ibaLogic-V4

Handbuch

Ausgabe 4.2.4

Messtechnik- und Automatisierungssysteme

Hersteller

iba AG

Königswarterstr. 44

90762 Fürth

Deutschland

Kontakte

Zentrale	+49 911 97282-0
Telefax	+49 911 97282-33
Support	+49 911 97282-14
Technik	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

Der Inhalt dieser Druckschrift wurde auf Übereinstimmung mit der beschriebenen Hard- und Software überprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass für die vollständige Übereinstimmung keine Garantie übernommen werden kann. Die Angaben in dieser Druckschrift werden jedoch regelmäßig aktualisiert. Notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten oder können über das Internet heruntergeladen werden.

Die aktuelle Version liegt auf unserer Website www.iba-ag.com zum Download bereit.

Schutzvermerk

Windows® ist eine Marke und eingetragenes Warenzeichen der Microsoft Corporation. Andere in diesem Handbuch erwähnte Produkt- und Firmennamen können Marken oder Handelsnamen der jeweiligen Eigentümer sein.

Version	Datum	Revision - Kapitel / Seite	Autor	Version SW
4.2.4	10.01.2013	Update Software	KF	4.2.4

Inhaltsverzeichnis

1 Zu diesem Handbuch

Dieses Handbuch bzw. diese Online-Hilfe beschreibt die Funktion, den Aufbau und die Anwendung der Software ibaLogic-V4.

1.1 Zielgruppe

Im Besonderen wendet sich dieses Handbuch an ausgebildete Fachkräfte, die mit dem Umgang mit elektrischen und elektronischen Baugruppen sowie der Kommunikations- und Messtechnik vertraut sind. Als Fachkraft gilt, wer auf Grund seiner fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Bestimmungen die ihm übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

1.2 Schreibweisen

In dieser Dokumentation werden folgende Schreibweisen verwendet:

Aktion	Schreibweise
Menübefehle	Menü „Funktionsplan“
Aufruf von Menübefehlen	“Schritt 1 – Schritt 2 – Schritt 3 – Schritt x” Beispiel: Wählen Sie Menü „Funktionsplan – Hinzufügen – Neuer Funktionsblock”
Tastaturtasten	<Tastenname> Beispiel: <Alt>; <F1>
Tastaturtasten gleichzeitig drücken	<Tastenname> + <Tastenname> Beispiel: <Alt> + <Strg>
Grafische Tasten (Buttons)	<Tastenname> Beispiel: <OK>; <Abbrechen>
Dateinamen, Pfade	"Dateiname", "Pfad" Beispiel: "Test.doc"

1.3 Verwendete Symbole

Wenn in dieser Dokumentation Sicherheitshinweise oder andere Hinweise verwendet werden, dann bedeuten diese:

Gefahr! Stromschlag!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die unmittelbare Gefahr des Todes oder schwerer Körperverletzung durch einen Stromschlag!

Gefahr!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die unmittelbare Gefahr des Todes oder der schweren Körperverletzung!

Warnung!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr des Todes oder schwerer Körperverletzung!

Vorsicht!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr der Körperverletzung oder des Sachschadens!

Hinweis

Hinweis, wenn es etwas Besonderes zu beachten gibt, wie z.B. Ausnahmen von der Regel usw.

Wichtiger Hinweis

Hinweis, wenn etwas Besonderes zu beachten ist, z . B. Ausnahmen von der Regel.

Tipp

Tipp oder Beispiel als hilfreicher Hinweis oder Griff in die Trickkiste, um sich die Arbeit ein wenig zu erleichtern.

Andere Dokumentation

Verweis auf ergänzende Dokumentation oder weiterführende Literatur.

2 Einführung

2.1 Identifikation

PAC (Soft-SPS) und Signalmanager „ibaLogic-V4".

2.2 Bestimmungsgemäßer Gebrauch

Das Produkt/System wird eingesetzt zum Messen und Steuern von technischen Anlagen.

ibaLogic ist nicht für sicherheitsgerichtete Systeme vorgesehen.

Eine andere oder erweiterte Nutzung des Produkts/Systems gilt als nicht bestimmungsgemäß und damit sachwidrig. In diesem Fall können die Sicherheit und der Schutz des Produkts/Systems beeinträchtigt werden. Für hieraus entstehende Schäden haftet das Unternehmen iba AG nicht.

Gefahr!

Gefahr durch Aktivierung von Funktionen oder weiteren Diensten!

Mögliche Personen- und Maschinenschäden durch Aktivierung von Funktionen und weiteren Diensten (PMAC, OPC …), die sich direkt auf das Verhalten der Anlage auswirken.

Sichern Sie die Anlage beim Arbeiten mit dem System ab! Beachten Sie geltende Sicherheitsvorschriften!

2.3 Freigabehinweise - Release Notes

2.3.1 Änderungshistorie-Datei

Eine Änderungshistorie-Datei (changelog.htm) zu Ihrer Software steht auf dem Installationsdatenträger zur Verfügung. Darin finden Sie u. a. wichtige Informationen zu folgenden Themen:

- ❑ Neue Funktionen
- ❑ Behobene Fehler

3 Software-Installation

3.1 Systemvoraussetzungen

3.1.1 Hardware

In der folgenden Tabelle sind die Hardware-Voraussetzungen aufgelistet.

	Mindestanforderung	Empfohlen oder höher
CPU-Geschwindigkeit	1600 MHz	2000 MHz
Anzahl CPUs	1	2
RAM	768 MByte	2048 MByte
Bildschirmauflösung	1024 x 768	1280 x 1024

Der minimale Speicherplatzbedarf sind ca. 650 MB. Eine SQL-Server Express-Datenbank kann bis auf maximal 4 GB anwachsen. Halten Sie genügend freien Speicherplatz für Ihre Anforderungen bereit.
Weitere Informationen siehe „Leistungsgrenzen , Seite 249".

Hinweis

Bei unterschreiten der Mindestanforderungen ist eine Installation möglich. Es können jedoch Leistungseinschränkungen auftreten.

3.1.2 Software

Eines der folgenden Betriebssysteme muss vorinstalliert sein:

- ❑ Windows XP Professional SP3
- ❑ Windows 2003 Server
- ❑ Windows 7 SP1 32bit

Hinweis

Sowohl für die Installation als auch für den Betrieb von ibaLogic-Server und -Client sind Administrator-Rechte notwendig.

Die aufgezählten Software-Pakete sind Bestandteil der Liefer-CD:

- ❑ Windows Installer 3.1
- ❑ MDAC 2.81
- ❑ .Net Framework 2.0 SP1
- ❑ MS SQL Server Express 2005 (9.0)
- ❑ OPC Core Components 2.0
- ❑ ibaWDM Treiber
- ❑ CB-USB Dongle Treiber
- ❑ Visual J# 2.0

Der Installationsassistent prüft, ob die Versionsstände der einzelnen Softwarepakete vorhanden sind. Sofern Softwarepakete fehlen oder veraltet sind, werden diese durch den Installationsassistenten installiert oder durch ein Update aktualisiert.

3.2 Lizenzaktivierung

Bei Lieferung ist der Dongle bereits kundenspezifisch eingerichtet. Der Kunden-Dongle generiert im System einen virtuellen Schlüssel, der die individuellen Funktionen freischaltet.

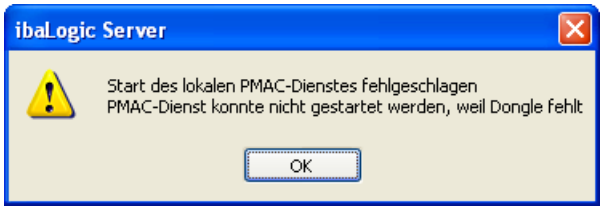
Vorsicht!

Gefahr durch Abschaltung des Laufzeitsystems PMAC nach Entfernen des Dongles!

Ohne den Dongle mit zugehöriger Lizenz kann das System nicht in Betrieb genommen werden. Die Lizenz bestimmt die Freischaltung von Funktionen.
Lassen Sie daher den Dongle während des gesamten Betriebes gesteckt!

Wird der Dongle während des Betriebes entfernt, schaltet sich der PMAC (Programmable Measurement and Automation Controller) nach mehreren Warnungen (ca. 5 min nach der ersten Warnung) ab.

Ohne Dongle startet der PMAC nicht. Stattdessen kann in den ibaLogic Server Optionen eine alternative Demo-Version des PMACs aktiviert werden (siehe "Einstellungen für den lokalen PMAC , Seite 46"). Wenn der PMAC manuell oder automatisch vom ibaLogic Server gestartet wird, erscheint eine Warnmeldung, dass kein Dongle gesteckt ist.



Die Demo-Version unterstützt keinen Hardwarezugriff, stattdessen werden etliche Geräte simuliert. Hierbei werden, soweit dies möglich ist, die verwendeten Ausgänge direkt auf die Eingänge rückgekoppelt (nicht möglich z.B. bei gepufferten Eingängen.)

Einige Funktionsblöcke sind deaktiviert, d.h. sie sind im Plan zwar projektierbar, werden aber nicht berechnet, dazu gehören u.a.: TCPIP_SendRecv, DatFileWrite, User-DLLs.

Abbildung 1: Dongle

Vorgehen

➲ Schließen Sie den Dongle an eine USB-Schnittstelle an.

Nach dem Starten des Servers und des Clients erscheint in der Konsolenansicht folgende Meldung: „Online Server: DriverStatus: Driver running for Dongle *Vxxxxxx*".

Hinweis

Diese Meldung erscheint nur wenn ein Projekt gestartet wurde.

3.3 Installieren der Software

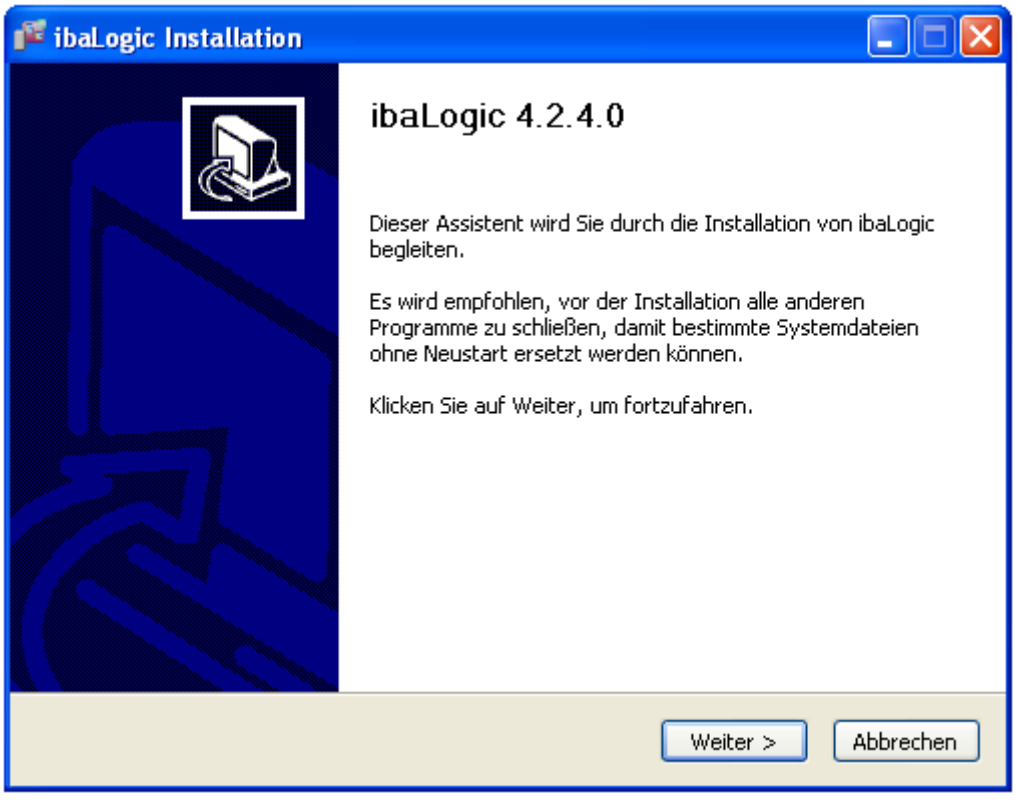
Zum Installieren der ibaLogic-Software folgen Sie den Anweisungen des Installations-Assistenten.

3.3.1 Voraussetzung

❑ Ihr System erfüllt die Voraussetzungen der Hard- und Software.

3.3.2 Vorgehen

➲ Doppelklicken Sie die Datei „Setup-4.*x.xx*.exe".

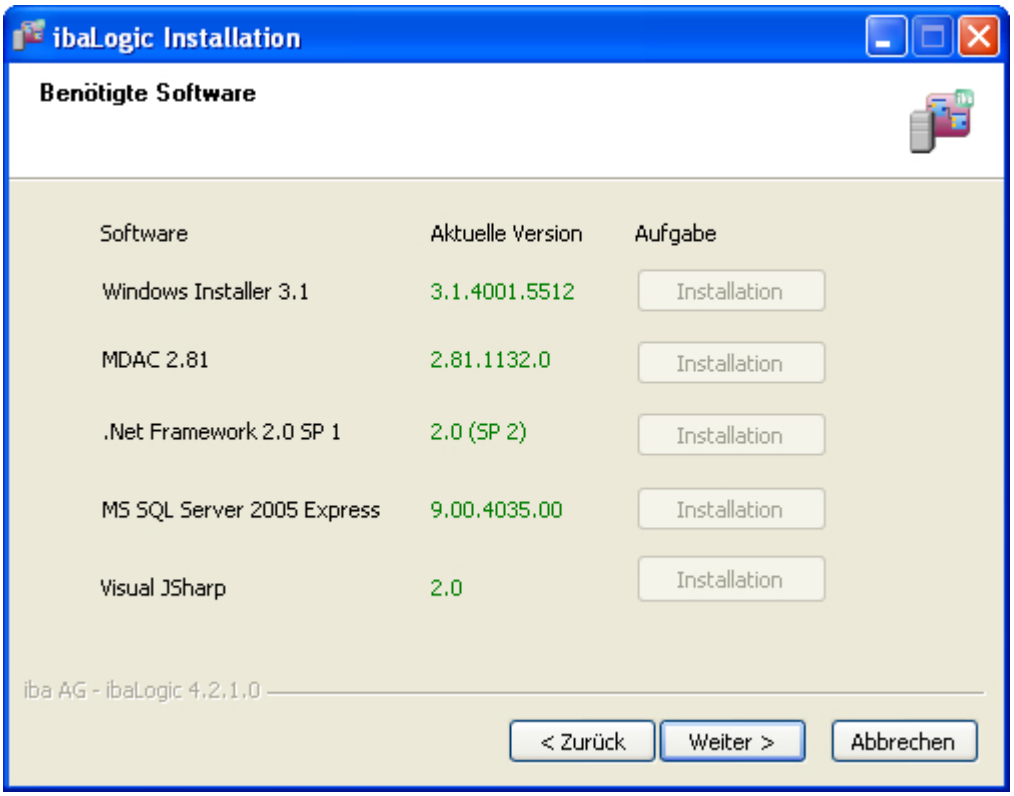


3.3.3 Benötigte Software

Unter „Software" werden die notwendigen Komponenten und Versionen angegeben.

Darstellung	Erklärung
Versionsnummer grün MS SQL Server 2005 Express 9.00.4035.00 Installation	Software ist installiert und Funktionsumfang ist ausreichend.
Versionsnummer rot MS SQL Server 2005 Express 0.0 Installation	Software ist nicht installiert oder nicht ausreichend. Der zugehörige Button <Installation> ist aktiviert.

➲ Führen Sie eine Installation bzw. Aktualisierung der Software-Komponente(n) durch, indem Sie auf den aktivierten Button <Installation> klicken.



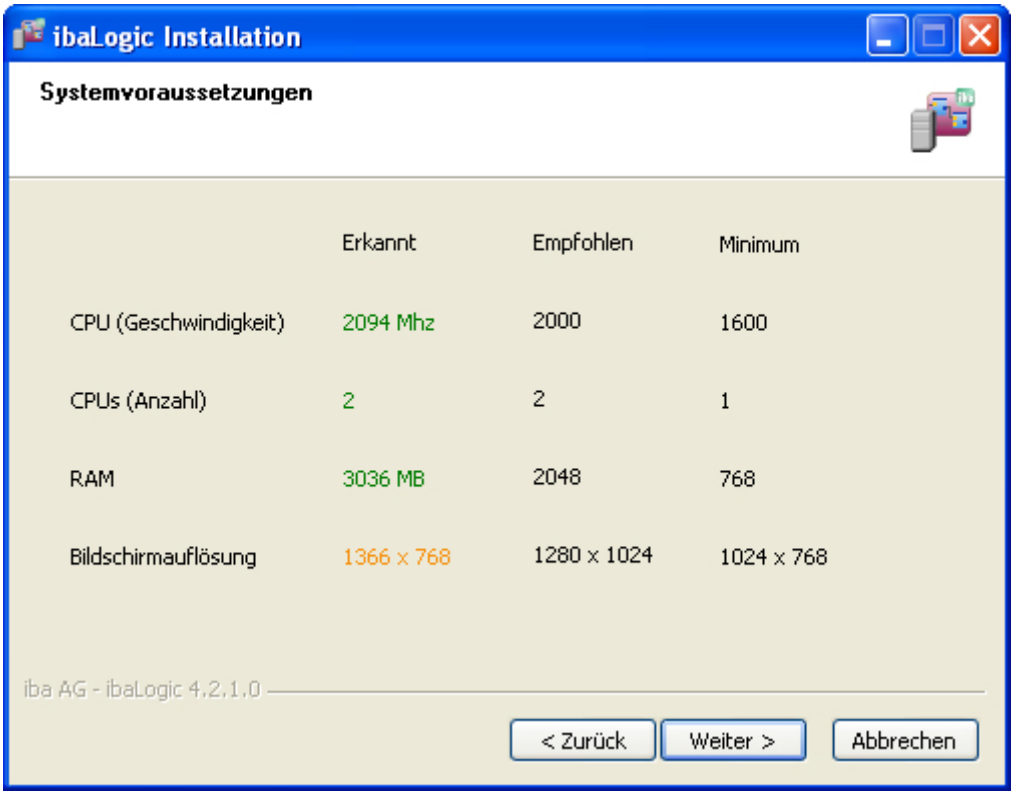
➲ Bestätigen Sie ggf. aufkommende Dialogfenster. Sind alle Software-Komponenten installiert bzw. aktualisiert, klicken Sie auf <Weiter>.

3.3.4 Systemvoraussetzungen

Vor der Installation der Softwarekomponenten erfolgt eine Überprüfung der Systemvoraussetzungen.

Darstellung	Erklärung
Grün	Empfohlene Anforderungen werden erfüllt.
Orange	Mindestanforderungen werden erfüllt.
Rot	Minimumanforderungen werden nicht erfüllt.

➲ Wenn die Systemvoraussetzungen die Mindestvoraussetzungen erfüllen, setzen Sie die Installation fort.

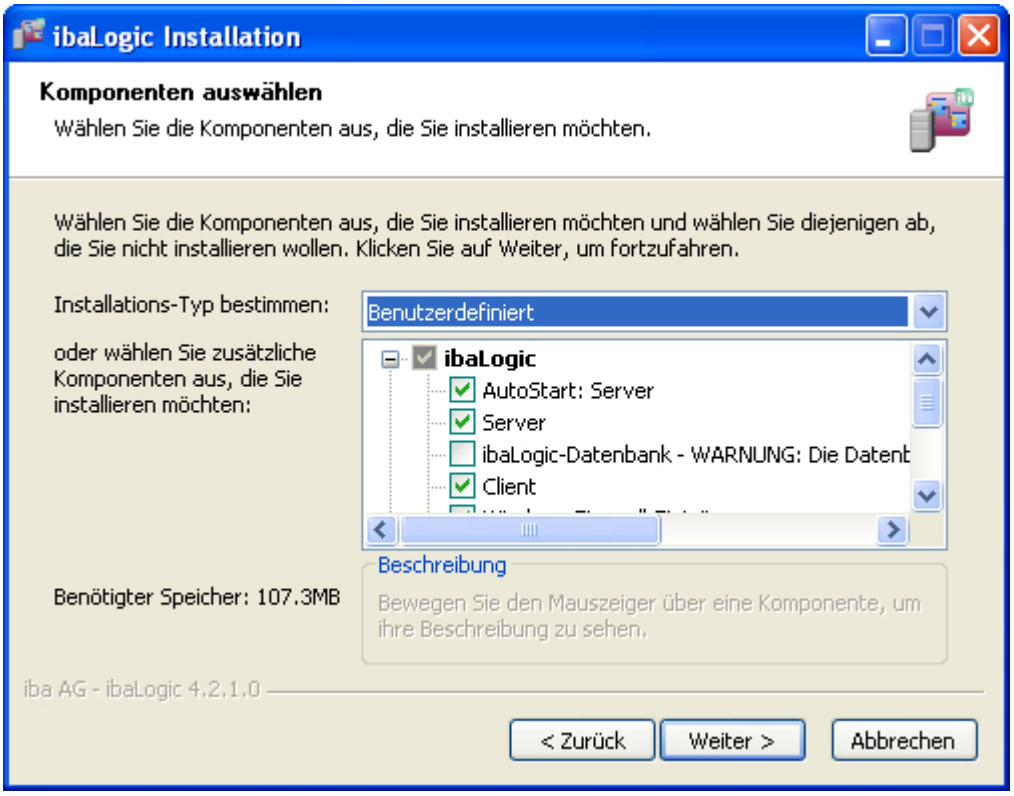


3.3.5 Komponenten auswählen

Mittels des Auswahlfelds „Installations-Typ" wird eine Auswahl von Komponenten vor der Installation ermöglicht.

Installations-Typ	Erklärung
Vollständig	Alle Komponenten werden installiert.
Nur Server	Nur die ibaLogic Server Komponenten werden installiert.
Nur Client	Es wird nur der ibaLogic Client installiert.
Benutzerdefiniert	Eine Auswahl der Komponenten ist möglich. Es werden nur die ausgewählten Komponenten installiert.

➲ Bestimmen Sie die ibaLogic-Komponenten durch Auswahl des Installations-Typs.



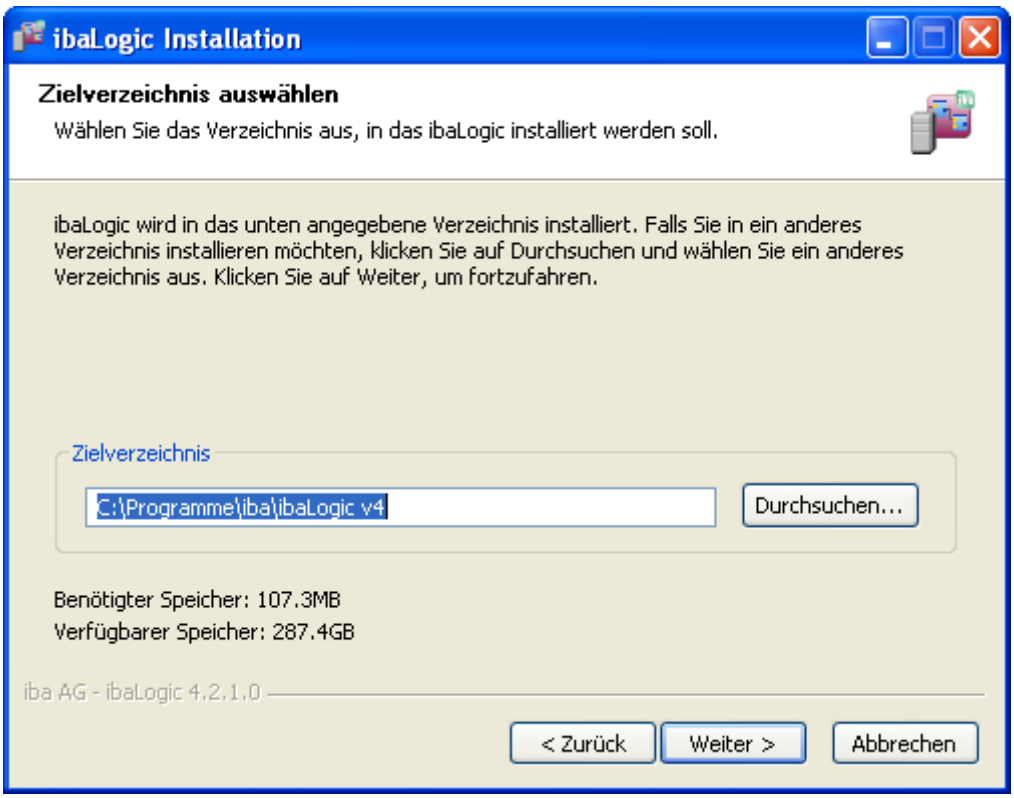
Vorsicht!

Bei einem Update ist die Option „ibaLogic-Datenbank" nicht gesetzt. Mit dem Setzen der Option wird im weiteren Verlauf der Installation die vorhandene Datenbank inklusive ihrer Projekte gelöscht. Eine Warnung „Die Datenbank existiert bereits und wird überschrieben!" wird angezeigt.

3.3.6 Zielverzeichnis auswählen

In diesem Verzeichnis wird die ibaLogic-Ordnerstruktur (Server, Client usw.) angelegt. iba empfiehlt die Vorgabe beizubehalten.

➲ Legen Sie das Zielverzeichnis fest.



3.3.7 SQL-Server-Auswahl

Wurde bei der Installation das Auswahlfeld "ibaLogic-Datenbank" aktiviert, sucht der Installer während der Installation auf dem lokalen Rechner nach Microsoft SQL-Servern und bietet Ihnen den Dialog "SQL Server Auswahl" zur Selektion der Datenbank-Instanz an.

➲ Wählen Sie die Default-Instanz „<PC-Name>\IBA" oder eine Instanz Ihrer Wahl aus und verlassen Sie den Dialog mit dem Button <OK>.
Die ausgewählte ibaLogic-Datenbank wird auf dem ausgewählten Server installiert.

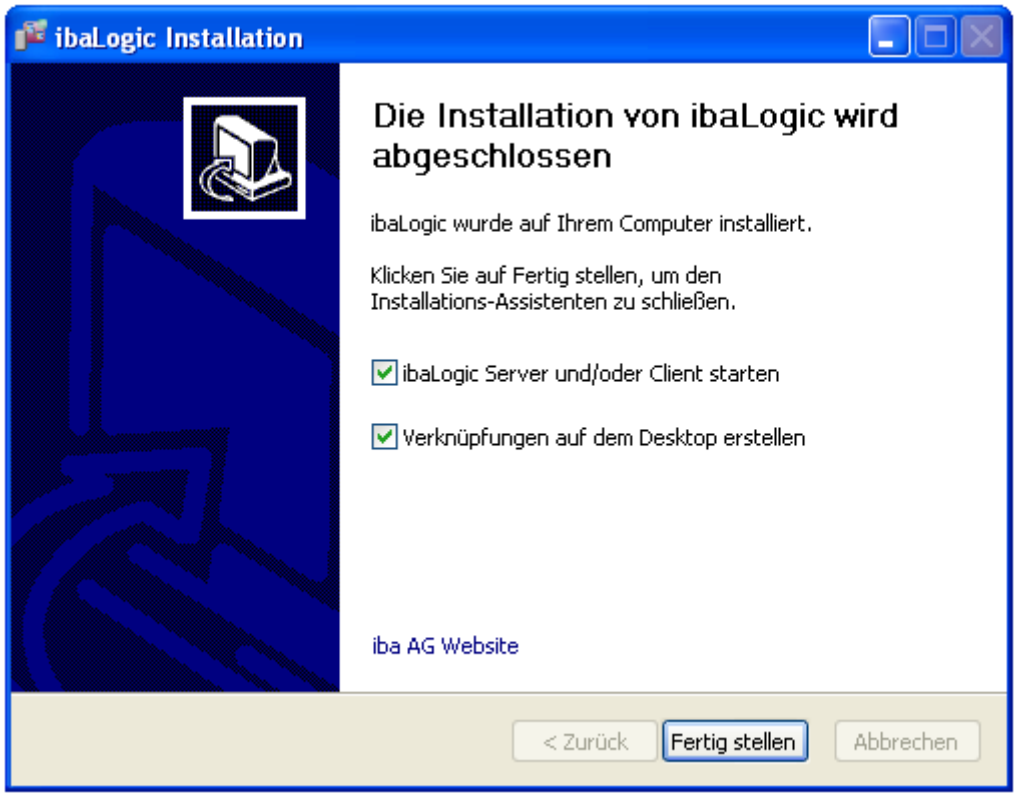
Hinweis

Ein Überschreiben einer bestehenden ibaLogic-Datenbank ist nur nach Bestätigen eines aufkommenden Dialogs möglich. Wenn Sie diesen Dialog mit dem Button <Nein> verlassen, dann bleibt Ihre bisherige Datenbank erhalten.

3.3.8 ibaLogic-Installation abschließen

ibaLogic-Installation abschließen

➲ Klicken Sie auf den Button <Fertig stellen>, um die Installation abzuschließen.



4 ibaLogic-Software

4.1 Einführung

Die iba AG hat sich bereits vor Jahren auf das Erfassen von Messwerten aus Anlagen der Schwerindustrie spezialisiert. Es handelt sich dabei schwerpunktmäßig um Werke zur Erzeugung und Verarbeitung von Stahl und Nicht-Eisen-Metallen.

Die Programme zur Erfassung[1] und zur Analyse[2] der aufgezeichneten Daten sind heute weltweit im Einsatz und werden von allen großen Ausrüstern der Maschinen- und Automatisierungstechnik weltweit eingesetzt.

Durch die breit gefächerten Anbindungsmöglichkeiten der iba-Messtechnik an unterschiedlichste Automatisierungstechnologien und -generationen, speziell auch an die gängigsten Feld- und Antriebsbusse, entwickelte sich bald ein Bedarf zur Verbindung dieser zum Teil sehr unterschiedlichen Welten. Aus dem „Einrichtungsverkehr" Messen mussten nunmehr im „Zweirichtungsverkehr" Informationen aus verschiedenen Automatisierungswelten ausgetauscht werden – typisch für den aufkommenden Markt der Modernisierungen älterer bereits automatisierter Anlagen (Revamp).

Um diesen Bedarf abzudecken, entwickelte die iba AG bereits seit 1995 einen frei programmierbaren Signalmanager. Die bereits in dieser Zeit entstandene Norm IEC 61131-3 für die Beschreibung technischer Abläufe mit Hilfe von grafischen Elementen und einfachen eingebetteten Programmiertechniken, erleichtert das Beschreiben komplexer Signalbehandlungen wesentlich.

Heute ist die aus dieser Norm abgeleitete grafische Programmierung[3] die Basis nahezu aller Automatisierungssysteme. Die grafische Programmierung ist damit kompatibel und damit in weiten Teilen portierbar.

Merkmale:

- ❑ Onlinechange
- ❑ ständige Projektsicherung
- ❑ Eingabeprüfung on the fly
- ❑ Visualisierungs- und Tracetool (ibaPDAExpress)

[1] ibaPDA
[2] ibaAnalyzer
[3] FBD ist eine graphische Programmiersprache, in der Funktionsblöcke miteinander verschaltet werden, anstatt eine Abfolge von textuellen Befehlen einzugeben, wie bei klassischen Programmiersprachen. Als Vorbild sind dabei Schaltpläne aus der Hardware-Entwicklung zu sehen. Diese Darstellung eines Programms kommt Entwicklern von Steuerungs-Software entgegen, deren technischer Hintergrund typischerweise eher aus der Elektrotechnik stammt. Die einzelnen Funktionsblöcke sind selbst oft in anderen SPS-Sprachen, wie z. B. "Structured Text" verfasst und können vom Hersteller des Automatisierungssystems als Standardbausteine mitgeliefert oder vom Anwender selbst verfasst bzw.importiert werden.

4.2 Anwendungsbereiche

ibaLogic wird für folgende Anwendungen genutzt:

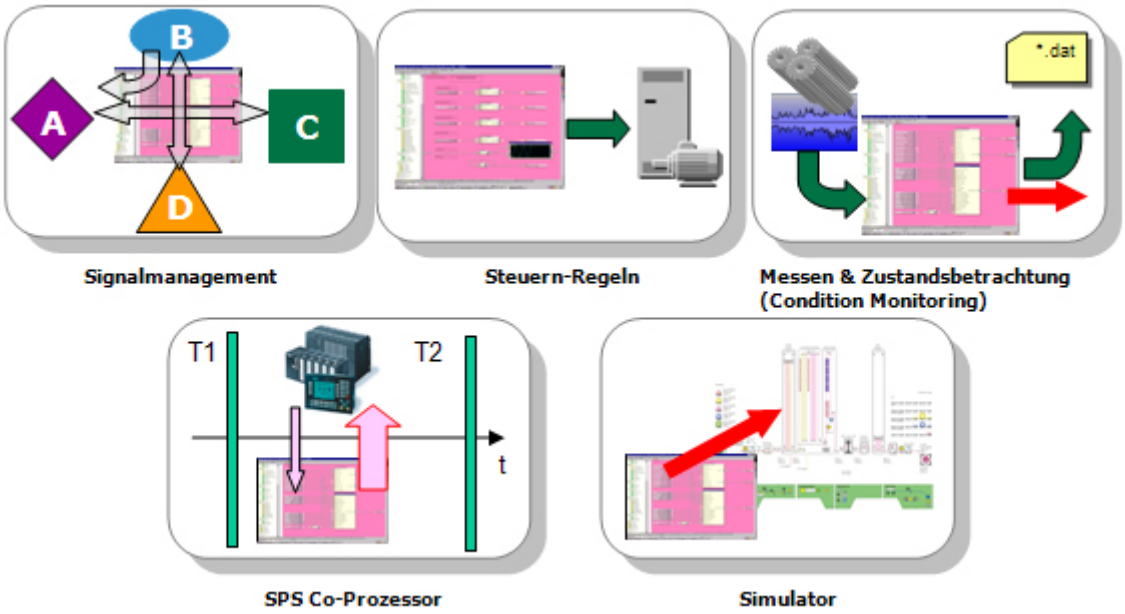


Abbildung 2: Anwendungsbereiche

Signalmanagement

Über die iba-Konnektivität können Verbindungen verschiedenster Generationen von Automatisierungswelten der unterschiedlichsten Hersteller erstellt werden.

Ein bidirektionaler Datenaustausch ermöglicht die Kommunikation von ansonsten inkompatiblen Steuerungen.

SPS-Co-Prozessor

In diesem Applikationsfall übernimmt ibaLogic die Funktion eines Co-Prozessors.

Im Bild „Anwendungsbereiche" stellen die grünen senkrechten Striche den Abtasttakt des Original- Automatisierungsgerätes dar. Nach einem SPS Takt (T1) werden Daten an die Soft-SPS (ibaLogic) übertragen. Mit den verfügbaren PC-Rechenleistung und den PC-Datenformaten, können komplexe Rechnungen in Echtzeit ausgeführt und die Ergebnisse vor T2 wieder zurück übertragen werden. Auch Modernisierungen können mit Zuhilfenahme solcher Methoden realisiert werden: Steuer- und Regelfunktionen der „alten" SPS werden schrittweise von der neuen ibaLogic-Automatisierung übernommen.

Messen & Zustandsbetrachtung (Condition Monitoring)

Neben dem Einsatz als Signalmanager lag es nahe, ibaLogic für komplexe Messaufgaben heranzuziehen, die im Standard-PDA nicht mehr bedienbar gewesen wären. Das Integrieren eines Bausteines zur Messwerterfassung ist eine der zentralen Funktionen von ibaLogic. Mit diesem „DAT_FILE_WRITE" können Messaufgaben ereignisgesteuert gemanagt und auf verschiedenen Datenträgern gespeichert werden. Eine Weiterverarbeitung und Analyse ist mit verschiedenen iba-Tools, z. B. ibaAnalyzer, ibaDatCoordinator, möglich.

Automatisierung (Steuern-Regeln)

Die Basis von ibaLogic ist die in der Norm IEC 61131-3 beschriebene grafische Programmiersprache. Diese Sprache wurde insbesondere für speicherprogrammierbare Steuerungssysteme (SPS) konzipiert. Die Entwicklungen der letzten Jahre haben aufgezeigt, dass Mess- und der Steuerungsmarkt immer stärker zusammenwachsen. Die logische Konsequenz ist, dass ibaLogic selbstverständlich auch für Automatisierungsaufgaben als vollwertige SPS eingesetzt werden kann.

Werden dabei die Aufgaben des Betriebs- und des Laufzeitsystems von einem PC übernommen, spricht man von einer PC-gestützten Soft-SPS oder kurz PAC (Programmable automation controller).

ibaLogic erlaubt eine Abtrennung des Laufzeitsystems vom PC in eine unterlagerte eigenständige Intelligenz, ibaPADU-S-IT. In einem solchen Fall kann der PC abgeschaltet werden, ohne dass das Laufzeitsystem anhält. Der PC wird in solchen Fällen, wie auch bei allen anderen SPS-Systemen, als Entwicklungsstation eingesetzt.

Simulator

Bei dieser Applikation handelt es sich um einen aktiven, mit Sprachmitteln von ibaLogic programmierten Simulator. Die Bedienungsschnittstelle und das Ergebnis der Simulation werden durch eine HMI-Visualisierung abgebildet. Die Verbindungen zwischen ibaLogic-Simulation und HMI erfolgen über die integrierte Standard-OPC-Schnittstelle.

4.3 Die Komponenten von ibaLogic

ibaLogic basiert auf dem Server-Client-Modell. Diese Architektur ermöglicht ein dezentrales Arbeiten sowie einen Multi-Client-Betrieb.

ibaLogic setzt sich aus folgenden Komponenten zusammen:

- ❑ Laufzeitsystem (PMAC)
- ❑ ibaLogic Server
- ❑ ibaLogic Client
- ❑ Datenbank
- ❑ OPC-Server

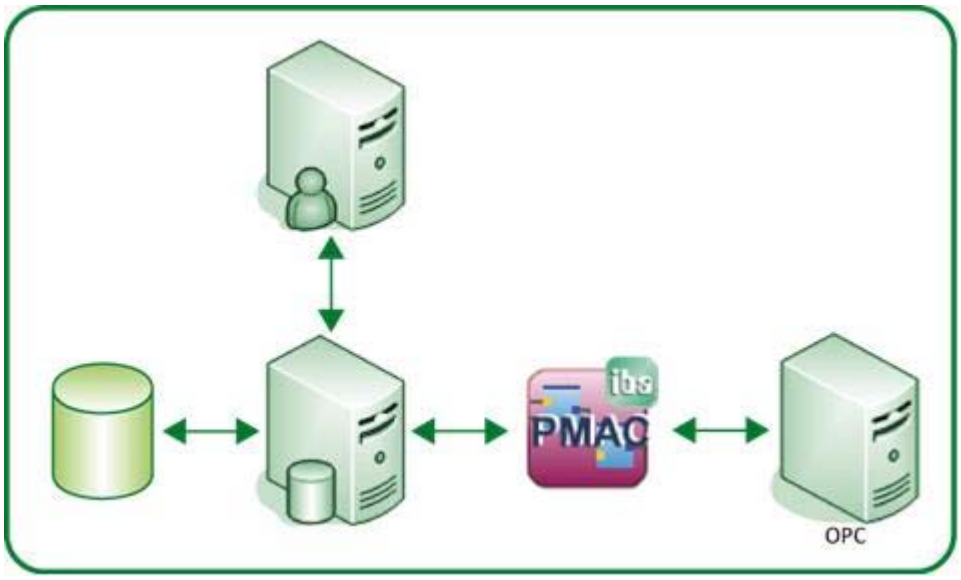
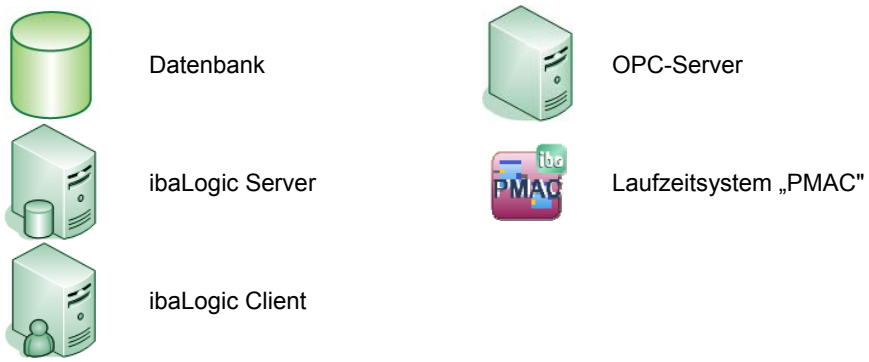


Abbildung 3: ibaLogic-Komponenten



Im einfachsten Fall befinden sich die oben genannten Komponenten auf einem Rechner. Allerdings ist es auch möglich, diese Komponenten auf separaten Rechnern zu betreiben.

4.3.1 Laufzeitsystem (PMAC)

Durch das Starten eines ibaLogic-Projekts im ibaLogic Client wird dieses kompiliert und in den „Programmable Measurement and Automation Controller" (PMAC) geladen. Dieses Laufzeitsystem kann sich auf einem Windows Rechner (als Windows-Dienst) oder auf einem kompatiblen Windows-CE-Gerät (ibaPADU-S-IT) befinden.

Das Laufzeitsystem liefert permanent aktuell berechnete Werte an den Client zurück. Diese werden in der grafischen Oberfläche des Projekts online angezeigt.

Wenn ein Projekt auf das Laufzeitsystem übertragen und gestartet wurde, dann ist dieses in der Lage autark ohne Server/Client zu laufen.

4.3.2 ibaLogic Server

ibaLogic ist ein Datenbank-basiertes System. Der ibaLogic Server übernimmt als zentraler Manager die Verwaltung der Datenbank und der Kommunikation zwischen dem ibaLogic Client und dem Laufzeitsystem.

Alle ibaLogic-Projekte werden vom ibaLogic Server in einer Datenbank verwaltet.

ibaLogic verwendet die lizenzfreie Microsoft SQL Express-Datenbank. Beim Installationsvorgang wird ein Microsoft SQL Express Server mit der dazugehörigen Datenbank, falls noch nicht vorhanden, installiert.

Der ibaLogic Server-Dialog dient auch zum Sichern und Wiederherstellen von ibaLogic-Applikationen. Gesichert wird ein Backup der Datenbank in einer externen Datei.

Kommt es zu Änderungen an ibaLogic-Projekten, werden diese automatisch in die Datenbank übernommen. Ein gezieltes Speichern während des Projektierens entfällt.

4.3.3 ibaLogic Client

Der ibaLogic Client ist die Programmierumgebung, in der ibaLogic-Projekte programmiert und konfiguriert werden. Dazu muss dieser mit einem ibaLogic Server verbunden sein. Dieser ibaLogic Server kann auf demselben Rechner oder im Netzwerk vorhanden sein.

Darüber hinaus steuert der ibaLogic Client mithilfe des ibaLogic Servers das Laden, Starten und Stoppen eines ibaLogic-Projekts im Laufzeitsystem.

4.3.4 OPC-Server

Der OPC-Server stellt alle Variablen, die als „OPC-sichtbar" deklariert wurden, den verbundenen OPC-Clients zur Verfügung. Bei den OPC-Clients handelt es sich im Allgemeinen um HMI-Systeme. Der OPC-Server läuft standardmäßig auf derselben Maschine wie der ibaLogic Server, kann aber auch auf anderen Rechnern im Netzwerk explizit gestartet werden und ist unabhängig davon direkt mit dem PMAC per TCP/IP verbunden.

4.4 Der Multi-Client-Betrieb und andere Systemkonfigurationen

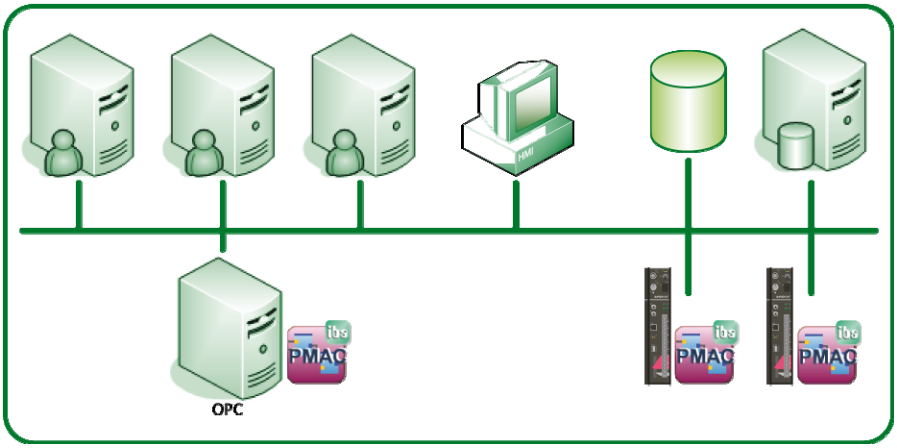
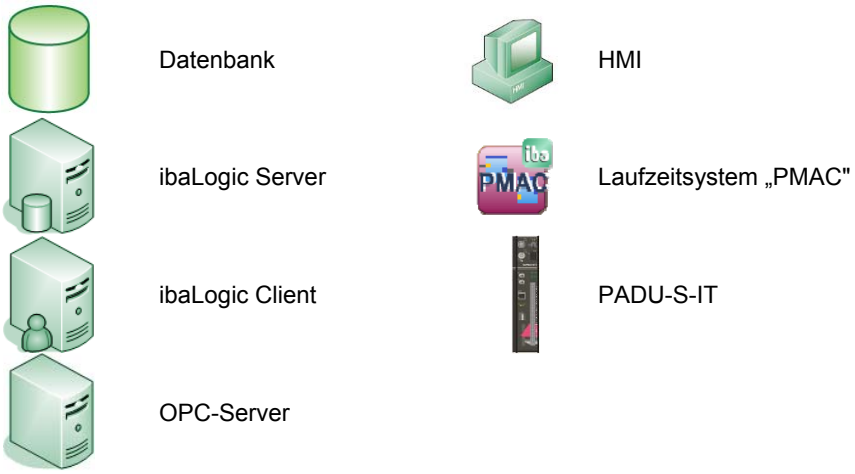


Abbildung 4: Mögliche Systemkonfiguration



Im einfachsten Fall kann ibaLogic mit all seinen Komponenten auf einem Windows-Rechner betrieben werden.

Die ibaLogic Komponenten können alternativ auch auf verschiedene Rechner verteilt laufen. In der obigen Beispielkonfiguration gibt es 3 ibaLogic-Applikationen. Eine läuft auf einem PC, 2 weitere jeweils auf einem PADU-S-IT-System. Der zentrale Server hat eine Verbindung zu einer Datenbank, auf der alle 3 Projekte gesichert werden. Auf diesem Server ist immer nur ein ibaLogic-Arbeitsbereich aus der Datenbank geladen, welcher drei Projekte entsprechend der drei PMACs enthalten kann.

Im Multi-Client-Betrieb ist es möglich, von verschiedenen Rechnern aus die Arbeitsbereiche zu bedienen und zu beobachten.

Wichtiger Hinweis

Änderungen an den Projekten eines Servers sollten nur von einem Client aus gleichzeitig durchgeführt werden.

In einem Arbeitsbereich ist wahlweise aber immer nur ein Projekt/Applikation „aktiv". Nur dieser aktive PMAC kann **online** bedient und beobachtet werden.

Über den OPC-Server kann das HMI-System versorgt werden. Auf dem PADU-S-IT kann kein OPC-Server laufen. HMI-Daten von und zu den PADU-S-IT Kopfeinheiten müssen damit über diesen Windows-Rechner laufen.

4.5 Betriebsarten und Verarbeitungsmodi

ibaLogic bietet eine Reihe von Betriebsarten, um den unterschiedlichen Anforderungen verschiedener Anwendungen Rechnung zu tragen. Da ibaLogic nicht nur als Soft-SPS, sondern auch als Signalmanager, Signalprozessor oder Simulator arbeiten kann, wurden mehrere Verarbeitungsmodi implementiert.

Folgende Betriebsarten sind in ibaLogic auswählbar:

- ❑ Messung
- ❑ Soft-SPS

Betriebsart einstellen

Vorgehen

1. Wählen Sie im Menü „Extras – I/O-Konfigurator". Das Fenster „I/O-Konfigurator" wird angezeigt.
2. Die Betriebsarten-Optionen finden Sie in der Register-Karte „Hardware-Konfiguration – Allgemeine Einstellungen".
3. Aktivieren Sie im Bereich „Allgemeine Einstellungen" die zu verwendende Betriebsart.
4. Klicken Sie abschließend auf <Übernehmen>.
5. Wenn Sie den I/O-Konfigurator schließen möchten, dann klicken Sie auf <OK>.

Folgender Verarbeitungsmodus ist in ibaLogic auswählbar:

„Buffered Mode" (=Paket-Übertragung)

Erläuterungen siehe „Zeitverhalten , Seite 239" und „Buffered Mode , Seite 195".

Verarbeitungsmodi einstellen

Vorgehen

1. Markieren Sie im linken Verzeichnisbaum die Hardware, deren Verarbeitungsmodi Sie einstellen möchten. Zu dieser Hardware wird das Register „Hardware-Konfiguration" angezeigt.
2. Aktivieren Sie im Bereich „Verbindungseinstellungen" den gewünschten Modus.
3. Klicken Sie abschließend auf <Übernehmen>.
4. Wenn Sie den I/O-Konfigurator schließen möchten, dann klicken Sie auf <OK>.

Hinweis

Weitere Informationen siehe „Allgemeine Einstellungen , Seite 182".

4.6 Aufbau einer ibaLogic-Applikation

Eine ibaLogic-Projekt-Applikation besteht aus den folgenden Elementen:

Arbeitsbereich

- ❑ Projekt 1
  - Task 1 / Programm 1
  - Task 2 / Programm 2
  - Task n / Programm n
- ❑ Projekt 2
  - Task 1 / Programm 1
  - Task 2 / Programm 2
  - Task n / Programm n
- ❑ Projekt n

Die Programme mit ihren Eigenschaften (Taskintervall usw.) können jeweils einem Projekt zugewiesen werden. Die Projekte sind wiederum in einem Arbeitsbereich organisiert.
Ein Projekt ist einem Laufzeitsystem/PMAC zugeordnet. Innerhalb eines Arbeitsbereiches ist nur ein Projekt aktiv gesetzt, d. h. es kann nur dieses aktive Projekt gestartet bzw. gestoppt werden.

Hinweis

Innerhalb einer Applikation kann nur ein Projekt aktiv sein.

4.6.1 Task-/Programm-Eigenschaften

Gemäß der IEC 61131-3 können mehrere Programme einem Task zugeordnet werden. ibaLogic unterstützt die feste Zuordnung von einem Programm zu einem Task.

Jedem Task können folgende Eigenschaften zugeordnet werden:

- ❑ Intervallzeit
- ❑ Priorität

Die Intervallzeit bestimmt das Zeitraster, in dem der Task immer wieder neu gestartet wird. Das minimale Zeitraster für die Intervallzeit beträgt 1 ms.

Mit der Einstellung der Priorität wird festgelegt, in welcher Reihenfolge die Intervall-Programme eines Projekts abgearbeitet werden, beginnend mit der Priorität 0.

Hinweis

Die Programmeigenschaften „Intervallzeit" und „Priorität" sind für die Projekt-Performance von wesentlicher Bedeutung. Eine genauere Betrachtung dieser Programmeigenschaften siehe „Zeitverhalten , Seite 239" und „Leistungsgrenzen , Seite 249".

4.6.2 Programmelemente

Ein ibaLogic-Programm kann aus folgenden Elementen bestehen:

- ❑ Funktionsbausteine
- ❑ Funktionsbausteine der integrierten Standard-Bibliothek
- ❑ Durch den Benutzer erstellte Funktionsbausteine mit Structured Text
- ❑ Durch den Benutzer erstellte Makroblöcke
- ❑ Durch den Benutzer erstellte Funktionsbausteine auf DLL basierend
- ❑ Verbindungselemente
- ❑ Hardware Eingangs- und Ausgangssignale
- ❑ Kommentare

4.6.2.1 Funktionsbausteine

ibaLogic verfügt über eine Bibliothek von Funktionsbausteinen. In dieser Bibliothek sind Standardfunktionsbausteine gemäß der IEC 61131-3 sowie ergänzende Funktionsbausteine enthalten.

Für eine übersichtlichere Programmstruktur und eine Kapselung von einzelnen grafischen Programmteilen können diese zu einem Makroblock zusammengefasst werden.

Es besteht ebenfalls die Möglichkeit, einen individuellen Baustein, der für eine spezielle Problemlösung benötigt wird, selbst zu erstellen.

Zu diesem Zweck bietet ibaLogic an, einen neuen Funktionsbaustein mithilfe von Structured Text zu erstellen. Der ST-Code ist hier für den Anwender einsehbar und veränderbar.

Eine Variante des selbst erstellten Funktionsbausteins ist das Erstellen eigener DLLs (mit einem von iba zur Verfügung gestellten DLL-Rahmen). Bei dieser Option ist der Code verborgen. Der so erzeugte Baustein steht wie ein Standardfunktionsbaustein zur Verfügung.

Andere Dokumentation

Weiterführende Informationen entnehmen Sie in der Dokumentation zur Erstellung von DLLs auf der mitgelieferten CD „ibaSoftware and Manuals".

4.6.2.2 Grafische Programmierung

Folgende Elemente stehen zur Verbindung von Funktionsbausteinen zur Verfügung:

- ❑ Verbindungslinien
- ❑ Intra-Page-Konnektoren (IPC)
- ❑ Off-Task-Konnektoren (OTC)
- ❑ Konvertierer
- ❑ Splitter
- ❑ Joiner

Die grafische Programmierung erfolgt durch das Verbinden der Funktionsbausteine mithilfe von Verbindungslinien. Für eine bessere Programmstrukturierung können Sie Intra-Page-Konnektoren benutzen.

Ein Intra-Page-Konnektor stellt lediglich eine zeichnerische Vereinfachung dar. Der IPC ersetzt dabei eine Verbindungslinie. Dies ist insbesondere dann vorteilhaft, wenn sehr viele Objekte auf einer Seite mit demselben Punkt verbunden werden müssen oder „lange" Verbindungen über mehrere Seiten erforderlich sind.

Off-Task-Konnektoren dienen als programmübergreifende Verbindungselemente. Diese sind immer dann erforderlich, wenn zwischen mehreren Programmen kommuniziert werden soll.

Off-Task-Konnektoren werden auch zur Kommunikation zwischen ibaLogic und OPC-Clients genutzt. Das ist im Off-Task-Konnektor parametrierbar.

Tipp

Weitere Informationen siehe auch "Grafische Verbindungen , Seite 155" sowie "Konvertierer, Splitter, Joiner , Seite 163".

4.6.2.3 Kommentare

Kommentare können zur Strukturierung und zur einfachen Programmbeschreibung genutzt werden. Diese können frei platziert oder auch an ein anderes Element „angedockt" werden.

4.6.2.4 Verfügbare Datentypen

ibaLogic unterstützt alle in der Norm IEC 61131-3 definierten elementaren und zusammengesetzten Datentypen (Ausnahme: WSTRING).

4.6.2.5 Integriertes Messen mit ibaPDA Express

Zur schnellen Darstellung einer Signalform dient das integrierte Werkzeug ibaPDA Express.

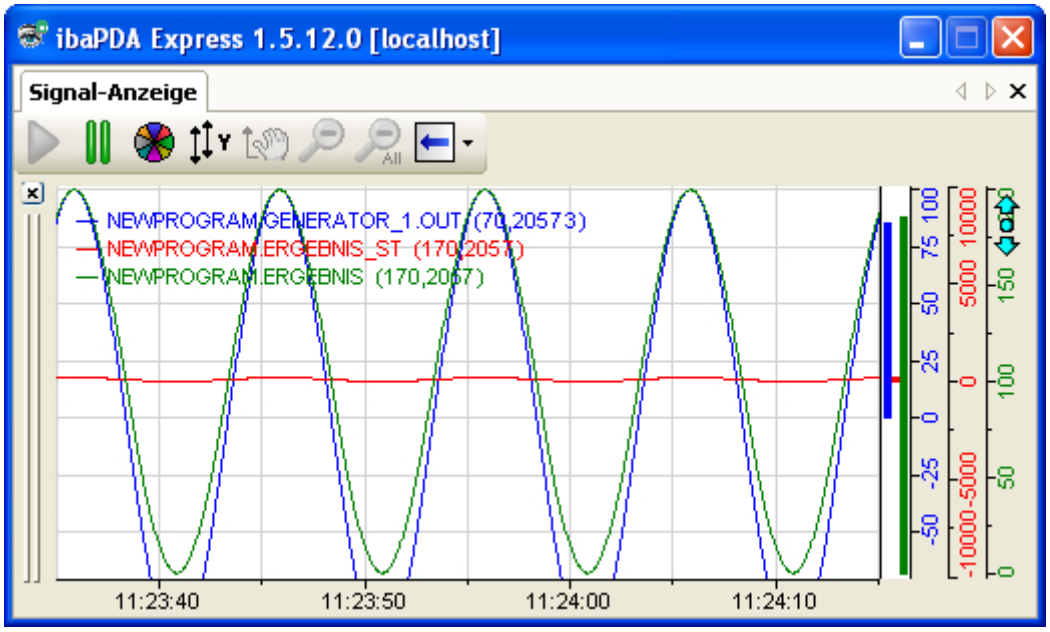


Abbildung 5: Integriertes Messen mit ibaPDA Express

Die Signale können bei gedrückter ALT-Taste mithilfe von Drag & Drop in das ibaPDA Express-Fenster gezogen und dargestellt werden.

ibaPDA Express speichert keine Daten für eine Langzeitaufzeichnung.

4.6.2.6 Messwertspeicherung

Eine Messwertspeicherung ist mithilfe des lizenzpflichtigen Funktionsbausteins „DAT_FILE_WRITE" möglich. Weitere Informationen siehe „DAT_FILE_WRITE (DFW-Funktionsbaustein) , Seite 97".

Tipp

Die Messsignale in den erzeugten Messdateien (*.dat) können mit der komfortablen Analyse-Software ibaAnalyzer dargestellt und ausgewertet werden.

4.7 Konnektivität

Die ibaLogic-Systeme sind in der Lage, untereinander über die im Windows-PC oder PADU-S-IT vorhandenen Schnittstellen zu kommunizieren (z. B. via TCP/IP, ibaNet etc).

In der Konnektivitätsübersicht wird die Kommunikation zu Fremdsystemen bzw. zu diskreten I/Os gezeigt.

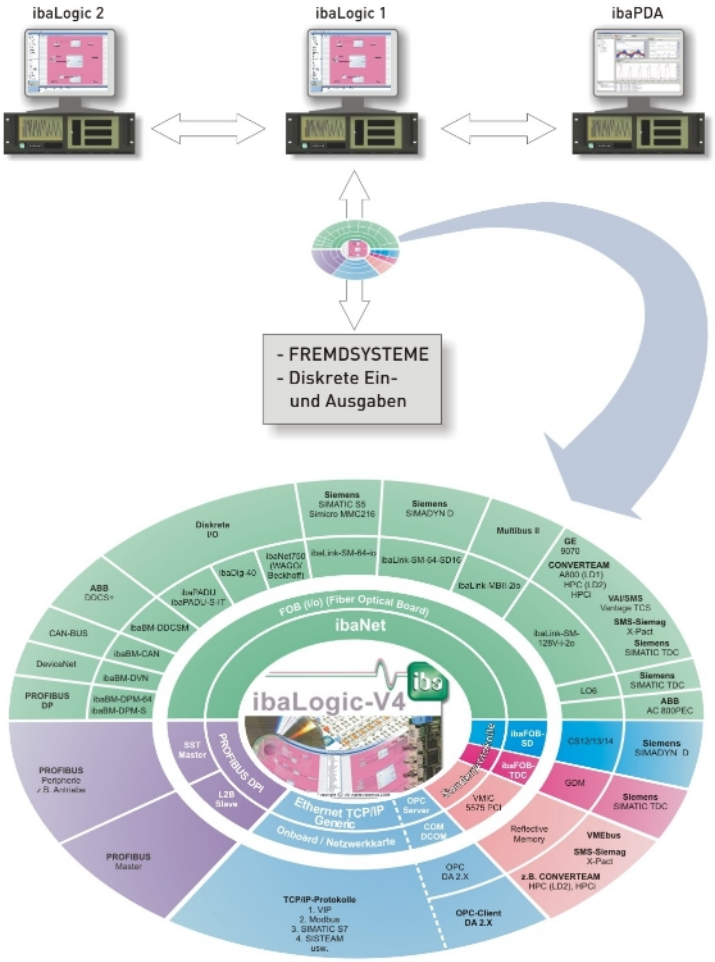


Abbildung 6: Konnektivität zu iba- und Fremdsystemen

5 ibaLogic Server

5.1 Funktionsübersicht des ibaLogic Servers

Der ibaLogic Server ist nicht nur der zentrale Punkt der Kommunikation zwischen ibaLogic Client und der PMAC, sondern dieser ist auch für die Verwaltung der ibaLogic-V4-Projekte in der Datenbank zuständig. Ebenso verwaltet dieser im Hintergrund eine Verbindung zu einem aktiven PMAC für Lade- und Start-/Stopp-Aktionen des PMACs. Diese Verbindung wurde über den ibaLogic Client eingerichtet.

Der Server kann daher in folgende Funktionen unterteilt werden:

- Server Bedienung
  - Starten/Stoppen/Schließen
  - Datenbank Aktionen
    Sichern und Wiederherstellen
    Rücksetzen der kompletten aktuellen Datenbank
- Administrative Einstellungen
  - Port-Nummer für Client-Verbindungen einstellen
  - Verbindungen zu ibaLogic-Datenbanken und deren Parameter einrichten
  - Autostart für Server und lokalen PMAC einrichten
  - Automatisches Sichern der aktuellen Datenbank einrichten
  - Allgemeine Server-Optionen einstellen
  - Datenbank-Skripte Status anzeigen/ausführen (nur für Support-Zwecke)

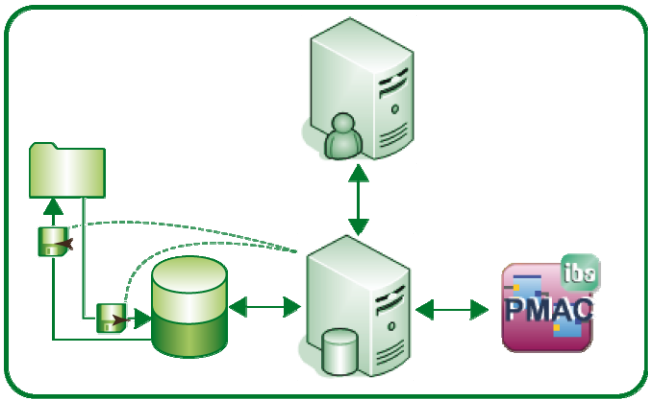
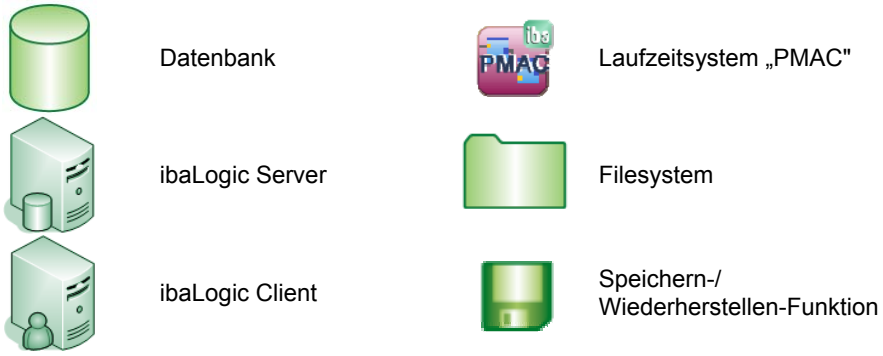


Abbildung 7: Funktionsübersicht des ibaLogic Servers



5.2 ibaLogic Server starten

Voraussetzung

Sie haben die ibaLogic Server-Verknüpfung auf dem Desktop.

Vorgehen

1. Doppelklicken Sie auf die Verknüpfung „ibaLogic Sever" auf dem Desktop.
   Der Dialog „ibaLogic Server" wird angezeigt.

Beim Öffnen des ibaLogic Servers geht der Server automatisch in den Start-Zustand. Im Infobereich werden folgende Symbole  angezeigt.

2. Klicken Sie auf den Button <Start>, um den ibaLogic Server zu starten. Sie können den Server auch über das Menü „Server - Start" starten.
   Im Serverdialog wird der Start-Stopp-Zustand durch ein Symbol, einer Meldung und Hinterlegen des aktiven Buttons angezeigt.

5.3 Bedienoberfläche – ibaLogic Server

Der ibaLogic Server wird verwendet zur:

- ❑ Konfiguration des Servers
- ❑ Versetzung in Start-/Stop-Modus
- ❑ Konfiguration der Datenbank und Sicherung

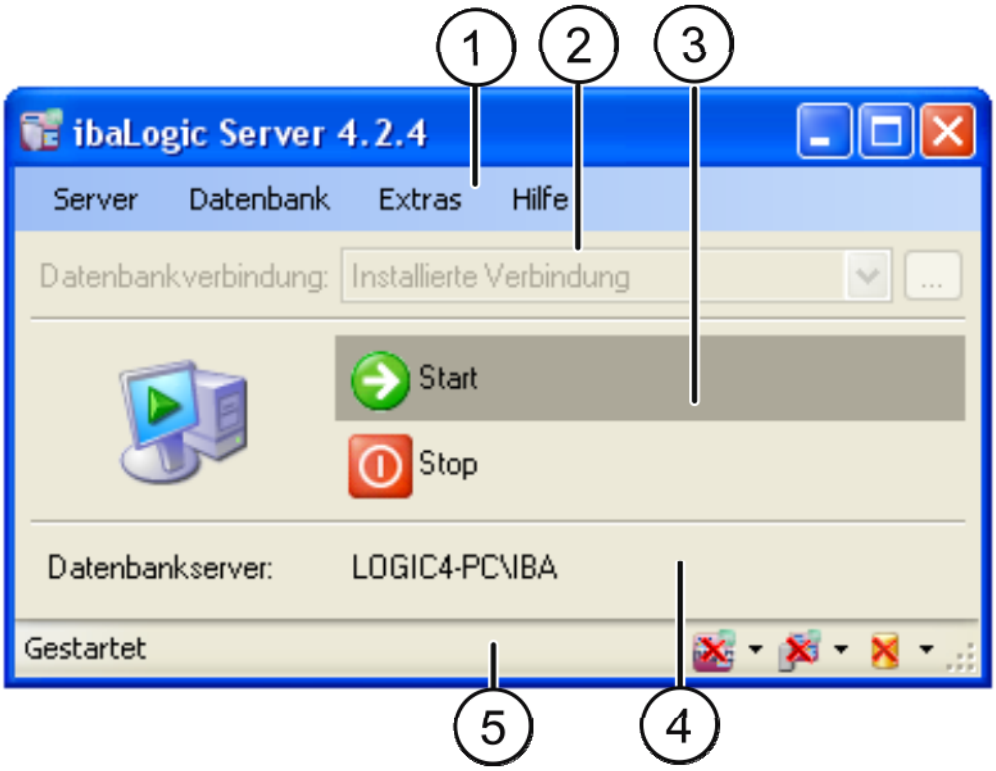


Abbildung 8: ibaLogic Server-Bedienoberfläche

1	Menüleiste	4	Aktuelle Datenbankverbindung
2	Einstellung Datenbankverbindung	5	Statusleiste
3	Button Start / Button Stop		

5.4 ibaLogic Server-Einstellung

5.4.1 Client-Port konfigurieren

Die Client-Portnummer dient als Verbindungsparameter für einen ibaLogic Client.

Hinweis

Diese Einstellung sollte nur dann verändert werden, wenn auf dem Server-Computer bereits ein Dienst ausgeführt wird, der diesen Port belegt. Im Client muss beim Verbinden mit diesem Server derselbe Port eingestellt werden. Tragen Sie bei Auswahl der Verbindung ebenfalls die neue Port-Nummer ein. Wählen Sie Menü „Datei - Mit Server verbinden..." des Clients.

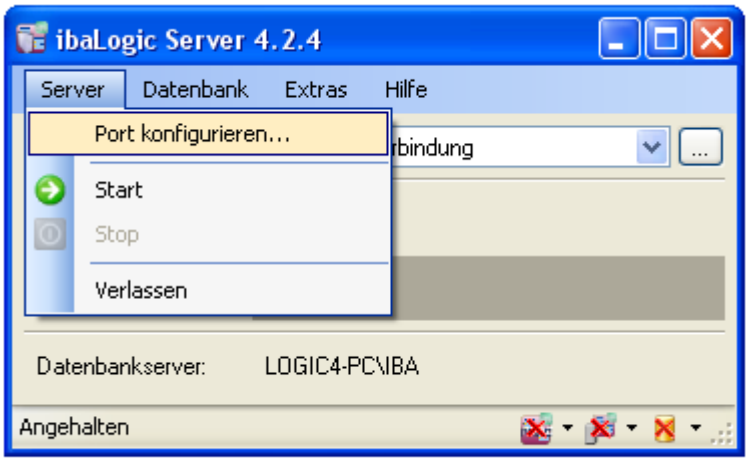
Der Port ist mit dem Default-Wert 8086 voreingestellt.

Voraussetzung

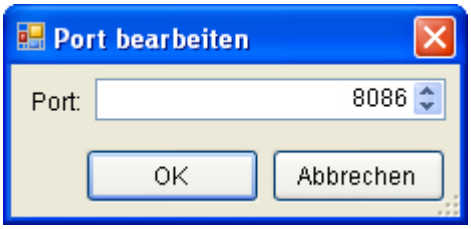
❑ Sie haben den ibaLogic Server gestoppt.

Vorgehen

1. Wählen Sie Menü „Server - Port konfigurieren…".



2. Geben Sie den Port des Clients direkt in das Eingabefeld ein oder stellen Sie den Port mittels des Spinners ein



5.4.2 Datenbankverbindungen konfigurieren

Wichtiger Hinweis

Wenn sich der Server nach einer Änderung des PC-Namens nicht mehr mit der lokalen Datenbank verbinden kann, dann ändern Sie bitte den Verbindungsnamen unter „DataSource“ vom alten Rechnernamen in „localhost“.
(Siehe "Datenbankschnittstelle konfigurieren , Seite 42")

In ibaLogic ist es möglich, mehrere Datenbankverbindungen einzurichten. Es ist aber immer nur eine aktiv. Beim Installieren von ibaLogic wird die Datenbank lokal angelegt und die Default-Einstellung bezieht sich darauf.

Hinweis

Nur wenn Sie sich mit einer **nicht lokalen** Datenbank verbinden wollen und diese nicht schon bei der Installation angegeben haben, müssen Sie die nachfolgend beschriebenen Aktionen durchführen.

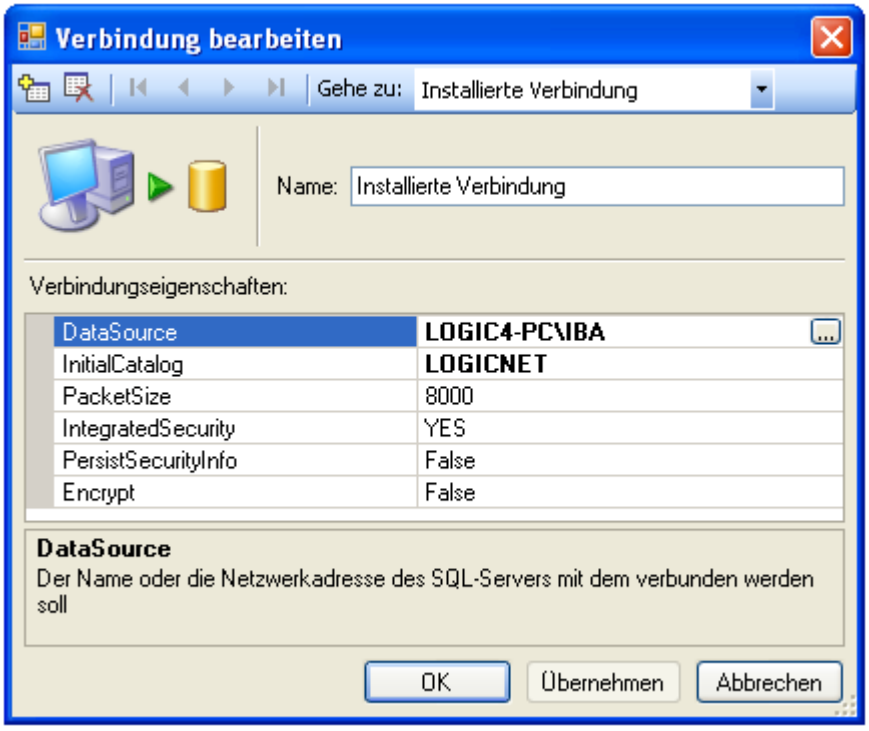
5.4.2.1 Datenbank verbinden

Vorgehen

1. Wählen Sie im Menü Datenbank den Punkt "Datenbankverbindungen...".

2. Wählen Sie in der Auswahlliste "Gehe zu:" eine ibaLogic-Datenbankverbindung. Voreingestellt ist die Datenbankenverbindung „Installierte Verbindung". Das ist die Verbindung zur lokalen Datenbank.

3. Rufen Sie mit dem Browser-Button <…> den Konfigurations-Dialog für Datenbankverbindungen auf. Der Browser-Button wird erst durch einen Klick in das Textfeld der Zeile "DataSource" sichtbar.



Vorsicht!

Die folgenden Parameter

- InitialCatalog
- PacketSize
- IntegratedSecurity
- PersistSecurityInfo
- Encrypt

müssen nur geändert werden, wenn Sie z. B. eine bereits vorhandene zentrale Datenbank auch für ibaLogic verwenden wollen.

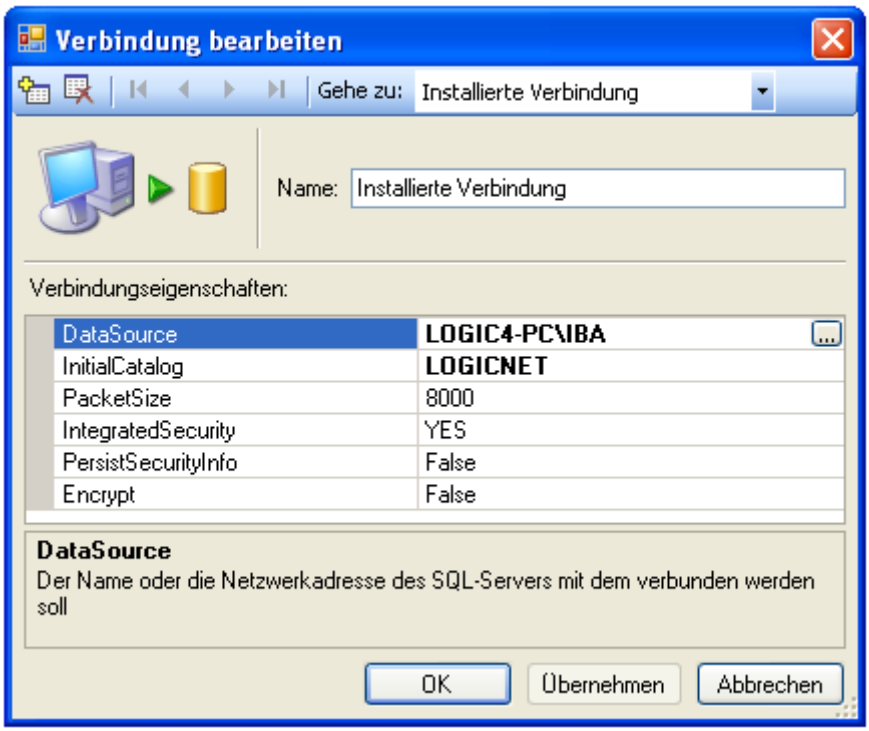
Dafür müssen Sie aber über grundlegende Datenbankkenntnisse verfügen.
Im Zweifelsfall lassen Sie diese Einstellungen durch einen Datenbankadministrator vornehmen.

5.4.2.2 Datenbankschnittstelle konfigurieren

Bei „DataSource" geben Sie den Rechner-Namen und die Instanz der ibaLogic-Datenbank an, mit der eine Verbindung hergestellt werden soll.

Vorgehen

➲ Geben Sie den Namen des Servers direkt in das Textfeld ein oder wählen Sie die Datenbank über den Verzeichnisbaum mithilfe des Browser-Buttons aus.
Der Browser-Button wird erst durch einen Klick in das Textfeld sichtbar.



5.4.2.3 SQL-Server auswählen

Der Dialog „SQL-Server auswählen" enthält 2 Register „Lokale Server" und „Netzwerk-Server".

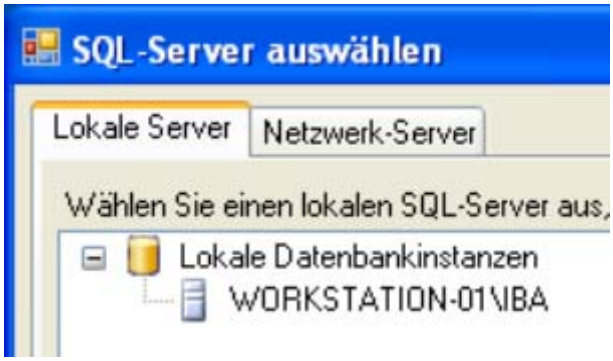


Abbildung 9: Lokale Server-Instanzen

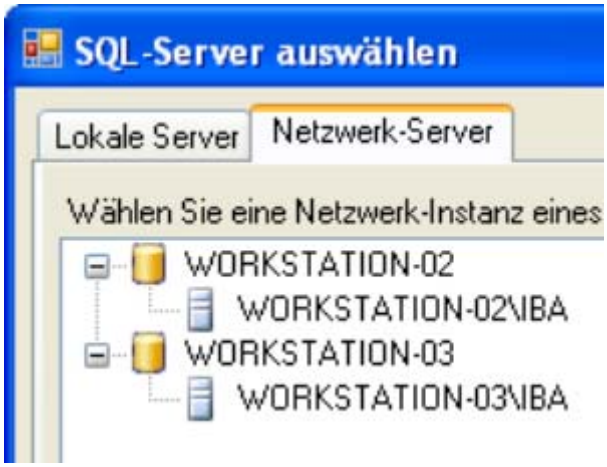


Abbildung 10: Netzwerk-Server-Instanzen

Unter „Lokale Datenbankinstanzen" sind alle auf dem Rechner verfügbaren SQL-Datenbankinstanzen aufgelistet.

Nach Aufruf des Registers „Netzwerk-Server" wird das gesamte verfügbare Netzwerk nach SQL-Instanzen durchsucht. Dieser Vorgang kann einige Zeit in Anspruch nehmen. Solange das Netzwerk durchsucht wird, erscheint im Textfeld die Meldung „Informationen werden ausgelesen".

Vorgehen

1. Klicken Sie auf die SQL-Serverinstanz mit der sich der ibaLogic Server verbinden soll.
2. Klicken Sie abschließend auf <OK>. Die SQL-Serverinstanz wird übernommen. Der Dialog wird geschlossen.

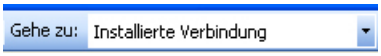
Ergebnis

Die SQL-Serverinstanz wird mit dem ibaLogic Server verbunden.

Anmerkung

Weitere Datenbankverbindungen können installiert oder gelöscht werden.

Für die Konfiguration der Datenbankverbindung stehen folgende Symbole zur Verfügung:

Symbole/Auswahlfeld	Tooltipp	Erklärung
	Hinzufügen	Hinzufügen einer neuen Datenbank-Verbindung
	Entfernen	Löschen einer Datenbank-Verbindung
	Anfang	Zur ersten Verbindung
	Zurück	Zur vorhergehenden Verbindung
	Weiter	Zur nächsten Verbindung
	Ende	Zur letzten Verbindung
	Objekt auswählen	Anwahlmöglichkeit einer Verbindung

5.4.2.4 Datenbankskripte verwalten

Wichtiger Hinweis

Die Liste der installierten Datenbankskripte dient nur zur Information und zur Kontrolle für den iba-Support. Nehmen Sie keine Änderungen vor.

Mit Aufruf der Funktion „Datenbankskripte" werden alle in ibaLogic implementierten Skripte mit den dazugehörigen Informationen wie der Versionsnummer, den Skriptnamen und dem Installationsdatum in einer Tabelle dargestellt. Um den Dialog „Datenbankskripte" aufzurufen, muss der ibaLogic Server gestoppt sein.

Normalerweise werden die Datenbankskripte über Versions-Updates abgefragt und nach vorheriger Abfrage automatisch durchgeführt.

5.4.3 Optionen

5.4.3.1 Autostart Server aktivieren

Bei jedem Start von Windows wird der ibaLogic Server-Dialog automatisch gestartet.

Sie können wählen, wo die Autostart-Optionen abgelegt werden:

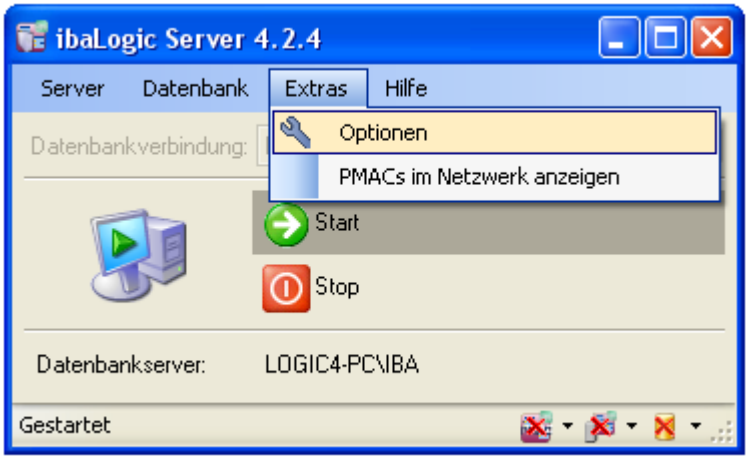
- ❑ In der Registrierung
- ❑ Unter dem Autostartordner das aktuellen Benutzers
- ❑ Unter dem Autostartordner für „Alle Benutzer"

Die standardmäßige Einstellung ist, dass beim Öffnen des ibaLogic Server-Dialogs auch der Server startet. Soll der Server-Dialog offen sein, aber der Server selbst im STOP-Modus, dann muss die Option „Autostart Server angehalten" aktiviert sein. In diesem Fall muss der Server manuell gestartet werden, damit die Client-Verbindungen akzeptiert werden.

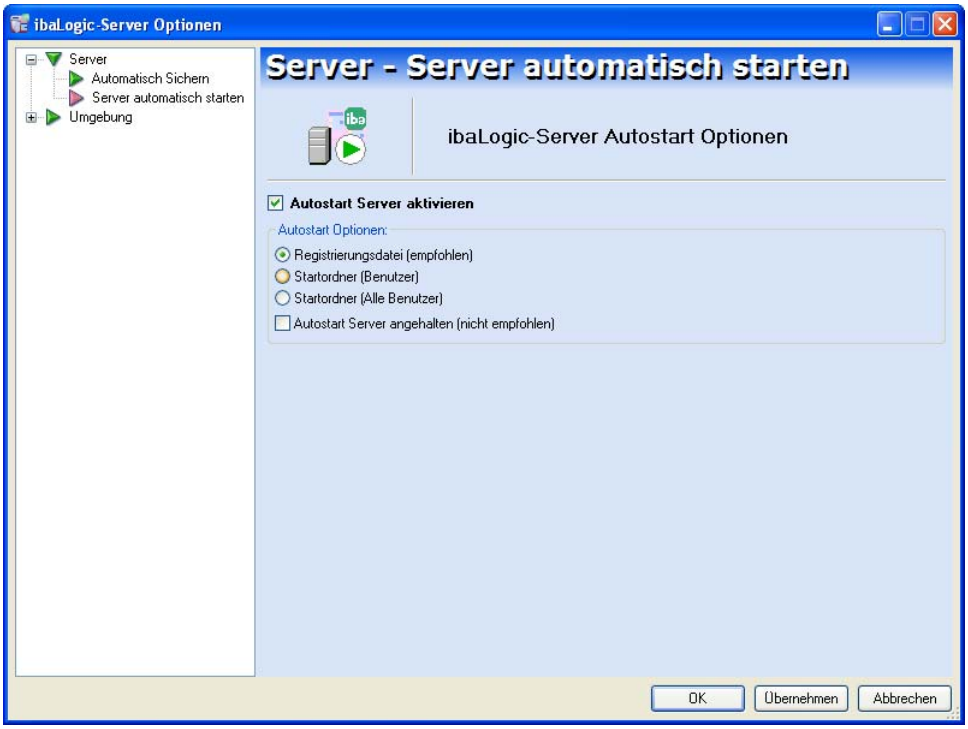
Autostart-Optionen	Erklärung
Registrierungsdatei (empfohlen)	Die Option bewirkt, dass die Autostart-Optionen in der Registrierungsdatei abgelegt werden.
Startordner (Benutzer)	Die Option bewirkt, dass die Autostart-Optionen in den Startordner eines Benutzers abgelegt werden.
Startordner (Alle Benutzer)	Die Option bewirkt, dass die Autostart-Optionen in den Startordner aller Benutzer abgelegt werden.
Autostart Server angehalten (nicht empfohlen)	Die Option bewirkt, dass der ibaLogic Server-Dialog geöffnet wird, aber der Server selbst wird nicht gestartet. Der Server bleibt im Stop-Modus.

Vorgehen

1. Wählen Sie Menü „Extras - Optionen".



2. Wählen Sie im Verzeichnisbaum „Server – Server automatisch starten".
3. Klicken Sie das Auswahlfeld „Autostart Server aktivieren" an.



4. Klicken Sie auf <Übernehmen>, um die Einstellungen zu aktivieren.

Ergebnis

Der ibaLogic Server-Dialog wird automatisch bei jedem Start von Windows geöffnet.

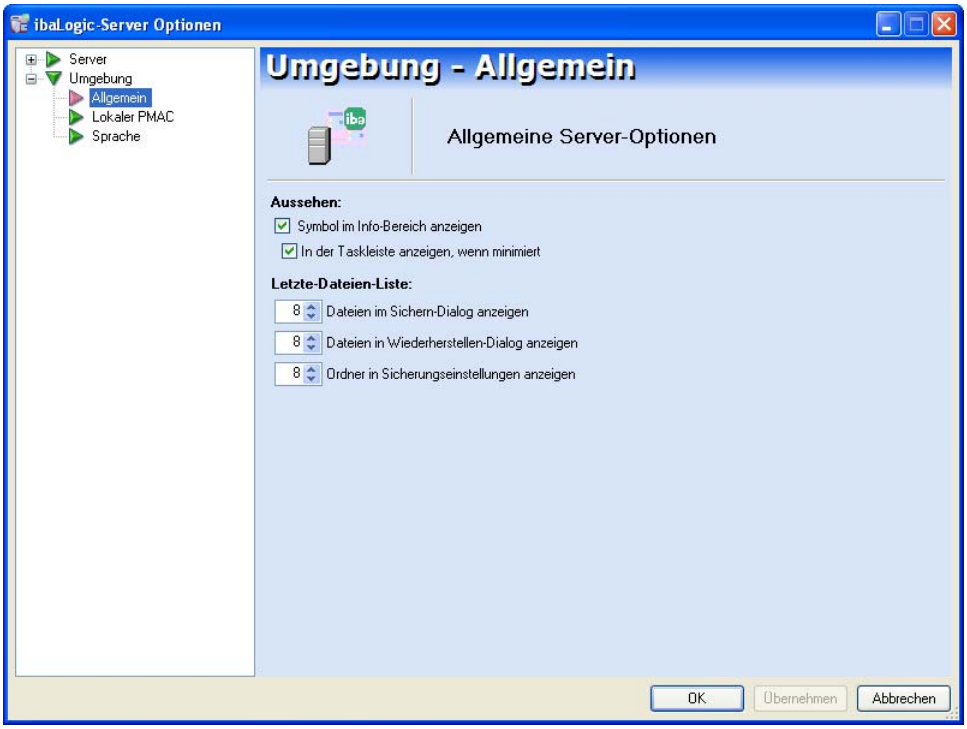
5.4.3.2 Allgemeine ibaLogic Server-Einstellungen konfigurieren

Folgende allgemeine ibaLogic Server-Einstellungen können konfiguriert werden:

Server-Option	Erklärung
Symbol im Info-Bereich anzeigen	Die aktivierte Option bewirkt, dass ein Symbol im Info-Bereich eingeblendet wird, wenn der ibaLogic Server-Dialog geöffnet ist.
In der Taskleiste anzeigen, wenn minimiert	Die aktivierte Option bewirkt, dass der ibaLogic Server-Dialog beim Minimieren in der Taskleiste erscheint.
Dateien im Sichern-Dialog anzeigen	Anzahl der Dateien, die im Sichern-Dialog angezeigt werden sollen. Auswählbar im Auswahlfeld „Sicherungsordner" unter „Server - automatisch sichern".
Dateien im Wiederherstellen-Dialog anzeigen	Anzahl der Dateien, die im Wiederherstellen-Dialog angezeigt werden sollen.
Ordner in Sicherungseinstellungen anzeigen	Anzahl der Ordner, die in den Sicherungseinstellungen anzeigt werden sollen.

Vorgehen

1. Wählen Sie Menü „Extras - Optionen".
2. Wählen Sie im Verzeichnisbaum „Umgebung - Allgemein".



3. Nehmen Sie die gewünschten Einstellungen vor.

5.4.3.3 Einstellungen für den lokalen PMAC

Der lokale PMAC ist als Windows-Dienst realisiert. Dessen Zustand und der Starttyp (Autostart-Optionen) werden hier eingestellt.

Zustandseinstellung:

- ❑ Aktivieren
- ❑ Deaktivieren

Zustands-Option	Beschreibung
Aktivieren	Beim Aktivieren wird der Dienst ggf. automatisch wieder installiert.
Deaktivieren	Über die Auswahl zwischen "PMAC-Vollversion" und "PMAC-Demoversion" kann zwischen den beiden Laufzeitversionen umgeschaltet werden. In der Vollversion ist ein Dongle zwingend erforderlich, in der Demoversion ist kein Hardware-Zugriff möglich, auch sind etliche Funktionen deaktiviert (keine Funktion in TCPIP_SendRecv, DatFileWrite, User-DLLs), auch wenn sie im Projekt verwendet werden können.

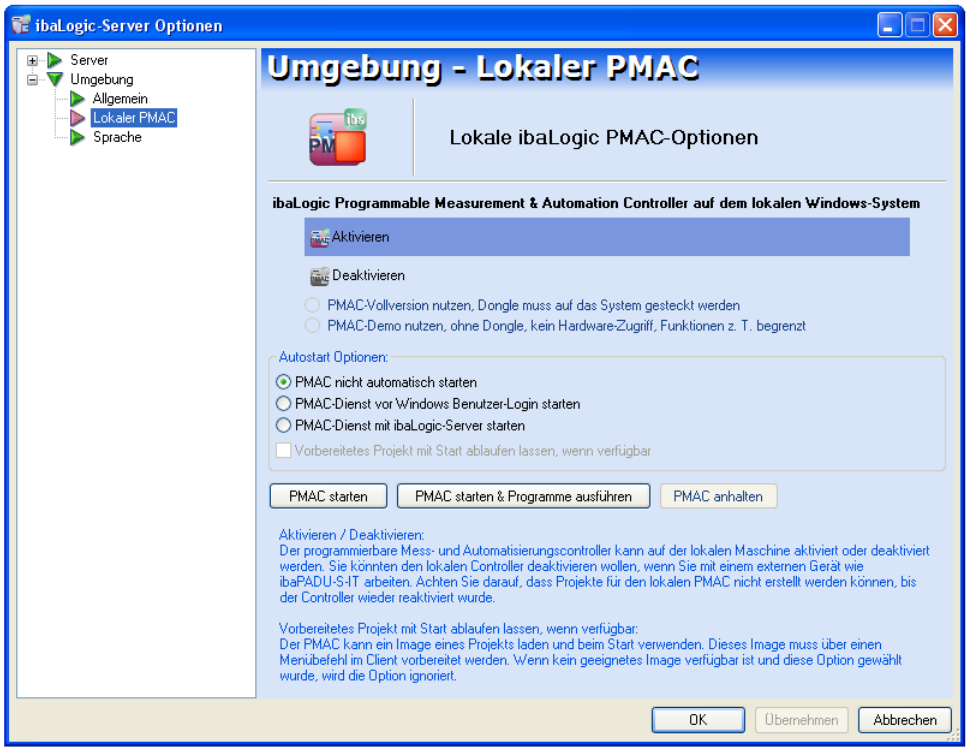
Einstellung der Autostart-Optionen (Starttyp):

Autostart-Option	Erklärung
PMAC nicht automatisch starten	Die Option bewirkt, dass kein Autostart des PMAC-Dienstes erfolgt.
PMAC-Dienst vor Windows Benutzer-Login starten	Die Option bewirkt, dass der PMAC-Dienst vor dem Windows-Benutzer-Login gestartet wird. Zusätzliche Option: Vorbereitetes Projekt mit Start ablaufen lassen, wenn verfügbar.
PMAC-Dienst mit ibaLogic Server starten	Die Option bewirkt, dass der PMAC-Dienst mit dem ibaLogic Server gestartet wird. Zusätzliche Option: Vorbereitetes Projekt mit Start ablaufen lassen, wenn verfügbar.
Vorbereitetes Projekt mit Start ablaufen lassen, wenn verfügbar	Die Option bewirkt, dass ein Image eines Projektes geladen und beim Start verwendet wird. Das Image muss zuvor über einen Menübefehl im Client vorbereitet werden. Wenn kein Image verfügbar ist und diese Option gewählt wurde, wird die Option ignoriert.

Beschreibung für das Umschalten der PMAC-Versionen (Demo- bzw. Vollversion)

Vorgehen

1. Im Dialog ibaLogic-Server Optionen unter "Umgebung - Lokaler PMAC" den PMAC anhalten und danach Deaktivieren sowie Deinstallieren lassen.



2. Jetzt die gewünschte Version des PMACs auswählen (Vollversion oder Demo), den Dienst wieder "Aktivieren" (wird dadurch installiert) und den <PMAC starten> bzw. je nach ausgewählter Autostart-Option mit dem nächsten Start des Servers.

3. Wählen Sie eine Autostart-Option aus.
   Wenn der Autostart-Dienst aktiviert ist, können Sie zwischen den angegebenen Autostart-Optionen auswählen.
   Zusätzlich können Sie einstellen, dass das Projekt gestartet werden soll.
   Dazu ist es notwendig, dass Sie dieses Projekt vorher „in PMAC gespeichert" haben.

4. Klicken Sie auf <Übernehmen>, um die Einstellungen zu aktivieren.Klicken Sie auf <Übernehmen>, um die Einstellungen zu aktivieren.

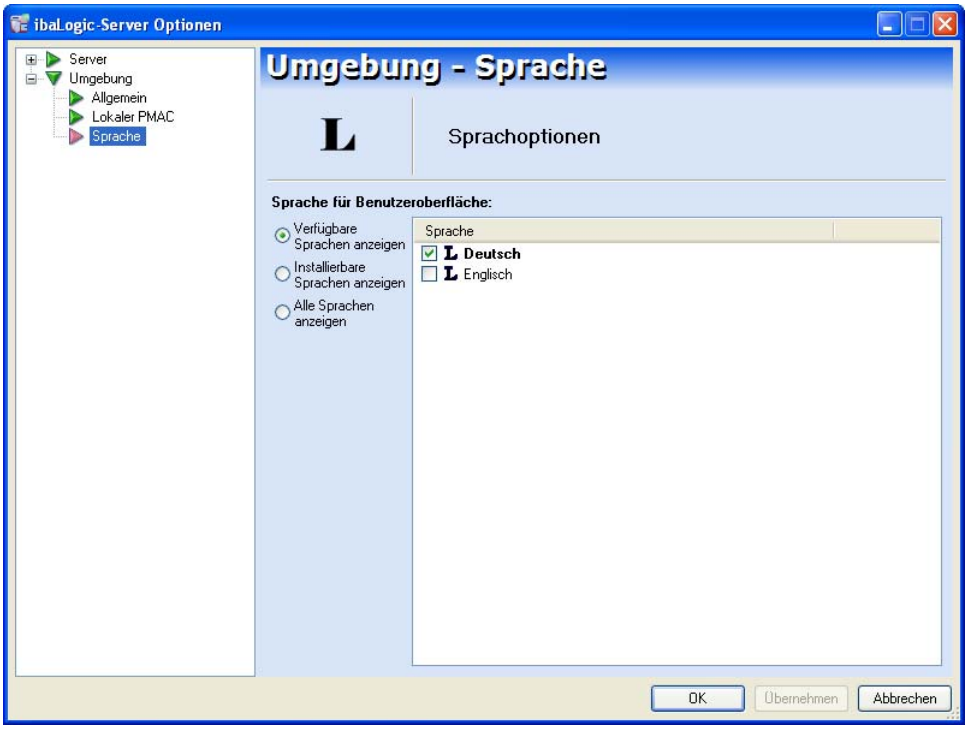
5.4.3.4 Sprache

Hier stellen Sie die Sprache des Server-Dialogs ein.

Die Sprache des Client-Dialogs muss extra eingestellt werden.

Vorgehen

1. Wählen Sie Menü „Extras - Optionen".
2. Wählen Sie im Verzeichnisbaum „Umgebung – Sprache".



3. Wählen Sie „Verfügbare Sprachen anzeigen".
4. Wählen Sie das Auswahlfeld mit der gewünschten Sprache aus.
5. Klicken Sie auf <Übernehmen>, um die Einstellungen zu aktivieren.

5.4.4 Statusleiste

Im ibaLogic Server-Dialog befinden sich 3 Symbole in der Statusleiste. Die Symbole dienen zum Aktivieren oder Deaktivieren der Autostart- und Sicherungseinstellungen.

Abbildung 11: Deaktivierte Funktionen in der Statulseiste

Abbildung 12: Aktivierte Symbole in der Statulseiste

Symbol	Einstellung	Beschreibung
	Autostart (Vor Login Starten)	Die Aktivierung bewirkt, dass bei jedem Start von Windows automatisch der PMAC gestartet wird.
	Autostart (Registry)	Die Aktivierung bewirkt, dass die Autostart-Optionen in der Registrierungsdatei abgelegt werden.
	Automatisch Sichern	Die Aktivierung bewirkt, dass die Datenbank entsprechend den Einstellungen gesichert wird.

6 Programmierumgebung – ibaLogic Client

Der ibaLogic Client wird dazu verwendet, um Programme zu erstellen und zu editieren.

6.1 ibaLogic Client starten

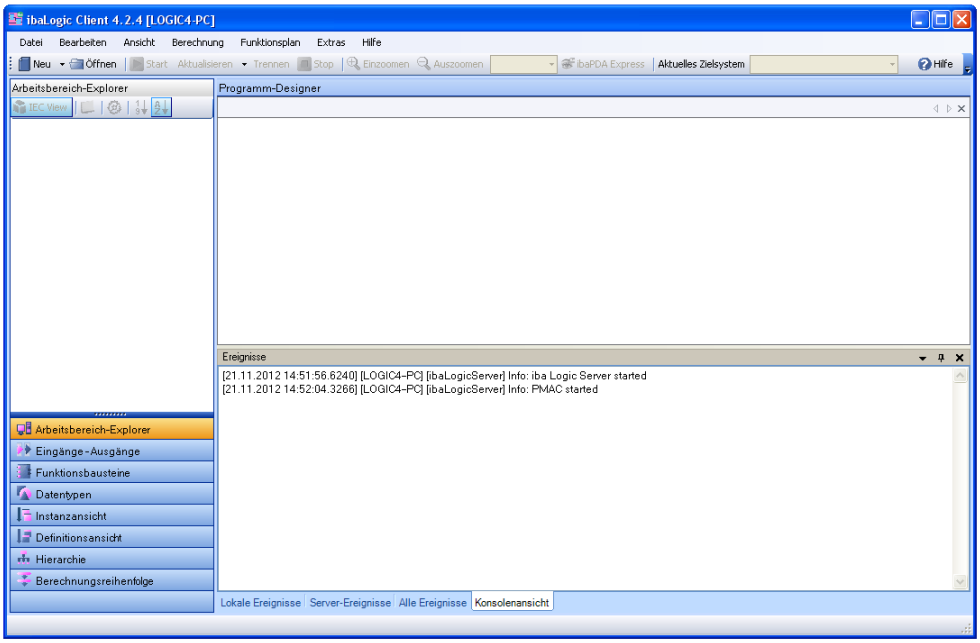
Die Programmierumgebung wird angezeigt.

Voraussetzung

- ❑ Sie haben die Start-Icons ibaLogic Client und ibaLogic Server auf dem Desktop angelegt (Standardinstallation).
- ❑ Sie haben ibaLogic Server gestartet.

Vorgehen

➲ Doppelklicken Sie auf das Symbol „ibaLogic Client" auf dem Desktop.
Nach kurzer Initialisierungsphase wird der ibaLogic Client-Dialog geöffnet.



Anmerkung

Das Ereignisfenster unter dem Programmfenster dokumentiert die Programmaktionen und mögliche Kollisionen.

6.2 Bedienoberfläche Programmierumgebung – Editor

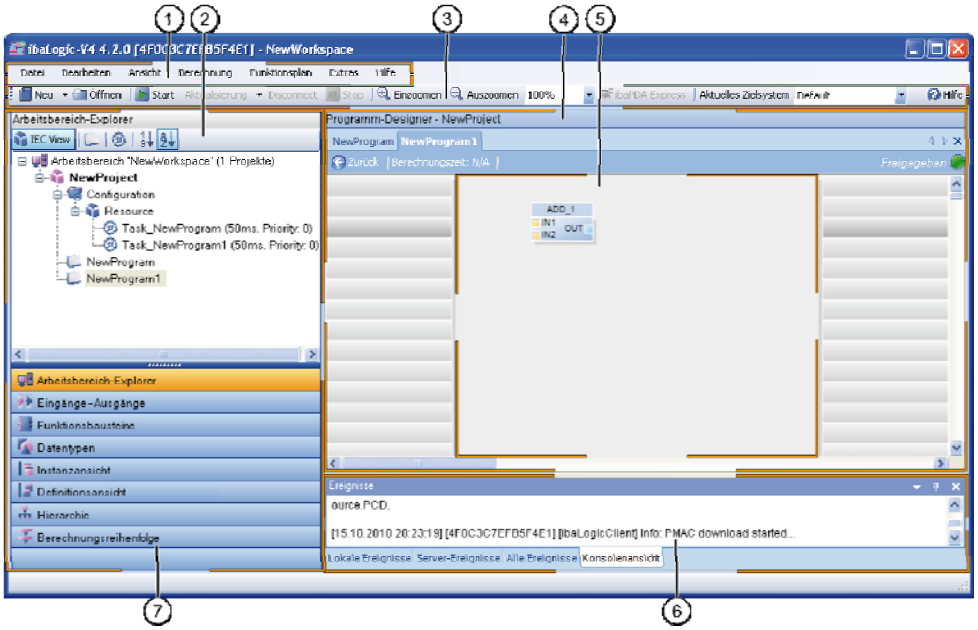


Abbildung 13: Bedienoberfläche

1	Menüleiste	5	Programmierfeld
2	Bereich zum Navigieren	6	Fenster für Ereignisse
3	Symbolleiste	7	Schaltflächen zum Navigieren
4	Programm-Designer		

6.2.1 Menüleiste

Die Menüleiste ist das zentrale Steuerelement des ibaLogic Clients.

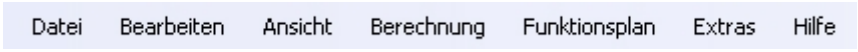


Abbildung 14: Menüleiste

6.2.2 Symbolleiste

Die Symbolleiste ist das untergeordnete Steuerelement des ibaLogic Clients.

Abbildung 15: Symbolleiste

Die Symbole der Symbolleiste sind gleichzeitig Buttons.

6.2.3 Navigationsbereich

Im Navigationsbereich befinden sich die Schaltflächen für:

- ❑ Arbeitsbereich-Explorer
- ❑ Eingänge – Ausgänge
- ❑ Funktionsbausteine
- ❑ Datentypen
- ❑ Instanzansicht
- ❑ Definitionsansicht
- ❑ Hierarchie
- ❑ Berechnungsreihenfolge

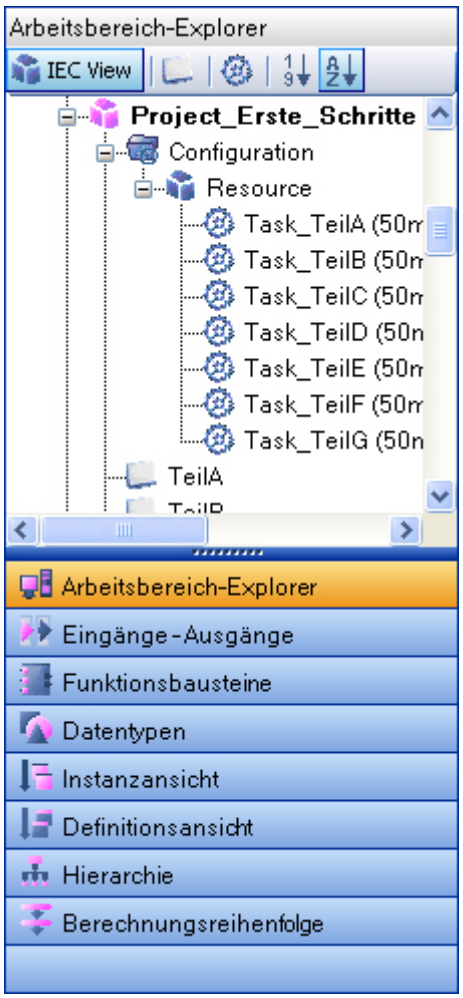


Abbildung 16: Navigationsschaltflächen

6.2.3.1 Ansichten im Arbeitsbereich-Explorer wechseln

In der Menüleiste des Arbeitsbereich-Explorers befinden sich 3 Buttons zum Umschalten der Ansicht des Arbeitsbereich-Explorers. In jeder der 3 Ansichten können die Einträge nach Priorität oder alphabetisch geordnet werden.

IEC View

Diese Ansicht ist die Standard-Ansicht und zeigt den Aufbau des ibaLogic-Arbeitsbereiches entsprechend der Norm IEC 61131-3.

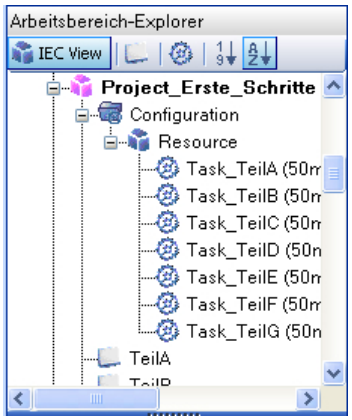


Abbildung 17: IEC View

Hinweis

Entgegen der Norm IEC 61131-3 ist jedem Task nur ein Programm zugeordnet.

Prog View

Kompakte Darstellung von Projekten und Programmen.

Abbildung 18: Prog View

Task View

Darstellung nach Tasks geordnet.

Abbildung 19: Task View

Dabei kann jeweils eine alphabetische oder eine Prioritäten-Ordnung gewählt werden.

Symbol	Erklärung
	Sortiert die Einträge nach Namen.
	Sortiert die Einträge nach ihrer Priorität.

6.2.3.2 Instanzansicht

In der Instanzansicht werden alle im Projekt verwendeten Bausteine nach deren Instanznamen aufgelistet. Getrennt durch einen Doppelpunkt wird der Definitionsname mit angegeben.

Durch einen Doppelklick auf den Instanznamen wird die Programmseite aufgeblendet in welcher der Baustein platziert ist. Der Baustein wird dabei markiert.

Hinweis Unterscheidung Definition – Instanz

In der globalen Bibliothek ist die Definition eines Bausteins abgelegt. Im Programm ist immer eine Instanz (quasi eine Kopie) des Bausteins zu sehen. Der Instanzname wird automatisch gebildet aus „Definitionsname*Index*". Sie können diesen aber auch ändern.

Wenn Sie den Inhalt eines Bausteins ändern, dann ändern Sie immer auch die Definition und die anderen Instanzen.

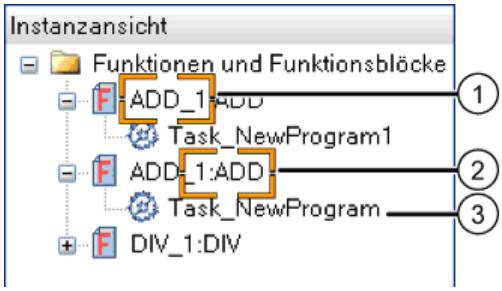


Abbildung 20: Instanzansicht

1 Instanzname
2 Definitionstyp
3 Task, in der dieser Baustein enthalten ist

Ist ein Baustein in einem verschachtelten Makro platziert, wird dies in einer Baumstruktur angezeigt:

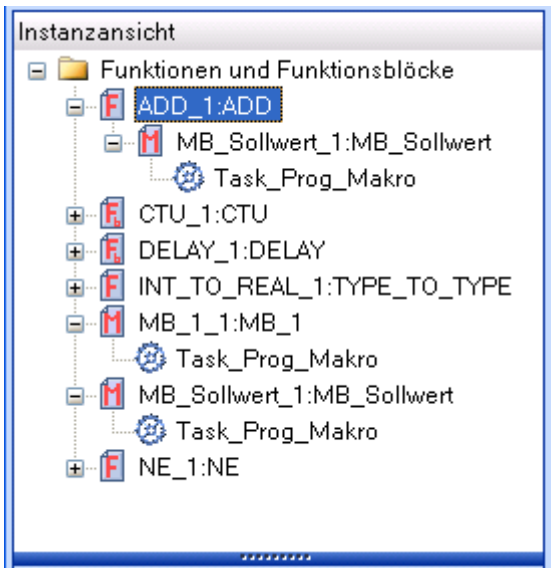


Abbildung 21: Instanzansicht - Baumstruktur

6.2.3.3 Definitionsansicht

In dieser Ansicht werden alle im Projekt vorhandenen Bausteine anhand ihres Definitionsnamens geordnet angezeigt. In der Baumstruktur befinden sich unterhalb des Bausteintyps die Instanzen und darunter evtl. die Makros und zuletzt der Task mit Programmnamen.

Durch einen Doppelklick auf den Instanznamen wird die Programmseite aufgeblendet, in welcher der Baustein platziert ist. Der Baustein wird dabei markiert.

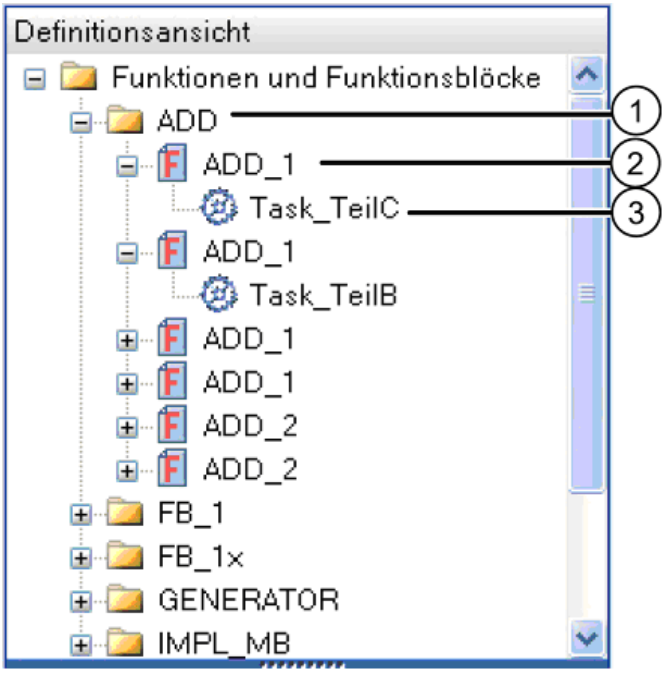


Abbildung 22: Definitionsansicht

1 Definitionsname

2 Instanzname

3 Taskname

6.2.3.4 Hierarchie

Innerhalb der Hierarchieansicht werden alle Instanzen der Bausteine alphabetisch nach Tasks geordnet angezeigt.

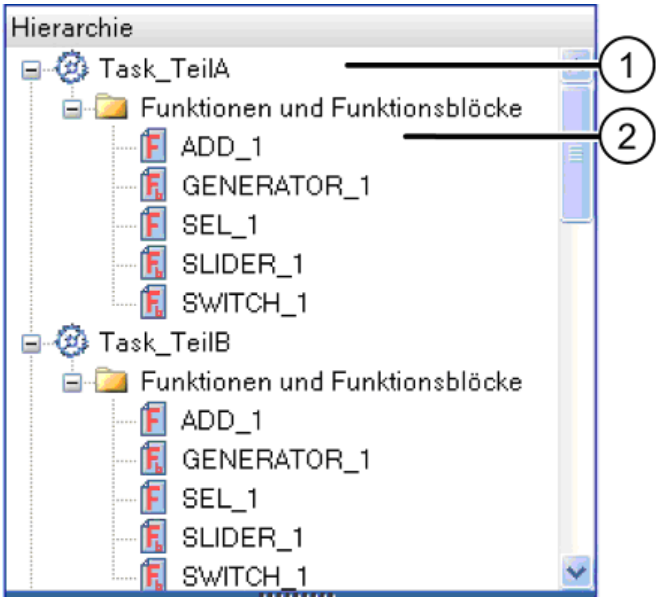


Abbildung 23: Hierarchie-Ansicht

1 Task 2 Instanzname

6.2.3.5 Berechnungsreihenfolge

Die Ansicht „Berechnungsreihenfolge" zeigt die Reihenfolge, in der die Programme und Bausteine innerhalb der Programme berechnet werden.

An oberster Stellen stehen die zuerst berechneten Bausteine.

Beispiel

2 Tasks mit gleicher Intervallzeit aber unterschiedlicher Priorität.

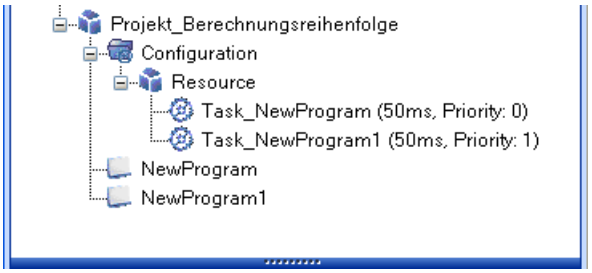


Abbildung 24: 2 Tasks mit gleicher Intervallzeit aber unterschiedlicher Priorität

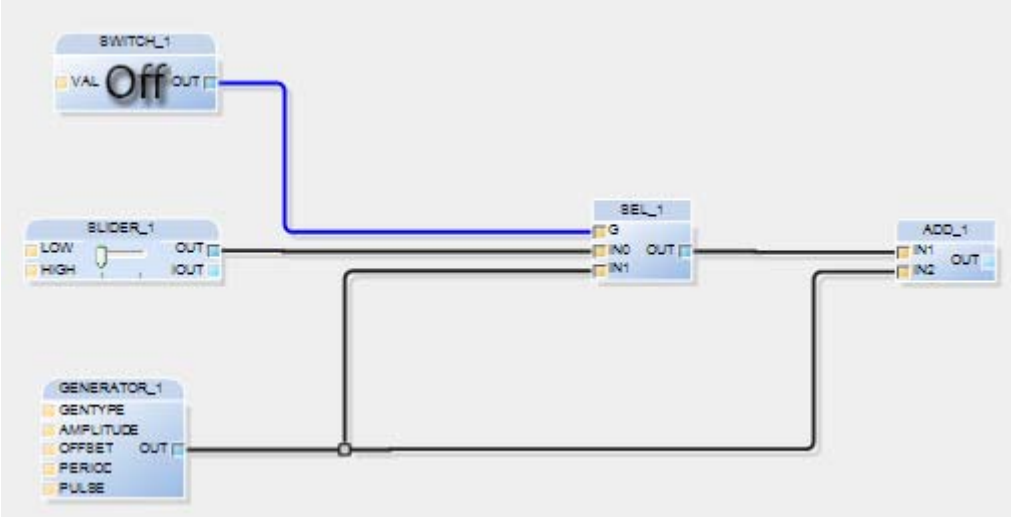


Abbildung 25: NewProgram mit folgendem Inhalt

Die Berechnungsreihenfolge wird wie folgt dargestellt:

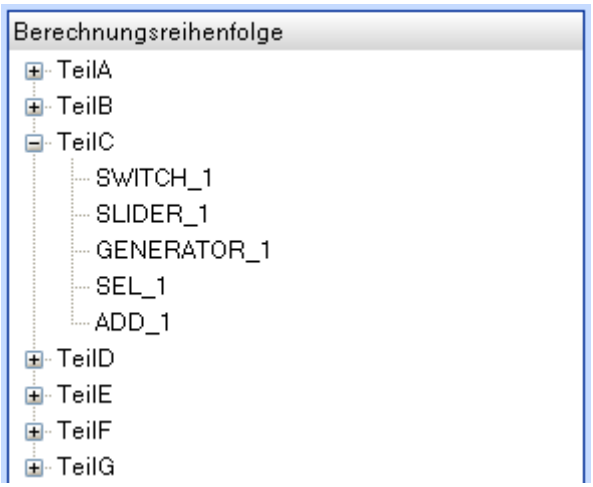


Abbildung 26: Berechnungsreihenfolge

Regeln für Berechnungsreihenfolge

- ❑ Es werden die Programme entsprechend dem Intervall ihres zugeordneten Tasks abgearbeitet.
- ❑ Gleichzeitig zu startende Tasks werden entsprechend ihrer Priorität berechnet, wobei 0 die höchste Priorität ist.
- ❑ Tasks werden nicht unterbrochen. D. h., auch ein höherpriorer Task unterbricht nicht einen gerade laufenden niederprioren.
- ❑ Innerhalb eines Programms erfolgt die Berechnung nach folgenden Regeln:
  - Nach Datenfluss werden zuerst immer die Bausteine berechnet, welche die Daten erzeugen. Danach werden die berechnet, welche die Daten verbrauchen.
  - Sind mehrere unabhängige Zweige in einem Programm, dann gilt zusätzlich die Regel, dass von links oben nach rechts unten berechnet wird. D. h., ein weiter oben stehender Zweig wird zuerst berechnet.
  - In einer Rückkopplungsschleife innerhalb eines Programms oder Makros wird der in der Schleife der am weitesten links oben platzierte Baustein als erster berechnet.

Gefahr!

Gefahr durch Änderungen der Berechnungsreihenfolge!

Das hat zur Folge, dass sich (nur bei Rückkopplungsschleifen) durch das Verschieben von Bausteinen die Berechnungsreihenfolge und damit die Ergebnisse ändern können.

Tipp

Um dies zu vermeiden, wird empfohlen Bausteine in einem Makro zu kapseln und die Rückkopplung außerhalb des Makros vorzunehmen. Damit ist die Berechnungsreihenfolge unabhängig von der Positionierung eindeutig definiert.

Der Makroblock ist ein Baustein. Darin werden die Bausteine nach den oben genannten Regeln berechnet. D. h., der berechnete Ausgang wird erst im nächsten Takt als neuer Eingangswert berücksichtigt.

Tipp

Der Inhalt eines Funktionsblocks wird in einem Zyklus berechnet.
D. h., wenn Sie zum Beispiel innerhalb einer For-Schleife auf einen Eingangskonnektor zugreifen, bekommen Sie in jedem Durchlauf denselben Wert, da der Konnektor erst wieder im nächsten Zyklus neu berechnet wird. Wollen Sie eine Schleife über mehrere Zyklen machen, dann müssen Sie die Laufvariable außerhalb des Bausteins zurückkoppeln.

6.2.4 Programm-Designer

Im Programm-Designer werden Funktionsbausteine platziert und miteinander verbunden.

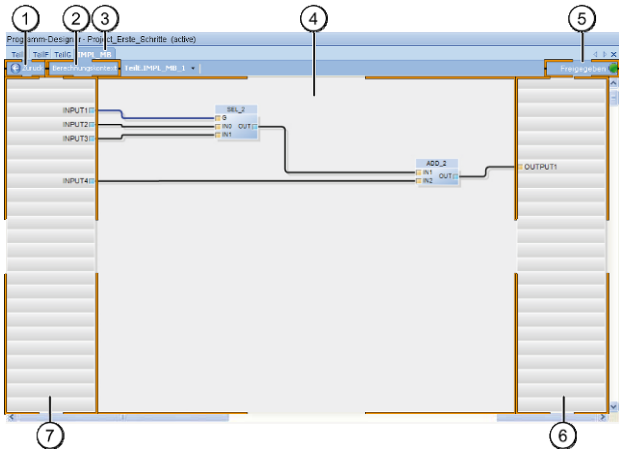


Abbildung 27: Bedienoberfläche Programm-Designer

1	Button <Zurück>	5	Button <Freigeben/Sperren>
2	Berechnungskontext	6	Ausgangsbereich
3	Registerlasche	7	Eingangsbereich
4	Programmierfenster		

6.2.5 Anordnung der Register und Programmierfenster

Die Anordnung kann mit der Maus festgelegt werden.

Folgende Möglichkeiten stehen zur Verfügung:

- ❑ Überlappend
- ❑ Nebeneinander/Untereinander

6.2.5.1 Register anordnen

Zum Anordnen der Register gehen Sie wie folgt vor:

Vorgehen

1. Klicken Sie auf die Registerlasche des betreffenden Programms.
2. Bewegen Sie das Register mit gedrückter Maustaste an die neue Position.

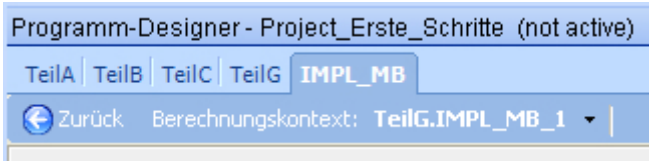


Abbildung 28: Anordnung in Form von Registerkarten

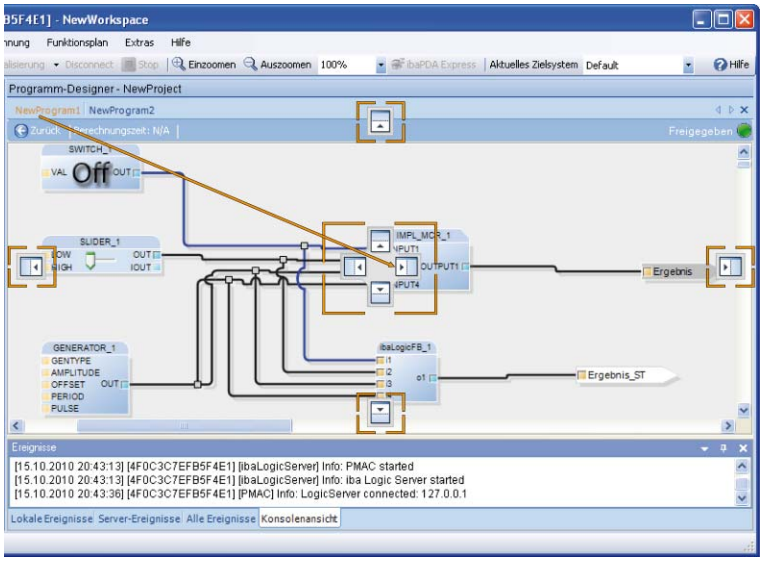
6.2.5.2 Programmierfenster anordnen

Andockfenster sind verschiebbare Fenster, die an beliebige Positionen auf dem Bildschirm verschoben und an den Andockmarken angekoppelt werden können.

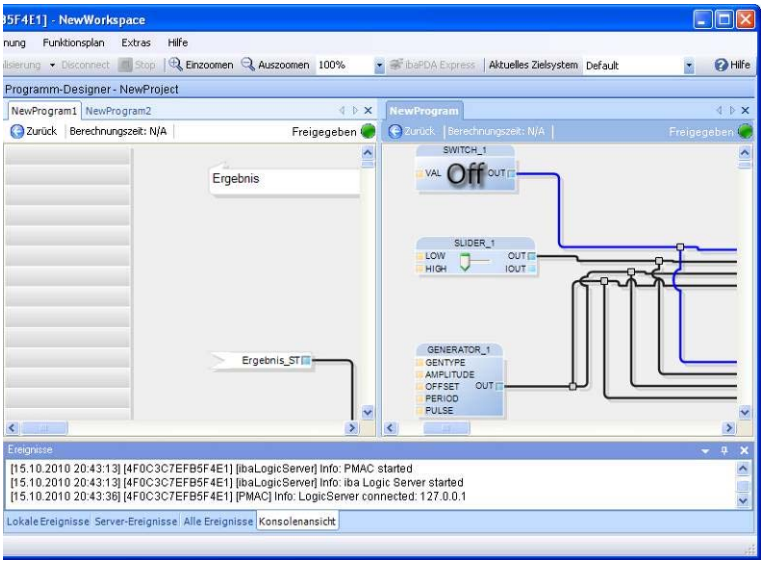
Zum Anordnen der Fenster gehen Sie wie folgt vor:

Vorgehen

1. Zeigen Sie mit der Maus auf das Register des Fensters, das Sie verschieben möchten.
2. Drücken Sie die linke Maustaste und halten Sie diese gedrückt.
3. Bewegen Sie das Fenster an die Andockmarke, an der Sie das Register positionieren möchten.



4. Lassen Sie die Maustaste los. Das Fenster wird an dieser Position verankert.



Hinweis

Programmfenster können nicht frei positioniert werden.

Im Programmierfeld werden die geöffneten Programme und Makros grafisch dargestellt. Da Makroblöcke genauso wie Programme dargestellt werden, werden sie in den folgenden Kapiteln nicht beschrieben.

Symbolleiste

In der Symbolleiste können folgende Funktionen genutzt werden:

- Zurück (

)
- Berechnungskontext (nur bei Makros)
- Berechnungszeit (nur im Online-Modus)

Zurück

Diese Taste dient zum Navigieren im Programmfenster. Das zuletzt angewählte Programm/Makro wird aufgeblendet.

Berechnungskontext (nur bei Makros)

Dient zur Anzeige der Programm- bzw. Makroebene, in der sich der aktuelle Plan befindet. Da sich die Instanzen eines Makroblocks mehrfach in einem oder in verschiedenen Programmen mit unterschiedlichen Aufrufparametern befinden können, können Sie sehen, unter welchem Programm und in welcher Instanz sich das aktuell geöffnete Makro befindet.

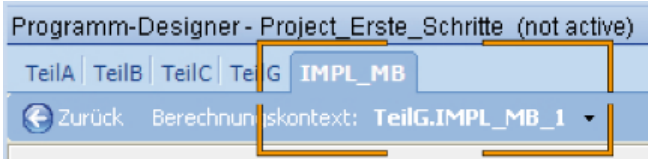
Liegt ein Makro in einem anderen Makro (Verschachtelung), wird der ganze Hierarchiezweig durch Punkte getrennt angezeigt:

Anzeige: Programm_name.Makro_1.Makro_2….

Innerhalb eines geöffneten Makros können Sie den Kontext auswählen, d. h. von einer Instanz in eine andere umschalten. Die in der Online-Ansicht zu sehenden Werte beziehen sich immer auf die angewählte Instanz.

1. Klicken Sie hierfür auf die Pfeilschaltfläche, rechts neben dem Makro-Instanznamen.

2. Sie können aus dem Makro heraus die aufrufende Ebene anwählen, wenn Sie direkt den Instanznamen im Berechnungskontext anklicken.



Der Bereich des Programmfeldfensters, der diese Makroinstanz enthält, wird aufgeblendet und der Makroblock wird markiert.

Berechnungszeit (nur im Online-Modus)

Sie wird in jedem Programm als Prozentwert im Verhältnis zum Taskintervall und als absoluter Wert in ms (gleitender Mittelwert) angezeigt.

6.2.5.3 Navigieren im Programm-Designer

Sie haben mehrere Möglichkeiten in einem Programm zu navigieren:

- ❑ Zoomen
- ❑ Programmübersicht
- ❑ Tastenkombinationen

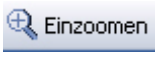
Zoomen

Sie haben die Möglichkeit, die Größe des Programmierfeldes im Programm-Designer zu vergrößern (Einzoomen) bzw. verkleinern (Auszoomen).

Folgende Möglichkeiten zum Vergrößern oder zum Verkleinern stehen zur Verfügung:

- ❑ Verwenden der Schaltflächen <Einzoomen> / <Auszoomen>
- ❑ Verwenden der Auswahlliste
- ❑ Verwenden der Tastenkombination

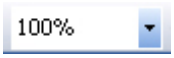
Schaltflächen

➲ Klicken Sie auf den Button 

oder 

in der ibaLogic Client-Symbolleiste, um die Ansicht im Programmierfeld des Programm-Designers bei jedem Klick um 20 % zu vergrößern oder zu verkleinern.

Auswahlliste

➲ Wählen Sie in der Auswahlliste 

zwischen den 5 voreingestellten Zoom-Werten aus.

oder

➲ Geben Sie einen Zoomfaktor mit der Tastatur zwischen 25 und 200 Prozent in der Auswahlliste ein.

Tastenkombination

➲ Drücken Sie <Strg> und drehen Sie gleichzeitig das Scroll-Rad der Maus.

Der minimale Zoomfaktor hängt von der Bildschirmauflösung ab.

Programmübersicht

Da eine Programmseite normalerweise über eine Bildschirmseite hinausgeht und je nach Zoom-Faktor nur ein Teilfenster des Programms zu sehen ist, gibt es die Programmübersicht als Orientierungshilfe.

Programmübersicht öffnen und anwenden

Vorgehen

1. Wählen Sie im Hauptmenü „Ansicht - Programmübersicht".

   Das Fenster „Programmübersicht" wird im Vordergrund des aktuellen Programmfensters in Miniaturansicht geöffnet.

   Der sichtbare Ausschnitt wird als transparentes Rechteck dargestellt.

2. Verschieben Sie das Rechteck - den sichtbaren Ausschnitt des Programms - mit der Maus, bis Sie den gewünschten Programmbereich im Programmierfeld sehen.

Anmerkung

Das Übersichtsfenster kann beliebig positioniert werden.

Auch außerhalb des Programmierfeldes oder an dessen Rand angedockt werden.

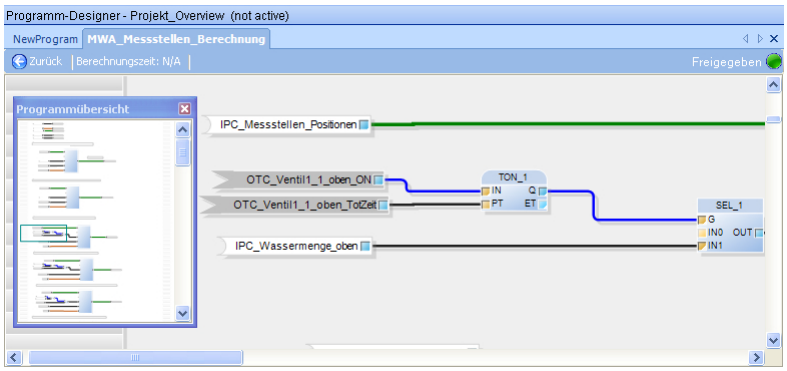


Abbildung 29: Programmierfeld mit eingeblendetem Programmübersichtsfenster

Navigieren im Programmierfeld

Die Navigation im Programmfenster erfolgt im Wesentlichen mit der Maus.

Hier sind die Mausfunktionen zusammenfassend erläutert:

- ❑ Scroll-Rad: Verschieben des sichtbaren Ausschnitts nach oben/unten
- ❑ Scroll-Rad + <Shift>: Verschieben des sichtbaren Ausschnitts nach links/rechts
- ❑ Scroll-Rad + <Strg>: Einzoomen/Auszoomen

Seite nach unten erweitern

Die Programmseite hat standardmäßig eine Breite von 2500 Pixeln und eine Höhe von 10000 Pixeln. Der sichtbare Ausschnitt ist abhängig von der Auflösung des Bildschirms.

Ist die Seitengröße nicht ausreichend, erweitern Sie die Seite nach unten.

Vorgehen

1. Öffnen Sie das Kontextmenü durch einen Klick mit der rechten Maustaste auf einen freien Bereich im Programmierfeld.
2. Wählen Sie im Kontextmenü „Seite erweitern (vertikal)".

Ergebnis

Ab der aktuellen Mauszeigerposition wird ein Bereich von 800 Pixelzeilen eingefügt.

Anmerkung

Wichtiger Hinweis

Achten Sie darauf, dass in der horizontalen Linie des Mauszeigers keine Bausteine liegen. Verbindungslinien, die diese gedachte waagrechte Linie kreuzen, werden verlängert.

Hinweis

Das Entfernen einer leeren Seite innerhalb des Tasks ist per Funktion nicht möglich. Ein leerer Bereich am unteren Ende des Tasks wird automatisch beim Laden eines Projekts entfernt.

Tipp

Ein leerer Zwischenraum innerhalb der Tasks kann durch starkes Verkleinern der Darstellung und gemeinsames Verschieben der Objekte erreicht werden.

6.2.6 Zugriff synchronisieren (Button <Freigegeben>/<Gesperrt>)

Im Multi-Client-Betrieb kann der Zugriff auf Fenster und Dialoge durch die Buttons <Freigegeben>/<Gesperrt> synchronisiert werden.

Abbildung 30: Button <Freigegeben>

Abbildung 31: Button <Gesperrt>

6.2.7 Ereignisfenster

Das Fenster „Ereignisse" unter dem Fenster „Programm-Designer" dokumentiert die Programmaktionen und mögliche Kollisionen.

Das Ereignisfenster ist mittels Reiter in 4 Ansichten geteilt.

Folgende Ansichten stehen zu Auswahl:

- ❑ Lokale Ereignisse
- ❑ Server-Ereignisse
- ❑ Alle Ereignisse
- ❑ Konsolenansicht

In den Ansichten „Lokale Ereignisse", „Server-Ereignisse" und „Alle Ereignisse" werden die Ereignis-Symbole verwendet.

Ereignis-Symbol	Ereignis	Niveau	Beschreibung
	Info	Info	Zeigt eine Info an.
	Warnung	Warning	Zeigt eine Warnung an.
	Fehler	Exception	Zeigt einen Fehler an.

6.2.7.1 Lokale Ereignisse

Die Ansicht „Lokale Ereignisse" wird dazu verwendet, um Ereignisse des Clients formatiert anzuzeigen.

Abbildung 32: Ereignisfenster – „Lokale Ereignisse"

6.2.7.2 Server-Ereignisse

Die Ansicht „Server-Ereignisse" wird zum Anzeigen von Ereignisses des Servers verwendet.

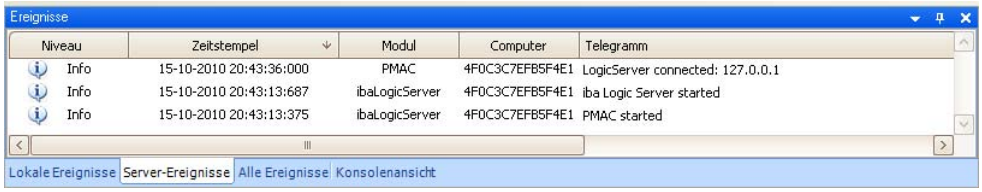


Abbildung 33: Ereignisfenster – „Server-Ereignisse"

6.2.7.3 Alle Ereignisse

Die Ansicht „Alle Ereignisse" wird dazu verwendet, um Ereignisse des Clients und des Servers formatiert anzuzeigen.

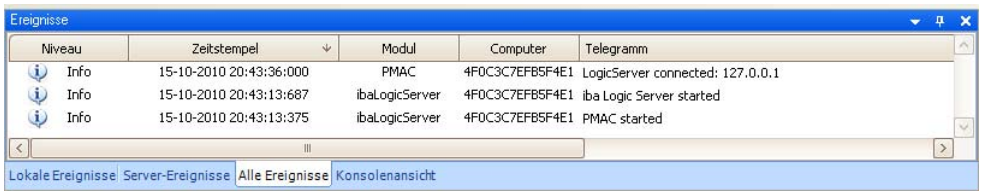


Abbildung 34: Ereignisfenster – „Alle Ereignisse"

6.2.7.4 Konsolenansicht

Die Ansicht „Konsolenansicht" wird dazu verwendet, um alle Ereignisse sortiert nach Auftreten mit zugehöriger Erläuterung als einfachen Text anzuzeigen.

Abbildung 35: Ereignisfenster – „Konsolenansicht"

6.3 Arbeitsbereich

Ein Arbeitsbereich wird dazu verwendet, um die Programme und Projekte sortiert ablegen zu können.

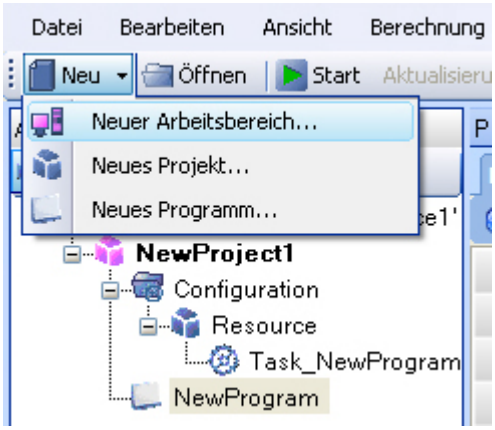
6.3.1 Arbeitsbereich anlegen

Voraussetzung

Sie haben keinen Arbeitsbereich geöffnet.
Geöffnete Arbeitsbereiche schließen Sie über „Datei - Arbeitsbereich schließen".

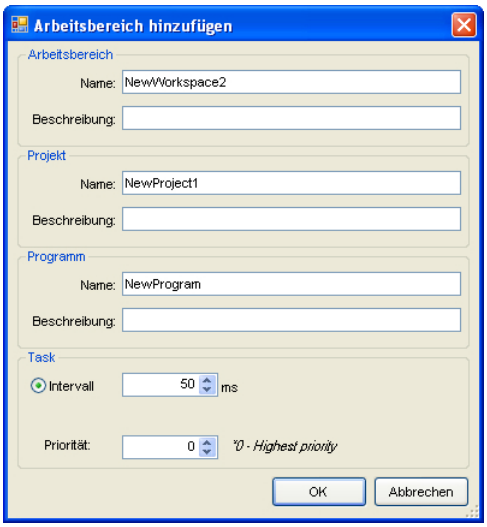
Vorgehen

1. Klicken Sie in der Symbolleiste auf den Pfeil des Symbols <Neu>.
2. Wählen Sie aus der Liste den Menüeintrag „Neuer Arbeitsbereich".



Dialog „Arbeitsbereich hinzufügen" wird angezeigt.

3. Vergeben Sie einen Namen und eine Beschreibung für den Arbeitsbereich.
ibaLogic legt automatisch selbst ein neues Projekt sowie einen Task an. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen.
Legen Sie bei Bedarf auch für den Task die Intervallzeit und dessen Priorität fest.

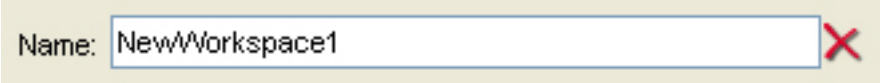


Anmerkung

Sie haben die Möglichkeit, alle Einstellungen später zu ändern.

Die Einstellungsmöglichkeit erreichen Sie über das jeweilige Kontextmenü „Eigenschaften."

Ein schon vergebener Name wird durch ein „X" nach dem Eingabefeld angezeigt.



Die Felder für die Beschreibung können Sie mit beliebigem Kommentar füllen. Diese Kommentare werden über dem Objekt als Tooltip angezeigt.

Hinweise

Die Namen müssen der IEC-Norm entsprechen. Weitere Informationen finden Sie in "Namenskonventionen , Seite 288".

Tipp

Die standardmäßige Vorbesetzung für die Namen können Sie unter „Extras – Optionen – Editoren – Arbeitsbereiche" einstellen.

6.3.2 Arbeitsbereich öffnen

Vorgehen

1. Drücken Sie auf den Button <Öffnen>. Das Fenster „Arbeitsbereich öffnen" wird angezeigt.
2. Wählen Sie den gewünschten Arbeitsbereich aus und klicken Sie auf <OK>.

6.3.3 Geöffneten Arbeitsbereich schließen

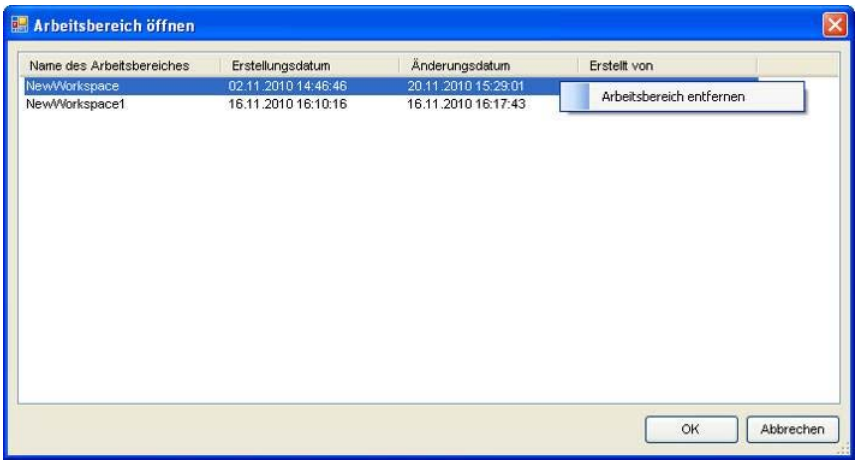
Wählen Sie im Menü „Datei – Arbeitsbereich schließen".

6.3.4 Arbeitsbereich aus der Datenbank entfernen

Vorgehen

1. Klicken Sie auf den Button „Öffnen". Das Fenster „Arbeitsbereich öffnen" wird angezeigt.
2. Markieren Sie den Arbeitsbereich, der entfernt werden soll.

3. Mit einem rechten Mausklick öffnen Sie das Kontextmenü „Arbeitsbereich entfernen".



4. Klicken Sie auf „Arbeitsbereich entfernen". Der Dialog „Arbeitsbereich entfernen" wird angezeigt.
5. Bestätigen Sie mit <Ja>.
6. Bestätigen Sie den Vorgang mit <OK>. Der Dialog wird geschlossen.

6.4 Projekte eines Arbeitsbereichs

ibaLogic legt bei der Erstellung eines neuen Arbeitsbereichs automatisch ein Projekt an. Sie haben die Möglichkeit, mehr als ein Projekt innerhalb eines Arbeitsbereiches anzulegen.

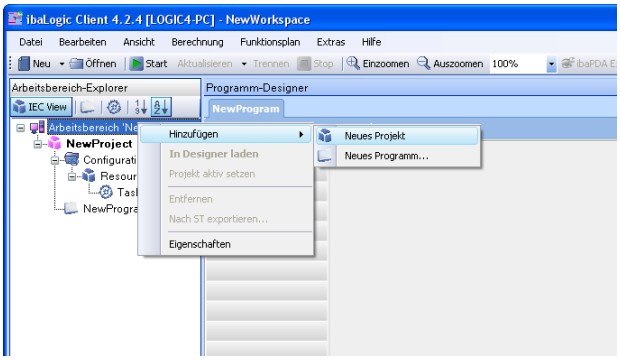
Voraussetzung

- ❑ Sie haben den ibaLogic Server sowie den ibaLogic Client gestartet.
- ❑ Sie haben mindestens einen Arbeitsbereich angelegt.

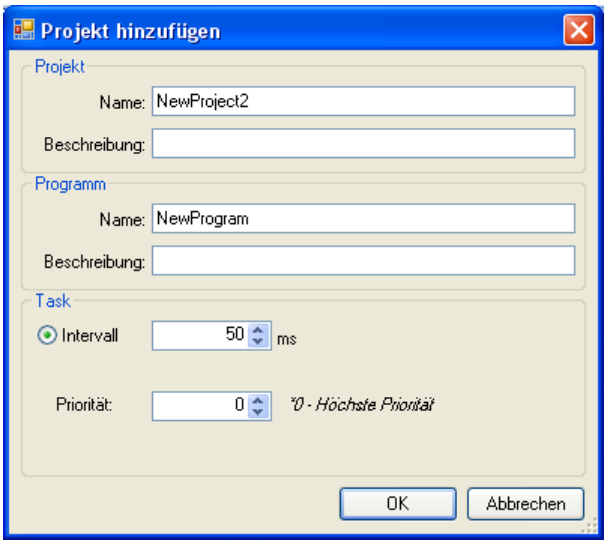
6.4.1 Projekt anlegen

Vorgehen

1. Klicken Sie im Arbeitsbereich-Explorer mit der rechten Maustaste auf den Arbeitsbereichsnamen zu dem das Projekt hinzugefügt werden muss.
2. Wählen Sie im Kontextmenü „Hinzufügen - Neues Projekt".



Der Dialog „Projekt hinzufügen" wird angezeigt.
Der sich öffnende Dialog ist der projektbezogene Teilbereich aus dem Arbeitsbereich-Dialog.
Benennen Sie das Projekt und das Programm. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen. Stellen Sie bei Bedarf für den Task dessen Intervallzeit und Priorität ein.



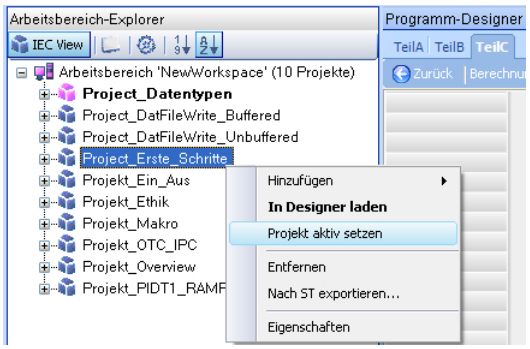
6.4.2 Projekt aktiv schalten

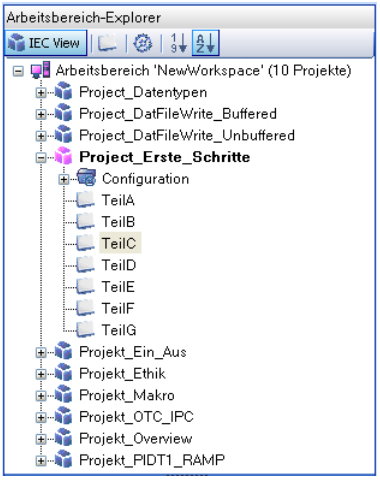
Hinweis

Von mehreren Projekten innerhalb eines Arbeitsbereiches kann nur eines aktiv sein.

Vorgehen

1. Öffnen Sie das Kontextmenü des zu aktivierenden Projektes mit der rechten Maustaste.
2. Wählen Sie „Projekt aktiv setzen" aus.





Ergebnis

Der Name des aktiven Projekts wird im Arbeitsbereich-Explorer auf „**Fett**" gesetzt und das zugehörige Symbol wechselt von der Farbe Blau in die Farbe Pink.

Hinweise

Wenn ein Projekt im Online-Modus ist (Programmierfeld-Hintergrund Pink), dann kann kein anderes Projekt aktiviert werden.

Durch Aktivieren werden die Programme des Projekts nicht automatisch in den Programmierbereich geladen, sondern müssen bei Bedarf von Hand angewählt werden.

Die Buttons in der Symbolleiste und im Arbeitsbereich-Explorer gelten nur für das aktive Projekt. Diese sind:

- ❑ <Start>
- ❑ <Stop>
- ❑ <Trennen>
- ❑ <Aktuelles Zielsystem>
- ❑ <I/O-Konfigurator>

6.4.3 Projekt im Programmier-Designer laden

Unabhängig davon, ob ein Projekt aktiv ist oder nicht, können Sie Programme in den Programmierbereich laden.

Vorgehen

1. Markieren Sie das Projekt, das Sie im Designer laden möchten.
2. Wählen Sie im Kontextmenü „In Designer laden" aus.

Ergebnis

Die geladenen Programme werden in den Registerlaschen am oberen Rand des Programmierbereichs angezeigt.

6.4.4 Projekteigenschaften bearbeiten

In den Projekteigenschaften können Sie den Namen und das Beschreibungsfeld ändern. Im Textfeld „Beschreibung" können Sie zusätzliche Informationen zum Projekt eingeben.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf das Projekt, dessen Projekteigenschaften Sie bearbeiten möchten.
2. Wählen Sie aus dem Kontextmenü „Eigenschaften" aus.
   Das Fenster „Arbeitsbereich bearbeiten" wird angezeigt.
3. Bearbeiten Sie die Eigenschaften des Projekts.
4. Klicken Sie auf <OK>.

6.4.5 Projekt entfernen

Das gewählte Projekt wird aus dem Arbeitsbereich und gleichzeitig aus der Datenbank entfernt.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf das Projekt, dass Sie aus dem Arbeitsbereich entfernen möchten.
2. Wählen Sie im Kontextmenü „Entfernen" aus. Der Dialog „Projekt entfernen" wird angezeigt.
3. Wenn Sie sicher sind, dass Sie dieses Projekt aus dem Arbeitsbereich entfernen möchten, dann klicken Sie auf <OK>.
4. Ist das Projekt gelöscht, wird zur Bestätigung der Dialog „Projekt gelöscht" angezeigt.
5. Klicken Sie abschließend auf <OK>.

6.5 Tasks/Programme

ibaLogic legt bei der Erstellung eines neuen Projekts automatisch ein Programm an. Es können weitere Programme in einem Projekt angelegt werden.

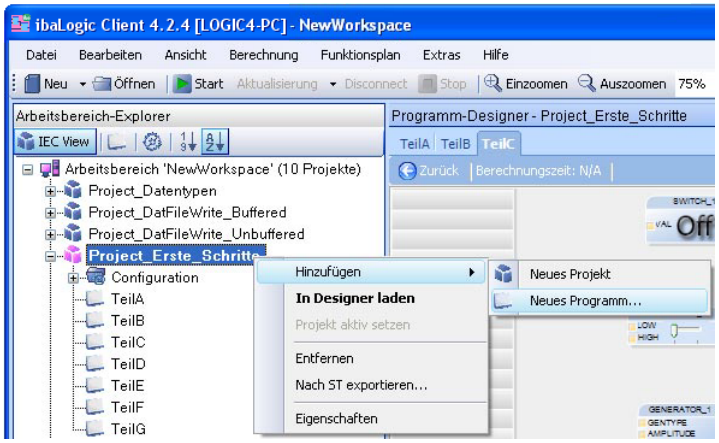
Hinweise

Beachten Sie, dass Sie den letzten Task nicht löschen können. Ein Task pro Projekt muss immer vorhanden sein.

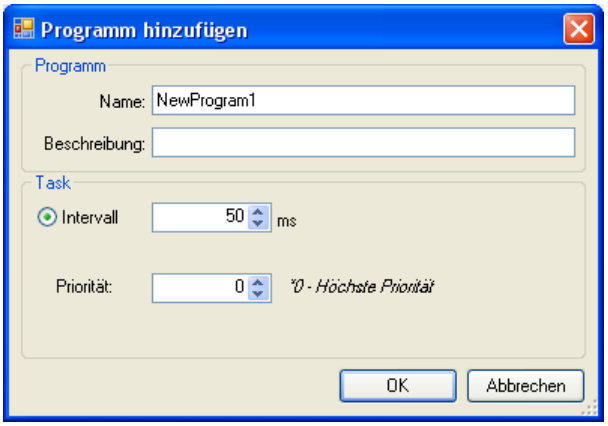
6.5.1 Task/Programm anlegen

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf das Projekt, in dem Sie ein neues Programm anlegen möchten.
2. Wählen Sie im Kontextmenü „Hinzufügen - Neues Programm".



Der Dialog „Programm hinzufügen" wird angezeigt.



3. Vergeben Sie einen Namen und eine Beschreibung für das Programm. Gleichzeitig legen Sie einen dazugehörigen Task an. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen. Weitere Informationen siehe Definition in „Namenskonventionen , Seite 288".
4. Klicken Sie auf <OK>, um das neue Programm hinzuzufügen.

Jedem Programm wird genau ein Task zugeordnet. Folgende Task-Parameter stehen zur Auswahl:

- ❑ Intervallzeit
- ❑ Priorität

Intervall

Das zum Task gehörende Programm wird, genau nach der angegebenen Intervallzeit wieder gestartet.

Dauert im Extremfall die Berechnungszeit (bzw. die Berechnung aller Programme) länger als die angegebene Intervallzeit, dann liegt eine Überlastung vor. Wie damit verfahren wird, ist in „Zeitverhalten , Seite 239" erläutert.

Vorbesetzung ist 50 ms. Das kleinste mögliche Intervall ist 1 ms, aber die Intervallzeit kann nicht kleiner sein als die im I/O-Konfigurator eingestellte Zeitbasis.

Hinweise

Die Standard-Vorbesetzungen von Programmname und Intervallzeit sind unter „Extras – Optionen – Editoren - Arbeitsbereich" änderbar.

Priorität

Jeder Task wird mit einer Priorität versehen. „0" bedeutet höchste Priorität.

Beachten Sie, dass kein Task durch einen höherprioren unterbrochen wird. Die Priorität ist nur für die Bearbeitungsreihenfolge relevant. Weitere Informationen siehe „Zeitverhalten , Seite 239".

Hinweise

Sie können den Task entweder alphabetisch oder nach Prioritäten geordnet im Arbeitsbereich anzeigen.

Benutzen Sie dazu die Symbole am oberen Rand des Navigationsbereichs 

6.5.2 Task/Programm öffnen

Doppelklicken Sie auf den Programmnamen im Arbeitsbereich-Explorer.

6.5.3 Task/Programm-Eigenschaften ändern

Ändern der Eigenschaften vorhandener Tasks/Programme.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den entsprechenden Task oder das Programm.
2. Wählen Sie im Kontextmenü „Eigenschaften" aus. Das Fenster „Arbeitsbereich bearbeiten" wird angezeigt.
3. Ändern Sie die Eigenschaften.
4. Klicken Sie abschließend auf <OK>.

6.5.4 Task/Programm entfernen

Entfernen eines Programms/Tasks aus einem Arbeitsbereich.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den entsprechenden Task oder das Programm.
2. Wählen Sie im Kontextmenü „Entfernen" aus. Der Dialog „Bestätigen" wird angezeigt.
3. Wenn Sie das Programm entfernen möchten, dann klicken Sie auf <Ja>.

Hinweise

Beachten Sie, dass Sie den letzten Task nicht löschen können. Ein Task pro Projekt muss immer vorhanden sein.

6.5.5 Programme importieren / exportieren

Um komplette Programme zwischen Projekten aus unterschiedlichen Datenbanken auszutauschen, gibt es die Export/Import-Funktionen.

Vorgehen Export

1. Klicken Sie mit der rechten Maustaste auf das entsprechende Programm.
2. Wählen Sie im Kontextmenü „Nach ST exportieren“ aus. Der Dialog „Export“ wird angezeigt.
3. Rufen Sie den Filebrowser auf  , wählen Sie ein Verzeichnis, geben Sie einen Dateinamen ein und klicken Sie auf "Sichern".
4. Aktivieren Sie die Option "Mit grafischen Zusatzinformationen…" und klicken Sie auf OK.

Vorgehen Import

1. Schalten Sie die Berechnung aus, da der Import nur im Offline-Modus zur Verfügung steht.
2. Wählen Sie unter Hauptmenu "Datei - Import - Structured Text".
   Der Dialog „Structured Text Import“ wird angezeigt.
3. Rufen Sie den Filebrowser auf und wählen Sie Verzeichnis und Datei (mit Dateiendung *.il4) und klicken Sie auf "Öffnen".
4. Wählen Sie die Option "Definition von Funktionsbausteinen als neu importieren" und klicken auf OK.

Beim Import werden die Namen der zu importierende Bausteine und Datentypen mit den bereits vorhandenen Namen verglichen. Sind Namen bereits vergeben, so werden in Abhängigkeit von der Option: "Definition von Funktionsbausteinen als neu importieren" die vorhandenen Definitionen überschrieben (TRUE) oder es wird eine neue Definition mit Namen+Index angelegt (FALSE).

Das importierte Programm wird zusammen mit dem Task neu angelegt. Ist der Programmname schon vergeben, werden sie mit Namen+Index angelegt.

Beim Importieren eines Projektes werden alle im Projekt enthaltenen Programme und Tasks neu angelegt, evtl. mit "name+index".

Hinweis

Export/Import-Beschreibung für **Bausteine** siehe "Bausteine exportieren , Seite 94" bzw. "Bausteine importieren , Seite 95".

6.6 Ein- und Ausgänge projektieren

Eingänge und Ausgänge sind projektierte Signale zur Peripherie.

In der Programmierumgebung arbeiten Sie mit „virtuellen" Signalen. Diese müssen Sie in einem Zuweisungsprozess auf die tatsächlich vorhandenen Hardware-Signale rangieren.

Diese Zuweisung können Sie aus Sicht der Programme oder aus Sicht der Hardware, für einzelne Signale oder gruppenweise durchführen.

Wichtiger Hinweis

Weitere Informationen siehe, „I/O-Konfiguration , Seite 177".

Im Navigationsbereich des I/O-Konfigurators sind die Ein- und Ausgänge der Hardware in einer Baumstruktur angeordnet.

Die Hierarchie-Ebenen sind:

Richtung → Gruppe → Signalunterteilung → Signale

- Richtung:
  „Eingänge" bzw. „Ausgänge"
- Gruppe:
  entweder manuell angelegter Gruppenname oder aus der Signalzuweisung übernommener Modulname.
- Signalunterteilung:
  Die Signale werden in Analog- und Digitalsignale aufgeteilt.

- Signale:
  - entweder manuell angelegter Signalname oder aus der Signalzuweisung übernommene Bezeichnung,
  - dahinter in Klammern der Datentyp,
  - dann rechts vom Pfeil (->) der Hardwaresignalname

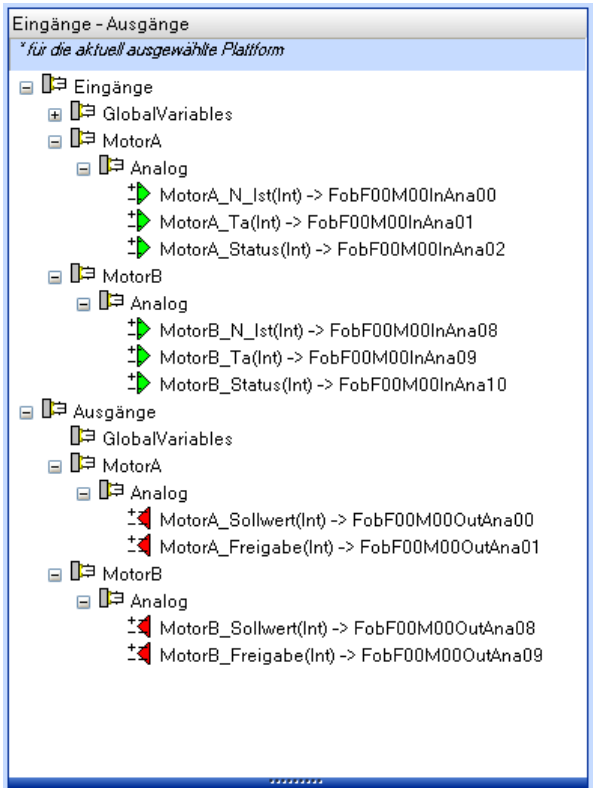


Abbildung 36: Eingänge - Ausgänge

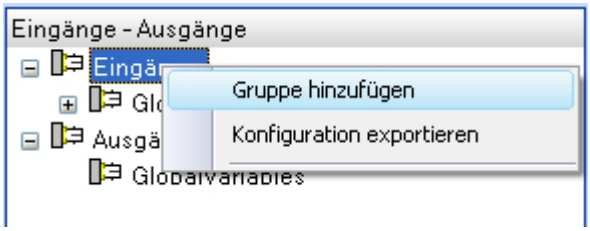
6.6.1 Signale anlegen

Signale werden Gruppen zugeordnet. Dies ermöglicht eine sinnvolle Strukturierung.

6.6.1.1 Gruppe definieren

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf „Eingänge - Ausgänge" und wählen den Menüpunkt "Gruppe hinzufügen" aus.



2. Im Dialog "Gruppenname einstellen" legen Sie den Namen für die neue Gruppe fest..

Ergebnis

Die Neue Gruppe wird unter "Eingänge" sowie "Ausgänge" angelegt.
Nicht benötigte Gruppen können mit Hilfe des Kontextmenüs bzw. mit der Entf-Taste wieder gelöscht werden.

Beispiel

Gruppen: MotorA, MotorB

Eingangssignale: Drehzahl-Istwert, Motor-Temperatur

Ausgangssignale: Sollwert, Parameter

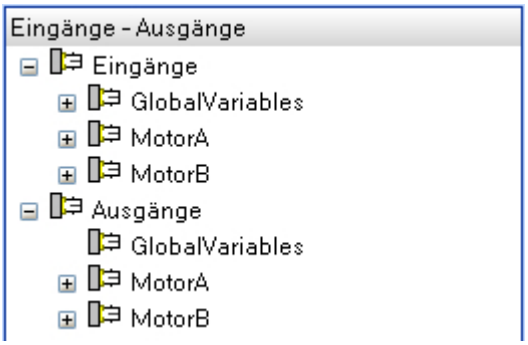


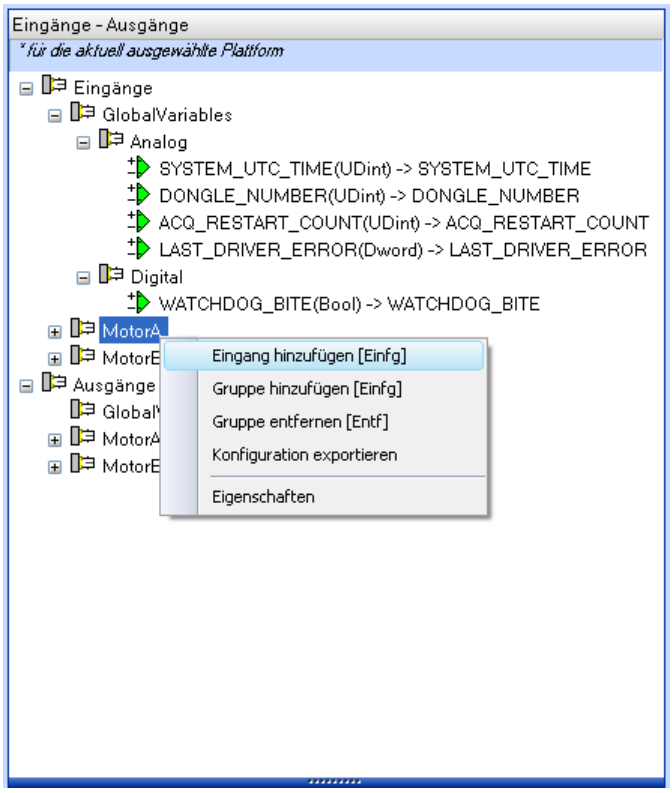
Abbildung 37: Ansicht „Eingänge – Ausgänge"

6.6.2 Signale definieren

Vorgehen

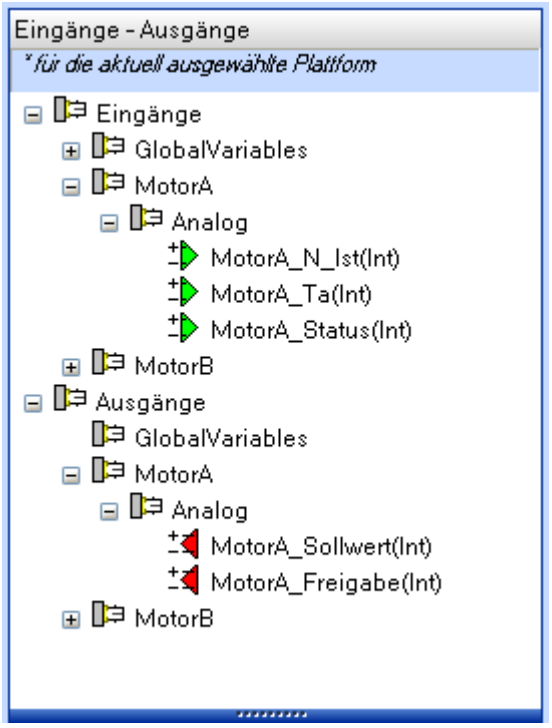
1. Klicken Sie mit der rechten Maustaste auf eine Eingangs- oder Ausgangsgruppe.
2. Wählen Sie „Eingang hinzufügen" oder „Ausgang hinzufügen" im Kontextmenü aus.

   Es öffnet sich der Dialog „Ein- und Ausgänge bearbeiten".
3. Vergeben Sie einen Signalnamen, Datentyp und evtl. eine Beschreibung. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen.



Beim Datentyp ist zu beachten, dass dieser von der Peripherie zur Verfügung gestellt wird. Wenn der Analogwert z. B. von einem ibaPADU-8 verwendet wird, dann sind es immer Integer-Werte.

Für unser Beispiel würde sich folgendes Bild ergeben:



Die Signale sind noch keiner Hardware zugeordnet.

6.6.3 Vorhandene Signale bearbeiten

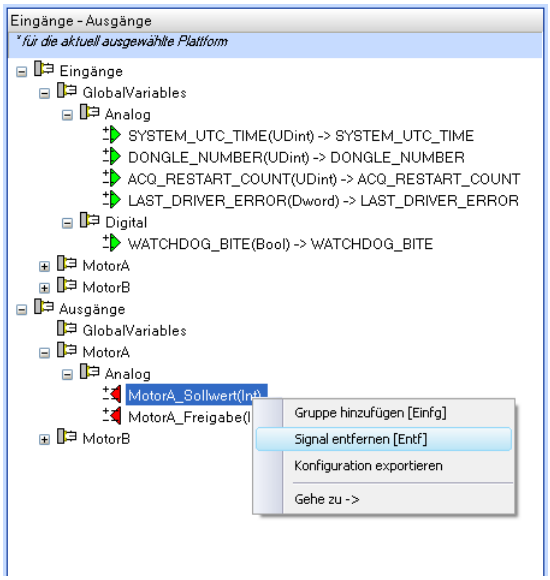
Vorgehen

1. Doppelklicken Sie auf ein definiertes Signal.
   Es öffnet sich der Bearbeitungsdialog.
2. Vergeben Sie einen Signalnamen, Datentyp und evtl. eine Beschreibung. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen.
3. Klicken Sie auf <OK>, um die Änderungen zu übernehmen.

6.6.4 Signale entfernen

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf ein definiertes Signal.
   Es wird das Kontextmenü geöffnet.
2. Wählen Sie im Kontextmenü „Signal entfernen".



Hinweis

Sie können Ein- bzw. Ausgänge nur dann bearbeiten und entfernen, wenn Sie diese noch nicht im Programm verwenden, d. h. sie noch nicht in der Ein- bzw. Ausgangsleiste zu sehen sind.

Hinweis

Die Zuweisung der hier definierten Signale an die zur Verfügung stehende Hardware wird in "Signalzuweisung , Seite 185" beschrieben.

Wichtiger Hinweis

Die Schnittstellenkarten fremder Hersteller liefern zum Teil Signale aus zusammengesetzten Daten und nicht nur einzelne Signale mit den elementaren Datentypen REAL und INT.
Für die Profibus-Masterkarte SST wird hier eine Datenstruktur als Ein-/Ausgang erzeugt, die von der Struktur der Konfiguration der Slaves abhängt (GSD-Datei). Für die Verwendung der darin enthaltenen Signale muss diese Struktur auch in ibaLogic definiert werden.

Auf der Liefer-CD ist ein dokumentiertes Beispiel für den Anschluss der Profibus-Masterkarte beigelegt.

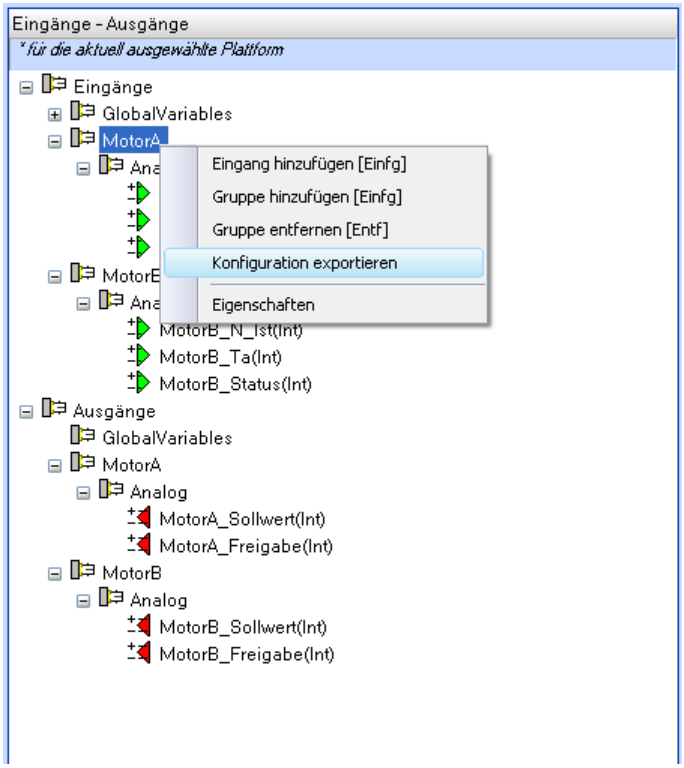
6.6.5 Signale exportieren/importieren

Oft stehen die virtuellen Signalnamen schon in externen Dokumenten zur Verfügung. Um diese nutzen zu können, verwenden Sie die Export- und Importfunktion.

Exportieren der gesamten I/O-Konfiguration

Vorgehen

1. Öffnen Sie das Kontextmenü einer Gruppe bzw. eines Signals und wählen Sie den Menüpunkt "Konfiguration exportieren" aus.

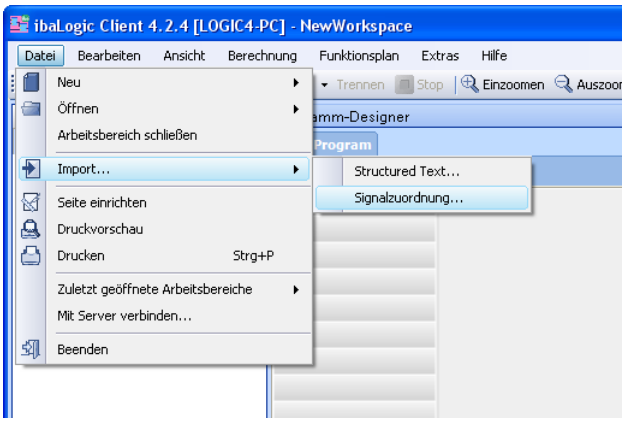


2. Öffnen Sie den Export z. B. in einem Tabellenkalkulationsprogramm.
3. Tragen Sie die virtuelle Gruppenbezeichnung und Signalnamen in die Spalten Gruppe und Symbole ein bzw. ändern Sie die vorhandenen Namen.

```
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;
```

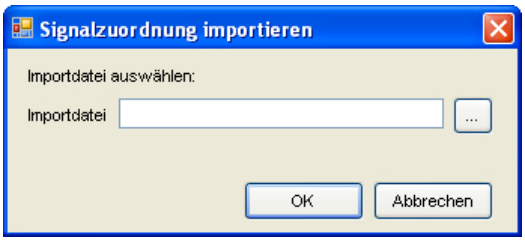
	A	B	C	D	E	F
1	**Adresse**	**Typ**	**InOut**	**Gruppe**	**Symbol**	**Kommentar**
2	FobF00M00InAna00	INT	Input	FobF00M00	FobF00M00InAna00	
3	FobF00M00InAna01	INT	Input	FobF00M00	FobF00M00InAna01	
4	FobF00M00InAna02	INT	Input	FobF00M00	FobF00M00InAna02	

4. Importieren Sie die Signalzuordnung indem Sie den Menüpunkt "Datei - Import - Signalzuordnung" verwenden..



Der Dialog „Signalzuordnung importieren" wird angezeigt.

5. Geben Sie den Dateinamen mit dem Zielpfad an.



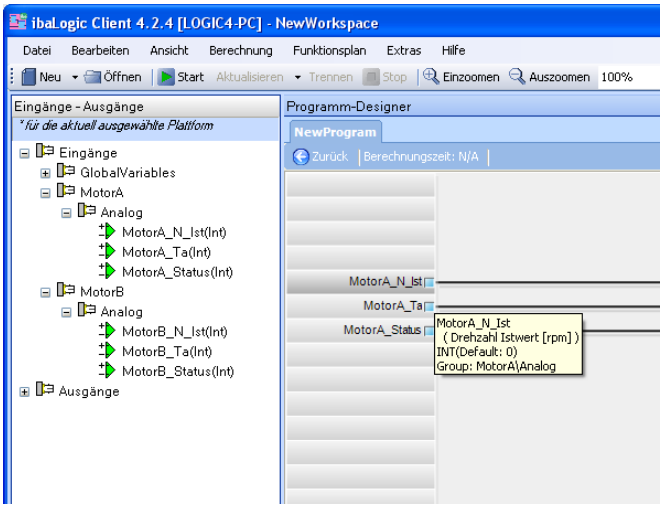
6. Klicken Sie auf <OK>, um den Import-Vorgang zu starten.

6.6.6 Signale im Programm verwenden

Damit Signale in einem Programm eines Projektes verwendet werden können, müssen diese auf die Eingangs- bzw. Ausgangsrandleiste gezogen werden.

Vorgehen

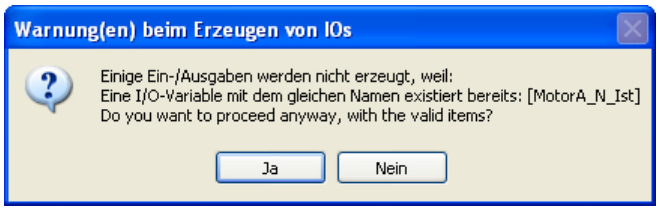
➲ Ziehen Sie das gewünschte Einzelsignal oder die ganze Gruppe auf die Randleiste des Ein- oder Ausgangs.



Beispiel

Das Signal „MotorA_N_Ist" wurde auf die Eingangsrandleiste gezogen.
Wenn mit der Maus auf das Konnektorsymbol des Signals gezeigt wird, dann wird ein Tooltip mit den Informationen zu diesem Signal angezeigt.

Wenn eine ganze Gruppe auf die Randleiste gezogen wird, dann werden alle Einzelsignale dort platziert. Befinden sich schon einzelne Signale aus der Gruppe in der Randleiste, wird folgende Warnung ausgegeben.



Hinweis

Weitere Informationen entnehmen Sie „I/O-Konfiguration , Seite 177".

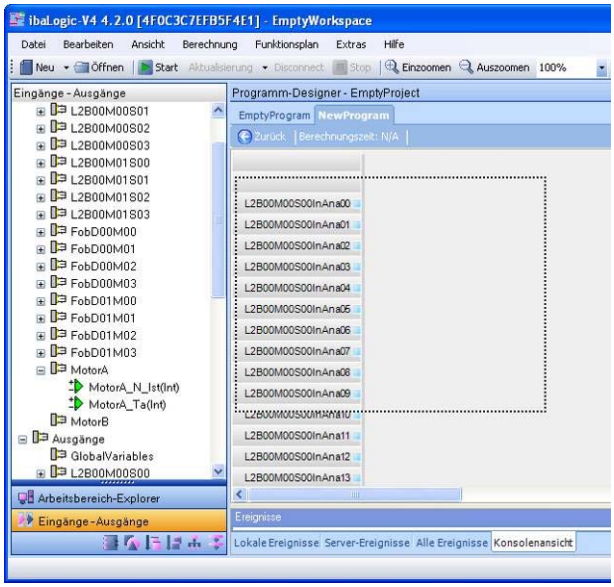
6.6.7 Signale im Programm entfernen

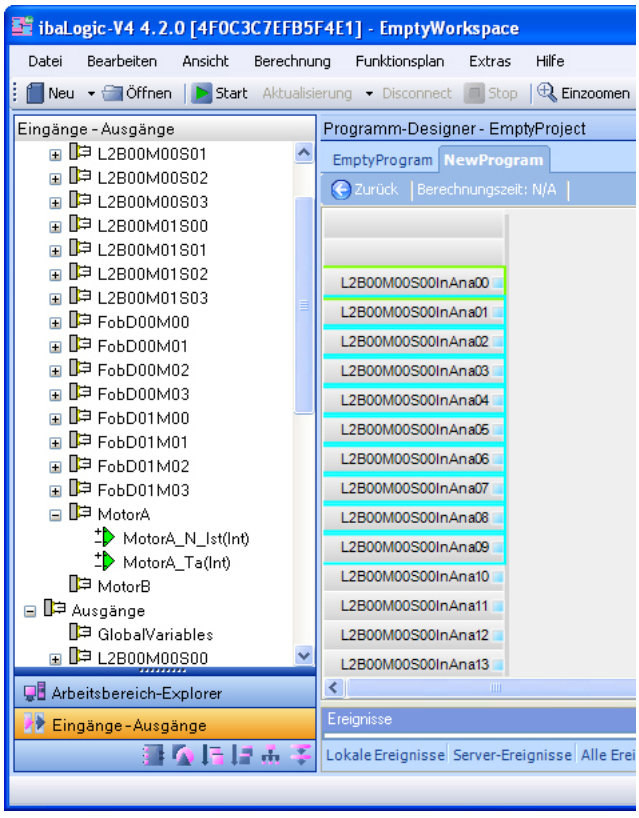
Vorgehen

1. Markieren Sie das Signal in der Ein- oder Ausgangsleiste.
2. Drücken Sie Funktionstaste <Entf>.

Anmerkung

Mehrfach-Selektion ist möglich mit linkem Mausklick und gleichzeitigem Drücken der <Shift>-Taste oder durch Aufziehen eines Rechtecks mit der linken Maustaste (Lasso) über den betreffenden Signalen. Beachten Sie, dass das Rechteck die Signale komplett umschließen muss.





7 Programmerstellung

7.1 Bausteine

In ibaLogic wird eine Vielzahl von Funktionsbausteinen in einer globalen Bibliothek zur Verfügung gestellt.

Zusätzlich zu den entsprechend der Norm IEC 61131-3 definierten Funktionsbausteinen, stellt iba eigene Funktionsbausteine und Anwenderbausteine zur Verfügung. Diese sind im Navigationsbereich der Ansicht „Funktionsbausteine" aufgelistet.

Jeder Funktionsbaustein enthält ein Symbol.
Die Symbole bedeuten:

- Bausteine der Norm IEC 61131-3
- iba-Bausteine
- Funktionen, die in Form einer DLL vorliegen
- Vom Anwender erstellte Funktionsbausteine und Makroblöcke

Die Funktionsbausteine sind in alphabetisch sortierten Gruppen und Untergruppen gemäß der Spezifikation IEC 61131-3 aufgelistet.

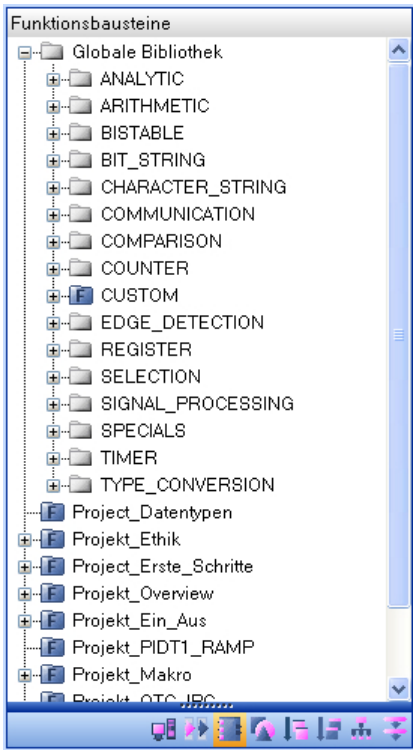


Abbildung 38: Funktionsbausteine „Globale Bibliothek"

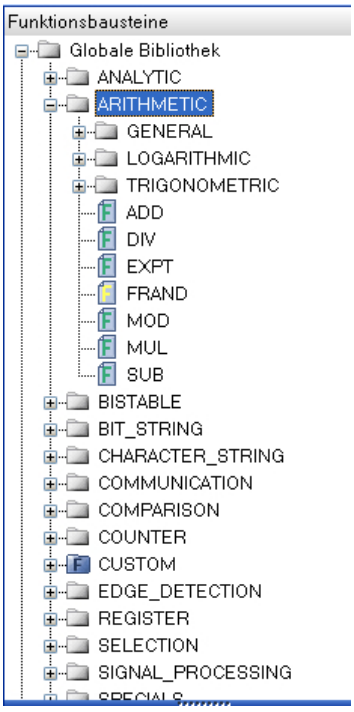


Abbildung 39: Funktionsbausteine „Arithmethic"

In der „Globalen Bibliothek" sind zusätzlich die Ordner „CUSTOM" und „NEW PROJECT" enthalten.

CUSTOM

Dort werden alle globalen Anwenderbausteine und Funktionen, die als DLL vorliegen, abgelegt.

SPECIALS

Dort sind die „Spezialitäten" von ibaLogic abgelegt. Weitere Informationen siehe "Specials , Seite 315".

7.1.1 Bausteine verwenden

In einem Projekt können Sie alle Bausteine aus der globalen Bibliothek und der Projekt-Bibliothek verwenden.

Wenn Sie einen FB aus der Globalen Bibliothek verwenden möchten, dann können Sie diesen per Drag & Drop in das Programmierfenster ziehen.

Auf Bausteine, die Sie in einem Projekt eines anderen Arbeitsbereiches definiert haben, haben Sie keinen Zugriff.

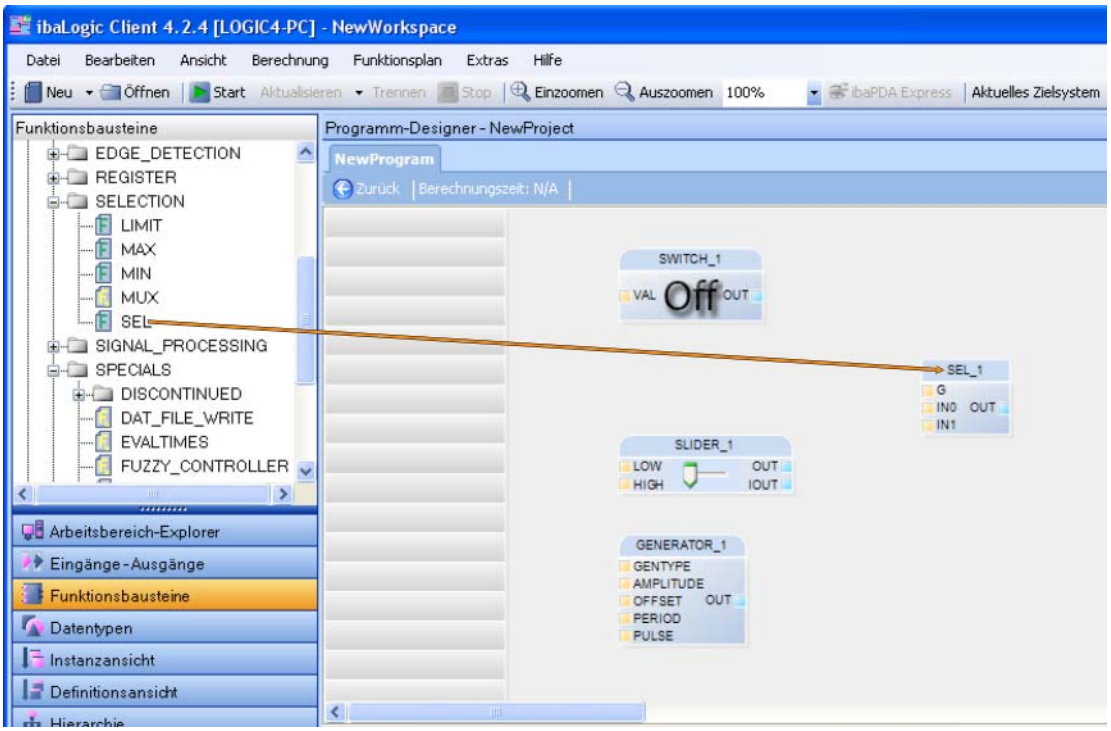
Wichtiger Hinweis

Wenn sie Inhalte von Bausteinen ändern, dann werden alle Instanzen und die Definition geändert.

Wenn Sie die Form, d. h. Ein- und Ausgänge oder interne Variablen, ändern wollen, werden Sie aufgefordert, einen anderen Definitionsnamen anzugeben. In diesem Fall wird ein neuer Bausteintyp angelegt, die alte Definition und deren Instanzen bleiben unverändert.
Allerdings nur, wenn mehr als eine Instanz vorhanden ist.

Vorgehen

1. Wählen Sie einen Baustein aus, den Sie im Programmierfeld platzieren möchten.
2. Ziehen Sie per Drag & Drop den ausgewählten Baustein in der „Funktionsbaustein-Übersicht" an eine noch freie Position im Programmierfeld.



7.1.2 Anwenderbausteine anlegen

Sie können einen Baustein auf unterschiedliche Arten anlegen:

- ❑ Im Programm
- ❑ Unter dem Projekt
- ❑ In der globalen Bibliothek

7.1.2.1 Im Programm

Erstellen eines von FBs oder MBs innerhalb eines Programms.

Vorgehen

1. Klicken Sie mit der rechten Maustaste an eine freie Stelle im Programmierfenster.
2. Wählen Sie im Kontextmenü „Neu… - Neuer Funktionsblock" oder „Neu… - Neuer Makroblock". Das Fenster „Funktionsblock erzeugen" wird angezeigt.
3. Erstellen Sie Ihren Baustein.
4. Mit <OK> wird bei fehlerfreier Syntax der Baustein angelegt.

7.1.2.2 Unter dem Projekt

Erstellen eines FBs innerhalb eines Projekts.

Vorgehen

1. Öffnen Sie die Ansicht „Funktionsbausteine".
2. Wechseln Sie in den Projekt-Ordner.
3. Klicken Sie im Funktionsbaum mit der rechten Maustaste auf das entsprechende Projekt.
4. Wählen Sie im Kontextmenü
   „Neu… - Neuer Funktionsblock" oder „Neu… - Neuer Makroblock".
   Das Fenster „Funktionsblock erzeugen" wird angezeigt.
5. Erstellen Sie Ihren Baustein.
6. Mit <OK> wird bei fehlerfreier Syntax der Baustein angelegt.

7.1.2.3 In der globalen Bibliothek

Erstellen eines FBs innerhalb der globalen Bibliothek.

Vorgehen

1. Klicken Sie mit der rechten Maustaste in der globalen Bibliothek auf die Gruppe „CUSTOM".
2. Wählen Sie im Kontextmenü „Neu… - Neuer Funktionsblock" oder „Neu… - Neuer Makroblock". Das Fenster „Funktionsblock erzeugen" wird angezeigt.
3. Erstellen Sie Ihren Baustein.
4. Mit <OK> wird bei fehlerfreier Syntax der Baustein angelegt.

Hinweis
Unterscheidung Definition - Instanz

Weitere Informationen siehe „Instanzansicht , Seite 56".

7.1.3 Bausteine verwalten

In die globale Bibliothek kopieren

Wollen Sie einen Baustein, den Sie in einem Programm oder Projekt definiert haben, auch in anderen Arbeitsbereichen verwenden, kopieren Sie diesen in die globale Bibliothek.

Vorgehen

1. Klicken Sie mit der rechten Maustaste den FB an, den Sie kopieren möchten.
2. Wählen Sie im Kontextmenü „In globale Bibliothek kopieren".
   Der Baustein wird in die Gruppe „CUSTOM" kopiert.

Hinweis

Ein Baustein, der in die globale Bibliothek kopiert wurde, enthält die Eigenschaften des Kopierzeitpunktes. Wird der FB im Projekt-Pfad geändert, so sind diese beiden Bausteine unterschiedlich.

Hinweis

Sie können FBs nicht aus der globalen Bibliothek in einen Projektkatalog kopieren oder verschieben. Beim Verwenden eines Bausteins aus der globalen Bibliothek wird dieser automatisch in den Projektkatalog kopiert.

Sie können auch Bausteine, die in einem anderen Projekt desselben Arbeitsbereiches definiert sind, verwenden. Dadurch werden diese auch automatisch unter den Projektbausteinen angelegt.

Weitere Informationen siehe „Bausteine verwenden , Seite 91".

7.1.4 Bausteine exportieren

Wollen Sie einen Baustein, den Sie in einer Datenbank unter der globalen Gruppe „CUSTOM" oder in einem Projekt definiert haben, auch in einer anderen Datenbank oder einem anderen Programmierwerkzeug verwenden, dann können Sie diesen in eine Textdatei exportieren.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf die Bausteindefinition unter „CUSTOM" oder unter dem Projekt.
2. Wählen Sie im Kontextmenü „Nach ST exportieren".
   Der Dialog „Export" wird angezeigt.
3. Geben Sie Zielverzeichnis und Dateinamen an.

Hinweis

Wenn Sie den exportierten Baustein in einer ibaLogic-V4-Umgebung verwenden wollen, dann müssen Sie den Baustein mit grafischen Zusatzinformationen exportieren.

Wollen Sie den Baustein in einem anderen System verwenden, dann sollten Sie diesen ohne grafische Zusatzinformationen exportieren.

Wichtiger Hinweis

Da die Implementierung der IEC-Norm herstellerabhängig ist, sollten Sie, bevor Sie den Baustein einbinden, das Fremdsystem dahingehend prüfen, ob diese Abweichungen von der IEC-Norm enthält.

7.1.5 Bausteine importieren

Voraussetzung

- ❑ ibaLogic läuft nicht im Online-Modus.
- ❑ Sie haben eine importierbare Datei (Baustein).

Vorgehen

1. Wählen Sie Menü „Datei – Import – Structured Text".
   Das Fenster „Structured text import" wird angezeigt.
2. Wählen Sie im Browser die zu importierende Datei aus.
3. Bestätigen Sie abschließend mit <OK>.

Auswahlfeld	Erklärung
Definition von Organisationseinheiten als neu importieren	Die Anwahl bewirkt, dass die importierte Organisationseinheit als ein neuer Baustein importiert wird.

7.1.6 Bausteine entfernen

Wenn Sie einen FB aus dem Programm entfernen, dann löschen Sie nur eine Instanz.

Vorgehen

1. Wählen Sie einen FB in der Bibliothek aus Ihrem Projekt aus.
2. Öffnen Sie das Kontextmenü.
3. Wählen Sie <Entfernen>.

Anmerkung

Wenn Sie die einzige vorhandene Instanz löschen, dann wird ein Dialog angezeigt. In dieser Dialogbox legen Sie fest, ob Sie auch die Definition löschen möchten. Bestätigen Sie abschließend mit <OK>. Die Definition wird gelöscht.

Der Funktions- bzw. Makroblock wird auch aus dem Projekt entfernt und ist damit nicht mehr verfügbar.

Eine globale Bausteindefinition in der Gruppe „CUSTOM" wird dadurch nicht gelöscht. Das können Sie mit einem rechten Mausklick auf den Baustein in der Gruppe „CUSTOM" im Kontextmenüpunkt „Entfernen" oder mit der Taste <Entf> vornehmen.

7.2 Standardbausteine

Im Anhang ist eine tabellarische Übersicht über alle Funktionen und Funktionsbausteine enthalten, die in ibaLogic zur Verfügung stehen.

7.3 Komplexe Funktionsbausteine

7.3.1 DAT_FILE_WRITE (DFW-Funktionsbaustein)

In ibaLogic ist die Funktion integriert, mit der Sie vorhandene Analog- und Digitalsignale zur Laufzeit in Messdateien speichern können.
Die Messdateien haben das iba-Datenformat und können deshalb mit den iba-Werkzeugen ibaDatCoordinator und ibaAnalyzer geöffnet, analysiert und weiterverarbeitet werden (z. B. Extraktion in eine Datenbank).

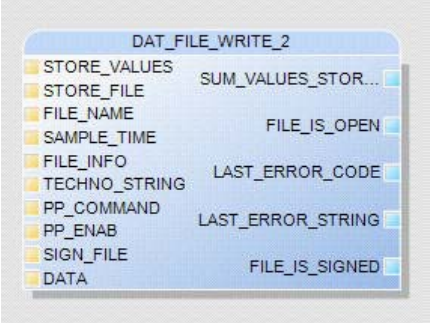


Abbildung 40: DAT_FILE_WRITE-Baustein

Sie können Einzelwerte oder Arrays von gepufferten Daten speichern. Daten vom Typ INTEGER, REAL und BOOL sind erlaubt. Die Speicherung zusätzlicher Informationen, wie Technostring, wird unterstützt. Weitere Informationen siehe „Buffered Mode , Seite 195".

Hinweis

Das Schreiben von Messdateien ist lizenzpflichtig. Ohne gültigen Dongle bleibt FILE_IS_SIGNED auf „FALSE" bei der Datenaufzeichnung und damit können die erzeugten Dateien nicht mit ibaAnalyzer ausgewertet werden.

Hinweis

Im Anhang befindet sich ein Beispiel zur Konfiguration.

7.3.1.1 Funktionsbaustein DFW bearbeiten

Nachdem Sie den DFW-Baustein aus dem Bausteinordner in das Programmfenster gezogen haben, können Sie diesen mit einem Doppelklick öffnen.

Das Fenster ist in folgende Register eingeteilt:

- „Argumente"
- „Grafisch"

Das Register „Grafisch" enthält die Unterregister:

- „Allgemeine Konfiguration"
- „Signalkonfiguration"

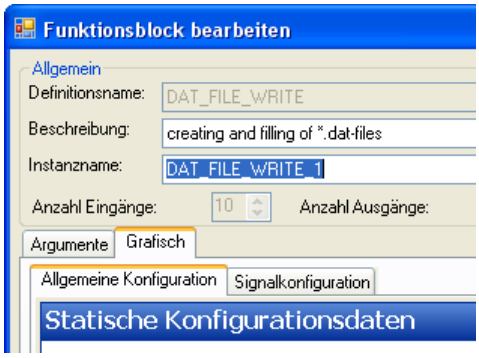


Abbildung 41: DAT_FILE_WRITE-Konfigurator

Register „Argumente"

Das Register „Argumente" stellt alle Eingänge, Ausgänge und die dazugehörigen Variablen und Datentypen in einer tabellarischen Ansicht dar. Auch diese Ansicht dient als Übersicht und im Online-Modus zur Anzeige der aktuellen Werte. Nehmen Sie die Einstellungen nicht in dieser Ansicht vor, sondern wechseln Sie in das Register „Grafisch". Auf Ausnahmen wird bei den einzelnen Eigenschaften hingewiesen.

Wichtiger Hinweis

Wenn Sie einen Eingangskonnektor verbinden, dann werden die standardmäßigen und im Funktionsblock eingestellten Werte überschrieben. Nach Beenden der Verbindung wird der letzte Wert beibehalten.

7.3.1.2 Unterregister „Allgemeine Konfiguration"

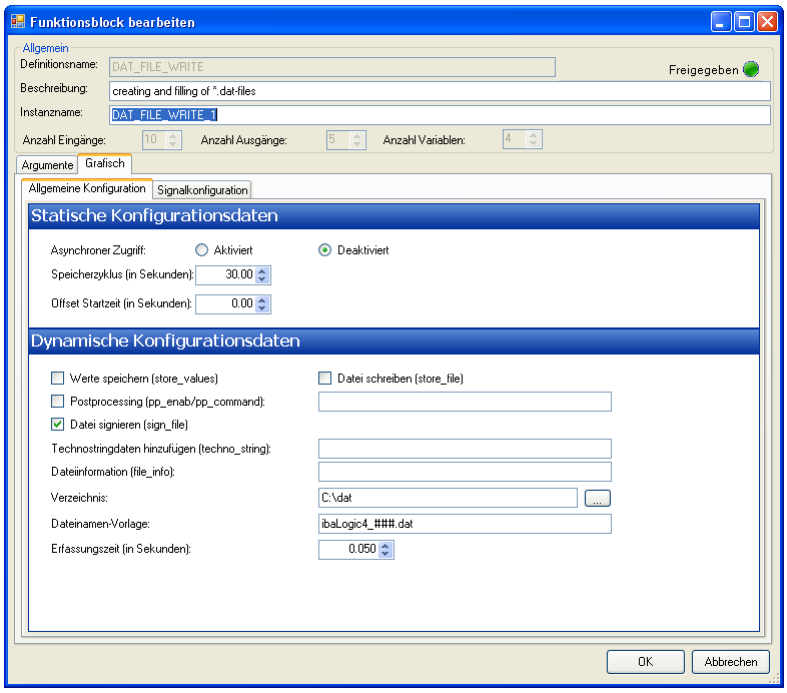


Abbildung 42: Unterregister „Allgemeine Konfiguration"

❑ Asynchroner Zugriff

Deaktiviert:
Die zu speichernden Daten werden intern gepuffert. Wenn der Puffer gefüllt ist, dann wird auf die Festplatte geschrieben. Ist die Aufzeichnung der Datei abgeschlossen, ist eine nachfolgende Analyse der Messdatei (mit ibaAnalyzer) möglich.

Aktiviert:
Der interne Puffer wird zyklisch nach der eingestellten Speicherzykluszeit auf die Festplatte geschrieben. Deshalb können Sie bereits mit der Analyse beginnen, während die Aufzeichnung noch läuft. Je kleiner der Speicherzyklus ist, desto früher stehen die Daten im ibaAnalyzer zur Verfügung (wenn Sie dort „automatisches Nachladen" eingestellt haben). Die Folge ist ein geringerer Komprimierungsgrad und eine höhere Auslastung des Rechners bzw. eines Netzwerkes.

❑ Offset Startzeit

Diese Zeit verschiebt die Zeitachse in den Messdateien in der Zeit rückwärts. Die Zeitachse in der Messdatei wird gebildet durch den Startzeitpunkt und die Samplezeit, d. h. durch die Anzahl der Messpunkte und die Zeit zwischen den einzelnen Messpunkten. Die Startzeit im Dat-File ist damit „normale Startzeit"-Offset Startzeit.

In Verbindung mit der Verzögerung der zu speichernden Daten (mit dem DELAY-Baustein) kann ein Trigger-Vorlauf (Pretrigger) erzeugt werden.

❑ Wenn der Trigger auslöst, dann wird die Aufzeichnung gestartet. Um in der Aufzeichnung die Verzögerung unwirksam zu machen, muss die Offset-Startzeit auf die Verzögerungszeit gesetzt werden.

❑ Datei schreiben (store_file)

Tipp

Aktivieren Sie dieses Feld nicht in diesem Register, sondern von außen über den Konnektor STORE_FILE.

Positive Flanke: Die Messdatei mit dem Namen und Pfad aus den Einstellungen wird angelegt. Die Aufzeichnung wird nur gestartet wenn STORE_VALUES = TRUE.

Negative Flanke: Die Messdatei wird geschlossen.

❑ Werte speichern (store_values)

Wenn dieser Wert „TRUE" ist und die Datei geöffnet ist, dann wird in jedem Berechnungszyklus ein Messdaten-Sample gespeichert.

Die Behandlung dieses Parameters ist abhängig von dem Modus der Daten:

Bei ungepufferten Werten aktivieren Sie dieses Feld und lassen den Konnektor STORE_VALUES unverbunden. Die Aufzeichnung steuern Sie mit STORE_FILE.

Hinweis

Wenn Sie die Aufzeichnung der ungepufferten Daten mit STORE_VALUES steuern, bekommen Sie eine verfälschte Zeitachse. Da diese im ibaAnalyzer durch Anzahl und Zeit der Samples gesteuert wird, entstehen durch dynamisches Aus- und Einschalten von STORE_VALUES keine Lücken auf der Zeitachse, sondern die Samples werden einfach aneinandergereiht und die Lücke entsteht am Ende der Messdatei.

Bei gepufferten Werten muss diese Variable anders behandelt werden. Weitere Informationen siehe „Unterregister „Signalkonfiguration" , Seite 103".

❑ Postprocessing (pp_enab/pp_command)

Postprocessing steht kompatibel zu ibaLogic-V3 zur Verfügung.
iba empfiehlt, die Nachbearbeitung durch den ibaDatCoordinator zu realisieren. Dieser stellt umfangreiche Funktionen zu Weiterbearbeitung zur Verfügung, z. B. Kopieren der Dateien auf einen Fileserver, Übergabe an den ibaAnalyzer zur Extraktion in Datenbank etc.

❑ Datei signieren (sign_file)

Durch das Auswählen „Datei signieren" wird die Datei signiert, um erweiterte Funktionen im ibaAnalyzer nutzen zu können. Lassen Sie diese Option immer gesetzt. Erst mit Erzeugung des Ausgangsfiles wird auch der zugehörige Ausgang gesetzt. Wird kein entsprechender Dongle gefunden, so ist dieser FALSE.

❑ Technostring-Daten hinzufügen (techno_string)

Ein Technostring enthält begleitende Daten zu einer Messung, z. B. Chargennummer, Materialkennungen u. a. Diese können im ibaAnalyzer gesondert ausgewertet werden.

Der Technostring wird beim Schließen der Datei gespeichert.

❑ Wenn Sie den Technostring nutzen wollen, dann verbinden Sie den Konnektor TECHNO_STRING mit einer Variablen von Typ String z. B. die Empfangsdaten eines TCP/IP-Bausteins.

❑ Dateiinformationen (file_info)

Das sind zusätzliche ASCII-Informationen, die beim Schließen der Datei gespeichert werden. Diese Informationen sind bei der Analyse unter „Info" zu finden.
iba empfiehlt, den Eingangskonnektor FILE_INFO zu verwenden, um dynamisch den Inhalt verändern zu können.

Die Info-Felder im Analyzer sind wie folgt aufgebaut:

Feldname: Text

Wird kein „:" im Text gefunden, dann setzt ibaLogic „UserField0" als Default-Wert ein.
Mehrere Info-Felder werden mit „;" getrennt.

Hinweis

Standard-Info-Feldnamen können nicht überschrieben werden. Einträge mit diesem Namen werden ignoriert.

Standard-Info-Feldnamen:

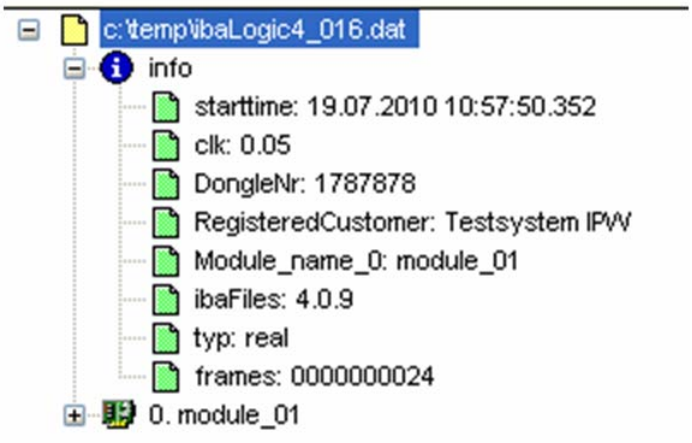


Abbildung 43: Standard-Info-Feldnamen

❑ Verzeichnis (Teil von file_name):

➲ Wählen Sie durch Klick auf den Browser-Button <...> das Laufwerk und den Pfad aus, auf dem die Messdateien gespeichert werden sollen.

Wichtiger Hinweis

Wenn sich das Laufzeitsystem auf dem Zielsystem PADU-S-IT befindet, dann können Sie nicht mit dem Windows-Filebrowser arbeiten. Geben Sie den Pfad und Dateinamen in der Vorbesetzung zum Konnektor FILE_NAME im Register „Argumente" an oder verbinden Sie den Konnektor mit einer String-Variablen, in dem Sie Pfad und Dateinamen dynamisch einstellen können. Der Eingang wird beim Öffnen der Datei (STORE_FILE) übernommen.

Als Verzeichnis sollte das voreingestellte „C:\dat“ verwendet werden. Die Daten können dann mit dem ibaDatCoordinator abgeholt werden. Im ibaDatCoordinator kann dann beispielsweise mit „\\S-IT-16-000074\RamDisk\dat“ zugegriffen werden. Dabei ist „S-IT-16-000074“ entweder der Hostname oder direkt die IP-Adresse des angesprochenen PADU-S-ITs, „RamDisk“ der interne Freigabenamen und „dat“ der angegebene Verzeichnisname.

❑ Dateinamen-Vorlage (Teil von file_name):

Vorgabe des Dateinamen und des Index. Bei jeder neuen Messdatei wird, falls sich der vorgegebene Pfad und Dateiname nicht ändert, der Index inkrementiert.

Als Platzhalter für den Index wird das Zeichen "#" verwendet. Mehrstellige Indizes geben Sie mit mehrfachen Platzhaltern an.

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

Wird der Pfad nicht verändert, dann werden nach Überlauf des Index die ältesten Messdateien überschrieben.

❑ Erfassungszeit (in Sekunden) (sample_time)

Dieser Wert ist die Zeitbasis der Messdatei. Diese beschreibt die Zeit zwischen zwei gespeicherten Werten eines Messsignals in Sekunden.

Bei ungepufferten Werten geben Sie hier den Taskintervall an, mit dem das Programm läuft, das den DFW-Baustein enthält.

Bei gepufferten Werten müssen Sie hier die Zeitdauer, die der Tiefe des Daten-Arrays entspricht, eingeben.
Weitere Informationen siehe "Unterregister „Signalkonfiguration" , Seite 103"".

Wichtiger Hinweis

Die Aufbereitung der Messwerte und der DFW-Baustein müssen im selben Task-intervall laufen. Wenn das nicht der Fall ist, dann wird die Zeitachse in der Messdatei gestreckt oder gestaucht.

Beispiel zur Erfassungszeit

Läuft der Task, in dem Sie die Messdaten bereitstellen, im 10-ms-Intervall, dann müssen Sie im DFW-Baustein die Erfassungszeit 0,01 s einstellen.

Wollen Sie, dass die Werte nur alle 50 ms gespeichert werden, dann genügt es nicht, nur die Erfassungszeit auf 0,05 s zu stellen (denn dann werden dieselben Daten gespeichert, aber die x-Achse spielt Ihnen einen 5-fach längeren Zeitraum vor), sondern Sie müssen an dem Konnektor STORE_VALUES einen Takt anlegen, der bei jedem 5. Zyklus auf „TRUE" geht.

Einfacher ist es aber, den Baustein in einem 50-ms-Task laufen zu lassen und den STORE_VALUES-Konnektor statisch auf „TRUE" stehen zu lassen.

Tipp

Sehen Sie in der Analyse trotz richtig eingestellter Parameter eine zu lange oder zu kurze Zeitachse, dann überprüfen Sie, ob die echte Taskintervallzeit (Baustein EVALTIMES) mit der projektierten übereinstimmt. Bei kleinen Zykluszeiten wird empfohlen, den Turbo-Modus zu aktivieren, um den Taskintervall konstant zu halten.

7.3.1.3 Unterregister „Signalkonfiguration"

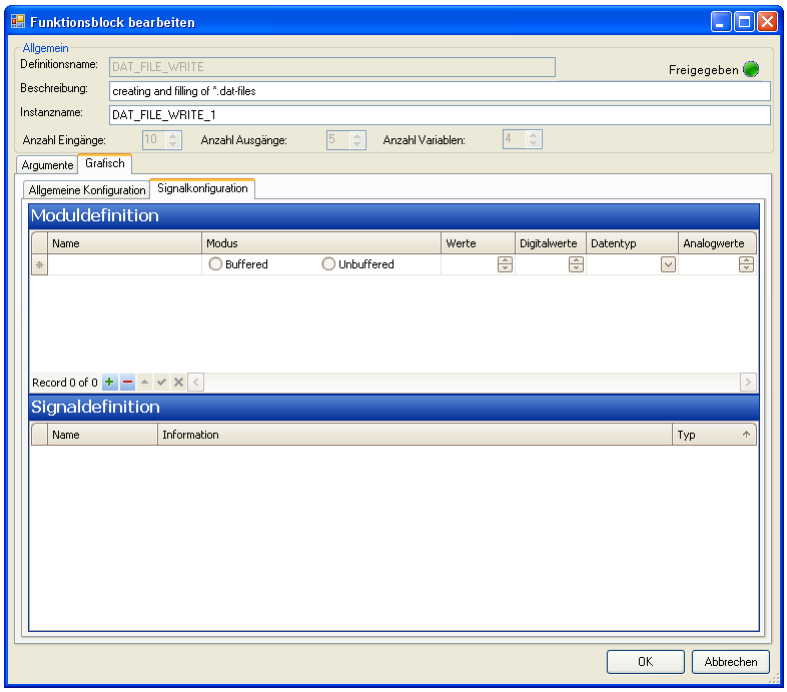


Abbildung 44: Unterregister „Signalkonfiguration"

Das Unterregister „Signalkonfiguration" enthält die Bereiche:

- Moduldefinition:
  legt die Modulnamen, Modultyp, die Anzahl der Signale und deren Datentyp fest
- Signaldefinition:
  legt die Signalnamen und die Signalbeschreibung fest

Wichtiger Hinweis

In der derzeitigen Version von ibaLogic sollten die kompletten Konfigurationen der Module und Signale im DFW-Baustein vor einem Anschluss eines Messsignals durchgeführt werden.

Grund: ibaLogic erstellt beim Kompilieren interne Struktur- und Arraydatentypen, die, wenn Sie einmal verwendet werden, nicht mehr geändert werden können. Ändern Sie nachträglich die Signalkonfiguration, müssen Sie entweder alle automatisch eingefügten Joiner entfernen und neu verdrahten oder alle entsprechenden Datentypen in den ST-Bausteinen anpassen.

Um ein neues Modul zu erstellen, wird Ihnen immer am Ende der Modulliste ein vorbereiteter Eintrag zur Verfügung gestellt, der ein leeres Namensfeld enthält. Tragen Sie hier einfach den Modulnamen ein und parametrieren Sie dieses Modul. Beim Verlassen wird automatisch wieder ein neuer vorbereiteter Eintrag am Ende der Liste erzeugt.

Prinzipiell können beliebig viele Module angelegt werden. Die maximale Anzahl ist des Weiteren abhängig von Ihrer Lizenz.

Beschreibung der Module:

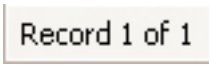
- Name:
  Modulname, muss der IEC-Norm entsprechen.
- Modus Unbuffered:
  Aufzeichnung Einzelwerte. Bei jedem Zyklus wird ein Datensample gespeichert. Der Wert in der Spalte „Werte" wird nicht berücksichtigt.
- Modus Buffered:
  Aufzeichnung Pakete. Bei jedem Zyklus wird ein Array von Datensamples gespeichert. Der Wert in der Spalte „Werte" gibt die Arraytiefe, d. h. die Anzahl der Samples, an.
  Weitere Informationen siehe „Buffered Mode , Seite 195".
- Werte:
  Ohne Bedeutung im Modus „Unbuffered".
  Anzahl der gespeicherten Samples pro Speicherzyklus im Modus „Buffered".
- Digitalwerte:
  Anzahl der Binärsignale in diesem Modul (max. 32)
- Datentyp der Analogwerte, zulässig sind REAL und INTEGER
- Analogwerte:
  Anzahl der Analogwerte in diesem Modul (max. 32)

Unter der Signaldefinition werden alle Signale des markierten Moduls angezeigt. Die Signale werden mit den Default-Namen (Digital_nn und Analog_nn) angelegt. Sie können die Signalnamen editieren und jedem Signal unter „Information" eine Beschreibung anfügen.

Hinweis

Die Signalkonfiguration können Sie nicht verändern, solange der Konnektor „DATA" mit Daten verbunden ist. Wollen Sie Änderungen durchführen, dann müssen Sie die Verbindung trennen. Dabei werden automatisch generierte Joiner entfernt.

Symbolleiste zur Bearbeitung des Moduldefinitionsdatensatzes:

Symbol	Name/Tooltip	Erklärung
	Append	Fügt einen neuen leeren Moduldefinitionsdatensatz hinzu.
	Delete	Entfernt den selektierten Moduldefinitionsdatensatz.
	Edit	Gibt den selektierten Moduldefinitionsdatensatz zur Bearbeitung frei.
	End Edit	Die eingestellten Daten des Moduldefinitionsdatensatzes werden übernommen.
	Cancel Edit	Die eingestellten Daten des Moduldefinitionsdatensatzes werden nicht übernommen. Die Eingabe wird abgebrochen.
	-	Anzahl der Datensätze

Weitere Informationen siehe Übungsbeispiele , Seite 257 im Anhang.

7.3.1.4 Ablagestruktur generieren

Der einfachste Fall, die erzeugten Messdateien abzulegen, ist ein festes Verzeichnis und einen festen Dateien-Basisnamen anzugeben, der dann von ibaLogic automatisch eine laufende Nummer zugewiesen bekommt.

Wird eine Ablage-Struktur mit Unterverzeichnissen und Dateinamen benötigt, die z. B. die aktuelle Chargennummer enthalten soll, dann muss diese programmiert werden.

Beispiel

Stündlich soll eine neue Datei angelegt werden. Der Dateiname soll den Stundenwert im Namen enthalten.

Realisierung

Der Dateiname wird aus der aktuellen Stunde gebildet. Mit den Bausteinen DELAY und EQ wird bei der Änderung des Wertes (Stundenumschaltung) ein Low-Impuls für den STORE_FILE Eingang des DFW. Damit wird beim Stundenwechsel der anstehende Name für die nächste Messdatei übernommen.

Der Dateiname wird mit CONCAT-Bausteinen gebildet. Am ersten CONCAT steht der Pfad- und Grund-Dateiname. Dem wird die Stunde hinzugefügt und mit dem 2. CONCAT noch die Erweiterung .dat angehängt. Damit wird hier als Datei-Name dann c:\iba11.dat erzeugt.

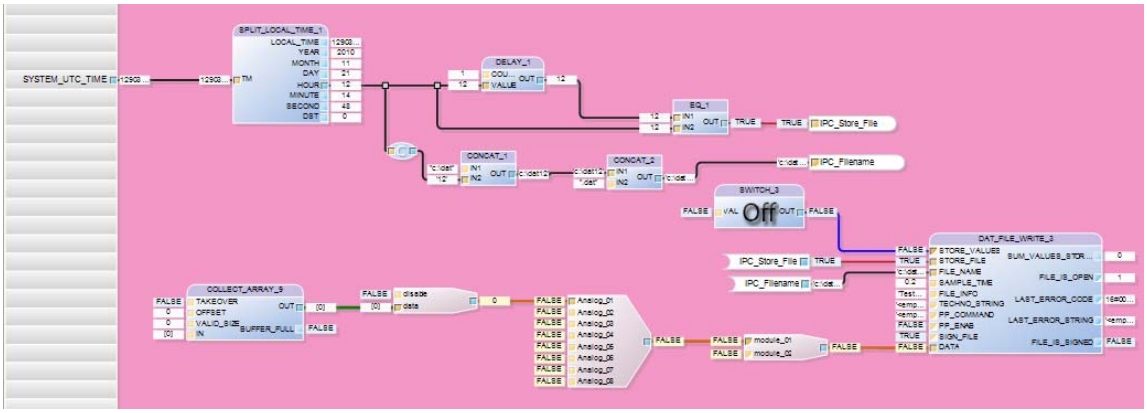


Abbildung 45: Beispiel: CONCAT-Bausteine

Genauso kann aber auch ein individueller Messdateiname samt Pfad-Angaben zusammengestellt werden. Wenn Sie einen neuen Pfad angeben, dann wird dieser von ibaLogic automatisch angelegt.

7.3.2 TCPIP_SENDRECV

Dieser Baustein ermöglicht das Senden und Empfangen von Daten via TCP/IP. Hierbei handelt es sich um Rohdaten, die über TCP/IP versendet werden. Somit können alle nativen TCP/IP-Protokolle nachgebaut werden.

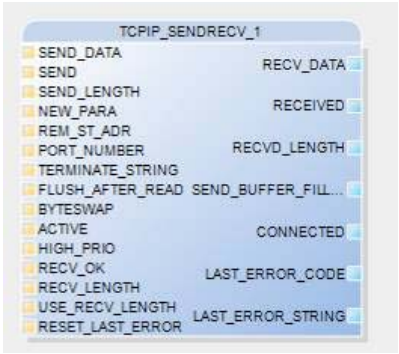


Abbildung 46: TCPIP_SENDRECV-Funktionsbaustein

Hinweis

Dieser Baustein ist lizenzpflichtig.

Wichtiger Hinweis

Unter Windows XP ist eine maximale Anzahl von erlaubten parallelen Verbindungen einstellbar. Der Wert hängt von der Windows-Konfiguration ab. Durch einen Eintrag in der Windows-Registrierungsdatenbank kann eine Anpassung vorgenommen werden.

Weitere Informationen siehe „Anzahl möglicher TCP/IP-Verbindungen , Seite 334".

7.3.2.1 Eingänge

Konnektor	Datentyp	Erklärung
SEND_DATA	Any	Zu sendende Daten. Der Datentyp ist ANY, d. h. er richtet sich nach dem angeschlossenen Datentyp z. B. eine Struktur, String, Array.
SEND	Bool	Bei „TRUE" werden die Daten, die an SEND_DATA anliegen, gesendet. Dieser Eingang ist nicht flankenorientiert, d. h. ein festes „TRUE" am Eingang bewirkt ein Senden in jedem Takt.
SEND_LENGTH	Udint	Länge der zu sendenden Daten in Bytes. Ist der Wert 0, werden alle anliegenden Daten gesendet
NEW_PARA	Bool	Übernehmen neuer Verbindungsparameter.
REM_ST_ADR	String	IP des Verbindungspartner. Diese IP wird nur benötigt für den aktiven Verbindungsaufbau, d. h. wenn der Eingang ACTIVE = TRUE ist. Ist ACTIVE = FALSE wartet der Baustein auf den Partner, dieser muss die IP-Adresse des PCs bzw. des PADU-S-IT angeben. Statt der IP-Adresse kann auch der Computername eingegeben werden. Diese Namensauflösung kann beim Anlegen je nach Betriebssystem-Konfiguration etwas Zeit benötigen.
PORT_NUMBER	Udint	Bei ACTIVE = TRUE: Portnummer des Partners (Nummer Gegenstelle). Bei ACTIVE = FALSE: Eigene Portnummer
TERMINATE_STRING	Udint	Strings werden mit einem NULL-Byte abgeschlossen. Dieser Eingang wird nur ausgewertet, wenn Strings am SEND_DATA Eingang angeschlossen sind.
FLUSH_AFTER_READ	Bool	Löscht den Empfangspuffer nach dem Lesen der Daten.
BYTESWAP	Int	= 1: Swappen nach Datentyp (AB CDEF → BA FEDC) = 2: jeweils 2 Bytes tauschen (ABCD → BADC) = 4: jeweils 4 Bytes tauschen (ABCD → DCBA)
ACTIVE	Bool	TRUE: (ibaLogic ist der TCP/IP-Client) Der Baustein versucht eine Verbindung zu der IP und dem PORT aufzubauen die in REM_ST_ADR und PORT_NUMBER angegeben sind. FALSE: (ibaLogic ist der TCP/IP-Server) Es wird an der eingestellten Portnummer auf eingehende Verbindungen gewartet. Die IP-Adresse wird nicht ausgewertet.
HIGH_PRIO	Bool	Die Daten werden mit einer höheren Priorität vom Windows-Netzwerkpuffer abgeholt und in den Eingangspuffer geschrieben. **HINWEIS** Standardmäßig sollte diese Funktion auf „FALSE" stehen.
RECV_OK	Bool	TRUE: Empfangene Daten sind OK, der Eingangspuffer kann mit neuen Daten gefüllt werden. FALSE: Die letzten empfangenen Daten bleiben bis zum erneuten Triggern des Eingangs erhalten.
RECV_LENGTH	Udint	Legt die Telegrammläge in Bytes fest. Dieser Eingang wird nur ausgewertet, wenn der Eingang USE_RECV_LENGTH =TRUE ist
USE_RECV_LENGTH	Bool	TRUE: Ist der Eingangspuffer größer als die eingestellte RECV_LENGTH steht am Ausgang RECV_DATA nur ein Telegramm mit der eingestellten Länge an. FALSE: Am Ausgang RECV_DATA steht ein Telegramm mit der maximalen Größe des an den Ausgang RECV_DATA angeschlossenen Datentyps an.
RESET_LAST_ERROR	Bool	Setzt die Error-Ausgänge zurück

7.3.2.2 Ausgänge

Konnektor	Datentyp	Erklärung
RECV_DATA	Any	Empfangsdaten. Datentyp richtet sich nach angeschlossenem Datentyp z. B. eine Struktur, String, Array.
RECEIVED	Bool	Sobald Daten empfangen wurden, wird dieser Ausgang TRUE
RECVD_LENGTH	Udint	Länge der empfangenen Daten in Bytes
SEND_BUFFER_FILLED	Bool	TRUE wenn der interne Sendepuffer voll ist. D. h. der Baustein kann die Daten nicht schnell genug versenden
LAST_ERROR_CODE	Dword	Letzte Fehlermeldung als DWORD im Hex-Format
LAST_ERROR_STRING	String	Letzte Fehlermeldung als Klartext

Beispiel für einen Sende-Empfangs-Vorgang

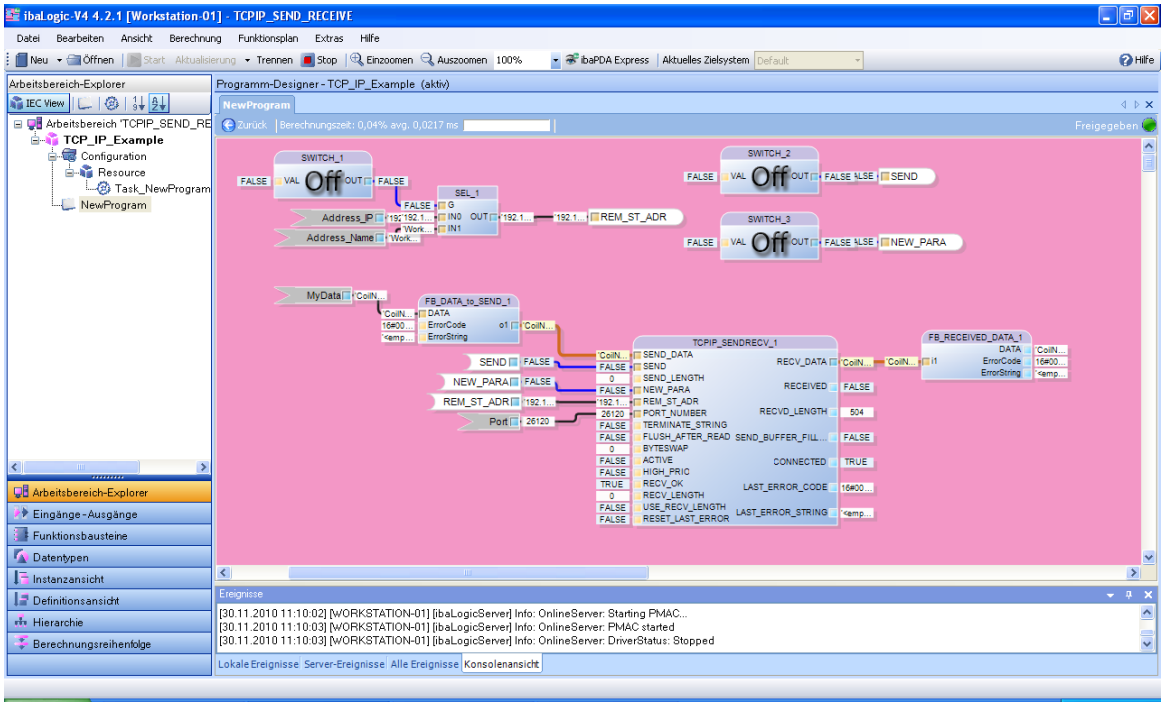


Abbildung 47: Beispiel für einen Sende-Empfangs-Vorgang

In diesem Beispiel liegen die Sendedaten in Form einer Struktur an. Die Daten werden hier durch einen manuellen Trigger gesendet.

Am NEW_PARA ist ein Switch, um bei Bedarf für Versuchszwecke oder bei Änderung von Verbindungsparameter die Verbindung neu anzustoßen.

Dieser Baustein ist passiv und und wartet, bis eine Verbindung durch den Kommunikationspartner aufgebaut wird.

Die empfangenen Daten kommen an RECV_DATA an. Der Eingang RECV_OK ist fest auf „TRUE" gesetzt, da die Daten nicht zwischengepuffert werden müssen, weil diese in der eingestellten Task-Takt-Zeit gut weiterverarbeitet werden können.

7.3.3 PIDT1_CONTROL

Universeller PIDT1-Regler, umschaltbar auf die Betriebsarten P-, I-, PI- und PIDT1-Regler.

- ❑ Anfangswert des Integrators setzen
- ❑ Momentanwert des Integrators festhalten
- ❑ Vorsteuerwert WP
- ❑ Reglerbegrenzung LL und LU
- ❑ Proportionalbeiwert KP
- ❑ Nachstellzeit TN
- ❑ Regelsinn umkehrbar
- ❑ Anzeige bei Erreichen der eingestellten Grenzen
- ❑ Anzeige der Regeldifferenz
- ❑ Anzeige der einzelnen P-, I-, und DT1-Reglerausgänge

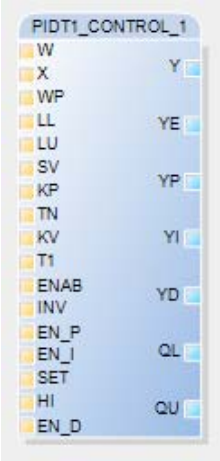


Abbildung 48: PIDT1_CONTROL-Funktionsbaustein

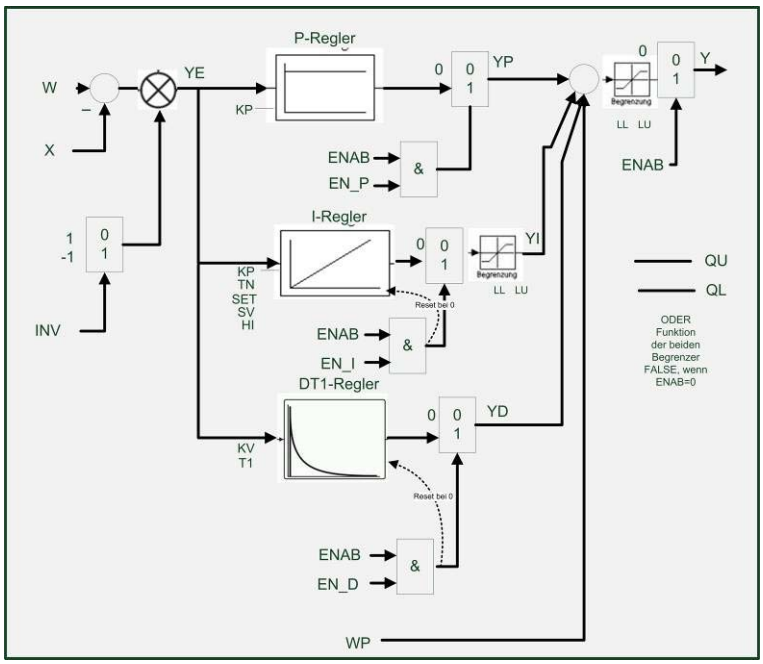


Abbildung 49: Blockschaltbild für einen universellen PIDT1-Regler

7.3.3.1 Eingänge

Konnektor	Datentyp	Erklärung
W	Lreal	Sollwert
X	Lreal	Istwert
WP	Lreal	Vorsteuerwert
LL	Lreal	Unterer Grenzwert (gilt für Y und YI)
LU	Lreal	Oberer Grenzwert (gilt für Y und YI)
SV	Lreal	Setzwert für Integrator wird mit SET übernommen
KP	Lreal	P-Verstärkung
TN	Time	Rücksetzzeit
KV	Lreal	D-Verstärkung
T1	Time	D-Zeitkonstante
ENAB	Bool	Reglerfreigabe
INV	Bool	Vorzeichenumkehr Regelabweichung aktivieren
EN_P	Bool	P-Regler aktivieren
EN_I	Bool	I-Regler aktivieren
SET	Bool	Integrator setzen mit Wert SV
HI	Bool	Integrator anhalten
EN_D	Bool	D-Regler aktivieren

7.3.3.2 Ausgänge

Konnektor	Datentyp	Erklärung
Y	Lreal	Stellwert = YP+YI+YD+WP
YE	Lreal	Regelabweichung = W-X
YP	Lreal	Ausgangswert P-Regler = KP*YE
YI	Lreal	Ausgangswert I-Regler = YI_{n-1}+KP*YE*Ta/TN
YD	Lreal	Ausgangswert D-Regler = α*YD_{n-1}+ α*KV*ΔYE α = 1/ (1+Ta/T1) ΔYE = (YE-YE_{n-1})
QL	Bool	unterer Grenzwert erreicht
QU	Bool	oberer Grenzwert erreicht

7.3.3.3 Details/Signalverläufe

In den folgenden Diagrammen werden verschiedene Signalverläufe der einzelnen Regleranteile dargestellt.

Stellwert Y:

Der Stellwert Y ist die Summe aus P-, I-, D-Anteil und dem Vorsteuerwert WP.

Ist der Eingang ENAB (Reglerfreigabe) nicht gesetzt, dann ist der Stellwert immer 0.0.

Eingang WP Vorsteuerwert:

Dieser Eingang wird auf den Ausgang Y addiert.

Eingang LL/LU Unterer/oberer Grenzwert:

Hinweis

Der Gesamtausgang Y, sowie YI werden begrenzt.

Die Ausgänge QL und QU gehen entsprechend auf TRUE.

In der Praxis kommen folgende Regler zum Einsatz:

PI-Regler

PD-Regler

PID-Regler

Im Folgenden werden der P-Anteil, der I-Anteil und der DT1-Anteil gesondert betrachtet.

7.3.3.4 P-Anteil: (Parameter: KP, EN_P)

Der P-Regelanteil berechnet sich aus KP*YE. Der Wert wird nur auf den Ausgangswert geschaltet, wenn EN_P gesetzt ist.

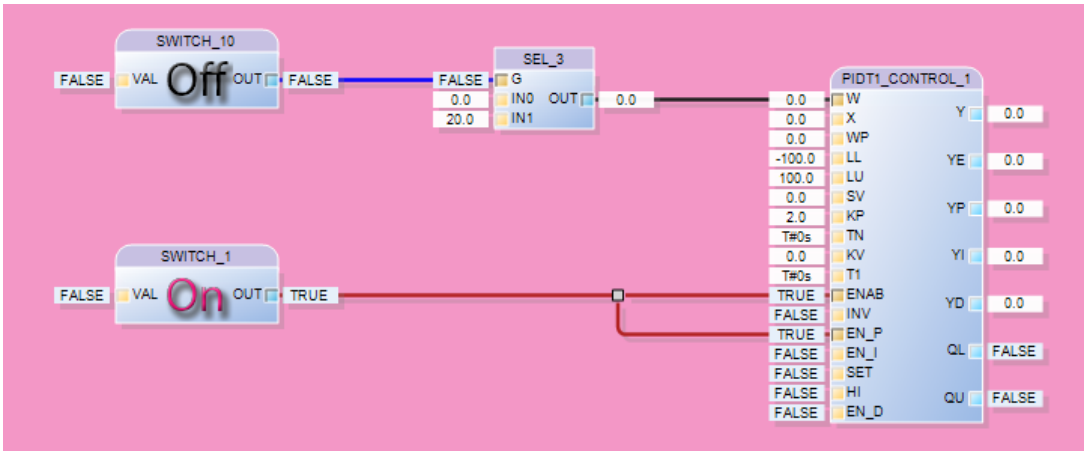


Abbildung 50: P-Anteil

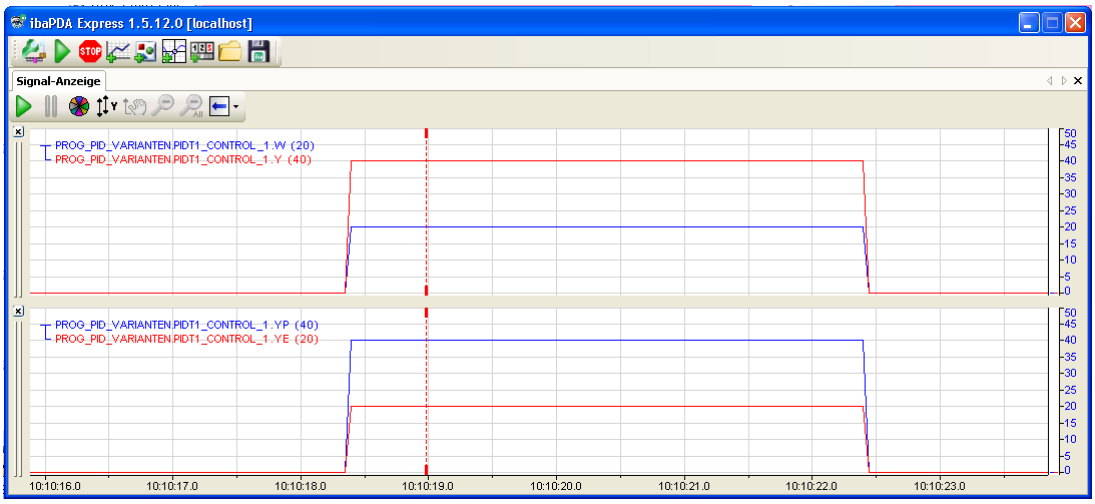


Abbildung 51: P-Anteil

7.3.3.5 I-Anteil: (Parameter KP, TN, SET, SV, HI, EN_I)

Der I-Regelanteil berechnet sich aus $YI_n := YI_{n-1} + KP * YE * Ta/TN$ (Ta = Taskzeit). Dieser Anteil kann mit dem Eingang SET auf den Wert vom Eingang SV gesetzt werden. Ein „TRUE" am HI-Eingang stoppt den Integrator. Der Wert wird nur auf den Ausgangswert geschaltet, wenn EN_I gesetzt ist.

Zusammenhänge

TN = KP*Ta = KP/KI

Beispiel 1

KP = 1.0

TN = 1 s

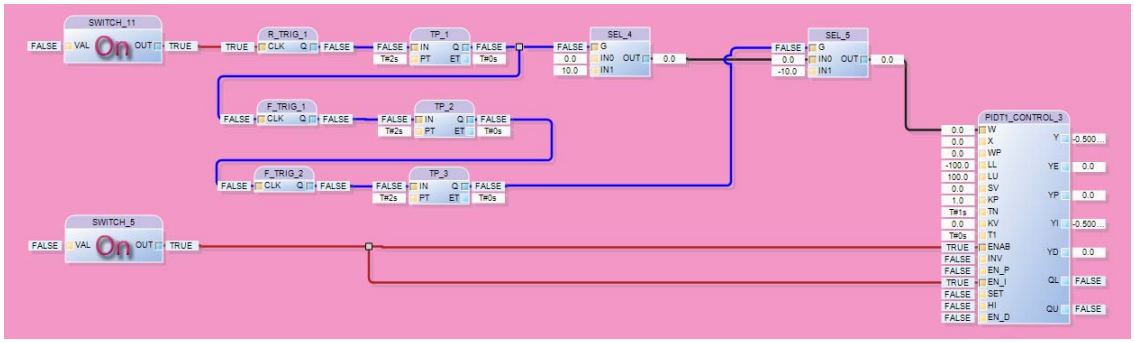


Abbildung 52: I-Anteil

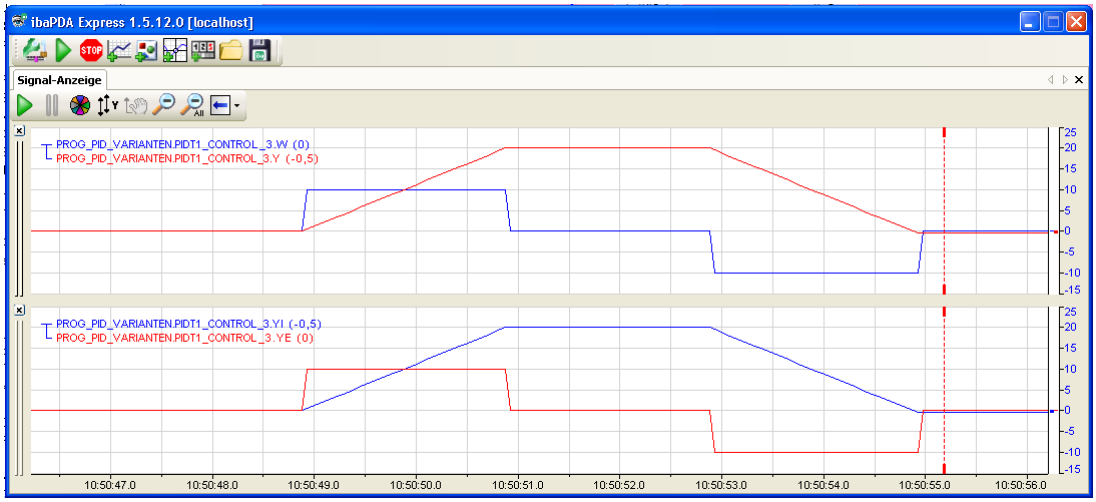


Abbildung 53: I-Anteil

7.3.3.6 DT1-Anteil: (Parameter KV,T1,EN_D)

Der DT1-Regelanteil berechnet sich aus

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

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

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

(Ta = Taskzeit)

Der Wert wird nur auf den Ausgangswert geschaltet, wenn EN_D gesetzt ist.

Beispiel 1

KV = 0.5

T1 = 1 s

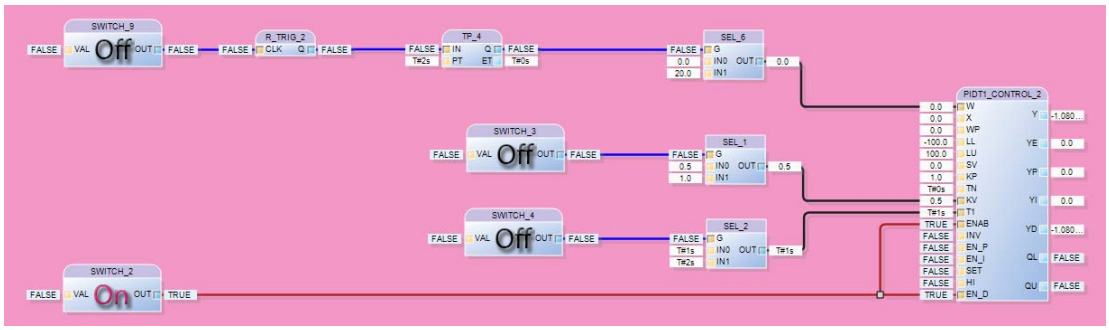


Abbildung 54: DT1-Anteil

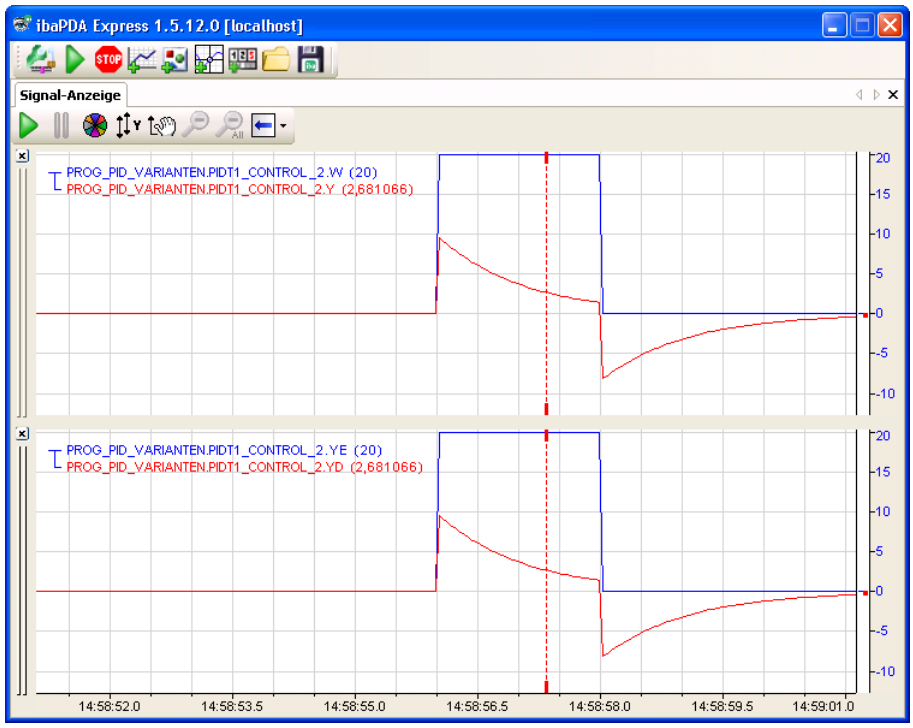


Abbildung 55: DT1-Anteil

Beispiel 2

KV = 1

T1 = 2 s

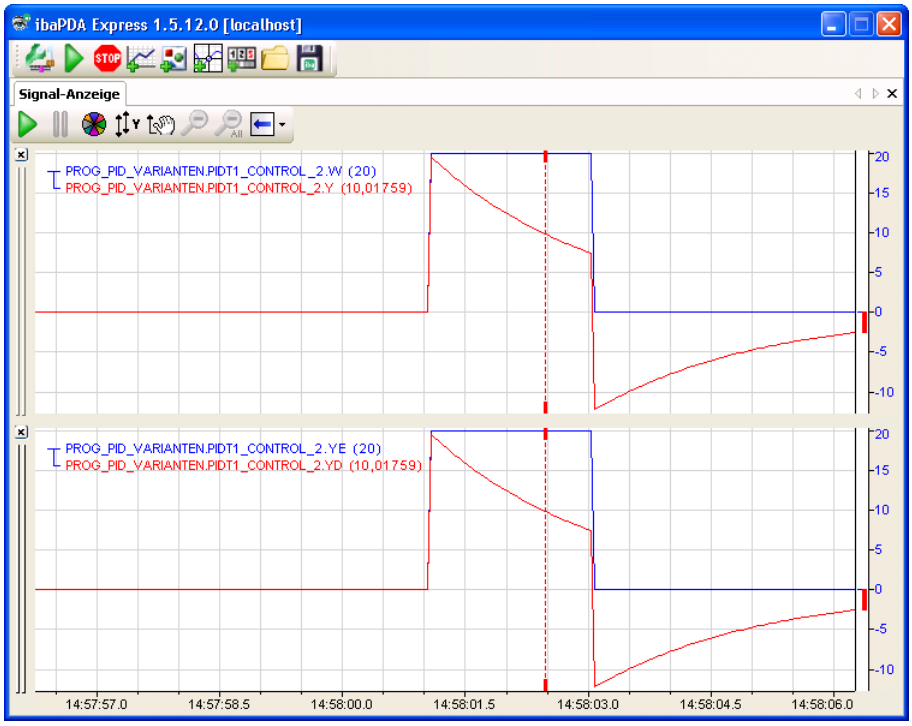


Abbildung 56: DT1-Anteil

7.3.3.7 PIDT1-Anteil – Verhalten gesamt

Beispiel 1

Beispiel für den kompletten PIDT1-Regler mit Signalverläufen.

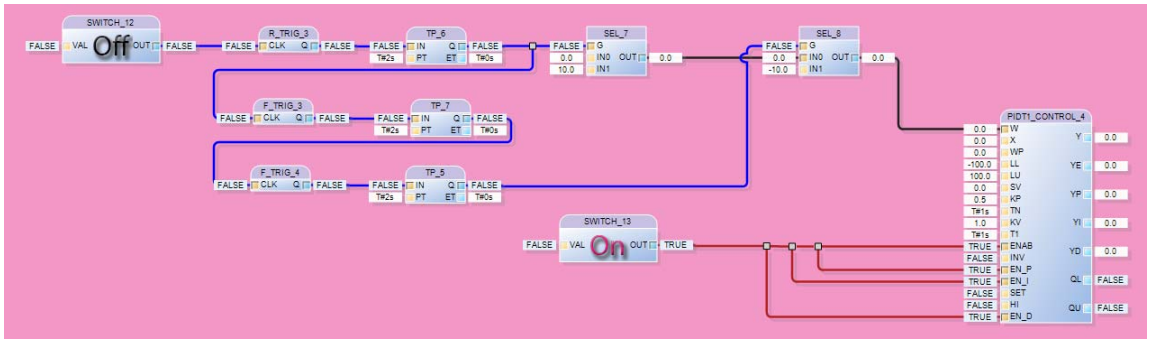


Abbildung 57: PIDT1-Regler mit Signalverläufen

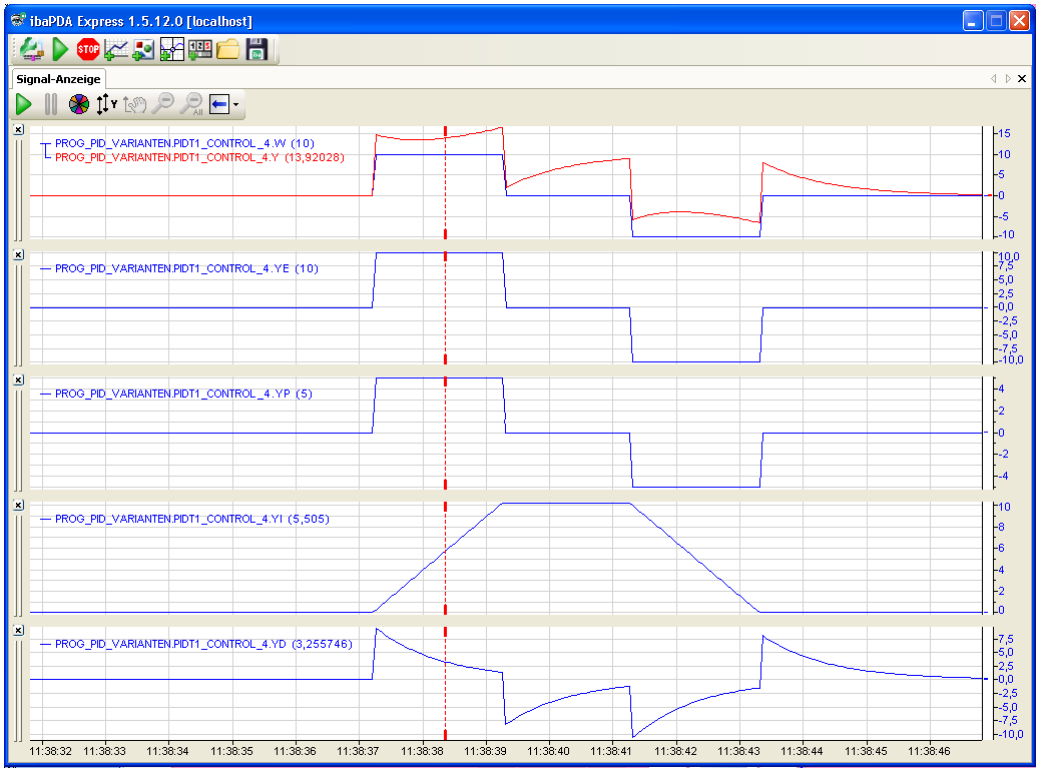


Abbildung 58: PIDT1-Regler mit Signalverläufen

7.3.4 RAMP

Rampenbaustein mit 2 verschiedenen Rampen: Manuell- und Automatik-Modus

- ❑ Sollwertbegrenzung
- ❑ Neuen Sollwert über Rampe anfahren
- ❑ Sollwert setzen
- ❑ Anzeige, wenn Grenzwerte überschritten

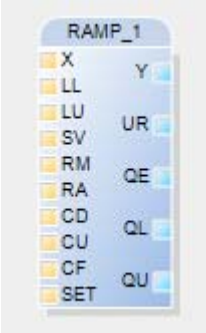


Abbildung 59: RAMP-Funktionsbaustein

7.3.4.1 Eingänge

Konnektor	Datentyp	Bedeutung/Verwendung
X	Lreal	Eingangswert (Sollwert)
LL	Lreal	Unterer Grenzwert
LU	Lreal	Oberer Grenzwert
SV	Lreal	Setzwert, Ausgang wird mit SET auf diesen Wert gesetzt
RM	Lreal	Manuelle Rampe (1/s), gilt für CD und CU
RA	Lreal	Automatische Rampe (1/s), gilt für CF
CD	Bool	Rampe fallend (manuelle Rampensteuerung)
CU	Bool	Rampe steigend (manuelle Rampensteuerung)
CF	Bool	Rampe gem. Eingangswert (automatische Rampensteuerung), Vorrang vor CD und CU
SET	Bool	Ausgangswert auf SV setzen.

7.3.4.2 Ausgänge

Konnektor	Datentyp	Bedeutung/Verwendung
Y	Lreal	Ausgangswert; $Y_n = Y_{n-1} + UR$ r = verwendete Rampe
UR	Lreal	verwendete Rampe (1/s)
QE	Bool	Ausgangswert = Eingangswert
QL	Bool	Unterer Grenzwert erreicht
QU	Bool	Oberer Grenzwert erreicht

7.3.4.3 Beispiel

Die Eingänge CD, CU und CF steuern die Rampen. Ist keiner der Eingänge aktiv, wird der letzte Ausgangswert festgehalten. Der Ausgang UR zeigt dann als benutzte Rampe den Wert 0 an.

Ist der Eingang CD aktiv, dann wird unabhängig vom Eingangswert der aktuelle Ausgangswert an der manuellen Rampe bis maximal zum unteren Limit runtergefahren.

Ist der Eingang CU aktiv, dann wird unabhängig vom Eingangswert der aktuelle Ausgangswert über die manuelle Rampe bis maximal zum oberen Limit aufgefahren.

Ist CD und CU gleichzeitig aktiv, dann wird UR auf 0 gesetzt. Der Ausgangswert verändert sich nicht.

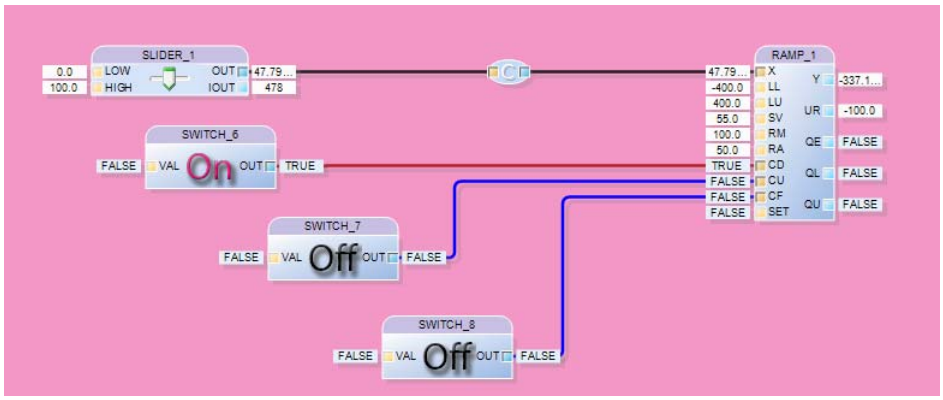


Abbildung 60: Steuern von Rampen

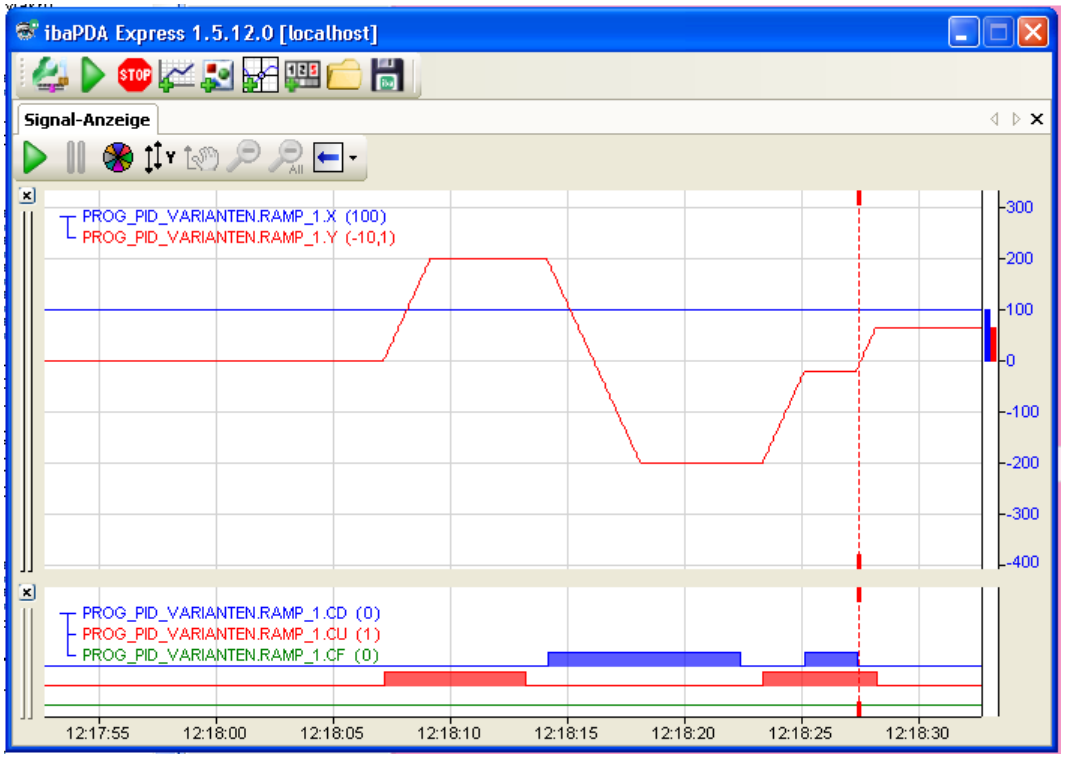


Abbildung 61: Steuern von Rampen

Ist der Eingang CF aktiv, dann folgt der Ausgangswert dem Eingangswert über die automatische Rampe. Überschreitet der Eingangswert die Grenzen, dann läuft der Ausgangswert über die Rampe nur bis zu den Grenzen.

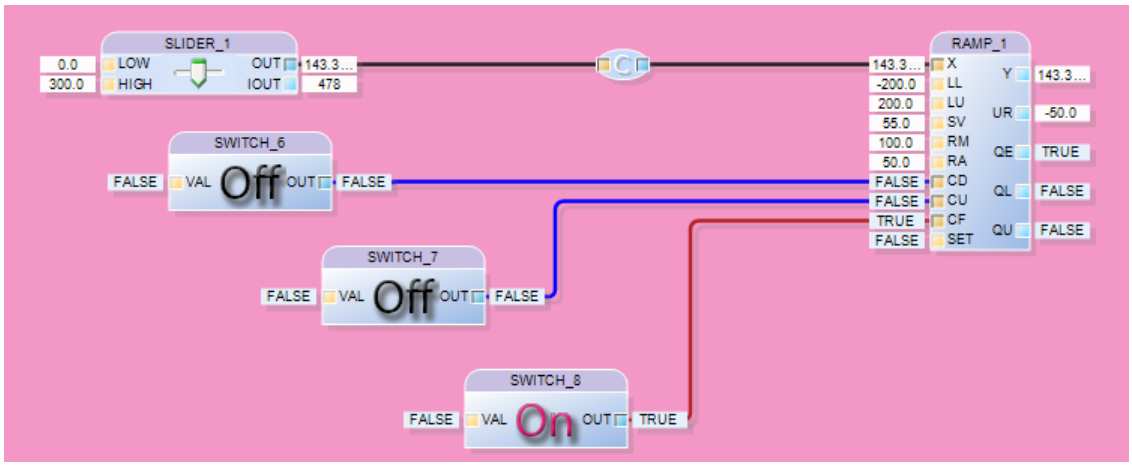


Abbildung 62: Steuern von Rampen

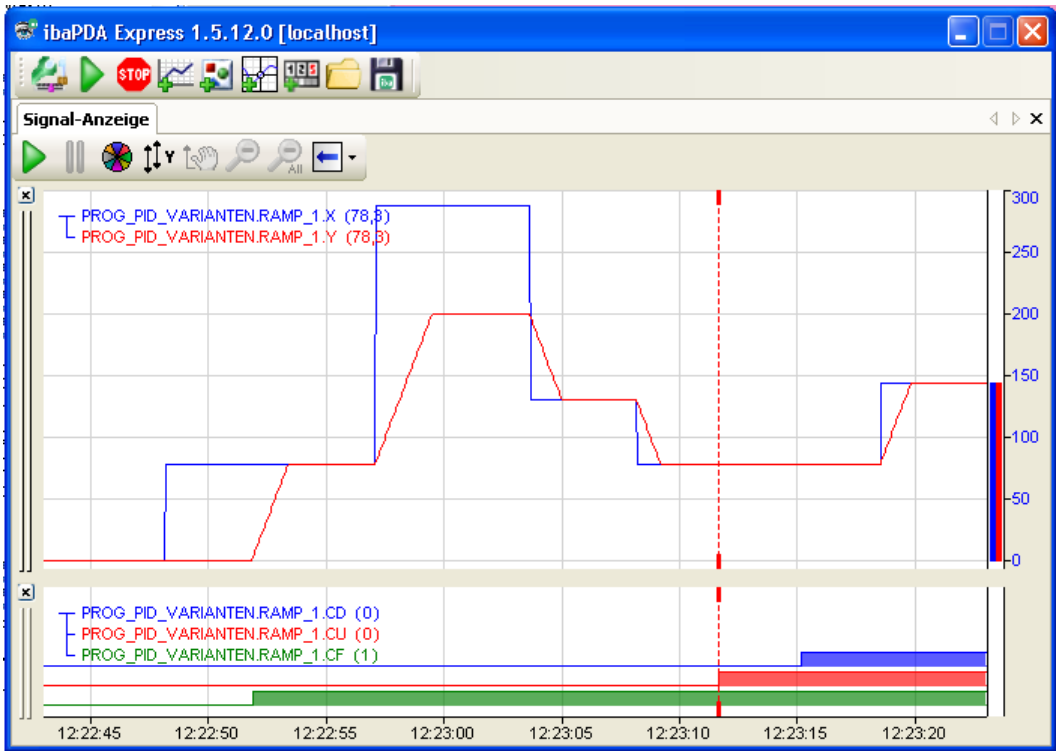


Abbildung 63: Steuern von Rampen

Ist CF gesetzt, dann haben die Eingänge CD und CU keine Auswirkung.

7.3.5 FUZZY_CONTROLLER

Fuzzylogic ist eine Theorie, die vor allem für die Modellierung von Unsicherheiten und Unschärfen von umgangssprachlichen Beschreibungen entwickelt wurde. Beispielsweise kann damit die Unschärfe von Angaben wie „ein bisschen", „ziemlich" oder „stark" mathematisch in Modellen erfasst werden.

Nützlich ist Fuzzylogic immer dann, wenn eine exakte logische oder mathematische Beschreibung für einen Prozess nicht vorhanden ist, nicht erstellt werden kann oder die Zusammenhänge zu komplex sind, aber z. B. als intuitives Knowhow eines Bedieners vorhanden ist.

Die Fuzzylogic basiert auf den Fuzzy-Mengen und sogenannten Zugehörigkeitsfunktionen, die Objekte auf Fuzzy-Mengen abbilden sowie passenden logischen Operationen auf diesen Mengen und deren Inferenz. Bei technischen Anwendungen müssen außerdem Methoden zur Umwandlung von Angaben und Zusammenhängen in Fuzzylogik betrachtet werden.

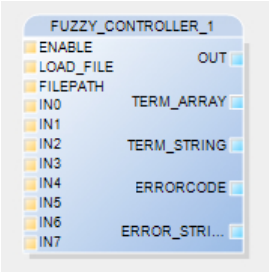


Abbildung 64: FUZZY_CONTROLLER-Funktionsbaustein

Für eine Regelung mit Fuzzylogic stellt ibaLogic den Funktionsbaustein FUZZY_CONTROLLER zur Verfügung.

Funktionsweise

Der Fuzzy-Controller ermittelt nach einem Parametersatz, den der Fuzzy-Controller einer auf die Anwendung bezogenen Parameterdatei entnimmt, einen Ausgangswert. Mit dem Start der Berechnung durch Konnektor ENABLE = TRUE werden die Parameter aus der als Pfadangabe an den Konnektor FILEPATH übermittelten Datei geladen und noch im selben Berechnungszyklus in ibaLogic für die Ermittlung des Ausgangswertes mit den neu geladenen Daten ausgeführt.

Die Parameterdatei kann über das Steuersignal LOAD_FILE auch während der laufenden Berechnung neu geladen werden, ohne die Ausgangswertermittlung oder Datenerfassung in ibaLogic zu unterbrechen. Der Aus-gangswert kann aus bis zu 8 Eingangswerten IN0 … IN7 gebildet werden und findet am Konnektor OUT als Sollwert oder Stellgröße eines Prozesses Verwendung.

7.3.5.1 Eingänge

Konnektor	Datentyp	Erklärung
ENABLE	Bool	Start des Berechnungsvorgangs, wenn am Eingang „TRUE" anliegt. Mit dem positiven Flankenwechsel werden die Daten aus dem Parameterfile übernommen.
LOAD_FILE	Bool	Übernahme der Daten des Parameterfiles mit dem positiven Flankenwechsel des Signals während der Berechnungszeit.
FILEPATH	String	Pfadangabe für die Daten des Parameterfile
IN0..IN7	Lreal	Eingangsdaten, nach denen der Ausgangswert berechnet wird.

7.3.5.2 Ausgänge

Konnektor	Datentyp	Erklärung
OUT	Lreal	Ausgangswert des Fuzzy-Controllers; im Fehlerfall oder enable = „FALSE" folgt output = 0,0
TERM_ARRAY	Lreal (0..8)	Höhe der Zugehörigkeitsgrade μ(x) der einzelnen linguistischen Terme, aus denen der Ausgangswert gebildet wird
TERM_STRING	String	Angabe der aktuellen linguistischen Terme, aus denen der Ausgangswert und die Prozentangabe der Zugehörigkeitsgrade gebildet werden.
ERROR_CODE	Int	Ausgabe des Fehlercodes
ERROR_STRING	String	Textausgabe einer kurzen Fehlermeldung.

Handhabung

Die Parametrierung erfolgt mit der Anwendung „ParamFuzzyTool.exe“.
Diese können Sie auf Anfrage beim iba-Support und Kontakt , Seite 347 erhalten.

Der Funktionsbaustein des Fuzzy-Controllers ist unter „Funktionsbausteine - SPECIALS" eingegliedert.

Der Pfad des Parameterfiles wird als String an den Konnektor FILEPATH angeschlossen. Mit der positiven Flanke des Boolean-Signals am Konnektor ENABLE werden die Daten aus der Parameterdatei übernommen und die Berechnung des Ausgangswertes startet umgehend. So lange das Enable-Signal „TRUE" ist, erfolgt die Berechnung des Ausgangswertes aus den LREAL-Werten an den Konnektoren IN0 ... IN7. Der Default-Wert des Konnektors ENABLE kann mit „TRUE" definiert werden. Dies bewirkt eine permanente Berechnung des Ausgangswertes.

Der Konnektor des Boolean-Signals LOAD_FILE am Funktionsblock des Fuzzy-Controllers bietet die Option an, während der laufenden Berechnung ein neues

Parameterfile zu laden. Die positive Flanke des LOAD_FILE-Signals bewirkt die Übernahme der Daten in den Fuzzy-Block als neue Parameter für die Grundlage der Berechnung des Ausgangwertes.

7.4 Anwenderspezifische Funktionsbausteine

ibaLogic verfügt über eine globale Bibliothek von vordefinierten Funktionsbausteinen. Dennoch kann es für eine effizientere Problemlösung notwendig sein, eigene Funktionsbausteine zu definieren. Zwischen drei Typen von Anwenderbausteinen wird unterschieden:

- Funktionsbausteine, die in ibaLogic in der höheren Programmiersprache Structured Text (ST) erstellt werden.
- Es gibt auch die Möglichkeit, vorhandene grafische Programmierung zu Makros zusammenzufassen.
- Funktionsbausteine, die außerhalb von ibaLogic in einer Hochsprache (C++, andere auf Anfrage z.B: FORTRAN) erstellt wurden und als DLL in ibaLogic eingebunden werden.

Nach der Erstellung werden alle diese Funktionsbausteine wie die Standardbausteine behandelt.

7.4.1 Funktionsbausteine

Für das Anlegen eines Funktionsbausteins gehen Sie mit dem Mauszeiger auf einen freien Platz im Programmfenster und wählen Sie im Kontextmenü „Neu - Neuer Funktionsblock…". Der Dialog „Funktionsblock erzeugen" wird angezeigt.

Im oberen Bereich befinden sich die Felder für die Bezeichnungen und die Tabelle zur Definition der Eingänge, der Ausgänge und der internen Variablen.

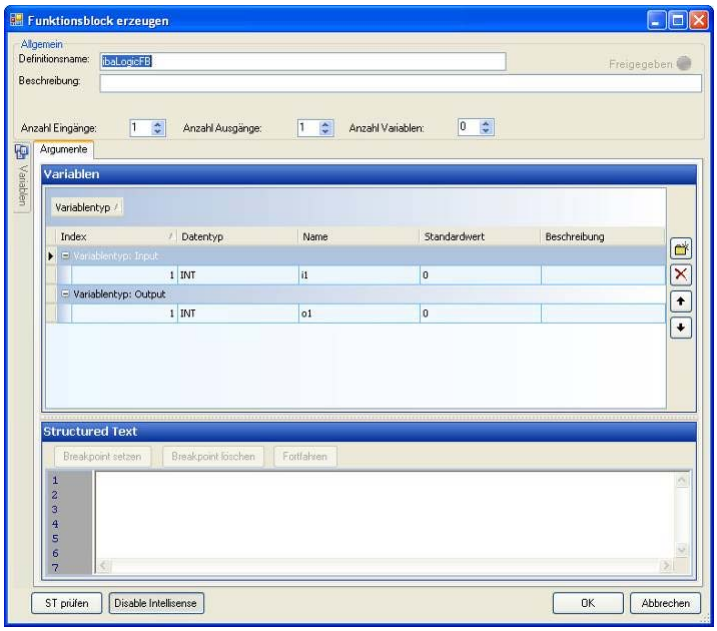


Abbildung 65: Dialog „Funktionsblock erzeugen"

7.4.1.1 Allgemeine Einstellungen

Definitionsname

Definitionsname ist der Name, mit dem der Baustein im Bausteinordner abgelegt wird. Aus diesem Namen wird durch Anfügen eines Index der Instanzname gebildet.

Hinweis Unterscheidung Definition - Instanz

Weitere Informationen siehe „Instanzansicht , Seite 56".

Hinweis

iba empfiehlt, unterschiedliche Präfixe für Funktionsbausteine und Makros festzulegen, z. B. „FB_" und „MB_". Die Einstellungen finden Sie unter Menü „Extras – Optionen – Funktionsbausteine".

Ebenso sind hier Vorbesetzungen für die Namen und Datentypen der Variablen zu finden. iba empfiehlt, die Vorbesetzungen der Namen „i", „o", „io" und „v" zu übernehmen. Datentypen können Sie nach Ihren Bedürfnissen einstellen.

Instanzname

Der Instanzname wird beim Anlegen noch nicht angezeigt. Erst beim Aufrufen eines im Projekt verwendeten Bausteins wird dieser angezeigt.

Beschreibung

In diesem Feld können Sie die Bausteinfunktion näher beschreiben. Diese Beschreibung wird Ihnen als Tooltip angezeigt, sobald Sie den Mauszeiger über dem Funktionsblocknamen in der Bibliothek bewegen.

Anzahl Eingänge / Ausgänge / Variablen

Hier legen Sie die Anzahl der verwendeten Variablen fest. Für jede Variable wird eine Zeile in der Tabelle angelegt.

Variablen

Die Liste der Variablen steht als Baumstruktur und als Tabelle zur Ansicht.

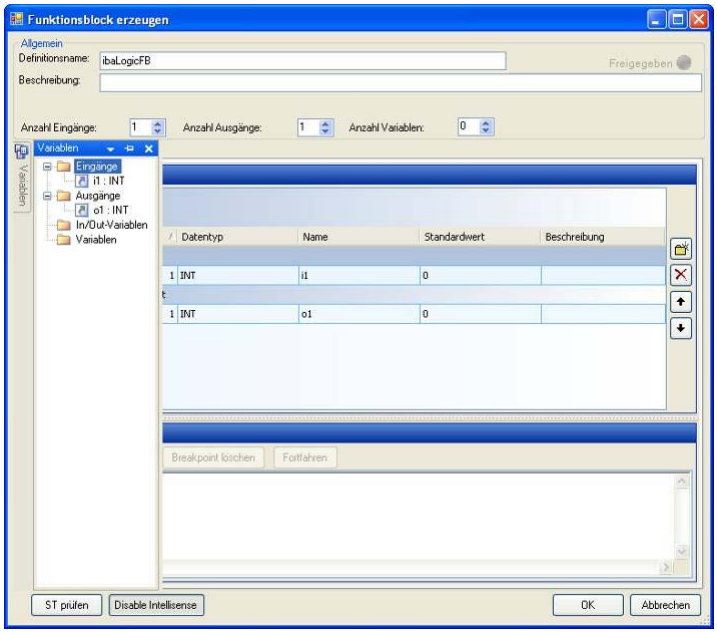


Abbildung 66: Dialog „Funktionsblock erzeugen" mit Liste von Variablen

➲ Die Baumstruktur öffnen Sie durch Klick auf die Lasche „Variablen" links von der Tabelle.

Diese Ansicht ist ausgeblendet, weil die Parametrierung nur in der Variablentabelle möglich ist.

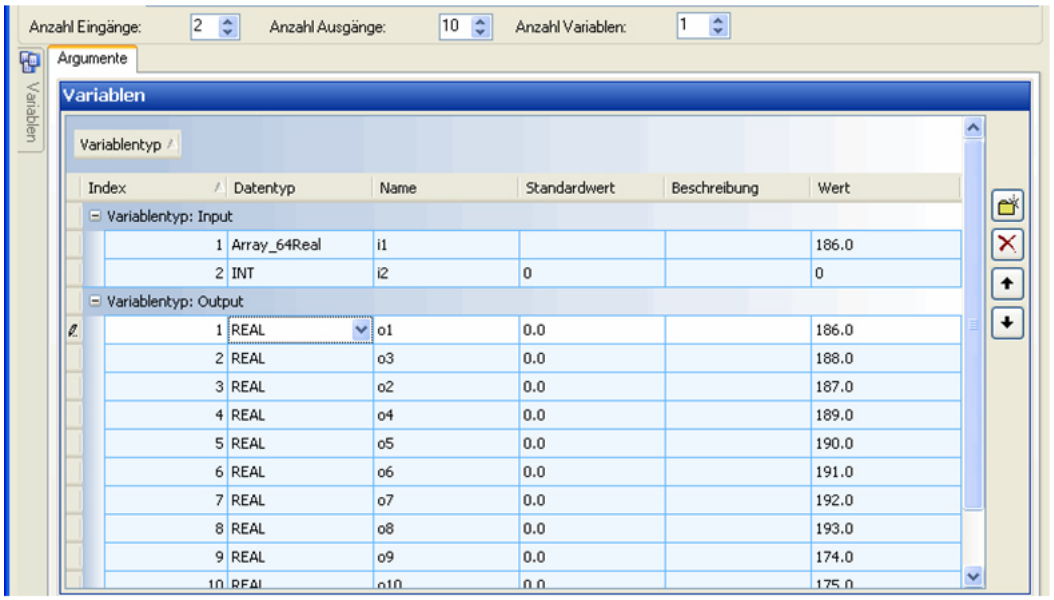


Abbildung 67: Funktionsblockvariablen

Variable

Spalte	Erklärung
Index	Jeder Variablentyp beginnt mit Index 1.
Datentyp	Auswahlfeld zur Übernahme eines definierten Variablentyps. Hier können Sie auch neue Anwendertypen erstellen. Die Voreinstellung finden Sie unter Menü „Extras - Optionen".
Name	Voreinstellung aus Präfix und Index. Sie können aber einen neuen Namen editieren.
Standardwert	Beachten Sie, dass die Notation der Werte abhängig vom Datentyp ist. Weitere Informationen siehe „Syntaxbeschreibung Structured Text , Seite 128". Der Wert ist der Variablen zugewiesen, solange keine Verbindung angeschlossen ist (bei Eingangsvariablen) oder keine Zuweisung innerhalb der Bausteincodes erfolgt.
Beschreibung	Textfeld, das als Tooltip im Bausteinordner angezeigt wird.
Wert	Aktuelle Werte der Variablen, bei Arrays und Strukturen wird nur das erste Element angezeigt (nur im Online-Mode).  **Tipp** Dieser Wert kann manuell geändert werden, wird aber im nächsten Zyklus neu berechnet und damit ggf. überschrieben.  **Wichtiger Hinweis** Wenn Sie im Online-Modus eine Verbindung zum Eingangskonnektor abziehen, dann bleibt der letzte Wert erhalten.

Mit den rechten Buttons können Sie die Reihenfolge der Variablen ändern, neue Variable an der markierten Stelle einfügen oder entfernen.

Symbol	Erklärung
	Einfügen eines Elements.
	Löschen eines Elements.
	Vertauschen der Elemente.

7.4.2 Structured Text-Editor

Im Structured Text-Editor können Sie die Funktionalität eines Funktionsbausteins festlegen.

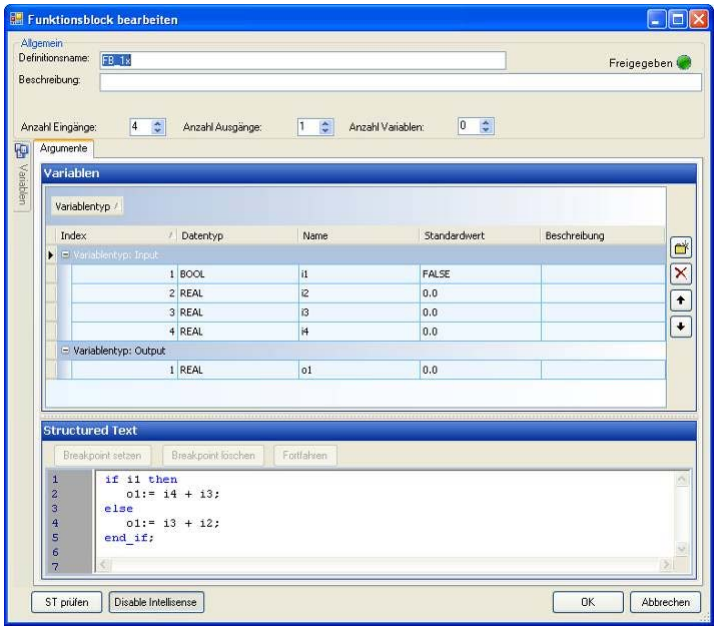


Abbildung 68: Structured Text-Editor

Oberhalb und unterhalb des Texteingabefeldes befinden sich folgende Buttons:

Button	Erklärung
Breakpoint setzen (nur aktiv im Online-Modus)	Mit Klick auf diesen Button wird der Ablauf an der Stelle im Programmcode angehalten an der die Cursor-Einfügemarke steht.
Breakpoint löschen (nur aktiv im Online-Modus)	Dient zum Entfernen der Breakpoints an der Einfügemarke.
Fortfahren (nur aktiv im Online-Modus)	Das Programm wird fortgesetzt. Im nächsten Zyklus bleibt der Programmablauf wieder an dem Breakpoint stehen.
ST prüfen	Syntaxtest des eingegeben Codes, ohne zu kompilieren
Enable/Disable Intellisense	Ein-/Abschalten des Hilfmittels IntelliSense

Gefahr!

Gefahr durch Anwendung von Funktionen im Online-Modus!

Wir raten dringend davon ab, diese Funktionen (Breakpoint setzen, Breakpoint löschen, Fortfahren) anzuwenden, wenn Sie ibaLogic im Online-Modus Ausgänge für Steuerungs- und Regelungsaufgaben verwenden, da dadurch die Gefahr von Personen- und Sachschaden besteht (siehe "Zeitverhalten , Seite 239").

7.4.2.1 IntelliSense

IntelliSense ist ein Hilfsmittel zur automatischen Eingabe-Vervollständigung. Dabei erhält der Programmierer während des Editierens zusätzliche Informationen und Auswahlmöglichkeiten, die ihm die Eingabe erleichtern.

Während der Erstellung von Bausteinen werden auch neue Variablen automatisch zu IntelliSense hinzugefügt.

Insbesondere das Arbeiten mit Strukturen wird wesentlich erleichtert, da bei Strukturvariablen nach Eingabe des Trennpunktes "." sofort alle definierten Elemente zur Auswahl vorgelegt werden.

Abbildung 69: Structured-Text-Editor mit aktivierter IntelliSense

Beispiel: Durch Eingabe von „IF" erscheint das Auswahl-Fenster von IntelliSense. Mit <Return> wird der markierte Teil eingetragen.

Anweisungen wie IF..THEN..ELSE / WHILE…/ REPEAT… werden nur mit der Eingabe des ersten Worts angeboten und können daher zu diesem Zeitpunkt komplett als Rahmen übernommen werden.

Die Auswahl erfolgt mit <Cursor auf> bzw. <Cursor ab>, die Übernahme mit <Tab>.

7.4.2.2 Syntaxbeschreibung Structured Text

Beispielsweise kann ein Structured Text folgendes Aussehen haben:

```
1 (* Differenz von i2 zu i1 *)
2 Differenz := i2 - i1;
3
4 (* Mittelwertbildung *)
5 Mittelwert := (i1 + i2) / 2.0;
```

Abbildung 70: Syntax von Structured Text

Schreibweisen:

- ❑ Kommentare werden in „(*" und „*)" eingeschlossen.
- ❑ Anweisungen sind mit Semikolon abzuschließen.
- ❑ Ergebniswert ist links von der Anweisung „:=" zu schreiben.
- ❑ Ausdrücke bestehen aus Operatoren und Operanden.

Wichtiger Hinweis

Neben den im Folgenden beschriebenen Operatoren und Anweisungen, können Sie teilweise Funktionen, die grafisch als Baustein zur Verfügung stehen, auch innerhalb von ST aufrufen. Hinweise ob und wie diese in ST angewendet werden können, finden Sie bei der Beschreibung der Funktionen in „Standard-Funktionsbausteine , Seite 292". Dort ist jeweils beim Baustein mit dem Schlüsselwort „ST:" ein Hinweis auf die Verwendbarkeit unter ST eingetragen.

7.4.2.3 Operatoren

Liste der Operatoren sortiert nach Priorität:

Operator	Beispiel	Wert des Beispiels	Beschreibung	Priorität
()	`(2+3) * (4+5)`	45	Klammerung	höchste
**	`3**4`	81	Potenzierung	
-	`-10`	-10	Negation	
NOT		NOT TRUE	logische Negation	
*	`10*3`	30	Multiplikation	
/	`6/2`	3	Division	
MOD	`17 MOD 10`	7	Modulo (Divisionsrest)	
+	`2+3`	5	Addition	
-	`4-2`	2	Subtraktion	
<, >, <=, >=	`4 > 12`	FALSE	Vergleich	
=	`T#26h = T#1d2h`	TRUE	Gleichheit	
<>	`8 <> 16`	TRUE	Ungleichheit	
&, AND	`TRUE & FALSE`	FALSE	Boolesches UND	
XOR	`TRUE XOR FALSE`	TRUE	Boolesches Exklusiv Oder	
OR	`TRUE OR FALSE`	TRUE	Boolesches Oder	niedrigste

7.4.2.4 Anweisungen

Schlüssel-wort	Beispiel	Beschreibung
;	`;`	Leeranweisung
:=	`Var1 := 12;`	Zuweisung des Wertes 12 an den linksstehenden Bezeichner.
f(i1, i2, …)	`o1 := concat(iDir, iFile, v1);`	Funktionsaufruf, siehe auch Beschreibung der Bausteine

IF	`IF i1 < i2 THEN`
	`  o1 := 1;`
	`[ ELSIF i1 =i2 THEN`
	`  o1 := 2; ]`
	`ELSE`
	`  o1 := 3;`
	`END_IF;`
		Bedingte Anweisung.
		Die Bedingung ist ein boolscher Ausdruck (der ein Ergebnis „TRUE" oder „FALSE" liefert)
		In eckigen Klammern […] stehen optionale Erweiterungen.

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;`
		Auswahlanweisung.
		Die Auswahl „i1" ist ein Ausdruck vom Typ ANY_INT oder ENUM.
		Pro Case gibt es eine oder mehrere Anweisungen.
		Ein Case kann mehrere Integer (3,4,5) oder Enumeratoren oder Bereiche von Integer (10…15) haben. Der ELSE-Zweig ist optional.
		In eckigen Klammern […] stehen optionale Erweiterungen.

Schlüssel-wort	Beispiel	Beschreibung

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 (* gefunden *)`
		Unbedingte Schleife (Iteration).
		Die Schrittweite (BY xx) ist optional. Wenn nicht vorhanden, ist die Schrittweite 1.
		Die Schleifenvariable ist vom Typ ANY_INT und darf innerhalb der Schleife nicht verändert werden.
		In eckigen Klammern […] stehen optionale Erweiterungen.
		Achtung
		Es besteht die Gefahr von Dauerschleifen

WHILE	`WHILE i1 > 1 DO`
	`  o1 := o1/2;`
	`END_WHILE;`
		Bedingte Schleife
		Achtung
		Es besteht die Gefahr von Dauerschleifen

REPEAT	`REPEAT`
	`  o1:= o1 * i1;`
	`UNTIL o1 > 10000`
	`END_REPEAT;`
		Bedingte Schleife
		Der Unterschied zu WHILE ist: Die Schleife wird mindestens einmal durchlaufen, auch wenn die Bedingung von Anfang an nicht erfüllt ist.
		Achtung
		Es besteht die Gefahr von Dauerschleifen

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 (* gefunden *)`
		Vorzeitiger Abbruch einer FOR-, WHILE- oder REPEAT-Schleife.
		Es wird die erste Anweisung nach dem nächsten Schleifenende ausgeführt, d. h. bei verschachtelten Schleifen wird in der nächsthöheren Ebene fortgefahren.
		In eckigen Klammern […] stehen optionale Erweiterungen.

RETURN	`oFLAG := FALSE;`
	`FOR ix:= 1 TO 100 [ BY 2 ]`
	`DO`
	`  IF o1[ix] = iy THEN`
	`  oFLAG := TRUE;`
	`  RETURN;`
	`  END_IF;`
	`END_FOR;`
		Rücksprunganweisung, vorzeitiger Abbruch des Funktionsbausteins.
		Beispiel: Ist der Wert iy im Array ix enthalten, dann ist das Ergebnis TRUE, andernfalls FALSE.
		In eckigen Klammern […] stehen optionale Erweiterungen.

ARRAY-Zugriff	`<ArrayType>[index,…]`
	
	`o1 := iArray[0];`
	`o1 := iArray[0,0,…];`
	`o1 := iArray[0][0];`
		Die Indizes sind in eckigen Klammern
		Zugriff auf 1-dimensionales Array
		Zugriff auf n-dimensionales Array
		Zugriff auf verschachteltes Array
		Weitere Informationen siehe „Gruppe ARRAY TYPE , Seite 148".
		Ausdrücke sind als Indizes nicht erlaubt, z. B.: MyArray[v1+1]

Schlüssel-wort	Beispiel	Beschreibung

ENUM-Zugriff	`Enumerator`
	
	`IF (i1 > 0) THEN`
	`  v1 := Vor;`
	`ELSIF (i1 < 0) THEN`
	`  v1 := Zurueck;`
	`ELSE`
	`  v1 := Halt;`
	`END_IF;`
		Beispiel: Die Variable v1 ist vom Typ „Schalter".
		„Schalter" ist ein ENUM-Type, „Vor", "Halt", „Zurueck" sind die Aufzählungen (Enumeratoren).
		Weitere Informationen siehe „Gruppe ENUM TYPE , Seite 146".

Struktur-Zugriff	`<STRUKTURNAME>.ELEMENT`
	
	`o1.Temperatur := i1;`
	`o1.Drehzahl := i2;`
		Die Strukturelemente sind durch "." von der Strukturvariablen getrennt.
		Beispiel:
		„o1" ist eine Variable vom Typ Struktur. „Temperatur" und „Drehzahl" sind Strukturelemente.
		Weitere Informationen siehe "Gruppe STRUCT TYPE , Seite 149".

7.4.2.5 Konstanten

Beschreibung	Beispiel
Integer und Bitstrings (außer BOOL)	`-12  0  123_456  +986`
Dezimale Darstellung	
Binäre Darstellung	`2#1111_1111 (255 dezimal)` `2#1111_0000 (240 dezimal)`
Hexadezimale Darstellung	`16#FF or 16#ff` `16#00F0_FFE0`
Real	`-12.0  0.0  0.4560  3.14159_26`

Real mit Exponent	`-1.34E-12 ODER -1.34e-12`
	`1.0E+6 ODER 1.0e+6`
	`1.234E6 ODER 1.234e6`

BOOL	`0 ODER FALSE` `1 ODER TRUE`

Zeit-Konstanten	Typkennzeichnung mit: „T#" ,
	Zeitangaben mit: „d" (day), „h" (hour), „m" (minute), „s" (second) und „ms" (millisecond).
	`T#12d12h17m42s`
	`T#16d_2h_5m`

Generell gilt in allen Angaben:	Ein einfacher Unterstrich zur optischen Strukturierung ist erlaubt.

7.4.2.6 Zeichenketten

Zeichenketten werden in einfache Anführungszeichen eingeschlossen.

Ein $-Zeichen gefolgt bei einer hexadezimalen Zahl wird als ASCII-Code interpretiert.

Hinweis

Die IEC erlaubt auch doppelte Anführungszeichen. Diese WSTRING sind derzeit nicht implementiert.

Erlaubte Sonderzeichen in Zeichenketten

Kombination	Interpretation beim Ausdruck
$$	Dollar-Zeichen
$'	Einfaches Anführungszeichen
$L or $l	Line feed (LF)
$N or $n	Newline (NL)
$P or $p	Form feed (page)
$R or $r	Carriage return (CR)
$T or $t	Tabulator

Hinweis

Ein RL (CarriageReturn/LineFeed) entspricht einem $N (NewLine) und wird in Funktionsbausteinen daher in der Weiterverarbeitung automatisch als $N angezeigt (siehe Eintrag in STANDARDWERT und Anzeige unter WERT).

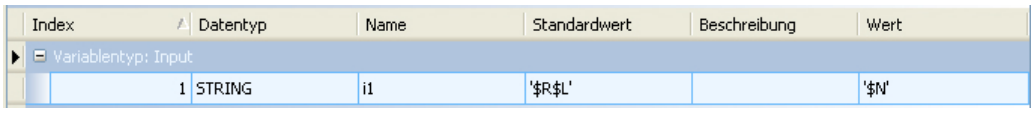


Abbildung 71: Variableneintrag

Besonderheiten und Beispiele für Zeichenketten

Beispiel	Erläuterung
''	Leerstring (Länge = 0)
'A'	String der Länge eins, enthält das A Zeichen
' '	String der Länge eins, enthält das Leer-Zeichen
'$''	String der Länge eins, enthält das einfache Anführungs-Zeichen
'"'	String der Länge eins, enthält das doppelte Anführungs-Zeichen
'RL'	String der Länge zwei, enthält das ASCII-Zeichen für CR und LF
'$$1.00'	String der Länge fünf, enthält damit „$1.00"
'ÄË' '$C4$CB'	String der Länge zwei, enthält "Ä" und "Ë"; Einmal direkt als ASCII-Zeichen und einmal in entsprechender Hex-Codierung der erweiterten Zeichentabelle (siehe "Zeichentabellen , Seite 338")

7.4.3 Makroblock

Makros verwenden Sie, um zusammengehörenden Funktionen zusammenzufassen und dadurch das Programmlayout übersichtlich zu gestalten.

Eigenschaften:

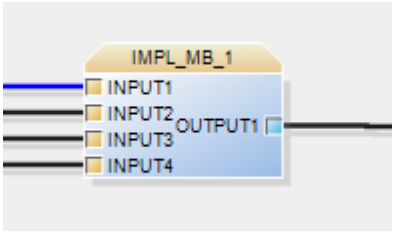
- ❑ Makros können exportiert werden.
- ❑ Makros können in den globalen Ordner kopiert werden und dadurch mehrfach, auch in anderen Projekten, verwendet werden.
- ❑ Makros können weitere Makros enthalten und natürlich auch selbsterstellte Funktionsbausteine.
- ❑ Innerhalb von Makros sind keine OTCs, keine Switches und keine Slider aber IPCs erlaubt. Verbindungen zu anderen Programmteilen sind nur mit Ein- und Ausgangskonnektoren erlaubt.
- ❑ Innerhalb von Makros können keine Hardware-Eingangs- und Ausgangsressourcen direkt verwendet werden.
- ❑ Ein erstelltes Makro kann auch wieder expandiert werden, d. h. das Makro wird aufgelöst und die enthaltenen Bausteine auf der nächsthöheren Ebene dargestellt.

7.4.3.1 Anlegen eines Makroblocks

Das manuelle Erstellen von Makros erfolgt auf dieselbe Weise wie das Anlegen von Funktionsbausteinen.

Vorgehen

1. Gehen Sie mit dem Mauszeiger auf einen freien Platz im Programmfenster und rufen Sie im Kontextmenü „Neu… - Neuer Makroblock…" auf. Sie erhalten eine Dialogbox zur Eingabe von Bausteinnamen und Variablen.
   Weitere Informationen siehe „Funktionsbausteine , Seite 123".
2. Verlassen Sie den Dialog mit <OK>. Es steht ein leerer Makroblock zur Verfügung.



3. Führen Sie einen Doppelklick auf den eingefügten Makroblock aus, um die innere grafische Programmieroberfläche zu öffnen.
4. Platzieren und verwalten Sie innerhalb des Makroblocks die Funktionsbausteine oder weitere Makroblöcke, so dass sich die gewünschte Funktionalität ergibt.

Beispiel

In diesem Beispiel wird der Integer-Eingang auf Änderungen überwacht, jede Änderung gezählt und am Ausgang angelegt.

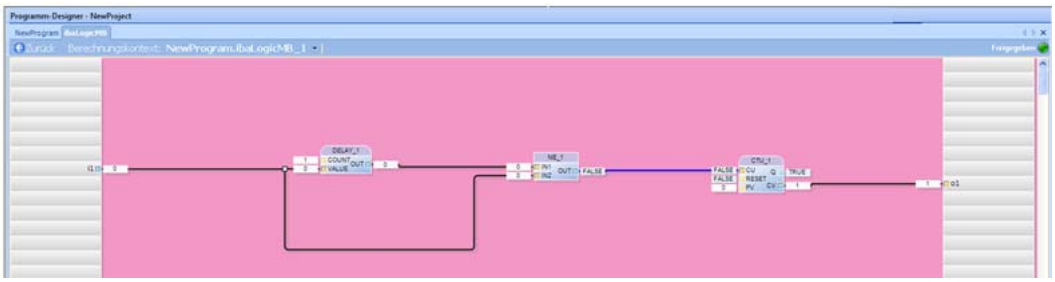


Abbildung 72: Integer-Überwachung

Für den Makroinhalt wird, wie bei jedem Programm, ein neues Register mit dem Makronamen innerhalb des Programmdesigners angelegt.

Beachten Sie hier den Berechnungskontext. Durch Klicken auf diesen kommen Sie in die aufrufende Ebene. Weitere Informationen siehe „Anordnung der Register und Programmierfenster , Seite 62".

7.4.3.2 Makro öffnen

Doppelklicken Sie auf die Makroblockinstanz in einem Programm oder Makro im Arbeitsbereich-Explorer.

7.4.3.3 Zusammenfassen von existierenden Teilen zu einem Makroblock

ibaLogic bietet die Möglichkeit mehrere schon existierende Bausteine zu einem Makroblock zusammenzufassen.

➲ Hierfür wählen Sie, wie im folgenden Bild dargestellt, die Bausteine, die zusammengefasst werden sollen.

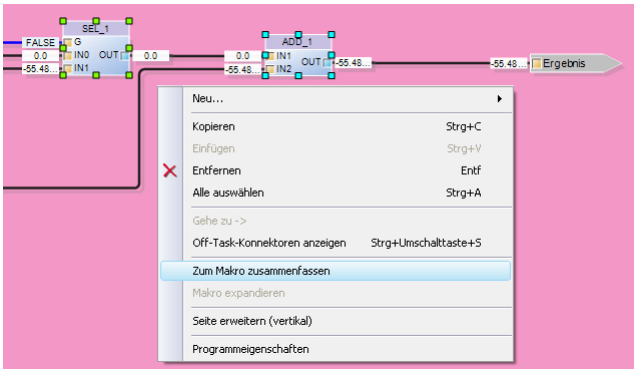
Hinweis

Achten Sie darauf,

- ❑ dass Sie bei der Auswahl auch die zugehörigen Verbindungen markieren.
- ❑ dass keine OTCs markiert sind.
- ❑ dass Verbindungen, deren Ziel- oder Quell-Baustein nicht mit angewählt wird, als Makro-Eingang oder Ausgang angelegt werden.

➲ Öffnen Sie das Kontextmenü über einem der ausgewählten Elemente.

➲ Wählen Sie „Zum Makro zusammenfassen" aus.
Der Dialog „Funktionsbaustein bearbeiten" wird angezeigt.



➲ Vergeben Sie dem neuen Makroblock einen sinnvollen Namen sowie für die Ein- und Ausgänge. Vergeben Sie aussagekräftige Bezeichnungen, die auch der IEC-Norm entsprechen.

Tipp

Sie können diese Anpassungen auch später durch Anklicken des erstellten Makros mit rechter Maustaste und Anwahl von Makroeigenschaften vornehmen.

Ergebnis

Als Ergebnis erhalten Sie einen neuen Makroblock (IMPL_MB_1) mit der gleichen Funktionalität wie die vorher selektierten Bausteine.

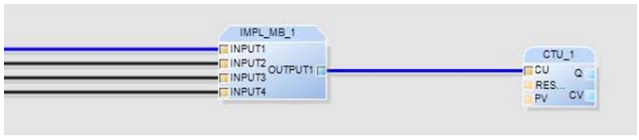


Abbildung 73: Makroblock (IMPL_MB_1)

➲ Durch Doppelklick können Sie das Makro öffnen und die grafischen Elemente weiter bearbeiten.

Im Online-Modus sehen Sie auch an den Value-Pads die aktuellen Werte in Abhängigkeit vom Berechnungskontext.

Weitere Informationen zum Berechnungskontext siehe „Anordnung der Register und Programmierfenster , Seite 62".

7.4.3.4 Expandieren eines Makroblocks

Ein bestehender Makroblock kann wieder expandiert werden. Dabei werden die enthaltenen Bausteine auf der nächsthöheren Ebene platziert.

Vorgehen

1. Markieren Sie dazu den Makroblock.
2. Wählen Sie im Kontextmenü „Makro expandieren".

Beispiel

Ein einfacher Makroblock mit einem internen Addierer, der die beiden Eingänge addiert, soll wieder aufgelöst werden.

Ergebnis

Nach dem Expandieren ergibt sich das folgende Bild:

- ❑ Der Addierer wurde herausgelöst.
- ❑ Die ursprüngliche Makro-Definition steht aber in der Baustein-Bibliothek weiterhin zur Verfügung.

Abbildung 74: Makroblock MB_Sollwert_1 Expandierte Schaltung

7.4.4 Erstellen eigener DLLs

Eigene Makros und Funktionsbausteine mit ST zu erstellen, ist eine sehr einfache Möglichkeit viele Aufgaben in der Automatisierungstechnik zu lösen. Aber so einfach es ist, die Makros und Funktionsblöcke zu erstellen, so einfach ist es auch, diese zu kopieren und deren Inhalt, d. h. ihre Funktion zu verstehen.

Manchmal ist es jedoch wünschenswert, die eigene technologische Kompetenz weniger offen zu legen und stattdessen eher, z. B. im Falle einer sehr intelligenten prozesstechnischen Lösung, der weiteren unkontrollierten Verbreitung des eigenen technologischen Know-hows vorzubeugen.

In einem solchen Fall ist die Möglichkeit der Erstellung eigener DLLs, die das Wissen nur in kompilierter Form enthalten und somit nicht einfach ausgelesen werden können, von Vorteil.

Diese speziellen Anbindungen können Sie auch realisieren,

- ❑ um komplexe Bausteine zu erstellen.
- ❑ um mit der Windows-Umgebung zu arbeiten.
- ❑ um Aufgaben in einem eigenen Thread laufen zu lassen u. a.

Auch die Integration einer anderen Hochsprache ist unter bestimmten Voraussetzungen damit möglich.

Die erstellte DLL ist dann in ibaLogic wie ein ganz normaler Funktionsbaustein, zu sehen mit Namen, Ein- und Ausgängen. Dieser ist von einem ST-Funktionsbaustein nur dadurch zu unterscheiden, dass im Programm-Teil kein Code zu sehen ist.

Andere Dokumentation

In diesem Kapitel ist nur eine kurze Übersicht gegeben, auf der Liefer-CD befinden sich detaillierte Anleitungen (z. B. Handbuch „ibaLogic_DLL-Erstellung_in_C++_v2.0_de.pdf" im Verzeichnis \ibaLogic_V4.x.x\Samples-DLL\).

Hinweis

Das Benutzen von DLLs ist lizenpflichtig. Ohne gültigen Dongle werden die DLLs nicht berechnet.

Compiler

Unterstützt werden alle DLLs, die in C++ oder Fortran geschrieben wurden.

Für das Schreiben und Kompilieren der DLLs, wurden folgende Compiler getestet:

- Microsoft Visual C++ 5.0
- Microsoft Visual C++ 6.0
- Intel Visual Fortran 10.0
- Microsoft Visual C++ 2005, 2008, 2010

Dabei gibt es noch Unterschiede für die beiden Geräteklassen bei ibaLogic (WinXP oder PADU-S-IT). DLLs für das Zielsystem PADU-S-IT müssen extra für Windows-CE kompiliert werden.

7.4.4.1 Benötigte Quelldateien und Beschreibungen

Die folgenden Quelldateien und Beschreibungen, die auf der ibaLogic Installations-CD mitgeliefert werden, sind für die DLL-Erstellung erforderlich:

Beschreibungen:

- Handbuch zum Erstellen einer DLL mit C++
  (für WinXP und PADU-S-IT)
- Handbuch zum Erstellen einer DLL mit Fortran
  (nur für Windows XP verfügbar)

Dateien: (der exakte Name der Dateien ist den Beschreibungen zu entnehmen)

- Rahmen-Datei:
  Enthält die Prozeduren und den DLL-Body; der Anwender kann Ein- oder Ausgaben hinzufügen oder Änderungen der Prozeduren InitEvaluation, Evaluate und ExitEvaluation vornehmen. Entweder in C++ oder Fortran.
- Weitere Dateien je nach Sprache:
  Zuordnung von DLL-Prozeduren und -Nummern, Schnittstellendefinition etc. Hier sind keine Veränderungen vom Anwender erforderlich.

7.4.4.2 Voraussetzungen und Hinweise

Bei der Verwendung von DLLs sollten Sie Folgendes beachten:

- ❑ Die DLL-Laufzeit verlängert die Laufzeit der Tasks, in der die DLL aufgerufen wird.
- ❑ iba empfiehlt, zeitaufwändige Funktionen in Threads auszulagern.
- ❑ Zum Testen der DLL kann ibaLogic als ausführendes Programm gestartet werden.

Wichtiger Hinweis

Von ibaLogic können Programmierfehler in einer DLL nicht erkannt und abgefangen werden, so dass derartige Fehler durchaus auch den „Absturz" von ibaLogic verursachen können. Achten Sie als Anwender beim Erstellen einer DLL selbst darauf.

7.4.4.3 Einbindung der DLL in ibaLogic

Wenn die DLL erstellt wurde, dann muss diese in ein Verzeichnis von ibaLogic kopiert werden. (Im Normalfall „C:\Program Files\iba\ibaLogic v4\Server\Dll").

Nach dem nächsten Neustart vom ibaLogic Server steht dann diese DLL als Funktions-Baustein im Verzeichnis „CUSTOM" zur Verfügung und kann wie jeder andere Baustein in ein Programm per Drag & Drop gezogen und eingebunden werden.

Beispiel

Die „Para_File_Read_Store_Dll" wurde erstellt und ins Verzeichnis kopiert. Diese ist in der Gruppe „CUSTOM" enthalten.

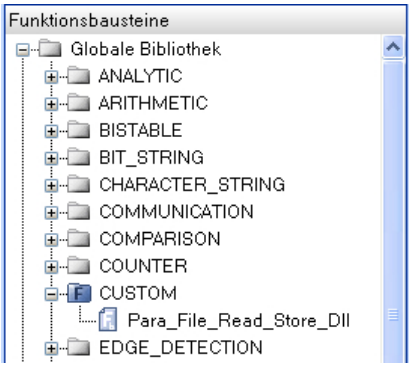


Abbildung 75: Para_File_Read_Store_Dll im Funktionsbaustein-Navigator

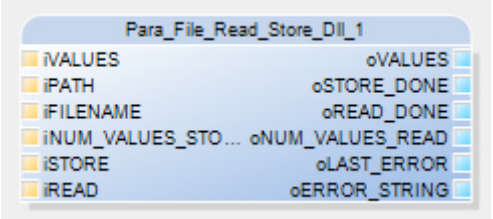


Abbildung 76: Para_File_Read_Store_Dll als Funktionsbaustein im Programm

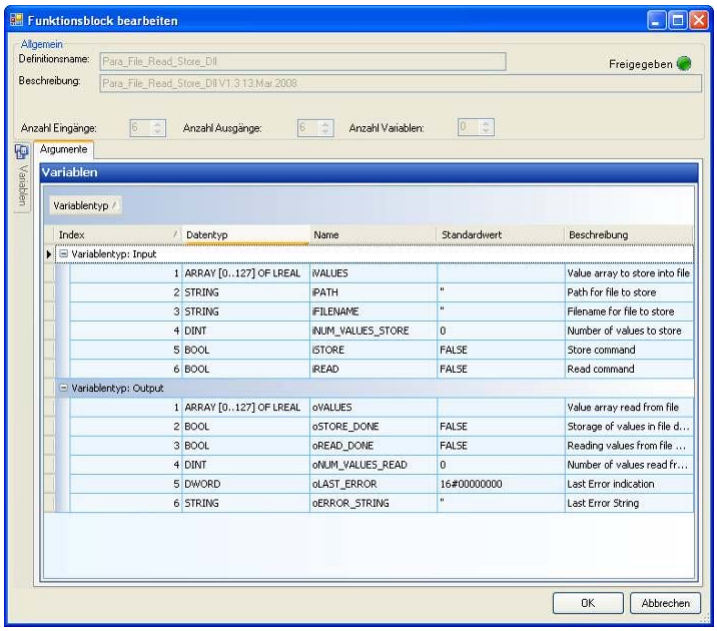


Abbildung 77: Para_File_Read_Store_Dll-Eigenschaften

➲ Durch Doppelklick auf den Funktionsbaustein erhalten Sie die Ansicht „Funktionsblock bearbeiten".
Der Code ist nicht zu sehen. Die Ein- und Ausgänge sowie deren Beschreibungen sind sichtbar.

7.5 Datentypen

Jeder Variablen ist ein Datentyp zugeordnet.

Im Gegensatz zur Version V3 unterstützt ibaLogic-V4 nicht nur die elementaren Datentypen und Arrays, sondern auch zusammengesetzte (Strukturen) und andere Anwenderdatentypen.

Die in ibaLogic verwendbaren Datentypen können in folgende Kategorien eingeteilt werden:

- ❑ Standard-Datentypen
- ❑ Zusammengesetzte Datentypen, wie ARRAY, STRUCT und ENUM,
- ❑ Abgeleitete Datentypen, die aus den beiden oben genannten Gruppen gebildet werden.

Sie haben als Anwender die Möglichkeit, eigene Datentypen der Kategorie „Zusammengesetzt" und „Abgeleitet" zu definieren.

Hinweis

Weitere Informationen siehe „Datentypen , Seite 290".

7.5.1 Datentyp definieren

➲ Klicken Sie auf die Schaltfläche „Datentypen".

Im Navigationsbereich wird der Verzeichnisbaum dargestellt.

Für die **nicht elementaren** Datentypen sind sowohl in einer globalen Bibliothek als auch unter jedem Projekt der Arbeitsgruppe die folgenden Ordner angelegt:

- **direct derived types**
  Standard-Datentyp mit festen Defaultwert
- **subrange types**
  Standard-Datentyp mit festen Defaultwert und eingeschränktem Wertebereich
- **string derived types**
  String-Datentyp mit fester Länge und Defaulttext
- **enum types**
  Aufzählungen: Anstatt Integerwerten werden Namen definiert
- **array types**
  Array von elementaren Datentypen mit fester Dimension und Tiefe
- **struct types**
  Struktur aus elementaren Datentypen

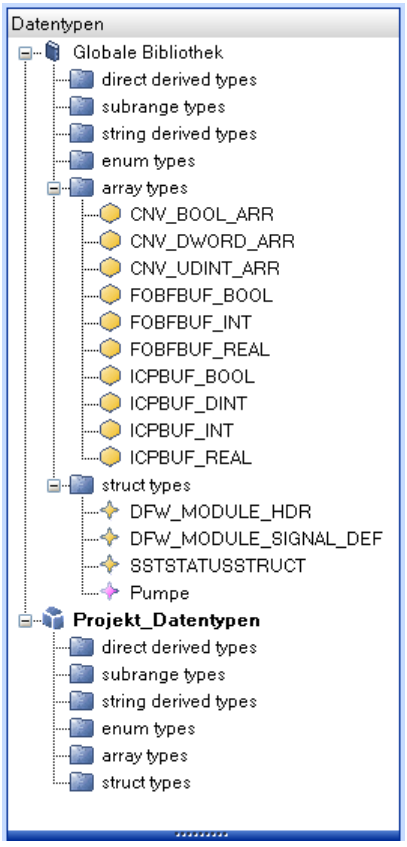


Abbildung 78: Datentypen

Unter „array types" und „struct types" sind Datentypen enthalten, die bereits durch ibaLogic vordefiniert sind.

Das sind:

- **CNV_BOOL_ARR / CNV_DWORD_ARR / CNV_UDINT_ARR**
  Eindimensionale Arrays mit 58 Elementen. Diese werden beim Import älterer ibaLogic V3 Projekte benötigt.
- **FOBFBUF_BOOL / _INT / _DINT / _REAL**
  Eindimensionale Arrays mit 256 Elementen, Verwendung im „Buffered Mode"
  Weitere Informationen siehe „Buffered Mode , Seite 195".
- **ICPBUF_BOOL / _INT / _REAL**
  Eindimensionale Arrays mit 1024 Elementen, Verwendung für den Anschluss von Analogeingaben im Zielsystem PADU-S-IT.
  Weitere Informationen siehe „Zielsystem ibaPADU-S-IT , Seite 209".
- **DFW_MODULE_HDR / DFW_MODULE_SIGNAL_DEF**
  Struktur zur Übergabe der Daten an den Baustein DAT_FILE_WRITE.
- **SSTSTATUSSTRUCT**
  Struktur zu Einkopplung der Diagnoseinformation der Profibus-Masterkarte SST.

Hinweis

Die Anzeige der vordefinierten und automatisch generierten Datentypen kann unterdrückt werden, wenn Sie im Menü „Extras – Optionen – Allgemein – System" anwählen und dort „Generierte Datentypen verbergen" aktivieren.

Auch wenn die Anzeige unterdrückt ist, können Sie die Datentypen im Programm verwenden.

Vorgehen

Sie können einen Datentyp auf unterschiedliche Weisen definieren:

- Unter dem Projekt
- In der globalen Bibliothek
- Bei der Erstellung eines Funktionsbausteins

7.5.1.1 Unter dem Projekt

1. Klicken Sie im Funktionsbaum mit der rechten Maustaste auf die gewünschte Kategorie unter dem Projekt.
2. Wählen im Kontextmenü „Neu".

7.5.1.2 In der globalen Bibliothek

1. Klicken Sie im Funktionsbaum mit der rechten Maustaste in der globalen Bibliothek auf die gewünschte Kategorie.
2. Wählen Sie im Kontextmenü „Neu".

7.5.1.3 Bei der Erstellung eines Funktionsbausteins

Im Auswahlfeld des Datentyps einer Variablen wird die Erstellung neuer Datentypen angeboten.

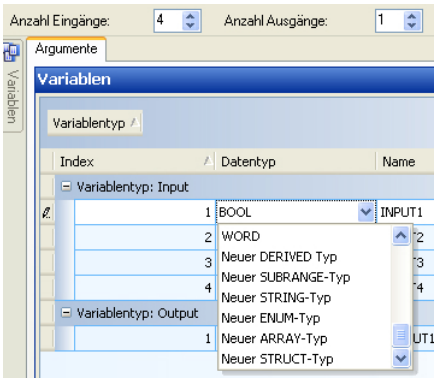


Abbildung 79: Datentypenerstellung eines Bausteins

Vorgehen

1. Erstellen Sie eine Variable mit dem entsprechenden Datentyp.
2. Testen Sie den erstellten Datentyp auf Fehlerfreiheit. Wenn der Test erfolgreich war, dann bestätigen Sie die Eingabe mit <OK>.

Ergebnis

Ist die Syntax ohne Fehler, dann wird der Datentyp in der Kategorie angelegt.

7.5.2 Datentyp ändern

Hinweis

Ein bereits verwendeter Datentyp kann nicht geändert werden.

Beim Verlassen des Dialogs mit <OK> oder <Übernehmen> werden Sie darauf hingewiesen, dass Sie eine Kopie unter einem anderen Namen anlegen können.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den Datentyp.
2. Wählen Sie im Kontextmenü „Eigenschaften".
3. Ändern Sie die Parameter.

7.5.3 Datentyp löschen

Voraussetzung

Sie verwenden den zu löschenden Datentyp nicht.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den Datentyp.
2. Wählen Sie im Kontextmenü „Entfernen" oder drücken Sie die Funktionstaste <Entf>.

7.5.4 Datentyp verwalten

In die globale Bibliothek kopieren

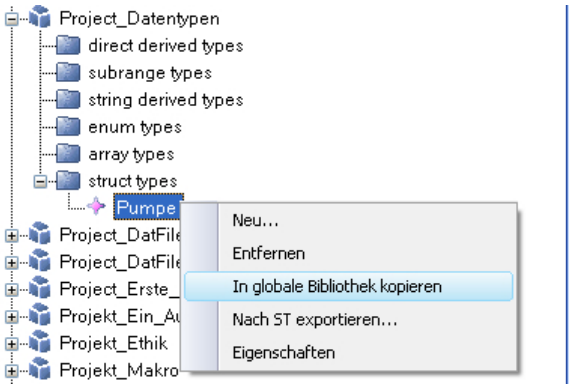
Soll ein Datentyp, der in einem Projekt definiert ist, auch in einem anderen Arbeitsbereich verwendet werden, dann muss er in die globale Bibliothek kopiert werden.

Hinweis

Wenn Sie Datentypen aus der globaler Bibliothek verwenden, werden diese automatisch ins Projekt kopiert.
Verwenden Sie einen Datentyp aus einem anderen Projekt, dann muss dieser zuerst in die globale Bibliothek kopiert werden.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den Datentyp unter dem Projekt.
2. Wählen Sie im Kontextmenü „In globale Bibliothek kopieren".



Hinweis

Wenn Sie ein Array in die globale Bibliothek kopiert haben und dann das Original ändern, haben Sie zwei Arrays mit gleichen Namen jedoch mit unterschiedlichem Inhalt.
Bei der Auswahl im FB wird vor das Array aus der globalen Bibliothek ein [GLB] gesetzt.

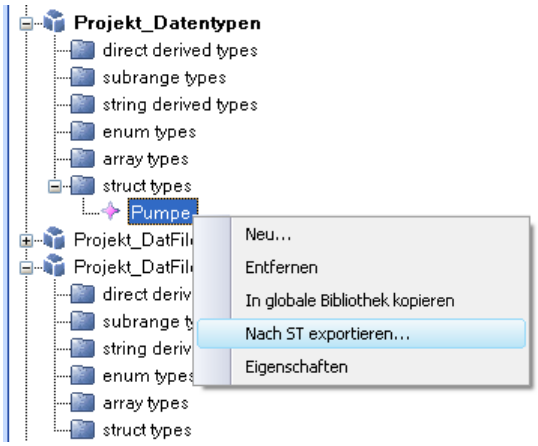
7.5.5 Datentyp exportieren

Soll ein Datentyp, der in einer anderen Datenbank unter der globalen Bibliothek oder in einem Projekt definiert ist, auch in einer anderen Datenbank oder einem anderen Programmierwerkzeug verwendet werden, dann muss der Datentyp per Textdatei exportiert werden.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den Datentyp.
2. Wählen Sie im Kontextmenü „Nach ST exportieren".

   Der Dialog „Exportieren" wird angezeigt.



3. Geben Sie Zielverzeichnis und Dateinamen an.

7.5.6 Datentyp importieren

Voraussetzung

ibaLogic ist nicht im Online-Modus.

Vorgehen

➲ Klicken Sie im Menü „Datei – Import – Structured Text".

7.5.7 Datentyp verwenden

Nach der Definition eines Datentyps können Sie diesen verwenden:

❑ bei der Erstellung eines Bausteins

❑ bei der Erstellung eines Struktur- und/oder Array-Datentyps

❑ beim Anlegen von Ein- und Ausgängen

7.5.7.1 Bei der Erstellung eines Bausteins

➲ Wählen Sie im Bausteineditor beim Anlegen einer Variablen unter der Spalte „Datentyp" den selbst definierten Datentyp aus.

7.5.7.2 Bei der Erstellung eines Struktur-Datentyps

➲ Wählen Sie im Datentypeditor beim Anlegen der Strukturelemente im Auswahlfeld „Datentyp" den selbst definierten Datentyp aus.

Hinweis

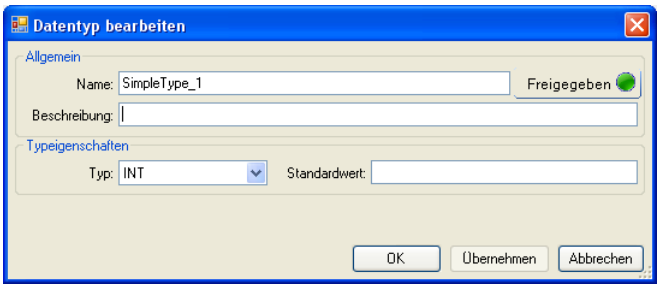
Auf Datentypen, die Sie in einem Projekt eines anderen Arbeitsbereiches definiert haben, haben Sie keinen direkten Zugriff.
Sie müssen hierzu über Export/Import oder über die globale Bibliothek gehen.

7.5.8 Anwenderdatentypen

Vorgehen

➲ Klicken Sie mit der rechten Maustaste auf eine Datentypgruppe.

Der Dialog „Datentyp bearbeiten" wird angezeigt.



Dieser Dialog besteht für alle Datentypen aus:

- ❑ Bereich „Allgemein" (für alle Datentypen gleich)
- ❑ Bereich „Typeneigenschaften"
- ❑ Bereich „Elemente"

Allgemein

- ❑ Name:
  Name für den eigenen Datentyp. Unter diesem Namen ist der Datentyp in den Auswahlfeldern zu finden.
- ❑ Beschreibung:
  Beliebiger Text zur Beschreibung dieses Typs. Die Beschreibung ist nur hier in der Definition des Datentyps zu sehen.

Typeigenschaften

- ❑ Typ:
  Definiert den Datentyp.
- ❑ Standardwert:
  Initialisierungswert (Voreinstellung)

7.5.8.1 Gruppe DIRECT DERIVED TYPE

Zur Definition eines elementaren Datentyps, dem ein neuer Name und ein Standardwert mitgegeben werden.

Damit werden z. B. Konstante, wie zum Beispiel die Zahl „Pi", mit dem Datentyp LREAL definiert.

7.5.8.2 Gruppe SUBRANGE TYPE

Dies ist ein Integer-Datentyp mit einem eingeschränkten Wertebereich und einem Standardwert.

Damit werden zum Beispiel Indizes für Arrays einer bestimmten Tiefe definiert.

Wichtiger Hinweis

Dieser Datentyp wird **nicht** begrenzt. Es erfolgt lediglich eine Fehler-Meldung zur Laufzeit bei der Bereichsüberschreitung.
Dieser Datentyp wird nur bei direkten Zuweisungen während des Kompilierens überprüft, zur Laufzeit gibt es keine Überprüfung zur Bereichsüberschreitung.

7.5.8.3 Gruppe STRING DERIVED TYPE

Dies ist ein String-Datentyp mit einer eingeschränkten Länge.

Damit werden konstant vorbesetzte Text-Strings, z. B. für Fehlermeldungstexte, definiert.

Hinweis

Die maximale Länge eines Strings beträgt 250 Zeichen.

7.5.8.4 Gruppe ENUM TYPE

Ein Datentyp der Kategorie ENUM TYPE ist eine Aufzählung.

Der Datentyp dient dazu, die Werte einer Variablen, z. B. die Stellung eines Schalters, symbolisch zu bezeichnen.

Beispiel: Datentyp „Schalter"

Sie wollen einen Datentyp „Schalter", der die 3 Stellungen VOR, HALT und ZURUECK hat, erstellen.

Dazu definieren Sie den ENUM TYPE-Schalter, mit Anzahl 3 und den Schalterstellungen als Enumeratoren.

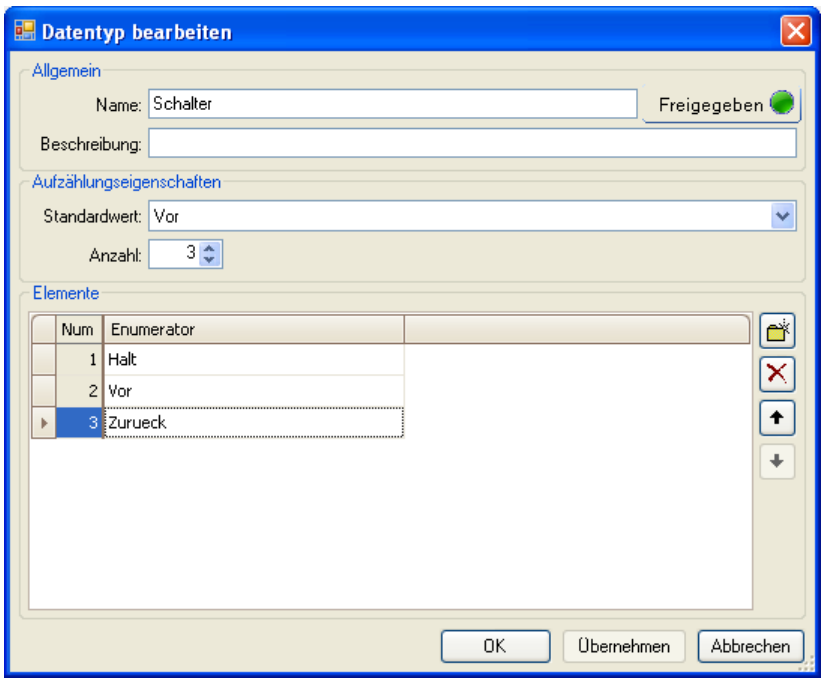


Abbildung 80: Dialog „Datentyp bearbeiten"

Beachten Sie, dass Sie in „Structured Text" auf die einzelnen Aufzählungswerte mit „Enumerator" zugreifen.

Werte zuweisen und abfragen:

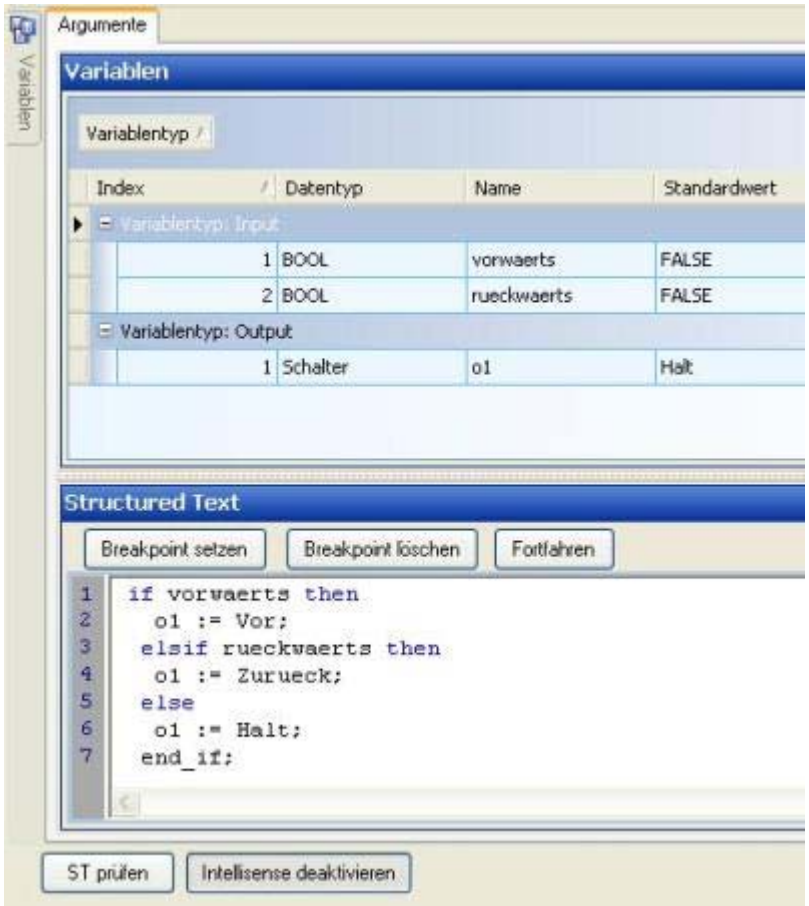


Abbildung 81: Variableneditor 1

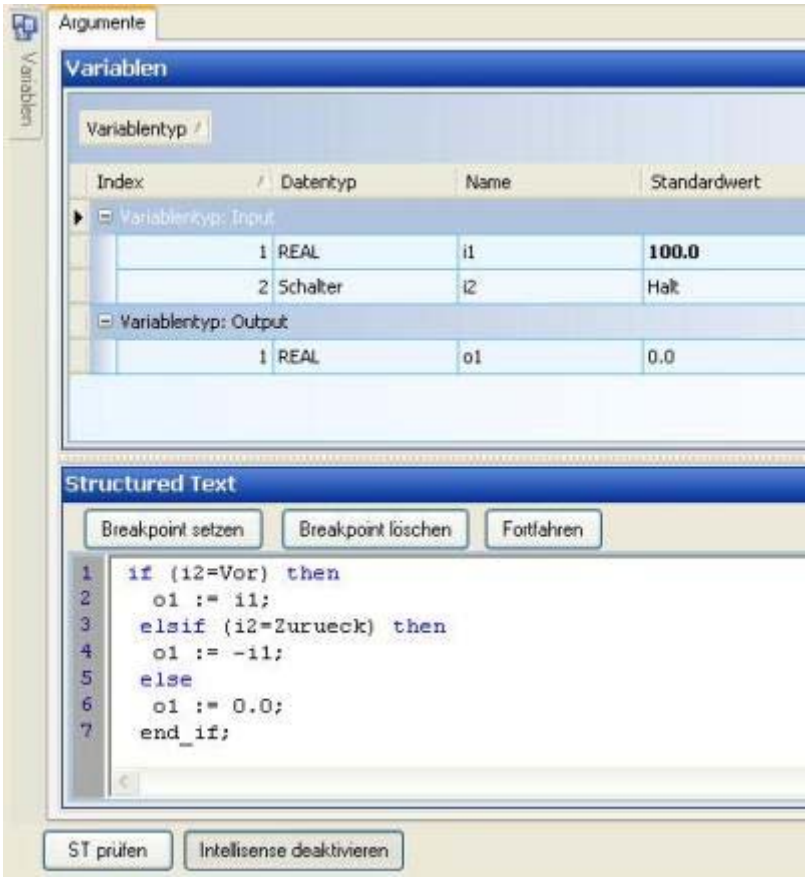


Abbildung 82: Variableneditor 2

Hinweis

Beachten Sie, dass Sie den Aufzählungen keine Integer-Werte zuweisen können.

Ausnahme:
Ein OPC-Konnektor wird als Enum-Typ deklariert und von außen gelesen oder geschrieben. In diesem Fall schreibt oder liest der OPC-Client die Aufzählungsnummer als Integer.

7.5.8.5 Gruppe ARRAY TYPE

Arrays sind ein- oder mehrdimensionale Felder. Alle Elemente eines Arrays haben denselben Datentyp. Dabei ist dieser nicht beschränkt auf die elementaren Datentypen, sondern Sie können auch Arrays von Anwenderdatentypen, Strukturen, Strings oder Arrays bilden.

Beispiel: 2-dimensionales Integer-Array

Parameter	Erklärung
Typ	Basistyp der Array-Elemente
Anzahl	Anzahl der Dimension

Standardwert	Default-Werte des Arrays
	Beispiele
	`1-dim-Array: [1.0,2.0,3.0]`
	`2-dim-Array: [[1.0,2.0,3.0],[4.0,5.0,6.0]]`

Untere/Obere Grenze	Der Wertebereich des Elementindex bestimmt die Tiefen der einzelnen Dimensionen. max. Wert: 0 bis 32766

Zugriffe auf die Elemente eines Arrays in Structured Text:

i1 ist eine Variable vom Array-Typ. *o1, o2...* sind Variablen vom Elementtyp;

- 1-dim- Array:
- 2-dim- Array:
- array_of_array:
- array_of_struct:

```
o1 := i1[0];
o1 := i1[0,0];      (* 1. Element der 1. Dim *)
o2 := i1[0,1];      (* 2. Element der 1. Dim *)
o1 := i1[0][0];     (* 1. Elem. des 1. Arrays *)
o2 := i1[0][1];     (* 2. Elem. des 1. Arrays *)
o3 := i1[1][0];     (* 1. Elem. des 2. Arrays *)
o1 := i1[0];        (* 1. Struktur des Arrays *)
o2 := i1[0].elem;   (* Elem. der 1. Struktur *)
```

Hinweis zu Structured Text

Die Indizes von Arrays können nur Variablen vom Datentyp „Int" sein. Im Gegensatz zu ibaLogic V3 sind **keine** Ausdrücke wie [ix+4] erlaubt.

7.5.8.6 Gruppe STRUCT TYPE

Unter einer Struktur können Sie - im Gegensatz zu Arrays - Variablen mit unterschiedlichen Datentypen zusammenfassen. Die Elemente müssen Sie einzeln definieren, dabei können Sie alle bisher definierten Datentypen, auch die Anwender-Datentypen und Arrays, verwenden.

Für jedes Strukturelement werden ein Name, eine Beschreibung und ein Default-Wert definiert.

Beispiel: Pumpe

Sie benötigen einen Datentyp "Pumpe" für die Pumpe "Typ E7F99" mit den Eigenschaften "Temperatur", Drehzahl", "Zustand" und "Fehler".

Dazu definieren sie den Datentyp "Pumpe" mit der Beschreibung "Pumpe Typ E7F99" und der Anzahl 4. Unter "Elemente" werden die dazugehörigen Eigenschaften definiert.

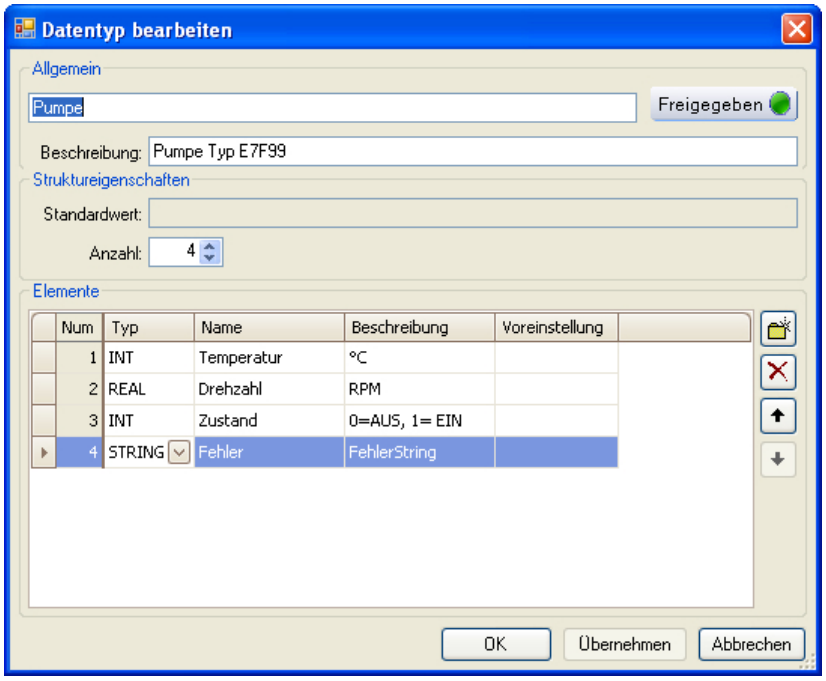


Abbildung 83: Dialog „Datentyp bearbeiten"

Parameter	Erklärung
Typ	Basistyp (auch Anwender-Datentypen sind zulässig).
Name	Name des Elements.
Beschreibung	Persönliche Erläuterung des Datentyps.
Voreinstellung	Initialisierung auf einen Standardwert (Voreinstellung).

Zugriffe auf die Elemente der Struktur in Structured Text:

```
(*o1 ist eine Variable vom Struktur-Typ Pumpe,   i1, i2... sind
Variablen vom Elementtyp; *)


         o1.Temperatur := i1;;          (* vom Datentyp INT *)
         o1.Drehzahl := i2;             (* vom Datentyp REAL *)

(*v1 ist eine Variable vom Struktur-Typ Pumpe;*)

if       (v1.Temperatur > 80 ) then
         v1.Zustand := 99;
         v1.Fehler := 'Temp. zu hoch';
else
         v1.Zustand := 0;
         v1.Fehler := 'kein Fehler';
end_if;
```

Abbildung 84: Struktur in Structured Text

8 Programmelemente

Ein grafisches ibaLogic-Programm enthält folgende Elemente:

- ❑ Bausteine
- ❑ Ein- und Ausgänge
- ❑ Verbindungen
- ❑ Automatisch eingefügte Konverter, Splitter, Joiner
- ❑ Kommentare

8.1 Programmelement anlegen

Es können alle Elemente erstellt werden, die ein grafisches ibaLogic-Programm enthalten kann.

Vorgehen

1. Öffnen Sie das Kontextmenü durch einen Klick mit der rechten Maustaste auf einen freien Bereich im Programmierfeld.
2. Wählen Sie im Kontextmenü „Neu…".
3. Wählen Sie das gewünschte Programmelement aus.

8.2 Programmelemente markieren

Einzelne oder mehrere Programmelemente können auf folgende Weise markiert werden.

Vorgehen

1. Wählen Sie das zu selektierende Element mit einem Klick mit der linken Maustaste an (Einfach-Selektion).
2. Wählen Sie die zu selektierenden Elemente mit einem Klick mit der linken Maustaste und gleichzeitiges Drücken der Taste <Shift> oder <Strg> an (Mehrfach-Selektion).
3. Ziehen Sie ein Rechteck (Lasso) über ein oder mehrere zu selektierende Elemente mit einem Klick der linken Maustaste auf.
   Damit werden auch ggf. vorhandene Verbindungslinien und Konverter zwischen den Bausteinen markiert.
4. Wählen Sie alle Elemente durch Drücken der Tasten <Strg> + <A> an.

Ergebnis

Die markierten Elemente werden mit grünen, blauen oder grauen Punkten angezeigt.

Bei Markierung von mehreren Elementen ist ein Element immer Grün. Das ist der Bezugspunkt der Gruppierung.

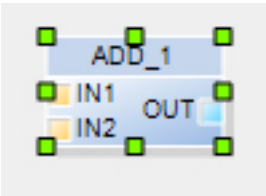
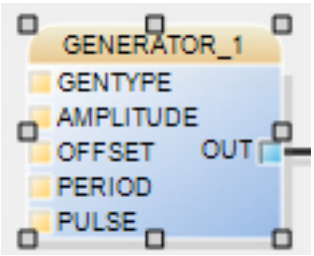
	Ein Element ist markiert.
	Ein Element ist markiert, aber der Fokus liegt auf einem anderen Element.
	Mehrere Elemente sind markiert. Das grüne ist immer das Hauptelement der markierten Gruppe.

Abbildung 85: Selektiertes Element

8.3 Programmelement verschieben

Die bereits markierten Bausteine (und Verbindungen) werden mit gedrückter linker Maustaste verschoben.

Vorgehen

➲ Verschieben Sie ein oder mehrere selektierte Elemente mit der gedrückten linken Maustaste.

Anmerkung

Für Linien gilt das nur eingeschränkt.

8.4 Programmelemente an einer Kante ausrichten

Alle Programmelemente können an einer Kante ausgerichtet werden. Das dient zur übersichtlichen Gestaltung des Bausteinplans.

Vorgehen

1. Markieren Sie die anzuordnenden Elemente (Bausteine, Intra-Page-Konnektoren und Off-Task-Konnektoren).
2. Wählen Sie die gewünschte Funktion im Menü „Funktionsplan - Ausrichten".

Anmerkung

Dies gilt nicht für Ein-/Ausgänge und Linien.

8.5 Programmelement kopieren

Einzelne oder mehrere Programmelemente können kopiert werden.

Vorgehen

1. Markieren Sie die zu kopierenden Elemente (Bausteine, Intra-Page-Konnektoren und Off-Task-Konnektoren).
2. Drücken Sie die Tastenkombination <Strg> + <C>, um die selektierten Elemente in die Zwischenablage zu kopieren.
3. Drücken Sie <Strg> + <V>, um die selektierten Elemente aus der Zwischenablage in das Programmierfeld einzufügen.

Tipp

Anstelle der Tastenkombination können Sie auch im Kontextmenü „Kopieren" und „Einfügen" wählen.

Wenn Sie mehrere Bausteine selektiert haben, dann werden auch die Verbindungslinien zwischen diesen Bausteinen mit kopiert.

Dies gilt nicht für Ein- und Ausgänge und einzeln markierte Linien.

8.6 Programmelement löschen

Einzelne oder mehrere Programmelemente können entfernt werden.

Vorgehen

1. Markieren Sie die zu löschenden Programmelemente (Bausteine, Intra-Page-Konnektoren und Off-Task-Konnektoren).
2. Drücken Sie die Taste <Entf>.

Anmerkung

Anstelle der Taste <Entf> können Sie auch im Kontextmenü „Entfernen" wählen.

8.7 Ein-/Ausgangsvariablen erzeugen

Voraussetzung

Sie haben die Schaltfläche „Eingänge - Ausgänge" angewählt.

Vorgehen

➲ Ziehen Sie eine Ein- oder Ausgangsvariable an eine Position in der linken bzw. rechten Ein- oder Ausgangs-Randleiste.

Hinweis

In einem **Programm** kann ein Eingang und ein Ausgang **nur einmal** angelegt werden. In einem **Projekt** kann ein **Eingang mehrfach** verwendet werden. Ein **Ausgang** nur **einmal**.

8.8 Grafische Verbindungen

In der grafischen Programmierung erfolgt die Übergabe der Ergebnisse einer Funktion an eine andere Funktion durch eine grafische Verbindung.

Dabei werden 3 Formen unterschieden:

- ❑ Direkte Verbindungslinien
- ❑ Intra-Page-Konnektoren
- ❑ Off-Task-Konnektoren

8.8.1 Direkte Verbindungslinien

8.8.1.1 Verbindungslinientypen

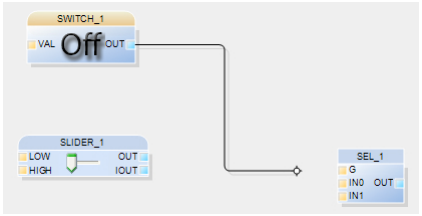
ibaLogic verwendet verschiedenfarbige Linientypen, die unterschiedliche Gruppen von Datentypen repräsentieren.

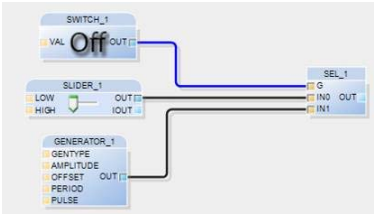
Linientyp	Erklärung
	Binäre Verbindungslinien werden je nach Zustand rot (TRUE) oder blau (FALSE) dargestellt.
	Arrays werden mit grünen Linienzügen dargestellt. Es können nur Arrays gleicher Länge und gleichen Datentyps miteinander verbunden werden.
	Strukturen werden orange dargestellt.
	Enum types werden gelb dargestellt.
	Alle anderen elementaren Datentypen werden durch schwarze Verbindungslinien gekennzeichnet (z. B. INT, REAL...)

8.8.1.2 Direkte Verbindungslinien erstellen

Vorgehen

1. Klicken Sie mit der Maus den Ausgangskonnektor eines Bausteins an.
2. Ziehen Sie bei gedrückter linker Maustaste eine Verbindungslinie zum Eingangskonnektor eines Bausteins.





Anmerkung

Wenn das Ergebnis eines Bausteins in mehreren Bausteinen verwendet wird, dann erzeugen Sie eine Verzweigung, indem Sie eine Linie von einem Eingangskonnektor auf eine bereits vorhandene Verbindungslinie ziehen.

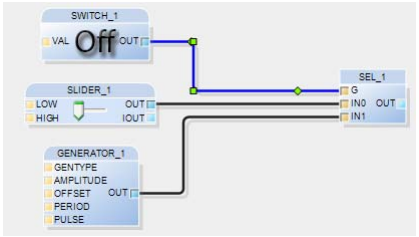
In der Nähe eines anschließbaren Konnektors oder einer anschließbaren Linie, springt die Maus auf den Konnektor bzw. die Verbindungslinie (Magnetwirkung).

8.8.1.3 Direkte Verbindungslinien ändern

Vorgehen

1. Markieren Sie die Verbindungslinie, die Sie ändern möchten.

   Die Markierung wird durch kleine grüne Quadrate und Rauten angezeigt.



2. Verändern Sie den Linienverlauf durch Verschieben der grünen Quadrate mit der Maus.
3. Klicken Sie die Verbindungslinie mit der Maus an der grünen Raute an, um den Linienanschluss zu verdrahten.
   Ziehen Sie das Ende in einen leeren Bereich, wird die Verbindungslinie gelöscht.
   Ziehen Sie das Ende auf einen anderen Konnektor. Die Verbindungslinie wird neu verbunden.

Hinweis

Verbindungslinien, die Sie manuell angeordnet haben, werden beim Bewegen des zugehörigen Bausteins vom Autorouter neu berechnet. Damit werden Ihre Änderungen verworfen.

8.8.2 Intra-Page-Konnektoren

Ein Intra-Page-Konnektor (IPC) stellt lediglich eine zeichnerische Vereinfachung dar. Der IPC ersetzt dabei eine Verbindungslinie.

Dies ist dann zu empfehlen, wenn sehr viele Objekte auf einer Seite verbunden werden müssen oder „lange" Verbindungen über mehrere Seiten erforderlich sind. Der IPC ist kein Programmier-Objekt, sondern agiert nur als Linienersatz.

Der IPC kann - wie direkte Verbindungslinien - nur innerhalb einer Programm- oder Makroebene angewendet werden. Verbindungen aus einem Makro in die Aufrufebene sind nicht möglich. Dazu müssen Sie Ein- bzw. Ausgänge im Makroblock definieren.

8.8.2.1 Intra-Page-Konnektor erstellen

Intra-Page-Konnektor als Linienersatz erstellen.

Voraussetzung

Einen IPC an einem Eingangskonnektor zu erzeugen ist nur möglich, wenn vorher eine „IPC-Quelle" definiert wurde.

Vorgehen

➲ Drücken Sie die Taste <Strg> und ziehen Sie gleichzeitig eine Verbindungslinie von einem Ausgangskonnektor in eine freie Stelle im Programmierfeld.

➲ Menü-Vorgehen ähnlich wie bei Off-Task-Konnektor erstellen. Allerdings größere Änderung.

- IPC Quelle anlegen
- IPC verbinden (wie hier beschrieben)
- IPC verbinden per Menü

Anmerkung

Um einen IPC an einen Eingang zu "verdrahten", verfahren Sie ebenso.
Halten Sie die <STRG>-Taste gedrückt und ziehen von einem Eingangskonnektor die Linie in einen freien Bereich im Programmfeld. Daraufhin öffnet sich der Dialog "Vorhandene IPCs". Wählen Sie hier den entsprechenden IPC aus und verlassen Sie den Dialog mit <OK>.

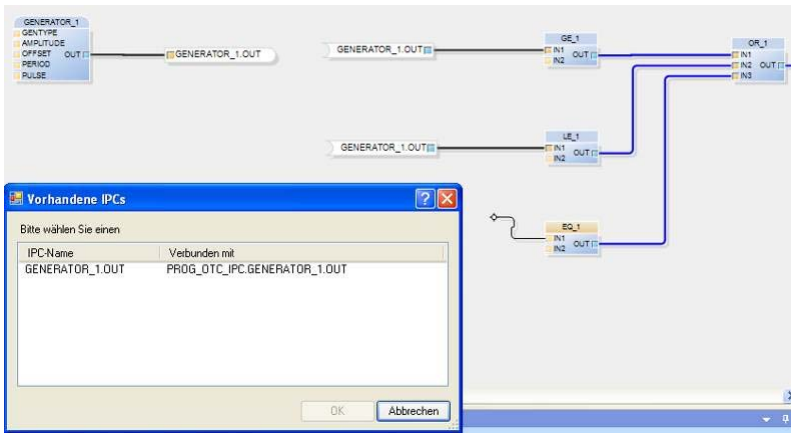


Abbildung 86: Eigenschaftsfenster

8.8.2.2 IPC-Namen ändern

ibaLogic erstellt automatisch einen Namen, bestehend aus „Blockinstanzname.Konnektorname". Dieser Name kann geändert werden.

Vorgehen

1. Doppelklicken Sie die IPC-Quelle.
   Der Dialog „IPC bearbeiten" wird angezeigt.
2. Vergeben Sie einen Namen und einen Kommentar für die IPC-Quelle. Vergeben Sie aussagekräftige Bezeichnungen.

Ergebnis

Die Änderung wird automatisch in alle verbundenen „IPC-Ziele" übernommen. „IPC-Ziele" können nicht direkt geändert werden.

8.8.2.3 IPC verfolgen

Die entsprechende Programmseite des gewählten IPC laden.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf einen IPC.
2. Wählen im Kontextmenü „Gehe zu ->".
   Sie sehen dann den verbundenen Erzeuger (Ausgang) und alle verbundenen Verbraucher (Eingänge).
3. Klicken Sie eine angezeigte Verbindung an.

Ergebnis

Es wird die entsprechende Programmseite geladen und der IPC markiert.

Kontextmenü	Erklärung
01. -> Generator_1.OUT	Erzeuger
02.°Generator_1.OUT ->	Markierter IPC
03. Generator_1.OUT ->	Verbraucher

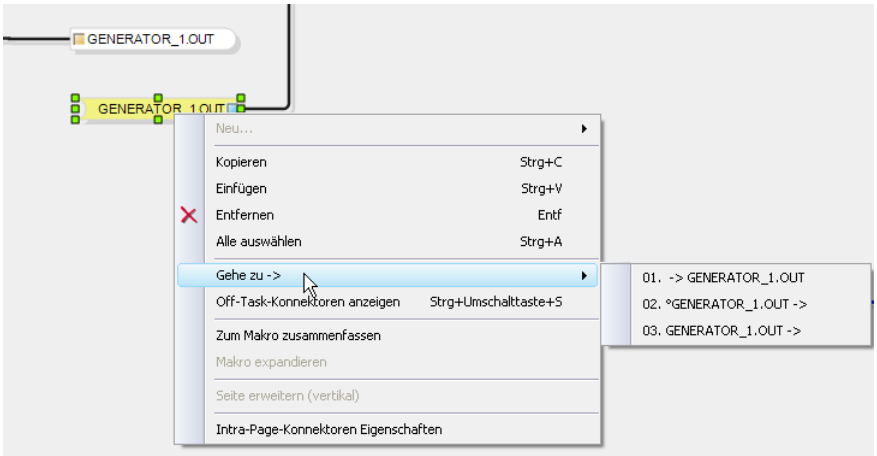


Abbildung 87: IPCs verfolgen

8.8.3 Off-Task-Konnektoren

Off-Task-Konnektoren (OTC) dienen als programmübergreifende Verbindungselemente und sind immer dann erforderlich, wenn zwischen Programmen kommuniziert werden soll.

Zusätzlich können OTCs für OPC-Clients lesbar und beschreibbar eingestellt werden.

8.8.3.1 Off-Task-Konnektor erstellen

Vorgehen

1. Platzieren Sie den Mauszeiger an einer freien Stelle im Programmierfeld.
2. Öffnen Sie das Kontextmenü mit einem Klick der rechten Maustaste.
3. Wählen Sie „Neu… - Neuer Off-Task-Konnektor".
   Der Dialog „Off-Task-Konnektor bearbeiten" wird angezeigt.

4. Vergeben Sie die notwendigen Parameter.

Hinweis

Der OTC Name muss der IEC-Namenskonvention entsprechen.
Siehe "Namenskonventionen , Seite 288".

Erstellen einer programmübergreifenden Verbindung

Vorgehen

Methode 1:

1. Erzeugen Sie zunächst den Ausgangs-OTC (Quelle) durch Ausfüllen des Dialogs.
2. Kopieren Sie den Ausgangs-OTC.
3. Fügen Sie den Ausgangs-OTC in das Zielprogramm ein.
   Dabei werden die Parameter übernommen, aber die Richtung umgedreht.

Methode 2:

1. Erstellen Sie im Zielprogramm einen OTC.
2. Wählen Sie den Namen des dazugehörigen Ausgangs-OTC aus dem Auswahlfeld.
   Dabei werden die anderen Parameter übernommen.
3. Sie müssen die Richtung auf „Eingang" stellen.

OPC-Eigenschaften

Folgende OPC-Eigenschaften können dem OPC mitgegeben werden.

Auswahlfelder OPC-Eigenschaften	Erklärung
OPC sichtbar	Legt fest, ob dieser Konnektor im OPC-Namensraum sichtbar ist.
OPC beschreibbar	Legt fest, ob ein OPC-Client auf diesen Konnektor schreiben darf.

Weitere Informationen siehe „Parametrierung der OPC-Variablen , Seite 214".

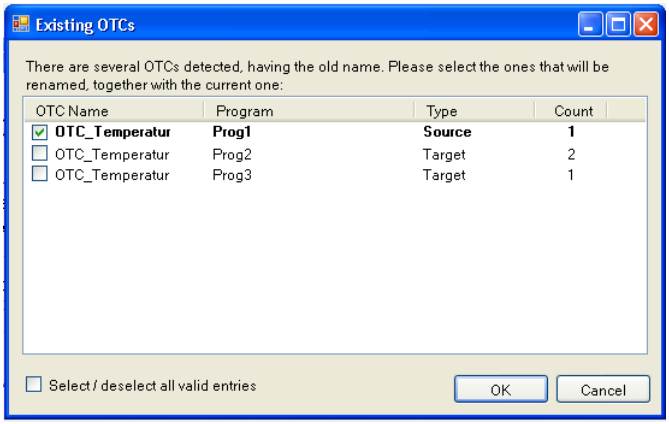
Regeln zur Erstellung von OTCs

- ❑ Ein Ausgangs-OTC muss im Projekt eindeutig sein.
- ❑ Zu einem Ausgangs-OTC können mehrere Eingangs-OTCs angelegt werden. Auch innerhalb eines Programms, aber nicht in dem Programm, in dem der Ausgangs-OTC platziert ist.
- ❑ Ein Eingangs-OTC kann nur eine Datenquelle haben, entweder OPC-beschreibbar oder ein zugehöriger Ausgangs-OTC.
- ❑ Wenn Sie einen Ausgangs-OTC anlegen mit dem Namen eines Eingangs-OTCs, dann ist dieser Eingangs-OTC nicht mehr OPC-beschreibbar.
- ❑ Ein Eingangs-OTC, der keine Datenquelle besitzt, kann als Konstante/Parameter verwendet werden.

8.8.3.2 OTC umbenennen

Vorgehen

1. Wählen Sie den OTC an, der umbenannt werden soll.
2. Öffnen Sie die Eigenschaften des OTCs mit Hilfe des Kontextmenüs oder durch doppelklicken auf den OTC.
3. Ändern Sie den Namen des OTCs und verlassen Sie den Eigenschaftendialog mit <OK>
   Wenn der OTC bereits verbundene Ziele aufweist, wird der Dialog "Vorhandene OTCs" angezeigt.
   In dieser Maske kann bestimmt werden, ob alle oder einzelne verbundene OTCs umbenannt werden sollen. Damit kann z. B. ein einzelner falsch benannter OTC durch Korrektur des Namens wieder richtig zugeordnet werden.



Nach dem Verlassen des Dialogs mit <OK> werden alle selektierten OTCs umbenannt.

Anmerkung

Wenn ein Ziel in einem Programm mehrfach vorhanden ist, dann ist die Anzahl über die Spalte COUNT erkennbar.

8.8.3.3 OTCs verfolgen

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf einen OTC.
2. Wählen Sie im Kontextmenü „Gehe zu ->".
   Sie sehen dann, in welchen Programmen der OTC erzeugt und verwendet wird.
3. Klicken Sie eine angezeigte Verbindung an.

Ergebnis

Es wird die entsprechende Programmseite geladen und der OTC markiert.

Kontextmenü	Erklärung
01. -> OTC_Temperatur: Prog1	Erzeuger
02. OTC_Temperatur -> : Prog3	Verbraucher

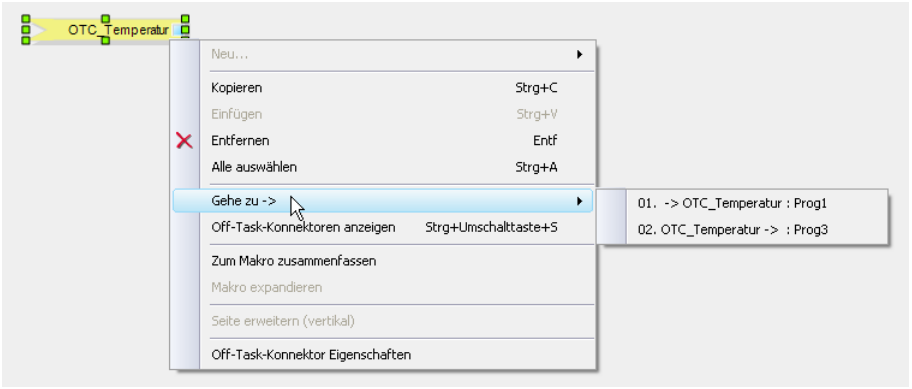


Abbildung 88: OTCs verfolgen

8.8.3.4 Liste aller OTCs

Anzeige aller definierten OTCs im Projekt.

Vorgehen

1. Platzieren Sie den Mauszeiger an einer freien Stelle im Programmierfeld.
2. Öffnen Sie das Kontextmenü.
3. Wählen Sie „Off-Task-Konnektoren anzeigen".
   Der Dialog, der die alphabetisch sortierte Liste aller definierten OTCs enthält, wird angezeigt.
   Navigieren Sie innerhalb der Liste durch Eingabe des Anfangsbuchstaben oder durch Doppelklick auf ein OTC.
   Es wird eine Liste aller definierten OTCs im Projekt gezeigt.

Anmerkung

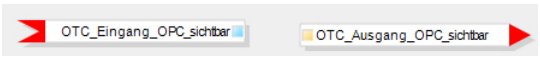
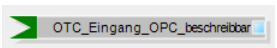
Sie erreichen die Funktion auch über den Menüpunkt „Funktionsplan - Off-Task-Konnektoren anzeigen" oder die Tastenkombination (DE) <STRG>+<UMSCHALT>+<S>.

Hinweis

Sie können den Dialog geöffnet lassen und im Programm weiterarbeiten. Die Liste wird automatisch beim Erstellen oder Entfernen von OTCs aktualisiert. Der Dialog ist frei positionierbar und kann auch an den Rand des Programmierfensters angedockt werden.

8.8.3.5 Darstellung

Zur besseren Orientierung sind die verschiedenen Eigenschaften farblich gekennzeichnet:

Darstellung der OPCs	Beschreibung
	Zeigt verbundene Ein- und Ausgänge
	Zeigt OPC-sichtbare, -lesbare Ein- und Ausgänge
	Zeigt einen OPC-lesbaren und -beschreibbaren Eingang
	Zeigt nicht verbundene Ein- und Ausgänge

8.9 Konvertierer, Splitter, Joiner

8.9.1 Konvertierer

Automatisches Einfügen der Datentyp-Konvertierung

In herkömmlichen CFC-Editoren können Sie keine Anschlüsse verbinden, die nicht denselben Datentyp haben. ibaLogic fügt, wenn eine sinnvolle Konvertierung möglich ist, automatisch einen Konverter ein.

Hinweis

Um größere Konverter zu verwenden, die die Konvertierung direkt darstellen, deaktivieren Sie in den Optionen die Funktion "Symbolische Anzeige der Konverter".

siehe: "Extras" - "Optionen" - [Editoren] - [Diagramm]

Dieser automatische Konverter wird verkleinert dargestellt, um im Programmierfeld Platz zu sparen.
Wenn Sie mit dem Mauszeiger darüber gehen, wird als Tooltip die darunter verborgene Konvertierung angezeigt.

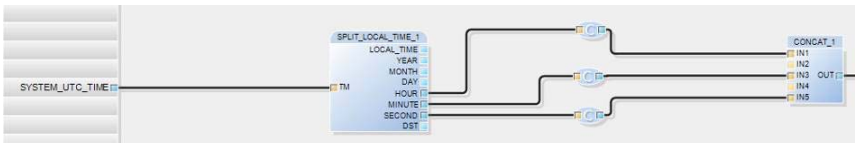


Abbildung 89: Datentyp-Konvertierung

Hinweis

Bausteine mit untypisierten Anschlüssen erhalten erst beim Anschluss eines Konnektors einen Datentyp. Dieser Datentyp wird dann für alle Anschlüsse übernommen und bleibt auch erhalten, wenn Sie die letzte Verbindung wieder abziehen.

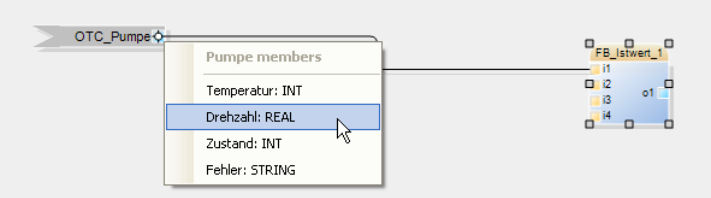
Wenn Sie zwei untypisierte Anschlüsse verbinden wollen, öffnet sich ein Dialog, in dem Sie einen zulässigen Datentyp selektieren können.

8.9.2 Splitter

Sie wollen Elemente für weitere Berechnungen aus einer Datenstruktur herausnehmen. Auf herkömmliche Weise müssen Sie einen Anwenderbaustein erstellen, in dem Sie unter Structured Text einzelne Elemente der Struktur den Ausgangskonnektoren zuweisen. ibaLogic erstellt für Sie automatisch diesen Baustein, genannt Splitter.

Vorgehen

1. Ziehen Sie eine Verbindungslinie zwischen einem Baustein-Eingangskonnektor zu einem Struktur-Ausgangskonnektor eines Bausteins oder eines OTC-Eingangs. Eine Auswahlbox wird eingeblendet.
2. Selektieren Sie ein Strukturelement.



Anmerkung

Damit können Sie dann auf die weiteren Strukturelemente zugreifen.

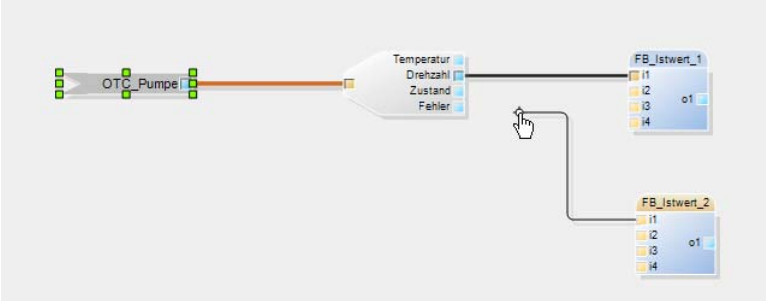


Abbildung 90: Splitter

8.9.3 Joiner

Wenn Sie versuchen einen Ausgangskonnektor mit einem Struktur-Eingangskonnektor zu verbinden, dann bietet ibaLogic ein Menü an, in dem Sie eines der Strukturelemente selektieren können. Daraufhin fügt ibaLogic einen Joiner-Baustein ein, an dessen Eingängen Sie auch weitere Signale anschließen können.

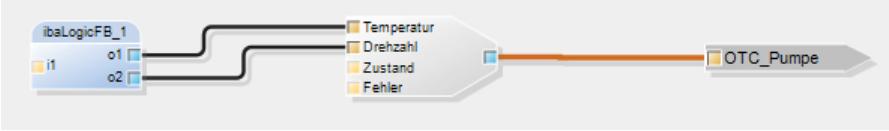
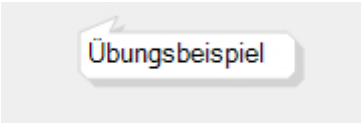


Abbildung 91: Joiner

8.10 Kommentare

Kommentare sind grafische Elemente, die Sie an jeder freien Stelle des Programmfeldes einfügen können. Verbindungslinien können überdeckt werden. Diese sind durch das transparente Kommentarfeld hindurch sichtbar.

Das Kommentarfeld verfügt über einen Zeiger, der auf die zu beschreibende Funktion angedockt werden kann.



Vorgang

1. Platzieren Sie den Mauszeiger an einer freien Stelle im Programmierfeld.
2. Öffnen Sie das Kontextmenü.
3. Wählen Sie „Neu… - Neuer Kommentar".

9 Laufzeitsystem PMAC

9.1 Überblick über Online- und Offline-Modus

Für die Bedienung des Laufzeitsystems stehen folgende Menüs bzw. Symbole im Menü zur Verfügung:

❑ Start	(Menü „Berechnung" und Button in der Symbolleiste)
❑ Stop	(Menü „Berechnung" und Button in der Symbolleiste)
❑ Programm auf PMAC speichern	(Menü „Berechnung")
❑ PMAC-Speicher löschen	(Menü „Berechnung")
❑ Trennen	(Button in der Symbolleiste)
❑ Aktualisierung	(Button in der Symbolleiste)

9.2 Laufzeitsystem starten

Im Gegensatz zu herkömmlichen Automatisierungssystemen erfolgen bei ibaLogic die Schritte „Kompilieren" und „Laden" automatisch im Hintergrund.

Vorgehen

1. Klicken Sie auf den Button <Start> in der Symbolleiste.

2. Bestätigen Sie den Dialog "Berechnung starten..." mit "Ja".

Hinweis

Diese Abfrage kann in den Optionen von ibaLogic abgeschaltet werden um im Entwicklungs- und Testumfeld die Berechnung durch einfaches drücken der <Start>-Taste bzw. F5 zu starten.

Um die Abfrage zu deaktivieren öffnen Sie die Optionen mit "Extras"->"Optionen" und aktivieren die Option "Programm: Berechnung starten" unter [Allgemein]->[Meldungen]->[Bestätigungen].

Ergebnis

Folgende Aktionen werden durchgeführt:

- ❑ Das Projekt wird kompiliert.
- ❑ Das Projekt wird in den PMAC übertragen.
- ❑ Der Programmablauf wird gestartet.
- ❑ In der Programmfenster-Symbolleiste wird die Berechnungszeit angezeigt.
- ❑ Alle Value-Pads werden angezeigt.
- ❑ Die Value-Pads im sichtbaren Bereich werden mit aktuellen Werten versorgt.
- ❑ Das Programmierfeld im Client wechselt die Hintergrundfarbe in Pink.
- ❑ Das Programm befindet sich nun im Online-Modus.

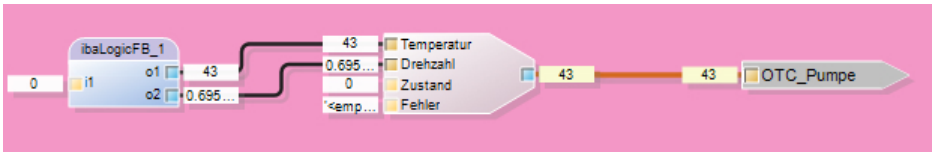


Abbildung 92: Online-Modus

Auftretende Fehler beim Kompilieren, Laden etc. werden im Ereignisfenster angezeigt. Standardmäßig liegt dieses unterhalb des Programmierfeldes. Es kann aber ausblendet oder an beliebiger Stelle positioniert und wieder andockt werden.

Tipp

Ein besonderes ibaLogic-Highlight ist die Tatsache, dass Sie (fast) die komplette Programmierung im Online-Modus durchführen können.

Ausnahmen:

- ❑ Konfigurieren des Zielsystems
- ❑ Konfigurieren der I/Os
- ❑ Importieren von Progammen/Bausteinen/Datentypen
- ❑ Konfiguration des Bausteins DAT_FILE_WRITE

Weitere Besonderheit:

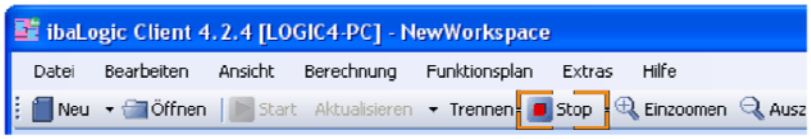
Sie können den Client im Online-Modus beenden, ohne den PMAC anzuhalten. Wenn Sie den Client wieder starten, dann verbindet sich dieser sofort mit dem PMAC im Online-Modus.

Das ist insbesondere dann interessant, wenn der PMAC auf einem anderen Zielsystem (anderer PC oder PADU-S-IT) läuft. Dann können Sie den ibaLogic-Rechner herunterfahren und sogar entfernen, während der PMAC weiterläuft. Nach Reboot und Start von Server und Client verbindet sich der Client automatisch wieder mit dem laufenden PMAC im Online-Modus.

9.3 Laufzeitsystem anhalten

Vorgehen

1. Klicken Sie auf den Button <Stop> in der Symbolleiste.



2. Bestätigen Sie den Dialog "Berechnung anhalten..." mit "Ja".

Hinweis

Diese Abfrage kann in den Optionen von ibaLogic abgeschaltet werden um im Entwicklungs- und Testumfeld die Berechnung durch einfaches drücken der <Stop>-Taste bzw. <Shift>+<F5> zu stoppen.

Um die Abfrage zu deaktivieren öffnen Sie die Optionen mit "Extras"->"Optionen" und aktivieren die Option "Programm: Berechnung anhalten" unter [Allgemein]->[Meldungen]->[Bestätigungen].

Ergebnis

Durch <Stop> werden folgende Aktionen durchgeführt:

- ❑ Der Programmablauf (PMAC) wird gestoppt.
- ❑ Das Programmierfeld im Client wechselt die Hintergrundfarbe zu Grau.
- ❑ Die Value-Pads werden ausgeblendet.
- ❑ Das Programm befindet sich nun im Offline-Modus.

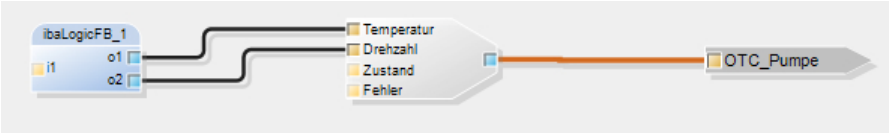


Abbildung 93: Offline-Modus

9.4 Laufzeitsystem – Autostart

9.4.1 Programm auf PMAC speichern

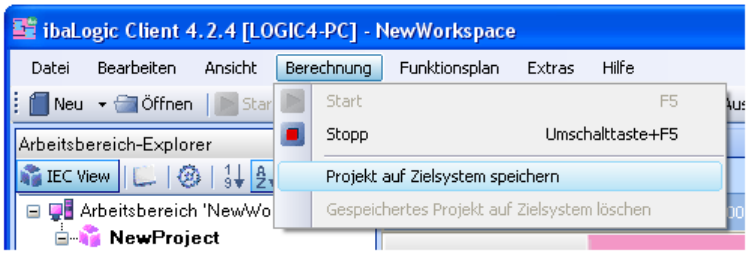
Wenn das Zielsystem mit einem lauffähigen Programm starten soll, dann muss das Programm vorher im PMAC gespeichert sein.

Voraussetzung

- ❑ ibaLogic ist im Online-Modus.
- ❑ Autostart ist aktiviert.

Vorgang

➲ Wählen Sie im Hauptmenü „Berechnung – Projekt auf Zielsystem speichern".



Ergebnis

Das Projekt wird auf dem Zielsystem gespeichert. Es wird eine Datei physikalisch erzeugt. Diese Datei wird der PMAC auf dem Zielsystem beim Hochfahren finden und ausführen.

Weitere Informationen siehe „Autostart Server aktivieren , Seite 45".

Hinweis

Beachten Sie, dass auch jede nachfolgende Programmänderung explizit im PMAC gespeichert werden muss.

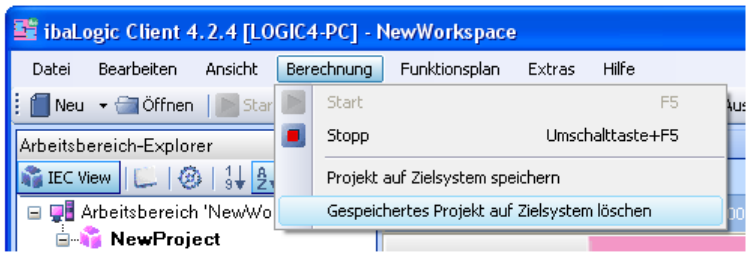
Um einen automatischen Hochlauf des PMAC zu verhindern, können Sie vorher den Speicher des Zielsystems löschen oder die Autostart-Optionen verändern.

9.4.2 Programm auf PMAC löschen

Damit werden die vorher mit dem Befehl „Programm auf PMAC speichern" physikalisch erzeugten Image-Files wieder gelöscht.

Vorgang

➲ Wählen Sie im Menü „Berechnung - PMAC-Speicher löschen".



Ergebnis

Der PMAC-Speicher, d. h. das angelegte Image-File wird gelöscht.

9.5 Verbinden/Trennen

Trennen wird verwendet, wenn - ohne das Programm zu stoppen - mehrere Programmänderungen nacheinander durchgeführt werden sollen, ohne die Schritte einzeln kompilieren zu müssen.

Vorgang

➲ Wählen Sie in der Symbolleiste <Trennen>.

9.5.1 Beispiel

Sie wollen einen Struktur-Datentyp, der im Programm mehrfach verwendet wird, erweitern.

Vorsicht!

Gefahr durch wirkende SWITCH/SLIDER im getrennten Zustand!

Bestehende SWITCH/SLIDER zum Zeitpunkt des Trennens wirken weiterhin auf den laufenden PMAC, um trotzdem noch Einfluss auf die Anlage zu haben. Dieses Verhalten erlaubt es, trotz TRENNEN, den laufenden PMAC zu bedienen.

Wird im Layout einer dieser SWITCH oder SLIDER entfernt und mit gleichen Namen wieder eingefügt, dann wirkt dieser sofort wieder auf den laufenden PMAC.

Wird im getrennten Zustand ein FB oder ein MB gelöscht und anschließend mit gleichen Namen wieder neu eingefügt, dann werden gleichlautende Ein- und Ausgänge sofort wieder visualisiert. Die Werte beziehen sich auf den aktuell laufenden Block im PMAC und nicht auf den neu angelegten Block. Der neu angelegte Block könnte z. B. einen komplett anderen Inhalt haben. Erst mit dem Übersetzen und Laden des Programms wird dann das neue Layout übernommen.

Vorgang

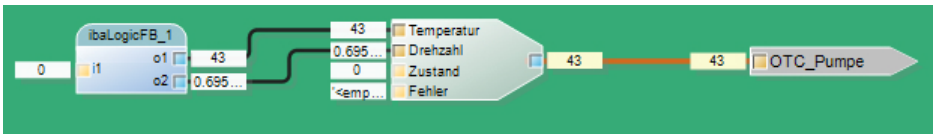
➲ Klicken Sie auf <Trennen>.

➲ Sie ändern den Struktur-Datentyp und legen diesen unter einem anderen Namen ab (weil dieser verwendet wird, kann er nicht unter demselben Namen gespeichert werden).

Danach werden folgende Aktionen durchgeführt:

❑ Der Programmablauf (PMAC) wird nicht gestoppt, sondern läuft unbeeinflusst weiter.

❑ Das Programmfenster im Client wechselt die Hintergrundfarbe zu Grün.



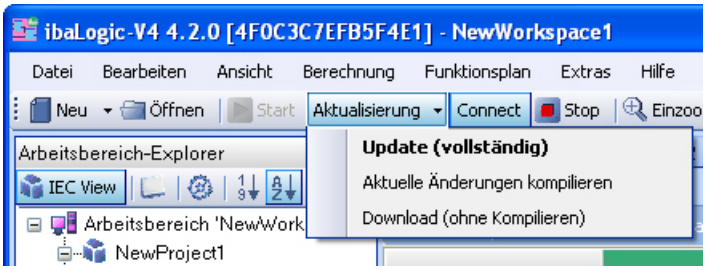
❑ Die Value-Pads werden weiterhin angezeigt und aktualisiert.

❑ Der Button <Trennen> wird geändert in <Verbinden>.

❑ Der Button <Aktualisieren> wird aktiv.

➲ Sie können jetzt der Reihe nach in allen Bausteinen oder OTCs, in denen dieser Datentyp verwendet wird, den alten Typ durch den neuen ersetzen. Die Änderungen werden dabei weder kompiliert noch in den PMAC geladen.

➲ Nach Fertigstellen Ihrer Änderungen können Sie nun gezielt kompilieren und in den PMAC laden. Entweder in einem Schritt (Update vollständig) oder in getrennten Schritten. Wählen Sie in der Symbolleiste <Aktualisieren>.



Dabei wird der Trennen-Modus nicht verlassen.

➲ Den Trennen-Modus verlassen Sie durch Klicken auf <Verbinden>.

Ergebnis

Dadurch werden alle Änderungen kompiliert und in den PMAC geladen.

10 Zielsysteme

Bevor Sie beginnen, die Schnittstellen zur Peripherie oder zu anderen Systemen zu konfigurieren, müssen Sie die Hardwarebasis, auf der das ibaLogic-Laufzeitsystem (PMAC) laufen soll, einstellen.

Zum jetzigen Zeitpunkt stehen für das Zielsystem zwei Geräteklassen zur Verfügung:

WinXP

Der PMAC läuft auf einem Windows-PC, worauf auch die anderen ibaLogic-Komponenten laufen.
Weitere Informationen siehe "Betriebsarten und Verarbeitungsmodi , Seite 30".

Hier wird die Verbindung zur dezentralen Peripherie und zu anderen Systemen durch PCI-Karten hergestellt.

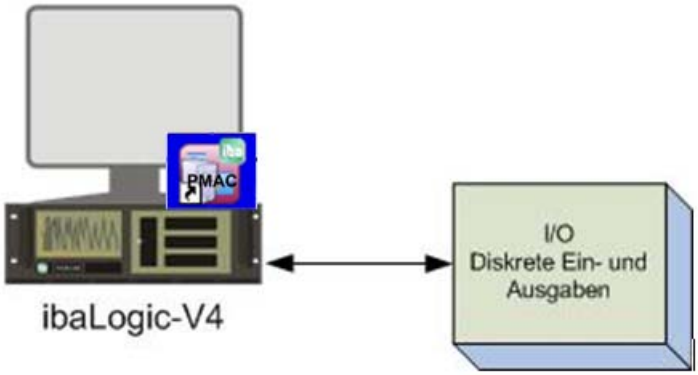


Abbildung 94: Peripherie-Schnittstelle Windows-PC

PADU-S-IT

Der PMAC läuft auf einer ibaPADU-S-IT-Station. Alle anderen ibaLogic-Komponenten liegen immer auf einem oder mehreren Windows-PCs.
Auf dem ibaPADU-S-IT stehen dem PMAC nur die lokalen I/O-Komponenten (Peripheriemodule) zur Verfügung. Für dezentrale Peripherie und Fremdsysteme gibt es einen bidirektionalen LWL-Anschluss sowie einen Netzwerkanschluss für TCP/IP-Kopplung.

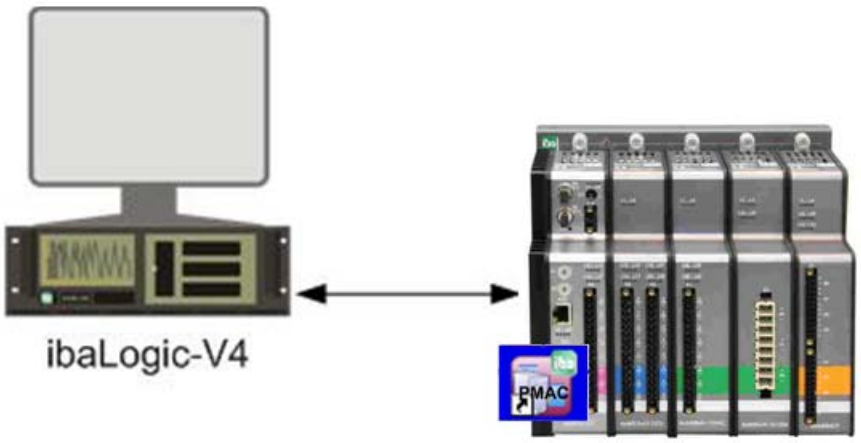


Abbildung 95: Peripherie-Schnittstelle PADU-S-IT

Nach dem ersten Aufruf des ibaLogic-V4-Clients wird der lokale Windows-PC, also Geräteklasse „WinXP" als Zielsystem voreingestellt.

10.1 Zielsystem konfigurieren

Die Dialogbox zur Konfiguration des Zielsystems finden Sie unter „Extras" in der Menüleiste des ibaLogic Clients.

Hinweis

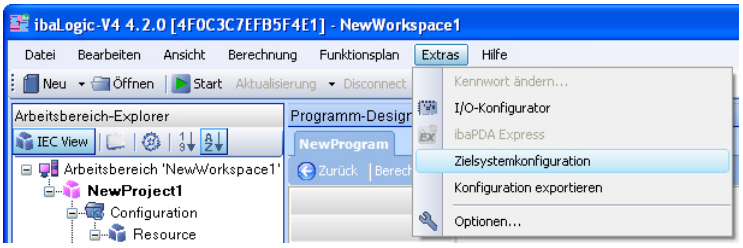
Die Zielsysteme werden projektspezifisch angelegt. Sie finden im Dialog „Zielsysteme konfigurieren" alle Projekte des Arbeitsbereiches und die darin konfigurierten Zielsysteme.

Voraussetzung

Sie haben ein Projekt geöffnet.

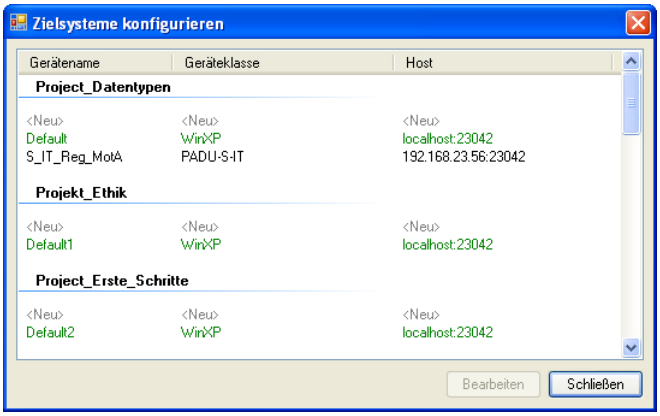
Vorgehen

➲ Wählen Sie im Menü „Extras - Zielsystemkonfiguration" aus.



Alternativ:
Liste neben "Aktuelles Zielsystem" in der Symbolleiste öffnen und den Punkt <Zielsystem hinzufügen> bzw. <Zielsystem bearbeiten> auswählen.

➲ Um ein Zielsystem anzulegen oder zu konfigurieren, klicken Sie im Dialog unter dem Projektnamen auf <Zielsystem hinzufügen> oder auf das betreffende Zielsystem.



Farbschema	Erklärung
Grün	Ist das aktive System.
Hellgrau	Anlegen eines neuen Systems.
Schwarz	Weitere verfügbare Systeme.

➲ Klicken Sie auf den Button <Bearbeiten>. Der Dialog „Zielsystemkonfiguration bearbeiten" wird angezeigt.

➲ Geben Sie einen Gerätenamen ein oder übernehmen Sie die Vorgabe-Einstellungen.
Der Name muss innerhalb des Arbeitsbereichs eindeutig sein.

➲ Wählen Sie die Geräteklasse WinXP oder PADU-S-IT.

➲ Geben Sie die Host/IP ein.

- Wenn Sie die Geräteklasse WinXP eingestellt haben, dann geben Sie, wenn der PMAC auf dem Rechner oder auf dem der Server läuft, „localhost" oder „127.0.0.1" ein.
- Liegt der PMAC auf einem anderen Rechner innerhalb des Netzwerks, dann geben Sie den Hostnamen bzw. die IP-Adresse dieses Rechners ein.
- Haben Sie die Geräteklasse PADU-S-IT eingestellt, dann geben Sie den Hostnamen bzw. die IP-Adresse des ibaPADU-S-IT Gerätes ein, auf dem der PMAC laufen soll.

➲ Bestätigen Sie Ihre Eingaben mit einem Klick auf den Button <OK>.

➲ Verlassen Sie die DialogBox mit <Schließen>.

10.2 Zielsystem auswählen

In der Symbolleiste des ibaLogic-V4-Clients wird das aktuell eingestellte Zielsystem angezeigt. Ein Umschalten auf ein anderes Zielsystem ist über die Auswahlbox möglich.

Wichtiger Hinweis

Beachten Sie, dass das Zielsystem immer für das aktive Projekt und nicht für das gerade bearbeitete eingestellt wird.

Vorgehen

1. Zum Umschalten klicken Sie auf die Auswahlbox.
2. Wählen Sie eines der vorher eingestellten Zielsysteme.

Wichtiger Hinweis

Beachten Sie, dass die I/O-Konfiguration abhängig vom Zielsystem ist.

Sie müssen nach dem Einstellen des Zielsystems die I/O-Konfiguration aktualisieren.

Im Menü „Extras – I/O-Konfigurator" Button <Hardware aktualisieren> auswählen.

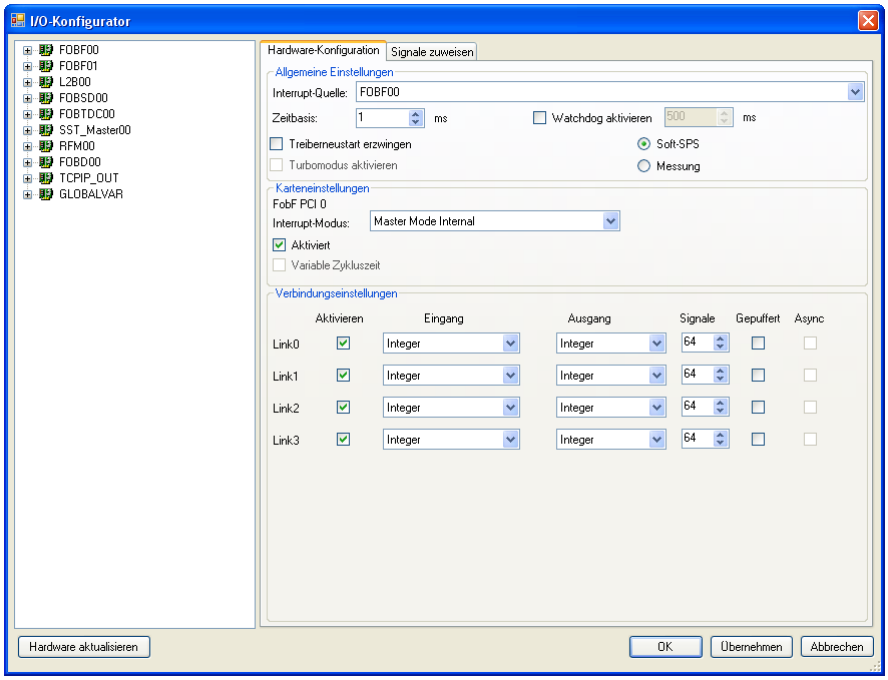
11 I/O-Konfiguration

Der I/O-Konfigurator ist der zentrale Dialog, in dem alle Konfigurationseinstellungen bezüglich der Eingangs- und Ausgangssignale sowie bestimmter Schnittstellen vorgenommen werden.

Hinweis

Ausnahme bilden alle Schnittstellen, die als Funktionsbausteine, z. B. Baustein TCPIP_SENDRECV oder die eingebaute OPC-Schnittstelle, vorliegen.

➲ Öffnen Sie den I/O-Konfigurator über Menü „Extras - I/O-Konfigurator".



Der I/O-Konfigurator-Dialog besteht aus 3 Bereichen:

- ❑ Verfügbare Ein-/Ausgangs-Ressourcen
- ❑ Hardware-Konfiguration
- ❑ Signalzuweisung

Verfügbare Ein-/Ausgangs-Ressourcen

Im linken Teil des Dialogs werden in einer Baumstruktur alle vom System erkannten und unterstützten Hardware- und Softwareschnittstellen angezeigt.

Hardware-Konfiguration

Hier werden spezielle Einstellungen zum Gesamtsystem und zu den einzelnen Karten vorgenommen.

Signale zuweisen

In ibaLogic kann in einem Projekt mit selbstdefinierten, sinnvollen Ein- und Ausgangsnamen gearbeitet werden (virtuelle Ein- und Ausgänge). Diese virtuellen Signale werden über die Signalzuweisung (Rangierung) den physikalischen Ein- und Ausgängen zugewiesen. Die virtuellen Signale sind in Gruppen eingeteilt.

11.1 Ressourcen

Der I/O-Konfigurator übernimmt beim Öffnen die zu dem aktiven Projekt gehörende I/O-Konfiguration.

Hinweis

Die I/O-Konfiguration wird nicht in der Datenbank gespeichert, sondern in zwei XML-Dateien im Pfad „…\ibaLogic v4\Server\HwMappings". Dadurch ist es möglich, das Projekt unabhängig von der verfügbaren Hardware zu bearbeiten.

Die I/O-Konfigurationsdateien werden beim Datenbank-Backup in ZIP-Dateien gespeichert und beim DB-Restore wieder geladen, so dass nach dem Transport von Projekten auch die originale I/O-Konfiguration zur Verfügung steht.

Beim Speichern des Datenbank-Backups in BAK-Dateien wird die I/O-Konfiguration nicht mit gesichert.

Button <Hardware aktualisieren>:

Durch Klicken auf den Button <Hardware aktualisieren>, wird die Hardware übernommen, die dem Rechner, auf dem der PMAC läuft, zur Verfügung steht. Das sind bei Zielsystem Windows-PC die unterstützten PCI-Karten (siehe „Hardware-Ressourcen , Seite 179"). Beim Zielsystem ibaPADU-S-IT die dort zur Verfügung stehenden Peripherie-Baugruppen.

Wichtiger Hinweis

Das Zielsystem muss vor der Bearbeitung der I/O-Konfiguration eingestellt werden, da bei einem Plattformwechsel die bisherigen Einstellungen des I/O-Konfigurators verloren gehen!

Hinweis

Die Anzahl erlaubter I/O-Signale ist abhängig von der erworbenen Lizenz (Dongle).

Baumstruktur

Voraussetzung

❑ Sie haben die entsprechende Verbindung in der Hardware-Konfiguration aktiviert und mit dem Button <Übernehmen> am unteren Rand der Dialogbox die Konfiguration übernommen.

Vorgehen

➲ Durch Klicken auf das +/-Zeichen vor dem Namen, können Sie die Baumstruktur bis zum einzelnen Signal öffnen.

Es sind folgende Hierarchieebenen dargestellt:

Schnittstelle → Module → Ein-/Ausgänge → Signale

Schnittstelle:	Namen und Index für die Kartentypen
Module:	Gruppe entsprechend der physikalischen Einteilung, abhängig vom Kartentyp.
Ein-/Ausgänge:	Module können nur Eingänge, nur Ausgänge oder beides haben
Signale:	Die Namen der Hardware-Signale werden gebildet aus Modulnamen, Richtung, Datentyp und laufender Nummer.

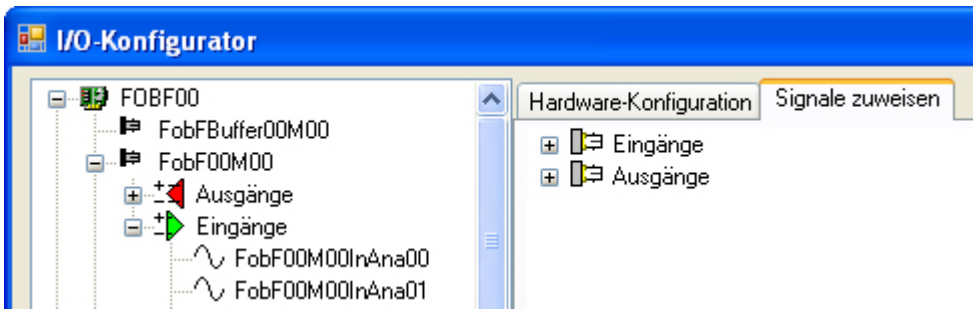


Abbildung 96: Signale zuweisen

11.1.1 Hardware-Ressourcen

ibaLogic unterstützt folgende Schnittstellen:

Zielsystem WinXP und PCI-Karten

Schnittstelle	Karten	Verbindungen	Protokoll
FOBF*nn*[4]	ibaFOB-io-S ibaFOB-4i-S ibaFOB-4o-S	1 Link LWL 4 Links LWL (simplex/duplex)	ibaNet mit 2 und 3.3 MBit
	ibaFOB-2io-X ibaFOB-4i-X ibaFOB-4o-X	2 Links LWL 4 Links LWL (simplex/duplex)	ibaNet mit 3.3 und 32 MBit (32 MBit nur Empfang)
FOBD*ii*	ibaFOB-2io-D ibaFOB-4io-D ibaFOB-4o-D	2 Links LWL 4 Links LWL (simplex/duplex)	ibaNet mit 2, 3.3 und 32 MBit und DMA
FOBSD*nn*	ibaFOB-SD	1 Link LWL duplex (ST)	SIMADYN D
FOBTDC*nn*	ibaFOB-TDC	1 Link LWL 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-5575	1 LWL duplex 1 Koax	Reflective Memory

Abbildung 97: Zielsystem WinXP

nn = Kartennummerierung 00 bis 03 (max. 4 Karten eines Typs sind erlaubt)

ii = Kartennummerierung 00 bis 07 (max. 8 Karten vom Typ FOB-D sind erlaubt).

Zielsystem PADU-S-IT

Schnittstelle	Peripherie
PADU-S-IT	lokale Peripherie, d. h. die Schnittstellenmodule am PADU-S Baugruppenträger. Sehen Sie dazu die PADU-S-Handbücher.

Andere Dokumentation - Zielsystem PADU-S-IT

Informieren Sie sich dazu im ibaPADU-S-IT-Handbuch.

[4] Karten dieser Schnittstelle sind nur noch für Altanlagen erhältlich.

11.1.2 Software-Ressourcen

ibaLogic unterstützt folgende auf Ethernet basierende Protokolle:

Anzeige	Protokoll
TCPIP_OUT	TCP/IP-Protokoll ibaLogic zu PDA: WinXP-Klasse: max. 16 Telegramme je 32 Real-Werte und 32 Binärwerte PADU-S-IT-Klasse: max. 16 Telegramme je 32 Real-Werte und 32 Binärwerte

Hinweis

Es stehen nur Ausgänge zur Verfügung.

11.1.3 Globale Systemvariablen

Es werden folgende globale Systemvariablen zur Verfügung gestellt:

Schnittstelle	Variablen
GLOBALVAR	Globale Eingangsvariablen: LAST_DRIVER_ERROR Letzter aufgetretener Fehler im Treiber als Hexcode. WATCHDOG_BITE Standardmäßig auf FALSE. Wenn dieser Wert TRUE wird hat der eingestellte Watchdog von ibaLogic angesprochen und die Ausgänge der Karten abgeschaltet. Weitere Informationen zu "Watchdog" siehe "Allgemeine Einstellungen , Seite 182" SYSTEM_UTC_TIME aktuelle Systemzeit im UTC-Format (UniversalTimeCoordinated). DONGLE_NUMBER Dongle-Nummer des gesteckten iba-Dongles am PC bzw. Seriennummer des PADU-S-IT. ACQ_RESTART_COUNT Zähler für internen Treiberneustart. Dieser dient zum Erkennen von Überlastungen im „Buffered Modus"

11.2 Hardware-Konfiguration

Der Dialog zur Hardware-Konfiguration wird über die Registerkarte „Hardware-Konfiguration" aufgerufen.

In der Hardware-Konfiguration sind drei Abschnitte vorhanden, in denen folgende Einstellungen vorgenommen werden können.

- ❑ Allgemeine ibaLogic-Einstellungen
- ❑ Karteneinstellungen
- ❑ Verbindungseinstellungen

Hinweis

Änderungen können nur vorgenommen werden, wenn die Berechnung nicht gestartet ist (grauer Hintergrund im Programmierbereich).

11.2.1 Allgemeine Einstellungen

Die allgemeinen Einstellungen sind für das ibaLogic-Laufzeitsystem und für alle Schnittstellenkarten gültig.

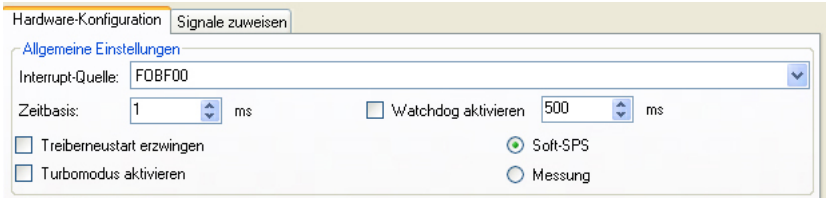


Abbildung 98: Allgemeine Einstellungen

Interrupt-Quelle

In diesem Listenfeld bzw. in dieser Auswahlliste werden die zur Verfügung stehenden iba-Baugruppen angezeigt. Wählen Sie aus dieser Liste eine beliebige Baugruppe aus, die als Interrupt-Quelle bzgl. des PCI-Busses arbeiten soll.

Befindet sich im System keine I/O-Karte, dann wird von ibaLogic ein interner Taktgeber verwendet und das Feld ist leer.

Unter Plattform PADU-S-IT wird nur PADU-S-IT angeboten.

Zeitbasis

Die Zeitbasis ist die kleinste verwendbare Zykluszeit. Beachten Sie, dass die einstellbaren Taskintervalle nicht kleiner als diese Zeitbasis sein können.

Turbomodus aktivieren

Wenn ein Mehrprozessor-PC verwendet wird, dann kann der Turbomodus aktiviert werden. Die Leistungsreserven des Systems können damit deutlich erhöht werden, da einer der Prozessoren ausschließlich für die Abarbeitung des PMAC zuständig ist und der andere die übliche Windows-Verwaltung übernimmt. Insbesondere bei Steuerungs- und Regelungsaufgaben (Software-SPS) sollte davon Gebrauch gemacht werden.
Detaillierte Beschreibung siehe „Zeitverhalten , Seite 239".

Soft-SPS

Dieser Modus ist für Regelungs- und Steuerungsaufgaben geeignet. Dieser stellt in ibaLogic sicher, dass nur die jüngsten Signalzustände verarbeitet werden. Im Gegensatz zum Messungsmodus ist es nicht entscheidend, ob Samples verloren gehen. Verwendet werden aktuelle Daten aus dem letzten I/O-Transferzyklus für die Berechnungen.

Detaillierte Beschreibung siehe „Zeitverhalten , Seite 239".

Messung

Dieser Modus stellt sicher, dass ibaLogic kein Eingangssample verliert. Dies gilt auch dann, falls einzelne Tasks innerhalb von ibaLogic verdrängt werden sollten. Das Ablaufsystem von ibaLogic stellt sicher, dass die Daten äquidistant im eingestellten Taskintervall zur Verfügung stehen. Bei Task-Verdrängungen werden Zyklen nachgeholt.

Detaillierte Beschreibung siehe „Zeitverhalten , Seite 239".

Watchdog aktivieren

Wenn diese Funktion aktiviert ist, dann wird in den vorhandenen Karten ein Timer aufgezogen, der durch das Beschreiben der Ausgänge von ibaLogic getriggert wird. Wenn innerhalb der eingestellten Zeit kein Schreibbefehl (= Trigger) von ibaLogic eintrifft, dann setzt die Karte selbstständig alle Ausgänge auf 0.

Da ibaLogic zyklisch die Ausgänge beschreibt, deutet ein Ansprechen der Watchdog-Funktion auf eine stehende oder überlastete Applikation hin (z. B. durch programmierte Endlosschleife).

Treiberneustart erzwingen

Diese Funktion ist speziell für SST-Karten und Reflektive Memory-Karten wichtig. Mit dem Treiberneustart werden diese Karten auch zurückgesetzt und damit die extern geänderte Konfiguration übernommen.
Ein Setzen dieser Option führt mit <OK> den Neustart durch. Diese Option wird automatisch wieder zurückgesetzt.

11.2.2 Karteneinstellungen

Für die Einstellungen markieren Sie bitte das entsprechende Interface in der linken Baumstruktur. Hier werden nur die Einstellungen beschrieben, die für (fast) alle Karten gültig sind.

❑ Interrupt-Modus

Dieses Feld wird nur für iba-Karten angeboten.

Zur Auswahl stehen folgende Modi:

❑ Master Mode Internal:
Der Master Mode Internal darf nur für eine Karte eingestellt werden. Diese erzeugt mit einem internen Timer ein Synchronsignal, das über ein Flachbandkabel an die anderen Karten verteilt wird.

❑ Master Mode External:
Im Unterschied zu Master Mode Internal wird das Synchronsignal nicht in der Karte erzeugt, sondern vom Zyklus des auf Link0 eingehenden LWL-Telegramms abgeleitet.
Sinnvoll ist dieser Modus nur, wenn ibaLogic mit einem externen Zyklus synchronisiert werden muss, z. B. bei variabler Interrupt-Zeit im gepufferten Modus (z. B. Planheitsmessung oder für die bussynchrone Messung mit dem Simolink-Monitor. Weitere Informationen siehe „Buffered Mode , Seite 195".

❑ Slave Mode:
Dieser Modus ist für alle anderen Karten einzustellen.

Hinweis

Befindet sich in Ihrer Konfiguration eine Karte vom Typ FOBSD oder FOBTDC, dann wählen Sie diese als Interrupt-Master. Andernfalls wählen Sie aus dem Typ FOB-X- oder FOB-D-Karte. Wenn weder das eine noch das andere vorhanden ist, können Sie eine FOB-S- oder eine L2B-Karte nehmen.

❑ Aktiviert

Nur aktivierte Karten können verwendet werden.

Eine Karte muss deaktiviert werden, wenn auf demselben Rechner ein ibaPDA-Server läuft, der diese Karte verwendet. Eine Karte kann nicht von ibaLogic und ibaPDA gleichzeitig verwendet werden.

Weitere Einstellungen sind kartenspezifisch und in „PCI-Schnittstellen (Windows-PC) , Seite 193" beschrieben.

11.3 Signalzuweisung

Um die physikalischen Ein- und Ausgänge verwenden zu können, müssen diese den virtuellen Signalen im Programm zugewiesen werden.

Es gibt zwei Verfahren für die Zuweisung von Signalen:

- ❑ vom Hardware-Signal zum Programm-Signal (aus Sicht der Hardware).
- ❑ vom Programm-Signal zum Hardware-Signal (aus Sicht des Programms).

Zwischen diesen grundlegenden Verfahren, ist auch ein gemischtes Vorgehen möglich.

11.3.1 Vorgehensweise aus Sicht der Hardware

Zu Beginn der Programmierung sind Ihnen die Schnittstellen nach außen, z. B. die Belegung der LWL-Telegramme oder die der Profibus Slaves schon bekannt. Aus diesen physikalischen Signalen erzeugen Sie virtuelle Signale, die Sie später im Programm verwenden können.

Sie können die automatisch generierten Namen übernehmen oder eigene Basisnamen vergeben. Richtungshinweis, Typ und Index werden automatisch hinzugefügt. Jeder Signalname kann anschließend einzeln geändert werden.

Nach Übernahme sind die virtuellen Signale im Navigationsbereich unter „Eingänge - Ausgänge" sichtbar. Auch dort können die Signalnamen geändert werden, solange die Signale noch nicht im Programm verwendet werden, d. h. aus dem Navigationsbereich auf die Randleisten des Programmierbereiches gezogen wurden.

11.3.1.1 Beispiel: Zuordnung aller Signale eines Moduls einer ibaFOB-io-S-Karte

Auf der linken Seite ist u. a. die FOB-Karte aufgelistet.

Im Register „Signale zuweisen" sind die Eingänge und Ausgänge noch nicht belegt.

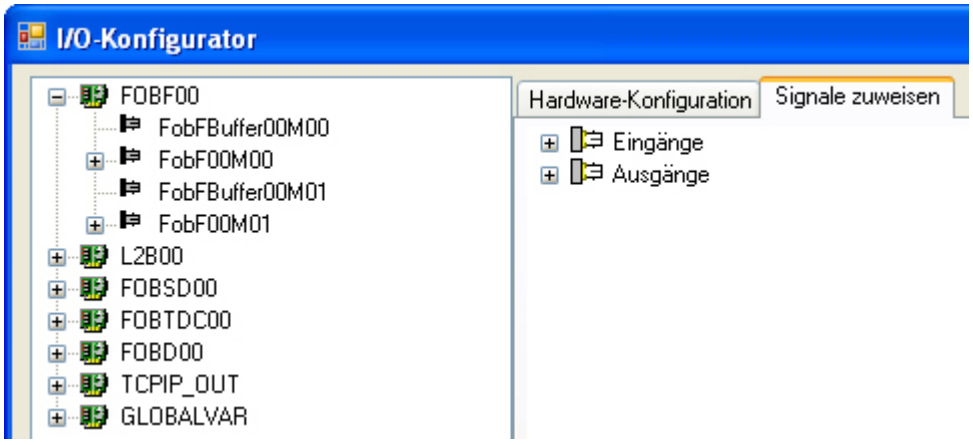


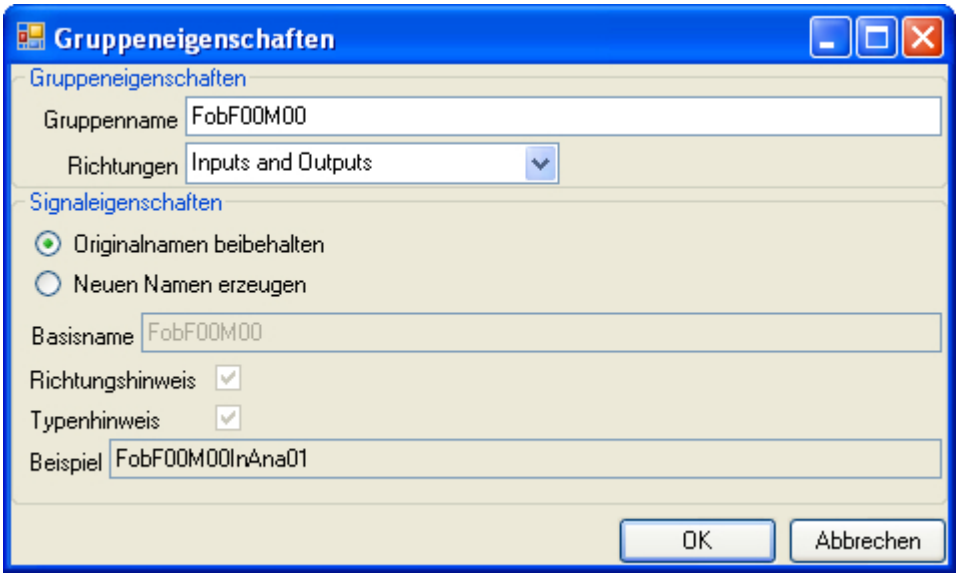
Abbildung 99: Zuordnung von Signalen der ibaFOB-io-S-Karte

Vorgehen

1. Öffnen Sie den Baum der FOB-Karte auf der linken Seite.
   Alle aktivierten Links der Karte und deren Bezeichnung werden angezeigt.

   Das gesamte Modul FOBF00M00 (entspricht dem Link 0 der FOB-Karte) kann in einem Schritt zugewiesen werden.

2. Ziehen Sie das Modul FOBF00M00 per Drag & Drop auf die rechte Seite auf die Ein- oder Ausgänge.
   Der Dialog „Gruppeneigenschaften" wird angezeigt.



Anmerkung

ibaLogic legt unter den Eingängen und Ausgängen eine Gruppe mit dem Gruppennamen an und bildet einen virtuellen Namen für jedes Hardwaresignal. Die Generierung der Namen können Sie komplett ibaLogic überlassen oder selbst festlegen.
Während der Festlegung wird die Zusammensetzung der Signalnamen aktuell angezeigt. Im Beispiel oben ist der Name „FOBF00M00InAna01" so zusammengesetzt:

FOBF00M00	Basisnamen (gleich dem Gruppennamen)
In	Richtungshinweis
Ana	Typenhinweis
01	laufende Nummer

Je nachdem, welche Richtungen Sie in dem Auswahlfeld gewählt haben, werden alle Signale dieses Moduls in der angezeigten Gruppe unter Eingänge und/oder Ausgänge angelegt.

Ergebnis

Die Zuweisung führt zu folgendem Ergebnis:

Links vom Pfeil werden die virtuellen Signalnamen, rechts die Hardware-Signalnamen angezeigt. Der Gruppenname ist mit dem physikalischen Modulnamen identisch.

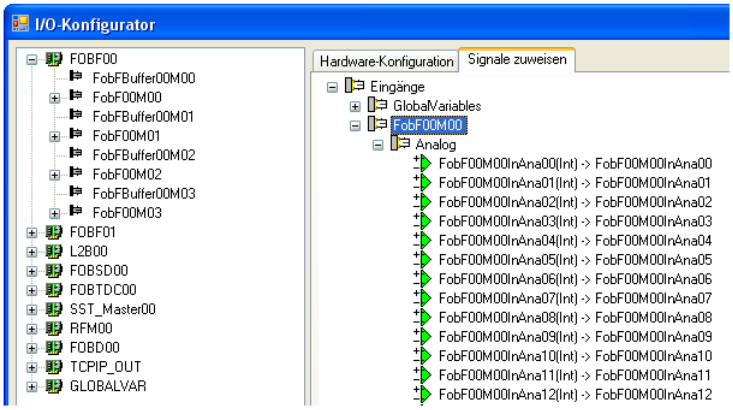


Abbildung 100: Signale zuweisen

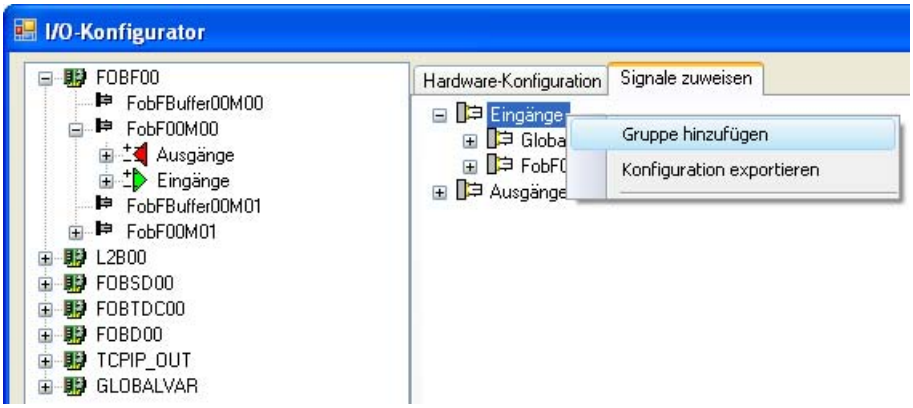
11.3.1.2 Beispiel: Zuordnung einzelner Signale einer ibaFOB-4i-S- oder ibaFOB-4o-S-Karte

Wenn Sie nur wenige Signale eines Moduls im Programm benötigen, können Sie gezielt die Zuweisung einzelner Signale vornehmen.

Vorgehen

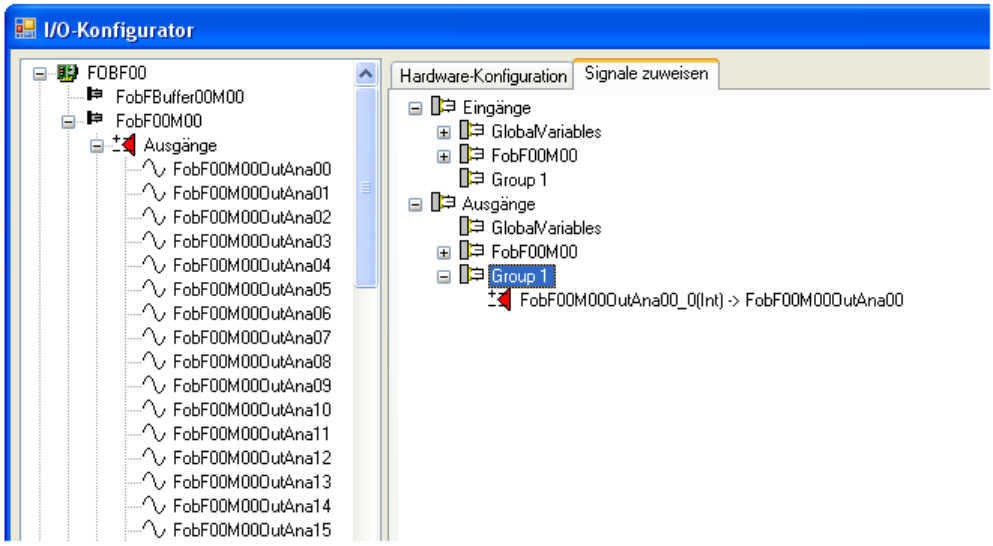
Gruppe hinzufügen

1. Öffnen Sie das Kontextmenü mit rechtem Mausklick.



2. Wählen Sie Menü „Gruppe hinzufügen", vergeben Sie einen Gruppennamen und verlassen Sie den Dialog "Gruppenname einstellen" mit OK.

3. Öffnen Sie den Signalbaum solange bis Sie die einzelnen Signale des Moduls sehen die Sie verwenden möchten.
4. Fügen Sie einzelne Signale per Drag & Drop der vorher definierten Gruppe hinzu.



Ergebnis

Einzelne Signale werden der Gruppe zugeordnet.

11.3.1.3 Signal- und Gruppennamen ändern

Sie können nach der Zuweisung die Namen der virtuellen Signale einzeln ändern.

Vorgehen

1. Den Gruppennamen ändern Sie durch Klicken der rechten Maustaste. Wählen Sie Menüpunkt „Eigenschaften".
2. Den Signalnamen ändern Sie durch Doppelklick auf diesen. Sie können auch eine Signalbeschreibung anfügen. Die Beschreibung wird im Programm als Tooltip angezeigt.
3. Bestätigen Sie die Einstellungen mit <Übernehmen> oder mit <OK>.

11.3.2 Vorgehensweise aus Sicht des Programms

Zu Beginn der Programmierung sind die Schnittstellen nach außen noch nicht bekannt, Sie wollen mit der Programmierung beginnen und wissen, welche Ein- und Ausgänge Sie benötigen.

1. Definieren Sie in dem Navigationsbereich der Ein- und Ausgänge die Gruppen und Signale. Sehen Sie dazu die Beschreibung in „Ein- und Ausgänge projektieren , Seite 80".
2. Sobald Sie wissen auf welchen physikalischen Schnittstellen die I/O-Signale liegen, können Sie in den I/O-Konfigurator wechseln und dort die Zuordnung zu den physikalischen Schnittstellen vornehmen.

11.3.2.1 Beispiel: Signale einer ibaFOB-4io-S-Karte (ganzes Modul)

Sie weisen physikalische Signale von Link0 der FOB-Karte zu.

Voraussetzung

Sie haben im Programm (im Navigationsbereich der Ein- und Ausgänge) eine Gruppe „MotorA" und darunter die Eingangssignale „MotorA_N_Ist(Int)", „MotorA_Ta(Int)" und „MotorA_Status(Int)" definiert.

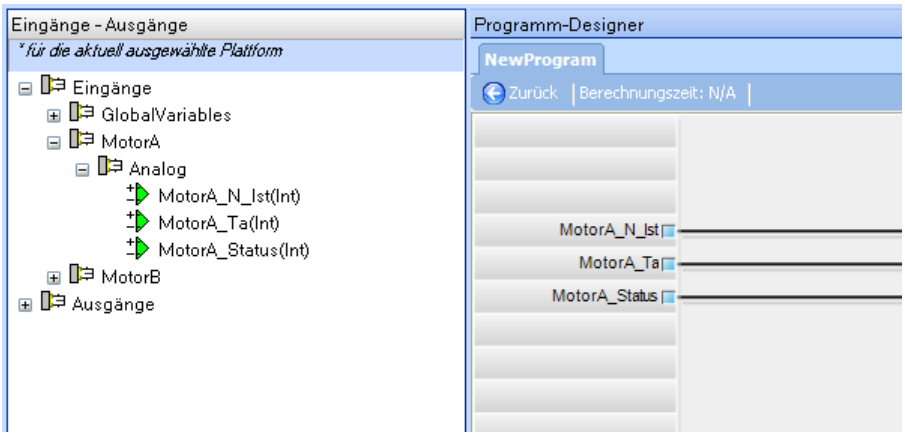
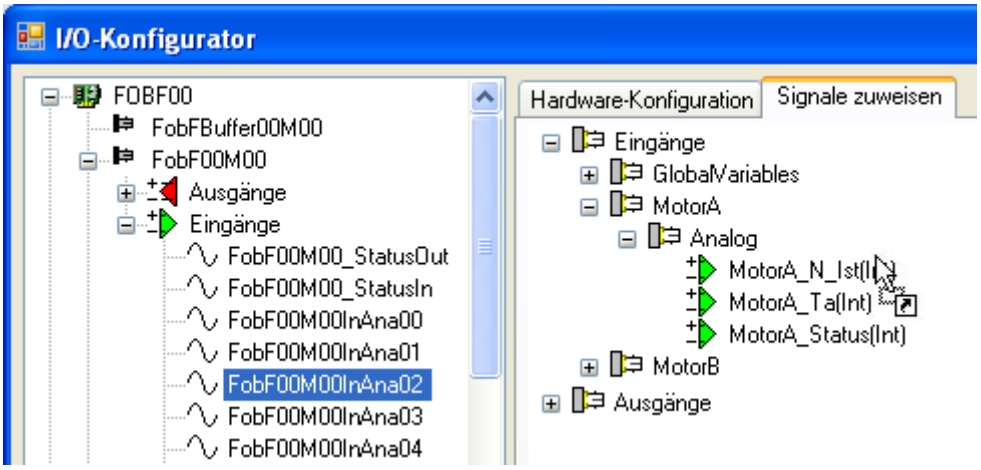


Abbildung 101: „Eingänge – Ausgänge"

Vorgehen

1. Öffnen Sie den I/O-Konfigurator.
2. Aktualisieren Sie die Hardware mit dem Button <Hardware aktualisieren>.
3. Aktivieren Sie den Link0 der FOB-Karte.
   Achten Sie darauf, dass der Datentyp des Links mit dem der bereits definierten Signale übereinstimmt.
   In der Lasche „Signale zuweisen" sehen Sie die definierten Signale im Projekt.
4. Wählen Sie aus der linken Baumstruktur ein Hardware-Signal aus.
   Ziehen Sie dieses per Drag & Drop auf das Signal.
   Damit haben Sie das virtuelle Signal auf ein physikalisches Signal rangiert.



Ergebnis

Die Zuordnung ist auch im Programmierbereich zu sehen.

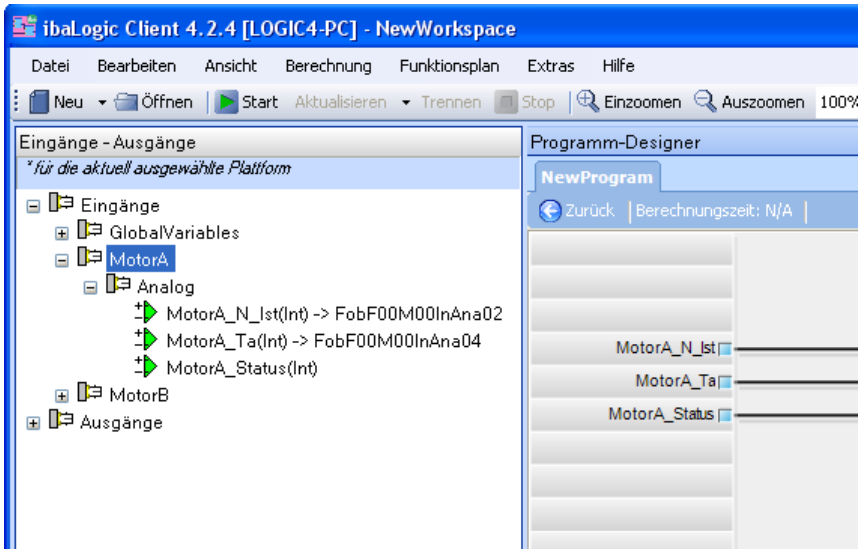


Abbildung 102: Signale zuweisen

11.3.3 Signalzuweisung ändern

Sie können bereits vorhandene Rangierungen ändern. Mehrere Vorgehensweisen sind möglich.

Vorgehen 1

➲ Ziehen Sie per Drag & Drop einfach ein anderes Hardwaresignal auf ein bereits verbundenes virtuelles Signal.
Damit ist die Rangierung geändert.

Vorgehen 2

➲ Ziehen Sie per Drag & Drop ein bereits verwendetes Hardwaresignal auf ein anderes virtuelles Signal.
Es wird eine Hinweismeldung aufgeblendet.

➲ Quittieren Sie die Hinweismeldung, die Rangierung wird geändert und die Verbindung zum alten Signal gelöst.

Hinweis

Sie können jedem Hardwaresignal nur ein virtuelles Signal zuweisen.

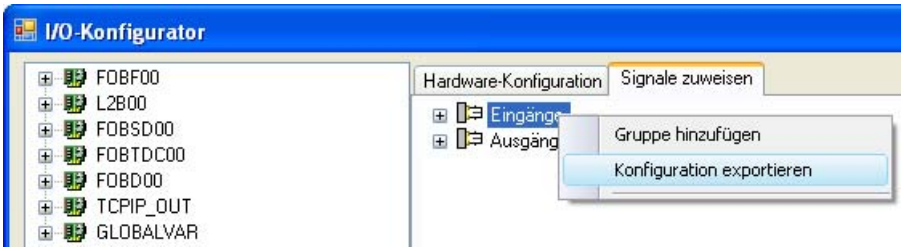
11.3.4 Verwenden von extern definierten Signalnamen

Mit der Export/Import-Funktion können Sie auch Signalnamen verwenden, die in externen Dokumenten, z. B. einem Tabellenkalkulationsprogramm zur Verfügung stehen.

Vorgehen

➲ Legen Sie die Hardware-Signale fest.

➲ Exportieren Sie die I/O-Konfiguration mit Hilfe des Kontextmenüs der Eingänge oder Ausgänge.



➲ Wählen Sie im Menü „Konfiguration exportieren". Eine Dialogbox wird angezeigt.

➲ Geben Sie Zielpfad und Dateiname an.

Ergebnis

Sie erhalten eine CSV-Datei, die Sie mit einem ASCII-Editor oder einem Tabellenkalkulationsprogramm öffnen können.

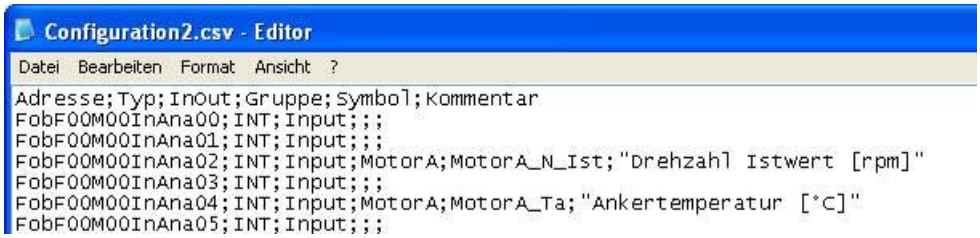


Abbildung 103: CSV-Datei im ASCII-Editor

Wichtiger Hinweis

In den Spalten Adresse, Typ und InOut sind die definierten Hardware-Signale zu sehen.

Ändern Sie diese nicht.

Unter den Spalten Gruppe, Symbol und Kommentar können Sie die virtuellen Signale editieren oder per Copy & Paste von dem externen Dokument übernehmen.
Achten Sie darauf, dass die Namen in Spalte „Symbol" der IEC-Norm entsprechen.

➲ Zum Importieren verlassen Sie den I/O-Konfigurator.

➲ Wählen Sie aus dem Hauptmenü „Datei - Import - Signalzuordnung".

11.4 PCI-Schnittstellen (Windows-PC)

In diesem Kapitel werden die speziellen Einstellungen der Schnittstellenkarten beschrieben.

Weitere Hinweise entnehmen Sie „Hardware-Ressourcen , Seite 179".

11.4.1 Verbindung zur „iba-Welt"

Für die LWL-Verbindung zur dezentralen iba-Peripherie (PADUs) und zu den iba-System-Schnittstellenbaugruppen gibt es folgende PCI-Karten. Diese finden Sie im Ressourcenbaum unter dem Schnittstellentyp **FOBFnn** oder **FOBDnn**.

- ibaFOB-S[5] (LWL-Link, 3,3-Mbit-Protokoll)
- ibaFOB-X[6] (LWL-Link, 32-Mbit-Protokoll)
- ibaFOB-D (LWL-Link, 2 MBit / 3,3 MBit / 5 MBit / 32-Mbit-Protokoll)

Für jeden Kartentyp gibt es die Varianten der Linkanzahl 1, 2 oder 4 für die Ein- und Ausgabe.

11.4.1.1 Karteneinstellungen

Wenn Sie in der linken Baumstruktur die Schnittstelle FOBF*xx*, FOBF*nn* oder FOBD*nn* markieren, werden rechts die dazugehörigen Karteneinstellungen angezeigt.

- Interrupt-Modus, siehe "Hardware-Ressourcen , Seite 179"
- Aktiviert, siehe "Hardware-Ressourcen , Seite 179"
- Variable Zykluszeit

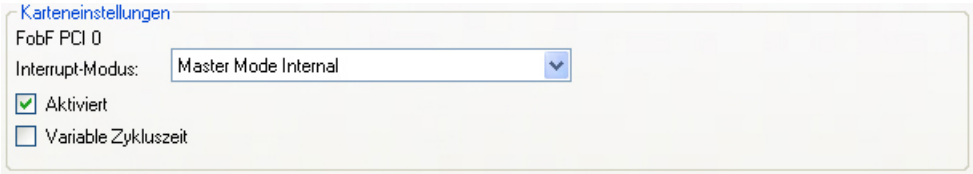


Abbildung 104: Karteneinstellungen

Auswahlfeld	Erläuterung
Aktiviert	Nur aktivierte Karten können verwendet werden.
Variable Zykluszeit	Dieser Modus ist nicht realisiert.

[5] Karten dieser Schnittstelle sind nur noch für Altanlagen erhältlich.
[6] Karten dieser Schnittstelle sind nur noch für Altanlagen erhältlich.

11.4.1.2 Verbindungseinstellungen

Im Abschnitt „Verbindungseinstellungen" werden die Konfigurationen der Kanäle angezeigt. Die Anzahl der Kanäle unterscheidet sich je nach der eingesetzten Variante der ibaFOB-io-Karte.

Einstellungen	Erklärung
Aktivieren	Hier können Sie einzelne Links aktivieren bzw. deaktivieren.
Eingabeformat	Stellen Sie hier das Datenformat der ankommenden Lichtwellenleitertelegramme ein. Abhängig von den angeschlossenen Geräten wählen Sie INTEGER, REAL oder S5REAL. Bei der Verwendung der FOB-X- und FOB-D-Karte werden weitere Datenformate für das 32 MBit LWL-Protokoll angeboten.

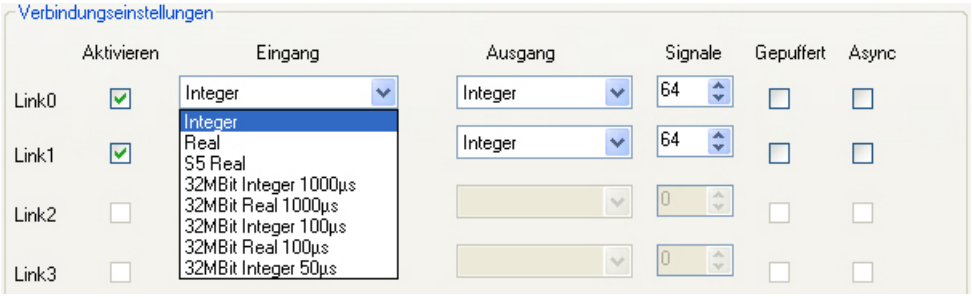


Abbildung 105: Verbindungseinstellungen

Der Datentyp muss mit dem Typ bzw. der Einstellung auf dem angeschlossenen Gerät übereinstimmen.

Gerät	Protokoll	Datentyp/Telegrammtyp
ibaPADU-8 ibaNet750	3,3 MBit	Integer
SIMATIC TDC/LO6	32 MBit	32 MBit Real 100 µs
ibaLink-SM-64-i-o	3,3 MBit	Integer, Real oder S5Real, abhängig von der Schaltereinstellung auf der Baugruppe.
ibaLink-SM-64-SD16 ibaLink-SM-128V-i-2 ibaBM-DPM-S-64	3,3 MBit	Integer oder Real, abhängig von der Einstellung auf der Baugruppe
iba-PADU-S-IT	32 MBit	abhängig von Einstellung in ibaPADU-S-IT
ibaBM-DPM-S (Profibus) ABB AC800PEC	32 MBit	32 MBit Real 1000 µs
ibaLink-VME	32 MBit	abhängig von Einstellung der Karte

Hinweis

Für FOB-X-Karten gilt: Wird eingabeseitig ein 32-MBit-Protokoll verwendet, kann der dazugehörige Ausgabelink nicht verwendet werden.

Einstellungen	Erklärung
Ausgabeformat	Datentyp des LWL-Telegramms in Ausgaberichtung. Die Einstellung ist abhängig vom angeschlossenen Gerät (siehe Eingabeformat).
Signale	Hier trägt ibaLogic, abhängig vom gewählten Datenformat, die maximal mögliche Signalanzahl ein. Sie können die Anzahl der Signale für Ihren Bedarf reduzieren. Die eingegebene Signalanzahl gilt für Analogwerte, Digitalwerte in Ein- und Ausgaberichtung. Für jeden aktivierten Link wird nach <Übernehmen> im Ressourcenbaum unter der entsprechenden Schnittstelle ein Modul „FobFnnMxx“ bzw. „FobDnnMxx“ angelegt, das „n" Analogsignale des eingestellten Typs und „n" Binärsignale hat (nn = Kartenindex, xx = Linknummer):
Gepufferter Modus	Damit können Sie für diesen Link einen Modus aktivieren, der die Empfangsdaten auch als Arrays zur Verfügung stellt. Sehen Sie dazu die folgende Beschreibung.

11.4.2 Buffered Mode

Der gepufferte Modus ist notwendig für die folgenden Anwendungsfälle:

- Die kleinste mögliche Intervallzeit von ibaLogic-V4 beträgt 1 ms. Wenn Signalwerte erfasst werden sollen, bei denen die Zykluszeit kleiner als 1 ms ist, müssen die Signalwerte gepuffert aufgezeichnet werden. Dieses ist vor allem im Zusammenhang mit den folgenden Baugruppen anzuwenden:
  - FOB-D und FOB-X im 32 MBit-Modus
  - Anschluss ibaPADU-S-IT (im I/O-Modus)
  - Anschluss der lokalen Peripherie beim Zielsystem PADU-S-IT
- Auch bei Peripheriesignalen, die im 1-ms-Zyklus kommen, aber aufgrund der hohen Signalanzahl und geforderten Rechenleistung deren Berechnung nicht im 1-ms-Intervall möglich ist.

Die eingelesenen Werte werden gepuffert und dem Programm als Array-Ressourcen zur Verfügung gestellt. Sinn dieses Modus ist es, das Programm von der Berechnung der einzelnen Signale zu entlasten, wenn diese nicht benötigt werden.

Beispiel:
Es werden Signale nur zur FFT-Analyse benötigt. Dann ist es nicht möglich (bei Sample-Zeit < 1 ms) und auch nicht sinnvoll, einzelne Signale zu erfassen und diese im Programm zu Arrays aufzusammeln, um sie der FFT-Analyse zuzuführen. Die Arrays werden in der richtigen Größe durch den gepufferten Modus erzeugt, so dass in dem Programm keine Rechenzeit zur Bearbeitung von Einzelwerten entsteht.

Der gepufferte Modus hat folgende Merkmale:

- Die Daten von dem FOB-Link werden vom Treiber in dem Zyklus der eingestellten Zeitbasis erfasst.
- Für jedes Signal werden die erfassten Daten in je einem Ringpuffer gesammelt. Die Größe des Ringpuffers und das Taktverhältnis für das Füllen werden vom Programm in den Ausgangsressourcen (s. u.) parametriert.
- Jedes Signal wird dem Programm in Form eines Arrays der Länge 256 zur Verfügung gestellt. Das Programm läuft in einem langsameren Taskintervall und liest die kompletten Arrays.
- Das Programm kann also keine einzelnen Samples bearbeiten, sondern nur Arrays von Samples, z. B. zur Berechnung einer FFT-Analyse oder zum Archivieren.
- Beispiel: Bei einer Erfassungsrate von 1 ms, einem Taskintervall von 50 ms kann trotzdem eine FFT-Analyse mit 128 Samples durchgeführt werden.
- Parallel zu den gepufferten Daten können auch die Einzelsignale z. B. für andere Programme erzeugt werden.

Hinweis

Gepufferter Modus ist nur möglich für den Erfassungsmodus „Messung".

11.4.2.1 Eingangsressourcen

Für jeden im gepufferten Modus aktivierten Link wird nach <Übernehmen> im Ressourcenbaum unter der entsprechenden Schnittstelle ein Modul FobFBuffer*nn*M*xx* bzw. FobDBuffer*nn*M*xx* angelegt (*nn* = Kartenindex, *xx* = Linknummer).

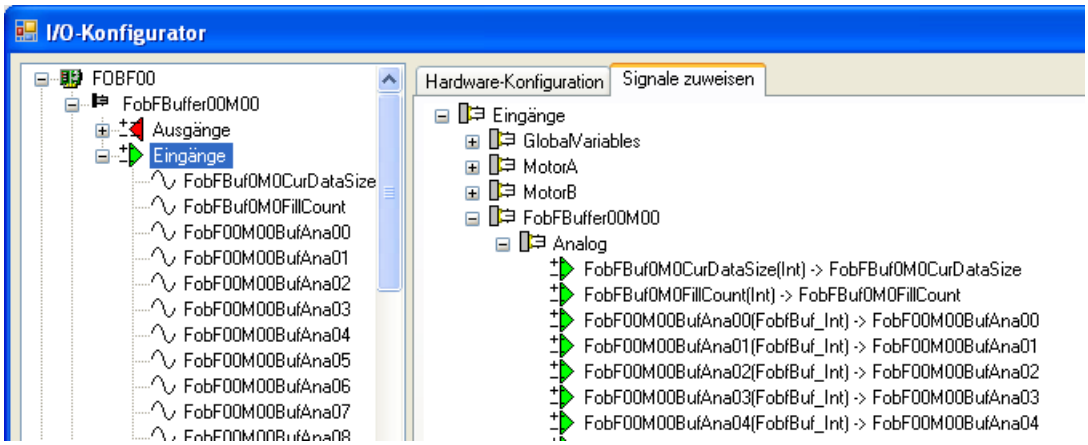


Abbildung 106: Eingangssignale zuweisen

Eingangssignale	Erklärung
CurDataSize	Rückmeldung der im Ausgangssignal parametrierten Puffergröße.
FillCount	Zähler, der erhöht wird, wenn das Eingangsarray bis zu der Länge „Datasize" gefüllt ist.
BufAna*ii*	Array vom Typ Integer bzw. Real der Länge 256.
BufDig*00*	Array vom Typ Bool der Länge 256.
Nur bei „variabler Zykluszeit"	
CurCycleTime	Rückmeldung der im Ausgangssignal parametrierten Interrupt-Zykluszeit.

11.4.2.2 Ausgangsressourcen

Die Ressourcen werden in dem Modultyp FobFBuffer*nn*M*xx* bzw. FobDBuffer*nn*M*xx* zur Verfügung gestellt (*nn* = Kartenindex, *xx* = Verbindungsindex).

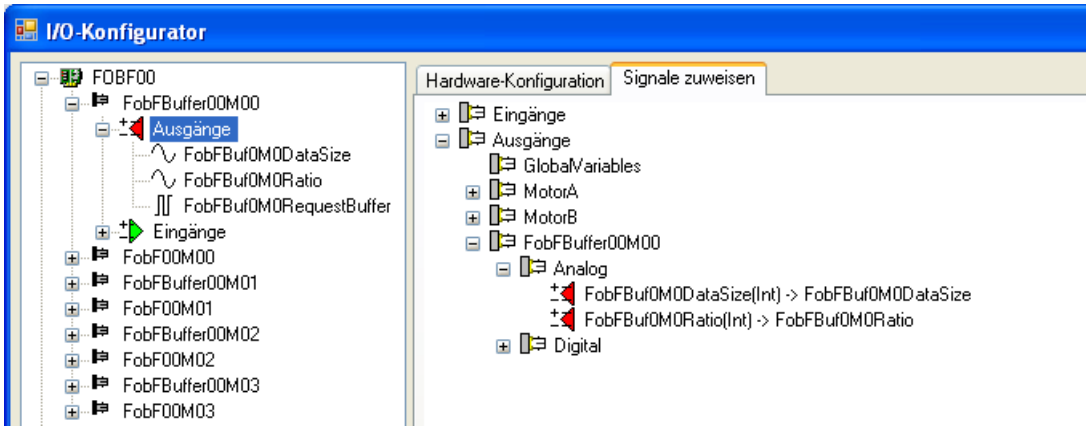


Abbildung 107: Ausgangsressourcen zuweisen

Ausgangssignale	Erklärung
Datasize	Anzahl der Messwerte, die vom Treiber in den Puffer eingetragen werden, bis der Füllzähler inkrementiert wird („Füllhöhe"). Zulässige Werte bis 256.
Ratio	Ganzzahliges Vielfaches der Zeitbasis, mit der die Puffer gefüllt werden. Z. B. Ratio=2 bedeutet, dass nur jeder 2. Messwert in den Puffer eingetragen wird.
RequestBuffer	Steuerung der Erfassung. Die Puffer werden nur gefüllt, wenn der Ausgang „TRUE" ist
Nur bei „variabler Zykluszeit"	
CycleTime	Variable Interruptzeit. Angabe in Mikrosekunden. Zulässiger Werte: 1000 … 10000.
SetCycleTime	Signal zur Übernahme der Zykluszeit (positive Flanke).

11.4.3 ibaLogic als Profibus-Slave

Der Profibus ist streng nach dem Master-Slave-Prinzip organisiert. Nach Norm DP-V0 findet die Kommunikation nur zwischen Master und Slaves statt, wobei die Verbindung vom Master aufgebaut und überwacht wird. ibaLogic kann sowohl Profibus-Master als auch Profibus-Slave sein.

Zum Beispiel muss für eine Verbindung zu einer SIMATIC S7, die selbst Master ist, ibaLogic als Slave arbeiten. Dazu dient die Profibus-Slave-Karte ibaCOM-L2B.

Unter dem Schnittstellentyp **L2B***nn* werden die folgenden Schnittstellenkarten angeordnet:

- ibaCOM-L2B 4/8
  (Profibus DP-Slave-Karte mit einer Buchse, 4 Slaves)
- ibaCOM-L2B 8/8
  (Profibus DP-Slave-Karte mit zwei Buchsen, 8 Slaves)

11.4.3.1 Karteneinstellungen

Wenn Sie in der linken Baumstruktur die Schnittstelle L2B*nn* markieren, dann werden rechts die dazugehörigen Karteneinstellungen angezeigt.

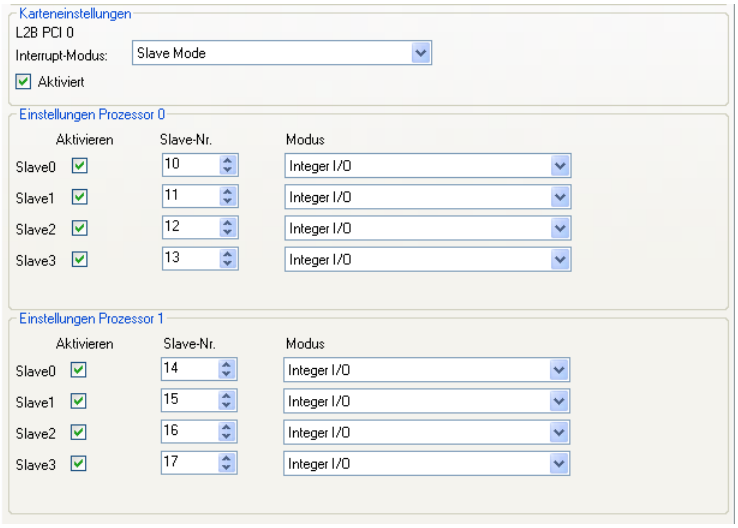


Abbildung 108: Karteneinstellungen ibaCom-L2B

- Interrupt-Modus, siehe "Hardware-Ressourcen , Seite 179"
- Aktiviert, siehe "Hardware-Ressourcen , Seite 179"

11.4.3.2 Einstellungen Busanschluss 0/1

Die L2B-Karte hat einen oder 2 Profibus-DP Anschlüsse. Für jeden Anschluss können 4 Slaves definiert werden.

Einstellungen	Erklärung
Aktivieren	Hier können Sie einzelne Slaves aktivieren bzw. deaktivieren.
Slave-Nr.	Geben Sie jedem aktivierten Slave eine eigene Stationsnummer.
Modus	Selektieren Sie für jeden aktivierten Slave ein Telegrammformat, das mit der Profibus-Master-Konfiguration übereinstimmt. Dazu liefert iba eine Reihe von GSD-Dateien, die den hier zur Auswahl stehenden Telegrammformaten entsprechen:

Telegramm-format	iba GSD-Datei	Inhalt	Datenrichtung
Integer_In	iba_0F01	32 Integer- und 32 Binärsignale	Master → ibaLogic
Real_In	iba_0F02	32 Real- und 32 Binärsignale	Master → ibaLogic
S7Real_In	iba_0F04	28 Real- und 32 Binärsignale	Master → ibaLogic
Integer_inOut	iba_0F08	32 Integer- und 32 Binärsignale	Master ←→ ibaLogic
Real_InOut	iba_0F09	32 Real- und 32 Binärsignale	Master ←→ ibaLogic
S7Real_InOut	iba_0F0B	28 Real- und 32 Binärsignale	Master ←→ ibaLogic

Für jeden aktivierten Slave wird nach <Übernehmen> im Ressourcenbaum unter der entsprechenden Schnittstelle ein Modul „L2B*nn*M*yy*S*xx*" angelegt, das die Analogsignale des eingestellten Typs und 32 Binärsignale hat (*nn* = Kartenindex 00-03, *yy* = Anschlussnummer 00-01), *xx* = Slave-Nummer 00-03).

11.4.4 ibaLogic als Profibus-Master

Wollen Sie von ibaLogic aus zum Beispiel eine ET200-Station ansprechen, dann muss ibaLogic als Master arbeiten. Dazu muss im ibaLogic-PC eine Profibus-Masterkarte installiert sein.

Unter dem Schnittstellentyp **SST_Master***nn* wird die folgende Schnittstellenkarte angeordnet:

- ❑ SST-PB3-PCU (ein Kanal, PCI)
- ❑ SST-PB3-PCU-2 (zwei Kanäle, PCI)
- ❑ SST-PB3-PCIE-1 (ein Kanal, PCI Express)
- ❑ SST-PB3-PCIE-2 (zwei Kanäle, PCI Express)

Hinweis

Die Verwendung der SST-Karte ist lizenpflichtig.

11.4.4.1 Kurzbeschreibung

Da diese Karte ein Produkt eines anderen Herstellers ist, weicht die Parametrierung vom Schema der iba-Karten ab.
Generell muss für den Profibus eine Konfiguration erzeugt werden, die alle am Profibus angeschlossenen Stationen und deren für die Kommunikation notwendigen Parameter enthält. Das Programm für die Erstellung der Konfiguration (SST Profibus Console) liefert als Ergebnis eine binäre Parameterdatei (.bss). Diese muss von ibaLogic im I/O-Konfigurator geladen und in die SST-Karte übertragen werden.

11.4.4.2 Karteneinstellungen

Wenn Sie in der linken Baumstruktur die Schnittstelle SST_Master*nn* markieren, dann werden rechts die dazugehörigen Karteneinstellungen angezeigt.

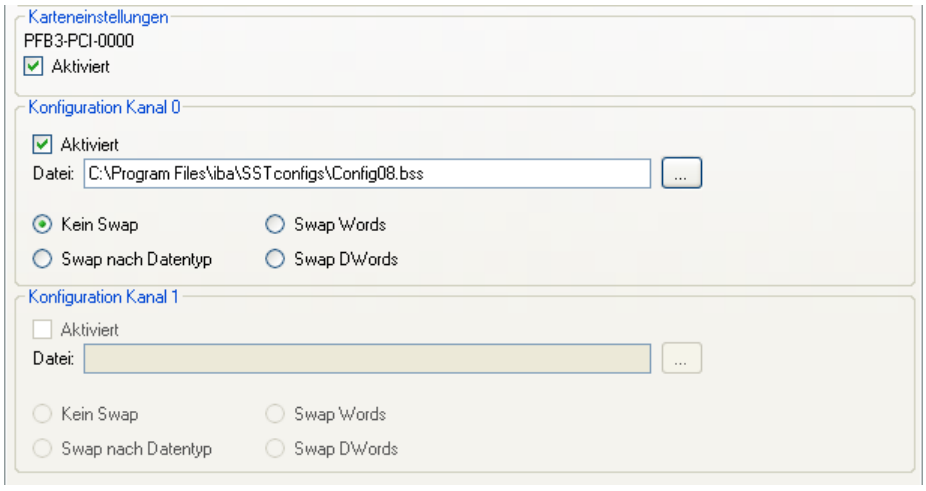


Abbildung 109: Karteneinstellung

„Aktiviert"

Mit dieser Option aktivieren bzw. deaktivieren Sie die Karte.

11.4.4.3 Konfiguration

„Datei"

Hier geben Sie die mit dem oben erwähnten CONSOLE-Programm erzeugte Datei an. Durch Klick auf den Button rechts davon öffnet sich ein Browser, mit dem Sie die Datei in der Dateiordnerstruktur auswählen können.

„Swap-Modes"

Je nach angeschlossenem Gerät ist eventuell ein Swap, d. h. ein Vertauschen von High- und Low-Teil des Datentyps erforderlich, damit die Daten in IEC-Norm lesbar in ibaLogic verarbeitet werden können.

Erläuterung der Swap-Methoden (jeder Buchstabe bedeutet ein Byte, Leerzeichen sind nur zur Verdeutlichung eingefügt):

Swap-Methode	Ausgang	Wird zu
Pro Datentyp	„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 Besonderheiten bei der Signalzuweisung

Wenn Sie die Parameterdatei der Karte mit <Übernehmen> aktiviert haben, werden im Ressourcenbaum unter SST_Master ein Modul mit dem Namen der SST-Karte „PFB3-PCI-000*x*" angelegt. Darunter sind für jeden möglichen Profibus-Slave folgende Signale sichtbar:

Richtung	Signal	Bedeutung
Eingang	SST*nn*InStatus*yyy*	Status- und Diagnoseinformation für Empfangstelegramm von Slave *yyy*. Datentyp: „SSTSTATUSSTRUCT"
Ausgang	SST*nn*OutStatus*yyy*	Status- und Diagnoseinformation für Sendetelegramm an Slave *yyy*. Datentyp: „SSTSTATUSSTRUCT"
Eingang	SST*nn*StructIn*yyy*	Empfangsdaten von Slave *yyy* Datentyp „SST_Struct"
Ausgang	SST*nn*StructOut*yyy*	Sendedaten an Slave *yyy* Datentyp "SST_Struct"

nn = Kartenindex 00-03, *yyy* = Profibus-Slave-Nummer von 002-127

Sie können nun für das komplette Modul die virtuellen Signale erzeugen (siehe „Signalzuweisung , Seite 185") oder nur für die von Ihnen projektieren Profibus-Slaves einzeln die Status-, Eingangs- und Ausgangssignale generieren.

Hinweis

Die hier erzeugten Signale sind Struktur-Datentypen.

Die intern generierte Statusstruktur „SSTSTATUSSTRUCT" besteht aus einer Nummer (ErrorCode) und einem Textstring (ErrorString).

Der Datentyp „SST_Struct" ist ein Platzhalter für die Strukturen der Nutzdatentelegramme, die Sie definieren müssen. Diese Struktur muss genau dem zu empfangenden bzw. zu sendenden Profibus-Telegramm entsprechen, wie Sie es in der Profibus-Konfiguration (Profibus Console) für jede Station festgelegt haben. Den Aufbau der Profibus-Telegramme (L2B) können Sie dem Handbuch der L2B-Karten entnehmen.

Weitere Informationen siehe „Datentypen , Seite 290".

Wichtiger Hinweis

Auf der Liefer-CD ist ein dokumentiertes Beispiel für den Anschluss der Profibus-Masterkarte beigelegt.

11.4.5 SIMADYN D-/SIMATIC TDC-Anbindung

Für die Anbindung an diese Systeme hat iba zwei PCI-Karten entwickelt, die sich nur durch die unterschiedlichen LWL-Anschlusstechniken und -Protokolle unterscheiden.

- Unter dem Schnittstellentyp **FOBSD** wird die Schnittstellenkarte ibaFOB-SD angeordnet. Diese ermöglicht den Anschluss an die SIMADYN D-Welt über die Rahmenkopplungsbaugruppen CS12, CS13 und CS14 sowie an die SIMATIC TDC-Rahmen mit der Kommunikationsbaugruppe CP53M0.
- Unter dem Schnittstellentyp **FOBTDC** wird die Schnittstellenkarte ibaFOB-TDC angeordnet. Diese ermöglicht den Anschluss an die SIMATIC TDC-Welt über den GDM (Global Data Memory, Schnittstellenbaugruppe CP52IO).

Hinweis

Die Einstellungen für FOBSD und FOBTDC-Module sind identisch. Deswegen werden hier die Einstellungen am Beispiel eine ibaFOB-SD-Karte erläutert.

11.4.5.1 Karteneinstellungen

Wenn Sie in der linken Baumstruktur die Schnittstelle FOBSD markieren, dann werden rechts die dazugehörigen Karteneinstellungen angezeigt.

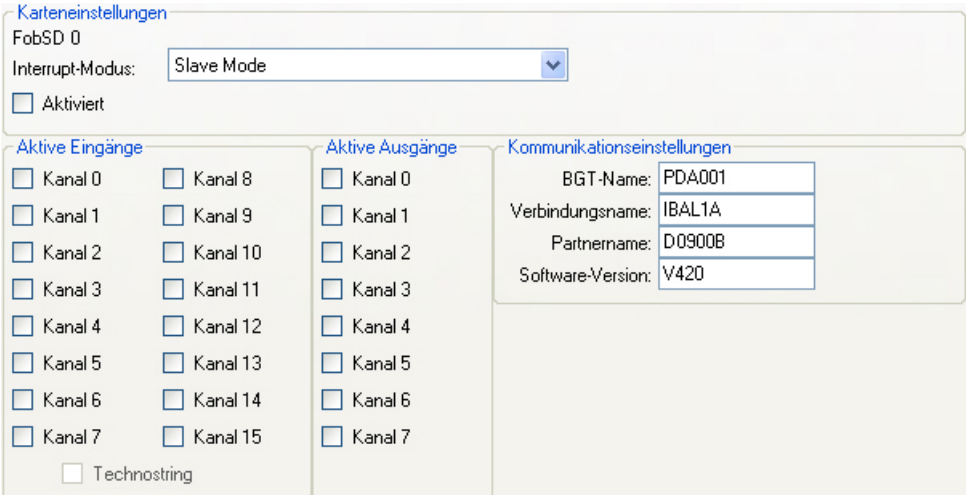


Abbildung 110: Karteneinstellungen ibaFOB-SD/TDC

- ❑ Interrupt-Modus, siehe "Hardware-Ressourcen , Seite 179"
- ❑ Aktiviert, siehe "Hardware-Ressourcen , Seite 179"

11.4.5.2 Verbindungseinstellungen

Aktive Eingänge

Selektieren Sie hier gezielt einen oder mehrere Eingänge aus insgesamt 16 Kanälen. Beachten Sie, dass für jeden gewählten Eingangskanal ein Sendetelegramm in SIMADYN D bzw. SIMATIC TDC vorhanden sein muss.
Ein Sendetelegramm enthält genau 32 Real-Werte (Datentyp NF bei SIMADYN D) und 32 Binärwerte (1 DWORD- bzw. V4-Wert).

Am Sendebaustein sind folgende Parameter einzustellen:

- ❑ AT (Kanalname): MxPDADAT (x = 0 … F für Kanal 0 bis 15)
- ❑ MOD (Kanalmode): R (für Refresh)
- ❑ LEN (Telegrammlänge): 132 (nur bei Sendebaustein CTV_P)

Aktive Ausgänge

Hier können gezielt einer oder mehrere Ausgänge aus insgesamt 8 Kanälen selektiert werden.

Beachten Sie, dass für jeden gewählten Ausgangskanal ein Empfangstelegramm in SIMADYN D bzw. SIMATIC TDC vorhanden sein muss.

Ein Empfangstelegramm enthält genau 32 Real-Werte (Datentyp NF bei SIMADYN D) und 32 Binärwerte (1 DWORD- bzw. V4-Wert).

Am Empfangsbaustein sind folgende Parameter einzustellen:

- ❑ AR (Kanalname): PDAMxDAT (x = 0 … 7 für Kanal 0 bis 7)
- ❑ MOD (Kanalmode): R (für Refresh)
- ❑ LEN (Telegrammlänge): 132 (nur bei Empfangsbaustein CRV_P)

Für jeden aktivierten Kanal wird nach <Übernehmen> im Ressourcenbaum unter der entsprechenden Schnittstelle ein Modul „FOBSD*nn*CH*xx*" angelegt, das insgesamt 32 Real- und 32 Binärsignale enthält (*nn* = Kartenindes, *xx* = Kanalindex)

Technostring

Die Funktion „Technostring" ist noch nicht freigegeben.

11.4.5.3 Kommunikationseinstellungen

- ❑ BGT-Name:
  Der PC muss der SD/TDC-Umgebung durch einen sechsstelligen Namen bekannt gemacht werden, z. B. „IBA001".
- ❑ Verbindungsname:
  Mit diesem Namen meldet sich die ibaFOB-SD beim Kommunikationspartner SIMADYN D bzw. SIMATIC TDC an. Dieser Name muss innerhalb der CS14 bzw. CP53M0-Kommunikationsinsel eindeutig sein, d. h. es dürfen sich keine anderen Siemens- oder iba-Baugruppen mit demselben Namen anmelden. Default ist „IBAL1A".
- ❑ Partnername:
  Tragen Sie hier den auf SIMADYN D bzw. SIMATIC TDC projektierten Name der Koppelpartnerbaugruppe ein.
  Bei SIMADYN D ist es der Name der CS14-Baugruppe, z. B. „D0500B",
  bei SIMATIC TDC (GDM) der Name der GDM-Baugruppe, normalerweise D01_P1.
- ❑ Software-Version:
  Tragen Sie hier die Softwareversion der SIMADYN D bzw. SIMATIC TDC Software ein: z. B. bei STRUC „V420", bei CFC „V610".

Hinweis

Sind diese Einstellungen nicht korrekt, dann findet keine Kommunikation statt.

11.4.6 Reflective Memory

Der Zugang zu VME-Bus-basierten Fremdsystemen (z. B.: GE FANUC, Converteam HPCi) ist mit Reflective-Memory-Karten möglich.

Unter dem Schnittstellentyp RFM werden die folgenden Schnittstellenkarten angeordnet:

- VMIC PCI-5565PIORC:
  (Reflective-Memory-Karte, 64 oder 128 MByte)
- VMIC PCI-5588 u.a.:
  (Reflective-Memory-Karten älterer Bauart)

11.4.6.1 Kurzbeschreibung

Weil diese Karte ein Produkt eines anderen Herstellers ist, weicht die Parametrierung vom Schema der iba-Karten ab.

Da der Speicher einer RFM-Karte keinen homogenen Datenbereich hat, sondern Bereiche mit unterschiedlichen Datentypen enthält, können nicht dieselben Parameter wie für iba-FOB-Karten verwendet werden.

Die Position und Struktur der verwendeten Daten müssen Sie in einer externen Parameterdatei definieren. Diese wird im ibaLogic I/O-Konfigurator geladen und daraus die Signale in den gewünschten Datentypen erzeugt.

11.4.6.2 Karteneinstellungen

Wenn Sie in der linken Baumstruktur die Schnittstelle RFM markieren, dann werden rechts die dazugehörigen Karteneinstellungen angezeigt.

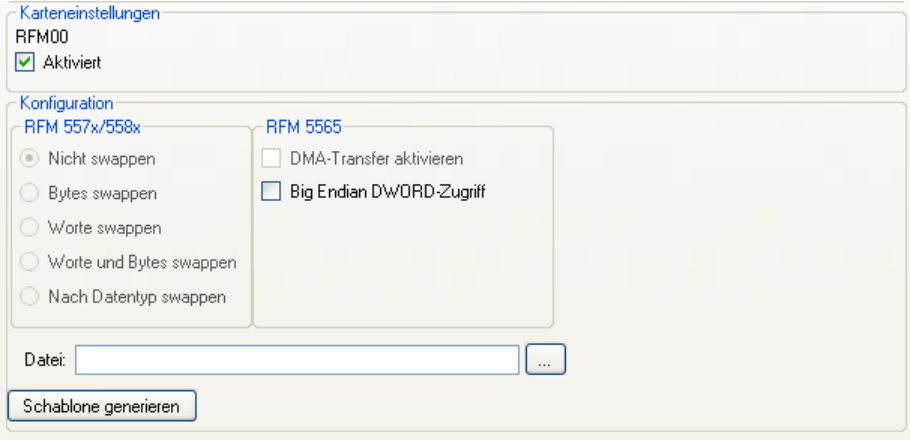


Abbildung 111: Reflective-Memory-Karten

- Aktiviert
- Mit dieser Option aktivieren bzw. deaktivieren Sie die Karte.

11.4.6.3 Konfiguration

- ❑ Swap-Modus

Der Swap-Modus ist nur für die älteren RFM-Karten verfügbar, die dies direkt in der Hardware unterstützen.
Neuere Karten (PCI-5565 oder PCIE-5565) unterstützen das nicht mehr.

Hinweis

Die Methoden sind hier anders definiert als bei der SST-Karte. Hier werden die Bezeichnungen von der VMIC übernommen, so dass diese mit den daran angeschlossenen RFM-Karten übereinstimmen.

Erläuterung der Swap-Methoden (jeder Buchstabe bedeutet 1 Byte, Leerzeichen sind nur zur Verdeutlichung eingefügt).

Swap-Mode	Erklärung
Nicht	„ABCD EF G H IJKL"
Bytes	„BADC FE H G JILK"
Worte	„CDAB GH E F KLIJ"
Worte and Bytes	„DCBA HG F E LKJI"
Nach Datatyp	„DCBA FE G H LKJ"

11.4.6.4 File/Datei

Parameterfile, das die Beschreibung der Signale enthält.

Vorgehen

1. Erzeugen Sie eine Vorlage für diese Daten mit dem Button <Schablone generieren>.
2. Diese Datei öffnen Sie anschließend mit einem Editor.
3. Tragen Sie für jedes Signal eine Zeile in folgendem Format ein:
   „Signalname, Datentyp, Speicheradresse, Bitnummer, Richtung, Kommentar"

Format	Erklärung
Signalname	Muss der IEC-Norm entsprechen und eindeutig sein, d.h. auch Input- und Outputsignalnamen müssen sich unterscheiden.
Datentyp	Elementarer Datentyp (siehe "Standard-Datentypen , Seite 290") BOOL BYTE, WORD, DWORD SINT, USINT, INT, UINT, DINT, UDINT REAL, LREAL
Speicheraddresse	Offset innerhalb des RFM-Speichers als Dezimal- oder Hexadezimalzahl. Die Umschaltung erfolgt durch ein Zeile mit „#hexval" bzw. „#intval" am Anfang.
Bitnummer	Nur bei Datentyp BOOL relevant, sonst 0
Richtung	„INPUT" oder „OUTPUT"
Kommentar	Beliebiger Text

Beispiel

```
#HexVal
TestSignal1,REAL,0x1000,0,INPUT, Testsignal input
TestSignal2,REAL,0x2000,0,OUTPUT, Testsignal 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
```

Ergebnis

Es wird eine Vorlage-Datei erzeugt.

11.4.6.5 Ablauf zur Parametrierung

Vorgehen

1. Drücken Sie den Button <Schablone generieren> und geben Sie Pfad und Dateinamen ein, um eine CSV-Datei als Vorlagen zu erzeugen.
2. Öffnen Sie diese Daten mit einem Editor und tragen Sie dort die Signale ein.
3. Öffnen Sie die geänderte Datei im I/O-Konfigurator und laden die Konfiguration in die RFM-Karte mit <Übernehmen>.
4. Warten Sie die Initialisierungsphase ab bis die RFM-Karte mit diesen Parametern versorgt wird.
5. Dann drücken Sie nochmals auf <Hardware aktualisieren>, damit die definierten Signale im Ressourcenbaum angezeigt werden.
6. Weisen Sie den generierten Signalen die virtuellen Signalnamen zu.

11.5 Zielsystem ibaPADU-S-IT

Die I/O-Konfiguration des ibaPADU-S-Systems ist nur verfügbar, wenn als Zielsystem ein Gerät ibaPADU-S-IT gewählt wurde.

ibaPADU-S-IT ist die Zentraleinheit für die ibaPADU-S-Familie der modularen Gerätereihe für intelligente, dezentrale Ein-/Ausgaben.

Das Modularkonzept basiert auf einem Baugruppenträger mit Rückwandbus, auf den die Zentraleinheit und bis zu 4 weitere Ein-/Ausgabemodule gesteckt werden können.

Als I/OModule stehen Baugruppen für analoge und digitale Ein-/Ausgaben für unterschiedliche Signalpegel, für Strom und Spannungssignale und für Erfassungsraten bis 1 kHz bei ungepuffertem Zugriff bzw. maximal 40 kHz bei gepuffertem Zugriff zur Verfügung.

Hardware Dokumentation

Detaillierte Informationen über die ibaPADU-S-Eigenschaften entnehmen Sie dem ibaPADU-S-IT-Handbuch (siehe "Support und Kontakt , Seite 347").

11.5.1 Einstellungen

Wenn Sie mit dem Zielsystem verbunden sind und den Button <Hardware aktualisieren>drücken, dann sehen Sie die folgenden PADU-S-Einstellungen.

PADU-S-Einstellungen	Erklärung
Interrupt-Quelle	DNS-Name des PADU-S-IT, z.B. "S-IT-16-000074"
Moduleinstellungen	je nach I/O-Ressource
Signaleinstellungen	je nach I/O-Ressource
I/O-Ressourcen	PADU-S-IT inkl. PADU-S-Module TCPIP_OUT GLOBALVAR

Moduleinstellungen

Im Abschnitt „Moduleinstellungen“ werden je nach Auswahl bei den I/O-Ressourcen alle verfügbaren Module des ibaPADU-S-Systems dargestellt.

❑ Aktiviert:

Einzelne Module können Sie komplett deaktivieren.

❑ Gepufferter Zugriff:

Falls der gepufferte Zugriff aktiviert ist, sind weitere Konfigurationen im ibaLogic-Programm vorzunehmen (siehe auch Handbuch des ibaPADU-S-IT-16).

Nur mit einem gepufferten Zugriff ist es möglich eine Erfassungsrate von bis zu 20 kHz auf dem ibaPADU-S-Systems zu erreichen.

Im ungepufferten Zugriff steht max. 1 kHz zur Verfügung (Task-Intervall = 1 ms)

❑ Werte in REAL wandeln:

Ist diese Option angewählt, werden die Signale nicht als INT, sondern als REAL erfasst und können so ohne weitere Konvertierung im Programm weiter verarbeitet werden.

Hinweis

Je nach Modultyp stehen die Optionen „Gepufferter Zugriff“ und „Werte in REAL wandeln“ nicht zur Verfügung.

Signaleinstellungen

Im Abschnitt „Signaleinstellungen“ werden je nach Auswahl bei den I/O-Ressourcen und entsprechenden Moduleinstellungen alle konfigurierbaren Signale der jeweiligen I/O-Resource angezeigt (siehe auch Handbuch des ibaPADU-S-IT-16).

11.6 TCP/IP-Kommunikation

Es gibt folgende Arten der TCP/IP-Kommunikation:

- ❑ allgemeine TCP/IP-Verbindung über den Baustein (weitere Informationen siehe „TCPIP_SENDRECV , Seite 106")
- ❑ ibaPDA-Datenübertragung per TCP/IP

Die TCP/IP-Kommunikation, die im I/O-Konfigurator parametriert werden kann, beschränkt sich zurzeit auf TCP/IP-Sendetelegramm an ibaPDA. Im Unterschied zu der nativen TCP/IP-Kommunikation mit dem Baustein TCPIP_SENDRECV wird hier mit einem speziellen Protokoll die Modulstruktur im ibaPDA unterstützt. Unter den Zielsystemen WinXP bzw. PADU-S-IT können insgesamt 16 Telegramme mit je 32 Real-Werten und 32 Digitalwerten an einen oder mehrere ibaPDA-Empfänger gesendet werden.

Hinweis

Beachten Sie, dass die Kommunikation über TCP/IP nicht echtzeitfähig ist. D. h. zyklisch gesendete Daten werden nicht zyklusgenau empfangen, evtl. gehen Telegramme verloren etc.

11.6.1 TCPI/IP-Verbindungseinstellungen

Für die Einstellung wählen Sie links im Baum den entsprechenden Kanal aus.

Sie können jeden der 16 Kanäle einzeln aktivieren und folgende Parameter eingeben:

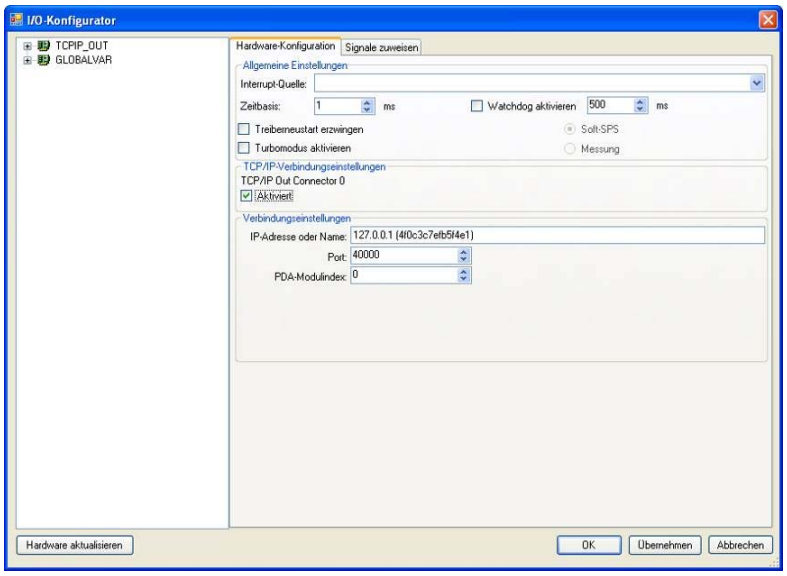


Abbildung 112: TCPI/IP-Verbindungseinstellungen

- ❑ IP-Adresse oder Name der Gegenstation, d. h. des Rechners auf dem der ibaPDA-Server läuft.
- ❑ Portnummer: Diese muss mit der in ibaPDA übereinstimmen. Default-Einstellung 40000.

Hinweis

Auf dem ibaPDA System muss der eingestellte Port in der Firewall freigeschaltet sein damit die Daten mit ibaPDA gemessen werden können.

- ❑ PDA-Modulnummer ist eine eindeutige Nummer 0..63. Diese muss einem Modulindex des angeschlossenen PDA-Systems entsprechen.

11.7 OPC-Kommunikation

Für die Verbindung zu HMI-Systemen (Human Machine Interface, Bedienen & Beobachten) oder zur Messung von langsamen Signalen hat sich der internationale OPC-Standard (OLE for Process Control) durchgesetzt.

11.7.1 OPC-Server

Der ibaLogic OPC-Server stellt alle als „OPC-sichtbar" definierten Variablen den OPC-Clients zur Verfügung, die sich mit dem OPC-Server verbinden. Der OPC-Server läuft im Regelfall auf derselben Maschine wie der ibaLogic Server und ist mit dem PMAC per TCP/IP verbunden.

Hinweis

Die Anzahl erlaubter OPC-Variablen ist abhängig von der erworbenen Lizenz (Dongle).

Hinweis

Es ist auch möglich den OPC-Server auf einem anderen Rechner laufen zu lassen. Dieses ist gesondert bei der iba anzufragen.

Ein OPC-Client findet den ibaLogic OPC-Server unter dem Namen **„iba.Logic4OPC.1".** Ist die Verbindung hergestellt, so können über einen BrowserDienst die OPC-Variablen durch ihren Variablennamen ausgewählt werden.

Der OPC-Server arbeitet nach der Spezifikation DA (Data Access) V2.05a.

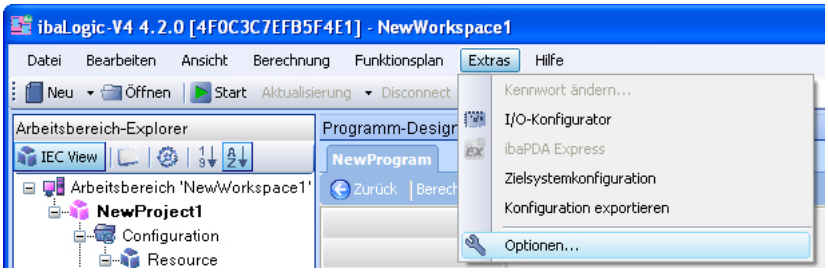
Hinweis

Für die Verbindung von OPC-Client zu -Server sind eine Reihe von Einstellungen in DCOM und Sicherheitsrichtlinien durchzuführen.

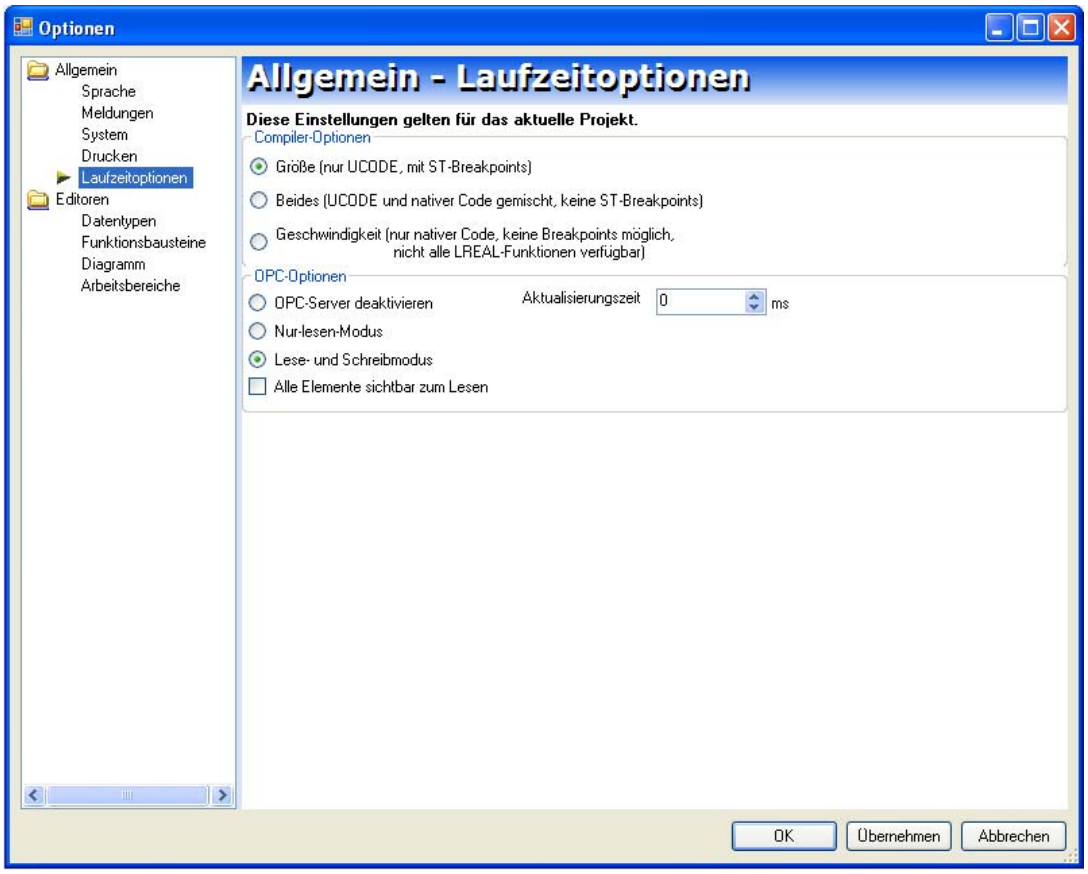
Dokumentation

Hierzu gibt es ein eigenes Dokument, das bei iba angefordert werden kann.

1. Um bestimmte OPC-Server-Eigenschaften einzustellen, öffnen Sie die Optionen von ibaLogic mit dem Menüpunkt „Extras - Optionen…".



2. Wechseln Sie links in der Baumstruktur auf „Laufzeitoptionen".



3. Stellen Sie die gewünschte Option ein.

Option	Erklärung
OPC-Server deaktivieren	Der OPC-Server wird komplett abgeschaltet.
Nur-Lesen Modus	Auch die als „OPC-beschreibbar" parametrierten Variablen können nur gelesen werden.
Lese- und Schreibmodus	Standardard-Einstellung: Die Zugriffe sind so wie in den Variablen parametriert.
Alle Elemente sichtbar zum Lesen	Auch die nicht „OPC-sichtbaren" Variablen können von OPC-Clients gelesen aber nicht beschrieben werden.

11.7.2 Parametrierung der OPC-Variablen

In ibaLogic muss für jedes gewünschte OPC-Signal ein Off-Task-Konnektor (OTC) angelegt werden. Im Dialog „Off-Task-Konnektor bearbeiten" müssen die OPC-Auswahlfelder aktiviert werden.

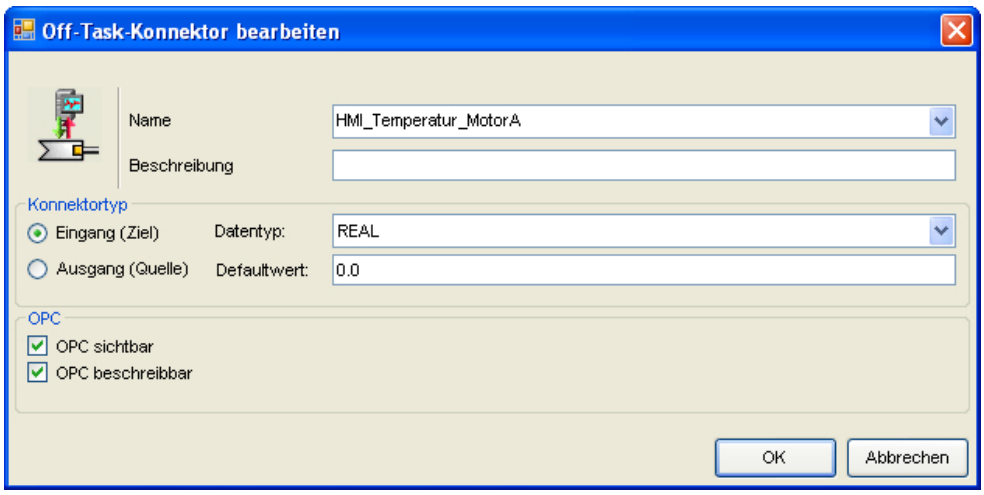


Abbildung 113: Off-Task-Konnektor bearbeiten

Welche Datentypen für die OPC-Verbindung erlaubt sind, hängt auch vom verwendeten Datentyp des OPC-Clients ab. Normalerweise können Sie die elementaren Standard-Datentypen und Arrays verwenden.

Die kürzeste Aktualisierungszeit beträgt 50 ms aus der Sicht von ibaLogic, allerdings ist zu beachten, dass dies stark von der Datenmenge abhängt.

Die OPC-Variablen sind farblich besonders gekennzeichnet. Mehr Informationen zur Parametrierung siehe „Off-Task-Konnektoren , Seite 158".

Der OPC-Server ist vom OPC-Server-Browser unter dem Namen „iba.Logic4OPC.1" zu finden.

12 Datenbankverwaltung

ibaLogic ist eine datenbankbasierte Anwendung. Um Zwischenstände zu speichern, sollten regelmäßige Sicherungen durchgeführt werden. Hierbei unterstützt Sie ibaLogic-V4 durch die Wahl zwischen einer automatischen und einer manuellen Sicherung.

12.1 Datenbank sichern

Eine Datenbanksicherung sollte immer in Betracht gezogen werden, bevor umfangreiche Änderungen vorgenommen werden.

12.1.1 Datenbank manuell sichern

Gesichert wird immer die komplette aktuelle ibaLogic-Datenbank. In dieser Datenbank können sich mehrere Arbeitsbereiche befinden. Welche Arbeitsbereiche aktuell in der Datenbank existieren, kann man im Client unter dem Menüpunkt „Arbeitsbereich öffnen" sehen.

Voraussetzung

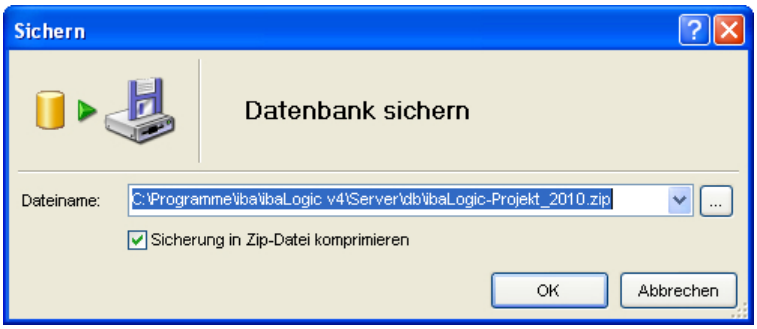
- ❑ Sie haben den ibaLogic Server-Dialog geöffnet.
- ❑ Sie haben eine Verbindung zur Datenbank.

Vorgehen

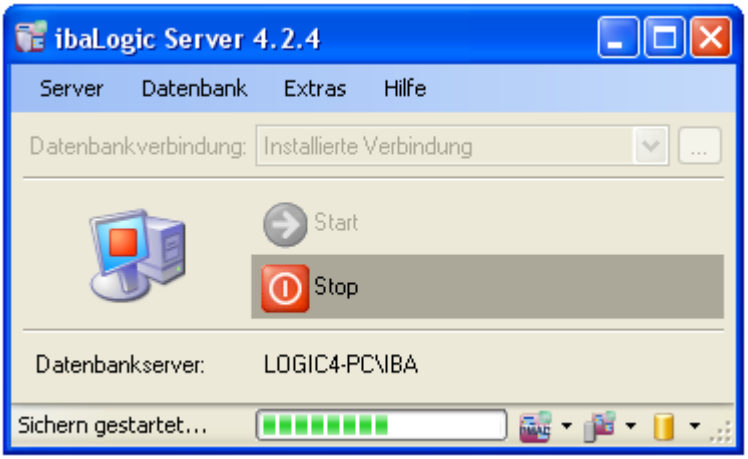
1. Wählen Sie Menü „Datenbank - Sichern…".

Es wird folgender Dialog geöffnet:

2. Wählen Sie einen Dateinamen und ein Verzeichnis aus, in dem die Sicherungsdatei gespeichert werden soll.
   Aktivieren Sie die Option „Sicherung in Zip-Datei komprimieren", wenn die Hardware-Konfiguration des I/O-Managers und alle vorhandene DLLs mit gesichert werden sollen.



3. Klicken Sie auf <OK>.



Wichtiger Hinweis

Insbesondere wird empfohlen eine Datenbanksicherung vor einem Versions-Update von ibaLogic vorzunehmen.

Im Zuge der Weiterentwicklung von ibaLogic, werden beim Update auch Änderungen in der Datenbank durch Datenbank-Skripte ausgeführt. Ein Rückrüsten auf eine ältere Version ist dann nicht mehr möglich.

12.1.2 Datenbank automatisch sichern

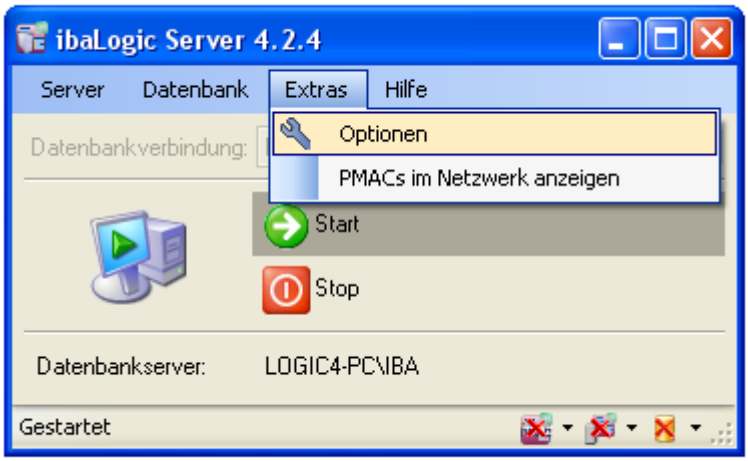
In ibaLogic stellen Sie das automatische Sichern der Datenbank ein.

Voraussetzung

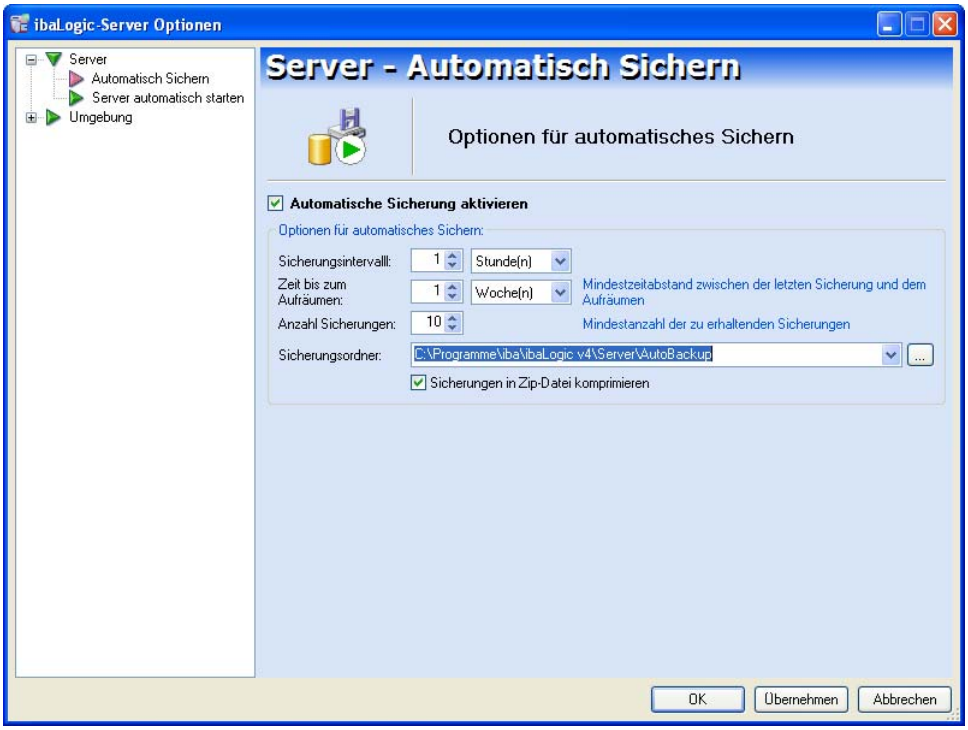
- ❑ Sie haben den ibaLogic Server-Dialog geöffnet.
- ❑ Sie haben eine Verbindung zur Datenbank.

Vorgehen

1. Wählen Sie Menü „Extras - Optionen".



2. Wählen Sie im Verzeichnisbaum „Server - Automatisch Sichern".



3. Setzen Sie ein Häkchen in der Checkbox „Automatische Sicherung aktivieren".
4. Wählen Sie den Zeitraum für das Sicherungsintervall.

Da nur mit ibaLogic Clients Änderungen an ibaLogic-Projekten vorgenommen werden können, wird das Intervall erst dann gestartet, wenn ein Client mit dem Server verbunden ist. Falls eine Änderung in der Datenbank erkannt wurde, wird nach Ablauf der Intervallzeit ein neues Backup in Form einer *.bak- bzw. einer *.zip -Datei erstellt.

Wichtiger Hinweis

Geben Sie unterschiedliche Ordner für das automatische und das manuelle Sichern an. Da die Aufräumstrategie nur den Ordner der automatischen Sicherung aufräumt, verhindern Sie dadurch, dass Ihre manuellen Sicherungen auch gelöscht werden.

Die Aufräumstrategie wird durch die Kombination der Felder

- „Zeit bis zum Aufräumen"
- „Anzahl Sicherungen"

bestimmt.

Option	Erklärung
Sicherungsintervall	Die Einstellung bewirkt, dass in dem angebenem Intervall Sicherungen erstellt werden.
Zeit bis zum Aufräumen	Die Einstellung bewirkt, dass Sicherungen solange abgelegt werden, bis der Mindestabstand der Sicherung und dem Aufräumen erreicht ist.
Anzahl Sicherungen	Die Option bestimmt die minimale Anzahl an Sicherungen die immer erhalten bleiben. Dabei wird noch der Zeitraum „Zeit bis zum Aufräumen" berücksichtig.
Sicherungsordner	Die Dateinamen werden von ibaLogic vorgegeben: *„Autobackup_ibaLogic4_<Datum, Uhrzeit>.bak"* bzw. *„.....zip"*. Datum und Uhrzeit sind in der Form JJJJMMTTHHMM definiert.
Sicherungen in Zip-Datei komprimieren	Die Option bewirkt, dass die Sicherung als eine Zip-Datei abgelegt wird.

Beispiel

Haben Sie die Einstellungen, die in dem oberen Fester zu sehen sind, dann werden alle Sicherungen, die älter als eine Woche sind, gelöscht. Es bleiben aber mindestens 10 Dateien erhalten. Diese können dann beliebig alt sein.

12.2 Datenbank wiederherstellen

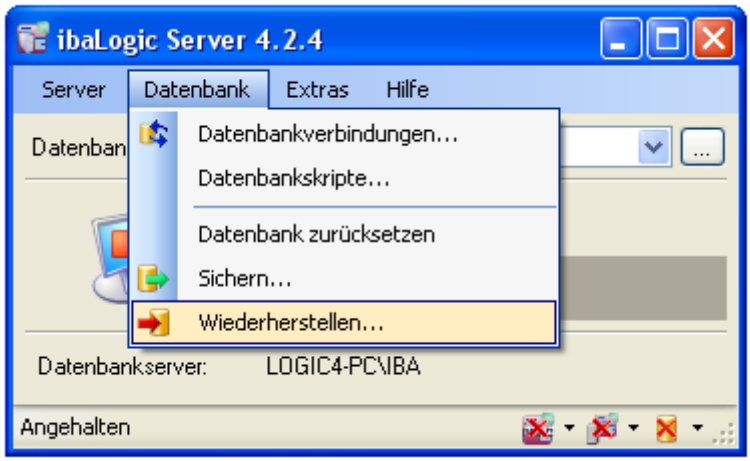
Wiederherstellen bedeutet, dass ein vorher gesicherter Stand einer Datenbank mit den darin enthaltenen Arbeitsbereichen zur Bearbeitung geladen werden wird.

Voraussetzung

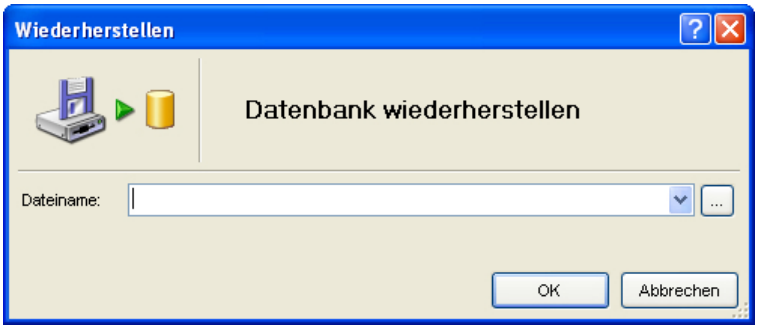
- ❑ Sie haben den ibaLogic Server-Dialog geöffnet.
- ❑ Sie haben eine Verbindung zur Datenbank.
- ❑ Sie haben den ibaLogic Server gestoppt.

Vorgehen

1. Wählen Sie Menü „Datenbank - Wiederherstellen…".

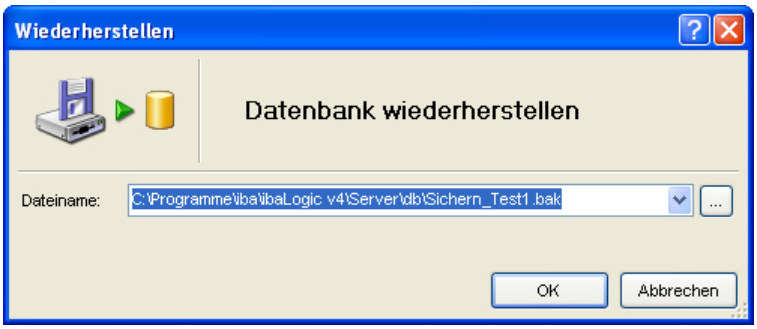


Der Dialog „Wiederherstellen" wird angezeigt.



2. Klicken Sie auf den Button <…> und wählen Sie eine Sicherungsdatenbank (ZIP- oder BAK-Datei) aus dem geöffneten Verzeichnis aus.

   ibaLogic bietet Ihnen standardmäßig das unten angegebene Verzeichnis an bzw. merkt sich den Pfad auf den zuletzt zugegriffen wurde.



3. Klicken Sie auf den Button <OK> zum Wiederherstellen.

   Der Fortschritt des Wiederherstellens wird Ihnen angezeigt. Danach befindet sich der Server im Zustand „angehalten".

4. Bestätigen Sie ggf. aufkommende Sicherheitsabfragen.
5. Starten Sie den Server für die Wiederaufnahme der Programmierarbeiten über den Client.

12.3 Datenbank zurücksetzen

Mit dieser Funktion setzten Sie Ihre aktuelle Datenbank auf deren Ursprungszustand (leer) zurück.

Wichtiger Hinweis

Damit werden auch alle Daten der Datenbank gelöscht.

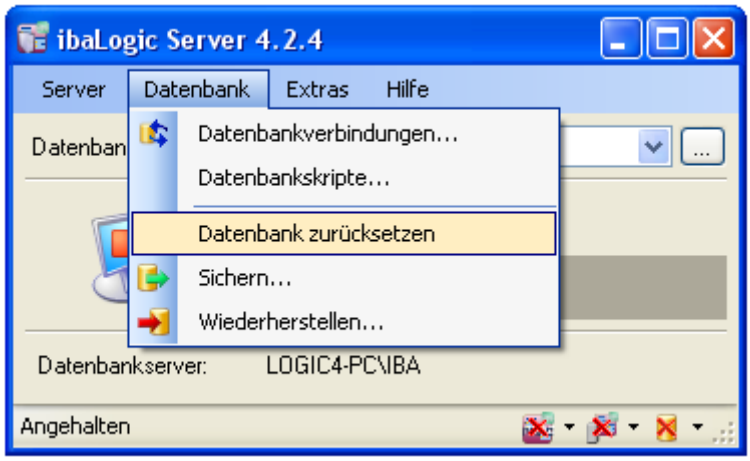
Führen Sie vorher eine Sicherung der Datenbank durch.

Voraussetzung

- ❑ Sie haben den ibaLogic Server-Dialog geöffnet.
- ❑ Sie haben eine Verbindung zur Datenbank.
- ❑ Sie haben den ibaLogic Server gestoppt.

Vorgehen

1. Wählen Sie Menü „Datenbanken – Datenbank zurücksetzen…".



2. Bestätigen Sie den Dialog mit OK wenn Sie die Datenbank wirklich zurücksetzen wollen.

13 Programmanalyse, Fehlersuche, Zeitverhalten

Ihnen stehen für die Programmanalyse und das Debugging verschiedene Verfahren und Tools zur Verfügung.

Unterschieden werden muss, ob die Anwendung noch im Test-Umfeld oder bereits im Einsatz aktiv ist.

Beschreibung	Testumfeld	Aktiver Einsatz
Debugging von Structured Text-Bausteinen über Breakpoints	Ja	Nein (Programm wird angehalten.)
Vom Anwender selbst erstellte Trace-Bausteine oder Log-DLLs (z. B. LogFile_String_WriteDll.dll, ……)	Ja	Bedingt (Zeitverhalten)
Analyse des Zeitverhaltens, Kurvenform etc. mit **ibaPDA Express**	Ja	Ja
Schreiben von DAT-Files für ibaAnalyzer mit Hilfe des **DAT_FILE_WRITE-Bausteins**	Ja	Ja

13.1 ibaPDA Express

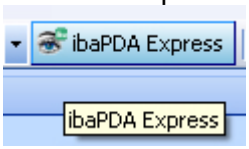
Zum schnellen Überprüfen einer Signalform dient ibaPDA Express.

Voraussetzung

Die Funktion steht nur zur Verfügung, wenn das Programm online geschaltet ist.

Vorgehen

➲ Starten Sie den ibaPDA Express mit einem Mausklick auf den Button <ibaPDA Express> in der Symbolleiste.



Ergebnis

ibaPDA Express wird mit einem eigenständigen Fenster innerhalb der ibaLogic-Anwendung geöffnet.

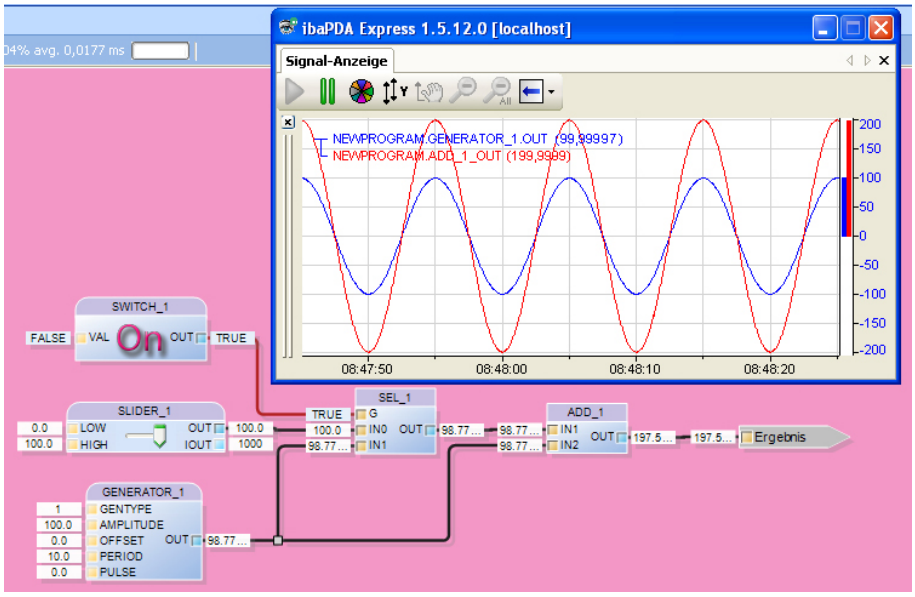


Abbildung 114: ibaPDA Express mit mehreren Signalen

13.1.1 Bedienung der Signalanzeige

Für die Bedienung der Signalanzeige steht folgende Symbolleiste zur Verfügung.

Abbildung 115: ibaPDA Express: Symbolleiste

In der folgenden Tabelle finden Sie Erläuterungen zu den Symbolen.

Symbol	Name	Tasten-Bedienung	Erläuterung
	Start Vorschub	<F6> (Umschaltung)	Startet die laufende Anzeige beim aktuellen Zeitpunkt. Aktiv, wenn „Pause Vorschub" gedrückt.
	Pause Voschub	<F6> (Umschaltung)	Anhalten der laufenden Anzeige. Nach Betätigung erscheint ein Lineal im Graphen, das mit der Maus bewegt werden kann und mit dem die Kurven vermessen werden können. Anzeige der Signalwerte in der Legende. Die X-Achse kann mit der Maus verschoben werden. Somit können Werte aus der Vergangenheit betrachtet werden. Aktiv, wenn Anzeige läuft.
	Signalfarben automatisch vergeben		Alle Kurven dieser Anzeige werden je Graph nach dem Standardschema eingefärbt.
	Alles autoskalieren	<F5>	Alle Kurven dieser Anzeige werden je Graph und Y-Achse automatisch skaliert.
	Manuelle Skalierung wiederherstellen		Manuelle Skalierungseinstellungen werden, wo definiert, nach Autoskalierung oder Zoomen wiederhergestellt. Aktiv, wenn manuelle Skalierung definiert wurde.
	Eine Stufe auszoomen	<F3>	Nur aktiv in gezoomter Darstellung. Auf letzte Zoomstufe zurückschalten (verkleinern).
	Alles auszoomen	<F4>	Nur aktiv in gezoomter Darstellung. Auf ursprüngliche (automatische) Darstellung zurückschalten.
	Vorschubrichtung		Änderung der Vorschubrichtung durch Auswahl im Pull-down-Menü möglich.

13.1.2 Signal auswählen

Signale können per Drag & Drop bei gedrückter <Alt>-Taste von einem Konnektor in das ibaPDA Express-Fenster gezogen werden.

Sie können wahlweise:

- ❑ Ein Signal in getrennten Signalstreifen anzeigen.
  Dann ziehen Sie das Signal auf die X-Achse, es wird ein neuer Streifen angelegt.
- ❑ Ein Signal in einen vorhandenen Streifen legen.
  Dann ziehen Sie das Signal in den Streifen, es wird eine weitere Y-Achse angelegt.
- ❑ Ein Signal an eine vorhandene Y-Achse anlegen (gleiche Skalierung mit einem anderen Signal).
  Dann ziehen Sie das Signal auf diese Y-Achse.

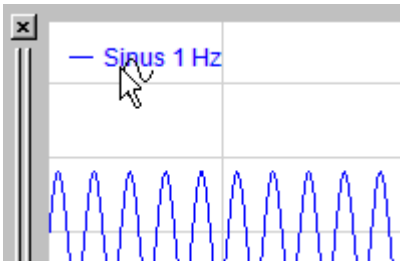
Die neuen Signale in einem Fenster erhalten automatisch eine neue Farbe zugewiesen. Im linken oberen Bereich des Streifens sind die jeweiligen Signalnamen mit derselben Farbe angeordnet. Signale mit gemeinsamer Achse sind mit einem Bindestrich verbunden.

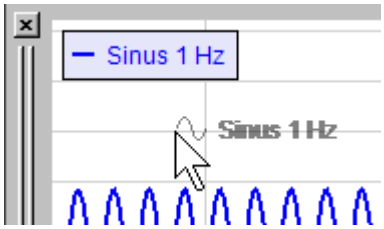
13.1.3 Signal verschieben

Signale können zwischen den Graphen, auch über Fenstergrenzen hinaus, hin- und hergeschoben werden. Dies bedeutet, dass ein Signal aus einem Graphen in einen anderen Graphen mit einem bereits existierenden Signal gezogen werden kann. Eine Unterscheidung der Signale erfolgt mittels automatischer Farbvergabe.

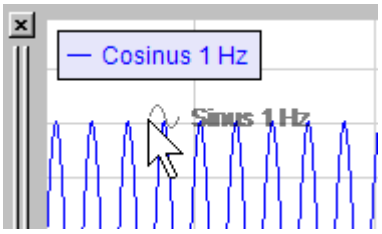
Vorgehen

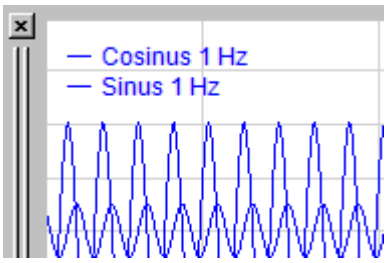
1. Gehen Sie mit dem Mauszeiger auf den Namen (Legende) des Signals, das verschoben werden soll. Der Mauszeiger zeigt mit einer Wellenlinie an, dass er das Signal erfasst hat.





2. Ziehen Sie mit gedrückter Maustaste das Signal in den anderen Graphen herüber, um dieses dort in einem freien Bereich fallen zu lassen.





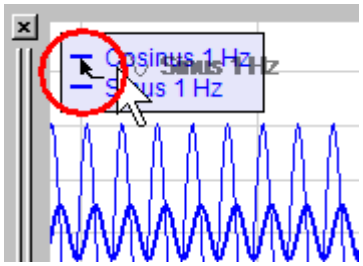
Ergebnis

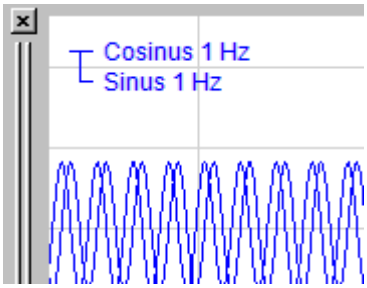
Es sind zwei Signale mit getrennter Y-Achse entstanden.

Anmerkung

Lassen Sie in Schritt 2 das Signal nicht los, sondern ziehen dieses bereits auf das vorhandene Signal, bis ein kleiner schwarzer Pfeil erscheint. Dann wird das Signal derselben Y-Achse zugeordnet.

Bei binären Signalen bestimmen Sie damit auch die Reihenfolge der Signale. Binäre Signale werden untereinander dargestellt. Je nachdem, ob der kleine schwarze Pfeil oben oder – wie unten – dargestellt am Signal andockt, wird das Binärsignal darüber oder darunter dargestellt.





Ergebnis

Es sind zwei Signale in einer gemeinsamen Y-Achse entstanden.

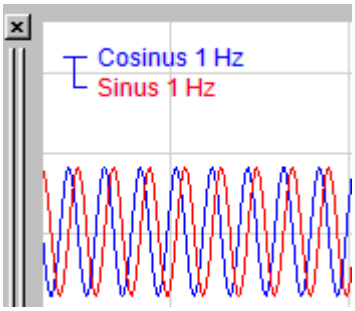
13.1.4 Signale farblich kennzeichnen

Sie können die Signale auf unterschiedliche Weise farblich kennzeichnen:

- ❑ Automatisch
- ❑ Manuelle Einstellung

Vorgehen

➲ Drücken Sie den Button <Signalfarben automatisch vergeben>, um den Signalen automatisch Farben zuzuordnen.



13.1.5 Signal aus der Anzeige entfernen

Vorgehen

1. Positionieren Sie den Mauszeiger im Graphen auf dem Namen (Legende) des Signals, das entfernt werden soll.
2. Klicken Sie mit der rechten Maustaste. Das Kontextmenü wird angezeigt.
3. Wählen Sie „Signal entfernen".

Hinweis

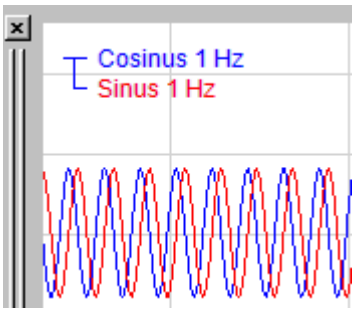
Mit dem Entfernen der Y-Achse werden alle Signale entfernt, die dieser Achse zugeordnet sind.

13.1.6 Graphen aus der Anzeige entfernen

Für das Entfernen eines Graphen stehen verschiedene Möglichkeiten zur Verfügung.

Vorgehen

1. Klicken Sie auf das kleine Kreuz links oben am Kopfbalken.



oder

2. Klicken Sie mit der rechten Maustaste in einem freien Bereich des Graphen. Das Kontextmenü wird angezeigt.
3. Wählen Sie „Graph entfernen".

13.1.7 Achsen skalieren

13.1.7.1 Autoskalierung

Um ein Signal in seiner ganzen Amplitude in einem Graphen darzustellen, empfiehlt sich die Funktion der Autoskalierung. Alle Signale bzw. alle Y-Achsen des Graphen werden entsprechend der größten Amplituden skaliert.

Vorgehen

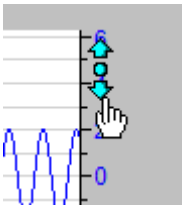
1. Drücken Sie die rechte Maustaste in den betreffenden Graphen. Das Kontextmenü wird angezeigt.
2. Wählen Sie „Autoskalierung".
3. Wollen Sie alle Graphen in einer Signalanzeige autoskalieren, dann drücken Sie die Taste <F5> oder wählen Sie den Button <Alles autoskalieren>.

13.1.7.2 Skalierung mit der Maus

Die Skalierung der Signale in Y-Richtung kann an den oberen Skalenenden der Y-Achse mit der Maus verändert werden.

Vorgehen

1. Bringen Sie den Mauszeiger soweit in die Nähe des Skalenendes bis blaue Pfeile erscheinen.
2. Halten Sie die Maus auf den Pfeil nach oben gedrückt: Skala wird gespreizt.
3. Halten Sie die Maus auf den Pfeil nach unten gedrückt: Skala wird gestaucht.
4. Halten Sie die Maus auf den Punkt zwischen den Pfeilen gedrückt: Eine Autoskalierung erfolgt.



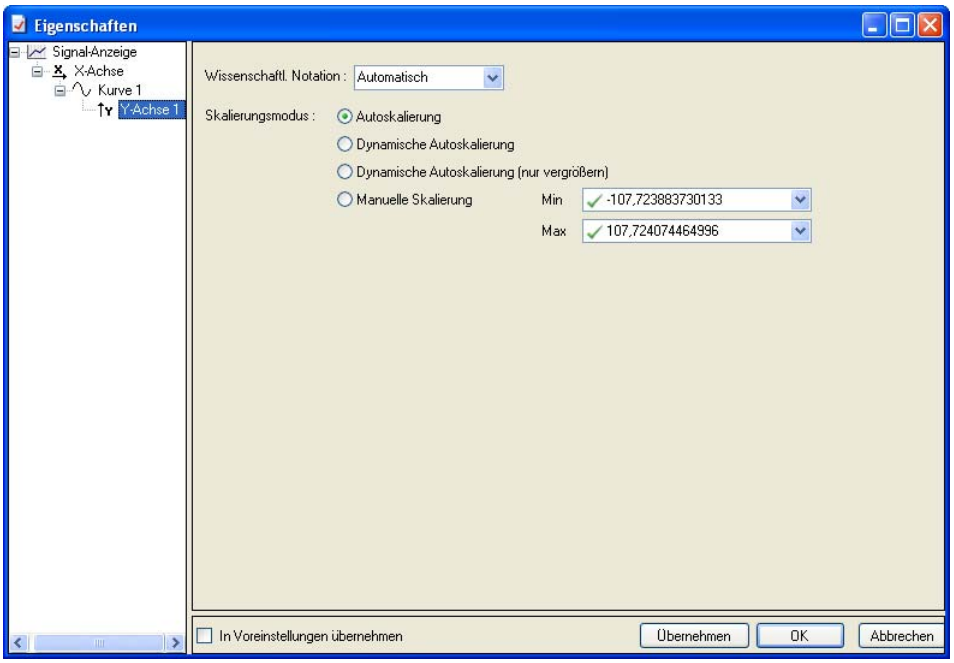
Tipp

Wenn Sie eine Maus mit Scroll-Rad verwenden, dann brauchen Sie nur den Mauszeiger auf die Skala zu legen. Mit dem Scroll-Rad können Sie dann die Skalierung verändern. Diese Funktionalität steht ebenfalls auf der X-Achse zur Verfügung.

13.1.7.3 Skalierung über die Anzeige-Einstellungen

Vorgehen

1. Klicken Sie mit der rechten Maustaste in den Bereich des gewünschten Graphs.
2. Wählen Sie „Eigenschaften".
   Der Dialog „Eigenschaften" wird angezeigt.
   Auf der Verzweigung „Y-Achse" kann eine manuelle Skalierung vorgegeben werden. Hat ein Graph mehrere Y-Achsen, so gibt es in dem Dialog für jede Y-Achse eine eigene Lasche.
3. Übernehmen Sie die Einstellungen mit <Übernehmen>.



Ergebnis

Alle Signale, die der entsprechenden Y-Achse zugeordnet sind, werden mit der gleichen Einstellung skaliert.

Anmerkung

Bei Anwahl der Option „In Voreinstellung übernehmen" wählen Sie die eingestellten Einstellungen als Standard.

13.1.8 Skalen verschieben

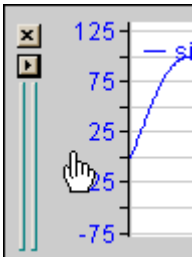
Im Pause-Modus der Anzeige kann die X-Achse in entsprechender Art und Weise verschoben werden.

Voraussetzung

Y-Achse: Keine Autoskalierung angewählt.

Vorgehen

1. Legen Sie den Mauszeiger auf die Y-Achse, bis das Handsymbol erscheint.
2. Drücken Sie die linke Maustaste, um die Skala nach oben/unten oder nach rechts/links zu verschieben.



13.1.9 Zoom-Funktion

Die Zoom-Funktion wirkt sowohl auf die X- als auch die Y-Richtung.

Wird in einem Graphen gezoomt, so werden alle anderen Graphen, die in derselben Signalanzeige liegen, mitgezoomt. Eine Signalanzeige kann immer nur eine Zeitbasis für alle enthaltenen Graphen beibehalten.

Bei laufender Anzeige wird mit dem Zoomen die Zeitbasis gedehnt und somit die Anzeige vergrößert. Die Signale laufen schneller durch, da die gleiche geometrische Länge der X-Achse auf weniger Zeiteinheiten umgerechnet wird.

Zoomen allgemein

1. Drücken Sie die <Umsch>-Taste und zoomen Sie gleichzeitig mit der Maus. Es wird nur die X-Achse gezoomt.
2. Mit dem Button 

oder der Taste <F4> wird auf die ursprüngliche, ungezoomte Darstellung zurückgeschaltet.

13.1.9.1 Einzoomen (vergößern)

Einzoomen ist überall in einem Streifen möglich. Im eingezoomten Zustand kann jederzeit die Skalierung in Y-Richtung verändert werden, ohne dass der gezoomte Ausschnitt der X-Achse beeinflusst wird.

Die Autoskalierung in Y-Richtung bezieht sich auf die Werte im gezoomten (= sichtbaren) Bereich.

Voraussetzung

Sie haben ausgezoomt.

Vorgehen

1. Ziehen Sie mit der linken Maustaste ein Rechteck auf, so dass der gewählte Bereich umschlossen wird.
2. Lassen Sie die Maus wieder los.

13.1.9.2 Auszoomen (verkleinern)

Voraussetzung

Sie haben eingezoomt.

Vorgehen

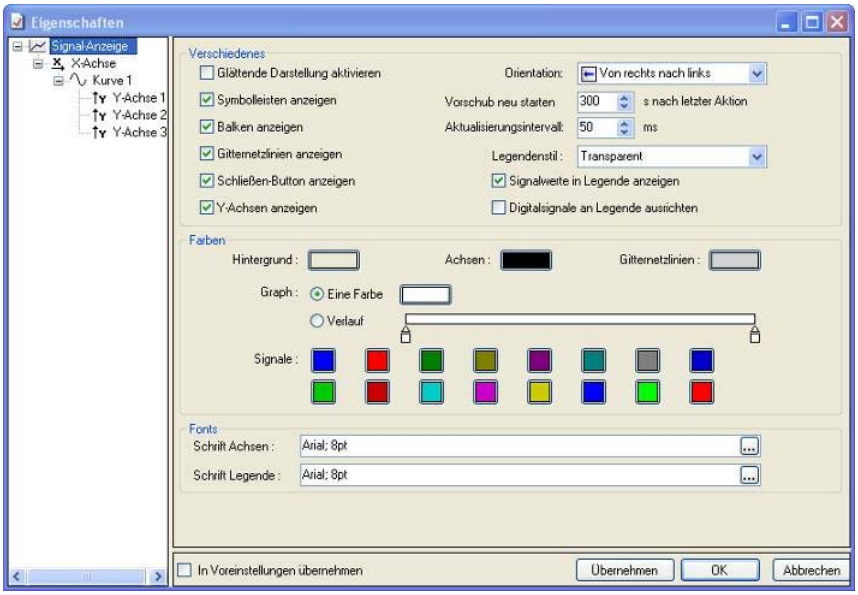
1. Drücken Sie den Button <Eine Stufe auszoomen> oder Taste <F3>, um stufenweise zu verkleinern.
   Dabei werden mit jeder Aktion alle vorangegangenen Zoomschritte nacheinander rückgängig gemacht.
2. Drücken Sie den Button <Alle Stufe auszoomen> oder der Taste <F4>, um auf die ursprüngliche, ungezoomte Darstellung zurückzugeschalten.

13.1.10 Signal-Anzeige-Eigenschaften

Im Signal-Anzeige-Eigenschaften-Dialog werden allgemeine Einstellungen für die Darstellung der Graphen vorgenommen.

Vorgehen

1. Klicken Sie mit der rechten Maustaste auf den Signalstreifen.
2. Klicken Sie mit der linken Maustaste auf „Eigenschaften...".
3. Wählen Sie links in der Struktur die „Signal-Anzeige".



13.1.10.1 Verschiedenes

Option	Erklärung
Glättende Darstellung aktivieren	Diese Option bewirkt, dass die Signal-/Graphlinien geglättet angezeigt werden.
Symbolleiste anzeigen	Diese Option bewirkt, dass die Symbolleiste angezeigt wird.
Signalwerte in Legende anzeigen	Diese Option bewirkt, dass die Signalwerte in der Legende angezeigt werden.
Balken anzeigen	Diese Option bewirkt, dass der zugehörige Balken zum Graphen eingblendet wird.
Transparente Legende	Diese Option bewirkt, dass die Legende transparent geschalten wird.
Vorschubrichtung	Es wird eine Vorschubrichtung eingestellt.
Vorschub neu starten	Einstellung einer bestimmten Zeit in Sekunden, welche nach der letzten Aktion den Vorschub neu startet.
Aktualisierungsintervall	Einstellung, wie oft die Anzeige aktualisiert werden soll.
Digitalsignale an Legende ausrichten	Diese Option bewirkt, dass die Digitalsignale an der Legende ausgerichtet werden.

13.1.10.2 Farben

In diesem Dialog wird das Farbschema für die Signal-Anzeige und die Stiftfarben für die Kurven verändert.

➲ Zum Verändern der Farben klicken Sie auf die jeweilige Farbfläche. Wählen Sie die gewünschte Farbe aus einer Palette aus.

❑ Hintergrund, Achsen, Gitternetzlinien:

➲ Zum Verändern der Farben klicken Sie auf die jeweilige Farbfläche. Wählen Sie die gewünschte Farbe aus der Palette aus.

❑ Graph:
Hintergrundfarbe in den Signalstreifen einheitlich oder mit Farbverlauf.

➲ Doppelklicken Sie auf die kleinen Quadrate an den Enden des Farbstrahls und wählen Sie die Farbe aus der Palette aus.
Bei Bedarf lassen sich mittels Doppelklick auf den Farbstrahl noch weitere Farbreiter hinzufügen und einfärben, die zudem noch verschiebbar sind. Um einen Farbreiter zu löschen, markieren Sie den Reiter mit einem Mausklick (schwarze Pfeilspitze) und drücken Sie die <DEL>-Taste.

❑ Signale:
Mit diesen Stiftfarben werden die 16 Kurvenfarben definiert, die für die Kurven-Anzeige zur Verfügung stehen. Anhand dieser 16 Farben nimmt das Programm die automatische Farbgebung der Kurven vor. In der hier sichtbaren Reihenfolge (zeilenweise von oben nach unten) werden die Stiftfarben auch in der Tabelle der Signaldefinitionen im Signalraster angeboten.

13.1.10.3 Schriftarten

Für die Beschriftung der Achsen und der Legende (Signalnamen) werden die Schriftarten festgelegt. Den Dialog zum Ändern der Schriftart, erreichen Sie über den Browserbutton <…> am Ende der Zeile.

13.1.10.4 Signale

Wird dieser Dialog in einem Graphen bzw. für eine vorhandene Signal-Anzeige aufgerufen, in der bereits Signale angezeigt werden, werden die Signale mit ihren aktuellen Einstellungen, inkl. der Farben aufgelistet.

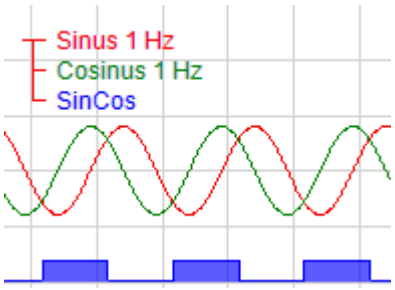


Abbildung 116: Signalanzeige

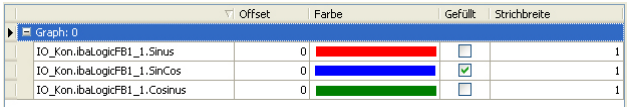


Abbildung 117: Graphische Signaleinstellungen

In den Zellen der Tabellenspalte „Farbe" wird für jedes Signal die Farbe anhand einer Auswahlliste gewählt.

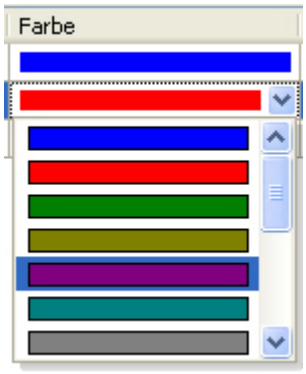


Abbildung 118: Farb-Auswahlliste

13.1.10.5 X-Achse

Zeitbereich

Anstelle einer automatischen Skalierung kann ein fester Zeitbereich in Sekunden vorgegeben werden, der in der Anzeige dargestellt wird. Damit beeinflussen Sie die Geschwindigkeit und die Dehnung des Signals in X-Richtung in der Darstellung.

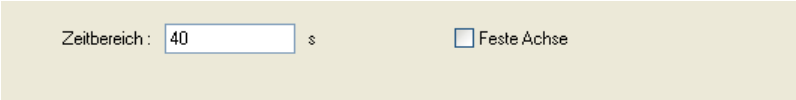


Abbildung 119: X-Achse: Eigenschaften

Feste Achse

Normalerweise läuft die Zeitachse mit dem Signal mit, so dass neue Messwerte stets am Rand des Graphen in die Anzeige einlaufen. Mit der Option „Feste Achse" wird die Zeitachse vom aktuellen Zeitpunkt an für die eingestellte Dauer (Zeitbereich) fixiert und die Messwerte werden in den leeren Graphen hineingeschrieben. Ist der Graph gefüllt, dann wird der sich anschließende (leere) Zeitbereich dargestellt und Messwerte weitergeschrieben.

13.1.10.6 Y-Achse

Wenn in einem Graphen mehr als eine Y-Achse angelegt ist, dann werden im Einstelldialog entsprechend viele Register „Y-Achse #" angeboten. So können für alle Y-Achsen individuelle Einstellungen vorgenommen werden.

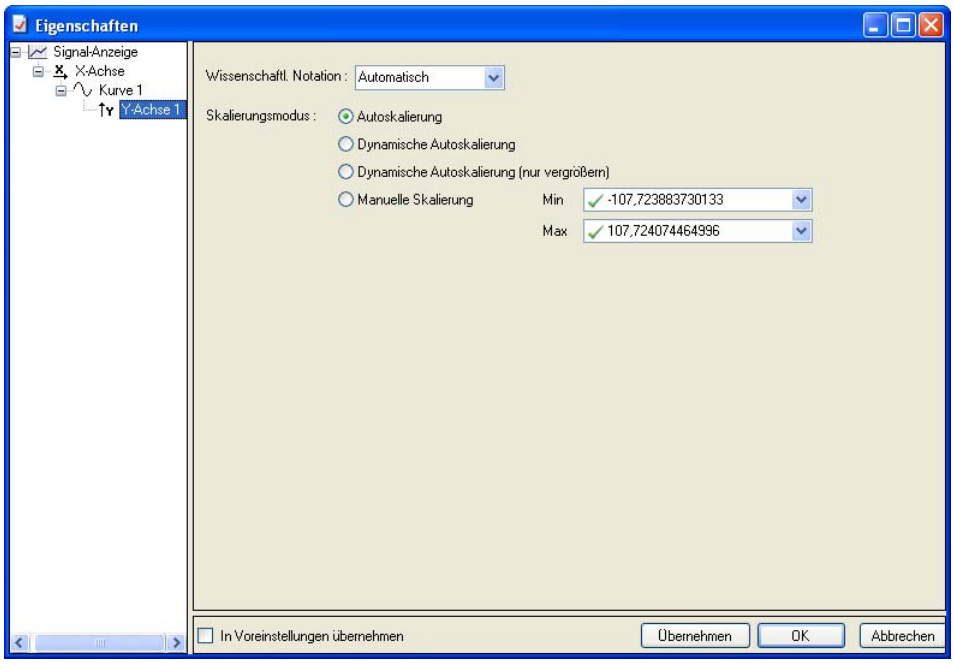


Abbildung 120: ibaPDA Express: Anzeige-Einstellungen

13.1.10.7 Wissenschaftliche Notation

- ❑ „Automatisch"
- ❑ „Immer"
- ❑ „Nie"

Option	Erklärung
Automatisch	Abhängig von der Größenordnung der Skalenwerte (Anzahl der Vor- bzw. Nachkommastellen) erfolgt die Skalenbeschriftung in wissenschaftlicher Notation (10er-Potenzen) oder nicht.
Immer	Skalenwerte in 10er-Potenzen
Nie	Skalenwerte immer mit Vor- und Nachkommastellen

13.1.10.8 Skalierungsmodus

- ❑ „Autoskalierung"
- ❑ „Dynamische Autoskalierung"
- ❑ „Dynamische Autoskalierung" (nur vergrößern)
- ❑ „Manuelle Skalierung"

Option	Erklärung
Autoskalierung	Standardeinstellung; beim Darstellen eines oder mehrerer Signale wird die Y-Achse des Streifens nach dem kleinsten und größten aller vorkommenden Werte einmalig (beim Hineinziehen eines Signals) skaliert.
Dynamische Autoskalierung	Wenn Sie diese Option aktivieren, dann wird die Skalierung stets den höchsten Signalamplituden angepasst. Verlassen die Amplituden den Signalstreifen wieder, wird die Skalierung auch wieder verkleinert.
Dynamische Autoskalierung (nur vergrößern)	Wenn Sie diese Option aktivieren, dann wird die Skalierung stets den höchsten Signalamplituden angepasst. Verlassen die Amplituden den Signalstreifen wieder, bleibt die Skalierung trotzdem erhalten.
Manuelle Skalierung	Bei Wahl dieser Option können der Skalenanfangs- (Min) und der Skalenendwert (Max) manuell vorgegeben werden. (Nur sichtbar, wenn Dialog über Kontextmenü im Signalstreifen geöffnet wird; nicht per Voreinstellungen.)

13.1.11 Erweiterte Funktionalität

Die erweiterte Funktionalität ist über das Symbol in der Titelleiste per Kontextmenü zuschaltbar.

Folgende Funktionen stehen zur Verfügung:

- „Symbolleiste"
- „Signalbaum"
- „Vollbildansicht"

Durch Anwahl der Menüs erweitert sich die Anzeige des ibaPDA Express.

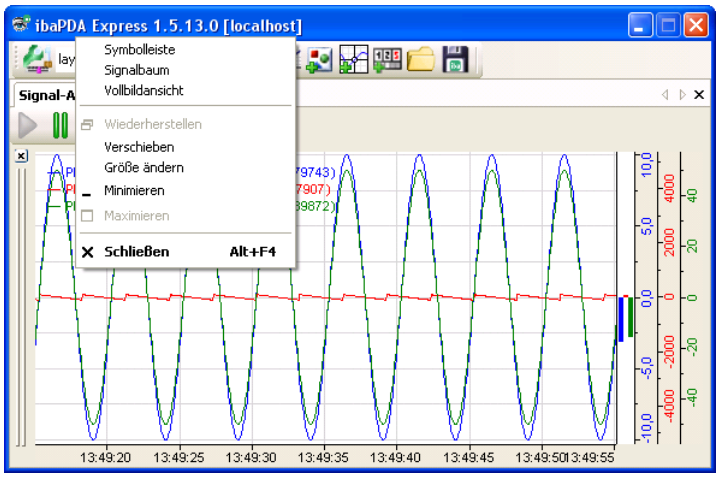


Abbildung 121: Kontextmenü „ibaPDA Express"

Symbolleiste

Zeigt eine Symbolleiste mit folgenden Elementen:

Symbol	Name	Erklärung
	Voreinstellungen verändern	Öffnet das Eigenschaften-Menü.
	Echtzeit	Start/Stop-Funktion aller Streifen.
	Signal-Anzeige hinzufügen	Hinzufügen einer weiteren Signal-Anzeige Die Signal-Anzeigen können beliebig angeordnet werden in dem man den Reiter „Signal-Anzeige" mit der linken Maustaste festhält und verschiebt. Es werden „Andock"-Punkte sichtbar.
	QPanel hinzufügen Scrope view hinzufügen Digital meter hinzufügen	Zusatzfunktionen QPanel, Scope view, Digital meter in den Express hinzufügen.
	Ansicht laden	Öffnen einer ibaPDA Express-Konfiguration als XML-File.
	Ansicht speichern	Abspeichern einer ibaPDA Express-Konfiguration als XML-File.

Signalbaum

Zeigt eine Baumansicht aller im Programm enthaltenen Variablen an. Diese können dann per Drag & Drop oder durch Doppelklick in eine Signal-Anzeige gebracht werden.

Vollbildansicht

ibaPDA Express wird im Vollbild-Modus angezeigt. Mit der Taste <F10> gelangt man wieder zurück.

Andere Dokumentation

Für die Beschreibung wird auf die entsprechende Zusatz-Dokumentation des ibaPDA-Systems verwiesen.

13.2 Zeitverhalten

ibaLogic bietet plattformabhängig ein deterministisches Zeitverhalten (Echtzeitverhalten).

Plattformen:

- Windows XP/7:
  nicht deterministisch, relativ stabile Zykluszeiten bei Task-Zeiten von ≥ 5 ms.
- PADU-S-IT:
  deterministisch, sehr stabile Zykluszeit bei Task-Zeiten von ≥ 1 ms.

Die Tasks von ibaLogic liegen auf einem Grundraster von minimal 1 ms, dieses richtet sich nach der konfigurierten Interrupt-Zeitbasis, die unter Menü „Extras - I/O-Konfigurator" eingestellt werden kann. Dies ist das minimale Taskintervall.

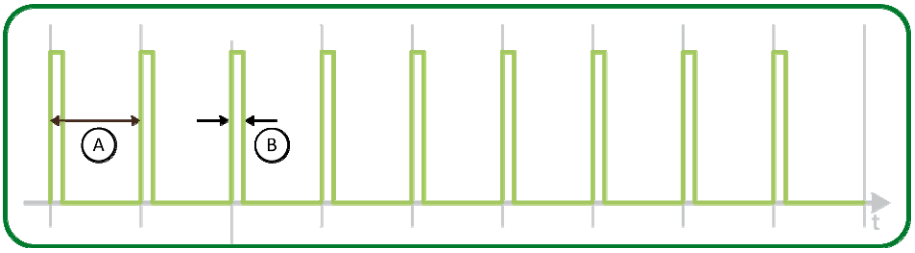
Schnellere Tasks sind nicht möglich. Möglich ist, dass bestimmte iba-Baugruppen Messdaten in 50 µs aufnehmen und diese als Array von Werten (Pakete) an ibaLogic weitergeben. ibaLogic verarbeitet dann die Werte in dem unterlagerten Takt. Weitere Informationen siehe „Buffered Mode , Seite 195".

Für das Task-Handling überprüft ibaLogic zum eingestellten Grundraster, welche Tasks zum Bearbeiten anstehen und trägt diese am Ende einer internen Taskliste ein. Diese Taskliste wird zyklisch von oben nach unten durchgerechnet und berechnete Tasks werden wieder aus der Liste entfernt.

Zu beachten ist auch, dass der eingestellte Grundtakt auch der Takt ist, in dem Eingaben gelesen und Ausgaben geschrieben werden können.

ibaLogic kennt nur den Intervall-Task.

Der Intervall-Task wird entsprechend dem eingestellten Intervall gestartet. Das angebundene Programm hat seine entsprechende Berechnungszeit.



A Intervall

B Berechnungszeit

Abbildung 122: Intervall-Task

13.2.1 Berechnungszeit

Wichtig für die Betrachtung des Zeitverhaltens sind auch die Programm-Berechnungszeiten der einzelnen Programme im gesamten Anwender-Projekt.

ibaLogic stellt dem Anwender für den einzelnen Intervall-Task die Berechnungszeit zur Auslastungskontrolle als Zahlenwert und Balken zur Verfügung.

Hier am Beispiel eines 1 ms Intervall-Tasks, bedeutet die Prozentangabe, dass 34,36 % von 1 ms benötigt werden. D. h. diese Angabe ist prozentual zur parametrierten Taskzeit zu sehen. 34,36 % bei 1 ms Taskzeit wären damit 0,3436 ms Programm-Rechenzeit.

13.2.2 Turbomodus

Um zu vermeiden, dass ibaLogic von Windows zeitweilig verdrängt wird, kann ibaLogic in Multicore-Systemen einen Prozessorkern exklusiv belegen.

Hinweis

Um einen möglichst deterministischen Ablauf zu gewährleisten, empfiehlt iba:

- Bei Task-Zeiten < 20 ms:
  Verwenden Sie eine iba-Interrupt-Quelle
  (ibaFOB-Karte o.ä.)
- Bei Task-Zeiten < 5 ms:
  Verwenden Sie den Turbomodus

Tipp

Das Zeitverhalten kann auch über die Compiler-Optionen im Menü „Extras - Optionen - Laufzeitoptionen" beeinflusst werden.

- „Größe:"
  Default-Einstellung, Interpretermodus (UCODE)
  mit ST-Breakpoints möglich
- „Beides:"
  Interpretermodus und nativer Code gemischt,
  keine ST-Breakpoints möglich
- „Geschwindigkeit:"
  nur nativer Code, keine Breakpoints, nicht alle LREAL-Funktionen
  verfügbar (z. B. alle Exponentialfunktionen)

ibaLogic unterscheidet weiterhin den Modus „Messung" und den Modus „Soft-SPS". Einstellungen siehe „Allgemeine Einstellungen , Seite 182".

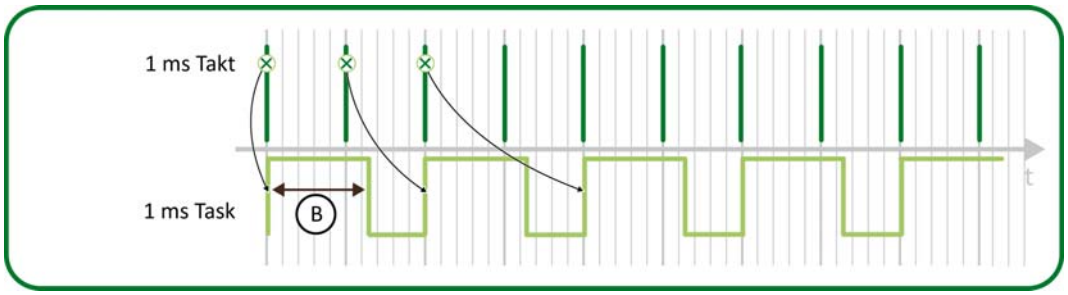
13.2.3 Messung

Diese Betriebsart stellt sicher, dass ibaLogic kein Eingangssample verliert. Dies gilt auch dann, falls einzelne Tasks innerhalb ibaLogic verdrängt werden sollten. Das Ablaufsystem von ibaLogic stellt hier sicher, dass die Daten äquidistant im eingestellten Taskintervall zur Verfügung stehen. Bei Taskverdrängungen werden Zyklen nachgeholt. Es kann demzufolge bei zeitlich begrenzter Verdrängung dazu kommen, dass ibaLogic zeitweise nur noch Werte der „Vergangenheit" berechnet.

Jedoch ist immer sichergestellt, dass z. B. für FFT-Analysen äquidistant korrekte Werte zur Verfügung stehen. Eine permanente Verdrängung führt zu einem Pufferüberlauf. Das ist eine inakzeptable Projektierung.

Für das Einlesen von Hardware-Eingangs-Signalen muss man noch eine Betrachtung bezüglich der möglichen Betriebsart durchführen.

Im Modus „Messung" wird der Hardware-Eingangs-Signal-Status entsprechend dem eingestellten Intervall des Tasks gepuffert. Der nächstmögliche Programmstart arbeitet dann mit dem ältesten gepufferten Wert. D. h., dass der Modus „Soft-SPS" und der Modus „Messung" gleich arbeiten, wenn die Programm-Bearbeitungszeiten < der Intervallzeit ist. Bei größeren Bearbeitungszeiten kommt es im Modus „Messung" zu einem Pufferüberlauf der Messwerte.



B Berechnungszeit

Abbildung 123: Pufferüberlauf – Verschiebungen

Beispiel: Der dunkelgrüne 1 ms Takt speichert jeweils den Wert mit dem bei Task-Beginn (hellgrün) gearbeitet wird. Der schwarze Pfeil gibt an, mit welchem Messwert der Task arbeitet und wie sich der Pufferüberlauf aufbaut. Die Berechnungszeit einer Task beträgt aber mehr als 1 ms, daher kommt es zu Verschiebungen.

13.2.4 Soft-SPS

In dieser Betriebsart, die für Regelungs- und Steuerungsaufgaben geeignet ist, stellt ibaLogic sicher, dass nur die jüngsten Signalzustände verarbeitet werden. Im Gegensatz zur Betriebsart „Messung" kommt es hier nicht darauf an, ob Samples verloren gehen oder nicht. Es ist ganz im Gegenteil erwünscht, nur möglichst aktuelle Daten - also aus dem letzten I/O-Transferzyklus - zu erhalten.

Die Eingangsressourcen werden bei jedem neuen Zyklus vor dem ersten Task eingelesen. Die Alterungszeit der Ressourcen ist durch den einstellbaren ibaLogic-Grundtakt bestimmt. Ist dieser Grundtakt z. B. 10 ms und der erste Task mit Zyklus 50 ms parametriert, so findet der erste Task Eingangsdaten vor, die garantiert nicht älter als 10 ms sind. Sie können jedoch jünger sein.

Ausgangswerte werden in beiden Modi von jedem Task in den Zyklus nach dem Ende der erforderlichen Task-Berechnungszeit geschrieben, sofern Ausgangsressourcen im Plan angeschlossen sind.

Im Modus „Soft-SPS" wird der aktuelle Hardware-Eingangs-Signal-Status zum Start der Task eingelesen und verarbeitet.

13.2.5 Zeitbetrachtungen bei mehreren Tasks

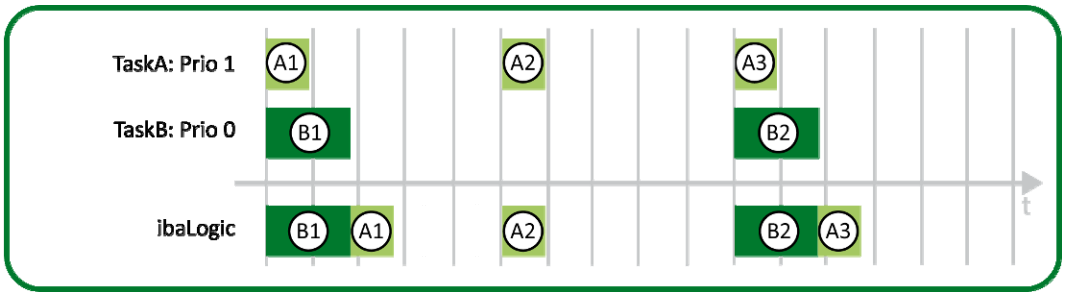


Abbildung 124: Berechnung ohne Überlauf – 2 Tasks mit unterschiedlicher Intervallzeit und Priorität

Die oberen 2 Reihen stellen die einzelnen Tasks dar in der theoretischen Berechnungsabfolge, wenn diese alleine laufen würden. Die Zahlen sind dabei ein Zähler für den Anstoß der Tasks (1. Anstoß, 2. Anstoß…).

In der obersten Reihe ist eine Intervalltask A mit einer Intervallzeit von 5 ms dargestellt. Die Priorität ist 1, d. h. niederprior gegenüber der 10 ms TaskB mit Priorität 0 (zweite Reihe). Die Breite des Balkens (Impulses) entspricht der Rechenzeit des Tasks des zugehörigen Programms. Der Hintergrund stellt ein Taktraster dar. Der Impuls beginnt immer wieder zur eingestellten Intervall-Zeit.

Praktisch werden die Tasks aber „seriell" abgearbeitet. Dies ergibt die unterste Reihe. ibaLogic sieht zum Startzeitpunkt, welche Tasks berechnet werden müssen und berechnet diese nacheinander, entsprechend der eingetragenen Priorität. Erst TaskB da diese die höhere Priorität hat, dann nach deren Berechnungszeit der TaskA…

Zur Verdeutlichung des Sachverhalts sind die dargestellten Programmberechnungszeiten sehr groß angenommen worden. Real liegen die Berechnungszeiten vorwiegend im µs-Bereich, so dass beispielsweise 20 Tasks ohne Probleme in 5 ms berechnet werden können (Erfahrungswert).

13.2.6 Worst-Case-Betrachtungen

Wird eine längere Programm-Rechenzeit von TaskA und TaksB angenommen, dann wird eine Verdrängung erzeugt. Verdrängung heißt, dass der Task nicht mehr zum erwarteten Zeitpunkt gestartet werden kann, da noch ein anderes Programm berechnet wird. Der höherpriore Task wird zum richtigen Zeitpunkt gestartet.

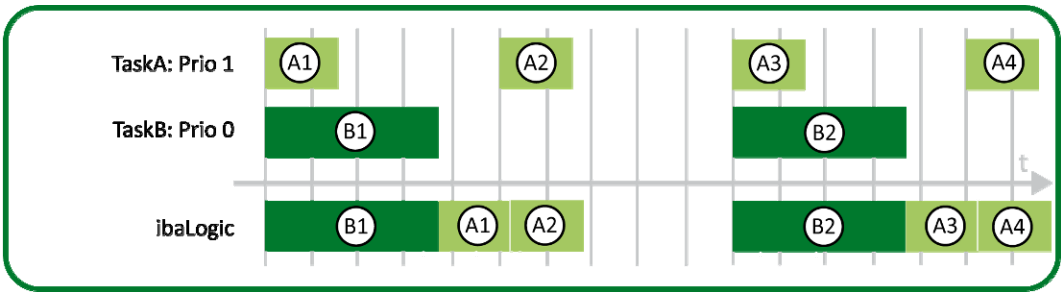


Abbildung 125: Taskberechnung mit Verdrängung

Die Berechnungszeiten sind so gewählt, dass beide Tasks zusammen größer als 5 ms sind und damit der TaskA nicht exakt zum vorgesehenen Intervall gestartet werden kann. Wenn ein 1-ms-Grundtakt eingestellt ist, so wird zu jedem Takt geprüft, ob ein Task angestoßen werden muss. Im Beispiel hier der TaskA (5 ms, Priorität 1) und der TaskB (10 ms, Priorität 0). Diese werden entsprechend ihrer Priorität in eine interne Liste eingetragen und dann gestartet.

13.2.7 Erläuterung zum obigen Fall

In der Abbildung „Taskberechnung mit Verdrängung - Auszug" sind die Jobs der internen Liste dargestellt.

ibaLogic sieht zu Beginn, dass TaskA (5 ms, Priorität 1) und TaskB (10 ms, Priorität 0) abgearbeitet werden müssen und trägt sie entsprechend ihrer Priorität in die interne Job-Liste ein. Gestartete Tasks werden aus der Job-Liste entfernt, neue werden hinzugefügt und so ergibt sich die obige Abbildung.

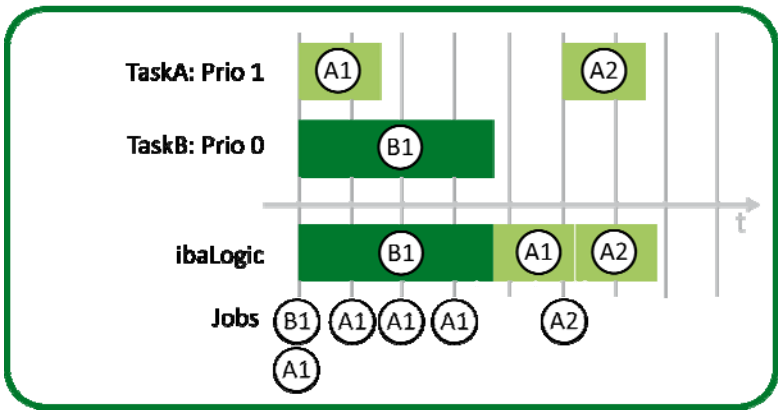


Abbildung 126: Taskberechnung mit Verdrängung – Auszug

13.2.8 Taskberechnung mit Verdrängung

Angenommen der TaskA wird mit einer Intervall-Zeit von 2 ms parametriert (bei gleichen Programmberechnungszeiten), dann gehen einige **Zyklen verloren**.

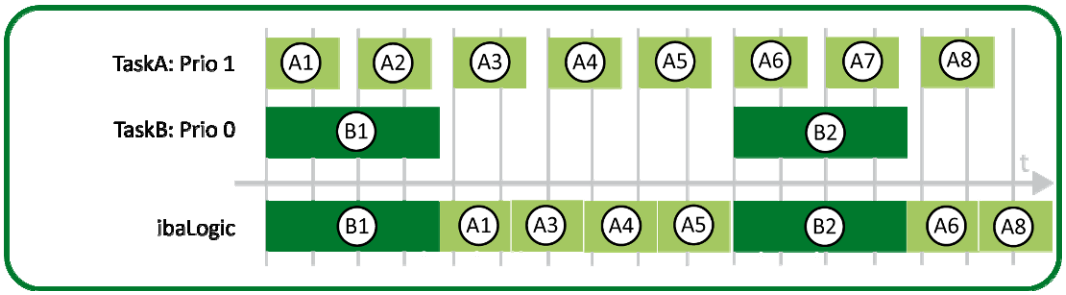


Abbildung 127: Taskberechnung mit Verdrängung

Ein anderes Bild ergibt sich, wenn die Prioritäten vertauscht werden.

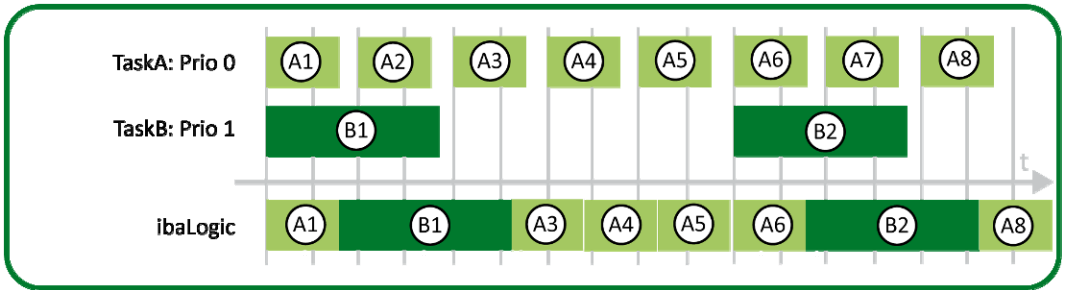


Abbildung 128: Taskberechnung mit Verdrängung (umgekehrte Priorität)

Eine weitere Betrachtung ist der Modus „Soft-SPS" und der Modus „Messung".

Im Modus „Soft-SPS", werden die Hardware-Eingänge immer am Beginn der Tasks gelesen (x-Punkt in der folgenden Abbildung).

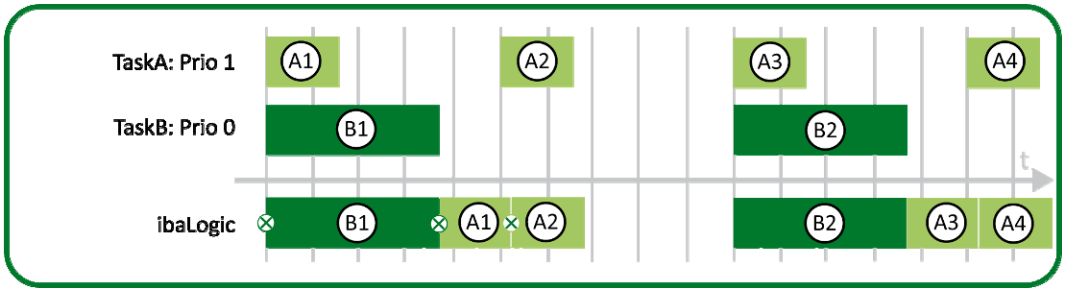


Abbildung 129: Hardware-Eingänge Modus „Soft-SPS"

Anders sieht es beim Modus „Messung" aus.

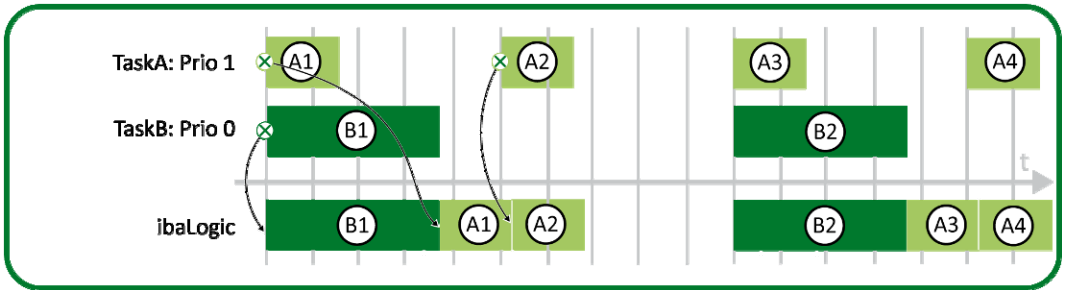


Abbildung 130: Hardware-Eingänge Modus „Messung"

Hier werden die Eingangs-Signale zeitrichtig gepuffert, aber im Verdrängungsfall verzögert berechnet.

Wichtiger Hinweis

Generell kommt es zu einer Verdrängung, wenn die Summe der Programm-Rechenzeiten die kleinste verwendete Intervallzeit überschreitet. Wenn diese Verdrängung permanent ist, dann kommt es zu einem Pufferüberlauf. Programm und Rechner arbeiten nicht mehr der Anforderung entsprechend. Bei einer zeitweiligen Verdrängung hängt dies von der jeweiligen Anwendung ab, ob dies toleriert werden kann.

Die Hardwareausgänge werden im nächsten Grund-Takt nach Beendigung der Rechenzeit geschrieben. Deswegen kann es sinnvoll sein, den Grundtakt schneller als das Task-Intervall einzustellen. Wenn z. B. die Berechnungszeit 50 µs, die Task-Intervallzeit 5 ms und der Grund-Takt 1 ms beträgt, dann werden die Ausgänge nach 1 ms geschrieben.

13.3 Fehlersuche

Folgende Fehler können auftreten:

- Programmfehler
- Kompilierungsfehler

13.3.1 Programmfehler

Häufige Fehler in Programmen:

- Fehler in Anwenderfunktionsbausteinen
- Division durch 0
- Fehlerhafte Signalverläufe
- Fehlerhafte Berechnungsreihenfolge

13.3.1.1 Fehler in Anwenderfunktionsbausteinen

Um logische Fehler finden zu können, bietet ibaLogic-V4 Ihnen die Möglichkeit in Funktionsblöcken sogenannte „Breakpoints" zu setzen, um die Ausführung ihres ST-Codes zu überprüfen. Weitere Informationen siehe „Structured Text-Editor , Seite 126".

13.3.1.2 Division durch 0

Sollten Sie in ihrem Ereignisfenster folgende Nachricht vorfinden, ist eine Division durch 0 in Ihrem Programm aufgetreten.

"Exception: OnlineServer: PMAC Status: Division by Zero in *program.functionblock*, Offset 0x0022, Stack 0x0001"

In der Meldung wird Ihnen der Ort an dem die Division durch 0 aufgetreten ist, angezeigt. In dem obigen Beispiel ist der Fehler im Funktionsbaustein „*functionblock*" im Programm „*program*" aufgetreten.

13.3.1.3 Fehlerhafte Signalverläufe

Um berechnete Werte überprüfen zu können, steht Ihnen in ibaLogic-V4 das Werkzeug ibaPDA Express zur Verfügung. Mithilfe dieses Werkzeugs können Sie die Signalverläufe in Echtzeit anzeigen lassen und so verfolgen, ob Ihr Baustein bei verschiedenen Eingangsparametern die erwarteten Ausgangswerte liefert.

13.3.1.4 Berechnungsreihenfolge

Wenn trotz fehlerfreier Funktionsbausteine und Makroblöcken die Berechnung nicht so läuft, wie Sie es erwarten, ist es möglich, dass die Berechnungsreihenfolge das Problem ist.

Um zu überprüfen, welcher Ihrer Bausteine zuerst berechnet wird, können Sie sich die Berechnungsreihenfolge des entsprechenden Programms ansehen und so eventuelle Fehler in der Reihenfolge aufdecken.

Weitere Informationen siehe „Berechnungsreihenfolge , Seite 59".

Sobald Ihr Programm Rückkopplungen enthält, ist es notwendig zu wissen, welcher Baustein in der Berechnung an erster bzw. letzter Stelle steht.

13.3.2 Kompilierungsfehler

Obwohl die ST-Syntax in den Anwender-FBs vor dem Kompilieren durch den Bausteingenerator geprüft wird, kann es in manchen Fällen vorkommen, dass die Kompilierung des generierten IL-Codes fehlschlägt.

In so einem Fall erhalten Sie im Ereignisfenster eine Fehlermeldung, die Sie darauf hinweist.

Ist in Ihrem Ereignisfenster eine Meldung zu sehen, die wie folgt aussieht, scrollen Sie bitte mit der Laufleiste am rechten Rand soweit hoch, dass Sie die erste Fehlermeldung sehen können.

Beispiel:

```
 1 [01.03.2010 14:19:14] [<Computername>] [ibaLogicClient] Info:
 2 Generation started...
 3 [01.03.2010 14:19:14] [<Computername>] [ibaLogicClient] Info:
 4 Compilation started...
 5 [01.03.2010 14:19:14] [<Computername>] [ibaLogicServer] Info: TIMER
 6 Generation: Ticks since start of IL generation: 31, that is 0,03
   seconds
 7 [01.03.2010 14:19:15] [<Computername>] [ibaLogicClient] Exception:
 8 IL compilation failed: Compilation ended with errors.
 9 [01.03.2010 14:19:15] [<Computername>] [ibaLogicServer] Info:
10 Building resource C:\Documents and
11 Settings\<Benutzername>\Application
   Data\ibaLogic\NewWorkspace\NewProject\$ENV$\Resource\Resource.MAK.
12 C:\DOCUMENTS AND SETTINGS\<Benutzername>\APPLICATION
13 DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\CustomTypes.typ
14 C:\DOCUMENTS AND SETTINGS\<Benutzername>\APPLICATION
15 DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\FB_STRINGOUT.POE(3,5,2): E:
16 S3023: Invalid operand type for this operation.
17 1 error(s), 0 warning(s) –
18 C:\DOCUMENTS AND SETTINGS\<Benutzername>\APPLICATION
19 DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\FB_STRINGOUT.POE.
20 C:\Documents and Settings\<Benutzername>\Application
21 Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE(2,9,14): E:
22 S3026: Undeclared identifier.
23 C:\Documents and Settings\<Benutzername>\Application
24 Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE(3,2,6): E:
25 S3005: This is not a function block instance.
26 3 error(s), 0 warning(s) - C:\Documents and
27 Settings\<Benutzername>\Application
28 Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE.
29 3 error(s), 0 warning(s).
```

Wenn die Kompilierung fehlschlägt, fangen Sie immer mit der ersten Fehlermeldung an, die Ihnen im Ereignisfenster gemeldet wird.

Um den Fehler finden zu können gehen sie folgendermaßen vor:

- Vergrößern Sie das Ereignisfenster, so dass die Ereignisse ab „**Compilation started**" angezeigt werden.
- Suchen Sie die erste Meldung, die einen Fehler enthält. Die weiteren Meldungen sind möglicherweise Folgefehler. In dem Beispiel oben ist es die Meldung **„C:\DOCUMENTS AND SETTINGS\<Benutzername>\APPLICATION DATA\IBALOGIC\NEWWORKSPACE \NEWPROJECT\FB_STRINGOUT.POE(3,5,2): E: S3023: Invalid operand type for this operation"**
- Kopieren Sie den Pfad, in dem der fehlerhafte Baustein liegt, in die Zwischenablage und fügen Sie ihn in die Adresszeile des Explorers ein. Im Beispiel oben ist das **„C:\DOCUMENTS AND SETTINGS\<Benutzername>\APPLICATION DATA\IBALOGIC \NEWWORKSPACE\NEWPROJECT"**
- Dort finden Sie den fehlerhaften Baustein, in unserem Fall „FB_STRINGOUT.POE". Öffnen Sie diesen mit einem ASCII-Editor, der die Zeilennummern anzeigt, z. B. NotePad++.
  Sie sehen z. B. folgenden Programmcode (Beispiel oben)

```
 1 FUNCTION_BLOCK FB_StringOUT
 2         VAR_INPUT
 3                 i1 : INT;
 4         END_VAR
 5
 6         VAR_OUTPUT
 7                 o1 : IBA_STRING;
 8         END_VAR
 9
10 (** o1 := 'TestString' + int_to_string(i1); **)
11         (* assign - Stmt *)
12         LD 'TestString'
13         ADD ( i1
14                 int_to_string
15         )
16         ST o1
17
18 END_FUNCTION_BLOCK
```

- Hinter dem Baustein, sehen Sie ein Nummern-Triple, z. B. (3,5,2).
  Das sagt Folgendes aus:

1. Zahl:	Bereich, in dem der Fehler aufgetreten ist.	
	1 = Programmname / FB-Name/Funktionsname	(oben Zeile 1)
	2 = Variablenbereich, beginnt mit „VAR"	(oben Zeile 2)
	3 = Programm, beginnt nach dem letzten END_VAR	(oben Zeile 9)
2. Zahl:	Zeilennummer innerhalb des Abschnitts	(5 entspricht Zeile 13)
3. Zahl:	Spaltennumer (bzw. Tabnummer)	in der betreffenden Zeile (2)

- die fehlerhafte Zeile ist also „ADD (i1 … ".
  Suchen Sie die letzte Kommentarzeile oberhalb dieser Anweisung. In dieser ist der Quelltext der ST-Anweisung zu sehen
  `o1 := 'TestString' + int_to_string(i1);`
- Fehler ist folgender: Der Operand ADD ist nicht für Strings zulässig.
  Evtl. Ursache: in anderen Compilern, z. B. in ibaLogic-V3, kann man mit '+' Strings verknüpfen. Der ibaLogic-V4 Compiler interpretiert '+' immer als Addition. Um Strings aneinanderzuhängen, verwenden Sie bitte die Funktion CONCAT.
- Richtige Anweisung für ibaLogic-V4 lautet:

```
1 v1 := int_to_string(i1);
2 o1 := concat('TestString',v1);
```

Manchmal können Sie aus der Datei „CompilerOut.txt" weitere Informationen entnehmen, die Ihnen helfen, den Fehler zu beseitigen.

Die Datei „CompilerOut.txt" finden Sie unter Windows-Systemen in folgendem Verzeichnis:

- deutsches System
  `C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten \ibalogic\<Workspacename>\<Projektname>\$GEN$\`
- englisches System
  `C:\Documents and Settings\<Username>\Application Data\ibalogic \<Workspacename>\<Projektname>\$GEN$\`

13.4 Leistungsgrenzen

ibaLogic-V4 wurde für die 32 bit-Variante von Windows entwickelt und hat durch dessen Architektur bedingte Einschränkungen:

- Maximale Arbeitsspeichergröße: 4 GB
- Maximale Prozessgröße (d. h. Speicher, den ein Laufzeitsystem belegen kann)
  - bei Zielsystem Win XP: 2 GB
  - bei Zielsystem PADU-S-IT: 32 MB

Der von ibaLogic-V4 verwendete `Microsoft SQL Server 2005 Express` hat folgende systembedingte Leistungsgrenzen:

- Maximale Datenbankgröße: 4 GB
- Maximal 16 Instanzen auf derselben Maschine
- Unterstützung für nur 1 CPU und 1 GB Arbeitsspeicher

Des Weiteren gibt es Beschränkungen durch den verwendeten Compiler, der in ibaLogic integriert ist.

Dieser hat eine maximale Segmentgröße von ca. 64 kB. Das bedeutet, dass Sie innerhalb eines Bausteins (Funktionsbaustein oder Makroblock) nicht beliebig viele Variable definieren können. Sollten Sie dennoch die erlaubte Grenze von 65292 Bytes überschreiten, wird eine Fehlermeldung ausgegeben.

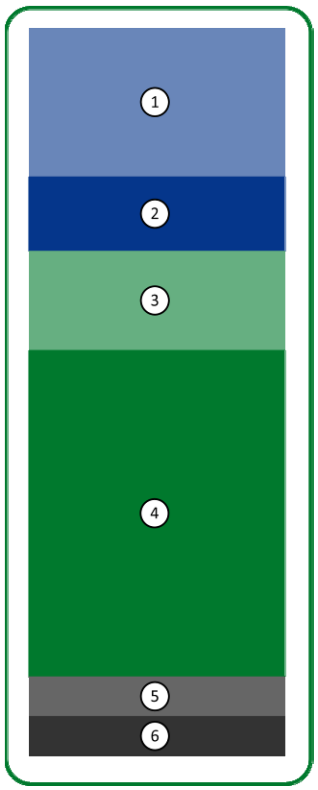
13.4.1 Beispiel

Sie haben ein Array mit 8158 LREAL-Elementen, jedes dieser Elemente belegt 8 Byte, das bedeutet, dass Sie mit diesem Array in einem Funktionsblock 65264 Byte belegen zzgl. des Array-Header von 12 Byte, also 65276 Bytes.

Damit Sie die gesetzte Grenze nicht überschreiten, können Sie den Funktionsbaustein nur mit einem Eingang und einen Ausgang ausstatten, der maximal 4 Byte belegt. Da der Header des FBs auch noch mal 8 Byte belegt.

Element	Bytes
FB-Kopf	8 Byte
Array-Kopf	12 Byte
Array [0 … 8157] LREAL	8158 * 8 Byte = 65264 Byte
Eingang i1 (DINT)	4 Byte
Ausgang o1 (DINT)	4 Byte
Gesamt	65292 Byte

Die übrigen 244 Byte werden vom Compiler für Verwaltungsinformationen benötigt. Die folgende Grafik zeigt den Aufbau des Datensegments in vereinfachter Form.



1 Verwaltung
2 FB-Kopf
3 Array-Kopf
4 Array
5 Eingang
6 Ausgang

Abbildung 131: Datensegment (64 KByte)

Sie können innerhalb eines Projektes bis zu 600 Programme/Tasks anlegen, was aus Gründen der Übersichtlichkeit eher eine theoretische Grenze ist.

Die Anzahl der Projekte innerhalb eines Arbeitsbereichs ist nur durch die maximale Datenbankgröße des Servers beschränkt.

Sollten Sie einen anderen SQL-Server einsetzen, können Sie diese Größenangabe bei Ihrem Systemadministrator in Erfahrung bringen.

14 Programmierregeln

In jedem Programmiersystem besteht die Gefahr, die Programmierung unstrukturiert zu gestalten und damit die Übersichtlichkeit und Lesbarkeit für Sie als Programmierer sowie für den Kunden oder einen anderen Bearbeiter sehr zu erschweren oder sogar unmöglich zu machen.

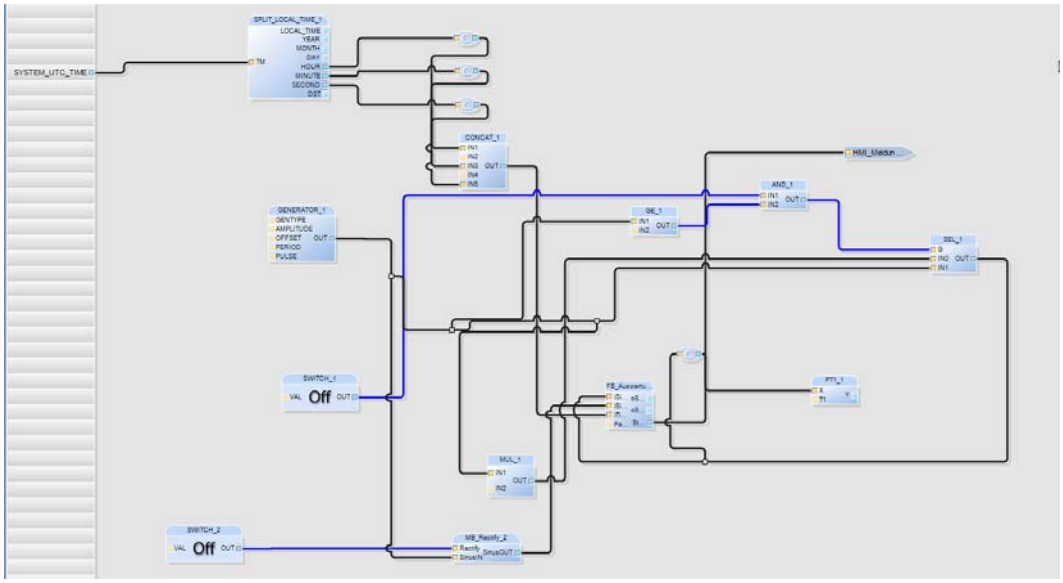


Abbildung 132: Beispiel unstrukturierte Programmierung

Das Beispiel in der oberen Abbildung wird für den Lösungsvorschlag neu strukturiert.

14.1 Lösungsansatz

Zwei Tasks mit folgender Strukturierung:

- ❑ Task 1 Datengenerierung
- ❑ Task 2 Datenverarbeitung

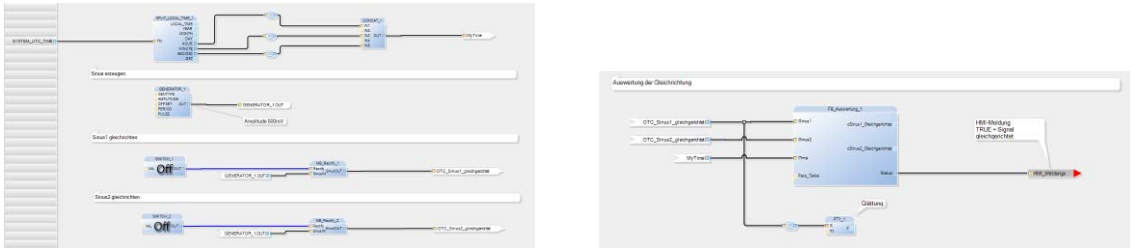


Abbildung 133: Beispiel einer strukturierten Programmierung

Die folgenden Richtlinien müssen Sie nicht zwingend einhalten, jedoch erleichtern sie das Arbeiten mit ibaLogic.

- ❑ Teilen Sie die Funktionen in mehrere Tasks/Programme mit einem bezeichnenden Namen auf.

Abbildung 134: Teilung Task/Programme

- ❑ Legen Sie je einen Task für die Hardware-Eingänge und die Hardware-Ausgänge an. Vergeben Sie die Prioritäten so, dass der Eingangstask als erstes und der Ausgangs-Task als letztes bearbeitet wird.
- ❑ Verwenden Sie innerhalb eines Tasks Intra-Page-Konnektoren, falls zu viele Kreuzungen der Linien das Layout unübersichtlich machen sollten
- ❑ Beschriften Sie die Teilfunktionen mit Kommentarfeldern.

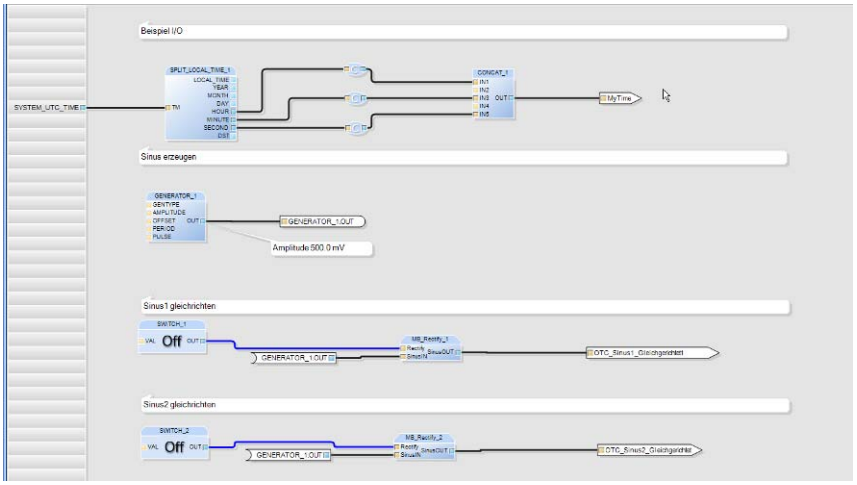


Abbildung 135: Kommentierung

- ❑ Fassen Sie wieder verwendbare Programmteile zu Makro-Blöcken zusammen und versehen Sie diese mit sinnvoller Beschreibung und Kommentaren.
- ❑ Fassen Sie komplexe, zu einer Funktion gehörende Schaltungen zu einem Makro zusammen, um die Übersichtlichkeit zu verbessern.
- ❑ Verwenden Sie Kommentare und Beschreibungen auch innerhalb eines FBs. Wir empfehlen einen Kopf mit Änderungsindex, Überschriften und sinnvolle Einrückungen der Programmzeilen.

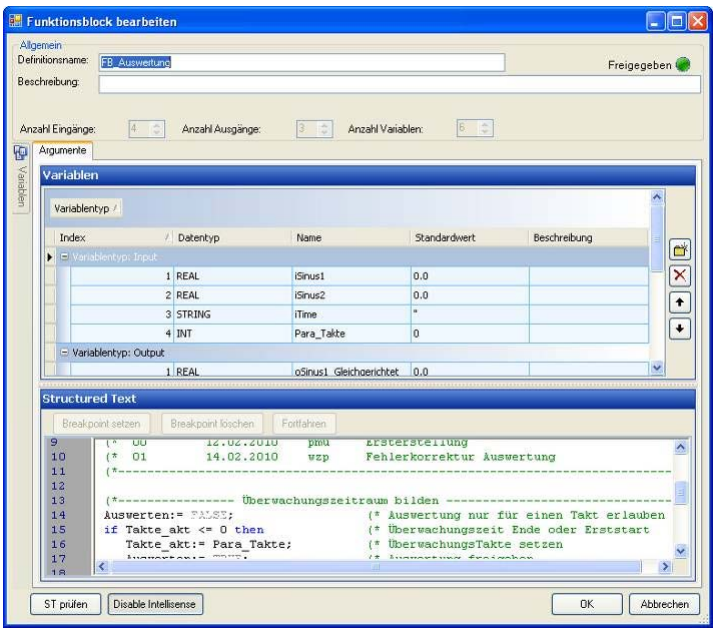


Abbildung 136: Programmcode-Kommentierung

- ❑ Ordnen Sie die Bausteine innerhalb eines Tasks so an, dass sie der Berechnungsreihenfolge entsprechen (von links oben nach rechts unten).
- ❑ Benennen Sie Off-Task-Konnektoren mit einer Präfix, z. B. „OTC_" oder, falls der OTC für ein HMI-System verwendet wird, mit „OPC_".
- ❑ Benennen Sie die Intra-Page-Konnektoren mit Präfix „IPC_". Falls ein „OTC_test" als Eingang vorhanden ist, kann dieser dann als „IPC_test" intern weiter benutzt werden und es tritt keine Namensgleichheit auf.
- ❑ Stellen Sie kurze Präfixe für die Namen der Bausteine, Makros und deren Konnektoren ein, z. B. „FB_", „MB_".
  (Einstellung unter „Extras – Optionen – Editoren – Funktionsbausteine").
- ❑ Geben Sie den Anwenderdatentypen Namen, die einen Hinweis entweder auf die logische Bedeutung (z. B. ST_WALZE) oder Inhalte (z. B. AR_64REAL) enthalten.
- ❑ Gegebenenfalls die Linienführung so verschieben, dass der Verlauf eindeutig zu erkennen ist. Vermeiden Sie Überlappungen.
- ❑ Nutzen Sie die Möglichkeit, Bausteine beliebig zu vergrößern. Damit werden Konnektornamen besser lesbar oder die Linienführung leichter verfolgbar.

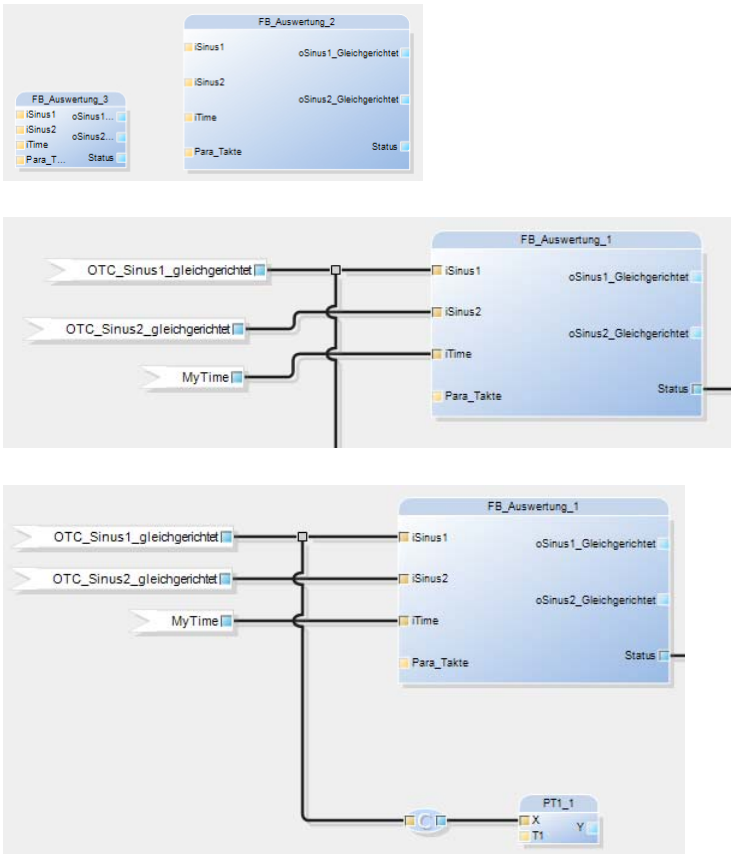


Abbildung 137: Beispiele der vergrößerten Darstellung

- ❑ Fangen Sie generell mögliche Fehlerquellen beim Programmieren ab, wie:
  - Division durch 0
  - Zugriff über Array-Grenzen hinaus
  - Mögliche Endlos-Schleifen

15 Deinstallieren von ibaLogic

Gefahr!

Während der Deinstallation können verschiedene Meldungen auftreten:

- ❑ Abfrage, ob auch die Benutzer-Backups gelöscht werden sollen.
- ❑ Abfrage, ob auch SQL Express deinstalliert werden soll
  (tritt nur auf, wenn ausschließlich die ibaLogic Datenbank existiert hat).

Gefahr!

Gefahr durch Aktivierung oder Deaktivierung von Funktionen!

Mögliche Personen- und Maschinenschäden durch Aktivierung und Deaktivierung von Funktionen und weiteren Diensten (PMAC, OPC …), die sich direkt auf das Verhalten der Anlage auswirken.

Sichern Sie die Anlage beim Arbeiten mit dem System ab!
Beachten Sie geltende Sicherheitsvorschriften!

Wichtiger Hinweis

Ein Deinstallieren der ibaLogic-Software ist nur dem Benutzer möglich, der mit Administrator-Rechten ausgestattet ist. Fragen Sie hierzu den Systemadministrator.

Voraussetzung

- ❑ Es sind alle ibaLogic-Programme geschlossen.

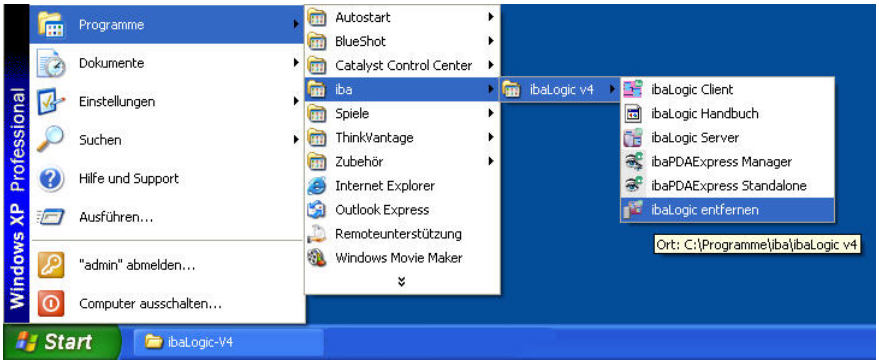
Hinweis

Meldungen während der Deinstallation

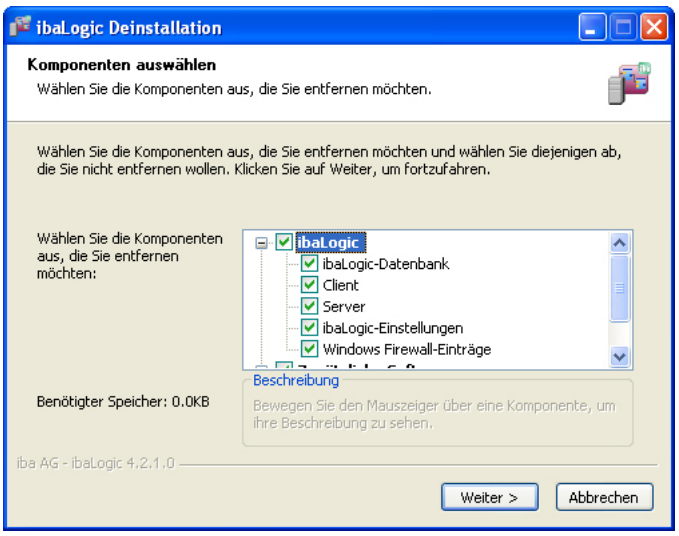
Die SQL Express Instanz wird beim Bestätigen der Frage mit <Ja> entfernt. Es wird die bei der Installation angelegte Datenbank gelöscht (*.ldf, *.mdf).

Vorgehen

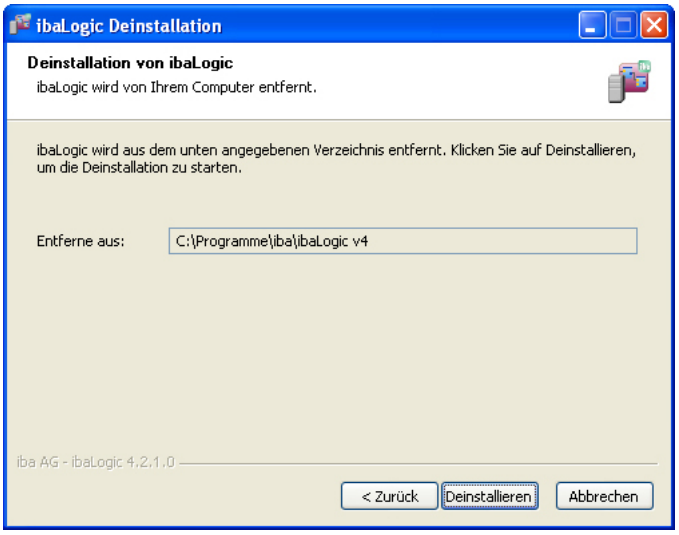
1. Wählen Sie „Start – iba – ibaLogic v4 – ibaLogic entfernen".



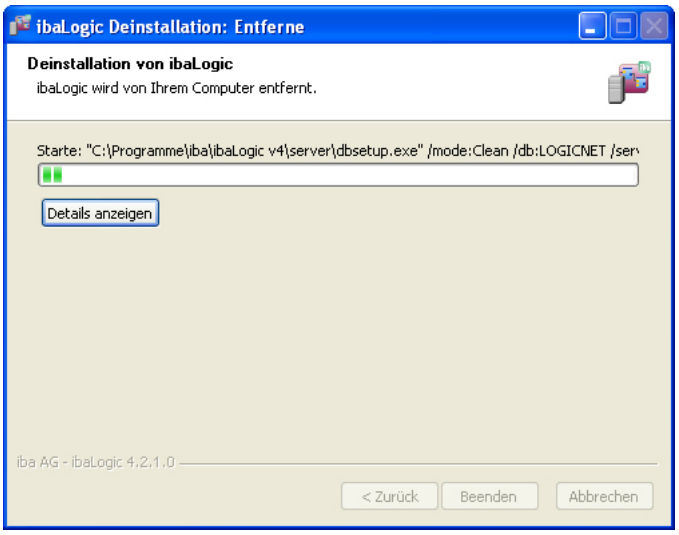
2. Wählen Sie die zu entfernenden Komponenten.



3. Starten Sie den Deinstallationsvorgang durch Betätigen des Buttons <Deinstallieren>.



4. Schließen Sie den Dialog durch Betätigen des Buttons <Beenden>.Bestätigen Sie ggf. Sicherheitsabfragen.



Wichtiger Hinweis

Sind im Installationsverzeichnis Datenbank-Backups abgelegt, wird bei der Deinstallation nachgefragt, ob diese ebenfalls gelöscht werden sollen. Bei Bestätigung mit <Nein> bleiben die Backups erhalten.

16 Übungsbeispiele

Mit diesem Abschnitt möchte iba dem „Einsteiger" bei seinen ersten Schritten mit ibaLogic begleiten.
Ein kleines Beispielprogramm soll den „Aha-Effekt" erzeugen und vor allen Dingen aufzeigen, was iba unter einer ergonomischen CFC-Umsetzung versteht und welche Bedeutung dabei der Online-Aktualisierung der statischen und dynamischen Variablen zugestanden wird.

Da es sich nur um ein einführendes Beispiel handelt, werden nicht alle Funktionen von ibaLogic gezeigt bzw. darauf eingegangen.

16.1 Erste Schritte Beispielprojekt

Es soll ein Programm erstellt werden, mit dem ein Sinussignal erzeugt wird. Diesem Sinussignal soll in Abhängigkeit von einem Schalter ein stufenlos einstellbarer Offset hinzugefügt werden. Das Beispiel lässt sich komplett mit den Standard-Funktionsbausteinen erstellen. Um aber auch die doch weit flexibleren Programmiermöglichkeiten der integrierten Programmiersprache „Strukturierter-Text" (ST) erklären zu können, soll das Beispiel auch mit dieser Alternative programmiert werden.

Die Eingangssignale und das Ergebnis sollen als Messverlauf dargestellt werden.

Damit die verwendeten Konnektoren laufend aktualisiert werden, wird das Beispiel „scharf" geschaltet. Der Farbumschlag zu Pink zeigt an, dass nun alles „ernst" wird: Die Anlage steht quasi unter Spannung!

Sollten bereits Ausgänge angeschlossen sein (Aktoren, Motoren etc.), so würden diese sofort auf Programmänderungen reagieren. Die übliche Vorgehensweise - Programmieren, Kompilieren, Linken und Laden sowie Starten - läuft automatisch im Hintergrund ab. Weiterhin werden zur Vereinfachung die voreingestellten Werte für die Projekt- und Programmbezeichnungen übernommen.

16.1.1 Übungsaufgabe Teil 1

16.1.1.1 Aufgabenstellung

Der sich periodisch ändernde Wert eines Generators (Sinussignal) soll in Abhängigkeit eines Schalters auf den Wert eines Schiebereglers addiert werden:

Schalterstellung 1: Generator + Generator

Schalterstellung 2: Generator + Schieberegler

Alle Variablen und selbstverständlich auch das Ergebnis sollen grafisch als Kurvenverlauf angezeigt werden.

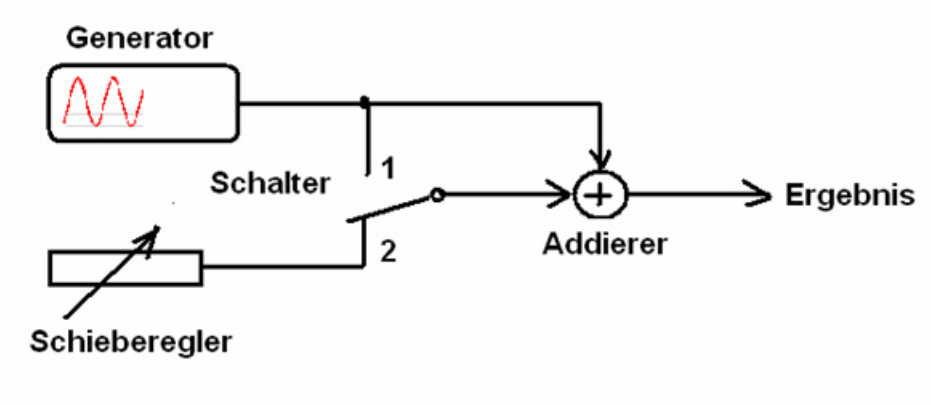


Abbildung 138: Schaltplan Übungsaufgabe

Das Beispiel soll im ersten Übungsteil mit Hilfe der standardmäßig in ibaLogic enthaltenen Funktionsbausteine (FB) realisiert werden.

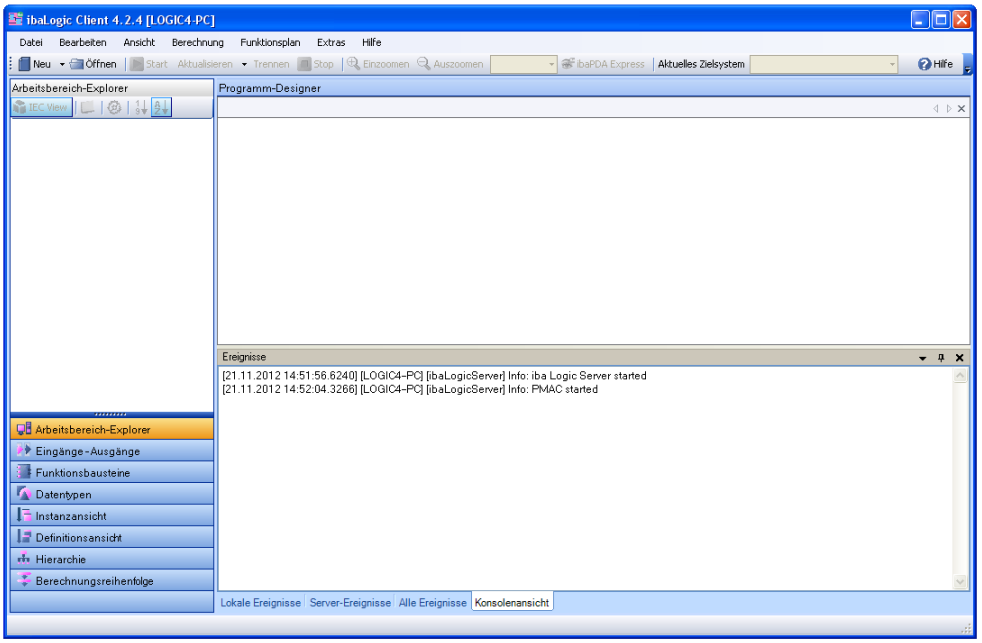
Da zum Testen der Übungsaufgabe nicht erst Laborgeräte wie Funktionsgenerator, Schieberegler und Taster an die Eingänge von ibaLogic anschgeschlossen müssen, sind solch effektive Funktionen als **Specials** zusammengefasst und können wie andere Bausteine platziert werden. Bei dem **Switch** und dem **Slider** ist eine aktive Bedienung zum Test der Schaltung möglich.

16.1.1.2 ibaLogic Server und ibaLogic Client starten

Vorgehen

1. Doppelklicken Sie auf das Icon „ibaLogic Server" auf dem Desktop.
   Nach der Initialisierungsphase öffnet sich der ibaLogic Server-Dialog. Der Server wird standardmäßig mit dem Öffnen des Dialogs automatisch gestartet.

2. Doppelklicken Sie auf das Icon „ibaLogic Client" auf dem Desktop.
   Nach der Initialisierungsphase öffnet sich der ibaLogic Client-Dialog.



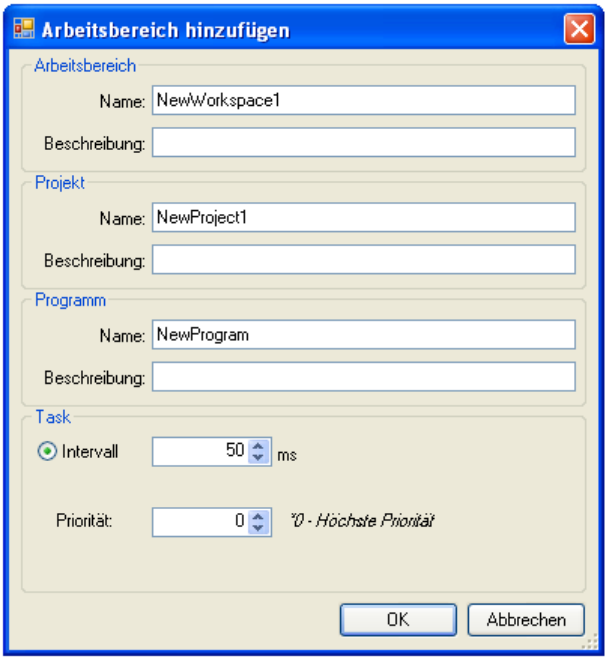
Anmerkung

Das Ereignisfenster unter dem Programmfenster dokumentiert die Programmaktionen und mögliche Kollisionen. Falls beim Start von Server oder Client Fehlermeldungen eingeblendet werden, finden Sie in der vorstehenden Dokumentation eine Hilfestellung.

16.1.1.3 Neues Projekt anlegen

Vorgehen

1. Betätigen Sie den Button <Neu> in der Symbolleiste.
   Dialog „Arbeitsbereich hinzufügen" wird angezeigt.



2. Bestätigen Sie die Angaben mit <OK>.

Ohne weitere Veränderung der Voreinstellung heißt Ihr Projekt „NewProjekt1" mit dem einzigen Programm „NewProgram". Die voreingestellte Intervallzeit von 50 ms ist für das Beispiel ausreichend.

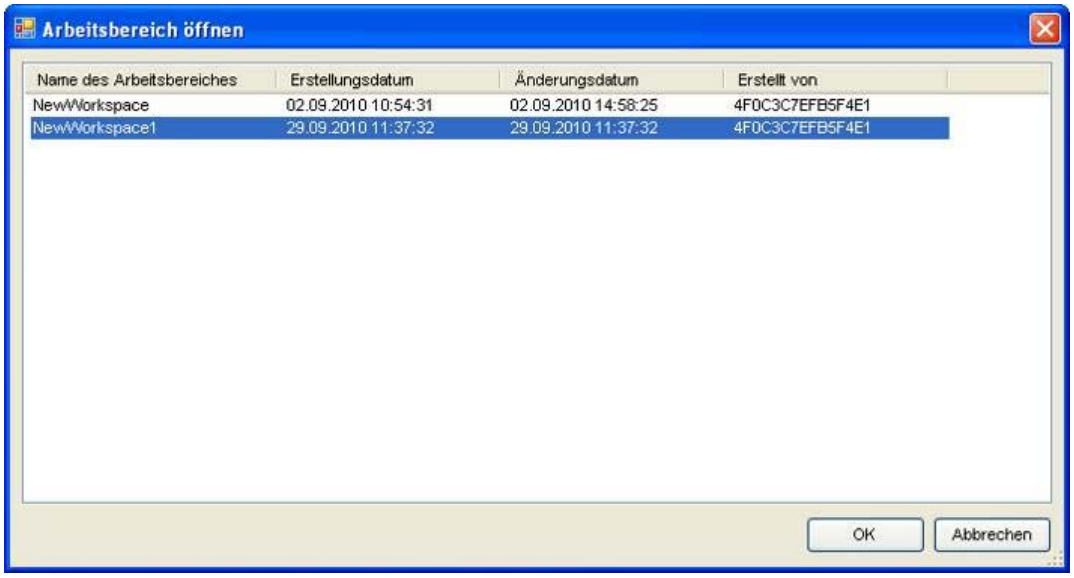
Anmerkung

Sollten Sie nach Beginn Ihrer Übung mit ibaLogic die „Sitzung" unterbrechen müssen oder einfach alle Programme (zuerst Client, dann Server) beendet haben, so müssen Sie nicht mit <NEU> beginnen.

Ihre Änderungen werden automatisch gespeichert.

Betätigen Sie in einem solchen Fortsetzungsfall einfach den Button „Öffnen“, öffnen den von Ihnen erstellten Arbeitsbereich und setzen Ihre Arbeit an der stelle fort an der Sie aufgehört haben.

Sollten Sie mehrere Arbeitsbereiche haben können Sie über das Feld "Änderungsdatum" die Suche eingrenzen.



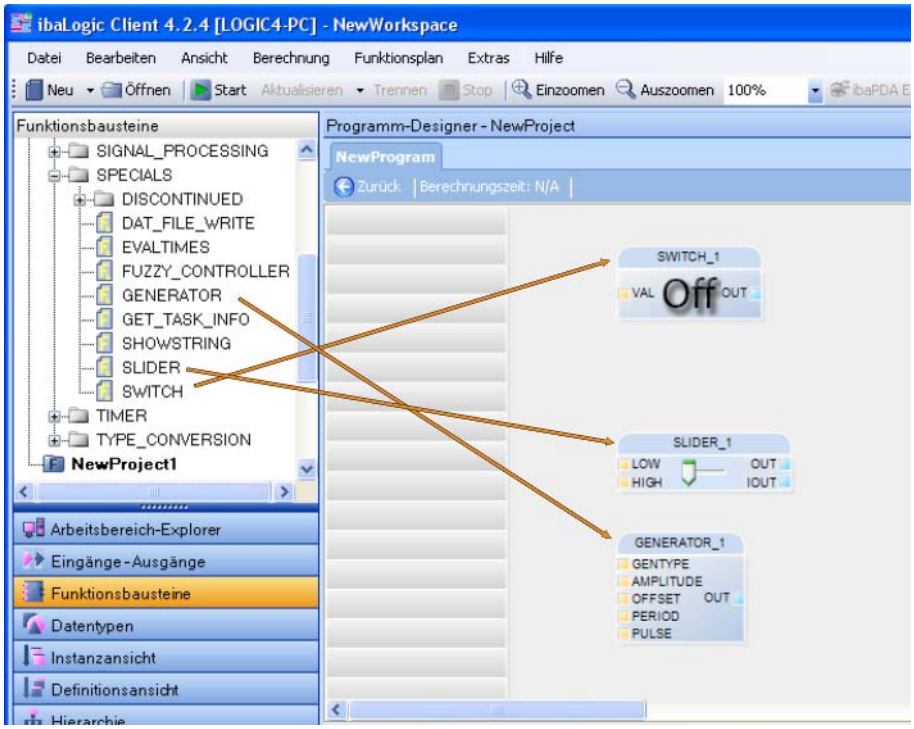
16.1.1.4 Platzieren der Testwerkzeuge

Für die Aufgabenstellung wird jeweils ein Funktionsbaustein benötigt:

- ❑ „Generator"
- ❑ „Switch"
- ❑ „Slider"

Vorgehen

1. Klicken Sie die Schaltfläche <Funktionsbausteine>. Es öffnet sich im Navigationsbereich ein Verzeichnisbaum.
2. Öffnen Sie den Ordner „Specials".
3. Ziehen Sie einen „Generator", einen „Slider" und einen „Switch" auf das Programmierfeld mit gedrückter linker Maustaste des Programm-Designers.



4. Ordnen Sie die Bausteine in der richtigen Reihenfolge an.

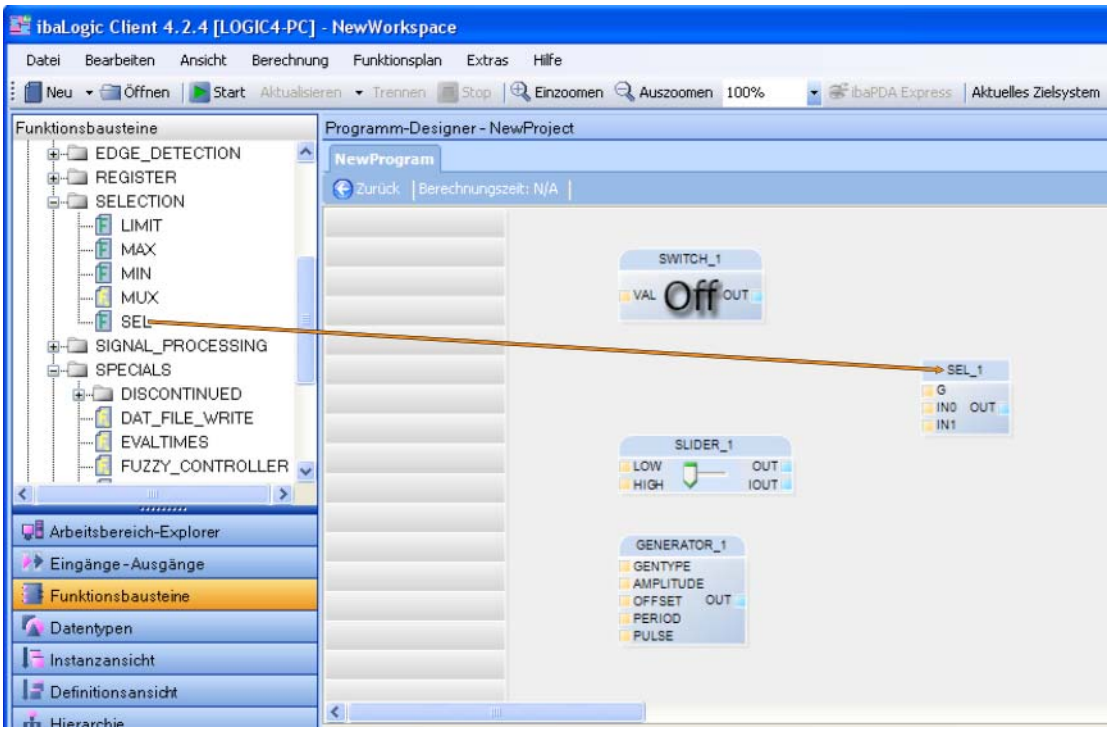
16.1.1.5 Platzieren der Rechenbausteine

Für die Aufgabenstellung wird jeweils ein Funktionsbaustein benötigt:

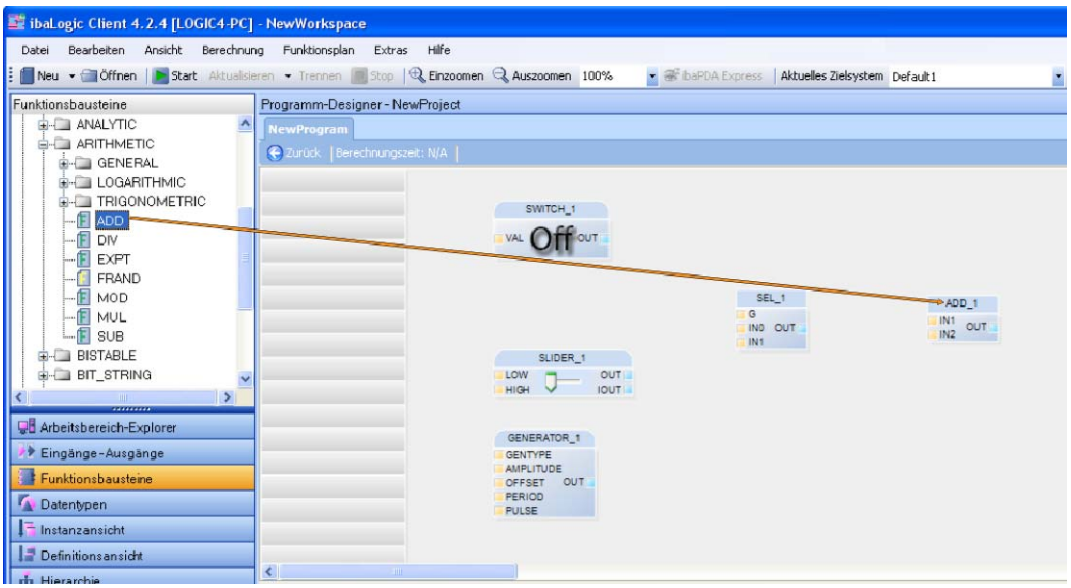
- ❑ „Selektor"
- ❑ „Addierer"

Vorgehen

1. Öffnen Sie den Ordner „Selection" im Verzeichnisbaum des Navigators.
2. Ziehen Sie den Selektor (SEL) auf das Programmierfeld mit gedrückter linker Maustaste des Programm-Designers.



3. Öffnen Sie den Ordner „Arithmetic" im Verzeichnisbaum des Navigators.
4. Ziehen Sie den Addierer (ADD) auf das Programmierfeld des Programm-Designers mit gedrückter linker Maustaste.



Anmerkung

Der SELEKTOR (SEL) verlangt ein binäres Signal als „Entscheider" (Selektor). Die für die Selektion infrage kommenden Werte IN0 und IN1 sind noch formatfrei - sie passen sich automatisch an den Datentyp an, mit dem sie verbunden werden.

Wie auch schon beim Selektor beschrieben, sind die zu addierenden Werte vorerst formatfrei. Erst die Verbindung, die als erstes an einen der Eingänge „angeklemmt" wird, bestimmt das Format.

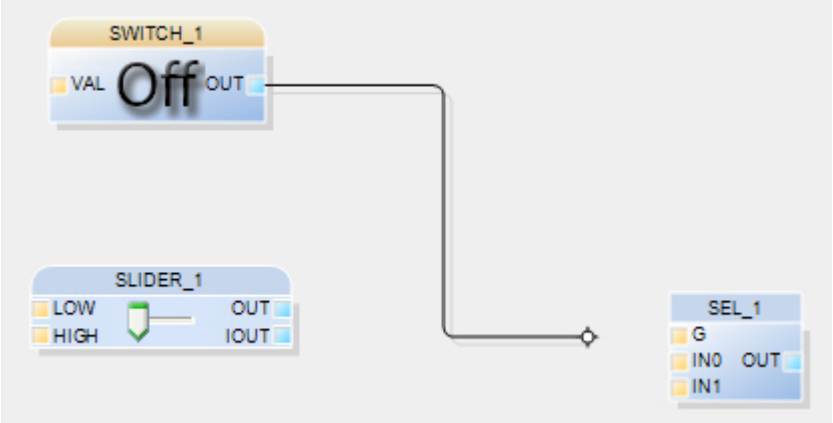
16.1.1.6 Verdrahten des Selektor-Bausteins mit den Testwerkzeugen

Verbunden werden muss:

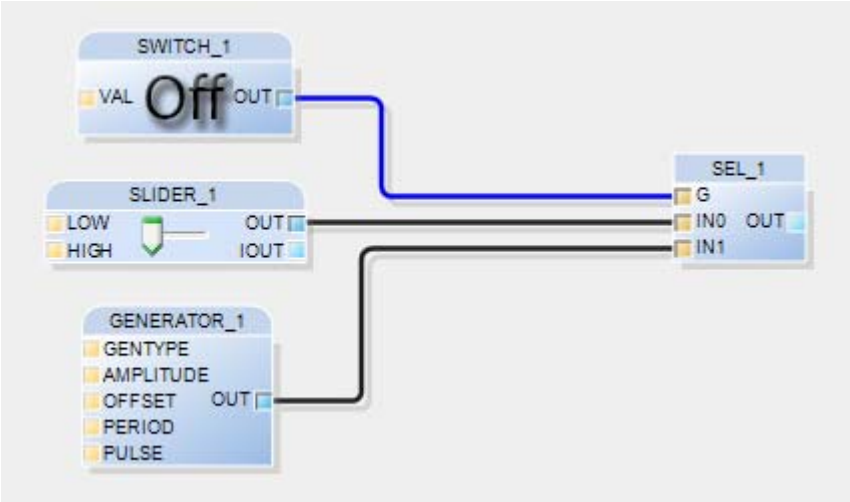
- ❑ Der Schalterausgang (SWITCH) OUT mit dem Entscheidungseingang G des Selektors (SEL).
- ❑ Den Schiebereglerausgang (SLIDER) OUT mit dem Eingang IN0 des Selektors (SEL).
- ❑ Der Generatorausgang OUT mit dem zweiten Eingang IN1 des Selektors (SEL).

Vorgehen

1. Positionieren Sie den Mauszeiger auf den Konnektor (OUT) des Schalters und ziehen Sie den Mauszeiger bei gedrückter linker Maustaste auf den Konnektor (G) des Selektors.



2. Erstellen Sie entsprechend die restlichen Verbindungen.



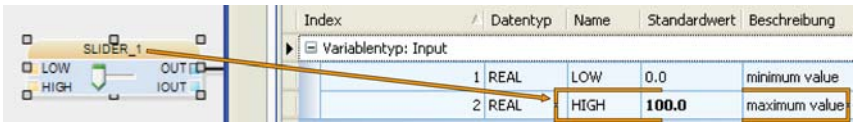
16.1.1.7 Parametrieren des Sliders und Generators

arametriert werden müssen:

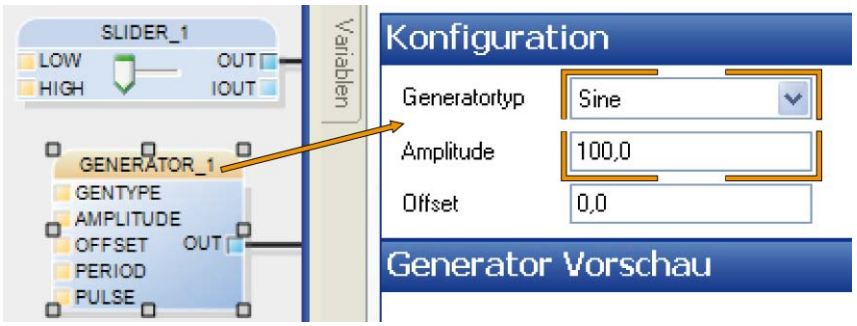
- ❑ „Slider"
- ❑ „Generator"

Vorgehen

1. Doppelklicken Sie auf den Slider. Es wird das Parametriermenü aufgeblendet.
2. Stellen Sie den Arbeitsbereich „maximum value" auf 100.0 ein. Der Slider arbeitet entsprechend der Schieberstellung zwischen 0 und 100.



3. Öffnen Sie das Parametriermenü des Generators.
4. Stellen Sie den Generatortyp Sinus ein.
5. Geben Sie einen Amplitudenwert von 100 ein.



Der Generator erzeugt mit dieser Einstellung Sinusschwingungen mit einem Wert zwischen +100,0 und -100,0.
Die Periodendauer der Schwingungen lassen Sie, wie voreingestellt auf 10 s stehen.

16.1.1.8 Teilverdrahtung online schalten

Indem die Berechnung gestartet wird, wird auch der Online-Compiler aktiviert.

- ❑ Die Hintergrundfarbe des Programm-Designers ändert sich in Pink.
- ❑ Alle binären Verbindungslinien werden zustandsabhängig eingefärbt.
- ❑ Die Bausteinkonnektoren werden laufend aktualisiert und angezeigt.
- ❑ Alle Verbindungen sind jetzt „scharf" geschaltet.

Im übertragenen Sinne ist dies mit dem Anlegen einer Spannung an eine Versuchsschaltung vergleichbar.

Im Ernstfall werden mit ibaLogic über die angeschlossenen Ausgänge die Maschinen angesteuert. Dann wäre die Auswirkung von „unter Spannung verdrahten" im Fall eines Programmierfehlers drastisch spürbar.

Vorgehen

➲ Klicken Sie auf den Button <Start> in der Symbolleiste, um die Berechnung des Übungsbeispiels zu starten. Bestätigen Sie die Abfrage mit <Ja>. Die oben erwähnten Online-Eigenschaften werden aktiviert.

16.1.1.9 Testen des Switches und Selektors

Da der Selektor bereits verdrahtet und zudem die Schaltung schon online geschaltet ist, kann seine Funktion abhängig vom Schaltzustand des Eingangs (G) gleich ausprobiert werden.
Die Funktionsweise der Selektoren ist im vorliegenden Handbuch unter den Standardbausteinen beschrieben. Nun können Sie diese Erklärungen in die Tat umsetzen.

Vorgehen

➲ Ändern Sie den Zustand des Switches durch Klicken der linken Maustaste auf den SWITCH.

Wie Sie erkennen, wird im ausgeschalteten Zustand (Off) der Slider-Ausgang (Wert = 43.5) auf den SEL-Ausgang (OUT) geschaltet.

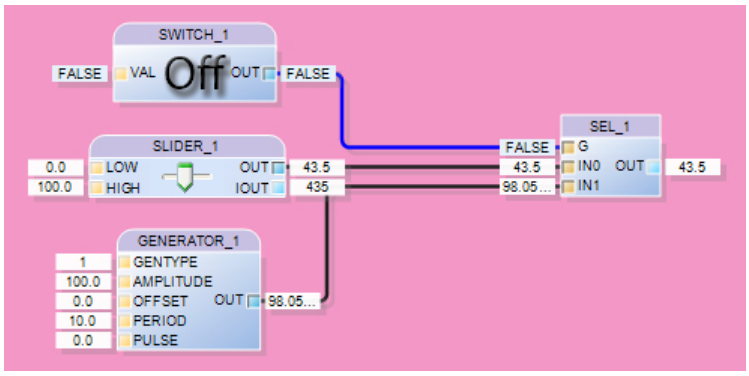


Abbildung 139: Slider-Ausgang (Wert = 43.5) auf den SEL-Ausgang (OUT) geschaltet

Bei eingeschaltetem Zustand (ON) wird der sich periodisch ändernde Generatorausgang auf den SEL-Ausgang (OUT) gelegt.

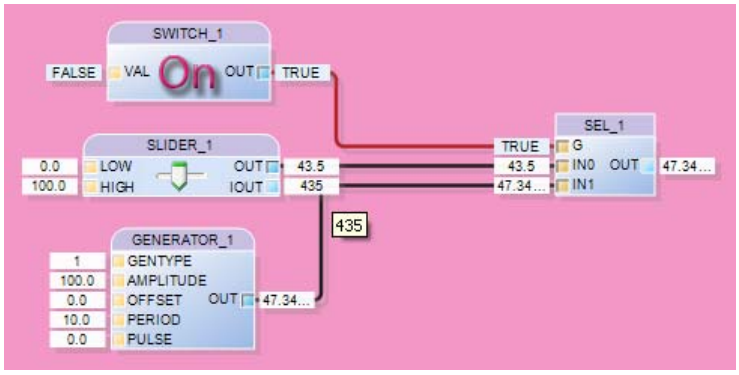


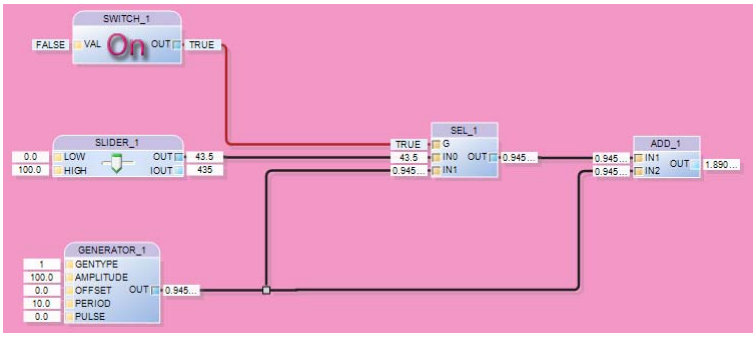
Abbildung 140: Generatorausgang auf den SEL-Ausgang (OUT) gelegt

16.1.1.10 Verdrahten des Addierers

Schließen Sie den Addierer an, um den Schaltplan gemäß Übungsaufgabe zu vervollständigen.

Vorgehen

1. Positionieren Sie den Mauszeiger auf den Selektor-Ausgang (OUT) und ziehen Sie den Mauszeiger bei gedrückter linker Maustaste auf den Eingang (IN1) des Addierers.



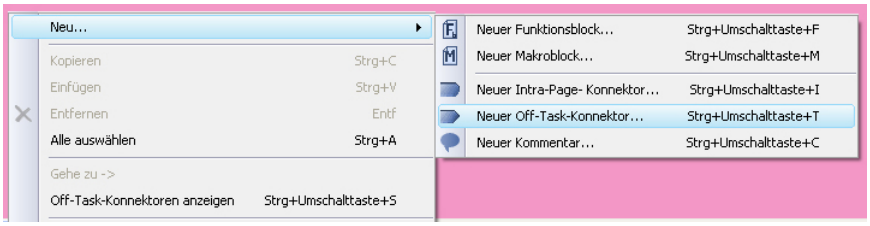
2. Positionieren Sie den Mauszeiger auf den Eingang (IN2) des Addierers und ziehen Sie den Mauszeiger bei gedrückter linker Maustaste auf den Generator-Ausgang (OUT).
   Der über den Mauszeiger bewegte Verbindungspunkt rastet automatisch ein und bildet einen Abzweig.

16.1.1.11 Erstellen eines OTC zur Veranschaulichung des Ergebnisses

In unserem Übungsbeispiel wird zur Veranschaulichung des Ergebnisses dieses einfachen Beispiels ein Off-Task-Konnektor platziert und auch verdrahtet. Der OTC wird „Ergebnis" genannt.

Vorgehen

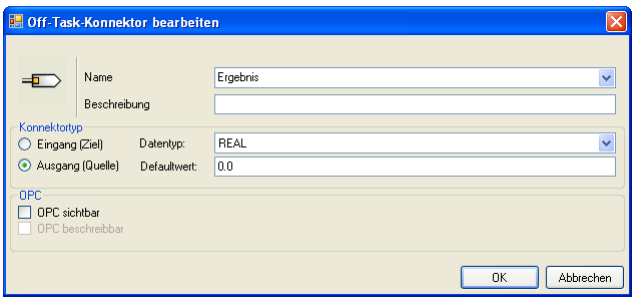
1. Klicken Sie mit der rechten Maustaste auf das Programmierfeld des ProgrammDesigners.
2. Wählen Sie im Kontextmenü „Neu… – Neuer Off-Task-Konnektor…".



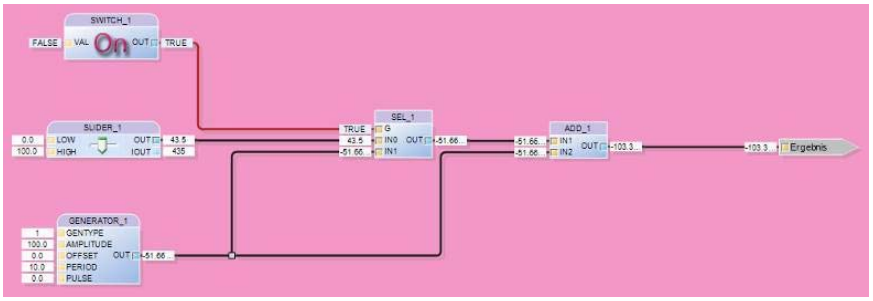
   Das Fenster „Off-Task-Konnektor bearbeiten" wird angezeigt.

3. Vergeben Sie im Eingabefeld den Namen „Ergebnis". Alle anderen Voreinstellungen bleiben wie eingestellt. Stellen Sie den Datentyp „REAL" und einen Default-Wert von 0.0 ein.

Die Voreinstellung als Ausgang stimmt, da dem OTC ein Wert zugeführt wird.
Ziel wäre der OTC, wenn er von einem anderen Programm über den OTC einen Wert übermittelt bekäme.



4. Verdrahten Sie den OTC „Ergebnis" mit dem Addierer.



Anmerkung

Ein OTC ist ein Schlüsselelement bei der grafischen Programmierung.

Der OTC erleichtert die Übersichtlichkeit eines Projektes, indem es auf mehrere Programme aufgeteilt wird, die über OTCs untereinander kommunizieren.
Innerhalb eines Programms wird dagegen der Inter-Page-Connector verwendet.

In diesem Handbuch finden Sie auch ein „Programmierregeln". Dort werden Anregungen geliefert, wie durch Verwendung von OTCs und IPCs eine Bandwurmprogrammierung (Drahtverhau) vermieden werden kann.

Ein ganz bedeutendes Kommunikationselement ist der OTC für die Verbindung mit einem übergeordneten Bedien-Beobachten-System (HMI).

16.1.1.12 Analyse der Schaltung

Mit der dynamischen Anzeige der Konnektoren kann das Ergebnis kontrolliert werden. Aber selbst in diesem einfachen nur mit 50-ms-Zykluszeit laufendem Programm fällt eine numerische Kontrolle des Ergebnisses äußerst schwer.

Beachten Sie die nachfolgende Übungsaufgabe Teil 2 , Seite 272.

16.1.2 Übungsaufgabe Teil 2

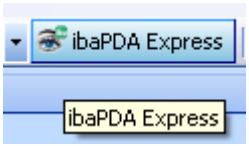
Mit diesem Übungsbeispiel wird der Komfort einer dynamischen Online-Kurvenanzeige zur Auswertung des Ergebnisses aufgezeigt.

16.1.2.1 Programmanalyse mittels ibaPDA Express

Im ibaLogic ist eine ibaPDA-Anzeige zur Online-Kurvenanzeige integriert.

Vorgehen

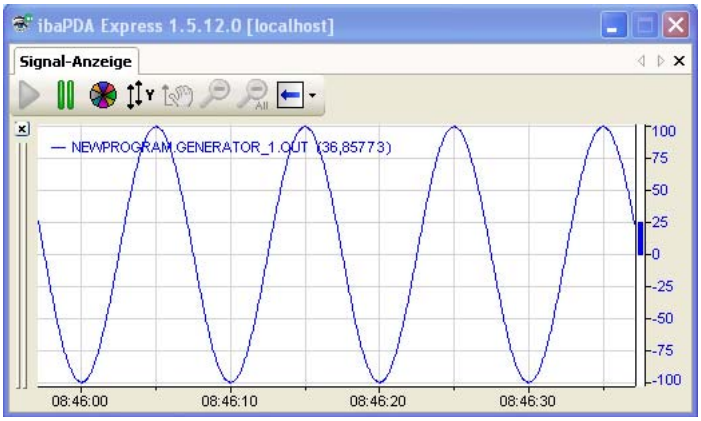
➲ Klicken Sie auf den integrierten ibaPDA Express in der Symbolleiste.
Dieses Programm ist in jeder ibaLogic-Version ohne Zusatzkosten integriert.



➲ Führen Sie den Mauszeiger auf den OUT-Konnektor des Generators und ziehen Sie diesen mit gedrückter <ALT>-Taste oder per Drag & Drop in das ibaPDA Express-Fenster.

➲ Verfolgen Sie den angewählten Zwischenwert (Generator-OUT) als Kurvenanzeige.

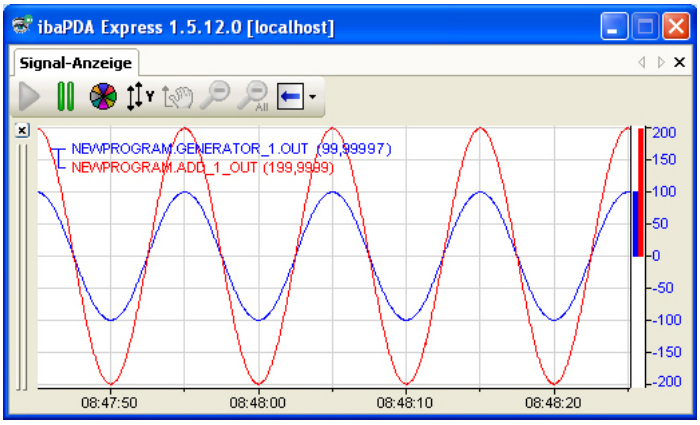
➲ Passen Sie die Y-Achse mit dem Button <Alles autoskalieren> an.



➲ Ziehen Sie für sich interessante Zwischenwerte und Ergebnisse in das ibaPDA Express-Fenster.

- Handelt es sich um gleich skalierte Größen, so ziehen Sie die Werte auf den Konnektortext des bereits angezeigten Signales.
- Liegen unterschiedliche Skalierungen vor, so ziehen Sie die Konnektoren per Drag & Drop in das Kurvenfenster - es entsteht eine neue Skala.
- Um ein neues Kurvenfenster zu öffnen, ziehen Sie den Konnektor auf einen Bereich außerhalb des aktuellen Kurvenfensters.

➲ Führen Sie eine Endkontrolle der gestellten Übungsaufgabe durch Betätigen der Switch-Stellung On/Off durch. Beobachten Sie den sich verändernden Kurvenverlauf.



16.1.3 Übungsaufgabe Teil 3

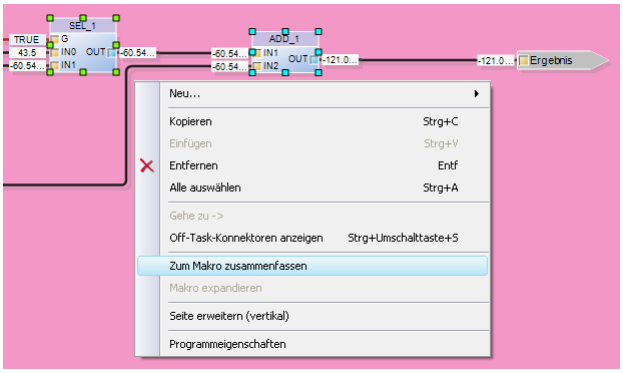
Verbesserung der Programm-Übersichtlichkeit

Um die Übersichtlichkeit eines Schaltungsentwurfes zu verbessern, können Funktionsbausteine zu Makros zusammengefasst werden.

16.1.3.1 Vorgehen

An diesem kleinen Übungsbeispiel wird diese Technik demonstriert.

➲ Markieren Sie die infrage kommenden FBs und deren Verbindungslinien bei gleichzeitig gedrückter <Strg>-Taste. In diesem Fall markieren Sie den Selektor, den Addierer und deren Verbindungslinie.



➲ Klicken Sie mit der rechten Maustaste bei gedrückter <Strg>-Taste, so erscheint das Makro-Menü.

➲ Wählen Sie im Makro-Menü „Zum Makro zusammenfassen".

➲ Bestätigen Sie die Zwischenmaske (Makro, Eingänge, Ausgänge).

16.1.3.2 Anmerkung

Es wird ein Makro erstellt. Das entstandene Makro mit einem voreingestellten Namen (IMPL_MB_1) erfüllt die gleichen Funktionen wie die zuvor angewählten Einzelbausteine. Es kann für eine aussagekräftige Dokumentation natürlich noch sinnvoll bezeichnet werden.

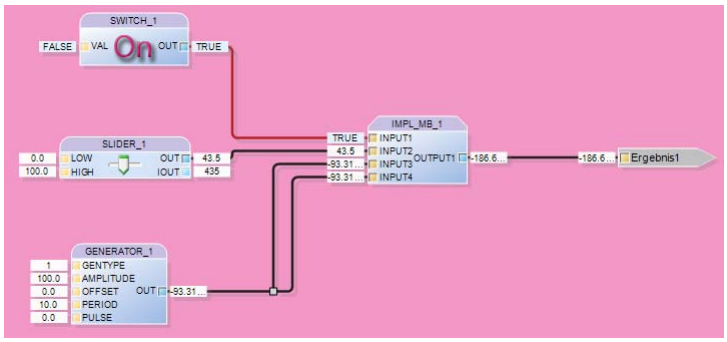
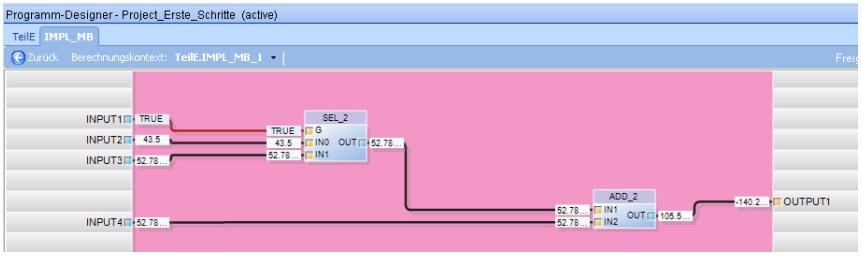


Abbildung 141: Makroerstellung

Tipp

Sollen eine größere Anzahl von FBs zu einem Makro zusammengefasst werden, so gibt es eine Lasso-Methode. Bei gedrückter linker Maustaste wird ein Markierungsrechteck über die gewünschten Objekte gezogen.

➲ Doppelklicken Sie auf das erzeugte Makro, um den Inhalt anzuzeigen. Zum Teil sind die Linien oder die FBs etwas unstrukturiert platziert. Sie können auch im Makro editieren, z. B. vergessene Verbindungslinien ergänzen oder auch nur grafisch „Aufräumen".



➲ Wählen Sie <Zurück>, um die Makro-Zoom-Anzeige zu verlassen.

16.1.4 Übungsaufgabe Teil 4

Erstellen von Funktionsbausteinen unter Verwendung von ST

Die standardmäßig zur Verfügung gestellten Funktionsbausteine sind normentsprechend sehr umfassend. Aus der Erfahrung eigener Applikationen und vor allem auf Anregung von ibaLogic-Nutzern, wurden im Laufe der Jahre einige weitere nützliche Sonder-FBs aufgenommen.

Um dem Benutzer die Möglichkeiten aufzuzeigen, Sonderbausteine selbst zu erstellen und wiederzuverwenden, wird zum Abschluss dieser Übungsaufgabe ein FB selbst programmiert. Zum Funktionstest läuft dieser neue FB parallel zu dem zuvor erstellten Makro. Die Ergebnisse müssen deckungsgleich sein.

In der IEC Norm 61131-3 wird der Programmiersprache „Structured Text" ein ganz besonderer Stellenwert zugeschrieben. Mit dieser Hochsprache können in Annäherung

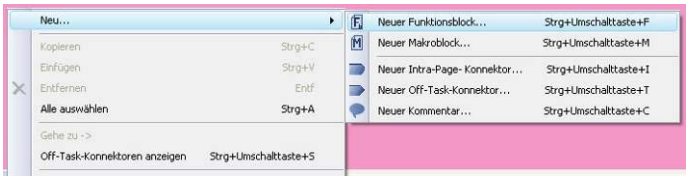
an die Pascal-Programmiersprache sehr übersichtliche und gut lesbare FBs programmiert werden. Die ST-Bausteine sind auch in Fremdsysteme portierbar.

Kommen Sie aus der klassischen SPS (PLC)-Welt, so sind Sie an die Verwendung von Standard-FBs gewöhnt.

Steigen Sie hingegen als Assembler- oder Hochsprachen-Programmierer in diese Art der Steuerungswelt ein, so kommen Sie wahrscheinlich mit der Structured Text-Programmierung besser zurecht.

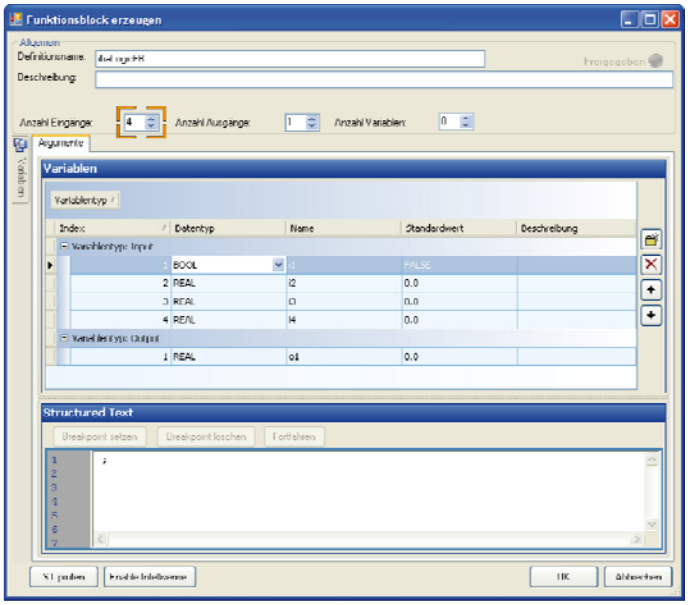
16.1.4.1 Vorgehen

➲ Wählen Sie im Kontextmenü „Neu… – Neuer Funktionsblock…".
Der Dialog „Funktionsblock erzeugen" wird angezeigt.



➲ Geben Sie als Definitionsname "FB_1x" ein.

➲ Stellen Sie bei „Anzahl Eingänge" 4 ein. Da das Makro aus der Übungsaufgabe Teil 3 nachgebaut werden soll, benötigt der FB genauso viele Ein- und Ausgänge (4 Eingänge und einen Ausgang).



➲ Stellen Sie die Datentypen für die Eingänge und den Ausgang ein.

Für die Eingänge

❑ 1x BOOL

❑ 3x REAL

Für den Ausgang

❑ 1x REAL

Mit dem Button „  " legen Sie eine neue Variable an.

➲ Geben Sie im Feld „Structured Text" ein Semikolon ein.

16.1.4.2 Anmerkung

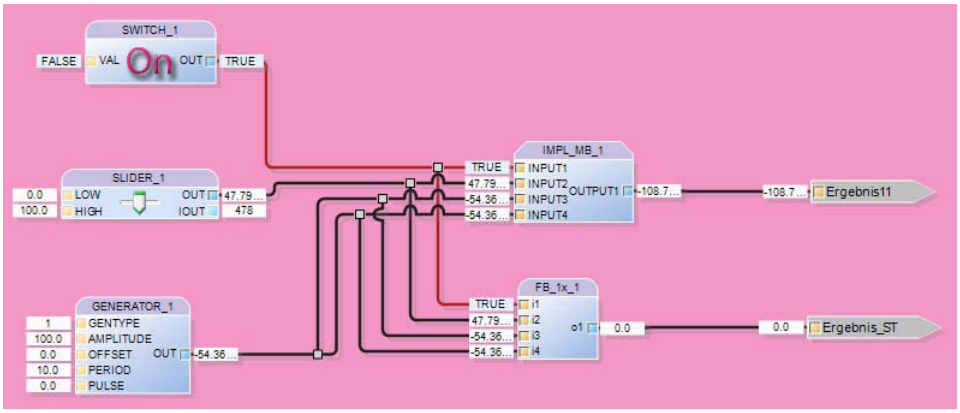
Im ersten Ansatz wird lediglich ein „leerer" Funktionsblock erstellt.

Aus formalen Gründen muss das ST-Feld mindestens ein Semikolon beinhalten.

Da die Ein- und Ausgänge bereits definiert sind, kann der neue FB in das Layout einfügt und auch parallel zum bestehenden Makro angeschlossen werden.

➲ Positionieren Sie den neuen FB passend in das Layout.

➲ Verdrahten Sie den neuen FB parallel zum bestehenden Makro. Um den zweiten Ergebnis-OTC anzulegen, kopieren Sie den vorhandenen mit <Strg+C>. Das Einfügen erfolgt mit Drücken von <Strg+V>. Vergeben Sie dem OTC einen neuen Namen z. B. „Ergebnis_ST".

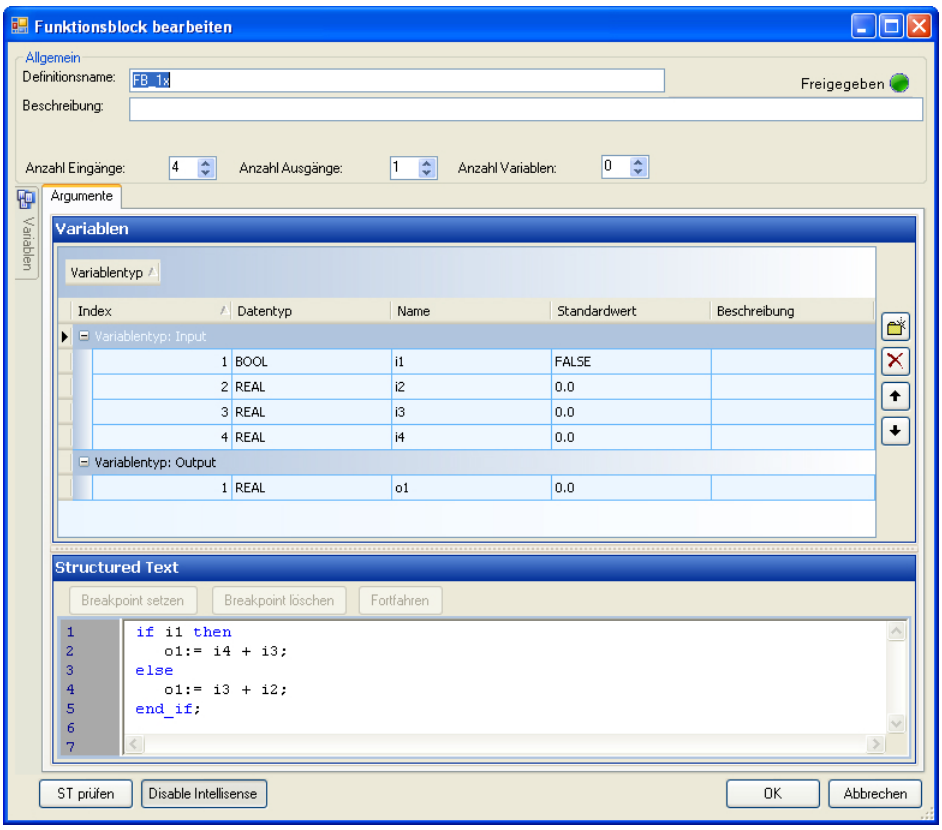


➲ Sie können erkennen, dass der FB noch nicht auf seine Eingänge reagiert. Dieser muss noch für seine Aufgabe programmiert werden.

➲ Doppelklicken Sie auf den neuen FB. Das Fenster „Funktionsblock bearbeiten" wird angezeigt.
Nun erfolgt die Programmierung mit den bei fast allen Hochsprachen üblichen „if", „then", „else" und „end_if"- Anweisungen.

➲ Geben Sie den folgenden Quellcode (wie im Screenshot s.u.) in das Feld „Structured Text" ein:

```
1 if i1 then
2     o1 := i4 + i3;
3 else
4     o1 := i3 + i2;
5 end_if;
```



16.1.4.3 Ergebnis

Das Beispiel beginnt mit der Abfrage des Schalters (if i1 = „TRUE" oder kürzer if i1) und addiert entsprechend die Eingänge i4 + i3 oder i3 + i2.

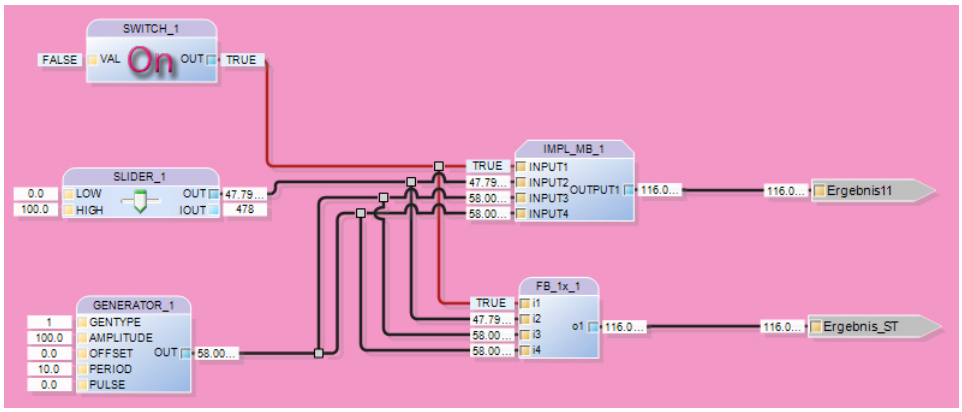


Abbildung 142: Beispiel Schaltung mit Addierungsabfrage

16.1.4.4 Anmerkung

Beachten Sie auch, dass alle Variablen im Baustein online aktualisiert werden!

➲ Ziehen Sie das Ergebnis des ST-Bausteins mit gleicher Skalierung in die ibaPDA Express-Anzeige.

Die zweite rote Kurve ist deckungsgleich zur blauen Kurve (Ergebnis aus dem Makro). Es ist nur die letzte Farbe (blau) der Kurve 2 sichtbar. Dass es sich um zwei einzelne Werte handelt, kann nur über die rechts eingeblendete Balkenanzeige erkannt werden.

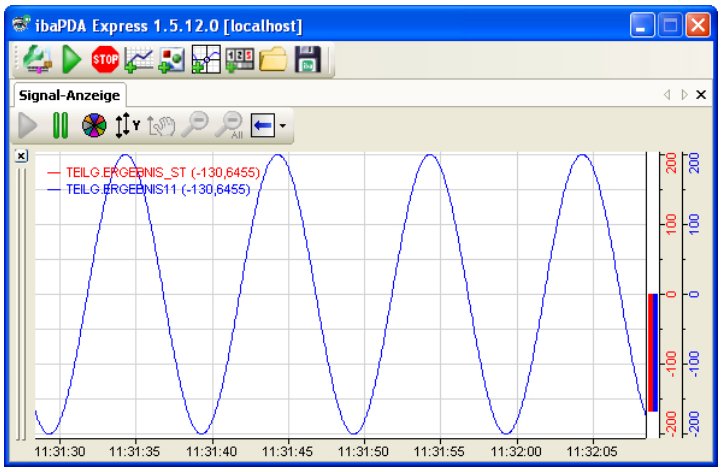


Abbildung 143: Deckungsgleichheit der Ergebnisse

16.2 Beispielprojekt DAT_FILE_WRITE

16.2.1 DAT_FILE_WRITE im Modus „Unbuffered"

Im Modus „Unbuffered" (Erläuterungen siehe "Buffered Mode , Seite 195") wird bei jedem Speicherzyklus ein Daten-Sample gespeichert.

Verwenden Sie diesen Modus, wenn der Abtastzyklus der zu speichernden Daten dem ibaLogic-Taskzyklus entspricht. Oder wenn Sie Daten speichern wollen, die in ibaLogic selbst erzeugt werden.

Aufgabe: Speichern 8 Analogwerte und 8 Digitalwerte aus ibaLogic

Voreingestellte Parameter

- Zielsystem: WinXP
- Taskintervall: 20 ms

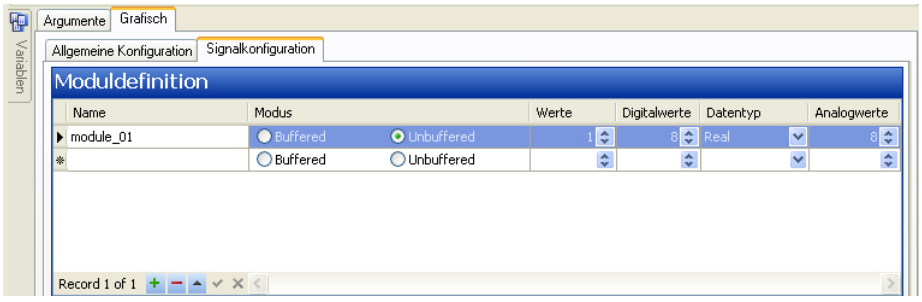
16.2.1.1 Schritt 1: Parametrieren Sie den DFW- Baustein

➲ Allgemeine Konfiguration

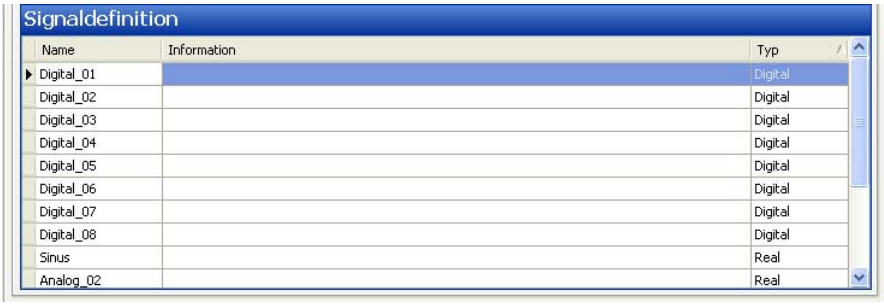
Asynchr. Zugriff:	Deaktiviert
Speicherzyklus:	10 (nicht relevant)
Offset Startzeit:	0
Werte speichern:	**Aktiviert**
Datei schreiben:	Deaktiviert (wird von außen gesteuert)
Postprocessing:	Deaktiviert
Datei signieren:	**Aktiviert**
Technostring:	leer
Dateiinformation:	leer
Verzeichnis:	Wählen Sie ein Verzeichnis auf einem lokalen Laufwerk mittels Browser-Button, z. B. „d:\dat\ibaLogic\".
Dateinamen:	Übernehmen Sie die Vorgabe.
Erfassungszeit:	Geben Sie die Taskintervallzeit an: „0,02" s

➲ Signalkonfiguration

Name:	„module_01"
Modus:	„Unbuffered"
Werte:	1 (nicht relevant)
Digitalwerte:	8
Datentyp:	REAL
Analogwerte:	8



➲ Klicken Sie mit der Maus in den Signaldefinitionsbereich. Dadurch wird das Modul mit 8 Digital- und 8 Real-Werten angelegt. Die Namen werden mit „Digital_xx" und „Analog_xx" vorbelegt.
Sie können die Standardsignalnamen umbenennen, z. B. nach „Sinus".



16.2.1.2 Schritt 2: Beschaltung des DFW

➲ Beenden Sie den Bausteineditor.

➲ Verbinden Sie den Anschluss „STORE_FILE" mit dem Ausgang eines Bausteins „SWITCH".

➲ Verbinden Sie das erste Messsignal, z. B. den Ausgang des Sinusgenerators (Datentyp REAL) mit dem Eingangskonnektor „DATA".

- Es öffnet sich ein Auswahlmenü, aus dem Sie das Modul wählen können, an den Sie das Messsignal anbinden wollen.
- Wenn Sie den Mauszeiger auf das gewünschte Modul ziehen, dann öffnet sich ein weiteres Auswahlmenü, aus dem Sie das gewünschte Signal (z. B. Sinus) auswählen.
- Ein weiteres Auswahlmenü wird geöffnet.
  Wählen Sie den Menüpunkt „data: REAL" aus.

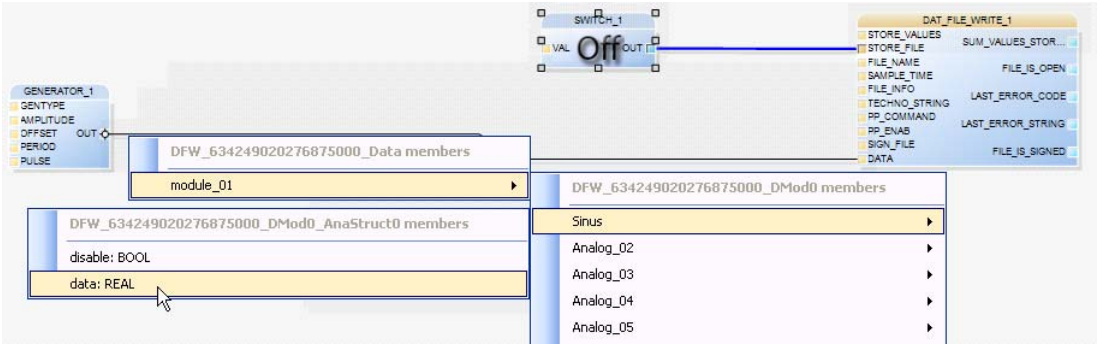


Abbildung 144: Messsignalzuweisung

Hinweis

Lassen Sie sich nicht durch den Namen der intern generierten Datenstruktur irritieren. Sie können ihn ignorieren (solange Sie nicht versuchen, die Joiner in ST nachzuprogrammieren, siehe unten).

Ergebnis

ibaLogic erzeugt daraufhin die Joiner, mit denen das gewählte Element aus den intern erzeugten Datenstrukturen adressiert wird.

Genauere Beschreibung von Joiner und Splitter finden Sie in "Konvertierer, Splitter, Joiner , Seite 163".

Es werden folgende „Joiner" erzeugt:

- ❑ „Module Joiner"
- ❑ „Signal Joiner"
- ❑ „Signal Property Joiner"

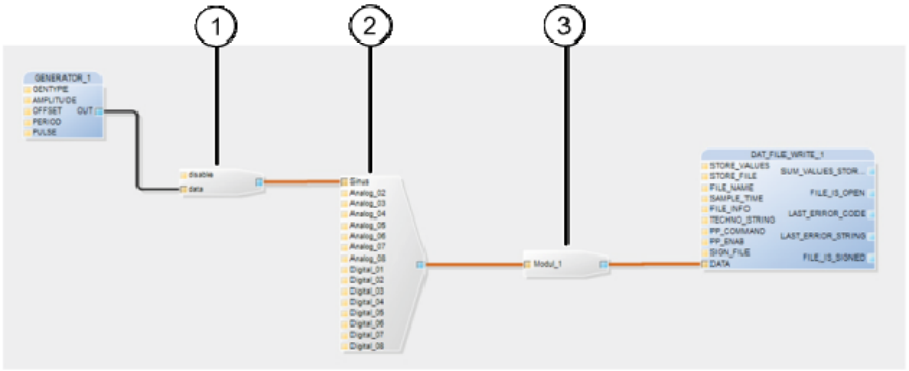


Abbildung 145: Joiner

1	Signal Property Joiner	3	Module Joiner
2	Signal Joiner		

16.2.1.3 Schritt 3: Legen Sie weitere Messsignale an

Die weiteren Signale können Sie direkt mit dem Signal-Joiner verbinden.

Hinweis

Verwenden Sie nicht die „disable"-Eingänge der „Signal Property Joiner", um die Aufzeichnung einzelner Signale zeitweise auszuschalten. Dies führt zu einer verfälschten Aufzeichnung.

Da die einzelnen Samples keinen Zeitstempel haben, wird der Signalverlauf immer kontinuierlich dargestellt. Die gewünschte Lücke ist dann am Ende der Messdatei.

16.2.1.4 Schritt 4: Starten der Aufzeichnung

➲ Starten Sie den PMAC.

➲ Setzen Sie den SWITCH am DAT_FILE_WRITE.STORE_FILE auf „On".

Ergebnis

Sie erkennen die laufende Aufzeichnung an dem inkrementierenden Wert des Anschlusses „DAT_FILE_WRITE_1.SUM_VALUES_STORED". Überprüfen Sie das Ergebnis, indem Sie die erzeugte Messdatei mit „ibaAnalyzer" öffnen.

16.2.1.5 Alternative: Joiner in ST programmieren

Um das Layout übersichtlicher zu gestalten, können Sie natürlich auch die Zuweisung der Messsignale an den Konnektor DATA in einem ST-Funktionsblock programmieren.

Beispiel, analog zur oben ausgeführten Konfiguration:

➲ Gehen Sie mit der Maus auf den Konnektor DATA des DFW-Bausteins. Der Tooltip zeigt Ihnen den intern generierten Struktur-Datentyp an, z. B. „DFW_634094511415156250_Data". Notieren Sie diesen!

➲ Generieren Sie einen Funktionsblock mit einer oder mehreren Eingangsvariablen vom Typ REAL und einer Ausgangsvariablen vom Typ des Konnektors DATA des DFW-Bausteins.

- Aktivieren Sie „Intellisense".
- Beginnen Sie mit der Zuweisung, schreiben Sie „o1.".
- Sobald Sie den Punkt eingegeben haben, zeigt Ihnen ibaLogic die Strukturelemente an (hier „modul_01"), da „o1" ein Strukturdatentyp ist. Wählen Sie die angebotenen Elemente mit den Cursortasten auf /ab , übernehmen Sie es mit „tab" und setzen Sie einen Punkt dahinter.
- Da auch „module_01" eine Struktur ist, zeigt Ihnen ibaLogic deren Strukturelemente an, hier alle Digital- und Analogsignale. Wählen Sie „Sinus" und setzen Sie einen Punkt dahinter.
- "Sinus" ist wiederum ein Strukturdatentyp, daher ...., zeigt Ihnen ibaLogic die Strukturelemente „data" und „disable" an. Wählen Sie „data" und fahren Sie fort mit der Zuweisung „ := i1;".

Ergebnis

Damit haben Sie das erste Signal fertiggestellt. Weitere Signale können Sie auf dieselbe Weise eintragen. Das Ergebnis sollte dann so aussehen:

```
1 o1.module_01.Sinus.data := i1;
2 o1.module_01.Analog_02.data := i2;
```

Sie können nur die Eingänge mit den dazu passenden Messsignalen und den Ausgang mit dem Konnektor DATA des DFW-Bausteins verbinden.

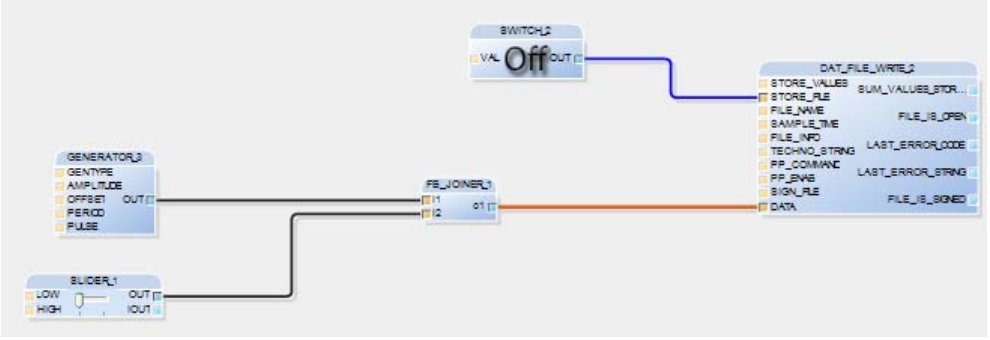


Abbildung 146: Beispielschaltung „Messsignalzuweisung"

Hinweis

Falls Sie nachträglich die Signalkonfiguration des DFW ändern wollen, müssen Sie die Verbindung am Konnektor DATA lösen und nach der Änderung dem Ausgangskonnektor des FB_JOINER den neuen Datentyp zuweisen. Evtl. müssen Sie auch die Zuweisungen in ST ändern.

16.2.2 DAT_FILE_WRITE im „Buffered Mode"

Im Modus „Buffered" (Erläuterungen siehe "Buffered Mode , Seite 195") wird bei jedem Speicherzyklus ein Array von n Daten-Samples gespeichert. Die zu speichernden Signale müssen als Arrays zur Verfügung stehen. Das hat zur Folge, dass einige Parameter des DFW-Bausteins eine leicht geänderte Bedeutung haben.

Verwenden Sie diesen Modus, wenn der Abtastzyklus der zu speichernden Daten schneller ist, als der schnellste im ibaLogic-Taskzyklus.

Aufgabe: Speichern 8 Analogwerte von ibaPADU8.

Voreingestellte Parameter

- Abtastrate ibaPADU8: 1 ms
- ibaPADU8 an ibaFOB-Link0, Modus Integer „Buffered"
- Interrupt-Zeitbasis: 1 ms
- Zielsystem Win XP
- Taskintervall: 20 ms

16.2.2.1 Schritt 1: Parametrierung der gepufferten Eingänge

➲ Parametrierung allgemein:
Für die Parametrierung der Datenpufferung / Datenübernahme sind die folgenden Hardwaresignale vorhanden (siehe Beschreibung "Buffered Mode , Seite 195"):

- Ausgangssignale
  „…DataSize", „…Ratio", „…RequestBuffer"
- Eingangsignale
  „…CurDataSize", „…FillCount"

➲ Wählen Sie die Parameter nach folgenden Gesichtspunkten:

- Die Arraytiefe im DFW muss größer oder gleich der „DataSize" sein.
- Erfassungszeit (im DFW) muss gleich sein der (Arraytiefe * Sampletime).
- Die „DataSize" muss so gewählt werden, dass folgende Bedingung erfüllt ist:

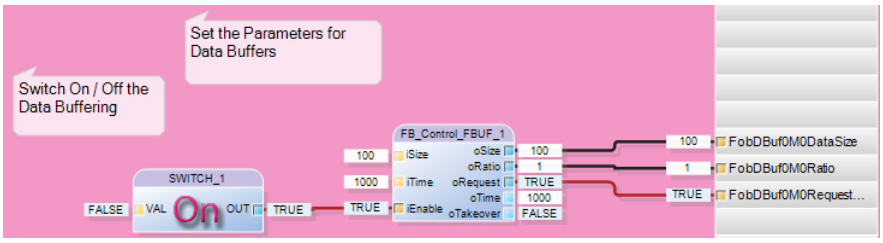
  Taskintervall ≤ Sampletime * DataSize / Ratio / 3

  (Der Faktor „3" dient zur Sicherstellung, dass keine Samples verloren gehen, auch wenn der Task verdrängt wird.)
- Für das Beispiel wählen Sie DataSize = 100,
  also muss sein: „Taskintervall ≤ 1ms * 100 / 1 / 3",
  d. h. das Taskintervall muss kleiner als 33 ms sein. Wählen Sie 20 ms.

➲ Legen Sie einen Anwenderbaustein an, mit dem Sie die Ausgangssignale für den gepufferten Modus vorbesetzen können, zum Beispiel

```
1   if (iEnable = TRUE) then
2      if (iEnable <> v1) then
3    oSize := iSize;
4    oRatio := 1;
5    oTime := iTime;
6    oTakeover := TRUE;
7      else
8    oTakeover := FALSE;
9      end_if;
10     oRequest := TRUE;
11  else
12     oRequest := FALSE;
13  end_if;
14
15  v1 := iEnable;
16
17
```

➲ Verbinden Sie den Baustein mit den Ausgangssignalen



(Die Konnektoren iTime, oTime und oTakeover werden hier nicht benötigt.)

16.2.2.2 Schritt 2: Parametrieren Sie den DFW-Baustein „Allgemeine Konfiguration"

➲ Gehen Sie offline.

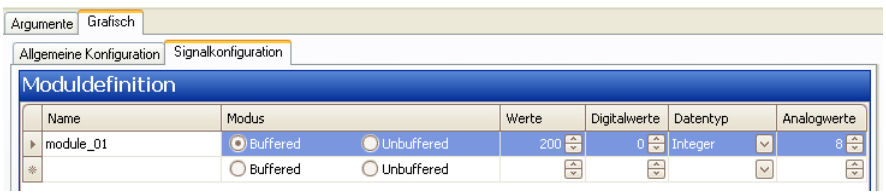
➲ Allgemeine Konfiguration

Asynchr. Zugriff:	Deaktiviert
Speicherzyklus:	10 (nicht relevant)
Offset Startzeit:	0
Werte speichern:	**Aktiviert**
Datei schreiben:	Deaktiviert (wird von außen gesteuert)
Postprocessing:	Deaktiviert
Datei signieren:	**Aktiviert**
Technostring:	leer
Dateiinformation:	leer
Verzeichnis:	Wählen Sie ein Verzeichnis auf einem lokalen Laufwerk mittels Browser-Button, z. B. „d:\dat\ibaLogic\"
Dateinamen:	Übernehmen Sie die Vorgabe.
Erfassungszeit:	Geben Sie den Speicherintervall an. Speicherintervall = Arraytiefe * Abtastzeit Wählen Sie eine Arraytiefe von 200, daraus ergibt sich die Speicherzeit von 0,2 s.

➲ Signalkonfiguration

Name:	„module_01"
Modus:	„Buffered"
Werte:	200
Digitalwerte:	8
Datentyp:	Integer
Analogwerte:	8

- In der Spalte „Werte" geben Sie die Arraygröße von 200 an. Das ergibt eine Speicherzeit von 200 ms (siehe oben).

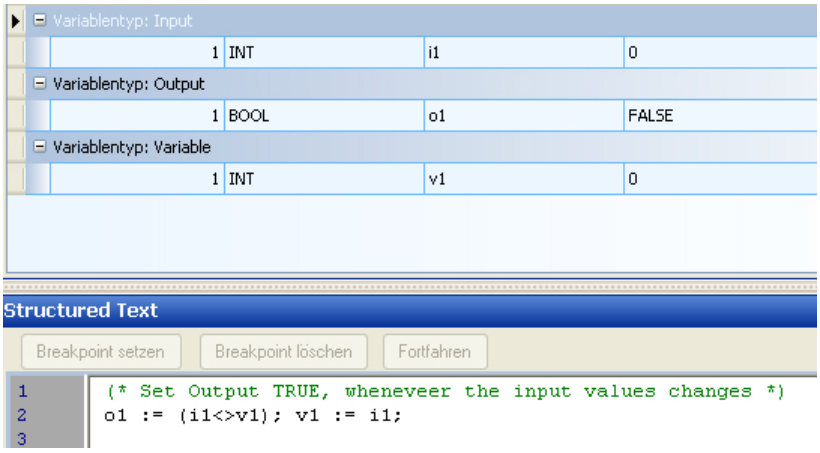


➲ Gehen Sie online.

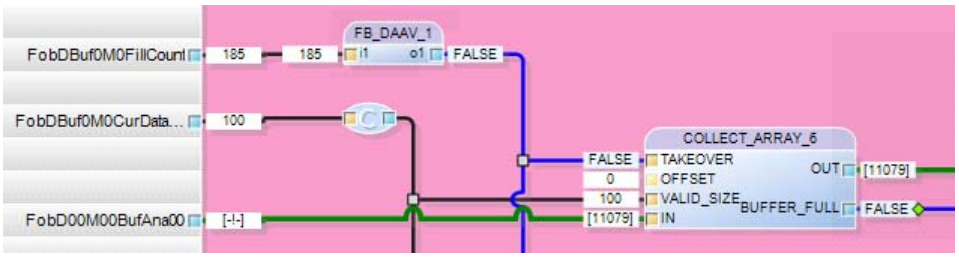
16.2.2.3 Schritt 3: Übernehmen der gepufferten Eingangssignale

➲ Ziehen Sie einen Baustein „COLLECT_ARRAY" aus dem Funktionsbausteinordner „Type Conversion" in das Arbeitsfenster.
Der Baustein dient zum Umspeichern des Eingangsarrays (Datentyp FOBFBUF_INT) in das Array für den DFW.

➲ Zur Erzeugung des Signals „TAKEOVER" für den COLLECT_ARRAY legen Sie einen Anwenderbaustein (FB_DAAV) mit folgenden Eigenschaften an:

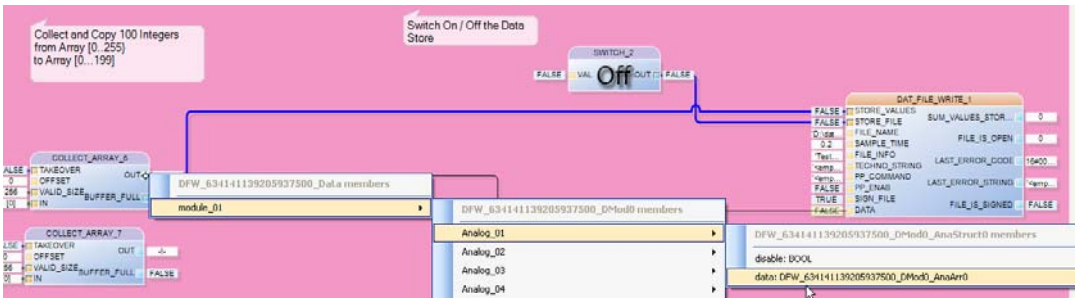


➲ Verschalten Sie die Bausteine wie auf dem nachfolgenden Screenshot.



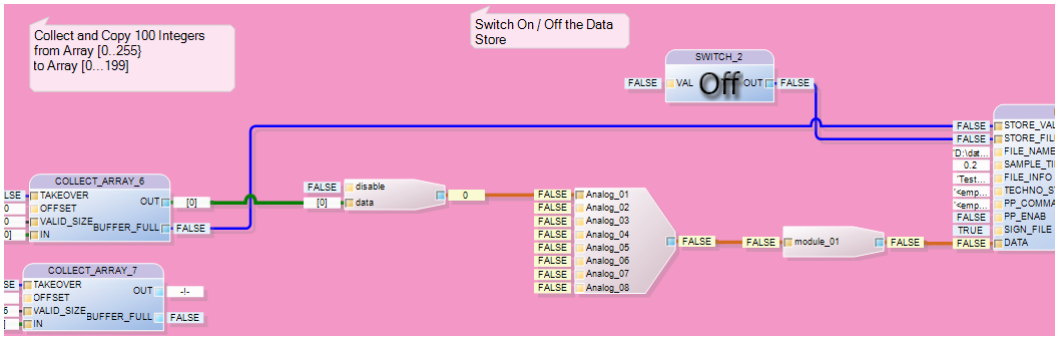
16.2.2.4 Schritt 4: Übergeben der Daten an den DAT_FILE_WRITE

- Legen Sie einen Baustein SWITCH an und verbinden Sie dessen Ausgang mit DAT_FILE_WRITE.STORE_FILES.
- Verbinden Sie das COLLECT_ARRAY.BUFFER_FULL mit DAT_FILE_WRITE.STORE_VALUES.
- Verbinden Sie den Ausgang COLLECT_ARRAY.OUT mit DAT_FILE_WRITE.DATA.
  ibaLogic blendet die Auswahlmenüs für die Joiner ein.
  Wählen Sie hier die „module_01 - Analog_01 - … AnaArr0".



Ergebnis

Die Daten werden an DAT_FILE_WRITE übergeben.

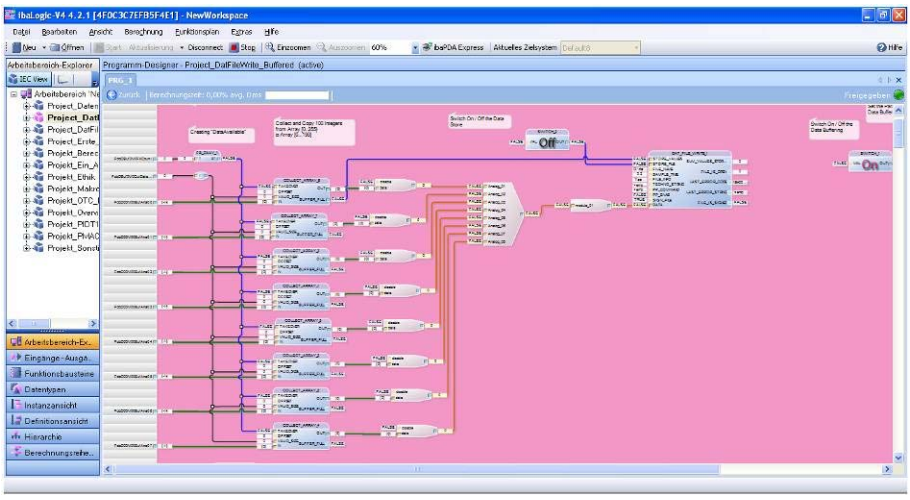


16.2.2.5 Schritt 5: Verdrahten der restlichen Eingänge

- Kopieren Sie für jeden Analogeingang einen COLLECT_ARRAY-Baustein.
- Verbinden Sie alle COLLECT_ARRAY.TAKEOVER mit dem DataAvailabe-Signal (FB_DAAV.o1).
- Verbinden Sie alle COLLECT_ARRAY.VALID_SIZE mit dem Ausgang des Konverters nach „ ... CurDataSize".
- Verbinden Sie jeden COLLECT_ARRAY.IN mit dem jeweiligen gepufferten Analogeingang „ … BufAna*nn*"
- Verbinden Sie jeden COLLECT_ARRAY.OUT mit dem jeweiligen Konnekor „Analog_*nn*" des Joiners.

16.2.2.6 Schritt 6: Starten der Aufzeichnung

➲ Setzen Sie den SWITCH am DAT_FILE_WRITE.STORE_FILE auf „On".



Ergebnis

Sie erkennen die laufende Aufzeichnung an dem inkrementierenden Wert des Anschlusses „DAT_FILE_WRITE_1.SUM_VALUES_STORED". Überprüfen Sie das Ergebnis, indem Sie die erzeugte Messdatei mit „ibaAnalyzer" öffnen.

Tipps

1. Sie können parallel zu den gepufferten Eingängen die nicht gepufferten Eingänge verwenden, um z. B. mit dem ibaPDA Express die Analogsignale zu visualisieren.
2. Überprüfen Sie die Messdatei: Entspricht die Zeitskala nicht der tatsächlich aufgezeichneten Zeit, so ist entweder die Erfassungszeit falsch eingestellt oder Taskintervall, DataSize und Arraytiefe sind nicht richtig aufeinander abgestimmt.

Dokumentation

Das obige Beispiel ist der Liefer-CD beigelegt.

17 Namenskonventionen

Ein Name ist eine Kette von Buchstaben, Ziffern und einem Unterstrich. Dabei gelten folgende Regeln:

- ❑ Groß- und Kleinschreibung ist nicht relevant, z. B. ABCD und abcd sind identisch.
- ❑ Namen müssen mit Buchstaben oder Unterstrich beginnen. Eine Ziffer am Anfang ist nicht erlaubt.
- ❑ Unterstriche sind signifikant in Namen, z. B. sind A_BCD und AB_CD unterschiedliche Namen (im Gegensatz dazu in Zahlenkonstanten).
- ❑ Unterstriche am Ende eines Namens sind nicht erlaubt, z. B. ABCD_
- ❑ Mehrfache Unterstriche sind nicht erlaubt, z. B. AB__CD
- ❑ Schlüsselworte, z. B. for und if, sind nicht erlaubt.

Besonderheiten bei ibaLogic:

Namen mit nur einem Buchstaben sind nicht erlaubt.

Hinweis

Diese Regeln gelten allgemein in ibaLogic, auch außerhalb von Funktionsbausteinen, z. B. bei den Namen von OTCs, IPCs, Ein- und Ausgangssignalen, Funktionsbausteinnamen etc.

18 Datentypen

18.1 Standard-Datentypen

ibaLogic unterstützt die folgenden elementaren Datentypen:

Typ	Bereich (min)	Bereich (max)	Erklärung
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 Wort
SINT	-128	127	8-bit Integer mit Vorzeichen
USINT	0	255	8-bit Integer ohne Vorzeichen
INT	-32768	32767	16-bit Integer mit Vorzeichen
UINT	0	65535	16-bit Integer ohne Vorzeichen
DINT	-2147483648	2147483647	32-bit Integer mit Vorzeichen
UDINT	0	4294967295	32-bit Integer ohne Vorzeichen
REAL	1.175494351 e-38	3.402823466 e+38	Gleitpunkt, einfache Genauigkeit, 32 bit
LREAL	2.225073858 … e-308	1.797693134862 … e+308	Gleitpunkt, doppelte Genauigkeit, 64 bit
TIME	-2147483648ms -24d_20h_31m_23s_648ms	2147483648ms 24d_20h_31m_23s_648ms	Zeit, intern abgebildet als DINT mit 1 ms Auflösung per Inkrement
STRING	0	250 Zeichen (249 für den Anwender, wegen NULL-Kennung am Ende)	Zeichenfolge mit Anzahl der Zeichen incl. Ende-Kennung (NULL).

18.2 Abgeleitete Datentypen

Typ	Erklärung
DIRECT_DERIVED	Elementare Datentypen mit festem Wert (Konstante)
SUBRANGE	Integer-Datentypen mit eingeschränktem Wertebereich
STRING_DERIVED	String mit festem Wert und Länge

18.3 Generische Datentypen

Typ	Erklärung
ENUM	Aufzählungstyp, statt Werte werden Namen definiert.
ARRAY	Struktur, bestehend aus einer beliebigen Folge eines der o.g. Datentypen (mit Ausnahme des String, der bereits ein Array darstellt); maximal vierdimensional maximale Anzahl Elemente: 32767
STRUCT	Struktur, bestehend aus einer beliebigen Folge der o.g. Datentypen maximale Anzahl Elemente: 1048576

Wichtiger Hinweis

Intern ist der Speicher auf eine interne Speicherbegrenzung von 63 kByte pro Betrachtungsebene begrenzt. So darf z. B. der Speicherplatzbedarf aller Eingangs- und Ausgangsvariablen eines Funktionsblocks diese Größe nicht überschreiten.

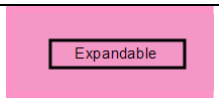
Weitere Informationen siehe "Leistungsgrenzen , Seite 249".

19 Standard-Funktionsbausteine

In diesem Abschnitt finden Sie eine tabellarische Übersicht über alle Funktionen und Funktionsbausteine, die in ibaLogic-V4 zur Verfügung stehen.

19.1 Tabelleninterpretation

In diesem Abschnitt finden Sie Hinweise zur Interpretation der tabellarischen Übersicht.

Spalte	Erklärung
Eingangsdatentyp	In den Spalten Eingangsdatentypen werden für jeden Konnektor die zulässigen Datentypen aufgeführt. Es gibt Bausteine, deren Konnektoren nicht von vorneherein auf einen Datentyp festgelegt sind, sondern deren Datentyp erst festgelegt wird, wenn eine Verbindung zu einem anderen Konnektor gezogen wird.
Bausteingrafik	
Grün 	Funktionen bzw. Funktionsbausteine sind definiert in der Norm IEC 61131-3.
Gelb 	Funktionen bzw. Funktionsbaustein sind definiert durch iba AG.
	Dieser Baustein ist erweiterbar, d. h. die Anzahl der Eingänge kann verändert werden. Öffnen Sie den Baustein durch Doppelklick und ändern Sie die „Anzahl Eingänge".
Ausgangsdatentyp	In den Spalten Ausgangsdatentypen werden für jeden Konnektor die zulässigen Datentypen aufgeführt. Es gibt Bausteine, deren Konnektoren sind nicht von vorneherein auf einen Datentyp festgelegt, sondern der Datentyp wird erst festgelegt, wenn eine Verbindung zu einem anderen Konnektor gezogen wird.
Erklärung, Beispiel, ST-Syntax	Zu jeder Funktion finden Sie einen Hinweis, ob und wie die Funktion innerhalb ST aufrufbar ist. Es können auch Funktionen als mehrzeiliger ST-Code nachgebildet werden. Dies wird aber nicht ausgeführt.

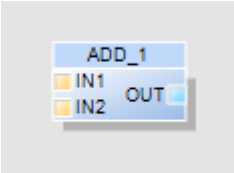
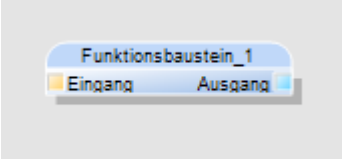
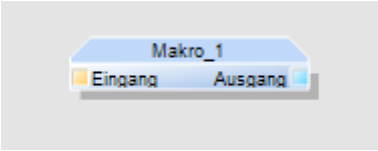
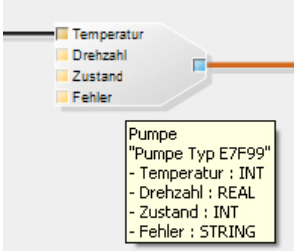
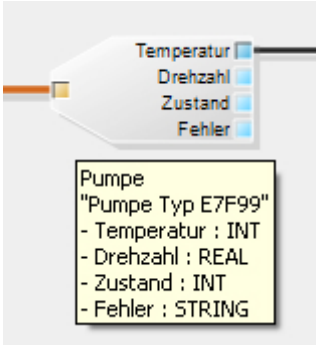
19.2 Datentypen

Für diese „untypisierten" Konnektoren wird der Datentyp „ANY" mit folgenden Varianten angezeigt:

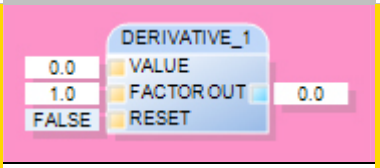
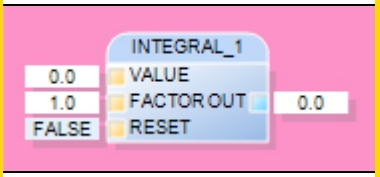
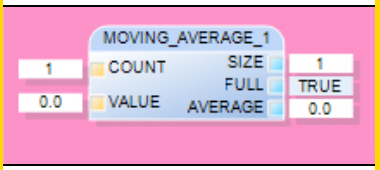
Datentyp „ANY" mit „untypisierten" Konnektoren	Erklärung
Any_Int	Alle Integer-Typen (SINT, INT, DINT, USINT, UINT, UDINT)
Any_Real	Alle Real-Typen (REAL, LREAL)
Any_Num	Alle numerischen Datentypen (alle Integers und Reals)
Any_Magnitude	Alle numerischen Datentypen und Type TIME
Any_Bit	Alle bitorientierten Typen (BOOL, BYTE, WORD, DWORD)
Any_String	Datentyp STRING
Any_Elementary	Alle elementaren Typen (Integers, Reals, TIME, STRING)
Any_Derived	Alle elementaren Datentypen, Arrays und Strukturen
Any	Jeder beliebige Datentyp

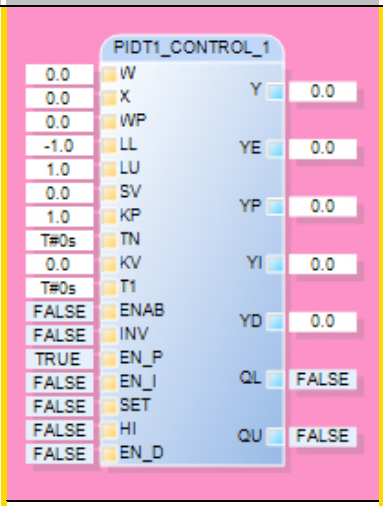
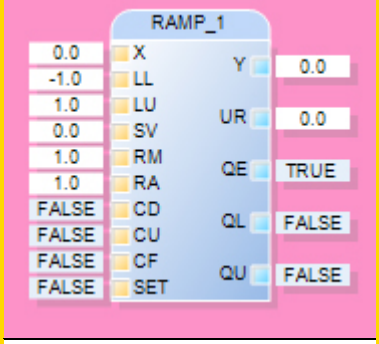
19.3 Bausteinart mit Funktionsplandarstellung

Funktionen, Funktionsbausteine und Makroblöcke werden im Programmierbereich wie folgt dargestellt:

Bausteinart mit Funktionsplandarstellung	Erklärung
Funktion 	Eine Funktion erkennt man an den Ecken.
Funktionsbaustein 	Einen Funktionsbaustein erkennt man an den abgerundeten Ecken.
Makroblock 	Einen Makroblock erkennt man an den abgeflachten Ecken.
Automatischer Typ-Konverter 	Einen automatischen Typ-Konverter erkennt man an dem Buchstaben „C" im Symbol.
Automatisch erzeugter Struktur-Joiner 	Automatisch erzeugter Struktur-Joiner Wird automatisch erzeugt, sobald man versucht eine Einzelgröße mit einer Struktur zu verbinden.
Automatisch erzeugter Struktur-Splitter 	Automatisch erzeugter Struktur-Splitter Wird automatisch erzeugt, sobald man versucht eine Einzelgröße mit einer Struktur zu verbinden.

19.4 Analytische Funktionen

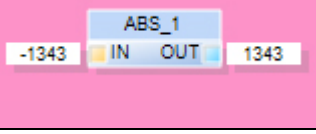
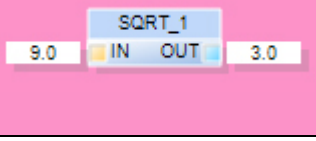
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Real Real Bool		Real	**DERIVATIVE:** Ableitung eines Wertes nach der Zeit Ausgangswert „OUT" ist die Ableitung des Eingangswert „VALUE", multipliziert mit einem Faktor „FACTOR" der Dimension Zeit. Mit Eingang „RESET" = „TRUE" wird der Ausgang zurückgesetzt. Implementierung: OUT:=(VALUEn-VALUEn-1)*FACTOR **ST:** nicht aufrufbar
Real Real Bool		Real	**INTEGRAL:** Werteintegral über die Zeit Ausgangswert „OUT" ist das Integral des Eingangswertes „VALUE", multipliziert mit einem Faktor „FACTOR" der Dimension Zeit. Mit Eingang „RESET" = „TRUE" wird der Ausgang zurückgesetzt. Implementierung: OUTn:=OUTn-1+(VALUEn*FACTOR); **ST:** nicht aufrufbar
Dint Real		Dint Bool Real	**MOVING_AVERAGE:** Gleitender Mittelwert Der Eingangswert „COUNT" definiert eine Anzahl Werte (= Samples), die für eine Mittelwertberechnung des Wertes „VALUE" herangezogen werden. Der Ausgangswert „SIZE" zeigt die Anzahl der zur Mittelwertberechnung verwendeten Werte. Der Ausgang „FULL" ist „TRUE", wenn die vorgegebene Werteanzahl erreicht ist. Der Ausgangswert „AVERAGE" gibt den kumulierten Mittelwert an. AVERAGE:= (Summe der letzten SIZE Werte) / SIZE. Die Mittelwertberechnung wird kontinuierlich durchgeführt. Die Anzahl der Werte kann jederzeit verändert werden. **ST:** nicht aufrufbar

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Lreal Lreal Lreal Lreal Lreal Lreal Lreal Time Lreal Time Bool Bool Bool Bool Bool Bool Bool		Lreal Lreal Lreal Lreal Lreal Bool Bool	**PIDT1_CONTROL:** PIDT1-Regelungsbaustein Universeller PIDT1-Regler, umschaltbar auf die Betriebsarten P-, I-, PI-, PIDT1-Regler. Eine detailierte Beschreibung finden Sie in "PIDT1_CONTROL , Seite 108". **ST:** nicht aufrufbar
Lreal Time		Lreal	**PT1:** Verzögerungsglied 1. Ordnung Die Eingangsgröße X wird, dynamisch verzögert um die Glättungszeitkonstante T1, auf den Ausgang Y gegeben. Implementierung: ti: = time_to_lreal(T1) / time_to_lreal(EvalDeltaTime[7]); Y: = 1.0 / (1.0 + t1) * (X + ti * Yold); Yold: = Y; **ST:** nicht aufrufbar
Lreal Lreal Lreal Lreal Lreal Lreal Bool Bool Bool Bool		Lreal Lreal Bool Bool Bool	**RAMP:** Rampenbaustein Baustein mit zwei verschiedenen Rampen, Manuell- und Automatik-Modus. Eine detailierte Beschreibung finden Sie in „RAMP , Seite 116". **ST:** nicht aufrufbar

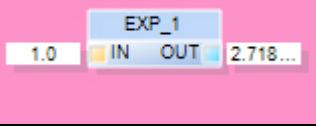
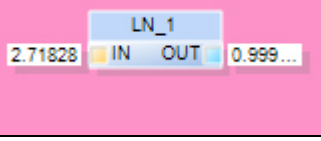
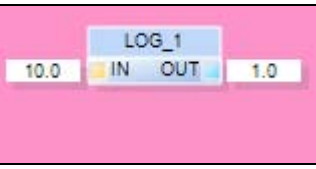
[7] EvaldeltaTime ist die Zeit zwischen zwei Bearbeitungszyklen (intern errechnet).

19.5 Arithmetische Funktionen

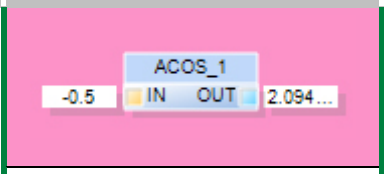
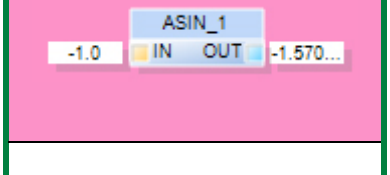
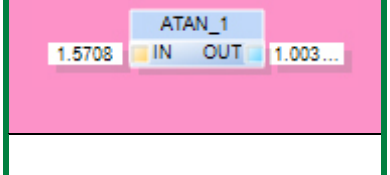
19.5.1 General

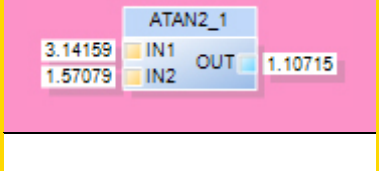
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_Num		Any_Num	**ABS:** Absolutwert Beispiel: +1343 = abs(-1343); **ST:** `OUT := abs(IN);`
Any_Real		Any_Real	**SQRT:** Quadratwurzel Beispiel: +3.0 = sqrt(9.0); **ST:** `OUT:= sqrt(IN);`

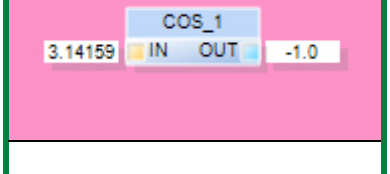
19.5.2 Logarithmic

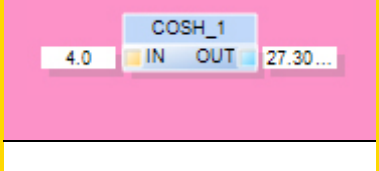
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_Real		Any_Real	**EXP:** Natürliches Exponential zur Basis e Ergebnis: = e^{arg}; Beispiele: 2.71828 = exp(1.0); 0.13533 = exp(-2.0); **ST:** `OUT:= exp(IN);`
Any_Real		Any_Real	**LN:** Natürlicher Logarithmus Beispiel: +1.0 = ln(2.71828); **ST:** `OUT:= ln(IN);`
Any_Real		Any_Real	**LOG:** Logarithmus zur Basis 10 Beispiel: +1.0 = log(10.0); **ST:** `OUT:= log(IN);`

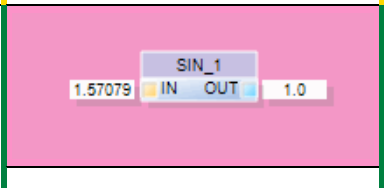
19.5.3 Trigonometric

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_Real		Any_Real	**ACOS:** Arcus-cosinus Beispiel: 1.57079 = acos(0.0); **ST:** `OUT:= acos(IN);`
Any_Real		Any_Real	**ASIN:** Arcus-sinus Beispiel: -1.57079 = asin(-1.0); **ST:** `OUT:= asin(IN);`
Any_Real		Any_Real	**ATAN:** Arcus-tangens Beispiel: 1.0000 = atan(π/2.0); **ST:** `OUT:= atan(IN);`

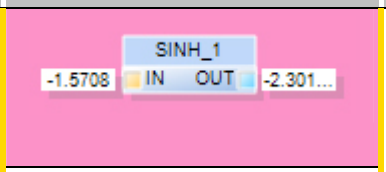
Any_Real
Any_Real
		Any_Real	**ATAN2:** Arcus-tangens
			Beispiel:
			1.1071 = atan2_real(**p**,**p**/2.0);
			ST:
			unterschiedliche Aufrufe für Datentypen REAL und LREAL
			`OUT:= atan2_real(IN1, IN2);`
			`OUT:= atan2_lreal(IN1, IN2);`

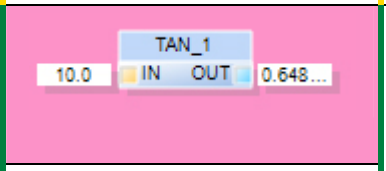
Any_Real		Any_Real	**COS:** Cosinus Beispiel: -1.0000 = cos(π); **ST:** `OUT:= cos(IN);`

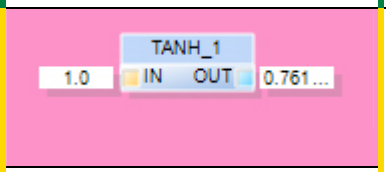
Any_Real		Any_Real	**COSH:** Cosinus hyperbolicus
			Beispiel:
			+27.3082 = cosh_real(4.0);
			ST:
			unterschiedliche Aufrufe für Datentypen REAL und LREAL
			`OUT:= cosh_real(IN);`
			`OUT:= cosh_lreal(IN);`

Any_Real		Any_Real	**SIN:** Sinus Beispiel: 1.0 = sin(π/2); **ST:** `OUT:=sin(IN);`

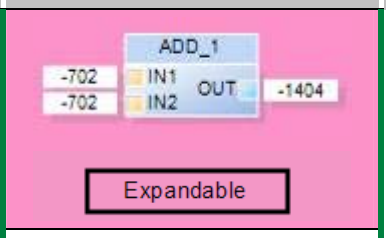
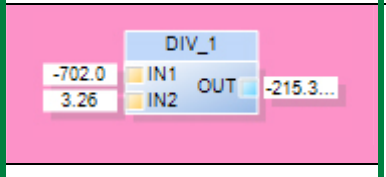
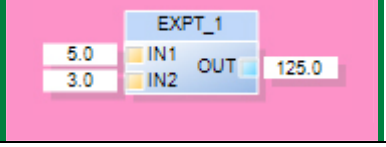
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

Any_Real		Any_Real	**SINH:** Sinus hyperbolicus
			Beispiel:
			-2.3013 = sinh_real(-π/2.0);
			ST:
			unterschiedliche Aufrufe für Datentypen REAL und LREAL
			`OUT:=sinh_real(IN);`
			`OUT:=sinh_lreal(IN);`

Any_Real		Any_Real	**TAN:** Tangens Beispiel: 0.648 = tan(10.0); **ST:** `OUT:=tan(IN);`

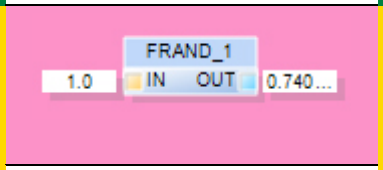
Any_Real		Any_Real	**TANH:** tangens hyperbolicus
			Beispiel:
			0.76159 = tanh_real(1.0);
			ST:
			unterschiedliche Aufrufe für Datentypen REAL und LREAL
			`OUT:=tanh_real(IN);`
			`OUT:=tanh_lreal(IN);`

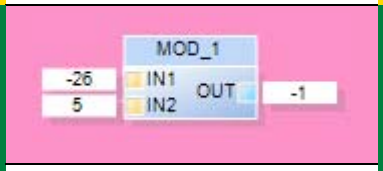
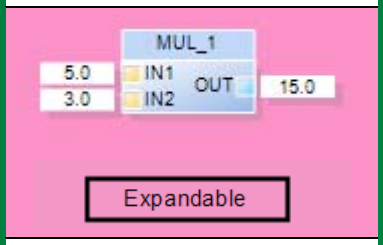
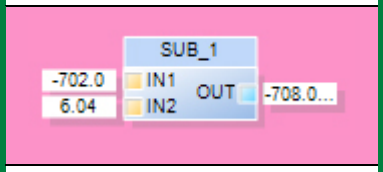
19.5.4 Sonstige

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_ Magnitude Any_ Magnitude		Any_ Magnitude	**ADD:** Addition Beispiel: -1404 = -702 + -702; **ST:** Operator verwenden: `OUT:= IN1 + IN2 + ... + INn;`
Any_Num Any_Num		Any_Num	**DIV:** Division Beispiel: -215.3 = -702.0 / 3.26; **ST:** Operator verwenden: `OUT:= IN1 / IN2;` **Achtung:** Ist der Divisor IN2 = 0, wird das Ergebnis 0 und im Ereignisfenster wird zyklisch eine Fehlermeldung „Division by Zero" ausgegeben.
Any_Real Any_Num		Any_Real	**EXPT:** Allgemeines Exponential zur Basis (IN2) Ergebnis: = $arg1^{arg2}$;

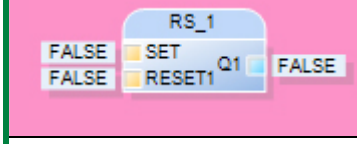
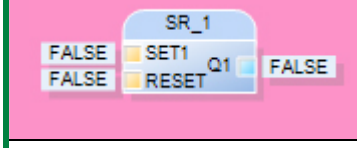
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

			Beispiele:
			125.0 = expt(5.0, 3.0);
			4.0 = 16.0 ** 0.5;
			ST:
			`OUT := expt(IN1, IN2);`
			oder
			`OUT := IN1 ** IN2;`

Any_Real		Any_Real	**FRAND:** Zufallszahl im Bereich {0 ... arg}
			Beispiele:
			+0.07116 = frand_real(1.00);
			+2.92457 = frand_lreal(6.00);
			ST: unterschiedliche Aufrufe für Datentypen REAL und LREAL
			`OUT:= frand_real(IN);`
			`OUT:= frand_lreal(IN);`

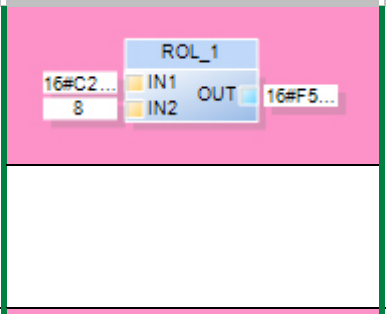
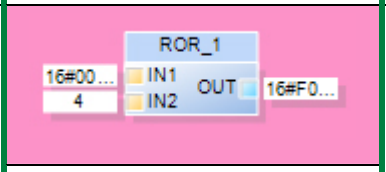
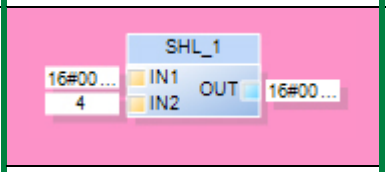
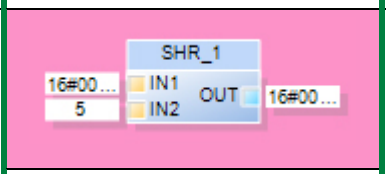
Any_Int Any_Int		Any_Int	**MOD:** Divisionsrest (Modulo) Beispiele: -1 = -26 mod 5; **ST:** Operator verwenden: `OUT:= IN1 mod IN2;`
Any_Num Any_Num		Any_Num	**MUL:** Multiplikation Beispiele: 15.0 = 5.0 * 3.0; 4 = 2 * 2; **ST:** Operator verwenden `OUT:= IN1 * IN2 * … * INn;`
Any_ Magnitude Any_ Magnitude		Any_ Magnitude	**SUB:** Subtraktion Beispiel: -708.04 = -702 – 6.04; **ST:** Operator verwenden: `OUT:= IN1 – IN2;`

19.6 Bistable

<table>
<tr><th>Eingangs-datentyp</th><th>Bausteingrafik</th><th>Ausgangs-datentyp</th><th>Erklärung, Beispiel, ST-Syntax</th></tr>
<tr><td>Bool
Bool</td><td> 
</td><td>Bool</td><td>RS: RS-Flip-Flop
(statischer Binärwertspeicher)
R-Dominant
Wahrheitstabelle:
<table>
<tr><th colspan="2">Eingangswerte</th><th>Ausgang</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>
ST: nicht aufrufbar</td></tr>
<tr><td>Bool
Bool</td><td> 
</td><td>Bool</td><td>SR: SR-Flip-Flop
(statischer Binärwertspeicher)
S-Dominant
Wahrheitstabelle:
<table>
<tr><th colspan="2">Eingangswerte</th><th>Ausgang</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>
ST: nicht aufrufbar</td></tr>
</table>

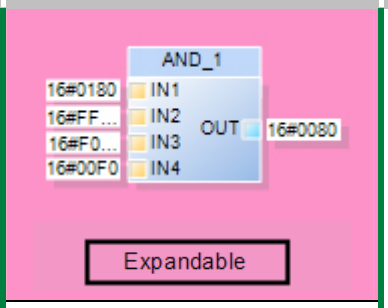
19.7 Bit-String

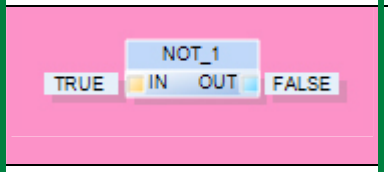
19.7.1 Bit-Shift

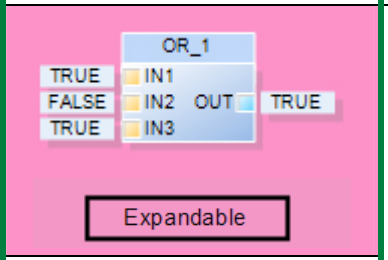
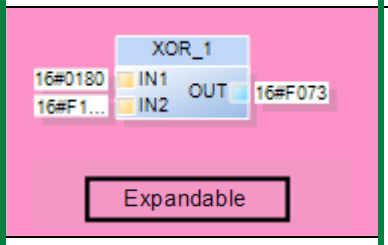
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_Bit Uint		Any_Bit	**ROL:** Links-Rotate von arg1 um arg2 Bits Beispiele: 16#F50000C2 =rol(16#C2F50000,8); 16#45678123 =rol(16#12345678,12); **ST:** `OUT := rol(IN1,IN2);`
Any_Bit Uint		Any_Bit	**ROR:** Rechts-Rotate von arg1 um arg2 Bits Beispiele: 16#F00000C2 = ror(16#C2F,4); 16#F500000C = ror(16#CF5,8); **ST:** `OUT := ror(IN1,IN2);`
Any_Bit Uint		Any_Bit	**SHL:** Links-Shift von IN1 um IN2 Bits, mit Nullen von rechts auffüllen Beispiel: 16#0D90 = shl(16#00D9,4); **ST:** `OUT := shl(IN1,IN2);`
Any_Bit Uint		Any_Bit	**SHR:** Rechts-Shift von IN1 um IN2 Bits, mit Nullen von links auffüllen Beispiele: 16#000C = shr(16#0180,5); 16#00D9 = shr(16#0D90,4); **ST:** `OUT := shr(IN1,IN2);`

19.7.2 Bitwise_Boolean

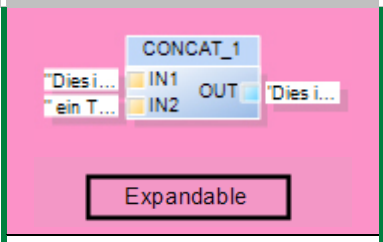
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

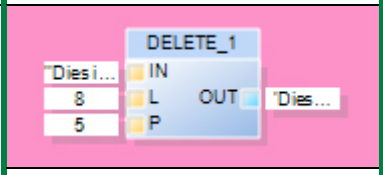
Any_Bit
Any_Bit
Any_Bit
Any_Bit
		Any_Bit	**AND:** Logische UND-Verknüpfung
			Beispiel:
			16#80=
			and(16#0180, 16#FFF0, 16#F0F0, 16#00F0);
			ST: Operator verwenden:
			`OUT := IN1 AND IN2 … AND INn;`

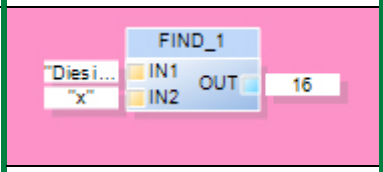
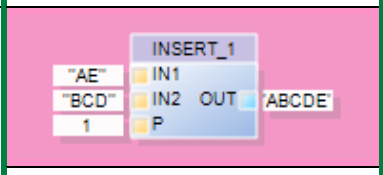
Any_Bit		Any_Bit	**NOT:** Logische NOT-Funktion
			Beispiele:
			FALSE = not(TRUE);
			16#FE7F = not(16#0180);
			ST:
			`OUT := not IN;`
			oder
			`OUT := not(IN);`

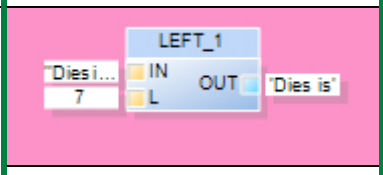
Any_Bit Any_Bit Any_Bit		Any_Bit	**OR:** Logische ODER-Verknüpfung Beispiele: 1 = or(1, 0, 1); 16#F3 = or(16#F0,16#03); **ST:** Operator verwenden: `OUT:= IN1 or IN2 or … INn;`
Any_Bit Any_Bit		Any_Bit	**XOR:** Logische XOR-Verknüpfung. Beispiele: FALSE = xor(TRUE,TRUE); 16#F073 = xor(16#0180,16#F1F3); **ST:** Operator verwenden: `OUT:= IN1 xor IN2 xor … INn;`

19.8 Character String

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_String Any_String		Any_String	**CONCAT:** Verknüpfung von Teilstrings Beispiele: 'Dies ist ein Text'= concat('Dies ist', ' ein Text') **ST:** `OUT:=concat(IN1, IN2,…,INn);`

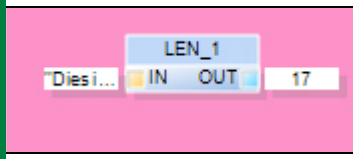
Any_String
Uint
Uint
		Any_String	**DELETE:**
			Löschen von L Zeichen eines Strings ab (inklusive) Position P. Das erste Zeichen hat die Position 1.
			Beispiele:
			'Dies Text'=
			delete('Dies ist ein Text',8,5);
			'DE' = delete('ABCDE',3,1);
			ST:
			`OUT:=delete(IN, L,P);`

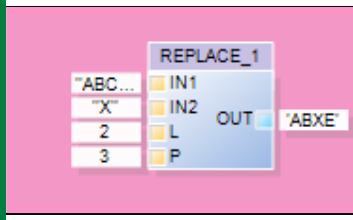
Any_String Any_String		Int	**FIND:** Suchen der ersten Übereinstimmung eines Zeichens IN2 im String IN1. Wird das Zeichen nicht gefunden, ist das Ergebnis = 0. Beispiele: 16 = find('Dies ist ein Text', 'x'); 1 = find('Dies ist ein Text', 'D'); **ST:** `OUT:=find(IN1, IN2);`
Any_String Any_String Uint		Any_String	**INSERT:** Einfügen von String IN2 in String IN1 nach Position P. Wenn P=0, wird IN2 am Anfang eingefügt. Beispiele: 'ABCDE' = insert('AE','BCD', 1); 'xABC' = insert('ABC','x',0); **ST:** `OUT:=insert(IN1, IN2,P);`

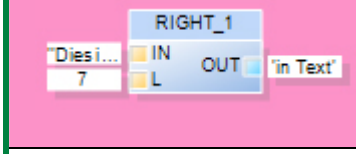
Any_String
Uint
		Any_String	**LEFT:**
			Linker Teil eines Strings IN mit Länge L
			Beispiel: 'Dies is'=
			left('Dies ist ein Text',7);
			ST:
			`OUT:=left(IN,L);`

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

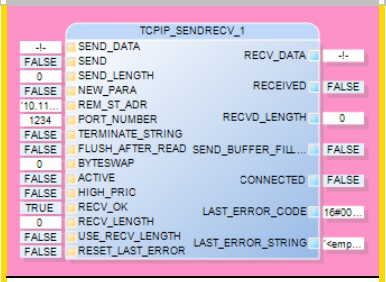
Any_String
Uint
Uint
		Any_String	**MID:**
			Ausschnitt eines Strings IN mit Länge L ab inklusive Position P.
			Beispiel:
			'es is' =
			mid('Dies ist ein Text',5,3);
			ST:
			`OUT:=mid(IN,L,P);`

Any_String		Int	**LEN:** Länge eines Strings (ohne Terminierungszeichen) Beispiele: 17= len('Dies ist ein Text'); 4= len('Text'); **ST:** `OUT:=len(IN);`

Any_String
Any_String
Uint
Uint
		Any_String	**REPLACE:**
			Ersetzen von L Zeichen des Strings IN durch IN2 ab inkl. Position P
			Beispiel:
			'ABXE' =
			replace('ABCDE','X', 2,3);
			ST:
			`OUT:=`
			`replace(IN1, IN2, L, P);`

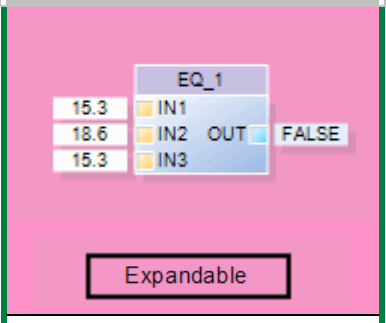
Any_String
Uint
		Any_String	**RIGHT:**
			Rechter Teil eines Strings mit Länge L
			Beispiel:
			'in Text'=
			right('Dies ist ein Text',7);
			ST:
			`OUT:= right(IN, L);`

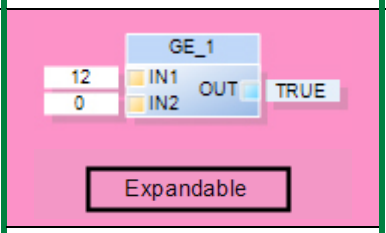
19.9 Communication

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any Bool Udint Bool String Udint Bool Bool Int Bool Bool Bool Udint Bool Bool		Any Bool Udint Bool Bool Dword String	**TCPIP_SendRecv:** Senden und Empfangen von Daten via TCP/IP. Hierbei handelt es sich um Rohdaten die über TCP/IP versendet werden, somit können fast alle Protokolle nachgebaut werden. Eine detaillierte Beschreibung finden Sie in "TCPIP_SENDRECV , Seite 106" **ST:** nicht aufrufbar innerhalb ST

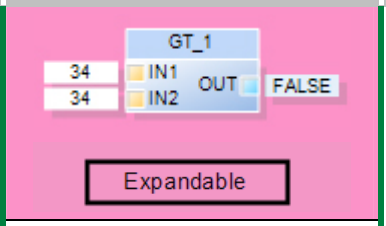
19.10 Comparison

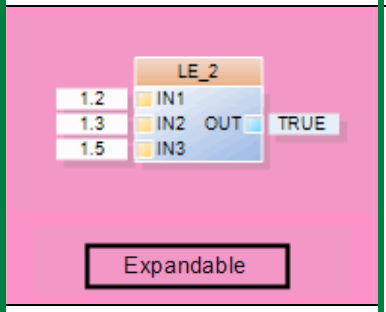
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

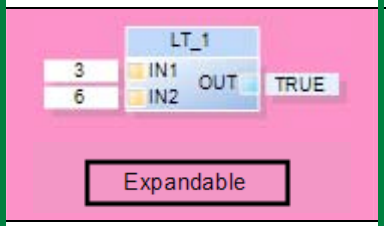
Any_
Elementary
Any_
Elementary
		Bool	**EQ:**
			Vergleich auf Gleichheit
			Ergebnis ist TRUE, wenn alle Argumente gleich sind.
			Beispiel:
			FALSE = (15.3 = 18.6 = 15.3)
			ST: Operator verwenden
			`OUT := IN1 = IN2 ;`
			Es sind nur zwei Argumente erlaubt. Realisierung durch logische Verknüpfung mehrerer Vergleiche:
			`OUT := (IN1 = IN2) AND (IN1 = IN3) ;`

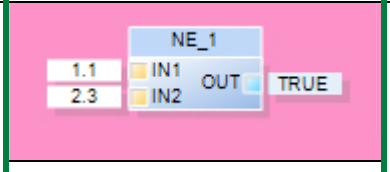
Any_
Elementary
Any_
Elementary
		Bool	**GE:**
			Vergleich auf Größer-Gleich
			Ergebnis ist TRUE, wenn IN1 größer gleich alle anderen Argumente ist.
			Beispiel:
			TRUE = 12 >= 0;
			ST: Operator verwenden:
			`OUT:= IN1 >= IN2;`
			Es sind nur zwei Argumente erlaubt. Realisierung durch logische Verknüpfung mehrerer Vergleiche:
			`OUT := (IN1 >= IN2) AND (IN1 >= IN3);`

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

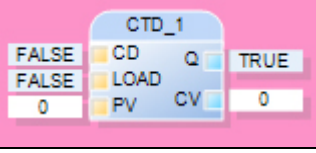
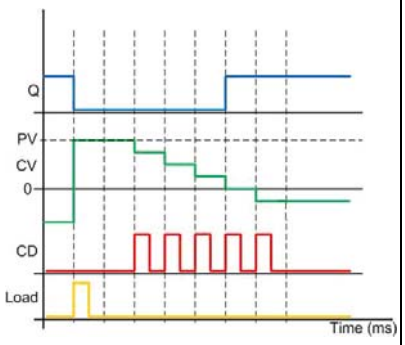
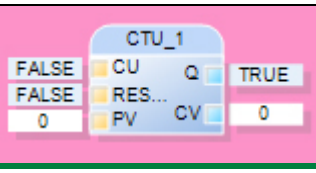
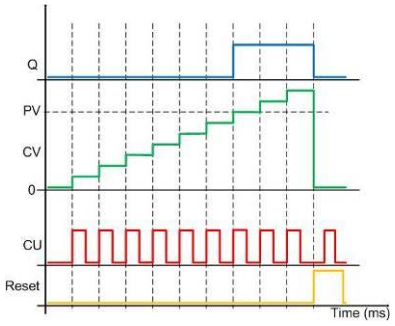
Any_ Elementary
Any_ Elementary
		Bool	**GT: Vergleich auf Größer-Als**
			Ergebnis ist TRUE, wenn IN1 größer als alle anderen Argumente ist.
			Beispiel:
			FALSE = 34 >34;
			ST: Operator verwenden:
			`OUT:= IN1 > IN2;`
			Es sind nur zwei Argumente erlaubt. Realisierung durch logische Verknüpfung mehrerer Vergleiche:
			`OUT := (IN1 > IN2) AND (IN1 > IN3);`

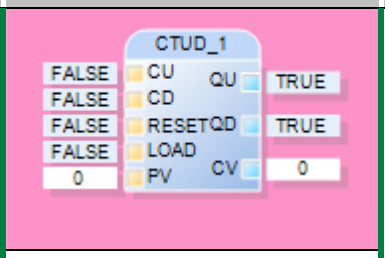
Any_ Elementary
Any_ Elementary
		Bool	**LE:**
			Vergleich auf Kleiner-Gleich
			Ergebnis ist TRUE, wenn IN1 kleiner gleich alle anderen Argumente ist.
			Beispiel:
			TRUE = (1.2 <= 1.3 <= 1.5);
			ST: Operator verwenden:
			`OUT:= IN1 <= IN2;`
			Es sind nur zwei Argumente erlaubt. Realisierung durch logische Verknüpfung mehrerer Vergleiche:
			`OUT := (IN1 <= IN2) AND (IN1 <= IN3);`

Any_ Elementary
Any_ Elementary
		Bool	**LT: Vergleich auf Kleiner-Als**
			Ergebnis ist TRUE, wenn IN1 kleiner als alle anderen Argumente ist.
			Beispiele:
			TRUE = (3 < 6);
			ST: Operator verwenden:
			`OUT:= IN1 < IN2;`
			Es sind nur zwei Argumente erlaubt. Realisierung durch logische Verknüpfung mehrerer Vergleiche:
			`OUT := (IN1 <= IN2) AND (IN1 <= IN3);`

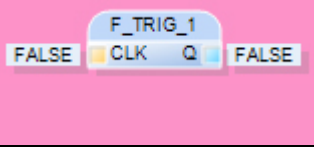
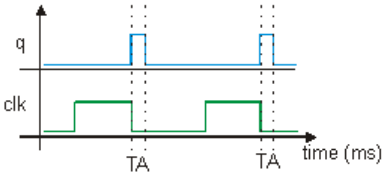
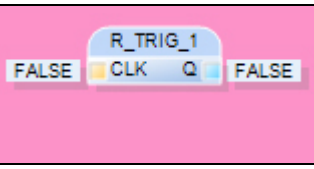
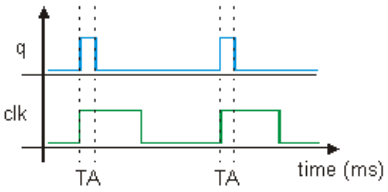
Any_ Elementary Any_ Elementary		Bool	**NE:** **Vergleich auf Ungleichheit** Ergebnis ist TRUE, wenn IN1 ungleich N1 ist. Beispiele: TRUE = ('Text 1' <> 'Text 2'); **ST:** Operator verwenden: `OUT:= IN1 <> IN2;`

19.11 Counter

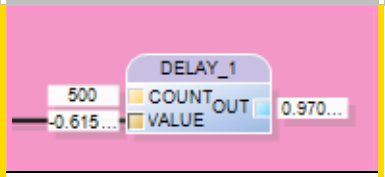
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Bool Int		Bool Int	**CTD:** Down-Counter (Abwärtszähler) Mit Setzeingang LOAD = 1 wird der Zählwert CV auf den Startwert PV gesetzt. Bei jeder positiven Flanke von CD wird der Zählwert CV um eins dekrementiert. Sobald Zählausgang CV <= 0 wird der Ausgang Q = 1 gesetzt. Der CV-Ausgang läuft bis zum Minimalwert -32768.  **ST:** nicht aufrufbar
Bool Bool Int		Bool Int	**CTU:** Up-Counter (Aufwärtszähler) Mit jeder positiven Flanke von CU wird der Zählwert CV um eins inkrementiert. Sobald Zählausgang CV >= Zählwert PV, wird der Ausgang Q = „TRUE" gesetzt. Mit „RESET" = 1 wird der Ausgang Q auf „FALSE" und der Ausgang CV auf 0 gesetzt. Der CV-Ausgang läuft maximal bis zum Höchstwert 32767.  **ST:** nicht aufrufbar

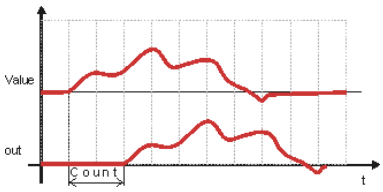
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Bool Bool Bool Int		Bool Bool Int	**CTUD:** Up-Down-Counter (Auf-/Abwärtszähler) Mit jeder positiven Flanke von CU wird der Zählwert „CV" pro Abtastzeit um eins inkrementiert. Bei Zählausgang „CV" >= Zählwert „PV" wird der Ausgang QU = 1 gesetzt (Ablaufdiagramm siehe „CTU"-FB). Mit Setzeingang „LOAD" = 1 wird der Zählwert „CV" auf den Startwert „PV" gesetzt. Mit jeder positiven Flanke von „CD" wird der Zählwert „CV" um eins dekrementiert. Sobald Zählausgang „CV" <= 0 wird der Ausgang „QD" = 1 gesetzt (Ablaufdiagramm siehe „CTD"-FB). Mit Rücksetzeingang „RESET" = 1 wird der Zählausgang auf 0 gesetzt. Wertebereich „CV": -32768 bis 32767 **ST:** nicht aufrufbar

19.12 Edge Detection

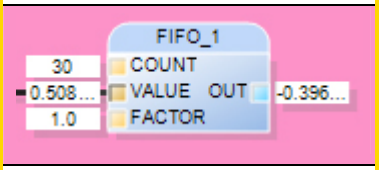
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool		Bool	**F_TRIG:** Erkennung fallende Flanken Bei einer fallenden Flanke am Eingang „CLK" wird der Ausgang Q für einen Taskzyklus auf „TRUE" gesetzt. Anlaufverhalten: Wenn der Eingang „CLK" zum Zeitpunkt des Einschaltens (Systemstart) = „FALSE" ist, dann generiert der Funktionsbaustein einen Impuls am Ausgang Q = „TRUE" für die Dauer eines Zykluses.  **ST:** nicht aufrufbar
Bool		Bool	**R_TRIG:** Erkennung steigende Flanken Bei steigender Flanke an Eingang „CLK" wird der Ausgang Q für einen Taskzyklus auf „TRUE" gesetzt. Anlaufverhalten: Wenn der Eingang „CLK" zum Zeitpunkt des Einschaltens (Systemstart) = „TRUE" ist, dann generiert der Funktionsbaustein einen Impuls am Ausgang Q = „TRUE" für die Dauer eines Zykluses.  **ST:** nicht aufrufbar

19.13 Register

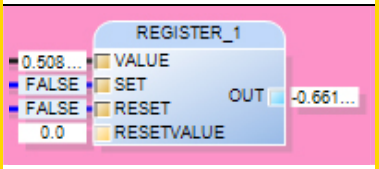
<table>
<tr><th>Eingangs-datentyp</th><th>Bausteingrafik</th><th>Ausgangs-datentyp</th><th>Erklärung, Beispiel, ST-Syntax</th></tr>
<tr>
<td>Dint
Any</td>
<td> 
</td>
<td>Any</td>
<td>DELAY: Verzögerungsfunktion

Der Ausgangswert „OUT" folgt dem Eingangswert „VALUE" mit einer Verzögerung, die über den Eingang „COUNT" in Anzahl Zyklen vorgegeben wird.
 

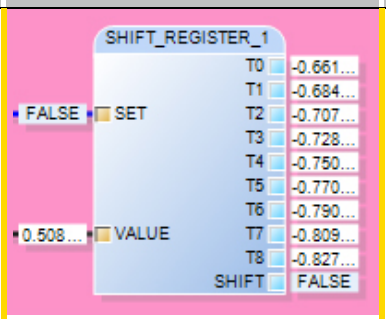
Bei Verwendung des Datentyps „ARRAY" („VALUE" und „OUT") ist der Baustein aus Gründen der Speicherkapazität begrenzt. Wenn die Anzahl der „ARRAY"-Elemente 64 überschreitet, dann wird der Wertebereich der Verzögerungstiefe von 65536 entsprechend reduziert.

ST: nicht aufrufbar</td>
</tr>
<tr>
<td>Dint
Real
Real</td>
<td> 
</td>
<td>Real</td>
<td>FIFO:
First-In_First_Out - Speicher

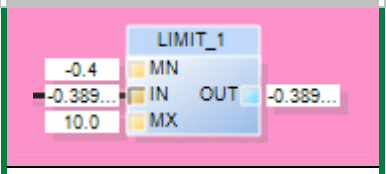
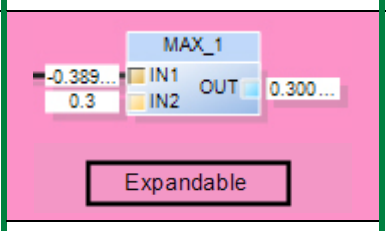
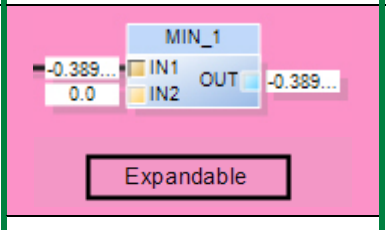
Der Ausgangswert „OUT" folgt dem Eingangswert „VALUE" mit einer Verzögerung, die über den Eingang „COUNT" in Anzahl Zyklen vorgegeben wird. Zusätzlich wird der Eingangswert mit „FACTOR" multipliziert.

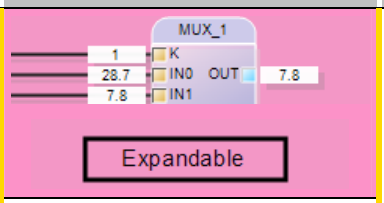
ST: nicht aufrufbar</td>
</tr>
<tr>
<td>Any_
Magnitude
Bool
Bool
Any_
Magnitude</td>
<td> 
</td>
<td>Any_Magnitude</td>
<td>REGISTER:
Registerspeicher
Der Baustein arbeitet mit Signalzustand, nicht mit Flanken.

Funktionstabelle:
<table>
<tr><th colspan="2">Eingangswerte</th><th>Ausgang</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>
ST: nicht aufrufbar</td>
</tr>
</table>

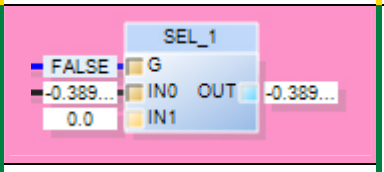
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Real		Real Real Real Real Real Real Real Real Bool	**SHIFT_REGISTER: Schieberegister** Solange der Eingang „SET" = „TRUE" ist, wird der Eingangswert „VALUE" in jedem Taskzyklus um einen Ausgang Ti weitergeschoben. Shift, wenn „SET" = „TRUE" T0: = VALUE(T_n) = aktueller Zyklus T1: = VALUE(T_{n-1}) = letzter Zyklus T8: = VALUE(T_{n-8}] = ältester Zyklus mit n = Taskzyklus **ST:** nicht aufrufbar

19.14 Selection

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Any_ Elementary Any_ Elementary Any_ Elementary		Any_ Elementary	**LIMIT:** Begrenzung Der Eingangswert IN wird auf die Grenzwerte MN (min.) und MX (max.) begrenzt. Beispiel: -0.389 = limit(-0.4, -0.389, 10.0); 15.3 = limit(8.9, 17.6, 15.3); **ST:** `OUT := limit(MN, IN, MX);`
Any_ Elementary Any_ Elementary		Any_ Elementary	**MAX:** Maximalwert Beispiele: 0.3 = max(-0.389, 0.3); 12 = max(0, 10, 12, 5); **ST:** `OUT := max(IN1, IN2, … , INn);`
Any_ Elementary Any_ Elementary		Any_ Elementary	**MIN:** Minimalwert Beispiele: -0.389 = min(-0.389, 0.3); 0 = min(0, 10, 12, 5); **ST:** `OUT := min(IN1, IN2, … , INn);`

<table>
<tr><th>Eingangs-datentyp</th><th>Bausteingrafik</th><th>Ausgangs-datentyp</th><th>Erklärung, Beispiel, ST-Syntax</th></tr>
<tr>
<td>Dint
Any
Any</td>
<td> 
</td>
<td>Any</td>
<td>MUX: Mehrfachselektor

Erweiterbarer Auswahlbaustein für beliebige Datentypen. Alle Auswahlwerte müssen vom selben Datentyp sein.
"K" = Selektor,
"IN0..IN63" Auswahlwerte,
"OUT" Ergebniswert.

ST: nicht aufrufbar</td>
</tr>
<tr>
<td>Bool
Any_
Elementary
Any_
Elementary</td>
<td> 
</td>
<td>Any_
Elementary</td>
<td>SEL: Selektor

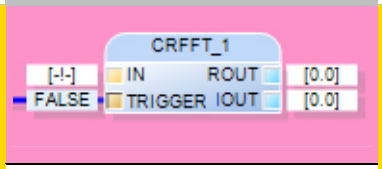
Auswahl (1 aus 2) mit binärem Signal „G"

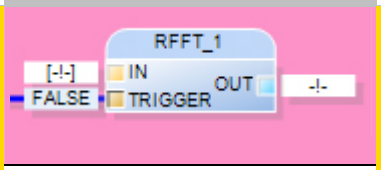
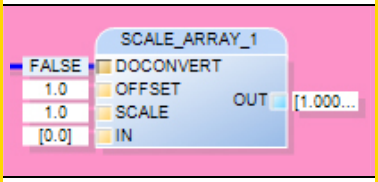
Funktionstabelle:
<table>
<tr><th>SEL</th><th>OUT</th></tr>
<tr><td>0</td><td>IN0</td></tr>
<tr><td>1</td><td>IN1</td></tr>
</table>
Beispiel:
-0.389 = sel(FALSE, -0.389, 0);

ST: unterschiedliche Aufrufe für Datentypen „REAL" und „INT", andere Datentypen sind nicht möglich.

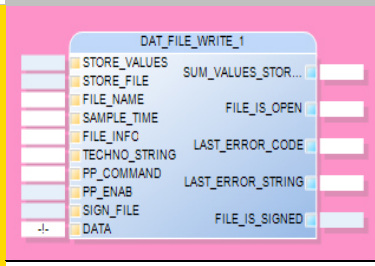
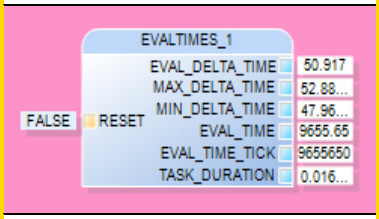
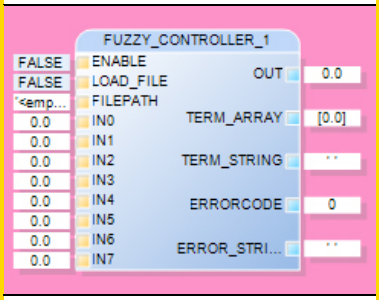
ST:
<code>OUT := sel_real(G,IN0, IN1);
OUT := sel_int(G, IN0,IN2);</code></td>
</tr>
</table>

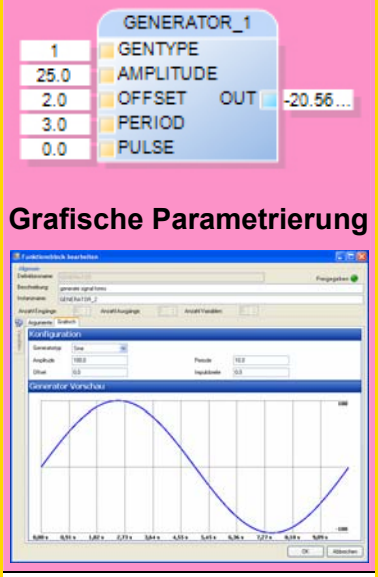
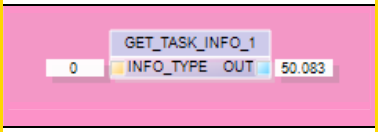
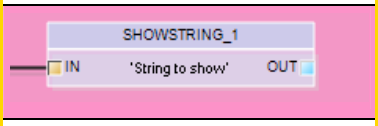
19.15 Signal Processing

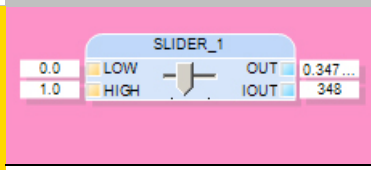
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Eindimensionales Real-Array mit der Tiefe 2^n Bool		Jeweils ein eindimensionales Real-Array mit der Tiefe 2^{n-1}	**CRFFT:** Schnelle Fourier-Transformation mit Imaginärteil Am Eingang „IN" muss ein eindimensionales Array vom Typ „REAL" und der Anzahl Elemente 2**n. Der Ausgang liefert dann zwei Arrays gleichen Typs mit der Länge 2 **(n-1). Beispiel: IN ← ARRAY [0…2047] OF REAL ROUT → ARRAY [0…1023] OF REAL IOUT → ARRAY [0…1023] OF REAL Die FFT-Berechnung wird nur aktiviert, wenn der Eingang „TRIGGER" = „TRUE" ist. Nur dann benötigt der Baustein Rechenzeit! Am Ausgang „ROUT" liefert dieser Funktionsbaustein den reellen Teil und am „IOUT" Ausgang den imaginären Teil einer FFT-Berechnung. Berechnungsmodus: Absolute Amplitude, alle Werte im Array sind gleich gewichtet (Rechteckfenster). **ST:** nicht aufrufbar

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Eindimensionales Real-Array mit der Tiefe 2^n Bool		Jeweils ein eindimensionales Real-Array mit der Tiefe 2^{n-1}	**RFFT:** Schnelle Fourier-Transformation) Am Eingang „IN" muss ein eindimensionales Array vom Typ REAL und der Anzahl Elemente 2**n. Der Ausgang liefert dann ein Array gleichen Typs mit der Länge 2 **(n-1). Beispiel: IN ← ARRAY [0 … 2047] OF REAL OUT → ARRAY [0 … 1023] OF REAL Die FFT-Berechnung wird nur aktiviert, wenn der Eingang „TRIGGER" = „TRUE" ist. Auch nur dann benötigt der Baustein Rechenzeit! Am Ausgang liefert dieser Funktionsbaustein das Ergebnis einer FFT nach dem Berechnungsmodus: Absolute Amplitude, alle Werte im Array sind gleich gewichtet (Rechteckfenster). **ST:** nicht aufrufbar
Bool Real Real Any_ Derived		Any_ Derived	**SCALE_ARRAY:** Solange der Eingang „DOCONVERT" = „TRUE" ist, wird jedes Element im Eingangsarray „IN" mit dem Faktor „SCALE" multipliziert und mit dem „OFFSET"-Eingang addiert. Am Ausgang „OUT" steht dann das skalierte Array zur Verfügung. **ST:** nicht aufrufbar

19.16 Specials

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Bool String Lreal Lreal Lreal Lreal Lreal Lreal Lreal Lreal		Dint Usint Dword String Bool	**DAT_FILE_WRITE:** Mit diesem Baustein lassen sich Signale zur späteren Analyse mit dem ibaAnalyzer direkt in ibaLogic aufzeichnen. Eine detaillierte Beschreibung finden Sie in "DAT_FILE_WRITE (DFW-Funktionsbaustein) , Seite 97". **ST:** nicht aufrufbar
Bool		Real Real Real Real Udint Real	**EVALTIMES:** **Ausgabe der Berechnungsdaten** EVAL_DELTA_TIME = aktuelle Zykluszeit des Tasks (in ms) MAX_DELTA_TIME = max. Zykluszeit des Tasks seit dem letzten Start (in ms) MIN_DELTA_TIME = minimale Zykluszeit EVAL_TIME = vergangene Zeit seit dem letzten Start (in ms) EVAL_TIME_TICK = vergangene Zeit seit dem letzten Start in µs TASK_DURATION = Berechnungszeit des aktuellen Tasks. **ST:** nicht aufrufbar
Bool Bool String Lreal Lreal Lreal Lreal Lreal Lreal Lreal Lreal		Lreal Array [0..8] of Lreal String Int String	**FUZZY_CONTROLLER:** Regelungsbaustein mit Fuzzy-Logic. Eine Kurz-Beschreibung finden Sie in "FUZZY_CONTROLLER , Seite 120". **ST:** nicht aufrufbar

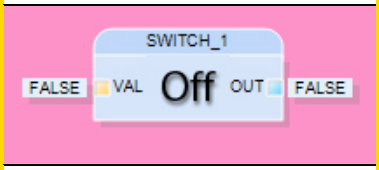
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Int Real Real Real Real		Real	**GENERATOR:** Funktionsgenerator für Sinus, Rechteck und Dreiecksignal. „GENTYPE" = 1 für Sinus, 2 für Rechteck, 3 für Dreiecksignal „AMPLITUDE" = Amplitudenwert; es gibt nur einen Wert, der symmetrisch zur X-Achse berechnet wird, d. h. er gilt sowohl positiv als auch negativ. „OFFSET" = Angabe des Offsets (Lage der Nulllinie); ist ein Signalverlauf gewünscht, der nicht negativ wird, dann muss der Offset mindestens so groß wie die Amplitude gewählt werden. „PERIODE" = Angabe der Periodendauer in Sekunden „PULSE" (Pulsbreite) = Angabe der Zeit für den ersten Puls in Sekunden, wird bei Sinus nicht benutzt. Der Wert darf nicht größer sein als die Periodendauer. Für ein symmetrisches Signal gilt Pulse = Periode / 2 Als Besonderheit hat dieser Baustein auch die Möglichkeit, die Signale grafisch zu parametrieren. Diese Oberfläche erhält man durch Doppelklick auf den Baustein. Mit der Maus lassen sich dann in der Grafik die Werte einstellen. **ST:** nicht aufrufbar
Int		Real	**GET_TASK_INFO:** Funktion zum Auslesen von Taskinformationen entsprechend des Parameters "INFO_TYPE". "INFO_TYPE" = 0: EvalDeltaTime, 1: EvalTime, 2: LastTaskDuration **ST:** nicht aufrufbar
String		String	**SHOWSTRING:** Anzeigeelement zum Darstellen von Strings. **ST:** nicht aufrufbar

<table>
<tr><th>Eingangs-datentyp</th><th>Bausteingrafik</th><th>Ausgangs-datentyp</th><th>Erklärung, Beispiel, ST-Syntax</th></tr>
<tr>
<td>Real
Real</td>
<td> 
</td>
<td>Real
Dint</td>
<td>SLIDER: Schieberegler

Abhängig von der Stellung des Schiebers liefert dieser Funktionsbaustein an seinem Ausgang „OUT" einen Wert, der in den Grenzen der Eingangsvorgaben (Min- und Max-Wert) liegt. Die Eingänge sind standardmäßig mit 0 bzw. 1 vorbelegt, können aber beliebig verändert werden. (Doppelklick auf Baustein und Default-Werte anpassen)
Der Ausgang „IOUT" liefert den relativen Stellwert des Schiebers in tausendstel Schritten (0 ... 1000).

Die Ausgänge werden erst gesetzt, wenn der Slider mit der Maustaste losgelassen wird.

Ist der Sliderzeiger markiert, kann er auch über die Cursortasten → und ← bewegt werden.

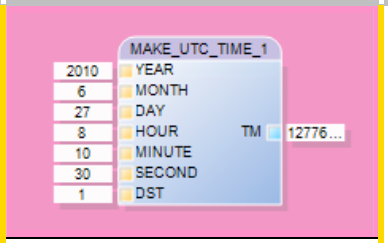
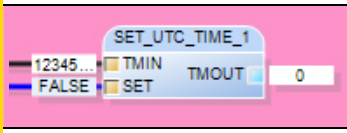
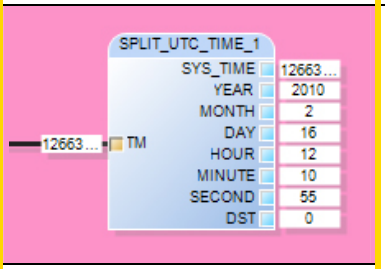
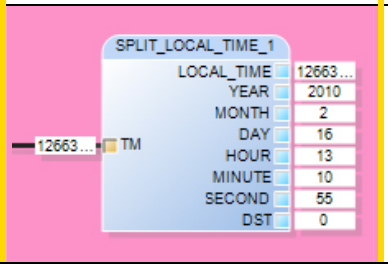
ST: nicht aufrufbar</td>
</tr>
<tr>
<td>Bool</td>
<td> 
</td>
<td>Bool</td>
<td>SWITCH: Schalter

Für die Schalterfunktion klickt man mit der linken Maustaste auf das „OFF"-Symbol um ein- bzw. auszuschalten („toggle").

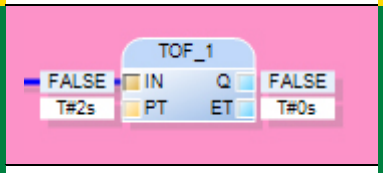
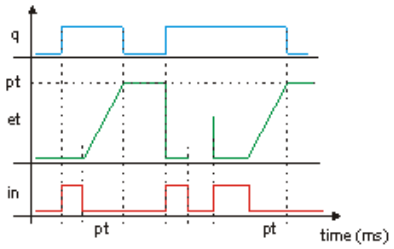
In Verbindung mit dem Eingang, hat man eine „OR"-Funktion zwischen Schalter-stellung und Eingang.

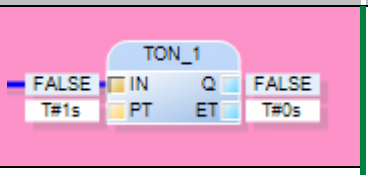
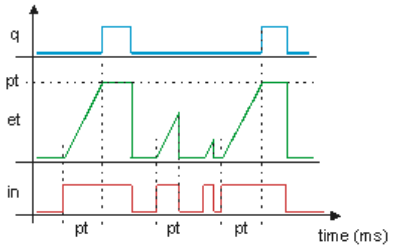
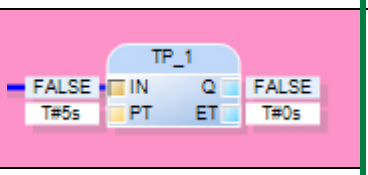
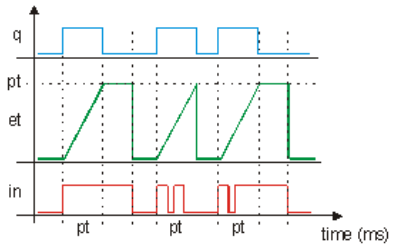
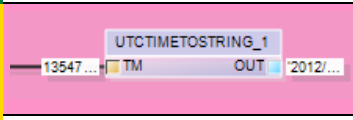
Wahrheitstabelle:
<table>
<tr><th>SWITCH</th><th>VAL</th><th>OUT</th></tr>
<tr><td>ON</td><td>FALSE</td><td>TRUE</td></tr>
<tr><td>ON</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td>OFF</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>OFF</td><td>TRUE</td><td>TRUE</td></tr>
</table>
ST: nicht aufrufbar</td>
</tr>
</table>

19.17 Timer

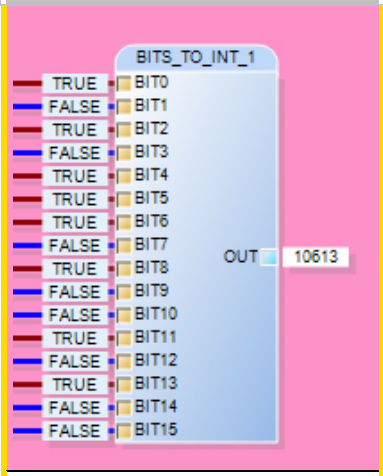
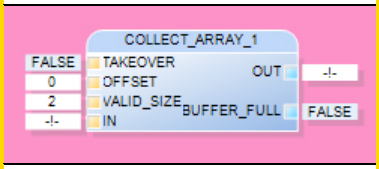
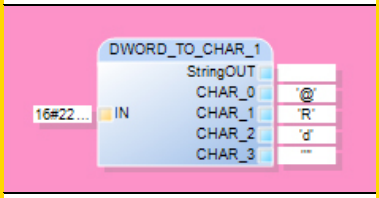
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Dint Dint Dint Dint Dint Dint Dint		Udint	**MAKE_UTC_TIME:** **Kodieren der UTC-Zeit**[8] Der Funktionsbaustein erzeugt die UTC-Zeit am Ausgang „TM" aus den Eingangsvariablen „YEAR", „MONTH", „DAY", „HOUR", „MINUTE" und „SECOND". Die lokale Zeitzone[9] wird **nicht** berücksicht. Die Sommerzeit\ -verschiebung können Sie am Eingang „DST" angeben. Beispiel: 27.06.2010/08:10:30 in der Zeitzone GMT+01 → TM = 1277626230 **ST:** nicht aufrufbar
Udint Bool		Udint	**SET_UTC_TIME:** Setzen der UTC-Zeit Der Funktionsbaustein setzt die „UTC"-Systemzeit des ibaLogic Zielsystems (Windows-PC oder PADU-S-IT) auf den Wert, der am Eingang „TMIN" anliegt, wenn Eingang „SET" = „TRUE" ist. Die lokale Zeitzone und die Sommerzeitverschiebung wird **nicht** berücksicht. **ST:** nicht aufrufbar
Udint		Udint Dint Dint Dint Dint Dint Dint Dint	**SPLIT_UTC_TIME:** **Dekodierung der** UTC-Zeit **in die GMT.** Der Funktionsbaustein wandelt aus der UTC-Zeit am Eingang „TM" die Ausgangsvariablen „YEAR", „MONTH", „DAY", „HOUR", „MINUTE" und „SECOND". Dabei wird die lokale Zeitzone nicht berücksichtigt. Die Sommerzeitverschiebung wird am Ausgang „DST" angezeigt. **ST:** nicht aufrufbar
Udint		Udint Dint Dint Dint Dint Dint Dint	**SPLIT_LOCAL_TIME:** **Dekodierung der** UTC-Zeit **in die lokale Zeit.** Der Funktionsbaustein wandelt aus der UTC-Zeit am Eingang TM die Ausgangsvariablen „YEAR", „MONTH", „DAY", „HOUR", „MINUTE" und „SECOND".

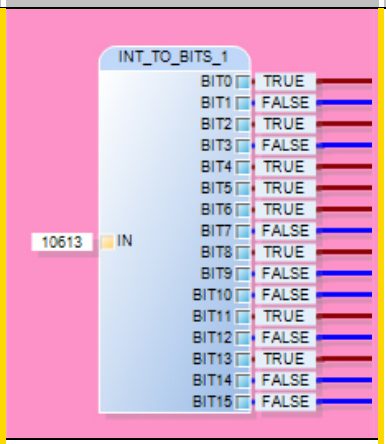
[8] Die UTC-Zeit (Universal Time Coordinated, koordinierte Weltzeit) enthält die Zeit in Sekunden seit 01.01.1970,00:00 Uhr, bezogen auf GMT+00.
[9] Die Information über Zeitzone und Sommerzeit wird aus den Betriebssystemeinstellungen unter WindowsXP bzw. WindowsCE übernommen.

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
		Dint	Dabei wird die lokale Zeitzone berücksichtigt. Die Sommerzeitverschiebung wird am Ausgang „DST" angezeigt. **ST:** nicht aufrufbar
Bool Time		Bool Time	**TOF:** Off-Delay (Ausschaltverzögerung) Mit „TRUE" am Eingang „IN" wird der Ausgang „Q" unverzögert auf „TRUE" gesetzt. Die fallende Flanke am Eingang „IN" startet die Verzögerungszeit PT. Nach Ablauf der Verzögerungszeit wird der Ausgang „Q" auf „FALSE" gesetzt. Ausgang „Q" bleibt unverändert, wenn die Ausschaltzeit von „IN" kürzer als die Verzögerungszeit ist. Ausgang „ET" zeigt die bereits abgelaufene Zeit an.  **ST:** nicht aufrufbar

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Time		Bool Time	**TON:** On-Delay (Einschaltverzögerung) Die ansteigende Flanke am Eingang „IN" startet die Verzögerungszeit PT. Nach Ablauf der Verzögerungszeit wird der Ausgang „Q" auf „TRUE" gesetzt. „FALSE" am Eingang „IN" wird sofort auf Ausgang „Q" durchgeschaltet. Der Ausgang „Q" wird nicht gesetzt, wenn die Einschaltzeit von „IN" kürzer als die Verzögerungszeit ist. Ausgang „ET" zeigt die bereits abgelaufene Zeit an.  **ST:** nicht aufrufbar
Bool Time		Bool Time	**TP:** Pulse Timer (Impulsverlängerung) Die ansteigende Flanke am Eingang „IN" setzt den Ausgang „Q" für die Impulszeit PT auf „TRUE". Der Ausgang „Q" kann während der Impulszeit nicht zurückgesetzt werden. Ausgang „ET" zeigt die bereits abgelaufene Zeit an.  **ST:** nicht aufrufbar
Udint		String	**UTCTIMETOSTRING:** Konvertiert UTC Zeit in einen formatierten String. **ST:** `OUT := UTCTIMETOSTRING(IN);`

19.18 Type Conversion

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool		Int	**BITS_TO_INT:** Konvertiert 16 Bits in einen Integer-Wert Beispiel: 10613 = 2#0010_1001_0111_0101, (Bit15 …….…….. Bit0) **ST:** nicht aufrufbar
Bool Uint Uint Any_ Derived		Any_ Derived Bool	**COLLECT_ARRAY** Mit diesem Baustein können Abschnitte aus einem Array in ein anderes übertragen werden. „TAKEOVER": Solange dieser Eingang „TRUE" ist, werden die Daten in das Ausgangsarray übertragen. „OFFSET": Gibt an, ab welchem Element im Eingangsarray die Daten kopiert werden. „VALID_SIZE": Anzahl der Elemente die kopiert werden. „IN": Eingangsarray mit beliebiger Größe. „OUT": Ausgangsarray „BUFFER_FULL": Ausgangsarray ist voll, Daten müssen ausgelesen werden. **ST:** nicht aufrufbar
Dword		String String String String String	**DWORD_TO_CHAR** Konvertierung eines „DWORD" in vier einzelne Zeichen von Typ „STRING" 16#22645240 = ' @', 'R', 'd', "" **ST:** nicht aufrufbar innerhalb ST

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Int		Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool	**INT_TO_BITS:** Konvertiert einen Integerwert in 16 Bits (Umkehrfunktion zu BITS_TO_INT) **ST:** nicht aufrufbar

19.18.1 Limiting Converter

Diese Konvertierungsbausteine nehmen eine Sonderstellung ein, da sie vor der Typ-Konvertierung den Wertebereich des Eingangstyps auf den Wertebereich des Ausgangstyps begrenzen. Den Unterschied zu einem Standard-Konverter sehen Sie im folgenden Beispiel.

Begrenzungskonvertierer limit:dint_to int(57700) liefert Ergebnis: 32767
:

(zuerst wird der Ausgangswert auf den Integer Wertebereich (-32768 … 32767) beschränkt, dann die Typumwandlung durchgeführt).

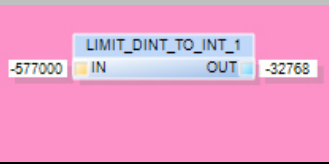
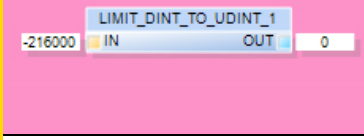
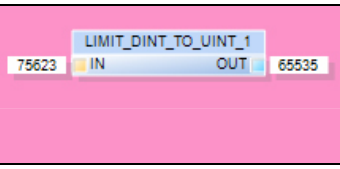
Standardkonvertierer: dint_to_int(577000) liefert Ergebnis: -12824

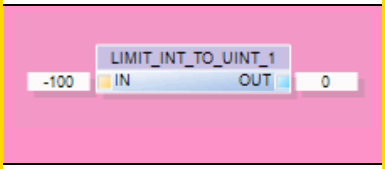
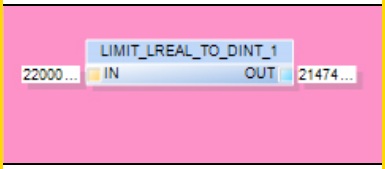
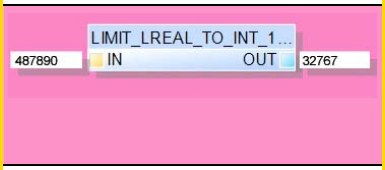
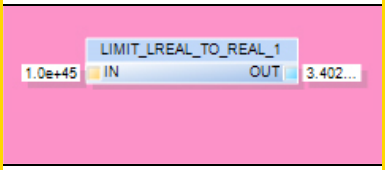
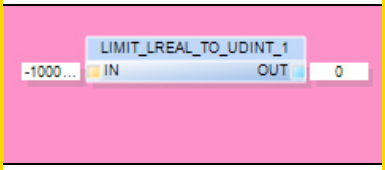
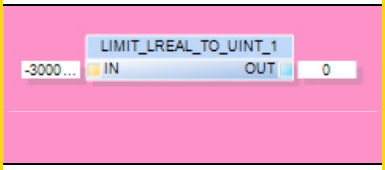
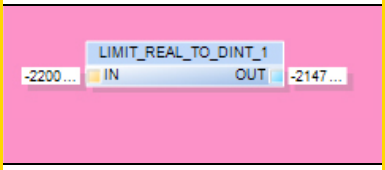
(es werden die 16 niederwertigen Bits herangezogen und diese werden konvertiert, dabei wird natürlich das höchstwertige Bit (Bit 15) als Vorzeichenbit interpretiert.)

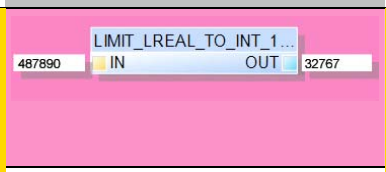
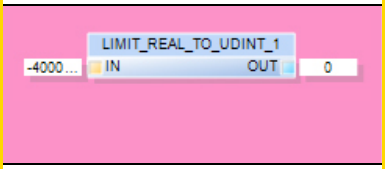
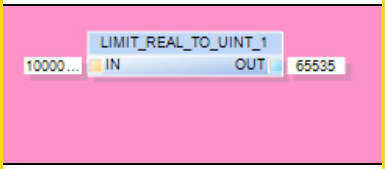
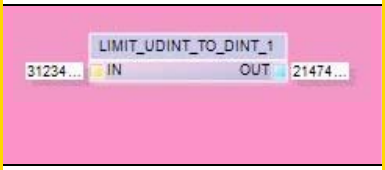
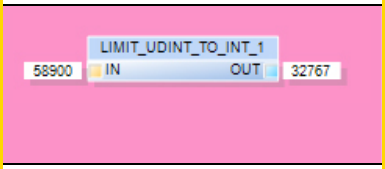
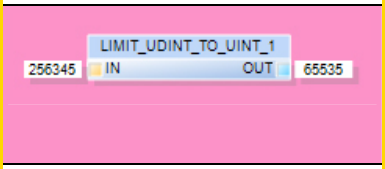
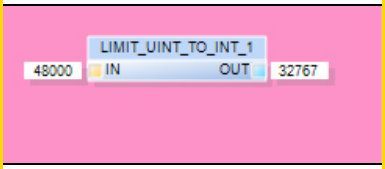
Wir empfehlen immer Begrenzungskonvertierer einzusetzen, wenn der Wertebereich des Zieltyps kleiner ist als der Wertebereich des Quelltyps. Dies betrifft folgende Konvertierungen:

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 sind die Begrenzungskonvertierer verfügbar durch die Funktionen: „limit_*quelltyp*_to_*zieltyp*"

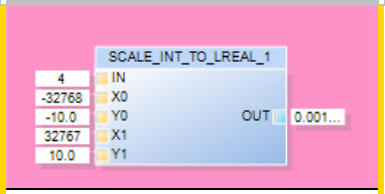
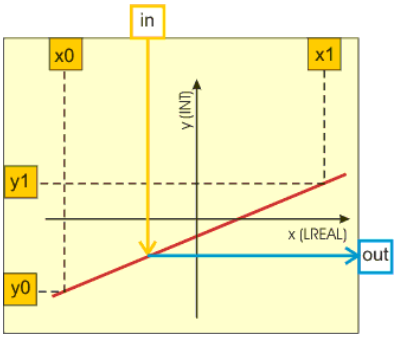
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Dint		Int	**LIMIT_DINT_TO_INT:** Beispiel: -577000 => -32768; **ST:** `OUT:= limit_dint_to_int(IN);`
Dint		Udint	**LIMIT_DINT_TO_UDINT:** Beispiel: -216000 => 0; **ST:** `OUT:= limit_dint_to_udint(IN);`
Dint		Uint	**LIMIT_DINT_TO_UINT:** Beispiel: 75623 => 65535

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
			ST: `OUT:= limit_dint_to_uint(IN);`
int		Uint	**LIMIT_INT_TO_UINT** Beispiel: -100 => 0 **ST:** `OUT:= limit_int_to_uint(IN);`
Lreal		Dint	**LIMIT_LREAL_TO_DINT:** Beispiel: 2.2 E+09 => 2147483648; **ST:** `OUT:= limit_lreal_to_dint(IN);`
Lreal		Int	**LIMIT_LREAL_TO_INT:** Beispiel: 248758 => 32767 **ST:** `OUT:= limit_lreal_to_int(IN);`
Lreal		Real	**LIMIT_LREAL_TO_REAL** Beispiel: 1E+45 => 3.402823466 E+38 **ST:** `OUT:= limit_lreal_to_real(IN);`
Lreal		Udint	**LIMIT_LREAL_TO_UDINT:** Beispiel: -1 E+12 => 0; **ST:** `OUT:= limit_lreal_to_udint (IN);`
Lreal		Uint	**LIMIT_LREAL_TO_UINT:** Beispiel: -3 E+12 => 0; **ST:** `OUT:= limit_lreal_to_uint(IN);`
Real		Dint	**LIMIT_REAL_TO_DINT:** Beispiel: -2.2 E+09 => -2147483648; **ST:** `OUT:= limit_real_to_dint(IN);`

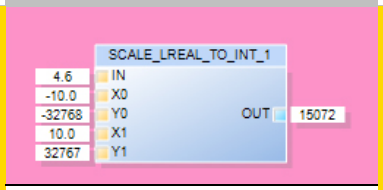
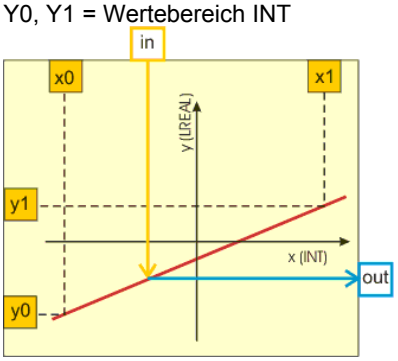
Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
Real		Int	**LIMIT_REAL_TO_INT:** Beispiel: 248758 => 32767 **ST:** `OUT:= limit_real_to_int(IN);`
Real		Udint	**LIMIT_REAL_TO_UDINT:** Beispiel: -4000.0 => 0 **ST:** `OUT:= limit_real_to_udint(IN);`
Real		Uint	**LIMIT_REAL_TO_UINT:** Beispiel: 1*E+12 => 65535; **ST:** `OUT:= limit_real_to_uint(IN);`
Udint		Dint	**LIMIT_UDINT_TO_DINT:** Beispiel: 3123456789 => 2147483647 **ST:** `OUT:= limit_udint_to_dint(IN);`
Udint		Int	**LIMIT_UDINT_TO_INT:** Beispiel: 558900 => 32767 **ST:** `OUT:= limit_udint_to_int(IN);`
Udint		Uint	**LIMIT_UDINT_TO_UINT:** Beispiel: 256345 => 65535 **ST:** `OUT:= limit_udint_to_uint(IN);`
Uint		Int	**LIMIT_UINT_TO_INT:** Beispiel: 48000 => 32767 **ST:** `OUT:= limit_uint_to_int(IN);`

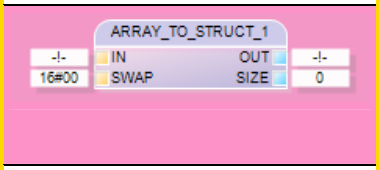
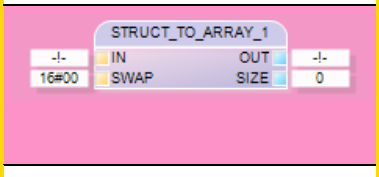
19.18.2 Scaling Converter

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

Int
Int
Lreal
Int
Lreal
		Lreal	**SCALE_INT_TO_LREAL:**
			Dieser Baustein wandelt einen „INTEGER"-Wert in einen „LREAL"-Wert und skaliert linear.
			
			Anwendung:
			Umrechnung Analogeingang (Integerwert -32768 … 32767) in eine physikalische Größe
			z. B. +/- 10 Volt.
			
			IN:
			Eingangswert (Analogeingang)
			X0, X1: Wertebereich Eingangswert (Int)
			Y0, Y1: Wertebereich Zielgröße (Lreal)
			
			Beispiel:
			4 → 0.0013733119 V
			X0, X1 = -32768 / +32767
			Y0, Y1 = -10.0 / + 10.0
			IN = 4
			OUT = 0.0013733119
			 
			Implementierung (Typkonvertierungen wurden der Übersichtlichkeit halber weggelassen):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 - y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out = aa*in + bb;`
			`end_if;`
			
			ST: nicht aufrufbar
			
			Wichtiger Hinweis:
			Da der Integerwertebereich prinzipiell nicht symmetrisch ist, führt eine 0 am Eingang nicht zu einer 0 am Ausgang. Da dies immer zu Fehlinterpretationen führt, empfiehlt iba AG, einen symmetrischen Eingangs-Wertebereich (-32767/+ 32767) anzugeben.

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax

Lreal
Lreal
Int
Lreal
Int
		Int	**SCALE_LREAL_TO_INT:**
			Dieser Baustein wandelt einen „LREAL"-Wert in einen „INTEGER"-Wert und skaliert linear.
			Anwendung: Umrechnung physikalische Größe (z. B. +/- 10 Volt) in einen Analogausgang (Integer-Wert -32768 … 32767)
			Beispiel:
			4.6 V → 15072
			X0, X1 = Bereich physikalisch (-/+ 10 V)
			Y0, Y1 = Wertebereich INT
			 
			Implementierung (Typkonvertierungen wurden der Übersichtlichkeit halber weggelassen):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 – y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out := aa*in + bb;`
			`end_if;`
			ST: nicht aufrufbar

Any_Array Byte		Any_Struct Int	**ARRAY_TO_STRUCT:** Erstellt aus einem beliebigen Array eine Struktur. SIZE beinhaltet die effektiv genutzte Datengröße. Mit SWAP lässt sich der Byteswap einstellen: 16#00 Aus, 16#01 nach Datentyp (AB CDEF → BA FEDC), 16#02 2 Bytes (ABCD → BADC), 16#04 4 Bytes (ABCD → DCBA). **ST:** nicht aufrufbar
Any_Struct Byte		Any_Array Int	**STRUCT_TO_ARRAY:** Erstellt aus einer beliebigen Struktur einen Array. (Parameter siehe ARRAY_TO_STRUCT) **ST:** nicht aufrufbar

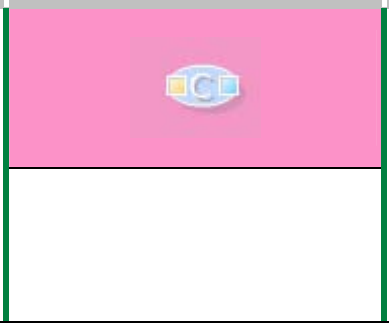
19.18.3 Standard Converter

Alle Standardkonviertierungsbausteine sind in einem Baustein zusammengefasst. Sobald Sie diesen Baustein per „Drag & Drop" in den Programmierbereich ziehen, erscheint ein Auswahldialog mit dem sich der Eingangs- und Ausgangsdatentyp festlegen lässt.

Konvertierungsregeln:

- ❑ Bei Zieltyp „BOOL" ist das Ergebnis „FALSE", wenn der Eingang den Wert 0, 0.0, 16#0 oder T#0ms hat, andernfalls wird „TRUE" ausgegeben.
- ❑ Bei Real-Datentypen nach Integer-Datentypen werden die Werte numerisch umgerechnet und nach den arithmetischen Regeln gerundet. Eine Begrenzung wird nicht durchgeführt.
  Beispiel: 82600.0(REAL) → 82600(DINT) → 17064(INT)
- ❑ Konvertierung von „INTEGER"- und „WORD"-Datentypen erfolgt durch Typkonvertierung ohne Änderung des Bit-Musters. Evtl. notwendige Begrenzung wird nicht durchgeführt, die höherwertigen Bits werden abgeschnitten.
- ❑ Bei Real-Datentypen nach „WORD"-Datentypen wird zuerst eine numerische Umrechnung nach „DINT" durchgeführt, danach erfolgt nur eine Datentypkonvertierung ohne Änderung des Bitmusters.

Wenn Sie zwei Konnektoren unterschiedlichen Datentyps miteinander verbinden, wird automatisch ein passender Konvertierungsbaustein angelegt, vorausgesetzt eine Konvertierung ist möglich. Weitere Informationen siehe „Konvertierer , Seite 164".

Eingangs-datentyp	Bausteingrafik	Ausgangs-datentyp	Erklärung, Beispiel, ST-Syntax
			**TYPE_TO_TYPE:** Sobald dieser Baustein per Drag & Drop in den Programmierbereich gezogen wird, erscheint ein Auswahldialog mit dem sich der Eingangs- und Ausgangsdatentyp festlegen lässt. **ST:** Funktionsname wird gebildet aus *Quelltyp*_to_*Zieltyp*, z. B. `OUT := real_to_int(IN);`

Hinweis

Wandlung nach TIME:

Integer-Werte werden als Millisekunden-Werte interpretiert.

Real-Werte werden als Sekunden-Werte interpretiert.

Beispiel:

Ein Integer-Wert von 4711 ergibt daher 4,177 Sekunden.

Ein Real-Wert von 4711.0 ergibt 4711 Sekunden
(dies sind 1 h, 18 min und 31 sec).

20 Fehlercodes

20.1 Fehlercodes DAT_FILE_WRITE

Errorcode	Bedeutung
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**

20.2 Fehlercodes TCPIP_SENDRECV

Fehlercode	Bedeutung	Lösungsvorschlag
HEX: 16#0000271d DEZ: 10013	Permission denied - Access to socket forbidden by acces permissions.	Melden Sie sich mit einem Benutzer an, der Administratorrechte hat.
16#00002740 10048	Address already in use - Only one usage of each socket address is permitted.	Die angegebene Adresse / Port wird bereits verwendet.
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	Connection reset by peer - An existing connection was	

Fehlercode	Bedeutung	Lösungsvorschlag
10054	forcibly closed by the remote host.	
16#00002737 10039	Destination address required - A required address was omitted from an operation on a socket.	
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.	Reduzieren Sie die Länge der zu sendenden Bytes.
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 implementation for it exists.	
16#00002753	Too many processes - A Windows Sockets implementation may have a limit on the number of	

Fehlercode	Bedeutung	Lösungsvorschlag
10067	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.	
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.	

Fehlercode	Bedeutung	Lösungsvorschlag
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.	
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.	

21 Besonderheiten bei TCP/IP

21.1 Anzahl möglicher TCP/IP-Verbindungen

Hinweis

Die Anzahl möglicher TCP/IP-Verbindungen in ibaLogic hängt von der Systemeinstellung „TcpNumConnections" ab.

Dazu ein Auszug von Microsoft:

"**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 - Maximale Anzahl (http://www.windowspage.de/tipps/021202.html) gleichzeitig geöffneter Verbindungen

21.2 Delayed Acknowledge-Problem

Problem

Messungen von Automatisierungsgeräten mittels TCP/IP funktionieren nicht mit Zykluszeiten < 200 ms.

Fehlerbild: Sequenzfehler, Unvollständige Telegramme, Unterschiedliche Empfangslängen.

Ursache

Es gibt im TCP/IP-Protokoll verschiedene Varianten, wie das Acknowledge behandelt wird:

1. Der Standard WinSocket arbeitet nach RFC1122 mit dem „delayed acknowledge" Mechanismus. Dieser sagt aus, dass das Acknowledge verzögert wird, bis weitere Telegramme eintreffen, um diese dann gemeinsam zu quittieren. Falls keine weiteren Telegramme ankommen, wird spätestens nach 200 ms (abhängig vom Socket) das ACK-Telegramm gesendet.
   Nach dem TCP/IP-Standard ist das möglich, denn mit der Datenflusssteuerung durch „Sliding Window" (Parameter Win = nnnn) gibt der Empfänger an, wie viele Bytes er empfangen kann, ohne eine Quittung zu senden.
2. Manche Controller akzeptieren dieses Verhalten nicht, sondern erwarten nach jedem Datentelegramm eine Quittung. Falls dieses nicht innerhalb einer bestimmten Zeit (200 ms) ankommt, wiederholt dieser das Telegramm und packt evtl. neu zu sendende Daten hinzu. Dies führt bei dem Empfänger zu einem Fehler, da das alte korrekt empfangen wurde.

Abhilfe

Das „delayed acknowledge" muss in Windows mittels des Parametereintrags in der Windows Registry abgeschalten werden:

„TcpAckFrequency" REG_DWORD = 1;

Der Parameter ist standardmäßig nicht vorhanden und muss unter diesem Pfad eingetragen werden:

"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{InterfaceGUID}"

Hinweis

Sie müssen das richtige Interface auswählen. Welches richtig ist, können Sie z. B. aus den aktuell eingestellten IP-Adressen ersehen.

Siehe auch folgende MS Seite:
Neuer Registrierungseintrag (http://support.microsoft.com/kb/328890) zum Überprüfen des TCP-Bestätigungsverhaltens in Windows XP und Windows Server 2003

Windows XP:

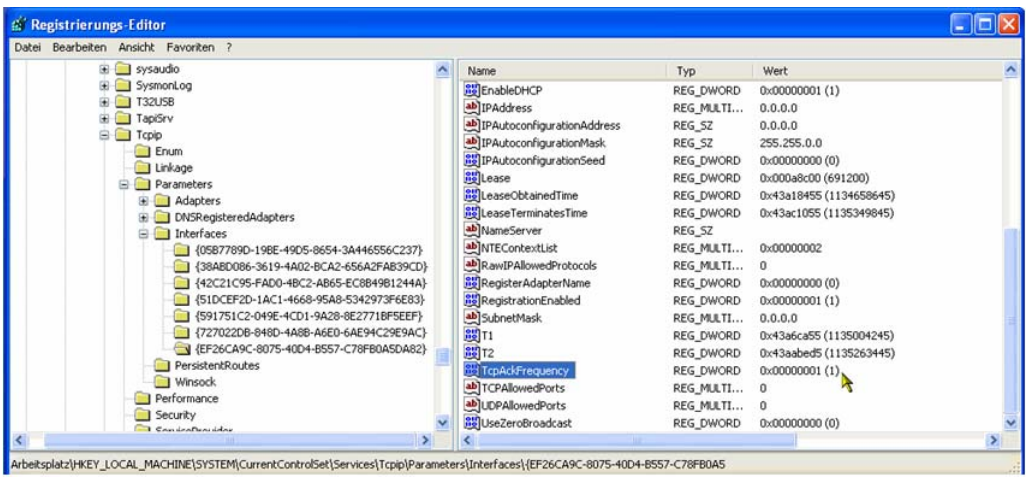


Abbildung 147: Windows XP-Registry

22 Tastenkombinationen

22.1 Client

Tastaturkombination	Erklärung
<Strg> + <P>	Drucken
<Strg> + <Z>	Rückgängig
<Strg> + <Y>	Wiederholen
<Strg> + <C>	Kopieren
<Strg> + <V>	Einfügen
<Strg> + <A>	Alles Auswählen
<Entf>	Entfernen
<F5>	Berechnung Starten
<Umschalt> + <F5>	Berechnung Stoppen
<Strg> + <Umschalt> + <F>	Hinzufügen – neuer Funktionsblock
<Strg> + <Umschalt> + <M>	Hinzufügen – neuer Makroblock
<Strg> + <Umschalt> + <I>	Hinzufügen – neuer Intra-Page-Konnektor
<Strg> + <Umschalt> + <T>	Hinzufügen – neuer Off-Task-Konnektor
<Strg> + <Umschalt> + <C>	Hinzufügen – neuer Kommentar
<Strg> + <Umschalt> + <S>	Off-Task-Konnektoren anzeigen
<F1>	Hilfe aufrufen

22.2 Mausfunktionen im Programmierfeld

Tastaturkombination	Erklärung
linker Mausklick + <Shift>	Mehrfach markieren
linker Mausklick + <Strg>	Markierung umschalten
Scroll-Rad + <Shift>	Sichtbaren Ausschnitt nach links/rechts verschieben.
Scroll-Rad + <Strg>	Einzoomen/Auszoomen
Scroll-Rad + <Alt>	Sichtbaren Ausschnitt nach oben/unten verfahren.
<Strg> + Konnektorein- oder -ausgang	IPC-Erstellung
Drag & Drop + <Alt>	Signale können per Drag & Drop bei gedrückter <Alt>-Taste von einem Konnektor in das ibaPDA Express-Fenster gezogen werden.

22.3 ibaPDA Express

Tastaturkombination	Erklärung
<F6> (Umschaltung)	Startet die laufende Anzeige beim aktuellen Zeitpunkt. Aktiv, wenn „Pause Vorschub" gedrückt.
<F6> (Umschaltung)	Anhalten der laufenden Anzeige. Nach Betätigung erscheint ein Lineal im Graphen, das mit der Maus bewegt und mit dem die Kurven vermessen werden können. Anzeige der Signalwerte in der Legende. Die X-Achse kann mit der Maus verschoben werden. Somit können Werte aus der Vergangenheit betrachtet werden. Aktiv, wenn Anzeige läuft.
<F5>	Autoskalieren
<F3>	Nur aktiv in gezoomter Darstellung. Auf letzte Zoomstufe zurückschalten (verkleinern).
<F4>	Nur aktiv in gezoomter Darstellung. Auf ursprüngliche (automatische) Darstellung zurückschalten.
<F10>	Zurück aus dem Vollbildmodus.

23 Zeichentabellen

ibalogic verwendet eine vereinfachte Hex-Codierung. Dadurch lassen sich die ersten 256 Unicode-Zeichen (U+0000 bis U+00FF) darstellen.

Standard ASCII-Zeichentabelle ($00 - $7F)

$00	$01	$02	$03	$04	$05	$06	$07
$08	$09	$0A	$0B	$0C	$0D	$0E	$0F
$10	$11	$12	$13	$14	$15	$16	$17
$18	$19	$1A	$1B	$1C	$1D	$1E	$1F
$20	$21 !	$22 "	$23 #	$24 $	$25 %	$26 &	$27 '
$28 (	$29)	$2A *	$2B +	$2C ,	$2D -	$2E .	$2F /
$30 0	$31 1	$32 2	$33 3	$34 4	$35 5	$36 6	$37 7
$38 8	$39 9	$3A :	$3B ;	$3C <	$3D =	$3E >	$3F ?
$40 @	$41 A	$42 B	$43 C	$44 D	$45 E	$46 F	$47 G
$48 H	$49 I	$4A J	$4B K	$4C L	$4D M	$4E N	$4F O
$50 P	$51 Q	$52 R	$53 S	$54 T	$55 U	$56 V	$57 W
$58 X	$59 Y	$5A Z	$5B [	$5C \	$5D]	$5E ^	$5F _
$60 `	$61 a	$62 b	$63 c	$64 d	$65 e	$66 f	$67 g
$68 h	$69 i	$6A j	$6B k	$6C l	$6D m	$6E n	$6F o
$70 p	$71 q	$72 r	$73 s	$74 t	$75 u	$76 v	$77 w
$78 x	$79 y	$7A z	$7B {	$7C |	$7D }	$7E ~	$7F

Erweiterte Zeichentabelle ($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

24 Abkürzungsverzeichnis

Abkürzung	Deutsch	Englisch
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
LWL	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
PADU	Parallel-Analog-Digital-Umsetzer (Varianten: PADU-S, PADU-S-IT)	Parallel analog digital unit
PC	Einzelplatzrechner	Personal Computer

Abkürzung	Deutsch	Englisch
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	Siehe 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

25 Stichwortverzeichnis

A

B

C

D

R

S

T

U

V

W

Z

26 Support und Kontakt

Support

Tel.: +49 911 97282-14

Fax: +49 911 97282-33

E-Mail: support@iba-ag.com

Hinweis

Wenn Sie Support benötigen, dann geben Sie die Seriennummer (iba-S/N) des Produktes an.

Kontakt

Zentrale

iba AG
Königswarterstraße 44
90762 Fürth
Deutschland

Tel.: +49 911 97282-0

Fax: +49 911 97282-33

E-Mail: iba@iba-ag.com

Kontakt: Harald Opel

Regional und weltweit

Weitere Kontaktadressen unserer regionalen Niederlassungen oder Vertretungen finden Sie auf unserer Webseite

www.iba-ag.com.