

**Система ЧПУ «АксиОМА Контрол»
Руководство программиста по созданию
программ для ПЛК**

Содержание

1	Назначение документа	4
1.1	Список сокращений	4
2	Базовые принципы построения и функционирования программно реализованного контроллера	5
3	Редактор программ ПЛК	7
3.1	Запуск программного пакета и подключение к вычислительному ядру	7
3.2	Создание/загрузка проекта	9
3.3	Базовый интерфейс редактора программ ПЛК	10
3.4	Работа с функциональными блоками	12
3.5	Работа с менеджером ПЛК переменных	17
3.6	Работа со связями между функциональными блоками	25
3.7	Создание пользовательских библиотек	26
3.8	Импорт и использование пользовательских блоков	30
3.9	Замена пользовательских блоков	33
3.10	Создание регионов	34
3.11	Настройки ПЛК редактора	37
3.12	Верификация, запуск и отладка программы	39
4	Описание функциональных блоков	42
4.1	Блоки входов и выходов	42
4.1.1	Вход	42
4.1.2	Выход	42
4.1.3	Вход с синхронизацией	42
4.1.4	Выход с синхронизацией	42
4.2	Логические блоки	43
4.2.1	AND: дизъюнкция (И, логическое умножение)	43
4.2.2	OR: конъюнкция (ИЛИ, логическое сложение)	43
4.2.3	NOT: отрицание (логическое НЕ)	43
4.2.4	XOR: исключающее ИЛИ	44
4.2.5	SHL: побитовый сдвиг операнда влево	44
4.2.6	SHR: побитовый сдвиг операнда вправо	45
4.2.7	PACK	45
4.2.8	UNPACK	45
4.3	Арифметические блоки	46
4.3.1	ADD: сложение	46
4.3.2	SUB: вычитание	46
4.3.3	MUL: умножение	47
4.3.4	DIV: деление	47
4.3.5	COMPARE: сравнение	48
4.3.6	MAX: определение максимального из значений	48
4.3.7	MIN: определение минимального из значений	48
4.3.8	ABS: получение значения по модулю	49
4.3.9	DOUBLE_TO_INT: преобразование	49
4.4	Триггеры	49
4.4.1	RS-trig: RS триггер	49
4.4.2	D-trig: D триггер (триггер задержки)	50
4.4.3	SR-trig: SR триггер	50
4.4.4	R-trig: импульс по переднему фронту	50
4.4.5	F-trig: импульс по заднему фронту	51
4.5	Счетные блоки	51
4.5.1	COUNTER: счетчик	51

4.5.2	COUNTER_ROUND: двунаправленный счетчик.....	52
4.5.3	TIMER2: простой таймер.....	52
4.5.4	TIMER: таймер	53
4.5.5	BLINK: генератор прямоугольных импульсов.....	53
4.6	Функции	54
4.6.1	MUX: мультиплексор.....	54
4.7	Управление движением.....	54
4.7.1	CDRIVE: блок управления приводом	54
5	Конфигуратор устройств ввода/вывода	56
5.1	Назначение и базовые принципы работы модуля конфигурирования устройств ввода/вывода	56
5.2	Базовый интерфейс конфигуратора устройств ввода/вывода	56
5.3	Создание конфигурации устройств ввода/вывода	57
5.4	Передача конфигурации в ядро и сохранение в файл.....	58
5.5	Пример создания конфигурации на базе устройств ODOT (EtherCAT)	58

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 контроллера

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.

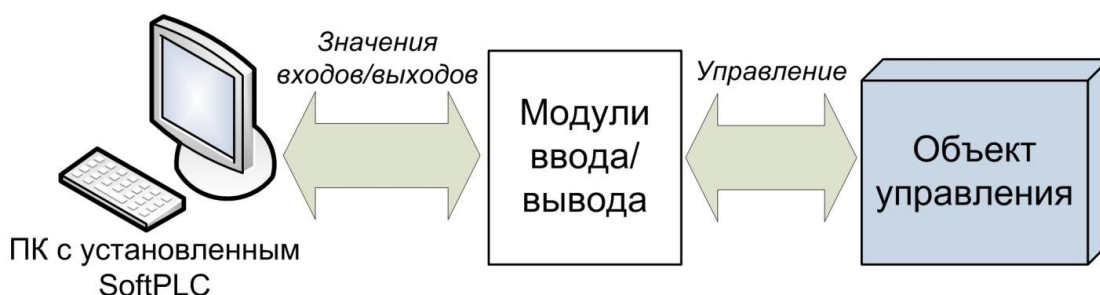


Рисунок 1 – Схема управления объектом с помощью программно реализованного контроллера (SoftPLC)

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. Особенности 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.

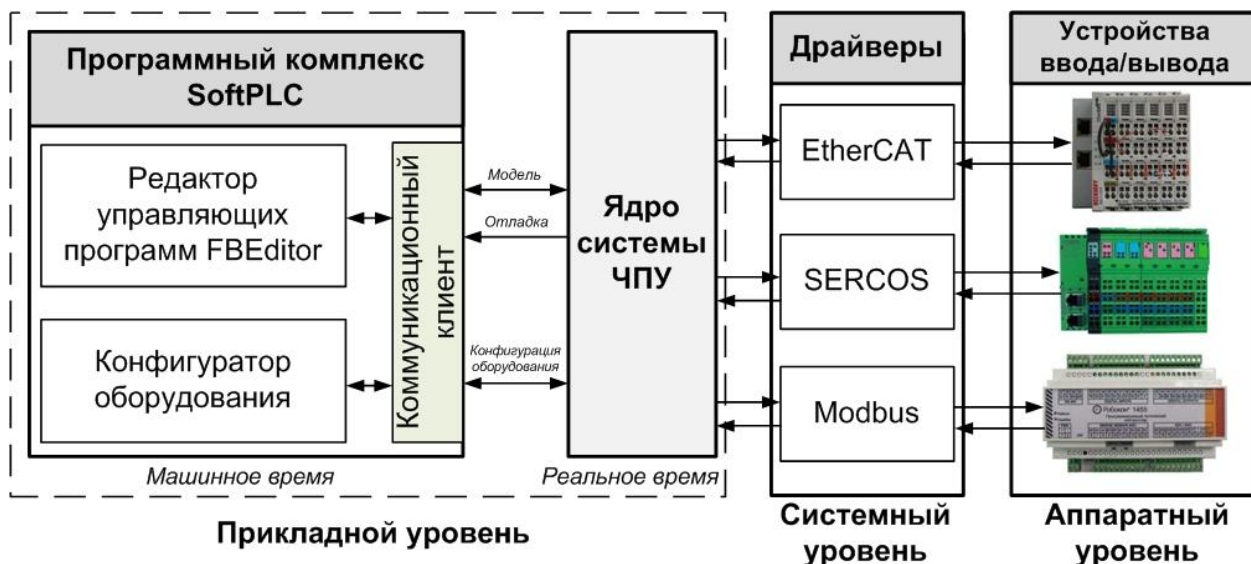


Рисунок 2 – Архитектура программно реализованного контроллера

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 инструментальный 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 конфигулятора 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).

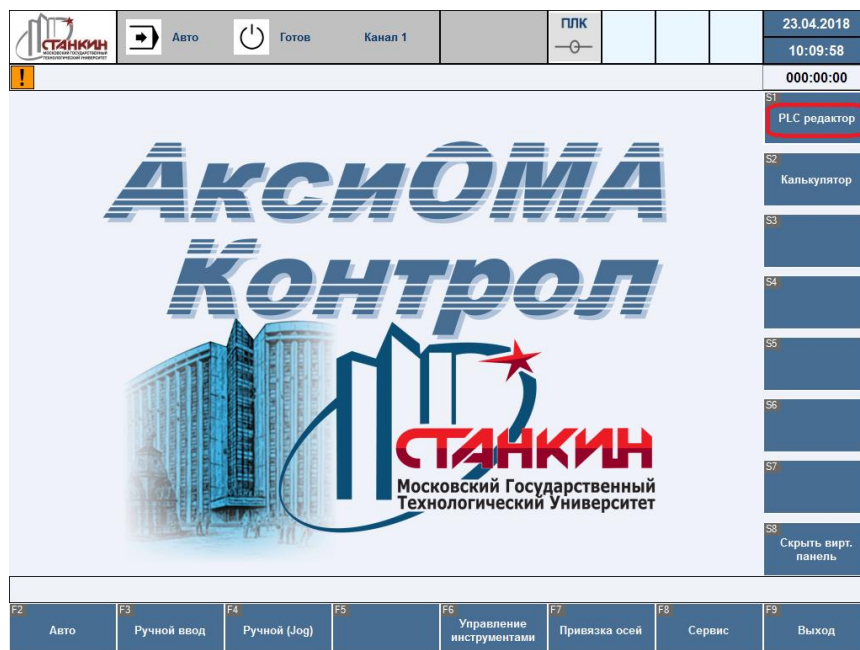


Рисунок 3 – Запуск редактора программ ПЛК в системе ЧПУ «АксиОМА Контрол»

Базовый экран при запуске редактора FBEEditor 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.

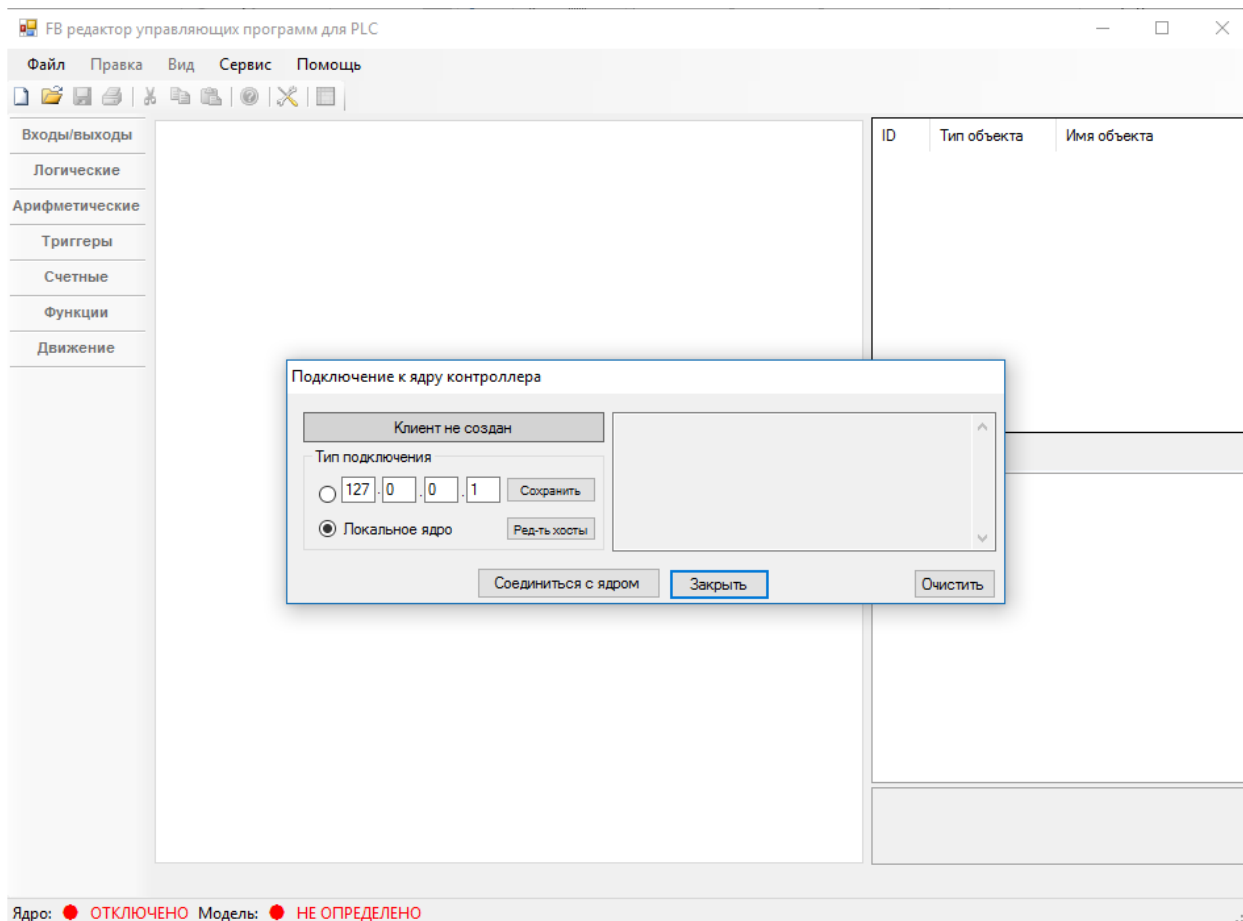


Рисунок 4 – Общий вид программного пакета при запуске

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.

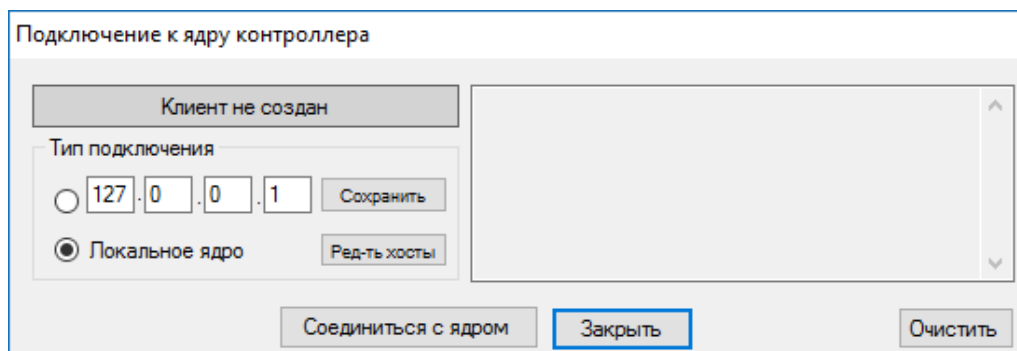


Рисунок 5 – Встроенный клиент подключения к вычислительному ядру контроллера

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).

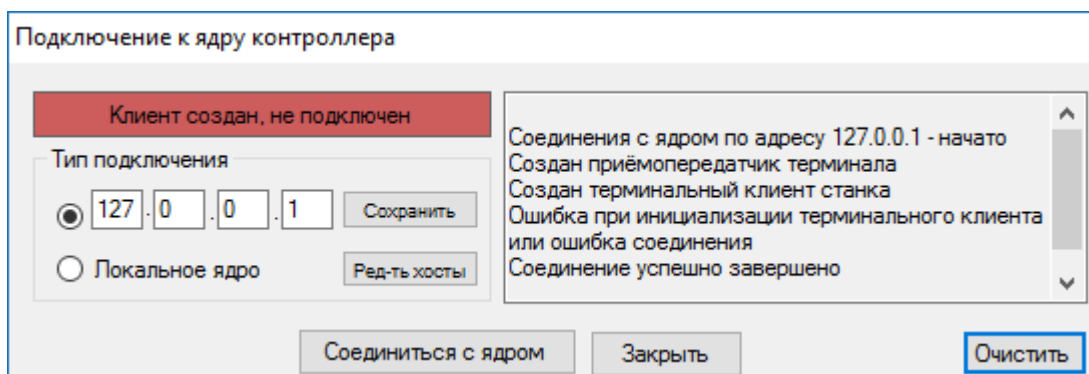


Рисунок 6 – Вид окна клиента подключения при неудачном соединении с ядром

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

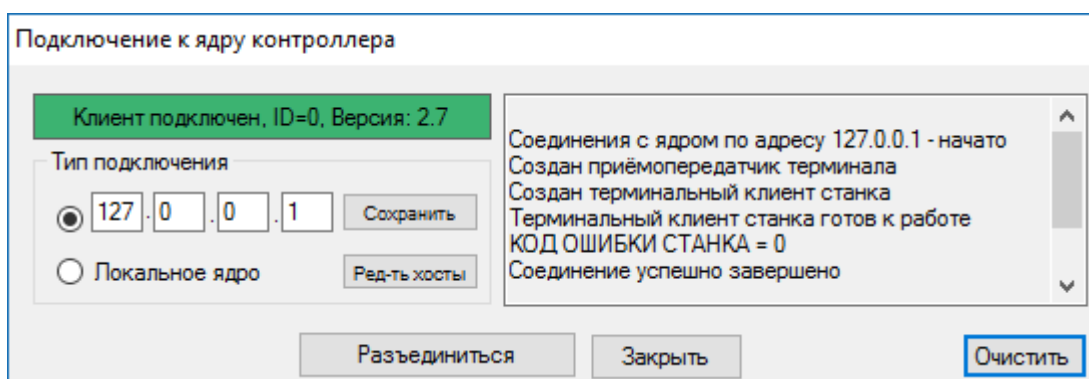


Рисунок 7 – Вид окна клиента подключения при успешном соединении с ядром

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.

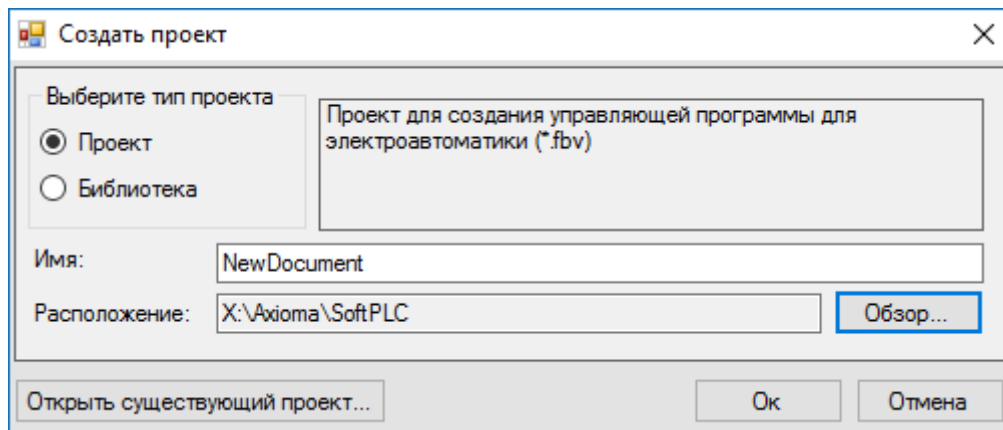


Рисунок 8 – Диалоговое окно создания/загрузки проекта

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.

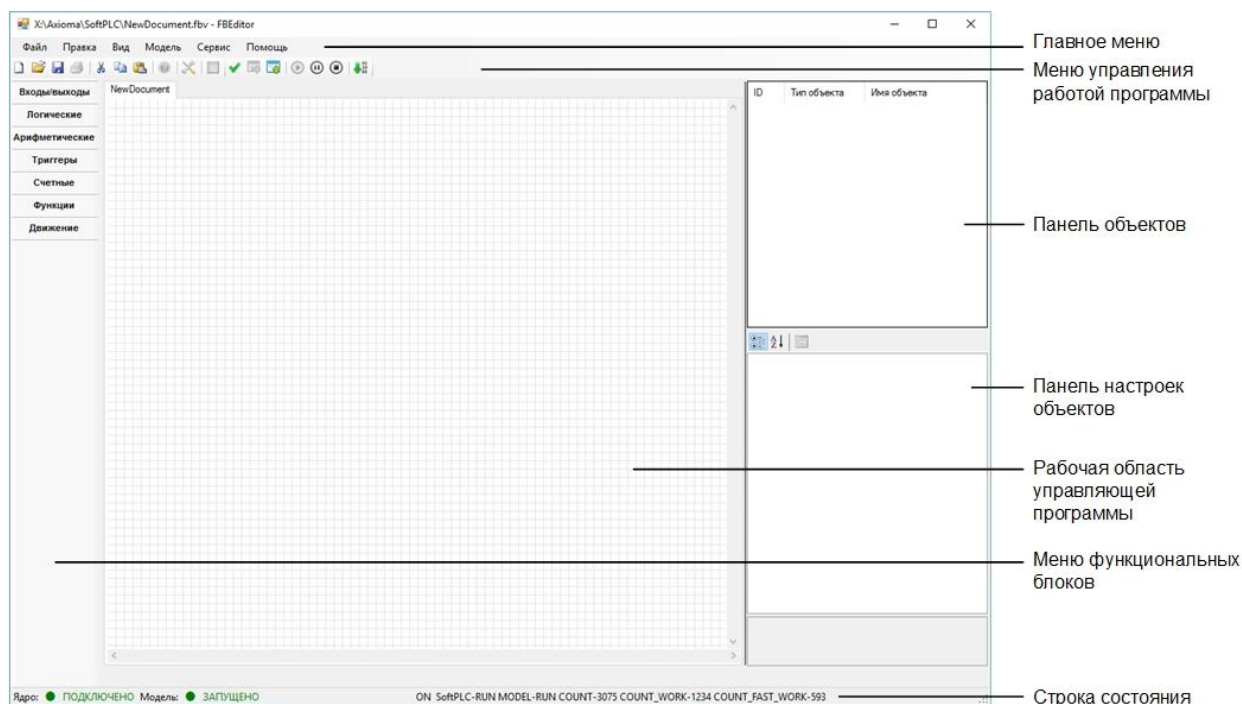


Рисунок 9 – Интерфейс основной среды разработки программ ПЛК

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».

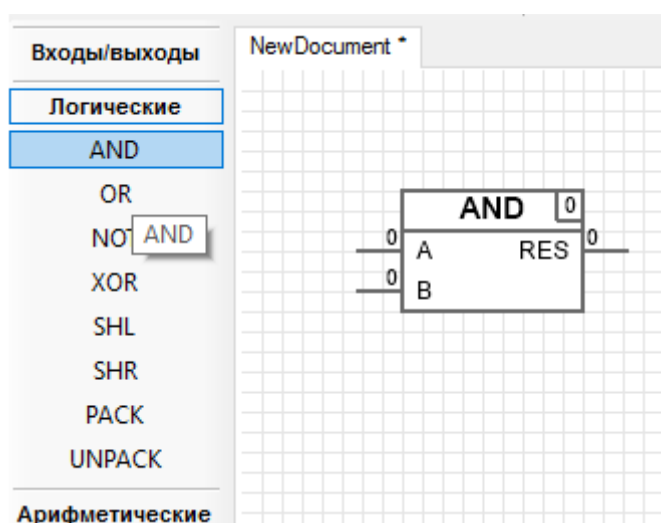


Рисунок 10 – Добавление функционального блока AND в программу

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.

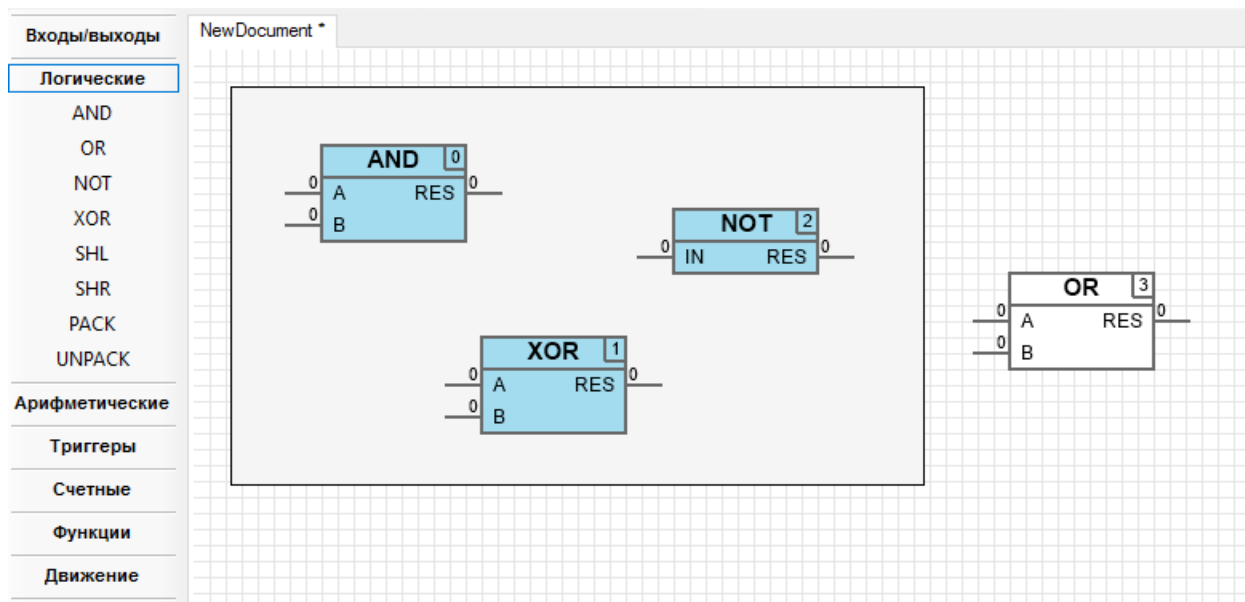


Рисунок 11 – Множественное выделение функциональных блоков

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.

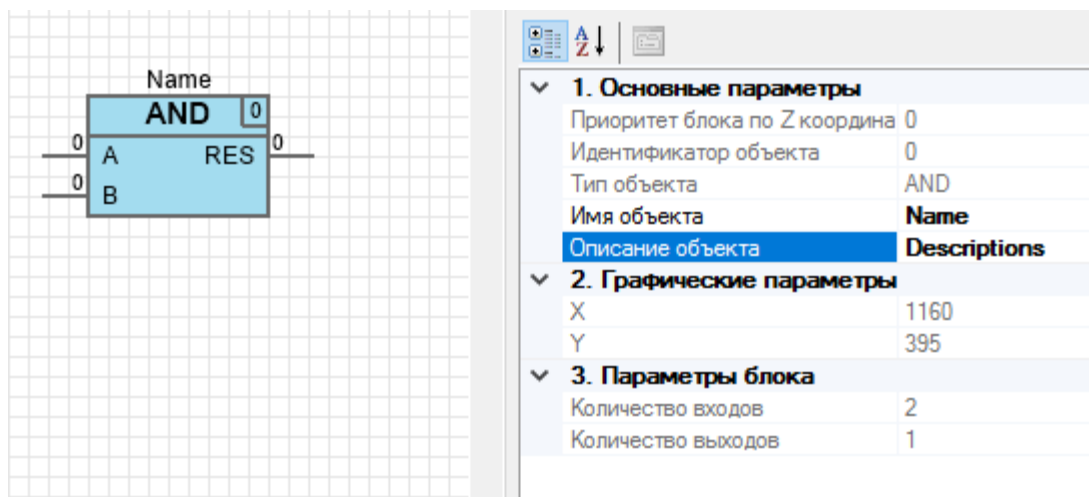


Рисунок 12 – Создание имени и описания функциональных блоков

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

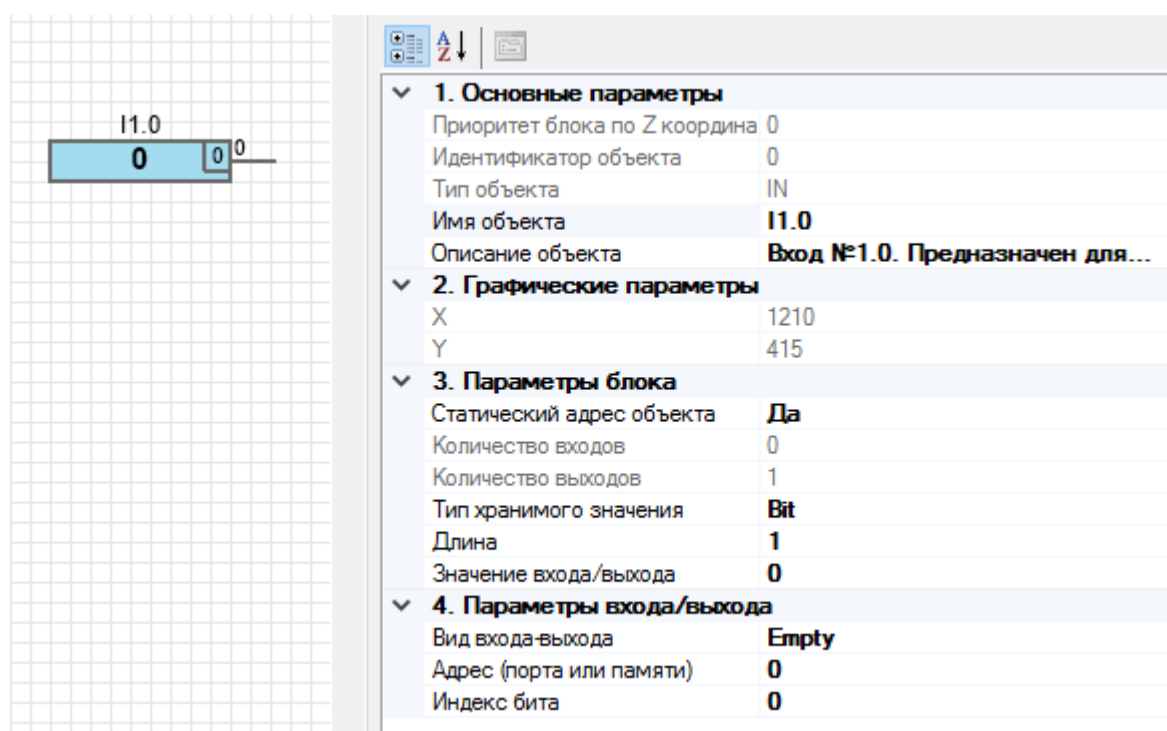


Рисунок 13 – Настройки входа

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. Возможные значения: <ul style="list-style-type: none"> - None – не задано; - Bit – булевское число; - Analog – число с плавающей точкой; - 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: <ul style="list-style-type: none"> - Empty – задается для реализации константных значений. Такие входы/выходы не принимают и не передают значений в память ядра. - CommonPlcMemory – ссылка на разделяемую память ядра контроллера. Выбирается для работы с аппаратными модулями ввода/вывода. - 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).

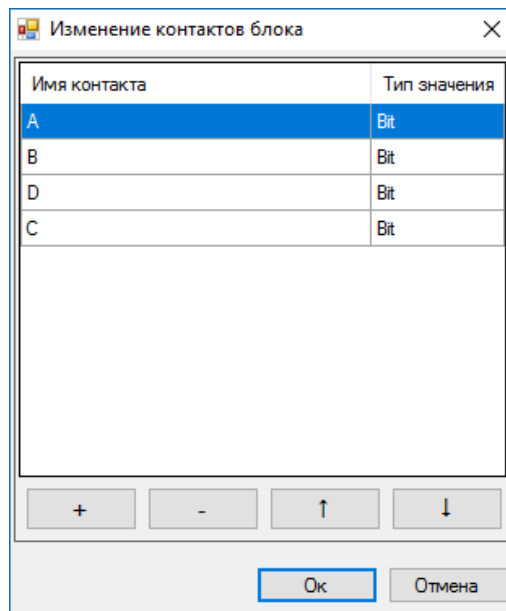


Рисунок 14 – Изменение входных контактов логических и арифметических блоков

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 с менеджером ПЛК переменных

В состав ПЛК редактора входит менеджер ПЛК переменных, позволяющий управлять (создавать и редактировать) ПЛК переменные и использовать их в дальнейшем в ПЛК программе. ПЛК переменная представляет собой сохраненный участок памяти в CommonPcMemory, для которого заданы основные параметры и которым можно оперировать в программе. Менеджер ПЛК переменных позволяет сохранять список переменных, необходимых для конкретной программы, а также производить мониторинг их состояния в режиме отладки программы.

Для вызова менеджера ПЛК переменных используется иконка с таблицей или в меню «Сервис – Менеджер переменных».

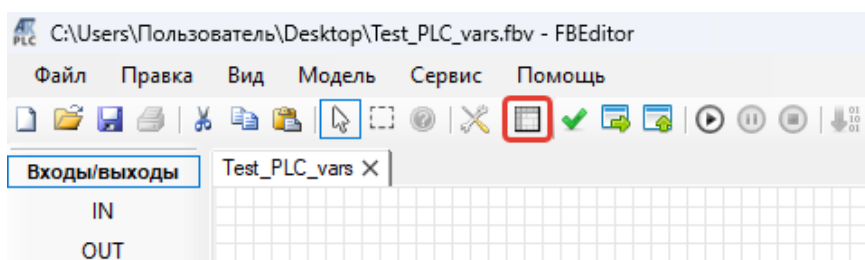


Рисунок 15 –Вызов менеджера ПЛК переменных

Менеджер ПЛК переменных представляет собой окно, в котором для пользователя реализованы функции по управлению переменными, а также их просмотру с учетом заданных фильтров.

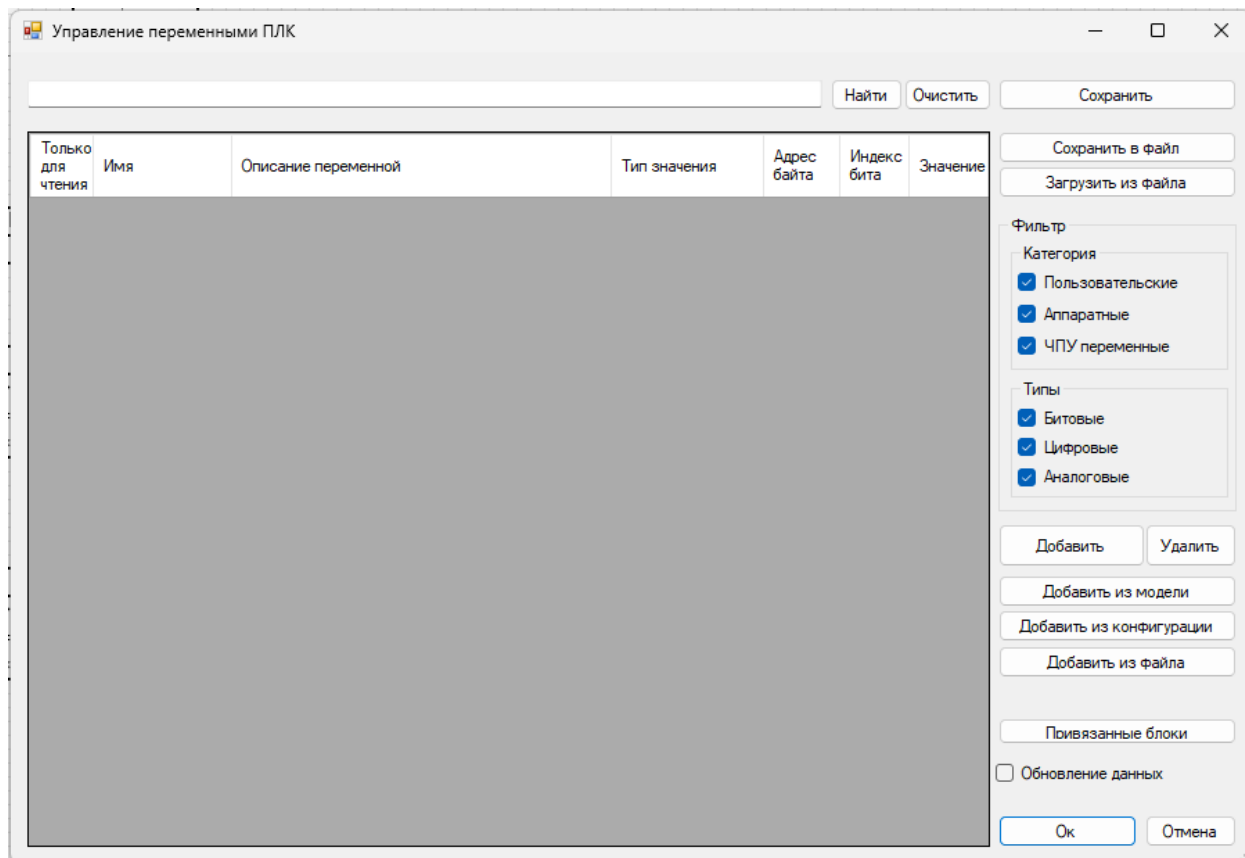


Рисунок 16 – Менеджер ПЛК переменных

Основные функции, предоставляемые пользователю в менеджере ПЛК переменных:

- создание новой переменной;
- удаление имеющейся переменной;
- сохранение набора переменных в отдельный файл (с расширением «*.io»);
- загрузка набора переменных из файла;
- поиск переменных в списке;
- фильтрация переменных в зависимости от категории, к которой она относится и типа хранимого значения;
- добавление набора переменных из ПЛК программы – производится создание набора переменных для всех блоков входов/выходов, содержащихся в ПЛК программе;
- добавление набора переменных из конфигурации – при вызове данной функции пользователю предлагается выбрать файл конфигурации устройств входов/выходов, на основании которого производится создание набора переменных, соответствующих входам и выходам устройств, заданных в конфигурации;
- добавление переменных из файла – позволяет расширить текущий набор переменных другими переменными, сохраненными в файл «*.io»;
- показать привязанные блоки – позволяет отобразить список блоков основной ПЛК программы, которые привязаны к выбранной переменной;
- включить обновление данных, при котором из ядра ПЛК считываются актуальные значения переменных и отображаются в таблице, что необходимо для проведения отладки работы ПЛК программы.

При создании новой переменной необходимо задать ее основные параметры: наименование, краткое описание, тип хранимого значения, адрес байта и индекс бита для привязки к ячейке памяти CommonPlcMemory.

ПЛК переменные могут относиться к одной из трех категорий:

- пользовательские (созданные оператором);
- ЧПУ переменные (для получения информации о работе системы);
- аппаратные (привязанные к модулям ввода/вывода).

Тип ПЛК переменной устанавливается автоматически – при создании переменной пользователем он сохраняется как «пользовательский», а в случае создания переменных на основе файла конфигурации категория задается как «аппаратная».

При нажатии на кнопку «Добавить» в таблице переменных появляется новая строка, которую необходимо заполнить соответствующими параметрами новой переменной.

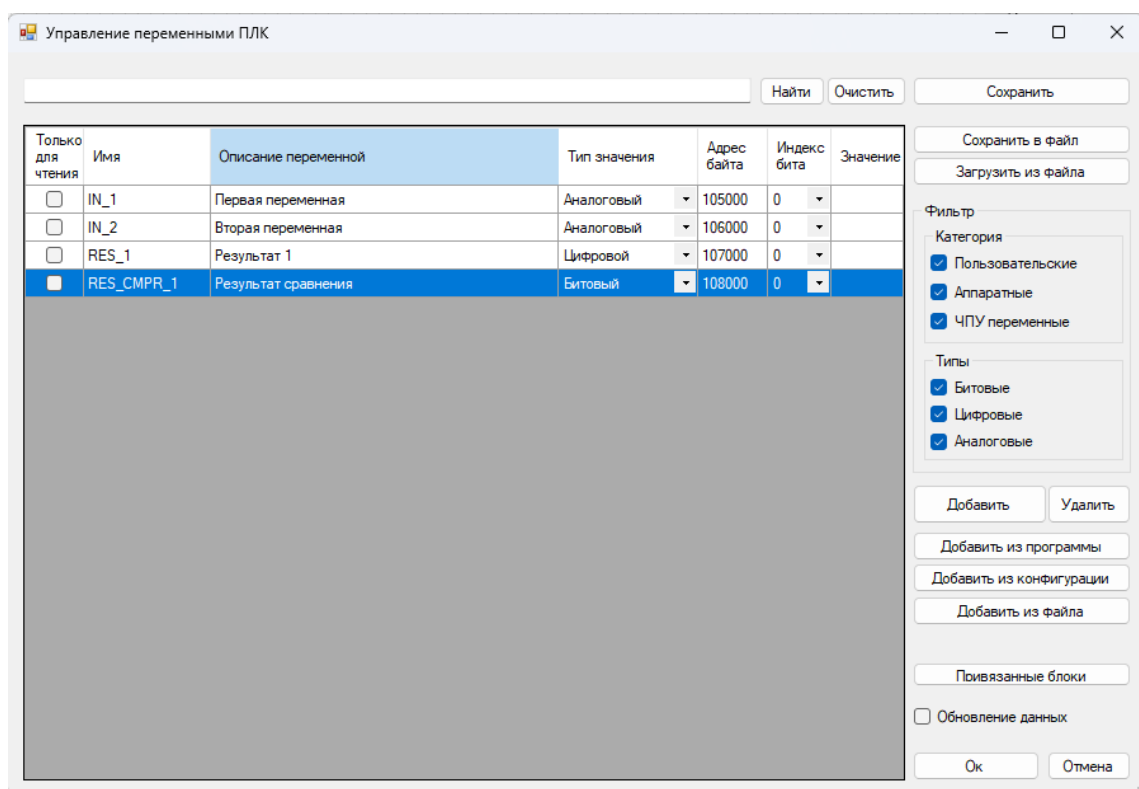


Рисунок 17 – Создание новых переменных

Список переменных автоматически привязывается к файлу ПЛК программы и загружается при ее открытии. Также имеется возможность сохранить список переменных в отдельный файл «*.io», для чего необходимо нажать кнопку «Сохранить в файл». В дальнейшем этот файл можно открыть и привязать к ПЛК программе, либо добавить к уже имеющемуся списку переменных.

После внесения изменений в состав переменных при закрытии окна менеджера переменных пользователю предлагается сохранить изменения.

Для того, чтобы привязать имеющуюся ПЛК переменную к блоку входа/выхода, необходимо в контекстном меню блока выбрать «Привязать переменную», после чего выбрать необходимую переменную в менеджере переменных и нажать «Ок».

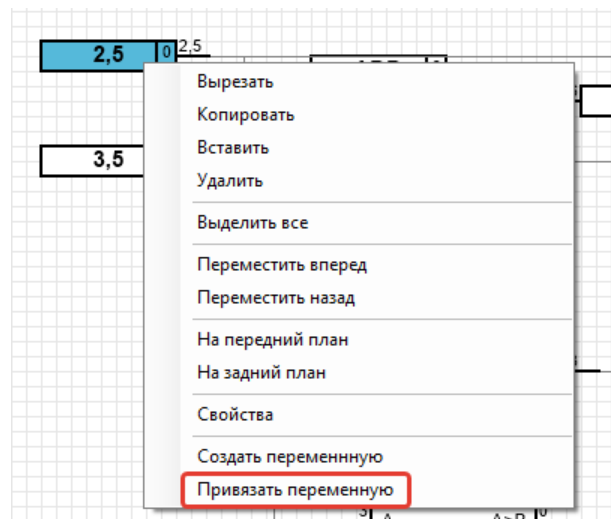


Рисунок 18 – Контекстное меню блока входа/выхода

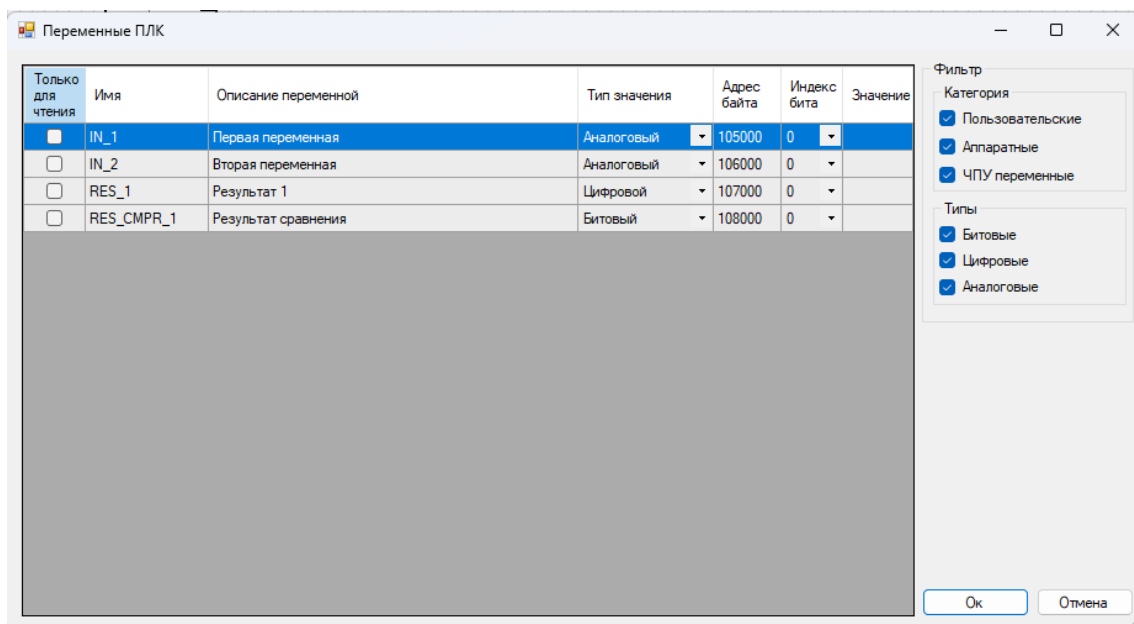


Рисунок 19 – Список переменных в режиме просмотра (выбора)

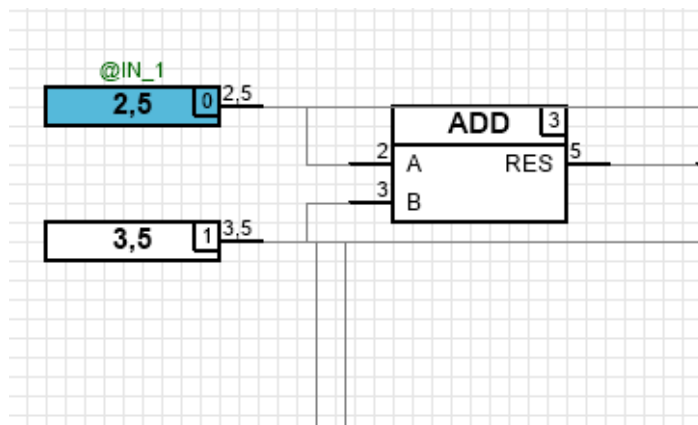


Рисунок 20 – Блок с привязанной ПЛК переменной

После успешной привязки ПЛК переменной к блоку его имя подсвечивается зеленым цветом, а название соответствует имени переменной с префиксом «@». Все остальные параметры блока входа/выхода автоматически настраиваются в соответствии с параметрами переменной.

1. Основные параметры	
Идентификатор объекта	0
Тип объекта	IN
Привязка к переменной	True
Имя объекта	@IN_1
Описание объекта	Первая переменная
2. Графические параметры	
X	50
Y	45
3. Параметры блока	
Статический адрес объекта	Да
Количество входов	0
Количество выходов	1
4. Параметры входа/выхода	
Тип хранимого значения	Analog
Длина в битах	32
Значение входа/выхода	2,5
Вид входа-выхода	CommonPlcMemory
Адрес (порта или памяти)	105000
Индекс бита	0
Разрешение изменять значен	False

Рисунок 21 – Параметры блока входа/выхода с привязанной переменной

При изменении параметров переменной в менеджере переменных автоматически изменяются параметры всех блоков, привязанных к ней.

В случае изменения пользователем параметров блока, привязанного к переменной, привязка к переменной автоматически сбрасывается, о чем выводится соответствующее предупреждение.

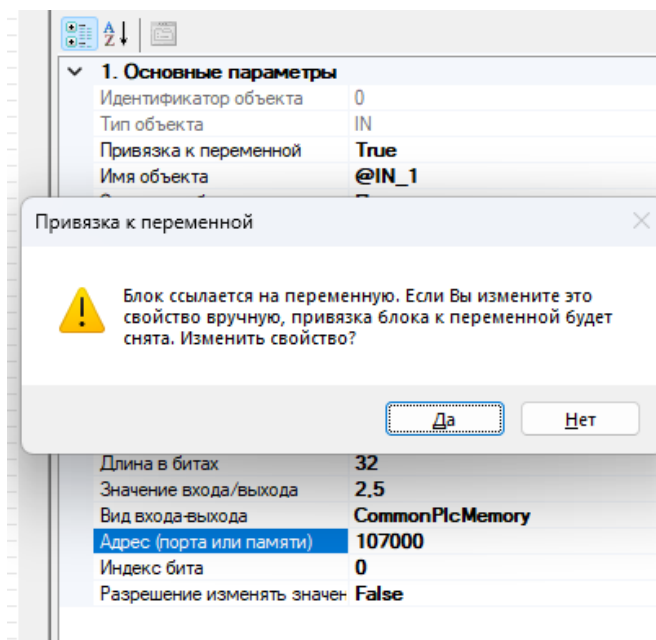


Рисунок 22 – Предупреждение о том, что привязка блока к переменной будет снята при ручном изменении параметров блока

Другим способом привязки блока к переменной является ручное задание его имени вида «@ + название переменной». В случае, если найдено название переменной с указанным именем, пользователю будет предложено привязать блок к данной переменной.

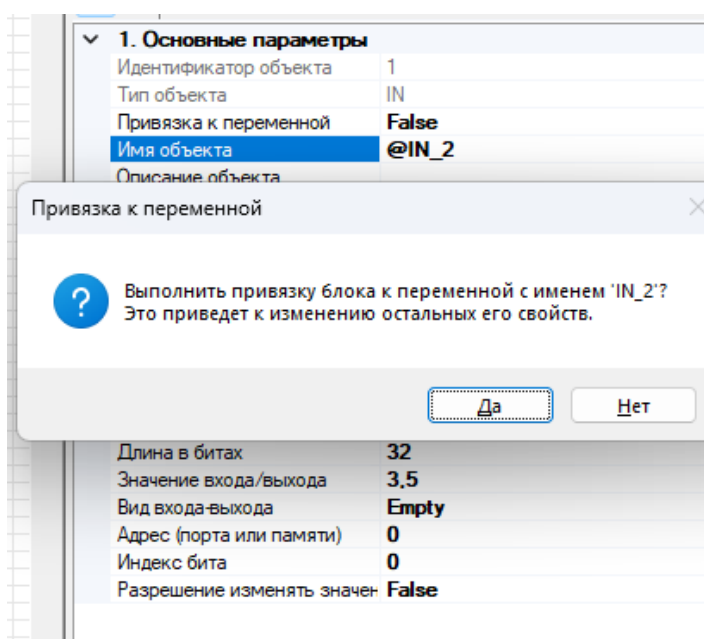


Рисунок 23 – Привязка блока к переменной путем изменения ее имени

Также в редакторе ПЛК программ имеется возможность создания ПЛК переменных на основе уже имеющихся блоков входа/выхода с заданными параметрами. Для этого в контекстном меню блока необходимо выбрать пункт «Создать переменную».

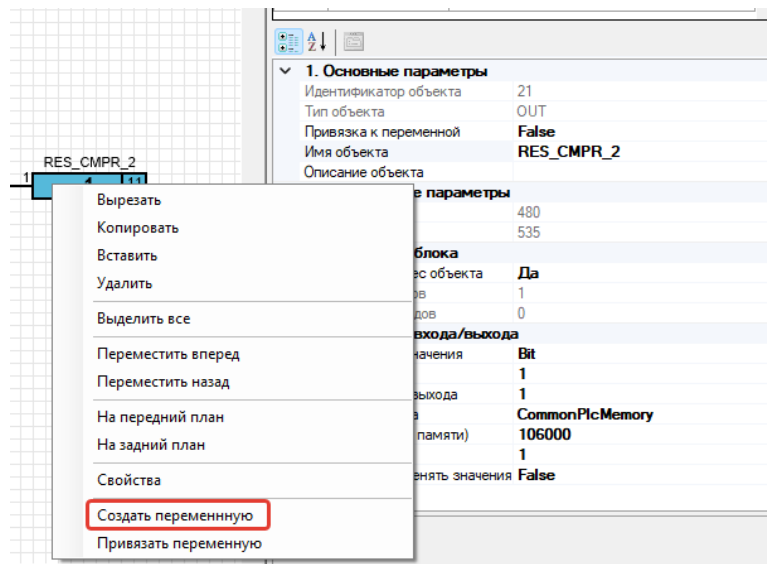


Рисунок 24 – Создание ПЛК переменной на основе имеющегося блока входа/выхода

После этого открывается окно, в котором необходимо задать основные параметры создаваемой переменной. В дальнейшем предполагается реализация функции выбора источника данных, например, выбор из заранее созданных переменных интерфейса ЧПУ–ПЛК.

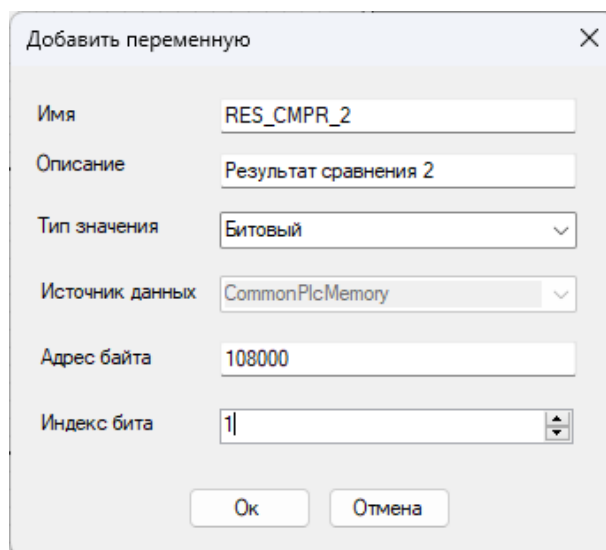


Рисунок 25 – Создание ПЛК переменной на основе имеющегося блока входа/выхода

После заполнения параметров в менеджере ПЛК переменных создается запись с данными переменной, а блок входа/выхода автоматически привязывается к нему.

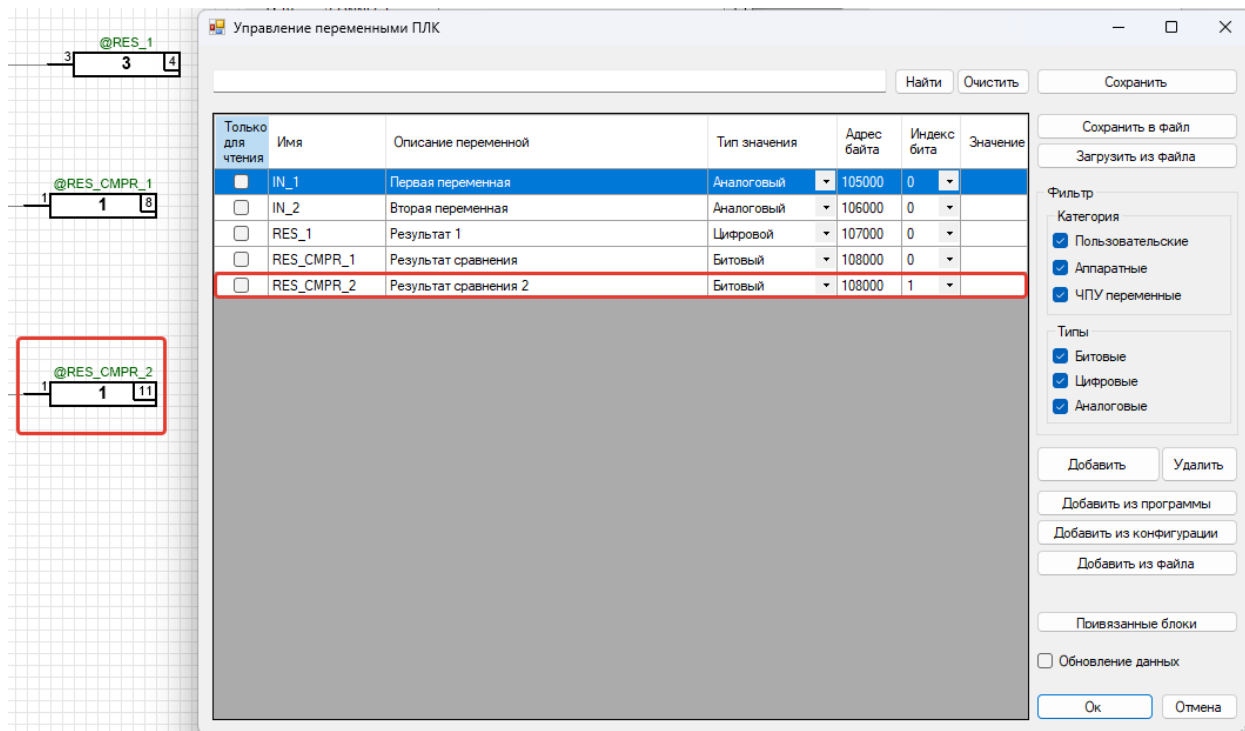


Рисунок 26 – ПЛК переменная «RES_CMPR_2» создана и привязана к блоку выхода

Функциональность в разработке: Менеджер ПЛК переменных позволяет выполнять отладку программ путем отображения в таблице текущих значений всех заданных переменных, для чего необходимо при запущенной ПЛК программе в менеджере переменных выбрать флаг «Обновление данных». Редактирование переменных при этом будет запрещено, и в таблице будет отображаться текущее значение каждой переменной.

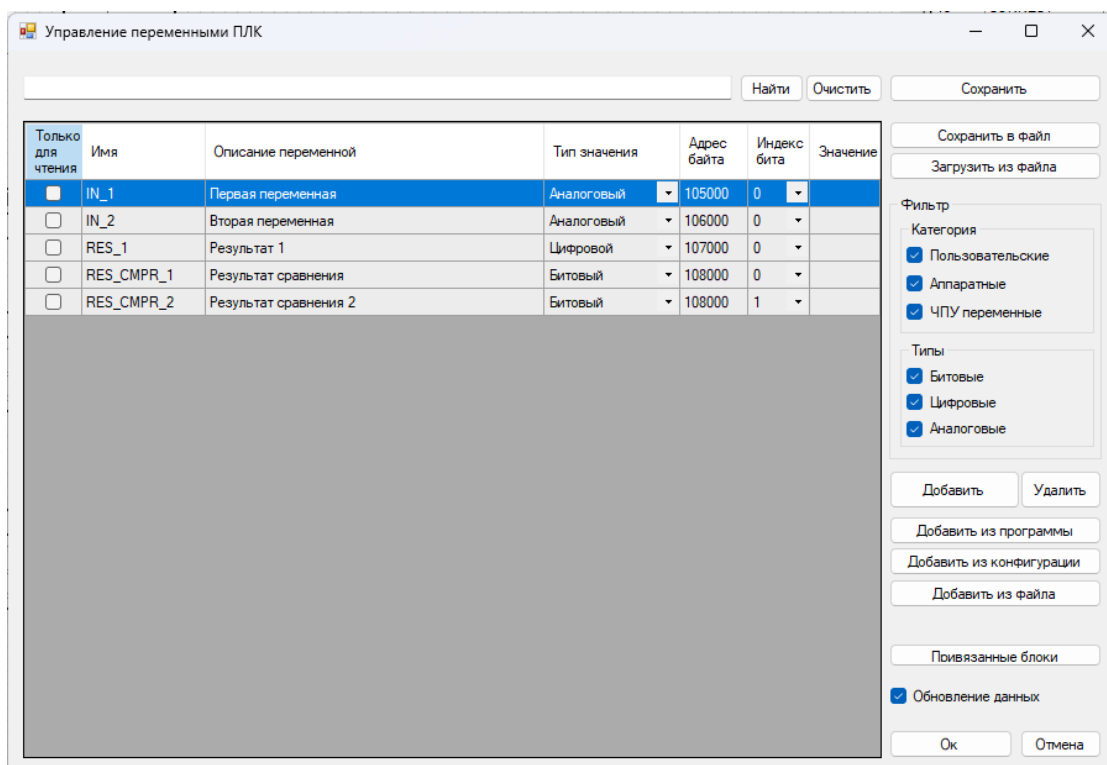


Рисунок 27 – Режим просмотра (отладки) текущих значений ПЛК переменных

3.6 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.

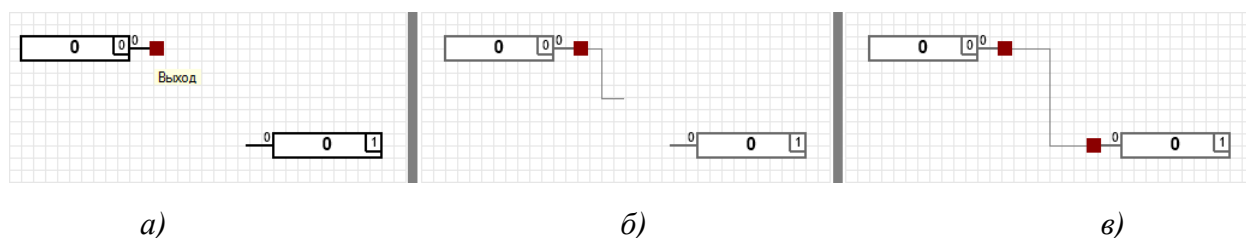


Рисунок 28 – Этапы создания связи между двумя блоками

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.

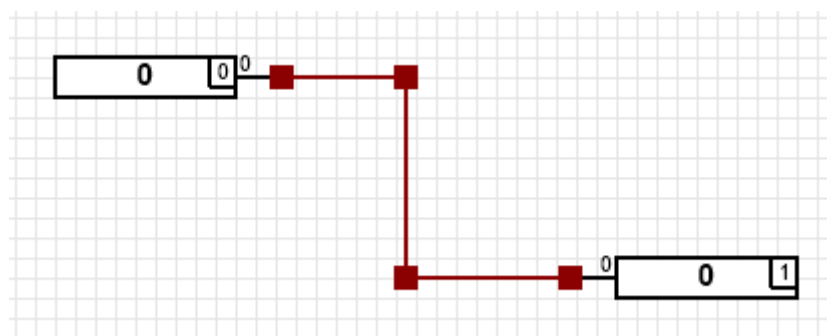


Рисунок 29 – Выделенная связь между блоками

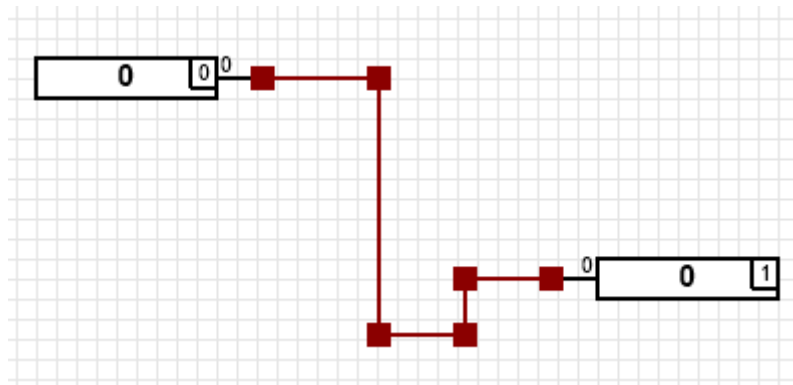


Рисунок 30 – Создание дополнительного излома у связи

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.

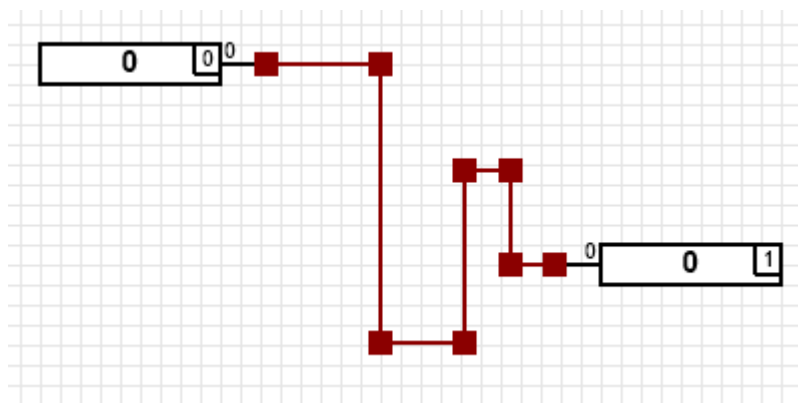


Рисунок 31 – Создание дополнительного излома у связи

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.

3.7 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 32).

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 32 представлен пример библиотеки для выполнения арифметических 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.

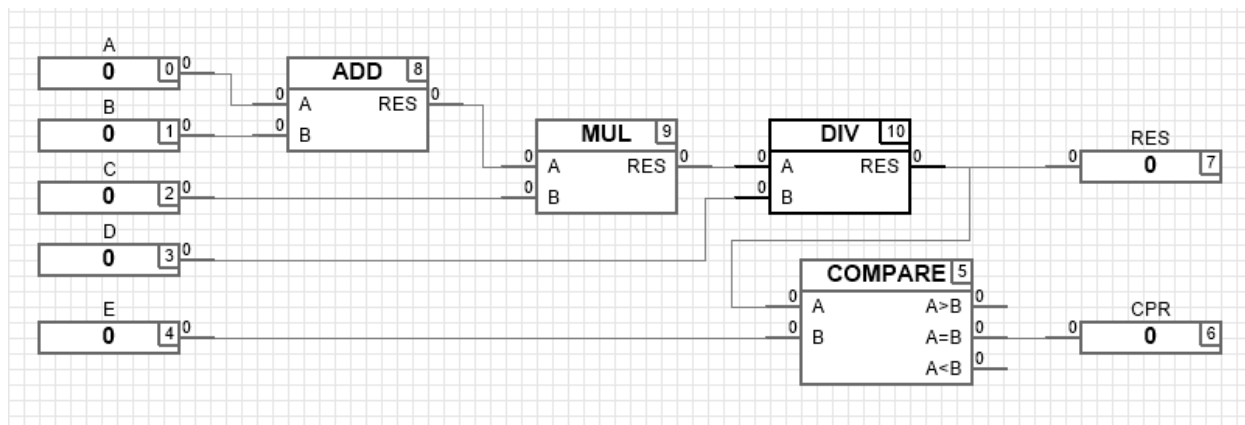


Рисунок 32 – Создание пользовательской библиотеки

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 33.

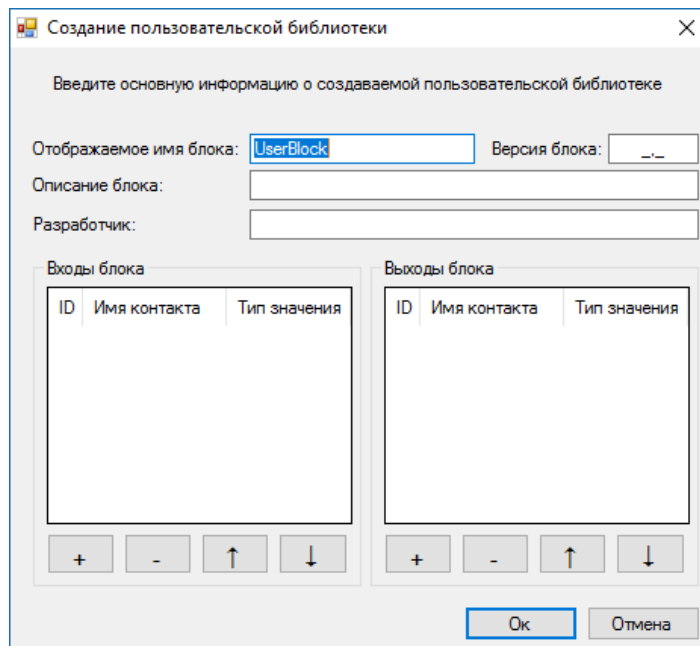


Рисунок 33 – Диалоговое окно привязки входов/выходов пользовательской библиотеки

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 «А», и нажать «Ок» (Рисунок 34).

Идентификатор объекта	Приоритет работы	Имя объекта	Описание объекта
0	0	A	
1	1	B	
2	2	C	
3	3	D	
4	4	E	

Рисунок 34 – 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.

Создание пользовательской библиотеки

Введите основную информацию о создаваемой пользовательской библиотеке

Отображаемое имя блока: Версия блока:

Описание блока:

Разработчик:

Входы блока

ID	Имя контакта	Тип значения
0	A	Digital
1	B	Digital
2	C	Digital
3	D	Digital
4	E	Digital

Выходы блока

ID	Имя контакта	Тип значения
7	RES	Digital
6	CPR	Bit

Buttons: +, -, ↑, ↓ (for both tables)

Buttons:

Рисунок 35 – 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.

3.8 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 36).

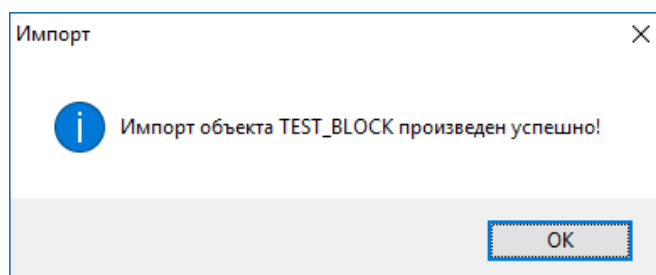


Рисунок 36 – Сообщение об успешном импорте пользовательской библиотеки

Также в меню функциональных блоков должен появиться пункт, соответствующий названию нового пользовательского блока (Рисунок 37).

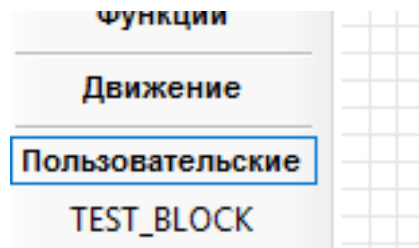


Рисунок 37 – Новый функциональный блок в меню объектов после импорта библиотеки

После этого данный пользовательский блок является полноценным функциональным объектом, который можно многократно использовать в любых программах. На Рисунок 38 показана программа с использованием созданного тестового пользовательского блока.

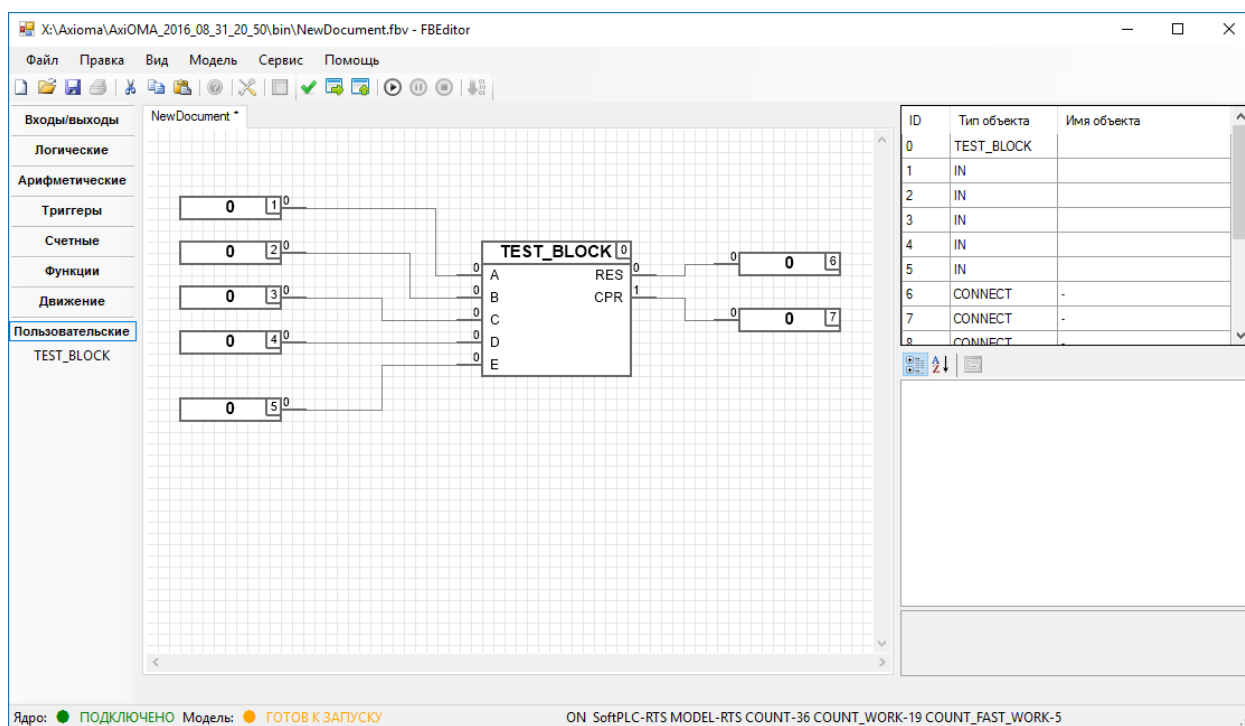


Рисунок 38 – Проект управляющей программы с использованием пользовательского блока TEST_BLOCK

На Рисунок 39 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 Е 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.

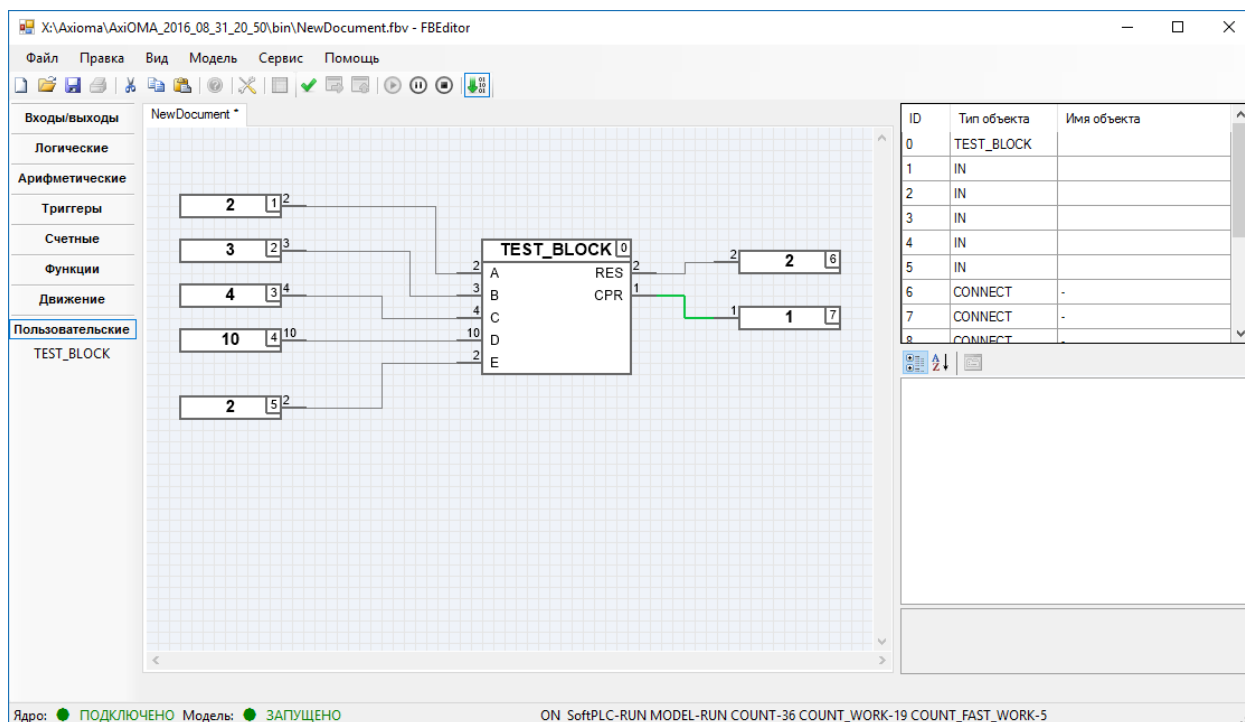


Рисунок 39 – Работа пользовательского блока в управляющей программе

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 40).

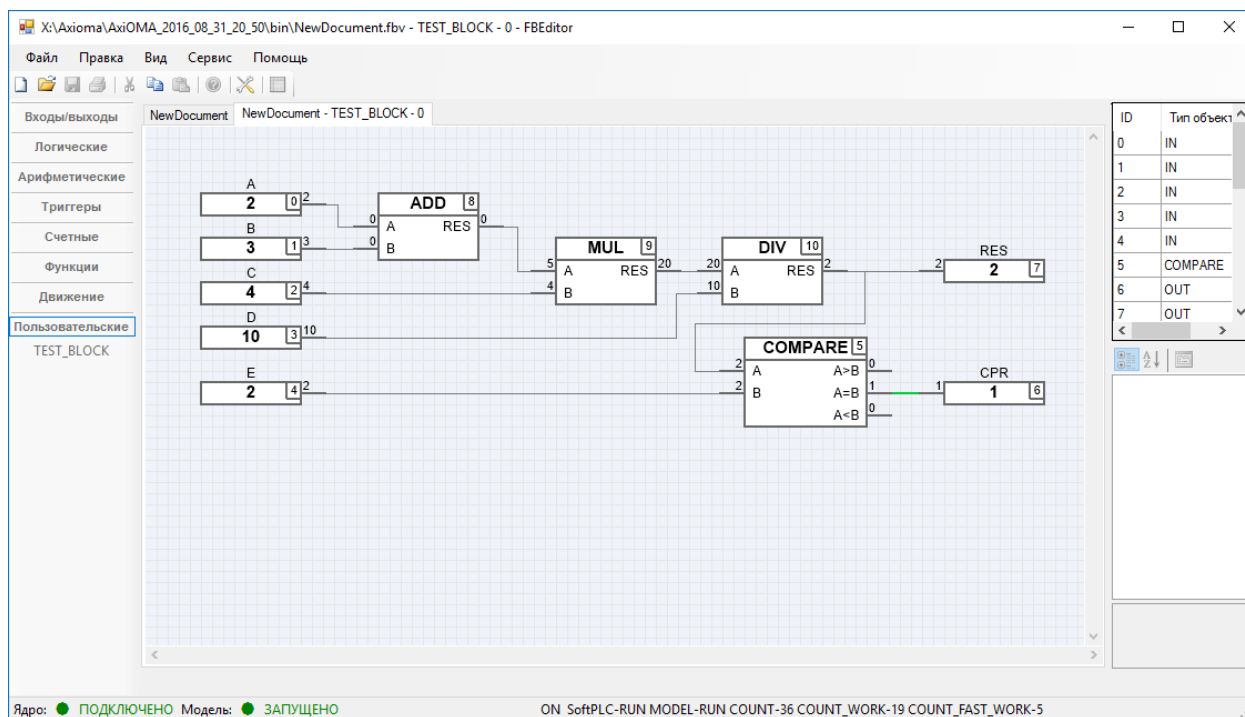


Рисунок 40 – Раскрытый пользовательский блок TEST_BLOCK в режиме отладки

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.

3.9 Замена пользовательских блоков

Чтобы заменить имеющийся пользовательский блок, необходимо в его контекстном меню в панели функциональных блоков выбрать «Заменить библиотеку» (Рисунок 41).

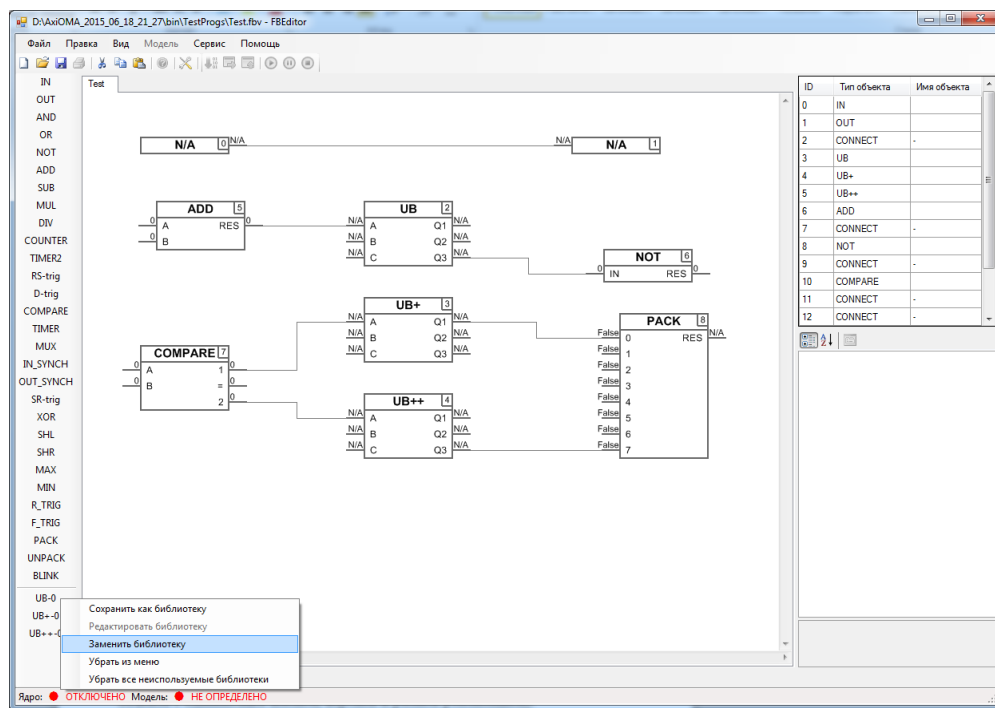


Рисунок 41 – Функция «Заменить библиотеку»

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.

3.10 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 42).

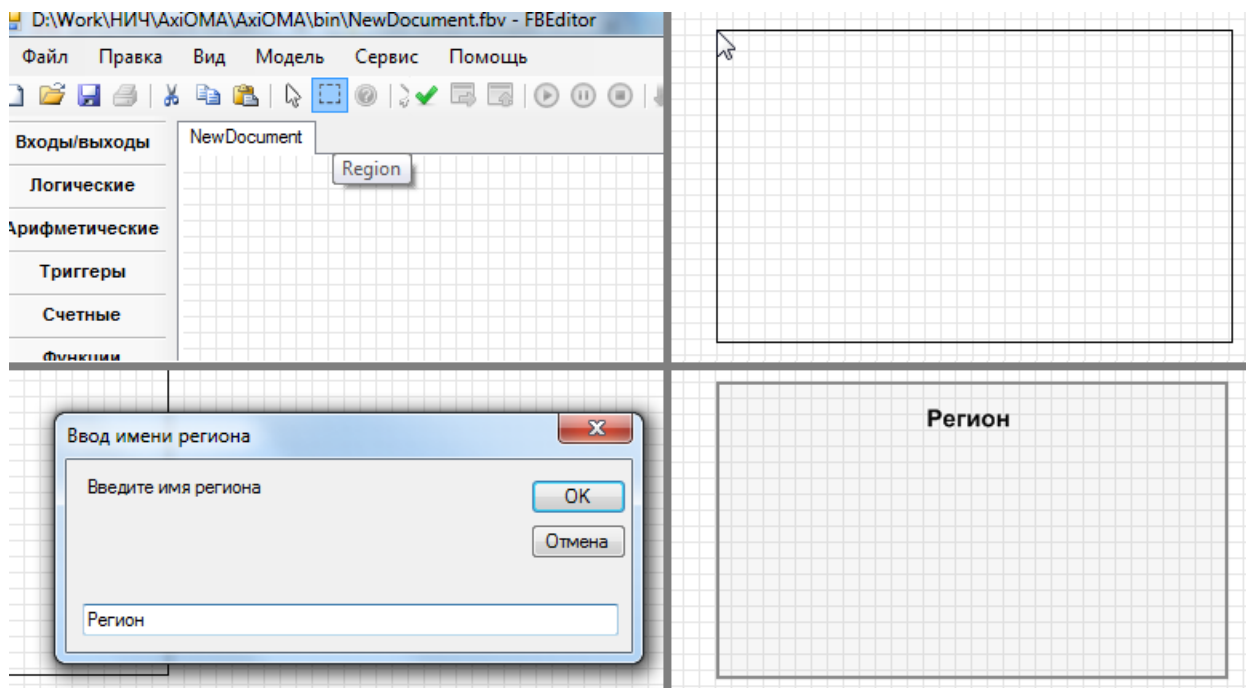


Рисунок 42 – Этапы создания региона

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 43).

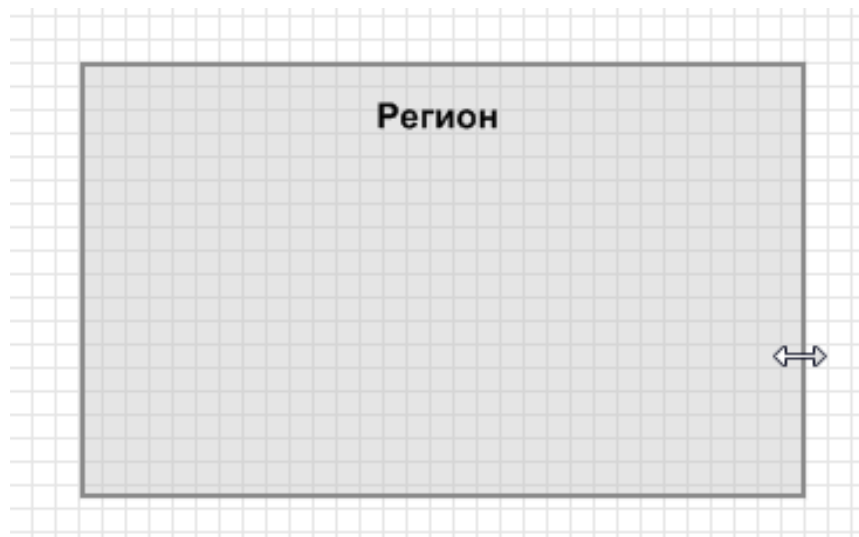


Рисунок 43 – Изменение размеров региона

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 44). Чтобы удалить регион, необходимо выделить его и нажать клавишу «Del» на клавиатуре.

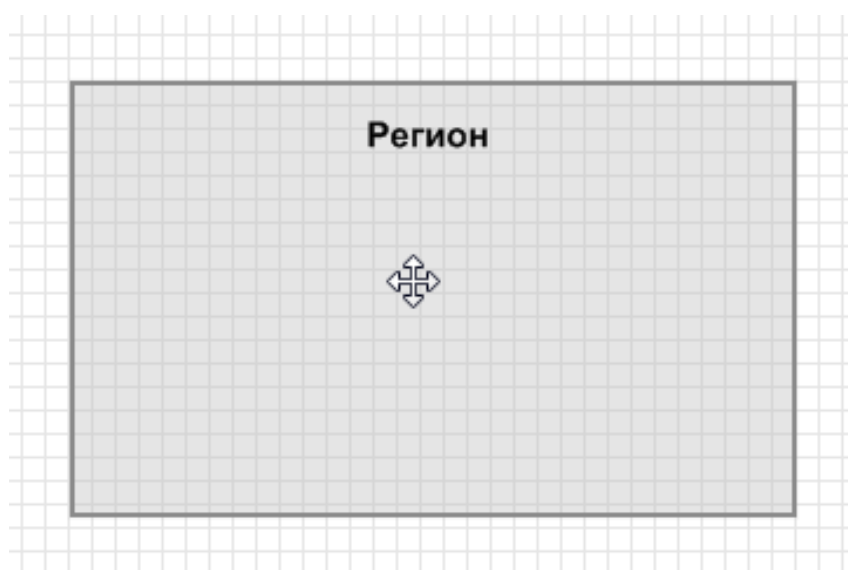


Рисунок 44 – Перемещение региона

При наведении указателя мыши на выделенный регион появляется всплывающая подсказка (Рисунок 45), 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.

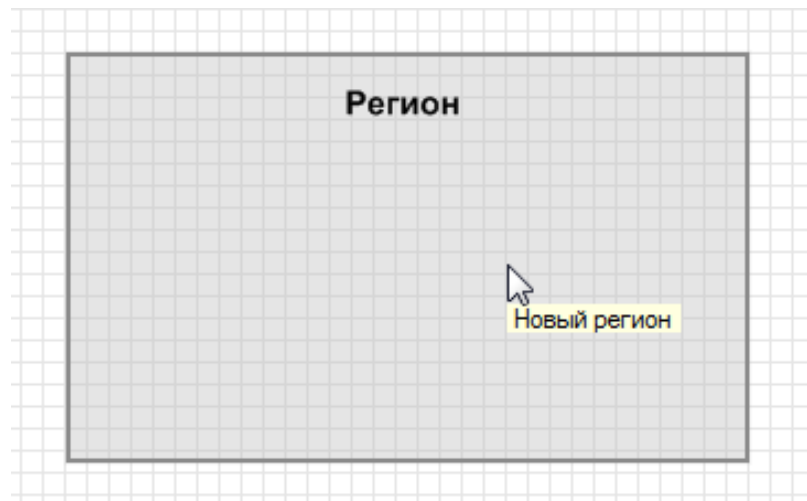


Рисунок 45 – Всплывающая подсказка с описанием региона

Пример использования регионов для визуального объединения группы элементов и представления оператору функциональности, выполняющейся в данной области программы, представлен на Рисунок 46.

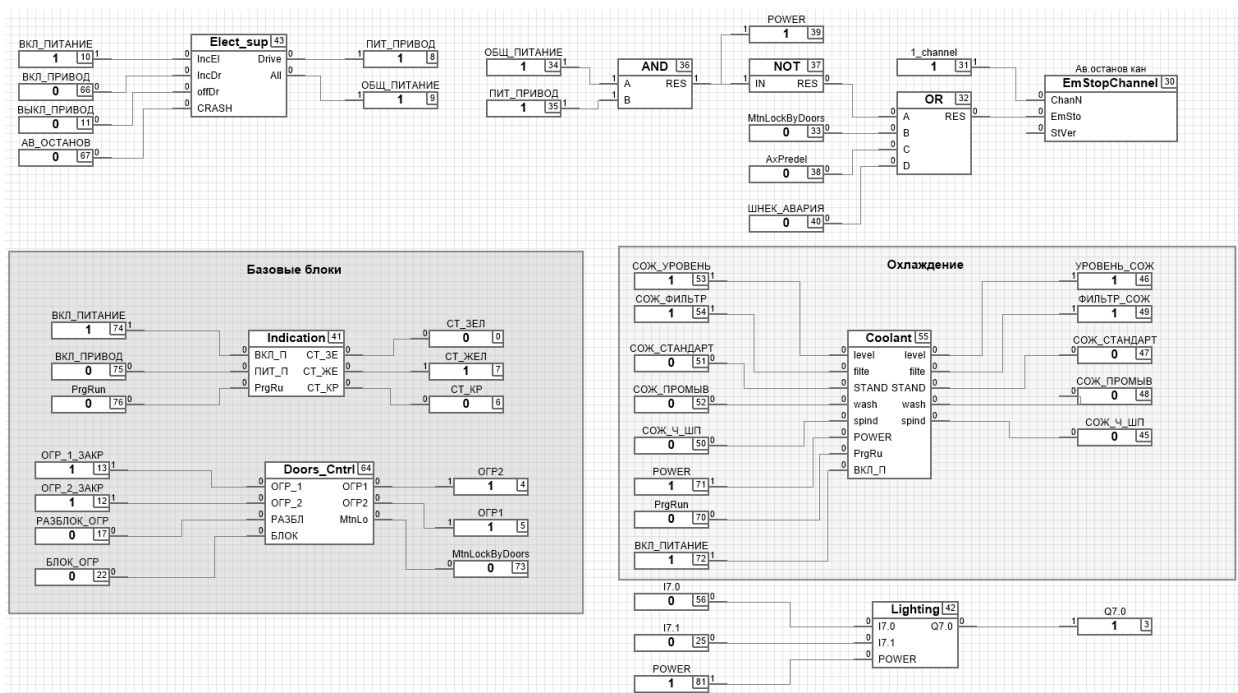


Рисунок 46 – Пример использования регионов

3.11 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 47).

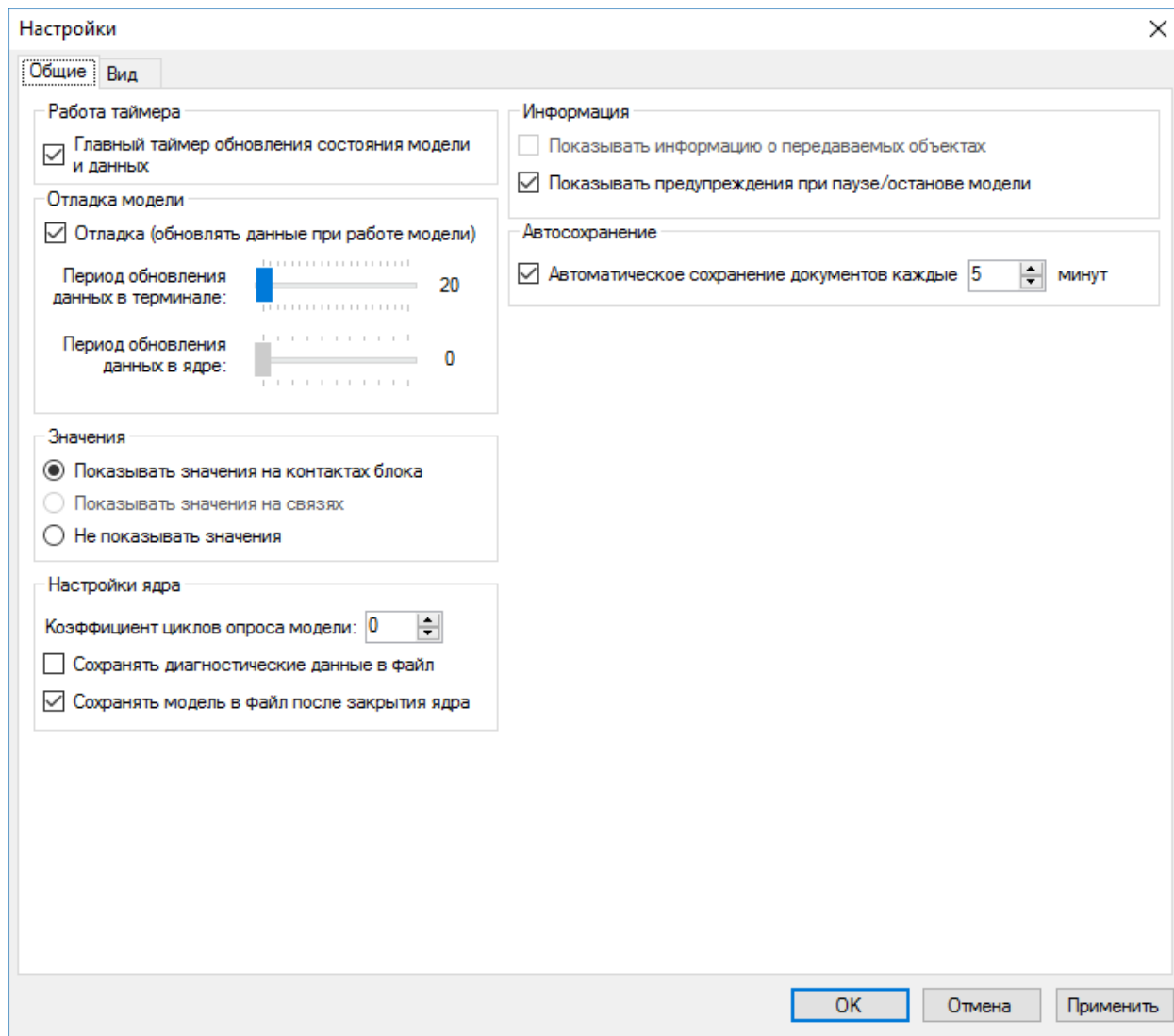


Рисунок 47 – Окно настроек редактора, вкладка общих настроек

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 48).

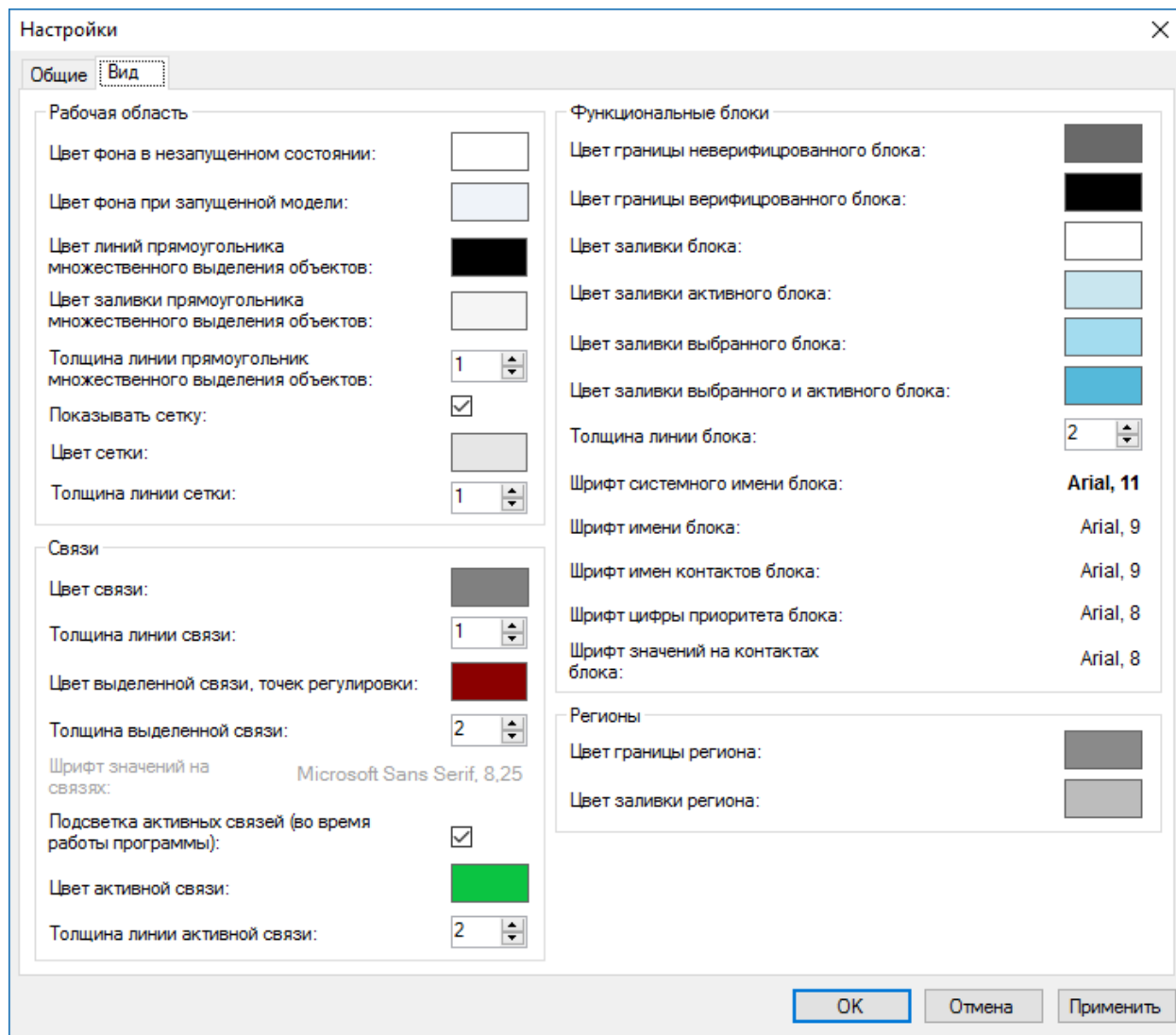


Рисунок 48 – Окно настроек редактора, вкладка настроек интерфейса

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».

3.12 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 49).

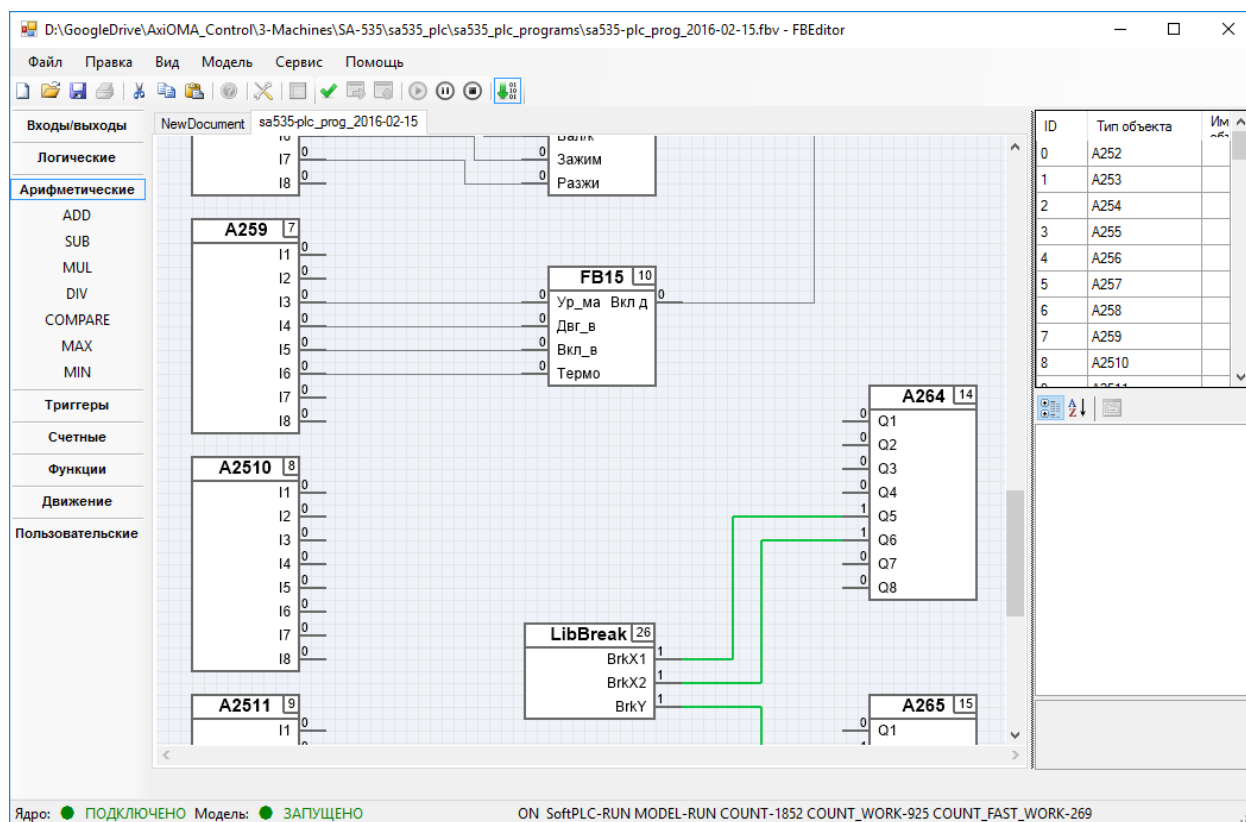


Рисунок 49 – Общий вид работающей программы в FB-редакторе

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.

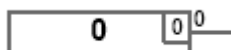
Для остановки работы программы следует нажать на кнопку «Стоп» меню управления работой модели или нажать комбинацию горячих клавиш «Shift+F5». После этого можно вносить изменения в программу. Чтобы снова запустить программу, требуется заново выполнить действия по отправке программы в ядро и её запуску.

4 Описание функциональных блоков

4.1 Блоки входов и выходов

4.1.1 Вход

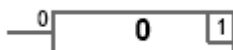
Блок позволяет задать ссылку на переменную из области памяти (CommonPlcMemory) или иметь константное значение, для его дальнейшего использования в программе.



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

4.1.2 Выход

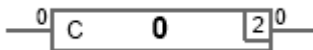
Блок позволяет записать значение в переменную, ссылающуюся на область памяти (CommonPlcMemory), или переменную-константу.



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

4.1.3 Вход с синхронизацией

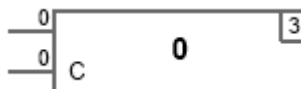
Блок позволяет задать ссылку на переменную из области памяти (CommonPlcMemory) или иметь константное значение, для его дальнейшего использования в программе. При этом значение передается только в том случае, если на вход “С” подана логическая единица.



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

4.1.4 Выход с синхронизацией

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

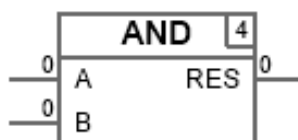


Вход/выход	Имя	Тип данных	Описание
Вход	-	Bit/Digital/Analog	Переменная, в которую записывается значение (выход)
Вход	C	Bit	Флаг работы объекта выхода

4.2 Логические блоки

4.2.1 AND: дизъюнкция (И, логическое умножение)

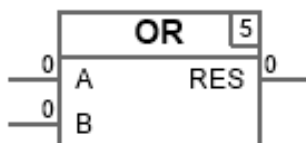
Блок реализует операцию логического умножения. Входы A и B принимают значения логических переменных, выход RES осуществляет вывод результата операции логического умножения. Имеется возможность увеличения входов до 16 штук.



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)

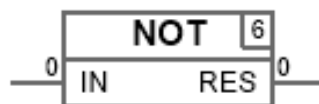
Блок реализует операцию логического сложения. Входы A и B принимают значения логических переменных, выход RES осуществляет вывод результата операции логического сложения. Имеется возможность увеличения входов до 16 штук.



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.

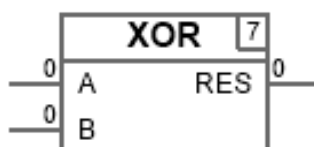


Вход/выход	Имя	Тип данных	Описание
Вход	IN	Bit	Входное логическое значение
Выход	RES	Bit	Результат логического отрицания

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.

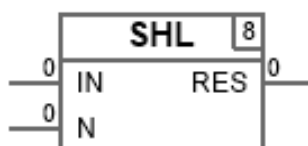
Возможно увеличение входов до 16. В этом случае входы обрабатываются попарно, затем к результатам опять применяется XOR. Такой алгоритм определён стандартом.



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

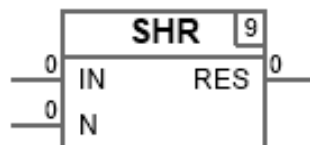
Если N превышает ширину типа данных, то операнды типов BYTE, WORD и DWORD заполняются нулями, операнды типов со знаками, таких как INT, получают арифметический сдвиг, т.е. заполняются значениями самого верхнего бита.



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

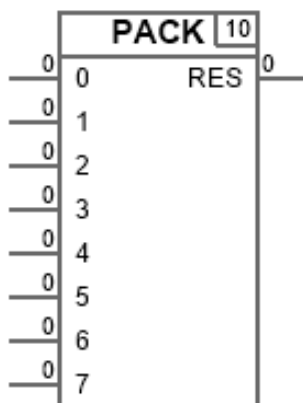
Если N превышает ширину типа данных, то операнды типов BYTE, WORD и DWORD заполняются нулями, операнды типов со знаками, таких как INT, получают арифметический сдвиг, т.е. заполняются значениями самого верхнего бита.



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

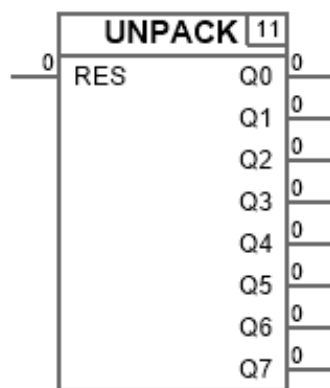
Блок преобразует восемь входных параметров типа BIT в один байт (DIGITAL).



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.



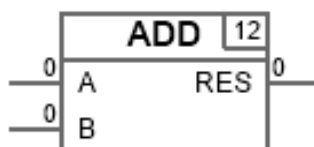
Вход/выход	Имя	Тип данных	Описание
Вход	RES	Digital	Входной байт
Выход	Q0	Bit	Выходное битовое значение
Выход	Q1	Bit	Выходное битовое значение
Выход	Q2	Bit	Выходное битовое значение
Выход	Q3	Bit	Выходное битовое значение
Выход	Q4	Bit	Выходное битовое значение
Выход	Q5	Bit	Выходное битовое значение
Выход	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 А и В поступают слагаемые, на выход RES осуществляется вывод результирующей суммы.

Имеется возможность увеличения входов до 16 штук.

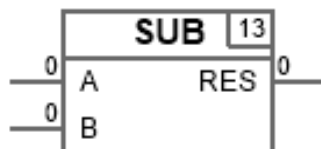


Вход/выход	Имя	Тип данных	Описание
Вход	A	Analog	Первое слагаемое
Вход	B	Analog	Второе слагаемое
Выход	RES	Analog	Сумма

4.3.2 SUB: вычитание

Блок реализует арифметическую операцию вычитания. На входы А поступает уменьшаемое, на вход В – вычитаемое, на выходе RES осуществляется вывод результата.

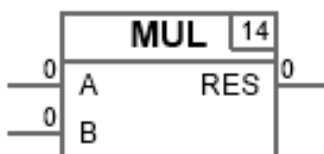
Имеется возможность увеличения входов до 16 штук.



Вход/выход	Имя	Тип данных	Описание
Вход	A	Analog	Уменьшаемое
Вход	B	Analog	Вычитаемое
Выход	RES	Analog	Результат

4.3.3 MUL: умножение

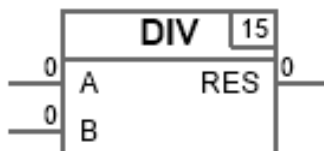
Блок реализует арифметическую операцию умножения. На входы A и B поступают множители, на выход RES осуществляется вывод результата операции. Имеется возможность увеличения входов до 16 штук.



Вход/выход	Имя	Тип данных	Описание
Вход	A	Analog	Множитель
Вход	B	Analog	Множитель
Выход	RES	Analog	Результат

4.3.4 DIV: деление

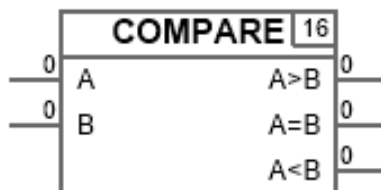
Блок реализует арифметическую операцию деления. На вход A поступает делимое, на вход B – делитель, на выход RES осуществляется вывод результата операции. Имеется возможность увеличения входов до 16 штук.



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

4.3.5 COMPARE: сравнение

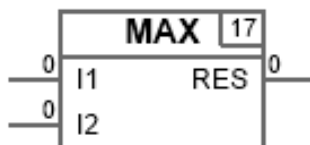
Блок реализует сравнение двух чисел. На входы подаются два значения, которые необходимо сравнить между собой. Выходы отвечают за результат сравнения.



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

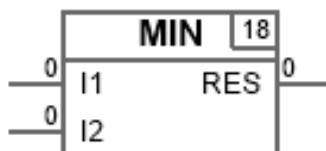
Возвращает наибольшее из входных значений. Имеется возможность увеличения входов до 16 штук.



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.

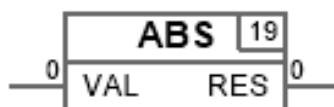


Вход/выход	Имя	Тип данных	Описание
Вход	I1	Analog	Значение выборки
Вход	I2	Analog	Значение выборки
Выход	RES	Analog	Результат (минимальное число)

			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.



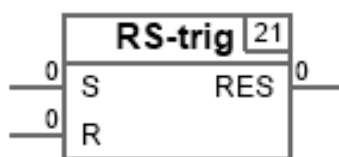
Вход/выход	Имя	Тип данных	Описание
Вход	VAL	Analog	Входное значение
Выход	RES	Digital	Преобразованное (округленное) значение

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.

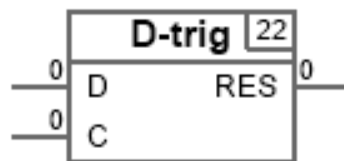


Вход/выход	Имя	Тип данных	Описание
------------	-----	------------	----------

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».

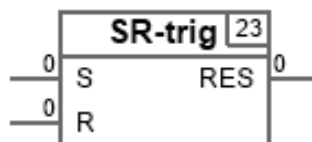


Вход/выход	Имя	Тип данных	Описание
Вход	D	Bit	Информационный вход
Вход	C	Bit	Вход задержки
Выход	RES	Bit	Результат

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».

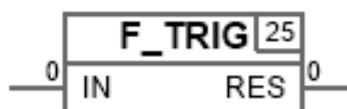


Вход/выход	Имя	Тип данных	Описание
Вход	IN	Bit	Управляющий вход
Выход	RES	Bit	Результат

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».

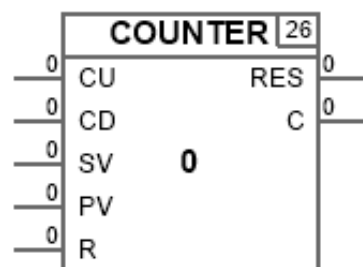


Вход/выход	Имя	Тип данных	Описание
Вход	IN	Bit	Управляющий вход
Выход	RES	Bit	Результат

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.

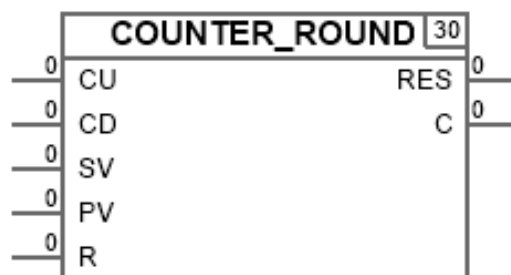


Вход/выход	Имя	Тип данных	Описание
Вход	CU	Bit	Увеличивает счётное значение на единицу
Вход	CD	Bit	Уменьшает счётное значение на единицу
Вход	SV	Digital	Стартовое (минимальное) значение счетчика

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

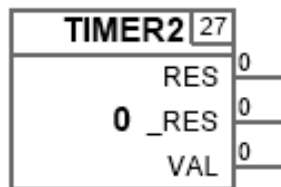
Блок реализует счетные операции, но при переходе через максимальное значение он продолжает счет с нуля. При переходе через ноль в отрицательную сторону устанавливает максимальное значение и продолжает уменьшаться.



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

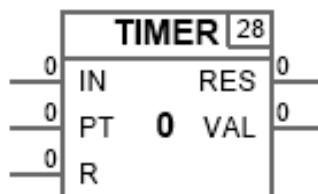
Блок выдает попеременно логические сигналы «0» и «1» на выходы в соответствии с заданным в параметрах блока интервалом обновления (интервал задается в миллисекундах).



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.

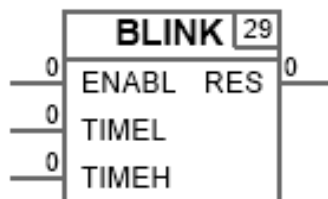


Вход/выход	Имя	Тип данных	Описание
Вход	IN	Bit	Входное логическое значение, запускающее работу таймера
Вход	PT	Digital	Значение интервала задержки таймера
Вход	R	Bit	Вход осуществляет сброс таймера в начальное значение
Выход	RES	Bit	Выходной логический сигнал
Выход	VAL	Digital	Актуальное значение таймера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

4.5.5 BLINK: генератор прямоугольных импульсов

Генератор запускается по входу ENABLE = «1». Длительность импульса задается входом TIMEN, 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
------------	-----	------------	----------

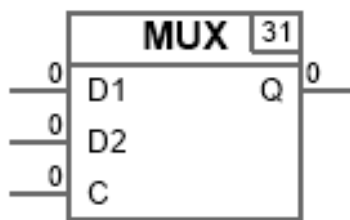
Вход	ENABL	Bit	Запуск генератора
Вход	TIMEL	Digital	Длительность паузы
Вход	TIMEN	Digital	Длительность импульса
Выход	RES	Bit	Выходной логический сигнал

4.6 Функции

4.6.1 MUX: мультиплексор

Позволяет передавать сигнал с одного из входов на выход, при этом выбор желаемого входа зависит от состояния управляющего сигнала C:

- при C=0: выход Q=D1,
- при C=1: выход Q=D2.

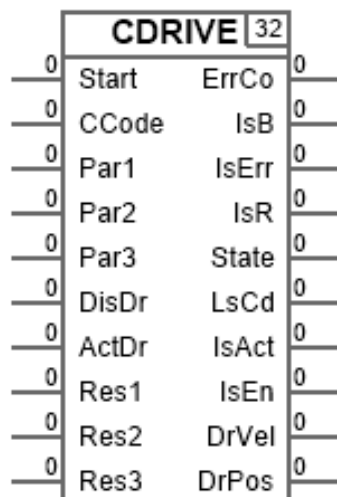


Вход/выход	Имя	Тип данных	Описание
Вход	D1	Analog	Первое цифровое значение
Вход	D2	Analog	Второе цифровое значение
Вход	C	Bit	Логическое значение, определяющее, какое из двух чисел будет на выходе
Выход	C	Analog	Результат

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 50).

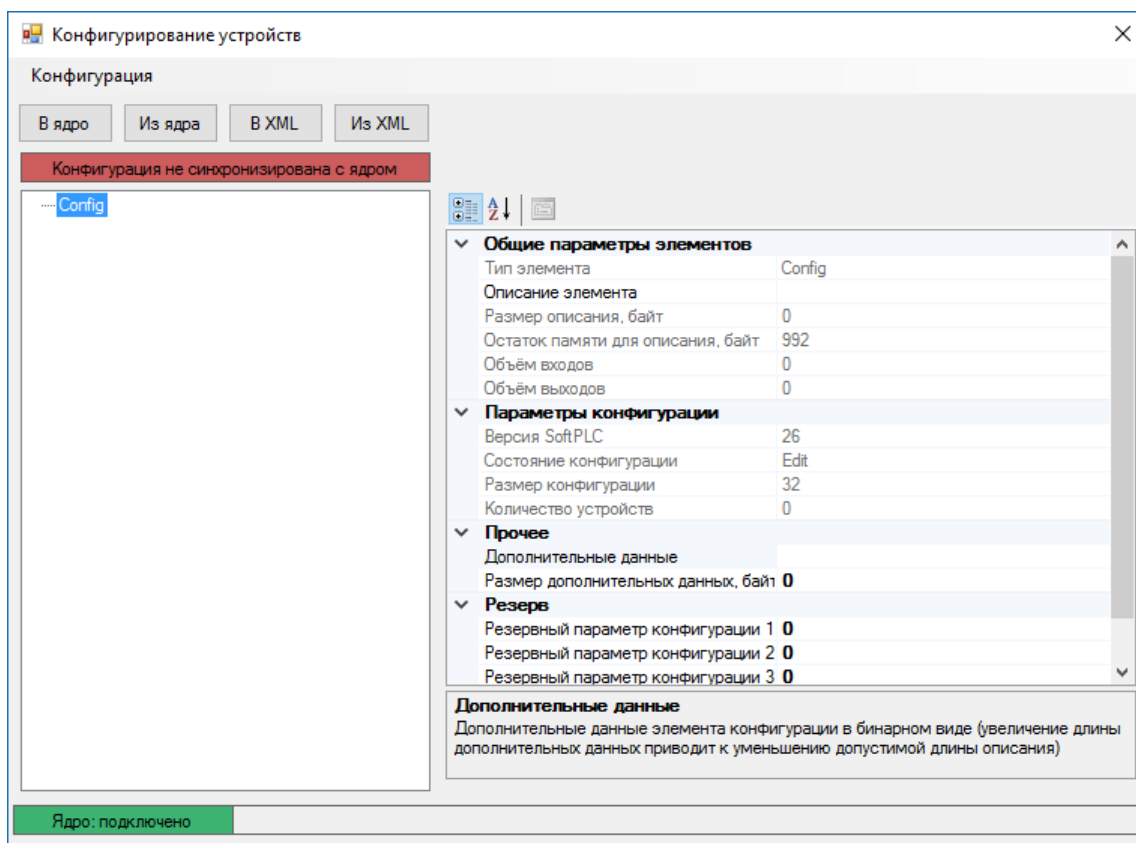


Рисунок 50 – Базовый интерфейс конфигуратора устройств ввода/вывода

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 51).

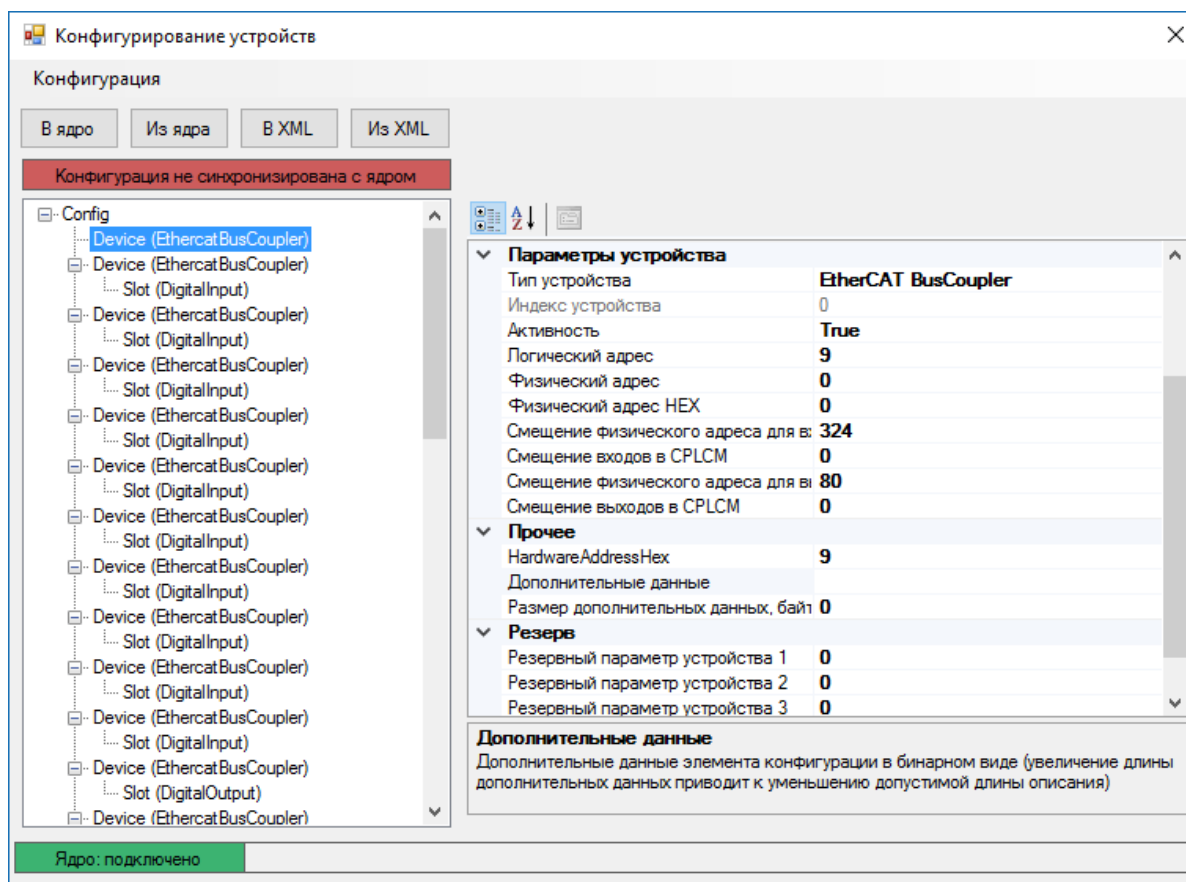


Рисунок 51 – Пример созданной конфигурации устройств ввода/вывода

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 52).

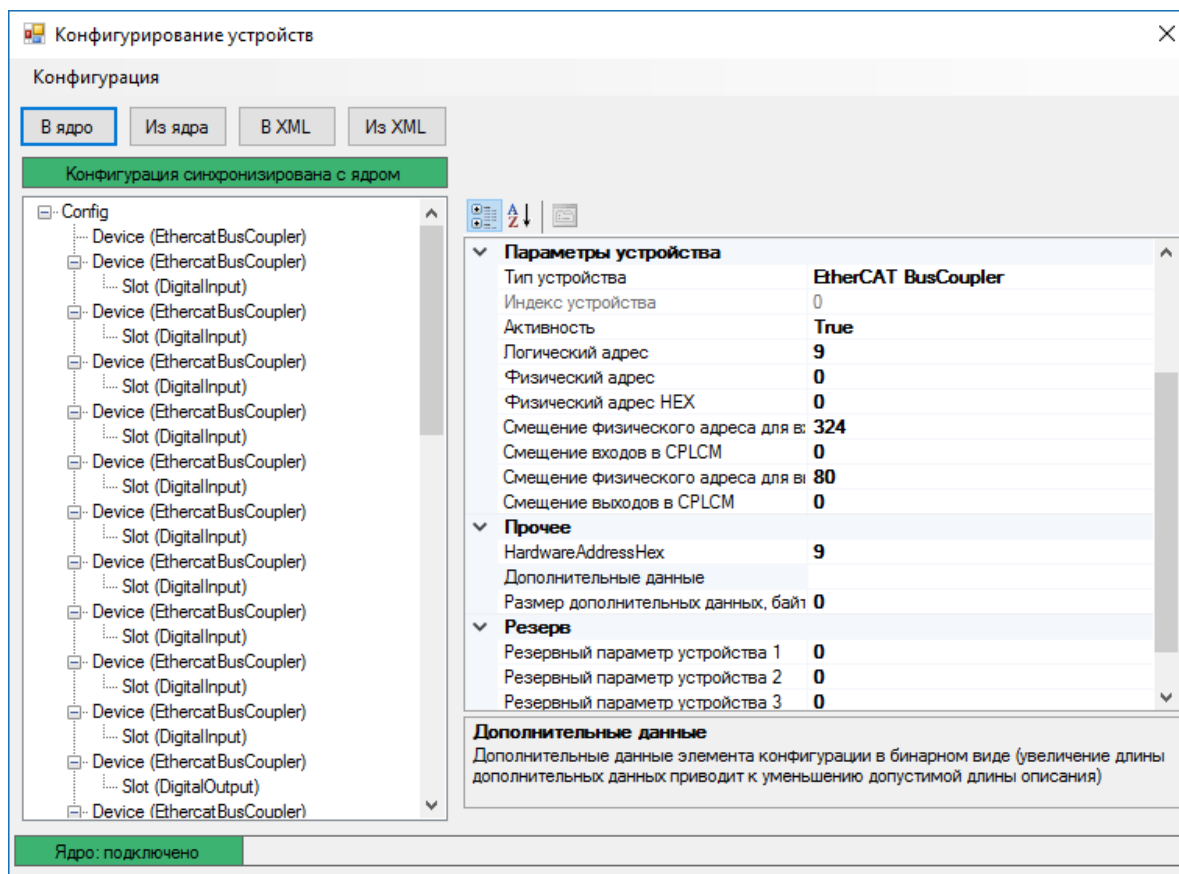


Рисунок 52 – Синхронизированная с ядром контроллера конфигурация устройств ввода/вывода

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 ODOT (EtherCAT)

Тестовый стенд (Рисунок 53) состоял из интерфейсного модуля ODOT CN-8033, 2-х модулей дискретных выходов CT-222F (каждый из которых имеет 16 выходов) и модуля дискретных входов CT-121F (16 входов).

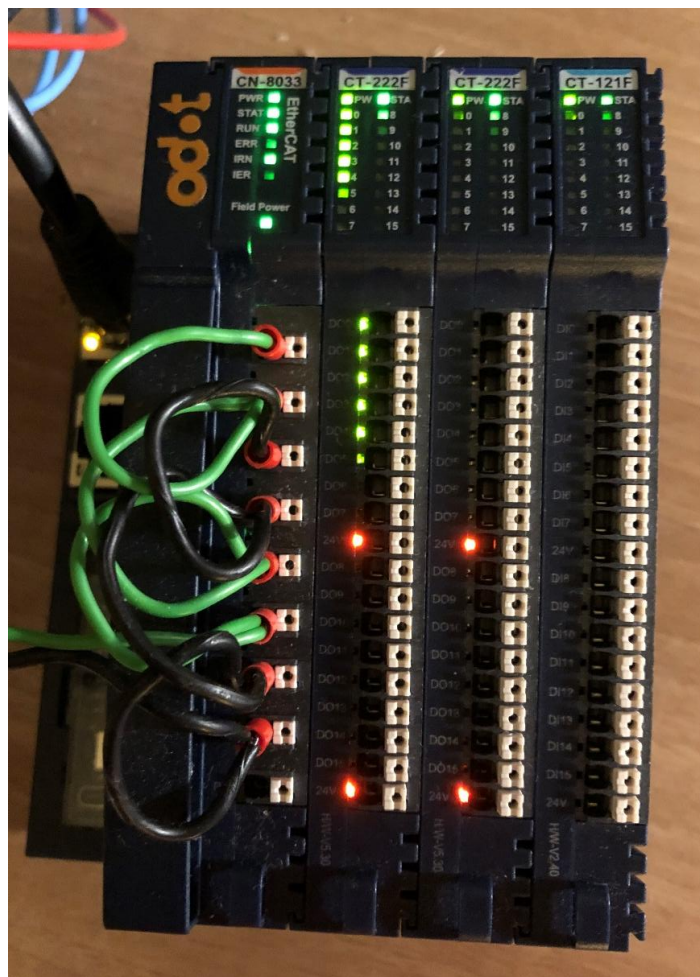


Рисунок 53 – Тестовый стенд ODOT

Интерфейсный модуль ODOT CN-8033 (Рисунок 53) предназначен для организации взаимодействия между мастером сети и модулями ввода-вывода.

Интерфейсный модуль ODOT CN-8033 вместе со всеми модулями ввода/вывода инициализируется в EtherCAT сети как одно устройство (один адрес, независимо от количества подключенных модулей).

Затем идут данные модулей. Смещения данных показаны на Рисунок 54:

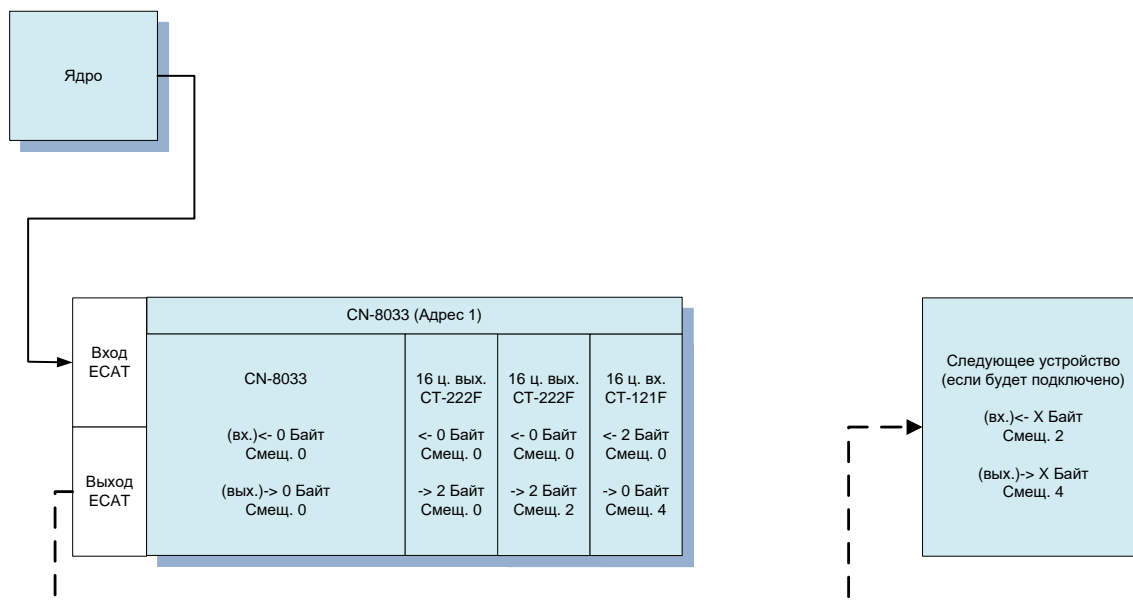


Рисунок 54 – Распределение данных модулей ввода/вывода ODOT в памяти ядра

Таким образом, для конфигурирования устройств, входящих в состав тестового стенда, необходимо:

- создать в конфигураторе устройство, соответствующее интерфейсному модулю ODOT CN-8033 (Рисунок 55) и задать необходимые параметры:

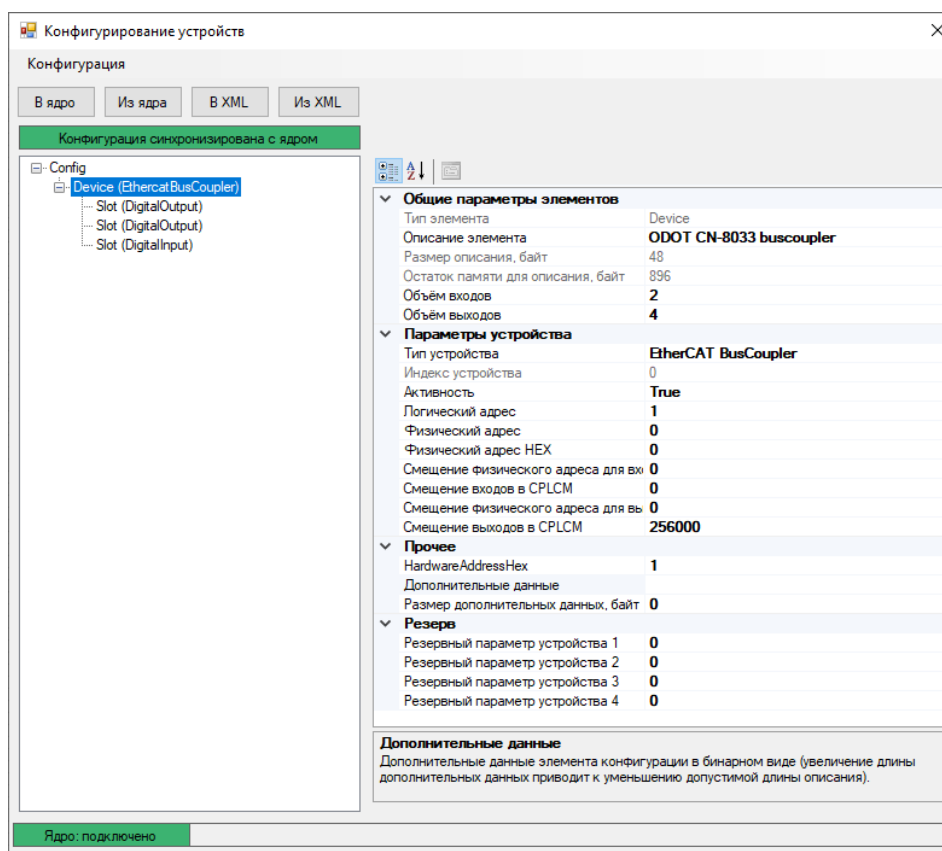


Рисунок 55 – Конфигурация устройства для интерфейсного модуля ODOT CN-8033

- создать и сконфигурировать два дочерних слота, соответствующих модулям дискретных выходов СТ-222F (Рисунок 56 и Рисунок 57), и слот, соответствующий модулю дискретных входов СТ-121F (Рисунок 58):

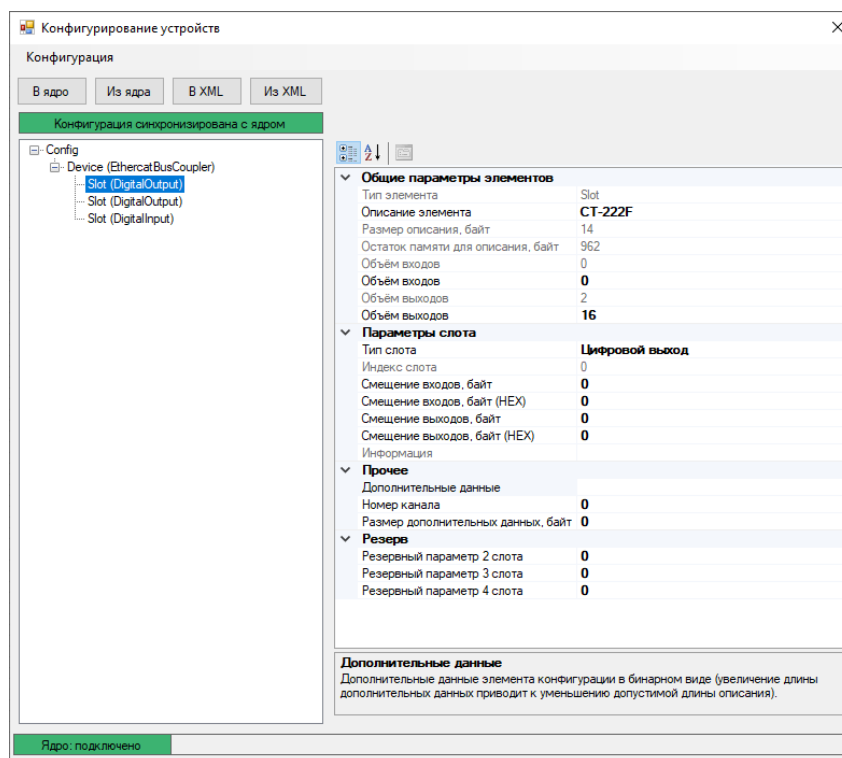


Рисунок 56 – Конфигурация первого модуля дискретных выходов СТ-222F

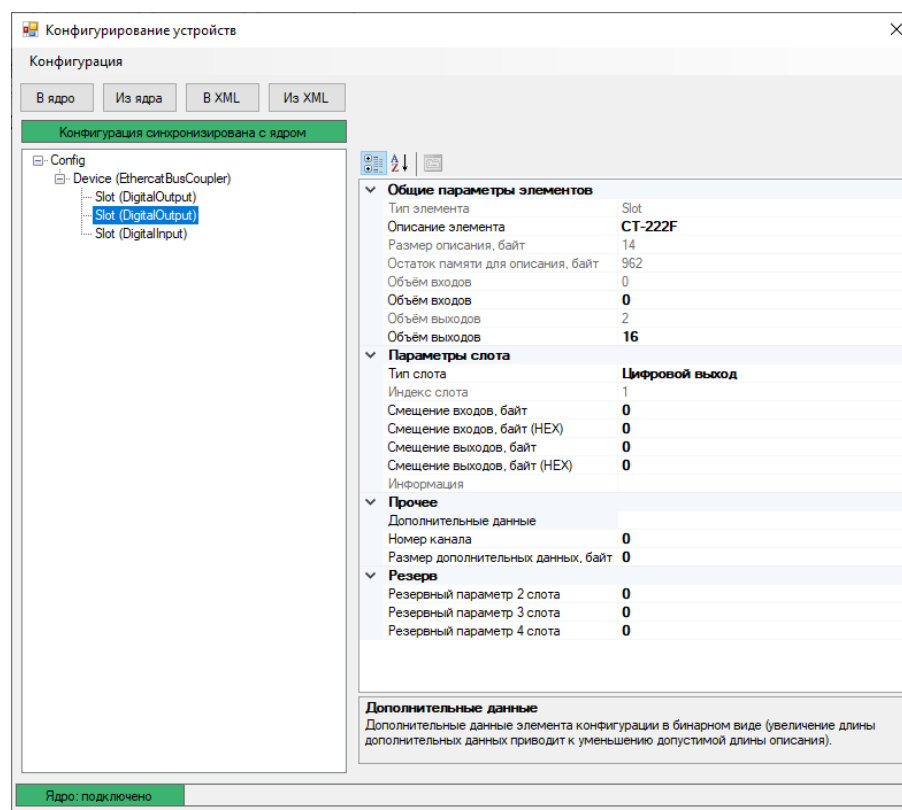


Рисунок 57 – Конфигурация второго модуля дискретных выходов СТ-222F

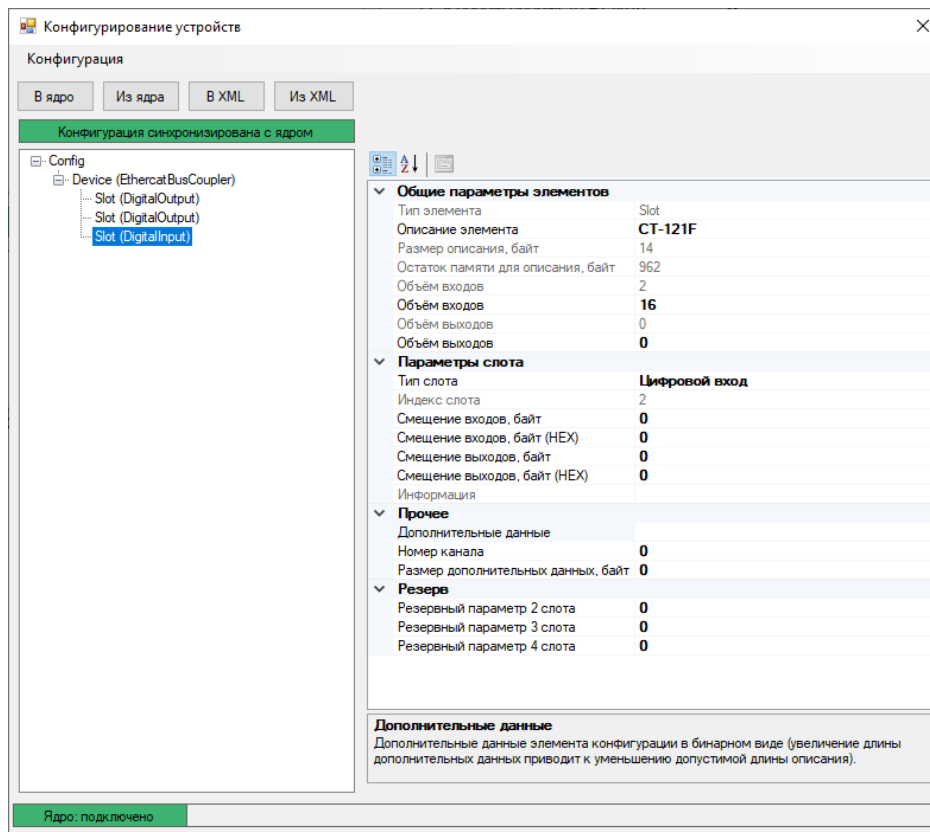


Рисунок 58 – Конфигурация модуля дискретных входов CT-121F

- послать конфигурацию в ядро и перезагрузить его.

Программа ПЛК, разработанная в FBEditor для тестирования выходов:

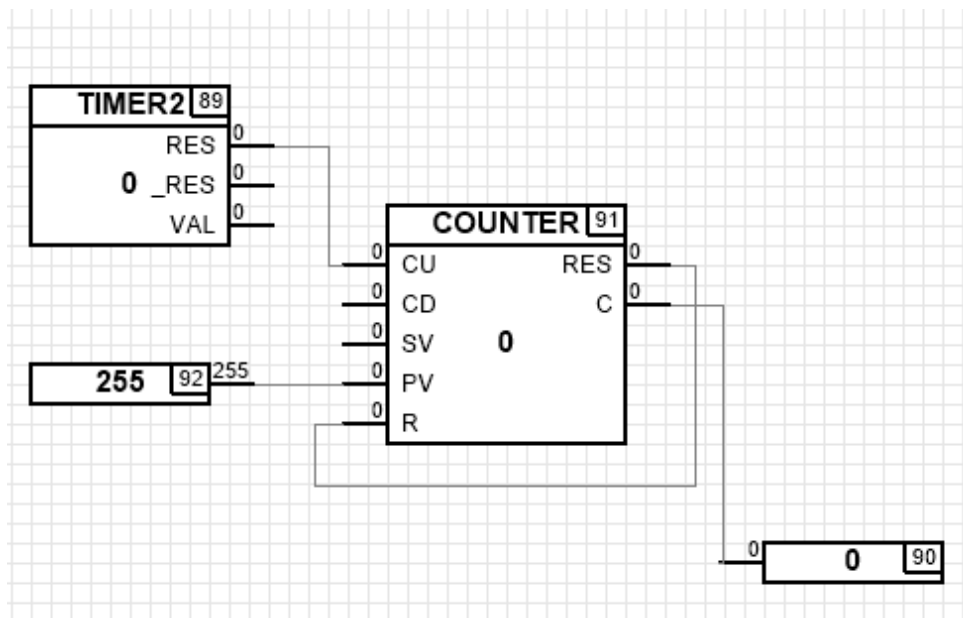


Рисунок 59 – Программа ПЛК для тестирования модуля дискретных выходов

Параметры выходного блока программы ПЛК показаны на Рисунок 60:

1. Основные параметры	
Идентификатор объекта	206
Тип объекта	OUT
Привязка к переменной	False
Имя объекта	
Описание объекта	
2. Графические параметры	
X	1480
Y	355
3. Параметры блока	
Статический адрес объект	Да
Количество входов	1
Количество выходов	0
4. Параметры входа/выхода	
Тип хранимого значения	Digital
Длина	4
Значение входа/выхода	0
Вид входа-выхода	CommonPlcMemory
Адрес (порта или памяти)	256000
Индекс бита	0
Разрешение изменять зна	False

Рисунок 60 – Параметры выходного блока ПЛК программы

На Рисунок 61 показано распределение памяти в CommonPlcMemory при работе указанной программы.

		Авто	Готов	Канал 1	ПЛК				19.11.2022																																																																											
									23:40:02																																																																											
									00:33:23																																																																											
Распределение общей памяти ПЛК									S1																																																																											
<table border="1"> <thead> <tr> <th>Адрес памяти</th> <th colspan="4">Распределение памяти</th> </tr> </thead> <tbody> <tr><td>0x03E800</td><td>B8 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E810</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E820</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E830</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E840</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E850</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E860</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E870</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E880</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E890</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E8A0</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E8B0</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E8C0</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> <tr><td>0x03E8D0</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td><td>00 00 00 00</td></tr> </tbody> </table>									Адрес памяти	Распределение памяти				0x03E800	B8 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E810	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E820	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E830	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E840	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E850	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E860	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E870	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E880	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E890	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E8A0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E8B0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E8C0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	0x03E8D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	S2
Адрес памяти	Распределение памяти																																																																																			
0x03E800	B8 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E810	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E820	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E830	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E840	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E850	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E860	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E870	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E880	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E890	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E8A0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E8B0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E8C0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
0x03E8D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00																																																																																
<table border="1"> <thead> <tr> <th colspan="2">Задание адреса в памяти</th> </tr> </thead> <tbody> <tr> <td>Адрес [Dec]</td> <td>256000</td> </tr> <tr> <td>Адрес, 0x[Hex]</td> <td>3E800</td> </tr> <tr> <td colspan="2">Задание значений</td> </tr> <tr> <td>Размер (Байт)</td> <td>8</td> </tr> <tr> <td>Тип</td> <td>Беззнаковое</td> </tr> <tr> <td>Значение (Dec)</td> <td>0</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>0</td> </tr> <tr> <td colspan="2">Текущее значение</td> </tr> <tr> <td>Значение (Dec)</td> <td>184</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>B8</td> </tr> </tbody> </table>									Задание адреса в памяти		Адрес [Dec]	256000	Адрес, 0x[Hex]	3E800	Задание значений		Размер (Байт)	8	Тип	Беззнаковое	Значение (Dec)	0	Значение, 0x(HEX)	0	Текущее значение		Значение (Dec)	184	Значение, 0x(HEX)	B8	S3																																																					
Задание адреса в памяти																																																																																				
Адрес [Dec]	256000																																																																																			
Адрес, 0x[Hex]	3E800																																																																																			
Задание значений																																																																																				
Размер (Байт)	8																																																																																			
Тип	Беззнаковое																																																																																			
Значение (Dec)	0																																																																																			
Значение, 0x(HEX)	0																																																																																			
Текущее значение																																																																																				
Значение (Dec)	184																																																																																			
Значение, 0x(HEX)	B8																																																																																			
<table border="1"> <thead> <tr> <th colspan="2">Подготовленное значение для записи в память (bin)</th> </tr> </thead> <tbody> <tr> <td>0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000</td> <td></td> </tr> <tr> <td>63 47 31 15 0</td> <td></td> </tr> <tr> <td colspan="2">Текущее значение в памяти (bin)</td> </tr> <tr> <td>0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000</td> <td></td> </tr> <tr> <td>63 47 31 15 0</td> <td></td> </tr> </tbody> </table>									Подготовленное значение для записи в память (bin)		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000		63 47 31 15 0		Текущее значение в памяти (bin)		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000		63 47 31 15 0		S4																																																															
Подготовленное значение для записи в память (bin)																																																																																				
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000																																																																																				
63 47 31 15 0																																																																																				
Текущее значение в памяти (bin)																																																																																				
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000																																																																																				
63 47 31 15 0																																																																																				
<table border="1"> <thead> <tr> <th colspan="2">Задание значения</th> </tr> </thead> <tbody> <tr> <td>Адрес [Dec]</td> <td>256000</td> </tr> <tr> <td>Адрес, 0x[Hex]</td> <td>3E800</td> </tr> <tr> <td colspan="2">Задание значений</td> </tr> <tr> <td>Размер (Байт)</td> <td>8</td> </tr> <tr> <td>Тип</td> <td>Беззнаковое</td> </tr> <tr> <td>Значение (Dec)</td> <td>0</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>0</td> </tr> <tr> <td colspan="2">Текущее значение</td> </tr> <tr> <td>Значение (Dec)</td> <td>184</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>B8</td> </tr> </tbody> </table>									Задание значения		Адрес [Dec]	256000	Адрес, 0x[Hex]	3E800	Задание значений		Размер (Байт)	8	Тип	Беззнаковое	Значение (Dec)	0	Значение, 0x(HEX)	0	Текущее значение		Значение (Dec)	184	Значение, 0x(HEX)	B8	S5																																																					
Задание значения																																																																																				
Адрес [Dec]	256000																																																																																			
Адрес, 0x[Hex]	3E800																																																																																			
Задание значений																																																																																				
Размер (Байт)	8																																																																																			
Тип	Беззнаковое																																																																																			
Значение (Dec)	0																																																																																			
Значение, 0x(HEX)	0																																																																																			
Текущее значение																																																																																				
Значение (Dec)	184																																																																																			
Значение, 0x(HEX)	B8																																																																																			
<table border="1"> <thead> <tr> <th colspan="2">Подготовленное значение для записи в память (bin)</th> </tr> </thead> <tbody> <tr> <td>0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000</td> <td></td> </tr> <tr> <td>63 47 31 15 0</td> <td></td> </tr> <tr> <td colspan="2">Текущее значение в памяти (bin)</td> </tr> <tr> <td>0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000</td> <td></td> </tr> <tr> <td>63 47 31 15 0</td> <td></td> </tr> </tbody> </table>									Подготовленное значение для записи в память (bin)		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000		63 47 31 15 0		Текущее значение в памяти (bin)		0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000		63 47 31 15 0		S6																																																															
Подготовленное значение для записи в память (bin)																																																																																				
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000																																																																																				
63 47 31 15 0																																																																																				
Текущее значение в памяти (bin)																																																																																				
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000																																																																																				
63 47 31 15 0																																																																																				
<table border="1"> <thead> <tr> <th colspan="2">Задание значения</th> </tr> </thead> <tbody> <tr> <td>Адрес [Dec]</td> <td>256000</td> </tr> <tr> <td>Адрес, 0x[Hex]</td> <td>3E800</td> </tr> <tr> <td colspan="2">Задание значений</td> </tr> <tr> <td>Размер (Байт)</td> <td>8</td> </tr> <tr> <td>Тип</td> <td>Беззнаковое</td> </tr> <tr> <td>Значение (Dec)</td> <td>0</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>0</td> </tr> <tr> <td colspan="2">Текущее значение</td> </tr> <tr> <td>Значение (Dec)</td> <td>184</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>B8</td> </tr> </tbody> </table>									Задание значения		Адрес [Dec]	256000	Адрес, 0x[Hex]	3E800	Задание значений		Размер (Байт)	8	Тип	Беззнаковое	Значение (Dec)	0	Значение, 0x(HEX)	0	Текущее значение		Значение (Dec)	184	Значение, 0x(HEX)	B8	S7																																																					
Задание значения																																																																																				
Адрес [Dec]	256000																																																																																			
Адрес, 0x[Hex]	3E800																																																																																			
Задание значений																																																																																				
Размер (Байт)	8																																																																																			
Тип	Беззнаковое																																																																																			
Значение (Dec)	0																																																																																			
Значение, 0x(HEX)	0																																																																																			
Текущее значение																																																																																				
Значение (Dec)	184																																																																																			
Значение, 0x(HEX)	B8																																																																																			
<table border="1"> <thead> <tr> <th colspan="2">Задание значения</th> </tr> </thead> <tbody> <tr> <td>Адрес [Dec]</td> <td>256000</td> </tr> <tr> <td>Адрес, 0x[Hex]</td> <td>3E800</td> </tr> <tr> <td colspan="2">Задание значений</td> </tr> <tr> <td>Размер (Байт)</td> <td>8</td> </tr> <tr> <td>Тип</td> <td>Беззнаковое</td> </tr> <tr> <td>Значение (Dec)</td> <td>0</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>0</td> </tr> <tr> <td colspan="2">Текущее значение</td> </tr> <tr> <td>Значение (Dec)</td> <td>184</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>B8</td> </tr> </tbody> </table>									Задание значения		Адрес [Dec]	256000	Адрес, 0x[Hex]	3E800	Задание значений		Размер (Байт)	8	Тип	Беззнаковое	Значение (Dec)	0	Значение, 0x(HEX)	0	Текущее значение		Значение (Dec)	184	Значение, 0x(HEX)	B8	S8																																																					
Задание значения																																																																																				
Адрес [Dec]	256000																																																																																			
Адрес, 0x[Hex]	3E800																																																																																			
Задание значений																																																																																				
Размер (Байт)	8																																																																																			
Тип	Беззнаковое																																																																																			
Значение (Dec)	0																																																																																			
Значение, 0x(HEX)	0																																																																																			
Текущее значение																																																																																				
Значение (Dec)	184																																																																																			
Значение, 0x(HEX)	B8																																																																																			
<table border="1"> <thead> <tr> <th colspan="2">Задание значения</th> </tr> </thead> <tbody> <tr> <td>Адрес [Dec]</td> <td>256000</td> </tr> <tr> <td>Адрес, 0x[Hex]</td> <td>3E800</td> </tr> <tr> <td colspan="2">Задание значений</td> </tr> <tr> <td>Размер (Байт)</td> <td>8</td> </tr> <tr> <td>Тип</td> <td>Беззнаковое</td> </tr> <tr> <td>Значение (Dec)</td> <td>0</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>0</td> </tr> <tr> <td colspan="2">Текущее значение</td> </tr> <tr> <td>Значение (Dec)</td> <td>184</td> </tr> <tr> <td>Значение, 0x(HEX)</td> <td>B8</td> </tr> </tbody> </table>									Задание значения		Адрес [Dec]	256000	Адрес, 0x[Hex]	3E800	Задание значений		Размер (Байт)	8	Тип	Беззнаковое	Значение (Dec)	0	Значение, 0x(HEX)	0	Текущее значение		Значение (Dec)	184	Значение, 0x(HEX)	B8	EN																																																					
Задание значения																																																																																				
Адрес [Dec]	256000																																																																																			
Адрес, 0x[Hex]	3E800																																																																																			
Задание значений																																																																																				
Размер (Байт)	8																																																																																			
Тип	Беззнаковое																																																																																			
Значение (Dec)	0																																																																																			
Значение, 0x(HEX)	0																																																																																			
Текущее значение																																																																																				
Значение (Dec)	184																																																																																			
Значение, 0x(HEX)	B8																																																																																			
F2	F3 Обновлять	F4 Перейти на байт влево	F5 Перейти на байт вправо	F6 Перейти	F7 Запись	F8 Регистры данных ПЛК	F9 Возврат																																																																													

Рисунок 61 – Распределение памяти в CommonPlcMemory