ibaLogic-V4

Руководство

Версия 4.2.1

Measurement and Automation Systems

Торговые марки

Windows® является названием и зарегистрированной торговой маркой компании Microsoft Corporation. Другие продукты и названия компаний, упомянутые в настоящем руководстве, также могут являться зарегистрированными торговыми марками и принадлежать соответствующим лицам.

Авторское право

Отказ от ответственности

Содержание данной публикации было проверено на предмет соответствия описанному аппаратному и программному обеспечению. Отклонения, однако, не могут быть исключены, поэтому гарантия полного совпадения не предоставляется. Информация, содержащаяся в настоящем руководстве, регулярно актуализируется. Необходимые изменения содержатся в последующих изданиях или могут быть загружены из Интернета.

Актуальную версию можно загрузить с нашего сайта www.iba-ag.com после регистрации.

Производитель не несет ответственности за утрату или порчу программного обеспечения, которые являются следствием неправильного использования или несоблюдения (также частичного) инструкций, описанных в настоящем руководстве пользователя.

Сертификаты

Продукт сертифицирован в соответствии с европейскими стандартами и директивами и соответствует требованиям безопасности и охраны здоровья.

Требования других международных и местных стандартов и указаний также были соблюдены.

Производитель

iba AG

Koenigswarterstrasse 44

90762 Fuerth

GERMANY

Контактная информация

Центральный офис: +49 911 97282-0

Факс: +49 911 97282-33

Тех. поддержка: +49 911 97282-14

Технологический отдел: +49 911 97282-13

E-mail: iba@iba-ag.com

Web: www.iba-ag.com

Версия	Дата	Исправление – Раздел / страница	Автор	Версия ПО
_en	05.2011	1-е издание	CH/KaFe	4.2.1

Содержание

Об этом руководстве пользователя

Это руководство пользователя предназначено для передачи специалистам, ответственным за установку, пуско-наладку и эксплуатацию оборудования, а также за обслуживание и обновление программного обеспечения.

Ответственное лицо должно удостовериться в том, что соответствующие специалисты были ознакомлены с информацией, содержащейся в настоящем руководстве и другой сопутствующей документации.

Назначение этого руководства

В этом руководстве пользователя содержится описание использования программного продукта ibaLogic-V4.

Данное руководство представляет собой справочный документ, который включает в себя все необходимые инструкции, которые должны соблюдаться для обеспечения общей безопасности, а также для правильной установки, эксплуатации и обслуживания ПО. Это руководство, включая все инструкции по безопасности (а также дополнительную документацию по модулям других поставщиков), должно:

- быть прочитано и понято специалистами, которые работают с ibaLogic, все рекомендации, содержащиеся в руководстве должны соблюдаться (особенно те, которые касаются обеспечения безопасности)
- быть легко доступно всем

Цели:

- Работа с программным обеспечением ibaLogic
- Предотвращение несчастных случаев

Хранение

Местонахождение руководства должно быть известно всем заинтересованным лицам. К информации, содержащейся в руководстве, необходимо обращаться в случае возникновения малейших сомнений в правильности дальнейших действий при работе с ibaLogic.

Примечание

В случае пропажи руководства немедленно свяжитесь с производителем программного продукта!

Храните руководство в таком месте, где оно будет доступно всем заинтересованным специалистам.

Целевая аудитория

Это руководство предназначено для специалистов, которые работают с электрическими и электронными модулями и обладают необходимыми знаниями в области коммуникационных и измерительных технологий.

К таким специалистам относятся лица, которые соблюдают правила техники безопасности и могут оценить возможные последствия и риски, исходя из своей

профессиональной подготовки, специальных знаний и опыта, а также знания соответствующих стандартных правил.

Описанное в данном руководстве программное обеспечение должно работать только в пределах указанных здесь параметров и использоваться только специально подготовленным персоналом. Специалисты, работающие с ПО, должны всегда соблюдать инструкции и правила, предусмотренные для конкретного приложения, особенно в отношении соблюдения правил безопасности.

Условные обозначения

В настоящем руководстве используются следующие условные обозначения:

Действие	Значение
Команда меню	Меню «Функциональная схема»
Вызов команды меню	«Шаг 1 – Шаг 2 – Шаг 3 – Шаг x» Пример: Выбор меню «Функциональная схема – Добавить – Новый функциональный блок»
Клавиши	<Название клавиши> Пример: <Alt>; <F1>
Одновременное нажатие клавиш	<Название клавиши> + <Название клавиши> Пример: <Alt> + <Ctrl>
Кнопки	<Название кнопки> Пример: <OK>; <Cancel>
Имена файлов, пути	«Имя файла» Пример: «Test.doc»
Исходный код	Пример: `if (i2=Schalter_Vor) then`

Используемые символы

При чтении этого руководства вам могут встретиться символы, которые имеют следующее значение:

⚠ DANGER

Несоблюдение техники безопасности может привести к травме или смертельному исходу.

⚠ WARNING

Несоблюдение этого правила безопасности может привести к травме или причинить материальный ущерб!

Примечание

В примечании указаны особые требования или действия, которые необходимо выполнить.

Важно

Указывает на некоторые особенности, например исключения из правил.

Совет

Советы, наглядные примеры и маленькие хитрости, позволяющие облегчить работу.

Дополнительная документация

Ссылка на дополнительную документацию или специальную литературу.

Введение

Идентификация

Контроллер PAC (Soft PLC) и менеджер сигналов "ibaLogic-V4".

Использование продукта

Продукт / система используется для осуществления измерений и управления агрегатами и системами.

ibaLogic не предназначена для использования с системами обеспечения безопасности.

В этом случае безопасность и защита продукта / системы могут быть нарушены. Компания iba AG не несет ответственности за потери данных или повреждение продукта / системы в случае нецелевого использования.

⚠ DANGER

Возможные проблемы при активации функций и других служб!

При активации функций и служб (PMAC, OPC …), которые оказывают непосредственное влияние на отклик системы, возможно причинение травм людям и нанесение ущерба оборудованию.

Соблюдайте правила безопасности при работе с системой!

Примечания к настоящей версии руководства

Файл с изменениями

На установочном диске имеется файл с изменениями (ChangeLog.txt), который содержит, помимо прочего, следующую важную информацию:

- Новые функции
- Исправленные ошибки

1 Установка программного обеспечения

1.1 Системные требования

1.1.1 Аппаратное обеспечение

Требования к аппаратному обеспечению содержатся в следующей таблице.

	Минимальные требования	Рекомендуемые или выше
Частота ЦП	1,6 ГГц	2,0 ГГц
Количество ЦП	1	2
Оперативная память	768 Мб	2 048 Мб
Разрешение экрана	1024 x 786	1280 x 1024

Минимальные требования к памяти: 650 Мб. Размер базы данных SQL server express может достигать 4 Гб. Убедитесь в том, что для ваших целей у вас достаточно свободного пространства памяти.
Более подробная информация содержится в пункте 11.4, "Ограничения производительности".

Примечание

Установка также возможна, если аппаратное обеспечение не соответствует минимальным требованиям, однако это может привести к ограничению производительности системы.

1.1.2 Программное обеспечение

Должна быть установлена одна из следующих операционных систем:

- Windows XP Professional SP3 или
- Windows 2003 Server

Нижеперечисленные программные пакеты содержатся на CD, который входит в объем поставки:

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

Мастер установки проверит наличие версий различных пакетов программного обеспечения. Если какой-либо из программных пакетов отсутствует или версия устарела, мастер установки его установит или выполнит обновление версии.

1.2 Активация лицензии

Аппаратный ключ поставляется уже сконфигурированным для нужд конкретного пользователя. Аппаратный ключ пользователя генерирует в системе виртуальный ключ, который разблокирует или активирует различные функции.

⚠ WARNING

В демо-режиме выключать контроллер (PMAC) опасно!

Нельзя выполнять пуско-наладку системы, если в аппаратном ключе отсутствует соответствующая лицензия. От лицензии зависит разблокировка и активация функций.

Во время выполнения всех операций по разблокированию и активации функций аппаратный ключ должен быть вставлен!

Если ключ не вставлен, то ibaLogic работает только в демо-режиме. В этом случае контроллер выключится через 4 часа.

В демо-режиме ibaLogic поддерживает следующее:

- Все карты iba и reflective memory
- Библиотеки DLL
- Запись файлов iba (в этом режиме отмечать файлы нельзя)
- 128 OPC-сигналов
- 64 сигнала ввода-вывода

Рис. 1: Аппаратный ключ

Последовательность действий

➲ Вставьте аппаратный ключ в USB-интерфейс.

После запуска сервера и клиента появится следующее сообщение: "Онлайн-сервер: состояние драйвера: запущен драйвер для аппаратного ключа Vxxxxxx" ("Online Server: DriverStatus: Driver running for Dongle Vxxxxxx").

1.3 Установка программного обеспечения

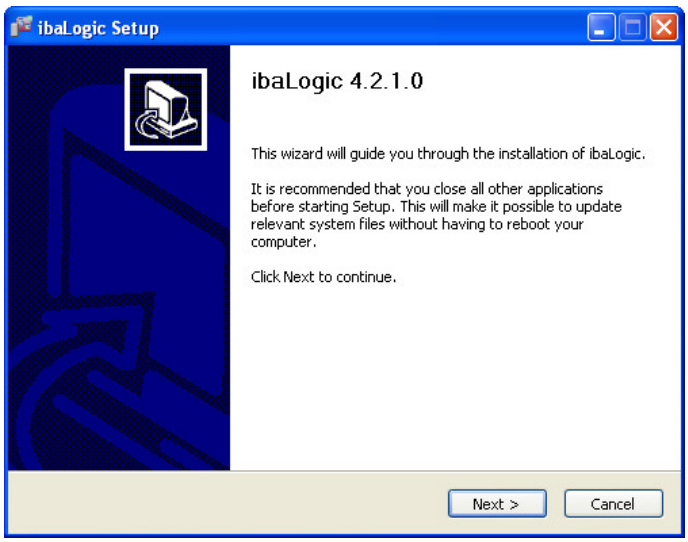
Следуйте инструкциям мастера установки ibaLogic.

Необходимые условия

- ❑ Соответствие требованиям к аппаратному обеспечению.
- ❑ Соответствие требованиям к программному обеспечению.

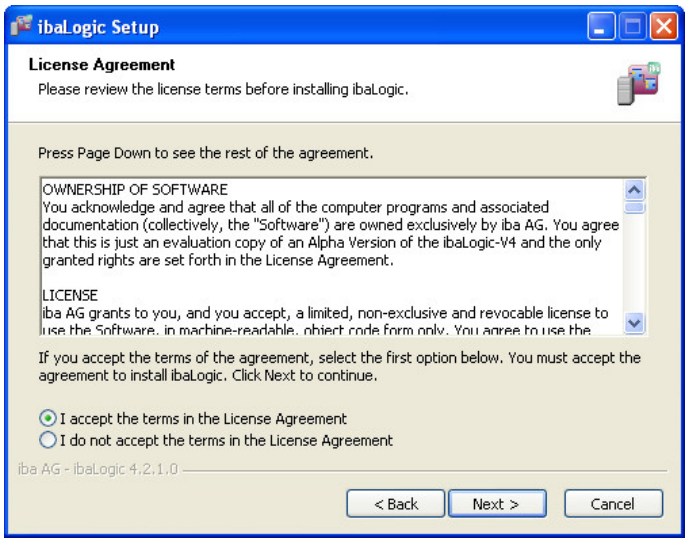
Последовательность действий

1. Двойным щелчком запустите файл "Setup-4.x.xx.exe".



Лицензионное соглашение

2. Чтобы начать установку ibaLogic, примите условия лицензионного соглашения.

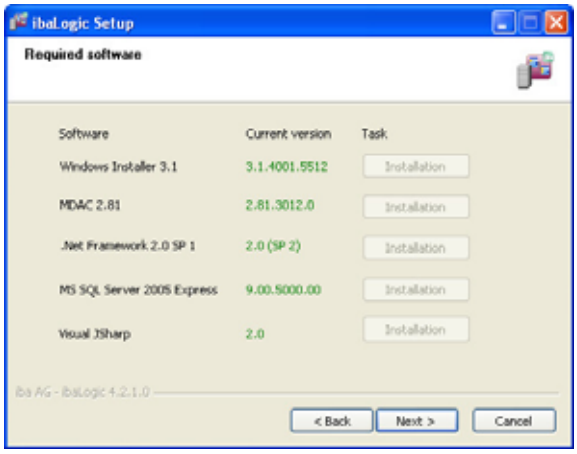


Необходимое программное обеспечение

Необходимые компоненты с указанием требуемых версий перечислены в пункте "Программное обеспечение".

Отображение	Объяснение
Номер версии отображается зеленым MS SQL Server 2005 Express 9.00.4035.00 Installation	Установленное ПО имеет необходмую функциональность.
Номер версии отображается красным MS SQL Server 2005 Express 0.0 Installation	ПО не установлено или не соответствует требованиям. Активирована кнопка <Установка>.

3. Установите или обновите компонент(ы) программного обеспечения, щелкнув по кнопке <Установка>.



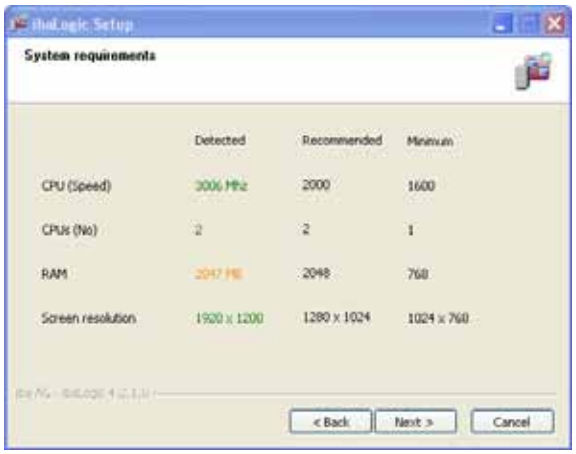
4. Если потребуется, щелкните кнопку подтверждения в появившемся диалоговом окне. После того как все программные компоненты были установлены или обновлены, щелкните <Далее>.

Системные требования

Перед установкой программных компонентов выполняется проверка системных требований.

Отображение	Объяснение
Зеленый	Рекомендуемые требования соблюдены.
Оранжевый	Минимальные требования соблюдены.
Красный	Минимальные требования не соблюдены.

5. Если системные требования выше минимальных, продолжайте установку.

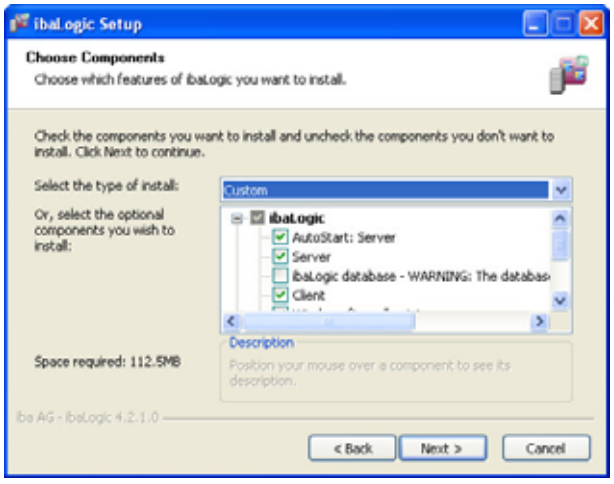


Выбор компонентов

Вы можете использовать поле выбора "Тип установки" для выбора компонентов перед собственно установкой.

Тип установки	Объяснение
Полная	Выбор компонентов невозможен. Устанавливается полный пакет ПО.
Только сервер	Выбор компонентов невозможен. Устанавливается полный пакет ПО для сервера.
Только клиент	Выбор компонентов невозможен. Устанавливается полный пакет ПО для клиента.
Пользовательская установка	Выбор компонентов возможен. Устанавливаются только выбранные компоненты.

6. Определите нужные вам компоненты ibaLogic, выбрав соответствующий тип установки.



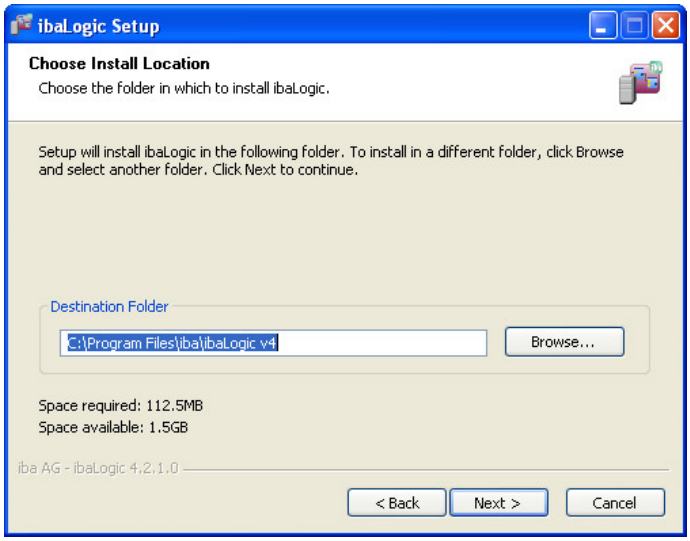
⚠ WARNING

В случае обновления ibaLogic напротив опции "База данных ibaLogic" ("ibaLogic Database") галочка не стоит. Если напротив этой опции будет поставлена галочка, то существующая база данных, включая все проекты, будет удалена в процессе установки. Также появляется предупреждающее сообщение: "База данных уже существует и будет перезаписана" ("The database already exists and will be overwritten").

Выбор каталога для установки ПО

В выбранном каталоге будет создана структура папок ibaLogic (сервер, клиент и т.д.). Компания iba рекомендует оставить для установки каталог, указанный по умолчанию.

7. Определите целевой каталог.

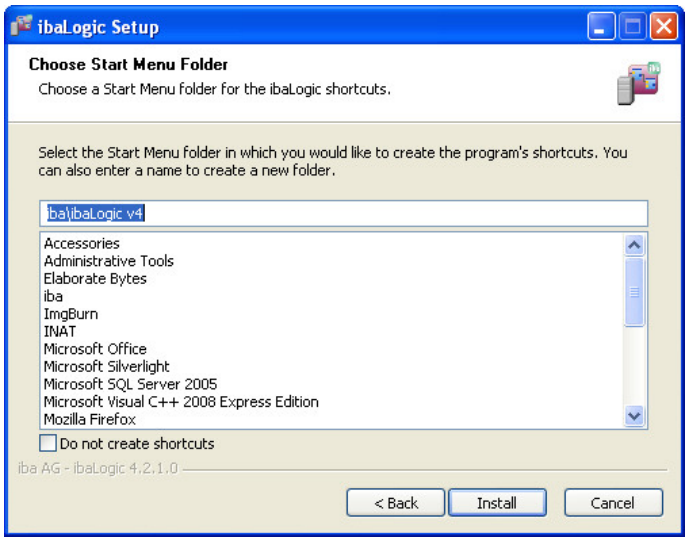


Определите папку для меню запуска

В "папке меню запуска" ("Start menu folder") создаются ярлыки для программы.

Если в использовании ярлыка на рабочем столе нет необходимости, поставьте галочку напротив опции "Не создавать ярлыки" ("Do not create shortcuts").

8. Ярлык для программы создается после щелчка по кнопке <Установить> (<Install>).

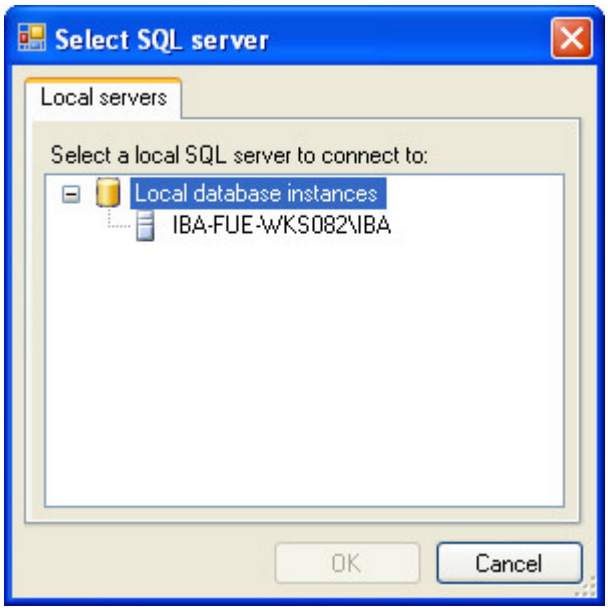


Выбор SQL-сервера

В процессе установки ПО мастер установки выполнит поиск серверов Microsoft SQL на локальном компьютере и отобразит диалоговое окно "Выбор SQL-сервера" ("Select SQL server").

Если при установке была поставлена галочка напротив опции "База данных iba-Logic", в диалоговом окне "Локальные сервера" будет отображен экземпляр базы данных IBA.

9. Выберите либо экземпляр по умолчанию "<Имя ПК>\IBA", либо другой на ваше усмотрение. Закройте диалоговое окно, щелкнув кнопку <OK>. Выбранная база данных ibaLogic будет установлена на выбранном сервере.

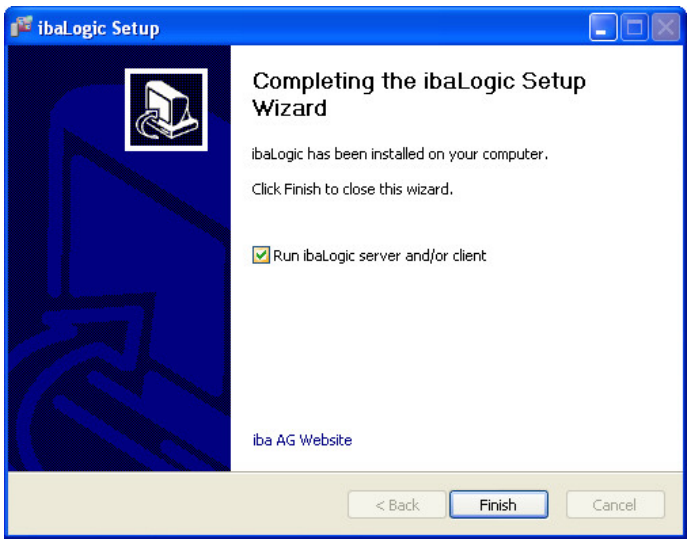


Примечание

Любая существующая база данных ibaLogic может быть перезаписана после подтверждения в появившемся диалоговом окне. Если вы закроете диалоговое окно нажатием кнопки <Нет>, то существующая база данных не будет изменена.

Завершение установки ibaLogic

10. Для завершения установки щелкните <Готово>.



Результат

ibaLogic успешно установлена.

2 Программное обеспечение ibaLogic

2.1 Введение

Компания iba AG имеет многолетний опыт в области технологий сбора измеренных данных на предприятиях тяжелой промышленности. Основное внимание специалистов компании было сосредоточено на агрегатах для производства и обработки стали и цветных металлов.

Программы компании iba AG для сбора[1] и анализа[2] записанных данных используются на предприятиях всего мира и входят в объем поставки крупных компаний, поставляющих оборудование для промышленности и занятых в сфере автоматизации технологических процессов.

В настоящее время существует большое количество разнообразных систем от разных производителей. В этой связи iba разработала решения, позволяющие соединить различные поколения устройств многих известных компаний. На данном этапе развития технологий место "однонаправленного потока" измерений занимает "двунаправленный поток", который обеспечивает обмен данными между различными автоматизированными системами управления - функция, которая является неотъемлемой частью современного рынка модернизации и реконструкции уже существующих автоматизированных систем.

Чтобы обеспечить соответствие этому требованию, в 1995 году компания iba AG разработала свободно программируемый менеджер сигналов. Стандарт IEC 1131-3, который уже был разработан в тот период, используется для описания производственных процессов при помощи графических элементов и встраиваемых программных технологий. Он значительно упрощает описание сигналов в сложных технологических процессах.

Графическое программирование[3], стимулом для развития которого послужил этот стандарт, в настоящее время является основой подавляющего большинства автоматизированных систем. В результате этого графическое программирование характеризуется высокой совместимостью и переносимостью из одной среды в другую.

[1] ibaPDA
[2] ibaAnalyzer
[3] FBD - это графический язык программирования, в котором используются взаимосвязанные функциональные блоки, в отличие от последовательности текстовых команд, как в случае классических языков программирования. В качестве примера того, как выглядит продукт, созданный с помощью графического языка программирования, можно привести схемы электрооборудования. Отображение программы соответствует требованиям разработчиков ПО для контроллеров, поскольку, как правило, профессиональный опыт таких специалистов связан с электротехникой. Различные функциональные блоки часто создаются отдельно с помощью других языков программирования, например структурированного текста, и могут предоставляться производителем автоматизированных систем, а также их могут писать или импортировать пользователи этих систем.

2.2 Сферы применения

ibaLogic используется для следующего:

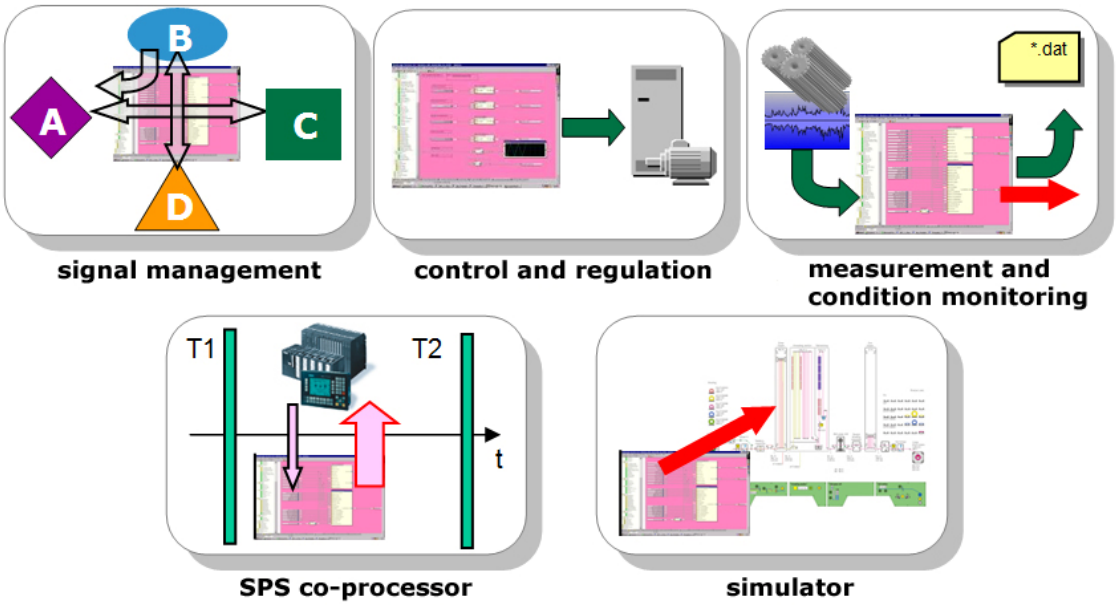


Рис. 2: Сферы применения

Управление сигналами

Предлагаемые iba возможности подключения и взаимодействия позволяют объединить разные поколения автоматизированных устройств и систем различных производителей.
Двунаправленный обмен данными делает возможным связь с контроллерами, которые, при иных условиях, были бы несовместимы.

Сопроцессор ПЛК

ibaLogic может также играть роль сопроцессора.

Вертикальные зеленые линии на рис. "Сферы применения" обозначают период дискретизации исходного автоматизированного оборудования. Передача данных в программный контроллер (ibaLogic) выполняется после одного тактового импульса ПЛК (T1). С помощью ibaLogic можно также выполнять сложные вычисления, результаты которых будут получены до T2, благодаря процессорной мощности ПК и доступных форматов данных. Проекты по модернизации и переоборудованию можно выполнять с помощью метода пошаговой передачи в новую автоматизированную систему на базе ibaLogic функций управления выполняемых «старыми» контроллерами.

Мониторинг измерений и состояний (Condition Monitoring)

Помимо использования ibaLogic для управления сигналами специалисты компании iba решили добавить программе функции, позволяющие выполнять сложные вычисления, которые невозможны в контексте стандартной системы PDA. Добавление блока, позволяющего выполнять сбор измеренных данных, значительно расширило функции ibaLogic. Пользователь может настроить триггерное измеренние данных и сохранение полученных значений на различных носителях с помощью "DAT_FILE_WRITER". Впоследствии собранные данные

могут обрабатываться и анализироваться посредством различных инструментов, разработанных компанией iba, например ibaAnalyzer, ibaDatCoordinator и т.д.

Автоматизация (контроль и управление)

ibaLogic создана на базе графического языка программирования, описанного в стандарте IEC 1131-3. Этот язык был специально разработан для программируемых логических контроллеров (ПЛК). Последние несколько лет продемонстрировали, что рынок как измерительных систем, так и систем управления развивается быстрыми темпами. Логическим выводом всего вышесказанного является то, что ibaLogic может использоваться как полноценный контроллер автоматизированной системы.

Если в автоматизированной системе задачи, выполняемые обычно аппаратным контроллером, выполняются программно, значит, речь идет о программируемом логическом контроллере на базе компьютера (Soft PLC или сокращенно PAC).

ibaLogic позволяет разделить ОС и исполнительную систему, посредством переноса последней на автономное "интеллектуальное" устройство ibaPADU-S-IT. В этом случае можно выключать ПК, поскольку исполнительная система при этом будет продолжать работать. Компьютер будет использоваться только как среда разработки, как и в случае других систем на базе ПЛК.

Симулятор

В данном случае речь идет об активном симуляторе, программируемом средствами ibaLogic. Интерфейс оператора и результаты симуляции выводятся на экране HMI. Для установления соединения между симулятором ibaLogic и HMI используется стандартный OPC-интерфейс.

2.3 Компоненты ibaLogic

Система ibaLogic основана на клиент-серверной модели. Такая концепция системы упрощает ее работу в децентрализованной и многопользовательской среде.

ibaLogic состоит из следующих компонентов:

- Исполнительная система (PMAC)
- Сервер ibaLogic
- Клиент ibaLogic
- База данных
- OPC-сервер

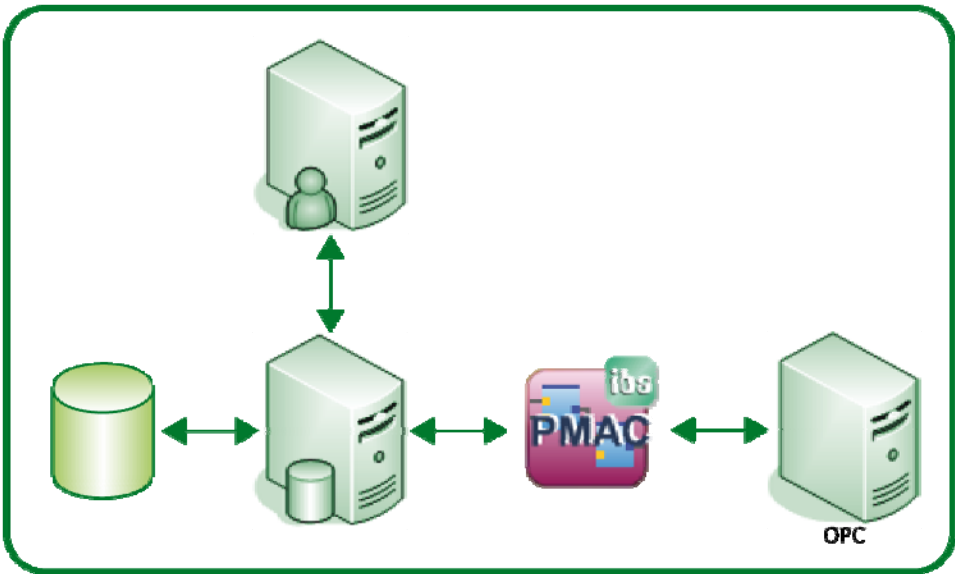


Рис. 3: ibaLogic: компоненты

База данных

OPC-сервер

Сервер ibaLogic

Исполнительная система "PMAC"

Клиент ibaLogic

В самом простом случае все вышеупомянутые компоненты находятся в одном компьютере. Однако, эти компоненты могут также работать на отдельных ПК.

Исполнительная система (PMAC)

При запуске проекта в клиенте ibaLogic он компилируется и загружается в "программируемый контроллер для измерений и автоматизации" (Programmable Measurement and Automation Controller" - PMAC). Исполнительная система может работать на компьютере с ОС Windows или на устройстве, совместимом с Windows CE (ibaPADU-S-IT).

Исполнительная система постоянно возвращает клиенту текущие вычисляемые значения. Эти значения отображаются онлайн в графическом пользовательском интерфейсе проекта.

Если проект был перенесен в исполнительную систему и запущен, то он может работать самостоятельно - без запуска сервера / клиента.

Сервер ibaLogic

ibaLogic - это система, основанная на базе данных. Сервер ibaLogic выполняет основные функции управления базой данных и обменом данных между клиентом ibaLogic и исполнительной системой.

Сервер ibaLogic управляет всеми проектами ibaLogic в базе данных.

ibaLogic использует базу данных Microsoft SQL Express, которая не треует лицензии. Во время установки ibaLogic создается также сервер Microsoft SQL Express с соответствующей базой данных (если это не было предварительно выполнено пользователем).

Диалоговое окно сервера ibaLogic также используется для создания резервных копий и восстановления приложений ibaLogic. Резервная копия базы данных сохраняется во внешнем файле.

Если в проекты ibaLogic вносятся какие-либо изменения, они автоматически сохраняются в базе данных. В процессе пользовательской настройки проекта нет необходимости специально сохранять изменения.

Клиент ibaLogic

Клиент ibaLogic - это среда программирования, в которой выполняется программирование и конфигурация проектов ibaLogic. Для этого должен быть подключен сервер ibaLogic, который работает на том же самом компьютере или в сети.

К тому же клиент ibaLogic управляет загрузкой, запуском и остановкой проекта ibaLogic в исполнительной системе с помощью сервера ibaLogic.

OPC-сервер

OPC-сервер предоставляет подключенным OPC-клиентам все переменные, которые были заявлены как OPC-видимые. В целом, OPC-клиенты представляют собой системы HMI (Human Machine Interface). По умолчанию OPC-сервер работает на одном ПК с сервером ibaLogic, однако он может также работать и на других компьютерах в сети и соединяться непосредственно с контроллером (PMAC) по TCP/IP.

Работа с несколькими клиентами и другие конфигурации системы

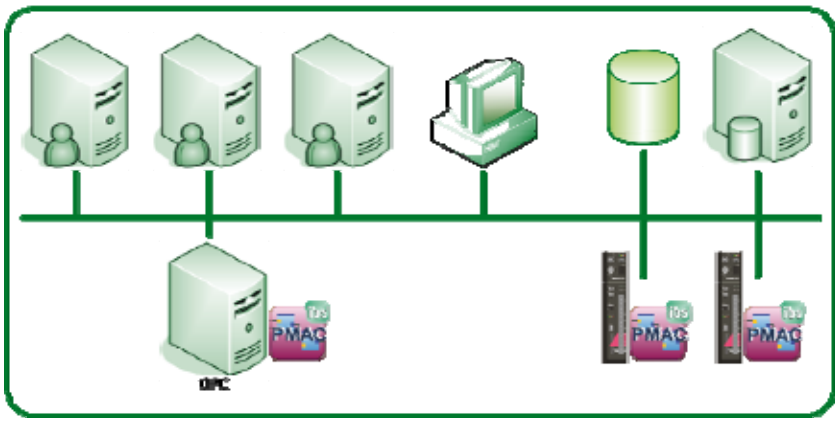


Рис. 4: Возможная конфигурация системы


База данных


HMI


Сервер ibaLogic

Исполнительная система "PMAC"


Клиент ibaLogic


PADU-S-IT


OPC-сервер

В самом простом случае ibaLogic и все ее компоненты могут работать на одном компьютере с ОС Windows.

Компоненты ibaLogic также могут быть распределены между несколькими компьютерами. На рис. выше приведен пример конфигурации, в которой используются 3 приложения ibaLogic. Одно работает на ПК, а каждое из двух других соответственно на одном головном узле - PADU-S-IT. Центральный сервер соединен с одной базой данных, в которой хранится резервная копия всех трех проектов. Из базы данных на этот сервер загружается только одна рабочая область ibaLogic, которая может содержать три проекта для трех контроллеров (PMAC).

В режиме работы с несколькими клиентами несколько пользователей могут одновременно работать в общей рабочей области.

Однако, в одной рабочей области в один период времени может быть активен только один проект / приложение. Выполнять мониторинг и работать **онлайн** можно только с этим активным PMAC.

В HMI-систему информация может передаваться через OPC-сервер. На устройстве PADU-S-IT OPC-сервер запустить нельзя. Соответственно, обмен данными между HMI и головными узлами PADU-S-IT должен осуществляться через ПК с Windows.

Важно

Эти опции доступны в архитектуре ibaLogic. Режим работы с несколькими клиентами в текущей версии, 4.2.1, пока недоступен.

2.4 Режимы работы и обработки данных

ibaLogic имеет несколько режимов работы, что позволяет обеспечить соответствие требованиям различных областей применения системы. Специалисты компании iba разработали несколько режимов обработки данных, поскольку ibaLogic используется не только как программный контроллер (soft PLC), но также как диспетчер сигналов, симулятор и программа обработки сигналов.

В ibaLogic можно выбрать следующие режимы работы:

- Измерение
- Программный контроллер (Soft PLC)

Настройка режима работы

Последовательность действий

1. Выберите в меню "Инструменты – Конфигуратор ввода-вывода" ("Extras – I/O configurator"). Появится окно конфигуратора ввода-вывода.
2. Опции режимов работы находятся во вкладке "Конфигурация аппаратного обеспечения – Общие настройки" ("Hardware configuration – General settings").
3. В "Общих настройках" активируйте тот режим работы, который вам необходим.
4. Щелкните <Применить> (<Accept>).
5. Чтобы закрыть конфигуратор ввода-вывода, щелкните <OK>.

В ibaLogic можно выбрать следующие режимы обработки данных:

"Буферный режим" ("Buffered mode") (= передача пакетов)

Дополнительная информация содержится в разделах 11.2, "Время отклика" и 9.4.2, ""Режим буферизации" FOB-карт".

Настройка режимов обработки данных

Последовательность действий

1. В дереве элементов в левой части окна выберите аппаратные средства, для которых нужно настроить режимы обработки данных. Для выбранного аппаратного обеспечения отобразится вкладка "Конфигурация аппаратного обеспечения".
2. В "Настройках соединения" активируйте требуемый режим.
3. Щелкните <Применить> (<Accept>).
4. Чтобы закрыть конфигуратор ввода-вывода, щелкните <OK>.

Примечание

Более подробная информация содержится в разделе 9.2.1, "Общие настройки"".

2.5 Структура приложения ibaLogic

Структура приложения ibaLogic состоит из следующих элементов:

- Проект 1
  - Задача 1 / Программа 1
  - Задача 2 / Программа 2
  - Задача n / Программа n
- Проект 2
  - Задача 1 / Программа 1
  - Задача 2 / Программа 2
  - Задача n / Программа n
- Проект n

Каждую программу и ее свойства (интервал задачи и т.д.) можно отнести только к одному проекту. Проекты, в свою очередь, должны быть организованы в рабочей области.

Для одной исполнительной системы / PMAC создается один проект. В пределах одной рабочей области только один проект может быть настроен как активный, т.е. только этот активный проект может быть запущен или остановлен.

Примечание

В рамках соответствующего приложения может быть активен только один проект.

2.5.1 Свойства задач / программ

В соответствии с IEC 1131-3 одной задаче может быть присвоено несколько программ. Однако в ibaLogic реализована строгая модель: одна задача - одна программа.

Для каждой задачи можно определить следующие свойства:

- Интервал
- Приоритет

Интервал определяет то, через какой промежуток времени будет выполняться перезапуск задачи. Минимальное значение этого параметра - 1 мс.

Настройка приоритетности определяет то, в какой последовательности будут исполняться программы проекта, начиная с нулевого (0) приоритета.

Примечание

Свойства программы "Интервал" и "Приоритет" оказывают большое влияние на быстродействие проекта. Подробное описание этих программных свойств содержится в разделах "Время отклика" и "Ограничения производительности".

2.5.2 Элементы программы

Программа ibaLogic может состоять из следующих элементов:

- Функциональные блоки
- Функциональные блоки интегрированной стандартной библиотеки
- Функциональные блоки, созданные пользователем с помощью структурированного текста
- Макросы, созданные пользователем.
- Функциональные блоки на основе DLL, созданные пользователем
- Соединительные элементы
- Комментарии

2.5.2.1 Функциональные блоки

В ibaLogic есть библиотека функциональных блоков, которая содержит как стандартные функциональные блоки в соответствии с IEC 1131-3, так и дополнительные.

Эти функциональные блоки пользователь может объединять в макросы, что улучшает структурированность программы, а также обеспечивает инкапсуляцию различных графических подпрограмм.

Пользователь может самостоятельно создавать отдельные функциональные блоки, которые необходимы ему для решения определенной задачи.

Для этих целей в ibaLogic предусмотрена функция создания новых функциональных блоков посредством структурированного текста. ST-код (структурированный текст) отображается для пользователя, который может внести в него изменения.

Один из вариантов создания пользовательских функциональных блоков - это пользовательские библиотеки DLL (с помощью DLL-фреймворка, предоставлемого компанией iba). В этом случае код не будет отображаться для пользователя. Созданный таким способом блок будет иметь вид стандартного функционального блока.

Дополнительная документация

Дополнительную информацию по созданию DLL можно найти на CD, который входит в объем поставки, в папке "iba software and manuals".

2.5.2.2 Графическое программирование

Для соединения функциональных блоков доступны следующие элементы:

- Соединительные линии
- Внутристраничные коннекторы
- Коннекторы вне задачи

В графическом программировании соединение функциональных блоков выполняется с помощью соединительных линий. Для получения более

структурированной программы можно использовать внутристраничные коннекторы.

Внутристраничные коннекторы (intra-page connectors - IPC) упрощают графическое отображение программы. В этом случае вместо соединительной линии используется IPC. Использование этой опции оптимизирует процесс программирования, особенно в том случае, если требуется соединить несколько объектов на одной странице с одной точкой или необходимы "длинные" соединения на несколько страниц.

Коннекторы вне задачи (off-task connectors) представляют собой независимые от программы соединительные элементы. Соединения такого типа необходимы в случае, если требуется связь между несколькими программами.

Коннекторы вне задачи также используются для обмена данными между ibaLogic и OPC-клиентами, которое можно сконфигурировать в коннекторе вне задачи.

2.5.2.3 Комментарии

Комментарии используются для структурирования и упрощения описания программы. Их можно размещать в любом месте и даже добавлять как отдельный элемент.

2.5.2.4 Доступные типы данных

ibaLogic поддерживает все элементарные типы данных, перечисленные в стандарте IEC 1131-3 (исключение: WSTRING).

2.5.2.5 Интегрированные измерения с помощью ibaPDA Express

Вы можете использовать ibaPDA Express для быстрого отображения волны сигнала.

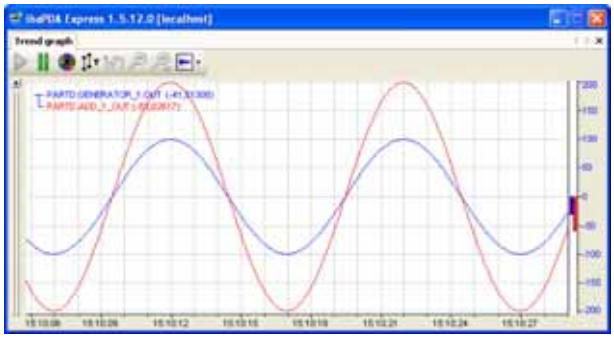


Рис. 5: Интегрированные измерения с помощью ibaPDA Express

Сигналы для отображения в ibaPDA Express можно перенести простым перетаскиванием (Drag & Drop).

ibaPDA Express не сохраняет данные для долговременной записи.

2.5.2.6 Сохранение измеренных значений

Сохранять измеренные значения можно с помощью лицензионного (охраняемого законом об авторском праве) функционального блока "DAT_FILE_WRITE". Более подробная информация содержится в пункте 5.3.1 "DAT_FILE_WRITE (функциональный блок DFW)".

Совет

Отображение и оценка измеренных сигналов, содержащихся в *.dat-файлах, может выполняться с помощью удобной и простой в использовании программы для анализа - ibaAnalyzer.

2.6 Возможности сетевого взаимодействия

Системы ibaLogic могут обмениваться данными посредством интерфейсов, доступных в компьютере с ОС Windows или в PADU-S-IT (например, посредством TCP/IP, ibaNet и т.д.).

Коммуникация с внешними системами или отдельными модулями ввода-вывода отображена на обзорной схеме сетевого взаимодействия.

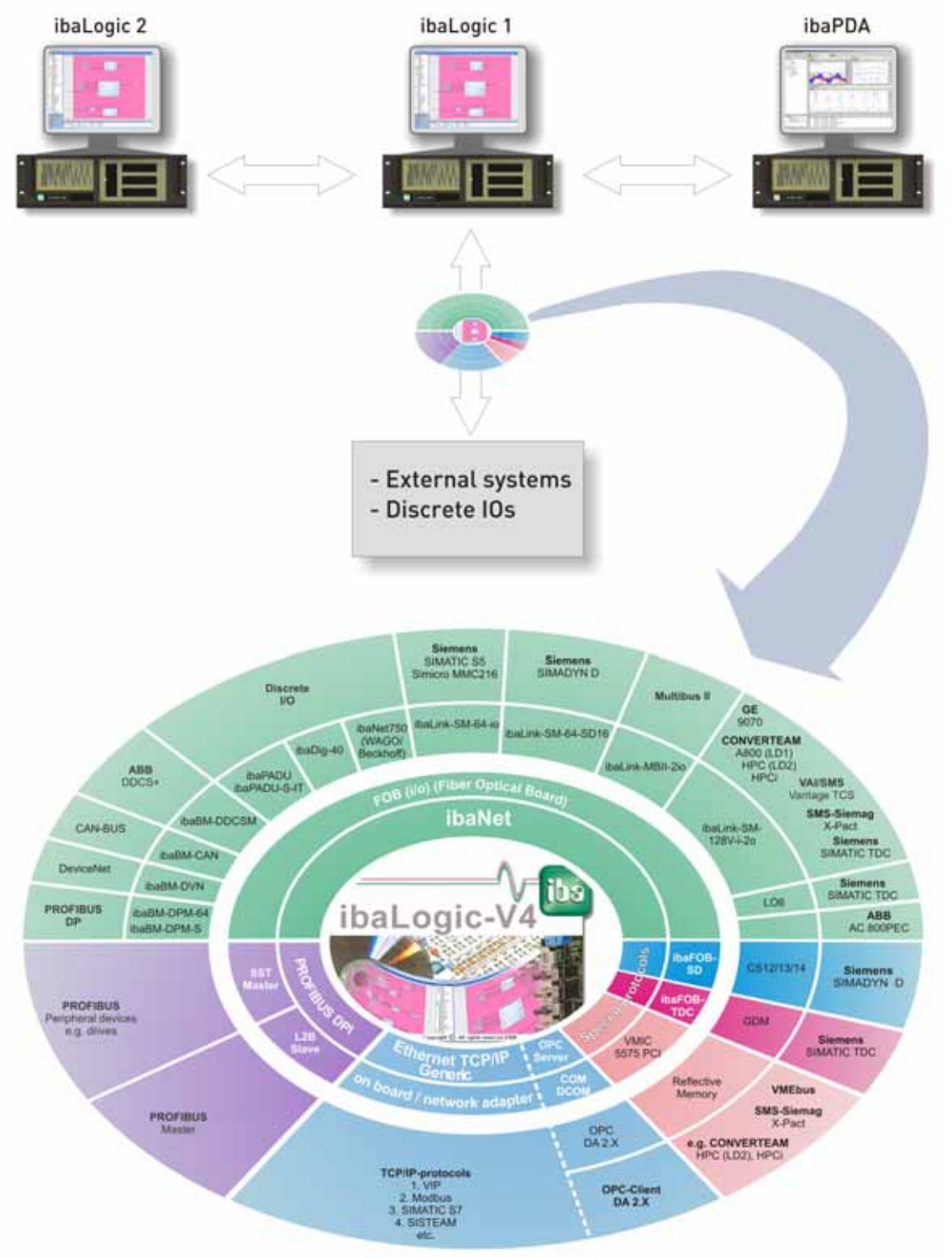


Рис. 6: Возможности сетевого взаимодействия iba и внешних систем

3 Сервер ibaLogic

3.1 Обзор функций сервера ibaLogic

Сервер ibaLogic не только является центральным элементом обмена данными между клиентом ibaLogic и контроллером (PMAC), но и управляет проектами ibaLogic V4 в базе данных. В фоновом режиме сервер управляет соединением с активным PMAC для загрузки / запуска / остановки PMAC. Это соединение устанавливается с помощью клиента ibaLogic.

Соответственно, функции сервера можно классифицировать следующим образом:

- Работа сервера
  - Запуск / Остановка / Закрытие
  - Действия, связанные с базой данных
    - Резервное копирование и восстановление
    - Сброс всей текущей базы данных
- Административные настройки
  - Настройка номера порта для клиентских подключений
  - Настройка соединений с базами данных ibaLogic и их параметров
  - Конфигурация автозапуска для сервера и локального PMAC
  - Настройка автоматического создания резервной копии текущей базы данных
  - Настройка общих параметров сервера
  - Отображение состояния / исполнение сценариев базы данных (только для целей тех. поддержки)

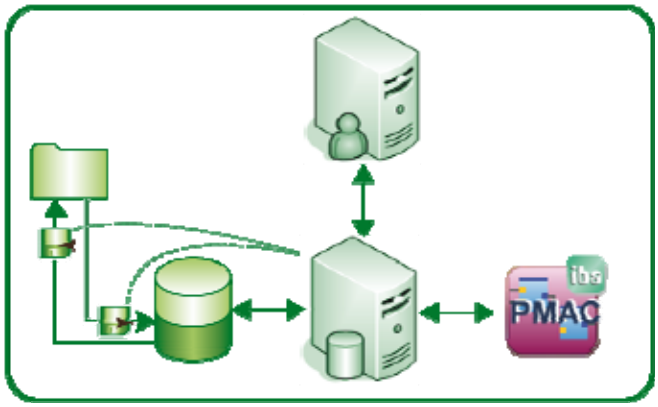


Рис. 7: Обзор функций сервера ibaLogic

База данных

Исполнительная система "PMAC"

Сервер ibaLogic

Файловая система

Клиент ibaLogic

Функция сохранения / восстановления

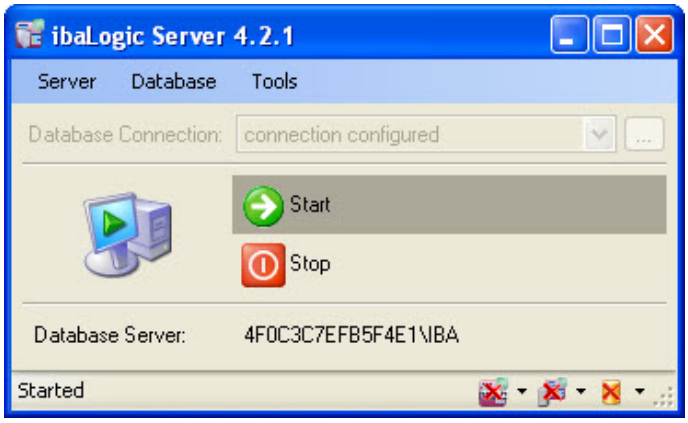
3.2 Запуск сервера ibaLogic

Необходимое условие

Ярлык сервера ibaLogic на рабочем столе.

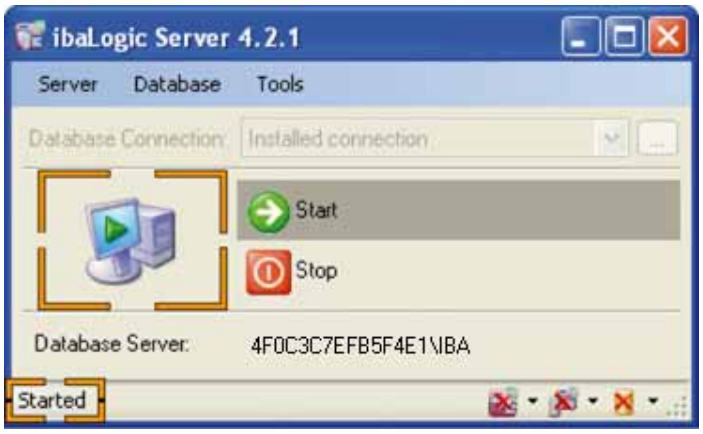
Последовательность действий

1. Запустите программу двойным щелчком по ярлыку "ibaLogic Sever" на рабочем столе.
   Появится диалоговое окно "Сервер ibaLogic".



   При открытии сервера ibaLogic автоматически отображается окно начального состояния. В в области уведомлений отображаются следующие значки:  .

2. Щелкните по кнопке <Запуск> (<Start>), чтобы запустить сервер ibaLogic. Запустить сервер можно также из меню "Запуск сервера".
Состояние "запуск / остановка" отображается в диалоговом окне сервера с помощью значка, сообщения и активной кнопки.



3.3 Пользовательский интерфейс – сервер ibaLogic

Сервер ibaLogic используется для:

- Конфигурирования сервера
- Выбора режима сервера: запуск / остановка
- Конфигурирования и создания резервной копии базы данных

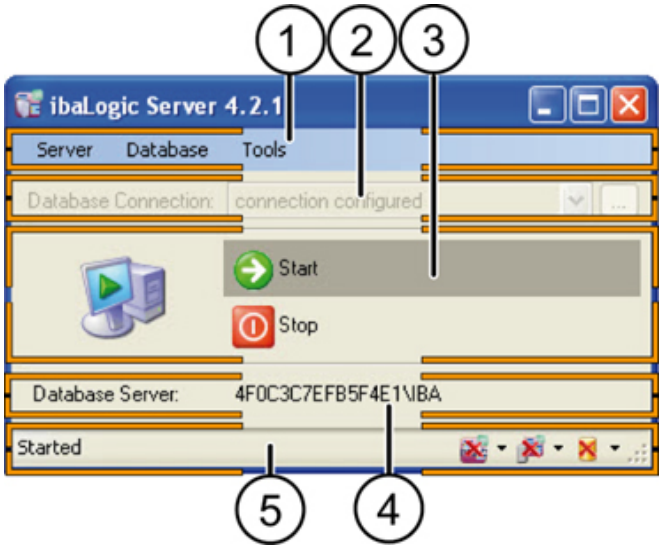


Рис. 8: Сервер ibaLogic – интерфейс пользователя

1	Панель меню	4	Текущее соединение с базой данных
2	Установление соединения с базой данных	5	Панель состояния
3	Кнопка запуска / остановки		

3.4 Настройка сервера ibaLogic

Необходимые условия

- ❑ Программа ibaLogic уже установлена.
- ❑ Диалоговое окно ibaLogic server открыто.

3.4.1 Конфигурация порта клиента

Номер порта клиента является параметром соединения для клиента ibaLogic.

Примечание

Пользователь должен изменить эту настройку только в том случае, если служебная программа, которая использует этот порт, запущена на серверном компьютере. Этот же порт необходимо указать в клиенте для связи с этим сервером. Новый номер порта вводится также при выборе соединения. В клиенте выберите меню "Файл - Соединение с сервером" ("File - Connect to server").

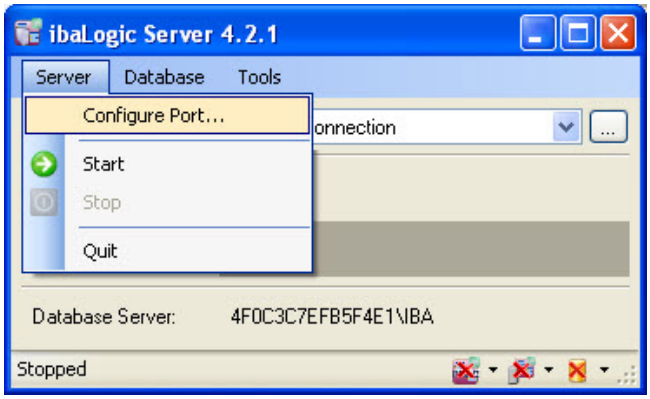
Номер порта по умолчанию: 8086.

Необходимые условия

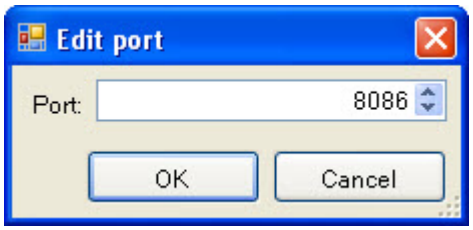
- ❑ Сервер ibaLogic остановлен.

Последовательность действий

1. Выберите пункт меню "Сервер - Конфигурировать порт".



2. Введите номер порта клиента вручную в поле ввода или воспользуйтесь элементом выбора.



3.4.2 Конфигурация соединений с базами данных

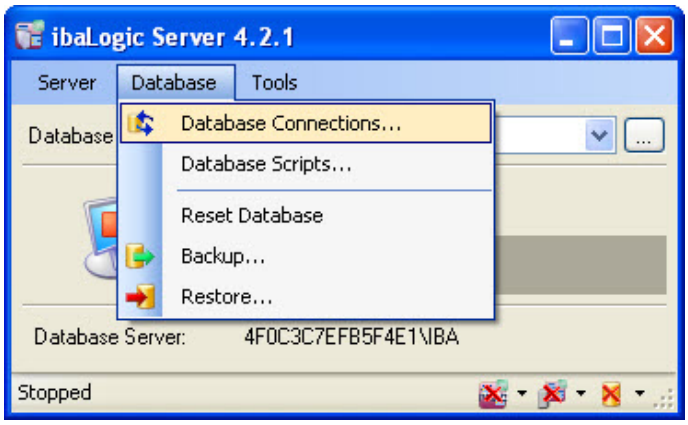
В ibaLogic можно настроить несколько соединений с базами данных, но при этом только одно из таких соединений будет активно в один период времени. Невозможно установить несколько соединений одновременно. При установке ibaLogic создается локальная база данных и значение этого параметра по умолчанию связывается с ней.

Важно

Нижеследующие действия необходимы только в случае, если вы хотите установить связь с базой данных, которая находится не на локальном ПК, и это не было указано в процессе установки программы.

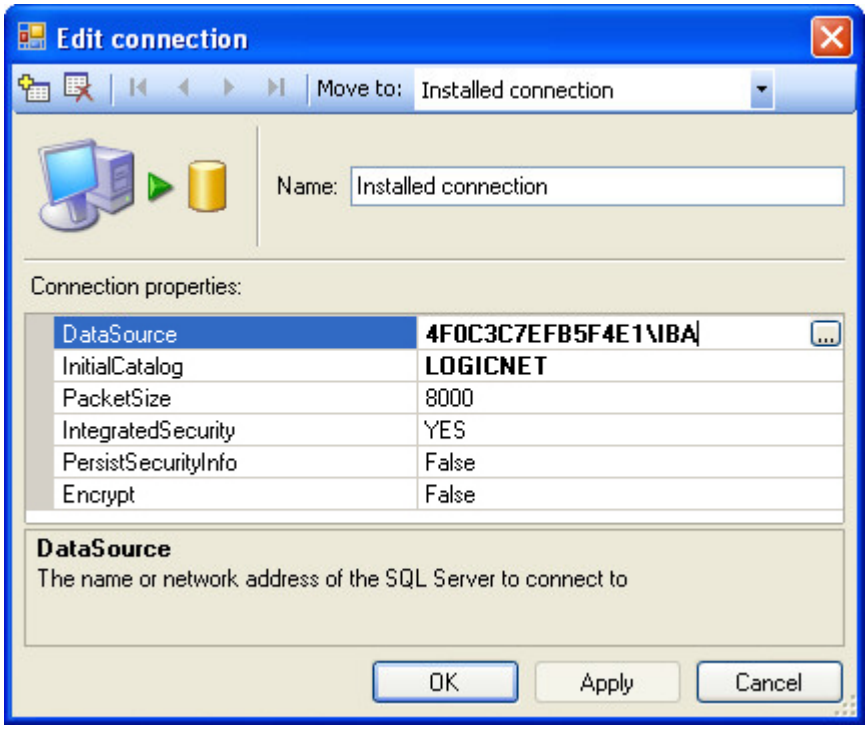
Последовательность действий

1. Выберите пункт меню "База данных - Соединения с базой данных".



2. В раскрывающемся списке "Имя" выберите соединение с базой данных iba-Logic.
   По умолчанию выбрано "Установленное соединение" - это соединение с локальной базой данных.

3. Щелкнув по кнопке просмотра: <…>, вызовите диалоговое окно для конфигурирования соединений с базой данных. Кнопка просмотра появляется только после щелчка по текстовому полю.



⚠ WARNING

Следующие параметры:

- Начальный каталог (InitialCatalog)
- Размер пакета (PacketSize)
- Встроенная защита (IntegratedSecurity)
- Информация о защите (PersistSecurityInfo)
- Шифровать (Encrypt)

требуют изменения только, если вы хотите, например, использовать центральную (общую) базу данных также для ibaLogic.

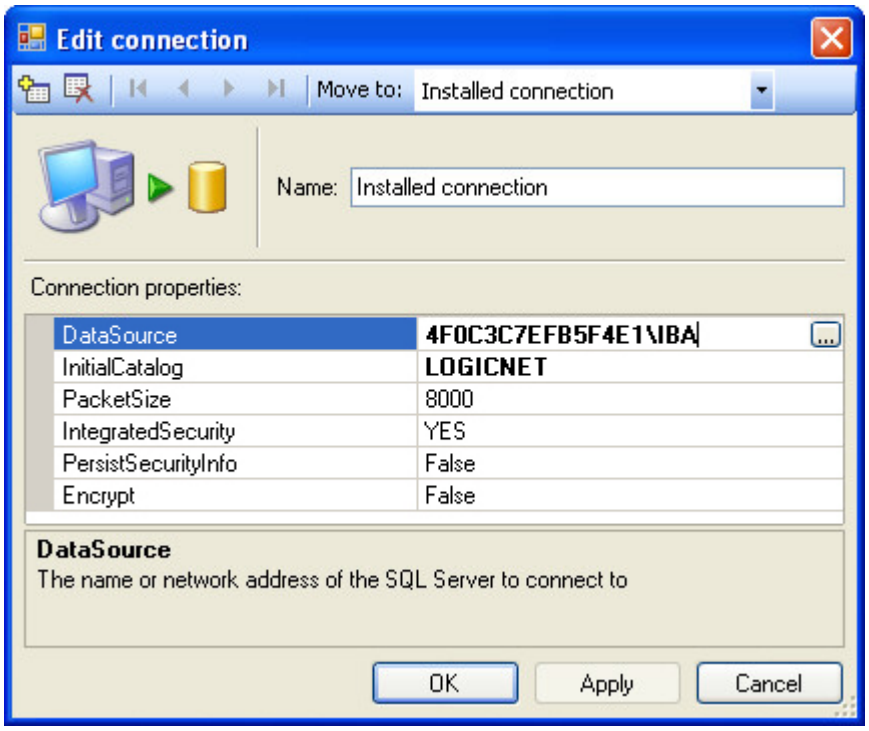
Однако, для настройки этих параметров необходимо знание баз данных. Если таковое отсутствует, обратитесь к администратору баз данных, который поможет вам правильно настроить эти специфические параметры.

3.4.2.1 Конфигурация интерфейса базы данных

В столбце напротив "Источник данных" ("DataSource") введите имя компьютера и экземпляр базы данных ibaLogic, с которой нужно установить соединение.

Последовательность действий

➲ Введите имя сервера непосредственно в текстовом поле или выберите базу данных с помощью кнопки просмотра.
Кнопка просмотра появляется только после щелчка по текстовому полю.



3.4.2.2 Выбор SQL-сервера

Диалоговое окно "Выбор SQL-сервера" ("Select SQL server") состоит из 2 вкладок: "Локальные серверы" ("Local servers") и "Сетевые серверы" ("Network servers").

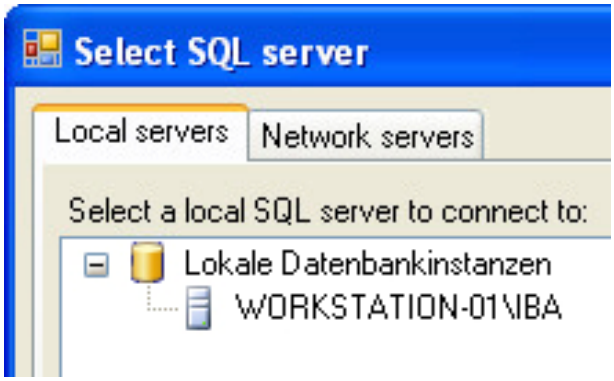


Рис. 9: Локальные серверы

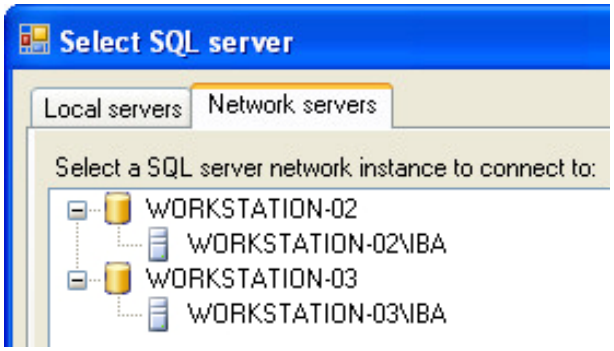


Рис. 10: Сетевые серверы

Все экземпляры базы данных SQL, доступные в компьютере, перечислены во вкладке "Экземпляры локальных баз данных" ("Local database instances").

После щелчка по вкладке "Сетевые серверы" программа выполнит поиск экземпляров SQL по всей сети. Этот процесс может занять некоторое время. Пока выполняется поиск, в текстовом поле пользователь видит сообщение "Считывание данных".

Необходимое условие

Программа ibaLogic server остановлена.

Последовательность действий

1. Щелкните по экземпляру SQL-сервера, к которому должен подключиться сервер ibaLogic.
2. Щелкните <OK>. Экземпляр SQL-сервера будет принят, и даиалоговое окно будет закрыто.

Результат

Экземпляр SQL-сервера соединен с сервером ibaLogic.

Примечания

Другие соединения с базами данных можно установить или удалить.

Для конфигурации соединения с базой данных можно использовать следующие значки:

Значки / блоки выбора	Значение	Объяснение
	Добавить	Добавить новое соединение с базой данных
	Удалить	Удалить соединение с базой данных
	В начало	К первому соединению
	Назад	К предыдущему соединению
	Далее	К следующему соединению
	В конец	К последнему соединению
	Выбор объекта	Опция выбора соединения

3.4.2.3 Управление сценариями базы данных

Важно

Список установленных сценариев базы данных приводится только в качестве дополнительной информации и необходим только при обращении в техподдержку iba. Не вносите в этот список никаких изменений.

Все сценарии, реализованные в ibaLogic, а также сопутствующая информация (номер версии, имя сценария и дата установки), отображаются в виде таблицы при вызове функции "Сценарии базы данных" ("Database scripts"). Для того чтобы вызвать диалоговое окно "Сценарии базы данных", нужно остановить сервер ibaLogic.

Как правило, сценарии базы данных проверяются при обновлении версии, которое выполняется автоматически.

3.4.2.4 Активация автозапуска сервера ibaLogic

ibaLogic server запускается автоматически при запуске Windows.

Пользователь может выбрать каталог, в котором будут сохранены опции автозапуска:

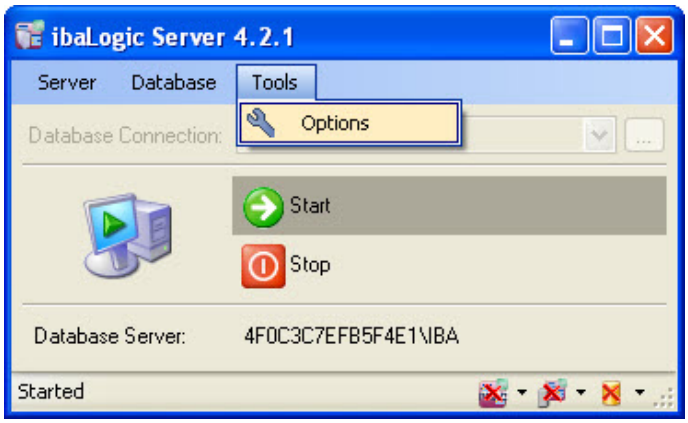
- В реестре
- В папке "Автозапуск" текущего пользователя
- В папке "Автозапуск" каталога "Все пользователи"

По умолчанию сервер запускается при открытии диалогового окна сервера ibaLogic. Если диалоговое окно должно быть открыто, но при этом сам сервер должен находится в режиме STOP, следует активировать опцию "Автозапуск при остановленном сервере" ("Autostart server stopped"). В этом случае нужно запустить сервер вручную, чтобы были приняты клиентские подключения.

Опции автозапуска	Объяснение
Файл реестра (рекомендуемая опция)	Опции автозапуска сохраняются в файле реестра.
Папка автозапуск (отдельный пользователь)	Опции автозапуска сохраняются в папке автозапуск для конкретного пользователя.
Папка автозапуск (все пользователи)	Опции автозапуска сохраняются в папке автозапуск для всех пользователей.
Автозапуск при выключенном сервере (не рекомендуется)	Диалоговое окно сервера ibaLogic будет запущено, а собственно сервер - нет. Сервер остается выключенным.

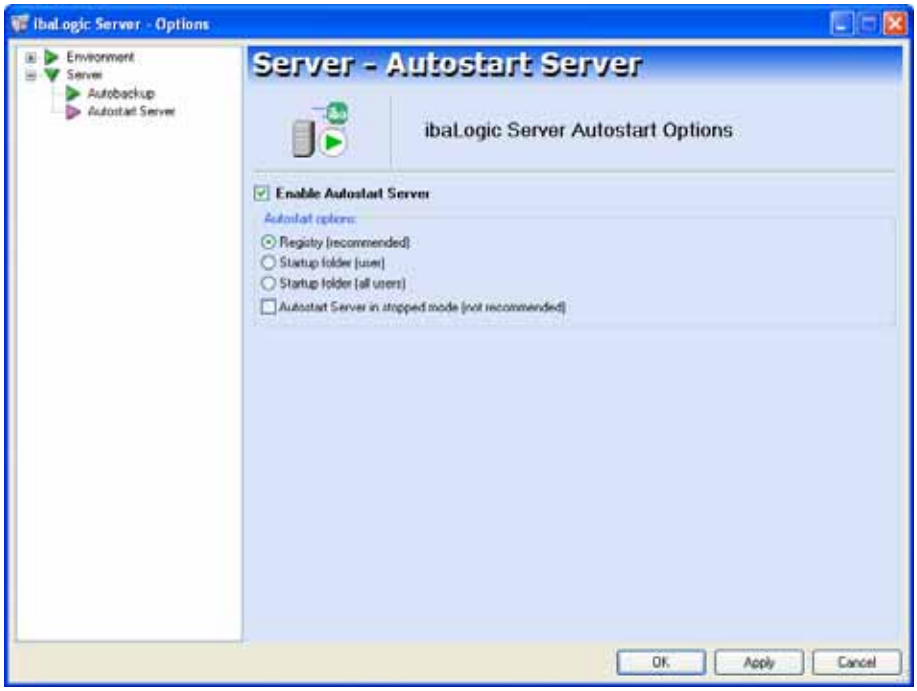
Последовательность действий

1. Выберите пункт меню "Инструменты – Опции" ("Extras – Options").



2. В дереве элементов выберите "Сервер – Автозапуск сервера" ("Server – Autostart Server").

3. Поставьте галочку напротив опции "Активировать автозапуск сервера" ("Enable Autostart Server").



4. Щелкните <Применить>, чтобы настройки были приняты системой.

Результат

Сервер ibaLogic будет запускаться автоматически при запуске Windows.

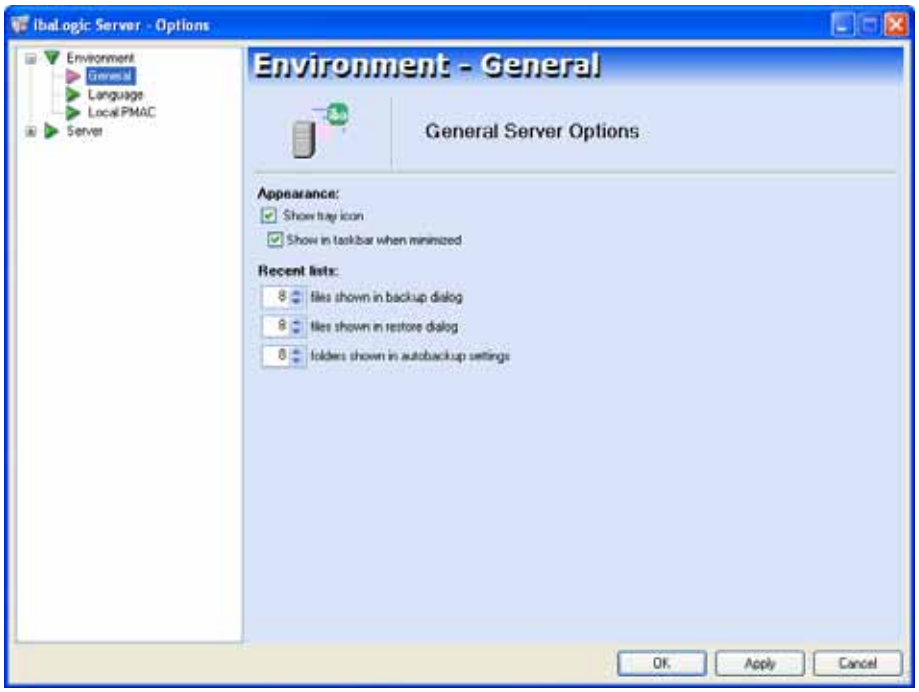
3.4.2.5 Конфигурация общих настроек сервера ibaLogic

Пользователь может настроить следующие общие настройки сервера ibaLogic:

Опция	Объяснение
Отображать значок в разделе Информация	Если выбрана эта опция, то при открытии диалогового окна сервера ibaLogic в разделе Информация появляется соответствующий значок.
Отображать программу в панели задач, если она свернута	Если выбрана эта опция, то значок сервера ibaLogic появляется в панели задач после сворачивания программы.
Отображать файлы в диалоге резервного копирования	Количество файлов, которые должны отображаться в диалоге резервного копирования. Это количество можно указать в опции выбора "Папка для резервных копий" ("Backup folder") под "Создать автоматическую копию сервера" ("Server Auto backup").
Отображать файлы в диалоге восстановления	Количество файлов, которые должны отображаться в диалоге восстановления.
Отображать файлы в диалоге настроек резервного копирования	Количество папок, которые должны отображаться в диалоге настроек резервного копирования.

Последовательность действий

1. Выберите пункт меню "Инструменты - Опции" ("Extras - Options").
2. В дереве элементов выберите "Системное окружение - Общее" ("Environment - General").



3. Вы можете изменить настройки, как вам нужно.
4. Щелкните <Применить>, чтобы настройки были приняты системой.

3.4.2.6 Настройки для локального контроллера (PMAC)

PMAC был реализован как служба. Конфигурация его состояния и типа запуска (опции автозапуска) выполняется здесь.

Конфигурация состояния:

- ❑ Активировать
- ❑ Деактивировать

Опция состояния	Описание
Активировать	Если выбрана эта опция, то, если необходимо, служба будет переустановлена автоматически.
Деактивировать	При выборе этой опции пользователь также должен решить, будет ли PMAC-служба при этом удалена. Удаление службы имеет смысл, если локальный контроллер не должен использоваться.

Конфигурация опций автозапуска (тип запуска):

Опции автозапуска	Объяснение
Нет автозапуска	PMAC-служба не будет запускаться автоматически.
Запуск PMAC-службы перед входом в Windows	PMAC-служба запускается перед входом пользователя в Windows. Дополнительная опция: Запустить проект при запуске службы, если возможно.
Запуск PMAC-службы одновременно с сервером ibaLogic	PMAC-служба запускается одновременно с сервером ibaLogic. Дополнительная опция: Запустить проект при запуске службы, если возможно.
Запустить проект при запуске службы, если возможно.	При запуске загружается образ проекта. Образ проекта должен быть создан заранее с помощью команды меню в клиенте. Если опция выбрана, а образ отсутствует, то программа ее игнорирует.

Последовательность действий

1. Выберите пункт меню "Инструменты - Опции" ("Extras -- Options").
2. В дереве элементов выберите "Системное окружение - Локальный PMAC" ("Environment – Local PMAC").

3. Выберите "Активировать". Если выбрана эта опция, то, если необходимо, служба будет переустановлена автоматически.
4. Выберите опцию перезапуска. Если активирована опция автозапуска, то пользователь может настроить его параметры.
   Дополнительно можно выбрать опцию запуска проекта.
   Для этого необходимо, чтобы проект был предварительно сохранен в PMAC.
5. Щелкните <Применить>, чтобы активировать настройки.

3.4.2.7 Язык

Здесь можно настроить язык диалогового окна сервера.

Язык диалогового окна клиента настраивается отдельно.

Последовательность действий

1. Выберите пункт меню "Инструменты - Опции" ("Extras -- Options").
2. В дереве элементов выберите "Системное окружение - Язык" ("Environment - Language").

3. Выберите "Языковые опции" ("Language options").
4. Поставьте галочку рядом с нужным вам языком.
5. Щелкните <Применить>, чтобы активировать настройки.

3.4.3 Панель состояния

В панели состояния диалогового окна сервера ibaLogic есть 3 значка, с помощью которых можно активировать/деактивировать настройки автозапуска и резервного копирования.

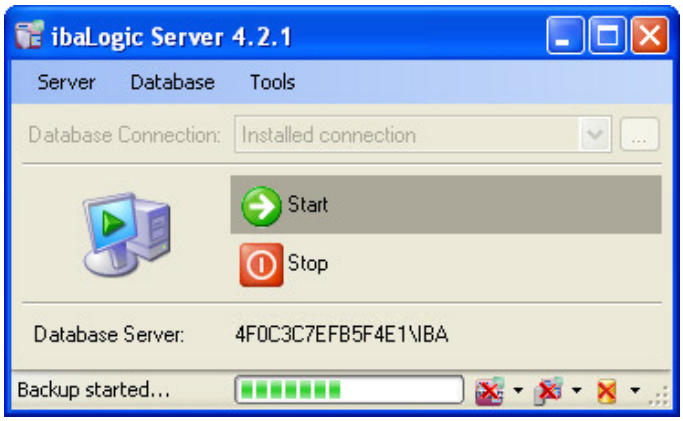


Рис. 11: Деактивированные функции в панели состояния

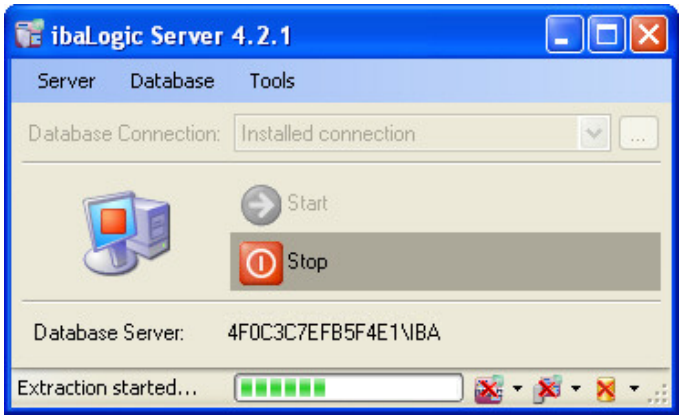


Рис. 12: Активированные функции в панели состояния

Значок	Настройка	Описание
	Автозапуск (запуск перед входом в систему)	PMAC будет запускаться автоматически при запуске Windows.
	Автозапуск (реестр)	Опции автозапуска будут сохранены в файле реестра.
	Автоматическое создание резервной копии	В соответствии с настройками резервного копирования будет создаваться резервная копия базы данных.

4 Среда разработки ПО - клиент ibaLogic

Клиент ibaLogic используется для создания и редактирования программ.

4.1 Запуск клиента ibaLogic

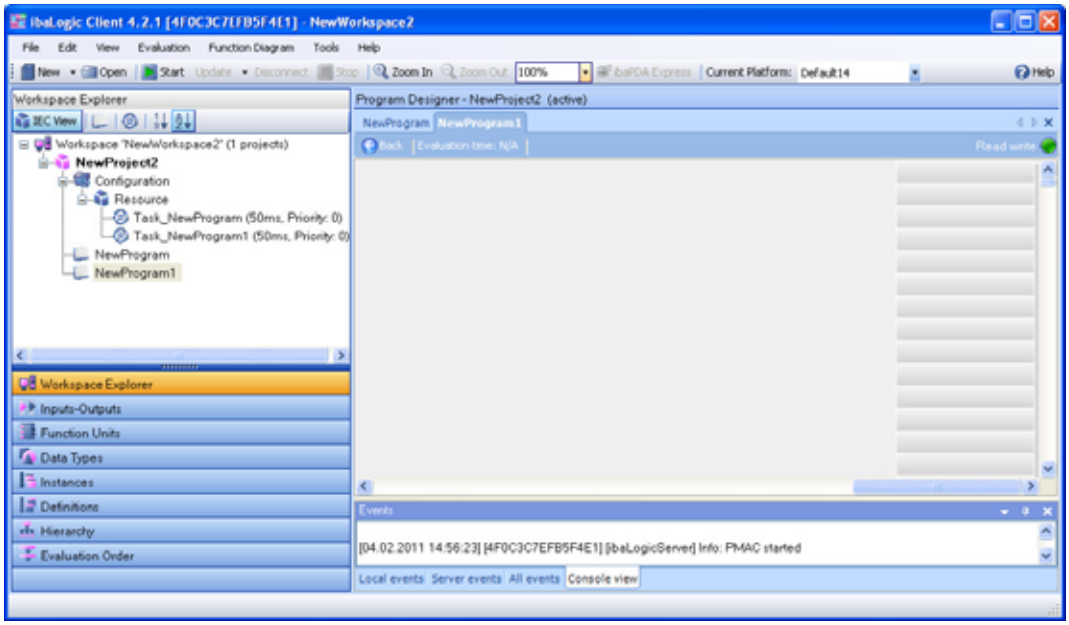
Отображается среда разработки ПО.

Необходимое условие

- ❑ На рабочем столе созданы ярлыки для запуска клиента и сервера ibaLogic (стандартная установка).
- ❑ Сервер ibaLogic запущен.

Последовательность действий

➲ Запустите программу двойным щелчком по ярлыку "ibaLogic Client" на рабочем столе.
После короткого этапа инициализации откроется диалоговое окно клиента ibaLogic.



Примечания

Под основным окном программы находится окно событий, в котором создается список действий и ошибок (если таковые возникали).

4.2 Пользовательский интерфейс среды разработки ПО - Редактор

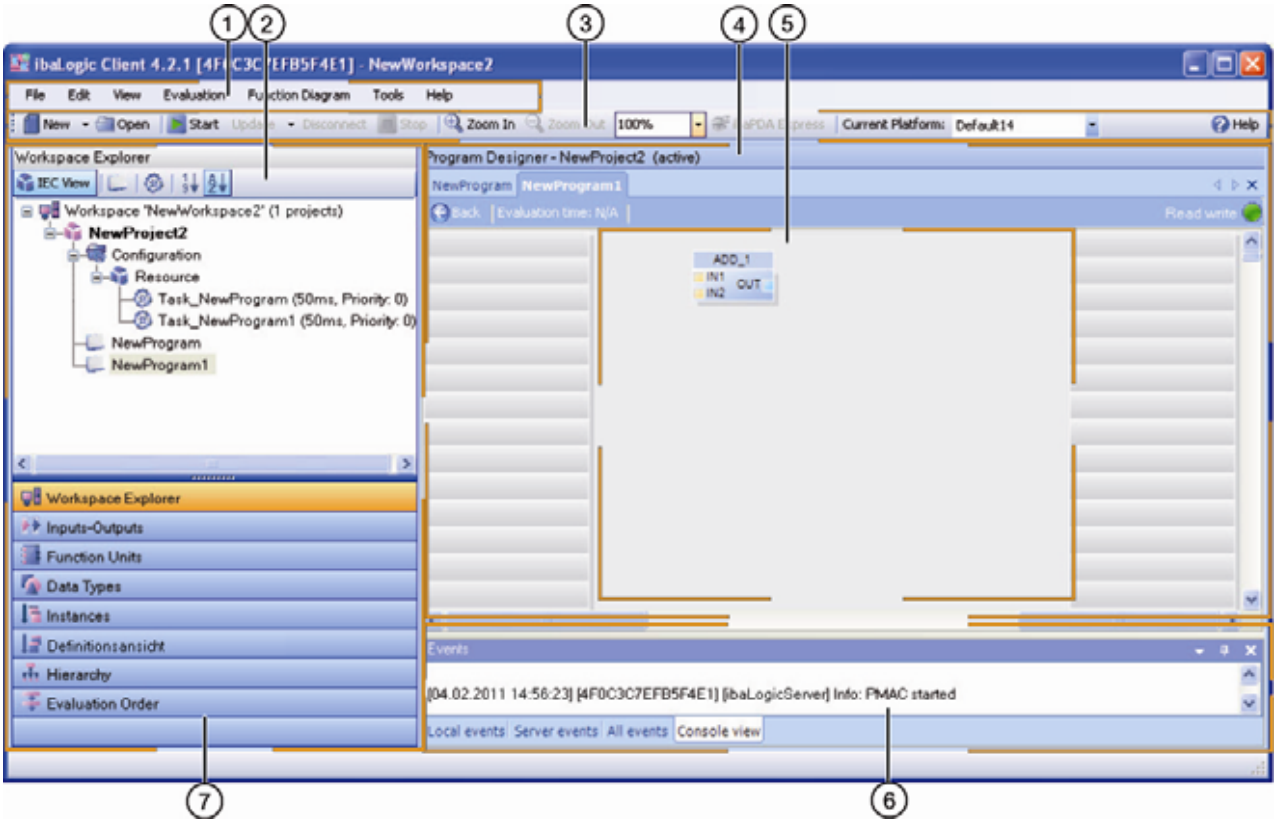


Рис. 13: Пользовательский интерфейс

1	Панель меню	5	Область программы
2	Область навигации	6	Окно событий
3	Панель инструментов	7	Элементы навигации
4	Область программирования		

4.2.1 Панель меню

Панель меню - это центральный элемент управления в клиенте ibaLogic.

Рис. 14: Панель меню

4.2.2 Панель инструментов

Панель инструментов - это второй по важности элемент управления в клиенте ibaLogic.

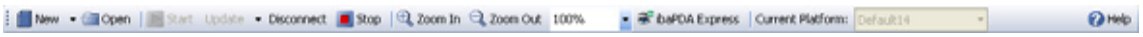


Рис. 15: Панель инструментов

Значки на панели инструментов одновременно являются кнопками с соответствующими функциями.

4.2.3 Область навигации

В области навигации находятся следующие кнопки:

- Проводник по рабочим областям
- Входы - выходы
- Функциональные блоки
- Типы данных
- Экземпляры
- Определения
- Иерархия
- Порядок вычисления

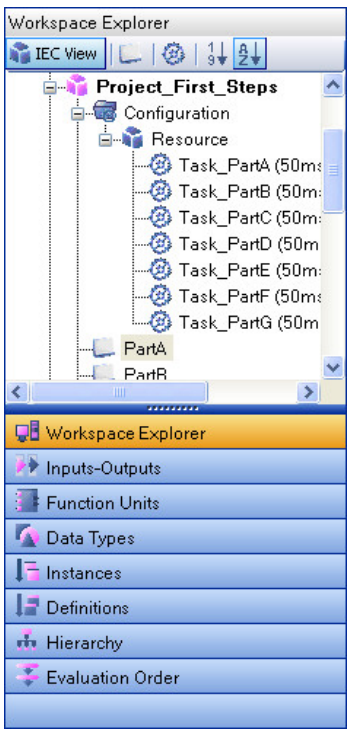


Рис. 16: Элементы навигации

4.2.3.1 Переключение видов в проводнике по рабочим областям

В панели меню проводника по рабочим областям есть три кнопки, которые позволяют переключать виды проводника. Это кнопки:

- IEC-вид
- Вид программ
- Вид задач

Во всех трех видах пользователь может расположить элементы удобным ему способом: в порядке приоритетности или в алфавитном порядке.

IEC-вид

Этот вид является стандартным, в нем отображается структура рабочей области ibaLogic (в соответствии со стандартом IEC 1131-3).

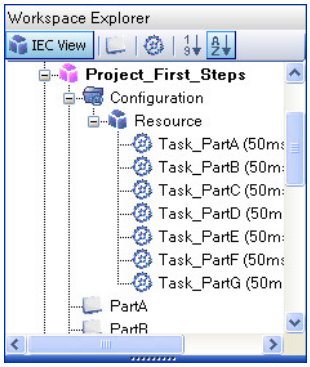


Рис. 17: IEC-вид

Примечание

В отличие от стандарта IEC 1131-3, каждой задаче присваивается только одна программа.

Вид программ

Компактное отображение проектов и программ.

Рис. 18: Вид программ

Вид задач

Отображение организовано в виде последовательности задач.

Рис. 19: Вид задач

Пользователь может выбрать между отображением задач в алфавитном порядке или в порядке приоритетности.

Значок	Объяснение
	Элементы сортируются по имени.
	Элементы сортируются по приоритетности.

4.2.3.2 Экземпляры

Все блоки, используемые в проекте, отображаются в виде списка в соответствии с именами экземпляров. Имя определения также отображается, разделителем служит двоеточие.

При двойном щелчке по имени экземпляра отображается страница программы, на которой находится функциональный блок. Имя блока будет выделено.

Примечание касательно разницы между определением и экземпляром

Определение блока сохраняется в глобальной библиотеке. В программе всегда находится экземпляр (фактически - копия) блока. Имя экземпляра автоматически создается по следующей модели: "Имя определения*Индекс*" ("Definition name*Index*"). Однако, пользователь может изменить эту модель по своему усмотрению.

При внесении изменений в содержимое блока, изменения также вносятся в определение и другие экземпляры.

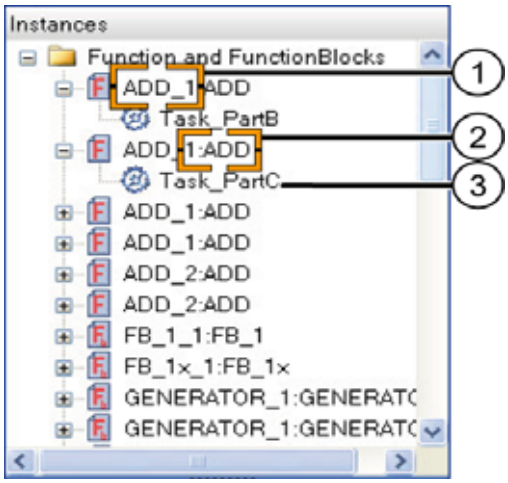


Рис. 20: Экземпляры

1 Имя экземпляра

2 Тип определения

3 Задача, которая содержит этот блок

Если блок находится во вложенном макроопределении, то он отображается в дереве элементов:

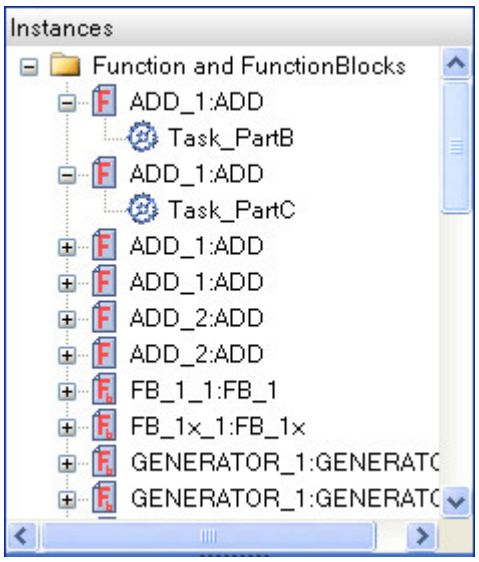


Рис. 21: Отображение экземпляров в виде дерева элементов

4.2.3.3 Определения

В этом виде отображаются все блоки, относящиеся к проекту, в порядке имен их определений. В дереве элементов отображаются экземпляры, которые располагаются под типом блока, ниже экземпляров - макросы (если есть), а затем - задача и имя программы.

При двойном щелчке по имени экземпляра отображается страница программы, на которой находится функциональный блок. Имя блока будет выделено.

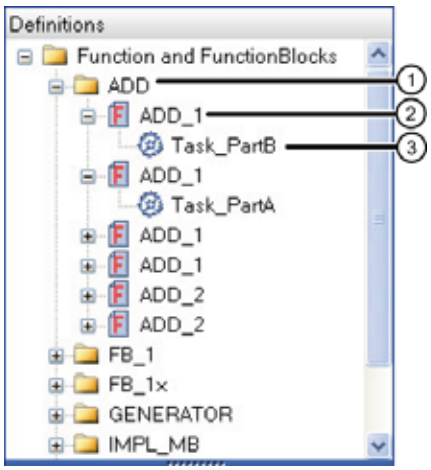


Рис. 22: Определения

1 Имя определения

2 Имя экземпляра

3 Имя задачи

4.2.3.4 Иерархия

В иерархическом виде отображаются все экземпляры блоков, перечисленные в алфавитном порядке по задачам.

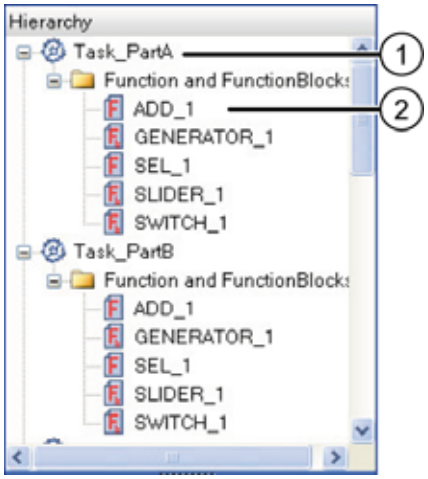


Рис. 23: Иерархический вид

1 Задача 2 Имя экземпляра

4.2.3.5 Порядок вычисления

В виде "Порядок вычисления" отображается последовательность, в которой выполняется вычисление программ и блоков внутри программ.

Блоки, которые вычисляются в первую очередь, располагаются в верхней части дерева.

Пример

2 задачи с одинаковым интервалом, но разной приоритетностью.

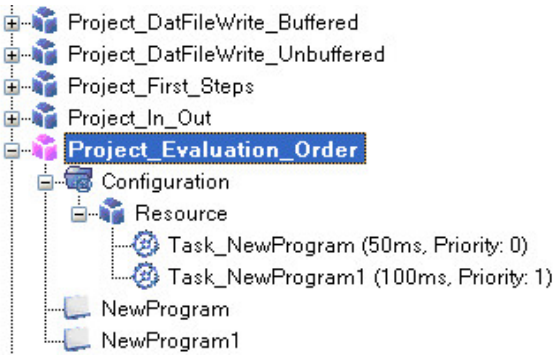


Рис. 24: 2 задачи с одинаковым интервалом, но разной приоритетностью

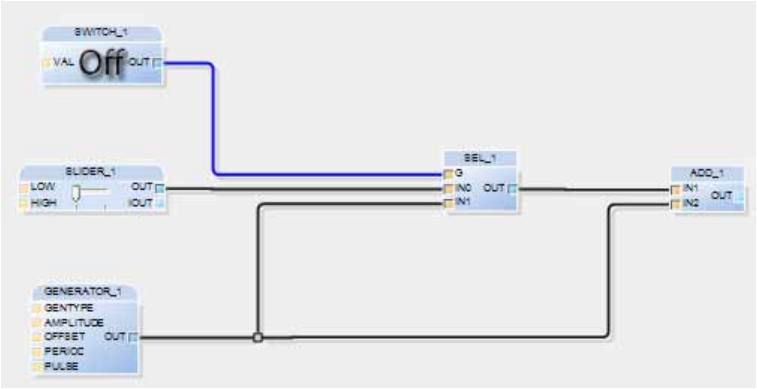


Рис. 25: Новая программа со следующей структурой

Порядок вычисления отображается следующим образом:

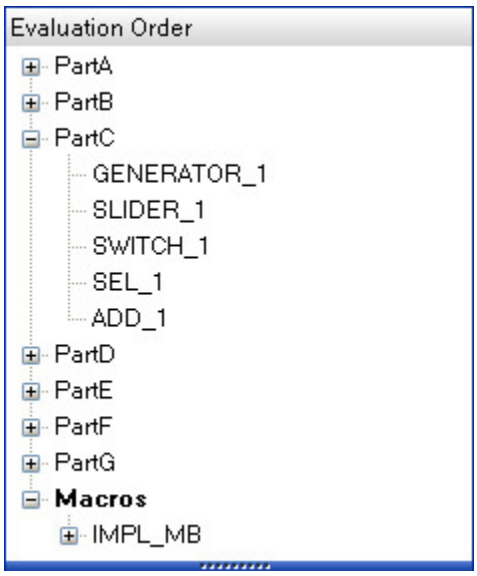


Рис. 26: Порядок вычисления

Правила порядка вычисления

- Программы исполняются в соответствии с интервалом относящихся к ним задач.
- Задачи, которые должны запускаться одновременно, вычисляются в соответствии с их приоритетностью (0 соответствует самому высокому приоритету).
- Выполнение задач не прерывается. Это значит, что даже задача с более высоким приоритетом не может прервать выполнение задачи с более низким приоритетом.
- Вычисление в рамках программы осуществляется в соответствии с нижеследующими правилами:
  - Первыми всегда вычисляются те блоки, которые генерируют данные. После этого вычисляются блоки, которые используют данные.
  - В программе есть множество независимых ветвей, поэтому используется дополнительное правило, согласно которому вычисления производятся по направлению от верхнего левого угла - к нижнему правому. Это означает, что ветвь программы, расположенная выше, всегда вычисляется в первую очередь.
  - В программе или макросе с обратной связью в первую очередь вычисляется блок, расположенный в самом верху слева.

⚠ DANGER

Почему опасно вносить изменения в порядок вычисления

При перемещении блоков (только в программах с обратной связью) может измениться порядок вычисления и, соответственно, результаты.

Совет

Для того чтобы этого избежать, рекомендуется встраивать блоки в макросы и использовать функцию обратной связи за рамками макроса. В этом случае определение порядка вычисления не будет зависеть от расположения.

Макрос - это разновидность блока. Блоки, которые включены в макрос, вычисляются согласно вышеописанным правилам. Вычисленное выходное значение становится новым входным значением только в следующем цикле.

Совет

Содержимое функционального блока вычисляется за один цикл.
Это означает, например, что при доступе к входному коннектору внутри цикла For будет получено одно и то же значение при каждой итерации цикла, поскольку коннектор пересчитывается только в следующем цикле. Если определенный внутренний цикл должен выполняться в течение нескольких циклов программы, то нужно использовать управляющую переменную, которая является внешней для блока.

4.2.4 Область программирования (Program Designer)

Область программирования используется для размещения и соединения функциональных блоков.

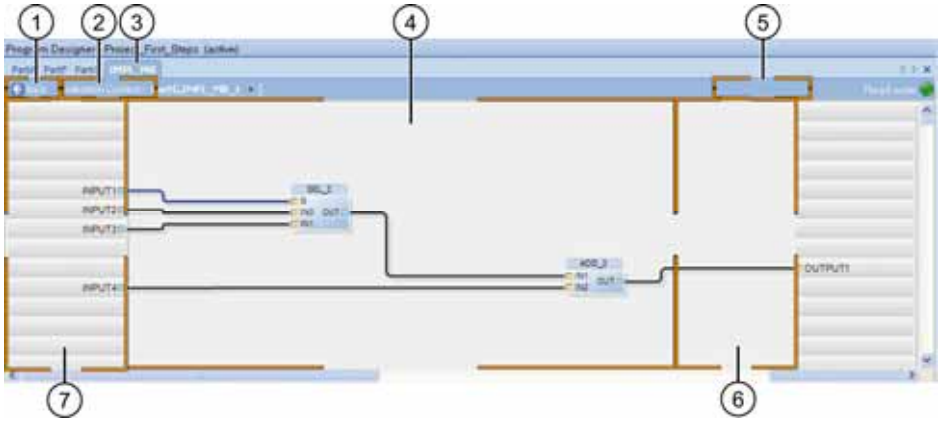


Рис. 27: Пользовательский интерфейс области программирования

1	Кнопка <Назад> (<Back>)	5	Кнопка <Разблокировать / Заблокировать> (<Release / Block>)
2	Контекст вычислений (Evaluation context)	6	Область вывода (Output area)
3	Вкладка (Tab)	7	Область ввода (Input area)
4	Окно программирования (Programming window)		

4.2.5 Расположение вкладок и окон программирования

Пользователь может сам определить нужное расположение с помощью мыши.

Доступны следующие опции:

- ❑ Перекрывание (Overlapping)
- ❑ Рядом / один под другим (Beside / Below one another)

4.2.5.1 Расположение вкладок

Для того чтобы расположить вкладки, действуйте следующим образом:

Последовательность действий

1. Щелкните по вкладке соответствующей программы.
2. Удерживая кнопку мыши, перетащите вкладку на новое место.

Рис. 28: Расположение в виде вкладок

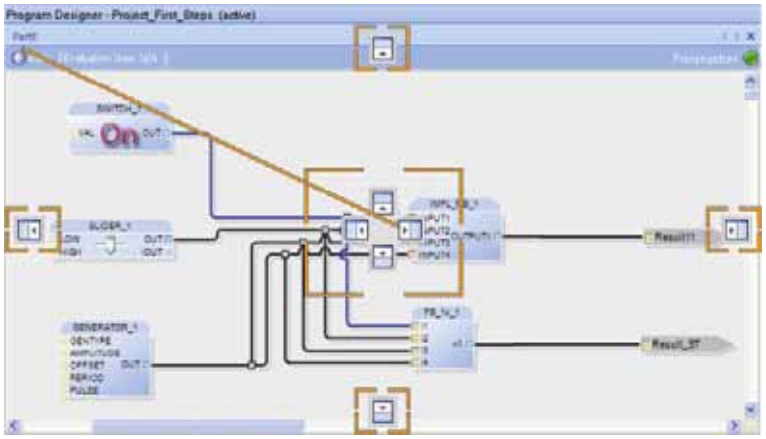
4.2.5.2 Расположение окон программирования

Стыкуемые окна (docking windows) - это окна, которые могут отстыковываться и пристыковываться к специальным областям стыковки, эти окна можно перемещать в любую область экрана.

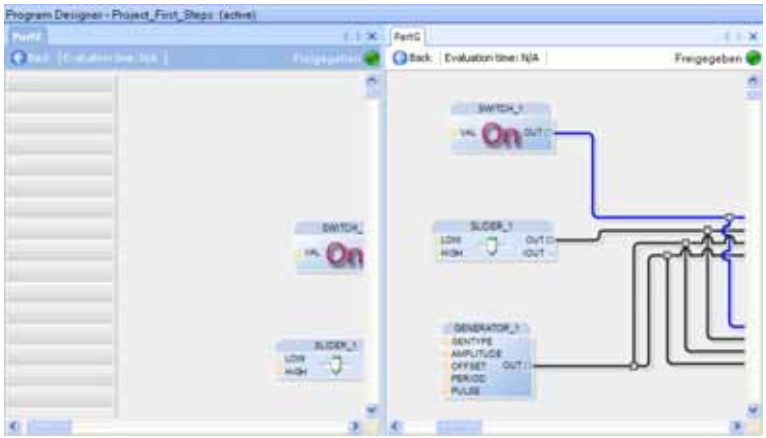
Для того чтобы расположить окна, действуйте следующим образом:

Последовательность действий

1. Установите курсор мыши на вкладку окна, которое вы хотите переместить.
2. Нажмите и удерживайте левую кнопку мыши.
3. Передвиньте окно к области стыковки, в которой вы хотите расположить вкладку.



4. Отпустите кнопку мыши. Теперь окно закреплено в выбранном положении.



Примечание

Пользователь не может расположить окна программ по своему желанию.

Открытые программы и макросы графически отображаются в области программы. Поскольку макросы отображаются абсолютно так же, как программы, их описание в нижеследующих разделах не приводится.

Панель инструментов

В панели инструментов пользователь может использовать следующие функции:

- Назад (

)
- Контекст вычислений (только для макросов)
- Время вычисления (только в онлайн-режиме)

Назад

Эта кнопка используется для навигации в окне программы. Отображается последняя выбранная программа / макрос.

Контекст вычислений (только для макросов)

Используется для отображения программы или макроса, в котором располагается текущий план. Поскольку несколько экземпляров одного макроса могут присутствовать в одной или нескольких программах с различными параметрами вызова, пользователь может посмотреть программу и экземпляр, в которых содержится макрос, открытый в настоящее время.

Если один макрос находится внутри другого (вложение), то отображается вся ветвь иерархии, а в качестве разделителей используются точки:

Отображение: Имя_программы.Макрос_1.Макрос_2….

(Program_name.Macro_1.Macro_2….)

Пользователь может выбрать контекст в рамках открытого макроса, т.е. переключаться между экземплярами. Значения, отображаемые в онлайн-виде, всегда относятся к выбранному экземпляру.

1. Для этого щелкните на кнопку со стрелкой в правой части окна рядом с именем экземпляра макроса.

2. В макросе можно выбрать уровень вызова. Для этого щелкните имя экземпляра непосредственно в контексте вычисления.

В области программы появляется окно, в котором содержится данный экземпляр макроса и выделяется сам макрос.

Время вычисления (только в онлайн-режиме)

Это время отображается в каждой программе как процентное значение в отношении интервала задачи и как абсолютное значение в мс (округленное среднее значение).

4.2.5.3 Навигация в области программирования

Для навигации в рамках программы существует несколько опций:

- Масштабирование (Zoom)
- Обзор программы (Program overview)

Приближение / отдаление (Zoom)

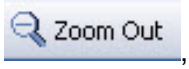
Существует опция для приближения (zoom in) и отдаления (zoom out) области программы в области программирования.

Приближение / отдаление регулируется следующим образом:

- Кнопками <Zoom in> / <Zoom out> (приблизить / отдалить)
- Выбором элементов в раскрывающемся списке
- Нажатием сочетания клавиш

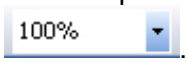
Кнопки <Zoom in> / <Zoom out> (приблизить / отдалить)

1. В панели инструментов клиента ibaLogic есть две кнопки: 

и 

, с помощью которых можно приблизить или отдалить содержимое области программы в области программирования на 20 % при каждом нажатии.

Раскрывающийся список

2. В раскрывающемся списке выберите одно из 5 значений для изменения масштаба изображения 

.

или

3. В окне значений раскрывающегося списка введите коэффициент масштабирования от 25 до 2 000 процентов.

Сочетания клавиш

4. Нажав и удерживая клавишу <Ctrl>, вращайте колесико мыши.

Минимальный коэффициент масштабирования зависит от разрешения экрана.

Обзор программы

Поскольку страница программы, как правило, выходит за пределы экрана компьютера и (в зависимости от коэффициента масштабирования) видна только ее часть, обзор программы помогает пользователю ориентироваться в программе.

Как использовать обзор программы

Последовательность действий

1. В основном меню выберите "Обзор программы" ("Program Overview").

   На переднем плане текущего окна программы появится маленькое окно "Обзора программы".

   Видимая часть программы будет отображена как прозрачный четырехугольник.

2. Передвиньте этот четырехугольник с помощью мыши на нужный участок программы.

Результат

В области программы отобразиться часть программы, выбранная вами в обзоре.

Примечания

Вы можете расположить окно обзора в любом месте области программы.

Помимо этого, обзор программы можно пристыковать к области программы.

Рис. 29: Область программы с расположенным поверх нее окном обзора

Навигация в области программы

Для навигации в окне программы в основном используется мышь.

Ниже приведены основные опции:

- ❑ Колесо прокрутки: Перемещение видимого участка вверх / вниз
- ❑ Колесо прокрутки + <Shift>: Перемещение видимого участка влево / вправо
- ❑ Колесо прокрутки + <Ctrl>: Приближение / отдаление

Увеличение длины страницы

Размер страницы по умолчанию: 2500 пикселей в ширину и 10 000 - в длину. Видимый участок страницы зависит от разрешения экрана.

Пользователь может увеличить длину страницы.

Последовательность действий

1. Откройте контекстное меню, щелкнув правой кнопкой мыши по свободному пространству в области программы.
2. Выберите "Увеличить длину страницы" ("Extend page (vertical)").

Результат

Длина страницы будет увеличена на 800 линий пикселей, которые будут добавлены, начиная с того места, где установлен курсор мыши.

Примечания

Важно

Убедитесь, что на одной линии с курсором нет блоков. Соединительные линии, которые пересекают горизонтальную линию, обозначающую продление страницы, не будут удлинены.

Примечание

Удалить пустую страницу в задаче невозможно. Пустое пространство в нижней части задачи будет автоматически удалено при загрузке проекта.

Совет

Пустое пространство в задаче можно создать, значительно уменьшив отображение и сократив расстояние между объектами.

4.2.6 Синхронизация доступа (кнопки <Чтение / запись> и <Только чтение> (<Read write> / <Read only>))

В режиме работы с несколькими клиентами доступ к окнам и диалоговым окнам синхронизируется с помощью кнопок <Чтение / запись> и <Только чтение>.

Рис. 30: Кнопка <Чтение / запись>

Рис. 31: Кнопка <Только чтение>

4.2.7 Окно событий

Под "Областью программирования" находится окно событий, в котором создается список действий и ошибок (если таковые возникали).

Окно событий содержит 4 вида, переключаться между которыми можно с помощью вкладок.

Виды:

- Локальные события
- Серверные события
- Все события
- Консольный вид

В видах "Локальные события", "Серверные события" и "Все события" используются следующие значки:

Значок события	Событие	Уровень	Описание
	Информация	Информация	Отображение информации
	Предупреждение	Предупреждение	Отображение предупреждения
	Ошибка	Исключение	Отображение ошибки

4.2.7.1 Локальные события

Пользователь может использовать вид "Локальные события" для отображения событий клиента в заданном формате.

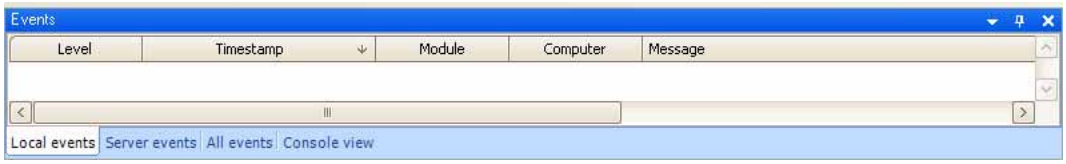


Рис. 32: Окно событий – "Локальные события"

4.2.7.2 Серверные события

Вид "Серверные события" используется для отображения событий, связанных с сервером.

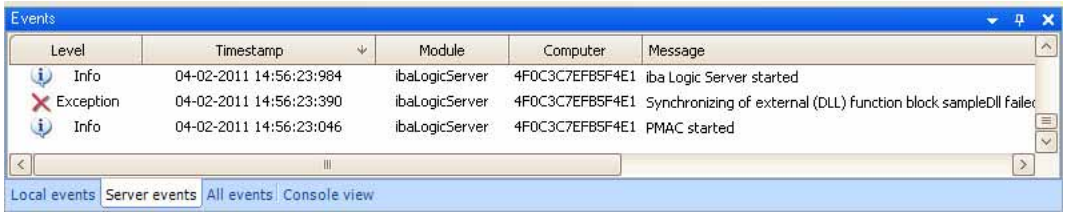


Рис. 33: Окно событий - "Серверные события"

4.2.7.3 Все события

Пользователь может использовать вид "Все события" для отображения событий клиента в заданном формате.

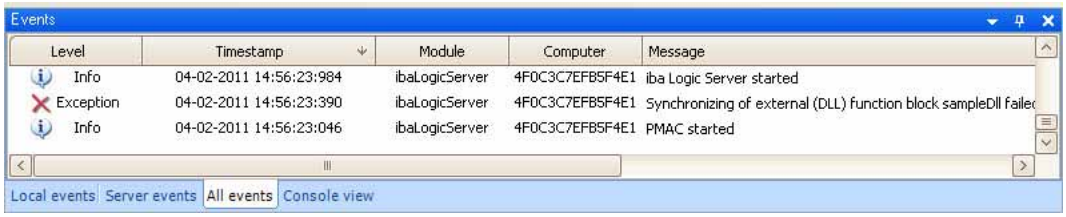


Рис. 34: Окно событий - "Все события"

4.2.7.4 Консольный вид

"Консольный вид" используется для отображения всех событий, перечисленных в хронологическом порядке с соответствующими пояснениями.

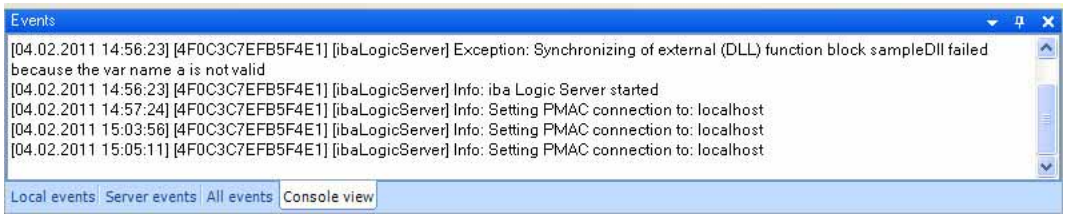


Рис. 35: Окно событий - "Консольный вид"

4.3 Рабочая область

Рабочая область используется для упорядоченного сохранения программ и проектов.

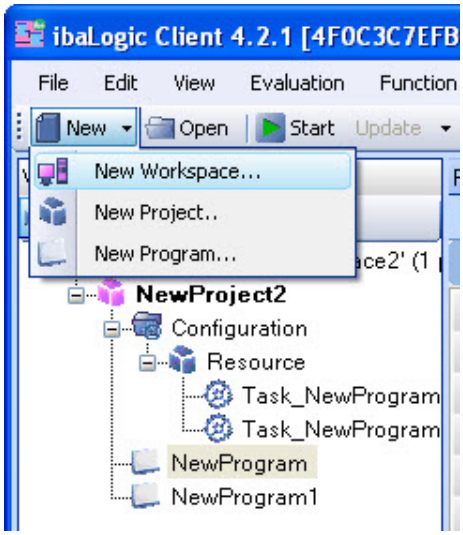
4.3.1 Создание рабочей области

Необходимые условия

- ❑ Выбран проводник по рабочим областям.
- ❑ Все рабочие области закрыты.
  Закрыть уже открытые рабочие области можно с помощью "Закрыть файл рабочей области" ("Close workspace file").

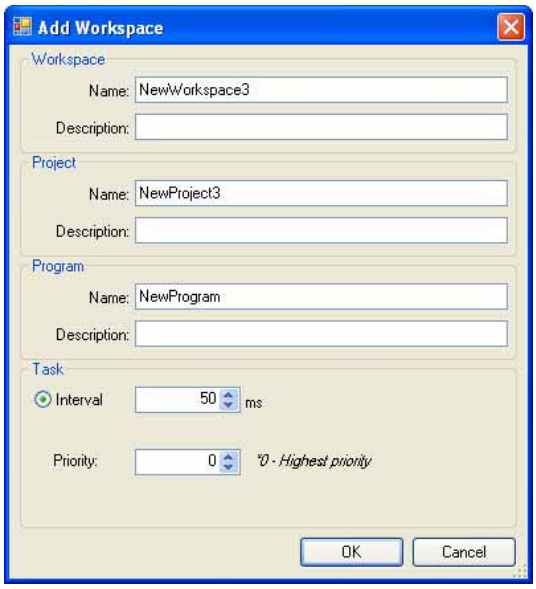
Последовательность действий

1. Щелкните стрелку на значке <Новый> (<New>) в панели инструментов.
2. В раскрывающемся списке выберите "Новая рабочая область" ("New Workspace").



Появится диалоговое окно "Добавить рабочую область" ("Add Workspace").

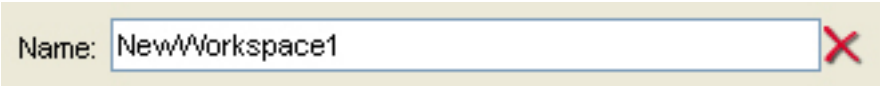
3. Присвойте имя рабочей области и создайте ее описание.
ibaLogic автоматически создаст новый проект и задачу. Имя и описание рабочей области должны соответствовать требованиям стандарта IEC. Если нужно, укажите также интервал и приоритет задачи.



Примечания

Все настройки можно впоследствии изменить с помощью контекстного меню "Свойства" ("Properties").

Имя, которое уже было присвоено, отображается в поле ввода с символом "X" в конце.



Поля для описания пользователь может заполнять любыми комментариями. Эти комментарии отображаются в виде подсказки, которая появляется над объектом.

Примечание

Присваиваемые пользователем имена должны соответствовать требованиям стандарта IEC. Более подробная информация содержится в приложении B "Соглашение о присвоении имен".

Совет

Задать имя, которое будет появляться по умолчанию, можно следующим образом: "Инструменты – Опции – Редакторы – Рабочие области" ("Extras – Options – Editors – Workspaces").

4.3.2 Открыть рабочую область

Последовательность действий

1. Щелкните кнопку <Открыть> (<Open>). Появится окно "Открыть рабочую область" ("Open Workspace").
2. Выберите требуемую рабочую область и щелкните <OK>.

4.3.3 Закрыть рабочую область

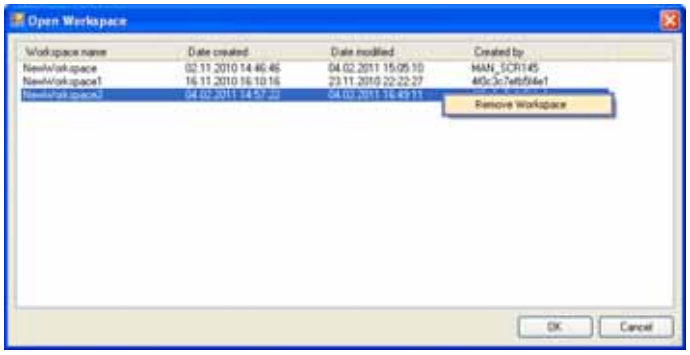
Последовательность действий

➲ Выберите рабочую область в меню "Файл – Закрыть рабочую область" ("File – Close Workspace").

4.3.4 Удалить рабочую область из базы данных

Последовательность действий

1. Щелкните кнопку <Открыть> (<Open>). Появится окно "Открыть рабочую область" ("Open Workspace").
2. Выделите рабочую область, которую нужно удалить.
3. Щелкните по выделенной рабочей области правой кнопкой мыши, и появится меню "Удалить рабочую область" ("Remove Workspace").



4. Щелкните "Удалить рабочую область" ("Remove Workspace"). Появится диалоговое окно "Удалить рабочую область.
5. Подтвердите удаление, нажав <Да>.
6. Когда рабочая область будет удалена, щелкните <OK>. Диалоговое окно будет закрыто.

4.4 Проекты в рабочей области

Когда пользователь создает новую рабочую область, ibaLogic автоматически создает в ней проект. В одной рабочей области может находиться несколько проектов.

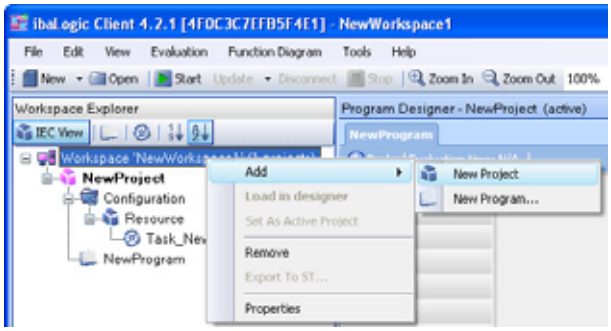
Необходимые условия

- ❑ Сервер ibaLogic установлен и запущен.
- ❑ Клиент ibaLogic установлен и запущен.
- ❑ Создана как минимум одна рабочая область.

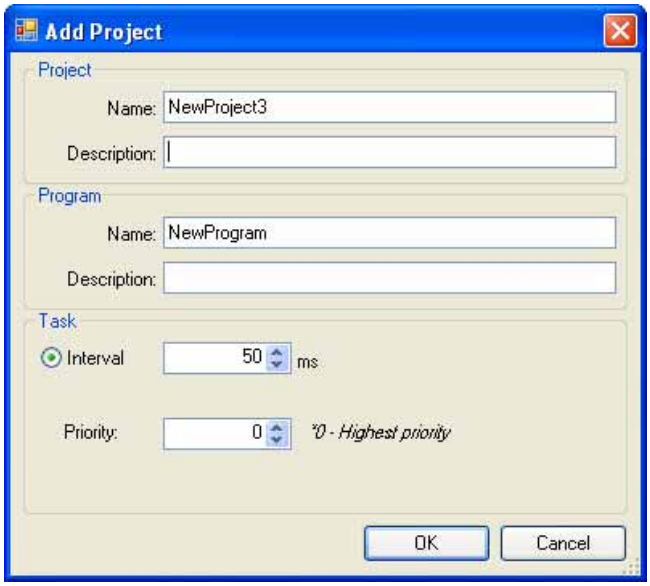
4.4.1 Создание проекта

Последовательность действий

1. В проводнике по рабочим областям щелкните правой кнопкой мыши по имени рабочей области, к которой нужно добавить проект.
2. В контекстном меню выберите "Добавить новый проект" ("Add New Project").



Появится диалоговое окно "Добавить проект" ("Add Project").
Присвойте имя проекту и программе. Присваиваемые имена должны соответствовать требованиям стандарта IEC. Если нужно, укажите также интервал и приоритет задачи.



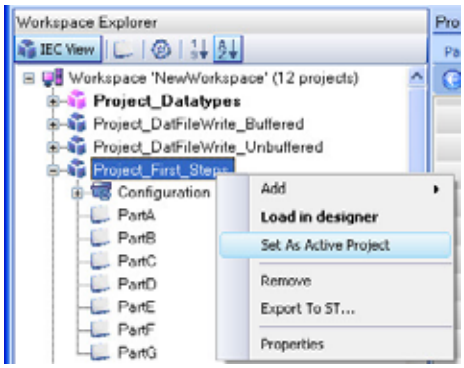
4.4.2 Назначить проект активным

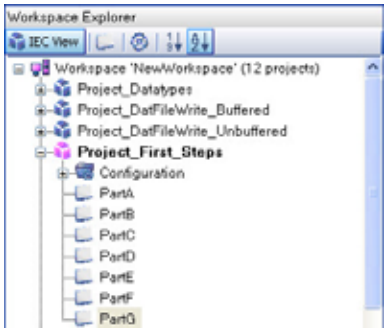
Примечание

В рамках соответствующей рабочей области может быть активен только один проект.

Последовательность действий

1. Щелкните правой кнопкой мыши по проекту, который вы хотите назначить активным.
2. В контекстном меню выберите "Назначить проект активным" ("Set As Active Project")





Результат

Имя активного проекта отображается в проводнике по рабочим областям **жирным шрифтом** и соответствующий значок меняет цвет с синего на розовый.

Примечание

Если проект находится в онлайн-режиме (фон области программы выделен розовым), то пользователь не может назначить активным другой проект.

Если проект назначен активным, то относящиеся к нему программы не загружаются в рабочую область автоматически, а должны выбираться пользователем вручную.

Кнопки в панели инструментов проводника по рабочим областям действуют только для активного проекта. Это кнопки:

- <Запуск> (<Start>)
- <Остановка> (<Stop>)
- <Отключить> (<Disconnect>)
- <Конфигуратор платформы> (<Platform configurator>)
- <Конфигуратор ввода-вывода> (<I/O configurator>)

4.4.3 Загрузка проекта в область программирования

Вне зависимости от того, активен проект или нет, пользователь может загрузить программы в область программирования.

Последовательность действий

1. Выделите проект, который нужно загрузить в область программирования.
2. В контекстном меню выберите "Загрузить в область программирования" ("Load in Designer").

Результат

Переключаться между загруженными программами можно с помощью вкладок, которые отображаются в верхней части рабочей области; для каждой загруженной программы добавляется своя вкладка.

4.4.4 Редактирование свойств проекта

Пользователь может изменить имя и описание проекта в разделе "Свойства" ("Properties"). Также, пользователь может указать дополнительную информацию о проекте в текстовом поле "Описание" ("Description").

Последовательность действий

1. Щелкните правой кнопкой мыши по проекту, свойства которого вы хотите редактировать.
2. В контекстном меню выберите "Свойства".
   Появится окно "Редактировать рабочую область" ("Edit Workspace").
3. Измените имя проекта и добавьте описание.
4. Щелкните <OK>.

4.4.5 Удаление проекта

Выбранный проект удаляется из рабочей области и базы данных одновременно.

Последовательность действий

1. Щелкните правой кнопкой мыши по проекту, который вы хотите удалить из рабочей области.
2. В контекстном меню выберите "Удалить" ("Remove"). Появится диалоговое окно "Удалить проект" ("Remove Project").
3. Если вы уверены, что хотите удалить данный проект из рабочей области, щелкните <OK>.
4. После удаления проекта появляется сообщение-подтверждение "Проект удален" ("Project removed").
5. Щелкните <OK>.

4.5 Задачи / Программы

При создании нового проекта ibaLogic автоматически создает программу. Однако, пользователь может добавить дополнительные программы к существующему проекту.

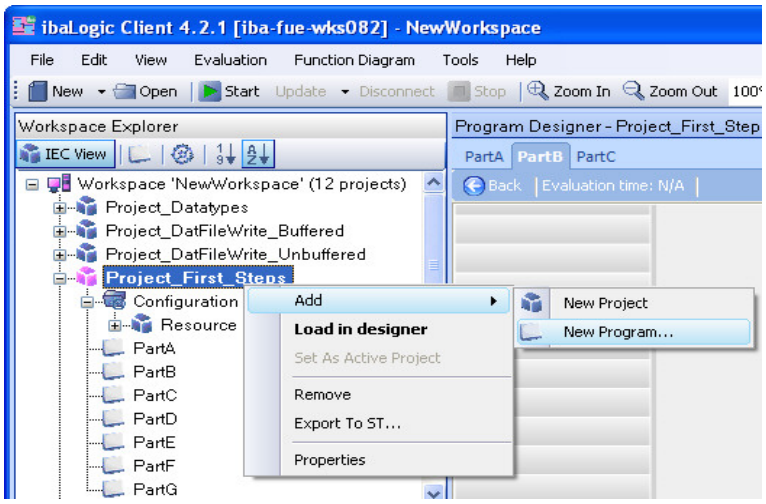
Примечание

Обратите внимание на то, что удаление последней задачи невозможно. В каждом проекте обязательно должна присутствовать минимум одна задача.

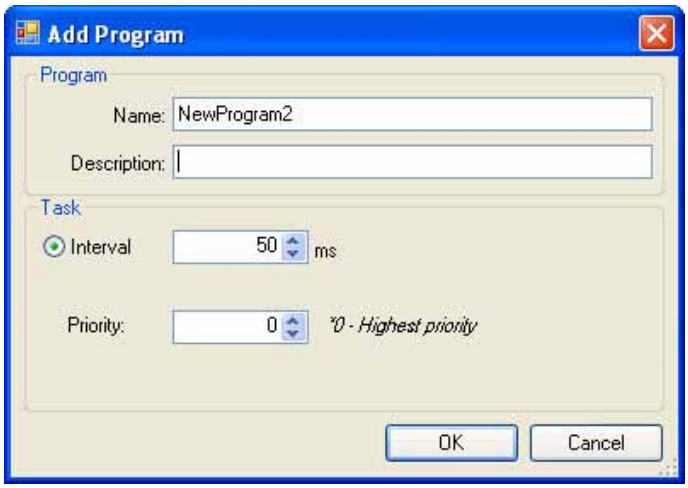
4.5.1 Создание задач / программ

Последовательность действий

1. Щелкните правой кнопкой мыши по проекту, в котором вы хотите создать новую программу.
2. В контекстном меню выберите "Добавить новую программу" ("Add New Program").



Появится диалоговое окно "Добавить программу" ("Add Program").



3. Присвойте программе имя и создайте ее описание. После этого создайте задачу для программы. Присваиваемые имена должны соответствовать

требованиям стандарта IEC. Более подробная информация содержится в приложении B "Соглашение о присвоении имен".

4. Щелкните <OK>, чтобы новая программа была добавлена.

С каждой программой может быть связана только одна задача. Пользователь может установить значения следующих параметров для задачи:

- ❑ Интервал
- ❑ Приоритет

Интервал

Программа, связанная с задачей, будет перезапускаться строго с указанным интервалом.

Если при особых обстоятельствах время исполнения программы (или всех программ) превышает указанный пользователем интервал, это означает, что система перегружена. В разделе 11.2, "Время отклика" содержится описание того, как нужно действовать в таких случаях.

По умолчанию это значение установлено на 50 мс. Самый короткий возможный интервал - 1 мс. Однако интервал не может быть меньше опорного времени, заданного в конфигураторе ввода-вывода.

Примечание

Изменить значения по умолчанию для имени программы и интервала исполнения можно следующим образом: "Инструменты – Опции – Редакторы – Рабочие области" ("Extras – Options – Editors – Workspaces").

Приоритет

Для каждой задачи устанавливается приоритет. Самый высокий приоритет обозначается цифрой "0".

Обратите внимание на то, что выполнение одной задачи не может быть прервано для выполнения другой, даже имеющей более высокий приоритет. Приоритет имеет значение только в контексте последовательности вычисления. Более подробная информация содержится в пункте 11.2, "Время отклика".

Примечание

Пользователь может настроить отображение задач в рабочей области либо в алфавитном порядке, либо в порядке приоритетности.

Для изменения критерия расположения используются значки в верхней части области навигации   .

4.5.2 Открыть задачи / программы

Последовательность действий

➲ Запустите программу двойным щелчком по ее имени в проводнике по рабочим областям.

4.5.3 Изменить свойства задачи / программы

Изменение свойств существующих задач / программ.

Последовательность действий

1. Щелкните правой кнопкой мыши по соответствующей задаче или программе.
2. В контекстном меню выберите "Свойства". Появится окно "Редактировать рабочую область" ("Edit Workspace").
3. Измените свойства.
4. Щелкните <OK>.

4.5.4 Удалить задачу / программу

Удаление программы / задачи из рабочей области.

Последовательность действий

1. Щелкните правой кнопкой мыши по соответствующей задаче или программе.
2. В контекстном меню выберите "Удалить" ("Remove"). Появится диалоговое окно для подтверждения удаления.
3. Если вы уверены, что хотите удалить программу, щелкните <Да>.

Примечание

Обратите внимание на то, что нельзя удалить последнюю задачу. В каждом проекте обязательно должна быть хотя бы одна задача.

4.6 Конфигурация входов и выходов

Входы и выходы конфигурируются с учетом периферийных устройств.

В среде разработки вы работаете с "виртуальными" сигналами. Эти сигналы должны быть связаны с реальными сигналами, поступающими от аппаратных средств.

Связать эти два типа сигналов можно как с точки зрения программы, так и аппаратного обеспечения; это можно выполнить как отдельно для каждого сигнала, так и для групп сигналов.

Важно

Более подробная информация содержится в разделе 9, "Конфигурация ввода-вывода".

Входы и выходы программного обеспечения перечислены в виде дерева в области навигации конфигуратора ввода-вывода.

Существуют следующие уровни иерархии:

Направление → группа → сигналы

- Направление:
  "Входы" или "выходы"
- Группа:
  имя группы может быть задано пользователем или соответствовать имени модуля в распределении сигналов.
- Сигналы:
  - имя сигнала может быть введено пользователем вручную, а также присваиваться в соответствии с именем сигнала из распределения сигналов.
  - за именем сигнала в скобках указывается тип данных,
  - затем после стрелки (->) следует имя сигнала аппаратного обеспечения

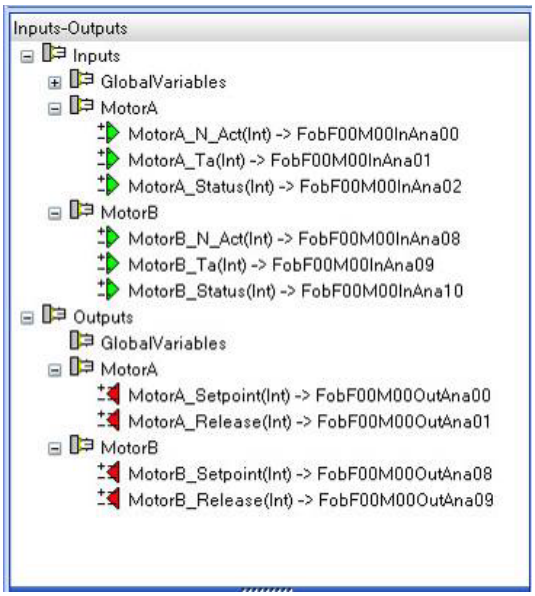


Рис. 36: Входы - выходы

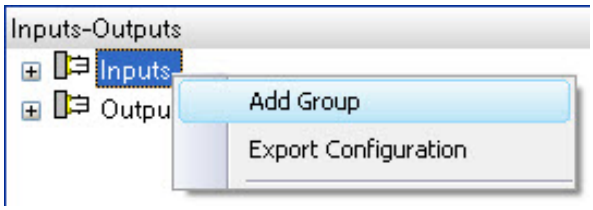
4.6.1 Создание сигналов

Сигналы разделяются на группы, что обеспечивает структурированность программы.

4.6.1.1 Определение группы

Последовательность действий

1. Правой кнопкой мыши щелкните "Входы - выходы" или вложенную папку.



2. Если необходимо, создайте сигналы в подгруппе.

Результат

Группа всегда создается как во входах, так и в выходах. Пользователь может удалить ненужную группу.

Пример

Группы: двигатель A, двигатель B

Входные сигналы: текущая скорость, температура двигателя

Выходные сигналы: заданное значение, параметр

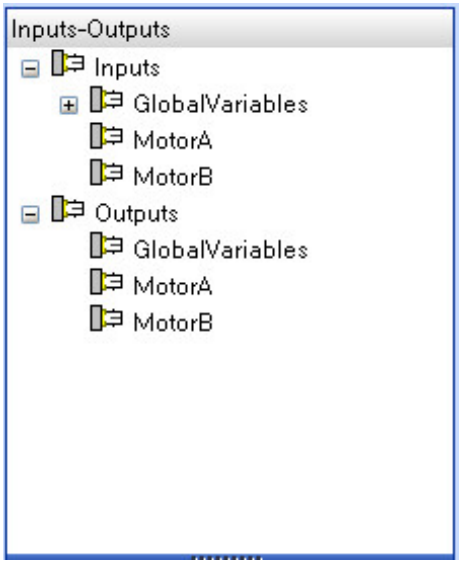


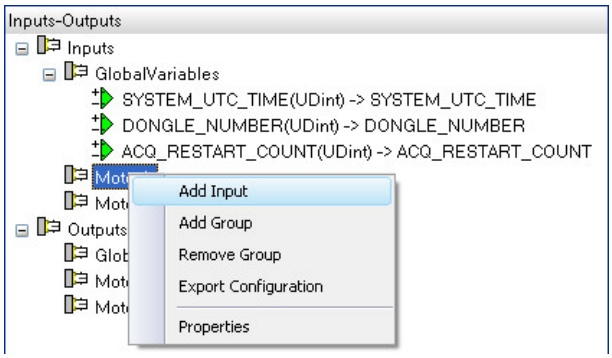
Рис. 37: Вид "Входы - выходы"

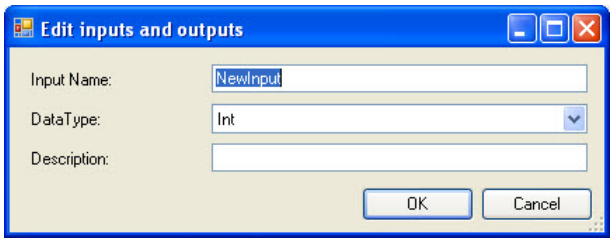
4.6.2 Определение сигналов

Последовательность действий

1. Щелкните правой кнопкой мыши по группе ввода или вывода.
2. Выберите "Добавить вход" ("Add Input") или "Добавить выход" ("Add Output").

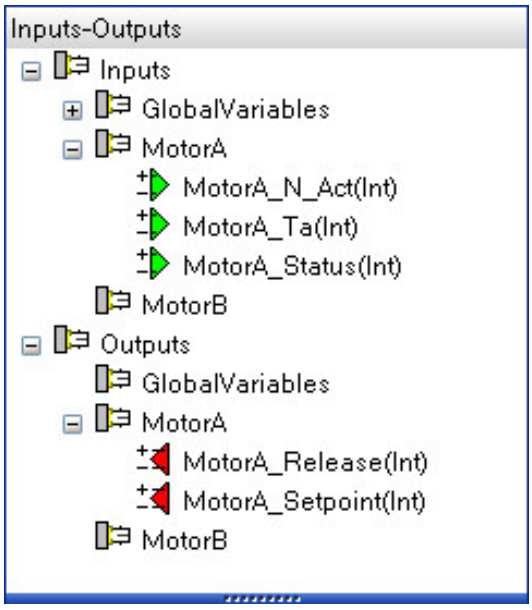
   Появится соответствующее диалоговое окно.
3. Присвойте сигналу имя и тип данных, создайте описание (если необходимо). Присваиваемые имена должны соответствовать требованиям стандарта IEC.





Тип данных зависит от связанного с сигналом периферийного устройства. Например, если используется аналоговое значение ibaPADU, то тип данных - integer.

В приведенном примере получится следующая ситуация:



Сигналы пока еще не соотнесены с программным обеспечением.

4.6.3 Редактирование существующих сигналов

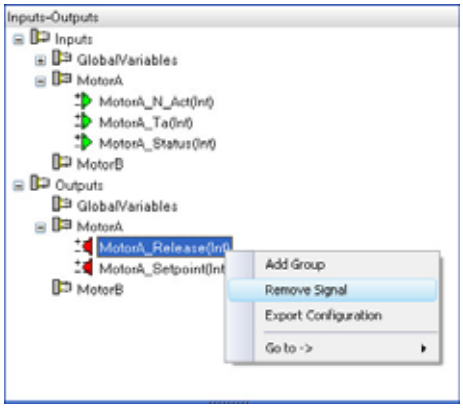
Последовательность действий

1. Щелкните по сигналу двойным щелчком.
   Откроется диалоговое окно для редактирования.
2. Присвойте сигналу имя и тип данных, создайте описание (если необходимо)
   Присваиваемые имена должны соответствовать требованиям стандарта IEC.
3. Щелкните <OK>, чтобы изменения были приняты системой.

4.6.4 Удаление сигналов

Последовательность действий

1. Щелкните правой кнопкой мыши по сигналу, который вы хотите удалить.
   Откроется контекстное меню.
2. В контекстном меню выберите "Удалить сигнал" ("Remove Signal").



Примечание

Вы можете редактировать и удалять только те входы и выходы, которые не используются в программе, т.е. не отображаются в панели входных или выходных сигналов.

Примечание

Процедура соотнесения сигналов с аппаратным обеспечением описана в пункте 9.3, "Присвоение сигналов".

Важно

Интерфейсные карты некоторых производителей не поддерживают отдельные сигналы с элементарными типами данных: REAL и INT. В этом случае структура данных для входов и выходов создается для карты SST ведущего устройства Profibus и зависит от конфигурации ведомого устройства (GSD-файл). Вы должны также определить эту структуру в ibaLogic, чтобы использовать эти сигналы.

Пример подключения ведущей карты Profibus содержится в документации на CD, который входит в объем поставки.

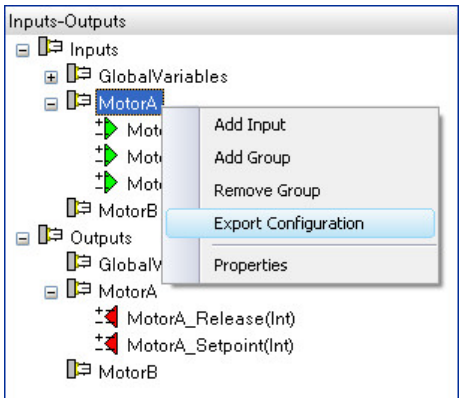
4.6.5 Экспорт / импорт сигналов

Имена виртуальных сигналов часто содержатся в проектной документации. Для использования этих сигналов в системе предусмотрены функции экспорта и импорта.

Экспорт всей конфигурации ввода-вывода

Последовательность действий

1. Выберите сигнал.
2. В дереве сигналов в контекстном меню выберите "Экспортировать конфигурацию" ("Export Configuration").

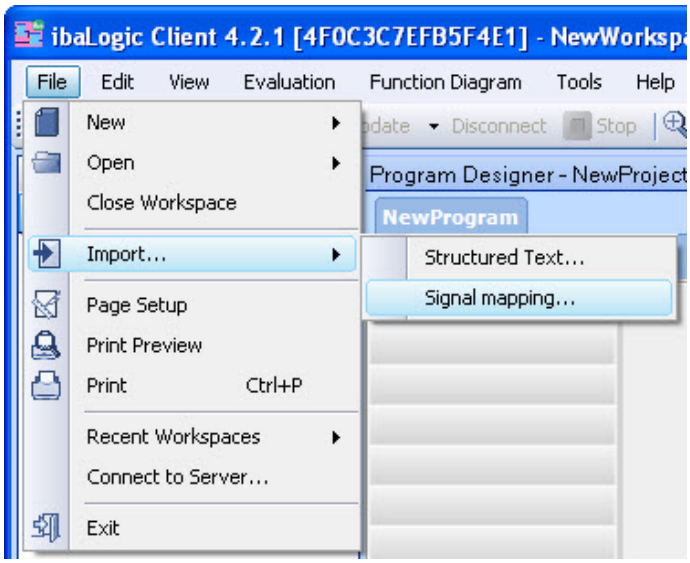


3. Функцию экспорта можно открыть, например, в программе для работы с электронными таблицами.
4. Введите название виртуальной группы и сигнала в столбцы "Группа" и "Значок" соответственно или измените уже существующие имена.

```
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;
```

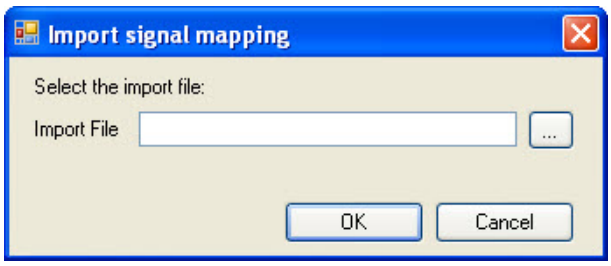
	A	B	C	D	E	F
1	**Adresse**	**Typ**	**InOut**	**Gruppe**	**Symbol**	**Kommentar**
2	FobF00M00InAna00	INT	Input	FobF00M00	FobF00M00InAna00	
3	FobF00M00InAna01	INT	Input	FobF00M00	FobF00M00InAna01	
4	FobF00M00InAna02	INT	Input	FobF00M00	FobF00M00InAna02	

5. Импорт назначения сигналов. В основном меню выберите "Файл – Импорт – Назначение сигналов" ("File – Import – Signal Assignment").



Появится диалоговое окно "Импорт назначения сигналов" ("Import Signal Assignment").

6. Укажите имя файла и путь к целевому каталогу.



7. Щелкните <OK>, чтобы запустить процесс импортирования.

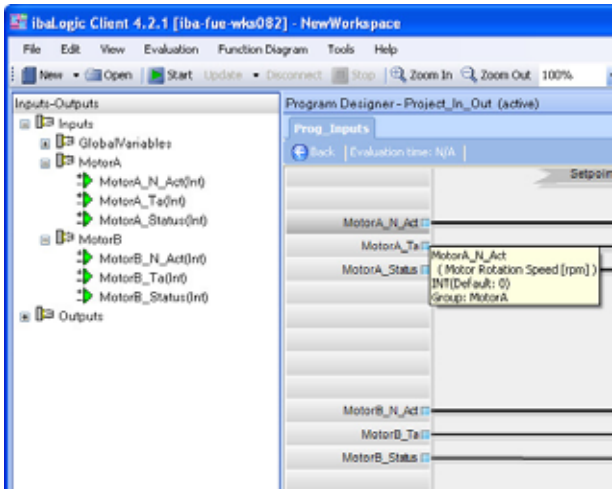
4.6.6 Использование сигналов в программе

Дополнительную информацию вы сможете получить, ознакомившись с разделом "Конфигуратор ввода-вывода".

Для того чтобы использовать сигналы в программе или проекте, их нужно перетащить в боковую панель входов или выходов.

Последовательность действий

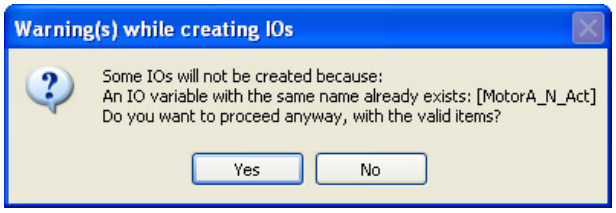
➲ Перетащите нужный вам сигнал или группу сигналов на боковую панель входов или выходов.



Пример

Сигнал "MotorA_N_Ist" был перемещен на боковую панель входов.
При наведении курсора мыши на символ коннектора сигнала, появляется подсказка с информацией о сигнале.

Если вы перетащите целую группу сигналов на боковую панель, то там будут размещены все сигналы этой группы. Если некоторые сигналы группы уже находятся на боковой панели, то появится следующее предупреждение:



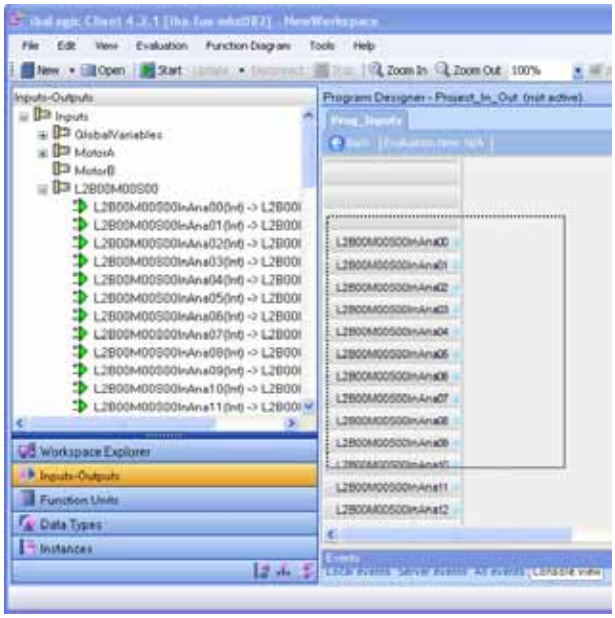
4.6.7 Удаление сигналов из программы

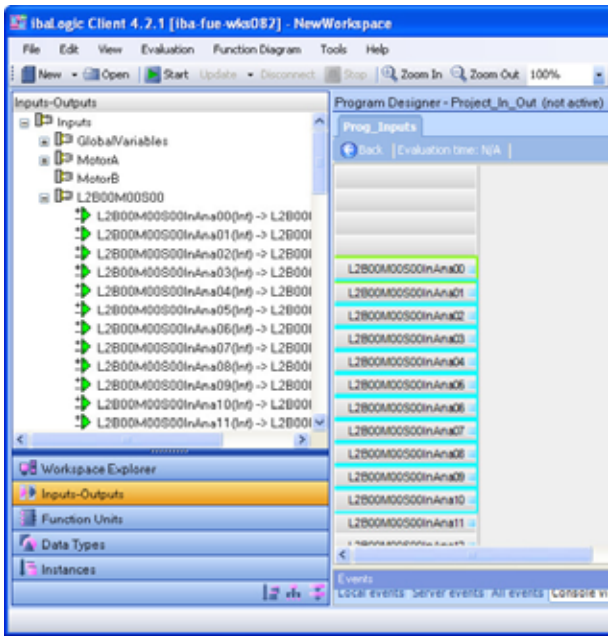
Последовательность действий

1. Выделите сигнал в панели входов или выходов.
2. Нажмите клавишу <DEL>.

Примечания

Можно выделить одновременно несколько сигналов. Для этого нажмите и удерживайте клавишу <Shift> и выделите курсором нужные сигналы. Также выделить сигналы можно, очертив вокруг них прямоугольник курсором мыши, нажав и удерживая левую кнопку мыши. Все сигналы, которые необходимо удалить, должны находиться в пределах границ прямоугольника.





5 Создание программы

5.1 Блоки

В глобальной библиотеке ibaLogic содержится большое количество функциональных блоков.

В дополнение к функциональным блокам, созданным в соответствии со стандартом IEC 1131-3, компания iba также предоставляет функциональные блоки собственной разработки и пользовательские блоки. Блоки перечисляются в области навигации вида "Функциональные блоки".

Каждый блок сопровождается значком. Эти значки имеют следующее значение:

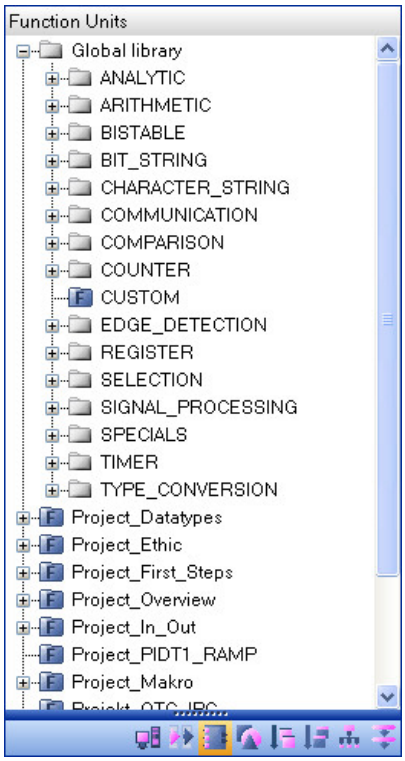
Блоки стандарта IEC 1131-3

Блоки компании iba

Функции, которые доступны как DLL

Функциональные блоки и макросы, созданные пользователем

Функциональные блоки распределяются по группам и подгруппам, которые перечисляются в алфавитном порядке в соответствии со спецификацией IEC 1131-3.



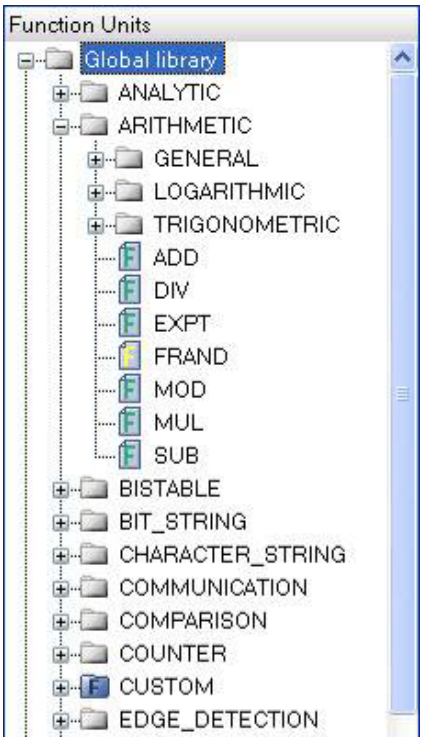


Рис. 38: Блоки "Глобальная библиотека" Блоки "Арифметические операции"

Глобальная библиотека также содержит папки "ПОЛЬЗОВАТЕЛЬСКИЕ БЛОКИ" ("CUSTOM") и "НОВЫЙ ПРОЕКТ" ("NEW PROJECT").

ПОЛЬЗОВАТЕЛЬСКИЕ БЛОКИ (CUSTOM)

В этой папке сохраняются все глобальные пользовательские блоки.

СПЕЦИАЛЬНЫЕ БЛОКИ (SPECIALS)

В этой папке сохраняются специальные блоки ibaLogic. Более подробная информация содержится в разделе D.13, "Специальные блоки".

5.1.1 Использование блоков

В проекте пользователь может использовать все блоки из глобальной библиотеки и библиотеки проекта.

Если вы хотите использовать блок из глобальной библиотеки, перетащите его (с помощью функции Drag & Drop) в окно программирования.

Доступ к блокам, которые были определены в проекте другой рабочей области, невозможен.

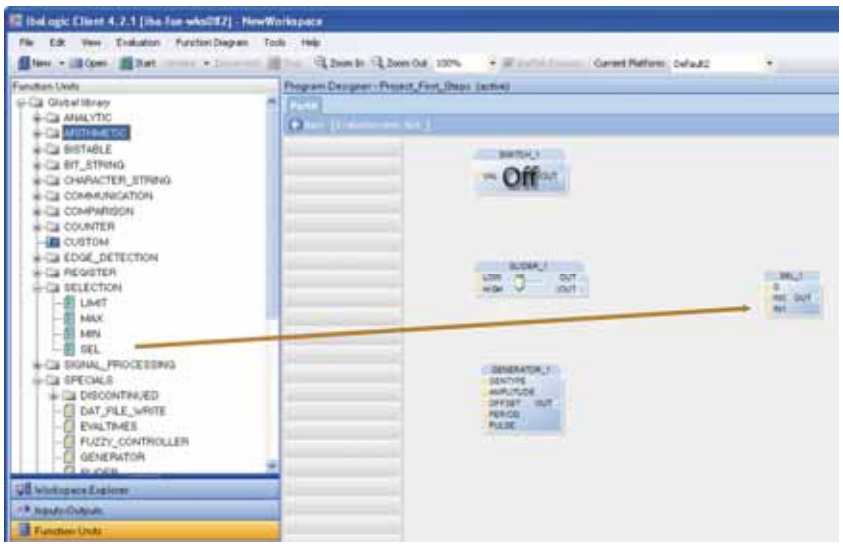
Важно

При внесении изменений в содержимое блока, изменения также вносятся во все экземпляры блока и его определение.

Если вы захотите изменить форму, т.е. входы и выходы или внутренние переменные, то необходимо будет указать другое имя для этого определения. В этом случае создается новый тип блока, а предыдущее определение блока и все его экземпляры не изменяются.

Последовательность действий

1. В "Обзоре функциональных блоков" ("Function Block Overview") выберите блок, который хотите поместить в область программы, и перетащите его в любое свободное место этой области.



5.1.2 Создание пользовательских блоков

Пользователь может создавать блоки несколькими способами:

- ❑ В программе
- ❑ В проекте
- ❑ В глобальной библиотеке

В программе

Создание функционального блока в программе.

Последовательность действий

1. Щелкните левой кнопкой мыши по любому свободному месту в окне программирования.
2. В контекстном меню выберите "Новый... Новый функциональный блок..." ("New… - New Function Block") или "Новый... Новый макрос..." ("New… New Macro Block"). Появится окно с заголовком "Создать функциональный блок" ("Create Function Block").
3. Создайте свой блок.
4. Если синтаксические ошибки отсутствуют, то после нажатия <OK> будет создан новый блок.

В проекте

Создание блока в проекте.

Последовательность действий

1. Откройте вид "Функциональные блоки" ("Function Blocks").
2. Перейдите к папке с проектами.
3. В дереве элементов правой кнопкой мыши щелкните по нужному проекту.
4. В контекстном меню выберите "Новый... Новый функциональный блок..." ("New… - New Function Block") или "Новый... Новый макрос..." ("New… New Macro Block"). Появится окно с заголовком "Создать функциональный блок" ("Create Function Block").
5. Создайте свой блок.
6. Если синтаксические ошибки отсутствуют, то после нажатия <OK> будет создан новый блок.

В глобальной библиотеке

Создание блока в глобальной библиотеке.

Последовательность действий

1. В глобальной библиотеке щелкните правой кнопкой мыши по группе "ПОЛЬЗОВАТЕЛЬСКИЕ БЛОКИ" ("CUSTOM") или подгруппе, которая была определена предварительно.
2. В контекстном меню выберите "Новый... Новый функциональный блок..." ("New… - New Function Block") или "Новый... Новый макрос..." ("New… New Macro Block"). Появится окно с заголовком "Создать функциональный блок" ("Create Function Block").

3. Создайте свой блок.
4. Если синтаксические ошибки отсутствуют, то после нажатия <OK> будет создан новый блок.

Примечание касательно разницы между определением и экземпляром

Более подробная информация содержится в пункте 4.2.3.2, "Экземпляры ".

5.1.3 Управление блоками

Копирование в глобальную библиотеку

Если вы хотите использовать блок, который вы определили в программе или проекте, в других рабочих областях, то этот блок нужно скопировать в глобальную библиотеку.

Последовательность действий

1. Щелкните правой кнопкой мыши по блоку, который вы хотите скопировать.
2. В контекстном меню выберите "Копировать в глобальную библиотеку" ("Copy in the global library").
   Блок будет скопирован в группу "ПОЛЬЗОВАТЕЛЬСКИЕ БЛОКИ".
3. Вы также можете просто перетащить блок в подгруппу.

Примечание

Блок, скопированный в глобальную библиотеку, сохраняет свойства, которые были у него в момент копирования. Если блок проекта будет изменен, то эти два блока будут разными.

Примечание

Нельзя копировать или перемещать блоки из глобальной библиотеки в каталог проекта.
Если пользователь использует блок из глобальной библиотеки, то он автоматически копируется в каталог проекта..

Вы можете использовать блоки одного проекта в другом проекте в рамках одной рабочей области. В результате такие блоки создаются автоматически в блоках проекта.

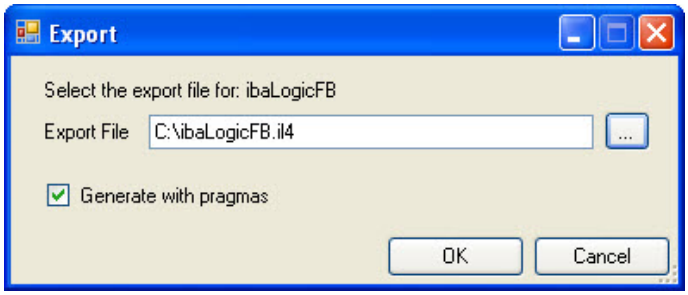
Более подробная информация содержится в пункте 5.1.1, "Использование блоков ".

5.1.4 Экспорт блоков

Если вы хотите использовать функциональный блок, который вы создали в проекте или в базе данных в группе пользовательских блоков, также в другой базе данных или на другом компьютере, то для этого нужно этот блок экспортировать в виде текстового файла.

Последовательность действий

1. Щелкните правой кнопкой мыши по блоку в "ПОЛЬЗОВАТЕЛЬСКИХ БЛОКАХ" или в проекте.
2. В контекстном меню выберите "Экспортировать в ST" ("Export to ST"). Появится диалоговое окно "Экспорт".
3. Определите целевой каталог и имя файла.



Примечание

Если вы хотите использовать блок, экспортированный в среду ibaLogic-V4, то его нужно экспортировать с дополнительной графической информацией.

Если вы хотите использовать блок в другой системе, то его можно экспортировать без этой дополнительной информации.

Важно

Поскольку реализация стандарта IEC зависит от конкретного производителя, то, прежде чем соединять блок, пользователь должен проверить, не имеет ли внешняя система несоответствий или отклонений от стандарта IEC.

5.1.5 Импорт блоков

Необходимые условия

- ibaLogic не работает в онлайн-режиме.
- Имеется файл (блок), который требуется импортировать.

Последовательность действий

1. Выберите меню "Файл – Импорт – Структурированный текст" ("File – Import – Structured Text").
   Появится окно "Импорт структурированного текста" ("Structured text import").
2. Выберите файл, который необходимо импортировать.
3. Подтвердите выбор, нажав <OK>. Импорт файла выполнен.

Окно выбора	Объяснение
Определить организационную единицу как новую при импорте	Организационная единица будет импортирована как новый блок.
Выбор версии файла	Организационная единица может быть импортирована только как файл формата *.il4. В данный момент доступен только формат файла для версии 4.

5.1.6 Удаление блоков

При удалении блока из программы удаляется только один его экземпляр.

Последовательность действий

1. Выберите блок из библиотеки проекта.
2. Откройте контекстное меню.
3. Выберите <Удалить> (<Remove>).

Примечания

Диалоговое окно появляется только при удалении единственного экземпляра блока. В этом диалоговом окне можно указать, должно ли быть также удалено определение. Подтвердите выбор, нажав <OK>. Блок удален.

Функция также удаляется из группы функций проекта и пользователь не сможет больше ее использовать.

Глобальное определение блока в группе "ПОЛЬЗОВАТЕЛЬСКИЕ БЛОКИ" вышеуказанным способом удалить нельзя. Удалить такой блок можно, щелкнув правой кнопкой мыши в контекстном меню "Удалить" по блоку в этой группе или нажав клавишу <Del>.

5.2 Стандартные блоки

В Приложении в виде таблицы содержится обзор всех функций и функциональных блоков, которые доступны в ibaLogic.

5.3 Комплексные функциональные блоки

5.3.1 DAT_FILE_WRITE (функциональный блок DFW)

В ibaLogic интегрирована функция, которая позволяет сохранять существующие аналоговые и цифровые сигналы в .dat-файлы в режиме исполнения программы. Формат файлов .dat - это формат iba, соответственно, файлы этого формата можно открывать, анализировать и дополнительно обрабатывать с помощью инструментов iba - ibaDatCoordinator и ibaAnalyzer (например, для извлечения данных в базу данных).

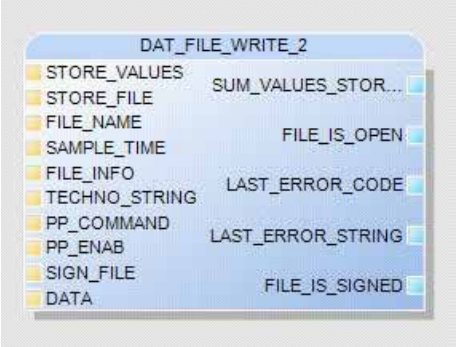


Рис. 39: Блок DAT_FILE_WRITE

Программа позволяет сохранять как отдельные значения, так и массивы буферизованных данных. Доступны следующие типы данных: INTEGER, REAL и BOOL. ibaLogic поддерживает также сохранение дополнительной информации, например техностроk. Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".

Примечание

Для записи данных в .dat-файлы требуется лицензия. Без соответствующего аппаратного ключа FILE_IS_SIGNED сохраняет статус "FALSE" ("ЛОЖЬ") для записи данных, следовательно, пользователь не может анализировать созданные файлы в ibaAnalyzer.

Примечание

Пример конфигурации приводится в Приложении.

5.3.1.1 Редактирование функционального блока DFW

После того как вы перетащили блок DFW в окно программы, вы можете открыть его двойным щелчком мыши.

Появившееся диалоговое окно содержит следующие вкладки:

- "Переменные" ("Variables")
- "Аргументы" ("Arguments")
- "Графическое отображение" ("Graphical")

Вкладка "Графическое отображение", в свою очередь, содержит вложенные вкладки:

- "Общая конфигурация"
- "Конфигурация сигналов"

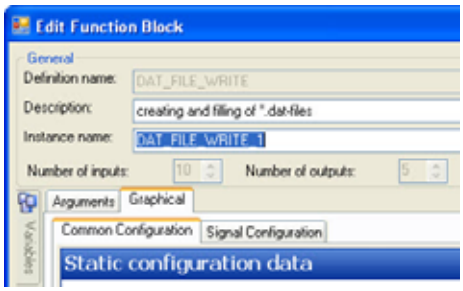


Рис. 40: Конфигуратор DAT_FILE_WRITE

Вкладка "Переменные"

Вкладка содержит всю информацию о входных и выходных переменных, а также внутренних переменных. Если эта вкладка не участвует в конфигурировании, то ее можно скрыть. Более подробная информация содержится в разделе 9.2 "Конфигурация аппаратного обеспечения".

Вкладка "Аргументы"

Во вкладке "Аргументы" в виде таблицы отображаются все входы, выходы и связанные с ними переменные и типы данных. Эта вкладка также используется для общего обзора и для отображения текущих значений в онлайн-режиме. Не изменяйте никакие настройки в этом виде, а перейдите к вкладке "Графическое отображение". Если возникнет необходимость сконфигурировать некоторые свойства во вкладке "Аргументы", программа сообщит вам об этом.

Важно

При соединении входного коннектора значения по умолчанию и значения, сконфигурированные в функциональном блоке, перезаписываются. Если пользователь удаляет входной коннектор, то программа оставляет предыдущее значение.

Вложенная вкладка "Общая конфигурация"

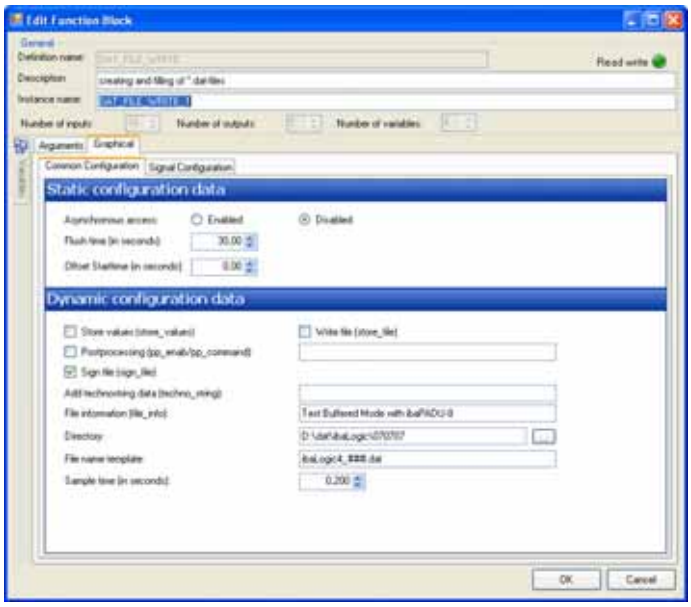


Рис. 41: "Вложенная вкладка "Общая конфигурация"

- Асинхронный доступ

Деактивирован:
Выполняется внутренняя буферизация данных для сохранения. Если буфер заполнен, данные записываются на жесткий диск. Если запись файлов завершена, можно приступить к последующему анализу .dat-файла (посредством ibaAnalyzer).

Активирован:
Содержимое внутреннего буфера циклически записывается на жесткий диск в соответствии со сконфигурированным циклом сохранения. Следовательно, выполнять анализ данных можно уже в процессе их записи. Чем короче цикл сохранения, тем раньше данные становятся доступны в ibaAnalyzer (если в этой программе настроена "автоматическая перезагрузка" ("automatic reload")). Следствием такого способа работы с данными является более низкая степень сжатия данных и более высокая загрузка компьютера или сети.

- Смещение начального времени (Start time offset)

Этот параметр сдвигает ось времени назад во времени в .dat-файлах. Ось времени в .dat-файле формируется точкой начального времени и периодом дискретизации, т.е. количеством дискретных значений и интервалом между отдельными значениями. Таким образом, начальное время .dat-файла: "нормальное начальное время" – смещение начального времени.

В связи с задержкой сохранения данных, можно создать "пре-триггер" (используя блок DELAY).

Запись данных начнется после активации триггера. Для того чтобы скомпенсировать задержку при записи, нужно выставить смещение начального времени, равное задержке.

- ❑ Записать файл (сохранить_файл) (Write file (store_file))

Не активируйте эту опцию во вкладке, а настройте эту функцию с помощью коннектора STORE_FILE.

Передний фронт: Создается .dat-файл с именем и путем (каталогом) сохранения, которые указаны в настройках. Запись начинается, только когда STORE_VALUES = TRUE.

Задний фронт: Файл будет закрыт.

- ❑ Сохранить значения (Save values (store_values))

Когда это значение "TRUE" и файл открыт, в каждом цикле вычислений сохраняется одно дискретное значение.

Этот параметр зависит от режима обработки данных:

Для значений, буферизация которых не выполняется: активируйте эту опцию и не соединяйте коннектор STORE_VALUES. Для контроля записи вы можете использовать STORE_FILE.

Примечание

Если вы будете контролировать запись небуферизованных данных с помощью STORE_VALUES, вы получите неверную шкалу времени. Поскольку ось времени определяется в ibaAnalyzer количеством дискретных значений и периодом между ними, динамическая активация/деактивация STORE_VALUES не приводит к пробелам на оси времени, напротив - дискретные значения образуют последовательности, а пробелы появляются в конце .dat-файлов.

В случае буферных значений эта переменная используется по-другому. Более подробная информация содержится в пункте 5.3.1.2, "Вложенная вкладка "Конфигурация сигналов" ("Signal Configuration")".

- ❑ Последующая обработка (pp_enab/pp_command)

Последующая обработка в ibaLogic V4 совместима с ibaLogic V3.
Для выполнения последующей обработки данных iba рекомендует использовать ibaDatCoordinator. Эта программа предоставляет все необходимые функции для последующей обработки: копирование файлов на файловый сервер, передачу данных в ibaAnalyzer для последующего их извлечения в базу данных и т.д.

- ❑ Отметить файл (sign_file)

Файл отмечается программой, если выбрана опция "Отметить файл". Это позволяет использовать дополнительные функции в ibaAnalyzer. Эта опция должна быть постоянно активна. Если аппаратный ключ не найден, то опция имеет значение FALSE.

- ❑ Добавить данные из техностроки (techno_string)

Техностока содержит данные, связанные с измерениями: номер партии, обозначение материала и т.д. Эти данные также можно анализировать в ibaAnalyzer.

Техностока сохраняется при закрытии файла.

Если вы хотите использовать техностроку, соедините коннектор TECHNO_STRING с переменной типа string, например данными, получаемыми модулем TCP/IP.

❑ Информация о файле (file_info)

Это дополнительная текстовая (ASCII) информация, которая сохраняется при закрытии файла. При выполнении анализа данных эта информация доступна в разделе "Info". iba рекомендует использовать входной коннектор FILE_INFO, который позволит динамически менять содержимое этого раздела.

Информационные поля в ibaAnalyzer имеют следующую структуру:

Имя поля: текст

Если в тексте нет ":", то по умолчанию ibaLogic устанавливает значение "UserField0".
Информационные поля разделяются с помощью знака ":".

Примечание

Стандартные информационные поля не перезаписываются. Записи с таким именем игнорируются.

Имена стандартных информационных полей:

Рис. 42: Имена стандартных информационных полей

❑ Папка (часть file_name)

➲ Выберите жесткий диск и путь для сохранения .dat-файлов, щелкнув кнопку просмотра <...>.

Важно

Если исполнительная система установлена на платформе PADU-S-IT, то работать с файловым менеджером Windows пользователь не может. Укажите путь сохранения и имя файла в настройках коннектора FILE_NAME во вкладке "Аргументы" или соедините коннектор с переменной типа строка (string), с помощью которой можно динамически задавать путь сохранения и имя файла. Введенные данные будут приняты при открытии файла (STORE_FILE).

В качестве пути сохранения используйте диск c:\. Затем данные можно перенести в ibaDatCoordinator и открыть их в этой программе, например: \\padu-s-it-16-000074\datfiles. Здесь "datfiles" - это внутреннее имя, а "padu-s-it-16-000074" - имя устройства PADU-S-IT, к которому адресуется программа (также можно непосредственно указать адрес TCP/IP).

- Шаблон имени файла (часть file_name)

Указание имени файла и индекса. При создании каждого нового .dat-файла индекс увеличивается, если путь сохранения и имя файла остаются без изменений.

Символ "#" используется в качестве символа-заполнителя для индекса. Для индексов, которые состоят из нескольких цифр указывайте, соответственно, несколько символов-заполнителей.

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

Если путь сохранения не был изменен, то, после того как все значения индекса закончатся, программа начнет перезаписывать .dat-файлы, начиная с самых старых.

- Период дискретизации (в секундах) (sample_time)

Это значение является опорным временем .dat-файла. Оно определяет интервал в секундах между 2 дискретными значениями измеряемого сигнала, которые сохраняются программой.

Если речь идет о небуферизованных значениях, указывайте интервал задачи для программы, в которой содержится блок DFW.

Необходимо указать период времени, соответствующий глубине массива данных для буферизованных значений.
Более подробная информация содержится в пункте 5.3.1.2, "Вложенная вкладка "Конфигурация сигналов" ("Signal Configuration")".

Важно

Подготовка сигналов для измерения и блок DFW должны исполняться с одинаковым интервалом задачи. В противном случае ось времени в .dat-файле либо растягивается, либо сжимается.

Пример периода дискретизации

Если задача, в которой поступают данные, выполняется с интервалом 10 мс, то в блоке DFW нужно указать период дискретизации, равный 0,01 сек.

Если необходимо, чтобы значения сохранялись, например, каждые 50 мс, то недостаточно только установить период дискретизации на 0,05 секунды (в этом случае сохраняются те же данные, но период времени по оси X становится длиннее в 5 раз). Пользователь должен указать тактовый цикл на коннекторе STORE_VALUES, который принимает значение "TRUE" только каждый 5 цикл.

Однако более простым вариантом в данном случае будет оставить для исполнения блока значение 50 мс, а для коннектора STORE_VALUES установить постоянное значение "TRUE".

Совет

Если, несмотря на правильно выполненную конфигурацию параметров, в процессе анализа вы заметите, что ось времени слишком длинна или коротка, проверьте, совпадает ли действительный интервал задачи (Block EVALTIMES) заданному при конфигурации. Если используются короткие циклы, то рекомендуется активировать режим турбо (turbo mode), чтобы интервал задачи был постоянным.

5.3.1.2 Вложенная вкладка "Конфигурация сигналов" ("Signal Configuration")

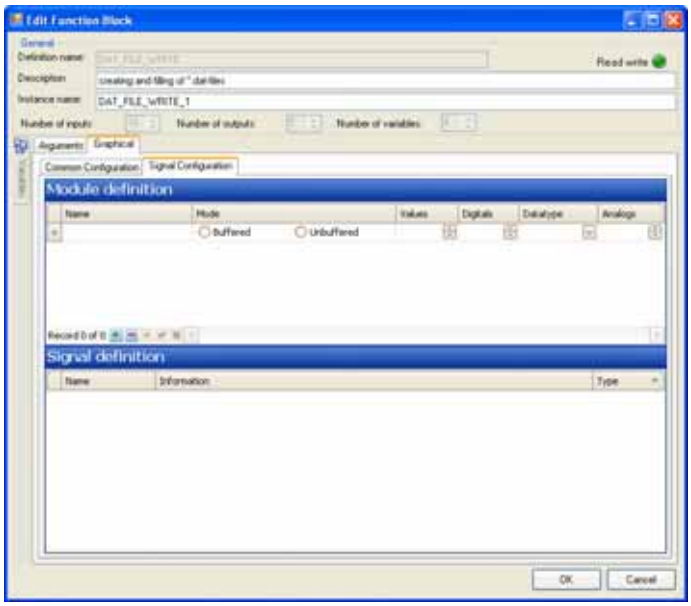


Рис. 43: Вкладка "Конфигурация сигналов"

Вкладка "Конфигурация сигналов" содержит следующие разделы:

- Определение модуля
  определяет имя модуля, тип, количество сигналов и тип данных
- Определение сигнала
  определяет имя и описание сигнала

Важно

В текущей версии ibaLogic перед соединением измерительного сигнала нужно сконфигурировать все модули и сигналы в блоке DFW.

Причина: ibaLogic создает внутреннюю структуру и типы массивов данных, которые невозможно изменить, после того как они уже начали использоваться. Если вы впоследствии изменяете конфигурацию сигналов, то вам нужно либо удалить все соединительные элементы, добавленные автоматически, и соединить их заново, либо изменить все соответствующие типы данных в ST-блоках.

Для создания нового модуля в конце списка модулей предусмотрен шаблон с пустым именем. От пользователя требуется только ввести в шаблон имя модуля и сконфигурировать его. После создания нового модуля в конце списка появляется новый пустой шаблон.

В принципе, пользователь может создать любое необходимое количество модулей. Максимальное количество зависит от производительности исполнительной системы ibaLogic.

Описание модулей:

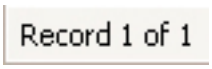
- Имя:
  Имя модуля, которое должно соответствовать требованиям стандарта IEC.
- Режим без буферизации (Unbuffered mode):
  Запись отдельных значений. Каждый цикл сохраняется одно дискретное значение. Значение в столбце "Значения" ("Values") не учитывается.
- Режим буферизации (Buffered mode):
  Запись пакетов данных. Каждый цикл сохраняется массив дискретных значений. Значение в столбце "Значения" соответствует глубине массива, т.е. обозначает количество дискретных значений сигнала.
  Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".
- Значения:
  В режиме без буферизации не используется.
  Количество дискретных значений, которые сохраняются за один цикл в режиме буферизации.
- Цифровые значения:
  Количество цифровых сигналов в модуле (макс. 32)
- Допускаются типы данных аналоговых значений, REAL и INTEGER
- Аналоговые значения:
  Количество аналоговых значений в модуле (макс. 32)

Все сигналы, относящиеся к выделенному модулю, отображаются в виде списка под определением сигналов. Создаваемым сигналам присваиваются имена по умолчанию (Digital_nn и Analog_nn). В разделе "Информация" пользователь может редактировать имена сигналов и создавать описание для каждого сигнала.

Примечание

Если коннектор "DATA" соединен с данными, изменять конфигурацию сигналов нельзя. Если требуется внести изменения, необходимо разорвать это соединение. При этом все соединители, созданные автоматически, будут удалены.

Панель инструментов для редактирования списка определений модулей:

Значок	Имя / Подсказка	Объяснение
	Добавить	Добавляется новое пустое определение модуля.
	Удалить	Выбранное определение модуля будет удалено.
	Редактировать	Разблокирует определение модуля для редактирования.
	Завершить редактирование	Данные измененного определения модуля будут приняты.
	Отменить редактирование	Данные измененного определения модуля не будут приняты.
	-	Количество записей

Более подробная информация содержится в Приложении.

5.3.1.3 Создание структуры хранилища

Самый простой способ настроить сохранение .dat-файлов - это указать определенную папку и определенное базовое имя для файлов, к которому iba-Logic будет автоматически добавлять порядковый номер.

Если вам необходима структура с вложенными папками и именами файлов, которые, например, будут содержать текущий номер партии, то это нужно специально программировать.

Пример

Новый файл должен создаваться каждый час. Имя файла должно содержать информацию о времени.

Реализация

Имя файла создается с использованием текущего значения времени. Когда значение времени (час) изменяется, на вход STORE_FILE блока DFW подается импульс низкого уровня с помощью блоков DELAY и EQ. Таким образом поступающее имя присваивается следующему .dat-файлу.

Имя файла создается с использованием блоков CONCAT. Путь сохранения и базовое имя файла содержатся в первом блоке CONCAT. К этой информации добавляются данные о времени, а также расширение .dat посредством 2-го блока CONCAT. Например, таким способом было создано следующее имя файла: c:\iba11.dat.

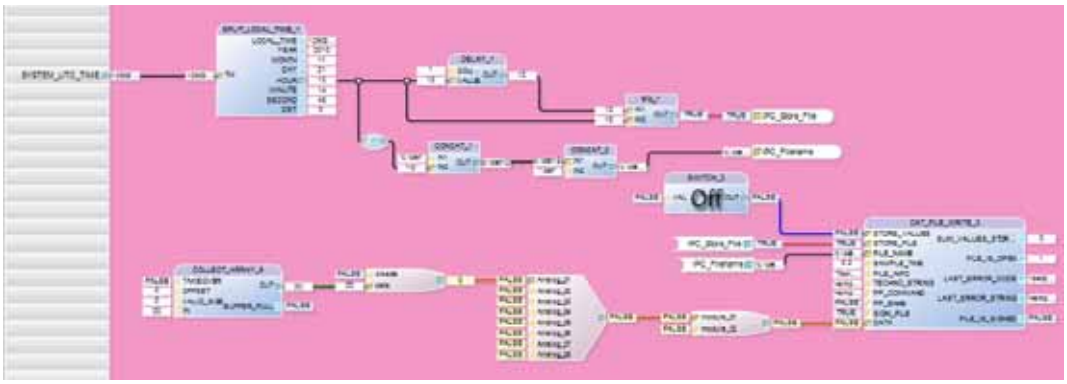


Рис. 44: Пример: блоки CONCAT

Точно таким же образом можно создать уникальное имя .dat-файла, включая путь сохранения. Если вы указываете новый путь сохранения, ibaLogic создает его автоматически.

5.3.2 TCPIP_SENDRECV

Этот блок позволяет получать и передавать исходные данные по TCP/IP. Таким образом, вы можете реализовать все "родные" протоколы TCP/IP.

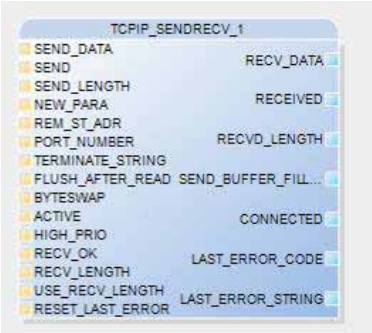


Рис. 45: Функциональный блок TCPIP_SENDRECV

Примечание

Для использования этого блока необходима лицензия.

Важно

Вы можете настроить максимально допустимое количество параллельных подключений в Windows XP. Это количество зависит от конфигурации Windows, изменить его можно в реестре Windows.

Более подробная информация содержится в пункте F1, "Количество возможных TCP/IP-соединений".

5.3.2.1 Входы

Коннектор	Тип данных	Объяснение
SEND_DATA	Любой	Данные для передачи. Тип данных ЛЮБОЙ (ANY), т.е. тип данных устанавливается в соответствии с типом данных интерфейса, например: structure, string, structure, array.
SEND	Boolean	Когда этот коннектор принимает значение "TRUE", выполняется передача данных из SEND_DATA. Этот вход не зависит от фронта, т.е. фиксированное значение "TRUE" на входе инициирует передачу данных каждый цикл.
SEND_LENGTH	Udint	Длина данных для передачи в байтах. Если значение = 0, то выполняется передача всех доступных данных.
NEW_PARA	Boolean	Принять новые параметры соединения.
REM_ST_ADR	String	IP другой стороны соединения. Этот IP необходим только для установления активного соединения, т.е. когда вход ACTIVE = TRUE. Если ACTIVE = FALSE, то блок ожидает данных от другого конца соединения: IP-адрес компьютера или PADU-S-IT. Вместо IP-адреса пользователь также может ввести имя компьютера. Установление имени можнт занять некоторое время (в зависимости от конфигурации ОС).
PORT_NUMBER	Udint	Если ACTIVE = TRUE: номер порта другой стороны соединения. Если ACTIVE = FALSE: собственный номер порта
TERMINATE_STRING	Udint	Строки заканчиваются нулевым байтом (NULL). Этот вход вычисляется, когда на входе SEND_DATA есть строки.
FLUSH_AFTER_READ	Boolean	Очищает буфер полученных данных после их считывания.
BYTESWAP	Integer	= 1: своппинг на основе типа данных (AB CDEF → BA FEDC) = 2: своппинг 2-х байтов (ABCD → BADC) = 3: своппинг 4-х байтов (ABCD → DCBA)
ACTIVE	Boolean	TRUE: (ibaLogic является клиентом TCP/IP) Блок выполняет попытки установить соединение с IP и портом, которые указаны в REM_ST_ADR и PORT_NUMBER соответственно. FALSE: (ibaLogic является сервером TCP/IP) ibaLogic ожидает входящие подключения на сконфигурированном номере порта. IP-адрес не определяется.
HIGH_PRIO	Boolean	Данные извлекаются из сетевого буфера Windows и записываются во входной буфер. **ПРИМЕЧАНИЕ** По умолчанию эта функция должна быть установлена на "FALSE".
RECV_OK	Boolean	TRUE: полученные данные OK, во входной буфер могут помещаться новые данные. FALSE: последние полученные данные сохраняются в буфере, пока вход не будет инициирован снова.
RECV_LENGTH	Udint	Определяет длину телеграммы в байтах. Этот вход вычисляется, только если вход USE_RECV_LENGTH = TRUE
USE_RECV_LENGTH	Boolean	TRUE: если входной буфер больше сконфигурированного RECV_LENGTH, то на выходе RECV_DATA будет только одна телеграмма заданной длины. FALSE: на выходе RECV_DATA имеется одна телеграмма с

Коннектор	Тип данных	Объяснение
		максимальным размером.
RESET_LAST_ERROR	Boolean	Выполняется сброс выходов с ошибкой.

5.3.2.2 Выходы

Коннектор	Тип данных	Объяснение
RECV_DATA	Любой	Полученные данные. Тип данных автоматически устанавливается в соответствии с типом данных интерфейса, например: structure, string, structure, array.
RECEIVED	Boolean	Этот выход принимает значение TRUE, как только данные получены.
RECVD_LENGTH	Udint	Длина полученных данных в байтах.
SEND_BUFFER_FILLED	Boolean	TRUE, если внутренний буфер передачи данных заполнен, т.е. блоки не могут передавать данные достаточно быстро
LAST_ERROR_CODE	Double word	Последнее сообщение об ошибке как DWORD в шестнадцатеричном представлении.
LAST_ERROR_STRING	String	Последнее сообщение об ошибке как простой текст.

Пример процедуры отправка – получение

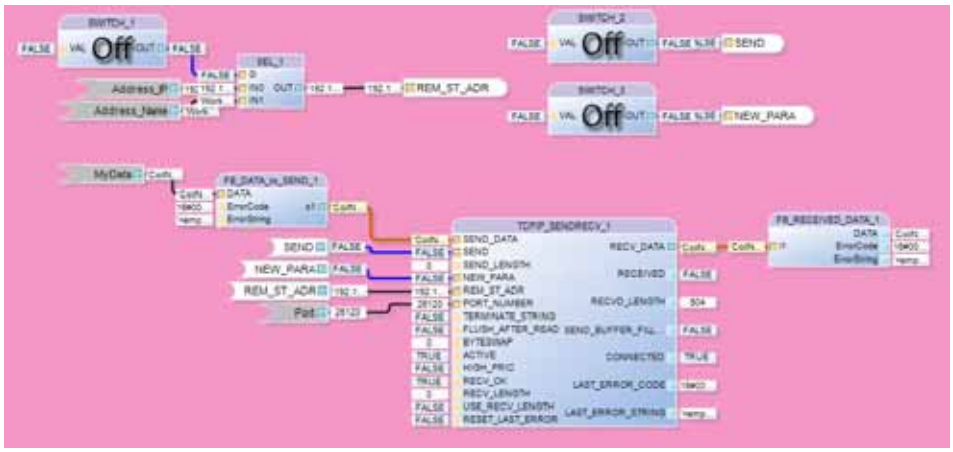


Рис. 46: Пример процедуры отправка - получение

В этом примере данные для передачи представлены как структура. Данные передаются при срабатывании триггера, установленного вручную.

В NEW_PARA есть ключ для восстановления соединения для целей тестирования или в случае изменения параметров соединения.

Этот блок пассивный, он ожидает установления соединения другой стороной.

Полученные данные поступают на RECV_DATA. Вход RECV_OK постоянно установлен на "TRUE", поскольку нет необходимости в промежуточной буферизации данных. Обработка данных может выполняться в пределах сконфигурированного цикла задачи.

5.3.3 PIDT1_CONTROL

Универсальный PIDT1-регулятор может переключаться между режимами P-, I-, PI- или PIDT1-регулятора.

- ❑ Установите начальное значение интегратора
- ❑ Удерживайте моментальное значение интегратора
- ❑ Предварительное значение WP
- ❑ Пределы регулирования LL и LU
- ❑ Пропорциональный коэффициент KP
- ❑ Время сброса TN
- ❑ Переключаемое направление регулирования
- ❑ Индентификация, когда заданные границы достигнуты
- ❑ Отображение сигнала об ошибке
- ❑ Отображение отдельных выходов P-, I- и DT1-регулятора

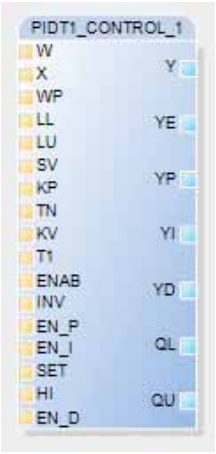


Рис. 47: Функциональный блок PIDT1_CONTROL

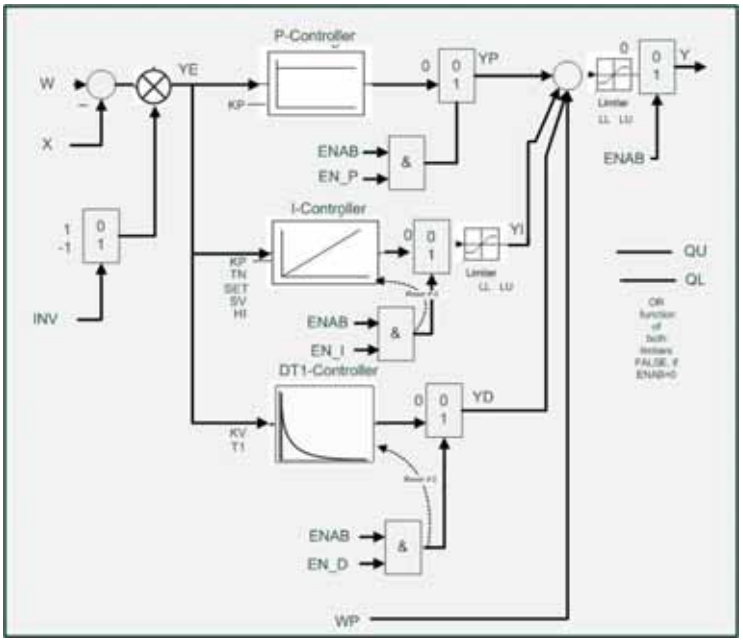


Рис. 48: Функциональная схема универсального PIDT1-регулятора

5.3.3.1 Входы

Коннектор	Тип данных	Объяснение
W	Lreal	Заданное значение
X	Lreal	Текущее значение
WP	Lreal	Предварительная контрольная величина
LL	Lreal	Нижнее предельное значение (действительно для Y и YI)
LU	Lreal	Верхнее предельное значение (действительно для Y и YI)
SV	Lreal	Установленное значение интегратора принимается с помощью SET
KP	Lreal	P-усиление
TN	Time	Сброс времени
KV	Lreal	D-усиление
T1	Time	Временная константа D
ENAB	Boolean	Разблокирование регулятора
INV	Boolean	Инвертирование знака отклонения регулирования
EN_P	Boolean	Активация регулятора P
EN_I	Boolean	Активация регулятора I
SET	Boolean	Установить интегратор значением SV
HI	Boolean	Остановить интегратор
EN_D	Boolean	Активация регулятора D

5.3.3.2 Outputs

Коннектор	Тип данных	Объяснение
Y	Lreal	Контрольная величина = YP+YI+YD+WP
YE	Lreal	Контрольное отклонение = W-X
YP	Lreal	Выходное значение регулятора P = KP*YE
YI	Lreal	Выходное значение регулятора I = Y_{In-1}+KI*YE*Ta/TN
YD	Lreal	Выходное значение регулятора D = α*YD_{n-1}+ α*KV*ΔYE α = 1/ (1+Ta/T1) ΔYE = (YE-YE_{n-1})
QL	Boolean	Достигнуто нижнее предельное значение
QU	Boolean	Достигнуто верхнее предельное значение

5.3.3.3 Подробности / Кривые сигналов

Различные кривые сигналов отдельных компонентов контроллера приведены в качестве примера на нижеследующих схемах.

Контрольное значение Y:

Контрольное значение Y - это сумма компонентов P, I и D, а также предварительной контрольной величины WP.

Если вход ENAB (разблокирование контроллера) не установлен, то контрольное значение всегда равно 0.0.

Вход WP, предварительное контрольное значение:

Этот вход добавляется к выходу Y.

Вход LL/LU, нижнее / верхнее предельное значение:

Примечание

И выход Y, и YI, ограничены.

Выходы QL and QU принимают значение TRUE.

На практике используются следующие регуляторы:

Регулятор PI Регулятор PD Регулятор PID

Компоненты P, I и DT1 будут подробнее рассмотрены ниже.

Компонент P: (параметр: KP, EN_P)

P-компонент контроллера вычисляется как KP*YE. Значение поступает на выход, только когда установлен EN_P.

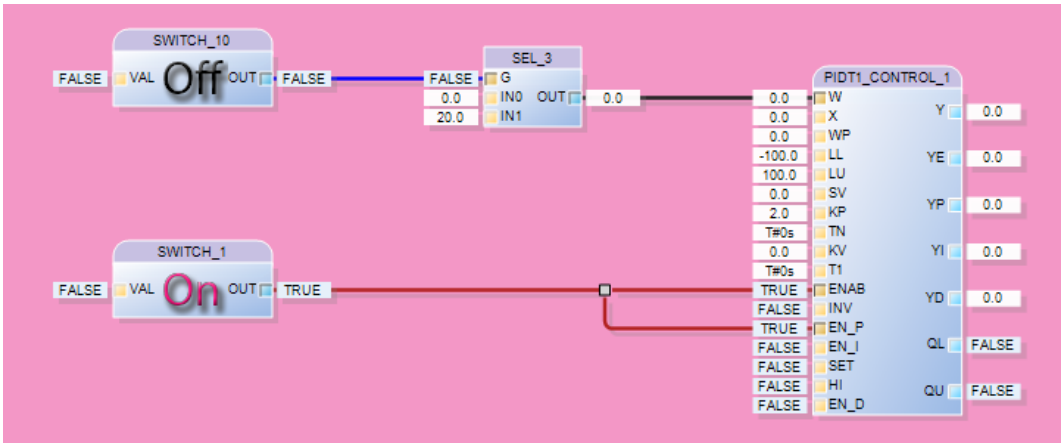


Рис. 49: Компонент P

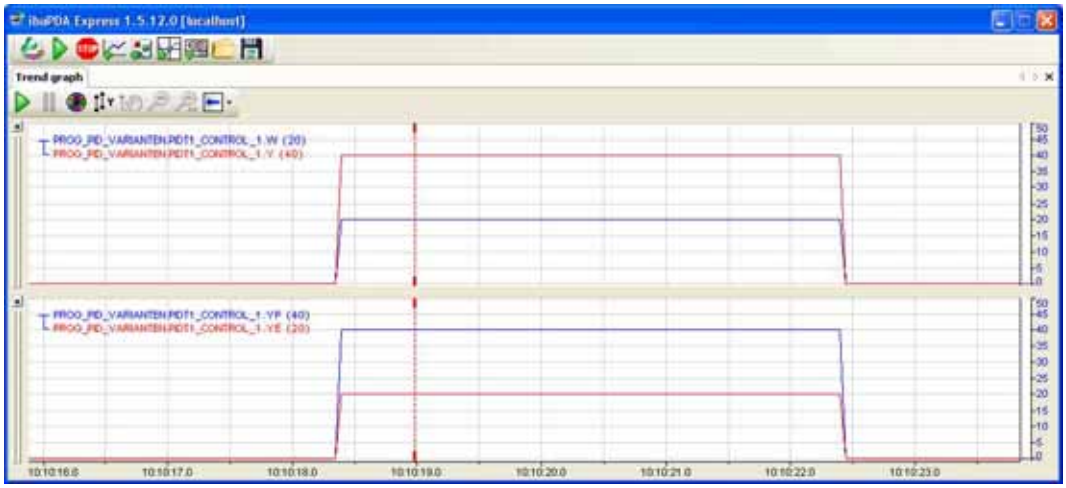


Рис. 50: Компонент P

Компонент I: (параметры: KP, TN, SET, SV, HI и EN_I)

Компонент I вычисляется как
$Y_{In} := Y_{In-1} + KP * YE * Ta/TN$ (Ta = время задачи).
Этот компонент может быть установлен на значение входа SV с помощью входа SET. Значение "TRUE" на входе HI останавливает интегратор. Значение передается на выход, только если установлен EN_I.

Корреляция

TN = KP*Ta = KP/KI

Пример 1

KP = 1.0

TN = 1 с.

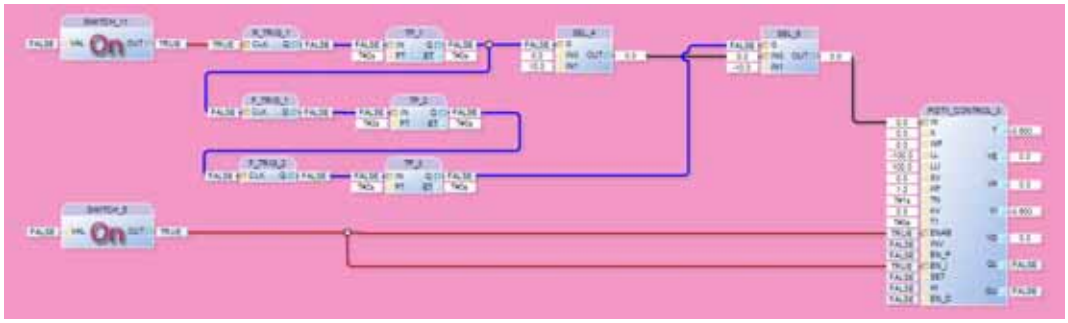


Рис. 51: Компонент I

Рис. 52: Компонент I

Компонент DT1: (параметры KV,T1 и EN_D)

Компонент DT1 контроллера поступает как YD: = $\alpha * YD_{n-1} + \alpha * KV * \Delta YE$

$\alpha = 1/ (1+Ta/T1)$ $\Delta YE = (YE-YE_{n-1})$. (Ta = время задачи)

Значение передается на выход, только если установлен EN_D.

Пример 1

KV = 0.5

T1 = 1 с.

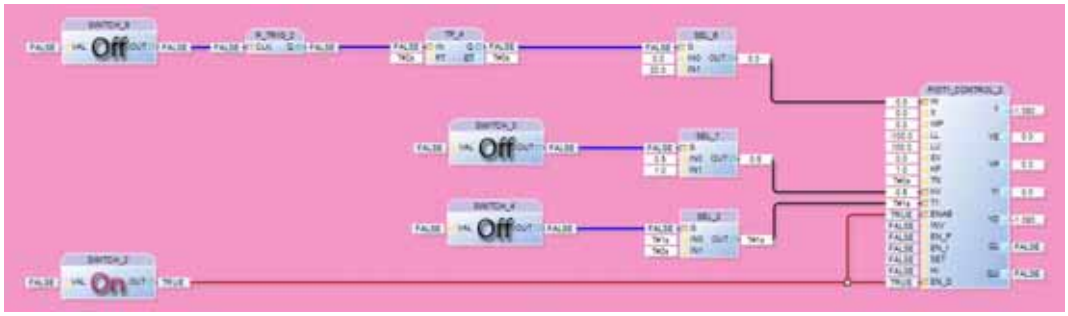


Рис. 53: Компонент DT1

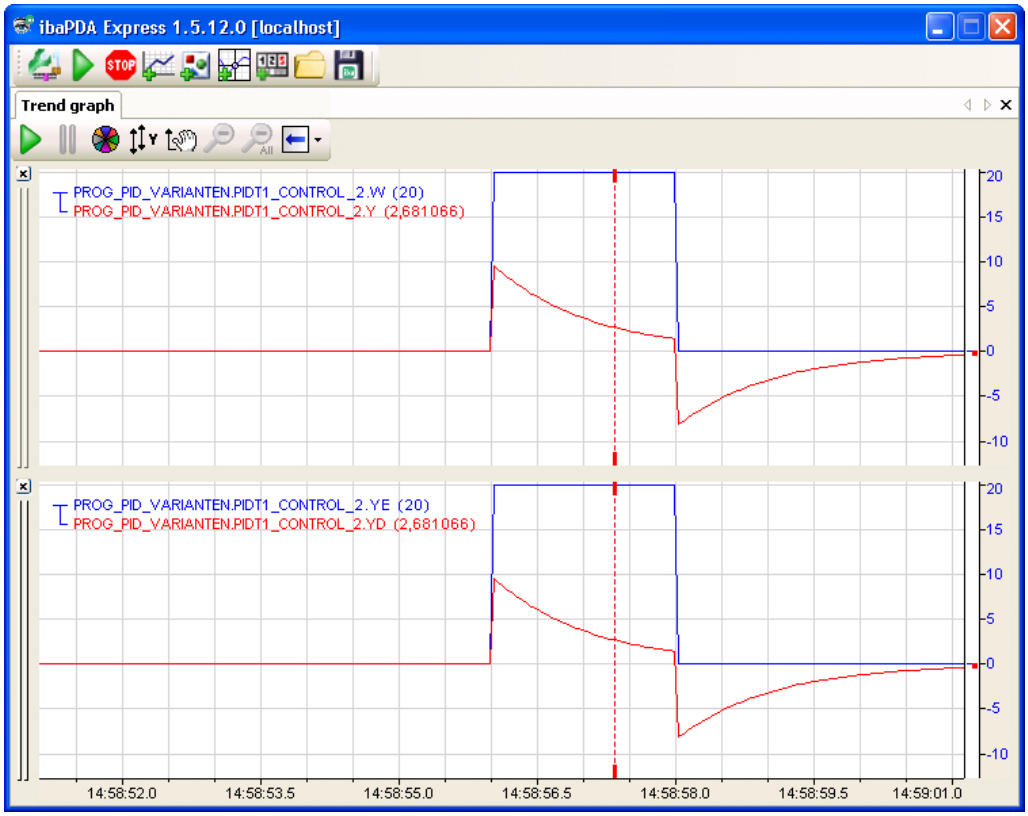


Рис. 54: Компонент DT1

Пример 2

KV = 1

T1 = 2 с.

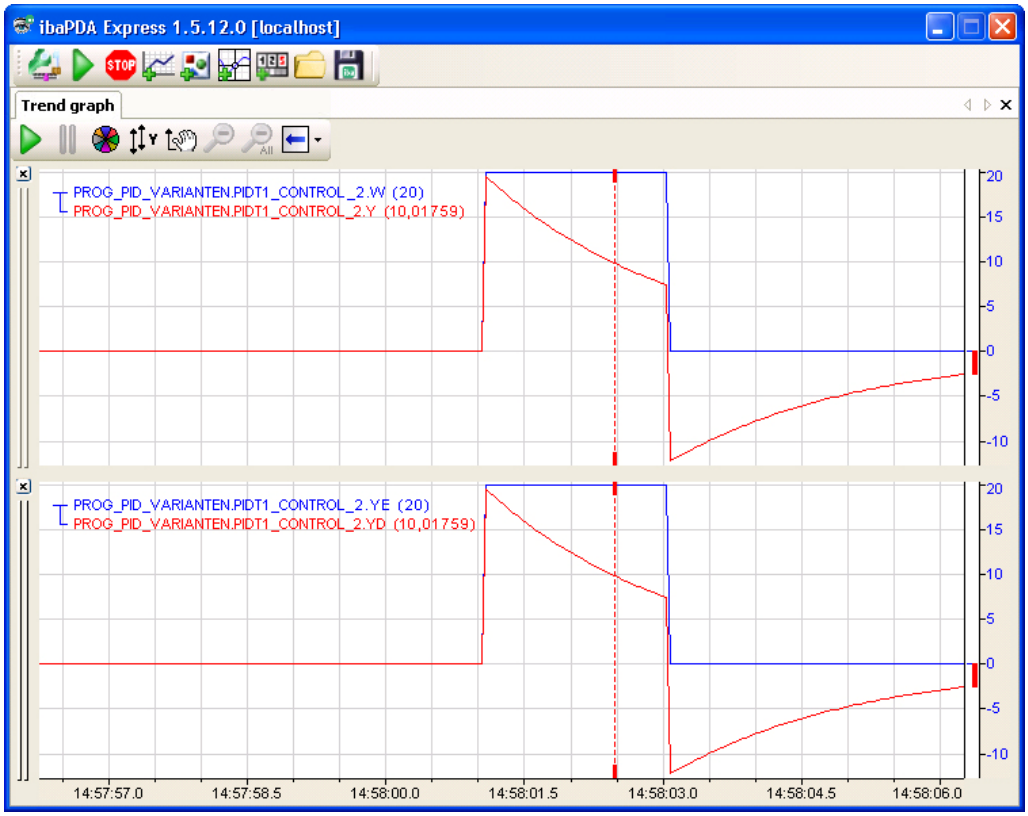


Рис. 55: Компонент DT1

Компонент PIDT1 – суммарный отклик

Пример 1

Пример всего регулятор PIDT1 с кривыми сигналов.

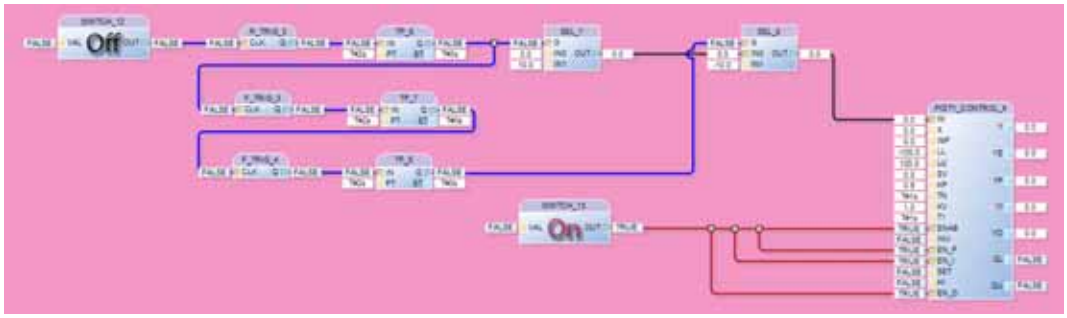


Рис. 56: Регулятор PIDT1 с кривыми сигналов

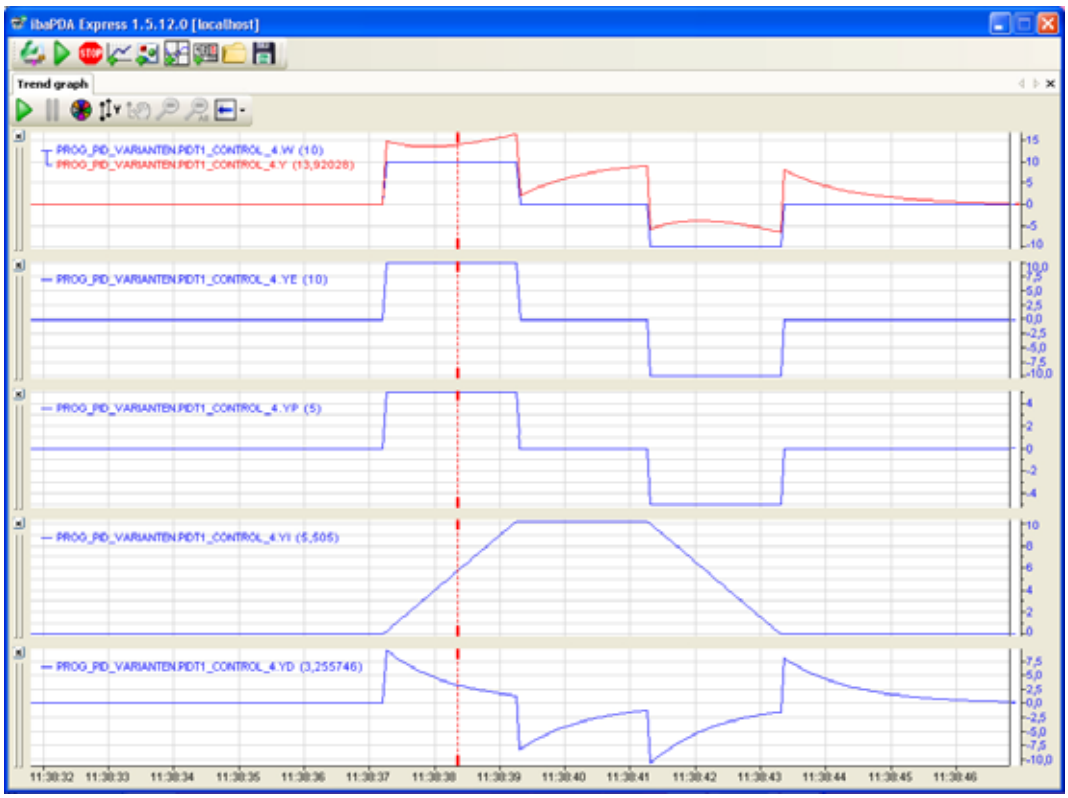


Рис. 57: Регулятор PIDT1 с кривыми сигналов

5.3.4 RAMP

Блок Ramp (блок линейно изменяющегося сигнала) с 2 различными режимами: автоматический и ручной

- Предел заданного значения
- Установка нового начального уровня сигнала
- Установка заданного значения
- Указание на превышение предельных значений

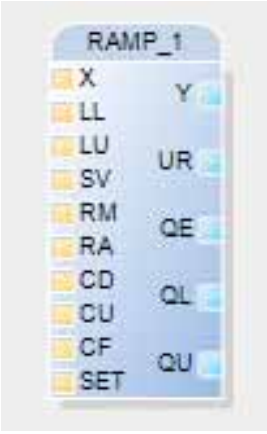


Рис. 58: Функциональный блок RAMP

5.3.4.1 Входы

Коннектор	Тип данных	Значение / Использование
X	Lreal	Входное значение (заданное значение)
LL	Lreal	Нижнее предельное значение
LU	Lreal	Верхнее предельное значение
SV	Lreal	Выход принимает это значение, когда SET = ‘TRUE’
RM	Lreal	Ручной режим (1/c), действителен для CD и CU
RA	Lreal	Автоматический режим (1/c), действителен для CF
CD	Boolean	Падающий сигнал (ручной режим)
CU	Boolean	Нарастающий сигнал (ручной режим)
CF	Boolean	Сигнал в зависимости от входного значения (автоматический режим), имеет приоритет перед CD и CU
SET	Boolean	Передать значение SV на выход

5.3.4.2 Выходы

Коннектор	Тип данных	Значение / Использование
Y	Lreal	Выходное значение; $Y_n = Y_{n-1} + UR$ r = используемый сигнал
UR	Lreal	Используемый сигнал (1/c)
QE	Boolean	Выходное значение = входное значение
QL	Boolean	Достигнуто нижнее предельное значение
QU	Boolean	Достигнуто верхнее предельное значение

5.3.4.3 Пример

Входы CD, CU и CF управляют линейно изменяющимся сигналом. Если ни один вход не активен, то сохраняется последнее выходное значение. В этом случае выход UR отображает скорость изменения сигнала как 0.

Если вход CD активен, то, вне зависимости от входного значения, текущее выходное значение в ручном режиме падает до нижнего предела.

Если вход CU активен, то, вне зависимости от входного значения, текущее выходное значение в ручном режиме нарастает до верхнего предела.

Если CD и CU активны одновременно, UR устанавливается на 0. Выходное значение не меняется.

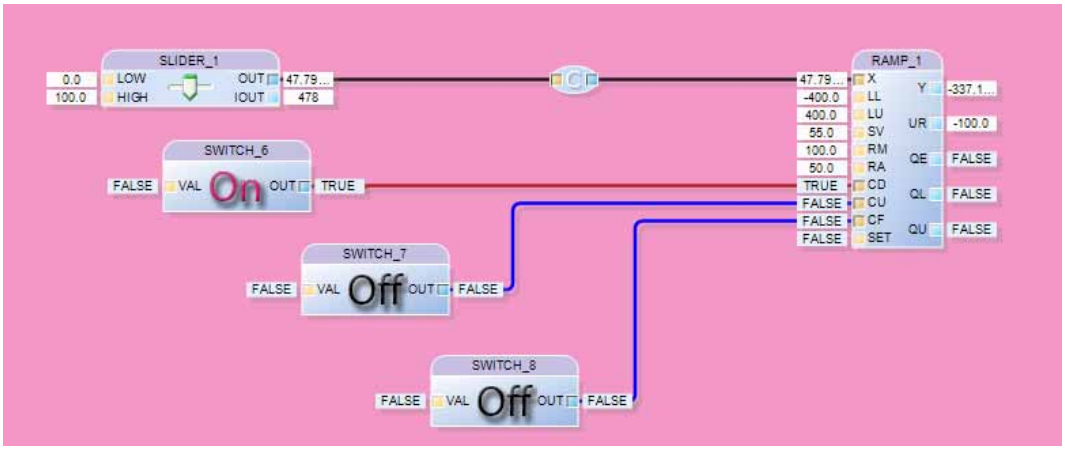


Рис. 59: Управление линейно изменяющимся сигналом

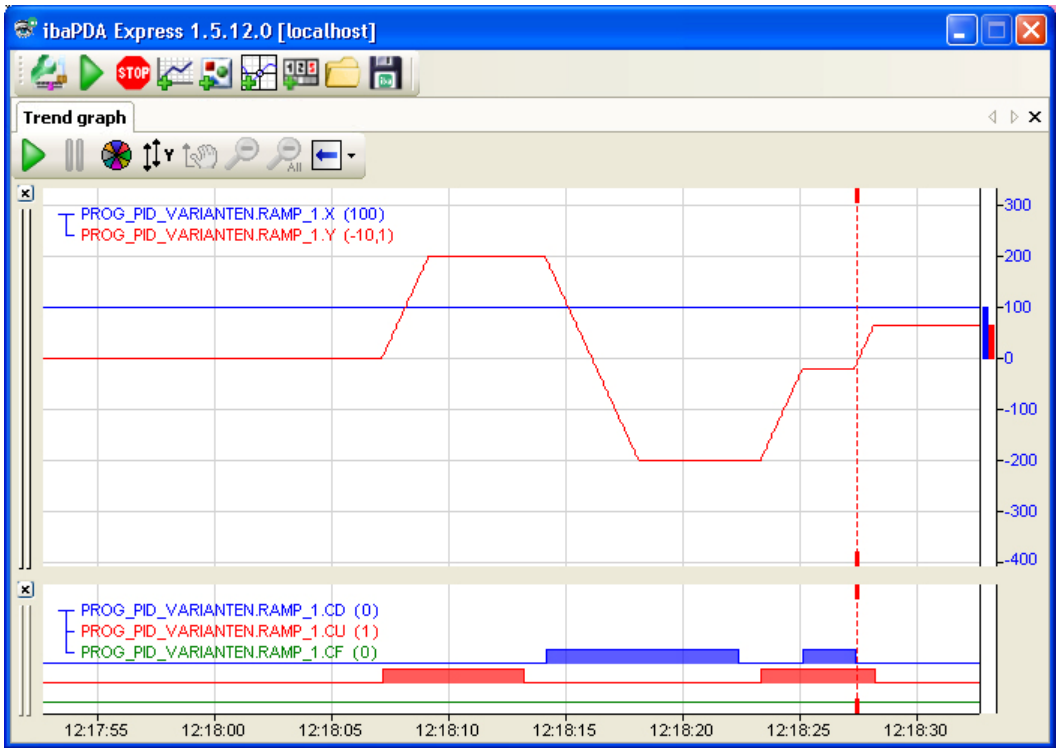


Рис. 60: Управление линейно изменяющимся сигналом

Если вход CF активен, выходное значение стремится достигнуть входного за счет автоматического изменения сигнала. Если входное значение выходит за установленные пределы, то выходное значение изменяется в соответствии с сигналом только до установленных пределов.

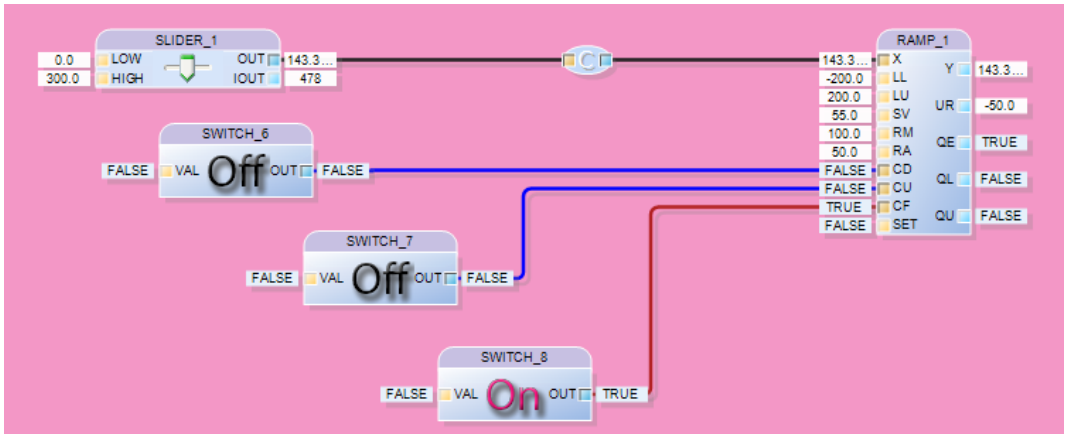


Рис. 61: Управление линейно изменяющимся сигналом

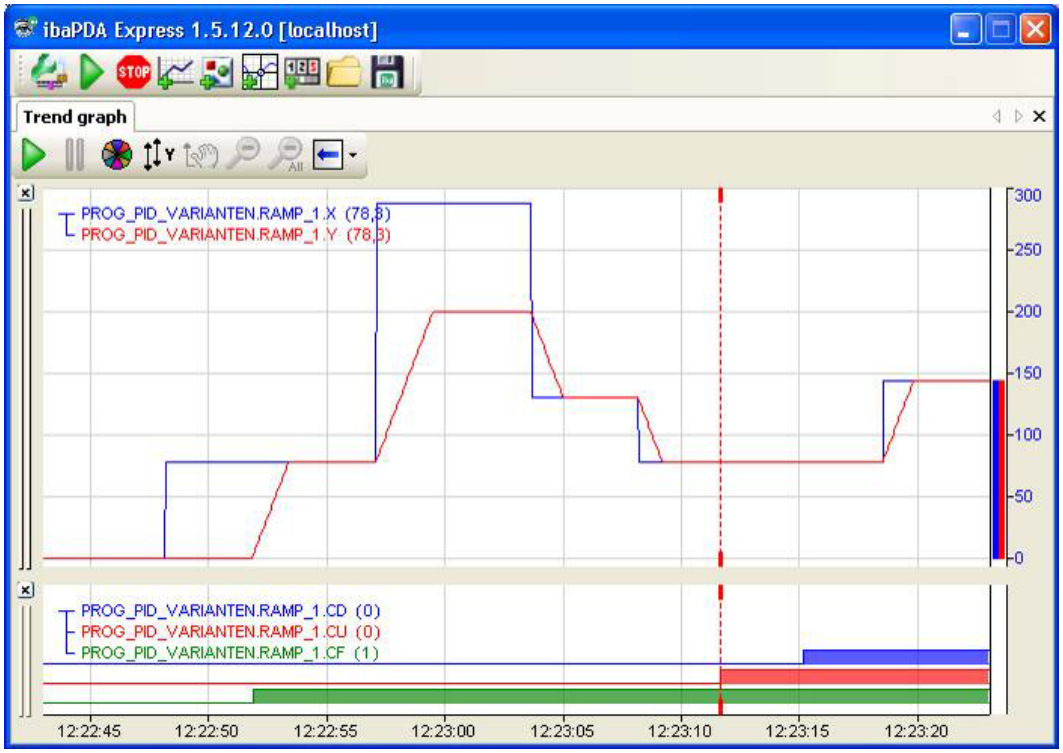


Рис. 62: Управление линейно изменяющимся сигналом

Если CF установлен, то входы CD и CU не имеют эффекта.

5.3.5 FUZZY_CONTROLLER (контроллер нечеткой логики)

Нечеткая логика - это теория, которая была разработана, в первую очередь, для моделирования неоднозначных и нечетких явлений или ситуаций. Эта логика может использоваться, например, чтобы в математических моделях отразить нечеткость таких определений как "чуть-чуть", "довольно-таки" или "значительно".

Использование нечеткой логики особенно необходимо в случаях, когда точное математическое описание процесса невозможно или чрезмерно затруднительно.

Нечеткая логика основывается на нечетких множествах и так называемых функций принадлежности, которые соотносят объекты с нечеткими множествами, над которыми выполняются соответствующие логические операции. На основе этого делаются соответствующие выводы. В случае использования нечеткой логики в технологических процессах необходимы методы преобразования спецификаций и взаимосвязей в понятия нечеткой логики.

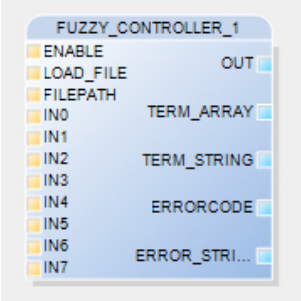


Рис. 63: Функциональный блок FUZZY_CONTROLLER

Для управления с помощью нечеткой логики в ibaLogic реализован функциональный блок FUZZY_CONTROLLER.

Принцип работы

Блок нечеткой логики получает выходное значение на основе набора параметров из связанного с приложением файла с параметрами. С началом вычисления (коннектор ENABLE = TRUE), будут загружены параметры из файла, путь к которому указан в FILEPATH, а вычисления выполняются в том же вычислительном цикле ibaLogic для получения выходного значения для новых загруженных данных. Файл с параметрами может быть перезагружен посредством управляющего сигнала LOAD_FILE даже в процессе выполнения вычислений, при этом вычисление выходных значений или сбор данных в iba-Logic прерываться не будут. Программа может получать выходное значение из макс. 8 входных значений IN0 … IN7. Это выходное значение используется как заданное значение на коннекторе OUT или как управляющая переменная процесса.

5.3.5.1 Входы

Коннектор	Тип данных	Объяснение
ENABLE	Boolean	Запустить процедуру вычислений, когда вход "TRUE". Программа принимает данные из файла с параметрами при появлении переднего фронта сигнала.
LOAD_FILE	Boolean	Принять данные в файле с параметрами при появлении переднего фронта сигнала во время выполнения вычислений.
FILEPATH	String	Указание пути сохранения для данных из файла с параметрами.
IN0..IN7	Lreal	Входные данные, на основе которых вычисляется выходное значение.

5.3.5.2 Выходы

Коннектор	Тип данных	Объяснение
OUT	Lreal	Выходное значение контроллера нечеткой логики; выход = 0,0 в случае ошибки или если enable = "FALSE".
TERM_ARRAY	Lreal (0..8)	Степень ассоциации μ(x) отдельных лингвистических понятий, из которых формируется выходное значение
TERM_STRING	String	Указание текущих лингвистических понятий, из которых формируется выходное значение и процентное значение степени ассоциации.
ERROR_CODE	Integer	Вывод кода ошибки
ERROR_STRING	String	Вывод текста краткого сообщения об ошибке.

Управление приложением

Конфигурирование и пользовательская настройка приложения с помощью программы "ParamFuzzyTool.exe" (предоставляется компанией iba) - это самый надежный способ обеспечить создание файлов без ошибок.

Функциональный блок контроллера нечеткой логики находится в "Функциональные блоки - СПЕЦИАЛЬНЫЕ блоки" ("Function blocks - SPECIALS").

Путь сохранения для файла с параметрами поступает на коннектор FILEPATH как строка. При появлении на коннекторе ENABLE переднего фронта сигнала типа Boolean, программа принимает данные из файла с параметрами и запускает вычисление выходного значения. Пока сигнал активации (Enable) имеет значение "ИСТИНА" ("TRUE"), выходное значение вычисляется с использованием значений типа LREAL на коннекторах IN0 ... IN7. В качестве значения по умолчанию для коннектора ENABLE можно установить "ИСТИНА" ("TRUE"). Если выбрано такое значение по умолчанию, выходное значение будет вычисляться постоянно.

Коннектор сигнала LOAD_FILE с типом данных Boolean на функциональном блоке контроллера нечеткой логики позволяет загрузить новый файл с параметрами в процессе выполнения вычислений. При появлении переднего фронта сигнала LOAD_FILE данные в блоке нечеткой логики принимаются программой в качестве нового набора параметров, которые являются основой для вычисления выходного значения.

Совет

Компания iba рекомендует пользователям, которые будут работать с программой, ознакомиться с примером, который приводится на диске, поставляемом с программой. На этом CD диске также содержится инструмент для конфигурирования и пользовательской настройки приложения: "ParamFuzzyTool.exe".

5.4 Пользовательские функциональные блоки

В ibaLogic содержится библиотека стандартных функциональных блоков. Тем не менее, у пользователя может возникнуть потребность определить собственные блоки, чтобы обеспечить более эффективное решение конкретной задачи. Существует три типа пользовательских функциональных блоков:

- Функциональные блоки, создаваемые в ibaLogic с помощью высокоуровневого языка программирования - структурированного текста (ST).
- Функциональные блоки, создаваемые в ibaLogic с помощью графического программирования. Далее в тексте руководства такие блоки будут называться "макросами".
- Функциональные блоки, создаваемые не в ibaLogic с помощью высокоуровневого языка (C++ или FORTRAN), которые затем интегрируются в ibaLogic как DLL.

После того как блоки добавлены, программа рассматривает их аналогично стандартным блокам.

Функциональные блоки создаются так же, как макросы. Содержимое кодируется по-разному. Помимо этого, макросы можно создавать автоматически, комбинируя существующие блоки.

5.4.1 Функциональные блоки

Для того чтобы создать новый функциональный блок, установите курсор мыши на свободное пространство в окне программы и выберите в контекстном меню "Новый - Новый функциональный блок..." ("New - New Function Block…"). Появится окно с заголовком "Создать функциональный блок" ("Create Function Block").

В верхней части диалогового окна находятся поле для ввода имени блока и таблица для определения входов, выходов и внутренних переменных.

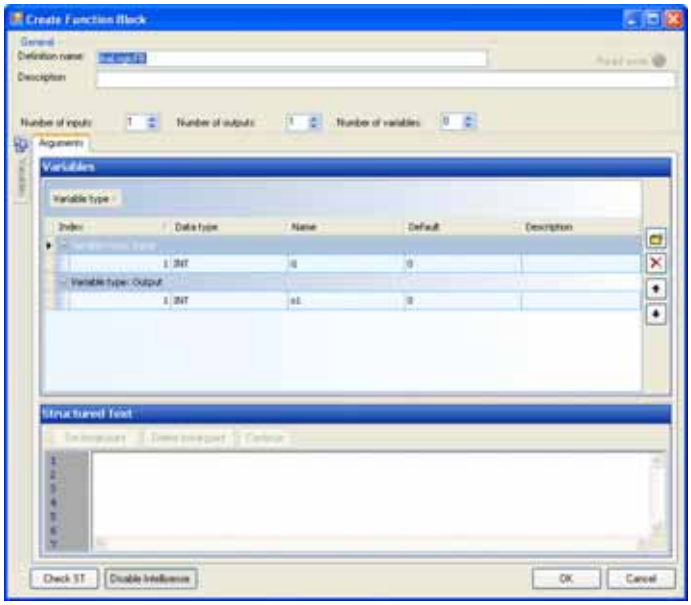


Рис. 64: Диалоговое окно "Создать функциональный блок"

5.4.1.1 Общие настройки

Имя определения

Имя определения - это имя, под которым блок сохраняется в папке с функциональными блоками. Имя экземпляра создается на основе этого имени с добавлением индекса.

Примечание касательно разницы между определением и экземпляром

Более подробная информация содержится в пункте 4.2.3.2, "Экземпляры".

Примечание

Компания iba рекомендует использовать разные префиксы для функциональных блоков и макросов, например для блоков: "FB_", а для макросов: "MB_". Сконфигурировать параметры можно в меню "Инструменты – Опции – Функциональные блоки".

Аналогичным образом, пользователь может найти значения по умолчанию для имен и типов данных переменных. iba рекомендует использовать имена "i", "o", "io" и "v". Пользователь может сконфигурировать данные, исходя из собственных потребностей.

Имя экземпляра

В процессе создания экземпляра его имя не отображается. Имя отображается, только при использовании блока в проекте.

Описание

В этом поле пользователь может ввести дополнительную информацию о функциональном блоке. Это описание отображается как подсказка при наведении курсора на имя функционального блока в библиотеке.

Количество входов / выходов / переменных

Здесь можно указать количество использованных переменных. Для каждой переменной в таблице создается отдельная строка.

Переменные

Список переменных отображается как в виде дерева элементов, так и в виде таблицы.

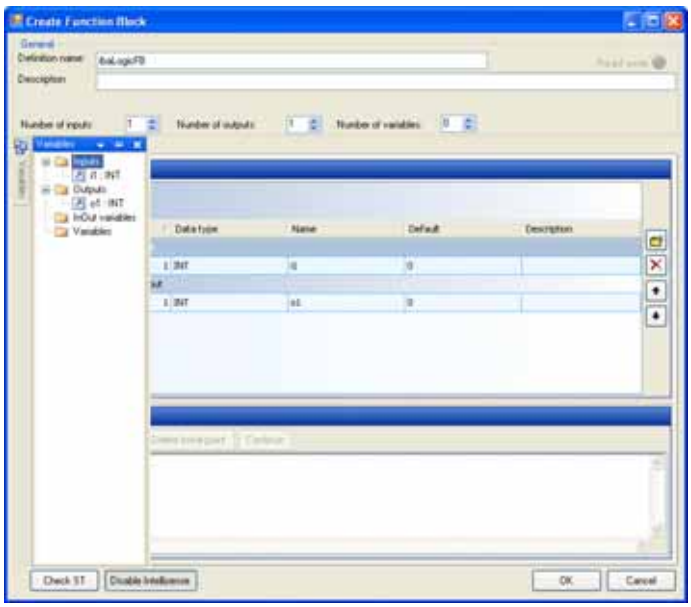


Рис. 65: Диалоговое окно "Создать функциональный блок" со списком переменных

➲ Открыть дерево переменных можно щелчком по вкладке "Переменные" в левой части таблицы.

Дерево элементов скрыто, поскольку конфигурирование и пользовательская настройка переменных выполняются только в таблице.

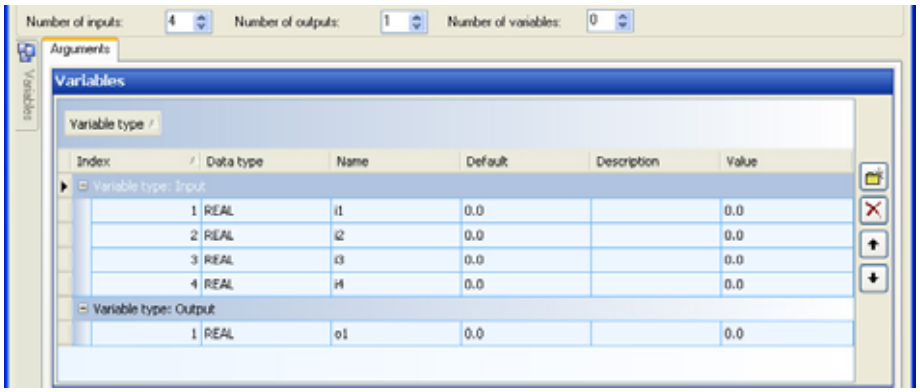


Рис. 66: Переменные функционального блока

Переменная

Столбец	Объяснение
Индекс	Каждый тип переменной начинается с индекса 1.
Тип данных	Выбор типа переменной. Здесь пользователь может также создать новый тип переменной. Значения по умолчанию содержатся в меню "Инструменты - Опции" ("Extras - Options").
Имя	По умолчанию имя состоит из префикса и индекса, однако пользователь может присвоить переменной другое имя.
Стандартное значение	Представление значений зависит от выбранного типа данных. Более подробная информация содержится в пункте 5.4.2.2, "Описание синтаксиса структурированного текста ". Переменной присваивается значение в том случае, если отсутствует соединение с ней (для входных переменных) или соответствующее значение не присвоено в коде блока.
Описание	Текст, который появляется в качестве подсказки при наведении курсора на имя блока.
Значение	В случае массивов и структур отображается текущее значение только первого элемента переменной (только в режиме онлайн).

Столбец	Объяснение
	Совет Это значение можно изменить вручную, но оно пересчитывается в следующем цикле и перезаписывается, если необходимо. **Важно** При удалении соединения с входным коннектором в режиме онлайн сохраняется последнее значение.

С помощью кнопок в правой части таблицы можно изменять последовательность переменных, а также удалять или добавлять новые переменные.

Значок	Объяснение
	Добавить элемент.
	Удалить элемент.
	Поменять элементы местами.

5.4.2 Редактор структурированного текста

В редакторе структурированного текста пользователь может назначить функции для функционального блока.

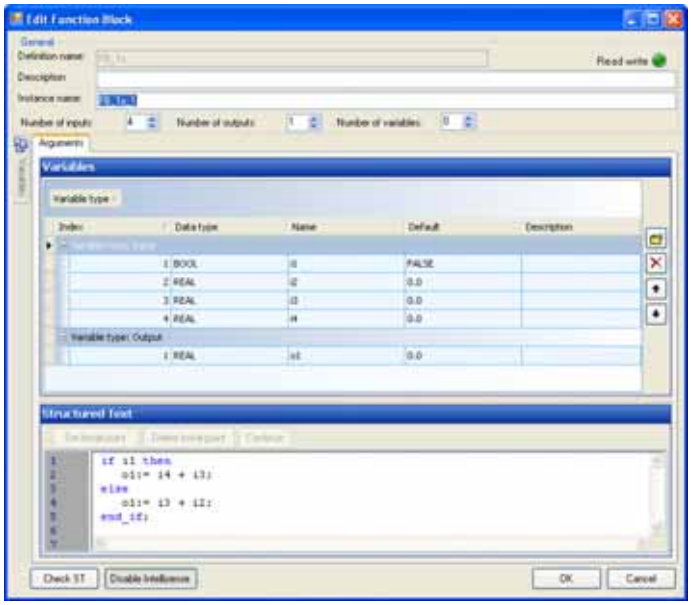


Рис. 67: Редактор структурированного текста

Над полем ввода текста и под ним располагаются следующие кнопки:

Кнопка	Объяснение
Установить точку останова (активно только в режиме онлайн)	При щелчке по этой кнопке исполнение программного кода будет остановлено в точке, где установлен курсор.
Удалить точку останова (активно только в режиме онлайн)	Используется для удаления точки останова в месте, где установлен курсор.
Продолжить (активно только в режиме онлайн)	Выполнение программы будет продолжено. Выполнение программы будет прервано на точке останова в следующем цикле.
Проверка ST	Проверка синтаксиса введенного кода без его компиляции
Активировать / деактивировать Intellisense	Активация или деактивация ресурса Intellisense

⚠ DANGER

Опасность использования перечисленных функций в режиме онлайн

Мы настоятельно рекомендуем не применять эти функции (установить точку останова, удалить точку останова, продолжить), если выходы используются для управления в онлайн-режиме ibaLogic, поскольку существует риск получить травму или нанести ущерб имуществу (см. пункт 11.2 "Время отклика").

5.4.2.1 IntelliSense

IntelliSense - это ресурс для автоматического заполнения элементов. Он предоставляет дополнительную информацию и опции для выбора, что облегчает заполнение элементов данных.

В частности, это значительно упрощает работу со структурами, поскольку после введения разделителя "." сразу перечисляются все элементы для выбора.

Рис. 68: Редактор структурированного текста с активированным IntelliSense

Пример: Окно выбора IntelliSense вызывается при вводе "IF". Для ввода выделенного элемента щелкните <Return>.

Выражения IF..THEN..ELSE / WHILE…/ REPEAT… появляются только после того, как пользователь введет первое слово. Эти выражения могут использоваться как фреймворк.

Чтобы сделать выбор, нажмите <Курсор вверх> (<Cursor up>) или <Курсор вниз> (<Cursor down>), чтобы подтвердить - <Tab>.

5.4.2.2 Описание синтаксиса структурированного текста

Структурированный текст может выглядеть, например, следующим образом:

```
1  (* Difference between i2 and i1 *)
2  Difference := i2 - i1;
3
4  (* Mean value calculation *)
5  Mean_value := (i1 + i2) / 2.0;
```

Рис. 69: Синтаксис структурированного текста

Обозначения:

- Комментарии заключаются в "(*" и "*)".
- В конце выражения ставится точка с запятой.
- Полученный результат записывается слева от ":=".
- Выражения состоят из операторов и операндов.

Важно

Помимо операторов и выражений, описанных ниже, пользователь может также вызывать некоторые функции, которые доступны в виде блока, даже если они не включены в ST. Примечания касательно возможности и способа использования этих функций в ST вы найдете в описании функций в разделе D.13 "Стандартные функциональные блоки". В этом разделе для каждого блока с ключевым словом "ST" содержится примечание касательно использования этого блока в структурированном тексте.

Операторы

Список операторов в порядке приоритетности:

Оператор	Пример	Значение в примере	Описание	Приоритет
()	`(2+3) * (4+5)`	45	Скобки	Самый высокий
**	`3**4`	81	Возведение в степень	
-	`-10`	-10	Отрицательная величина	
NOT		NOT TRUE	Логическое отрицание	
*	`10*3`	30	Умножение	
/	`6/2`	3	Деление	
MOD	`17 MOD 10`	7	Деление по модулю (остаток от деления)	
+	`2+3`	5	Сложение	
-	`4-2`	2	Вычитание	
<, >, <=, >=	`4 > 12`	FALSE	Сравнение	
=	`T#26h = T#1d2h`	TRUE	Равенство	
<>	`8 <> 16`	TRUE	Неравенство	
&, AND	`TRUE & FALSE`	FALSE	Boolean AND	
XOR	`TRUE XOR FALSE`	TRUE	Boolean Exclu-sive Or	
OR	`TRUE OR FALSE`	TRUE	Boolean Or	Самый низкий

Выражения

Ключевое слово	Пример	Описание
;	`;`	Пустое выражение
:=	`Var1 := 12;`	Присвоение значения 12 имени переменной в левой части выражения.
f(i1, i2, …)	`o1 := concat(iDir, iFile, v1);`	Вызов функции, см. также описание функциональных блоков

Ключевое слово	Пример	Описание

IF	`IF i1 < i2 THEN`
	`  o1 := 1;`
	`[ ELSIF i1 =i2 THEN`
	`  o1 := 2; ]`
	`ELSE`
	`  o1 := 3;`
	`END_IF;`
		Выражение, содержащее условие.
		Условием является логическое выражение (которое возвращает результат "ИСТИНА" или "ЛОЖЬ")
		Возможные расширения приводятся в квадратных скобках […].

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;`
		Выражение с оператором Case.
		Аргумент "i1" является выражением типа ANY_INT или ENUM.
		На один оператор case приходится одно или несколько выражений.
		Оператор case может содержать несколько целых чисел (3,4,5) или диапазонов целых чисел (10…15). Ветвь ELSE является опциональной.
		Возможные расширения приводятся в квадратных скобках […].

FOR	`FLAG := FALSE;`
	`FOR ix:= 1 TO 100 [ BY 2 ] DO`
	`  IF o1[ix] = iy THEN`
	`  FLAG := TRUE;`
	`  EXIT;`
	`  END_IF;`
	`END_FOR;`
	`IF FLAG THEN (* found *)`
		Цикл без условий (итерация).
		Шаг BY xx указывать не обязательно. Если шаг не указан, то по умолчанию используется шаг равный 1.
		Тип переменной внутреннего цикла: ANY_INT, в пределах цикла его не следует изменять.
		Возможные расширения приводятся в квадратных скобках […].
		Внимание
		Существует риск создать бесконечный цикл

WHILE	`WHILE i1 > 1 DO`
	`  o1 := o1/2;`
	`END_WHILE;`
		Цикл с условием
		Внимание
		Существует риск создать бесконечный цикл

Ключевое слово	Пример	Описание

REPEAT	REPEAT
	  o1:= o1 * i1;
	UNTIL o1 > 10000
	END_REPEAT;
		Цикл с условием
		Отличие от WHILE: цикл выполняется мин. один раз, даже если условие не соблюдено в начале цикла.
		Внимание
		Существует риск создать бесконечный цикл

EXIT	FLAG := FALSE;
	FOR ix:= 1 TO 100 [BY 2] DO
	  IF o1[ix] = iy THEN
	  FLAG := TRUE;
	  EXIT;
	  END_IF;
	END_FOR;
	IF FLAG THEN (* found *)
		Досрочное завершение цикла FOR, WHILE или REPEAT.
		Первое выражение выполняется после следующего завершения цикла, т.е. в случае вложенных циклов выполнение продолжается на следующем, более высоком, уровне.
		Возможные расширения приводятся в квадратных скобках […].

RETURN	oFLAG := FALSE;
	FOR ix:= 1 TO 100 [BY 2] DO
	  IF o1[ix] = iy THEN
	  oFLAG := TRUE;
	  RETURN;
	  END_IF;
	END_FOR;
		Выражение Return, досрочное завершение функционального блока.
		Пример: если значение iy содержится в массиве ix, то результат TRUE, иначе - FALSE.
		Возможные расширения приводятся в квадратных скобках […].

ARRAY, доступ	<ArrayType>[index,…]
	
	o1 := iArray[0];
	o1 := iArray[0,0,…];
	
	o1 := iArray[0][0];
		Индексы приводятся в квадратных скобках […].
		Доступ к одномерному массиву
		Доступ к n-мерному массиву
		Доступ к вложенному массиву
		Более подробная информация содержится в пункте 5.5.8.5 "Группа ARRAY TYPE (массив)".
		Использование выражений в качестве индексов недопустимо, например: MyArray[v1+1]

Ключевое слово	Пример	Описание

ENUM, доступ	<EnumType>_Enumerator
	
	IF (i1 > 0) THEN
	v1 := Switch_forward;
	ELSIF (i1 < 0) THEN
	v1 := Switch_back;
	ELSE
	v1 := Switch_stop;
	END_IF;
		Перечисляемые типы разделяются символом "_" - Enumtype.
		Пример:
		"Switch" - это тип ENUM, "Forward", "Stop" и "Back" - это нумераторы.
		Тип переменной v1 - "Switch".
		Более подробная информация содержится в пункте 5.5.8.4, "Группа ENUM TYPE".

Structure, доступ	<STRUCTURE NAME>.ELEMENT
	
	o1.Temperature := i1;
	o1.Speed := i2;
		Элементы структуры отделяются друг от друга с помощью символа ".".
		Пример:
		"o1" - это переменная типа "структура".
		"Температура" и "скорость" - элементы структуры.
		Более подробная информация содержится в пункте 5.5.8.6, "Группа STRUCT TYPE (структуры)".

Константы

Описание	Пример
Целые числа и битовые строки (кроме BOOL)	-12 0 123_456 +986
Десятичное представление	
Двоичное представление	2#1111_1111 (десятичное: 255) 2#1110_0000 (десятичное: 240)
Шестнадцатеричное представление	16#FF or 16#ff 16#00F0_FFE0
Real	-12.0 0.0 0.4560 3.14159_26
Real с экспонентой	-1.34E-12 или -1.34e-12 1.0E+6 или 1.0e+6 1.234E6 или 1.234e6
BOOL	0 или FALSE 1 или TRUE
Константы времени	Идентификация типа: "T#" , Обозначение времени: "д" (день), "ч" (час), "м" (минута), "с" (секунда) и "мс" (миллисекунда). T#12d12h17m42s T#16d_2h_5m
Для всех обозначений:	Допустимо использовать символ подчеркивания для лучшего зрительного восприятия обозначения времени.

Строки (Strings)

Строки заключаются в одиночные кавычки.

Знак $ и следующее за ним шестнадцатеричное число интерпретируются как код ASCII.

Примечание

Стандарт IEC также допускает использование двойных кавычек. Такие WSTRING в настоящее время не реализованы.

Специальные символы, использование которых допустимо в строках

Сочетание символов	Интерпретация при распечатке
$$	Знак доллара
$'	Одиночная кавычка
$L или $l	Перевод строки (LF)
$N или $n	Новая строка (NL)
$P или $p	Новая страница
$R или $r	Возврат каретки (CR)
$T или $t	Табуляция

Примечание

Сочетание символов RL (возврат каретки / перевод строки) равнозначен $N (новая строка), соответственно, такое сочетание автоматически отображается в функциональном блоке во время обработки как $N (см. элемент в STANDARD VALUE и отображение под VALUE).

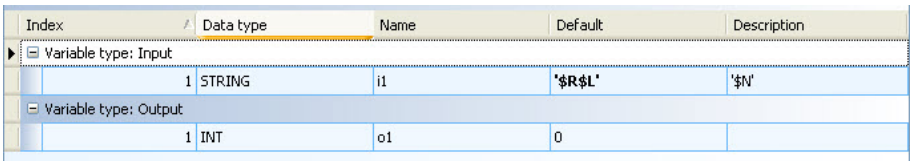


Рис. 70: Ввод переменной

Характеристики и примеры строк

Пример	Объяснение
''	Пустая строка (Длина = 0)
'A'	Стока длиной в один символ, содержит символ A
' '	Стока длиной в один символ, содержит пробел
'$''	Стока длиной в один символ, содержит одиночную кавычку
'"'	Стока длиной в один символ, содержит двойную кавычку
'RL'	Стока длиной в два символа, содержит символы ASCII для CR и LF
'$$1.00'	Стока длиной в пять символов, содержит "$1.00"
'ÄË' '$C4$CB'	Эти строки идентичны. Первая строка содержит символы ASCII, а вторая - соответствующий шестнадцатеричный код.

5.4.3 Макросы

Макросы используются для комбинирования функций и, соответственно, достижения более четкой структуры программы.

Свойства:

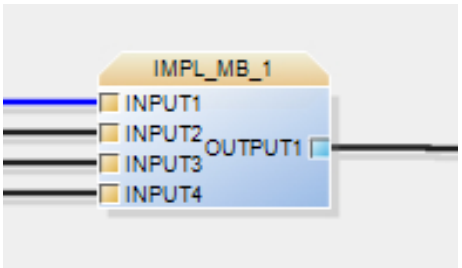
- Макросы можно экспортировать.
- Макросы можно копировать в глобальную папку и использовать несколько раз, а также в других проектах.
- Макросы могут содержать другие макросы и, естественно, пользовательские функциональные блоки.
- В макросах не допускается использовать коннекторы вне задачи (OTC), переключатели и слайдеры, однако внутристраничные коннекторы (IPC) использовать можно. Связь с другими компонентами программы возможна только посредством входных и выходных коннекторов.
- Ресурсы ввода и вывода аппаратного обеспечения непосредственно в макросах использоваться не могут.
- Уже созданный макрос можно дополнить, если его "раскрыть", чтобы блоки, которые он содержит, отобразились на более высоком уровне.

5.4.3.1 Создание макроса

Макросы создаются вручную, аналогично функциональным блокам.

Последовательность действий

1. Установите курсор мыши на свободное пространство в окне программы и в контекстном меню выберите "Новый... Новый макрос" ("New… - New Macro Block…"). Появится диалоговое окно для ввода имен и переменных блока. Более подробная информация содержится в пункте 5.4.1 "Функциональные блоки".
2. Для того чтобы закрыть диалоговое окно, щелкните <OK>. Появится пустой макрос.



3. Щелкните двойным щелчком по макросу, чтобы открыть внутренний графический интерфейс программирования.
4. В пределах редактируемого макроса пользователь может разместить другие макросы и функциональные блоки и организовать их таким образом, чтобы получить нужную функциональность.

Пример

В этом примере программа постоянно следит за изменением целого значения входа, учитывает каждое изменение и передает на выход.

Рис. 71: Мониторинг целого значения

Как и в любой другой программе, для сохранения содержимого макроса в области программирования создается новая папка, которой присваивается имя макроса.

Обратите внимание на контекст вычислений. При щелчке вы перейдете на вызывающий уровень. Более подробная информация содержится в пункте 4.2.5 "Расположение вкладок и окон программирования".

5.4.3.2 Как открыть макрос

Последовательность действий

➲ Чтобы открыть макрос, щелкните двойным щелчком по его экземпляру в программе или в проводнике по рабочим областям.

5.4.3.3 Объединение существующих компонентов в макрос

В ibaLogic есть опция объединения нескольких уже существующих блоков в один макрос.

1. Для этого выберите блоки, которые нужно объединить, как указано на скриншоте ниже.

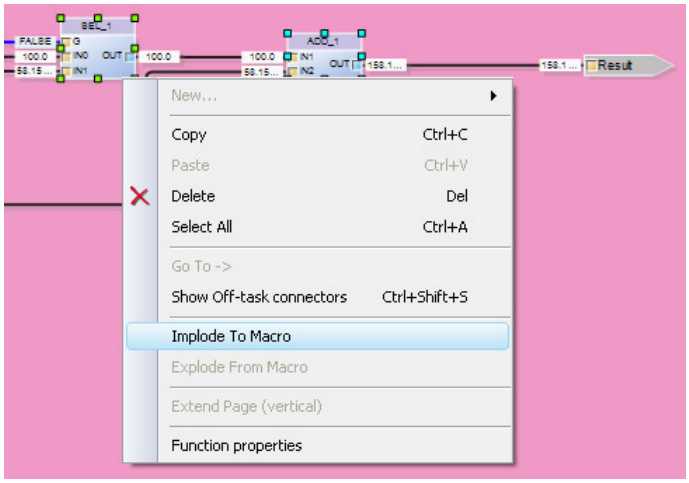
Примечание

Обратите внимание на то, что

- ❑ что при этом выделяются также соответствующие соединения между блоками.
- ❑ коннекторы вне задачи (OTC - Off-task connectors) не выделены.
- ❑ соединения, исходные или целевые блоки которых не выбраны, становятся входом или выходом макроса.

2. Откройте контекстное меню, используя один из выбранных элементов.

3. Выберите пункт меню "Объединить в макрос" ("Implode To Macro"). Появится диалоговое окно "Редактировать функциональный блок" ("Edit Function Block").



4. Присвойте корректные имена новому макросу, а также его входам и выходам. Имена должны соответствовать требованиям стандарта IEC.

Совет

Эти изменения также можно внести впоследствии, щелкнув по созданному макросу и выбрав его свойства.

Результат

В результате получится новый макрос (IMPL_MB_1), свойства которого совпадают со свойствами ранее выбранных блоков.

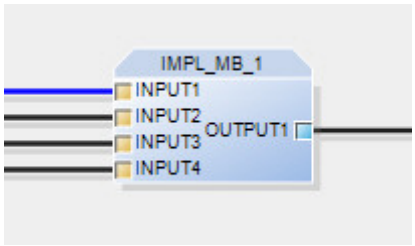


Рис. 72: Макрос (IMPL_MB_1)

5. Пользователь также может открыть функциональный блок двойным щелчком и продолжить редактирование графических элементов.

В режиме онлайн в соответствующих областях можно также посмотреть текущие значения (в зависимости от контекста вычисления).

Дополнительная информация касательно контекста вычисления содержится в пункте 4.2.5 "Расположение вкладок и окон программирования".

5.4.3.4 Расширение макроса

Существующий макрос можно дополнительно расширить. При этом уже заданные блоки помещаются на более высокий уровень.

Последовательность действий

6. Выделите макрос.

7. В контекстном меню выберите "Расширить макрос" ("Expand From Macro").

Пример

Есть простой макрос, содержащий внутренний сумматор, который суммирует оба входа. Этот макрос необходимо расширить.

Результат

После расширения макроса мы получили следующую схему:

- ❑ Отображается сумматор.
- ❑ Исходная структура макроса тем не менее по-прежнему сохраняется в библиотеке блоков.

Рис. 73: Макрос MB_Set-point_1 ➔ Расширенное соединение

5.4.4 Создание собственных библиотек DLL

Создание макросов и функциональных блоков с помощью структурированного текста представляет собой очень простой способ обеспечить выполнение различных задач в области автоматизации. Эти блоки не только легко создавать, их можно также копировать и разобраться в их содержании, т.е. функциях.

Однако если, например, речь идет о высокоинтеллектуальном процессно-ориентированном технологическом решении, нужно не допустить неконтролируемого распространения этого ноу-хау.

В таком случае очень полезно иметь возможность создавать собственные библиотеки DLL, которые содержат данные в скомпилированном виде, что не позволяет легко их извлечь.

Этот принцип также позволяет реализовать специальные соединения для:

- создания сложных блоков.
- работы в среде Windows.
- выполнения каждой задачи в своем собственном потоке.

При определенных условиях существует также возможность интеграции другого высокоуровневого языка.

Открытый интерфейс позволяет пользователю интегрировать DLL в ibaLogic.

Этот файл DLL отображается в программе как обычный функциональный блок, у которого есть имя, входы и выходы. DLL отличается от функционального блока ST только тем, что его исходный код невозможно посмотреть.

Дополнительная документация

В настоящем разделе содержится только краткий обзор использования DLL, подробное описание можно найти на CD диске, который входит в объем поставки (например, руководство "Создание DLL" ("Creating a DLL") в папке iba-Logic_390l\DLL Development\Sample folder).

Чтобы получить эту информацию, направьте запрос компании iba.

Примечание

Для использования библиотек DLL необходима лицензия. Если нет соответствующего аппаратного ключа, то DLL работать не будут.

Компилятор

Программа поддерживает все файлы DLL, написанные на C++ или Fortran.

Специалистами компании были протестированы следующие компиляторы на предмет соответствия для создания и компиляции файлов DLL:

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

Однако, существует разница при создании DLL для использования на ПК (WinXP) или PADU-S-IT. Чтобы использовать DLL на платформе PADU-S-IT, их нужно компилировать для Windows-CE.

5.4.4.1 Необходимые исходные файлы и описания

Для создания DLL необходимы следующие исходные файлы и описания, содержащиеся на установочном CD диске ibaLogic:

Описания:

- Руководство для создания DLL на C++
- Руководство для создания DLL на Fortran

Файлы: (точные названия файлов см. в описаниях)

- Файл фреймворка:
  Содержит процедуры и "тело" DLL; пользователь может добавлять входы или выходы, а также изменять процедуры: InitEvaluation, Evaluate и ExitEvaluation. Это возможно как в C++, так и в Fortran.
- Другие файлы в зависимости от языка:
  Присвоение процедур и номеров DLL, определение интерфейсов и т.д. Здесь внесение изменений не требуется.

5.4.4.2 Необходимые условия и примечания

При создании DLL нужно учитывать следующее:

- Время исполнения DLL увеличивает время исполнения задач, в которых они вызываются.
- iba рекомендует перенести времязатратные функции в треды (потоки команд).
- Для тестирования DLL ibaLogic можно запустить как исполняемую программу.

Важно

ibaLogic не может распознать и устранить ошибки программирования в DLL. Это означает, что такие ошибки могут привести к сбою в работе ibaLogic. Этот аспект необходимо учитывать при создании DLL.

5.4.4.3 Интеграция файлов DLL в ibaLogic

После создания DLL нужно переместить в папку с ibaLogic (как правило: "C:\Program Files\iba\ibaLogic v4\Server\Dll").

После перезапуска сервера ibaLogic этот DLL появляется как функциональный блок в папке "ПОЛЬЗОВАТЕЛЬСКИЕ". Теперь его можно перетащить, как любой другой блок, и интегрировать в программу.

Пример

Файл "Para_File_Read_Store_Dll" был создан и помещен в папку с программой. Он находится в группе "ПОЛЬЗОВАТЕЛЬСКИЕ".

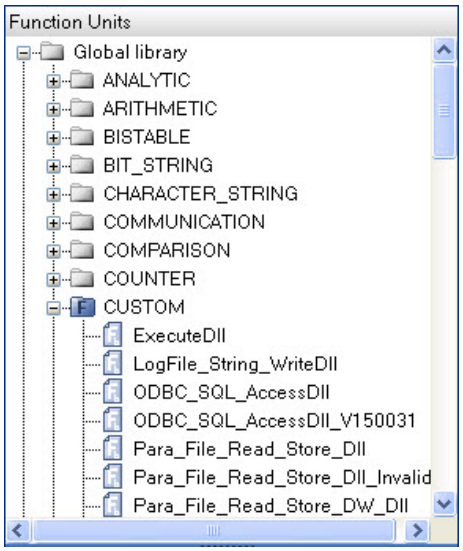


Рис. 74: Файл Para_File_Read_Store_Dll в дереве функциональных блоков

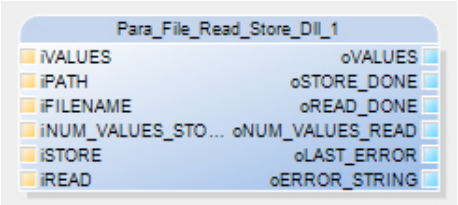


Рис. 75: Файл Para_File_Read_Store_Dll как функциональный блок в программе

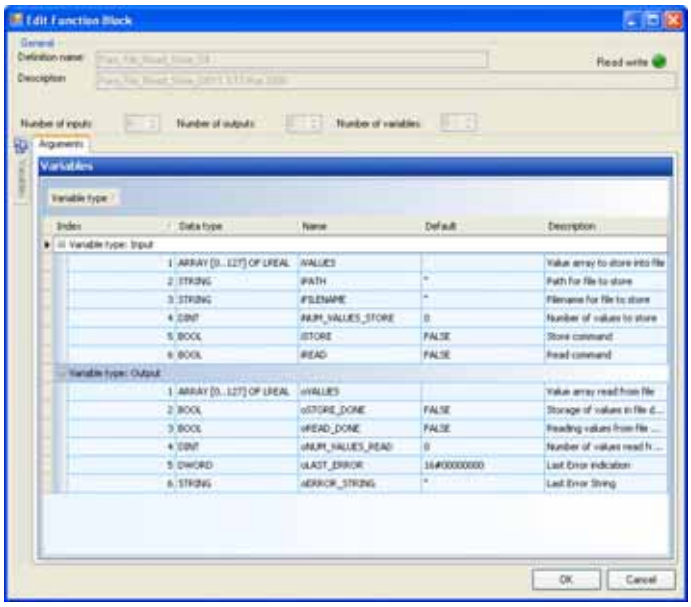


Рис. 76: Свойства файла Para_File_Read_Store_Dll

➲ При двойном щелчке по функциональному блоку вызывается окно с опциями для его редактирования ("Edit Function Block").
При этом код блока пользователь посмотреть не может. Отображаются только входы и выходы с их описаниями.

5.5 Типы данных

Каждой переменной присваивается тип данных.

В отличие от третьей версии (V3), ibaLogic V4 поддерживает не только элементарные типы данных и массивов, но и сложные, а также типы данных, определяемые пользователем.

Типы данных, которые могут использоваться в ibaLogic, можно разделить на следующие категории:

- Стандартные типы данных
- Сложные типы данных, такие как: ARRAY, STRUCT и ENUM
- Производные типы данных, которые формируются из типов данных двух вышеупомянутых групп.

Пользователь может определять собственные типы данных, подпадающие под категории "сложные" и "производные".

Примечание

Более подробная информация содержится в Приложении C, "Типы данных".

5.5.1 Определение типов данных

➲ Щелкните кнопку "Типы данных" ("Data Types").

В дереве элементов в области навигации отобразится соответствующая папка.

Для **неэлементарных** типов данных в глобальной библиотеке и под каждым проектом в рабочей группе создаются следующие папки:

- Непосредственно производные типы
  Стандартный тип данных с фиксированным значением по умолчанию
- Тип-диапазон
  Стандартный тип данных с фиксированным значением по умолчанию и ограниченным диапазоном значений
- Строковые производные типы
  Стандартный тип данных с фиксированной длиной и текстом по умолчанию
- Перечисляемые типы
  Перечисления: вместо целых значений определяются имена.
- Массивы
  Массив элементарных типов данных с фиксированной размерностью и глубиной.
- Структуры
  Структура, состоящая из элементарных типов данных

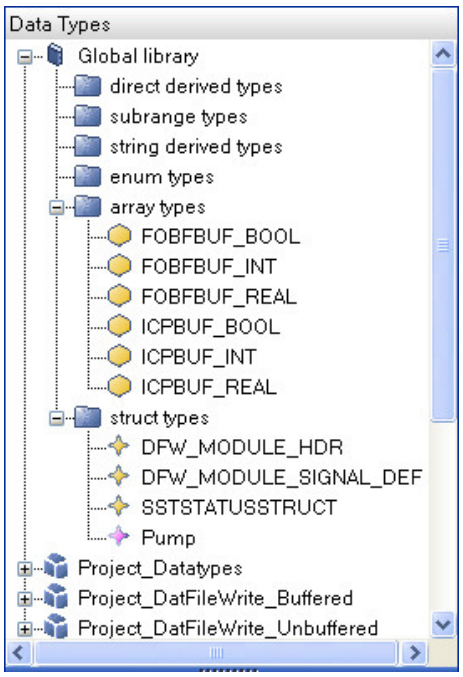


Рис. 77: Типы данных

В сложных типах "Массив" и "Структура" содержатся типы данных, которые были предварительно определены программой ibaLogic. К таким типам относятся:

- FOBFBUF_BOOL / _INT / _REAL
  Одноразмерные массивы с 256 элементами, используются в "Режиме буферизации". Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".
- ICPBUF_BOOL / _INT / _REAL
  Одноразмерные массивы с 1024 элементами, используются для соединения аналоговых вводов на платформе PADU-S-IT.
  Более подробная информация содержится в пункте 9.5 "Локальная периферия (PADU-S-IT)".
- DFW_MODULE_HDR / DFW_MODULE_SIGNAL_DEF
  Структура для передачи данных в блок DAT_FILE_WRITE.
- SSTSTATUSSTRUCT
  Структура для получения диагностической информации о ведущей карте Profibus SST.

Примечание

Пользователь может блокировать отображение предварительно заданных и автоматически генерируемых типов данных. Для этого нужно перейти в меню "Инструменты – Опции – Общее – Система" ("Extras – Options – General – System") и поставить галочку напротив "Блокировать отображение сгенерированных типов данных" ("Suppress Generated Data Types").

Эти данные могут использоваться в программе, даже если их отображение блокировано.

Последовательность действий

Пользователь может определить тип данных различными способами:

- ❑ В проекте
- ❑ В глобальной библиотеке
- ❑ При создании функционального блока

5.5.1.1 В проекте

1. В дереве функций правой кнопкой мыши щелкните по нужной категории в проекте.
2. В контекстном меню выберите "Новый".

5.5.1.2 В глобальной библиотеке

1. В дереве функций правой кнопкой мыши щелкните по нужной категории в глобальной библиотеке.
2. В контекстном меню выберите "Новый".

5.5.1.3 При создании функционального блока

Опция создания нового типа данных содержится в окне выбора типа данных переменной.

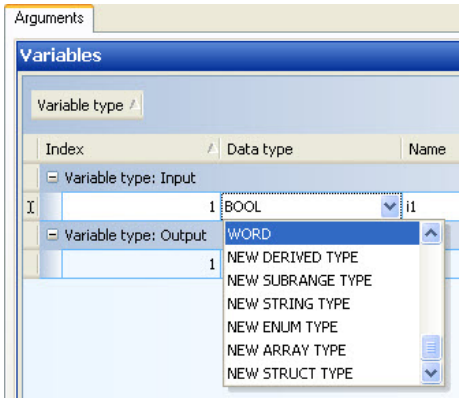


Рис. 78: Создание типа данных для блока

Последовательность действий

1. Создайте переменную с соответствующим типом данных.
2. Протестируйте созданный тип данных, чтобы убедиться в отсутствии ошибок. Если тест прошел успешно, щелкните <OK>, чтобы данные были приняты программой.

Результат

Если в синтаксисе нет ошибок, то под выбранной категорией создается тип данных.

5.5.2 Изменение типа данных

Примечание

Тип данных, который уже используется, изменить нельзя.

Когда пользователь закрывает диалоговое окно с помощью кнопки <OK> или <Применить>, появится сообщение о том, что можно создать копию типа данных под другим именем.

Последовательность действий

1. Щелкните правой кнопкой мыши по типу данных.
2. В контекстном меню выберите "Свойства".
3. Измените параметры.

5.5.3 Удаление типа данных

Необходимое условие

Тип данных, который нужно удалить, не используется программой.

Последовательность действий

1. Щелкните правой кнопкой мыши по типу данных.
2. В контекстном меню выберите "Удалить" ("Remove") или нажмите <Del>.

5.5.4 Управление типами данных

Копирование в глобальную библиотеку

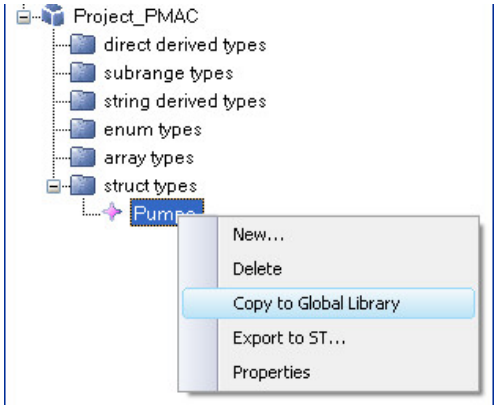
Если нужно использовать тип данных, который определен в одном проекте, также и в другой рабочей области, то его необходимо скопировать в глобальную библиотеку.

Примечание

Нельзя копировать или перемещать типы данных из глобальной библиотеки в каталог проекта. Аналогичным образом, невозможно копировать / перемещать данные между проектами. Если вы используете тип данных, который определен в одном проекте, то его нужно скопировать в глобальную библиотеку.

Последовательность действий

1. Щелкните правой кнопкой мыши по типу данных в проекте.
2. В контекстном меню выберите "Копировать в глобальную библиотеку" ("Copy To Global Library").



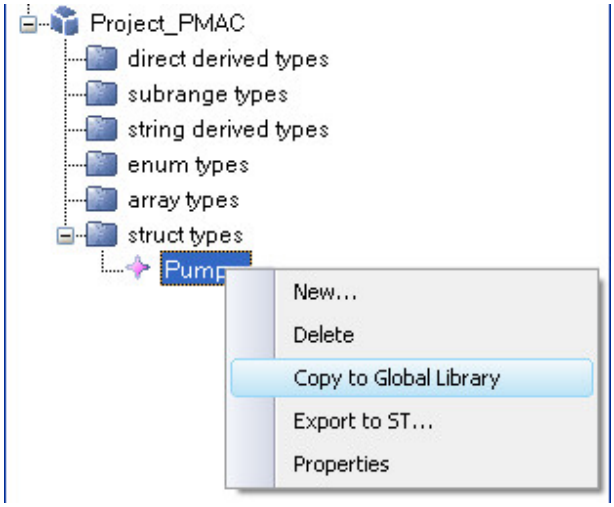
5.5.5 Экспорт типов данных

Если тип данных, который определен в базе данных под глобальной библиотекой или проектом, нужно также использовать в другой базе данных или другом программном инструменте, то этот тип данных экспортируется как текстовый файл.

Последовательность действий

1. Щелкните правой кнопкой мыши по типу данных.
2. В контекстном меню выберите "Экспортировать в ST" ("Export to ST").

   Появится диалоговое окно "Экспорт".



3. Определите целевой каталог и имя файла.

5.5.6 Импорт типа данных

Необходимое условие

ibaLogic не находится в режиме онлайн.

Последовательность действий

➲ Выберите меню "Файл – Импорт – Структурированный текст" ("File – Import – Structured Text").

5.5.7 Использование типа данных

После того как тип данных был определен, он может использоваться:

- при создании функционального блока.
- при создании структурного типа данных.
- при создании входов и выходов.

5.5.7.1 При создании функционального блока

➲ При создании переменной в редакторе блоков в столбце "Тип данных" выберите пользовательский тип данных

5.5.7.2 При создании структурного типа данных

➲ При создании элементов структуры в редакторе типов данных в окне выбора "Тип данных" выберите пользовательский тип данных.

Примечание

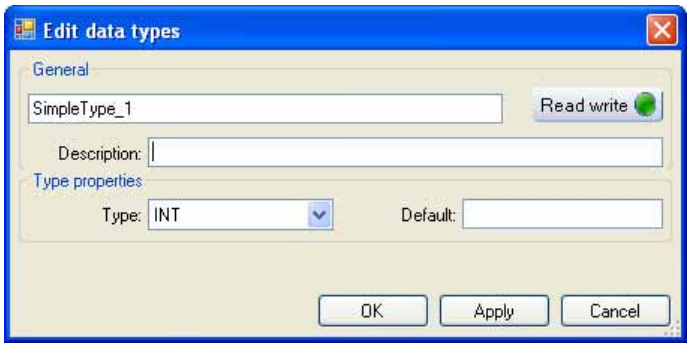
Доступ к типам данных, которые были определены в проекте в другой рабочей области, невозможен.

5.5.8 Пользовательские типы данных

Последовательность действий

1. Щелкните правой кнопкой мыши по группе типов данных.

Появится диалоговое окно "Редактировать типы данных" ("Edit data types").



Для всех типов данных это диалоговое окно состоит из:

- Раздела "Общее" ("General") (одинаковый для всех типов данных)
- Раздела "Свойства типа" ("Type Properties")
- Раздела "Элементы" ("Elements")

Общее

- Имя:
  Имя пользовательского типа данных. Под этим именем тип данных будет доступен в окнах выбора.
- Описание:
  Любой текст для описания типа данных. Описание будет отображаться только в определении.

Свойства типа данных

- Тип:
  Определяет тип данных.
- Значение по умолчанию:
  Исходное значение (заданное)

5.5.8.1 Группа DIRECT DERIVED TYPE (непосредственно производный тип)

Используется для определения элементарного типа данных, для которого можно указать новое имя и значение по умолчанию.

Например, таким образом такие константы, как число пи могут быть определены с помощью типа данных LREAL.

5.5.8.2 Группа SUBRANGE TYPE (тип-диапазон)

Это целочисленный (integer) тип данных с ограниченным диапазоном значений и значением по умолчанию.

Он может использоваться, например, для определения индексов массивов с определенной глубиной.

Важно

Этот тип данных **не** имеет ограничений. Просто в случае выхода за пределы диапазона в процессе исполнения программы появляется сообщение об ошибке.

5.5.8.3 Группа STRING DERIVED TYPE (строковые производные)

Это строковый тип данных, имеющий ограниченную длину.

Он может использоваться для определения строк текста, которые имеют фиксированное начальное значение, например, сообщений об ошибке.

Примечание

Максимальная длина строки: 250 символов.

5.5.8.4 Группа ENUM TYPE (перечисляемый тип)

Тип данных категории ENUM TYPE - это перечисление.

Этот тип данных используется для символьного обозначения значений переменной, например положения переключателя.

Пример: тип данных "переключатель" ("Switch")

Пользователь хочет создать тип данных "переключатель" ("Switch"), который содержит 3 положения: ВПЕРЕД, СТОП и НАЗАД.

Для этого нужно определить ENUM TYPE Switch, с количеством возможных значений - 3 и положениями переключателя в качестве перечисляемых значений.

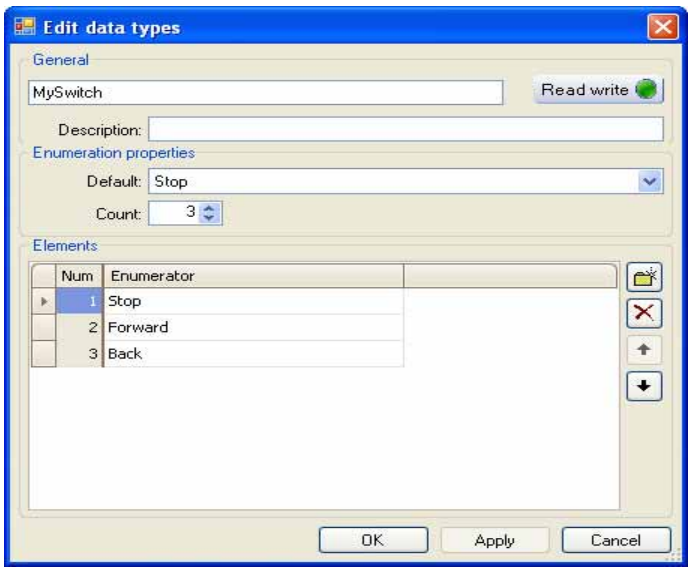


Рис. 79: Диалоговое окно "Редактировать типы данных"

Обратите внимание на то, что отдельные перечисляемые значения доступны через "Имя типа_перечисление" ("Type name_Enumerator") в "структурированном тексте" ("Structured Text").

Присвоение и запрос значений:

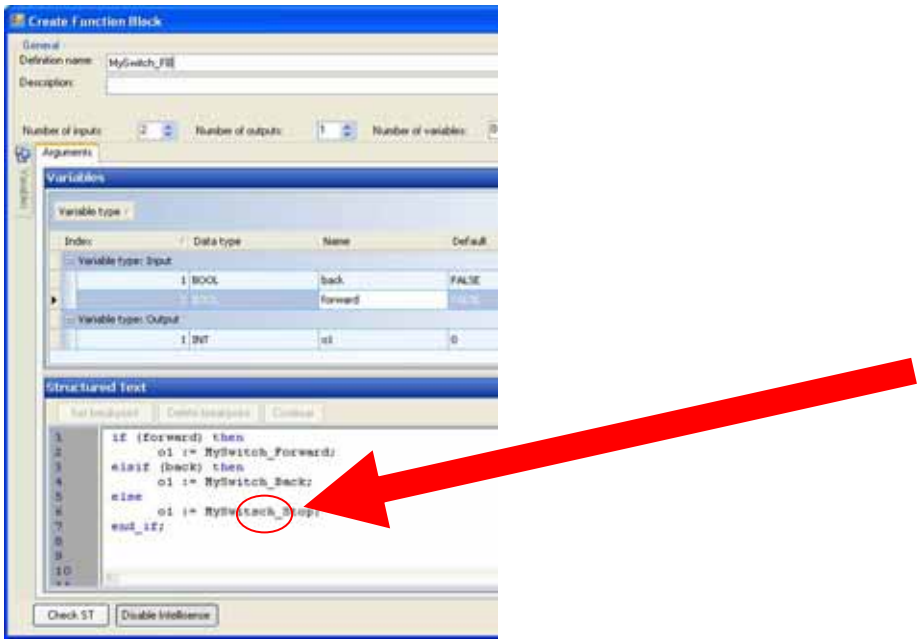


Рис. 80: Редактор переменных

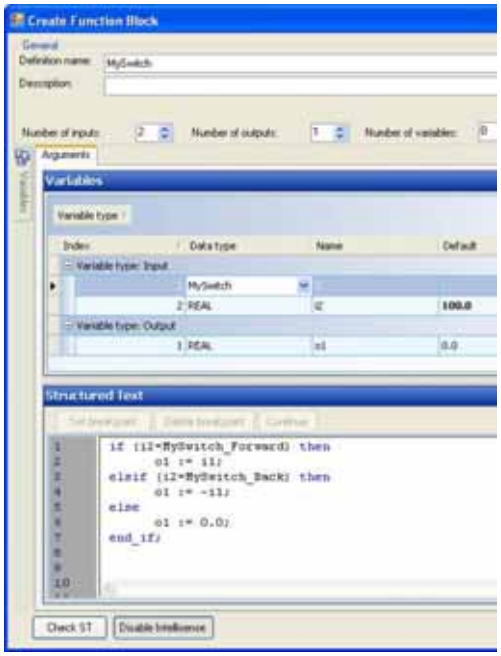


Рис. 81: Редактор переменных

Примечание

Напоминаем о том, что перечисляемыми значениями не могут быть целые числа (integer values).

Исключение:
OPC-коннектор указан как тип Enum, он считывается и записывается извне. В этом случае OPC-клиент записывает или считывает перечисляемое значение как целое.

5.5.8.5 Группа ARRAY TYPE (массив)

Массивы - это одномерные и многомерные поля. Все элементы массива принадлежат к одному и тому же типу данных. Однако, это не обязательно должен быть элементарный тип данных, пользователь может создавать массивы из пользовательских типов данных, структур, строк или массивов.

Пример: 2-х мерный массив целых чисел

Параметр	Объяснение
Тип	Базовый тип элементов массива
Количество	Количество измерений

Значение по умолчанию	Значения по умолчанию для массива
	Примеры
	`Одномерн. массив: [1.0,2.0,3.0]`
	`Двухмерн. массив: [[1.0,2.0,3.0],[4.0,5.0,6.0]]`

Нижний / верхний предел	Диапазон значений индекса элементов определяет глубину отдельных измерений. Максимальное значение: от 0 до 32 766

Доступ к элементам массива в структурированном тексте:

i1 - это переменная типа "массив". *o1, o2...* переменные элементарного типа;

- Одномерн. массив:
```
o1 := i1[0];
```
- Двухмерн. массив:
```
o1 := i1[0,0];    (* 1-й элемент 1-го измерения *)
o2 := i1[0,1];    (* 1-й элемент 2-го измерения *)
```
- Массив_массивов (Array_of_array):
```
o1 := i1[0][0];   (* 1-й элемент 1-го массива *)
o2 := i1[0][1];  (* 1-й элемент 2-го массива *)
o3 := i1[1][0];  (* 2-й элемент 1-го массива *)
```
- Массив_структур (Array_of_struct):
```
o1 := i1[0];
 (* 1-я структура массива *)
o2 := i1[0].elem.; (* элем. 1-й структуры *)
```

Примечание касательно структурированного текста

Индексами массивов могут быть только переменные с типом данных "Int". В отличие от ibaLogic V3, использование таких выражений, как [ix+4], **не** допускается.

5.5.8.6 Группа STRUCT TYPE (структуры)

В отличие от массивов, в структурах пользователь может объединять переменные, принадлежащие к разным типам данных. Определение элементов выполняется отдельно, при этом можно использовать все ранее заданные типы данных, включая пользовательские типы и массивы.

Пользователь может определить имя, описание и значение по умолчанию для каждого элемента структуры.

Пример: насос

Информация, относящаяся к насосу, собрана под типом данных "Насос".

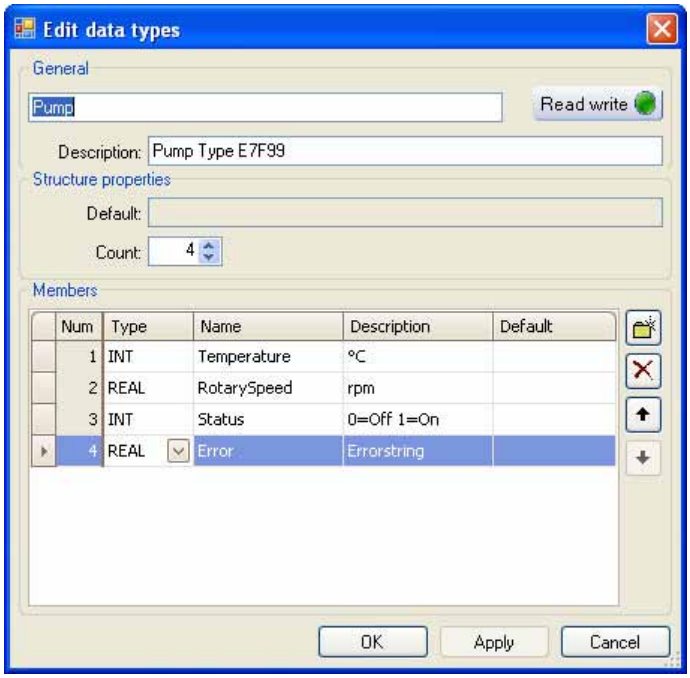


Рис. 82: Диалоговое окно "Редактировать типы данных"

Параметр	Объяснение
Тип	Базовый тип (допускается использовать пользовательские типы данных)
Количество	Количество элементов
Имя	Имя элемента
Описание	Пользовательское описание типа данных
Заданное значение	Инициализация по значению по умолчанию (заданному значению)

Доступ к элементам структуры в структурированном тексте:

```
1  (*o1 - переменная структурного типа "насос",          i1, i2... переменные
2  элементарного типа; *)
3
4  o1.●●●●●●●●●●● := i1;;  (* ●●● ●●●●●● INT *)
5  o1.Скорость := i2;       (* тип данных REAL *)
6
7  (*v1 - переменная структурного типа "насос";*)
8
9  if (v1.Temperature > 80 ) then
10      v1.Status := 99;
11      v1.Error := 'Temp. too high';
12 else
13      v1.Status := 0;
14      v1.Error := 'No error';
15 end_if;
```

Рис. 83: Структура в структурированном тексте

6 Элементы программы

Графическая программа ibaLogic содержит следующие элементы:

- Блоки
- Входы и выходы
- Соединительные элементы
- Преобразователи, элементы разделения и объединения, которые добавляются автоматически
- Комментарии

6.1 Создание элемента программы

Пользователь может создавать любые элементы, которые может содержать графическая программа ibaLogic.

Последовательность действий

1. Откройте контекстное меню, щелкнув правой кнопкой мыши по свободному пространству в области программы.
2. В контекстном меню выберите "Новый".
3. Выберите нужный элемент программы.

6.2 Выделение элементов программы

Один или несколько элементов программы можно выделить следующим образом:

Последовательность действий

1. Выберите нужный элемент, щелкнув по нему левой кнопкой мыши (выбор одного элемента).
2. Выберите нужные элементы, щелкая по ним левой кнопкой мыши и одновременно удерживая клавишу <Shift> или <Ctrl> (выбор нескольких элементов).
3. Нажав и удерживая левую кнопку мыши, очертите прямоугольник вокруг нескольких элементов, которые должны быть выбраны.
   При этом также выделяются существующие соединительные линии и преобразователи, если таковые имеются.
4. Чтобы отменить выбор одного или нескольких элементов, нажмите одновременно клавишу <Ctrl> и левую кнопку мыши.

Результат

Выбранные элементы отображаются в рамке из зеленых, голубых или серых прямоугольников.

Если выделено несколько элементов, то один из них будет выделен зеленым. Такой элемент является ключевым в группе.

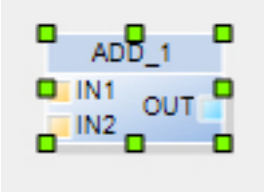
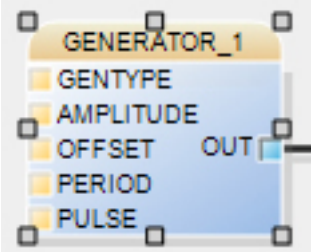
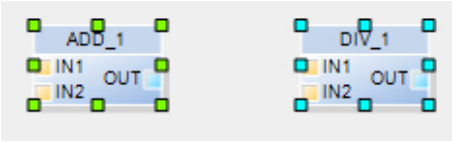
Маркировка	Объяснение
	Отмечен один элемент.
	Отмечен один элемент, но в данный момент выбран другой.
	Отмечено несколько элементов. Элемент, отмеченный зеленым, является ключевым элементом группы.

Рис. 84: Выбранный элемент

6.3 Перемещение элемента программы

Нажав и удерживая левую кнопку мыши, пользователь может перемещать предварительно выбранные блоки (и соединения).

Последовательность действий

➲ Нажав и удерживая левую кнопку мыши, вы можете переместить один или несколько выбранных элементов в нужное место.

Комментарий

Это касается линий с ограничениями.

6.4 Выравнивание элементов программы по краю

Все элементы программы можно выровнять по одному краю. Это может потребоваться для того, чтобы получить более четкую схему блоков.

Последовательность действий

1. Выделите элементы, которые нужно выровнять (блоки, внутристраничные коннекторы и коннекторы вне задачи).
2. Выберите требуемую функцию в меню "Диаграмма функциональных блоков - Выровнять".

Комментарий

Эта функция не применима к входам / выходам и линиям.

6.5 Копирование элемента программы

В программе можно копировать как один, таки несколько элементов одновременно.

Последовательность действий

1. Выделите элементы, которые нужно скопировать (блоки, внутристраничные коннекторы и коннекторы вне задачи).
2. Нажмите одновременно клавиши <Ctrl> + <c>, чтобы поместить выбранные элементы в буфер обмена.
3. Нажмите одновременно клавиши <Ctrl> + <v>, чтобы перенести выбранные элементы из буфера обмена в область программы.

Совет

Вместо сочетания клавиш можно также выбрать "Копировать" и "Вставить" в контекстном меню.

При выборе нескольких блоков соединительные линии между ними также будут скопированы.

Эта функция не применима к входам / выходам и линиям, которые выделяются отдельно.

6.6 Удаление элемента программы

Пользователь может также удалять элементы программы (один или несколько).

Последовательность действий

1. Выделите элементы, которые нужно удалить (блоки, внутристраничные коннекторы и коннекторы вне задачи).
2. Нажмите клавишу <Del>.

Комментарий

Можно также удалить элемент программы через опцию "Удалить" в контекстном меню.

6.7 Создание входных / выходных переменных

Необходимое условие

Выбрана кнопка "Входы - Выходы".

Последовательность действий

➲ Перетащите входную или выходную переменную в любое место левой или правой границы входов или выходов.

6.8 Графические соединения

В графическом программировании графическое соединение используется для передачи результатов одной функции в другую.

Существуют 3 различных типа соединений:

- ❑ Прямые коннекторы
- ❑ Внутристраничные коннекторы
- ❑ Коннекторы вне задачи

6.8.1 Прямые коннекторы

6.8.1.1 Типы коннекторов

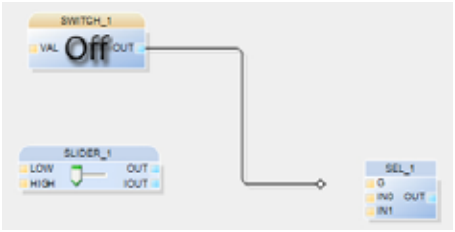
Для обозначения различных групп типов данных в ibaLogic используются соединительные линии разных цветов.

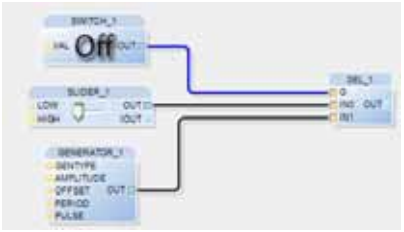
Тип линии	Объяснение
	Бинарные коннекторы отображаются в соответствии со своим состоянием: красный (ИСТИНА) или синий (ЛОЖЬ).
	Массивы отображаются зелеными линиями. Только массивы с одинаковой длиной и типом данных могут быть соединены между собой.
	Структуры отображаются оранжевым цветом.
	Перечисляемые типы отображаются желтым цветом.
	Все элементарные типы данных (например, INT, REAL...) обозначаются черными коннекторами.

6.8.1.2 Создание прямых коннекторов

Последовательность действий

1. Щелкните правой кнопкой мыши по выходному коннектору блока.
2. Нажав и удерживая правую кнопку мыши, перетащите коннектор в область входного коннектора блока.





Комментарии

Если результат одного блока должен использоваться в нескольких других, создайте ветвь, соединив линией один входной коннектор с другим существующим коннектором.

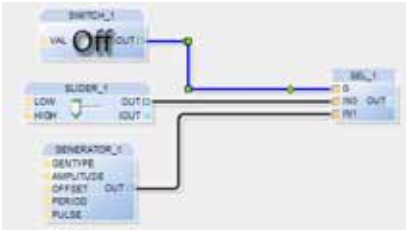
В непосредственной близости от подключаемого коннектора или линии курсор мыши автоматически перескакивает на коннектор или линию (эффект магнита).

6.8.1.3 Изменение прямых коннекторов

Последовательность действий

1. Выделите коннектор, который вы хотите изменить.

   Этот коннектор будет выделен маленькими зелеными квадратами и ромбами.



2. Измените эту линию нужным вам образом, перемещая зеленые четырехугольники с помощью курсора мыши.
3. Для того чтобы подключить соединение к блоку, щелкните по зеленому ромбу на соединительной линии.
4. Перетащите конец соединительной линии в пустую область, чтобы удалить соединительную линию.
5. Прикрепите конец соединительной линии к другому коннектору.

Примечание

При перемещении блока соединительные линии, распределенные пользователем вручную, будут автоматически перераспределены. В результате этого изменения, сделанные пользователем, будут отменены.

6.8.2 Внутристраничные коннекторы

Внутристраничные коннекторы (IPC - intra-page connector) упрощают графическое отображение программы. В этом случае вместо соединительной линии используется IPC.

Использование этой опции оптимизирует процесс программирования, особенно в том случае, если требуется соединить несколько объектов на одной странице или необходимы "длинные" соединения на несколько страниц. IPC - это не программный объект, а просто замена линии.

IPC могут использоваться как прямые коннекторы только в программе или на уровне макроса. Невозможно создать соединение от макроса к уровню вызова. Для этой цели в макросе должны быть определены входы и выходы.

6.8.2.1 Создание внутристраничных коннекторов

Внутристраничные коннекторы создаются в качестве замены линий.

Необходимое условие

Вы можете создать IPC на входном коннекторе, только если ранее был определен "Источник IPC" ("IPC Source").

Последовательность действий

➲ Нажав и удерживая клавишу <Ctrl> перетащите соединительную линию от одного выходного коннектора в свободное пространство области программы.

Комментарий

При выполнении вышеуказанных действий, появится диалоговое окно, в котором нужно выбрать соответствующий "IPC Source".

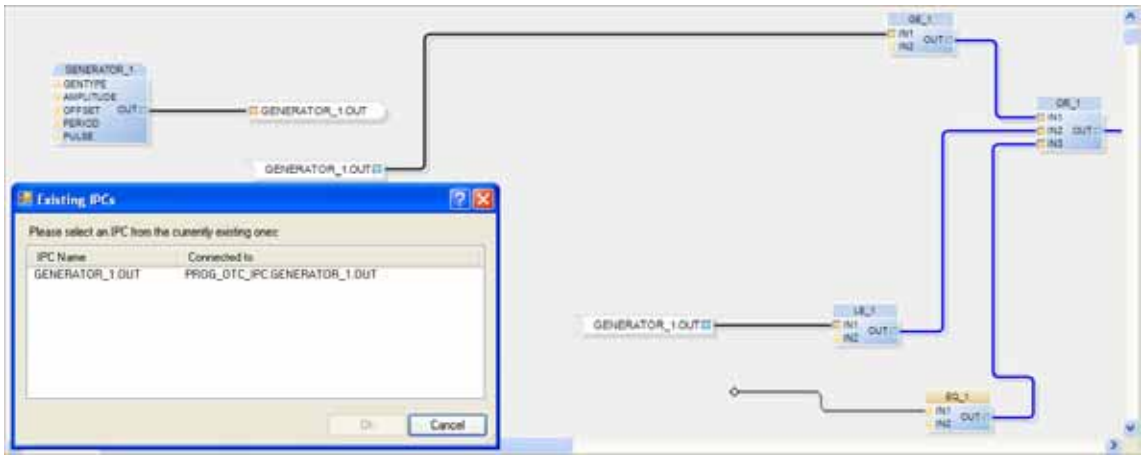


Рис. 85: Окно свойств

6.8.2.2 Изменение имен IPC

ibaLogic автоматически создает имя, которое имеет структуру: "Имя экземпляра блока.имя коннектора". Пользователь может изменить имя IPC по собственному усмотрению.

Последовательность действий

1. Щелкните двойным щелчком по источнику IPC.
   Появится диалоговое окно "Редактировать IPC" ("Edit IPC").
2. Присвойте источнику IPC имя и создайте комментарий. Присваиваемые имена должны соответствовать требованиям стандарта IEC.

Результат

Изменения автоматически принимаются для всех подключенных "IPC Targets", которые не могут быть изменены вручную.

6.8.2.3 Отслеживание внутристраничных коннекторов

Загрузите соответствующую страницу программы для выбранного IPC.

Последовательность действий

1. Щелкните правой кнопкой мыши по IPC.
2. В контекстном меню выберите "Перейти к ->" ("Go To ->").

   Будет отображен выход и все соединенные с ним входы.
3. Щелкните по любому из соединений.

Результат

Будет загружена соответствующая страница программы и выделен IPC.

Контекстное меню	Объяснение
01. -> Generator_1.OUT	Генератор
02.°Generator_1.OUT ->	Выделенный IPC
03. Generator_1.OUT ->	Потребитель

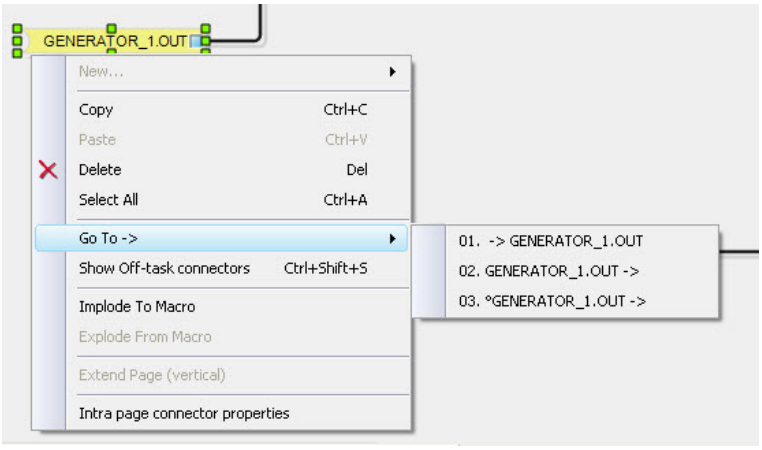


Рис. 86: Отслеживание IPC

6.8.3 Коннекторы вне задачи

Коннекторы вне задачи (Off-task connectors - OTC) используются как независимые соединительные элементы, которые необходимы, если требуется реализовать обмен данными между программами.

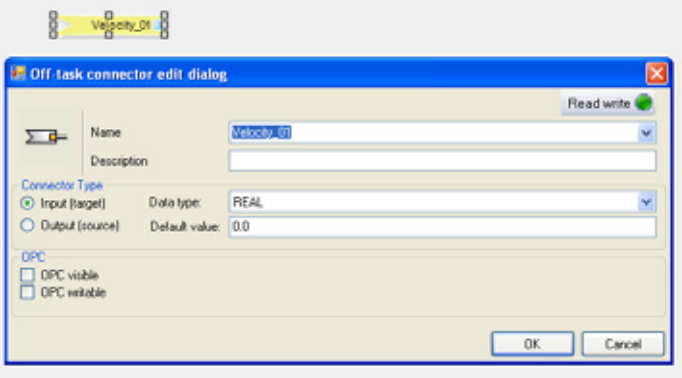
Можно настроить возможность записи и считывания OTC клиентами OTC.

6.8.3.1 Создание коннекторов вне задачи

Последовательность действий

1. Поместите курсор мыши в любое свободное место в области программы.
2. Щелчком правой кнопкой мыши откройте контекстное меню.
3. Выберите "Новый… - Новый коннектор вне задачи" ("New… - New Off-Task Connector").

   Появится диалоговое окно "Редактировать коннектор вне задачи" ("Off-task connector edit").



4. Присвойте необходимые параметры.

Создание соединения, не зависящего от программы

Последовательность действий

Метод 1:

1. Сначала создайте выход OTC (источник), введя информацию в диалоговом окне.
2. Скопируйте выход OTC.
3. Вставьте выход OTC в целевую программу.
   При этом параметры будут приняты, но направление меняется на противоположное.

Метод 2:

1. Создайте OTC в целевой программе.
2. В окне выбора выберите имя соответствующего выхода OTC.
   При этом другие параметры также будут приняты.
3. Нужно настроить направление на "Вход".

Свойства OPC

Пользователь может определить следующие свойства для OPC.

Окна выбора Свойства OPC	Объяснение
Видимый для OPC (OPC visible)	Определяет, будет ли этот коннектор отображаться в пространстве имен OPC.
Запись OPC активирована (OPC write-enabled)	Определяет, должен ли OPC-клиент записывать данные в этот коннектор.

Более подробная информация содержится в пункте 9.7.2 "Настройка параметров переменных OPC".

Правила создания OTC

- ❑ Выходной OTC должен быть уникальным в проекте.
- ❑ Для выходного OTC можно создать несколько входных OTC. Это можно сделать даже в программе, однако не в той, в которой расположен выходной OTC.
- ❑ Входной OTC может иметь только один источник данных: либо OPC-совместимый, либо связанный с ним выходной OTC.
- ❑ При создании выходного OTC, имя которого совпадает с входным OTC, этот входной OTC больше не доступен для OPC-записи.
- ❑ Входной OTC, который не имеет источника данных, может использоваться как константа / параметр.

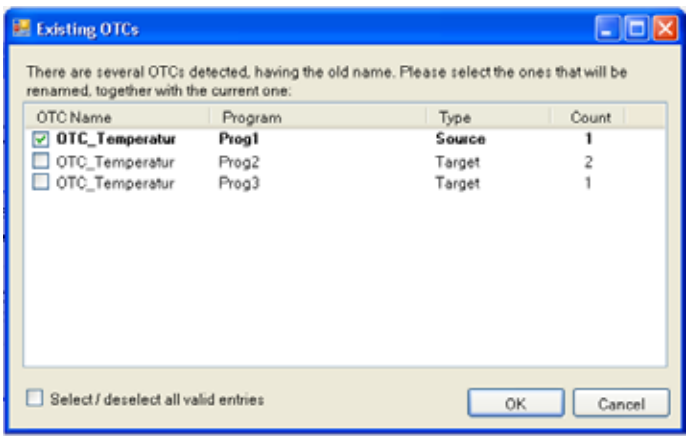
6.8.3.2 Изменение имени OTC

Последовательность действий

1. Выберите OTC, который нужно переименовать.

   Появится диалоговое окно "Существующие OTC" ("Existing OTCs").

   При переименовании соединенных OTC появляется окно, в котором можно выбрать, должны ли все соединенные OTC быть переименованы или только часть. Таким образом, например, можно присвоить правильное имя OTC, которому ранее было присвоено некорректное имя.



2. Подтвердите изменения, щелкнув по кнопке <OK>.

Результат

Все выбранные OTC переименованы.

Комментарий

Если один OTC используется в программе несколько раз, то это количество можно увидеть в столбце СЧЕТ (COUNT).

6.8.3.3 Отслеживание OTC

Последовательность действий

1. Щелкните правой кнопкой мыши по OTC.
2. В появившемся контекстном меню выберите "Перейти к ->" ("Go To ->"). Будут отображены программы, в которых созданы и используются OTC.
3. Щелкните по любому из соединений.

Результат

Будет загружена соответствующая страница программы и выделен OTC.

Контекстное меню	Объяснение
01. -> OTC_Температура: Прог1	Генератор
02. OTC_ Температура -> : Прог3	Потребитель

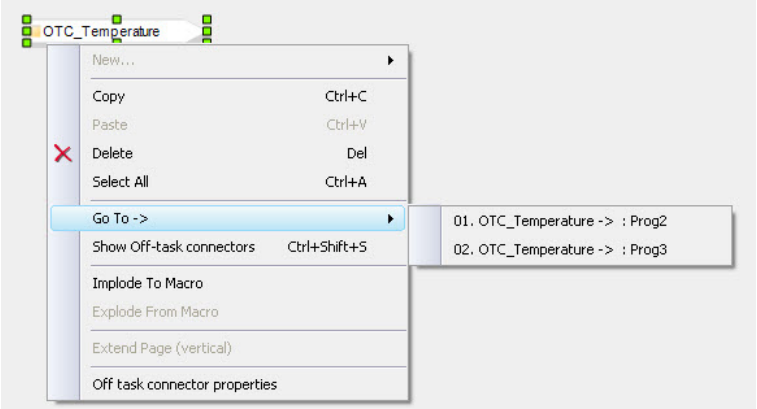


Рис. 87: Отслеживание OTC

6.8.3.4 Список всех OTC

Отображается список всех OTC, созданных в программе.

Последовательность действий

1. Поместите курсор мыши в любое свободное место в области программы.
2. Откройте контекстное меню.
3. Выберите "Отобразить коннекторы вне задачи" ("Display Off-task Connectors").

   Появится диалоговое окно, содержащее список всех существующих OTC, отсортированных в алфавитном порядке.
4. Поиск по списку осуществляется вводом начальной буквы или двойным щелчком по OTC.

Результат

Отображается список всех OTC, созданных в программе.

Комментарий

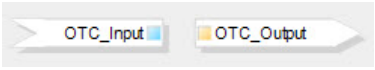
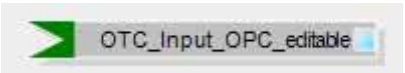
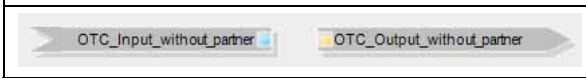
Эту функцию также можно вызвать через меню "Диаграмма функциональных блоков - Отобразить коннекторы вне задач" ("Function diagram -- Display Off-task Connectors").

Примечание

Вы можете продолжить работу в программе, не закрывая это диалоговое окно. При добавлении или удалении OTC список обновляется автоматически. Диалоговое окно может располагаться в любом месте, а также пристыковываться к границе окна программирования.

6.8.3.5 Отображение

Для удобства пользователя различные свойства обозначаются разными цветами:

Отображение OPC	Описание
	Обозначение входов и выходов, которые соединены
	Обозначение входов и выходов, видимых для OPC и доступных для чтения и записи
	Обозначение входа, который доступен OPC для чтения и записи
	Обозначение входов и выходов, которые не соединены

6.9 Преобразователи, элементы разделения и объединения

6.9.1 Преобразователи

Автоматическое добавление преобразования типа данных

В стандартных CFC-редакторах можно соединять интерфейсы с разными типами данных. ibaLogic автоматически добавляет преобразователь, если возможно корректное преобразование. При отображении этот автоматический преобразователь имеет очень маленький размер, что экономит место в области программы. При наведении курсора на преобразователь появляется подсказка, которая содержит информацию о преобразовании.

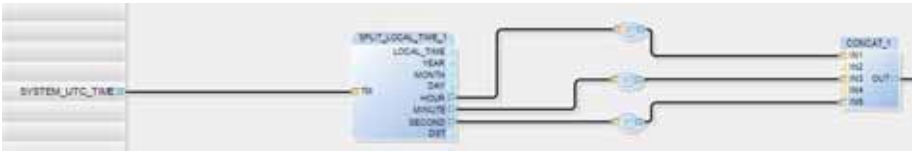


Рис. 88: Преобразование типов данных

Примечание

Блокам с соединениями, которые не принадлежат к какому-либо типу, присваивается тип данных только при размещении коннектора. Этот тип данных принимается для всех соединений и сохраняется до тех пор, пока последнее соединение не будет удалено.

Если вы хотите соединить два соединения, которым не присвоен тип данных, появится диалоговое окно, в котором можно выбрать допустимый тип данных.

6.9.2 Элементы разделения

Вы хотите удалить элементы из структуры данных. При использовании стандартных методов придется создать пользовательский блок, в котором для присвоения структуры выходных коннекторов отдельным элементам используется структурированный текст. IbaLogic создает этот блок, который называется разделителем, автоматически.

Последовательность действий

1. Расположите соединительную линию между входным коннектором одного блока и структурным выходным коннектором другого блока или входа OTC. Появится окно выбора.
2. Выберите элемент структуры.

Комментарий

Таким образом обеспечивается доступ к остальным элементам структуры.

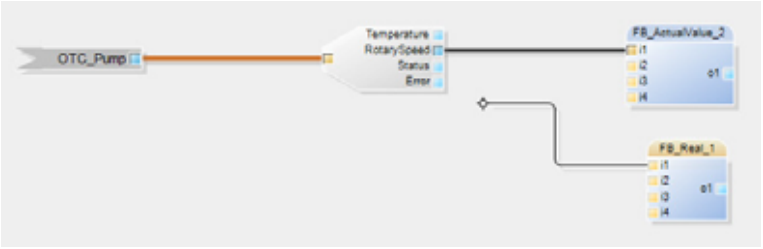


Рис. 89: Элемент разделения

6.9.3 Элементы объединения

Если вы хотите соединить выходной коннектор со входным коннектором структуры, то для этого в ibaLogic доступно окно выбора, в котором можно выбрать один из элементов структуры. ibaLogic добавит блок объединения, со входами которого вы сможете соединить другие сигналы.

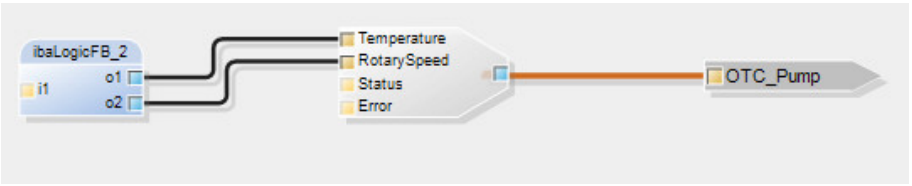


Рис. 90: Элемент объединения

6.10 Комментарии в программе

Комментарии - это графические элементы, которые можно разместить в любом свободном пространстве области программы. Комментарии не должны накладываться на функциональные блоки, OTC и IPC. Однако размещение комментариев на коннекторах возможно. Коннекторы будут видны, поскольку поля комментариев прозрачны.

Поле комментария имеет указатель, который можно пристыковать к функции, которую он описывает.

Последовательность действий

1. Поместите курсор мыши в любое свободное место в области программы.
2. Откройте контекстное меню.
3. Выберите "Новый… - Новый комментарий" ("New… - New Comment").

7 Исполнительная система PMAC

7.1 Обзор режимов онлайн и офлайн

В основном меню управления исполнительной системой доступны следующие меню и значки:

- ❑ Пуск (Start) (меню "Вычисление" ("Evaluation") и кнопка в панели инструментов)
- ❑ Стоп (Stop) (меню "Вычисление" ("Evaluation") и кнопка в панели инструментов)
- ❑ Сохранить программу в PMAC (Save program on the PMAC) (меню "Вычисление")
- ❑ Очистить память PMAC (Delete PMAC memory) (меню "Вычисление")
- ❑ Разъединить (Disconnect) (кнопка в панели инструментов)
- ❑ Обновление (Updating) (кнопка в панели инструментов)

7.2 Запуск исполнительной системы

В отличие от стандартных автоматизированных систем, этапы "Компиляция" и "Загрузка" выполняются автоматически в фоновом режиме.

Последовательность действий

➲ Щелкните кнопку <Пуск> в панели инструментов.

Результат

Выполнение следующих действий:

- ❑ Проект скомпилирован.
- ❑ Проект передан в PMAC.
- ❑ Начинается исполнение программы.
- ❑ Время вычисления отображается в панели инструментов окна программы.
- ❑ Отображены все области со значениями.
- ❑ В видимой части программы все области со значениями содержат текущие значения.
- ❑ Цвет фона области программы меняется на розовый.
- ❑ Программа работает в режиме онлайн.

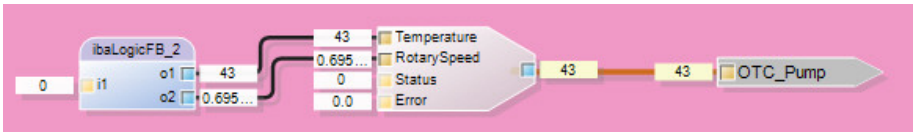


Рис. 91: Режим онлайн

Ошибки, произошедшие во время компиляции, загрузки и т.д., отображаются в окне событий. По умолчанию это окно располагается под областью программы. Пользователь может скрыть окно событий или расположить его по собственному усмотрению.

Совет

Важной особенностью ibaLogic является то, что пользователь может осуществлять программирование (почти полностью) в режиме онлайн.

Исключения:

Конфигурирование платформы

Конфигурирование входов/выходов

Импорт программ / блоков / типов данных

Конфигурирование блока DAT_FILE_WRITE

Еще одна особенность:

Пользователь может закрыть клиент в режиме онлайн, не останавливая PMAC. При перезапуске клиента он незамедлительно соединяется с PMAC в режиме онлайн.

Это особенно полезно, если PMAC запущена на другой платформе (другой ПК или PADU-S-IT). После этого пользователь может выключить компьютер с ibaLogic, а PMAC продолжит работать. После перезагрузки, запуска сервера и клиента последний автоматически соединяется с PMAC, работающей в режиме онлайн.

7.3 Останов исполнительной системы

После создания программы вы можете сразу же запустить ее щелчком по кнопке <Пуск>.

Последовательность действий

➲ Щелкните кнопку <Стоп> в панели инструментов.

Результат

После нажатия кнопки <Стоп> будут выполнены следующие действия:

- ❑ Исполнение программы (PMAC) будет остановлено.
- ❑ Цвет фона области программы меняется на серый.
- ❑ Области со значениями скрыты.
- ❑ Программа находится в режиме офлайн.

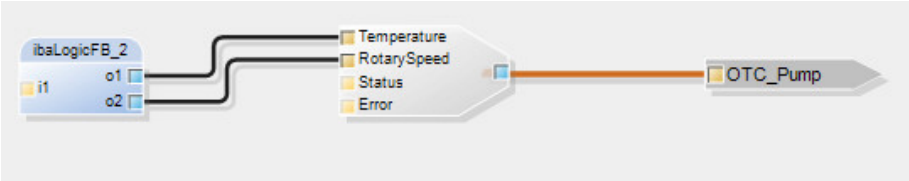


Рис. 92: Режим офлайн

7.4 Исполнительная система – автозапуск

7.4.1 Сохранение программы в PMAC

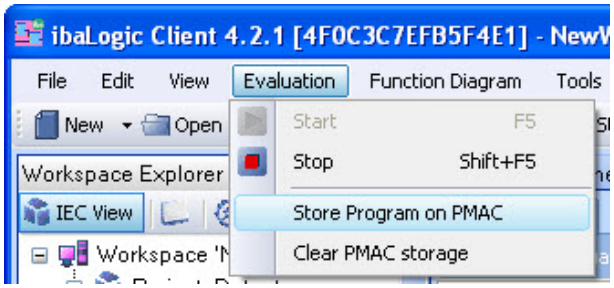
Если вы хотите запустить платформу с исполняемой программой, то эту программу необходимо предварительно сохранить в PMAC.

Необходимые условия

- ibaLogic не находится в режиме онлайн.
- Автозапуск активирован.

Последовательность действий

➲ В главном меню выберите "Вычисление – Сохранить программу в PMAC" ("Evaluation – Store Program on PMAC").



Результат

Программа сохранена в PMAC. Файл создан физически. При запуске PMAC обнаружит эту программу и начнет ее выполнение.

Более подробная информация содержится в пункте 3.4.2.4, "Активация автозапуска сервера ibaLogic ".

Примечание

Обратите внимание на то, что все последующие изменения, вносимые в программу, нужно обязательно сохранять в PMAC.

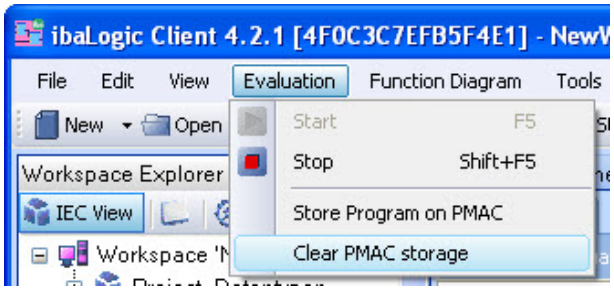
Чтобы избежать автоматического запуска PMAC, можно очистить память системы или изменить параметры автозапуска.

7.4.2 Удаление программы в PMAC

Будет удален файл-образ, созданный ранее с помощью команды "Сохранить программу в PMAC".

Последовательность действий

➲ В главном меню выберите "Вычисление – Очистить хранилище данных PMAC" ("Evaluation – Clear PMAC storage").



Результат

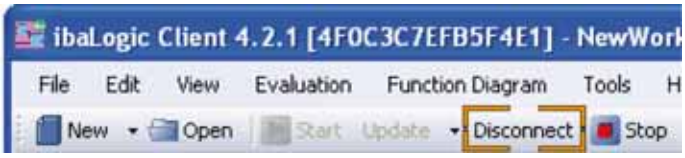
Созданный ранее файл-образ будет удален из памяти PMAC.

7.5 Соединить / разъединить

Опция разъединения используется, если требуется последовательно внести несколько изменений в программу, не останавливая при этом саму программу и не выполняя поэтапную компиляцию.

Последовательность действий

➲ В панели инструментов выберите <Разъединить> (<Disconnect>).



Пример

Вы хотите расширить структурный тип данных, который много раз используется в программе.

⚠ WARNING

Опасность при использовании активных блоков SWITCH / SLIDER в состоянии отсутствия соединения!

Любой блок SWITCH / SLIDER, существующий в момент разъединения, продолжает оставаться активным для запущенного PMAC и продолжает влиять на систему. Это поведение делает возможным выполнение операций с запущенным PMAC, несмотря на отсутствие соединения.

Если какой-либо из таких блоков (SWITCHES или SLIDER) будет удален, а затем добавлен снова под тем же именем, он незамедлительно оказывает влияние на запущенный PMAC.

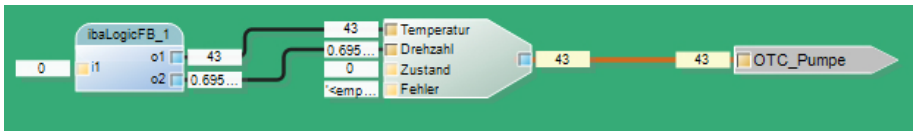
Если какой-либо функциональный блок или макрос будет удален, а затем добавлен снова под тем же именем, идентичные входы и выходы будут незамедлительно отображены. Значения относятся к блоку, запущенному в настоящий момент в PMAC, а не к только что созданному блоку, который может иметь совсем другое содержимое. Новая конфигурация будет принята после компиляции и загрузки программы.

Последовательность действий

1. Щелкните <Разъединить>.
2. Вы изменяете структурный тип данных и сохраняете его под новым именем (поскольку исходный тип данных используется, его нельзя изменить и сохранить под таким же именем).

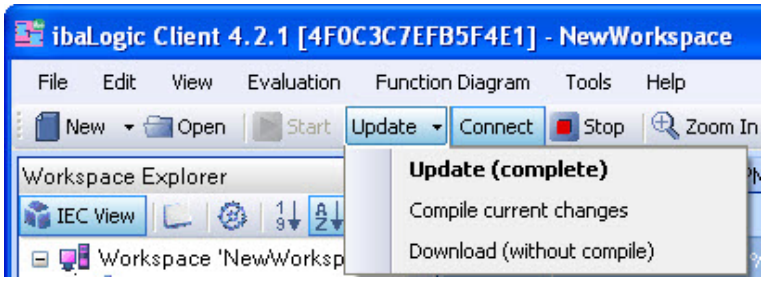
Будут выполнены следующие действия:

- ❑ Исполнение программы (PMAC) не завершается, а продолжается без помех.
- ❑ Цвет фона окна программы в клиенте меняется на зеленый.



- ❑ Области со значениями по-прежнему отображаются, данные в них обновляются.
- ❑ Вместо кнопки <Разъединить> появляется кнопка <Соединить>.
- ❑ Кнопка <Обновить> (<Update>) становится активной.

3. Теперь вы можете заменить предыдущий тип данных новым во всех блоках и OTC, в которых он используется. При этом изменения не компилируются и не загружаются в PMAC.
4. После внесения всех необходимых изменений вы можете целенаправленно выполнить компилирование и загрузить обновленные данные в PMAC. Это можно сделать за один прием (полное обновление) или в несколько этапов. В панели инструментов выберите <Обновить>.



При этом вы не выходите из режима "Разъединить".

5. Чтобы выйти из этого режима, щелкните <Соединить>.

Результат

Все изменения скомпилированы и загружены в PMAC.

8 Платформы

Прежде чем приступить к конфигурации интерфейсов с периферийными устройствами или другими системами, необходимо настроить базовое аппаратное обеспечение, на котором будет работать исполнительная система ibaLogic (PMAC).

В настоящее время для использования в качестве платформы доступны 2 класса устройств:

WinXP

PMAC работает на компьютере с ОС Windows, на котором также установлены другие компоненты ibaLogic.
Более подробная информация содержится в пункте 2.4, "Режимы работы и обработки данных ".

В данном случае соединение с распределенной периферией устанавливается с помощью PCI-карт.

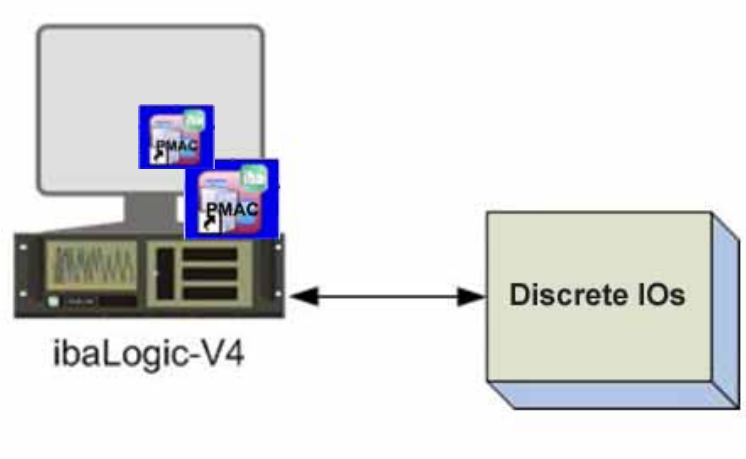


Рис. 93: Периферийный интерфейс, ПК с Windows

PADU-S-IT

PMAC установлен на станции ibaPADU-S-IT. Все остальные компоненты ibaLogic работают на одном или нескольких компьютерах с Windows.

В ibaPADU-S-IT для PMAC доступны только локальные компоненты ввода-вывода (периферийные модули). Для распределенных устройств и внешних систем в устройстве предусмотрен двунаправленный OFC-порт, а также Ethernet-порт для соединения по TCP/IP.

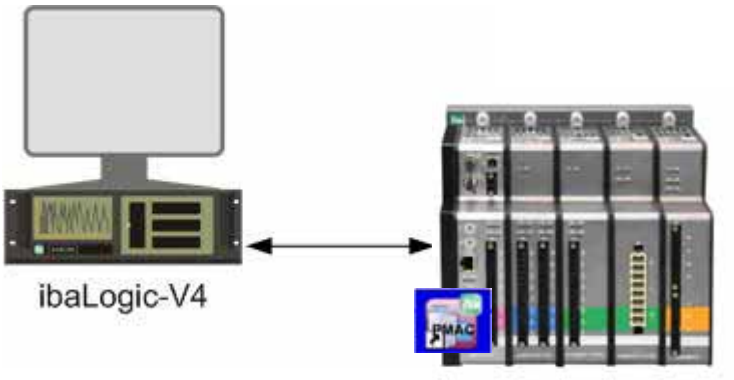


Рис. 94: Периферийный интерфейс PADU-S-IT

После первичного вызова клиента ibaLogic-V4, в качестве платформы устанавливается локальный ПК с Windows, то есть "WinXP".

8.1 Конфигурирование платформы

Диалоговое окно для конфигурирования платформы вызывается через меню "Инструменты" ("Extras"), которое находится в панели меню клиента ibaLogic.

Примечание

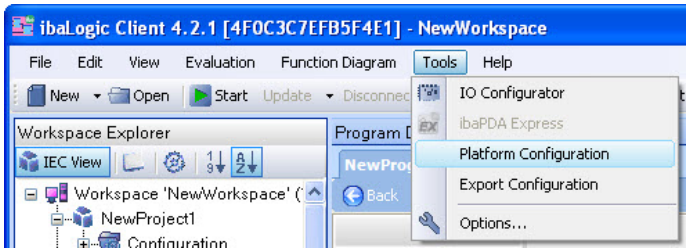
Платформы создаются отдельно под каждый проект. Все проекты рабочей области и сконфигурированные в них платформы содержатся в диалоговом окне "Конфигурация платформы" ("Platform Configuration").

Необходимое условие

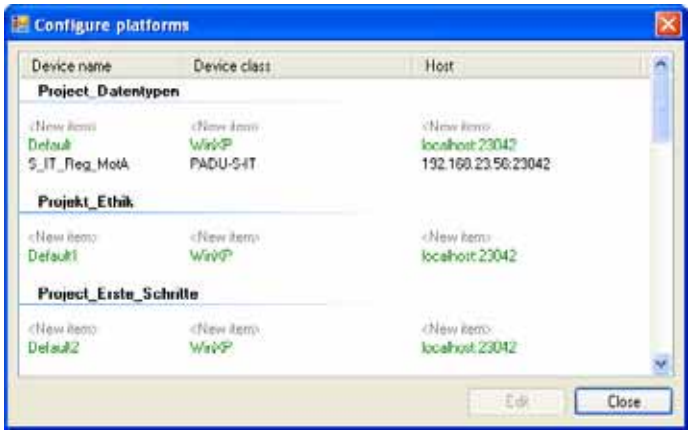
Открыт нужный проект.

Последовательность действий

1. Выберите пункт меню "Инструменты – Конфигурация платформы" ("Extras – Platform Configuration").

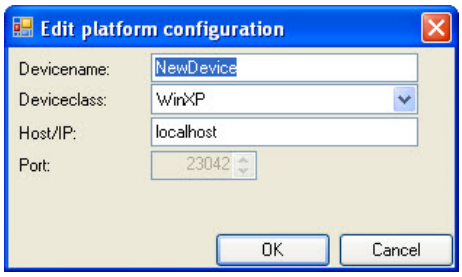


2. Для того чтобы создать или сконфигурировать платформу, в диалоговом окне под именем проекта щелкните <НОВЫЙ> (<NEW>) или щелкните соответствующую платформу.



Цвет	Объяснение
Зеленый	Активная система.
Светло-серый	Создание новой платформы.
Черный	Другие доступные платформы.

3. Щелкните кнопку <Редактировать> (<Edit>). Появится диалоговое окно "Редактировать конфигурацию платформы" ("Edit Platform Configuration").



4. Введите имя устройства или щелкните <OK>, чтобы принять уже имеющиеся настройки.
В рамках одной рабочей области это имя должно быть уникальным.

5. Выберите класс устройства: WinXP или PADU-S-IT.

6. Введите IP-адрес хоста.

- ❑ Если вы выбрали класс устройства WinXP, то, если PMAC работает на этом ПК или сервере, введите "Local host" или "127.0.0.1".
- ❑ Если PMAC находится на другом компьютере в сети, то укажите имя хоста или IP-адрес соответствующего компьютера.
- ❑ Если вы выбрали класс устройства PADU-S-IT, то введите имя хоста или IP-адрес устройства ibaPADU-S-IT, в котором будет использоваться PMAC.

7. Подтвердите указанную вами информацию, щелкнув <OK>.

8. Закройте диалоговое окно щелчком по кнопке <Закрыть> (<Close>).

8.2 Выбор платформы

Платформа, которая выбрана в настоящий момент, отображается в панели инструментов клиента ibaLogic-V4. С помощью окна выбора вы можете переключиться на другую платформу.

Последовательность действий

1. Щелкните окно выбора, чтобы переключить платформу.
2. Выберите одну из предварительно сконфигурированных платформ.

Важно

Учитывайте, что конфигурация ввода-вывода зависит от платформы.

После настройки платформы необходимо обновить конфигурацию ввода-вывода.

В меню "Инструменты – Конфигуратор ввода-вывода" ("Extras – I/O Configurator") выберите кнопку <Обновить апп. обесп.> (<Update Hardware>).

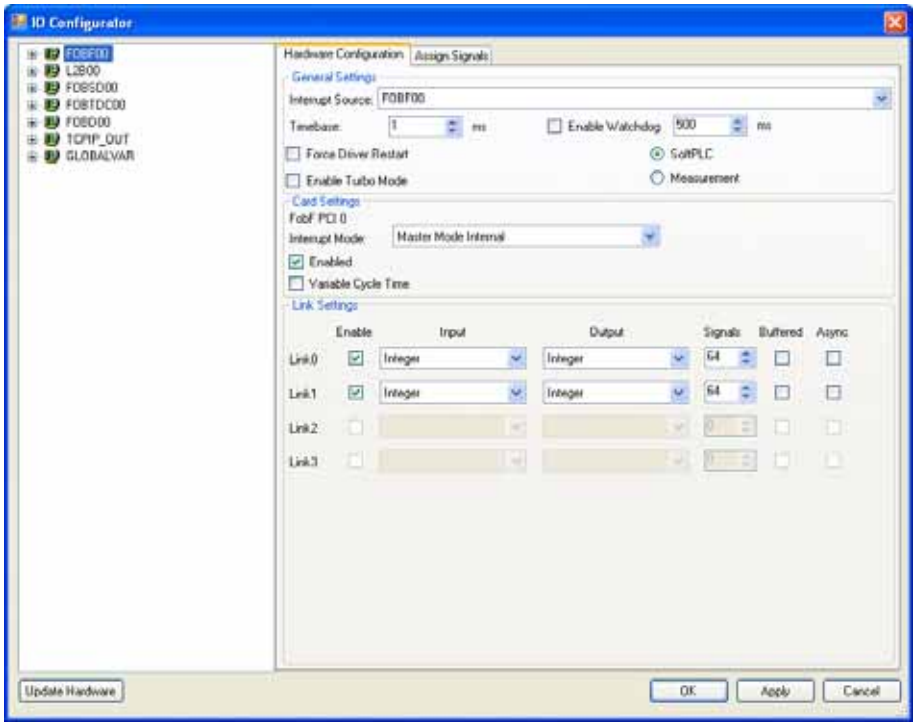
9 Конфигурация ввода-вывода

Конфигуратор ввода-вывода - это основное диалоговое окно, в котором пользователь может сконфигурировать параметры, имеющие отношение к входным и выходным сигналам, а также к некоторым интерфейсам.

Примечание

Исключение составляют те интерфейсы, которые представляют собой функциональные блоки, например блок TCPIP_SENDRECV или интегрированный OPC-интерфейс.

➲ Откройте конфигуратор ввода-вывода через меню "Инструменты".



Диалоговое окно конфигуратора ввода-вывода состоит из 3 разделов:

- Доступные ресурсы ввода/вывода
- Конфигурация аппаратного обеспечения
- Присвоение сигналов

Доступные ресурсы ввода/вывода

Все интерфейсы аппаратного и программного обеспечения, которые система распознает и поддерживает, отображаются в дереве элементов в левой части диалогового окна.

Конфигурация аппаратного обеспечения

В этом разделе можно сконфигурировать специальные настройки для всей системы и отдельных карт.

Присвоение сигналов

В ibaLogic пользователь может присваивать имена входам и выходам по своему усмотрению (виртуальные входы и выходы). В разделе "Присвоение сигналов" эти

виртуальные сигналы соотносятся с физическими входами и выходами. Виртуальные сигналы разделяются на группы.

9.1 Ресурсы

При открытии конфигурации ввода-вывода активного проекта она принимается конфигуратором ввода-вывода.

Примечание

Конфигурация ввода-вывода сохраняется не в базе данных, а в двух XML-файлах в папке "…\ibaLogic v4\Server\HwMappings". Соответственно, пользователь может редактировать проект, не принимая во внимание имеющиеся в наличии аппаратные средства.

При создании резервной копии базы данных файлы с конфигурацией ввода-вывода сохраняются в формате ZIP и заново загружаются при восстановлении базы данных. Благодаря этому даже после перемещения проектов исходная конфигурация ввода-вывода остается доступной.

Кнопка <Обновить апп. обесп.> (<Update Hardware>)

После щелчка по кнопке <Обновить апп. обесп.> аппаратное обеспечение, доступное компьютеру, на котором работает PMAC, будет принято программой. К таким аппаратным средствам относятся PCI-карты, которые поддерживает платформа Windows PC (см. раздел 9.1.1, "Hardware Resources"), а также периферийные модули платформы ibaPADU-S-IT.

Важно

Перед редактированием конфигурации ввода-вывода нужно сконфигурировать платформу, поскольку параметры, заданные в конфигураторе ввода-вывода, будут потеряны при переключении платформы.

Примечание

Количество сигналов ввода-вывода зависит от приобретенной лицензии (в аппаратном ключе).

Дерево элементов

Необходимые условия

- ❑ Вы активировали соответствующее соединение в конфигурации аппаратного обеспечения и щелкнули кнопку <Принять> (<Accept>) в нижней части диалогового окна.

Последовательность действий

➲ Дерево элементов раскрывается вплоть до отдельных сигналов, для этого нужно щелкать по символам +/- перед именами элементов.

Отображаются следующие уровни иерархии:

Интерфейс → модули → входы / выходы → сигналы

Интерфейс: Имя и индекс для типов карт

Модули: Деление в соответствии с типом карты

Входы / выходы: У модуля могут быть только входы, только выходы или и то, и другое

Сигналы: Имена сигналов аппаратного обеспечения создаются на базе имени модуля, направления, типа данных и серийного номера.

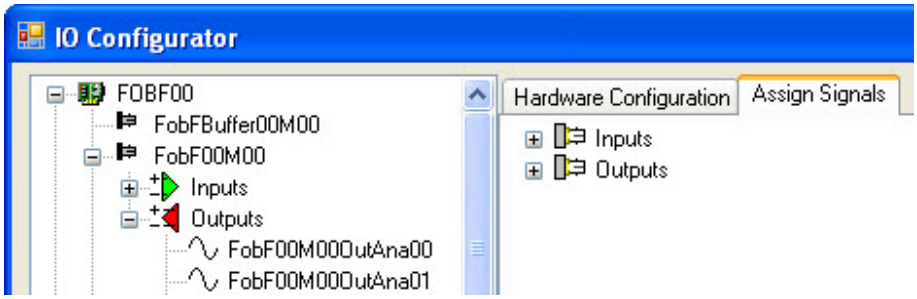


Рис. 95: Присвоение сигналов

9.1.1 Аппаратные ресурсы

ibaLogic поддерживает следующие интерфейсы:

Платформа WinXP и PCI-карты

Интерфейс	Карты	Соединения	Протокол
FOBF*nn* [4]	ibaFOB-io-S ibaFOB-4i-S ibaFOB-4o-S	1 канал OFC 4 канала OFC (симплекс/дуплекс)	ibaNet, 2 и 3,3 Мбит
	ibaFOB-2io-X ibaFOB-4i-X ibaFOB-4o-X	2 канала OFC 4 канала OFC (симплекс/дуплекс)	ibaNet, 3,3 и 32 Мбит (32 Мбит только получение)
FOBD*ii*	ibaFOB-2io-D ibaFOB-4io-D ibaFOB-4o-D	2 канала OFC 4 канала OFC (симплекс/дуплекс)	ibaNet, 2, 3,3 и 32 Мбит, а также DMA
FOBSD*nn*	ibaFOB-SD	1 канал OFC дуплекс (ST)	SIMADYN D
FOBTDC*nn*	ibaFOB-TDC	1 канал OFC дуплекс (SC)	SIMATIC TDC
L2B*nn*	ibaCOM-L2B 4/8 ibaCOM-L2B 8/8	4 ведомых устройства 8 ведомых устройств	Ведомые устройства Profibus DP

[4] Карты с таким интерфейсом доступны только для старых систем.

Интерфейс	Карты	Соединения	Протокол
SST_Master*nn*	SST-PCB3	Макс. 125 ведомых устройств	Ведущее устройство Profibus DP
RFM*nn*	VMIC-5565 VMIC-5575	1 дуплексный OFC 1 коаксиальный	Reflective Memory

Рис. 96: Платформа WinXP

nn = нумерация карт от 00 до 03 (допускается макс. 4 карты одного типа)

ii = нумерация карт от 00 до 07 (допускается макс. 8 карт типа FOB-D).

Платформа PADU-S-IT

Интерфейс	Периферийные устройства
PADU-S-IT	Локальные периферийные устройства, т.е. интерфейсные модули в модульной стойке PADU-S. См. информацию о модулях PADU-S.

Дополнительная документация - платформа PADU-S-IT

См. руководство для устройства ibaPADU-S-IT.

9.1.2 Программные ресурсы

ibaLogic поддерживает следующие протоколы на базе Ethernet:

Отображение	Протокол
TCPIP_OUT	Протокол TCP/IP, передача данных ibaLogic - PDA: Класс WinXP: макс. 16 телеграмм, в каждой из которых 32 значений real и 32 бинарных значений Класс PADU-S-IT: макс. 8 телеграмм, в каждой из которых 32 значений real и 32 бинарных значений

Примечание

Доступны только выходы.

9.1.3 Глобальные системные переменные

Доступны следующие глобальные системные переменные:

Интерфейс	Переменные
GLOBALVAR	Глобальные входные переменные: SYSTEM_UTC_TIME Текущее системное время в формате UTC (UniversalTimeCoordinated). DONGLE_NUMBER Номер аппаратного ключа iba, который вставлен в компьютер, или серийный номер PADU-S-IT. ACQ_RESTART_COUNT Счетчик перезапуска внутреннего драйвера. Используется для детектирования перегрузок системы в режиме буферизации.

9.2 Конфигурация аппаратного обеспечения

Диалоговое окно конфигурации аппаратного обеспечения вызывается через вкладку "Конфигурация аппаратного обеспечения" ("Hardware Configuration").

В этом диалоговом окне содержатся 3 раздела, в которых можно настроить следующее:

- Общие настройки ibaLogic
- Настройки карты
- Настройки соединения

Примечание

Пользователь может вносить изменения, только если вычисление еще не началось (серый фон области программирования).

9.2.1 Общие настройки

Общие настройки относятся к исполнительной системе ibaLogic и ко всем интерфейсным картам.

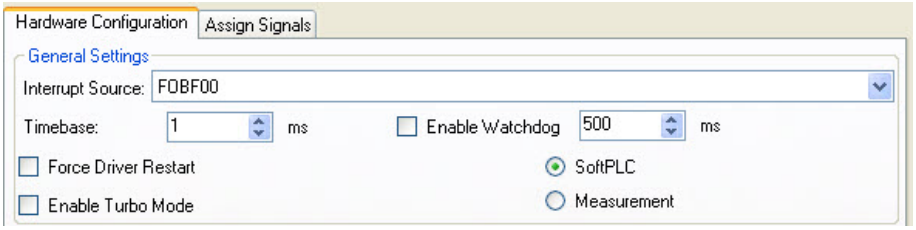


Рис. 97: Общие настройки

Источник прерываний

В строке с раскрывающимся списком перечисляются все доступные модули iba. В этом списке выберите модуль, который должен использоваться как источник прерываний для шины PCI.

Если в системе нет карт ввода-вывода, то используется тактовая частота ibaLogic, а поле выбора карты остается пустым.

Для платформы PADU-S-IT в этом случае существует только один вариант - PADU-S-IT.

Опорное время

Опорное время - это наименьшее возможное время цикла. Обратите внимание на то, что интервалы задач не могут быть меньше значения опорного времени.

Активация режима Turbo

Режим turbo можно активировать, если используется многопроцессорный компьютер. Благодаря этому можно значительно увеличить производительность системы, поскольку один из процессоров будет отвечать только за работу PMAC, а второй - за работу Windows. Это особенно важно для осуществления функций управления и регулирования.
Более подробная информация содержится в пункте 11.2**Fehler! Verweisquelle konnte nicht gefunden werden.** "Время отклика".

Программный контроллер (Soft PLC)

Этот режим используется для осуществления функций управления и регулирования. В этом режиме в ibaLogic обрабатываются только актуальные значения сигналов. В отличие от режима измерения в данном случае потеря некоторых значений не играет роли. Для вычисления используются текущие данные из последнего цикла передачи входов-выводов.

Более подробная информация содержится в пункте 11.2**Fehler! Verweisquelle konnte nicht gefunden werden.** "Время отклика".

Измерение

Использование этого режима гарантирует, что ibaLogic не потеряет ни одного входного значения. Это касается также случаев, когда необходимо заменить отдельные задачи в ibaLogic. Исполнительная система ibaLogic обеспечивает поступление данных через равные промежутки времени с заданным интервалом задачи. Если происходит замена задачи, то система компенсирует время цикла.

Более подробная информация содержится в пункте 11.2**Fehler! Verweisquelle konnte nicht gefunden werden.** "Время отклика".

Активация сторожевой схемы

Если эта функция активирована, то в существующих картах устанавливается таймер, который запускается программой ibaLogic при выполнении записи на выходах. Если за установленное время от ibaLogic не поступает команда записи (= триггер), карта автоматически сбрасывает все выходы на 0.

Поскольку ibaLogic записывает выходы циклически, любое срабатывание сторожевой схемы указывает на приложение, которое приостановлено или перегружено (например, бесконечным программным циклом).

Принудительный перезапуск драйвера

Эта функция имеет особое значение для SST-карт и карт reflective memory. При перезапуске драйвера выполняется сброс этих карт и принимается измененная внешняя конфигурация.
Выбор этой опции нужно подтвердить, щелкнув <OK>. Затем она автоматически сбрасывается.

9.2.2 Настройки карты

В левой части диалогового окна в дереве элементов выберите соответствующий интерфейс. Ниже описаны только те настройки, которые применимы к (почти) всем картам.

- Режим прерываний

Это поле относится только к картам производства iba.

Следующие режимы доступны для выбора:

- MasterModeInternal:
  Режим MasterModeInternal может быть установлен только для одной карты. В этом режиме для создания синхронизационного сигнала используется внутренний таймер. Синхросигнал поступает на остальные карты по плоскому ленточному кабелю.
- MasterModeExternal:
  В отличие от MasterModeInternal, в этом режиме синхросигнал не генерируется картой, а вычисляется на основе цикла OFC-телеграмм, поступающих на Link0.
  Этот режим имеет смысл использовать только тогда, когда требуется синхронизация ibaLogic с внешним циклом, например переменным периодом прерываний в буферном режиме (например, измерения планшетности или измерения, синхронизированные с шиной, с использованием устройства мониторинга шины Simolink). Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".
- SlaveMode:
  Этот режим должен быть установлен для всех остальных карт.

Примечание

Если в вашей конфигурации есть карта типа FOBSD или FOBTDC, то выберите эту карту в качестве ведущего по прерываниям. В противном случае выберите одну из карт типа FOB-X или FOB-D. Если ни одной из вышеуказанных карт нет в системе, можно использовать карту FOB-S или L2B.

- Активные

Вы можете использовать только активные карты.

Если сервер ibaPDA работает на компьютере, где используется карта, то эту карту необходимо деактивировать. Одна карта не может одновременно использоваться ibaLogic и ibaPDA.

Остальные настройки связаны с конкретным типом карты и описаны в разделе 9.4 "Интерфейсы PCI (ПК с ОС Windows)".

9.3 Присвоение сигналов

Для того чтобы использовать физические входы и выходы, их необходимо соотнести с виртуальными сигналами в программе.

Существует два способа распределения сигналов:

- от аппаратного сигнала к программному (с точки зрения аппаратного обеспечения)
- от программного сигнала к аппаратному (с точки зрения программы)

Можно также использовать сочетание обоих этих способов.

9.3.1 Способ от аппаратного сигнала к программному

Прежде чем начать программировать, у вас уже есть информация о внешних интерфейсах, например распределении OFC-телеграмм или телеграмм от ведомых устройств Profibus. Исходя из этих физических сигналов, вы создаете виртуальные сигналы, которые впоследствии будут использоваться в программе.

Можно оставить автоматически сгенерированные имена сигналов или присвоить собственные. Тэг направления, тип и индекс присваиваются автоматически. Помимо этого каждый сигнал можно редактировать отдельно.

После того как все сигналы будут приняты программой, они будут отображены в области навигации под "Входы - Выходы" ("Inputs -- Outputs"). В области навигации имена сигналов также можно редактировать, если они еще не используются в программе, т.е. не были перемещены к границам области программирования.

Пример: присвоение всех сигналов карты ibaFOB-io-S

Карта FOB находится в списке элементов в левой части диалогового окна.

Во вкладке "Присвоение сигналов" ("Assign Signals") входы и выходы пока не присвоены.

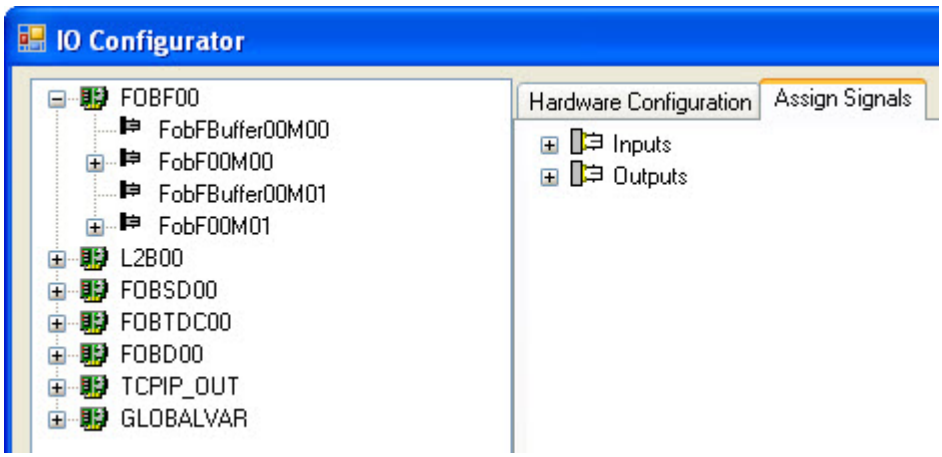


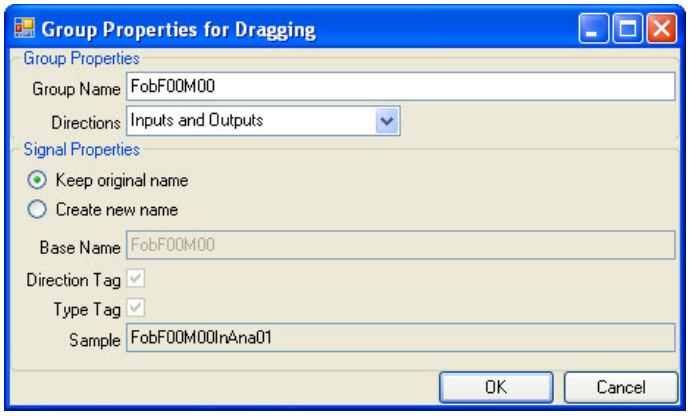
Рис. 98: Присвоение сигналов карты ibaFOB-io-S

Последовательность действий

1. Разверните дерево элементов FOB-карты, которое находится в левой части окна.
   Появится список всех активных соединений карты с их обозначениями.

Вы можете присвоить сразу весь модуль FOBF00M00 (соответствует link 0 FOB-карты).

2. Перетащите модуль FOBF00M00 в правую часть ко входам или выходам.
   Появится диалоговое окно "Свойства группы для перетаскивания" ("Group Properties for Dragging").



Комментарий

Под входами ibaLogic создает группу и присваивает ей имя, а также генерирует виртуальное имя для каждого аппаратного сигнала. Имена могут быть как присвоены пользователем, так и автоматически созданы ibaLogic.
При создании имен сигналов отображается структура текущего имени сигнала. В примере выше имя "FOBF00M00InAna01" образовано следующим образом:

FOBF00M00	базовое имя (совпадает с именем группы)
In	тэг направления
Ana	тип сигнала
01	порядковый номер

В зависимости от направления, выбранного в окне выбора, все сигналы этого модуля создаются в группе под входами и/или выходами.

Результат

В результате присвоения сигналов получается следующее:

Виртуальные сигналы отображаются слева от стрелки, а аппаратные - справа. Имя группы аналогично имени физического модуля.

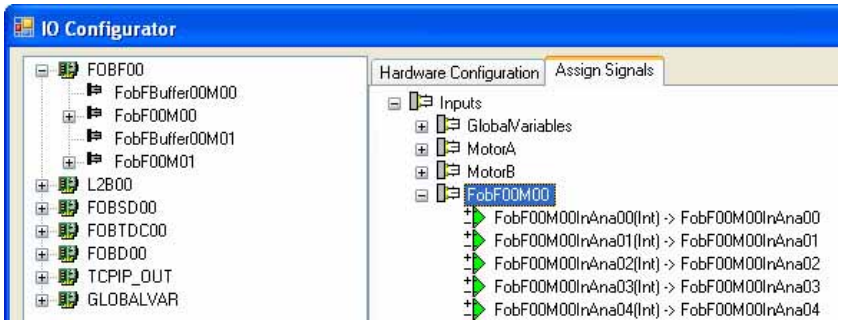


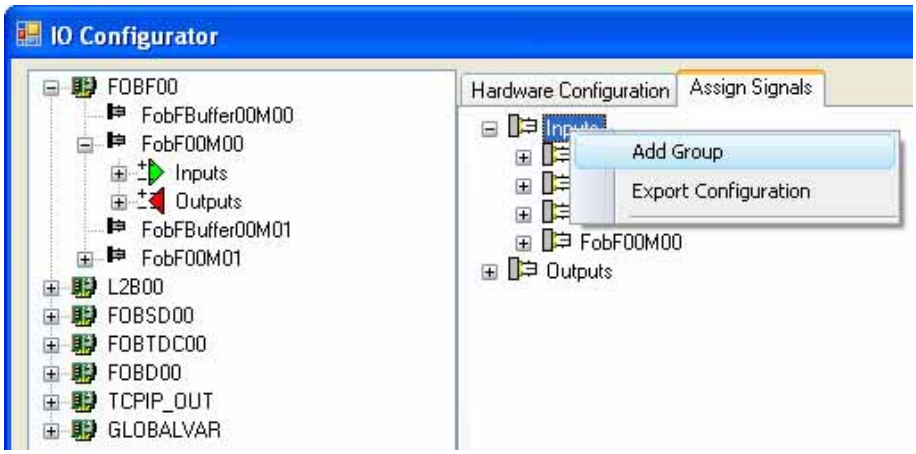
Рис. 99: Присвоение сигналов

Пример: присвоение отдельных сигналов карты ibaFOB-4i-S или ibaFOB-4o-S

Если вам требуется всего несколько сигналов от модуля в программе, вы можете присвоить только эти сигналы.

Добавление группы

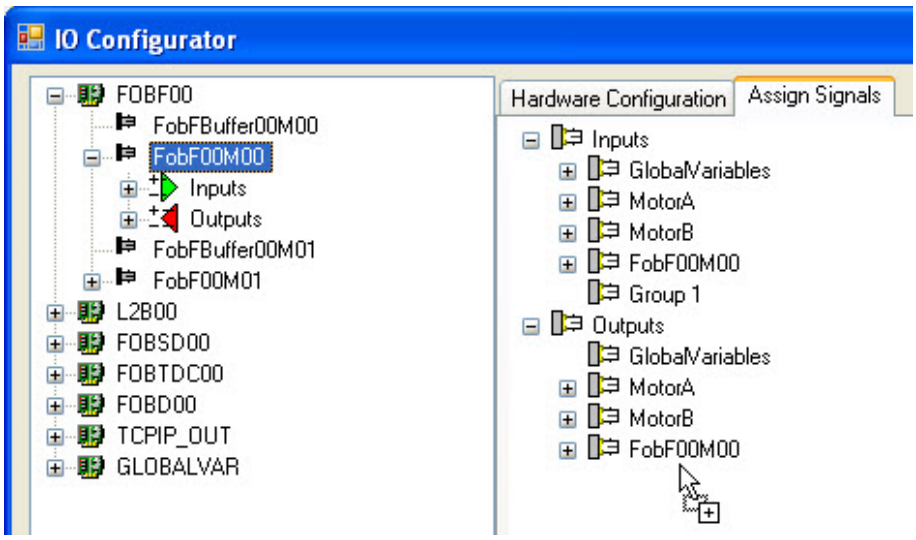
1. Откройте контекстное меню щелчком правой кнопкой мыши.



2. Выберите пункт "Добавить группу".

Последовательность действий

1. Выделите входы или выходы.
2. Вы также можете перетащить отдельные сигналы в ранее созданную группу.



Результат

Отдельные сигналы присвоены группе.

Изменение имени сигнала

Вы можете изменить имена виртуальных сигналов после их присвоения.

Последовательность действий

1. Вы можете изменить имя группы, щелкнув по ней правой кнопкой мыши и выбрав меню "Свойства".
2. Вы можете изменить имя сигнала, щелкнув по нему двойным щелчком. Вы можете также добавить описание для сигнала, которое будет отображаться в программе в виде подсказки, появляющейся при наведении курсора на сигнал.
3. Подтвердите настройки, нажав <OK> или <Применить>.

Результат

Имя сигнала изменено.

9.3.2 Способ от программного сигнала к аппаратному

У вас пока нет информации о внешних интерфейсах, однако вы хотите приступить к программированию и выяснить, какие входы и выходы вам нужны.

1. В области навигации определите входы и выходы, группы и сигналы. Чтобы получить более подробную информацию по этому поводу, см. описание в разделе 4.6 "Конфигурация входов и выходов".
2. Когда у вас будет информация о физических интерфейсах, имеющих сигналы ввода-вывода, вы сможете перейти в конфигуратор ввода-вывода и присвоить сигналы физическим интерфейсам.

Пример: сигналы карты ibaFOB-4io-S (модуль полностью)

Вы присваиваете физические сигналы link0 FOB-карты.

Необходимое условие

В области навигации вы создали группу "ДВИГАТЕЛЬ_1" ("MOTOR_1"), а под ней - входные сигналы "Двигатель_1_скорость" ("Motor_1_speed") и "Двигатель_1_крутящий момент" ("Motor_1_torque").

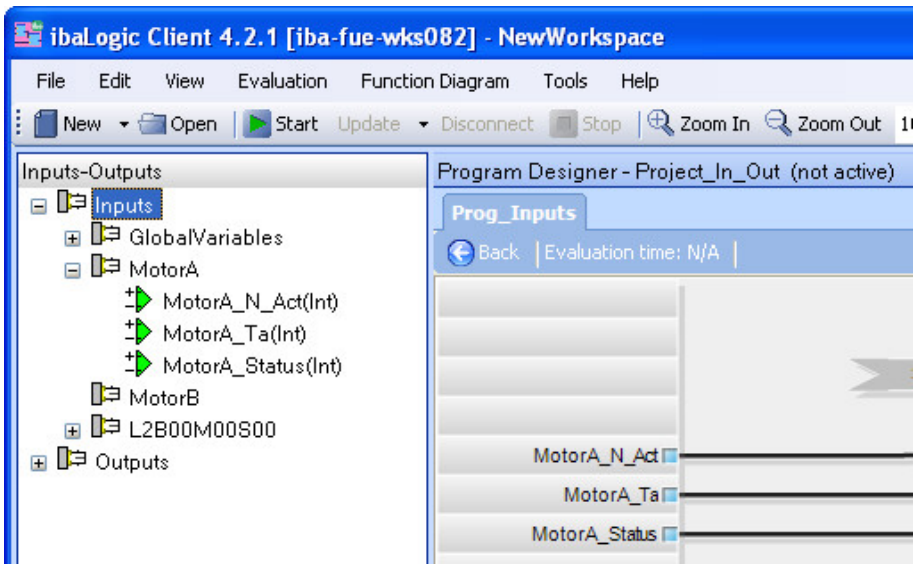
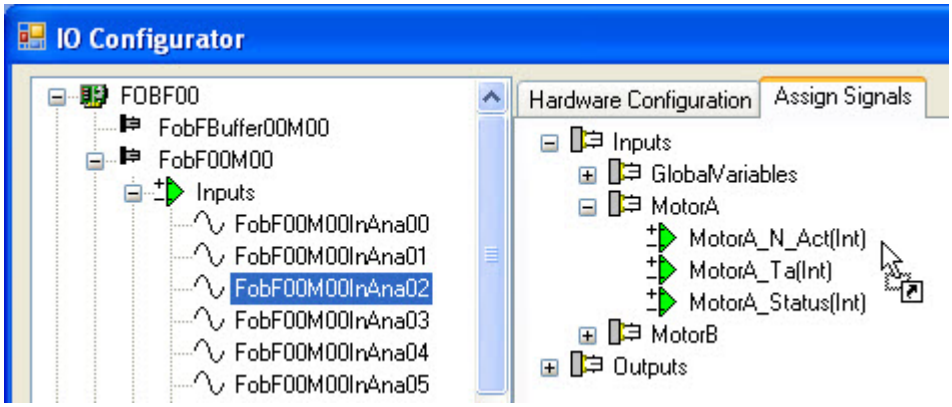


Рис. 100: "Входы – Выходы"

Последовательность действий

1. Откройте конфигуратор ввода-вывода.
2. Выполните обновление аппаратного обеспечения, нажав кнопку <Обновить апп. обесп.> (<Update Hardware>).
3. Активируйте link0 на FOB-карте.
   Обращаем ваше внимание на то, что тип данных соединения совпадает с типом данных уже заданных сигналов.
   Список сигналов, созданных в проекте, можно посмотреть во вкладке "Присвоение сигналов" ("Assign Signals").
4. В дереве элементов в левой части диалогового окна выберите аппаратный сигнал.

   Перетащите его в правую часть окна.
   Таким образом виртуальный сигнал будет присвоен физическому.



Результат

Присвоенные сигналы также можно посмотреть в области программирования.

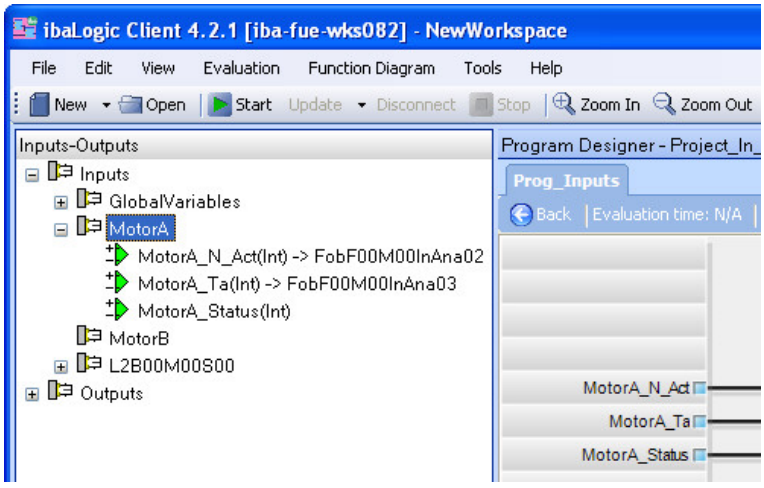


Рис. 101: Присвоение сигналов

9.3.3 Изменение назначения сигналов

Уже существующие назначения сигналов можно изменить несколькими способами.

Способ 1

1. Просто перетащите другой аппаратный сигнал на виртуальный сигнал, который уже был присвоен ранее.
   Его назначение изменится.

Способ 2

2. Перетащите аппаратный сигнал, который уже используется, на виртуальный сигнал.
   Появится информационное сообщение.
3. Подтвердите это сообщение, чтобы назначение сигналов было изменено программой и удалена связь с предыдущим сигналом.

Способ 3

4. Этот способ также можно использовать для изменения назначения целых модулей.

Примечание

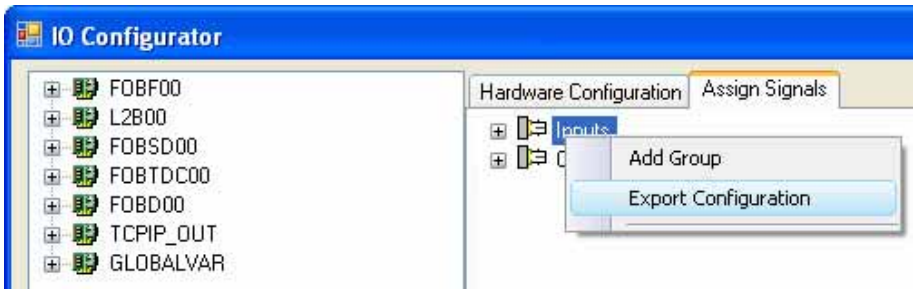
Одному аппаратному сигналу может быть присвоен только один виртуальный.

9.3.4 Использование имен сигналов из внешних источников

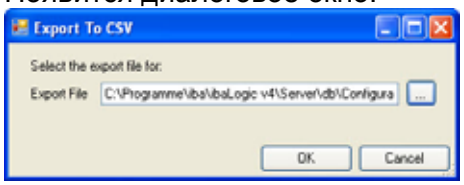
Функции экспорта / импорта позволяют использовать имена сигналов, которые содержатся во внешних источниках, например программах для работы с электронными таблицами.

Последовательность действий

1. Укажите аппаратные сигналы.
2. С помощью контекстного меню входов и выходов экспортируйте конфигурацию ввода-вывода.



3. В меню выберите "Экспортировать конфигурацию" ("Export Configuration").
   Появится диалоговое окно.



4. Укажите путь и имя файла.

Результат

Программа создаст файл CSV, который можно открыть с помощью редактора ASCII или программы для работы с электронными таблицами.

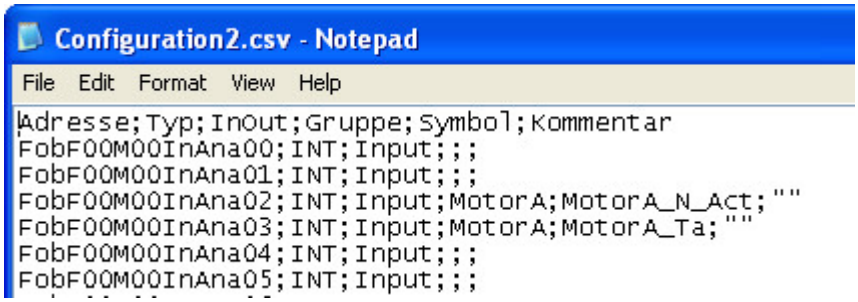


Рис. 102: Файл CSV в редакторе ASCII (блокнот)

Важно

Аппаратные сигналы содержатся в столбцах Адрес, Тип и ВхВых (InOut).

Их изменять нельзя.

Вы можете редактировать виртуальные сигналы в столбцах Группа, Символ и Комментарии или использовать копирование и вставку, чтобы добавить их из других документов.
Имена в столбце "Символ" должны соответствовать требованиям стандарта IEC.

5. Закройте конфигуратор ввода-вывода, чтобы импортировать данные.
6. В основном меню выберите "Файл – Импорт – Назначение сигналов" ("File – Import – Signal Assignment").

9.4 Интерфейсы PCI (ПК с ОС Windows)

В этом разделе описываются специальные настройки интерфейсных карт.

Более подробная информация содержится в пункте 9.1.1 "Аппаратные ресурсы".

9.4.1 Связь с "миром устройств iba"

Для OFC-соединения с распределенной периферией iba (PADU) и с интерфейсными картами iba существуют следующие PCI-карты. Эти карты содержатся в дереве ресурсов под интерфейсом **FOBFnn**.

- ibaFOB-S (OFC-соединение, протокол 3,3 Мбит) [5]
- ibaFOB-X (OFC-соединение, протокол 32 Мбит) [6]
- ibaFOB-D (OFC-соединение, протокол 2 Мбит / 5 Мбит / 3,3 Мбит / 32 Мбит)

Для каждого типа карты есть несколько вариантов соединений: 1, 2 или 4 для ввода и вывода.

Настройки карты

Если в дереве элементов выделен интерфейс FOBFxx, то в правой части отображаются соответствующие настройки карты.

- Режим прерываний, см. раздел 9.1.1
- Карта активирована, см. раздел 9.1.1
- Изменяемое время цикла

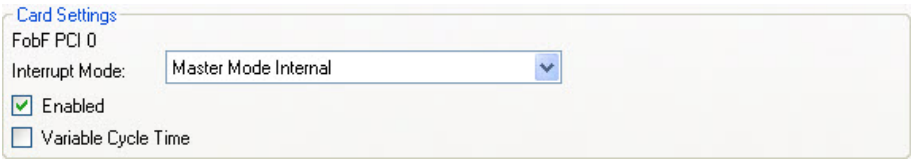


Рис. 103: Настройки карты

Окно выбора	Объяснение
Активирована	Могут использоваться только активированные карты.
Изменяемое время цикла	Эту опцию можно использовать, чтобы сделать опорное время (как правило фиксированное) изменяемым. Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".

[5] Карты с таким интерфейсом доступны только для старых систем.
[6] Карты с таким интерфейсом доступны только для старых систем.

Настройки соединения

Параметры канала отображаются в разделе "Настройки соединения". Количество каналов зависит от версии используемой карты ibaFOB-io.

Настройки	Объяснение
Активировать	Здесь можно активировать или деактивировать отдельные соединения.
Формат данных	Здесь задайте формат данных для входящих телеграмм. В зависимости от подключенных устройств, выберите один из типов данных: INTEGER, REAL или S5REAL. Если в вашей системе установлена карта FOB-X или FOB-D, то в качестве опции предлагаются дополнительные форматы данных для OFC-протокола 32 Мбит.

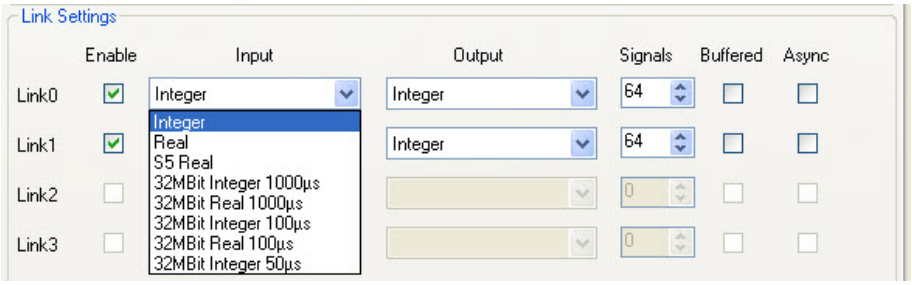


Рис. 104: Настройки соединения

Тип данных должен соответствовать типу данных, заданному для подключенного устройства.

Устройство	Протокол	Тип данных
ibaPDA ibaNet750 ibaDDSCM	3,3 Мбит	INTEGER
SIMATIC TDC/LO6	32 Мбит	FobX-Integer128
ibaLink-SM-64-i-o	3,3 Мбит	INTEGER, REAL или S5REAL, в зависимости от настроек переключателя на модуле.
ibaLink-SM-64-SD16 ibaLink-SM-128V-i-2 ibaLink-MBII-2io ibaBM-DPM-S-64	3,3 Мбит	INTEGER или REAL, в зависимости от настроек переключателя на модуле.
iba-PADU-S-IT ABB AC800PEC	32 Мбит	FobX-Real512 или FobX-Integer1024 FobX-Real512
ibaBM-DVN (DeviceNet) ibaBM-CAN (CAN bus) ibaBM-DPM-S (Profibus)	32 Мбит	FobX-Real512

Примечание

Если на стороне ввода используется протокол 32 Мбит, то связанный с ним канал вывода использоваться не может.

Настройки	Объяснение
Формат выходных данных	Тип данных OFC-телеграммы в направлении вывода. Этот параметр зависит от подключенного устройства (см. формат ввода).
Сигналы	Здесь ibaLogic автоматически указывает максимальное количество сигналов, которое зависит от выбранного формата данных. Количество сигналов можно уменьшить в соответствии с вашими требованиями. Это количество сигналов включает аналоговые и цифровые сигналы как в направлении ввода, так и в направлении вывода. После щелчка по кнопке <Принять> в дереве ресурсов под соответствующим интерфейсом создается модуль "FobFnnMxx", содержащий "n" аналоговых сигналов сконфигурированного типа и "n" цифровых (nn = индекс карты, xx = номер канала).
Режим буферизации (Buffered mode)	С помощью этой опции можно активировать режим для соединения, в котором полученные данные также доступны как массивы. Более подробно этот режим описан ниже.

9.4.2 "Режим буферизации" FOB-карт

Режим буферизации необходим в следующих случаях:

- Минимально возможный интервал в ibaLogic-V4 составляет 1 мс. Если требуется получать значения сигналов с циклом меньше 1 мс, то эти значения должны записываться в режиме буферизации. Прежде всего, этот режим должен использоваться в контексте следующих модулей:
  - Соединение с ibaPADU-S-IT (в режиме ввода-вывода)
  - Соединение локальных периферийных устройств платформы PADU-S-IT
- Даже для периферийных сигналов, которые поступают с циклом 1 мс, но не могут вычисляться за интервал 1 мс в связи с большим их количеством и требуемой вычислительной мощностью.

Считываемые значения помещаются в буфер и поступают в программу как массивы. Этот режим используется с целью избавить программу от необходимости вычислять отдельные сигналы в тех случаях, когда в этом нет необходимости.

Пример: Сигналы нужны только для FFT-анализа. В этом случае невозможно (период дискретизации < 1 мс) и не имеет смысла получать значения отдельных сигналов и организовывать их в массивы, чтобы выполнить FFT-анализ. В режиме буферизации создаются массивы заданного размера, в связи с чем программа не тратит машинное время на обработку отдельных значений.

Режим буферизации имеет следующие свойства:

- Дискретизация данных, получаемых по FOB-соединению, выполняется драйвером с частотой, определяемой заданным опорным временем.
- Значения каждого сигнала помещаются в кольцевой буфер. Размер циклического буфера и частота дискретизации для его заполнения устанавливаются в выходных ресурсах программы (см. ниже).
- Каждый сигнал доступен в программе в виде массива с длиной 256. Программа работает с более длинным интервалом выполнения задач и считывает целые массивы.
- Таким образом, программа не может обрабатывать отдельные значения, а только массивы значений, например для выполнения FFT-анализа или для архивирования.
- Пример: несмотря на частоту дискретизации 1 мс и интервал задачи 50 мс, вы все равно можете выполнять FFT-анализ на основе 128 значений.
- Параллельно с буферизацией данных также может выполняться генерирование отдельных сигналов для других программ.

Примечание

Использование режима буферизации возможно только в режиме дискретизации "Измерение" ("Measurement").

В этом случае можно также использовать функцию **изменяемого времени цикла**, с помощью которой вы можете настроить изменение опорного времени драйвера в соответствии со скоростью процесса. В качестве примера можно привести мониторинг и оценку планшетности полосы. Поскольку ленты конвейера двигаются с разной скоростью, но для выполнения FFT-анализа требуется одинаковое количество значений по каждому участку, то период дискретизации должен динамически изменяться в зависимости от скорости конвейера.

Изменяемое время цикла имеет смысл использовать только:

- В режиме дискретизации "Измерение".
- В режиме "буферизации" соединения.
- В режиме прерываний "MasterModeExternal".
- Если оптоволоконное кольцо установлено на канале 0 (link 0).
  Иными словами, если сигнал с выхода канала 0 поступает на каскад PADU и затем возвращается на вход канала 0.

Только при такой настройке возможно обеспечить то, что оптическая телеграмма будет передана и возвращена в рамках сконфигурированного цикла, и прерывание сгенерировано.

Примечание

При использовании изменяемого времени цикла интервал задачи представлен не миллисекундами, а "количеством прерываний". Другими словами, если опорное время было сконфигурировано как 1500 мкс, а интервал задачи составляет 10, это соответствует интервалу 15 мс. Использование этого режима имеет смысл только в режиме дискретизации "Измерение" ("Measurement").

Ресурсы ввода

После щелчка по кнопке <Принять> в дереве ресурсов под соответствующим интерфейсом создается модуль "FobBuf*nn*M*xx*" (nn = индекс карты, xx = номер канала) для каждого канала, активированного в режиме буферизации.

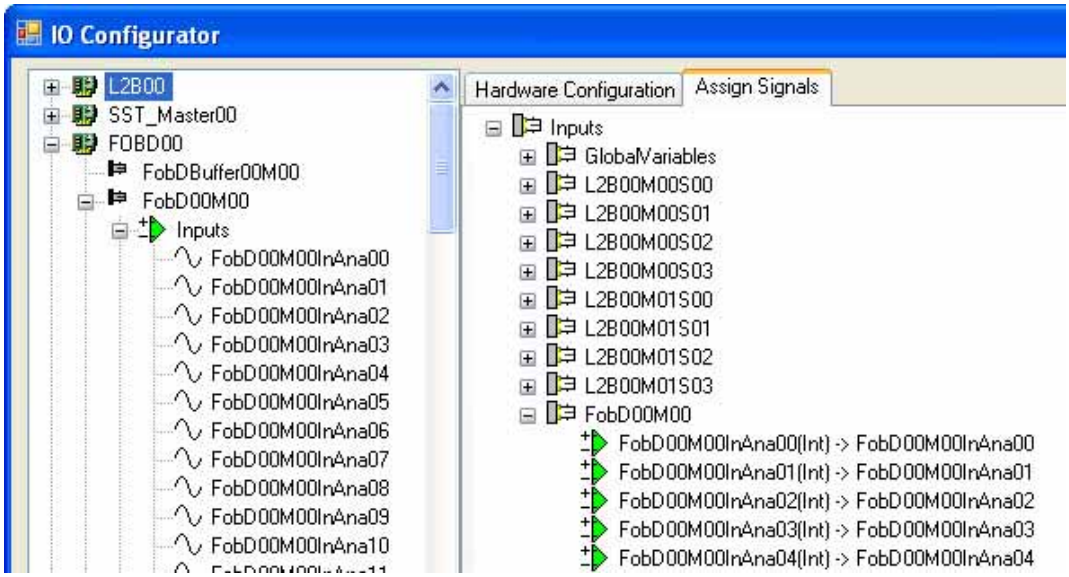


Рис. 105: Назначение входных сигналов

Входные сигналы	Объяснение
CurDataSize	Обратная связь о сконфигурированном размере буфера в выходном сигнале.
FillCount	Счетчик, значение которого увеличивается, когда входной массив полностью заполнен в соответствии с "размером данных" ("Datasize").
BufAna*ii*	Массив типа integer или real длиной 256.
BufDig*00*	Массив типа Boolean длиной 256.
Только для изменяемого времени цикла	
CurCycleTime	Обратная связь о сконфигурированном цикле прерываний в выходном сигнале.

Ресурсы вывода

Ресурсы модуля типа "FobFBuffer*nn*M*xx*" (nn = индекс карты, xx = индекс канала).

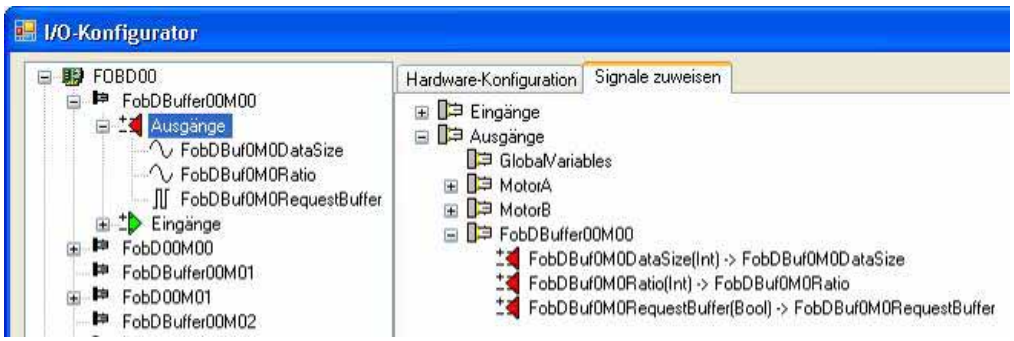


Рис. 106: Назначение ресурсов вывода

Выходные сигналы	Объяснение
Размер данных (Datasize)	Количество сигналов для измерения, которое помещается драйвером в буфер, прежде чем значение счетчика увеличится ("уровень заполнения" - "Filling level"). Допускаются значения до 256.
Коэффициент (Ratio)	Целочисленный множитель опорного времени, на основе которого происходит заполнение буфера. Например, коэффициент=2 означает, что только каждое 2-е значение сигнала будет помещаться в буфер.
RequestBuffer	Управление дискретизацией. Буферы заполняются только в том случае, если выход имеет значение "TRUE"
Только для изменяемого времени цикла	
Время цикла (CycleTime)	Изменяемый период прерываний. Указывается в микросекундах. Допустимые значения: 1000 … 10000.
SetCycleTime	Сигнал для принятия времени цикла (передний фронт).

9.4.3 ibaLogic как ведомое устройство Profibus

Сеть Profibus организована по принципу обмена данными между ведущим и ведомыми устройствами или несколькими ведущими устройствами. В соответствии со стандартом DP-V0 обмен данными осуществляется только между ведущим устройством и ведомыми, при этом соединение устанавливается и контролируется ведущим устройством. ibaLogic может выполнять функции как ведущего устройства, так и ведомого.

Например, чтобы соединиться с системой SIMATIC S7, которая является ведущим устройством, ibaLogic должна работать как ведомое устройство. Для этой цели используется карта ibaCOM-L2B.

Под интерфейсом **L2B*nn*** располагаются следующие интерфейсные карты:

- ibaCOM-L2B 4/8 (карта ведомого устройства Profibus DP с 1 разъемом, можно подключить 4 ведомых устройства)
- ibaCOM-L2B 8/8 (карта ведомого устройства Profibus DP с 2 разъемами, можно подключить 8 ведомых устройств)

Настройки карты

Если в дереве элементов выделен интерфейс L2B*nn*, то в правой части отображаются соответствующие настройки карты.

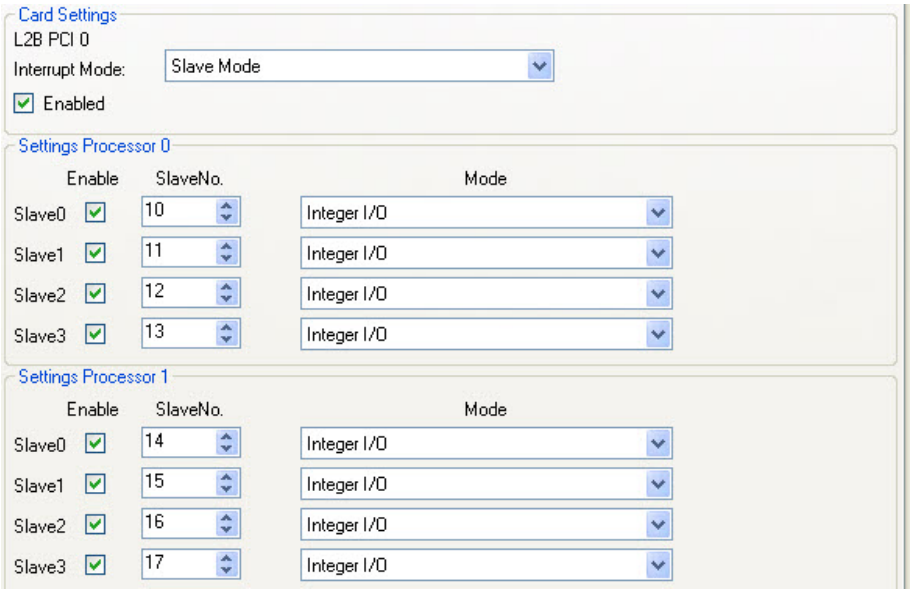


Рис. 107: Настройки карты для ibaCom-L2B

- ❑ Режим прерываний, см. раздел 9.1.1
- ❑ Карта активирована, см. раздел 9.1.1

Настройки интерфейса шины 0/1

Карта L2B оборудована 1 или 2 интерфейсами Profibus DP. Для каждого интерфейса можно использовать 4 ведомых устройства.

Настройки	Объяснение
Активировать	Здесь можно активировать или деактивировать отдельные ведомые устройства.
Номер ведом. устр-ва	Каждому активному устройству нужно присвоить индивидуальный номер станции.
Режим	Для каждого активного ведомого устройства определите формат телеграмм, который соответствует конфигурации ведущего устройства Profibus. Для этого iba предоставляет несколько GSD-файлов, которые соответствуют формату телеграмм, которые доступны для выбора.

Формат телеграммы	GSD-файл iba	Содержимое	Направление данных
Integer_In	iba_0F01	32 целых и 32 бинарных сигнала	Ведущ. устр-во → ibaLogic
Real_In	iba_0F02	32 сигнала real и 32 бинарных сигнала	Ведущ. устр-во → ibaLogic
S7Real_In	iba_0F04	28 сигналов real и 32 бинарных сигнала	Ведущ. устр-во → ibaLogic

Integer_inOut	iba_0F08	32 целых и 32 бинарных сигнала	Ведущ. устр-во ←→ ibaLogic
Real_InOut	iba_0F09	32 сигнала real и 32 бинарных сигнала	Ведущ. устр-во ←→ ibaLogic
S7Real_InOut	iba_0F0B	28 сигнала real и 32 бинарных сигнала	Ведущ. устр-во ←→ ibaLogic

После щелчка по кнопке <Принять> в дереве ресурсов под соответствующим интерфейсом для каждого ведомого устройства создается модуль "L2BnnMyySxx", который содержит аналоговые сигналы заданного типа и 32 бинарных сигнала (nn = индекс карты 00-03, yy = номер интерфейса 00-01), xx = номер ведомого устройства 00-03).

9.4.4 ibaLogic как ведущее устройство Profibus

Если вы хотите, например, обратиться к станции ET2000 из ibaLogic, то она должна выполнять функции ведущего устройства. Для этого в компьютере с ibaLogic должна быть установлена карта ведущего устройства Profibus.

Под интерфейсом **SST_Master*nn*** располагается следующая интерфейсная карта:

- ❑ SST-PFB3-PCI (карта ведущего устройства Profibus DP, макс. 125 ведомых устройств)

Примечание

Для использования SST-карты необходима лицензия.

Краткое описание

Поскольку эта карта является разработкой другого производителя, ее конфигурирование и пользовательская настройка отличаются от конфигурирования и настройки карт iba.

В целом, необходимо создать конфигурацию Profibus, содержащую параметры, которые нужны для всех подключенных к Profibus станций, а также для обмена данными. Программа для создания конфигурации (SST Profibus Console) генерирует бинарный файл с параметрами (.bss). Этот файл нужно загрузить в SST-карту с помощью конфигуратора ввода-вывода.

Настройки карты

Если в дереве элементов выделен интерфейс SST_Master*nn*, то в правой части отображаются соответствующие настройки карты.

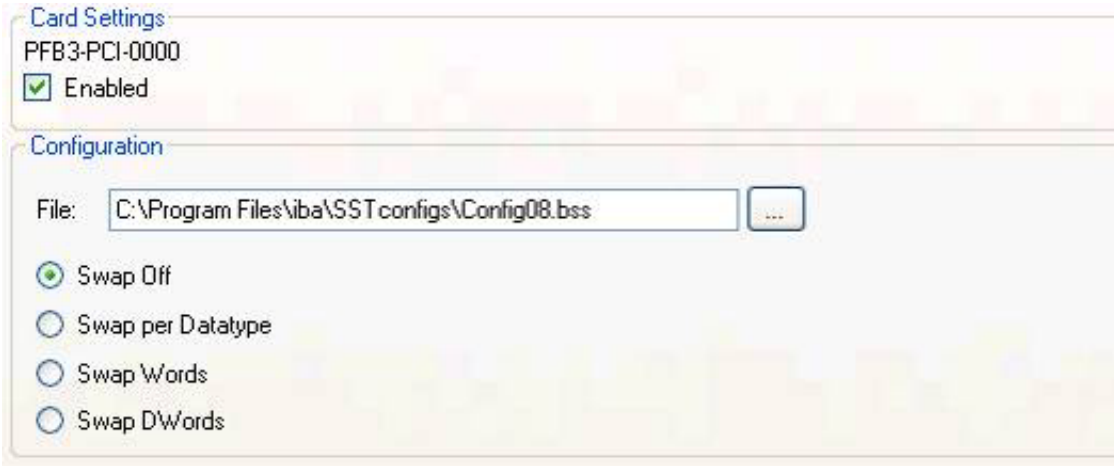


Рис. 108: Настройки карты

"Активирована" ("Enabled")

Эта опция используется для активации или деактивации карты.

Конфигурация

"Файл" ("File")

Здесь требуется указать файл, созданный посредством вышеупомянутой консольной программы. Вы можете щелкнуть по кнопке справа от строки ввода, чтобы отобразился раскрывающийся список, в котором можно выбрать нужный файл.

"Режимы своппинга" ("Swap modes")

В зависимости от подключенного устройства может понадобиться использовать свопиинг, т.е. изменение порядка байтов в типе данных, чтобы ibaLogic могла считать и обработать данные в соответствии со стандартом IEC.

Пояснение касательно режимов своппинга (каждая буква соответствует одному байту, пробелы добавлены для наглядности):

Способ своппинга	Исходные данные	Результат
Тип данных	"ABCD EF G H IJKL"	"DCBA FE G H LKJI"
Слово	"ABCD EF G H IJKL"	"BADC FE H G JILK"
Двойное слово	"ABCD EF G H IJKL"	"DCBA HG F E LKJI"

Особенности, связанные с назначением сигналов

Если вы щелкнули кнопку <Принять> и тем самым активировали настройки, содержащиеся в файле с конфигурацией, то в дереве ресурсов под SST_Master появится модуль с именем SST-карты - "PFB3-PCI-000*x*". Под картой отображаются следующие сигналы для всех возможных ведомых устройств Profibus.

Направление	Сигнал	Значение
Ввод	SST*nn*InStatus*yyy*	Информация о состоянии и диагностическая информация по телеграммам, полученным от ведомого устройства *yyy*. Тип данных: "SSTSTATUSSTRUCT"

Вывод	SST*nn*OutStatus*yyy*	Информация о состоянии и диагностическая информация по телеграммам, переданным ведомому устройству *yyy*. Тип данных: "SSTSTATUSSTRUCT"
Ввод	SST*nn*StructIn*yyy*	Данные, полученные от ведомого устройства *yyy* Тип данных: "SST_Struct"
Вывод	SST*nn*StructOut*yyy*	Данные, переданные ведомому устройству *yyy* Тип данных: "SST_Struct"

nn = индекс карты: 00-03, *yyy* = номер ведомого устройства Profibus: 002-127

Вы можете создавать виртуальные сигналы только для целого модуля (см. раздел 9.3 "Присвоение сигналов") или только для отдельных сигналов состояния, входных и выходных сигналов ведомых устройств Profibus, которые вы сконфигурировали и настроили.

Примечание

Создаваемые здесь сигналы имеют структурный тип данных.

Структура состояния "SSTSTATUSSTRUCT", генерируемая устройством, состоит из числа (ErrorCode - код ошибки) и текстовой строки (ErrorString - текст ошибки).

Тип данных "SST_Struct" представляет собой заполнитель для структур телеграмм с пользовательскими данными, которые нужно создать. Эта структура должна в точности соответствовать передаваемой или получаемой телеграмме Profibus, которая была определена в конфигурации Profibus (Profibus Console) для каждой станции. Более подробную информацию о структуре телеграмм Profibus (L2B) вы можете найти в руководстве для карт L2B.

Более подробная информация содержится в пункте C "Типы данных".

Важно

Пример подключения ведущей карты Profibus содержится в документации на CD, который входит в объем поставки.

9.4.5 Соединение SIMADYN D / SIMATIC TDC

iba разработала две PCI-карты для подключения к этим системам. Эти карты отличаются друг от друга только OFC-интерфейсом и протоколами.

- Карта ibaFOB-SD располагается под типом интерфейса **FOBSD**. Эта карта обеспечивает доступ к контроллерам SIMADYN D посредством соединительных модулей CS12, CS13 и CS14, а также к стойке SIMATIC TDC посредством коммуникационного модуля CP53M0.
- Карта ibaFOB-TDC располагается под типом интерфейса **FOBTDC**. Эта карта обеспечивает доступ к контроллерам SIMATIC TDC посредством GDM (Global Data Memory, интерфейсный модуль CP52IO).

Примечание

Настройки модулей FOBSD и FOBTDC идентичны, поэтому в настоящем руководстве объяснения касательно них даны на примере карты ibaFOB-SD.

Настройки карты

Если в дереве элементов выделен интерфейс FOBSD, то в правой части отображаются соответствующие настройки карты.

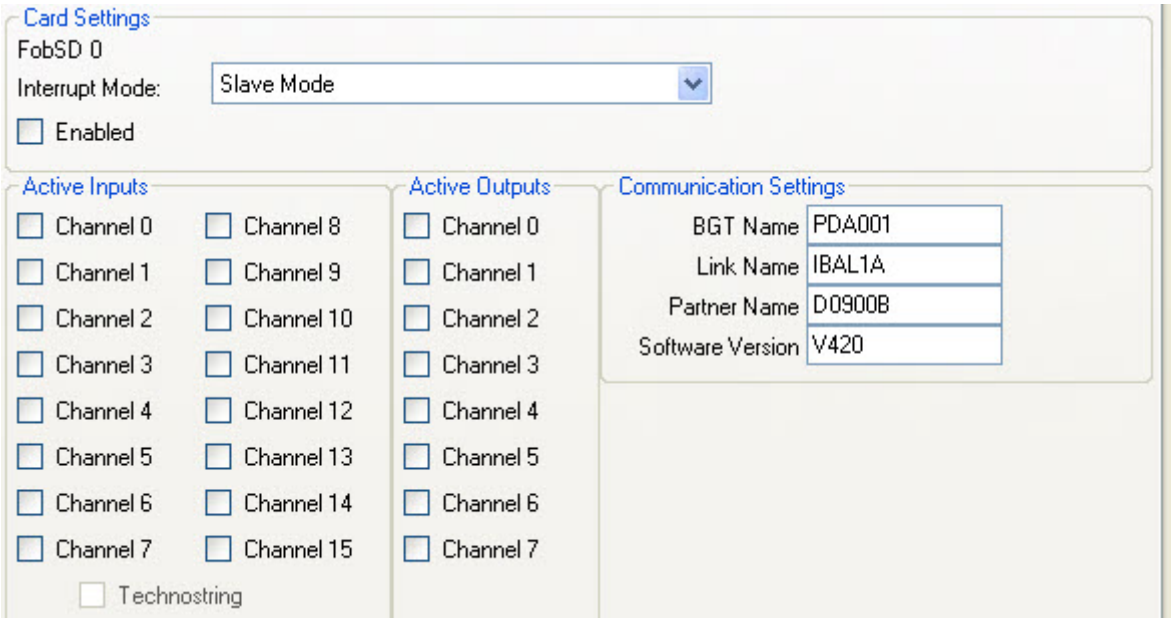


Рис. 109: Настройки карты для ibaFOB-SD/TDC

- Режим прерываний, см. раздел 9.1.1
- Активирована, см. раздел 9.1.1

Настройки соединения

Активные входы

Выберите один или несколько входов из имеющихся 16 каналов.
Обратите внимание на то, что для каждого выбранного вами канала ввода в SIMADYN D или SIMATIC TDC должна быть создана телеграмма передачи данных. Одна телеграмма содержит 32 значения real (для SIMADYN D тип данных NF) и 32 бинарных значения (1 DWORD или значение V4).

Для модуля передачи данных нужно сконфигурировать следующие параметры:

- AT (имя канала): MxPDADAT (x = 0 … F для каналов от 0 до 15)
- MOD (режим канала): R (обновление)
- LEN (длина телеграммы): 132 (только для модуля передачи CTV_P)

Активные выходы

Выберите один или несколько выходов из имеющихся 8 каналов.

Обратите внимание на то, что для каждого выбранного вами канала ввода в SIMADYN D или SIMATIC TDC должна быть создана телеграмма получения данных.

Одна телеграмма содержит 32 значения real (для SIMADYN D тип данных NF) и 32 бинарных значения (1 DWORD или значение V4).

Для модуля получения данных нужно сконфигурировать следующие параметры:

- AR (имя канала): PDAMxDAT (x = 0 … 7 для каналов от 0 до 7)
- MOD (режим канала): R (обновление)
- LEN (длина телеграммы): 132 (только для модуля получения CRV_P)

После щелчка по кнопке <Принять> в дереве ресурсов под соответствующим интерфейсом создается модуль "FOBSD*nn*CH*xx*" (nn = индекс карты, xx = индекс канала) для каждого активного канала.

Техностроka

Функция "техностроka" пока не была добавлена в программу.

Настройки коммуникации

- Имя стойки:
  Для взаимодействия с SD/TDC компьютеру должно быть присвоено имя, состоящее из шести символов, например "IBA001".
- Имя соединения:
  ibaFOB-SD использует это имя для авторизации соединения с партнером по коммуникации: SIMADYN D или SIMATIC TDC. Это имя должно быть уникальным в пределах коммуникационного сегмента CS14 или CP53M0, т.е. в нем нельзя регистрировать другие модули Siemens или iba под тем же именем. По умолчанию присваивается имя "IBAL1A".
- Имя партнера:
  Здесь введите имя партнера по коммуникации, сконфигурированное в SIMADYN D или SIMATIC TDC.
  Для SIMADYN D это имя модуля CS14, например "D0500B", а для SIMATIC TDC (GDM) это имя модуля GDM, обычно D01_P1.
- Версия ПО:
  Здесь введите версию ПО для SIMADYN D или SIMATIC TDC, например для STRUC "V420" и для CFC "V610".

Примечание

Если в настройках есть ошибки, обмен данными невозможен.

9.4.6 Reflective Memory

Карты reflective memory обеспечивают доступ к сторонним системам на базе шины VME (например, GE FANUC, Converteam HPCi).

Под интерфейсом RFM располагаются следующие интерфейсные карты:

- VMIC PCI-5565PIORC: (карта reflective memory card, 64 или 128 Мб)
- VMIC PCI-5588 among others: (более ранние версии карт reflective memory)

Краткое описание

Поскольку эта карта является разработкой другого производителя, ее конфигурирование и пользовательская настройка отличаются от конфигурирования и настройки карт iba.

Поскольку карта RFM не имеет однородного участка данных, а содержит разные типы данных, параметры карт iba-FOB в данном случае неприменимы.

Расположение и структуру используемых данных необходимо определить во внешнем файле с параметрами. Этот файл затем загружается в конфигуратор ввода-вывода ibaLogic, в котором создаются сигналы с требуемыми типами данных.

Настройки карты

Если в дереве элементов выделен интерфейс RFM, то в правой части отображаются соответствующие настройки карты.

Рис. 110: Карты Reflective Memory

- Активирована

Эта опция используется для активации или деактивации карты.

Конфигурация

- Режим своппинга

В зависимости от подключенного устройства может понадобиться использовать своппинг, т.е. изменение порядка байтов в типе данных, чтобы ibaLogic могла считать и обработать данные.

Примечание

Режимы своппинга, которые описаны ниже, отличаются от режимов, используемых для SST-карты. В данном случае обозначения берутся из VMIC, чтобы соответствовать подключенным картам RM.

Пояснение касательно режимов своппинга (каждая буква соответствует одному байту, пробелы добавлены для наглядности).

Режим своппинга	Объяснение
Нет своппинга	"ABCD EF G H IJKL"
Byte-своппинг	"BACD FE H G JILK"
Word-своппинг	"CDAB GH E F KLIJ"
Word- и byte-своппинг	"DCBA HG F E LKJI"
Своппинг по типу данных	"DCBA FE G H LKJI"

Файл

Файл с параметрами, который содержит описание сигналов.

Последовательность действий

1. Создайте шаблон для этих данных, щелкнув кнопку <Шаблон> (<Template>).
2. Затем откройте файл в редакторе или MS Excel.
3. Для каждого сигнала укажите следующие данные, соблюдая указанный формат:
   "Имя сигнала, тип данных, адрес памяти, номер бита, направление, комментарии"

Формат	Объяснение
Имя сигнала	Должно соответствовать требованиям стандарта IEC.
Тип данных	Элементарный тип данных (см. раздел C.1) BOOLEAN BYTE, WORD, DWORD SINT, USINT, INT, UINT, DINT, UDINT REAL и LREAL
Адрес памяти	Смещение внутри RFM-памяти в десятичном или шестнадцатеричном формате. Переключаться между форматами можно с помощью строки, начинающейся с "#hexval" или "#intval".
Номер бита	Имеет значение только для типа данных BOOLEAN, иначе 0.
Направление	"INPUT" или "OUTPUT"
Комментарии	Любой текст

Пример

```
#hexval
Test_Signal,REAL,1000,0,INPUT, Testsignal input
Test_Signal2,REAL,2000,0,OUTPUT, Testsignal output
```

```
#intval
TestBit_0,BOOL,2048,0,OUTPUT, Bit 0
TestBit_1,BOOL,2048,1,OUTPUT, Bit 0
TestBit_2,BOOL,2048,2,OUTPUT, Bit 0
```

Результат

Шаблон создан.

Последовательность настройки параметров

Последовательность действий

1. Щелкните кнопку <Шаблон> и введите имя и путь сохранения файла, чтобы создать CSV-файл в качестве шаблона.
2. Откройте этот файл с помощью редактора и внесите в него сигналы.
3. Загрузите получившийся файл в конфигуратор ввода-вывода, нажав <Принять>.
4. Дождитесь завершения инициализации и передачи параметров RFM-карте.
5. Еще раз нажмите <Обновить апп. обесп.> (<Update Hardware>), чтобы заданные сигналы отобразились в дереве ресурсов.
6. Соотнесите созданные сигналы с именами виртуальных сигналов.

Важно

Не забудьте еще раз щелкнуть <Обновить апп. обесп.> после <Принять>, поскольку только после этого ibaLogic загрузит параметры в RFM-карту. Кнопка <Обновить апп. обесп.> позволяет передать параметры от карты в конфигуратор ввода-вывода.

9.5 Локальная периферия (PADU-S-IT)

Локальная периферия доступна только в том случае, если в качестве платформы выбрано устройство ibaPADU-S-IT.

- ibaPADU-S-IT это основной модуль семейства ibaPADU-S для "интеллектуального" и распределенного ввода / вывода. Модульная концепция представляет собой стойку для модулей с кросс-платой, на которую устанавливается основной модуль и макс. 4 дополнительных модуля ввода / вывода. В стойку могут быть установлены модули ввода-вывода для аналоговых и цифровых сигналов разного уровня, для сигналов тока и напряжения. Модули поддерживают частоту дискретизации до 1 кГц или 20 кГц.

Настройки

Если вы подключились к платформе и щелкнули кнопку <Обновить апп. обесп.>, то вы увидите следующие настройки для PADU-S-IT.

Настройки PADU-S-IT	Объяснение
Источник прерываний	PADU-S-IT
Настройки карты	PADU-S-IT
Ресурсы	PADU-S-IT TCPIP_OUT GLOBALVAR
Под PADU-S-IT	Установленные модули

Настройки карты

"Активирована" ("Enabled")

Для платформы PADU-S-IT эта опция выбрана всегда.

"Настройки модулей" ("Module settings")

Все модули ibaPADU-S-IT описаны в разделе "Настройки модулей".

Отдельные модули можно деактивировать.

Каждому модулю присваивается внутреннее имя модуля. Свойства модуля можно посмотреть в соответствующей строке таблицы:

Столбец 1: "HW" обозначает отдельные модули ввода и вывода.
"FOC" обозначает оптоволоконный интерфейс.

Столбец 2: "x/y" обозначает количество аналоговых / цифровых входов

Столбец 3: "x/y" обозначает количество аналоговых / цифровых выходов

Для типа модуля … FOC … вы можете уменьшить количество сигналов в соответствии с вашими требованиями. Щелкните кнопку (…) перед первым столбцом, чтобы указать нужное количество сигналов.

Документация по аппаратному обеспечению

Подробная информация касательно свойств ibaPADU-S содержится в руководстве по ibaPADU-S-IT.

9.6 Коммуникация по протоколу TCP/IP

Доступны следующие типа коммуникации по TCP/IP:

- Общее TCP/IP-соединение посредством модуля (дополнительная информация содержится в разделе 5.3.2, "TCPIP_SENDRECV")
- Передача данных по TCP/IP в ibaPDA

Коммуникация по TCP/IP (соответствующие параметры можно настроить в конфигураторе ввода-вывода) в настоящее время ограничена отправкой телеграмм в ibaPDA. В отличие от стандартного обмена данными по TCP/IP посредством модуля TCPIP_SENDRECV, поддержка модульной структуры в ibaPDA реализована с использованием специального протокола. На платформе WinXP можно передавать 16 телеграмм, и 8 телеграмм - на платформе PADU-S-IT. Каждая телеграмма может содержать 32 значения real и 32 цифровых значения, телеграммы могут отправляться одной или нескольким системам PDA.

Примечание

Обратите внимание на то, что коммуникация по TCP/IP осуществляется не в реальном времени. Это означает, что данные, передаваемые циклически, поступают не в реальном времени, следовательно, некоторые телеграммы могут быть потеряны.

Настройка параметров соединения TCP/IP

В левой части диалогового окна в дереве элементов выберите нужный канал.

Вы можете отдельно активировать каждый из 8 или 16 каналов и ввести следующие параметры:

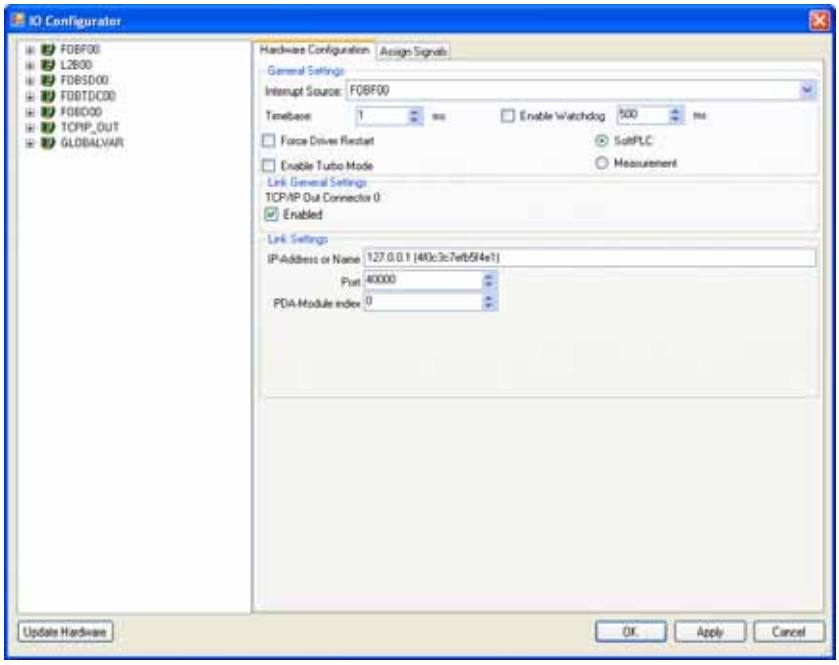


Рис. 111: Настройка параметров соединения TCP/IP

- IP-адрес или имя удаленной станции, т.е. компьютера, на котором работает сервер ibaPDA.
- Номер порта: должен соответствовать указанному в настройках ibaPDA. Значение по умолчанию: 40000.

- Модулю PDA присваивается уникальный номер от 0 до 31. Этот номер должен совпадать с индексом модуля подключенной системы PDA.

9.7 Коммуникация по протоколу OPC

Международный стандарт OPC (OLE for Process Control) широко используется для соединения с HMI-системами (Human Machine Interface, control & monitoring), а также для измерения медленных сигналов.

9.7.1 OPC-сервер

OPC-сервер предоставляет подключенным к нему OPC-клиентам все переменные, которые были определены как "видимые для OPC". OPC-сервер, как правило, работает на той же машине, что и сервер ibaLogic и соединяется с PMAC посредством TCP/IP.

Примечание

Допустимое количество OPC-переменных зависит от приобретенной лицензии (в аппаратном ключе).

Примечание

OPC-сервер может также работать на отдельном компьютере. Однако, для этого нужно направить отдельный запрос iba.

OPC-клиент ищет OPC-сервер ibaLogic под именем **"iba.Logic4OPC.1"**. После того как соединение установлено, вы можете выбрать переменные OPC путем выбора имен соответствующих переменных в раскрывающемся списке.

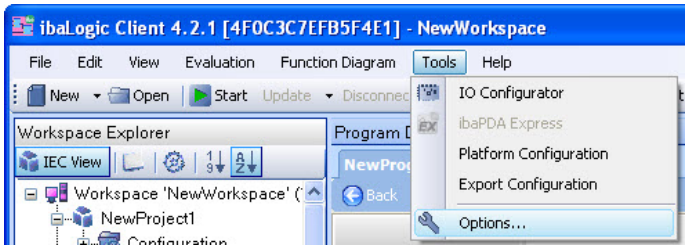
OPC-сервер работает в соответствии со спецификацией DA (Data Access) V2.05a.

Примечание

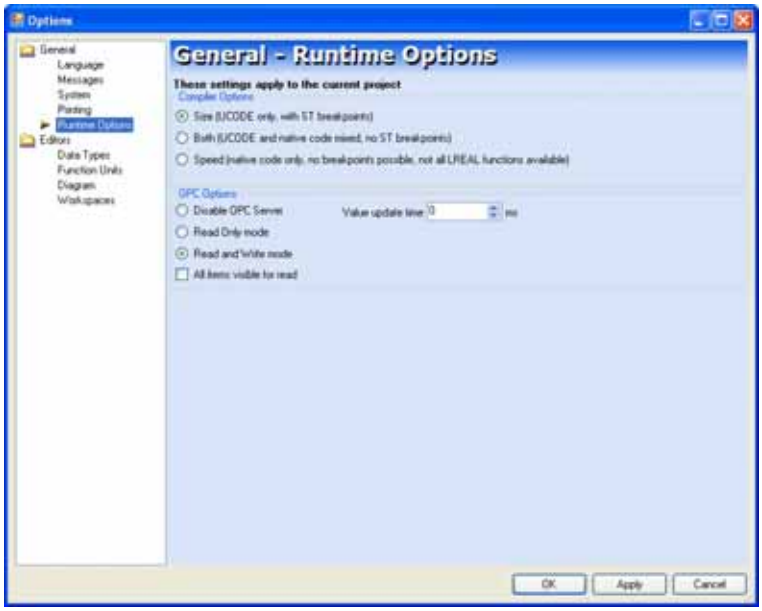
Для того чтобы установить соединение между OPC-клиентом и OPC-сервером, нужно настроить ряд параметров DCOM и соблюсти требования безопасности.

Более подробно эта процедура описана в отдельном документе, который вы можете получить, обратившись в компанию iba.

1. Чтобы настроить некоторые параметры OPC-сервера, выберите меню "Инструменты - Опции..." ("Extras -- Options…").
Появится диалоговое окно "Опции".



2. В дереве элементов слева выберите "Опции режима исполнения" ("Runtime Options").



3. Выберите нужную опцию.

Опция	Объяснение
Деактивировать OPC-сервер	OPC-сервер будет полностью отключен.
Режим "только чтение"	Могут считываться даже те переменные, которые были определены как "активные для OPC-записи".
Режим "чтение и запись"	По умолчанию: доступ в соответствии с параметрами переменных.
Все элементы доступны для считывания	Даже те переменные, которые не "отображаются для OPC" могут быть считаны OPC-клиентами, однако не могут быть записаны.

9.7.2 Настройка параметров переменных OPC

Для каждого OPC-сигнала в ibaLogic нужно создать коннектор вне задачи (OTC). В диалоговом окне редактирования коннекторов нужно активировать поля выбора для OPC.

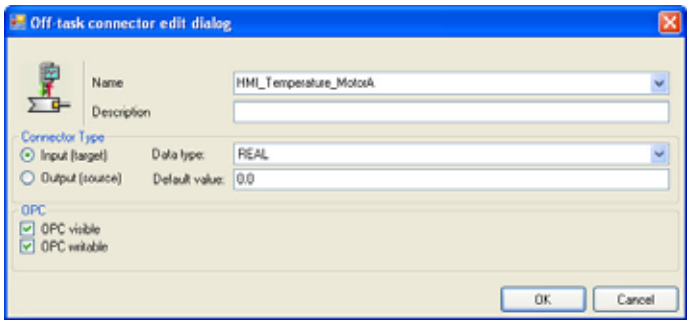


Рис. 112: Редактирование коннекторов вне задачи

Типы данных, доступные для OPC-соединения, зависят от типа данных OPC-клиента. Как правило, можно использовать элементарные типы данных и массивы.

Самый короткий интервал обновления составляет 50 мс (на стороне ibaLogic), однако он в значительной степени зависит от объема данных.

Переменные OPC выделяются разными цветами, благодаря чему их легко идентифицировать. См. описание настройки параметров в разделе 6.8.3 "Коннекторы вне задачи".

OPC-сервер можно найти в раскрывающемся списке под именем "iba.Logic4OPC.1".

10 Управление базой данных

ibaLogic - это приложение, которое работает на основе базы данных. Для сохранения промежуточных данных и результатов, нужно регулярно выполнять резервное копирование базы данных. Для этих целей в ibaLogic V4 имеются опции автоматического создания резервной копии, а также выполнения этого вручную.

10.1 Создание резервной копии базы данных

Прежде чем вносить серьезные изменения в базу данных имеет смысл создать ее резервную копию.

10.1.1 Создание резервной копии базы данных вручную

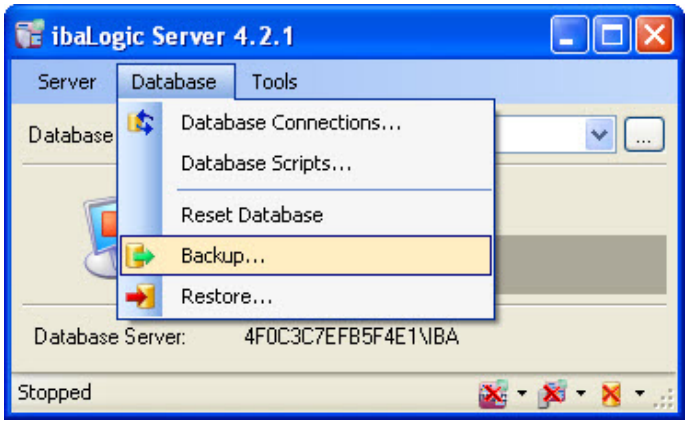
При создании резервной копии всегда сохраняется вся информация, содержащаяся в базе данных на момент сохранения. База данных может содержать несколько рабочих областей. Выбрав пункт меню "Открыть рабочую область" ("Open Workspace"), вы увидите рабочие области, которые существуют на данный момент в базе данных клиента.

Необходимые условия

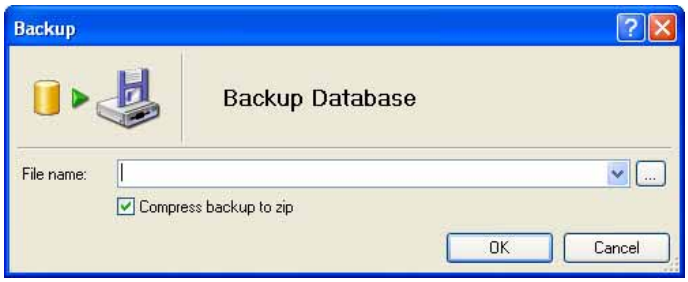
- ❑ Диалоговое окно сервера ibaLogic открыто.
- ❑ Соединение с базой данных установлено.

Последовательность действий

1. В меню выберите пункт "Резервирование базы данных…" ("Backup Database…").



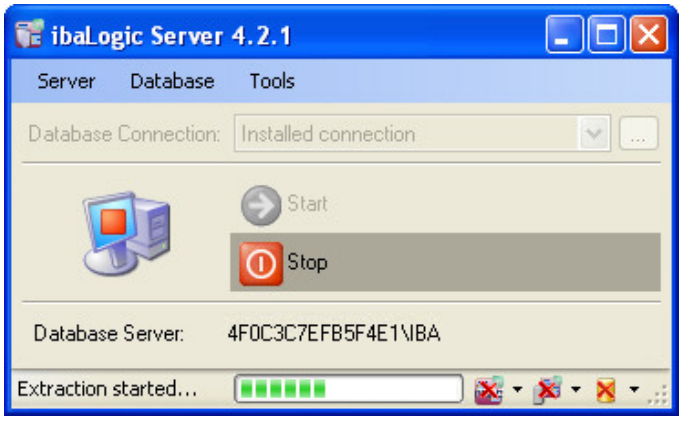
Откроется следующее диалоговое окно:



2. Выберите имя файла и папку, в которую будет сохранена резервная копия. Активируйте опцию "Архивировать файл в zip" ("Compress backup to zip"),

если нужно также сохранить конфигурацию аппаратного обеспечения из диспетчера ввода-вывода и все имеющиеся DLL.

3. Щелкните <OK>.



Примечание

Настоятельно рекомендуется создавать резервную копию базы данных перед обновлением версии ibaLogic.

При обновлении версии ibaLogic изменения также вносятся в базу данных при помощи скриптов базы данных. После этого возврат к предыдущей версии становится невозможен.

10.1.2 Автоматическое создание резервной копии базы данных

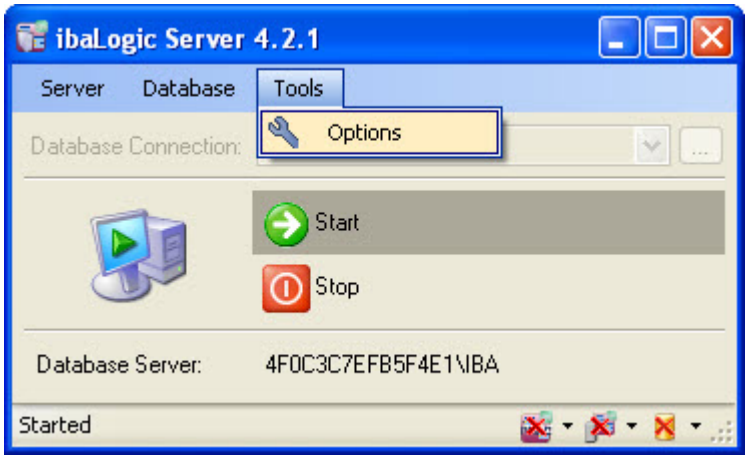
Для выполнения автоматического резервирования требуется настроить некоторые настройки в ibaLogic.

Необходимые условия

- ❑ Диалоговое окно сервера ibaLogic открыто.
- ❑ Соединение с базой данных установлено.

Последовательность действий

1. Выберите пункт меню "Инструменты - Опции" ("Extras -- Options").



2. В дереве элементов выберите "Сервер -- Авторезервирование" ("Server -- Autobackup").

3. Поставьте галочку напротив опции "Активировать авторезервирование" ("Enable Autobackup").

4. Установите длительность интервала между резервированием.

Поскольку изменения в проекты ibaLogic можно вносить только через клиенты ibaLogic, то отсчет времени интервала запускается только после соединения клиента с сервером. Если в базе данных обнаружены изменения, то по истечении времени интервала создается новая резервная копия в виде файла формата *.bak или *.zip.

Важно

Определите отдельные папки для автоматического и ручного резервного копирования соответственно. Поскольку стратегия очистки предполагает удаление данных только из папки для автоматического резервирования, файлы, содержащиеся в папке для ручного резервирования, не будут удалены.

Стратегия очистки определяется в следующих полях:

- "Время до очистки" (смещение) ("Time until cleanup" (Cleanup offset))
- "Количество резервных копий" ("Number of backup copies")

Опция	Объяснение
Интервал между резервированием	Резервная копия создается по истечении указанного промежутка времени.
Время до очистки (смещение)	При использовании этой настройки резервные копии создаются до того момента, когда истечет минимальный период резервного копирования и смещения очистки.
Количество резервных копий	Этот параметр определяет минимальное количество резервных копий, которые присутствуют постоянно. При этом принимается во внимание период времени до очистки (смещение).
Папка для сохранения резервных копий	Имена файлов присваиваются программой ibaLogic: *"Autobackup_ibaLogic4_<Date, Time>.bak"* или *"…..zip"*. Дата и время указываются в формате ГГГГММДДЧЧММ.
Архивировать файл с резервной копией в формат ZIP	Файл резервной копии сохраняется в формате ZIP.

Пример

Если параметры резервного копирования в вашей программе совпадают с выставленными на скриншоте выше, то все резервные копии, созданные неделю назад и ранее, будут удаляться. Однако, мин. 10 файлов будут сохранены вне зависимости от даты создания.

10.2 Восстановление базы данных

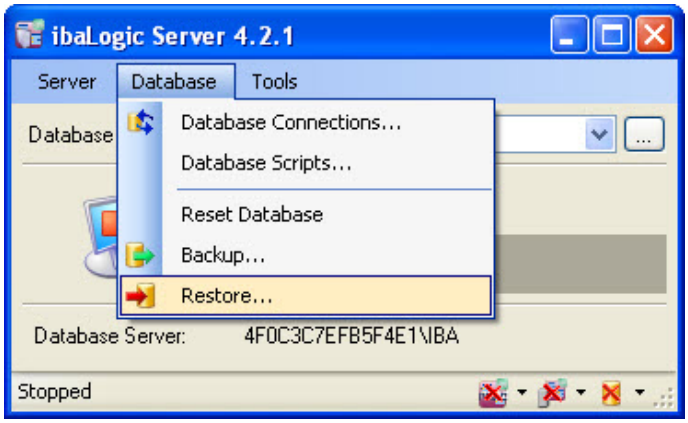
При восстановлении базы выполняется загрузка ранее сохраненной версии (со всеми рабочими областями) для редактирования.

Необходимые условия

- ❑ Диалоговое окно сервера ibaLogic открыто.
- ❑ Соединение с базой данных установлено.
- ❑ Сервер ibaLogic остановлен.

Последовательность действий

1. В меню выберите пункт "Восстановление базы данных…" ("Restore Database…").



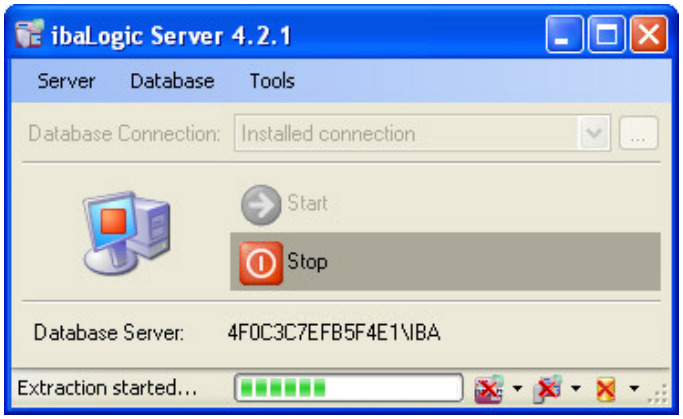
Появится диалоговое окно "Восстановление".

2. Щелкните кнопку <…> и в открывшейся папке выберите резервную копию базы данных.

   По умолчанию ibaLogic указывает папку, которая использовалась в предыдущий раз, или следующую папку:

3. Щелкните <OK>, чтобы восстановить базу данных из резервной копии.

   Ход выполнения процесса восстановления отображается на экране. После этого сервер будет остановлен.



4. При появлении каких-либо сообщений, требуется просто их подтвердить.
5. Снова запустите сервер, чтобы продолжить программировать в клиенте.

10.3 Сброс базы данных

Эта функция используется для того, чтобы вернуть базу данных в ее исходное состояние (очистить).

Важно

При сбросе базы данных из нее удаляются все данные.

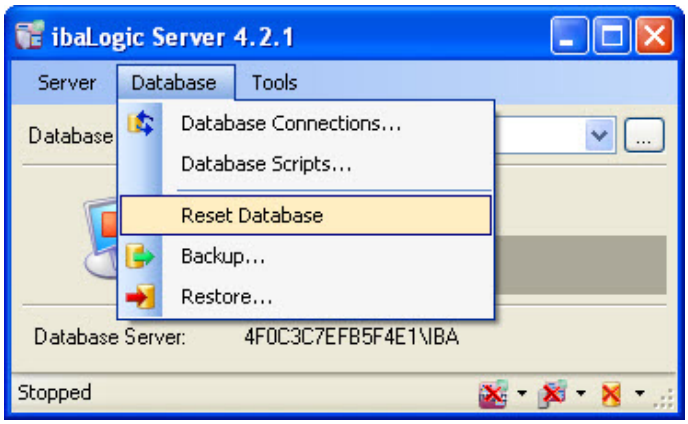
Следовательно, имеет смысл предварительно создать ее резервную копию.

Необходимые условия

- ❑ Диалоговое окно сервера ibaLogic открыто.
- ❑ Соединение с базой данных установлено.
- ❑ Сервер ibaLogic остановлен.

Последовательность действий

1. Выберите пункт меню "Базы данных - Сброс базы данных" ("Database – Reset Database…").



Результат

Сброс базы данных выполнен.

11 Анализ программы, отладка и время отклика

Существуют различные способы и инструменты для анализа работы и отладки программы.

При этом необходимо учитывать различия между программой, которая еще тестируется, и той, которая уже активно используется.

Описание	Режим тестирования	Активное использование
Отладка блоков структурированного текста с использованием точек останова	Да	Нет (Программа остановлена.)
Блоки слежения или файлы Log DLL, созданные пользователем (например, LogFile_String_WriteDll.dll, ……)	Да	При определенных условиях (время отклика)
Анализвремени отклика, формы кривой и т.д. с помощью **ibaPDA Express**	Да	Да
Запись DAT-файлов для ibaAnalyzer с помощью **блока DAT_FILE_WRITE**	Да	Да

11.1 ibaPDA Express

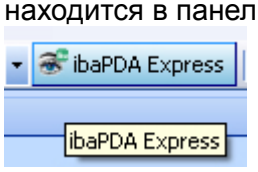
ibaPDAExpress используется для быстрой проверки формы сигнала.

Необходимое условие

Эта функция доступна только при работе программы в режиме онлайн.

Последовательность действий

➲ Запустите ibaPDA Express щелчком по кнопке <ibaPDA Express>, которая находится в панели инструментов.



Результат

ibaPDA Express открывается в отдельном окне в рамках приложения ibaLogic.

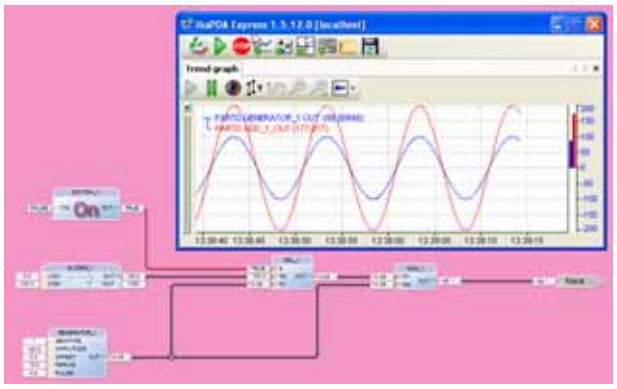


Рис. 113: ibaPDA Express с несколькими сигналами

11.1.1 Управление отображением сигналов

Для управления отображением сигналов доступна следующая панель инструментов.

Рис. 114: ibaPDA Express: панель инструментов

Нижеследующая таблица содержит пояснения касательно функций значков в панели инструментов.

Значок	Имя	Клавиша	Объяснение
	Начать прокрутку	<F6> (Переключение)	Запускает непрерывное отображение с текущего значения времени. Кнопка активна, если нажата кнопка "Остановить прокрутку".
	Остановить прокрутку	<F6> (Переключение)	Останавливает непрерывное отображение. После нажатия этой кнопки на графике появляется линейка, которую можно перемещать курсором мыши и измерять с ее помощью кривые. Значения сигналов приводятся в комментариях к графику. Ось X можно передвигать с помощью курсора мыши. Таким образом можно просматривать значения за предыдущие периоды времени. Кнопка активна, если запущено отображение сигналов.
	Автоматическое присвоение цветов сигналам.		Все кривые в области отображения сигналов выделяются разными цветами в соответствии со стандартной схемой цветов для графика.

	Полное автомасштабирование	<F5>	Выполняется автоматическое масштабирование всех кривых области отображения для каждого графика и оси Y.
	Восстановить ручное масштабирование		Выполняется восстановление (после применения автомасштабирования или приближения/отдаления) введенных вручную параметров масштабирования. Кнопка активна, если были определены параметры ручного масштабирования.
	Отдалить на один шаг	<F3>	Кнопка активна, если масштаб отображения был изменен. Возвращает отображение к предыдущему коэффициенту масштабирования (уменьшает)
	Отдалить все	<F4>	Кнопка активна, если масштаб отображения был изменен. Возврат к исходному (автоматически определенному) масштабу отображения.
	Направление прокрутки		Вы можете выбрать нужное направление прокрутки в раскрывающемся меню.

11.1.2 Выбор сигналов

Нажав и удерживая клавишу <Alt>, вы можете перетащить сигналы из коннектора в окно ibaPDA Express.

Возможны также следующие варианты:

- Отображение сигнала в виде отдельной полосы.
  Для этого перетащите сигнал на ось X, чтобы создать новую полосу отображения сигнала.
- Размещение сигнала на существующей полосе.
  Для этого перетащите сигнал на полосу, чтобы создать новую ось Y.
- Размещение сигнала на существующей оси Y (на одной шкале с другим сигналом).
  Для этого перетащите сигнал на уже существующую ось Y.

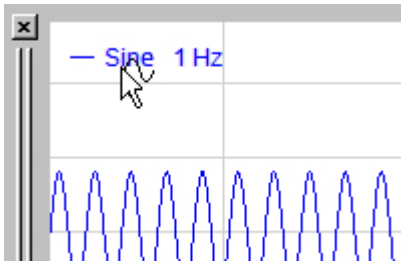
Новым сигналам в пределах одного окна автоматически присваиваются новые цвета. Имена этих сигналов обозначаются соответствующими цветами и размещаются в верхней левой части области отображения полос сигналов. Сигналы, располагающиеся на одной оси, разделяются тире.

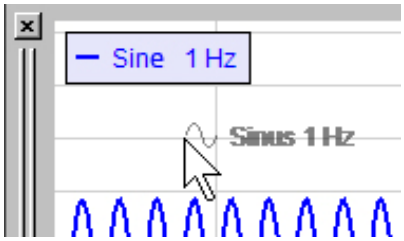
11.1.3 Перемещение сигналов

Сигналы можно перемещать между графиками, а также выносить за пределы окна. Это означает, что сигнал можно перетаскивать из одного графика на другой, который уже содержит сигнал. Различать сигналы помогает автоматическое присвоение цветов.

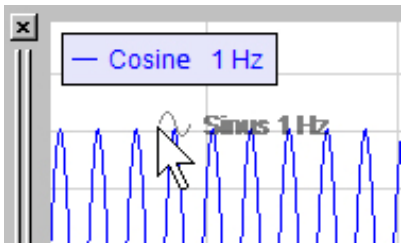
Последовательность действий

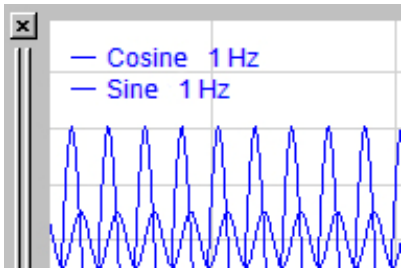
2. Установите курсор мыши на имя сигнала (в комментариях к графику), который вы хотите переместить. Рядом с курсором появится волнистая линия.





3. Удерживая кнопку мыши нажатой, перетащите сигнал на другой график и разместите в любом свободном пространстве.





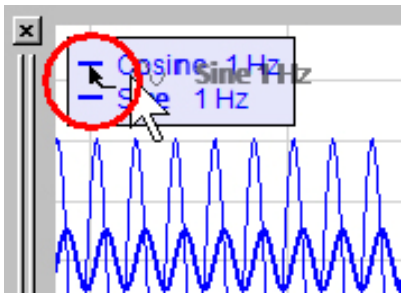
Результат

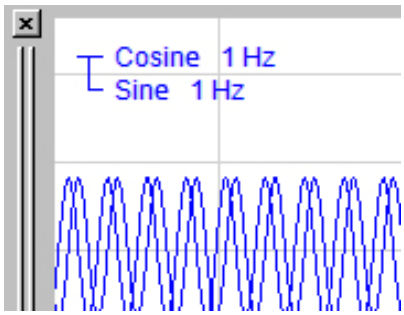
Вы создали 2 сигнала с разными осями Y.

Примечания

Не останавливайтесь на втором этапе перемещения сигнала, а перетащите новый сигнал на уже существующий, чтобы появилась маленькая черная стрелка. В этом случае сигналам будет присвоена одна ось Y.

В случае бинарных сигналов вы можете также задать их последовательность. Такие сигналы располагаются один над другим. В зависимости от того, где располагается маленькая черная стрелка: вверху или, как на скриншоте, внизу бинарного сигнала, он будет размещен над ней или под ней.





Результат

Вы создали 2 сигнала с общей осью Y.

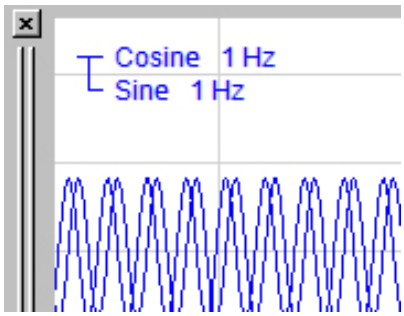
11.1.4 Цветовая маркировка сигналов

Вы можете выделить сигналы с помощью цветов. Существует два способа это сделать:

- ❑ Автоматически
- ❑ Ручная настройка

Последовательность действий

➲ Для автоматического присвоения цветов сигналам щелкните кнопку <Автоматическое присвоение цветов сигналам> (<Assign signal colors automatically>).



11.1.5 Удаление сигналов из области отображения

Последовательность действий

1. Установите курсор мыши на имя сигнала (в комментариях к графику), который вы хотите удалить.
2. Щелкните по нему правой кнопкой мыши. Откроется контекстное меню.
3. Выберите "Удалить сигнал" ("Remove Signal").

Примечание

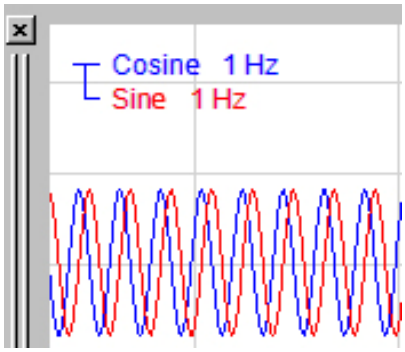
При удалении оси Y удаляются все сигналы, присвоенные этой оси.

11.1.6 Удаление графиков из области отображения

Существует несколько способов удалить график.

Последовательность действий

1. Щелкните по маленькому крестику в верхнем левом углу графика.



или

1. Щелкните правой кнопкой мыши по свободной области на графике. Откроется контекстное меню.
2. Выберите "Удалить график" ("Remove Graph").

11.1.7 Масштаб осей

11.1.7.1 Автоматическое масштабирование

Для отображения полного диапазона амплитуд сигнала на одном графике рекомендуется использовать функцию автомасштабирования. Масштаб всех сигналов всех осей Y на графике устанавливается в соответствии с самой высокой амплитудой.

Последовательность действий

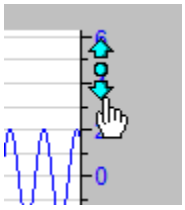
1. Щелкните правой кнопкой мыши по свободной области на соответствующем графике. Откроется контекстное меню.
2. Выберите пункт "Автомасштабирование" ("Auto scaling").
3. Если вы хотите задать автоматический масштаб для всех графиков области отображения сигналов, щелкните клавишу <F5> или выберите кнопку <Автомасштабировать все> (<Auto scale all>).

11.1.7.2 Масштабирование с помощью мыши

Вы можете изменить масштаб сигналов по оси Y с помощью курсора мыши.

Последовательность действий

1. Наведите курсор мыши на верхнюю часть оси графика, чтобы появился элемент управления масштабированием (синяя стрелка).
2. Нажмите и удерживайте кнопку мыши на стрелке, которая указывает вверх: шкала будет увеличена.
3. Нажмите и удерживайте кнопку мыши на стрелке, которая указывает вниз: шкала будет уменьшена.
4. Нажмите и удерживайте кнопку мыши на точке между стрелками: будет выполнено автомасштабирование.



Совет

Если вы используете мышь с колесом прокрутки, то достаточно просто установить курсор мыши на шкале, чтобы изменять ее масштаб с помощью колеса прокрутки. Это касается не только оси X, но и Y.

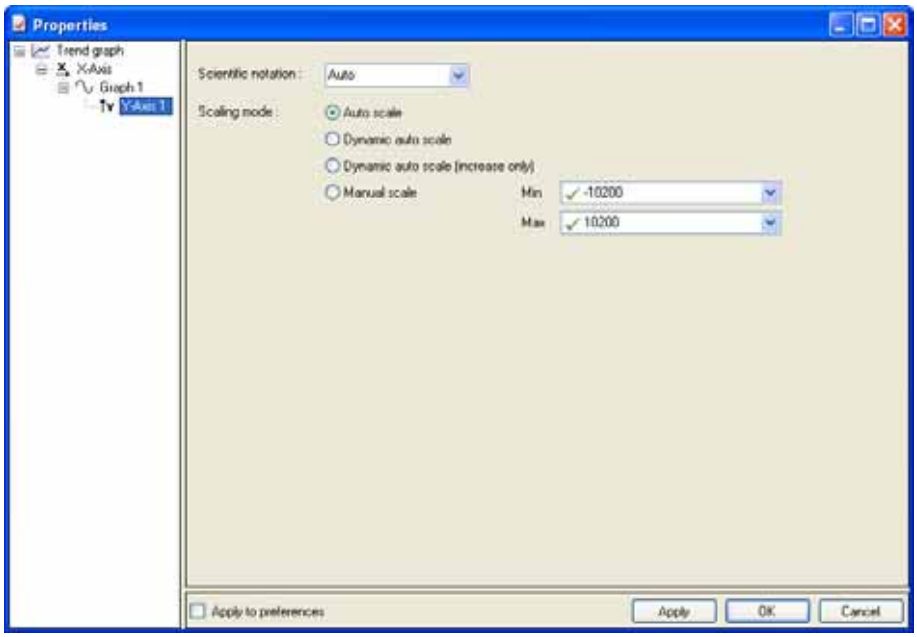
11.1.7.3 Масштабирование с использованием параметров отображения

Последовательность действий

1. Щелкните правой кнопкой мыши по свободной области на соответствующем графике.
2. Выберите пункт меню "Свойства".
   Появится диалоговое окно настройки свойств.

Вы можете настроить ручное масштабирование, используя опцию "Ось Y" ("Y-axis"). Если график содержит несколько осей Y, то для каждой из них в диалоговом окне отводится отдельная вкладка.

3. Примените настройки (щелкните <Применить>).



Результат

Все сигналы, которые относятся к одной оси Y, масштабируются в соответствии с параметрами, сконфигурированными для данной оси.

Комментарий

Если выбрана опция "Применить к параметрам" ("Apply to preferences"), то сконфигурированные настройки будут применяться по умолчанию.

11.1.8 Перемещение шкалы

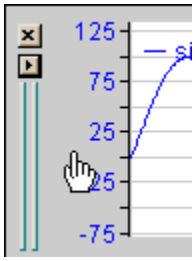
Вы можете перемещать ось X, если отображение сигналов находится в режиме паузы.

Необходимое условие

Ось Y: опция автомасштабирования не выбрана.

Последовательность действий

1. Наведите курсор на ось Y, чтобы появился значок перемещения (см. скриншот ниже).
2. Нажмите и удерживайте левую кнопку мыши, чтобы двигать шкалу вверх/вниз или влево/вправо.



11.1.9 Функция приближения/отдаления

Функция приближения/отдаления действует как на ось X, так и Y.

При приближении/отдалении графика, все остальные графики этой области отображения изменяются соответствующим образом. Область отображения поддерживает только одно опорное время для всех графиков, которые она содержит.

Если отображение активно, то функция приближения расширяет опорное время и таким образом увеличивает область отображения. Кривые сигналов начинают двигаться быстрее, поскольку геометрическая длина оси X делится на меньшее количество единиц времени.

Общее управление приближением/отдалением

1. Нажав и удерживая клавишу <Shift>, используйте мышь для приближения/отдаления. При этом будет изменяться масштаб только оси X.
2. Щелкнув кнопку  или нажав клавишу <F4>, вы восстановите исходный вид области отображения.

Приближение (увеличение)

Вы можете приближать любой участок полосы сигнала. После приближения пользователь может менять масштаб оси Y, не меняя при этом отображение приближенного участка оси X.

Автомасштабирование по оси Y действует на значения в приближенной/отдаленной (= видимой) области.

Необходимое условие

Область отображения сигналов была отдалена.

Последовательность действий

1. Нажав и удерживая левую кнопку мыши очертите прямоугольник вокруг области, которую вы хотите приблизить.
2. Отпустите кнопку мыши.

Отдаление (уменьшение)

Необходимое условие

Область отображения сигналов была приближена.

Последовательность действий

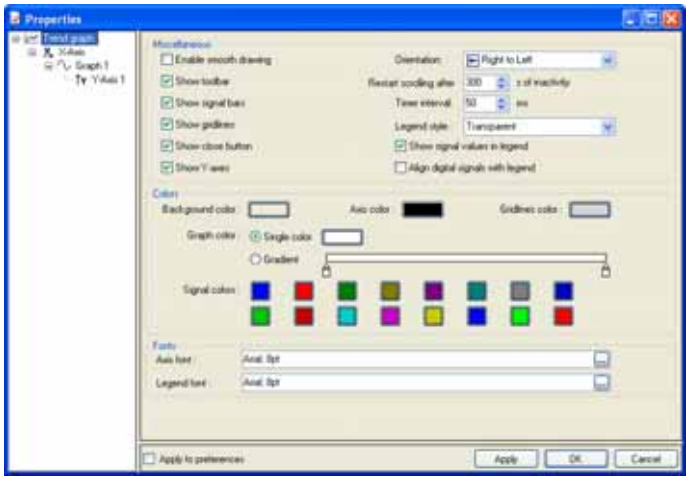
1. Для пошагового отдаления щелкайте кнопку <Отдалить на один шаг> (<Zoom out one step>) или нажимайте клавишу <F3>.
   Таким образом будет выполняться пошаговая отмена предыдущего приближения.
2. Щелкнув кнопку <Отменить приближение> (<Zoom out all steps>) или нажав клавишу <F4>, вы восстановите исходный вид области отображения.

11.1.10 Свойства графика тренда

В диалоговом окне свойств графика тренда вы можете настроить общие параметры для отображения графиков.

Последовательность действий

1. Щелкните правой кнопкой мыши по полосе сигнала.
2. В появившемся меню выберите пункт "Свойства...".
3. В дереве элементов в левой части диалогового окна выберите "График тренда" ("Trend Graph").



Разное

Опция	Объяснение
Активировать сглаженное отображение	Линии графиков сглаживаются при отображении.
Отобразить панель инструментов	Отображается панель инструментов.
Отображать значения сигналов в комментариях к графикам.	Значения сигналов приводятся в комментариях к графикам.
Отображать столбики	Эта опция позволяет отображать столбик, связанный с графиком сигнала.
Прозрачный фон комментариев	Фон комментариев к графикам становится прозрачным.
Направление прокрутки	Устанавливает направление прокрутки.
Перезапуск прокрутки	Определенный период отсутствия активности, по истечении которого выполняется перезапуск прокрутки.
Интервал обновления	Период времени по истечении которого происходит обновление области отображения.
Выравнивание цифровых сигналов и комментариев	Выполняется выравнивание цифровых сигналов и комментариев относительно друг друга.

Цвета

В этом диалоговом окне пользователь может изменить цветовую схему, которая используется для отображения графиков тренда и кривых.

➲ Для изменения цветового обозначения щелкните кнопку с соответствующим цветом. Выберите нужный цвет в цветовой палитре.

❑ Фон, оси, линии сетки:

➲ Для изменения цветового обозначения щелкните кнопку с соответствующим цветом. Выберите нужный цвет в цветовой палитре.

❑ График:
Фон полосы сигнала: однородный или с переходом цвета.

➲ Щелкните двойным щелчком по маленькой ячейке в конце полосы цветов и выберите нужный цвет в палитре.
Двойным щелчком мыши по цветовой схеме пользователь может добавлять вкладки цветов и окрашивать их, а также перемещать. Чтобы удалить вкладку цвета, выделите ее щелчком кнопкой мыши и нажмите клавишу <DEL>.

❑ Сигналы
Эти цвета пера можно использовать для маркировки 16 кривых, которые используются для отображения тренда. Как правило, программа автоматически маркирует кривые, исходя из имеющихся 16 цветов. Цвета пера также содержатся в области определения сигнала, см. последовательность ниже (направление последовательности характеристик: сверху вниз).

Шрифты

Шрифты назначаются для буквенного обозначения осей и создания комментариев к графикам (имена сигналов). Пользователь может изменить шрифт с помощью кнопки просмотра <…>, которая находится в конце линии.

Сигналы

При вызове этого диалогового окна в графике сигналов или существующем графике тренда с отображением сигналов появляется список сигналов с параметрами, включая цветовые обозначения.

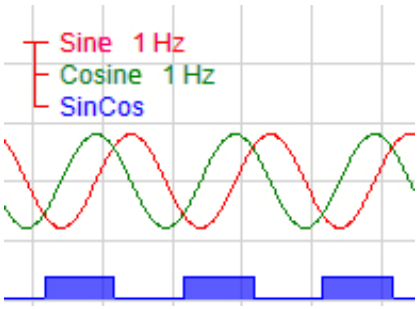


Рис. 115: График тренда

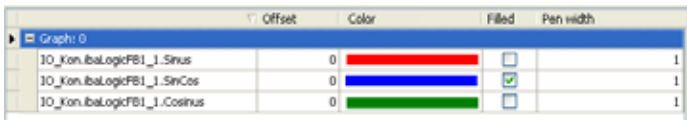


Рис. 116: Графические параметры сигналов

Вы можете выбрать нужный вам цвет сигнала в раскрывающемся списке в столбце "Цвет".

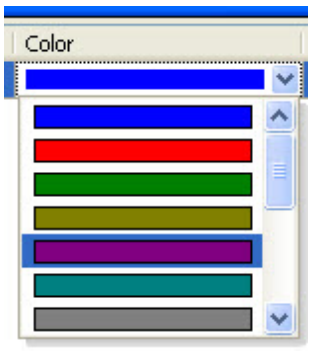


Рис. 117: Список цветов для выбора

Ось X

Интервал времени

Вместо использования автоматического масштабирования вы можете указать интервал времени в секундах, который будет использоваться для отображения сигналов. Эта функция позволяет пользователю контролировать скорость и продвижение сигнала по оси X.

Рис. 118: Ось X: свойства

Фиксированное положение оси

Как правило, ось времени двигается вместе с сигналом, поэтому новые значения отображаются на границе графика в области отображения. Если используется опция фиксации положения оси, то ось времени фиксируется от текущего момента времени на заданный период (интервал времени). Соответственно, значения сигналов записываются в пустой график. После заполнения графика отображается следующий интервал времени (пустой), в который продолжают записываться значения.

Ось Y

Если вы создали несколько осей Y на одном графике, то в диалоговом окне настроек будет содержаться несколько вкладок с номерами осей Y ("Y-axis #"). Следовательно, вы можете сконфигурировать отдельно каждую ось Y.

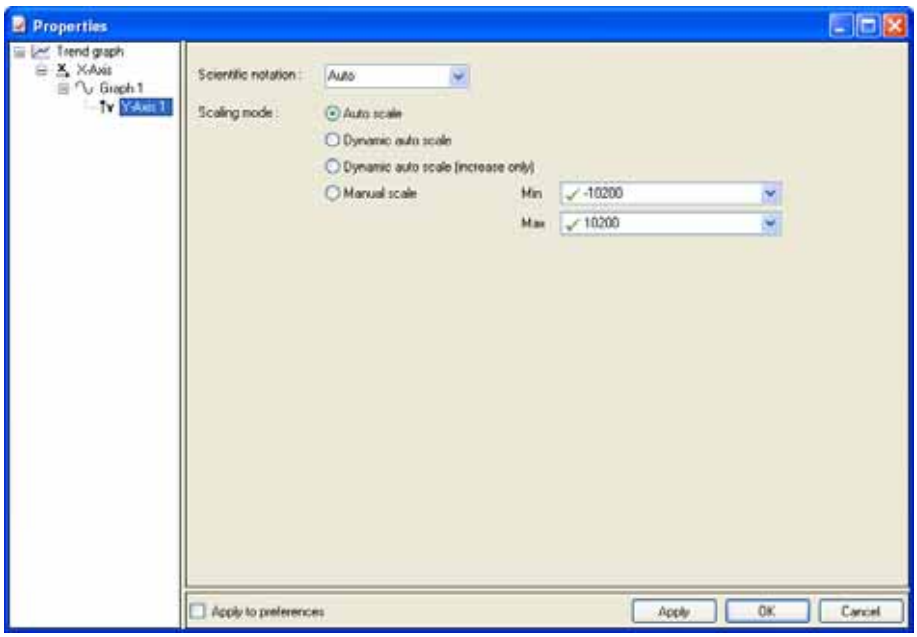


Рис. 119: ibaPDA Express: настройки отображения

Научная нотация

- "Автоматически" ("Automatically")
- "Всегда" ("Always")
- "Никогда" ("Never")

Опция	Объяснение
Автоматически	В зависимости от размера значений (количество знаков до и после запятой), шкала либо размечается в научной нотации (степень 10), либо нет.
Всегда	Использовать значения со степенью 10
Никогда	Всегда использовать значения со знаком до и после запятой

Режим масштабирования

- "Автомасштабирование" ("Auto scaling")
- "Динамическое автомасштабирование" ("Dynamic auto scaling")
- "Динамическое автомасштабирование" (только для увеличения) ("Dynamic auto scaling" (only for enlarging))
- "Ручное масштабирование" ("Manual scaling")

Опция	Объяснение
Автомасштабирование	По умолчанию; при отображении одного или нескольких графиков масштаб оси Y устанавливается в соответствии с наименьшим и наибольшим из всех полученных значений (если речь идет о сигнале).
Динамическое автомасштабирование	Если эта опция активна, то масштаб постоянно регулируется в соответствии с самой высокой амплитудой сигнала. Если амплитуда выходит за пределы полосы сигнала, масштаб уменьшается.
Динамическое автомасштабирование (только для увеличения)	Если выбрана эта опция, то масштаб постоянно регулируется в соответствии с самой высокой амплитудой сигнала. Если амплитуда выходит за пределы полосы сигнала, масштаб не меняется.
Ручное масштабирование	При выборе этой опции пользователь может указать начальное (мин.) и конечное (макс.) значения шкалы. (Отображается, только если диалоговое

	окно открыто посредством контекстного меню полосы сигнала, а не в предварительных настройках.)

11.1.11 Дополнительные функции

Дополнительные функции можно активировать с помощью контекстного меню, которое вызывается щелчком по значку в строке заголовка ibaPDA Express.

Доступны следующие опции:

- ❑ "Панель инструментов" ("Toolbar")
- ❑ "Дерево сигналов" ("Signal tree")
- ❑ "Полноэкранный режим" ("Full screen view")

При выборе меню область отображения ibaPDA Express увеличивается.

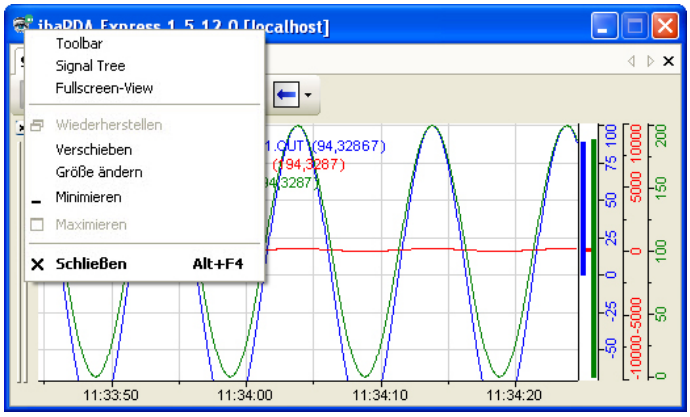


Рис. 120: Контекстное меню "ibaPDA Express"

- ❑ Панель инструментов

Панель инструментов содержит следующие элементы:

Значок	Имя	Объяснение
	Изменить предустановленные значения	Открывается меню свойств.
	Реальное время	Запуск / остановка всех полос.
	Добавить график тренда	Будет добавлен еще один график тренда Графики тренда можно располагать по собственному усмотрению. Для этого нужно щелкнуть левой кнопкой мыши по "Графику тренда" и переместить его в нужно место, удерживая кнопку мыши. При этом появляются значки стыковки.
	Добавить QPanel Добавить элемент scope view Добавить элемент digital meter	В Express будут добавлены дополнительные функции: QPanel, а также элементы scope view и цифровой индикатор (digital meter).
	Загрузить вид	Будет открыта конфигурация ibaPDA Express (файл XML).
	Сохранить вид	Сохранение файла с конфигурацией ibaPDA Express (файл XML).

- Дерево сигналов

Отображается дерево элементов, включающее все переменные, которые содержатся в программе. Эти переменные можно добавить в график тренда двойным щелчком или перетаскиванием.

- Полноэкранный режим

ibaPDA Express отображается в полноэкранном режиме. Вы можете выйти из этого режима, нажав клавишу <F10>.

Дополнительная документация

Более подробная информация касательно вышеописанных функций содержится в соответствующей дополнительной документации по системе ibaPDA.

11.2 Время отклика

ibaLogic обеспечивает детерминированное время отклика (отклик в реальном времени).

Задачи в ibaLogic имеют базовый интервал времени, который составляет мин. 1 мс и зависит от времени прерываний, которое можно настроить в меню "Инструменты – Конфигуратор ввода-вывода" ("Extras -- I/O configurator"). Это минимальный интервал времени для выполнения задачи.

Более быстрое выполнение задач невозможно. Некоторые модули iba могут выполнять дискретизацию данных с циклом 50 мкс и передавать эти данные в виде массивов (пакетов) в ibaLogic. Затем ibaLogic обрабатывает эти значения на базе вспомогательного тактового цикла. Более подробная информация содержится в пункте 9.4.2, ""Режим буферизации" FOB-карт".

В целях управления задачами ibaLogic, используя базовый тактовый цикл, проверяет, какие задачи ожидают обработки, и добавляет их во внутренний список задач. Этот список циклически анализируется программой в направлении сверху вниз, при этом уже выполненные задачи удаляются из списка.

Необходимо также учитывать, что сконфигурированный базовый тактовый цикл также используется для считывания входов и записи выходов.

IbaLogic известна только интервальная задача.

Интервальная задача запускается в соответствии со сконфигурированным промежутком времени. Связанная с ней программа имеет собственное время вычисления.

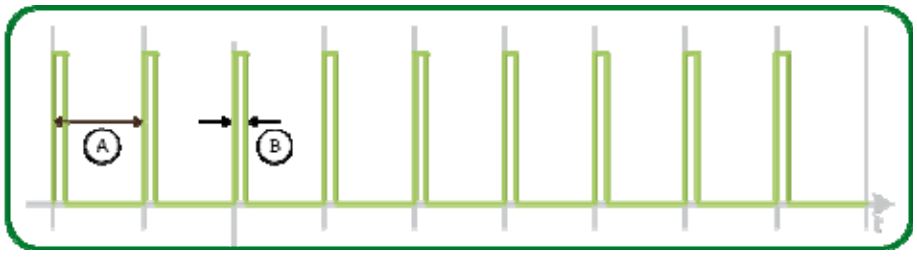


Рис. 121: Интервальная задача

A Интервал

B Время вычисления

Время вычисления

Время вычисления различных программ во всем проекте пользователя также имеет значение при рассмотрении времени отклика.

В ibaLogic время вычисления различных интервальных задач отображается в процентах и как индикатор выполнения, что позволяет пользователю проверить загрузку системы.

В этом примере (учитывая, что интервальная задача составляет 1 мс) значение в процентах означает, что требуется 34,36 % от 1 мс. Иными словами, это процентное значение от интервала времени, сконфигурированного для задачи. 34,36 % от интервала времени задачи в 1 мс соответствует 0,3436 мс от времени процессора, которое требует программа.

Режим Turbo (Turbo mode):

Для того чтобы избежать временной блокировки программы ibaLogic операционной системой Windows, в многоядерной системе вы можете назначить одно ядро исключительно для обработки задач ibaLogic.

Примечание

Для того чтобы обеспечить детерминированное выполнение программы и хорошую производительность, iba рекомендует:

- Для задач с временем выполнения < 20 мс: используйте источник прерываний iba (карту ibaFOB или аналогичную карту)
- Для задач с временем выполнения < 5 мс: используйте режим turbo

Совет

Время отклика также можно контролировать посредством настройки опций компилятора: меню "Инструменты – Опции – Опции режима исполнения" ("Extras – Options – Runtime options")..

- Размер (Size) – настройка по умолчанию: режим интерпретации (UCODE) с возможностью использовать точки останова в структурированном тексте (ST breakpoints)
- Оба (Both) – сочетание режима интерпретации и режима собственного кода, использование точек останова невозможно
- Скорость (Speed) – только режим собственного кода, нет точек останова и доступны не все функции LREAL (например, все экспоненциальные функции)

Помимо этого в ibaLogic существуют различия между режимами "Soft PLC" и "Измерение". Информация касательно настроек содержится в разделе 9.2.1 "Общие настройки".

Измерение

Использование этого режима гарантирует, что ibaLogic не потеряет ни одного входного значения. Это касается также случаев, когда необходимо приостановить отдельные задачи в ibaLogic. Исполнительная система ibaLogic обеспечивает поступление данных через равные промежутки времени за заданный интервал задачи. Если происходит приостановка задачи, то система компенсирует время цикла. В результате временной приостановки задачи может получиться так, что ibaLogic будет выполнять вычисление только тех значений, которые относятся к "прошлому".

Тем не менее, программа обеспечивает, например, наличие равноудаленных и корректных значений для FFT-анализа. Постоянная приостановка или блокировка приводит к переполнению буфера. Такая пользовательская настройка неприемлема.

Пользователь должен также учитывать различия в режимах работы, связанные со считыванием входных сигналов аппаратного обеспечения.

В режиме "Измерение" буферизация состояния входных сигналов аппаратного обеспечения выполняется в соответствии со сконфигурированным интервалом задачи. После выхода из программы при последующем ее запуске она начинает обработку с наиболее старых значений в буфере. Иными словами, режимы работы "Soft PLC" и "Измерение" идентичны в том случае, если время обработки данных < длительности интервала. Если время обработки данных превышает длительность интервала, то в режиме "Измерение" произойдет переполнение буфера значений.

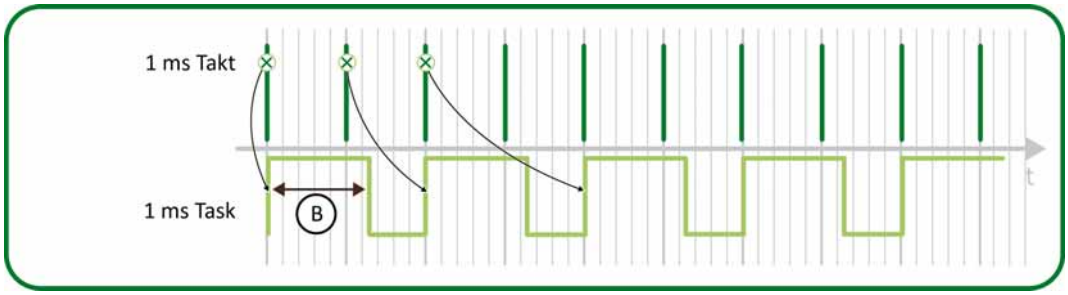


Рис. 122: Переполнение буфера - сдвиги

B Время вычисления

Пример. Обозначения темно-зеленого цвета соответствуют такту 1 мс, на базе которого сохраняется значение, которое обрабатывается при запуске задачи (обозначение светло-зеленого цвета). Черные стрелки соответствует значению, которое обрабатывает задача, а также обозначают постепенное переполнение буфера. Время вычисления задачи превышает 1 мс, соответственно, происходит временной сдвиг.

Soft PLC (программный контроллер)

Этот режим работы предназначен для задач по управлению и регулированию. В этом режиме ibaLogic обрабатывает только последние состояния сигналов. В отличие от режима "Измерение" в данном случае не имеет значения, если какие-либо значения будут потеряны. С другой стороны, желательно, чтобы были доступны (насколько это возможно) текущие данные, т.е. данные последнего цикла передачи значений ввода-вывода.

Данные со входных ресурсов считываются каждый цикл перед выполнением первой задачи. Возраст данных из ресурсов определяется базовым циклом iba-Logic, который можно настроить. Если, например, этот цикл составляет 10 мс, а первая задача сконфигурирована с циклом 50 мс, то она выбирает данные гарантировано не старше 10 мс. Данные, однако, могут быть новее.

В обоих режимах выходные значения записываются каждой задачей, входящей в цикл, в конце времени, необходимого для ее вычисления (при условии, что выходные ресурсы вообще включены в план).

В режиме "Soft PLC" текущее состояние входных сигналов аппаратного обеспечения считывается и обрабатывается в начале задачи.

Распределение времени в случае нескольких задач

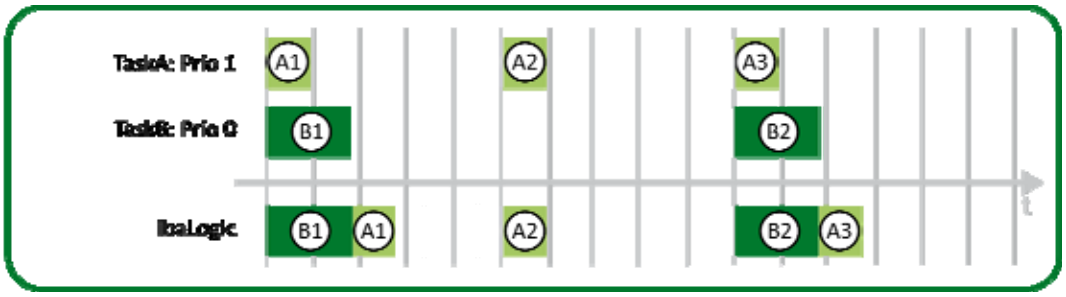


Рис. 123: Вычисление без переполнения буфера - 2 задачи с разным интервалом и приоритетом

Два ряда сверху обозначают отдельные задачи в теоретической последовательности вычислений, выполнение которых происходит независимо друг от друга. Номера соответствуют триггерам для задач (1-й триггер, 2-й триггер…).

Интервальная задача A с длительностью интервала 5 мс отображается в самом верхнем ряду. Приоритет этой задачи - 1, т.е. ниже по сравнению с задачей B (12 мс), приоритет которой - 0 (второй ряд). Ширина полосы (импульс) эквивалентна времени вычисления, которое расходует программа на выполнение задачи. Фон представляет собой координатную сетку тактового цикла. Импульс всегда начинается через установленный интервал.

На практике, задачи, как правило, выполняются последовательно, как это отображено на самом нижнем ряду. После запуска ibaLogic проверяет, какие задачи должны быть выполнены и выполняет их одну за другой в соответствии с указанным приоритетом…

Для наглядности на рисунке выше изображены очень большие периоды вычисления. На практике время вычисления измеряется в мкс, т.е., например, 20 задач обрабатываются без проблем за 5 мс (эмпирическое значение).

Неблагоприятное распределение времени

Если вы настроите более длительное время вычисления для задач A и B, то произойдет приостановка или временной сдвиг. Это означает, что выполнение задачи не начнется в ожидаемый момент времени, поскольку не будет завершено вычисление другой программы. Задача с более высоким приоритетом начинает выполняться в требуемый момент времени.

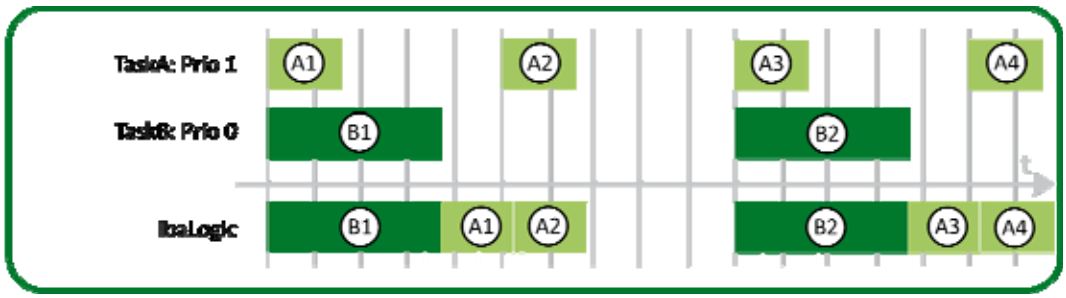


Рис. 124: Вычисление задач с временным сдвигом (приостановка)

Время вычисления обоих задач было выбрано таким образом, что для их выполнения требуется более 5 мс, следовательно, программа не может приступить к вычислению задачи A через заданный интервал. Если был сконфигурирован базовый цикл 1 мс, то каждый цикл программа выполняет проверку, чтобы определить, какая задача должна быть инициирована. В этом примере показаны две задачи: A (5 мс, приоритет 1) и B (10 мс, приоритет 0). Эти задачи программа вносит во внутренний список в порядке их приоритетности, а затем приступает к их выполнению.

Комментарии к вышеописанному примеру

Задания во внутреннем списке изображены на рис. "Вычисление задач с временным сдвигом - фрагмент".

Программа ibaLogic обнаруживает, что необходимо выполнить задачу A (5 мс, приоритет 1) и задачу B (10 мс, приоритет 0), поэтому она вносит их во внутренний список заданий в порядке их приоритетности. Задачи, выполнение которых уже началось, удаляются из списка заданий, и добавляются новые: таким образом список приобретает вид, как на рис. выше.

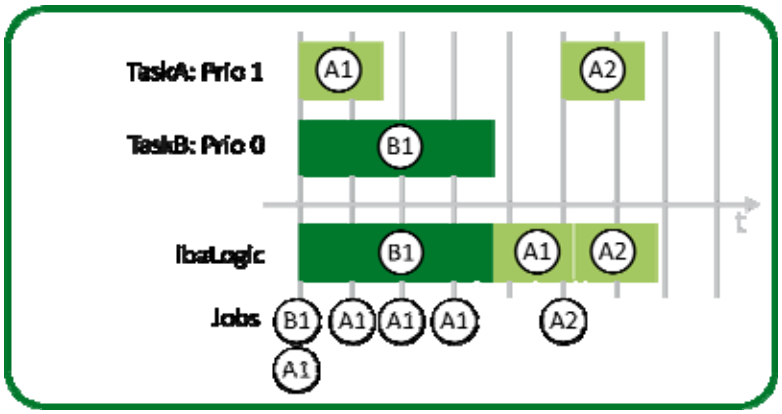


Рис. 125: Вычисление задач с временным сдвигом - фрагмент

Вычисление задач с временным сдвигом (приостановка)

Предположим, что для задачи A был сконфигурирован интервал 2 мс (при аналогичном времени вычисления программы). В этом случае некоторые **циклы будут потеряны**.

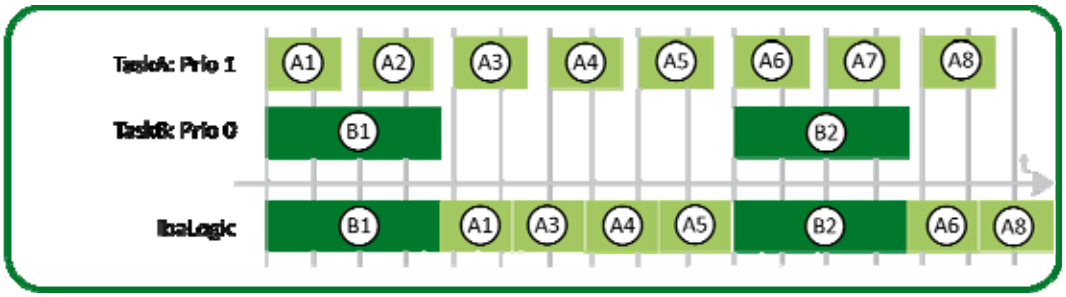


Рис. 126: Вычисление задач с временным сдвигом

При изменении приоритета задач получается иная ситуация.

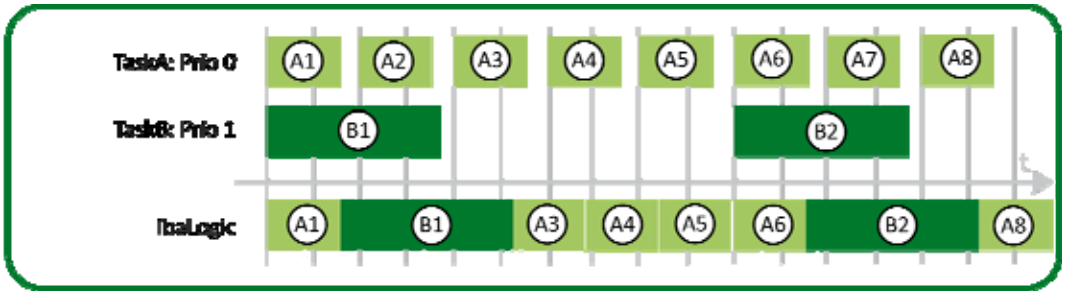


Рис. 127: Вычисление задач с временным сдвигом (изменение приоритета)

Также необходимо принимать во внимание разницу между режимами "Soft PLC" и "Измерение".

В режиме "Soft PLC" входы аппаратного обеспечения всегда считываются в начале задачи (обозначение "x" на рис. ниже).

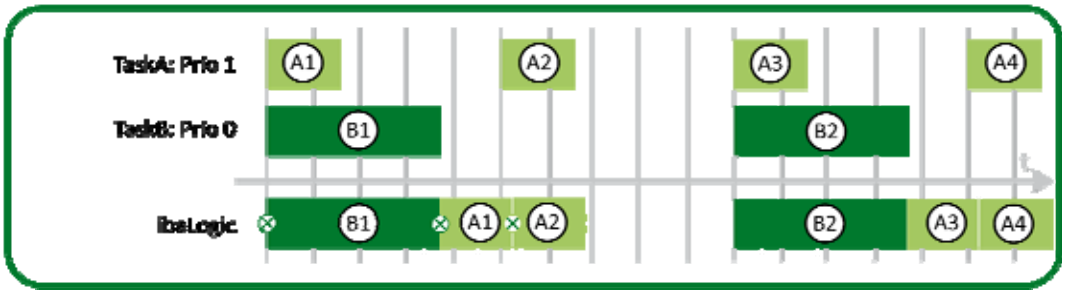


Рис. 128: Входы аппаратного обеспечения в режиме "Soft PLC"

В режиме "Измерение" ситуация будет иная.

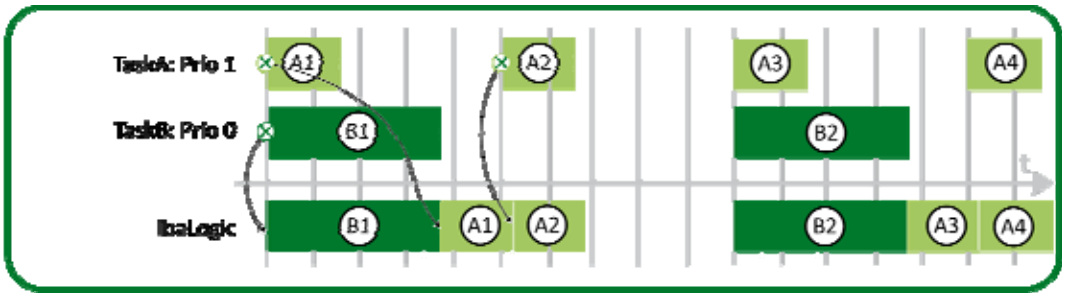


Рис. 129: Входы аппаратного обеспечения в режиме "Измерение"

В данном случае выполняется буферизация входных сигналов, а их вычисление может осуществляться с задержкой в случае временного сдвига или приостановки задачи.

Важно

Как правило, временной сдвиг или приостановка задачи происходят, если сумма времени вычисления программы превышает длительность наименьшего интервала. Если временной сдвиг присутствует постоянно, происходит переполнение буфера. Программы и компьютер не могут работать в соответствии с требованиями пользователя. В случае недолгого временного сдвига или приостановки задачи на короткое время, только от приложения зависит, приведет это к негативным воздействиям или нет.

Данные записываются на выходы аппаратного обеспечения в течение следующего основного тактового цикла после завершения времени вычисления. Таким образом, может быть полезно сделать основной тактовый цикл более коротким по сравнению с интервалом задачи. Если, например, время вычисления составляет 50 мкс, интервал задачи равен 5 мс и основной тактовый цикл - 1 мс, то данные записываются на выходы после 1 мс.

11.3 Отладка

Могут возникнуть следующие ошибки:

- Программные ошибки
- Ошибки компиляции

11.3.1 Программные ошибки

Часто возникающие ошибки в программах:

- Ошибки в пользовательских функциональных блоках
- Деление на 0
- Неверные последовательности сигналов
- Неверный порядок или последовательность вычислений

Ошибки в пользовательских функциональных блоках

Для того чтобы "вылавливать" логические ошибки, в ibaLogic V4 существует опция настройки точек останова в функциональных блоках, чтобы пользователь мог проверить выполнение своего ST-кода. Более подробная информация содержится в пункте 5.4.2 "Редактор структурированного текста".

Деление на 0

Если в окне событий появляется следующее сообщение, то это означает, что было выполнено деление на 0.

```
"Exception: OnlineServer: PMAC Status: Division by Zero in
program.functionblock, Offset 0x0022, Stack 0x0001"
```

Это сообщение содержит информацию о том, где именно произошло деление на 0. В примере выше ошибка произошла в функциональном блоке "functionblock" в программе "program".

Неверные последовательности сигналов

Для проверки вычисляемых значений, в ibaLogic V4 предусмотрен инструмент ibaPDA Express. Этот инструмент используется для отображения кривых сигналов в реальном времени, что позволяет пользователю проследить, возвращает ли блок ожидаемые выходные значения с различными входными параметрами.

Последовательность вычисления

Если несмотря на отсутствие ошибок в функциональных блоках и макросах вычисление выполняется некорректно, то возможно проблема заключается в последовательности вычисления.

Чтобы проверить блоки, которые вычисляются в первую очередь, пользователь может посмотреть последовательность вычисления соответствующей программы и выяснить, нет ли там ошибок.

Более подробная информация содержится в пункте 4.2.3.5 "Порядок вычисления ".

Если программа содержит пути с обратной связью, то необходимо знать блок, который располагается первым или последним в последовательности вычислений.

11.3.2 Ошибки компиляции

Несмотря на то что синтаксис ST пользовательского функционального блока проверяется генератором блоков перед компиляцией, в некоторых случаях может произойти сбой компиляции IL-кода.

В этом случае в окне событий появится соответствующее сообщение об ошибке.

Если в окне сообщений появилось следующее сообщение, прокрутите его вверх, чтобы увидеть первое сообщение об ошибке.

Пример:

```
[01.03.2010 14:19:14] [<Computername>] [ibaLogicClient] Info: Generation started...
[01.03.2010 14:19:14] [<Computername>] [ibaLogicClient] Info: Compilation started...
[01.03.2010 14:19:14] [<Computername>] [ibaLogicClient] Info: TIMER Generation: Ticks since start of IL generation: 31, that is 0.03 seconds
[01.03.2010 14:19:15] [<Computername>] [ibaLogicClient] Exception: IL compilation failed: Compilation ended with errors.
[01.03.2010 14:19:15] [<Computername>] [ibaLogicServer] Info: Building resource C:\Documents and Settings\<Username>\Application Data\ibaLogic\NewWorkspace\NewProject\$ENV$\Resource\Resource.MAK.
C:\DOCUMENTS AND SETTINGS\<Username>\APPLICATION DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\CustomTypes.typ
C:\DOCUMENTS AND SETTINGS\<Username>\APPLICATION DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\FB_STRINGOUT.POE(3,5,2): E: S3023: Invalid operand type for this operation.
1 error(s), 0 warning(s) -
C:\DOCUMENTS AND SETTINGS\<Username>\APPLICATION DATA\IBALOGIC\NEWWORKSPACE\NEWPROJECT\FB_STRINGOUT.POE.
C:\Documents and Settings\<Username>\Application Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE(2,9,14): E: S3026: Undeclared identifier.
C:\Documents and Settings\<Username>\Application Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE(3,2,6): E: S3005: This is not a function block instance.
3 error(s), 0 warning(s) - C:\Documents and Settings\<Username>\Application Data\ibaLogic\NewWorkspace\NewProject\T00_INPUT.POE.

3 error(s), 0 warning(s).
```

Если происходит сбой компиляции, нужно всегда начинать искать причину с прочтения первого сообщения об ошибке, которое появляется в окне событий.

Для того чтобы обнаружить ошибки, выполните следующие действия:

- Увеличьте окно сообщений, чтобы отобразить все события, которые произошли с момента начала компиляции ("`Compilation started`").
- Просмотрите первое сообщение об ошибке. Все последующие сообщения, вероятнее всего, являются следствием первой ошибки. В примере выше таким сообщением является:
  "`C:\DOCUMENTS AND SETTINGS\<Username>\APPLICATION DATA\IBALOGIC\NEWWORKSPACE \NEWPROJECT\FB_STRINGOUT.POE(3,5,2): E: S3023: Invalid operand type for this operation`"
- Скопируйте путь доступа к блоку, который содержит ошибку, и вставьте в адресную строку проводника. В примере выше это следующий путь:
  "`C:\DOCUMENTS AND SETTINGS\<Username>\APPLICATION DATA\IBALOGIC \NEWWORKSPACE\NEWPROJECT`"
- Таким образом вы обнаружите блок с ошибкой, в нашем случае: "`FB_STRINGOUT.POE`". Откройте его в редакторе ASCII с функцией отображения номеров строк, например NotePad++.
  Вы увидите, например, следующий программный код (пример приведен выше)

```
 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
```

- В конце блока содержатся три числа, например (3,5,2).
  Эти числа соответствуют следующему:
  - 1-е число: область, в которой произошла ошибка.
    1 = Имя программы / имя FB / имя функции (строка 1 выше)
    2 = Диапазон переменных, начинается с "VAR" (строка 2 выше)
    3 = Программа, начинается после END_VAR (строка 9 выше)
  - 2-е число: номер строки в разделе (5 соответствует 13 строке)
  - 3-е число: номер столбца (или вкладки) в рассматриваемой строке (2)
- строка с ошибкой: "ADD (i1 … ".
  Найдите последний комментарий над этим выражением. Здесь содержится исходный текст выражения ST
  `o1 := 'TestString' + int_to_string(i1);`

- Ошибка заключается в следующем: операнд ADD недопустим для строк. Возможная причина: в других компиляторах, например ibaLogic V3, строки можно объединять с помощью '+'. Компилятор ibaLogic V4 воспринимает '+' только как знак сложения. Чтобы соединить строки, используйте функцию CONCAT.
- Корректным выражением для ibaLogic V4 будет следующее:

```
1  v1 := int_to_string(i1);
2  o1 := concat('TestString',v1);
```

В некоторых случаях более подробную информацию, которая поможет устранить ошибку, можно получить из файла "CompilerOut.txt".

В ОС Windows файл "CompilerOut.txt" сохраняется в папку:

- Немецкая версия системы

```
C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten
\ibalogic\<Workspacename>\<Projektname>\$GEN$\
```

- Английская версия системы

```
C:\Documents and Settings\<Username>\Application Data\ibalogic
\<Workspacename>\<Projektname>\$GEN$\
```

11.4 Ограничения производительности

ibaLogic V4 разработана для 32-х битной версии Windows и имеет некоторые ограничения, связанные с архитектурой программы:

- Максимальный размер оперативной памяти: 4 Гб
- Максимальный размер памяти (т.е. память, которую может занимать исполнительная система)
  - Для платформы Win XP: 2 Гб
  - Для платформы PADU-S-IT: 32 Мб

Программа **Microsoft SQL Server 2005 Express**, используемая ibaLogic V4, имеет следующие ограничения производительности, связанные с системой:

- Максимальный размер базы данных: 4 Гб
- Макс. 16 экземпляров на одном компьютере
- Поддерживает только 1 ЦП и 1 Гб оперативной памяти

Помимо этого существуют ограничения, вызванные использованием интегрированного компилятора ibaLogic.

Макс. размер сегмента составляет 64 КБ. Это означает, что количество переменных в блоке (функциональном или макросе) ограничено. Если максимальное количество в 65 292 байтов превышено, программа выдает сообщение об ошибке.

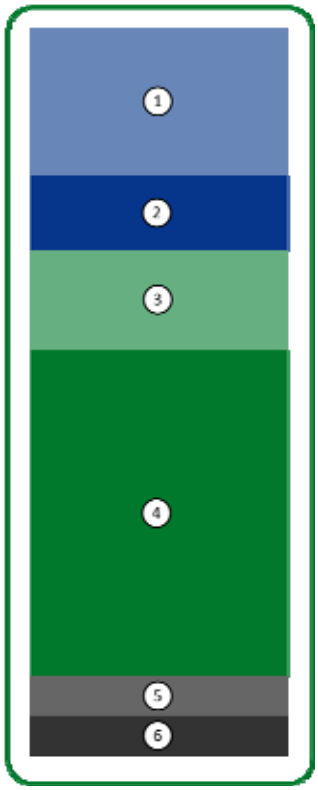
Пример:

У вас есть массив данных из 8 158 элементов LREAL, каждый из которых занимает 8 байтов. Это означает, что в функциональном блоке этот массив занимает 65 264 байта, плюс заголовок массива 12 байтов, что дает в сумме 65 276 байтов.

Вы можете создать функциональный блок только с одним входом и выходом, который будет занимать 4 байта, что позволит соблюсти ограничение по объему (поскольку заголовок занимает еще 8 байтов).

Элемент	Кол-во байтов
Заголовок функц. блока	8 байтов
Заголовок массива	12 байтов
Массив [0 … 8157] LREAL	8 158 * 8 байтов = 65 264 байта
Вход i1 (DINT)	4 байта
Выход o1 (DINT)	4 байта
Итого	65 292 байта

Оставшиеся 243 байта необходимы компилятору для административной информации. На рисунке ниже в упрощенной форме изображена структура сегмента данных.



1	Административная информация
2	Заголовок функц. блока
3	Заголовок массива
4	Массив
5	Вход
6	Выход

Рис. 130: Сегмент данных (64 КБ)

В рамках проекта может быть создано до 600 программ / задач, но это скорее теоретическое ограничение, вызванное соображениями ясности.

Количество проектов в рабочей области ограничено только максимальным объемом базы данных на сервере.

Если вы используете другой SQL-сервер, обратитесь к вашему системному администратору, чтобы узнать требования касательно объема данных.

12 Правила программирования

В каждой среде программирования существует риск создания плохо структурированной программы, разобраться в которой будет сложно как собственно программисту, так и заказчику или любому другому человеку, которому придется с ней работать.

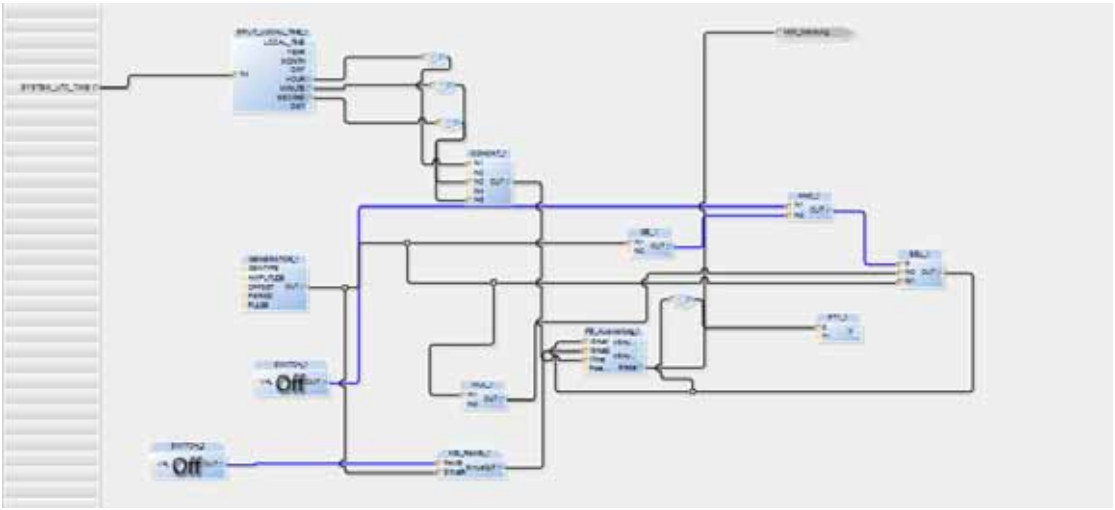


Рис. 131: Пример плохо структурированной программы

Программу из примера выше можно реструктурировать в соответствии с рекомендуемым решением.

Подход к решению

Две задачи имеют следующую структуру:

- Задача 1 Генерирование данных
- Задача 2 Обработка данных

Рис. 132: Пример структурированной программы

Пользователь не обязан соблюдать следующие рекомендации, однако они упрощают работу с ibaLogic.

- Распределяйте функции между несколькими задачами / программами, которым присвоены информативные имена.

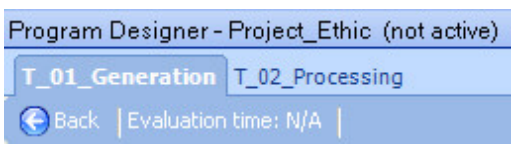


Рис. 133: Распределение задач / программ

- Создайте одну задачу для входов аппаратного обеспечения и одну - для выходов. Задайте степень приоритетности таким образом, чтобы задача для входов обрабатывалась в первую очередь, а для выходов - в последнюю.
- В рамках одной задачи используйте внутристраничные коннекторы: большое количество пересекающихся линий загромождают схему программы.
- Добавьте комментарии к подфункциям.

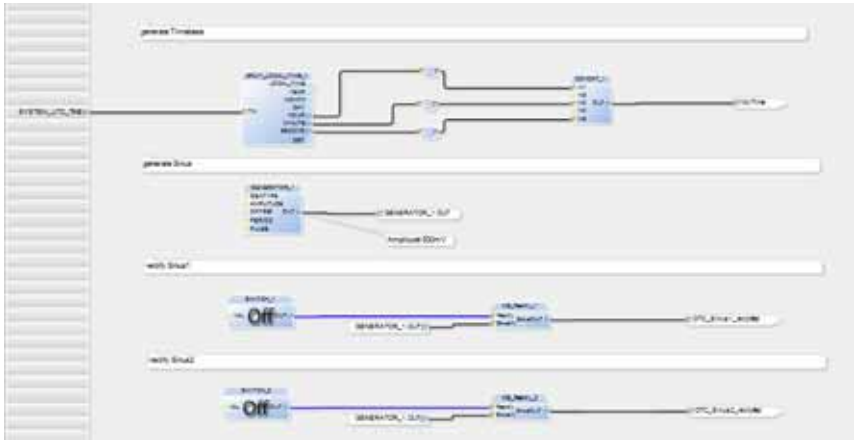


Рис. 134: Комментарии

- Объедините компоненты программы, которые нужно использовать несколько раз, в макросы и создайте для них описание и комментарии.
- Объедините сложные соединения, относящиеся к функции, в макрос, чтобы внести дополнительную ясность в программу.
- Используйте комментарии и описания даже в рамках одного функционального блока. Мы рекомендуем использовать заголовки, которые содержат индекс и значимую информацию.

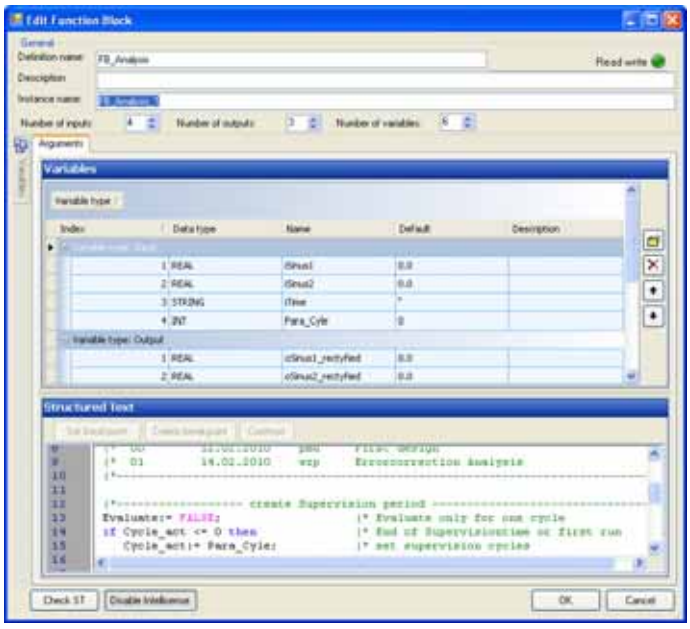


Рис. 135: Комментарии к программному коду

- Расположите блоки в задаче таким образом, чтобы они совпадали с последовательностью вычислений (из верхней левой части к нижней правой).
- Присваивайте коннекторам вне задач префикс, например "OTC_", или "OPC_", если OTC используется для HMI-системы.
- Для обозначения внутристраничных коннекторов используйте префикс "IPC_". Если "OTC_test" используется как вход, то его также можно продолжать использовать внутри как "IPC_test" без возникновения конфликта имен.
- Сконфигурируйте короткие префиксы для имен блоков, макросов и их коннекторов, например: "FB_", "MB_".
  (Соответствующие настройки находятся в пункте меню "Инструменты– Опции – Редакторы – Функциональные блоки" ("Extras – Options – Editors – Function Blocks")).
- Пользовательским типам данных присваивайте такие имена, которые сообщают информацию об их логическом значении (например, ST_ROLLING) или содержимом (например, AR_64REAL).
- Если необходимо, измените расположение линий таким образом, чтобы каждую можно было легко отследить. Избегайте наложения линий.
- Можно также использовать опцию увеличения блоков, чтобы были лучше видны имена коннекторов и было легче отследить линии.

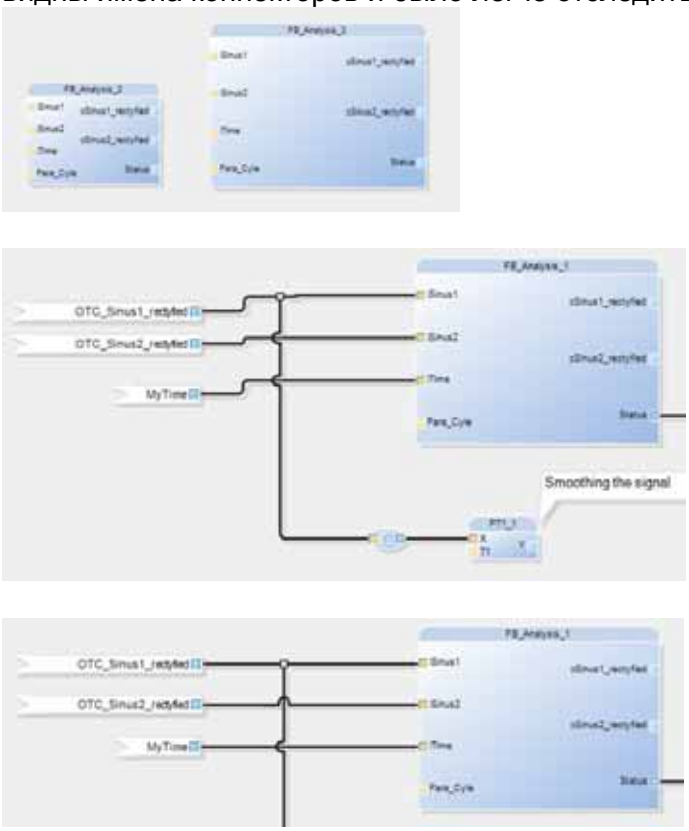


Рис. 136: Пример увеличенного отображения

- В целом, на стадии программирования нужно стараться "отлавливать" возможные источники ошибок, как то:
  - Деление на 0

- Попытка доступа к данным за рамками массива
- Бесконечные циклы

13 Удаление ibaLogic

Существует два способа удалить ibaLogic:

- с помощью установленной функции деинсталляции
- через панель управления

DANGER

Возможные проблемы при активации и деактивации функций!

При активации/деактивации функций и служб (PMAC, OPC …), которые оказывают непосредственное влияние на отклик системы, возможно причинение травм людям и нанесение ущерба оборудованию.

При работе с системой соблюдайте правила безопасности!

Важно

Только пользователи с правами администратора могут удалить ПО ibaLogic. Обратитесь к вашему системному администратору.

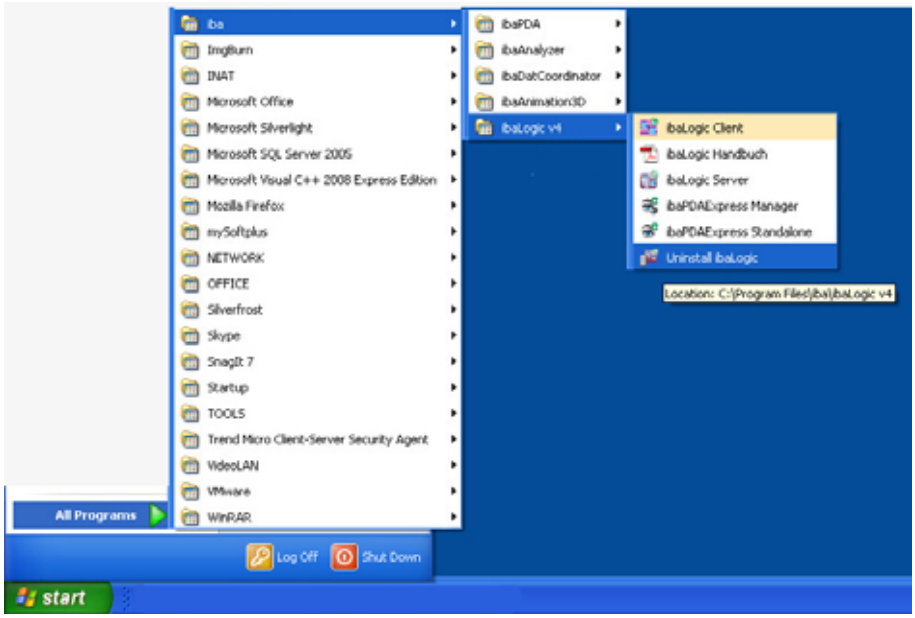
13.1 Удаление программы с помощью функции деинсталляции

Необходимое условие

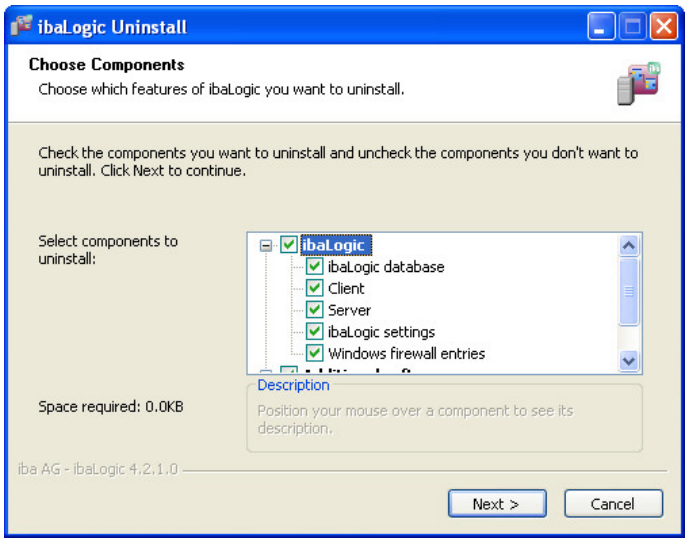
- Все программы ibaLogic закрыты.

Последовательность действий

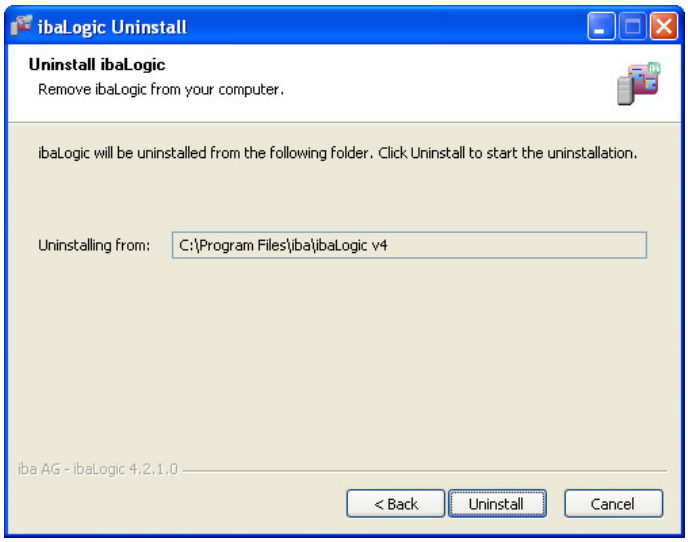
1. Выберите "Пуск – iba – ibaLogicv4 – Удалить ibaLogic" ("Start – iba – ibaLogicv4 – Uninstall ibaLogic").



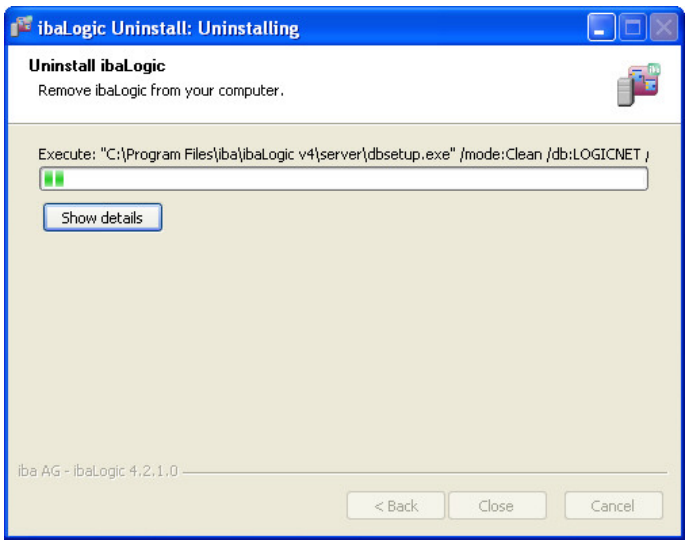
2. Выберите компоненты, которые вы хотите удалить.



3. Запустите процесс деинсталляции, щелкнув кнопку <Удалить>.

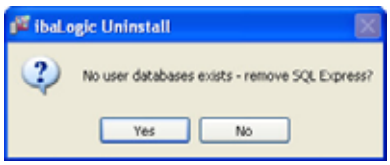


4. Закройте диалоговое окно, щелкнув кнопку <Закрыть>.
При появлении каких-либо сообщений требуется просто их подтвердить.



Важно

Если в установочной папке есть резервные копии базы данных, то в процессе деинсталляции появится сообщение с просьбой подтвердить их удаление. Если вы щелкнете <Нет>, то копии не будут удалены.



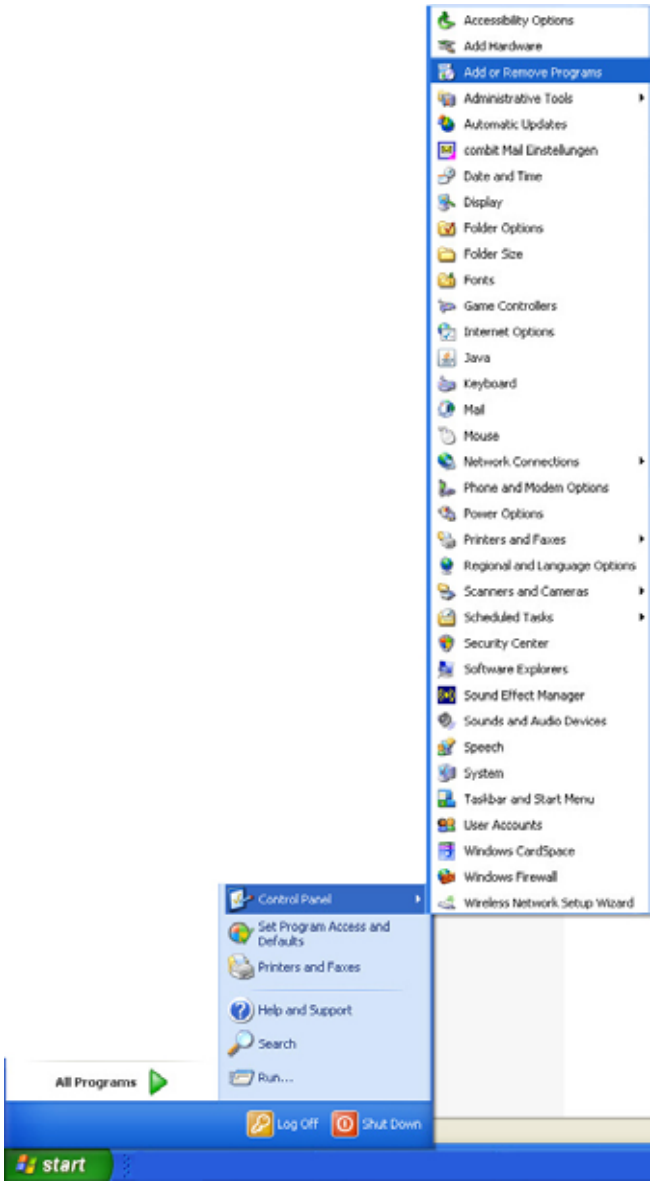
13.2 Удаление через панель управления

Необходимое условие

- ❑ Все программы ibaLogic закрыты.

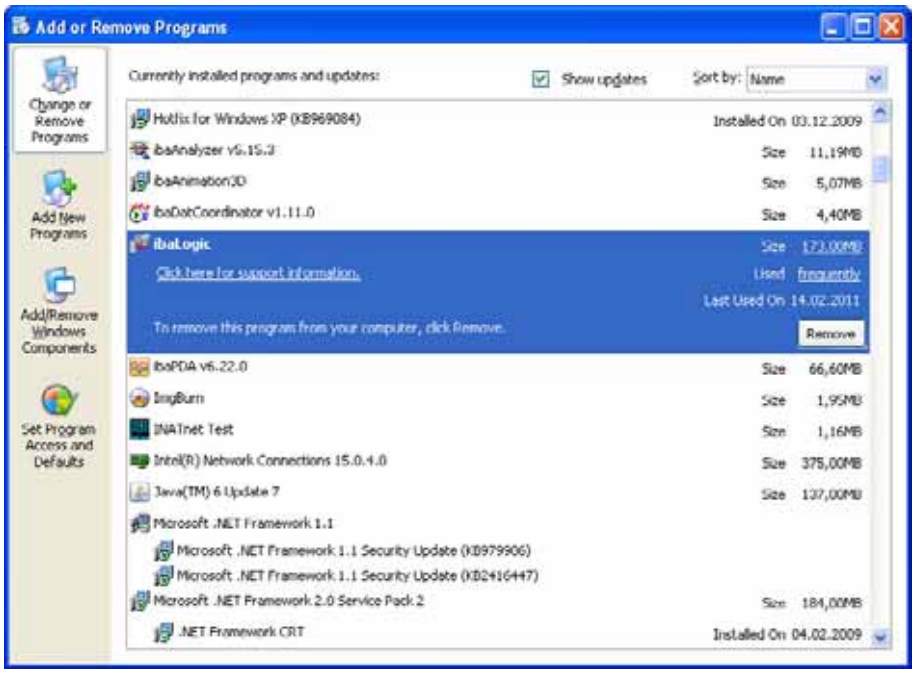
Последовательность действий

1. Выберите "Пуск – Панель управления – Установка и удаление программ" ("Start – Control Panel – Add or Remove Programs").



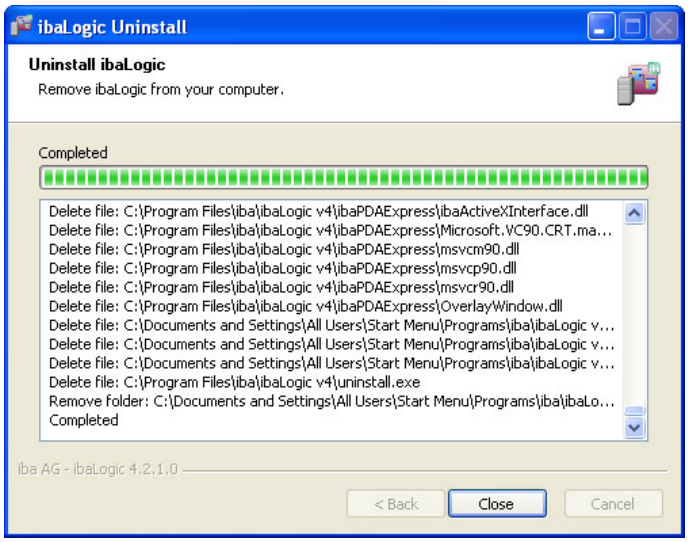
2. Щелкните двойным щелчком по пункту меню Установка и удаление программ. Откроется окно установки и удаления программ.
3. В списке установленных программ и обновлений выберите "ibaLogic".

4. Щелкните кнопку <Удалить> (<Remove>).



5. При появлении каких-либо сообщений требуется просто их подтвердить.

6. Закройте диалоговое окно, щелкнув кнопку <Закрыть>.
При появлении каких-либо сообщений требуется просто их подтвердить.

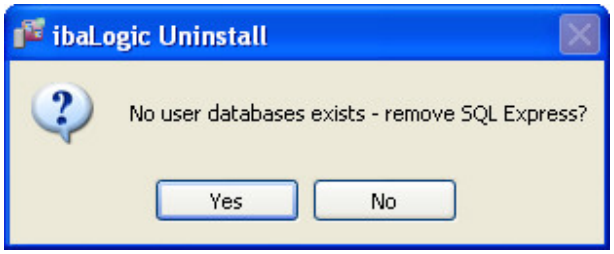


Результат

Программа ibaLogic деинсталлирована.

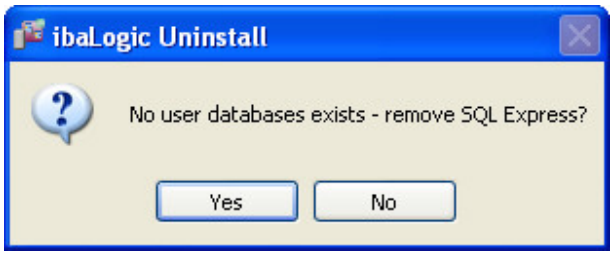
13.3 Сообщения во время деинсталляции

SQL Express будет удалена при подтверждении сообщения нажатием кнопки <Да>. Созданная при установке база данных будет удалена (*.ldf, *.mdf).



Сообщение, которое требуется подтвердить, при удалении ibaLogic.

Сообщение, которое требуется подтвердить, при удалении ibaLogic.



Приложение

A. Примеры из практики

Цель этого раздела помочь начинающим пользователям в освоении ibaLogic. Небольшая программа, которая используется как пример, помогает пользователю разобраться в программе и, прежде всего, демонстрирует подход iba к созданию эргономичной реализации CFC, а также позволяет понять значение онлайн-обновления статических и динамических переменных.

Поскольку данная программа является лишь примером для понимания основ, в ней не задействованы все функции ibaLogic.

A.1. Первые шаги - пример проекта

Требуется создать программу для генерирования синусоидального сигнала. В зависимости от состояния переключателя для этого синусоидального сигнала нужно добавить регулируемое смещение. Этот пример можно создать, используя только стандартные функциональные блоки. Тем не менее, для иллюстрации широких возможностей и свойств интегрированного языка программирования - структурированного текста (ST) - программирование примера имеет смысл также выполнить с помощью этого языка.

Входные сигналы и результат должны отображаться в виде графиков тренда.

Программа-пример должна имитировать работу программы в режиме онлайн, т.е. коннекторы должны обновляться постоянно. Изменение цвета программы на розовый означает, что программа в настоящий момент выполняет свои функции: агрегат работает.

Если выходы уже подключены (исполнительные устройства, двигатели и т.д.), то они сразу же реагируют на изменения, вносимые в программу. Стандартная процедура - программирование, компиляция, компоновка, загрузка и запуск - выполняется автоматически в фоновом режиме. Помимо этого, в целях облегчения работы с программой, в проекте принимаются предварительно установленные значения для проекта и имен программ.

A.1.1. Упражнение, часть 1

A.1.1.1. Описание задачи

Периодически меняющееся значение, получаемое от генератора (синусоидальный сигнал), должно суммироваться со значением, получаемым от реостата, в зависимости от положения переключателя:

Положение 1: генератор + генератор

Положение 2: генератор + реостат

Все переменные и, конечно, результаты должны отображаться графически в виде кривой.

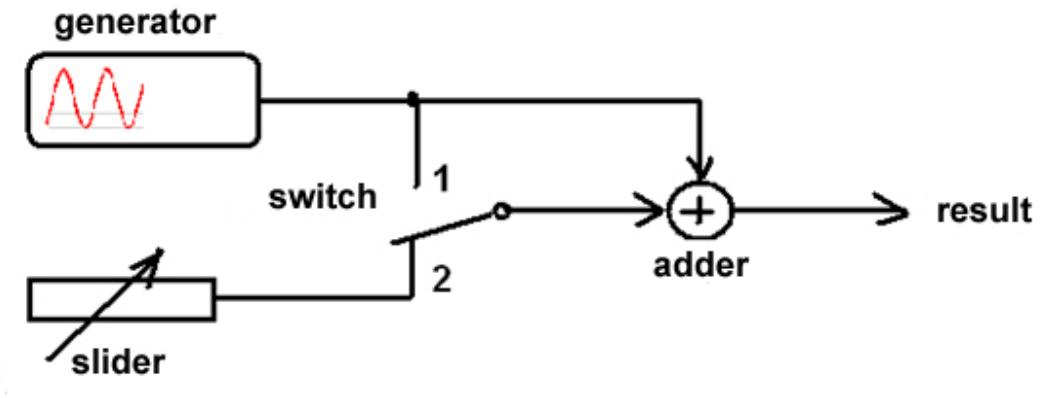


Рис. 137: Принципиальная схема упражнения

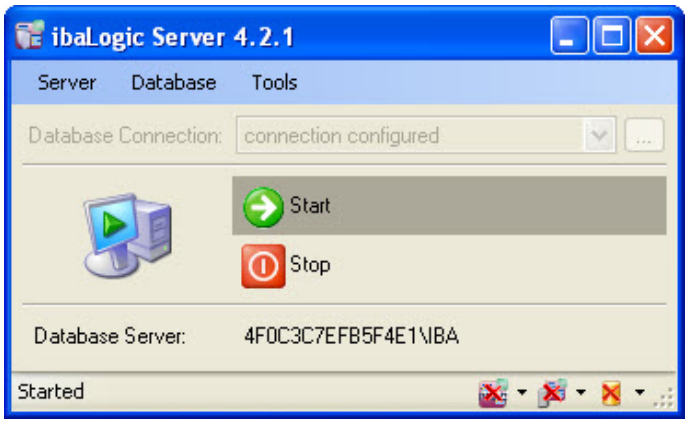
В первой части упражнения образец программы должен быть реализован с помощью функциональных блоков (FB), которые доступны как стандартные блоки в ibaLogic.

Поскольку лабораторное оборудование (генератор функций, реостат и клавиши) не должно быть соединено со входами ibaLogic, то такие эффективные функции компилируются как **Специальные** (**Specials**) и могут размещаться, как остальные блоки. В целях тестирования схемы можно выполнять активные действия с **переключателем** (**Switch**) и **реостатом** (**Slider**).

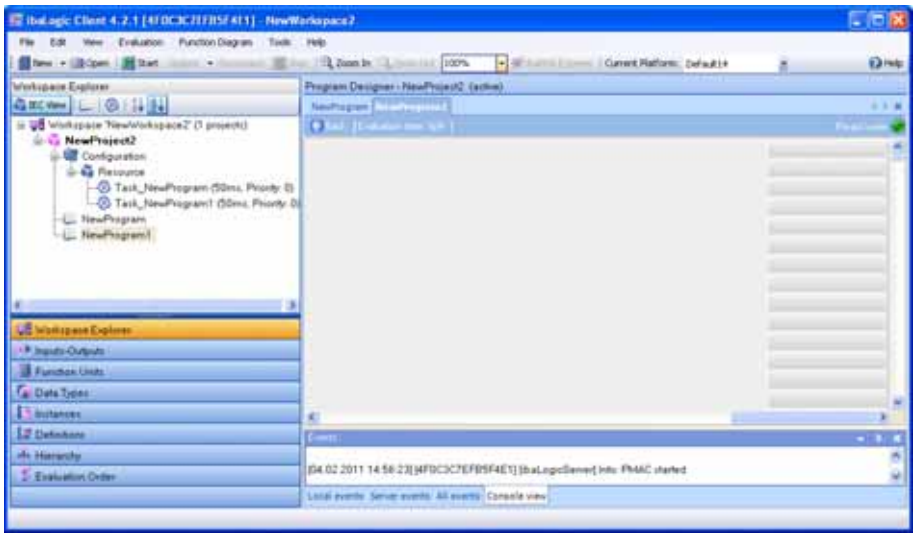
A.1.1.2. Запуск сервера и клиента ibaLogic

Последовательность действий

1. Запустите сервер ibaLogic двойным щелчком по ярлыку "ibaLogic Server" на рабочем столе.
   После этапа инициализации откроется диалоговое окно сервера ibaLogic. По умолчанию при открытии диалогового окна сервер запускается автоматически.



2. Запустите клиент двойным щелчком по ярлыку "ibaLogic Client" на рабочем столе.
   После этапа инициализации откроется диалоговое окно клиента ibaLogic.



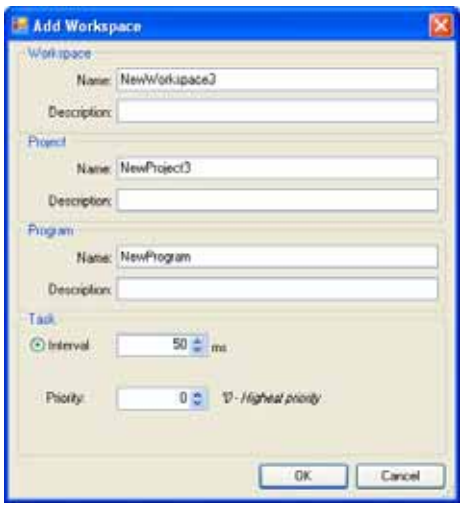
Комментарий

Под основным окном программы находится окно событий, в котором создается список действий и ошибок (если таковые возникали). Если при запуске сервера или клиента появляются сообщения об ошибке, см. настоящее руководство на предмет их устранения.

A.1.1.3. Создание нового проекта

Последовательность действий

1. Щелкните кнопку <Новый> в панели инструментов.
Появится диалоговое окно "Добавить рабочую область" ("Add Workspace").



2. Подтвердите введенную информацию щелчком по кнопке <OK>.

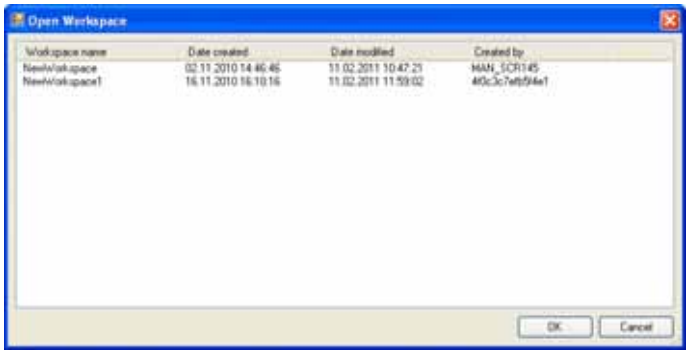
Если вы не меняете данные, введенные по умолчанию, то проект будет называться "NewProject1" и будет содержать только одну программу - "NewProgram". Интервал 50 мс, заданный по умолчанию, подходит для нашего примера.

Комментарий

Если после запуска упражнения в ibaLogic вам пришлось прервать сеанс или вы просто закрыли все программы (сначала клиент, затем сервер), то нет необходимости начинать все заново с нажатия <Новый>.

В таком случае для продолжения нужно щелкнуть кнопку "Открыть" ("Open"). В окне "Открыть рабочую область" ("Open Workspace") содержится информация о том, кто и когда завершал предыдущие сеансы.

Если вы щелкнете по последней строке в этом окне, будет восстановлено последнее состояние проекта.



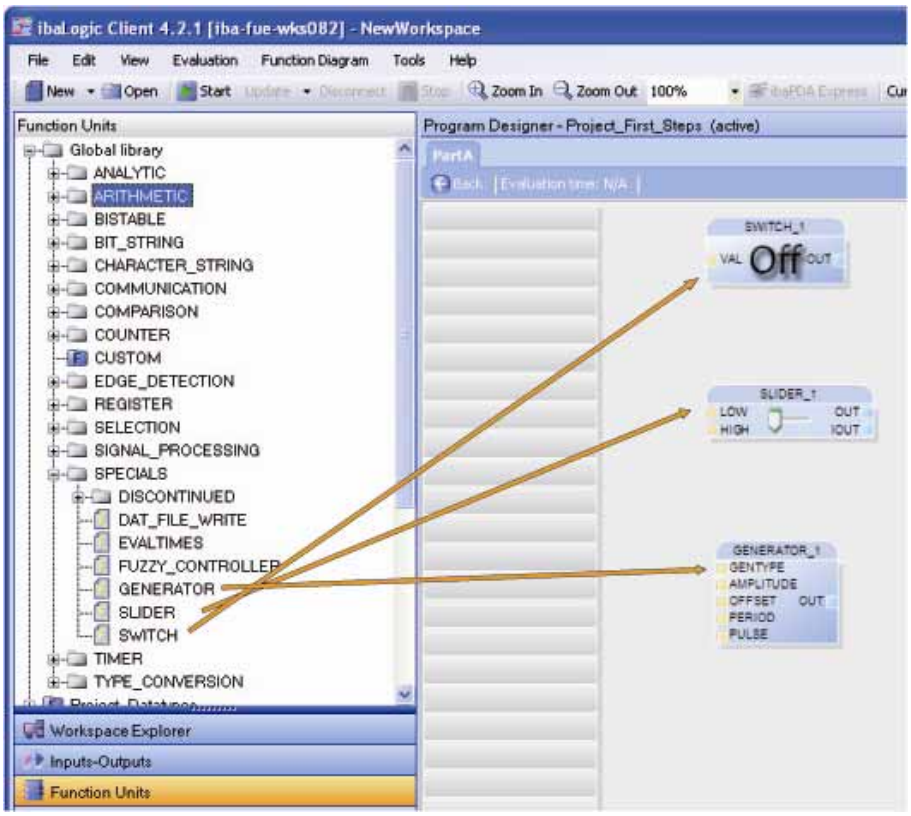
A.1.1.4. Размещение инструментов для тестирования

Для описания задачи требуются следующие функциональные блоки:

- "Генератор" ("Generator")
- "Переключатель" ("Switch")
- "Реостат" ("Slider")

Последовательность действий

1. Щелкните кнопку <Функциональные единицы> (<Function Units>). В области навигации отобразится дерево каталогов.
2. Откройте папку "Специальные" ("Specials").
3. Нажав и удерживая левую кнопку мыши, перетащите один "Генератор", один "Реостат" и один "Переключатель" в рабочую область области программирования.
   Если при размещении блоков один блок окажется слишком близко к другому, то появится выделенная область, обозначающая "личное пространство" блока, в которое нельзя помещать другие блоки.



4. Расположите блоки в нужной последовательности.

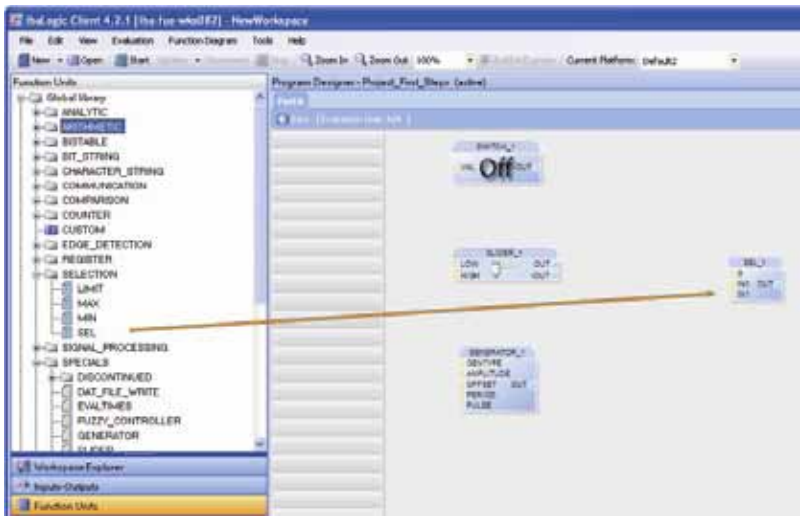
A.1.1.5. Размещение блоков вычисления

Для описания задачи требуются следующие функциональные блоки:

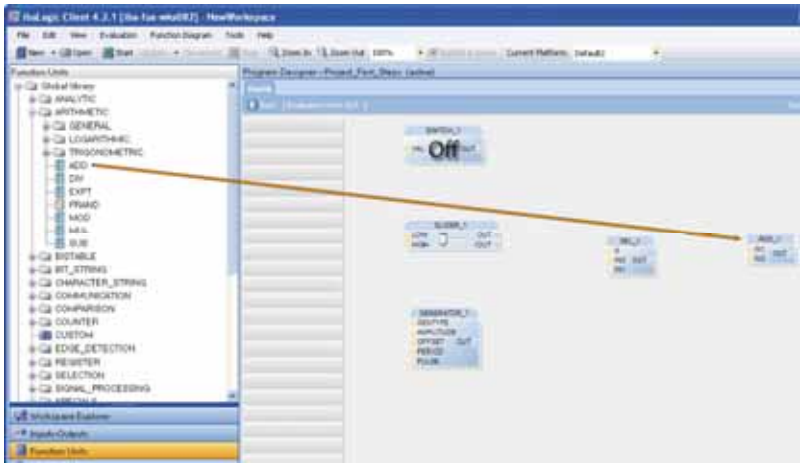
- "Блок выбора" ("Selector")
- "Сумматор" ("Adder")

Последовательность действий

1. Откройте папку "Выбор" ("Selection") в дереве каталогов в области навигации.
2. Нажав и удерживая левую кнопку мыши, перетащите блок выбора (SEL) в рабочую область области программирования.



3. Откройте папку "Арифметические" ("Arithmetic") в дереве каталогов в области навигации.
4. Нажав и удерживая левую кнопку мыши, перетащите сумматор (ADD) в рабочую область области программирования.



Комментарий

Блок SELECTOR (SEL) требует бинарный сигнал, чтобы "принимать решения". Значениям IN0 и IN1 не нужно присваивать формат, они автоматически подстраиваются под соответствующий тип данных. Как и в случае блока выбора, значения, которые добавляются, не должны иметь определенный формат. Формат определяет тот коннектор, который первым соединяется с одним из входов.

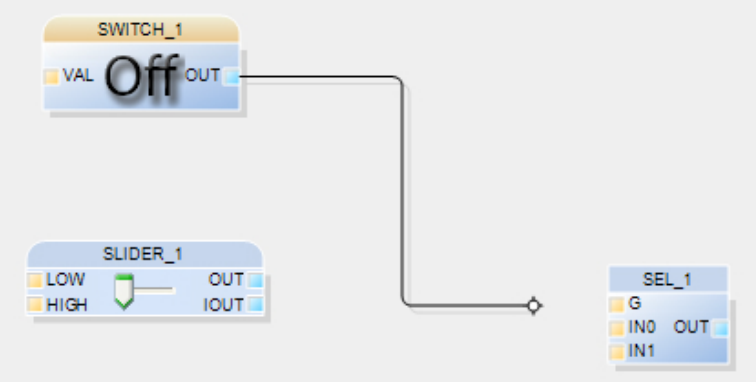
A.1.1.6. Соединение блока выбора с помощью инструментов для тестирования

Необходимо соединить следующее:

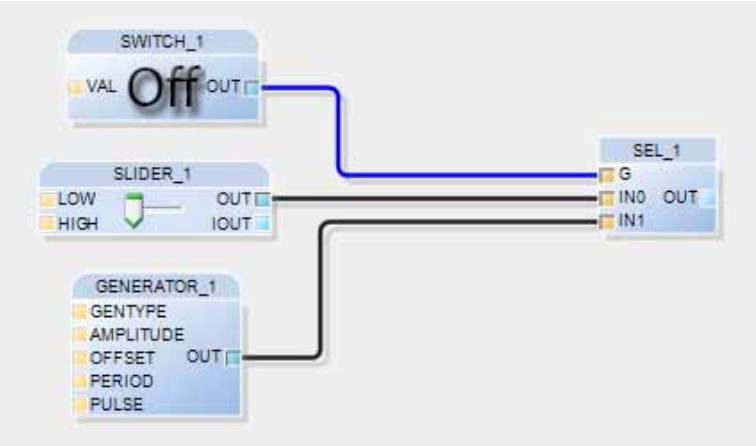
- ❑ Выход переключателя (SWITCH) OUT с "решающим" входом (G) блока выбора (SEL).
- ❑ Выход реостата (SLIDER) OUT со входом ("IN"0) блока выбора (SEL).
- ❑ Выход генератора (OUT) со вторым входом ("IN"1) блока выбора (SEL).

Последовательность действий

1. Наведите курсор мыши на коннектор (OUT) переключателя и, удерживая левую кнопку мыши нажатой, переведите курсор на коннектор (G) блока выбора.



2. Аналогичным образом создайте остальные соединения.



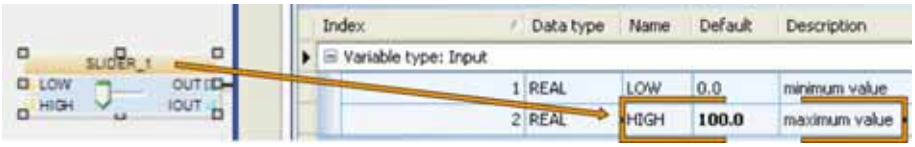
A.1.1.7. Конфигурирование реостата и генератора

Необходимо сконфигурировать следующее:

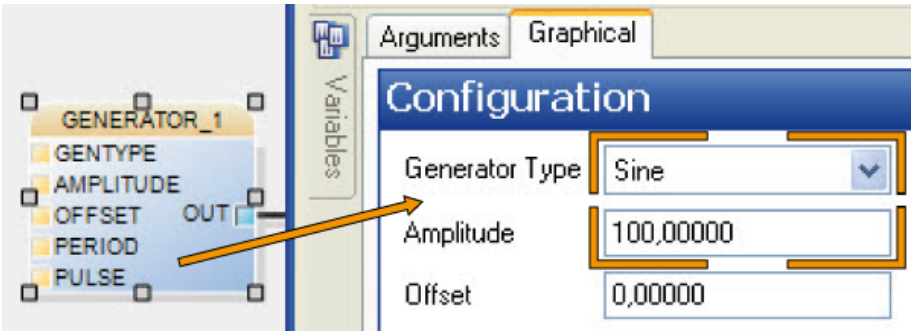
- ❑ "Реостат" ("Slider")
- ❑ "Генератор" ("Generator")

Последовательность действий

1. Щелкните двойным щелчком по реостату. Появится меню для конфигурирования.
2. Установите "максимальное значение" используемого диапазона на 100.0. Реостат работает со значением в диапазоне от 0 до 100, в зависимости от его положения.



3. Откройте конфигурационное меню генератора.
4. Установите тип генератора на синусоидальный.
5. Введите значение амплитуды 100.



При такой настройке параметров генератор формирует синусоидальный колебательный сигнал, который может иметь значение от +100,0 до -100,0.

Для периода колебаний оставьте значение, указанное по умолчанию: 10 сек.

A.1.1.8. Перевод части соединений в режим онлайн

При запуске вычислений также активируется онлайн-компилятор.

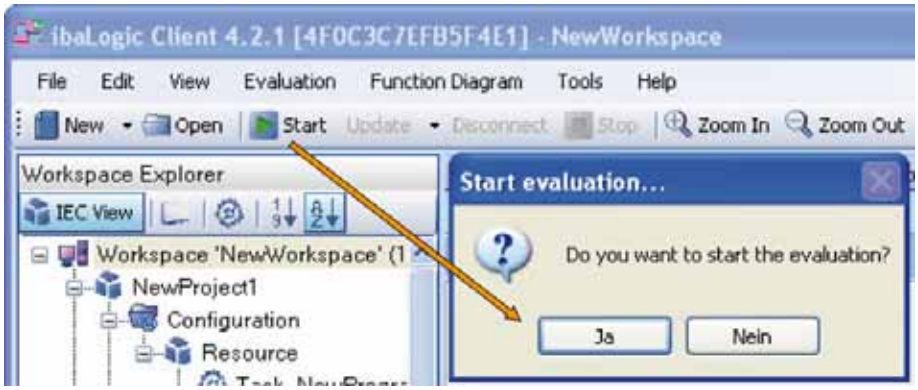
- Цвет фона области программирования становится розовым.
- Цвета всех бинарных соединительных линий зависят от их состояния.
- Коннекторы блоков отображаются и обновляются постоянно.
- Теперь все соединения работают "в режиме реального времени".

Фигурально выражаясь, это можно сравнить с подачей напряжения на испытательную схему.

На практике оборудованием управляет ibaLogic посредством подключенных выходов. В этом случае возникновение программной ошибки в реальной ситуации может серьезно сказаться на процессе производства.

Последовательность действий

1. Щелкните по кнопке <Запуск> (<Start>) в панели инструментов, чтобы запустить вычисление программы-примера. Подтвердите запуск вычислений щелчком по кнопке <Да>. Вышеописанные онлайн-свойства программы будут активированы.



A.1.1.9. Тестирование переключателя и блока выбора

Поскольку блок выбора уже соединен и схема работает в режиме онлайн, его функции можно протестировать независимо от состояния переключателя входа (G).
Принцип функционирования блоков выбора описан в настоящем руководстве в разделе, посвященном стандартным блокам. Теперь у вас есть возможность применить эти знания на практике.

Последовательность действий

➲ Измените состояние переключателя, щелкнув по блоку SWITCH левой кнопкой мыши.

Как видно на рисунке, в выключенном состоянии (Off), выход реостата (значение = 39.09) соединен с выходом SEL (OUT).

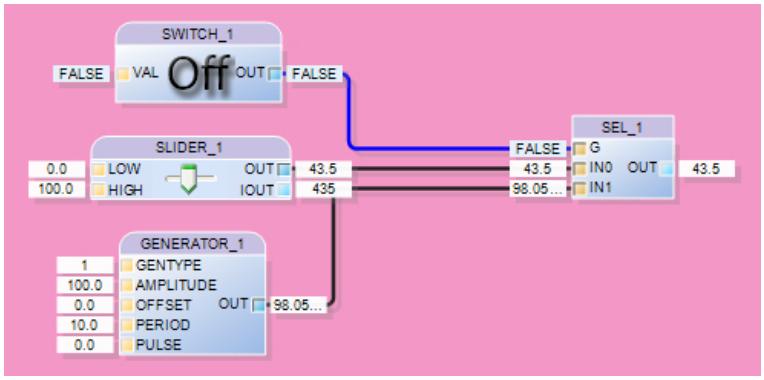


Рис. 138: Выход реостата (значение = 43.50) соединен с выходом SEL (OUT)

Во включенном состоянии (ON) выход генератора, значение которого периодически меняется, соединен с выходом SEL (OUT).

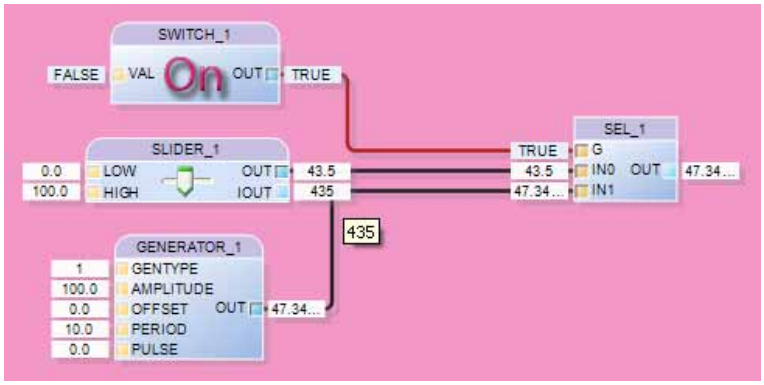


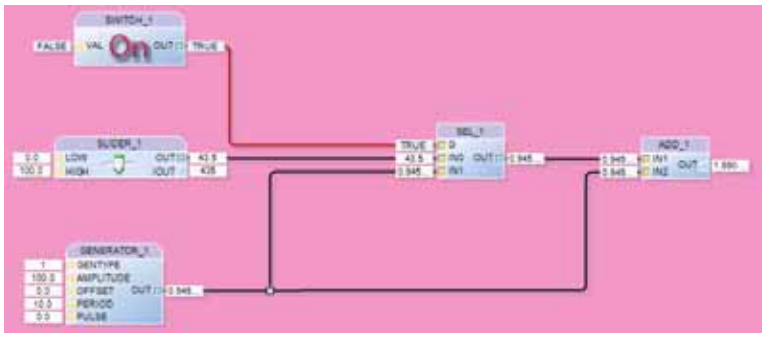
Рис. 139: Выход генератора соединен с выходом SEL (OUT)

A.1.1.10. Соединение сумматора

Соедините сумматор, чтобы завершить схему, как указано в упражнении.

Последовательность действий

1. Наведите курсор мыши на выход блока выбора (OUT) и, удерживая левую кнопку мыши нажатой, переведите курсор на вход (IN1) сумматора.



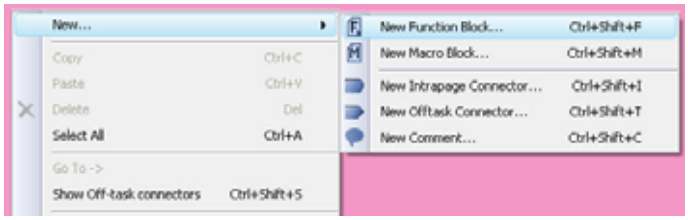
2. Наведите курсор мыши на вход сумматора (IN2) и, удерживая левую кнопку мыши нажатой, переведите курсор на выход генератора (OUT).
   Точка соединения над курсором автоматически прикрепляется, и формируется ветвь.

A.1.1.11. Создание OTC, чтобы проиллюстрировать результат

В нашем упражнении требуется также разместить и соединить коннектор вне задачи, чтобы проиллюстрировать результат этого простого примера. Этот OTC называется "Результат".

Последовательность действий

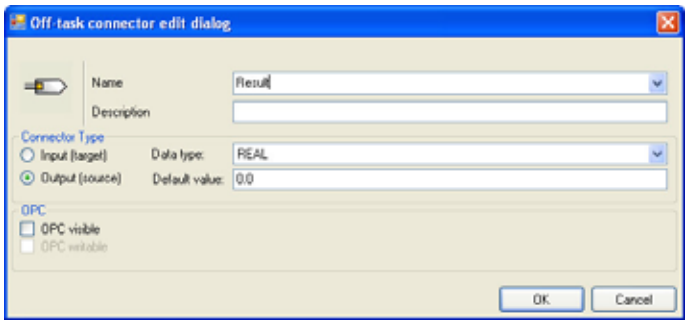
1. Щелкните правой кнопкой мыши по рабочей области в области программирования.
2. Выберите "Новый… -– Новый коннектор вне задачи" ("New… –- New Off-Task Connector").



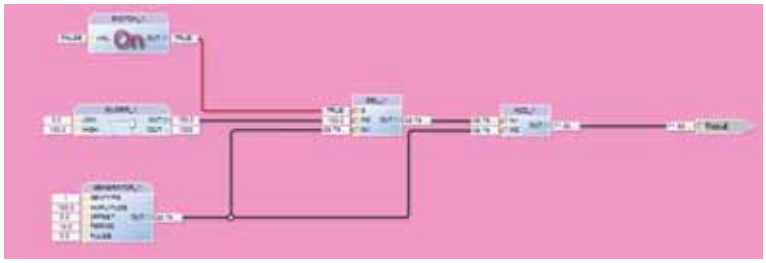
Появится диалоговое окно "Редактировать коннектор вне задачи" ("Off-task connector edit").

3. В поле ввода присвойте имя "Результат". Все остальные предварительно установленные значения изменять не нужно. Сконфигурируйте тип данных "REAL" и задайте значение по умолчанию 0.0.

   Выбор предустановленного значения для выхода является правильным, поскольку это значение поступает на OTC.
   Выбор OTC был бы необходим в том случае, если требовалось бы получать значение, передаваемое из другой программы по OTC.



4. Соедините OTC "Результат" с сумматором.



Комментарий

OTC является ключевым элементом в графическом программировании.

OTC помогает сделать проект более ясным, поскольку распространяется на несколько программ, которые обмениваются данными друг с другом посредством OTC.
В отличие от OTC, внутристраничные коннекторы (IPC) используются в рамках одной программы.

В настоящем руководстве также содержится раздел 12 "Правила программирования", в котором вы найете сведения о том, как использовать OTC и IPC, чтобы избежать запутанных соединений при программировании.

Как коммуникационный элемент OTC имеет большое значение для соединения с контролирующей системой HMI (Human-Machine Interface).

A.1.1.12. Анализ схемы

Вы можете контролировать результат с помощью динамического отображения коннекторов. Однако, даже в этой простой программе, работающей с циклом 50 мс, проверка числовых результатов является трудной задачей.

Обратите внимание на вторую часть упражнения, которая следует ниже.

A.1.2. Упражнение, часть 2

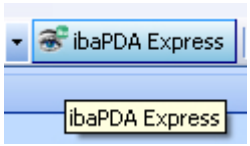
Целью этого упражнения является продемонстрировать удобство онлайн-отображения сигналов в виде кривой, которое позволяет оценить результат.

A.1.2.1. Анализ программы с помощью ibaPDA Express.

Для онлайн-отображения кривых в ibaLogic интегрированы функции отображения ibaPDA.

Последовательность действий

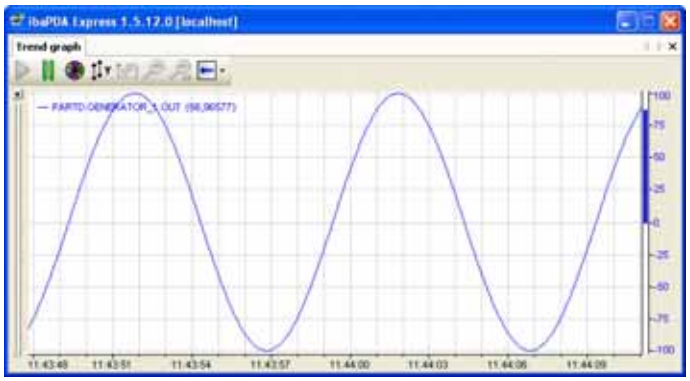
1. Щелкните кнопку ibaPDA Express в панели инструментов.
Эта программа бесплатно интегрирована во все версии ibaLogic.



2. Наведите курсор на коннектор OUT генератора и перетащите его, нажав и удерживая клавишу <ALT>, в окно ibaPDA Express.

3. Вы увидите выбранное промежуточное значение (Generator OUT) в виде кривой в области отображения.

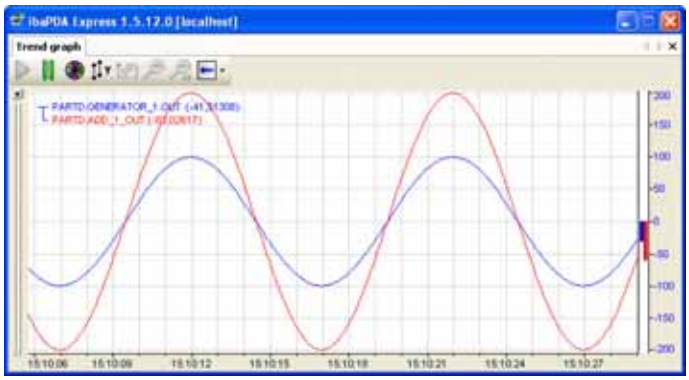
4. Настройте ось Y с помощью кнопки <Автомасштабировать все>.



5. Перетащите интересующие вас значения и результаты в окно ibaPDA Express.

- Если эти переменные имеют одинаковый масштаб, то они помещаются к уже отображаемому сигналу.
- Если масштаб разный, то при перетаскивании коннекторов в окно кривой появится новая шкала.
- Для того чтобы открыть новое окно кривой, переместите коннектор в область за пределами текущего окна кривой.

6. Выполните окончательную проверку упражнения, меняя положение переключателя вкл / выкл (on / off). Пронаблюдайте за изменениями кривой.



A.1.3. Упражнение, часть 3

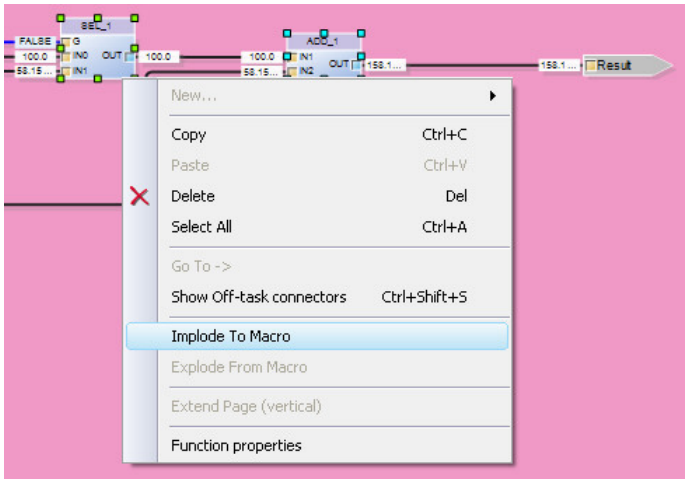
Улучшение ясности и читабельности программы

Для того чтобы улучшить ясность и читабельность схемы-примера, можно объединять функциональные блоки в макросы.

Этот способ можно проиллюстрировать с помощью этого маленького упражнения.

Последовательность действий

1. Нажав и удерживая клавишу <Ctrl>, выделите нужные функциональные блоки и соответствующие соединительные линии. В данном случае требуется выделить блок выбора и сумматор, а также линию, которая их соединяет.



2. Удерживая клавишу <Ctrl>, нажмите правую кнопку мыши, чтобы появилось меню макросов.
3. В этом меню выберите пункт "Объединить в макроблок" ("Implode To Macro").
4. Подтвердите промежуточную схему (макрос, входы, выходы).

Комментарий

Макрос создан. Ему автоматически присваивается имя (IPL_MB_1), и он выполняет те же функции, что и отдельные функциональные блоки, из которых он образован. Однако, пользователь может переименовать макрос для создания значимой и должным образом оформленной документации.

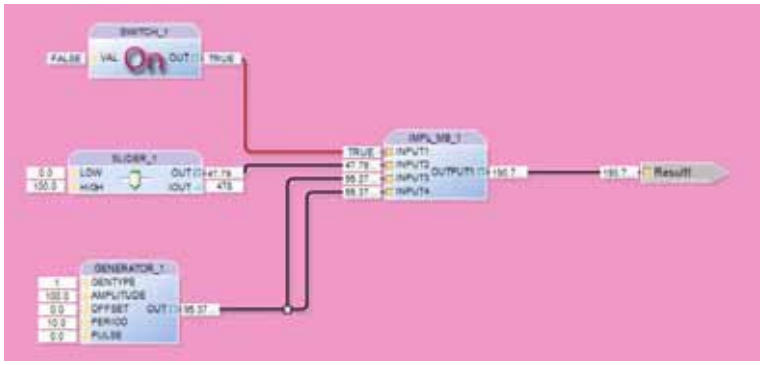
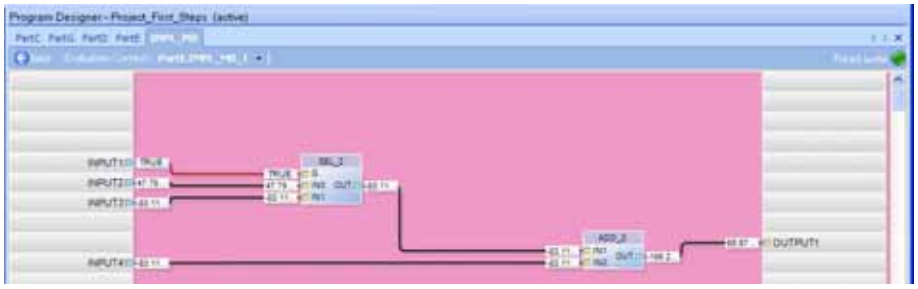


Рис. 140: Создание макроса

Совет

Для объединения большого количества функциональных блоков в один макрос можно также использовать мышь. Нажав и удерживая левую кнопку мыши, очертите прямоугольник вокруг нужных объектов.

5. Чтобы просмотреть содержимое макроса, щелкните по нему двойным щелчком. Некоторые из линий или блоков размещаются несколько беспорядочным образом. Макросы также можно редактировать, например добавлять соединительные линии или перераспределять уже существующие.



6. Макросы также можно редактировать, например добавлять соединительные линии.

A.1.4. Упражнение, часть 4

Создание функциональных блоков с помощью структурированного текста (ST)

Библиотека функциональных блоков содержит стандартные блоки для самых различных целей. На основе собственного опыта, а также, что немаловажно, на основе предложений пользователей ibaLogic, специалисты iba разработали и добавили в библиотеку некоторые дополнительные полезные функциональные блоки.

Для того чтобы продемонстрировать пользователям возможности и способы самостоятельного создания специальных блоков, в конце этого упражнения содержится пример программирования одного функционального блока. Для тестирования работы нового блока, он используется параллельно с созданным ранее макросом. Результаты работы блоков должны быть равнозначны.

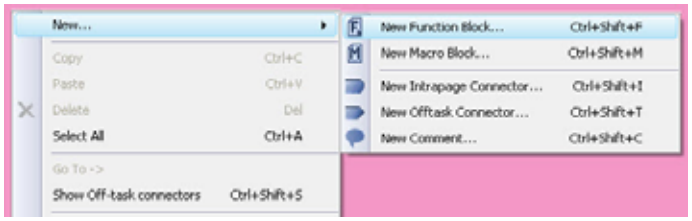
В языке программирования "структурированный текст" особое внимание уделяется соблюдению стандарта IEC 1131-3. Этот высокоуровневый язык программирования позволяет программировать четкие и легко читаемые функциональные блоки, наподобие того как это выполняется в языке Pascal. Блоки, созданные посредством ST, можно портировать в сторонние системы.

Если у вас есть опыт работы с классическими контроллерами, это значит, что вы уже знаете, как использовать стандартные функциональные блоки.

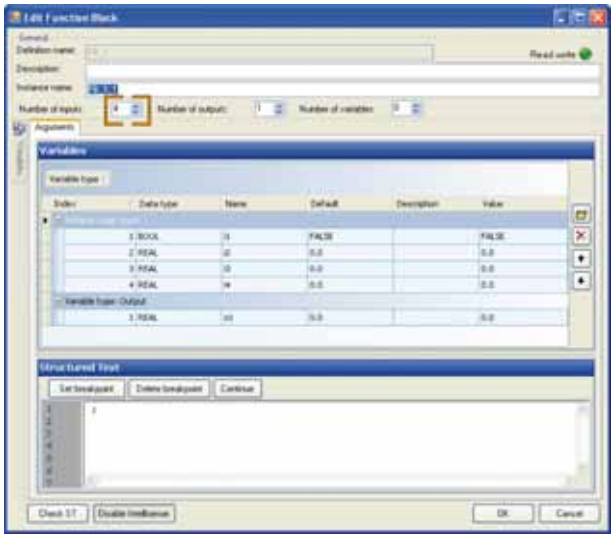
С другой стороны, если вы начали работать с контроллерами, имея опыт программирования на Assembler или на высокоуровневом языке программирования, то вам, возможно, будет проще использовать структурированный текст.

Последовательность действий

1. В контекстном меню выберите "Новый… – Новый функциональный блок…" ("New… – New Function Block…").
   Появится диалоговое окно "Новый функциональный блок".



2. Установите "Количество входов" на 4. Поскольку необходимо "клонировать" макрос из упражнения 3, функциональный блок требует такое же количество входов и выходов (4 входа и 1 выход).



3. Сконфигурируйте типы данных для входов и выхода.

Для входа

- 1 x BOOL
- 3 x REAL

Для выхода

- 1 x REAL

Для создания новой переменной используйте кнопку "  ".

4. В поле "Структурированный текст" ("Structured Text") поставьте точку с запятой.

Комментарий

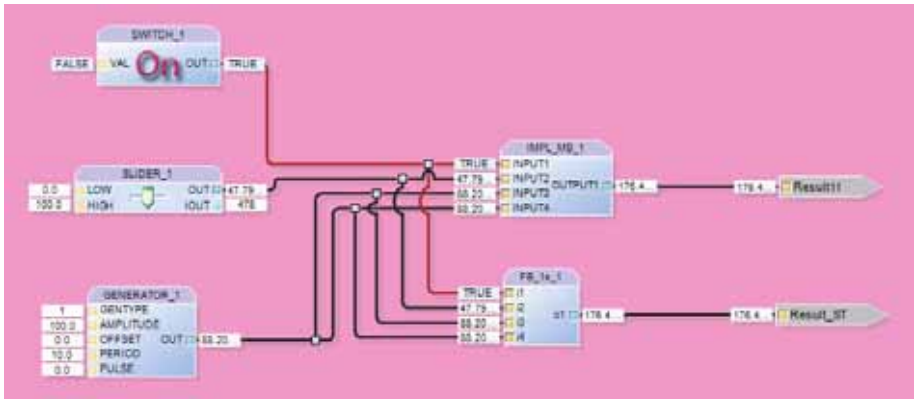
Сначала создается только пустой функциональный блок.

В поле ST должна стоять хотя бы одна точка с запятой (это нужно с формальной точки зрения).

Поскольку входы и выходы уже определены, новый блок может быть добавлен в схему и подключен параллельно с уже существующим макросом.

5. Разместите новый функциональный блок на схеме.

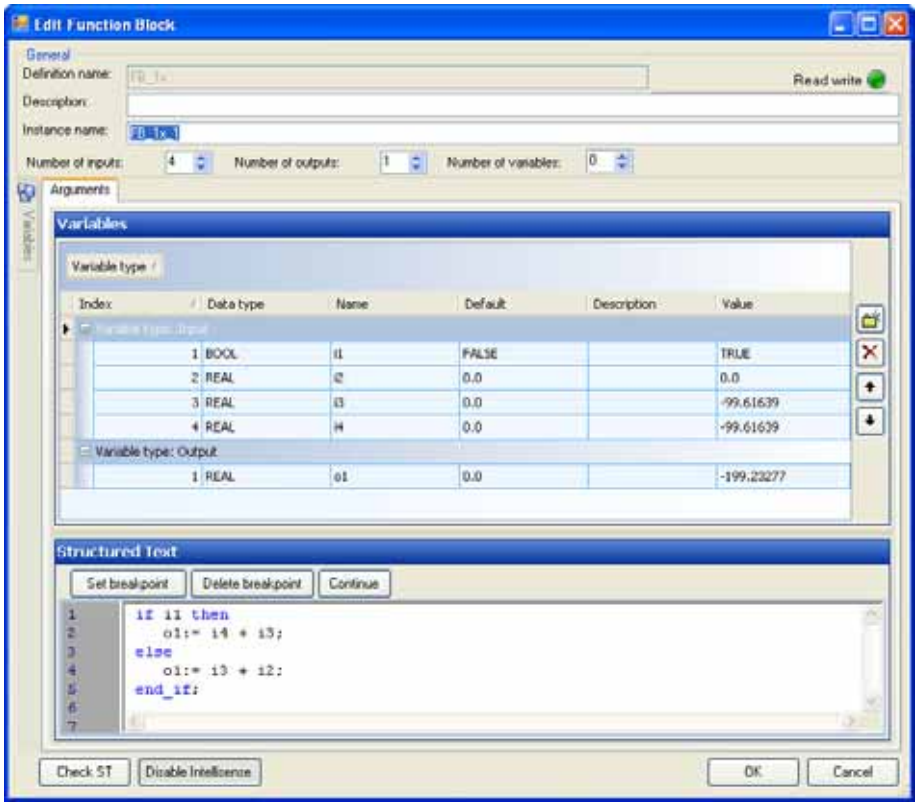
6. Соедините новый блок параллельно с макросом. Для того чтобы создать второй ОТС с результатом, скопируйте (<Ctrl+C>) и вставьте (<Ctrl+V>) уже существующий. Присвойте новому ОТС имя, например "Result_ST".



7. Функциональный блок пока не реагирует на входы. Это еще предстоит запрограммировать.

8. Выделите новый блок двойным щелчком. Появится окно редактирования функционального блока ("Edit Function Block").
Теперь можно выполнить программирование блока с помощью стандартных выражений "if", "then", "else" и "end_if", которые являются частью почти всех высокоуровневых языков программирования.

9. В поле "Структурированный текст" ("Structured Text") введите следующий исходный код.



Результат

Пример начинается с запроса состояния переключателя (if i1 = "TRUE" или короче: if i1), суммируются блоки i4 + i3 или i3 + i2 соответственно.

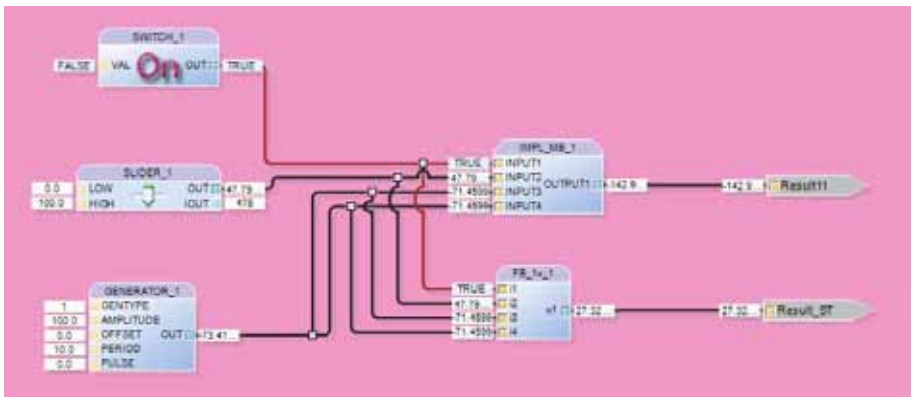


Рис. 141: Схема-пример с запросом суммирования

Комментарий

Обратите внимание на то, что все переменные блока обновляются онлайн!

10. Перетащите результат ST-блока в область отображения ibaPDA Express на ту же ось X, где находится результат макроса.

Красная кривая совпадает с синей (результат макроса). В ibaPDA Express отображается только вторая кривая (синяя). Однако, исходя из данных в правой части области отображения, мы можем сделать вывод о том, что она содержит два отдельных значения.

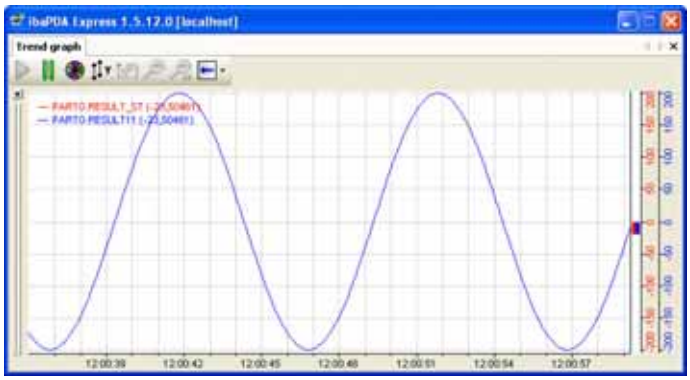


Рис. 142: Равнозначность результатов

A.2. DAT_FILE_WRITE - пример проекта

A.2.1. DAT_FILE_WRITE в режиме без буферизации

В режиме без буферизации (объяснение приводится в разделе 9.4.2, ""Режим буферизации" FOB-карт") каждый цикл сохранения сохраняется одно значение.

Этот режим используется в том случае, если цикл дискретизации данных для сохранения совпадает с циклом задачи ibaLogic, или если вы хотите, чтобы сохранялись данные, генерируемые собственно ibaLogic.

Задача: сохранить 8 аналоговых и 8 цифровых значений, полученных от ibaLogic

Предварительно установленные параметры

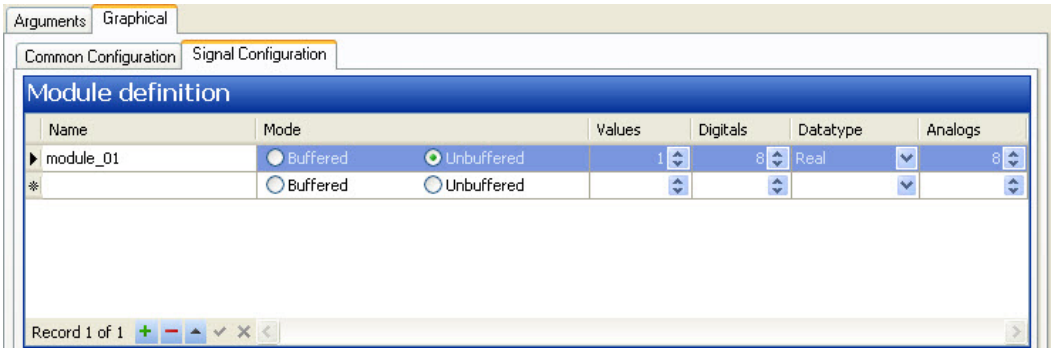
- Платформа: Win XP
- Интервал задачи: 20 мс

Шаг 1: конфигурирование блока DFW

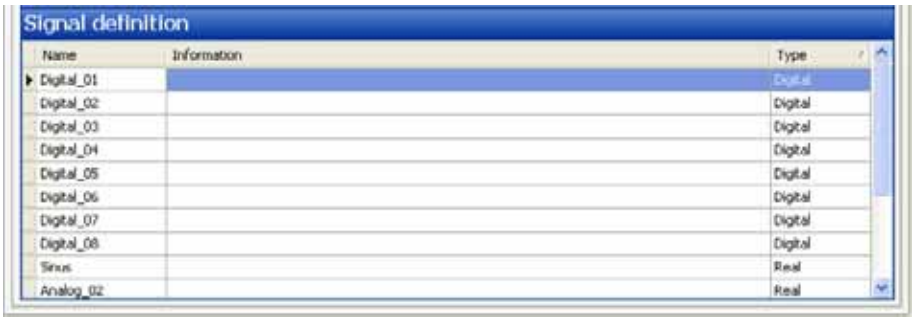
1. Общая конфигурация

- Асинхр. доступ: отключен
- Цикл сохранения: 10 (не имеет значения)
- Смещение начального времени: 0
- Сохранение значений: **Активно**
- Запись в файл: Отключена (управляется внешним источником)
- Последующая обработка: Отключена
- Отметить файл: **Активно**
- Техностока: Пусто
- Информация о файле: Пусто
- Папка: С помощью кнопки просмотра выберите папку на локальном диске, например: "d:\dat\ibaLogic\".
- Имя файла: Не изменяйте значение по умолчанию.
- Период дискретизации: введите интервал задачи: "0,02" с.

2. Конфигурация сигналов
   - Имя: "Module_01"
   - Режим: Без буферизации ("Unbuffered")
   - Значения: 1 (не имеет значения)
   - Цифровые значения: 8
   - Тип данных: REAL
   - Аналоговые значения: 8



   - Щелкните кнопкой мыши по области определения сигнала. Будет создан модуль с 8 цифровыми значениями и 8 значениями real. Имена сигналам будут присвоены автоматически: "Цифровой_xx" и "Аналоговый_xx". Пользователь может изменить имя сигнала по своему усмотрению, например назвать его "Синус".



Шаг 2: соединение DFW

1. Закройте редактор модулей.
2. Соедините коннектор "STORE_FILE" с выходом блока "SWITCH".

3. Соедините первый измеренный сигнал, например выход генератора синусоидального сигнала (тип данных REAL), со входным коннектором "DATA".
   - Откроется меню выбора, в котором можно выбрать модуль для соединения с измеренным сигналом.
   - При наведении курсора мыши на модуль появляется другое меню выбора, в котором можно выбрать нужный сигнал. (например, Синус).
   - Откроется еще одно меню выбора.
     Выберите опцию меню: "data: REAL".

Рис. 143: Назначение измеренных сигналов

Примечание

Не обращайте внимание на имя структуры данных, которое присваивается автоматически. Это имя можно игнорировать (при условии, что вы не будете пытаться изменить код в ST - см. ниже).

Результат

На основе этого ibaLogic создает элемент объединения, посредством которого выполняется адресация к выбранному элементу из внутренней структуры данных.

Подробное описание элементов объединения и разделения содержится в разделе 6.9 "Преобразователи, элементы разделения и объединения".

Создаются следующие элементы объединения:

- ❑ "Элемент объединения модулей" ("Module Joiner")
- ❑ "Элемент объединения сигналов" ("Signal Joiner")
- ❑ "Элемент объединения свойств сигналов" ("Signal Property Joiner")

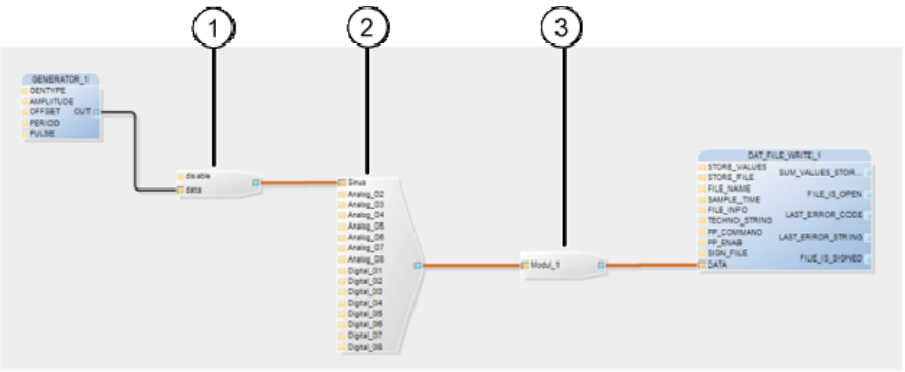


Рис. 144: Элемент объединения

1 Элемент объединения свойств сигналов

2 Элемент объединения сигналов

3 Элемент объединения модулей

Шаг 3: создание дополнительных сигналов для измерения

Вы можете соединить другие сигналы непосредственно с помощью элемента объединения сигналов.

Примечание

Для того чтобы временно отключить запись отдельных сигналов, не следует использовать опцию "Деактивировать" для входов "Элемента объединения свойств сигналов". Это приведет к ошибкам при записи данных.

Поскольку отдельные сигналы не имеют штампа времени, кривая тренда всегда отображается непрерывно. Пустой промежуток образуется в конце .dat-файла.

Шаг 4: запуск записи

1. Запустите PMAC.
2. Установите SWITCH в DAT_FILE_WRITE.STORE_FILE на "Вкл" ("On").

Результат

О том, что запись началась, свидетельствует увеличивающееся значение на интерфейсе "DAT_FILE_WRITE_1.SUM_VALUES_STORED". Для проверки результата откройте созданный .dat-файл в программе "ibaAnalyzer".

Альтернативный способ: программирование элемента объединения в ST

Для того чтобы схема программы была четкой, ясной и структурированной, вы можете запрограммировать распределение измеренных сигналов в коннекторе DATA функционального блока ST.

Пример, который аналогичен вышеприведенной конфигурации:

1. Наведите курсор мыши на коннектор DATA модуля DFW. Появится подсказка с информацией о сгенерированном программой структурном типе данных, например "DFW_634094511415156250_Data". Сохраните эту информацию!
2. Создайте функциональный блок, содержащий одну или несколько входных переменных типа REAL и одну выходную переменную того же типа, что коннектор DATA модуля DFW.
   - Активируйте "Intellisense".
   - Можно приступать к присвоению сигналов: введите "o1.".
   - После того как вы введете точку, в ibaLogic отобразит структурный элемент (здесь: "module_01"), поскольку "o1" является структурным типом данных. Выберите / отмените выбор предлагаемых элементов; чтобы принять изменения, нажмите "Tab" и поставьте точку в конце.
   - Поскольку даже "module_01" - это структура, ibaLogic отображает его структурные элементы, здесь: все цифровые и аналоговые сигналы. Выберите "Синус" ("Sine") и поставьте в конце точку.
   - Поскольку "Синус" - это структурный тип данных, ibaLogic отображает его структурные элементы: "данные" ("data") и "деактивировать" ("disable"). Выберите "данные" и введите " := i1;".

Результат

Вы запрограммировали первый сигнал. Теперь вы можете ввести аналогичным образом остальные сигналы. Результат должен выглядеть так:

```
1 o1.module_01.Sine.data := i1;
2 o1.module_01.Analog_02.data := i2;
3
```

Вы можете соединить только входы с соответствующими измеренными сигналами и выход с коннектором DATA модуля DFW.

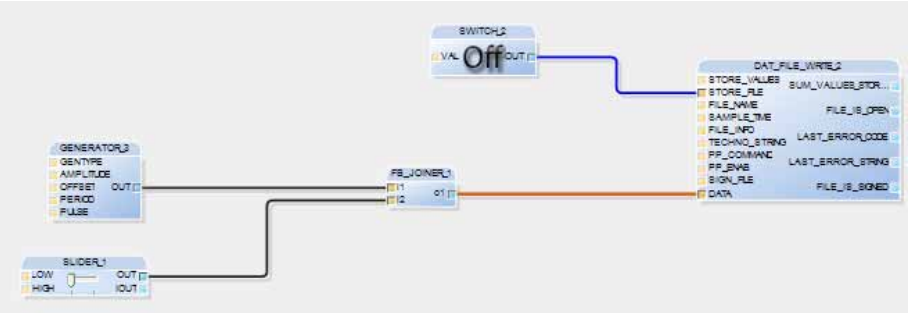


Рис. 145: Схема-пример "Присвоение измеренных сигналов"

Примечание

Если впоследствии вы захотите изменить конфигурацию сигналов DFW, нужно будет удалить соединение коннектора DATA и после этого изменения присвоить новый тип данных выходному коннектору FB_JOINER. Возможно придется также изменить распределение сигналов в ST.

A.2.2. DAT_FILE_WRITE в режиме буферизации

В режиме буферизации (объяснение приводится в разделе 9.4.2, ""Режим буферизации" FOB-карт") за каждый цикл сохранения программа сохраняет массив значений. Сигналы для сохранения должны быть доступны как массивы. Вследствие этого некоторые параметры модуля DFW будут иметь несколько иное значение.

Этот режим используется в том случае, если цикл дискретизации данных для сохранения короче самого короткого цикла задачи ibaLogic.

Задача: сохранить 8 аналоговых значений от ibaPADU8.

Предварительно установленные параметры

- Частота дискретизации ibaPADU8: 1 мс
- ibaPADU8 - ibaFOB-Link0, режим буферизации, целые числа
- Опорное время прерываний: 1 мс
- Платформа Win XP
- Интервал задачи: 20 мс

Шаг 1: конфигурирование входов с буферизацией

1. Общая конфигурация:
Следующие аппаратные сигналы доступны для конфигурации буферизации / передачи данных (см. описание в разделе 9.4.2, ""Режим буферизации" FOB-карт"):

- Выходные сигналы
"…DataSize", "…Ratio", "…RequestBuffer"
- Входные сигналы
"…CurDataSize", "…FillCount"

2. При выборе параметров учитывайте следующее:

- Глубина массива в DFW должна быть больше или равна "DataSize".
- Период дискретизации (в DFW) должен совпадать с (Глубина массива * Период дискретизации).
- Сигнал "DataSize" должен быть выбран с соблюдением следующего условия:

  Интервал задачи ≤ период дискретизации * DataSize / Ratio / 3

  (Коэффициент "3" используется для того, чтобы ни одно значение не было "потеряно", даже в случае, если задача будет приостановлена.)
- Для примера выберите DataSize = 100,
соответственно, должно быть следующее: "Интервал задачи ≤ 1мс * 100 / 1 / 3", т.е. интервал задачи должен составлять менее 33 мс. Выберите 20 мс.

3. Создайте пользовательский блок, в котором вы сможете задать выходные сигналы для режима буферизации, например:

```
if (iEnable = TRUE) then
   if (iEnable <> v1) then
 oSize := iSize;
 oRatio := 1;
 oTime := iTime;
 oTakeover := TRUE;
   else
 oTakeover := FALSE;
   end_if;
   oRequest := TRUE;
else
   oRequest := FALSE;
end_if;

v1 := iEnable;
```

4. Соедините блок с выходными сигналами.

(Коннекторы iTime, oTime и oTakeover в данном случае не требуются.)

Шаг 2: установите параметры модуля DFW, "Общая конфигурация"

1. Перейдите в режим офлайн.
2. Общая конфигурация
   - Асинхр. доступ: отключен
   - Цикл сохранения: 10 (не имеет значения)
   - Смещение начального времени: 0
   - Сохранение значений: **активно**
   - Запись в файл: отключена (управляется внешним источником)
   - Последующая обработка: отключена
   - Отметить файл: **активно**
   - Техностока: пусто
   - Информация по файлу: пусто
   - Папка: С помощью кнопки просмотра выберите папку на локальном диске, например: "d:\dat\ibaLogic\"
   - Имя файла: Не изменяйте значение по умолчанию.
   - Период дискретизации: Укажите интервал сохранения. Интервал сохранения = глубина массива * период дискретизации Выберите глубину массива 200, и на основе этого время сохранения будет равняться 0,2 сек.

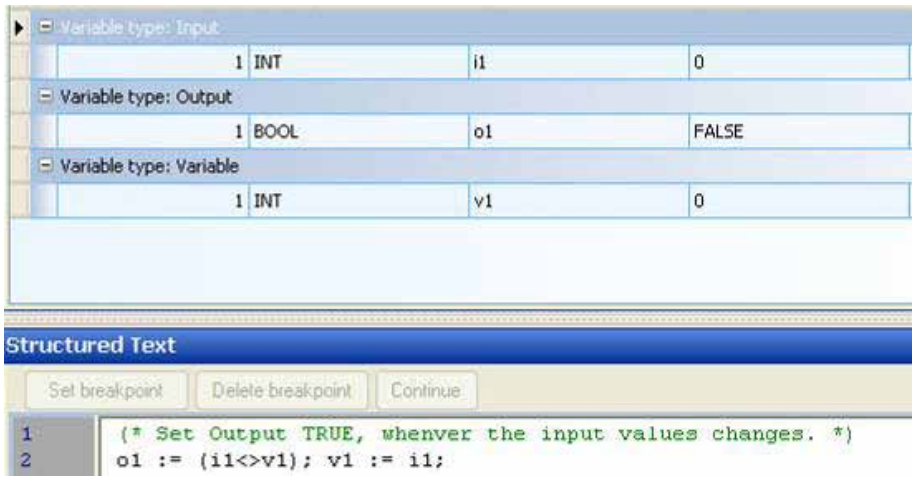
3. Конфигурация сигналов
   - Имя: "Module_01"
   - Режим: "Буферизация"
   - Значения: 200
   - Цифровые значения: 8
   - Тип данных: Integer
   - Аналоговые значения: 8
   - В столбце "Значения" ("Values") укажите размер массива 200. В этом случае время сохранения составляет 200 мс (см. выше).

4. Перейдите в режим онлайн.

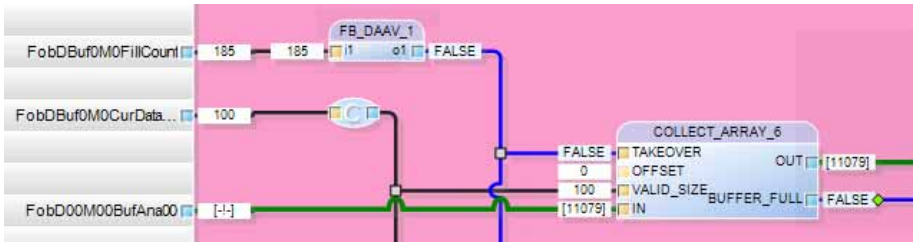
Шаг 3: принять входные сигналы с буферизацией

1. Перетащите блок "COLLECT_ARRAY" из папки функционального блока "Преобразование типа" ("Type Conversion") в окно рабочей области.
   Этот блок используется для преобразования входного массива (тип данных FOBFBUF_INT) в массив для DFW.

2. Создайте пользовательский функциональный блок (FB_DAAV) с нижеследующими свойствами для генерирования сигнала "TAKEOVER" для COLLECT_ARRAY:



3. Соедините блоки, как показано на скриншоте ниже.

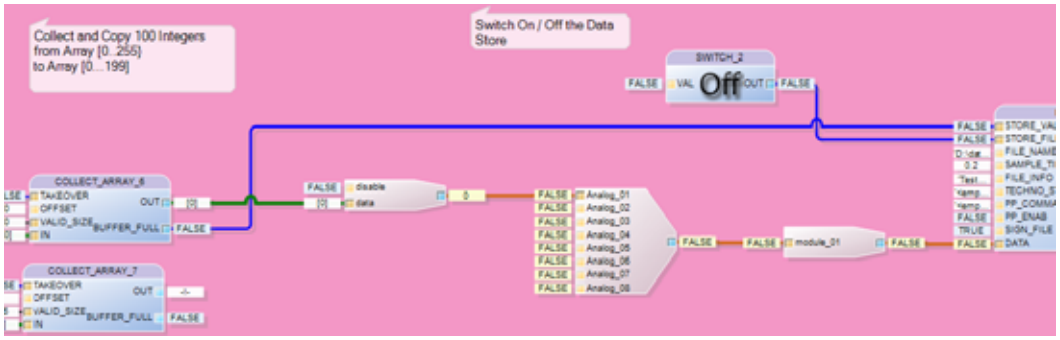


Шаг 4: передача данных в DAT_FILE_WRITE

1. Создайте блок SWITCH и соедините его выход с DAT_FILE_WRITE.STORE_FILES.
2. Соедините COLLECT_ARRAY.BUFFER_FULL с DAT_FILE_WRITE.STORE_VALUES.
3. Соедините выход COLLECT_ARRAY.OUT с DAT_FILE_WRITE.DATA.
   В ibaLogic появится меню выбора для блока объединения. В меню выберите "module_01 – Analog_01 -- … AnaArr0".

Результат

Данные передаются в DAT_FILE_WRITE.

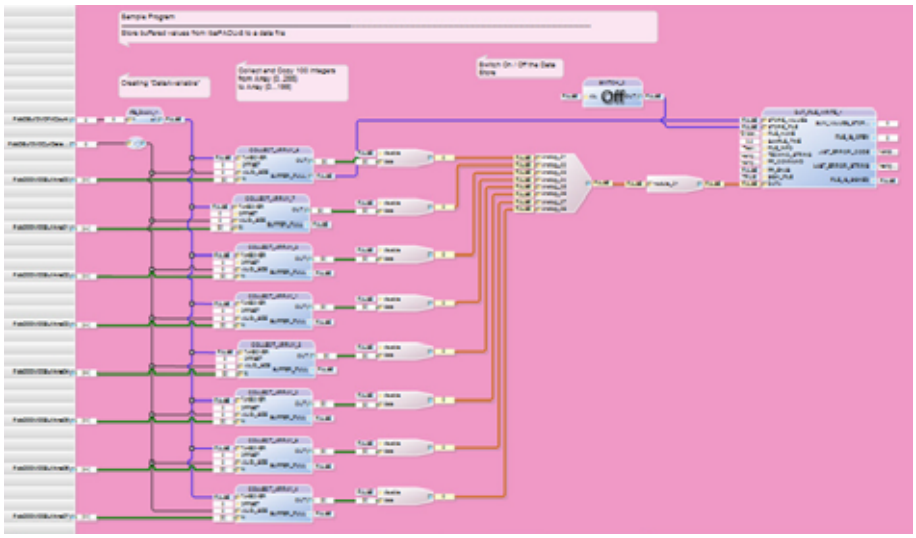


Шаг 5: подключение (соединение) остальных входов

1. Скопируйте один блок COLLECT_ARRAY для каждого аналогового входа.
2. Соедините все блоки COLLECT_ARRAY.TAKEOVER с сигналом DataAvailabe (FB_DAAV.o1).
3. Соедините все блоки COLLECT_ARRAY.VALID_SIZE с выходом преобразователя для " ... CurDataSize".
4. Соедините каждый COLLECT_ARRAY.IN с соответствующим буферным аналоговым входом " … BufAna*nn*"
5. Соедините каждый блок COLLECT_ARRAY.OUT с соответствующим коннектором "Analog_*nn*" блока объединения.

Шаг 6: запуск записи

1. Установите SWITCH в DAT_FILE_WRITE.STORE_FILE на "Вкл" ("On").



Результат

О том, что запись началась, свидетельствует увеличивающееся значение на интерфейсе "DAT_FILE_WRITE_1.SUM_VALUES_STORED". Проверьте результат, открыв .dat-файл, создаваемый "ibaAnalyzer".

Советы

1. Параллельно со входами с буферизацией можно использовать входы без буферизации, например для визуализации аналоговых сигналов с помощью ibaPDA Express.
2. Проверка .dat-файла: если шкала времени не совпадает с действительно записанным временем, это значит, что либо период дискретизации был сконфигурирован некорректно, либо интервал задачи, размер данных (DataSize) и глубина массива несовместимы друг с другом.

Документация

Вышеприведенный пример содержится на CD, который входит в объем поставки.

B. Правила присвоения имен

Имя представляет собой последовательность, которая может состоять из букв, цифр и подчеркиваний. Для присвоения имен действуют следующие правила:

- Можно использовать как заглавные, так и строчные буквы, например: ABCD и abcd.
- Имя должно начинаться с буквы или нижнего подчеркивания. Имя не может начинаться с цифры.
- Нижнее подчеркивание - значимый символ в имени, например: A_BCD и AB_CD - это разные имена (в числовых константах этот символ значимым не является).
- Использование нижнего подчеркивания в конце имени недопустимо, например ABCD_.
- Несколько нижних подчеркиваний подряд использовать нельзя, например AB__CD.
- Нельзя использовать ключевые слова, например for и if.

Особенности ibaLogic:

Нельзя использовать имя, состоящее только из одной буквы.

Примечание

Эти правила относятся к ibaLogic, причем не только к именам функциональных блоков, но и к именам OTC, IPC, входных и выходных сигналов и т.д.

C. Типы данных

C.1. Стандартные типы данных

ibaLogic поддерживает следующие элементарные типы данных:

Тип	Диапазон (мин.)	Диапазон (макс.)	Объяснение
BOOL	0 (FALSE)	1 (TRUE)	
BYTE	16#00	16#FF	8 бит
WORD	16#0000	16#FFFF	16 бит
DWORD	16#00000000	16#FFFFFFFF	32-битное слово
SINT	-128	127	8-битное знаковое целое
USINT	0	255	8-битное беззнаковое целое
INT	-32768	32767	16-битное знаковое целое
UINT	0	65535	16-битное беззнаковое целое
DINT	-2147483648	2147483647	32-битное знаковое целое
UDINT	0	4294967295	32-битное беззнаковое целое
REAL	1.175494351 e-38	3.402823466 e+38	Плавающая точка, обычная точность, 32 бита
LREAL	2.225073858 … e-308	1.797693134862 … e+308	Плавающая точка, двойная точность, 64 бита
TIME	-2147483648мс -24д_20ч_31м_23s_648мс	2147483648мс 24д_20ч_31м_23s_648мс	Время, внутренне отображенное на DINT с разрешением 1 мс на единицу
STRING	0	250 символов (249 пользовательских, по причине наличия символа NULL в конце)	Строка с несколькими символами, включая символ конца строки (NULL).

C.2. Производные типы данных

Тип	Объяснение
DIRECT_DERIVED	Элементарные типы данных с фиксированным значением (константы)
SUBRANGE	Целочисленные (integer) типы данных с ограниченным диапазоном значений
STRING_DERIVED	Строка, которая имеет фиксированное значение и длину

C.3. Родовые типы данных

Тип	Объяснение
ENUM	Перечисляемый тип, вместо значений определяются имена.
ARRAY	Структура, состоящая из одного из вышеупомянутых типов данных (за исключением строки, которая уже представляет собой массив); макс. 4 измерения. Максимальное количество элементов: 32767
STRUCT	Структура, состоящая из любой последовательности вышеупомянутых типов данных. Максимальное количество элементов: 1 048 576

Важно

Память ограничена внутренним размером 63 Кб для каждого уровня. Следовательно, например, объем памяти всех входных и выходных переменных функционального блока не должен превышать указанный размер.

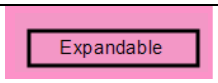
Более подробная информация содержится в пункте 11.4 "Ограничения производительности".

D. Стандартные функциональные блоки

В приложении содержится таблица с обзором всех функций и функциональных блоков, которые доступны в ibaLogic V4.

Интерпретация таблицы

В этом разделе приводятся советы и инструкции по интерпретации обзорных таблиц.

Столбец	Объяснение
Типы входных данных	В столбцах типов входных данных перечисляются типы данных, допустимые для каждого коннектора. Существуют коннекторы блоков, для которых тип данных изначально не задается, а определяется только при установлении соединения с другим коннектором.
Строение блока	
Зел.	Функции блоков определены в соответствии со стандартом IEC 1131-3.
Желт.	Функции блоков были определены специалистами iba AG.
	Этот блок является расширяемым, т.е. количество входов можно изменить. Откройте блок двойным щелчком левой кнопкой мыши и измените "Количество входов" ("Number of inputs").
Тип выходных данных	В столбцах типов выходных данных перечисляются типы данных, допустимые для каждого коннектора. Существуют коннекторы блоков, для которых тип данных изначально не задается, а определяется только при установлении соединения с другим коннектором.
Объяснение, пример, синтаксис ST	Для каждой функции есть примечание касательно того, возможен ли (если да - то каким образом) ее вызов в ST. Вы также можете клонировать функции как несколько строк ST-кода.Однако эти строки не будут исполняться.

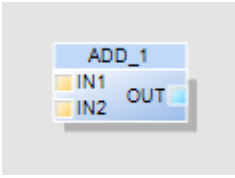
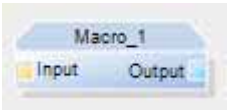
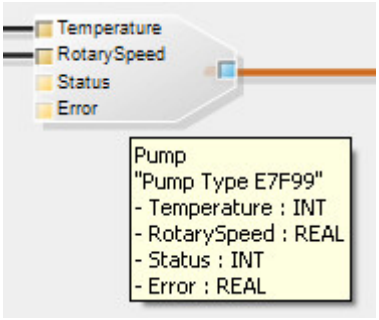
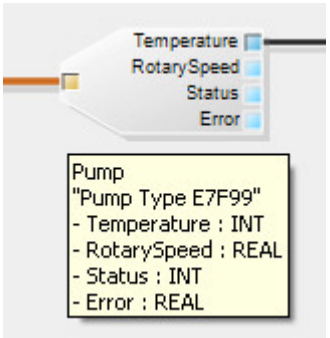
Типы данных

Для коннекторов, которым не присвоены типы данных, отображается тип данных "ANY" (любой) со следующими вариантами:

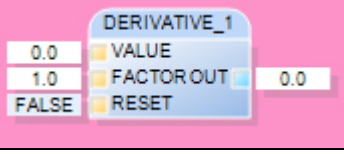
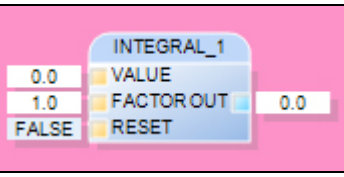
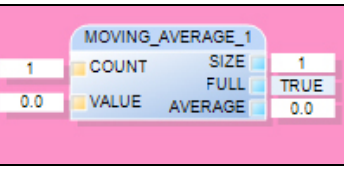
Тип данных "ANY" с "нетиповыми" коннекторами	Объяснение
Any_Int	Все целочисленные типы (SINT, INT, DINT, USINT, UINT и UDINT)
Any_Real	Все типы real (REAL, LREAL)
Any_Num	Все числовые типы данных (все real и целочисленные значения)
Any_Magnitude	Все числовые типы данных и тип TIME
Any_Bit	Все бит-ориентированные типы (BOOL, BYTE, WORD и DWORD)
Any_String	Строковый тип данных
Any_Elementary	Все элементарные типы (целые числа, значения real, TIME и STRING)
Any_Derived	Все элементарные типы данных, массивы и структуры
Any	Любой тип данных

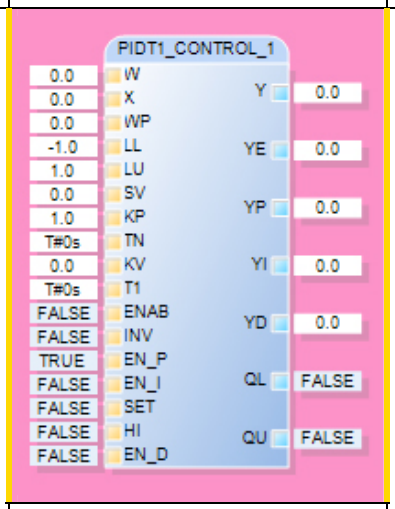
Тип блока с отображением функциональной диаграммы

Функции, функциональные блоки и макросы отображаются в области программирования следующим образом:

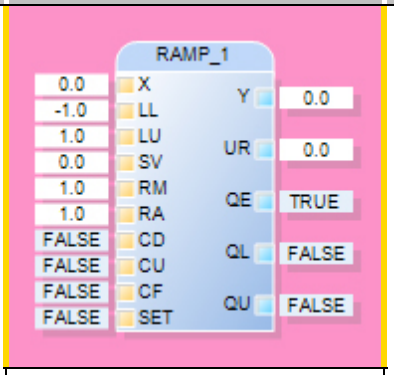
Тип блока с отображением функциональной диаграммы	Объяснение
Функция 	Функция отображается в виде прямоугольника с острыми углами.
Функциональный блок 	Функциональный блок отображается в виде прямоугольника с закругленными углами.
Макрос 	Макрос отображается в виде прямоугольника со срезанными углами.
Автоматический преобразователь типа 	Преобразователь типа отображается в виде значка с буквой "C".
Автоматически создаваемый блок объединения структуры 	Автоматически создаваемый блок объединения структуры Этот блок создается автоматически при попытке соединить отдельный параметр со структурой.
Автоматически создаваемый блок разделения структуры 	Автоматически создаваемый блок разделения структуры Этот блок создается автоматически при попытке соединить отдельный параметр со структурой.

D.1. Аналитические функции

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Real Real Bool		Real	**ПРОИЗВОДНАЯ:** Производная значения на основе времени Выходное значение "OUT" является производной входного значения "VALUE", умноженного на коэффициент "FACTOR" измерения времени. Сброс выхода выполняется, если вход "RE-SET" = "TRUE". **ST:** вызов невозможен
Real Real Bool		Real	**ИНТЕГРАЛ:** Интеграция значения по времени Выходное значение "OUT" является интегралом входного значения "VALUE", умноженного на коэффициент "FACTOR" измерения времени. Сброс выхода выполняется, если вход "RE-SET" = "TRUE". **ST:** вызов невозможен
Dint Real		Dint Bool Real	**MOVING_AVERAGE:** Значение скользящей средней Входное значение "COUNT" определяет несколько значений, из которых вычисляется среднее значение "VALUE". Выходное значение "SIZE" соответствует количеству значений, которые используются для вычисления среднего значения. Если было достигнуто указанное количество значений, выход "FULL" принимает значение "TRUE". Выход "AVERAGE" возвращает суммарное среднее значение. Вычисление среднего значения выполняется постоянно. Количество значений для его вычисления можно изменить в любое время.

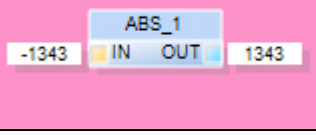
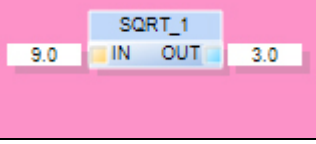
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
			ST: вызов невозможен
Lreal Lreal Lreal Lreal Lreal Lreal Lreal Time Lreal Time Bool Bool Bool Bool Bool Bool Bool		Lreal Lreal Lreal Lreal Lreal Bool Bool	**PIDT1_CONTROL:** Блок управления PIDT1 Универсальный PIDT1-регулятор, для которого можно настроить режим работы: P, I, PI или PIDT1-регулятор. Подробное описание содержится в разделе 5.3.3, "Комплексные функциональные блоки". **ST:** вызов невозможен
Lreal Time		Lreal	**PT1:** Элемент задержки времени 1-го порядка Сглаживающая постоянная времени T1 выполняет динамическую задержку входной переменной X, которая поступает на выход Y. Реализация: ti: = time_to_lreal(T1) / time_to_lreal(EvalDeltaTime); [7] Y: = 1.0 / (1.0 + t1) * (X + ti * Yold); Yold: = Y; **ST:** вызов невозможен

[7] EvaldeltaTime - это промежуток времени между двумя циклами обработки (вычисляется внутри).

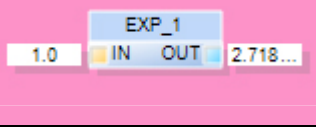
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Lreal Lreal Lreal Lreal Lreal Lreal Bool Bool Bool Bool		Lreal Lreal Bool Bool Bool	**RAMP:** Функциональный блок RAMP Блок с двумя разными пилообразными функциями, для ручного и автоматического режимов. Подробное описание содержится в разделе 5.3 "Комплексные функциональные блоки". **ST:** вызов невозможен

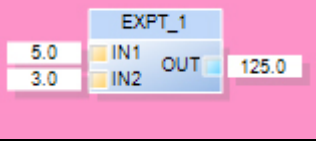
D.2. Арифметические функции

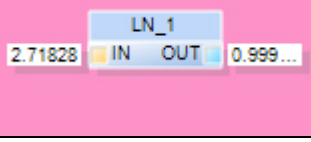
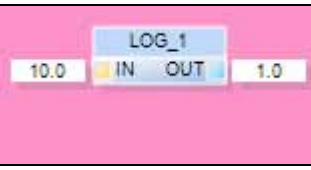
D.2.1. Общие

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_Num		Any_Num	**ABS:** абсолютное значение Пример: +1343 = abs(-1343); **ST:** `OUT := abs(IN);`
Any_Real		Any_Real	**SQRT:** квадратный корень Пример: +3.0 = sqrt(9.0); **ST:** `OUT:= sqrt(IN);`

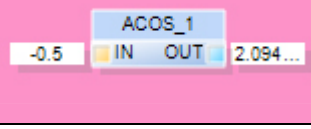
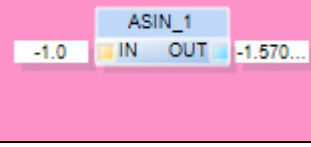
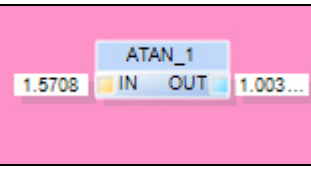
D.2.2. Логарифмические

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_Real		Any_Real	**EXP:** Натуральная экспонента с основанием e Результат: = e^{arg}; Примеры: 2.71828 = exp(1.0); 0.13533 = exp(-2.0); **ST:** `OUT:= exp(IN);`

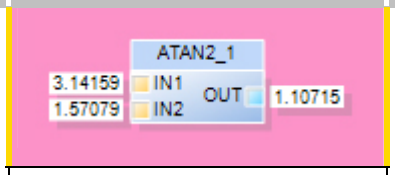
Any_Real
Any_Num
		Any_Real	**EXPT:**
			Экспонента с основанием (IN2)
			Результат: = $arg1^{arg2}$;
			Примеры:
			125.0 = expt(5.0, 3.0);
			4.0 = 16.0 ** 0.5;
			ST:
			`OUT := expt(IN1, IN2);`
			or
			`OUT := IN1 ** IN2;`

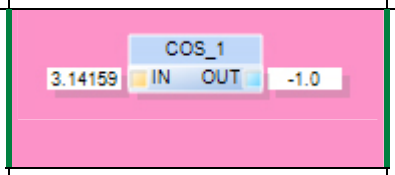
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_Real		Any_Real	**LN:** Натуральный логарифм Пример: +1.0 = ln(2.71828); **ST:** `OUT:= ln(IN);`
Any_Real		Any_Real	**LOG:** Логарифм с основанием 10 Пример: +1.0 = log(10.0); **ST:** `OUT:= log(IN);`

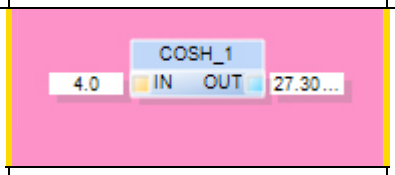
D.2.3. Тригонометрические

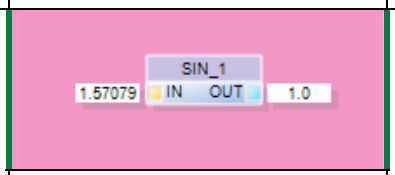
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_Real		Any_Real	**ACOS:** Арккосинус Пример: 1.57079 = acos(0.0); **ST:** `OUT:= acos(IN);`
Any_Real		Any_Real	**ASIN:** Арксинус Пример: -1.57079 = asin(-1.0); **ST:** `OUT:= asin(IN);`
Any_Real		Any_Real	**ATAN:** Арктангенс Пример: 1.0039 = atan(π/2.0); **ST:** `OUT:= atan(IN);`

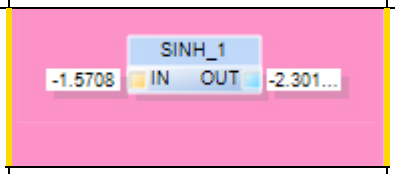
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

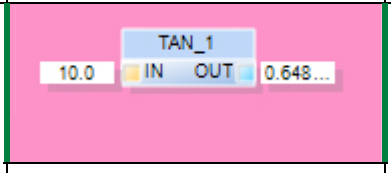
Any_Real
Any_Real
		Any_Real	**ATAN2:** Арктангенс
			Пример:
			1.1071 = atan2_real(π,π/2.0);
			ST:
			Разный вызов типов данных REAL и LREAL
			`OUT:= atan2_real(IN1, IN2);`
			`OUT:= atan2_lreal(IN1, IN2);`

Any_Real		Any_Real	**COS:** Косинус Пример: -1.0000 = cos(π); **ST:** `OUT:= cos(IN);`

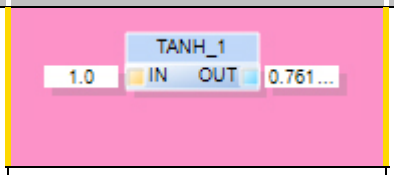
Any_Real		Any_Real	**COSH:** Гиперболический косинус
			Пример:
			+27.3082 = cosh_real(4.0);
			ST:
			Разный вызов типов данных REAL и LREAL
			`OUT:= cosh_real(IN);`
			`OUT:= cosh_lreal(IN);`

Any_Real		Any_Real	**SIN:** Синус Пример: 1.0 = sin(π/2); **ST:** `OUT:=sin(IN);`

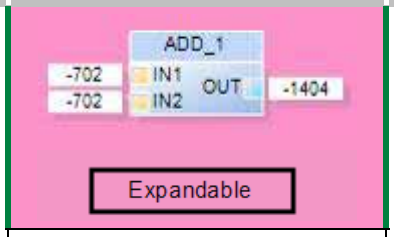
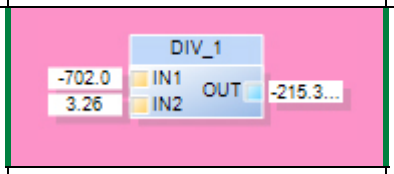
Any_Real		Any_Real	**SINH:** Гиперболический синус
			Пример:
			-2.3013 = sinh_real(-π/2.0);
			ST:
			Разный вызов типов данных REAL и LREAL
			`OUT:=sinh_real(IN);`
			`OUT:=sinh_lreal(IN);`

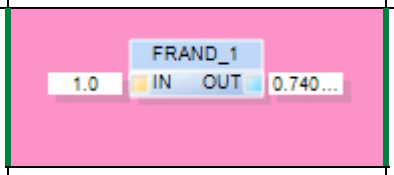
Any_Real		Any_Real	**TAN:** Тангенс Пример: 0.647 = tan(10.0); **ST:** `OUT:=tan(IN);`

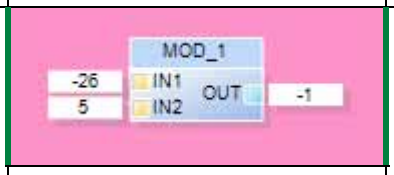
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

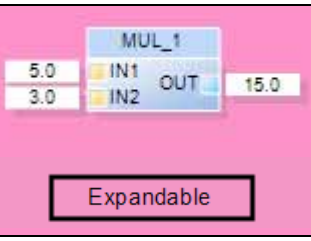
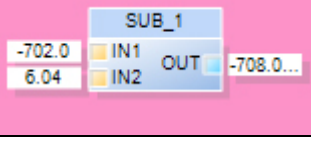
Any_Real		Any_Real	**TANH:** Гиперболический тангенс
			Пример:
			0.76159 = tanh_real(1.0);
			ST:
			Разный вызов типов данных REAL и LREAL
			`OUT:=tanh_real(IN);`
			`OUT:=tanh_lreal(IN);`

D.2.4. Разные

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_ Magnitude Any_ Magnitude		Any_ Magnitude	**ADD:** Сложение Пример: -1404 = -702 + -702; **ST:** Используемый оператор: `OUT:= IN1 + IN2 + ... + INn;`
Any_Num Any_Num		Any_Num	**DIV:** Деление Пример: -215.3 = -702.0 / 3.26; **ST:** Используемый оператор: `OUT:= IN1 / IN2;` **Внимание:** Если делитель IN2 = 0, то результат будет = 0 и в окне событий будет циклически появляться сообщение "Деление на ноль" ("Division by Zero").

Any_Real		Any_Real	**FRAND:** Случайное число в диапазоне {0 ... arg}
			Примеры:
			+0.07116 = frand_real(1.00);
			+2.92457 = frand_lreal(6.00);
			ST: разный вызов типов данных REAL и LREAL
			`OUT:= frand_real(IN);`
			`OUT:= frand_lreal(IN);`

Any_Int Any_Int		Any_Int	**MOD:** Остаток после деления (Modulo) Примеры: - 1 = -26 mod 5; **ST:** Используемый оператор: `OUT:= IN1 mod IN2;`

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_ Magnitude Any_ Magnitude		Any_ Magnitude	**ADD:** Сложение Пример: -1404 = -702 + -702; **ST:** Используемый оператор: `OUT:= IN1 + IN2 + ... + INn;`
Any_Num Any_Num		Any_Num	**MUL:** Умножение Примеры: 15.0 = 5.0 * 3.0; 4 = 2 * 2; **ST:** Используемый оператор `OUT:= IN1 * IN2 * … * INn;`
Any_ Magnitude Any_ Magnitude		Any_ Magnitude	**SUB:** Вычитание Пример: -708.04 = -702 – 6.04; **ST:** Используемый оператор: `OUT:= IN1 – IN2;`

D.3. Триггер

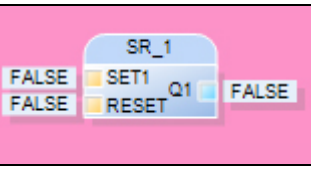
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool		Bool	**RS:** RS-триггер (для хранения статического двоичного значения) Приоритет R Таблица истинности: **Входные значения** / **Выход** SET | RESET1 | Q1 0 | 0 | Q1 0 | 1 | 0 1 | 0 | 1 1 | 1 | 0 **ST:** вызов невозможен
Bool Bool		Bool	**SR:** SR-триггер (для хранения статического двоичного значения) Приоритет S

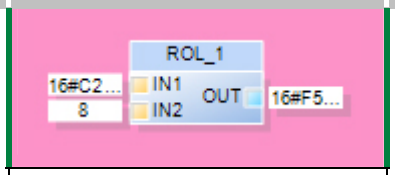
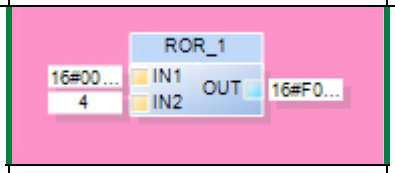
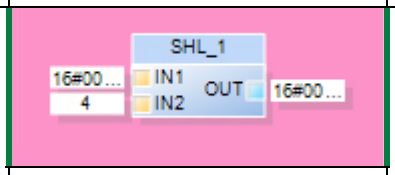
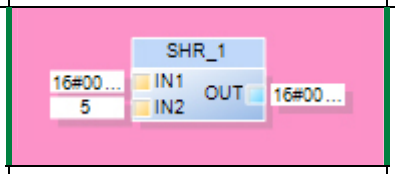
Таблица истинности:

Входные значения		Выход
SET1	RESET	Q1
0	0	Q1
0	1	0
1	0	1
1	1	1

ST: вызов невозможен

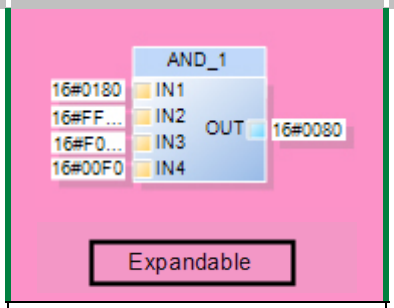
D.4. Битовая строка

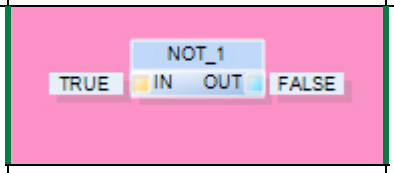
D.4.1. Побитовый сдвиг

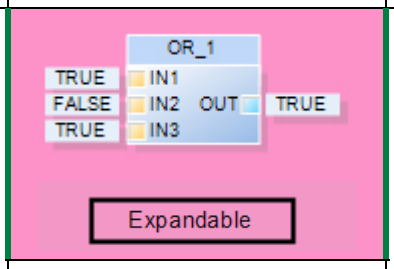
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_Bit Uint		Any_Bit	**ROL:** циклический сдвиг arg1 на arg2 битов влево Примеры: 16#F50000C2 =rol(16#C2F50000,8); 16#45678123 =rol(16#12345678,12); **ST:** `OUT := rol(IN1,IN2);`
Any_Bit Uint		Any_Bit	**ROL:** циклический сдвиг arg1 на arg2 битов вправо Примеры: 16#F00000C2 = ror(16#C2F,4); 16#F500000C = ror(16#CF5,8); **ST:** `OUT := ror(IN1,IN2);`
Any_Bit Uint		Any_Bit	**SHL:** сдвиг IN1 влево на IN2 бит, заполнение правых разрядов нулями Пример: 16#0D90 = shl(16#00D9,4); **ST:** `OUT := shl(IN1,IN2);`
Any_Bit Uint		Any_Bit	**SHR:** сдвиг IN1 вправо на IN2 бит, заполнение левых разрядов нулями Примеры: 16#000C = shr(16#0180,5); 16#00D9 = shr(16#0D90,4); **ST:** `OUT := shr(IN1,IN2);`

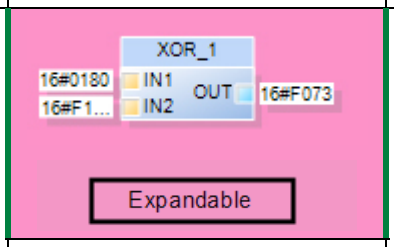
D.4.2. Bitwise_Boolean (побитовая логика)

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

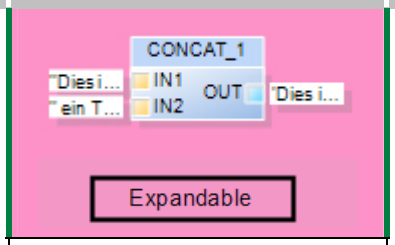
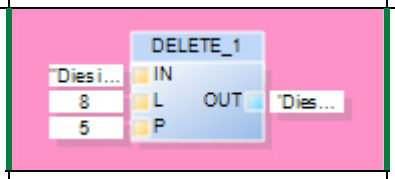
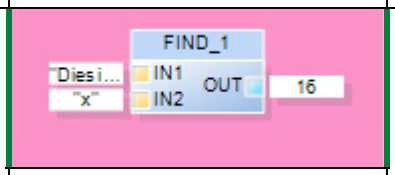
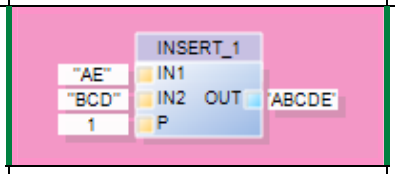
Any_Bit
Any_Bit
Any_Bit
Any_Bit
		Any_Bit	**AND:** Логическое И
			Пример:
			16#80=
			and(16#0180, 16#FFF0, 16#F0F0, 16#00F0);
			ST: Используемый оператор:
			`OUT := IN1 AND IN2 … AND INn;`

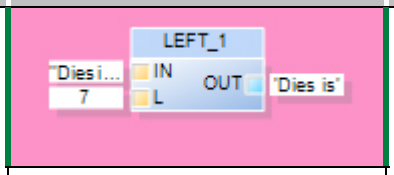
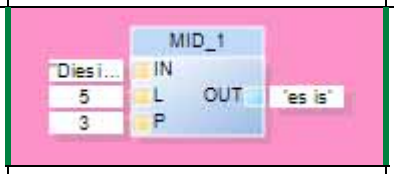
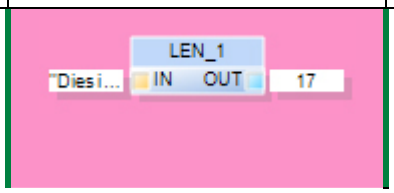
Any_Bit		Any_Bit	**NOT:** Логическое НЕТ
			Примеры:
			FALSE = not(TRUE);
			16#FE7F = not(16#0180);
			ST:
			`OUT := not IN;`
			или
			`OUT := not(IN);`

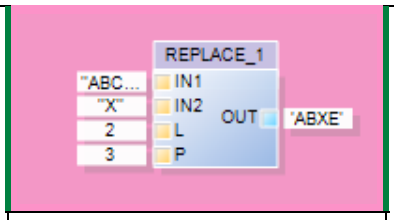
Any_Bit Any_Bit Any_Bit		Any_Bit	**OR:** Логическое ИЛИ Примеры: 1 = or(1, 0, 1); 16#F3 = or(16#F0,16#03); **ST:** Используемый оператор: `OUT:= IN1 or IN2 or … INn;`

Any_Bit
Any_Bit
		Any_Bit	**XOR:** Логическое исключающее ИЛИ
			Примеры:
			FALSE = xor(TRUE,TRUE);
			16#F073 =
			xor(16#0180,16#F1F3);
			ST: Используемый оператор:
			`OUT:= IN1 xor IN2 xor … INn;`

D.5. Символьная строка

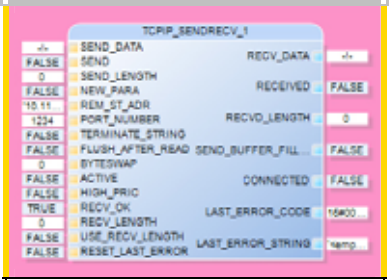
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_String Any_String		Any_String	**CONCAT:** Комбинирование (объединение) субстрок Примеры: 'This is a text'= concat('This is', ' a text') **ST:** `OUT:=concat(IN1, IN2,…,INn);`
Any_String Uint Uint		Any_String	**DELETE:** Удаление L символов строки, начиная с позиции P (включительно). Первый символ находится в поз. 1. Примеры: 'This a text'= delete('This is a text',3,5); 'DE' = delete('ABCDE',3,1); **ST:** `OUT:=delete(IN, L,P);`
Any_String Any_String		Int	**FIND:** Поиск первого совпадения с символом IN2 в строке IN1. Если символ не найден, то результат = 0. Примеры: 13 = find('This is a text', 'x'); 1 = find('This is a text', 'T'); **ST:** `OUT:=find(IN1, IN2);`
Any_String Any_String Uint		Any_String	**INSERT:** Вставить строку IN2 в строку IN1 после поз. P. Если P=0, то IN2 помещается в начало. Примеры: 'ABCDE' = insert('AE','BCD', 1); 'xABC' = insert('ABC','x',0); **ST:** `OUT:=insert(IN1, IN2,P);`

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_String Uint		Any_String	**LEFT:** Левая часть строки IN длиной L Пример: 'This is'= left('This is a text',7); **ST:** `OUT:=left(IN,L);`
Any_String Uint Uint		Any_String	**MID:** Часть строки IN длиной L, начиная с поз. P (включительно). Пример: 'is is' = mid('This is a text',5,3); **ST:** `OUT:=mid(IN,L,P);`
Any_String		Int	**LEN:** Длина строки Результат:= len(string); Примеры: 17= len('This is a text'); 4= len('Text'); **ST:** `OUT:=len(IN);`

Any_String
Any_String
Uint
Uint
		Any_String	**REPLACE:**
			Замена L символов строки IN на IN2, начиная с поз. P (включительно).
			Пример:
			'ABXE' =
			replace('ABCDE','X', 2,3);
			ST:
			`OUT:=`
			`replace(IN1, IN2, L, P);`

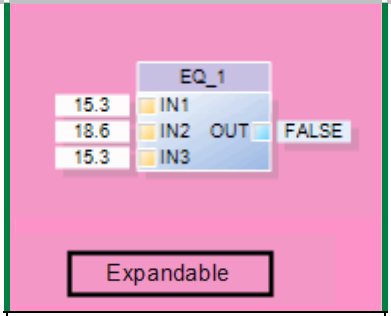
Any_String
Uint
		Any_String	**RIGHT:**
			Правая часть строки длиной L
			Пример:
			'a text'=
			right('This is a text',6);
			ST:
			`OUT:=`
			`right(IN, L);`

D.6. Обмен данными

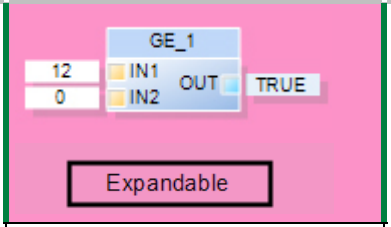
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any Bool Udint Bool String Udint Bool Bool Int Bool Bool Bool Udint Bool Bool		Any Bool Udint Bool Bool Dword String	**TCPIP_SendRecv:** Передача и получение данных по TCP/IP. Здесь выполняется отправка исходных данных по TCP/IP. Таким образом могут быть воссозданы все протоколы TCP/IP. Подробное описание содержится в разделе 5.3.2, "Комплексные функциональные блоки". **ST:** вызов в ST невозможен

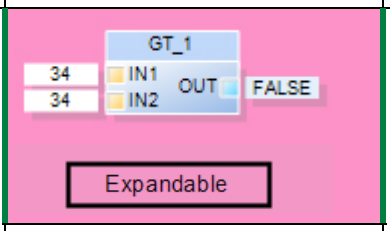
D.7. Сравнение

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

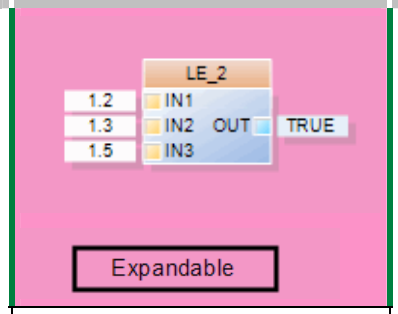
Any_
Elementary
Any_
Elementary
		Bool	**EQ:**
			Проверка равенства
			Результат TRUE, если все аргументы идентичны.
			Пример:
			FALSE = (15.3 = 18.6 = 15.3)
			ST: Используемый оператор
			`OUT := IN1 = IN2 ;`
			Допустимы только два аргумента. Реализация с использованием логической комбинации нескольких сравнений:
			`OUT := (IN1 = IN2) AND (IN1 = IN3) ;`

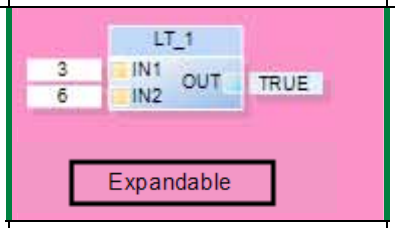
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

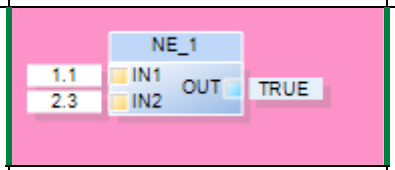
Any_ Elementary
Any_ Elementary
		Bool	**GE: Сравнение "больше или равно"**
			Результат TRUE, если IN1 больше или равен всем остальным аргументам.
			Пример:
			TRUE = 12 >= 0;
			ST: Используемый оператор:
			`OUT:= IN1 >= IN2;`
			Допустимы только два аргумента. Реализация с использованием логической комбинации нескольких сравнений:
			`OUT := (IN1 >= IN2) AND (IN1 >= IN3);`

Any_ Elementary
Any_ Elementary
		Bool	**GT: Сравнение "больше, чем"**
			Результат TRUE, если IN1 больше, чем все остальные аргументы.
			Пример:
			FALSE = 34 >34;
			ST: Используемый оператор:
			`OUT:= IN1 > IN2;`
			Допустимы только два аргумента. Реализация с использованием логической комбинации нескольких сравнений:
			`OUT := (IN1 > IN2) AND (IN1 > IN3);`

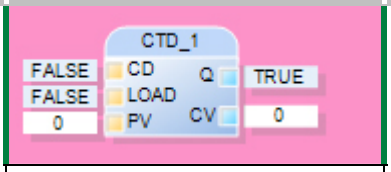
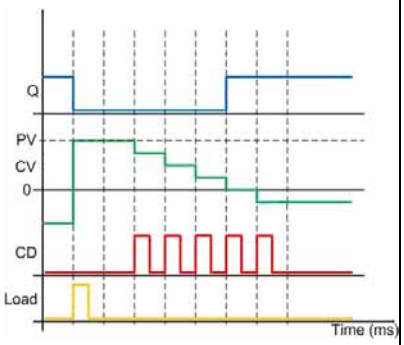
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

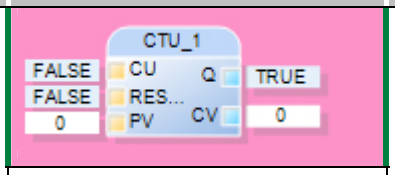
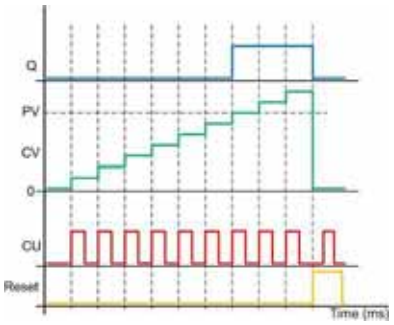
Any_ Elementary Any_ Elementary		Bool	**LE:**
			Сравнение "меньше или равно"
			Результат TRUE, если IN1 меньше или равен всем остальным аргументам.
			Пример:
			TRUE = (1.2 <= 1.3 <= 1.5);
			ST: Используемый оператор:
			`OUT:= IN1 <= IN2;`
			Допустимы только два аргумента. Реализация с использованием логической комбинации нескольких сравнений:
			`OUT := (IN1 <= IN2) AND (IN1 <= IN3);`

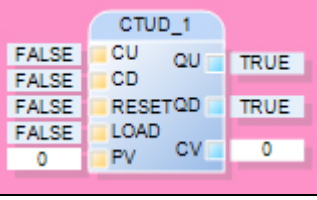
Any_ Elementary Any_ Elementary		Bool	**LT: Сравнение "меньше, чем"**
			Результат TRUE, если IN1 меньше, чем все остальные аргументы.
			Пример:
			TRUE = (3 < 6);
			ST: Используемый оператор:
			`OUT:= IN1 < IN2;`
			Допустимы только два аргумента. Реализация с использованием логической комбинации нескольких сравнений:
			`OUT := (IN1 < IN2) AND (IN1 < IN3);`

Any_ Elementary Any_ Elementary		Bool	**NE:** **Проверка неравенства** Результат TRUE, если IN1 не равен N1. Пример: TRUE = ('Text 1' <> 'Text 2'); **ST:** Используемый оператор: `OUT:= IN1 <> IN2;`

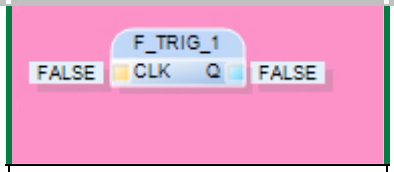
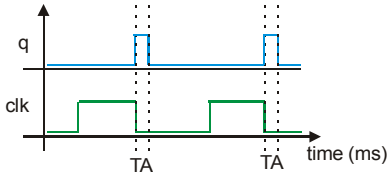
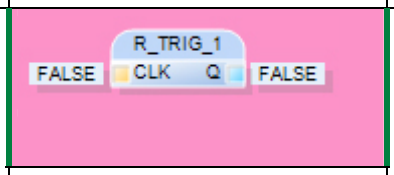
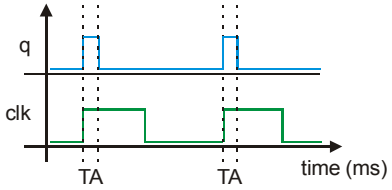
D.8. Счетчик

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool Int		Bool Int	**CTD:** Нисходящий счетчик Если счетчик установлен на LOAD = 1, то значение счетчика CV устанавливается на исходное значение PV. При каждом переднем фронте CD значение счетчика CV уменьшается на единицу. Как только выход счетчика CV <= 0, выход Q устанавливается на 1. Выход CV достигает минимального значения: -32,768.  **ST:** вызов невозможен

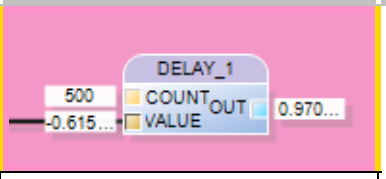
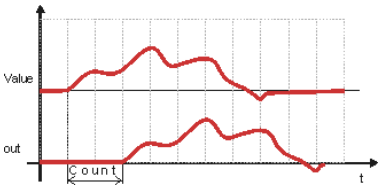
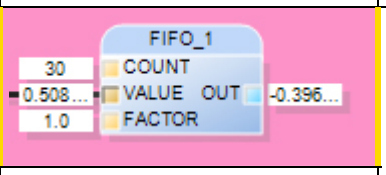
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool Int		Bool Int	**CTU:** Суммирующий счетчик При каждом переднем фронте CU значение счетчика CV увеличивается на единицу. Как только выход счетчика CV >= значение счетчика PV, выход Q устанавливается на "TRUE". Когда "RESET" = 1, выход Q устанавливается на "FALSE" и выход CV устанавливается на 0. Выход CV достигает максимального значения: 32 767.  **ST:** вызов невозможен

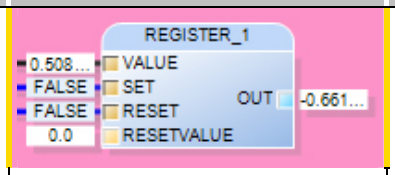
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool Bool Bool Int		Bool Bool Int	**CTUD:** Реверсивный счетчик (суммирование / вычитание) При каждом переднем фронте CU значение счетчика CV увеличивается на единицу за один период дискретизации. Когда выход счетчика "CV" >= значение счетчика "PV", выход QU устанавливается на 1 (схема потока данных, см. блок "CTU"). Если счетчик установлен на LOAD = 1, то значение счетчика CV устанавливается на исходное значение PV. При каждом переднем фронте CD значение счетчика CV уменьшается на единицу. Как только выход счетчика CV <= 0, выход "QD" устанавливается на 1 (схема потока данных, см. блок "CTD"). При сбросе счетчика ("RESET" = 1) его выход принимает значение 0. Диапазон значений для "CV": от -32 768 до 32 767 **ST:** вызов невозможен

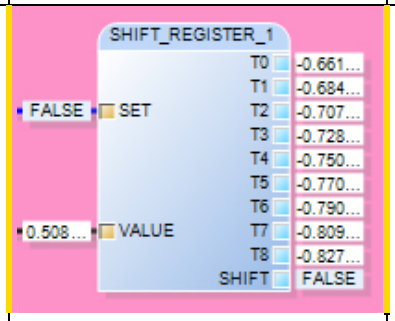
D.9. Детектирование фронта

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool		Bool	**F_TRIG:** Детектирование задних фронтов При заднем фронте на входе "CLK" выход Q принимает значение "TRUE" на время одного цикла задачи. Если вход "CLK" имеет значение "FALSE" при запуске системы, то функциональный блок генерирует импульс на выходе Q = "TRUE" в течение одного цикла.  **ST:** вызов невозможен
Bool		Bool	**R_TRIG:** Детектирование передних фронтов При переднем фронте на входе "CLK" выход Q принимает значение "TRUE" на время одного цикла задачи. Ситуация при запуске: Если вход "CLK" имеет значение "TRUE" при запуске системы, то функциональный блок генерирует импульс на выходе Q = "TRUE" в течение одного цикла.  **ST:** вызов невозможен

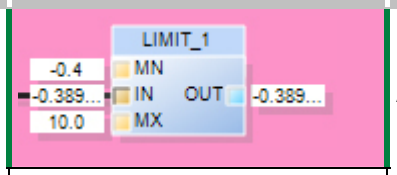
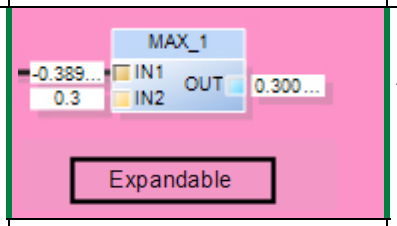
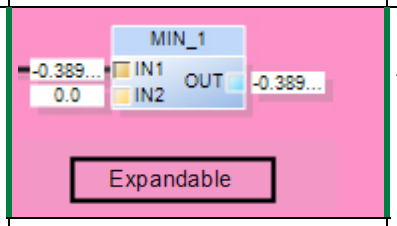
D.10. Регистрация

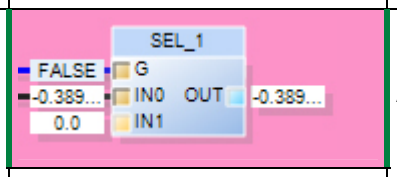
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Dint Any		Any	**DELAY:** Функция задержки Выходное значение "OUT" следует за входным значением "VALUE" с задержкой, которая определяется входом "COUNT" на некоторое количество циклов.  Если вы используете тип данных "ARRAY" ("VALUE" и "OUT"), то для блока действуют ограничения, связанные с объемом памяти. Если количество элементов массива ("ARRAY") превышает 64, то диапазон значений задержки (65 536) сокращается соответствующим образом. **ST:** вызов невозможен
Dint Real Real		Real	**FIFO:** **First In First Out - хранилище** Выходное значение "OUT" следует за входным значением "VALUE" с задержкой, которая определяется входом "COUNT" на некоторое количество циклов. Помимо этого, входное значение умножается на "FACTOR". **ST:** вызов невозможен

<table>
<tr><th>Тип входных данных</th><th>Строение блока</th><th>Тип выходных данных</th><th>Объяснение, пример, синтаксис ST</th></tr>
<tr>
<td>Any_
Magnitude
Bool
Bool
Any_
Magnitude</td>
<td> 
</td>
<td>Any_Magnitude</td>
<td>REGISTER:
Регистровая память
Блок работает с состоянием сигнала, а не фронтами.
Таблица функции:
<table>
<tr><th colspan="2">Входные значения</th><th>Выход</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: вызов невозможен</td>
</tr>
<tr>
<td>Bool

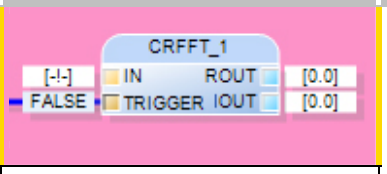
Real</td>
<td> 
</td>
<td>Real
Real
Real
Real
Real
Real
Real
Real
Bool</td>
<td>SHIFT_REGISTER:
Сдвиговый регистр
Пока значение входа "SET" = "TRUE", входное значение "VALUE" смещается на значение выхода Ti каждый цикл задачи.
Смещение, если "SET" = "TRUE"
T0: = VALUE(T_n) = текущий цикл
T1: = VALUE(T_{n-1}) = предыдущий цикл
T8: = VALUE(T_{n-8}] = самый старый цикл
где n = цикл задачи
ST: вызов невозможен</td>
</tr>
</table>

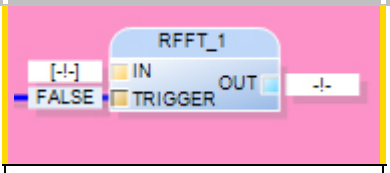
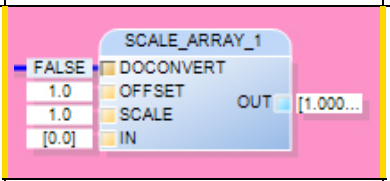
D.11. Выбор

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Any_ Elementary Any_ Elementary Any_ Elementary		Any_ Elementary	**LIMIT:** Предельное значение Входное значение IN ограничено предельными значениями MN (мин.) и MX (макс.). Примеры: -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:** Максимальное значение Примеры: 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:** Минимальное значение Примеры: -0.389 = min(-0.389, 0.3); 0 = min(0, 10, 12, 5); **ST:** `OUT := min(IN1, IN2, … , INn);`

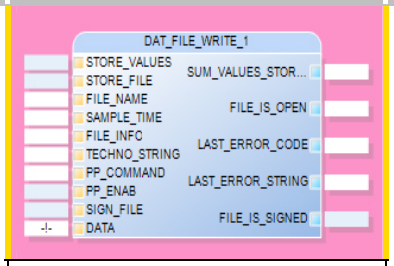
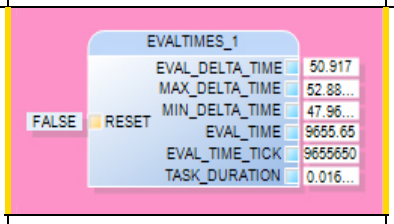
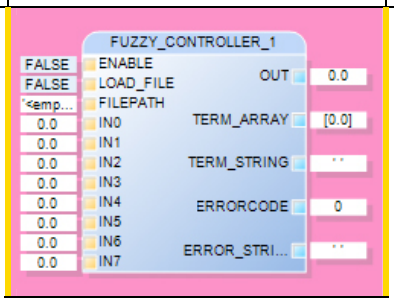
Bool
Any_ Elementary
Any_ Elementary
		Any_ Elementary	**SEL:** Selector (блок выбора)
			Выбор (1 из 2) с помощью бинарного сигнала "G"
			Таблица функции:
			SEL | OUT
			0 | IN0
			1 | IN1
			Пример:
			-0.389 = sel(FALSE, -0.389, 0);
			ST: Разный вызов для типов данных "REAL" и "INT", другие типы данных невозможны.
			ST:
			`OUT := sel_real(G,IN0, IN1);`
			`OUT := sel_int(G, IN0,IN2);`

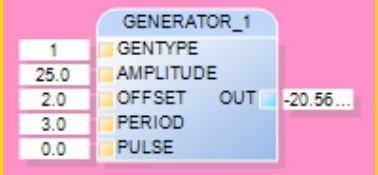
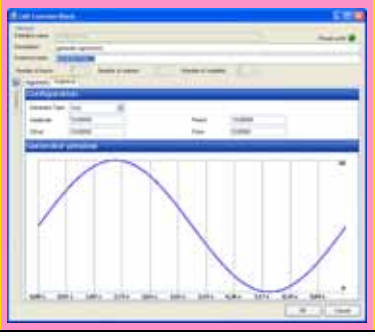
D.12. Обработка сигналов

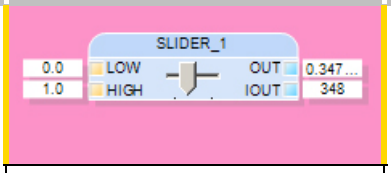
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Одномерный массив типа real с глубиной 2^n Bool		Одномерные массивы типа real с глубиной 2^{n-1}	**CRFFT:** Быстрое преобразование Фурье (FFT) с мнимым компонентом Должен быть одномерный массив типа "REAL", имеющий 2**n элемента на входе "IN". Выход в таком случае всегда представляет собой два массива одинакового типа длиной 2 **(n-1). Пример: IN ← ARRAY [0…2047] OF REAL ROUT → ARRAY [0…1023] OF REAL IOUT → ARRAY [0…1023] OF REAL Вычисление FFT активируется, когда вход "TRIGGER" = "TRUE". Только в этом случае на обработку блока затрачивается машинное время. Этот функциональный блок передает действительную часть на выход "ROUT" и мнимую часть на выход "IOUT" вычисления FFT. Режим вычисления: Абсолютная амплитуда, все значения массива имеют одинаковый вес (прямоугольное окно). **ST:** вызов невозможен

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Одномерный массив типа real с глубиной 2^n Bool		Одномерные массивы типа real с глубиной 2^{n-1}	**RFFT:** Быстрое преобразование Фурье (FFT) Должен быть одномерный массив типа "REAL", имеющий 2**n элемента на входе "IN". Выход в таком случае всегда представляет собой один массив аналогичного типа длиной 2 **(n-1). Пример: IN ← ARRAY [0 … 2047] OF REAL OUT → ARRAY [0 … 1023] OF REAL Вычисление FFT активируется, когда вход "TRIGGER" = "TRUE". Только в этом случае на обработку блока затрачивается машинное время. Функциональный блок передает результат FFT на выход в соответствии с режимом вычисления: Абсолютная амплитуда, все значения массива имеют одинаковый вес (прямоугольное окно). **ST:** вызов невозможен
Bool Real Real Any_ Derived		Any_ Derived	**SCALE_ARRAY:** Если вход "DOCON-VERT" = "TRUE", то каждый элемент входного массива "IN" умножается на "SCALE" и складывается со значением входа "OFFSET". Теперь массив с измененным масштабом доступен на выходе "OUT". **ST:** вызов невозможен

D.13. Специальные блоки

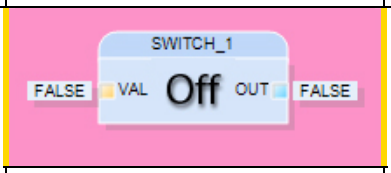
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool String Lreal Lreal Lreal Lreal Lreal Lreal Lreal Lreal		Dint Usint Dword String Bool	**DAT_FILE_WRITE:** Этот блок можно использовать для записи сигналов непосредственно в ibaLogic для последующего анализа в ibaAnalyzer. Подробное описание содержится в разделе 5.3.1, "Комплексные функциональные блоки". **ST:** вызов невозможен
Bool		Real Real Real Real Udint Real	**EVALTIMES:** **Вывод данных вычислений** EVAL_DELTA_TIME = текущее время цикла задачи (в мс) MAX_DELTA_TIME = макс. время цикла задачи с момента предыдущего запуска (в мс) MIN_DELTA_TIME = минимальное время цикла EVAL_TIME = время, прошедшее с момента предыдущего запуска (в мс) EVAL_TIME_TICK = время, прошедшее с момента предыдущего запуска в мкс TASK DURATION = время вычисления текущей задачи. **ST:** вызов невозможен
Bool Bool String Lreal Lreal Lreal Lreal Lreal Lreal Lreal Lreal		Lreal Array [0..8] of Lreal String Int String	**FUZZY_CONTROLLER:** Блок управления на базе нечеткой логики. Подробное описание содержится в разделе 5.3.5, "Комплексные функциональные блоки". **ST:** вызов невозможен

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Int Real Real Real Real	 **Graphical Configuration** 	Real	**GENERATOR:** Генератор функций для синусоидальных, прямоугольных и треугольных (пилообразные колебания) сигналов. "GENTYPE" = 1 - синусоидальный, 2 - прямоугольный, 3 - треугольный сигнал "AMPLITUDE" = Значение амплитуды. Есть только одно значение, которое вычисляется симметрично оси X, т.е. применимо как к

<table>
<tr><th>Тип входных данных</th><th>Строение блока</th><th>Тип выходных данных</th><th>Объяснение, пример, синтаксис ST</th></tr>
<tr><td>Real
Real</td><td> 
</td><td>Real
Dint</td><td>SLIDER: Регулятор ползунка

В зависимости от позиции ползунка этот функциональный блок отправляет значение на выход "OUT", значение которого находится в интервале между минимальным и максимальным. По умолчанию входы установлены на 0 и 1, но эти значения можно изменить. (Изменить значения по умолчанию можно, щелкнув по модулю двойным щелчком)
Выход "IOUT" возвращает относительное значение положения ползунка с шагом в одну тысячную (0 ... 1000).

Стрелку ползунка также можно двигать с помощью символов → и ←.

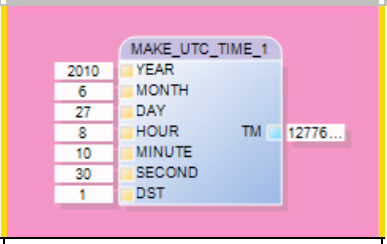
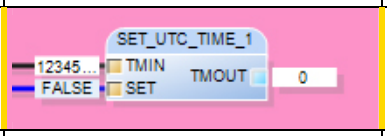
ST: вызов невозможен</td></tr>
<tr><td>Bool</td><td> 
</td><td>Bool</td><td>SWITCH: Переключатель

Для переключения состояния щелкните левой кнопкой мыши по значку "OFF".

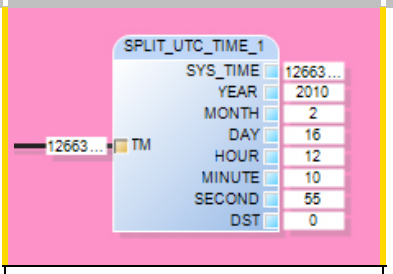
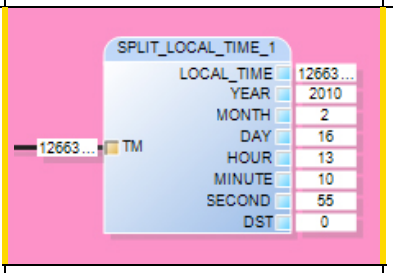
Между положением переключателя и входом есть функция ИЛИ.

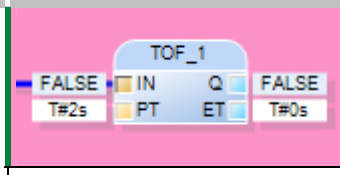
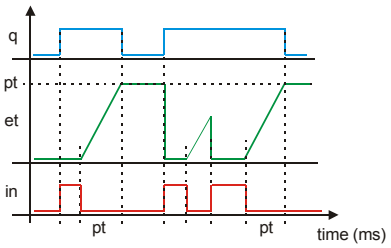
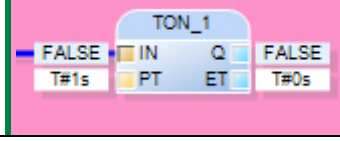
Таблица истинности:
<table><tr><th>ПЕРЕКЛЮЧ.</th><th>ЗНАЧ.</th><th>ВЫХ.</th></tr><tr><td>ON</td><td>FALSE</td><td>TRUE</td></tr><tr><td>ON</td><td>TRUE</td><td>TRUE</td></tr><tr><td>OFF</td><td>FALSE</td><td>FALSE</td></tr><tr><td>OFF</td><td>TRUE</td><td>TRUE</td></tr></table>
ST: вызов невозможен</td></tr>
</table>

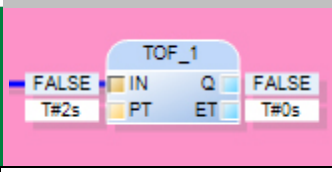
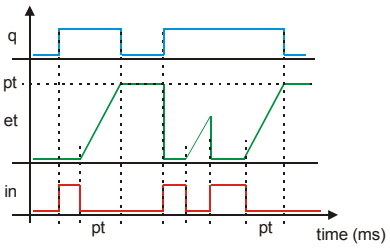
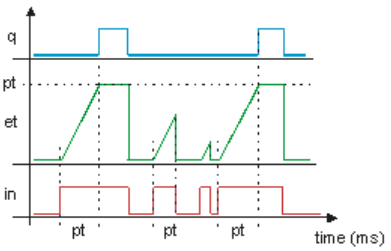
D.14. Таймер

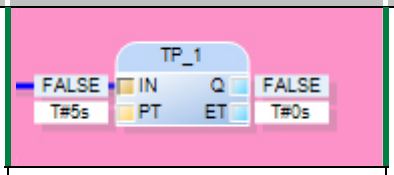
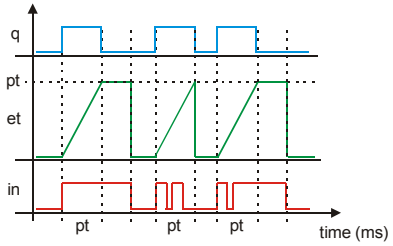
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Dint Dint Dint Dint Dint Dint Dint		Udint	**MAKE_UTC_TIME: Кодирование универсального координированного времени (UTC)** [8] Этот функциональный блок генерирует универсальное координированное время (UTC) на выходе "TM" из входных переменных: год ("YEAR"), месяц ("MONTH"), день ("DAY"), час ("HOUR"), минута ("MINUTE") и секунда ("SECOND"). Локальная временная зона [9] **не** принимается во внимание. На входе "DST" вы можете задать летнее время (Daylight Savings Time). Пример: 27.06.2010/08:10:30 во временной зоне GMT+01 → TM = 1277626230 **ST:** вызов невозможен
Udint Bool		Udint	**SET_UTC_TIME:** Установка универсального координированного времени (UTC) Этот функциональный блок устанавливает значение "UTC System Time" платформы ibaLogic (Windows PC или PADU-S-IT) на значение входа "TMIN", когда вход "SET" = "TRUE". Локальная временная зона и летнее время (DST) **не** принимаются во внимание. **ST:** вызов невозможен

[8] Время UTC (универсальное координированное время) содержит время в секундах с 01.01.1970, 00:00 полночь, относительно GMT+00.
[9] Информация касательно временной зоны и летнего времени (DST) извлекается из настроек операционной системы Windows XP или Windows CE.

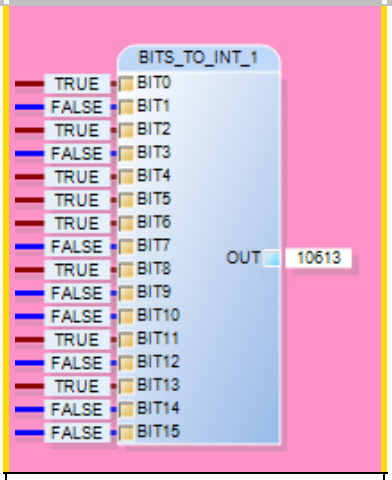
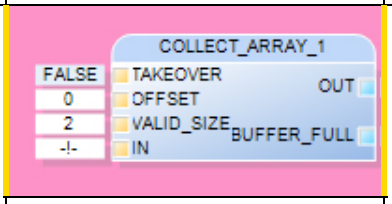
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Udint		Udint Dint Dint Dint Dint Dint Dint Dint	**SPLIT_UTC_TIME: Перекодирование времени** UTC **в GMT.** Из времени UTC функциональный блок генерирует на входе "TM" выходные переменные: год ("YEAR"), месяц ("MONTH"), день ("DAY"), час ("HOUR"), минута ("MINUTE") и секунда ("SECOND"). Локальная временная зона не принимается во внимание. Летнее время (Daylight Savings Time - DST) отображается на выходе "DST". **ST:** вызов невозможен
Udint		Udint Dint Dint Dint Dint Dint Dint Dint	**SPLIT_LOCAL_TIME: Перекодирование времени** UTC **в локальное время.** Из времени UTC функциональный блок генерирует на входе "TM" выходные переменные: год ("YEAR"), месяц ("MONTH"), день ("DAY"), час ("HOUR"), минута ("MINUTE") и секунда ("SECOND"). Локальная временная зона не принимается во внимание. Летнее время (Daylight Savings Time - DST) отображается на выходе "DST". **ST:** вызов невозможен

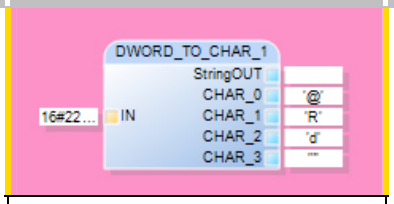
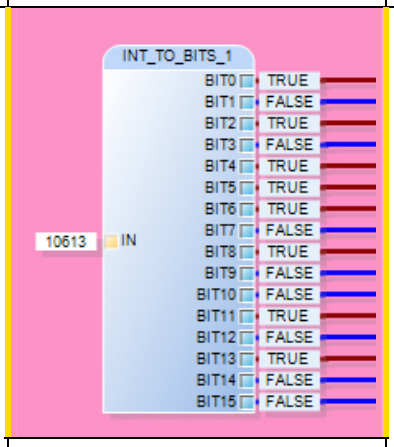
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Time		Bool Time	**TOF:** Задержка выключения (выключение с задержкой по времени) Если вход "IN" имеет значение "TRUE", то выход "Q" принимает значение "TRUE" без задержки по времени. Задний фронт на входе "IN" запускает временную задержку PT. По истечении времени задержки выход "Q" принимает значение "FALSE". Значение выхода "Q" не изменяется, если время выключения "IN" меньше временной задержки. Выход "ET" содержит значение, соответствующее уже истекшему времени.  **ST:** вызов невозможен
Bool Time		Bool Time	**TON:** Вкл. задержки (включение временной задержки) Задержка включения (включение с задержкой по

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Time		Bool Time	**TOF:** Задержка выключения (выключение с задержкой по времени) Если вход "IN" имеет значение "TRUE", то выход "Q" принимает значение "TRUE" без задержки по времени. Задний фронт на входе "IN" запускает временную задержку PT. По истечении времени задержки выход "Q" принимает значение "FALSE". Значение выхода "Q" не изменяется, если время выключения "IN" меньше временной задержки. Выход "ET" содержит значение, соответствующее уже истекшему времени.  **ST:** вызов невозможен времени) Передний фронт на входе "IN" запускает временную задержку PT. По истечении времени задержки выход "Q" принимает значение "TRUE". Сигнал "FALSE" на входе "IN" незамедлительно передается на выход "Q". Выход "Q" не получает значения, если время включения "IN" меньше временной задержки. Выход "ET" содержит значение, соответствующее уже истекшему времени.  **ST:** вызов невозможен

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Time		Bool Time	**TP:** импульс времени (pulse extension) Восходящий фронт на входе "IN" устанавливает на "TRUE" выход "Q" на время импульса PT. Выход "Q" не может быть сброшен во время импульса. Выход "ET" содержит значение, соответствующее уже истекшему времени.  **ST:** вызов невозможен

D.15. Преобразование типа данных

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool		Int	**BITS_TO_INT:** Преобразование 16 бит в целое значение Пример: 10613 = 2#0010_1001_0111_0101, (Bit15 …….…….. Bit0) **ST:** вызов невозможен
Bool Uint Uint Any_ Derived		Any_ Derived Bool	**COLLECT_ARRAY** Этот блок используется для передачи частей одного массива в другой. "TAKEOVER": Пока значение этого входа "TRUE", данные передаются в выходной массив. "OFFSET": Соответствует элементу входного массива, начиная с которого должны копироваться данные. "VALID_SIZE": Количество элементов для копирования. "IN": Входной массив любого размера. "OUT": Выходной массив "BUFFER_FULL": Выходной массив заполнен, данные должны быть считаны. **ST:** вызов невозможен

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Dword		String String String String String	**DWORD_TO_CHAR** Преобразование "DWORD" в четыре отдельных символа типа "STRING" 16#22645240 = ' @', 'R', 'd', "" **ST:** вызов в ST невозможен
Int		Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool Bool	**INT_TO_BITS:** Преобразует целое значение в 16 бит (функция, обратная BITS_TO_INT) **ST:** вызов невозможен

D.15.1. Ограничивающий преобразователь

Эти блоки преобразования выполняют особую функцию: перед выполнением преобразования типа они ограничивают диапазон значений входного типа в соответствии с диапазоном значений выходного типа. Следующий пример иллюстрирует отличия по сравнению со стандартным преобразователем.

Ограничивающий преобразователь: ограничить: dint_to_int(57700) результат: 32767

(Сначала выходное значение ограничивается диапазоном значений (-32768 … 32767), а затем выполняется преобразование типа).

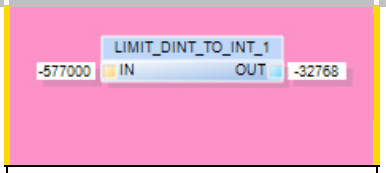
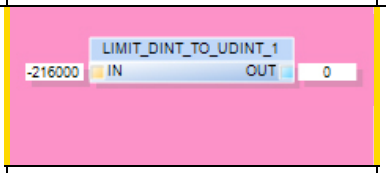
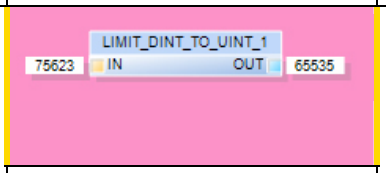
Стандартный преобразователь: dint_to_int(577000) результат: -12824

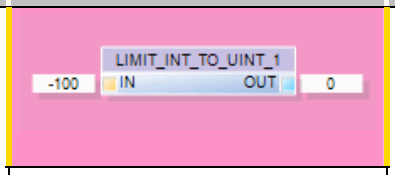
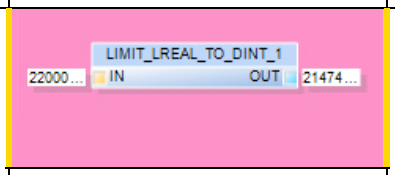
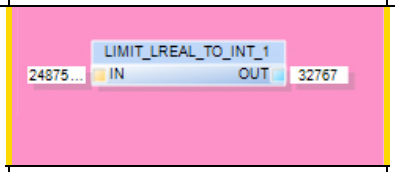
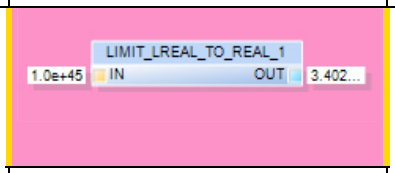
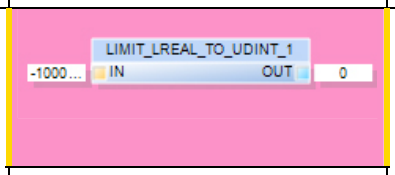
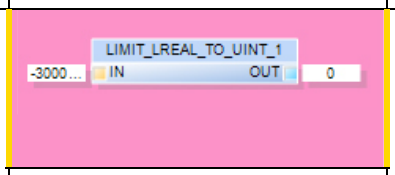
(16 младших битов сохраняются и конвертируются, при этом старший бит (бит 15) интерпретируется как знаковый бит.)

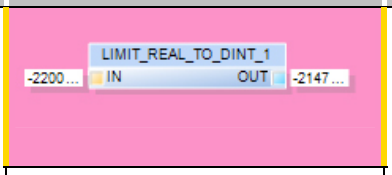
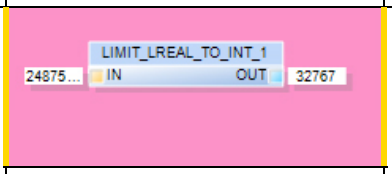
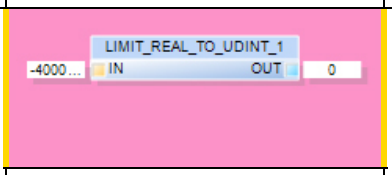
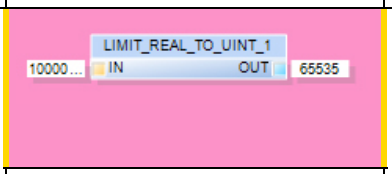
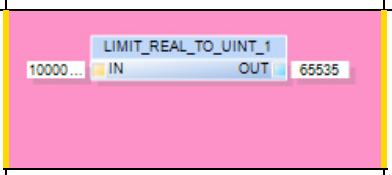
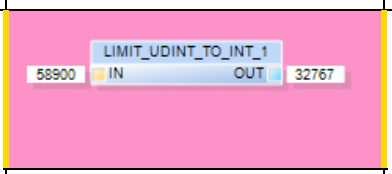
Мы рекомендуем использовать ограничивающий преобразователь, если диапазон значений целевого типа меньше диапазона значений исходного типа. Это касается следующих типов преобразования:

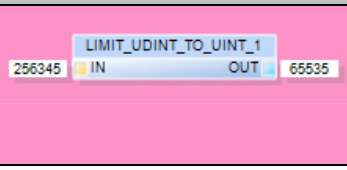
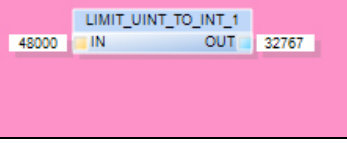
INT	UINT	DINT	UDINT	REAL	LREAL
INT → UINT INT → SINT INT → USINT	UNIT → INT UINT → SINT UINT → USINT	DINT → INT DINT → UDINT DINT → UINT DINT → SINT DINT → USINT	UDINT → DINT UDINT → INT UDINT → UINT UINT → SINT UINT → USINT	REAL → DINT REAL → INT REAL → UDINT REAL → UINT REAL → SINT REAL → USINT	LREAL → DINT LREAL → INT LREAL → REAL LREAL → UDINT LREAL → UINT LREAL → SINT LREAL → USINT

В структурированном тексте ограничивающие преобразователи доступны с помощью функций:
"limit_*source_type*_to_*target_type*"

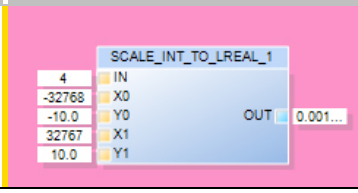
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Dint		Int	**LIMIT_DINT_TO_INT:** Пример: -577000 => -32768; **ST:** `OUT:= limit_dint_to_int(IN);`
Dint		Udint	**LIMIT_DINT_TO_UDINT:** Пример: -216000 => 0; **ST:** `OUT:= limit_dint_to_udint(IN);`
Dint		Uint	**LIMIT_DINT_TO_UINT:** Пример: 75623 => 65535 **ST:** `OUT:= limit_dint_to_uint(IN);`

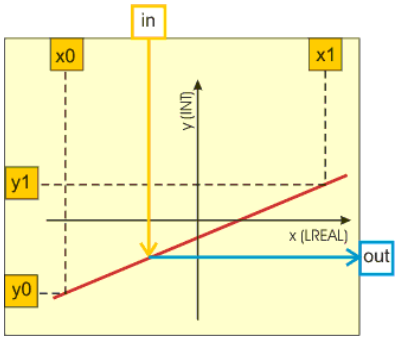
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Int		Uint	**LIMIT_INT_TO_UINT** Пример: -100 => 0 **ST:** `OUT:= limit_int_to_uint(IN);`
Lreal		Dint	**LIMIT_LREAL_TO_DINT:** Пример: 2.2 E+09 => 2147483648; **ST:** `OUT:= limit_lreal_to_dint(IN);`
Lreal		Int	**LIMIT_LREAL_TO_INT:** Пример: 248,758 => 32,767; **ST:** `OUT:= limit_lreal_to_int(IN);`
Lreal		Real	**LIMIT_LREAL_TO_REAL** Пример: 1E+45 => 3.402823466 E+38 **ST:** `OUT:= limit_lreal_to_real(IN);`
Lreal		Udint	**LIMIT_LREAL_TO_UDINT:** Пример: -1 E+12 => 0; **ST:** `OUT:= limit_lreal_to_udint (IN);`
Lreal		Uint	**LIMIT_LREAL_TO_UINT:** Пример: -3 E+12 => 0; **ST:** `OUT:= limit_lreal_to_uint(IN);`

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Real		Dint	**LIMIT_REAL_TO_DINT:** Пример: -2.2 E+09 => -2147483648; **ST:** `OUT:= limit_real_to_dint(IN);`
Real		Int	**LIMIT_REAL_TO_INT:** Пример: 248,758 => 32,767; **ST:** `OUT:= limit_real_to_int(IN);`
Real		Udint	**LIMIT_REAL_TO_UDINT:** Пример: -4000.0 => 0 **ST:** `OUT:= limit_real_to_udint(IN);`
Real		Uint	**LIMIT_REAL_TO_UINT:** Пример: 1*E+12 => 65,535; **ST:** `OUT:= limit_real_to_uint(IN);`
Udint		Dint	**LIMIT_UDINT_TO_DINT:** Пример: 3123456789 => 2147483647 **ST:** `OUT:= limit_udint_to_dint(IN);`
Udint		Int	**LIMIT_UDINT_TO_INT:** Пример: 558900 => 32767 **ST:** `OUT:= limit_udint_to_int(IN);`

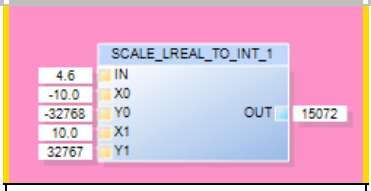
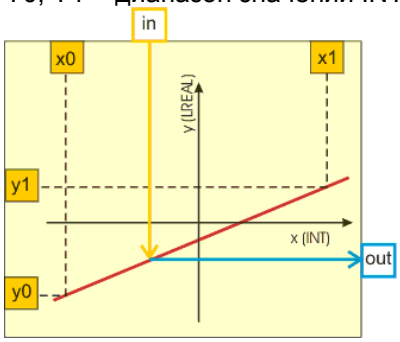
Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Udint		Uint	**LIMIT_UDINT_TO_UINT:** Пример: 256,345 => 65,535 **ST:** `OUT:= limit_udint_to_uint(IN);`
Uint		Int	**LIMIT_UINT_TO_INT:** Пример: 48,000 => 32,767 **ST:** `OUT:= limit_uint_to_int(IN);`

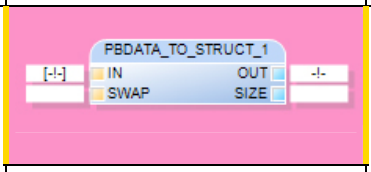
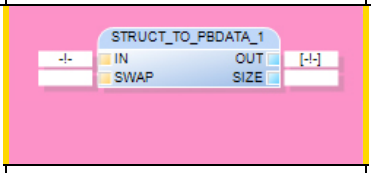
D.15.2. Масштабирующий преобразователь

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
Int Int Lreal Int Lreal		Lreal	**SCALE_INT_TO_LREAL:** Этот модуль преобразует значение "INTEGER" в значение "LREAL" и выполняет его линейное

			масштабирование.
			
			Применение:
			Преобразование аналогового входа (целое значение - 32768 ... 32767) в физический параметр, например +/- 10 вольт.
			
			IN:
			Входное значение (аналоговый вход)
			X0, X1: Диапазон значений для входного значения (Int)
			Y0, Y1: Диапазон значений Целевой параметр (Lreal)
			
			Пример:
			4 → 0.0013733119 V
			X0, X1 = -32768 / +32767
			Y0, Y1 = -10.0 / + 10.0
			IN = 4
			OUT = 0.0013733119
			
			 
			
			Реализация (преобразования типов были опущены в целях ясности):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 - y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out = aa*in + bb;`
			`end_if;`
			ST: вызов невозможен
			
			Важно:
			Поскольку диапазон целых значений в принципе несимметричен, 0 на входе не дает 0 на выходе. Поскольку это всегда приводит к ошибкам, iba AG рекомендует указывать симметричный диапазон значений для входа (-32767/+ 32767).

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST

Lreal
Lreal
Int
Lreal
Int
		Int	**SCALE_LREAL_TO_INT:**
			Этот модуль преобразует значение "LREAL" в значение "INTEGER" и выполняет его линейное масштабирование.
			Применение: преобразование физического параметра (например, +/- 10 вольт) в аналоговый выход (целое значение - 32768 … 32767)
			Пример:
			4.6 V → 15072
			X0, X1 = диапазон физических значений (-/+ 10 В)
			Y0, Y1 = диапазон значений INT
			 
			Реализация (преобразования типов были опущены в целях ясности):
			`dx := x1-x0;`
			`if (dx <> 0.0)`
			`then`
			`  aa := (y1 - y0) / dx;`
			`  bb := y0 - aa*x0;`
			`  out := aa*in + bb;`
			`end_if;`
			ST: вызов невозможен

Array[0..60] of Dword Int		Any_Struct Byte	**PBDATA_TO_STRUCT:** Создает структуру из массива "DWORD", состоящего из 61 элемента (использование с помощью карты Profibus SST) **ST:** вызов невозможен
Any_Struct Byte		Array[0..60] of Dword Int	**STRUCT_TO_PBDATA:** Создает из структуры массив "DWORD", состоящий из 61 элемента (использование с помощью карты Profibus SST) **ST:** вызов невозможен

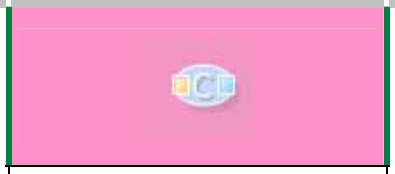
D.15.3. Стандартный преобразователь

Все стандартные блоки преобразования скомбинированы в один блок. После перетаскивания блока в область программирования, появится диалоговое окно выбора, в котором можно определить типы данных для входов и выходов.

Правила преобразования:

- Для целевого типа "BOOL" результат "FALSE", если вход имеет значение 0, 0.0, 16#0 или T#0ms, в противном случае будет результат "TRUE".
- При преобразовании типов данных real в целочисленные типы данных значения преобразуются численно и округляются в соответствии с правилами арифметики. Значения не имеют ограничений.
  Пример: 82600,0(REAL) → 82600(DINT) → 17064 → (INT)
- Преобразование типов данных "INTEGER" и "WORD" выполняется с помощью преобразования типа без изменения порядка битов. Требуемые ограничения не выполняются, и биты высшего порядка отбрасываются.
- Для типов данных от real до "WORD" сначала выполняется численное преобразование значения в "DINT", а затем - преобразование типа данных без изменения порядка битов.

Если вы объединяете два коннектора с разными типами данных, то автоматически создается соответствующий блок преобразования (при условии, что преобразование возможно). Более подробная информация содержится в пункте 6.9.1, "Преобразователи ".

Тип входных данных	Строение блока	Тип выходных данных	Объяснение, пример, синтаксис ST
			**TYPE_TO_TYPE:** После перетаскивания блока в область программирования, появится диалоговое окно выбора, в котором можно определить типы данных для входов и выходов. **ST:** Имя образуется из, например, *Исходный_тип*_в_*целевой_тип* (*Source_type*_to_*Target_type*). Function name is formed from *Source_type*_to_*Target_type*, например. `OUT := real_to_int(IN);`

Примечание

Преобразование в TIME:

Целочисленные значения интерпретируются как миллисекунды.

Значения типа Real интерпретируются как секунды.

Пример:

Целочисленное значение 4711, соответственно, возвращает 4,177 секунды.

Значение real 4711,0 возвращает 4711 секунд (то есть 1 час 18 минут и 31 секунда).

E. Коды ошибок

E.1. Коды ошибок DAT_FILE_WRITE

Код ошибки	Значение
16#FFFFFFF1	PP_COMMAND не определена!
16#FFFFFFF2	Сбой при выполнении PP_COMMAND!
16#FFFFFFF3	Сбой запуске PP_COMMAND!
16#FFFFFFF4	Сбой при закрытии файла xxxx.dat
16#FFFFFFF5	Период дискретизации установлен на 0.0. Установите верный период дискретизации!
16#FFFFFFF6	Кол-во сигналов в конфигурации DatfileWrite превышает допустимое значение в апп. ключе
16#FFFFFFF7	Сбой при записи данных в файл. Проверьте наличие свободного пространства на диске
16#FFFFFFF8	Сбой при создании файла. Проверьте имя файла и наличие свободного пространства на диске!
16#FFFFFFF9	Не найдены обработчики данных для модуля x
16#FFFFFFFA	Ошибка при получении обработчиков данных для модуля x
16#FFFFFFFB	Не найдены обработчики данных
16#FFFFFFFC	Не найдены данные конфигурации модуля для модуля x
16#FFFFFFFD	Ошибка при считывания данных конфигурации модуля для модуля x
16#FFFFFFFE	Конфигурация модуля не определена!
16#FFFFFFFF	Не найдены данные конфигурации

E.2. Коды ошибок TCPIP_SENDRECV

Код ошибки	Значение	Предлагаемое решение
HEX: 16#0000271d DEC: 10013	Разрешение отклонено - доступ к сокету запрещен в соответствии с правами доступа.	Зайдите под именем пользователя с правами администратора.
16#00002740 10048	Адрес уже используется - разрешено только одно использование адреса сокета.	Указанный адрес / порт уже используется.
16#00002741 10049	Невозможно назначить требуемый адрес - запрашиваемый адрес недопустим в его контексте.	
16#0000273f 10047	Семейство адресов не поддерживается семейством протоколов - был использован адрес, который несовместим с запрошенным протоколом.	
16#00002735 10037	Операция уже осуществляется - неблокирующий сокет, на котором предпринята операция, уже выполняет операцию.	
16#00002745 10053	Программа вызвала аварийное завершение соединения - установленное соединение прервано программным обеспечением на вашей хост-машине.	
16#0000274d	Соединение отклонено - невозможно установить	

Код ошибки	Значение	Предлагаемое решение
10061	соединение, поскольку удаленная машина его отвергает.	
16#00002746 10054	Соединение сброшено удаленной системой - существующее соединение принудительно закрыто удаленной стороной.	
16#00002737 10039	Требуется указание удаленного адреса - при операции с сокетом не указан требуемый адрес.	
16#0000271e 10014	Неверный адрес - система обнаружила неверный указатель на адрес при попытке использовать его в вызове функции.	
16#00002750 10064	Хост не работает - сбой операции с сокетом, поскольку удаленный хост не работает.	
16#00002751 10065	Нет маршрута к хосту - попытка обращения к хосту, к которому невозможно определить маршрут.	
16#00002734 10036	Операция выполняется - выполняется блокирующая операция.	
16#00002714 10004	Прерван вызов функции - блокирующая операция прервана вызовом WSACancelBlockingCall.	
16#00002726 10022	Недопустимый аргумент - передан недопустимый аргумент.	
16#00002748 10056	Сокет уже подключен - запрос соединения сделан на уже подключенный сокет.	
16#00002728 10024	Слишком много открытых файлов - слишком много открытых сокетов.	
16#00002738 10040	Слишком длинное сообщение - сообщение, посланное в сокет, превышает длину внутреннего буфера, или буфер, используемый для приема датаграмм, меньше чем датаграмма.	Сократите длину байтов для передачи.
16#00002742 10050	Сеть отключена - операция с сокетом обнаружила неактивную сеть.	
16#00002744 10052	Сеть сбросила соединение - соединение было разорвано в связи с тем, что функция проверки активности обнаружила сбой во время выполнения операции.	
16#00002743 10051	Сеть недостижима - попытка осуществить операцию с сокетом на недостижимой сети.	
16#00002747 10055	Нет свободного буферного пространства - невозможно осуществить операцию, поскольку системе не хватает буферного пространства или переполнена очередь.	
16#0000273° 10042	Неверная опция протокола - при вызове getsockopt или setsockopt указана неизвестная, недопустимая или неподдерживаемая опция или уровень.	
16#00002749 10057	Сокет не подключен - запрос на отправку или получение данных был запрещен, поскольку сокет не подключен.	
16#00002736 10038	Попытка операции на чем-то, что не является сокетом - была предпринята попытка операции на чем-то, что сокетом не является.	
16#0000273d	Операция не поддерживается - предпринятая операция	

Код ошибки	Значение	Предлагаемое решение
10045	не поддерживается для ссылающегося объекта.	
16#0000273e 10046	Семейство протоколов не поддерживается - семейство протоколов не сконфигурировано в системе или для него не существует реализации.	
16#00002753 10067	Слишком много процессов - реализация Windows Sockets может иметь предельное количество приложений, которые могут использовать ее одновременно.	
16#0000273b 10043	Протокол не поддерживается - протокол не сконфигурирован в системе или для него не существует реализации.	
16#00002739 10041	Неверный тип протокола для сокета - при вызове функции socket указан протокол, который не поддерживает семантику запрошенного типа сокета.	
16#0000274a 10058	Невозможно послать данные после закрытия сокета - запрос на отправку или получение данных был запрещен, поскольку сокет уже закрыт.	
16#0000273c 10044	Неподдерживаемый тип сокета - это семейство адресов не поддерживает указанный тип сокета.	
16#0000274c 10060	Истекло время ожидания соединения - попытка соединения завершилась неудачей, поскольку не поступил ответ от удаленной стороны.	
16#0000277d 10109	Класс не найден - указанный класс не найден.	
16#0000277a 10106	Не удается инициализировать нужного поставщика услуг - либо DLL поставщика услуг не могут быть загружены, либо произошел сбой функции WSPStartup/NSPStartup поставщика.	
16#00002af9 11001	Хост не найден - указанный хост неизвестен.	
16#0000276d 10093	Не выполнен вызов функции WSAStartup - либо приложение еще не сделало вызов WSAStartup, либо вызов WSAStartup завершился неудачей.	
16#00002afc 11004	Верное имя, но отсутствует запись данных запрошенного типа - запрошенное имя является верным и было найдено в базе данных, но оно не имеет корректных связанных данных.	
16#00002afb 11003	Невосстановимая ошибка - при просмотре базы данных произошла невосстановимая ошибка.	
16#0000277b 10107	Сбой системного вызова - возникает при сбое системного вызова, который должен всегда выполняться.	
16#0000276b 10091	Сетевая подсистема недоступна - низлежащая система, предоставляющая сетевые службы, недоступна.	
16#00002afa 11002	Неавторизованный хост не найден - временная ошибка, возникающая в процессе разрешения имени и означающая, что локальный сервер не получил ответа от авторитетного сервера.	
16#0000276a 10092	Неверная версия файла WINSOCK.DLL - текущая реализация Windows Sockets не поддерживает версию спецификации, запрашиваемую приложением.	

Код ошибки	Значение	Предлагаемое решение
16#00002775 10101	Процесс аккуратного закрытия - удаленная сторона инициировала процедуру аккуратного закрытия соединения.	
16#00002778 10104	Неверная таблица процедур от поставщика услуг - поставщик услуг вернул неверную таблицу процедур в WS2_32.DLL.	
16#00002779 10105	Неверный номер версии поставщика услуг - поставщик услуг вернул номер версии, отличный от 2.0.	
16#00002733 10035	Ресурс временно недоступен - должен быть выполнен повтор операции.	
16#00000006 6	Неверный указатель объекта события - приложение пытается использовать объект события, но заданный указатель неверен.	
16#00000008 8	Недостаточно памяти - функция Win32 Socket указывает на недостаток ресурсов памяти.	
16#00000057 87	Недопустимые параметры - функция Win32 Socket указывает на проблему с одним или несколькими параметрами.	
16#000003e3 995	Перекрывающаяся операция завершена преждевременно - перекрывающая операция была отменена в связи с закрытием сокета.	
16#000003e4 996	При восместном доступе к объекту ввода-вывода объект не находился с свободном состоянии. Приложение пыталось определить состояние операции совместного доступа, которая еще не была завершена.	
16#000003e5 997	Перекрывающиеся операции будут завершены позже - приложение инициировало перекрывающуюся операцию, которая не может быть завершена немедленно.	

F. Характеристики TCP/IP

F.1. Количество возможных TCP/IP-соединений

Примечание

Количество TCP/IP-соединений, которое возможно использовать в ibaLogic, зависит от настроек системы XP: "TcpNumConnections".

Отрывок из данных Microsoft по этому вопросу:

"**TcpNumConnections**

Реестр:

`HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
Ключевое слово: TcpNumConnections

Тип данных	Диапазон	Значение по умолчанию
REG_DWORD	0x0 – 0xFFFFFE	0

Описание

Определяет максимальное количество TCP-подключений, которые могут быть открыты одновременно. Если здесь указано значение 0, то ни одно соединение не может быть установлено.

Примечание

Windows не добавляет этот элемент в реестр. Этот элемент может быть добавлен при редактировании реестра или использовании программы, которая редактирует реестр.

ССЫЛКА

http://technet.microsoft.com/en-us/library/cc938216.aspx

F.2. Проблема задержки подтверждения

Проблема

Измерения, выполняемые автоматизированным оборудованием с помощью TCP/IP, невозможны для циклов < 200 мс.

Образец ошибки: ошибка последовательности, неполные телеграммы и различная длина полученных сообщений.

Причина

Существуют различные варианты обработки "Подтверждения" в контексте протокола TCP/IP:

1. Стандартный интерфейс программного программирования WinSocket работает в соответствии с RFC1122, используя механизм "задержки подтверждения" ("delayed acknowledge"). Подтверждение не отправляется до тех пор, пока не будут получены остальные телеграммы. После чего будет отправлено общее для всех телеграмм подтверждение. При отсутствии других телеграмм ACK-телеграмма будет отправлена самое позднее через 200 мс (в зависимости от сокета).

В соответствии со стандартом TCP/IP это возможно, поскольку при управлении потоком данных с помощью "Sliding Window" (параметр Win = nnnn) получатель указывает количество байтов, которое может быть получено без отправки подтверждения.

2. Некоторые типы контроллеров не поддерживают такой режим, а ожидают подтверждения после каждой телеграммы данных. Если подтверждение не получено в течение определенного времени (200 мс), контроллер выполняет повторную отправку телеграммы, иногда с добавлением новых данных. Это приводит к ошибкам на стороне получателя, поскольку предыдущая телеграмма была получена корректно.

Устранение

Нужно отключить "задержку подтверждения" путем добавления параметра в реестр Windows:

"TcpAckFrequency" REG_DWORD = 1;

По умолчанию этот параметр отсутствует, его необходимо ввести, используя следующий путь:

"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{InterfaceGUID}"

Примечание

Необходимо выбрать правильный интерфейс. Проверить правильность выбранного интерфейса можно, например, посмотрев, какие IP-адреса установлены в настоящий момент.

Чтобы получить дополнительную информацию по этому вопросу, посетите следующую страницу http://support.microsoft.com/kb/328890

Windows XP:

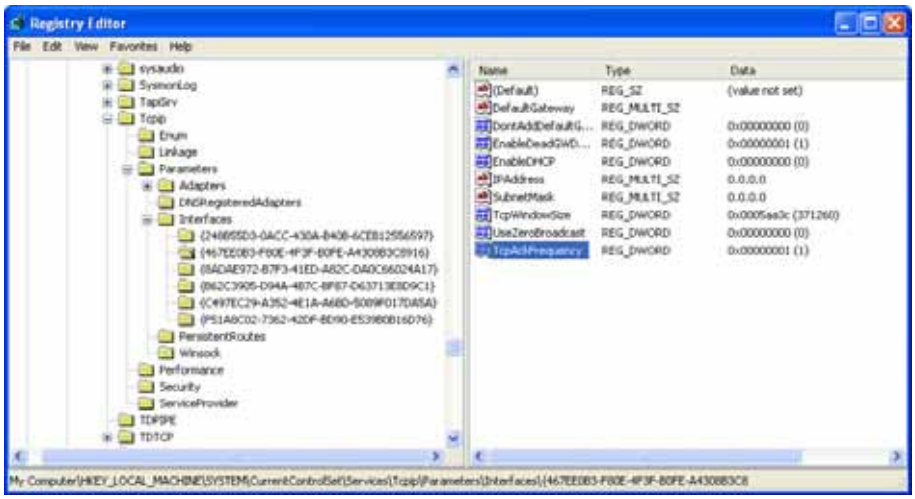


Рис. 146: Реестр Windows XP

G. Сочетания клавиш

G.1. Клиент

Сочетание клавиш	Объяснение
<Ctrl> + <P>	Печать
<Ctrl> + <Z>	Отмена действия
<Ctrl> + <Y>	Повтор действия
<Ctrl> + <C>	Копировать
<Ctrl> + <V>	Вставить
<Ctrl> + <A>	Выбрать все
<Del>	Удалить
<F5>	Запустить вычисление
<Shift> + <F5>	Остановить вычисление
<Ctrl> + <Shift> + <F>	Добавить – новый функциональный блок
<Ctrl> + <Shift> + <M>	Добавить – новый макрос
<Ctrl> + <Shift> + <I>	Добавить – новый внутристраничный коннектор
<Ctrl> + <Shift> + <T>	Добавить – новый коннектор вне задачи
<Ctrl> + <Shift> + <C>	Добавить – новый комментарий
<Ctrl> + <Shift> + <S>	Отобразить коннекторы вне задачи
<F1>	Помощь

G.2. Функции мыши в области программы

Сочетание клавиш	Объяснение
Щелчок левой кнопкой мыши + <Shift>	Выделение нескольких элементов
Щелчок левой кнопкой мыши + <Ctrl>:	Переключить выделение
Колесо прокрутки + <Shift>	Передвинуть область обзора влево / вправо.
Колесо прокрутки + <Ctrl>	Приближение / отдаление
Колесо прокрутки + <Alt>	Передвинуть область обзора вверх / вниз.
<Ctrl> + вход или выход коннектора	Создать IPC
Drag & Drop + <Alt>	Нажав и удерживая клавишу <Alt>, вы можете перетащить сигналы из коннектора в окно ibaPDA Express.

G.3. ibaPDA Express

Сочетание кнопок	Объяснение
<F6> (переключение)	Запускает непрерывное отображение с текущего значения времени. Кнопка активна, если нажата кнопка "Остановить прокрутку".
<F6> (переключение)	Останавливает непрерывное отображение. После нажатия этой кнопки на графике появляется линейка, которую можно перемещать курсором мыши и измерять с ее помощью кривые. Значения сигналов приводятся в комментариях к графику. Ось X можно передвигать с помощью курсора мыши. Таким образом можно просматривать значения за более ранние периоды времени. Кнопка активна, если запущено отображение сигналов.
<F5>	Автоматическое масштабирование
<F3>	Кнопка активна, если масштаб отображения был изменен. Возвращает отображение к предыдущему коэффициенту масштабирования (уменьшает).
<F4>	Кнопка активна, если масштаб отображения был изменен. Возвращает к исходному (автоматически заданному) масштабу отображения.
<F10>	Выход из полноэкранного режима.

H. Список сокращений

Сокращение	Немецкий	Английский	Русский
ASCII	Zeichenkodierung	American Standard Code for Information Interchange	Американская стандартная кодировочная таблица для печатных символов и некоторых специальных кодов
AWL	Anweisungsliste (Siehe STL)	Statement List (see STL)	Список операторов (см. STL)
CE	Übereinstimmung mit EU-Richtlinien	FR.: Conformité Européenne	FR.: Conformité Européenne – Европейское соответствие
CFC	Funktionsblockdiagramm	Continuous Function Chart	Функциональный план
CPU	Prozessor	Central Processing Unit	Центральный процессор
CR	Wagenrücklauf	Carriage Return	Возврат каретки
CSV		Comma Separated Values or Character Separated Values	Значения, разделенные запятыми, или значения, разделенные символами
DA	Datenzugriff	Data Access	Доступ к данным
DCOM		Distributed Component Object Model	Распределенная компонентная объектная модель
DFW		DAT_FILE_WRITE	DAT_FILE_WRITE
DIN	Deutsches Institut für Normung		Институт стандартизации ФРГ
DLL	Dynamische Verbindungsbibliothek	Dynamic Link Library	Динамически подключаемая библиотека
E/A	Eingang/Ausgang	INPUT/OUTPUT	Вход/выход
FB	Funktionsbaustein	Function Block	Функциональный блок
FBD	Funktionsblockdiagramm	Function Block Diagram	Диаграмма функциональных блоков
FFT		Fast Fourier Transformation	Быстрое преобразование Фурье
FOB	Lichtwellenleiterkarte	Fiber optical board	Оптическая карта
FUP	Funktionsplan	Function Chart	Функциональная схема
GDM		Global Data Memory	Память глобальных данных
GMT		Greenwich Mean Time	Гринвичское среднее время
GSD	Gerätestammdaten-Datei (Profibus)	Generic Station Description (Profibus)	Описание устройства общего вида (Profibus)
HMI	Bedien-Beobachten-System	Human Machine Interface	Человеко-машинный интерфейс
HW	Hardware	Hardware	Аппаратное обеспечение
I/O	Eingang/Ausgang	Input/Output	Вход/выход
IEC	ein Normungsgremium für Elektrotechnik	International Electrotechnical Commission	Международная электротехническая комиссия

Сокращение	Немецкий	Английский	Русский
IL	Anweisungsliste	Instruction List	Технические требования
IP	Internet-Protokoll	Internet Protocol	Internet-протокол
IPC	Intra-Page-Konnektor	Intra-page connector	Внутристраничный коннектор
KOP	Kontaktplan (Siehe LAD)	Ladder Diagram (See LAD)	Многозвенная логическая схема (См. LAD)
LAD	Kontaktplan (Siehe KOP)	Ladder Diagram (See KOP)	Многозвенная логическая схема (См. KOP)
LF	Zeilenvorschub	Line Feed	Перевод строки
LWL	Lichtwellenleiter	Fiber optical conductor	Оптоволоконный кабель
MB	Makroblock	Macro Block	Макрос
MDAC		Microsoft Data Access Components	Компоненты доступа к данным Windows DAC
MS SQL		Microsoft Structured Query Language	Microsoft Structured Query Language
NL	Zeilenvorschub	Newline	Перевод строки
OLE	Objekt-Verknüpfung und -Einbettung	Object linking and embedding	Связывание и встраивание объектов
OPC	Datenaustauschprotokoll (Schnittstelle)	OLE for Process Control	OLE для управления технологическим процессом
OTC	Off-Task-Konnektor	Off-task connector	Коннектор вне задачи
PAC	Programmierbare Automatisierungseinheit	Programmable automation controller	Программируемый контроллер автоматизации
PADU	Parallel-Analog-Digital-Umsetzer (Varianten: PADU-S, PADU-S-IT)	Parallel analog digital unit	Параллельный аналого-цифровой преобразователь
PC	Einzelplatzrechner	Personal Computer	Персональный компьютер
PCI	PC-Bussystem	Peripheral Component Interconnect	Локальная шина соединения периферийных устройств
PDA	Prozess-Daten-Aufzeichnung	Process data acquisition	Сбор технологических данных
PLC	Siehe SPS	Programmable Logic Controller	Программируемый логический контроллер
PMAC	Programmierbare Mess- und Automatisierungseinheit	Programmable Measurement and Automation Controller	Программируемый контроллер измерения и автоматизации
RAM	Arbeitsspeicher	Random Access Memory	Оперативная память
RFM		Reflective Memory	Reflective Memory
SD	SIMADYN D	SIMADYN D	SIMADYN D
SPS	Speicherprogrammierbare Steuerung	See PLC	См. PLC
SST	Profibuskarte		Плата Profibus
ST	Strukturierter Text	Structured Text	Структурированный текст

Сокращение	Немецкий	Английский	Русский
STL	Anweisungsliste (Siehe AWL)	Statement List (See AWL)	Список операторов (см. AWL)
SW	Software	Software	Программное обеспечение
Tab	Tabulator	Tabulator	Клавиша табуляции
TCP/IP	Netzwerk-Protokoll	Transmission Control Protocol/Internet Protocol	Протокол управления передачей / Internet-протокол
TDC	Automatisierungssystem (Siemens)		Система автоматизации (Siemens)
UCODE	Bytecode	Bytecode	Байтовый код
USB		Universal Serial Bus	Универсальная последовательная шина
UTC	koordinierte Weltzeit	Universal Time Coordinated	Универсальное координированное время
WDM		Windows Driver Model	Модель драйверов Windows
XML		Extensible Markup Language	Язык XML
XP	Windows XP	Windows XP	Windows XP

Алфавитный указатель

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

А

Б

В

Г

З

И

К

P

C

T

У

Ф

Ч

Э

Я

I. Техническая поддержка и контактная информация

Техническая поддержка

Тел.: +49 911 97282-14

Факс: +49 911 97282-33

E-mail: support@iba-ag.com

Note

При обращении в службу техподдержки, сообщайте, пожалуйста, серийный номер (iba-S/N) продукта.

Контактная информация

Центральный офис

iba AG

Koenigswarterstrasse 44

90762 Fuerth

GERMANY

Тел.: +49 911 97282-0

Факс: +49 911 97282-33

E-mail: iba@iba-ag.com

Конт. лицо: Harald Opel

По всему миру и в регионах

Контактную информацию касательно вашего местного представителя или представительства компании iba вы можете найти на нашем сайте:

www.iba-ag.com.