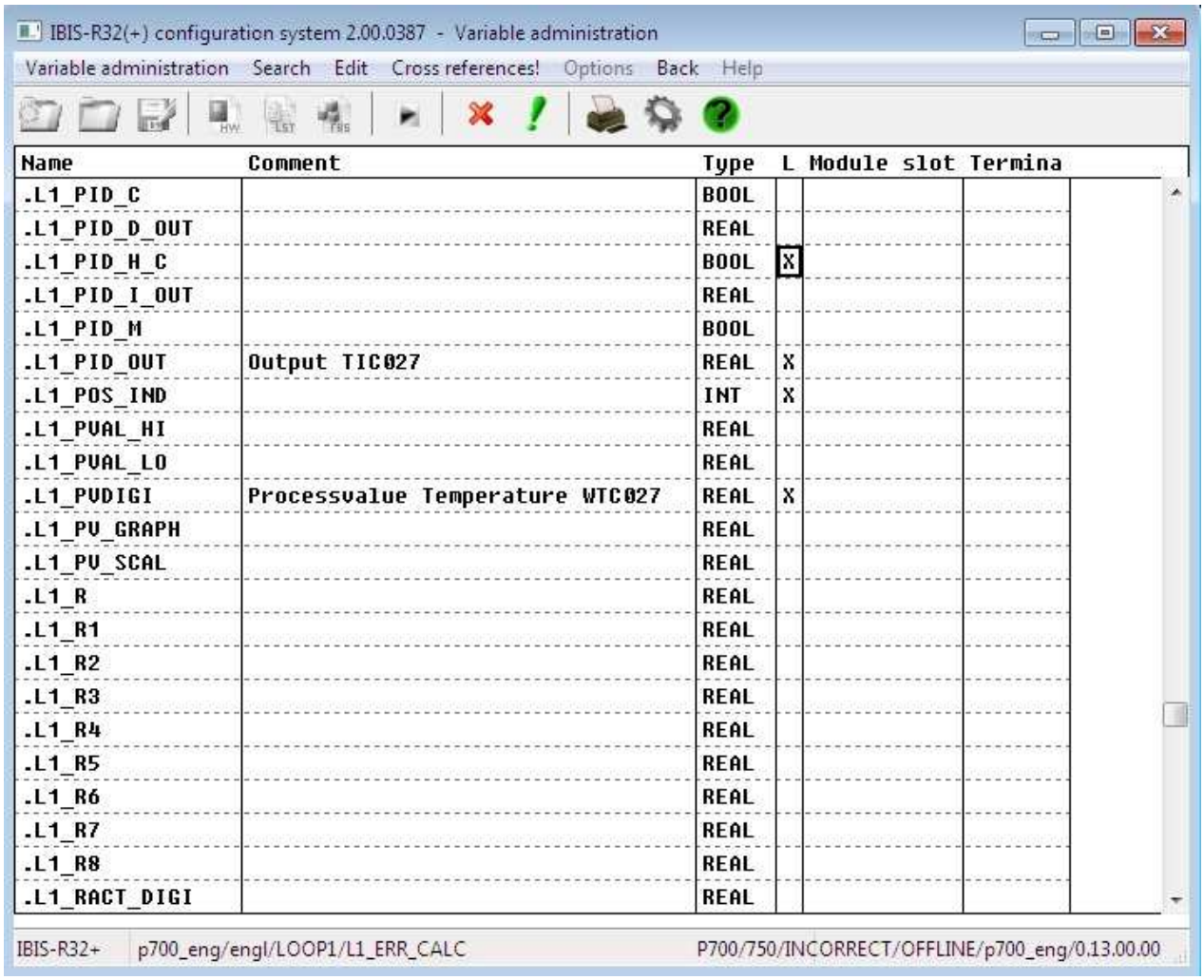
The input mask for the lateral communication in IBIS-R32 can be obtained with
→Free Conf.!→Project-Tree→Lateral Communication
or
→Free Conf.!→Manager→Variable-Manager→Variable-Manager→Lateral Communication.

The possibly pre-existing connection file contained in this input mask which must have an absolute path and the extension .LTC is created with →[OK] upon exiting the input, if not already available. If access to an already existing connection file is desired but its stored path is unknown, this can be determined by changing to a windows-specific selection and input dialog with →[Search...].

The connection file has been provided in the form of a readable ASCII file. This should only be opened with a text editor for control purposes. No modifications may be made.

The subscriber number is predefined with [↓]. A predefined list appears with the possible subscriber numbers for the library. It is from this list that the subscriber number for this project can be selected.

8.4 Definition of transmission behaviour



The question of which variable should be sent from the lateral communication to all subscriber devices can be determined from the variable definition. In it each variable in the column carrying the designation „L“ can be included in the transmitted data by selection.
The selected variables are then included in the connection file during plausibility check or when explicitly exporting the transmitted data into the connection file. A dynamic check of the maximum number of bytes required for the transmission file is not stated on selection but only after plausibility check or during the explicit export of transmitted data into the connection file.
This makes it possible to post-correct previously made invalid inputs which were not corrected immediately.
Should variables from another device be imported, the subscriber number of the sender of the received variables shall be entered into the „L“ column.

The displayed variables can also be limited to only those involved in the lateral communcation by defining a search criterium for the „L“ column. Likewise, it is also possible to sort according to the lateral information in the „L“ column. In such case, the variables which are not part of the lateral communication shall be listed first, followed by the variables with subscriber numbers and finally by ones own transmitted data.

The following number of bytes will be required for transmitting the lateral data:

REAL 4 Byte
DINT 4 Byte
BOOL 2 Byte
INT 2 Byte

In order not to limit the maximum number of transmittable BOOL-type variables to 32 (32 * 2 Byte = 64 Byte), two functional modules (integers to packed Bool and packed Bool to integers) have been provided in library 3.4.0 under →Modules→Stan-dard→Converter for packing or unpacking 16 BOOL variables into an INT variable or from INT variables to 16 BOOL variables. This makes it possible to transmit up to 512 BOOL variables from one subscriber.

8.5 Definition and access to laterally communicated variables

Access to the variables received per lateral communication is obtained by modifying the name. To do this, differentiate between the predefined (a variable name contains as first feature an „.“) and the user-defined variables (no „.“ at start of name).

Predefined variables („.variable name“) of the subscriber with the No. X are provided for all associated projects as „SXvariable name“. The letter S is derived from “System“.

User-defined variables („variable name“, no „.“) of the subscriber with the No. X are provided for all associated projects as „UXvariable name“. The letter U is derived from „User-defined“.

8.5.1 Examples

The device with the subscriber number 3 transmits the predefined and user-defined variables. These variables are available to all subscribers of lateral communication under the following names:

.L1_REGLER_MAN	S3L1_REGLER_MAN
.L1_REGLER_AUTO	S3L1_REGLER_AUTO
X_FIC2745	U3X_FIC2745
Y_FIC2745	U3Y_FIC2745

By means of the supplied code, it is possible to identify similar variable names from different subscribers, should they happen to be in operation at the same time.

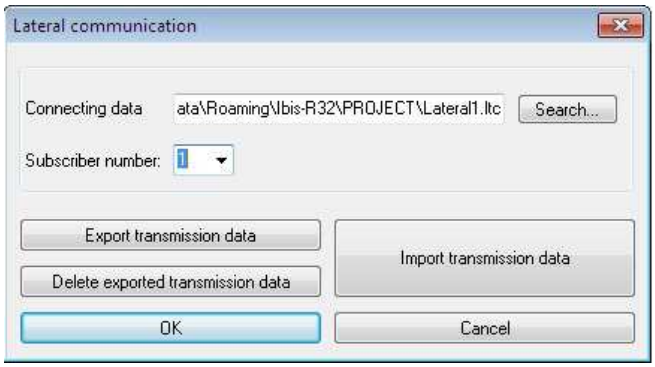
.

After the plausibility check or the the explicit importation of received data via the input mask of the lateral communication, all participating variables are recorded in the variable manager.

As long as no connection file has been recorded, variables with the modified names can be defined as user-defined variables and used for an initial commissioning without lateral communication.

It is only after importing a connection file that these variable names can be checked for their existence and correctness

8.6 Tasks of the input mask Lateral Communication



The connection file is not required constantly during work on a project. This is the case only during plausibility check or when explicitly exporting transmitted data and importing received data. It is then possible to work simultaneously on the projects forming part of the lateral communication from different positions.

→[Search...]
calls up a windows-specific selection and input dialog for selecting the connection file.

→[Export transmitted data]
writes the variables selected in the „L“ column of the variable manager under the subscriber number into the connection file.

→[Delete exported transmitted data]
deletes the variables entered in the connection file for this subscriber.

→[Import received data]
enters the variables of all subscribers listed in the connection file into the variable manager.

→[Abort]
rejects the name changes to the connection file and subscriber numbers and terminates the input dialog.

Attention

All actions undertaken in the meantime such as [export transmitted data], [delete exported transmitted data] and [import received data] will remain valid.

→[OK]
accepts the name changes to the connection file and subscriber numbers and terminates the input dialog.

8.7 Plausibility checks

During the plausibility check of a free configuration, apart from checking errors, all received data in the connection file are automatically copied and entered into the variable manager as variable names with their corresponding subscriber-specific name changes. Likewise, all trasmitted data are entered into the connection file, if the plausibility check does not reveal any errors.

This feature guarantees that after a plausibility check only the most current version of the lateral communication system is used and that the most current version will be provided to other subscribers.

If during the plausibility check a difference is noted between the current variables of the lateral communication in the variable manager and the connection file, an interrogation takes place to ascertain if the new description of the connection file should be used or not.

Should there be differences in the description of one or several subscribers in the time since the last plausibility check, and should no new plausibility check have taken place, this can be determined during commissioning by interrogating the status information for lateral communication. A plausibility check involving a download of the entire project must then be conducted.

8.8 Laterally variable subscriber number and communication status

It is possible to read out the specific subscriber number defined for lateral communication as a value of the variable .LATERALNR.

To enable an interrogation of the communication status of up to 6 subscribers, the value of the variables .LATERAL1 to .LATERAL6 reveals the status of the communication to the respective subscriber.

The following status information is possible:

0	Lateral communication data are being correctly received, are also required for FBD or AL editing and correspond to the structural description which was valid in the connection file during plausibility check.
1	Lateral communication data are being correctly received but do **not** correspond to the structural description provided in the connection file during plausibility check. Data however not required for the editing of FBD or AL.
2	Lateral communication data are being correctly received, do **not** however correspond to the structural description provided in the connection file during plausibility check. This data is nevertheless required for the editing of FBD or AL.
3	Your data was not transmitted for 5 seconds. This is usually the case when a lateral communication subscriber has not yet been connected to a second subscriber via the RS-485.
4	The data of a device participating in lateral communication have not been received for 5 seconds, even though they are required. This happens when the respective subscriber has a fault.
5	Despite participation in lateral communication, no description has been provided for the data to be transmitted.
6	The transmitted data buffer is faulty.
7	Subscriber can neither transmit nor receive lateral communication data.
8	No RS-485 module found. This information refers to the current device, but is usually output in the case of all third party devices.
9	The reporting subscriber has a failure.
255	The subscriber has not been configured for lateral communication.

Likewise, with one exception, all variables transmitted per lateral communication are entered in the variable manager and can be selected and displayed in the value and trend windows through the modified names

S1..., U1..., S2..., U2..., ... , S6..., U6.

The exception is represented by the self-transmitted variables bearing the modifeid names S... and U... Even though these are available in the variable manager, since they themselves cannot receive, they cannot be filled with useful values. If it is necessary to access this information, this can be done by accessing the original variables on the transmitting device marked for lateral communication in the variable manager.

In order to acces the most important information on a device without PC-controller-configuration software, a submenu called „Info Latcom“ has been provided under the menu „Service“.

Upon entering this submenu, Latcom-Nr. X will output your your own subscriber number for the X.

With <Enter> you can gain access to the outputs of the status information of all 6 participating devices as „LatStatus1:“ up to „LatStatus6:“. The respective value is displayed in the upper value line. The usual display of the controlled variable is in this case displayed in the lower value line.

9 Description of the global (predefined) variables used in generating the standard configuration (English variable names)

All the predefined variables in a device have a "." as their first character. These variables cannot be deleted through the Variable Manager.

As a rule all the variables are available (with the exception of hardware-related variables). However, their use and content depend on the use of an FBD programme produced by the standard generation procedure. It is especially important to note that these variables lose their significance if they are modified through free configuration in their control.

Variable name	Meaning
.AA01 (.AO01)	Value (REAL) of AA01 in the dimension [%]. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.AA01R (.AO01R)	Value (INTEGER) of the D/A conversion for AA01. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.AA01ERR (.AO01ERR)	Status (BOOL) of the AA01 load monitoring. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.AE01 (.AI01)	Value (REAL) of AE01 in the configured physical dimensions. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.AE01ERR (.AI01ERR)	Status (BOOL) of the fault status message for AE01. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.AEKTY0 (.AICJCM0)	Value (INTEGER) of the terminal temperature measurement of the basic model. Each analogue input module (.AEKTY1 -.AEKTY7) that is included in the hardware configuration has a corresponding variable.
.AE01R (AI01R)	Value (INTEGER) of the A/D conversion for AE01. All analogue outputs that are included in the hardware configuration have a corresponding variable.
.A_DIM	Contains the ASCII characters for dimensioning the IND loop display in the form of a string of 4 characters. Not accessible via the variable management
.A_LOOP (.D_LOOP)	Number (INTEGER) of the displayed control loop (currently 0..3).
.A_MAXWERT (.D_MAX)	Maximum value (REAL) of the value in the display that is currently selected by the <IND> key and enabled for modification. This value is read and evaluated by the operation.
.A_MINWERT (.D_MIN)	Minimum value (REAL) of the value in the display that is currently selected by the <IND> key and enabled for modification. This value is read and evaluated by the operation.
.A_NAME	Contains, in the form of a 3-ASCII-character string, the text describing the value for the IND loop display. Not accessible via the variable management
.A_TAST_M (.D_KEY_M)	Status (BOOL) of the < > key. Evaluates to TRUE when the key is pressed and to FALSE when it is not pressed. This status is set by the operation.
.A_TAST_W (.D_KEY_L)	Status (BOOL) of the < > key. Evaluates to TRUE when the key is pressed and to FALSE when it is not pressed. This status is set by the operation.
.A_WERT (.D_VALUE)	Value in the display currently selected by the <IND> key. This value is read by the operation and written if any change is effected through the <◄> and <►> keys.

.BA01 (.DO01)	Value (BOOL) of the internal status of BA01. The active sense has not yet been taken into account in this variable. All binary outputs that are included in the hardware configuration have a corresponding variable.
.BA01R (.BO01R)	Value (BOOL) of the terminal status of BA01. The active sense has already been taken into account in this variable. All binary outputs that are included in the hardware configuration have a corresponding variable.
.BE01 (.DI01)	Value (BOOL) of the internal status of BE01 that takes the active sense into account. All binary inputs that are included in the hardware configuration have a corresponding variable.
.BE01R (.BI01R)	Value (BOOL) of the terminal status of BE01. All binary inputs that are included in the hardware configuration have a corresponding variable.
.BEINV1 to (.DIINV1...6) .BEINV6	Reserved (BOOL).
.BESYNC (.DISYNC)	The function of this variable is to enable the function block ANZSL to read variables in a synchronised manner. Is set to TRUE (BOOL) by the operation and cleared (.BESYNC = FALSE) by the function block ANZSL.
.COMAKTIV (.COMAKTIVE)	Reserved (BOOL).
.CONST1_B0, .CONST2_B0	BOOL variables with the value 0.
.CONST1_B1, .CONST2_B1	BOOL variables with the value 1.
.CONST_I0	INTEGER variable with the value 0.
.CONST_I1	INTEGER variable with the value 1.
.CONST_L0	DOUBLE-INTEGER variable with the value 0.
.CONST_L1	DOUBLE-INTEGER variable with the value 1.
.CONST_R0	REAL variable with the value 0.0.
.CONST_R1	REAL variable with the value 1.0.
.CONST_R100	REAL variable with the value 100.0.
.DN_A (.D_DIGITS)	Number (INTEGER) of places after the decimal point for the displayed value. This variable is written by the operation from the function block ANZSL for displaying values.
.FLAG_1 to .FLAG_6	Status TRUE switches the corresponding flag on the front panel, while FALSE switches it off.
.INDS_LOOP1 to .INDS_LOOP4 (.L1.. L4_POS_IND)	Selected position (INTEGER) of the IND loop. This variable is written for a specific loop by the function block ANZSL.
.L1_B01_F01 (.L1_B01_Q01)	Contains the value (INTEGER) that was configured for L1_B01_F01 in the list configuration process as a response number.
.L1_A_VORB (.L1_A_PREP)	Indicates operating mode AUTOMATIC for the LED's next to the <MAC> key if the controller is operating in that mode. If this status is different from the status .L1_REGLER_AUTO and if the variable .MACCOUNT is set to TRUE, the associated LED A next to the <MAC> key will flash.
.L1_B1	Switch status of the assigned value of L1_B04_F06.
.L1_BA_YOUT (.L1_OUT_M)	Contains the value in [%] of the correcting variable to be output in MANUAL operating mode.

.L1_BETART_UM (.L1_MODE_SW)	If the <MAC> key is activated, the operation sets this variable (BOOL) to true. When does it get set back to FALSE again.
.L1_C_VORB (.L1_C_PREP)	Indicates operating mode CASCADE for the LED's next to the <MAC> key if the controller is operating in that mode. If this status is different from the status .L1_REGLER_C and if the variable .MACCOUNT is set to TRUE, the associated LED C next to the <MAC> key will flash.
.L1_D	Contains the value (REAL) of the resulting controlled variable in physical units for further processing as an injection for the D-action of the PID controller.
.L1_D_PRZ (.L1_D_PRC)	Contains the value (REAL) of the resulting controlled variable in physical units as an injection for the D-action of the PID controller.
.L1_ES1 to (.L1_IC1…4) .L1_ES4	Contains the value of the variable assigned via L1_B04_F01 to F04.
.L1_ES5 (.L1_IC5)	Reserved (REAL).
.L1_GW1_OUT to .L1_GW4_OUT (.L1_AL1…4_OUT)	Status TRUE indicates that the associated limit value has been violated. If the limit value has not been violated, this variable shows FALSE.
.L1_HAND_M (.L1_M_INC)	If the step controller is configured and the <▸> key is activated, the operation sets this variable to TRUE. It is set to FALSE when the key is released.
.L1_HAND_W (.L1_M_DEC)	If the step controller is configured and the <◂> key is activated, the operation sets this variable to TRUE. It is set to FALSE when the key is released.
.L1_KP_STEUER (.L1_GAIN)	Contains the value (REAL) of the active gain factor of the PID controller.
.L1_KS_STEUER (.L1_DTP_GAIN)	Contains the value (REAL) of the default amplification constant of the PID controller for processing the Smith predictor.
.L1_K1 to (.L1_CONST1…4) .L1_K4	Contains the value (REAL) of the weighting factors K1 to K4 that can be accessed.
.L1_LAMBDA	Reserved (REAL).
.L1_LASTBA (.L1_LAST_MODE)	Variable (INTEGER) in which the controller operating mode in use at the time of a voltage failure is made available for the controller's restart.
.L1_MAN_AUTO	Status TRUE indicates whether the control loop is operating in MANUAL or AUTOMATIC mode.
.L1_MAN_CAS	Status TRUE indicates whether the control loop is operating in MANUAL or CASCADE mode.
.L1_M_VORB (.L1_M_PREP)	Indicates operating mode MANUAL for the LED's next to the <MAC> key if the controller is operating in that mode. If this status is different from the status .L1_REGLER_MAN and if the variable .MACCOUNT is set to TRUE, the associated LED M next to the <MAC> key will flash.
.L1_OBJNR_GW (.L1_OBJNO_AL)	Reserved (REAL).
.L1_OBJNR_PID (.L1_OBJNO_PID)	Reserved (REAL).
.L1_OBJNR_REGBA (.L1_OBJNO_IND	Reserved (REAL).
.L1_PID_D_OUT	Contains the value (REAL) of the differential component of the controller's output variable in [%].
.L1_PID_I_OUT	Contains the value (REAL) of the integral component of the controller's output variable in [%].
.L1_PID_PS (.L1_PID_H_C)	In control loops with separate PID parameter sets for heating and cooling this variable (BOOL) is used to indicate the required parameter set.

.L1_PID_Y-OUT (.L1_PID_OUT)	Contains the most recent value output by the PID function block regardless of the operating mode.
.L1_REGLER_AUTO (.L1_PID_A)	
.L1_REGLER_C (.L1_PID_C)	
.L1_REGLER_MAN (.L1_PID_M)	Indicate the mode in which the control loop is operating - AUTOMATIC (AUTO), CASCADE (C) and MANUAL (MAN).
.L1_SKALV (.L1_SCAL_R	Reserved (REAL).
.L1_SPAKTIV (.L1_SFT_ACTIV)	At the start of automatic parameter definition the operation sets this variable (BOOL) to TRUE, and at the end of that process to FALSE.
.L1_T1_STEUER (.L1_DTP_T1)	Contains the value (REAL) of the default time constant [min] of the PID controller for processing the Smith predictor.
.L1_TN_STEUER (.L1_T_RESET)	Contains the value (REAL) of the current reset time [min] of the PID controller.
.L1_TT_STEUER (.L1_DTP_TT	Contains the value (REAL) of the default dead time [min] of the PID controller for processing the Smith predictor.
.L1_TV_STEUER (.L1_T_DERIV)	Contains the value (REAL) of the current lead time [min] of the PID controller.
.L1_V (.L1_R)	Contains the value (REAL) of the ratio set point. Set by the operation.
.L1_VISTDIGI (.L1_RACT_DIGI)	Actual value (REAL) of the ratio control as a proportional value (not the actual value of the controlled variable in physical units).
.L1_V_F (.L1_R_FV)	Status TRUE signifies that the controller is operating as a fixed-value controller despite the configured ratio control. FALSE signifies that the controller is operating as a ratio controller. Set by the operation.
.L1_WAKT (.L1_SP_ACT)	Contains the value (REAL) of the target set point in physical units after the ramping function.
.L1_WANA (.L1_SP_GRAPH)	Contains the value (REAL) of the set point used for the additional processing to produce the analogue bar graph display.
.L1_WANA_SKAL (.L1_SP_SCAL)	Contains the value (REAL) of the set point used in a range from 0.0 to 1.0 for feeding in to the analogue bar graph display.
.L1_WCOMPUTER (.L1_SPCOMP)	Contains the value (REAL) of the values that can be described via the port via L1_B05_F06.
.L1_WDIG (.L1_SPDIGI)	Contains the value (REAL) of the set point used in physical units.
.L1_WEXT (.L1_SPEXT)	Contains the value (REAL) of the variable assigned via L1_B05_F06 as the external set point, or the set point predefined by a master controller for a slave controller in a cascade control loop.
.L1_WEXT_AKTIV (.L1_SPEXT_ACT)	Status TRUE signifies that the external set point is active.
.L1_WSOLL0 to .L1_WSOLL3 (.L1_SP1...SP4)	Variables (REAL) for altering the 4 set points used internally. L1_WSOLL0 for set point 1, L1_WSOLL3 for set point 4.
.L1_WW (.L1_SPTARGET)	Contains the value (REAL) of the target set point in physical units before the ramping function. Set by the operation.
.L1_W_FOLGE (.L1_OUT_TRACK_C)	Contains the value (REAL) of output variable tracking for the master controller by the set point of a slave controller in a cascade control loop.
.L1_W_STATUS (.L1_SP_STATUS)	Reserved (BOOL).
.L1_XANA (.L1_PV_GRAPH)	Contains the value (REAL) of the resulting controlled variable used for the additional processing to produce the analogue bar graph display.
.L1_XANA_SKAL (.L1_PV_SCAL)	Contains the value (REAL) of the resulting controlled variable in a range from 0.0 to 1.0 for feeding in to the analogue bar graph display.

.L1_XDIGI (.L1_PVDIGI)	Contains the value (REAL) of the resulting controlled variable in physical units.
.L1_XW .L1_XW_EU (.L1_DEV_EU)	Contains the value (REAL) of the calculated control deviation in physical units.
.L1_XW_PRZ (.L1_DEV_PRC)	Contains the value (REAL) of the calculated control deviation in [%].
.L1_Y0_STEUER (.L1_MR)	Contains the value (REAL) for the current position of the PID controller in the operation.
.L1_YCOMPUTER (.L1_OUTCOMP)	During DDC control the correcting variable in [%] that is to be output through the port is written to this variable (REAL).
.L1_YEXT (.L1_OUTEXT)	Reserved (REAL).
.L1_YHAND (.L1_OUT_MVAL)	When in MANUAL mode, the operation writes to this variable the value (REAL) of the correcting variable in [%] to be output.
.L1_YIN (.L1_OUT_IN)	REAL
.L1_YMAX (.L1_OUTMAX)	Contains the value of the variable assigned via L1_B10_F08.
.L1_YMAX_BR (.L1_OUTMAX_SC0)	This variable (REAL) is used for calculating the upper output limit for processing in the PID block, for use by the override controller when in override control.
.L1_YMAX_HR (.L1_OUTMAX_PC)	This variable (REAL) is used for calculating the upper output limit for processing in the PID block, for use by the main controller when in override control.
.L1_YMIN (.L1OUTMIN)	Contains the value of the variable assigned via L1_B10_F09.
.L1_YMIN_BR (.L1_OUTMIN_SC)	This variable (REAL) is used for calculating the lower output limit for processing in the PID block, for use by the override controller when in override control.
.L1_YMIN_HR (.L1_OUTMIN_PC)	This variable (REAL) is used for calculating the lower output limit for processing in the PID block, for use by the main controller when in override control.
.L1_YSRUECK (.L1_OUT_FB)	Contains the value of the variable assigned via L1_B10_F04 for position feedback for the step controller.
.L1_YTRACK (.L1_OUTTRACK)	Contains the value of the variable assigned via L1_B10_F10.
.MACCOUNT	If the <MAC> key is activated the operation sets this variable (BOOL) to TRUE. If this variable is later set to FALSE the operating mode is really switched over.
.NOCONNECT_B0 to .NOCONNECT_B5	BOOL variables that are only needed in order to generate the standard configuration with no errors. These variables may be deleted in a free configuration along with their associated links.
.NOCONNECT_I	INTEGER variable that is only needed in order to generate the standard configuration with no errors. These variables may be deleted in a free configuration along with its associated links.
.NOCONNECT_L	DOUBLE-INTEGER variable that is only needed in order to generate the standard configuration with no errors. These variables may be deleted in a free configuration along with its associated links.
.NOCONNECT_R0 to .NOCONNECT_R3	REAL variables that are only needed in order to generate the standard configuration with no errors. These variables may be deleted in a free configuration along with their associated links.
.PG_BETRIEB (.SPG_OPERATE)	This variable is set to status TRUE in order to start the time scheduler. FALSE causes the time scheduler to halt. Set by the operation.
.PG_LAUF (.SPG_RUNTIME)	The total run time of the programme being processed is displayed in this variable (DINTEGER) as a millisecond count.
.PG_NR_AKT (.SPG_NO _PG	The number of the time scheduler programme currently being processed is shown in this variable (INTEGER).

.PG_NR_SEL (.SPG_NR_SEL)	The operation sets this variable (INTEGER) to the number of time scheduler programme selected by the operation.
.PG_RESET (.SPG_RESET)	If the value TRUE is written to this variable while the time scheduler is not running then the programme that has been processed will be reset to start again. Set by the operation.
.PG_SCHNELL (.SPG_FAST)	If the value 1 is written to this variable while the time scheduler is not running then fast forward will be activated. If this variable is set to 2 then fast rewind will be activated. Setting it to 0 cancels either fast forward or fast rewind. Set by the operation.
.PG_SEG (.SPG_SEG)	This variable (INTEGER) shows the number of the segment most recently processed by the time scheduler.
.POS_WW (:POS_SP)	Variable (BOOL) that is set by the operation in order to switch over to displaying the current set point in the IND Loop.
.POS_Y (.POS_OUT)	Variable (BOOL) that is set by the operation in order to switch over to displaying the value of the output variable in the IND loop. When is it set to false? If a step output is configured and no position feedback signal is present, then the switchover will not be executed.
.PRGM_SEL (.SPG_SEL)	When set to TRUE this variable indicates that the time scheduler has been selected in a loop as the set-point source.
.PRG_BA1…4 (.SPG_DO1…4)	This variable (BOOL) contains the allocated status in a programme segment of the associated binary trail.
.PRG_ENDE (.SPG_END)	A status of TRUE signifies the completion of the programme being processed. If the status is FALSE this signifies that the selected programme is being processed.
.SLH_LOOP1…4 (. L1…4_SLH)	If these variables (BOOL) are positive, then the control loop display on the front panel will switch over to the associated (control) loop.
.STEPS_B (.STEPS_IND_B)	If this variable (BOOL) is positive, the result will be a switch in the IND loop to the previous display position.
.STEPS_F (.STEPS_IND_F)	If this variable (BOOL) is positive, the result will be a switch in the IND loop to the next display position.
.STEPW_B (STEP_SP_B)	If this variable (BOOL) is positive, the result will be a switch in the IND loop to the previous display position. There is no associated change of set point.
.STEPW_F (STEP_SP_F)	If this variable (BOOL) is positive, the result will be a switch in the IND loop to the next display position. There is no associated change of set point.
.TAB1 to .TAB4	Reserved (REAL).
.TAB4AE (.TAB4_IC5)	Contains the value (REAL) of the variable assigned via L1_B04_F05 (Q05).
.TMPOR	Temporary variable (BOOL) used in the generation of the standard configuration if more statuses have been fed to one binary output than the number of lines in an FBD programme.
.T_ENTER (.KEY_ENTER)	Pressing >Enter< button for mode switching in a multiple cascade appears as a state (BOOL)
.VERST (.ADJUST)	Contains the versatility attribute (INTEGER). It is used by the operation to determine whether or not a variable in the IND loop is versatile. This value is set by the function block ANZSL display loop (0 = no change poosible, 1 = change with <▲ >/<▼ >, 2 = change with <◄ >/<▼ >)
.WW_LOOP1…4 (L1…4_SP_SEL)	Contains the index (INTEGER) of the selected set point (0 = W1, 1 = W2, 2 = W3, 3 = W4, 4 = Wext, 5 = Wcomputer, 6 = time scheduler).

.WW_UM (.IND_SPW) Boolean variable that is set by the function block ANZSL while the set point is being changed over through the <SP-W> key. If this variable is TRUE the three-character text describing the value will flash.

.ZK01 (.SC1) Result of the calculation of status correction 1 in physical units.

.ZK02 (.SC2) Result of the calculation of status correction 2 in physical units.

The variables from **loop 2** to **loop 4** relate to those with variable names .L2_... to .L4_...

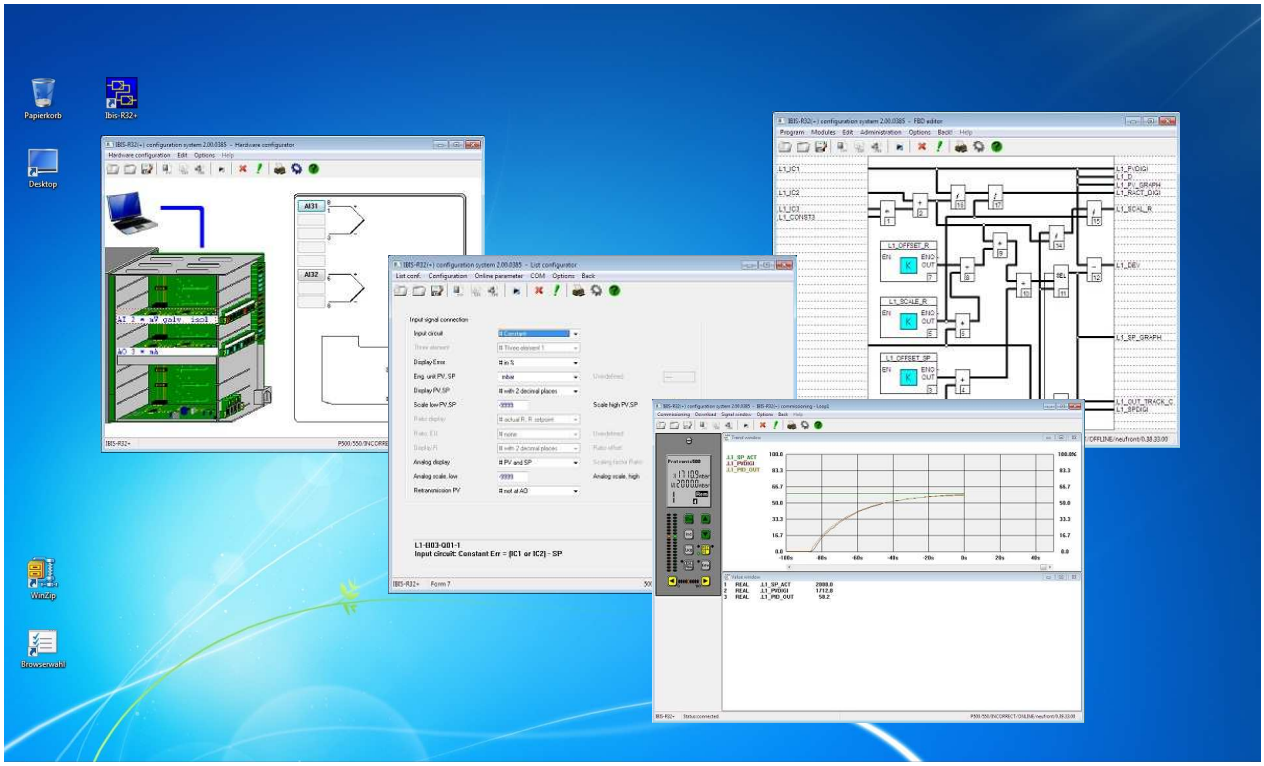
IBIS-R32+

Configuration and parameter setting software
for Digitrenic 500/700
and Protrenic 100/500/550/700
since version 2.00.360

Supplement to Manual

42/62-52030-1 Z1 EN

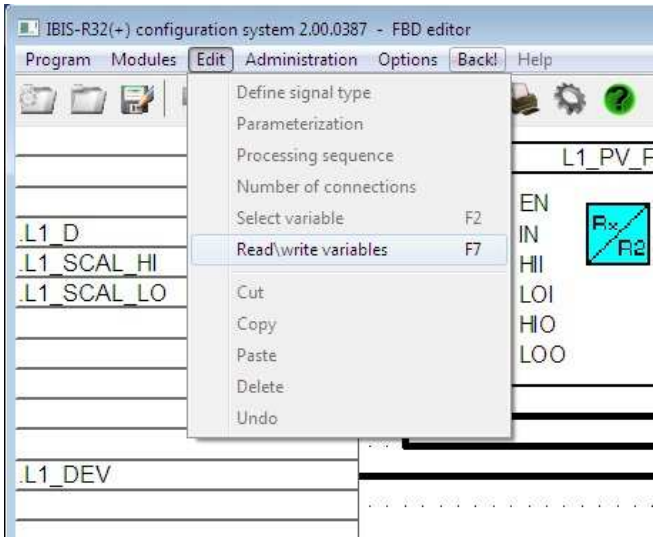
Rev. 01

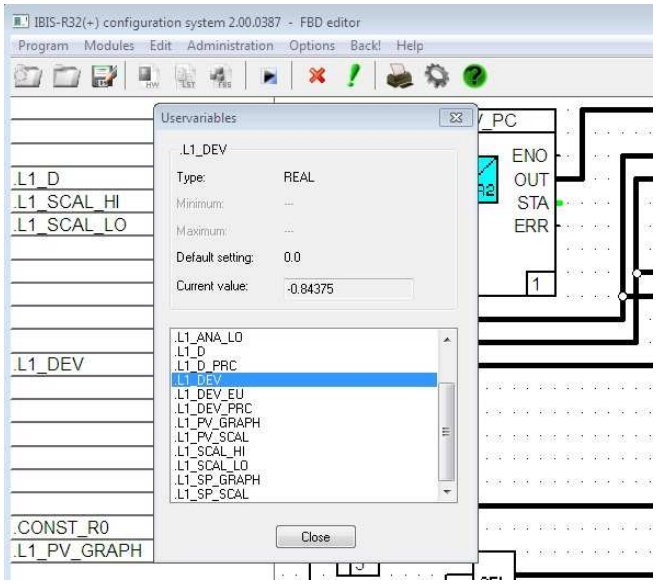


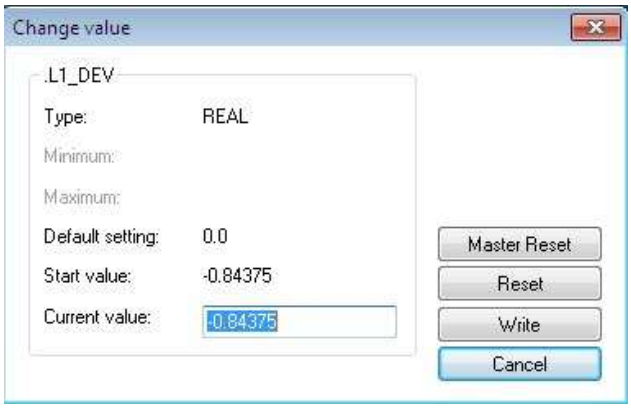
ENA Control

Extension of the Commissioning

Reading/Writing of variables in the editors of the free configuration







For the sake of quick commissiong of projects with free configuration, directly access to the variables of a coupled controller in the editors of FBS and IL can be gained without having to state these in the value window of the commissioning feature.

There are 2 possibile selection methods:

Firstly, you can select a variable of your choice (with a simple click on the variable) and then call up the function “Read/write variables” in the “Edit” menu, in order to display the selected read value online.

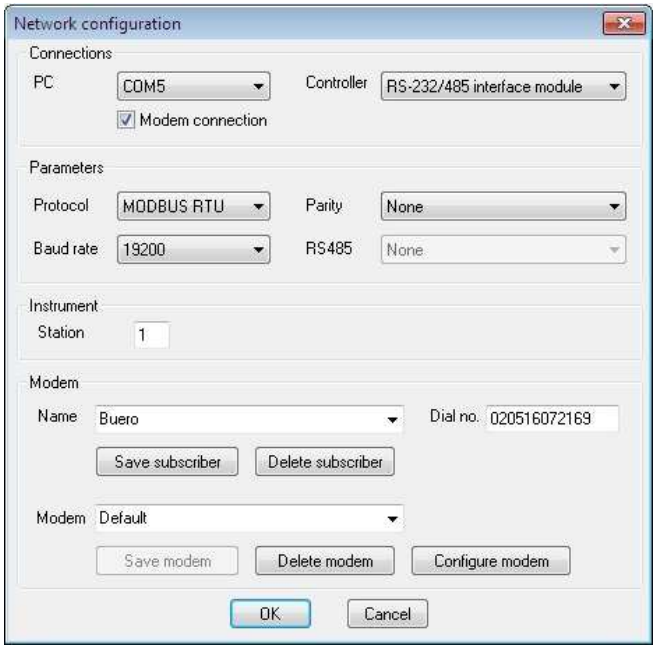
After choosing the variable, the function key <F7> can also be used for accelerated selection.

The second possibility involves calling up this menu item or the accelerated selector without previously selecting a variable.

In both cases, a window containing all variables used in the program -both user-defined and predefined -is displayed. The variable to be read online can be selected from this list. If a variable had been previously selected, this shall be displayed with its data type and online value. The contents of the window correspond to those in the window of the variable list for commissioning.

The value is constantly updated. Within the list, changing over to another variable is possible at any time. This selection can also be made using the <↑> and <↓> keys.

Access to controller via modem



A modem connection can also be used to gain online access to the unit. With it, systems can also be commissioned without being on site. No PC is required on the controller side for this type of communication. The modem can be operated directly with the connected controllers. Should more than one controller be connected, the RS-232 interface of the modem must be adapted to match to RS-485 with the help of an adapter.

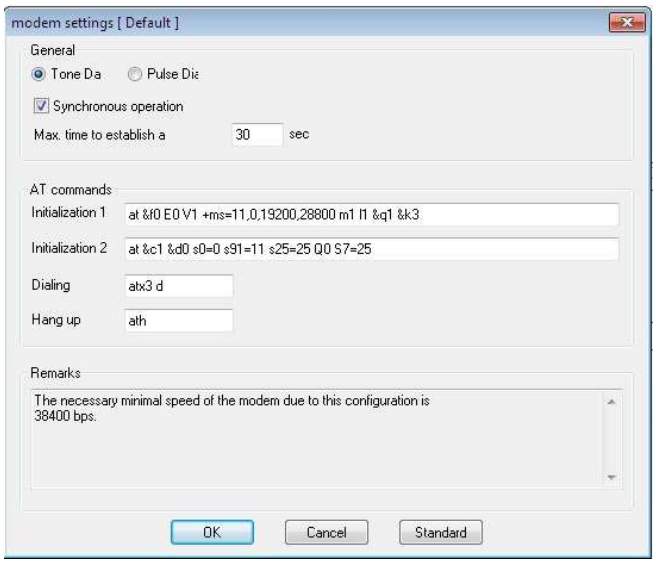
The telephone connection via modem can be established and controlled directly via IBIS-R32. Additionally, the IBIS-R32 also enables the application of a telephone directory.

In order to configure the modem connection, the “Communications Parameter” dialog must be called up.

In the field marked “Connections”, indicate whether a modem can be used for obtaining the next online access by ticking off. This setting remains valid until the next change, if the dialog is terminated with [OK]. A modem connection can only be established via an apparatus equipped with an interface module. Modem ope-ration via the TTL interface is not possible.

If a modem connection was ticked off in the given field, the “Modem” becomes activated. In it, a name for the system to be addressed should be selected from a list. This list represents a telephone directory. The appropriate telephone number and the name itself can be input or modified here. If a name or telephone number was input or modified, this can be stored in the telephone directory by pressing [Save subscriber]. Selected names can be deleted again from the telephone directory, including their corresponding numbers, by pressing [Delete subscriber].

The H-288e made by Messrs Häussler has been set at the factory. If new settings are required for a different modem, this should be changed via [Configure modem]. The appropriate parameters for telephone dialing can also be set in the same way.



The basic parameters for creating a telephone connection can be entered into the “General” field. The selection of either tone or pulse is determined by the type of telephone system used on the PC side. If the type used is unknown, contact the manufacturer. The “Maximum time for dialing” is the time required by the modem on the controller side to respond after the modem has been dialed, e.g. through a call for start-up via modem. If the modem responds, communication is exchanged between the modems on the type of data connection to be made. This must be achieved within the time frame “Maximum time for establishing a connection”, otherwise the entire dialing exercise shall unsuccessful and subsequently be aborted.

In addition to the general parameters, the “AT command sequences” for controlling the modem on the PC side for initialization, dialing and engagement can be configured. These settings can be accepted into the INI file of IBIS-R32 by pressing [OK]. If modifications must be made in order to match to other modems, such modifications are normally indicated in the manual of the modem in question. For operation with IBIS-R32, a synchronous data operation must be configured.

The modem H-288e made by Messrs Häussler has been set at the factory. These settings can be called up at any time by pressing [Standard] in the input windows of the “AT command sequences”, and the required inputs of the INI file can be overwritten by quitting the dialog with [OK].

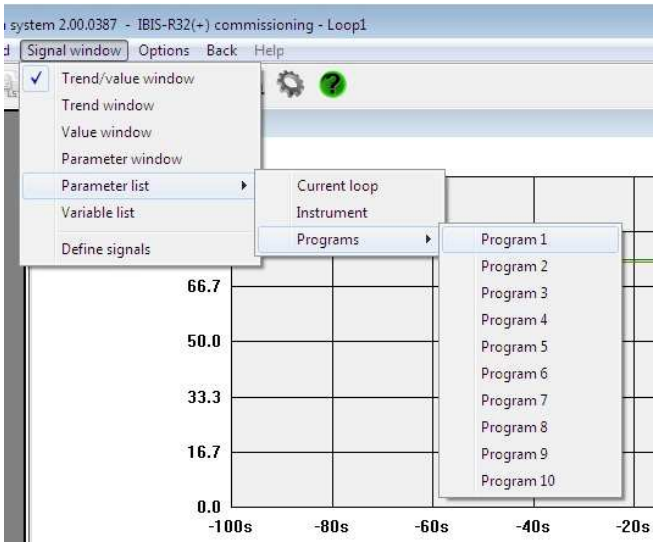
Information required for operating the modem is also provided in the lower part of the dialog. The most important information is the maximum baud rate required for the modem. Since data is transmitted on the telephone line with a double baud rate for addressing the PC or controller, this fact must be taken into account when applying the IBIS-R32 and the controllers. The baud rate applied in the controller configuration for the RS-232 or RS-485 module and the baud rate set in the communications parameters of IBIS-R32 must be identical and less than or equal to half of the maximum permissible baud rate. When using modems with a maximum baud rate of 28.8 kBaud, this amounts to 9600 baud for the controller, since 14.4 kBaud is not available for selection.

The dialog for configuring the modem can be quit with [OK] -all modifications made shall be saved -or with [Cancel] -all modifications made shall be rejected.

If the modem is used for the uploading or downloading of controller information or for commissioning, there will be an automatic query as to if the established telephone connection should be maintained when quitting this program section. Yes means avoiding delays in making new connections. However, keeping the line means incurring extra telephone costs. If the connection is to be initially kept and interrupted later, this can be done in the program section called “Project management” via →Project→Close modemconnection.

If modems are operated on transmission paths, the modem on the controller side should be configured before connections are established -normally before being supplied to the end user. The instructions on how to do this are supplied in the modem manual.

Operation of programmer parameters



During commissioning, the list of parameters of the respective program for the programmer can be called up for parameter inputs, in addition to the parameters of the just selected control loops or of the unit, with →Signalwindow→Parameterlist→Pro-grams. Following this, the usual dialog with which parameter lists can be selected and modified is displayed, the appropriate values being then transmitted to the coupled controller.

Up and downloading

During display of the process sequences for the uploading and downloading of projects, a motioned graphic is windowed.

New predefined variables

Newly predefined variables are introduced with the library 3.5.0. These are variables of the data type INT and variables for the display of alarm pointers in the bar diagrams of Protrenic 550.

Name of variable	Significance
.INT_01 .INT_02 to .INT_32	Variables (INT) for interconnecting customer-specific data are provided in the free configuration. Since the variables can be addressed via the Modbus-RTU and Profibus-DP protocols, these can be employed by systems to gain access to Protronic 500/550 and Digitric 500 data which cannot be numerically processed in REAL.
.L1_GWMAX_GRAPH (.L1_ALHI_GRAPH)	Contains the normed value (REAL) of the upper (maximum) alarm threshold for the bar diagram with number 1 of Protronic 500. This represents the control variable, except in the case of ratio control. Value range: 0.0...1.0.
.L1_GWMIN_GRAPH (.L1_ALLO_GRAPH)	Contains the normed value (REAL) of the lower (minimum) alarm threshold for the bar diagram with number 1 of Protronic 500. This represents the control variable, except in the case of ratio control. Value range: 0.0 ... 1.0.
.L1_VGWMAX (.L1_RAL_HI)	Contains the value (REAL) of the upper (maximum) alarm value pointer in physical units for the ratio in the case of ratio control. When using ratio control and depicting the actual ratio as bar diagram, this value is displayed as an alarm value pointer via .L1_GWMAX_GRAPH.
.L1_VGWMIN (.L1_RAL_LO)	Contains the value (REAL) of the lower (minimum) alarm value pointer in physical units for the ratio in the case of ratio control. When using ratio control and depicting the actual ratio as bar diagram, this value is displayed as an alarm value pointer via .L1_GWMIN_GRAPH.

.L1_XGWMAX (.L1_PVAL_HI)	Contains the value (REAL) of the upper (maximum) alarm value pointer in physical units for the first used maximum alarm value of the control variable.
.L1_XGWMIN (.L1_PVAL_LO)	Contains the value (REAL) of the lower (minimum) alarm value pointer in physical units for the first used minimum alarm value of the control variable.
.L1_YGWMAX (.L1_OUTAL_HI)	Contains the value (REAL) of the upper (maximum) alarm value pointer in physical units for the first used maximum alarm value of the output variable. The alarm value pointer would be windowed for alarm value used in the bar diagram bearing the number 3. Value range: -5.0...105.0.
.L1_YGWMIN (.L1_OUTAL_LO)	Contains the value (REAL) of the lower (minimum) alarm value pointer in physical units for the first used minimum alarm value of the output variable. The alarm value pointer would be windowed for alarm value used in the bar diagram bearing the number 3. Value range: -5.0...105.0.

This is also true for the alarm value pointer of the control loops 2, 3 and 4, as far as variables .L2_GWMAX_GRAPH to .L4_YGWMIN are concerned.

The alarm value pointers for the bar diagram number 1 are faded out when both .Lx_XGWMAX and .Lx_VGWMAX accept the value 100000.0 as the upper alarm value pointer. This is true for the lower alarm value pointer when both variables .Lx_XGWMIN and .Lx_VGWMIN accept the value -100000.0.

Should any of the variables have a value which is not equal to the stated control value, the alarm value pointer mark shall be displayed.

This also applies to the alarm value pointers of the bar diagram number 3 for the control variable. However, everything refers to the variables .Lx_YGWMAX and .Lx_YGWMIN.

Extension of the standard generation of alarm pointers

With the introduction of the alarm value pointers for the control variable, actual ratio and output variable in the library 3.5.0 this means they are integrated into the standard generation routine. Compared to library 3.4.0 this would lead to a change of the FBD programs Lx_GRENZ4 and the introduction of new FBD pro- grams Lx_GW550. In accordance with the list configuration therefore, the variables to be used for Protrenic 550 have been correctly assigned to the alarm value pointers.

If a configuration from a library 3.4.0 is to be exported or accepted earlier, it is recommended to add an interconnection of the new variables. Unused maximum alarm value pointers should then be set to 100000.0 and the minimum alarm values to -100000.0.

Fading out bar diagrams

When applying the firmware for Protrenic 500/550/700, Digitrenic 500/700 or higher, using library 3.5.0, the bar diagrams for the control variable, the set point and the output variable can be faded out (dark) individually.

To enable this, the value of variables .Lx_XANA_SKAL, .Lx_WANA_SKAL and .L1_PID_Y_OUT must be set lower than or equal to -100.0.

Library management

IBIS-R32 uses the library 3.6.0.
In order to configure the controller with older libraries, should a suitable firmware update be upgraded.

Older Configurations can
IBIS-R32 only
"Import" (see Section 2.3 on page 15)
the .CSV files
or
"Submit" (see section 2.4, page 16)
be edited from a regulator.

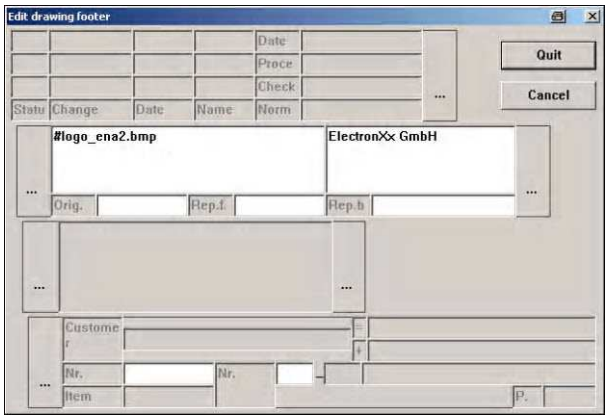
Use of dongles

In this new version the dongle is only required for conducting verification. With it, list configurations for the control of internal wirings leading to the free configuration can be converted.

It is also possible to edit and store free configurations without dongles.

Extension of documentation

Foot of drawing



The documentation module provides the possibility to display bitmaps in the plotted windows (Windows *.BMP).

This can be done by using →Project→Projecthead→Editdrawing footerin the project manager or →Edit→Head→Drawingfooterin the project tree to fill the existing 3 fields with the required information. It is in these fields, as suggested in the figure, that the name of a bitmap can be input. In order to differentiate the text input, prefix the name with #.

Since no absolute directory path can be stated, the bitmaps used are taken from the directory \IBIS-R32\ BITMAPS. This directory is generated automatically upon installing the software. Please store all used bitmaps in the aforementioned directory before generating a documentation. If a required bitmap is not stored in this directory, the bitmap name with preceeding “#” is printed instead.

List configuration

```
Device configuration
Remote control
B04-Q01: 4
Set Module type
B12-Q02: 2              B12-Q04: 50
Communication
B30-P01: 17             B30-Q02: 8

Configuration loop 1
Input signal connection
B03-Q04: 3              B03-Q06: 1              B03-P07: 20.000
B03-P08: 120.00

Configure AI
Analog input 01
B01-Q01: 5              B01-Q02: 14             B01-Q03: 3

Configure AO
Analog output 41
B41-Q01: 2

Configure binary I/O
binary-input/output 04
B04-Q01: 3

Online parameter loop 1
Tag name
P199: WTC027
```

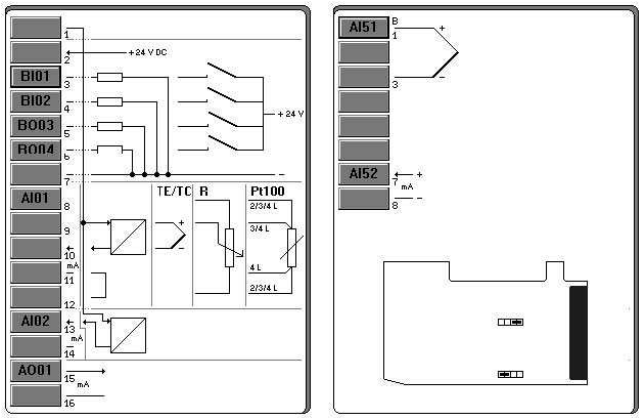
The documentation of the list configuration offers the possibility to print out only deviations or modifications from the factory setting. This enables the compilation of a very abridged and concise documentation, which only contains data on the most important and modified catalogs.

```
 Device configuration

Remote control
B04-Q01: Remote control: 4 -   LOCAL and REMOTE

Set Module type
B12-Q02: Slot 2: 2 -   Interface RS-232/485
B12-Q04: Slot 4: 50 -   Analog output 3 * mA

Communication
B30-P01: Modbus-Address: 17
B30-Q02: Baud rate: 8 -   38400

 Configuration loop 1

Input signal connection
B03-Q04: Eng. unit PV, SP: 3 -   °C
B03-Q06: Display PV,SP: 1 -   with 1 decimal place
B03-P07: Scale low PV,SP: 20.000
B03-P08: Scale high PV,SP: 120.00

 Configure AI

Analog input 01
B01-Q01: Signal type: 5 -   Pt100 3-wire
B01-Q02: Linearization: 14 -   Pt100 -200..200 °C
B01-Q03: Eng. unit: 3 -   °C

 Configure AO

Analog output 41
B41-Q01: Signal type: 2 -   4..20mA

 Configure binary I/O

binary-input/output 04
B04-Q01: Signal type: 3 -   Binary output, normally open (0)

 Online parameter loop 1

Tag name
P199: Tag name: WTC027
```

This type of documentation can also be used for making long documentations, whose question and response texts are also to be printed out.

In order to utilize these possibilities, select “Deviations from default settings only” from the list data of the documentation

Hardware configuration



The documentation of the hardware configuration also permits the printout of interconnected displays stored in the hardware configurator for the modules, instead of a simple display of a unit’s listed configured modules.

In order to utilize this possibility, use the hardware assignment to select “with connencting diagrams” in the documentation module.

Using IBIS-R32 with Windows XP

When using Windows XP to download a configuration, an error message to the tune that parts of the configuration (“Domains”) are not fully loaded could be displayed.
This error can only be remedied by changing the basic settings of the used COM interface in the system control.
By following the steps below, the basic settings can be so modified that the reported error can be prevented.

[Start]
→Settings
→ControlPanel
Double click “System”

→DeviceManager
Swing open Ports (COM and LPT) with [+]
Double click the used COM interface

→Port-Settings
→Advanced

Read the descriptive text in the displayed dialog and set the transmit buffer to “low” (1). Upon completing the dialog, the error normally ceases to occur when downloading.

ElectronXx
Haberstrasse 46
D-42551 Velbert
DEUTSCHLAND

Tel: +49 2051/60721-50
Fax: +49 2051/60721-65
E-Mail: info@electronxx.de

www.electronxx.de

ElectronXx has Sales & Customer Support

Printed in the Fed. Rep. of Germany (03.2013)

ENA Control

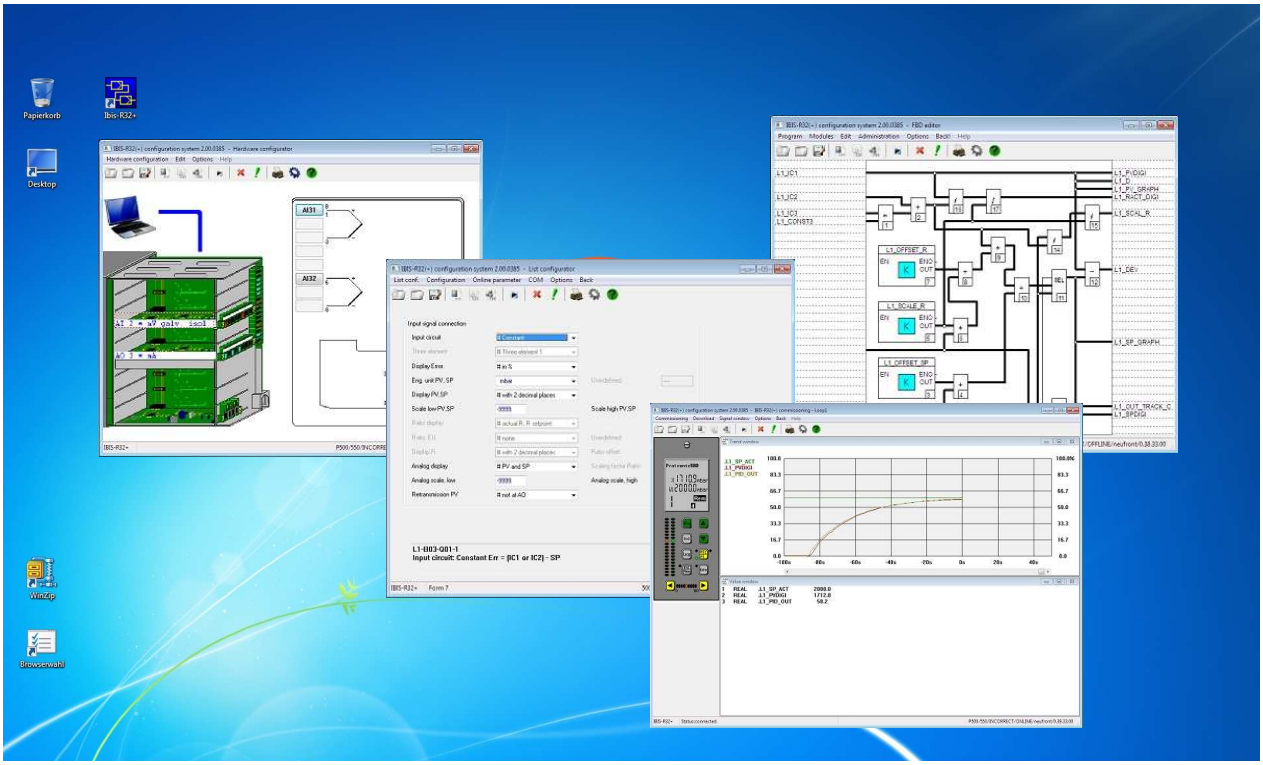
IBIS-R32

Configuration and parameter setting software for
Digitrenic 500/700 and Protrenic 100/500/550/700

since Version 2.00.0360

Supplement to manual
Rev.01

ENA42/62-52030-1 Z2 EN



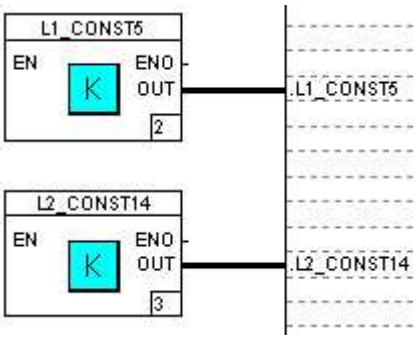
ENAControl

Utilization of the free online parameters K5 to K16

When using the free configuration, the operator is often asked to enter selfdefined variables for online parameters. For such application, the free constants K5 to K16 have been supplemented to each control loop in library 3.6.0 for free configurable units. the Access to the values of this online parameter is gained by using the constants of the functional module

To enable this, the name of the module must contain the number of the control loop (e.g. L1_; L2_ etc.) and subsequently after the word CONST the number of the constants (e.g. L1_CONST5; L2_CONST14).

Serving as an example is the illustration of the access to constant K5 in control loop 1 and to constant K14 in control loop 2.



In addition, new predefined variables should also be generated for each control loop to enable a uniform use of the constants. Their names should give clues to their control loop number and the number of constants. All the same, any other variable of data type REAL can be used. The value of the associated online parameters contains these variables only when the connection. to the constants of the function module and the respective module name was effected, as illustrated on the left side, in free configuration.

Cross-reference to variables

In the edition of variables, possible illustration of cross-references for a variable can now also be generated directly in the FBD/AL editor. To enable this, the variable must be selected: pressing the right mouse key or the function key <F5> or *Edit→variables-cross-reference*:

leads to the display of the associated cross-references:

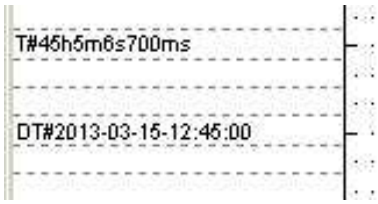
To illustrate the significance of the variable name in the 2nd line see Section on “Foreign-language support”.

This dialog contains a list of all programs, in which the selected variable can be used. The program whose output field contains the variable or which is described with a value carries the identification (:=) after the program name. As usual, every program in question can be called up directly from here.

Notice
There are some default variables which are normally described by the local operation or by the internal program segments of the controller (e.g. .AE01R). Such variables do not carry the (:=)identification.

Input of constants

The time and date constants can be directly entered into the input field of the FBD editor (Data type DINT). It is no longer necessary to convert to seconds/milliseconds:



In contrast to the real-time module, variables with a time format can contain hours exceeding 23. These values represent relative times.

Time display format:

T#..d..h..m..s..ms

(d = days, h = hours, m = minutes, s = seconds, ms = milli-
seconds)

The value corresponds to many times more than milliseconds. Individual components can be left out but the sequence in the order of importance must be kept. Examples for possible entries:

T#3d / T#8h15m / T#18h10s / T#2d9h0m15s750ms

Display format for date:

DT#yyyy-mm-dd-hh:mm:ss

(yyyy = year, mm = month, dd = day, hh = hours, mm = minutes, ss = seconds)

The value corresponds to several times that of seconds. Due to the seconds exactitude, it must be borne in mind during comparison with other statements of dates that equality may be given for a duration of just one second. With the exception of the input for seconds, all components must be stated. It is then that the seconds will be automatically set to 0.
Examples for possible inputs:

DT#1999-08-27-12:45 / DT#2000-03-15-08:00:15

These input formats are also available for input into the dialog “Write variables”.

Calculations in the date format

The difference of two absolute dates can be calculated with the existing SUB function for the data type DINT.

The difference of two absolute clock times can be calculated with the SUB function provided for the data type DINT.

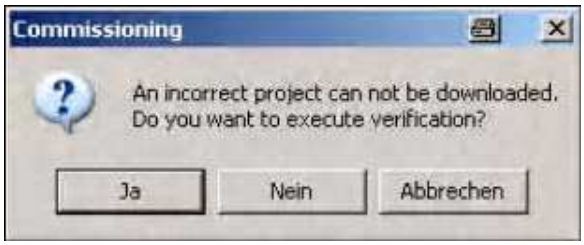
The conversion of a date into a time format, e.g. by using differential calculations, can be effected by multiplying by 1000 for the data type DINT. If necessary, the date value can be checked to see if it is smaller/greater than 2147483 and 0x20C49B respectively. The multiplication produces a result that can be displayed in the DINT format.

In order to convert clock time into a date format, the data type DINT can be divided by 1000. A precheck of the clock time variable is not necessary.

Comparisons such as equalness or the overshooting of date and time inputs can be effected with the provided comparator in the standard group. It is thus possible to obtain binary information for the generation of certain actions at a particular time or on particular days.

Begin of start-up

If start-up is begun with incorrect configuration lists, a dialog shall appear to query if the plausibility check should be called up:



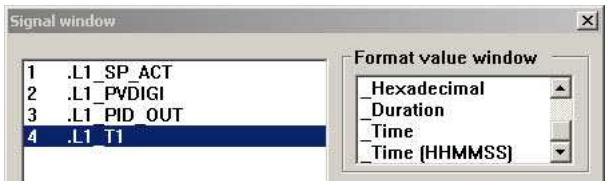
[Yes] calls up the plausibility check in the list configurator

[No] initiated the start-up despite incorrect project, which can then not be downloaded onto the unit

[Abort] aborts the call-up of start-off

Trend and value windows for start-up

The display formats for time in the value window of DINT variables have been expanded to include “duration” and “time of day”:

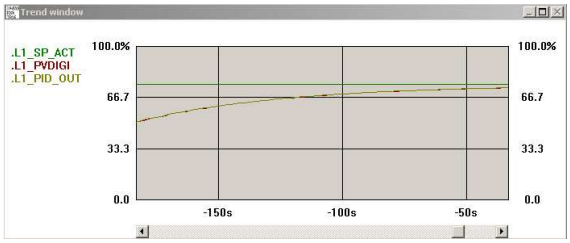


Outputs of these formats appear as follows in the order *duration*, *time (HHMMSS)* and *time of day*:

4	DINT	.L1_D1	T#6h30m56s789ms
	DINT	.L1_D1	T#06h30m56s
	DINT	.L1_D1	DT#1970-09-29-11:46:29

Just as in the case of constant inputs in the input field of the FBD editor, value inputs with the formats *duration* and *time (HHMMSS)* are illustrated by a leading T# and for the *time of day* format by a leading DT#.

The trend window has been expanded to include the display of the physically scaled variables. A click on the name of a variable in the trend window colours the name and duly displays the Physical scaling from 0 to 100% on the left side of the time windows:

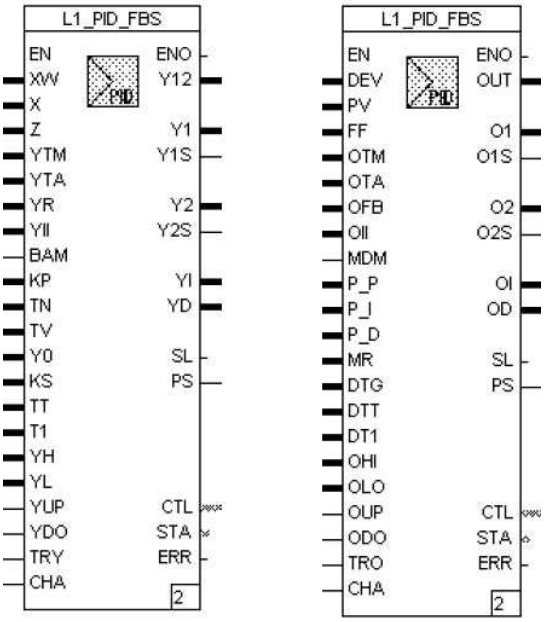


Foreign-language support

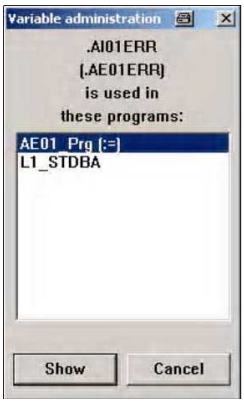
As of library 3.6.0 names of default variables can also be stated in English or French. Predefined variable names are stated in the German language. The type of language is selected with *Options →Language→use of language-defined variables* in the project editor:

Apart from the names of default variables, pin names of the functional modules can also be changed over to the language in question:

hitherto: after changeover:



Upon selecting a cross-reference for variable, the name of the originally defined variable is displayed in the second line, if this name differs from that of the selected variable:

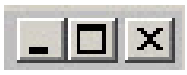


The changeover to a different language is enabled by this foreign language support feature only when the project is closed.

“Minimise” and “Close”

The standard operations “Minimise” and “Close” (<Alt> + <F4>) of Windows are featured.

These actions can be executed either via the standard switching field



or via the menu:

The action “Close” can also be implemented by double klicking the program symbol:

The action “Close” (following a storage query) leads to the termination of IBIS-R+ only in the program segment called “Project Management”. In all other program segments, this program part shall be terminated only upon simultaneously activating another appropriate program segment.

Higher VDU resolutions

In addition to the hitherto existing 640 ×480 pixels for screen resolution, higher resolutions are now also possible. In certain cases, display problems can occur as far as variable names in the FBD editor are concerned, especially where fonts are missing or are not installed. In such cases, it may be necessary to use the following list to modify the font adjustments

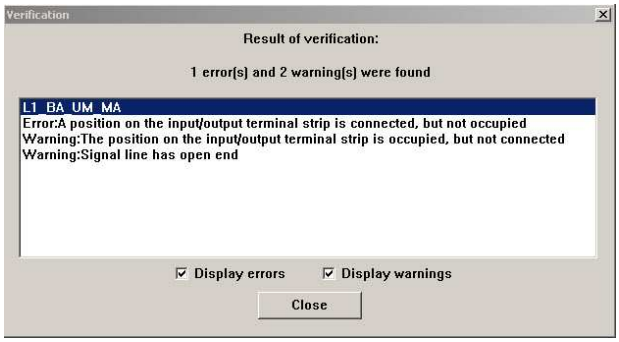
FBSFONT_1600=Courier New
FBSFONT_1280=Small Fonts
FBSFONT_1152=Small Fonts
FBSFONT_1024=Arial
FBSFONT_800=Arial
FBSFONT_640=Small Fonts

in the IBIS_RP.INI file in the Windows directory. Only the character types following the equation signs and existing in the Windows\fonts directory may be used.

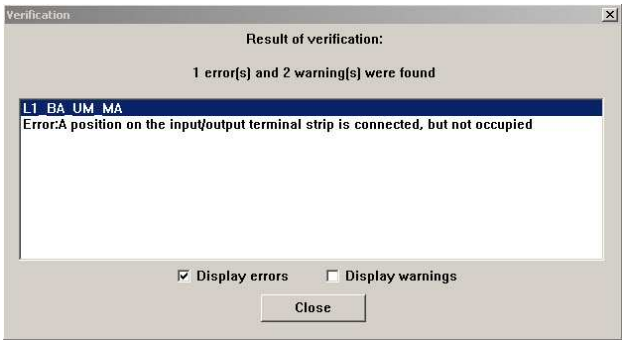
Display of plausibility check information

Plausibility check results can also be displayed in parts, i.e.
as errors only or as warnings. To enable this, the corresponding fields in the output window *display errors* or *display warnings* should be selected. The selected setting is always maintained for the next plausibility check. .

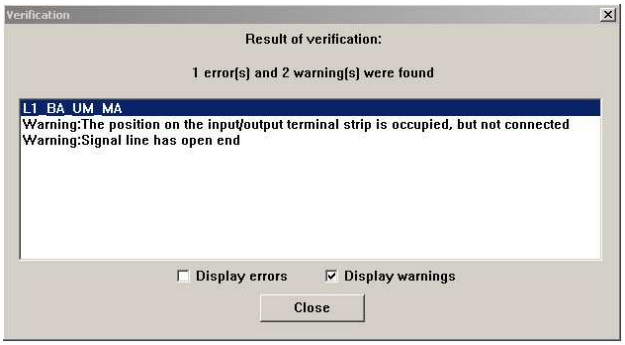
Display of errors and alarms:



Display of errors without alarms:

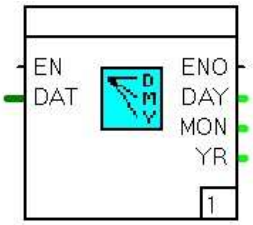


Display of alarms without errors:



D2INT: Date to INT

Icon and module



Library

as of 3.6.0

Function

Breaks down the date in its component parts of day, month, year and makes these available in the outputs.

Inputs

EN	BOOL according to IEC 61131-3
DAT	DINT date (in the desired date format) for breakdown

Outputs

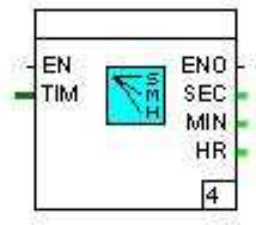
ENO BOOL as according to IEC 61131-3 DAY INT contains the day of the date of input DAT MON INT contains the month of the date of input DAT YR INT contains the year of the date of input DAT

Parameter definitions

none

T2INT: Time to INT

Icon and module



Library

as of 3.6.0

Function

Breaks down the time in its component parts of seconds, minutes, hours and makes these available in the outputs.

Inputs

EN	BOOL according to IEC 61131-3
TIM	DINT time in clock format for breakdown

Outputs

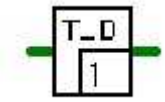
ENO	BOOL according to IEC 61131-3	
SEC	INT	contains the seconds of the clock of the input TIM
MIN	INT	contains the minutes of the clock of the input TIM
HR	INT	contains the hours of the clock of the input TIM

Parameter definitions

none

T_D: Time to Date

Function display



Function

Converts a time element from the clock format (see variable .RTC_ZEIT) into a date in the date format (see variable .RTC_DATUM).

Library

as of 3.6.0

DAY: Date

Function display

Function

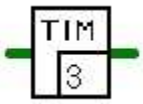
Removes the time component from the date variable at the input. This signal is interpreted as a value in the date format. The result at the output thus remains constant for 24 hours, as long as the date of the real-time clock module remains switched on.

Library

as of 3.6.0

TIM: Time

Function display



Function

Removes the date component from the date variable set at the initial input. The result thus accepts only a value range of between T#00h00m00s and T#23h59m59s.

Library
as of 3.6.0

DOW: Day of the week

Function display

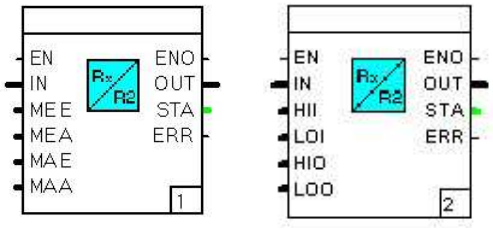
Library
as of 3.6.0

Function
States the weekday of the date variable set at the input as INT.
Codes:

1 Monday
2 Tuesday
3 Wednesday
4 Thursday
5 Friday
6 Saturday
7 Sunday

SKL: Variable range scale

Icon and module



Library
as of 3.6.0

Function
The function module displays an analog signal **IN** of the type REAL in another numerical range and provides this value as a signal at the output **OUT.** For this illustration, a pair of values must be stated for the input and output fields. Should the input value lie outside the measuring range of the input, it must be determined if this value should be kept within the limits or if it should also be effective beyond the limits. The output value would then also be beyond the parameter defined measuring range.

The conversion equation is:

$$Ausgang = \frac{Eingang - MEE}{MEE - MAE} * (MEA - MAA) + MAA$$

MAA Initial value, measuring range

MAE Final value, measuring range output

MEA Initial value, measuring range input

MEE Final value, measuring range

The value of the start of measuring range must be smaller than the end of measuring range at the input and output. Both the value of the measuring range input and measuring range out output can be stated in default as signals or constant parameters.

Inputs

EN	BOOL according to IEC 61131-3
IN	REAL input signal for rescaling
MEE	REAL end of measuring range, input signal
MEA	REAL start of measuring range, input signal
MAE	REAL end of measuring range, of output signal
MAA	REAL start of measuring range, output signal

Outputs

ENO BOOL according to IEC 61131-3

OUT	DINT	rescaled signal
STA	INT	error status
		0 no error
		1 the initial value violated the input measuring range
		2 division by 0.0 occurs
ERR	BOOL	error
		FALSE, if STA = 0
		TRUE, if STA <> 0

Parameter definitions

Measuring range input:

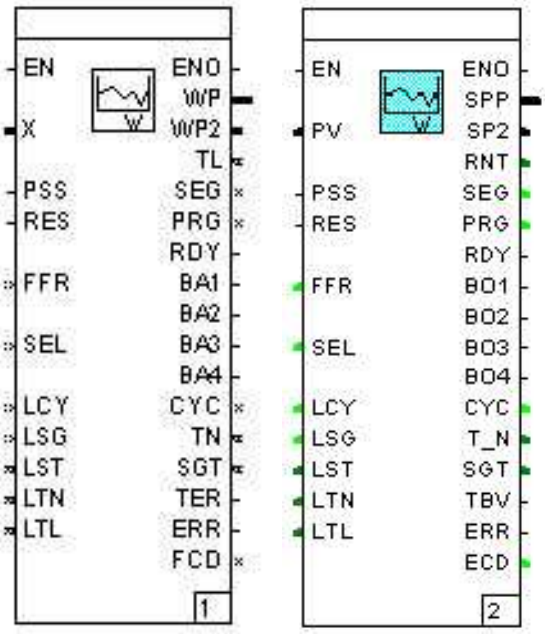
Start of measuring range	lower value of the input signal
End of measuring range	upper value of the ...
Limitation	Limitation of the input signal on the measuring range is utilized.

Measuring range output:

Start of measuring range	lower value of the output signal
End of measuring range	upper value ...

PG2: Programmer 2

Icon and component



Library

as of 3.6.0

Function

This function module provides a programmer for the supply of default set point curves (programs). Up to 10 programs can be predefined. The set point is provided at the output **WP**.

A TRUE signal of the programmer can be initiated via the input **PSS**. The input **SEL** is evaluated at the start. This input predefines the program to be utilized: 1...10. If a fixed program is defined as a parameter, it will not be possible to modify this input.

The set point curve can be reset to original position with a TRUE signal to **RES** by stopping the programmer. On stopping the programmer, it can be switched to quick run or backward run with the input **FFR**. A segment will then complete a cycle in 5 s, irrespective of the segment time defined in the parameter.

For the programmer to continue running from the point of interruption after a power breakdown, its very first run requires: information on the number of loops already completed in **LCY** during the loop cycle runs, the last executed segment in **LSG**, the already expired time in the completed segment in **LST**, the runtime already taken by the entire program without halt/tolerance times in **LTN** and the total runtime of the entire program, including the halt/tolerance times in **LTL**.

Since it is possible to stop set point ramps in accordance with their controlled variable, these ramps can be injected at the input
X. If this function is activated, it is reported at the output with the setting **TER**.

The total runtime of the programmer is stated with the number of the selected program output **PRG** at the **TL** output in milliseconds. The program segment just used is stated at output **SEG**.

On completion of a full program cycle, the program is identified with a TRUE signal at output **RDY**. Each program segment of a set point curve can be predefined as a binary track for up to four binary signals. These binary signals are provided at outputs **BA1** to **BA4**.

For continuation after power failure, the information required for further execution is provided at the outputs **TL**, **SEG**, **CYC**, **TN** and **SGT**. These should be linked to variables which can be given fairlure-free storage.

INPUTS

EN	BOOL	according to IEC 61131-3
X	REAL	controlled variable for tolerance checks
PSS	BOOL	start/stop input. TRUE for Start, FALSE for stop, when RES = FALSE
RES	BOOL	reset input to the start Of a program, is only executed when PSS = FALSE
FFR	INT	quicker forward and backward run, when the programmer is stopped 0 programmer stops 1 quick forward run 2 quick backward run
SEL	INT	Number of the selected program. Count from 0 to 9 for programs 1 to 10.
LCY	INT	loop counter prior to power failure
LSG	INT	edited segment prior to power failure
LST	DINT	runtime in the edited segment prior to power failure
LTN	DINT	total runtime without halt/tolerance times prior to power failure
LTL	DINT	total runtime including halt/tolerance times prior to power failure.

Outputs

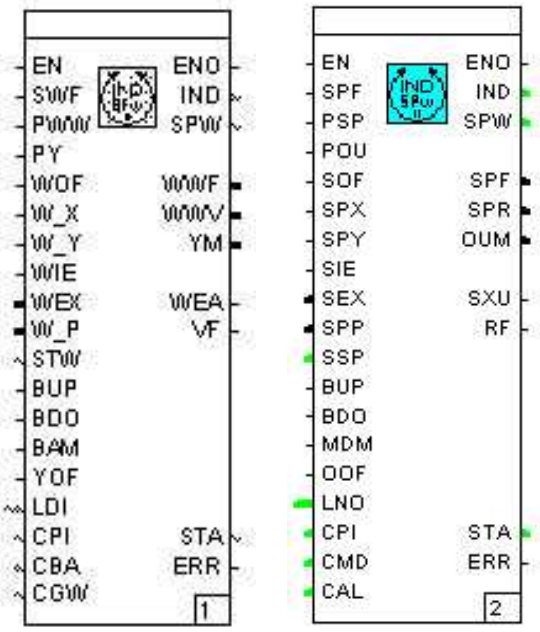
ENO	BOOL according to IEC 61131-3	
WP	REAL	current programmer set point
WP2	REAL	unused
TL	DINT	current total runtime including halt and tolerance times
SEG	INT	currently edited segment
PRG	INT	currently edited program (count from 0 to 9 for programs 1 to 10)
RDY	BOOL	End of program
BA1	BOOL	Binary track 1
BA2	BOOL	Binary track 2
BA3	BOOL	Binary track 3
BA4	BOOL	Binary track 4
CYC	INT	current number of loop editings
TN	DINT	current total runtime without halt/tolerance times
SGT	DINT	current runtime in segment without halt/tolerance times
TER	BOOL	Tolerance function is active
ERR	BOOL	unutilized
FCD	INT	unutilized

Parameter definitions

Program 1 ... Program 10	pressing one of the swithcing fields selects the desired parameter definition mask of the program
Continuation after power failure	upon selection, the programmer starts off from the last point prior to the power failure. Otherwise it starts from very first segment of the selected program.
Selected program	Statement of a set point curve which cannot be entered at the input. Values which can be entered correspond exactly to the possible answers and online parameters of the particular program.

ANZS2: Display loop 2

Icon and module



Library

as of 3.6.0

Function

The function module controls the display and operation of the controlled variable display and the IND display loop. It continues to provide the editor with either the value of the adjusted active set point source or the nominal ratios. Apart from the default elements of the IND display loop, each control loop can display and, if need be, edit 8 free variables of the data type REAL and 2 variables of the data type TIME (display version of the data type DINT).

The next valid set point source is connected via a positive flank at the input **SWF**. The position of the valid set point or controlled variable can be selected via a TRUE signal at the inputs **PWW** and **PY** respectively.

Inadvertent wrongful setting of the set point can be blocked by way of TRUE at the input **WOF**.

The inputs **W_X** and **W_Y** are used for changing over the up to 4 internal set points. If only **W_X** is utilized, the changeover will take place between **W1** and **W2**.

A TRUE signal at input **WIE** switches from the internal to the external set point. In the course of this process, the value connected via input **WEX** is used as an external set point. The use of the active set point is displayed by the function module via TRUE at the output **WEA**.

The input **W_P** is used to connect the default set point of the programmer.

The input **STW** is used for the direct selection of a configured set point source. Upon selecting this source, it is displayed by output **SPW**. If a non-configured set point source is selected, the output **SPW** does not change its value. The values used can be inferred from the output/input list.

The inputs **BUP** and **BDO** are required when making parameter definitions for the remote adjustment of set point or output variable. The speed at a constant pulse is around 100 %/min.

The MANUAL mode of the control loop is manifested to the function module with TRUE at the input **BAM**.

The controlled variable adjustment can be blocked via TRUE at the input **YOF**.

The control loop currently being displayed is connected to the input **LDI** as a figure.

To enable display and execution, this module requires information from the function modules PID universal controller, (PID), mode selector switch (REGBA) and alarms (GW4). Access is gained by connecting the CTL outputs of the modules to the inputs **CPI** (for PID), **CBA** (for REGBA) and **CGW** (for GW4).

The output **IND** shows which entry of the IND display loop shall be displayed on the front panel. The output **SPW** shows the index of the current active set point source (1 = W1, 2 = W2/Vw1, 3 = W3/Vw2, 4 = W4, Vw3, 5 = Wext, 6 = W computer, 7 = W program).

The effective set point is displayed at the output **WWF**, the effective set point ratio is displayed at the output **WWV**.

At the operation status HAND the manual correction value is output at **YM**.

Error states are stated at output **STA**.

In case of an error status unequal to 0, the output **ERR** is set.

Inputs

EN	BOOL	according to IEC 61131-3
SWF	BOOL	switches to the next configured set point source as effective set point
PWW	BOOL	selects the position of the effective set point in the display loop
PY	BOOL	selects the position of the controlled variable in the display loop
WOF	BOOL	blocks inadvertent setting of the set point displacement at TRUE

W_X BOOL switches between W1 and W2 or between W3
and W4 respectively

W_Y BOOL stipulates if W_X changeover between W1 and
W2 or between W3 and W4 shall be executed.
W1/W2 for FALSE, W3/W4 for TRUE

WIE BOOL switches between internal set point W1 to W4
and external set point for TRUE

WEX REAL external set point

W_P REAL Programmer set point

STW INT direct selection of set point source
1 W1
2 W2/Vw1
3 W3/Vw2
4 W4/Vw3
5 Wext
6 W-Computer
7 W-Program

BUP BOOL remote set point adjustment greater
BDO BOOL remote set point adjusment smaller
BAM BOOL mode of operation Manual
YOF BOOL inhibits the output variable adjustment at TRUE
LDI INT displayed loop
CPI INT connection to PID function module
CBA INT connection to the function module mode
CGW INT connection to the alarm value function mode

Outputs

ENO BOOL according to IEC 61131-3
IND INT position of the size of the displayed display loop (see description of variable.INDS_LOOP1)
SPW INT effective set point source
1 W1
2 W2/Vw1
3 W3/Vw2
4 W4/Vw3
5 Wext
6 W-Computer
7 W-Programm
WWF REAL effective set point
WWV REAL effective nominal set point during ratio control
YM BOOL output variable at MANUAL
VF BOOL Display of ratio control at FALSE
STA INT Error status:
0 no error
1 no valid control loop number
2 no new position of the IND-display loop found
3 access to set pointinformation not possible
4 set input circuit is invalid
5 invalid variable at the inputs CPI or CBA or CGW during initialisation
6 invalid variable at the inputs CPI or CBA or CGW during the cyclical execution

ERR BOOL error
FALSE, if STA = 0
TRUE, if STA <> 0

Parameter definitions

Input switching
Settings of the used input circuits.
Loop-No.
No. of the control loop in which this function module operates.
Alarm value adjustment possibilities
Each of the 4 alarm values can be defined to ensure if it can be adjusted
− during display at the operating level (IND display loop)
− only at the operating and parameter-definition levels or
− if it is not displayed at the operating level but only at the parameter-definition level, and is only there adjustable.
Dimension W
4-letter text on the right side of the value, during display of the set points on the front panel. If USER is stated there, the predefinable text for 'USER': will be reproduced. For 'USER' 4-letter text, which, as a user-defined dimension, is shown to the right of the value displayed on the front panel.
Decimal points W
The number of decimal points to be used for displaying the set points. Both fix and floating decimal point displays are optionally selectable.
Display Xw
To display a control deviation, one can select between a display in % and a display in physical units [EU].
Dimension V
Choice between no statement of dimension, dimension % and a user-defined dimension for illustrating set point and actual ratios. To illustrate the user-defined ratios, the predefinable text stored under 'USER': will be reproduced.
Decimal points V
The number of decimal points to be used to display ratios. Both fix and floating decimal point displays are optionally selectable.
Display V
When using the ratio control feature, one can differentiate between display of the set point/actual value ratio or display of set point/actual value in physical units for digital displays.
Release of the remote adjustment blocked:
No remote adjustment can take place.
only Y (in manual):
in the operation mode MANUAL the output variable can be remote adjusted via the inputs **BUP**, **BDO**.
only W (all operation modes): in the operation mode AUTOMATIC the set point variable can be remote adjusted via the inputs **BUP**, **BDO**.
W (in Auto), Y (in Manual):
per remote adjustment, it is via the inputs **BUP**, **BDO** that the output variable is changed to MANUAL. The set point is also remotely adjusted to the AUTOMATIC mode via the same inputs.

set point alarms

W1-Min.	lower set point alarm
W1-Max.	upper set point alarm
V Min.	lower alarm of the nominal ratio
V Max.	upper alarm of nominal ratio

set point changeover with BE

AUS — changeover of the internal set point via **W_X** and **W_Y** is not utilized.

W1-W2
BEx — Changeover via **W_X** occurs only between W1 and W2.

W1-W4
BEy — Changeover via **W_X** and **W_Y** occur only between W1, W2, W3 and W4.

W_X	**W_Y**	set point
FALSE	FALSE	W1
TRUE	FALSE	W2/Vw1
FALSE	TRUE	W3/Vw2
TRUE	TRUE	W4/Vw3

W int/ext with BE

ACTIVE — The input **WIE** is only used for the changeover between the internal and external set point. **WIE** = TRUE switches to the external set point.

OFF — The changeover between internal and external set point is not in function.

W-inhibition with BE

Active — The input **WOF** is used to block the set point adjustor. **WOF** = FALSE means adjustor is enabled.

OFF — The inhibitor of the set point adjustor is not in function.

W-Tracking
manual
OFF — In the MANUAL mode, the efective set point is not traced to the controlled variable.
manual
ON — In the MANUAL mode, the effective set point is traced to the controlled variable.

DDC: set point at computer breakdown

W-current — the adjusted set point is used as effective set point in case of computer breakdown.

W-Comp. — the last computer set point is used as effective set point in case of computer breakdown.

X-current — the current controlled variable is used as effective set point in case of computer breakdown.

set point 1
W1 — Parameter value of the first set point W1.
AUS — set point W1 is not in function.
ON — set point W1 is in function.
follows
active W — set point W1 is in function and if a different set point source is used, this value will be traced to it.

Type W1
no parameters: — The value of set point W1 is not set via the parameter variable but via the local operator only.
Parameter — The set point value W1 is only set via the parameter variable.

set point2/Vw1
W2 — Parameter value of the second set point W2 and ratio set point Vw1 respectively.
OFF — W2 / Vw1 not utilized.
ON — W2 / Vw1 utilized.
Parameter — The value for W2 / Vw1 is set via the parameter value only.

delta
Parameter — The variable is added as delta to W1 for the calculation of the new set point.

Vw1
follows
active
ratio — In the case of ratio control, Vw1 is tracked to the current ratio, if a different source is used for the set point ratio.

set point3/Vw2
W3 — Parameter value of the third point W3 and the ratio set point Vw2 respectively.
OFF — W3 / Vw2 is not utilized.
ON — W3 / Vw2 is utilized.
Parameter — The value for W3 / Vw2 is only set via the parameter value.

delta
Parameter — The value is added to W1 as delta for the calculation of the new set point.

set point4/Vw3
W4 — Parameter value of the fourth set point W4 or of the ratio set point Vw3 respectively.
OFF — W4 / Vw3 is not utilized.
ON — W4 / Vw3 is utilized.
Parameter — The value for W4 / Vw3 is only set via the parameter value.

delta
Parameter — The value is added to W1 as delta for the calcullation of the new set point.

W-Extern
OFF — No external set point is utilized.
ON — An external set point is utilized.

W-Computer
OFF — The set point (via interface) of a higher-level computer is not utilized.
ON — The set point (via interface) of a higher-level computer is utilized.

W programmer
OFF — The programmer is not used as set point source.
ON — The programmer is used as set point

Set points

Name Default text in the display loop for respective set point source.

Diverse

ON/OFF On/Off switching of the respective input inot the IND display loop.

Name 3-line short text, which stands left of the value during display of the value on the front panel.

Controlled variable, Alarm values

Display

ON/OFF On/Off switching of the respective input into the IND display loop.

Name 3-leter short text which stands left of the value during display of the value on the front panel.

Dimension 4-letter text which stands right of the value during display of the value on the front panel. If “USER” has been input there, the predefinable text under “User:” can be displayed.

User 4-letter text which stands right of the value displayed as a user defined dimension during display of the value on the front panel.

K the number of post decimal points required for the displaying the value. Statement of the figure 5 corresponds to a floating decimal point format.

Global variables in the display loop

Variables .Lx_R1 to .Lx_R8 as well as .Lx_T1 and .Lx_T2 can be included in the display loop by tick-off (x represents the number of the control loop).

Name 3-letter short text which stands to the left of the value during display of the value on the front panel.

Dimension 4-letter text which stands to the right of the value during display of the value on the front panel. If “USER” is stated there, the predefinable text for “User:” will be reproduced for display.

“User” 4-letter text which stands to the right of the value during display of the value on the front panel and which is used as a user-defined dimension.

K The number of post decimal places to be used for displaying the value. The figure 5 corresponds to a floating decimal format.

V [ ] Variable value is only displayed but cannot be utilized.

[X] Variable value is displayed and can be utilized.

Global predefined variables

The following variables are new introductions to library 3.6.0:

.L1_SCAL_LO	This value (REAL) displays the contents of the parameter L1-B03-P07 “Lower control loop scaling“ of a List configuration. In the case of a free configuration, this is the parameter of the function module L1_SCALE_LO which is provided as output.
.L1_SCAL_HI	This value (REAL) displays the contents of the parameter L1-B03-P08 “Upper control loop scaling“ of a list configuration. In the case of a free configuration, this is the parameter of the function module L1_SCALE_HI which is provided as output.
.L1_ANA_LO	This value (REAL) displays the contents of the parameter L1-B03-P16 “Lower bargraph scaling“ of a list configuration. In the case of a free configuration, this is the parameter of the function module L1_ANA_LO which is provided as output.
.L1_ANA_HI	This value (REAL) displays the contents of the parameter L1-B03-P17 “Upper bargraph scaling“ of a list configuration. In the case of a free configuration, this is the parameter of the function module L1_ANA_HI which is provided as output.
.L1_SETZ_MAN (.L1_SET_MAN)	The value TRUE in the variable (BOOL) effects, if an interface module is used, a direct mode changeover to MANUAL. The value is automatically reset to FALSE. Changeover and reset are effected only when the configuration accepts this operation mode.
.L1_SETZ_AUTO (.L1_SET_AUTO)	The value TRUE in the variable (BOOL) effects, if an interface module is used, a direct mode changeover to AUTOMATIC. The value is automatically reset to FALSE. Changeover and reset are effected only when the configuration accepts this operation mode.
.L1_SETZ_CASC (.L1_SET_CASC)	The value TRUE in the variable (BOOL) effects, if an interface module is used, a direct mode changeover to CASCADE. The value is automatically reset to FALSE. Changeover and reset are effected only when the configuration accepts this operation mode.
.L1_SETZ_W (.L1_SET_SP)	The value in the variable (INT) activates the required set point source. However, this is only possible when the configuration enables it. Codes/significance: 1 set point 1 2 set point 2 / ratio set point 1 3 set point 3 / ratio set point 2 4 set point 4 / ratio set point 3 5 external set point 6 computer set point 7 programmer set point The value of the variables is automatically set to the value of the real activated set point. This value corresponds to the contents of the variables .WW_LOOP1.
.L1_K5 to .L1_K16 (.L1_CONST5 to .L1_CONST16)	Variables (REAL) are used for the further processing of the online parameters K5 to K16 which can be stated via the front panel during free configuration.
.RTC_DATUM (.RTC_DATE)	If using the real-time clock module, (DINT) contains the current date. Date format: contains the number of seconds since 1.1.1970. Also included, apart from the date of day, the expired seconds of the day. The value thus changes its value every second. Leap years and leap seconds are accounted for in this variable, likewise summer/winter time.
.RTC_ZEIT (.RTC_TIME)	If using the real-time clock module, (DINT) contains the current time. Time format: contains the number of milliseconds of the day since des 0:00 o’clock. The format is compatible with the time format of Protrenic/Digitrenic existing hitherto.

.RTC_ERROR The value in the variables (INT) show which problems can crop up with the real-time clock module. The codes for the respective problems are coded each in a bit of the INT variables, in such manner that several codes are possible at the same time. Codes (decimal, hexadecimal) / significance: 1, 0x1 Replace battery. 2, 0x2 Real time clock module has reset itself due to an interruption in the power supply/battery buffer
(the problem the date to be output as 1.1.1970 and the time as 0:00 o'clock).

- 4, 0x4 Error in the reading of the time of the real-time clock module. Date and time maintained on the module to the next power failure by separate counting with less accuracy. The module is defective and should be checked.
- 8, 0x8 The clock of the real-time module must be reset (the problem causes the date to be output as 1.1.1970 and the time as 0:00 o'clock).
- 16, 0x10 Date and time on the real-time clock module are no longer considered useful. Date and time on the module are kept till the next power failure with an extra counter of less accuracy. The module is defective and should be checked.
- 32, 0x20 The real-time clock module has been restarted. Time and date cannot be read off yet. This problem occurs only directly when switching on the power supply or in case the mains unit is defective.
- 64, 0x40 Date and time of the real-time clock module are being set. Values in the variables .RTC_DATUM and .RTC_ZEIT remain frozen until the Stellvorgang is terminated.

.RTC_STATUS The value in the variable (INT) displays whether the real-time clock module is equipped with a battery and if summer time can be displayed. The codes for the respective information are enclosed in a bit of the INT variable in such way that multiple codes can be stated at the same time. Code (decimal, hexadezimal) / significance: 1, 0x1 real-time clock module is equipped with a battery. 2, 0x2 summer time is displayed in the variables .RTC_DATUM and .RTC_ZEIT.

.SETZ_DATUM (.SET_DATE) The value TRUE in the (BOOL) variable sets the time of the real-time clock module to the value of the variable .NEU_DATUM.

.NEU_DATUM (.NEW_DATE) In the case of the variable TRUE for .SETZ_DATUM, the value of this variable (DINT) is transferred to the clock of the real-time clock module. This time should always be stated as winter time. During summer time, the variable .RTC_DATUM automatically converts this time to summer time.

.MOD0ERR
.MOD1ERR
to
.MOD7ERR The contents of the variables (INT) indicate an error for a value greater than 1 during the execution of the plugin modules 1 to 7. By way of .MOD0ERR this is also displayed for the input/output levels of the basic unit.
Significance:

- 0 no module available.
- 1 module performs without error.
- 2 no communication to module possible. Should this error be displayed at length, a module fault can be assumed. This status is not set for the interface module RS-232/485.

.DPAKTIV (.PROFIBUS_ACTIVE) The contents of the variables (BOOL) display with the value TRUE that the cyclical Profibus DP communication is functioning without problem. The FALSE value represents a fault in the communication. Precondition is the configuration of the list configuration queries G-B30-F05 referring to Timeout and G-B30-F09 referring to the utilization of the timeout for the Profibus also.

.PG_NLAUF (.SPG_NLAUF) Displays (DINT) the total runtime of the programmer, however reduced by halt and tolerance times. This time thus represents the pure runtime, which can be calculated from the parameters of the used program.

.PG_SEGZEIT (.SPG_SEG_TIME) displays (DINT) the time which has run in the segment just executed.

.PG_ZYKLEN (SPG_CYCLES) Displays the completed loop cycles in the programmer during loop executions. Whenever the programmer is not in the loop execution mode, the value is 0.

The control loop-related variables .L1_... also stand as .L2_... etc. for further control loops.

Subject to technical changes.

ENA Control

ElectronXx
Haberstrasse 46
D-42551 Velbert
DEUTSCHLAND

Tel: +49 2051/60721-50
Fax: +49 2051/60721-65
E-Mail: info@electronxx.de

www.electronxx.de

ElectronXx has Sales & Customer Support

Printed in the Fed. Rep. of Germany (03.2013)

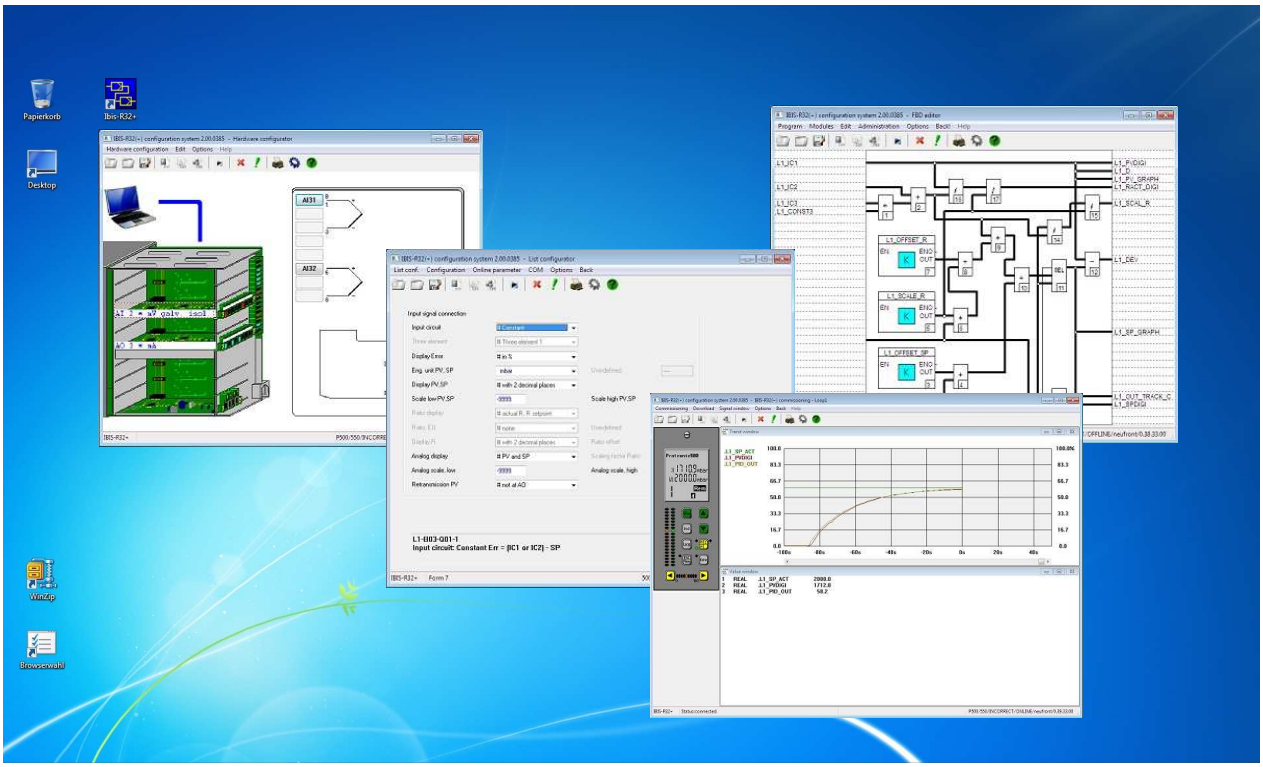
IBIS-R32+

Archiving

Configuration and parameter setting software for
Digitrenic 500/700 and
Protrenic 100/500/550/700

since Version 2.00.0360

Supplement to manual ENA42/62-52030-1 Z3 EN



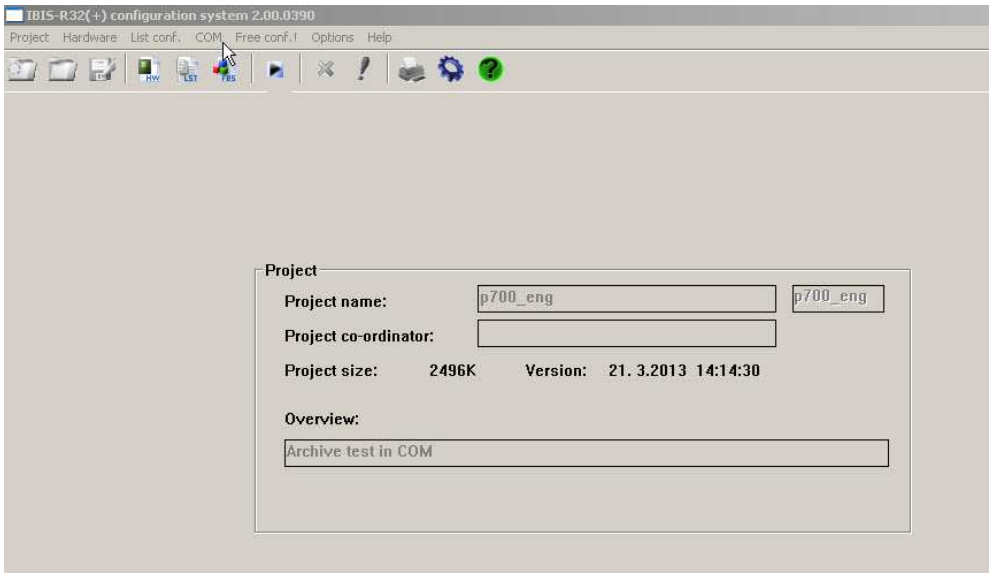
ENA Control

1 Function description Archiving

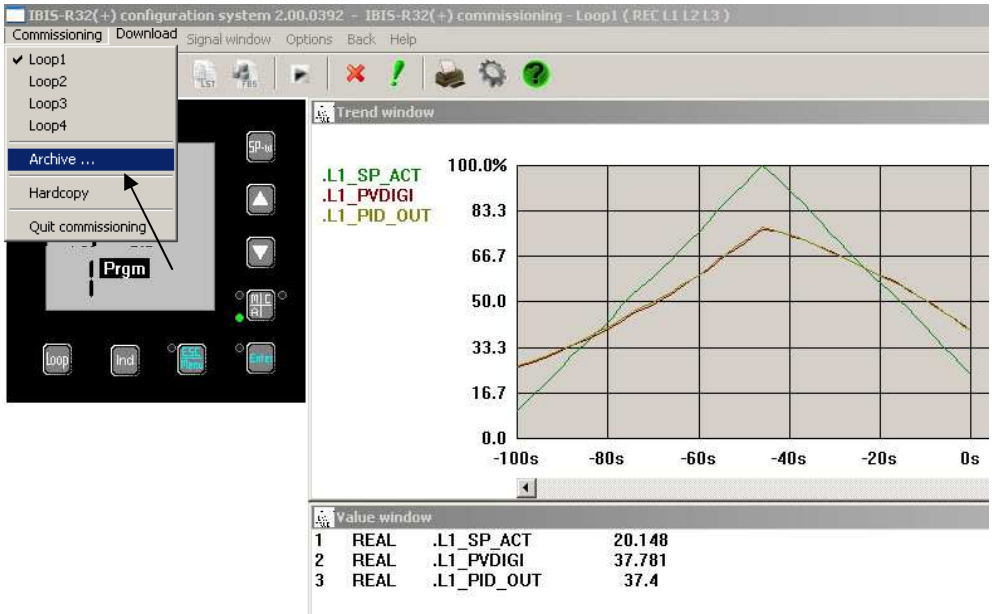
In the Digitrenic/Protrenic configuration software IBIS-R32+ an archiving function will be available from version 2.00.0366 onwards. It will allow you to make a continuous recording of process and controller values of each loop during the commissioning phase (COM). The IBIS-R32+-Log-File (.ilf) can be managed with the file browser. Corresponding spread-sheet programs (e.g. Excel) are used to evaluate respectively process the data…

2 Procedure

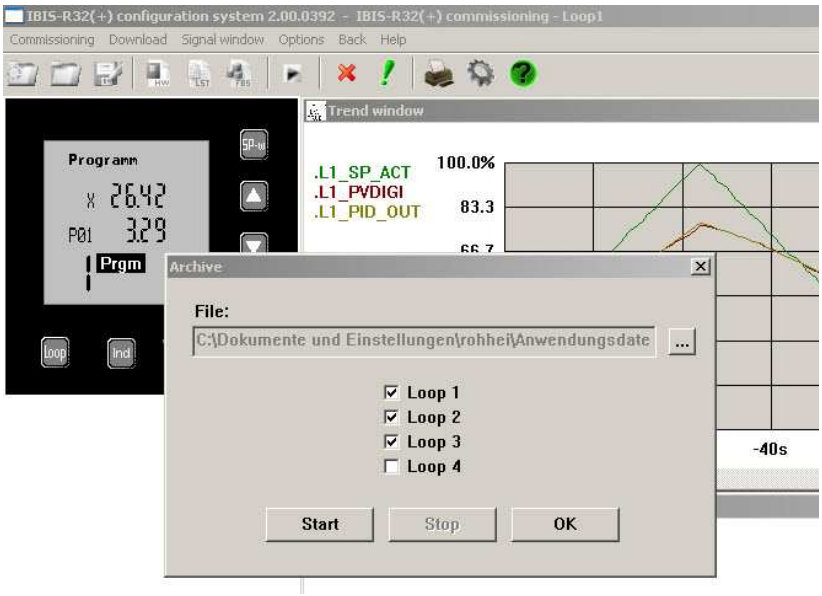
1 With → **COM** you will get into the **Commissioning modus**,



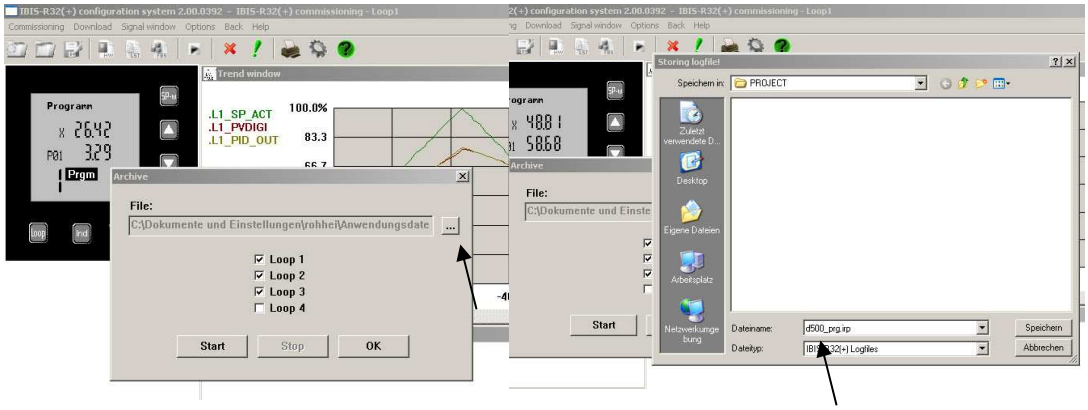
2 Under **Commissioning** you find in the selection → **Archive…**



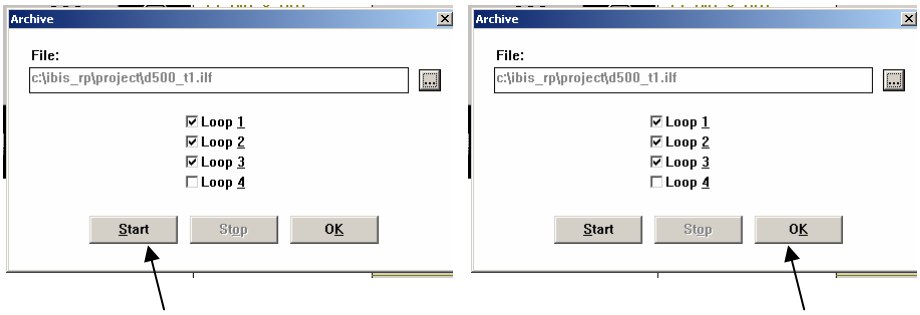
3 Choose those control loops in the “archiving window” you would like to record. All variables (signals) defined before in the “value widow” of the corresponding control loop are now being archived, independent of the momentarily observed loop.



4 Name and locate the Log-File, were the data shall be collected. You can place the Log-File directly into an existing folder. Or you crate a new folder using the file browser where you want the Log-Files located later on. After pressing →  the “Storing-log-file!-window” opens, Here you can name the archiving file and choose for where to store it.



5 To start the archiving with → [Start] and with → [OK] you close the dialog.
To pause or stop the recording → [Stop] and to close the dialog again with → [OK] if necessary.



3 Explanations

Status
See the status bar (caption) for the momentarily recorded loops (in brackets). The term in brackets is no Longer present as soon as the recording is interrupted or stopped.

Data recording
The Recording of data happens in the IBIS-R+-Log-File named by yourself. When the recording is interrupted and restarted again, data will be added to the end of the file, without overwriting the existing data. This is independent of the momentarily connected controller. The same happens, when IBISR+ is restarted and the “old” file name is used!

Create a new data file
A new data file is created by finding a new file name (same procedure as in chapter 2, selection “4” described).

Data structure
Data is being sampled in tables and can be evaluated with the help of spread-sheet programs. The first row determines variable respectively signal names of each controller loop, e.g. .L_WAKT or .L2_XDIGI etc.
The variables are sorted in the same order beginning with loop 1 up to loop 4 as defined in the “value window”

DATE	TIME	.L1_WAKT	.L1_XDIGI	.L1_PID_Y_OUT	.L2_WAKT	.L2_XDIGI	.L2_PID_Y_OUT	.L3_WAKT .L3_XDIGI .L3_PID_Y_OUT

In the first column the recording data can be find, the format is *year month day.* The second column includes the time in *hours minutes seconds.*

DATE	TIME
2012.10.18	12:18:24
2012.10.18	12:18:25
2012.10.18	12:20:40
2012.10.18	12:20:41

With each new controller variable in the “value window” a new line with the variable name is inserted into the table giving you the new attachment. The new vatables are sorted into the corresponding loop and the remaining data is being moved to the right. Please carefully pay attention to this when importing data into a spread-sheet program!

60	408	50,00043	10	1020	0	
.L3_WAKT	.L3_XDIGI	.L3_PID_Y_OUT	.L3_REGLER_AUTO	.L3_REGLER_MAN	.L4_WAKT	.L4_XDIGI
60	408	50,00043	0	1	10	1020

Subject to technical changes.

ENA Control

ElectronXx
Haberstrasse 46
D-42551 Velbert
DEUTSCHLAND

Tel: +49 2051/60721-50
Fax: +49 2051/60721-65
E-Mail: info@electronxx.de

www.electronxx.de

ElectronXx has Sales & Customer Support

Printed in the Fed. Rep. of Germany (07.2013)