# Руководство программиста по созданию программ для ПЛК

**(Версия 1.0.03)**

**Апрель 2018**

## Содержание

# 1 Назначение документа

Настоящий документ предназначен для разработчиков программ электроавтоматики для системы ЧПУ «АксиОМА Контрол». Документ содержит описание базовых принципов построения и функционирования программно-реализованного логического контроллера (ПЛК) и описание среды разработки программ для него в соответствии со стандартом МЭК 61131.

Примеры, представленные в тексте, не предназначены для практического исполнения. Они служат для пояснения принципов программирования и моделируют практические ситуации. Примеры программирования даны на языке FBD.

Разработчик оставляет за собой право вносить изменения по мере развития системы.

## 1.1 Список сокращений

В настоящем руководстве использованы следующие сокращения:

- ПЛК – программируемый логический контроллер.
- PLC – Programmable Logic Controller.
- SoftPLC – Soft Programmable Logic Controller (программно-реализованный логический контроллер).
- FBD – Functional Block Diagram (язык функциональных блоков).
- ЧПУ – числовое программное управление.

# 2 Базовые принципы построения и функционирования программно-реализованного котроллера

В условиях современного автоматизированного производства наметилась устойчивая тенденция решения логической задачи управления технологическим оборудованием в рамках общего программного обеспечения систем управления без привлечения дополнительной аппаратуры и системного программного обеспечения программируемых логических контроллеров (ПЛК).

Описанный в настоящем руководстве контроллер по своему принципу построения является программно-реализованным, т.е. подразумевает замену аппаратного ПЛК на программно-математическое обеспечение. Схема управления некоторым объектом с использованием программно-реализованного контроллера представлена на Рисунок 1 и требует наличия персонального компьютера с установленным комплексом SoftPLC, пассивных устройств ввода/вывода и непосредственно объекта управления.

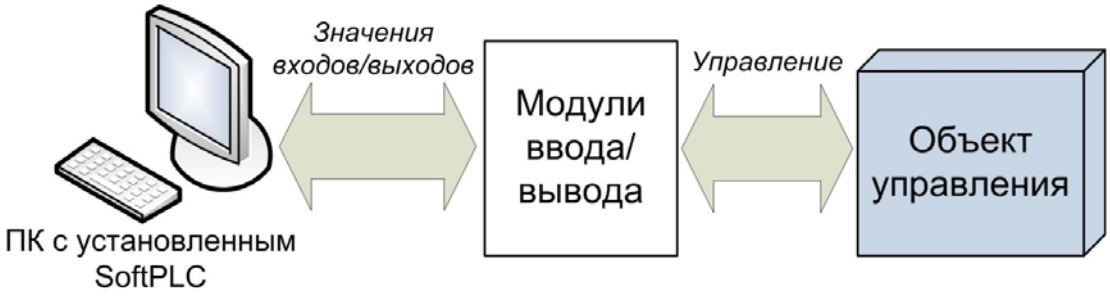



**Рисунок 1 – Схема управления объектом с помощью программно-реализованного контроллера (SoftPLC)**

Программно-реализованный контроллер (SoftPLC) может функционировать как в виде автономного решения, так и встроенного в систему ЧПУ.

В системе ЧПУ «АксиОМА Контрол» он реализован как часть общего программно-математического обеспечения системы и решает логическую задачу управления (Рисунок 2). Программное обеспечение виртуального программно-реализованного контроллера работает в одной операционной среде с программным обеспечением системы ЧПУ, за счёт чего достигается наибольший положительный эффект от его применения. Особенностями контроллера являются: аппаратная и платформенная независимость, отсутствие платной лицензии, а также наличие возможностей отладки управляющих программ. Контроллер имеет полный набор функциональности для решения задач автоматизации различной сложности. Выбранный подход построения контроллера позволяет обеспечить:

- работу контроллера в рамках кроссплатформенного ядра системы;
- открытость на уровне удаленных модулей ввода/вывода (возможность применения модулей, работающих по различным стандартным протоколам, в том числе SERCOS, EtherCAT, Modbus);
- уменьшение времени пуско-наладочных работ;
- механизм открытости для станкостроителя.

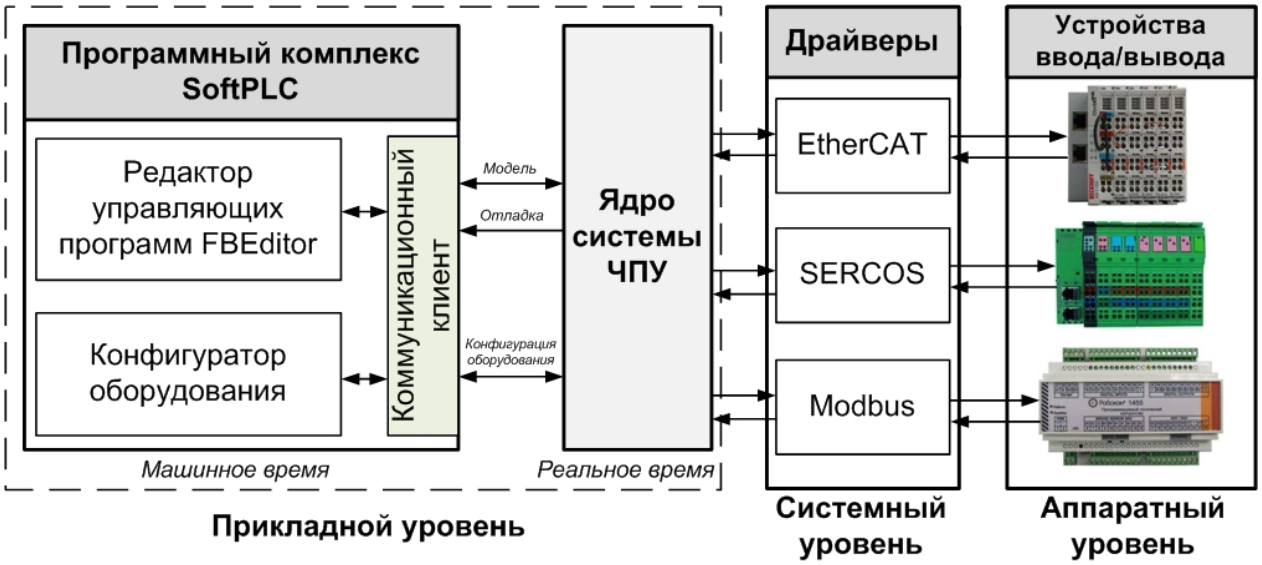



**Рисунок 2 – Архитектура программно-реализованного контроллера**

Работа управляющей программы и все математические вычисления производятся в программном ядре системы ЧПУ. Также ядро производит управление аппаратными входами/выходами различных типов (аналоговые, дискретные), которые, в свою очередь, управляют аппаратной частью. Стоит отметить, что контроллер может работать в «офлайн» режиме, т.е. без использования аппаратных средств.

Прикладной инструментарий контроллера включает в себя программный комплекс, содержащий три основных инструмента:

- среда разработки и отладки управляющих программ (FBEditor);
- конфигуратор устройств;
- клиент взаимодействия с ядром контроллера (системы ЧПУ).

Среда разработки и отладки управляющих программ FBEditor является универсальным инструментом для создания, визуализации и отладки ПЛК-программ и адаптирована для работы с аппаратной частью различных производителей. Программирование логической задачи управления осуществляется в среде проектирования и разработки управляющих программ на языке функциональных блоков (Function Block Diagram, FBD), являющимся одним из языков программирования стандарта МЭК 61131.

С помощью конфигуратора устройств производится настройка конфигурации подключенного оборудования для корректной работы с ним.

# 3 Редактор программ ПЛК

## 3.1 Запуск программного пакета и подключение к вычислительному ядру

В системе ЧПУ «АксиОМА Контрол» редактор программ ПЛК запускается по нажатию клавиши S1 «PLC редактор» на стартовом экране системы (Рисунок 3).

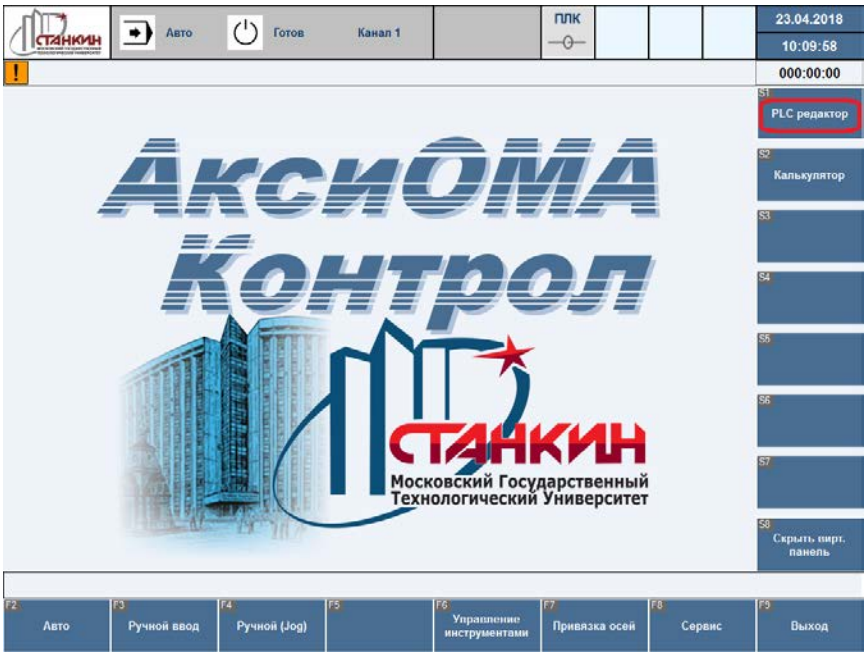



**Рисунок 3 – Запуск редактора программ ПЛК в системе ЧПУ «АксиОМА Контрол»**

Базовый экран при запуске редактора FBEditor представлен на рисунке Рисунок 4. Вместе с редактором производится автоматический запуск клиента для подключения к вычислительному ядру контроллера (системы ЧПУ). Это необходимо для запуска ПЛК-программ и возможностей их отладки.

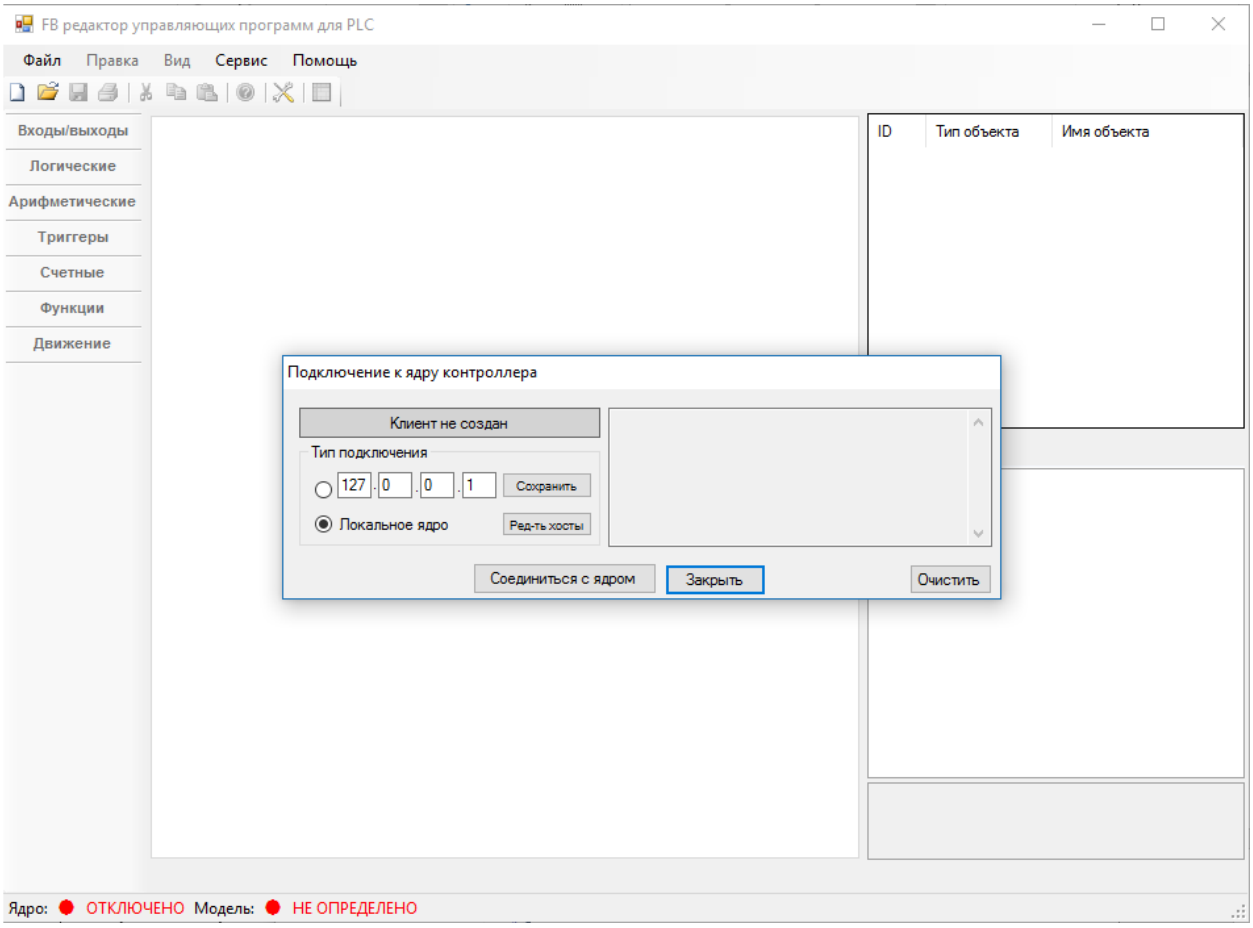



**Рисунок 4 – Общий вид программного пакета при запуске**

В открытом диалоговом окне необходимо указать тип подключения к ядру контроллера: удалённое или локальное. Для подключения к удалённому ядру нужно указать IP-адрес компьютера, на котором оно работает, и выбрать пункт слева от строк ввода IP-адреса. Кнопка «Сохранить» позволяет запомнить указанный IP-адрес и при последующих запусках редактора снова указывать его. Для подключения к локальному ядру, т.е. приложению ядра, запущенном на этом же компьютере, необходимо выбрать «Локальное ядро», или в графе IP-адреса указать адрес «127.0.0.1» (выбирается по умолчанию).

Если не требуется производить запуск ПЛК-программ и отлаживать их, можно просто закрыть окно клиента и производить операции с ПЛК-программой в офлайн-режиме.

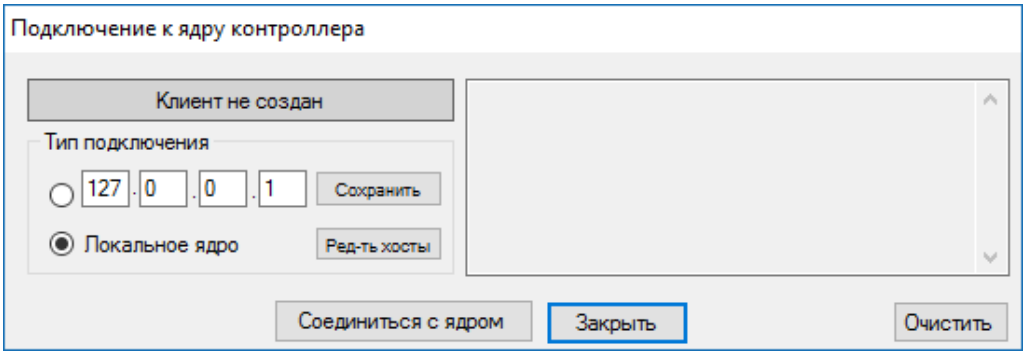



**Рисунок 5 – Встроенный клиент подключения к вычислительному ядру контроллера**

После указания типа подключения нужно нажать кнопку «Соединиться с ядром». По нажатию на неё производится активация клиента взаимодействия, статус которого отображается в информационном поле справа. По истечении некоторого времени (5-15 сек) в цветовой графе отображается результат подключения. При отсутствии подключения статусная строка подсвечивается красным цветом с соответствующим сообщением (Рисунок 6).

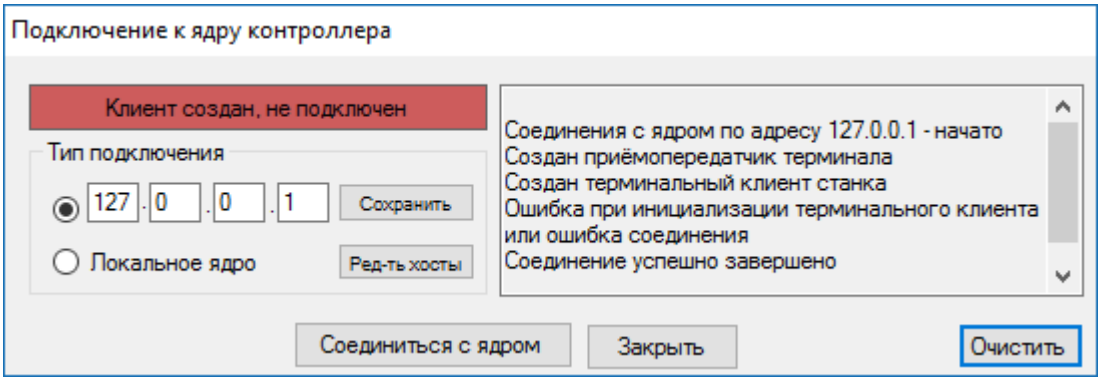



**Рисунок 6 – Вид окна клиента подключения при неудачном соединении с ядром**

Если клиенту удалось подключиться к ядру, то статусная строка подсвечивается зелёным с соответствующим сообщением, после чего следует закрыть клиент подключения, нажав кнопу «Закрыть».

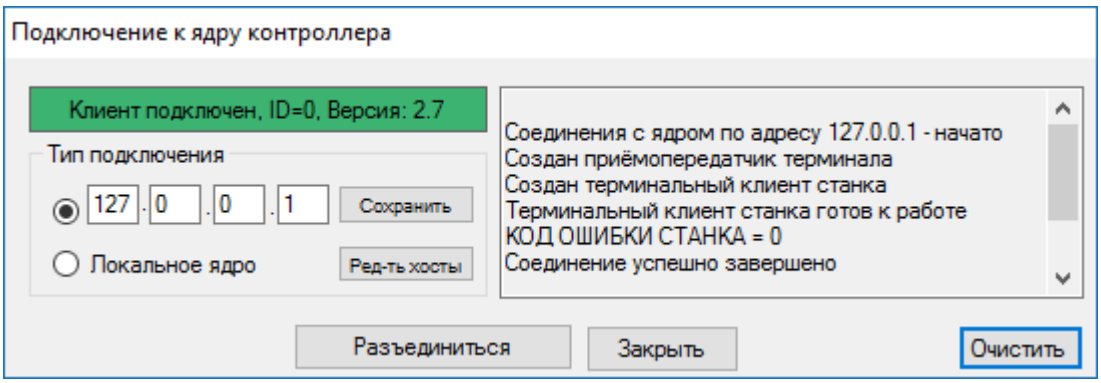



**Рисунок 7 – Вид окна клиента подключения при успешном соединении с ядром**

## 3.2 Создание/загрузка проекта

После завершения работы с клиентом подключения в редакторе открывается диалоговое окно создания/загрузки файла программы, которая может быть двух типов:

- проект – программа электроавтоматики;
- библиотека для создания пользовательских блоков.

Для создания нового проекта необходимо выбрать его тип, ввести имя, выбрать расположение и нажать кнопку «Ок». Для открытия существующего файла нужно нажать соответствующую кнопку и выбрать файл в проводнике.

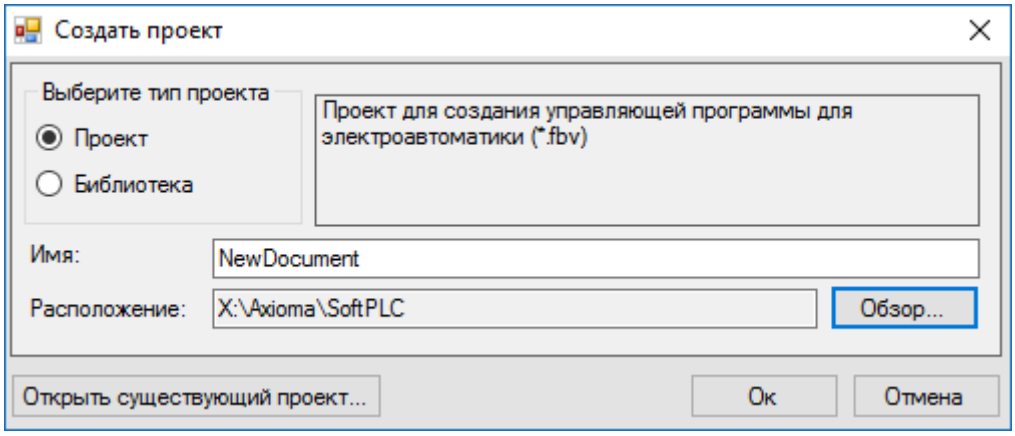



**Рисунок 8 – Диалоговое окно создания/загрузки проекта**

После этого открывается основная среда программирования ПЛК (FB редактор), имеющая широкие возможности для разработки, документирования, отладки и верификации управляющих программ.

## 3.3 Базовый интерфейс редактора программ ПЛК

Интерфейс среды разработки управляющих программ (Рисунок 9) позволяет производить разработку управляющей программы, настройку параметров функциональных блоков при помощи панели настроек, а также отладку программы.

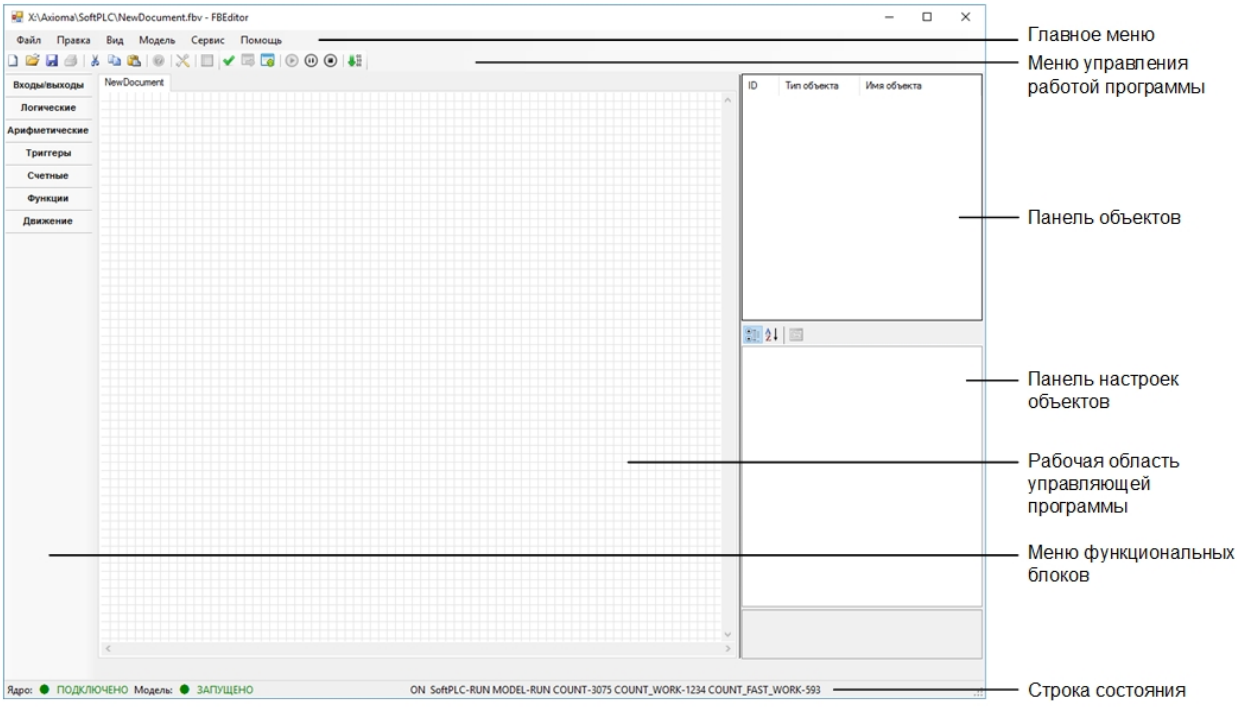



**Рисунок 9 – Интерфейс основной среды разработки программ ПЛК**

Интерфейс содержит следующие основные компоненты:

- главное меню;
- меню работы с файлами управления работой программы;

- панель объектов – список с добавленными на данный момент объектами, с помощью которого можно быстро находить необходимые объекты и производить их выбор;
- панель настроек объектов, служащая для настройки параметров функциональных блоков, а также объектов входов/выходов;
- рабочая область управляющей программы – главная часть интерфейса, содержащая визуализацию программы ПЛК;
- меню функциональных блоков – панель, содержащая кнопки, соответствующие различным функциональным блокам, которыми можно оперировать при создании управляющей программы;
- строка состояния редактора – информационная панель для отображения текущего состояния подключения к ядру системы, а также статуса работы программы.

Основное меню программы содержит весь функционал, которым обладает редактор для создания и отладки программ ПЛК. Подробное описание функций меню представлено в Таблица 1.

**Таблица 1 – Описание функций меню редактора**

| Пункт меню | Команда | «Горячая» клавиша | Описание |
|---|---|---|---|
| Файл | Создать | Ctrl-N | Создание нового проекта или пользовательской библиотеки |
| | Открыть | Ctrl-O | Открытие ранее сохраненного проекта или библиотеки |
| | Сохранить | Ctrl-S | Сохранение проекта или библиотеки |
| | Сохранить как… | Ctrl-Shift-S | Сохранение проекта или библиотеки в указанное месторасположение |
| | Импорт пользовательской библиотеки | – | Добавление пользовательской библиотеки для использования его в открытой программе (см. раздел 3.1) |
| | Настройки документа | – | Открытие диалогового окна с настройками документа |
| | Выход | Alt-F4 | Выход из комплекса программирования |
| Правка | Отменить | Ctrl-Z | Отмена последнего выполненного действия. |
| | Повторить | Ctrl-R | Повтор последнего отмененного действия |
| | Вырезать | Ctrl-X | Вырезать (скопировать в буфер обмена и удалить) выделенные объекты |
| | Копировать | Ctrl-C | Копирование выделенных объектов в буфер обмена |
| | Вставить | Ctrl-V | Вставка объектов, находящихся в буфере обмена, в модель |
| | Удалить | Del | Удаление выделенных объектов |
| | Очистить | Ctrl+Shift+L | Очистка открытого документа |
| | Выделить все | Ctrl-A | Выделение всех объектов в открытом |

| Пункт меню | Команда | | «Горячая» клавиша | Описание |
|---|---|---|---|---|
| | | | | документе |
| | На передний план | | – | Перемещение функционального блока на передний план по отношению к остальным |
| | На задний план | | – | Перемещение функционального блока на задний план по отношению к остальным |
| Вид | Масштаб | 20% | – | Выбор фиксированного масштаба отображения проекта – 20% |
| | | 50% | – | Выбор фиксированного масштаба отображения проекта – 50% |
| | | 100% | – | Выбор фиксированного масштаба отображения проекта – 100% |
| | | 200% | – | Выбор фиксированного масштаба отображения проекта – 200% |
| | | 400% | – | Выбор фиксированного масштаба отображения проекта – 400% |
| | | Подогнать под размер окна | – | Автоматическая подгонка масштаба открытого проекта под размер окна |
| | Упорядочить блоки и связи | | – | Исправление ошибок с отрисовкой связей между блоками и выравнивание блоков по сетке (обычно используется при открытии файлов проектов, сохранённых в более старых версиях редактора) |

## 3.4 Работа с функциональными блоками

Программно-реализованный логический контроллер программируется удобным для пользователя методом комбинирования специально предназначенных для этого функциональных блоков. Задача разбивается на несколько этапов, на каждом из которых могут быть задействованы несколько блоков. Такой способ программирования упрощает разработку. Функциональные блоки запрограммированы для выполнения определённых задач, но в то же время обладают достаточной гибкостью для удовлетворения индивидуальных потребностей. Пользователь может собирать сложное устройство простыми малыми шагами, начиная от входа и продвигаясь дальше согласно логике работы. Контроллер производит сбор, обработку информации и обеспечивают программу необходимыми функциями управления в соответствии с алгоритмом работы системы.

Разработка ПЛК-программ в редакторе производится на языке Functional Block Diagram (FBD). Программа на языке FBD внешне напоминает функциональную схему логического устройства – совокупность элементов (блоков), входы и выходы которых соединены линиями связи. Каждый блок программы, представляющий собой некоторую функцию, может иметь в общем случае произвольное количество входов и выходов. Начальные значения переменных задаются с помощью специальных входов или констант, выходные цепи могут быть связаны либо с физическими выходами контроллера, либо с глобальными переменными программы. Связи, соединяющие выходы одних блоков с входами других, являются переменными программы и служат для пересылки данных между блоками. Среди основных преимуществ FBD можно отметить его визуальную

наглядность и возможность наиболее удобной отладки программы, поскольку данный язык является полностью графическим.

Добавление функциональных блоков производится по клику на нужном объекте в панели объектов. Для удобства все функциональные блоки разделены на следующие группы: входы/выходы, логические, арифметические, триггеры, счетные, функции, движение. Более подробное описание всех блоков см. в разделе 4 «Описание функциональных блоков».

Открытие нужной группы производится по клику левой кнопкой мыши на её названии в панели объектов. Затем для добавления нужного блока следует один раз кликнуть по его названию. Функциональный блок по умолчанию добавляется в верхний левый угол экрана редактора. На Рисунок 10 показано добавление функционального блока, реализующего функцию логического «И».

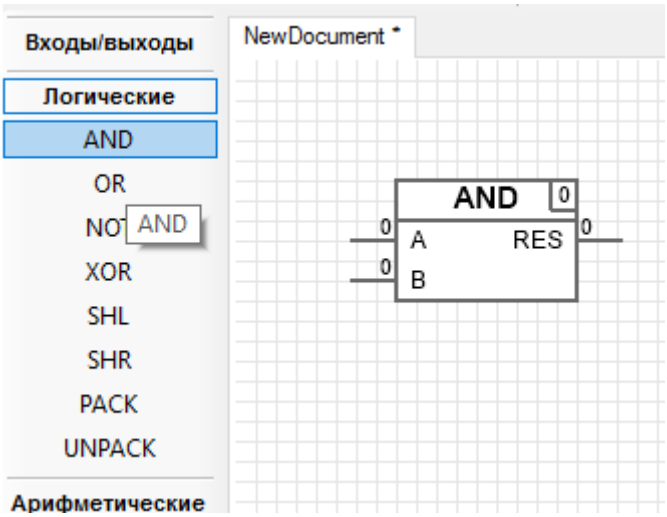



**Рисунок 10 – Добавление функционального блока AND в программу**

Чтобы выполнить любую операцию с блоком (перемещение, удаление, настройка параметров), его необходимо выбрать (выделить) однократным нажатием левой кнопки мыши по блоку, цвет блока при этом изменяется.

Для выделения нескольких блоков необходимо последовательно кликать на блоках, зажав клавишу «Ctrl» на клавиатуре. Также можно в пустой области рабочего поля зажать левую кнопку мыши и, не отпуская её, выбрать область, в которой находятся блоки, которые необходимо выделить.

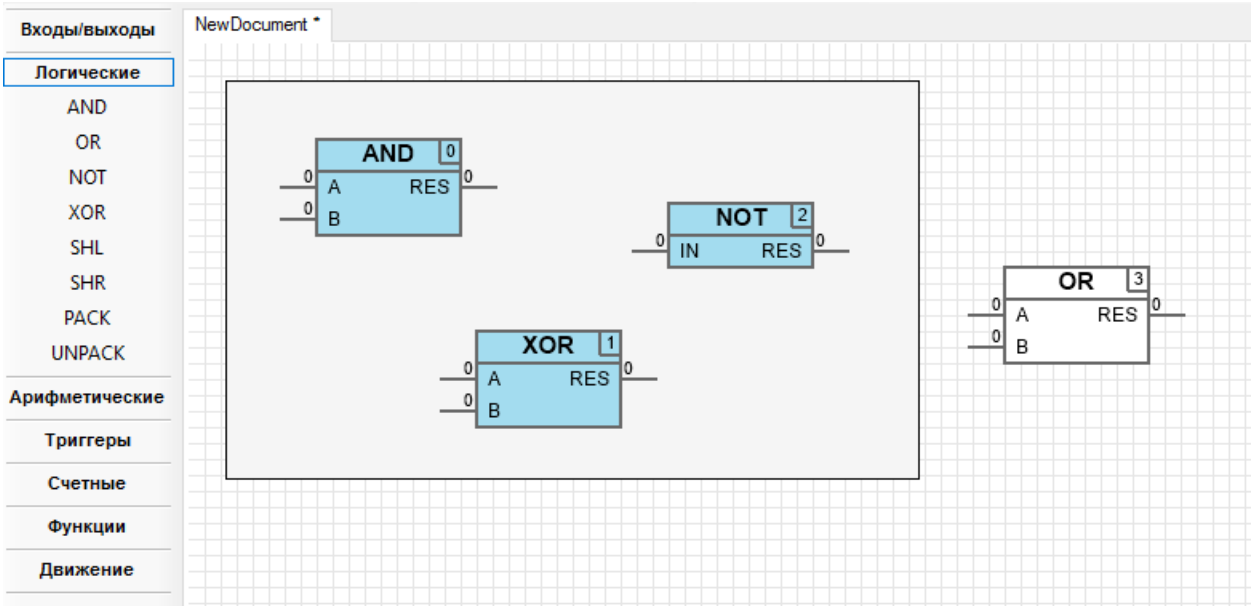



**Рисунок 11 – Множественное выделение функциональных блоков**

Перемещение блока (блоков) осуществляется путём их выделения и последующего нажатия на клавиши стрелок на клавиатуре (вверх, вниз, влево, вправо) или путём перетаскивания их с нажатой левой кнопкой мыши по рабочей области редактора.

Чтобы удалить блок (блоки), необходимо выделить их и нажать клавишу «Del» на клавиатуре, либо выбрать пункт меню «Удалить» в меню «Правка».

Для наглядности представления программы в редакторе имеется возможность перемещать блок на передний/задний планы относительно других объектов программы. Это реализуется путём выбора соответствующих пунктов в контекстном меню блока.

Все функциональные блоки имеют параметры, доступные пользователю, которые дают возможность корректно запрограммировать работу для любых случаев. Их можно изменять с помощью панели настроек функциональных блоков, располагающейся в правой нижней части экрана. У большинства функциональных блоков набор параметров идентичен, и включает в себя основные параметры, графические параметры и параметры блока. Редактируемыми являются только параметры «Имя объекта» и «Описание объекта», находящиеся в основных параметрах. В имени объекта может быть задана строка длиной до 15 символов, которая отображается над блоком и позволяет оператору лучше ориентироваться в программе. Описание объекта представляет собой более подробное описание блока, отображающееся только в панели параметров. Все это позволяет именовать и описывать функциональные блоки так, чтобы программа была понятна различным пользователям.

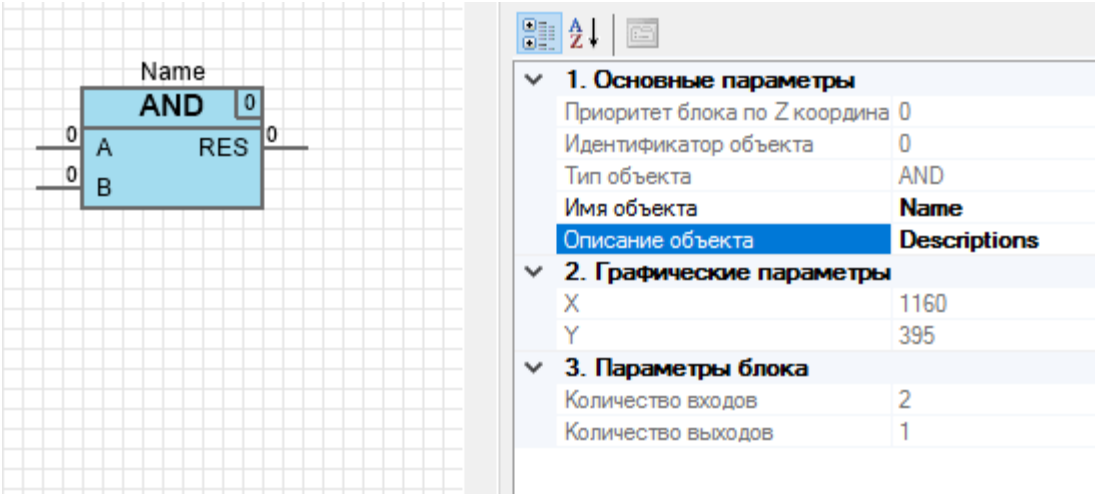



**Рисунок 12 – Создание имени и описания функциональных блоков**

У входов и выходов набор параметров шире, и включает в себя также параметры для настройки их привязки к аппаратным модулям ввода/вывода.

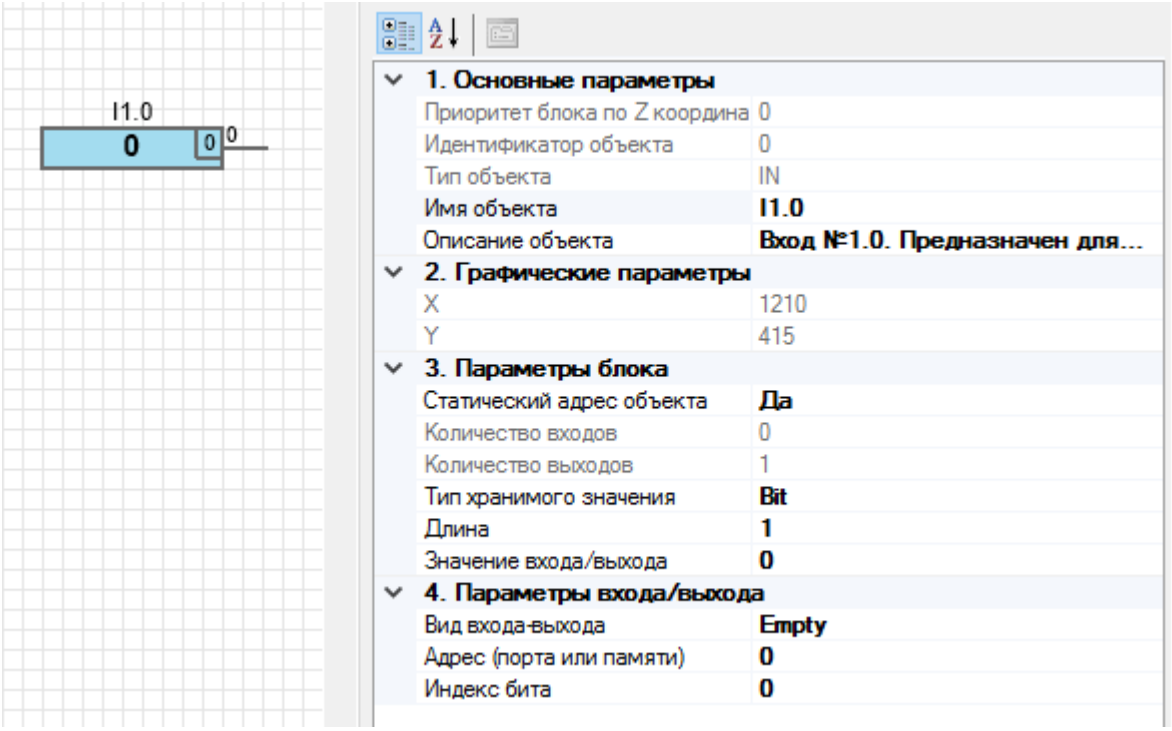



**Рисунок 13 – Настройки входа**

Помимо стандартных вышеописанных настроек у входов/выходов имеются дополнительные параметры, описанные в Таблица 2.

**Таблица 2 – Параметры входов/выходов**

| Параметр | Описание |
|---|---|
| Статический адрес объекта | При указании значения «Да» (по умолчанию) адрес байта, на который ссылается объект, берётся из указанного в настройках (см. ниже). |

| | |
|---|---|
| | При выборе значения «Нет» (т.е., динамического адреса) у блока появляется дополнительный вход, на который можно передать адрес байта. |
| Тип хранимого значения | Тип значения, хранимого в блоке.<br>Возможные значения:<br>- None – не задано;<br>- Bit – булевское число;<br>- Analog – число с плавающей точкой;<br>- Digital – целое число. |
| Длина | Длина хранимого значения. Задается автоматически при задании типа хранимого значения, но есть возможность его изменения для решения нестандартных задач. |
| Значение входа/выхода | Заданное константное значение входа/выхода (применимо только для вида «Empty», см. ниже). |
| Вид входа/выхода | Возможные значения:<br>- Empty – задается для реализации константных значений. Такие входы/выходы не принимают и не передают значений в память ядра.<br>- CommonPlcMemory – ссылка на разделяемую память ядра контроллера. Выбирается для работы с аппаратными модулями ввода/вывода.<br>- LPT_Port – ссылка на LPT порт компьютера. |
| Адрес (порта или памяти) | Номер байта, на который ссылается вход/выход для определения значения или его задания. |
| Индекс бита | Номер бита в указанном байте, в котором хранится значение для дискретных входов/куда передается значение для дискретных выходов |

Практически все логические и арифметические блоки по умолчанию имеют два входа и один результирующий выход. Однако для выполнения операций с большим количеством входных значений имеется возможность добавлять и удалять входы на блоках данного типа. Это реализуется путём выбора в контекстном меню блока пункта «Изменить входы», который отображается только у тех блоков, с которыми логически можно провести данную операцию (Рисунок 14).

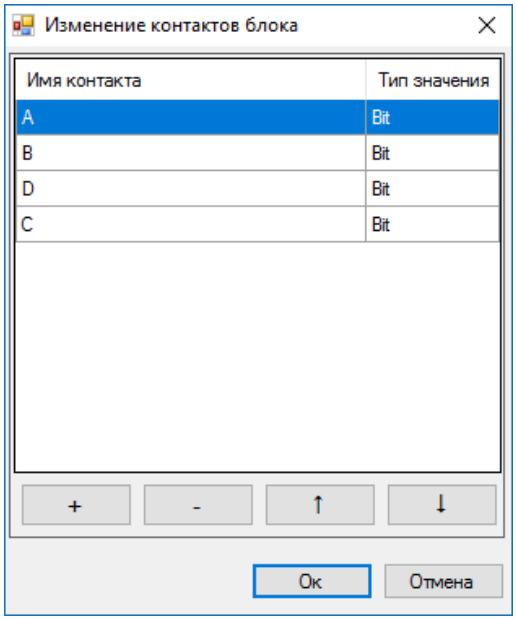


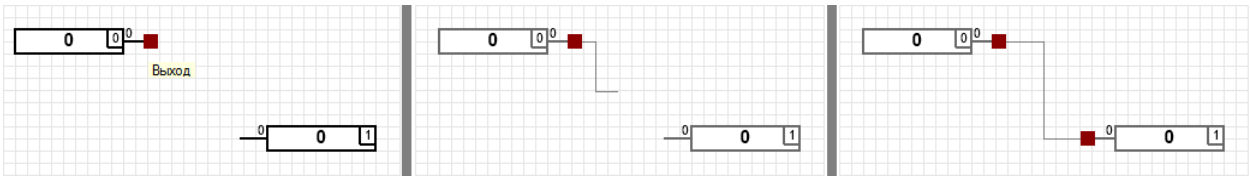

**Рисунок 14 – Изменение входных контактов логических и арифметических блоков**

Добавить и удалить входные контакты можно по нажатию кнопок «+» и «-» соответственно. Стрелки предназначены для перемещения выбранного входа выше или ниже в списке входных контактов. После завершения создания входов необходимо нажать «Ок».

## 3.5 Работа со связями между функциональными блоками

Программирование ПЛК сводится к созданию программы на основе имеющегося набора функциональных блоков, их параметризации и соединению их контактов связями для передачи значений между ними.

Соединение функциональных блоков связями реализуется путём наведения указателя мыши на требуемый контакт блока (входа или выхода) и его ведению с зажатой левой кнопкой мыши до контакта (выхода или входа соответственно), с которым создаётся связь. Связи создаются между выходом и входом блоков, причём на вход всегда подается только одна связь, чтобы избежать неоднозначности работы программы. От одного выхода можно создать любое количество исходящих связей.




*а)* *б)* *в)*

**Рисунок 15 – Этапы создания связи между двумя блоками**

При создании связь по умолчанию имеет две точки привязки к контактам и две точки излома. В редакторе имеется возможность редактировать изломы линий связи, что позволяет избегать наложения линий связи на функциональные блоки и сделать

программу более наглядной и удобной. Для этого следует выделить связь путём одиночного клика мышью в любом месте нужной связи и потянуть за одну из точек излома вверх или вниз.

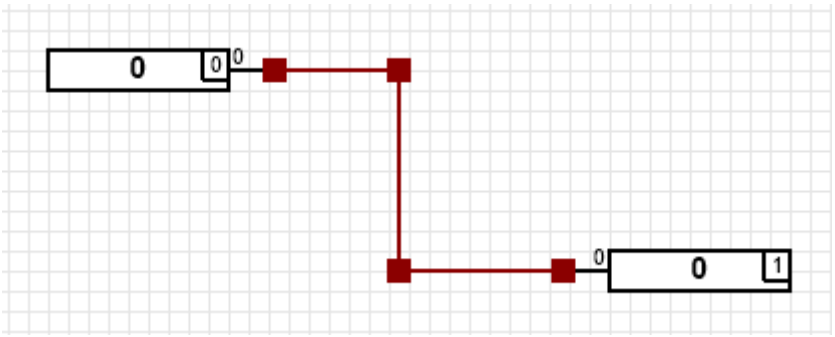



**Рисунок 16 – Выделенная связь между блоками**

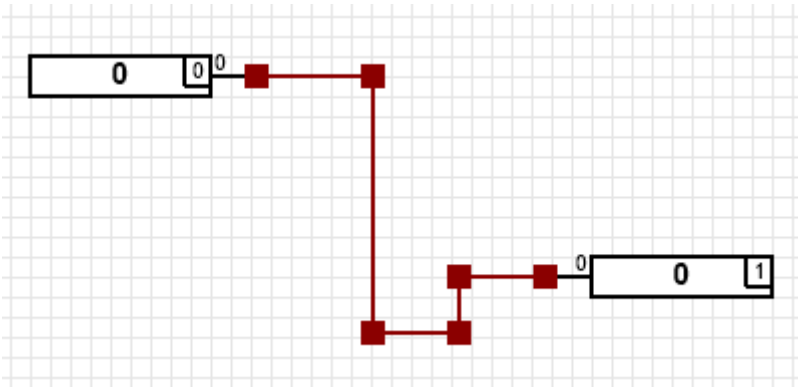



**Рисунок 17 – Создание дополнительного излома у связи**

При этом добавится точка излома по центру горизонтальной части связи, от которой производится её «оттягивание» мышью. Таким образом, можно создавать неограниченное количество изломов у связи для их наглядного отображения в программе и избежания пересечений с другими элементами программы.

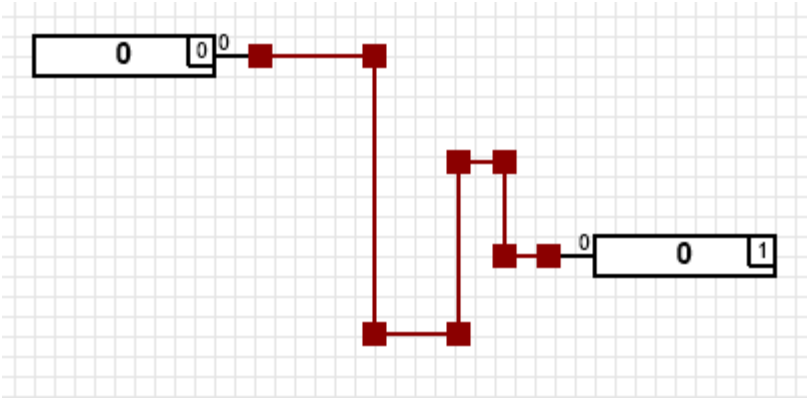



**Рисунок 18 – Создание дополнительного излома у связи**

Для удаления какого-либо из изломов связи (кроме двух базовых), необходимо либо дважды кликнуть по точке излома, которую необходимо убрать, либо вручную совместить одну точку излома с другой.

Чтобы удалить связь, необходимо выделить её и нажать клавишу «Del» на клавиатуре, либо выбрать пункт меню «Удалить» в меню «Правка» или контекстного меню.

## 3.6 Создание пользовательских библиотек

В среде программирования контроллеров SoftPLC имеется два типа блоков, которыми пользователь может оперировать при разработке управляющей программы: стандартные и пользовательские.

Стандартными функциональными блоками являются: объекты входов/выходов, объекты математических, логических операций, таймеры, счетчики, и т.д.

Пользовательские функциональные блоки – блоки, созданные из стандартных компонентов, объединенные в единый блок с указанным пользователем количеством входов и выходов. Данный тип компонентов предназначен для значительного сокращения размера управляющей программы, а также для многократного использования набора блоков, реализующих определённую логику (Рисунок 19).

Для создания пользовательского блока нужно создать новый проект и выбрать тип проекта «Библиотека». При этом среда создания проекта никак не отличается от создания обычной программы. Принцип создания программы для описания логики библиотеки остаётся тем же, единственное изменение заключается в том, что данный проект нельзя отправлять в ядро и производить его запуск, а можно лишь сохранять для дальнейшего импорта и использования в простых программах. Соответственно пункт главного меню «Модель», используя который производится запуск программы, заменяется на пункт «Библиотека» для работы с ней.

Основным принципом создания библиотеки является то, что созданные в проекте входы и выходы можно в итоге назначить как входы/выходы будущей библиотеки. На Рисунок 19 представлен пример библиотеки для выполнения арифметический операций: на вход подаются 5 чисел (A,B,C,D,E). На первый выход передаётся результат следующей арифметической операции: RES = A+B*C/D. Второй выход хранит логическое значение, соответствующее признаку равенства результата арифметических операций и числа, поданного на вход E.

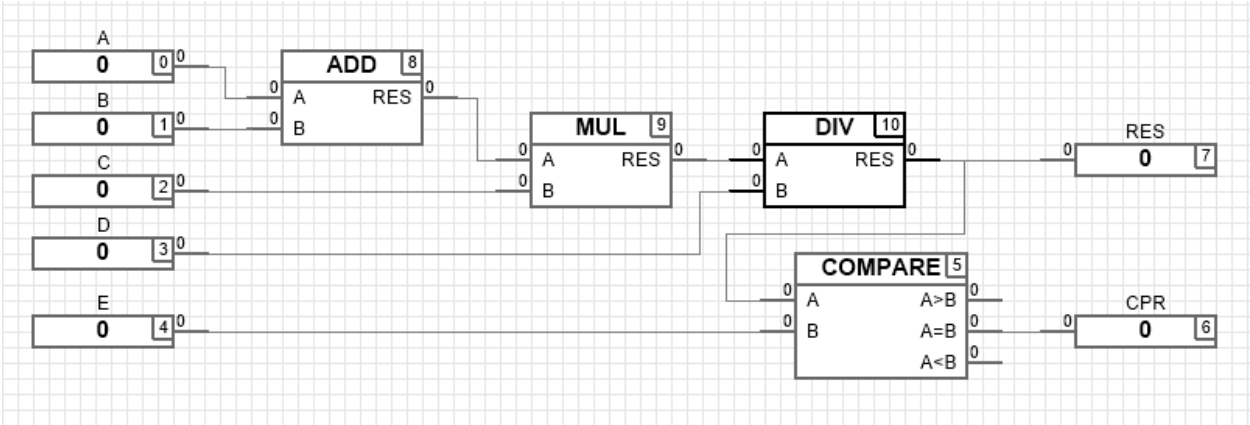



**Рисунок 19 – Создание пользовательской библиотеки**

Следующим этапом создания библиотеки является привязка объектов входов/выходов к будущим контактам пользовательского блока. Т.е. необходимо указать, какие функциональные блоки входов/выходов будут соответствовать входным/выходным контактам пользовательского блока и в каком порядке они будут располагаться. Для этого нужно вызвать окно привязки входов/выходов, находящееся в меню «Библиотека»→«Привязка входов/выходов».

При вызове данной команды открывается диалоговое окно привязки входов/выходов и основных настроек пользовательской библиотеки, представленное на Рисунок 20.

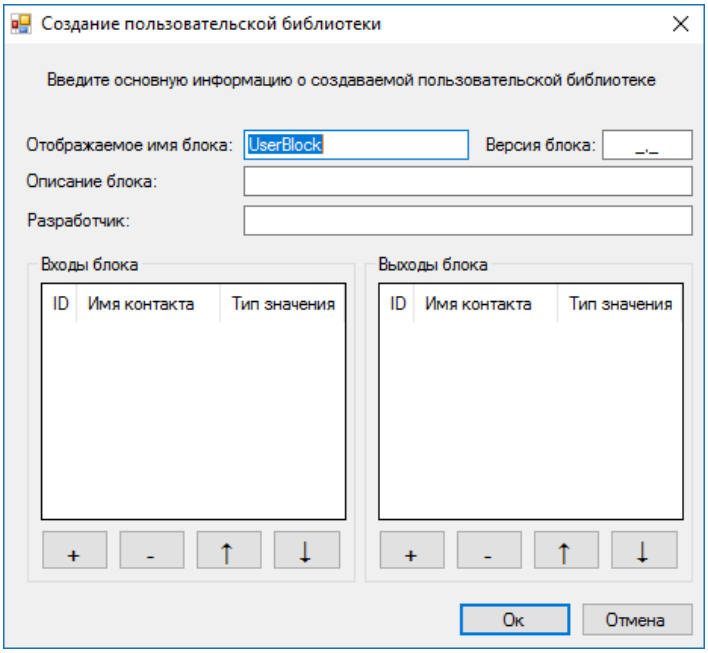



**Рисунок 20 – Диалоговое окно привязки входов/выходов пользовательской библиотеки**

Окно привязки входов/выходов содержит следующие настраиваемые параметры:

- Отображаемое имя блока – имя пользовательского функционального блока, которое будет на нём отображаться (по умолчанию «UserBlock»).
- Версия блока – поле для указания номера и подномера версии создаваемой библиотеки, по которому можно определить итерацию создания библиотеки.
- Описание блока – подробное описание работы пользовательского блока.
- Разработчик – поле для информации о создателе библиотеки.
- Входы/выходы блока – панели с возможностью привязки входных/выходных объектов к будущим контактам блока. Данные панели имеют кнопки «+» и «-» для добавления контакта с привязанным входом/выходом или его удаления соответственно.

Для указанного примера необходимо привязать пять входов и два выхода. Для этого следует нажать кнопку «+» в панели «Входы блока», указать первый объект входа с именем «A», и нажать «Ок» (Рисунок 21).

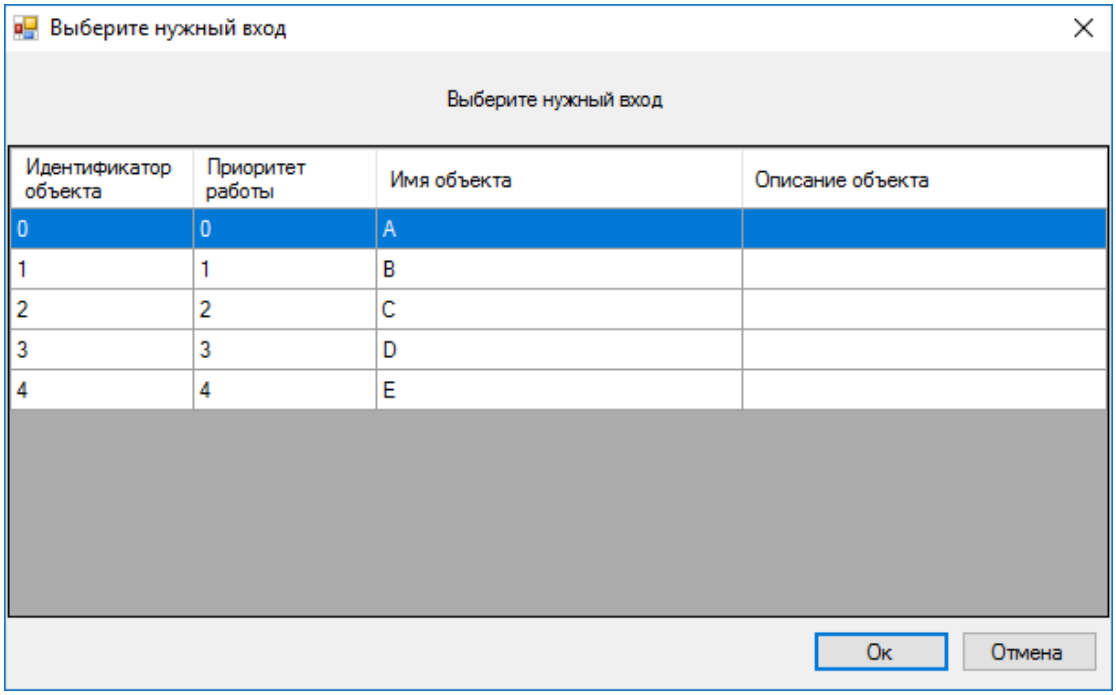



**Рисунок 21 – Окно привязки входов**

Затем выполнить те же действия с остальными входами.

С выходами необходимо проделать аналогичную процедуру и добавить два функциональных объекта выхода. Каждый вход/выход можно использовать в привязке только один раз. Для изменения порядка следования входов/выходов в библиотеке имеются соответствующие кнопки со стрелками.

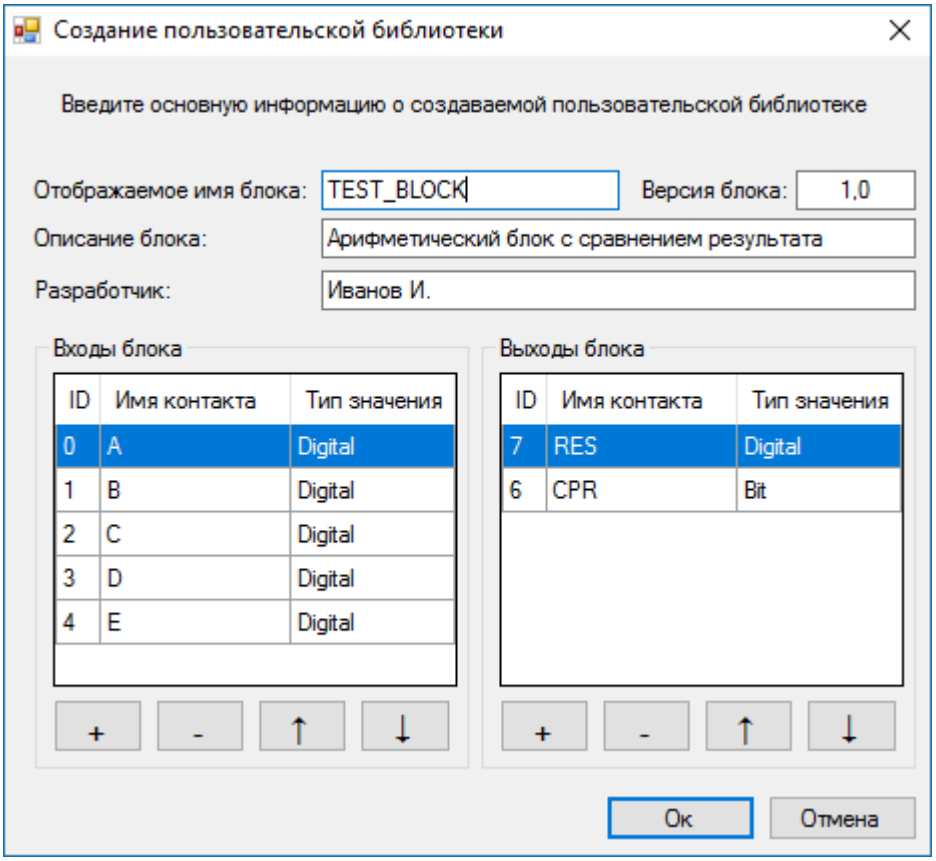



**Рисунок 22 – Заполненная форма привязки входов/выходов пользовательской библиотеки**

После выполнения вышеописанных процедур реализуется создание библиотеки, которую можно в дальнейшем многократно использовать в простых управляющих программах для ПЛК.

## 3.7 Импорт и использование пользовательских блоков

Чтобы использовать в программе ПЛК разработанный пользовательский блок, его нужно импортировать в проект. Для этого в окне проекта необходимо выбрать команду в главном меню «Файл»→«Импорт пользовательской библиотеки». Затем найти сохранённый файл библиотеки (с расширением «fbl») и нажать кнопку «Открыть». При успешном импорте библиотеки должно появиться сообщение о данном действии (Рисунок 23).

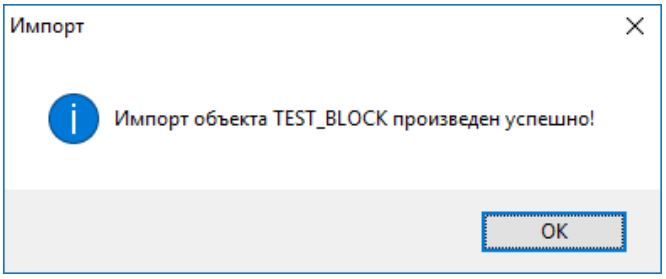



**Рисунок 23 – Сообщение об успешном импорте пользовательской библиотеки**

Также в меню функциональных блоков должен появиться пункт, соответствующий названию нового пользовательского блока (Рисунок 24).

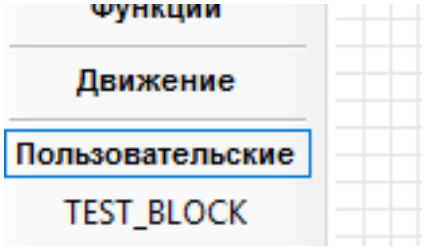



**Рисунок 24 – Новый функциональный блок в меню объектов после импорта библиотеки**

После этого данный пользовательский блок является полноценным функциональным объектом, который можно многократно использовать в любых программах. На Рисунок 25 показана программа с использованием созданного тестового пользовательского блока.

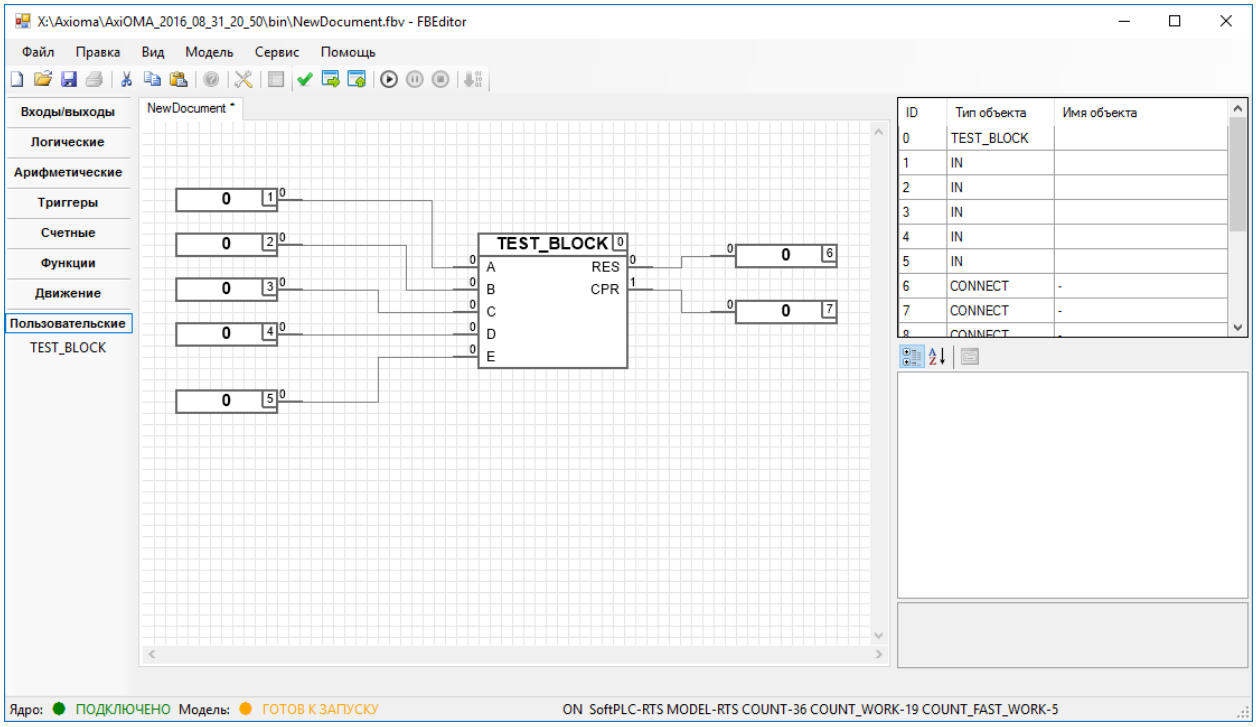



**Рисунок 25 – Проект управляющей программы с использованием пользовательского блока TEST_BLOCK**

На Рисунок 26 представлен вид работающей программы с использованием пользовательского блока TEST_BLOCK. Для проверки входам/выходам были заданы значения и программа запущена для отладки. В представленном примере RES=2+3*4/10=2. Заданное на вход E библиотеки совпадает с результатом. На выходе библиотеки получен результат в виде цифрового значения «2» и логической единицы на втором выходе, говорящей о совпадении двух чисел.

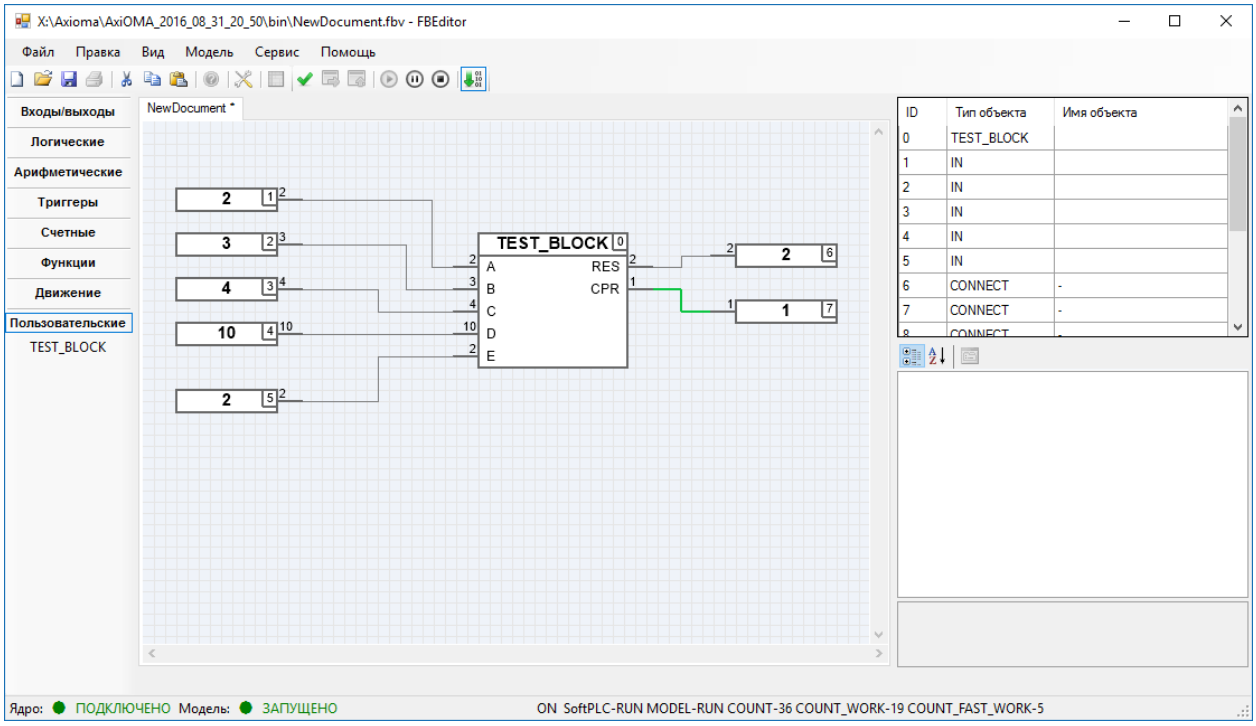



**Рисунок 26 – Работа пользовательского блока в управляющей программе**

Основным предназначением пользовательских библиотек является возможность их многократного использования, что позволяет сократить время на разработку программ и уменьшить их размер.

При работе программы, имеющей в составе пользовательский блок, есть возможность просмотра его внутренней логики путём двойного клика на нём. При этом в новой вкладке в режиме чтения открывается внутренняя часть логики пользовательского блока с обновлением значений, если включен режим отладки (Рисунок 27).

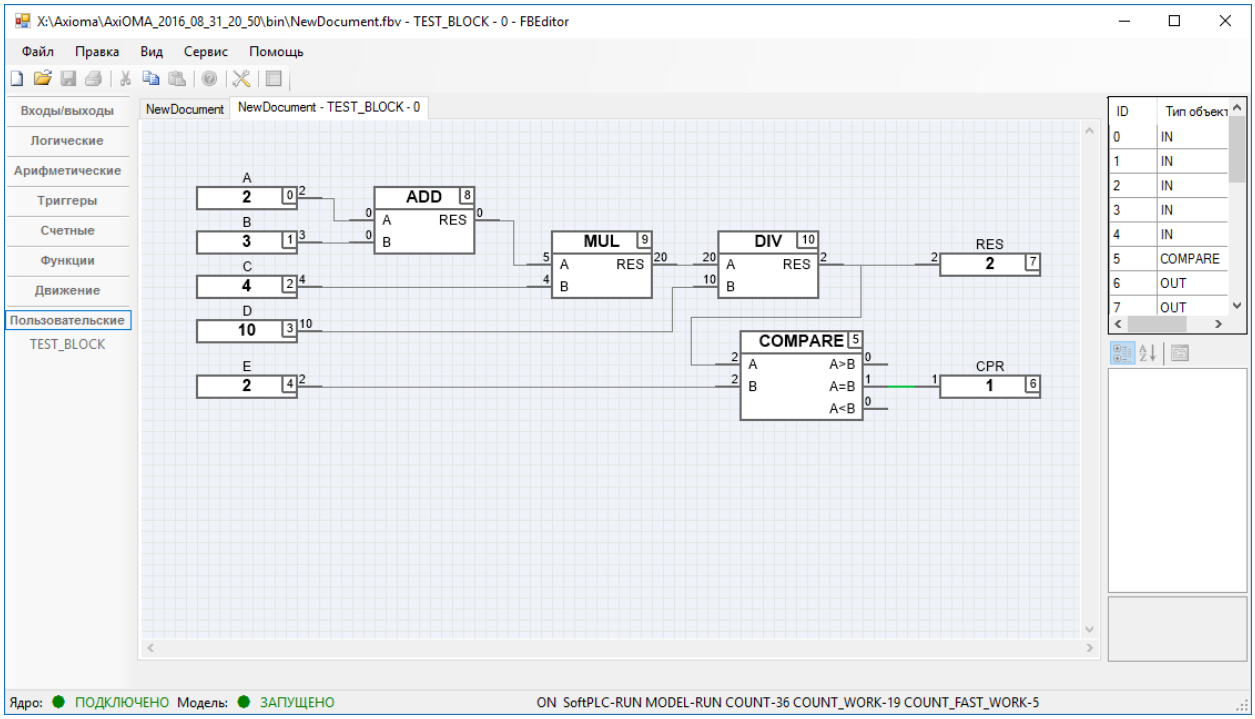



**Рисунок 27 – Раскрытый пользовательский блок TEST_BLOCK в режиме отладки**

Как видно из рисунка, в библиотеке можно наблюдать все актуальные значения на входах и выходах, а также на контактах блоков, что очень важно для процесса отладки программы.

Изменение внутренней логики библиотеки осуществляется только через открытие файла библиотеки и внесение изменений в него. Затем для замены библиотеки в программе необходимо правой кнопкой мыши кликнуть на её имени в панели объектов, выбрать пункт «Заменить библиотеку» и выбрать файл её расположения. В том же меню можно и убрать библиотеку из меню, если она не используется в программе.

Программа ПЛК может иметь неограниченное количество пользовательских библиотек. При сохранении проекта программы ПЛК в главном проекте содержатся все сведения об используемых библиотеках. Соответственно, для хранения всей программы и входящих в её состав пользовательских библиотек достаточно иметь один файл.

## 3.8 Замена пользовательских блоков

Чтобы заменить имеющийся пользовательский блок, необходимо в его контекстном меню в панели функциональных блоков выбрать «Заменить библиотеку» (Рисунок 28).

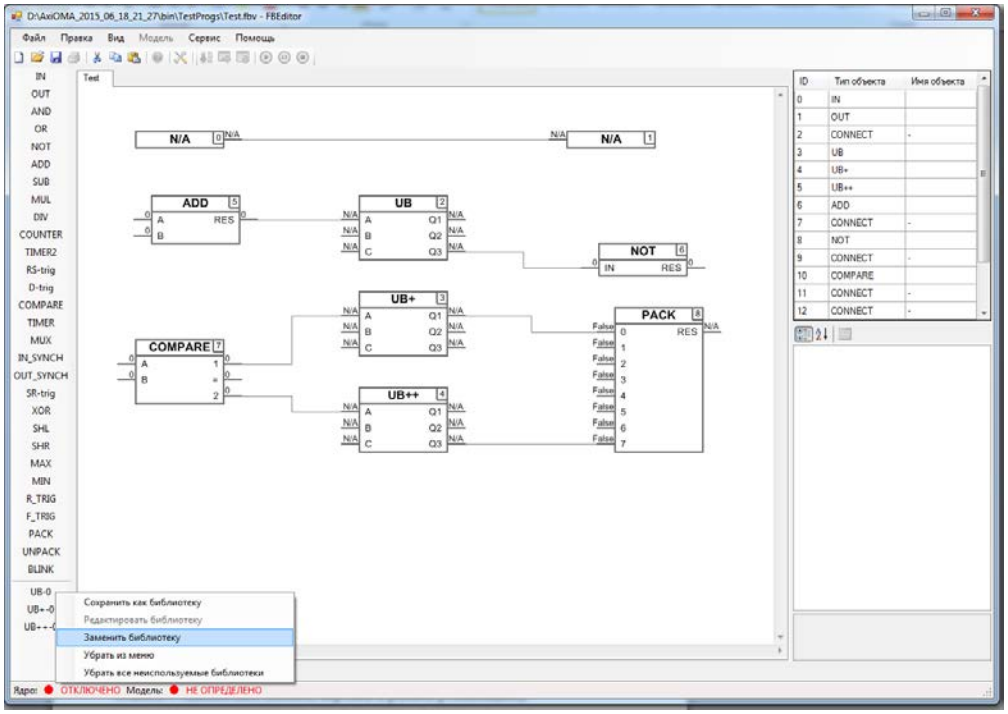



**Рисунок 28 – Функция «Заменить библиотеку»**

При этом откроется меню выбора нового файла, в котором необходимо выбрать файл библиотеки. Если имя и версия новой библиотеки совпадает с именем и версией выбранной (заменяемой) библиотеки – замена разрешена, поскольку целостность и неповторяемость блоков в меню не нарушится.

Если имя и версия новой библиотеки совпадает с именем и версией любой другой (кроме заменяемой) библиотеки из меню – замена запрещается, иначе будет два объекта с одинаковым именем и версией, но разной внутренней реализацией.

Замена пользовательских блоков производится рекурсивно по всем открытым документам, а также пользовательским блокам. Т.е. если внутри другой (других) библиотек имеется заменяемая библиотека, она также будет заменена.

В текущей реализации все подключенные к блоку связи остаются подключенными так же, как и были. Если количество контактов увеличилось – то они добавляются ниже имеющихся, если количество контактов уменьшилось, то необходимо удалить «свободные» связи вручную.

Помимо этого имеется возможность убрать из меню все неиспользуемые библиотеки. Это реализуется также через контекстное меню пользовательского блока. Это бывает удобно, когда импортировано много блоков, которые никак не используются на открытых документах.

## 3.9 Создание регионов

Графическое выделение группы логически связанных блоков в ПЛК программе осуществляется с использованием регионов. Регион представляет собой прозрачный прямоугольник с названием, в который можно объединять произвольное число функциональных блоков. Регион не влияет на последовательность выполнения управляющей программы. Основной функциональностью региона является возможность

перемещения региона вместе со всеми блоками, которые находятся в нём. Размер региона можно задавать произвольно и регулировать.

Чтобы создать регион, необходимо нажать кнопку «Region» на панели инструментов и щёлкнуть в окне редактора на точку, где будет располагаться левый верхний угол региона. Не отпуская левую кнопку мыши, необходимо провести мышью до места, где будет создан регион. После того, как левая кнопка мыши будет отпущена, пользователю предоставляется возможность задать имя региона. После нажатия кнопки «ОК» в диалоге задания имени региона в окне редактора создаётся регион (рисунок Рисунок 29).

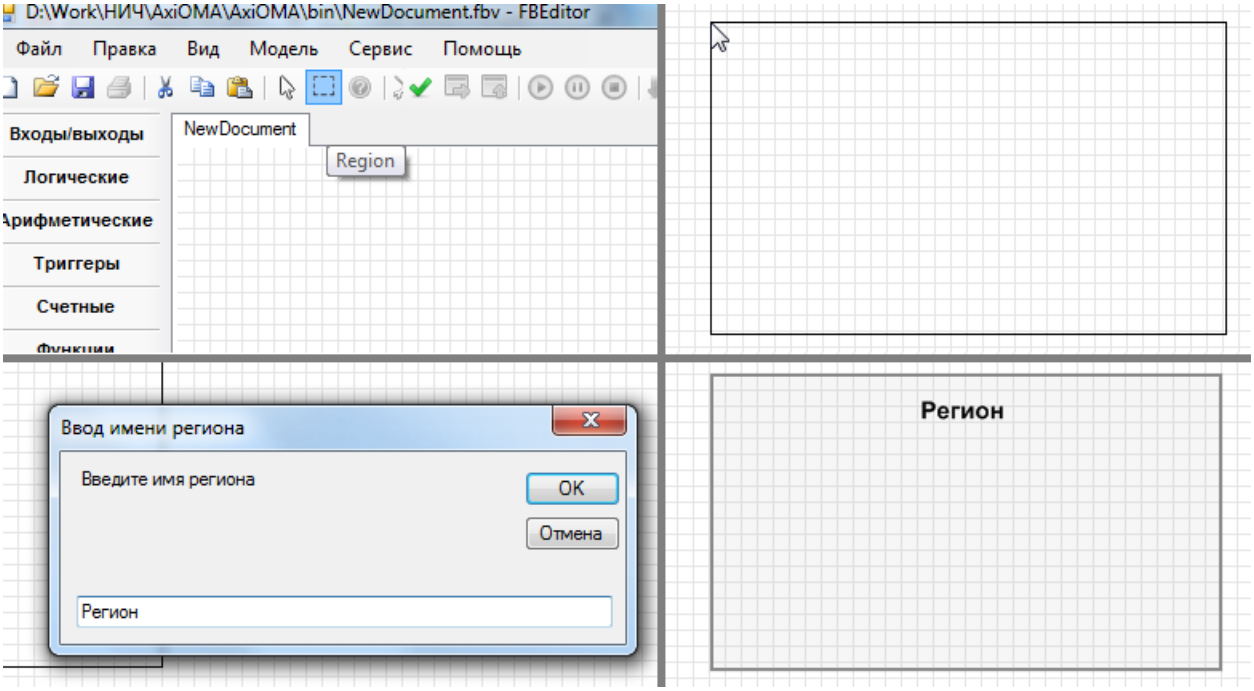



**Рисунок 29 – Этапы создания региона**

Чтобы изменить размеры региона, необходимо выделить регион, щёлкнув левой кнопкой мыши в области региона. После этого необходимо подвести указатель мыши к одной из границ региона. Курсор поменяется со стандартной стрелки на двойную стрелку изменения размеров объекта. Потянув мышью за грань региона, можно изменить его размер (Рисунок 30).

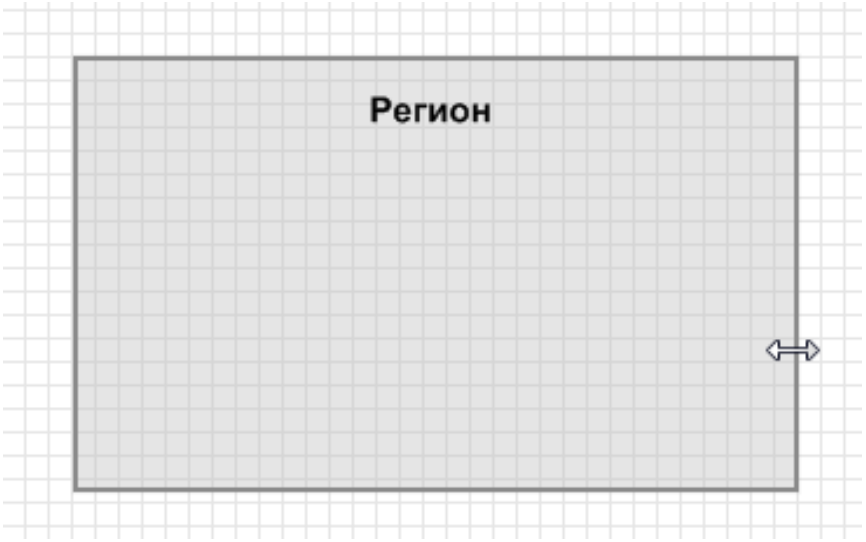



**Рисунок 30 – Изменение размеров региона**

Перемещение региона осуществляется после его выделения и последующего нажатия на клавиши стрелок на клавиатуре (вверх, вниз, влево, вправо) или путём его перетаскивания с нажатой левой кнопкой мыши по рабочей области редактора (Рисунок 31). Чтобы удалить регион, необходимо выделить его и нажать клавишу «Del» на клавиатуре.

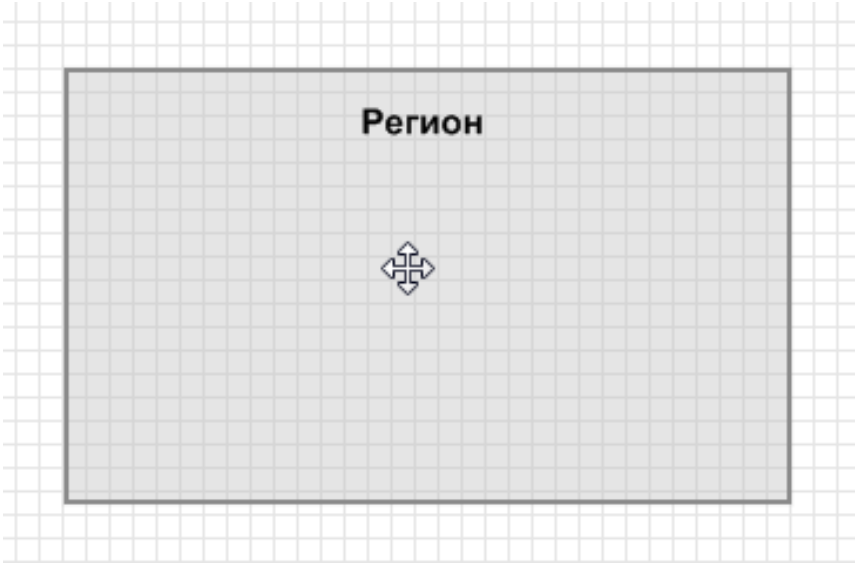



**Рисунок 31 – Перемещение региона**

При наведении указателя мыши на выделенный регион появляется всплывающая подсказка (Рисунок 32), в которой содержится описание региона. Задать описание региона можно в редакторе свойств объекта в поле «Описание объекта». В случае, если описание не задано, всплывающая подсказка не появляется.

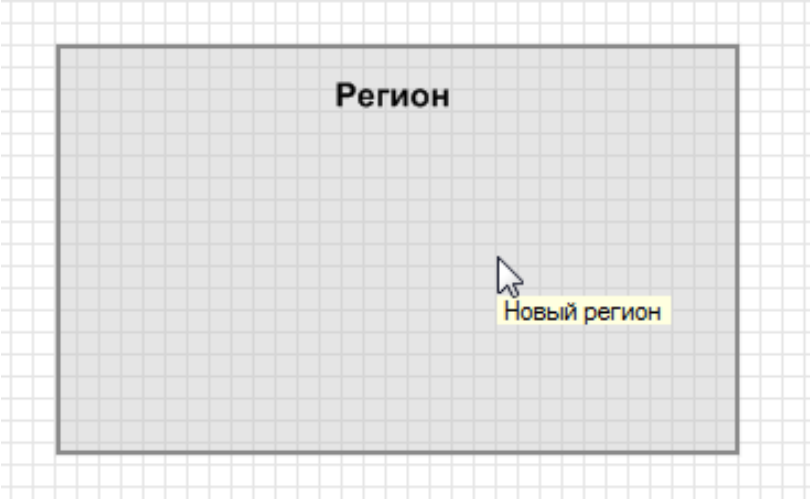



**Рисунок 32 – Всплывающая подсказка с описанием региона**

Пример использования регионов для визуального объединения группы элементов и представления оператору функциональности, выполняющейся в данной области программы, представлен на Рисунок 33.

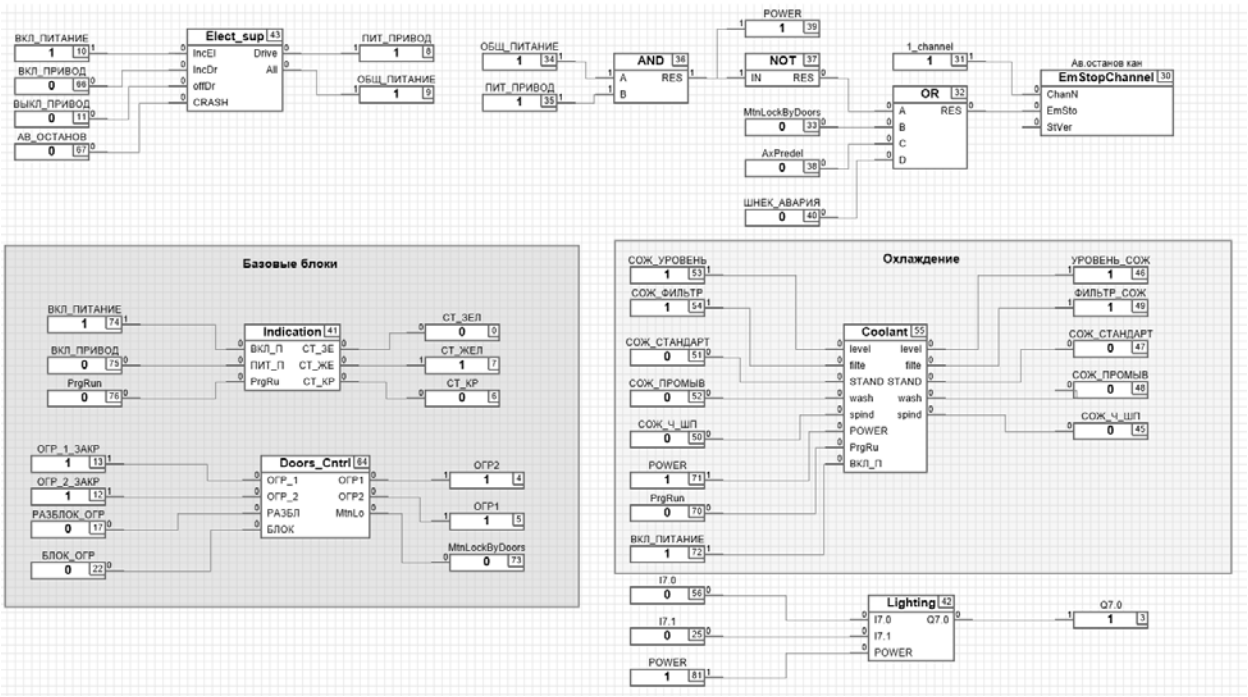



**Рисунок 33 – Пример использования регионов**

## 3.10 Настройки ПЛК редактора

ПЛК редактор предоставляет пользователю возможность настройки его работы, а также гибкие возможности настройки его интерфейса. Для этого необходимо выбрать команду в главном меню «Сервис»→«Настройки» (Рисунок 34).

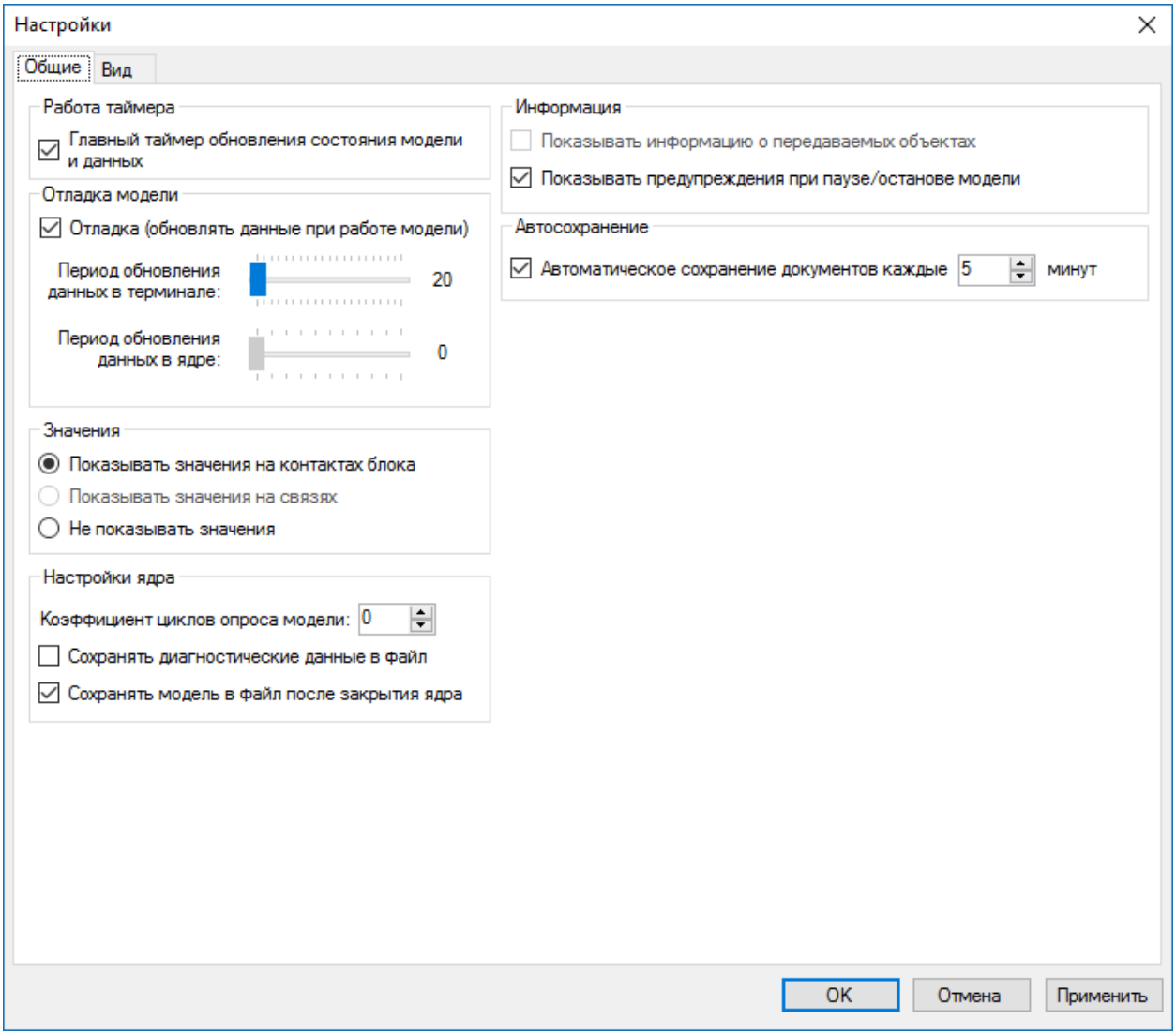



**Рисунок 34 – Окно настроек редактора, вкладка общих настроек**

На вкладке «Общие» имеется возможность настроить следующие параметры:

- Главный таймер обновления состояния модели и данных – отвечает за обновление всех данных из ядра контроллера. При его отключении при отладке статусы работы ядра не поступают в редактор и отладка программы делается невозможной.
- Отладка – флаг, позволяющий включить/выключить функцию отладки программы.
- Периоды обновления данных в терминале и ядре.
- Тип отображения значений.
- Настройки ядра: сохранение диагностических данных, сохранение модели в файл, а также цикл опроса модели.
- Настройки отображения диалоговых окон с информационными сообщениями.
- Автоматическое сохранение документов и его интервал.

Вкладка «Вид» настроек содержит основные параметры отображения всех элементов редактора (Рисунок 35).

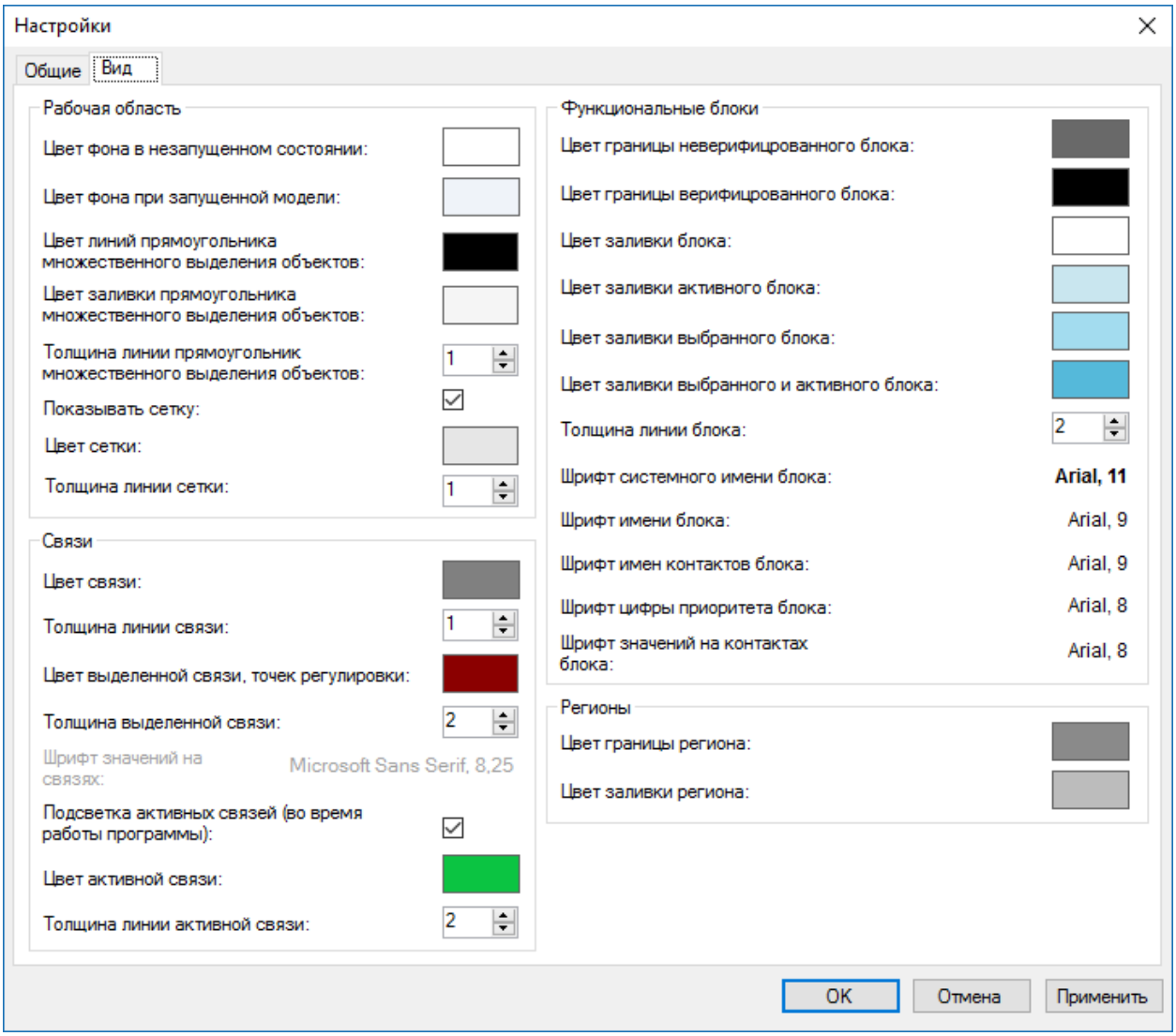



**Рисунок 35 – Окно настроек редактора, вкладка настроек интерфейса**

В настройках интерфейса можно настраивать внешний вид рабочей области создания программы, вид связей, функциональных блоков, а также регионов. Для смены цвета необходимо кликнуть на значок с его отображением и выбрать нужный цвет из палитры.

После изменения настроек необходимо нажать кнопку «ОК».

## 3.11 Верификация, запуск и отладка программы

Для проведения отладки программы ПЛК необходимо произвести её запуск. Поскольку все основные расчёты и операции для работы ПЛК программы производятся в вычислительном ядре контроллера, перед запуском программу необходимо передать в ядро. Первым условием для этого является активное подключение к ядру (см. раздел 3.1).

ПЛК программа не может быть отправлена в ядро (соответствующие кнопки для этого не будут активными), если в ядре в данный момент уже функционирует ПЛК

программа. В этом случае необходимо сначала произвести останов находящейся в ядре программы путём нажатия на кнопку «Стоп» меню управления работой модели.

Перед отправкой программы в ядро целесообразно произвести её верификацию путём нажатия на соответствующую кнопку в меню управления работой модели. Будет произведена проверка на повторяющиеся идентификаторы, неподключенные связи, и т.д. При успешном прохождении проверки будет показано оповещение в виде диалогового окна.

Отправка в ядро программы может быть осуществлена посредством вызова соответствующей команды из главного меню («Модель»→«Передать модель в ядро»), либо посредством нажатия горячей клавиши F6, либо путём нажатия на кнопку «Отправить в ядро» в меню управления работой модели.

Если программа передана успешно, появится соответствующее сообщение. После этого, когда программы в терминальной части и ядре системы ЧПУ являются полностью идентичными, можно производить запуск программы.

Запуск реализуется путём нажатия на соответствующую кнопку в меню управления работой программы, или по нажатию горячей клавиши F5. После запуска программы, при включенном режиме отладки (по умолчанию), ядро с заданной частотой производит отправку в терминальную часть редактора информацию о блоках, значения в которых изменились. Таким образом, имеется возможность в режиме реального времени производить отладку управляющей программы и следить за актуальными значениями входов/выходов функциональных блоков ПЛК программы. Для наглядности процесса отладки программы связи, передающие дискретные значения, подсвечиваются при передаче значения «1» (Рисунок 36).

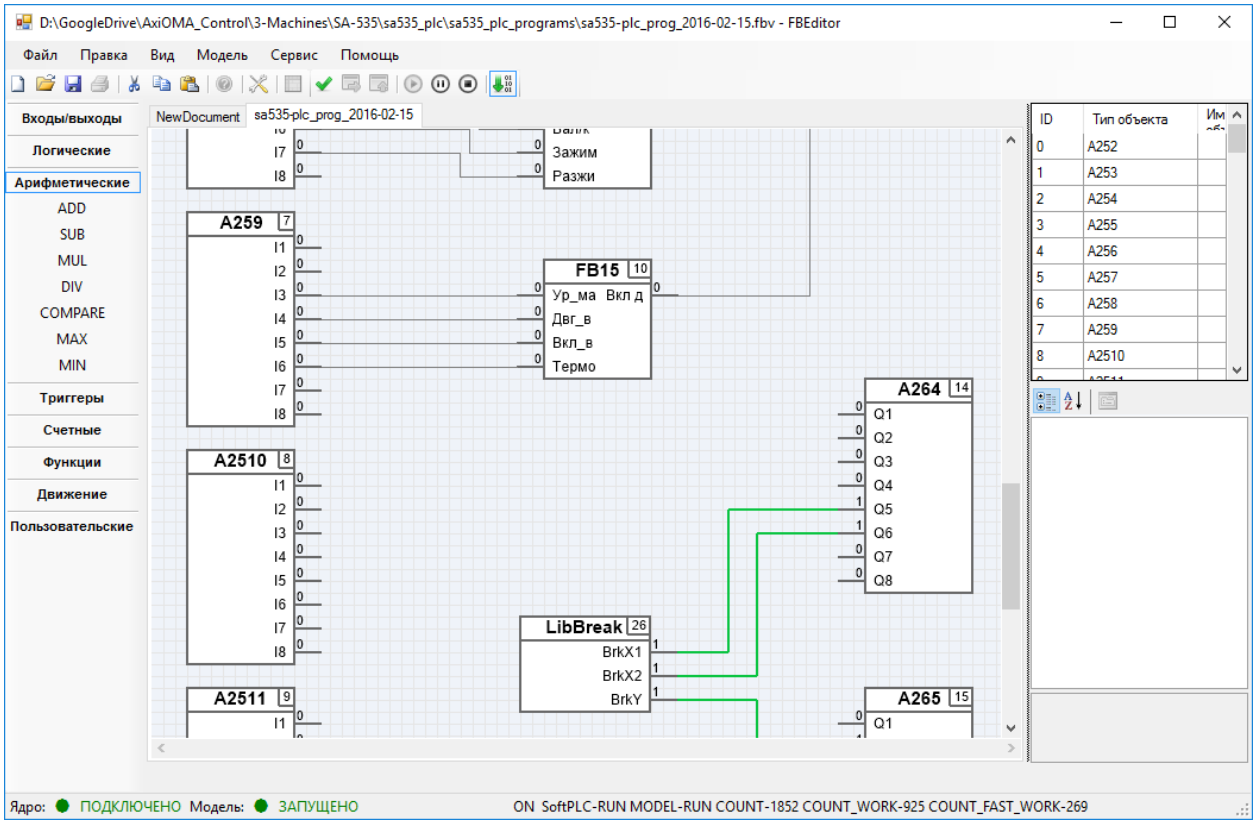



**Рисунок 36 – Общий вид работающей программы в FB-редакторе**

В режиме отладки запрещено вносить изменения в программу, поскольку она находится в рабочем состоянии. Можно отключить режим отладки в редакторе, нажав соответствующую кнопку на панели управления работой модели.

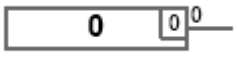
Для остановки работы программы следует нажать на кнопку «Стоп» меню управления работой модели или нажать комбинацию горячих клавиш «Shift+F5». После этого можно вносить изменения в программу. Чтобы снова запустить программу, требуется заново выполнить действия по отправке программы в ядро и её запуску.

# 4 Описание функциональных блоков

## 4.1 Блоки входов и выходов

### 4.1.1 Вход

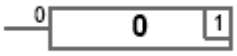
Блок позволяет задать ссылку на переменную из области памяти (CommonPlcMemory) или иметь константное значение, для его передачи далее в программе.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Выход | - | Bit/Digital/Analog | Переменная, из которой берется значение (вход) |

### 4.1.2 Выход

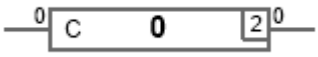
Блок позволяет записать в переменную, ссылающуюся на область памяти (CommonPlcMemory), или некую константу, заданное значение.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Выход | - | Bit/Digital/Analog | Переменная, в которую записывается значение (выход) |

### 4.1.3 Вход с синхронизацией

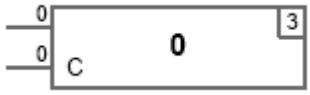
Блок позволяет задать ссылку на переменную из области памяти (CommonPlcMemory) или иметь константное значение, для его передачи далее в программе. При этом значение передается только в том случае, если на вход “C” подана логическая единица.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | C | Bit | Флаг работы объекта входа |
| Выход | - | Bit/Digital/Analog | Переменная, из которой берется значение (вход) |

### 4.1.4 Выход с синхронизацией

Блок позволяет записать в переменную, ссылающуюся на область памяти (CommonPlcMemory), или некую константу, заданное значение. При этом запись производится только в том случае, если на вход “C” подана логическая единица.

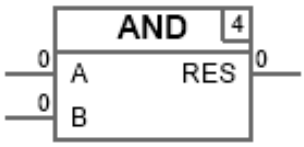



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | - | Bit/Digital/Analog | Переменная, в которую записывается значение (выход) |
| Вход | C | Bit | Флаг работы объекта выхода |

## 4.2 Логические блоки

### 4.2.1 AND: дизъюнкция (И, логическое умножение)

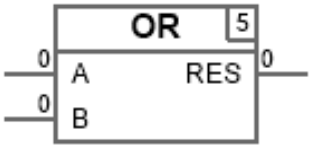
Блок реализует операцию логического умножения. Входы A и B принимают значения логических переменных, выход RES осуществляет вывод результата операции логического умножения. Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Bit | Входное логическое значение |
| Вход | B | Bit | Входное логическое значение |
| Выход | RES | Bit | Результат логического умножения |

### 4.2.2 OR: конъюнкция (ИЛИ, логическое сложение)

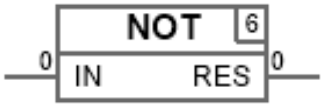
Блок реализует операцию логического сложения. Входы A и B принимают значения логических переменных, выход RES осуществляет вывод результата операции логического сложения. Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Bit | Входное логическое значение |
| Вход | B | Bit | Входное логическое значение |
| Выход | RES | Bit | Результат логического сложения |

### 4.2.3 NOT: отрицание (логическое НЕ)

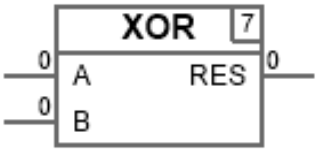
Элемент реализует операцию логического отрицания. Вход IN принимает значение переменной, выход RES осуществляет вывод результата операции логического отрицания.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Входное логическое значение |
| Выход | RES | Bit | Результат логического отрицания |

### 4.2.4 XOR: исключающее ИЛИ

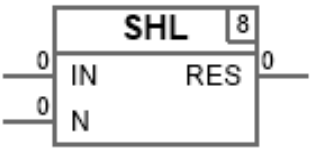
Побитовое исключающее ИЛИ. Если один из входных битов равен 1, то итоговый бит равен 1, в противном случае – 0.
Возможно увеличение входов до 16. В этом случае входы обрабатываются попарно, затем к результатам опять применяется XOR. Такой алгоритм определён стандартом.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Bit | Входное логическое значение |
| Вход | B | Bit | Входное логическое значение |
| Выход | RES | Bit | Результат операции исключающего ИЛИ |

### 4.2.5 SHL: побитовый сдвиг операнда влево

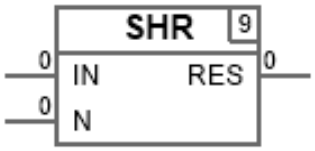
Если N превышает ширину типа данных, то операнды типов BYTE, WORD и DWORD заполняются нулями, операнды типов со знаками, таких как INT, получают арифметический сдвиг, т.е. заполняются значениями самого верхнего бита.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Сдвигаемый операнд |
| Вход | N | Bit | Количество бит, на которое сдвигается операнд IN влево |
| Выход | RES | Bit | Результирующий бит |

### 4.2.6 SHR: побитовый сдвиг операнда вправо

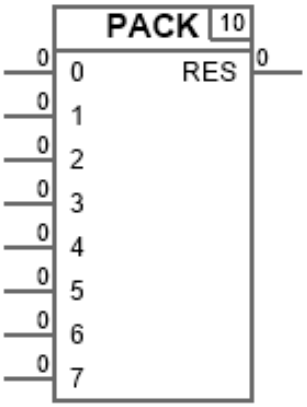
Если N превышает ширину типа данных, то операнды типов BYTE, WORD и DWORD заполняются нулями, операнды типов со знаками, таких как INT, получают арифметический сдвиг, т.е. заполняются значениями самого верхнего бита.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Сдвигаемый операнд |
| Вход | N | Bit | Количество бит, на которое сдвигается операнд IN вправо |
| Выход | RES | Bit | Результирующий бит |

### 4.2.7 PACK

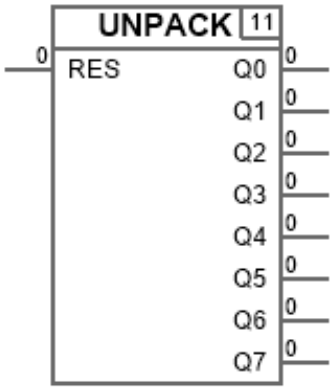
Блок преобразует восемь входных параметров типа BIT в один байт (DIGITAL).




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | 0 | Bit | Входное битовое значение |
| Вход | 1 | Bit | Входное битовое значение |
| Вход | 2 | Bit | Входное битовое значение |
| Вход | 3 | Bit | Входное битовое значение |
| Вход | 4 | Bit | Входное битовое значение |
| Вход | 5 | Bit | Входное битовое значение |
| Вход | 6 | Bit | Входное битовое значение |
| Вход | 7 | Bit | Входное битовое значение |
| Выход | RES | Digital | Результат |

### 4.2.8 UNPACK

Блок преобразует один входной байт (Digital) в восемь выходных параметров типа BIT.

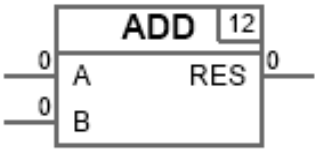



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | RES | Digital | Входной байт |
| Выход | Q0 | Bit | Выходное битовое значение |
| Выход | Q1 | Bit | Выходное битовое значение |
| Выход | Q2 | Bit | Выходное битовое значение |
| Выход | Q3 | Bit | Выходное битовое значение |
| Выход | Q4 | Bit | Выходное битовое значение |
| Выход | Q5 | Bit | Выходное битовое значение |
| Выход | Q6 | Bit | Выходное битовое значение |
| Выход | Q7 | Bit | Выходное битовое значение |

## 4.3 Арифметические блоки

### 4.3.1 ADD: сложение

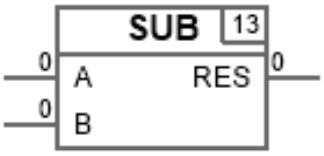
Блок реализует арифметическую операцию сложения. На входы A и B поступают слагаемые, на выход RES осуществляется вывод результирующей суммы.
Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Digital | Первое слагаемое |
| Вход | B | Digital | Второе слагаемое |
| Выход | RES | Digital | Сумма |

### 4.3.2 SUB: вычитание

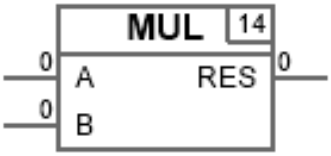
Блок реализует арифметическую операцию вычитания. На входы A поступает уменьшаемое, на вход B – вычитаемое, на выходе RES осуществляется вывод результата.
Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Digital | Уменьшаемое |
| Вход | B | Digital | Вычитаемое |
| Выход | RES | Digital | Результат |

### 4.3.3 MUL: умножение

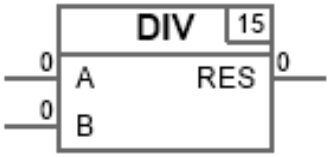
Блок реализует арифметическую операцию умножения. На входы A и B поступают множители, на выход RES осуществляется вывод результата операции.
Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Digital | Множитель |
| Вход | B | Digital | Множитель |
| Выход | RES | Digital | Результат |

### 4.3.4 DIV: деление

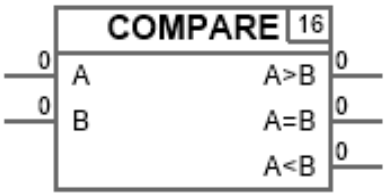
Блок реализует арифметическую операцию деления. На вход A поступает делимое, на вход B – делитель, на выход RES осуществляется вывод результата операции.
Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Analog | Делимое |
| Вход | B | Analog | Делитель |
| Выход | RES | Analog | Результат |

### 4.3.5 COMPARE: сравнение

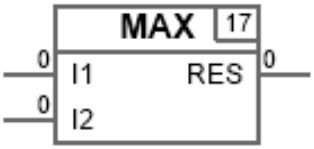
Блок реализует сравнение двух чисел. На входы подаются два значения, которые необходимо сравнить между собой. Выходы отвечают за результат сравнения.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | A | Analog | Значение для сравнения 1 |
| Вход | B | Analog | Значение для сравнения 2 |
| Выход | A>B | Bit | Логическая «1», если A>B |
| Выход | A=B | Bit | Логическая «1», если A=B |
| Выход | A<B | Bit | Логическая «1», если A<B |

### 4.3.6 MAX: определение максимального из значений

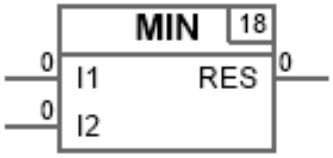
Возвращает наибольшее из входных значений. Имеется возможность увеличения входов до 16 штук.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | I1 | Analog | Значение выборки |
| Вход | I2 | Analog | Значение выборки |
| Выход | RES | Analog | Результат (максимальное число из набора входных значений) |

### 4.3.7 MIN: определение минимального из значений

Возвращает наименьшего из входных значений. Имеется возможность увеличения входов до 16 штук.

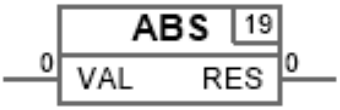



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | I1 | Analog | Значение выборки |
| Вход | I2 | Analog | Значение выборки |
| Выход | RES | Analog | Результат (минимальное число |

| | | | |
|---|---|---|---|
| | | | из набора входных значений) |

### 4.3.8 ABS: получение значения по модулю

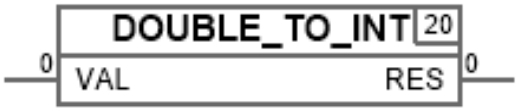
Возвращает на выходе значение по модулю.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | VAL | Analog | Входное значение |
| Выход | RES | Analog | Значение по модулю |

### 4.3.9 DOUBLE_TO_INT: преобразование

Блок преобразует входное значение числа с плавающей точкой в целое число.



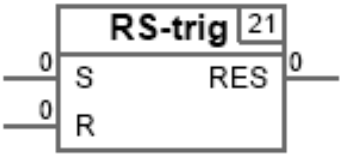

| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | VAL | Analog | Входное значение |
| Выход | RES | Digital | Преобразованное (округленное) значение |

## 4.4 Триггеры

### 4.4.1 RS-trig: RS триггер

Имеет два управляющих входа S (вход установки) и R (вход сброса), выполняющих функции:

- установки RES=1 при S=1 и R=0;
- сброса RES=0 при S=0 и R=1;
- при S=R=0 триггер работает в режиме хранения;
- S=R=1 – запрещённая комбинация (установка и сброс одновременно) – приводит к неопределённости состояния Q. Приоритет входа сброса R выше, чем входа установки S, т.е. когда на оба входа R и S одновременно поступает логическая 1, выход Q переходит в состояние логического 0.

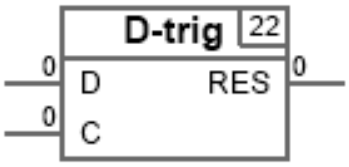



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Вход | S | Bit | Управляющий вход |
| Вход | R | Bit | Управляющий вход |
| Выход | RES | Bit | Результат |

### 4.4.2 D-trig: D триггер (триггер задержки)

Сохраняет значение на выходе, равное входному значению D то тех пора, пока на вход C подана логическая «1».

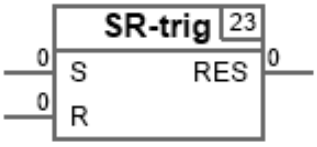



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | D | Bit | Информационный вход |
| Вход | C | Bit | Вход задержки |
| Выход | RES | Bit | Результат |

### 4.4.3 SR-trig: SR триггер

Имеет два управляющих входа S (вход установки) и R (вход сброса), выполняющих функции:

- установки RES=1 при S=1 и R=0;
- сброса RES=0 при S=0 и R=1;
- при S=R=0 триггер работает в режиме хранения;
- S=R=1 – запрещённая комбинация (установка и сброс одновременно) – приводит к неопределённости состояния Q. Приоритет входа установки S выше, чем входа сброса R, т.е. когда на оба входа R и S одновременно поступает логическая 1, выход Q переходит в состояние логического 1.

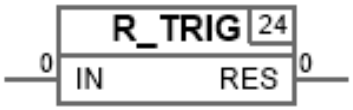



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | S | Bit | Управляющий вход |
| Вход | R | Bit | Управляющий вход |
| Выход | RES | Bit | Результат |

### 4.4.4 R-trig: импульс по переднему фронту

Значение на выходе RES равно логическому «0» до тех пор, пока значение на входе равно «0». Как только вход получает значение логическую «1», RES устанавливается в «1», но только на один такт контроллера.

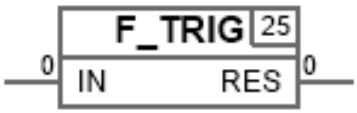
Таким образом, блок выдает единичный импульс при каждом переходе входного канала из «0» в «1».




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Управляющий вход |
| Выход | RES | Bit | Результат |

### 4.4.5 F-trig: импульс по заднему фронту

Значение на выходе RES равно логическому «0» до тех пор, пока значение на входе равно «1». Как только вход получает значение логическую «0», RES устанавливается в «1», но только на один такт контроллера.
Таким образом, блок выдает единичный импульс при каждом переходе входного канала из «1» в «0».

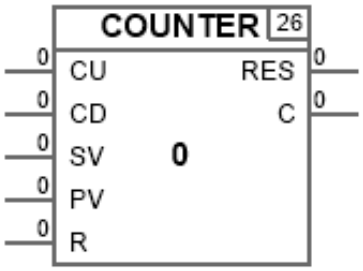



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Управляющий вход |
| Выход | RES | Bit | Результат |

## 4.5 Счетные блоки

### 4.5.1 COUNTER: счетчик

Блок реализует счетные операции, т.е. увеличение или уменьшение выходного значения, путем подачи логической «1» на вход для увеличения (CU) или вход уменьшения значения (CD) соответственно.

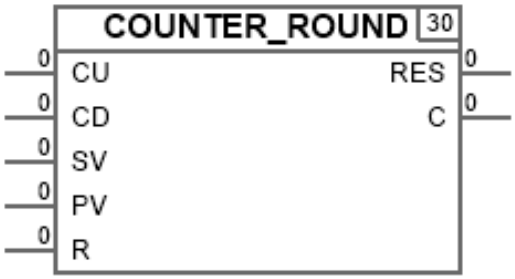



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | CU | Bit | Увеличивает счётное значение на единицу |
| Вход | CD | Bit | Уменьшает счётное значение на единицу |
| Вход | SV | Digital | Стартовое (минимальное) значение счетчика |

| | | | |
|---|---|---|---|
| Вход | PV | Digital | Значение уставки (максимальное значение счетчика) |
| Вход | R | Bit | Осуществляет сброс счётчика в начальное значение |
| Выход | RES | Bit | Выход, активизируется в момент достижения счетчиком значения уставки |
| Выход | C | Digital | Текущее счетное значение |

### 4.5.2 COUNTER_ROUND: двунаправленный счетчик

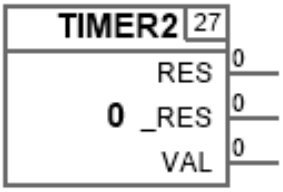
Блок реализует счетные операции, но при переходе через максимальное значение он продолжает счет с нуля. При переходе через ноль в отрицательную сторону устанавливает максимальное значение и продолжает уменьшаться.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | CU | Bit | Увеличивает счётное значение на единицу |
| Вход | CD | Bit | Уменьшает счётное значение на единицу |
| Вход | SV | Digital | Стартовое (минимальное) значение счетчика |
| Вход | PV | Digital | Значение уставки (максимальное значение счетчика) |
| Вход | R | Bit | Осуществляет сброс счётчика в начальное значение |
| Выход | RES | Bit | Выход, активизируется в момент достижения счетчиком значения уставки |
| Выход | C | Digital | Текущее счетное значение |

### 4.5.3 TIMER2: простой таймер

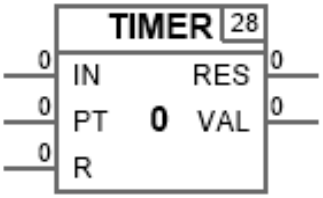
Блок выдает попеременно логические сигналы «0» и «1» на выходы в соответствии с заданным в параметрах блока интервалом обновления (интервал задается в миллисекундах).




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Выход | RES | Bit | Выходное логическое значение |
| Выход | _RES | Bit | Инвертированное выходное логическое значение |
| Выход | VAL | Digital | Актуальное значение таймера по ходу работы программы |

### 4.5.4 TIMER: таймер

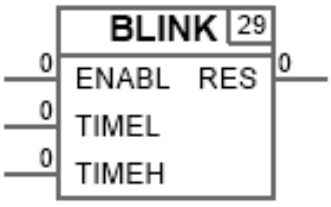
Блок позволяет сгенерировать выходной логический сигнал с заданной задержкой.




| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | IN | Bit | Входное логическое значение, запускающее работу таймера |
| Вход | PT | Digital | Значение интервала задержки таймера |
| Вход | R | Bit | Вход осуществляет сброс таймера в начальное значение |
| Выход | RES | Bit | Выходной логический сигнал |
| Выход | VAL | Digital | Актуальное значение таймера по ходу работы программы |

### 4.5.5 BLINK: генератор прямоугольных импульсов

Генератор запускается по входу ENABLE = «1». Длительность импульса задается входом TIMEH, длительность паузы – входом TIMEL.
При переходе входа ENABLE в «0», выход RES остается в том состоянии, в котором он был в момент подачи ENABLE.



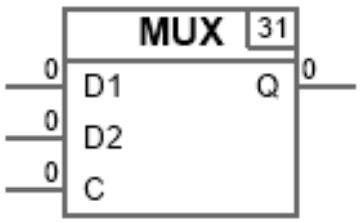

| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Вход | ENABL | Bit | Запуск генератора |
| Вход | TIMEL | Digital | Длительность паузы |
| Вход | TIMEH | Digital | Длительность импульса |
| Выход | RES | Bit | Выходной логический сигнал |

## 4.6 Функции

### 4.6.1 MUX: мультиплексор

Блок принимает два цифровых значения на вход, и выдает на выходе одно из них, в зависимости от состояния логического входа C.

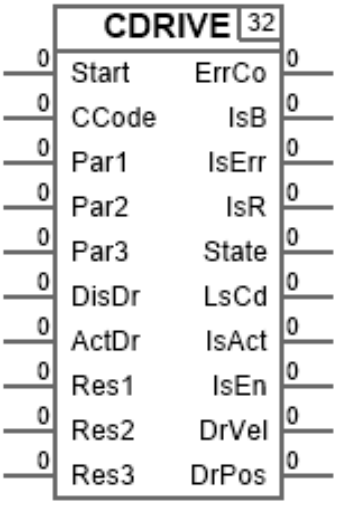



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | D1 | Digital | Первое цифровое значение |
| Вход | D2 | Digital | Второе цифровое значение |
| Вход | C | Bit | Логическое значение, определяющее, какое из двух чисел будет на выходе |
| Выход | C | Digital | Результат |

## 4.7 Управление движением

### 4.7.1 CDRIVE: блок управления приводом

Блок для управления движением. Возможно использовать для задания простых перемещений оси привода, установки разрешающих сигналов работы с приводом, мониторинга текущих состояний привода.



| Вход/выход | Имя | Тип данных | Описание |
|---|---|---|---|
| Вход | Start | Digital | Сигнал активности блока (при Start=0 входы игнорируются) |
| Вход | CCode | Digital | Код команды (воспринимается только при его изменении) |
| Вход | Par1 | Digital | Параметр 1 (для каждой команды свой) |
| Вход | Par2 | Digital | Параметр 2 (для каждой команды свой) |
| Вход | Par3 | Digital | Параметр 3 (для каждой команды свой) |
| Вход | DisDr | Bit | Снять Enable привода |
| Вход | ActDr | Bit | Установить/снять Active привода |
| Вход | Res1 | Digital | |
| Вход | Res2 | Digital | |
| Вход | Res3 | Digital | |
| Выход | ErrCo | Digital | Код текущей ошибки |
| Выход | IsB | Digital | Флаг занятости привода |
| Выход | IsErr | Digital | Флаг наличия ошибки |
| Выход | IsR | Digital | Флаг готовности привода к управлению (255-все в норме) |
| Выход | State | Digital | Текущее состояние привода |
| Выход | LsCd | Digital | Последняя выполненная команда управления |
| Выход | IsAct | Digital | Состояние Active привода |
| Выход | IsEn | Digital | Состояние Enable привода |
| Выход | DrVel | Digital | Текущая скорость оси привода в дискретах привода |
| Выход | DrPos | Digital | Текущая позиция оси привода в дискретах привода |

# 5 Конфигуратор устройств ввода/вывода

## 5.1 Назначение и базовые принципы работы модуля конфигурирования устройств ввода/вывода

Программный модуль конфигурирования устройств ввода/вывода позволяет описывать конфигурацию подключенных к контроллеру модулей ввода/вывода для работы с ними.

Основным принципом работы контроллера с устройствами ввода/вывода является механизм разделяемой памяти (Common Plc Memory), в которой хранятся данные о значениях, поступающих в модули ввода, а также о значениях, которые назначаются контроллером для модулей вывода. Конфигуратор устройств ввода/вывода позволяет настраивать подключенное оборудование и указывать области памяти, с которыми будут взаимодействовать аппаратные устройства.

## 5.2 Базовый интерфейс конфигуратора устройств ввода/вывода

Базовый интерфейс включает в себя дерево конфигурации с добавленными устройствами, панель их настроек и кнопок для управления передачей конфигурации в ядро и её документированием в файл (Рисунок 37).

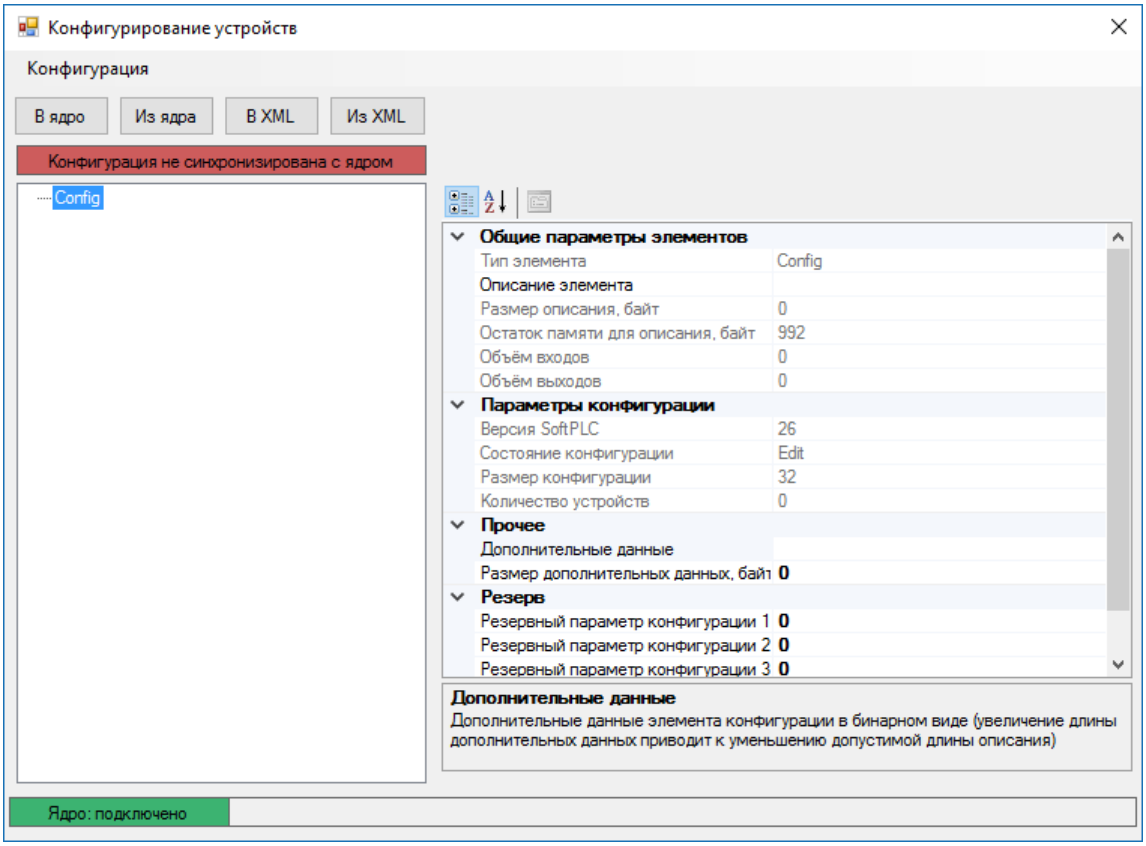



**Рисунок 37 – Базовый интерфейс конфигуратора устройств ввода/вывода**

## 5.3 Создание конфигурации устройств ввода/вывода

Создание конфигурации устройств ввода/вывода заключается в создании в дереве устройств компонентов: устройств (Device) и слотов (Slot), входящих в состав каждого из устройств. Для этого нужно по клику правой кнопки мыши на строке «Config» в дереве устройств выбрать нужную операцию (Рисунок 38).

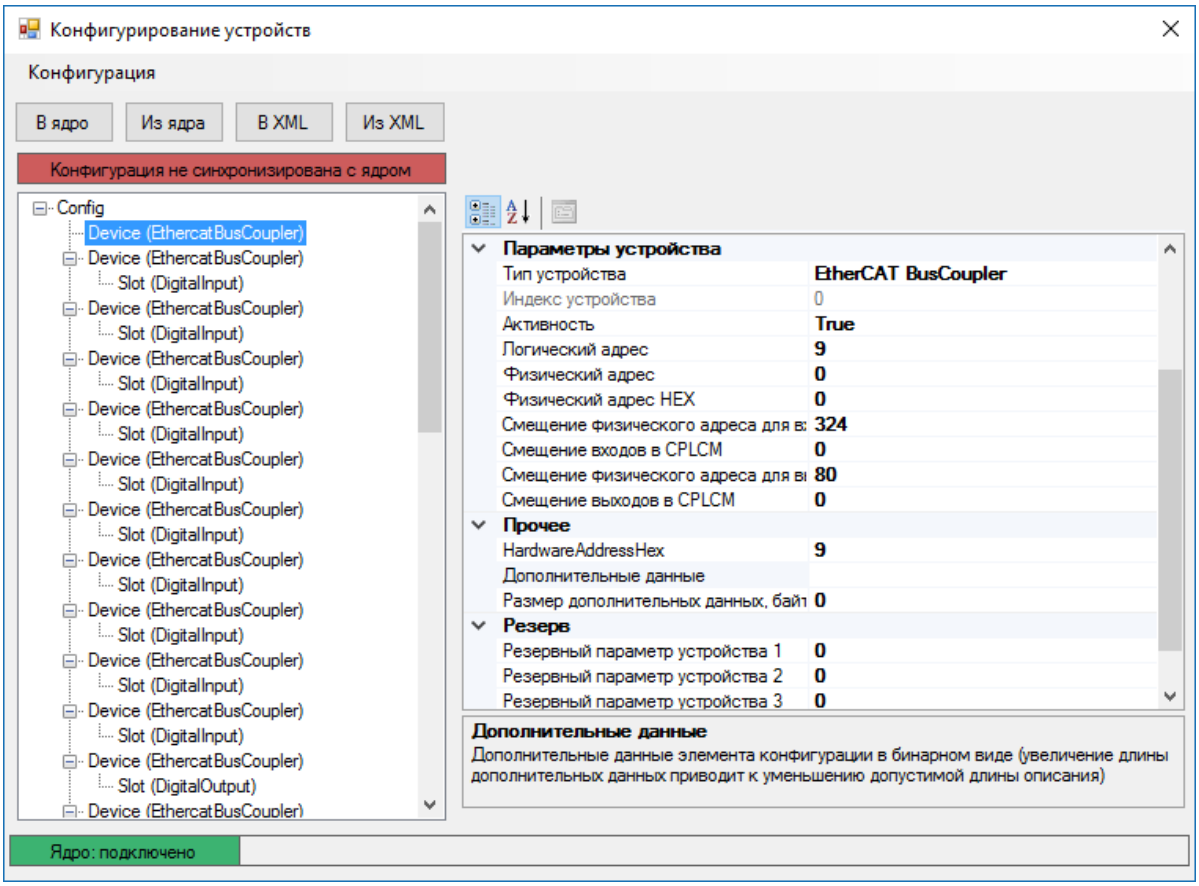



**Рисунок 38 – Пример созданной конфигурации устройств ввода/вывода**

После создания дерева конфигурации каждому из устройств необходимо назначить наиболее важные параметры, такие как:

- Тип устройства – определяет, какое устройство подключено и по какому протоколу работает.
- Активность – определяет, учитывается ли данное устройство при работе программы.
- Логический адрес – адрес, по которому данное устройство располагается в ряду устройств ввода/вывода.
- Смещение входов в CPLCM (Common Plc Memory) – адрес ячейки памяти, в которую будет записываться информация с входов устройств.
- Смещение выходов в CPLCM – адрес ячейки памяти, в которую контроллером будет записываться информация с выходов устройства.

## 5.4 Передача конфигурации в ядро и сохранение в файл

В нижней части конфигуратора отображается статус его подключения к ядру. При работающем подключении к ядру контроллера для возможности взаимодействия ПЛК с аппаратными устройствами необходимо отправить созданную конфигурацию в ядро. Это реализуется путём нажатия кнопки «В ядро». Для приёма имеющейся конфигурации из ядра следует нажать соответствующую кнопку. При идентичной конфигурации в терминале и ядре контроллера отображается соответствующее сообщение (Рисунок 39).

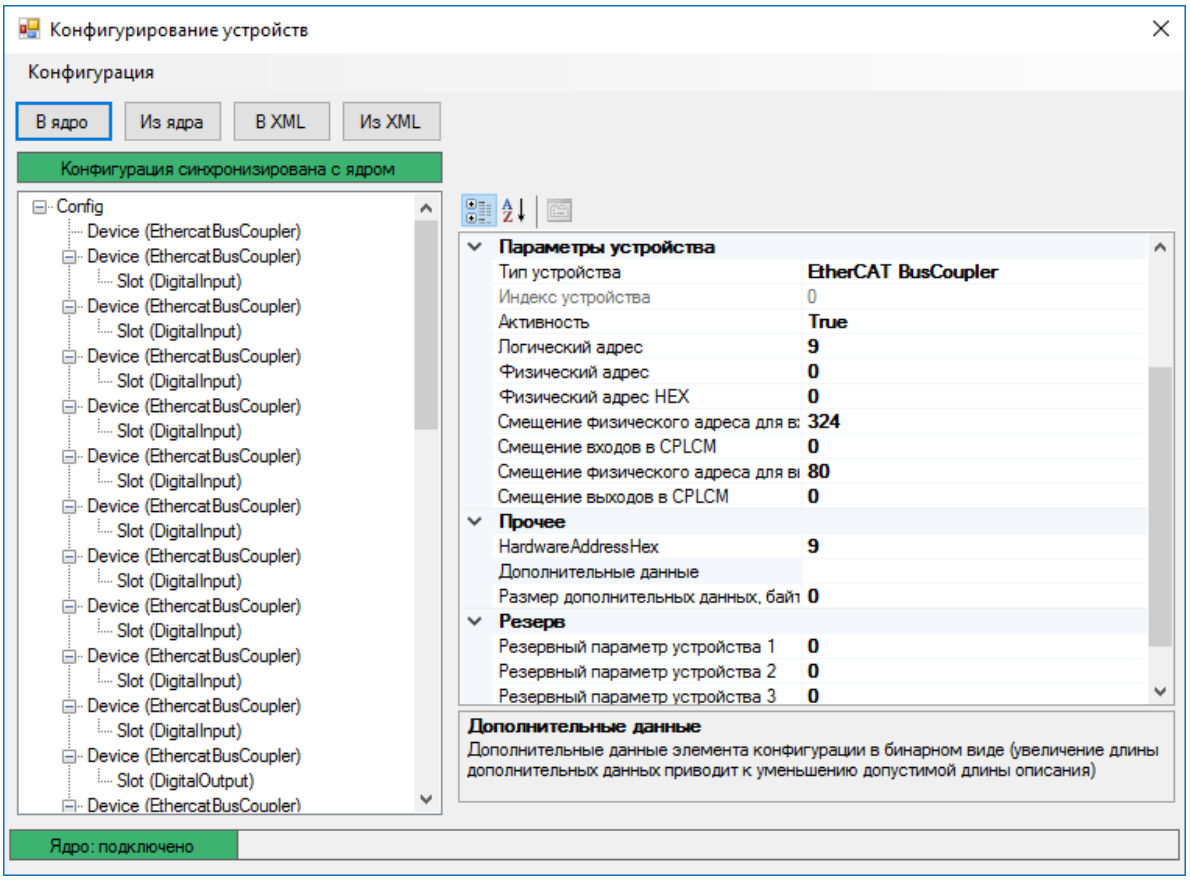



**Рисунок 39 – Синхронизированная с ядром контроллера конфигурация устройств ввода/вывода**

Конфигуратор устройств предоставляет возможности документирования созданной конфигурации путём сохранения в XML-файл, для чего в нём имеются соответствующие кнопки «В XML» – сохранение имеющейся конфигурации и «Из XML» – её загрузка из имеющегося файла.

## 5.5 Пример создания конфигурации на базе устройств BECKHOFF (EtherCAT)

### 5.5.1 Головной модуль EK1100

Головной модуль Beckhoff EK1100 (Рисунок 40) предназначен для организации взаимодействия между мастером сети и модулями ввода-вывода.

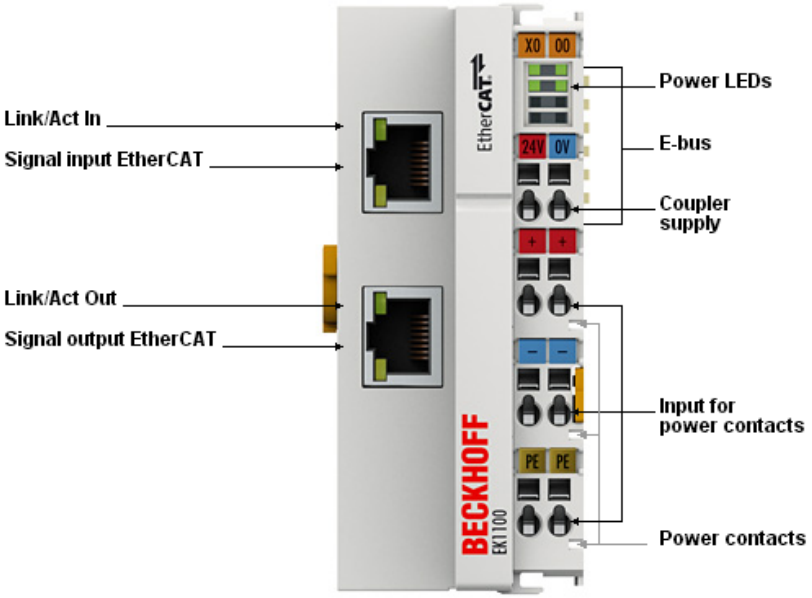



**Рисунок 40 – Головной модуль Beckhoff EK1100**

Для обеспечения синхронизации данных между модулем и ядром системы ЧПУ необходимо заполнить следующие поля конфигурации устройства:

1. **Device type** – *EthercatBusCouple*.
2. **Enabled –** *True*.
3. **InpCplcmOffset –** *204800*.

   Данный параметр указывает смещение в адресном пространстве памяти CommonPlcMemory для записи значений, полученных из входных слотов (дискретных, аналоговых) данного устройства.

   Например, указанное значение 204800 означает, что значения, считываемые из входных слотов, подключенных к данному устройству, будут проецироваться в область памяти CommonPlcMemory, начиная с байта № 204800.

4. **OutCplcmOffset –** 256000.

Аналогично предыдущему, данный параметр указывает смещение в адресном пространстве памяти CommonPlcMemory для записи значений, передаваемых на выходные слоты (дискретные, аналоговые) данного устройства.

Например, указанное значение 256000 означает, что значения, предназначенные для записи на выходы, подключенные к данному устройству, будут проецироваться из области памяти CommonPlcMemory, начиная с байта № 256000.

На Рисунок 41 представлен пример конфигурации устройства типа EtherCAT BusCoupler.

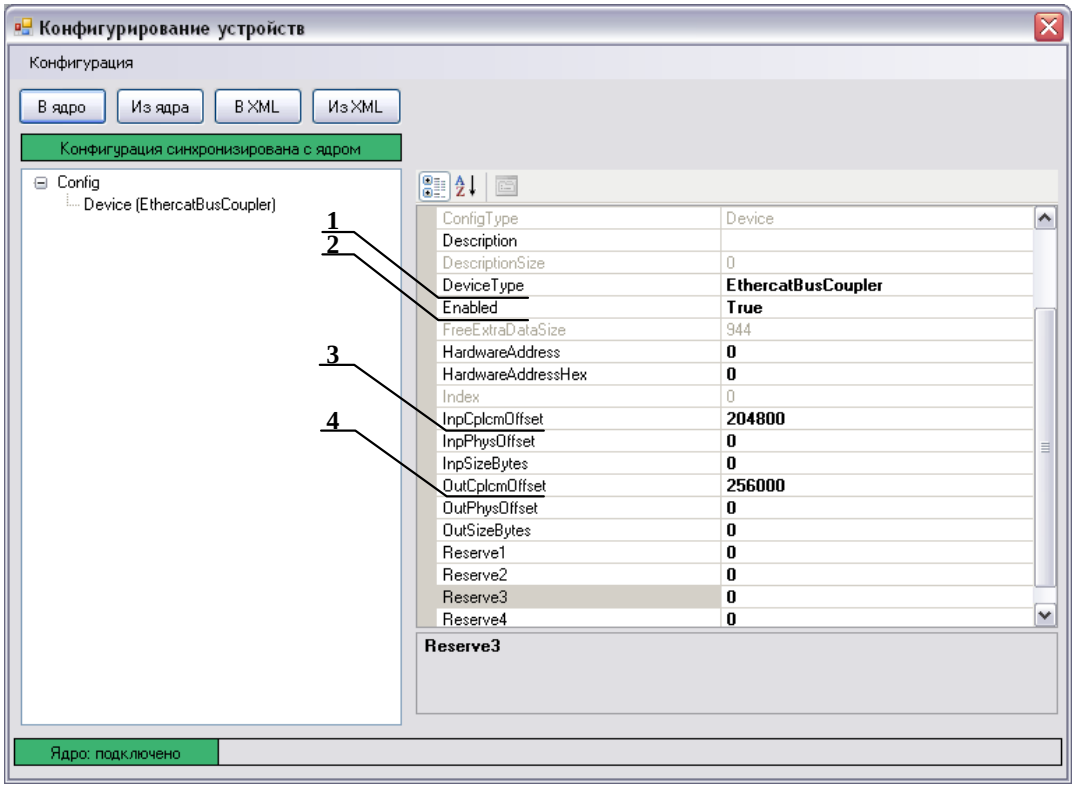



**Рисунок 41 – Пример конфигурации модуля EK1100**

## 5.5.2 Слот дискретных входов EL1008

8-канальный слот дискретных входов EL1008 (Рисунок 42) предназначен для приёма бинарных сигналов от объекта управления и передачи их в контроллер электроавтоматики.

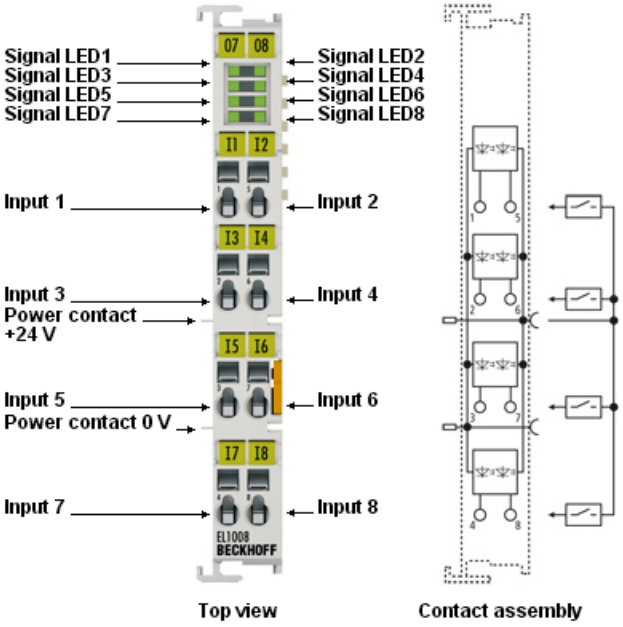



**Рисунок 42 – 8-канальный модуль дискретных входов EL1008**

Размер передаваемого пакета данных составляет 1 байт (8 бит). В Таблица 3 представлено распределение физических входов (I1, I2,…I8) слота EL1008 по битам.

**Таблица 3 – Распределение физических входов по битам**

| **№ входа** | **I1** | **I2** | **I3** | **I4** | **I5** | **I6** | **I7** | **I8** |
|---|---|---|---|---|---|---|---|---|
| **№ бита** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

В конфигурации слота необходимо указать следующие поля:

1. **InpLenBits** – длина пакета входных данных в битах.
   Например, длина пакета входных данных слота EL1008 равна 8.
2. **InpOffset** – смещение номера байта для входных данных относительно начала памяти CommonPlcMemory.
   Например, если слот дискретных входов установлен 1-ым, InpOffset = 0.
3. **InpSizeBytes** – длина пакета входных данных слота – целое число байт, вмещающих длину пакета входных данных слота.
   Например, длина пакета входных данных модуля EL1008 составляет 1 байт.
4. **SlotType** – тип конфигурируемого слота – *DigitalInput*.

На Рисунок 43 представлен пример конфигурации слота типа *Digital Input*.

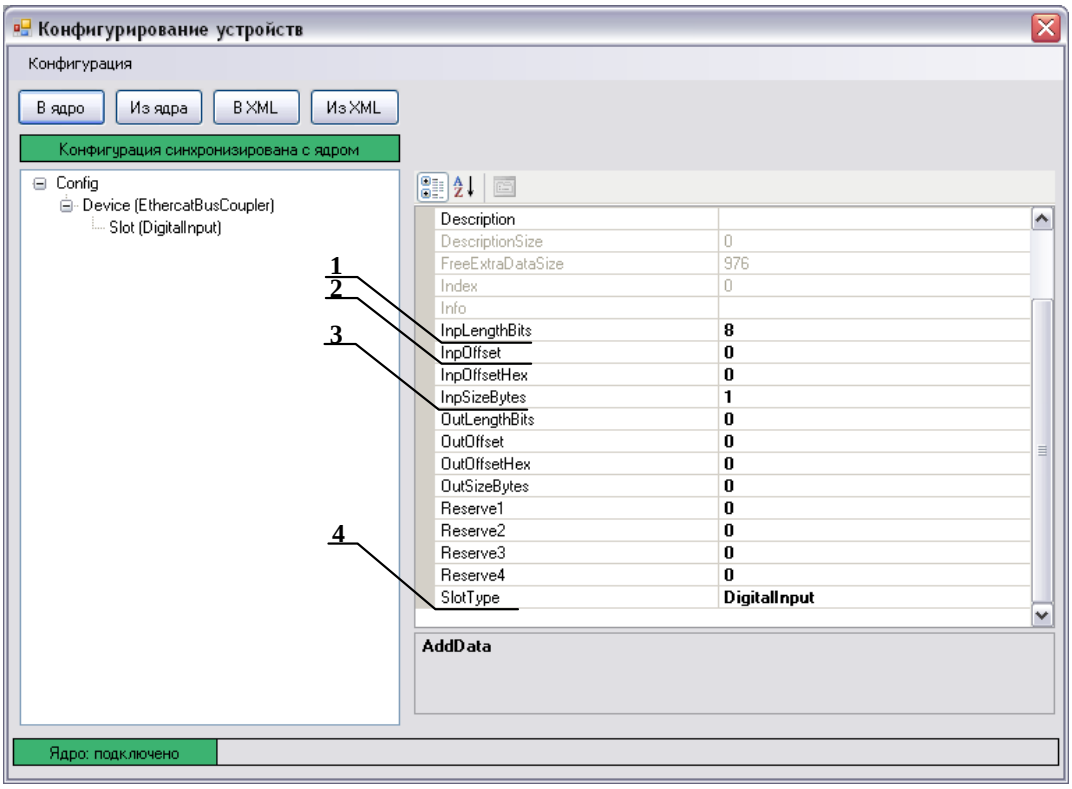



**Рисунок 43 – Пример конфигурации слота дискретных входов EL1008**

### 5.5.3 Слот дискретных выходов EL2008

8-канальный слот дискретных выходов EL2008 (Рисунок 44) предназначен для передачи объекту управления бинарных сигналов из контроллера электроавтоматики.

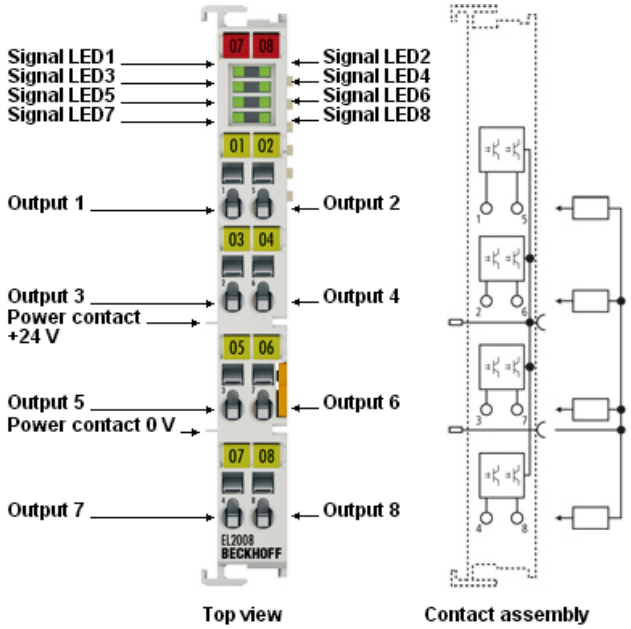



**Рисунок 44 – 8-канальный модуль дискретных выходов EL2008**

Размер передаваемого пакета данных составляет 1 байт (8 бит). В Таблица 4 представлено распределение физических входов (O1, O2,…O8) слота EL2008 по битам:

**Таблица 4 - Распределение физических выходов по битам**

| **№ входа** | **O1** | **O2** | **O3** | **O4** | **O5** | **O6** | **O7** | **O8** |
|---|---|---|---|---|---|---|---|---|
| **№ бита** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

В конфигурации слота необходимо заполнить следующие поля:

1. **OutLenBits** – длина пакета выходных данных в битах.
   Например, длина пакета выходных данных слота EL2008 равна 8.
2. **OutOffset** – смещение номера байта для входных данных относительно начала памяти CommonPlcMemory.
   Например, если слот дискретных входов установлен 1-ым, OutOffset = 0.
3. **OutSizeBytes** – длина пакета выходных данных слота – целое число байт, вмещающих длину пакета входных данных слота.
   Например, длина пакета входных данных модуля EL2008 составляет 1 байт.
4. **SlotType** – тип конфигурируемого слота – *DigitalOutput*.

На Рисунок 45 представлен пример конфигурации слота типа *Digital Output*.

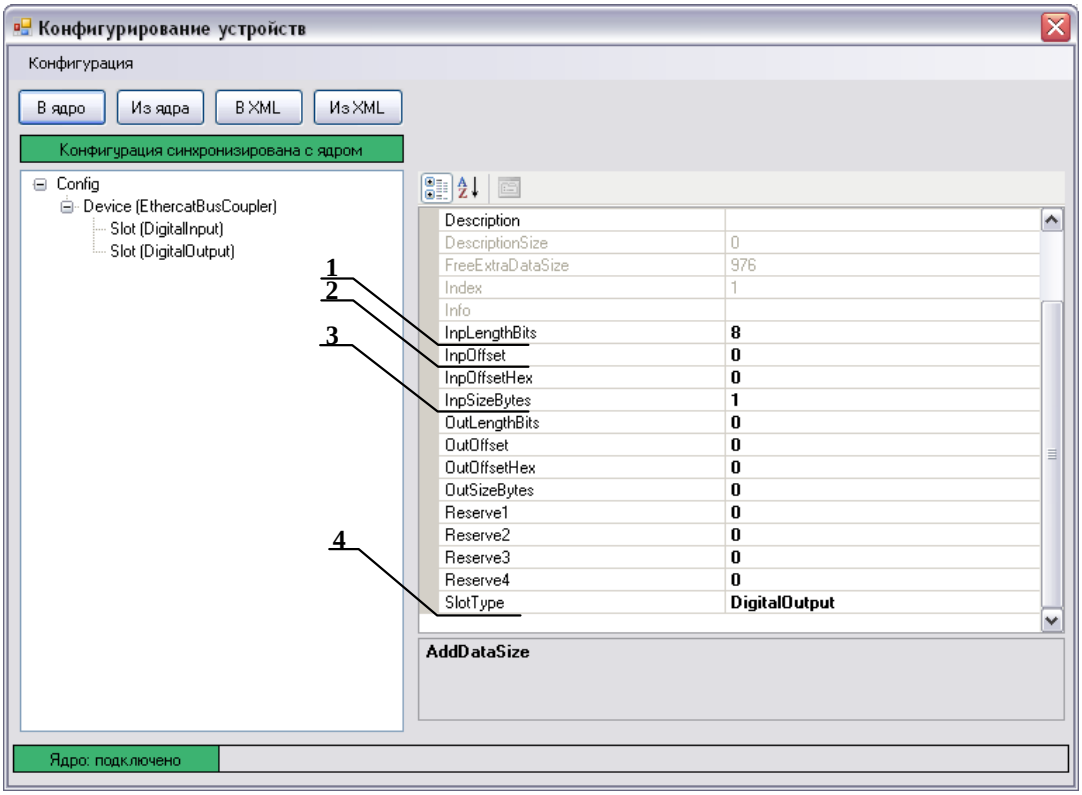



**Рисунок 45 – Пример конфигурации слота дискретных выходов EL2008**

## 5.5.4 Слот аналоговых входов EL3004

4-канальный слот аналоговых входов EL3004 (Рисунок 46) предназначен для приема аналоговых сигналов в диапазоне ±10В от объекта управления и передачи их в контроллер электроавтоматики.

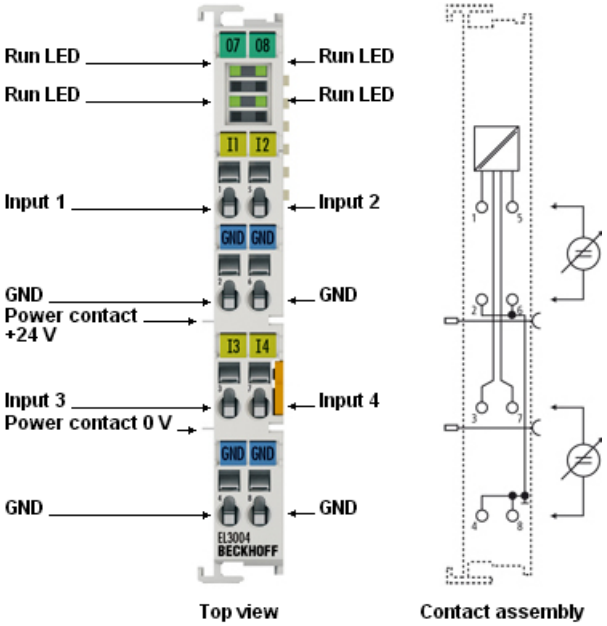



**Рисунок 46 – 4-канальный модуль аналоговых входов EL3004**

Размер передаваемого пакета данных составляет 16 байт (128 бит). В Таблица 5 представлено распределение физических входов (Input1 – Input4) слота EL3004 по байтам:

**Таблица 5 - Распределение физических входов по байтам**

| **№ входа** | **Input 1** | | **Input 2** | | **Input 3** | | **Input 4** | |
|---|---|---|---|---|---|---|---|---|
| **№ байта** | 2 | 3 | 6 | 7 | 10 | 11 | 14 | 15 |
| | Мл. байт | Ст. байт | Мл. байт | Ст. байт | Мл. байт | Ст. байт | Мл. байт | Ст. байт |

Байты 0, 1, 4, 5, 8, 9, 12, 13 не используются.

Значение на каждом входе хранится в двух байтах. Байт с меньшим порядковым номером является младшим, с большим порядковым номером – старшим.


**Рисунок 47 – Распределение битов аналогового входа**

Первые 4 бита младшего байта не используются. 7 бит старшего байта служит для обозначения знака («0» – положительное число; «1» – отрицательное число). Остальные биты используются для хранения значения.

Для преобразования числа A в напряжение номиналом U (U > 0) необходимо выполнить следующее преобразование:

$$U = \frac{10 * A}{2047 * 16} = \frac{A}{3275{,}2}$$

Например, если в старшем и младшем байтах записаны числа 127 (0x7F) и 240 (0xF0) соответственно, то число A = 32752 (0x7FF0) соответствует напряжению номиналом +10В.

Для преобразования числа A в напряжение номиналом U (U < 0) необходимо выполнить следующее преобразование:

$$U = \frac{65520 - A}{3275{,}2}$$

Например, если в старшем и младшем байтах записаны числа 128 (0x80) и 0 (0x00) соответственно, то число A = 32768 (0x8000) соответствует напряжению -10В.

В конфигурации слота необходимо заполнить следующие поля:

1. **InpLenBits** – длина пакета входных данных в битах.
   Длина пакета входных данных слота EL3004 равна 128.
2. **InpOffset** – смещение номера байта для входных данных относительно начала памяти CommonPlcMemory.
   Например, если слот аналоговых входов EL3004 установлен после слота дискретных входов EL1008, то InpOffset = 1.
3. **InpSizeBytes** – длина пакета входных данных слота – целое число байт, вмещающих длину пакета входных данных слота.
   Длина пакета входных данных слота EL3004 составляет 16 байт.
4. **SlotType** – тип конфигурируемого слота – *AnalogInput*.

На Рисунок 48 представлен пример конфигурации слота типа *Analog Input*.

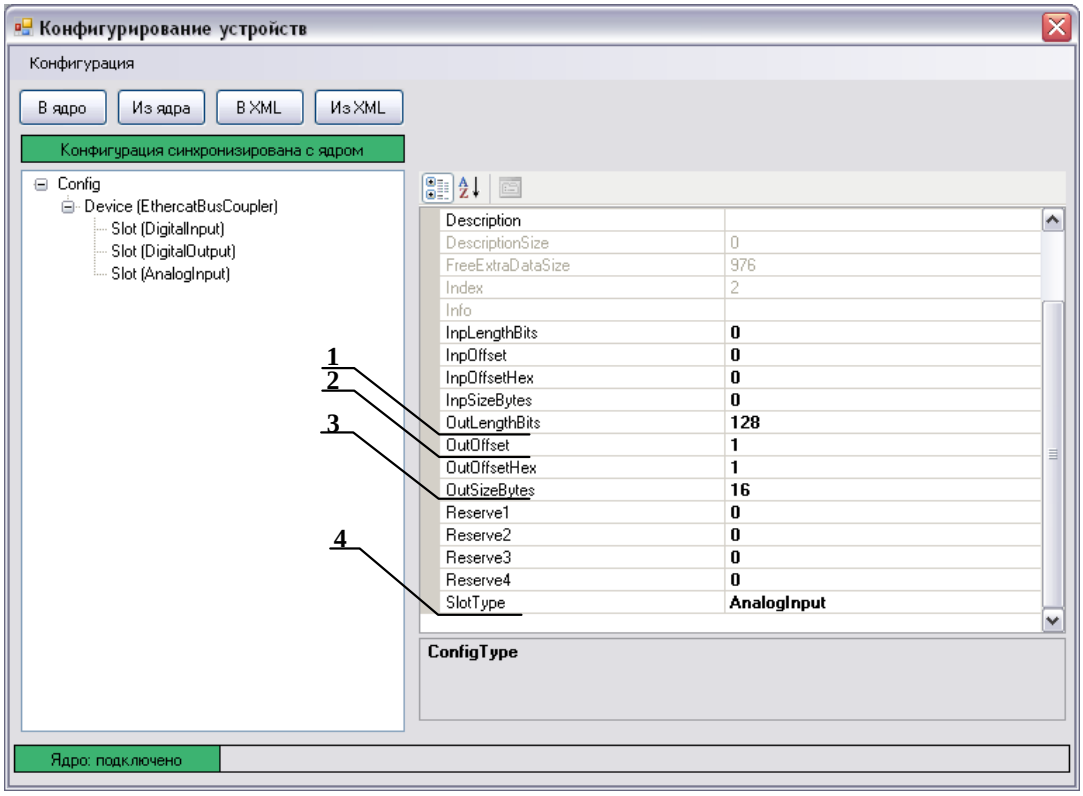



**Рисунок 48 – Пример конфигурации слота аналоговых входов EL3004**

### 5.5.5 Слот аналоговых выходов EL4034

4-канальный слот аналоговых выходов EL4034 (Рисунок 49) предназначен для выдачи объекту управления аналоговых сигналов в диапазоне ±10В из контроллера электроавтоматики.

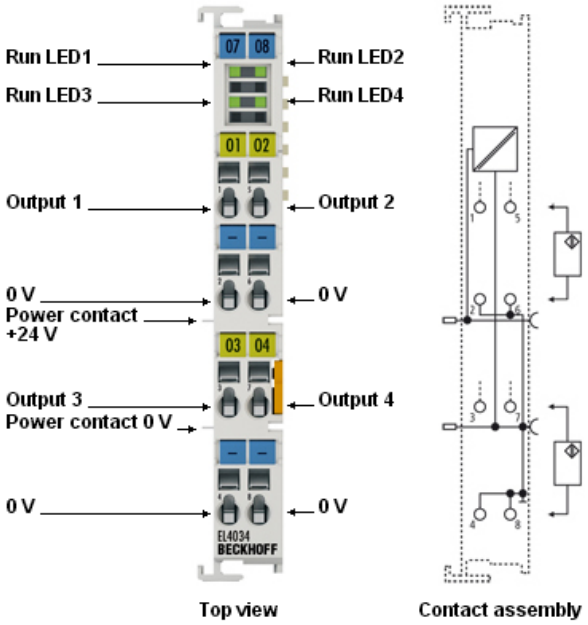



**Рисунок 49 – 8-канальный модуль аналоговых выходов EL4034**

4-канальный слот аналоговых выходов EL4034 (Рисунок 49) предназначен для выдачи объекту управления аналоговых сигналов в диапазоне ±10В из контроллера электроавтоматики.

Размер передаваемого пакета данных составляет 8 байт (4 бит). В Таблица 6 представлено распределение физических выходов (O1, O2, O3 и O4) слота EL4034 по байтам:

**Таблица 6 – Распределение физических входов по байтам**

| **№ выхода** | **Output 1** | | **Output 2** | | **Output 3** | | **Output 4** | |
|---|---|---|---|---|---|---|---|---|
| **№ байта** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | Мл. байт | Ст. байт | Мл. байт | Ст. байт | Мл. байт | Ст. байт | Мл. байт | Ст. байт |

Значение каждого выхода хранится в двух байтах. Байт с меньшим порядковым номером является младшим, с большим порядковым номером – старшим.

| **Старший байт** | | | | | | | | **Младший байт** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Знак | Значение | | | | | | | Не используются | | | | | | | |

**Рисунок 50 – Распределение битов аналогового выхода**

Первые 4 бита младшего байта не используются. 7 бит старшего байта служит для обозначения знака («0» – положительное число; «1» – отрицательное число). Остальные биты используются для хранения значения.

Для выдачи напряжения номиналом U (U > 0) необходимо записать в старший и младший байты число A, рассчитываемое по формуле:

$$A = \frac{2047 * U}{10} * 16 = 3275{,}2 * U$$

Например, для выдачи напряжения номиналом +10В необходимо записать число A = 32752 (0x7FF0), что соответствует записи значений 127 (0x7F) и 240 (0xF0) в старший и младший байты соответственно.

Для выдачи напряжения номиналом U (U < 0) необходимо записать в старший и младший байты число A, рассчитываемое по формуле:

$$A = 65520 - \frac{2047 * U}{10} * 16 = 65520 - 3275{,}2 * U$$

Например, для выдачи напряжения номиналом -10В необходимо записать число A = 32768 (0x8000), что соответствует записи значений 128 (0x80) и 0 (0x00) в старший и младший байты соответственно.

| Номинальное напряжение, В | Значение старшего байта | | | | | | | | | | Значение младшего байта | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bin | | | | | | | | Dec | Hex | Bin | | | | | | | | Dec | Hex |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x00 | 0 | 0 | 0 | 0 | - | - | - | - | 0 | 0x00 |
| +10 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 127 | 0x7F | 1 | 1 | 1 | 1 | - | - | - | - | 240 | 0xF0 |
| -10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 128 | 0x80 | 0 | 0 | 0 | 0 | - | - | - | - | 0 | 0x00 |

В конфигурации слота необходимо инициализировать следующие поля:

1. **OutLenBits** – длина пакета выходных данных в битах.

   Длина пакета входных данных слота EL4034 равна 64.

2. **OutOffset** – смещение № байта для входных данных относительно начала памяти CommonPlcMemory.

   Например, если слот аналоговых выходов EL4034 установлен после слота дискретных выходов EL2008, то OutOffset = 1.

3. **OutSizeBytes** – длина пакета входных данных слота – целое число байт, вмещающих длину пакета входных данных слота.

   Длина пакета входных данных слота EL4034 составляет 8 байт.

4. **SlotType** – тип конфигурируемого слота – *AnalogOutput*.

На Рисунок 51 представлен пример конфигурации слота типа *Analog Output*.

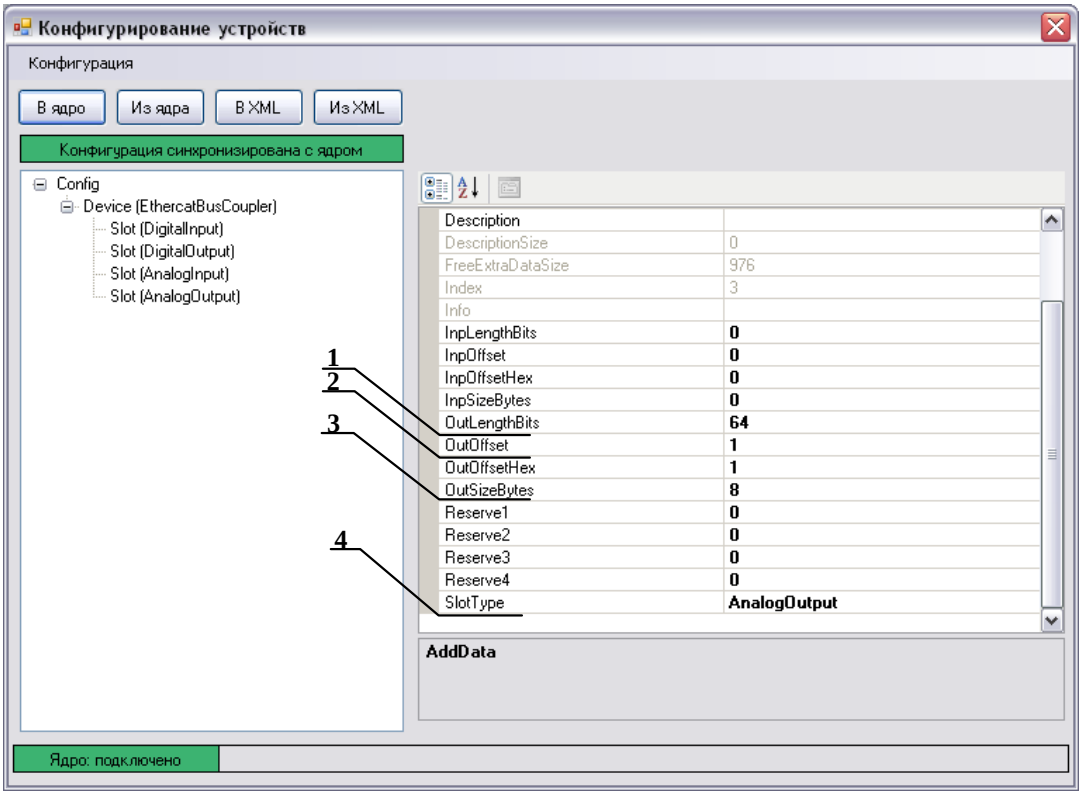



**Рисунок 51 – Пример конфигурации слота аналоговых выходов EL4034**

# 6 Пример управляющей программы для управления электроавтоматикой станка

На Рисунок 52 представлен пример управляющей программы для ПЛК, созданной для управления основными узлами электроавтоматики вертикально-фрезерного обрабатывающего центра.

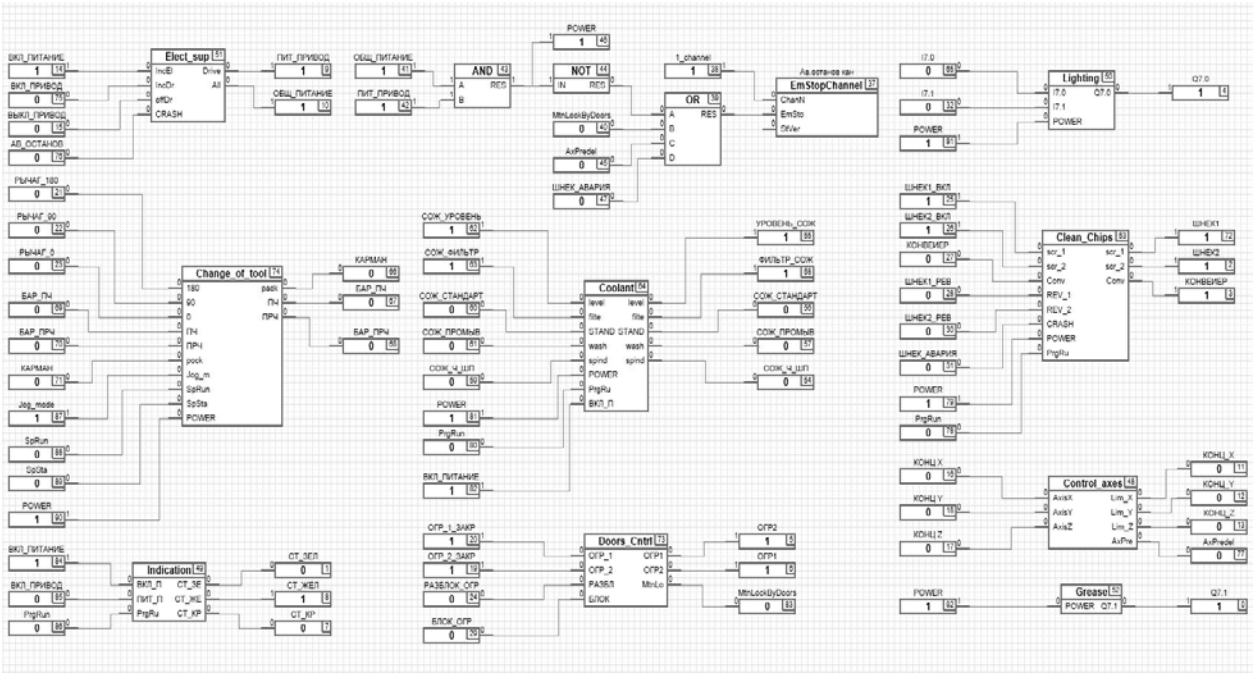



**Рисунок 52 – Пример управляющей программы**

В программе можно выделить ряд пользовательских библиотек, каждая из которых отвечает за управление определённым узлом электроавтоматики станка:

- Управление питанием – блок «Elect_Sup».
- Управление индикацией работы станка – блок «Indication».
- Управление автоматической сменой инструмента – блок «Change_of_tool».
- Управление защитными ограждениями – блок «Doors_Cntrl».
- Управление механизмом подачи СОЖ – блок «Coolant».
- Управление механизмом уборки стружки – блок «Clean_chips».
- Управление концевыми выключателями – блок «Control_axes»;
- Управление освещение и смазкой направляющих – блок «Grease».