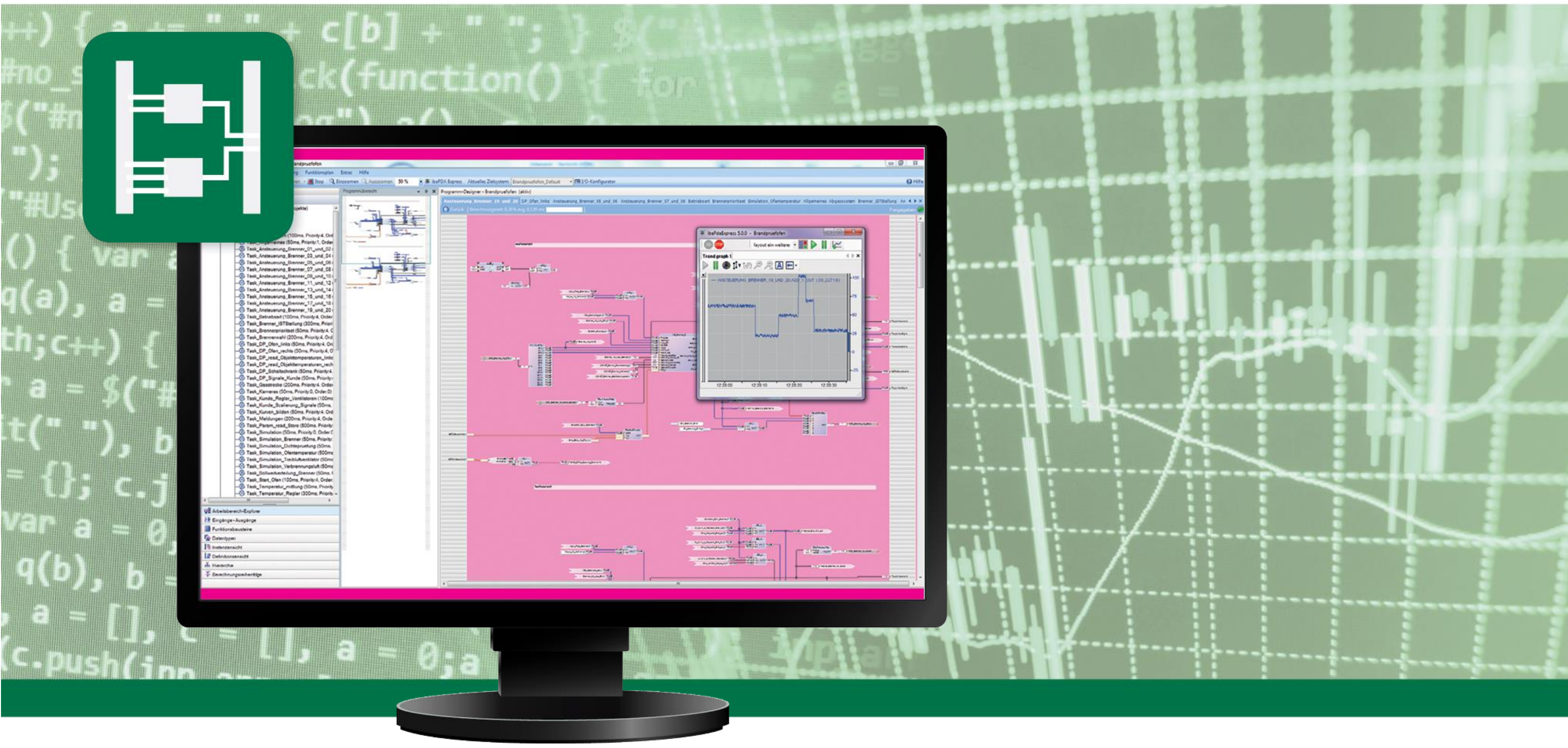
iba

See the Big Picture



ibaLogic-V5

Manual

Issue 5.3

Measurement and
Automation Systems

Manufacturer

iba AG

Koenigswarterstr. 44

90762 Fuerth

Germany

Contacts

Main office	+49 911 97282-0
Fax	+49 911 97282-33
Support	+49 911 97282-14
Engineering	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

The content of this publication has been checked for compliance with the described hardware and software. Nevertheless, deviations cannot be excluded completely so that the full compliance is not guaranteed. However, the information in this publication is updated regularly. Required corrections are contained in the following issues or can be downloaded from the Internet.

The current version is available for download on our web site http://www.iba-ag.com.

Issue	Date	Revision	Author	Version SW
5.3	23.04.2018	Update Software	ST	5.3.0

Table of Contents

1 About this manual

This document describes the function and application of the software *ibaLogic-V5*.

1.1 Target group

This manual addresses in particular the qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing safety and recognizing possible consequences and risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

1.2 Notations

In this manual the following notations are used:

Action	Notation
Menu command	Menu "Logic diagram"
Calling the menu command	"Step 1 – Step 2 – Step 3 – Step x" Example: Select the menu "Logic diagram - Add - New function block".
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
File names, paths	"Filename", "Path" Example: "Test.doc"

1.3 Used symbols

If safety instructions or other notes are used in this manual, they mean:

⚠ DANGER

The non-observance of this safety information may result in an imminent risk of death or severe injury:

- ❑ From an electric shock!
- ❑ Due to the improper handling of software products which are coupled to input and output procedures with control function!

⚠ WARNING

The non-observance of this safety information may result in a potential risk of death or severe injury!

⚠ CAUTION

The non-observance of this safety information may result in a potential risk of injury or material damage!

Note

A note specifies special requirements or actions to be observed.

Important note

Note if some special features must be observed, for example exceptions from the rule.

Tip

Tip or example as a helpful note or insider tip to make the work a little bit easier.

Other documentation

Reference to additional documentation or further reading.

Example

Configuration and application examples for a better understanding

2 Introduction

2.1 Identification

PAC (Soft PLC) and signal manager "ibaLogic-V5".

2.2 Proper Use

The product / system is used for the measurement and control of technical plants and systems.

ibaLogic is not designed for safety-related systems.

Any other or extended use of the product / system is deemed to be improper, and hence, misuse. In this case, the safety and protection of the product / system may get impaired or compromised. iba AG is not liable for any loss or damage resulting from such misuse.

⚠ DANGER

Danger by enabling functions or other services!

Possibility of human injuries and damage to machinery by enabling functions and other services (PMAC, OPC …), which have direct impact on the response of the system.

Secure the system while working on it! Follow the safety regulations applicable!

2.3 Release Notes

2.3.1 Change log file

A change log file (versions_il5.htm) for your software is available on the installation media.
This includes information on the following topics:

- ❑ New functions
- ❑ Improvements
- ❑ Error corrections
- ❑ System requirements

3 Software Installation

3.1 System Requirements

3.1.1 Hardware

The hardware requirements are listed in the following table.

	Minimum requirement	Recommended or higher
CPU speed	1.6 GHz	2 GHz
Number of CPUs	1	2
RAM	1 GByte	2 GByte
Screen resolution	1024 x 768	1280 x 1024

The minimum memory requirement is about 650 MB. An SQL server express database can grow in size up to 4 GB. Keep sufficient free memory space for your requirements. For more information, please refer to "*Performance limits*, page 256".

Note

Installation is possible if the minimum requirements are not met. However, performance limitations may arise.

3.1.2 Software

One of the following operating systems must be pre-installed:

❑ Windows 2008 Server

❑ Windows 7 SP1 32bit/64bit

❑ Windows 8 32bit/64bit, Windows 8.1 32bit/64bit

❑ Windows 2012 Server 64bit

❑ Windows 10 32bit/64bit

Note

Administrator rights are required the installation of the ibaLogic Server and Client.

The software packages listed below are part of the DVD supplied:

❑ Windows Installer 4.5

❑ MDAC 2.81

❑ .Net Framework 4.5.2

❑ MS SQL Server 2014 Express

❑ OPC Core Components 2.0

❑ ibaWDM driver

❑ CB-USB Dongle driver

The installation wizard checks whether the versions of various software packages are available. If any software packages are missing or are an older version, they are installed by the installation wizard or updated.

3.2 Versions

ibalogic-V5 is available in three variants.

Functions	ibaLogic-V5-Lite (free)	ibaLogic-V5	embedded ibaPADU-S-IT-2x16
Unlimited I/O	x	x	x
Cycle time	>= 50 ms	>= 1 ms	>= 1 ms
ibaPDA interface**	x	x	x
iba I/O hardware (FOB) supported	-	x	x
Playback of iba data files	x	x	x
TCP/IP	4 conn.	x	x
Data storage in iba data file format DWF (signals)	-	64*/256*/1024*	256
OPC DA Server or OPC UA Server		unlimited	128
Other interfaces (DLL/RFM/SST/Profibus ...)		Interfaces on request*	

* Upgrade available (fee-based)

** ibaPDA requires separate interface license (if not on the same PC).

The lite version is freely available and - unlike the ibaLogic-V5 full version- does not require a hardware dongle.

The embedded version only runs on an ibaPADU-S-IT-2x16 device and is included in delivery.

Note

The present manual generally refers to the ibaLogic-V5 version. Restrictions of the lite version are set out in the table. In the manual itself, it is not always explicitly referred to.

3.3 License activation

(does not apply to the lite version)

The dongle is already customized at the time of delivery. The customer dongle generates a virtual key in the system that releases various functions.

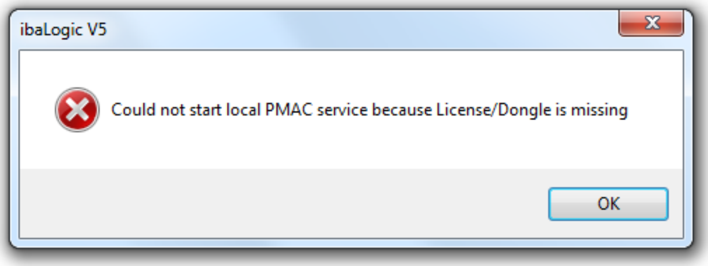
⚠ CAUTION

Danger due to switching off the runtime system PMAC after removing the dongle!

The system cannot be commissioned without the dongle having the associated license.
The license determines the release of functions.
Thus, leave the dongle inserted during the entire operation!

If the dongle is removed during the operation, the PMAC (Programmable Measurement and Automation Controller) switches off following repeated warnings (approx. 5 min after the first warning).

The PMAC does not start without a dongle. If the PMAC is started manually or automatically by the ibaLogic server, a warning message occurs saying that no dongle is inserted.



Could not start local PMAC service because License/Dongle is missing

Instead, an alternative lite version of the PMAC can be enabled in the ibaLogic server options (see "*Settings for the local PMAC*, page 45").
The lite version is restricted (see overview in "*Versions*, page 15")

Figure 1: Dongle

Procedure

➲ Connect the dongle to a USB interface.

After starting the server and the client, the following message appears in the console view: "[RTS] Info: DriverStatus: Driver running for Dongle *Vxxxxxx*".

Note

This message only occurs if a project has been started.

3.4 Installing the software

Follow the instructions of the installation wizard to install the ibaLogic software.

There are two set-up versions

❑ ibaLogic-V5-Lite (freeware)

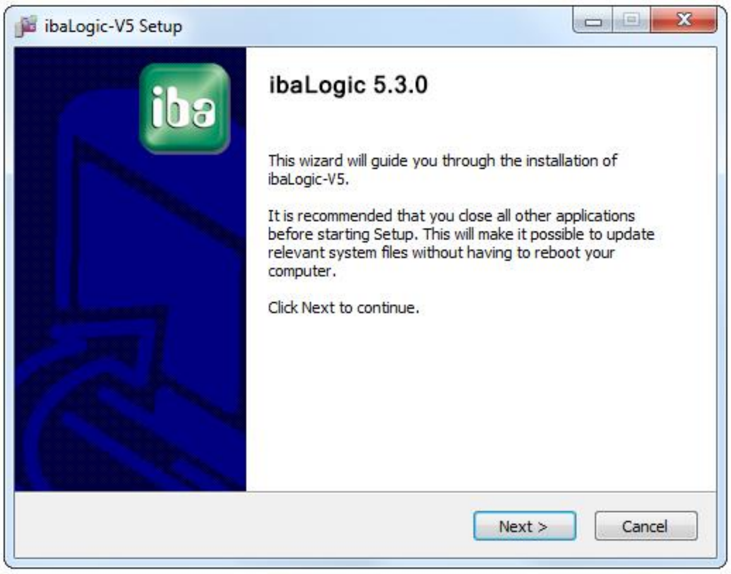
❑ ibaLogic-V5 (extensible, version subject to licensing)

3.4.1 Prerequisite

❑ Your system meets the hardware and software requirements.

3.4.2 Procedure

➲ Double click on the file "ibaLogicSetup-5.3.0.exe".



ibaLogic 5.3.0

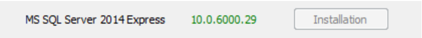
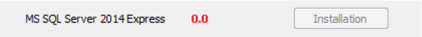
This wizard will guide you through the installation of ibaLogic-V5.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

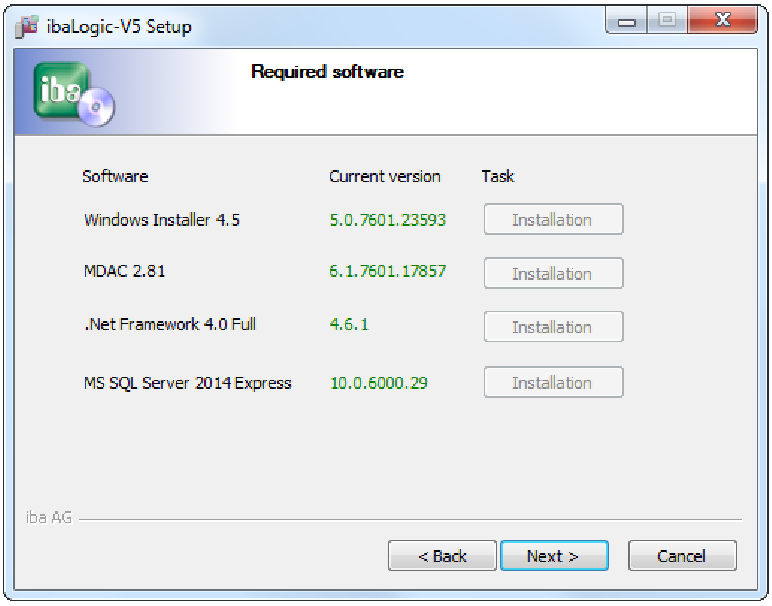
Click Next to continue.

3.4.3 Required software

The components and versions required are specified under "Software".

Display	Explanation
Version number green  MS SQL Server 2014 Express 10.0.6000.29 Installation	Software is installed with adequate functionality.
Version number red  MS SQL Server 2014 Express 0.0 Installation	Software is not installed or inadequate. The associated <Installation> button is enabled.

➲ Install or update the software component(s) by clicking on the enabled <Installation> button.



ibaLogic-V5 Setup

Required software

Software	Current version	Task
Windows Installer 4.5	5.0.7601.23593	Installation
MDAC 2.81	6.1.7601.17857	Installation
.Net Framework 4.0 Full	4.6.1	Installation
MS SQL Server 2014 Express	10.0.6000.29	Installation

iba AG

➲ If required, confirm the dialog window that appears. If all software components have been installed or updated, click on <Next>.

Note

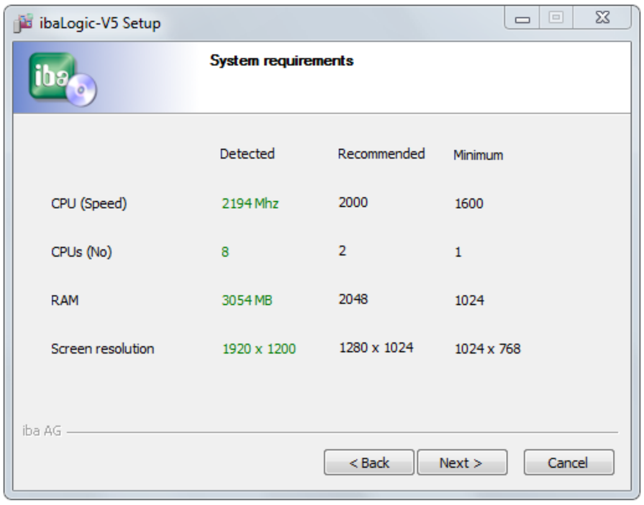
If the SQL server shows a "green" version, with the installation button still being operable, an SQL version was found, however not the 2008 Express Version. The SQL server database found can now be used or an additional SQL version installed.

3.4.4 System requirements

The system requirements are checked prior to installation of the software components.

Display	Explanation
Green	The recommended requirements are met.
Orange	The minimum requirements are met.
Red	The minimum requirements are not met.

➲ If the system requirements meet the minimum requirements, continue the installation.



System requirements

	Detected	Recommended	Minimum
CPU (Speed)	2194 Mhz	2000	1600
CPUs (No)	8	2	1
RAM	3054 MB	2048	1024
Screen resolution	1920 x 1200	1280 x 1024	1024 x 768

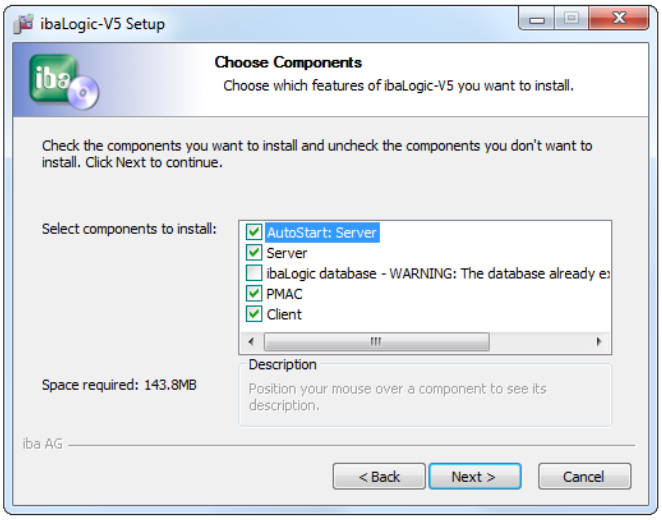
3.4.5 Selecting components

The current computer can be configured using the check boxes.

Server, client and PMAC (runtime) can be installed on different computers. For learning about the system, it is recommended to install all components on a computer first. (Please also refer to "*ibaLogic components*, page 25")

Installation type	Explanation
PMAC:	Runtime environment, can run by itself with a program
ibaLogic database:	Database for ibaLogic projects
Client:	User and configuration interface, needs a started server as connection to the database, to the project
Server:	Central point for the connection of client, database and PMAC
AutoStart:	Server - is needed if the server is installed on this computer and you want it to start directly when starting the computer.

➲ Define the ibaLogic components by selecting the installation type.



Choose Components

Choose which features of ibaLogic-V5 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

☑ AutoStart: Server
☑ Server
☐ ibaLogic database - WARNING: The database already e
☑ PMAC
☑ Client

Space required: 143.8MB

Description

Position your mouse over a component to see its description.

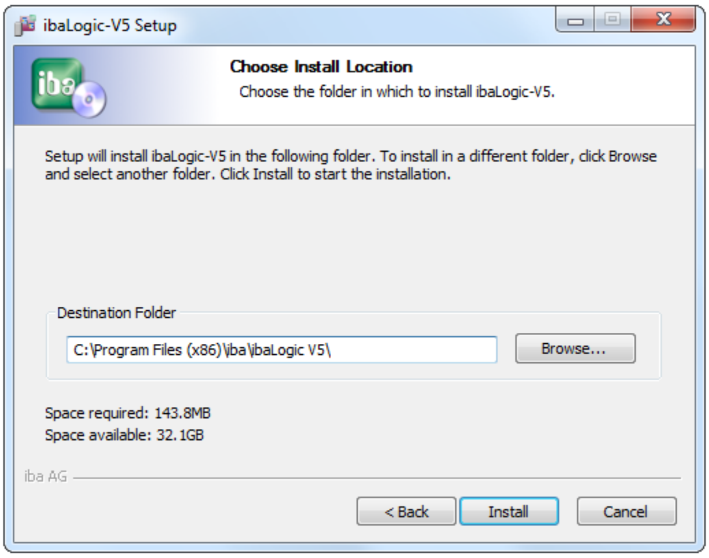
⚠ CAUTION

The "ibaLogic Database" option is not set in case of an update. When the option is set, the existing database including its projects are deleted in the course of installation. "The database already exists and will be overwritten" is displayed as a warning message.

3.4.6 Choose Installation Location

The ibaLogic folder structure (server, client etc.) is created in this folder. iba recommends using the default folder specification.

➲ Define the target folder.



ibaLogic-V5 Setup

Choose Install Location

Choose the folder in which to install ibaLogic-V5.

Setup will install ibaLogic-V5 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files (x86)\iba\ibaLogic V5\

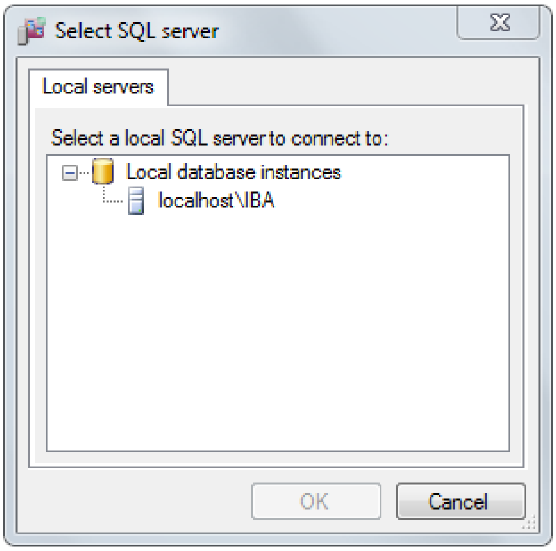
Space required: 143.8MB
Space available: 32.1GB

iba AG

3.4.7 Select SQL server

If the "ibaLogic Database" check box has been enabled when installing, the installer searches for Microsoft SQL servers on the local computer during the installation process and offers you the "Select SQL server" dialog for selecting the database instance.

➲ Select the "localhost\IBA" default instance or an instance of your choice and quit the dialog window using the <OK> button.
The ibaLogic database selected is installed on the server selected.



Select SQL server

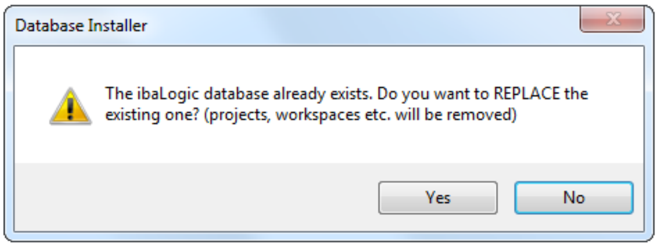
Local servers

Select a local SQL server to connect to:

Local database instances
localhost\IBA

Note

Any existing ibaLogic database can be overwritten after confirmation in the dialog window that appears. If you exit this dialog window using the <No> button, the existing database remains unchanged.

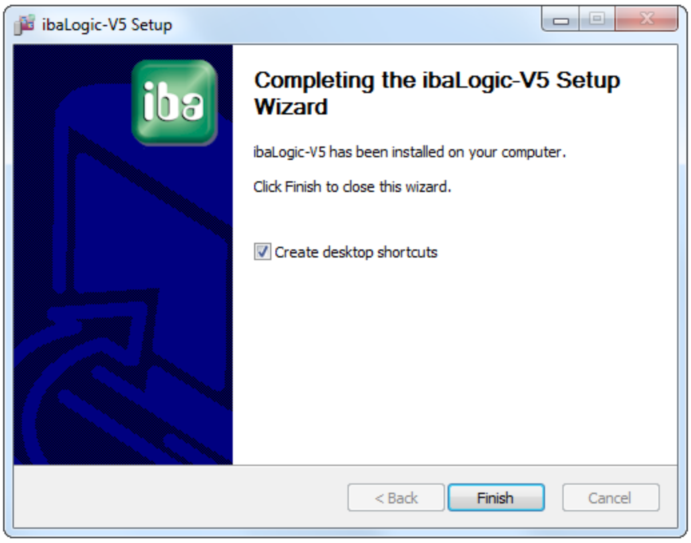


The ibaLogic database already exists. Do you want to REPLACE the existing one? (projects, workspaces etc. will be removed)

3.4.8 Complete ibaLogic installation

Complete ibaLogic installation

➲ Click on the <Finish> button to complete the installation.



Completing the ibaLogic-V5 Setup Wizard

ibaLogic-V5 has been installed on your computer.

Click Finish to close this wizard.

☑ Create desktop shortcuts

4 ibaLogic Software

4.1 Introduction

iba AG has already specialized in the field of measured value acquisition in heavy industry plants for many years. The segment focus has been on plants for the production and processing of steels and non-ferrous metals.

The programs for the *acquisition*[1] and *analysis*[2] of the data recorded are in use all across the globe today, and are deployed by all large suppliers of machinery and automation technology worldwide.

As a result of the wide-ranging options for the connection of the iba measurement technology to the most diversified automation technologies and generations, particularly even to the most prevalent field and drive buses, the need to connect these highly diverse worlds, developed rather quickly. From now on, "the unidirectional flow" of measurement had to become "bidirectional flow" for information exchange between various automation systems – this is typical for the upcoming market of modernization or revamping of already automated systems.

In order to address this requirement, iba AG has already developed a freely programmable signal manager since 1995. The standard, IEC 61131-3, which had already been formulated during this period, for describing technical work flows with the help of graphical elements and easily embedded programming techniques, simplifies the descriptions of complex signal processes considerably.

The graphical mode of *programming*[3] that has been derived from this standard, forms the basis of almost all automation systems today. Hence, graphical programming is compatible and portable to a large extent.

Features:

❑ Onlinechange

❑ Permanent project backup

❑ On-the-fly input check

❑ Visualization and trace tool (ibaPDAExpress)

❑ Interval and event tasks

[1] ibaPDA

[2] ibaAnalyzer

[3] FBD is a graphical programming language, in which function units are interconnected with one another instead of a sequence of textual commands, as in the case of classical programming languages. Circuit diagrams of hardware development can be considered a model in this case. This representation of a program meets the requirements of developers of controller software, whose technical background is typically one of electrical engineering. The various function units are themselves often created using other PLC languages, such as, for example, "structured text", and can be supplied as standard function blocks by the manufacturer of the automation systems or written by the user himself or they can even be imported.

4.2 Areas of application

ibaLogic is used for the following applications:

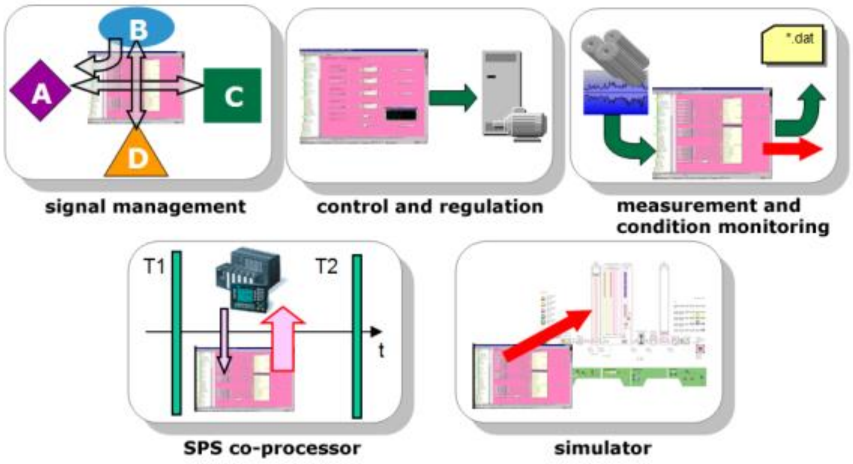


Figure 2: Areas of application

Signal management

You can establish links between the most diverse generations of automation systems from the largest variety of manufacturers with the help of iba connectivity.

Bi-directional data exchange enables communication with controllers that are otherwise incompatible.

PLC-Co Processor

In this area of application, the ibaLogic plays the role of a co-processor.

The vertical green lines in the figure "Areas of application" represent the sampling time of the original automation equipment. Data are transmitted after one PLC clock pulse (T1) to the Soft PLC (ibaLogic). Complex calculations can be performed in real time and the results transmitted back before T2 using the PC processing power and the PC data formats available. Modernizations too can be implemented with the help of such methods: Open-loop and closed-loop control and regulations functions of the "old" PLC are taken over by the new ibaLogic automation system step by step.

Observation of measurements and condition (Condition Monitoring)

Apart from the use as a signal manager, it was also desired to involve ibaLogic for complex measurement tasks, which would not have been possible using a standard ibaPDA. Integrating a function unit for measured value acquisition is one of the core functions of ibaLogic. You can achieve event-driven management of measurement tasks and save the values on various media using this "DAT_FILE_WRITE". You can then process and analyze the data using various iba tools, e. g. ibaAnalyzer, ibaDatCoordinator, etc.

Note

DAT_FILE_WRITE is subject to licensing.

Automation (Open-loop and closed-loop control)

The graphical programming language described in the IEC 61131-3 standard forms the basis of ibaLogic. This language has been conceived particularly for programmable logic controllers (PLC). The developments in recent years have shown that the market for measurement and control systems is growing closer together. The logical consequence is that ibaLogic can obviously also be used for automation tasks as a full-fledged PLC.

If, in the process, the tasks of the Operating and Runtime system are handled by a PC, we speak of a PC-aided Soft PLC or PAC, in short (Programmable automation controller).

ibaLogic permits disconnection of the Runtime system of the PC into a secondary stand-alone intelligent system, ibaPADU-S-IT-2x16. In such a case you can shut down the PC without stopping the Runtime system. The PC is used as a development station in such cases, as in all other PLC systems.

Simulator

This application is an active simulator programmed using the language tools provided by ibaLogic. HMI visualization presents the operator interface and the result of the simulation. The standard OPC interface is used to establish the connections between ibaLogic simulation and the HMI.

4.3 ibaLogic components

ibaLogic is based on the server-client model. This architecture facilitates decentralized operation.

ibaLogic is composed of the following components:

❑ Runtime system (PMAC)

❑ ibaLogic server

❑ ibaLogic client

❑ Database

❑ OPC server (OPC DA or OPC UA)

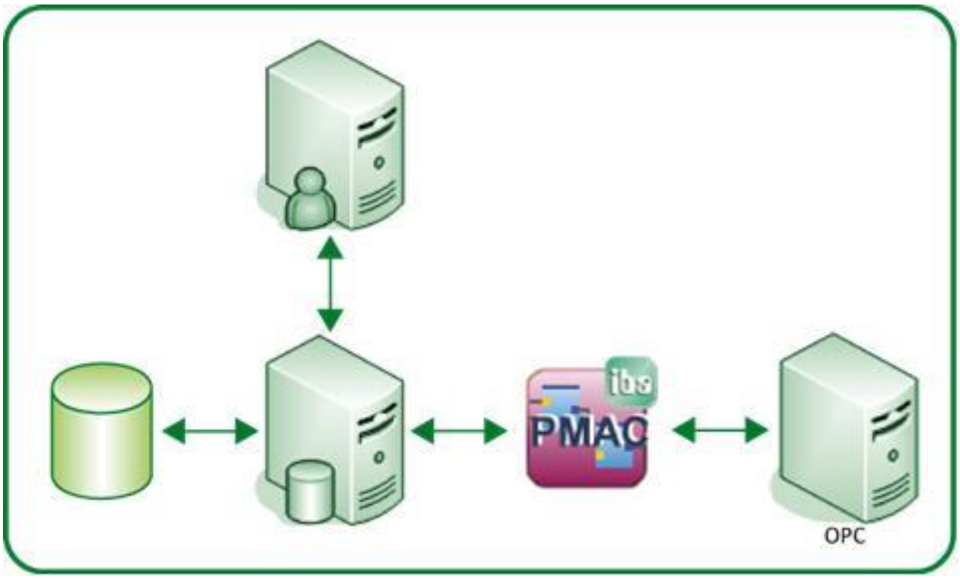
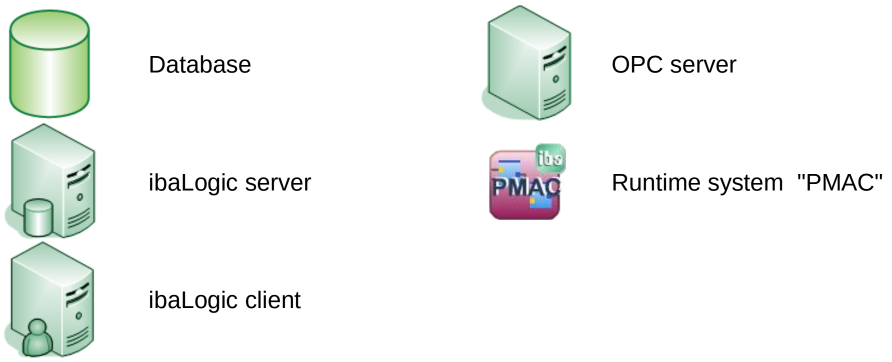
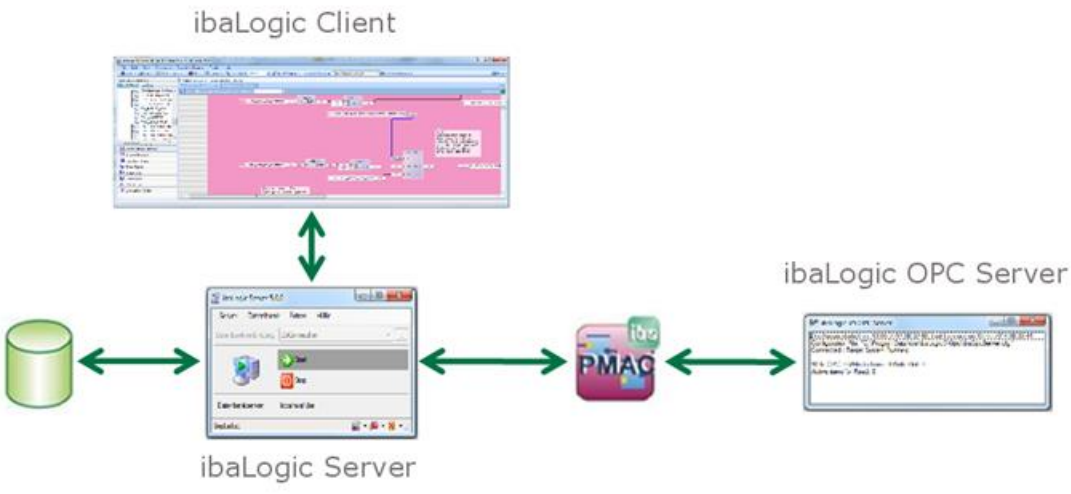


Figure 3: ibaLogic components



In the simplest case, the components mentioned above are located in one computer. However, you can also run these components on separate computers.

Each of these components has its own user and visualization interface.



The PMAC user interface opens by right-clicking on the tray icon in the Windows task bar and opening STATUS.

Note

The figure above shows the program interface of OPC DA. The differences between OPC DA and OPC UA are described in the "OPC" chapter.

4.3.1 Runtime system (PMAC)

By starting an ibaLogic project in the ibaLogic client, it gets compiled and loaded in the "Programmable Measurement and Automation Controller" (PMAC). This runtime system can be located on a Windows computer (as Windows service) or on a compatible WEC (Windows Embedded Compact) device (ibaPADU-S-IT-2x16).

The runtime system continuously returns values that are calculated currently to the client. These are displayed on-line in the graphical user interface of the project.

If a project has been transferred to the Runtime system and started, it is capable of running independently without the server / client running.

4.3.2 ibaLogic Server

ibaLogic is a database-based system. The ibaLogic Server takes over the management of the database and the communication between the ibaLogic client and the Runtime system as the central manager.

All ibaLogic projects are managed by the ibaLogic Server in a database.

ibaLogic uses the license-free Microsoft SQL Express database. During installation, a Microsoft SQL Express server with the associated database is installed if it is not already present.

The ibaLogic Server dialog is also used to backup and restore ibaLogc applications. The backup of the database is saved in an external file.

If there are any modifications made to ibaLogic projects, these are automatically saved in the database. Specific saving during the customization of a project is omitted.

4.3.3 ibaLogic Client

The ibaLogic Client is the programming environment in which ibaLogic projects are programmed and configured. An ibaLogic Server must be connected for this purpose. This ibaLogic Server can be present on the same computer or in the network.

Moreover, the ibaLogic Client controls loading, starting and stopping an ibaLogic project in the Runtime system with the help of the ibaLogic Server.

4.3.4 OPC server

The OPC Server provides all variables, which have been declared as OPC visible, to the OPC Clients connected. In general, OPC Clients are HMI (Human Machine Interface) systems. By default, the OPC Server runs on the same machine as the ibaLogic Server, but it can also be explicitly started on other computers in the network, and is then connected directly with the PMAC via TCP/IP independently. ibaLogic supports OPC DA or OPC UA Server.

4.4 Server-client Operation and other System Configurations

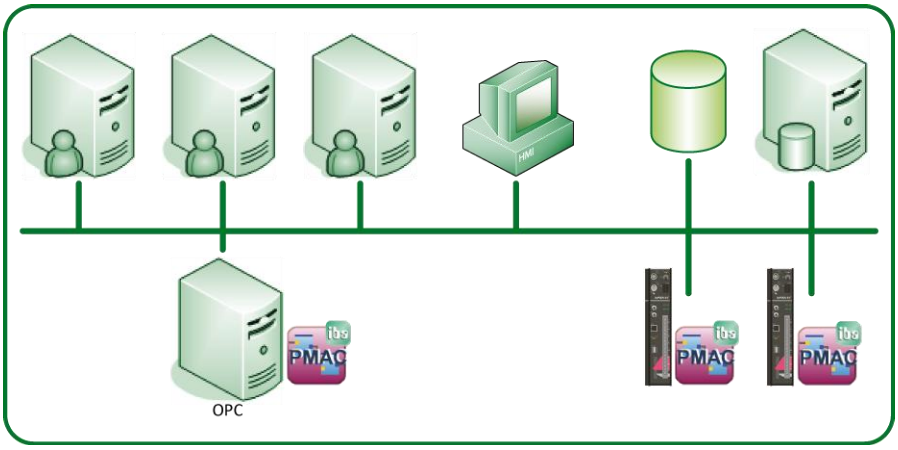
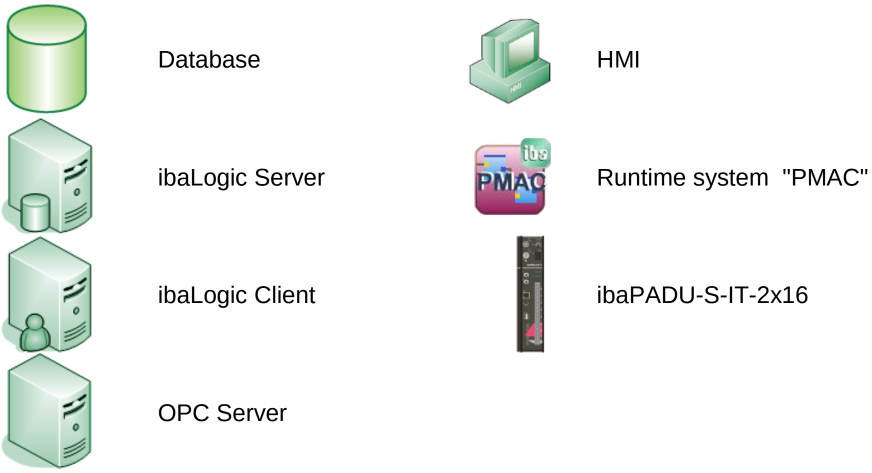


Figure 4: Possible system configuration



In the simplest case, ibaLogic can run with all its components on a single Windows computer.

Alternatively, the ibaLogic components can also be distributed and run on different computers. There are 3 ibaLogic applications in the sample configuration illustrated above. One runs on a PC and 2 others on a separate ibaPADU-S-IT-2x16 system. The central server has a connection to one database in which all 3 projects are saved. Only one ibaLogic workspace is always loaded on this server from the database, which can contain three projects corresponding to the three PMACs.

It is possible to control and monitor the workspaces from different computers in multi-server-client mode of operation.

Important Note

Only one client at a time should carry out modifications regarding the projects of a server.

However, only one project / application as selected is always "active" in one workspace. You can only work with and monitor this active PMAC **online**.

The HMI system can be supplied with information by the OPC server. No OPC Server can run on ibaPADU-S-IT-2x16. As a result, HMI data from and to ibaPADU-S-IT-2x16 central units must run via this Windows computer.

4.5 Operating modes

ibaLogic provides two operating modes in order to meet the various requirements of different applications.

You can choose the following modes of operation in ibaLogic:

- ❑ Measurement = buffered mode
- ❑ Soft PLC

Set operating mode

Procedure

1. Select "Configuration – I/O configurator" in the menu. The "I/O configurator" window is displayed.
2. You will find the operating mode options on the tab "Hardware configuration – General settings".
3. Activate the operating mode to be used under "General settings".
4. Finally, click on <Apply>.
5. If you wish to close the I/O configurator, click on <OK>.

Note

For more information, please refer to "*General settings*, page 181".

For further explanations, please refer to "*Time response*, page 248" and “*Buffered Mode*, page 193".

4.6 Structure of an ibaLogic application

An ibaLogic project application consists of the following elements:

Workspace

- ❑ Project 1
  - Task 1 / Program 1
  - Task 2 / Program 2
  - Task n / Program n
- ❑ Project 2
  - Task 1 / Program 1
  - Task 2 / Program 2
  - Task n / Program n
- ❑ Project n

You can assign the programs with their properties (task interval etc.) to one project each. The projects, in turn, are organized in a workspace.

One project is assigned to one Runtime system / PMAC. Only one project is set as active within one workspace, i. e. only this active project can be started or stopped.

Note

Only one project can be active within a given application.

4.6.1 Task/program properties

According to IEC 61131-3, several programs can be assigned to one task. ibaLogic supports the fixed assignment of one program to one task.

ibaLogic has two task types and associated parameters:

- ❑ Interval task
  - Priority
  - Interval time
  - Order
- ❑ Event task
  - Priority
  - Triggering event

Tasks can interrupt each other, so there is priority. This applies primarily to both task types. Priority 0 is the highest priority.

The interval time determines the time slot in which the interval task is restarted again and again. The minimum time slot for the interval time is 1ms.

The order determines the processing with the same priority. The order number can be assigned only once.

Note

To receive the same behavior as in ibaLogic V4 where interval tasks could not interrupt each other, they will all have the same priority with ascending order number.

The event task requires a triggering signal. Currently, only binary input signals and analog integer input signals are allowed. The triggering event is the rising edge in case of binary signals, a change in value in case of integer input signals.

A higher priority event task always interrupts lower priority event or interval tasks.

Note

The program properties "Interval time" and "Priority" are of considerable significance for the project performance. For more detailed information on these program properties, please refer to "*Time response*, page 248" and "*Performance limits*, page 256".

4.6.2 Program elements

An ibaLogic program may consist of the following elements:

❑ Function units of the integrated standard library

❑ Function blocks created by the user with structured text

❑ Macro blocks created by the user

❑ DLL-based function blocks created by the user (additional license required)

❑ Linking elements

❑ Hardware input and output signals

❑ Comments

4.6.2.1 Function units

ibaLogic has a library of function units. This library contains standard function units in accordance with IEC 61131-3 as well as supplementary functions and function blocks.

You can combine these into a macro block to have a clearer program structure and provide encapsulation of various graphics subprograms.

You also have the option to create a separate function block yourself that is required for a specialized solution to a problem.

For this purpose, ibaLogic provides the feature of creating a new function block by means of structured text. The ST (Structured Text) code is visible to the user and can be modified.

One variant of the self-created function block is creating your own DLLs (using a DLL framework provided by iba). The code is hidden with this option. The function block created in this manner is available as a standard function unit (additional license required).

Further documentation

Further information on creating DLLs on request.

4.6.2.2 Graphics Programming

The following elements are available to link the function units:

- ❑ Connection lines
- ❑ Intra-Page Connectors (IPC)
- ❑ Off-Task Connectors (OTC)
- ❑ Converter
- ❑ Splitter
- ❑ Joiner

You need to connect the function units with the help of connection lines for graphics programming. You can use intra-page connectors for better program structuring.

An intra-page connector merely represents a drawn simplification. In the process, the IPC replaces a connecting line. This is beneficial particularly when several objects on one page need to be connected with the same point or "long" connections are required across multiple pages.

Off-task connectors serve as program-independent connecting elements. These are required whenever you need communication between several programs.

Off-task connectors are also used for communication between ibaLogic and OPC Clients. This can be configured in the off-task connector.

Tip

For further information, also refer to "*Graphical Connections*, page 148" and "*Converters, splitters, joiners*, page 157".

4.6.2.3 Comments

You can use comments to structure and simplify the program description. These can be placed wherever desired or even "docked" to another element.

4.6.2.4 Available data types

ibaLogic supports all elementary and combined data types defined in the IEC 61131-3 standard (Exception: WSTRING).

4.6.2.5 Integrated measurement using ibaPDA Express

You can use the integrated ibaPDA Express tool for quick display of a signal waveform.

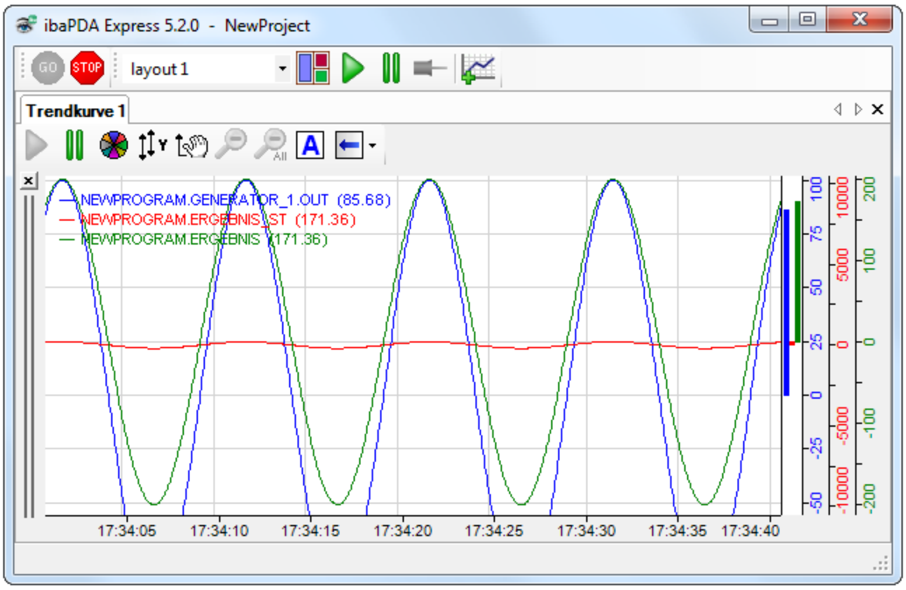


Figure 5: Integrated measurement using ibaPDA Express

With the ALT key pressed, you can move the signals to the ibaPDA Express window using Drag & Drop and display them there.

ibaPDA Express does not save any data for long-term recording.

4.6.2.6 Measured value storage

You can store measured values using the licensed (rights-managed) function block, "DAT_FILE_WRITE". For more information, please refer to "*DAT_FILE_WRITE (DFW Function Block)*, page 95".

Tip

The measurement signals in the generated measurement files (*.dat) can be displayed and evaluated with the convenient ibaAnalyzer analysis software.

4.7 Connectivity

The ibaLogic systems are capable of communicating with one another via the interfaces available in the Windows PC or ibaPADU-S-IT-2x16 (e. g. via TCP/IP, ibaNet etc.).

The communication with external systems or discrete I/Os is illustrated in the connectivity overview. (If required, inquire current connectivity at "*iba support*, Page 358".)

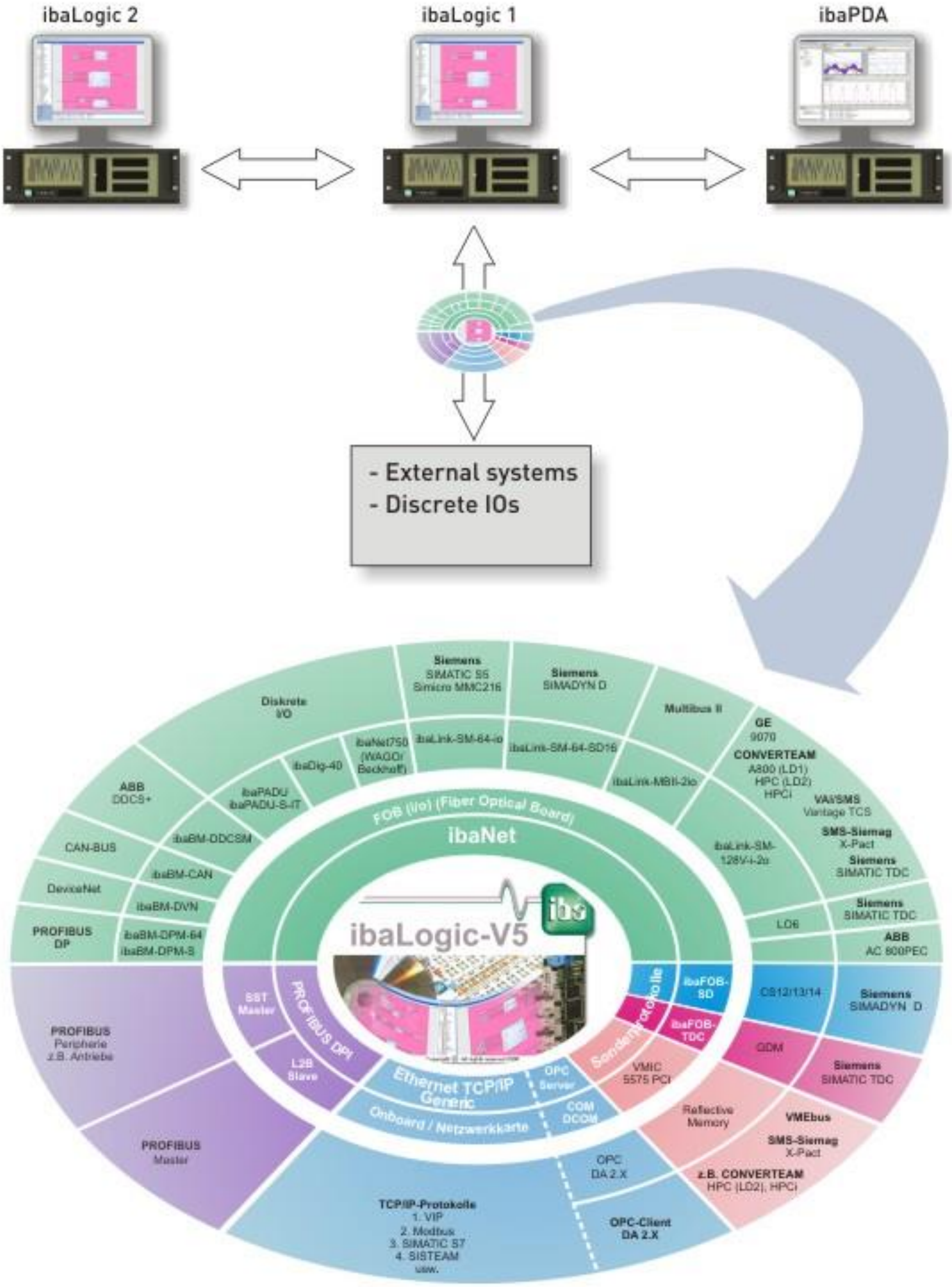


Figure 6: Connectivity to iba and external systems

5 ibaLogic Server

5.1 Functional overview of the ibaLogic Server

The ibaLogic Server is not only the central point of communication between ibaLogic Client and the PMAC, but it is also responsible for the management of the ibaLogic projects in the database. Similarly, in the background, it manages a connection to an active PMAC for loading / starting / stopping actions of the PMAC. This link is established via the ibaLogic Client.

Hence, the server can be divided into the following functions:

- ❑ Server operation
  - ▪ Start / Stop / Close
  - ▪ Database actions
Backup and restore
Resetting the entire current database
- ❑ Administrative settings
  - ▪ Set the port number for client connections
  - ▪ Set up connections to ibaLogic databases and their parameters
  - ▪ Configure auto start for the server and the local PMAC
  - ▪ Set up automatic backup of the current database
  - ▪ Set the general server options
  - ▪ Display status of / execute the database scripts (only for support purposes)

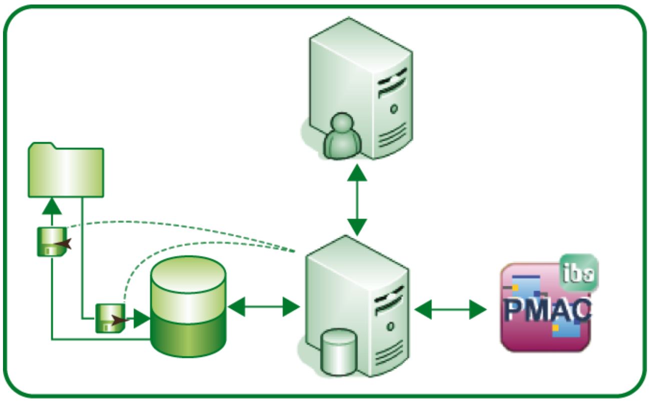
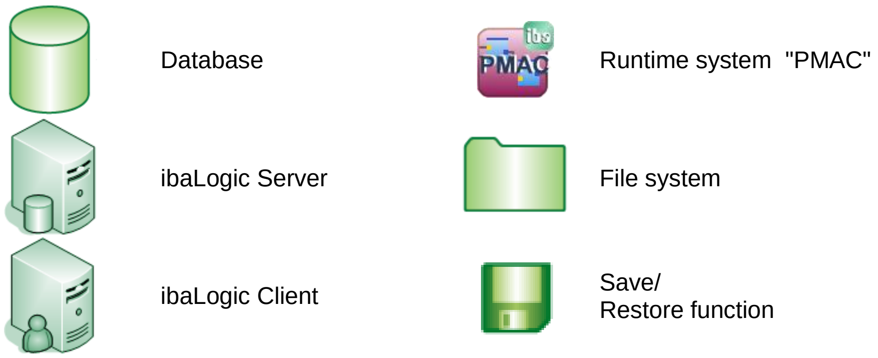


Figure 7: Functional overview of the ibaLogic Server



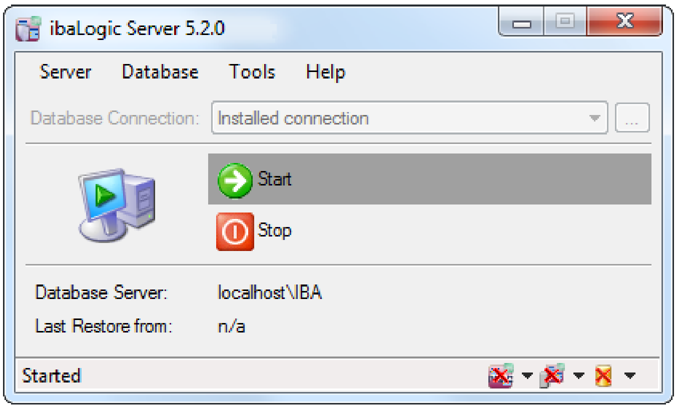
5.2 Start ibaLogic Server

Requirement

You have the ibaLogic Server shortcut on the desktop.

Procedure

1. Double click on the shortcut "ibaLogic Sever" on the desktop.
   The "ibaLogic Server" dialog box is displayed.



Database Connection: Installed connection

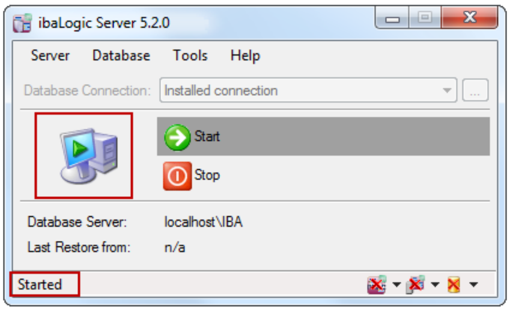
Database Server: localhost\IBA

Last Restore from: n/a

Started

When opening the ibaLogic Server, it goes automatically to the start status. The following icons are displayed in the info section.

2. Click on the <Start> button to start the ibaLogic Server. You can also start the server via the menu "Server - Start".
   The start / stop status is displayed in the server dialog box by an icon, a message and the active button.

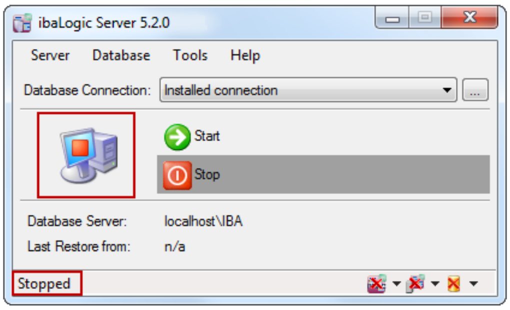


Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: n/a

Started



Database Connection: Installed connection

Database Server: localhost\IBA

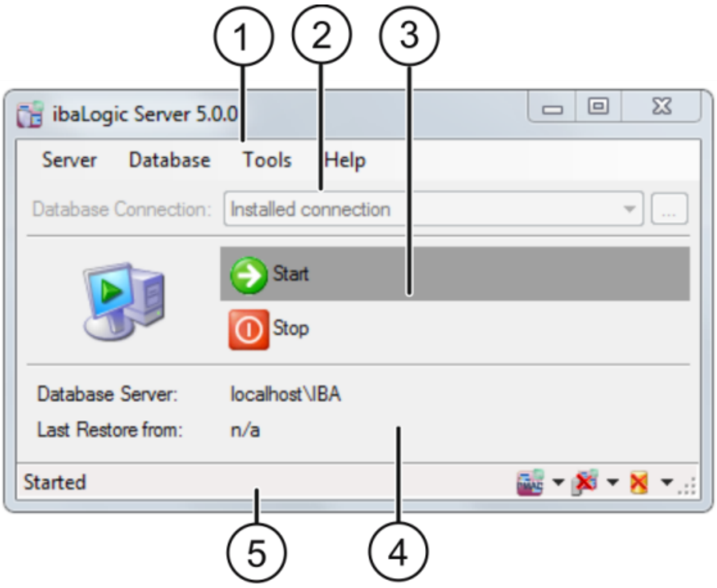
Last Restore from: n/a

Stopped

5.3 User Interface – ibaLogic Server

The ibaLogic Server is used to:

- ❑ Configure the server
- ❑ Put it in the start / stop mode
- ❑ Configure and save the database



Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: n/a

Started

Figure 8: ibaLogic Server - User interface

1	Menu bar	4	Current database connection
2	Setting up the database connection	5	Status bar
3	Start button / Stop button		

5.4 ibaLogic Server Setting

5.4.1 Configuring the Client port

The client port number serves as a link parameter for an ibaLogic Client.

Note

This setting should be changed only if a service that uses this port is being executed on the server computer. The same port must be configured in the client for connecting with this server. Enter the new port number also while selecting the link. Select the menu „File - Connect with Server..." of the client.

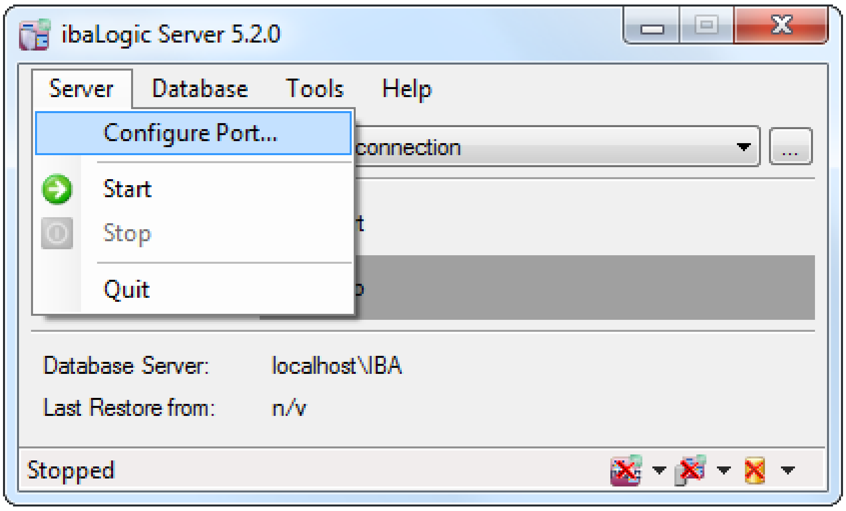
The default value of the port is set to 6510.

Prerequisite

❑ You have stopped the ibaLogic Server.

Procedure

1. Select the menu "Server - Configure Port…".

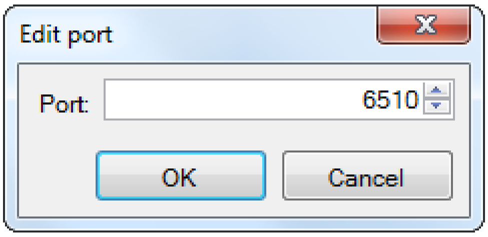


Database Server: localhost\IBA

Last Restore from: n/v

Stopped

2. Enter the port number of the client directly in the input field or set the port number using the spinner.



Port: 6510

5.4.2 Configuring the Database Connections

Important Note

If the server can no longer connect to the local database after changing the PC name, please change the connection name from the old computer name to "localhost" under "DataSource" . (See "*Configuring the Database Interface*, page 41")

You can set up multiple database connections in ibaLogic. However, only one of them is active at any given time. While installing ibaLogic, the database is created locally and the default setting refers to this.

If you change the computer name of your ibaLogic computer or if you are logged in with another user than the one who installed ibaLogic, then you may not be able to start ibaLogic or set it online, etc.

In this case, there are two new functions for the ibaLogic Server.

Add current user to database: The current user is added to the database so that he has access to the workspaces.

Rename computer in database instance: If you have changed the computer name, or changed the workgroup or domain, you can adjust the database to the computer name.

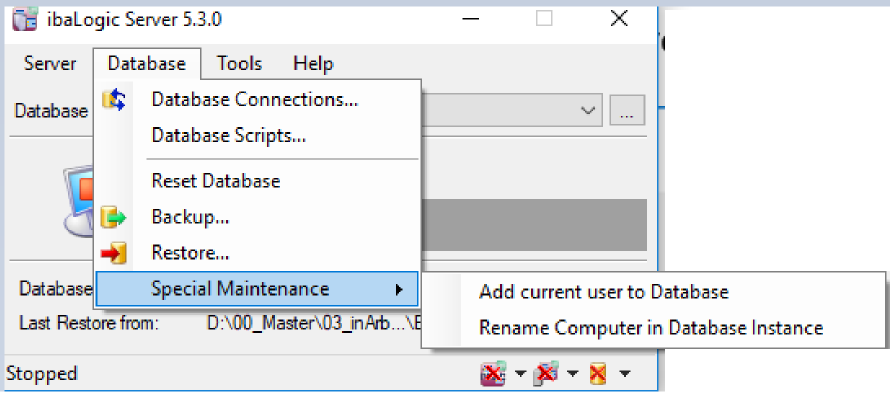


Figure 9: Menu Special maintenance

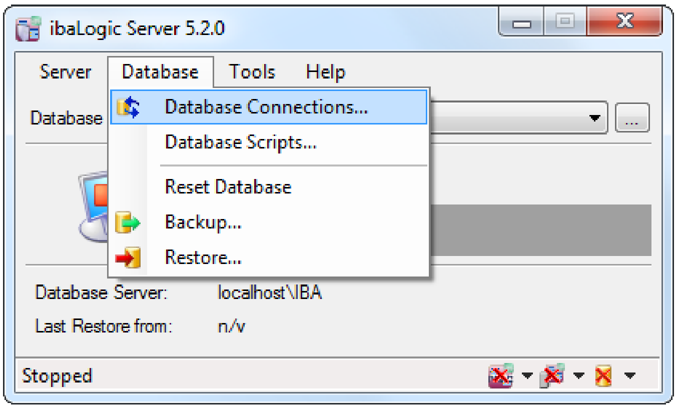
Note

You need to perform the following actions only if you want to connect to a **non-local** database which was not already specified at the time of the installation.

5.4.2.1 Connect database

Procedure

1. Click on "Database Connections..." in the Database menu.



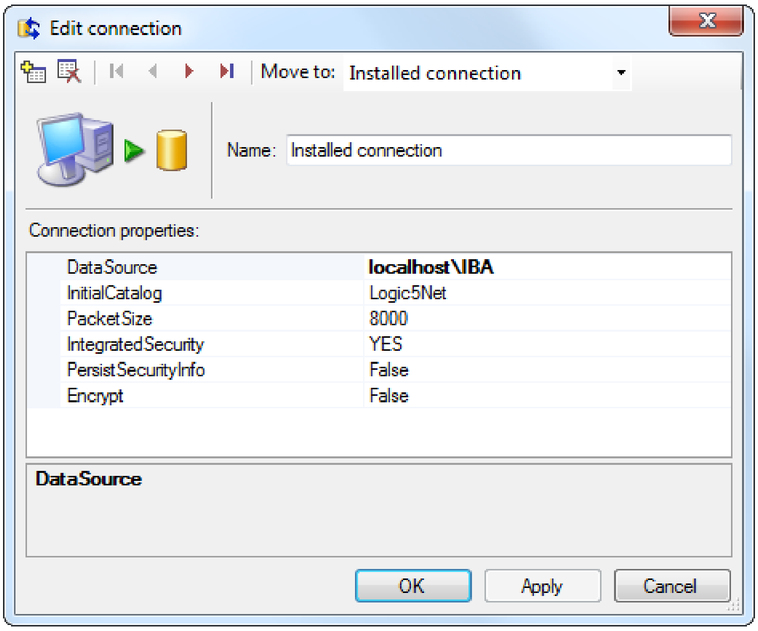
Database Server: localhost\IBA

Last Restore from: n/v

Stopped

2. Choose an ibaLogic database connection in the "Move To" drop down box. The default setting is the "Installed connection" database connection. This is the connection to the local database.

3. Call up the configuration dialog box for database connections with the <…> browser button. The browser button becomes visible only after clicking in the text field of the "DataSource" line.



Move to: Installed connection

Name: Installed connection

Connection properties:

DataSource	**localhost\IBA**
InitialCatalog	Logic5Net
PacketSize	8000
IntegratedSecurity	YES
PersistSecurityInfo	False
Encrypt	False

DataSource

⚠ CAUTION

The following parameters

- InitialCatalog
- PacketSize
- IntegratedSecurity
- PersistSecurityInfo
- Encrypt

need to be changed only if you, e. g. wish to use a database that is already available centrally for ibaLogic also.

However, you must have basic knowledge on databases for this purpose.
In case of doubt, please ask a database administrator to configure the settings.

5.4.2.2 Configuring the Database Interface

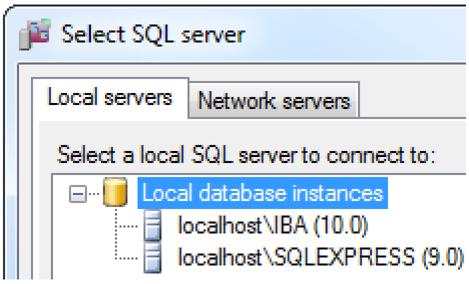
Enter the computer name and the instance of the ibaLogic database with which a connection needs to be established under "DataSource".

Procedure

➲ Enter the name of the server directly into the text field or select the database using the tree with the help of the browser button.
The browser button becomes visible only after clicking in the text field.

5.4.2.3 Select SQL server

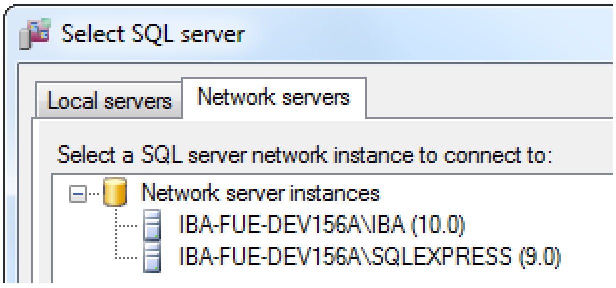
The "Select SQL server" dialog box contains 2 tabs "Local servers" and "Network servers".



Select a local SQL server to connect to:

- Local database instances
  - localhost\IBA (10.0)
  - localhost\SQLEXPRESS (9.0)

Figure 10: Local server instances



Select a SQL server network instance to connect to:

- Network server instances
  - IBA-FUE-DEV156A\IBA (10.0)
  - IBA-FUE-DEV156A\SQLEXPRESS (9.0)

Figure 11: Network server instances

All SQL database instances available on the computer are listed under "Local database instances".

After selecting the "Network servers" tab, the entire network available is searched for SQL instances. This process may take some time. As long as the network is being searched, a message "Information is being read" appears in the text field.

Procedure

1. Click on the SQL server instance to which the ibaLogic Server should connect.
2. Finally, click on <OK>. The SQL server instance is applied. The dialog box is closed.

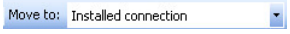
Result

The SQL server instance is connected with the ibaLogic Server.

Remark

Other database connections may be installed or deleted.

The following icons are available for configuring the database connection:

Icons / Selection box	Tooltipp	Explanation
➕	Add	Add a new database connection
✖	Remove	Delete a database connection
⏮	Start	To the first connection
◀	Back	To the previous connection
▶	Next	To the next connection
⏭	End	To the last connection
 Move to: Installed connection ▾	Choose object	Option for selecting a connection

5.4.2.4 Manage Database scripts

Important Note

The list of installed database scripts is used to provide information for the iba support. Please do not make any modifications.

All scripts implemented in ibaLogic along with the associated information such as the version number, script name and the date of installation are displayed in tabular form when you call up the function "Database scripts". The ibaLogic Server must be stopped in order to call up the "Database scripts" dialog.

Generally, the database scripts are checked via version updates and automatically installed after a previous check.

5.4.3 Options

5.4.3.1 Activate Autostart Server

The ibaLogic Server is automatically started on Windows startup.

You can choose where the autostart options are saved:

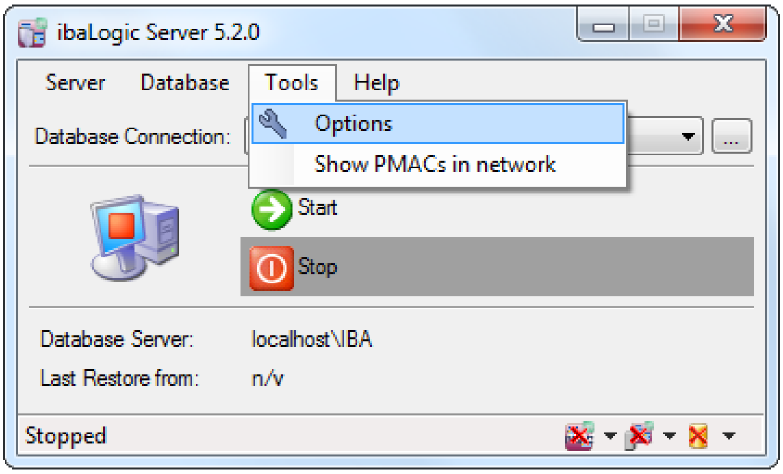
- ❑ In the registry
- ❑ In the autostart folder of the current user
- ❑ In the autostart folder for "All users"

The default setting is that the server also starts up when the ibaLogic Server dialog is opened. If the server dialog should be open, but the server itself in STOP mode, the option "Autostart server stopped" must be enabled. In this case, the server must be started up manually so that the client connections are accepted.

Autostart options	Explanation
Registry (recommended)	The option saves the autostart options in the registry file.
Startup folder (User)	The option saves the autostart options in the start-up folder of a given user.
Startup folder (All users)	The option saves the autostart options in the start-up folder for all users.
Autostart Server in stopped mode (not recommended)	This option opens the ibaLogic Server dialog, but the server itself is not started. The server remains in the stop mode.

Procedure

1. Select the "Tools - Options" menu.



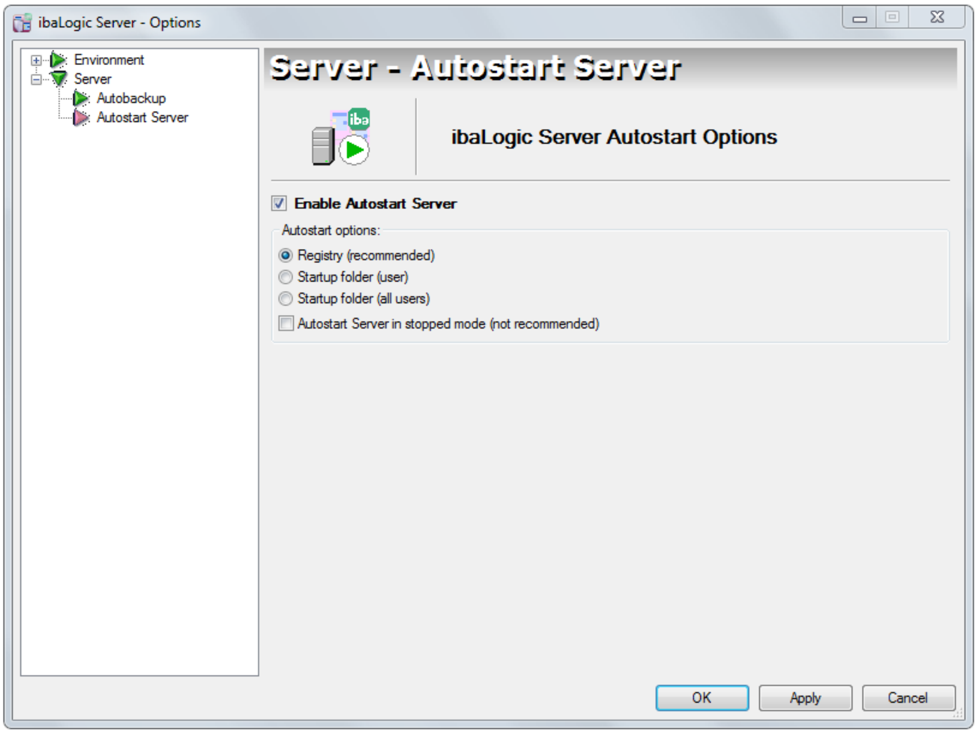
Database Connection:

Database Server: localhost\IBA

Last Restore from: n/v

Stopped

2. Choose "Server – Autostart Server" in the tree.
3. Click on the selection box "Enable Autostart Server".



ibaLogic Server Autostart Options

☑ Enable Autostart Server

Autostart options:

● Registry (recommended)

○ Startup folder (user)

○ Startup folder (all users)

□ Autostart Server in stopped mode (not recommended)

4. Click on <Apply> to activate the settings.

Result

The ibaLogic Server is automatically started on Windows startup.

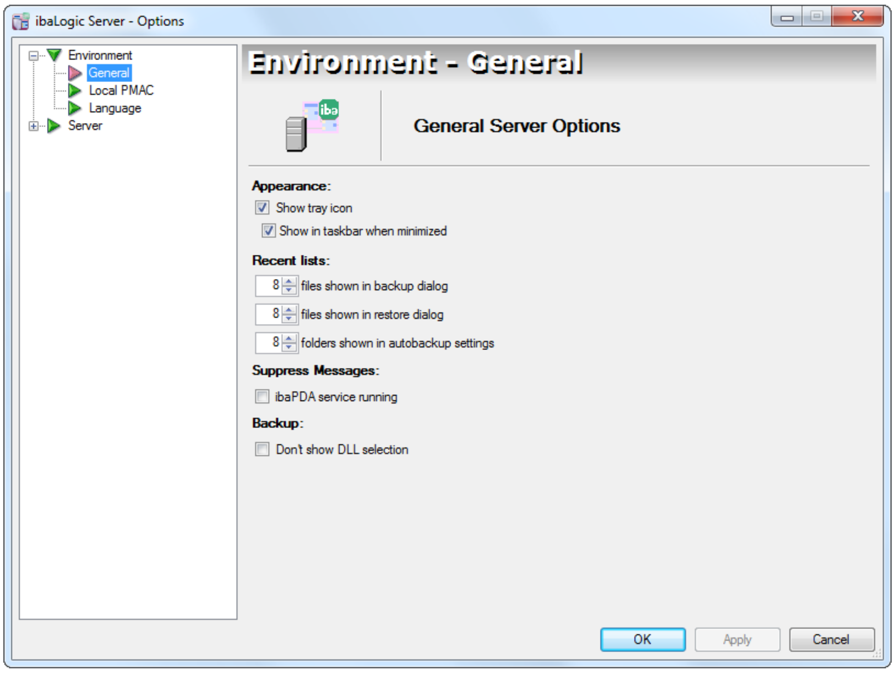
5.4.3.2 Configuring general ibaLogic server settings

The following general ibaLogic server settings can be configured:

Server option	Explanation
Show tray icon	If this option is enabled, an icon is displayed in the Info section when the ibaLogic server dialog is opened.
Show in taskbar when minimized	If this option is enabled, the ibaLogic server dialog appears in the task bar when minimized.
files shown in backup dialog	Number of files that are to be shown in the backup dialog screen. This can be chosen in the "Backup folder" selection box under "Server - Auto backup".
files shown in restore dialog	Number of files that are to be shown in the restore dialog screen.
folders shown in autobackup settings	Number of folders that are to be shown in the backup settings screen.
PDA service running	When starting the ibaLogic server, suppress the message that also ibaPDA is running on this computer
Backup: Don't show DLL selection	When a backup is stored, a dialog displays all DLLs in the directory. The dialog with the DLL list can be suppressed.

Procedure

1. Select the "Tools - Options" menu.
2. Select "Environment - General" in the tree.



Appearance:

☑ Show tray icon

☑ Show in taskbar when minimized

Recent lists:

8 files shown in backup dialog

8 files shown in restore dialog

8 folders shown in autobackup settings

Suppress Messages:

☐ ibaPDA service running

Backup:

☐ Don't show DLL selection

3. Configure the settings as desired.
4. Click on <Apply> to activate the settings.

5.4.3.3 Settings for the local PMAC

The local PMAC has been realized as a Windows service. Its status and startup type (Autostart options) are configured here.

Status setting:

❑ Activate

❑ Deactivate

Status option	Description
Activate	When enabling, the service might be automatically reinstalled.
Deactivate	By selecting between "PMAC full version" and "PMAC lite version", it can be switched between the two run time versions. In the full version, a dongle is mandatory. The lite version has some restrictions (see overview in "*Versions*, page 15")

Configuring the autostart options (start-up type):

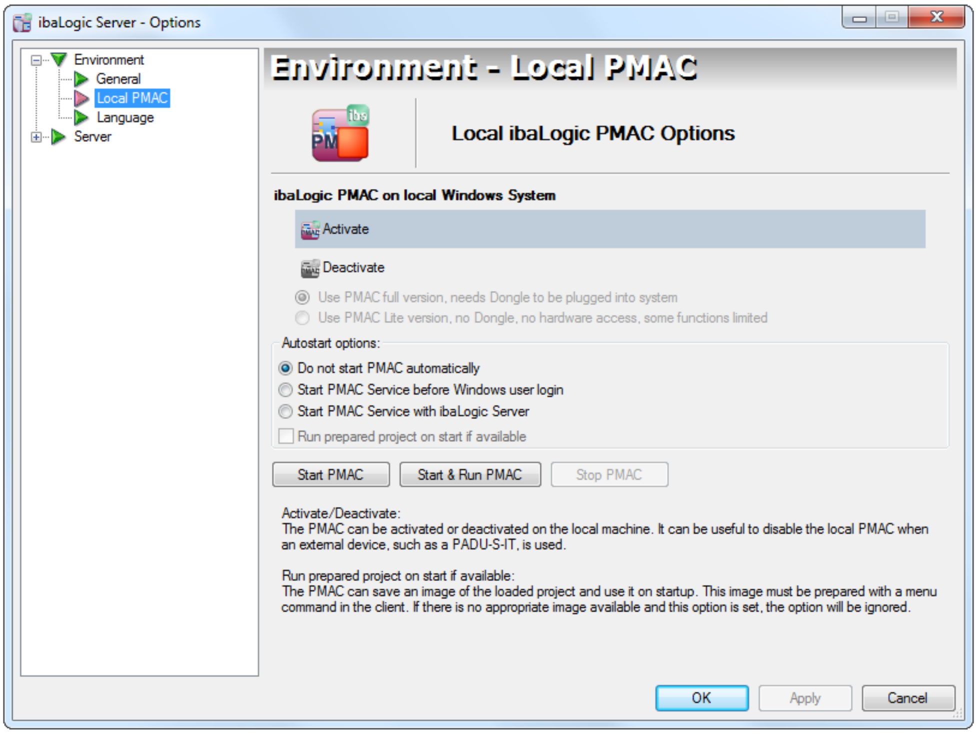
Autostart option	Explanation
Do not start PMAC automatically	The option ensures that the PMAC service does not start automatically.
Start PMAC Service before Windows user login	This option starts the PMAC service before the Windows user login. Additional option: Execute a prepared project at start-up, if available.

Start PMAC Service with the ibaLogic Server	This option starts the PMAC service along with the ibaLogic server. Additional option: Execute a prepared project at startup, if available.
Run prepared project on start if available	This option causes an image of a project to be loaded and used on startup. The image must be prepared in advance via a menu command in the client. If no image is available and this option was selected, the option is ignored.

Description for switching the PMAC versions (lite or full version)

Procedure

1. Stop the PMAC in the ibaLogic Server options dialog under "Environment - Local PMAC" and then disable and uninstall it.



ibaLogic PMAC on local Windows System

○ Use PMAC full version, needs Dongle to be plugged into system

○ Use PMAC Lite version, no Dongle, no hardware access, some functions limited

Autostart options:

● Do not start PMAC automatically

○ Start PMAC Service before Windows user login

○ Start PMAC Service with ibaLogic Server

□ Run prepared project on start if available

Activate/Deactivate:
The PMAC can be activated or deactivated on the local machine. It can be useful to disable the local PMAC when an external device, such as a PADU-S-IT, is used.

Run prepared project on start if available:
The PMAC can save an image of the loaded project and use it on startup. This image must be prepared with a menu command in the client. If there is no appropriate image available and this option is set, the option will be ignored.

2. Now select the requested PMAC version (full version or lite), "Activate" the service again (it will be installed in doing so) and <Start PMAC> or, depending on the selected autostart option, with the next server start.

3. Select an autostart option.
If the autostart service is enabled, you can choose from the autostart options provided.
In addition, you can choose that the project should be started.
For this, it is necessary that this project has been "saved in PMAC" previously.

4. Click on <Apply> to activate the settings.

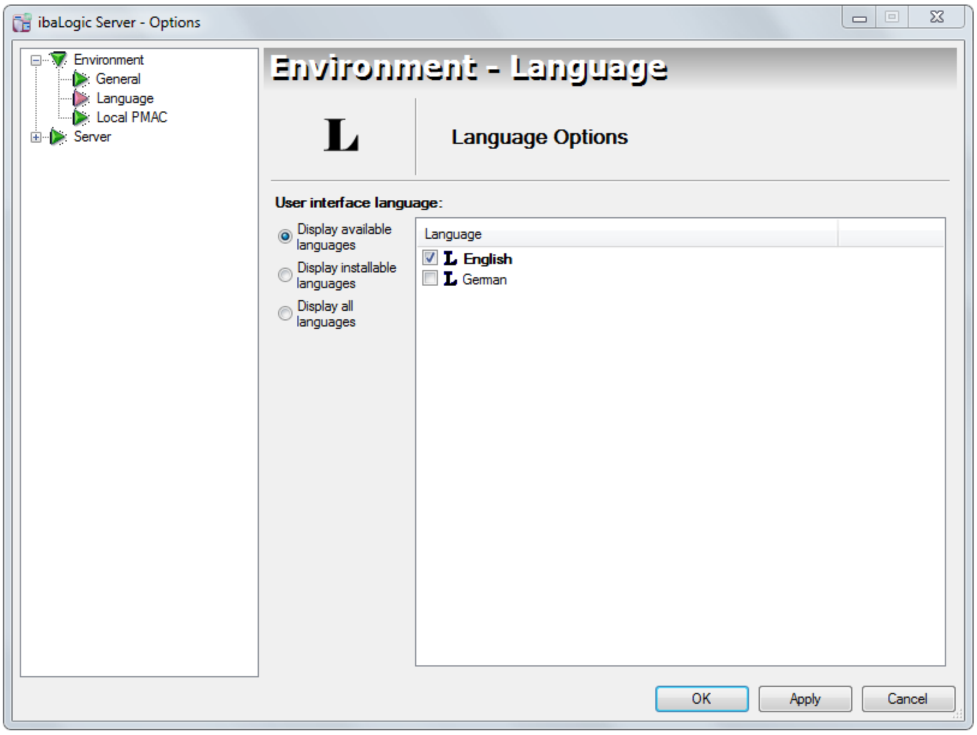
5.4.3.4 Language

Within this section, you configure the language of the server dialog.

The language of the client dialog has to be configured separately.

Procedure

1. Select the "Tools - Options" menu.
2. Select "Environment – Language" in the tree.



User interface language:

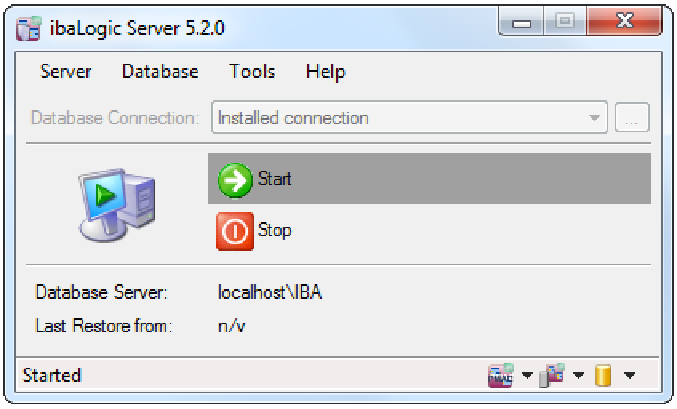
● Display available languages
○ Display installable languages
○ Display all languages

Language
☑ English
□ German

3. Select "Language Options".
4. Choose the selection box with the desired language.
5. Click on <Apply> to activate the settings.

5.4.4 Status bar

There are 3 icons in the status bar in the ibaLogic Server dialog screen. The icons are used to activate or deactivate the autostart and save settings.



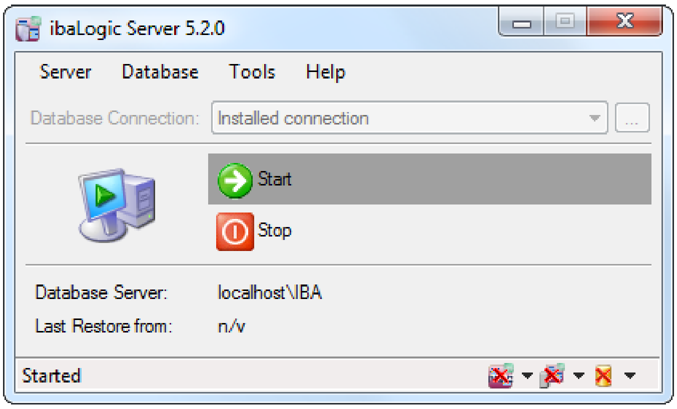
Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: n/v

Started

Figure 12: Functions activated in the status bar



Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: n/v

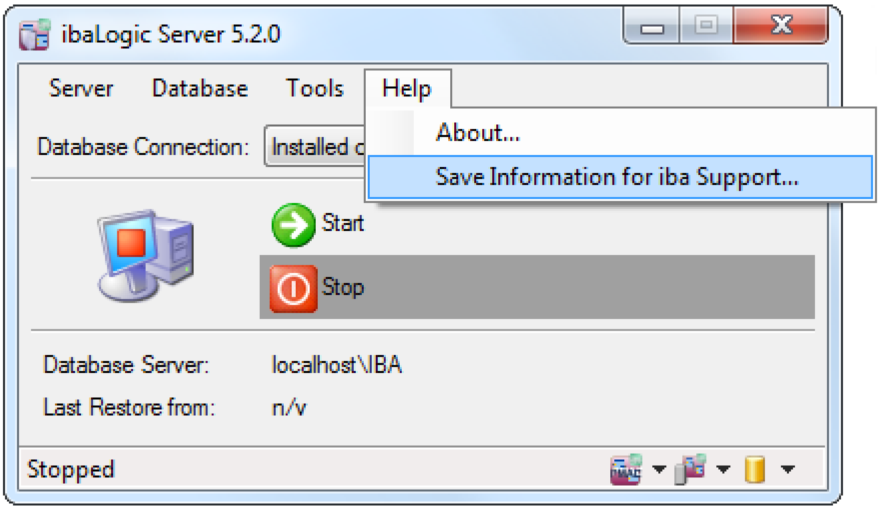
Started

Figure 13: Functions deactivated in the status bar

Icon	Setting	Description
PMAC	Autostart (Start before login)	Activation causes the PMAC to be started automatically every time Windows starts up.
	Autostart (Registry)	Activation causes the autostart options to be saved in the registry file.
	Automatic backup	Activation causes the database to be backed up in accordance with the settings.

5.4.5 Save information for the iba Support

Under “Help - Save Information for iba Support...”, a zipped file can be generated which makes it easier for the iba Support to detect or localize the error in case of an error.



Database Connection: Installed c

Database Server: localhost\IBA

Last Restore from: n/v

Stopped

This file contains the current project, the hardware configuration, the log files and other information.

6 Programming Environment – ibaLogic Client

The ibaLogic Client is used to create and edit programs.

6.1 Start ibaLogic Client

The programming environment is displayed.

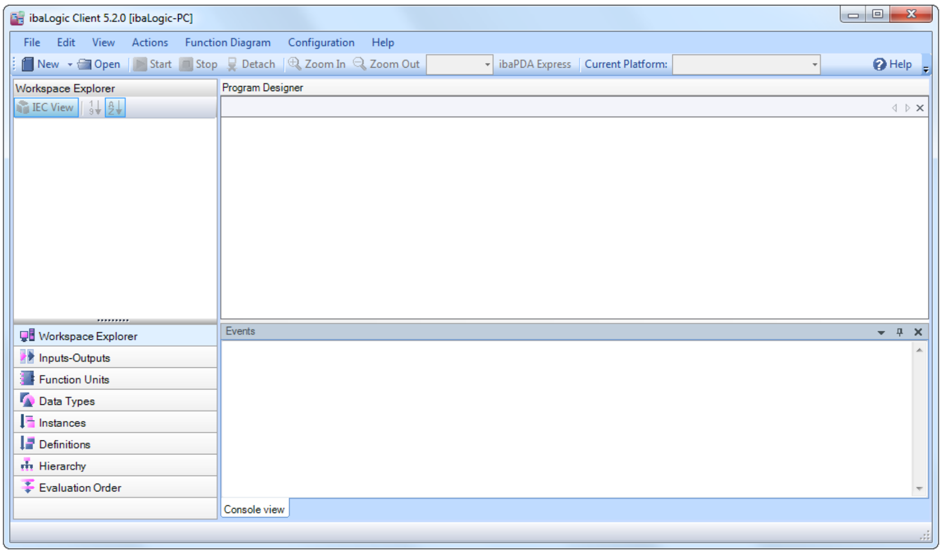
Prerequisite

❑ You have created the start icons for ibaLogic Client and ibaLogic Server on the desktop (Standard installation).

❑ You have started ibaLogic Server.

Procedure

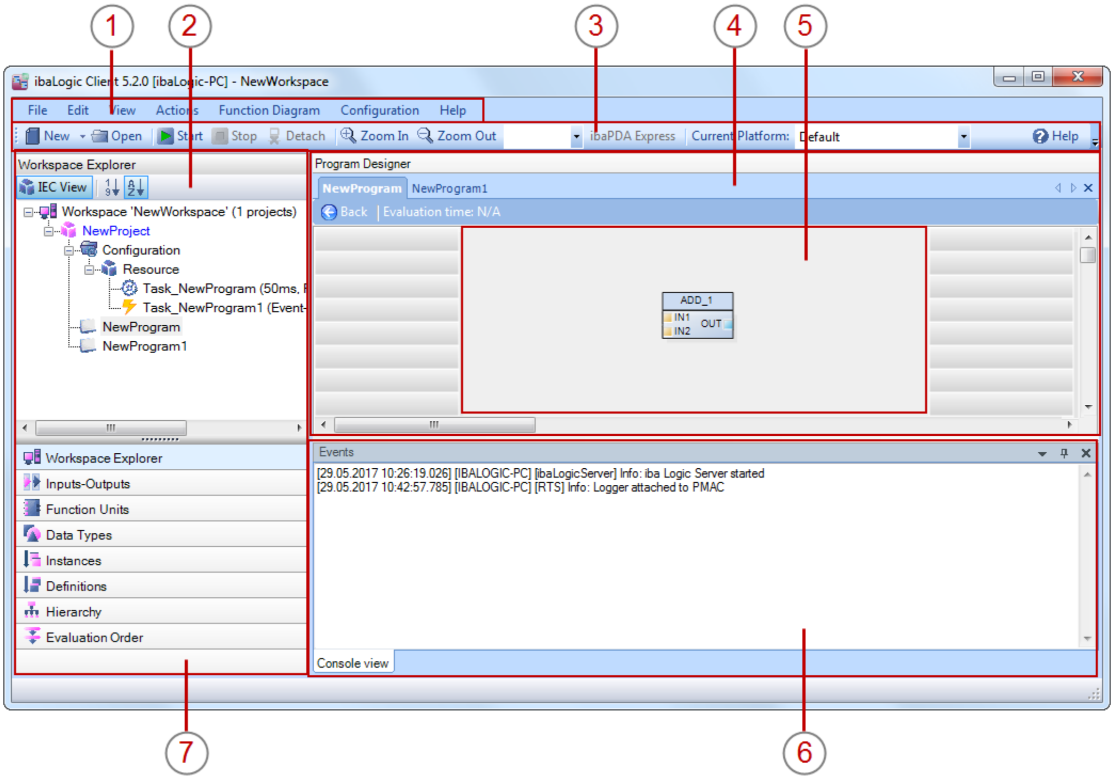
➲ Double click on the "ibaLogic Client" icon on the desktop.
The ibaLogic client dialog screen opens after a brief initialization phase.



Remarks

The event window below the program window documents the program actions and eventual collisions.

6.2 User Interface of Programming Environment – Editor



[29.05.2017 10:26:19.026] [IBALOGIC-PC] [ibaLogicServer] Info: iba Logic Server started
[29.05.2017 10:42:57.785] [IBALOGIC-PC] [RTS] Info: Logger attached to PMAC

Figure 14: User Interface

1	Menu bar	5	Programming field
2	Navigation area	6	Window for events
3	Toolbar	7	Navigation buttons
4	Program designer		

6.2.1 Menu Bar

The menu bar is the central control element of the ibaLogic Client.

Figure 15: Menu Bar

6.2.2 Toolbar

The toolbar is the secondary control element of the ibaLogic Client.

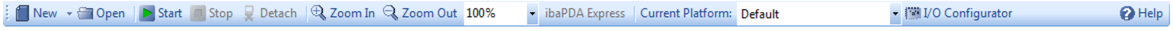

New Open Start Stop Detach Zoom In Zoom Out 100% ibaPDA Express Current Platform: Default I/O Configurator Help

Figure 16: Toolbar

The icons of the toolbar are also buttons at the same time.

6.2.3 Navigation Area

The navigation area contains buttons for:

❑ Workspace Explorer

❑ Inputs – Outputs

❑ Function Units

□ Data Types

□ Instances

□ Definitions

□ Hierarchy

□ Evaluation Order

□ Feedbacks

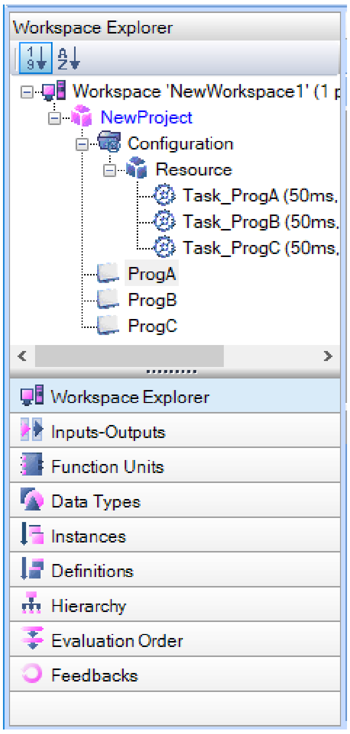


Figure 17: Navigation buttons

6.2.3.1 Switch views in the Workspace Explorer

In the menu bar of the Workspace Explorer, there are buttons to switch the view of the Workspace Explorer. It can be arranged by priority or in alphabetic order.

IEC View

This view shows the structure of the ibaLogic workspace according to the IEC 61131-3 standard.

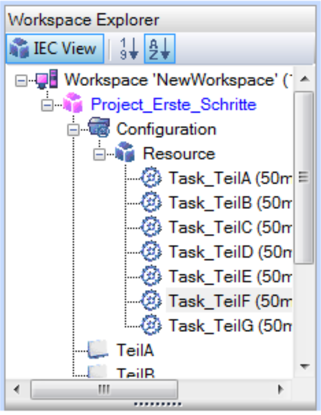


Figure 18: IEC View

Note

In contrast to the IEC 61131-3 standard, only one program is assigned to each task.

This view can be sorted by name or priority.

Symbol	Explanation
AZ↓	Sorts the entries by names.
19↓	Sorts the entries by their priority.

6.2.3.2 Instances

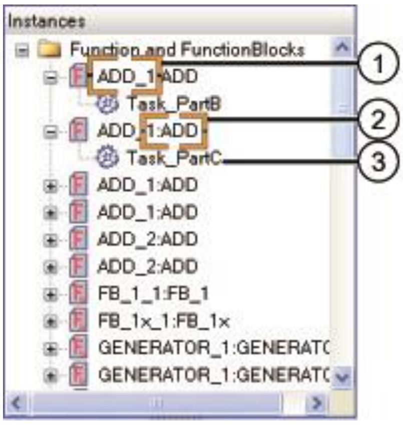
All function units used in the project are listed by instance names in the instances view. The definition name is also displayed, separated by a colon.

Double clicking on the instance name displays the program page in which the function unit is placed. The function unit is marked in the process.

Note on difference between Definition – Instance

The definition of a function unit is saved in the project library. The program always contains an instance (virtually, a copy) of the function unit. The instance name is automatically formed from the "Definition name_Index". However, you have the possibility to change the name afterwards.

When you modify the contents of a function unit, you also modify the definition and the other instances.



```
Instances
⊟ Function and FunctionBlocks
  ⊟ ADD_1:ADD
      Task_PartB
  ⊟ ADD_1:ADD
      Task_PartC
  ⊞ ADD_1:ADD
  ⊞ ADD_1:ADD
  ⊞ ADD_2:ADD
  ⊞ ADD_2:ADD
  ⊞ FB_1_1:FB_1
  ⊞ FB_1x_1:FB_1x
  ⊞ GENERATOR_1:GENERATC
  ⊞ GENERATOR_1:GENERATC
```

Figure 19: Instances

1	Instance name	3	Task containing this function unit
2	Definition type		

If a function unit is placed in a nested macro, it is displayed in a tree structure:

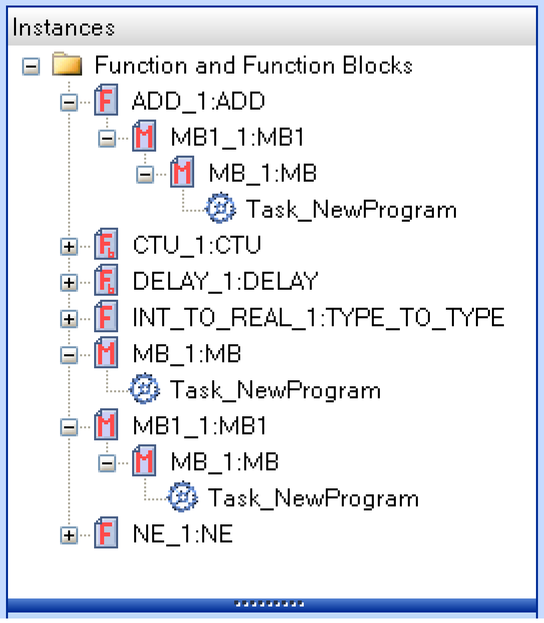

Instances
Function and Function Blocks
ADD_1:ADD
MB1_1:MB1
MB_1:MB
Task_NewProgram
CTU_1:CTU
DELAY_1:DELAY
INT_TO_REAL_1:TYPE_TO_TYPE
MB_1:MB
Task_NewProgram
MB1_1:MB1
MB_1:MB
Task_NewProgram
NE_1:NE

Figure 20: Instance view as a tree structure

6.2.3.3 Definitions

All function units present in the project are displayed in this view arranged in the order of their definition names. The tree structure contains instances located below the function block type, and below these, macros, if any, and finally the task with the program names.

Double clicking on the instance names displays the program page in which the function unit is placed. The function unit is marked in the process.

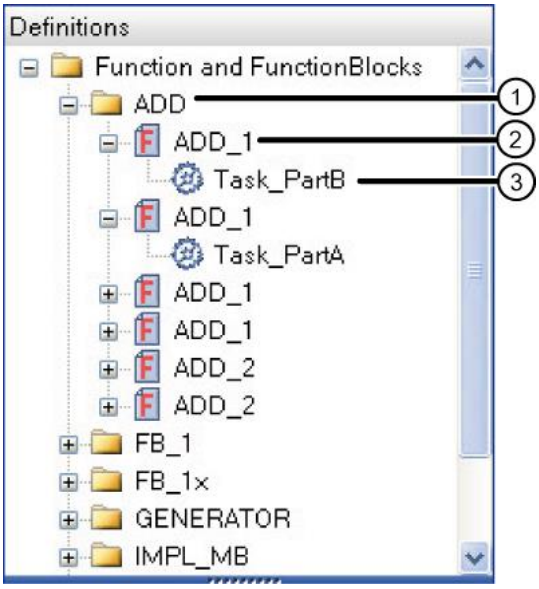

Definitions
Function and FunctionBlocks
ADD
ADD_1
Task_PartB
ADD_1
Task_PartA
ADD_1
ADD_1
ADD_2
ADD_2
FB_1
FB_1x
GENERATOR
IMPL_MB

Figure 21: Definitions

1	Definition name	3	Task name
2	Instance name		

6.2.3.4 Hierarchy

All instances of the function unit arranged alphabetically by tasks are displayed within the hierarchy view.

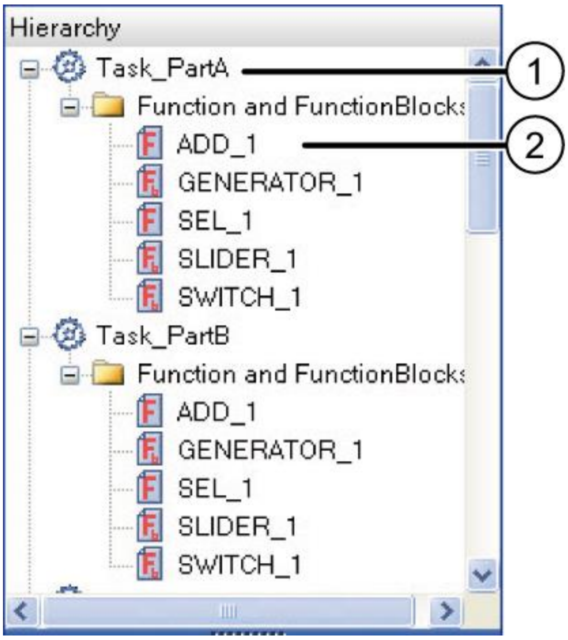
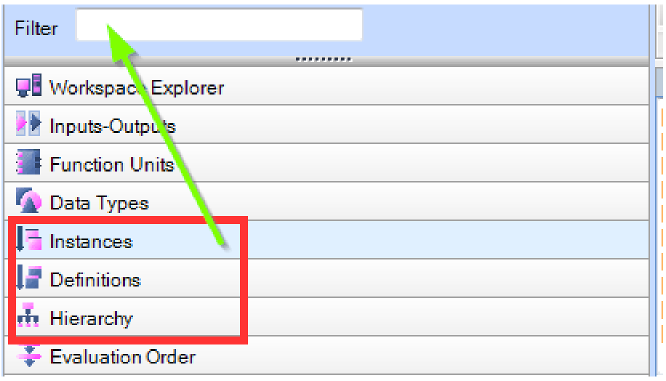


Figure 22: Hierarchy view

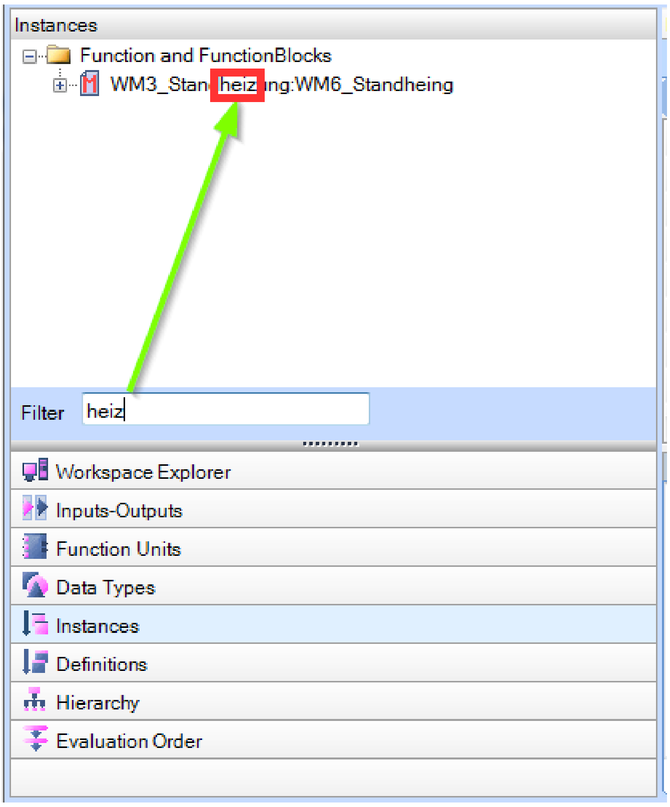
1 Task 2 Instance name

6.2.3.5 Search filter

A filter field for a text search is available in the tree structure. The filter field is availabe in the instance view, definition view and hierarchy.



The filter function can be used to limit the number of blocks accordingly in order to find them more quickly. Example:



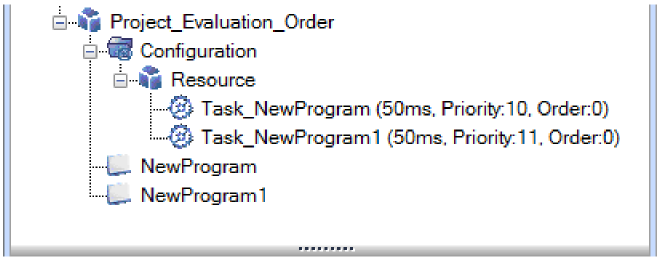
6.2.3.6 Evaluation order

The "Evaluation Order" view shows the sequence in which the programs and function units are evaluated within the programs.

The function units evaluated first are displayed at the top.

Example

2 tasks with identical interval time but different priority.



Project_Evaluation_Order
Configuration
Resource
Task_NewProgram (50ms, Priority:10, Order:0)
Task_NewProgram1 (50ms, Priority:11, Order:0)
NewProgram
NewProgram1

Figure 23: 2 tasks with identical interval time but different priority

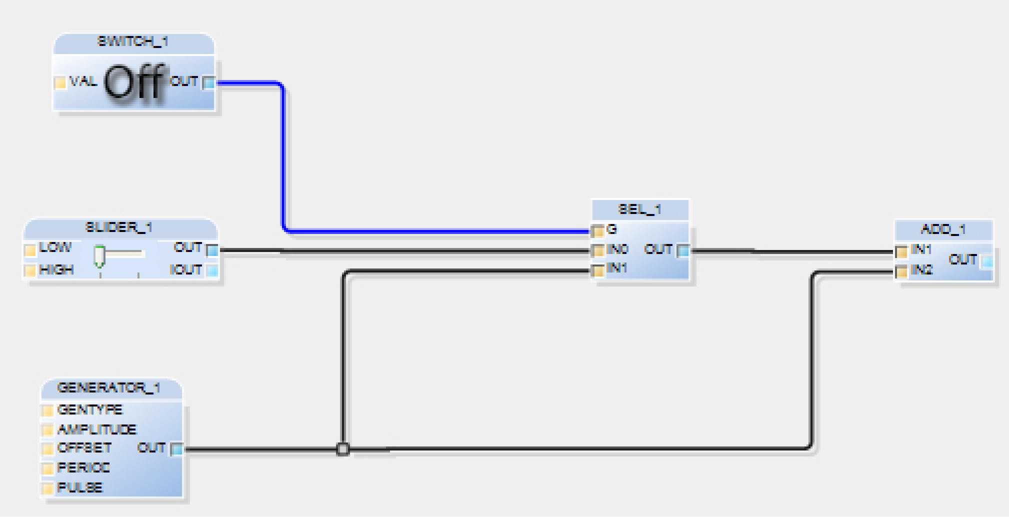
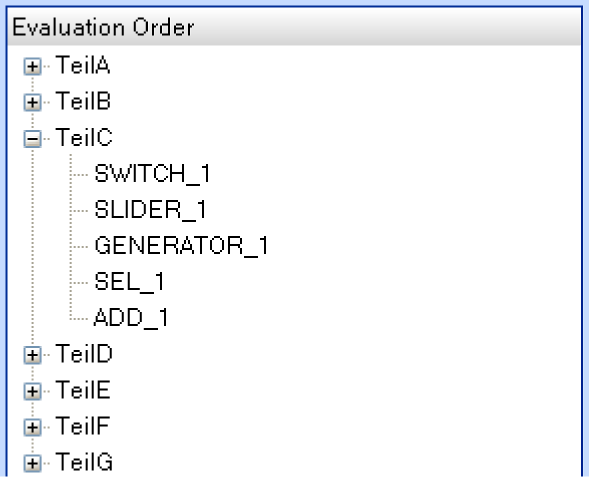


Figure 24: NewProgram with the following contents

The Evaluation Order is displayed as follows:



Evaluation Order

- TeilA
- TeilB
- TeilC
  - SWITCH_1
  - SLIDER_1
  - GENERATOR_1
  - SEL_1
  - ADD_1
- TeilD
- TeilE
- TeilF
- TeilG

Figure 25: Evaluation Order

Rules for the Evaluation Order

❑ The interval tasks are executed according to the interval assigned.

❑ Interval tasks that are to be started simultaneously are evaluated according to their order, whereby 0 is the first within the order.

❑ Event tasks are started with the triggering event.

❑ Tasks are interrupted. This means that a higher priority task interrupts a task with a lower priority that is currently running.

❑ The evaluation within a program takes place in accordance with the following rules:

- Based on the data flow, those function units generating the data are always evaluated first. Thereafter, those are evaluated that use the data.
- If there are multiple independent branches in a program, the supplementary rule applicable is that evaluation takes place from the top left to the bottom right. This means that a branch located above is evaluated first.
- In a feedback loop within a program or macro, the function unit placed at the uppermost left position is evaluated first.

⚠ DANGER

Danger due to modifications in the Evaluation Order!

This has the consequence that (only in feedback loops) by shifting the function units, the evaluation order and, thus, the results may change.

Tip

In order to prevent this, it is recommended to encapsulate function units in a macro and to implement the feedback outside the macro. In this manner, the evaluation order is defined uniquely independent of the positioning.

The macro block is a function unit. Within the macro the function units are evaluated in accordance with the rules given above. In other words, the output evaluated is considered as a new input value only in the next clock cycle.

Tip

The content of a function block is evaluated in one cycle.

This means, for example, that when you access an input connector within a For loop, you get the same value in each execution of the loop, since the connector is re-evaluated only in the next cycle. If you wish to have a loop across multiple cycles, you must obtain the feedback from the control variable external to the function block.

6.2.3.7 Export of the evaluation order

Using the export function the evaluation order is exported as txt files.

Select *Evaluation order* in the navigation area. Mark a project, open the context menu with the right mouse button and select one of the following options:

❑ *Export Project*: Export the entire evaluation order to a single file

❑ *Export Project as Fileset*: Each task is exported to a single file

❑ *Export Selected Branch*: Only the selected branch is exported to a file. This allows you to export only one task / macro etc.

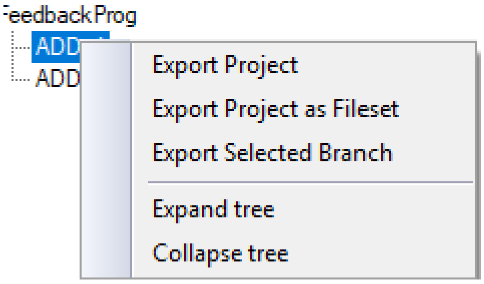


Figure 26: Context menu Export Project

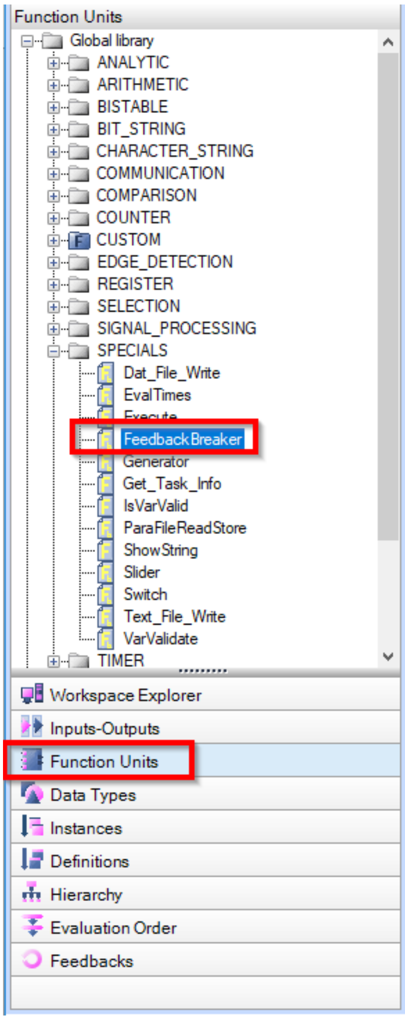
Note

The calculation algorithm was improved between ibaLogic versions 5.2.4 and 5.3.0. A fundamentally different behaviour of ibaLogic is not generally to be expected. However, in individual cases it could lead to a different behavior for timing tricks and calculations. With the export function it is possible to compare the different calculation sequences. The user is responsible for the evaluation, he must ensure the correctness of the calculations by suitable corrections in the program.

6.2.3.8 Feedback handling

The calculation order for feedbacks automatically depends on the logic of the interconnected blocks, i.e. from top to bottom and from right to left. Therefore, depending on the layout, a different order of evaluation results.

However, the order can also be fixed by determining the feedback point with the FeedbackBreaker function block.

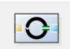
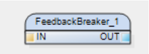


Function Units
- Global library
  - ANALYTIC
  - ARITHMETIC
  - BISTABLE
  - BIT_STRING
  - CHARACTER_STRING
  - COMMUNICATION
  - COMPARISON
  - COUNTER
  - CUSTOM
  - EDGE_DETECTION
  - REGISTER
  - SELECTION
  - SIGNAL_PROCESSING
  - SPECIALS
    - Dat_File_Write
    - EvalTimes
    - Execute
    - FeedbackBreaker
    - Generator
    - Get_Task_Info
    - IsVarValid
    - ParaFileReadStore
    - ShowString
    - Slider
    - Switch
    - Text_File_Write
    - VarValidate
  - TIMER

Workspace Explorer
Inputs-Outputs
Function Units
Data Types
Instances
Definitions
Hierarchy
Evaluation Order
Feedbacks

Figure 27: Function block FeedbackBreaker

The graphical representation of the block is bound to the option *Iconic display of the converters* in the options dialog:

	option is set (default)
	option is not set

The FeedbackBreaker can be placed as a feedback point and marks the end point of the feedback. The order is now clearly defined.

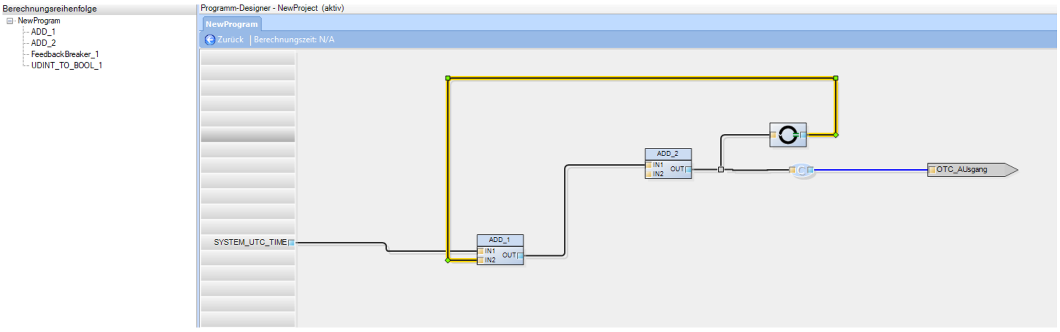


Figure 28: FeedbackBreaker as feedback point

For a better understanding and for a better overview, it is recommended to work with IntraPage connectors. Example:

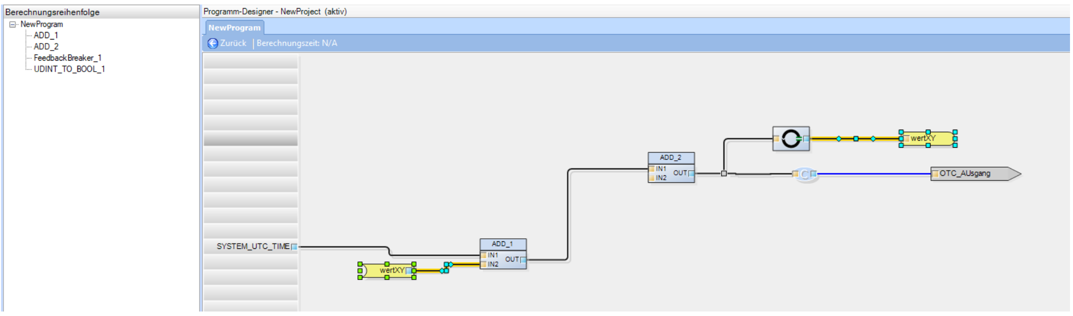


Figure 29: Layout with IntraPage connectors

6.2.3.9 Directory of feedbacks

In the *Feedbacks* view, you will find all the feedbacks present in the project.

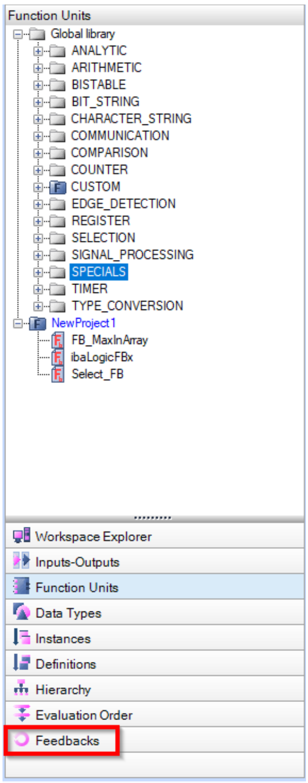


Figure 30: Feedbacks view

By double-clicking on the feedback, the corresponding blocks are marked.

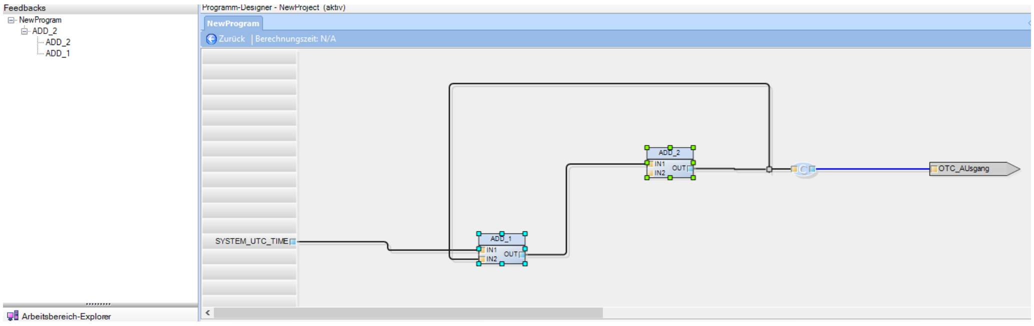


Figure 31: Marked blocks in a feedback

If you want to eliminate a feedback in older large layouts, it may be difficult to find the feedback point.

If you are looking for feedbacks, it is recommended to use the Program Overview window for help.

Procedure:

1. Mark feedback blocks by double-clicking on the feedback in the feedback directory.
2. Now you can navigate in the overview window and search for the blocks. By means of the markings you can also determine which is the last building block.
3. You can also jump to the individual blocks of the feedback by double-clicking them. If you look at the overview window, you might see which function block is at the bottom of the layout. Often the feedback has an output at its end.
4. You can also select a line while pressing the <Alt> key. This selects the complete connection, even beyond IPCs. By scrolling through the overview window, you can see if this line reappears at the top of the layout. This can help finding a feedback.

6.2.4 Program designer

Function units are placed and linked with one another in the program designer.

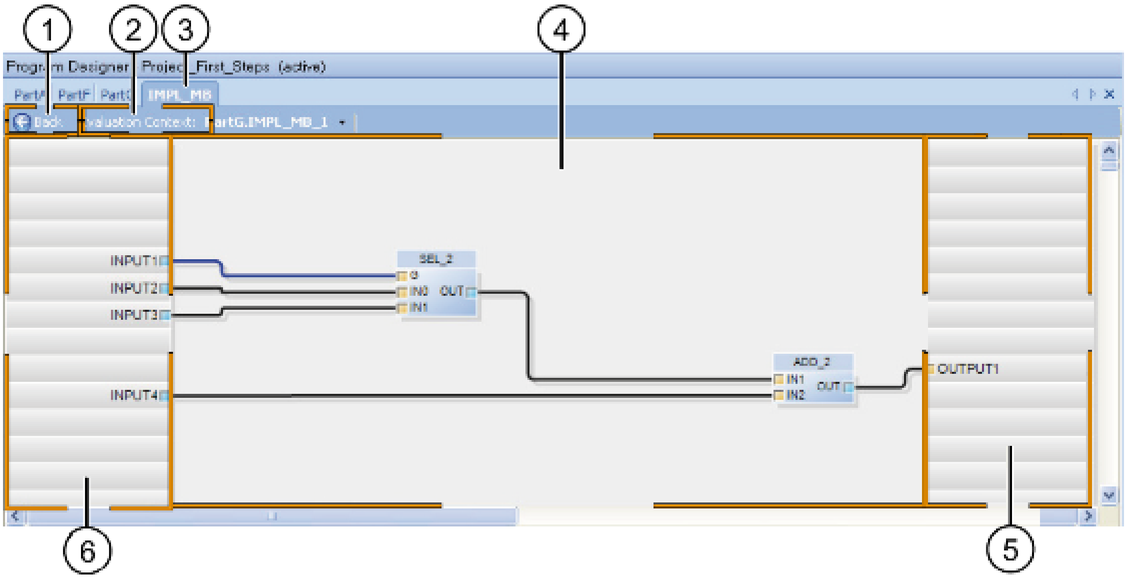


Figure 32: User interface of the program designer

1	<Back> button	4	Programming window
2	Evaluation context	5	Output area
3	Tab	6	Input area

6.2.5 Arrangement of the Tabs and Programming Windows

You can define the arrangement using the mouse.

The following options are available:

- ❑ Overlapping
- ❑ Horizontally or vertically

6.2.5.1 Arrange tabs

Proceed as follows to arrange the tabs:

Procedure

1. Click on the tab of the program concerned.
2. Keep the mouse button pressed, and move the tab to the new position.

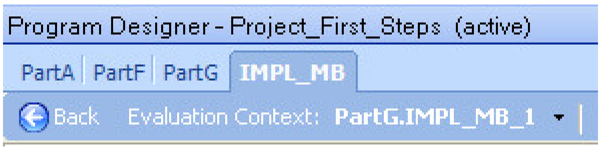

Program Designer - Project_First_Steps (active)
PartA
PartF
PartG
IMPL_MB
Back
Evaluation Context: PartG.IMPL_MB_1

Figure 33: Arrangement in the form of register tabs

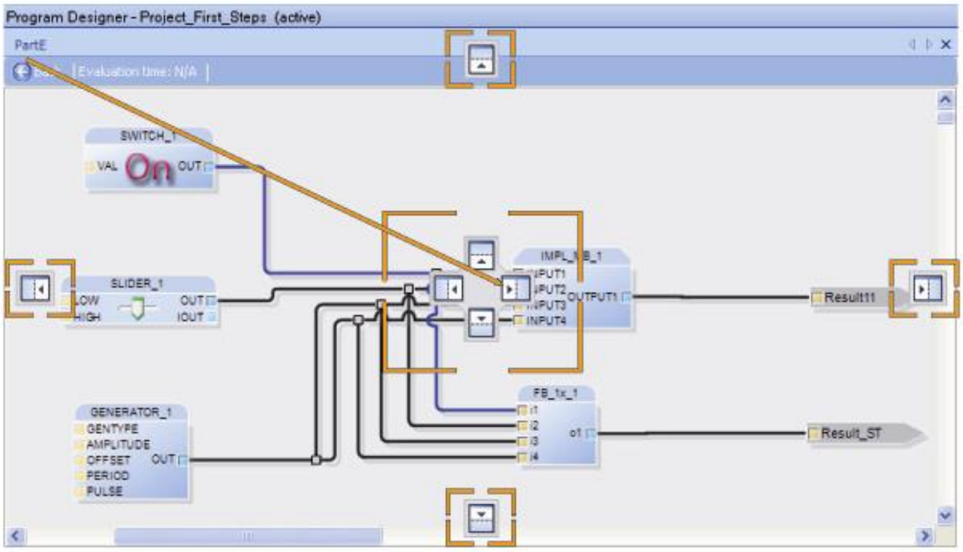
6.2.5.2 Arrange programming windows

Dockable windows are movable windows that can be shifted to any position on the screen and can be coupled to the docking marks.

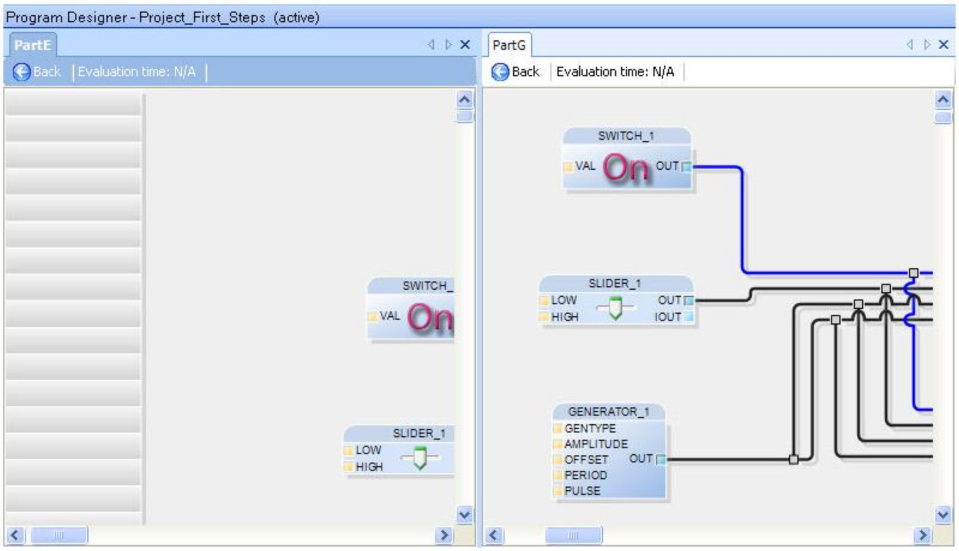
Proceed as follows to arrange the windows:

Procedure

1. Move the mouse pointer to the tab of the window that you wish to shift.
2. Press the left mouse button and keep it pressed.
3. Move the window to the docking mark at which you wish to position the tab.



4. Release the mouse button. The window is now anchored at this position.



Note

Program windows cannot be positioned as desired.

The programs and macros that are opened are displayed graphically in the programming field. Since macro blocks are displayed exactly in the same way as programs, they are not described in the following sections.

Toolbar

You can use the following functions in the toolbar:

- ❑ Back ( )
- ❑ Evaluation context (only with macros)
- ❑ Evaluation time (only in the online mode)

Back

You can use this button to navigate in the program window. The last selected program / macro is displayed.

Evaluation context (only with macros)

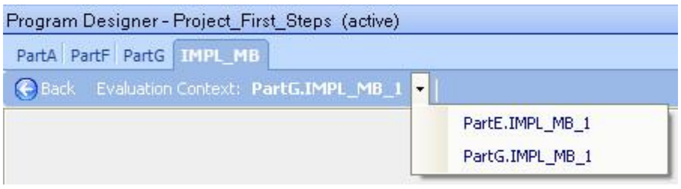
It is used to display the program or macro level in which the current plan is located. Since the instances of a macro block can occur several times in one or in various programs with different call parameters, you can see the program and instance in which the macro currently open is located.

If one macro is located in another macro (nesting), the entire hierarchy branch is displayed separated by dots:

Display: Program_name.Macro_1.Macro_2...

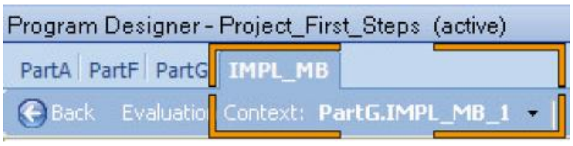
You can select the context within an open macro, i. e. you can switch from one instance to another. The values seen in the online view are always related to the instance selected.

1. To do this, click on the arrow button on the right side beside the macro instance name.



Program Designer - Project_First_Steps (active)
PartA
PartF
PartG
IMPL_MB
Back
Evaluation Context: PartG.IMPL_MB_1
PartE.IMPL_MB_1
PartG.IMPL_MB_1

2. You can select the calling level from the macro when you click the instance name in the evaluation context directly.



Program Designer - Project_First_Steps (active)
PartA
PartF
PartG
IMPL_MB
Back
Evaluation Context: PartG.IMPL_MB_1

The area of the program field window that contains this macro instance is displayed and the macro block is marked.

Note

If several macros are nested, it is possible to jump back one level by clicking on the evaluation context.

Evaluation time (online mode only)

It is displayed in each program as a percentage value with respect to the task interval and as an absolute value in ms (floating mean value).

6.2.5.3 Navigating in the Program Designer

You have several options to navigate in a program:

- ❑ Zoom
- ❑ Program overview
- ❑ Hot keys

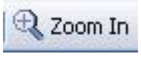
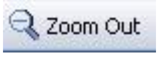
Zoom

You have the option to enlarge (zoom in) or reduce (zoom out) the size of the programming field in the program designer.

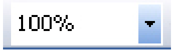
The following possibilities for zooming out or zooming in are available:

- ❑ Use of the buttons <Zoom In> / <Zoom Out>
- ❑ Use of the picklist
- ❑ Use of the hot keys

Buttons

➲ Press the button.  or  in the ibaLogic Client toolbar, to zoom in or zoom out the view by 20 % at each click in the programming field of the program.

Pick list

➲ Select between the 5 preset zoom-values in the pick list 

or

➲ Enter a zoom factor using the keyboard from 25 to 200 percent in the drop down box (pick-list).

Key combination

➲ Press <Ctrl> and roll simultaneously the scroll wheel of the mouse.

The minimum zoom factor depends on the screen resolution.

Program overview

Since one program page generally exceeds one screen page and only a partial window of the program can be seen depending on the zoom factor, the program overview serves as an orientation guide.

Open and use program overview

Procedure

1. Select "View - Program Overview" in the main menu.

   A miniature view of the "Program Overview" window is opened in the foreground of the current program window.

The visible section is displayed as a transparent rectangle.

2. Move the rectangle using the mouse until you see the desired program area in the program designer.

Annotation

The overview window can be placed arbitrarily.

It can also be docked outside the programming field or at its margin.

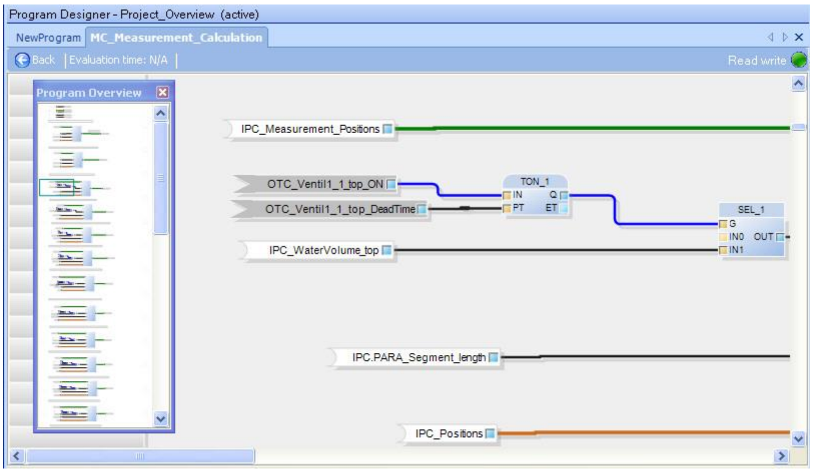


Figure 34: Programming field with displayed program overview window

Navigate in the programming field

The navigation in the program window is generally done with the mouse.

In the following, the mouse functions are explained in summary:

❑ Scroll wheel:	Scrolling of the visible area to the top/bottom
❑ Scroll wheel + <Shift>:	Scrolling of the visible area to the left/right
❑ Scroll wheel + <Ctrl>:	Zoom out/zoom in

Enlarge page to the bottom

The program page has a default width of 2500 pixels and a default height of 10000 pixels. The visible segment depends on the screen resolution.

If the page size is not sufficient, enlarge the page to the bottom.

Procedure

1. Open the context menu by clicking with the right mouse button on a free area in the programming field.

2. Select "Extend page (vertical)" in the context menu.

Result

From the current mouse cursor position, a range of 800 pixel lines is added.

Annotation

Important note

Please ensure that there are no function units in the horizontal line of the mouse pointer. Connecting lines that cross this intended horizontal line are extended.

Note

You cannot remove an empty page within the task using a function. An empty area at the lower end of the task is removed automatically when loading a project.

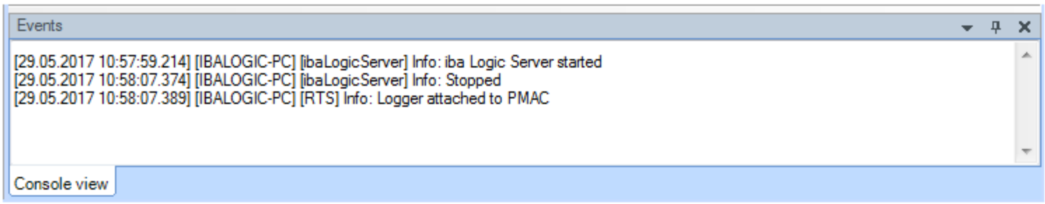
Tip

An empty space within the tasks can be achieved by reducing the zoom-level and shifting the objects jointly.

6.2.6 Event window

The "Events" window below the "Program Designer" window documents the program actions and possible error messages.

The view is used to display all events sorted chronologically by their occurrence with the corresponding explanation as simple text.



```
[29.05.2017 10:57:59.214] [IBALOGIC-PC] [ibaLogicServer] Info: iba Logic Server started
[29.05.2017 10:58:07.374] [IBALOGIC-PC] [ibaLogicServer] Info: Stopped
[29.05.2017 10:58:07.389] [IBALOGIC-PC] [RTS] Info: Logger attached to PMAC
```

Figure 35: Event window

6.3 Workspace

You can use a workspace to save programs and projects in a sorted manner.

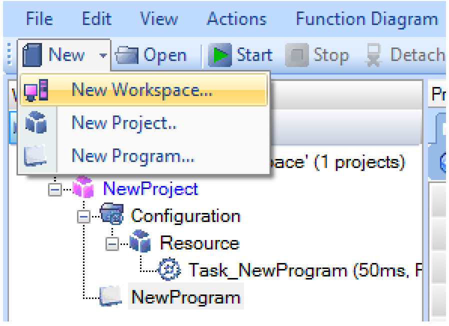
6.3.1 Create workspace

Prerequisite

You have not opened any workspace.
You can close open workspaces via "File - Close Workspace".

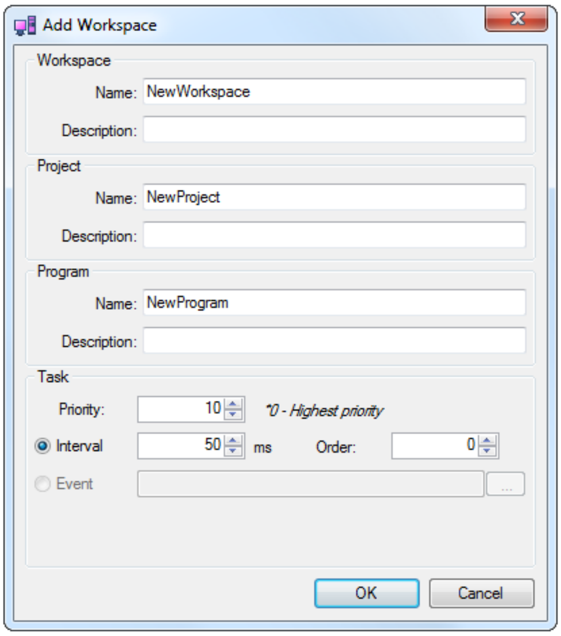
Procedure

1. Click on the arrow of the <New> icon in the toolbar.
2. Select the "New Workspace" menu entry from the list.



The "Add Workspace" dialog box is displayed.

3. Assign a name and a description for the workspace.
ibaLogic automatically creates a new project and a task. Assign meaningful descriptions that also conform to the IEC standard.
If required, also define the interval time and its priority for the task.
At that time, an event task cannot be created, because the hardware is not yet known to the system and therefore a trigger signal is not available.


Add Workspace

Workspace
Name: NewWorkspace
Description:

Project
Name: NewProject
Description:

Program
Name: NewProgram
Description:

Task
Priority: 10 '0 - Highest priority
● Interval 50 ms Order: 0
○ Event

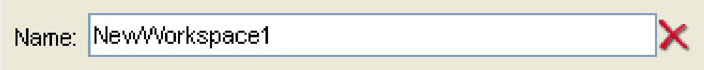
OK Cancel

Note

You can modify all settings subsequently.

You can modify the settings via the respective context menu for "Properties".

A name that has already been assigned is displayed with an "X" at the end of the input field. If you move the mouse on the “X”, a tooltip appears showing a more detailed error message.



Name: NewWorkspace1

You can fill the fields for the description with any comments. These comments are displayed above the object as a tooltip.

Notes

The names must comply with the IEC standard. For further information, please refer to "*Naming conventions*, page 299".

Tip

You can set the default value for the names under “Tools – Options – Editors – Workspaces".

6.3.2 Open workspace

Procedure

1. Press the <Open> button. The "Open Workspace" window is displayed.
2. Select the desired workspace and click <OK>.

6.3.3 Close Opened Workspace

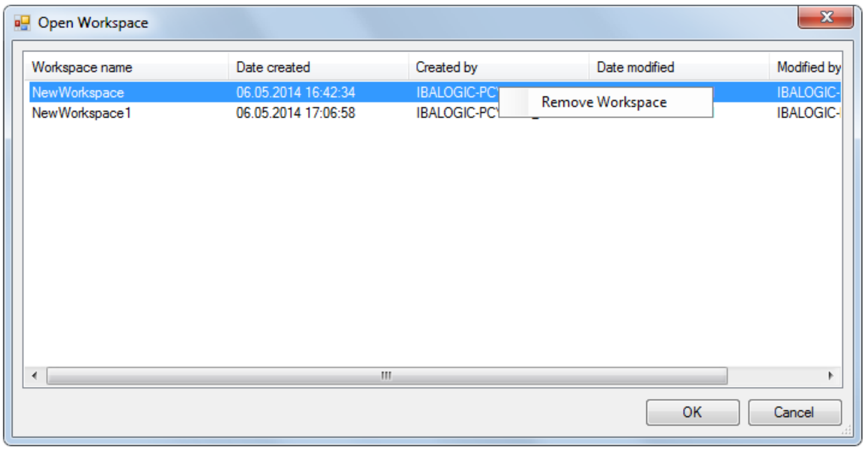
Select the workspace in the "File – Close Workspace" menu.

6.3.4 Remove Workspace from the Database

Procedure

1. Click on the <Open> button. The "Open Workspace" window is displayed.

2. Mark the workspace that needs to be removed.
3. Open the "Remove Workspace" context menu by clicking on the right mouse button.



Workspace name	Date created	Created by	Date modified	Modified by
NewWorkspace	06.05.2014 16:42:34	IBALOGIC-PC\		IBALOGIC-
NewWorkspace1	06.05.2014 17:06:58	IBALOGIC-PC\		IBALOGIC-

4. Click on "Remove Workspace". The "Remove Workspace" dialog box is displayed.
5. Confirm with <Yes>.
6. Confirm the procedure with <OK>. The dialog box is closed.

6.4 Workspace Projects

ibaLogic automatically creates a project while creating a new workspace. You can create more than one project within one workspace.

Prerequisite

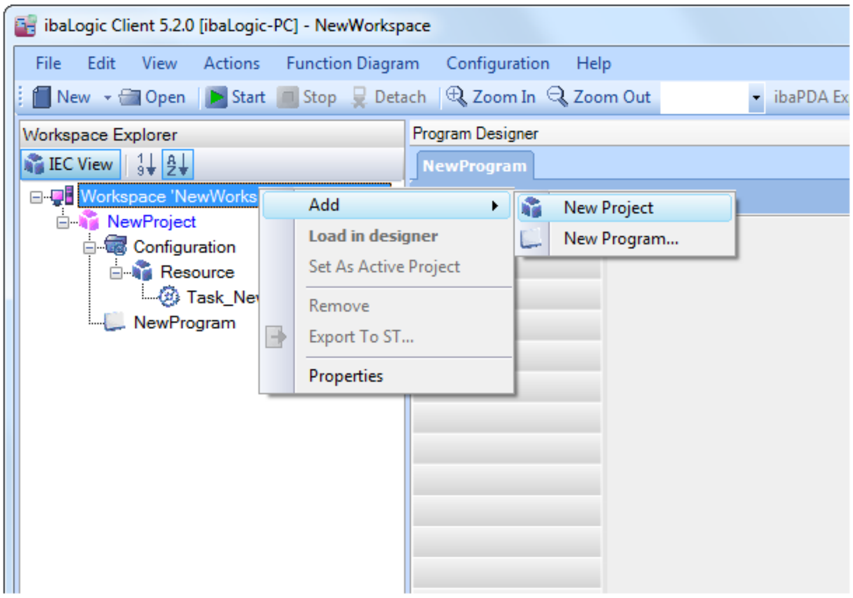
❑ You have started the ibaLogic Server and the ibaLogic Client.

❑ You have created at least one workspace.

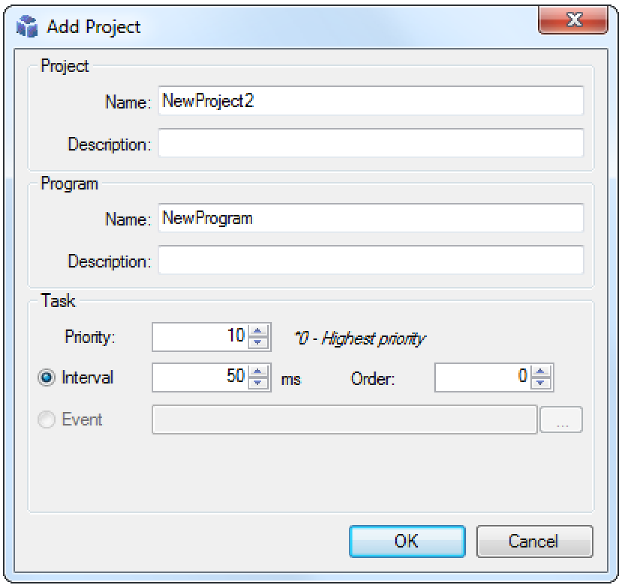
6.4.1 Create Project

Procedure

1. Click with the right mouse button on the workspace name to which the project needs to be added in the Workspace Explorer.
2. Select "Add - New Project" in the context menu.



The "Add Project" dialog box is displayed.
The dialog box that opens is the project-specific section from the workspace dialog. Assign a name to the project and the program. Assign meaningful names that conform to the IEC standard. If required, set up the interval time and priority for the task.


Add Project

Project

Name: NewProject2

Description:

Program

Name: NewProgram

Description:

Task

Priority: 10 *0 - Highest priority*

● Interval 50 ms Order: 0

○ Event

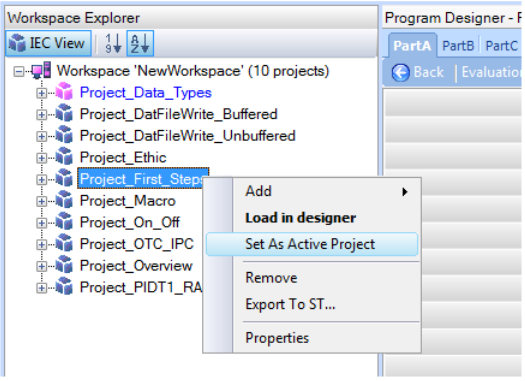
6.4.2 Set project as active

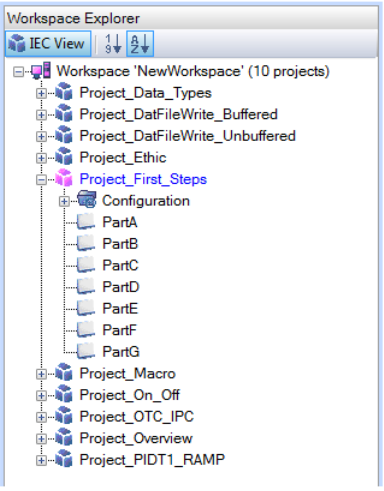
Note

Only one of several projects can be active within a workspace.

Procedure

1. Open the context menu of the project to be activated by clicking on the right mouse button.
2. Select "Set As Active Project".





Result

The name of the active project is displayed in the Workspace Explorer in "**Blue**" and the associated icon changes its color from blue to pink.

Notes

By setting a project as active, the programs of the project are not automatically loaded in the design area, but need to be selected manually as required.

The buttons in the toolbar and in the Workspace Explorer are valid only for the active project. They include:

- ❑ <Start>
- ❑ <Stop>
- ❑ <Detach>
- ❑ <Current platform>
- ❑ <I/O configurator>

It is possible to switch between different online projects, see chapter *Switching between several online projects*, page 74.

6.4.3 Load Project in the Program Designer

Regardless of whether a project is active or not, you can load programs in the design area.

Procedure

1. Mark the project that you would like to load in the designer.
2. Select "Load in designer" in the context menu.

Result

The programs loaded are displayed as tabs at the upper margin of the design area.

6.4.4 Edit project properties

You can modify the name and description field in the project properties. You can enter supplementary information about the project in the "Description" text field. The description also appears in the tooltip of the project.

Procedure

1. Right-click on the project whose project properties you would like to edit.
2. Select "Properties" from the context menu.
   The "Edit workspace" window is displayed.
3. Edit the properties of the project.
4. Click on <OK>.

6.4.5 Remove Project

The project selected is removed from the workspace and from the database at the same time.

Procedure

1. Click with the right mouse button on the project that you would like to remove from the workspace.

2. Select "Remove" in the context menu. The "Remove Project" dialog box is displayed.
3. If you are sure that you would like to remove this project from the workspace, click on <OK>.
4. If the project has been deleted, the "Project removed" dialog is displayed for confirmation.
5. Finally, click on <OK>.

6.4.6 Exporting/importing projects

The export/import functions exchange complete projects between different databases.

Exporting

1. Right-click on the corresponding project.
2. Select "Export To ST" in the context menu. The "Export" dialog box is displayed.
3. Open the file browser, ... select a directory, enter a file name and click on “Save”.
4. Enable the option “Generate with additional graphical information” and click on OK.

Note

When exporting no untyped (ANY) input/output connectors should exist, since otherwise, only an export is possible, but no import. This happens if unconnected function units etc. are in the program. The export also shows the corresponding message.

Importing

1. Switch off the calculation because the import is only available in the offline mode.
2. Select “File – Import – Structured Text" under main menu.
   The “Structured text import” dialog is displayed.
3. Open the file browser and select directory and file (with file extension *.il5) and click “Open".
4. Select the “Import Function Unit definitions as new ones” option and click on OK.

During the import, the names of the function units and data types to be imported are compared to already existing names. If names are already assigned, the definitions are overwritten (TRUE) in relation to the option: “Import Function Unit definitions as new ones” or a new definition is created with name+index (FALSE).

The imported program is newly created together with the task. If the project name is already assigned, they are created with name+index.

During the import of a project, all programs and tasks in the project are newly assigned, probably with “name+index”.

Note

Export/import description for **function units**, refer to “Exporting Function Units" or "*Importing Function Units*, page 92".

Note

When importing in addition to already existing programs, the same priority and order might occur.
This must be changed manually. The order has to be unique.

Note

Importing an ibaLogic-V4 export with a DatFileWrite function block does not work, since the DFW has changed.

In order to get this import in ibaLogic-V5, you need to execute the import under V4 first before completely restoring this database in V5.
The “dummy” DFW resulting from this can then be replaced by a new DFW; after that, an import file valid for V5 can be created via export.

6.4.7 Switching between several online projects

Different projects can exist in one workspace, all running on different target hardware. However, only the active project can be viewed online.

This project can then be started and stopped again. If you want to switch to one of the other projects online, you can use the "Detach" function in the menu bar. The current project is detached. Then you can select another project as active one and press "Start" for this project. (It is also advisable to select a program from this project in the Designer beforehand.) The "Start" command recognizes that this project is running on its target system and offers a "Connect" option. Then the other ongoing project is connected.

You can switch back accordingly.

6.5 Tasks/Programs

ibaLogic creates a program automatically while creating a new project. You can add other programs to a given project.

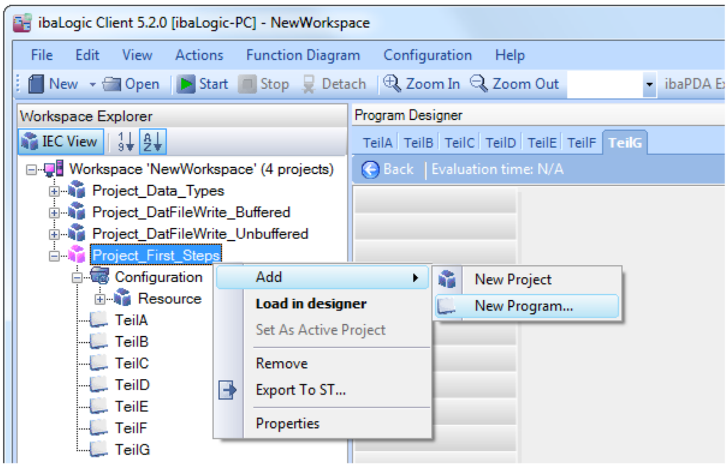
Notes

Please keep in mind that you cannot delete the last task. One task must always be present for each project.

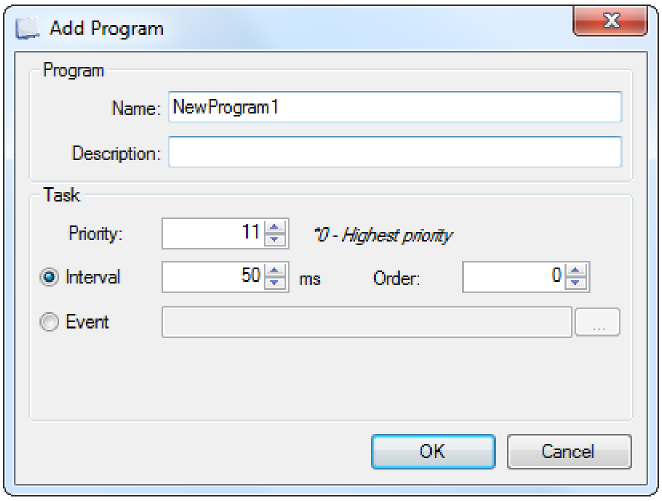
6.5.1 Creating tasks/programs

Procedure

1. Click with the right mouse button on the project in which you would like to create a new program.
2. Select “Add - New Program...” in the context menu.


ibaLogic Client 5.2.0 [ibaLogic-PC] - NewWorkspace
File Edit View Actions Function Diagram Configuration Help
New Open Start Stop Detach Zoom In Zoom Out ibaPDA E
Workspace Explorer
IEC View
Workspace 'NewWorkspace' (4 projects)
Project_Data_Types
Project_DatFileWrite_Buffered
Project_DatFileWrite_Unbuffered
Project_First_Steps
Configuration
Resource
TeilA
TeilB
TeilC
TeilD
TeilE
TeilF
TeilG
Add
Load in designer
Set As Active Project
Remove
Export To ST...
Properties
New Project
New Program...
Program Designer
TeilA TeilB TeilC TeilD TeilE TeilF TeilG
Back
Evaluation time: N/A

The “Add Program" dialog is displayed.



Name: NewProgram1

Description:

Priority: 11 '0 - Highest priority

● Interval 50 ms Order: 0

○ Event

3. Assign a name and a description for the program. Create an associated task at the same time. Assign meaningful names that conform to the IEC standard. For further information, please see the "*Naming conventions*, page 299“ definition.

4. Click on <OK> to add the new program.

Exactly one task is assigned to each program. You can choose from the following task parameters:

- ❑ Interval or event
- ❑ Priority
- ❑ Order

Interval

The program appertaining to the task is restarted exactly after the specified time interval.

If, under extreme circumstances, the evaluation time (or the evaluation of all programs) is longer than the interval time specified, it means that the system is overloaded. How to proceed is explained in „*Time response*, page 248".

The default value is 50 ms. The smallest interval possible is 1 ms, but the time interval cannot be less than the base time set in the I/O configurator.

Notes

You can change the default values for the program name and the time interval under “Tools – Options – Editors – Workspace".

Priority

Each task is assigned a priority. "0" means highest priority.

Please note that higher priority tasks interrupt lower priority ones. For more information, please refer to “*Time response*, page 248".

Notes

You can display the tasks either in alphabetic order or in the order of their priorities in the workspace.
You can use the icons at the upper margin of the navigation section for this purpose.
1 9 ↓ A Z ↓

Order

In case of interval tasks having the same interval and priority, the order specifies the execution order.

6.5.2 Open Tasks/Program

Double click on the program name in the Workspace Explorer.

6.5.3 Change Task / Program Properties

Change the properties of existing tasks / programs.

Procedure

1. Click with the right mouse button on the corresponding task or program.
2. Select "Properties" in the context menu. The "Edit" window is displayed.

3. Change the properties.
4. Finally, click on <OK>.

6.5.4 Remove Task / Program

Remove a program / task from a workspace.

Procedure

1. Click with the right mouse button on the corresponding task or program.
2. Select "Remove" in the context menu. The "Confirm" dialog box is displayed.
3. If you wish to remove the program, click on <Yes>.

Notes

Please keep in mind that you cannot delete the last task. One task must always be present for each project.

6.5.5 Import/export programs

The export/import function exchanges complete programs between projects of different databases.

Exporting

1. Right-click on the respective program.
2. Select "Export To ST" in the context menu. The "Export" dialog box is displayed.
3. Open the file browser, ... select a directory, enter the data name and click on “Save”.
4. Enable the option “Generate with additional graphical information” and click on OK.

Importing

1. Switch off the calculation because the import is only available in the offline mode.
2. Select “File – Import – Structured Text" under main menu.
   The “Structured text import” dialog is displayed.
3. Call up the file browser and select directory and file (with file extension *.il5) and click “Open".
4. Select the “Import Function Unit definitions as new ones” option and click on OK.

During the import, the names of the function units and data types to be imported are compared to already existing names. If names are already assigned, the definitions are overwritten (TRUE) in relation to the option: “Import Function Unit definitions as new ones” or a new definition is created with name+index (FALSE).

The imported program is newly created together with the task. If the program name is already assigned, they are created with name+index.

During the import of a project, all programs and tasks in the project are newly created, probably with “name+index”.

Note

Export/import description for **function units**, refer to “Exporting Function Units" or "*Importing Function Units*, page 92".

Note

When importing in addition to already existing programs, the same priority and order might occur.
This must be changed manually. The order has to be unique.

6.6 Configure Inputs and Outputs

Inputs and outputs are are routed signals to peripheral devices.

You work with "virtual" signals in the programming environment. You must assign these using an allocation process to hardware signals that are actually present.

You can do this assignment from the viewpoint of the program or that of the hardware, for each signal separately or for groups of signals.

Important note

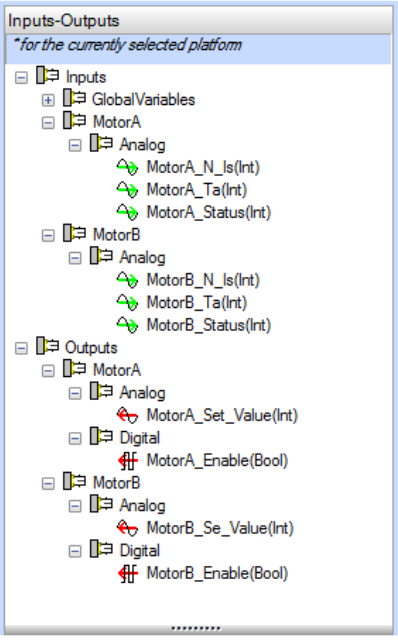
For more information, please refer to "IO Configuration".

The inputs and outputs of the hardware are arranged in a tree structure within the navigation section of the I/O configurator.

The hierarchy levels are:

Direction → group → signal subdivision → signals

- ❑ Direction:
  "Inputs" or "Outputs"
- ❑ Group:
  group name either created manually or taken over from the module name of the signal assignment.
- ❑ Signal subdivision:
  The signals are divided in analog and digital signals.
- ❑ Signals:
  - signal name either created manually or taken over from the name of the signal assignment.
  - behind that the data type in brackets,
  - then the hardware signal name to the right of the arrow (->)



```
Inputs-Outputs
*for the currently selected platform
⊟ Inputs
  ⊞ GlobalVariables
  ⊟ MotorA
    ⊟ Analog
        MotorA_N_Is(Int)
        MotorA_Ta(Int)
        MotorA_Status(Int)
  ⊟ MotorB
    ⊟ Analog
        MotorB_N_Is(Int)
        MotorB_Ta(Int)
        MotorB_Status(Int)
⊟ Outputs
  ⊟ MotorA
    ⊟ Analog
        MotorA_Set_Value(Int)
    ⊟ Digital
        MotorA_Enable(Bool)
  ⊟ MotorB
    ⊟ Analog
        MotorB_Se_Value(Int)
    ⊟ Digital
        MotorB_Enable(Bool)
```

Figure 36: Inputs - Outputs

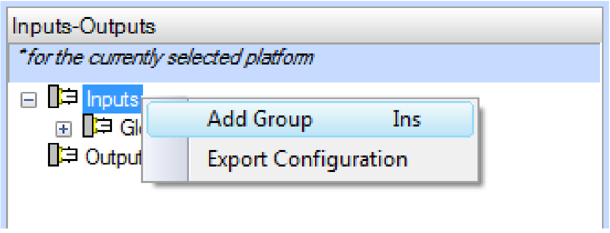
6.6.1 Create Signals

Signals are assigned to groups. This facilitates meaningful structuring.

6.6.1.1 Define Group

Procedure

1. Click with the right mouse button on "Inputs - Outputs" and select the menu item "Add group".



2. You determine the name for the new group in the dialog “Setup group name”.

Result

The new group is created under “Inputs" as well as "Outputs".
Groups which are not necessary can be deleted by means of the context menu and/or the <Delete> key.

Example

Groups:	Motor A, Motor B
Input signals:	Actual speed, motor temperature
Output signals:	Set-point value, parameter

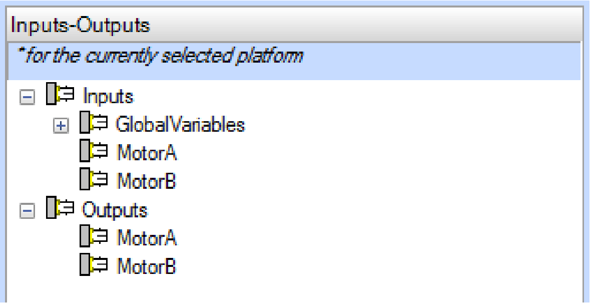
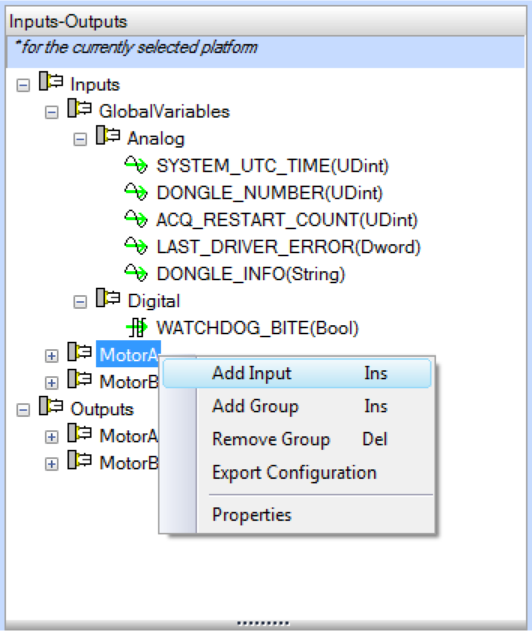


Figure 37: View "Inputs – Outputs"

6.6.2 Define Signals

Procedure

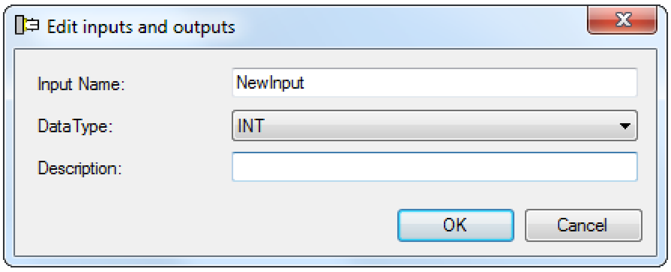
1. Click with the right mouse button on an input or output group.
2. Select "Add input" or "Add output" in the context menu.
   The "Edit inputs and outputs" dialog will be opened.
3. Assign a signal name, data type and, a description, if any. Assign meaningful names that conform to the IEC standard.



Inputs-Outputs

*for the currently selected platform

- Inputs
  - GlobalVariables
    - Analog
      - SYSTEM_UTC_TIME(UDint)
      - DONGLE_NUMBER(UDint)
      - ACQ_RESTART_COUNT(UDint)
      - LAST_DRIVER_ERROR(Dword)
      - DONGLE_INFO(String)
    - Digital
      - WATCHDOG_BITE(Bool)
  - MotorA
  - MotorB
- Outputs
  - MotorA
  - MotorB



Edit inputs and outputs

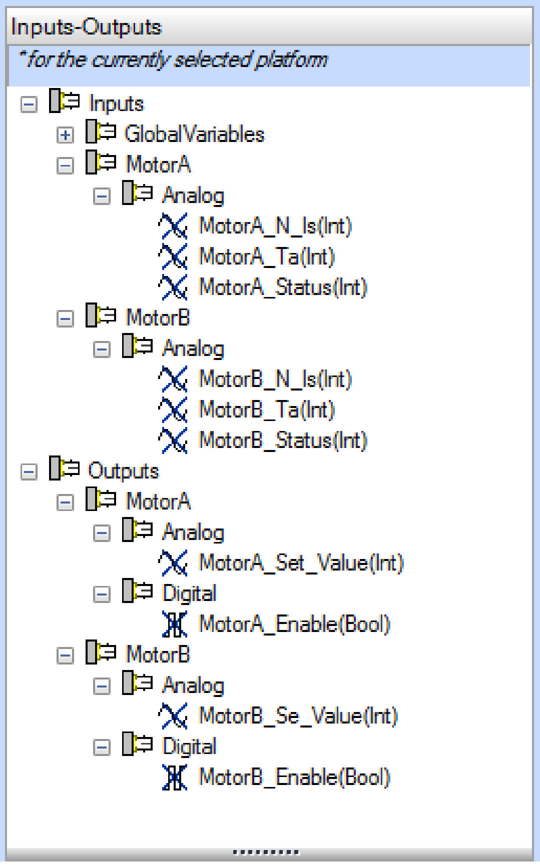
Input Name: NewInput

Data Type: INT

Description:

Please note that the data type is provided by the peripheral device. If the analog value from an ibaPADU-8 is used, for example, it is always an integer value.

In our example, the following scenario would emerge:



Inputs-Outputs

for the currently selected platform

- Inputs
  - GlobalVariables
  - MotorA
    - Analog
      - MotorA_N_Is(Int)
      - MotorA_Ta(Int)
      - MotorA_Status(Int)
  - MotorB
    - Analog
      - MotorB_N_Is(Int)
      - MotorB_Ta(Int)
      - MotorB_Status(Int)
- Outputs
  - MotorA
    - Analog
      - MotorA_Set_Value(Int)
    - Digital
      - MotorA_Enable(Bool)
  - MotorB
    - Analog
      - MotorB_Se_Value(Int)
    - Digital
      - MotorB_Enable(Bool)

The signals have not yet been assigned to any hardware.

6.6.3 Edit Existing Signals

Procedure

1. Double click on a signal that has been defined.
   The editing dialog box opens.

2. Assign a signal name, data type and a description, if any. Assign meaningful names that conform to the IEC standard.

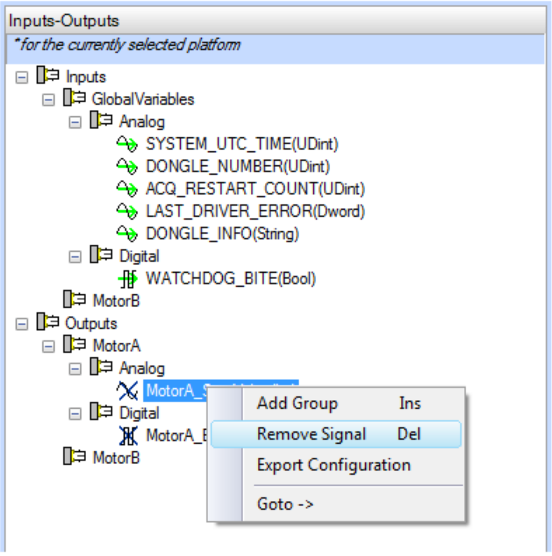
3. Click on <OK> to accept the modifications.

6.6.4 Remove signals

Procedure

1. Click with the right mouse button on a signal that has been defined.
   The context menu opens.

2. Select "Remove Signal" in the context menu.



Note

You can edit and remove inputs and/or outputs only if you do not yet use them in the program, i. e. they are not yet visible in the input and/or output bar.

Note

The assignment of the signals defined here to the hardware available is described in sction "*Signal assignment*, page 183".

Important note

The interface cards of third party manufacturers sometimes do not provide individual signals with the elementary data types, REAL and INT. A data structure as an input / output is generated here for the Profibus master card SST, and it depends on the configuration of the slave (GSD file). You must also define this structure in ibaLogic to be able to use the signals included there.

An example of the connection of the Profibus master card is documented and included in the DVD supplied.

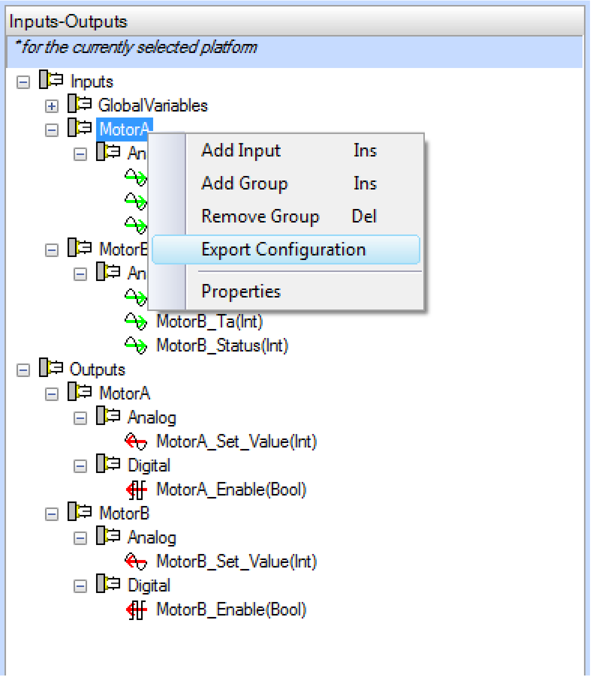
6.6.5 Export / Import Signals

The virtual signal names are often already provided in external documents. Use the export and import functions in order to use them.

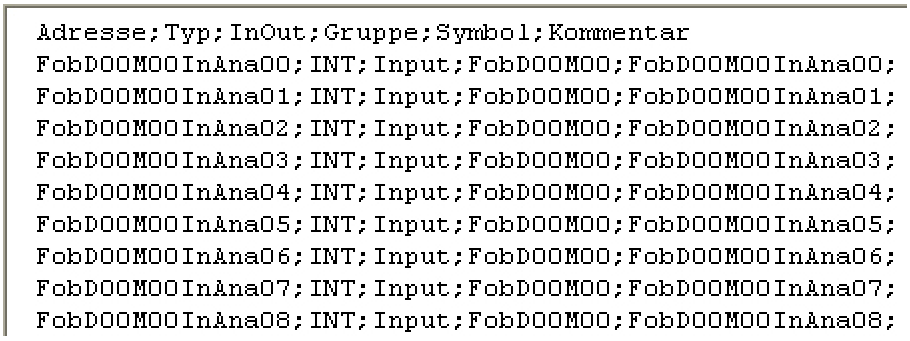
Exporting the entire I/O configuration

Procedure

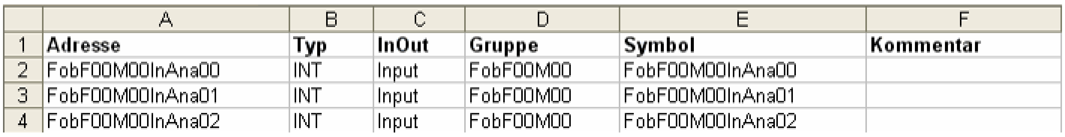
1. Open the context menu of a group or of a signal and select the menu option "Export configuration".



2. Open the export, for example, in a spreadsheet program.
3. Enter the virtual group designation and signal name in the group and icon columns or modify the existing names.

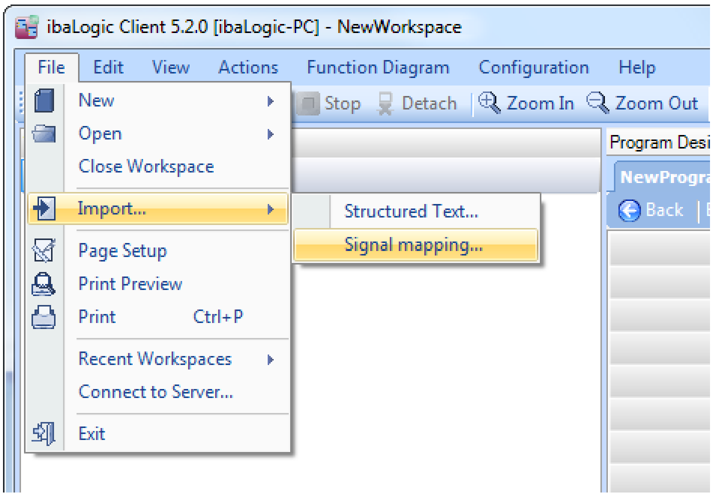


```
Adresse;Typ;InOut;Gruppe;Symbol;Kommentar
FobD00M00InAna00;INT;Input;FobD00M00;FobD00M00InAna00;
FobD00M00InAna01;INT;Input;FobD00M00;FobD00M00InAna01;
FobD00M00InAna02;INT;Input;FobD00M00;FobD00M00InAna02;
FobD00M00InAna03;INT;Input;FobD00M00;FobD00M00InAna03;
FobD00M00InAna04;INT;Input;FobD00M00;FobD00M00InAna04;
FobD00M00InAna05;INT;Input;FobD00M00;FobD00M00InAna05;
FobD00M00InAna06;INT;Input;FobD00M00;FobD00M00InAna06;
FobD00M00InAna07;INT;Input;FobD00M00;FobD00M00InAna07;
FobD00M00InAna08;INT;Input;FobD00M00;FobD00M00InAna08;
```



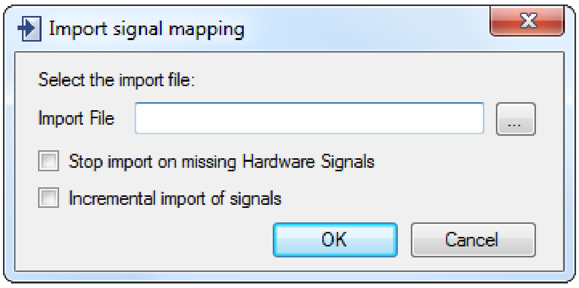
	A	B	C	D	E	F
1	**Adresse**	**Typ**	**InOut**	**Gruppe**	**Symbol**	**Kommentar**
2	FobF00M00InAna00	INT	Input	FobF00M00	FobF00M00InAna00	
3	FobF00M00InAna01	INT	Input	FobF00M00	FobF00M00InAna01	
4	FobF00M00InAna02	INT	Input	FobF00M00	FobF00M00InAna02	

4. Import the signal assignment by using the menu item "File - Import... - Signal mapping...".



The "Import signal mapping" dialog is displayed.

5. Specify the file name with the target path.


Import signal mapping
Select the import file:
Import File
...
□ Stop import on missing Hardware Signals
□ Incremental import of signals
OK
Cancel

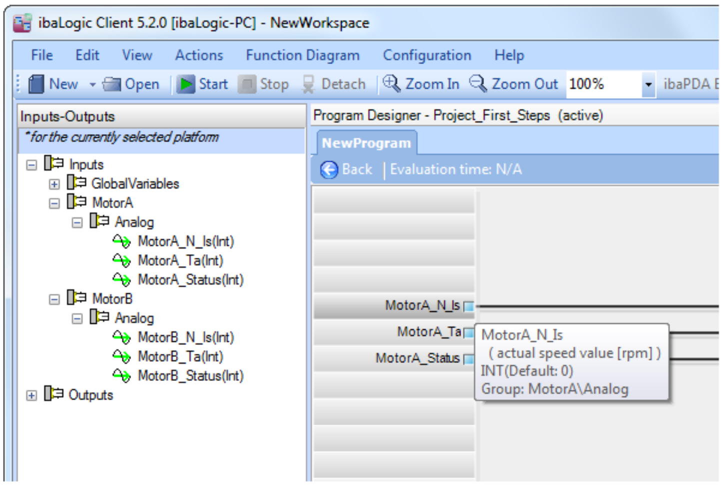
6. Click on <OK> to start the import.

6.6.6 Using Signals in the Program

To be able to use signals in a project program these must be dragged to the input or output sidebar.

Processing

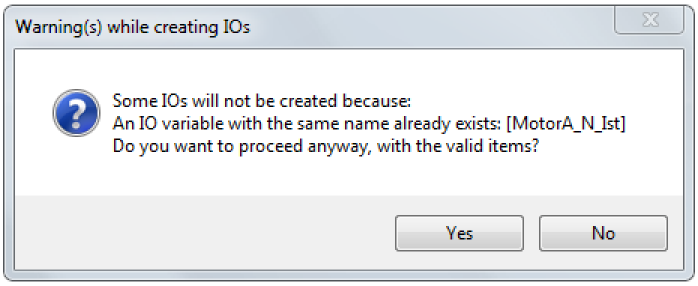
➲ Drag the desired individual signal or the entire group to the sidebar of the input or output.



Example

The signal "MotorA_N_Ist" has been dragged to the input sidebar.
When you point with the mouse on the connector symbol of the signal, a tooltip is displayed with information pertaining to this signal.

If you drag an entire group to the sidebar, all individual signals are placed there. If certain signals of the group are already placed in the sidebar, the following warning message is output.



Warning(s) while creating IOs

Some IOs will not be created because:
An IO variable with the same name already exists: [MotorA_N_Ist]
Do you want to proceed anyway, with the valid items?

Note

For further information please see "IO Configuration".

6.6.7 Remove Signals in the Program

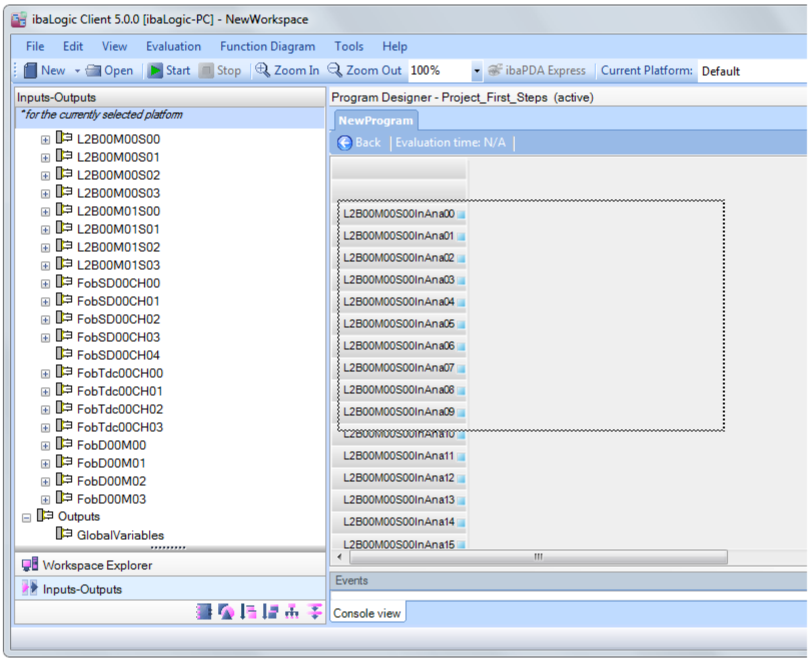
Procedure

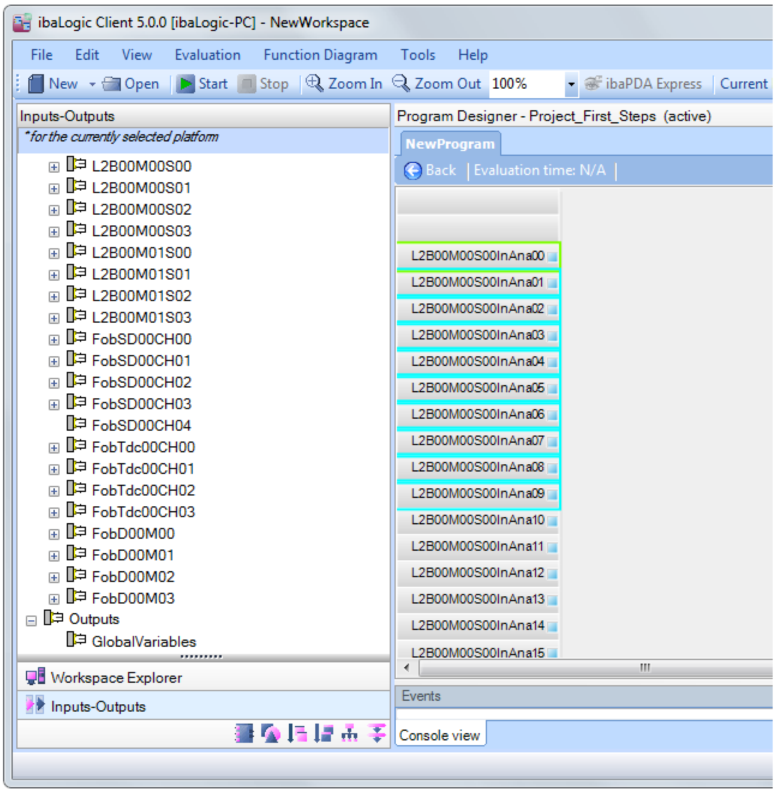
1. Mark the signal in the input or output bar.
2. Press the <Del> function key.

Remarks

You can make multiple selections by clicking the left mouse button and pressing the <Shift> key simultaneously or by dragging a rectangle using the left mouse button

(Lasso) over the signals concerned. Please note that the rectangle must completely enclose the signals.





7 Program Creation

7.1 Function Units

In ibaLogic, a large number of function units are provided in a global library.

In addition to the function units defined in accordance with the IEC 61131-3 standard, iba also provides its own function blocks and user blocks. These are listed in the navigation area of the "Function Units" view.

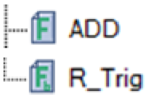
In this view a symbol gives detailed information about the type of the function unit.
The symbols mean the following:

Function Units of the IEC 61131-3 standard

iba Function Units

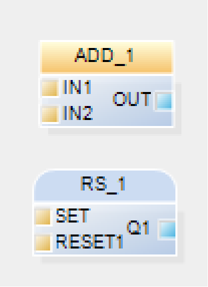
Function blocks that are available in the form of a DLL

Function blocks and macro blocks created by the user



Functions are defined non-storing: Symbol: F

Function blocks are defined storing: Symbol: F_b

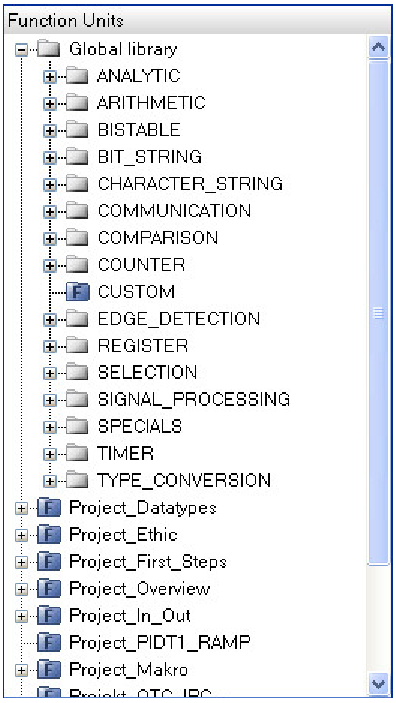


Functions have sharp edges

(storing) function blocks have rounded edges

The difference is up to the behavior of disconnecting a link. Function blocks store the last value. Functions jump to their default value.

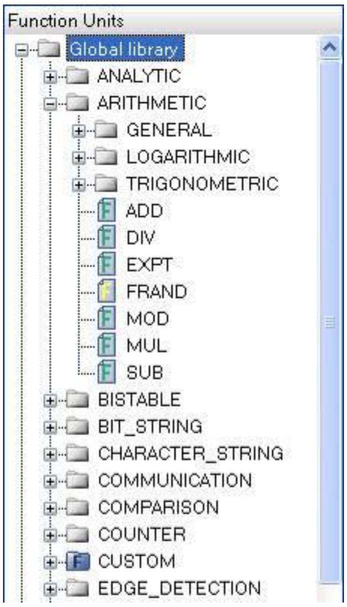
The function units are listed in groups and sub-groups that are sorted alphabetically according to the IEC 61131-3 specification.



Function Units

- Global library
  - ANALYTIC
  - ARITHMETIC
  - BISTABLE
  - BIT_STRING
  - CHARACTER_STRING
  - COMMUNICATION
  - COMPARISON
  - COUNTER
  - CUSTOM
  - EDGE_DETECTION
  - REGISTER
  - SELECTION
  - SIGNAL_PROCESSING
  - SPECIALS
  - TIMER
  - TYPE_CONVERSION
- Project_Datatypes
- Project_Ethic
- Project_First_Steps
- Project_Overview
- Project_In_Out
- Project_PIDT1_RAMP
- Project_Makro
- Projekt_OTC_IPC

Function Units "Global Library"



Function Units

- Global library
  - ANALYTIC
  - ARITHMETIC
    - GENERAL
    - LOGARITHMIC
    - TRIGONOMETRIC
    - ADD
    - DIV
    - EXPT
    - FRAND
    - MOD
    - MUL
    - SUB
  - BISTABLE
  - BIT_STRING
  - CHARACTER_STRING
  - COMMUNICATION
  - COMPARISON
  - COUNTER
  - CUSTOM
  - EDGE_DETECTION

Function Units "Arithmetic"

The "Global Library" also contains the folders “CUSTOM" and "NEW PROJECT".

CUSTOM

There, all global user blocks and function blocks being available as DLL are placed.

SPECIALS

The "Specialties" of ibaLogic are stored here. For more information, please refer to "*Specials*, page 320".

7.1.1 Using Function Units

You can use all Function Units from the global library and the project library in a given project.

If you wish to use a FB from the global library, you can use Drag & Drop to move it into the programming window.

You have no access to function units that you have defined in a project of another workspace.

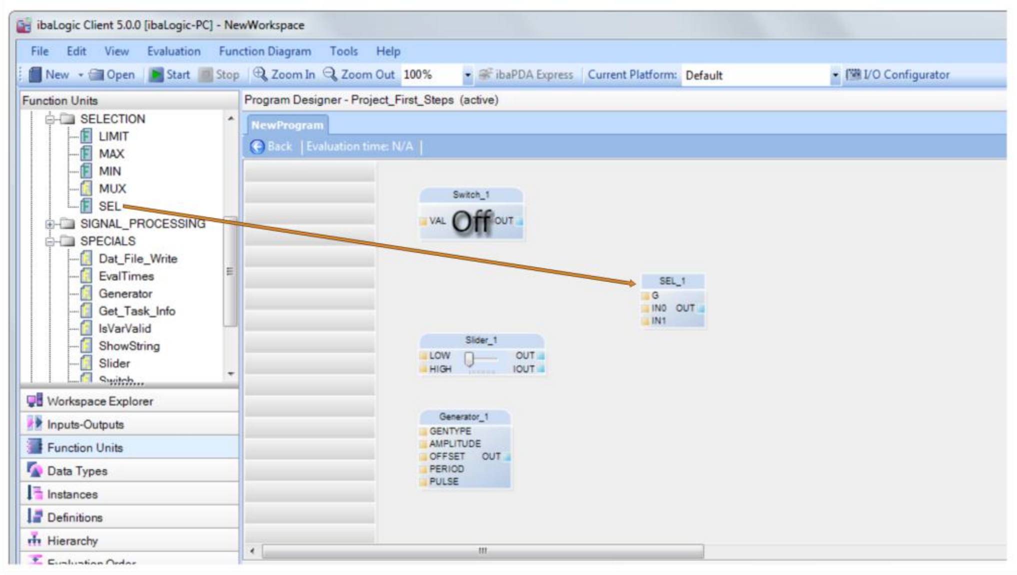
Important Note

When you modify the contents of a function unit, all instances and the definition of the block are modified.

If you would like to modify the form, i. e. inputs and outputs or internal variables, you will be asked to specify another name for the definition. In this case, a new block type is created, the old definition and its instances remain unchanged.
However, this is only the case if more than one instance is available.

Procedure

1. Select a function unit that you wish to place in the programming field.
2. Drag & Drop the block selected in the "Function Unit Overview" at any position that is free in the programming field.



7.1.2 Create Function Units

You can create a function unit in different ways:

❑ In the program

❑ Under the project

❑ In the global library

7.1.2.1 In the Program

Creating an FB or MB within a program.

Procedure

1. Click with the right mouse button at any free location in the programming window.
2. In the context menu, select "New... - New Function Block" or "New... - New Macro Block". The "Create Function Block" window is displayed.
3. Create your function unit.
4. By pressing <OK>, the function unit is created if there are no syntax errors.

7.1.2.2 Under the project

Creating an FB within a project.

Procedure

1. Open the "Function Units" view.
2. Switch to the project folder.
3. Click in the function tree with the right mouse button on the desired project.

4. In the context menu, select
"New... - New Function Block" or "New... - New Macro Block".
The "Create Function Block" window is displayed.
5. Create your function unit.
6. By pressing <OK>, the function unit is created if there are no syntax errors.

7.1.2.3 In the Global Library

Creating a FB within the global library.

Procedure

1. Click with the right mouse button in the global library on the "CUSTOM" group.
2. In the context menu, select "New... - New Function Block" or "New... - New Macro Block". The "Create Function Block" window is displayed.
3. Create your function unit.
4. By pressing <OK>, the function unit is created if there are no syntax errors.

Note
Difference between Definition - Instance

For more information, please see "*Instances*, page 52".

7.1.3 Managing Function Units

Copy a function unit to the global library

If you would like to use a function unit, which you have defined in a program or project, even in other workspaces, you must copy it to the global library.

Procedure

1. Click with the right mouse button on the FB that you would like to copy.
2. Select "Copy to the global library" in the context menu.
The function unit is copied into the "CUSTOM" group.

Copy all function units to the global library

Function unit and macros can also be copied to the global library together.

Procedure

1. Click with the right mouse button on the project folder that you would like to copy.
2. Select "Copy to the global library" in the context menu.

Note

A function unit that has been copied into the global library has the properties of the time point when it was copied. If the FB is modified in the project path, these two function units are different.

Note

You cannot copy or move FBs from the global library to a project catalog. When using a function unit from the global library, it is automatically copied into the project catalog.

You can also use function units that have been defined in another project of the same workspace. As a result, these are created automatically under the project function blocks.

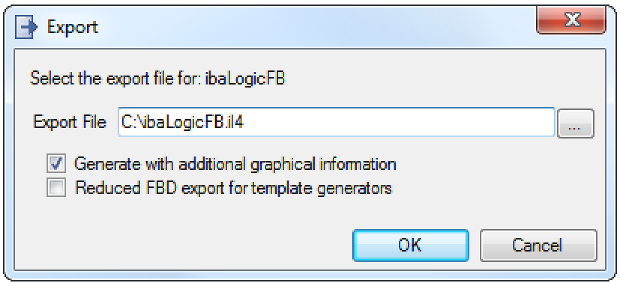
For more information, please refer to "*Using Function Units*, page 89".

7.1.4 Export Function Units

If you would like to use a function unit, which you have defined in a database under the global group "CUSTOM" or in a project, also in another database or another program tool, then you can export this into a text file.

Procedure

1. Click with the right mouse button on the function unit definition under "CUSTOM" or under the project.
2. Select "Export To ST" in the context menu. The "Export" dialog box is displayed.
3. Specify the target folder and file name.



Export

Select the export file for: ibaLogicFB

Export File C:\ibaLogicFB.il4

☑ Generate with additional graphical information

☐ Reduced FBD export for template generators

Note

If you wish to use the function unit exported in an ibaLogic-V5 environment, you must export the function unit with additional graphical information.

If you would like to use the function unit in another system, you should export it without additional graphical information.

Important Note

Since the implementation of the IEC standard is manufacturer-dependent, before you link the function unit, you must check in the external system whether it contains deviations or non-conformances with respect to the IEC standard.

7.1.5 Importing Function Units

Prerequisite

❑ ibaLogic is not running in the online mode.

❑ You have a file (function unit) that can be imported.

Procedure

1. Select the "File – Import – Structured Text" menu.
   The "Structured Text Import" window is displayed.
2. Select the file to be imported in the browser.
3. Finally, confirm with <OK>.

Check box	Explanation
Import Function Unit definitions as new ones	The selection imports the function unit as a new function unit definition.

7.1.6 Removing Function Units

When you remove a FB from the program, you are deleting only an instance of it.

Procedure

1. Select a FB from the library of your project.
2. Open the context menu.
3. Select <Remove>.

Remark

A dialog box is displayed if you delete the only instance that exists. You can specify in this dialog box whether you would also like to delete the definition. Finally, confirm with <OK>. The definition is deleted.

The function or macro block is also removed from the project and is thus no longer available.

A global function unit definition in the "CUSTOM" group is not deleted by this method. You can do this by clicking with the right mouse button on the function unit in the "CUSTOM" group and selecting "Remove" in the context menu or pressing the <Del> key.

7.1.7 Set default values and online function unit values

Default values that can be specified in the elements of programming are subject to specific rules:

1. Default values are taken over only at program start, i.e. when setting ibaLogic online
2. When disconnecting an online connection from a function unit, the resulting value at the now open connector depends on the block type (function or function block)

In the display, functions have sharp edges and function blocks have rounded edges. Functions are non-storing, therefore, an open connector will jump to its default value field.

Function blocks are storing and will therefore keep the last value that was present before the last disconnecting.

Adapting a default value online will only work with the field ACTUAL VALUE which can be seen only within an FB. If this value should be kept when restarting ibaLogic, it also has to be entered in the default value.

A change of the actual value applies as soon as the input field has been left. Thereby, it is also possible to e.g. let an FB open and observe the impact by changing the values. If the actual value is calculated in the program, it will be overwritten in clocktime immediately.

“Fixing” the value is not possible (unless it is written directly into the ST code).

When changing a default value in an instance of a function block it will apply only for this instance.

When changing a default value in a definition it will be entered in all instances without an individual default value immediately. The individual default value shall not be overwritten.

7.2 Standard Function Blocks

You will find a tabular overview of all functions and function blocks that are available in ibaLogic-V5 in "*Standard Function Units*, page 301".

7.3 Complex Function Blocks

7.3.1 DAT_FILE_WRITE (DFW Function Block)

ibaLogic integrates the function which allows you to save existing analog and digital signals in .dat files at run time.
The .dat files have the iba data format and hence, they can be opened, analyzed and processed further with the iba tools, ibaDatCoordinator and ibaAnalyzer (e. g. extraction into a database).

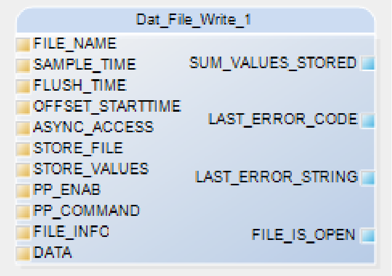


Figure 38: DAT_FILE_WRITE block

You can save either individual values or arrays of buffered data. Data of type INTEGER, REAL and BOOL are permitted. Saving additional information, such as technostring, is supported. For further information, please refer to *Buffered Mode*, page 193.

Note

Writing to .dat files and the number of channels require a license. Without a valid dongle, FILE_IS_SIGNED remains "FALSE" for recording data and hence, the files generated cannot be evaluated using ibaAnalyzer.

Note

An example of the configuration is given in "*DAT_FILE_WRITE Sample Project*, page 288".

7.3.1.1 Edit DFW Function Block

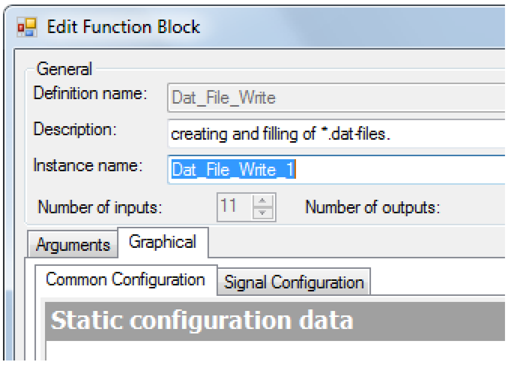
After you have dragged the DFW function block and dropped it in the program window, you can open it by double clicking on it.

The window is divided in the following tabs:

- "Arguments"
- "Graphical"

The "Graphical" tab contains the sub-tabs:

- "Common Configuration"
- "Signal Configuration"



Edit Function Block

General

Definition name: Dat_File_Write

Description: creating and filling of *.dat-files.

Instance name: Dat_File_Write_1

Number of inputs: 11 Number of outputs:

Figure 39: DAT_FILE_WRITE configurator

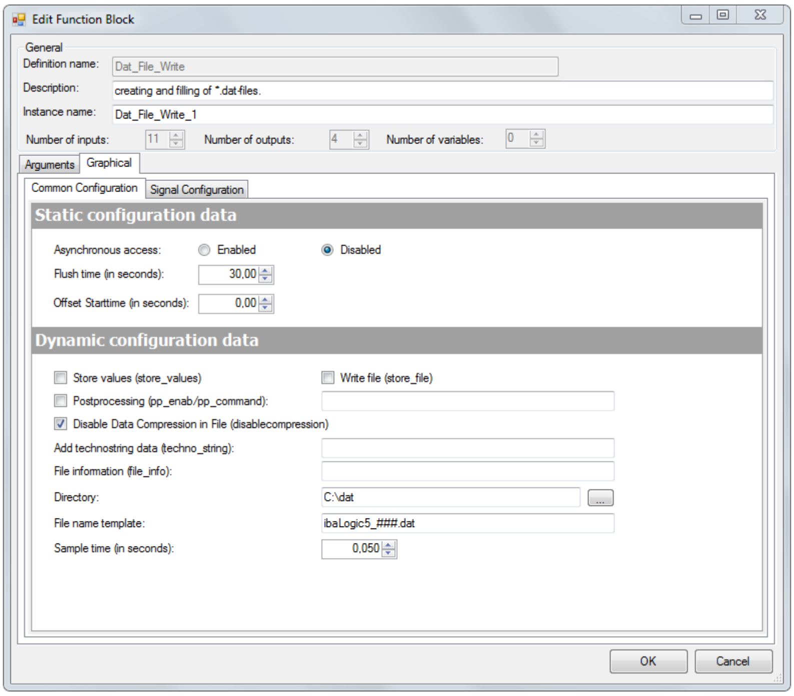
"Arguments" tab

The "Arguments" tab displays all inputs, outputs and the associated variables and data types in a tabular view. This view is also used as an overview and to display the current values in the online mode. Do not configure any settings in this view, but switch to the "Graphical" tab. Your attention shall be drawn to any exceptions for individual properties.

Important Note

When you link an input connector, the default values and those configured in the function block are overwritten. After terminating the connection, the last value is retained.

7.3.1.2 "Common Configuration" sub-tab



Edit Function Block

General

Definition name: Dat_File_Write

Description: creating and filling of *.dat-files.

Instance name: Dat_File_Write_1

Number of inputs: 11 Number of outputs: 4 Number of variables: 0

Static configuration data

Asynchronous access: ○ Enabled ● Disabled

Flush time (in seconds): 30,00

Offset Starttime (in seconds): 0,00

Dynamic configuration data

□ Store values (store_values) □ Write file (store_file)

□ Postprocessing (pp_enab/pp_command):

☑ Disable Data Compression in File (disablecompression)

Add technostring data (techno_string):

File information (file_info):

Directory: C:\dat

File name template: ibaLogic5_###.dat

Sample time (in seconds): 0,050

Figure 40: "Common Configuration" sub-tab

❑ Asynchronous access

Disabled:

The data to be saved is buffered internally. Data is written to the hard disk when the buffer is full. When the file recording is complete, you can analyze the .dat file subsequently (using ibaAnalyzer).

Enabled:
The internal buffer is written cyclically to the hard disk after the configured storage cycle time. Hence, you can start analyzing while data is still being recorded. The smaller the value of the storage cycle is, the earlier data is available in the ibaAnalyzer (if you have configured "automatic reload" there). The consequence is a lower degree of compression and a higher load on the computer or a network.

❑ Offset Starttime

This time shifts the time axis backwards in time in the .dat files. The time axis in the .dat file is formed by the starting time point and the sampling time, i. e. by the number of samples and the time between the individual samples. Thus, the start time in the .dat file is the
"normal start time" – [minus] Offset Starttime.

A pre-trigger can be generated in conjunction with the delay in the data to be saved (using the DELAY function block).

❑ Recording starts when the trigger is activated. In order to cancel the delay in the recording, the Offset Starttime must be set to the time delay.

❑ Write file (store_file)

Tip

Do not activate this field in this tab, but externally via the STORE_FILE connector.

Rising edge: the .dat file with the name and path configured in the settings is created. Recording is only started when STORE_VALUES = TRUE.

Falling edge: the .dat file is closed.

❑ Store values (store_values)

When this value is "TRUE" and the file is open, one .dat sample is stored in each evaluation cycle.

The handling of this parameter depends on the mode of the data:

For values that are not buffered, activate this field and leave the STORE_VALUES connector unconnected. You can use STORE_FILE to control the recording.

Note

If you control the recording of the unbuffered data using STORE_VALUES, you get an incorrect time axis. Since this is controlled in ibaAnalyzer by the number and time of the samples, switching STORE_VALUES on and off dynamically does not lead to any gaps on the time axis, but instead, the samples are arranged serially and gaps occur at the end of the .dat file.

This variable must be handled differently for buffered values. For more information, please refer to "*Sub-tab "Signal Configuration"*, page 102".

- ❑ Postprocessing (pp_enab/pp_command)

Post-processing compatible with ibaLogic V3 is available.
iba recommends using ibaDatCoordinator to realize post-processing. It provides comprehensive functions for further processing, e. g. copying files to a file server, transfer to ibaAnalyzer for extraction to a database etc.

- ❑ Information fields (FILE_INFO (FILEINFO / TECHNO_STRING /DISABLECOMPRESSION /RENAME_AFTER_CLOSE))

The FILE_INFO connection consists of a structure. This is supplied by an automatically generated Joiner, when you try to set a value.

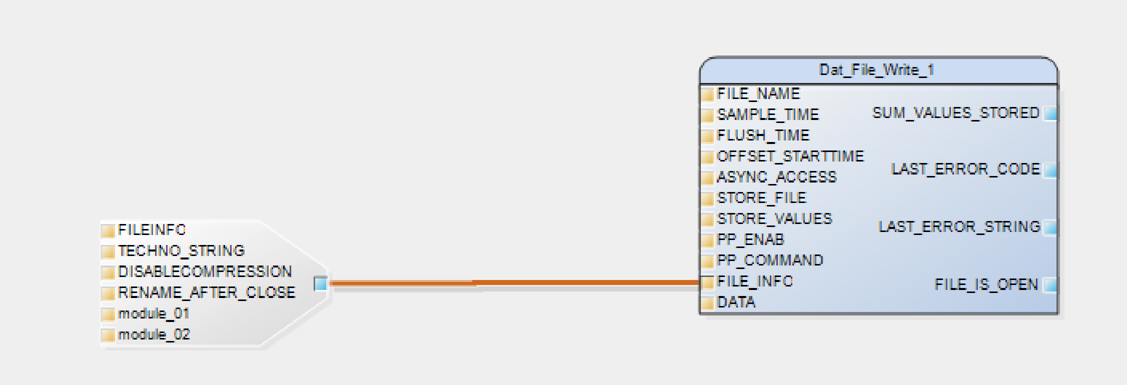


Figure 41: File information (FILE_INFO)

This is extra ASCII information that is saved when the file is closed. This information is available under "Info" when you conduct an analysis.
iba recommends that you use the FILE_INFO input connector in order to be able to modify the contents dynamically.

The information fields for the FILEINFO string in ibaAnalyzer are built up as follows:

Field name: text

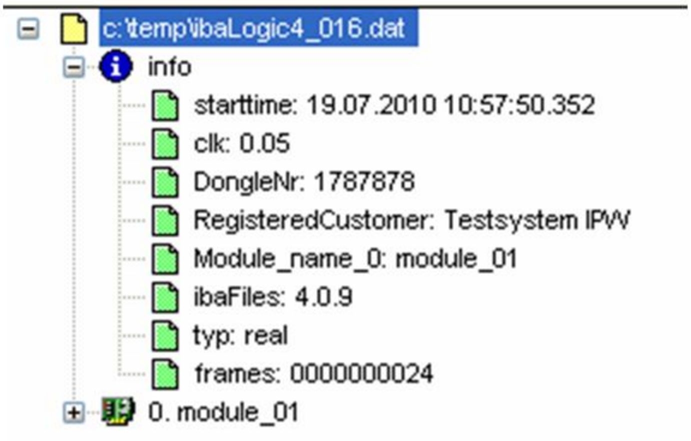
If no ":" is found in the text, ibaLogic sets "UserField0" as the default value.
Multiple info fields are separated by ";".

Note

Standard info field names cannot be overwritten. Entries with this name are ignored.

Exceptions are the module names. Module names can be formed dynamically by writing to the String FILEINFO. The part "Module_name_x:" is fixed and the desired name must be placed behind. Several module names are separated by ";". There is, however, a restriction: only 1024 Bytes are available for the String. Therefore it is better to use short module names. Note that every Byte of the string counts, i. e. even the standard module names occupy 12 Bytes each.

Standard info field names:



c:\temp\ibaLogic4_016.dat
- info
  - starttime: 19.07.2010 10:57:50.352
  - clk: 0.05
  - DongleNr: 1787878
  - RegisteredCustomer: Testsystem IPW
  - Module_name_0: module_01
  - ibaFiles: 4.0.9
  - typ: real
  - frames: 0000000024
- 0. module_01

Figure 42: Standard info field names

❑ Disable Data Compression in File (disablecompression)

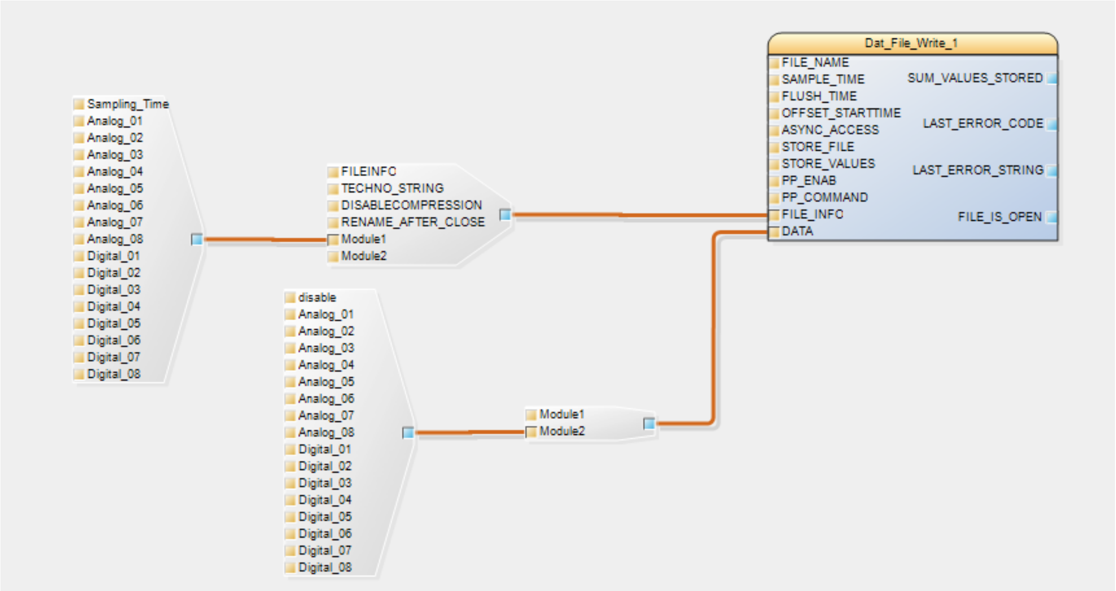
The data compression can thereby be prohibited. Compressed data files have a longer generation time and are in some cases larger than non-compressed data files. When having a constant value, the value and the amount of equivalent values will always be stored. If the value changes continually, it will always be stored with a number=1. The data file is thereby larger than it would have been when written uncompressed. This is to be preferred as an option especially at ibaPADU-S-IT-2x16.

Note

Use ibaAnalyzer V 6.4.2. and higher to be able to read uncompressed iba data files.

❑ Information data of the module (here: Module1/module2)

Here, the signal names per module can be given as string.
This is also a structure again. This can be received with an automatically generated joiner when trying to set a value.



In this example, the module consists of 8 analog and 8 digital values.

In addition, there is the Sampling_Time. This time serves to the subsampling per module.

The SAMPLE_TIME which is centrally set in the DFW function block is taken at a 0 value.

With this time it is possible to write buffered and unbuffered values with the DFW at the same time.

Example: A task runs in a 5 ms interval, whose signals are set on module1. Buffered values are sent by a peripheral every 20 ms for this 20 ms segment, those signals are set on module2.

Set the SAMPLE_TIME at DFW on 5 ms = for the unbuffered values and the SAMPLE_TIME at module2 for the buffered values on 20 ms.

The user shall set the buffered data on DISABLE=TRUE in the cycles so that the values buffered in the intercycles will not be written.

The analog/digital information fields serve to set online the signal names of the signals to be recorded. When common names such as Analog_01... etc. have been used in the DFW itself, it is possible to now set a meaningful name for this assigned signal. This information will be taken over into the iba .dat file within the next change of the file name.

Thereby, it is possible to sign online new signals on reserved channels and to name them, whereby the restriction of the DFW, which can only be parameterized offline internally, can be evaded.

❑ Directory (Part of file_name):

➲ By clicking on the browser button <...> select the drive and path where the .dat files should be saved.

Important note

You cannot work with the Windows file browser if the runtime system is located on the ibaPADU-S-IT-2x16 target system. Specify the path and file name in the preset to the FILE_NAME connector in the "Arguments" tab or link the connector with a string variable in which you can set the path and file name dynamically. The input gets accepted when the file (STORE_FILE) is opened.

The default "C:\dat" should be used as directory. The data can then be retrieved with the ibaDatCoordinator. In ibaDatCoordinator, data can then be accessed with "\\S-IT-16-000074\RamDisk\dat", for example. Here, "S-IT-16-000074" is either the host name or the IP address of the addressed ibaPADU-S-IT-2x16, "RamDisk" is the internal release name and "dat" the specified directory name.

C:\	= RAM memory
D:\	= Flash memory

Please refer to the ibaPADU-S-IT-2x16 manual, which memory is to be used for which task.

Please note: if a DAT file is written asynchronously, it will be a hidden file on ibaPADU-S-IT-2x16. The file can only be seen, when the folder options are set to show hidden files.

❑ File name template (part of file_name):

Specification of the file name and the index. The index is incremented for each new .dat file if the path and file name specified are not modified.

The "#" character is used as a placeholder for the index. Specify multiple placeholders for multi-digit indices.

#: 0 ... 9 → 10 files
##: 00 … 99 → 100 files

If the path is not modified, the oldest .dat files are overwritten after the index values have overflown.

❑ Sample time (in seconds) (sample_time)

This value is the time base for the .dat file. It defines the time in seconds between two values of a measurement signal that are saved.

In case of unbuffered values, specify the task interval for the program that contains the DFW function block.

Important note

The preparation of the signals to be measured and the DFW function block must run in the same task interval. If this is not the case, the time axis in the .dat file is either stretched or compressed.

Example of sample time

If the task in which you provide the data runs in an interval of 10 ms, you must set the sampling time in the DFW function block to 0.01 sec.

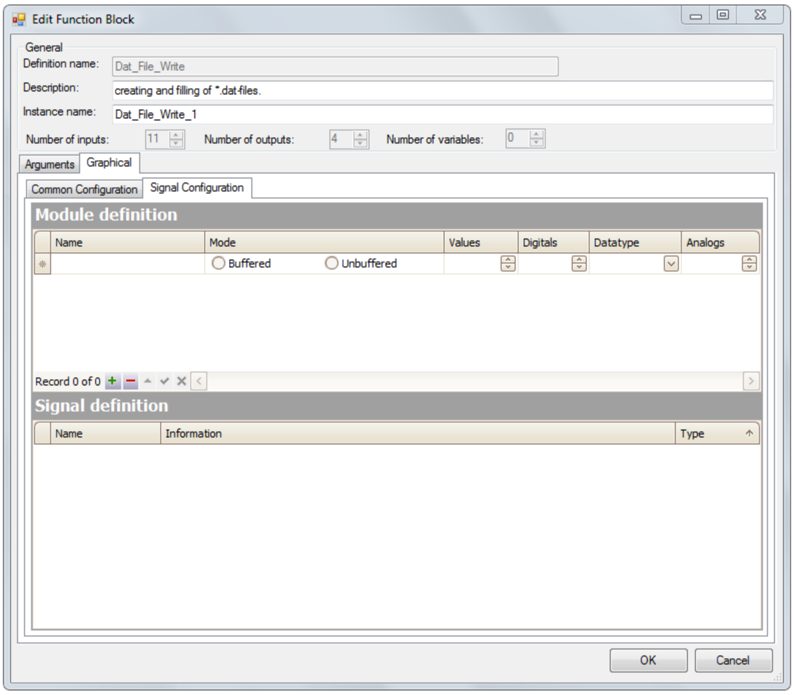
If you want the values to be saved only every 50 ms, it is not sufficient to set only the sample time to 0.05 sec. (in this case, the same data gets saved, but the X-axis shows a time period that is 5 times longer), but instead, you must specify a clock cycle at the STORE_VALUES connector that becomes "TRUE" at every 5th cycle.

However, it is simpler to allow the function block to run in a 50 ms task, and to let the STORE_VALUES connector remain static with "TRUE".

Tip

If you see a time axis in the analysis that is too long or too short, despite correct settings of the parameters, please check whether the real task interval time (function block EVALTIMES) matches the one configured.

7.3.1.3 Sub-tab "Signal Configuration"



Definition name: Dat_File_Write

Description: creating and filling of *.dat-files.

Instance name: Dat_File_Write_1

Number of inputs: 11 Number of outputs: 4 Number of variables: 0

Figure 43: Sub-tab "Signal Configuration"

The "Signal configuration" sub-tab contains the following areas:

- ❑ Module definition:
  defines the module names, module type, the number of signals and their data type
- ❑ Signal definition:
  defines the signal names and the signal description

Important note

In the present version of ibaLogic, you need to configure the modules and signals in the DFW function block completely before connecting a measurement signal.

Reason: ibaLogic creates internal structure and array data types that cannot be modified once they have been used. If you modify the signal configuration subsequently, you must either remove all joiners that have been inserted automatically and rewire them or adapt all corresponding data types in the ST function blocks.

However, if it is only about the number of channels, make sure you have enough reserves and use FILEINFO for the dynamic assignment of the associated channel names as described in FILEINFO.

In order to create a new module, an entry is provided at the end of the module list that contains a blank name field. Simply enter the module name here and configure this module. When quitting, a new entry is again generated automatically at the end of the list.

In principle, any number of modules can be created. Furthermore, the maximum number depends on your license.

Description of the modules:

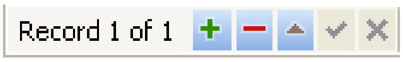
- Name:
  Module name that must conform to the IEC standard.
- Unbuffered mode:
  Recording of individual values. One data sample is saved in each cycle. The value in the "Values" column is not considered.
- Buffered mode:
  Recording packets. An array of data samples is saved in each cycle. The value in the "Values" column specifies the array depth, i. e. the number of samples.
  For further information, please see "*Buffered Mode*, page 193".
- Values:
  Meaningless in the "Unbuffered" mode.
  Number of samples saved per cycle in the "Buffered" mode.
- Digital values:
  Number of binary signals in this module (max. 32)
- Data type of the analog values, REAL and INTEGER are permissible
- Analog values:
  Number of analog values in this module (max. 32)

All signals pertaining to the module marked are displayed under the signal definition. The signals are created with the default names (Digital_nn and Analog_nn). You can edit the signal names and enter a description for each signal under "Information".

Note

You cannot modify the signal configuration as long as you are online and the connector “DATA” is connected with data. You have to go offline and disconnect if you wish to make any modifications. In the course of this, any joiners that have been generated automatically are removed.

Toolbar for editing the module definition record:



Symbol	Name/Tooltip	Explanation
+	Append	Adds a new blank module definition record.
−	Delete	Removes the module definition record selected.
▲	Edit	Releases the module definition record for editing.
✓	End Edit	The data of the modified module definition record is accepted.
✕	Cancel Edit	The data of the modified module definition record is not accepted. The input is canceled.
Record 1 of 1	-	Number of records

For more information, please refer to "*Practice Examples*, page 271“.

7.3.1.4 Generate Storage Structure

The simplest case of storing the .dat files generated is to specify a fixed folder and a fixed base name for the files, with ibaLogic assigning a serial number to the base name automatically.

If you need a storage structure with sub-folders and file names, which, for example, should contain the current batch number, this needs to be programmed.

Example

A new file should be created every hour. The file name should contain the hour value in the form of a name.

Implementation

The file name is formed using the current hour value. A low pulse is fed to the STORE_FILE input of the DFW when the value changes (change of hour) using the DELAY function block and the EQ function. In this manner, the upcoming name is accepted for the next .dat file.

The file name is formed with the CONCAT function. The path name and base file name are available at the first CONCAT. The hour is appended to this and with the 2nd CONCAT the .dat extension is appended. Thus, the file name c:\iba11.dat is generated here.

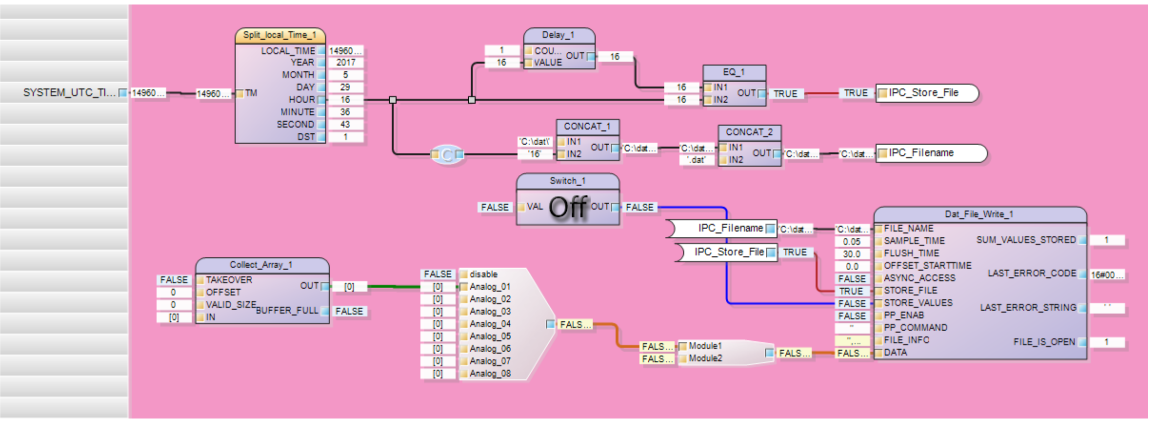


Figure 44: Example: CONCAT functions

You can also compose a unique .dat file name including the path in exactly the same way. If you specify a new path, ibaLogic creates it automatically.

7.3.1.5 Generate a vector by DWF

A vector can also be created with the DFW block, according to the group vector in ibaPDA. Please make the following entries:

ibaLogic Version: V5

❑ For every vector create his name in the header as an info-field:

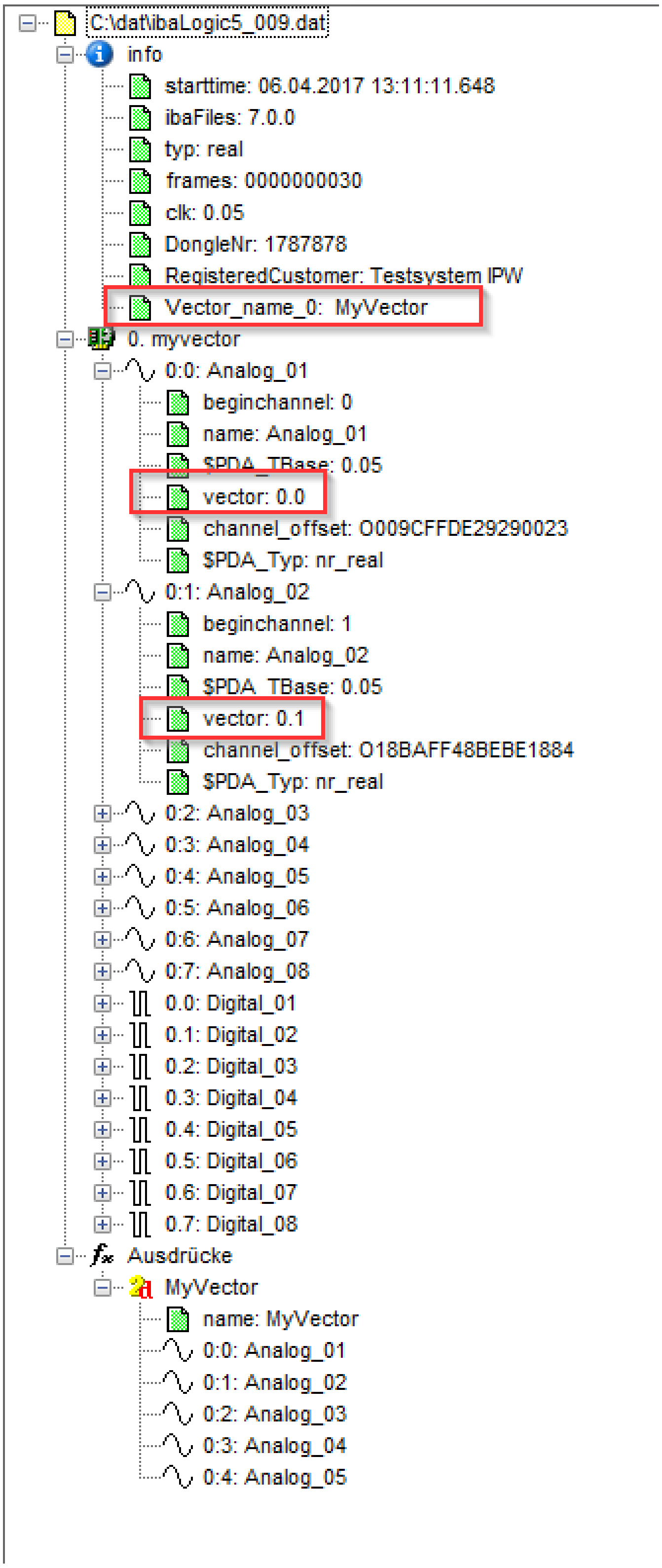
- Vector_name_0: MyVector

  Vector_name_x = fixed name with increasing number for x
  MyVector = name for my vector

❑ Every signal must get an info-field:

- vector:0.0

  vector: = fixed field name
  0.0 = Vector_index:Signal_index e.g. here it means that Analog_01 is the first signal inside the vector MyVector
  Important note: write this info-field without blanks between!



```
C:\dat\ibaLogic5_009.dat
  info
    starttime: 06.04.2017 13:11:11.648
    ibaFiles: 7.0.0
    typ: real
    frames: 0000000030
    clk: 0.05
    DongleNr: 1787878
    RegisteredCustomer: Testsystem IPW
    Vector_name_0:  MyVector
  0. myvector
    0:0: Analog_01
      beginchannel: 0
      name: Analog_01
      $PDA_TBase: 0.05
      vector: 0.0
      channel_offset: 0009CFFDE29290023
      $PDA_Typ: nr_real
    0:1: Analog_02
      beginchannel: 1
      name: Analog_02
      $PDA_TBase: 0.05
      vector: 0.1
      channel_offset: 018BAFF48BEBE1884
      $PDA_Typ: nr_real
    0:2: Analog_03
    0:3: Analog_04
    0:4: Analog_05
    0:5: Analog_06
    0:6: Analog_07
    0:7: Analog_08
    0.0: Digital_01
    0.1: Digital_02
    0.2: Digital_03
    0.3: Digital_04
    0.4: Digital_05
    0.5: Digital_06
    0.6: Digital_07
    0.7: Digital_08
  Ausdrücke
    MyVector
      name: MyVector
      0:0: Analog_01
      0:1: Analog_02
      0:2: Analog_03
      0:3: Analog_04
      0:4: Analog_05
```

Figure 45: Vectors in the DAT-file

The vector is then built automatically in the DAT-file.

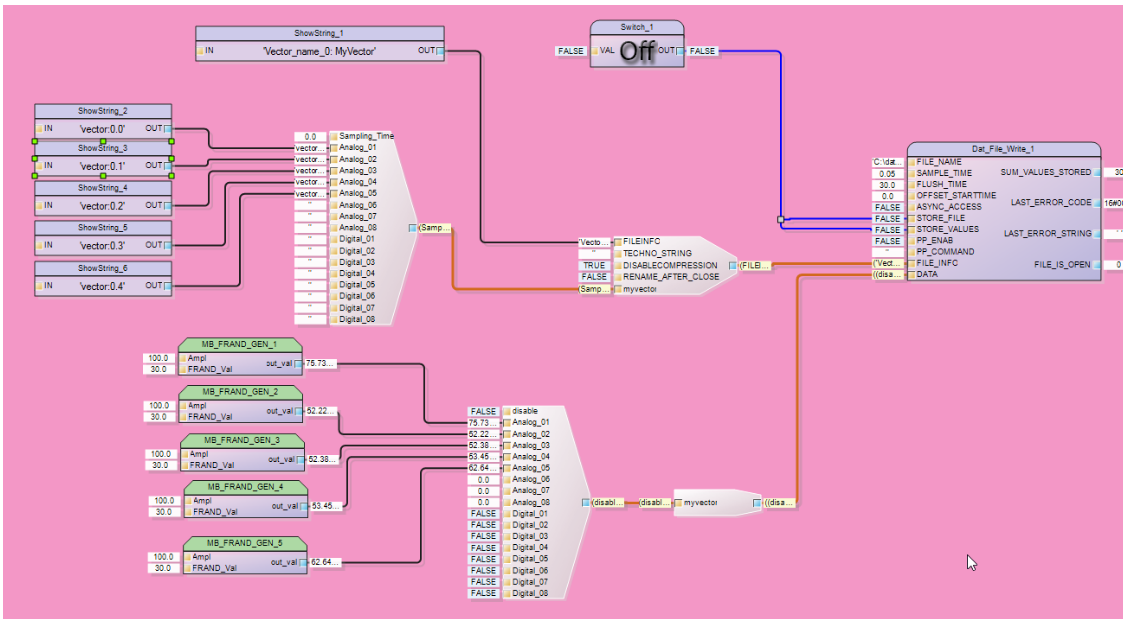


Figure 46: Online display

If you want to hide the signal in the normal signals add *hidden:1* to the signal.

In this example, the first signal is hidden:

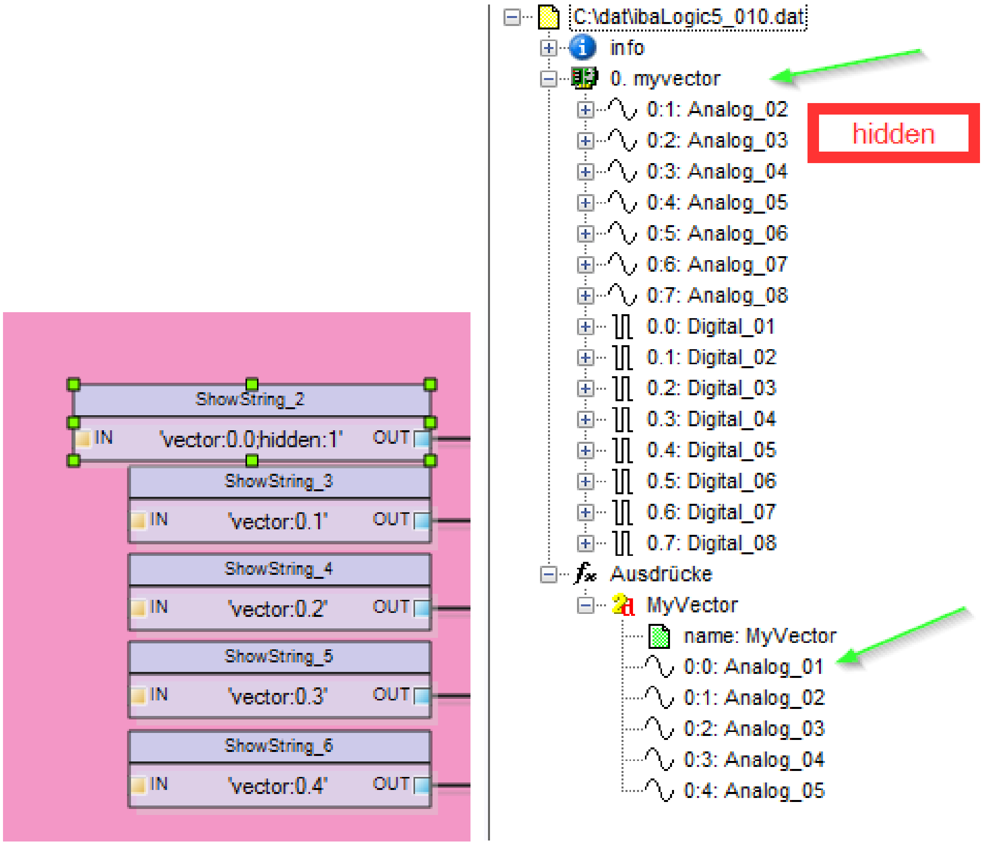


Figure 47: Hide signals

7.3.2 TCPIP_SENDRECV

This function block enables transmission and reception of data via TCP/IP.
The data here is raw data that is sent via TCP/IP. In this manner, all native TCP/IP protocols can be reconstructed.

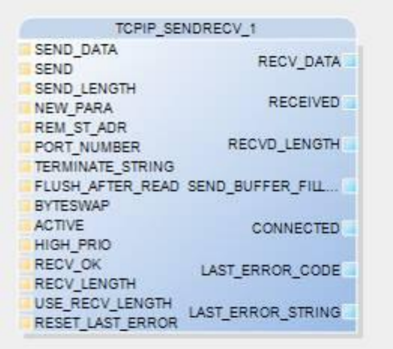


Figure 48: TCPIP_SENDRECV Function block

Note

This function block requires a license. There is a maximum of 4 function blocks available in the lite version. There is no limitation in the full version, the computer/network performance is decisive here.

7.3.2.1 Inputs

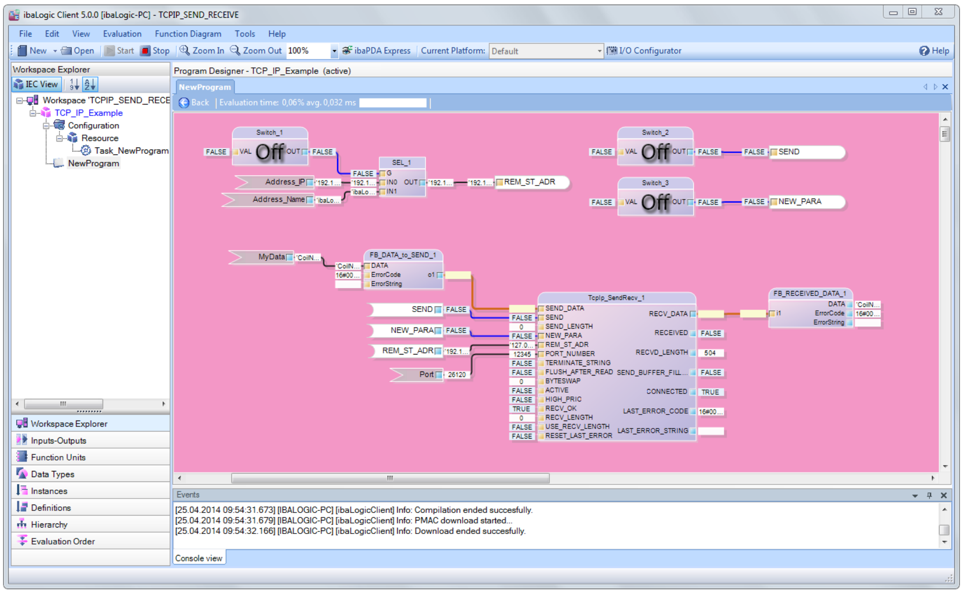
Connector	Data type	Explanation
SEND_DATA	Any	Data to be transmitted The data type is ANY, i.e. it aligns itself with the interface data type, e. g. a structure, string, array.
SEND	Bool	When it is "TRUE", data available at SEND_DATA is transmitted. This input is not edge-oriented, i. e. a fixed "TRUE" at the input initiates transmission in every cycle.
SEND_LENGTH	Udint	Length of the data to be transmitted in bytes. If the value is 0, all data available is transmitted
NEW_PARA	Bool	Accept new link parameters.
REM_ST_ADR	String	IP of the link partner. This IP is required only for active link establishment, i. e. when the input ACTIVE = TRUE. If ACTIVE = FALSE, the function block waits for the partner, which must specify the IP address of the PC or the ibaPADU-S-IT. You can also enter the computer name instead of the IP address. Resolving the name may take some time during creation depending on the configuration of the Operating System.
PORT_NUMBER	Udint	When ACTIVE = TRUE: Port number of the partners. When ACTIVE = FALSE: Own port number
TERMINATE_STRING	Udint	Strings are terminated with a NULL byte. This input is evaluated only when strings are available at the SEND_DATA input.
FLUSH_AFTER_READ	Bool	Deletes the receive buffer after reading the data.
BYTESWAP	Int	= 1: Swap based on data type (AB CDEF → BA FEDC) = 2: Swap 2 Bytes respectively (ABCD → BADC) = 4: Swap 4 Bytes respectively (ABCD → DCBA)

Connector	Data type	Explanation
ACTIVE	Bool	TRUE: (ibaLogic is the TCP/IP client) The function block attempts to establish a link to the IP and the PORT, which are specified in REM_ST_ADR and PORT_NUMBER respectively. FALSE: (ibaLogic is the TCP/IP server) It waits for incoming connections at the port number configured. The IP address is not evaluated.
HIGH_PRIO	Bool	The data is fetched with a higher priority from the Windows network buffer and written in the input buffer. **NOTE** By default, this function should be set to "FALSE".
RECV_OK	Bool	TRUE: Data received is OK, and the input buffer can be filled with the new data. FALSE: The data last received remains until the input is triggered again.
RECV_LENGTH	Udint	Defines the telegram length in bytes. This input is evaluated only if the input, USE_RECV_LENGTH = TRUE
USE_RECV_LENGTH	Bool	TRUE: If the input buffer is larger than the RECV_LENGTH configured, there is only one telegram with the length configured at the output, RECV_DATA. FALSE: At the output, RECV_DATA, there is one telegram with the maximum size of the data type available at the output, RECV_DATA.
RESET_LAST_ERROR	Bool	Resets the error outputs

7.3.2.2 Outputs

Connector	DataType	Explanation
RECV_DATA	Any	Received data. The data type depends on the connected data type, e. g. a structure, string, array.
RECEIVED	Bool	This output is TRUE as soon as any data is received
RECVD_LENGTH	Udint	Length of the data received in bytes
SEND_BUFFER_FILLED	Bool	TRUE when the internal buffer is full, i. e. the function block cannot transmit data fast enough
CONNECTED	Bool	TRUE as soon as the connection is set up
LAST_ERROR_CODE	Dword	Last error message as a DWORD in hex format
LAST_ERROR_STRING	String	Last error message as clear text

Example of a Send - Receive procedure



```
[25.04.2014 09:54:31.673] [IBALOGIC-PC] [ibaLogicClient] Info: Compilation ended succesfully.
[25.04.2014 09:54:31.679] [IBALOGIC-PC] [ibaLogicClient] Info: PMAC download started...
[25.04.2014 09:54:32.166] [IBALOGIC-PC] [ibaLogicClient] Info: Download ended succesfully.
```

Figure 49: Example of a Send - Receive procedure

In this example, the data to be transmitted is available in the form of a structure. The data is transmitted with the help of a manual trigger.

There is a switch at NEW_PARA in order to re-establish the link for test purposes or in case of modification in the link parameters, if required.

This function block is passive and waits until a connection has been established by the communication partner.

The data received arrive at RECV_DATA. The input RECV_OK is set permanently to "TRUE", since the data does not have to be buffered intermediately, because it can be processed well within the configured task cycle time.

7.3.3 PIDT1_CONTROL

Universal PIDT1 controller that can be switched to operating modes as a P, I, PI or PIDT1 controller.

❑ Set the starting value of the integrator

❑ Hold the instantaneous value of the integrator

❑ Pre-control value WP

❑ Controller limits LL and LU

❑ Proportional coefficient KP

❑ Reset time TN

❑ Control direction reversible

❑ Indication when the limits configured are reached

❑ Display of the error signal

❑ Display of separate P, I and DT1 controller outputs

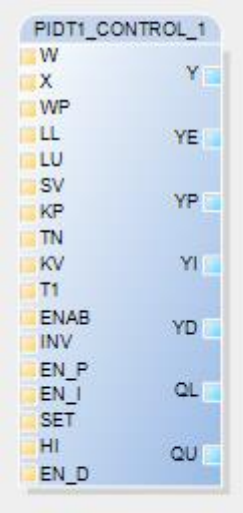


Figure 50: PIDT1_CONTROL Function block

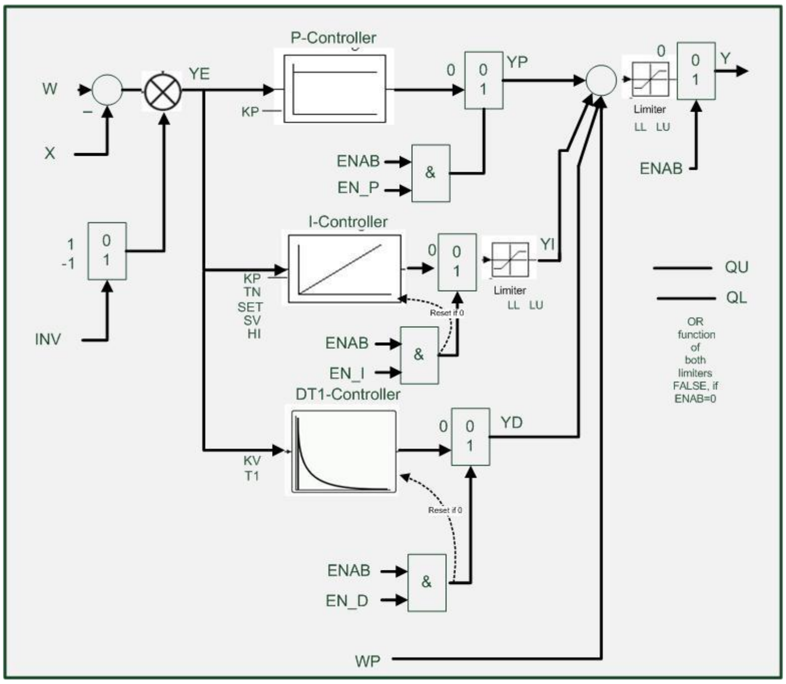


Figure 51: Block diagram of a universal PIDT1 controller

7.3.3.1 Inputs

Connector	Data Type	Explanation
W	Lreal	Set-point value
X	Lreal	Actual value
WP	Lreal	Pre-control value
LL	Lreal	Lower limit value (valid for Y and YI)
LU	Lreal	Upper limit value (valid for Y and YI)
SV	Lreal	Set value for the integrator is accepted with SET
KP	Lreal	P gain

Connector	Data Type	Explanation
TN	Time	Reset time
KV	Lreal	D gain
T1	Time	D time constant
ENAB	Bool	Controller release
INV	Bool	Invert the sign of the control deviation
EN_P	Bool	Activate the P controller
EN_I	Bool	Activate the I controller
SET	Bool	Set the integrator with the value SV
HI	Bool	Stop the integrator
EN_D	Bool	Activate the D controller

7.3.3.2 Outputs

Connector	Data type	Explanation
Y	Lreal	Control value = YP+YI+YD+WP
YE	Lreal	Control deviation = W-X
YP	Lreal	Output value of P controller = KP*YE
YI	Lreal	Output value of I controller = YI_{n-1}+KP*YE*Ta/TN
YD	Lreal	Output value of D controller = α*YD_{n-1}+ α*KV*ΔYE α = 1/ (1+Ta/T1) ΔYE = (YE-YE_{n-1})
QL	Bool	Lower limit value reached
QU	Bool	Upper limit value reached

7.3.3.3 Details / Signal trends

The various signal trends of the individual controller components are illustrated in the following diagrams.

Control value Y:

The control value Y is the sum of the P, I and D components and the pre-control value WP.

If the input, ENAB (controller release) is not set, the control value is always 0.0.

Input WP, pre-control value:

This input is added to the output Y.

Input LL/LU, lower / upper limit value:

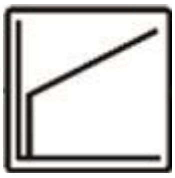
Note

The total output Y, as well as YI are limited.

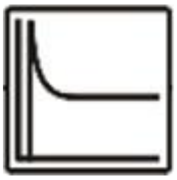
The outputs, QL and QU, accordingly take up the value TRUE.

The following controllers are used in practice:

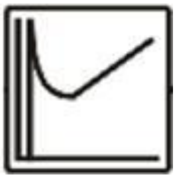
PI controller



PD controller



PID controller



The P, I and DT1 components are given special consideration in the following.

7.3.3.4 P component: (Parameter: KP, EN_P)

The P component of the controller is calculated as KP*YE. The value is fed to the output value only when EN_P is set.

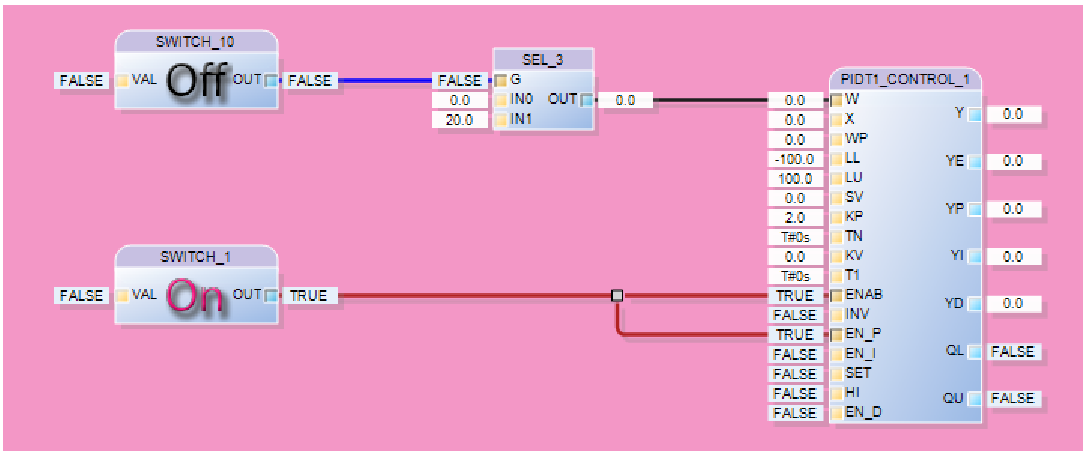


Figure 52: P component

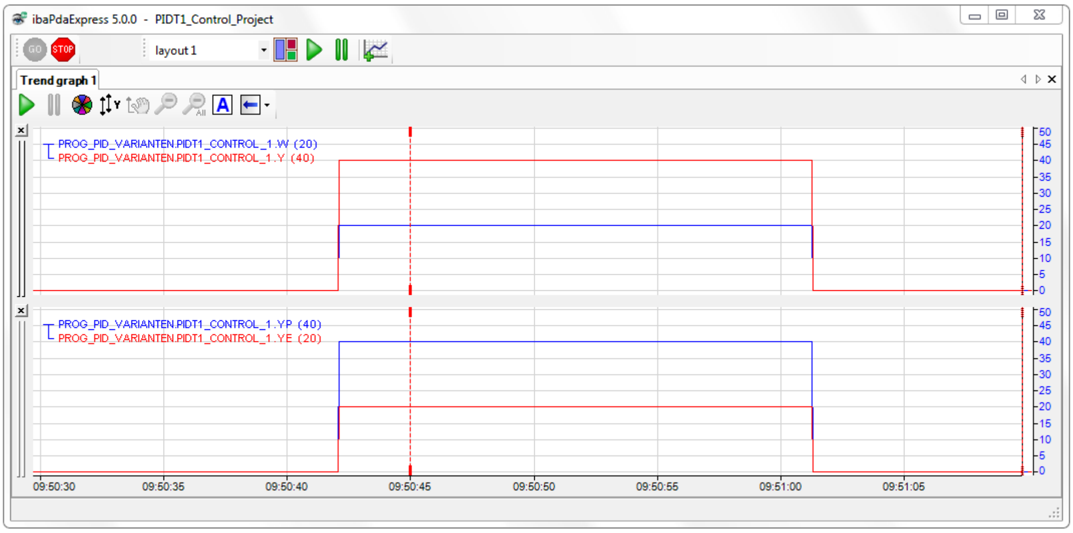

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.PIDT1_CONTROL_1.W (20)
PROG_PID_VARIANTEN.PIDT1_CONTROL_1.Y (40)
PROG_PID_VARIANTEN.PIDT1_CONTROL_1.YP (40)
PROG_PID_VARIANTEN.PIDT1_CONTROL_1.YE (20)
50
45
40
35
30
25
20
15
10
5
0
09:50:30
09:50:35
09:50:40
09:50:45
09:50:50
09:50:55
09:51:00
09:51:05

Figure 53: P component

7.3.3.5 I component: (Parameters KP, TN, SET, SV, HI and EN_I)

The I component of the controller is calculated as
$YI_n := YI_{n-1} + KP * YE * Ta/TN$ (Ta = Task time).
This component can be set to the value of the input SET with the input SV. A value of "TRUE" at the HI input stops the integrator. The value is fed to the output value only when EN_I is set.

Correlations

TN = KP*Ta = KP/KI

Example 1

KP = 1.0

TN = 1 s

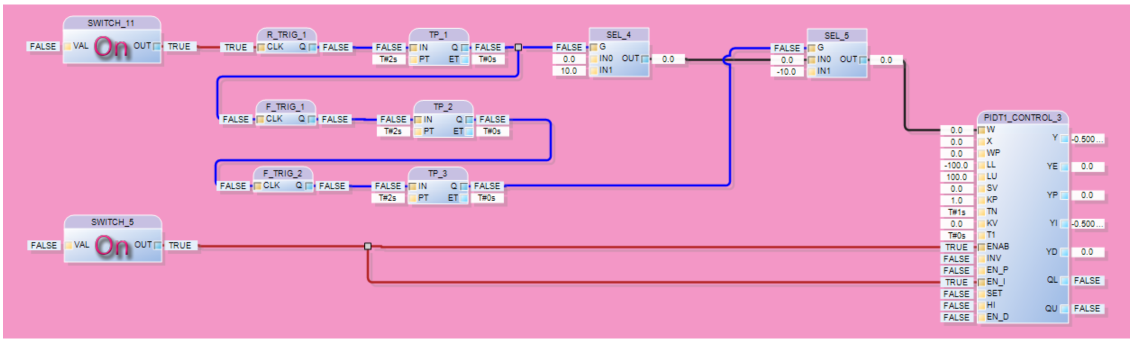


Figure 54: I component

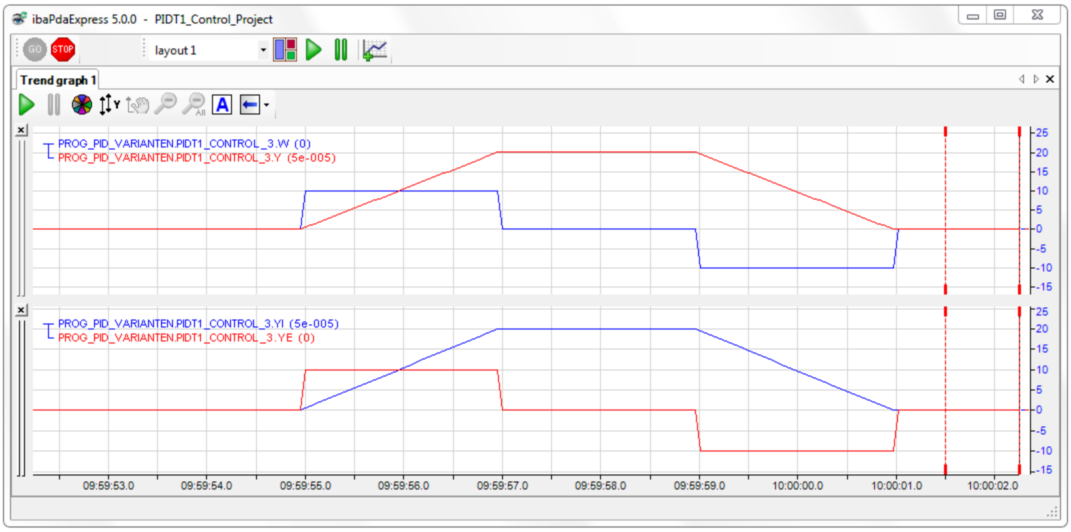


Figure 55: I component

7.3.3.6 DT1 component: (Parameter KV,T1,EN_D)

The DT1 component of the controller is obtained as

$YD: = \alpha*YD_{n-1} + \alpha*KV*\Delta YE$

$\alpha = 1/ (1+Ta/T1)$

$\Delta YE = (YE-YE_{n-1})$

(Ta = Task time)

The value is fed to the output value only when EN_D is set.

Example 1

KV = 0.5

T1 = 1 s

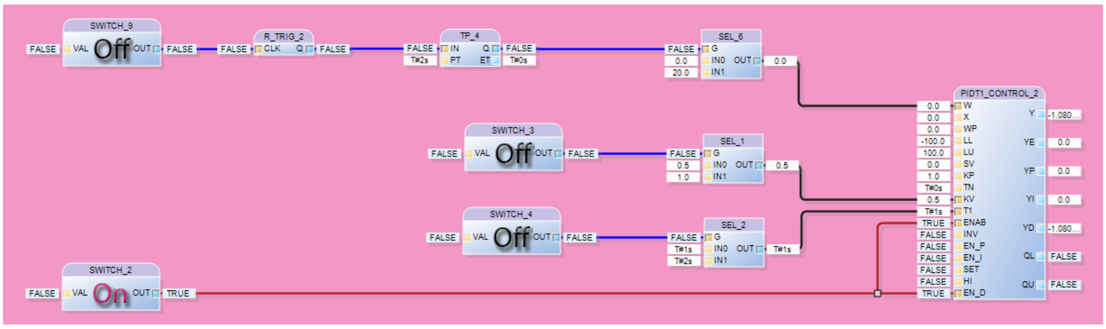


Figure 56: DT1 component

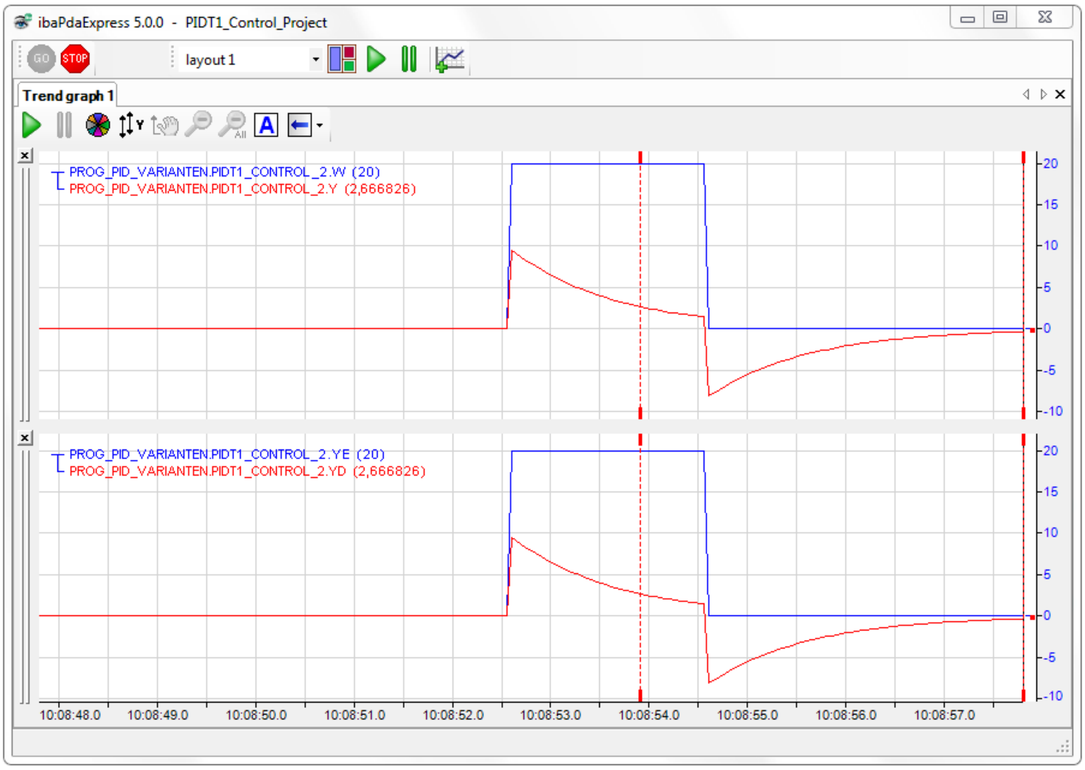

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.W (20)
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.Y (2,666826)
20
15
10
5
0
-5
-10
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.YE (20)
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.YD (2,666826)
20
15
10
5
0
-5
-10
10:08:48.0
10:08:49.0
10:08:50.0
10:08:51.0
10:08:52.0
10:08:53.0
10:08:54.0
10:08:55.0
10:08:56.0
10:08:57.0

Figure 57: DT1 component

Example 2

KV = 1

T1 = 2 s

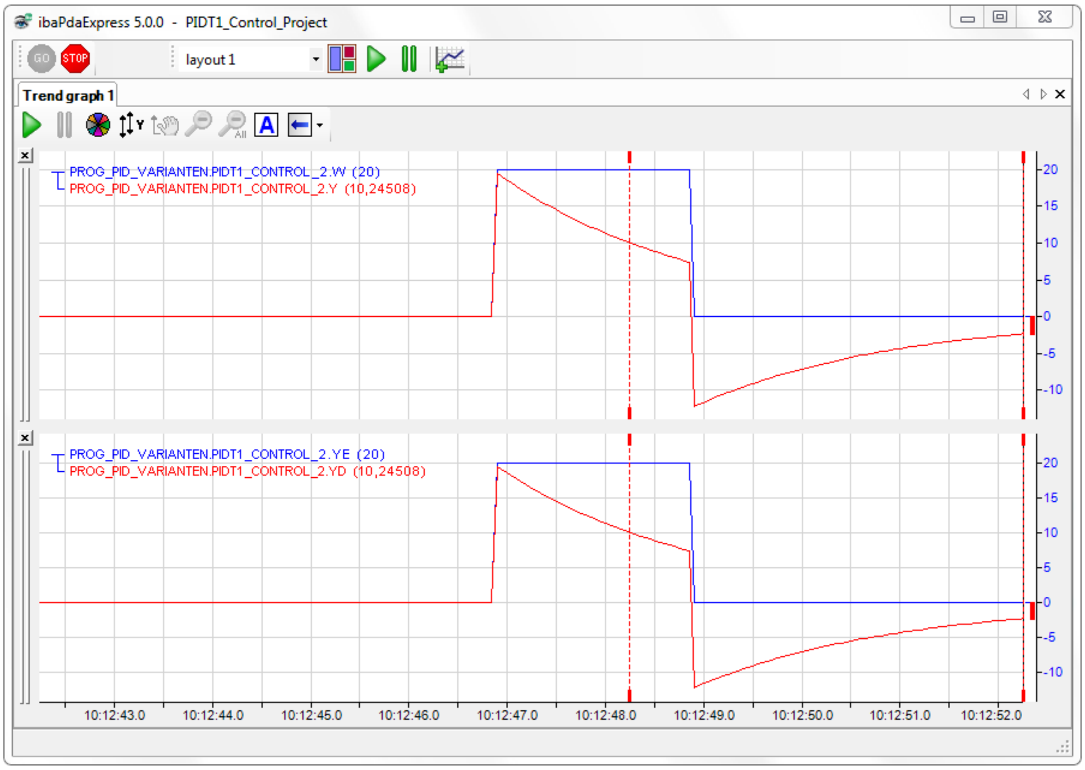

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.W (20)
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.Y (10,24508)
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.YE (20)
PROG_PID_VARIANTEN.PIDT1_CONTROL_2.YD (10,24508)
20
15
10
5
0
-5
-10
10:12:43.0
10:12:44.0
10:12:45.0
10:12:46.0
10:12:47.0
10:12:48.0
10:12:49.0
10:12:50.0
10:12:51.0
10:12:52.0

Figure 58: DT1 component

7.3.3.7 PIDT1 component – Total response

Example 1

Example of the complete PIDT1 controller with signal trends.

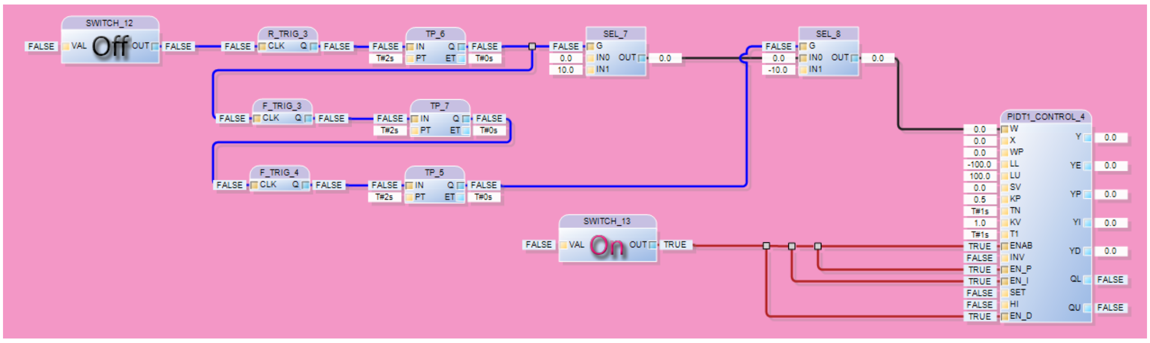


Figure 59: PIDT1 controller with signal trends

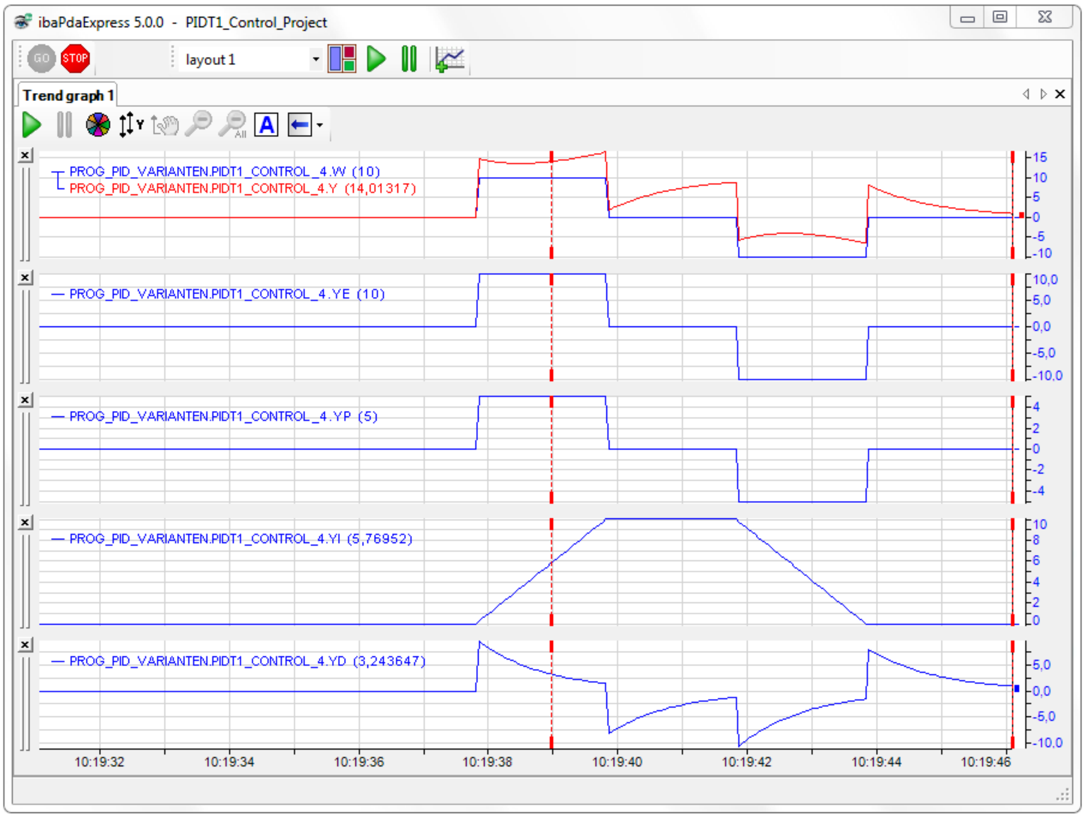

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.W (10)
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.Y (14,01317)
15
10
5
0
-5
-10
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.YE (10)
10,0
5,0
0,0
-5,0
-10,0
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.YP (5)
4
2
0
-2
-4
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.YI (5,76952)
10
8
6
4
2
0
PROG_PID_VARIANTEN.PIDT1_CONTROL_4.YD (3,243647)
5,0
0,0
-5,0
-10,0
10:19:32
10:19:34
10:19:36
10:19:38
10:19:40
10:19:42
10:19:44
10:19:46

Figure 60: PIDT1 controller with signal trends

7.3.4 RAMP

Ramp function block with 2 different ramps: Manual and automatic mode

- ❑ Set-point value limit
- ❑ Start up the new set-point value via the ramp
- ❑ Set the set-point value
- ❑ Indication when the limit values are exceeded

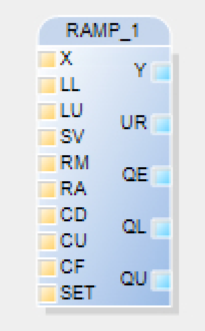


Figure 61: RAMP Function block

7.3.4.1 Inputs

Connector	Data Type	Meaning / Usage
X	Lreal	Input value (Set-point)
LL	Lreal	Lower limit value
LU	Lreal	Upper limit value
SV	Lreal	Set value, output is set to this value with SET
RM	Lreal	Manual ramp (1/s), valid for CD and CU
RA	Lreal	Automatic ramp (1/s), valid for CF
CD	Bool	Ramp falling (manual ramp control)
CU	Bool	Ramp rising (manual ramp control)
CF	Bool	Ramp as per the input value (automatic ramp control), has precedence over CD and CU
SET	Bool	Set output value to SV

7.3.4.2 Outputs

Connector	Data Type	Meaning / Usage
Y	Lreal	Output value; $Y_n = Y_{n-1}$ + UR r = ramp used
UR	Lreal	Ramp used (1/s)
QE	Bool	Output value = Input value
QL	Bool	Lower limit value reached
QU	Bool	Upper limit value reached

7.3.4.3 Example

The inputs CD, CU and CF control the ramps. If none of the inputs is active, the last output value is fixed. The output UR then displays the ramp used as the value 0.

If the input CD is active, regardless of the input value, the current output value at the manual ramp is scaled down to a maximum of the lower limit.

If the input CU is active, regardless of the input value, the current output value is raised via the manual ramp up to a maximum of the upper limit.

If both CD and CU are active simultaneously, UR is set to 0. The output value does not change.

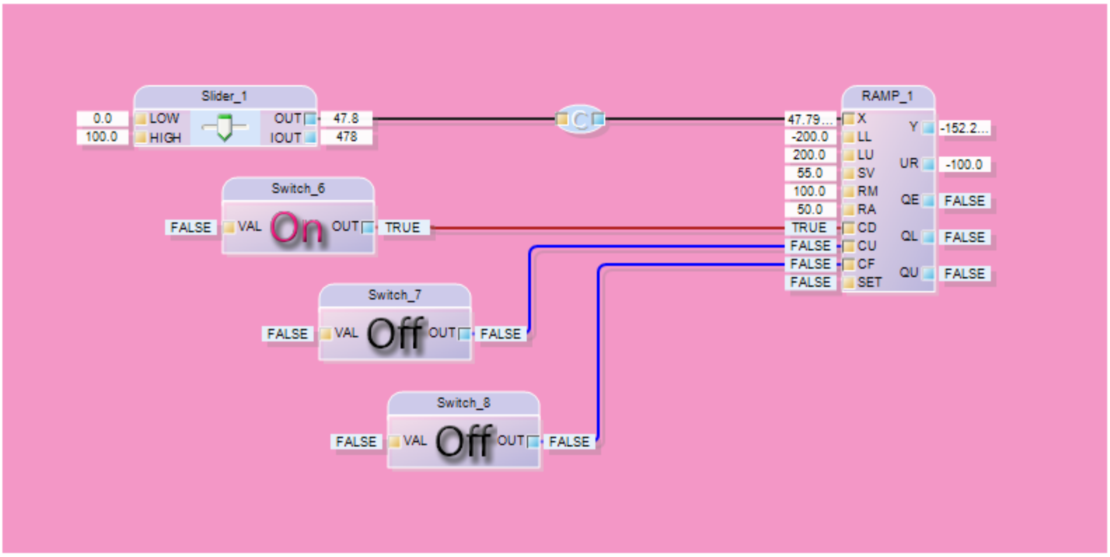


Figure 62: Controlling ramps

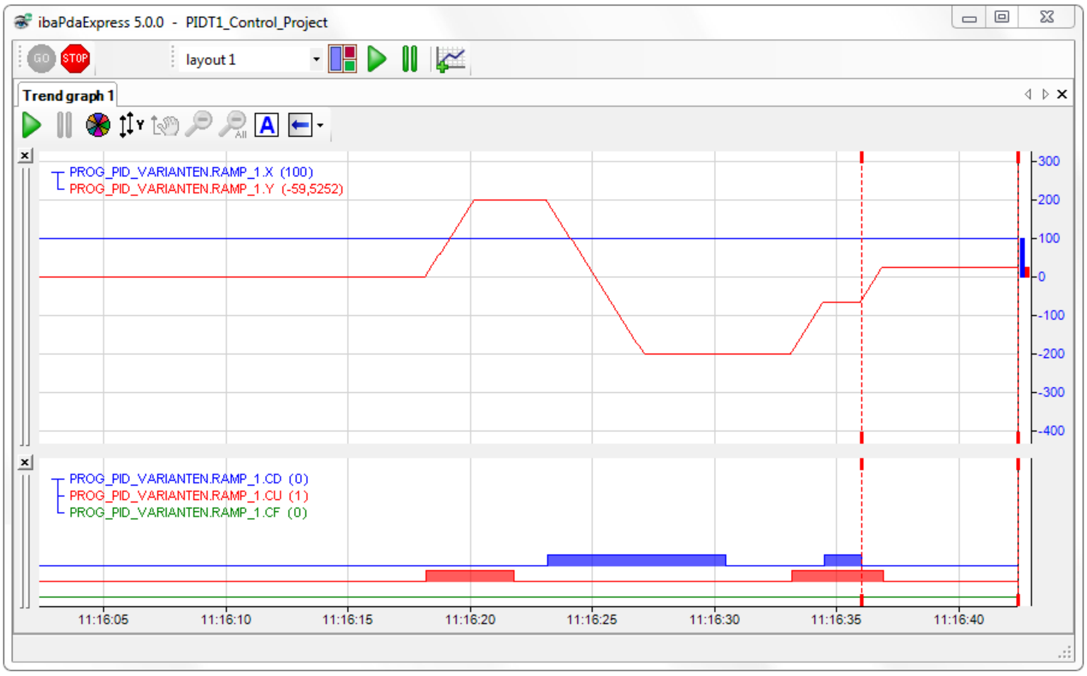

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.RAMP_1.X (100)
PROG_PID_VARIANTEN.RAMP_1.Y (-59,5252)
300
200
100
0
-100
-200
-300
-400
PROG_PID_VARIANTEN.RAMP_1.CD (0)
PROG_PID_VARIANTEN.RAMP_1.CU (1)
PROG_PID_VARIANTEN.RAMP_1.CF (0)
11:16:05
11:16:10
11:16:15
11:16:20
11:16:25
11:16:30
11:16:35
11:16:40

Figure 63: Controlling ramps

If the input CF is active, the output value tracks the input value via the automatic ramp. If the input value exceeds the limit, the output value changes in line with the ramp only up to the limits.

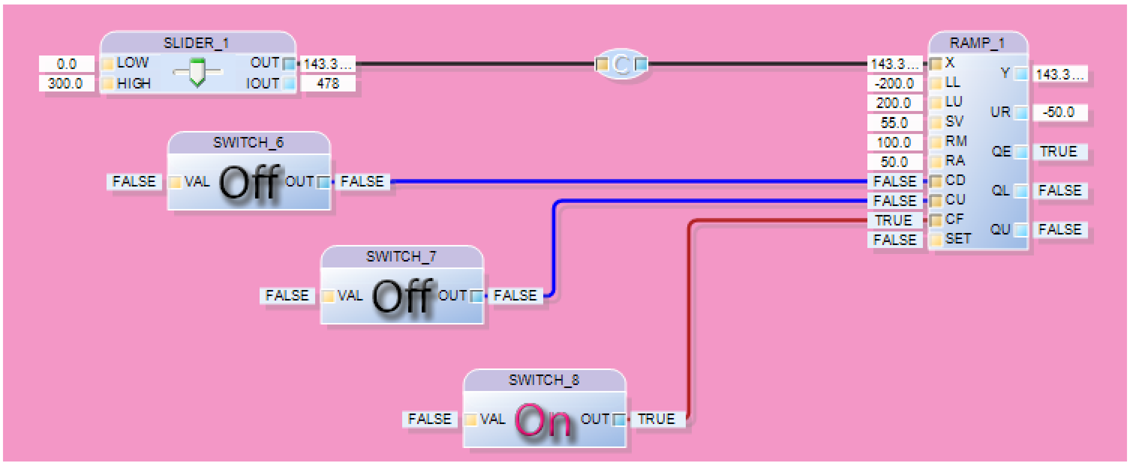


Figure 64: Controlling ramps

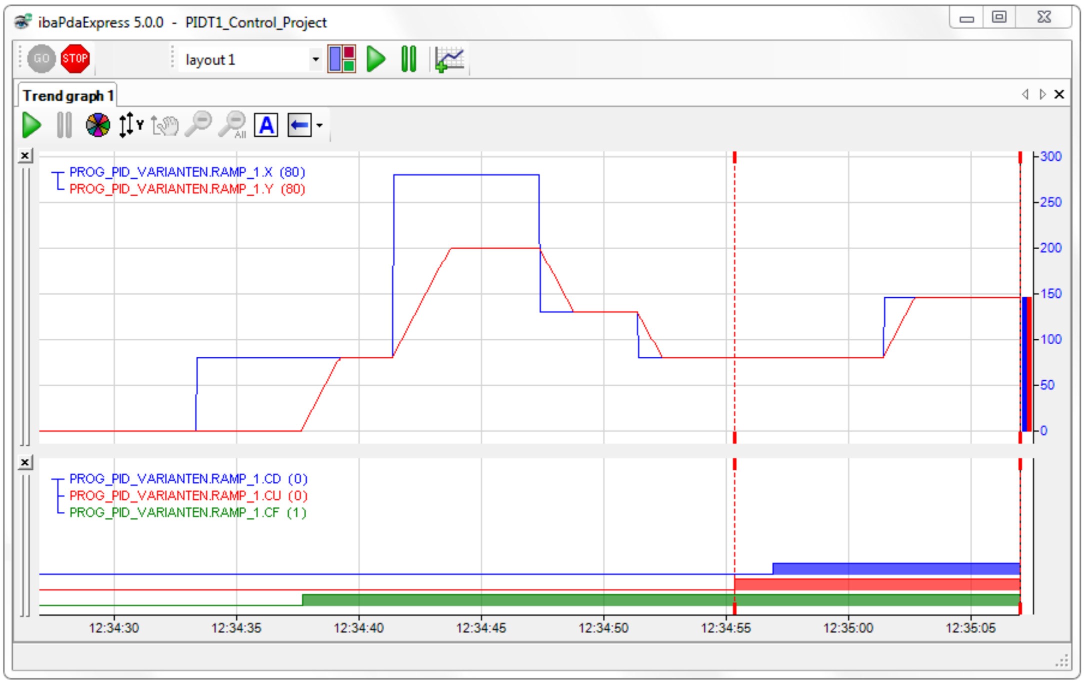

ibaPdaExpress 5.0.0 - PIDT1_Control_Project
GO
STOP
layout 1
Trend graph 1
PROG_PID_VARIANTEN.RAMP_1.X (80)
PROG_PID_VARIANTEN.RAMP_1.Y (80)
300
250
200
150
100
50
0
PROG_PID_VARIANTEN.RAMP_1.CD (0)
PROG_PID_VARIANTEN.RAMP_1.CU (0)
PROG_PID_VARIANTEN.RAMP_1.CF (1)
12:34:30
12:34:35
12:34:40
12:34:45
12:34:50
12:34:55
12:35:00
12:35:05

Figure 65: Controlling ramps

If CF is set, the inputs CD and CU do not have any effect.

7.4 User-specific function units

ibaLogic has a global library of pre-defined function units. Nonetheless, it may be necessary to define your own function units for a more efficient solution to your problem. There are three types of user-specific function units:

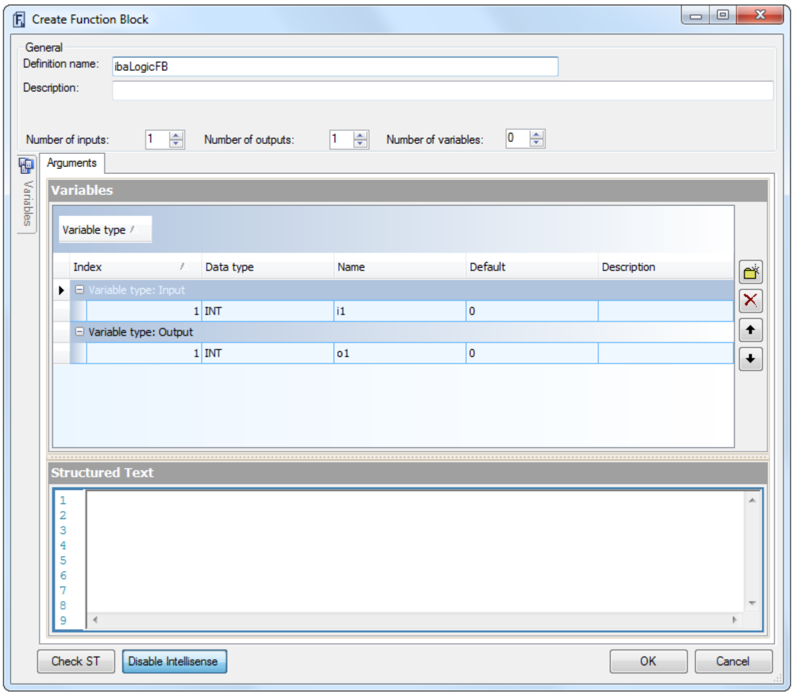
- ❑ Function blocks that were created in ibaLogic using the high-level programming language, Structured Text (ST).
- ❑ It is also possible to combine existing graphics programming to macros.
- ❑ Function blocks that were not created in ibaLogic in a high-level language (C++, others on request e.g.: FORTRAN) and are integrated as DLL into ibaLogic (requires a licence).

After having been created, all these function units are treated like standard function blocks.

7.4.1 Function Blocks

In order to create a new function block, position the mouse pointer to a free location in the program window and select "New... - New Function Block..." in the context menu. The "Create Function Block" dialog box is displayed.

The fields for the name and the table for defining the inputs, outputs and internal variables are located at the top of the box.



Create Function Block

General

Definition name: ibaLogicFB

Description:

Number of inputs: 1 Number of outputs: 1 Number of variables: 0

Arguments

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	INT	i1	0	
Variable type: Output				
1	INT	o1	0	

Structured Text

Figure 66: "Create Function Block" dialog box

7.4.1.1 General settings

Definition name

The definition name is the name with which the function unit is saved in the units folder. The instance name is formed from this name by appending an index.

Note on difference between Definition - Instance

For more information, please see "*Instances*, page 52".

Note

iba recommends using different prefixes for function blocks and macros, e.g. "FB_" and "MB_". You can configure the settings under the "Tools – Options – Function Units" menu.

You can also find the default values for the names and data types of the variables, and iba recommends using the names "i", "o", "io" and "v". You can configure data types based on your requirements.

Instance name

The instance name is not displayed during creation. It is displayed only when you retrieve a function unit to use it in a project.

Description

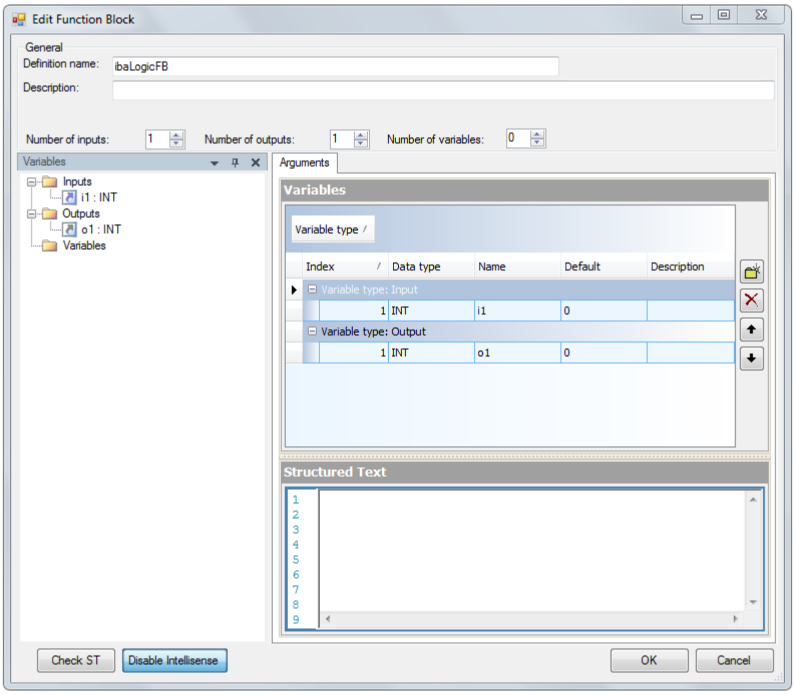
You can describe the unit function in more detail in this field. This description is displayed as a tooltip when you move the mouse pointer over the function unit name in the library.

Number of inputs / outputs / variables

You specify the number of variables used here. One line is created in the table for each variable.

Variables

The list of variables is available either as a tree or as a table for display.



Edit Function Block

General

Definition name: ibaLogicFB

Description:

Number of inputs: 1 Number of outputs: 1 Number of variables: 0

Variables
- Inputs
  - i1 : INT
- Outputs
  - o1 : INT
- Variables

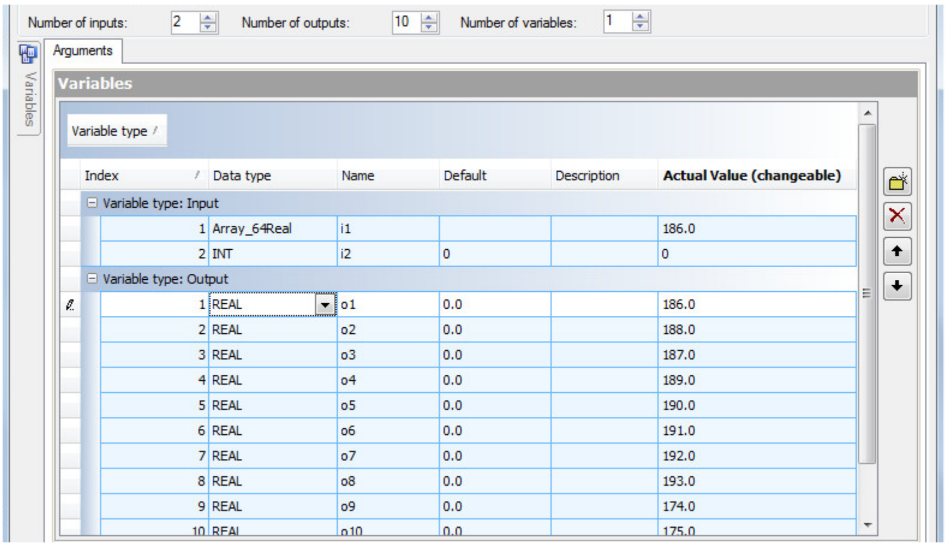
Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	INT	i1	0	
Variable type: Output				
1	INT	o1	0	

Figure 67: "Edit Function Block" dialog box with the list of variables

➲ You can open the tree by clicking on the "Variables" tab to the left of the table. It only serves for the view without any functions.

This view is hidden, since you can configure and customize the variables only in the variables table.



Number of inputs: 2 Number of outputs: 10 Number of variables: 1

Variables

Index	Data type	Name	Default	Description	Actual Value (changeable)
Variable type: Input					
1	Array_64Real	i1			186.0
2	INT	i2	0		0
Variable type: Output					
1	REAL	o1	0.0		186.0
2	REAL	o2	0.0		188.0
3	REAL	o3	0.0		187.0
4	REAL	o4	0.0		189.0
5	REAL	o5	0.0		190.0
6	REAL	o6	0.0		191.0
7	REAL	o7	0.0		192.0
8	REAL	o8	0.0		193.0
9	REAL	o9	0.0		174.0
10	REAL	o10	0.0		175.0

Figure 68: Function block variables

Variable

Column	Explanation
Index	Each variable type begins with index 1.
Data type	Selection box for accepting a defined variable type. You can also create new user types here. You can find the default value under the "Tools Options" menu. **Tip** The fast selection of a data type in the field can be done by entering the initial letter.
Name	Default value consists of prefix and index. You can, however, also edit a new name.
Default	Please note that the notation of the values depends on the data type. For more information, please refer to "*Syntax Description of Structured Text*, page 124". The value is assigned to the variable provided that no connection is established (for input variables) or there is no assignment within the block code. The default value will only be evaluated when starting ibaLogic. Online changes can be made with the field Actual Value .
Description	*Text field*[4] that is displayed as a tooltip in the function unit folder.

[4] Annotation: If the description is changed in the function block definition, it will appear in the instance only after reopening of the workspace.

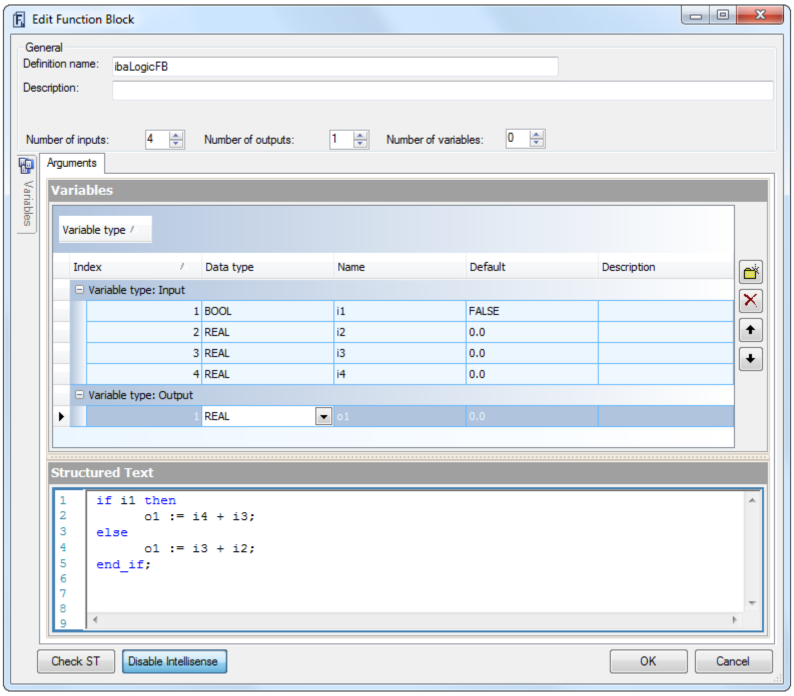
Column	Explanation
Actual value	Current values of the variables, in case of arrays and structures, only the first element is displayed. This field is only visible in the online mode (bold). Changes are only temporary. Possibly change the default value, too, if the value is supposed to be kept when restarting. **Tip** This value can be modified manually, but it is re-evaluated in the next cycle and hence, if required, overwritten. **Important note** If you remove a link to the input connector in the online mode, the last value remains.

Using the buttons on the right, you can change the sequence of the variables, add and delete new variables at the marked position.

Symbol	Explanation
✱	Adding an element.
✕	Deleting an element.
↑ ↓	Interchanging the elements.

7.4.2 Structured Text Editor

You can define the functionality of a function block using the Structured Text editor.



Edit Function Block

General

Definition name: ibaLogicFB

Description:

Number of inputs: 4 Number of outputs: 1 Number of variables: 0

Arguments

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	BOOL	i1	FALSE	
2	REAL	i2	0.0	
3	REAL	i3	0.0	
4	REAL	i4	0.0	
Variable type: Output				
1	REAL	o1	0.0	

Structured Text

```
if i1 then
      o1 := i4 + i3;
else
      o1 := i3 + i2;
end_if;
```

Figure 69: Structured Text Editor

The following buttons are located above and below the text input field:

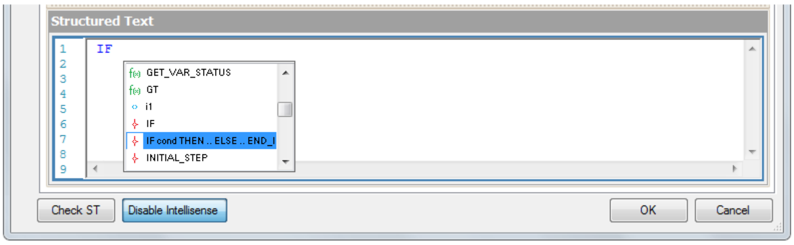
Button	Explanation
Check ST	Syntax test of the entered code, without compiling it
Enable / Disable Intellisense	Enable or disable the Intellisense resource

7.4.2.1 IntelliSense

IntelliSense is a resource for completing entries automatically. It provides additional information and selection options to the programmer to facilitate the completion of data entry.

During the creation of function blocks, also new variables are automatically added to IntelliSense.

In particular, it simplifies working with structures considerably, since with structure variables, all defined elements are provided immediately for selection after entering the "." separator.



```
IF
```

Figure 70: Structured Text editor with IntelliSense enabled

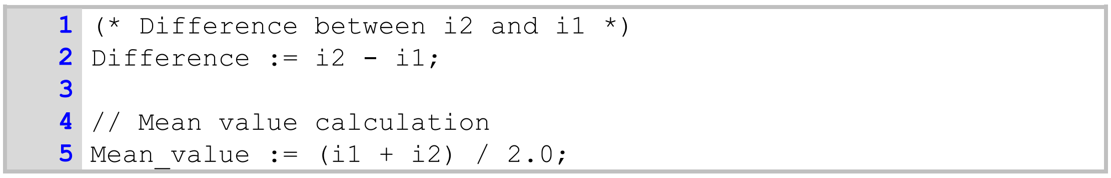
Example: The IntelliSense selection window appears by entering "IF". You can enter the choice highlighted with <Return>.

Statements such as IF..THEN..ELSE / WHILE…/ REPEAT… are provided only after you enter the first word, and can, hence, be taken over completely at this time as a framework.

You can make the selection using <Cursor up> or <Cursor down>, and accept the entry with <Tab>.

7.4.2.2 Syntax Description of Structured Text

For example, Structured text can look like the following:



```
(* Difference between i2 and i1 *)
Difference := i2 - i1;

// Mean value calculation
Mean_value := (i1 + i2) / 2.0;
```

Figure 71: Syntax of Structured Text

Notations:

- Comments are enclosed in "(*" and "*)".
- Single line comments must start with //.
- Statements must be terminated with a semicolon.
- The value of the result must be written on the left of ":=".
- Expressions consist of operators and operands.

Important Note

Apart from the operators and statements described in the following, you can also call up some functions that are available graphically as a block, even from within the ST. You will find notes on whether and how these can be used in ST in the function description in the section "*Standard Function Units*, page 301". There, for each function unit with the keyword "ST:" a note has been provided regarding its usability in ST.

7.4.2.3 Operators

List of the operators sorted by precedence:

Operator	Example	Value in the example	Description	Priority
()	`(2+3) * (4+5)`	45	Brackets	Highest
**	`3**4`	81	Exponentiation	
-	`-10`	-10	Negation	
NOT		NOT TRUE	Logical negation	
*	`10**3`	30	Multiplication	
/	`6/2`	3	Division	
MOD	`17 MOD 10`	7	Modulus (Division remainder)	
+	`2+3`	5	Addition	
-	`4-2`	2	Subtraction	
<, >, <=, >=	`4 > 12`	FALSE	Comparison	
=	`T#26h = T#1d2h`	TRUE	Equality	
<>	`8 <> 16`	TRUE	Inequality	
&, AND	`TRUE & FALSE`	FALSE	Boolean AND	
XOR	`TRUE XOR FALSE`	TRUE	Boolean Exclusive Or	
OR	`TRUE OR FALSE`	TRUE	Boolean Or	Lowest

7.4.2.4 Statements

Key-word	Example	Description
;	`;`	Blank statement
:=	`Var1 := 12;`	Assigning the value 12 to the variable name given on the left.

f(i1, i2, …)	`o1 :=`
	`concat (iDir, iFile, v1);`
		Function call, see also the description of the function blocks

Key-word	Example	Description

IF	IF i1 < i2 THEN
	  o1 := 1;
	[ELSIF i1 =i2 THEN
	  o1 := 2;]
	ELSE
	  o1 := 3;
	END_IF;
		Conditional statement.
		The condition is a Boolean expression (that yields the result "TRUE" or "FALSE")
		Optional extensions are given in square brackets [...].

CASE	CASE i1 OF
	1: o1:=3;
	2: o1:=4;
	3,4,5: o1 := 5;
	     o2 := 6;
	11..15: o1:= 11;
	[ELSE o1 := 0;
	     o2 := 0;]
	END_CASE;
		Case statement.
		The argument "i1" is an expression of type ANY_INT or ENUM.
		There are one or more statements per case.
		A case can have several integers (3,4,5) or enumerators or ranges of integers (10...15). The ELSE branch is optional.
		Optional extensions are given in square brackets [...].

FOR	FLAG := FALSE;
	FOR ix:= 1 TO 100 **[** BY 2 **]** DO
	     IF o1[ix] = iy THEN
	        FLAG := TRUE;
	        EXIT;
	     END_IF;
	END_FOR;
	IF FLAG THEN (* found*)
		Unconditional loop (iteration).
		The step (BY xx) is optional. If it is not present, the step is 1.
		The type of the loop variable is ANY_INT and it should not be modified within the loop.
		Optional extensions are given in square brackets [...].
		Except the Index Array information.
		Attention
		There is a risk of endless (infinite) loops

WHILE	WHILE i1 > 1 DO
	     o1 := o1/2;
	END_WHILE;
		Conditional loop
		Attention
		There is a risk of endless (infinite) loops

REPEAT	REPEAT
	     o1:= o1 * i1;
	UNTIL o1 > 10000
	END_REPEAT;
		Conditional loop
		The difference to WHILE is: the loop is executed at least once, even if the condition is not met right at the beginning.
		Attention
		There is a risk of endless (infinite) loops

EXIT	FLAG := FALSE;
	FOR ix:= 1 TO 100 **[** BY 2 **]** DO
	     IF o1[ix] = iy THEN
	        FLAG := TRUE;
	        EXIT;
	     END_IF;
	END_FOR;
	IF FLAG THEN (* found *)
		Premature termination of a FOR, WHILE or REPEAT loop.
		The first statement is executed after the next end of the loop, i. e. with nested loops, execution continues with the next higher level.
		Optional extensions are given in square brackets [...].

RETURN	oFLAG := FALSE;
	FOR ix:= 1 TO 100 **[** BY 2 **]** DO
	     IF o1[ix] = iy THEN
	        oFLAG := TRUE;
	        RETURN;
	     END_IF;
	END_FOR;
		Return statement, premature termination of the function block.
		Example: if the value iy is contained in the array ix, the result is TRUE, otherwise it is FALSE.
		Optional extensions are given in square brackets [...].

Key-word	Example	Description

ARRAY access	`<ArrayType>[index,…]`
	
	`o1 := iArray[0];`
	`o1 := iArray[0,0,…];`
	`o1 := iArray[0][0];`
		The indices are given in square brackets
		Access to a 1-dimensional array
		Access to an n-dimensional array
		Access to a nested array
		For more information, please see "*ARRAY TYPE Group*, page 143".

ENUM access	`Enumerator`
	
	`IF (i1 > 0) THEN`
	`     v1 := Forward;`
	`ELSIF (i1 < 0) THEN`
	`     v1 := Switch back;`
	`ELSE`
	`     v1 := Stop;`
	`END_IF;`
		Example: the type of the variable v1 is "Switch".
		"Switch" is an ENUM type, "Forward", "Stop" and "Switch back" are the enumerators.
		For more information, please refer to "ENUM TYPE Group

Structure access	`<STRUCTURE NAME>.ELEMENT`
	
	`o1.Temperature := i1;`
	`o1.Speed := i2;`
		The structure elements are separated with "." from the structure elements.
		Example:
		"o1" is a variable of the structure type. "Temperature" and "Speed" are structure elements.
		For more information, please refer to "*STRUCT TYPE Group*, page 144".

7.4.2.5 Constants

Description	Example
Integer and bit strings (except BOOL)	`-12  0  123_456  +986`
Decimal representation	

Binary representation	`2#1111_1111 (255 decimal)`
	`2#1111_0000 (240 decimal)`

Hexadecimal representation	`16#FF or 16#ff`
	`16#00F0_FFE0`

Real	`-12.0  0.0  0.4560  3.14159_26`

Real with exponent	`-1.34E-12 OR -1.34e-12`
	`1.0E+6 OR 1.0e+6`
	`1.234E6 OR 1.234e6`

BOOL	`0 OR FALSE`
	`1 OR TRUE`

Time constants	Type identification with: „T#" ,
	Time specification with: „d" (day), „h" (hour), „m" (minute), „s" (second) and „ms" (millisecond).
	`T#12d12h17m42s`
	`T#16d_2h_5m`

For all specifications, in general:	It is permitted to use a simple underscore for optical structuring.

7.4.2.6 Strings

Strings are enclosed in single quotation marks.

A $ character followed by a hexadecimal number is interpreted as ASCII code.

Note

IEC also permits double quotation marks. These WSTRING have not been implemented at present.

Special characters allowed in strings

Combination	Interpretation when printing out
$$	Dollar character
$'	Single quotation mark
$L or $l	Line feed (LF)
$N or $n	New line (NL)
$P or $p	Form feed (page)
$R or $r	Carriage return (CR)
$T or $t	Tabulator

Characteristics and examples of strings

Example	Explanation
''	Blank string (Length = 0)
'A'	String of length one, contains the character A
' '	String of length one, contains the blank character
'$''	String of length one, contains the single quotation mark
'"'	String of length one, contains the double quotation mark
'$R$L'	String of length two, contains the ASCII characters for CR and LF
'$$1.00'	String of length five, contains "$1.00"
'ÄË' '$C4$CB'	String of length two, contains "Ä" and "Ë"; In one case, directly as ASCII characters and in the second case, with the corresponding hexadecimal code of the extended character table (see "*Character tables*, page 350")

7.4.3 Macro block

You use macros in order to combine associated function units and thus, achieve a clear program layout.

Properties:

❑ You can export macros.

❑ You can copy macros to the global folder, and, as a result, you can use them several times and even in other projects.

❑ Macros may contain other macros and, of course, user-created function blocks, too.

❑ No OTCs, switches and sliders are permissible within macros, but IPCs are allowed. Links to other program components are permissible only with input and output connectors.

❑ You cannot use any hardware input and output resources directly within macros.

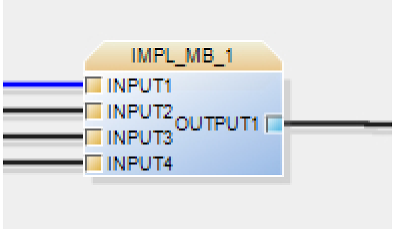
❑ A macro that has been created can be expanded again, i. e. the macro is resolved and the function units that it contains are displayed at the next higher level.

7.4.3.1 Creating a Macro Block

Macros are created manually in the same way that function blocks are created.

Procedure

1. Position the mouse pointer on a free location in the program window and call up "New... - New Macro Block..." in the context menu. A dialog box is displayed for entering the function unit names and variables. For more information, please see "*Function Blocks*, page 120".
2. Exit the dialog with <OK>. A blank macro block is displayed.



3. Double click on the macro block displayed in order to open the internal graphical programming interface.
4. Place and manage the function units or other macro blocks within this macro block so that you achieve the desired functionality.

Example

In this example the integer input is monitored for changes and every change is counted and placed at the output.

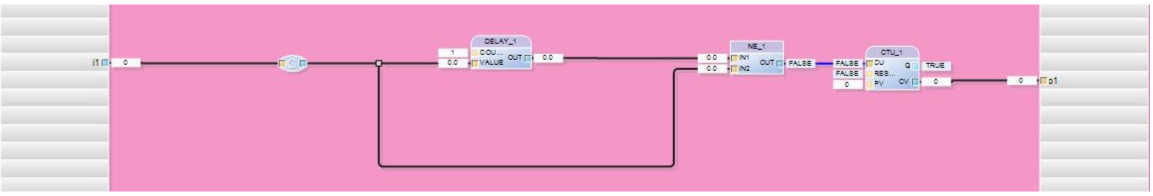


Figure 72: Integer monitoring

As in the case of every program, a new register is created for the contents of the macro with the macro name within the program designer.

Please note the evaluation context here. You come to the calling level by clicking on this. For further information, see "*Arrangement of the Tabs and Programming Windows*, page 61".

7.4.3.2 Opening a Macro

1. Double click on the macro block instance in a program.

7.4.3.3 Combining existing components into a macro block

ibaLogic provides the option of combining several function units already existing into one macro block.

➲ To do this, select the function units that need to be combined, as illustrated in the following diagram.

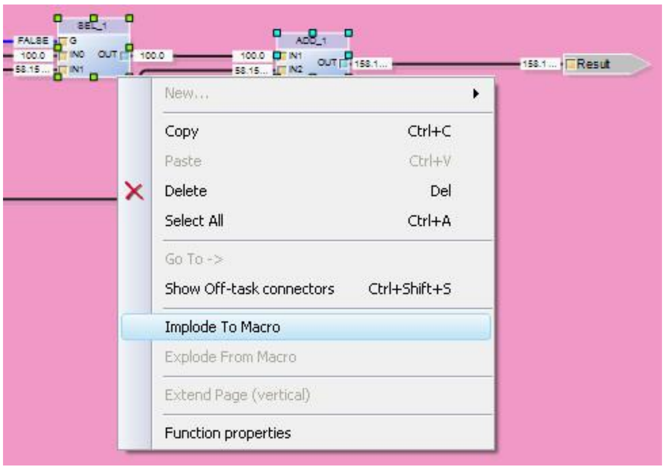
Note

Please note that

- you also mark the associated links/nodes when making the selection. Nodes usually are located externally.
- no OTCs, Switches or Sliders have been marked.
- links whose target or source function unit are not selected are created as macro input or output.

➲ Open the context menu using one of the elements selected.

➲ Select the option "Implode To Macro". The "Edit Function Block" dialog box is displayed.



➲ Assign a meaningful name to the new macro block and to the inputs and outputs. Assign meaningful names that conform to the IEC standard.

Tip

You can also make these changes subsequently by clicking on the macro created using the right mouse button and selecting the macro properties.

Result

The result is a new macro block (IMPL_MB_1) having the same functionality as the function units selected previously.

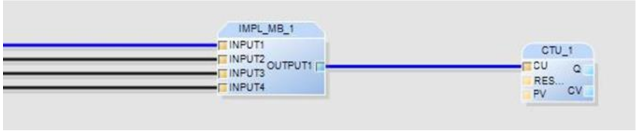


Figure 73: Macro block (IMPL_MB_1)

➲ You can open the macro by double-clicking and continue to edit the graphical elements.

In the online mode, you can also see the current values in the value pads depending on the evaluation context.

For further information on the evaluation context, please see "*Arrangement of the Tabs and Programming Windows*, page 61".

7.4.3.4 Expanding a Macro Block

An existing macro block can be expanded again. In doing so, the blocks defined are placed at the next higher level.

Procedure

1. To do this, mark the macro block.
2. Select "Explode From Macro" in the context menu.

Example

A simple macro block having an internal adder that adds both inputs needs to be expanded again.

Result

The following diagram appears after expanding:

- ❑ The adder has been revealed.
- ❑ The original macro definition, however, continues to be available in the function unit library.

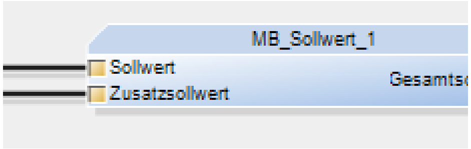


Figure 74: Macro block MB_Set-point_1

➔

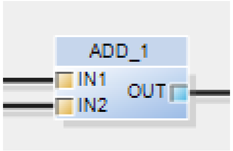


Figure 75: Expanded connection

7.4.4 Creating your own DLLs

Creating your own macros and function blocks using ST is a very easy option for handling several tasks in the field of automation technology. However, as easy as it is to create the macros and function blocks, it is also simple to copy them and to understand their contents, i. e. their function.

However, sometimes it is desirable to disclose less of one's own technological competence and, instead, for example, in case of a highly intelligent process-oriented technical solution, it is desired to prevent further uncontrolled proliferation of this technological know-how.

In such a case, it is beneficial to have the option of creating your own DLLs that contain the knowledge only in compiled form and, thus, cannot be extracted easily.

You can also realize these special connections

- ❑ in order to create complex function blocks.
- ❑ in order to work with the Windows environment.
- ❑ in order to allow tasks to execute in your own thread, and others.

In this manner, you can also integrate another high-level language under certain conditions.

You can then see the DLL created in ibaLogic as a completely normal function block with names, inputs and outputs. This can be differentiated from an ST function block only in the fact that you do not see any code in the program component.

Further documentation

This section contains only a brief overview. For a detailed description please contact the iba support.

Note

Using DLLs requires a license. The DLLs are not evaluated without a valid dongle that contains the DLL release.

Compiler

All DLLs written in C++ or Fortran are supported.

The following compilers have been tested for writing and compiling the DLLs:

- ❑ Intel Visual Fortran 10.0
- ❑ Microsoft Visual C++ 2005, 2008, 2010, 2012, 2015

However, there are still differences for the two device classes with ibaLogic (Windows PC or ibaPADU-S-IT-2x16). DLLs for the ibaPADU-S-IT-2x16 target system must be compiled specifically for WEC (Windows Embedded Compact).

7.4.4.1 Source files and descriptions required

The following source files and descriptions are necessary for creating a DLL. The files and a description are available on request. Please contact the iba support.

Descriptions:

- ❑ Manual on creating a DLL with C++
  (for Windows and ibaPADU-S-IT-2x16)

Files: (please refer to the descriptions for the exact names of the files)

- ❑ Framework file:
  It contains the procedures and the DLL body; the user can add inputs or outputs or modify the procedures, InitEvaluation, Evaluate and ExitEvaluation. Either in C++ or Fortran.
- ❑ Other files depending on the language:
  Assignment of DLL procedures and numbers, interface definition, etc. It is not necessary for the user to make any modifications here.

7.4.4.2 Requirements and Notes

You should take note the following when creating DLLs:

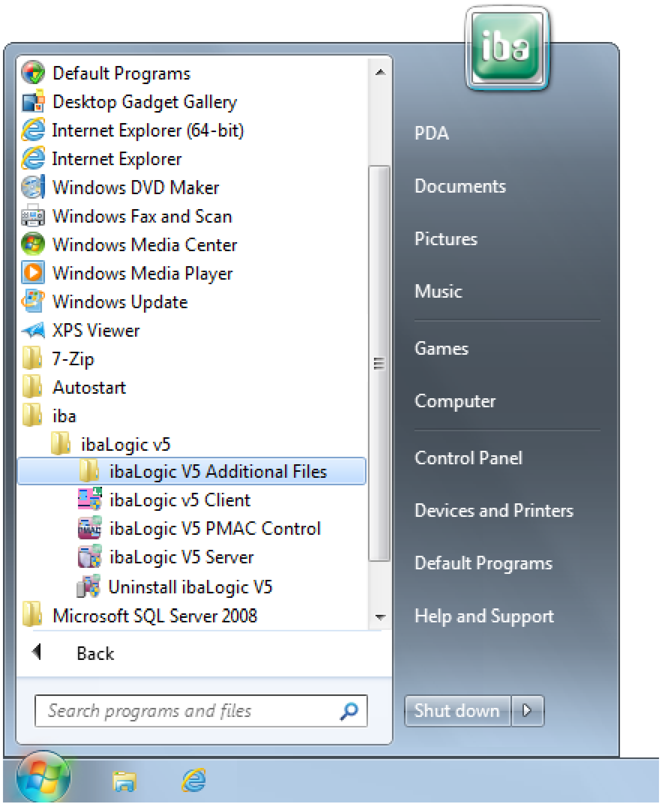
- ❑ The runtime of the DLL increases the runtime of the tasks in which they are called.
- ❑ iba recommends that you remove time-consuming functions to threads.
- ❑ ibaLogic can be started as the executing program to test the DLL.

Important Note

ibaLogic cannot detect and trap programming error in a DLL, which means that such errors can lead to ibaLogic "crashing". Please bear this in mind as a user when creating a DLL.

7.4.4.3 Integrating the DLL into ibaLogic

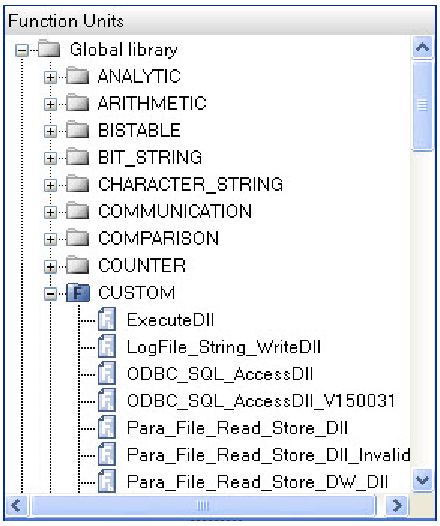
When the DLL has been created, it must be copied to a folder of ibaLogic. (Usually, “C:\...\ibaLogic v5\Server\Dll").
Open the "ibaLogic V5 Additonal Files” folder as shown below to find the directory. The DLL directory can be found in the server directory there.



After the ibaLogic server will have restarted the next time, this DLL will be available as a function block in the "CUSTOM" folder and can be dragged & dropped like any other function unit in a program and integrated in it.

Example

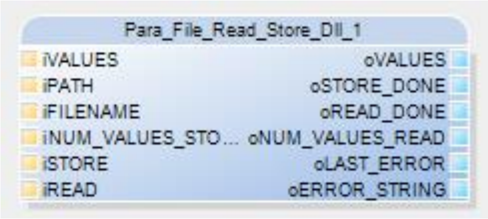
The "Para_File_Read_Store_Dll" has been created and copied to the folder. It is contained in the "CUSTOM" group.



Function Units

- Global library
  - ANALYTIC
  - ARITHMETIC
  - BISTABLE
  - BIT_STRING
  - CHARACTER_STRING
  - COMMUNICATION
  - COMPARISON
  - COUNTER
  - CUSTOM
    - ExecuteDll
    - LogFile_String_WriteDll
    - ODBC_SQL_AccessDll
    - ODBC_SQL_AccessDll_V150031
    - Para_File_Read_Store_Dll
    - Para_File_Read_Store_Dll_Invalid
    - Para_File_Read_Store_DW_Dll

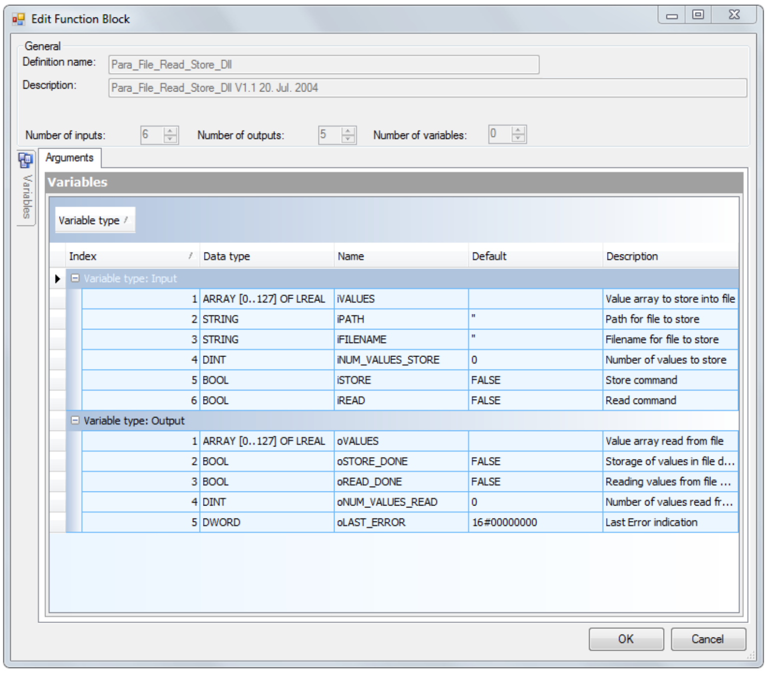
Figure 76: Para_File_Read_Store_Dll in the Function Unit navigator



Para_File_Read_Store_Dll_1

iVALUES	oVALUES
iPATH	oSTORE_DONE
iFILENAME	oREAD_DONE
iNUM_VALUES_STO...	oNUM_VALUES_READ
iSTORE	oLAST_ERROR
iREAD	oERROR_STRING

Figure 77: Para_File_Read_Store_Dll as a function block in the program



Edit Function Block

General

Definition name: Para_File_Read_Store_Dll

Description: Para_File_Read_Store_Dll V1.1 20. Jul. 2004

Number of inputs: 6 Number of outputs: 5 Number of variables: 0

Arguments

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	ARRAY [0..127] OF LREAL	iVALUES		Value array to store into file
2	STRING	iPATH	''	Path for file to store
3	STRING	iFILENAME	''	Filename for file to store
4	DINT	iNUM_VALUES_STORE	0	Number of values to store
5	BOOL	iSTORE	FALSE	Store command
6	BOOL	iREAD	FALSE	Read command
Variable type: Output				
1	ARRAY [0..127] OF LREAL	oVALUES		Value array read from file
2	BOOL	oSTORE_DONE	FALSE	Storage of values in file d...
3	BOOL	oREAD_DONE	FALSE	Reading values from file ...
4	DINT	oNUM_VALUES_READ	0	Number of values read fr...
5	DWORD	oLAST_ERROR	16#00000000	Last Error indication

Figure 78: Para_File_Read_Store_Dll properties

➲ The "Edit Function Block" window is displayed when you double-click on the function block.
You cannot see the code. The inputs and outputs including their descriptions are visible.

7.5 Data types

A data type is assigned to each variable.

In contrast to version V3, ibaLogic-V5 not only supports the elementary data types and arrays, but also composite (structures) and other user-defined data types.

The data types that can be used in ibaLogic can be divided into the following categories:

❑ Standard data types

❑ Composite data types such as ARRAY, STRUCT and ENUM,

❑ Derived data types that are formed from the groups mentioned above.

As a user, you can define your own specific data types of the "composite" or "derived" category.

Note

For more information, please refer to "*Data types*, page 300".

7.5.1 Define Data Type

➲ Click on the "Data Types" button.

The folder is displayed in the navigation area as a tree.

The following folders are created for the **non-elementary** data types in a global library and also under each project of the workgroup:

❑ **Direct derived types**
Standard data type with a fixed default value

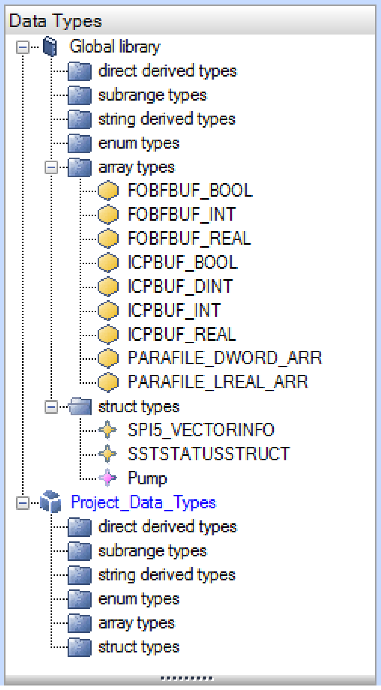
❑ **Sub-range types**
Standard data type with a fixed default value and limited range of values

❑ **String derived types**
String data type with a fixed length and default text.

❑ **Enum types**
Enumerations: Names are defined instead of integer values

❑ **Array types**
Array of elementary data types with a fixed dimension and depth.

❑ **Struct types**
Structure consisting of elementary data types



Data Types
- Global library
  - direct derived types
  - subrange types
  - string derived types
  - enum types
  - array types
    - FOBFBUF_BOOL
    - FOBFBUF_INT
    - FOBFBUF_REAL
    - ICPBUF_BOOL
    - ICPBUF_DINT
    - ICPBUF_INT
    - ICPBUF_REAL
    - PARAFILE_DWORD_ARR
    - PARAFILE_LREAL_ARR
  - struct types
    - SPI5_VECTORINFO
    - SSTSTATUSSTRUCT
    - Pump
- Project_Data_Types
  - direct derived types
  - subrange types
  - string derived types
  - enum types
  - array types
  - struct types

Figure 79: Data Types

The "Array types" and "Struct types" contain data types that have already been predefined by
ibaLogic.
These are:

❑ **FOBFBUF_BOOL / _INT / _DINT / _REAL**
Single-dimensional arrays with 256 elements, usage in "Buffered Mode". For more information, please see section "*Buffered Mode*, page 193".

❑ **ICPBUF_BOOL / _INT / _REAL**
Single-dimensional arrays with 1,024 elements, used for connecting analog inputs in the ibaPADU-S-IT-2x16 platform.
For more information, please see "*ibaPADU-S-IT-2x16 Platform*, page 203".

❑ **PARAFILE_DWORD_ARR / PARAFILE_LREAL_ARR**
Single-dimensional arrays, 128 DWORD, 128 LREAL, used for ParaFileReadStore function unit

❑ **SPI5_VECTORINFO**
Special application

❑ **SSTSTATUSSTRUCT**
Structure for coupling the diagnostics information to the Profibus master card SST.

Note

You can suppress the display of the data types that are predefined and generated automatically if you select the "Tools – Options – General – System" menu and enable the "Suppress Generated Data Types" option.

You can use the data types in the program even if the display is suppressed.

Procedure

You can define a data type in different ways:

❑ Under the project

❑ In the global library

❑ During the creation of a function block

7.5.1.1 Under the project

1. Click in the function tree with the right mouse button on the desired category under the project.
2. Select "New" in the context menu.

7.5.1.2 In the global library

1. Click on the function tree with the right mouse button in the global library on the desired category.
2. Select "New" in the context menu.

7.5.1.3 When creating a Function Unit

The option of creating a new data type is provided in the selection box for the data type of a variable.

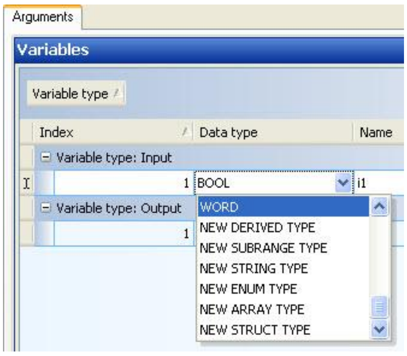


Figure 80: Creating a data type in a function block

Procedure

1. Create a variable with the appropriate data type.
2. Test the data type created to ensure that it is error-free. If the test was successful, confirm the entry with <OK>.

Result

If the syntax is error-free, the data type is created under the category selected.

7.5.2 Modify Data Type

Note

A data type that is already in use cannot be modified.

When you exit the dialog with <OK> or <Accept>, your attention is drawn to the fact that you can create a copy under a different name.

Procedure

1. Click with the right mouse button on the data type.
2. Select "Properties" in the context menu.
3. Modify the parameters.

7.5.3 Delete Data Type

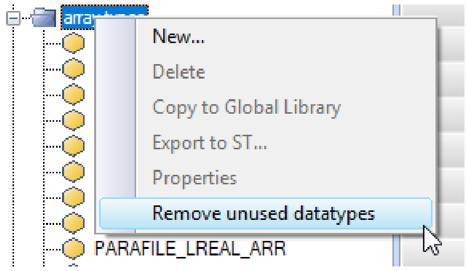
Requirement

You are not using the data type to be deleted.

Procedure

1. Click with the right mouse button on the data type.
2. Select "Delete" in the context menu or press the <Del> function key.

Using the command "Remove unused datatypes" all data types can be removed which are not used.



7.5.4 Manage Data Type

Copy to the global library

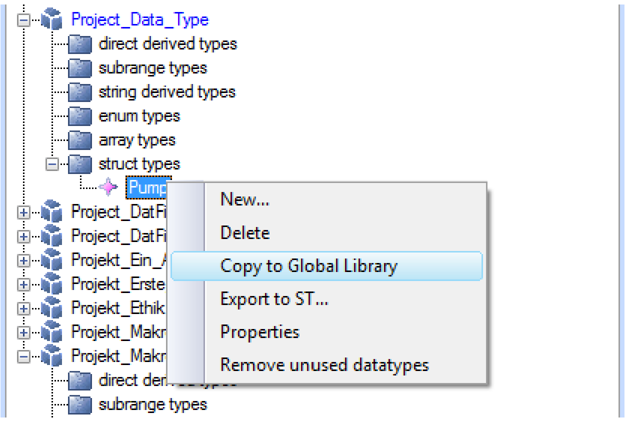
If you need to use a data type, which is defined in one project, also in another workspace, the data type must be copied to the global library.

Note

If you use data types from the global library, they are automatically copied to the project. If you use a data type from another project, this has to be copied to the global library first.

Procedure

1. Click with the right mouse button on the data type under the project.
2. Select "Copy to Global Library" in the context menu.



Note

If you copied an array to the global library and then change the original, you have two arrays with the same name, however, with different contents.
When selecting in the FB, [GLB] is put in front of the array from the global library.

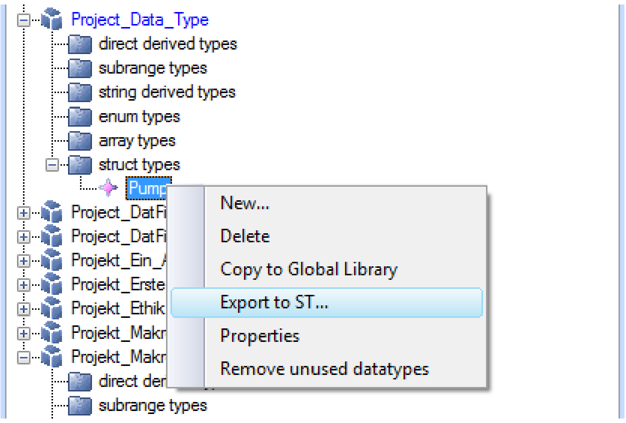
7.5.5 Export Data Type

If a data type, which is defined in another database under the global library or in a project, also needs to be used in another database or in another programming tool, the data type must be exported as a text file.

Procedure

1. Click with the right mouse button on the data type.
2. Select "Export to ST" in the context menu.

   The "Export" dialog box is displayed.



3. Specify the target folder and file name.

7.5.6 Import Data Type

Requirement

ibaLogic is not in the online mode.

Procedure

➲ Click on the "File – Import – Structured Text" menu.

7.5.7 Use Data Type

After a data type has been defined you can use it:

❑ during the creation of a function unit

❑ during the creation of a structure and/or array data type

❑ when creating inputs and outputs

7.5.7.1 During the Creation of a Function Unit

➲ Select the user-defined data type under the
"Data Type" column while creating a variable in the function block editor.

7.5.7.2 During the Creation of a Structure Data Type

➲ Select a user-defined data type in the "Data Type" selection box while creating the structure elements in the data type editor.

Note

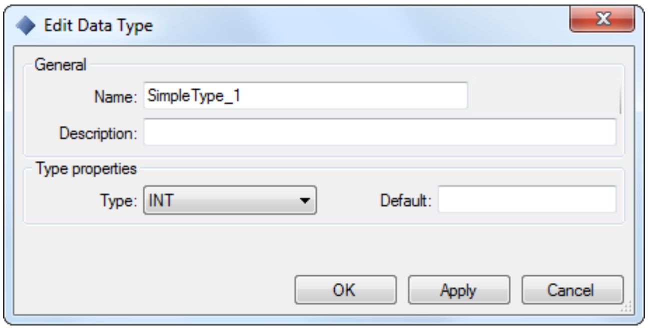
You cannot directly access data types which you defined in a project of another workspace.
For doing so, use the Export/Import function or the global library.

7.5.8 User-defined Data Types

Procedure

➲ Click with the right mouse button on a data type group.

The "Edit data types" dialog box is displayed.



Edit Data Type

General

Name: SimpleType_1

Description:

Type properties

Type: INT

Default:

For all data types, this dialog box consists of:

- "General" section (identical for all data types)
- "Type Properties" section
- "Elements" section

General

- Name*:*
  Name for the user-defined data type. The data type is then available under this name in the selection boxes.
- Description:
  Any text for the description of this type. The description can be seen only here in the definition.

Type properties

- Type:
  Defines the data type.
- Default value:
  Initial value (Preset value)

7.5.8.1 DIRECT DERIVED TYPE Group

This is to define an elementary data type for which a new name and a default value can be specified.

With this, for example, constants such as the number "Pi" can be defined with the LREAL data type.

7.5.8.2 SUBRANGE TYPE Group

This is an integer data type with a limited range of values and a default value.

You can use it, for example, to define indices for arrays having a specific depth.

Important Note

This data type is **not** limited.

This data type is only checked in case of direct assignments during compilation, there is no runtime checking as to whether the range is exceeded.

7.5.8.3 STRING DERIVED TYPE Group

This is a string data type having a limited length.

You can use it to define text strings having a fixed initial value, for example, for error messages.

7.5.8.4 ENUM TYPE Group

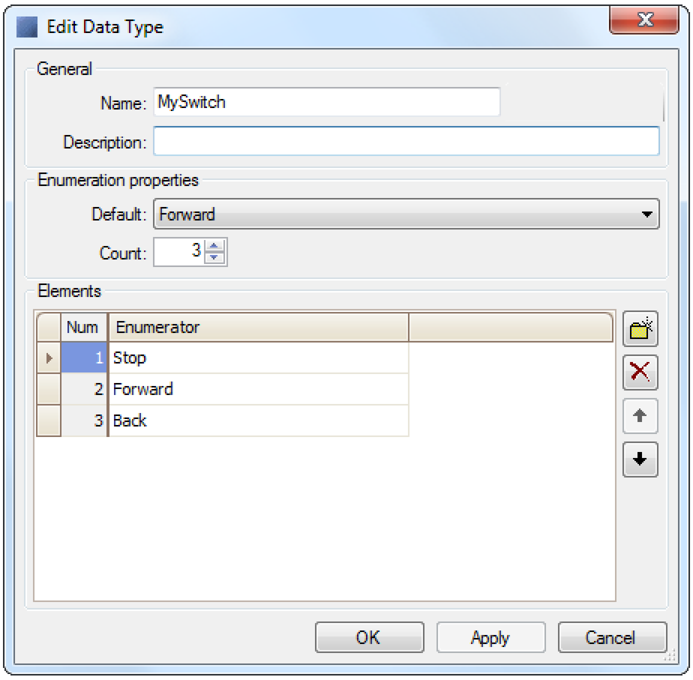
A data type of the category ENUM TYPE is an enumerator.

The data type is used to designate the values of a variable symbolically, e. g. the position of a switch.

Example: Data type "Switch"

You would like to create a data type "Switch", which has the 3 positions, FORWARD, STOP and BACK.

To do this, you need to define the ENUM TYPE Switch, with number 3 and the switch positions as enumerators.



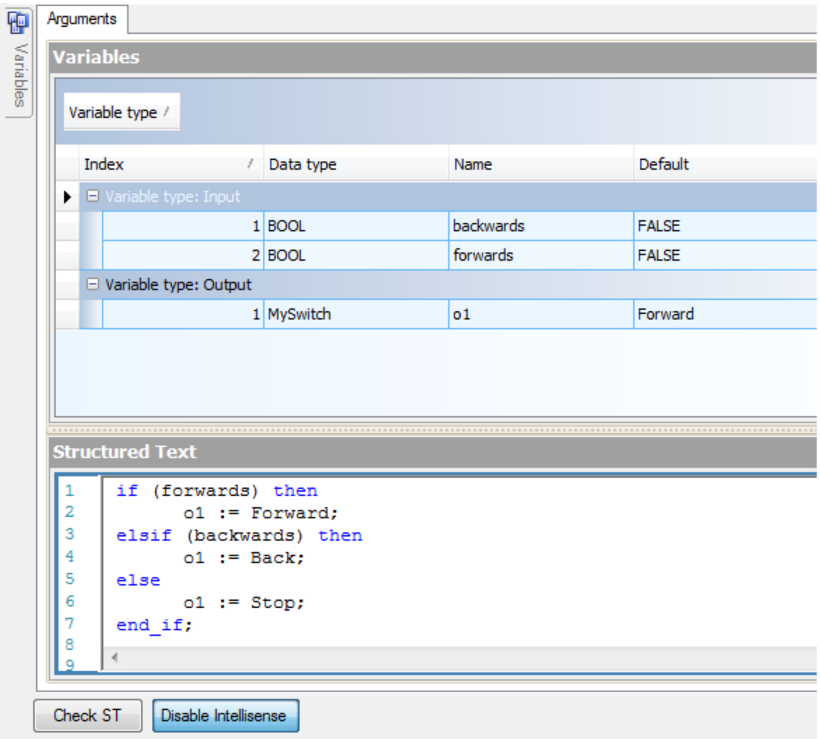
Name: MySwitch
Description:
Default: Forward
Count: 3

Num	Enumerator
1	Stop
2	Forward
3	Back

Figure 81: „Edit data types" dialog box

Please note that you access the individual enumerator values using "Enumerator" in "Structured Text".

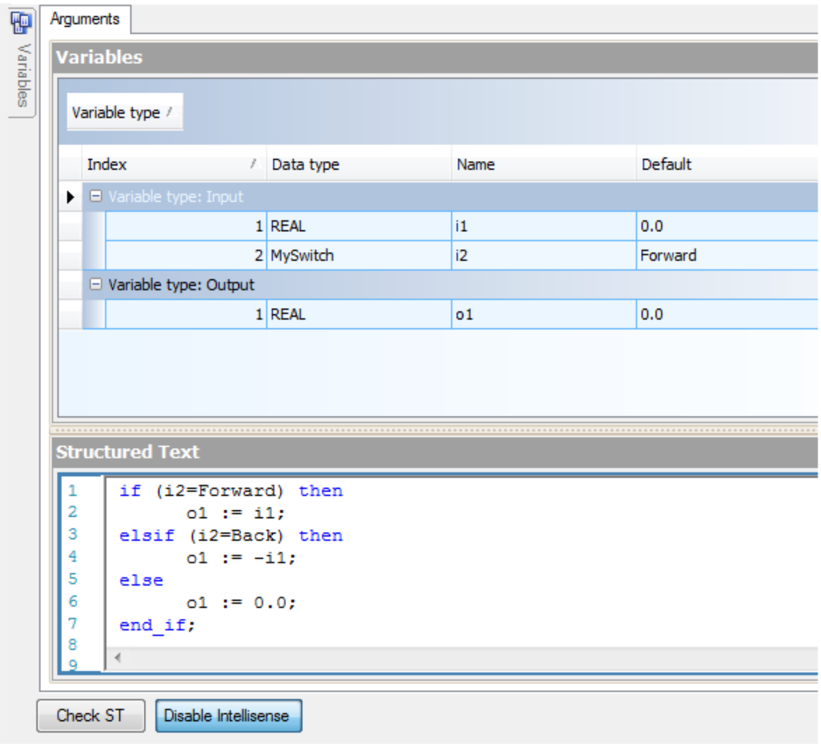
Assign and retrieve values:



Index	Data type	Name	Default
Variable type: Input			
1	BOOL	backwards	FALSE
2	BOOL	forwards	FALSE
Variable type: Output			
1	MySwitch	o1	Forward

```
if (forwards) then
        o1 := Forward;
elsif (backwards) then
        o1 := Back;
else
        o1 := Stop;
end_if;
```

Figure 82: Variables Editor 1



Index	Data type	Name	Default
Variable type: Input			
1	REAL	i1	0.0
2	MySwitch	i2	Forward
Variable type: Output			
1	REAL	o1	0.0

```
if (i2=Forward) then
      o1 := i1;
elsif (i2=Back) then
      o1 := -i1;
else
      o1 := 0.0;
end_if;
```

Figure 83: Variables Editor 2

Note

Please note that you cannot assign integer values to the enumerators.

Exception:
An OPC connector is declared as type Enum and read or written externally. In this case, the OPC Client writes or reads the enumerator number as an integer.

7.5.8.5 ARRAY TYPE Group

Arrays are single-dimensional or multi-dimensional fields. All elements of an array have the same data type. However, this is not restricted to the elementary data types, but you can also form arrays of user-defined data types, structures, strings or arrays.

Example: 2-dimensional integer array

Parameter	Explanation
Type	Base type of the array elements
Number	Number of dimensions

Default value	Default values of the array
	Examples
	`1-dim-Array: [1.0,2.0,3.0]`
	`2-dim-Array: [[1.0,2.0,3.0],[4.0,5.0,6.0]]`

Lower / Upper limit	The value range of the element index determines the depth of the individual dimensions. Maximum value: 0 to 32,766

Access to the elements of an array in Structured Text:

i1 is a variable of array type. *o1, o2...* are variables of element type;

❑ 1-dim- Array:	`o1 := i1[0];`	
❑ 2-dim- Array:	`o1 := i1[0.0];`	`(* 1st element of the 1st dim *)`
	`o2 := i1[0,1];`	`(* 2nd element of the 1st dim *)`
❑ array_of_array:	`o1 := i1[0][0];`	`(* 1st elem. of the 1st array *)`
	`o2 := i1[0][1];`	`(* 2nd elem. of the 1st array *)`
	`o3 := i1[1][0];`	`(* 1st element of the 2nd array *)`
❑ array_of_struct:	`o1 := i1[0];`	`(* 1st structure of the array *)`
	`o2 := i1[0].elem;`	`(* elem. of the 1st structure *)`

Note on Structured Text

The indices of arrays can be only variables having "Int" data type.

7.5.8.6 STRUCT TYPE Group

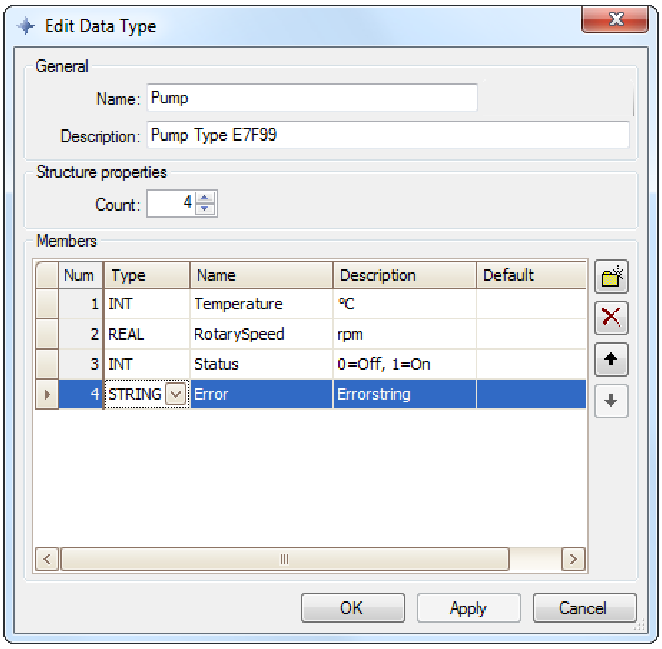
In contrast to arrays, you can group variables having different data types under a structure. You have to define the elements separately, and while doing so, you can use all data types defined so far, including the user-defined data types and arrays.

You can define a name, description and default value for each element of the structure.

Example: Pump

You need a "Pump" data type for the pump "Type E7F99" with the properties "temperature", "speed", "state" and "error".

For this purpose, you define the "Pump" data type with the description "Pump Type E7F99" and the number 4. "Elements" defines the associated properties.



Edit Data Type

General

Name: Pump

Description: Pump Type E7F99

Structure properties

Count: 4

Members

Num	Type	Name	Description	Default
1	INT	Temperature	°C	
2	REAL	RotarySpeed	rpm	
3	INT	Status	0=Off, 1=On	
4	STRING	Error	Errorstring	

Figure 84: "Edit data types" dialog box

Parameter	Explanation
Type	Base type (even user-defined data types are allowed).
Name	Name of the element.
Description	Personal explanation of the data type.
you select the settings	Initialization to a default value (Preset).

Access to the elements of the structure in Structured Text:

```
(*o1 is a variable of structure type pump,   i1, i2... are variables of element type; *)


         o1.Temperature := i1;;          (* of INT data type *)
         o1.Speed := i2;                 (* of REAL data type *)

(*v1 is a variable of Pump structure type;*)

if       (v1.Temperature > 80 ) then
         v1.Status:= 99;
         v1.Error:= 'Temp. too high';
else
         v1.Status:= 0;
         v1.Error := 'No error';
end_if;
```

Figure 85: Structure in Structured Text

8 Program Elements

A graphical ibaLogic program contains the following elements:

- ❑ Functions and Function Blocks
- ❑ Inputs and Outputs, Off Task Connectors
- ❑ Links (Connections), Intra Page Connectors
- ❑ Converters, splitters or joiners added automatically
- ❑ Comments

8.1 Create Program Element

You can create all elements that a graphical ibaLogic program can contain.

Procedure

1. Open the context menu by clicking with the right mouse button on a free area in the programming field.
2. Select "New..." in the context menu.
3. Select the desired program element.

8.2 Mark program elements

You can mark individual or multiple program elements in the following manner.

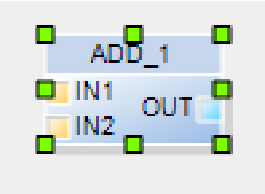
Procedure

1. Select the desired element by clicking with the left mouse button (single selection).
2. Select the desired elements by clicking with the left mouse button and pressing the <Shift> or <Ctrl> simultaneously (multiple selection).
3. Drag a rectangle (Lasso) over one or more elements to be selected by clicking on the left mouse button.
   In doing so, existing connection lines and converters between the function units, if any, are also marked.
4. Select all elements by pressing the <Ctrl> + <A> keys.

Result

The elements marked are displayed with green, blue or gray dots.

When marking several elements, one element is always green. This is the reference point of the grouping.

	One element is selected.

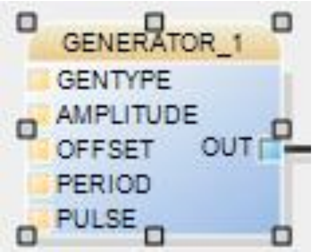
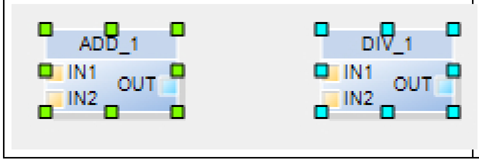
 GENERATOR_1 GENTYPE AMPLITUDE OFFSET PERIOD PULSE OUT	One element is selected, but the focus lies outside the editor, e.g. on the tree structure on the left.
	Many elements are selected. The green element is always the main element of the group marked.

Figure 86: Selected element

8.3 Move Program Element

You can move the function units (and connections) already marked by keeping the left mouse button pressed or using the arrow keys.

Procedure

➲ Move one or more elements selected by keeping the left mouse button pressed.

Remarks

This applies to lines with limitations.

The following applies when using the arrow keys: If nothing is selected, the arrows are used to move the plan. If one or more elements are selected, then these will be moved with the arrow keys. If you additionally hold down the <Ctrl> key, movement is done with a fine resolution.

8.4 Align Program Elements along an Edge

All program elements can be aligned along an edge. You can use this for obtaining a clear layout.

Procedure

1. Mark the elements to be aligned (function units, Intra-page connectors and Off-task connectors).
2. Select the desired function the "Function Diagram - Align".

Remarks

This is not applicable to inputs / outputs and lines.

8.5 Copy Program Element

You can copy individual or multiple program elements.

Procedure

1. Mark the elements to be copied (function units, Intra-page connectors and Off-task connectors).
2. Press the key combination <Ctrl> + <C> to copy the elements selected to the clipboard.

3. Press the key combination <Ctrl> + <V> to copy the elements from the clipboard into the programming field.

Tip

Instead of the key combination, you can also select "Copy" and "Add" in the context menu.

If you have selected multiple function units, the connecting lines between them also get copied.

This is not applicable to inputs and outputs and lines marked individually.

8.6 Delete Program Element

You can remove individual or multiple program elements.

Procedure

1. Mark the elements to be deleted (function units, Intra-page connectors and Off-task connectors).
2. Press the <Del> key.

Remarks

Instead of the <Del> key, you can also select "Remove" in the context menu.

8.7 Generate Input / Output Variables

Prerequisite

You have selected the "Inputs - Outputs" button.

Procedure

➲ Drag an input or output variable at any position in the left or right input or output border.

Note

In a **program,** an input and output can be created **only once**.
In a **project,** an input can be used **several times**. An **output** can be used only **once**.

8.8 Graphical Connections

In graphical programming, a graphical connection is used to transfer the results of one function to another.

There are 3 different forms for this:

❑ Direct connectors

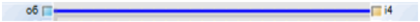
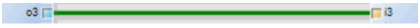
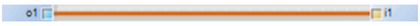
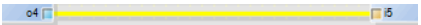
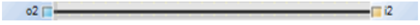
❑ Intra-page connectors

❑ Off-task connectors

8.8.1 Direct Connectors

8.8.1.1 Types of connection lines

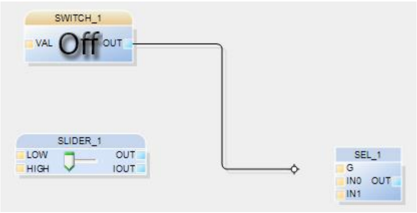
ibaLogic uses line types of different colors that represent different groups of data types.

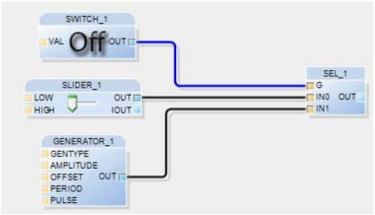
Line type	Explanation
	Binary connectors are displayed according to their status, red (TRUE) or blue (FALSE).
	Arrays are displayed using green lines. Only arrays having identical length and data type can be connected with one another.
	Structures are displayed in orange color.
	Enum types are displayed in yellow color.
	All other elementary data types are marked with black connectors (e.g. INT, REAL...)

8.8.1.2 Create Direct Connector

Procedure

1. Click with the mouse on the output connector of a function unit.
2. Keep the left mouse button pressed and drag a connector to the input connector of a function unit.





Remark

If the result of one function unit is used in multiple function units, generate a branch by dragging a line from one input connector to another existing one.

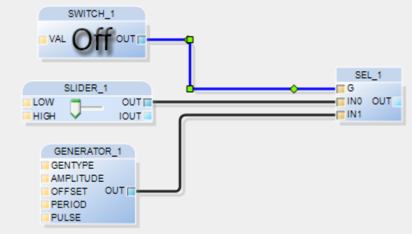
Near a connectable connector or connectable line, the mouse cursor jumps to the connector or the connecting line (Magnetic effect).

8.8.1.3 Modify Direct Connectors

Procedure

1. Mark the connector that you wish to modify.

   The marking is displayed by small green squares and diamonds.



2. Modify the line by moving the green squares with the mouse.
3. Click with the mouse on the green diamond on the connecting line in order to wire the line connection.
4. Drag the end to a blank area, which deletes the connecting line.
5. Drag the end to another connector. The connecting line is reconnected.

Note

Connecting lines that you have arranged manually are re-evaluated by the auto-router when the associated function unit is moved. Your modifications are rejected as a result.

8.8.1.4 Highlight connectors

Selected connection lines will be hightlighted with yellow background. This makes it easier to follow the line paths.

A single click on a line activates the hightlighting to the next connecting point. This could be a line connector or any other connector.

By pressing the ALT-Key, the whole line network including line connectors and IPCs (IntraPage connectors) will be hightligthed.

Examples:

Single click on a line leads to the yellow highlighting to the next connector point:

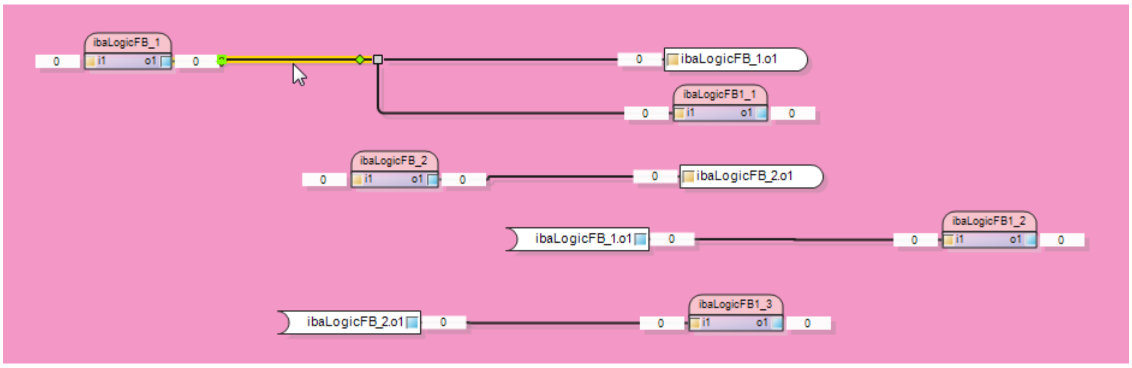


Figure 87: Highlight the line

Clicking with the <ALT> key depressed leads to highlighting of the complete network beyond nodes and IPCs..

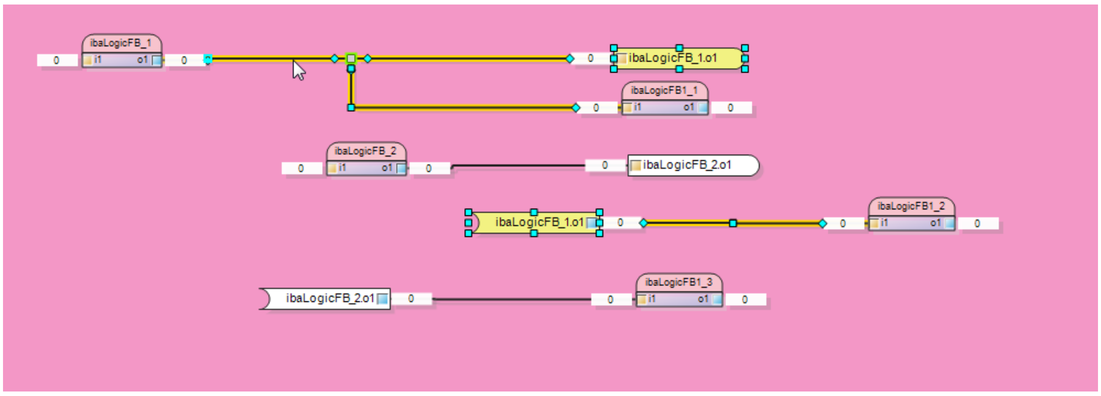


Figure 88: Highlight the complete network

8.8.2 Intra-Page Connectors

An intra-page connector (IPC) merely represents a drawing simplification. In the process, the IPC replaces a connecting line.

This is recommended when several objects on one page need to be connected or "long" connections are required across multiple pages. The IPC is not a programming object, but merely acts as a line substitute.

The IPC can - be used as direct connectors - only within a program or macro level. You cannot have connections from a macro to the call level. You must define inputs and outputs in the macro block for this purpose.

8.8.2.1 Create Intra-Page Connectors

Create Intra-Page connectors as line substitutes.

Prerequisite

You can generate an IPC at an input connector only if an "IPC Source" has been defined earlier.

Procedure

➲ Press the <Ctrl> button and simultaneously drag a connecting line from one output connector to a free location in the programming field.

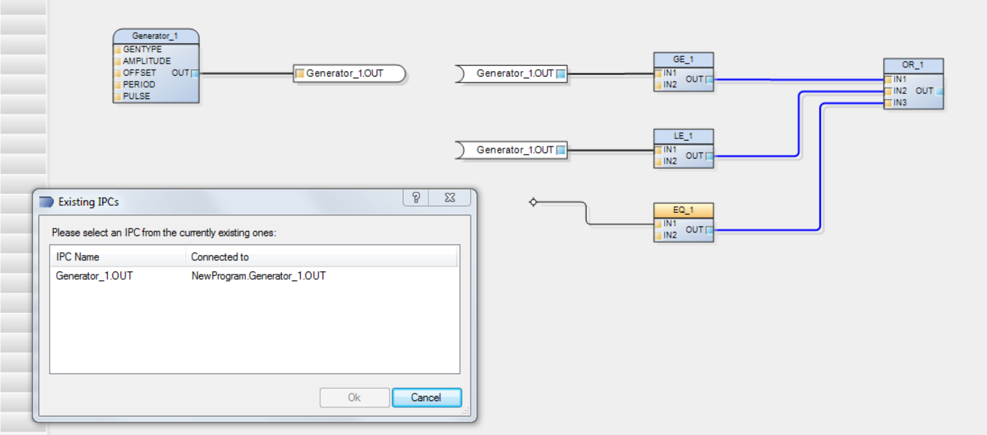
➲ Menu procedure similar to creating off-task connectors. However, major modification.

- Create IPC source
- Connect IPC (as described here)
- Connect IPC via menu

Remark

To connect an IPC to an input, follow the same procedure.
Hold down the <Ctrl> key and drag the line from an input connector to a free area in the

program field. Subsequently, the "Existing IPCs" dialog opens. Select the corresponding IPC and quit the dialog by clicking <OK>.



Please select an IPC from the currently existing ones:

IPC Name	Connected to
Generator_1.OUT	NewProgram.Generator_1.OUT

Figure 89: Properties window

8.8.2.2 Modify IPC Names

ibaLogic creates a name automatically, consisting of
"function unit instance name.connector name". This name can be changed.

Procedure

1. Double click on the IPC source.
   The "Edit IPC " dialog box is displayed.
2. Assign a name and a comment to the IPC source. Assign meaningful names.

Result

The modification is accepted automatically in all "IPC Targets" connected. "IPC Targets" cannot be directly modified.

8.8.2.3 Track IPC

Load the corresponding program page of the selected IPC.

Procedure

1. Right-click on an IPC.
2. Select "Go to ->" in the context menu.
   You can then see the connected generator (output) and all connected consumers (inputs).
3. Click on any connection displayed.

Result

The appropriate program page is loaded and the IPC is marked.

Context menu	Explanation
01. -> Generator_1.OUT	Generator
02.°Generator_1.OUT ->	Consumer (marked IPC)
03. Generator_1.OUT ->	Consumer

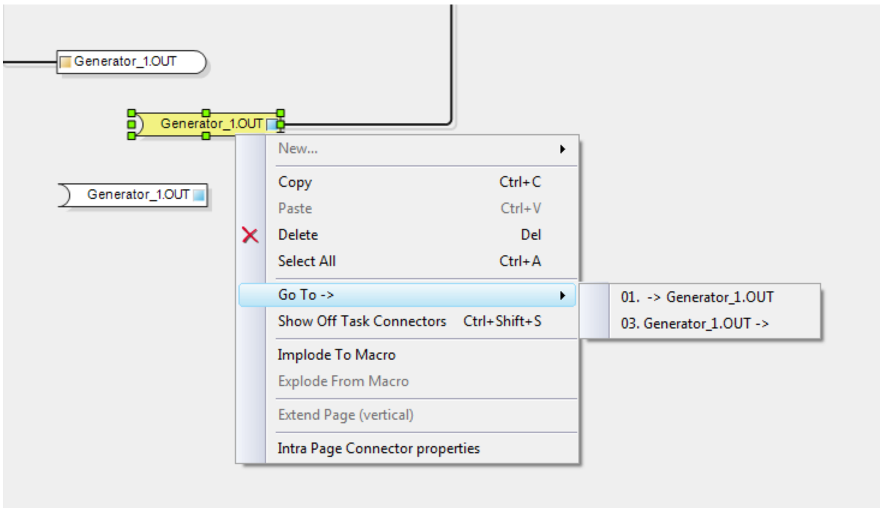


Figure 90: Track IPCs

8.8.3 Off-task connectors

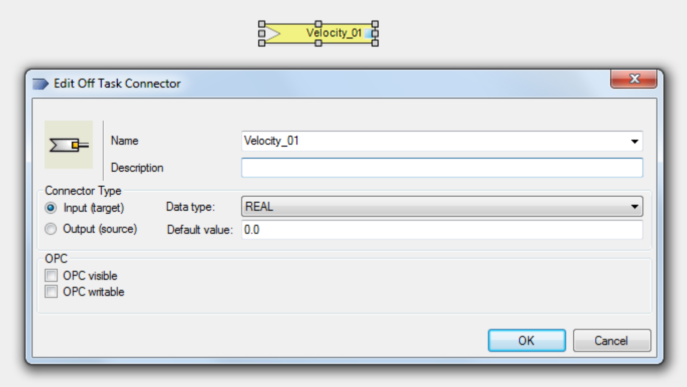
Off-task connectors (OTC) are used as program-independent connecting elements are always required when there is communication between programs.

In addition, OTCs can be set as read-enabled and write-enabled for OPC Clients.

8.8.3.1 Create Off-Task Connectors

Procedure

1. Position the mouse pointer at a free location in the programming field.
2. Open the context menu by clicking on the right mouse button.
3. Select "New… - New Off-task connector".
The "Edit Off-task connector" dialog box is displayed.



Name: Velocity_01
Description:
Connector Type
● Input (target)
○ Output (source)
Data type: REAL
Default value: 0.0
OPC
□ OPC visible
□ OPC writable

4. Assign the parameters required.

Note

The OTC name must comply with the IEC naming convention.
See "*Naming conventions*, page 299".

Establish a program-independent connection

Procedure

Method 1:

1. First generate the output OTC (Source) by filling up the dialog box.
2. Copy the output OTC.
3. Add the output OTC in the target program.
   By doing so, the parameters are accepted but the direction is reversed.

Method 2:

1. Create an OTC in the target program.
2. Select the name of the associated output OTC from the selection box.
   In doing so the other parameters get accepted.
3. You must set the direction to "Input".

OPC Properties

You can specify the following OPC properties to the OPC.

Selection boxes OPC Properties	Explanation
OPC visible	Specifies whether this connector is visible in the OPC name space.
OPC write-enabled	Specifies whether an OPC Client should write to this connector.

Further information, please refer to "*Setting the OPC DA Variable Parameters*, page 208".

Rules for creating OTCs

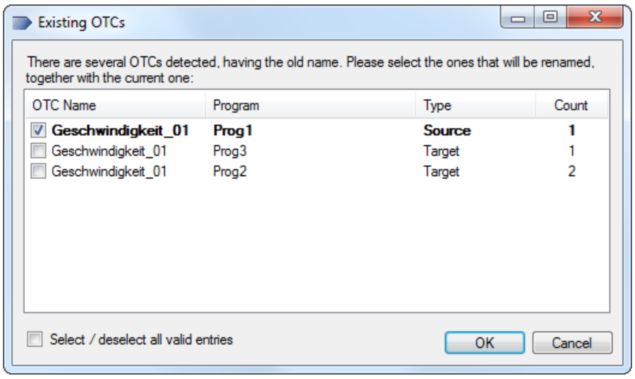
- ❑ An output OTC must be unique in the project.
- ❑ Multiple input OTCs can be created for an output OTC. You can do this even within a program, but not in the program in which the output OTC is placed.
- ❑ An input OTC can have only one data source, either OPC enabled or an associated output OTC.
- ❑ If you create an output OTC with the name of the input OTC, this input OTC is no longer write-enabled for the OPC.
- ❑ An input OTC that does not have any data source can be used as a constant / parameter.

8.8.3.2 Rename OTC

Procedure

1. Select the OTC that is to be renamed.
2. Open the OTC properties by means of the context menu or by double-clicking on OTC.
3. Change the name of the OTC and leave the properties dialog by clicking <OK>
   If the OTC already has connected targets, the "Existing OTCs" dialog box is displayed.

In this screen, it can be determined whether all or individual connected OTCs are to be renamed. In this manner, for example, you can assign the right name to an OTC having an incorrect name by correcting the name.



Existing OTCs

There are several OTCs detected, having the old name. Please select the ones that will be renamed, together with the current one:

OTC Name	Program	Type	Count
☑ **Geschwindigkeit_01**	**Prog1**	**Source**	**1**
□ Geschwindigkeit_01	Prog3	Target	1
□ Geschwindigkeit_01	Prog2	Target	2

□ Select / deselect all valid entries

After quitting the dialog by clicking <OK>, all selected OTCs are renamed.

Remark

If a target is present several times in a program, the number can be seen from the COUNT column.

8.8.3.3 Track OTCs

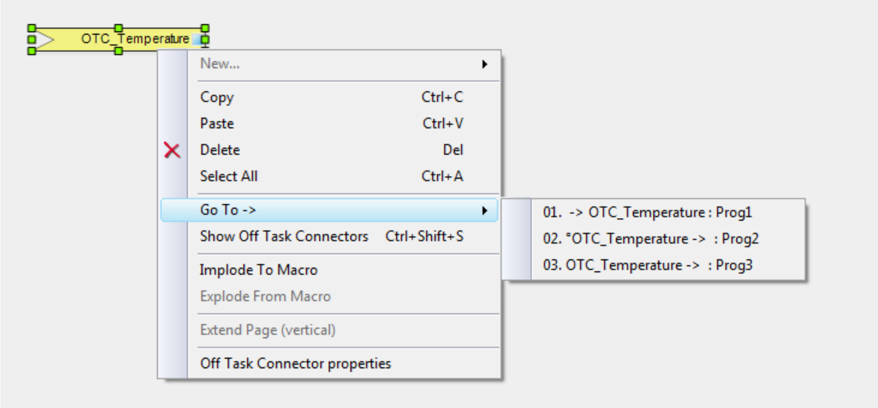
Procedure

1. Click with the right mouse button on an OTC.
2. Select "Go To ->" in the context menu.
   You can then see the programs in which the OTC is generated and used.
3. Click on any connection displayed.

Result

The appropriate program page is loaded and the OTC is marked.

Context menu	Explanation
01. -> OTC_Temperature: Prog1	Generator
02.°OTC_Temperature -> : Prog2	Consumer (marked OTC)
03. OTC_Temperature -> : Prog3	Consumer



OTC_Temperature
New...
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Select All Ctrl+A
Go To ->
Show Off Task Connectors Ctrl+Shift+S
Implode To Macro
Explode From Macro
Extend Page (vertical)
Off Task Connector properties
01. -> OTC_Temperature : Prog1
02. °OTC_Temperature -> : Prog2
03. OTC_Temperature -> : Prog3

Figure 91: Track OTCs

8.8.3.4 List of all OTCs

Display of all defined OTCs in the project.

Procedure

1. Position the mouse pointer at a free location in the programming field.
2. Open the context menu.
3. Select "Display Off-Task Connectors"*.*
   The dialog box that contains the list of all OTCs defined and sorted in alphabetical order is displayed.
   Navigate within the list by entering the starting alphabet or by double clicking on an OTC.
   This displays a list of all OTCs defined in the project.

Remark

You can also call this function via the
"Function diagram - Display Off-Task Connectors" menu option or the key combination <Ctrl>+<Shift>+<S>.

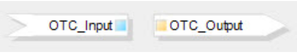
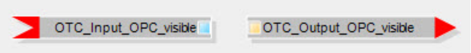
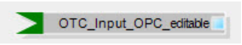
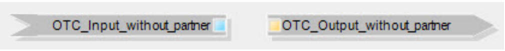
Note

You can leave the dialog box open and continue working with the program. The list is updated automatically when OTCs are created or removed. The dialog box can be positioned wherever desired and can also be docked to the border of the programming window.

A filter function allows searching for OTC names.

8.8.3.5 Display

The various properties are identified in color for better orientation:

Display of the OPCs	Description
 OTC_Input OTC_Output	Displays inputs and outputs that are connected
 OTC_Input_OPC_visible OTC_Output_OPC_visible	Displays inputs and outputs that are visible to the OPC, write-enabled and read-enabled
 OTC_Input_OPC_editable	Displays an input that is read-enabled and write-enabled for the OPC
 OTC_Input_without_partner OTC_Output_without_partner	Displays inputs and outputs that are not connected

8.9 Converters, splitters, joiners

8.9.1 Converter

Automatic Addition of the Data Type Conversion

In conventional CFC editors you can connect interfaces that are not of the same data type. ibaLogic adds a converter automatically if a meaningful conversion is possible.

Note

To use detailed converters directly displaying the conversion, disable the "Iconic Display of the Converter" function in the options.

see: "Tools" - "Options" - [Editors] - [Diagram]

This automatic converter is displayed with reduced size in order to save space in the programming field.
When you go over it with the mouse pointer, it displays the conversion hidden below it as a tooltip.

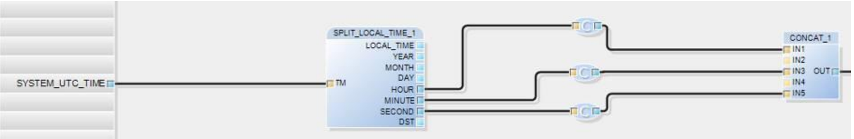


Figure 92: Data type conversion

Note

Function units having non-typed connections get a data type only when a connector is put in place. This data type is then accepted for all connections and remains when you remove the last connection again.

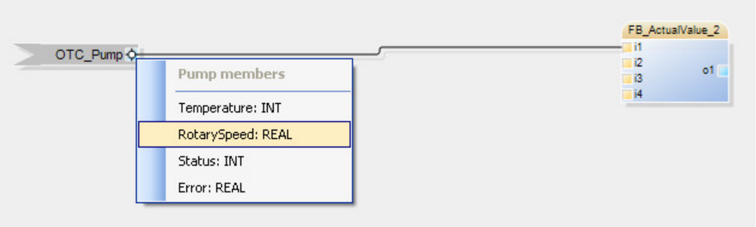
If you wish to connect two non-typed connections a dialog box opens in which you can select a permissible data type.

8.9.2 Splitter

You would like to remove elements from a data structure for other evaluations. Using conventional methods, you have to create a user-defined function block in which you use structured text to assign the structure of the output connectors to individual elements. ibaLogic automatically creates this block, known as splitter, for you.

Procedure

1. Drag a connecting line between one input connector to a structure output connector of a function unit or an OTC input.
   A selection box pops up.
2. Select a structure element.



Remarks

In this manner, you can access other structure elements.

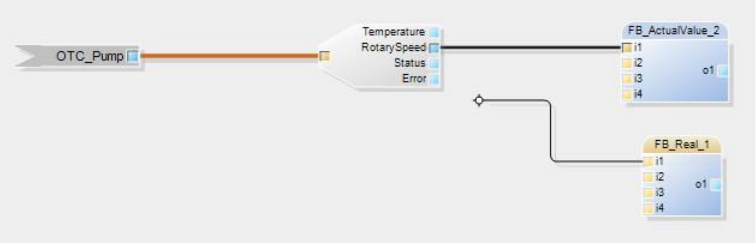


Figure 93: Splitter

8.9.3 Joiner

If you try to connect an output connector with the input connector of a structure, ibaLogic provides a menu where you can select one of the structure elements. Based on this, ibaLogic adds a joiner block to whose inputs you can connect other signals.

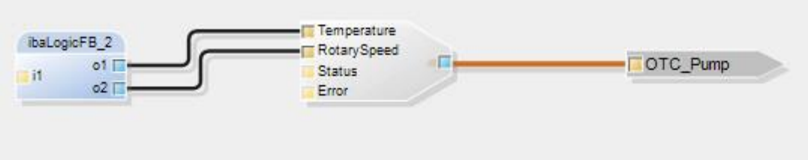
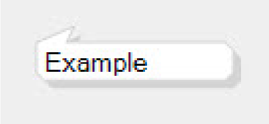


Figure 94: Joiner

8.10 Comments

Comments are graphical elements that you can add at any free position in the programming field. You can cover connection lines. These are visible through a transparent comment field.

The comment field has a pointer that can be docked to the function to be described.



Procedure

1. Position the mouse pointer at a free location in the programming field.

2. Open the context menu.

3. Select "New… - New comment*".*
The font type for comments can also be set under
"Tools - Options - Editors - Diagram".

9 PMAC Runtime System

9.1 Overview of online and offline mode

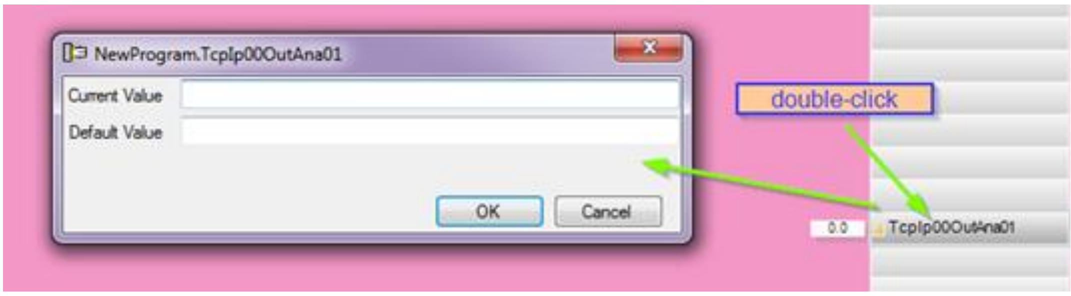
The following menus or icons are available in the menu for operating the runtime system:

❑ Start	(“Evaluation” menu and button in the toolbar)
❑ Stop	(“Evaluation” menu and button in the toolbar)
❑ Store Program on Target	(“Evaluation” menu)
❑ Delete stored program from Target	(“Evaluation” menu)

9.1.1 Hardware outputs support default values

It is possible to set the current value and default values of not linked hardware output signals.

A double-click on the output connector will show the following dialog:



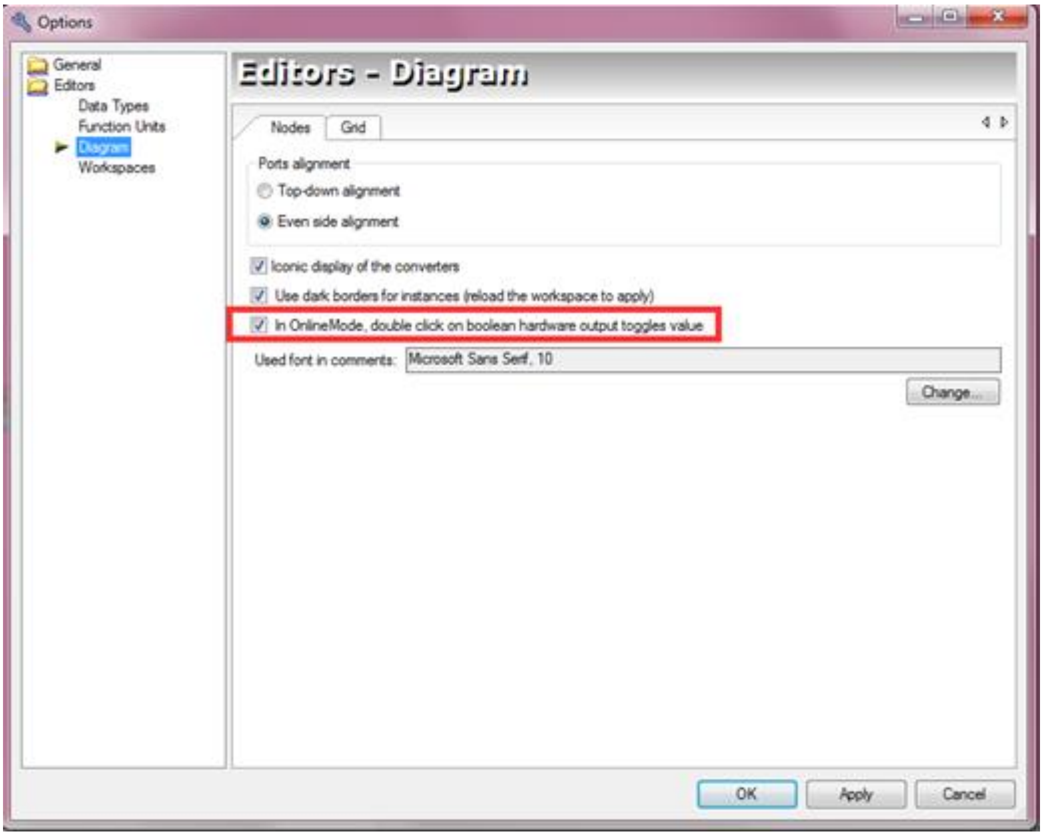
The current value and also the default value (value at start of evaluation) can be set.

When a link to the hardware output signal is added, then these values will be overriden.

9.1.2 Toggle unconnected boolean outputs

A double click on an unconnected boolean output toggles its value.

This must be enabled in the options dialog:



Ports alignment

○ Top-down alignment

● Even side alignment

☑ Iconic display of the converters

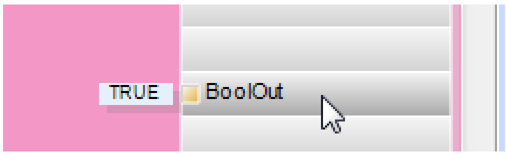
☑ Use dark borders for instances (reload the workspace to apply)

☑ In OnlineMode, double click on boolean hardware output toggles value

Used font in comments: Microsoft Sans Serif, 10

Figure 95: Enable toggle option

Example: A double click in the connector field changes the value.



Note

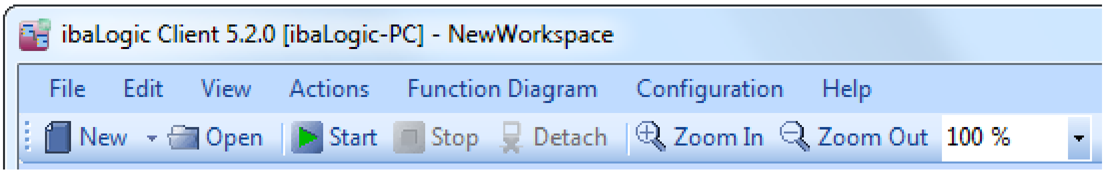
To change the default value of this boolean output in online mode the option must be disabled first.

9.2 Start Runtime System

In contrast to conventional automation systems, the steps of "Compilation" and "Loading" take place automatically in the background.

Procedure

1. Click on the <Start> button in the toolbar.



2. Confirm the "Start Evaluation..." dialog with "Yes".

Note

This request can be disabled in the ibaLogic options to start the evaluation in the development and test environment by pressing the <Start> button or F5.

To disable the query, open the options with "Tools" - "Options" and enable the "Program: Start evaluation" option under "General" - "Messages" - "Confirmations".

Result

The following actions are performed:

- □ The project is compiled.
- □ The project is transferred to the PMAC.
- □ Program execution commences.
- □ The evaluation time is displayed in the program window toolbar.
- □ All value pads are displayed.
- □ The value pads in the visible region are provided with current values.
- □ The background color of the programming field in the client changes to pink.
- □ The program is now in online mode.

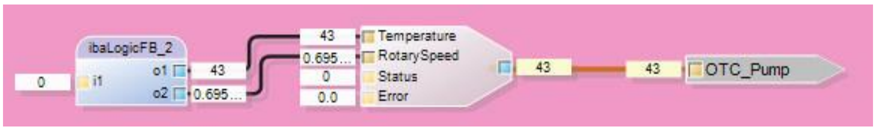


Figure 96: Online mode

Errors during compilation, loading, etc. are displayed in the event window. By default, this is located below the programming field. It can, however, be concealed or placed at any position desired and docked there.

Tip

A special highlight of ibaLogic is the fact that you can (almost) carry out the entire programming in the online mode.

Exceptions:

- Configuring the platform
- Configuring the I/Os
- Importing programs / function units / data types
- Configuring the DAT_FILE_WRITE function block

Another feature:

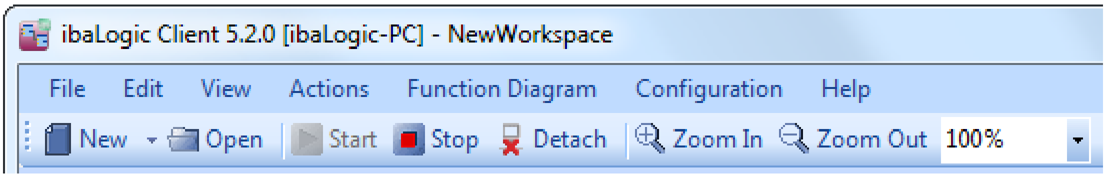
You can end the client in the online mode without stopping the PMAC. When you restart the client, it connects with the PMAC immediately in the online mode.

This is particularly useful when the PMAC is running on another platform (another PC or ibaPADU-S-IT-2x16). You can then shut down the ibaLogic computer and, in fact, remove it, while the PMAC continues to run. After rebooting and starting the server and client, the client reconnects automatically with the PMAC running in the online mode.

9.3 Stop the Runtime system

Procedure

1. Click on the <Stop> button in the toolbar.



2. Confirm the "Stop Evaluation..." dialog with "Yes".

Note

This request can be disabled in the ibaLogic options to stop the evaluation in the development and test environment by pressing the <Stop> button or <Shift>+<F5>.

To disable the request, open the options with "Tools - Options" and enable the "Program: Stop evaluation" option under "General - Messages - Confirmations".

Result

The following actions are performed with <Stop>:

- Program execution (PMAC) terminates.
- The background color of the programming field in the client changes to gray.
- The value pads are concealed.
- The program is now in the offline mode.

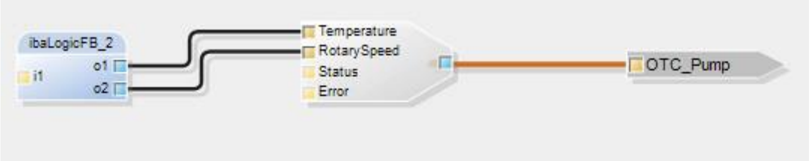


Figure 97: Offline mode

9.4 Runtime System – Autostart

9.4.1 Storing program on PMAC

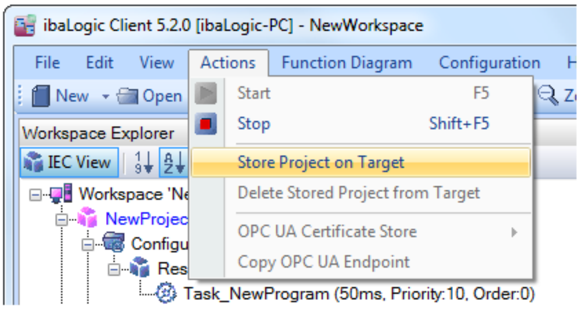
If you wish to start the platform with an executable program, this program must first be saved in the PMAC.

Prerequisite

- ibaLogic is in online mode.
- Autostart is enabled.

Procedure

➲ Select “Evaluation – Store Project on Target” in the main menu.



Result

The project is saved to the platform. A file is physically generated. The PMAC will find this file on the platform on startup and execute it.

For more information, please see “*Activate Autostart Server*, page 43".

Note

This function can also be used for quickly going online in case of large projects. Since a normal START executes the complete compilation process, this might take a while. A direct start of the project through the PMAC option "Execute a project at start-up, if available" directly executes the loaded image (=translated project on PMAC) without having to translate it.

Note

Please note that any subsequent program modification must be saved explicitly in the PMAC.

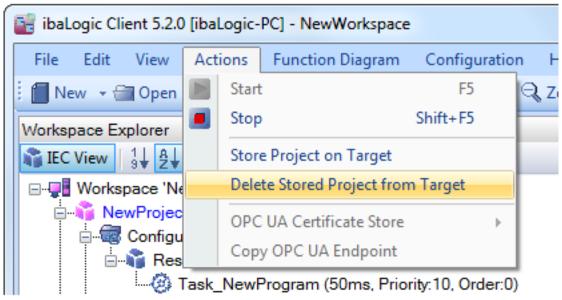
In order to prevent automatic startup of the PMAC, you can first delete the platform memory or modify the autostart options.

9.4.2 Delete Program on the PMAC

This deletes the image file generated earlier physically with the command "Store Program on Target".

Procedure

➲ Select "Evaluation - Delete stored project from target" in the main menu.



Result

The PMAC memory, i. e. the image file created is deleted.

10 Platforms

Before you start configuring the interfaces to the peripherals or to other systems, you have to have to set up the base hardware on which the ibaLogic runtime system (PMAC) needs to run.

At present, there are two device classes available for the platform:

Windows PC

The PMAC runs on a Windows PC on which the other ibaLogic components are running. For more information, please refer to "*Operating modes*, page 29".

The link to decentralized peripherals and to other systems is established using PCI cards here.

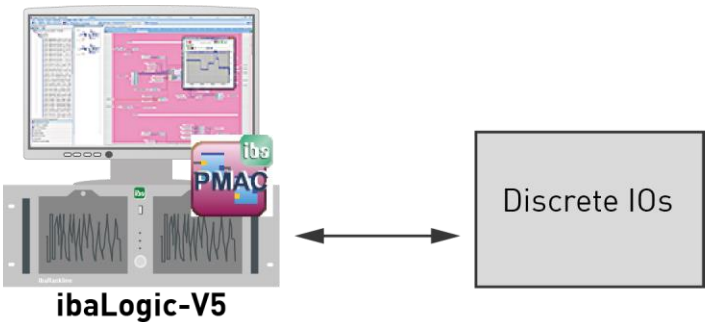


Figure 98: Peripheral interface Windows PC

ibaPADU-S-IT-2x16

The PMAC runs on an ibaPADU-S-IT-2x16 station. Only the newer ibaPADU-S-IT versions are supported from ibaPADU-S-IT-2x16 upwards.
All other ibaLogic components are always located on one or multiple Windows PCs.

Only the local I/O components (peripheral modules) are available to the PMAC on the ibaPADU-S-IT-2x16. There is a bi-directional FO connections and a network connection for TCP/IP coupling available for decentralized peripherals and external systems.

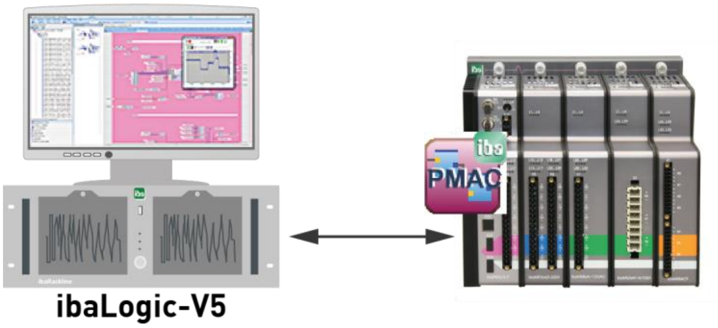


Figure 99: Peripheral interface of the PADU-S-IT-2x16

After calling the ibaLogic-V5 Client for the first time, the local Windows PC, that is, device class "Windows PC" is preset as the platform.

10.1 Configuring the Platform

The dialog box for configuring the platform is available under "Tools" in the menu bar of the ibaLogic Client.

Note

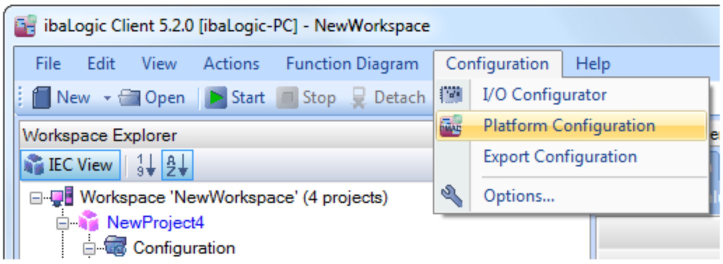
The platforms are created specifically for each project. All projects of the workspace and the platforms configured in it are available in the "Platform Configuration" dialog box.

Prerequisite

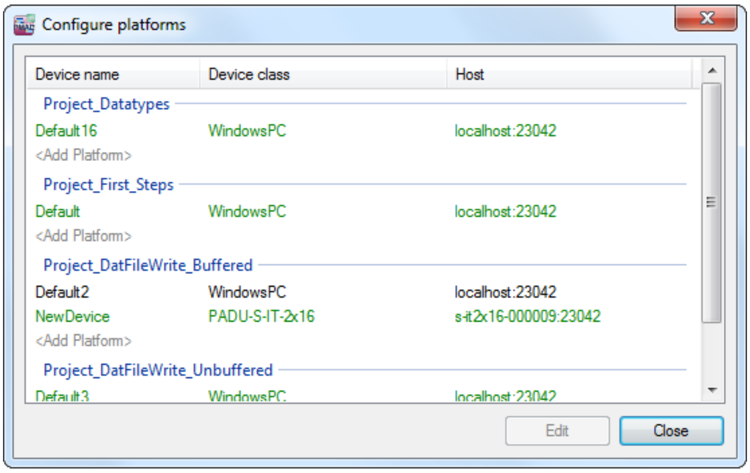
You have opened a project.

Procedure

➲ Select "Configuration - Platform Configuration" in the menu.



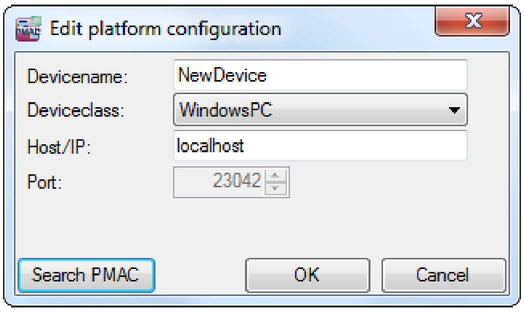
➲ Click in the dialog box under the project name on <Add Platform> or on the appropriate platform in order to create a platform or to configure it.



Device name	Device class	Host
Project_Datatypes		
Default16	WindowsPC	localhost:23042
<Add Platform>		
Project_First_Steps		
Default	WindowsPC	localhost:23042
<Add Platform>		
Project_DatFileWrite_Buffered		
Default2	WindowsPC	localhost:23042
NewDevice	PADU-S-IT-2x16	s-it2x16-000009:23042
<Add Platform>		
Project_DatFileWrite_Unbuffered		
Default3	WindowsPC	localhost:23042

Color	Explanation
Green	Is the active system.
Light gray	Creation of a new platform.
Black	Other platforms available.

➲ Click on the <Edit> button. The "Edit platform configuration" dialog box is displayed.


Edit platform configuration

Devicename: NewDevice
Deviceclass: WindowsPC
Host/IP: localhost
Port: 23042

➲ Enter a device name or accept the settings specified.
The name must be unique within the workspace.

➲ Select the device class WindowsPC or PADU-S-IT2x16.

➲ Enter the host IP.

- If you have selected the Windows PC device class, enter "localhost" or "127.0.0.1" when the PMAC runs on the computer or on the server.
- If the PMAC is located on another computer within the network, enter the host name or the IP address of this computer.
- If you have set PADU-S-IT2x16 as the device class, enter the host name or the IP address of the ibaPADU-S-IT-2x16 device on which the PMAC should run.

➲ Confirm your inputs by clicking on the <OK> button.

➲ Exit the dialog box with <Close>.
Alternatively:
open the selection box next to "Current Platform" in the toolbar and select <Add Platform> or <Edit Platform>.

Important note

When adding a new platform using the <Add platform> item in the selection box, a connection to the PMAC is immediately established on the newly configured platform by clicking the <OK> button.

10.2 Selecting the Platform

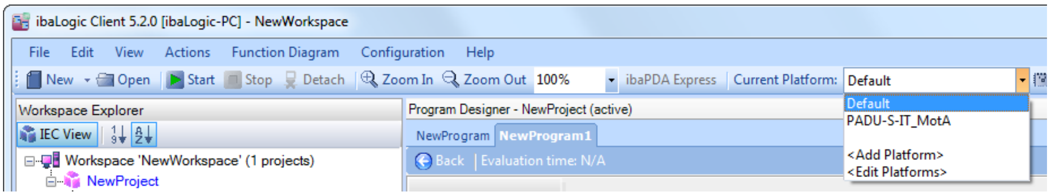
The currently set platform is displayed in the toolbar of the ibaLogic-V5 Client. You can use the selection box to switch to another platform.

Important Note

Please note that the platform is always set for the active project and not for the project being edited.

Procedure

1. Click on the selection box to switch the platform.
2. Select one of the already configured platforms.



Current Platform: Default

Default
PADU-S-IT_MotA
<Add Platform>
<Edit Platforms>

Important Note

Please note that the I/O configuration depends on the platform.

After setting up the platform, you have to update the I/O configuration.

Select the "Configuration – I/O Configurator" button <Update Hardware >.

10.3 Update the platform

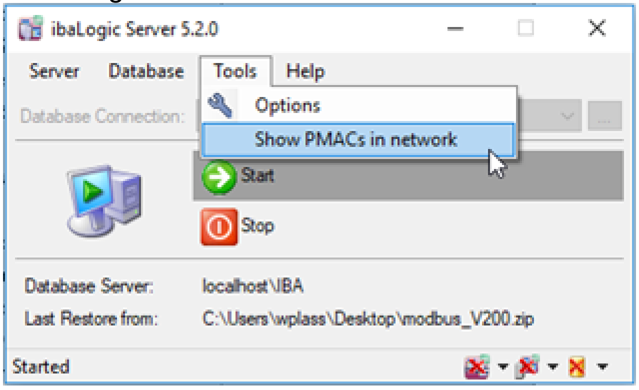
When connecting to an ibaPADU-S-IT2x16 platform, the ibaLogic Client automatically checks if the client version is compatible to the PMAC version. If the two versions are not identical, an update option is offered automatically.

With older versions of ibaLogic, it may happen that only one message is displayed that the two versions of ibaLogic do not match. In this case you should perform the update manually.

Proceed

➲ The manual update is started in the display of the available PMACs in the network. There are 2 options to display the available PMACs:

❑ In ibaLogic Server: select "Show PMACs in the network" the "Tools" menu

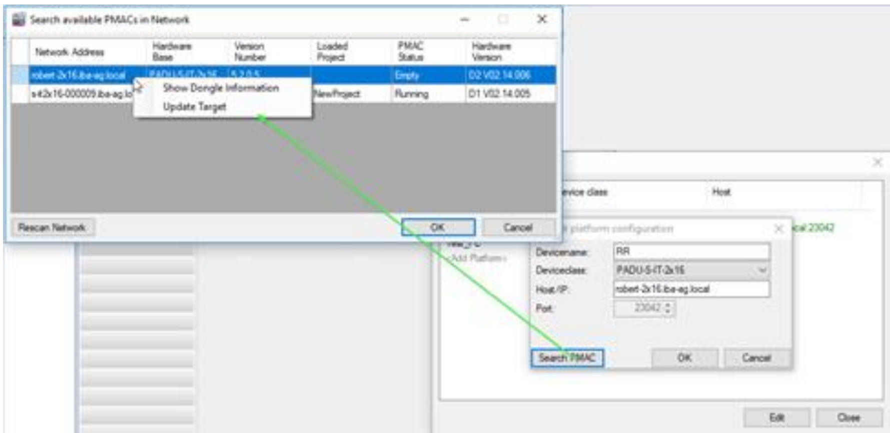


Database Server: localhost\IBA
Last Restore from: C:\Users\wplass\Desktop\modbus_V200.zip

Started

or

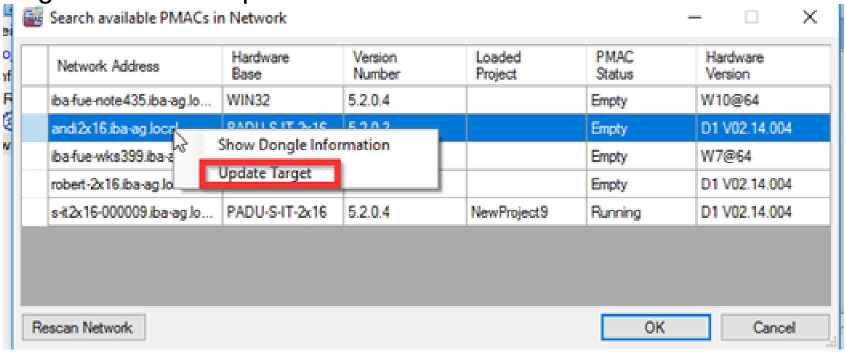
❑ In ibaLogic Client: select "Platform configuration" in the "Configuration" menu. A click on the button <Edit> opens the dialog "Edit platform configuration". A click on the button <Search PMAC> opens a list of all PMACs available in the network.



Network Address	Hardware Base	Version Number	Loaded Project	PMAC Status	Hardware Version
robert-2x16.iba-ag.local	PADU-S-IT-2x16	5.2.0.5		Empty	D2 V02.14.006
s-it2x16-000009.iba-ag.lo[illegible]			NewProject	Running	D1 V02.14.005

Devicename: RR
Deviceclass: PADU-S-IT-2x16
Host/IP: robert-2x16.iba-ag.local
Port: 23042

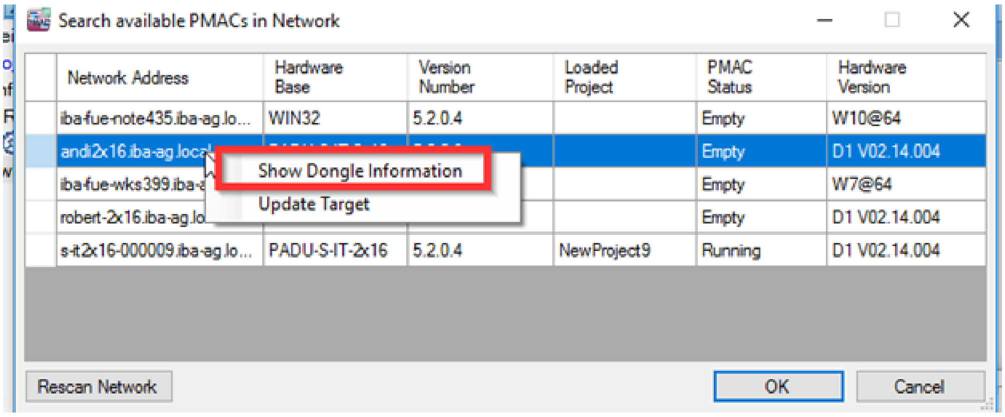
➲ Select the desired platform and open the context menu with a right-click. "Update target" starts the update to the new version.



Network Address	Hardware Base	Version Number	Loaded Project	PMAC Status	Hardware Version
iba-fue-note435.iba-ag.lo...	WIN32	5.2.0.4		Empty	W10@64
andi2x16.iba-ag.local	PADU-S-IT-2x16	5.2.0.3		Empty	D1 V02.14.004
iba-fue-wks399.iba-a				Empty	W7@64
robert-2x16.iba-ag.lo				Empty	D1 V02.14.004
s-it2x16-000009.iba-ag.lo...	PADU-S-IT-2x16	5.2.0.4	NewProject9	Running	D1 V02.14.004

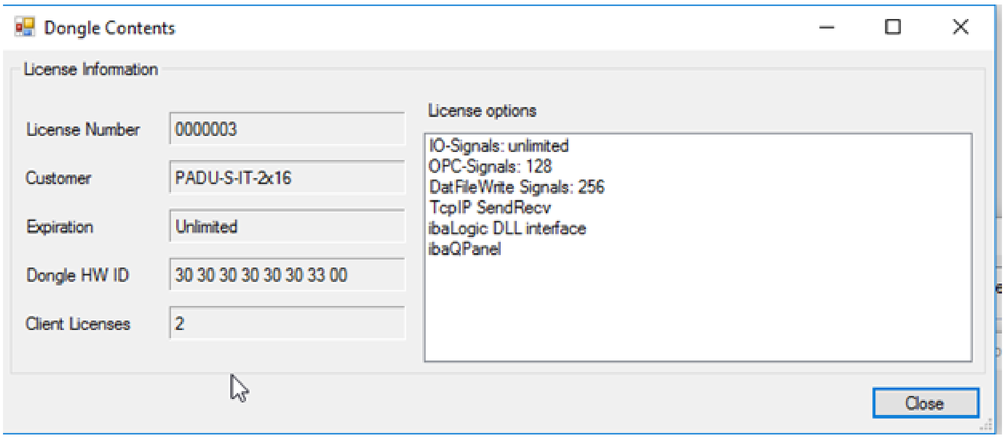
10.3.1 Show dongle information

The list of PMACs offers the possibility to display the dongle information. Open the list of PMACs available in the network as described above. Right-click the desired platform and select "Show dongle information" from the context menu.



Network Address	Hardware Base	Version Number	Loaded Project	PMAC Status	Hardware Version
iba-fue-note435.iba-ag.lo...	WIN32	5.2.0.4		Empty	W10@64
andi2x16.iba-ag.local				Empty	D1 V02.14.004
iba-fue-wks399.iba-a				Empty	W7@64
robert-2x16.iba-ag.lo				Empty	D1 V02.14.004
s-it2x16-000009.iba-ag.lo...	PADU-S-IT-2x16	5.2.0.4	NewProject9	Running	D1 V02.14.004

Figure 100: Context menu "Show dongle information"



License Number: 0000003

Customer: PADU-S-IT-2x16

Expiration: Unlimited

Dongle HW ID: 30 30 30 30 30 30 33 00

Client Licenses: 2

License options

IO-Signals: unlimited
OPC-Signals: 128
DatFileWrite Signals: 256
TcpIP SendRecv
ibaLogic DLL interface
ibaQPanel

Figure 101: Dongle contents

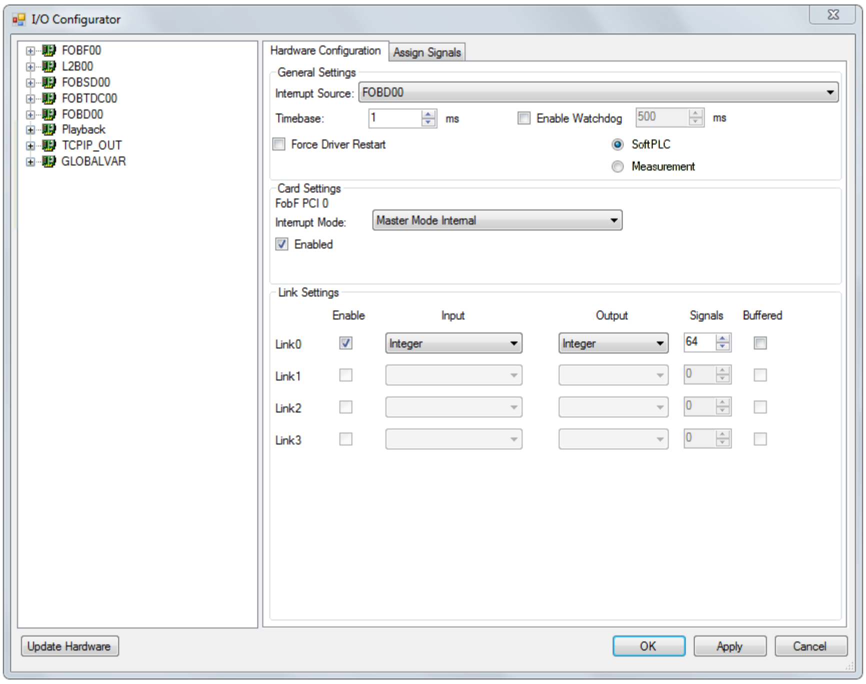
11 IO Configuration

The IO Configurator is the centralized dialog in which you can make all configuration settings relating to the input and output signals as well as certain interfaces.

Note

Exceptions are all those interfaces that are available as function blocks, e. g. TCPIP_SENDRECV function block or the in-built OPC interface.

➲ Open the IO Configurator with the menu "Tools - IO Configurator".



General Settings

Interrupt Source: FOBD00

Timebase: 1 ms ☐ Enable Watchdog 500 ms

☐ Force Driver Restart ● SoftPLC ○ Measurement

Card Settings

FobF PCI 0

Interrupt Mode: Master Mode Internal

☑ Enabled

Link Settings

	Enable	Input	Output	Signals	Buffered
Link0	☑	Integer	Integer	64	☐
Link1	☐			0	☐
Link2	☐			0	☐
Link3	☐			0	☐

The IO Configurator dialog screen is divided into 3 sections:

- ❑ Input / output resources available
- ❑ Hardware Configuration
- ❑ Assign Signals

Input / output resources available

All hardware and software interfaces detected and supported by the system are displayed in a tree structure on the left side of the dialog screen.

With playback, signals from an ibaPDA DAT file can be parameterized as input signals.

Hardware Configuration

This is where you can configure special settings for the entire system and for individual cards.

Assign Signals

You can work with self-defined and meaningful input and output names in ibaLogic (virtual inputs and outputs). These virtual signals are assigned with the help of signal assignment to the physical inputs and outputs. The virtual signals are divided in groups.

11.1 Resources

The I/O Configurator applies the I/O configuration associated with the active project when it is opened.

Note

The I/O configuration is not saved in the database, but in two XML files in the path “…\ibaLogic v5\Server\HwMappings". Hence, it is possible to edit the project regardless of the hardware available.

The I/O configuration files are saved during database backup in ZIP files and are reloaded when the database is restored so that after transporting the projects, even the original I/O configuration is available.

When saving the database backup in BAK files, the I/O configuration is not saved.

<Update Hardware> button:

By clicking on the <Update Hardware> button, the hardware that is available to the computer on which the PMAC is running is applied. This includes the PCI cards supported on the Windows PC platform (see "*Hardware resources*, page 173"). For the ibaPADU-S-IT-2x16 platform, these are the peripheral modules available there.

Important note

The platform must be configured prior to editing the I/O configuration, since the settings of the I/O Configurator are lost when you switch the platform!

Tree structure

Prerequisite

❑ You have activated the corresponding link in the hardware configuration and accepted the configuration using the <Apply> button at the lower border of the dialog box.

Procedure

➲ You can open the tree structure right up to each individual signal by clicking on the +/- symbol in front of the names.

The following hierarchy levels are displayed:

Interface → module → inputs/outputs → signals

Interface:	Name and index for the card types
Modules:	Group according to the physical division, depending on the type of card.
Inputs / Outputs:	Modules may have only inputs, only outputs or both
Signals:	The names of the hardware signals consist of the module name, direction, data type and serial number.

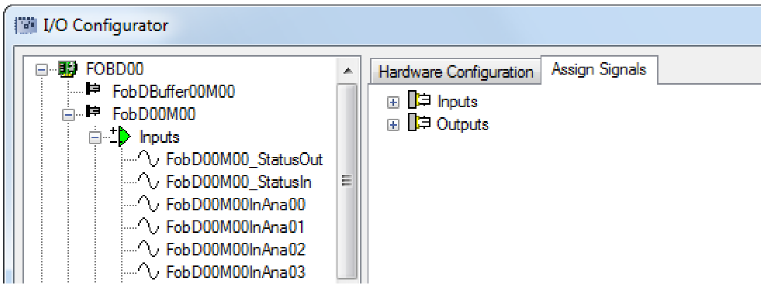


Figure 102: Assign Signals

11.1.1 Hardware resources

ibaLogic supports the following interfaces:

Windows PC platform and PCI cards

Interface	Cards	Connections	Protocol
FOBFnn[5]	ibaFOB-io-S ibaFOB-4i-S ibaFOB-4o-S	1 FO link 4 FO links (simplex/duplex)	ibaNet with 2 and 3.3 MBit
	ibaFOB-2io-X ibaFOB-4i-X ibaFOB-4o-X	2 FO links 4 FO links (simplex/duplex)	ibaNet with 3.3 and 32 MBit (32 MBit only for reception)
FOBD*ii*	ibaFOB-io-D ibaFOB-2io-D ibaFOB-4io-D ibaFOB-4o-D	2 FO links 1 FO link 4 FO links (simplex/duplex)	ibaNet with 2, 3.3 and 32 MBit and DMA
FOBSD*nn*	ibaFOB-SD	1 FO link duplex (ST)	SIMADYN D
FOBTDC*nn*	ibaFOB-TDC	1 FO link duplex (SC)	SIMATIC TDC
L2B*nn*	ibaCOM-L2B 4/8 ibaCOM-L2B 8/8	4 slaves 8 slaves	Profibus DP Slaves
SST_Master*nn*	SST-PCB3	max. 125 slaves	Profibus DP Master
RFM*nn*	VMIC-5565 VMIC-5576 VMIC-5579 VMIC-5587 VMIC-5588	1 FO duplex 1 coaxial 1 FO duplex 1 FO duplex 1 FO duplex	Reflective Memory

Table 1: Windows PC platform

nn = Card numbering 00 to 03 (max. 4 cards of one type are allowed)

ii = Card numbering 00 to 07 (max. 8 cards of type FOB-D are allowed).

[5] Cards for this interface are available only for old systems, however, they are not executable with Windows operating system 7/64 bit.

ibaPADU-S-IT-2x16 platform

Interface	Peripherals
ibaPADU-S-IT-2x16	Local peripherals, i. e. the interface modules on the ibaPADU-S module rack.

Other documentation - ibaPADU-S-IT-2x16 platform

For detailed information, please refer to the ibaPADU-S-IT-2x16 manual.

11.1.2 Software resources

ibaLogic supports the following protocols that are based on Ethernet:

Display	Protocol
TCPIP_OUT	TCP/IP protocol ibaLogic to ibaPDA: Windows PC class: max. 16 telegrams for every 32 real values and 32 binary values ibaPADU-S-IT-2x16 class: max. 16 telegrams for every 32 real values and 32 binary values

Note

Only outputs are available.

11.1.3 Global System Variables

The following global system variables are provided:

Interface	Variables
GLOBALVAR	Global input variables: LAST_DRIVER_ERROR Last error occurred on the driver displayed as hex code. WATCHDOG_BITE FALSE by default. If this value becomes TRUE, the configured watchdog of ibaLogic has responded and switched off the outputs of the cards. For more information on "Watchdog", please refer to "*General settings*, page 181". SYSTEM_UTC_TIME Current system time in UTC format (UniversalTimeCoordinated). DONGLE_NUMBER Dongle number of the iba dongle plugged in on the PC or the serial number of the ibaPADU-S-IT-2x16. ACQ_RESTART_COUNT Counter for internal driver restart. This is used to detect system overloads in the "Buffered Mode"

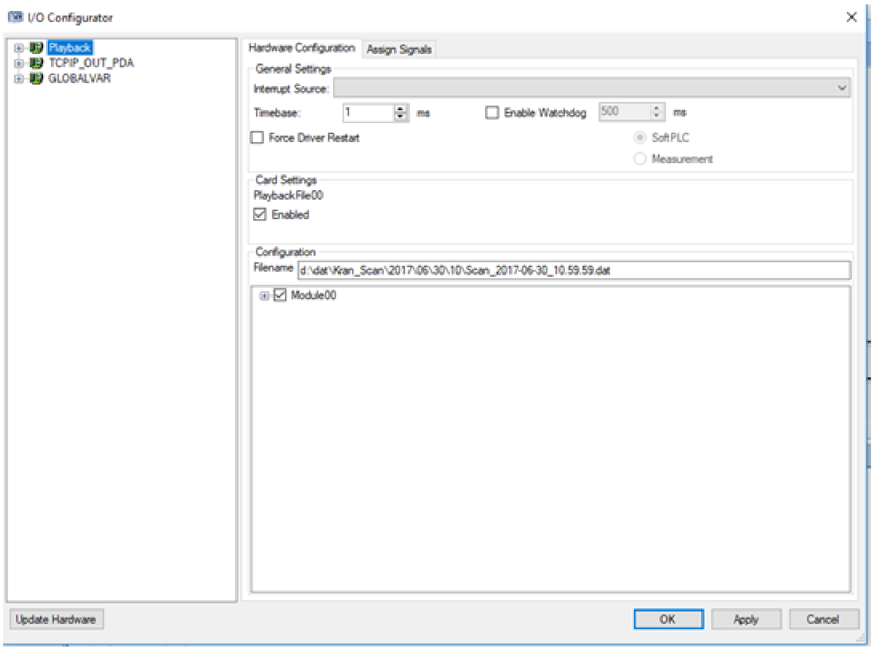
	DONGLE_INFO A string which indicates the serial number and the customer name of the dongle. As soon as a problem concerning the dongle is detected (e. g. dongle is removed during operation), then the string contains: "SerNo: No Dongle found". As a result a warning message can be sent before the program will be stopped. after approx. 5 minutes.

11.1.4 Playback

The playback mode is particularly helpful for simulating processes. In playback mode, a data file that was generated by an iba online measuring system, e.g. ibaPDA, ibaQDR, etc., can be played similar to an audio tape and used as input signal source.

What is special about this is that this is real data of a system or a process leading to much higher accuracy for simulation and test setups as compared with model calculations. This is particularly interesting for modernization projects. Moreover, in playback mode, the signals from one file can be mixed with signals of the hardware inputs.

The configuration is performed via the I/O Configurator.

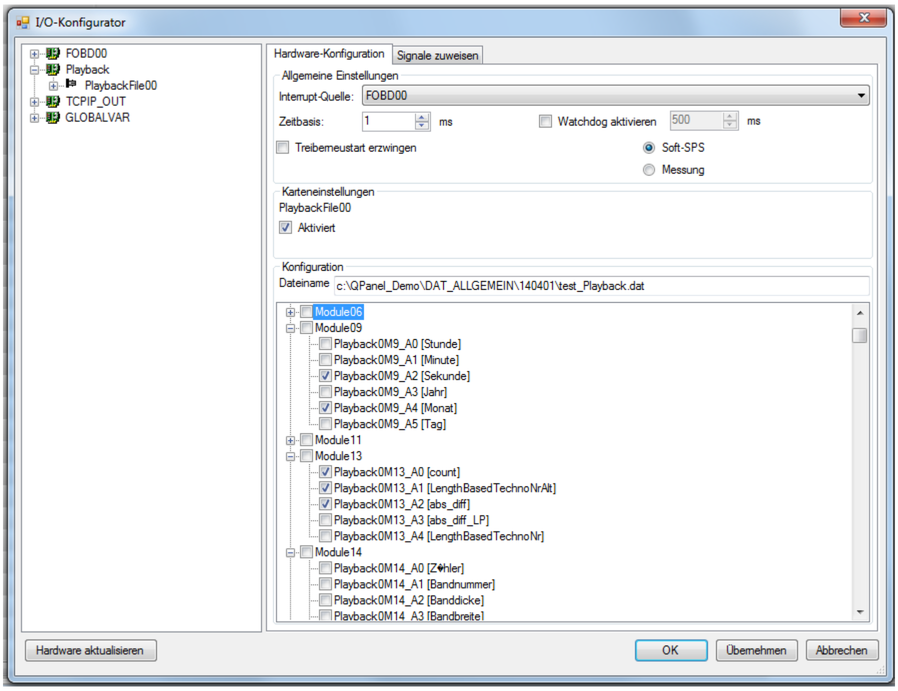


General Settings
Interrupt Source:
Timebase: 1 ms
□ Enable Watchdog 500 ms
□ Force Driver Restart
● Soft PLC
○ Measurement

Card Settings
PlaybackFile00
☑ Enabled

Configuration
Filename d:\dat\Kran_Scan\2017\06\30\10\Scan_2017-06-30_10.59.59.dat
☑ Module00

1. You activate the card and specify a corresponding iba data file (*.dat).
   Clicking “Apply”, you get all modules and associated signals existing in this data file.



I/O-Konfigurator

- FOBD00
- Playback
  - PlaybackFile00
- TCPIP_OUT
- GLOBALVAR

Hardware-Konfiguration | Signale zuweisen

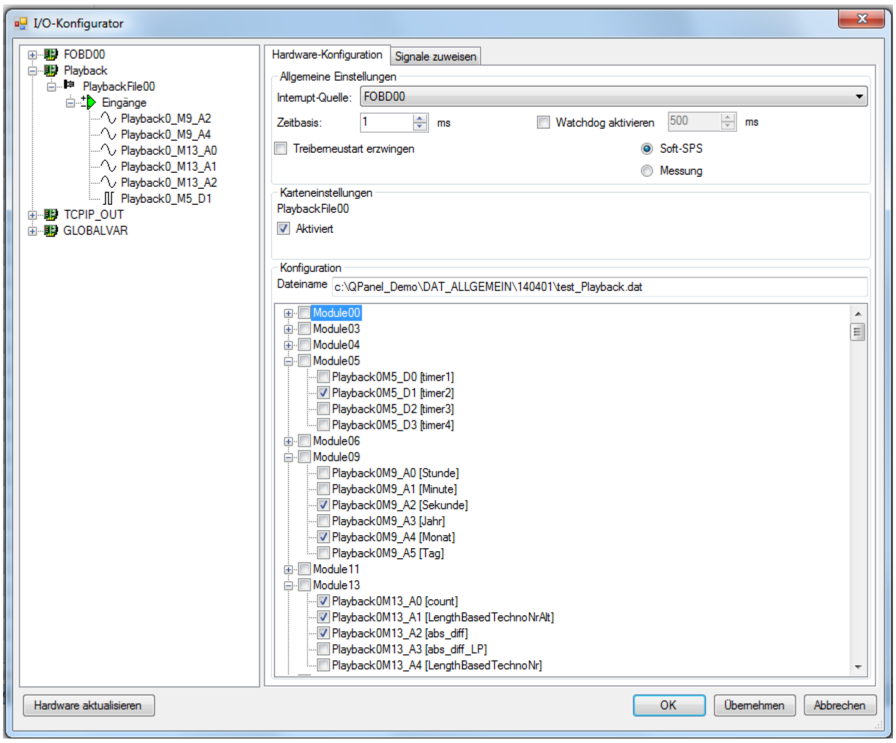
Allgemeine Einstellungen
Interrupt-Quelle: FOBD00
Zeitbasis: 1 ms ☐ Watchdog aktivieren 500 ms
☐ Treiberneustart erzwingen ● Soft-SPS ○ Messung

Karteneinstellungen
PlaybackFile00
☑ Aktiviert

Konfiguration
Dateiname c:\QPanel_Demo\DAT_ALLGEMEIN\140401\test_Playback.dat

- ☐ Module06
- ☐ Module09
  - ☐ Playback0M9_A0 [Stunde]
  - ☐ Playback0M9_A1 [Minute]
  - ☑ Playback0M9_A2 [Sekunde]
  - ☐ Playback0M9_A3 [Jahr]
  - ☑ Playback0M9_A4 [Monat]
  - ☐ Playback0M9_A5 [Tag]
- ☐ Module11
- ☐ Module13
  - ☑ Playback0M13_A0 [count]
  - ☑ Playback0M13_A1 [LengthBasedTechnoNrAlt]
  - ☑ Playback0M13_A2 [abs_diff]
  - ☐ Playback0M13_A3 [abs_diff_LP]
  - ☐ Playback0M13_A4 [LengthBasedTechnoNr]
- ☐ Module14
  - ☐ Playback0M14_A0 [Z♦hler]
  - ☐ Playback0M14_A1 [Bandnummer]
  - ☐ Playback0M14_A2 [Banddicke]
  - ☐ Playback0M14_A3 [Bandbreite]

2. Select the desired signals and click “Apply” again.
   Now, all selected signals are available as input.



I/O-Konfigurator

- FOBD00
- Playback
  - PlaybackFile00
    - Eingänge
      - Playback0_M9_A2
      - Playback0_M9_A4
      - Playback0_M13_A0
      - Playback0_M13_A1
      - Playback0_M13_A2
      - Playback0_M5_D1
- TCPIP_OUT
- GLOBALVAR

Hardware-Konfiguration | Signale zuweisen

Allgemeine Einstellungen
Interrupt-Quelle: FOBD00
Zeitbasis: 1 ms ☐ Watchdog aktivieren 500 ms
☐ Treiberneustart erzwingen ● Soft-SPS ○ Messung

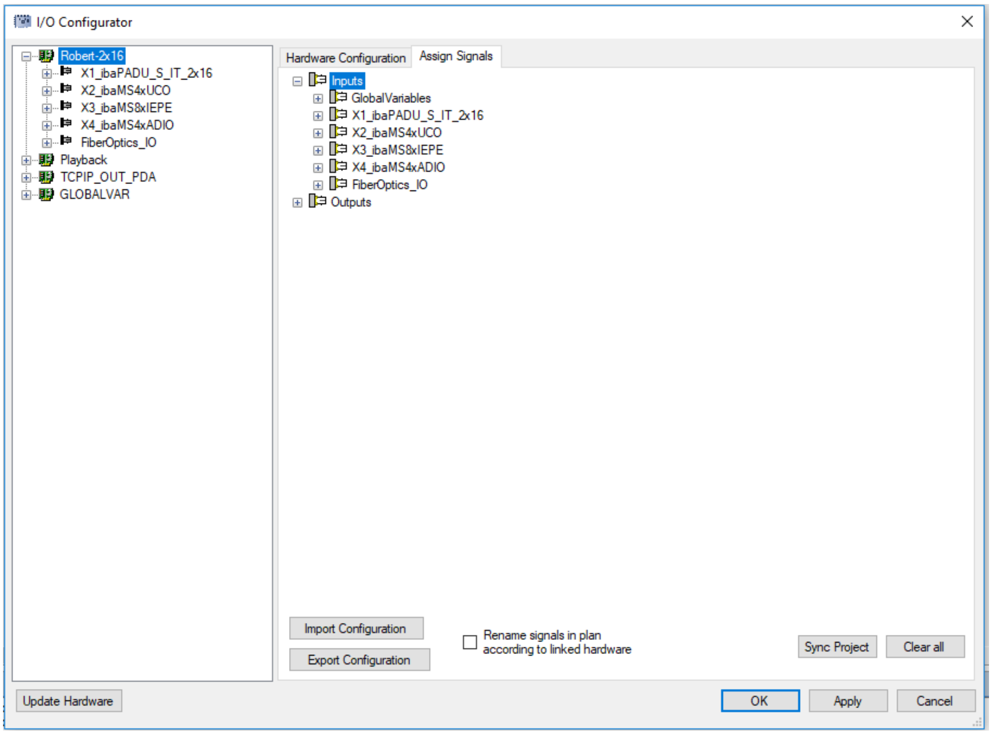
Karteneinstellungen
PlaybackFile00
☑ Aktiviert

Konfiguration
Dateiname c:\QPanel_Demo\DAT_ALLGEMEIN\140401\test_Playback.dat

- ☐ Module00
- ☐ Module03
- ☐ Module04
- ☐ Module05
  - ☐ Playback0M5_D0 [timer1]
  - ☑ Playback0M5_D1 [timer2]
  - ☐ Playback0M5_D2 [timer3]
  - ☐ Playback0M5_D3 [timer4]
- ☐ Module06
- ☐ Module09
  - ☐ Playback0M9_A0 [Stunde]
  - ☐ Playback0M9_A1 [Minute]
  - ☑ Playback0M9_A2 [Sekunde]
  - ☐ Playback0M9_A3 [Jahr]
  - ☑ Playback0M9_A4 [Monat]
  - ☐ Playback0M9_A5 [Tag]
- ☐ Module11
- ☐ Module13
  - ☑ Playback0M13_A0 [count]
  - ☑ Playback0M13_A1 [LengthBasedTechnoNrAlt]
  - ☑ Playback0M13_A2 [abs_diff]
  - ☐ Playback0M13_A3 [abs_diff_LP]
  - ☐ Playback0M13_A4 [LengthBasedTechnoNr]

3. In the signal assignment, these signals can be assigned to the virtual signals now. (See general *Signal assignment*, page 183).

   Quick assignment is achieved by opening the ”Assign Signals” tab and dragging all signals (PlaybackFile00) via drag & drop to this view.



4. The signals are now created and can be used in the project.

11.1.4.1 Playback control signals

Control signals are available for the playback module.

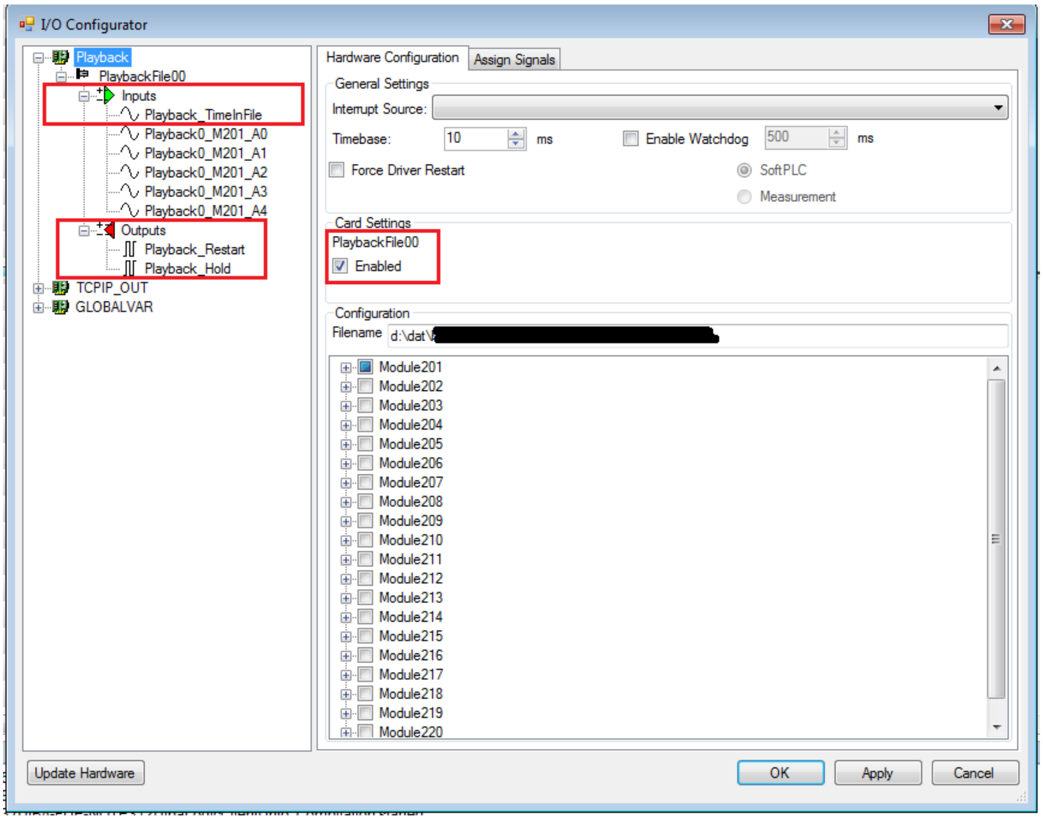
When playback is enabled and a playback file (=ibaPDA-Dat-File) is configured, the following new signals are available after applying and signal assignment:

❑ Control signals (=output):

Playback_Restart (boolean)	Rising edge = Restart the playback from the beginning of the file
Playback_Hold (boolean)	1 = pause playback, all inputs stay at the last value 0 =continue playback
Playback_FileName_Takeover (bool)	1 = A rising edge takes over the file specified in FileName for playback and starts at the beginning of the file
Playback_FileName (String)	Selects the playback file with path and name

❑ Control signal (=input):

Playback_TimeInFile (analog, Real)	Time of the current playback sample (in seconds from the start of the file)
Playback_ActiveFile (String)	Displays the path and name of the currently playing playback file



Interrupt Source:

Timebase: 10 ms

☐ Enable Watchdog 500 ms

☐ Force Driver Restart

● SoftPLC

○ Measurement

PlaybackFile00

☑ Enabled

Filename d:\dat\[redacted]

Figure 103: Control and status signals

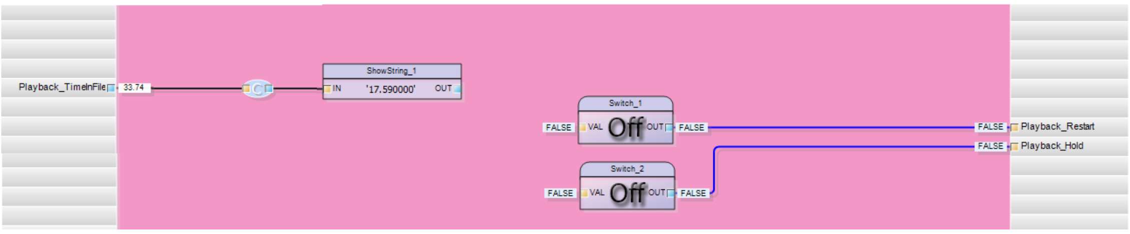
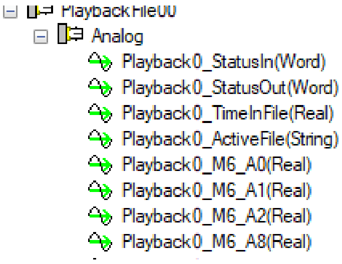
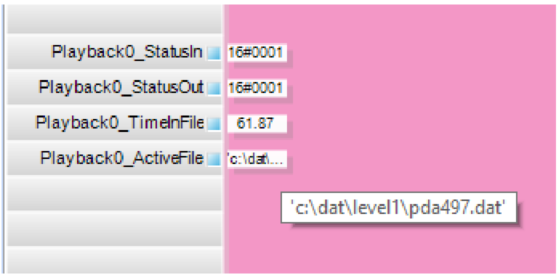


Figure 104: Online display



- PlaybackFile00
  - Analog
    - Playback0_StatusIn(Word)
    - Playback0_StatusOut(Word)
    - Playback0_TimeInFile(Real)
    - Playback0_ActiveFile(String)
    - Playback0_M6_A0(Real)
    - Playback0_M6_A1(Real)
    - Playback0_M6_A2(Real)
    - Playback0_M6_A8(Real)

Figure 105: Control signals for replaying different files



Playback0_StatusIn	16#0001
Playback0_StatusOut	16#0001
Playback0_TimeInFile	61.87
Playback0_ActiveFile	'c:\dat\...

'c:\dat\level1\pda497.dat'

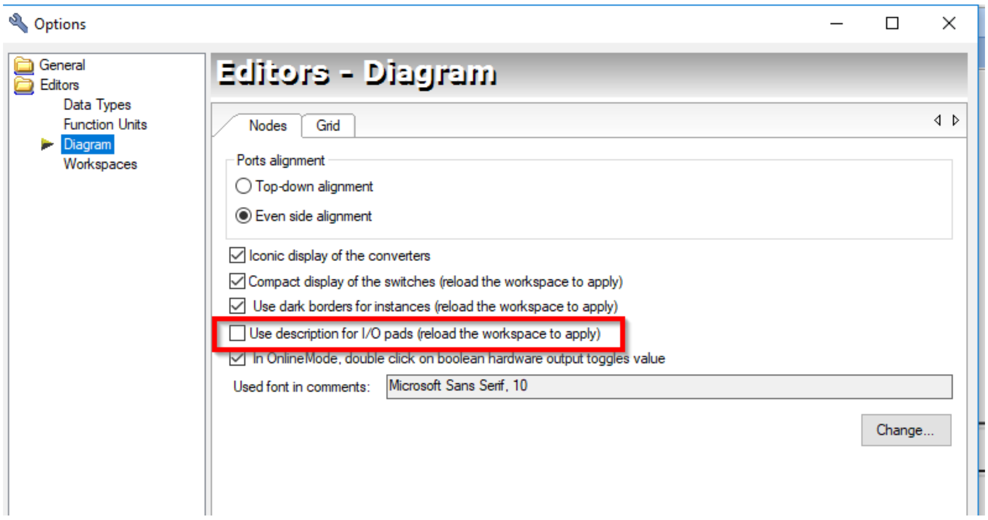
Figure 106: Online display

11.1.4.2 Optional display of signal name or signal description

For playback files or other configuration, the actual signal name is hidden in the signal description and is only visible in the tooltip.

The reason was that signal names must be IEC compliant and certain signal import or playback signal names do not meet this standard. Therefore, the signal name is generated generically and the actual signal name is moved into the description.

However, it is possible in the options to display the original name instead of the signal name. This only affects the display in the plan, for evaluation or display in ibaPDA Express still the IEC compliant signal name is used.



Ports alignment
○ Top-down alignment
● Even side alignment

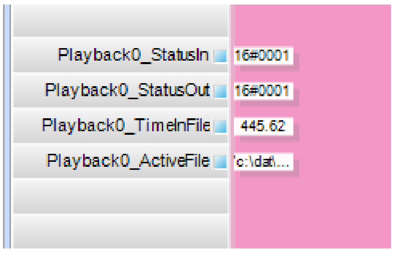
☑ Iconic display of the converters
☑ Compact display of the switches (reload the workspace to apply)
☑ Use dark borders for instances (reload the workspace to apply)
☐ Use description for I/O pads (reload the workspace to apply)
☑ In OnlineMode, double click on boolean hardware output toggles value

Used font in comments: Microsoft Sans Serif, 10

Figure 107: Setting to display the original name

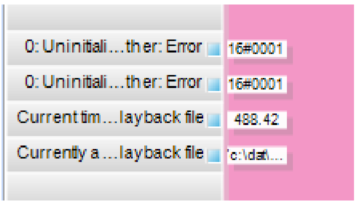
For the option to take effect, you may need to reload the work area.

Example:



Playback0_StatusIn 16#0001
Playback0_StatusOut 16#0001
Playback0_TimeInFile 445.62
Playback0_ActiveFile 'c:\dat\...

Figure 108: Option not selected



0: Uninitiali…ther: Error 16#0001
0: Uninitiali…ther: Error 16#0001
Current tim…layback file 488.42
Currently a…layback file 'c:\dat\...

Figure 109: Option selected

In order to represent longer names, a representation typical for Microsoft was chosen. The first and last characters are displayed, the rest is suppressed by .... Again, the tooltip shows the full name.

11.2 Hardware Configuration

The hardware configuration dialog screen is called via the "Hardware Configuration" tab.

There are 3 sections available in the hardware configuration in which you can configure the following settings.

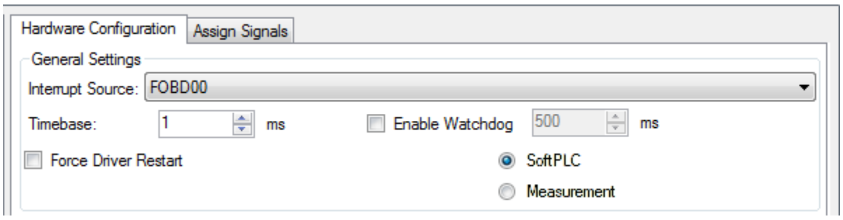
- General Settings for ibaLogic
- Card Settings
- Link (Connection) Settings

Note

You can make modifications only if the evaluation has not yet been started (Gray background in the design area).

11.2.1 General settings

The general settings are valid for the ibaLogic runtime system and to all interface cards.



Interrupt Source: FOBD00

Timebase: 1 ms

□ Enable Watchdog 500 ms

□ Force Driver Restart

● SoftPLC

○ Measurement

Figure 110: General settings

Interrupt source

The available iba modules are displayed in this list field or selection list. Select any module from this list that should work as the interrupt source with respect to the PCI bus.

If there is no I/O card in the system, ibaLogic uses an internal clock and the field is empty.

Under ibaPADU-S-IT-2x16 platform, only this is available.

Time base

The time base is the smallest cycle time that can be used. Please note that the configurable task intervals cannot be less than this time base.

Soft PLC (unbuffered mode)

This mode is suitable for open-loop and closed-loop control. This ensures in ibaLogic that only the latest signal states are processed. In contrast to the measurement mode, it is not decisive if samples are lost. Current data from the latest I/O transfer cycle are used for the evaluations.

For a detailed description, see "*Time response*, page 248".

Measurement (buffered mode)

This mode ensures that ibaLogic does not lose any input samples. This is also true when individual tasks within ibaLogic should be superseded. The runtime system of ibaLogic ensures that the data are made available equidistantly in the configured task interval. If tasks get superseded, the system makes up for the cycles. In buffered mode, also the unbuffered values are available as additional channel.

For a detailed description, see "*Time response*, page 248".

Enable watchdog

When this function is activated, a timer is started up in the cards available, and this timer is triggered by ibaLogic when there is a write operation at the outputs. If there is no write command (= trigger) from ibaLogic within the time configured, the card automatically sets all outputs to 0.

Since ibaLogic writes to the outputs cyclically, any trigger of the watchdog points to an application that has paused or is overloaded (e. g. by programmed endless loop).

Force Driver Restart

This function is particularly important for SST cards and reflective memory cards. These cards are also reset with the driver restart and, with it, the modified external configuration is applied, which eventually has been adapted with the configuration tool of the cards outside of ibaLogic.
Setting this option leads to restart with <OK>. This option is again reset automatically.

11.2.2 Card settings

Please mark the appropriate interface in the tree structure on the left for the settings. Only those settings are described here that are applicable to (almost) all cards.

❑ Interrupt mode

This field is provided only for iba cards.

The following modes are available for selection:

❑ Master Mode Internal:
The Master Mode Internal is to be set for one card in the PC only. This uses the internal timer to generate a synchronization signal that is distributed via a flat ribbon cable to the other cards.

❑ Slave Mode:
This mode must be set for all other cards.

Note

If there is a card of type FOB-SD or FOB-TDC in your configuration, select it as interrupt master. Otherwise, select one from the FOB-X or FOB-D type of cards. If neither of these cards is present, you can use the FOB-S or L2B card.

❑ Enabled

You can only use cards that have been enabled.

A card must be deactivated if an ibaPDA server is running on the same computer that uses this card. A card cannot be used by ibaLogic and ibaPDA simultaneously.

Other settings are specific to each card and described in "*PCI Interfaces (Windows PC)*, page 191".

11.3 Signal assignment

In order to use the physical inputs and outputs, you must assign them to the virtual signals in the program.

There are two methods for assigning signals:

- ❑ from the hardware signal to the program signal (seen from the hardware).
- ❑ from the program signal to the hardware signal (seen from the program).

It is also possible to use a mixed method between these two basic methods.

11.3.1 Method as seen from the hardware

At the start of programming, you already know the external interfaces, e.g. the assignment of the FOC telegrams or that of the Profibus slaves. From these physical signals you generate virtual signals that you can use subsequently in the program.

You can accept the names generated automatically or assign your own base names. The direction tag, type and index are added automatically. Finally, each signal name can be changed separately.

After acceptance, the virtual signals are visible in the navigation area under "Inputs - Outputs". Here, too, you can modify the signal names provided that they are not yet used in the program, i.e. dragged from the navigation area to the borders of the design area.

11.3.1.1 Example: Assigning all signals of a module of an ibaFOB-io-D card

The FOB card is listed, among others, on the left side.

The inputs and outputs are not yet assigned in the "Assign signals" tab.

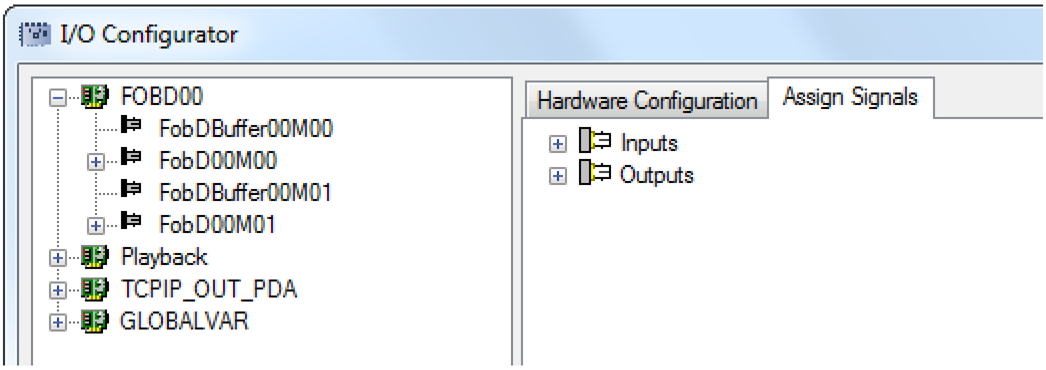


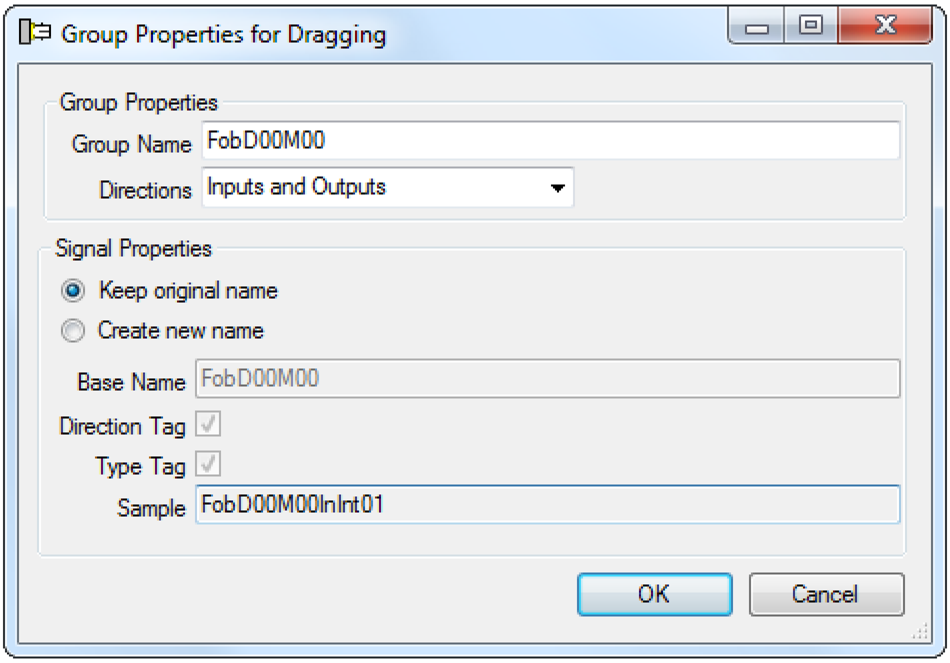
Figure 111: Assigning signals of the ibaFOB-io-D card

Procedure

1. Open the tree of the FOB card on the left side. All activated links of the card and their designations are displayed.

   The entire FobD00M00 module (matches the link 0 of the FOB card) can be assigned in one step.

2. Drag the FOBF00M00 module and drop it on the right side on the inputs or outputs. (This accordingly applies in case of buffered values for the buffered values, e.g. FobDBuffer00M00)
   The dialog “Group Properties" is displayed.



Group Properties for Dragging

Group Properties

Group Name: FobD00M00

Directions: Inputs and Outputs

Signal Properties

● Keep original name

○ Create new name

Base Name: FobD00M00

Direction Tag ☑

Type Tag ☑

Sample: FobD00M00InInt01

Note

ibaLogic creates a group with the group name under the inputs and outputs and forms a virtual name for each hardware signal. You can either leave the generation of the names entirely to ibaLogic or specify your own names.
During the generation, the composition of the current signal name is displayed. In the example given above, the name “FobD00M00InAna01” is composed of the following:

FobD00M00	Base name (same as the group name)
In	Direction Tag
Ana	Type Tag
01	Serial number

Depending on the direction that you have chosen in the selection box, all signals of this module are created in the group displayed under inputs and/or outputs.

Result

The assignment leads to the following result:

The virtual signal names are displayed on the left of the arrow and the hardware signal names on the right. The group name is identical to the physical module name.

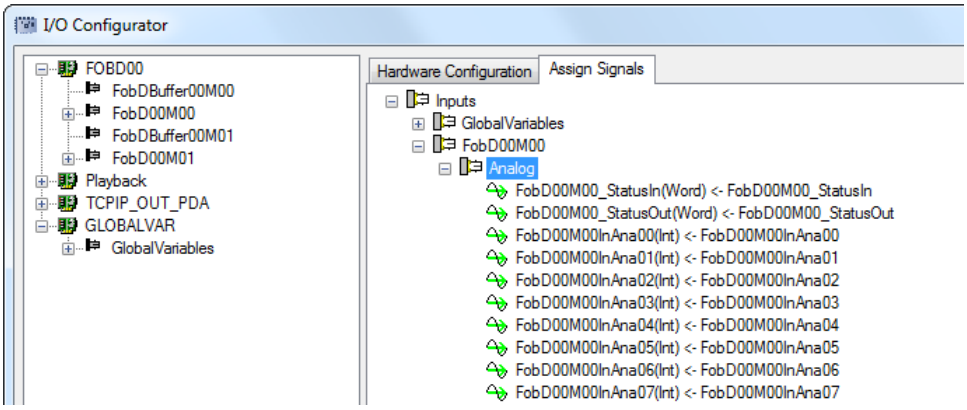

I/O Configurator
FOBD00
FobDBuffer00M00
FobD00M00
FobDBuffer00M01
FobD00M01
Playback
TCPIP_OUT_PDA
GLOBALVAR
GlobalVariables
Hardware Configuration
Assign Signals
Inputs
GlobalVariables
FobD00M00
Analog
FobD00M00_StatusIn(Word) <- FobD00M00_StatusIn
FobD00M00_StatusOut(Word) <- FobD00M00_StatusOut
FobD00M00InAna00(Int) <- FobD00M00InAna00
FobD00M00InAna01(Int) <- FobD00M00InAna01
FobD00M00InAna02(Int) <- FobD00M00InAna02
FobD00M00InAna03(Int) <- FobD00M00InAna03
FobD00M00InAna04(Int) <- FobD00M00InAna04
FobD00M00InAna05(Int) <- FobD00M00InAna05
FobD00M00InAna06(Int) <- FobD00M00InAna06
FobD00M00InAna07(Int) <- FobD00M00InAna07

Figure 112: Assign Signals

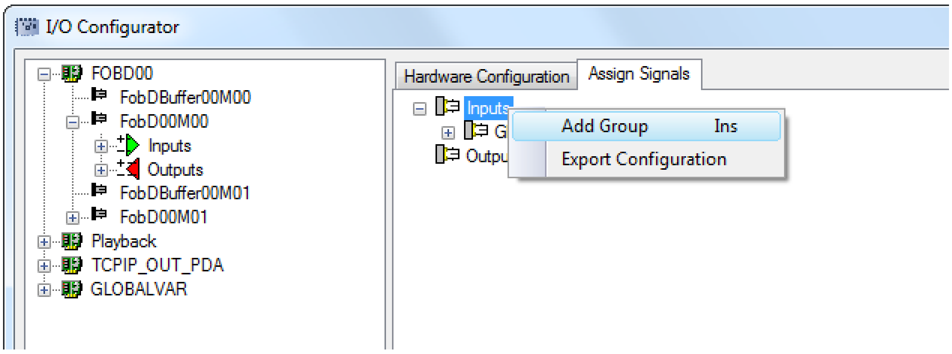
11.3.1.2 Example: Assignment of individual signals of an ibaFOB card

If you need only a few signals of a module in the program, you can assign only these signals as required.

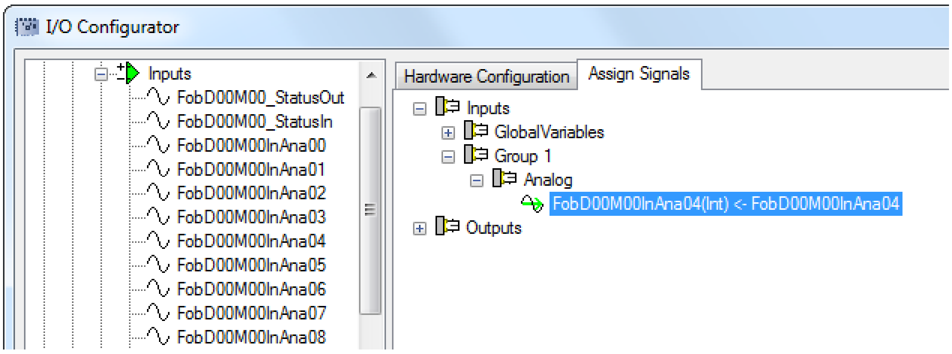
Procedure

Add group

1. Open the context menu by clicking the right mouse button.



2. Select the "Add Group" menu item, assign a group name and leave the "Setting the Group Name" dialog box by clicking OK.
3. Open the signal tree until you can see the individual signals of the module you want to use.
4. Add individual signals to the group defined earlier by drag and drop.



Result

Individual signals are assigned to the group.

11.3.1.3 Change Signal and Group Names

After the assignment, you can change the names of the virtual signals individually.

Procedure

1. You can change the group name by clicking on the right mouse button. Select the "Properties" menu item.
2. You can modify the signal name by double clicking on it. You can also add a description for the signal. The description is displayed in the program as a tooltip.
3. Confirm the settings with <Accept> or with <OK>.

11.3.2 Procedure as seen from the program

At the start of programming the external interface are not yet known, but you would like to commence with programming and find out the inputs and outputs that you need.

1. Define the inputs and outputs, groups and signals in the navigation area. Please see the description in "*Configure Inputs and Outputs*, page 79" for this purpose.
2. As soon as you know the physical interfaces at which the IO Signals are available, you can switch to the IO Configurator and assign these signals to the physical interfaces.

11.3.2.1 Example: Signals of an ibaFOB-2io-D card (complete module)

You assign the physical signals of link0 of the FOB card.

Prerequisite

You have defined a group "MotorA" and under this, the input signals "MotorA_N_Ist(Int)", "MotorA_Ta(Int)" and "MotorA_Status(Int)" in the program (in the navigation area of the inputs and outputs).

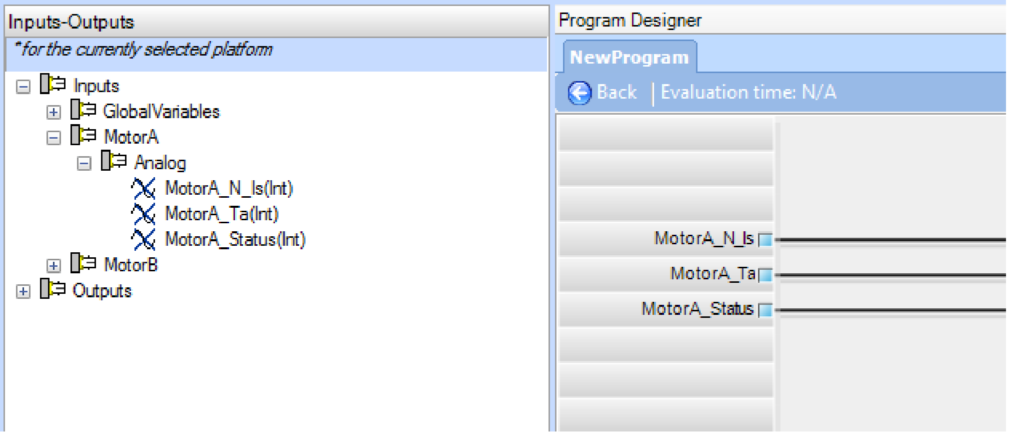


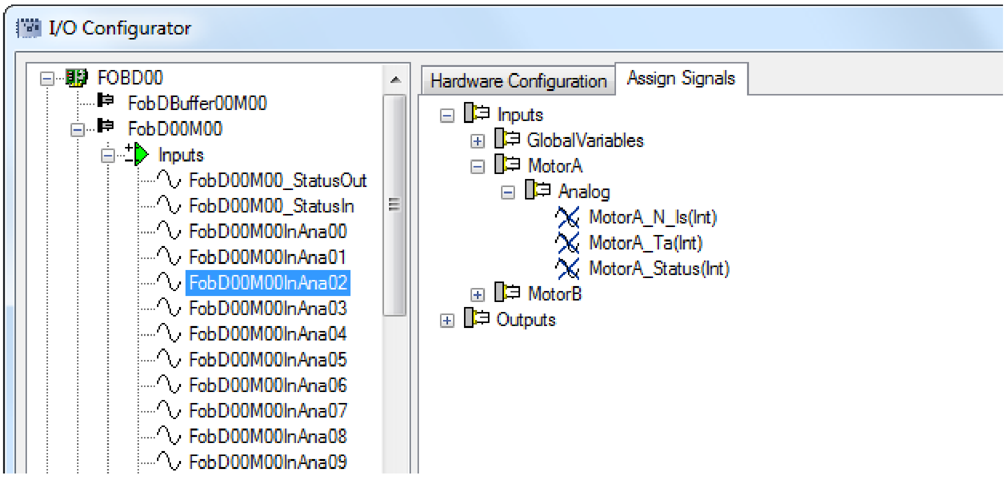
Figure 113: "Inputs – Outputs"

Procedure

1. Open the I/O configurator.
2. Update the hardware using the <Update Hardware> button.

3. Activate link0 on the FOB card.
   Please make sure that the data type of the link matches that of the signals already defined.
   You can see the signals defined in the project in the "Assign Signals" tab.

4. Select a hardware signal from the tree structure on the left.
   Drag & drop it on the signal.
   You have, thus, assigned the virtual signal to a physical signal.



Result

You can also see the assignment as tooltip in the design area.

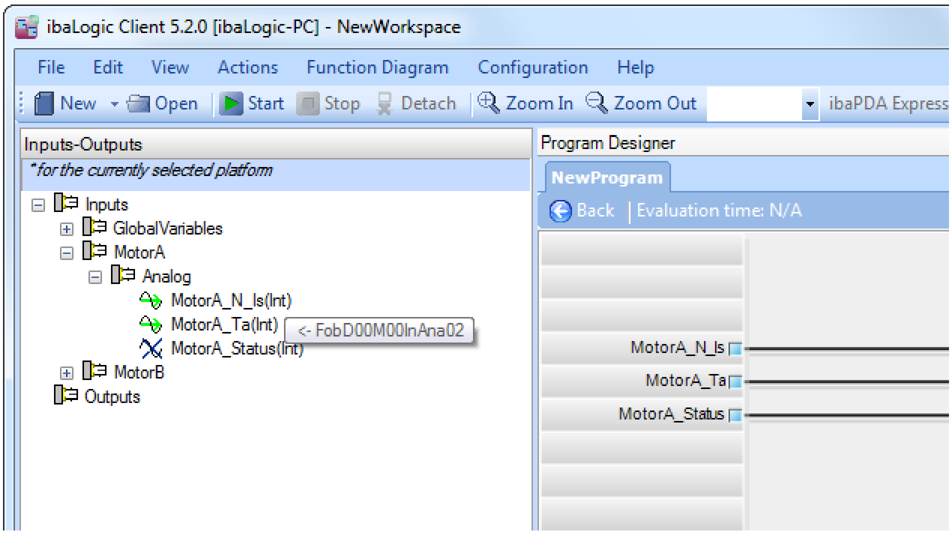


Figure 114: Assign signals

11.3.3 Change signal assignment

You can modify existing assignments in the I/O configurator. Multiple methods are possible.

Method 1

➲ Simply drag another hardware signal and drop it on a virtual signal that has already been assigned.
The assignment is modified by doing this.

Method 2

➲ Drag a hardware signal that is already in use and drop it on a virtual signal.
A message pops up.

➲ Acknowledge the information message, the assignment is modified and the link to the old signal is deleted.

Note

You can assign only one virtual signal to a hardware signal.

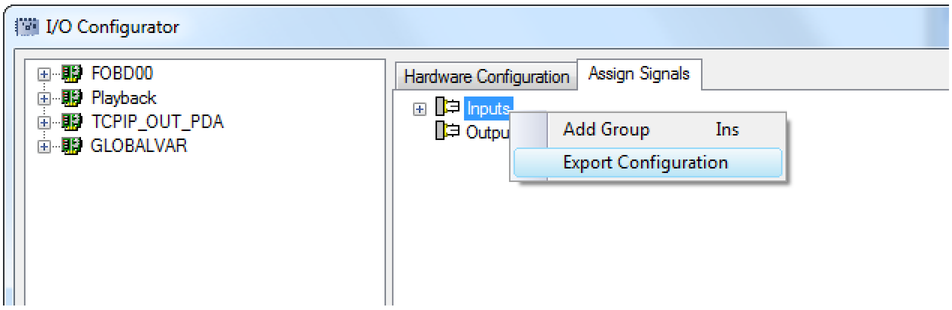
11.3.4 Using externally defined signal names

Using the export / import function, you can also use signal names that are available in external documents, e. g. in a spreadsheet program.

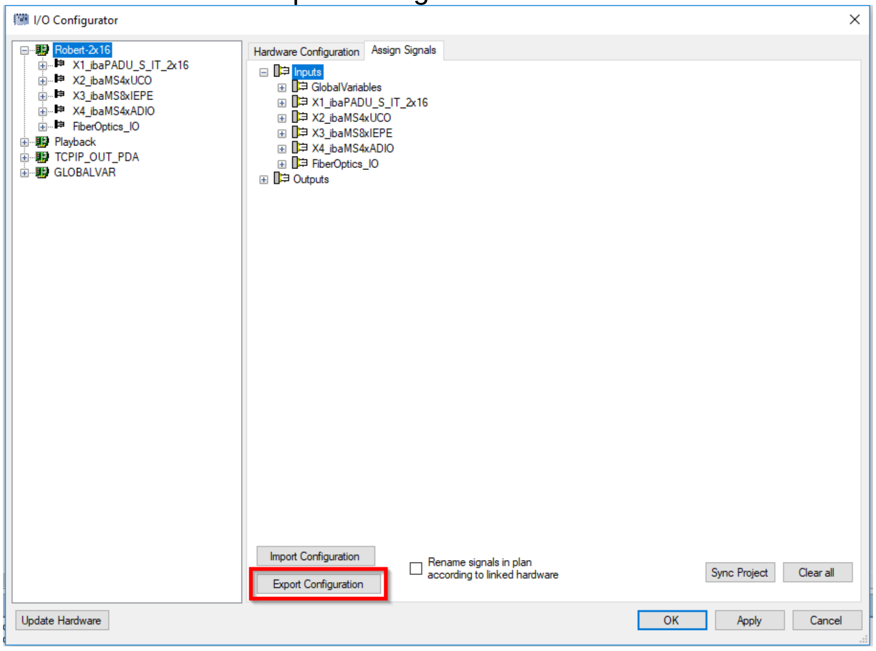
Procedure

➲ Specify the hardware signals.

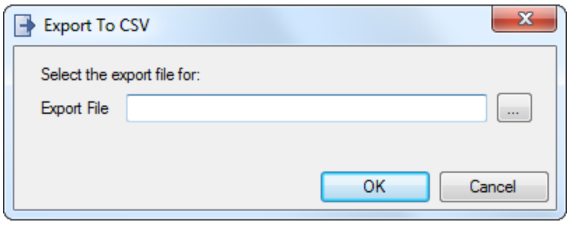
➲ Export the I/O configuration using the context menu of the inputs and outputs.



or click the button <Export configuration>



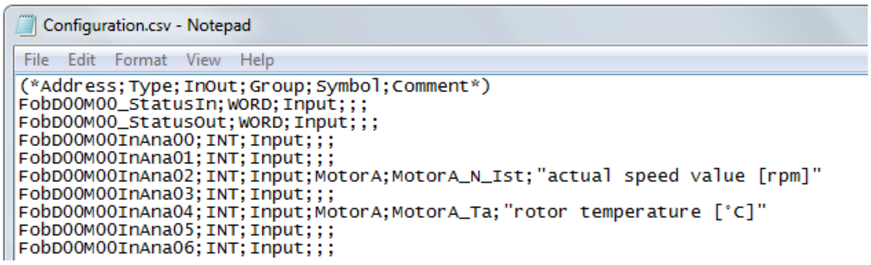
➲ A dialog box is displayed.



➲ Specify the target path and file name.

Result

You get a CSV file that you can open using an ASCII editor or a spreadsheet program.



```
(*Address;Type;InOut;Group;Symbol;Comment*)
FobD00M00_StatusIn;WORD;Input;;;
FobD00M00_StatusOut;WORD;Input;;;
FobD00M00InAna00;INT;Input;;;
FobD00M00InAna01;INT;Input;;;
FobD00M00InAna02;INT;Input;MotorA;MotorA_N_Ist;"actual speed value [rpm]"
FobD00M00InAna03;INT;Input;;;
FobD00M00InAna04;INT;Input;MotorA;MotorA_Ta;"rotor temperature [°C]"
FobD00M00InAna05;INT;Input;;;
FobD00M00InAna06;INT;Input;;;
```

Figure 115: CSV file in ASCII editor (Notepad)

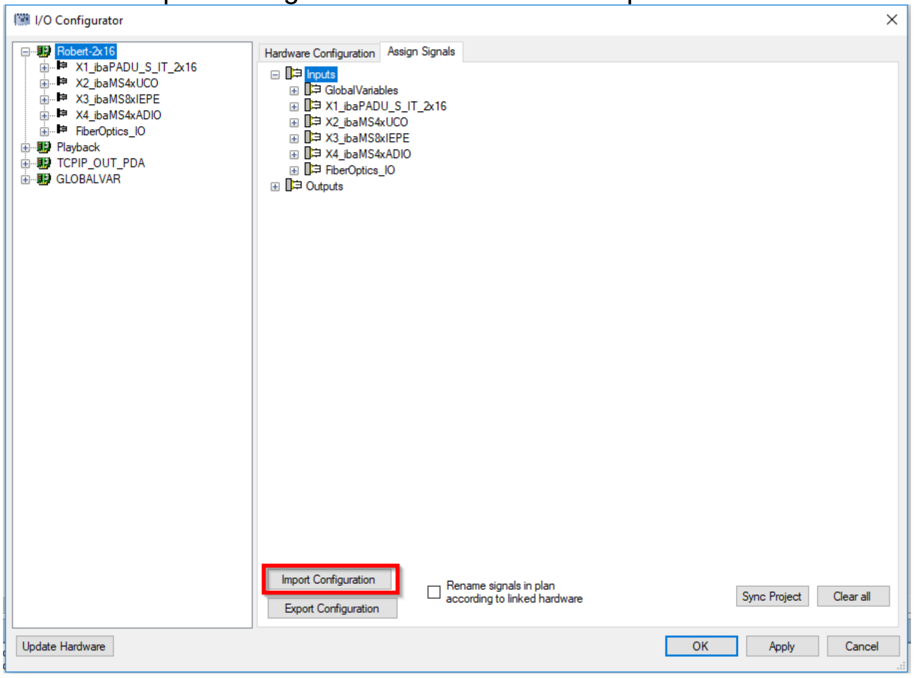
Important Note

You can see the hardware signals defined in the columns, Address, Type and InOut.

Do not modify these.

You can edit the virtual signals under the columns, Group, Symbol and Comments, or copy & paste them from the external document.
Please note that the names in the "Symbol" column conform to the IEC standard.

➲ Use the <Import configuration> button for the import.



➲ or exit the IO Configurator and select "File - Import - Signal mapping" from the main menu.

Rename signals in plan according to...: When a configuration is imported, the configured hardware is assigned to each virtual signal. Virtual signals not present in the plan are then available in the signal tree. If you want to rename virtual signals with the import, this option must be selected. The system then checks which virtual signals are currently attached to which hardware. These are overwritten according to the new configuration in the assignment and in the plan.

11.3.5 Synchronize a project

All virtual signals created in the ibaLogic layout without hardware connection are taken into a "synchronized" group so that they can be connected to the hardware signals.

11.4 PCI Interfaces (Windows PC)

This section describes the special settings of the interface cards.

For more information please refer to "*Hardware resources*, page 173".

11.4.1 Connection to the "iba World"

The following PCI cards are available for FOC connection to decentralized iba peripherals (PADUs) and to the iba system interface cards. You will find them in the resource tree under the **FOBFnn** or **FOBDnn** interface type.

❑ *ibaFOB-S*[6] (FOC link, 3 Mbit Protocol)

❑ *ibaFOB-X*[7] (FOC link, 3 Mbit / 32 Mbit Protocol)

❑ ibaFOB-D (FOC link, 2 Mbit / 3 Mbit / 5 Mbit / 32 Mbit Protocol)

For each type of card, there are variants for the number of links being 1, 2 or 4 for the input and output.

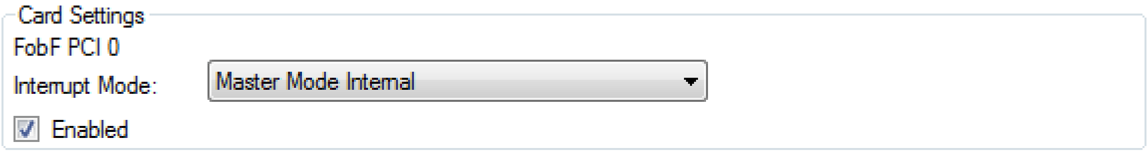
11.4.1.1 Card Settings

When you mark the FOBF*nn* or FOBD*nn* interface in the tree on the left, the associated card settings are displayed on the right.

❑ Interrupt mode, see "*Hardware resources*, page 173"

❑ Enabled, see "*Hardware resources*, page 173"

❑ Variable cycle time



Card Settings

FobF PCI 0

Interrupt Mode: Master Mode Internal

☑ Enabled

Figure 116: Card settings

Check box	Explanation
Enabled	You can use only cards that have been enabled.

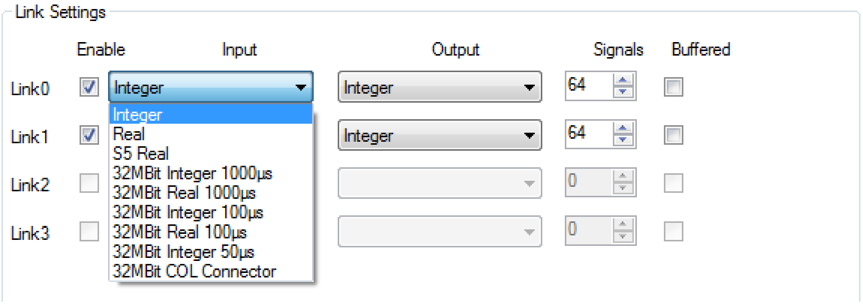
11.4.1.2 Link Settings

The channel configurations are displayed in the section on "Link settings". The number of channels varies depending on the variant of the ibaFOB-io card used.

Settings	Explanation
Enable	You can enable or disable individual links here.
Input format	Set the data format for the incoming optical fiber cable telegrams here. Select INTEGER, REAL or S5REAL depending on the devices connected. If you are using the FOB-X or the FOB-D card, other data formats for the 32 Mbit FOC protocol are provided as options.

[6] Cards for this interface are available only for old systems, however, they are not executable with Windows operating system 7/64 bit.

[7] Cards for this interface are available only for old systems, however, they are not executable with Windows operating system 7/64 bit.



Link Settings

	Enable	Input	Output	Signals	Buffered
Link0	☑	Integer	Integer	64	□
Link1	☑		Integer	64	□
Link2	□			0	□
Link3	□			0	□

Figure 117: Link settings

The data type must match the type or the setting configured on the connected device.

Device	Protocol	Data type/telegram type
ibaPADU-8 ibaNet750	3 Mbit	Integer
SIMATIC TDC/LO6	32 Mbit	32 Mbit Real 100 µs
ibaLink-SM-64-i-o	3 Mbit	Integer, Real or S5Real, depending on the switch setting on the module.
ibaLink-SM-64-SD16 ibaLink-SM-128V-i-2 ibaBM-DPM-S-64	3 Mbit	Integer or Real, depending on the setting on the module
iba-PADU-S-IT	32 Mbit	depending on the setting in ibaPADU-S-IT
ibaBM-DPM-S (Profibus) ABB AC800PEC	32 Mbit	32 Mbit Real 1000 µs
ibaLink-VME	32 Mbit	depending on the card setting

Note

Applicable to FOB-X cards: If a 32 Mbit protocol is used on the input side, the associated output link cannot be used.

Settings	Explanation
Output Format	Data type of the FOC telegram in the output direction. The setting depends on the connected device (see input format).
Signals	Here, ibaLogic enters the maximum number of signals possible depending on the selected data format. You can reduce the number of signals for your requirement. The number of signals entered applies to analog values and digital values in the input and output direction. An "FobFnnMxx" or "FobDnnMxx" module having "n" analog signals of the type configured and "n" binary signals (nn = card index, xx = link number) is created for each link enabled after you press <Accept> in the resources tree under the appropriate interface.

Buffered mode	With this, you can enable a mode for this link in which the data received is also provided as arrays. See the following description for this purpose.

11.4.2 Buffered Mode

11.4.2.1 Applications

The buffered mode is necessary for the following applications:

- ❑ The least possible interval time of ibaLogic-V5 is 1 ms. If signal values need to be sampled with a cycle time of less than 1 ms, the signal values must be recorded in buffered mode. Above all, this must be used in the context of the following modules:
  - FOB-D and FOB-X in 32 Mbit mode
  - Connection to ibaPADU-S-IT-2x16 (in I/O mode)
  - Connection of the local peripherals for the ibaPADU-S-IT-2x16 platform
- ❑ Even for peripheral signals that arrive with a cycle time of 1 ms, but which cannot be evaluated in the 1 ms interval on account of the large number of signals and the computing power required.

The values read in are buffered and provided to the program as array resources. The purpose of this mode is to relieve the program from the evaluation of the individual signals if this is not necessary.

Example:
Signals are required merely for FFT analysis. Then it is not possible (with a sampling time < 1 ms) and also not meaningful to sample individual signals and to collect them in arrays in order to perform an FFT analysis on them. The arrays are generated by the buffered mode with the proper size so that no computing time is required by the program for handling individual values.

The buffered mode has the following features:

- ❑ Data from the FOB link are acquired by the driver at a rate determined by the time base configured.
- ❑ The data sampled is collected in a cyclic buffer for each signal. The size of the cyclic buffer and the sampling rate for filling it up are configured by the program in the output resources (see below).
- ❑ Each signal is made available to the program in the form of an array with a length of 1024. The program runs in a slower task interval and reads the complete arrays.
- ❑ The program, thus, cannot process individual samples, but instead, only arrays of samples, e. g. for evaluating an FFT analysis or for archiving.
- ❑ Example: despite a sampling rate of 1 ms, and a task interval of 50 ms, you can still perform an FFT analysis with 128 samples.
- ❑ In parallel to the buffered data, individual signals can, for example, also be generated for other programs.

11.4.2.2 Input resources

An FobFBuffer*nn*M*xx* or FobDBuffer*nn*M*xx* module is created (*nn* = card index, *xx* = link number) for each link enabled in buffered mode after pressing <Apply> in the resource tree under the appropriate interface.

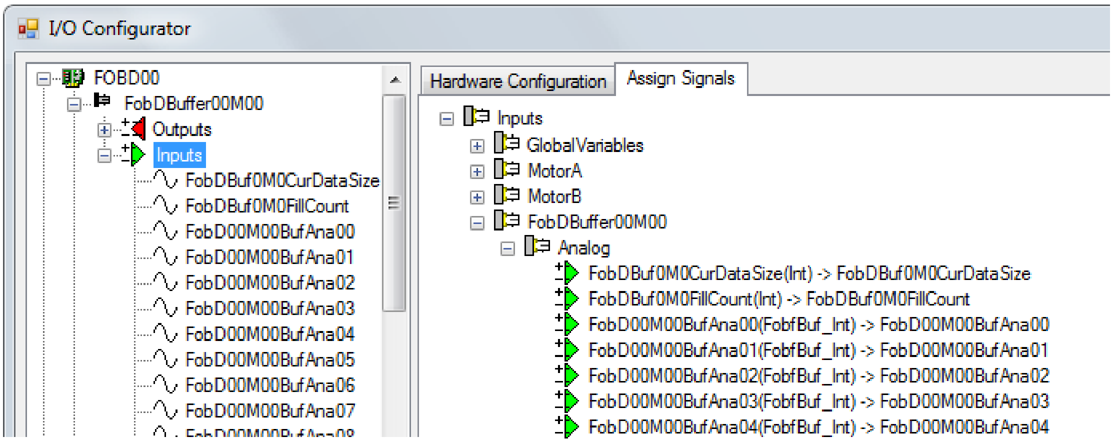

I/O Configurator
FOBD00
FobDBuffer00M00
Outputs
Inputs
FobDBuf0M0CurDataSize
FobDBuf0M0FillCount
FobD00M00BufAna00
FobD00M00BufAna01
FobD00M00BufAna02
FobD00M00BufAna03
FobD00M00BufAna04
FobD00M00BufAna05
FobD00M00BufAna06
FobD00M00BufAna07
Hardware Configuration
Assign Signals
Inputs
GlobalVariables
MotorA
MotorB
FobDBuffer00M00
Analog
FobDBuf0M0CurDataSize(Int) -> FobDBuf0M0CurDataSize
FobDBuf0M0FillCount(Int) -> FobDBuf0M0FillCount
FobD00M00BufAna00(FobfBuf_Int) -> FobD00M00BufAna00
FobD00M00BufAna01(FobfBuf_Int) -> FobD00M00BufAna01
FobD00M00BufAna02(FobfBuf_Int) -> FobD00M00BufAna02
FobD00M00BufAna03(FobfBuf_Int) -> FobD00M00BufAna03
FobD00M00BufAna04(FobfBuf_Int) -> FobD00M00BufAna04

Figure 118: Assigning input signals

Input signals	Explanation
CurDataSize	Feedback of the buffer size configured in the output signal.
FillCount	Counter that is incremented when the input array has been filled up to the length "Datasize".
BufAna*ii*	Array of type integer or real having a length of 256.
BufDig*00*	Array of type Boolean having a length of 256.

11.4.2.3 Output resources

The resources are provided in the module type FobFBuffer*nn*M*xx* or FobDBuffer*nn*M*xx* (*nn* = card index, *xx* = link index).

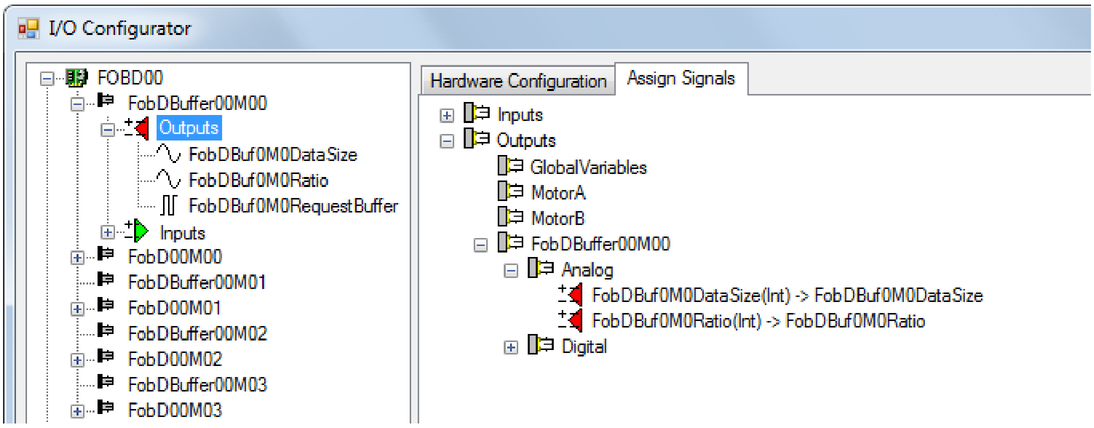


Figure 119: Assigning output resources

Output signals	Explanation
Data Size	Number of signals to be measured that are entered by the driver in the buffer until the counter is incremented ("filling level"). Values up to 256 are permissible.
Ratio	Integral multiple of the time base, with which the buffers are filled up. For example, ratio=2 means that only every 2nd measured value is entered in the buffer.
RequestBuffer	Controlling the acquisition. The buffers are filled up only if the output is "TRUE"

11.4.3 ibaLogic as Profibus Slave

The Profibus is organized strictly according to the master-slave principle. In accordance with the DP-V0 standard, communication takes place only between the master and slaves, whereby the link is set up and monitored by the master. ibaLogic can be both a Profibus master and a Profibus slave.

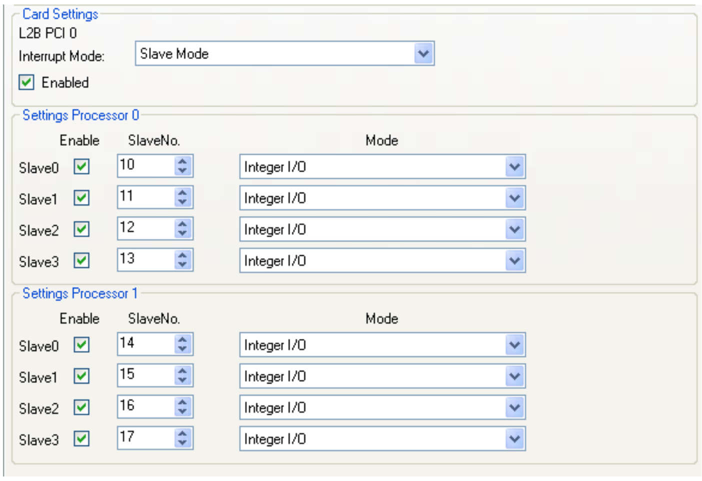
For example, for a link to a SIMATIC S7 system that is the master, ibaLogic must work as a slave. The Profibus slave card, ibaCOM-L2B is used for this purpose.

The following interface cards are arranged under the **L2B***nn* type of interface:

- ibaCOM-L2B 4/8
  (Profibus DP slave card with a jack, 4 slaves)
- ibaCOM-L2B 8/8
  (Profibus DP slave card with two jacks, 8 slaves)

11.4.3.1 Card settings

If you select the interface L2B*nn* in the left tree structure, the corresponding card settings are displayed on the right.



Card Settings
L2B PCI 0
Interrupt Mode: Slave Mode
☑ Enabled

Settings Processor 0

	Enable	SlaveNo.	Mode
Slave0	☑	10	Integer I/O
Slave1	☑	11	Integer I/O
Slave2	☑	12	Integer I/O
Slave3	☑	13	Integer I/O

Settings Processor 1

	Enable	SlaveNo.	Mode
Slave0	☑	14	Integer I/O
Slave1	☑	15	Integer I/O
Slave2	☑	16	Integer I/O
Slave3	☑	17	Integer I/O

Figure 120: Card settings ibaCom-L2B

- Interrupt mode, see "*Hardware resources*, page 173"
- Enabled, see "*Hardware resources*, page 173"

11.4.3.2 Settings for bus interface 0/1

The L2B card has one or 2 Profibus DP interfaces. You can define 4 slaves for each interface.

Settings	Explanation
Enable	You can enable or disable individual slaves here.
Slave no.	Assign a separate station number to each slave that is enabled.
Mode	Select a telegram format for each enabled slave that matches with the Profibus master configuration. For this purpose, iba provides a number of GSD files that correspond to the telegram formats available here for selection:

Telegram format	iba GSD file	Contents	Data direction
Integer_In	iba_0F01	32 integer and 32 binary signals	Master → ibaLogic
Real_In	iba_0F02	32 real and 32 binary signals	Master → ibaLogic
S7Real_In	iba_0F04	28 real and 32 binary signals	Master → ibaLogic
Integer_inOut	iba_0F08	32 integer and 32 binary signals	Master ←→ ibaLogic
Real_InOut	iba_0F09	32 real and 32 binary signals	Master ←→ ibaLogic
S7Real_InOut	iba_0F0B	28 real and 32 binary signals	Master ←→ ibaLogic

A module "L2B*nn*M*yy*S*xx*" is created for each slave enabled after pressing <Accept> in the resource tree under the appropriate interface, and this module has the analog signals of the type configured and 32 binary signals (*nn* = card index 00-03, *yy* = interface number 00-01, *xx* = slave number 00-03).

11.4.4 ibaLogic as Profibus Master

If you would like to address, for example, an ET2000 station from ibaLogic, it must work as the master. A Profibus master card must be installed in the ibaLogic PC for this purpose.

The following interface card is plugged in under the **SST_Master***nn* type of interface:

- ❑ SST-PB3-PCU (one channel, PCI)
- ❑ SST-PB3-PCU-2 (two channels, PCI)
- ❑ SST-PB3-PCIE-1 (one channel, PCI Express)
- ❑ SST-PB3-PCIE-2 (two channels, PCI Express)

Note

Using the SST card requires a license.

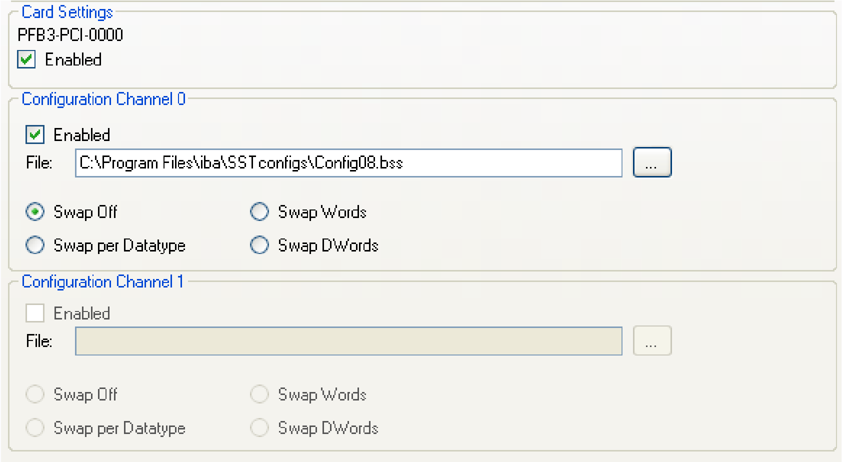
11.4.4.1 Brief Description

Since this card is the product of another manufacturer, its configuration and customization varies from the scheme for the iba cards.

In general, you must generate a configuration for the Profibus that contains the parameters required for all the stations connected to the Profibus and for the communication. The program for creating the configuration (SST Profibus Console) generates a binary parameter file (.bss) as result. This must be loaded by ibaLogic in the I/O configurator and transferred to the SST card.

11.4.4.2 Card settings

If you select the interface SST_Master*nn* in the left tree structure, the corresponding card settings are displayed on the right.



Card Settings
PFB3-PCI-0000
☑ Enabled

Configuration Channel 0
☑ Enabled
File: C:\Program Files\iba\SSTconfigs\Config08.bss ...
● Swap Off ○ Swap Words
○ Swap per Datatype ○ Swap DWords

Configuration Channel 1
□ Enabled
File: ...
○ Swap Off ○ Swap Words
○ Swap per Datatype ○ Swap DWords

Figure 121: Card settings

„Enabled"

With this option you enable or disable the card.

11.4.4.3 Configuration

"File"

This is where you specify the file generated using the CONSOLE program mentioned above. You can click on to its right to open a browser with which you can select the file in the folder structure.

"Swap modes"

Depending on the device connected, you may have to swap, i. e. interchange the high and low parts of the data type so that the data can be read in and processed in ibaLogic according to the IEC standard.

Explanation of the swap methods (each alphabet means one byte, blanks have been added only for the sake of clarity):

Swap method	Output	Becomes
per Datatype	`"ABCD EF   G H IJKL"`	`"DCBA FE G H LKJI"`
Words	`"ABCD EF   G H IJKL"`	`"BADC FE H G JILK"`
Dwords	`"ABCD EF   G H IJKL"`	`"DCBA HG F E LKJI"`

11.4.4.4 Peculiarities with signal assignment

If you have enabled the configuration settings file of the card with <Accept>, a module having the name of the SST card "PFB3-PCI-000*x*" is created in the resource tree under SST_Master. The following signals for each possible Profibus slave are visible under this:

Direction	Signal	Meaning
Input	SST*nn*InStatus*yyy*	Status and diagnostics information for telegrams received from the slave *yyy*. Data type: "SSTSTATUSSTRUCT"
Output	SST*nn*OutStatus*yyy*	Status and diagnostics information for telegrams transmitted to the slave *yyy*. Data type: "SSTSTATUSSTRUCT"
Input	SST*nn*StructIn*yyy*	Data received from the slave *yyy* Data type "SST_Struct"
Output	SST*nn*StructOut*yyy*	Data transmitted to the slave *yyy* Data type "SST_Struct"

nn = card index 00-03, *yyy* = Profibus slave number from 002-127

Now, you can generate the virtual signals for the complete module (see section "*Signal assignment*, page 183") or only for the status, input and output signals individually for the Profibus slaves that you have configured and customized.

Note

The signals generated here have the structure data types.

The status structure generated internally, "SSTSTATUSSTRUCT" consists of a number (ErrorCode) and a text string (ErrorString).

The "SST_Struct" data type is a placeholder for the structures of the user data telegrams that you have to define. This structure must match the Profibus telegram to be transmitted or received exactly, as you have defined in the Profibus configuration (Profibus Console) for each station. You can refer to the manual of the L2B cards for the structure of the Profibus telegrams (L2B).

For more information, please refer to "*Data types*, page 300".

Important Note

An example of the connection of the Profibus master card is documented and included in the CD supplied.

11.4.5 SIMADYN D / SIMATIC TDC Connection

iba has developed two PCI cards for connection to these systems and these cards differ from one another merely in the FOC interface technology and protocols.

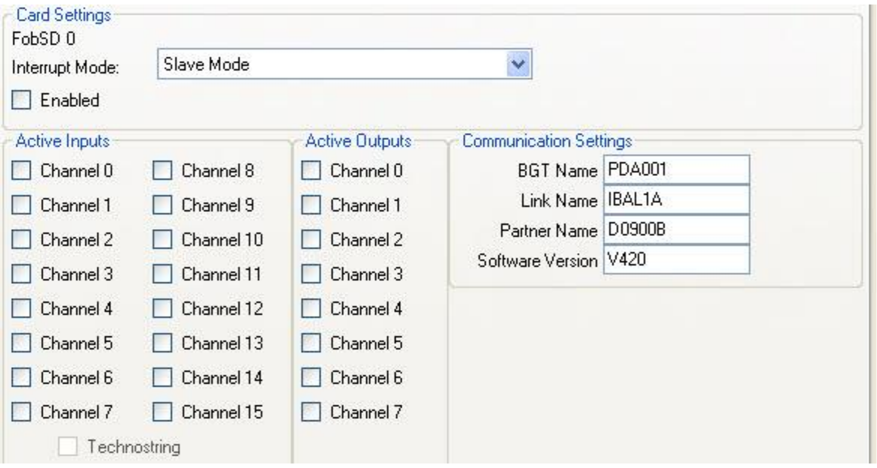
- The ibaFOB-SD card is plugged in at the **FOBSD** interface type. This enables connection to the world of SIMADYN D via the rack coupling modules CS12, CS13 and CS14 as well as to the SIMATIC TDC rack using the CP53M0 communication module.
- The ibaFOB-TDC card is plugged in at the **FOBTDC** interface type. This enables connection to the world of SIMATIC TDC via the GDM (Global Data Memory, CP52IO interface module).

Note

The settings for the FOBSD and FOBTDC modules are identical. Hence, the settings are explained here using the ibaFOB-SD card as an example.

11.4.5.1 Card settings

When you mark the FOBSD interface in the tree on the left, the associated Card Settings are displayed on the right.



Card Settings
FobSD 0
Interrupt Mode: Slave Mode
□ Enabled

Active Inputs
□ Channel 0 □ Channel 1 □ Channel 2 □ Channel 3 □ Channel 4 □ Channel 5 □ Channel 6 □ Channel 7
□ Channel 8 □ Channel 9 □ Channel 10 □ Channel 11 □ Channel 12 □ Channel 13 □ Channel 14 □ Channel 15
□ Technostring

Active Outputs
□ Channel 0 □ Channel 1 □ Channel 2 □ Channel 3 □ Channel 4 □ Channel 5 □ Channel 6 □ Channel 7

Communication Settings
BGT Name: PDA001
Link Name: IBAL1A
Partner Name: D0900B
Software Version: V420

Figure 122: Card Settings for ibaFOB-SD/TDC

- Interrupt Mode, see "*Hardware resources*, page 173"
- Enabled, see "*Hardware resources*, page 173"

11.4.5.2 Link settings

Active Inputs

Select one or more inputs here from a total of 16 channels.
Please note that for each selected input channel a transmit telegram must be available in the SIMADYN D or SIMATIC TDC.
One transmit telegram contains exactly 32 real values (Data type NF for SIMADYN D) and 32 binary values (1 DWORD or V4 value).

The following parameters must be configured on the transmitter module:

- AT (Channel name): MxPDADAT (x = 0 … F for channel 0 to 15)
- MOD (Channel mode): R (for Refresh)
- LEN (Telegram length): 132 (only for the CTV_P transmitter module)

Active Outputs

Here, you can specifically select one or more outputs from a total of 8 channels.

Please note that for each selected output channel a receive telegram must be available in the SIMADYN D or SIMATIC TDC.

A receive telegram contains exactly 32 real values (Data type NF for SIMADYN D) and 32 binary values (1 DWORD or V4 value).

The following parameters must be configured on the receiver module:

- AR (Channel name): PDAMxDAT (x = 0 … 7 for channel 0 to 7)
- MOD (Channel mode): R (for Refresh)
- LEN (Telegram length): 132 (only for the CRV_P receiver module)

An "FOBSD*nn*CH*xx*" (*nn* = card index, *xx* = channel index) module containing a total of 32 real and 32 binary signals is created for each enabled channel after pressing <Accept> in the resource tree under the appropriate interface.

Technostring

The "Technostring" function has not yet been released.

11.4.5.3 Communication Settings

- BGT Name:
  The PC must be defined to the SD/TDC environment using a name consisting if six characters, e. g. "IBA001".
- Link Name:
  The ibaFOB-SD uses this name to register with the communication partner SIMADYN D or SIMATIC TDC. This name must be unique within the CS14 or CP53M0 communication island, i. e. no other Siemens or iba modules should register with the same name. The default name is "IBAL1A".
- Partner Name:
  Enter here the name configured in the SIMADYN D or SIMATIC TDC for the communication partner module.
  For SIMADYN D, it is the name of the CS14 module, e. g. "D0500B",
  and for SIMATIC TDC (GDM), it is the name of the GDM module, usually D01_P1.
- Software Version:
  Enter here the software version of the SIMADYN D or SIMATIC TDC software: e. g. for STRUC V420", and for CFC "V610".

Note

No communication takes place if these settings are incorrect.

11.4.6 Reflective Memory

You can access VME bus-based third-party systems (e.g.: GE FANUC, Converteam HPCi) using reflective memory cards.

The following interface cards are plugged in under the RFM type of interface:

- VMIC PCI-5565PIORC:
  (Reflective memory card, 64 or 128 MByte)
- VMIC PCI-5588 among others:
  (Reflective memory cards of older design)

11.4.6.1 Brief Description

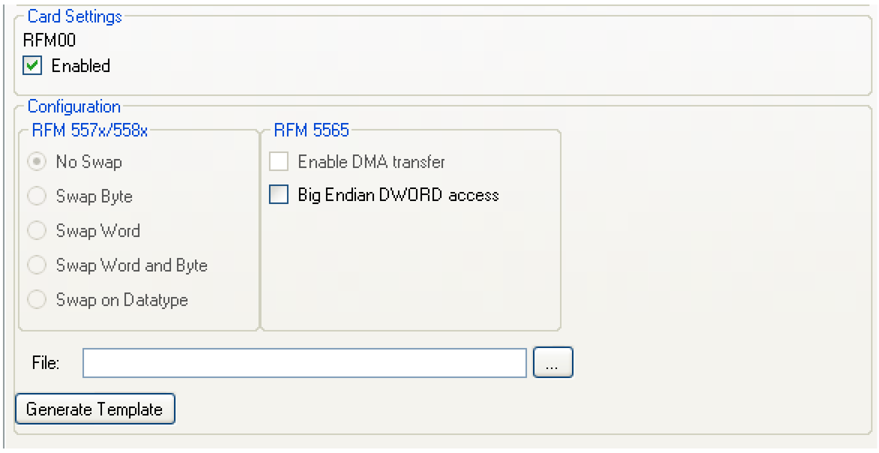
Since this card is the product of another manufacturer, its configuration and customization varies from the scheme of the iba cards.

As the memory of an RFM card does not have any homogeneous data range, but contains ranges with different data types, it is not possible to use the same parameters as for ibaFOB cards.

You have to define the position and structure of the used data in an external parameter file. This is loaded in the ibaLogic I/O configurator and signals in the desired data types are generated from there.

11.4.6.2 Card settings

If you select the RFM interface in the left tree structure, the corresponding card settings are displayed on the right.



Card Settings
RFM00
☑ Enabled

Configuration

RFM 557x/558x
● No Swap
○ Swap Byte
○ Swap Word
○ Swap Word and Byte
○ Swap on Datatype

RFM 5565
□ Enable DMA transfer
□ Big Endian DWORD access

File:

Figure 123: Reflective-Memory cards

- Enabled

With this option you enable or disable the card.

11.4.6.3 Configuration

- Swap mode

The swap mode is only available for the older RFM cards which directly support this in the hardware.
Newer cards (PCI-5565 or PCIE-5565) do no support this anymore.

Note

The methods defined here are different from those for the SST card. The designations here are taken over from the VMIC so that they match connected the RFM cards.

Explanation of the swap methods (each alphabet means one byte, blanks have been added only for the sake of clarity):

Swap mode	Explanation
Not	`"ABCD EF G H IJKL"`
Bytes	`"BADC FE H G JILK"`
Words	`"CDAB GH E F KLIJ"`
Words and bytes	`"DCBA HG F E LKJI"`
based on data type	`"DCBA FE G H LKJ"`

11.4.6.4 File

Parameter file that contains the description of the signals.

Procedure

1. Generate a template for this data using the <Generate Template> button.
2. Next, open this file using an editor.
3. Enter a line in the following format for each signal:
   "Signal name, data type, memory address, bit number, direction, comments"

Format	Explanation
Signal name	Must conform to the IEC standard and be unique, i.e. also input and output signal names have to be different.
Data type	Elementary data type (see "*Standard data types*, page 300") BOOL BYTE, WORD, DWORD SINT, USINT, INT, UINT, DINT, UDINT REAL, LREAL
Memory address	Offset within the RFM memory in decimal or hexadecimal. You can switch from one to the other using a line with "#hexval" or "#intval" at the beginning.
Bit number	Relevant only for BOOLEAN data type, otherwise 0
Direction	"INPUT" or "OUTPUT"
Comments	Any text

Example

```
#HexVal
TestSignal1,REAL,0x1000,0,INPUT, test signal input
TestSignal2,REAL,0x2000,0,OUTPUT, test signal output
#IntVal
TestBit_0,BOOL,2048,0,OUTPUT, Bit 0
TestBit_1,BOOL,2048,1,OUTPUT, Bit 1
TestBit_2,BOOL,2048,2,OUTPUT, Bit 2
```

Result

A template file is generated.

11.4.6.5 Flow of Setting Parameters

Procedure

1. Press the <Generate Template> button and enter the path and file name in order to generate a CSV file as a template.
2. Open this data using an editor and enter the signals into it.
3. Open the modified file in the I/O configurator and load the configuration in the RFM card by clicking on <Accept>.
4. Wait until the initialization phase is complete and the RFM card has been fed with these parameters.
5. Then press <Update Hardware> once again so that the signals defined are displayed in the resource tree.
6. Assign the generated signals to the virtual signal names.

11.5 ibaPADU-S-IT-2x16 Platform

The I/O configuration of the ibaPADU-S system is available only if a device of type ibaPADU-S-IT-2x16 has been selected as the platform.

ibaPADU-S-IT-2x16 is the central unit for the ibaPADU-S family of modular devices for intelligent and decentralized inputs/outputs.

The modular concept is based on a module rack having a backplane bus, to which the central unit and up to 4 other input/output modules can be plugged.

There are modules available as I/O modules for analog and digital inputs/outputs for different signal levels, for current and voltage signals and for sampling rates of up to 1 kHz (buffered access) or max. 40 kHz (unbuffered access).

Note

The current ibaPADU-S-IT-2x16 firmware for the ibaLogic version can be found in: C:\ProgramData\iba\ibaLogic5\PMAC\Firmware

Hardware Documentation

Please refer to the ibaPADU-S-IT manual for detailed information on the ibaPADU-S properties (see "*Support and contact*, page 358").

11.5.1 Settings

When you are connected with the platform and press the <Update Hardware> button, you see the following PADU-S settings.

PADU-S settings	Explanation
Interrupt source	DNS name of the ibaPADU-S-IT-2x16, e.g. "S-IT2x16-000074"
Module settings	depending on the I/O resource
Signal settings	depending on the I/O resource
I/O resources	ibaPADU-S-IT-2x16 incl. ibaPADU-S module TCPIP_OUT GLOBALVAR

Module settings

In the "Module settings" section, all available modules of the ibaPADU-S system are displayed depending on the selected I/O resources.

❑ Enabled:

You can completely disable specific modules.

❑ Buffered access:

If the buffered access is enabled, further configurations in the ibaLogic program must be performed (see ibaPADU-S-IT manual).

Only with a buffered access it is possible to reach a sampling rate of up to 20 kHz on the ibaPADU-S system.

In unbuffered access, max. 1 kHz is available (task interval = 1 ms)

❑ Convert values into REAL:

If this option is selected, the signals are acquired not as INT, but as REAL and can be processed in the program without further conversion.

Note

Depending on the module type, the "Buffered access" and "Convert values into REAL" options are not available.

Signal settings

In the "Signal settings" section, all configurable signals of the particular I/O resource are displayed depending on the selected I/O resources and corresponding module settings (see ibaPADU-S-IT-2x16 manual).

11.6 TCP/IP Communication

The following types of TCP/IP communication are available:

- General TCP/IP connection via the module (for more information, please refer to section "*TCPIP_SENDRECV*, page 107")
- ibaPDA data transmission using TCP/IP

TCP/IP communication, whose parameters can be configured in the I/O configurator, is presently limited to TCP/IP telegram transmission to the ibaPDA. In contrast to the native TCP/IP communication with the TCPIP_SENDRECV module, the module structure in the ibaPDA is supported here using a special protocol. On the Windows PC or ibaPADU-S-IT-2x16 platforms, you can transmit a total of 16 telegrams each having 32 real values and 32 digital values to one or more ibaPDA receivers.

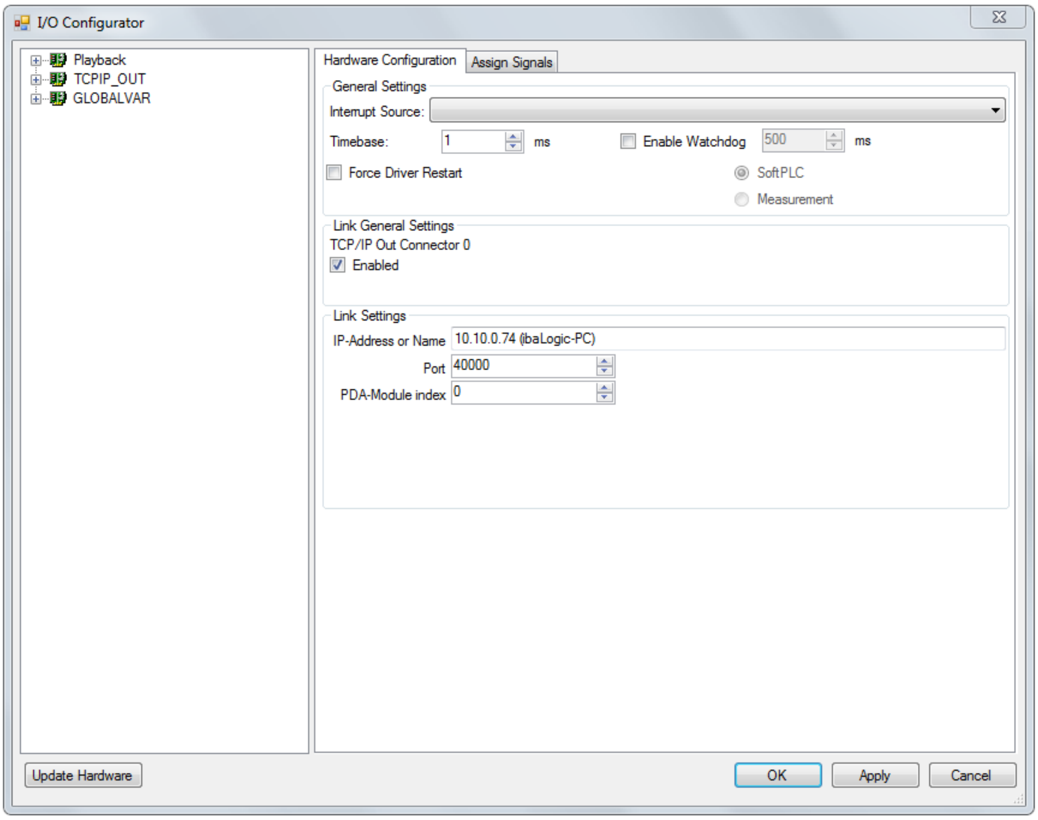
Note

Please note that the communication via TCP/IP is not real-time enabled. This means that data transmitted cyclically is not received in real time, and some telegrams may get lost etc.

11.6.1 TCP/IP Connection Settings

Select the appropriate channel from the tree on the left for the setting.

You can enable each of the 16 channels individually and enter the following parameters:



General Settings

Interrupt Source:

Timebase: 1 ms

☐ Enable Watchdog 500 ms

☐ Force Driver Restart

● SoftPLC

○ Measurement

Link General Settings

TCP/IP Out Connector 0

☑ Enabled

Link Settings

IP-Address or Name 10.10.0.74 (ibaLogic-PC)

Port 40000

PDA-Module index 0

Figure 124: TCP/IP Connection Settings

- IP address or name of the remote station, i. e. the computer on which the ibaPDA Server is running.

❑ Port number: This must match the port in ibaPDA.
Default setting is 40000.

Note

On the ibaPDA system, the configured port must be enabled in the firewall to be able to measure the data with ibaPDA.

❑ PDA module number is a unique number 0..63. This must correspond to a module index of the connected PDA system.

11.7 OPC Communication

The international OPC standard (OLE for Process Control) has prevailed for the connection to HMI systems (Human Machine Interface, control & monitoring) or for the measurement of slow signals.

11.7.1 OPC DA

11.7.1.1 OPC DA server

The ibaLogic OPC DA server provides all variables defined as "OPC visible" to the OPC DA clients, which have been connected to the OPC DA server. The OPC DA server generally runs on the same machine as the ibaLogic server and is connected with the PMAC via TCP/IP.

Note

You can also run the OPC DA server on another computer. You have to make a special inquiry with iba for this purpose.

An OPC DA client finds the ibaLogic OPC DA server under the name
“iba.Logic5OPC.1". You can select the OPC variables using their variable names with the help of a browser service if the link is established.

The OPC DA server works according to the DA (Data Access) specification V2.05a.

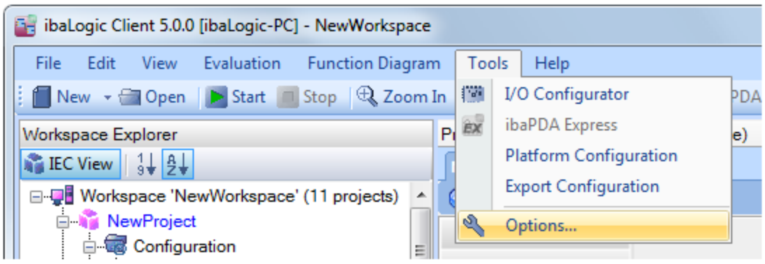
Note

You have to make a number of settings in DCOM and carry out safety guidelines for connecting the OPC DA client to the OPC DA server.

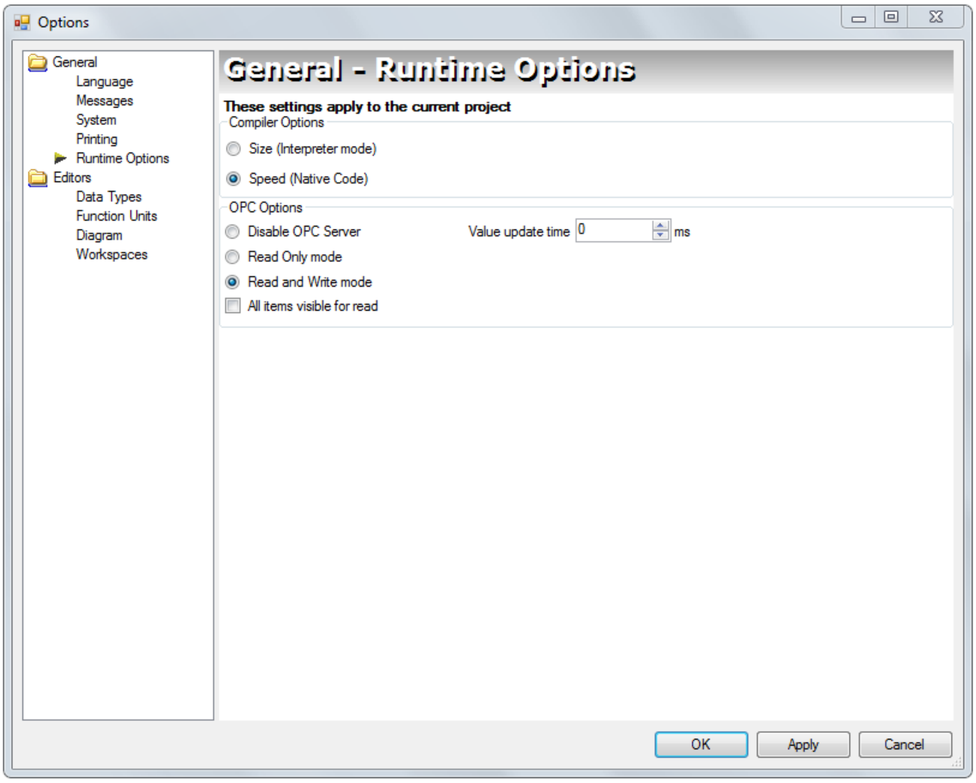
Documentation

There is a separate document for this purpose that you can get from iba on request.

1. In order to set certain OPC DA server properties, open the options of ibaLogic by using the menu item
"Tools - Options…"


ibaLogic Client 5.0.0 [ibaLogic-PC] - NewWorkspace
File Edit View Evaluation Function Diagram Tools Help
New Open Start Stop Zoom In
Workspace Explorer
IEC View
Workspace 'NewWorkspace' (11 projects)
NewProject
Configuration
I/O Configurator
ibaPDA Express
Platform Configuration
Export Configuration
Options...

2. Switch to the tree on the left and select "Runtime options".



General - Runtime Options

These settings apply to the current project

Compiler Options

○ Size (Interpreter mode)

● Speed (Native Code)

OPC Options

○ Disable OPC Server

Value update time 0 ms

○ Read Only mode

● Read and Write mode

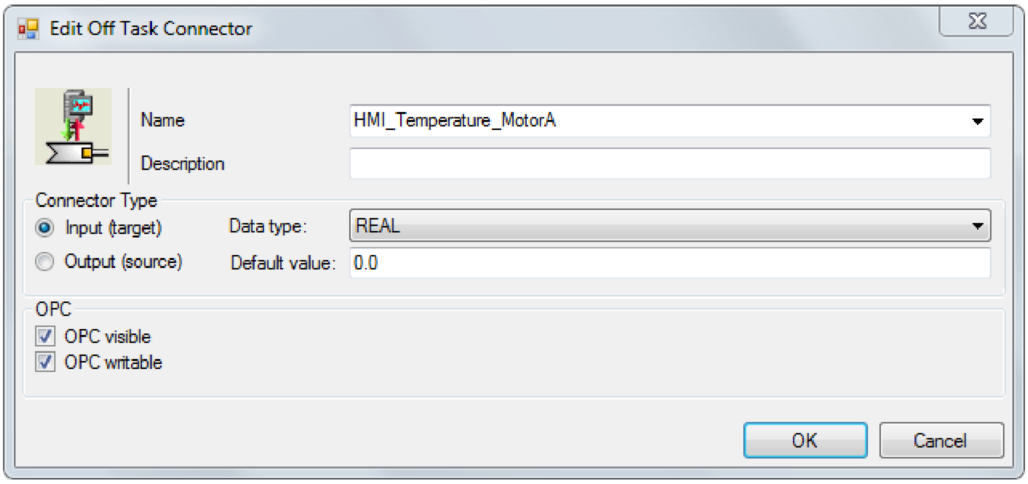
□ All items visible for read

3. Set the desired option.

Option	Explanation
Disable OPC DA server	The OPC DA server is shut down completely.
Read Only mode	Even those variables that have been defined as "OPC write-enabled" can only be read.
Read and Write mode	Default setting: access is as defined in the variable parameters.
All items visible for read	Even the variables that are not "OPC visible" can be read by the OPC DA clients but cannot be written.

11.7.1.2 Setting the OPC DA Variable Parameters

An off-task connector (OTC) must be created for each OPC signal desired in ibaLogic. The OPC selection fields must be enabled in the "Off-task connector edit" dialog box.



Edit Off Task Connector

Name: HMI_Temperature_MotorA

Description:

Connector Type

● Input (target)

○ Output (source)

Data type: REAL

Default value: 0.0

OPC

☑ OPC visible

☑ OPC writable

Figure 125: Edit Off-task connector

The data types allowed for the OPC DA connection depends on the data type of the OPC DA Client used. Normally, you can use the elementary data types and arrays.

The shortest update time is 50 ms as seen from ibaLogic, however, you must note that this depends on the data volume.

The OPC DA variables are marked in special colors. For more information on setting parameters, see "*Off-task connectors*, page 153".

You can find the OPC DA Server using the OPC DA Server browser under the name "iba.Logic5OPC.1".

11.7.2 OPC UA

OPC UA is the new standard of the OPC Foundation. It allows a simpler and faster connection than the older OPC DA standard.

ibaLogic provides an OPC UA server. Thus, data with an OPC UA client, e.g. ibaPDA are exchanged.

OPC UA data can be transmitted encrypted, but will always be signed (= sender recognition).

For the security (signing) so-called certificates are used. These are encrypted security files that are exchanged between the server and the client to allow data traffic.

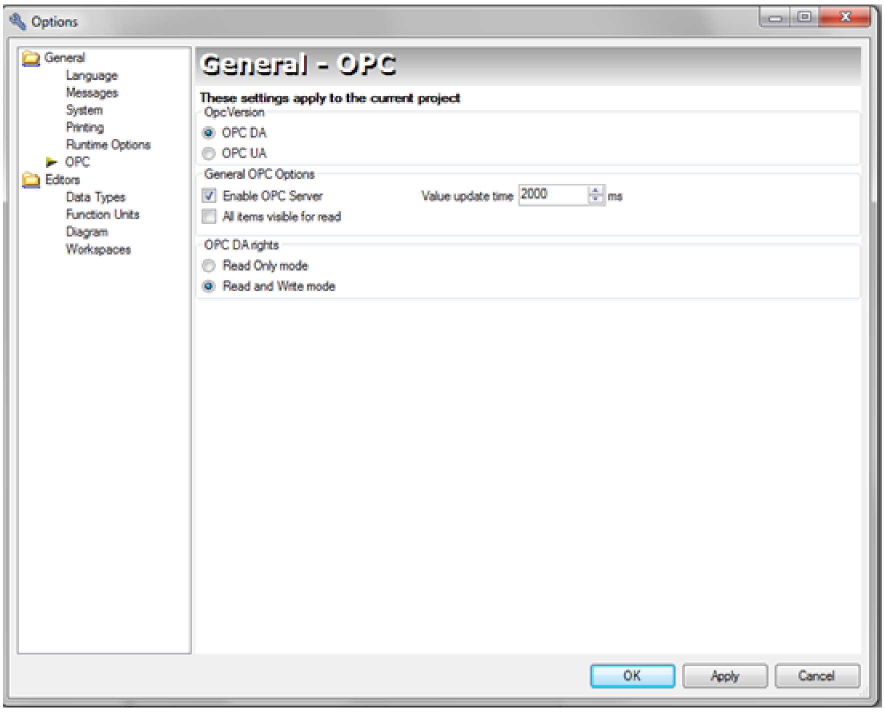
In addition to this signature, the transmitted data can then also be encrypted by various methods.

11.7.2.1 Selection between OPC UA and OPC DA Server

It is possible to choose between the previous OPC-DA servers and the new OPC-UA server. Both are not available in parallel.

The selection is made via the menu "Configuration - Options - OPC".

❑ OPC DA (previous OPC Server)



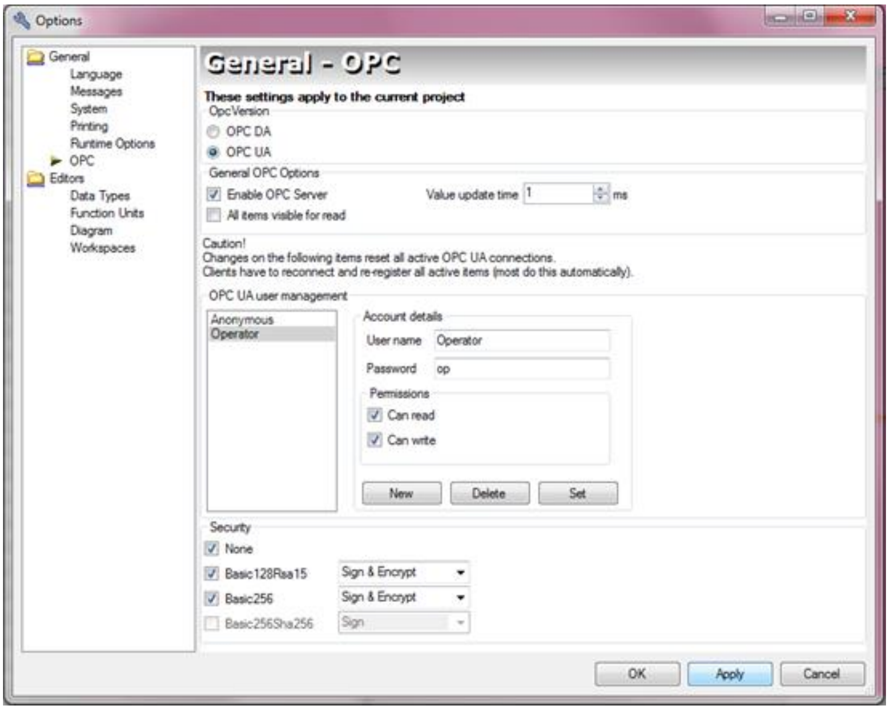
These settings apply to the current project

OpcVersion
● OPC DA
○ OPC UA

General OPC Options
☑ Enable OPC Server   Value update time 2000 ms
□ All items visible for read

OPC DA rights
○ Read Only mode
● Read and Write mode

- ❑ OPC UA (new OPC Server)



Options

General - OPC

These settings apply to the current project

OpcVersion
○ OPC DA
● OPC UA

General OPC Options
☑ Enable OPC Server Value update time 1 ms
□ All items visible for read

Caution!
Changes on the following items reset all active OPC UA connections.
Clients have to reconnect and re-register all active items (most do this automatically).

OPC UA user management
Anonymous
Operator

Account details
User name Operator
Password op

Permissions
☑ Can read
☑ Can write

Security
☑ None
☑ Basic128Rsa15 Sign & Encrypt
☑ Basic256 Sign & Encrypt
□ Basic256Sha256 Sign

- ❑ Enable OPC Server: Enable OPC UA
- ❑ All items visible for read

Not only the OTCs marked as available for OPC, but all internal variables, e.g. all inputs and outputs of a block and also internal variables of the block.

- ❑ Value update time: 20ms…x ms

Transfer time of the data. 20ms is the fastest transfer rate (depends also on the amount of data)

OPC UA user management

Any user can be created for the OPC UA connection and their rights can be set. Anonymous is a standard user without a password.

- ❑ Permissions:
  - Can read: Read access to the OPC UA data
  - Can write: Write access to the OPC UA data
- ❑ New/Delete/Set:

Create / delete a user and apply the changes.

Security: Security settings

OPC UA has different security settings. The OPC UA server (here: ibaLogic) specifies which types of security settings are allowed for a connection. Only these types can then used by an OPC UA client for a connection.

There are the security guidelines and the message security mode:

Security guidelines are:

- None: no security guidelines
- Basic128Rsa15: 128 is the bit width of the encryption and RSA15 is the encryption method used (with certificates). (RSA = Rivest, Shamir, Adleman)
- Basic256: Encryption on a bit-width of 256 (without certificates)

- Basic256Sha256: (Not yet implemented) 256 is the bit width of the encryption and SHA256 is the encryption method (with certificates) (SHA = Secure Hash Algorithm)

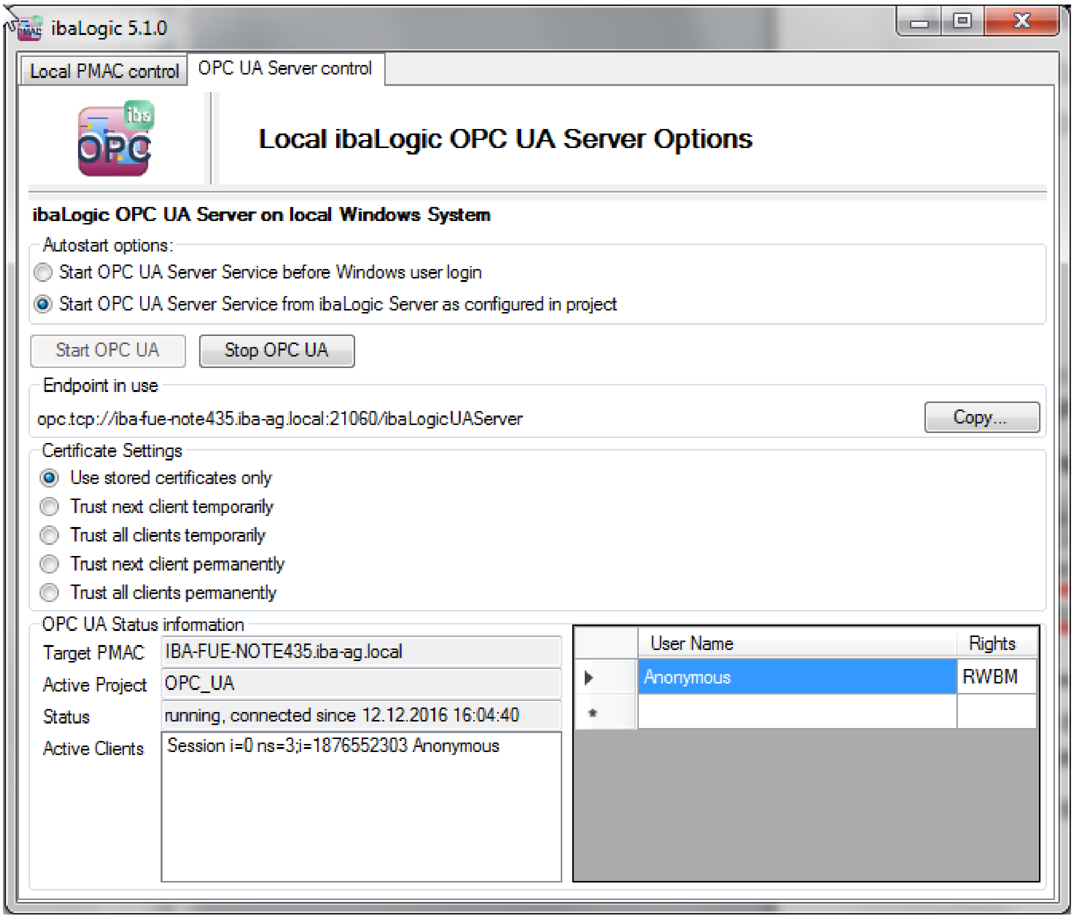
Message-SecurityMode: (Depending on selected safety guidelines)

- None: nothing
- Sign: The message transmission is signed so that the sender is clear
- Sign&Encrypt: The message is encrypted in addition to "Sign"
- Sign, Sign&Encrypt: allows all encryption, so only "Sign" as also "Sign&Encrypt".

11.7.2.2 OPC UA Server

The OPC UA server in ibaLogic is available as a service.

The settings are made via the PMAC status mask.



ibaLogic 5.1.0

Local PMAC control | OPC UA Server control

Local ibaLogic OPC UA Server Options

ibaLogic OPC UA Server on local Windows System

Autostart options:

○ Start OPC UA Server Service before Windows user login

● Start OPC UA Server Service from ibaLogic Server as configured in project

Start OPC UA | Stop OPC UA

Endpoint in use

opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer Copy...

Certificate Settings

● Use stored certificates only

○ Trust next client temporarily

○ Trust all clients temporarily

○ Trust next client permanently

○ Trust all clients permanently

OPC UA Status information

Target PMAC: IBA-FUE-NOTE435.iba-ag.local

Active Project: OPC_UA

Status: running, connected since 12.12.2016 16:04:40

Active Clients: Session i=0 ns=3;i=1876552303 Anonymous

	User Name	Rights
▶	Anonymous	RWBM
*		

Figure 126: OPC UA Server options

❑ Autostart options: Here you define when to start the service

❑ Manual Start/Stop:

❑ Endpoint in use:

The OPC UA endpoint is displayed. This endpoint is important for the connection to the OPC UA client.

With the button <Copy…>, the end point is copied to the clipboard and can easily be inserted into the OPC UA client.

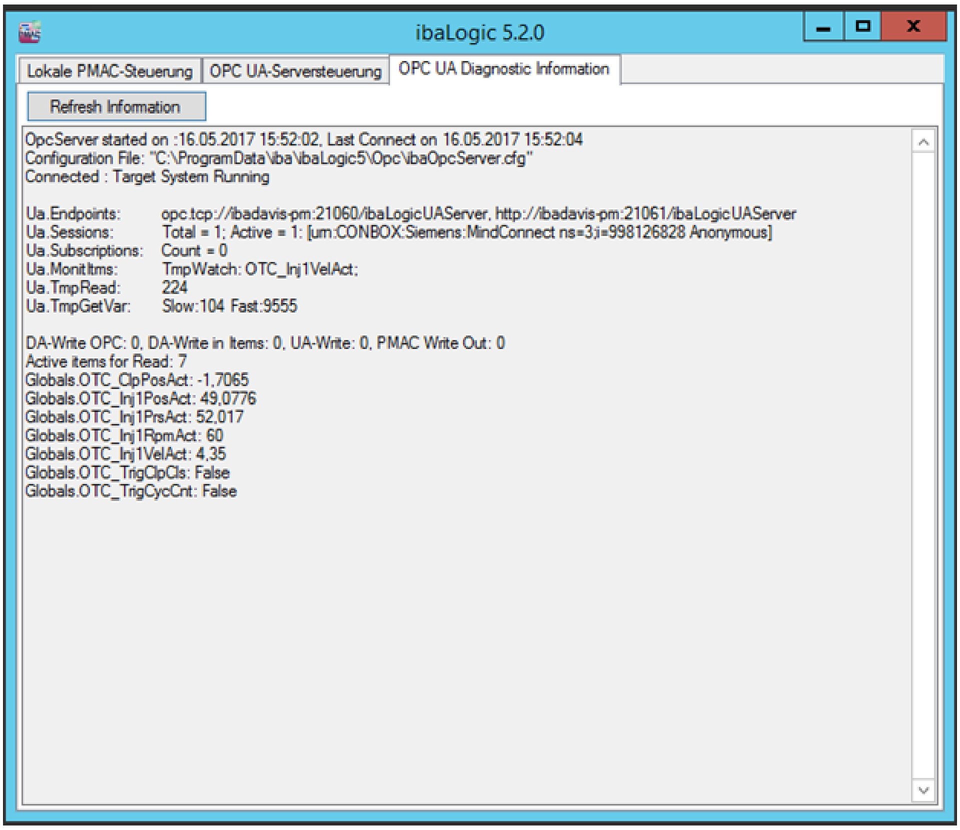
Example: opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer

❑ Ceritficate Settings:
(see the following: exchange of certificates)

❑ OPC UA Status information:
Display of connected OPC UA clients with users.

OPC UA Diagnosis

Diagnostic information is displayed in the "OPC UA Diagnostic Information" tab. With the button <Refresh Information> the data is updated..



```
OpcServer started on :16.05.2017 15:52:02, Last Connect on 16.05.2017 15:52:04
Configuration File: "C:\ProgramData\iba\ibaLogic5\Opc\ibaOpcServer.cfg"
Connected : Target System Running

Ua.Endpoints:      opc.tcp://ibadavis-pm:21060/ibaLogicUAServer, http://ibadavis-pm:21061/ibaLogicUAServer
Ua.Sessions:       Total = 1; Active = 1: [urn:CONBOX:Siemens:MindConnect ns=3;i=998126828 Anonymous]
Ua.Subscriptions:  Count = 0
Ua.MonitItms:      TmpWatch: OTC_Inj1VelAct;
Ua.TmpRead:        224
Ua.TmpGetVar:      Slow:104 Fast:9555

DA-Write OPC: 0, DA-Write in Items: 0, UA-Write: 0, PMAC Write Out: 0
Active items for Read: 7
Globals.OTC_ClpPosAct: -1,7065
Globals.OTC_Inj1PosAct: 49,0776
Globals.OTC_Inj1PrsAct: 52,017
Globals.OTC_Inj1RpmAct: 60
Globals.OTC_Inj1VelAct: 4,35
Globals.OTC_TrigClpCls: False
Globals.OTC_TrigCycCnt: False
```

Figure 127: OPC UA Diagnostic information

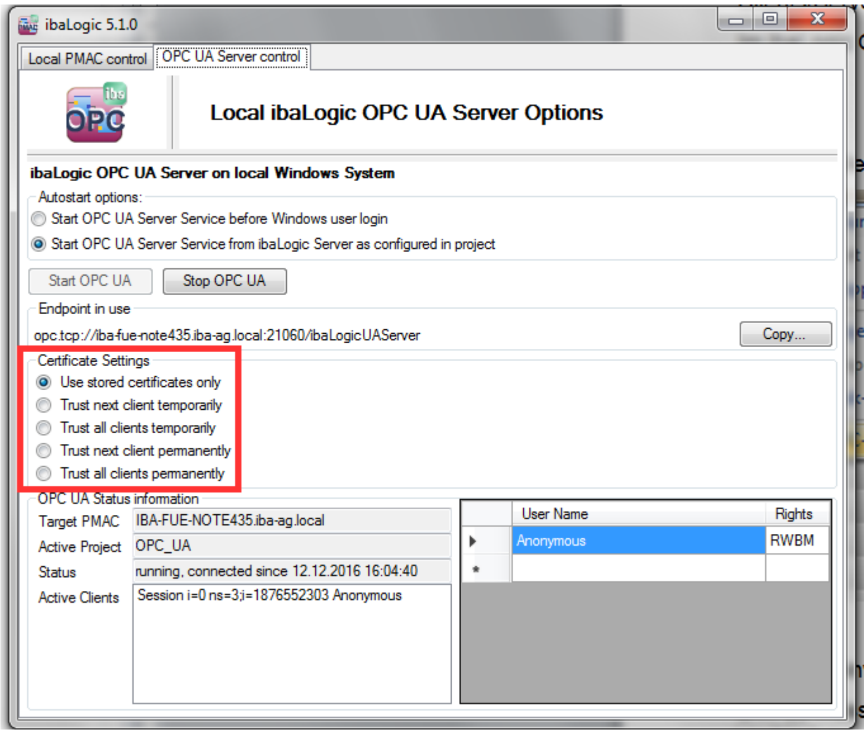
11.7.2.3 Exchange of certificates

Since secure OPC UA connections only work via certificates, they must be exchanged. For this purpose, each OPC UA server and OPC UA client has a directory for placing its own certificate and that of its connection partners so that a connection can be made securely.

This exchange of the certificates can be done manually by copying. Most OPC UA clients and servers also offer an automated option.

In the ibaLogic PMAC Control and also directly in the ibaLogic Client, the following options are selectable.

- ibaLogic PMAC Control:



ibaLogic 5.1.0

Local PMAC control | OPC UA Server control

Local ibaLogic OPC UA Server Options

ibaLogic OPC UA Server on local Windows System

Autostart options:

○ Start OPC UA Server Service before Windows user login

● Start OPC UA Server Service from ibaLogic Server as configured in project

Start OPC UA | Stop OPC UA

Endpoint in use

opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer

Copy...

Certificate Settings

● Use stored certificates only

○ Trust next client temporarily

○ Trust all clients temporarily

○ Trust next client permanently

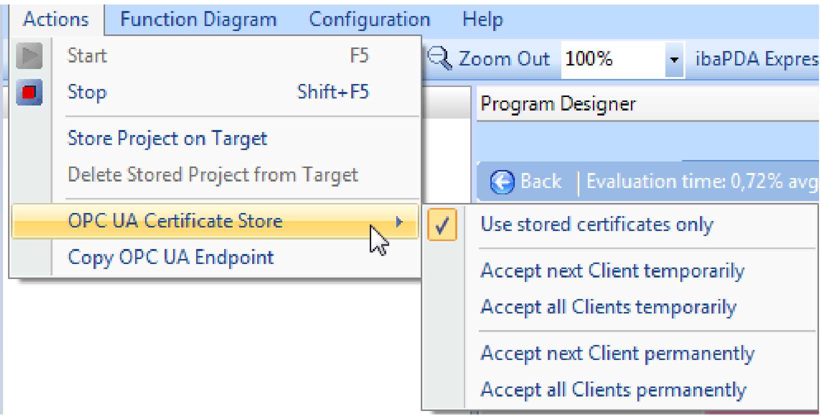
○ Trust all clients permanently

OPC UA Status information

Target PMAC	IBA-FUE-NOTE435.iba-ag.local
Active Project	OPC_UA
Status	running, connected since 12.12.2016 16:04:40
Active Clients	Session i=0 ns=3;i=1876552303 Anonymous

User Name	Rights
Anonymous	RWBM

- ibaLogic Client:


Actions
Function Diagram
Configuration
Help
Start F5
Stop Shift+F5
Store Project on Target
Delete Stored Project from Target
OPC UA Certificate Store
Copy OPC UA Endpoint
Zoom Out
100%
ibaPDA Expres
Program Designer
Back
Evaluation time: 0,72% avg
✓ Use stored certificates only
Accept next Client temporarily
Accept all Clients temporarily
Accept next Client permanently
Accept all Clients permanently

The submenu offers options for exchanging certificates. The option "Use stored certficates only" is selected by default. The necessary certificates have to exist on both sides, otherwise no connection is established.

If you now want to connect an OPC UA client to ibaLogic, you must select one of the following 4 options before making a connection setup from an OPC UA client.

- Accept next Client temporarily:
  When an OPC UA client attempts to establish a connection to the ibaLogic OPC UA server, its certificate is classified as "safe" and stored with the ibaLogic OPC UA server. Afterwards, the selection automatically reverts to "Do not trust". However, as soon as you close this OPC connection, these certificates are deleted again.

- Accept all Clients temporarily:
  All attempts made by one or more OPC UA clients to establish a connection to the ibaLogic OPC UA server is classified as "safe" and its certificates are stored with the ibaLogic OPC UA server. However, as soon as you close this OPC connection, these certificates are deleted again.

- ❑ Accept next Client permanently:
  When an OPC UA client attempts to establish a connection to the ibaLogic OPC UA server, its certificate is classified as "safe" and stored with the ibaLogic OPC UA server. Afterwards, the selection automatically reverts to "Do not trust". This connection can then be set up again and again without the need for any further certificates to be exchanged.
- ❑ Accept all Clients permanently:
  All attempts made by one or more OPC UA clients to establish a connection to the ibaLogic OPC UA server are classified as "safe" and their certifications are stored with the ibaLogic OPC UA Server. The certificates are retained even when the connection is disconnected.

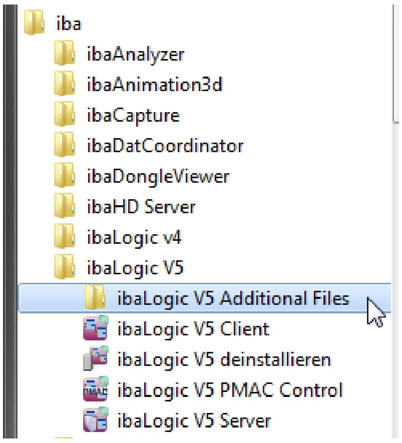
Important note

Make sure that a "Accept all Clients permanently" setting can also be a security risk, as all incoming connections are running during the ibaLogic OPC UA server, are automatically classified as secure, and a permanent certificate for each of these incoming connections is stored. This allows these clients to connect without problems after restarting. This mode should therefore only be used for test purposes.

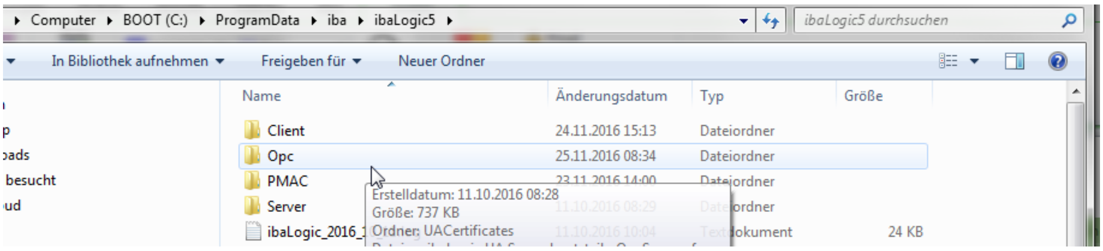
11.7.2.4 Certificate storage and control

ibaLogic stores all certificates in a separate folder. This can be found as follows:

Via"Start - All programs – iba - ibaLogic V5 - ibaLogic V5 Additional Files" …



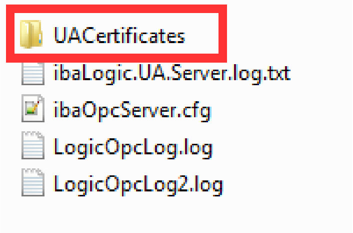
you will get to this directory and select the folder "OPC".



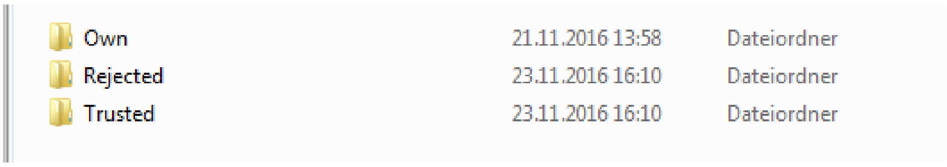
Name	Änderungsdatum	Typ	Größe
Client	24.11.2016 15:13	Dateiordner	
Opc	25.11.2016 08:34	Dateiordner	
PMAC	23.11.2016 14:00	Dateiordner	
Server	11.10.2016 08:29	Dateiordner	
ibaLogic_2016_1	11.10.2016 10:04	Textdokument	24 KB

Erstelldatum: 11.10.2016 08:28
Größe: 737 KB
Ordner: UACertificates

Select the folder „UACertificates":



UACertificates
ibaLogic.UA.Server.log.txt
ibaOpcServer.cfg
LogicOpcLog.log
LogicOpcLog2.log



Own	21.11.2016 13:58	Dateiordner
Rejected	23.11.2016 16:10	Dateiordner
Trusted	23.11.2016 16:10	Dateiordner

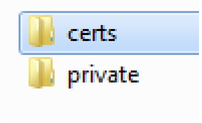
The following folders are located here:

- ❑ Own: own certificates from ibaLogic
- ❑ Rejected: rejected certificates of unauthorized connections.
- ❑ Trusted: allowed and accepted certificates

Important note

If you delete these folders, all OPC UA connections are no longer valid.

Under "Own", there are the subfolders:



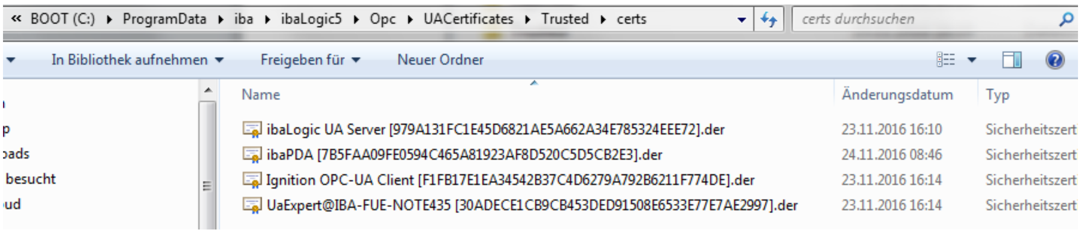
certs
private

There is always a private and a public certificate for OPC UA. The private can not be passed on. The public (here, under "certs") is that which is sent to the opponent, so that a connection can be established if it is accepted by the opponent.

You can also copy this certificate simply at the opposing place into the appropriate "Trusted" directory and thus it is also a permitted connection.

And also you can enter an originally rejected connection (see the folder "Rejected") by moving to the "Trusted" folder as a valid connection.

Examples of "trusted" certificates:



« BOOT (C:) ▸ ProgramData ▸ iba ▸ ibaLogic5 ▸ Opc ▸ UACertificates ▸ Trusted ▸ certs
certs durchsuchen
In Bibliothek aufnehmen ▾
Freigeben für ▾
Neuer Ordner
p
ɔads
besucht
ud

Name	Änderungsdatum	Typ
ibaLogic UA Server [979A131FC1E45D6821AE5A662A34E785324EEE72].der	23.11.2016 16:10	Sicherheitszert
ibaPDA [7B5FAA09FE0594C465A81923AF8D520C5D5CB2E3].der	24.11.2016 08:46	Sicherheitszert
Ignition OPC-UA Client [F1FB17E1EA34542B37C4D6279A792B6211F774DE].der	23.11.2016 16:14	Sicherheitszert
UaExpert@IBA-FUE-NOTE435 [30ADECE1CB9CB453DED91508E6533E77E7AE2997].der	23.11.2016 16:14	Sicherheitszert

Important note

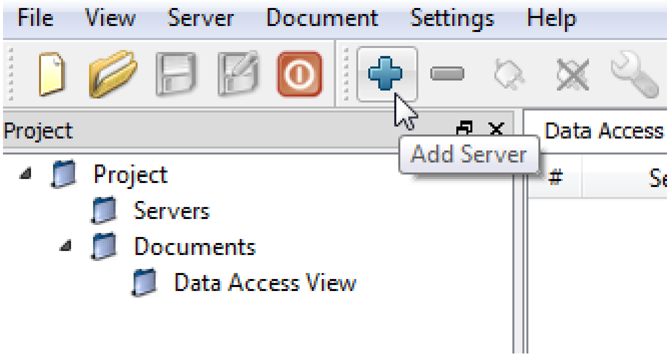
All certificates have a runtime and are invalid after the runtime. The ibaLogic certificate is set for a term of 50 years. In the next versions, corresponding display options and handling for the expiry warning will be included here.

11.7.2.5 Example: Establishing a connection to the UAExpert client

The following describes how to set up a connection via the freely available OPC UA client "UAExpert".

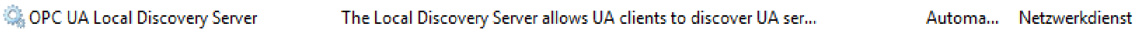
Procedure:

1. First you have to integrate the OPC UA server from ibaLogic with a click on <+>.



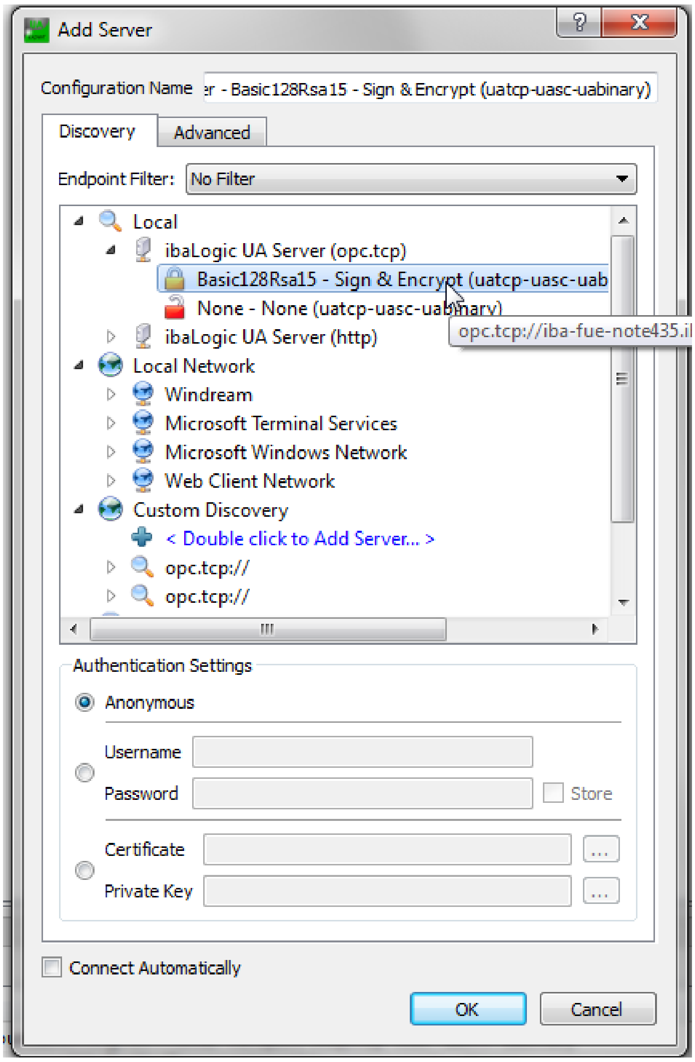
2. If the ibaLogic UA server is not found, check whether
   - ibaLogic is online and the OPC Server of ibaLogic is runnung, i.e. the OPC window is visible
   - the Discovery service is started

For searching the existing OPC UA server the OPC UA Discovery Server must be existing and running:



OPC UA Local Discovery Server   The Local Discovery Server allows UA clients to discover UA ser...   Automa...   Netzwerkdienst

Here you can also see what kind of access ibaLogic works with (Basic128Rsa15).



Add Server

Configuration Name: er - Basic128Rsa15 - Sign & Encrypt (uatcp-uasc-uabinary)

Discovery | Advanced

Endpoint Filter: No Filter

- Local
  - ibaLogic UA Server (opc.tcp)
    - Basic128Rsa15 - Sign & Encrypt (uatcp-uasc-uab
    - None - None (uatcp-uasc-uabinary)
  - ibaLogic UA Server (http)
- Local Network
  - Windream
  - Microsoft Terminal Services
  - Microsoft Windows Network
  - Web Client Network
- Custom Discovery
  - < Double click to Add Server... >
  - opc.tcp://
  - opc.tcp://

opc.tcp://iba-fue-note435.il

Authentication Settings

● Anonymous

○ Username: ___ Password: ___ □ Store

○ Certificate: ___ ... Private Key: ___ ...

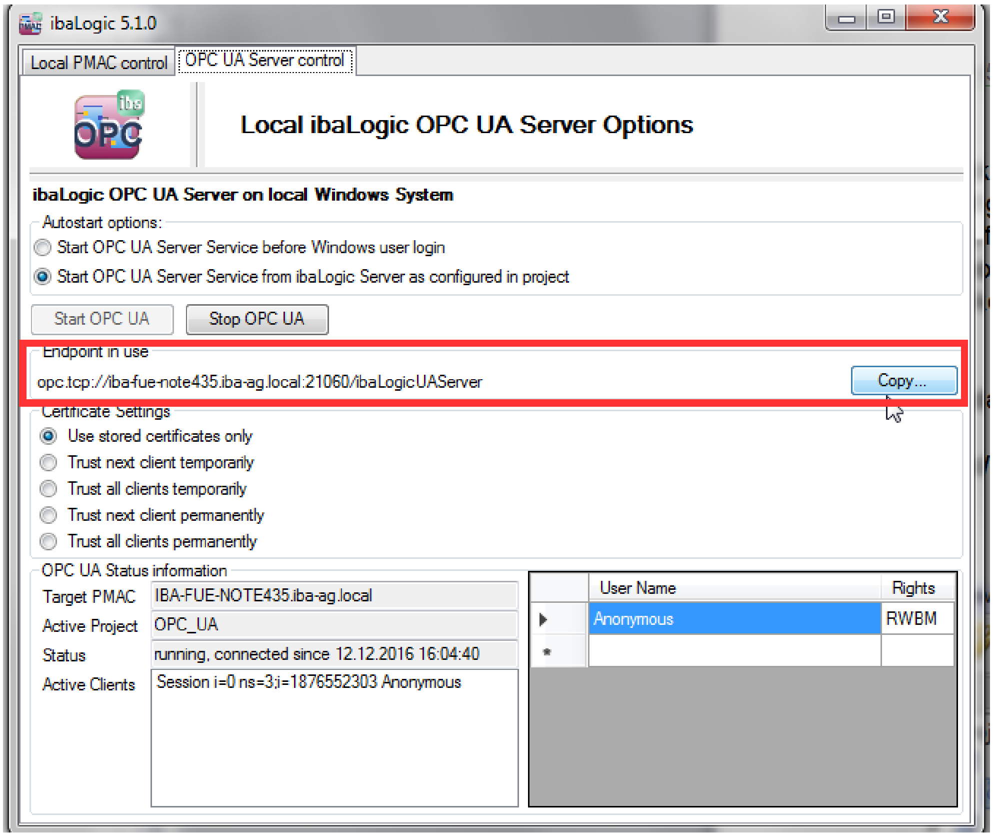
□ Connect Automatically

OK | Cancel

Double-click on the selected connection set this URL which is necessary for the connection automatically and closes the dialog.

The URL can also be found under *Advanced* and could also be copied from the OPC UA server of ibaLogic.

Click the button <Copy> in order to copy the entry in the ibaLogic OPC UA Server.



ibaLogic 5.1.0

Local PMAC control | OPC UA Server control

Local ibaLogic OPC UA Server Options

ibaLogic OPC UA Server on local Windows System

Autostart options:

○ Start OPC UA Server Service before Windows user login

● Start OPC UA Server Service from ibaLogic Server as configured in project

Start OPC UA | Stop OPC UA

Endpoint in use

opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer Copy...

Certificate Settings

● Use stored certificates only

○ Trust next client temporarily

○ Trust all clients temporarily

○ Trust next client permanently

○ Trust all clients permanently

OPC UA Status information

Target PMAC: IBA-FUE-NOTE435.iba-ag.local

Active Project: OPC_UA

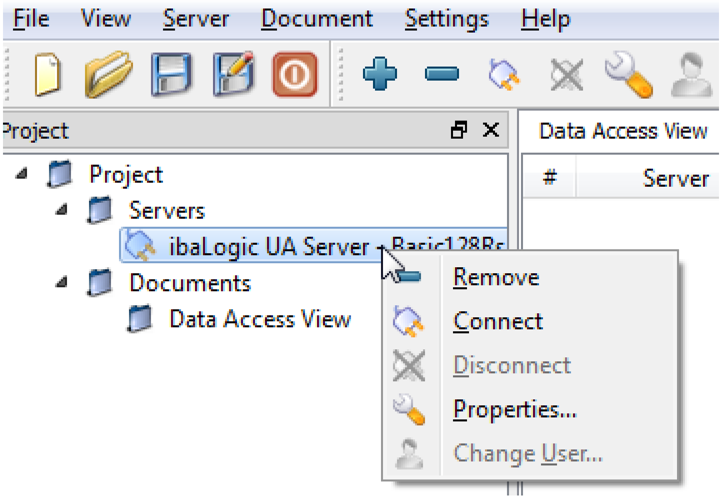
Status: running, connected since 12.12.2016 16:04:40

Active Clients: Session i=0 ns=3;i=1876552303 Anonymous

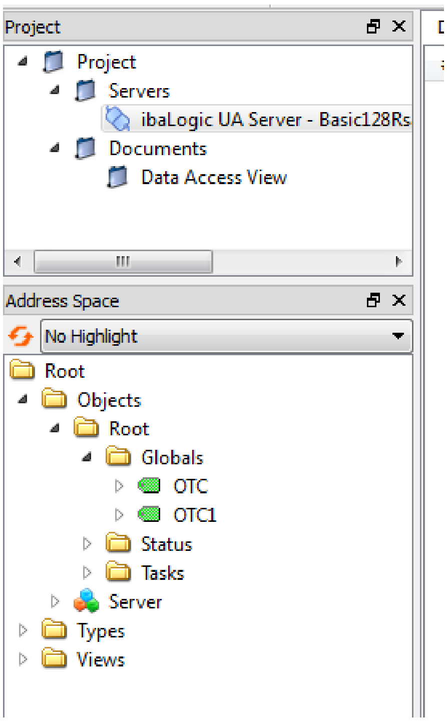
User Name	Rights
Anonymous	RWBM

The connection address in the example is:

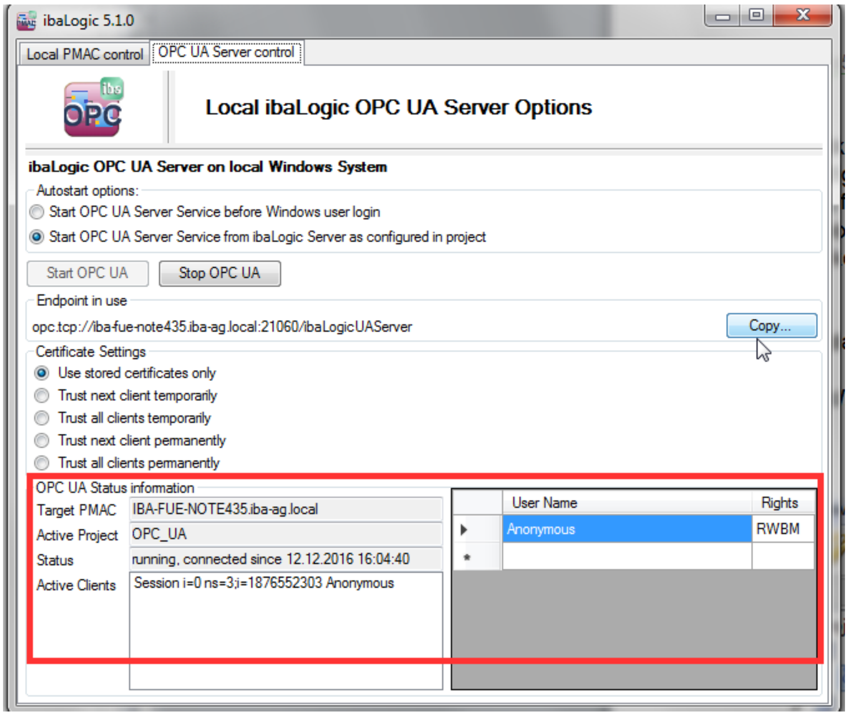
opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer



Select "Connect" from the context menu to establish the connection to the ibaLogic OPC UA Server. The access to the variables appears in the window below.



In the ibaLogic OPC UA Server display in the PMAC Control the currently logged in user is displayed.



Local ibaLogic OPC UA Server Options

ibaLogic OPC UA Server on local Windows System

Autostart options:

○ Start OPC UA Server Service before Windows user login

● Start OPC UA Server Service from ibaLogic Server as configured in project

Endpoint in use

opc.tcp://iba-fue-note435.iba-ag.local:21060/ibaLogicUAServer

Certificate Settings

● Use stored certificates only

○ Trust next client temporarily

○ Trust all clients temporarily

○ Trust next client permanently

○ Trust all clients permanently

OPC UA Status information

Target PMAC: IBA-FUE-NOTE435.iba-ag.local

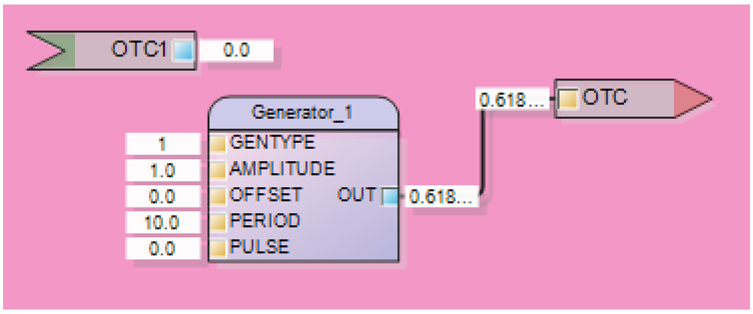
Active Project: OPC_UA

Status: running, connected since 12.12.2016 16:04:40

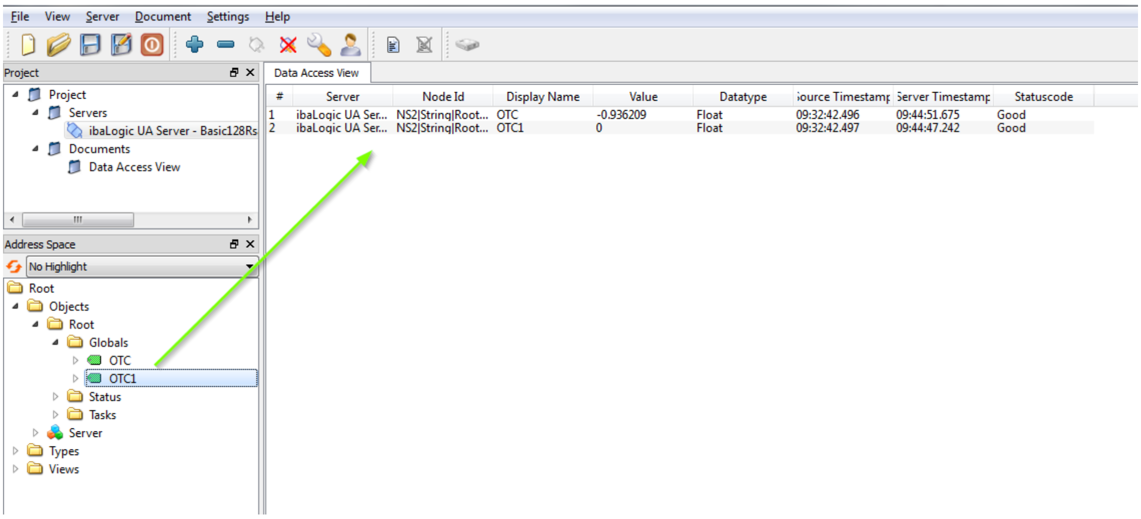
Active Clients: Session i=0 ns=3;i=1876552303 Anonymous

User Name	Rights
Anonymous	RWBM

If you want to look at the data,

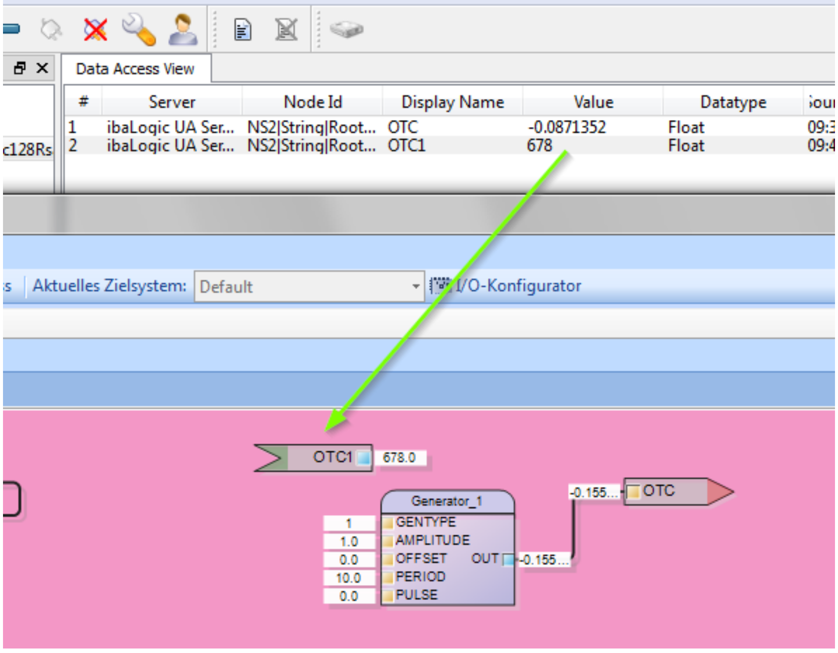


then you can find the variables in the UAExpert and can use them by drag & drop.



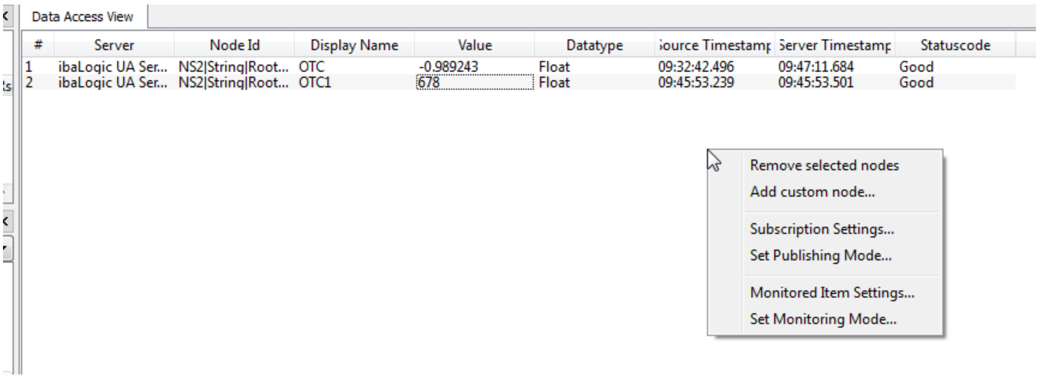
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	ibaLogic UA Ser...	NS2|String|Root...	OTC	-0.936209	Float	09:32:42.496	09:44:51.675	Good
2	ibaLogic UA Ser...	NS2|String|Root...	OTC1	0	Float	09:32:42.497	09:44:47.242	Good

Input values can be entered directly in the "Value" field. If necessary, the user must be given write permission.



#	Server	Node Id	Display Name	Value	Datatype	Sour
1	ibaLogic UA Ser...	NS2|String|Root...	OTC	-0.0871352	Float	09:3
2	ibaLogic UA Ser...	NS2|String|Root...	OTC1	678	Float	09:4

Update time of data

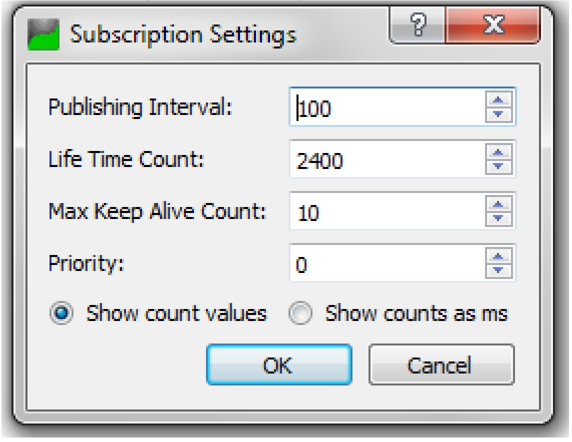


Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	ibaLogic UA Ser...	NS2|String|Root...	OTC	-0.989243	Float	09:32:42.496	09:47:11.684	Good
2	ibaLogic UA Ser...	NS2|String|Root...	OTC1	678	Float	09:45:53.239	09:45:53.501	Good

With the right mouse button this menu is displayed. The most important menu items are:

- ❑ Subscription Settings



Subscription Settings

Publishing Interval: 100

Life Time Count: 2400

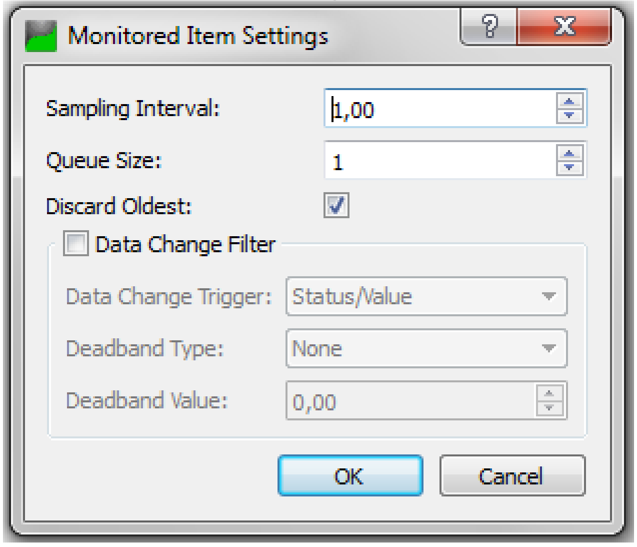
Max Keep Alive Count: 10

Priority: 0

● Show count values ○ Show counts as ms

A publishing interval is possible up to 100 ms.

- ❑ Monitored Item Settings



Monitored Item Settings

Sampling Interval: 1,00

Queue Size: 1

Discard Oldest: ☑

□ Data Change Filter

Data Change Trigger: Status/Value

Deadband Type: None

Deadband Value: 0,00

The sampling interval in ms, here e.g. 1 ms.

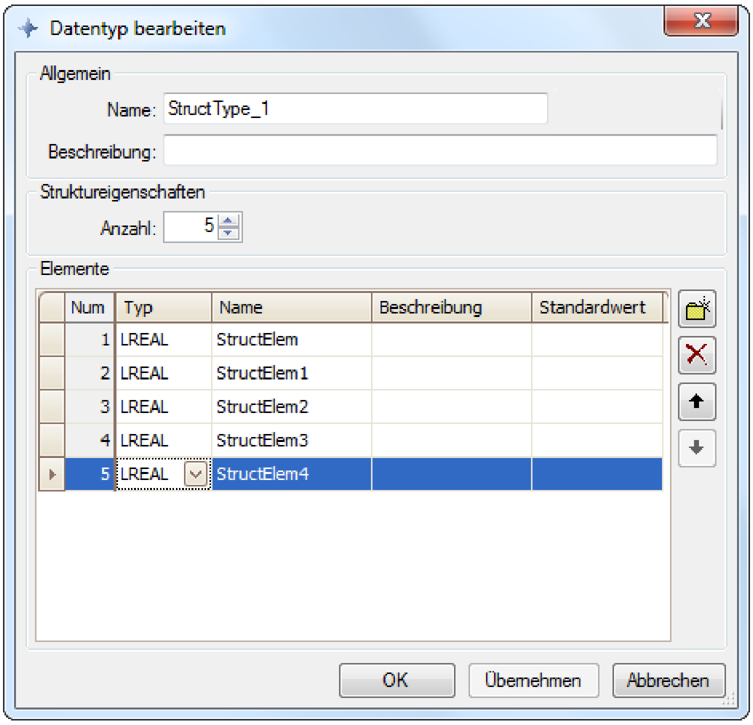
The queue size indicates how many values are received in the sampling interval. This is similar to buffered values: An array of data is sent at all x intervals.

This function is supported by the ibaLogic OPC UA server and allows HMI systems to display more detailed curves, despite a slower pick-up rate.

Data types: (Examples)

The following is an example of arrays and structures:

OTC1 is a structure in ibaLogic.



Datentyp bearbeiten

Allgemein

Name: StructType_1

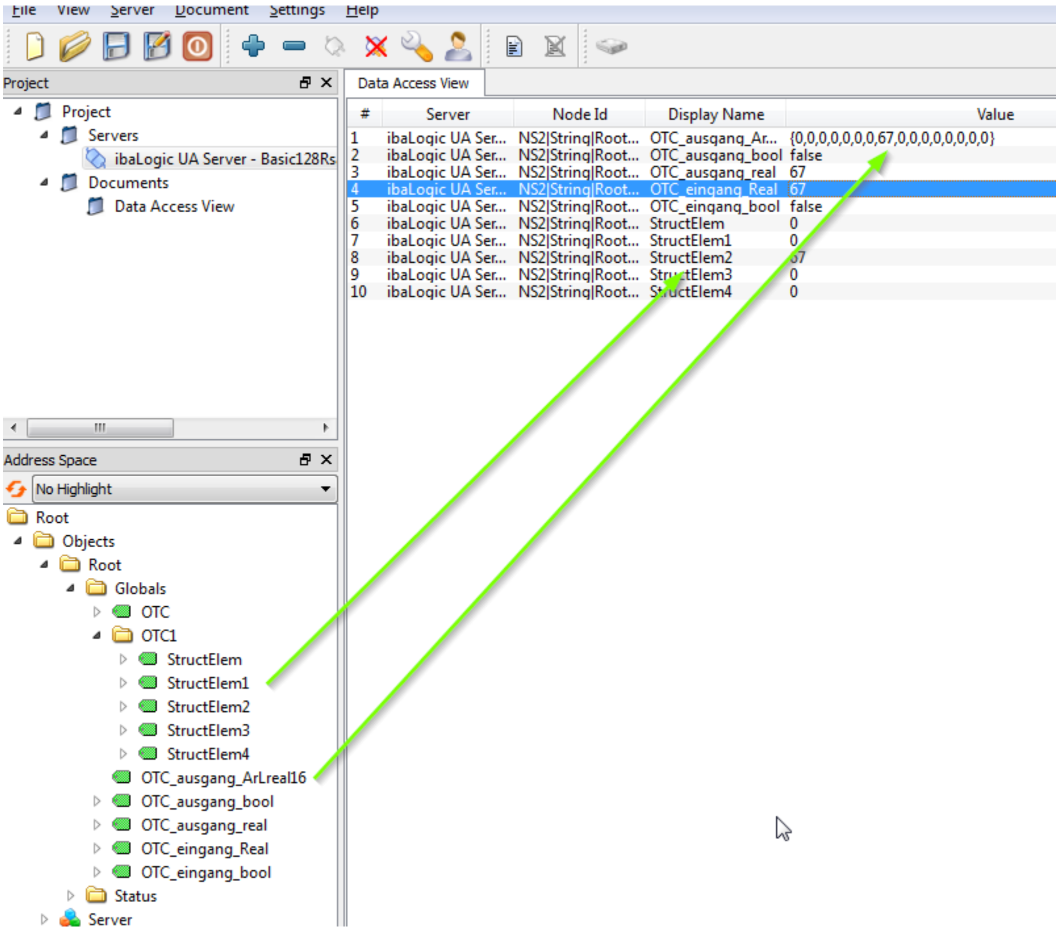
Beschreibung:

Struktureigenschaften

Anzahl: 5

Elemente

Num	Typ	Name	Beschreibung	Standardwert
1	LREAL	StructElem		
2	LREAL	StructElem1		
3	LREAL	StructElem2		
4	LREAL	StructElem3		
5	LREAL	StructElem4		



#	Server	Node Id	Display Name	Value
1	ibaLogic UA Ser...	NS2|String|Root...	OTC_ausgang_Ar...	{0,0,0,0,0,0,0,67,0,0,0,0,0,0,0,0}
2	ibaLogic UA Ser...	NS2|String|Root...	OTC_ausgang_bool	false
3	ibaLogic UA Ser...	NS2|String|Root...	OTC_ausgang_real	67
4	ibaLogic UA Ser...	NS2|String|Root...	OTC_eingang_Real	67
5	ibaLogic UA Ser...	NS2|String|Root...	OTC_eingang_bool	false
6	ibaLogic UA Ser...	NS2|String|Root...	StructElem	0
7	ibaLogic UA Ser...	NS2|String|Root...	StructElem1	0
8	ibaLogic UA Ser...	NS2|String|Root...	StructElem2	67
9	ibaLogic UA Ser...	NS2|String|Root...	StructElem3	0
10	ibaLogic UA Ser...	NS2|String|Root...	StructElem4	0

12 Database Management

ibaLogic is a database-based application. In order to save intermediate results, you must backup the database regularly. ibaLogic-V5 provides support for this purpose by offering the options of automatic or manual backup.

12.1 Backup Database

You should always consider database backup before making comprehensive modifications to it.

12.1.1 Manual Database Backup

Backup always saves the complete and current ibaLogic database. This database may have several workspaces. You can see the workspaces that currently exist in the database in the client by selecting the "Open Workspace" menu option.

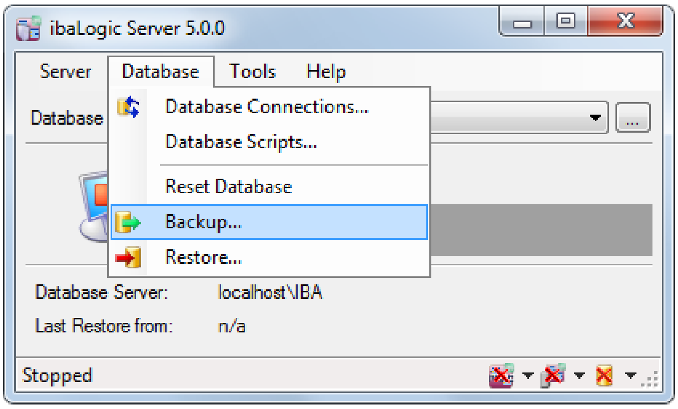
Prerequisite

❑ You have opened the ibaLogic Server dialog box.

❑ You have a connection to the database.

Procedure

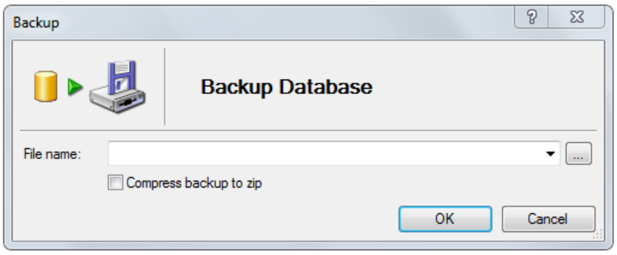
1. Select the "Database - Backup…" menu.



Database Server: localhost\IBA

Last Restore from: n/a

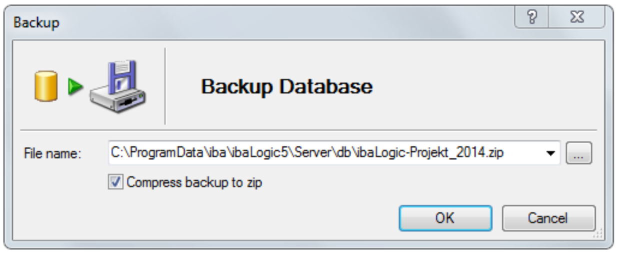
The following dialog box opens:



File name:

☐ Compress backup to zip

2. Choose a file name and a folder in which the backup file should be saved.
Enable the option "Compress backup to zip", if the hardware configuration of the I/O manager and all DLLs present should also be saved.

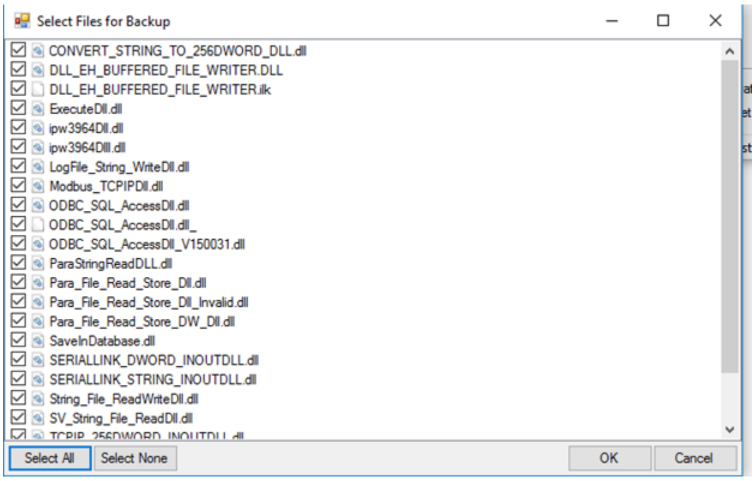

Backup

Backup Database

File name: C:\ProgramData\iba\ibaLogic5\Server\db\ibaLogic-Projekt_2014.zip

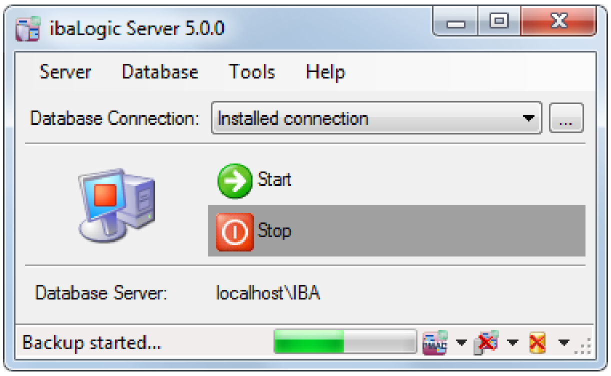
☑ Compress backup to zip

3. A dialog opens, where the DLLs can be selected, which shall be saved in the backup file. This means that those DLLs can be specifically saved, which belong to the project and may be used at a later time.


Select Files for Backup

☑ CONVERT_STRING_TO_256DWORD_DLL.dll
☑ DLL_EH_BUFFERED_FILE_WRITER.DLL
☑ DLL_EH_BUFFERED_FILE_WRITER.ilk
☑ ExecuteDll.dll
☑ ipw3964Dll.dll
☑ ipw3964Dlll.dll
☑ LogFile_String_WriteDll.dll
☑ Modbus_TCPIPDll.dll
☑ ODBC_SQL_AccessDll.dll
☑ ODBC_SQL_AccessDll.dll_
☑ ODBC_SQL_AccessDll_V150031.dll
☑ ParaStringReadDLL.dll
☑ Para_File_Read_Store_Dll.dll
☑ Para_File_Read_Store_Dll_Invalid.dll
☑ Para_File_Read_Store_DW_Dll.dll
☑ SaveInDatabase.dll
☑ SERIALLINK_DWORD_INOUTDLL.dll
☑ SERIALLINK_STRING_INOUTDLL.dll
☑ String_File_ReadWriteDll.dll
☑ SV_String_File_ReadDll.dll
☑ TCPIP_256DWORD_INOUTDLL.dll

4. Click on <OK>.

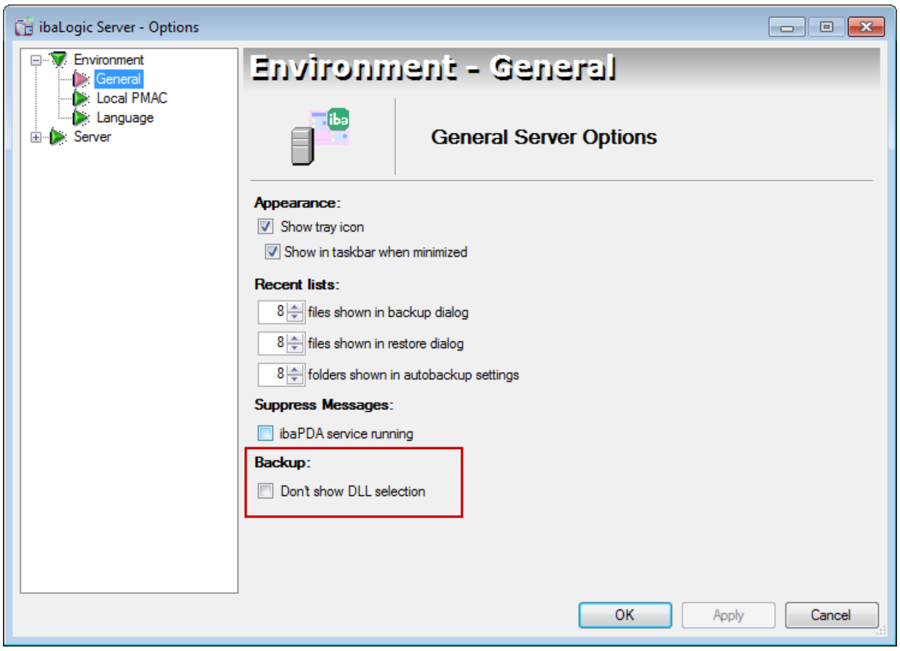

ibaLogic Server 5.0.0

Database Connection: Installed connection

Database Server: localhost\IBA

Backup started...

The manual selection of the DLLs can be enabled or disabled in the general server options.



Appearance:

☑ Show tray icon

☑ Show in taskbar when minimized

Recent lists:

8 files shown in backup dialog

8 files shown in restore dialog

8 folders shown in autobackup settings

Suppress Messages:

□ ibaPDA service running

Backup:

□ Don't show DLL selection

Important Note

It is particularly recommended to backup the database before updating the ibaLogic version.

In the course of further advanced development of ibaLogic, during updates of ibaLogic, modifications are also made to the database with the help of database scripts. It is then no longer possible to revert to an older version.

12.1.2 Backing up the database automatically

You have to configure the setting in ibaLogic for automatic database backup.

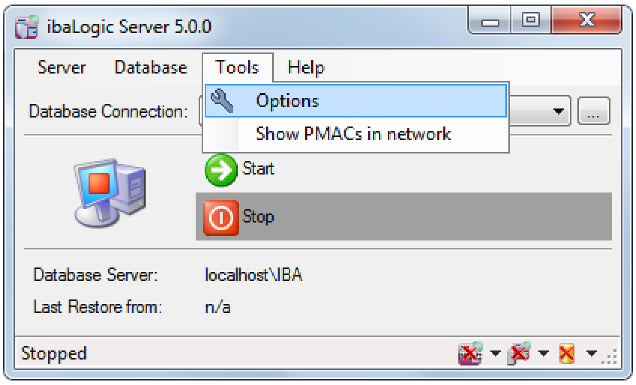
Requirement

❑ You have opened the ibaLogic Server dialog box.

❑ You have a connection to the database.

Procedure

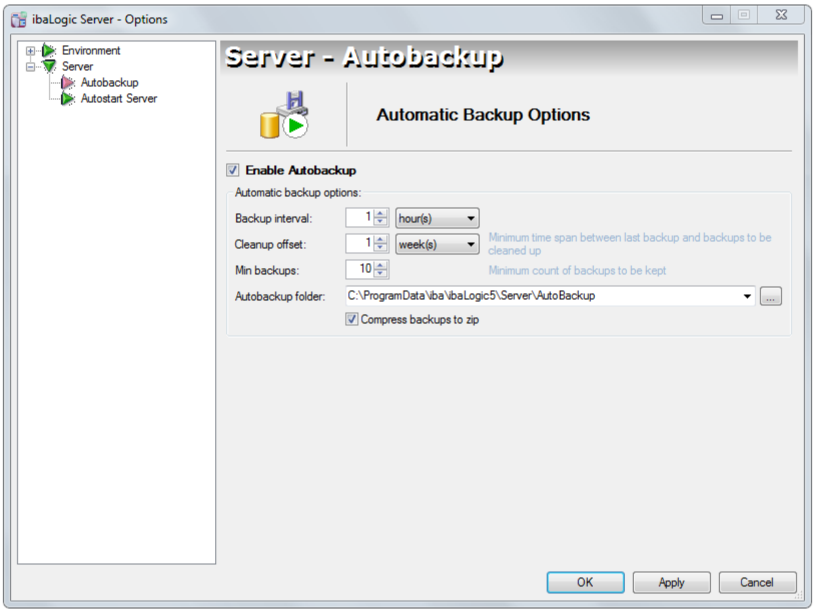
1. Select the "Tools - Options" menu.



Database Server: localhost\IBA

Last Restore from: n/a

2. Select "Server - Autobackup" in the folder tree.



☑ Enable Autobackup

Automatic backup options:

Backup interval: 1 hour(s)

Cleanup offset: 1 week(s) Minimum time span between last backup and backups to be cleaned up

Min backups: 10 Minimum count of backups to be kept

Autobackup folder: C:\ProgramData\iba\ibaLogic5\Server\AutoBackup

☑ Compress backups to zip

3. Tick the checkbox "Enable Autobackup ".
4. Select the time period for the backup interval.

Since modifications to ibaLogic projects can be made only with ibaLogic Clients, the interval is started only when a client is connected to the server. If a modification has been detected in the database, a new backup in the form of a *.bak or *.zip file is created after the interval time has elapsed.

Important Note

Specify different folders for automatic and manual backup respectively. Since the cleanup strategy cleans up only the folder specified for automatic backup, you can thus prevent your manual backup copies from being deleted.

The cleanup strategy is determined by the combination of the

- ❑ "Time until cleanup" (Cleanup offset)
- ❑ "Number of backup copies"

fields.

Option	Explanation
Backup interval	The setting creates a backup at the time interval specified.
Cleanup offset	The setting creates backup copies until the minimum time span of the backup and cleanup offset is reached.
Min. backups	The option determines the minimum number of backup copies that remain at all times. The time period " Cleanup offset " is taken into consideration in the process.
Autobackup folder	The file names are assigned by ibaLogic: *"Autobackup_ibaLogic5_<Date, Time>.bak"* or *"…..zip"*. Date and time are defined in YYYYMMDDHHMM format.
Compress backups to ZIP	The option saves the backup as a ZIP file.

Example

If you have the settings as illustrated in the window given above, all backup copies that are older than one week are deleted. However, at least 10 files remain. These can be of any date.

12.2 Restore Database

Restore means that a previous version of the database with its workspaces will be loaded for editing.

Prerequisite

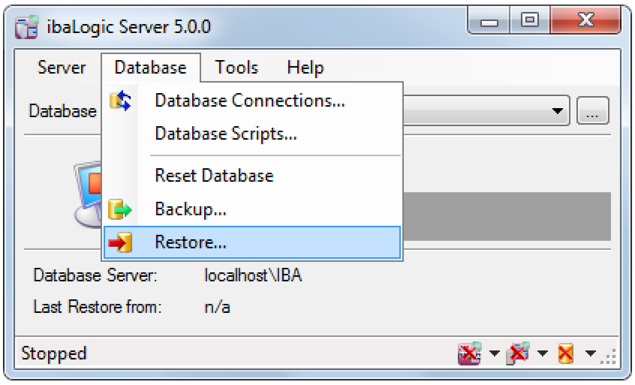
❑ You have opened the ibaLogic Server dialog box.

❑ You have a connection to the database.

❑ You have stopped the ibaLogic Server.

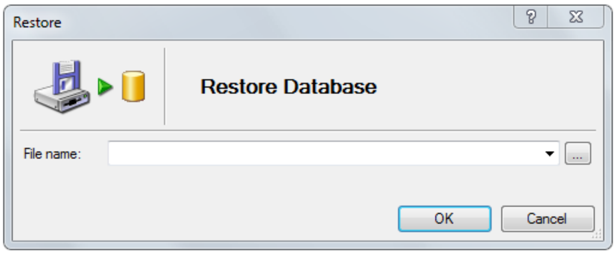
Procedure

1. Select the "Database - Restore…" menu.



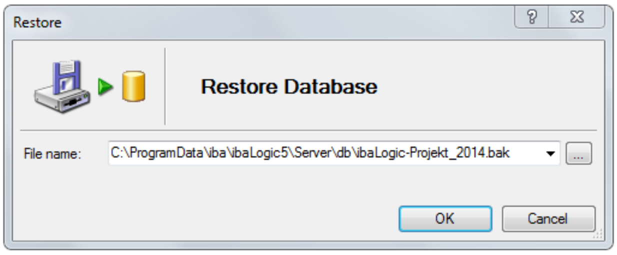
Database Server: localhost\IBA
Last Restore from: n/a

The "Restore Database" dialog is displayed.



2. Click the <…...> button and select a backup database (ZIP or BAK file) from the folder that is open.

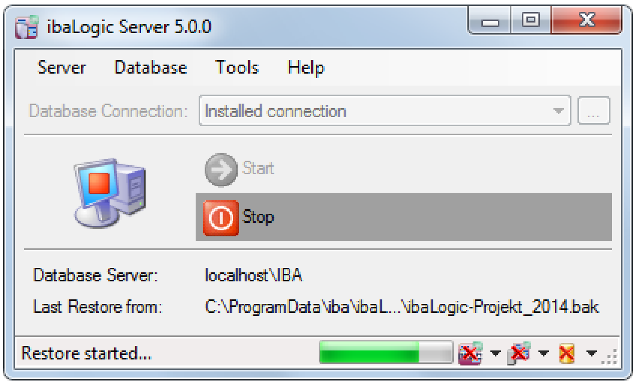
By default, ibaLogic provides the folder given below or notes the path to which the last access was made.



File name: C:\ProgramData\iba\ibaLogic5\Server\db\ibaLogic-Projekt_2014.bak

3. Click the <OK> button to restore the backup copy.

The progress of the backup process is displayed on the screen. The server then changes to "stopped" state.



Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: C:\ProgramData\iba\ibaL...\ibaLogic-Projekt_2014.bak

Restore started...

4. If required, confirm any confirmation prompts that pop up.
5. Start the server for continuing the programming work via the client.

12.3 Reset Database

You can use this function to reset your current database to its original state (empty).

Important Note

This also deletes all data in the database.

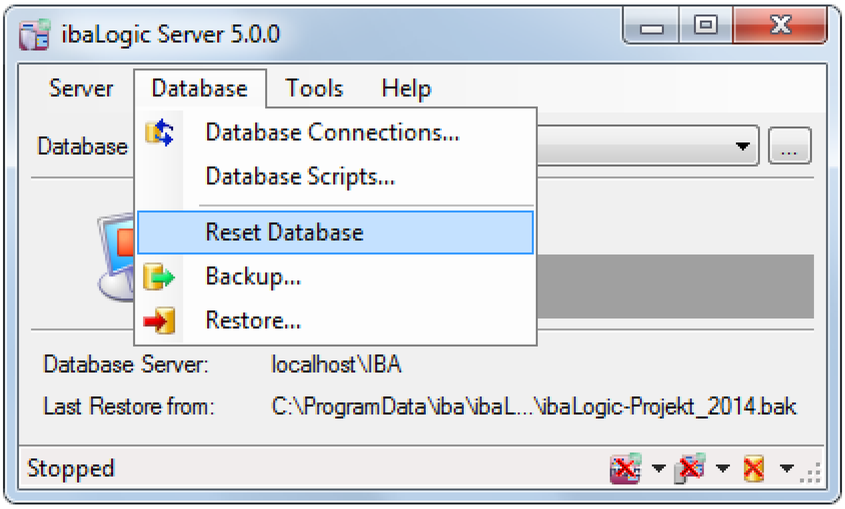
First, make a backup copy of the database.

Prerequisite

- ❑ You have opened the ibaLogic Server dialog box.
- ❑ You have a connection to the database.
- ❑ You have stopped the ibaLogic Server.

Procedure

1. Select the "Database – Reset Database......" menu.



Database Server: localhost\IBA

Last Restore from: C:\ProgramData\iba\ibaL...\ibaLogic-Projekt_2014.bak

Stopped

2. Confirm the dialog with OK if you really want to reset the database.

13 Program analysis, debugging and time response

You have various methods and tools available for program analysis and debugging.

You need to differentiate between whether the application is in the test environment or is already actively in use.

Description	Test environment	Active use
Trace function blocks or Log DLLs created by the user (e. g. LogFile_String_WriteDll.dll, ……)	Yes	Conditional (time response)
Analysis of the time response, curve shape, etc. with **ibaPDA Express**	Yes	Yes
Writing of iba data files for ibaAnalyzer using the **DAT_FILE_WRITE function block**	Yes	Yes
Examine the event window of Windows for notifications	Yes	No

Note

If a problem that can be ascribed to an error of ibaLogic-V5 should occur, it is helpful for our support if you create a support file.
You can generate this through the HELP menu on the ibaLogic Server.
Choose "*Save information for the iba Support*, page 48".
Send this file with notes about the problem to "*Support*, Page 358".

13.1 Invalid identification

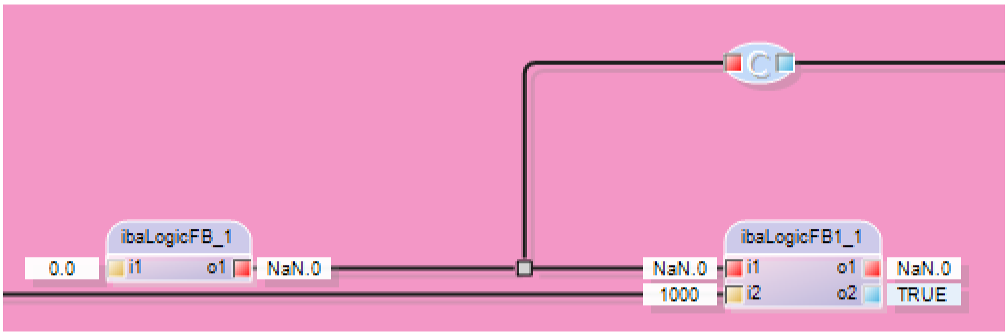
If an evaluation should recognize an error in a function unit, the associated outputs of the function unit will be marked with an ‘invalid’ color code. This will also spread to possible following function units. This code is non-storing i.e. as soon as the error is eliminated, the color code will disappear. In certain cases, a notification is placed in the event window at the same time.

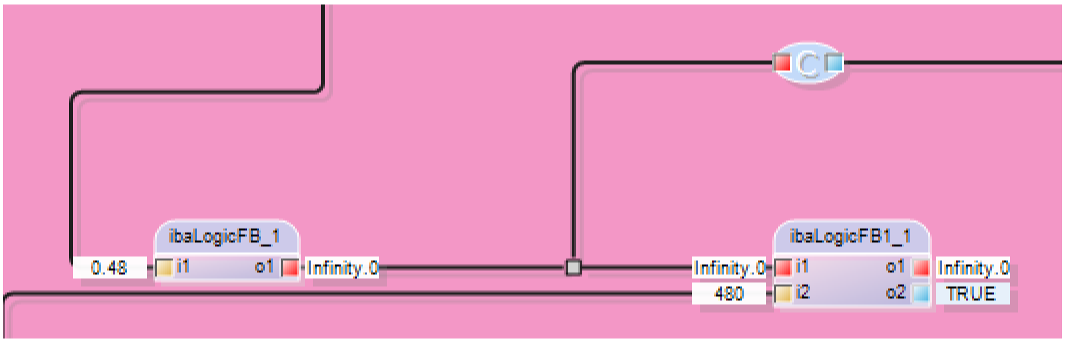
The value field of the output shows NaN (= Not a Number) as an additional error identification. For example, at 0.0/0.0

Infinity.0 can also be displayed as an error, e.g.: at 0.48/0.0;

Example:

A division by zero in the function block ibaLogicFB_1. This will spread. The ibaLogicFB1_1.o1 output is also invalid because the defective i1 is used for its evaluation. Output o2, however, does only depend on i2 and is not invalid.





Tip

To get an output VALID again, the error has to be eliminated or the error has to be intercepted using the VarValidate standard function block.

It has to be taken in consideration that the arrays and structures do have a global valid bit only and are not based on the single elements.

13.2 ibaPDA Express

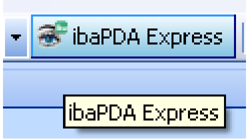
The ibaPDA Express is used for checking a signal waveform quickly.

Requirement

The function is available only when the program has been switched online.

Procedure

➲ Start ibaPDA Express by clicking with the mouse on the <ibaPDA Express> button in the toolbar.



Result

ibaPDA Express is opened with its own window within the ibaLogic application.

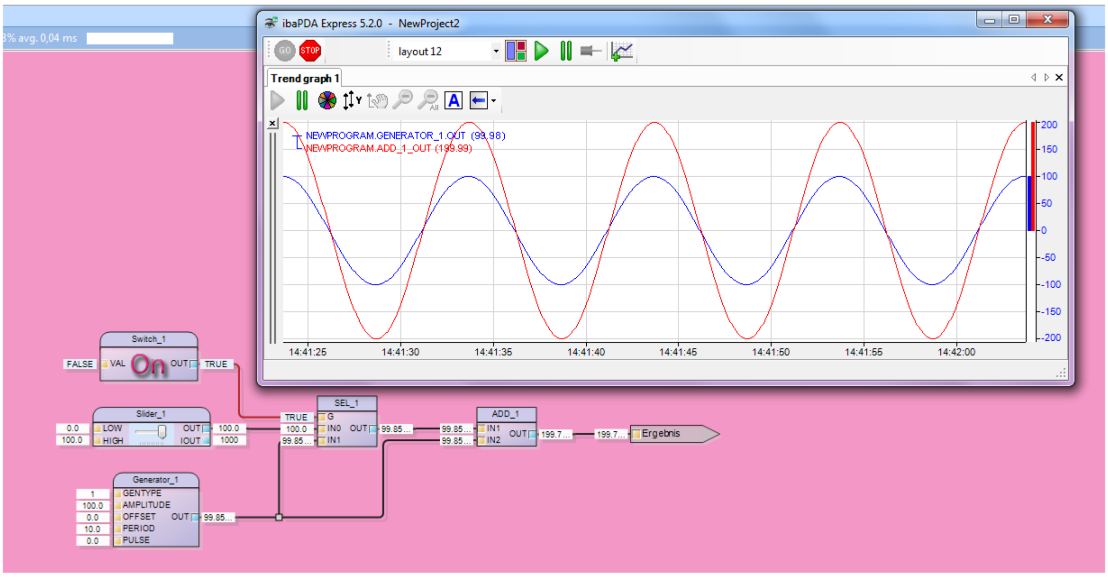


Figure 128: ibaPDA Express with several signals

The ibaPDA Express window can be fixed (using the pin button) that it always remains in the foreground.

13.2.1 Controlling the Signal Display

The following toolbar is available for controlling the signal display.

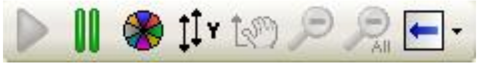


Figure 129: ibaPDA Express: Toolbar

The following table contains the explanation of the icons.

Icon	Name	Key operation	Explanation
▶	Start scrolling	<F6> (Switch)	Starts continuous display with the current time point. Active, when "Pause scroll" is pressed.
‖	Pause scroll	<F6> (Switch)	Stops the continuous display. After pressing this, a ruler appears in the graph that can be moved with the mouse and used to measure the curves. The signal values are displayed in the legend. You can move the X-axis using the mouse. In this manner, you can browse values from the past. Active, when the display is on.
●	Assign signal colors automatically		All curves of this display are colored in accordance with the default scheme for each graph.
↕Y	Auto scale all	<F5>	All curves of this display are scaled automatically for each graph and the Y-axis.
✋	Restore manual scaling		Manual settings for scaling, where defined, are restored after auto scaling or zooming. Active, if manual scaling has been defined.
🔍	Zoom out by one step	<F3>	Active only when the display has been zoomed. Return to the previous zoom factor (reduce).

Icon	Name	Key operation	Explanation
All	Zoom out all	<F4>	Active only when the display has been zoomed. Return to the initial (automatic) display.
← ▾	Scroll direction		You can change the scroll direction by selection in the pull-down menu.

13.2.2 Select Signals

You can drag signals by keeping the <Alt> button pressed from a connector and drop them into the ibaPDA Express window.

Optionally, you can:

❑ Display a signal in a separate signal strip.
To do this, drag the signal on the X-axis and a new strip is created.

❑ Place a signal in an existing strip.
To do this, drag the signal to the strip, and another Y-axis is created.

❑ Place a signal on an existing Y-axis (same scaling with one other signal).
To do this, drag the signal to this Y-axis.

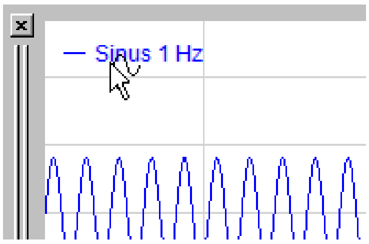
The new signals in one window are automatically assigned a new color. Those signal names having the same color are arranged in the top left section of the strip. Signals having a common axis are joined with a dash.

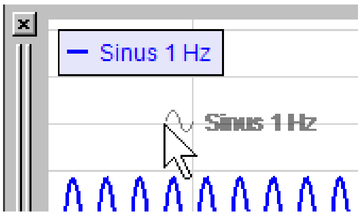
13.2.3 Move signal

Signals can be moved between graphs and also beyond the limits of the window. This means that a signal can be dragged from one graph to another graph that already has a signal. You can differentiate between the signals with the help of the automatic color assignment.

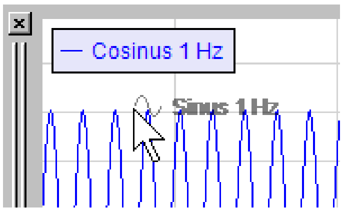
Procedure

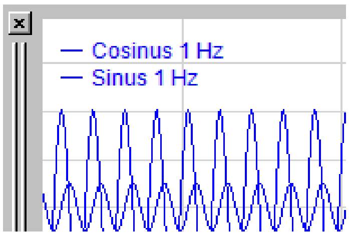
1. Move the mouse pointer to the name (Legend) of the signal that needs to be moved. The mouse pointer indicates with a wavy line that it has acquired the signal.





2. Drag the signal, keeping the mouse button pressed, to the other graph in order to drop it there in a free area.





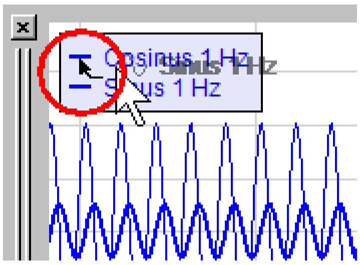
Result

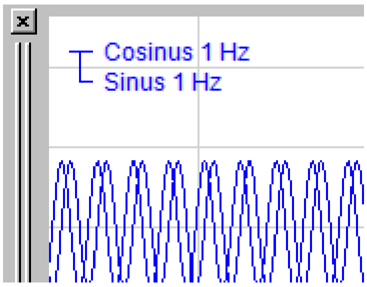
You have created two signals with separate Y-axis.

Remark

Do not leave the signal in step2, but drag it to the existing signal until a small black arrow appears. In this manner, the same Y-axis is assigned to the signal.

In case of binary signals, you also determine the sequence of the signals. Binary signals are displayed below one another. Depending on whether the small black arrow docks above or – as illustrated below – the signal, the binary signal is displayed above or below it.





Result

You have created two signals with a common Y-axis.

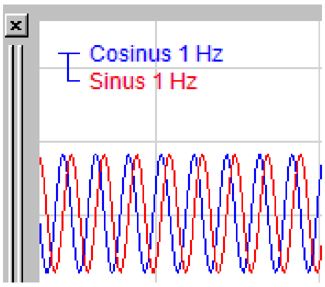
13.2.4 Mark the signals with color

You can mark the signals with colors in different ways:

- ❑ Automatically
- ❑ Manual setting

Procedure

➲ Press the <Assign signal colors automatically> button to assign colors to the signals automatically.



13.2.5 Remove Signal from the Display

Procedure

1. Place the mouse pointer in the graph on the name (Legend) of the signal that needs to be removed.
2. Click the right mouse button. The context menu is displayed.
3. Select "Remove signal".

Note

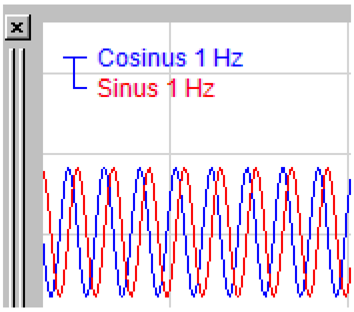
By removing the Y-axis, all signals are removed that are assigned to this axis.

13.2.6 Remove Graphs from the Display

There are different options to remove a graph.

Procedure

1. Click on the small cross at the top left above the top of the bar.



or

2. Click the right mouse button in a free area within the graph.
   The context menu is displayed.
3. Select "Remove graph".

13.2.7 Scale Axes

13.2.7.1 Auto scaling

In order to display a signal over its entire amplitude range in one graph, it is recommended that you use the auto scaling feature. All signals or all Y-axes of the graph are scaled accordingly with respect to the largest amplitude.

Procedure

1. Press the right mouse button in the appropriate graph. The context menu is displayed.
2. Select "Auto scale".
3. If you would like to auto scale all graphs in one signal display, press the <F5> key or select the <Auto scale all> button.

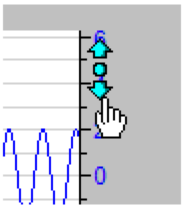
13.2.7.2 Scaling with the mouse

You can change the scale of the signals in the Y-direction at the upper ends of the Y-axis scale using the mouse.

Procedure

1. Bring the mouse pointer close to the end of the scale until blue arrows appear.
2. Keep the mouse button pressed on the arrow pointing upwards: The scale gets expanded.

3. Keep the mouse button pressed on the arrow pointing downwards: The scale gets reduced.
4. Keep the mouse button pressed on the dot between the arrows: Auto scaling is carried out.



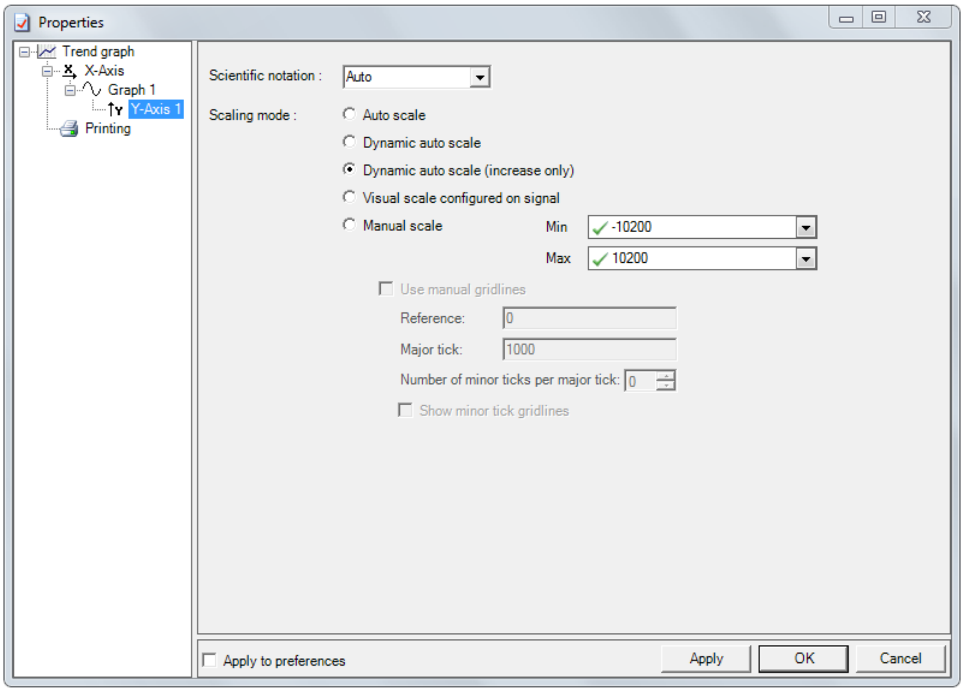
Tip

If you are using a mouse with a scroll wheel, you only need to position the mouse pointer on the scale. You can change the scale using the scroll wheel. This functionality is also available on the X-axis.

13.2.7.3 Scaling using the display settings

Procedure

1. Click the right mouse button in the area within the desired graph.
2. Select "Properties". The "Properties" dialog box is displayed. You can specify manual scaling using the "Y-axis" option. If a graph has multiple Y-axes, there is a separate tab in the dialog box for each Y-axis.
3. Accept the settings with <Apply>.



Scientific notation : Auto

Scaling mode :
○ Auto scale
○ Dynamic auto scale
● Dynamic auto scale (increase only)
○ Visual scale configured on signal
○ Manual scale Min ✓ -10200 Max ✓ 10200

□ Use manual gridlines

Reference: 0

Major tick: 1000

Number of minor ticks per major tick: 0

□ Show minor tick gridlines

□ Apply to preferences

Result

All signals that are assigned to the corresponding Y-axis are scaled with the same setting.

Remarks

By selecting the "Apply to preferences" option, you select the settings configured as the default settings.

13.2.8 Move Scales

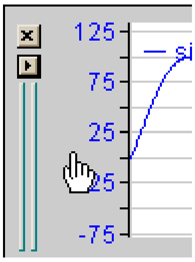
You can move the X-axis as desired in the pause mode of the display.

Requirement

Y-axis: Auto scaling is not selected.

Procedure

1. Position the mouse pointer on the Y-axis until the hand icon appears.
2. Press the left mouse button in order to move the scale upwards / downwards or to the left / right.



13.2.9 Zoom Function

The zoom function affects both the X and Y directions.

If you zoom in a graph, all other graphs located in the same display also get zoomed. A signal display can always maintain only one time base for all the graphs that it contains.

When the display is active and running, zooming expands the time base and hence enlarges the display. The signals run through faster, since the same geometric length of the X-axis is converted to fewer units of time.

Zooming in general

1. Press the <Shift> button and zoom simultaneously with the mouse. Only the X-axis is zoomed.
2. You can restore the original and un-zoomed display using the 🔍All button or the <F4> key.

13.2.9.1 Zooming in (Enlarge)

You can zoom in all over in one strip. In the zoomed in state, you can change the scale in the Y-direction without affecting the zoomed section of the X-axis.

Auto scaling in the Y-direction pertains to the values in the zoomed (= visible) area.

Requirement

You have zoomed out.

Procedure

1. Draw a rectangle using the left mouse button so that the area selected is enclosed.
2. Release the mouse button.

13.2.9.2 Zoom out (Reduce)

Requirement

You have zoomed in.

Procedure

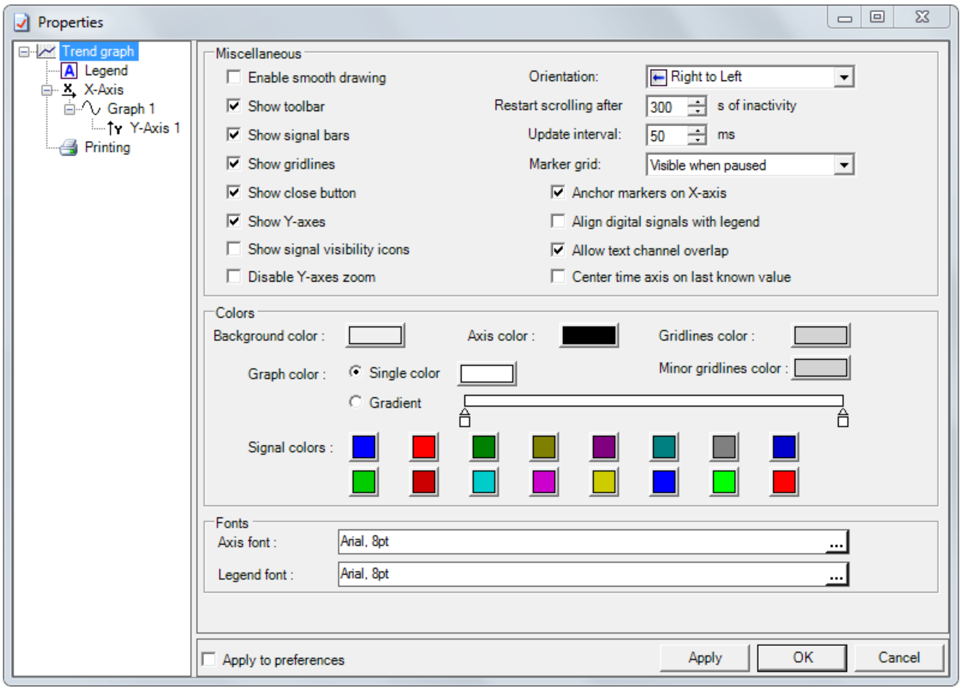
1. Press the <Zoom out one step> button or the <F3> key to achieve reduction step by step.
   Thus, with each action, all previous zoom steps are reversed one after another.
2. You can restore the original and un-zoomed display using the <Zoom out all steps>button or the <F4>key.

13.2.10 Trend graph Properties

You can configure general settings for the display of graphs in the Trend graph Properties dialog box.

Procedure

1. Click with the right mouse button on the signal strip.
2. Click with the left mouse button on "Properties...".
3. Select "Trend graph" in the structure on the left.



Miscellaneous

- □ Enable smooth drawing
- ☑ Show toolbar
- ☑ Show signal bars
- ☑ Show gridlines
- ☑ Show close button
- ☑ Show Y-axes
- □ Show signal visibility icons
- □ Disable Y-axes zoom

Orientation: Right to Left
Restart scrolling after 300 s of inactivity
Update interval: 50 ms
Marker grid: Visible when paused

- ☑ Anchor markers on X-axis
- □ Align digital signals with legend
- ☑ Allow text channel overlap
- □ Center time axis on last known value

Colors

Background color :
Axis color :
Gridlines color :
Graph color : ● Single color ○ Gradient
Minor gridlines color :
Signal colors :

Fonts

Axis font : Arial, 8pt
Legend font : Arial, 8pt

□ Apply to preferences

13.2.10.1 Miscellaneous

Option	Explanation
Enable smooth drawing	This option smoothens the graph lines as they are displayed.
Show toolbar	This option displays the toolbar.
Show signal bars	This option causes that the bar associated with the graph is displayed.
Show gridlines	This option causes that the trend graph is indicated with gridlines
Show close button	This option causes that the close button of a signal strip (not of the entire trend graph) is displayed.
Show Y-axes	This option causes that the Y-axis including the value scale is displayed.
Show signal visibility icons	This option displays symbols next to the signal names in the legend which can be used to hide and show the signals.
Disable Y-axes zoom	This option causes the closing of all Y-zoom actions
Orientation	The scroll direction is set.
Restart scrolling after	Configuration of a certain time in seconds which restarts scrolling after inactivity for this period of time.
Update interval	Setting for the time intervals at which the display should be refreshed.
Legend style	Choose view of the legend transparent/opak/invisible
– Never visible:	no display
– Visible when paused:	When stopping the trend graph drawing, the value table will open up automatically
– Always visible:	The value table is always visible, whether in started or stopped state
– Manual:	The value table is shown/hidden with a button in the headline
Anchor markers on X-axis	This option causes that the two markers that are displayed when the trend graph is stopped are being laid to the edge when zooming into an area outside the markers. If this option is not chosen, the markers will stay in their position autonomous of the zoom area.

Option	Explanation
Align digital signals with legend	This option causes that the digital signals are not displayed on top of each other at the bottom of the trend graph, but directly at the corresponding legend.
Allow text channel overlap	without function
Center time axis on last known value	After the restart, the time/x-axis is set according to the last value

13.2.10.2 Colors

You can use this dialog screen to change the color scheme for the trend graph display and the pen colors for the curves.

➲ Click on the respective color button to change the color. Select the desired color from a color palette.

❑ Background, axes, gridlines, ledger lines for the gridline:

➲ Click on the respective color button to change the color. Select the desired color from the color palette.

❑ Graph:
Background color in the signal strip uniform or with color gradient.

➲ Double-click on the small box at the end of the color bar and select the color from the color palette.
If required, double-click on the color chart to add more color tags and color them; in addition, they can also be moved. In order to delete a color tab, mark it with a mouse click (black arrow tip) and press the <DEL> key.

❑ Signals: you can use these pen colors to define 16 curve colors that are available for the trend curve-display. The program automatically assigns colors to the trend curves based on these 16 colors. The pen colors are also provided in the signal definition table in the signal grid in the sequence shown here (line wise from the top to the bottom).

13.2.10.3 Fonts

The fonts are defined for the lettering of the axes and the legends (Signal names). You can open the dialog box to change the font using the <...>browser button at the end of the line.

13.2.10.4 Signals

If you call up this dialog box in a graph or for an existing trend graph, in which signals are being displayed, the signals with their current setting are listed, including the colors.

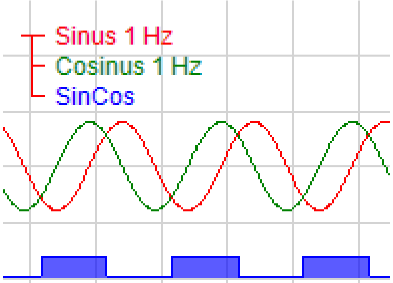
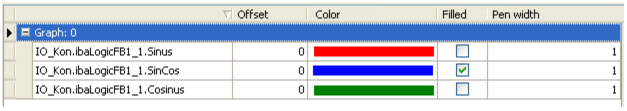


Figure 130: Trend graph



	Offset	Color	Filled	Pen width
Graph: 0				
IO_Kon.ibaLogicFB1_1.Sinus	0	🟥	☐	1
IO_Kon.ibaLogicFB1_1.SinCos	0	🟦	☑	1
IO_Kon.ibaLogicFB1_1.Cosinus	0	🟩	☐	1

Figure 131: Graphical signal settings

You can choose the color for each signal from a selection list in the cells of the "Color" column of the table.

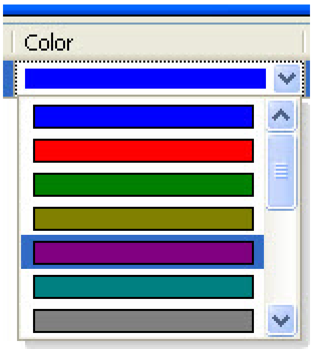
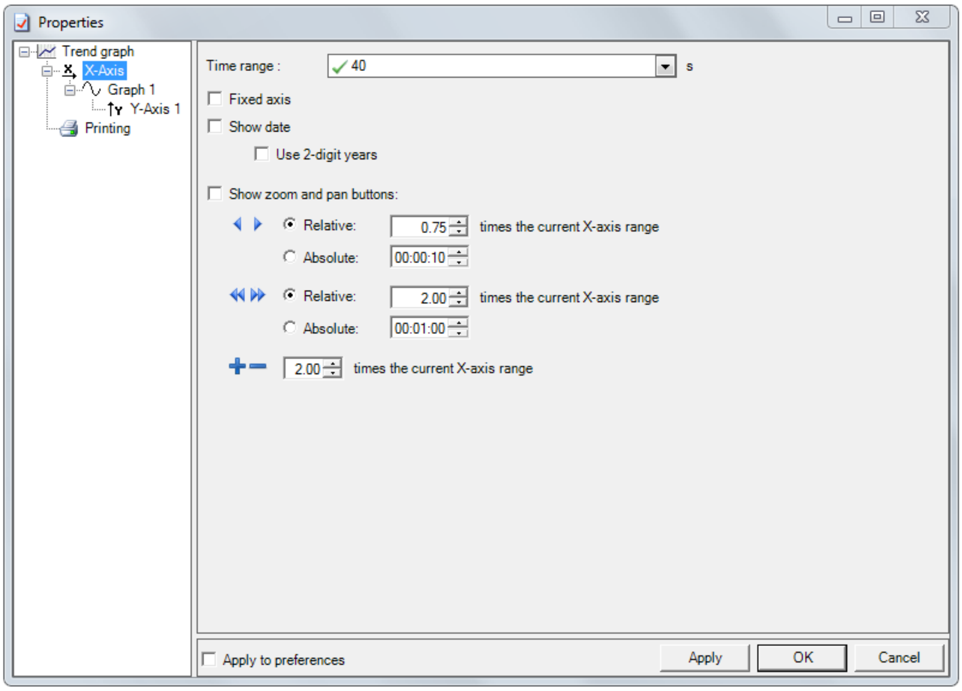


Figure 132: Color selection list

13.2.10.5 X-axis

Time range

You can specify a fixed time range in seconds instead of automatic scaling, and this is shown in the display. In this manner, you control the speed and the expansion of the signal in the X-direction in the display.



Properties

- Trend graph
  - X-Axis
    - Graph 1
      - Y-Axis 1
  - Printing

Time range : ✓ 40 s

□ Fixed axis

□ Show date

□ Use 2-digit years

□ Show zoom and pan buttons:

● Relative: 0.75 times the current X-axis range

○ Absolute: 00:00:10

● Relative: 2.00 times the current X-axis range

○ Absolute: 00:01:00

2.00 times the current X-axis range

□ Apply to preferences

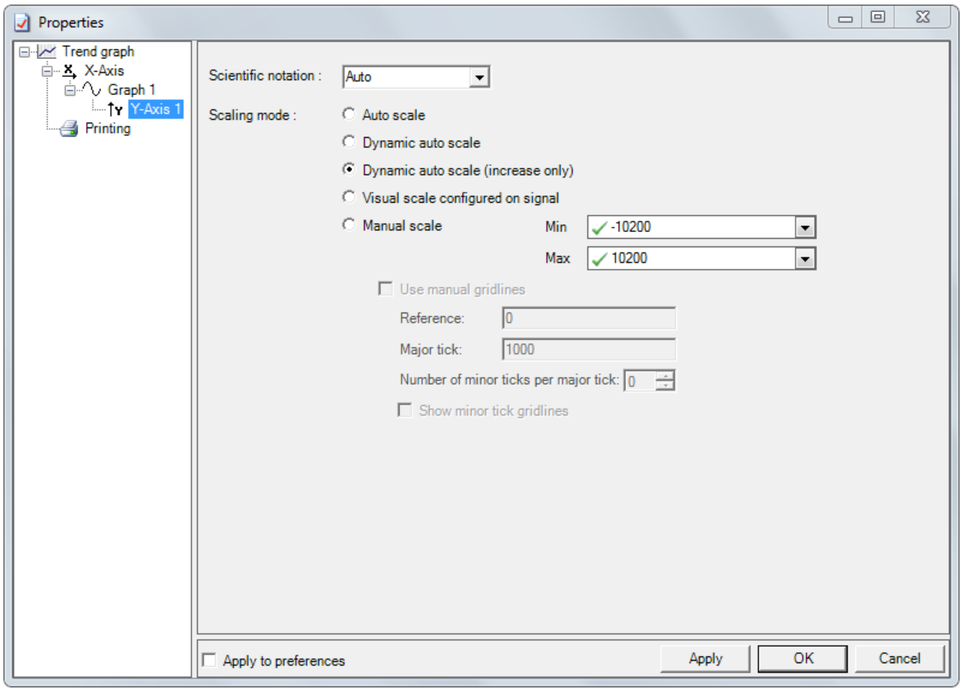
Figure 133: X-axis: Properties

Fixed axis

Normally, the time axis moves with the signal so that new values sampled are always shown at the border of the graph in the display. Using the "Fixed axis" option, the time axis from the current time point for the period (time range) configured is fixed and the sampled values are written into the empty graph. If the graph is filled, the next (empty) time range is displayed and sampled values continue to be written.

13.2.10.6 Y-axis

If you have created more than one Y-axis in a graph, the settings dialog screen has multiple "Y-axis #" tabs. Thus, you can configure settings for all Y-axes separately.



Scientific notation : Auto

Scaling mode :
○ Auto scale
○ Dynamic auto scale
● Dynamic auto scale (increase only)
○ Visual scale configured on signal
○ Manual scale Min ✓ -10200 Max ✓ 10200

□ Use manual gridlines
Reference: 0
Major tick: 1000
Number of minor ticks per major tick: 0
□ Show minor tick gridlines

□ Apply to preferences

Figure 134: ibaPDA Express: Display settings

13.2.10.7 Scientific notation

- "Auto"
- "Always"
- "Never"

Option	Explanation
Auto	Depending on the size of the scale values (number of places before or after the decimal point), the scales are labeled in scientific notation (power of 10) or not.
Always	Scale values as power of 10
Never	Scale values always with digits before and after the decimal point

13.2.10.8 Scaling mode

- "Auto scale"
- "Dynamic auto scale"
- "Dynamic auto scale" (increase only)
- "Manual scale"

Option	Explanation
Auto scale	Default setting; when displaying one or more graphs, the Y-axis of the strip is scaled once in accordance with the lowest and highest of all values occurring (when involving a signal).
Dynamic auto scale	When you enable this option, the scaling is continuously adjusted with the highest signal amplitudes. If the amplitudes go beyond the signal strip again, the scaling is further reduced.
Dynamic auto scale (increase only)	When you enable this option, the scaling is continuously adjusted with the highest signal amplitudes. If the amplitudes go beyond the signal strip again, the scaling remains unchanged.
Manual scale	You can specify the starting (Min.) and end (Max.) value of the scale manually when you select this option. (Visible only when the dialog box is opened from the context menu in the signal strip, and not with the presets.)

13.2.11 Display of arrays in ibaPDA Express

Arrays can also be displayed in ibaPDA Express. If you drag and drop a one-dimensional array into the *chart* display in ibaPDA Express, the indices of the array are displayed on the x-axis and the y-axis displays the value.

Open a chart for the array display by clicking on the chart symbol.

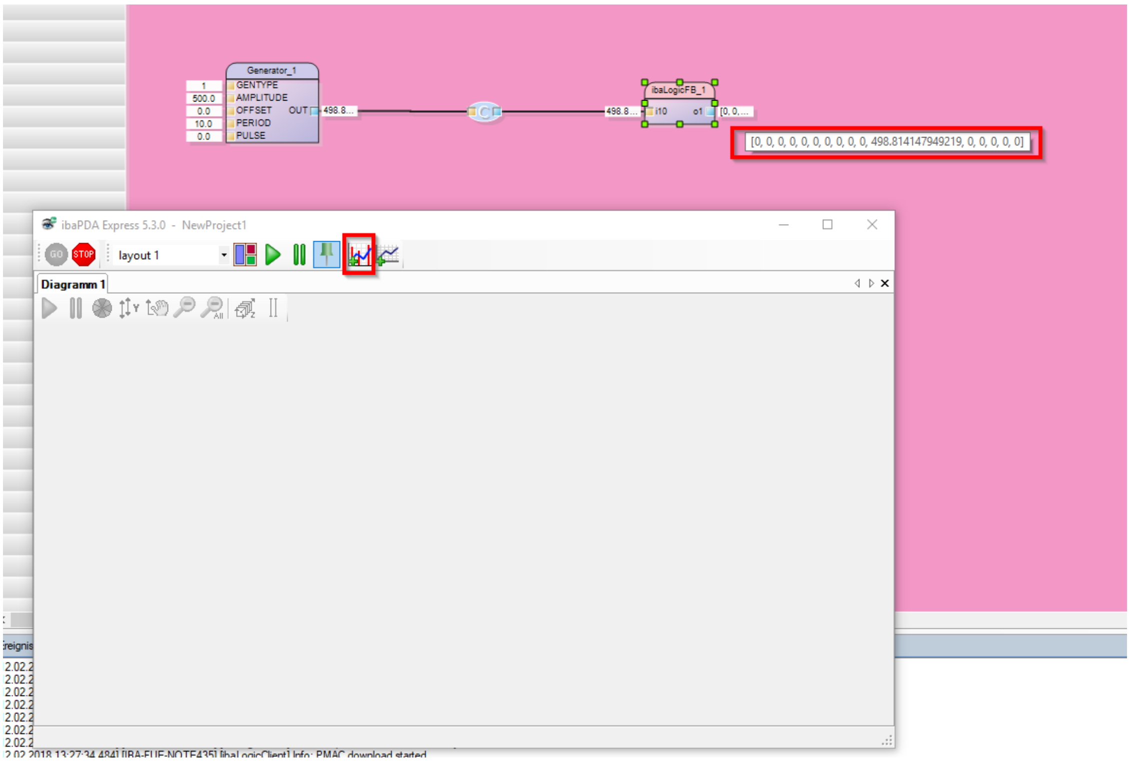
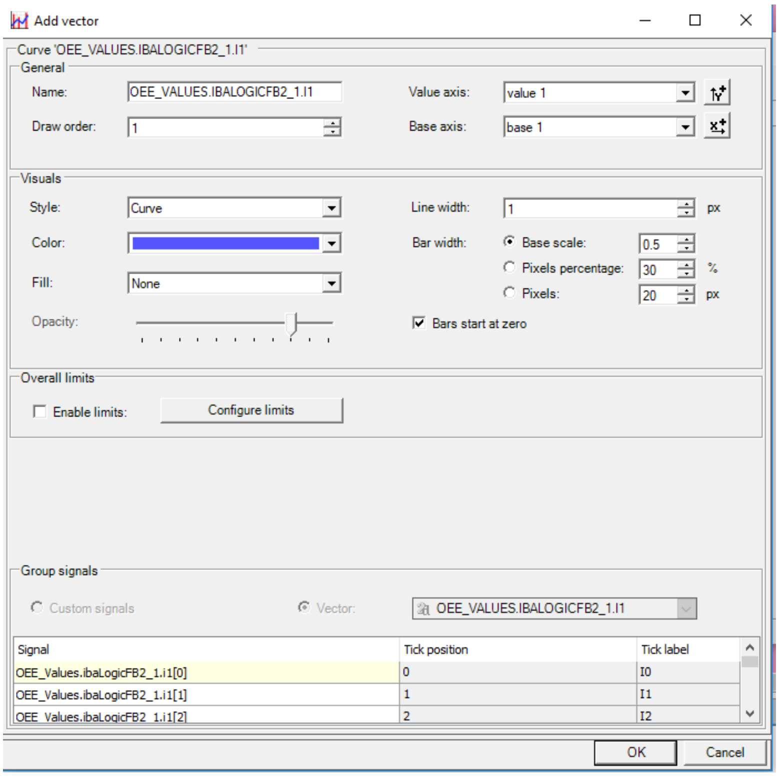

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 498.814147949219, 0, 0, 0, 0, 0]

Figure 135: Open a chart in ibaPDA Express

Drag the value to the chart by holding down the <Alt> key.



Add vector

Curve 'OEE_VALUES.IBALOGICFB2_1.I1'

General

Name: OEE_VALUES.IBALOGICFB2_1.I1

Value axis: value 1

Draw order: 1

Base axis: base 1

Visuals

Style: Curve

Line width: 1 px

Color:

Bar width: ● Base scale: 0.5

Fill: None

○ Pixels percentage: 30 %

Opacity:

○ Pixels: 20 px

☑ Bars start at zero

Overall limits

☐ Enable limits: Configure limits

Group signals

○ Custom signals ● Vector: OEE_VALUES.IBALOGICFB2_1.I1

Signal	Tick position	Tick label
OEE_Values.ibaLogicFB2_1.i1[0]	0	I0
OEE_Values.ibaLogicFB2_1.i1[1]	1	I1
OEE_Values.ibaLogicFB2_1.i1[2]	2	I2

OK Cancel

Figure 136: Properties dialog

Confirm with <OK>.

Note

The settings in the dialog correspond to the settings of the *chart* in ibaQPanel. You can find the description in the ibaQPanel manual.

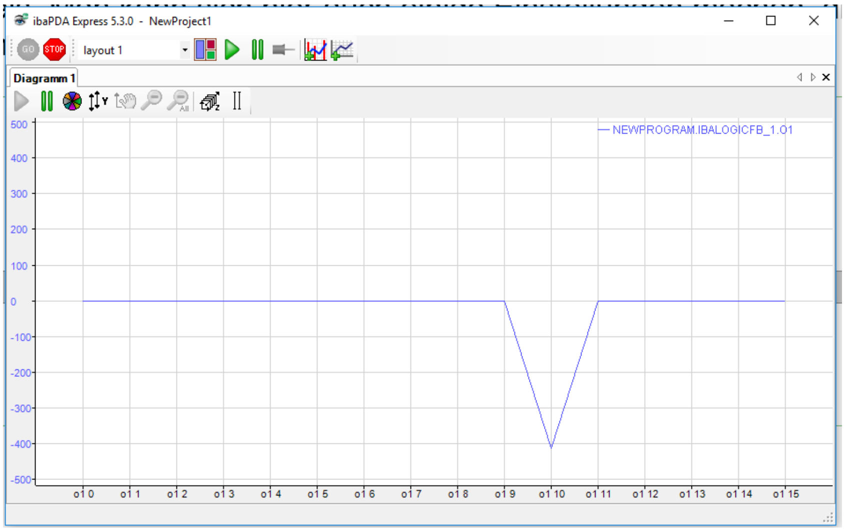


Figure 137: Display of an array in the chart view

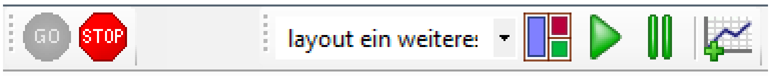
The array is displayed as graph, starting on the left side with the lowest index and on the right side with the highest array index (here: 15).

Zooming and other known practices are available. You can also see how the values change online, e.g. follow the array result of an FFT block live.

13.2.12 Extended functionality

You can enable the extended functionality using the icon in the title bar from the context menu.

Toolbar



It displays a toolbar having the following elements:

Symbol	Name	Explanation
GO STOP	Real time	Start / stop function of all strips.
▦	Manage layouts	Call of the layout management to create layouts, to arrange the layout tree, to copy layouts, etc.
▶	Start scrolling of signal displays	A previously stopped signal display can be reset to the online mode.
⏸	Stop scrolling of signal displays	Pause of a signal display to observe the curve, to measure values, etc.
📈	Adding a trend graph	Add another trend graph The trend graphs can be arranged as desired by holding the "Trend graph" tab with the left mouse button and moving it. The docking points become visible.

Further documentation

You are requested to refer to the appropriate add-on documentation of the ibaPDA system for the description.

13.3 Time response

Depending on the platform, ibaLogic provides a deterministic time response (real-time response).

Platforms:

- ❑ WindowsPC:
  non-deterministic, relatively stable cycle times for task times of ≥ 5 ms.
- ❑ PADU-S-IT-2x16:
  deterministic, very stable cycle time for task times of ≥ 1 ms.

The tasks of ibaLogic have a base time slot of minimum 1 ms, and this is based on the interrupt time base, which can be configured under the "Tools - I/O Configurator" menu. This is the minimum task interval.

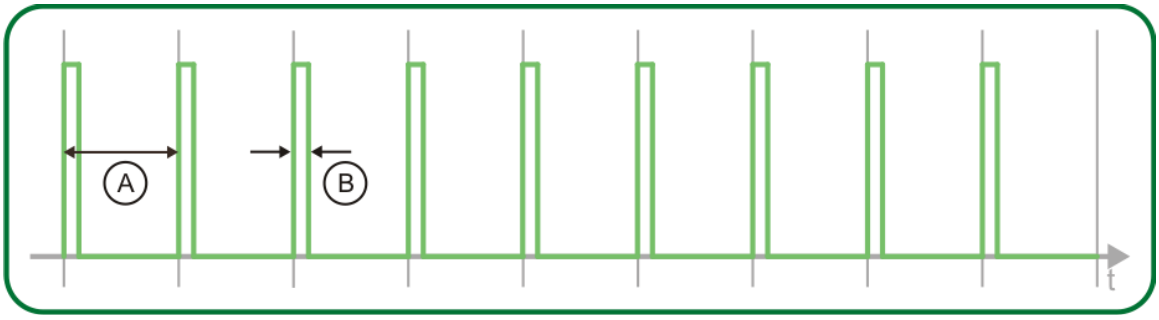
You cannot have faster tasks. It is possible that certain iba modules sample data at 50 µs and forward these as an array of values (Packets) to ibaLogic. ibaLogic then processes the values in the secondary clock cycle. For further information, please refer to"*Buffered Mode*, page 193".

For the purpose of task handling, ibaLogic checks the tasks pending for processing at the basic clock cycle configured, and enters these in an internal task list. This task list is evaluated cyclically from the top to the bottom and tasks evaluated are removed from the list.

It must also be noted that the basic clock cycle configured is also the clock cycle in which inputs can be read and outputs can be written.

ibaLogic knows only the interval task and the event task.

The interval task is started in accordance with the time interval configured. The program linked with it has its own evaluation time.



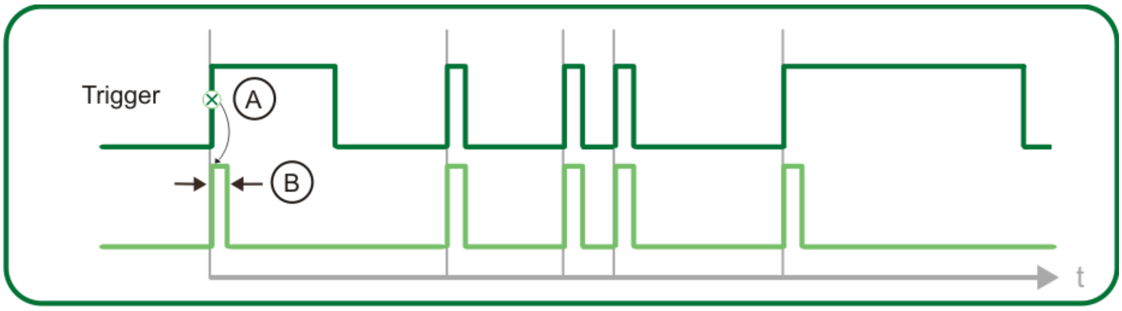
A Interval

B Evaluation time

Figure 138: Interval task

All interval tasks start at time 0. This means, that the first start of all interval tasks is at the start of ibaLogic, the next one according to the interval set.

The event task is triggered once by a trigger signal (rising edge)



A Trigger time

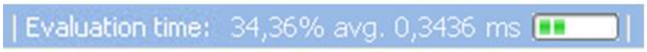
B Evaluation time

Figure 139: Event task

13.3.1 Evaluation time

The program evaluation times of various programs in the entire user project is also important for the consideration of the time response.

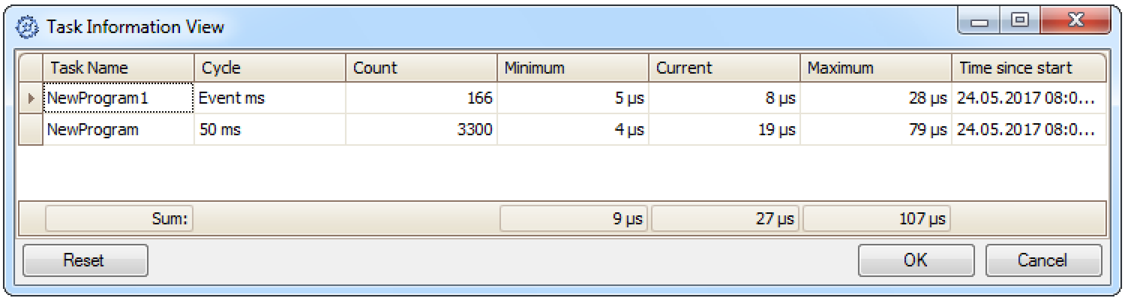
ibaLogic provides the evaluation time for various interval tasks to the user to check the system loading as a number and as a bar.



| Evaluation time: 34,36% avg. 0,3436 ms |

In this example, considering a 1 ms interval task, the percentage value means that 34.36 % of 1 ms is required. This means that this value is a percentage of the time slot configured for the task. 34.36 % for a task time slot of 1 ms works out to be 0.3436 ms of program evaluation time for the program.

There is also an overview of all tasks.



Task Information View

Task Name	Cycle	Count	Minimum	Current	Maximum	Time since start
NewProgram1	Event ms	166	5 µs	8 µs	28 µs	24.05.2017 08:0...
NewProgram	50 ms	3300	4 µs	19 µs	79 µs	24.05.2017 08:0...
Sum:			9 µs	27 µs	107 µs	

Figure 140: Overview evaluation time

The button <Reset> can be used to reset the minimum and maximum values.

Task name	
Cycle:	Time interval for interval tasks, notice or remark to event at event tasks
Count:	Amount of the former calls since start
Minimum:	Minimum evaluation time since start
Current:	Actual evaluation time
Maximum:	Maximum evaluation time since start

Time since start:	Passed time since start
Sum:	Sum of the columns = approximate overall duration of the evaluations

13.3.2 Turbo mode (internally fixed set)

In order to prevent ibaLogic from getting temporarily blocked by Windows, ibaLogic assigns a   processor core exclusively in multi-core systems.

Note

In order to ensure flow and performance as deterministic as possible, iba recommends:

- For task times < 20 ms:
  Use an iba interrupt source
  (ibaFOB card or similar)

In addition, ibaLogic differentiates between the "Measurement" mode and the "Soft PLC" mode. Settings see "*General settings*, page 181".

13.3.3 Priority of processing (inputs, evaluation, outputs)

The general order of the processing is basically the following:

❑ write new output data if present

❑ Read the input data

❑ Check if the processing of a new task needs to be triggered (event or interval). Thereby, the priorities will be checked and the tasks will be interrupted, too, or it will be waited for the end of tasks with higher priority.

The output data will not be written at the end of the evaluation time, but always at the next clock pulse because writing to the hardware requires time. If every task wrote output values at the end, this would cause a general slowdown of the system; therefore, this option has been chosen for ibaLogic.

13.3.4 Measurement (buffered values)

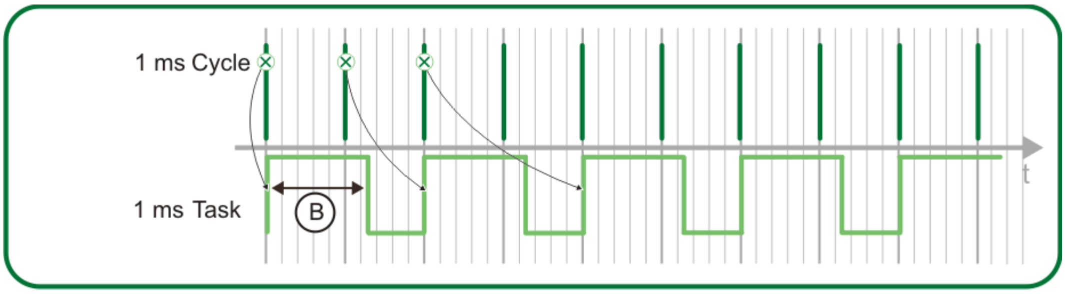
The operation type is chosen automatically when setting buffered values in the I/O configurator.

This operating mode ensures that ibaLogic does not lose any input sample. This is also true when individual tasks within ibaLogic need to be suspended. The runtime system of ibaLogic ensures that the data are made available equidistantly in the task interval configured. If tasks get suspended, the system makes up for the cycles. As a result, with task suspension for limited time, it may happen that ibaLogic, at times, evaluates only those values that belong to the "past".

Nonetheless, it is always ensured that, for example, values that are equidistant and correct are available for FFT analyses. Permanent suspension leads to buffer overflow. Such configuration is not acceptable.

You must make considerations regarding the modes of operation possible for reading in the hardware signal inputs.

In the "Measurement” mode, the hardware input signal status is buffered in accordance with the task interval configured. The program then works with the oldest buffered value at the next possible program start. This means that the “Soft PLC" mode and the "Measurement" mode work the same way when the program processing times < the interval time. If the processing times are greater, you have buffer overflow of the sample values in the "Measurement" mode.



B Evaluation time

Figure 141: Buffer overflow – Shifts

Example: the dark green 1 ms clock cycle saves the value that is processed when the task begins (light green). The black arrow indicates the sampled value with which the task works and how the buffer overflow condition develops. The evaluation time of a task, however, is more than 1 ms, and hence, there is a time shift.

13.3.5 Soft PLC

In this operating mode, which is suitable for open-loop and closed-loop control tasks, ibaLogic ensures that only the latest signal states are processed. In contrast to the "Measurement” operating mode, it does not matter here whether samples get lost or not. On the other hand, it is desired to obtain only the most up-to-date data from the last I/O transfer cycle.

Data is read in from the input resources with every new basic clock cycle.

In the "Soft PLC" mode, the current hardware input signal status is read in and processed at the start of the task. That means, in case of displacement, the most recent data would always be used.

13.3.6 Time observation for several interval tasks

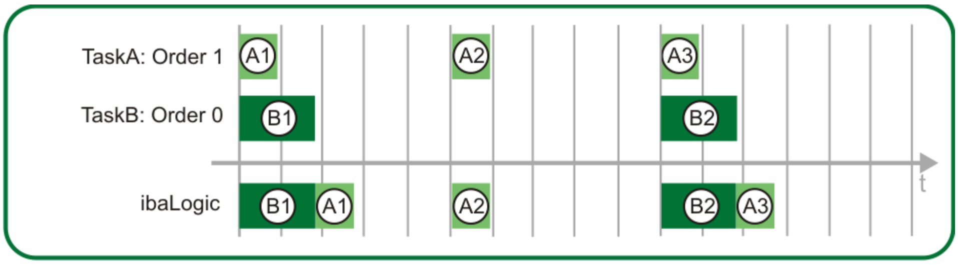


Figure 142: Evaluation without overflow – 2 tasks having different interval times, equal priority and a different order

The 2 rows above represent the individual tasks in the theoretical evaluation sequence if they were to run independently. The numbers are a counter for triggering the tasks (1st trigger, 2nd trigger ...).

An interval task A having an interval time of 5 ms is displayed in the uppermost row. The order is 1, i. e. TaskB with order 0 is evaluated first (second row). The width of the bar (impulse) is equivalent to the evaluation time of the task of the associated program. The background represents a clock cycle grid. The impulse always begins at the interval time set.

Due to the same priority, the tasks are executed serially in the practice. This forms the bottom row. At the starting time, ibaLogic sees the tasks that need to be evaluated, and evaluates them one after another in accordance with the priority entered. Starting with the evaluation time of TaskB, followed by TaskA …

To clarify the situation, the program evaluation times shown are taken to be very large. In reality, the evaluation times are primarily of the order of µs, so that, for example, 20 tasks can be evaluated in 5 ms without any problem (empirical value).

13.3.7 Worst-case considerations for interval tasks at equal priority with time shift and order

If you assume a longer evaluation time for task A and task B, suspension or time shift is generated. Suspension or time shift means that the task is no longer started at the expected time point, since another program is still being evaluated. The task with the lowest priority number will be started at the correct time.

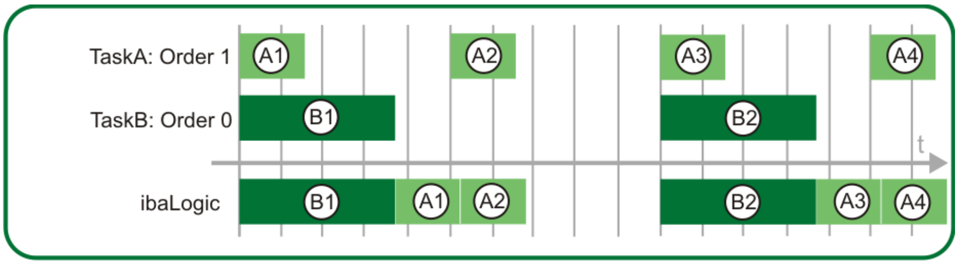


Figure 143: Task evaluation with time shift

The evaluation times have been selected in such a manner that both tasks together require more than 5 ms, and hence, task A cannot be started at the exact time interval foreseen. If a base cycle of 1 ms has been configured, a check is conducted at each cycle to see whether a task needs to be triggered. In the example here, task A (5 ms, priority 1) and task B (10 ms, priority 0). These will be started according to their priority.

Explanation on the case above

The jobs of the internal list are illustrated in the figure "Task evaluation with time shift - Excerpt".

At the outset, ibaLogic sees that task A (5 ms, priority 1) and task B (10 ms, piority 0) need to be executed and enters them in the internal job list according to their priority. Tasks that have been started are removed from the job list, new ones are added, and this is how the above figure emerges.

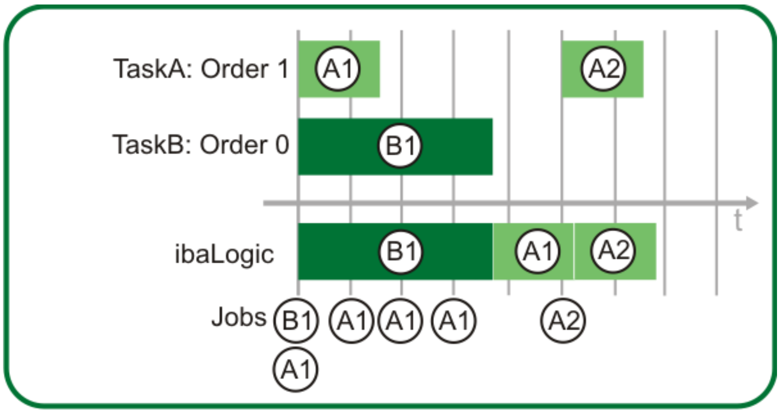


Figure 144: Task evaluation with time shift – Excerpt

Evaluation of interval tasks with time shift and priority

Let us assume that task A has been configured with an interval time of 2 ms (with the same program evaluation times), in which case, certain **cycles are lost**.

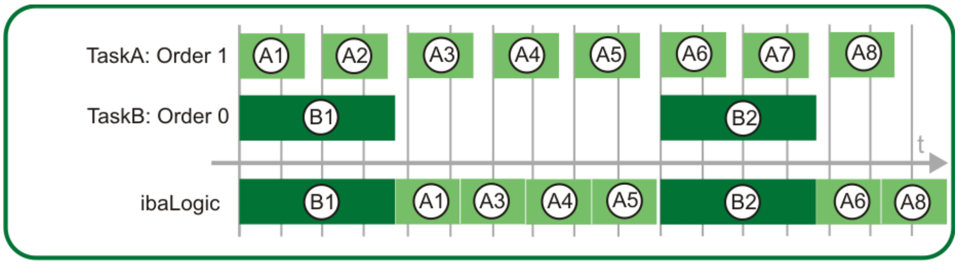


Figure 145: Task evaluation with time shift

A different picture emerges if the priorities are interchanged.

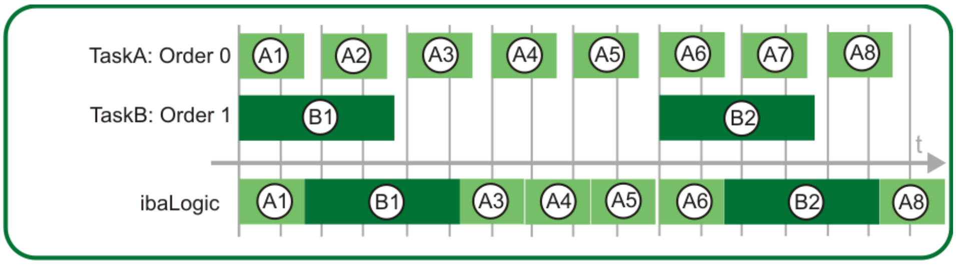


Figure 146: Task evaluation with time shift (reversed priority)

Another consideration is the "Soft PLC" mode and the "Measurement" mode.

In the "Soft PLC" mode, the hardware inputs are always read at the beginning of the task (x point in the following figure).

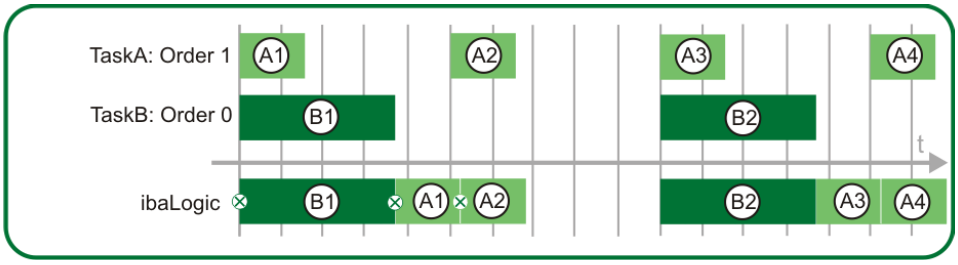


Figure 147: Hardware inputs in the "Soft PLC" mode

The situation in the "Measurement" mode is different.

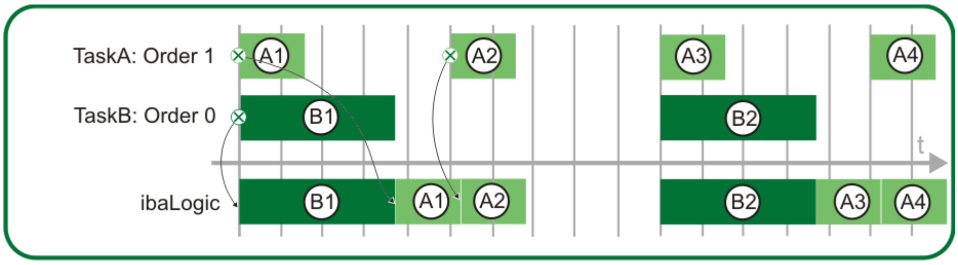


Figure 148: Hardware inputs in the "Measurement" mode

Here, the input signals are buffered in time, but are evaluated with a delay in case of a time shift.

Important Note

In general, there is a time shift or task suspension if the sum of the program evaluation times exceeds the smallest interval time used. There is buffer overflow if this time shift is permanent. The programs and the computer no longer work in line with the requirements. In case of temporary time shift or task suspension, it depends on the respective application whether this can be tolerated.

Data is written to the hardware outputs in the next base clock cycle after the evaluation time has ended. Hence, it may be useful to configure the base clock cycle to be faster than the task interval. If, for example, the evaluation time is 50 µs, the task interval time is 5 ms and the base clock cycle is 1 ms, data is written to the outputs after 1 ms.

13.3.8 Event tasks

Event tasks can be started by an input signal. Thereby, a low-high edge is needed for Boolean signals, a change of value is needed for integer input values.

13.3.9 Priority and interruptibility

Tasks with a higher priority will interrupt tasks with a lower priority. There is no limitation for the possible nesting depth.

With the interruptability, the user is also asked to have a clear concept behind his task, otherwise the agreement of interruption and evaluations can lead to undesired results.

Here, it should particularly be realized which input data and which evaluated data are processed.

An interrupted task evaluates with its own values only and cannot evaluate values of interrupting tasks.

Tasks can always have the values of already evaluated tasks as an output basis only.

13.4 Debugging

The following errors may occur:

- ❑ Program errors
- ❑ Compilation errors

13.4.1 Program errors

Frequently occurring errors in programs:

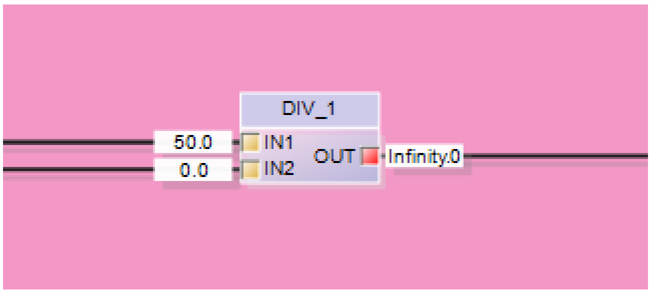
- ❑ Errors in user-defined function blocks
- ❑ Division by 0
- ❑ Incorrect signal trends
- ❑ Incorrect evaluation order or sequence

13.4.1.1 Errors in the user-defined function blocks

In order to trap logical errors, ibaLogic-V5 provides the option of generating an auxiliary output and to show it with ibaPDAExpress.

13.4.1.2 Division by 0/array limit

In case of errors like division by 0, Array transgression or similar errors, the output of the affected function unit turns to INVALID.
This will be displayed by a red mark on the affected function unit connector.



13.4.1.3 Incorrect signal trends

In order to be able to check evaluated values, ibaLogic-V5 provides the tool ibaPDAExpress. You can display the signal trends in real time with the help of this tool and thus, track whether your function unit is yielding the expected output values with various input parameters.

13.4.1.4 Evaluation sequence

If, in spite of error-free function blocks and macro blocks, the evaluation does not run as you expect it to, it is possible that the evaluation sequence is the problem.

In order to check the function units that are evaluated first, you can view the evaluation sequence of the corresponding program and thus, unearth any errors in the sequence.

For more information, please refer to "Evaluation Order".

If your program contains feedback paths, it is necessary to know the function unit that is in the first or last position in the evaluation sequence.

13.4.1.5 Compilation error

Although the syntax of the ST in the user-defined FBs is checked by the function block generator prior to compilation, it may happen under certain circumstances, that compilation of the generated IL code fails.

In such a case, you receive an error message in the event window.

If there is a message in your event window that appears as follows, please scroll up using the slider on the right border until you can see the first error message.

If compilation fails, always begin with the first error message that appears in the event window.

13.5 Performance limits

ibaLogic-V5 has been developed for the 32 bit variant of Windows and has certain limitations on account of its architecture:

❑ Maximum process size: 2 GB (i. e. memory that a runtime system can occupy)

The `Microsoft SQL Server 2008 Express` used by ibaLogic-V5 has the following system-bound performance limits:

❑ Maximum database size: 4 GB

❑ Maximum 50 instances on the same machine

❑ Support for only 1 CPU and 1 GB RAM

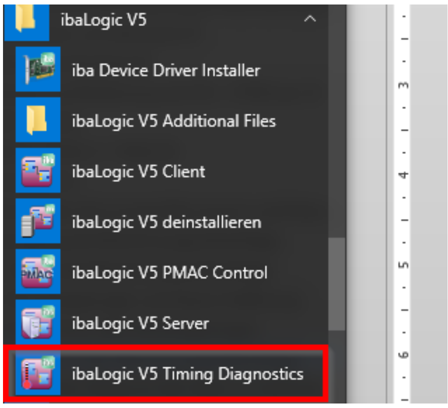
13.6 ibaLogic V5 Timing Diagnostics Tool

The ibaLogic-V5 Timing Diagnostics Tool enables a diagnosis of the tasks and their calculation behavior. This tool writes a dat file with all task state changes. With the help of these files, the online behavior of the entire project can be easily analyzed with ibaAnalyzer with regard to the states and task runtimes. The application is only available in English.

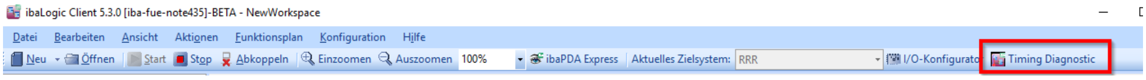
The diagnostic tool is subject to license and requires a dongle activation. Each ibaLogic PMAC in your system can be analyzed with a license that must be available in the dongle of the PC running the diagnostic tool.

There are several ways to call the tool:

❑ Select ibaLogic V5 Timing Diagnostics from the Windows start menu.

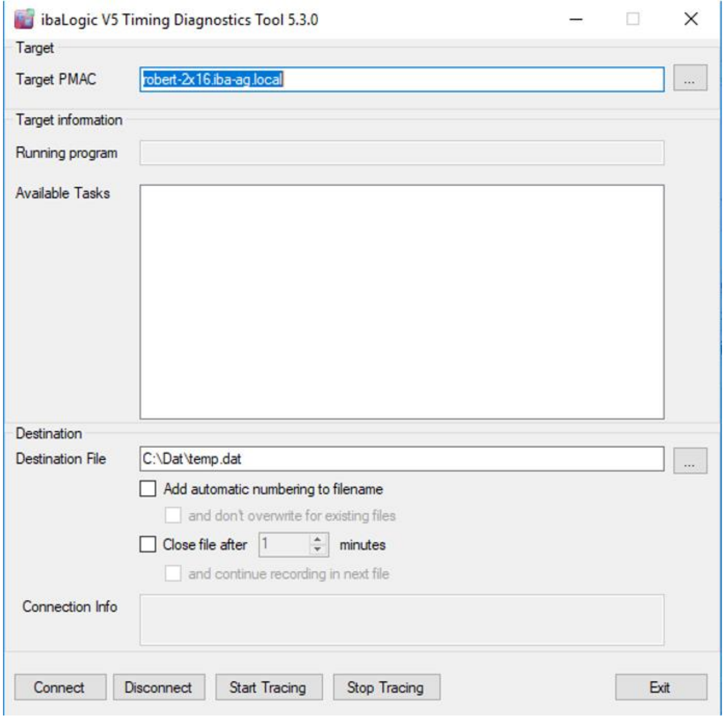

ibaLogic V5
iba Device Driver Installer
ibaLogic V5 Additional Files
ibaLogic V5 Client
ibaLogic V5 deinstallieren
ibaLogic V5 PMAC Control
ibaLogic V5 Server
ibaLogic V5 Timing Diagnostics

- ❑ Or click *Timing Diagnostics* on the toolbar in the ibaLogic client.


ibaLogic Client 5.3.0 [iba-fue-note435]-BETA - NewWorkspace
Datei Bearbeiten Ansicht Aktionen Funktionsplan Konfiguration Hilfe
Neu Öffnen Start Stop Abkoppeln Einzoomen Auszoomen 100% ibaPDA Express Aktuelles Zielsystem: RRR I/O-Konfigurato Timing Diagnostic

Procedure:

1. Start the ibaLogic V5 Timing Diagnostics tool.



Target PMAC: robert-2x16.iba-ag.local

Running program:

Destination File: C:\Dat\temp.dat

☐ Add automatic numbering to filename

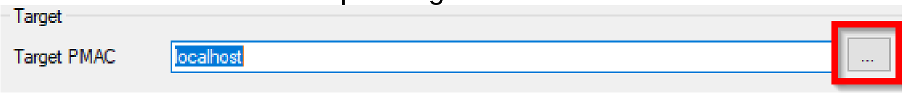
☐ and don't overwrite for existing files

☐ Close file after 1 minutes

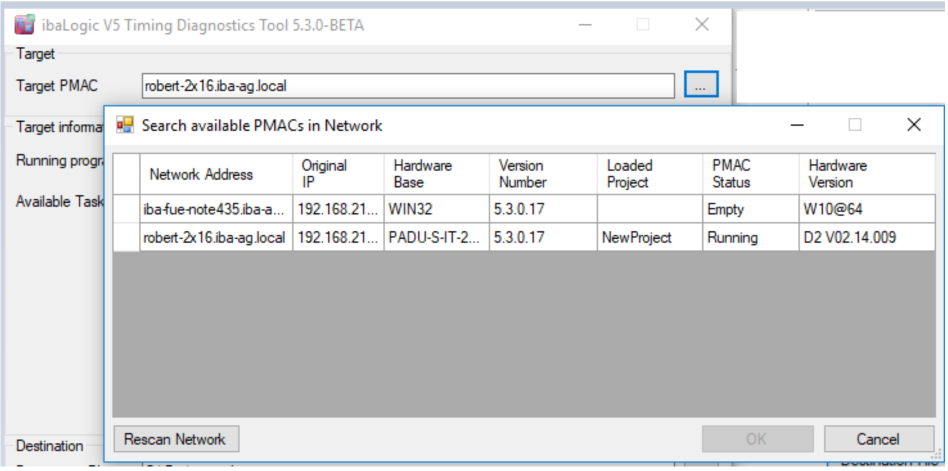
☐ and continue recording in next file

2. Connect to the corresponding PMAC.

3. Use the browser button to open a dialog to search for available PMACs in the network. Select the corresponding PMAC.



Target PMAC: localhost



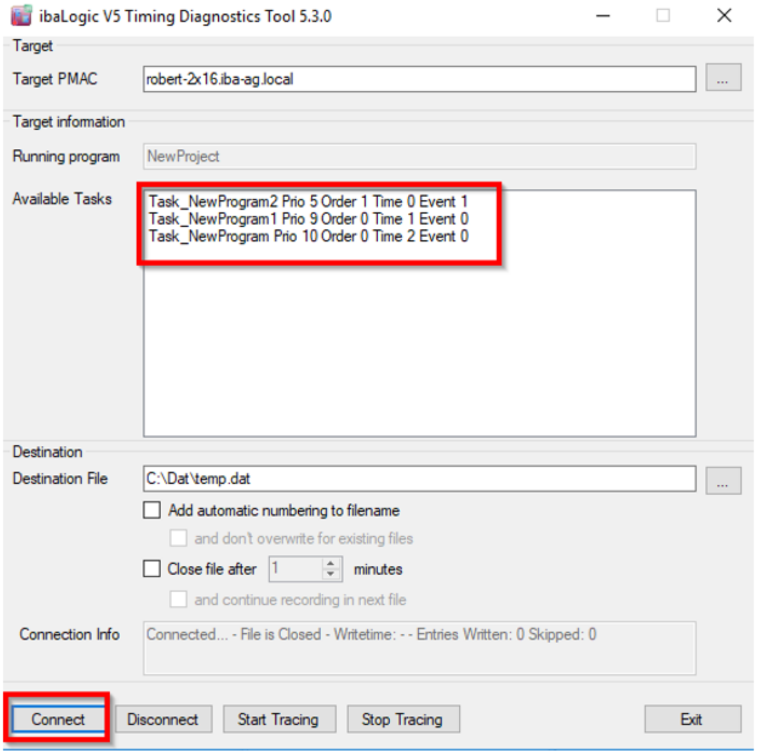
Target PMAC: robert-2x16.iba-ag.local

Network Address	Original IP	Hardware Base	Version Number	Loaded Project	PMAC Status	Hardware Version
iba-fue-note435.iba-a...	192.168.21...	WIN32	5.3.0.17		Empty	W10@64
robert-2x16.iba-ag.local	192.168.21...	PADU-S-IT-2...	5.3.0.17	NewProject	Running	D2 V02.14.009

4. Enter a file name in the field *Destination file* or select path and file via the browse button.

5. Further options can be used to specify the storage of the dat files e.g.
   - add automatic numbering
   - close file after x minutes

6. You can now connect to a running PMAC by <Connect>. The tasks found are displayed in the task overview.



ibaLogic V5 Timing Diagnostics Tool 5.3.0

Target

Target PMAC: robert-2x16.iba-ag.local

Target information

Running program: NewProject

Available Tasks:
Task_NewProgram2 Prio 5 Order 1 Time 0 Event 1
Task_NewProgram1 Prio 9 Order 0 Time 1 Event 0
Task_NewProgram Prio 10 Order 0 Time 2 Event 0

Destination

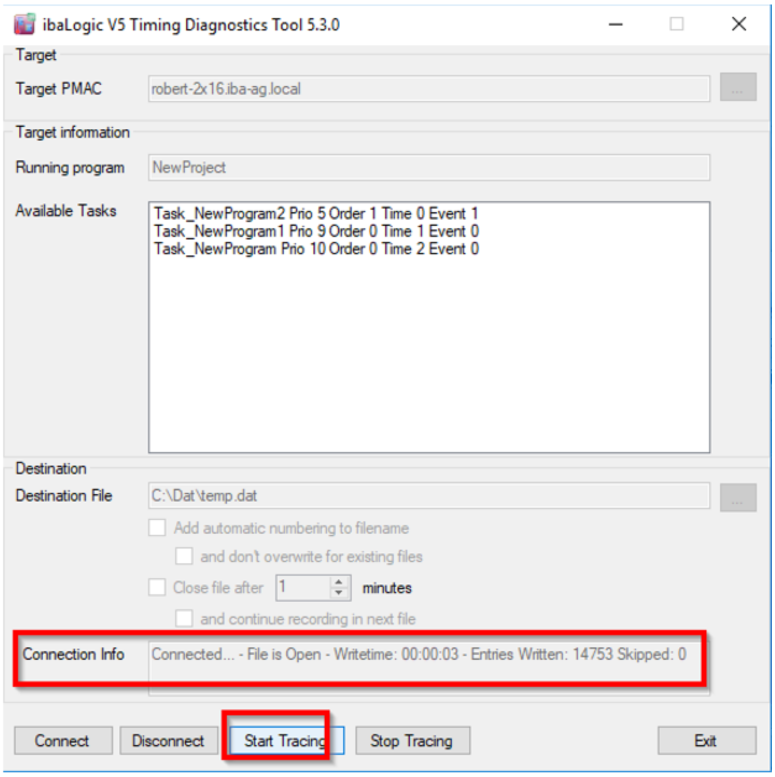
Destination File: C:\Dat\temp.dat

□ Add automatic numbering to filename
□ and don't overwrite for existing files
□ Close file after 1 minutes
□ and continue recording in next file

Connection Info: Connected... - File is Closed - Writetime: - - Entries Written: 0 Skipped: 0

7. <Disconnect> disconnects the connection.

8. A recording is started with <Start Tracing>.
   Note: If no PMAC is yet connected, <Start Tracing> automatically performes a connect command.



ibaLogic V5 Timing Diagnostics Tool 5.3.0

Target

Target PMAC: robert-2x16.iba-ag.local

Target information

Running program: NewProject

Available Tasks:
Task_NewProgram2 Prio 5 Order 1 Time 0 Event 1
Task_NewProgram1 Prio 9 Order 0 Time 1 Event 0
Task_NewProgram Prio 10 Order 0 Time 2 Event 0

Destination

Destination File: C:\Dat\temp.dat

□ Add automatic numbering to filename
□ and don't overwrite for existing files
□ Close file after 1 minutes
□ and continue recording in next file

Connection Info: Connected... - File is Open - Writetime: 00:00:03 - Entries Written: 14753 Skipped: 0

The *Connection Info* line shows if data is recorded. The data counter *Written* is constantly increasing. The parameter *Writetime* shows the current duration of the active dat file.

9. <Stop Tracing> stops the recording and the written dat file can be viewed with ibaAnalyzer.

13.6.1 Evaluation of the task diagnostics file in ibaAnalyzer

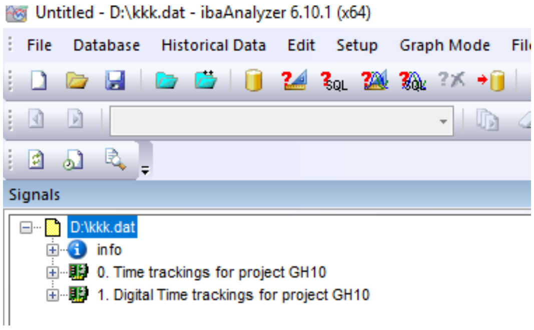
Each task can have three states:

State	Analog value	Digital value
pause	0	0
running	1	1
interrupted	-1	1

These values are recorded in 1 µs accuracy in the data file.

If all tasks have the same priority they cannot be interrupted. For this case, recording as a digital value is more suitable. This makes it easier to evaluate the runtimes by double-clicking.

When opening the data file in ibaAnalyzer you will get the following view with an analog and a digital node:



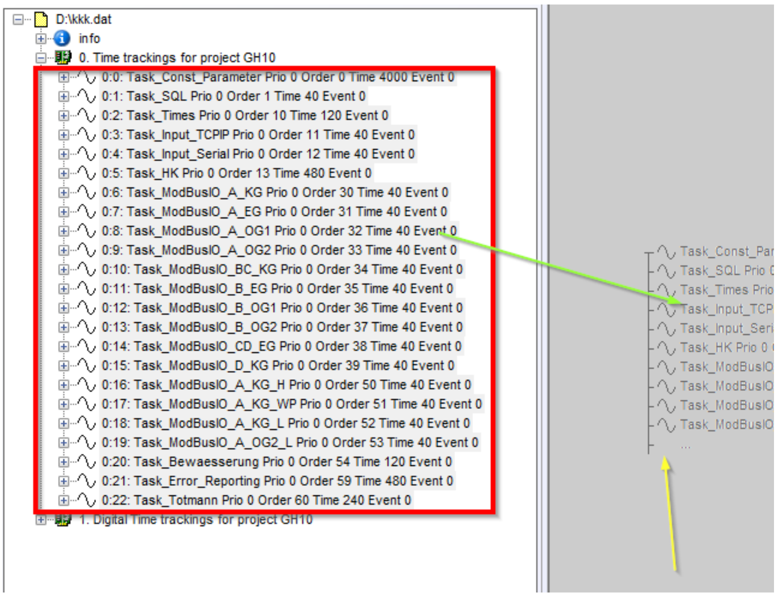
Untitled - D:\kkk.dat - ibaAnalyzer 6.10.1 (x64)
File Database Historical Data Edit Setup Graph Mode Fil
Signals
D:\kkk.dat
info
0. Time trackings for project GH10
1. Digital Time trackings for project GH10

Figure 149: Structure of the diagnostics file

Display the analog values

For analog values the following procedure is recommended:

Mark all signals and drag them into the view via drag & drop (green arrow). When dragging, press and hold the <Shift> button. By this, all signals are placed into the graph with the same scale.



```
D:\kkk.dat
  info
  0. Time trackings for project GH10
    0:0: Task_Const_Parameter Prio 0 Order 0 Time 4000 Event 0
    0:1: Task_SQL Prio 0 Order 1 Time 40 Event 0
    0:2: Task_Times Prio 0 Order 10 Time 120 Event 0
    0:3: Task_Input_TCPIP Prio 0 Order 11 Time 40 Event 0
    0:4: Task_Input_Serial Prio 0 Order 12 Time 40 Event 0
    0:5: Task_HK Prio 0 Order 13 Time 480 Event 0
    0:6: Task_ModBusIO_A_KG Prio 0 Order 30 Time 40 Event 0
    0:7: Task_ModBusIO_A_EG Prio 0 Order 31 Time 40 Event 0
    0:8: Task_ModBusIO_A_OG1 Prio 0 Order 32 Time 40 Event 0
    0:9: Task_ModBusIO_A_OG2 Prio 0 Order 33 Time 40 Event 0
    0:10: Task_ModBusIO_BC_KG Prio 0 Order 34 Time 40 Event 0
    0:11: Task_ModBusIO_B_EG Prio 0 Order 35 Time 40 Event 0
    0:12: Task_ModBusIO_B_OG1 Prio 0 Order 36 Time 40 Event 0
    0:13: Task_ModBusIO_B_OG2 Prio 0 Order 37 Time 40 Event 0
    0:14: Task_ModBusIO_CD_EG Prio 0 Order 38 Time 40 Event 0
    0:15: Task_ModBusIO_D_KG Prio 0 Order 39 Time 40 Event 0
    0:16: Task_ModBusIO_A_KG_H Prio 0 Order 50 Time 40 Event 0
    0:17: Task_ModBusIO_A_KG_WP Prio 0 Order 51 Time 40 Event 0
    0:18: Task_ModBusIO_A_KG_L Prio 0 Order 52 Time 40 Event 0
    0:19: Task_ModBusIO_A_OG2_L Prio 0 Order 53 Time 40 Event 0
    0:20: Task_Bewaesserung Prio 0 Order 54 Time 120 Event 0
    0:21: Task_Error_Reporting Prio 0 Order 59 Time 480 Event 0
    0:22: Task_Totmann Prio 0 Order 60 Time 240 Event 0
  1. Digital Time trackings for project GH10
```

Figure 150: Marking the signals

Display of all signals in a graph:

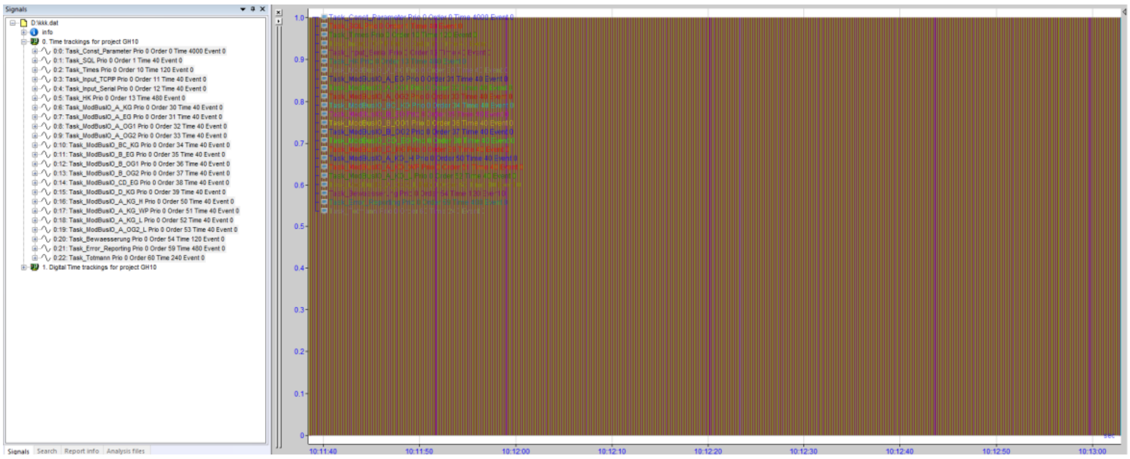
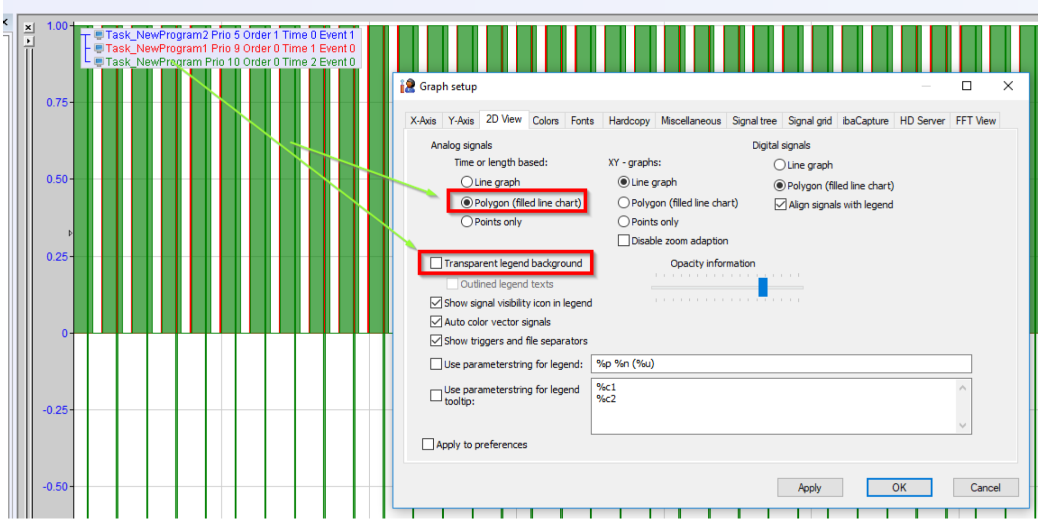


Figure 151: Display of all signals in a graph

The following settings provide a better overview. Open the setup menu with the right mouse button.



Graph setup

Analog signals

Time or length based:

○ Line graph
● Polygon (filled line chart)
○ Points only

XY - graphs:

● Line graph
○ Polygon (filled line chart)
○ Points only
□ Disable zoom adaption

Digital signals

○ Line graph
● Polygon (filled line chart)
☑ Align signals with legend

□ Transparent legend background
□ Outlined legend texts
☑ Show signal visibility icon in legend
☑ Auto color vector signals
☑ Show triggers and file separators

Opacity information

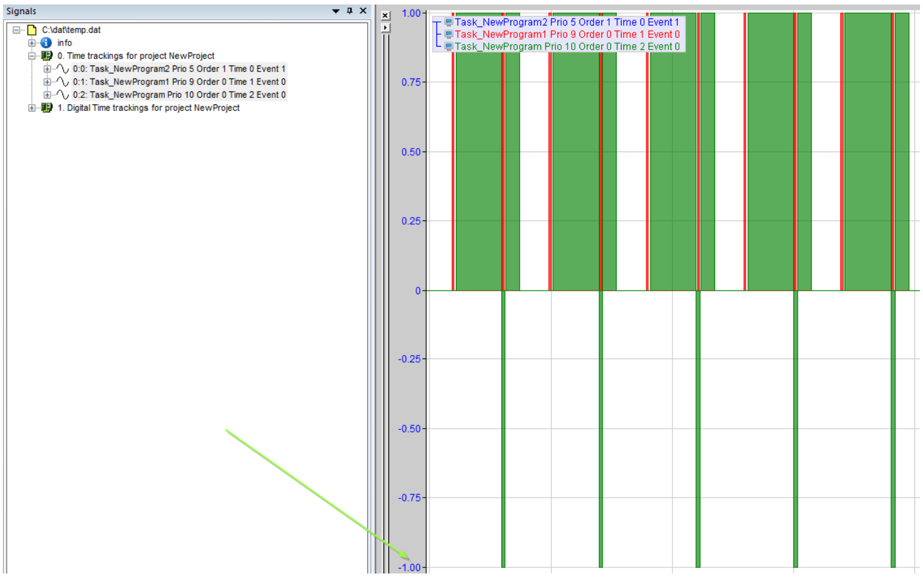
□ Use parameterstring for legend: %p %n (%u)

□ Use parameterstring for legend tooltip: %c1
%c2

□ Apply to preferences

Figure 152: Display settings

The filled line chart clearly shows the individual states, especially when zooming in.


Signals
1.00
0.75
0.50
0.25
0
-0.25
-0.50
-0.75
-1.00
Task_NewProgram2 Prio 5 Order 1 Time 0 Event 1
Task_NewProgram1 Prio 9 Order 0 Time 1 Event 0
Task_NewProgram Prio 10 Order 0 Time 2 Event 0

C:\dat\temp.dat
info
0. Time trackings for project NewProject
0:0: Task_NewProgram2 Prio 5 Order 1 Time 0 Event 1
0:1: Task_NewProgram1 Prio 9 Order 0 Time 1 Event 0
0:2: Task_NewProgram Prio 10 Order 0 Time 2 Event 0
1. Digital Time trackings for project NewProject

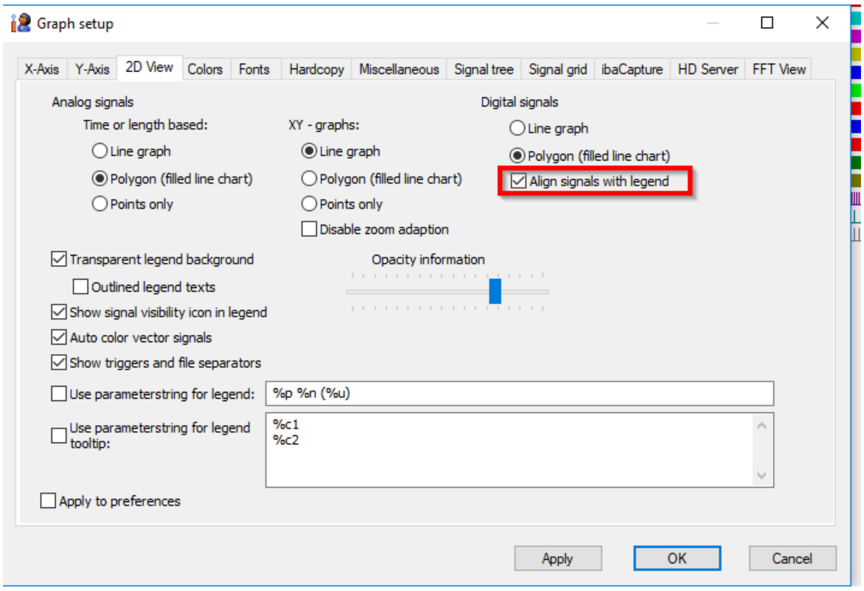
Figure 153: Display of states

Each task is now visible with its evaluation duration and its time of evaluation in relation to all other tasks. Values of -1 indicate that tasks have been interrupted here due to their priority. If there are no interruptions, the digital display is easier to handle.

Display of digital values

If you only use one priority in all tasks, you can also use the digital values.

The digital signals are marked like the analog signals and dragged and dropped into the view while pressing the <Shift> key.



Analog signals

Time or length based:
- ○ Line graph
- ● Polygon (filled line chart)
- ○ Points only

XY - graphs:
- ● Line graph
- ○ Polygon (filled line chart)
- ○ Points only
- □ Disable zoom adaption

Digital signals
- ○ Line graph
- ● Polygon (filled line chart)
- ☑ Align signals with legend

☑ Transparent legend background

□ Outlined legend texts

☑ Show signal visibility icon in legend

☑ Auto color vector signals

☑ Show triggers and file separators

Opacity information

□ Use parameterstring for legend: %p %n (%u)

□ Use parameterstring for legend tooltip: %c1 %c2

□ Apply to preferences

Figure 154: Settings for digital signals

With the "Align signals with legend" setting, the signals are displayed at legend level:

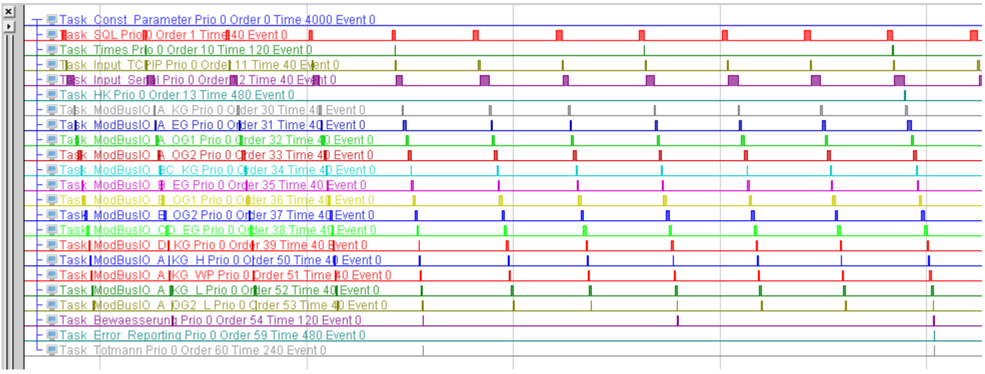

Task Const Parameter Prio 0 Order 0 Time 4000 Event 0
Task SQL Prio 0 Order 1 Time 40 Event 0
Task Times Prio 0 Order 10 Time 120 Event 0
Task Input TCPIP Prio 0 Order 11 Time 40 Event 0
Task Input Serial Prio 0 Order 12 Time 40 Event 0
Task HK Prio 0 Order 13 Time 480 Event 0
Task ModBusIO A KG Prio 0 Order 30 Time 40 Event 0
Task ModBusIO A EG Prio 0 Order 31 Time 40 Event 0
Task ModBusIO A OG1 Prio 0 Order 32 Time 40 Event 0
Task ModBusIO A OG2 Prio 0 Order 33 Time 40 Event 0
Task ModBusIO BC KG Prio 0 Order 34 Time 40 Event 0
Task ModBusIO B EG Prio 0 Order 35 Time 40 Event 0
Task ModBusIO B OG1 Prio 0 Order 36 Time 40 Event 0
Task ModBusIO B OG2 Prio 0 Order 37 Time 40 Event 0
Task ModBusIO CD EG Prio 0 Order 38 Time 40 Event 0
Task ModBusIO D KG Prio 0 Order 39 Time 40 Event 0
Task ModBusIO A KG H Prio 0 Order 50 Time 40 Event 0
Task ModBusIO A KG WP Prio 0 Order 51 Time 40 Event 0
Task ModBusIO A KG L Prio 0 Order 52 Time 40 Event 0
Task ModBusIO A OG2 L Prio 0 Order 53 Time 40 Event 0
Task Bewaesserung Prio 0 Order 54 Time 120 Event 0
Task Error Reporting Prio 0 Order 59 Time 480 Event 0
Task Totmann Prio 0 Order 60 Time 240 Event 0

Figure 155: Display of digital signals

Double-clicking on the respective high state displays the runtime directly. Double-clicking on low state can display the distances.

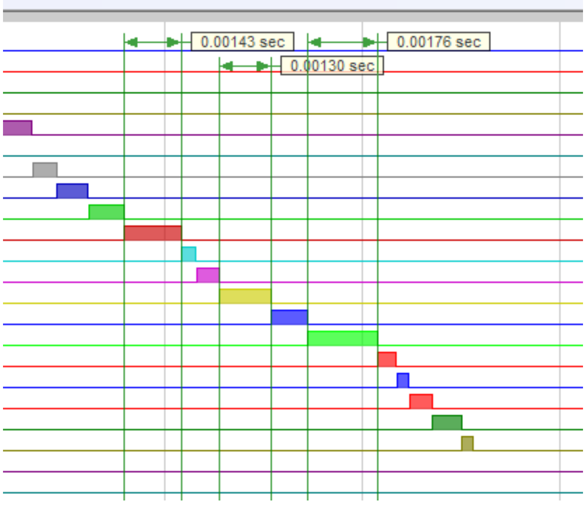


Figure 156: Measurement of runtimes at a mouse click

Note

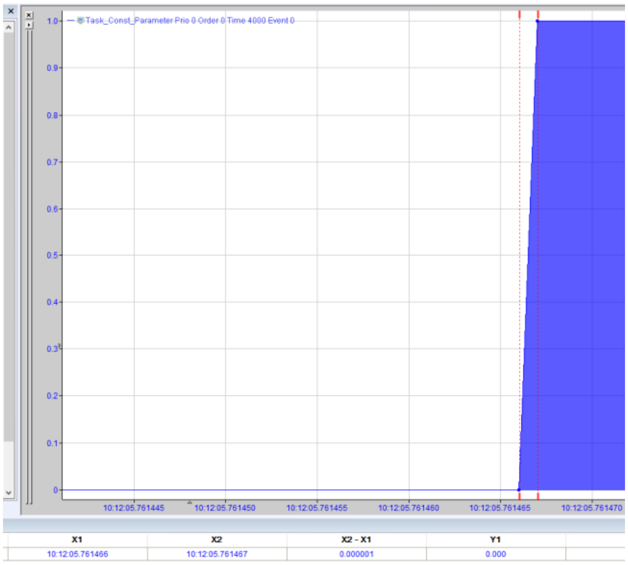
The recording is done with non-equidistant values. Only the status changes are recorded. Therefore, the signals are not suitable to be used in formulas in ibaAnalyzer, as this can only process equidistant signal values.

However, if it is necessary, you can create equidistant signals by the function *Resample*.

Annotation:

Normally, a change of state would make the analog signal look like drawing a slanting line from the last state to the new state. This would make an analysis of the values cumbersome. Therefore, two values are always recorded for each status change.

1 µs before each new state the previous state is repeated. This results in a steep slope that you see only when zooming in extremely. But for the normal view, the analog states can be clearly represented.



X1	X2	X2 - X1	Y1
10:12:05.761466	10:12:05.761467	0.000001	0.000

Figure 157: State change of analog signals

14 Programming rules

Every programming system has an underlying risk of the programming being done in an unstructured manner, and hence, the outcome that the program is very difficult or, in fact, impossible to comprehend for you as a programmer as well as for the customer or any other person working with the system.

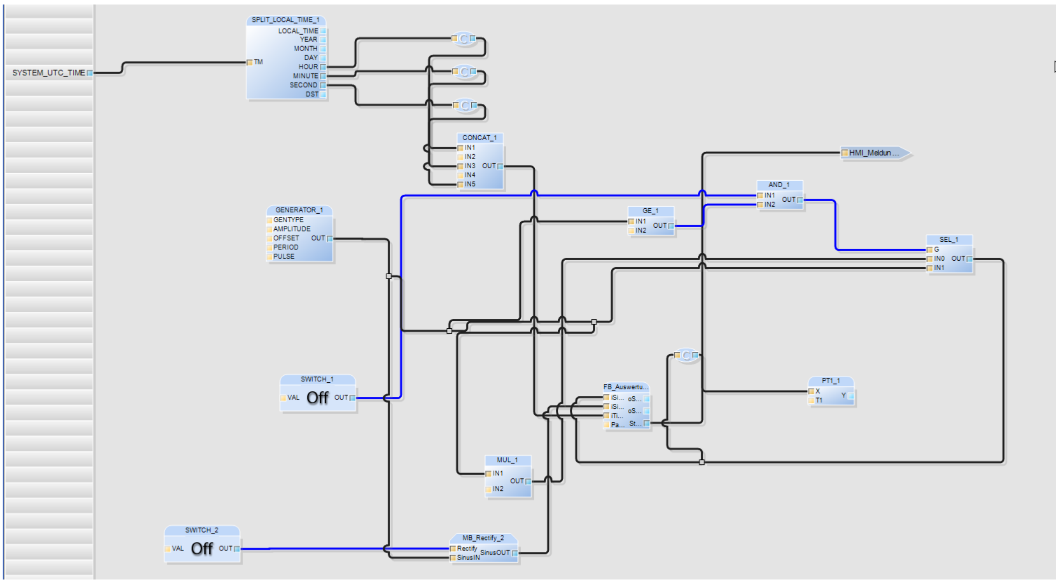


Figure 158: Example of unstructured programming

The example in the figure given above is restructured for the recommended solution.

14.1 Approach for the solution

Two tasks having the following structure:

- ❑ Task 1 Data generation
- ❑ Task 2 Data processing

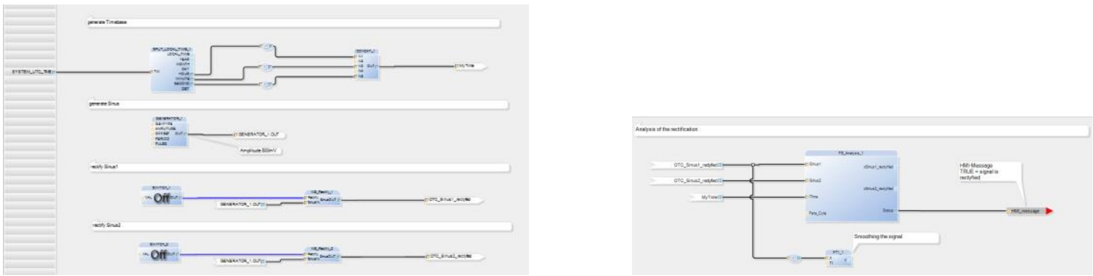


Figure 159: Example of structured programming

You do not have to comply with the following guidelines, but however, they simplify working with ibaLogic.

- ❑ Distribute the functions across multiple tasks / programs having illustrative names.

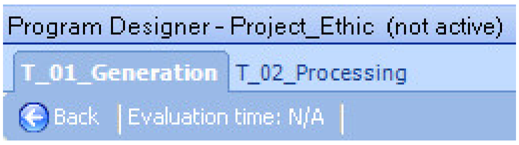


Figure 160: Division of tasks / programs

❑ Create one task each for the hardware inputs and the hardware outputs. Assign the priorities in such a manner that the input task is the first and the output task the last to be processed.

❑ Use intra-page connectors within a task, if too many intersections of the lines make the layout cluttered

❑ Tag the sub-functions using comment fields.

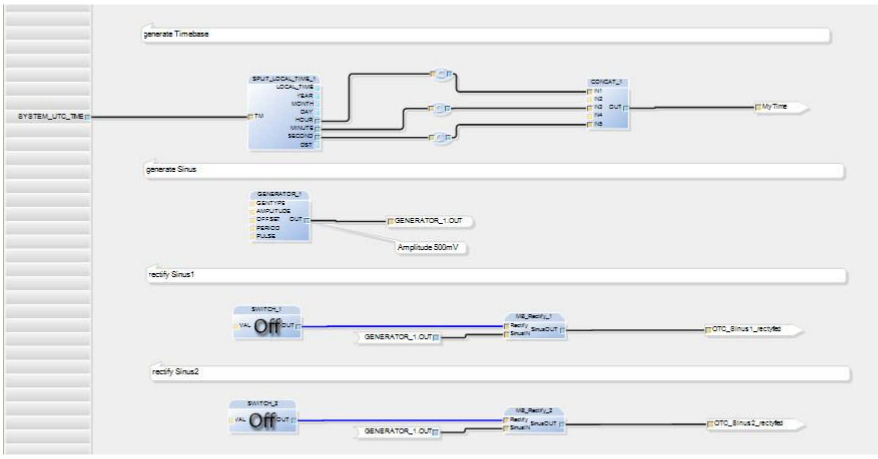
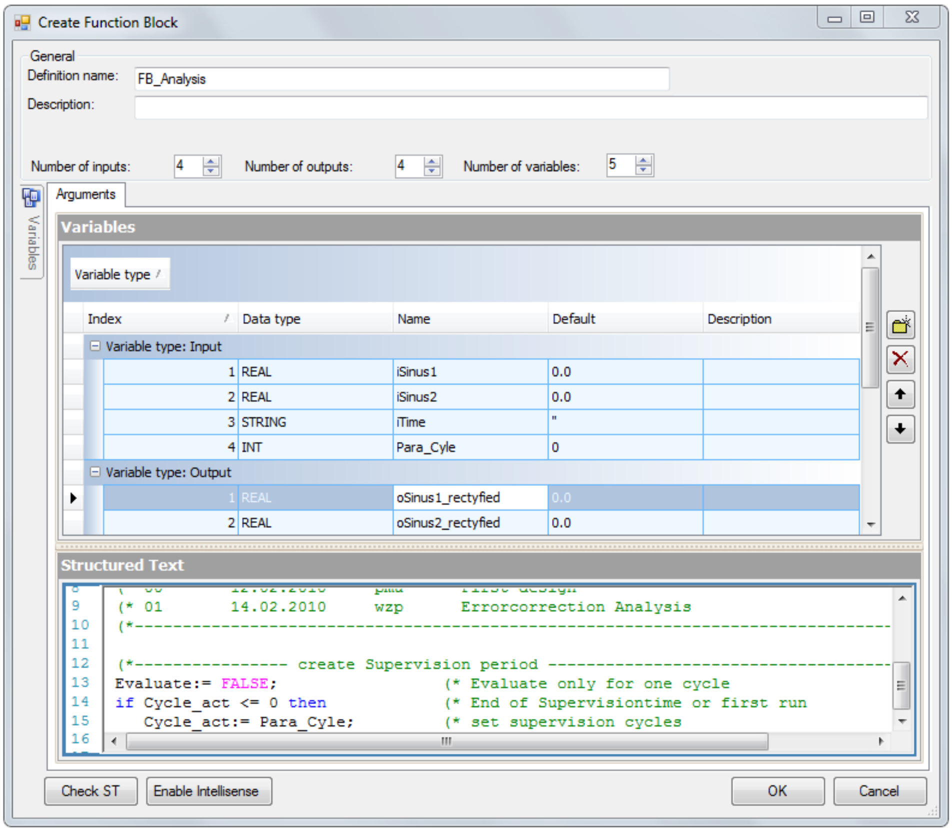


Figure 161: Comments

❑ Merge reusable program components into macro blocks and create meaningful description and comments for them.

❑ Combine complex connections pertaining to a function into a macro in order to improve overall clarity.

❑ Use comments and descriptions even within an FB. We recommend a header with a change index, titles and meaningful indentations of the program lines.



Definition name: FB_Analysis

Description:

Number of inputs: 4 Number of outputs: 4 Number of variables: 5

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	REAL	iSinus1	0.0	
2	REAL	iSinus2	0.0	
3	STRING	iTime	''	
4	INT	Para_Cyle	0	
Variable type: Output				
1	REAL	oSinus1_rectyfied	0.0	
2	REAL	oSinus2_rectyfied	0.0	

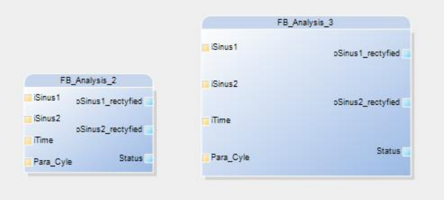
Structured Text

```
(* 00       12.02.2010     pma      First design
(* 01       14.02.2010     wzp      Errorcorrection Analysis
(*---------------------------------------------------------------------------------

(*--------------- create Supervision period ------------------------------------
Evaluate:= FALSE;                          (* Evaluate only for one cycle
if Cycle_act <= 0 then                     (* End of Supervisiontime or first run
   Cycle_act:= Para_Cyle;                  (* set supervision cycles
```

Figure 162: Program code comments

- ❑ Arrange the function units within a task in such a manner that they match the evaluation sequence (from the top left to the bottom right).
- ❑ Designate off-task connectors with a prefix, e.g. "OTC_" or, if the OTC is used for an HMI system, with "OPC_".
- ❑ Designate the intra-page connectors with the prefix "IPC_". If an "OTC_test" is present as an input, it can continue to be used internally as "IPC_test" and there is no repetition of names.
- ❑ Configure short prefixes for the names of function blocks, macros and their connectors, e.g. "FB_", "MB_".
  (Setting under "Tools – Options – Editors – Function Blocks").
- ❑ Assign names to user-defined data types that give a reference to either the logical meaning (e. g. ST_ROLLING) or the contents (e. g. AR_64REAL) of the user data types.
- ❑ If necessary, move the lines in such a manner that you can trace them clearly. Avoid any overlapping.
- ❑ Utilize the option of enlarging the function unit as desired. In doing so, connector names become more legible or the lines are easier to trace.



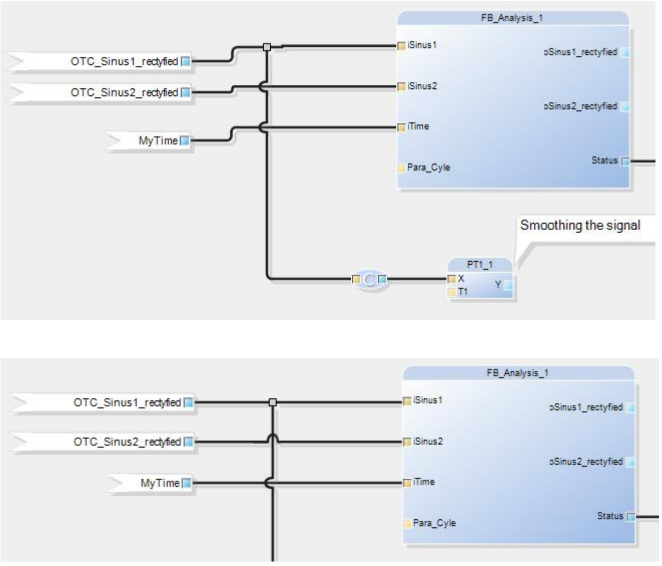


Figure 163: Examples of enlarged display

❑ In general, eliminate possible sources of error while programming, such as:

- Division by 0
- Access beyond the array limits
- Possible endless loops

15 Uninstall ibaLogic

⚠ DANGER

During uninstalling, various messages can occur:

- Query whether the user backups are to be deleted, too.
- Query whether SQL Express is to be uninstalled, too
  (only occurs if the ibaLogic database existed exclusively).

⚠ DANGER

Danger by enabling or disabling functions!

Possibility of human injuries and damage to machinery by enabling or disabling functions and other services (PMAC, OPC ...), which have a direct impact on the response of the system.

Secure the system while working on it!
Follow the safety regulations applicable!

Important Note

Only those users having administrator privileges can uninstall ibaLogic software. Please ask your system administrator.

Prerequisite

❑ All ibaLogic programs are closed.

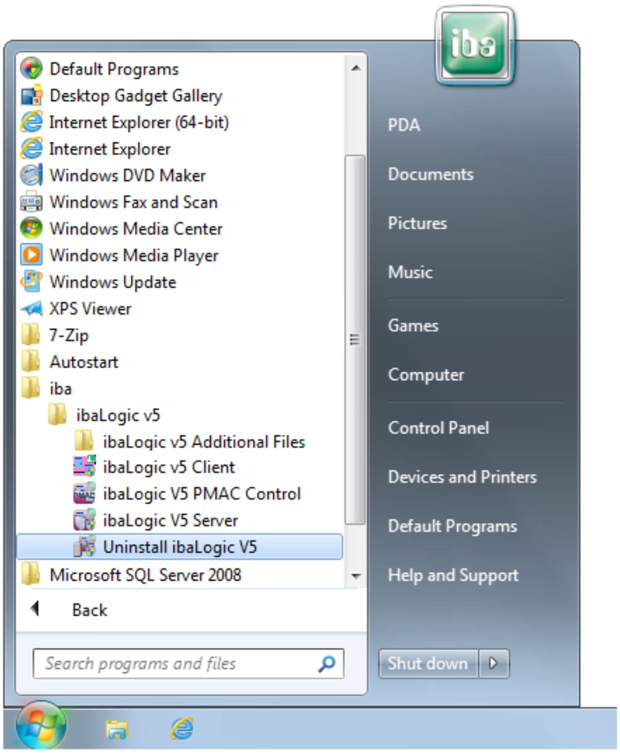
Note

Messages during uninstalling

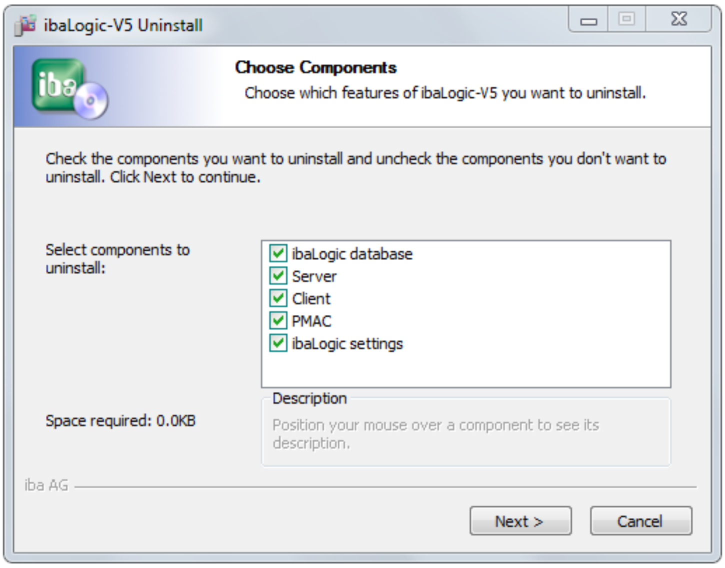
SQL Express Instance is removed by confirming the prompt with <Yes>. The database created during installation is deleted (*.ldf, *.mdf).

Procedure

1. Select "Start – iba – ibaLogic v5 – Uninstall ibaLogic V5".



2. Choose the components that you wish to uninstall.



Choose Components

Choose which features of ibaLogic-V5 you want to uninstall.

Check the components you want to uninstall and uncheck the components you don't want to uninstall. Click Next to continue.

Select components to uninstall:

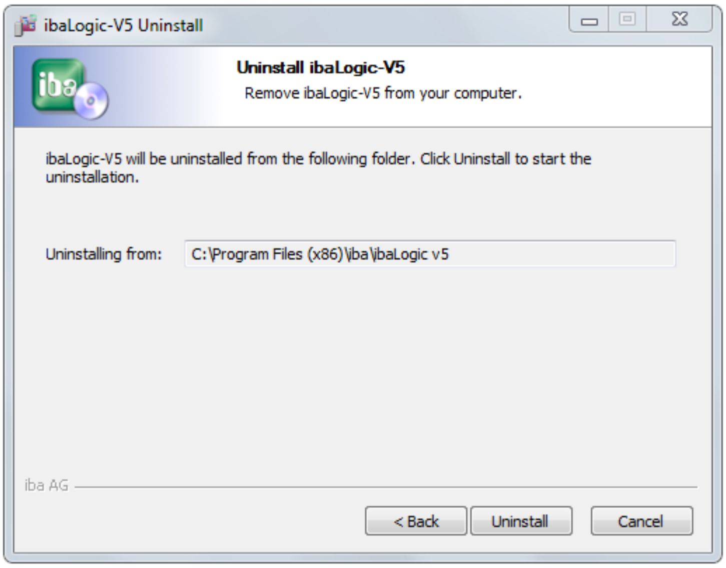
☑ ibaLogic database
☑ Server
☑ Client
☑ PMAC
☑ ibaLogic settings

Description

Position your mouse over a component to see its description.

Space required: 0.0KB

3. Start the uninstalling procedure by pressing the <Uninstall> button.



ibaLogic-V5 Uninstall

Uninstall ibaLogic-V5

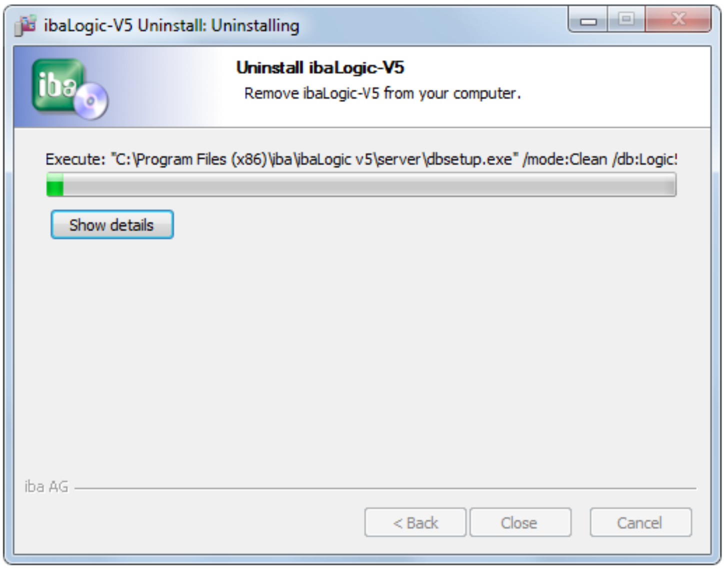
Remove ibaLogic-V5 from your computer.

ibaLogic-V5 will be uninstalled from the following folder. Click Uninstall to start the uninstallation.

Uninstalling from: C:\Program Files (x86)\iba\ibaLogic v5

iba AG

4. Close the dialog box by pressing the <Close> button. Confirm any confirmation prompts if required.



ibaLogic-V5 Uninstall: Uninstalling

Uninstall ibaLogic-V5

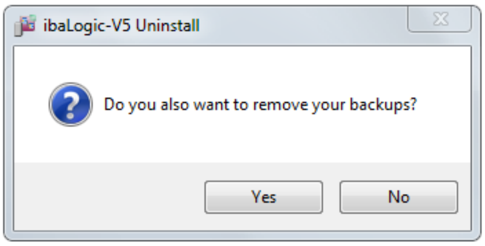
Remove ibaLogic-V5 from your computer.

Execute: "C:\Program Files (x86)\iba\ibaLogic v5\server\dbsetup.exe" /mode:Clean /db:Logic!

iba AG

Important Note

If there are any database backup copies in the installation folder, you are asked during the uninstall process whether they should also be deleted. If you confirm with <No>, the backup copies remain.



ibaLogic-V5 Uninstall

Do you also want to remove your backups?

16 Practice Examples

The aim of this section is to accompany the "beginner" with the first steps in using ibaLogic.
It is intended that a small sample program generates the "Aha effect" and, above all, demonstrates what iba understands of ergonomic CFC implementation, and what meaning is given in the process to online update of static and dynamic variables.

Since this is merely an introductory example, it does not illustrate or deal with all functions of ibaLogic.

16.1 First Steps - Sample Project

The task is to create a program for generating a sinusoidal signal. A smoothly adjustable offset should be added to this sinusoidal signal depending on the status of a switch. You can create the example completely with the help of standard function blocks. Nonetheless, in order to be able to explain the highly flexible programming opportunities and features of the integrated programming language, "Structured Text" (ST), the example should also be programmed using this alternative.

The input signals and the result need to be displayed as trend graphs.

The sample program is made "live" so that the connectors used are updated continuously. The change of color to pink indicates that everything is now "serious": The plant is virtually live!

If outputs are already connected (actuators, motors, etc.), these would respond immediately to the program modifications. The customary procedure - programming, compiling, linking and loading, as well as starting - runs automatically in the background. Moreover, for the sake of simplification, the preset values for the project and program names are accepted.

16.1.1 Sample Exercise Part 1

16.1.1.1 Task Description

The periodically changing value of a generator (sinusoidal signal) needs to be added to the value of a slide controller depending on the status of a switch:

Switch position 1: Generator + Generator

Switch position 2: Generator + slide controller

All variables and, of course, even the result needs to be displayed in a graphics format as a trend curve.

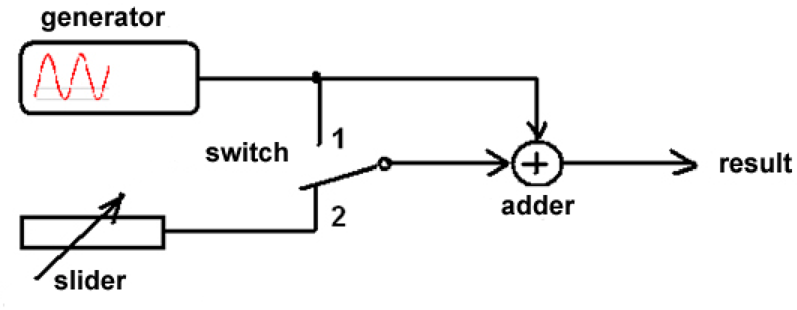


Figure 164: Circuit diagram of the sample exercise

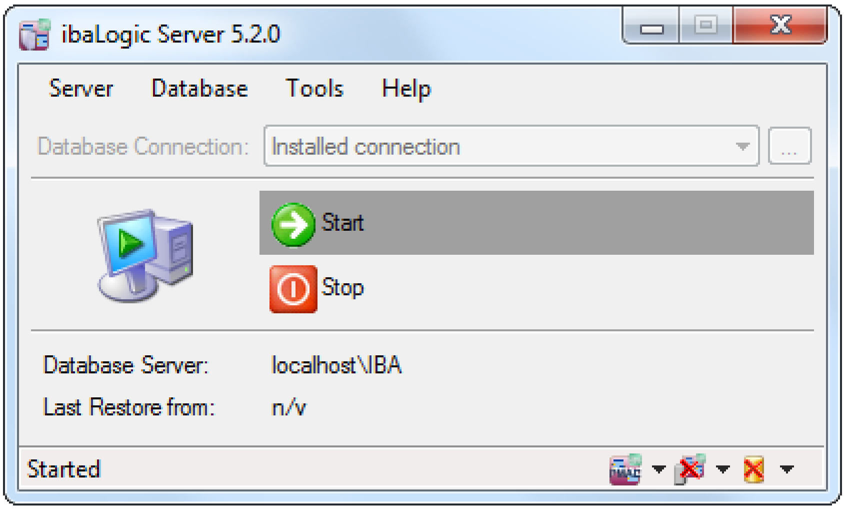
In the first part of the exercise, the example should be implemented with the help of function blocks (FB) that are available as standard function blocks in ibaLogic.

Since laboratory equipment such as a function generator, slide controller and keys do not have to be connected to the inputs of ibaLogic, such effective functions have been compiled as **Specials** and can be placed like other function blocks. You can perform active operations on the **Switch** and the **Slider** to test the circuit.

16.1.1.2 Start ibaLogic Server and ibaLogic Client

Procedure

1. Double click on the "ibaLogic Server" icon on the desktop.
   The ibaLogic server dialog box opens after the initialization phase. By default, the server is started automatically when the dialog box opens.



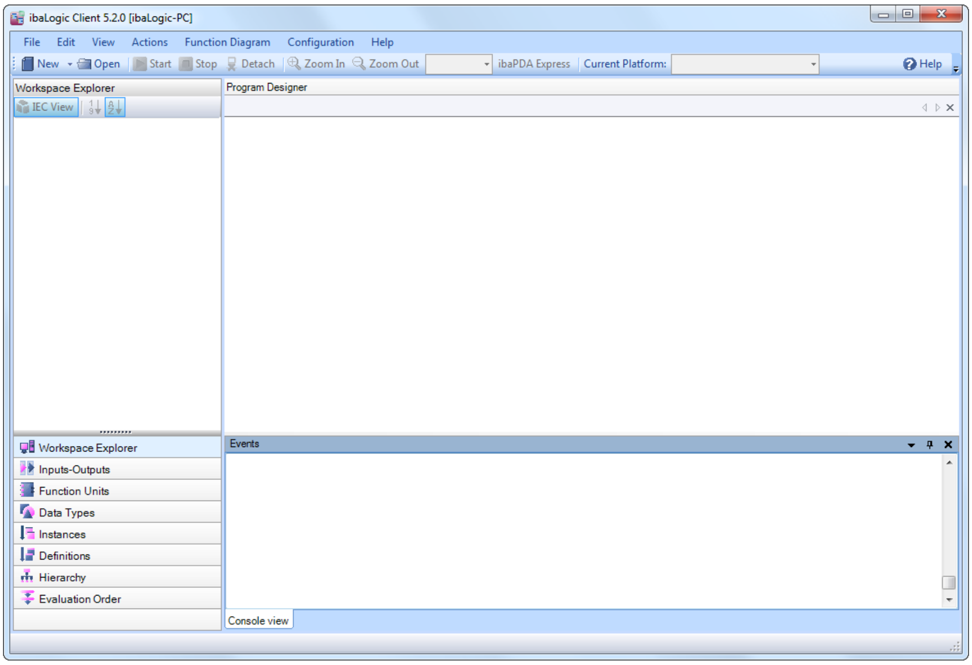
Database Connection: Installed connection

Database Server: localhost\IBA

Last Restore from: n/v

Started

2. Double click on the "ibaLogic Client" icon on the desktop.
   The ibaLogic client dialog box opens after the initialization phase.
   (It might be necessary to acknowledge the connection once again).


ibaLogic Client 5.2.0 [ibaLogic-PC]
File Edit View Actions Function Diagram Configuration Help
New Open Start Stop Detach Zoom In Zoom Out ibaPDA Express Current Platform: Help
Workspace Explorer
IEC View
Program Designer
Workspace Explorer
Inputs-Outputs
Function Units
Data Types
Instances
Definitions
Hierarchy
Evaluation Order
Events
Console view

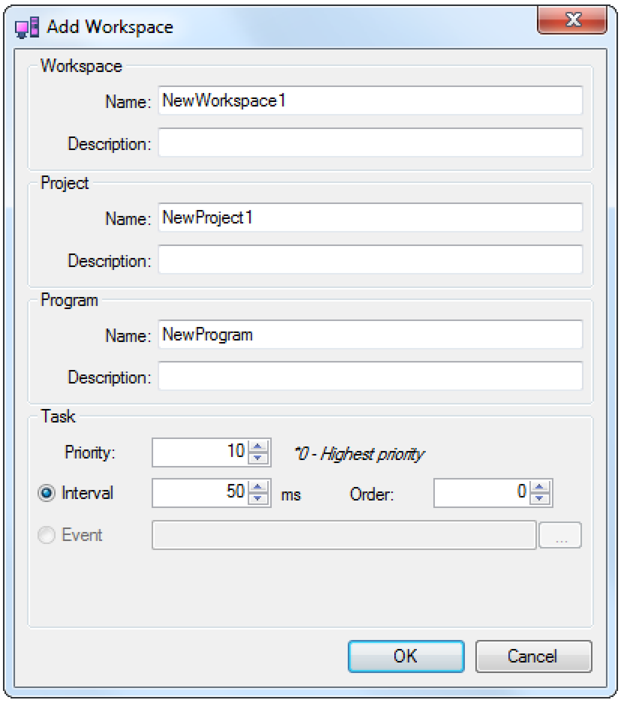
Remark

The event window below the program window documents the program actions and collisions, if any. If error messages pop up during the startup of the server or the client, please refer to this documentation for assistance.

16.1.1.3 Create a New Project

Procedure

1. Press the <New> button in the toolbar.
   The "Add Workspace" dialog box is displayed.



Workspace
Name: NewWorkspace1
Description:

Project
Name: NewProject1
Description:

Program
Name: NewProgram
Description:

Task
Priority: 10 "0 - Highest priority
● Interval 50 ms Order: 0
○ Event

2. Confirm the entries with <OK>.

When you do not change the presets, your project is called "NewProject1" having just one program "NewProgram". The preset interval time of 50 ms is adequate for the example.

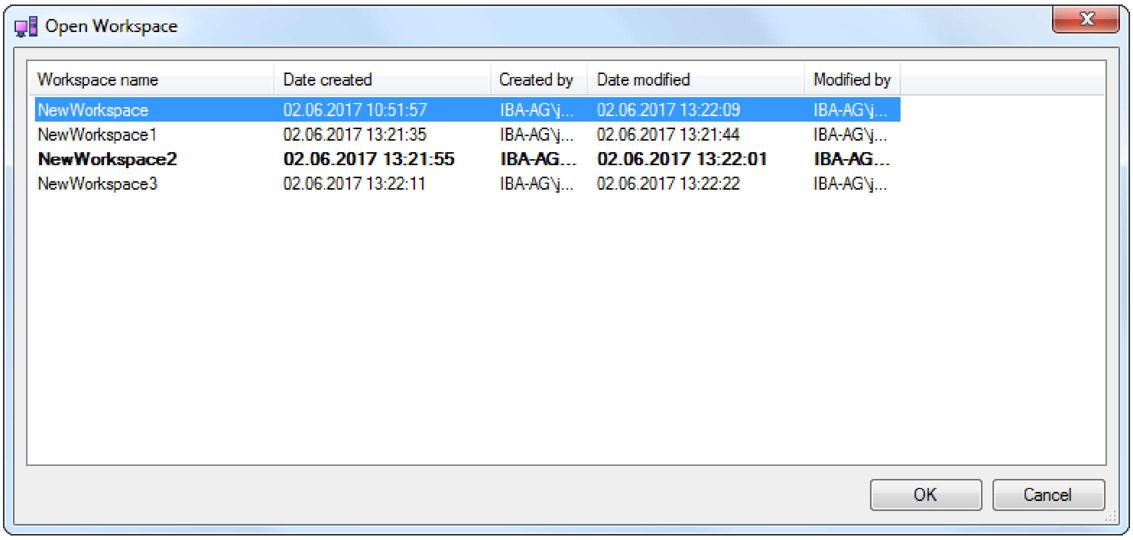
Remark

If you should have to interrupt the "session", after beginning your exercise with ibaLogic, or you have simply closed all programs (first the client and then the server), you do not have to begin with <New>.

Your changes are saved automatically.

When you continue in such a case, press the "Open" button, open the workspace you created and continue working at the position where you have stopped.

If you have several workspaces, the search can be limited by the "Date modified" field.



Workspace name	Date created	Created by	Date modified	Modified by
NewWorkspace	02.06.2017 10:51:57	IBA-AG\j...	02.06.2017 13:22:09	IBA-AG\j...
NewWorkspace1	02.06.2017 13:21:35	IBA-AG\j...	02.06.2017 13:21:44	IBA-AG\j...
NewWorkspace2	**02.06.2017 13:21:55**	**IBA-AG...**	**02.06.2017 13:22:01**	**IBA-AG...**
NewWorkspace3	02.06.2017 13:22:11	IBA-AG\j...	02.06.2017 13:22:22	IBA-AG\j...

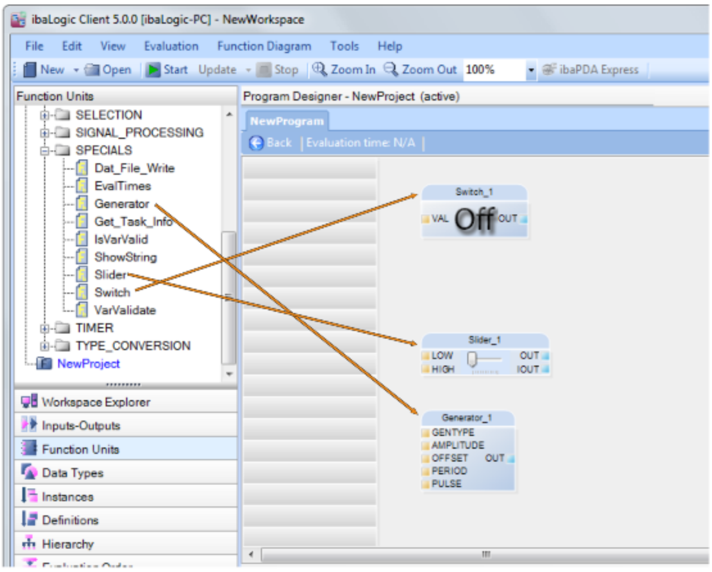
16.1.1.4 Placing the Test Tools

One function block each is required for the task description:

- ❑ "Generator"
- ❑ "Switch"
- ❑ "Slider"

Procedure

1. Click on the <Function Units> button. A folder tree opens in the navigation area.
2. Open the "Specials" folder.
3. Drag one "Generator", one "Slider" and one "Switch" to the design area of the program designer keeping the left mouse button pressed.



4. Arrange the function blocks in the proper sequence.

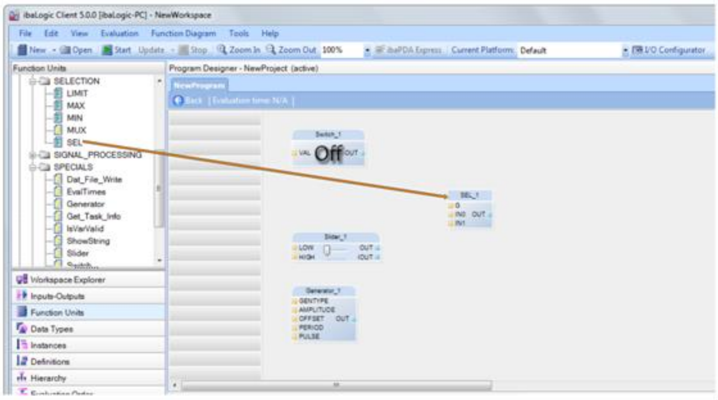
16.1.1.5 Placing the function units

One function unit each is required for the task description:

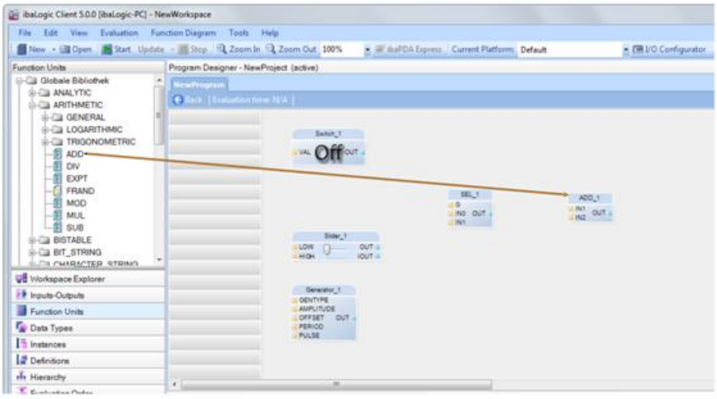
- ❑ "Selector"
- ❑ "Adder"

Procedure

1. Open the "Selection" folder in the directory tree of the navigator.
2. Drag the selector (SEL), keeping the left mouse button pressed, to the design area of the program designer.



3. Open the "Arithmetic" folder in the directory tree of the navigator.
4. Drag the adder (ADD), keeping the left mouse button pressed, to the design area of the program designer.



Remarks

The SELECTOR (SEL) requires a binary signal as the "Decision-maker" (Selector). The values IN0 and IN1 under consideration for the selection are format-free and they adapt themselves automatically to the data type with which they are connected.

As already described with the selector, the values to be added are format-free to begin with. It is only the connection that is first "joined" to one of the inputs that determines the format.

16.1.1.6 Connecting the selector function with the test tools

The following need to be connected:

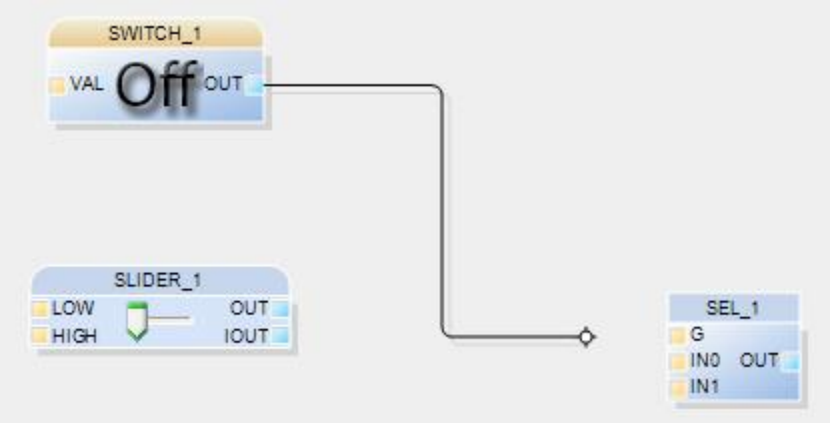
❑ The switch output (SWITCH) OUT with the decision-making input G of the elector (SEL).

❑ The slider output (SLIDER) OUT with the input IN0, of the selector (SEL).

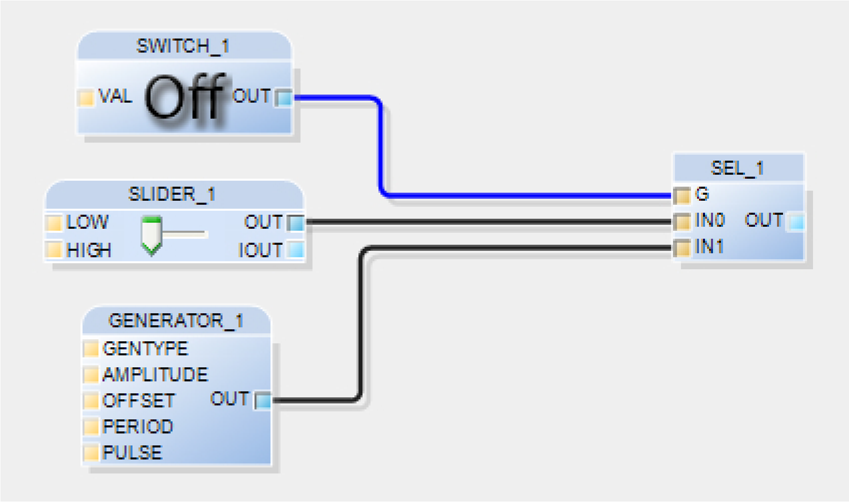
❑ The generator output OUT with the second input IN1 of the selector (SEL).

Procedure

1. Place the mouse pointer on the connector (OUT) of the switch and drag the mouse pointer, keeping the left mouse button pressed, to the connector (G) of the selector.



2. Complete the remaining connections accordingly.



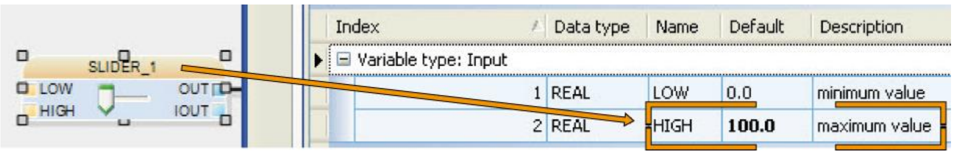
16.1.1.7 Configuring the slider and generator

The following need to be configured:

- ❑ "Slider"
- ❑ "Generator"

Procedure

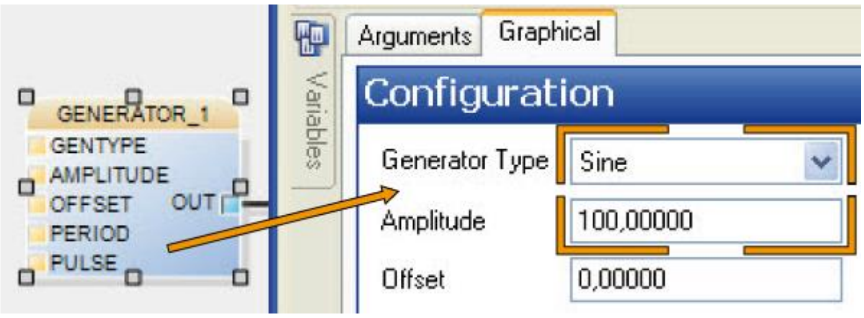
1. Double click on the slider. The configuration menu pops up.
2. Set the "maximum value" of the working range to 100.0. The slider works with a value between 0 and 100 depending on the slider position.



Variable type: Input

Index	Data type	Name	Default	Description
1	REAL	LOW	0.0	minimum value
2	REAL	HIGH	**100.0**	maximum value

3. Open the configuration menu of the generator.
4. Set the generator type to sinusoidal.
5. Enter an amplitude value of 100.



Configuration

Generator Type: Sine

Amplitude: 100,00000

Offset: 0,00000

With this setting, the generator produces a sinusoidal waveform having a value between +100.0 and -100.0.
Leave the period of the oscillations, as preset, at 10 sec.

16.1.1.8 Switch the partial connections online

The online compiler is also activated by starting the evaluation.

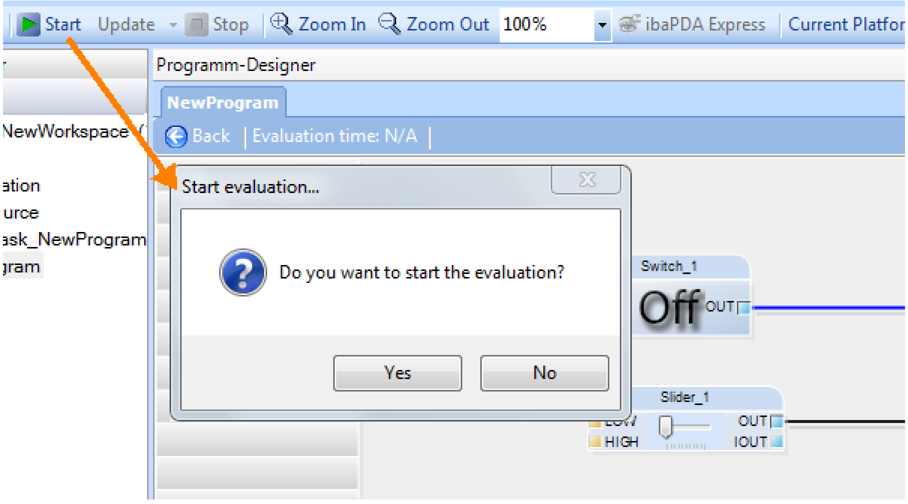
- The background color of the program designer changes to pink.
- All binary connecting lines are colored depending on their status.
- The function unit connectors are updated continuously and displayed.
- All connections are now "live".

Figuratively speaking, this is comparable to applying a voltage to a test circuit.

In a real-life situation, machines are controlled by ibaLogic via the outputs connected. In such a case, the impact of "Connecting live" in the case of a program bug would be felt drastically.

Procedure

- Click on the <Start> button in the toolbar to start the evaluation of the sample exercise. Confirm the prompt with <Yes>. The online properties mentioned above get activated.



Start evaluation...

Do you want to start the evaluation?

16.1.1.9 Testing the switch and selector

Since the selector is already connected and, moreover, the circuit is switched online, its function can be tested immediately independent of the switch status of the input (G). How the selectors work is described under "*Standard Function Units*, page 301" in this manual. You can now implement these explanations in practice.

Procedure

- Change the status of the switch by right-clicking on the SWITCH.

As you can see, in the switched off status (Off), the slider output (value = 43.5) is connected to the SEL output (OUT).

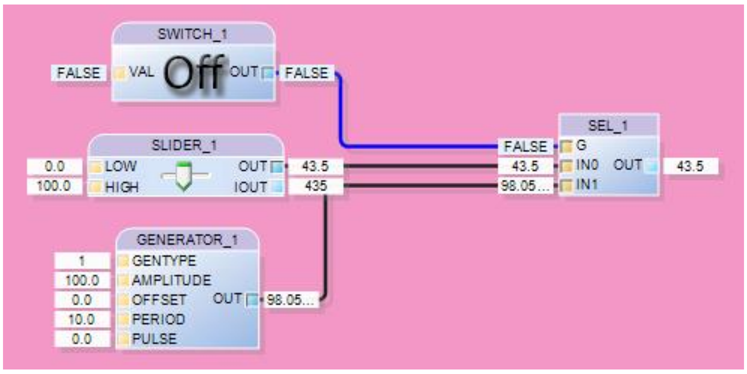


Figure 165: Slider output (Value = 43.5) connected to the SEL output (OUT)

In the switched on state (ON), the periodically changing generator output is connected to the SEL output (OUT).

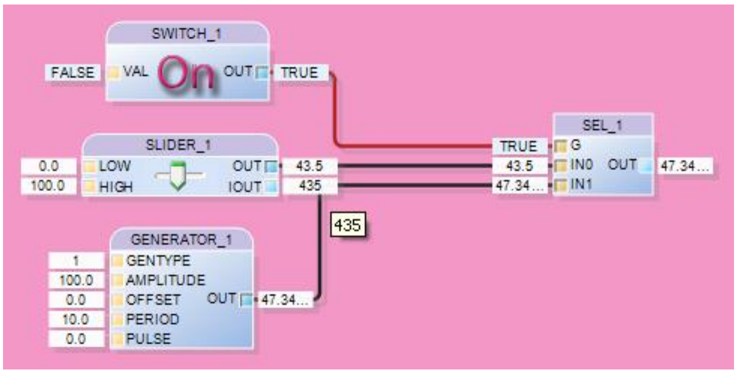


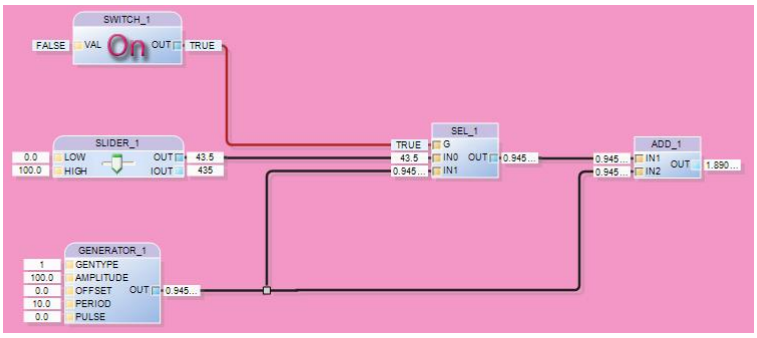
Figure 166: Generator output connected to the SEL output (OUT)

16.1.1.10 Connecting the adder

Connect the adder in order to complete the circuit as given in the sample exercise.

Procedure

1. Place the mouse pointer on the selector output (OUT) and drag the mouse pointer, keeping the left mouse button pressed, to the input (IN1) of the adder.



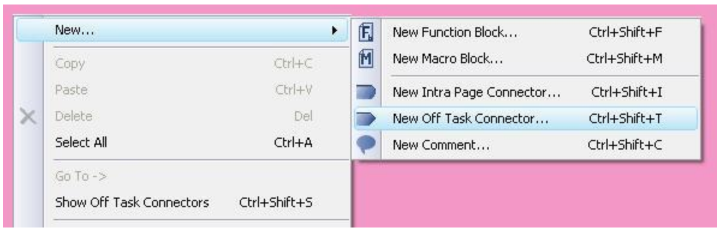
2. Place the mouse pointer on the input (IN2) of the adder, and drag the mouse pointer, keeping the left mouse button pressed, to the generator output (OUT).
The connection point above the mouse pointer moved locks in position automatically and forms a branch.

16.1.1.11 Create an OTC to illustrate the result

In our sample exercise, an off-task connector is placed and also connected to illustrate the result of this simple example. The OTC is called "Result".

Procedure

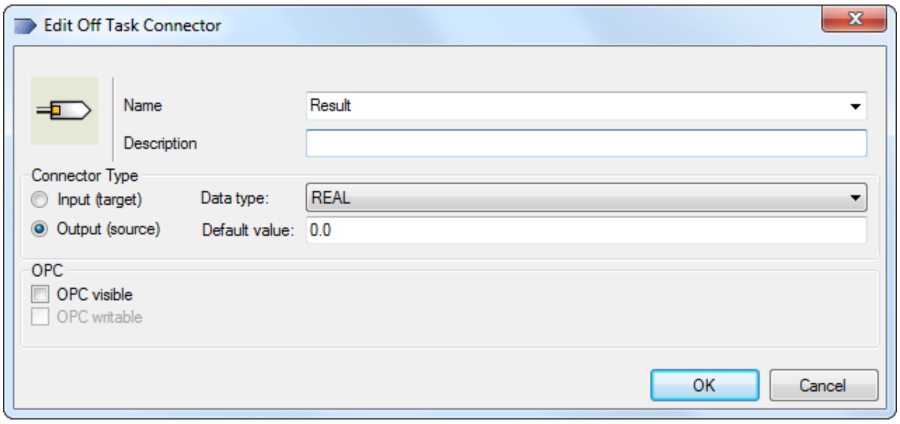
1. Click with the right mouse button on the design area of the program designer.
2. Select "New... – New Off Task Connector...".


New...
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Select All Ctrl+A
Go To ->
Show Off Task Connectors Ctrl+Shift+S
New Function Block... Ctrl+Shift+F
New Macro Block... Ctrl+Shift+M
New Intra Page Connector... Ctrl+Shift+I
New Off Task Connector... Ctrl+Shift+T
New Comment... Ctrl+Shift+C

The window "Off-Task connector edit dialog" is displayed.

3. Assign the name "Result" in the input field. All other preset values remain as they are. Configure the data type as "REAL" and set a default value of 0.0.

   The preset value as output is correct, since a value is fed to the OTC.
   The objective would be the OTC, if it were to receive a value transferred from another program via the OTC.



Name: Result

Description:

Connector Type

○ Input (target)

● Output (source)

Data type: REAL

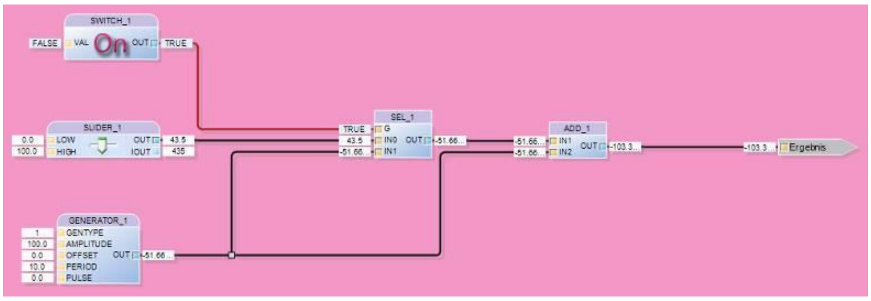
Default value: 0.0

OPC

□ OPC visible

□ OPC writable

4. Connect the OTC "Result" with the adder.



Remarks

An OTC is a key element in graphics programming.

The OTC facilitates the clarity of a project by virtue of the fact that it is distributed over several programs that communicate with one another via OTCs.
In contrast, the inter-page connector is used within a program.

In this manual, you will also find "*Programming rules*, page 264". This section gives ideas about how you can make use of OTCs and IPCs in order to avoid tapeworm programming (entangled connections).

A highly significant communication element is the OTC for connecting with a supervisory HMI (Human-Machine Interface) system.

16.1.1.12 Analysis of the circuit

You can control the result with the help of the dynamic display of the connectors. However, even in this simple program running with a 50 ms cycle time, checking the numerical results is extremely difficult.

Please pay attention to the "*Sample Exercise Part 2*, page 281" in the following.

16.1.2 Sample Exercise Part 2

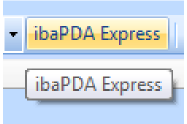
This sample exercise demonstrates the comfort of a dynamic online trend curve display to evaluate the result.

16.1.2.1 Program analysis using the ibaPDA Express

An ibaPDA display is integrated into ibaLogic for online display of trend curves.

Procedure

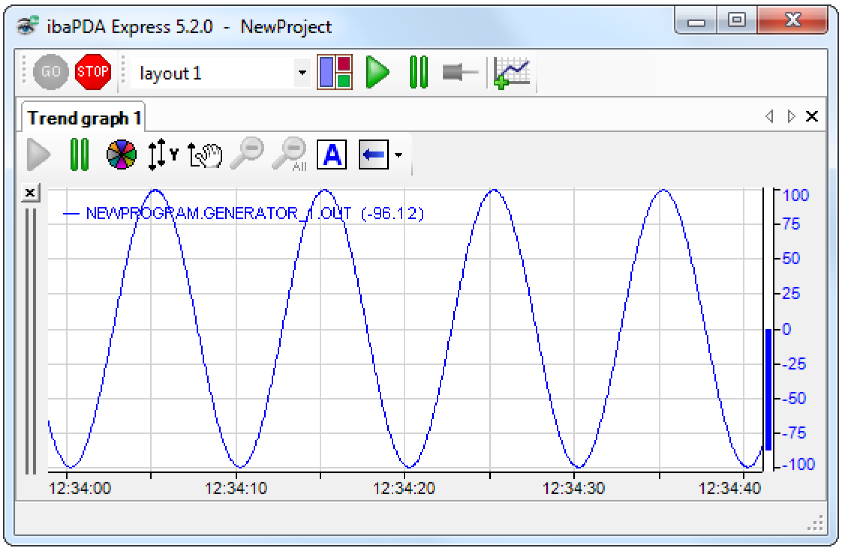
➲ Click on the integrated ibaPDA Express in the toolbar.
This program is integrated in every version of ibaLogic at no extra cost.



➲ Move the mouse pointer to the OUT connector of the generator and drag it keeping the <ALT> button pressed and drop it in the ibaPDA Express window.

➲ Track the intermediate value selected (Generator OUT) as a trend curve display.

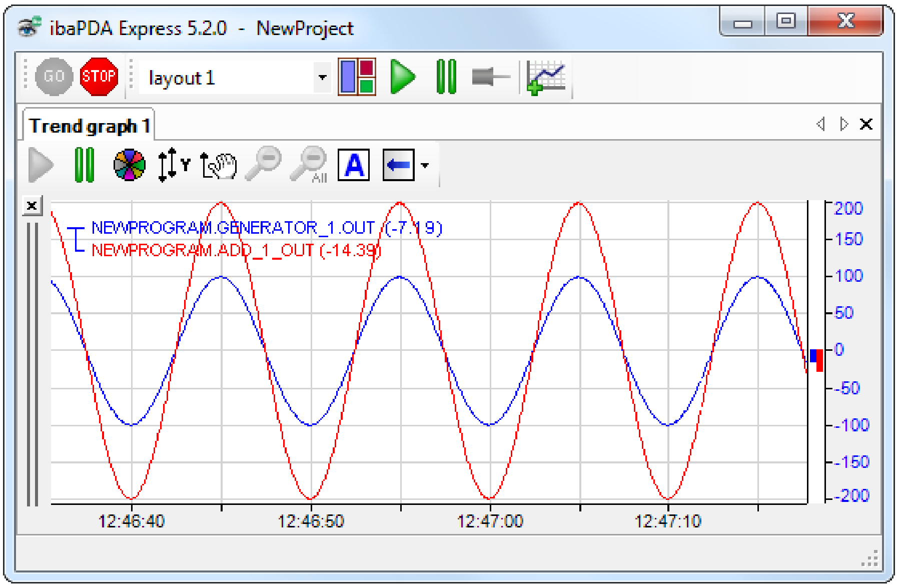
➲ Adjust the Y-axis using the <Auto scale All> button.



➲ Drag the intermediate values and results of interest into the ibaPDA Express window.

- If these are variables having the same scale, drag the values to the connector text of the signal already being displayed.
- If the scales are different, drag and drop the connectors into the trend graph window and a new scale is formed.
- In order to open a new trend graph window, pull the connector to an area outside the current trend graph window.

➲ Carry out a final check on the sample exercise by changing the switch position to on / off. Observe the changes in the trend graph display.


ibaPDA Express 5.2.0 - NewProject
GO
STOP
layout 1
Trend graph 1
NEWPROGRAM.GENERATOR_1.OUT (-7.19)
NEWPROGRAM.ADD_1_OUT (-14.39)
200
150
100
50
0
-50
-100
-150
-200
12:46:40
12:46:50
12:47:00
12:47:10

16.1.3 Sample Exercise Part 3

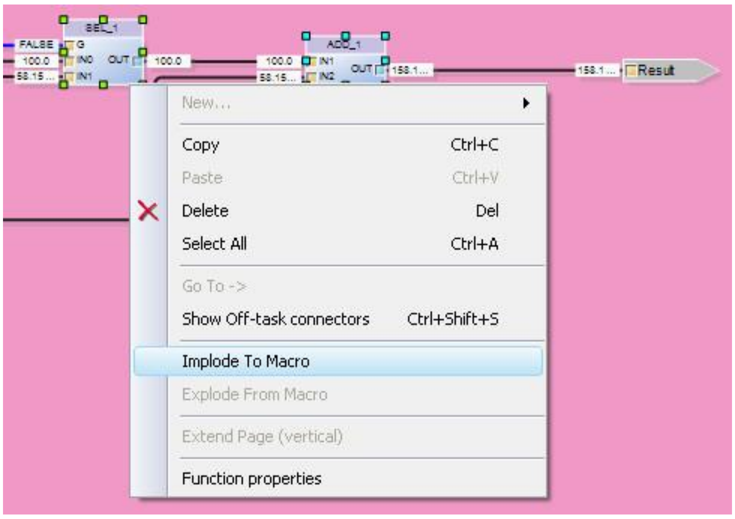
Improving the program clarity and readability

You can combine function units to macros in order to improve the clarity and readability of a draft circuit.

16.1.3.1 Procedure

This technique is demonstrated with this small sample exercise.

- Mark the relevant FBs and their connecting lines with the <Ctrl> button pressed at the same time. In this case, mark the selector and adder, and their connecting line.



- Click the right mouse button with the <Ctrl> key pressed, which displays the macro menu.
- Select "Implode To Macro" in the macro menu.
- Confirm the intermediate screen (Macro, inputs, outputs).

16.1.3.2 Remark

A macro is created. The macro created with a preset name (IMPL_MB_1) performs the same functions as the individual function units selected previously. However, it can be designated appropriately for a meaningful and proper documentation.

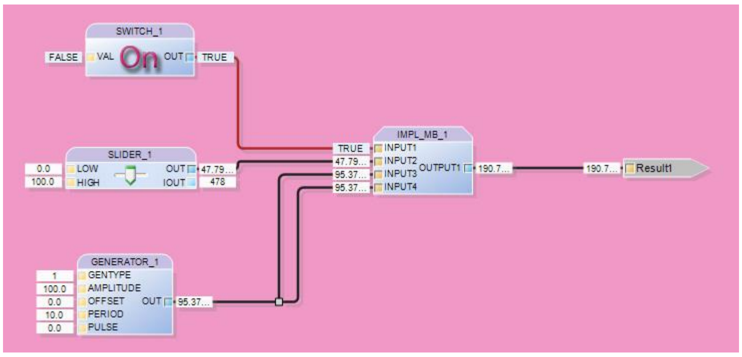
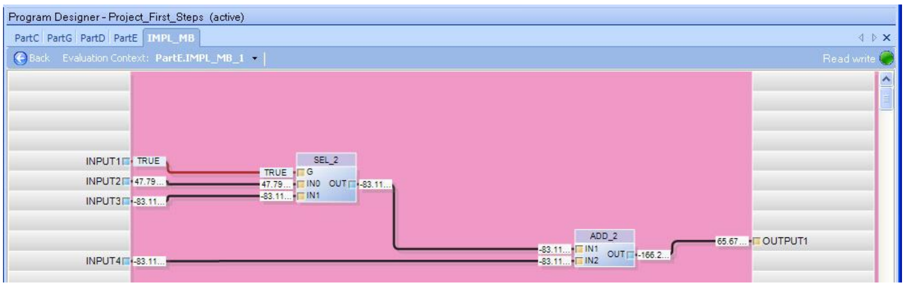


Figure 167: Macro creation

Tip

You can use the lasso method to combine a larger number of FBs into one macro. Keep the left mouse button pressed and mark a rectangle across the desired objects.

- Double click on the macro generated to display its contents. Some of the lines or the FBs are placed in a somewhat haphazard manner. You can also edit in the macro, e. g. add connecting lines that you have overlooked or only "clean it up" graphically.



➲ Select <Back> to quit the macro zoom display.

16.1.4 Sample Exercise Part 4

Creating function blocks using ST

The function blocks provided by default are highly comprehensive in accordance with the standard. Based on the experience of their own applications and, above all, based on the suggestions of ibaLogic users, some other useful and special FBs have been developed and included as standard function blocks.

In order to demonstrate to the users the options and methods of creating special function blocks on their own, one FB will be programmed at the end of this sample exercise. As a functional test, this new FB runs in parallel to the macro created earlier. The results must be congruent with one another.

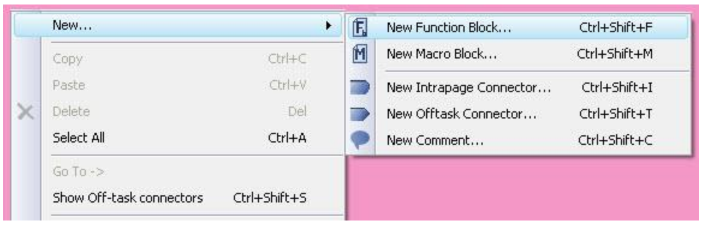
The IEC 61131-3 standard has given special importance to the "Structured Text" programming language. With the help of this higher-level language, you can program very clear and easily readable FBs along the lines of the Pascal programming language. The ST function blocks can also be ported to third-party systems.

If you come from the classical PLC world, you are familiar with the use of standard FBs.

On the contrary, if you are an assembler or high-level language programmer, you would probably cope better with Structured Text programming.

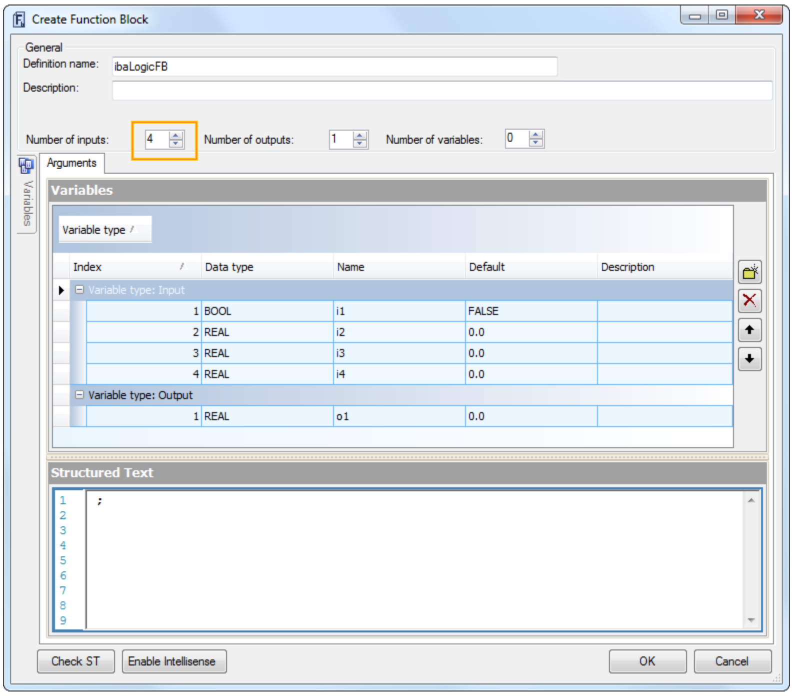
16.1.4.1 Procedure

➲ Select "New... – New Function Block…" in the context menu.
The "Create Function Block" dialog box is displayed.


New...
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Select All Ctrl+A
Go To ->
Show Off-task connectors Ctrl+Shift+S
New Function Block... Ctrl+Shift+F
New Macro Block... Ctrl+Shift+M
New Intrapage Connector... Ctrl+Shift+I
New Offtask Connector... Ctrl+Shift+T
New Comment... Ctrl+Shift+C

➲ Enter "FB_1x" as definition name.

➲ Set the "Number of inputs" to 4. Since the macro from the sample exercise part 3 needs to be cloned, the FB needs as many inputs and outputs (4 inputs and one output).



Definition name: ibaLogicFB

Description:

Number of inputs: 4 Number of outputs: 1 Number of variables: 0

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	BOOL	i1	FALSE	
2	REAL	i2	0.0	
3	REAL	i3	0.0	
4	REAL	i4	0.0	
Variable type: Output				
1	REAL	o1	0.0	

Structured Text

```
;
```

➲ Configure the data types for the inputs and the output.

For the inputs

❑ 1x BOOL

❑ 3x REAL

For the output

❑ 1x REAL

Using the button "", you create a new variable.

➲ Enter a semicolon in the "Structured text" field and click <OK>.

16.1.4.2 Remark

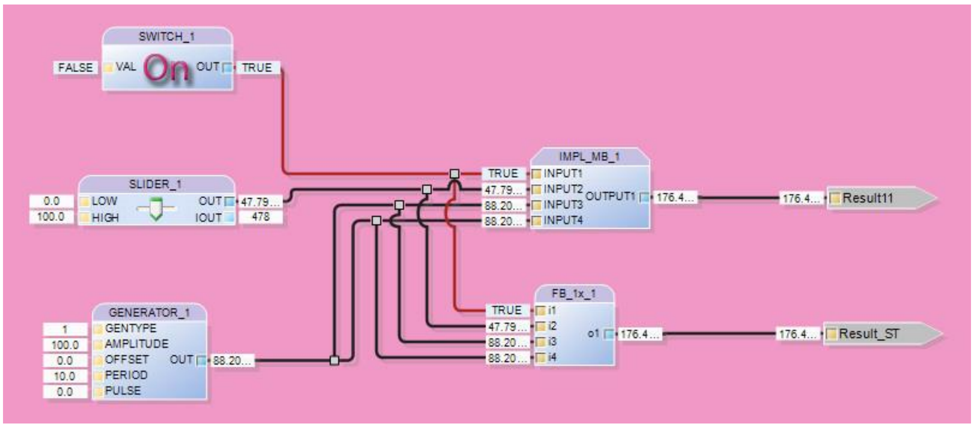
In the first approach, merely an "empty" function block is created.

The ST field must contain at least one semicolon on formal grounds.

Since the inputs and outputs have already been defined, the new FB can be added in the layout and can also be connected in parallel to the existing macro.

➲ Place the new FB appropriately in the layout.

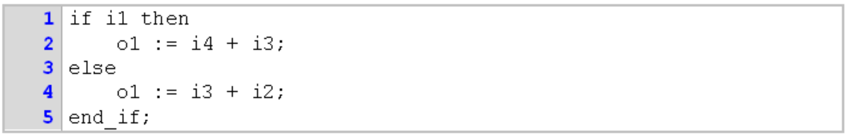
➲ Connect the new FB in parallel to the macro. In order to create the second result OTC, copy the existing one with <Ctrl+C>. Add it by pressing <Ctrl+V>. Assign a new name to the OTC, e. g. "Result_ST".



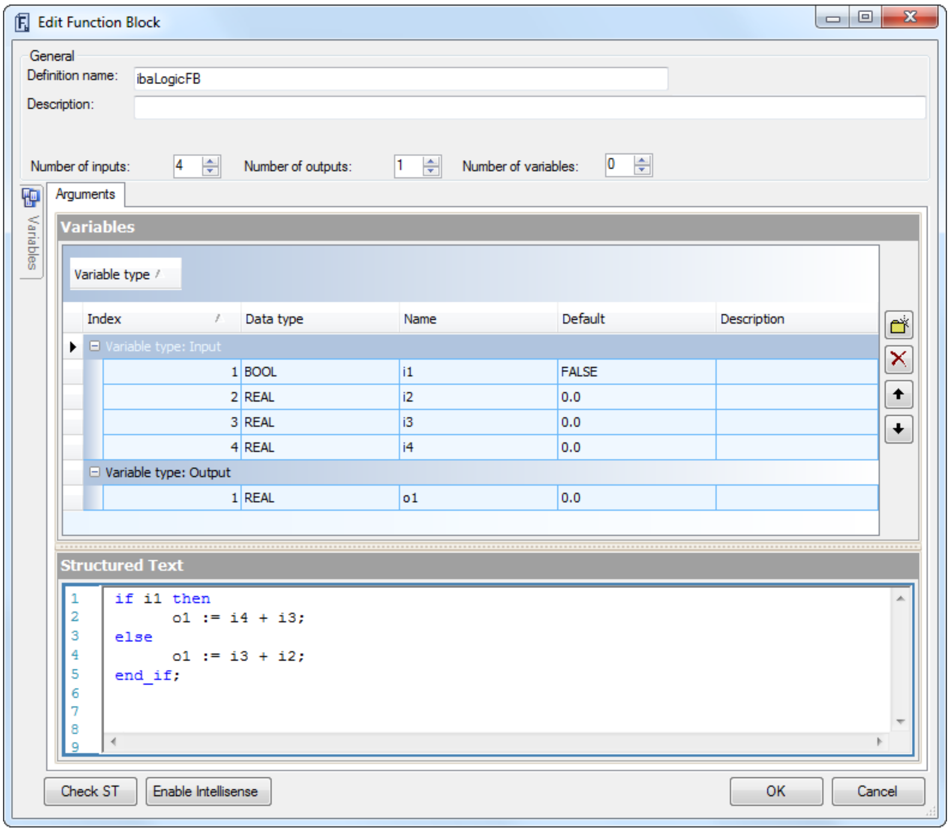
➲ You can see that the FB is not yet responding to its inputs. This still needs to be programmed for its task.

➲ Double click on the slider. The "Edit Function Block" window is displayed.
You can now carry out the programming using the customary "if", "then", "else" and "end_if" statements available in almost all high-level languages.

➲ Enter the following source code (as illustrated in the below screenshot) in the "Structured Text" field.



```
if i1 then
    o1 := i4 + i3;
else
    o1 := i3 + i2;
end_if;
```



Definition name: ibaLogicFB

Description:

Number of inputs: 4 Number of outputs: 1 Number of variables: 0

Variables

Variable type

Index	Data type	Name	Default	Description
Variable type: Input				
1	BOOL	i1	FALSE	
2	REAL	i2	0.0	
3	REAL	i3	0.0	
4	REAL	i4	0.0	
Variable type: Output				
1	REAL	o1	0.0	

Structured Text

```
if i1 then
        o1 := i4 + i3;
else
        o1 := i3 + i2;
end_if;
```

16.1.4.3 Result

The example begins with the inquiry of the switch status (if i1 = "TRUE" or shorter, if i1) and adds accordingly, the inputs i4 + i3 or i3 + i2.

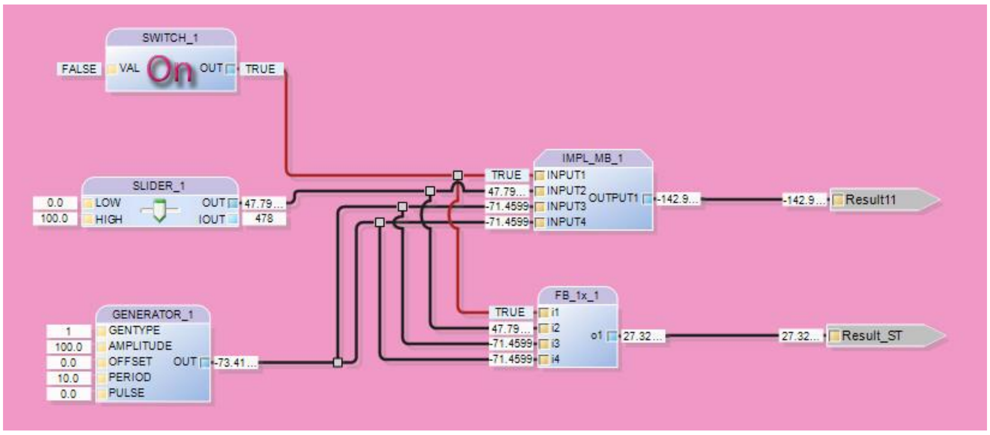


Figure 168: Sample circuit with addition inquiry

16.1.4.4 Remarks

Please also note that all variables in the function block are updated online!

➲ Drag the result of the ST function block with the same scale into the ibaPDA Express display.

The second red curve is congruent with the blue curve (Result from the macro). Only the last color (blue) of curve 2 is visible. You can see from the bar displayed on the right that there are two individual values.

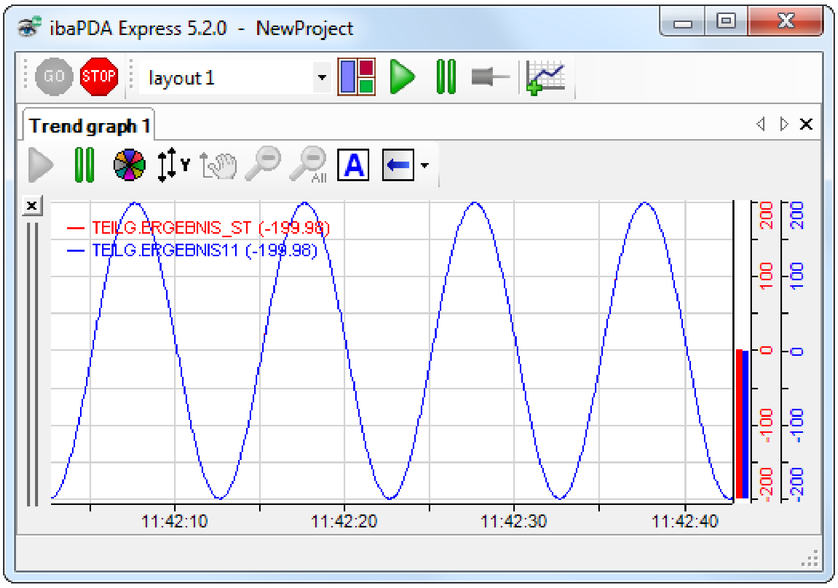


Figure 169: Congruency of the results

16.2 DAT_FILE_WRITE Sample Project

16.2.1 DAT_FILE_WRITE in “Unbuffered” mode

In the "Unbuffered" mode (explanation see "*Buffered Mode*, page 193“), one data sample is stored in each storage cycle.

Use this mode when the sampling cycle of external data to be saved matches that of the ibaLogic task cycle. Or when you wish to save data that is generated in ibaLogic itself.

Task: Store 8 analog values and 8 digital values from ibaLogic

Preset parameters

- Platform: Windows PC
- Task interval: 20 ms

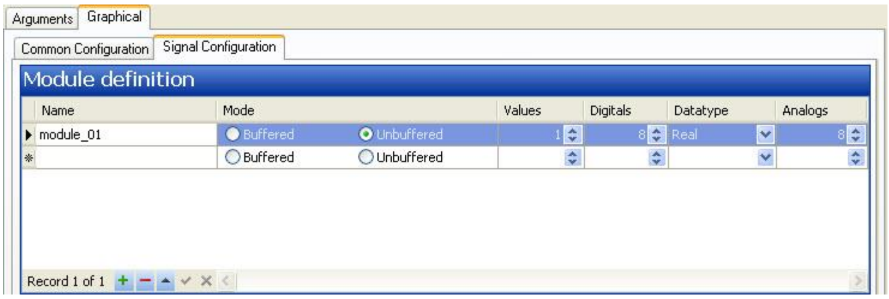
16.2.1.1 Step 1: Configure the DFW function block

➲ General Configuration

Asynchr. access:	Disabled
Storage cycle:	10 (not relevant)
Start time offset:	0
Save values:	**Enabled**
Write to file:	Disabled (is controlled externally)
Post-processing:	Disabled
Sign the file:	**Enabled**
Technostring:	Empty
File information:	Empty
Folder:	Choose a directory on a local drive using the browser button, e. g. "d:\dat\ibaLogic\"
File name:	Accept the default value specified.
Sampling time:	Specify the task interval time: „0.02" s

➲ Signal configuration

Name:	"module_01"
Mode:	„Unbuffered"
Values:	1 (not relevant)
Digital values:	8
Data type:	REAL
Analog values:	8



Arguments | Graphical

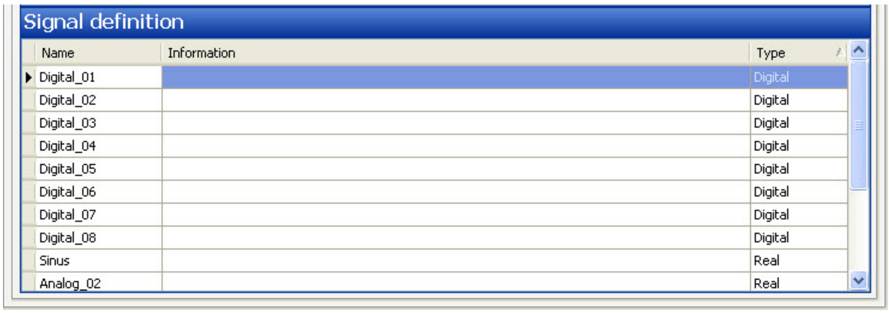
Common Configuration | Signal Configuration

Module definition

	Name	Mode		Values	Digitals	Datatype	Analogs
▶	module_01	○ Buffered	● Unbuffered	1	8	Real	8
*		○ Buffered	○ Unbuffered				

Record 1 of 1

- Click with the mouse in the signal definition area. In doing so, the module is created with 8 digital and 8 real values. The names are preset with "Digital_xx" and "Analog_xx".
  You can rename the standard signal names, e. g. to "Sine".

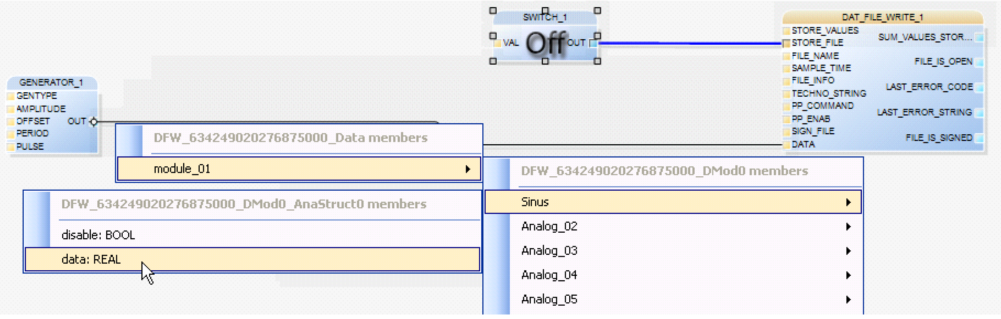


Signal definition

	Name	Information	Type
▶	Digital_01		Digital
	Digital_02		Digital
	Digital_03		Digital
	Digital_04		Digital
	Digital_05		Digital
	Digital_06		Digital
	Digital_07		Digital
	Digital_08		Digital
	Sinus		Real
	Analog_02		Real

16.2.1.2 Step 2: Connection of the DFW

- End the module editor.
- Join the connection "STORE_FILE" with the output of a "SWITCH" function block.
- Join the first measured signal, e. g. the output of the sinusoidal generator (Data type REAL) with the input connector "DATA".
  - A selection menu opens from which you can choose the module to which you wish to connect the measured signal.
  - When you move the mouse pointer to the desired module, another selection menu opens from which you can choose the desired signal (e. g. Sine).
  - Another selection menu is opened.
    Select the menu option "data: REAL".



DFW_634249020276875000_Data members
- module_01 ▶

DFW_634249020276875000_DMod0 members
- Sinus ▶
- Analog_02 ▶
- Analog_03 ▶
- Analog_04 ▶
- Analog_05 ▶

DFW_634249020276875000_DMod0_AnaStruct0 members
- disable: BOOL
- data: REAL

Figure 170: Measured signal assignment

Note

Do not let yourself get annoyed by the name of the data structure generated internally. You can ignore it (provided you do not try to reprogram the joiner in the ST, see below).

Result

Based on this, ibaLogic generates the joiner with which the element selected is addressed from the data structure generated internally.

For detailed description of joiners and splitters, please refer to section "*Converters, splitters, joiners*, page 157".

The following "joiners" are generated:

❑ "Signal Joiner"

❑ "Module Joiner"

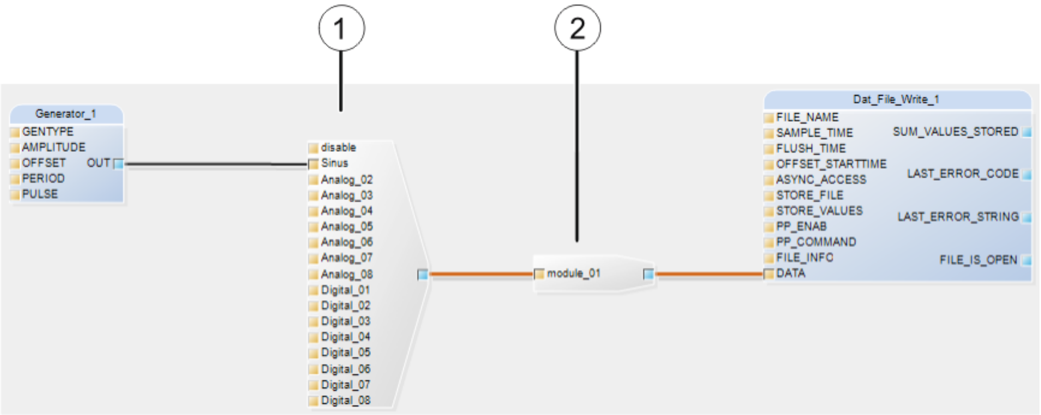


Figure 171: Joiner

1 Signal Joiner 2 Module Joiner

16.2.1.3 Step 3: Create other measure signals

You can connect the other signals directly with the signal joiner.

Note

Do not use the "Disable" inputs of the "Signal Property Joiner" in order to switch off the recording of individual signals for a period of time. This leads to incorrect recording.

Since the individual samples do not have a time stamp, the trend graph is always displayed contiguously. The gap desired is then at the end of the .dat file.

16.2.1.4 Step 4: Starting the recording

➲ Start the PMAC.

➲ Set the SWITCH in DAT_FILE_WRITE.STORE_FILE to "On".

Result

You recognize the ongoing recording from the incrementing value of the interface "DAT_FILE_WRITE_1.SUM_VALUES_STORED". Check the result by opening the .dat file generated with "ibaAnalyzer".

16.2.1.5 Alternative: Program joiner in ST

In order to keep the layout clear, you can, of course, also program the assignment of the measured signals at the DATA connector in an ST function block.

Note

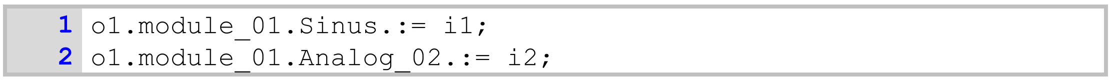
The option under "Tools - Options - General - System - Hide Generated Data Types" should **not** be selected.

Example, similar to the configuration given above:

- ➲ Go with the mouse to the DATA connector of the DFW module. The tooltip shows you the structure data type generated internally, e. g. "DFW_634094511415156250_Data". Write this down!
- ➲ Generate a function block having one or more input variables of type REAL and one output variable of type DATA connector of the DFW module.
  - Activate "Intellisense".
  - Begin with the assignment, and write "o1.".
  - As soon as you have entered the point, ibaLogic displays the structure elements (here "module_01"), since "o1" is a structure data type.
    Select the elements provided using the up/down cursor keys, accept them with "tab" and place a dot at the end.
  - Since even "module_01" is a structure, ibaLogic displays its structure elements, i.e. all digital and analog signals.

Result

With this, you have completed the first signal. You can enter other signals in the same manner. The result should then look like this:



```
o1.module_01.Sinus.:= i1;
o1.module_01.Analog_02.:= i2;
```

You can only connect the inputs with the matching measured signals and the output with the DATA connector of the DFW module.

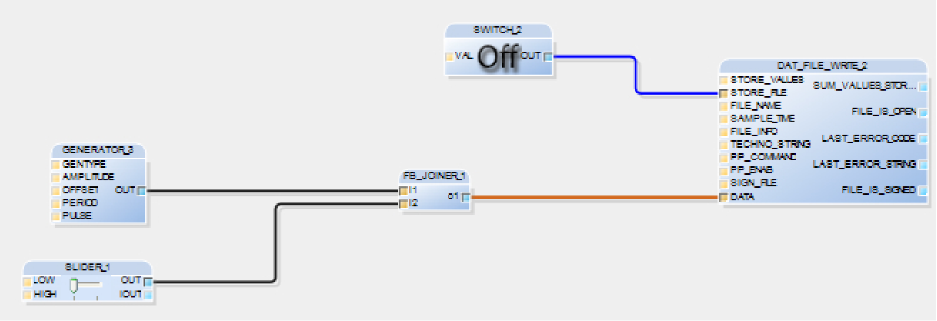


Figure 172: Sample circuit "Measured signal assignment"

Note

If you would like to modify the signal configuration of the DFW afterwards, you have to remove the connection at the DATA connector and, after the modification, assign the new data type to the output connector of the FB_JOINER. You may possibly have to modify the assignments in ST.

16.2.2 DAT_FILE_WRITE in "Buffered Mode"

In the "Buffered" mode (Explanation see "*Buffered Mode*, page 193") an array of n data samples is stored in each storage cycle. The signals to be stored must be available as arrays. As a consequence, certain parameters of the DFW module have a slightly changed meaning.

Use this mode when the sampling cycle of the data to be saved is faster than the fastest one in the ibaLogic task cycle.

Task: Store 8 analog values from ibaPADU8.

Preset parameters

- Sampling rate of ibaPADU8: 1 ms
- ibaPADU8 to ibaFOB-Link0, mode integer "Buffered"
- Interrupt time base: 1 ms
- Win XP platform
- Task interval: 20 ms

16.2.2.1 Step 1: Configuration of the buffered inputs

➲ General configuration:
The following hardware signals are available for the configuration of data buffering / data transfer (see description in section "*Buffered Mode*, page 193"):

- Output signals
  "…DataSize", "…Ratio", "…RequestBuffer"
- Input signals
  "…CurDataSize", "…FillCount"

➲ Create an event task and take BufferFilledCount as trigger

➲ Choose the parameters based on the following considerations:

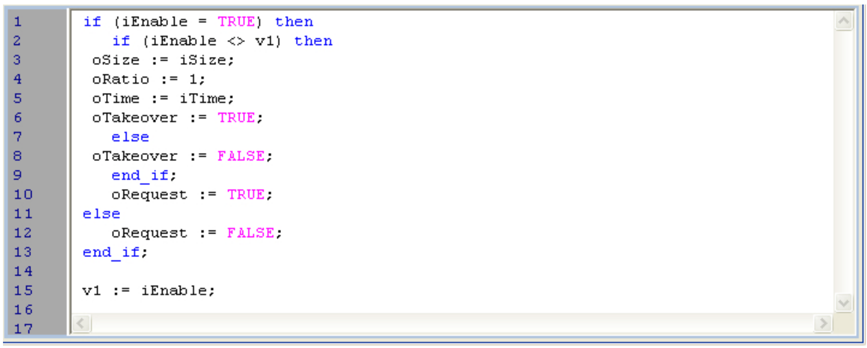
- The array depth in DFW must be greater than or equal to "DataSize".
- Sampling time (in DFW) must be the same as (array depth * sample time).
- The "DataSize" must be selected so that the following condition is met:

  Task interval ≤ Sampletime * DataSize / Ratio / 3

  (The factor "3" is used to ensure that no samples are lost, even if the task gets suspended.)

- For the example, choose DataSize = 100,
  That means: "Task interval ≤ 1ms * 100 / 1 / 3",
  i.e. the task interval must be less than 33 ms. Select 20 ms.

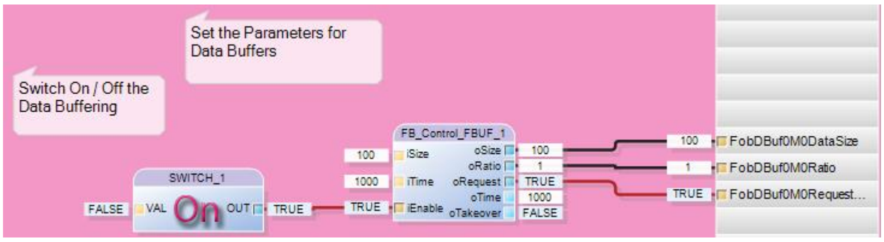
➲ Create a user-defined function block, with which you can preset the output signals for the buffered mode, for example



```
if (iEnable = TRUE) then
   if (iEnable <> v1) then
 oSize := iSize;
 oRatio := 1;
 oTime := iTime;
 oTakeover := TRUE;
   else
 oTakeover := FALSE;
   end_if;
   oRequest := TRUE;
else
   oRequest := FALSE;
end_if;

v1 := iEnable;
```

➲ Connect the function block with the output signals



(The connectors iTime, oTime and oTakeover are not required here.)

16.2.2.2 Step 2: Configure the DFW function block “General Configuration”

➲ Go offline.

➲ General configuration

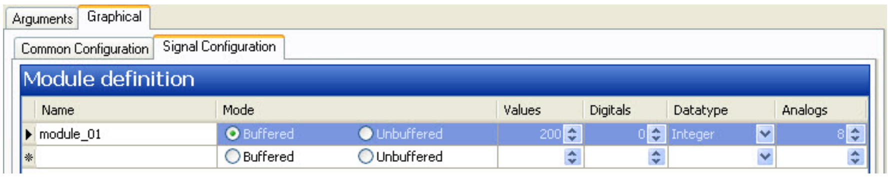
Asynchr. access:	Disabled
Storage cycle:	10 (not relevant)
Start time offset:	0
Save values:	**Enabled**
Write to file:	Disabled (is controlled externally)
Post-processing:	Disabled
Data compression:	**Enabled**
Technostring:	Empty
File information:	Empty
Folder:	Choose a directory on a local drive using the browser button, e. g. "d:\dat\ibaLogic\"
File name:	Accept the default value specified.

Sampling time: Specify the storage interval.
Storage interval = array depth * sampling time
Select an array depth of 200, based on which the storage time works out to 0.2 sec.

- Signal configuration

Name:	"module_01"
Mode:	"Buffered"
Values:	200
Digital values:	8
Data type:	Integer
Analog values:	8

- Specify the array size as 200 in the "Values" column. This yields a storage time of 200 ms (see above).



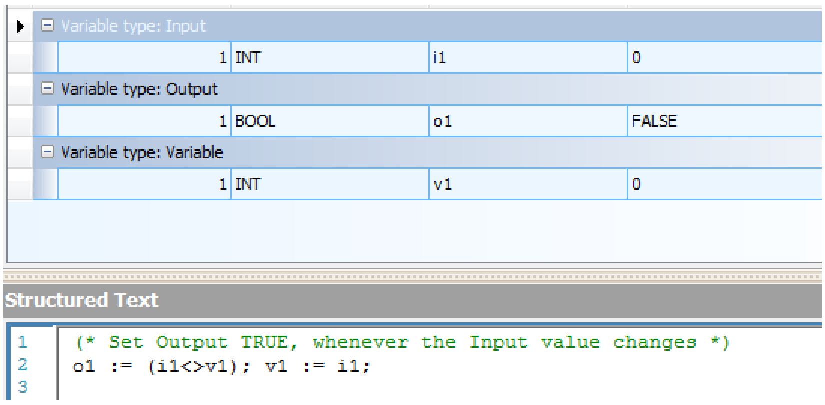
Module definition

Name	Mode	Values	Digitals	Datatype	Analogs
module_01	● Buffered ○ Unbuffered	200	0	Integer	8
	○ Buffered ○ Unbuffered				

- Go online.

16.2.2.3 Step 3: Apply the buffered input signals

- Drag a "COLLECT_ARRAY" function block from the "Type Conversion" function unit folder and drop it in the workspace window.
The function block is used to transfer the input array
(data type FOBFBUF_INT) into the array for the DFW.
- Create a user-defined function block (FB_DAAV) having the following properties to generate the "TAKEOVER" signal for the COLLECT_ARRAY:

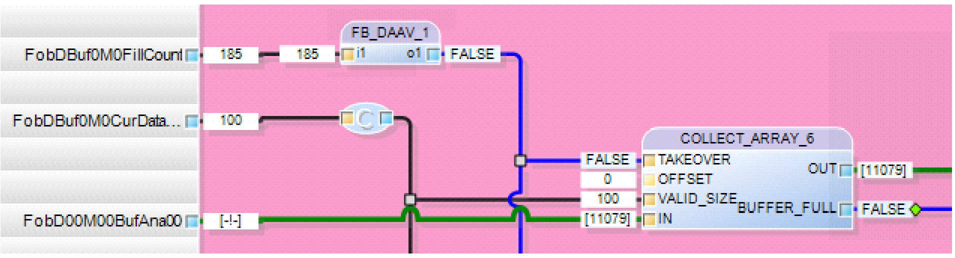


Variable type: Input			
1	INT	i1	0
Variable type: Output			
1	BOOL	o1	FALSE
Variable type: Variable			
1	INT	v1	0

Structured Text

```
(* Set Output TRUE, whenever the Input value changes *)
o1 := (i1<>v1); v1 := i1;
```

➲ Connect the function blocks as illustrated in the following screenshot.



Note

Instead of querying the "BufferFillCount" with an interval task, you can also use an event task with "BufferFillCount" as trigger.

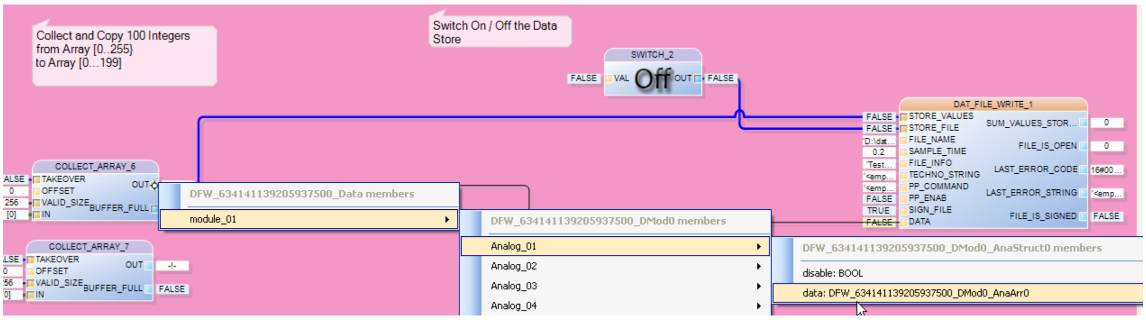
Please note that you need two tasks then. An interval task specifying the "BufferSize", "Ratio" and "Request" and an interval task triggered with "BufferFillCount" Triggered and writing the data in the DFW.

16.2.2.4 Step 4: Transfer the data to DAT_FILE_WRITE

➲ Create a SWITCH function block and connect its output with DAT_FILE_WRITE.STORE_FILES.

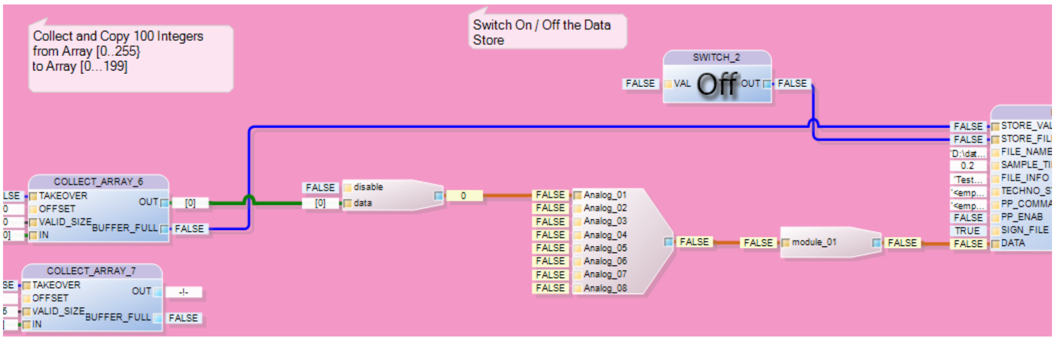
➲ Connect the COLLECT_ARRAY.BUFFER_FULL with DAT_FILE_WRITE.STORE_VALUES.

➲ Connect the output COLLECT_ARRAY.OUT with DAT_FILE_WRITE.DATA.
ibaLogic pops up the selection menus for the joiner.
Select here the "module_01 - Analog_01 - ... AnaArr0".



Result

The data is transferred to DAT_FILE_WRITE.

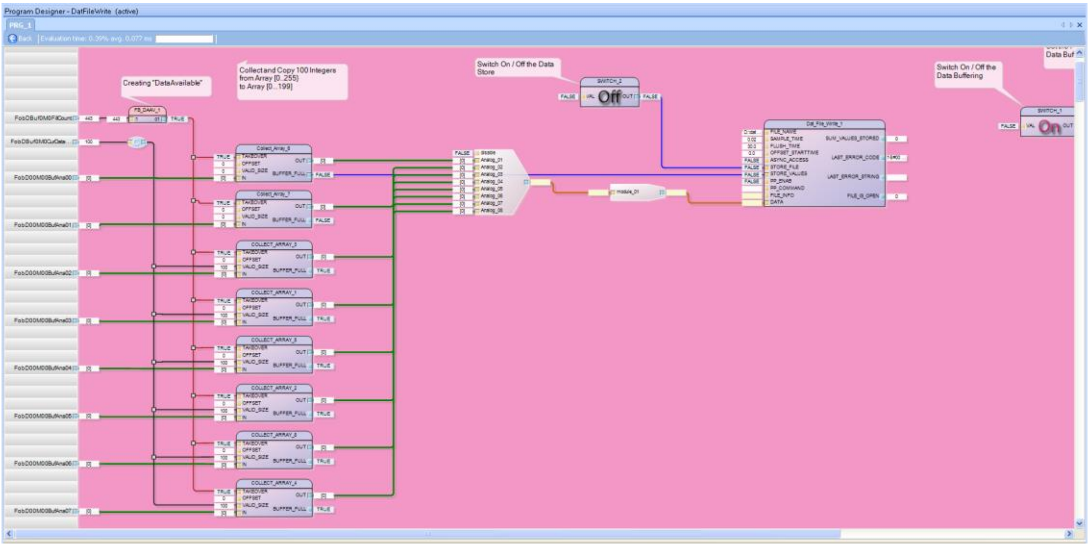


16.2.2.5 Step 5: Wiring (Connecting) the remaining inputs

- Copy one COLLECT_ARRAY function block for each analog input.
- Connect all COLLECT_ARRAY.TAKEOVER with the DataAvailabe signal (FB_DAAV.o1).
- Connect all COLLECT_ARRAY.VALID_SIZE with the output of the converter to " ... CurDataSize".
- Connect each COLLECT_ARRAY.IN with the respective buffered analog input " … BufAna*nn*"
- Connect each COLLECT_ARRAY.OUT with the respective connector "Analog_*nn*" of the joiner.

16.2.2.6 Step 6: Start the recording

- Set the SWITCH in DAT_FILE_WRITE.STORE_FILE to "On".



Result

You recognize the ongoing recording from the incrementing value of the interface "DAT_FILE_WRITE_1.SUM_VALUES_STORED". Check the result by opening the generated data file with "ibaAnalyzer".

Tips

- You can use the unbuffered inputs in parallel to the buffered inputs, for example, to visualize the analog signals using the ibaPDA Express.
- Check the data file:
  If the time scale does not match the actual recorded time, either the sampling time has not been configured correctly or the task interval, DataSize and array depth are not compatible with one another.

Documentation

The example given above is included in the DVD supplied.

16.2.3 DAT_FILE_WRITE in mix mode (buffered/unbuffered)

The DFW can decide for each module whether to record buffered or unbuffered values. This might make sense if you receive buffered values from the peripherals while also storing current values of the ibaLogic layout.

Example: You receive 50 buffered values, 100µs each. This would be a clock pulse of 50*100µs = 5ms. This is a high priority task so that no samples get lost and is triggered with “BufferFilledCount”.

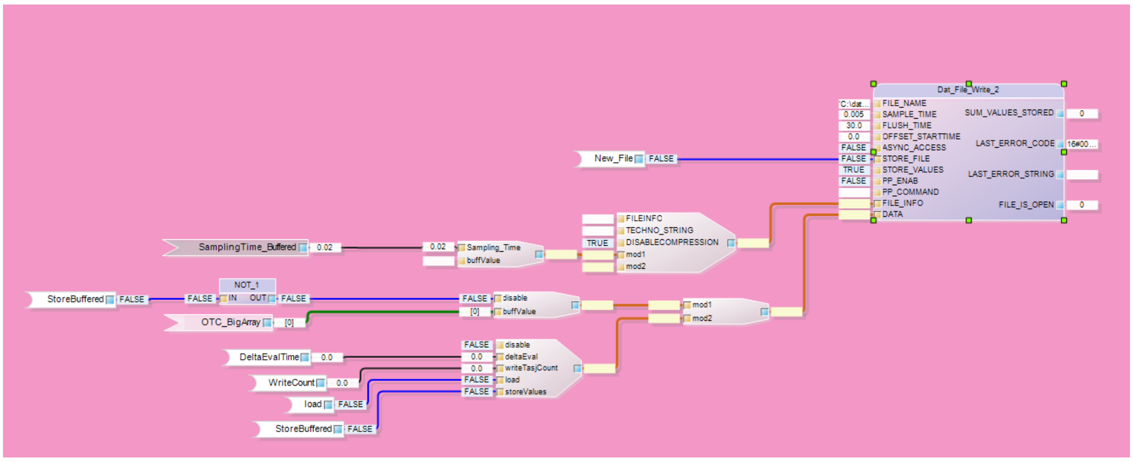
These values are to be stored in another low priority task in a DFW. This second task is to be executed every 20 ms.

Configuration:

In the high priority task, the 5ms buffers (50 values each) are stored in a buffer of 4*50 = 200 values. By doing so, the buffer is filled with 200 values every 20 ms. For decoupling, this data is copied to a second buffer and an OTC with “full buffer” is set. By doing so, the first buffer can continue to be written in the ring method and the second buffer has the data constantly available for 20 ms.

The second low priority task is also an event task that is also triggered with “BufferedFillCount”. Thus, also this task is started after the high priority task. It checks the OTC with “full buffer” of the first task and prompts writing the buffered values. In addition to that, however, further unbuffered values are written every 5 ms in case of the triggers in-between.

This shows the basic configuration:



The DFW is operated with a sampling_time of 5 ms. This corresponds to the event clock pulse. Module “mod2” is configured as unbuffered values.

Module “mod1” is to record the buffered values. For this purpose, the Sampling_Time for this module is to be 20 ms. This can be set for each module individually via the FILE_INFO structure. (see SamplingTime_Buffered).

However, it is only allowed to enter buffered data if this clock pulse/event also contains new ones. So, in the “empty events”, writing the buffered data with the “Disabled” input = TRUE must be prevented at the corresponding module.

Tip

If the timespan of the buffered data is too long or too short in the recorded measurement file (DAT file), the reason could be, that the time specified as Sampling_Time in FILE_INFO is wrong (SamplingTime_Buffered in the example). Or the DISABLE input of the buffered module is not set correctly.

Tip

Please note, that buffered data are entered with an offset which is equal to the timespan of the buffered data, because the timestamp is given in the DFW. The offset can be compensated with the OFFSET_STARTTIME input.

Tip

When buffered data is used, note that it is useful to synchronize the writing of data with the opening/closing of a DAT file. This means, if the file shall be closed, you should still wait for the next write command of the buffered data to ensure that the recording is complete. Otherwise data will get lost in this DAT file. If a new file is opened immediately, the next one would contain more data and there would be a time offset of this timespan.

FILE_INFO has a new input RENAME_AFTER_CLOSE in the structure.

The input exists when either a new DatFileWrite is created or the signals are changed in an existing one (number of signals, names, number of modules, ...). Renaming is done only when RENAME_AFTER_CLOSE is set to TRUE and the filename is different to the original one. If the new name already exists, the file keeps the old name and a warning message is sent at the error output. If a postprocessing should be executed with PP_COMMAND, the "@" sign can be used as wildcard for the filename.

17 Naming conventions

A name is a string of letters, digits and an underscore. The following rules are applicable:

- ❑ Capital and small letters are not relevant, for example, ABCD and abcd are identical.
- ❑ Names must begin with a letter or an underscore. A name cannot begin with a digit.
- ❑ Underscores are significant in the names, for example, A_BCD and AB_CD are different names (in contrast to this in number constants).
- ❑ Underscores at the end of a name are not permissible, e. g. ABCD_
- ❑ Multiple underscores are not allowed, e. g .AB__CD
- ❑ Keywords, e. g. for and if, are not allowed.

Peculiarities with ibaLogic:

Names having only one letter are not allowed.

Note

These rules, in general, are applicable to ibaLogic, and even beyond function units, e. g. for the names of OTCs, IPCs, input and output signals, function block names etc.

18 Data types

18.1 Standard data types

ibaLogic supports the following elementary data types:

Type	Range (Min.)	Range (Max.)	Explanation
BOOL	0 (FALSE)	1 (TRUE)	
BYTE	16#00	16#FF	8-bit
WORD	16#0000	16#FFFF	16-bit
DWORD	16#00000000	16#FFFFFFFF	32-bit word
SINT	-128	127	8-bit signed integer
USINT	0	255	8-bit unsigned integer
INT	-32768	32767	16-bit signed integer
UINT	0	65535	16-bit unsigned integer
DINT	-2147483648	2147483647	32-bit signed integer
UDINT	0	4294967295	32-bit unsigned integer
REAL	1.175494 e-38	3.4028234 e+38	Floating point, single accuracy, 32 bit
LREAL	2.225073858 … e-308	1.797693134862 … e+308	Floating point, double accuracy, 64 bit
TIME	-10675199d2h48m5s478ms	10675199d2h48m5s478ms	Time, mapped internally as 64-bit integer with 1 ms resolution per increment
STRING	'empty'	1024 characters	String with number of characters

18.2 Derived data types

Type	Explanation
DIRECT_DERIVED	Elementary data types with a fixed value (Constants)
SUBRANGE	Integer data types with a limited range of values
STRING_DERIVED	String having a fixed value and length, max. 8912 characters

18.3 Generic data types

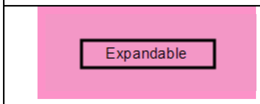
Type	Explanation
ENUM	Enumerator type, names are defined instead of values.
ARRAY	Structure, consisting of any sequence of **one** of the a.m. data types (with the exception of the string that already represents an array); maximum four-dimensional Maximum number of elements: 262149
STRUCT	Structure, consisting of any sequence of the data types mentioned above Maximum number of elements: 1024

19 Standard Function Units

This section contains a tabular overview of all functions and function blocks that are available in ibaLogic-V5.

19.1 Table interpretation

This section provides tips and instructions on interpreting the tabular overview.

Column	Explanation
Input data type	The input data type columns list the data types permissible for each connector. There are function units for whose connectors the data types are not defined right from the beginning, but whose data type is defined only when a connection is drawn to another connector.
Function unit design	
Green 🟩	Functions or function blocks have been defined in conformity with the IEC 61131-3 standard.
Yellow 🟨	Functions or function blocks have been defined by iba AG.
	This function unit is expandable, i. e. you can change the number of inputs. Open the function unit by double clicking on it and modify the "Number of inputs".
Output data type	The output data type columns list the data types permissible for each connector. There are function units for whose connectors the data types are not defined right from the beginning, but whose data type is defined only when a connection is drawn to another connector.
Explanation, example, ST syntax	There is a note provided for each function regarding whether and how you can call up the function within ST. You can also clone functions as multiple lines of ST code. This, however, is not executed.

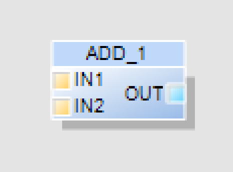
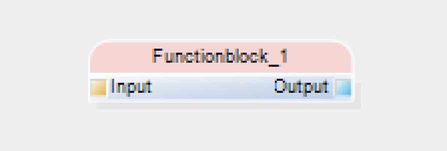
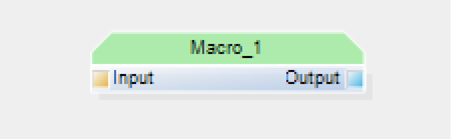
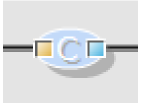
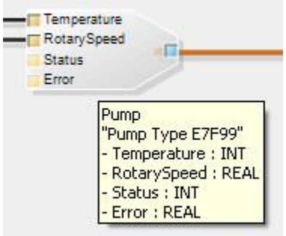
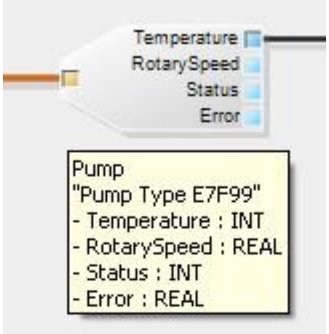
19.2 Data types

The data type "ANY" with the following variants is displayed for the "untyped" connectors:

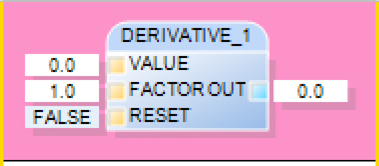
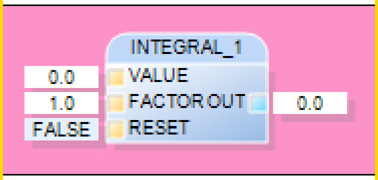
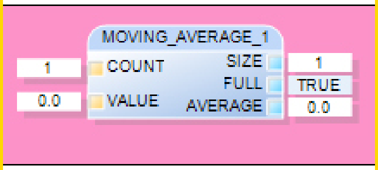
Data type "ANY" with "untyped" connectors	Explanation
Any_Int	All integer types (SINT, INT, DINT, USINT, UINT and UDINT)
Any_Real	All real types (REAL, LREAL)
Any_Num	All numerical data types (all integers and real values)
Any_Magnitude	All numerical data types and the TIME type
Any_Bit	All bit-oriented types (BOOL, BYTE, WORD and DWORD)
Any_String	STRING data type
Any_Elementary	All elementary types (Integers, real values, TIME and STRING)
Any_Derived	All elementary data types, arrays and structures
Any	Any data type

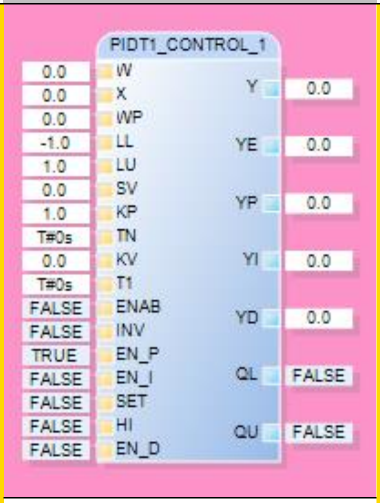
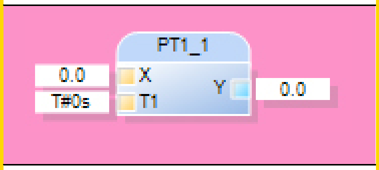
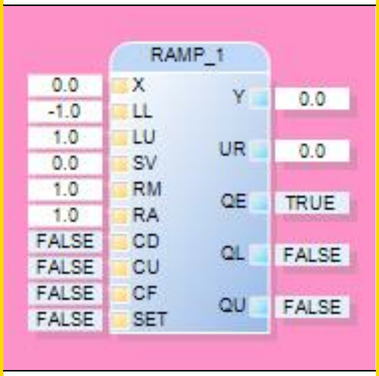
19.3 Function Block type with function diagram display

Functions, function blocks and macro blocks are displayed in the design area as follows:

Function Block type with function diagram display	Explanation
Function 	You can recognize a function by its corners.
Function block 	You can recognize a function block by the rounded corners.
Macro block 	You can recognize a macro block by the flattened corners.
Automatic type converter 	You can recognize a type converter by the letter "C" in the icon.
Automatically generated structure joiner  Pump "Pump Type E7F99" - Temperature : INT - RotarySpeed : REAL - Status : INT - Error : REAL	Automatically generated structure joiner This is generated automatically as soon as you try to connect a single parameter with a structure.
Automatically generated structure splitter  Pump "Pump Type E7F99" - Temperature : INT - RotarySpeed : REAL - Status : INT - Error : REAL	Automatically generated structure splitter This is generated automatically as soon as you try to connect a single parameter with a structure.

19.4 Analytical Functions Blocks

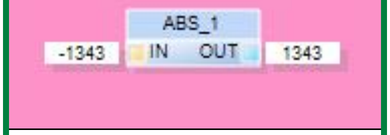
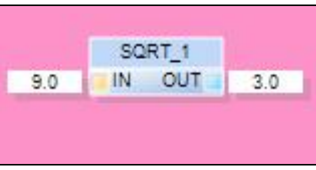
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Real Real Bool	 DERIVATIVE_1 0.0 1.0 FALSE VALUE FACTOR RESET OUT 0.0	Real	**DERIVATIVE:** Derivative of a value based on the time The output value "OUT" is the derivative of the input "VALUE" on the time dimension, multiplied by a factor "FACTOR". The output is reset with the input "RESET" = "TRUE". Implementation: $OUT:=(VALUE_n-VALUE_{n-1})*FACTOR/\Delta t$ Δt = task interval **ST:** cannot be called up
Real Real Bool	 INTEGRAL_1 0.0 1.0 FALSE VALUE FACTOR RESET OUT 0.0	Real	**INTEGRAL:** Integration of the value over time The output value "OUT" is the integral of the input "VALUE" over the time dimension, multiplied by a factor "FACTOR". The output is reset with the input "RESET" = "TRUE". Implementation: $OUT_n:=OUT_{n-1}+(VALUE_n-VALUE_{n-1})*FACTOR*\Delta t$ Δt = task interval **ST:** cannot be called up
Dint Real	 MOVING_AVERAGE_1 1 0.0 COUNT VALUE SIZE FULL AVERAGE 1 TRUE 0.0	Dint Bool Real	**MOVING_AVERAGE:** Moving average value The input value "COUNT" defines a number of values (= samples) that are considered for average calculation of the value "VALUE". The output value "SIZE", indicates the number of values used for the calculation of the average value. The output "FULL", is "TRUE" if the number of values (samples) specified has been reached. The output value "AVERAGE" yields the cumulative average value. AVERAGE:= (sum of the last SIZE values) / SIZE. The average calculation of the value is performed continuously. The number of samples can be modified whenever required. **ST:** cannot be called up

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Lreal Lreal Lreal Lreal Lreal Lreal Lreal Time Lreal Time Bool Bool Bool Bool Bool Bool Bool	 PIDT1_CONTROL_1 0.0 W 0.0 X 0.0 WP -1.0 LL 1.0 LU 0.0 SV 1.0 KP T#0s TN 0.0 KV T#0s T1 FALSE ENAB FALSE INV TRUE EN_P FALSE EN_I FALSE SET FALSE HI FALSE EN_D Y 0.0 YE 0.0 YP 0.0 YI 0.0 YD 0.0 QL FALSE QU FALSE	Lreal Lreal Lreal Lreal Lreal Bool Bool	**PIDT1_CONTROL:** PIDT1 control function block Universal PIDT1 controller that can be switched to operating modes as a P, I, PI or PIDT1 controller. You will find a detailed description in "*PIDT1_CONTROL*, page 109". **ST:** cannot be called up
Lreal Time		Lreal	**PT1:** Time delay element of the 1st order The input variable X is delayed dynamically by the smoothing time constant, T1, and fed to the output Y. Implementation: ti: = time_to_lreal(T1) / time_to_lreal(*EvalDeltaTime*[8]); Y: = 1.0 / (1.0 + t1) * (X + ti * Yold); Yold: = Y; **ST:** cannot be called up
Lreal Lreal Lreal Lreal Lreal Lreal Bool Bool Bool Bool	 RAMP_1 0.0 X -1.0 LL 1.0 LU 0.0 SV 1.0 RM 1.0 RA FALSE CD FALSE CU FALSE CF FALSE SET Y 0.0 UR 0.0 QE TRUE QL FALSE QU FALSE	Lreal Lreal Bool Bool Bool	**RAMP:** Ramp function block Function block with two different ramps for the manual and automatic modes. You will find a detailed description in "*RAMP*, page 116". **ST:** cannot be called up

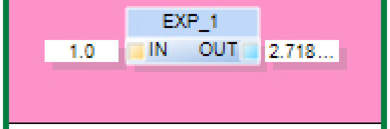
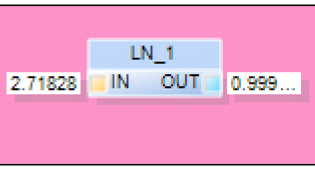
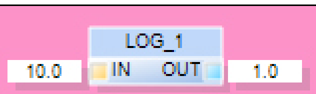
[8] EvaldeltaTime is the time between two processing cycles (calculated internally).

19.5 Arithmetical Functions

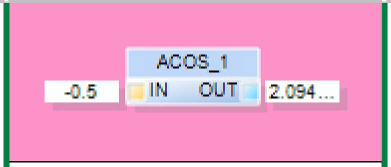
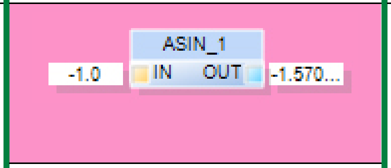
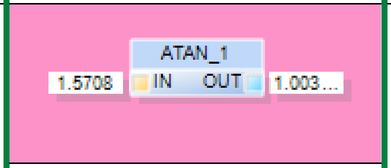
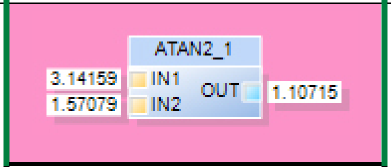
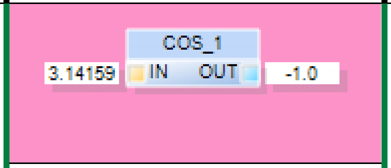
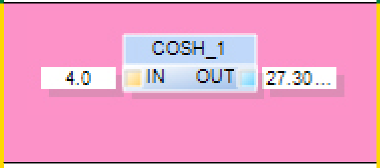
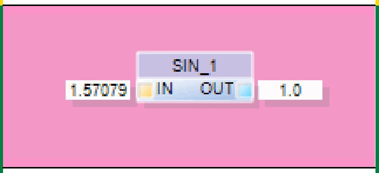
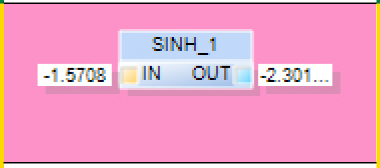
19.5.1 General

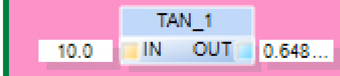
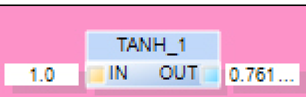
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Num		Any_Num	**ABS:** Absolute value Example: +1343 = abs(-1343); **ST:** `OUT := abs(IN);`
Any_Real		Any_Real	**SQRT:** Square root Example: +3.0 = sqrt(9.0); **ST:** `OUT:= sqrt(IN);`

19.5.2 Logarithmic

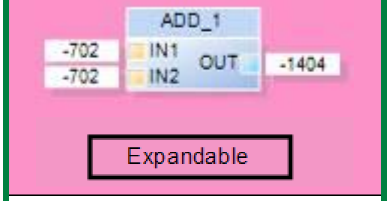
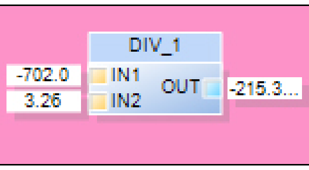
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Real		Any_Real	**EXP:** Natural exponent to the base e Result: = e^{arg}; Examples: 2.71828 = exp(1.0); 0.13533 = exp(-2.0); **ST:** `OUT:= exp(IN);`
Any_Real		Any_Real	**LN:** Natural logarithm Example: +1.0 = ln(2.71828); **ST:** `OUT:= ln(IN);`
Any_Real		Any_Real	**LOG:** Logarithm to the base 10 Example: +1.0 = log(10.0); **ST:** `OUT:= log(IN);`

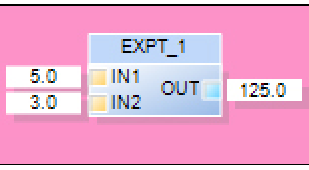
19.5.3 Trigonometric

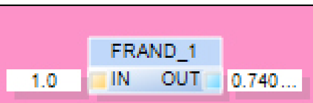
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Real		Any_Real	**ACOS:** Arc cosine Example: 1.57079 = acos(0.0); **ST:** `OUT:= acos(IN);`
Any_Real		Any_Real	**ASIN:** Arc sine Example: -1.57079 = asin(-1.0); **ST:** `OUT:= asin(IN);`
Any_Real		Any_Real	**ATAN:** Arc tan Example: 1.0000 = atan(π/2.0); **ST:** `OUT:= atan(IN);`
Any_Real Any_Real	 ATAN2_1 3.14159 1.57079 IN1 IN2 OUT 1.10715	Any_Real	**ATAN2:** Arc tan Example: 1.1071 = atan2_real(π,π/2.0); **ST:** `OUT:= atan2(IN1, IN2);`
Any_Real		Any_Real	**COS:** Cosine Example: -1.0000 = cos(π); **ST:** `OUT:= cos(IN);`
Any_Real		Any_Real	**COSH:** Hyperbolic cosine Example: +27.3082 = cosh_real(4.0); **ST:** `OUT:= cosh(IN);`
Any_Real		Any_Real	**SIN:** Sine Example: 1.0 = sin(π/2); **ST:** `OUT:=sin(IN);`
Any_Real		Any_Real	**SINH:** Hyperbolic sine Example: -2.3013 = sinh_real(-π/2.0); **ST:** `OUT:=sinh(IN);`

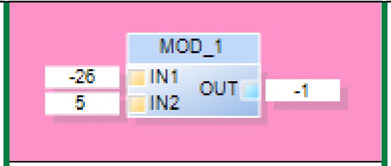
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Real		Any_Real	**TAN:** Tangent Example: 0.648 = tan(10.0); **ST:** `OUT:=tan(IN);`
Any_Real		Any_Real	**TANH:** Hyperbolic tangent Example: 0.76159 = tanh_real(1.0); **ST:** `OUT:=tanh(IN);`

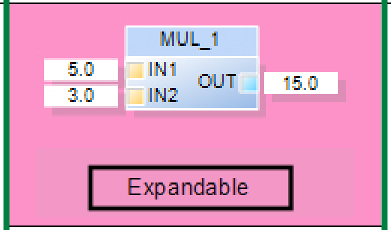
19.5.4 Basic functions

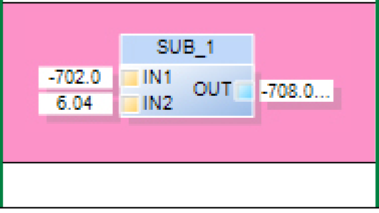
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_ Magnitude Any_ Magnitude	 ADD_1 -702 -702 IN1 IN2 OUT -1404 Expandable	Any_ Magnitude	**ADD:** Addition Example: -1404 = -702 + -702; **ST:** Use operator: `OUT:= IN1 + IN2 + ... + INn;`
Any_Num Any_Num		Any_Num	**DIV:** Division Example: -215.3 = -702.0 / 3.26; **ST:** Use operator: `OUT:= IN1 / IN2;` **Attention:** If the divisor IN2 = 0, the result is set to infinite and the output to "INVALID".

Any_Real
Any_Num
		Any_Real	**EXPT:**General exponent to the base (IN2)
			Result: = arg1^{arg2};
			Examples:
			125.0 = expt(5.0, 3.0);
			4.0 = 16.0 ** 0.5;
			ST:
			`OUT := expt(IN1, IN2);`
			or
			`OUT := IN1 ** IN2;`

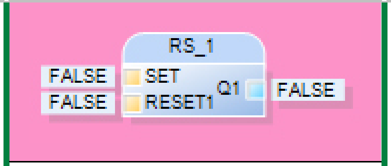
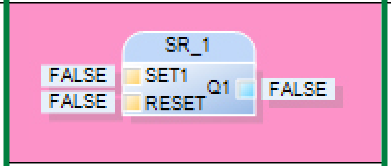
Any_Real		Any_Real	**FRAND:** Random number in the range {0 ... arg}
			Example:
			+0.07116 = frand_real(1.00);
			+2.92457 = frand_lreal(6.00);
			ST: Different calls for the REAL and LREAL data types
			`OUT:= frand(IN);`

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Int Any_Int		Any_Int	**MOD:** Division remainder (Modulo) Example: -1 = -26 mod 5; **ST:** Use operator: `OUT:= IN1 mod IN2;`

Any_Num
Any_Num
	 MUL_1
	5.0
	3.0
	IN1
	IN2
	OUT
	15.0
	Expandable
		Any_Num	**MUL:** Multiplication
			Example:
			15.0 = 5.0 * 3.0;
			4 = 2 * 2;
			ST: Use operator:
			`OUT:= IN1 * IN2 * … * INn;`

Any_ Magnitude Any_ Magnitude		Any_Magnitude	**SUB:** Subtraction Example: -708.04 = -702 – 6.04; **ST:** Use operator: `OUT:= IN1 – IN2;`

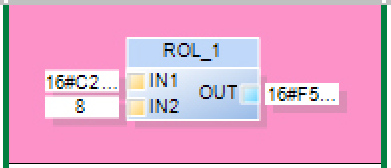
19.6 Bistable

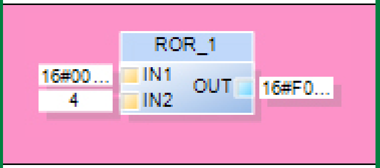
<table>
<tr><th>Input data type</th><th>Function unit design</th><th>Output data type</th><th>Explanation, example, ST syntax</th></tr>
<tr><td>Bool
Bool</td><td> </td><td>Bool</td><td><b>RS:</b> RS flip-flop
(static binary value store)
R-dominant
Truth table:
<table>
<tr><th colspan="2">Input values</th><th>Output</th></tr>
<tr><td>SET</td><td>RESET1</td><td>Q1</td></tr>
<tr><td>0</td><td>0</td><td>Q1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
<b>ST:</b> cannot be called up</td></tr>
<tr><td>Bool
Bool</td><td> </td><td>Bool</td><td><b>SR:</b> SR flip-flop
(static binary value store)
S-dominant
Truth table:
<table>
<tr><th colspan="2">Input values</th><th>Output</th></tr>
<tr><td>SET</td><td>RESET1</td><td>Q1</td></tr>
<tr><td>0</td><td>0</td><td>Q1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
<b>ST:</b> cannot be called up</td></tr>
</table>

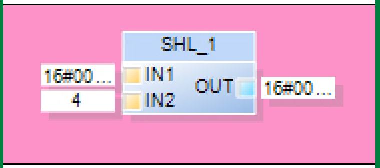
19.7 Bit String

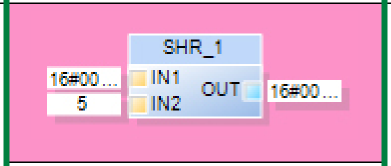
19.7.1 Bit shift

Input data type	Function unit design	Output data type	Explanation, example, ST syntax

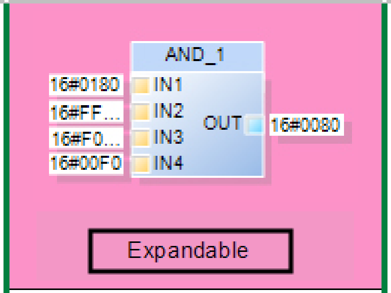
Any_Bit
Uint
		Any_Bit	**ROL:** Rotate arg1 left by arg2 Bits
			Examples:
			16#F50000C2 =rol(16#C2F50000,8);
			16#45678123 =rol(16#12345678,12);
			ST:
			`OUT := rol(IN1,IN2);`

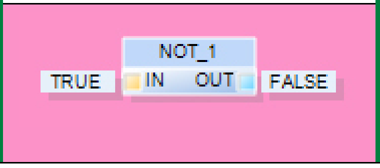
Any_Bit
Uint
		Any_Bit	**ROR:** Rotate arg1 right by arg2 bits
			Examples:
			16#F00000C2 = ror(16#C2F,4);
			16#F500000C = ror(16#CF5,8);
			ST:
			`OUT := ror(IN1,IN2);`

Any_Bit Uint		Any_Bit	**SHL:** Shift IN1 left by IN2 bits and fill up zeros on the right Example: 16#0D90 = shl(16#00D9,4); **ST:** `OUT := shl(IN1,IN2);`

Any_Bit
Uint
		Any_Bit	**SHR:** Shift IN1 right by IN2 bits and fill up zeroes on the left
			Examples:
			16#000C = shr(16#0180,5);
			16#00D9 = shr(16#0D90,4);
			ST:
			`OUT := shr(IN1,IN2);`

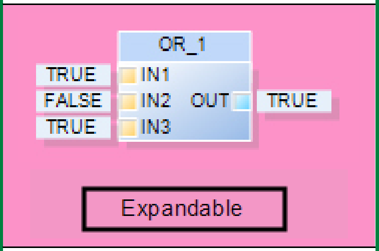
19.7.2 Bitwise_Boolean

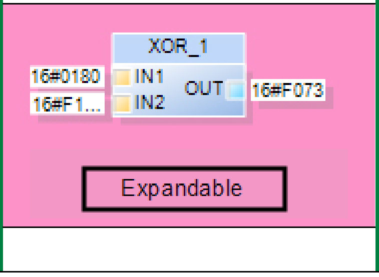
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Bit Any_Bit Any_Bit Any_Bit	 AND_1 16#0180 16#FF… 16#F0… 16#00F0 IN1 IN2 IN3 IN4 OUT 16#0080 Expandable	Any_Bit	**AND:** Logical AND combination Example: 16#80= and(16#0180, 16#FFF0, 16#F0F0, 16#00F0); **ST:** Use operator: `OUT := IN1 AND IN2 … AND INn;`

Any_Bit		Any_Bit	**NOT:** Logical NOT function
			Examples:
			FALSE = not(TRUE);
			16#FE7F = not(16#0180);

Input data type	Function unit design	Output data type	Explanation, example, ST syntax

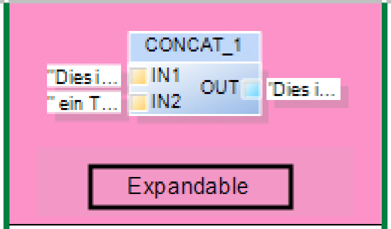
			ST:
			`OUT := not IN;`
			or
			`OUT := not(IN);`

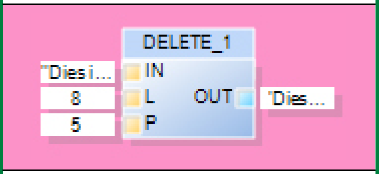
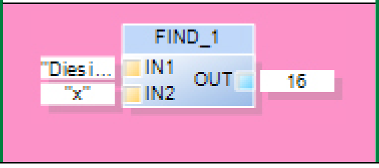
Any_Bit
Any_Bit
Any_Bit
	 OR_1
	TRUE
	FALSE
	TRUE
	IN1
	IN2
	IN3
	OUT
	TRUE
	Expandable
		Any_Bit	**OR:** Logical OR combination
			Examples:
			1 = or(1, 0, 1);
			16#F3 = or(16#F0,16#03);
			ST: Use operator:
			`OUT:= IN1 or IN2 or … INn;`

Any_Bit
Any_Bit
	 XOR_1
	16#0180
	16#F1...
	IN1
	IN2
	OUT
	16#F073
	Expandable
		Any_Bit	**XOR:** Logical XOR combination.
			Examples:
			FALSE = xor(TRUE,TRUE);
			16#F073 = xor(16#0180,16#F1F3);
			ST: Use operator:
			`OUT:= IN1 xor IN2 xor … INn;`

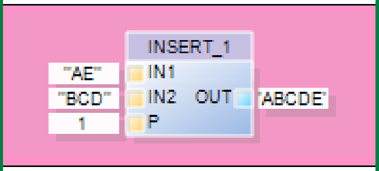
19.8 Character String

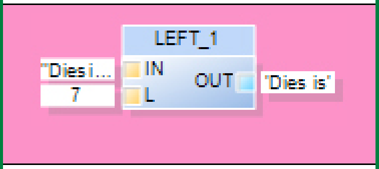
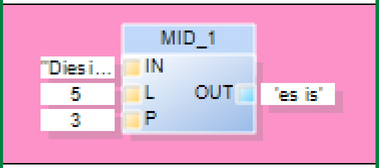
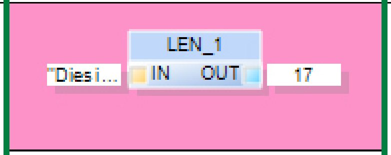
Input data type	Function unit design	Output data type	Explanation, example, ST syntax

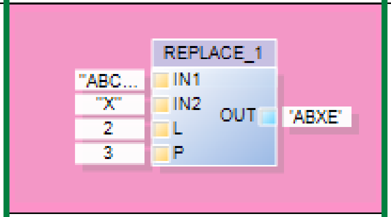
Any_String
Any_String
	 CONCAT_1
	'Dies i...
	' ein T...
	IN1
	IN2
	OUT
	'Dies i...
	Expandable
		Any_String	**CONCAT:** Combining (joining) sub-strings
			Examples:
			'This is a text'= concat('This is', ' a text')
			ST:
			`OUT:=concat (IN1, IN2,…,INn);`
			or
			`OUT:=IN1+IN2+…+INn);`

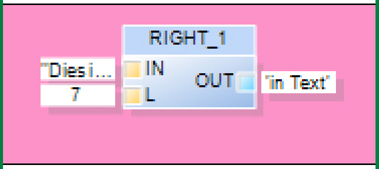
Any_String Uint Uint	 DELETE_1 'Dies i... 8 5 IN L P OUT 'Dies...	Any_String	**DELETE:** Delete L characters of a string from (including) position P. The first character has the position 1. Examples: 'This text'= delete('This is a text',8,5); 'DE' = delete('ABCDE',3,1); **ST:** `OUT:=delete(IN, L,P);`
Any_String Any_String		Int	**FIND:** Search for the first match of character IN2 in string IN1. If the character is not found, the result = 0.

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
			Examples: 16 = find('This is a text', 'x'); 1 = find('This is a text', 'D'); **ST:** `OUT:=find(IN1, IN2);`

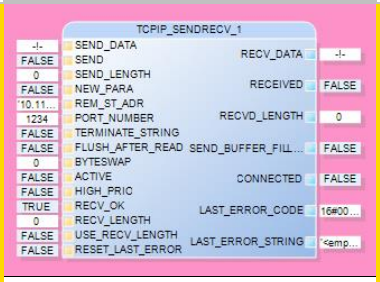
Any_String
Any_String
Uint
	 INSERT_1
	"AE"
	"BCD"
	1
	IN1
	IN2
	P
	OUT
	'ABCDE'
		Any_String	**INSERT:**
			Inserting string IN2 in string IN1 after the position P. If P=0, IN2 is inserted at the beginning.
			Examples:
			'ABCDE' = insert('AE','BCD', 1);
			'xABC' = insert('ABC','x',0);
			ST:
			`OUT:=insert(IN1, IN2,P);`

Any_String Uint		Any_String	**LEFT:** The left part of a string IN having length L Example: 'This is'= left('This is a text',7); **ST:** `OUT:=left(IN,L);`
Any_String Uint Uint	 MID_1 "Dies i... 5 3 IN L P OUT 'es is'	Any_String	**MID:** Section of a string IN having length L from and including position P. Example: 'is is' = mid('This is a text',5,3); **ST:** `OUT:=mid(IN,L,P);`
Any_String		Int	**LEN:** length of a string (without termination characters) Examples: 17= len('This is a text'); 4= len('text'); **ST:** `OUT:=len(IN);`

Any_String
Any_String
Uint
Uint
	 REPLACE_1
	"ABC...
	"X"
	2
	3
	IN1
	IN2
	L
	P
	OUT
	'ABXE'
		Any_String	**REPLACE:**
			Replace L characters of the string IN by IN2 starting from and including position P
			Example:
			'ABXE' =
			replace('ABCDE','X', 2,3);
			ST:
			`OUT:=`
			`replace(IN1, IN2, L, P);`

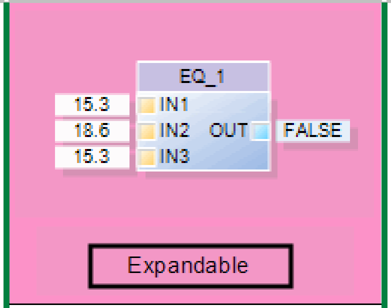
Any_String Uint		Any_String	**RIGHT:** The right part of a string having length L Example: 'a text'= right('This is a text',6); **ST:** `OUT:= right(IN, L);`

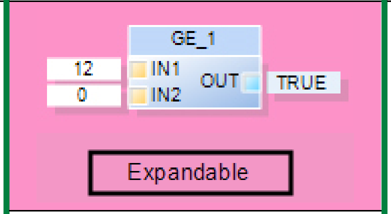
19.9 Communication

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any Bool Udint Bool String Udint Bool Bool Int Bool Bool Bool Udint Bool Bool	 TCPIP_SENDRECV_1 -!- FALSE 0 FALSE '10.11... 1234 FALSE FALSE 0 FALSE FALSE TRUE 0 FALSE FALSE SEND_DATA SEND SEND_LENGTH NEW_PARA REM_ST_ADR PORT_NUMBER TERMINATE_STRING FLUSH_AFTER_READ BYTESWAP ACTIVE HIGH_PRIO RECV_OK RECV_LENGTH USE_RECV_LENGTH RESET_LAST_ERROR RECV_DATA RECEIVED RECVD_LENGTH SEND_BUFFER_FILL... CONNECTED LAST_ERROR_CODE LAST_ERROR_STRING -!- FALSE 0 FALSE FALSE 16#00... '<emp...	Any Bool Udint Bool Bool Dword String	**TCPIP_SendRecv:** Transmission and reception of data via TCP/IP. The data here is raw data that is sent via TCP/IP. In this manner, all native TCP/IP protocols can be re-created. You will find a detailed description in "*TCPIP_SENDRECV*, page 107". **ST:** cannot be called up within ST

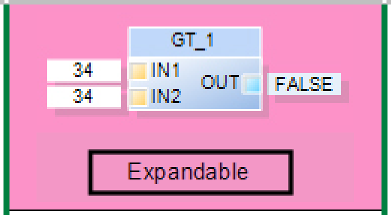
19.10 Comparison

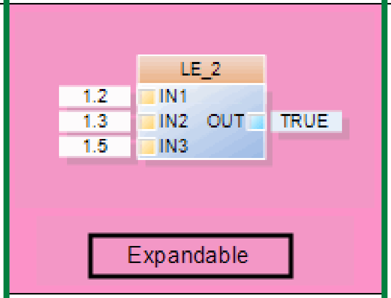
Input data type	Function unit design	Output data type	Explanation, example, ST syntax

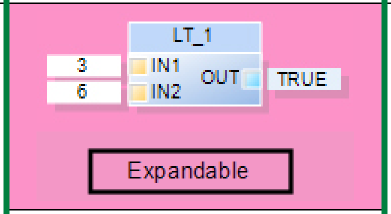
Any_
Elementary
Any_
Elementary
	 EQ_1
	15.3
	18.6
	15.3
	IN1
	IN2
	IN3
	OUT
	FALSE
	Expandable
		Bool	**EQ:**
			Comparison of equality
			The result is TRUE if all arguments are identical.
			Example:
			FALSE = (15.3 = 18.6 = 15.3)
			ST: Use operator:
			`OUT := IN1 = IN2 ;`
			Only two arguments are allowed. Implementation with logical combination of multiple comparisons:
			`OUT :=`
			`(IN1 = IN2) AND (IN1 = IN3) ;`

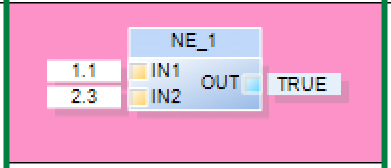
Any_
Elementary
Any_
Elementary
		Bool	**GE:**
			Comparison for greater than or equal to
			The result is TRUE if IN1 is greater than or equal to all other arguments.
			Example:
			TRUE = 12 >= 0;
			ST: Use operator:
			`OUT:= IN1 >= IN2;`
			Only two arguments are allowed. Implementation with logical combination of multiple comparisons:
			`OUT :=`
			`(IN1 >= IN2) AND (IN1 >= IN3);`

Input data type	Function unit design	Output data type	Explanation, example, ST syntax

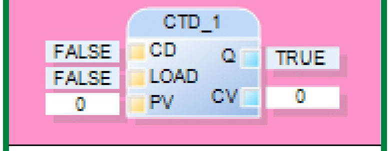
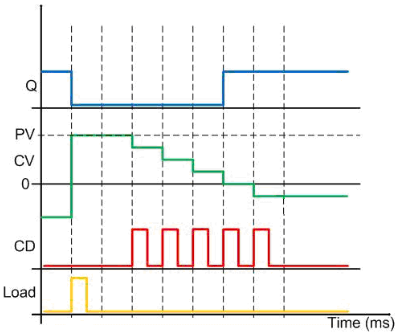
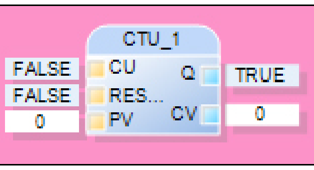
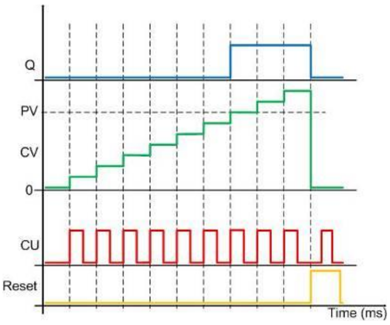
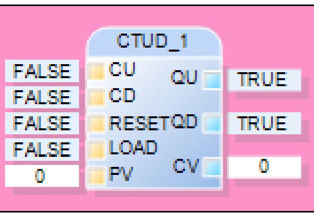
Any_ Elementary
Any_ Elementary
	 GT_1
	34
	34
	IN1
	IN2
	OUT
	FALSE
	Expandable
		Bool	**GT:** Comparison for greater than
			The result is TRUE if IN1 is greater than all other arguments.
			Example:
			FALSE = 34 >34;
			ST: Use operator:
			`OUT:= IN1 > IN2;`
			Only two arguments are allowed. Implementation with logical combination of multiple comparisons:
			`OUT := (IN1 > IN2) AND (IN1 > IN3);`

Any_ Elementary
Any_ Elementary
	 LE_2
	1.2
	1.3
	1.5
	IN1
	IN2
	IN3
	OUT
	TRUE
	Expandable
		Bool	**LE:**
			Comparison for less than or equal to
			The result is TRUE if IN1 is less than or equal to all other arguments.
			Example:
			TRUE = (1.2 <= 1.3 <= 1.5);
			ST: Use operator:
			`OUT:= IN1 <= IN2;`
			Only two arguments are allowed. Implementation with logical combination of multiple comparisons:
			`OUT := (IN1 < IN2) AND (IN1 < IN3);`

Any_ Elementary
Any_ Elementary
		Bool	**LT:** Comparison for less than
			The result is TRUE if IN1 is less than all other arguments.
			Example:
			TRUE = (3 < 6);
			ST: Use operator:
			`OUT:= IN1 < IN2;`
			Only two arguments are allowed. Implementation with logical combination of multiple comparisons:
			`OUT := (IN1 < IN2) AND (IN1 < IN3);`

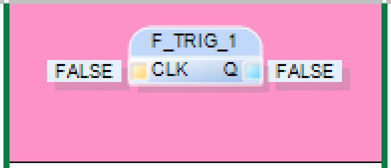
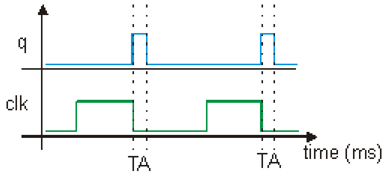
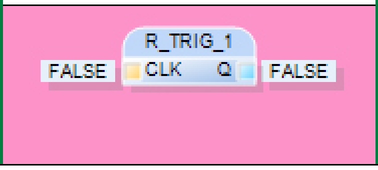
Any_ Elementary Any_ Elementary		Bool	**NE:** Comparison of inequality The result is TRUE if IN1 is not equal to N1. Example: TRUE = ('Text 1' <> 'Text 2'); **ST:** Use operator: `OUT:= IN1 <> IN2;`

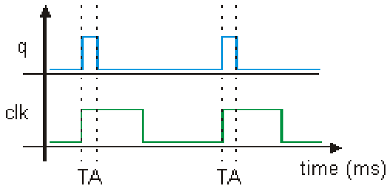
19.11 Counter

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool Bool Int	 CTD_1 FALSE FALSE 0 CD LOAD PV Q CV TRUE 0	Bool Int	**CTD:** Count down (Downwards counter) When the counter is set with LOAD = 1 the counter value CV is set to the initial value PV. With each rising edge of CD, the counter value CV is decremented by one. As soon as the counter output is CV <= 0, the output is set Q = 1. The CV output runs down to a minimum value of -32,768.  **ST:** cannot be called up
Bool Bool Int	 CTU_1 FALSE FALSE 0 CU RES... PV Q CV TRUE 0	Bool Int	**CTU:** Count up (Upwards counter) With each rising edge of CU, the counter value CV is incremented by one. As soon as the counter output CV > = count value PV, the output Q is set to "TRUE". When "RESET" = 1, the output Q is set to "FALSE" and the output CV is set to 0. The CV output runs up to a maximum value of 32767.  **ST:** cannot be called up
Bool Bool Bool Bool Int	 CTUD_1 FALSE FALSE FALSE FALSE 0 CU CD RESET LOAD PV QU QD CV TRUE TRUE 0	Bool Bool Int	**CTUD:** Counter for counting up and down Up/Down counter) With each rising edge of CU, the counter value "CV" is incremented by one per sampling time. When the

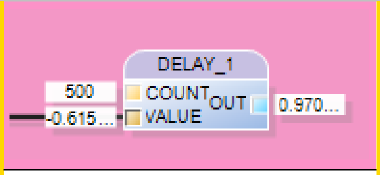
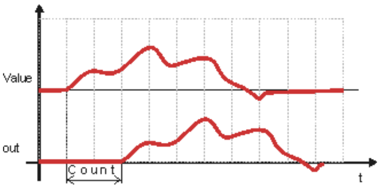
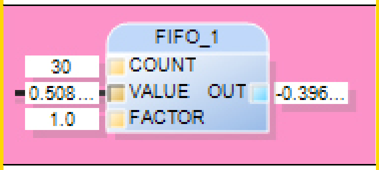
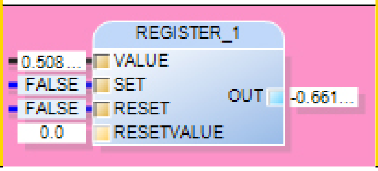
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
			counter output is "CV" >= count value "PV", the output is QU = 1 (Flow diagram see"CTU" FB). When the counter is set with "LOAD" = 1, the counter value "CV" is set to the initial value "PV". With each rising edge of "CD", the counter value "CV" is decremented by one. As soon as the counter output is "CV" <= 0, the output is "QD" = 1 (Flow diagram see "CTD" FB). When the counter is "RESET" = 1, the counter output is set to 0. Range of values for "CV": -32,768 to 32,767 **ST:** cannot be called up

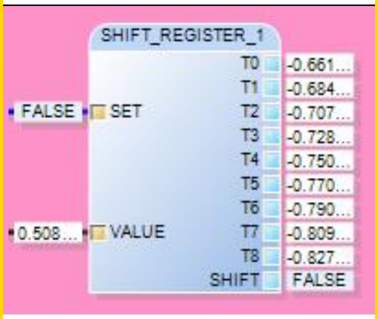
19.12 Edge Detection

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool		Bool	**F_TRIG:** Detecting falling edges With a falling edge at the input "CLK", the output Q is set to "TRUE" for one task cycle. Start-up behavior: When the input "CLK", is "FALSE" at the time of system start-up, the function block generates a pulse at the output Q = "TRUE" for a period of one cycle.  **ST:** cannot be called up
Bool		Bool	**R_TRIG:** Detecting falling edges

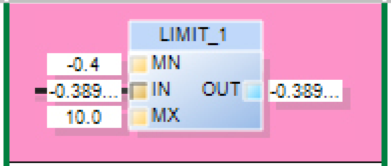
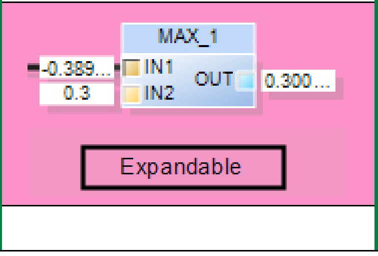
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
			With a rising edge at the input "CLK", the output Q is set to "TRUE" for one task cycle. Start-up behavior: When the input; "CLK", is "TRUE" at the time of system start-up, the function block generates a pulse at the output Q = "TRUE" for a period of one cycle.  **ST:** cannot be called up

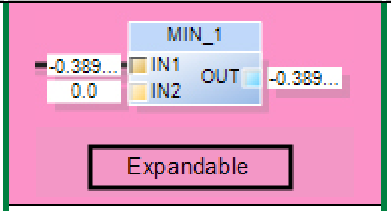
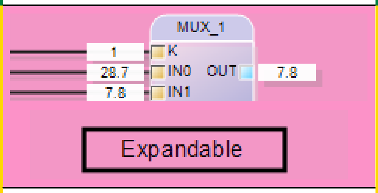
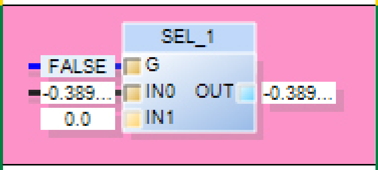
19.13 Register

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Dint Any	 DELAY_1 500 -0.615... COUNT VALUE OUT 0.970...	Any	**DELAY:** Time delay feature The output value "OUT", follows the input value "VALUE", with a time delay that is specified by the "COUNT" input in number of cycles.  When you use the "ARRAY" data type ("VALUE" and "OUT"), the function block is limited on account of memory capacity. If the number of "ARRAY" elements exceeds 64, the range of values of the time delay of 65,536 is reduced accordingly. **ST:** cannot be called up
Dint Real Real	 FIFO_1 30 0.508... 1.0 COUNT VALUE FACTOR OUT -0.396...	Real	**FIFO:** First In First Out - Storage The output value "OUT", follows the input value "VALUE", with a time delay that is specified by the "COUNT" input in number of cycles. In addition, the input value is multiplied with "FACTOR". **ST:** cannot be called up
Any_Magnitude Bool Bool	 REGISTER_1 0.508... FALSE FALSE 0.0 VALUE SET RESET RESETVALUE OUT -0.661...	Any_Magnitude	**REGISTER:** Register memory

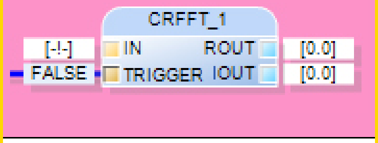
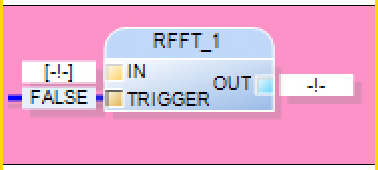
<table>
<tr><th>Input data type</th><th>Function unit design</th><th>Output data type</th><th>Explanation, example, ST syntax</th></tr>
<tr><td>Any_
Magnitude</td><td></td><td></td><td>The function block works with the signal state and not with the signal edges.
Function table:
<table>
<tr><th colspan="2">Input values</th><th>Output</th></tr>
<tr><td>SET</td><td>RESET</td><td>OUT</td></tr>
<tr><td>0</td><td>0</td><td>OUTn-1</td></tr>
<tr><td>0</td><td>1</td><td>RESETVALUE</td></tr>
<tr><td>1</td><td>0</td><td>VALUE</td></tr>
<tr><td>1</td><td>1</td><td>VALUE</td></tr>
</table>
<b>ST:</b> cannot be called up</td></tr>
<tr><td>Bool
Real</td><td>
</td><td>Real
Real
Real
Real
Real
Real
Real
Real
Bool</td><td><b>SHIFT_REGISTER:</b> Shift register
As long as the input "SET" = "TRUE", the input value, "VALUE" is shifted by an output Ti in every task cycle.
Shift, if "SET" = "TRUE"
T0: =
VALUE(T_n) = current cycle
T1: =
VALUE(T_{n-1}) = previous cycle
T8: =
VALUE(T_{n-8}] = oldest cycle
where n = task cycle
<b>ST:</b> cannot be called up</td></tr>
</table>

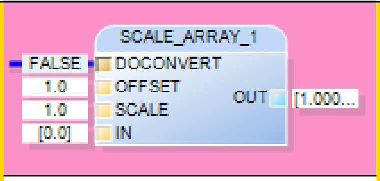
19.14 Selection

<table>
<tr><th>Input data type</th><th>Function unit design</th><th>Output data type</th><th>Explanation, example, ST syntax</th></tr>
<tr><td>Any_
Elementary
Any_
Elementary
Any_
Elementary</td><td>
</td><td>Any_Elementary</td><td><b>LIMIT:</b> Limit value
The input value IN is limited to the limit values MN (min.) and MX (max.).
Example:
-0.389 =
limit(-0.4, -0.389, 10.0);
15.3 =
limit(8.9, 17.6, 15.3);
<b>ST:</b>
<code>OUT :=</code>
<code>limit(MN, IN, MX);</code></td></tr>
<tr><td>Any_
Elementary
Any_
Elementary</td><td>
</td><td>Any_
Elementary</td><td><b>MAX:</b> Maximum value
Examples:
0.3 = max(-0.389, 0.3);
12 = max(0, 10, 12, 5);
<b>ST:</b>
<code>OUT :=</code>
<code>max(IN1, IN2, …, INn);</code></td></tr>
</table>

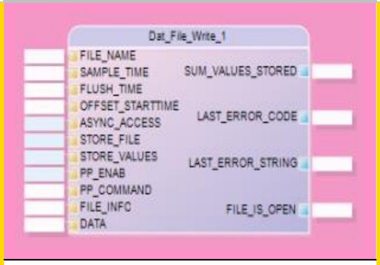
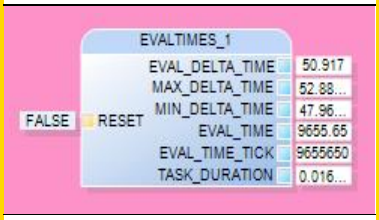
<table>
<tr><th>Input data type</th><th>Function unit design</th><th>Output data type</th><th>Explanation, example, ST syntax</th></tr>
<tr>
<td>Any_ Elementary
Any_ Elementary</td>
<td>
</td>
<td>Any_ Elementary</td>
<td><b>MIN:</b> Minimum value
Examples:
-0.389 = min(-0.389, 0.3);
0 = min(0, 10, 12, 5);
<b>ST:</b>
<code>OUT := min(IN1, IN2, …, INn);</code></td>
</tr>
<tr>
<td>Dint
Any
Any</td>
<td>
</td>
<td>Any</td>
<td><b>MUX:</b> Multiple selector
Expandable selection function block for any data types. All selection values have to be of the same data type.
"K" = Selector,
"IN0..IN63" selection values,
"OUT" resulting value.
<b>ST:</b> cannot be called up</td>
</tr>
<tr>
<td>Bool
Any_ Elementary
Any_ Elementary</td>
<td>
</td>
<td>Any_ Elementary</td>
<td><b>SEL:</b> Selector
Selection (1 out of 2) with binary signal "G"
Function table:
<table>
<tr><th>SEL</th><th>OUT</th></tr>
<tr><td>0</td><td>IN0</td></tr>
<tr><td>1</td><td>IN1</td></tr>
</table>
Example:
-0.389 = sel(FALSE, -0.389, 0);
<b>ST:</b> Different calls for "REAL" and "INT" data types, and other data types are not possible.
<b>ST:</b>
<code>OUT := sel_real(G,IN0, IN1);</code>
<code>OUT := sel_int(G, IN0,IN2);</code></td>
</tr>
</table>

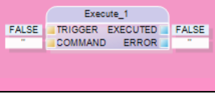
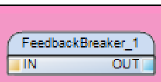
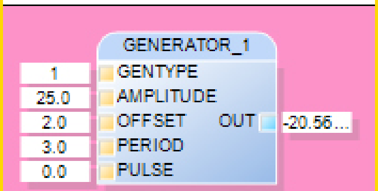
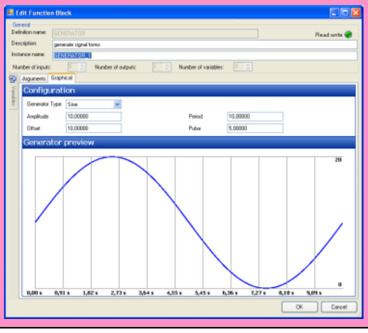
19.15 Signal Processing

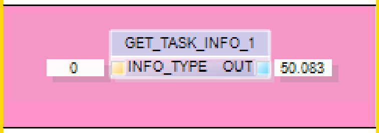
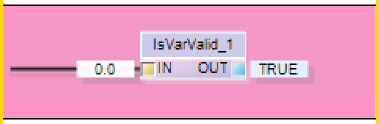
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
One-dimensional real array having a depth of 2^n Bool	 CRFFT_1 [-!-] IN ROUT [0.0] FALSE TRIGGER IOUT [0.0]	One one-dimensional real array each having a depth of 2^{n-1}	**CRFFT:** Fast Fourier Transformation with an imaginary component There must be a one-dimensional array of "REAL" type and having 2**n elements at the input "IN". The output is then always two arrays of the same type having the length 2 **(n-1). Example: IN ← ARRAY [0…2047] OF REAL ROUT → ARRAY [0…1023] OF REAL IOUT → ARRAY [0…1023] OF REAL The FFT evaluation is enabled when the input, "TRIGGER" = "TRUE". It is only then that the function block requires computing time! This function block delivers the real part at the output "ROUT" and the imaginary part at the output, "IOUT", of an FFT evaluation. Evaluation mode: Absolute amplitude, all values in the array have the same weight (Rectangular window). **ST:** cannot be called up
One-dimensional real array having a depth of 2^n Bool		One one-dimensional real array each having a depth of 2^{n-1}	**RFFT:** Fast Fourier Transformation There must be a one-dimensional array of "REAL" type and having 2**n elements at the input "IN". The output is then always two arrays of the same type having the length 2 **(n-1). Example: IN ← ARRAY [0 … 2047] OF REAL OUT → ARRAY [0 … 1023] OF REAL The FFT evaluation is enabled when the input, "TRIGGER" = "TRUE". Here, too, it is only then that the function block requires computing time! This function block delivers the result of an FFT at the output according to the evaluation mode: Absolute amplitude, all values in the array have the same weight (Rectangular window). **ST:** cannot be called up

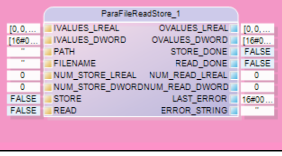
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool Real Real Any_ Derived	 SCALE_ARRAY_1 FALSE DOCONVERT 1.0 OFFSET 1.0 SCALE [0.0] IN OUT [1.000...	Any_ Derived	**SCALE_ARRAY:** As long as the input "DOCONVERT" = "TRUE", each element in the input array "IN", is multiplied with "SCALE" and added with the value at the "OFFSET" input. You then have the scaled array available at the output "OUT". **ST:** cannot be called up

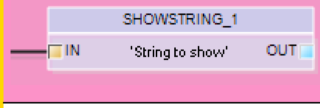
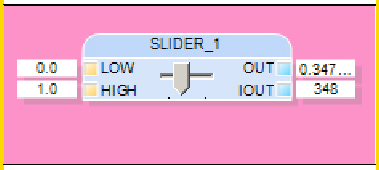
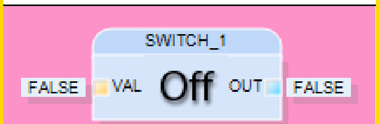
19.16 Specials

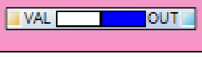
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
String Real Real Real Bool Bool Bool Bool String Any(_Struct) Any(_Struct)	 Dat_File_Write_1 FILE_NAME SAMPLE_TIME FLUSH_TIME OFFSET_STARTTIME ASYNC_ACCESS STORE_FILE STORE_VALUES PP_ENAB PP_COMMAND FILE_INFO DATA SUM_VALUES_STORED LAST_ERROR_CODE LAST_ERROR_STRING FILE_IS_OPEN	Dint Dword String Usint	**Dat_File_Write:** You can use this function block to record signals directly in ibaLogic for subsequent analysis using the ibaAnalyzer. You will find a detailed description in "*DAT_FILE_WRITE (DFW Function Block)*, page 95". **ST:** cannot be called up
Bool	 EVALTIMES_1 FALSE RESET EVAL_DELTA_TIME 50.917 MAX_DELTA_TIME 52.88... MIN_DELTA_TIME 47.96... EVAL_TIME 9655.65 EVAL_TIME_TICK 9655650 TASK_DURATION 0.016...	Real Real Real Real Udint Real	**EVALTIMES:** Output of the evaluation data EVAL_DELTA_TIME = current cycle time of the task (in ms) MAX_DELTA_TIME = max. cycle time of the task since the previous start (in ms) MIN_DELTA_TIME = minimum cycle time EVAL_TIME = time elapsed since the previous start (in ms) EVAL_TIME_TICK = time elapsed since the previous start in µs TASK_DURATION = evaluation time of the current task. **ST:** cannot be called up

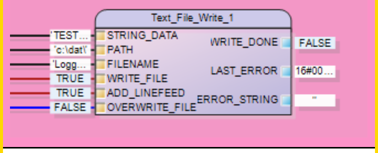
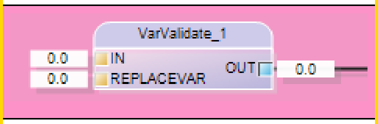
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool String	 Execute_1 FALSE TRIGGER EXECUTED FALSE COMMAND ERROR	Bool String	**Execute:** The function block calls up programs and batch files, etc. Equal to a call using the Windows command line CMD. "COMMAND" = Command line e.g. c:\myBatch.bat "TRIGGER" = Rising edge triggers the command "EXECUTED" = Goes to TRUE if the command has been executed successfully "ERROR" = Error as text message **ST:** cannot be called up
Any_derived	 symbolic display: 	Any_derived	**FeedbackBreaker:** The function block identifies the endpoint of feedbacks. Description see chapter *Feedback handling*, page 58
Int Real Real Real Real	 GENERATOR_1 1 GENTYPE 25.0 AMPLITUDE 2.0 OFFSET OUT -20.56… 3.0 PERIOD 0.0 PULSE **Graphical configuration**  Edit Function Block General Definition name: GENERATOR Read write Description: generate signal forms Instance name: GENERATOR_1 Number of inputs: 5 Number of outputs: 1 Number of variables: 0 Arguments Graphical Configuration Generator Type: Sine Amplitude: 10,00000 Period: 10,00000 Offset: 10,00000 Pulse: 5,00000 Generator preview Variables 20 0 0,00 s 0,91 s 1,82 s 2,73 s 3,64 s 4,55 s 5,45 s 6,36 s 7,27 s 8,18 s 9,09 s OK Cancel	Real	**GENERATOR:** Function generator for sinusoidal, rectangular and triangular signals. "GENTYPE" = 1 for sinusoidal, 2 for rectangular and 3 for triangular signal "AMPLITUDE" = Amplitude value; there is only one value that is evaluated symmetrically to the X-axis, i. e. it applies to both positive and negative values. "OFFSET" = Specification of the offset (position of the X-axis); if you desire a trend graph in which the value is non-negative, you must choose the offset at least as large as the amplitude.

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
			"PERIOD" = Specification of the time period in seconds "PULSE" (Pulse width) = Specification of the time for the first pulse in seconds; it is not used for sinusoidal waveforms. The value must not be greater than the time period. For a symmetric signal, Pulse = Period / 2 The specialty of this function block is that you also have the option of configuring the signals graphically. You can select this interface by double-clicking on the function block. You can use the mouse to set values in the graphical display. **ST:** cannot be called up
Int		Real	**GET_TASK_INFO:** Function to extract task information corresponding to the "INFO_TYPE" parameter. "INFO_TYPE" = 0: EvalDeltaTime, 1: EvalTime, 2: LastTaskDuration **ST:** cannot be called up
Any		Bool	**IsVarValid:** Checks a value for validity and indicates the result at the output. **ST:** `o1:= IsVarValid(i1);`

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
ParaFile_Lreal_Arr ParaFile_Dword_Arr String String Dint Dint Bool Bool	 ParaFileReadStore_1 [0, 0, ... [16#0... '' '' 0 0 FALSE FALSE IVALUES_LREAL IVALUES_DWORD PATH FILENAME NUM_STORE_LREAL NUM_STORE_DWORD STORE READ OVALUES_LREAL OVALUES_DWORD STORE_DONE READ_DONE NUM_READ_LREAL NUM_READ_DWORD LAST_ERROR ERROR_STRING [0, 0, ... [16#0... FALSE FALSE 0 0 16#00... ''	ParaFile_Lreal_Arr ParaFile_Dword_Arr Bool Bool Dint Dint Dword String	**ParaFileReadStore:** The function block allows to store parameters in a text file and to re-import them. The parameters can be written and read as LREAL or DWORD. The suitable inputs and outputs are available in each case. Always both file types are stored in the text file and read according to the configured number. "IVALUES_LREAL" = Parameters to be written as LREAL. Array with 128 LREAL, fixed iba file type: PARAFILE_LREAL_ARR "IVALUES_DWORD" = Parameters to be written as DWORD. Array with 128 DWORD, fixed iba file type: PARAFILE_DWORD_ARR "PATH" = Path of the text file "FILENAME" = Name of the text file (the file is created automatically) "NUM_STORE_LREAL" = Number of LREAL parameters "NUM_STORE_DWORD" = Number of DWORD parameters "STORE" = Writing the parameters to the text file with rising edge "READ" = Reading the parameters with rising edge "OVALUES_LREAL" = Parameters read back from the text file as LREAL. Array with 128 LREAL, fixed iba file type: PARAFILE_LREAL_ARR "OVALUES_DWORD" = Parameters read back from the text file as DWORD. Array with 128 DWORD, fixed iba file type: PARAFILE_DWORD_ARR "STORE_DONE" = Rising edge with every saving process READ_DONE" = Rising edge with every reading process "NUM_READ_LREAL" = Number of read LREAL values "NUM_READ_DWORD" = Number of read DWORD values "ERROR_STRING" = Error as text message **ST:** cannot be called up

<table>
<tr><th>Input data type</th><th>Function unit design</th><th>Output data type</th><th>Explanation, example, ST syntax</th></tr>
<tr><td>String</td><td>
</td><td>String</td><td><b>SHOWSTRING:</b>
Display element for displaying strings.
<b>ST:</b> cannot be called up</td></tr>
<tr><td>Real
Real</td><td>
</td><td>Real
Dint</td><td><b>SLIDER:</b> slide controller
Depending on the position of the slider, this function block delivers a value at its output "OUT", where the value lies between the limits of the input specifications (Minimum and maximum value). The inputs are preset by default to 0 and 1, but can be modified as required. (Double-click on the module and adjust the default values)
The output "IOUT" delivers the relative positional value of the slider in steps of one-thousandth (0 ... 1000).
The outputs are set only when the mouse button releases the slider.
If the slider pointer is marked, it can also be moved using the cursor keys → and ← .
<b>ST:</b> cannot be called up</td></tr>
<tr><td>Bool</td><td>

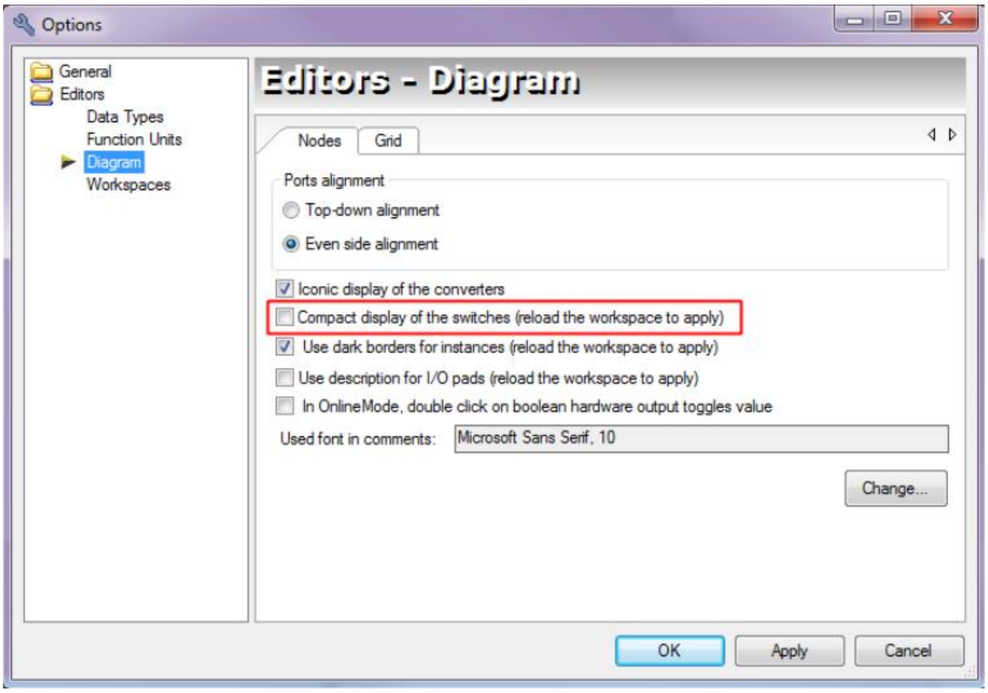
Display as compact symbol:

</td><td>Bool</td><td><b>SWITCH:</b> Switch
The SWITCH can be switch or pushbutton.
Left mouse button = PUSHBUTTON,
Right mouse button = SWITCH.
In conjunction with the input, you have an "OR" function between the switch position and the input.
Truth table:
<table>
<tr><th>SWITCH</th><th>VAL</th><th>OUT</th></tr>
<tr><td>ON</td><td>FALSE</td><td>TRUE</td></tr>
<tr><td>ON</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td>OFF</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>OFF</td><td>TRUE</td><td>TRUE</td></tr>
</table>
<b>ST:</b> cannot be called up
Display switch as compact symbol, see chapter <i>Display switch symbol</i>, page 326</td></tr>
</table>

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
String String String Bool Bool Bool	 Text_File_Write_1 'TEST... 'c:\dat\' 'Logg... TRUE TRUE FALSE STRING_DATA PATH FILENAME WRITE_FILE ADD_LINEFEED OVERWRITE_FILE WRITE_DONE FALSE LAST_ERROR 16#00... ERROR_STRING ''	Bool Dword String	**Text_File_Write:** This function block can write text files for logging. "STRING_DATA": ASCII data to be written to the file "PATH": Path to the output file "FILENAME": Name of the output file "WRITE_FILE": TRUE = write, so with constant TRUE the string_data is written in each cycle, if the last write command was terminated successfully. "ADD_LINEFEED": A line feed is automatically appended to the STRING_DATA text. This will write each entry into a new line. "OVERWRITE_FILE": FALSE = All data will be appended to the existing content in the text file. TRUE = Only the last text will be written to the file as it will be completely overwritten "WRITE_DONE": TRUE when file is written "LAST_ERROR": Hex error code "ERROR_STRING": error message as a string
Any Any	 VarValidate_1 0.0 0.0 IN REPLACEVAR OUT 0.0	Any	**VarValidate:** This function block can convert an invalid value (INVALID) to a valid one. "IN": Input value "REPLACEVAR": Replacement value if "IN" becomes INVALID. "OUT": valid output value **ST:** cannot be called up

19.16.1 Display switch symbol

It is possible to display the switch as a compact symbol.

The display option provides the option "compact display":



Options

Editors - Diagram

- General
- Editors
  - Data Types
  - Function Units
  - Diagram
  - Workspaces

Nodes | Grid

Ports alignment

○ Top-down alignment

● Even side alignment

☑ Iconic display of the converters

□ Compact display of the switches (reload the workspace to apply)

☑ Use dark borders for instances (reload the workspace to apply)

□ Use description for I/O pads (reload the workspace to apply)

□ In OnlineMode, double click on boolean hardware output toggles value

Used font in comments: Microsoft Sans Serif, 10

Figure 173: Setting for switch display

If this option is activated, the switches are now displayed as follows:

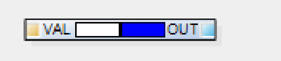


Figure 174: Compact switch display

Keep in mind that the inner area is the switching area. In order to move the switch select it with the mouse in the area of the inscription VAL / OUT.

The compact representation now allows to place the switches opposite to the connectors of a block.

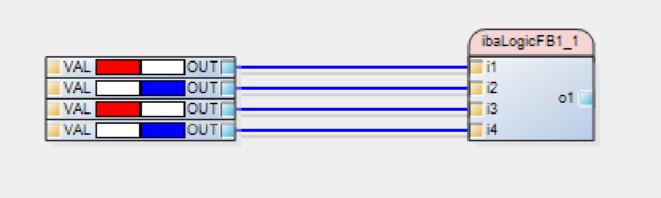


Figure 175: Alignment of switches and connectors

To change existing switches to compact display, the workspace must be reopened once the option has been set.

Existing switches are still displayed in their size unchanged and must be corrected manually by adjusting the height.

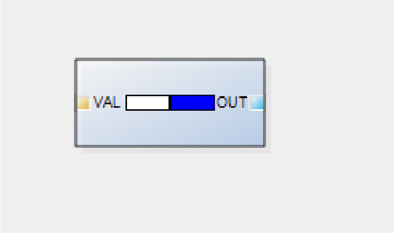
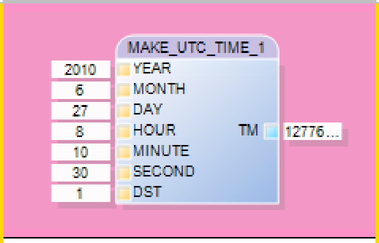
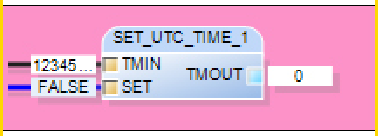
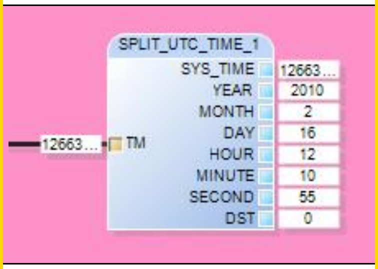


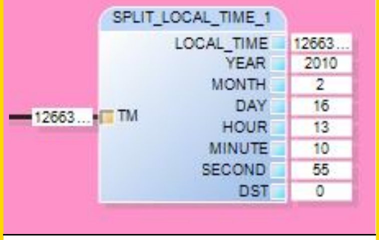
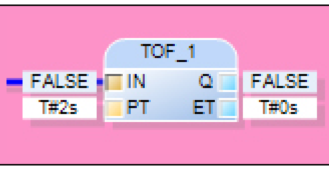
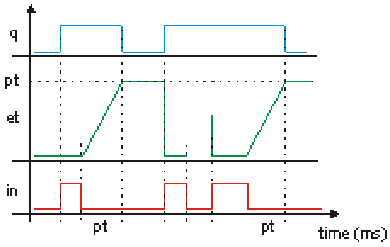
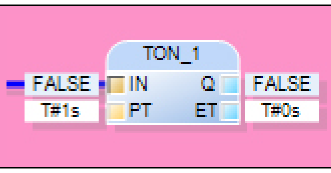
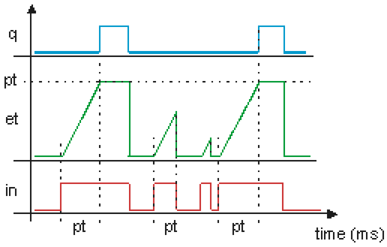
Figure 176: Existing switches after conversion

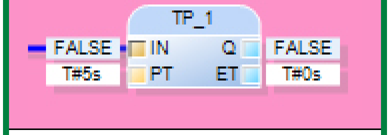
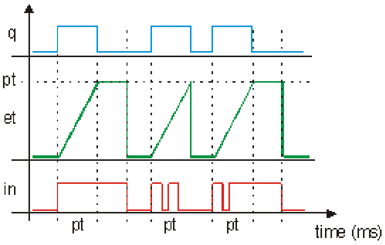
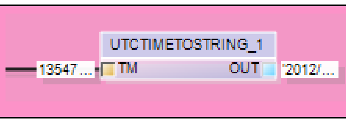
19.17 Timer

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Dint Dint Dint Dint Dint Dint Dint	 MAKE_UTC_TIME_1 2010 YEAR 6 MONTH 27 DAY 8 HOUR 10 MINUTE 30 SECOND 1 DST TM 12776...	Udint	**MAKE_UTC_TIME:** Coding the *UTC time*[9] The function block generates the UTC time at the output "TM" from the input variables "YEAR", "MONTH", "DAY", "HOUR", "MINUTE" and "SECOND". The local *timezone*[10] is**not** taken into consideration. You can specify the Daylight Savings Time at the "DST" input. Example: 27.06.2010/08:10:30 in the timezone GMT+01 → TM = 1277626230 **ST:** cannot be called up
Udint Bool		Udint	**SET_UTC_TIME:** Set the UTC time The function block set the "UTC System Time" of the ibaLogic platform (Windows PC or PADU-S-IT) to the value at the "TMIN" input when the input "SET" = "TRUE". The local time zone and the Daylight Savings Time (DST) is **not** taken into consideration. **ST:** cannot be called up
Udint	 SPLIT_UTC_TIME_1 12663... TM SYS_TIME 12663... YEAR 2010 MONTH 2 DAY 16 HOUR 12 MINUTE 10 SECOND 55 DST 0	Udint Dint Dint Dint Dint Dint Dint Dint	**SPLIT_UTC_TIME:** Decoding of UTC time in GMT The function block generates the output variables, "YEAR", "MONTH", "DAY", "HOUR", "MINUTE" and "SECOND" from the UTC time at the TM input. The local time zone is not taken into consideration. The Daylight Savings Time is displayed at the "DST" output. **ST:** cannot be called up

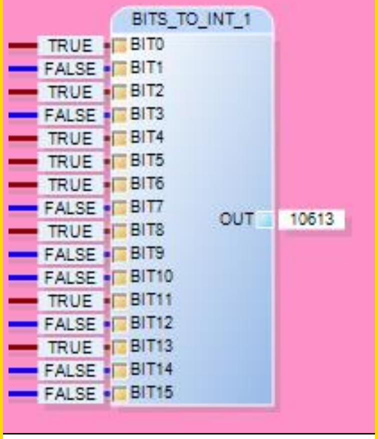
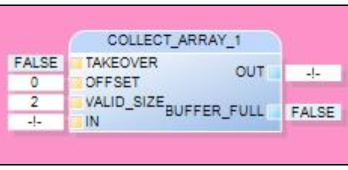
[9] Annotation: If the description is changed in the function block definition, it will appear in the instance only after reopening of the workspace.

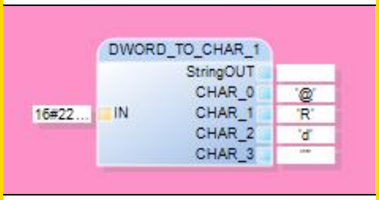
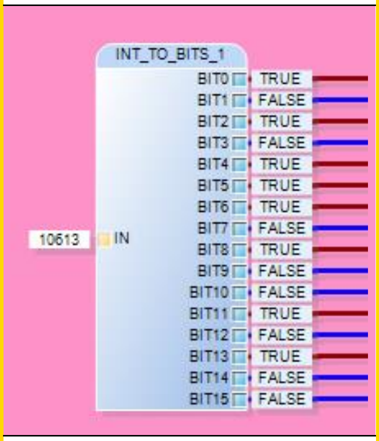
[10] The information regarding the time zone and summer time (DST) is taken over from the Operating System settings under Windows XP or Windows CE.

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Udint	 SPLIT_LOCAL_TIME_1 12663... TM LOCAL_TIME 12663... YEAR 2010 MONTH 2 DAY 16 HOUR 13 MINUTE 10 SECOND 55 DST 0	Udint Dint Dint Dint Dint Dint Dint Dint	**SPLIT_LOCAL_TIME:** Decoding of UTC time in local time The function block generates the output variables "YEAR", "MONTH", "DAY", "HOUR", "MINUTE" and "SECOND" from the UTC time at the TM input. The local time zone is not taken into consideration. The Daylight Savings Time is displayed at the "DST" output. **ST:** cannot be called up
Bool Time	 TOF_1 FALSE IN Q FALSE T#2s PT ET T#0s	Bool Time	**TOF:** Off delay (Switch off time delay) If the input "IN", is "TRUE", the output "Q", is set to "TRUE" without any time delay. The falling edge at the "IN" input starts the time delay PT. After the delay time has elapsed, the output "Q", is set to "FALSE". The output "Q", remains unchanged if the switch-off time of "IN" is shorter than the time delay. The output "ET", indicates the time that has already elapsed.  **ST:** cannot be called up
Bool Time	 TON_1 FALSE IN Q FALSE T#1s PT ET T#0s	Bool Time	**TON:** On delay (Switch on time delay) The rising edge at the "IN" input starts the time delay PT. After the delay time has elapsed, the output "Q", is set to "FALSE". A "FALSE" signal at the input "IN", is immediately transferred to the output "Q". The output "Q", is not set if the switch-on time of "IN" is shorter than the time delay. The output "ET", indicates the time that has already elapsed.  **ST:** cannot be called up

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool Time	 TP_1 FALSE IN Q FALSE T#5s PT ET T#0s	Bool Time	**TP:** Timer pulse (Pulse extension) The rising edge at the input "IN", sets the output "Q", for the pulse time PT to "TRUE". The output "Q", cannot be reset during the timer pulse. The output "ET", indicates the time that has already elapsed.  **ST:** cannot be called up
Udint		String	**UTCTIMETOSTRING:** Converts UTC time in a formatted string. **ST:** `OUT := UTCTIMETOSTRING(IN);`

19.18 Type Conversion

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool	 BITS_TO_INT_1 TRUE BIT0 FALSE BIT1 TRUE BIT2 FALSE BIT3 TRUE BIT4 TRUE BIT5 TRUE BIT6 FALSE BIT7 TRUE BIT8 FALSE BIT9 FALSE BIT10 TRUE BIT11 FALSE BIT12 TRUE BIT13 FALSE BIT14 FALSE BIT15 OUT 10613	Int	**BITS_TO_INT:** Converts 16 bits to an integer value Example: 10613 = 2#0010_1001_0111_0101, (Bit15 ……….…………… Bit0) **ST:** cannot be called up
Bool Uint Uint	 COLLECT_ARRAY_1 FALSE TAKEOVER 0 OFFSET 2 VALID_SIZE -!- IN OUT -!- BUFFER_FULL FALSE	Any_ Derived	**COLLECT_ARRAY** You can use this function block to transfer sections of one array to another.

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Any_Derived		Bool	"TAKEOVER": As long as this input is "TRUE", data is transferred to the output array. "OFFSET": This specifies the element in the input array from which data needs to be copied. "VALID_SIZE": Number of elements to be copied. "IN": Input array of any size. "OUT": Output array "BUFFER_FULL": Output array is full, and data has to be read out. **ST:** cannot be called up Note: The 'COLLECT_ARRAY' has an internal output buffer. Using BUFFER_FULL the internal buffer is copied to OUT so that the data is statically present until the next BUFFER_FULL. The new data is collected in the internal buffer in the meantime.
Dword	 DWORD_TO_CHAR_1 16#22... IN StringOUT CHAR_0 CHAR_1 CHAR_2 CHAR_3 '@' 'R' 'd' ''	String String String String String	**DWORD_TO_CHAR** Conversion of a "DWORD" to four separate characters of "STRING" type 16#22645240 = ' @', 'R', 'd', ''" **ST:** cannot be called up within ST
Int	 INT_TO_BITS_1 10613 IN BIT0 TRUE BIT1 FALSE BIT2 TRUE BIT3 FALSE BIT4 TRUE BIT5 TRUE BIT6 TRUE BIT7 FALSE BIT8 TRUE BIT9 FALSE BIT10 FALSE BIT11 TRUE BIT12 FALSE BIT13 TRUE BIT14 FALSE BIT15 FALSE	Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool	**INT_TO_BITS:** Converts an integer value to 16 bits (Inverse function of BITS_TO_INT) **ST:** cannot be called up

19.18.1 Limiting Converter

These conversion function units take on a special role, since, prior to the type conversion, they limit the range of values of the input type to the range of values of the output type. The following example illustrates the difference with respect to a standard converter.

Limit converter:	limit:dint_to int(57700)	delivers the result:	32767

(First, the output value is limited to the range of values (-32768 … 32767), and then type conversion is carried out).

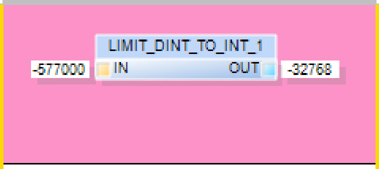
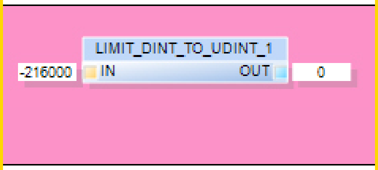
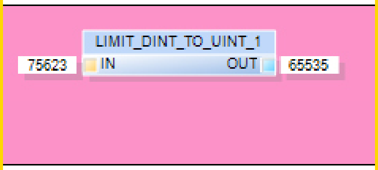
Standard converter:	dint_to_int(577000)	delivers the result:	-12824

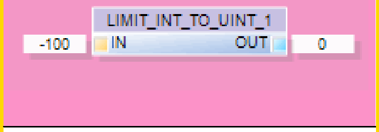
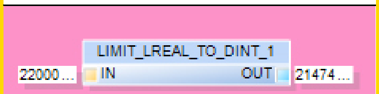
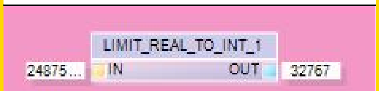
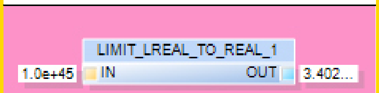
(The 16 lower order bits are considered and these are converted, whereby, naturally, the highest order bit (Bit 15) is interpreted as a sign bit.)

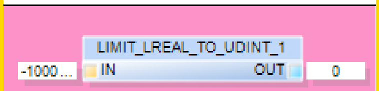
We recommend that you use a limiting converter if the range of values of the target type is smaller than the range of values of the source type. This concerns the following conversions:

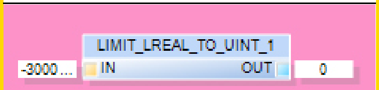
INT	UINT	DINT	UDINT	REAL	LREAL
INT → UINT INT → SINT INT → USINT	UNIT → INT UINT → SINT UINT → USINT	DINT → INT DINT → UDINT DINT → UINT DINT → SINT DINT → USINT	UDINT → DINT UDINT → INT UDINT → UINT UINT → SINT UINT → USINT	REAL → DINT REAL → INT REAL → UDINT REAL → UINT REAL → SINT REAL → USINT	LREAL → DINT LREAL → INT LREAL → REAL LREAL → UDINT LREAL → UINT LREAL → SINT LREAL → USINT

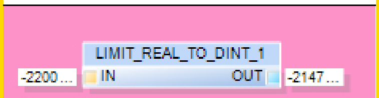
In Structured Text, the limiting converters are provided by the functions: "limit_*source_type*_to_*target_type*"

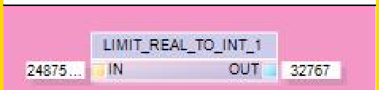
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Dint		Int	**LIMIT_DINT_TO_INT:** Example: -577000 => -32768; **ST:** `OUT:= limit_dint_to_int(IN);`
Dint		Udint	**LIMIT_DINT_TO_UDINT:** Example: -216000 => 0; **ST:** `OUT:= limit_dint_to_udint(IN);`
Dint		Uint	**LIMIT_DINT_TO_UINT:** Example: 75623 => 65535 **ST:** `OUT:= limit_dint_to_uint(IN);`

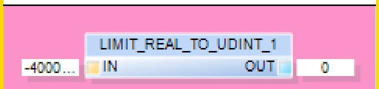
Input data type	Function unit design	Output data type	Explanation, example, ST syntax
Int		Uint	**LIMIT_INT_TO_UINT** Example: -100 => 0 **ST:** `OUT:= limit_int_to_uint(IN);`
Lreal		Dint	**LIMIT_LREAL_TO_DINT:** Example: 2.2 E+09 => 2147483648; **ST:** `OUT:= limit_lreal_to_dint(IN);`
Lreal		Int	**LIMIT_LREAL_TO_INT:** Example: 248758.0 => 32767 **ST:** `OUT:= limit_lreal_to_int(IN);`
Lreal		Real	**LIMIT_LREAL_TO_REAL** Example: 1E+45 => 3.402823466 E+38 **ST:** `OUT:= limit_lreal_to_real(IN);`

Lreal		Udint	**LIMIT_LREAL_TO_UDINT:**
			Example:-1 E+12 => 0;
			ST:
			`OUT:= limit_lreal_to_udint (IN);`

Lreal		Uint	**LIMIT_LREAL_TO_UINT:**
			Example:-3 E+12 => 0;
			ST:
			`OUT:= limit_lreal_to_uint(IN);`

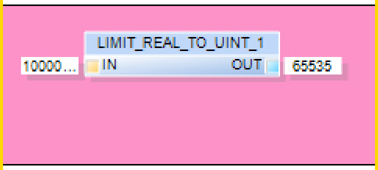
Real		Dint	**LIMIT_REAL_TO_DINT:** Example: -2.2 E+09 => -2147483648; **ST:** `OUT:= limit_real_to_dint(IN);`

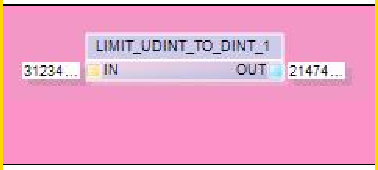
Real		Int	**LIMIT_REAL_TO_INT:**
			Example: 248758.0 => 32767;
			ST:
			`OUT:= limit_real_to_int(IN);`

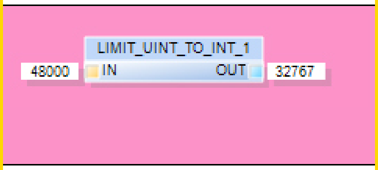
Real		Udint	**LIMIT_REAL_TO_UDINT:** Example: -4000.0 => 0 **ST:**

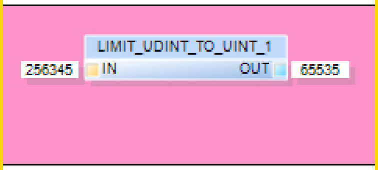
Input data type	Function unit design	Output data type	Explanation, example, ST syntax

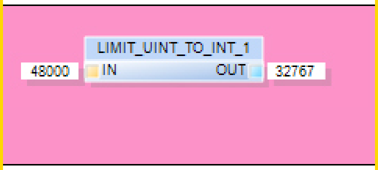
			`OUT:=`
			`limit_real_to_udint(IN);`

Real		Uint	**LIMIT_REAL_TO_UINT:**
			Example:
			1*E+12 => 65535;
			ST:
			`OUT:=`
			`limit_real_to_uint(IN);`

Udint		Dint	**LIMIT_UDINT_TO_DINT:**
			Example:
			3123456789 => 2147483647;
			ST:
			`OUT:=`
			`limit_udint_to_dint(IN);`

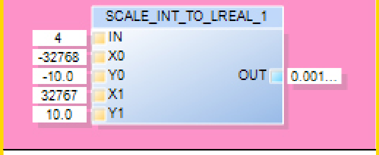
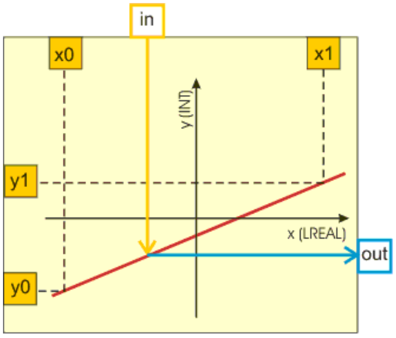
Udint		Int	**LIMIT_UDINT_T_INT:**
			Example:
			558900 => 32767
			ST:
			`OUT:=`
			`limit_udint_to_int(IN);`

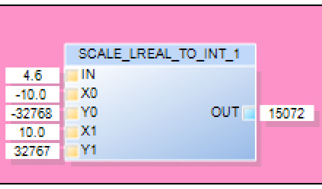
Udint		Uint	**LIMIT_UDINT_TO_UINT:**
			Example:
			256345 => 65535
			ST:
			`OUT:=`
			`limit_udint_to_uint(IN);`

Uint		Int	**LIMIT_UINT_TO_INT:**
			Example: 48000 => 32767;
			ST:
			`OUT:=`
			`limit_uint_to_int(IN);`

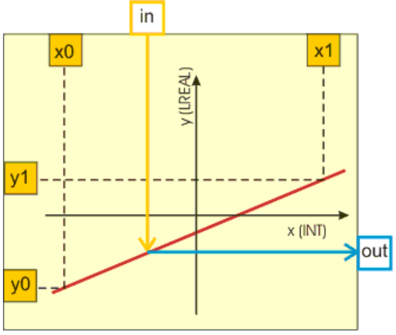
19.18.2 Scaling Converter

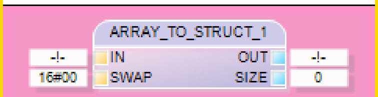
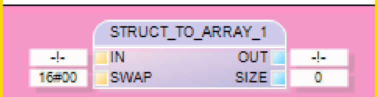
Input data type	Function unit design	Output data type	Explanation, example, ST syntax

Int
Int
Lreal
Int
Lreal
	
	SCALE_INT_TO_LREAL_1
	4 IN
	-32768 X0
	-10.0 Y0
	32767 X1
	10.0 Y1
	OUT 0.001...
		Lreal	**SCALE_INT_TO_LREAL:**
			This module converts an "INTEGER" value to an "LREAL" value and scales it linearly.
			Application:
			Conversion of an analog input (Integer value -32768 … 32767) to a physical parameter
			e. g. +/- 10 Volt.
			IN:
			Input value (Analog input)
			X0, X1: Range of values of input value (Int)
			Y0, Y1: Range of values
			Target parameter (Lreal)
			Example:
			4 → 0.0013733119 V
			X0, X1 = -32768 / +32767
			Y0, Y1 = -10.0 / + 10.0
			IN = 4
			OUT = 0.0013733119
			
			
			Implementation (Type conversions have been omitted for the sake of clarity):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 - y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out = aa*in + bb;`
			`end_if;`
			ST: cannot be called up
			Important note:
			Since the integer range of values is not symmetric in principle, a 0 at the input leads to a 0 at the output. Since this always leads to misinterpretations, iba AG recommends that you specify a symmetric range of values for the input (-32,767/+ 32,767).

Lreal Lreal Int	 SCALE_LREAL_TO_INT_1 4.6 IN -10.0 X0 -32768 Y0 10.0 X1 32767 Y1 OUT 15072	Int	**SCALE_LREAL_TO_INT:** This module converts an "LREAL" value to an "INTEGER" value and scales it linearly.

Input data type	Function unit design	Output data type	Explanation, example, ST syntax

Lreal
Int
			Application: Conversion of physical parameter
			(e. g. +/- 10 Volt) to an analog output
			(Integer value -32,768 ... 32,767)
			Example:
			4.6 V → 15072
			X0, X1 = physical range (-/+ 10 V)
			Y0, Y1 = range of values INT
			
			Implementation (Type conversions have been omitted for the sake of clarity):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 - y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out := aa*in + bb;`
			`end_if;`
			ST: cannot be called up

Any_Array Byte	 ARRAY_TO_STRUCT_1 -!- 16#00 IN SWAP OUT SIZE -!- 0	Any_Struct Int	**ARRAY_TO_STRUCT:** Creates a structure from any array. SIZE includes the effectively used data size. SWAP sets the byte swap: 16#00 Off, 16#01 depending on data type (AB CDEF ♋♦ BA FEDC), 16#02 2 bytes (ABCD ♋♦ BADC), 16#04 4 bytes (ABCD ♋♦ DCBA). **ST:** cannot be called up
Any_Struct Byte	 STRUCT_TO_ARRAY_1 -!- 16#00 IN SWAP OUT SIZE -!- 0	Any_Array Int	**STRUCT_TO_ARRAY:** Creates an array from any structure. (Parameter see ARRAY_TO_STRUCT) **ST:** cannot be called up

19.18.3 Standard Converter

All standard conversion function units are combined into one unit. As soon as you drag this function unit and drop it in the design area, a selection dialog box appears with which you can define the input and output data types.

Conversion rules:

- For target type "BOOL", the result is "FALSE" if the input has the value 0, 0.0, 16#0 or T#0ms, otherwise, "TRUE" is output.
- For conversion of real data types to integer data types, the values are converted numerically and rounded up or off in accordance with the rules of arithmetic. The values are not limited.
  Example: 82600.0(REAL) → 82600(DINT) → 17064(INT)
- Conversion from "INTEGER", "REAL" and "WORD" data types takes place using type conversion without changing the bit pattern. Any limiting required is not carried out and the higher orders bits are truncated.

If you join two connectors of different data types with one another, a suitable conversion function unit is automatically created, provided that conversion is possible. For more information, please refer to "Converter".

Input data type	Function unit design	Output data type	Explanation, example, ST syntax
			TYPE TO TYPE: As soon as you drag and drop this function unit in the design area, a selection dialog box appears with which you can define the input and output data type. **ST:** Function name is formed from *Source_type*_to_*Target_type*, e.g. `OUT := real_to_int(IN);`

Note

Conversion to TIME:

Integer values are interpreted as millisecond values.

Real values are interpreted as seconds values.

Example:

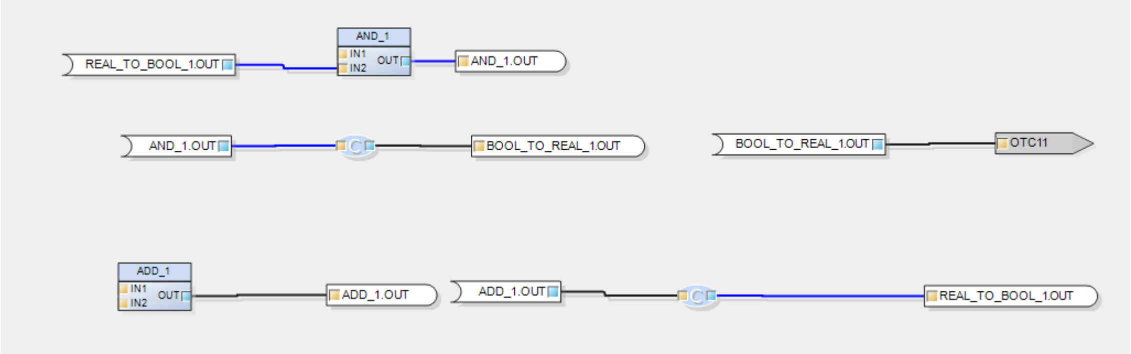
An integer value of 4,711 hence, yields 4,177 seconds.

A real value of 4711.0 yields 4711 seconds
(that is, 1 h, 18 min and 31 sec).

20 Export/Import for external generated programs

A reduced import was realized from the requirement to have ibaLogic plans generated automatically.

A reduced import creates such a layout, for example:



All connections are replaced by IntraPageConnectors (IPCs). This means that it is not necessary to provide exact information about the "connection path", i. e. the path of the connection.

Also other unnecessary information was removed from this reduced import.

Using an export, you can create a template for such an import file.

As a user you could now generate such a file with reduced information via an own written program (e.g. an C# programm). That can be imported and so an automatically generated ibaLogic program can be build.

Example:

An export has been generated for a plant part. This is however required depending on the customer 1..x times. This export can be used as a basis and and can be modified with the help of the program by copying and replacing.

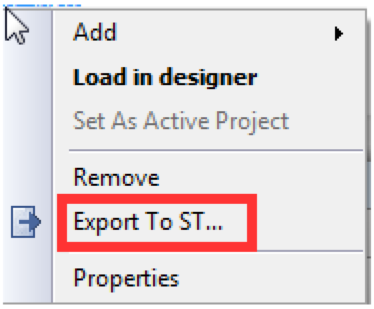
The export template contains GUIDs. These must then be adapted accordingly when copying. This means that if a template is to be used multiple times in the target, the GUIDs used must be modified for each copy so that the same GUIDs of the original are put together on new GUIDs, that the references between the connections are retained, but all GUIDs per se are unique.

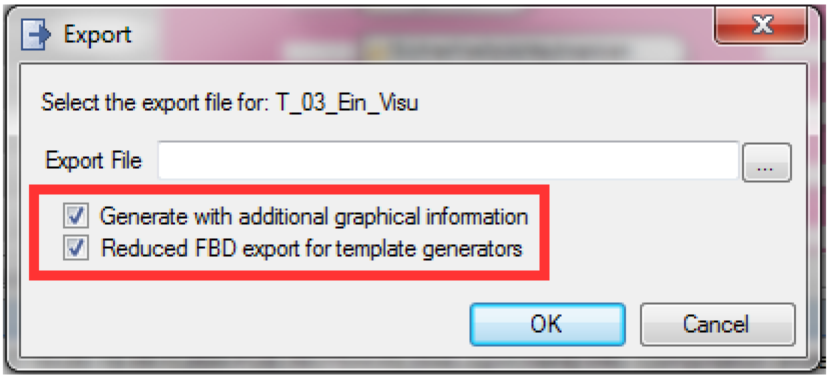
For the placement of inputs and outputs, note that these have a height of 22.03 and an offset of 3.01. Thus, the position of a connector can be easily calculated using (Index * 22.03) +3.01.

The naming of the inputs and outputs depends on the used hardware.

Reduced Export:

The reduced export is called up in the context menu of the project, the individual …..





Select the export file for: T_03_Ein_Visu

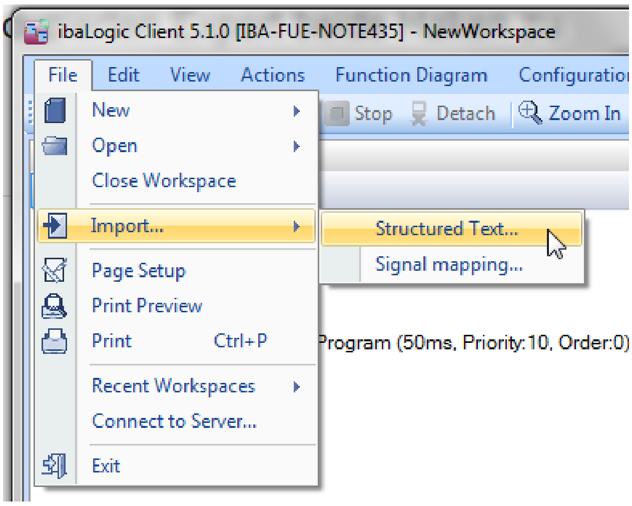
Export File

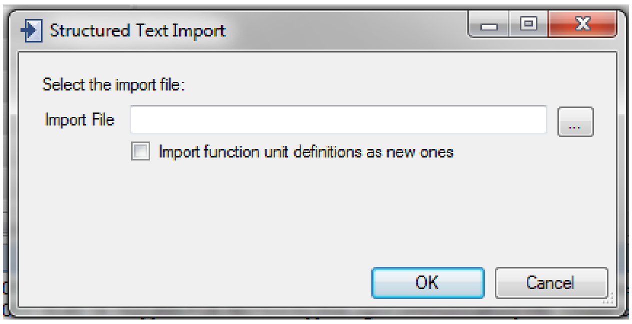
☑ Generate with additional graphical information
☑ Reduced FBD export for template generators

For a reduced export, both checkboxes must be set.

Reduced import:

The import is done as usual via "File – Import - Structured Text …"





Select the import file:

Import File

□ Import function unit definitions as new ones

A reduced import is then automatically detected.

21 Parallel operation of ibaLogic-V5 and ibaPDA on one computer

If you want to run ibaLogic-V5 and ibaPDA simultaneously on one computer, the following must be taken into consideration:

❑ The hardware cards must be clearly assigned to one of the programs. For this purpose, the automatically found cards must be activated in the IO configurator of ibaLogic and ibaPDA or deactivated in the other program.

❑ If both software products are installed, it is possible that the software which has been started first detects the hardware and reserves it for itself. This can lead to driver problems.
In this case the configuration of ibaLogic and ibaPDA has to be corrected and the computer has to be restarted, so that any occupied drivers can be released again.

❑ Assignments are always based on cards, not links.
It might make sense that, depending on the application type, only one program has access to all cards and sends its data to the other program.

ibaLogic on ibaDAQ-S

If ibaLogic is installed on an ibaDAQ-S central unit, this is detected automatically and the hardware driver is automatically disabled. The ibaDAQ-S hardware and the IO modules are only used by ibaPDA.

ibaLogic can then be used as a co-processor to ibaPDA or other tasks.

22 ibaLogic differences in version/conversion V3/V4/V5

Tip

Please contact the "*Support*, Page 358" for the conversion from V3 or V4 to V5. Here, the iba AG layout should be briefly checked to define the required steps.

What are the special features of the versions and what is to be observed when converting.

Innovations in version V5 compared to version V4

❑ Version V5 is based on a new runtime system

❑ ibalogic-V5 does not support ibaPADU-S-IT-16, but only ibaPADU-S-IT-**2x16**

❑ Now, faster configuring is possible

❑ In addition to interval tasks, there are also event tasks now

❑ Now, the tasks interrupt themselves according to their priority

❑ There is a proper control element (TrayIcon) for the PMAC

❑ Playback of iba .dat files for the input signal simulation is possible

❑ A totals overview of the task runtimes exists

❑ The 64kByte memory capacities of V4 are removed

❑ Strings can be defined freely again, no 256 limitation

❑ Index evaluations, e.g. array indices, are possible.

❑ There is an INVALID display at the function units (red connector field) if evaluation errors appear, e.g. division by 0

❑ The DatFileWrite function block (DFW) has been newly implemented. Therefore, the former function block has to be replaced and reconnected. For this purpose, the old DFW is read as a dummy when opening the V4 project in V5. It can be opened and the old configuration can be seen. This serves as a little help to configure the new function block.

  - For each module, it can be specified if buffered or unbuffered values apply. Thereby, buffered and unbufferd values can be recorded together now.
  - a dynamic allocation of names of the channels is possible.
  - iba data files (*.dat) can be stored compressed or uncompressed.

❑ V5 has its own donglebit.

❑ V5 also has its own database, therefore, V4 and V5 can be open on the computer when offline. Only one PMAC can be started.

❑ There is a default and an actual value field in the online mode of the function unit. A change of the default field will become applicable only after the next evaluation start. Changes in the associated actual value field will apply immediately.

❑ ibaPDA Express is only available as stand-alone version

❑ The turbo mode is hard coded now.

- ❑ The measurement mode and soft PLC mode have a slightly different meaning: Measurement = buffered mode / SOFT PLC = unbuffered mode.
- ❑ Both variants are available in measurement mode now, the buffered and the unbuffered value
- ❑ The master mode external does no longer exist, only the master/slave mode exists.
- ❑ For easier handling, only one TreeView (IEC view) program has been implemented
- ❑ The intersections of the lines do no longer have curves. This makes the view smoother.
- ❑ At the moment, there is no QPanel display directly in ibaLogic
- ❑ The fuzzy function block has been taken out
- ❑ A multiclient operation is not supported
- ❑ Breakpoints in ST are not supported anymore (security)

Note: to be observed with V5

- Due to the interruptibility of tasks, a certain base load emerges. The basic clock cycle of ibaLogic should therefore always be adapted to the minimum interval task, otherwise unnecessary load will be produced.
- User DLL that are not used have an influence on the startup time and the time of configuring.
  It is therefore recommended to delete user DLLs that are not needed from the DLL directory.
- An export from V4 with DFW has to be imported into V4 first, then completely loaded and corrected in V5 and after that an export for V5 can be generated again. This is caused by the new implementation of the DFW
- V4 had a wrong default value at UDINT, e.g. 2.0, the V5 notices this and displays possible errors
- There is a new UserDLL f. ExecuteDLL;
  instead of the executeDLL, there is an execute-FB under *Specials*, page 320
- There is an internal FB TextFileWrite instead of the external Log_File_Write DLL.

Note: Creating V4 behavior with V5

In order to receive the same behavior under V5 as it was in V4, the following needs to be considered:

In V4, all tasks have been evaluated one after the other, an interruption was not possible. There have been interval tasks only.

If you want to have the same behavior, you therefore need to pay attention in V5 that

- all tasks are still interval tasks
- all tasks need to have the **same priority** (thereby, they will not be interrupted)
- the tasks need to have an **order** which conforms to their order in V4.

23 Error Codes

23.1 DAT_FILE_WRITE Error Codes

Error Code	Meaning
16#FFFFFFF1	`No PP_COMMAND specified!`
16#FFFFFFF2	`Failed to execute PP_COMMAND!`
16#FFFFFFF3	`Failed to start PP_COMMAND!`
16#FFFFFFF4	`Failed to close file xxxx.dat`
16#FFFFFFF5	`Sample Time is set to 0.0. Please set a valid sample time!`
16#FFFFFFF6	`DatfileWrite Configuration exceeds allowed signals in Dongle`
16#FFFFFFF7	`Failed to write data into file. Please check disk space`
16#FFFFFFF8	`Failed to create file. Please check file name and disk space!`
16#FFFFFFF9	`Could not find Data handlers for module x`
16#FFFFFFFA	`Error collecting Data handlers for module x`
16#FFFFFFFB	`Could not find Data handlers`
16#FFFFFFFC	`Could not find Module configuration data for module x`
16#FFFFFFFD	`Error reading Module configuration data for module x`
16#FFFFFFFE	`No Module Configuration defined!`
16#FFFFFFFF	`Could not find Configuration data`

23.2 TCPIP_SENDRECV Error Codes

Error Code	Meaning	Solution suggested
HEX: 16#0000271d DEZ: 10013	Permission denied - Access to socket forbidden by access permissions.	Please login with a username that has administrator privileges.
16#00002740 10048	Address already in use - Only one usage of each socket address is permitted.	The address / port is already used.
16#00002741 10049	Cannot assign requested address - The requested address is not valid in its context.	
16#0000273f 10047	Address family not supported by protocol family - An address incompatible with the requested protocol was used.	
16#00002735 10037	Operation already in progress - An operation was attempted on a non-blocking socket that already had an operation in progress.	
16#00002745 10053	Software caused connection abort - An established connection was aborted by host machine.	
16#0000274d 10061	Connection refused - No connection could be made because the target machine actively refused it.	
16#00002746 10054	Connection reset by peer - An existing connection was forcibly closed by the remote host.	
16#00002737 10039	Destination address required - A required address was omitted from an operation on a socket.	

Error Code	Meaning	Solution suggested
16#0000271e 10014	Bad address - The system detected an invalid pointer address in attempting to use a pointer argument of a call.	
16#00002750 10064	Host is down - A socket operation failed because the destination host was down.	
16#00002751 10065	No route to host - A socket operation was attempted to an unreachable host.	
16#00002734 10036	Operation now in progress - A blocking operation is currently executing.	
16#00002714 10004	Interrupted function call - A blocking operation was interrupted by a call to WSACancelBlockingCall.	
16#00002726 10022	Invalid argument - Some invalid argument was supplied.	
16#00002748 10056	Socket is already connected - A connect request was made on an already connected socket.	
16#00002728 10024	Too many open files - Too many open sockets.	
16#00002738 10040	Message too long - A message sent to socket was larger than the internal message buffer or the buffer used to receive was smaller than the datagram itself.	Reduce the length of the bytes to be transmitted.
16#00002742 10050	Network is down - A socket operation encountered a dead network.	
16#00002744 10052	Network dropped connection on reset - The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress.	
16#00002743 10051	Network is unreachable - A socket operation was attempted to an unreachable network.	
16#00002747 10055	No buffer space available - An operation could not be performed because the system lacked sufficient buffer space or because a queue was full.	
16#0000273° 10042	Bad protocol option - An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.	
16#00002749 10057	Socket is not connected - A request to send or receive data was disallowed because the socket is not connected.	
16#00002736 10038	Socket operation on non-socket - An operation was attempted on something that is not a socket.	
16#0000273d 10045	Operation not supported - The attempted operation is not supported for the type of object referenced.	
16#0000273e 10046	Protocol family not supported - The protocol family has not been configured into the system or no implemen\-tation for it exists.	
16#00002753 10067	Too many processes - A Windows Sockets implementation may have a limit on the number of applications that may use it simultaneously.	
16#0000273b 10043	Protocol not supported - The requested protocol has not been configured into the system, or no implementation for it exists.	
16#00002739 10041	Protocol wrong type for socket - A protocol was specified in the socket function call that does not support the semantics of the socket type requested.	

Error Code	Meaning	Solution suggested
16#0000274a 10058	Cannot send after socket shutdown - A request to send or receive data was disallowed because the socket had already been shut down.	
16#0000273c 10044	Socket type not supported - The support for the specified socket type does not exist in this address family.	
16#0000274c 10060	Connection timed out - A connection attempt failed or established connection failed because connected host has failed to respond.	
16#0000277d 10109	Class type not found - The specified class was not found.	
16#0000277a 10106	Unable to initialize a service provider - Either a service provider's DLL could not be loaded or the provider's WSPStartup/NSPStartup function failed.	
16#00002af9 11001	Host not found - No such host is known.	
16#0000276d 10093	Successful WSAStartup not yet performed - Either the application hasn't called WSAStartup or WSAStartup failed.	
16#00002afc 11004	Valid name, no data record of requested type - The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for.	
16#00002afb 11003	This is a non-recoverable error - This indicates some sort of non-recoverable error occurred during a database lookup.	
16#0000277b 10107	System call failure - Returned when a system call that should never fail does.	
16#0000276b 10091	Network subsystem is unavailable - The underlying system to provide network services is currently unavailable.	
16#00002afa 11002	Non-authoritative host not found - Temporary error during hostname resolution, the local server did not receive a response from an authoritative server.	
16#0000276a 10092	WINSOCK.DLL version out of range - The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application.	
16#00002775 10101	Graceful shutdown in progress - The remote party has initiated a graceful shutdown sequence.	
16#00002778 10104	Invalid procedure table from service provider - A service provider returned a bogus proc table to WS2_32.DLL.	
16#00002779 10105	Invalid service provider version number - A service provider returned a version number other than 2.0.	
16#00002733 10035	Resource temporarily unavailable - Operation should be retried later.	
16#00000006 6	Specified event object handle is invalid - An application attempts to use an event object, but the specified handle is not valid.	
16#00000008 8	Insufficient memory available - The Win32 Socket function is indicating a lack of required memory resources.	

Error Code	Meaning	Solution suggested
16#00000057 87	One or more parameters are invalid - The Win32 Socket function is indicating a problem with one or more parameters.	
16#000003e3 995	Overlapped operation aborted - An overlapped operation was canceled due to the closure of the socket.	
16#000003e4 996	Overlapped I/O event object not in signaled state - The application has tried to determine the status of an overlapped operation which is not yet completed.	
16#000003e5 997	Overlapped operations will complete later - The application has initiated an overlapped operation which cannot be completed immediately.	

24 Characteristics of TCP/IP

24.1 Number of TCP/IP connections possible

Note

The number of TCP/IP connections possible in ibaLogic depends on the system settings "TcpNumConnections".

An excerpt of Microsoft for this:

"**TcpNumConnections**

Registry:

`HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
Keyname: TcpNumConnections

Data type	Range	Default value
REG_DWORD	0x0 – 0xFFFFFE	0

Description

Determines the maximum number of connections that TCP can have open simultaneously. If the value of this entry is 0, you cannot establish any connections.

Note Image Note

Windows does not add this entry to the registry. You can add it by editing the registry or by using a program that edits the registry."

LINKS

TcpNumConnections (*http://technet.microsoft.com/en-us/library/cc938216.aspx*)

TCP/IP - Maximum *number* (*http://www.windowspage.de/tipps/021202.html*) of simultaneously open connections

24.2 Delayed Acknowledge Problem

Problem

Measurements made by automation equipment using TCP/IP do not work with cycle times < 200 ms.

Error pattern: Sequence error, incomplete telegrams and different lengths received.

Cause

There are different variants of handling 'Acknowledge' in the TCP/IP protocol:

1. The standard WinSocket works in accordance with RFC1122 using the "delayed acknowledge" mechanism. This states that the acknowledge is delayed until other telegrams arrive in order to acknowledge them jointly. If no other telegrams arrive, the ACK telegram is sent latest after 200 ms (depending on the socket).

   According to the TCP/IP standard this is possible since with data flow control using

"Sliding Window" (Parameter Win = nnnn) the receiver specifies the number of bytes that it can receive without sending an acknowledgment.

2. Some controllers do not accept this response, but instead, wait for an acknowledgment after each data telegram. If this does not arrive within a specific time period (200 ms), it repeats the telegram and possibly also adds new data to be transmitted. This leads to an error in the receiver, since the older telegram was received correctly.

Remedy

The "delayed acknowledge" must be disabled in Windows with the help of the parameter entry in the Windows registry:

"TcpAckFrequency" REG_DWORD = 1;

The parameter is not present by default and has to be entered in this path:

"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{InterfaceGUID}"

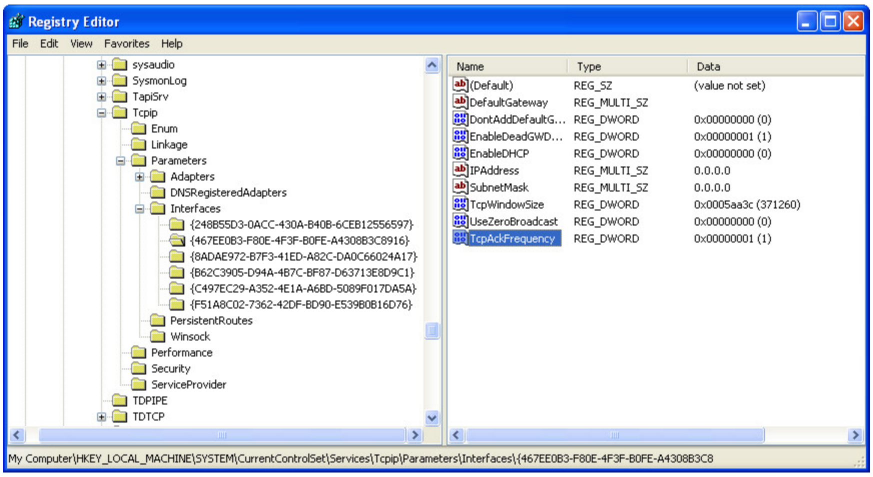
Note

You must select the correct interface. You can see which one is the correct one, for example, from the IP addresses set currently.

See also the following MS

New *registration entry* (*http://support.microsoft.com/kb/328890*) for checking the TCP confirmation response in Windows XP and Windows Server 2003

Windows XP:



```
⊞ sysaudio
⊞ SysmonLog
⊞ TapiSrv
⊟ Tcpip
    Enum
    Linkage
  ⊟ Parameters
    ⊞ Adapters
      DNSRegisteredAdapters
    ⊟ Interfaces
        {248B55D3-0ACC-430A-B40B-6CEB12556597}
        {467EE0B3-F80E-4F3F-B0FE-A4308B3C8916}
        {8ADAE972-B7F3-41ED-A82C-DA0C66024A17}
        {B62C3905-D94A-4B7C-BF87-D63713E8D9C1}
        {C497EC29-A352-4E1A-A6BD-5089F017DA5A}
        {F51A8C02-7362-42DF-BD90-E539B0B16D76}
      PersistentRoutes
      Winsock
    Performance
    Security
    ServiceProvider
  TDPIPE
⊞ TDTCP
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
DefaultGateway	REG_MULTI_SZ	
DontAddDefaultG...	REG_DWORD	0x00000000 (0)
EnableDeadGWD...	REG_DWORD	0x00000001 (1)
EnableDHCP	REG_DWORD	0x00000000 (0)
IPAddress	REG_MULTI_SZ	0.0.0.0
SubnetMask	REG_MULTI_SZ	0.0.0.0
TcpWindowSize	REG_DWORD	0x0005aa3c (371260)
UseZeroBroadcast	REG_DWORD	0x00000000 (0)
TcpAckFrequency	REG_DWORD	0x00000001 (1)

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{467EE0B3-F80E-4F3F-B0FE-A4308B3C8

Figure 177: Windows XP Registry

25 Key Combinations

25.1 Client

Key combination	Explanation
<Ctrl> + <P>	Print
<Ctrl> + <Z>	Undo
<Ctrl> + <Y>	Undoes <Ctrl> + <Z>
<Ctrl> + <C>	Copy
<Ctrl> + <V>	Paste
<Ctrl> + <A>	Select all
<Del>	Delete
<F5>	Start evaluation
<Shift> + <F5>	Stop evaluation
<Ctrl> + <Shift> + <F>	Add – new function block
<Ctrl> + <Shift> + <M>	Add – new macro block
<Ctrl> + <Shift> + <I>	Add – new intra-page connector
<Ctrl> + <Shift> + <T>	Add – new off-task connector
<Ctrl> + <Shift> + <C>	Add – new comment
<Ctrl> + <Shift> + <S>	Display off-task connectors
<F1>	Help
Arrow keys	Move marked blocks, etc. in the arrow direction

25.2 Mouse Functions in the Programming Field

Key combination	Explanation
Left mouse button click + <Shift>:	Mark multiple elements
Left mouse button click + <Ctrl>	Switch the marking
Scroll wheel + <Shift>:	Move visible section to the left / right.
Scroll wheel + <Ctrl>:	Zoom in / zoom out
Scroll wheel + <Alt>	Move visible section up / down.
<Ctrl> + connector input or output	Create IPC
Drag & Drop + <Alt>	You can drag signals by keeping the <Alt> button pressed from a connector and drop them into the ibaPDA Express window.

25.3 ibaPDA Express

Key combination	Explanation
<F6> (Switch)	Starts continuous display with the current time point. Active, when "Stop scrolling" is pressed.
<F6> (Switch)	Stop the continuous display. After pressing this, a ruler appears in graphs that can be moved with the mouse and with which the curves can be measured. The signal values are displayed in the legend. You can move the X-axis using the mouse. In this manner, you can browse values from the past.

Key combination	Explanation
	Active, when the display is on.
<F5>	"Auto scale"
<F3>	Active only when the display has been zoomed. Return to the previous zoom factor (reduce).
<F4>	Active only when the display has been zoomed. Return to the initial (automatic) display.
<F10>	Exit from the full screen mode.

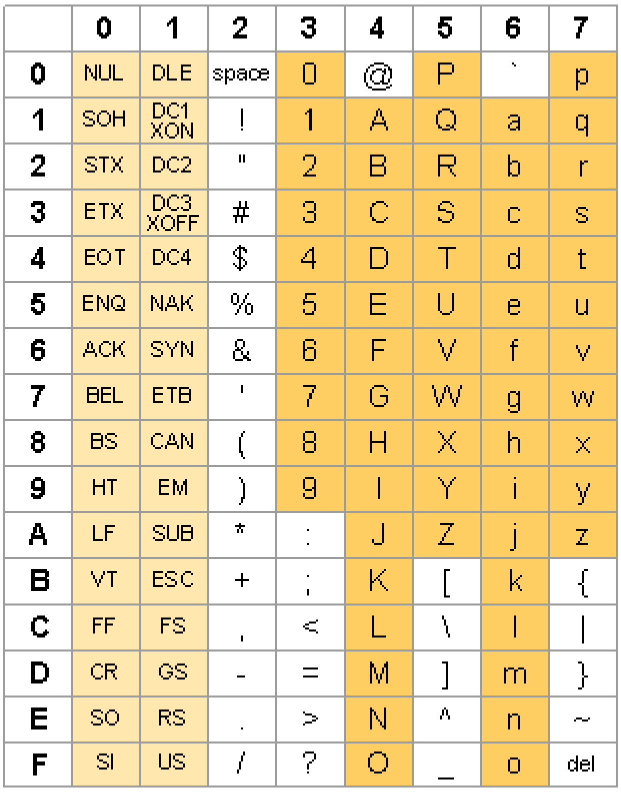
26 Character tables

ibaLogic uses simplified Hex coding.
Thus, the first 256 Unicode characters (U+0000 to U+00FF) can be displayed.

Note

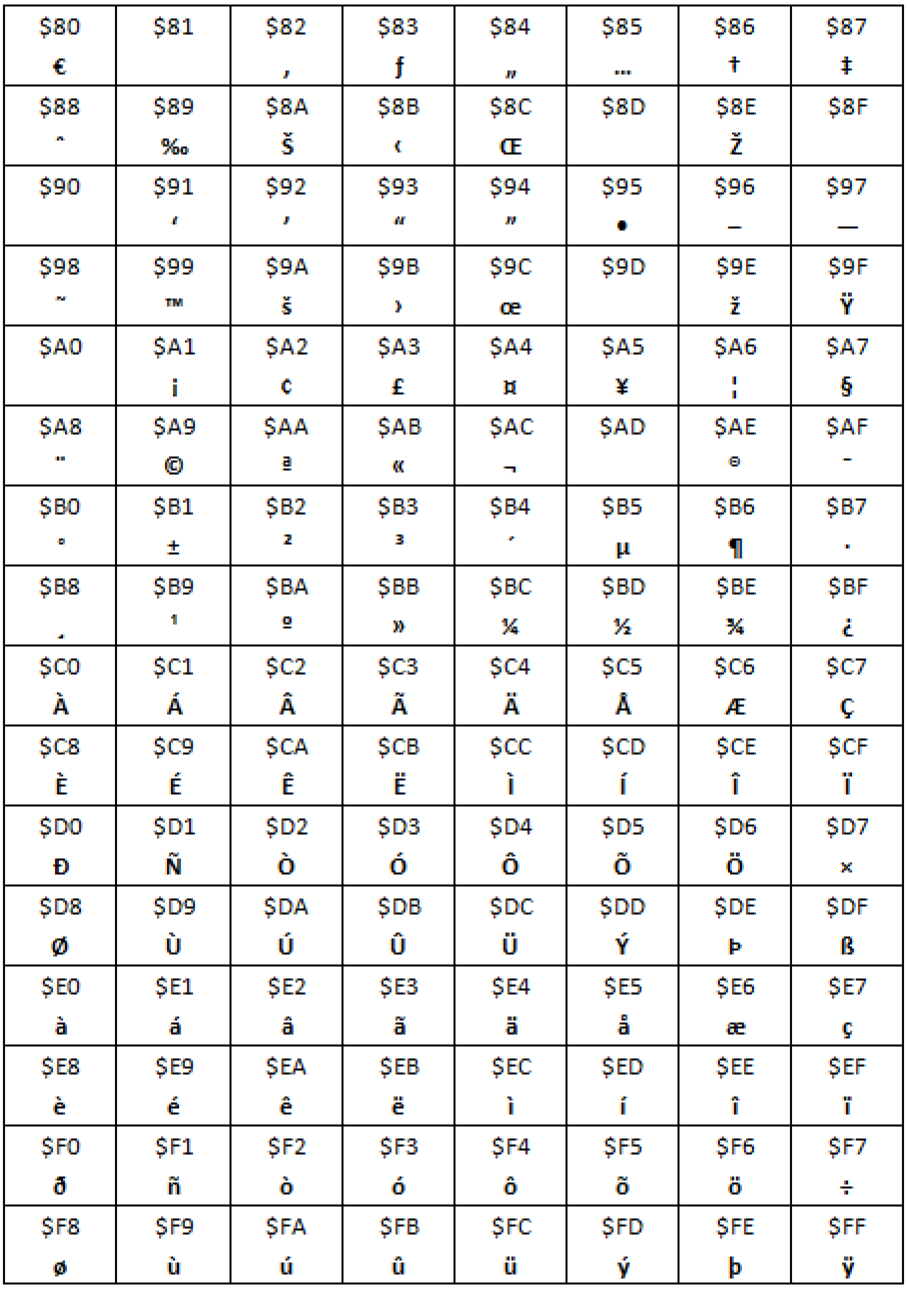
ibaLogic uses regional conversions, as they are used for displays, also for regionally specific special characters like e.g. the German *Ä*. Thus, the results on diffferent machines can differ, depending on the regional settings.

Standard ASCII character table ($00 - $7F)



	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Extended character table ($80 - $FF)



$80 €	$81	$82 ‚	$83 ƒ	$84 „	$85 …	$86 †	$87 ‡
$88 ˆ	$89 ‰	$8A Š	$8B ‹	$8C Œ	$8D	$8E Ž	$8F
$90	$91 ‘	$92 ’	$93 “	$94 ”	$95 •	$96 –	$97 —
$98 ˜	$99 ™	$9A š	$9B ›	$9C œ	$9D	$9E ž	$9F Ÿ
$A0	$A1 ¡	$A2 ¢	$A3 £	$A4 ¤	$A5 ¥	$A6 ¦	$A7 §
$A8 ¨	$A9 ©	$AA ª	$AB «	$AC ¬	$AD	$AE ®	$AF ¯
$B0 °	$B1 ±	$B2 ²	$B3 ³	$B4 ´	$B5 µ	$B6 ¶	$B7 ·
$B8 ¸	$B9 ¹	$BA º	$BB »	$BC ¼	$BD ½	$BE ¾	$BF ¿
$C0 À	$C1 Á	$C2 Â	$C3 Ã	$C4 Ä	$C5 Å	$C6 Æ	$C7 Ç
$C8 È	$C9 É	$CA Ê	$CB Ë	$CC Ì	$CD Í	$CE Î	$CF Ï
$D0 Ð	$D1 Ñ	$D2 Ò	$D3 Ó	$D4 Ô	$D5 Õ	$D6 Ö	$D7 ×
$D8 Ø	$D9 Ù	$DA Ú	$DB Û	$DC Ü	$DD Ý	$DE Þ	$DF ß
$E0 à	$E1 á	$E2 â	$E3 ã	$E4 ä	$E5 å	$E6 æ	$E7 ç
$E8 è	$E9 é	$EA ê	$EB ë	$EC ì	$ED í	$EE î	$EF ï
$F0 ð	$F1 ñ	$F2 ò	$F3 ó	$F4 ô	$F5 õ	$F6 ö	$F7 ÷
$F8 ø	$F9 ù	$FA ú	$FB û	$FC ü	$FD ý	$FE þ	$FF ÿ

27 Index of Abbreviations

Abbreviation	German	English
ASCII	Zeichenkodierung	American Standard Code for Information Interchange
AWL	Anweisungsliste	Statement List
CE	Übereinstimmung mit EU-Richtlinien	FR.: Conformité Européenne
CFC	Funktionsblockdiagramm	Continuous Function Chart
CPU	Prozessor	Central Processing Unit
CR	Wagenrücklauf	Carriage Return
CSV		Comma Separated Values or Character Separated Values
DA	Datenzugriff	Data Access
DCOM		Distributed Component Object Model
DFW		DAT_FILE_WRITE
DIN	Deutsches Institut für Normung	
DLL	Dynamische Verbindungsbibliothek	Dynamic Link Library
E/A	Eingang/Ausgang	INPUT/OUTPUT
FB	Funktionsbaustein	Function Block
FBD	Funktionsblockdiagramm	Function Block Diagram
FFT		Fast Fourier Transformation
FOB	Lichtwellenleiterkarte	Fiber optical board
FUP	Funktionsplan	Function Chart
GDM		Global Data Memory
GMT		Greenwich Mean Time
GSD	Gerätestammdaten-Datei (Profibus)	Generic Station Description
HMI	Bedien-Beobachten-System	Human Machine Interface
HW	Hardware	Hardware
I/O	Eingang/Ausgang	Input/Output
IEC	ein Normungsgremium für Elektrotechnik	International Electrotechnical Commission
IL	Anweisungsliste	Instruction List
IP	Internet-Protokoll	Internet Protocol
IPC	Intra-Page-Konnektor	Intra-page connector
KOP	Kontaktplan	
LAD		Ladder Diagram
LF	Zeilenvorschub	Line Feed
FOC	Lichtwellenleiter	Fiber optical conductor
MB	Makroblock	Macro Block
MDAC		Microsoft Data Access Components
MS SQL		Microsoft Structured Query Language
NL	Zeilenvorschub	Newline
OLE	Objekt-Verknüpfung und -Einbettung	Object linking and embedding
OPC	Datenaustauschprotokoll (Schnittstelle)	OLE for Process Control
OTC	Off-Task-Konnektor	Off Task Connector
PAC	Programmierbare Automatisierungseinheit	Programmable automation controller

Abbreviation	German	English
PADU	Parallel-Analog-Digital-Umsetzer (Varianten: PADU-S, PADU-S-IT)	Parallel analog digital unit
PC	Einzelplatzrechner	Personal Computer
PCI	PC-Bussystem	Peripheral Component Interconnect
PDA	Prozess-Daten-Aufzeichnung	Process data acquisition
PLC	Siehe SPS	Programmable Logic Controller
PMAC	Programmierbare Mess- und Automatisierungseinheit	Programmable Measurement and Automation Controller
RAM	Arbeitsspeicher	Random Access Memory
RFM		Reflective Memory
SD	SIMADYN D	SIMADYN D
SPS	Speicherprogrammierbare Steuerung	See PLC
SST	Profibuskarte	
ST	Strukturierter Text	Structured Text
STL	Anweisungsliste	Statement List
SW	Software	Software
Tab	Tabulator	Tabulator
TCP/IP	Netzwerk-Protokoll	Transmission Control Protocol/Internet Protocol
TDC	Automatisierungssystem (Siemens)	
UCODE	Bytecode	
USB		Universal Serial Bus
UTC	koordinierte Weltzeit	Universal Time Coordinated
WDM		Windows Driver Model
XML		Extensible Markup Language
XP	Windows XP	Windows XP

28 Classified Index

29 Support and contact

Support

Phone:	+49 911 97282-14
Fax:	+49 911 97282-33
Email:	support@iba-ag.com

Note

If you require support, indicate the serial number (iba-S/N) of the product.

Contact

Headquarters

iba AG
Koenigswarterstr. 44
90762 Fuerth
Germany

Phone:	+49 911 97282-0
Fax:	+49 911 97282-33
Email:	iba@iba-ag.com
Contact:	Mr Harald Opel

Regional and worldwide

For contact data of your regional iba office or representative please refer to our web site

www.iba-ag.com.