УТВЕРЖДЕН
РСКЮ.20507-01 33 01-УД

Программа «Среда разработки и отладки программ» (СРиО)

Руководство программиста

РСКЮ.20507-01 33 01

МН

Листов 200

Версия:

от 17 января 2024

2024

Литера « »

Инв. № подл.	**Подп. и дата**	**Взам. инв. №**	**Инв. № дубл.**	**Подп. и дата**

АННОТАЦИЯ

В настоящем «Руководстве программиста» представлено описание порядка работы с программой «Среда разработки и отладки программ» (СРиО), входящей в состав программного изделия «Программная платформа для разработки и отладки программного обеспечения программируемых контроллеров семейства Багет» (ПП ПО ПЛК Багет).

Документ содержит информацию о назначении программы, условиях применения, элементах графического интерфейса пользователя, порядке разработки прикладных программ, о плагинах внешних устройств связи с объектом (УСО).

Описан процесс работы с редакторами языков программирования ПЛК стандарта МЭК 61131-3-2016 и режимом отладки программ. Приведены рисунки, связанные с сообщениями графического интерфейса пользователя и описание их содержания.

В Приложении А приведена информация об установке программы.

Настоящее «Руководство программиста» относится к версии программы «srio-1.0.15» («Программное обеспечение Багет-ПЛК1» (ПО ПЛК1) ЮКСУ.91292-01 12 01) от 02 декабря 2024.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Назначение и функции программы

Программа «Среда разработки и отладки программ» (далее – СРиО) предназначена для создания и отладки прикладных программ на языках стандарта МЭК 61131-3-2016 для программируемых логических контроллеров семейства Багет (далее по тексту – ПЛК).

Программа СРиО включает средства разработки специального программного обеспечения (далее – СПО), которые обеспечивают:

- создание программ на языках МЭК и Си;
- создание карты соответствия входных и выходных сигналов оборудования объекта автоматизации переменным прикладной программы ПЛК;
- синтаксическую верификацию программ, написанных на текстовых языках МЭК, Си;
- верификацию промежуточного (текстового) представления программ, написанных на графических языках МЭК;
- коллективную разработку СПО для ПЛК в формате проектов (создание, модификация, удаление, импорт, экспорт проектов);
- симуляцию программ в инструментальной среде без использования ПЛК;
- управление версиями проектов и программ;
- дистанционную отладку и управление СПО ПЛК.

Программа СРиО обеспечивает следующие возможности отладки СПО:

- загрузку и запуск СПО;
- управление выполнением СПО;
- установку точек снятия протоколов отладки (трассы) и спецификацию переменных проекта, для которых возможен просмотр/редактирование значений в этих точках;
- визуальное отслеживание выполнения программ, написанных на графических языках МЭК, и изменение значения различных графических элементов языка;
- отображение отслеживаемых значений переменных в виде графика;
- выгрузку на инструментальную ЭВМ протокола отладки (трассы) и просмотр протокола;
- опрос состояния ПЛК.

В качестве языков описания алгоритмов и логики работы данных программ, могут выступать:

– Structured Text (структурированный текст, далее – ST);
– Instruction List (список инструкций, далее – IL);
– Function Block Diagram (функциональные блоковые диаграммы, далее FBD);
– Ladder Diagram (релейно-контактная схема, далее – LD);
– Sequential Function Chart (последовательная функциональная схема, далее SFC).

1.2. Условия применения

Для выполнения программы необходима инструментальная ЭВМ (ИЭВМ) с микропроцессорной архитектурой Intel с тактовой частотой процессора не ниже 1000 МГц (64-битный) с 1 Гб ОЗУ и не менее 250 Мб свободного дискового пространства, укомплектованная сетевым адаптером Ethernet, обеспечивающим взаимодействие с ПЛК «Багет-ПЛК1».

Инструментальная ЭВМ должна функционировать под управлением ОС Linux (Astra Linux SE версии 1.7.5 «Орел» или Fedora 38 Workstation). При установке операционной системы необходимо указать основной язык – русский.

Для запуска и работы СРиО, необходима наличие интерпретатора Python (версии 2.7) с набором установленных библиотек. Порядок установки программы СРиО на ИЭВМ приведен в Приложении А.

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

Программа СРиО позволяет работать в конфигурационном режиме и в режиме исполнения прикладной программы. В конфигурационном режиме происходит создание прикладной программы, написание алгоритмов и логики её основных программных модулей, их связывание с внешними модулями УСО (устройств связи с объектом). В режиме исполнения среда разработки работает с прикладной программой на целевом устройстве или в режиме симуляции для инструментальной ЭВМ.

Задачей среды СРиО является разработка и отладка программ (СПО ПЛК), написанных на языках МЭК.

Для решения задачи применяются следующие подходы:

– для создания и редактирования программ на языках МЭК и Си используются технологии на базе кода свободно распространяемой среды разработки приложений Beremiz;
– для задания карты соответствия сигналов оборудования объекта автоматизации переменным программы СПО ПЛК используется механизм плагинов УСО (устройств связи с объектом);
– для симуляции программ в инструментальной среде ЭВМ без использования ПЛК используется сервис симуляции для инструментальной ЭВМ (СС ИЭВМ) РСКЮ.20513-01;
– для дистанционной отладки СПО ПЛК применяются библиотека поддержки протокола отладки для среды ПЛК (БПП ПЛК) РСКЮ.20509-01 и библиотека поддержки протокола отладки для среды инструментальной ЭВМ (БПП ИЭВМ) РСКЮ.20510-01;
– для управления выполнением СПО и отслеживания изменений переменных используется сервис исполнения для ПЛК (СИ ПЛК) РСКЮ.20506-01.

Программа СРиО разработана на базе технологий, используемых в свободно распространяемой среде разработки приложений Beremiz.

3. АРХИТЕКТУРА ПРОГРАММЫ

С точки зрения архитектуры СРиО представляет собой расширение для свободно распространяемой среды разработки приложений Beremiz, которое обеспечивает разработку и отладку программ, написанных на языках МЭК для ПЛК «Багет-ПЛК1» (ЮКСУ.421457.002).

СРиО написана на языке программирования Python 2.7 в объектно-ориентированном стиле. Класс главного приложения – BagetIDELauncher наследует класс BeremizIDELauncher из состава СПО Beremiz, дополняет его новыми функциями и переопределяет уже существующие:

1) добавляет поддержку протокола БПП ИЭВМ;
2) включает в СРиО средства для использования модулей ввода-вывода устройств ПЛК, библиотек поддержки протоколов передачи данных Modbus RTU, Modbus TCP, OPC UA, МЭК 101, МЭК 104 и MLCP;
3) реализует удалённые вызовы к сервису управления для ПЛК (СИ ПЛК) для передачи команд и обмена данными с СПО ПЛК на этапе отладки;
4) реализует локальные и удалённые вызовы к сервису симуляции для среды инструментальной ЭВМ (СС ИЭВМ) для тестирования СПО без загрузки исполняемого образа на ПЛК;
5) реализует интеграцию СРиО с компилятором СКРВ Багет 3.3;
6) обеспечивает сборку исполняемого образа для Багет ПЛК1.

4. ОБРАЩЕНИЕ К ПРОГРАММЕ, ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Общая схема по созданию прикладной программы в среде разработки и отладки приведена на рис. 1.

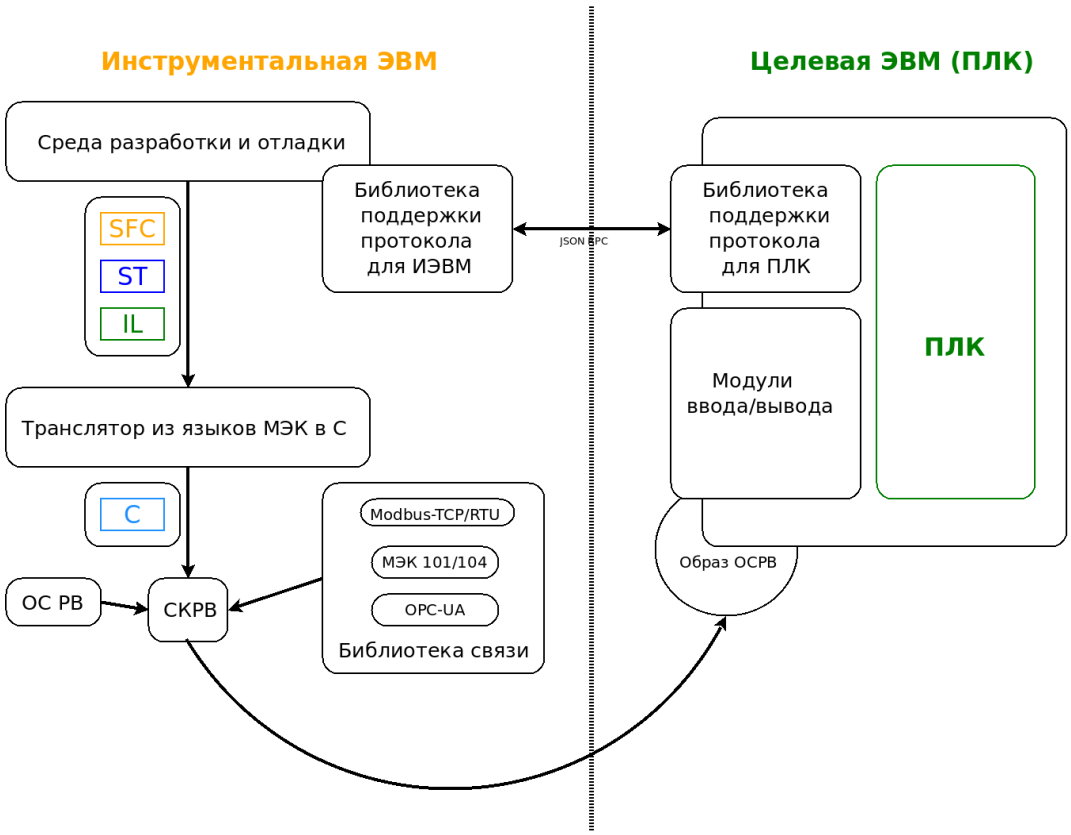


Рисунок 1 – Обобщённая схема создания прикладной программы в СРиО

Входными данными являются программные модули, написанные программистом (инженером по автоматизации) на текстовых (ST, IL) и/или графических (FBD, SFC, LD) языках в соответствии со стандартом МЭК 61131-3-2016, объединённые в проект. Каждый такой проект представлен в формате XML и хранится в файле формата .bgt, представляющем собой zip-архив с опциональной защитой паролем с шифрованием AES-256.

Выходными данными является сгенерированный исходный код, библиотечные файлы, исполняемый образ для Багет ПЛК1:

1) файл <название проекта>.st, содержащий промежуточный код на языке ST, сгенерированный для всех программных модулей и ресурсов, транслируемый в язык С;
2) файлы: config.c config.h, POUS.h, POUS.c и файлы, соответствующие ресурсам, содержат код (на языке С) реализации алгоритмов и логики работы программных модулей и ресурсов проекта;

3) файлы `plc_common_main.c` и `plc_debugger.c` содержащие код специфичный для целевой архитектуры и код для отладки прикладной программы на целевом устройстве из среды разработки соответственно;
4) файлы, содержащие код драйверов на языке С для взаимодействия с внешними модулями УСО;
5) библиотечный файл, компилируемый из вышеперечисленных С файлов;
6) исполняемый образ для Багет ПЛК1 с включённым библиотечным файлом.

Сгенерированный С код, с помощью СКРВ Багет 3.3 (ЮКСУ.90977-01) компилируется в исполняемый бинарный файл, включаемый в состав библиотечного файла, который, в свою очередь, включается в состав исполняемого образа ОС РВ Багет 2.7.

На целевом устройстве (ПЛК «Багет-ПЛК1») исполняемый образ в процессе работы выполняет следующие действия:

– с помощью драйверов модулей УСО обменивается данными с внешними модулями;
– с помощью драйверов протоколов обмениваются данными с внешними устройствами;
– исполняет алгоритмы и логику, определённую пользователем в программных модулях проекта;
– предоставляет данные для трансляции в системы верхнего уровня;
– сохраняет и транслирует информацию для отладки прикладных программ.

Компиляция в двоичный код осуществляется компилятором для целевой платформы (СКРВ Багет 3.3). Вызов компилятора не интегрирован в транслятор ТР ПЛК (РСКЮ.20505-01), а осуществляется средствами среды разработки и отладки программ.

Транслятор ТР ПЛК осуществляет лексический, синтаксический и семантический анализы (детальное описание действий в рамках каждого шага описано в документации на ТР ПЛК) и в случае успеха преобразует программы с языка МЭК в программы на языке Си.

В случае если в ходе работы ТР ПЛК обнаруживаются ошибки в исходном коде на экран выводятся диагностические сообщения на каждую найденную ошибку, а генерация Си-кода отменяется.

ТР ПЛК генерирует исходный код на языке Си, пригодный к последующей компиляции компилятором СКРВ Багет 3.3 и включёния полученных бинарных файлов в состав исполняемого образ ОС РВ Багет 2.7 для ПЛК «Багет-ПЛК1».

Формирование исполняемого образа ОС РВ с СПО осуществляется автоматически. Список библиотек, включаемых в ОС РВ Багет 2.7 (поле

конфигуратора «Подключение прикладных программ / Библиотеки») должен содержать полное имя файла программы СПО ПЛК в загрузочном виде:

```
<PLCBASE>/lastbuildPLC.a
```

а в качестве точки входа «`User thread entry`» должна быть указана функция `__start()`.

Здесь и ниже `<PLCBASE>` - директория сборки проекта, которая задаётся на панели настройки сборки проекта для ОС РВ Багет в среде СРиО.

5. ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

IEC 61131-3 – раздел международного стандарта МЭК 61131 (также существует соответствующий европейский стандарт EN 61131), описывающий языки программирования для программируемых логических контроллеров.

ГОСТ Р МЭК 61131-3-2016 – национальный стандарт Российской Федерации, идентичный международному стандарту IEC 61131-3:2013.

Устройство связи с объектом (УСО) - устройство для объединения аналоговых и цифровых параметров реального объекта автоматизации. Предназначено для ввода сигналов с объекта в автоматизированную систему и вывода сигналов на объект.

Специальное программное обеспечение (СПО) – пользовательское или прикладное программное обеспечение.

Среда разработки для языков стандарта IEC 61131-3 – система программных средств, используемая инженерами по автоматизации, для разработки прикладного программного обеспечения на высокоуровневых языках стандарта IEC 61131-3 под различные целевые платформы, которая включает в себя:

- текстовые и графические редакторы языков стандарта IEC 61131-3;
- транслятор диаграмм графических языков в текстовый язык;
- транслятор текстового языка в язык С;
- механизмы плагинов для взаимодействия с модулями УСО;
- механизмы соединений с целевыми устройствами;
- отладчик.

Модули УСО – модули ввода/вывода, обеспечивающие подключение датчиков и исполнительных механизмов.

Целевое устройство – аппаратное средство с определённой архитектурой процессора, на котором могут исполняться различные исполняемые файлы, обращающиеся с помощью него к модулям УСО.

Прикладная программа (исполняемый файл) для целевого устройства – скомпилированный и скомпонованный файл, который может выполняться на целевом устройстве (в т. ч. путем включения в состав ОС).

Плагин для модуля УСО – интерфейс, состоящий из специальных драйверов и элементов пользовательского интерфейса для среды разработки и отладки, позволяющий связывать переменные модулей УСО с переменными программных модулей, из которых состоит проект.

Плагин для протокола коммуникации – интерфейс, состоящий из специальных драйверов и элементов пользовательского интерфейса для среды разработки и отладки, позволяющий связывать объекты информационного обмена по коммуникационным протоколам с переменными программных модулей, из которых состоит проект.

Проект – совокупность программных модулей (программ, функциональных блоков, функций), плагинов внешних модулей УСО, ресурсов, пользовательских типов данных, сборка (компиляция и компоновка) которых, представляет собой прикладную программу для целевого устройства. Каждый проект сохраняется в отдельном файле.

Переменная – область памяти, в которой находятся данные, с которыми оперирует программный модуль.

Ресурс – элемент, отвечающий за конфигурацию проекта: глобальные переменные и экземпляры проекта, связываемыми с программными модулями типа «Программа» и задачами.

Программный модуль (POU) – элемент, представляющий собой функцию, функциональный блок или программу. Каждый программный модуль состоит из раздела объявлений и кода. Для написания всего кода программного модуля используется только один из языков программирования стандарта IEC 61131-3.

Функция – программный модуль, который возвращает только единственное значение, которое может состоять из одного и нескольких элементов (если это битовое поле или структура).

Функциональный блок – программный модуль, который принимает и возвращает произвольное число значений, а также позволяет сохранять своё состояние (подобно классу в различных объектно-ориентированных языках). В отличие от функции функциональный блок не формирует возвращаемое значение.

Программа – программный модуль, представляющий собой единицу исполнения, как правило, связывается (ассоциируется) с задачей.

Задача – элемент представляющий время и приоритет выполнения программного модуля типа «Программа» в рамках экземпляра проекта.

Экземпляр – представляет собой программу, как единицу исполнения, связанную (ассоциированную) с определённой задачей. Так же, как экземпляр, рассматриваются переменные, определённые в программных модулях: программа и функциональный блок.

Пользовательский тип данных – тип данных, добавленный в проект и представляющий собой: псевдоним существующего типа, поддиапазон существующего типа, перечисление, массив или структуру.

Сервис симуляции для инструментальной ЭВМ (СС ИЭВМ) - предназначен для тестирования СПО на ЭВМ без загрузки исполняемого образа на ПЛК.

ИЭВМ - инструментальная ЭВМ с микропроцессорной архитектурой Intel, под управлением ОС Linux, в соответствии с п.1.2 «Условия применения».

URI системы исполнения – унифицированный (единообразный) идентификатор ресурса (под ресурсом понимается целевое устройство). Указывается в формате:

`JSONRPC://<IP-адрес целевого устройства>:<порт>`

6. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Пользовательский интерфейс СРиО включает в себя интерфейсные компоненты, обеспечивающие доступ к функциональным возможностям данной программы, в том числе:

– главное меню программы;
– панель инструментов;
– отображение структуры проекта;
– доступ к настройкам проекта и к файлам проекта;
– средства для работы с переменными и константами проекта;
– отображение промежуточного кода;
– окно текстового редактора языка Си;
– окно текстовых редакторов языков ST и IL;
– окно графических редакторов языков FBD, SFC и LD;
– средства редактирования ресурса;
– доступ к экземплярам проекта;
– доступ к библиотеке функций и функциональным блокам;
– средства поиска элементов в проекте;
– доступ к средствам отладки;
– средства отображения графика изменения значения переменной в режиме отладки;
– средства настройки параметров компиляции и сборки исполняемого образа для «Багет ПЛК1»;
– консоль компиляции и сборки;
– панель симуляции программ в инструментальной среде.

6.1. Главное меню программы

Главное меню содержит следующие пункты (рис. 2):

– «Файл»;
– «Редактировать»;
– «Вид»;
– «Помощь».

Рисунок 2 – Главное меню программы

6.1.1. Пункт меню «Файл»

Предназначен для работы с проектом и предоставляет следующие пункты:

– «Новый» – создание нового проекта;
– «Открыть» – открытие существующего проекта;
– «Недавние проекты» – быстрое открытие одного из десяти последних, недавно редактированных проектов;
– «Сохранить» – сохранение текущего проекта пункт;
– «Сохранить как» – сохранение текущего проекта в папку отличную от той, в которой он сохранён на данный момент;
– «Экспорт» – экспорт проекта в XML файл;
– «Импорт» – импорт ранее экспортированного кода или кода другого проекта в текущий;
– «Закрыть вкладку» – закрытие активной вкладки (например, вкладки переменных плагина, конфигурации и т.д.) для открытого проекта;
– «Закрыть проект» – закрыть текущий, открытый проект;
– «Подключить библиотеку» – открывает диалоговое окно для подключения к текущему открытому проекту библиотеки на языке С в виде библиотеки в двоичном формате (файл *.a) и каталога с заголовочными файлами. Файл библиотеки добавляется к флагам компоновщика в разделе конфигурации проекта (LDFLAGS), каталог с заголовочными файлами библиотеки добавляется к флагам Си компилятора в разделе конфигурации проекта (CFLAGS);
– «Настройки страницы» – настройка параметров страницы для печати на принтере активной программы, представленной в виде диаграммы;
– «Просмотр» – просмотр активной программы перед печатью на принтере;
– «Печать» – печать на принтере активной программы;
– «Выход» – закрытие текущего проекта и выход из среды разработки и отладки.

6.1.2. Пункт меню «Редактировать»

Предназначен для работы с редакторами языков стандарта IEC 61131-3 и предоставляет следующие возможности:

– «Отмена» – отмена последней манипуляции в редакторе;
– «Повторить» – повтор отменённой манипуляции в редакторе;
– «Вырезать» – удалить в буфер обмена выделенный(е) элемент(ы) в редакторе;
– «Копировать» – копировать в буфер обмена выделенный(е) элемент(ы) в редакторе;

– «Вставить» – вставить из буфера обмена находящиеся там элемент(ы) в редактор;
– «Поиск» – поиск в текущем функциональном блоке;
– «Поиск следующего» – подсветка следующего вхождения строки поиска;
– «Поиск предыдущего» – подсветка предыдущего вхождения строки поиска;
– «Поиск в проекте» – вызов диалога поиска данных в проекте;
– «Добавить элемент» – добавление одного из следующих элемента в текущий проект:
    1) «Тип данных» – нового типа данных;
    2) «Функции» – новой функции;
    3) «Функциональный блок» – нового функционального блока;
    4) «Программы» – новую программу;
    5) «Ресурсы» – новый ресурс;
    6) «С-расширение» – добавить в проект расширение на языке Си;
    7) Плагины для модулей УСО (см. п. 7.6);
– «Выделить всё» – выделение всех элементов в активной вкладке редактора;
– «Удалить» – удаление программного модуля, выделенного в дереве проекта (см. п. 6.3.2).

6.1.3. Пункт меню «Вид»

Предназначен для работы с редакторами языков стандарта IEC-61131 и предоставляет следующие возможности:

– «Обновить» – обновление данных и снятие выделения в редакторе;
– «Очистить ошибки» – очистка указателей ошибок в редакторе;
– «Приближение» – выбор масштаба в процентах;
– «Сменить представление» – скрывает боковые панели и разворачивает активное окно редактора на весь экран;
– «Полный экран» – разворачивает окно среды на весь экран (для выхода следует использовать Shift+F12);
– «Сброс расположения панелей» – восстановление расположения панелей среды разработки и отладки программ в исходное состояние.

6.1.4. Пункт меню «Помощь»

Предназначен для вывода справочной информации:

– «О программе» – вывод краткой информации о программе СРиО, включая текущую версию.
– «Документация» – открывает «Руководство программиста» через приложение для просмотра документов.

6.2. Панель инструментов

Панель инструментов представляет собой панель с кнопками для быстрого обращения к часто используемым функциям среды разработки и отладки программ. Она состоит из нескольких панелей, содержащих кнопки: главного меню, сборки проекта и установки связи с целевым устройством.

Подробнее об этих панелях будет рассказано ниже. При редактировании программных модулей, написанных на графических языках, появляются дополнительные панели с кнопками. Они рассмотрены при описании редакторов графических языков стандарта МЭК 61131-3-2016 (см. п. 6.9).

6.2.1. Панель инструментов главного меню

Панель инструментов, содержащая кнопки главного меню представлена на рисунке 3.

Рисунок 3 – Панель инструментов

Список кнопок и их функций описывается в табл. 1.

Таблица 1 – Кнопки панели инструментов

Кнопка	Наименование кнопки	Функции кнопки
	Новый	Создать новый проект
	Открыть	Открыть существующий проект
	Сохранить	Сохранить текущий проект
	Сохранить как…	Сохранить текущий проект в определённую папку
	Печать	Печать текущей программы на принтере
	Отменить	Отмена последнего действия в среде разработки
	Повторить	Повтор отменённого действия в среде разработки
	Вырезать	Удалить в буфер обмена выделенный(е) элемент(ы) в редакторе
	Копировать	Копировать в буфер обмена выделенный(е) элемент(ы) в редакторе

Кнопка	Наименование кнопки	Функции кнопки
	Вставить	Вставить из буфера обмена находящиеся там элемент(ы) в редактор
	Поиск в проекте	Вызов диалога поиска данных в проекте
	Переключить в полноэкранный режим	Разворачивает/сворачивает окно среды на весь экран
	Настроить связь с ПЛК	Вызов диалога настройки связи с ПЛК
	Логин	Вызов диалога для авторизации пользователя со вводом логина и пароля
	Логаут	Окончание работы авторизованного пользователя
	Перезагрузка	Перезагрузка ПЛК
	Перезагрузка в технологический образ	Перезагрузка в технологический образ
	Обновить технологический образ	Обновление технологического образа
	Настройка пользователей	Вызов диалога для настройки списка пользователей
	Получить журнал безопасности	Сохранить журнал безопасности в определённую папку
	Запустить конфигуратор модулей ввода-вывода	Вызов диалога для запуска конфигуратора модулей ввода-вывода
	Получить список логов	Вызов диалога для выбора файлов истории и сохранения их в определённую папку
	Скачать проект	Сохранение проекта, загруженного в ПЛК, в определённую папку
	Получить статус ПЛК	Получение статуса ПЛК

6.2.2. Панель инструментов работы с проектом

Панель, содержащая кнопки сборки проекта и соединения с целевым устройством, позволяет скомпилировать и скомпоновать текущий проект и, в случае, если эта операция завершилась успешно (данную информацию можно увидеть в отладочной консоли (см. п. 6.14), передать и запустить полученный исполняемый файл на целевом устройстве.

На рисунке 4 приведено состояние данной панели, когда соединение с целевым устройством не установлено.

Рисунок 4 – Панель сборки проекта, когда соединение с целевым устройством не установлено

На рисунке 5 приведено состояние данной панели, когда соединение с целевым устройством установлено, но ПЛК не запущен.

Рисунок 5 – Панель сборки проекта, когда соединение с целевым устройством установлено, но ПЛК не запущен

На рисунке 6 приведено состояние данной панели, когда соединение с целевым устройством установлено и ПЛК запущен.

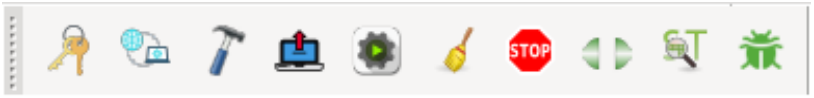


Рисунок 6 – Панель сборки проекта, когда соединение с целевым устройством установлено и ПЛК запущен

На рисунке 7 приведено состояние данной панели, когда соединение с целевым устройством установлено, ПЛК запущен и активирован режим пошаговой отладки.

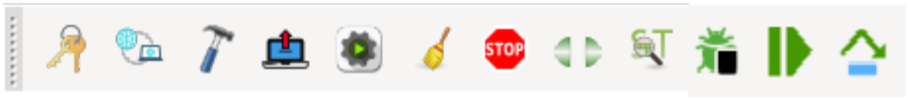


Рисунок 7 – Панель сборки проекта, когда соединение с целевым устройством установлено, ПЛК запущен, активирован режим пошаговой отладки

В зависимости от того, произведено в настоящий момент времени соединение с целевым устройством или выполняется ли прикладная программа на нём, появляются и скрываются некоторые кнопки.

В случае, когда при установке соединения произошли ошибки, данная информация будет выведена в отладочную консоль (см. п. 6.14).

Полный список всех кнопок, относящихся к сборке проекта и соединению с целевым устройством, представлен в табл. 2.

Таблица 2 – Кнопки сборки проекта и связи с целевым устройством на панели инструментов

Кнопка	Наименование кнопки	Функции кнопки
	Пароль проекта	Вызов диалога установки пароля проекта
	Настройка сети	Вызов диалога настройки сетей Ethernet целевой ПЛК
	Сборка проекта в директории сборки	Полная сборка (компиляция и компоновка) текущего проекта в папку build директории проекта
	Загрузить образ на ПЛК	Загрузка образа в ППЗУ ПЛК
	Загрузить и запустить образ на ПЛК	Загрузка образа в ОЗУ ПЛК и запуск программы
	Очистить директорию сборки проекта	Удаление папки build, где был собран проект
	Подключиться к целевому ПЛК	Соединиться с целевым устройством по адресу URI, который указан в настройках проекта (п. 7.4.2)
	Отключиться от целевого устройства	Разрыв соединения с целевым устройством
	Показать код, сгенерированный PLCGenerator	Отобразить в отдельной панели (см. п. 6.5.4) код на языке ST, который был транслирован из кода блоков языков МЭК
	Запуск прикладной программы	Запустить на исполнение собранную прикладную программу на целевом устройстве
	Остановить работающую прикладную программу	Остановить исполнение прикладной программы на целевом устройстве
	Включить/выключить пошаговую отладку	Включить/выключить пошаговую отладку прикладной программы на целевом устройстве
	Продолжить пошаговую отладку до следующей точки останова	Продолжить пошаговую отладку прикладной программы на целевом устройстве до следующей точки останова
	Продолжить пошаговую отладку до следующей строки	Продолжить пошаговую отладку прикладной программы на целевом устройстве до следующей строки

6.3. Отображение структуры проекта

Структура проекта расположена в левой части окна среды разработки и отладки (см. рис. 8) и отображает структуру элементов, из которых состоит проект.

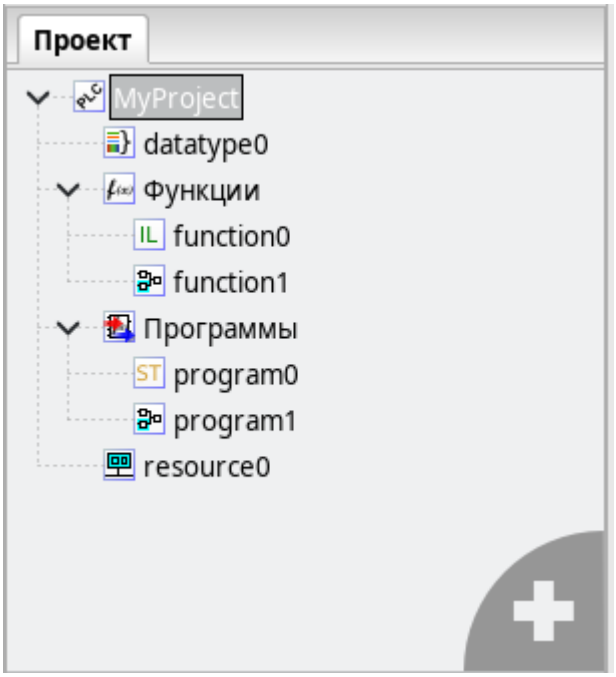


Рисунок 8 – Структура проекта

В роли элементов могут выступать:

– ресурсы;
– программные модули (функции, функциональные блоки и программ) и их составные части;
– типы данных;
– плагины модулей УСО;
– плагины протоколов коммуникации;
– расширения.

Структура проекта позволяет добавлять, удалять элементы. Операции копирования и вставки доступны только для программных модулей.

6.3.1. Добавление элемента в структуру проекта

В правом нижнем углу дерева проекта находится кнопка «+», выделено красным на рис.9

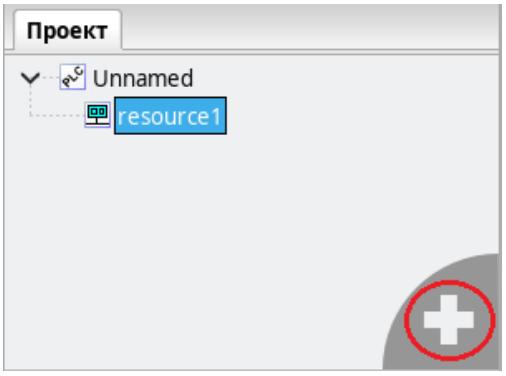


Рисунок 9 – Кнопка добавления элемента в структуру проекта

При нажатии на эту кнопку, появляется меню (см. рис.10) выбора элемента.

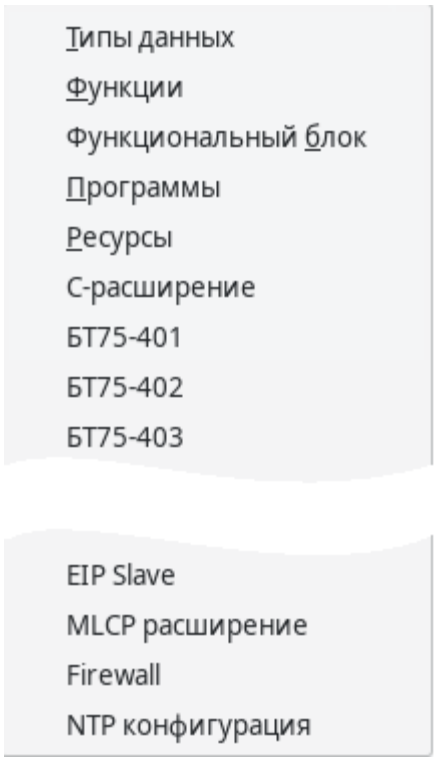


Рисунок 10 – Всплывающее меню добавления элементов проекта на панели проекта

В случае добавления программного модуля, то есть выбора пункта «Функции», «Функциональный блок» или «Программа», появится диалог «Создать новый POU» (см. рис. 11).

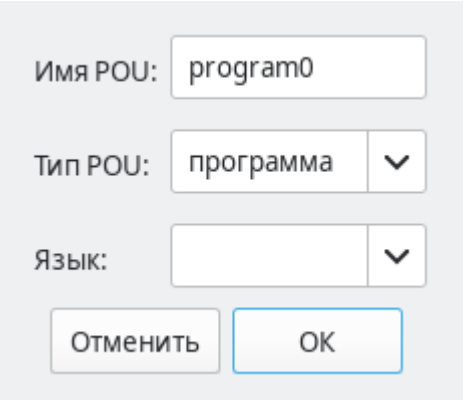


Рисунок 11 – Диалог добавления программного модуля

В данном диалоге три поля:

– «Имя POU»;
– «Тип POU»;
– «Язык».

Имя, присвоенное по умолчанию, может быть заменено на имя, соответствующее назначению данного программного модуля. В зависимости от того, какой программный модуль был выбран во всплывающем меню, в поле «Тип POU» будет подставлено именование данного программного модуля. В поле «Язык» необходимо выбрать из списка (см. рис. 12) один из языков стандарта МЭК 61131-3-2016 (IL, ST, LD, FBD, SFC), на котором будут реализованы алгоритмы и логика работы данного добавляемого программного модуля.

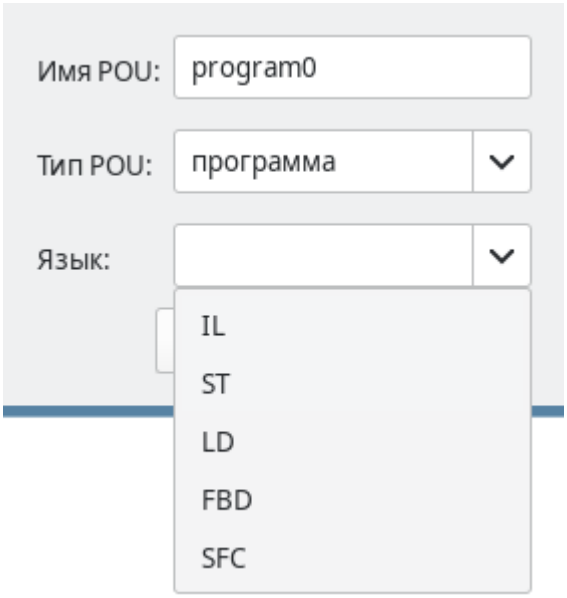


Рисунок 12 – Выбор языка для программного модуля

В случае выбора добавления типа данных, появится новый элемент в узле дерева проекта (см. рис. 13).

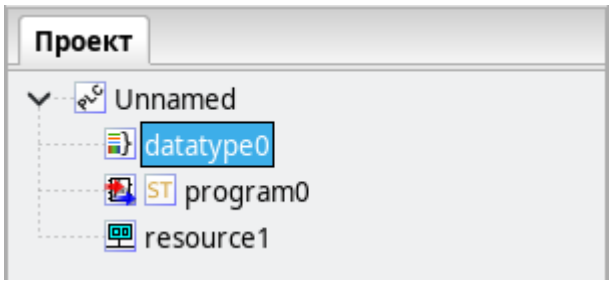


Рисунок 13 – Новый тип данных в дереве проекта

Добавление нескольких элементов одного типа, например: программ, функций, функциональных блоков приводит к их группировке в дереве проекта.

Добавление нового элемента или выбор существующего в дереве проекта приводит к появлению панели редактирования и настроек соответствующего элемента:

– панели настроек проекта;
– панели текстового редактора языков ST и IL;
– панели графического редактора диаграмм языков FBD, SFC, LD;
– панели настроек ресурса;
– панели редактирование типа данных;
– панели настроек плагинов модулей УСО.

Каждая перечисленная панель редактирования будет рассмотрена ниже.

6.3.2. Удаление элемента в структуре проекта

Удаление осуществляется наведением на определённый элемент в структуре проекта и нажатием на него правой клавишей мыши, а далее в появившемся меню выбирается пункт «Удалить» (см. рис. 14).

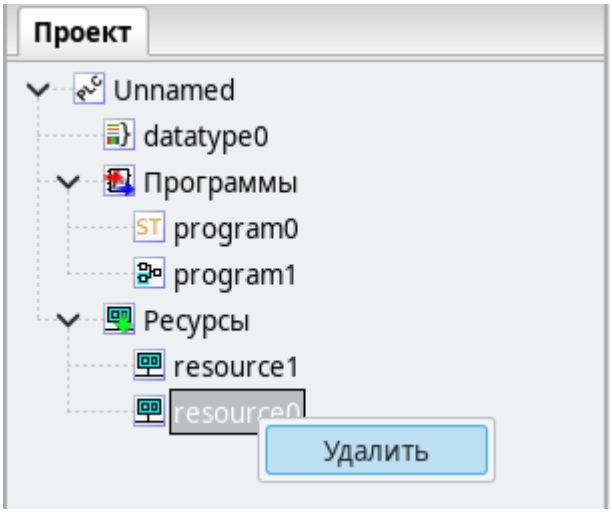


Рисунок 14 – Удаление элемента

6.3.3. Переименование, копирование и вставка программных модулей

Структура проекта позволяет выполнять операции переименования, копирования и вставки для программных модулей. Копирование или переименование осуществляются с помощью нажатия правой клавиши мыши на элемент (см. рис. 15), соответствующий программному модулю в структуре проекта, и выбором нужного пункта в открывшемся меню.

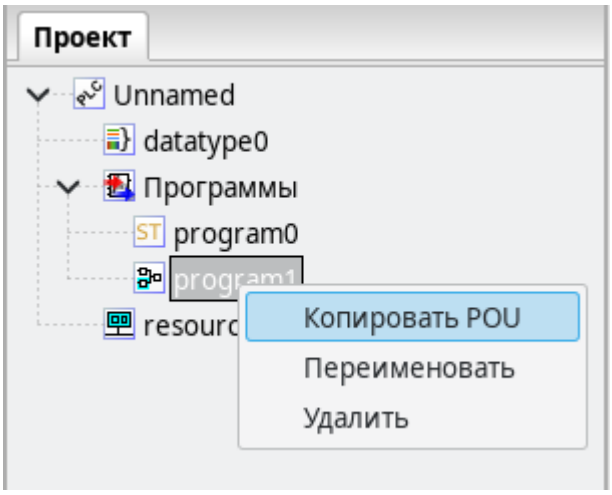


Рисунок 15 – Пункт «Копировать программный модуль»

Вставка программного модуля осуществляется в меню (нажатие правой клавишей мыши по данному элементу) корневого элемента структуры проекта, соответствующего проекту (см. рис. 16):

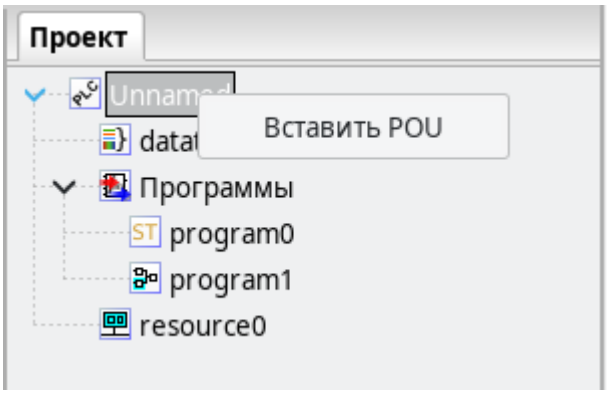


Рисунок 16 – Пункт «Вставка программного модуля»

Другим способом выполнения вышеописанной операции является вызов меню для элемента группировки программных модулей одного типа (см. рис. 17).

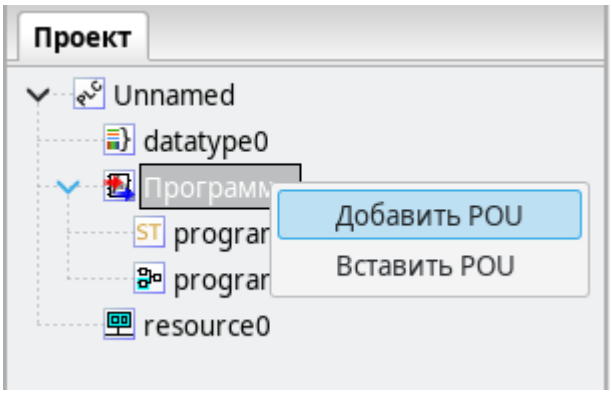
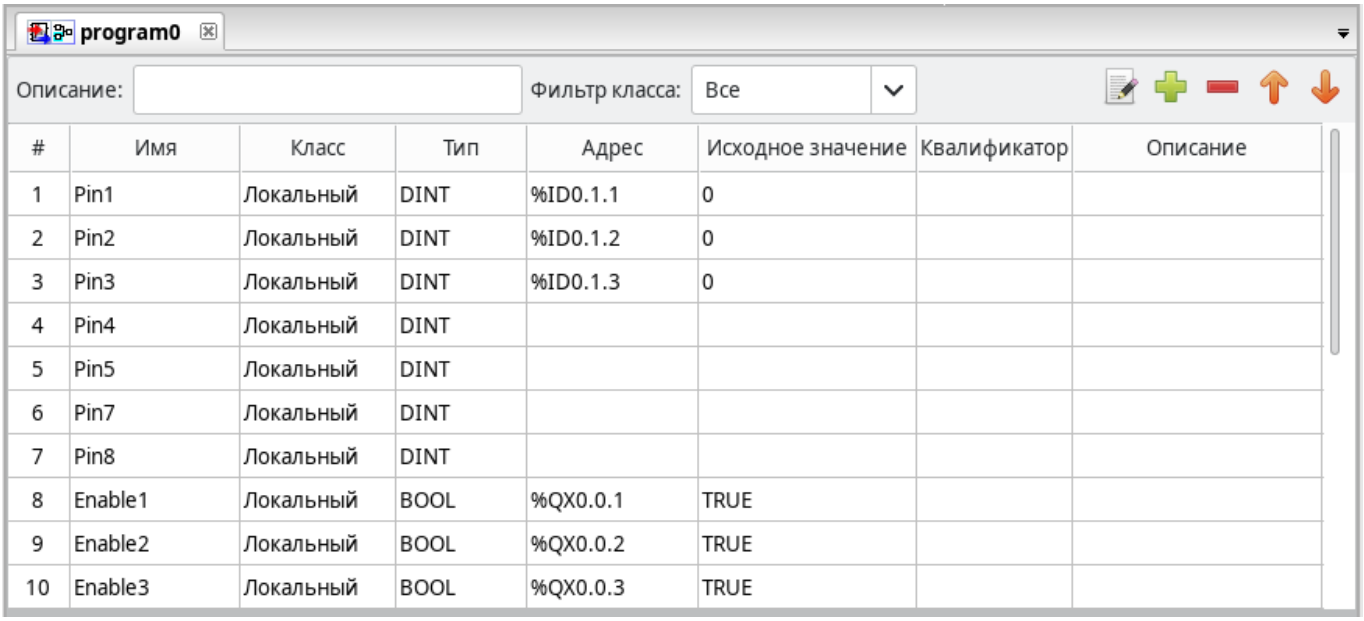


Рисунок 17 – Пункт «Вставка программного модуля»

Далее приводится описание средств для работы с переменными и константами, которые присутствуют при редактировании проекта, ресурса и программных модулей (функции, функционального блока, программы).

6.4. Средства для работы с переменными и константами проекта

Панель списка переменных и констант содержит таблицу переменных и констант, соответствующих выбранному программному модулю, ресурсу или проекту, рис. 18.



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Описание
1	Pin1	Локальный	DINT	%ID0.1.1	0		
2	Pin2	Локальный	DINT	%ID0.1.2	0		
3	Pin3	Локальный	DINT	%ID0.1.3	0		
4	Pin4	Локальный	DINT				
5	Pin5	Локальный	DINT				
6	Pin7	Локальный	DINT				
7	Pin8	Локальный	DINT				
8	Enable1	Локальный	BOOL	%QX0.0.1	TRUE		
9	Enable2	Локальный	BOOL	%QX0.0.2	TRUE		
10	Enable3	Локальный	BOOL	%QX0.0.3	TRUE		

Рисунок 18 – Панель переменных и констант

Каждая переменная имеет следующие параметры:

– имя, представляющее собой уникальный идентификатор переменной в пределах её области видимости и действия;
– класс: «Глобальная», «Входная», «Выходная», «Входная/Выходная», «Локальная», «Внешняя», «Временная»;
– тип, определяющий тип переменной и может принадлежать базовому типу (в соответствии со стандартом IEC 61131-3), пользовательскому типу (псевдониму и поддиапазону существующего типа, перечислению, массиву, структуре) или типу функционального блока (стандартному или пользовательскому);
– адрес – идентификатор, необходимый для связывания данной переменной с переменного плагина модуля УСО (подробнее в п. 7.6);
– исходное значение – инициализация переменной некоторым начальным значением;
– квалификатор – задание константности, реманентности (сохранение её значения в энергонезависимой памяти) и неремanентности переменной, в текущей версия сервиса исполнения для ПЛК не поддерживает хранение переменных в энергонезависимой памяти;
– описание – комментарий к назначению данной переменной или константы.

Первый символ имени переменной или константы должен быть буквой, или символом подчёркивания, далее могут следовать цифры, буквы латинского алфавита и символы подчёркивания. Набор возможных вариантов классов переменных зависит от типа элемента проекта, редактирования которого осуществляется.

Двойной клик на поле «Адрес» вызывает появление кнопки «…», выделенной красным на рис. 19.

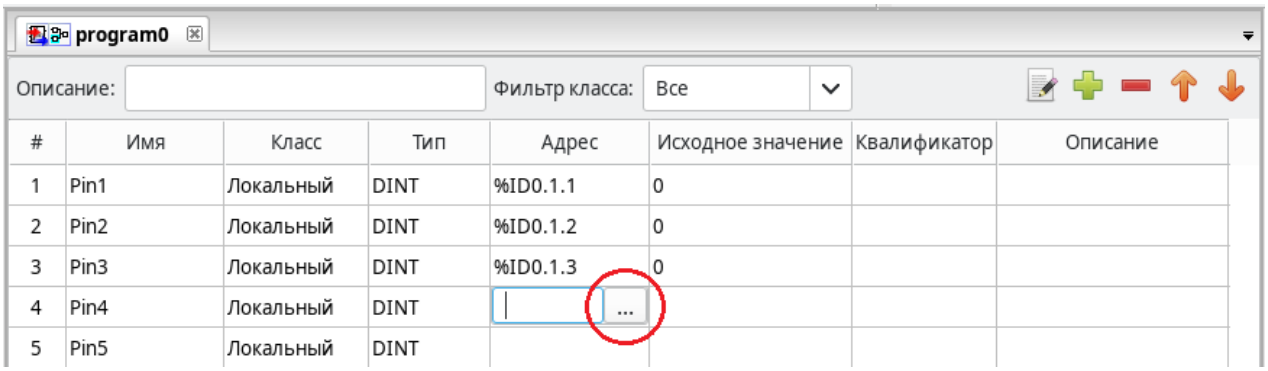


Рисунок 19 – Поле «Адрес» панели переменных и констант

Нажатие на данную кнопку приводит к появлению диалога «Просмотр адресов», рис. 20. Выводится список переменных модулей УСО, которые могут быть связанны с переменой в панели переменных и констант. При выборе в данном диалоге определённой переменной и нажатии клавиши «ОК» в поле «Адрес» будет добавлен адрес переменной внешнего модуля УСО.

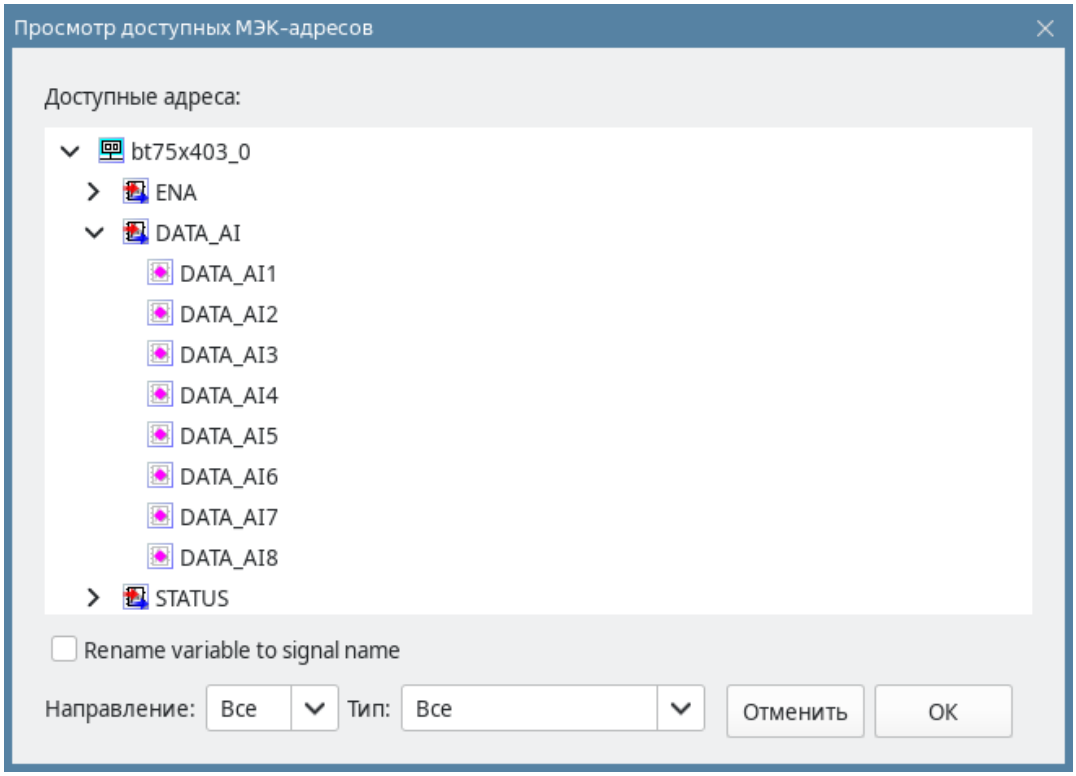


Рисунок 20 – Диалог «Просмотр адресов», вызываемый из поля «Адрес»

Поле опции позволяет определить переменную как константу. Соответственно, если компилятор обнаружит в коде фрагмент, в котором происходит изменение этой переменной – будет выведена ошибка компиляции «Assignment to CONSTANT variables is not be allowed» в «Отладочной консоли». Квалификатор «Константа» не может быть использован в объявлении функциональных блоков. Переключение отображения переменных в виде таблицы в отображение в виде текста и обратно, добавление, удаление и перемещение переменных происходит с помощью специальных кнопок на панели переменных и констант. Для удобства пользователь может задать отображение переменных в виде текста с помощью кнопки «Переключить отображение таблица/текст». Важно отметить, что изменения будут применены только при возврате в просмотр в виде таблицы или ручном сохранении переменных с помощью кнопки «Сохранить переменные», которая отображается только при включённом режиме отображения переменных в виде текста.

Панель переменных и констант предоставляет возможность фильтровать отображаемые переменные по их конкретным классам («Вход», «Выход», «Вход/Выход», «Внешний», «Локальный», «Временный») или сгруппированным классам («Интерфейс» и «Переменные»). Данная операция выполняется с помощью функции «Фильтр класса», рис. 21.

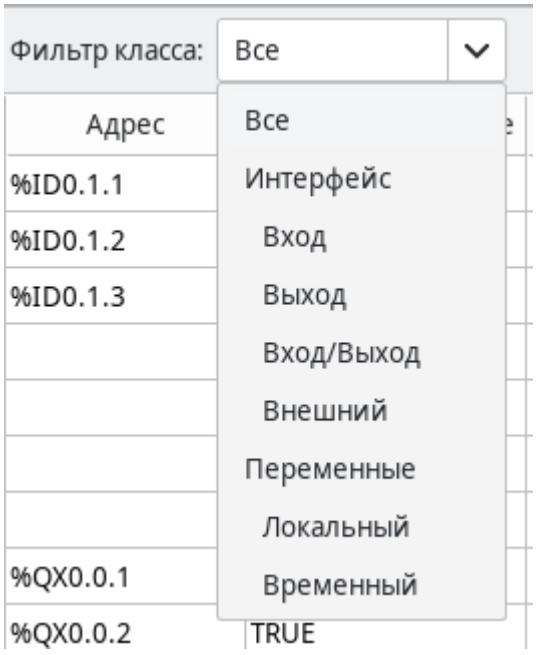


Рисунок 21 – Фильтрация отображения переменных в панели переменных и констант

В нижней части рабочей области может располагаться панель с настройками проекта, панель ресурсов, редакторы текстовых и графических языков стандарта МЭК 61131-3-2016, редакторы для настройки параметров внешних плагинов, визуализация процесса отладки графических языков стандарта МЭК 61131-3-2016 и другие элементы.

Описание кнопок редактирования переменных и констант приведено в табл. 3

Таблица 3 – Кнопки редактирования переменных на панели переменных и констант

Кнопка	Наименование кнопки	Функции кнопки
	Сохранить переменные	Сохраняет отредактированные переменные при включённом режиме отображения переменных в виде текста.
	Переключить отображение таблица/текст	Переключает отображение переменных в виде таблицы в отображение в виде текста и обратно.
	Добавить переменную	Добавить новую переменную в панель переменных и констант со значениями по умолчанию
	Удалить переменную	Удалить выделенную переменную или константу
	Переместить переменную вверх	Перемещение переменной в таблице переменных и констант вверх на одну позицию
	Переместить переменную вниз	Перемещение переменной в таблице переменных и констант вниз на одну позицию

6.5. Доступ к настройкам и файлам проекта

В панель настройки проекта входят страницы переменных и констант (рис. 22), данных о проекте (рис. 232323), а также настроек сборки проекта (рис.24 – рис.25).

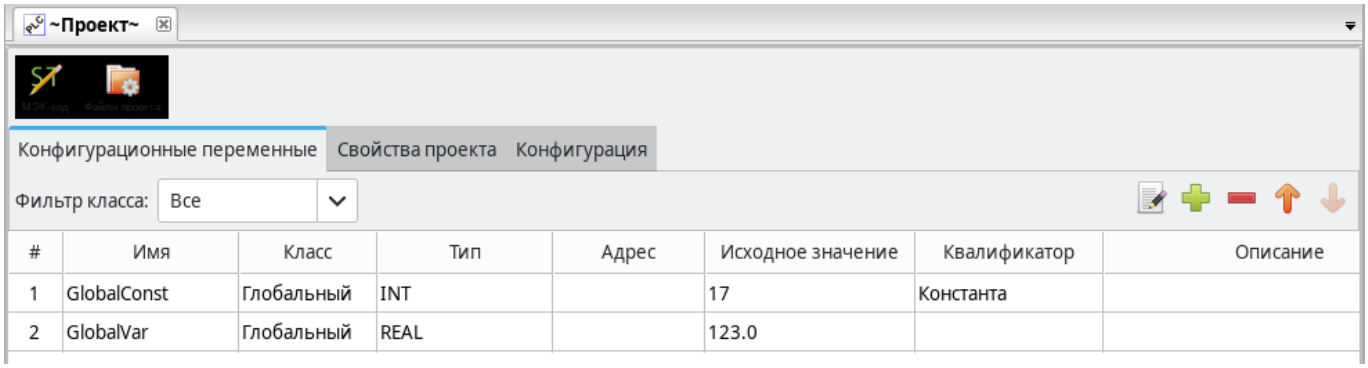


Рисунок 22 – Страница переменных и констант панели настроек проекта

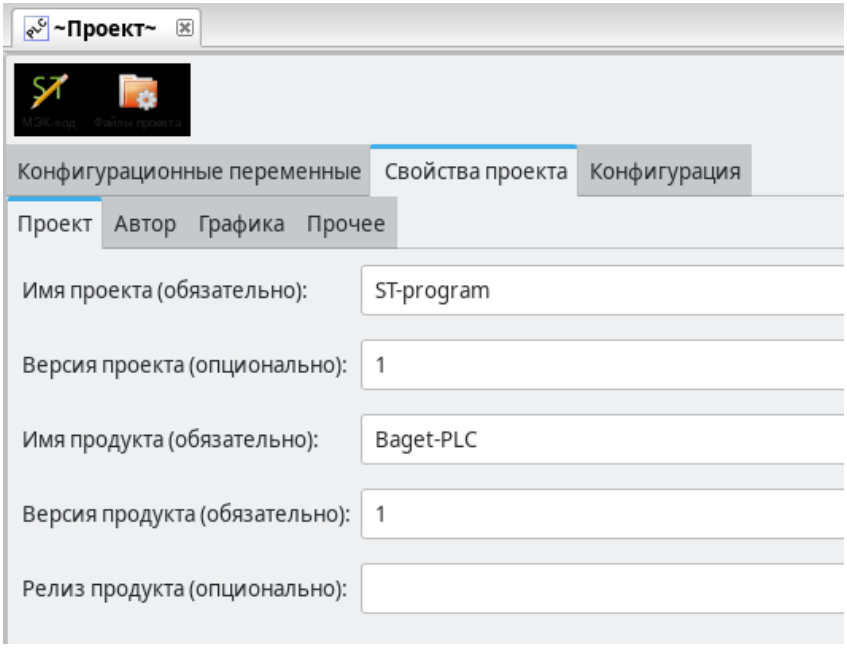


Рисунок. 23 – Страница данных о проекте

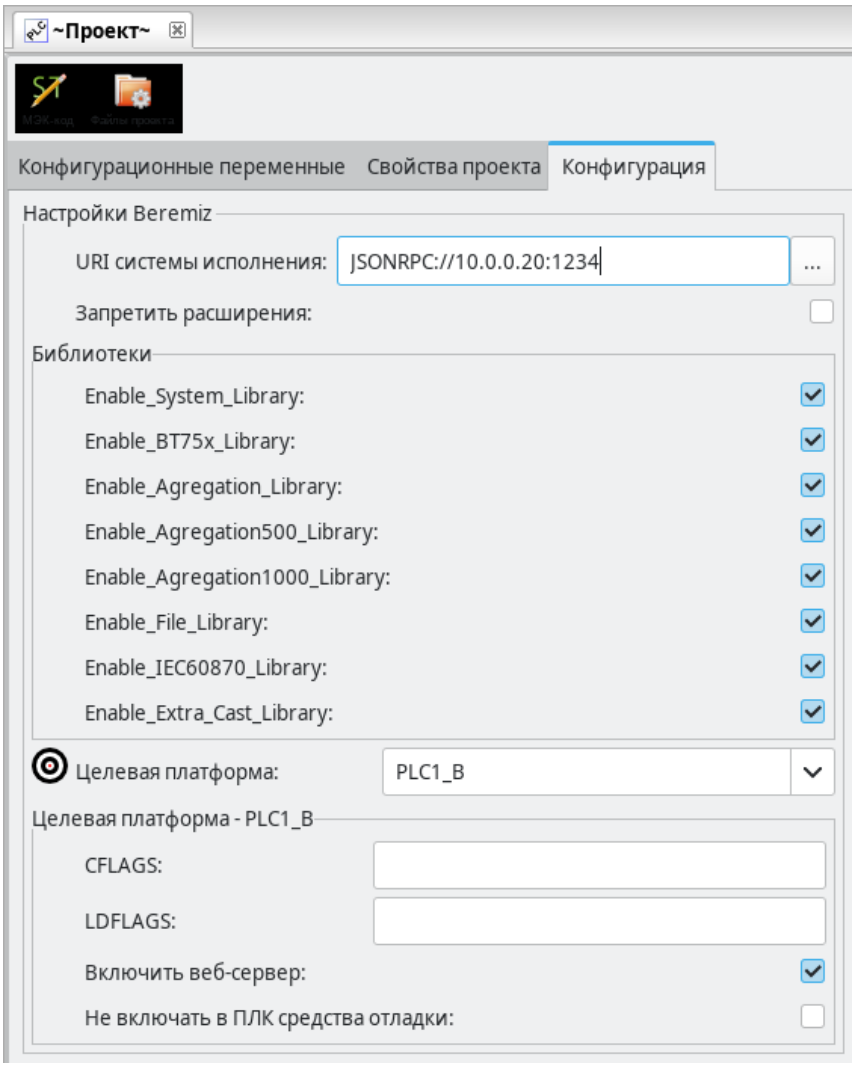


Рисунок 24 – Страница настроек сборки проекта для ПЛК-1В

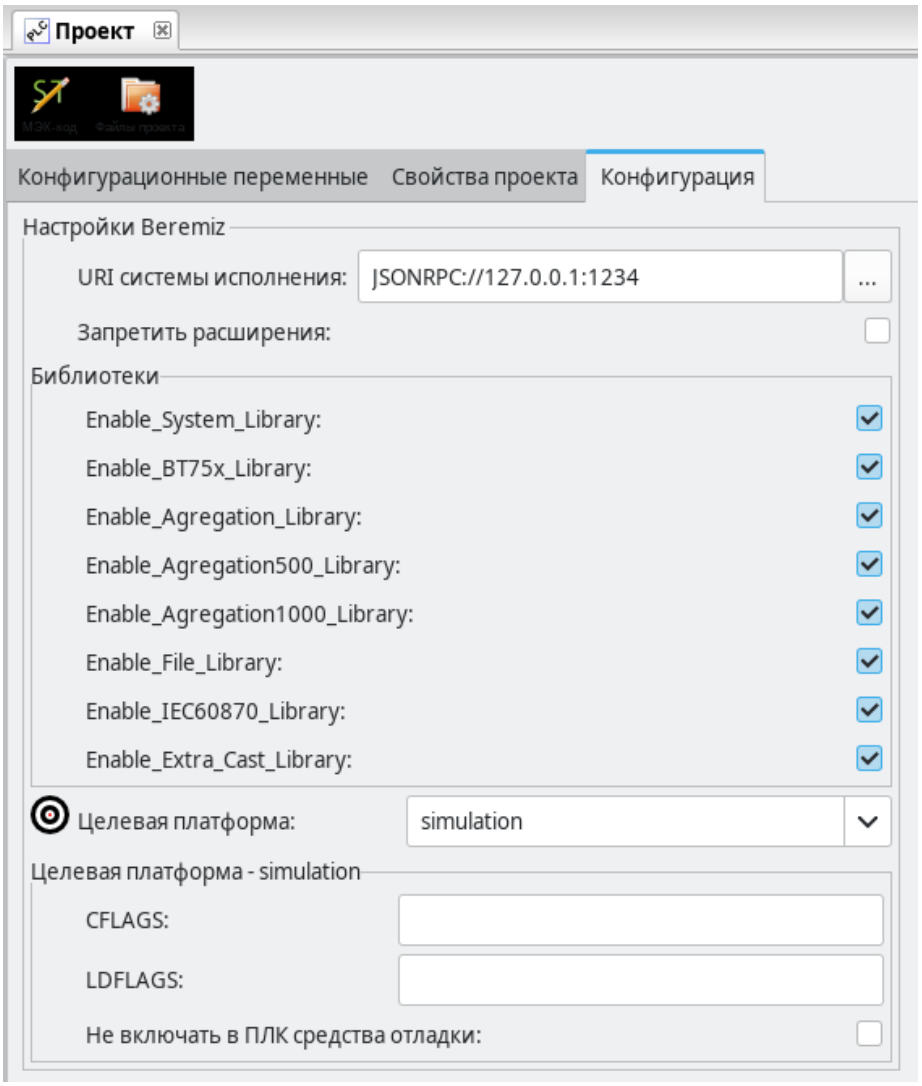


Рисунок 25 – Страница настроек сборки проекта для СИ ИЭВМ

Страница переменных и констант панели настроек проекта, позволяющая задать глобальные переменные на уровне проекта, располагается вверху. Ниже две области: слева настройки параметров сборки данного проекта, а справа поля с различной информацией о проекте.

Также в настройках сборки проекта имеются две кнопки, описание которых приведено в табл. 4.

Таблица 4 – Кнопки в панели настройки сборки проекта

Кнопка	Наименование кнопки	Функции кнопки
	Промежуточный МЭК-код	Вызов панели отображения промежуточного кода, для вывода кода, из которого генерируется ST код всего проекта
	Файлы проекта	Вызов «Панели файлов проекта», в которой можно выбрать файлы необходимые для передачи на целевое устройство вместе с исполняемым файлом (см. п. 6.5.5)

6.5.1. Страница настройки сборки проекта панели настроек проекта для ОС РВ Багет

Настройки сборки проекта (рис.24) позволяют задать следующие параметры:

– «URI системы исполнения» – унифицированный идентификатор ресурса (под ресурсом понимается целевое устройство). Данный адрес необходим для режима отладки, о котором подробно рассказано в п. 7.4.3.

  Пример URI системы исполнения:

  – «`JSONRPC://10.0.0.20:1234`» – для целевых платформ «`PLC1_A`» и «`PLC1_B`»;
  – «`JSONRPC://127.0.0.1:1234` – для СС ИЭВМ (simulation) ;

  Для ИЭВМ рекомендуется задать IP адрес: `10.0.0.30`.

– «Запретить расширения» – установка данного флага позволяет не учитывать при сборке проекта внешние плагины (см. п. 7.4.3);
– «Целевая платформа» – выбор из списка компилятора для архитектуры целевого устройства:
  – «`PLC1_A`» – для «Багет-ПЛК1» на основе процессорного модуля БТ75-201А с микропроцессором 1890ВМ108;
  – «`PLC1_B`» – для «Багет-ПЛК1» на основе процессорного модуля БТ75-201Б с микропроцессором К5500ВК018;
  – «`simulation`» – для СС ИЭВМ;
– «CFLAGS» – указание флагов Си компилятора, в том числе пути к заголовочным файлам библиотек и компонент, если они (пути) не заданы в соответствующие переменные окружения;
– «LDFLAGS» – указание флагов компоновщика, в том числе пути к файлам библиотек и компонент, если они (пути) не заданы в соответствующие переменные окружения;
– Включить веб-сервер: технологические передачи осуществляются через веб-сервер. Если веб-сервер отключен, часть кнопок недоступна, в том числе «Подключится к целевому ПЛК» и кнопки, связанные с отладкой. Функция реализована для управления ПЛК только физическими переключателями.
– «Не включать в ПЛК средства отладки» – отключает в образе ОС РВ Багет (СПО) функции отладки:
  – если средства отладки отключены, то запуск образа ОС РВ Багет (СПО) выполняется автоматически (по включению питания, сбросу) (см. п.8.2.4) или по нажатию кнопки: «Загрузить и запустить образ на ПЛК» (см. п.8.2.5). Контроль запуска СПО возможен за счет вывода программой сообщений в терминал.
  – если средства отладки включены, то запуск образа ОС РВ Багет (СПО) осуществляется по нажатию кнопок: «Загрузить и запустить образ на ПЛК», «Подключиться к целевому ПЛК», «Запустить ПЛК» (см. п.8.2.5 - 8.2.7).

6.5.2. Страница настройки сборки проекта панели настроек проекта для СС ИЭВМ

Настройки сборки проекта для СС ИЭВМ (см. рис. 25) позволяют задать следующие параметры:

6.5.3. Страница отображения данных о проекте панели настроек проекта

Вкладка «Проект» (см. рис. 26) позволяет задать: имя проекта, версию проекта, имя продукта, версию продукта и релиз продукта.

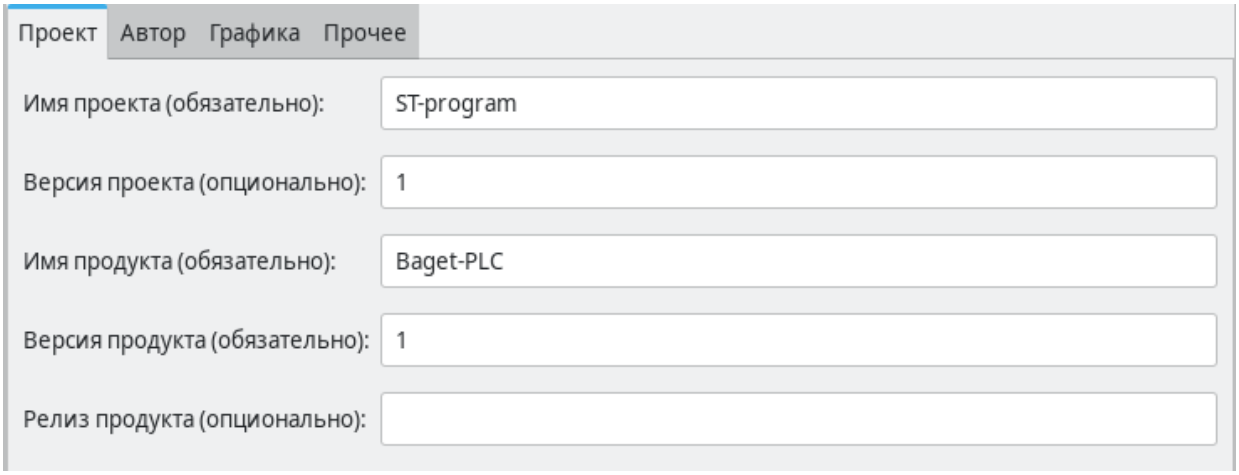


Рисунок 26 – Вкладка с информацией о проекте

Вкладка «Автор» (см. рис. 27) позволяет указать компанию, адрес сайта компании, имя автора, название организации.

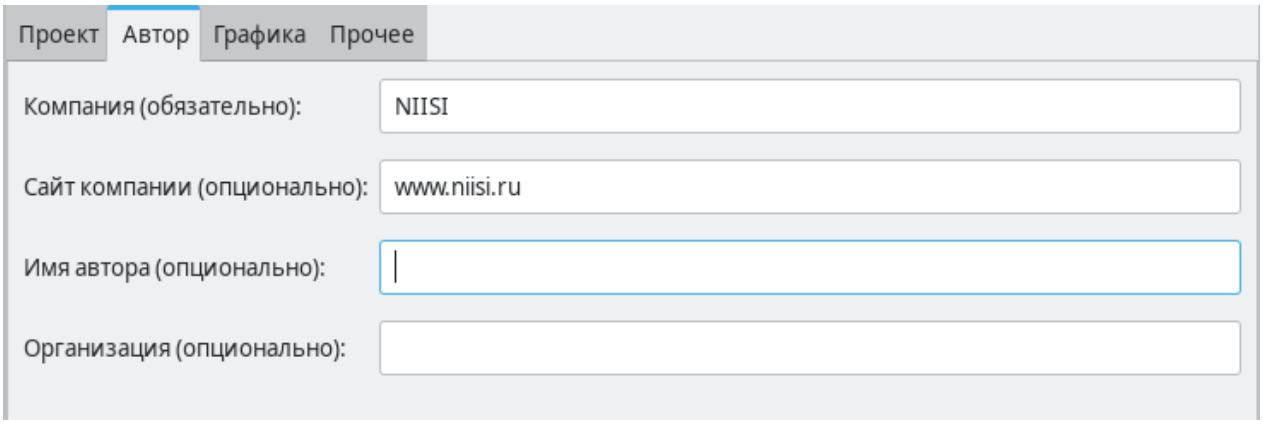


Рисунок 27 – Вкладка с информацией об авторе проекта

Вкладка «Графика» (см. рис. 28) позволяет задать размеры страницы и разрешение сетки для редакторов диаграмм графических языков FBD, LD и SFC.

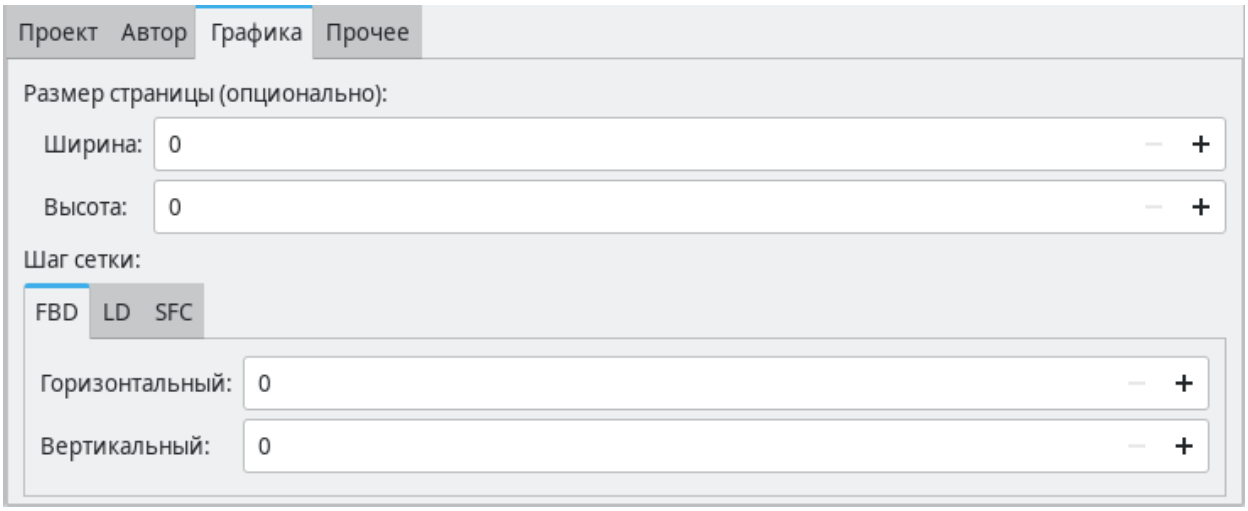


Рисунок 28 – Вкладка настроек параметров редакторов графических языков

Вкладка «Прочее», изображённая на рис. 29, позволяет выбрать язык интерфейса для среды разработки и отладки программ и указать дополнительное текстовое описание для проекта.

При запуске среды разработки и отладки языком по умолчанию является язык, соответствующий текущей локали операционной системы, если файл для данной локали присутствует. В случае отсутствия данных файлов, устанавливается английская локаль, которая доступна всегда. Файлы доступных локалей располагаются в каталоге beremiz/locale.

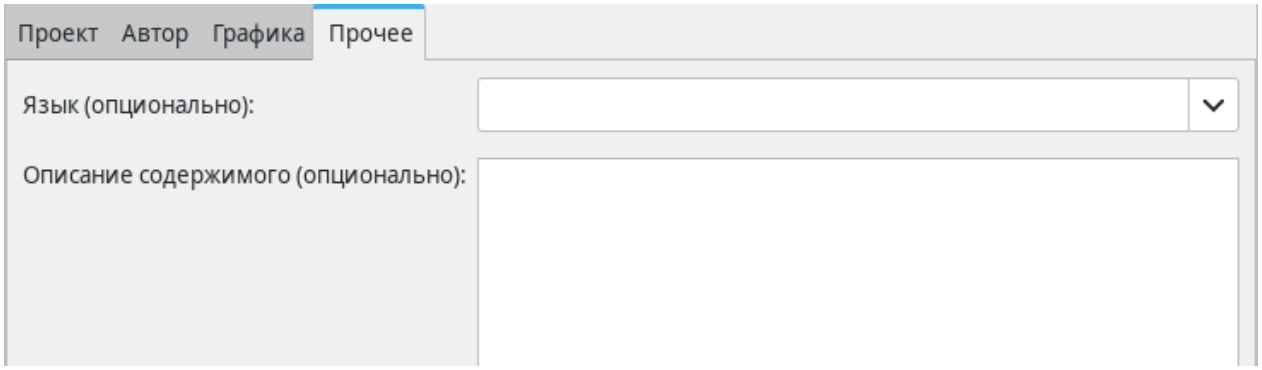


Рисунок 29 – Вкладка прочими настройками

Далее рассмотрены панели отображения промежуточного кода на языке ST и панель файлов проекта, вызываемые с помощью кнопок, описанных в таблице 4 и располагающихся на панели настроек проекта.

6.5.4. Панель отображения промежуточного кода

Данная панель (см. рис. 30) представляет собой текстовый редактор, отображающий с подсветкой синтаксиса и нумерацией строк код на языке ST, доступный только для чтения, без возможности редактирования.

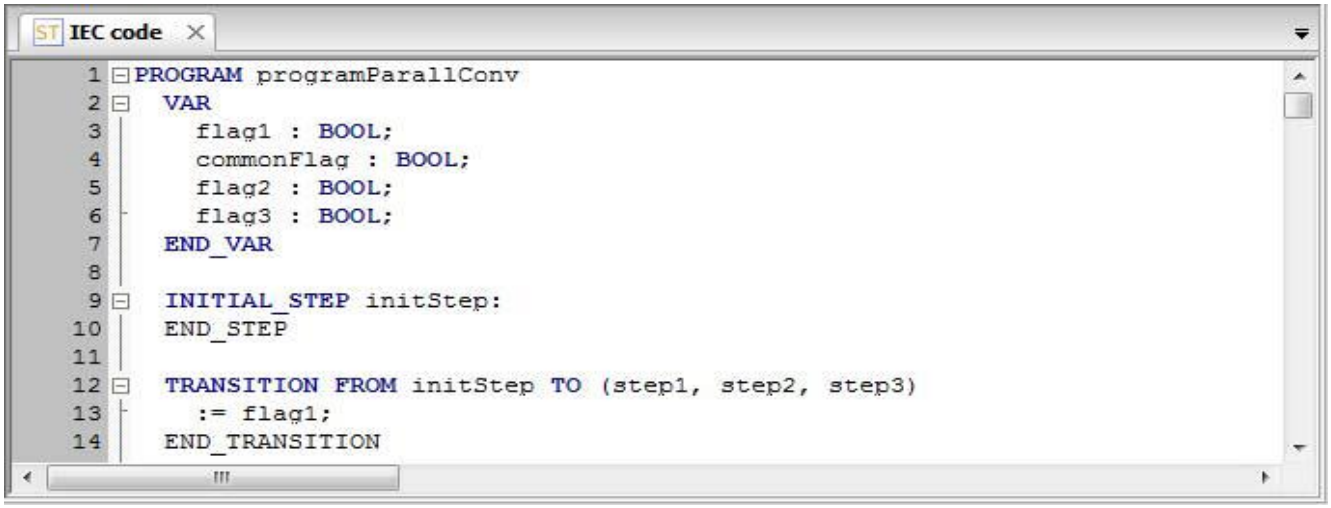


Рисунок 30 – Панель отображения промежуточного кода на языке ST

Открытие данной панели доступно после сборки проекта с помощью соответствующей кнопки (см. таблицу 4).

6.5.5. Панель файлов проекта

Панель файлов проекта (см. рис. 31) содержит встроенный проводник файлов (справа), в котором файлы могут быть выделены и перенесены в левую часть.

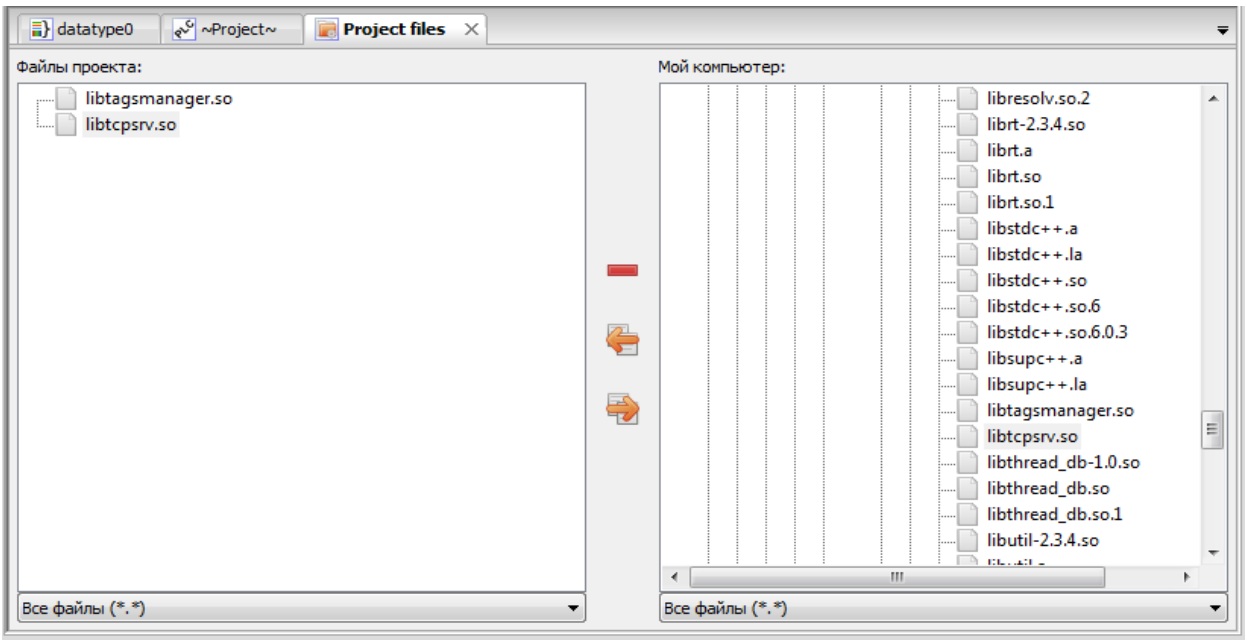


Рисунок 31 – Панель файлов проекта

Все манипуляции с файлами осуществляются с помощью кнопок, расположенных в середине данной панели. Их описание приведено в табл. 5.

Таблица 5 – Кнопки в панели файлов проекта

Кнопка	Наименование кнопки	Функции кнопки
	Удалить файл списка файлов проекта	Удаление выделенного файла из левого списка добавленных файлов в проект
	Добавить файл в проект	Добавить выделенный файл из проводника файлов в проект
	Добавить файл в проводник	Добавить в текущую папку проводника файлов слева выделенный файл в списке файлов проекта

Как правило, этими дополнительными файлами проекта являются сторонние библиотеки, необходимые для корректной работы плагинов модулей УСО.

6.6. Отображение промежуточного кода

Окно отображения IEC code позволяет просматривать промежуточной код, сгенерированный PLCGenerator. Кнопка выделена красным на рис. 32.

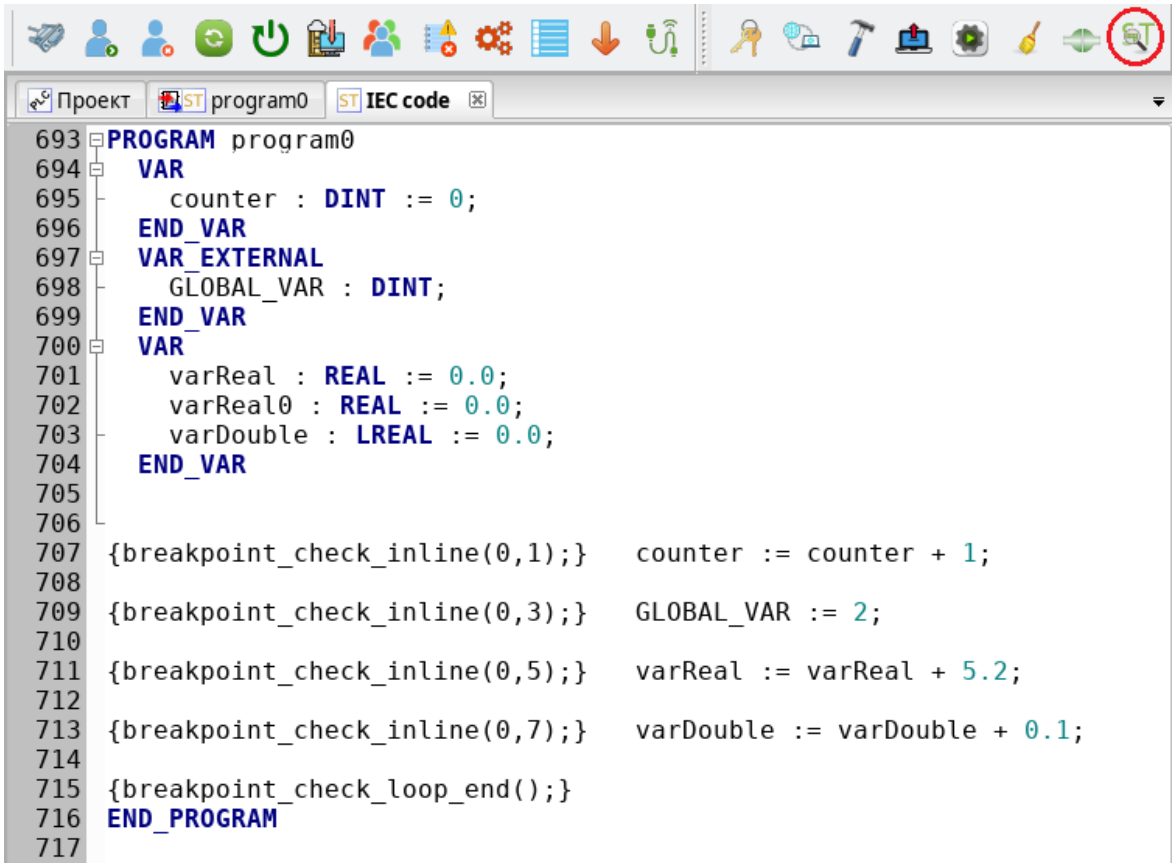


Рисунок 32 – Окно отображения промежуточного кода

6.7. Окно текстового редактора языка Си

Текстовый редактор языка Си (см. рис. 33) позволяет создавать и редактировать алгоритмы и логику выполнения программных модулей на языке Си.

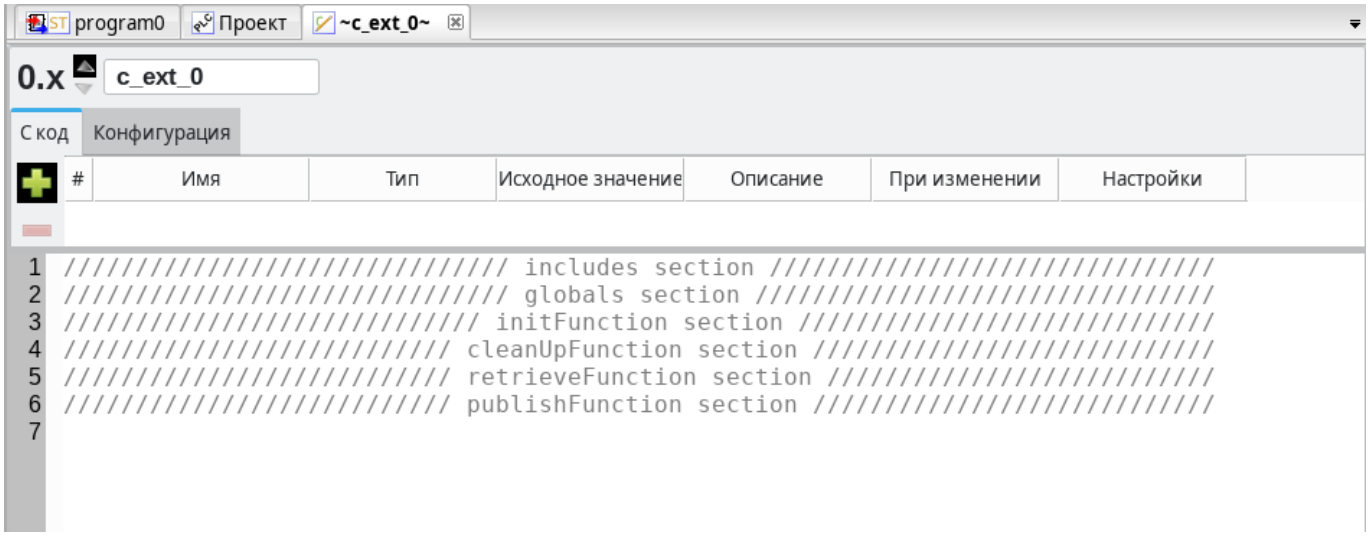


Рисунок 33 – Окно текстового редактора языка Си

При нажатии на пункт меню «Си-расширение» в дереве проекта, его параметры и содержимое отобразятся в области редактирования. Имя расширения можно изменить, используя поле редактирования в верхней части окна.

Создавать локальные переменные для Си-расширения возможно таким же образом, как и локальные переменные для других модулей. Глобальные переменные также доступны, но имеют ограничения по именованию. Поскольку языки программирования МЭК нечувствительны к регистру, транслятор ТР ПЛК приводит все глобальные переменные в верхний регистр, даже если в исходном коде ST они указаны в нижнем регистре. По этой причине рекомендуется глобальные переменные, которые предполагается использовать в Си-расширении, изначально объявлять с именами в верхнем регистре.

В окне редактора можно видеть следующие секции:

– «includes section»;
– «globals section»;
– «initFunction section»;
– «cleanUpFunction section»;
– «retrieveFunction section»;
– «publishFunction section».

Удалять заголовки разделов нельзя. Важно поместить Си-код в правильный раздел.

Раздел «includes section» должен содержать заголовочные файлы, необходимые для компиляции функций Си, объявленных в Си-расширении.

Раздел «globals section» должен содержать тела функций Си, которые могут быть вызваны из функциональных блоков.

Код из раздела «initFunction section» вызывается единожды перед запуском программы.

Код из раздела «cleanUpFunction section» вызывается единожды в ходе завершения программы.

Раздел «retrieveFunction section» вызывается перед каждым циклом и предназначен для работы с переменными, хранимыми в энергонезависимой памяти (в текущей версии СИ ПЛК такие переменные не поддерживаются).

Раздел «publishFunction section» вызывается в рамках каждого цикла работы СПО.

Комплексный тест ПП ПО ПЛК Багет (РСКЮ.20514-01) содержит рабочий пример использования Си-расширения в проекте, в том числе приведены примеры работы с глобальными переменными проекта из Си-расширения.

6.8. Окно текстовых редакторов языков ST и IL

Текстовый редактор языков ST и IL, рис. 34 позволяет создавать и редактировать алгоритмы и логику выполнения программных модулей на языках ST и IL.

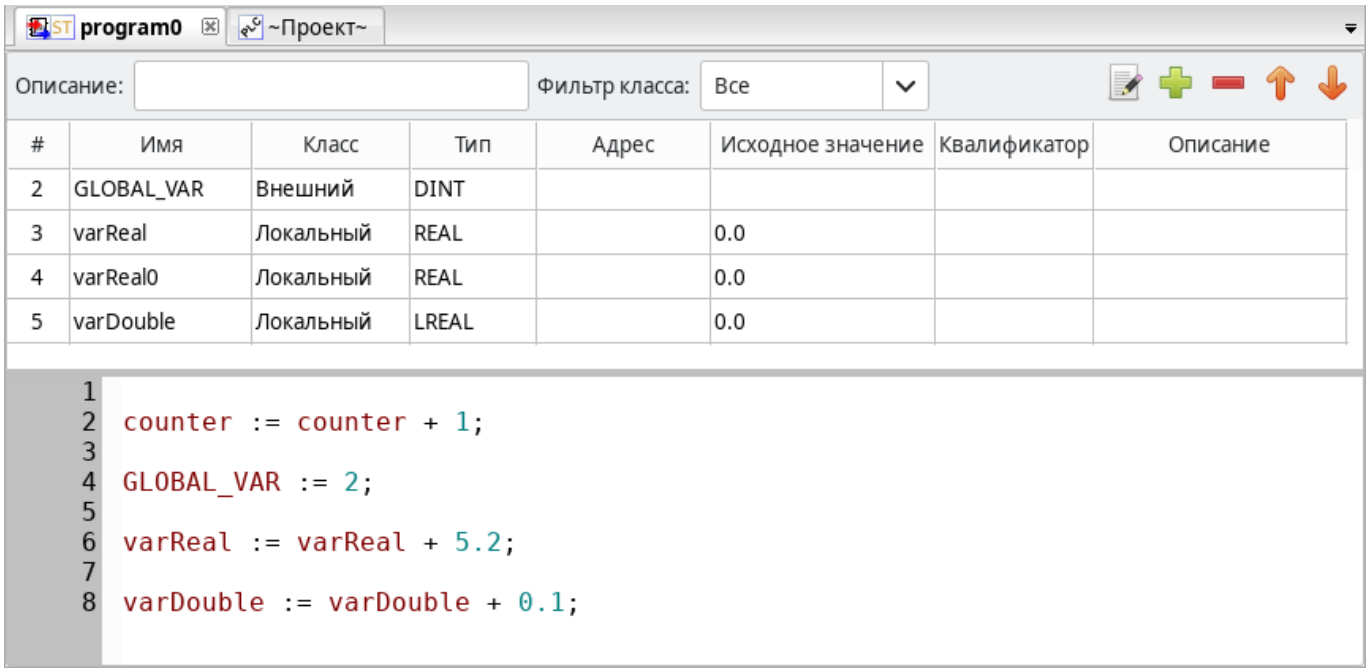


Рисунок 34 – Редактор языков ST и IL

Редактор языков ST и IL обеспечивает следующие возможности:

– подсветку синтаксиса кода, написанного пользователем, т.е. выделения особыми параметрами шрифта ключевых слов данных языков;
– нумерации строк, что может быть полезным при возникновении ошибок в программе, так как транслятор кода ST в C выдаёт номер строки, в которой найдена ошибка;
– сворачивание кода структурных элементов языка: определения функции, определение типа и так далее.

Увеличение или уменьшение размера шрифт выполняется с помощью Ctrl + <колёсико мыши>.

Описание синтаксиса, основных конструкций языков ST и IL описаны в ГОСТ Р МЭК 61131-3-2016.

6.9. Окно графических редакторов языков FBD, SFC и LD

Данные редакторы позволяют создавать и редактировать алгоритмы логику выполнения программных модулей, написанных на языках FBD, SFC и LD.

6.9.1. Редактор языка FBD

Основными элементами языка FBD являются: переменные, функциональные блоки и соединения. При редактировании FBD диаграммы, в панели инструментов появляется следующая панель, рис. 35.

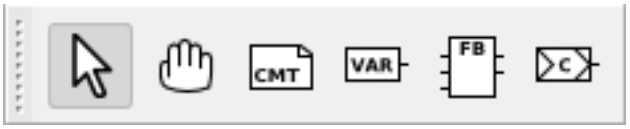


Рисунок 35 – Панель редактирования FBD диаграмм

С помощью данной панели можно добавить все элементы языка FBD. Назначение каждой кнопки описано в табл. 6.

Таблица 6 – Кнопки панели редактирования FBD диаграммы

Кнопка	Наименование кнопки	Функции кнопки
	Выделение объектов на диаграмме	Перевод указателя мыши в состояние, при котором можно осуществлять выделение объектов редакторе одного из графических языков
	Перемещение диаграммы	Перевод указателя мыши в состояние, при котором можно изменять размеры редактора одного из графических языков, с помощью его перемещения
	Создать новый комментарий	Вызов диалога создания комментария
	Добавить переменную	Вызов диалога добавления переменной
	Добавить функциональный блок	Вызов диалога добавления функционального блока
	Добавить соединение	Вызов диалога добавления соединения

Для этого необходимо указателем мыши выбрать необходимую кнопку и нажать на свободное место в области редактирования FBD диаграммы. В зависимости от выбранного элемента появятся определённые диалоги добавления данного элемента.

Аналогичные действия можно выполнить с помощью всплывающего меню в области редактирования FBD диаграмм. Вызов данного меню происходит нажатием

правой клавишей мыши и выбором пункта «Добавить», в котором будет: «Блок», «Переменная», «Подключение», «Комментарий», рис. 36.

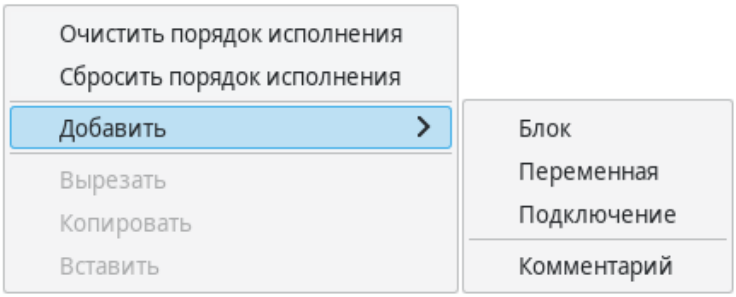


Рисунок 36 – Всплывающее меню редактора языка FBD

Далее рассмотрено добавление каждого элемента в отдельности.

6.9.1.1. Добавление функционального блока

При добавлении функционального блока одним из описанных выше способов, появится диалог «Свойства блока», рис. 37.

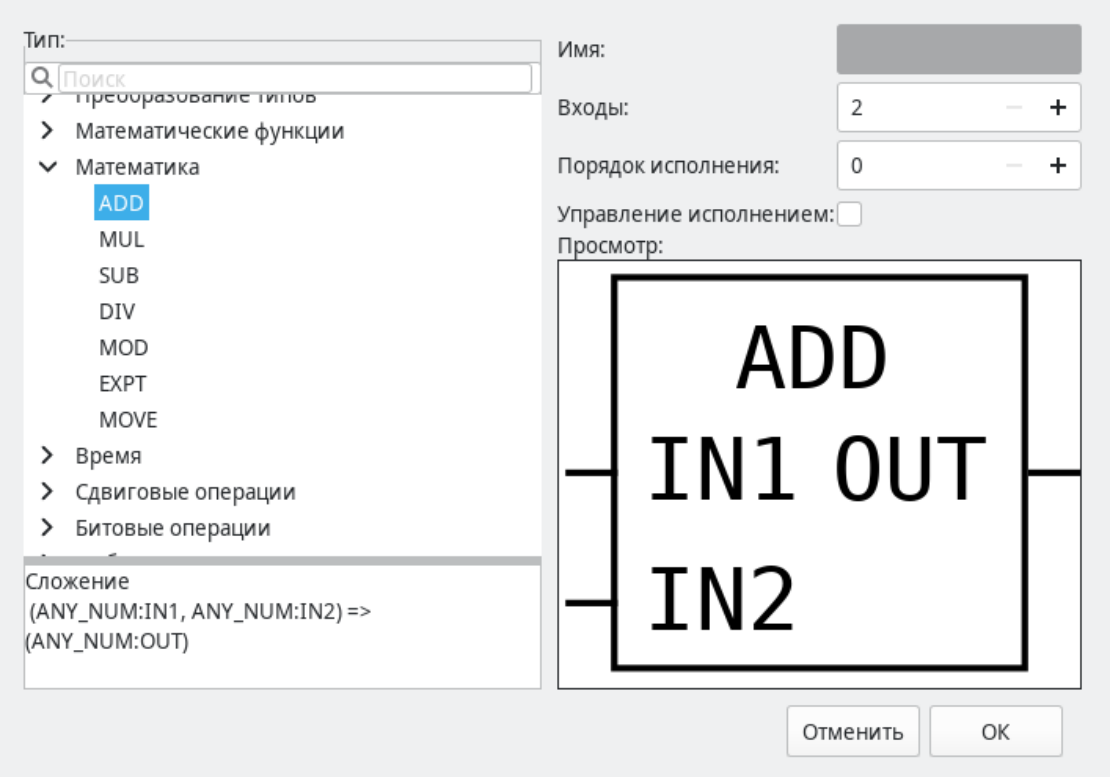


Рисунок 37 – Свойства функционального блока

В диалоге «Свойства блока» приведено краткое описание функционального блока и предоставлена возможность задать некоторые свойства (имя, количество входов, порядок выполнения и так далее).

Элемент выбора (checkbox) «Управление выполнением» добавляет в функциональный блок дополнительные параметры «Enable»: EN и ENO.

Добавление (путем копирования существующего блока), удаление и переименование функционального блока осуществляется при помощи команд меню «Редактирование» в главном меню или с помощью всплывающего меню диаграммы (см. рис. 36).

Также, функциональный блок может быть добавлен из «Панели библиотеки функций и функциональных блоков» (см. п. 6.13), перетаскиванием мыши (Drag&Drop) выбранного блока на панель редактирования диаграммы FBD.

6.9.1.2. Добавление переменной

Переменные добавляются из панели переменных и констант с помощью перетаскивания (Drag&Drop) левой клавишей мыши за область, выделенную красным цветом на рис. 38, в область редактирования FBD диаграмм.

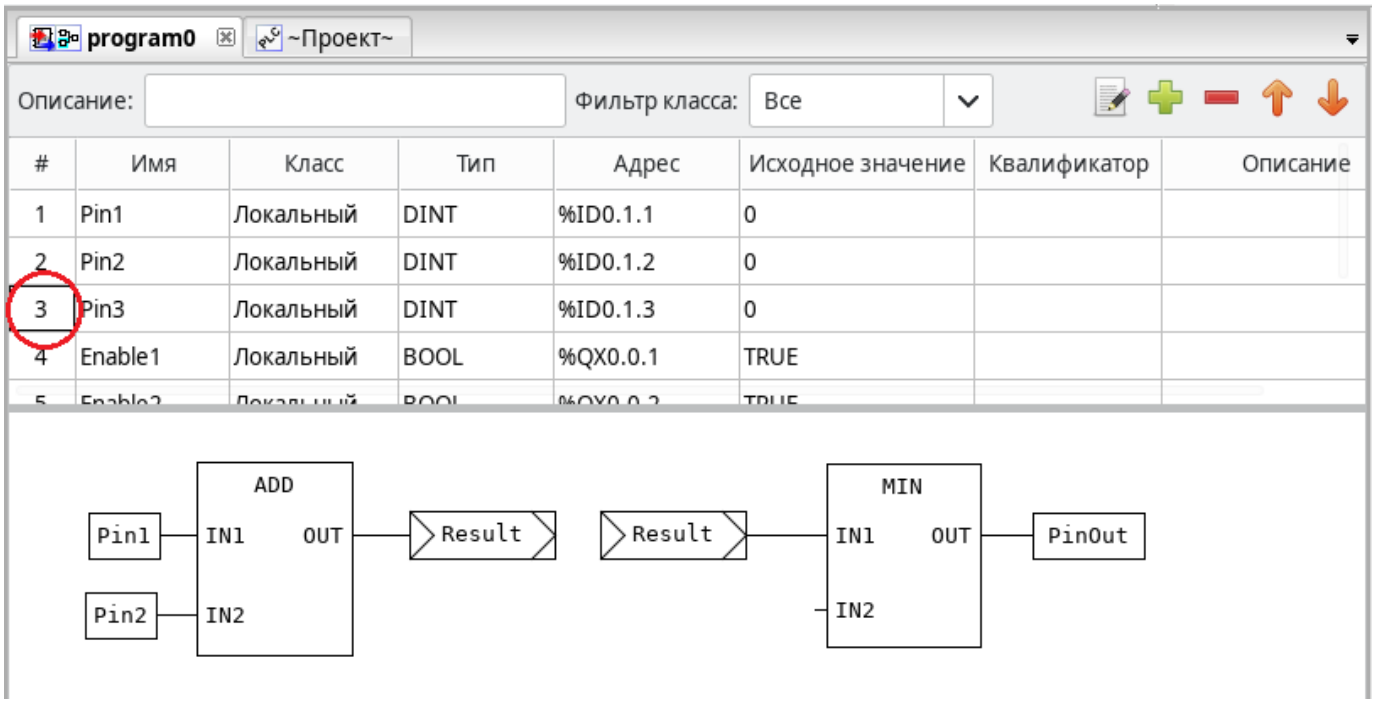


Рисунок 38 – Добавление переменной из панели переменных и констант

Изменить параметры переменной можно в диалоге «Свойства переменной», рис. 39, нажав на неё два раза левой клавишей мыши.

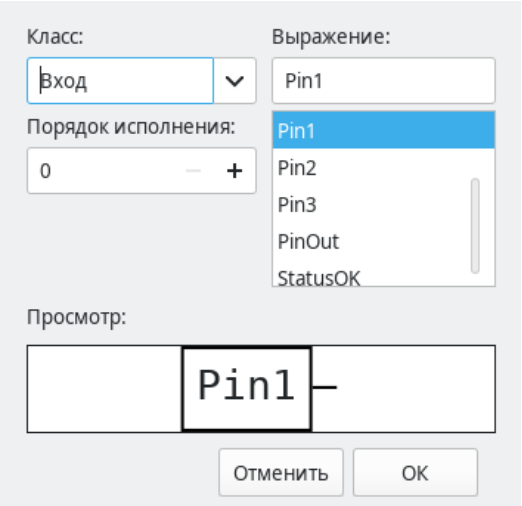


Рисунок 39 – Свойства переменной

В данном диалоге можно задать порядок выполнения переменной и изменить её класс («Вход», «Выход», «Вход/Выход»).

6.9.1.3. Добавление соединения

В тех случаях, когда необходимо передать выходное значение одного функционального блока на один из входов другого, удобно использовать элемент «Подключение». При прямом соединении с помощью перетаскивания выхода одного функционального блока к входу другого получится прямое соединение с помощью чёрной соединительной линии. На схемах с большим количеством функциональных блоков элемент «Подключение» позволяет избежать пересечений прямых соединений.

После выбора добавления элемента «Подключение» появится диалог «Свойства подключения», рис. 40.

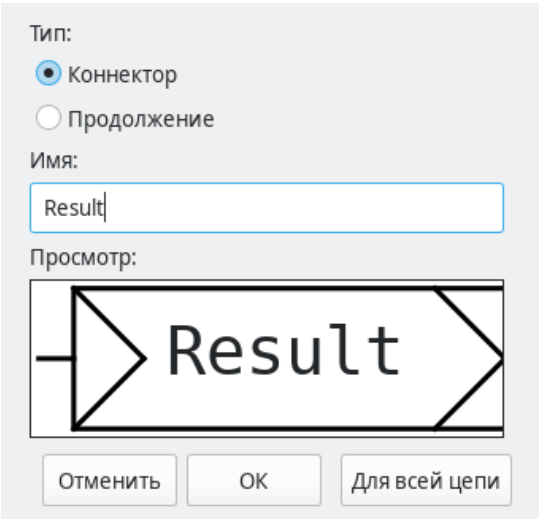


Рисунок 40 – Диалог добавления подключения для FBD

В данном диалоге можно выбрать тип соединения: «Коннектор» – для выходного значения, «Продолжение» – для входного значения, а также необходимо указать имя данного подключения. На рис. 41 представлен пример использования соединений.

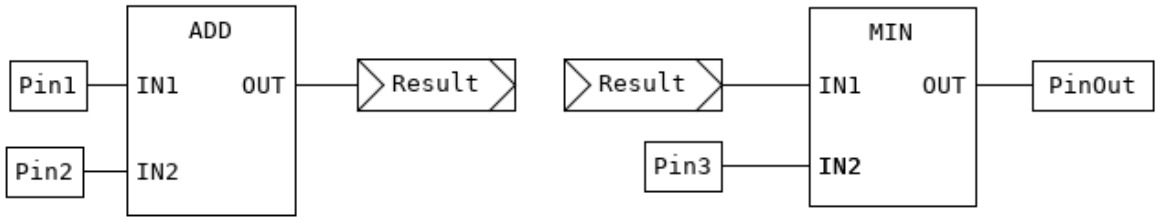


Рисунок 41 – Пример FBD диаграммы с использованием подключений

Функция «ADD» на выходе «OUT» имеет значение, которое с помощью подключения «RESULT» передаётся на вход «IN1» в функцию «MIN». В функции «ADD» используется подключение типа «Коннектор», в функции «MIN» – типа «Продолжение». Имена у этих соединений, соответственно, одинаковые.

6.9.1.4. Добавление комментариев

Редактор FBD диаграмм (и остальные редакторы, о которых будет рассказано ниже) позволяют добавлять комментарии на диаграмму. После выбора на панели редактирования комментария и добавления его в область редактирования появится диалог (см. рис. 42) для ввода текста комментария.

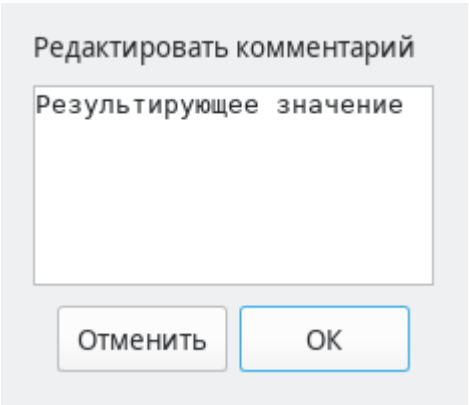


Рисунок 42 – Диалог добавления комментария

После нажатия кнопки OK комментарий появится на диаграмме, рис. 43.

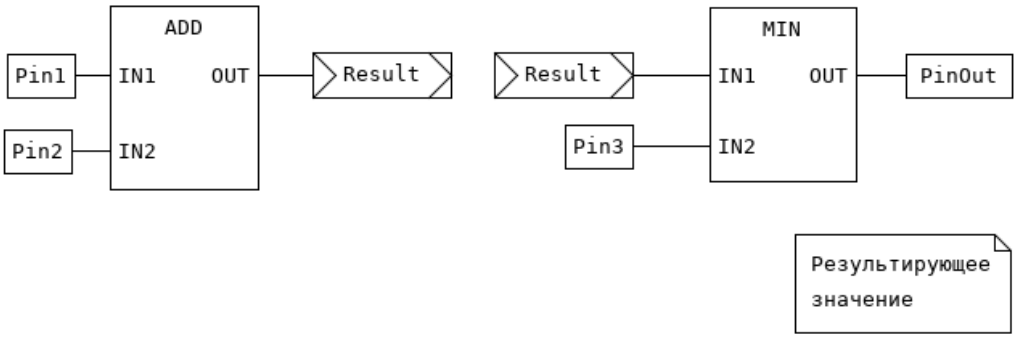


Рисунок 43 – Добавленный комментарий к FBD диаграмме

6.9.1.5. Порядок выполнения функций и функциональных блоков

Последовательность исполнения функций и функциональных блоков определяется порядком их выполнения. Автоматически он регламентируется следующим образом: чем выше и левее расположен верхний левый угол, описывающего функцию или функциональный блок прямоугольника, тем раньше данная функция или функциональный будет выполнен.

Если обратиться к рис. 44, то порядок выполнения функций будет следующим: 1 – ADD; 2 –MIN; 3 – SUB.

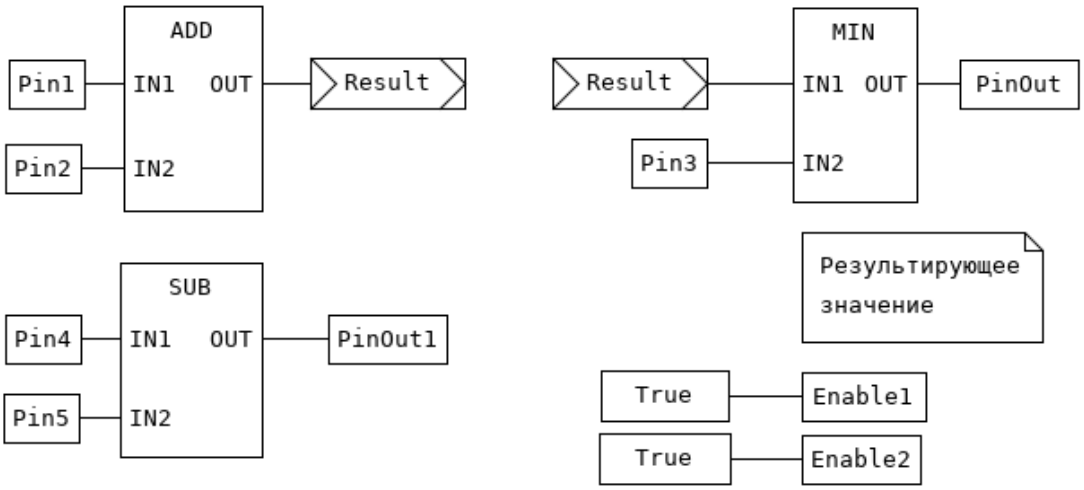


Рисунок 44 – Схема, содержащая функции с порядком выполнения (обсчёта) по расположению

Порядок выполнения может быть изменён вручную с помощью диалога свойств. Данная опция «Порядок исполнения» выделена красным цветом на рис. 45.

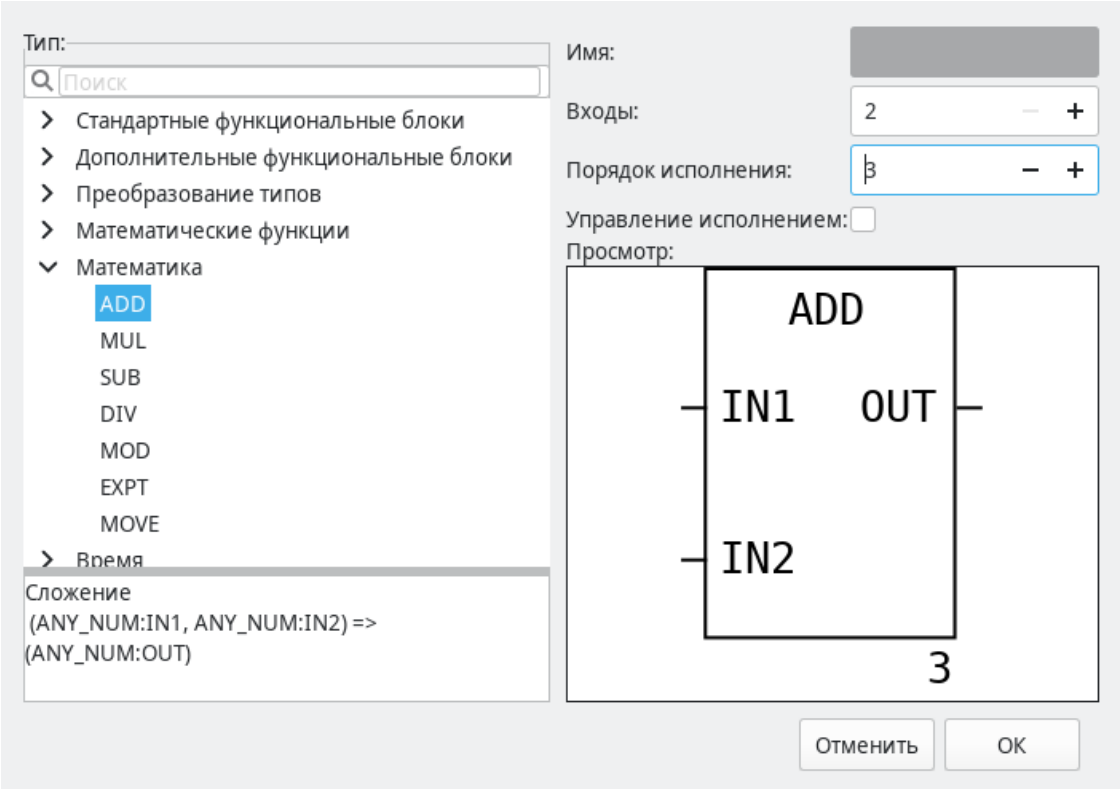


Рисунок 45 – Свойство «порядок исполнения» функции или функционального блока

После задания порядка выполнения для каждой функции или функционального блока на схеме в правом нижнем углу будет указан его порядковый номер выполнения. Пример представлен на рис. 46.

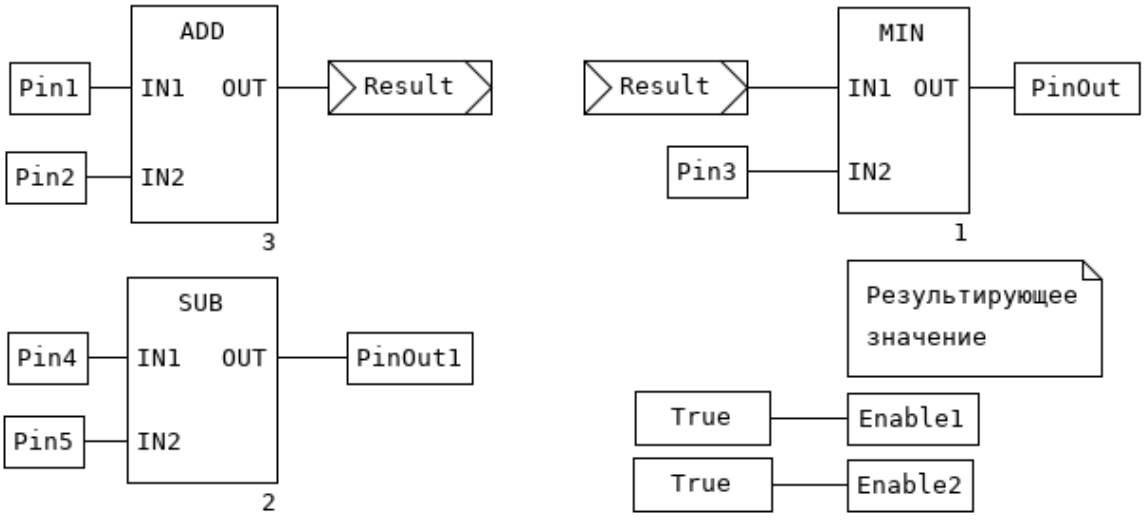


Рисунок 46 – Схема, содержащая функции с порядком выполнения, заданным вручную

Описание языка FBD, основных его конструкций приведены в ГОСТ Р МЭК 61131-3-2016.

6.9.2. Редактор языка LD

Язык LD или РКС (Релейно-Контактные Схемы) представляет собой графическую форму записи логических выражений в виде контактов и катушек реле. Основными элементами языка LD являются: шина питания, катушка, контакт. Добавить данные элементы, так же, как и элементы языка FBD, можно несколькими способами.

Как только активной становится вкладка с редактированием LD диаграммы, в панели инструментов появляется панель (см. рис. 47) с элементами языка LD.

Рисунок 47 – Панель редактирования LD диаграмм

Аналогично редактору языка FBD с помощью данной панели можно добавить все элементы языка LD, а также и FBD, так как есть возможность комбинированного применения языков на одной диаграмме. В таблице 7 приведено описание кнопок данной панели. Описание остальных кнопок, относящихся к языку FBD, находится в табл.7.

Таблица 7 – Кнопки панели редактирования LD диаграммы

Кнопка	Наименование кнопки	Функции кнопки
	Создать новую шину питания	Вызов диалога создания новой шины питания
	Создать новую катушку	Вызов диалога создания новой катушки
	Создать новый контакт	Вызов диалога создания нового контакта

Во всплывающем меню для редактора LD диаграмм, рис. 48, так же, как и в панели инструментов, помимо элементов LD языка, доступны элементы языка FBD.

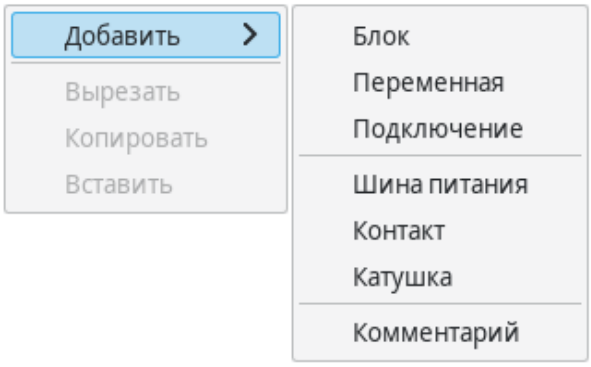


Рисунок 48 – Всплывающее меню редактора языка LD

6.9.2.6. Добавление шины питания

При добавлении шины питания, одним из описанных выше способов, появится диалог «Свойства шины питания», рис. 49.

В данном диалоге указываются следующие свойства:

– тип шины питания: левая или правая;
– количество контактов на добавляемой шине питания.

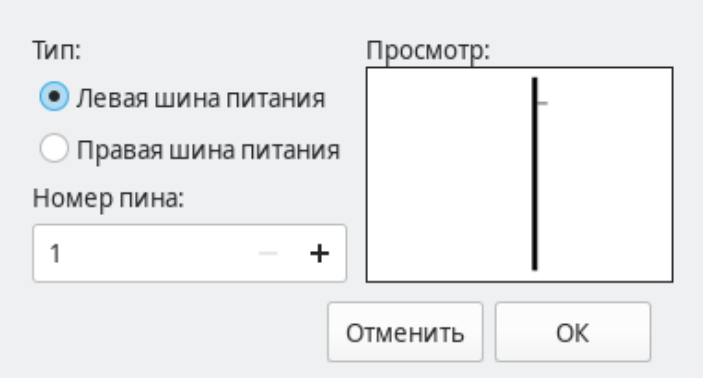


Рисунок 49 – Свойство шины питания

На рис. 50 приведены две добавленные шины питания: левая с тремя контактами и правая с одним контактом.

Рисунок 50 – Шины питания на LD диаграмме

6.9.2.7. Добавление контакта

При добавлении контакта на LD диаграмму появится диалог «Редактирование значения контакта», рис. 51.

Данный диалог позволяет определить модификатор данного контакта:

– «Обычный»;
– «Инверсия»;
– «Нарастающий фронт»;
– «Спадающий фронт».

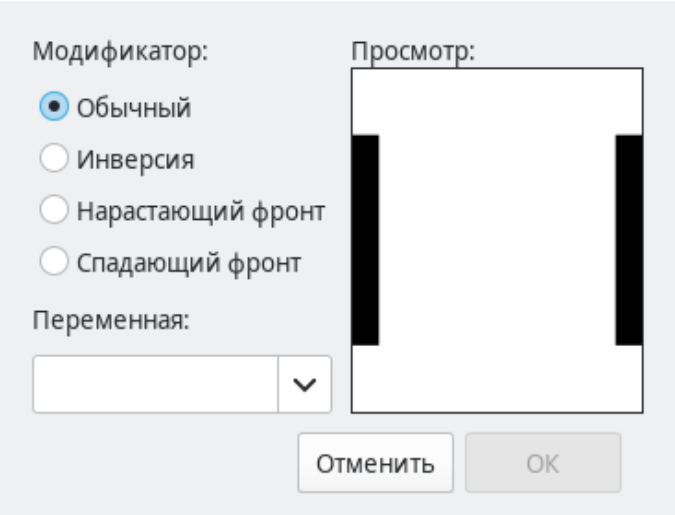


Рисунок 51 – Редактирование контакта

Кроме того, диалог позволяет выбрать из списка «Переменная» переменную, «связываемую» с данным контактом. Следует отметить, что «связываемые» переменные должны быть определены в панели переменных и констант для данного программного модуля типом BOOL.

Еще одним способом добавления контакта на диаграмму является метод Drag&Drop из панели переменных и констант переменной типа BOOL. Для этого необходимо зажать левой кнопкой мыши за первый столбец (который имеет заголовок #) переменную, удовлетворяющую описанным выше критериям и перенести в область редактирования диаграммы, рис. 52.

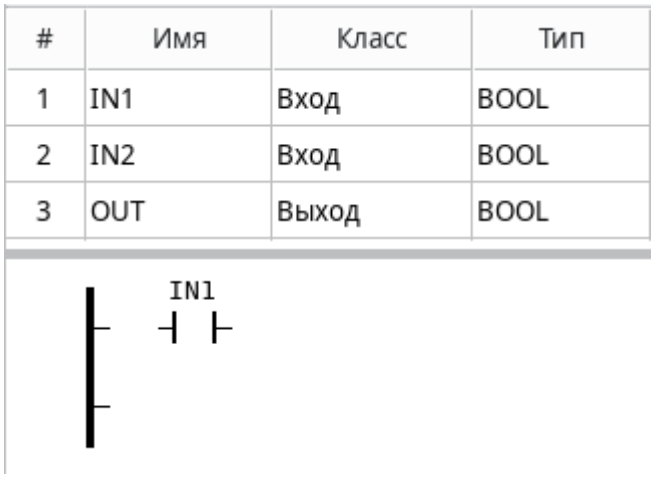


Рисунок 52 – Добавление контакта на диаграмму из панели переменных и констант

6.9.2.8. Добавление катушки

При добавлении катушки на LD диаграмму появится диалог «Редактирование значения катушки», рис. 53.

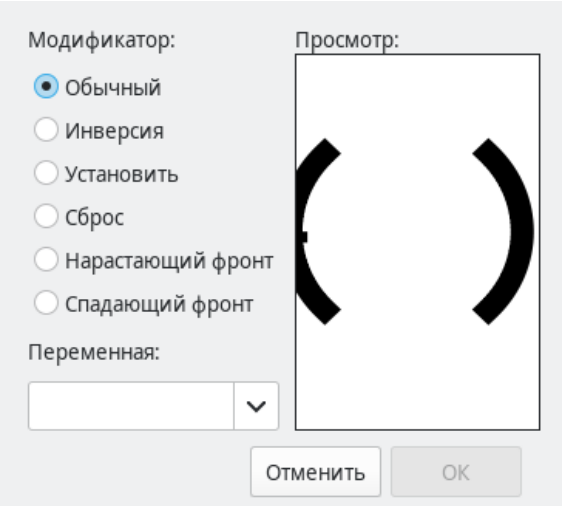


Рисунок 53 – Редактирование катушки

В данном диалоге можно определить модификатор данного контакта:

– «Обычный»;
– «Инверсия»;
– «Установить»;
– «Сброс»;
– «Нарастающий фронт»;
– «Спадающий фронт».

Из списка «Переменная» производится выбор переменной, «связываемой» с данным контактом. Эти переменные, как и для контактов, должны быть определены в панели переменных и констант для данного программного модуля с типом BOOL.

Аналогично, с помощью Drag&Drop, можно добавить и катушки, но в данном случае переменная должна относиться к классу «Выход», рис. 54.

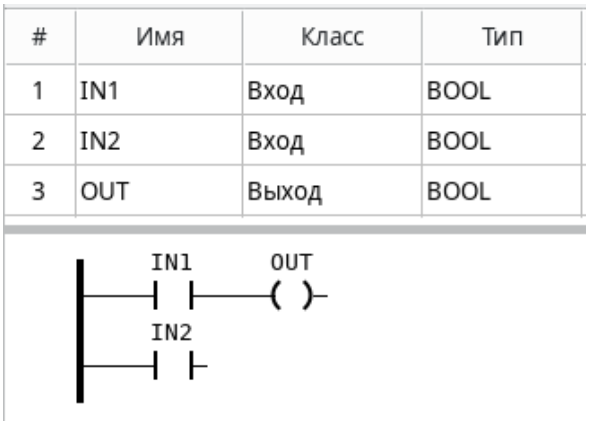


Рисунок 54 – Добавление катушки на диаграмму из панели переменных и констант

Описание языка LD, основных конструкций приведены в ГОСТ Р МЭК 61131-3-2016.

6.9.3. Редактор языка SFC

Основными элементами языка SFC являются: начальный шаг, шаг, переход, блок действий, дивергенции, «прыжок». Программа на языке SFC состоит из набора шагов, связанных переходами.

Как только активной становится вкладка с редактированием SFC диаграммы, в панели инструментов появляется следующая панель (см. рис. 55).

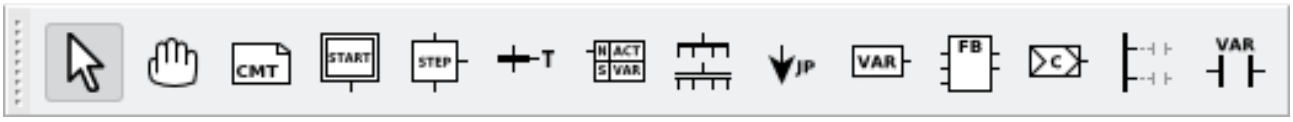


Рисунок 55 – Панель редактирования SFC диаграмм

В табл. 8 приведено описание кнопок данной панели. Описание остальных кнопок, относящихся к языку FBD и LD (за исключением катушки) и так же находящихся на этой панели, приведены в табл. 6 и 7 соответственно.

Таблица 8 – Кнопки панели редактирования SFC диаграммы

Кнопка	Наименование кнопки	Функции кнопки
	Создать новый начальный шаг	Вызов диалога редактирования шага
	Создать новый шаг	Вызов диалога редактирования шага
	Создать новый переход	Вызов диалога редактирования перехода
	Создать новый блок действий	Вызов диалога редактирования блока действий
	Создать новую дивергенцию	Вызов диалога создания новой дивергенции и конвергенции
	Создать новый «прыжок»	Вызов диалога создания «прыжка»

Далее даётся описание добавления приведённых в таблице 8 элементов языка SFC.

6.9.3.9. Добавление исходного шага

Процедура добавления исходного шага и обычного шага ничем не отличается. В обоих случаях вызывается диалог «Редактировать шаг», рис. 56.

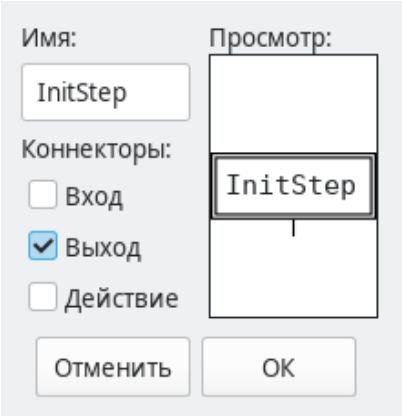


Рисунок 56 – Диалоги редактирования исходного шага и обычного шага SFC диаграммы

Согласно стандарту IEC 61131-3, на SFC диаграмме должен быть один шаг инициализации, который характеризует начальное состояние SFC-диаграммы и отображается со сдвоенными линиями на границах, рис. 57.

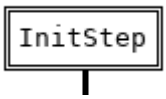


Рисунок 57 – Шаг инициализации языка SFC

В случае, если при добавлении шага не указано его имя – будет выдана ошибка, рис. 58.

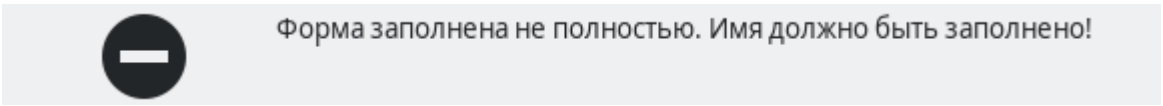


Рисунок 58 – Ошибка отсутствия имени шага во время его добавления

При добавлении шага появляется диалог, в котором можно указать, с помощью галочек его соединители, рис. 59:

– «Вход»;
– «Выход»;
– «Действие».

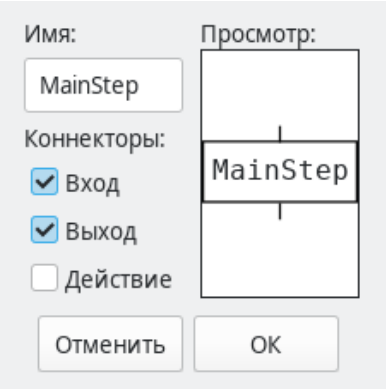


Рисунок 59 – Добавление шага на SFC диаграмму

«Действие» добавляет соединитель для связывания данного шага с блоком действий. «Вход» и «Выход» соединители, как правило, соединены с переходом. Соответственно, после нажатия кнопки OK, на диаграмму будет добавлен шаг с указанными соединителями, рис. 60.

Рисунок 60 – Шаг SFC диаграммы с соединителями входа и действия

6.9.3.10. Добавление перехода

При добавлении на SFC диаграмму перехода, появится диалог «Редактировать переход», рис. 61.

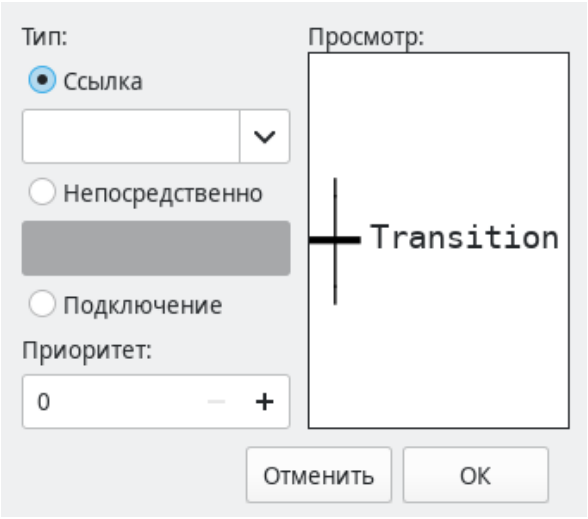


Рисунок 61 – Диалог редактирования перехода для SFC диаграммы

В данном диалоге необходимо выбрать тип перехода и его приоритет. Тип перехода может быть:

– «Ссылка»;
– «Непосредственно»;
– «Подключение».

При выборе типа перехода «Ссылка» в открывающемся списке, рис. 62 будут доступны переходы, предопределённые в дереве проекта для данного программного модуля, написанного на языке SFC. Добавление предопределённого перехода описывается ниже после описания всех добавляемых элементов языка SFC.

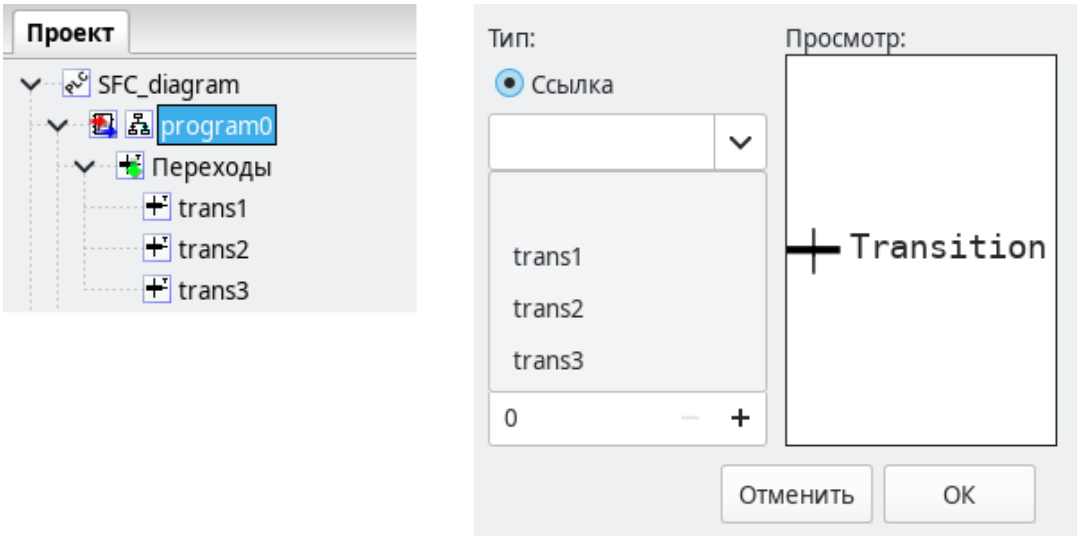


Рисунок 62 – Всплывающий список с доступными предопределёнными переходами

При выборе типа перехода «Непосредственно», рис. 63, условие перехода можно написать в виде выражения на языке ST.

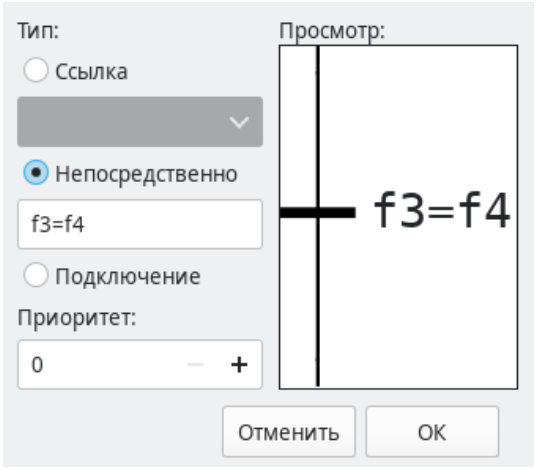


Рисунок 63 – Условие перехода в виде встроенного кода, написанного на языке ST

Реализация перехода таким способом удобна в случае, когда необходимо короткое условие, например, переменные «f3» и «f4» типа INT равны. Встроенный код для такого условия выглядит следующим образом, (см. рис. 63):

```
f3 = f4
```

Так же можно в качестве условия просто указать переменную. В случае её значения равного 0 – будет означать FALSE, все остальные значения – TRUE.

При выборе типа перехода «Подключение», рис. 64, в качестве условия перехода можно использовать выходные значения элементов языка FBD или LD.

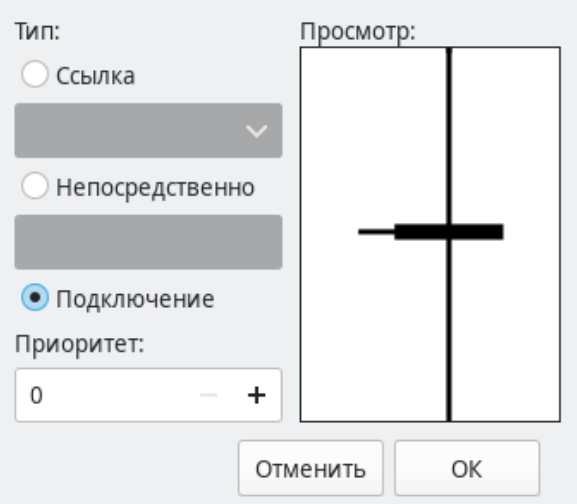


Рисунок 64 – Выбор условия перехода как соединение с элементами других графических языков IEC 61131-3

При выборе типа перехода «Подключение», у добавленного перехода появится слева контакт, который необходимо соединить с выходным значением, например, функционального блока языка FBD или катушки LD диаграммы. Стоит отметить, что данное выходное значение должно быть типа BOOL. Ниже, на рис. 65 красным цветом выделен пример перехода, условия которого задано с помощью языка LD – двух последовательно соединённых контактов языка LD.

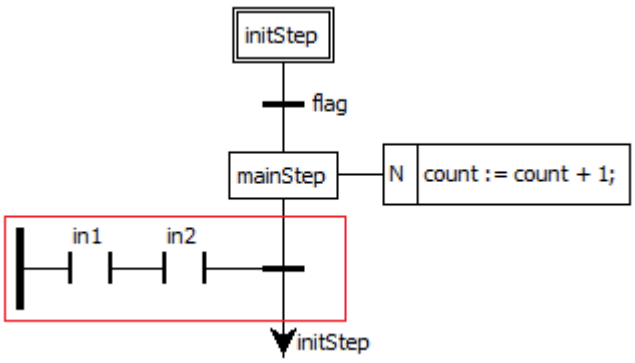


Рисунок 65 – Пример SFC диаграммы, в которой один из переходов задан с помощью языка LD

6.9.3.11. Добавление блока действий

При добавлении блока действий («`Action Block`») на диаграмму, появится диалог «Редактировать свойство блока действий», рис. 66.

Рисунок 66 – Диалог «Редактировать свойство блока действий»

Данный блок действий может содержать набор действий. Добавить новое действие можно нажав кнопку «Добавить» и установив необходимые параметры:

– «Спецификатор»;
– «Длительность»;
– «Тип»: «Действие», «Переменная», «Непосредственно»;
– «Значение»;
– «Индикатор».

Поле «Спецификатор» определяет момент времени, когда действие начинается, сколько времени продолжается и когда заканчивается. Выбрать квалификатор можно из списка, рис. 67.

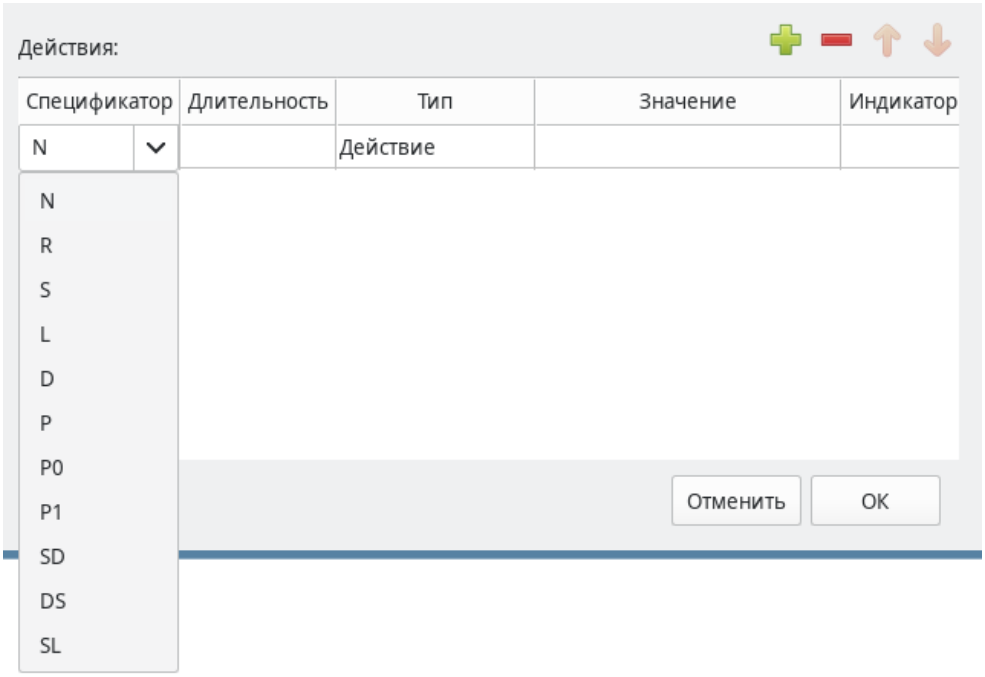


Рисунок 67 – Меню выбора спецификатора для действия в диаграмме SFC

Подробное описание спецификаторов, которые выбираются из предлагаемого списка при добавлении действия приведено в табл. 9.

Таблица 9 – Спецификаторы действий SFC диаграммы

Имя спецификатора	**Поведение блока действия**
N	Действие выполняется, пока данный шаг активен
R	Действие выполняется, когда диаграмма деактивизируется
S	Действие активируется и остаётся активным пока SFC диаграмма выполняется
L	Действие выполняется в течение некоторого заданного интервала времени, после чего выполнение действия останавливается
D	Действие начинает выполняться через некоторое заданное время (если шаг ещё активен) и выполняется до тех пор, пока данный шаг активен
P	Действие выполняется один раз, как только шаг стал активен
P0	Действие выполняется один раз, как только шаг стал неактивен
P1	Действие выполняется один раз, как только шаг стал активен
SD	Действие начинается выполняться через некоторое время, даже в том случае если шаг уже не активен
DS	Действие начинается выполняться через некоторое заданное время, только в том случае если шаг ещё активен
SL	Действие активно в течении некоторого, заданного интервала

Поле «Длительность» необходимо для установки интервала времени необходимого для некоторых спецификаторов, описанных выше в таблице 9.

«Тип» определяет код или конкретную манипуляцию, которая будет выполняться во время активации действия. В случае выбора «Действие» появляется возможность, как и в случае с переходом, использовать предопределённые действия в дереве проекта для данного программного модуля, написанного на языке SFC, рис. 68.

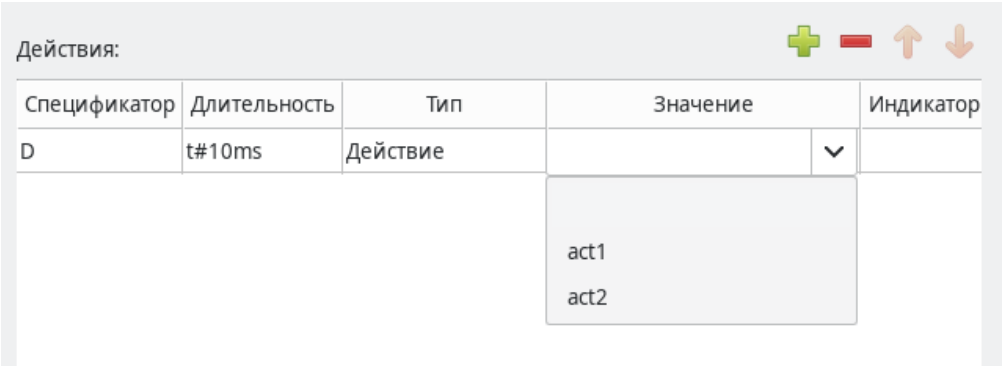


Рисунок 68 – Выбор предопределённого действия

Добавление предопределённого действия также как добавление предопределённого перехода описывается ниже после описания всех добавляемых элементов языка SFC.

В случае выбора типа действия «Переменная» в поле «Значение» появляется возможность выбрать переменные, рис. 69, относящиеся к данному программному модулю.

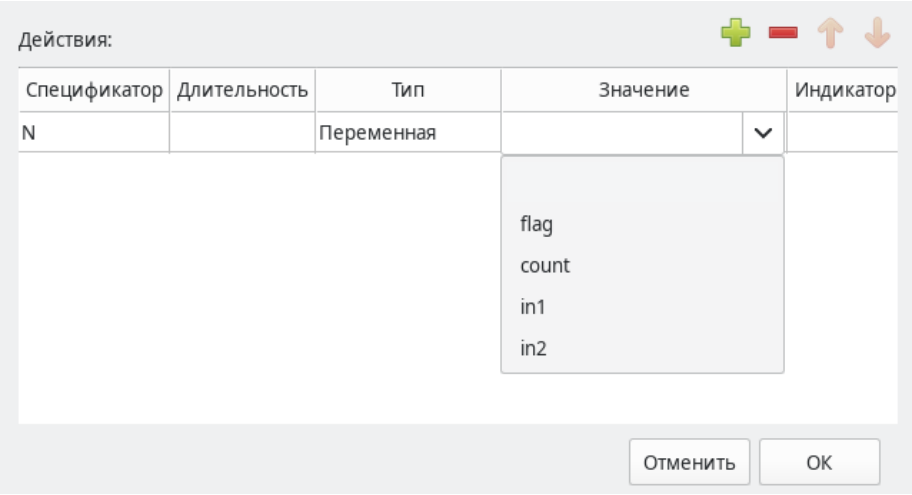


Рисунок 69 – Выбор предопределённой переменной

Как только шаг становится активным, данная переменная в зависимости от своего типа принимает значение 0, 0.0, FALSE и другие нулевые значения типов. Как только действие начинает выполняться, переменная принимает значение 1, 1.0, TRUE и другие единичные значения типов. В случае если действие прекратило своё выполнение переменная снова принимает значение 0, 0.0, FALSE и другое нулевое значение, в зависимости от своего типа.

В случае выбора «Непосредственно», появляется возможность в поле «Значение» написать на языке ST код, который будет выполняться, когда действие становится активным, рис. 70.

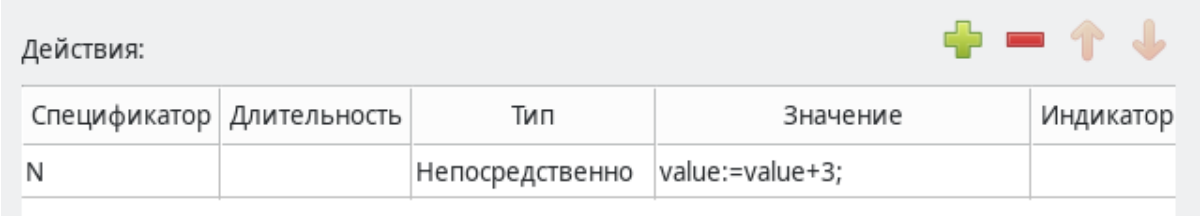


Рисунок 70 – Написание встроенного кода для действия

Следует отметить, что в конце встроенного кода для действия необходимо поставить «;», в отличие от встроенного кода для перехода.

После добавления блока действия на диаграмму необходимо его ассоциировать с конкретным шагом. Данная операция выполняется обычным соединением правого контакта у шага и левого контакта у действия, рис. 71.

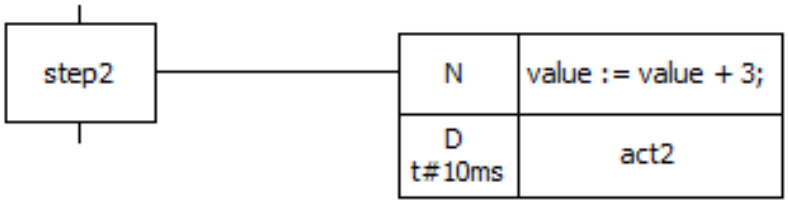


Рисунок 71 – Ассоциирование шага step2 блоком действия, содержащим два действия

6.9.3.12. Добавление ветвления или объединения

При добавлении дивергенции, появится диалог «Создать новое ветвление или объединение», рис. 72.

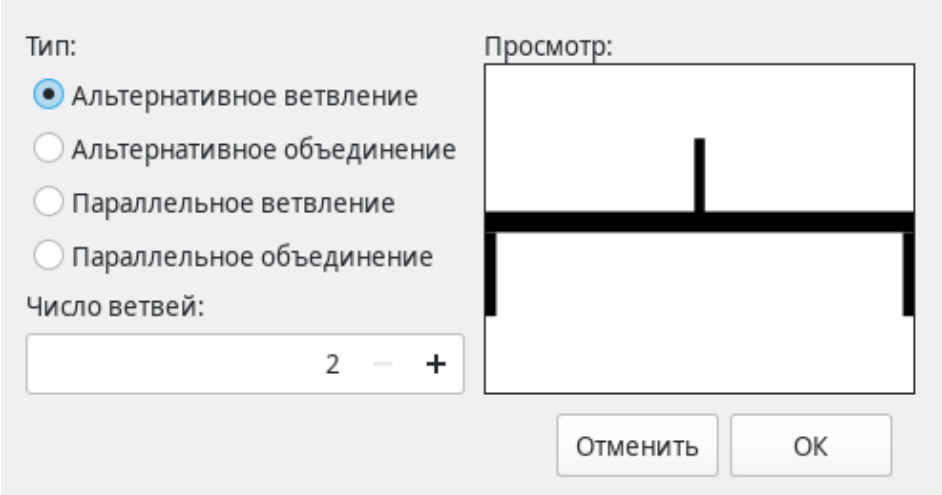


Рисунок 72 – Добавление дивергенции (ветвления)

В первую очередь следует выбрать тип ветвления или объединения:

– «Альтернативное ветвление»;
– «Альтернативное объединение»;
– «Параллельное ветвление»;
– «Параллельное объединение».

Вторым параметром является количество разветвлений, которое определяет на сколько ветвей будет либо расходится (в случае выбора типа дивергенции «Альтернативное ветвление» или «Параллельное ветвление») одна ветвь, либо сколько ветвей будет сходиться в одну ветвь (в случае выбора типа дивергенции «Альтернативное объединение» или «Параллельное объединение»).

Пример дивергенции с двумя разветвлениями показан на рис. 73 и выделен красным цветом.

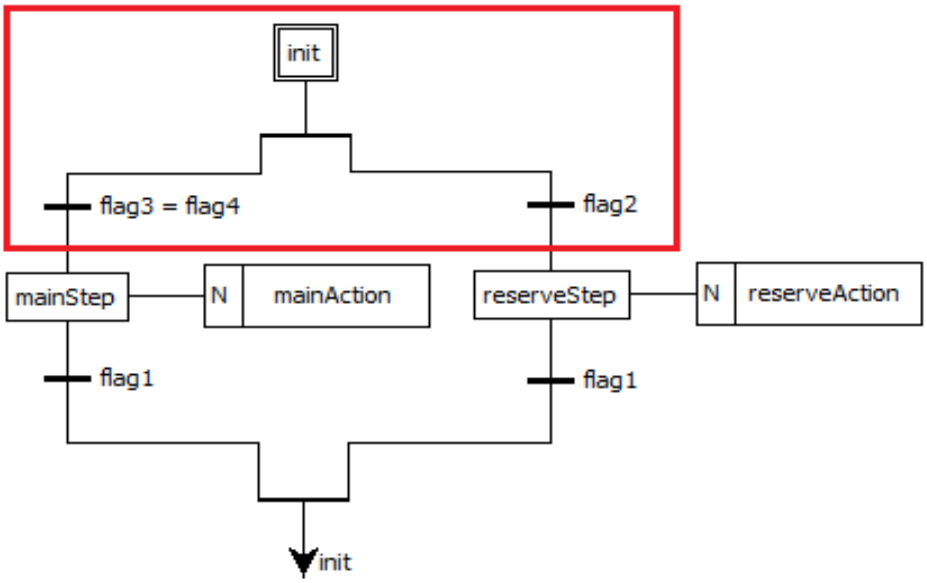


Рисунок 73 – Пример SFC диаграммы, содержащей дивергенцию

Примечание: добавление в SFC диаграмму предопределенных действий («`mainAction`» и «`reserveAction`»), показано ниже (см. п.6.9.3.14).

Пример конвергенции (объединения) выделен красным цветом на рис. 74.

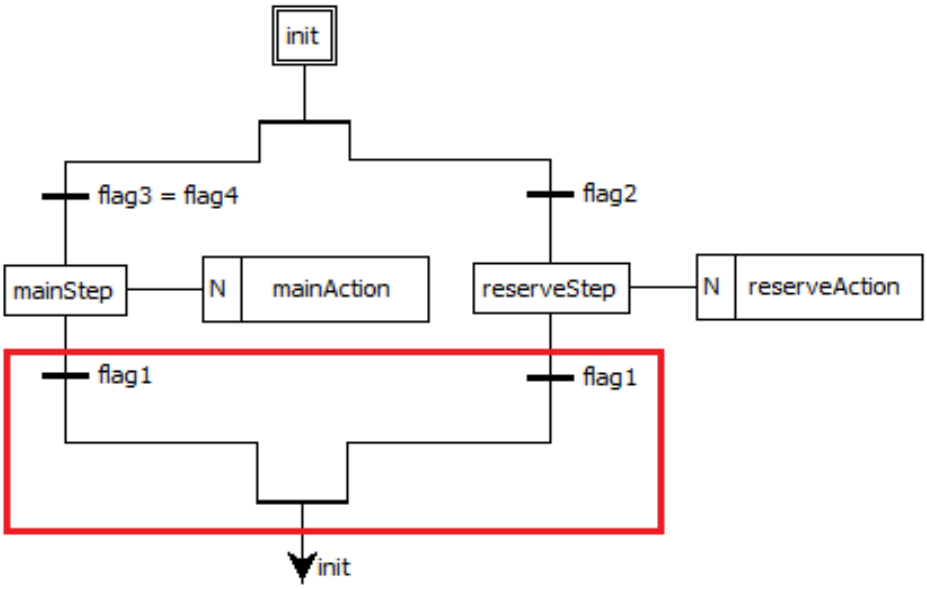


Рисунок 74 – Пример SFC диаграммы, содержащей конвергенцию (объединение)

Примечание: при отладке SFC диаграммы (см. п.8.3.4), будет переключаться только один переход «`flag1`». Тот, который задается "мышью". При этом на диаграмме (см. рис. 74), к переменной «`flag1`», привязаны два перехода.

Пример параллельной дивергенции показан на рис. 75 и выделен красным цветом.

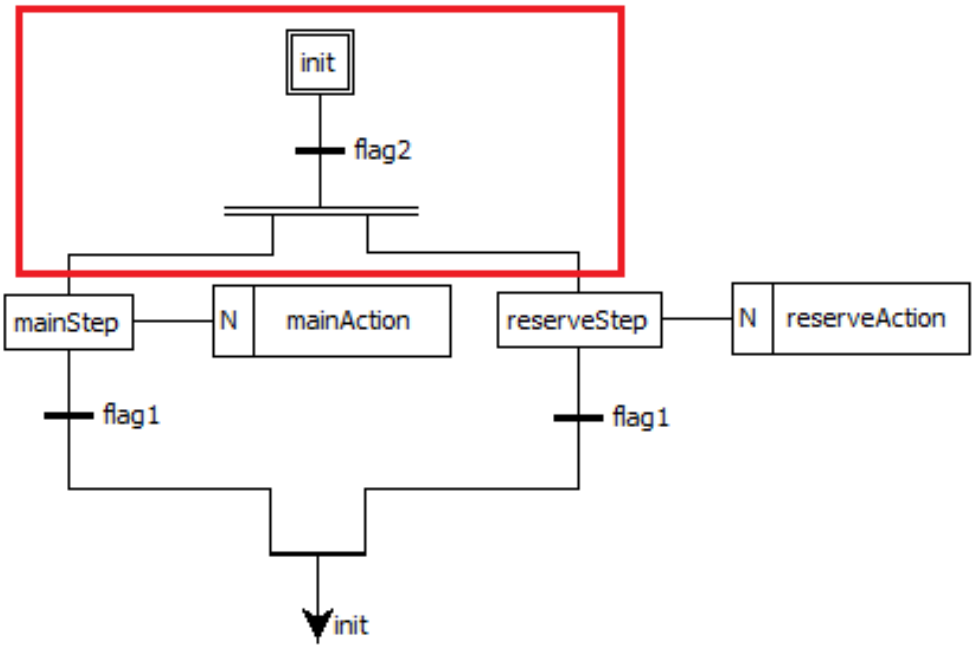


Рисунок 75 – Пример SFC диаграммы, содержащей параллельную дивергенцию

6.9.3.13. Добавление безусловного перехода

Элемент «прыжок» на SFC диаграмме подобен выполнению оператора GOTO при переходе на определённую метку в коде в различных языках программирования. После выбора добавления «прыжка» на SFC диаграмму, появится диалог, рис. 76, в котором необходимо выбрать из списка шаг, к которому будет происходить «прыжок» – переход от одного шага SFC диаграммы к другому.

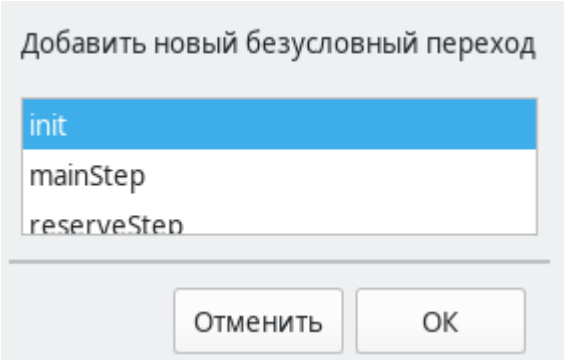


Рисунок 76 – Диалог добавления «прыжка»

В данном диалоге также присутствует и шаг инициализации (начальный шаг). После выбора шага и нажатия кнопки OK. На SFC диаграмме появится стрелочка, которую нужно соединить с переходом, рис. 77. Справа от стрелочки находится имя шага, к которому осуществляется переход в случае выполнения условия перехода, находящегося выше и соединённого с ней.

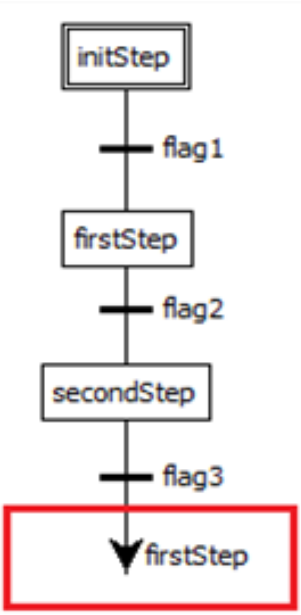


Рисунок 77 – «Прыжок» с шага «secondStep» на «firstStep»

Согласно стандарту IEC 61131-3, между шагом и «прыжком» должен обязательно быть определён переход.

6.9.3.14. Предопределённые условия перехода и действия в структуре проекта

В случае, если необходимо использовать определённое условие перехода между множеством шагов, есть возможность определить данное условие перехода в структуре SFC диаграммы. Данная операция выполняется нажатием на данную SFC диаграмму на структуре проекта (п. 0) правой клавишей мыши и выбором «Добавить переход», рис. 78.

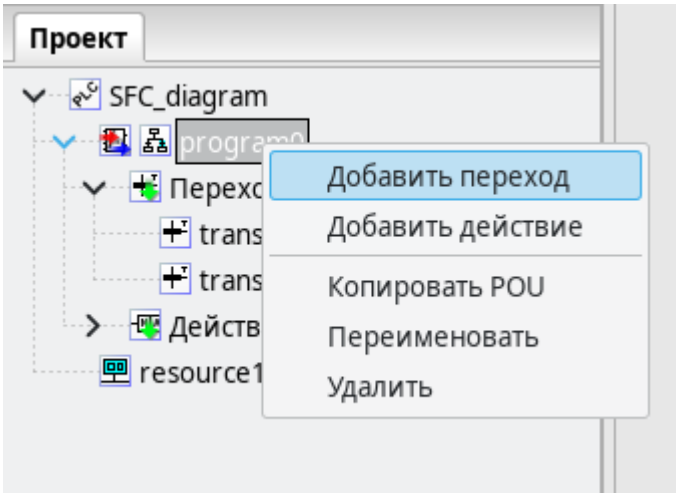


Рисунок 78 – Всплывающее меню SFC диаграммы в панели проекта

Далее появится диалог под названием «Создать новый переход», рис. 79. В нём необходимо выбрать уникальное имя перехода и язык, в котором будет описано данное условие.

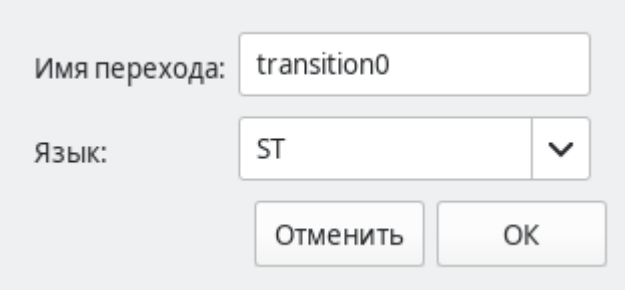


Рисунок 79 – Диалог «Создать новый переход»

В случае, если переходы с введённым именем уже существуют, то будет выведено сообщение об ошибке, рис. 80.

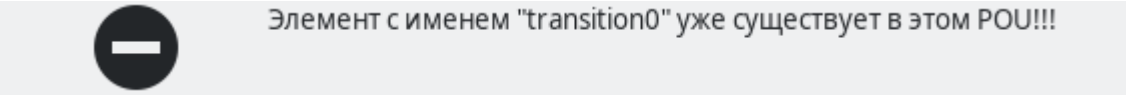


Рисунок 80 – Сообщение об ошибке добавления
существующего программного модуля

Добавление действия в структуру SFC диаграммы происходит аналогично добавлению перехода в данную структуру, рис. 81.

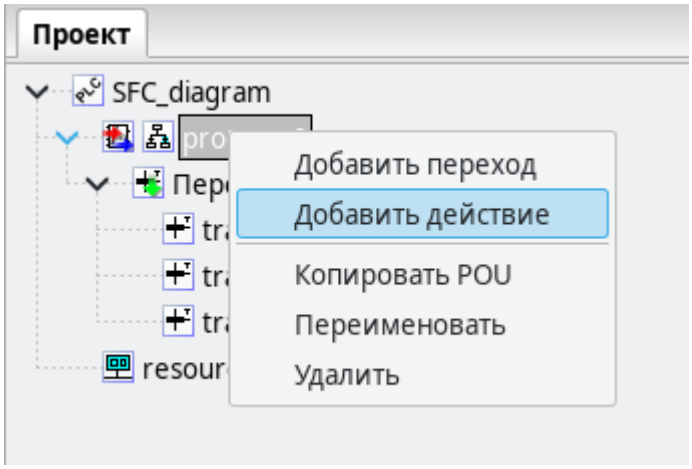


Рисунок 81 – Всплывающее меню SFC для структуры диаграммы

После выбора «Добавить действие» во всплывающем меню, вызванном с помощью нажатия правой клавиши мыши по программному модулю, написанном с помощью языка SFC, появится диалог «Создать новое действие», рис. 82.

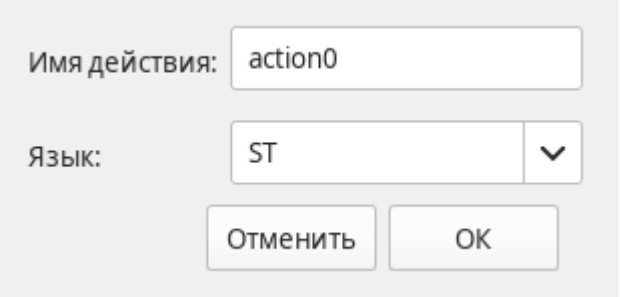


Рисунок 82 – Диалог «Создать новое действие»

В данном диалоге необходимо указать «Имя действия» (должно быть уникальным) и выбрать язык (ST, IL, FBD, LD), на котором будет написано данное действие. Если имя действия не заполнено будет выведено сообщение об ошибке, рис. 83.

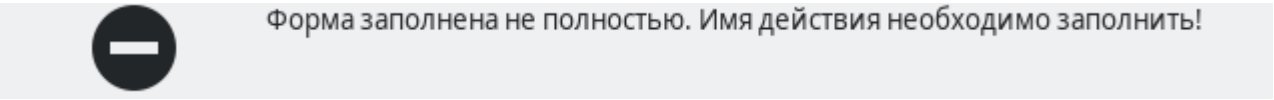


Рисунок 83 – Ошибка заполнения имени действия при его добавлении

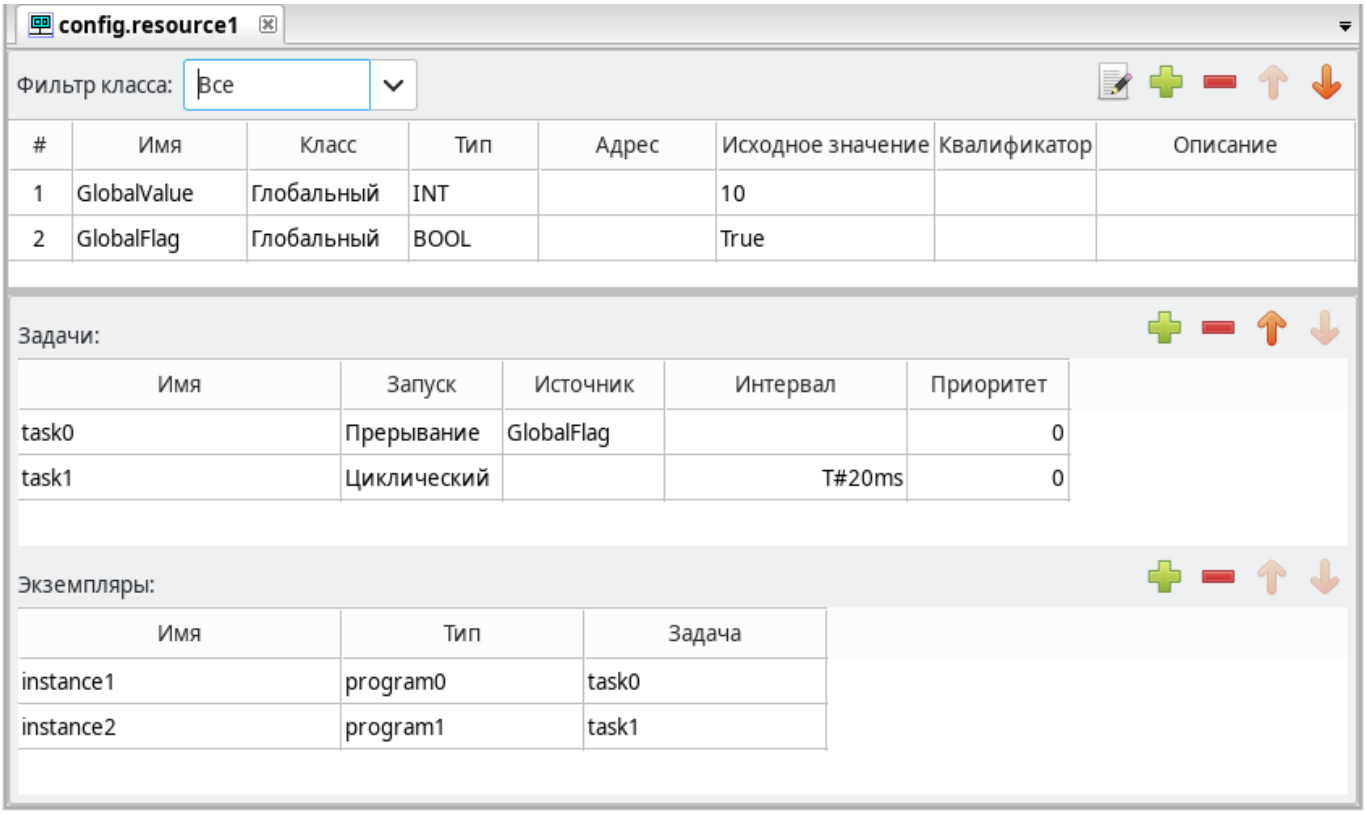
После того как действие добавлено, необходимо реализовать его код на текстовом или графическом языке, в зависимости от языка, который был выбран в диалоге «Создать новое действие» (см. рис. 82). После добавления переходов и действий в дерево проекта они будут доступны для множественного использования.

Описание языка SFC, основных конструкций приведено в ГОСТ Р МЭК 61131-3-2016.

6.10. Средства редактирования ресурса

Панель редактирования ресурса, рис. 84 содержит панель переменных и констант, которая позволяет определять глобальные переменные на уровне ресурса и панели, содержащие задачи и экземпляры.

Добавление переменных в ресурс ничем не отличается от добавления переменных в программные модули, единственное исключение – переменные могут быть только класса «Глобальный». Основной задачей данной панели является возможность добавить экземпляр, указать для него программный модуль типа «Программы», из ранее определённых в проекте, для поля «Тип» и выбрать задачу из добавленных в список «Задачи».



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Описание
1	GlobalValue	Глобальный	INT		10		
2	GlobalFlag	Глобальный	BOOL		True		

Имя	Запуск	Источник	Интервал	Приоритет
task0	Прерывание	GlobalFlag		0
task1	Циклический		T#20ms	0

Имя	Тип	Задача
instance1	program0	task0
instance2	program1	task1

Рисунок 84 – Вкладка ресурс главной рабочей области

6.11. Панель редактирования типа данных

Панель редактирования типа данных, рис. 85, позволяет определить различные параметры создаваемого пользовательского типа данных.

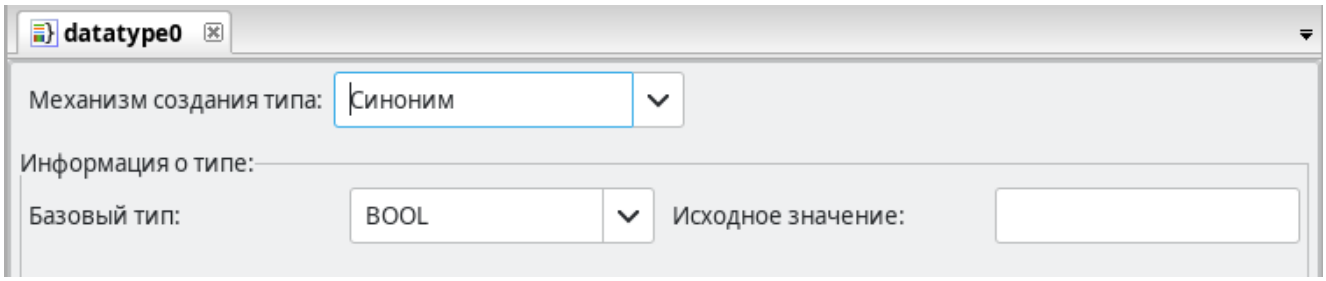


Рисунок 85 – Вкладка создания нового типа данных

Главным параметром является список под названием «Механизм создания нового типа», позволяющим выбрать следующие типы:

– синоним;
– поддиапазон существующего типа (выделение диапазона значений стандартного типа);
– перечисляемый тип;
– массив;
– структура, позволяющая определять тип, основанный на объединении несколько типов.

Далее рассмотрены подробнее параметры для каждого из вышеперечисленных типов

6.11.1. Синоним

При выборе «Синоним», рис. 86, из списка указывается базовый тип и его начальное значение.

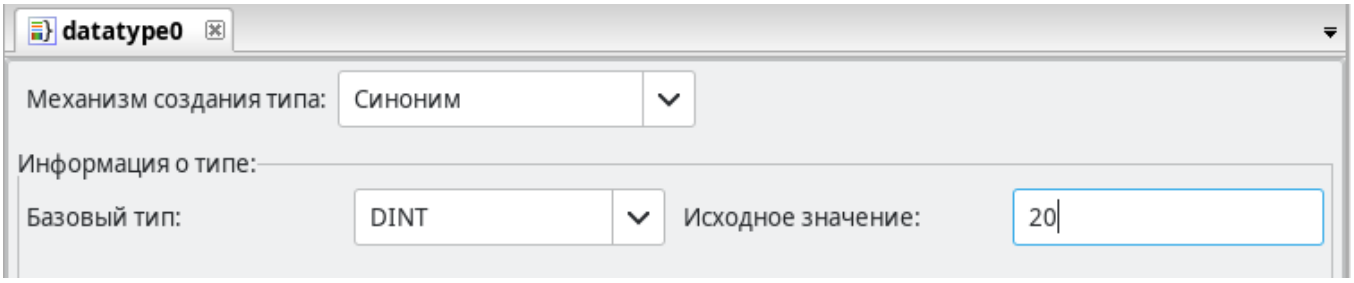


Рисунок 86 – Добавление типа данных - синоним

Созданный тип представляет собой псевдоним (например, аналогично использованию typedef в языке С) уже существующего типа.

6.11.2. Поддиапазон существующего типа

В случае выбора механизма создания нового типа «Поддиапазон существующего типа», помимо базового типа и начального значения производится установка параметров «Минимум» и «Максимум», рис. 87, то есть соответственно минимального и максимального значения, которое может принимать создаваемый тип данных.

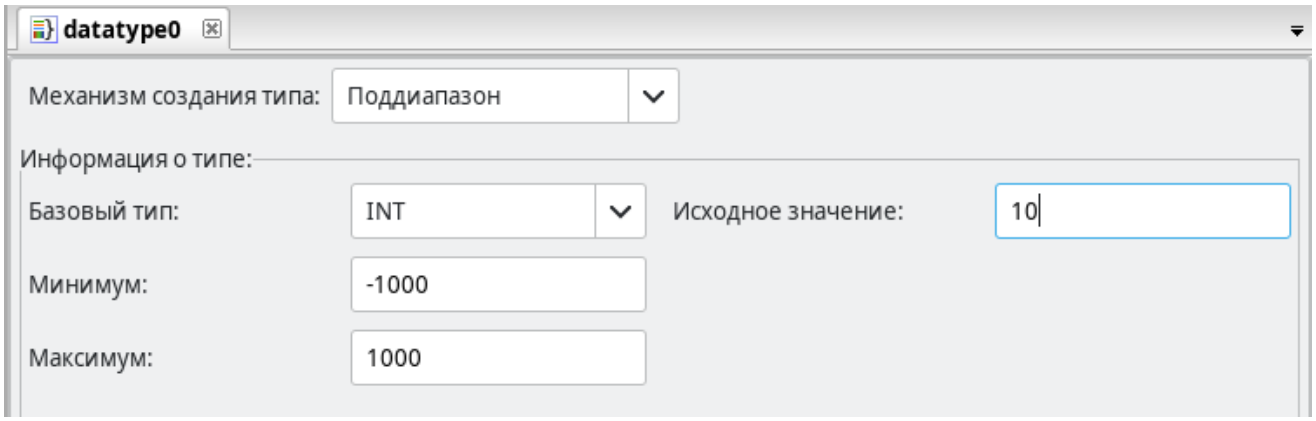


Рисунок 87 – Добавление нового типа данных - поддиапазон

6.11.3. Перечисляемый тип

При выборе механизма создания нового типа «Перечисление», рис. 88, появится панель, содержащая таблицу, в которой можно задать список возможных значений данного перечисляемого типа.

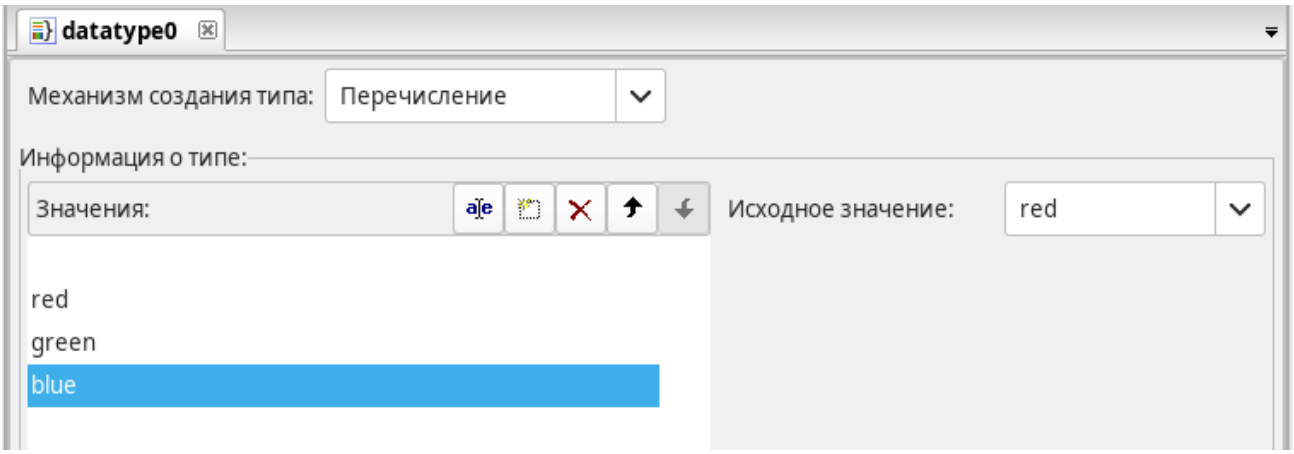


Рисунок 88 – Добавление нового типа данных - перечисление

Добавление, редактирование, удаление, перемещение данных значений осуществляется с помощью кнопок, описание которых приведено в табл. Г.1 .

Таблица 10 – Кнопки редактирования значений перечисляемого типа

Кнопка	Наименование кнопки	Функции кнопки
	Редактировать элемент	Редактировать выделенное поле в таблице
	Добавить элемент	Добавить новое поле в таблицу
	Удалить	Удалить выделенное поле в таблице
	Переместить выше	Переместить вверх выделенное поле в таблице
	Переместить ниже	Переместить вниз выделенное поле в таблице

Также есть возможность задать начальное значение данного перечисляемого типа в поле «Исходное значение».

6.11.4. Массив

При выборе механизма создания нового типа «Массив», рис. 89 появится панель, в которой необходимо указать базовый тип, начальное значение, а также размерность массива.

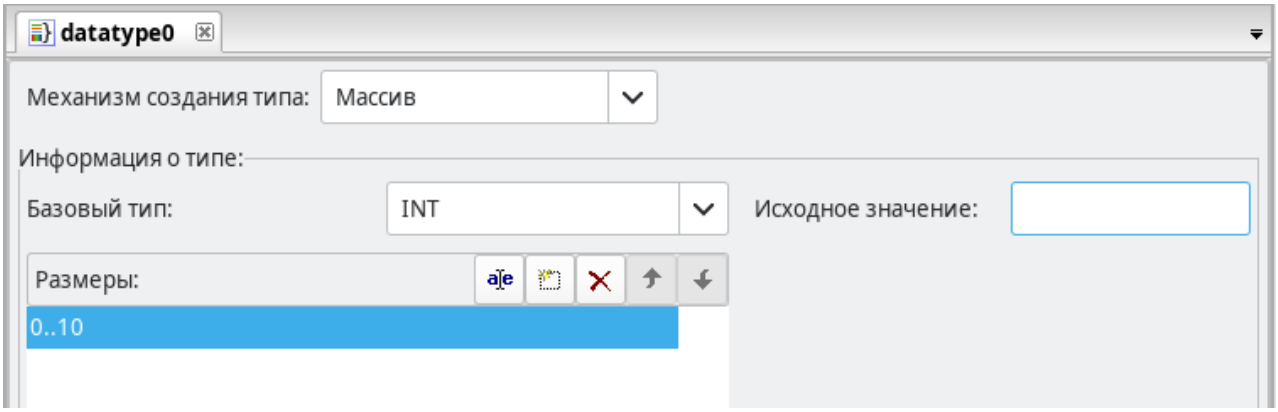


Рисунок 89 – Добавление типа данных – массива

Размерность массива задаётся в следующем формате:

```
<начальное значение>..<конечное значение>
```

6.11.5. Структура

При выборе механизма создания нового типа «Структура», рис. 90, в появившейся таблице необходимо добавить необходимое количество полей структуры. Каждое поле имеет своё имя, тип и начальное значение. Также поля могут иметь комментарии.

Для удобства пользователь может задавать переменные в виде текста с помощью соответствующей кнопки.

Важно! Изменения будут применены только при возврате в просмотр в виде таблицы или ручном сохранении переменных с помощью соответствующей кнопки.

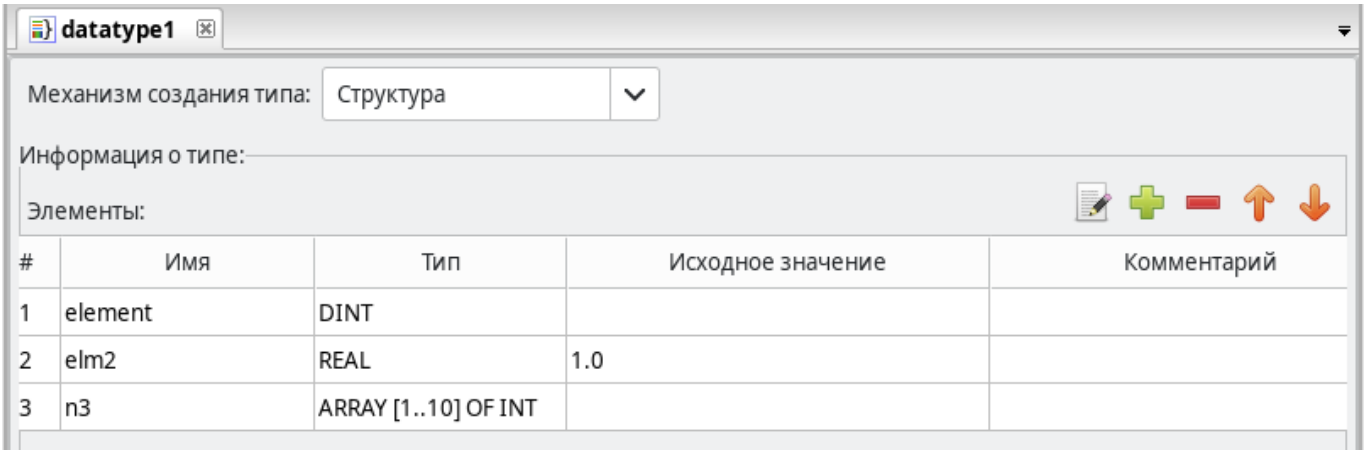


Рисунок 90 – Добавление типа данных – структуры

Добавленные типы данных могут использоваться также, как и стандартные при реализации алгоритмов и логики выполнения программных модулей.

Выше были рассмотрены варианты редактирования различных элементов, из которых состоит проект, согласно стандарту МЭК 61131-3-2016. Далее будет продолжено рассмотрение остальных компонент среды разработки и отладки программ.

6.12. Доступ к экземплярам проекта

Панель экземпляров проекта, рис. 91, обычно располагается слева в среде разработки и отладки программ и отображаемые в ней экземпляры зависят от выбранного элемента в структуре проекта.

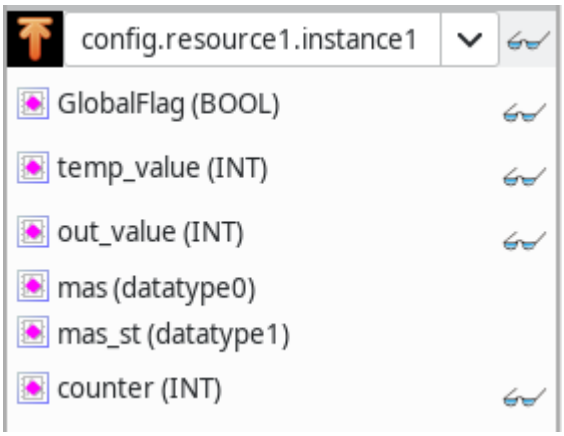


Рисунок 91 – Панель экземпляров проекта

Примечание: в примере, представленном на рис.91, «mas» является массивом, а «mas_st» - структурой. Они соответствуют примерам, приведенным выше (см. рис.89–90).

При выборе в структуре проекта элемента, соответствующего ресурсу, в панели экземпляров проекта будут отображены экземпляры (см. п. 0), определённые в данном ресурсе, а также глобальные переменные ресурса. На рис. 92 показано, как в панели редактирования ресурса определены две глобальных переменные «GlobalValue» и «GlobalFlag», а также два экземпляра программных модулей «inctance0» и «inctance1».

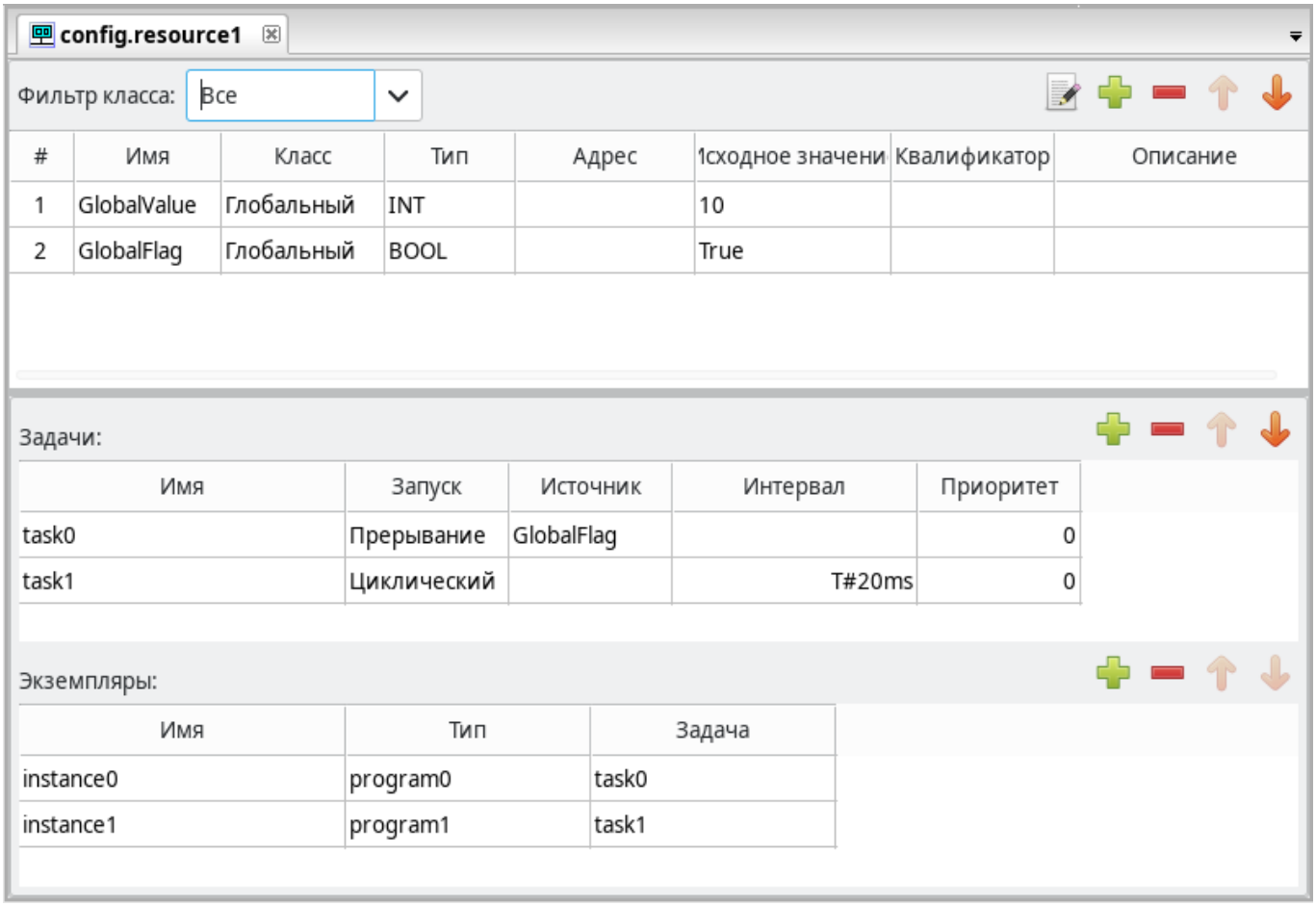


Рисунок 92 – Два экземпляра проекта в панели редактирования ресурса

Соответственно, при выборе в структуре проекта ресурса, в котором определены экземпляры (описанные выше) и глобальная переменная, панель экземпляров будет выглядеть, как показано на рис. 93.

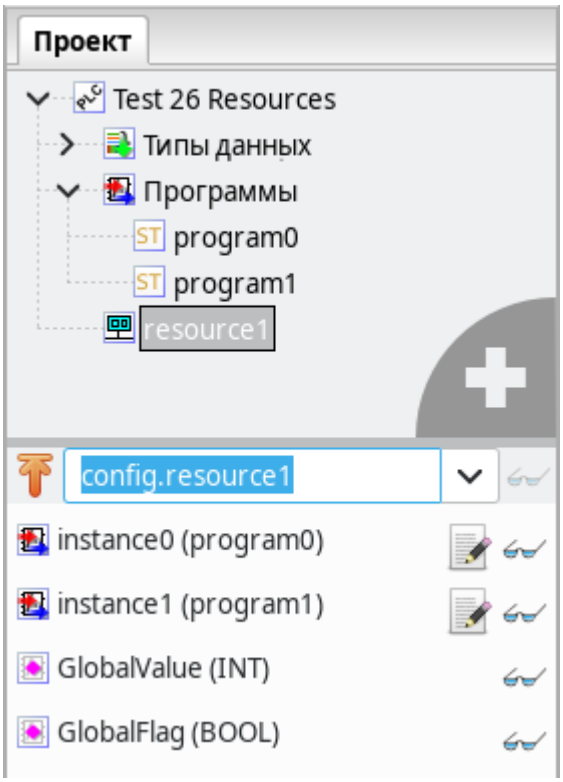
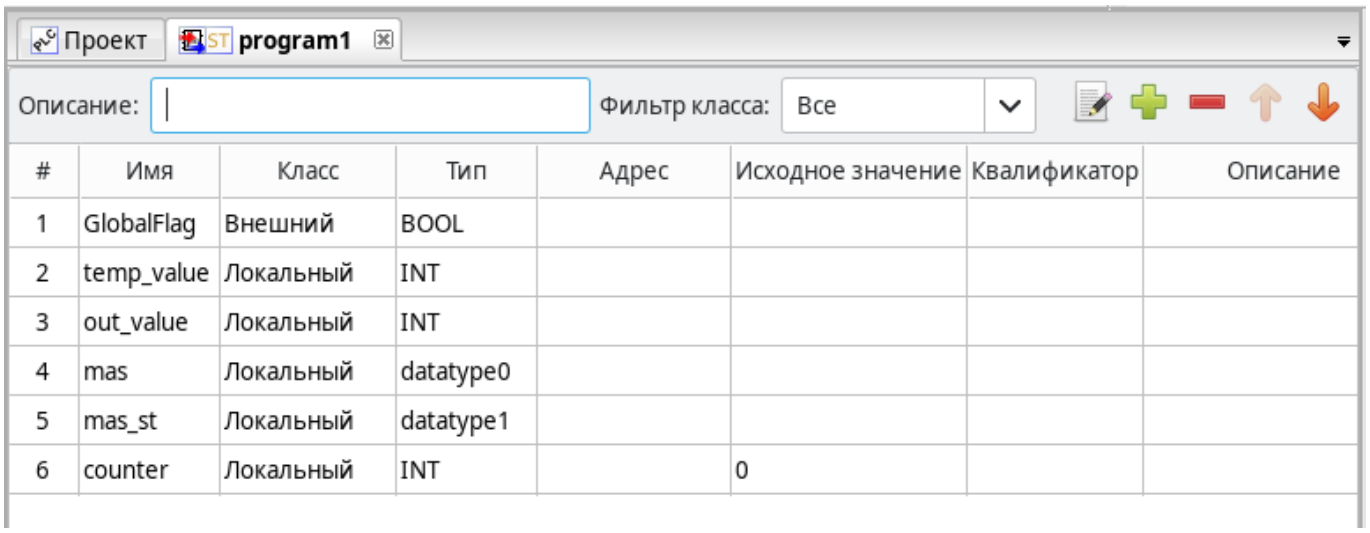


Рисунок 93 – Панель экземпляров при выборе элемента ресурса в структуре проекта

При выборе в структуре проекта элемента, соответствующего программным модулям «Программа» и «Функциональный блок» в панели экземпляров, будут отображены переменные, определённые в них. Ниже на рис. 94 приведён пример программного модуля типа «Программа» с именем «program0», в котором определено 6 переменных различных классов.



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Описание
1	GlobalFlag	Внешний	BOOL				
2	temp_value	Локальный	INT				
3	out_value	Локальный	INT				
4	mas	Локальный	datatype0				
5	mas_st	Локальный	datatype1				
6	counter	Локальный	INT		0		

Рисунок 94 – Программный модуля типа «Программа»

Соответственно, при выборе в дереве проекта данного программного модуля, в панели экземпляров будут отображены, переменные, определённые выше, рис. 95.

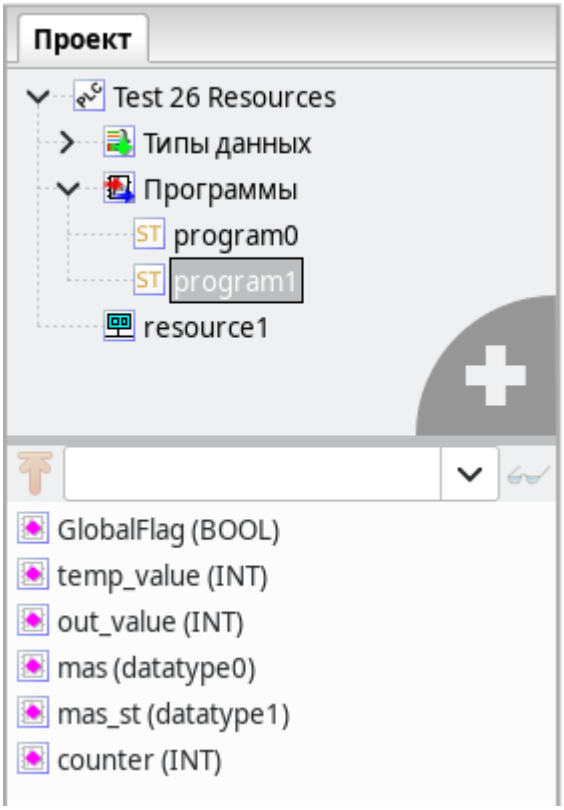


Рисунок 95 – Панель экземпляра проекта при выборе в дереве проекта программного модуля типа «Программа»

В случае выбора других элементов в структуре проекта, панель отладки будет пустой. С правой стороны от каждого элемента в панели отладки располагаются кнопки, назначение которых описано в табл. 11.

Таблица 11 – Кнопки на панели экземпляров проекта

Кнопка	Функции кнопки
	Кнопка добавления переменных из панели экземпляров
	Кнопка запуска режима отладки для экземпляра

В случае нажатия кнопки запуска режима отладки, для экземпляра программы, написанной на одном из графических языков (FBD, LD или SFC), откроется вкладка с панелью, на которой диаграмма будет отображена в режиме отладки (см. п. 6.12). Если кнопка запуска режима отладки нажимается для элемента переменной, то переменная будет добавлена в панель отладки (см. п. 6.12).

Описанные выше кнопки доступны только в режиме отладки прикладной программы. Про данный режим подробнее рассказывается в п. 8.

6.13. Доступ к библиотеке функций и функциональным блокам

Панель библиотеки функций и функциональных блоков (см. рис. 96), как правило, располагается справа в среде разработки и отладки программ. Она содержит коллекцию стандартных функций и функциональных блоков, разделённых по разделам в соответствии с их назначением, которые доступны при написании алгоритмов и логики работы программных модулей.

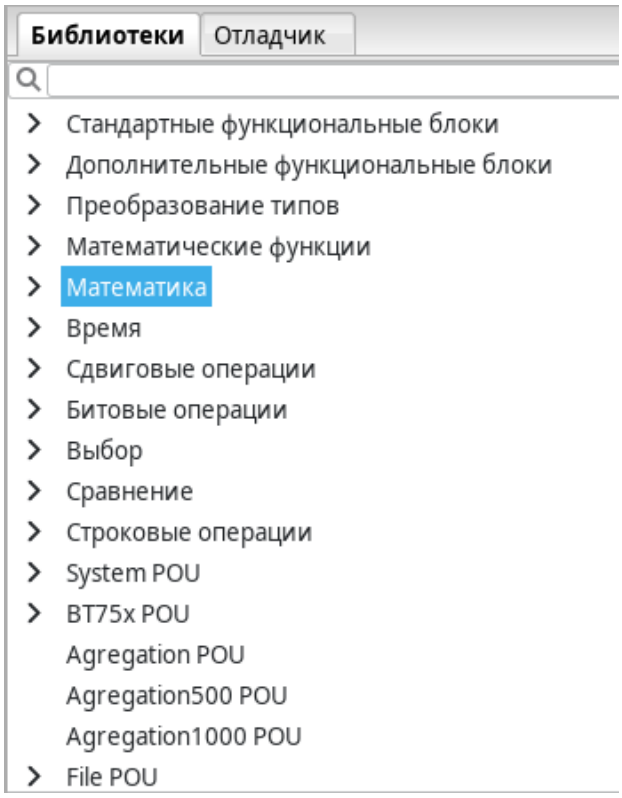


Рисунок 96 – Панель библиотеки функций и функциональных блоков

Выделены следующие разделы для функций и функциональных блоков: стандартные, дополнительные, преобразование типов, математические функции, время, сдвиговые операции, битовые операции, выбор, сравнение, строковые операции.

Помимо стандартных функций и функциональных блоков, данная панель содержит раздел «Пользовательские POU». В него попадают функции и функциональные блоки, добавленные в конкретный проект, то есть содержащиеся в структуре проекта.

Использование данных функций и функциональных блоков осуществляется перетаскиванием необходимого блока с помощью зажатой левой кнопки мыши (Drag&Drop) в область редактирования: либо текстовый редактор, либо графический редактор.

6.14. Отладочная консоль

Панель, содержащая отладочную консоль, как правило, располагается в правом нижнем углу СРиО, рис. 97.

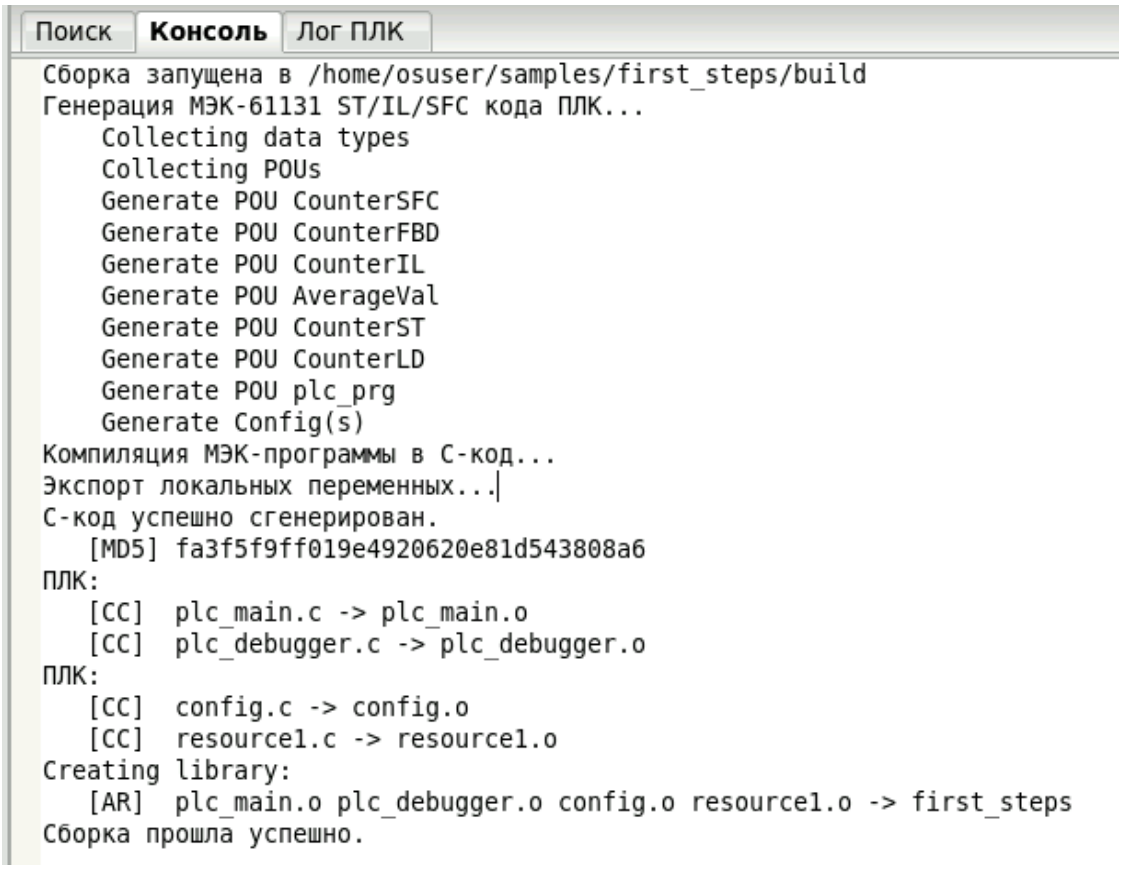


Рисунок 97 – Успешная сборка в отладочной консоли

Она служит для отображения текстовых сообщений:

– результатов генерации ST и C кода;
– результатов компиляции и компоновки прикладной программы;
– процесса соединения и передачи прикладной программы на целевое устройство;
– различных промежуточных манипуляций в процессы создания прикладной программы.

Предупреждения СРиО или ошибки компиляторов (MatIEC или C кода) выводятся красным шрифтом. Критические ошибки выделяются красным цветом на жёлтым фоне, рис. 98.

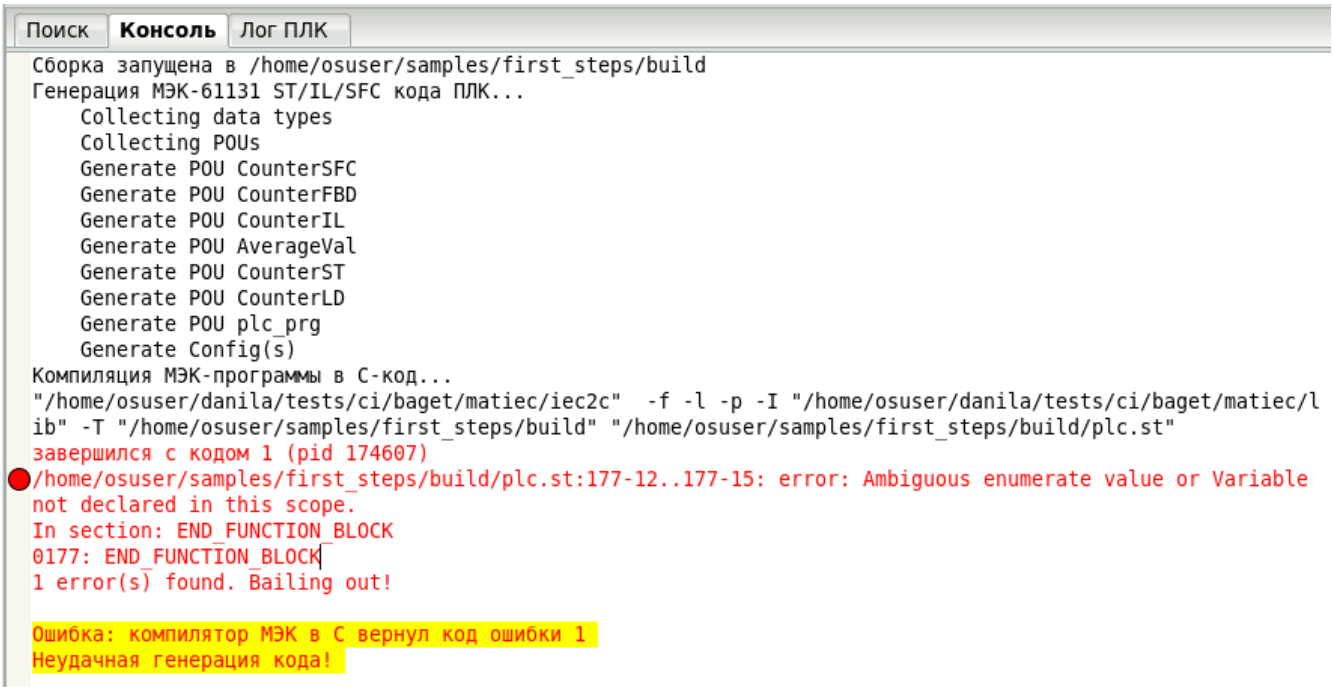


Рисунок 98 – Ошибка сборки проекта в отладочной консоли

6.15. Средства поиска элементов в проекте

Для поиска интересующего элемента в проекте используется диалог «Поиск в проекте» (см. рис. 99). Его вызов происходит с помощью главного меню программы (п. 6.1) или панели инструментов (п. 6.2).

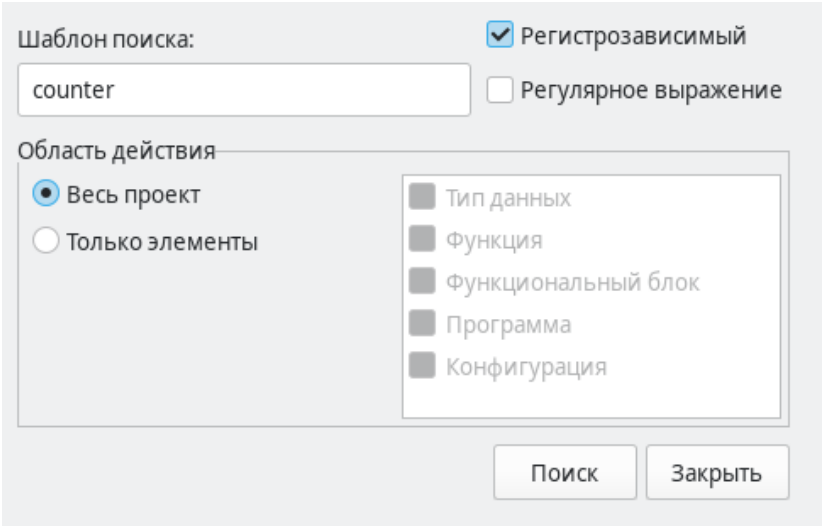


Рисунок 99 – Диалог поиска в проекте

В появившемся диалоге можно установить различные параметры поиска: шаблон поиска, область поиска, чувствительность к регистру при поиске, а также записать шаблон поиска в виде регулярного выражения. После того как все параметры установлены, необходимо нажать кнопку «Поиск» в этом диалоге. На рис. 100 приведён пример поиска элемента с именем «counter».

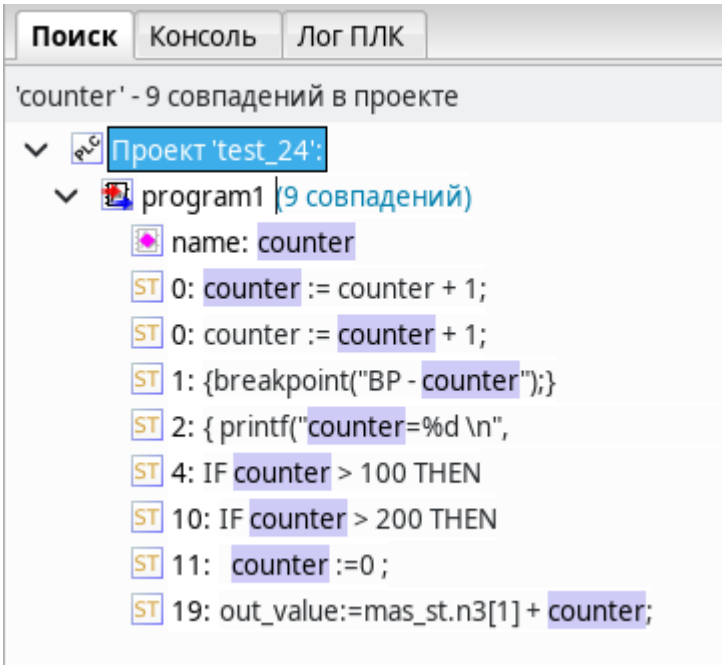


Рисунок 100 – Результат поиска элемента в проекте

Результат поиска выводится в иерархической структуре. При двойном щелчке по одному из результатов – данный элемент будет выделен в проекте оранжевым цветом.

6.16. Доступ к средствам отладки

Панель отладки располагается в правой части среды разработки и отладки программ, рис. 101.

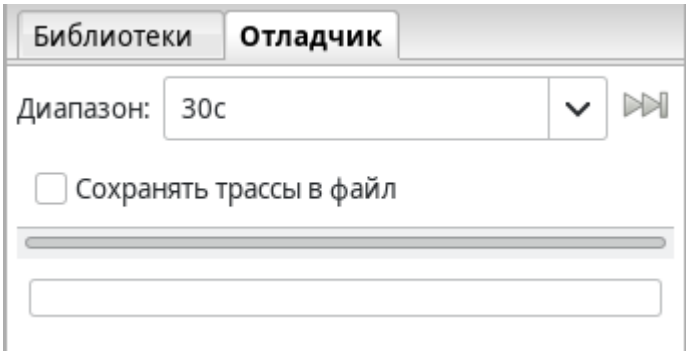


Рисунок 101 – Панель отладки

Данная панель отображает переменные и их значения. Добавление переменных осуществляется с помощью панели экземпляров проекта (см. п. 6.12).

Изменение значений переменной во время отладки программы осуществляется нажатием клавишей мыши на значок форсирования (задания)

значения переменной, выделен красным на рис. 102. Перед этим рекомендуется вывести график изменения значений переменной, при помощи кнопки (см. табл. 12).

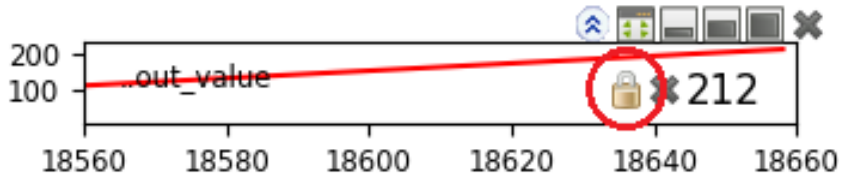


Рисунок 102 – Меню манипуляций со значением переменной

Далее появится диалог ввода значения для выбранной переменной, рис. 103.

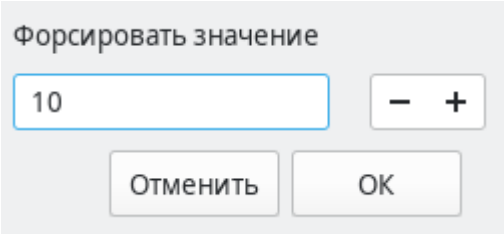


Рисунок 103 – Установка значения переменной во время отладки

Примечание: пользовательские типы данных (см. рис.89 – 91) не доступны для отладки в текущей реализации СРиО.

6.17. Средства отображения графика изменения значения переменной в режиме отладки

Данное средство позволяет просматривать график изменения значения переменной, рис. 104, в случае, если в панели экземпляров проекта (см. п. 6.12) для переменной нажимается кнопка отображения графика изменения значения переменной в режиме отладки (см. табл.12).

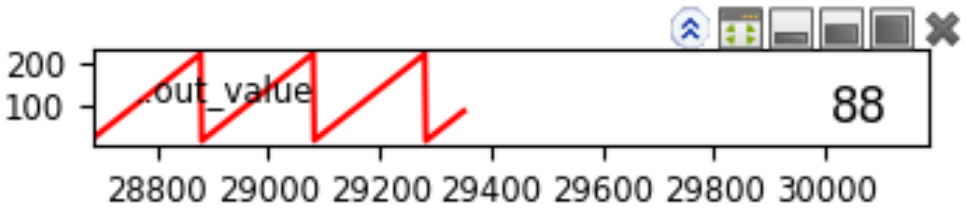


Рисунок 104 – График изменения значения переменной во время выполнения прикладной программы

На данной панели есть возможность изменять диапазон отображения значения.

Также на данной панели по наведению мыши отображаются вспомогательные кнопки. Описание кнопок приведено в табл. 12.

Таблица 12 – Кнопки на панели экземпляров проекта

Кнопка	Функции кнопки
	Вывод графика изменения значений переменной
	Сворачивание графика
	Изменяет подписи (масштаб) для оси «Y»
	Изменение размера окна с графиком
	Удаление переменной из отладки
 	Форсирование значения переменной Освобождение значения переменной

6.18. Режим выгрузки файла трасс на ИЭВМ

Режим выгрузки трассы в файл включается установкой соответствующего флага, рис. 105.

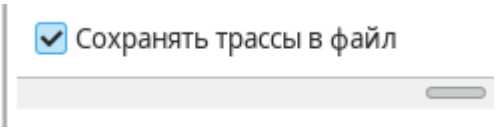


Рисунок 105 – Включение режима сбора трасс в файл

После активизации режима в директории traces в дереве проектов начинают формироваться файлы с трассами отслеживаемых переменных. Имя файла имеет вид `<временная_метка>.json`, например `2022-04-19T13:35:47.826858.json`. Каждый раз, когда происходит подключение к новому ПЛК или изменяется состав отслеживаемых переменных, создаётся новый файл. Формат экспорта протокола отладки приведён в Приложении Б.

6.19. Добавление дополнительных точек снятия трассы

По умолчанию съем трассы и значений переменных происходит единожды за тик выполнения ПЛК. В некоторых случаях этого может быть недостаточно для задач отладки. Для этого в СРиО добавлена поддержка механизма добавления дополнительных именованных точек снятия трасс. Эти точки оформляются в коде как Си-вставки следующего синтаксиса:

```
{breakpoint(“name”);}
```

Где: `name` – произвольное имя точки. Это имя будет отображаться в трассах, сохраняемых в файл (см. Прил. Б).

Пример использования точек остановки в коде ST приведён на рисунке 106.

```
1  IF Reset THEN
2    Cnt := ResetCounterValue;
3  ELSE
4    Cnt := ADD(Cnt,1);
5    {breakpoint("BP1 - start");}
6    Cnt := ADD(Cnt,5);
7    IF GT(Cnt,100) THEN
8      Cnt := ResetCounterValue;
9      {breakpoint("BP - if");}
10   END_IF;
11   {breakpoint("BP2 - fin");}
12 END_IF;
13
14 Out := ADD(Cnt,10);
15
```

Рисунок 106 – Использования именованных точек остановки в коде ST

6.20. Средства настройки параметров компиляции и сборки исполняемого образа для Багет ПЛК1

Панель проекта (см. пункт 6.5) содержит настройки параметров компиляции и сборки исполняемого образа для Багет ПЛК1.

В окне настроек проекта на вкладке «Конфигурация» доступна для выбора «Целевая платформа» «PLC1».

После выбора целевой платформы «PLC1» становится доступным блок параметров «Целевая платформа – PLC1» с возможностью указания параметров «Prefix», «CFLAGS», «LDFLAGS», «OSBASE», «PLCBASE», «Директория сборки ОСРВ Багет», «Команда сборки ОСРВ Багет», «Не включать в ПЛК средства отладки» (см. рис. 107).

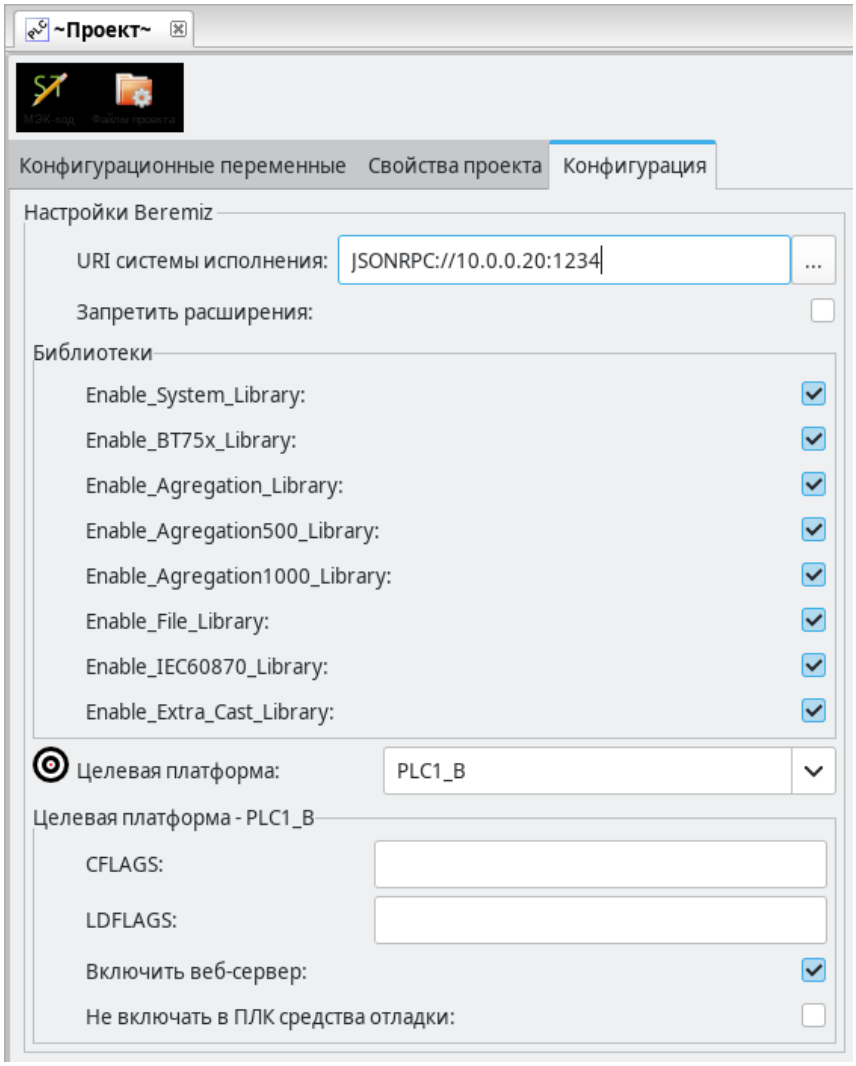


Рисунок 107 – Параметры компиляции и сборки исполняемого образа для Багет ПЛК1

6.21. Консоль компиляции и сборки

Функционал консоли компиляции и сборки в среде СРиО объединён с функционалом отладочной консоли (см. раздел 6.14). Среда разработки и отладки программ отображает в виде текстовых сообщений результаты компиляции и компоновки прикладных программ.

6.22. Панель симуляции программ в инструментальной среде

Панель проекта (см. пункт 6.5) позволяет выбрать режим симуляции программы.

В окне настроек проекта на вкладке «Конфигурация» доступна для выбора «Целевая платформа» «simulation».

После выбора целевой платформы «simulation» становится доступным блок параметров «Целевая платформа – simulation» с возможностью указания параметров «CFLAGS», «LDFLAGS», «Не включать в ПЛК средства отладки» (см. рис. 108).

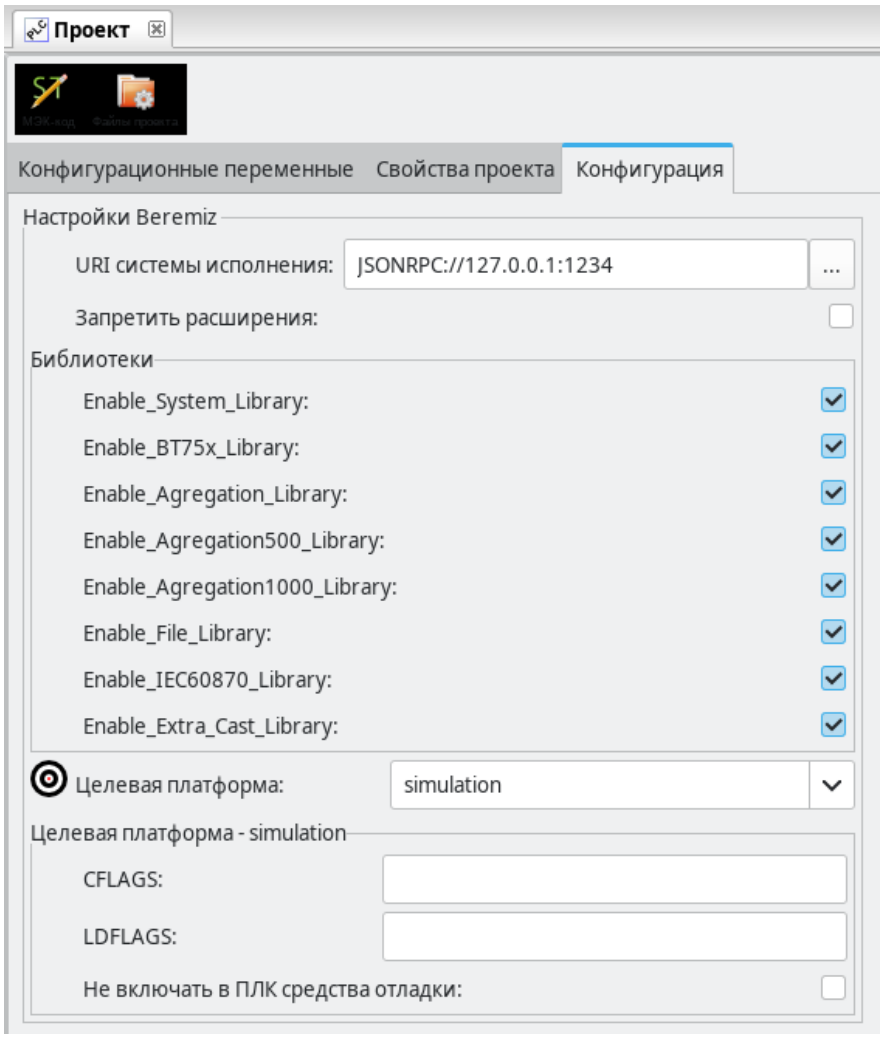


Рисунок 108 – Выбор режима симуляции

Далее производится очистка сборочной директории и выполняется сборка проекта (см. рис. 109).

Рисунок 109 – Сборка проекта

После выполнения сборки в каталоге проекта будет создан каталог build, будет создан бинарный файл с названием проекта. Данный файл должен быть запущен средствами ОС из консоли ОС. После этого можно выполнить подключение, рис. 110, добавить переменные к отладке (см. пункт 6.16) и запустить симуляцию, рис. 111.

Рисунок 110 – Подключение к целевому ПЛК

Рисунок 111 – Запуск ПЛК

Для симуляции будет доступна отладочная информация (см. пункт 6.16).

7. РАБОТА С ПРОЕКТОМ

Проект – это совокупность программных модулей (программ, функциональных блоков, функций), плагинов внешних модулей УСО, ресурсов, пользовательских типов данных, сборка (компиляция и компоновка) которых, представляет собой прикладную программу для целевого устройства (ПЛК). Ниже дано описание основных положений функционирования ПЛК под управлением приложения, разработанного в рамках исполнения проекта программы СРиО.

Основной цикл ПЛК (такт) – это постоянно повторяющийся временной период работы ПЛК между последовательными запусками процедуры выполнения задач.

В начале очередного такта происходит опрос внешних устройств. Полученные в результате опроса данные используются теми задачами, которые начали исполняться в текущем такте. Каждая задача имеет период. Период – это интервал времени, через который задача должна быть запущена снова. Периоды задач задаются при конфигурировании и составляют целое число микросекунд.

Длительность основного цикла (такта) задается автоматически и равняется наибольшему общему делителю периодов циклических задач проекта. Например, если периоды задач равны 30, 40, 100 мс соответственно, то такт равен 10 мс. Другой пример – периоды задач равны 3, 5, 7 мс соответственно, то такт равен 1 мс.

Задачи естественно упорядочены в порядке их описания в СРиО. На каждом такте задачи выполняются последовательно, одна за другой. Задачи выполняются в естественном порядке. После завершения последней задачи происходит передача на управляющие устройства полученных во время выполнения задач данных. Затем ПЛК простаивает до начала следующего такта.

Если одна (или несколько) задач не успевает завершиться (и начаться) до окончания такта, то выполнение этих задач продолжается до тех пор, пока все неисполненные задачи не завершатся. После завершения последней задачи ПЛК простаивает до начала следующего такта.

Приложение может включать несколько ресурсов. Ресурсы естественно упорядочены в порядке их описания в конфигураторе. Ресурсы последовательно выполняются в естественном порядке, задаваемом при конфигурировании.

В данном разделе рассмотрены основные приёмы работы в среде разработки и отладки программ, которые необходимы при создании прикладной программы.

7.1. Создание нового проекта

Новый проект создаётся с помощью главного меню «Файл» – «Новый», рис. 112, либо с помощью кнопки «Новый» на панели управления.

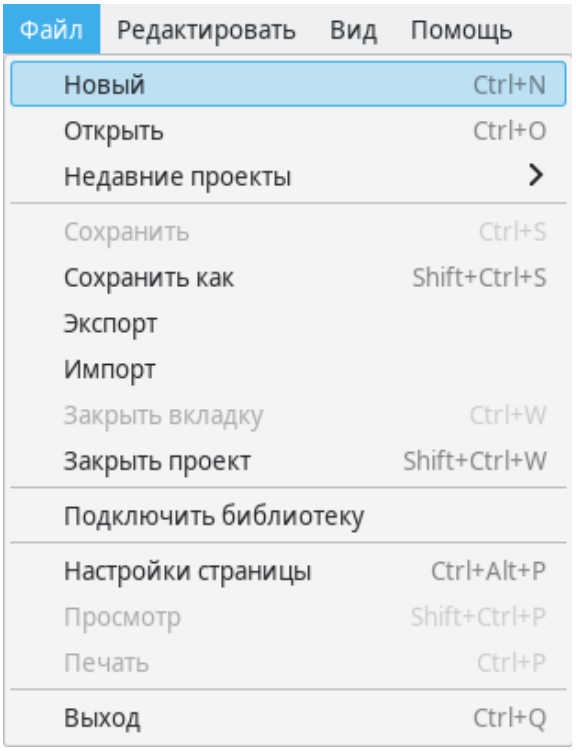


Рисунок 112 – Создание нового проекта с помощью главного меню

В открывшемся окне «Выберите путь для нового проекта», рис. 113, указывается полное название создаваемых каталога и имени проекта. По умолчанию к имени файла проекта будет добавлено расширение «`.bgt`». Проект в каталоге хранится в виде файла, который представляет собой ZIP-архив.

Каталог должен быть обязательно пустым и не защищён от записи. Если в каталоге уже есть файлы, будет выдана соответствующая ошибка. В созданном каталоге будут сохранены следующие файлы и каталоги:

– «`beremiz.xml`» – в данном XML файле сохраняются настройки специфичные для среды разработки Beremiz относительно проекта;
– «`plc.xml`» – в данном XML файле сохраняется полное описание проекта: всех программных модулей, ресурсов, пользовательских типов данных, данных
о проекте, настроек редакторов графических языков IEC 61131-3;
– каталоги с настройками плагинов внешних модулей УСО.

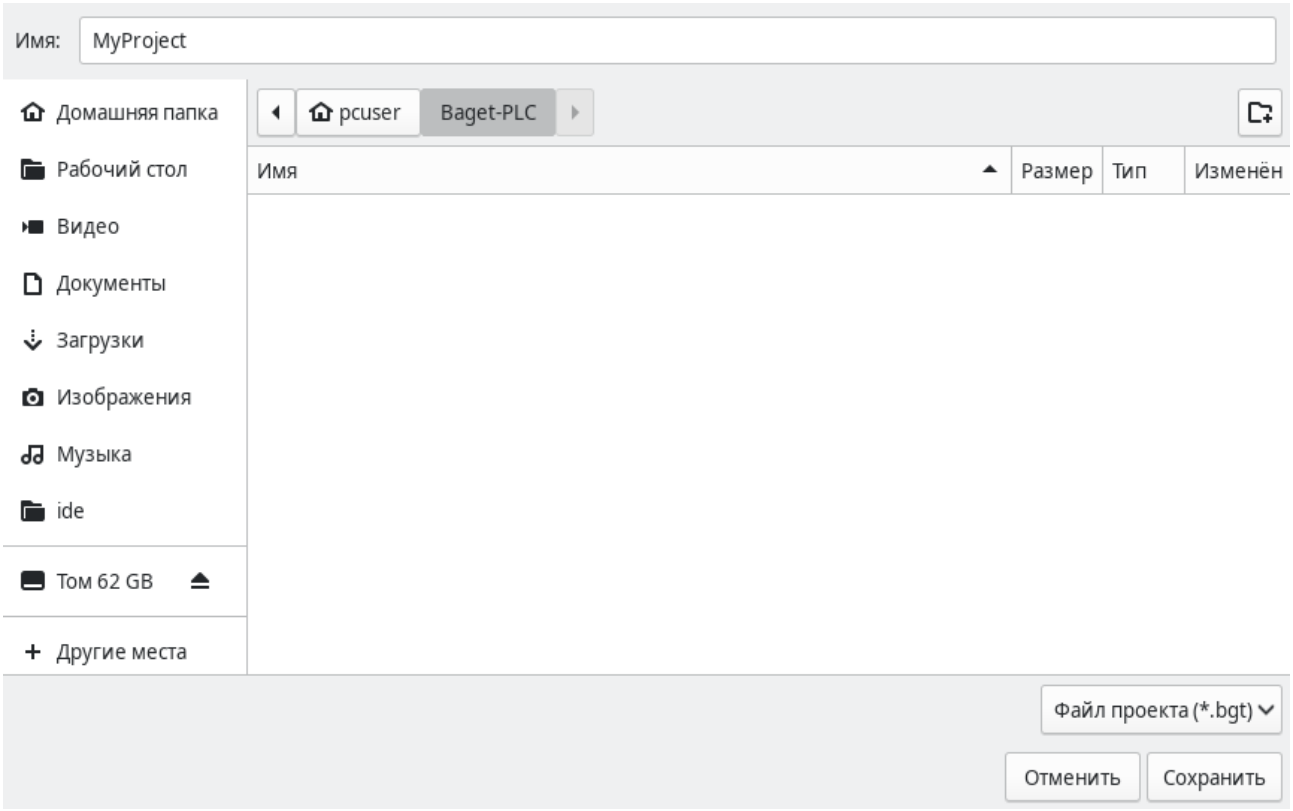


Рисунок 113 – Диалог выбора каталога для нового проекта

При сборке проекта во временном каталоге проекта «/tmp/Beremiz_<XXXXX>/имя проекта» (где <XXXXX> - случайные смиволы), который создаётся и существует только на время работы с проектом в среде Beremiz, создаётся подкаталог «build», в котором хранится генерируемый ST и C код, а также получаемый исполняемый бинарный файл.

При активации кнопки «Сохранить» проект будет сохранен в указанный каталог с заданным именем.

Далее появится диалог «Создать новый POU», в котором можно выбрать имя программы, язык МЭК, либо отменить создание программного модуля, активировав кнопку «Отменить». Более подробно работа с диалогом «Создать новый POU» описана в п. 7.5.

7.2. Импорт проекта

Текущий проект можно заполнить информацией, импортируемой из файла формата XML, в котором хранятся данные другого проекта. Для этого необходимо после открытия/создания проекта активировать пункт главного меню “Файл-Импорт”. В результате откроется диалог, представленный на рис. 113.

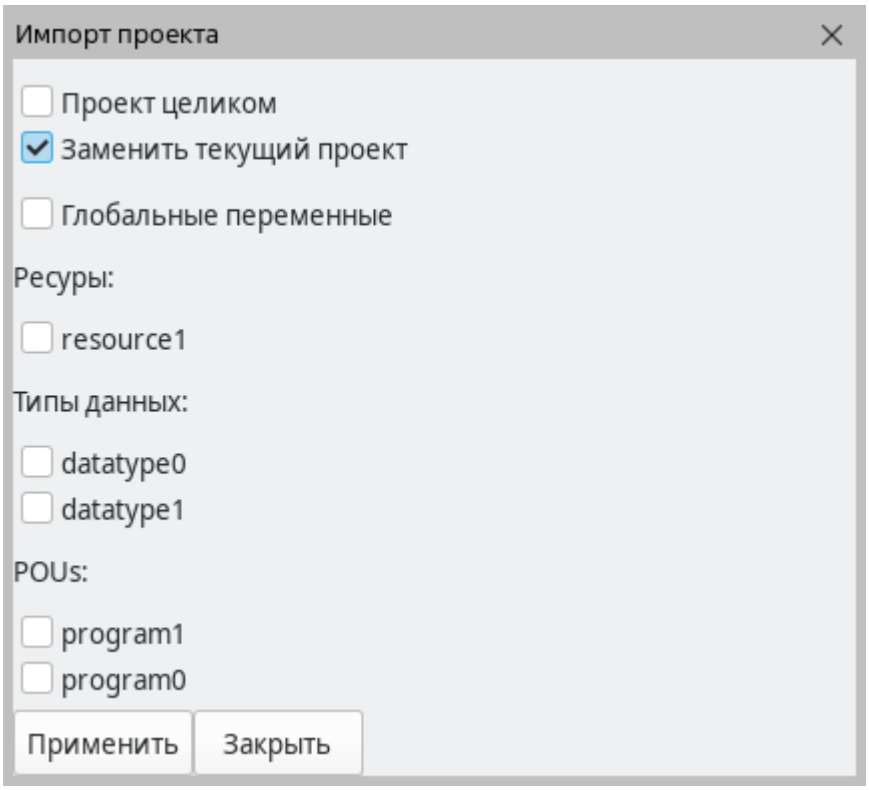


Рис. 114 – Диалог импорта проекта

Импорт проекта возможен, как целиком, так и отдельно можно импортировать глобальные переменные, ресурсы, типы данных и POU-модули. Если активировано поле “Заменить текущий проект”, то старая информация проекта удаляется, а затем в проект добавляется информация импортируемого файла.

7.3. Экспорт проекта

Данные текущего проекта можно экспортировать в файл формата XML. Для этого необходимо после открытия/создания проекта активировать пункт главного меню “Файл – Экспорт”. В результате откроется диалог, пример которого представлен на рис. 115115115115115115115115115115.115115115

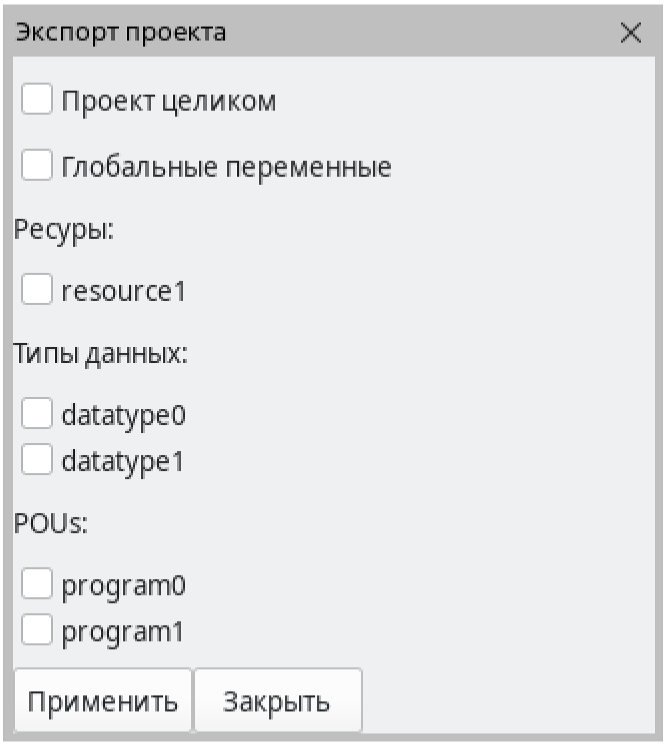


Рисунок 115 – Диалог экспорта проекта

Программой предоставляется возможность экспорта как проекта целиком, так и отдельно можно экспортировать глобальные переменные, ресурсы, типы данных и POU-модули.

7.4. Настройка проекта

Как правило, первым шагом после создания проекта является его настройка, включающая в себя задание глобальных переменных, установку параметров компиляции и компоновки, а также заполнение данных о проекте. Вызов панели настройки проекта осуществляется при выборе (двойном щелчке левой кнопкой мыши) корневого элемента дерева проекта, который по умолчанию, сразу после создания проекта называется «`Unnamed`», выделено красным на рис. 116.

Примечание: в текущей реализации СРиО, автоматически предлагается название переменной «`LocalVar0`». Рекомендуется изменить его, например, на «`globalValue`».

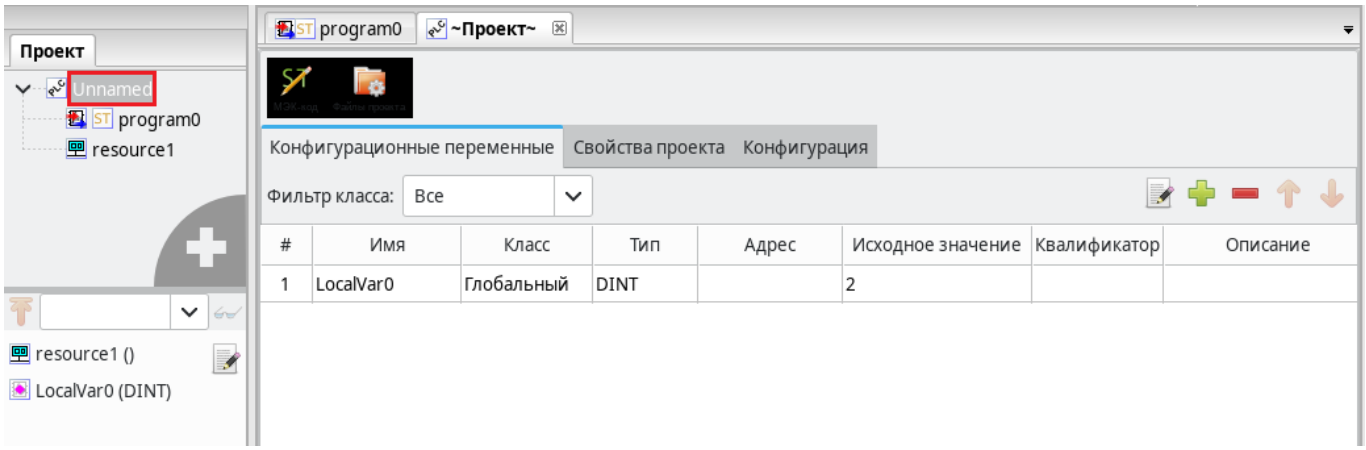


Рисунок 116 – Вызов панели настройки проекта с помощью корневого элемента структуры проекта

7.4.1. Установка пароля проекта

Для задания пароля проекта необходимо активизировать кнопку «Пароль проекта», (см. табл.2). Будет вызван диалог: «Задайте пароль проекта», рис. 117).

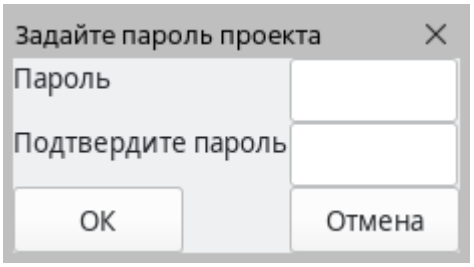


Рисунок 117 – Диалог задания пароля проекта

При задании пароля должны соблюдаться следующие требования:

– пароль должен содержать не меньше 10 символов;
– пароль должен содержать знаки хотя трех из четырех категорий (цифры, буквы, символы разного регистра, специальные символы);
– пароль не должен содержать последовательности повторяющихся символов, либо символов, последовательно находящихся на клавиатуре.

В случае нарушения указанных требований будет выдано сообщение «Пароль недостаточно надежен!», рис.118.

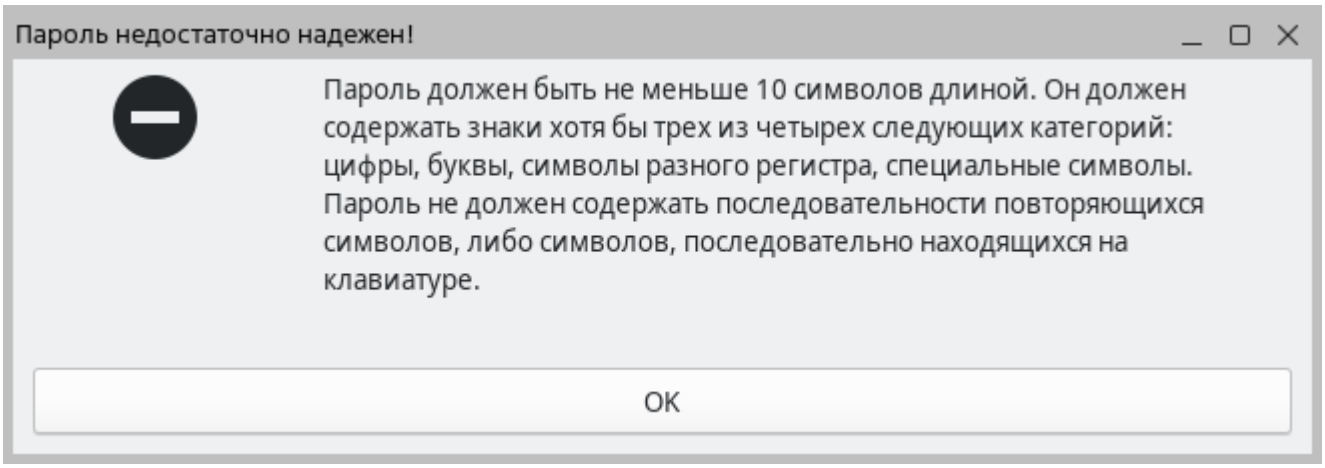


Рисунок 118 – Сообщение о нарушении требований к создаваемому паролю

7.4.2. Глобальные переменные проекта

Глобальные переменные позволяют программным модулям общие переменные, которые будут определены в глобальной области видимости проекта.

Ниже, на рис. 119, в панели переменных и констант создана глобальная переменная «`globalValue`» с начальным значением 100, с помощью кнопки «`Добавить переменную`» (см. таблицу 3).

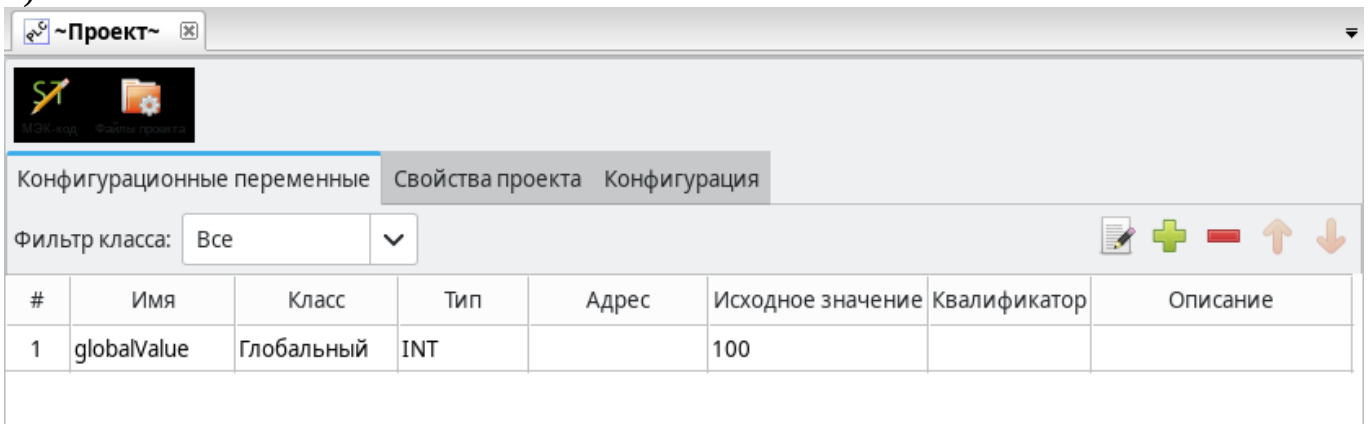


Рисунок 119 – Объявление глобальной переменной проекта

Для того чтобы к данной глобальной переменной можно было обращаться из программных модулей необходимо в их панели редактирования в панели переменных и констант создать переменную с таким же именем, как и ранее объявленная глобальная, и установить её класс «Внешний». На рис. 120 приведён пример объявления в программном модуле «program0» переменной «`globalValue`» класса «Внешняя», типа INT и изменение её значения с помощью языка ST.

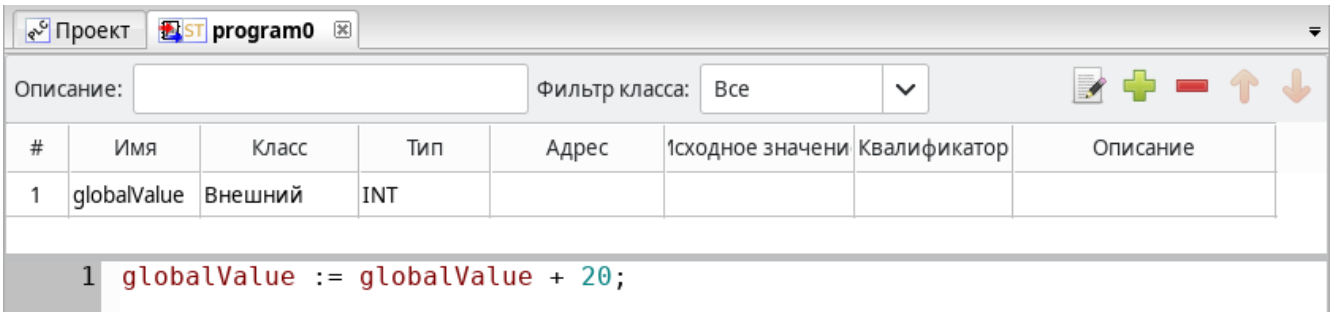


Рисунок 120 – Использование глобальной переменной в программном модуле

На рис. 121 в программном модуле «program1» также определена переменная «globalValue» класса «Внешняя» и изменение её значения с помощью языка ST.

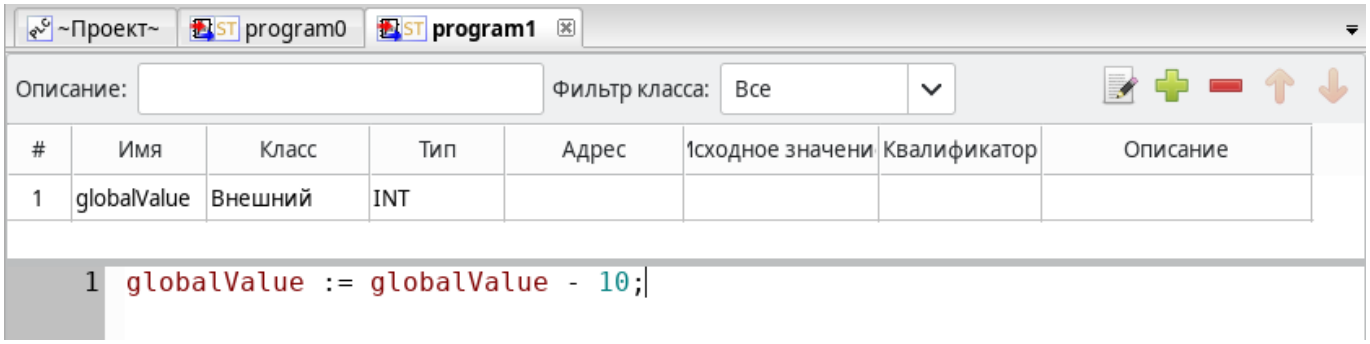


Рисунок 121 – Использование глобальной переменной в программном модуле

Соответственно, переменная «globalValue» будет изменяться во время выполнения в двух программных модулях: «program0» и «program1».

7.4.3. Настройки сборки проекта и соединения с целевым устройством

Для использования написанной прикладной программы необходимо её собрать (скомпилировать и скомпоновать), то есть получить исполняемый файл и передать на целевое устройство для отладки или просто исполнения. В связи с этим основными настройками являются: «URI системы исполнения» целевого устройства и целевая платформа, указывающая архитектуру платформы целевого устройства (см. рис. 122).

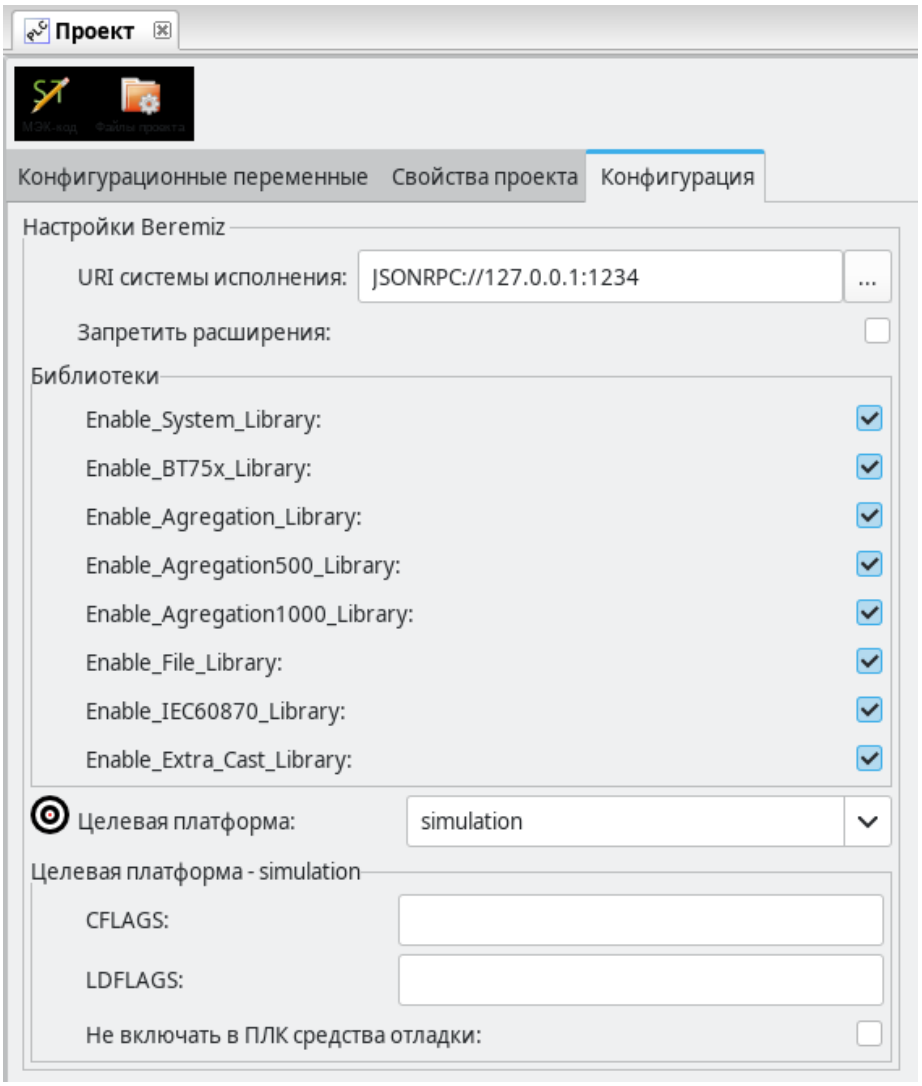


Рисунок 122 – Выбор целевой платформы для сборки прикладной программы

Как правило, «URI адрес» указывается в формате:

`JSONRPC://<IP-адрес целевого устройства>:<порт>`

IP адрес целевого устройства должен быть известен и необходима возможность подключения к нему по локальной сети (требуется обратить внимание на сетевые настройки операционной системы). По умолчанию порт имеет номер 1234, изменить номер порта можно указанием дефайна BPP_PORT в поле CFLAGS настроек проекта, например –DBPP_PORT=8888.

Значение целевой платформы должно быть равно «PLC1», если целевое устройство ПЛК «Багет-ПЛК1», и «simulation» для отладки с использованием сервиса симуляции СС ИЭВМ без загрузки СПО на ПЛК.

7.4.4. Данные о проекте

При создании нового проекта, все обязательные поля в настройках информации о проекте заполняются значениями по умолчанию. Рекомендуется заменить данные настройки по умолчанию на релевантную информацию, рис. 123, позволяющую удобным образом различать проекты.

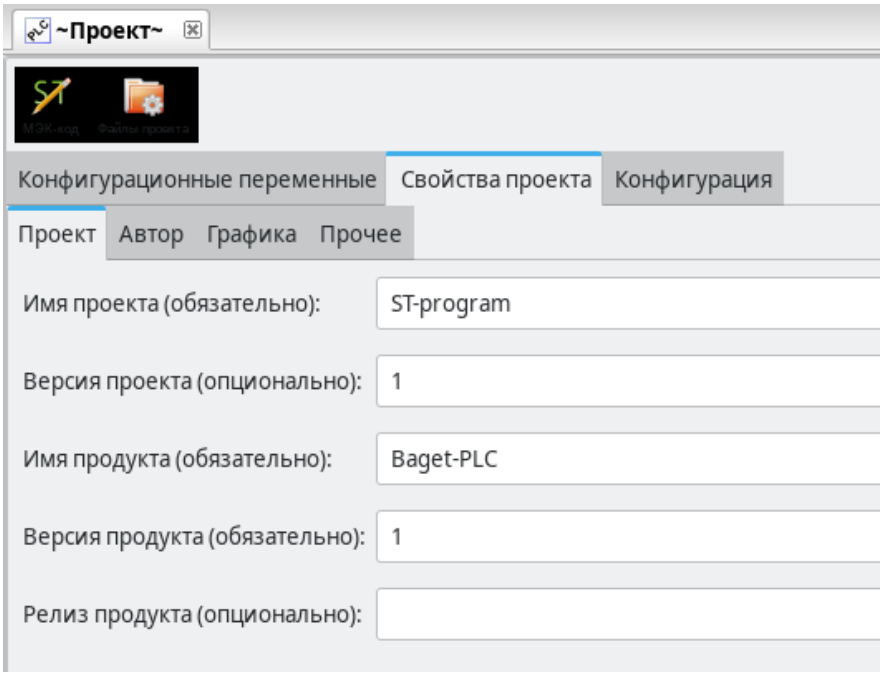


Рисунок 123 – Указание данных о проекте

Большая часть данных в информации проекте являются необязательным для заполнения, но некоторые должны быть всегда заполнены. Это указывается в подсказках в именовании каждого пункта.

После задания настроек проекта, как правило, следует добавление в проект необходимых программных модулей (функций, функциональных блоков и программ), реализация их алгоритмов и логики работы с помощью текстовых и графических языков стандарта МЭК 61131-3-2016.

7.5. Программные модули

Добавление программных модулей (программ, функций, функциональных блоков (POU)) осуществляется с помощью всплывающего меню структура проекта, в котором необходимо выбрать пункт «Функция», «Функциональный блок» или «Программа». Далее появится диалог «Создать новый POU», рис. 124.

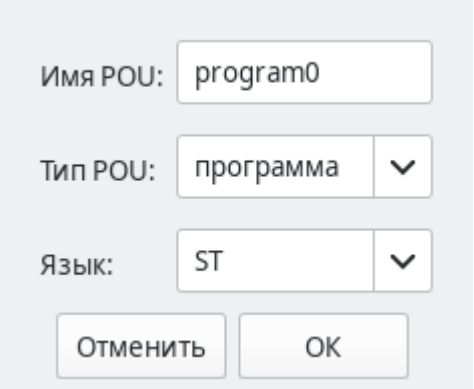


Рисунок 124 – Диалоговое окно «Создать новый POU»

Имя, присвоенное по умолчанию, может быть заменено на имя, соответствующее назначению данного программного модуля. В зависимости от того, какой программный модуль был выбран во всплывающем меню, в поле «Тип POU» будет подставлено именование данного программного модуля.

В поле «Язык», необходимо выбрать из списка один из языков стандарта МЭК 61131-3-2016 (IL, ST, LD, FBD, SFC), на котором будет реализованы алгоритмы и логика работы программного модуля, рис. 125.

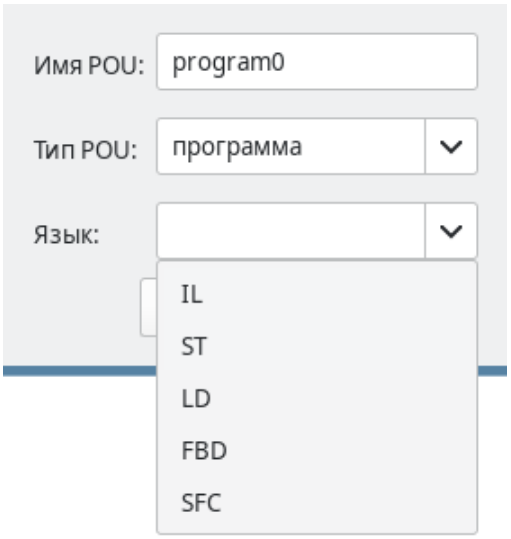


Рисунок 125 – Выбор языка для программного модуля

Далее рассмотрено добавление каждого программного модуля в отдельности.

7.5.1. Программа

Ниже будет приведён пример добавления в проект программы, написанной на языке FBD. Логика и алгоритм работы данного программного модуля следующие: определены две глобальные переменные «globalValue» и «globalLevel», если значение «globalValue» больше 10.0, то присвоить переменной «globalLevel» значение 100, в противном случае присвоить «globalLevel» значение 50.

Сначала следует добавление программы в проект, осуществляемое с помощью меню структура проекта, выбором пункта «Программы», выделено красным на рис. 126.

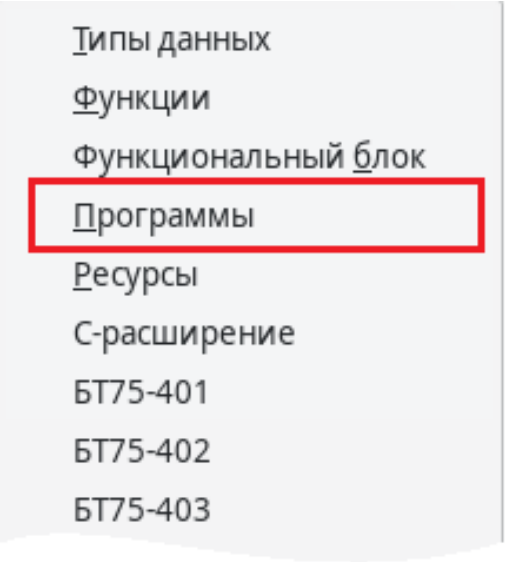


Рисунок 126 – Выбор в структуре проекта добавления программы

В появившемся диалоге (см. рис. 127) выбирается язык FBD и нажимается кнопка «OK».

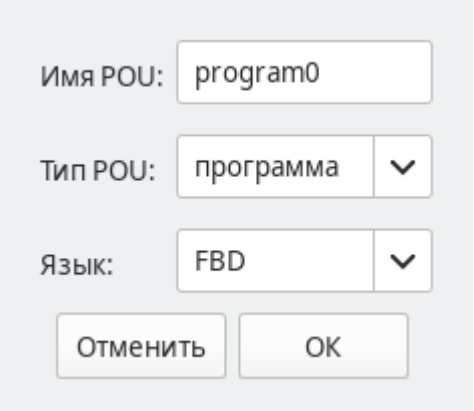


Рисунок 127 – Выбор языка FBD

Далее в открывшейся вкладке с панелью редактирования данного программного модуля в панели переменных и констант (см. рис. 128) добавляются переменные: «globalValue» типа REAL, класса «Внешний» и «globalLevel» типа INT, класса «Внешний».

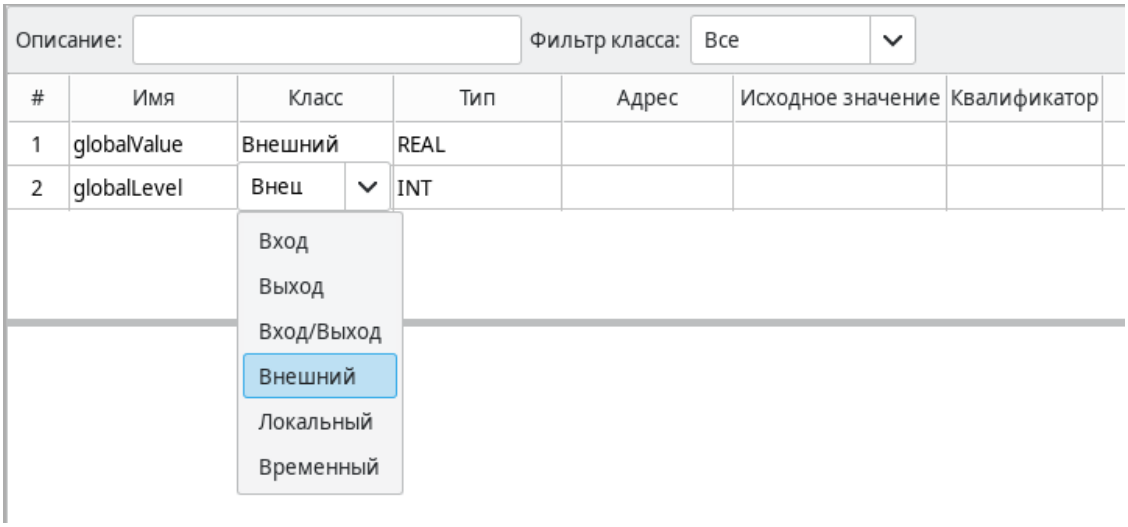


Рисунок 128 – Объявление в программе внешних переменных

Предполагается, что эти переменные уже определены глобальными переменными проекта в панели редактирования проекта, рис. 129, как описывается в п. 7.4.1.

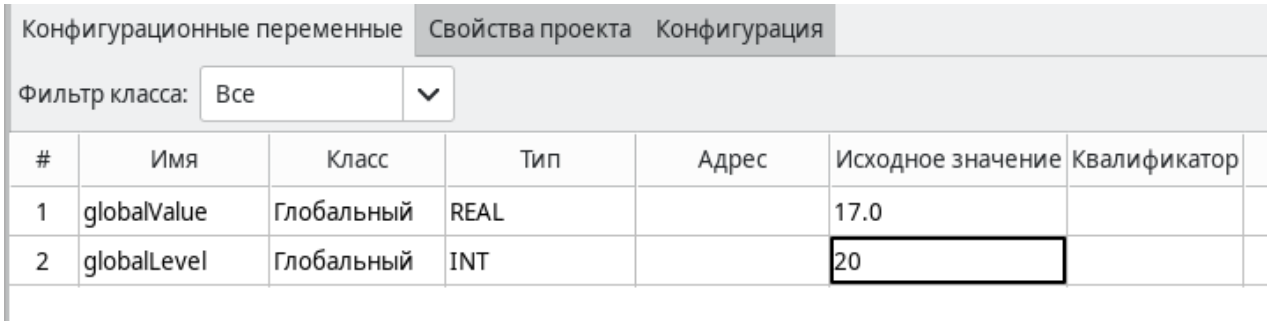


Рисунок 129 – Глобальные переменные проекта

Далее необходимо обратиться к редактору языка FBD. Для написания алгоритма и логики выполнения данной программы будут добавлены две функции: «GT» и «SEL». Функция «GT» обозначает сравнение «Больше чем» и находится во вкладке «Сравнение». Она может содержать от 2 до 20 входных значений (в данном примере их будет 2) и одно выходное значение «OUT». Если значение «IN1» больше значения «IN2», то на выходе «OUT» будет TRUE, в противном случае FALSE.

Функция «SEL» обозначает «Выбор одного из двух значений» и находится во вкладке «Операции выбора». Она содержит три входных переменных «G», «IN0», «IN1» и одну выходную «OUT». Если «G» равно 0 (или FALSE), то выходной

переменной «OUT» присваивается значение «IN0». Если «G» равно 1 (или TRUE), то выходной переменной «OUT» присваивается значение «IN1».

Добавление данных функций удобнее осуществить переносом соответствующей функции с помощью мыши (Drag&Drop) из панели библиотеки функций и функциональных блоков в область редактирования FBD диаграммы данного программного модуля (см. рис. 130).

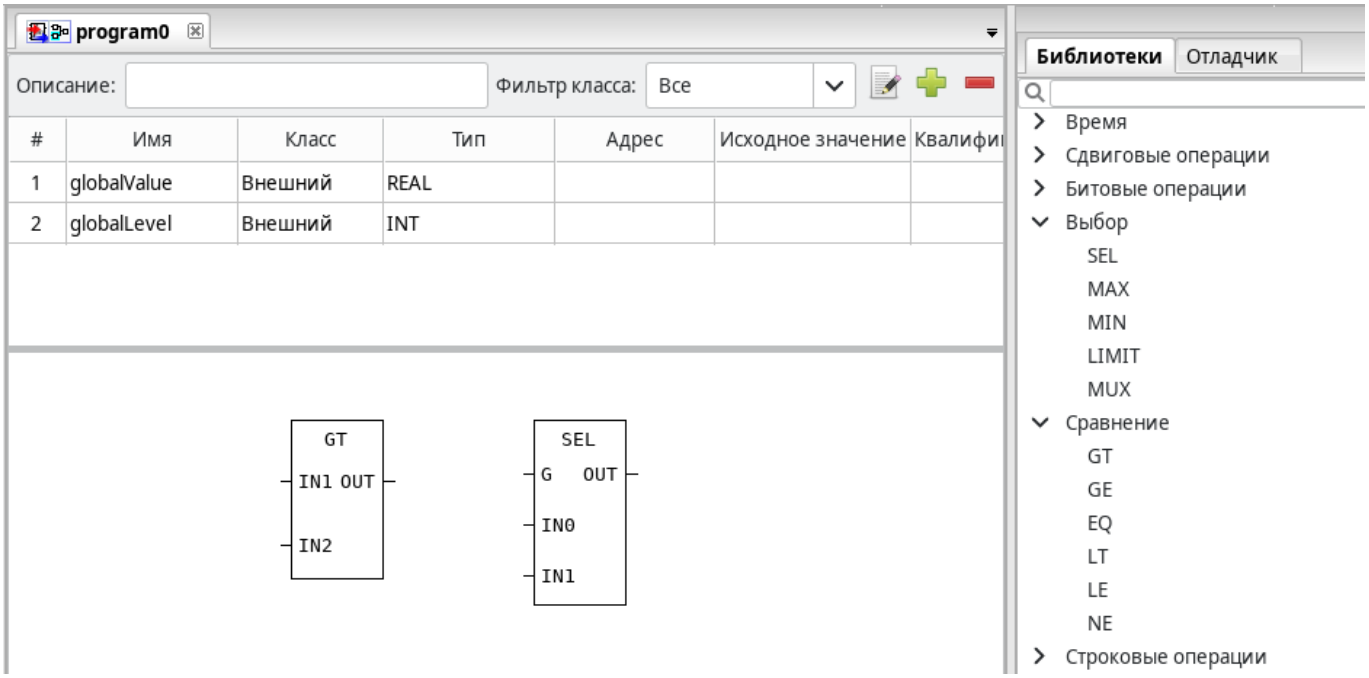


Рисунок 130 – Добавление двух функций на FBD диаграмму

Далее, также используя мышь, переносятся переменные «globalValue» и «globalLevel» на FBD диаграмму. Как уже упоминалось ранее (см. п. 6.9.1.2), необходимо левой клавишей мыши зажать столбец «#» для переменной в панели переменных и констант, далее перенести указатель на область редактирования FBD диаграммы и отпустить кнопку мыши (Drag&Drop), выделено красным на рис. 131. Такую манипуляцию нужно произвести для обоих переменных.

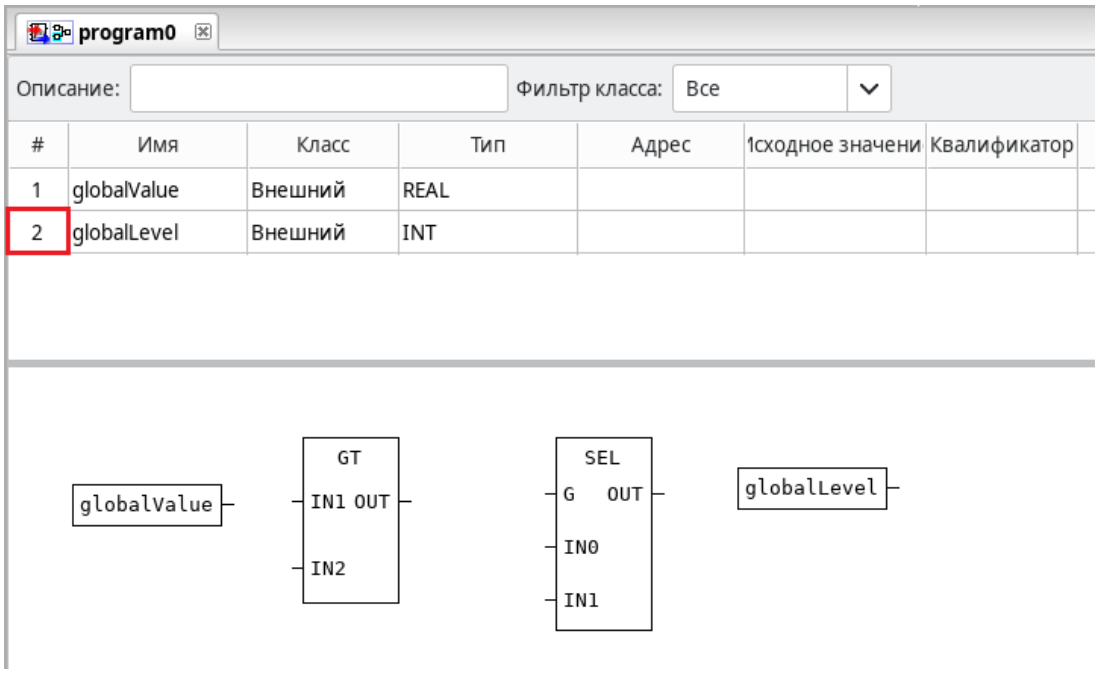


Рисунок 131 – Перенесённые переменные на FBD диаграмму

Переменную «globalValue» можно соединить с входом «IN1» функции «GT». Чтобы переменную «globalValue» можно было соединить с выходом «OUT» функции «SEL», необходимо сделать двойной щелчок левой кнопкой мыши по «globalLevel» на FBD диаграмме и в появившемся диалоге «Свойства переменной» указать класс «Выход», выделено красным на рис. 132.

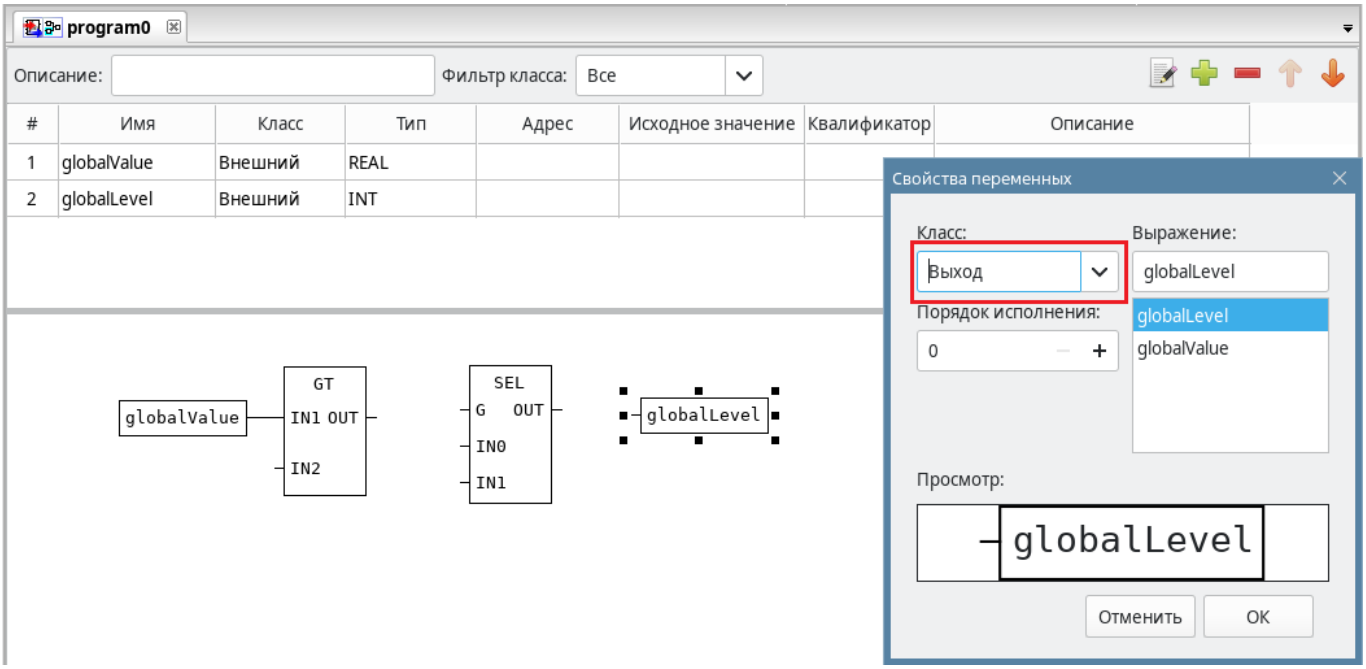


Рисунок 132 – Указание класса «Выход» для переменной

После этого соединить переменную «globalLevel» с выходом «OUT» функции «SEL», и выход «OUT» функции «GT» с входом «G» функции «SEL», рис. 133.

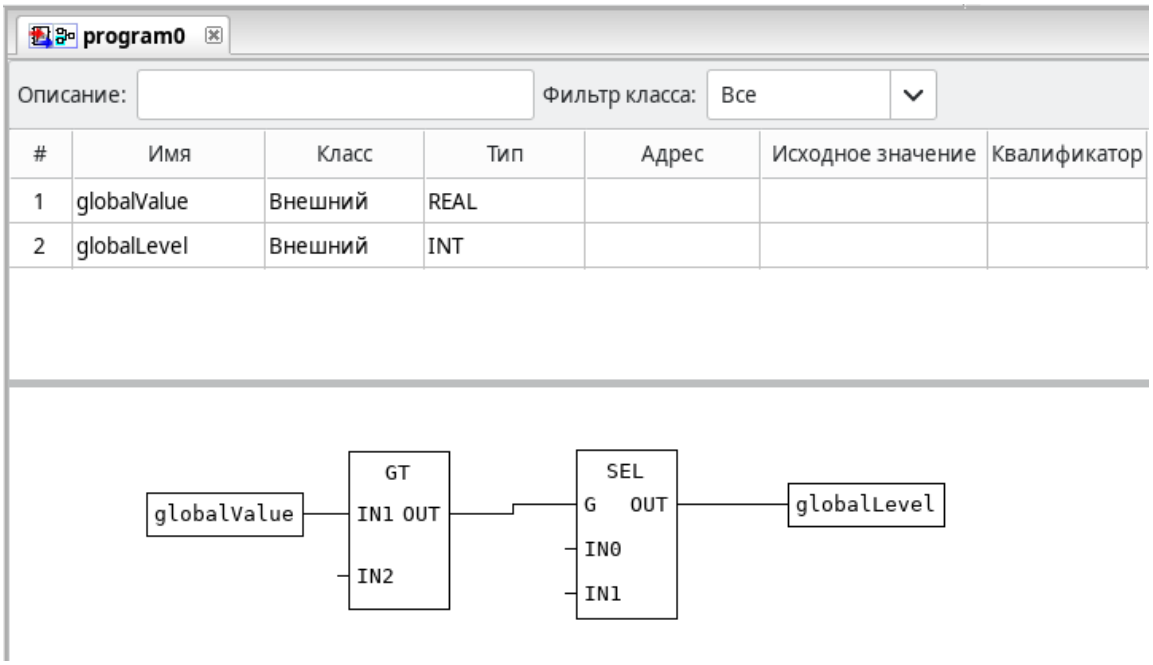


Рисунок 133 – Соединение входов и выходом функций на диаграмме FBD

Далее необходимо добавить 3 числовых литерала, которые будут напрямую соединены с входом «IN2» функции «GT» и входами «IN0» и «IN1» функции «SEL». В панели редактирования FBD диаграммы выбирается кнопка «Добавить переменную» (см. таблицу 6) и в появившемся диалоге «Свойства переменных» в поле выражения пишется «10.0», рис. 134. Нажимается кнопка «OK».

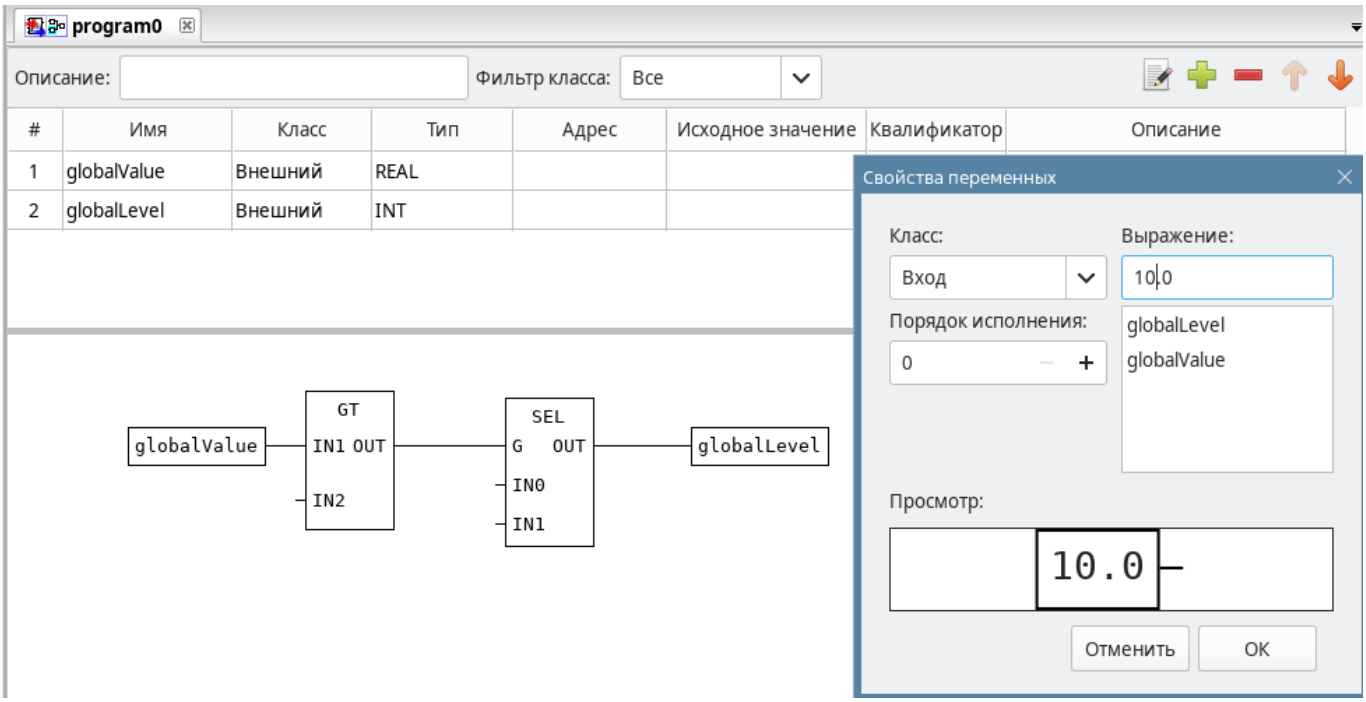


Рисунок 134 – Добавление переменной на FBD диаграмму

Добавленный литерал соединяется с входом «IN2» функции «GT», как показано на рис. 135.

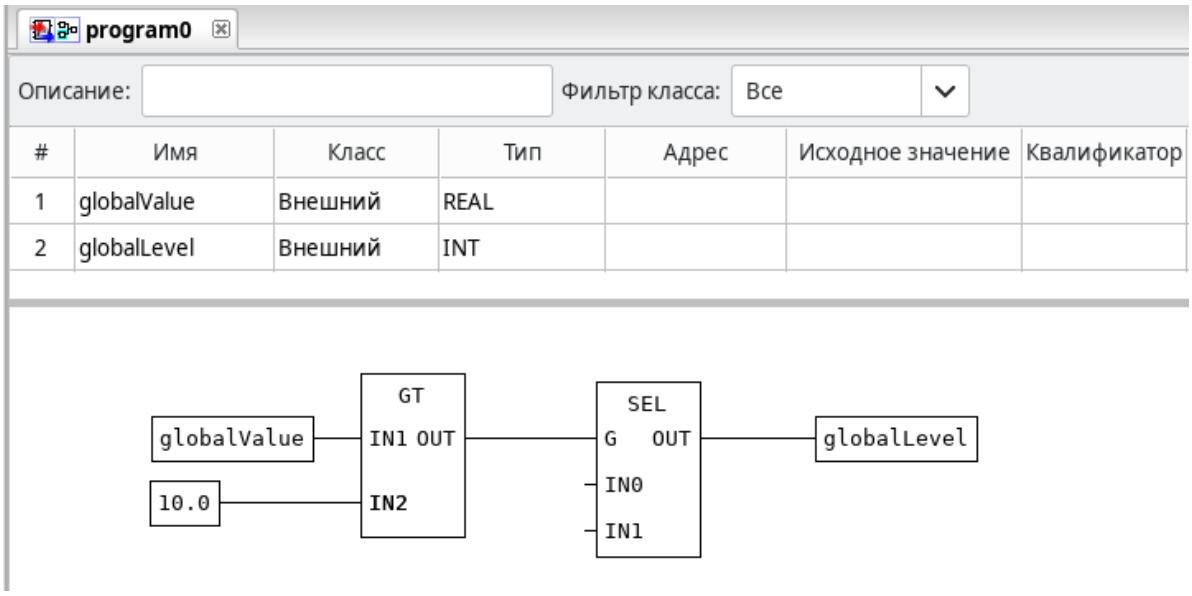


Рисунок 135 – Соединение добавленной переменной с выходом функции

Аналогичным образом создаются литералы со значениями 50 и 100 для входов «IN0» и «IN1» функции «SEL» и соединяются с ними, рис. 136.

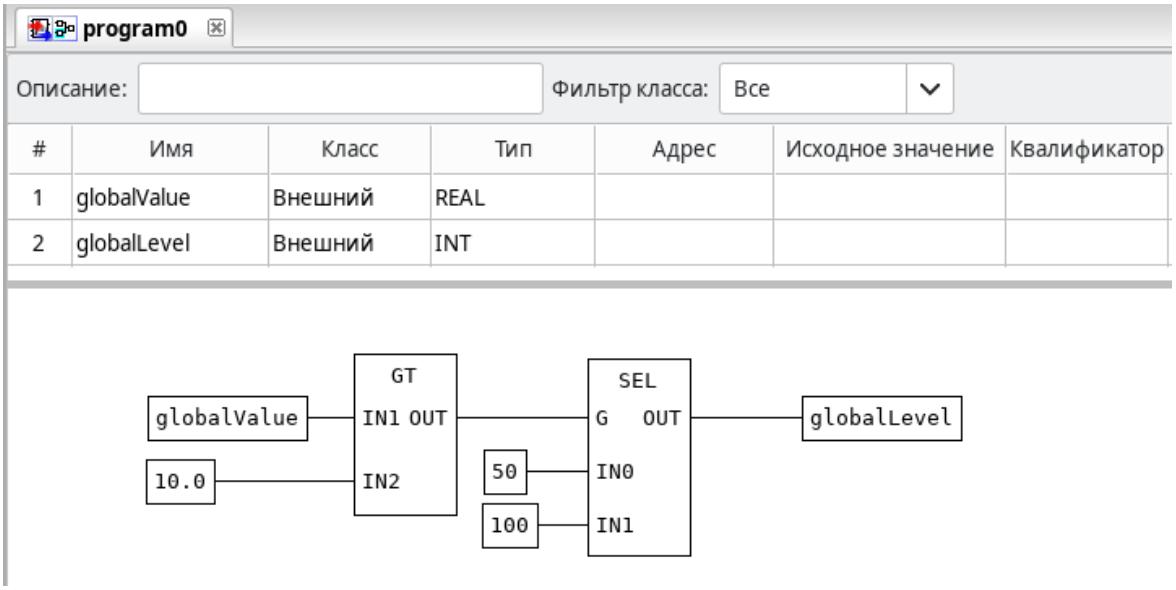


Рисунок 136 – Пример программного модуля, написанного на языке FBD

Соответственно, в случае если значение переменной «globalValue» больше 10.0, то на выходе «OUT» функции «GT» будет значение TRUE, тем самым на вход функции «SEL» поступит тоже True и выходное значение «OUT» будет равно «IN1», т.е. 100.

Далее будет рассмотрен пример добавления программного модуля типа «Функция».

7.5.2. Функция

Ниже будет приведён пример добавления в проект функции, и её использование в программном модуле типа «Программа». Добавление осуществляется с помощью меню структура проекта, выбором пункта «Функции», выделено красным на рис. 137.

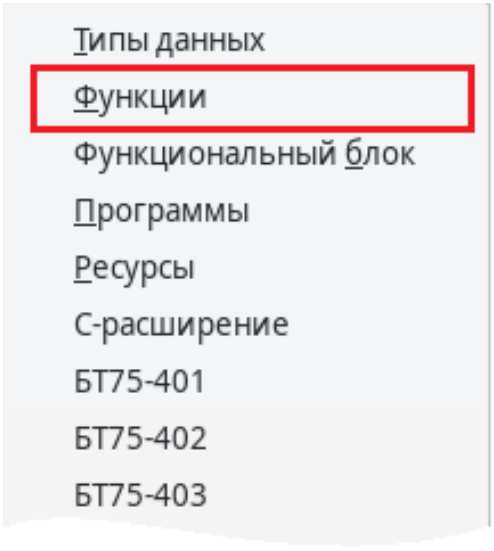


Рисунок 137 – Выбор в структуре проекта добавления функции

В появившемся диалоге «Создать новый POU» в поле «Имя программного модуля» укажем имя «GetStatus» и выберем «Язык» ST из списка языков стандарта IEC 61131-3 (см. рис. 138). Язык SFC не может быть использован для описания алгоритма и логики работы функции.

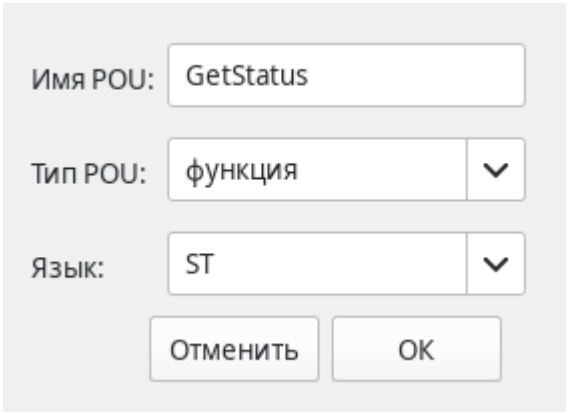


Рисунок 138 – Добавление пользовательской функции

В появившейся панели редактирования функции выберем тип возвращаемого значения функции – STRING (см. рис. 139).

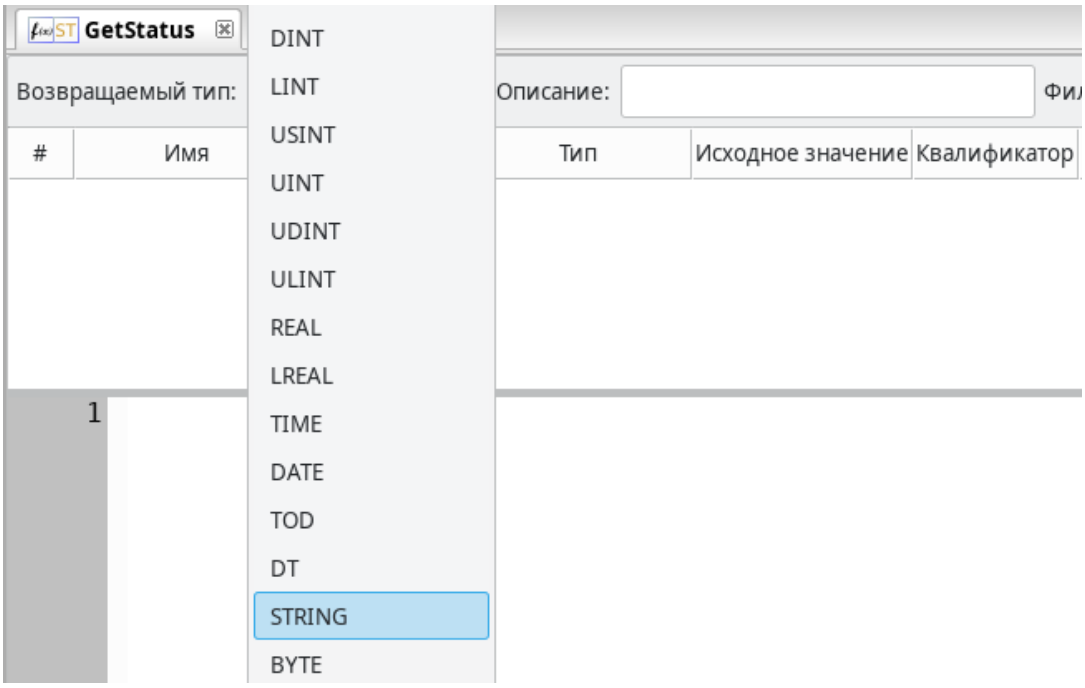


Рисунок 139 – Выбор возвращаемого значения функцией

Далее в панели переменных и констант указываются переменная «value» (хотя переменных может быть несколько) класса «Вход». В редакторе языка ST пишется алгоритм и логика работы данной функции, как показано на рис. 140.

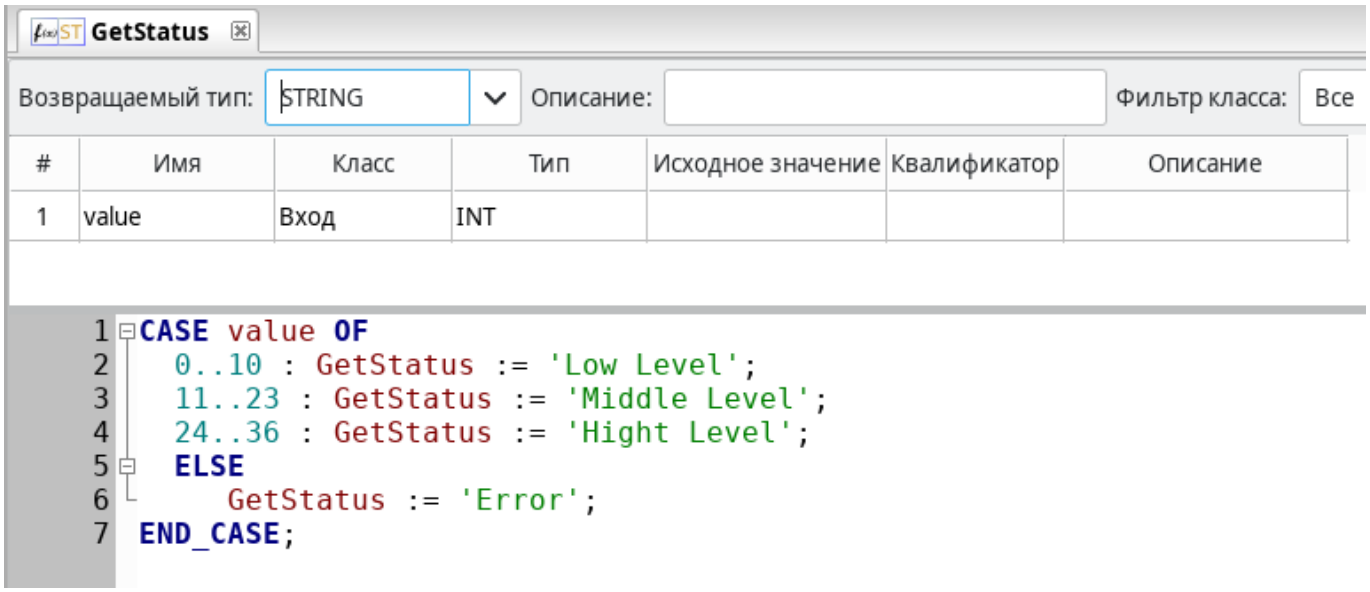


Рисунок 140 – Определение функции алгоритма и логики выполнения функции

С помощью блока CASE…OF определяются возвращаемые значения функции в зависимости от значения «value». Если ни одно из 3 условий не подходит, то возвращается «Error».

Далее необходимо создать программный модуль «Программа» на языке FBD и в появившейся панели его редактирования добавить две переменные «in_value» и «out_status», как показано на рис. 141.

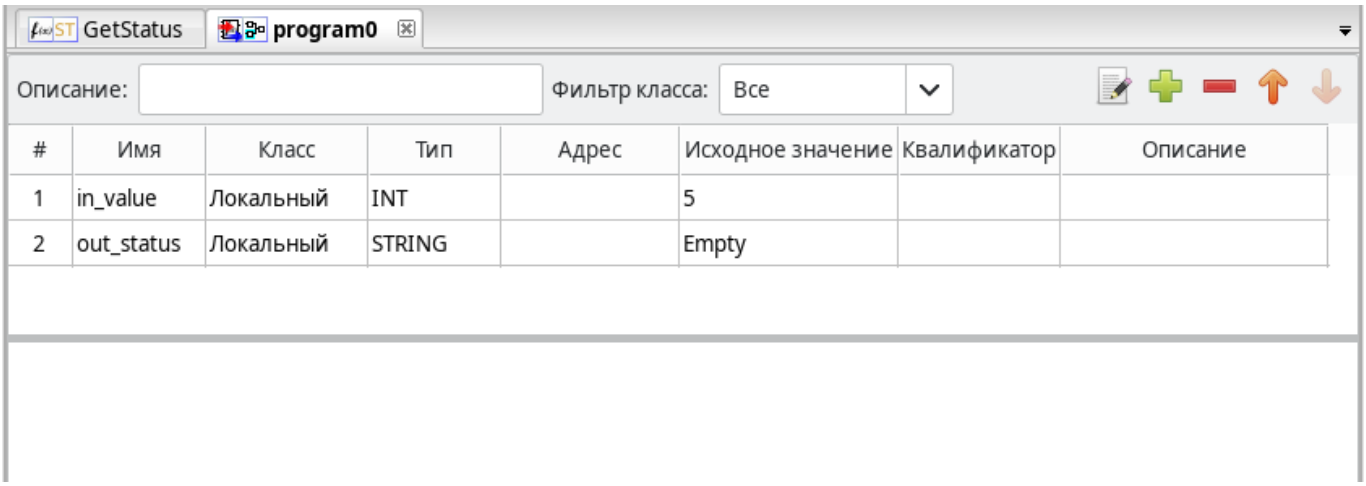


Рисунок 141 – Добавленный программный модуль «Программа» на языке FBD

На панели «Библиотеки» в разделе «Пользовательские POU» необходимо выбрать функцию «GetStatus» и с помощью указателя мыши (зажав левую кнопку мыши) перенести данную функцию (Drag&Drop) в область редактирования FBD диаграммы программного модуля «program0», выделено красным на рис. 142.

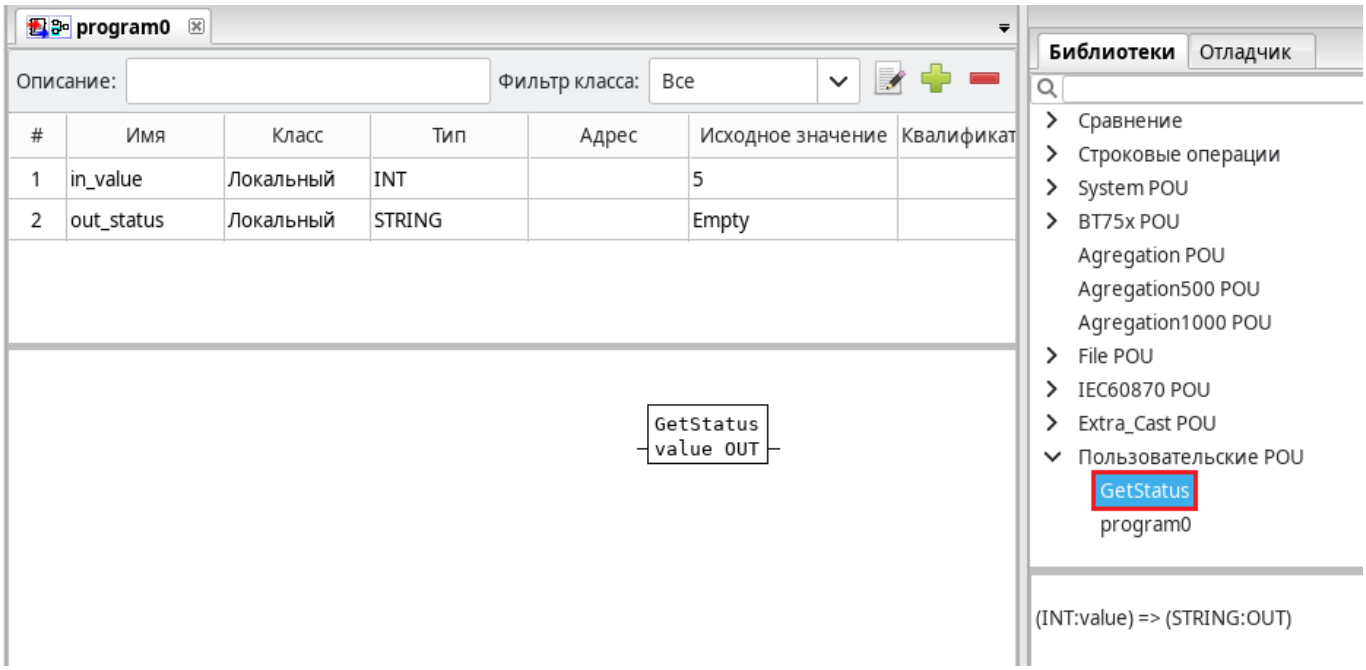


Рисунок 142 – Добавление на FBD диаграмму пользовательской функции

Аналогичным образом «перетаскиваются» переменные из панели переменных и констант в область редактирования FBD диаграммы, выделено красным на рис. 143.

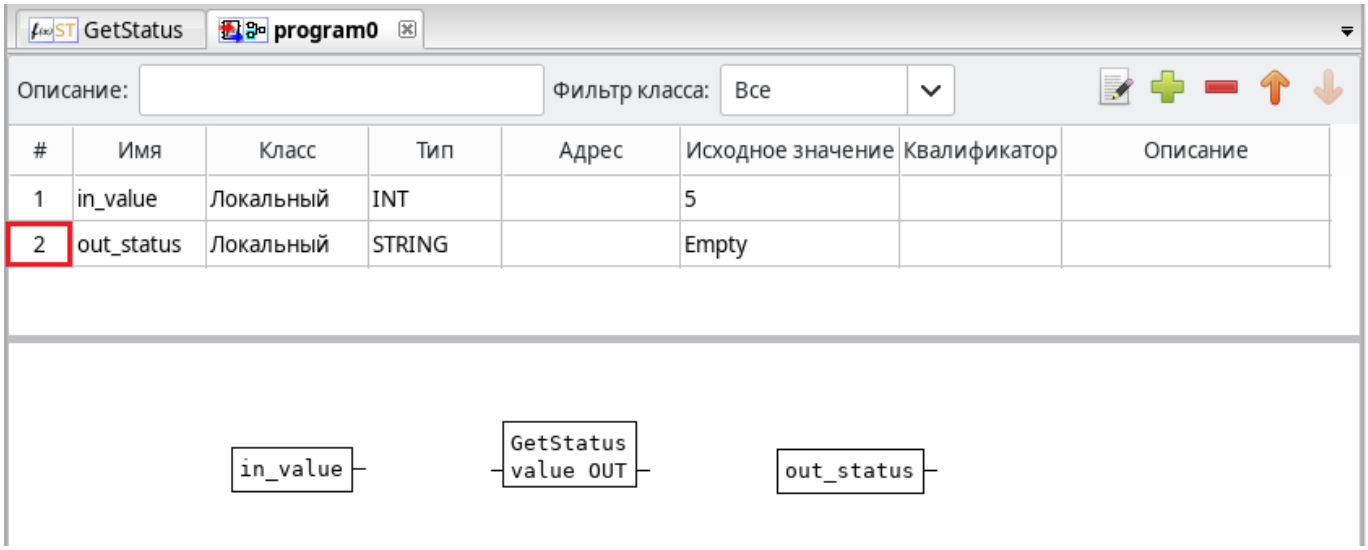


Рисунок 143 – Добавление на FBD диаграмму переменных
из панели переменных и констант

Для того чтобы переменную «out_status» можно было соединить с «выходом» функции «GetStatus», необходимо в диалоге свойств данной переменной (который вызывается двойным щелчком левой кнопки мыши по переменной в области редактирования FBD диаграммы) указать класс «Выход», рис. 144.

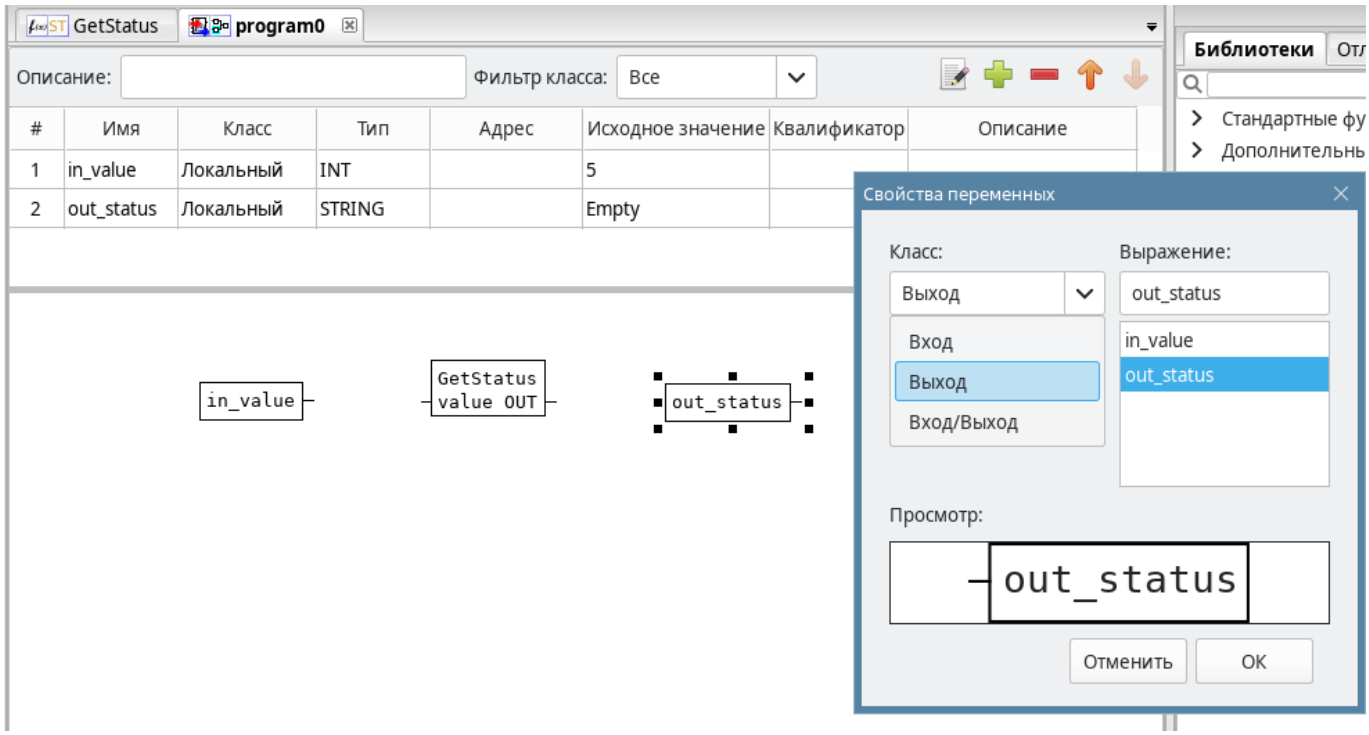


Рисунок 144 – Изменение «класса» переменной FBD диаграммы

Теперь переменные «in_value» и «out_status» можно соединить, соответственно, с «входом» и «выходом» функции «GetStatus», рис. 145.

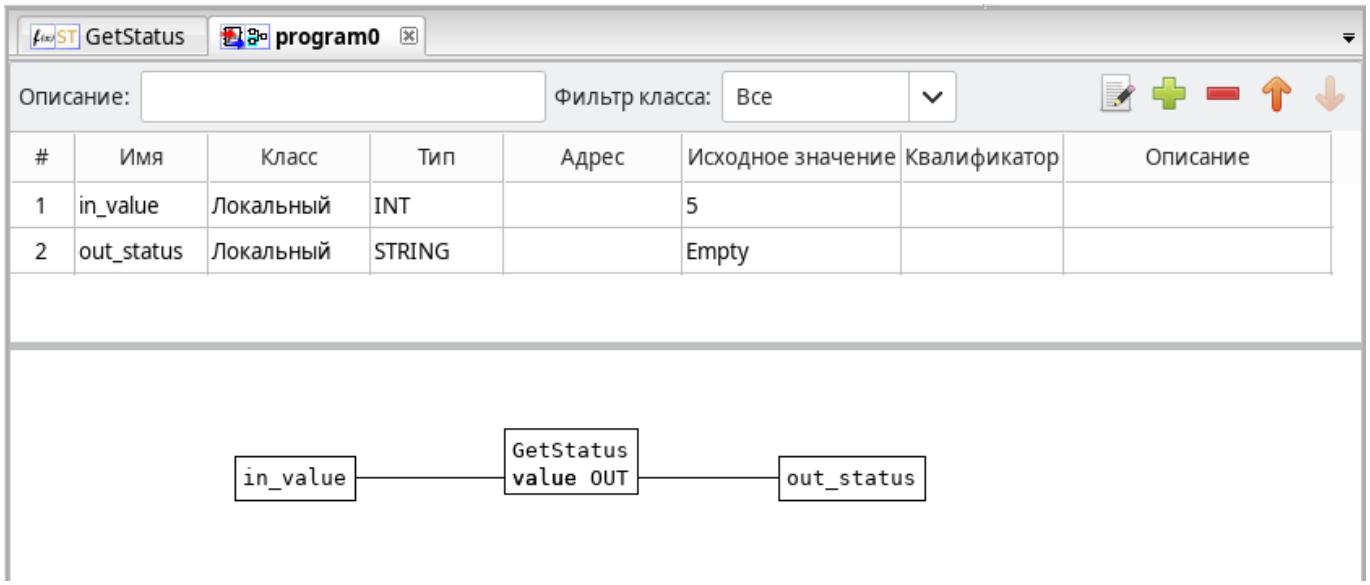


Рисунок 145 – Пример FBD диаграммы с использованием пользовательской функции

В результате созданная функция будет возвращать текстовое значение в переменную «out_status» в зависимости от значения переменной «in_value». Стоит отметить, что в данном примере значение переменной «in_value» постоянно равно 5 (для упрощения примера), но она также может зависеть от других переменных или быть связана, используя поле «Адрес», например, с переменными внешних модулей УСО.

7.5.3. Функциональный блок

Добавление пользовательского функционального блока происходит аналогично добавлению функции. Ниже приведён пример создания функционального блока и его использования. После выбора в дереве проекта добавить «Функциональный блок», создаётся функциональный блок с именем «RectParams», который будет считать площадь и периметр прямоугольника с заданными сторонами, рис. 146.

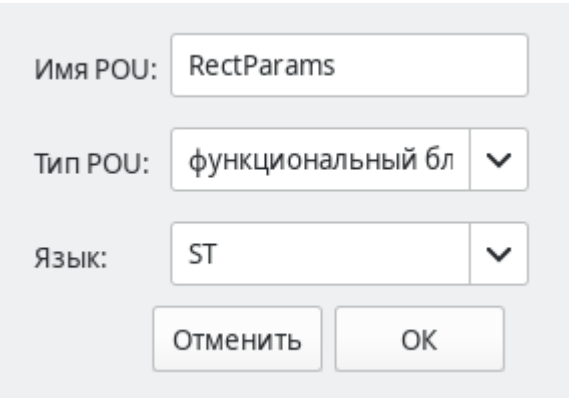


Рисунок 146 – Добавление пользовательского функционального блока

Примечание: при создании нового проекта (см. п. 7.1), автоматически будет вызван диалог "Создать новый POU". Сначала нужно создать программу (для рассматриваемого примера, имя: «program0», тип «программа», язык: FBD). И только после этого добавлять функциональный блок, в соответствии (см. рис. 146).

В отличие от функции, функциональный блок может быть описан на любом языке стандарта IEC 61131-3, включая язык SFC. На рис. 147 показана реализация данного функционального блока на языке ST.

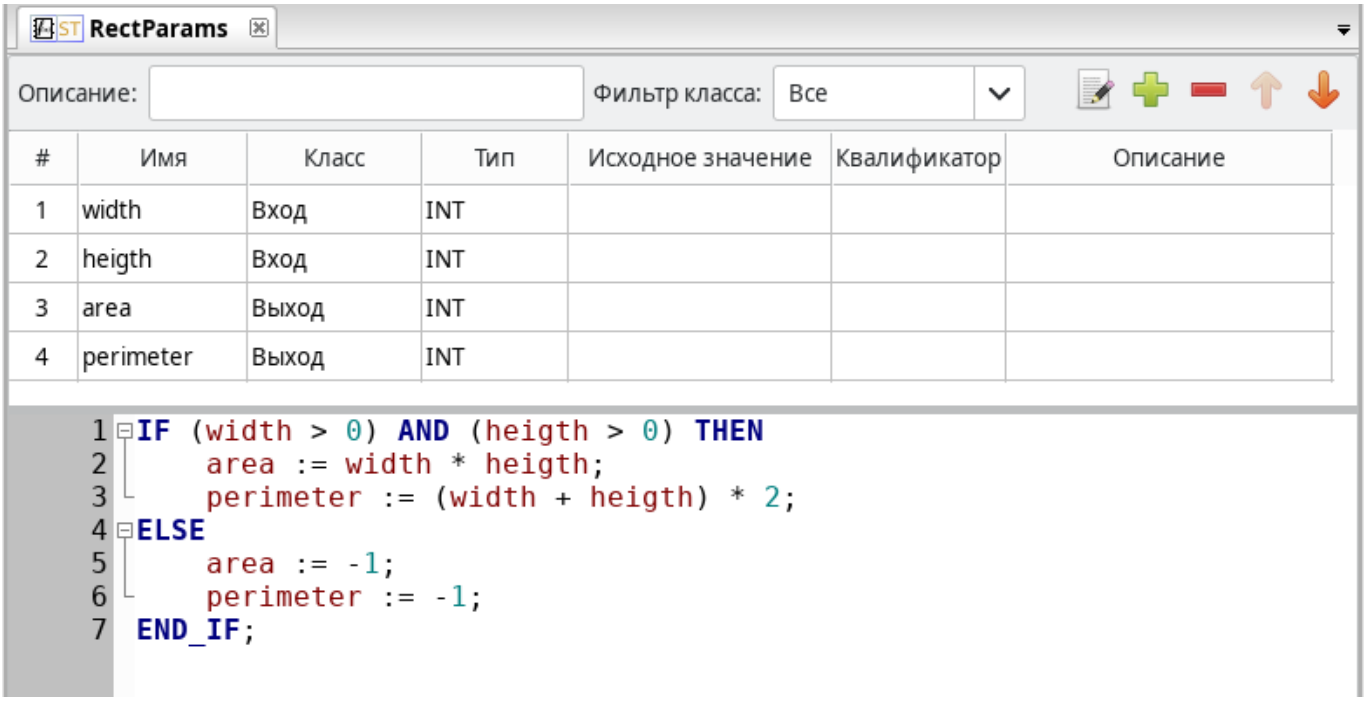
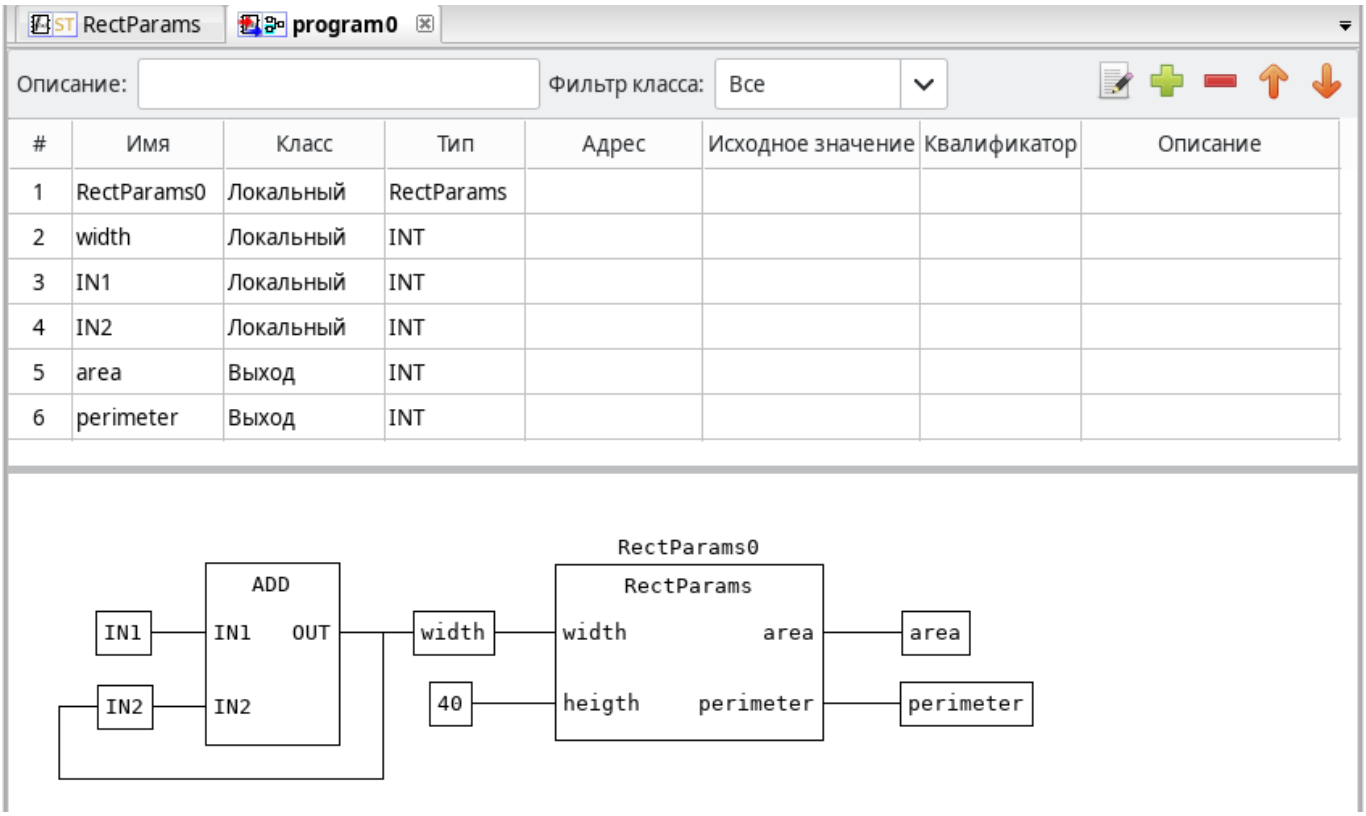


Рисунок 147 – Описание пользовательского функционального блока на языке ST

Возвращаемого значения у функционального блока нет. Добавленные переменные, обозначающие ширину – «width» и высоту – «height» имеют класс «Вход» и тип INT. Выходные значения: площадь – «area» и периметр – «perimeter» определены соответственно классом «Выход» и так же типа INT. Ниже находится текст алгоритма расчёта площади и периметра на языке ST. Реализованный функциональный блок становится доступным в панели библиотеки функций и функциональных блоков (см. п. 6.13) и может использоваться в программных модулях типа «Программа» и «Функциональный блок». На рис. 148 показано использование созданного функционального блока «RectParams» в FBD диаграмме.



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Описание
1	RectParams0	Локальный	RectParams				
2	width	Локальный	INT				
3	IN1	Локальный	INT				
4	IN2	Локальный	INT				
5	area	Выход	INT				
6	perimeter	Выход	INT				

Рисунок 148 – Использование созданного функционального блока в FBD диаграмме

С входом «width» соединена переменная «width», а с входом «height» литерал «40». Результат выполнения данного функционального блока также помещается, соответственно, в «area» и «perimeter».

Следует отметить, что при попытке удаления функции или функционального блока из проекта, где эти добавленные программные модули уже используются, будет выдана ошибка, рис. 149.

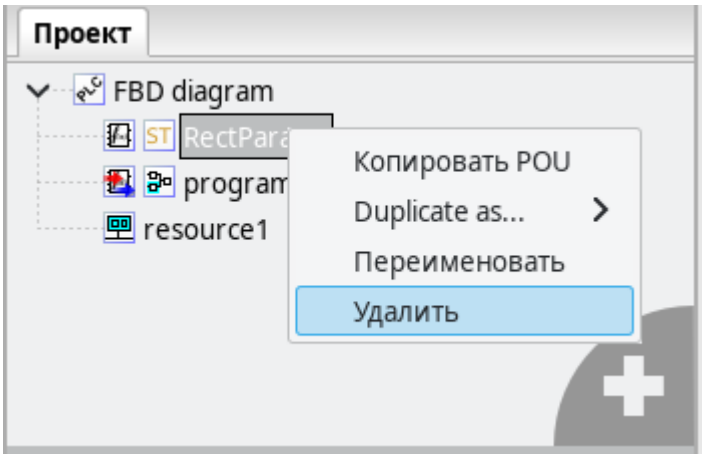


Рисунок 149 – Удаление функционального блока

Пример сообщения об ошибке удаления программного модуля приведён на рис. 150.

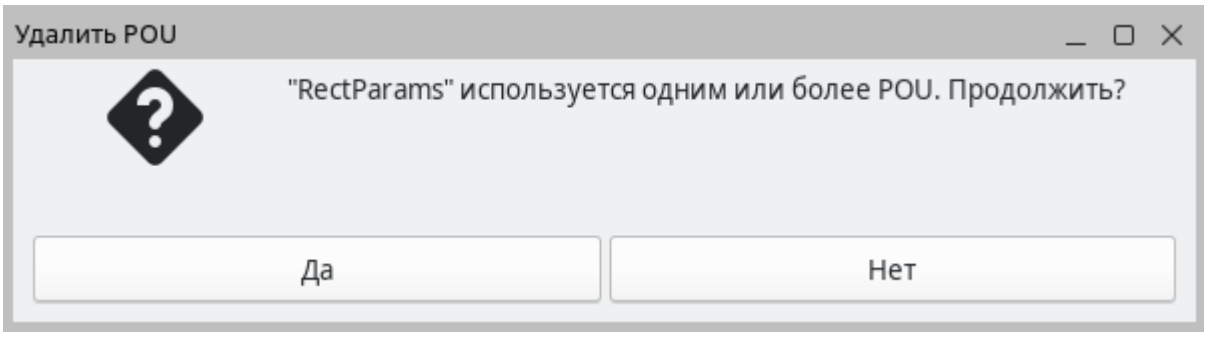


Рисунок 150 – Сообщение об ошибке при удалении функционального блока

7.6. Плагины модулей УСО, протоколов коммуникации, функций, типов данных

Панель редактирования плагинов для модулей УСО содержит общие настройки для всех плагинов и специфичные в зависимости от назначения плагина.

В проект, содержащий программный модуль типа «Программа», написанный на любом из поддерживаемых языков МЭК, добавляются плагины модулей УСО или плагины протоколов коммуникации. Данная операция осуществляется с помощью соответствующих пунктов меню структура проекта, рис. 151.

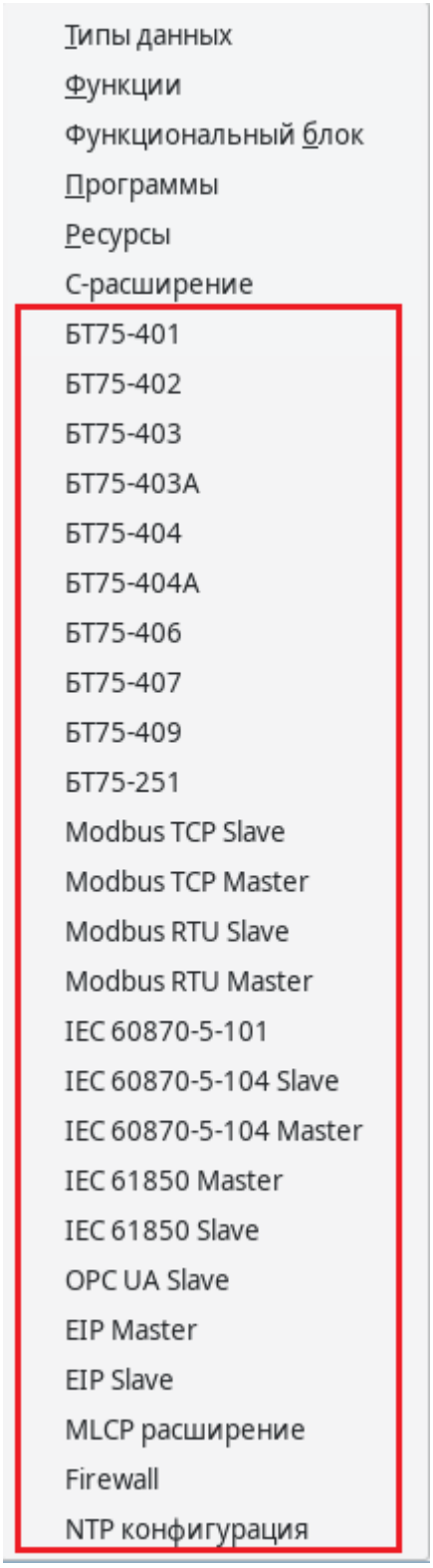


Рисунок 151 – Выбор плагинов для добавления в проект

После добавления плагинов в структуру проекта добавляется соответствующий узел. Образец того, как это может выглядеть приведён на рис. 152.

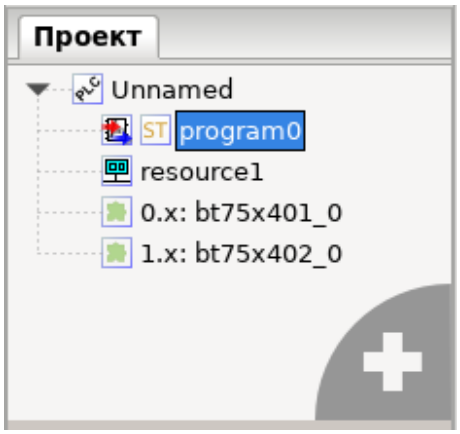


Рисунок 152 – Структура проекта после добавления двух плагинов модулей УСО

После добавления плагинов в программном модуле появляется возможность создать переменные, связанные с переменными модулей, которые были добавлены выше. Данную операцию следует осуществить с помощью кнопки «Добавить переменную» (см. таблицу 3) на панели переменных и констант.

Для связи добавленной переменной с переменной модуля, необходимо нажать на поле «Адрес» и в появившемся диалоговом окне выбрать модуль и необходимую переменную, как показано на рис. 153. Тип добавленной переменной будет автоматически заменён на тип выбранной переменной в плагине.

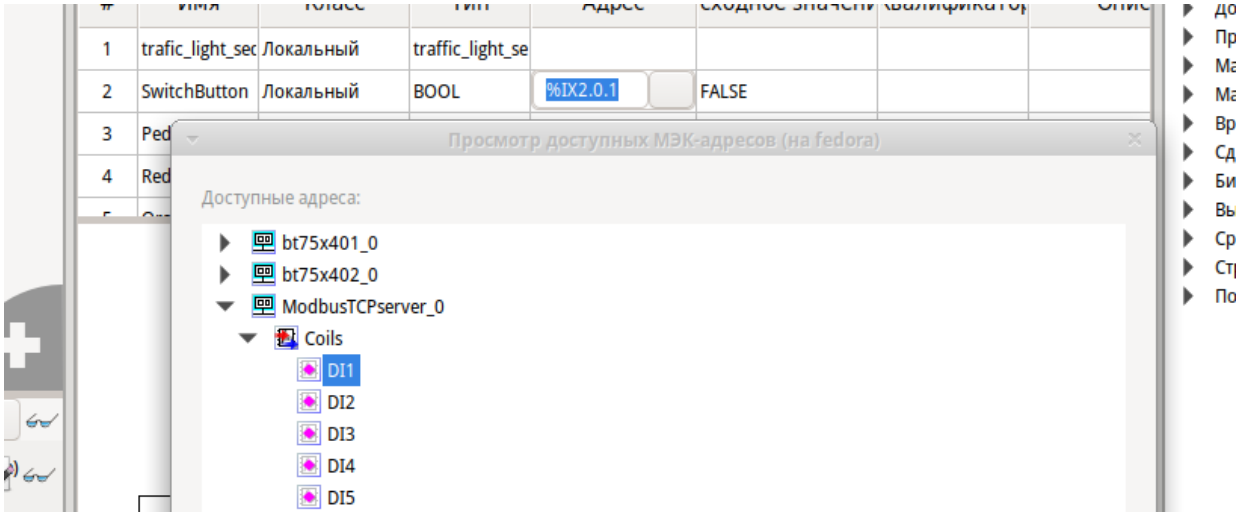
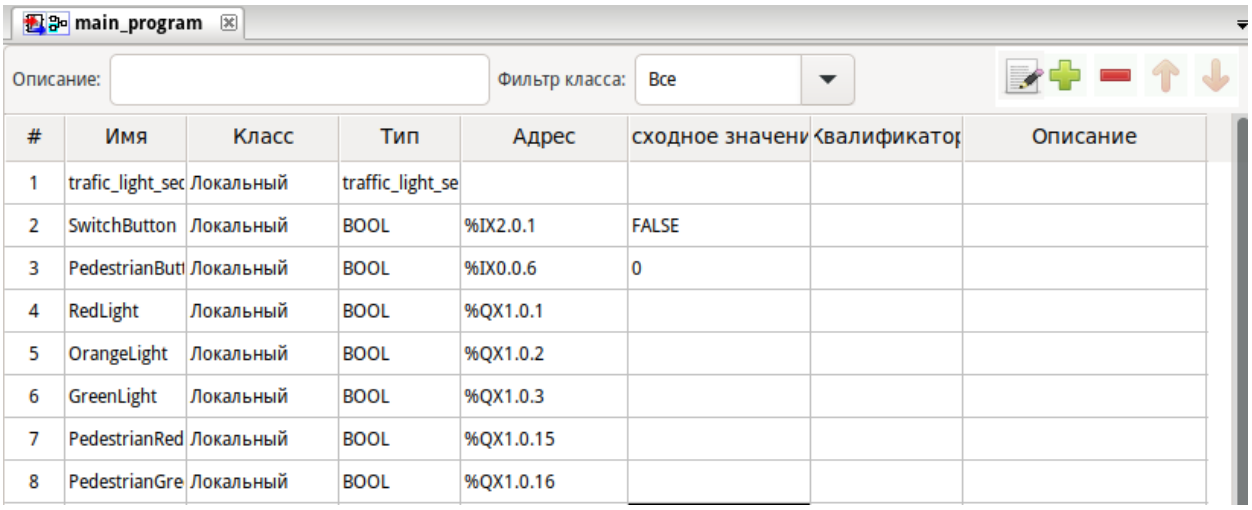


Рисунок 153 – Добавление переменных модуля «ModbusTCPserver_0»

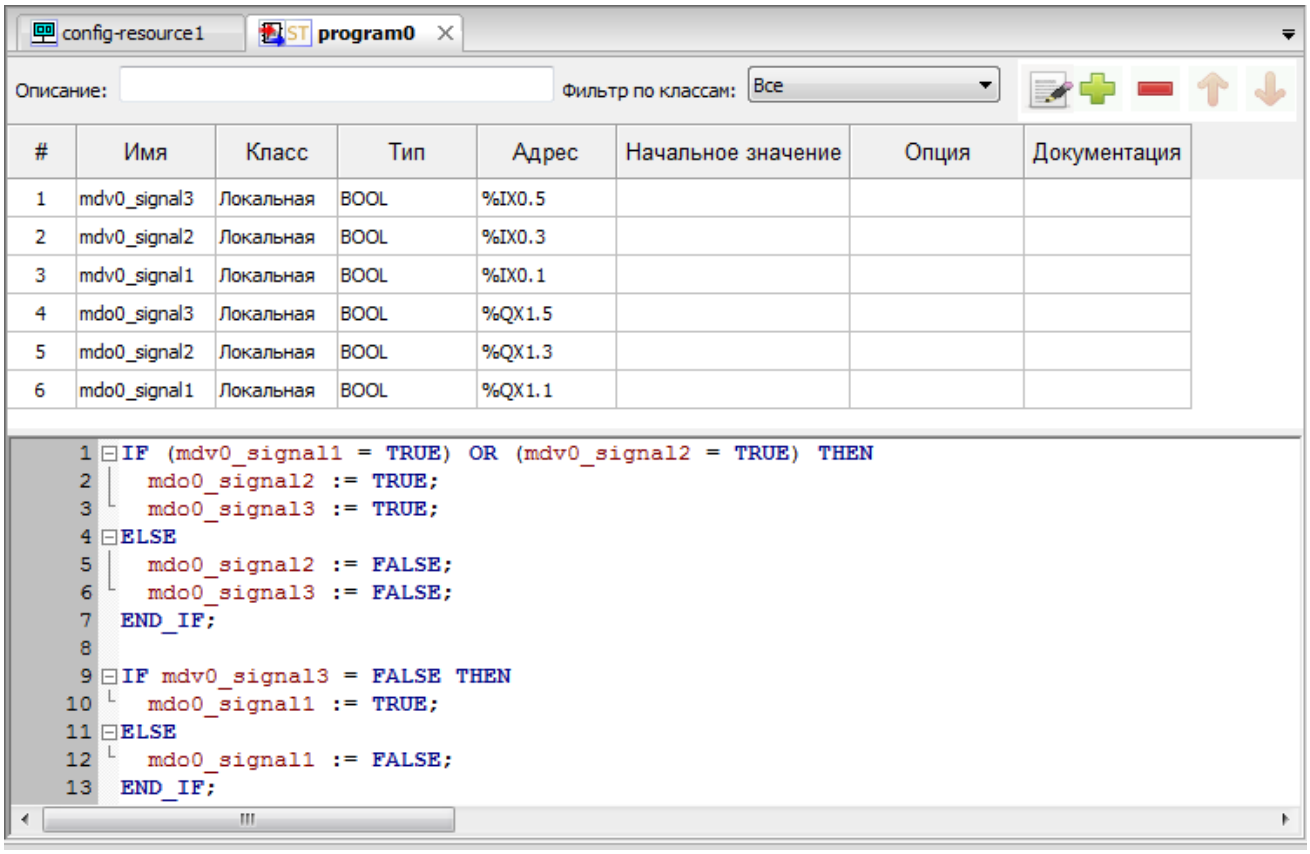
После того, как все необходимые переменные добавлены, в панели переменных и констант будут отображены все переменные соответствующего типа и класса с установленным адресом (см. рис. 154).



#	Имя	Класс	Тип	Адрес	сходное значени	Квалификатор	Описание
1	trafic_light_sec	Локальный	traffic_light_se				
2	SwitchButton	Локальный	BOOL	%IX2.0.1	FALSE		
3	PedestrianButt	Локальный	BOOL	%IX0.0.6	0		
4	RedLight	Локальный	BOOL	%QX1.0.1			
5	OrangeLight	Локальный	BOOL	%QX1.0.2			
6	GreenLight	Локальный	BOOL	%QX1.0.3			
7	PedestrianRed	Локальный	BOOL	%QX1.0.15			
8	PedestrianGre	Локальный	BOOL	%QX1.0.16			

Рисунок 154 – Панель переменных и констант после добавления переменных модулей УСО и модулей протоколов коммуникации

Следует отметить, что имена добавленных переменных можно изменить. Далее переменные, связанные с модулями, могут быть использованы в коде программы на языке МЭК как обычная переменная (см. пример на рис. 155):



#	Имя	Класс	Тип	Адрес	Начальное значение	Опция	Документация
1	mdv0_signal3	Локальная	BOOL	%IX0.5			
2	mdv0_signal2	Локальная	BOOL	%IX0.3			
3	mdv0_signal1	Локальная	BOOL	%IX0.1			
4	mdo0_signal3	Локальная	BOOL	%QX1.5			
5	mdo0_signal2	Локальная	BOOL	%QX1.3			
6	mdo0_signal1	Локальная	BOOL	%QX1.1			

```
1  IF (mdv0_signal1 = TRUE) OR (mdv0_signal2 = TRUE) THEN
2      mdo0_signal2 := TRUE;
3      mdo0_signal3 := TRUE;
4  ELSE
5      mdo0_signal2 := FALSE;
6      mdo0_signal3 := FALSE;
7  END_IF;
8
9  IF mdv0_signal3 = FALSE THEN
10     mdo0_signal1 := TRUE;
11 ELSE
12     mdo0_signal1 := FALSE;
13 END_IF;
```

Рисунок 155 – Панель редактирования программного модуля

7.6.1. Плагины дискретных модулей ввода/вывода серии БТ75-4XX

Плагины дискретных модулей ввода/вывода серии БТ75-4XX поддерживают работу с модулями и работы цифровыми сигналами:

– БТ75-401: модуль ввода цифровых сигналов (16 каналов);
– БТ75-402: модуль вывода цифровых сигналов (16 каналов);
– БТ75-406: модуль ввода цифровых сигналов 220 В (6 каналов);
– БТ75-407: модуль вывода цифровых сигналов 220 В (6 каналов);
– БТ75-409: модуль ввода быстрых цифровых сигналов (6 каналов).

7.6.1.1. Модуль ввода цифровых сигналов БТ75-401

Для вызова меню параметров модуля необходимо дважды кликнуть по нему в дерева проекта.

Редактор модуля дискретного ввода БТ75-401 с 16 каналами (рис. 156), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;
– «Адрес модуля на шине».

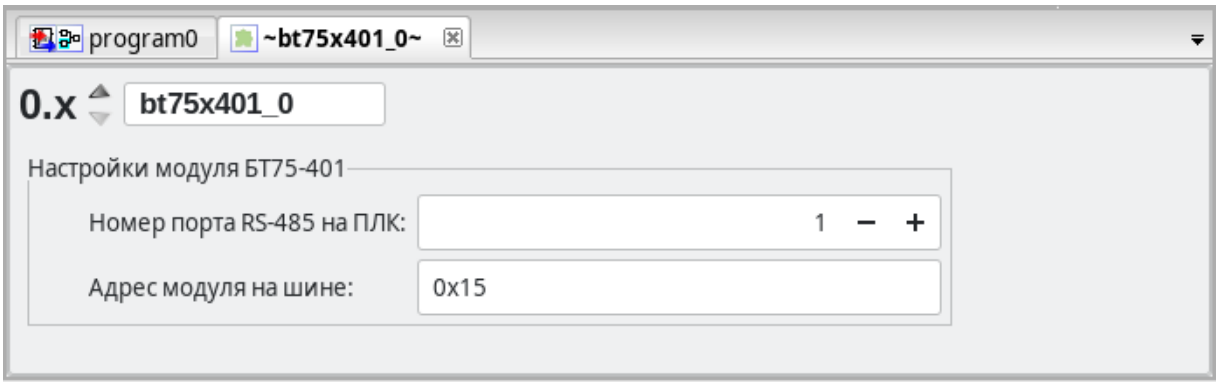


Рисунок 156 – Редактор модуля ввода цифровых сигналов БТ75-401

Указанные параметры соответствуют описанию в ЮКСУ.91167-01 33 01 «ПО ПЛК. Руководство программиста». Значение «Номер порта RS-485 на ПЛК» задаётся в десятичном виде, а значение «Адрес модуля на шине» - в шестнадцатеричном или десятичном виде. Изменение значения «Номер порта RS-485 на ПЛК» доступно либо кнопками «-», «+», либо выделением и вводом нового значения с клавиатуры, изменение значения значение «Адрес модуля на шине» осуществляется вводом нового значения с клавиатуры.

Как уже было отмечено в п. 7.6, с помощью программы в проект можно добавить переменную для связи с переменной модуля. Тип добавленной переменной будет автоматически заменён на тип выбранной переменной в плагине. Описание переменных модуля представлено в табл. 13.

Таблица 13 – Переменные модуля БТ75-401

Переменная модуля	Описание
BOOL IN[1-16]	Состояние входов, формат поля. Запись/чтение
BOOL ENA_SET[1-16]	Включение/выключение каналов DI. Если бит установлен, то соответствующий канал используется. Запись/чтение.
BOOL WB[1-16]	Признак обрыва каналов. Если установлен, то соответствующий канал DI зафиксировал обрыв подключения датчика. Устанавливается если соответствующий бит WB_ENA установлен. Чтение.
BOOL CNT_ENA[1-16]	Включение/выключение счета каналов DI. При установленном бите: счётный вход канала включён. Запись/чтение
BOOL POS_EDGE[1-16]	Тип изменения входного состояния счётного канала: при установленном бите счётчик увеличивается при переходе канала из состояния «ВЫКЛЮЧЕНО» в состояние «ВКЛЮЧЁНО». Запись/чтение.
DWORD CNT[1-16]	Счётчики сигналов для каналов.
UINT TIME[1-16]	Временной отсчёт для каналов DI. Чтение.
BOOL STATUS_OK	Статус работоспособности модуля. True — работает, False — нет.

Для работы с сигналами, период изменений которых меньше периода выполнения задачи, рекомендуется использовать счётчики сигналов (переменная CNT), чтобы избежать потери сигналов. При использовании счётчика на каждой итерации задачи необходимо проверять, изменилось ли значение переменной счётчика. Изменение значения свидетельствует о прохождении сигнала.

Регистры устройства соответствуют ЮКСУ.91167-01 33 01 «ПО ПЛК. Руководство программиста».

Поля переменных модуля БТ75-401 представлены на рис. 157.

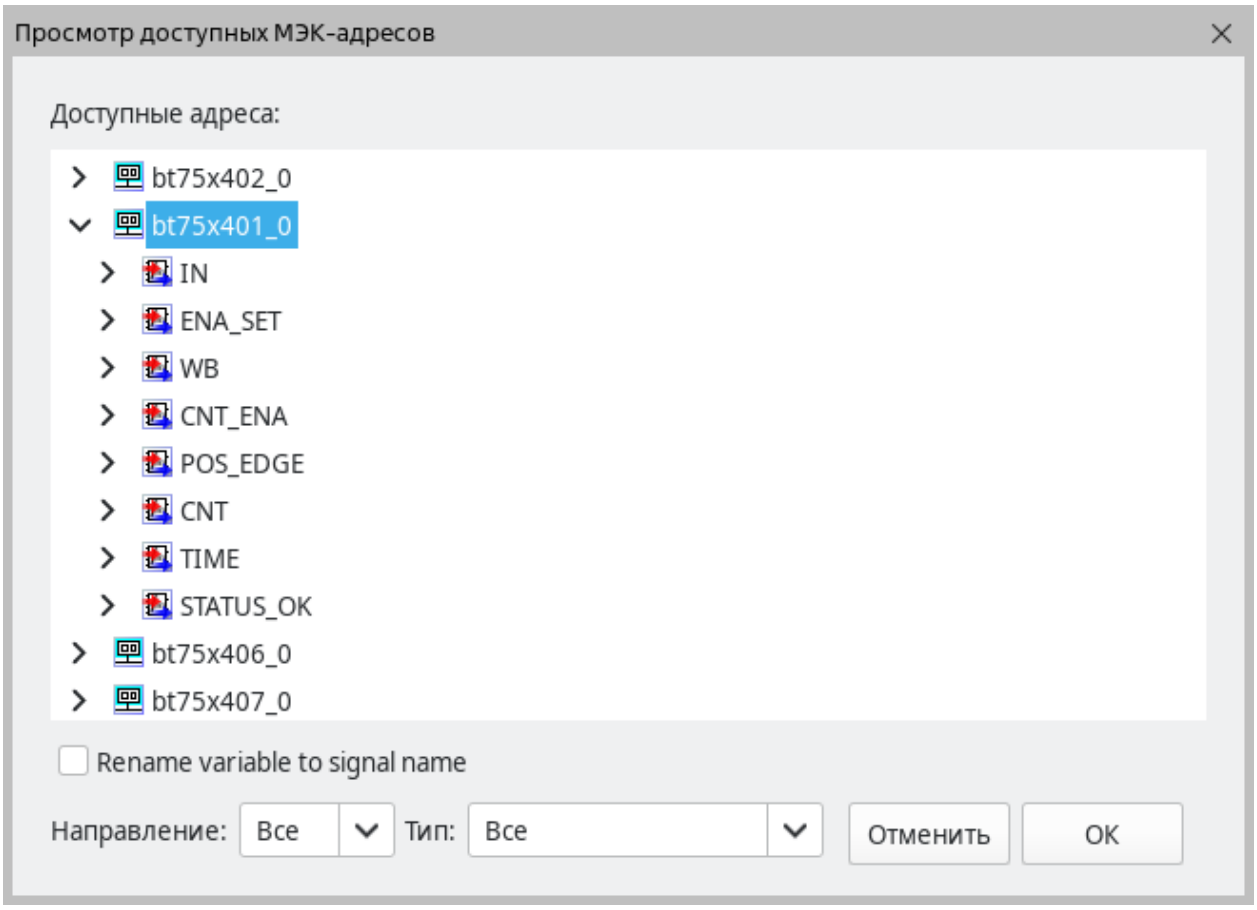


Рисунок 157 – Поля переменных модуля БТ75-401

7.6.1.2. Модуль вывода цифровых сигналов БТ75-402

Для вызова меню параметров модуля необходимо дважды кликнуть по нему в дереве проекта.

Редактор модуля дискретного вывода БТ75-402 с 16 каналами (рис. 158), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

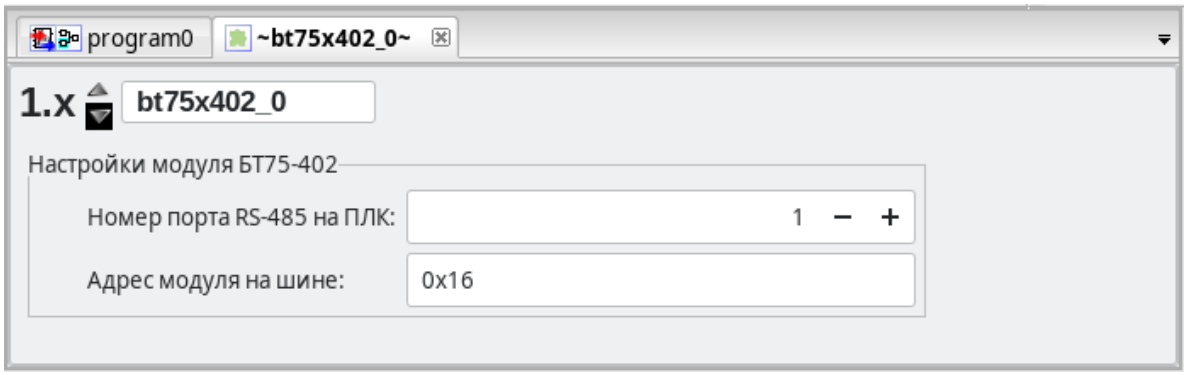


Рисунок 158 – Редактор модуля вывода цифровых сигналов БТ75-402

Описание переменных модуля, доступных для программы, представлено в табл. 141414.

Таблица 14 – Переменные модуля БТ75-402

Переменная модуля	Описание
`BOOL OUT[1-16]`	Установка выходного сигнала.
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-402 представлены на рис. 159.

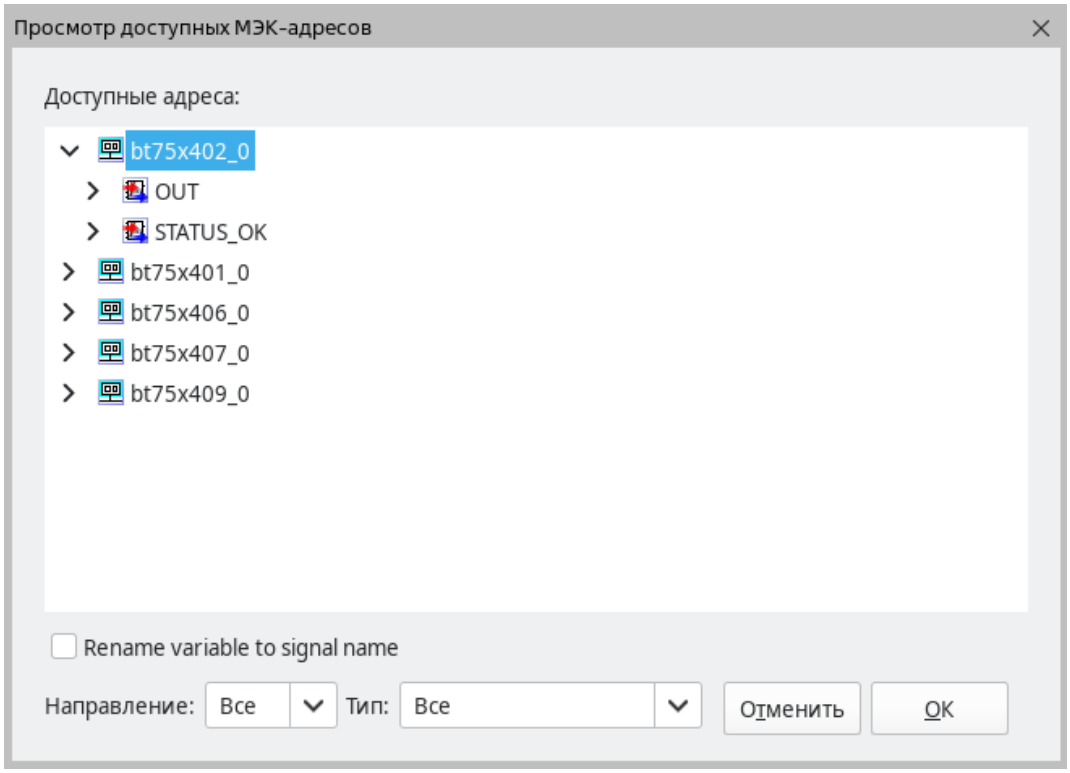


Рисунок 159 – Поля переменных модуля БТ75-402

7.6.1.3. Модуль ввода цифровых сигналов 220 В БТ75-406

Для вызова меню параметров модуля необходимо дважды кликнуть по нему в дереве проекта.

Редактор модуля ввода цифровых сигналов 220 В БТ75-406 с шестью каналами (рис. 160), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

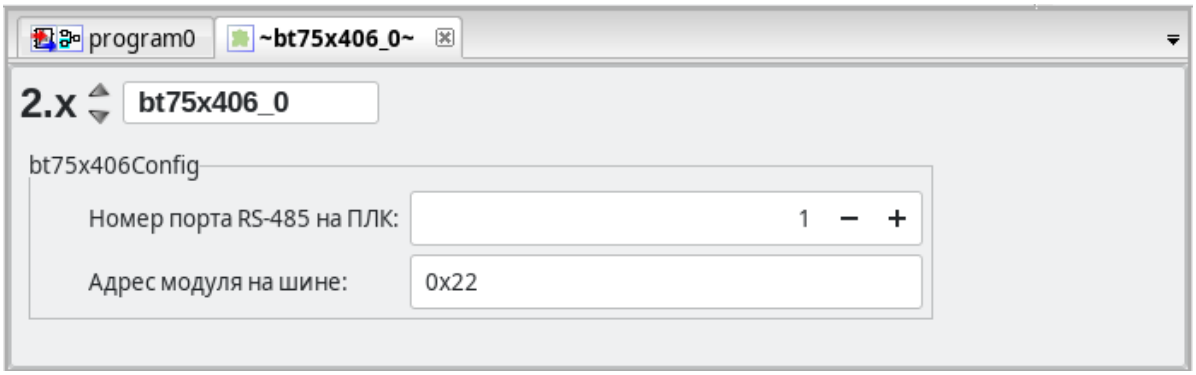


Рисунок 160 – Редактор модуля ввода цифровых сигналов 220 В БТ75-406

Описание переменных модуля, доступных для программы, представлено в табл. 151515.

Таблица 15 – Переменные модуля БТ75-406

Переменная модуля	Описание
`BOOL IN[1-6]`	Состояние входов, формат поля: Запись/чтение
`BOOL ENA[1-6]`	Включение/выключение каналов DI. Если бит установлен, то соответствующий канал используется. Запись/чтение.
`DWORD CNT[1-6]`	Счётчики каналов.
`BOOL CNT_ENA[1-6]`	Включение/выключение счета каналов DI. При установленном бите: счётный вход канала включён. Запись/чтение
`BOOL POS_EDGE[1-6]`	Тип изменения входного состояния счётного канала: при установленном бите счётчик увеличивается при переходе канала из состояния «ВЫКЛЮЧЕНО» в состояние «ВКЛЮЧЁНО». Запись/чтение.
`UINT TIME[1-6]`	Временной отсчёт для каналов DI. Чтение.
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-406 представлены на рис. 161.

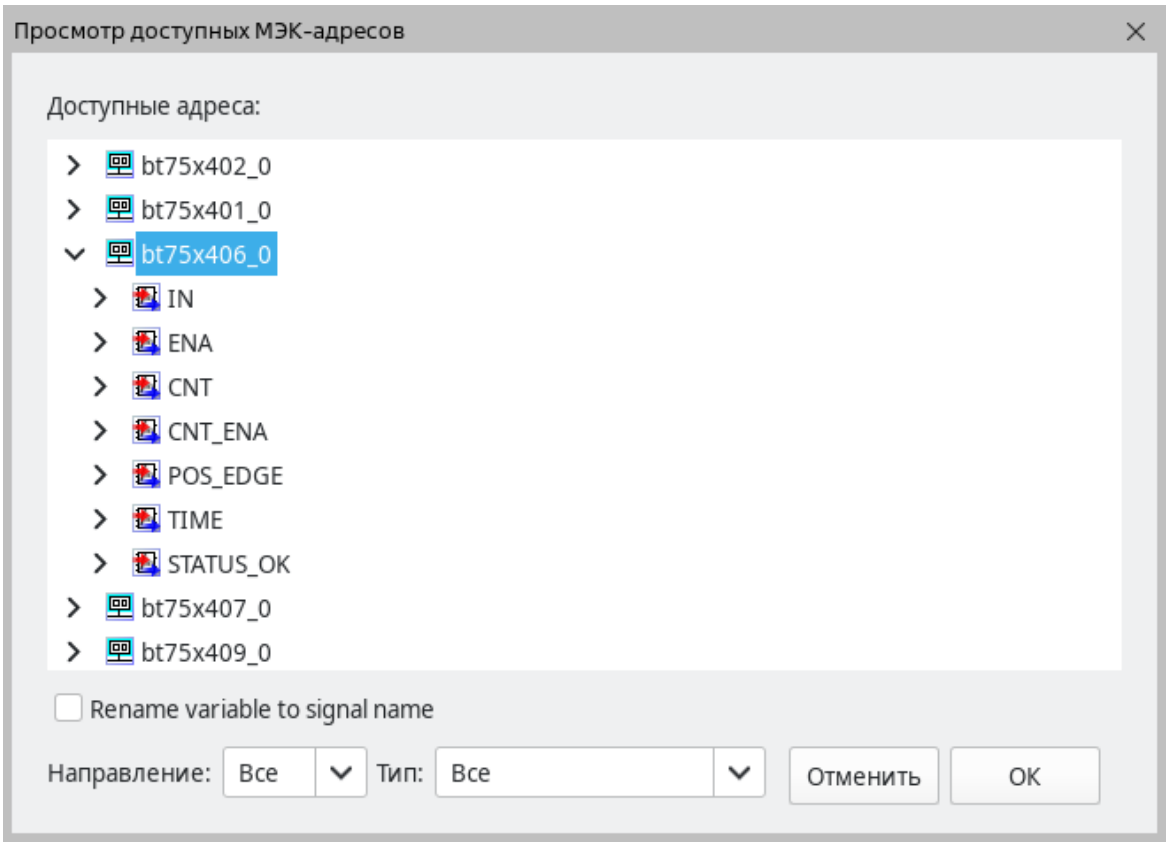


Рисунок 161 – Поля переменных модуля БТ75-406

Для работы с сигналами, период изменений которых меньше периода выполнения задачи, рекомендуется использовать счётчики сигналов (переменная CNT), чтобы избежать потери сигналов. При использовании счётчика на каждой итерации задачи необходимо проверять, изменилось ли значение переменной счётчика. Изменение значения свидетельствует о прохождении сигнала.

7.6.1.4. Модуль вывода цифровых сигналов 220 В БТ75-407

Для вызова меню параметров модуля необходимо дважды кликнуть по нему в дереве проекта.

Редактор модуля вывода цифровых сигналов 220 В БТ75-407 с шестью каналами (рис. 162), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

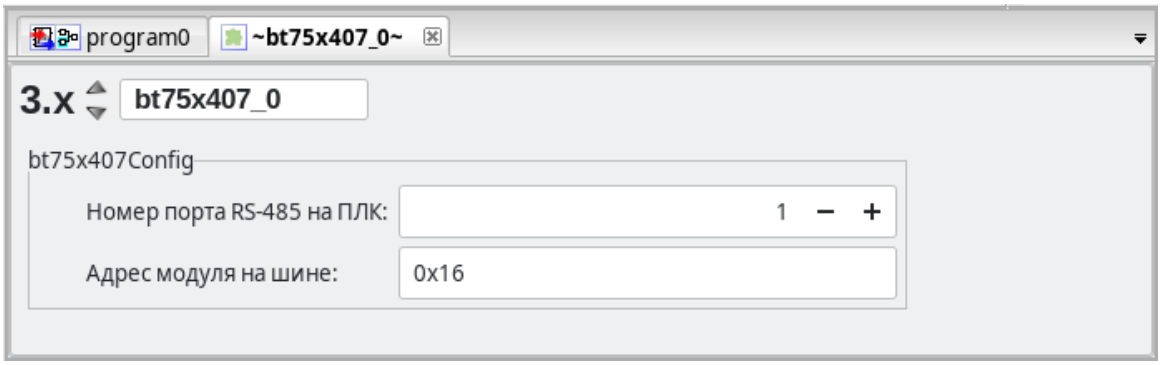


Рисунок 162 – Редактор модуля вывода цифровых сигналов 220 В БТ75-407

Описание переменных модуля, доступных для программы, представлено в табл. 16.

Таблица 16 – Переменные модуля БТ75-407

Переменная модуля	Описание
`BOOL OUT[1-6]`	Установка выходного сигнала.
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-407 представлены на рис. 163.

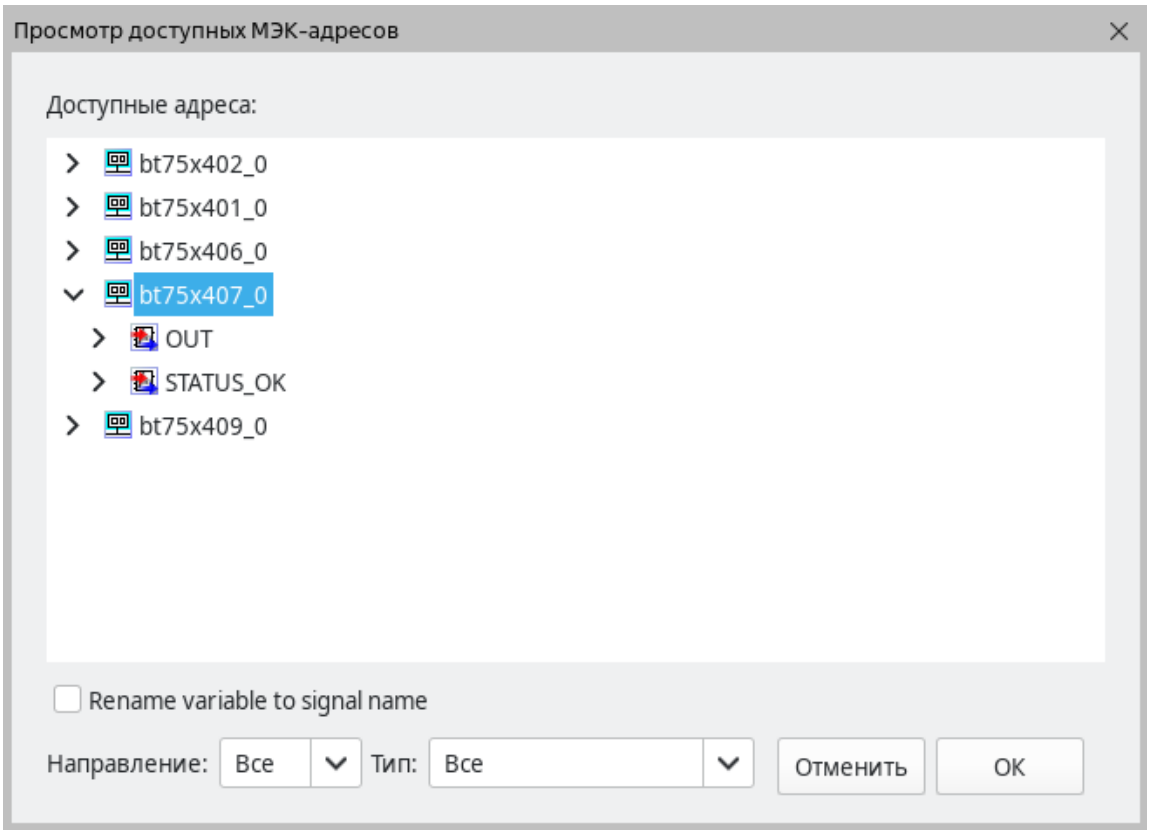


Рисунок 163 – Поля переменных модуля БТ75-407

7.6.1.5. Модуль ввода быстрых цифровых сигналов БТ75-409

Для вызова меню параметров модуля необходимо дважды кликнуть по нему в дереве проекта.

Редактор модуля ввода быстрых цифровых сигналов БТ75-409 с шестью каналами (рис. 164), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

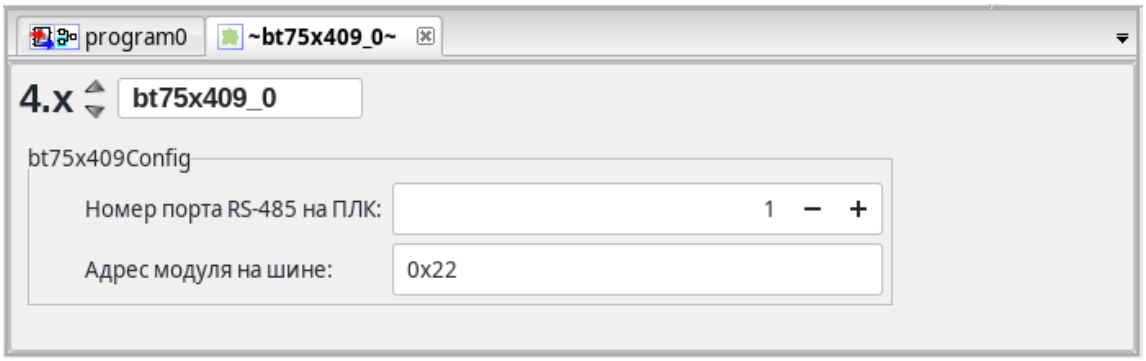


Рисунок 164 – Редактор модуля ввода быстрых цифровых сигналов БТ75-409

Описание переменных модуля, доступных для программы, представлено в табл. 17.

Таблица 17 – Переменные модуля БТ75-409

Переменная модуля	Описание
`BOOL IN[1-6]`	Состояние входов, формат поля: Запись/чтение
`BOOL ENA[1-6]`	Включение/выключение каналов DI. Если бит установлен, то соответствующий канал используется. Запись/чтение.
`DWORD CNT[1-16]`	Счётчики каналов.
`BOOL CNT_ENA[1-6]`	Включение/выключение счета каналов DI. При установленном бите: счётный вход канала включён. Запись/чтение
`BOOL POS_EDGE[1-6]`	Тип изменения входного состояния счётного канала: при установленном бите счётчик увеличивается при переходе канала из состояния «ВЫКЛЮЧЕНО» в состояние «ВКЛЮЧЁНО». Запись/чтение.
`UINT TIME[1-6]`	Временной отсчёт для каналов DI. Чтение.
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-409 представлены на рис. 165.

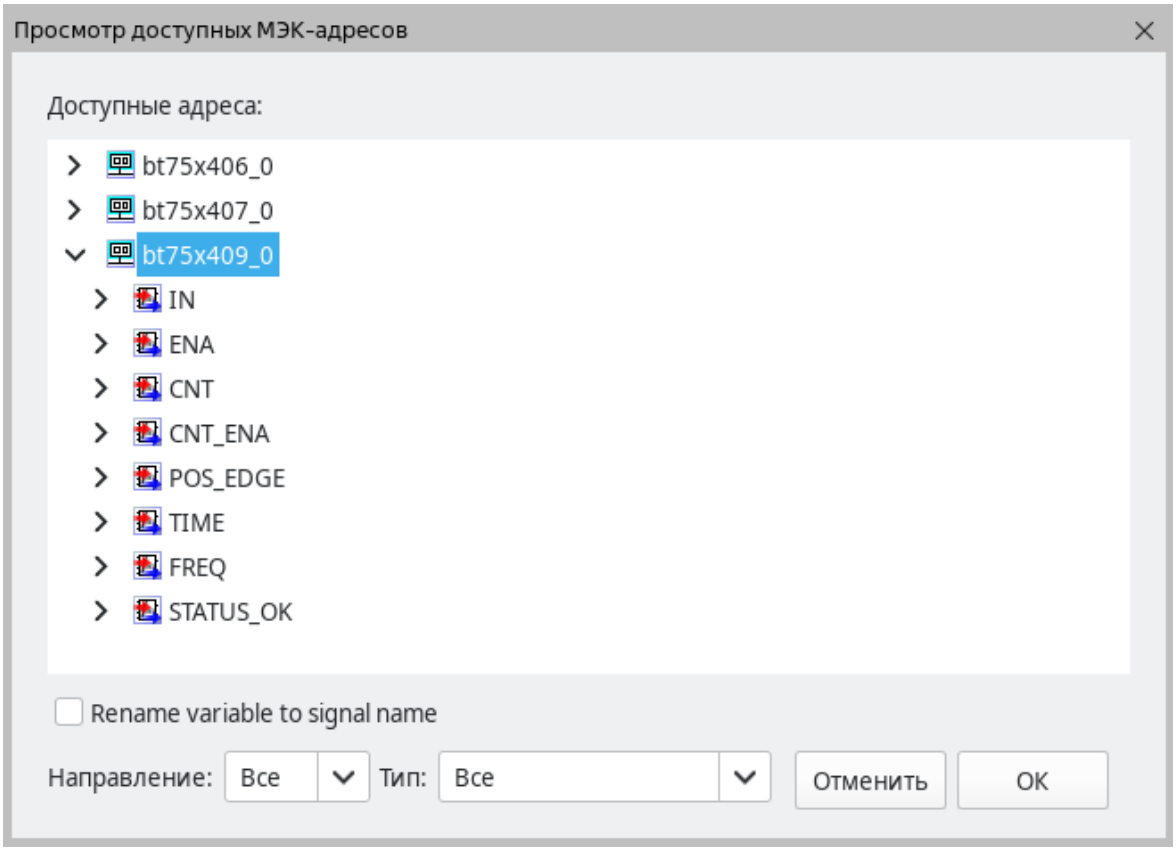


Рисунок 165 – Поля переменных модуля БТ75-409

Для работы с сигналами, период изменений которых меньше периода выполнения задачи, рекомендуется использовать счётчики сигналов (переменная CNT), чтобы избежать потери сигналов. При использовании счётчика на каждой итерации задачи необходимо проверять, изменилось ли значение переменной счётчика. Изменение значения свидетельствует о прохождении сигнала.

7.6.2. Плагины аналоговых модулей ввода/вывода серии БТ75-4XX

Плагины аналоговых модулей ввода/вывода серии БТ75-4XX поддерживают работу с модулями работы с аналоговыми сигналами:

– БТ75-403: модуль ввода аналоговых сигналов тока (8 каналов);
– БТ75-403А: модуль ввода аналоговых сигналов напряжения (8 каналов);
– БТ75-404: модуль вывода аналоговых сигналов тока (4 канала);
– БТ75-404А: модуль вывода аналоговых сигналов напряжения (4 канала).

7.6.2.1. Модуль ввода аналоговых сигналов тока БТ75-403

Редактор модуля дискретного ввода БТ75-403 с восемью каналами, (рис. 166), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;
– «Адрес модуля на шине»;
– «Hi_1» - верхний предел сигнала тока для канала (входа) 1;
– «Lo_1» - нижний предел сигнала тока для канала 1;
– включить/выключить фильтр подавления дребезга для канала 1;
– «Hi_2» - верхний предел сигнала тока для канала 2;
– «Lo_2» - нижний предел сигнала тока для канала 2;
– включить/выключить фильтр подавления дребезга для канала 2;
– …..;
– «Hi_8» - верхний предел сигнала тока для канала 8;
– «Lo_8» - нижний предел сигнала тока для канала 8;
– включить/выключить фильтр подавления дребезга для канала 8.

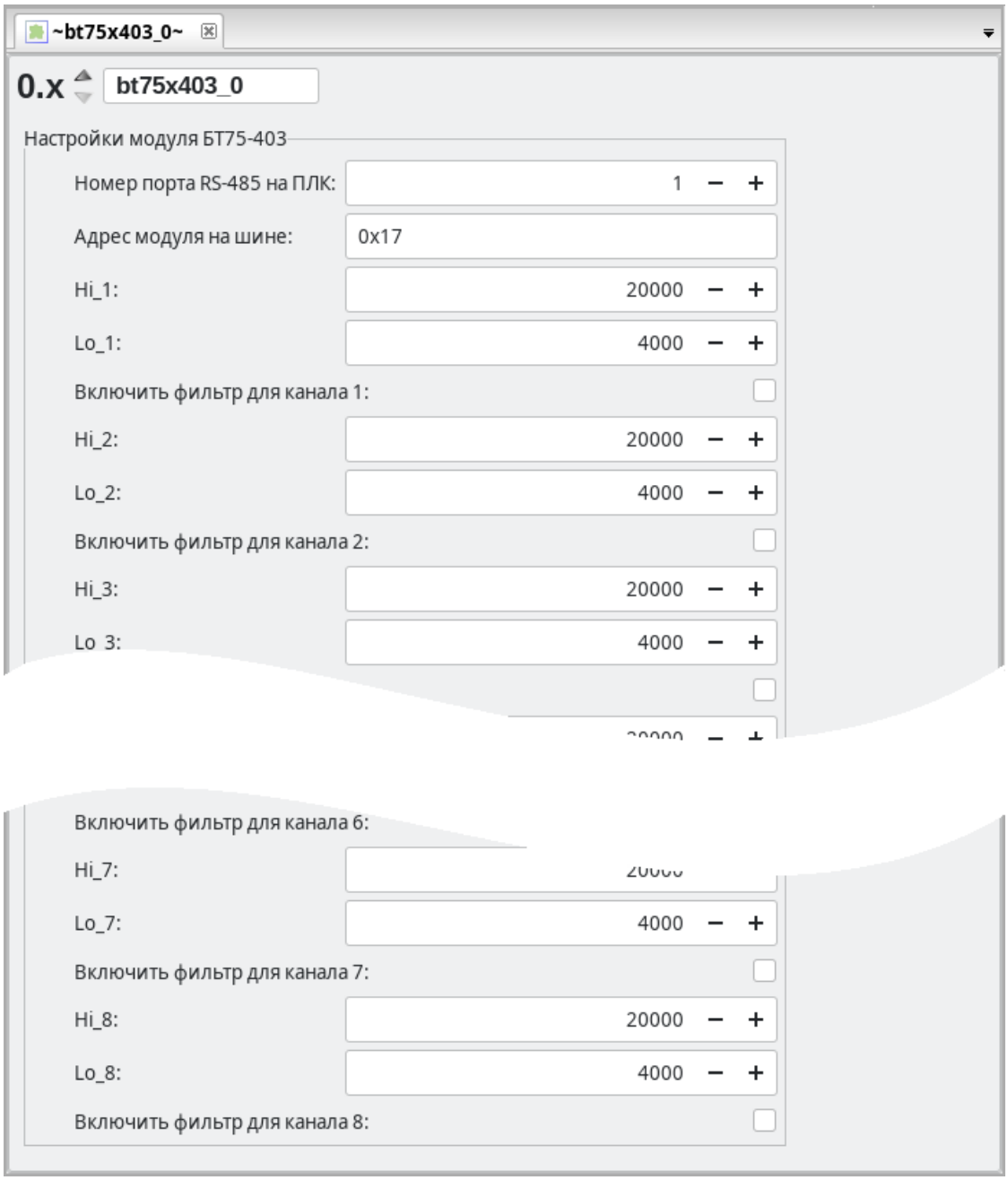


Рисунок 166 – Редактор модуля ввода аналоговых сигналов тока БТ75-403

Указанные параметры соответствуют описанию в ЮКСУ.91167-01 33 01 «ПО ПЛК. Руководство программиста». Значение «Номер порта RS-485 на ПЛК» задаётся в десятичном виде, а значение «Адрес модуля на шине» - в шестнадцатеричном или десятичном виде. Изменение значения «Номер порта RS-485 на ПЛК» доступно либо кнопками «-», «+», либо выделением и вводом нового значения с клавиатуры, изменение значения значение «Адрес модуля на шине» осуществляется вводом нового значения с клавиатуры. В полях «Hi_i», «Lo_i» задаются целочисленные значения порогов в мкА.

Описание переменных модуля, доступных для программы, представлено в табл. 18.

Таблица 18 – Переменные модуля БТ75-403

Переменная модуля	Описание
BOOL ENA[1-8]	Включение/выключение канала. TRUE – включен.
DINT DATA_AI[1-8]	Данные каналов мА или В умноженные на 1000.
BYTE STATUS[1-8]	Статус канала: 00 – нет ошибок; 01 – превышен максимальный уровень инженерной величины; 10 – данные меньше минимального уровня инженерной величины; 11 – ошибка измерения.
BOOL POS_EDGE[1-16]	Тип изменения входного состояния счётного канала: при установленном бите счётчик увеличивается при переходе канала из состояния «ВЫКЛЮЧЕНО» в состояние «ВКЛЮЧЁНО». Запись/чтение.
UINT TIME[1-16]	Временной отсчёт для канала.
BOOL STATUS_OK	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-403 представлены на рис. 167.

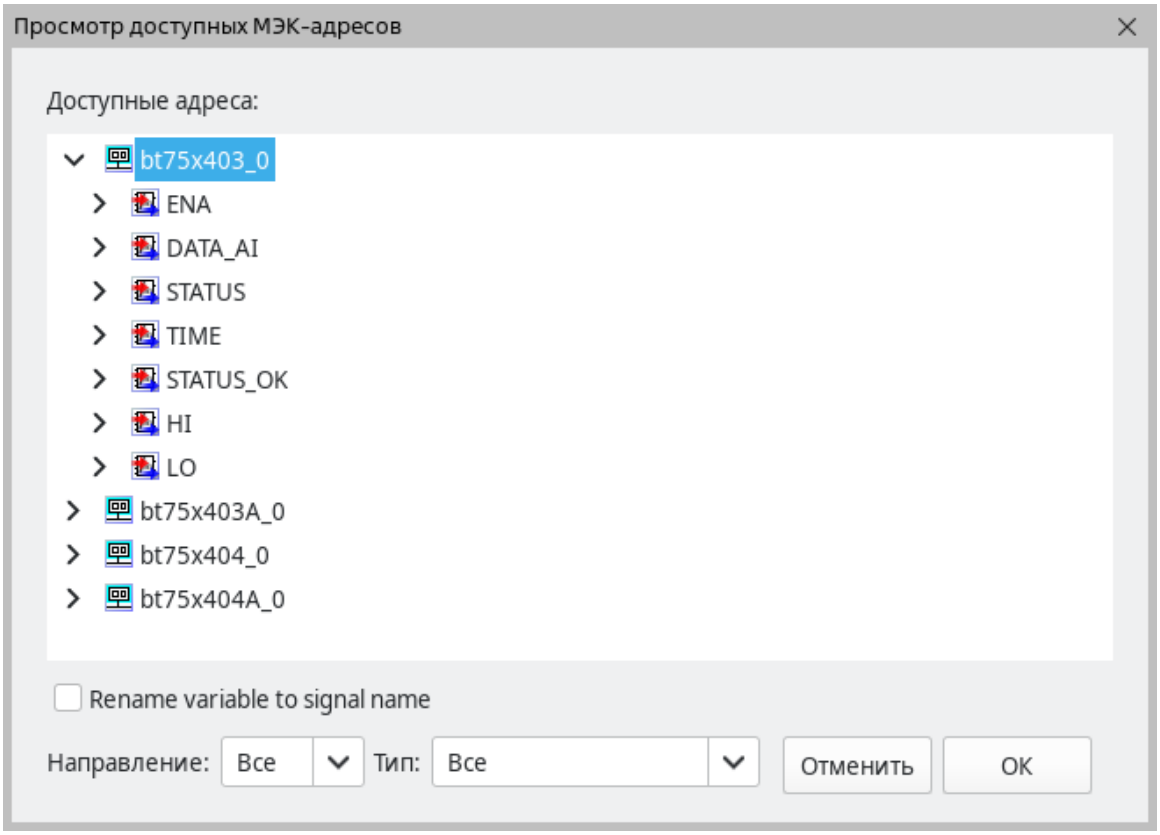


Рисунок 167 – Поля переменных модуля БТ75-403

7.6.2.2. Модуль вывода аналоговых сигналов напряжения БТ75-403А

Редактор модуля вывода аналоговых сигналов напряжения БТ75-403А с восемью каналами (рис. 168), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;
– «Адрес модуля на шине»;
– «Hi_1» - верхний предел сигнала напряжения для выхода 1;
– «Lo_1» - нижний предел сигнала напряжения для выхода 1;
– включить/выключить фильтр подавления дребезга для выхода 1;
– …..;
– «Hi_8» - верхний предел сигнала напряжения для выхода 8;
– «Lo_8» - нижний предел сигнала напряжения для выхода 8;
– включить/выключить фильтр подавления дребезга для выхода 8.

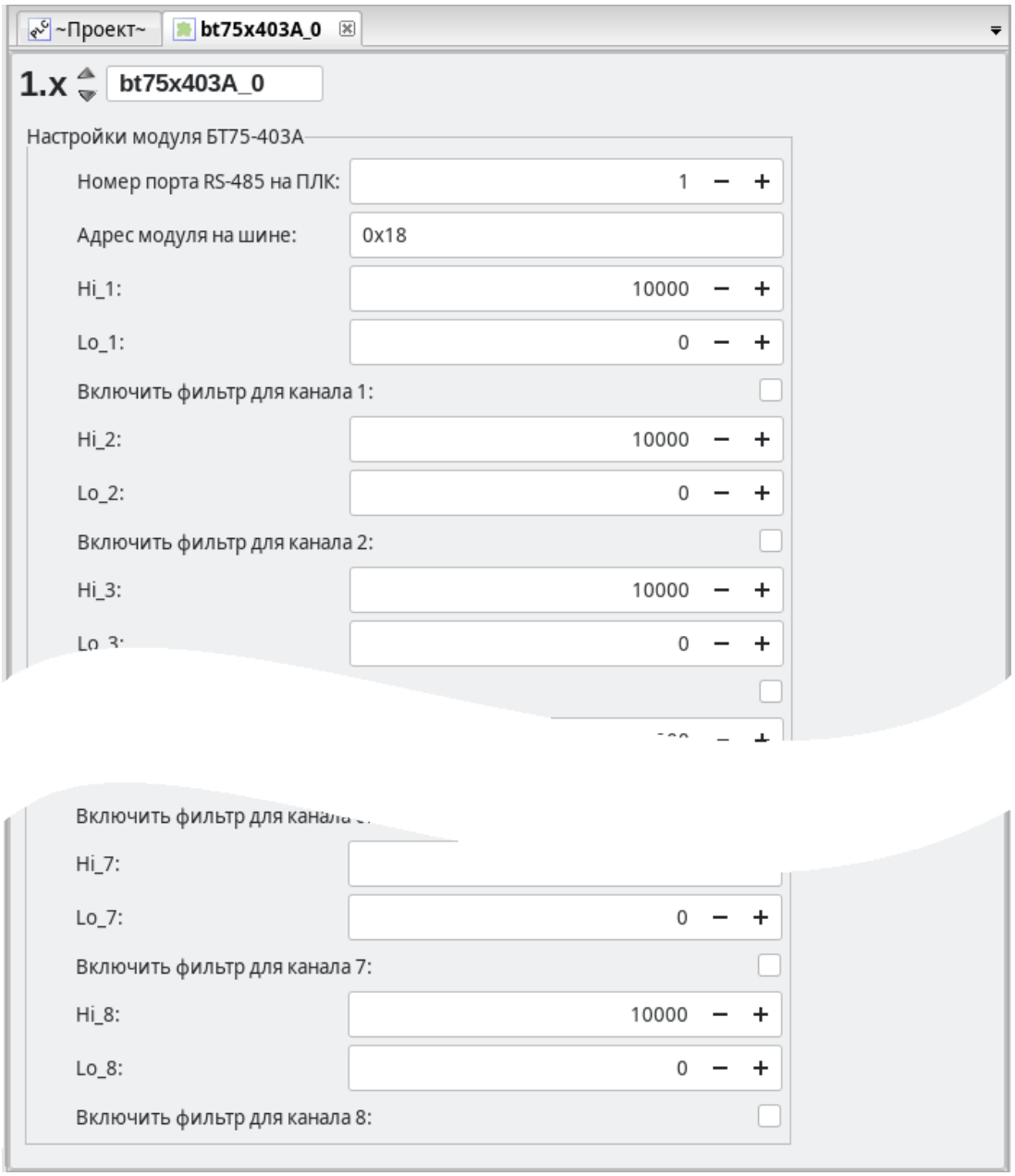


Рисунок 168 – Редактор модуля вывода аналоговых сигналов напряжения БТ75-403А

Описание переменных модуля, доступных для программы, представлено в табл. 19.

Таблица 19 – Переменные модуля БТ75-403А

Переменная модуля	Описание
BOOL ENA[1-8]	Включение/выключение канала. TRUE – включен.
DINT DATA_AI[1-8]	Данные каналов мА или В умноженные на 1000.
BYTE STATUS[1-8]	Статус канала: 00 – нет ошибок; 01 – превышен максимальный уровень инженерной величины; 10 – данные меньше минимального уровня инженерной величины; 11 – ошибка измерения.
BOOL POS_EDGE[1-16]	Тип изменения входного состояния счётного канала: при установленном бите счётчик увеличивается при переходе канала из состояния «ВЫКЛЮЧЕНО» в состояние «ВКЛЮЧЁНО». Запись/чтение.
UINT TIME[1-16]	Временной отсчёт для канала.
BOOL STATUS_OK	Статус работоспособности модуля. True — работает, False — нет.

7.6.2.3. Модуль вывода аналоговых сигналов тока БТ75-404

Редактор модуля вывода аналоговых сигналов тока БТ75-404 с четыремя каналами (рис. 169), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

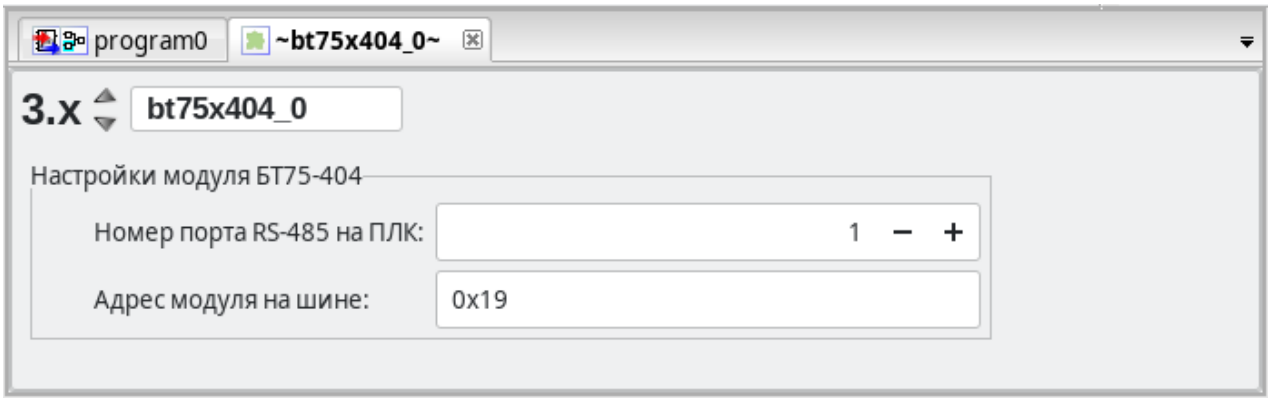


Рисунок 169 – Редактор модуля вывода аналоговых сигналов тока БТ75-404

Описание переменных модуля, доступных для программы, представлено в табл. 20.

Таблица 20 – Переменные модуля БТ75-404

Переменная модуля	Описание
`BOOL DAC_OUT[1-4]`	Установка выходного аналогового сигнала каналов мА, умноженных на 1000. (значение 4000 соответствует 4мА).
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

7.6.2.4. Модуль вывода аналоговых сигналов напряжения БТ75-404А

Редактор модуля вывода аналоговых сигналов напряжения БТ75-404А с четырьмя каналами (рис. 170), позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;

– «Адрес модуля на шине».

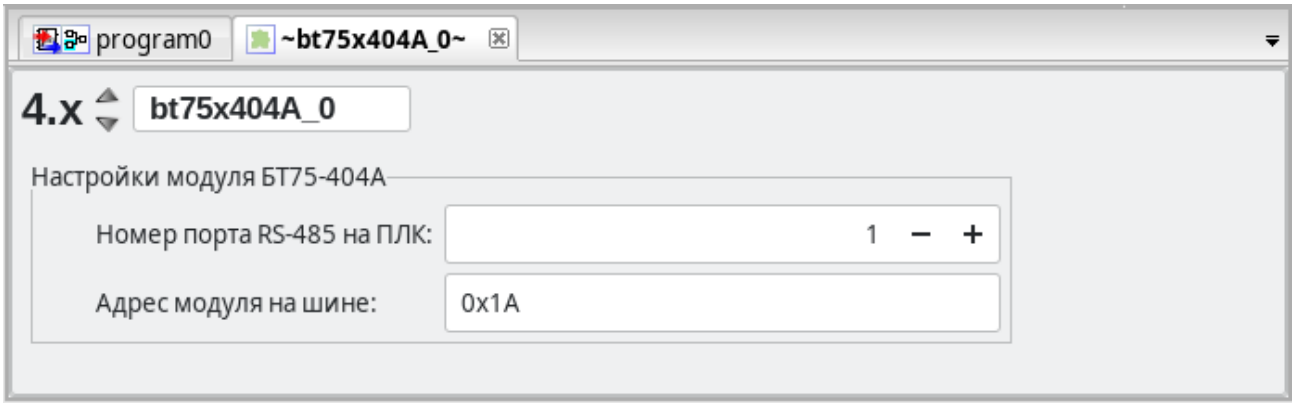


Рисунок 170 – Редактор модуля дискретного вывода БТ75-404А

Описание переменных модуля, доступных для программы, представлено в табл. 21.

Таблица 21 – Переменные модуля БТ75-404А

Переменная модуля	Описание
`BOOL DAC_OUT[1-4]`	Установка выходного аналогового сигнала каналов в Вольтах, умноженных на 1000. (значение 4000 соответствует 4В).
`BOOL STATUS_OK`	Статус работоспособности модуля. True — работает, False — нет.

Поля переменных модуля БТ75-404А представлены на рис. 171.

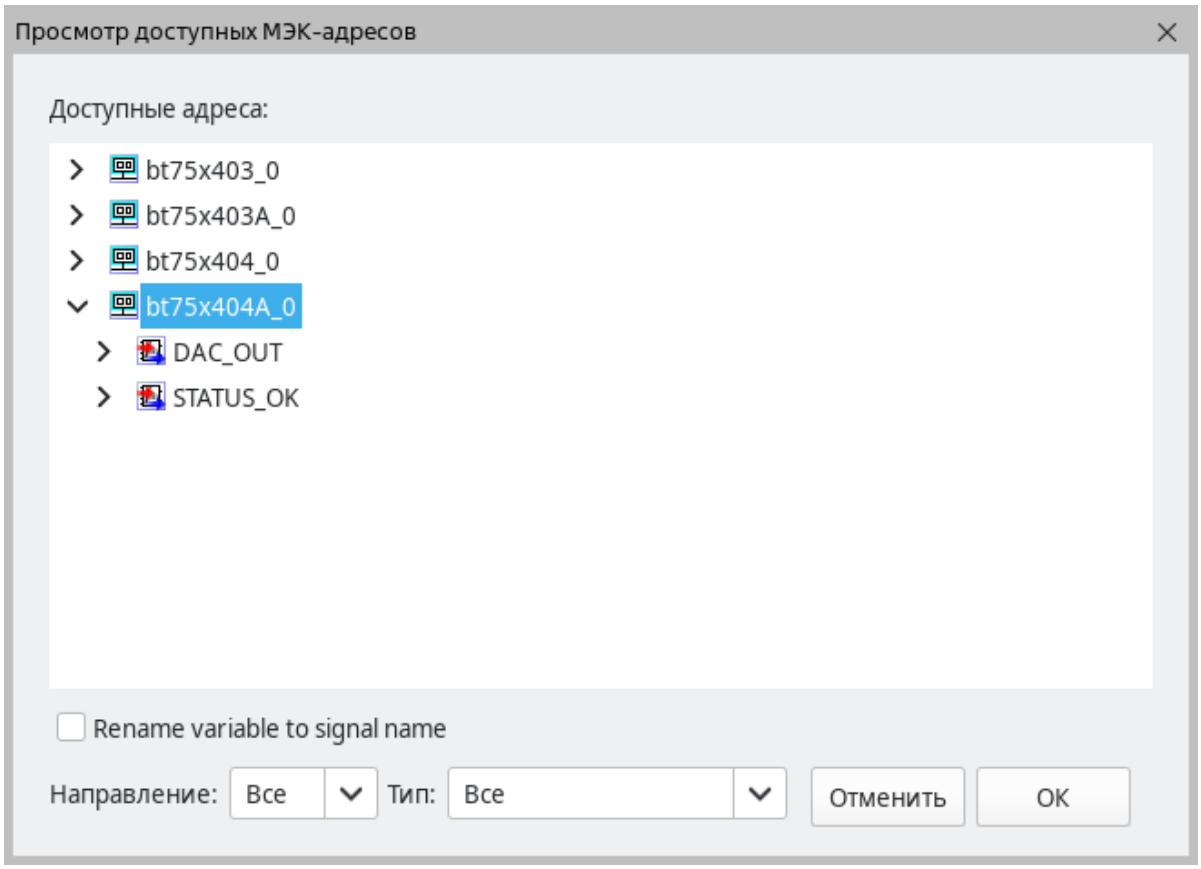


Рисунок 171 – Поля переменных модуля БТ75-404А

7.6.3. Модуль коммуникационный БТ75-251

Редактор коммуникационного модуля БТ75-251 с четырьмя каналами, позволяет задать:

– «Номер порта RS-485 на ПЛК» данного модуля;
– «Адрес модуля на шине»;
– Количество запросов в секунду для режима виртуального последовательного порта;
– Режим работы каждого из 4 портов.

Для каждого последовательного порта коммуникационного модуля БТ75-251 может быть задан один из двух вариантов работы:

– Режим ретранслятора (`Custom`);
– Режим виртуального последовательного порта (`VirtualPty`).

При работе в режиме ретранслятора взаимодействие с портом происходит через функции:

– `BT75X251_SEND` – отправка строки в удалённый порт. Аргументы: идентификатор порта, строка для отправки, возвращаемое значение True, в случае ошибки.

– `BT75X251_RECV` – чтение строки из удалённого порта. Аргументы: идентификатор порта, строка, в которую запишется значение.

– `BT75X251_AVAIL` – количество байт доступное для чтения из буфера модуля. Аргументы: идентификатор и число, в которое запишется количество доступных байтов.

Все функции возвращают значение «`True`» в случае ошибки.

Идентификатор порта необходимо связать с переменной в проекте.

При работе в режиме виртуального последовательного порта взаимодействие с удалёнными портами происходит путем задания имени устройства «`vpty<n>`», где `<n>` – номер виртуального порта, уникальный для проекта. После указания виртуального порта в настройках порта коммуникационного модуля, тот же порт следует указать в настройках модуля, работающего через последовательный порт, например, Modbus RTU (например, «`/dev/aux0`» заменить на «`vpty<0>`»).

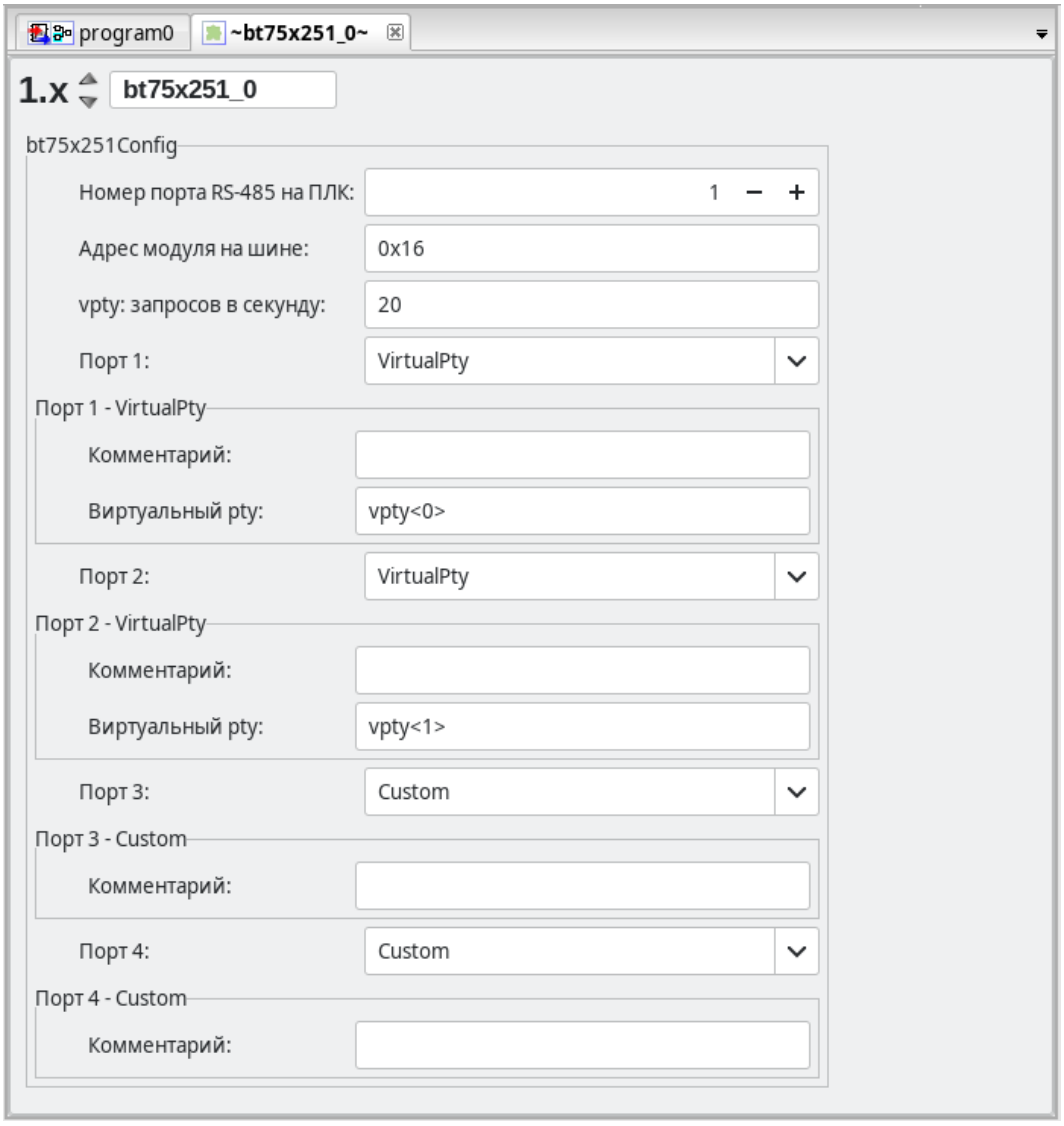


Рисунок 172 – Редактор коммуникационного модуля БТ75-251

7.6.4. Модуль Modbus TCP Slave

Плагин модуля Modbus TCP Slave реализует поддержку библиотеки поддержки протокола Modbus TCP ЮКСУ.91165-01 по сети Ethernet.

Редактор модуля (рис. 173), позволяет задать:

– «IP адрес», на котором будет запущен Modbus TCP Slave (по умолчанию – 0.0.0.0);
– «TCP порт», на котором будет запущен Modbus TCP Slave;
– «Максимальное количество TCP клиентов», которое будет обрабатывать сервер;
– «Используемые выходные биты (Coils)» - количество переменных типа Coils, доступных по протоколу;
– «Используемые входные биты (Discrete Inputs)» - количество переменных типа Discrete Inputs, доступных по протоколу;
– «Используемые выходные регистры (Holding Registers)» - количество переменных типа Holding Registers, доступных по протоколу;
– «Используемые входные регистры (Input Registers)» - количество переменных типа Input Registers, доступных по протоколу.

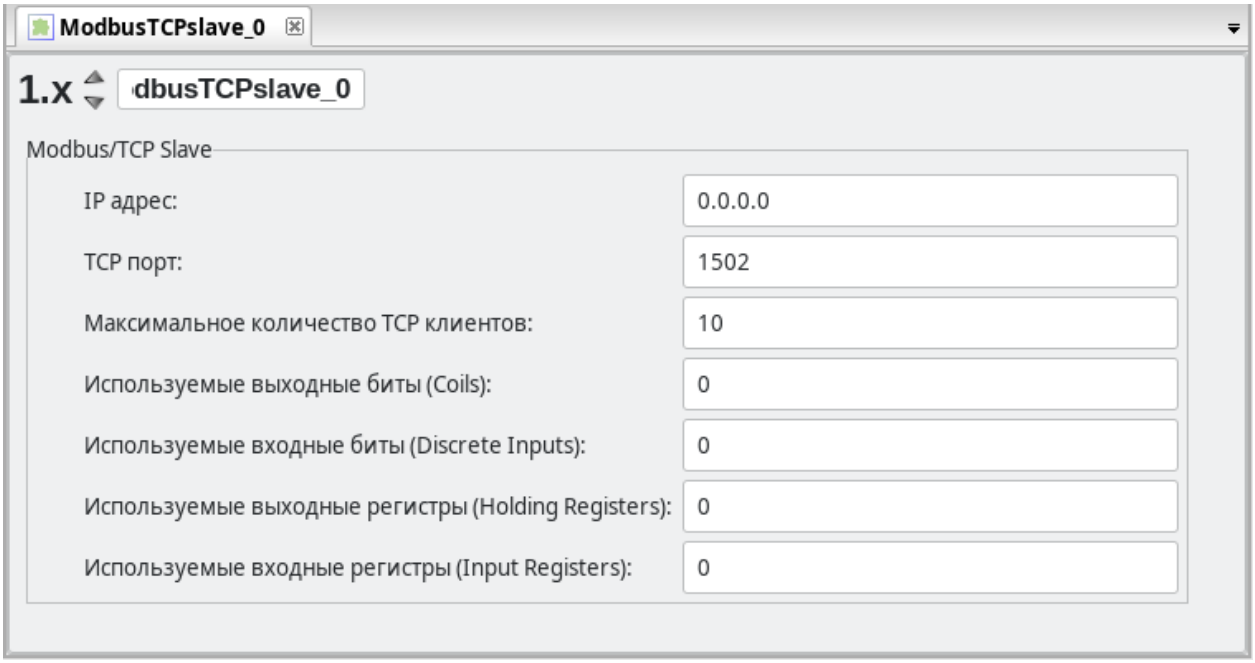


Рисунок 173 – Редактор модуля Modbus TCP Slave

7.6.5. Модуль Modbus TCP Master

Плагин модуля Modbus TCP Master реализует поддержку библиотеки поддержки протокола Modbus TCP ЮКСУ.91165-01 по сети Ethernet.

Редактор модуля (см. рис. 174), позволяет задать:

– «IP адрес» Modbus TCP Slave, к которому будет обращаться Master;
– «TCP порт», на котором запущен удалённый Modbus TCP сервер;
– «Используемые выходные биты (Coils)» - количество переменных типа Coils, доступных по протоколу;
– «Используемые входные биты (Discrete Inputs)» - количество переменных типа Discrete Inputs, доступных по протоколу;
– «Используемые выходные регистры (Holding Registers)» - количество переменных типа Holding Registers, доступных по протоколу;
– «Используемые входные регистры (Input Registers)» - количество переменных типа Input Registers, доступных по протоколу.
– «Таймаут милисек» - таймаут.

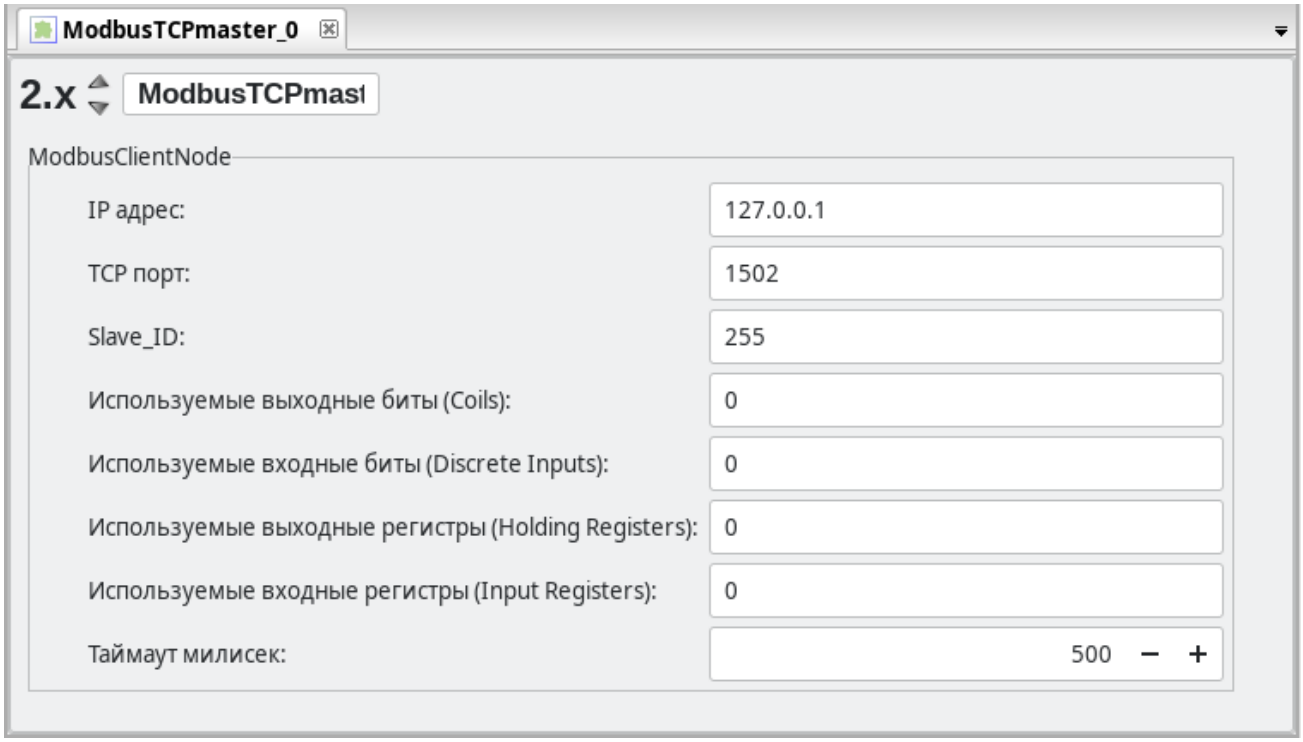


Рисунок 174 – Редактор модуля Modbus TCP Master

7.6.6. Модуль Modbus RTU Slave

Плагин модуля Modbus RTU Slave реализует поддержку библиотеки поддержки протокола Modbus RTU ЮКСУ.91172-01 по последовательному порту.

Редактор модуля (рис. 175), позволяет задать:

– «Последовательный порт» - мнемоническое наименование устройства последовательного порта в ОС РВ, например: `/dev/aux0`;
– «Скорость» - скорость передачи данных (бод);
– «Чётность» - необходимость контроля чётности;
– «Бит данных» - количество бит данных;
– «Стоп-биты»;
– «Идентификатор сервера» - уникальный идентификатор сервера в соответствии с протоколом ModbusRTU;
– «Используемые выходные биты (Coils)» - количество переменных типа Coils, доступных по протоколу;
– «Используемые входные биты (Discrete Inputs)» - количество переменных типа Discrete Inputs, доступных по протоколу;
– «Используемые выходные регистры (Holding Registers)» - количество переменных типа Holding Registers, доступных по протоколу;
– «Используемые входные регистры (Input Registers)» - количество переменных типа Input Registers, доступных по протоколу.

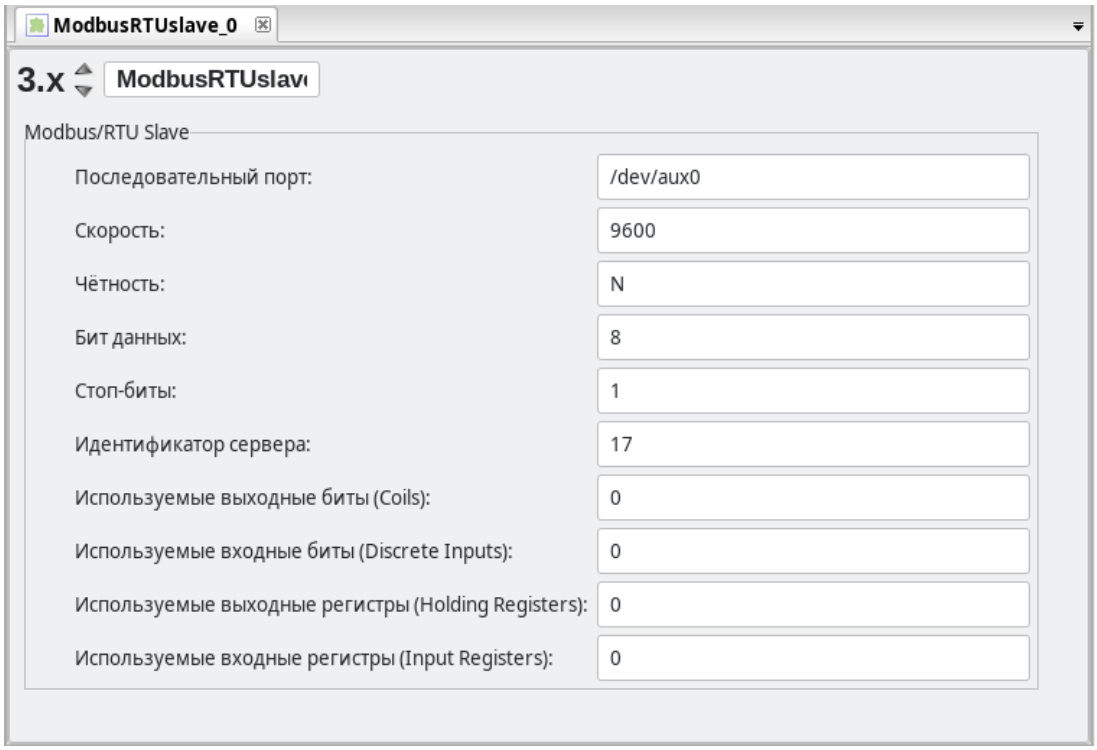


Рисунок 175 – Редактор модуля Modbus RTU Slave

7.6.7. Модуль Modbus RTU Master

Плагин модуля Modbus RTU Master реализует поддержку библиотеки поддержки протокола Modbus RTU ЮКСУ.91172-01 по последовательному порту.

Редактор модуля (рис. 176), позволяет задать параметры Modbus RTU Master.

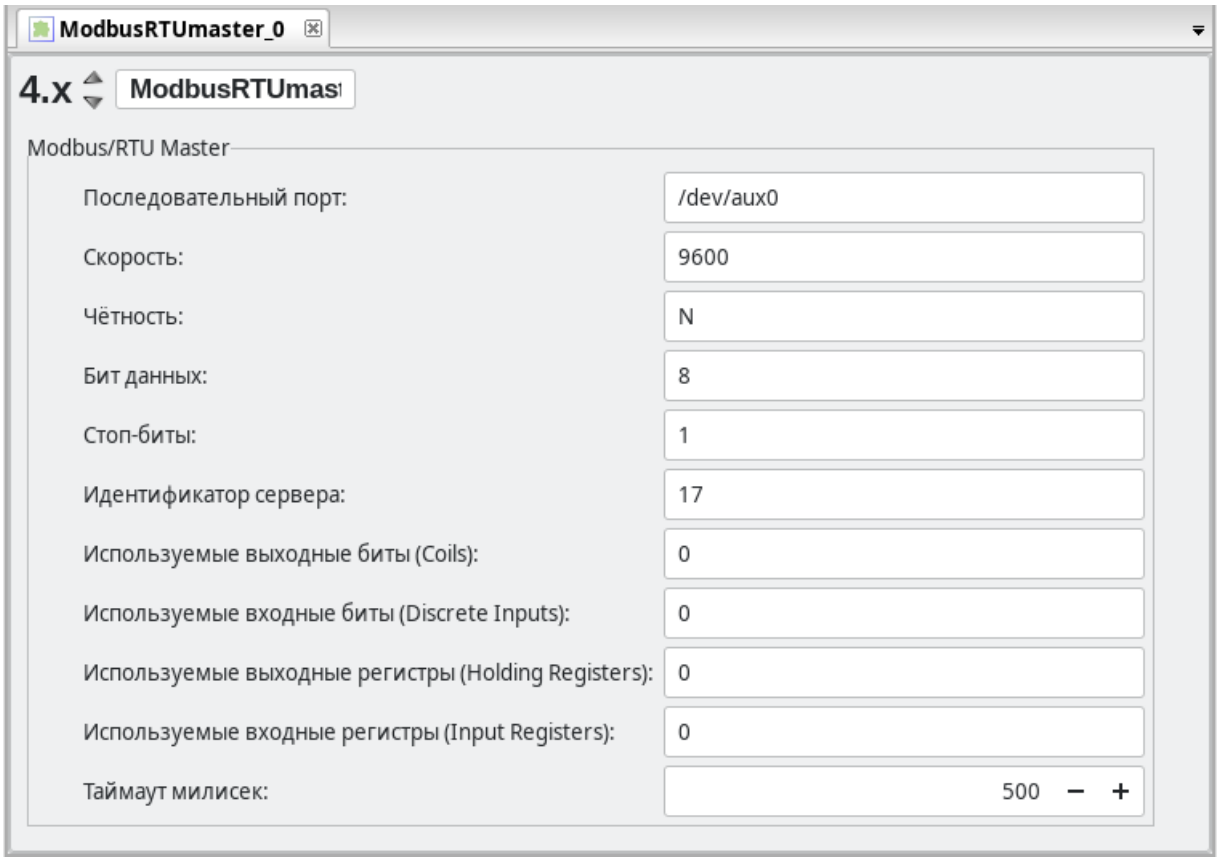


Рисунок 176 – Редактор модуля Modbus RTU Master

7.6.8. Модуль IEC 60870-5-101 Slave

Плагин модуля IEC 60870-5-101 Slave реализует ведомое устройство (Slave), то есть принимает запросы и команды от ведущего. В зависимости от команд он либо возвращает значения своих переменных, либо выполняет команду, которая пришла от клиента. Работает по последовательному порту.

Редактор модуля (рис. 177), позволяет задать:

– «Последовательный порт» - мнемоническое наименование устройства последовательного порта в ОС РВ, например, /dev/aux0;
– «Адрес» - адрес устройства в соответствии с протоколом IEC 60870-5-101;
– «Скорость» - скорость передачи данных (бод);
– «Чётность» - необходимость контроля чётности;
– «Бит данных» - количество бит данных;
– «Стоп-биты»;

– «Адрес удалённого устройства» в соответствии с протоколом IEC 60870-5-101;
– «Обрабатывать команду опроса» - если опция выбрана, то устройство будет принимать и обрабатывать команды опроса, в противном случае они будут игнорироваться;
– «Обрабатывать команду синхронизации часов» - если опция выбрана, то устройство будет принимать выполнять команды установки системного времени на устройстве;
– «Обрабатывать команды от мастера» - если опция выбрана, то устройство будет принимать от ведущего устройства (мастера) команды и помещать их аргументы в соответствующие переменные.

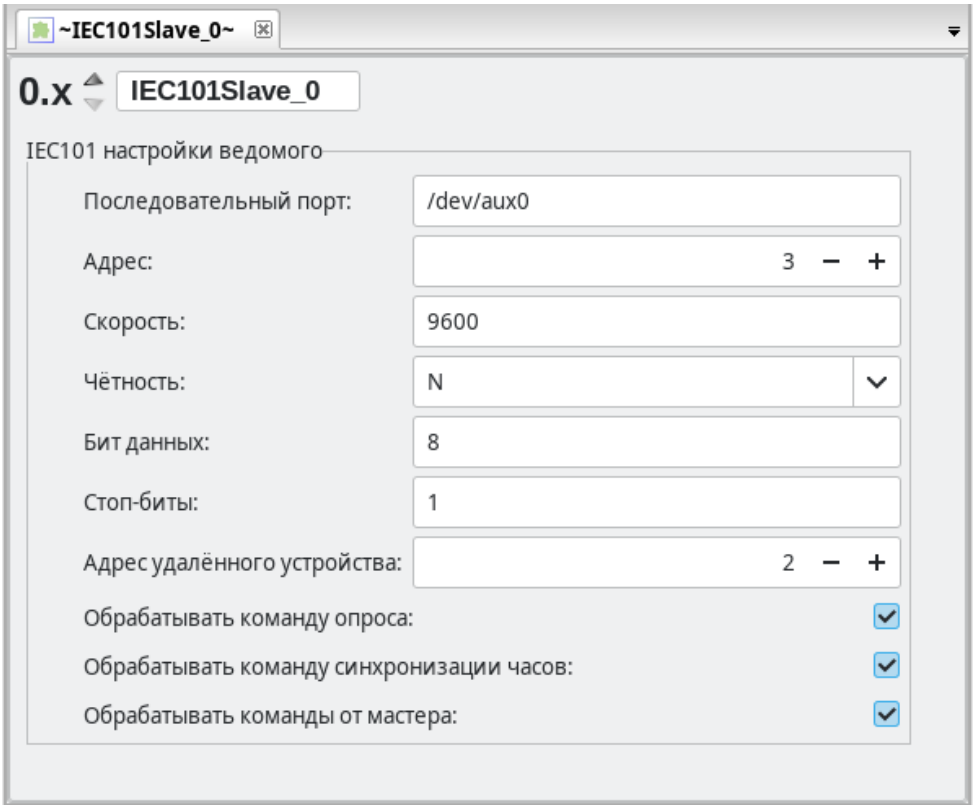


Рисунок 177 – Редактор модуля IEC 60870-5-101 Slave

Через контекстное меню узла возможно добавление информационного объекта (рис. 178).

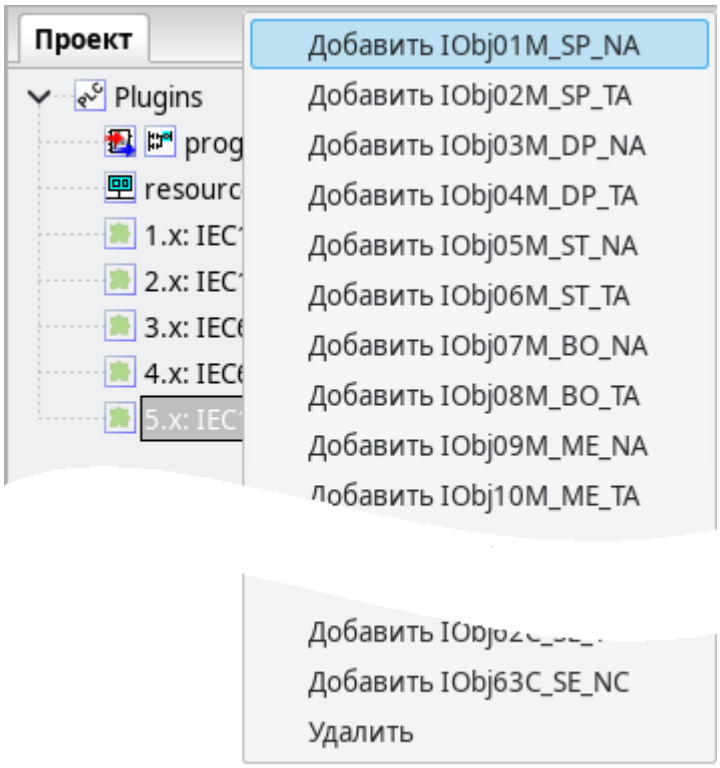


Рисунок 178 – Информационные объекты модуля IEC 60870-5-101

Номера и типы информационных объектов соответствуют МЭК 60870-5-101.

Объект измерения – информационный объект, который отправляет данные на мастер (по запросу, при изменении или по команде опроса).

Параметры объекта измерения представлены на рис. 179.

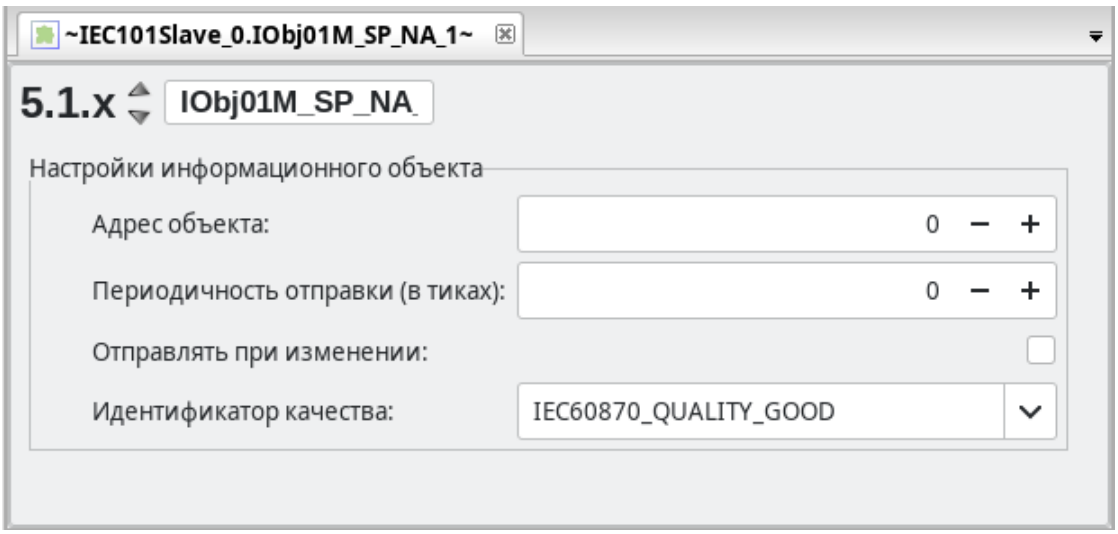


Рисунок 179 – Информационный объект измерения

Параметры объекта измерения:

– «Адрес объекта» - адрес информационного объекта в соответствии с протоколом IEC 60870-5-101;
– «Периодичность отправки (в тиках)»: периодичность отправки, 0 – не отправляется никогда, значение – периодичность в системных тиках;
– «Отправлять при изменении» - значение отправляется спорадически по изменению переменной;
– «Идентификатор качества» согласно стандарту IEC 60870-5-101.

Информационные объекты типа “команды” связаны с входными переменными. При получении команды от ведущего аргументы команды помещаются в связанную с модулем переменную и доступны в программе МЭК.

Параметры объекта команды согласно стандарту МЭК 60870-5-101 (рис. 180):

– «Адрес объекта» - адрес информационного объекта в соответствии с протоколом IEC 60870-5-101.

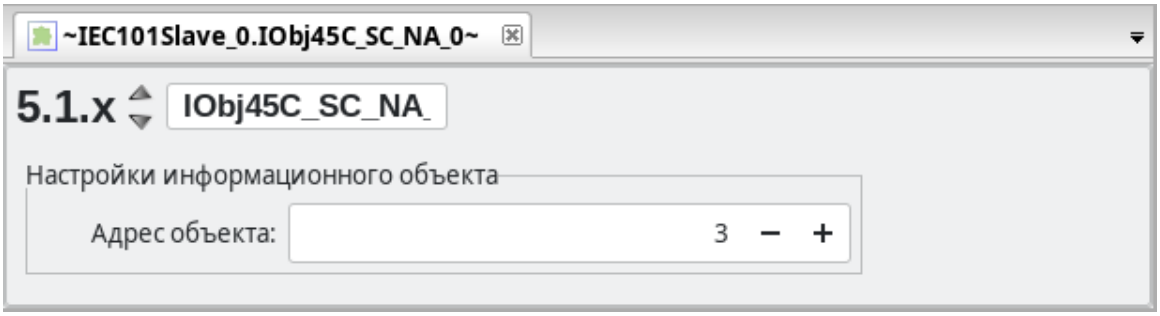


Рисунок 180 – Информационный объект команды

7.6.9. Модуль IEC 60870-5-104 Slave

Плагин модуля IEC 60870-5-104 Slave реализует ведомое устройство (slave), то есть только принимает запросы и команды от ведущего. В зависимости от команд он либо возвращает значения своих переменных, либо выполняет команду, которая пришла от клиента. Работает по TCP.

Редактор модуля (см. рис. 181), позволяет задать:

– «IP адрес» - адрес сервера;
– «TCP порт» - порт сервера;
– «Common_Address» - адрес сервера по протоколу МЭК 60870-5-104;
– «t0», «t1», «t2», «t3», «w», «k» - таймауты соединения в соответствии с протоколом МЭК 60870-5-104;
– «Allways_Apply_Time» - если опция выбрана, то устройство будет принимать выполнять команды установки системного времени на устройстве, не зависимо от статуса обновления времени по протоколу NTP.

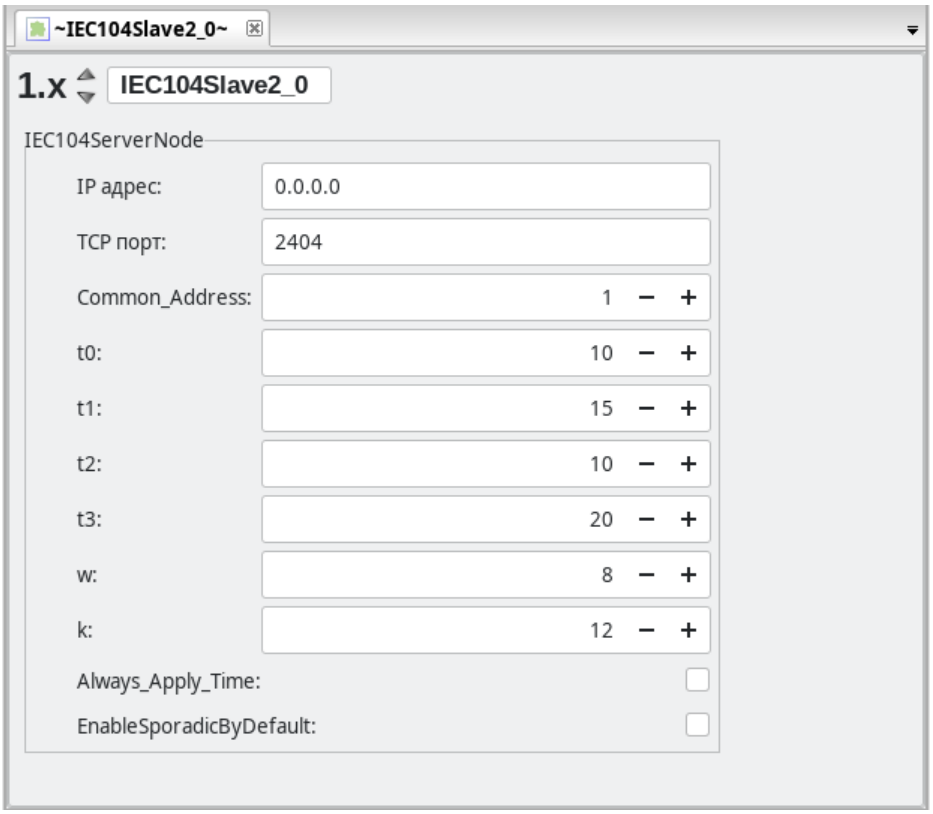


Рисунок 181 – Редактор модуля IEC 60870-5-104 Slave

Через контекстное меню узла возможно добавление информационного объекта (см. рис. 183). Параметры объекта информации согласно стандарту МЭК 60870-5-104 (рис. 182):

– «Адрес объекта» - адрес информационного объекта в соответствии с протоколом IEC 60870-5-104.

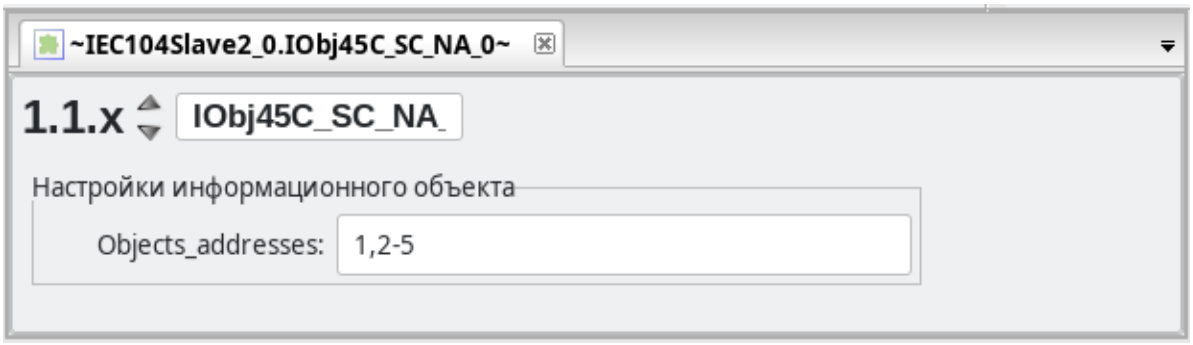


Рисунок 182 – Информационный объект команды

Каждый информационный объект содержит две переменные: значение информационного объекта и описатель качества.

Помимо информационных объектов в контекстном меню есть объект Control. Объект Control содержит две переменные:

– SporadicCtrl – тип BOOL, предназначена для управления включением спорадической отправки, если установлено значение TRUE, то для всех переменных будет производиться отправка при изменении значения;
– ConnetctedClients – тип SINT, предназначена для получения количества подключённых клиентов.

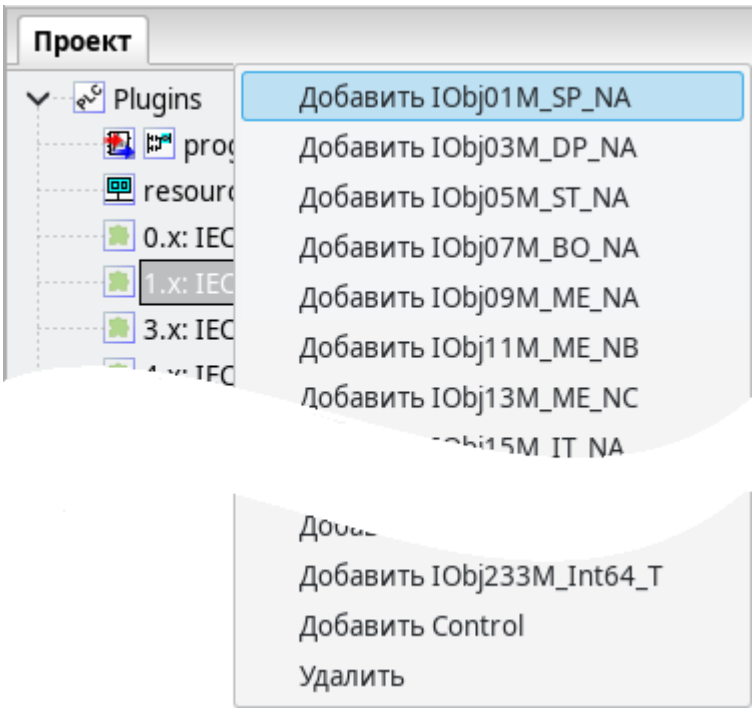


Рисунок 183 – Информационные объекты модуля IEC 60870-5-104 Slave

7.6.10. Модуль IEC 60870-5-104 Master

Плагин модуля IEC 60870-5-104 Master реализует ведущее устройство.

Редактор модуля, позволяет задать:

– «IP адрес» - адрес сервера;
– «TCP порт» - порт сервера;
– «Common_Address» - адрес сервера по протоколу МЭК 60870-5-104;
– «t0», «t1», «t2», «t3», «w», «k» - таймауты соединения в соответствии с протоколом МЭК 60870-5-104;
– «Sent_Interrogation_At_Connect» - отправка команды общего опроса при подключении к удалённому устройству;
– «Sent_Interrogation_Every_Sec» - отправка команды общего опроса с периодом в секундах, при 0 отправка не осуществляется;
– «Apply_Time_At_Connect» - отправка команды установки времени при подключении к удалённому устройству;

– «Apply_Time_Every_Sec» - отправка команды установки времени с периодом в секундах, при 0 отправка не осуществляется.

Редактор модуля IEC 60870-5-104 Master представлен на рис. 184.

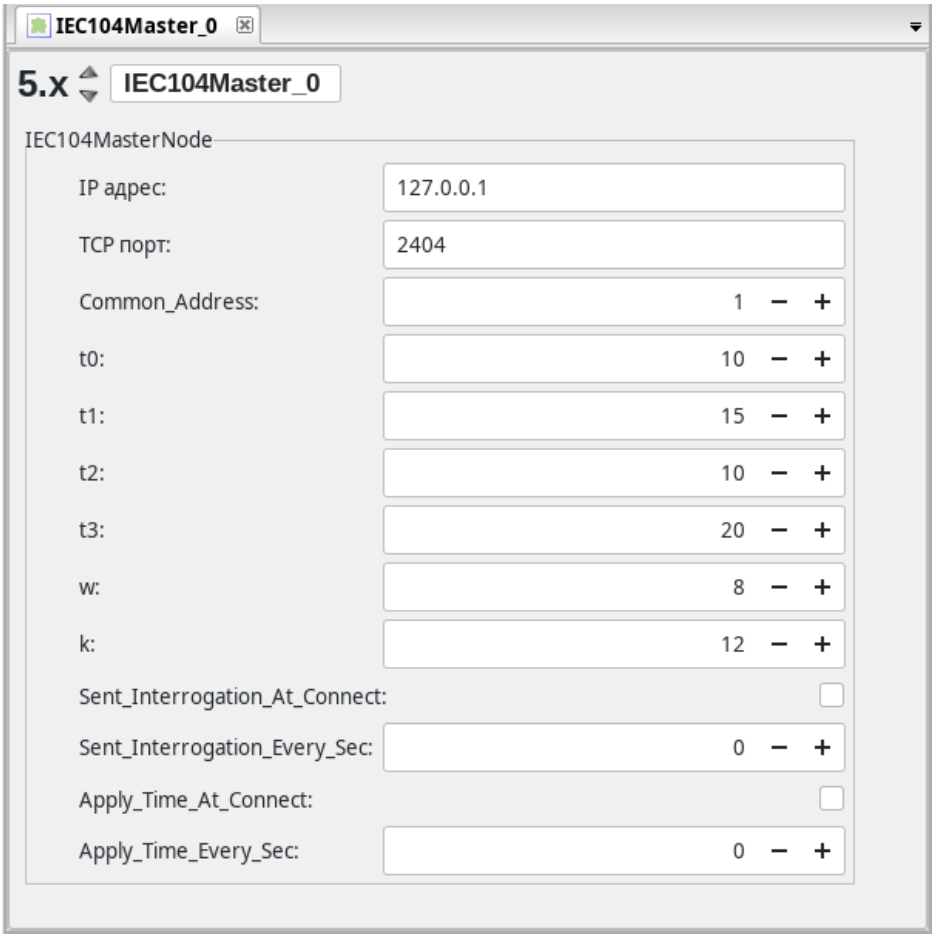


Рисунок 184 – Редактор модуля IEC 60870-5-104 Master

Через контекстное меню узла возможно добавление информационного объекта. Параметры объекта информации согласно стандарту МЭК 60870-5-104:

– «Адрес объекта» - адрес информационного объекта в соответствии с протоколом IEC 60870-5-104.

На рис. 185 изображен информационный объект команды.

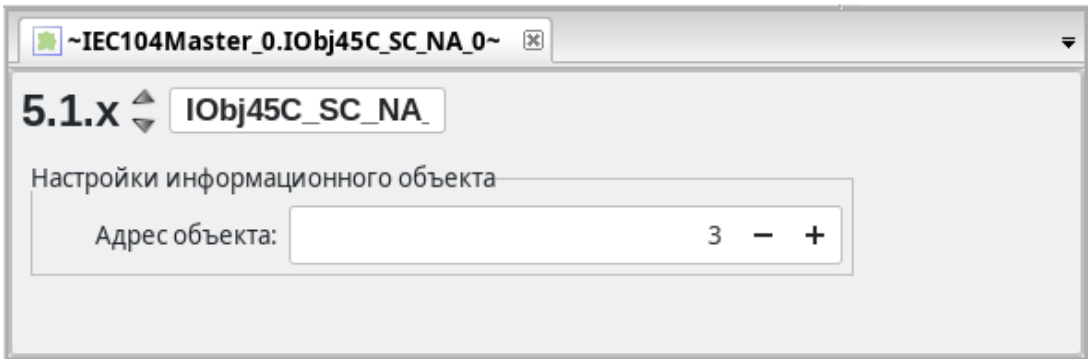


Рисунок 185 – Информационный объект команды

Каждый информационный объект содержит две переменные: значение информационного объекта и описатель качества.

На рис. 186 представлены информационные объекты модуля IEC 60870-5-104 Master.

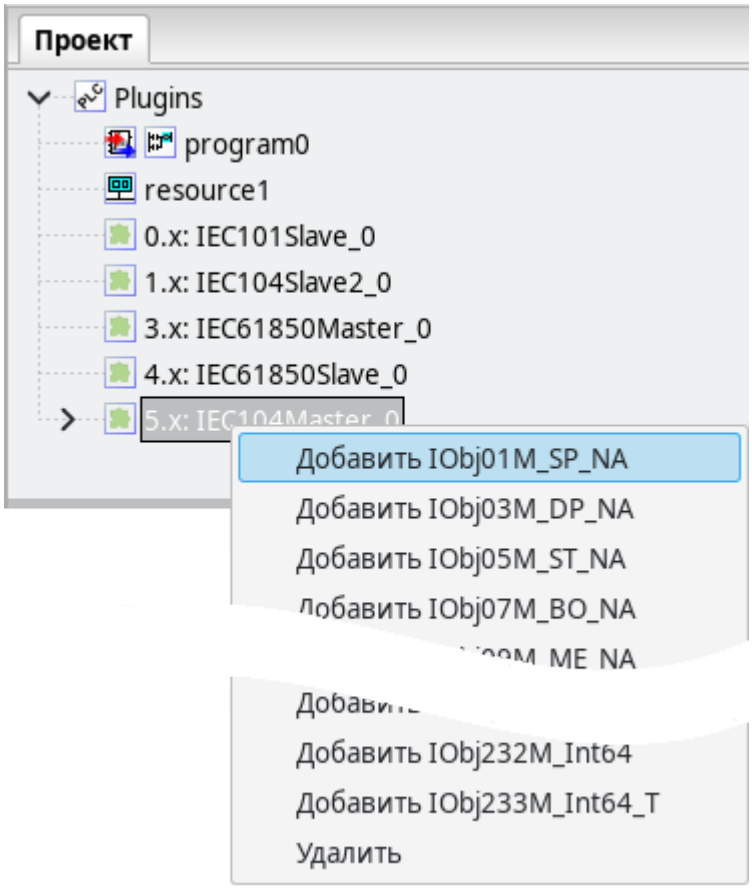


Рисунок 186 – Информационные объекты модуля IEC 60870-5-104 Master

7.6.11. Модуль OPC UA Сервер

Плагин модуля OPC UA реализует библиотеку поддержки протокола OPC UA ЮКСУ.91173-01. Работает по протоколу TCP.

Редактор модуля (рис. 187), позволяет задать:

– «IP адрес» - адрес сервера.

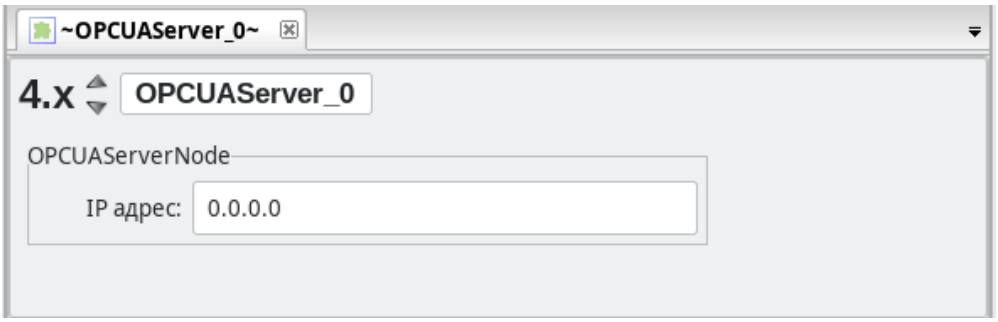


Рисунок 187 – Редактор модуля OPC UA Сервер

Плагин поддерживает узлы, указанные на рис. 188.

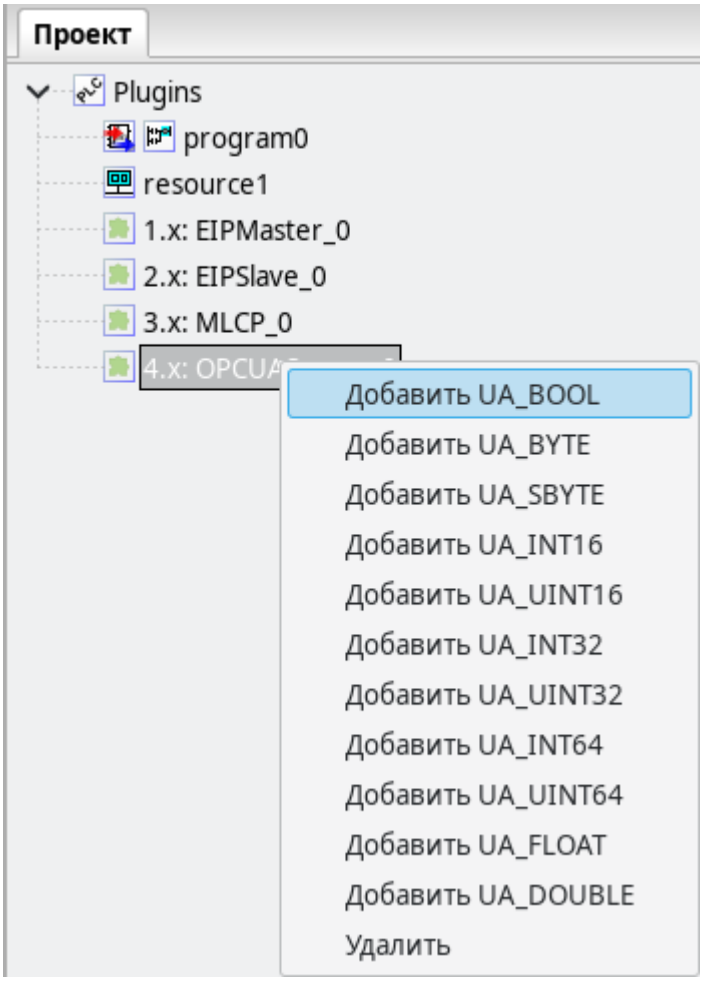


Рисунок 188 – Узлы модуля OPC UA Сервер

Каждый узел модуля имеет параметры, представленные на рис. 189.

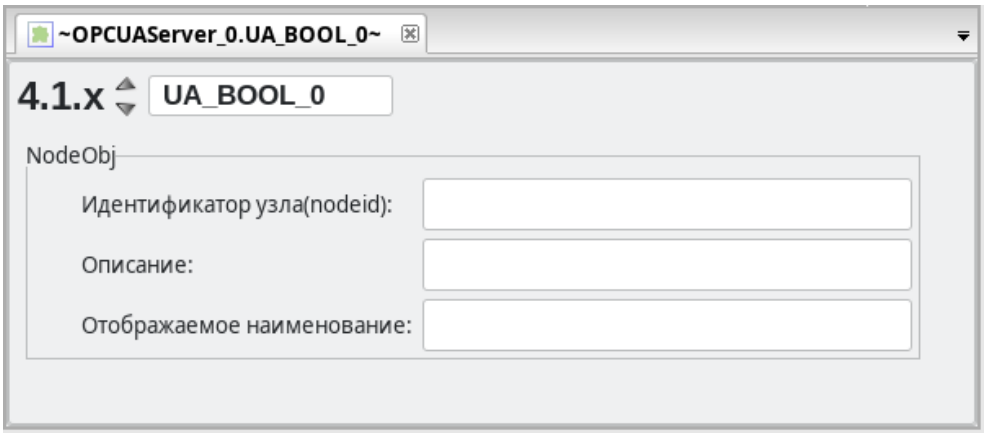


Рисунок 189 – Параметры узла модуля OPC UA

Параметры узла модуля соответствуют атрибутам объекта в протоколе OPC UA:

– «Идентификатор узла (nodeid)» - именование объекта;
– «Описание» - произвольное описание объекта;
– «Отображаемое наименование» - именование, отображаемое в клиенте.

7.6.12. Модуль Ethernet/IP (EIP Master)

Плагин модуля EIP Master реализует ведущее устройство протокола Ethernet/IP.

Редактор модуля, позволяет задать:

– «IP адрес» - адрес ведомого устройства;
– «TCP порт» - порт ведомого устройства;
– «Timeout» - таймаут подключения для медленных запросов в секундах;
– «Path_Segment_8_bit» - использовать размер сегмента равный 8 bit;
– «IOTimeout» - таймаут для быстрых запросов в миллисекундах;

Через контекстное меню узла возможно добавление медленного параметра (ExplicitParameter) и быстрого соединения (ImplicitConnection).

Редактор модуля Ethernet/IP представлен на рис. 190.

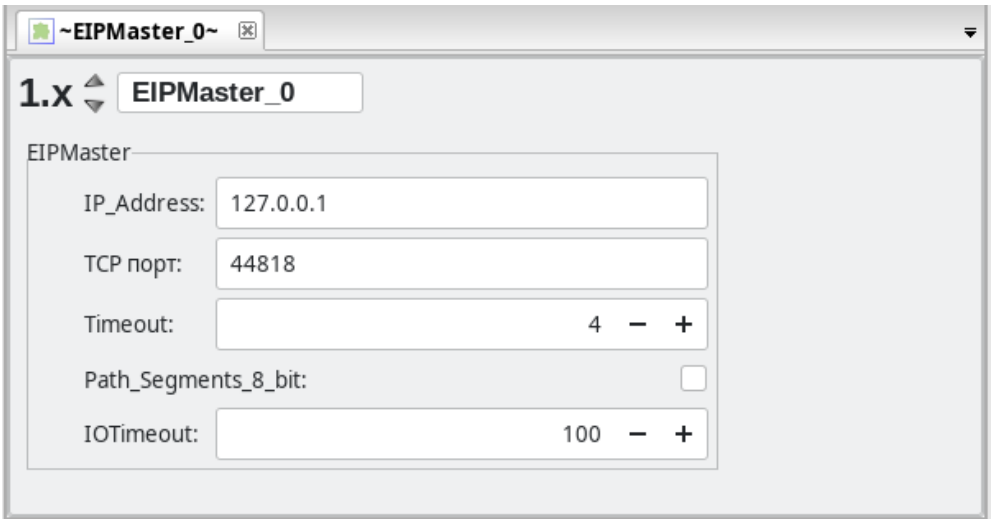


Рисунок 190 – Редактор модуля Ethernet/IP

Параметры ExplicitParameter (рис. 191):

– Path - путь в соответствии с конфигурацией устройства;
– ElementsNumber - количество параметров, получаемых/отправляемых в запросе;
– TypeFormat - типы данных параметров. Могут быть перечислены через запятую или пробел;
– Service - тип запроса запись или чтение;
– Code - код запроса, может быть задан вручную, если требуется нестандартный код.

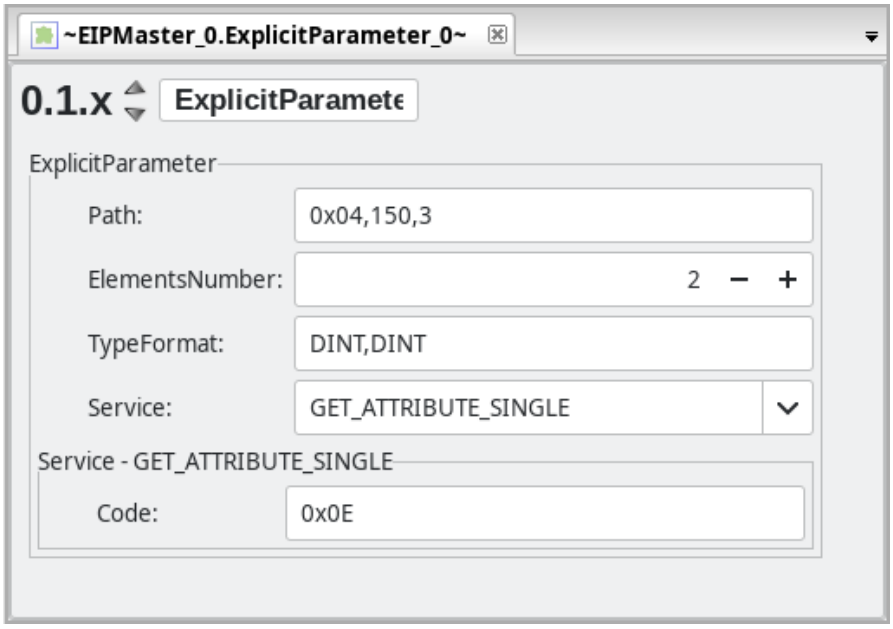


Рисунок 191 – Конфигурация ExplicitParameter

Параметры ImplicitConnection (рис. 192):

– Path - путь в соответствии с конфигурацией устройства;
– Input_Num - количество входных параметров;
– Input_Format - типы данных входных параметров. Могут быть перечислены через запятую или пробел;
– Output_Num - количество выходных параметров;
– Output_Format - типы данных выходных параметров. Могут быть перечислены через запятую или пробел;
– Out_Request_Packet_Interval - интервал запросов записи в микросекундах;
– In_Request_Packet_Interval - интервал запросов чтения в микросекундах;
– Network_Params_Mask - маска сетевых параметров;
– Trigger_and_Transport_Mask - маска сервисных параметров.

Параметры следует взять из EDS конфигурационного файла устройства, поставляемого производителем.

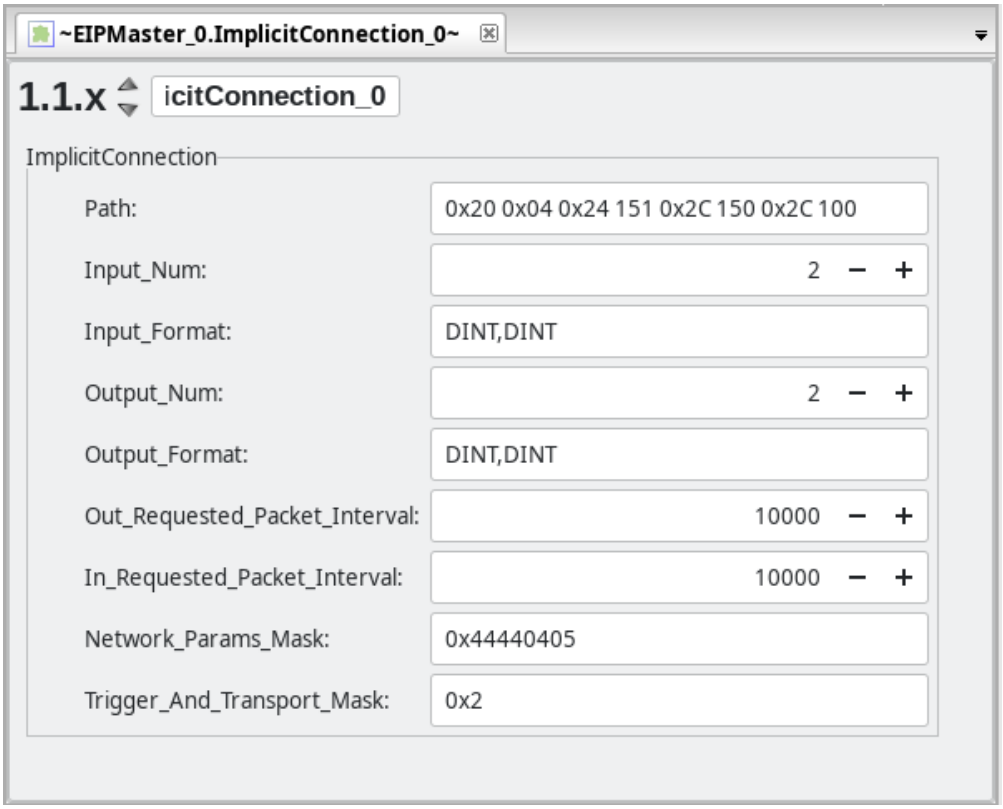


Рисунок 192 – Конфигурация ImplicitConnection

7.6.13. Модуль MLCP расширение

Плагин модуля MLCP расширения реализует поддержку отправки и получения тегов по протоколу Monitoring Library Communication Protocol (ЮКСУ.91168-01 33 01). Плагин позволяет, как забирать (переменные типа «вход»), так и отправлять (переменные типа «выход» или «память») теги MLCP. Работает по TCP.

Редактор модуля (рис. 193), позволяет задать:

– «IP адрес» адрес сервера MLCP;
– «URL чтения тегов»;
– «URL записи тегов»;
– «Идентификатор контроллера»;
– «Периодичность отправки (в тиках)» - периодичность (в тиках) отправки и получения тегов с сервера.

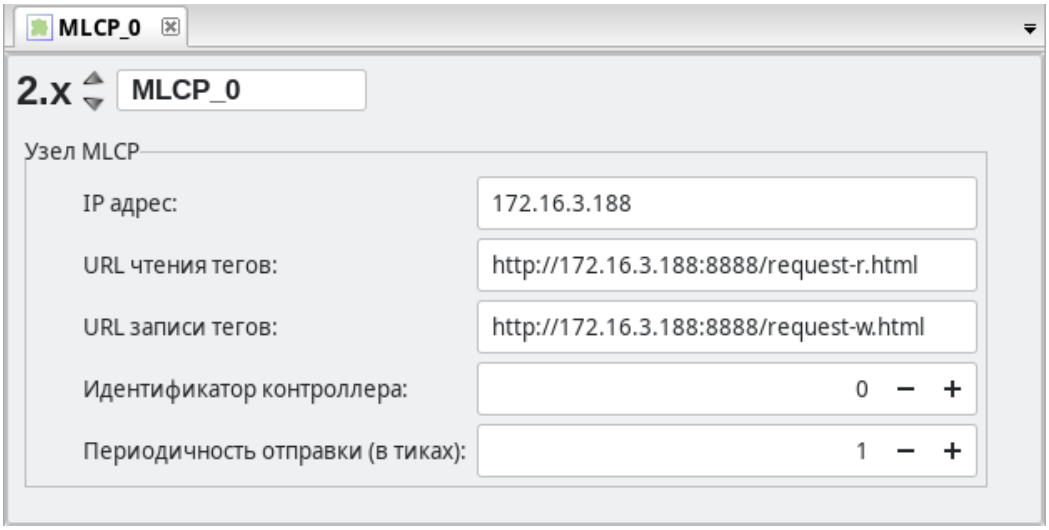


Рисунок 193 – Редактор модуля MLCP

Плагин поддерживает добавление тегов (рис. 194).

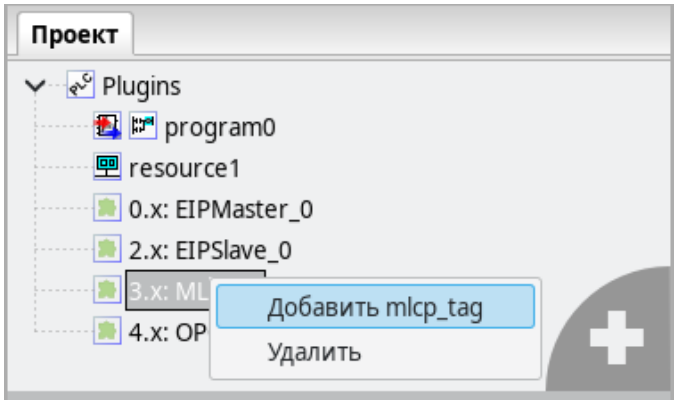


Рисунок 194 – Узел модуля MLCP

Каждый узел с описанием тега имеет параметры, представленные на рис. 195.

Рисунок 195 – Параметры узла тега MLCP

Параметры узла тега:

– «Имя тега».

7.6.14. Модуль NTP конфигурация

Плагин модуля NTP реализует поддержку получения и установки времени по протоколу NTP. Плагин позволяет запрашивать время от NTP сервера с заданным интервалом времени (в сек).

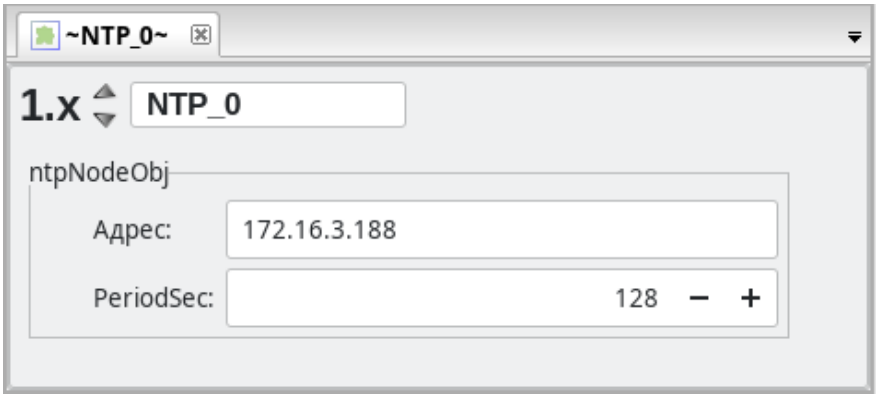


Рисунок 196 – Редактор модуля NTP

Параметры:

– «Адрес» - IP-адрес NTP-сервера;

– «PeriodSec» - интервал времени (в сек) между запросами.

7.6.15. Модуль сетевого экрана

Плагин модуля сетевого экрана (Firewall) позволяет ограничивать доступ по локальной сети для заданных ip адресов.

Параметры узла модуля позволяют задать правила по умолчанию для обработки входящих (параметр FireWall - «InDefaultDeny») и исходящих соединений (параметр FireWall - «OutDefaultDeny»).

Узел fw_open_port позволяет открыть порт для входящих соединений.

В параметрах узла можно указать ip адрес сетевого интерфейса, на котором будут ожидаться входящие соединения, и номер порта.

Узел fw_filter_in позволяет задать правило для входящих соединений.

В параметрах узла можно указать ip адрес источника соединения, порт для подключения, а также режим – разрешено или запрещено.

Узел fw_filter_out позволяет задать правило для исходящих соединений.

В параметрах узла можно указать ip адрес и порт для подключения, а также режим – разрешено или запрещено.

Узел fw_filter_fw позволяет задавать правила перенаправления соединений. В параметрах узла можно задать ip адрес и порт, на котором будет ожидаться соединение, и ip адрес и порт, на которые будет перенаправлено подключение.

7.6.16. Набор специализированных типов Aggregation

Набор специализированных типов Aggregation предназначен для объединения нескольких переменных в массив для последующей работы с ними через интерфейс массива.

Начальное значение агрегационной переменной должно быть указано в формате [n], где n - количество переменных, идущих следом за агрегационной переменной. То есть, если агрегационная переменная A инициализируется со значением по умолчанию [3], и после неё объявлены переменные A1, A2, A3, то в программе обращение к переменной A1 должно осуществляться как A[1], к A2 как A[2] и т.д. При этом нельзя напрямую изменять значения переменных, объединенных в массив.

Типы Aggregation делятся на следующие подгруппы. Типы подгруппы Aggregation могут содержать не более 100 элементов. Типы подгруппы Aggregation500 могут содержать не более 500 элементов. Типы подгруппы Aggregation1000 могут содержать не более 1000 элементов. Такое разделение возникло связи с тем что в языке ST можно определять массивы только статического размера. Поэтому для переменной типа Aggregation500 будет выделено 500 элементов, не зависимо от того, сколько из них используется.

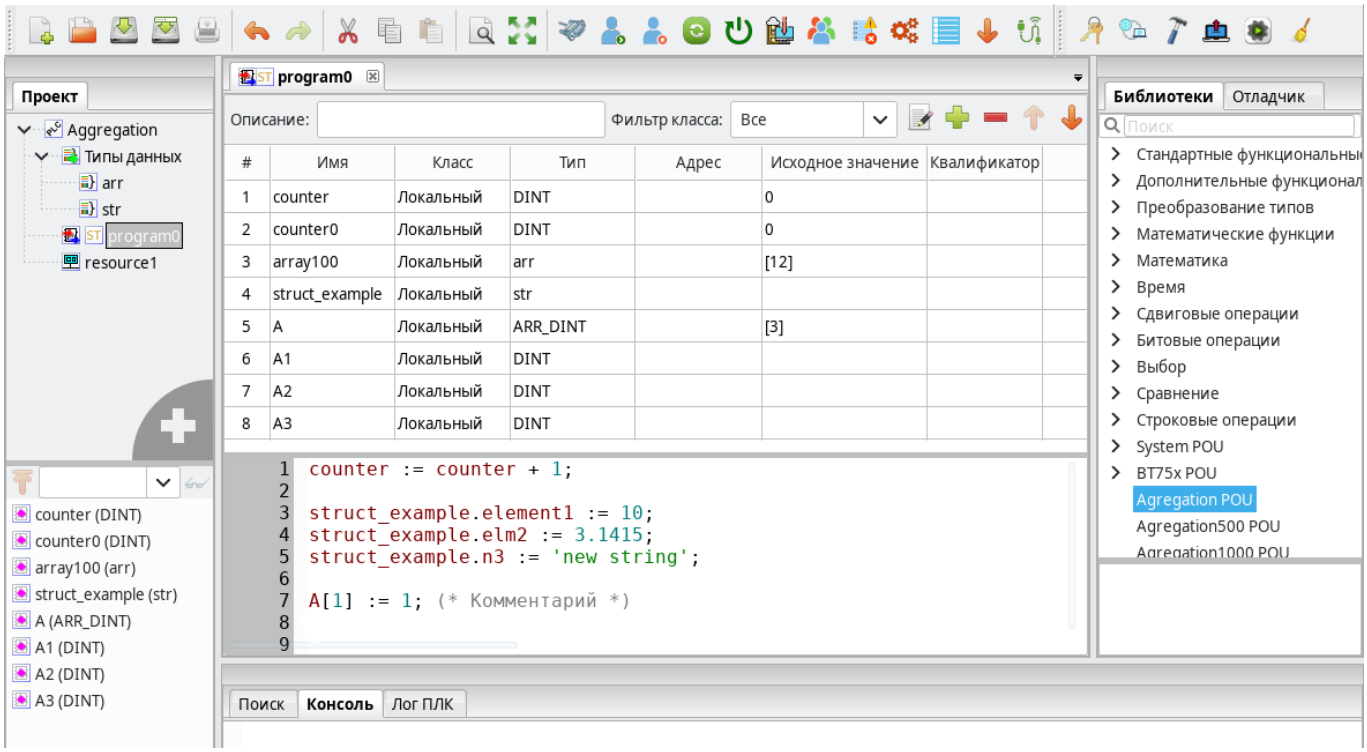


Рисунок 197 – Типы Aggregation

7.6.17. Функции для работы с переменными на языке Си

На языке ST есть возможность добавлять программные вставки на языке Си.

Текущая реализация СРиО позволяет делать вставки не длиннее одной строки.

Программные вставки на языке Си выделяются фигурными скобками. Пример применения вставок представлен ниже:

```
{printf("Insert C\n");}
```

Для удобства работы с переменными языка ST при программировании вставок на языке Си в программу СРиО добавлены специальные функции `GetFbVar()`, `SetFbVar()`. Имя переменной в данном случае нужно писать большими буквами. Для глобальных переменных и переменных, с которыми связан адрес, функция возвращает указатель, а не значение, поэтому для получения их значения нужно разыменовать указатель.

`GetFbVar(VAR)` — функция возвращает значение переменной `var`. Имя переменной нужно писать большими буквами. Для глобальных переменных и переменных, с которыми связан адрес, функция возвращает указатель, а не значение, поэтому для получения их значения нужно разыменовать указатель.

Примеры вызова:

Обычные переменные:

```
{printf("last_time = %d, average_time = %d\n",
GetFbVar(LAST_TIME), GetFbVar(LAST_TIME_AVERAGE));}
```

Глобальная переменная:

```
{printf("PROGRAM GLOBAL_VAR = %d\n", *GetFbVar(GLOBAL_VAR));}
```

Переменная с адресом (`mbus_var1 AT %QD0.2.5 : DINT := 5`):

```
{printf("MBUS_VAR1 = %d\n", *GetFbVar(MBUS_VAR1));}
```

`SetFbVar(VAR, VAL)` — функция устанавливает значение переменной `var` равным `val`. Имя переменной нужно писать большими буквами.

Пример вызова:

```
{SetFbVar(TMP, 1);}
```

Для глобальных переменных и переменных, с которыми связан адрес, нужно использовать макрос **`__SET_LOCATED`**.

`__SET_LOCATED(data__->,VAR,,VAL)` — макрос устанавливает значение `val` глобальной переменной или переменной, с которой связан адрес, `var`.

Примеры вызова:

Глобальная переменная:

```
{__SET_LOCATED(data__->,GLOBAL_VAR,,124);}
```

Переменная с адресом (`mbus_var1 AT %QD0.2.5 : DINT := 5`):

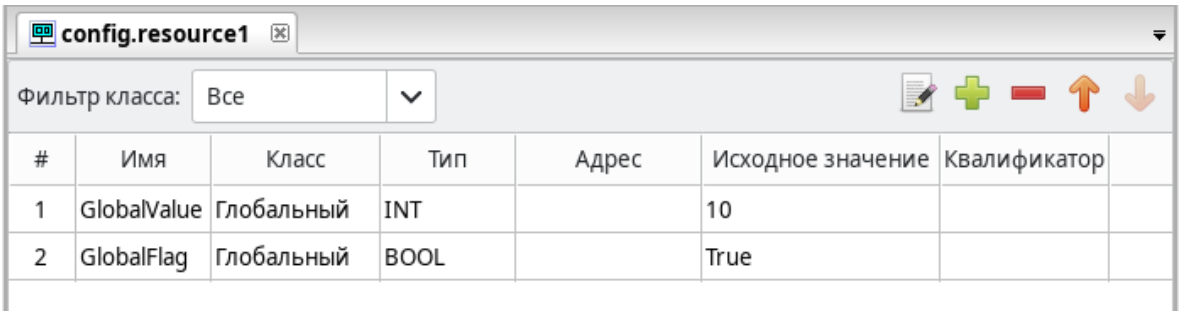
```
{__SET_LOCATED(data__->,MBUS_VAR1,,3);}
```

7.7. Ресурс

Согласно стандарту МЭК 61131-3-2016, каждый проект должен иметь как минимум один ресурс, с определённым в нём как минимум одним «экземпляром». Экземпляр представляет собой элемент, связанный с программным модулем типа «Программа» и одной определённой задачей. По умолчанию, среда разработки и отладки программ создаёт для нового проекта один ресурс.

7.7.1. Глобальные переменные ресурса

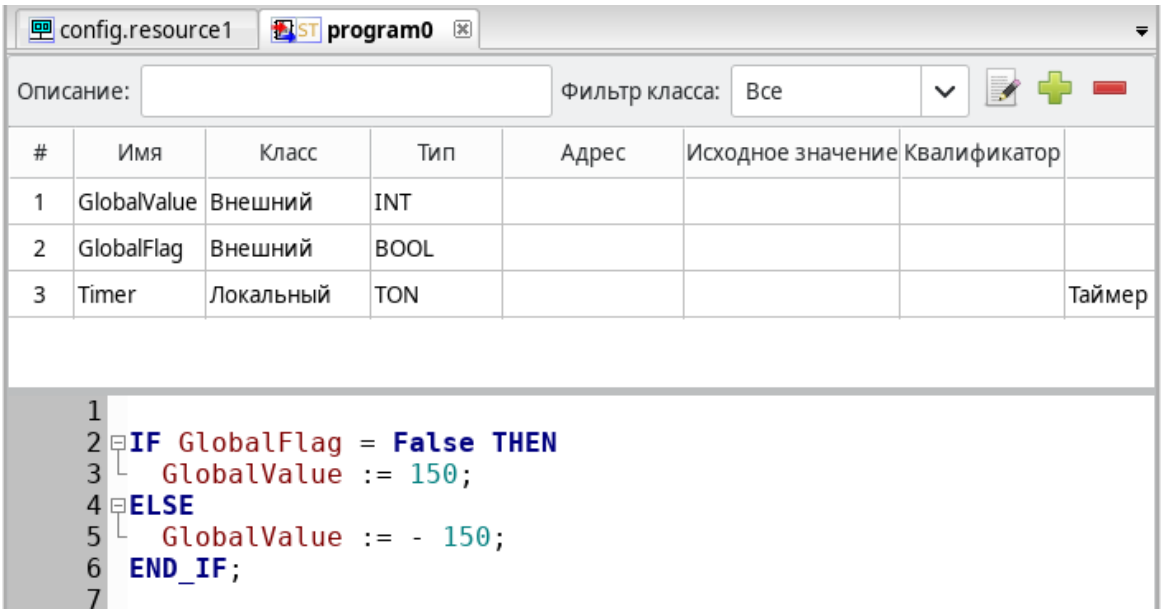
Глобальные переменные ресурса объявляются аналогично глобальным переменным проекта (см. п. 7.4.1) на панели переменных и констант выбранного ресурса с использованием кнопки «Добавить переменную» (см. табл. 3), рис. 198.



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	
1	GlobalValue	Глобальный	INT		10		
2	GlobalFlag	Глобальный	BOOL		True		

Рисунок 198 – Пример объявления глобальных переменных ресурса

Использование данных глобальных переменных на уровне ресурса аналогично использованию глобальных переменных проекта в программных модулях. На рис. 199, показано, как в программном модуле «`program0`» добавлены две переменных класса «`Внешний`» с такими же именами, как и глобальными переменные, объявленные для ресурса.



#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	
1	GlobalValue	Внешний	INT				
2	GlobalFlag	Внешний	BOOL				
3	Timer	Локальный	TON				Таймер

```
1
2 IF GlobalFlag = False THEN
3   GlobalValue := 150;
4 ELSE
5   GlobalValue := - 150;
6 END_IF;
7
```

Рисунок 199 – Объявление и использование глобальных переменных ресурса

В редакторе ST кода написан алгоритм работы данного программного модуля с использованием данных глобальных переменных.

7.7.2. Задачи и экземпляры ресурса

Для создания экземпляра необходимо наличие в проекте как минимум одного программного модуля типа «Программа» и как минимум одной задачи, определённой в панели редактирования ресурса.

После добавления задачи с помощью кнопки «Добавить» (аналогично кнопке «Добавить» на панели переменных и констант), необходимо задать её уникальное имя в поле «Имя», и выбрать тип выполнения задачи в поле «Запуск» (выделено красным на рис. 200):

– «Циклический» – выполнение программного модуля типа «Программа» через заданный интервал времени, указанный в поле «Интервал»;
– «Прерывание» – выполнение программного модуля типа «Программа» один раз при наступлении значения TRUE глобальной переменной типа BOOL, определённой на уровне проекта, либо на уровне ресурса и указанной в поле «Источник».

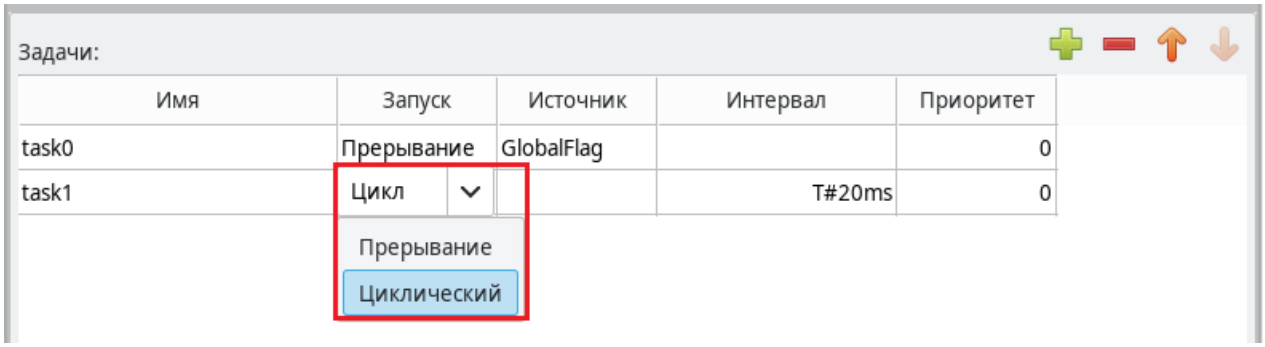


Рисунок 200 – Выбор типа выполнения задачи

В случае выбора типа выполнения «Циклический», в поле «Интервал» необходимо указать интервал, с которым будет выполняться данная задача. Двойной щелчок левой кнопкой мыши по полю «Интервал» приводит к появлению кнопки «...», выделено красным на рис. 201.

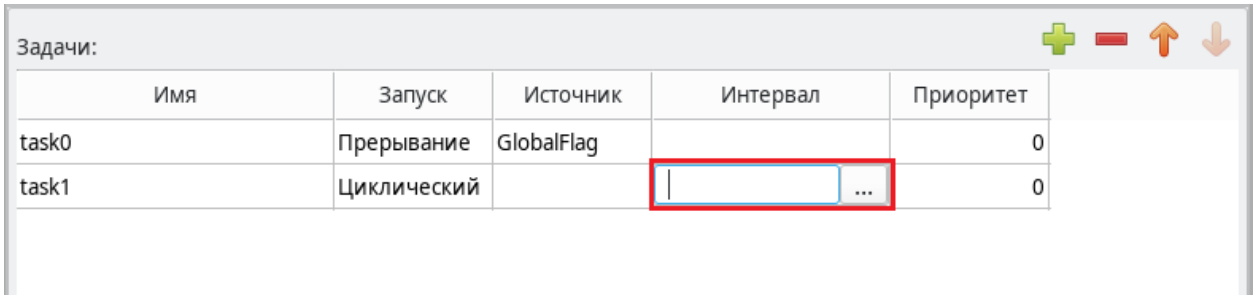


Рисунок 201 – Добавление задачи с цикличным режимом выполнения

Нажатие данной кнопки вызывает диалог «Редактировать длительность», в котором можно указать время, используя микросекунды, миллисекунды, секунды, минуты, часы и дни, рис. 202.

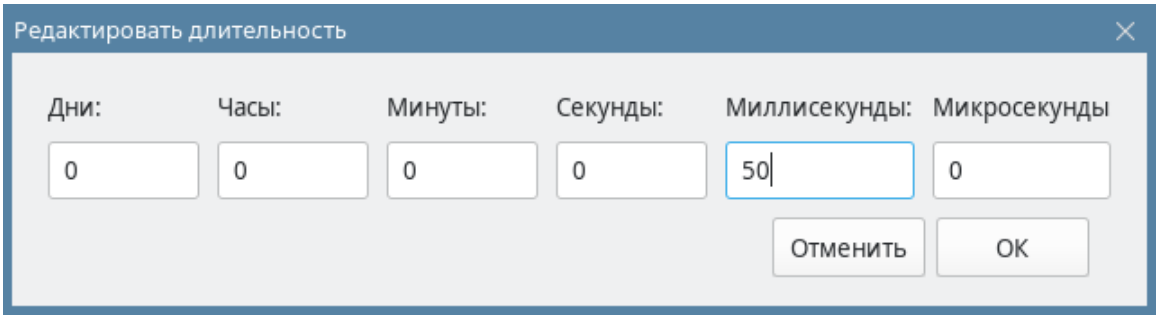


Рисунок 202 – Диалог редактирования продолжительности задачи

Завершение ввода кнопкой «OK» приводит к закрытию диалога и добавлению заданного интервала времени в поле «Интервал», рис. 203.

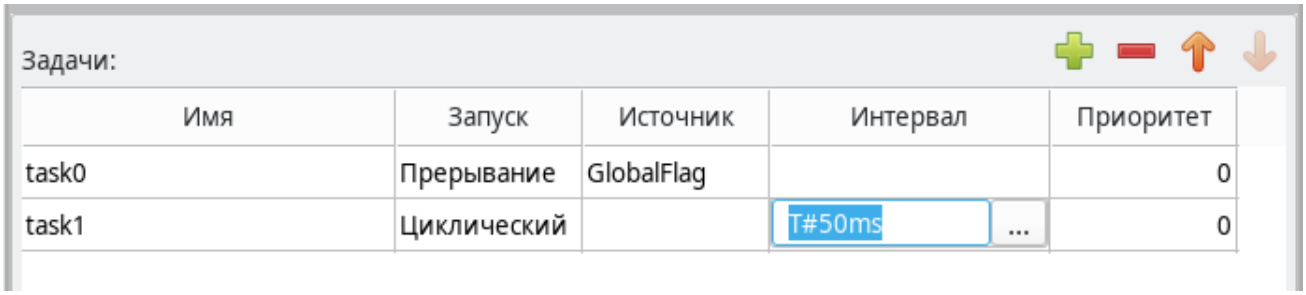


Рисунок 203 – Добавленный интервал выполнения

Для типа выполнения «Прерывание» в поле «Источник» необходимо указать переменную типа BOOL, определённую глобально либо на уровне проекта (см. п. 7.4.1), либо на уровне ресурса (см. п. 7.7.1). На рис. 204 красным выделен выбор переменной «globalFlag», определённой в данном ресурсе.

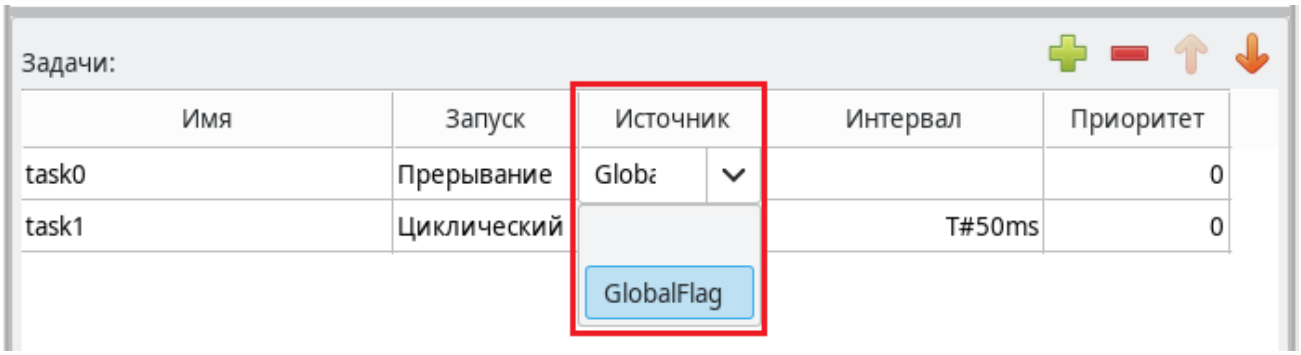


Рисунок 204 – Выбор переменной типа BOOL для определения условия выполнения задачи

Задача будет выполнена один раз, как только значение переменной, определённой в этом поле, будет TRUE.

Поле «Приоритет» позволяет указать приоритет выполнения задачи. По умолчанию все задачи имеют приоритет 0. В текущей реализации программы СРиО работа с приоритетами выполнения задачи не предусмотрена.

Следует отметить, что в ресурсе должна быть определена как минимум одна задача с типом выполнения «Циклический», в противном случае будет ошибка в компиляции в отладочной консоли (см. п. 6.14).

После того как задачи определены, их можно использовать в экземплярах. Создание экземпляра происходит аналогичным образом с помощью кнопки «Добавить». Необходимо выбрать уникальное имя экземпляра и далее указать программный модуль типа «Программа» в поле «Тип» и одну из задач в поле «Задача». Например, в проекте определено два программных модуля типа «Программа»: «program0» и «program1», рис. 205.

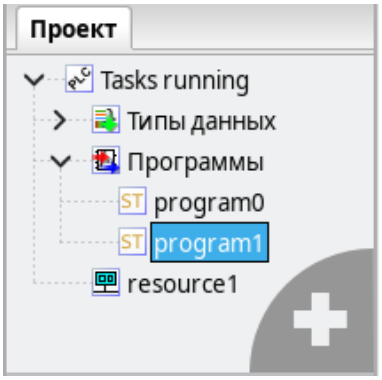


Рисунок 205 – Проект, содержащий два программных модуля типа «Программа»

Соответственно, при создании экземпляра в поле «Тип» оба этих программных модуля будут доступны, выделено красным на рис. 206.

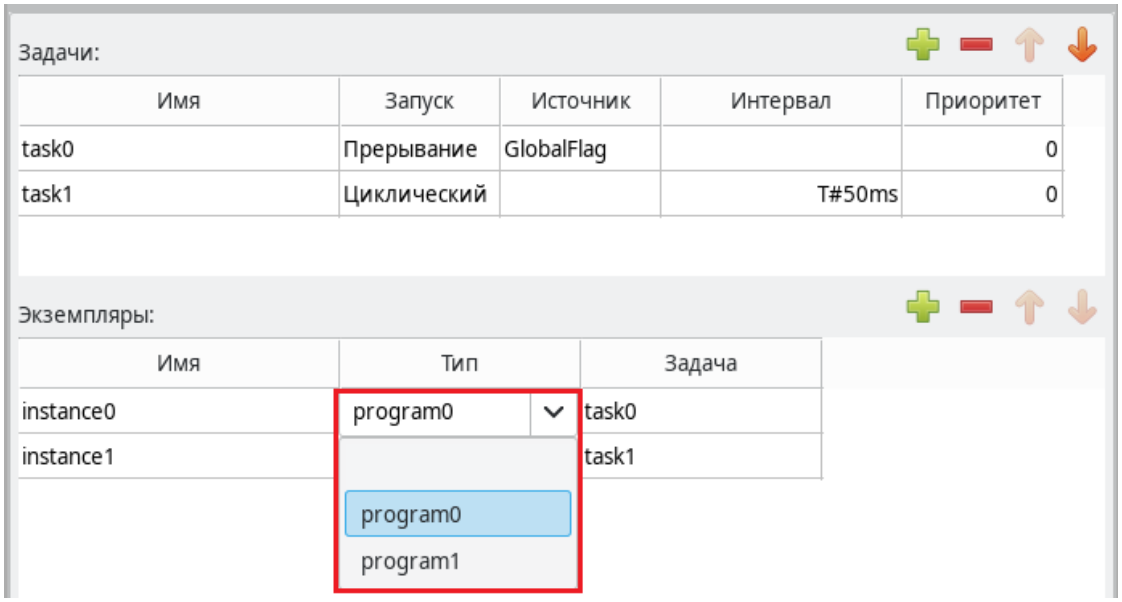


Рисунок 206 – Выбор программного модуля типа «Программа» для экземпляра

Аналогично выбирается задача из списка, в котором будут отображены определённые ранее задачи, выделено красным на рис. 207.

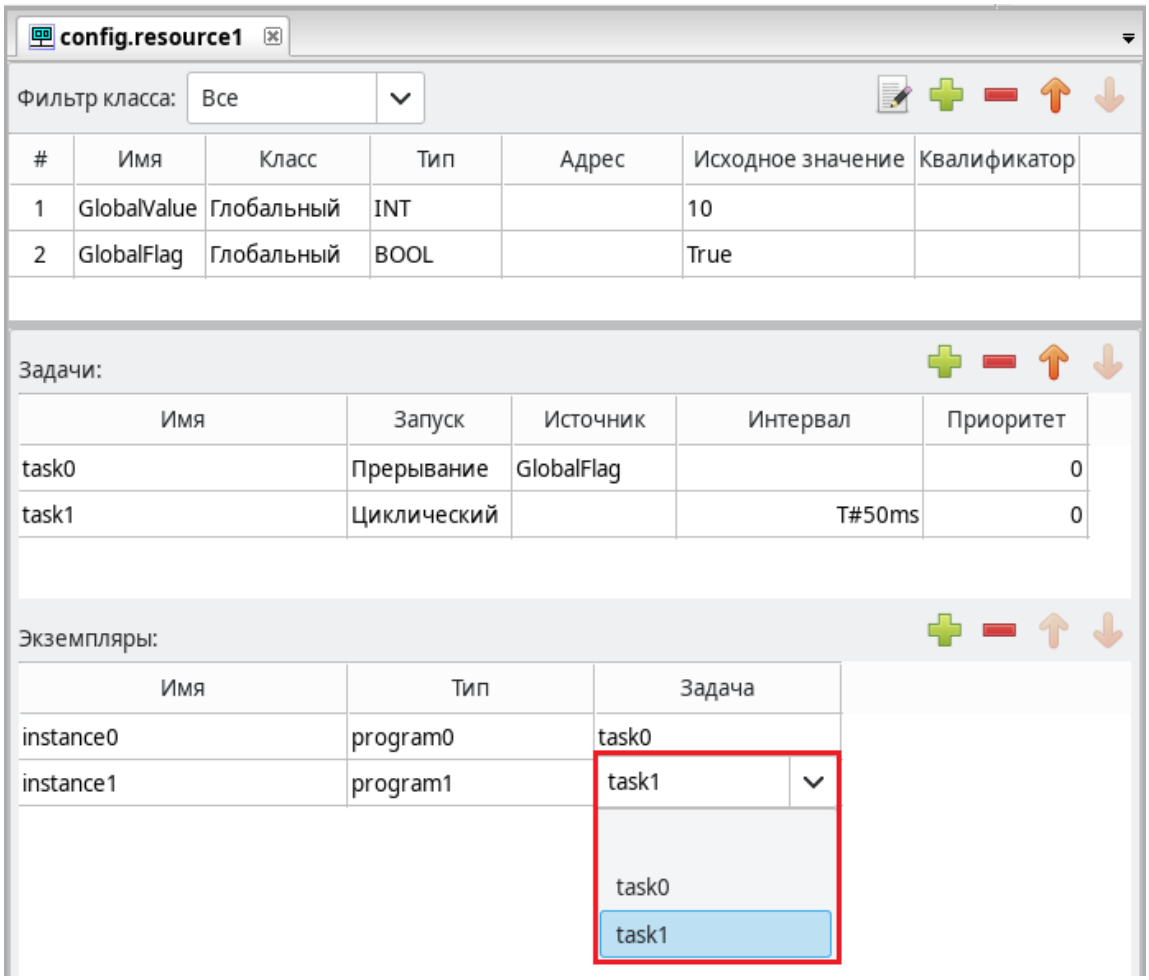


Рисунок 207 – Выбор задачи для экземпляра

В каждом проекте в ресурсе должен быть определён как минимум один экземпляр, в противном случае будет выдана ошибка компиляции в отладочной консоли (см. п. 6.14).

7.8. Создание новых типов данных проекта

Добавление типа данных происходит выбором пункта «Типа данных» в меню структура проекта (выделено красным на рис. 208) в уже созданный проект, содержащий программный модуль типа «Программа» – «program0».

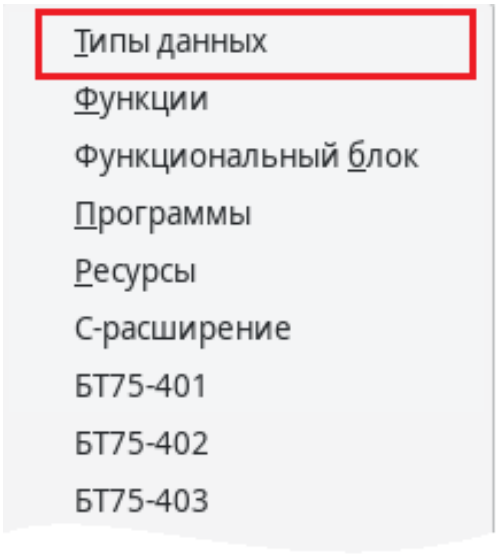


Рисунок 208 – Выбор пункта меню добавления пользовательского типа данных в структуре проекта

Пример создания нового типа данных - массива типа INT размерностью 11 элементов. В текущей реализации СРиО, имя создаваемого типа данных - «MyArrayType» можно ввести после двойного щелчка левой кнопкой мыши, рис. 209.

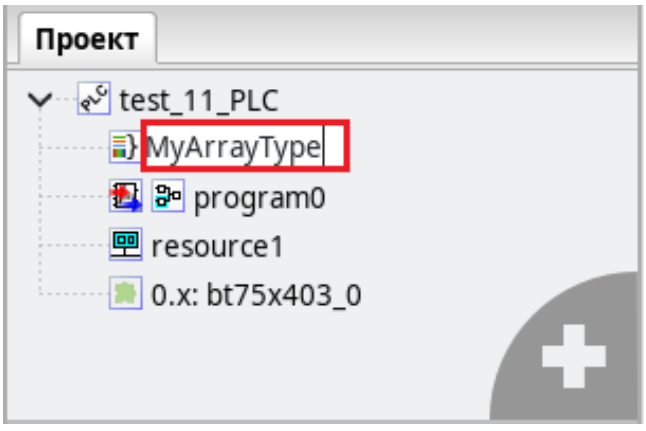


Рисунок 209 – Диалог ввода имени создаваемого пользовательского типа данных

В структуре проекта появляется панель редактирования добавленного типа данных с именем «MyArrayType» (см. п. 0). В поле «Механизм создания нового типа» необходимо выбрать «Массив» и указать тип INT, выделено красным на рис. 210.

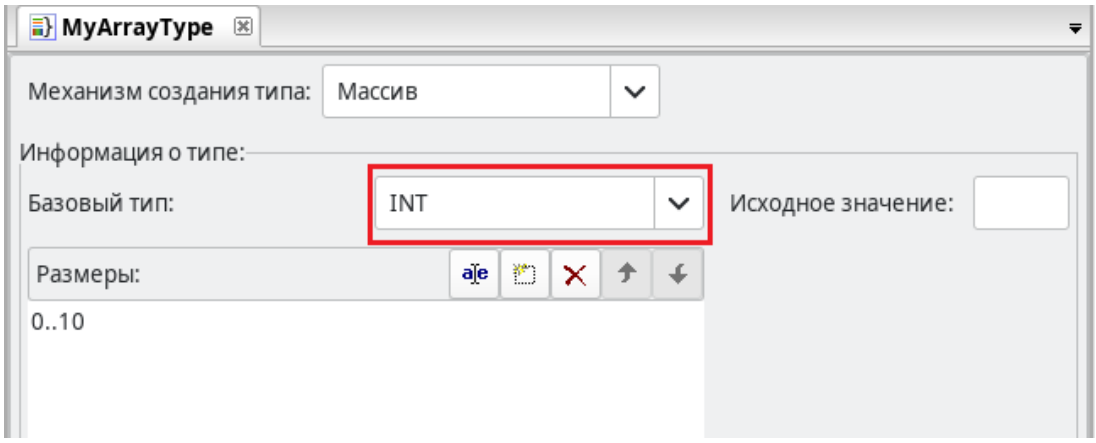


Рисунок 210 – Выбор базового типа для массива

С помощью кнопки «Редактировать элемент» указывается размерность массива, выделено красным на рис. 211.

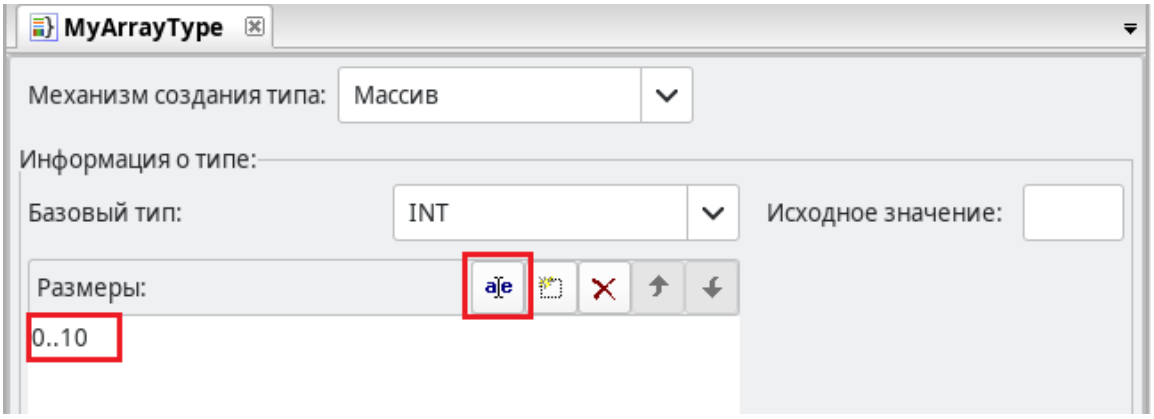


Рисунок 211 – Задание размерности для массива

После выполнения вышеперечисленных операций тип «MyArrayType» может быть использован для определения переменных в программных модулях, так же, как и базовые типы данных, рис. 212.

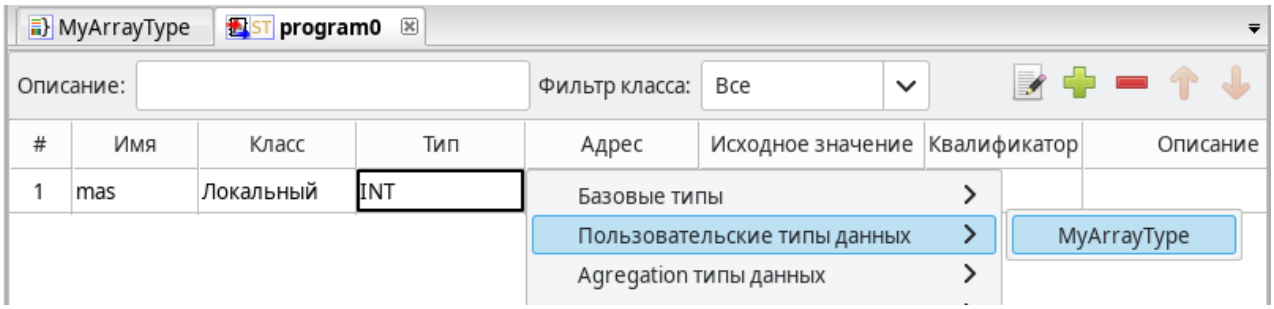


Рисунок 212 – Выбор добавленного типа данных в панели переменных и констант программного модуля

Добавим две переменные «temp_value» и «out_value» типа INT в панель переменных и констант. Ниже, на рис. 213, пример кода на языке ST, в котором используется переменная «mas», тип которой массив, добавленный выше.

Сначала локальной переменной «temp_value» присваивается значение 10:

```
temp_value := 10;
```

Далее первому элементу массива «mas» (нумерация элементов в массиве начинается с 0) присваивается значение переменной «temp_value»:

```
mas[0] := temp_value;
```

Второму элементу массива «mas» присваивается значение 20:

```
mas[1] := 20;
```

И в конце второй локальной переменной присваивается первый элемент массива «mas»:

```
out_value := mas[0];
```

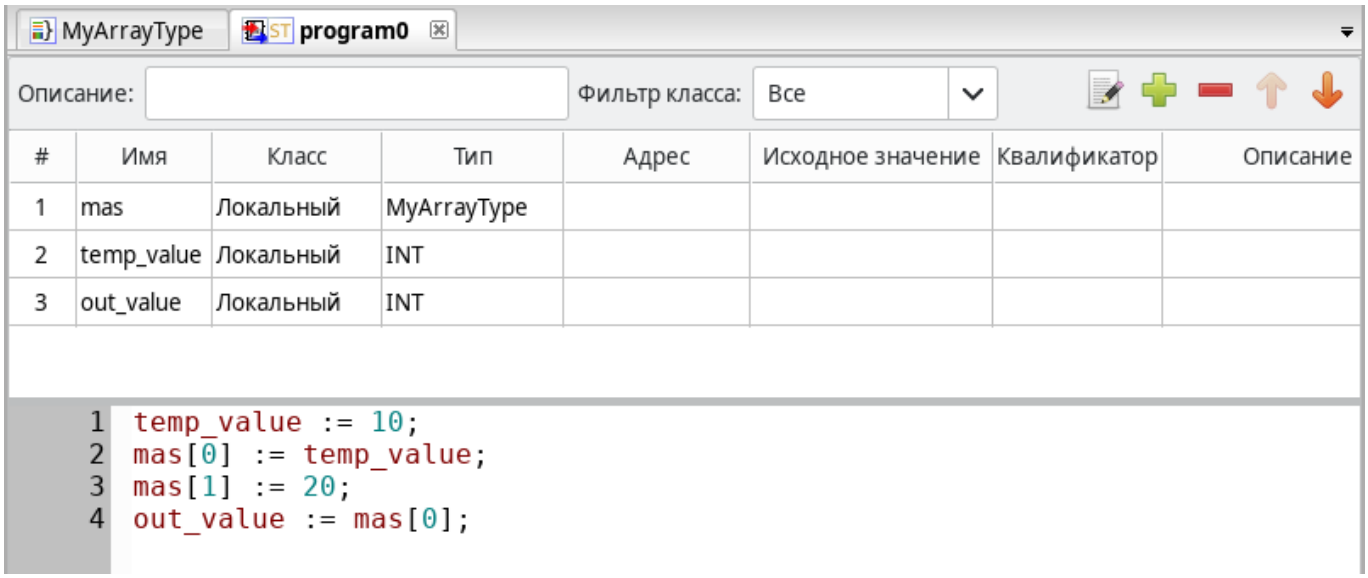


Рисунок 213 – Пример использования массива в коде на языке ST

Далее будет рассмотрена процедура сборки и отладки проекта.

8. СБОРКА, ЗАПУСК И ОТЛАДКА ПРОЕКТА

8.1. Сборка проекта

8.1.1. Очистка директории сборки проекта

Очистка директории сборки проекта осуществляется с помощью кнопки «Очистка директории сборки проекта». На рис. 215 данная кнопка выделена красным:

Рисунок 214 – Кнопка очистки директории сборки проекта

8.1.2. Сборка проекта

Сборка проекта осуществляется с помощью кнопки «Сборка проекта в директории сборки». На рис. 215 данная кнопка выделена красным:

Рисунок 215 – Кнопка сборки проекта

Во время сборки проекта необходимо следить за отладочной консолью, в которой в случае наличия ошибок, будут выведены соответствующие сообщения.

В случае успешной сборки, в отладочную консоль будет выведено сообщение:

«Сборка прошла успешно»

После успешной сборки проекта, его необходимо передать на целевое устройство для целевой платформы ПЛК Багет-ПЛК1 или же запустить непосредственно на ИЭВМ для целевой платформы “simulation”. Затем запустить проект на выполнение.

8.1.3. Просмотр сгенерированного кода

Просмотр сгенерированного кода с помощью кнопки «Показать код, сгенерированный PLCGenerator». На рис. 2157 данная кнопка выделена красным:

Рисунок 216 – Кнопка просмотра сгенерированного кода

8.2. Запуск проекта

Если в проект (программу) включены средства отладки, то для его запуска на Багет-ПЛК1 следует выполнить следующие действия:

– Настройка сетей целевой ПЛК;
– Настройка связи с ПЛК;
– Авторизация пользователя для работы с ПЛК (с вводом логина и пароля);
– Загрузка и запуск образа в ОЗУ ПЛК или Загрузка образа в ППЗУ ПЛК;
– Подключение к целевому ПЛК;
– Запуск ПЛК.

Для завершения работы с проектом (программой) запущенной на Багет-ПЛК1 следует выполнить следующие действия:

– Остановка запущенного ПЛК (если включены средства отладки);
– Отключение от ПЛК;
– Окончание работы авторизованного пользователя.

8.2.1. Настройка сетей целевой ПЛК

Настройка сетей целевой ПЛК осуществляется с помощью кнопки «Настройка сети проекта». На рис. 217 данная кнопка выделена красным:

Рисунок 217 – Кнопка настройки сети проекта

При нажатии на кнопку «Настройка сети проекта» появляется диалоговое окно для задания параметров сети (см. рис. 218).

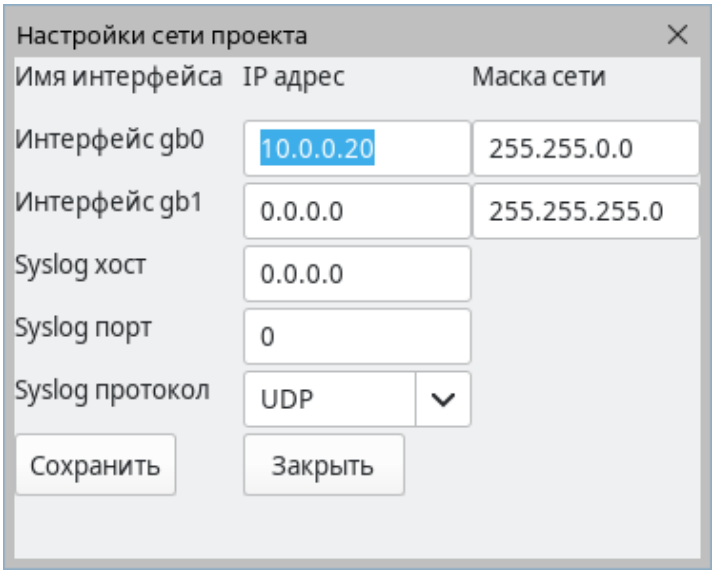


Рисунок 218 – Диалоговое окно для задания параметров сети

Примечание: На рис.218 приведены настройки для конфигурации: 10.0.0.30 – IP адрес ИЭВМ, 10.0.0.20 – IP адрес ПЛК. В режиме СС ИЭВМ (simulation), в поле «Интерфейс gb0» рекомендуется ввести значение: 10.0.0.20.

8.2.2. Настройка связи с ПЛК

Выбор IP-адреса, порта, последовательного порта для загрузки образа в ПЛК осуществляется с помощью кнопки «Настроить связь с ПЛК». На рис. 219 данная кнопка выделена красным:

Рисунок 219 – Кнопка «Настроить связь с ПЛК»

При нажатии на кнопку «Настроить связь с ПЛК» появляется диалоговое окно (см. рис. 220):

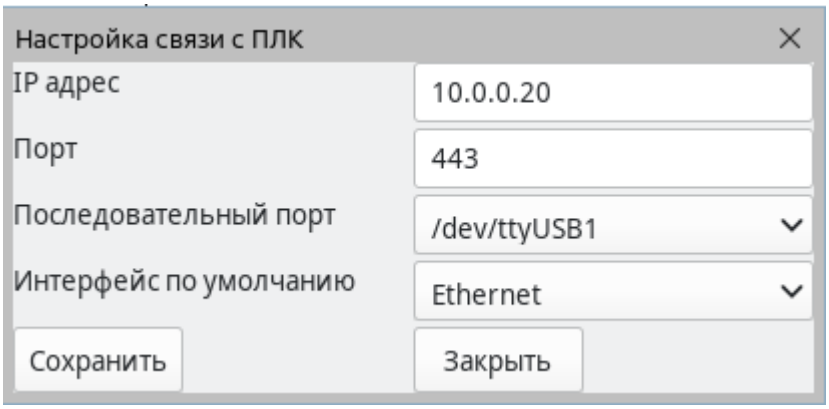


Рисунок 220 – Диалоговое окно «Настройка связи с ПЛК»

Примечания:

1. На рис.224 приведены настройки для конфигурации: 10.0.0.30 – IP адрес ИЭВМ, 10.0.0.20 – IP адрес ПЛК;
2. В режиме СС ИЭВМ (simulation), диалог «Настройка связи с ПЛК» не функциональна. Рекомендуется задать IP адрес: 10.0.0.20, Порт: 443.
3. Требуется настроить IP-адрес и порт в форме «Проект -> Конфигурация», поле «URI системы исполнения:»
   - JSONRPC://10.0.0.20:1234 – для целевых платформ «PLC1_A» и «PLC1_B».
   - JSONRPC://127.0.0.1:1234 – для режима СС ИЭВМ (simulation), см. п.7.4.3 «Настройки сборки проекта и соединения с целевым устройством».

8.2.3. Авторизация пользователя для работы с ПЛК

Авторизация пользователя для работы с ПЛК осуществляется с помощью кнопки «Логин». На рис. 221 данная кнопка выделена красным:

Рисунок 221 – Кнопка «Логин»

При нажатии на кнопку «Логин» появляется диалоговое окно для выбора последовательного порта (см. рис. 222).

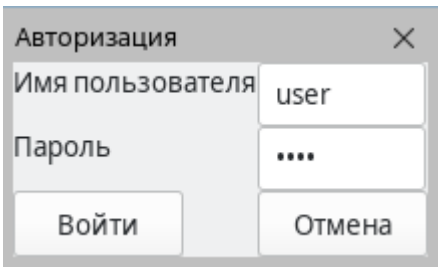


Рисунок 222 – Диалоговое окно для авторизации пользователя для работы с ПЛК

Примечание: в режиме СС ИЭВМ (simulation), «Авторизация для работы с ПЛК» не функциональна (не имеет смысла).

8.2.4. Загрузка образа в ППЗУ ПЛК

Загрузка образа в ППЗУ ПЛК осуществляется с помощью кнопки «Загрузить образ на ПЛК». На рис. 2237 данная кнопка выделена красным:

Рисунок 2237 – Кнопка загрузки образа в ППЗУ ПЛК

На лицевой панели процессорного модуля ПЛК находится тумблер «RUN/PRG». В режиме «PRG» осуществляется запись образа ОСРВ в ПЗУ модуля. В режиме «RUN» образ ОСРВ запускается. Либо следует перезагрузиться в технологический образ (см. п.8.2.12);

Во время загрузки образа в ППЗУ ПЛК необходимо следить за отладочной консолью, в которой в случае возникновения ошибок, будут выведены соответствующие сообщения. Сообщение: «`Образ успешно записан на ПЛК`» соответствует успешной загрузке.

Если в проект не включены средства отладки (см. п.6.5.1), и тумблер ««RUN/PRG» находится в положении «RUN», то запуск образа ОСРВ "Багет" осуществляется автоматически по включению питания или сбросу. Рекомендуется подключить дополнительный терминал к технологическому интерфейсу USB 2.0 процессорного модуля ПЛК (`minicom -D /dev/ttyUSB1`). Контроль запуска СПО возможен за счет вывода программой сообщений в терминал (см. п 7.6.17).

8.2.5. Загрузка и запуск образа в ОЗУ ПЛК

Загрузка образа ОСРВ "Багет" в ОЗУ ПЛК осуществляется с помощью кнопки «Загрузить и запустить образ на ПЛК». На рис. 228 – данная кнопка выделена красным:

Рисунок 224 – Кнопка загрузки и запуска образа на ПЛК

Если в проект не включены средства отладки (см. п.6.5.1), то запуск образа ОСРВ "Багет" осуществляется автоматически. Контроль запуска СПО возможен по выводу программой сообщений в терминал.

8.2.6. Подключение к целевому ПЛК

Подключение к целевому ПЛК осуществляется с помощью кнопки «Подключится к целевому ПЛК». На рис.230 данная кнопка выделена красным:

Рисунок 225 – Кнопка подключения к целевому ПЛК

После подключения вид панели инструментов работы с проектом изменится, станет доступной кнопка «Запустить ПЛК».

8.2.7. Запуск проекта на ПЛК

Если в проект включены средства отладки (см. п.6.5.1), запуск ПЛК осуществляется с помощью кнопки «Запустить ПЛК». На рис. 230 данная кнопка выделена красным:

Рисунок 226 – Кнопка запуска ПЛК

После подключения вид панели инструментов работы с проектом изменится, станет доступной кнопка «Остановить ПЛК».

Для запуска проекта на ПЛК требуется:

– Перевести тумблер «RUN/PRG» в положение «PRG» либо перезагрузиться в технологический образ (см. п.8.2.12);
– Загрузить образ в ОЗУ ПЛК (см. п.8.2.5);
– Подключиться к целевому ПЛК (см. п.8.2.6);
– Нажать кнопку «Запустить ПЛК».

Запуск ПЛК и инициализации сервера, отвечающего за соединение, занимает продолжительное время. Рекомендуется подключить дополнительный терминал к технологическому интерфейсу USB 2.0 процессорного модуля ПЛК (`minicom -D /dev/ttyUSB1`). Вывод в этот терминал сообщений: «`CRC ok`» и «`starting threads`» соответствует готовности ПЛК к загрузке образа ОСРВ в ОЗУ.

Во время загрузки образа в ОЗУ ПЛК нужно следить за отладочной консолью, в которую будут выведены сообщения в случае наличия ошибок.

Вывод в отладочную консоль сообщения:

```
[PLC] Waiting for connections…
```

Соответствует окончанию загрузки образа ОСРВ "Багет" в ОЗУ ПЛК.

8.2.8. Остановка запущенного ПЛК

Остановка запущенного ПЛК осуществляется с помощью кнопки «Остановить запущенный ПЛК». На рис. 22330 данная кнопка выделена красным:

Рисунок 22730 – Кнопка остановки запущенного ПЛК

После подключения вид панели инструментов работы с проектом изменится, станет доступной кнопка «Запустить ПЛК».

8.2.9. Отключение от ПЛК

Отключение от ПЛК осуществляется с помощью кнопки «Отключиться от ПЛК». На рис. 22330 данная кнопка выделена красным:

Рисунок 22831 – Кнопка отключения от ПЛК

8.2.10. Окончание работы авторизованного пользователя с ПЛК

Окончание работы авторизованного пользователя с ПЛК с текущим логином осуществляется с помощью кнопки «Логаут». На рис. 229 данная кнопка выделена красным:

Рисунок 229 – Кнопка завершения работы авторизованного пользователя с ПЛК

8.2.11. Перезагрузка ПЛК

Перезагрузка ПЛК осуществляется с помощью кнопки «Перезагрузка». На рис. 23329 данная кнопка выделена красным:

Рисунок 230 – Кнопка «Перезагрузка»

Загружается образ программы, который записан в ППЗУ ПЛК.

8.2.12. Перезагрузки ПЛК в технологический образ

При нажатии на кнопку «Перезагрузка в технологический образ», если есть подключение к ПЛК, происходит отправка команды перезагрузки в технологический образ. После перезагрузки в ПЛК будет загружен технологический образ. При обычной перезагрузке в ПЛК, (тумблер в положении «RUN»), загружается образ ОСРВ «Багет» с программой (проектом) пользователя, ранее записанным в ППЗУ ПЛК (см. п.8.2.4).

На рис. 23330 данная кнопка выделена красным:

Рисунок 231 – Кнопка «Перезагрузка в технологический образ»

Если есть подключение, но не произведён вход в систему под существующем пользователем, открывается окно авторизации.

В текущей реализации СРИО следующие функции доступны только в технологическом образе:

– Обновление технологического образа;

– Настройка пользователей;
– Получение журнала безопасности;
– Получение списка логов;
– Скачивание проекта;
– Загрузка образа на ПЛК;
– Загрузка и запуск образа на ПЛК.

Для выполнения этих команд нужно перезагрузиться в технологический образ.

8.2.13. Обновление технологического образа на ПЛК

При нажатии на кнопку «Обновить технологический образ», если мы подключены к ПЛК, происходит отправка команды обновления технологического образа. На рис. 2331 данная кнопка выделена красным:

Рисунок 232 – Кнопка «Обновить технологический образ»

Если есть подключение, но не произведён вход в систему под существующем пользователем, открывается окно авторизации. Нужно войти под учетной записью администратора.

Рекомендуется подключить дополнительный терминал к технологическому интерфейсу USB 2.0 процессорного модуля ПЛК (`minicom -D /dev/ttyUSB1`). Процесс загрузки технологического образа в ППЗУ ПЛК рекомендуется контролировать через этот терминал.

Также рекомендуется (но не обязательно) перевести тумблер «RUN/PRG» на лицевой панели процессорного модуля ПЛК в положение «PRG».

8.2.14. Настройка учетных записей пользователей для работы с ПЛК

После установки ОСРВ на ПЛК в системе отсутствуют установленные пользователем учетные записи пользователей. Поэтому при первом запуске ОСРВ для получения доступа к системе должна использоваться предустановленная учетная запись, именуемая как "Временный администратор". Имя и пароль Временного администратора встроены в исполняемый образ ОСРВ и назначаются при его конфигурировании и сборке. Временный администратор обладает всеми возможностями администратора.

Наличие Временного администратора является мерой, направленной на обеспечение дополнительной безопасности на этапе начальной настройки ОСРВ. Для завершения настройки Временному администратору необходимо создать первую учетную запись пользователя системы (должен являться администратором), после чего учетная запись Временного администратора будет автоматически

деактивирована. Учетная запись Временного администратора не должна использоваться при штатной эксплуатации ОСРВ.

Настройка учетных записей пользователей для работы с ПЛК осуществляется с помощью кнопки «Настройка пользователей». На рис. 233 данная кнопка выделена красным:

Рисунок 233 – Кнопка настройки учетных записей пользователей для работы с ПЛК

При нажатии на кнопку «Настройка пользователей» появляется диалоговое окно для настройки списка пользователей (см. рис. 234).

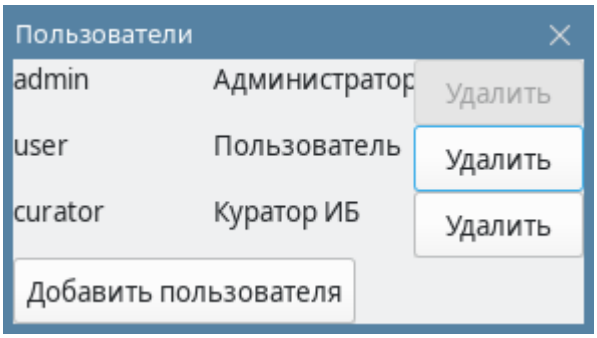


Рисунок 234 – Диалоговое окно для настройки списка пользователей ПЛК

Настройку учетных записей пользователей может выполнять только администратор ПЛК. «Куратор ИБ» не может просматривать или редактировать список пользователей.

При задании пароля должны соблюдаться следующие требования:

– пароль должен содержать не меньше 10 символов;
– пароль должен содержать знаки хотя бы трех из четырех категорий (цифры, буквы, символы разного регистра, специальные символы);
– пароль не должен содержать последовательности повторяющихся символов, либо символов, последовательно находящихся на клавиатуре.

В случае нарушения указанных требований будет выдано соответствующее предупреждение (рис. 118).

Администратор может удалять других пользователей, в том числе и других администраторов, но не может удалять сам себя.

Для режима СС ИЭВМ (simulation), «Настройка учетных записей пользователей для работы с ПЛК» не функциональна.

8.2.15. Получение журнала безопасности

Получение журнала безопасности из ПЛК осуществляется с помощью кнопки «Получить журнал безопасности». На рис. 235 данная кнопка выделена красным:

Рисунок 235 – Кнопка получения журнала безопасности

При нажатии на кнопку «Получить журнал безопасности» появляется диалоговое окно для сохранения журнала безопасности в определённую папку. Получать журнал безопасности может только администратор.

8.2.16. Получение файлов истории

Получение файлов истории из ПЛК осуществляется с помощью кнопки «Получить список логов». На рис. 236 данная кнопка выделена красным:

Рисунок 236 – Кнопка получения файлов истории

При нажатии на кнопку «Получить список логов» появляется диалоговое окно для выбора файлов истории и сохранения их в определённую папку.

8.2.17. Сохранение загруженного в ПЛК проекта

Сохранение загруженного в ПЛК проекта осуществляется с помощью кнопки «Скачать проект». На рис. 237 данная кнопка выделена красным:

Рисунок 237 – Кнопка для скачивания проекта, загруженного в ПЛК

При нажатии на кнопку «Скачать проект» появляется диалоговое окно для сохранения проекта, загруженного в ПЛК, в определённую папку

Примечание: Кнопка «Скачать проект» работает только для модуля PLC1_A, в котором установлена NAND-память. В модуле PLC1_B NAND-память отсутствует, и кнопка «Скачать проект» работать не будет. Перед сохранением проекта нужно перезагрузить ПЛК в технологический образ. Для режима СС ИЭВМ (simulation), «Скачивание проекта» не доступно.

8.2.18. Конфигурирование модулей ввода-вывода ПЛК

Конфигурирование модулей ввода-вывода ПЛК осуществляется с помощью кнопки «Запустить конфигуратор модулей ввода-вывода». На рис. 238 данная кнопка выделена красным:

Рисунок 238 – Кнопка для конфигурирования модулей ввода-вывода ПЛК

При нажатии на кнопку «Запустить конфигуратор модулей ввода-вывода» откроется диалоговое окно (рис. 239).

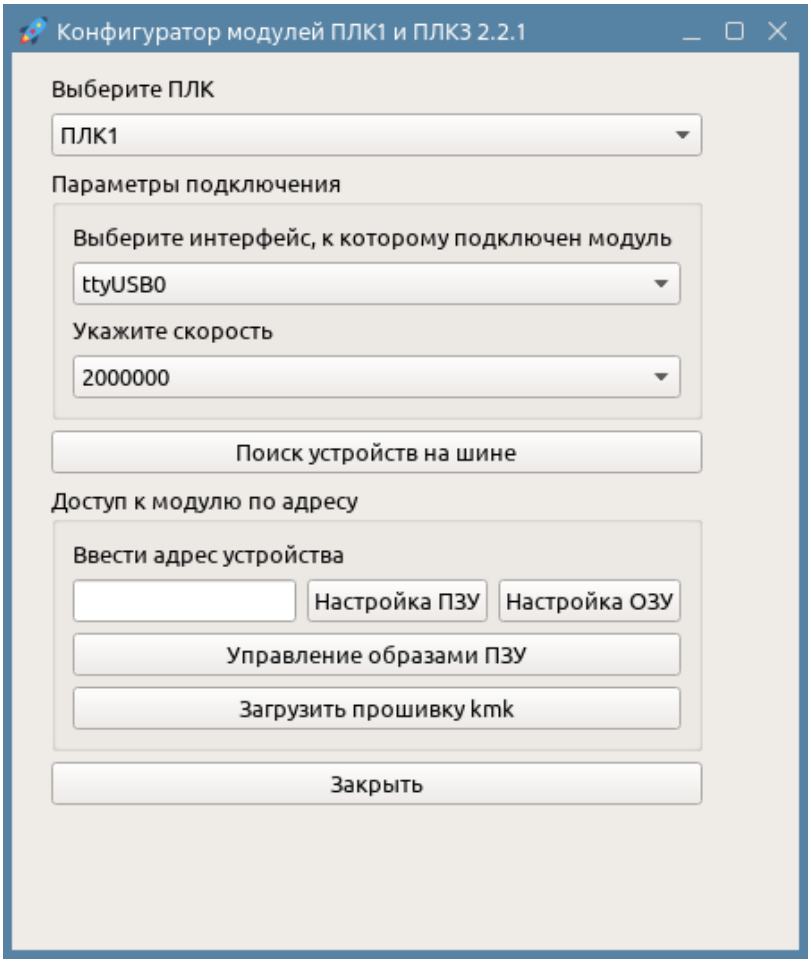


Рисунок 239 – Диалоговое окно для выбора конфигурируемого модуля

Описание «Конфигуратора» и сведения о настройке модулей ввода-вывода приведено в документе «Программа «Конфигуратор модулей ввода-вывода для программируемого логического контроллера Багет-ПЛК1» (Конфигуратор Багет-ПЛК1) Руководство оператора» ЮКСУ.91238-01 34 01.

Модули ввода-вывода подключаются к общей внутренней шине ПЛК через интерфейс RS-485 (Modbus RTU). Для работы «Конфигуратора» требуется адаптер USB-Serial UART. Адаптер подключается к шине RS-485 (соединители расположены на задней стенке модулей), и к USB-порту ИЭВМ. Во время работы «Конфигуратора» следует переключить процессорный модуль ПЛК в режим «PRG».

Диалоговое окно позволяет вводить следующие настройки:

– «Выберете ПЛК» – нужно выбрать: «ПЛК1»;

– «Выберете интерфейс к которому подключен модуль» –USB порт ИЭВМ, к которому подключен адаптер USB-Serial UART;

– «Укажите скорость» – нужно выбрать: 2000000;

– «Поиск устройств на шине» – поиск модулей ввода-вывода ПЛК, подключенных к общей внутренней шине (Modbus RTU). Адреса модулей, записанные в ПЗУ, должны быть корректными (два одинаковых адреса вызовут ошибку). Если модули найдены, будет выведен их список;

– «Ввести адрес устройства» – доступ к модулю ввода-вывода ПЛК по известному адресу;

– «Настройка ПЗУ» – настройки модуля ввода-вывода ПЛК, записываемые в ПЗУ: адрес на шине, таймаут обращений ведущего (после таймаута включится индикатор «ERR»), включенные каналы, и др. (см. «Руководство оператора» ЮКСУ.91238-01 34 01).

– «Настройка ОЗУ» – настройки модуля ввода-вывода ПЛК, связанные с ОЗУ: статус модуля, статусы каналов и др. (см. «Руководство оператора» ЮКСУ.91238-01 34 01).

– «Управление образами ПЗУ» – загрузка образа ПЗУ в модуль ввода-вывода ПЛК. Или сохранение образа ПЗУ на диск ИЭВМ. Загружаемый образ должен соответствовать типу модуля, иначе возможна потеря функциональности. После завершения загрузки/сохранения будет выведено сообщение. Например, «Запись образа ПЗУ прошла успешно» или «Образ ПЗУ успешно сохранен»;

– «Загрузить прошивку kmk» – каждый модуль ввода-вывода ПЛК1 содержит микроконтроллер «Комдив-МК» (kmk), обеспечивающий внутренний цикл работы: приём команд и передачу ответов для управляющего устройства, опрос и выставление команд для контроллеров ввода-вывода. Изменение этих параметров осуществляется путем прошивки микроконтроллера файлом « .bin».

– «Закрыть» – закрыть окно «Конфигуратор модулей ввода-вывода».

8.2.19. Получение статуса ПЛК

Получение статуса ПЛК осуществляется с помощью кнопки «Получить статус ПЛК». На рис.240 данная кнопка выделена красным:

Рисунок 240 – Кнопка для получения статуса ПЛК

При нажатии на кнопку «Получить статус ПЛК» в отладочную консоль выводится одно из сообщений:

– «`Не могу получить статус ПЛК`», если не удалось получить статус;
– Информация о статусе ПЛК.

Информация о статусе ПЛК выводится по следующей форме:

– Статус: `<status>;`
– Запущен технологический образ: `<tech>;`
– Модули:

Тип модуля: `<type>`, адрес: `<address>`, статус: `<status>`, слово ошибок: `<error>`, версия прошивки: `<version>`.

Где:

– `<status>` - числовое значение текущего статуса ПЛК или модуля ввода-вывода;
– `<tech>` - информация, запущен ли сейчас технологический образ (true или false);
– `<type>` - тип модуля;
– `<address>` - адрес модуля на шине.
– `<error>` - числовое значение слова ошибок модуля ввода-вывода;
– `<version>` - числовое значение версии прошивки модуля ввода-вывода.

Поле `<status>` может принимать значения:

– Ноль, если нет ошибок в ходе выполнения программы.

– Если ошибки найдены, устанавливаются биты:

бит 0 — несовпадение контрольной суммы проекта при старте;
бит 1 — ошибка связи с модулями ввода-вывода;
бит 2 — ошибка модулей ввода-вывода;
бит 3 — ошибка программы.

Информация о модулях ввода-вывода выводится при их наличии в проекте.

8.2.20. Запуск проекта для цели «simulation» (СС ИЭВМ)

Для запуска проекта в режиме «Сервис симуляции для ИЭВМ» ─ СС ИЭВМ (simulation) следует выполнить следующие действия:

- Создать новый проект или открыть существующий;
- Задать цель «`simulation`» в форме «Проект -> Конфигурация», поле «Целевая платформа», см. п.7.4.3 «Настройки сборки проекта и соединения с целевым устройством»;
- Настроить IP-адрес и порт в форме «Проект -> Конфигурация», поле «URI системы исполнения: `JSONRPC://127.0.0.1:1234`, см. п.7.4.3 «Настройки сборки проекта и соединения с целевым устройством».
- Нажать кнопку: «Настройка сети проекта», см. п.8.2.1 «Настройка сетей целевой ПЛК». В поле «Интерфейс gb0» рекомендуется ввести значение: 10.0.0.20;
- Нажать кнопку: «Настроить связь с ПЛК», см. п.8.2.2 «Настройка связи с ПЛК». В режиме СС ИЭВМ (simulation), диалог настройка связи с ПЛК не функционален (на ПЛК ничего не передается). Рекомендуется ввести значение: IP адрес: 10.0.0.20, Порт: 443.
- Нажать кнопку: «Сборка проекта», см. п.8.1.2 «Сборка проекта»
- Нажать кнопку: «Загрузить и запустить образ на ПЛК», см. п.8.2.5 «Загрузка и запуск образа в ОЗУ ПЛК».
- Нажать кнопку: «Подключится к целевому ПЛК», см. п.8.2.6 «Подключение к целевому ПЛК»;
- Нажать кнопку: «Запустить ПЛК», см. п.8.2.7 «Запуск проекта на ПЛК».

Для цели “simulation”: СС ИЭВМ вместе с ПО ПЛК доступен в директории build проекта и после запуска средствами ОС доступен для подключения из СРиО.

Для цели “simulation”: ПЛК не подключается, поэтому следующие функции не будут работать:

- Авторизация для работы с ПЛК» (Логин);
- Настройка пользователей;
- Скачать проект;
- Загрузить образ на ПЛК;
- Получить статус ПЛК;
- Настройка пользователей;
- Обновить технологический образ.

8.3. Отладка проекта

8.3.1. Отладка программ на текстовых языках ST и IL

После установки соединения с целевым устройством и запуском прикладной программы на выполнение, СРиО позволяет отслеживать и изменять значения переменных программных модулей, из которых состоит проект. Необходимо выбрать в выпадающем списке отлаживаемый экземпляр POU, затем обратиться к панели экземпляров и используя кнопки, выделенные красным цветом на рисунке 241, добавить переменных в панель отладки:

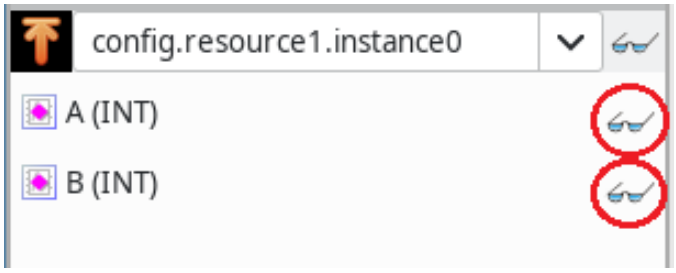


Рисунок 241 – Кнопки добавления переменных для отладки из панели экземпляров

На панели отладки отобразятся текущие значения добавленных переменных (рис. 242).

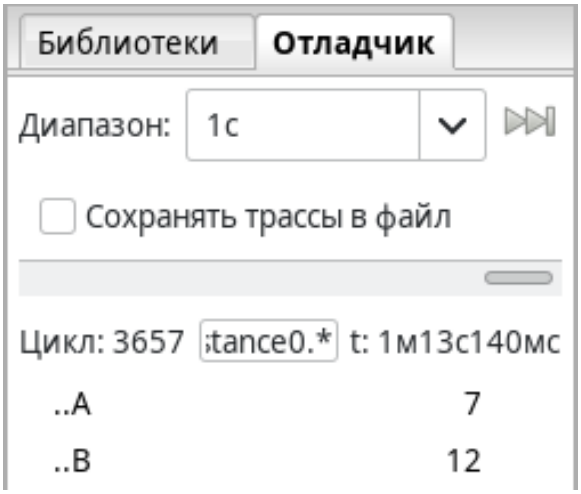


Рисунок 242 – Панель отладки значений переменных

Для того чтобы изменить значение переменной во время исполнения проекта, необходимо нажать левой клавишей мыши на значение интересующей переменной. Затем нажать на кнопку задания значения, выделенную красным на рис. 244.

Примечание: в текущей реализации СРиО рекомендуется сначала вывести график изменения значений переменной, нажав на кнопку, выделенную красным на рис. 243.

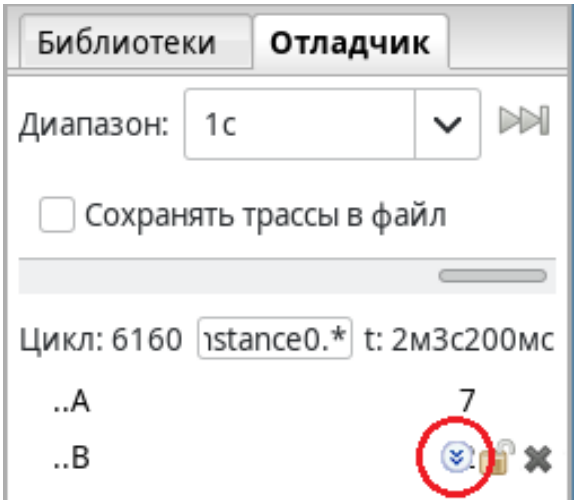


Рисунок 243 – Кнопка вывода графика изменения значений переменной

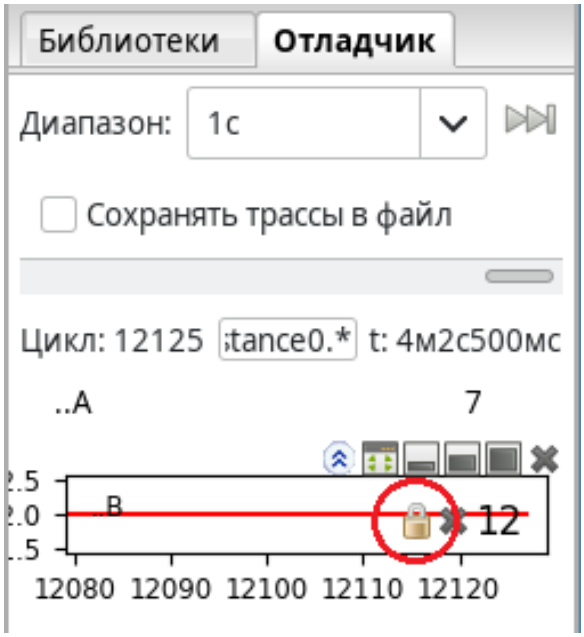


Рисунок 244 – Кнопка задания значения переменной

Появится диалог, показанный на рис. 245.

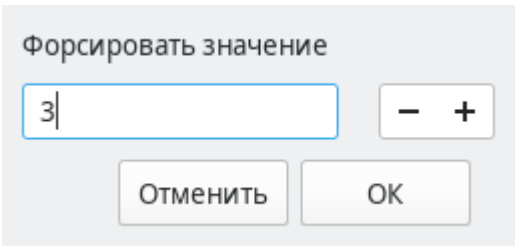


Рисунок 245 – Установка значения переменной во время отладки

Вводимое значение должно соответствовать типу переменной, иначе будет выведено сообщение об ошибке, рис. 246.

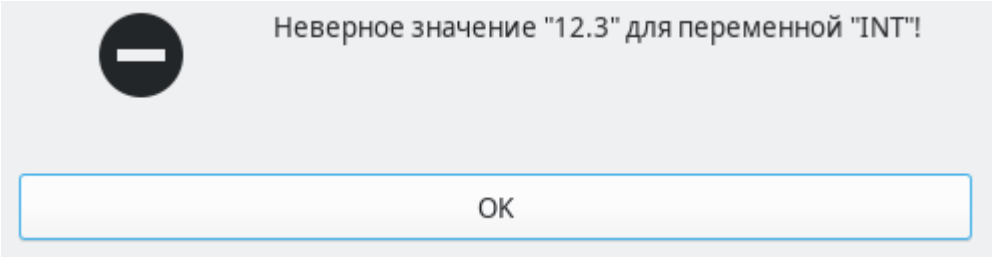


Рисунок 246 – Ошибка при вводе недопустимого значения изменяемой переменной в режиме отладки

Для возврата переменной в «нормальный» режим (т.е. вывести из режима «Форсировать значения»), необходимо нажать кнопку «Освободить значение», выделенную красным на рис. 247.

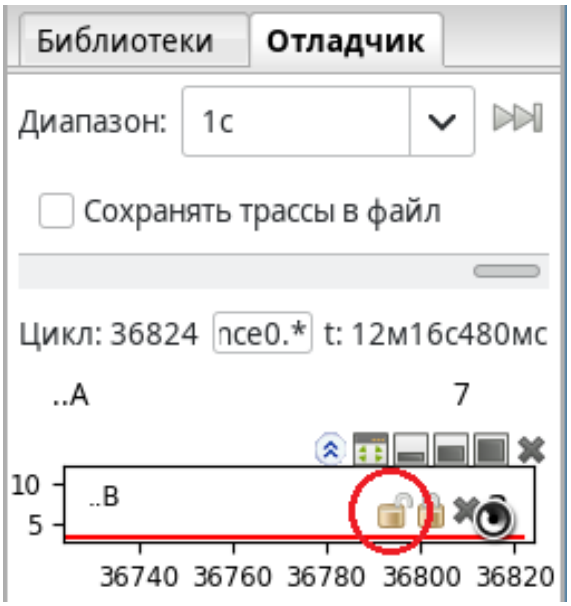


Рисунок 247 – Всплывающее меню манипуляций со значением переменной

Примечание: пользовательские типы данных (см. рис.89 – 91) не доступны для отладки в текущей реализации СРиО.

8.3.2. Отладка FBD диаграмм

Во время отладки прикладной программы, в которой часть программных модулей написано на графических языках, есть возможность видеть изменения всех значений на диаграмме и вносить необходимые изменения прямо на ней. В случае нажатия кнопки запуска режима отладки (на рис. 248 выделена красным цветом) для экземпляра программы, написанной на одном из графических языков, откроется вкладка с панелью диаграммы в режиме отладки.

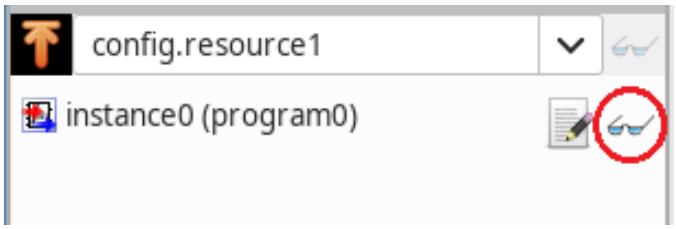


Рисунок 248 – Панель экземпляров проекта

В режиме отладки FBD диаграммы есть возможность устанавливать входные и выходные значения переменных (с помощью всплывающего меню, которые вызывается нажатием правой клавишей по соединению) для функциональных блоков, а также в целом видеть все остальные значения на входах и выходах элементов диаграммы (рис. 249).

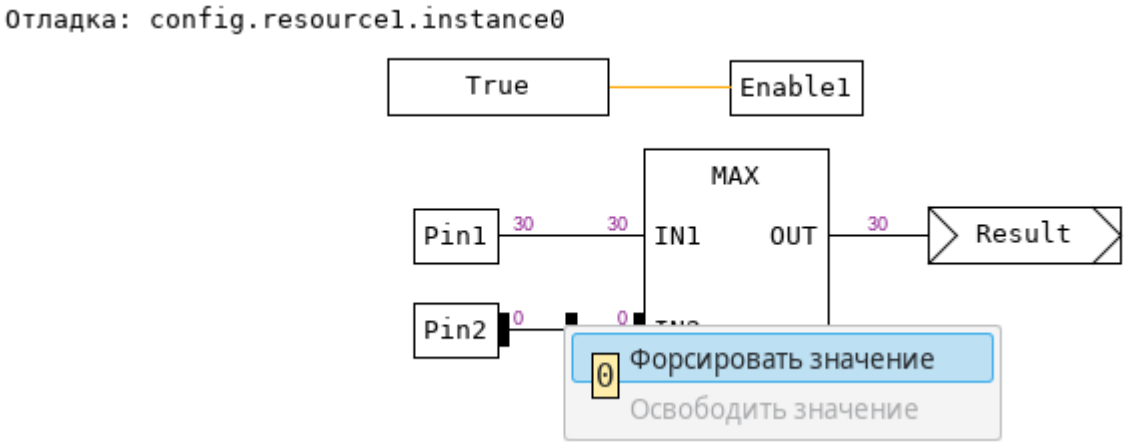


Рисунок 249 – Пример отладки FBD диаграммы

Изменённые значения в режиме отладки выделяются синим цветом. После выбора во всплывающем меню «Освободить значение» значение возвращается в то, которое получается в результате выполнения логики и алгоритма данного модуля на данном участке, а соединение на диаграмме становятся исходного цвета.

Примечание: После остановки ПЛК (кнопкой «STOP»), для внесения исправлений в проект, рекомендуется закрыть окно отладки (например, «config.resource1.instance0»). Правки в проект вносятся в окне редактирования (например, «program0», «config.resource1», «Проект»).

8.3.3. Отладка LD диаграммы

Отладка LD диаграммы осуществляется аналогично отладке FBD диаграммы. Для вызова, всплывающего меню (рис. 250), в котором можно установить желаемое значение для контакта или катушки, необходимо нажать правую клавишу мыши.

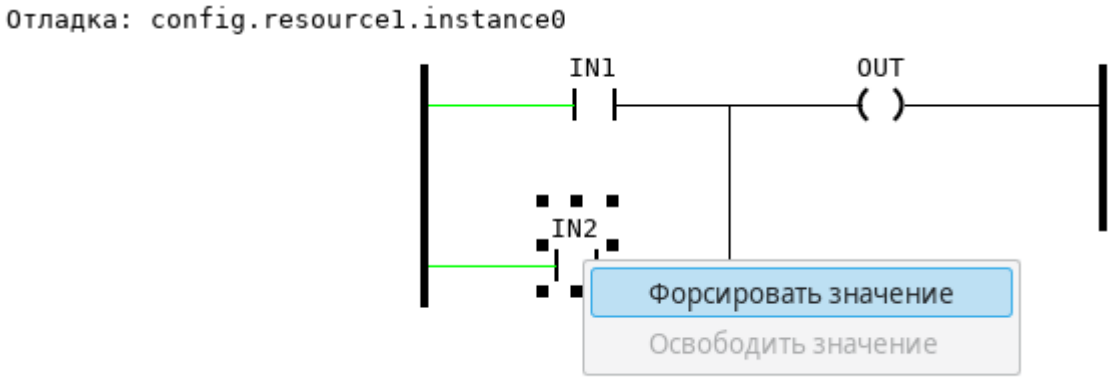


Рисунок 250 – Отладка LD диаграммы, форсирование значения

Появится диалог (рис.251), в котором нужно переключить значение типа BOOL: TRUE – контакт «ON», FALSE – контакт «OFF».

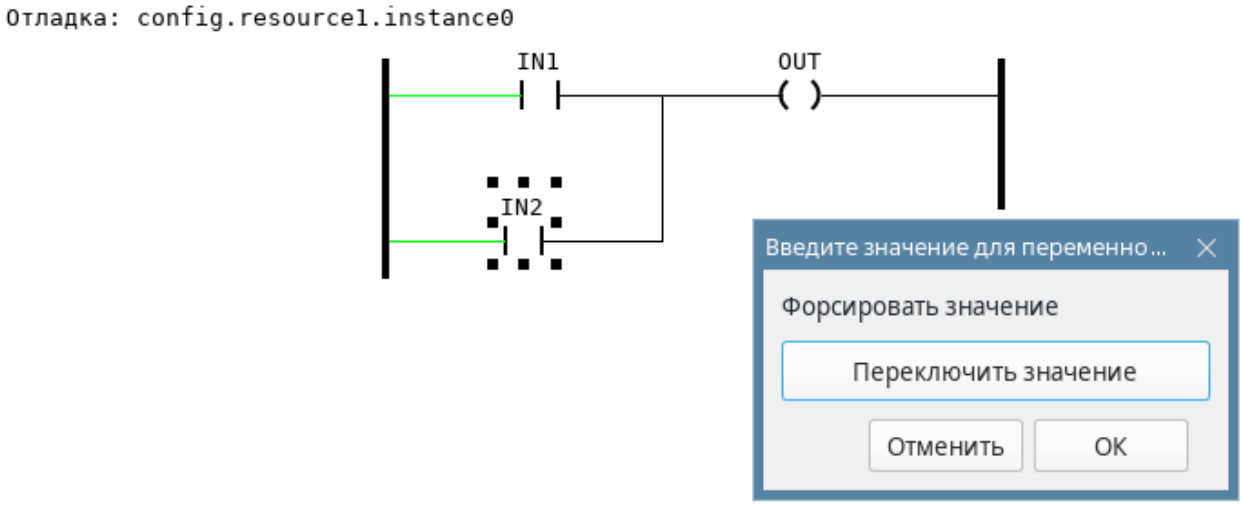


Рисунок 251 – Отладка LD диаграммы, переключение значения

После установки в режиме отладки, для данного примера, контакта «IN2» (рис.252), в значение TRUE катушка «OUT1» приняла значение TRUE, т.к. данная схема представляет собой реализацию «Логического ИЛИ» для «IN1» и «IN2».

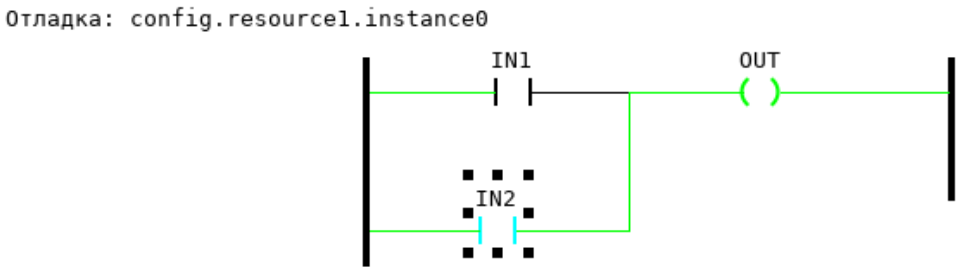


Рисунок 252 – Отладка LD диаграммы, результат переключения

8.3.4. Отладка SFC диаграммы

Отладка SFC диаграммы происходит аналогично отладке диаграмм FBD и LD. С помощью всплывающего меню (рис. 253), есть возможность устанавливать активность для шагов и переходов.

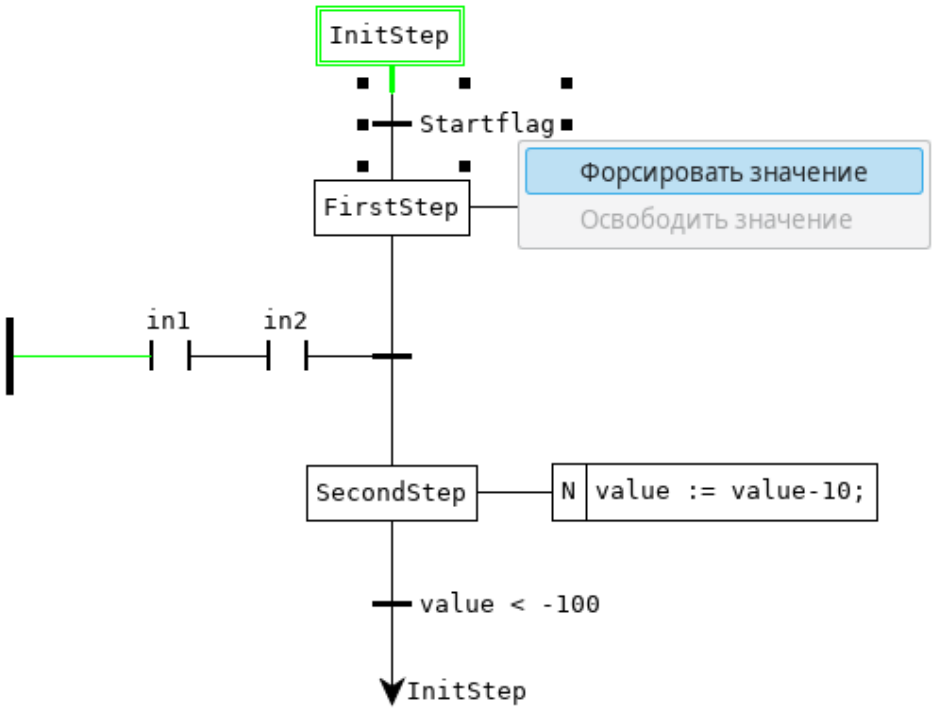


Рисунок 253 – Пример отладки SFC диаграммы

На рис. 254 показано, как устанавливается значение (после выбора «*Установить значение*», появится диалог) TRUE для шага «firstStep».

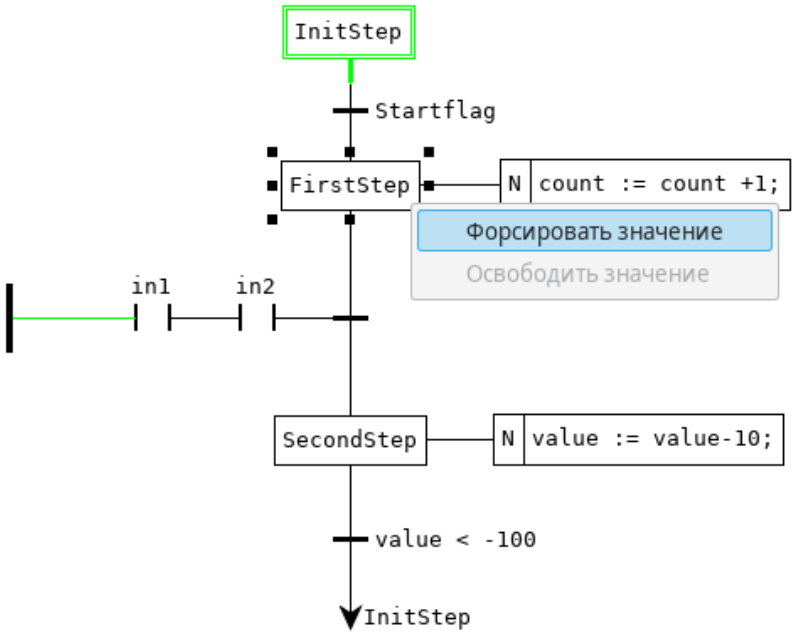


Рисунок 254 – Пример отладки SFC диаграммы

После установки значения шага в TRUE с помощью режима отладки, шаг будет выделен голубым цветом. Как можно заметить по рис. , т.к. шаг «firstStep» стал активным, блок действий, ассоциированный с ним, так же стал активным (выделен зелёным цветом), а действие внутри него, в данном случае: «`count := count + 1;`» стало выполняться.

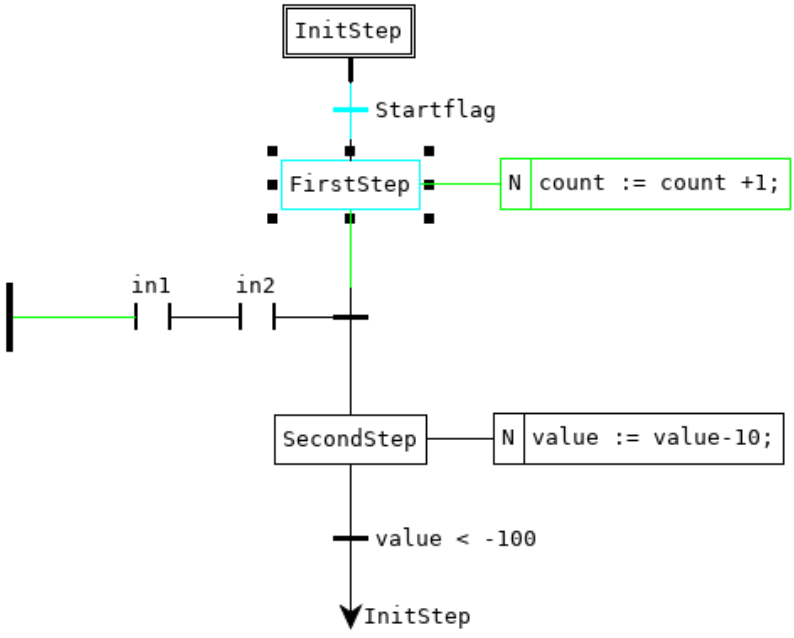


Рисунок 255 – Пример отладки SFC диаграммы

Так как квалификатор этого действия – N, то оно будет выполняться до тех пор, пока шаг активен.

8.3.5. Пошаговая отладка

В программе СРиО присутствует возможность пошаговой отладки проекта для языка ST. При пошаговой отладке отладка, описанная в п. 8.3.1 должна быть отключена. Для пошаговой отладки в редакторе языка ST необходимо с помощью левой клавиши мыши установить точки останова (выделены красным на рис.256).

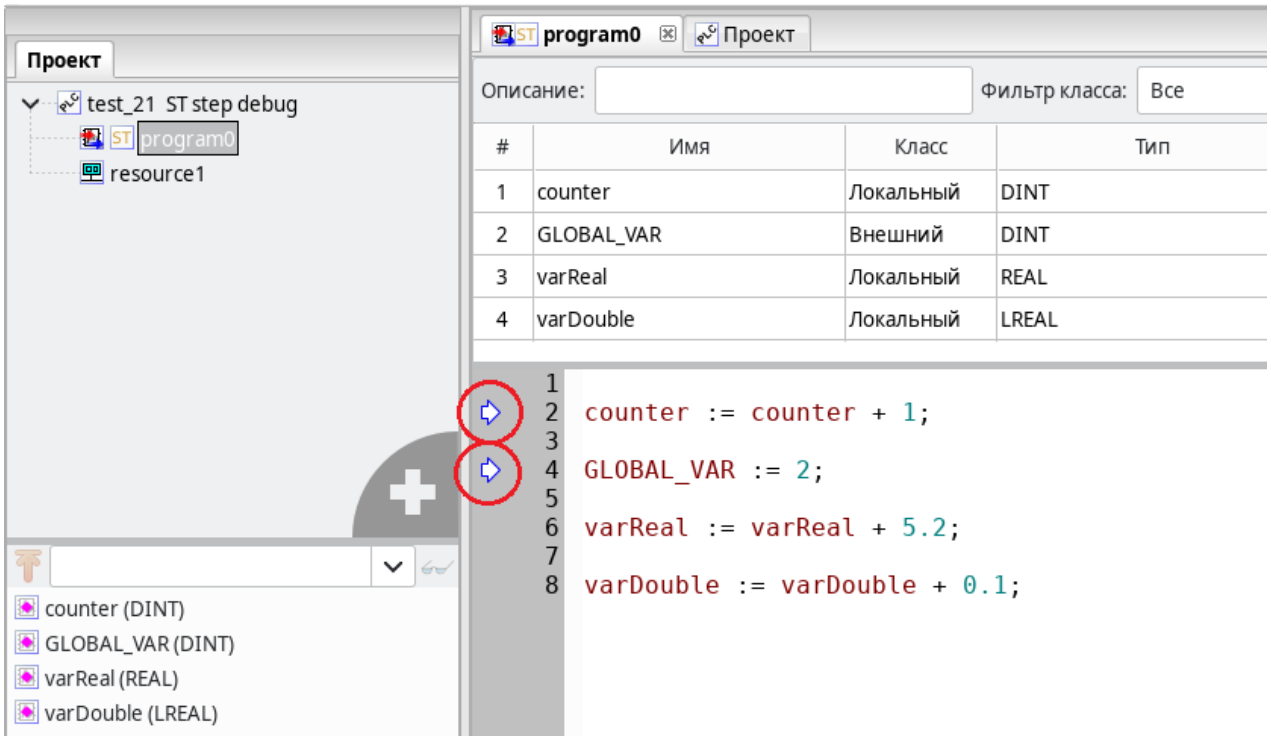


Рисунок 256 – Точки останова программы

Затем следует собрать проект и запустить его на выполнение. Нажать кнопку «Включить/выключить пошаговую отладку» на панели инструментов работы с проектом, выделена красным цветом на рис. 257:

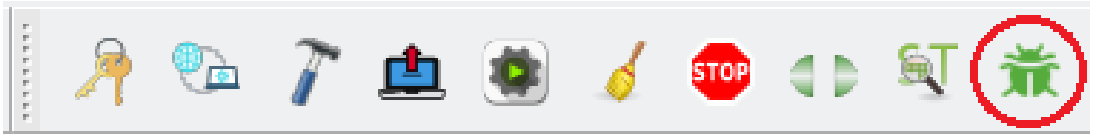


Рисунок 257 – Включение пошаговой отладки

Выполнение будет остановлено на первой точке останова. Далее работа продолжается по нажатию одном из кнопок:

– Включить/выключить пошаговую отладку;
– Продолжить пошаговую отладку до следующей точки останова;
– Продолжить пошаговую отладку до следующей строки.

Эти кнопки выделены красным цветом на рис.258:

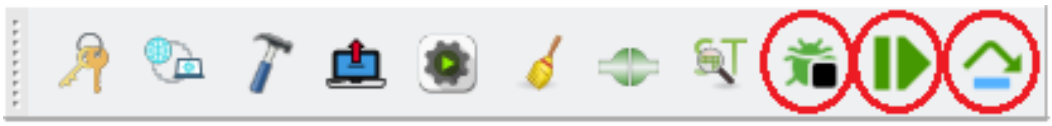


Рисунок 258 – Кнопки пошаговой отладки

Результаты пошаговой отладки отображаются в отладочную консоль, рис.259.

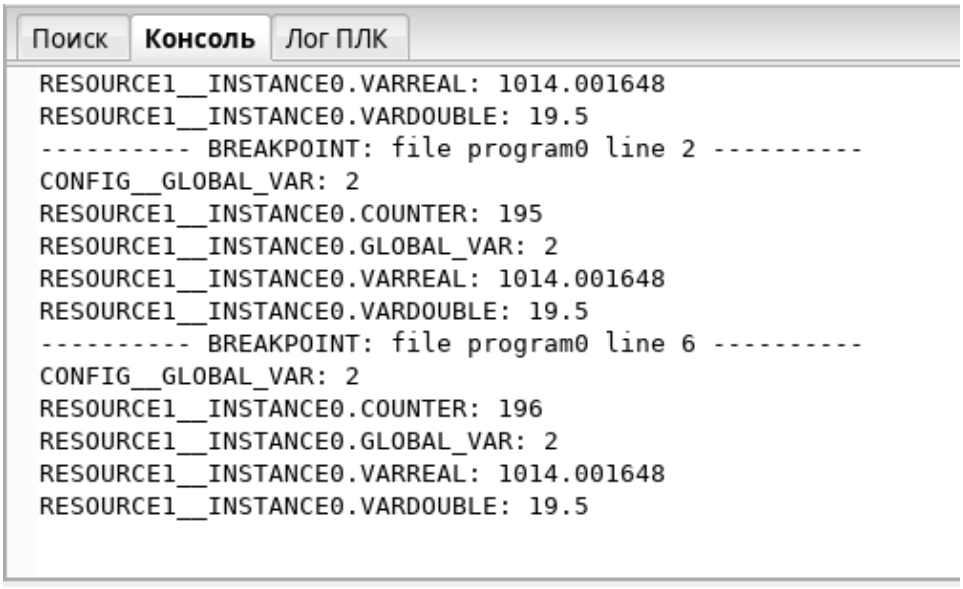


Рисунок 259 – Результаты пошаговой отладки

В режиме пошаговой отладки предлагается возможность исполнения программы либо до следующей точки останова, либо до следующей строки кода программы. В консоли выполнения проекта выводится информация о текущей точке останова, текущих значениях переменных базовых типов.

8.3.6. График изменения значения переменной

СРиО позволяет отображать в виде графика изменение значения переменной в режиме отладки. Рассмотрим вывод графика на примере программы на языке ST, в которой переменная `value` изменяется по синусоиде (рис 260).

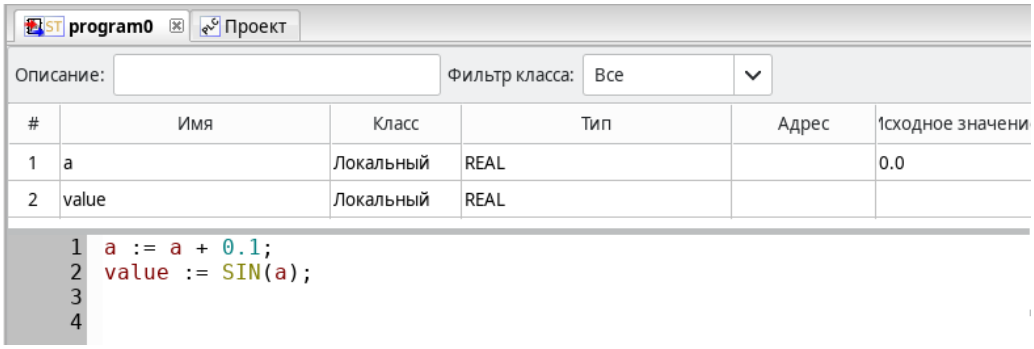


Рисунок 260 – Пример программы на языке ST

Для вывода графика, в панели «Отладчик» следует нажать кнопку, выделенную красным цветом на рис.261.

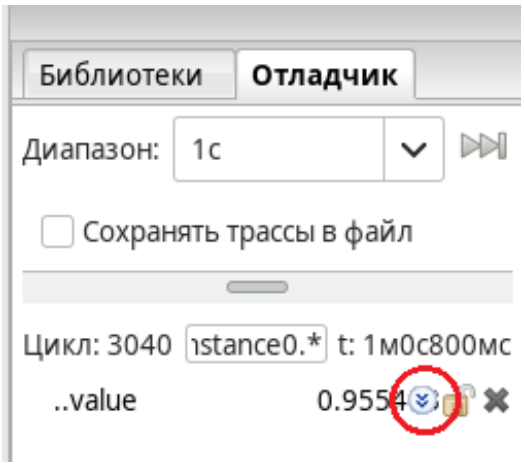


Рисунок 261 – Кнопка отображения графика изменения значения переменной

Появится график позволяющий отслеживать изменение значения переменной во времени. Рекомендуется увеличить размер окна отладчика и нажать кнопку, выделенную красным (рис.262).

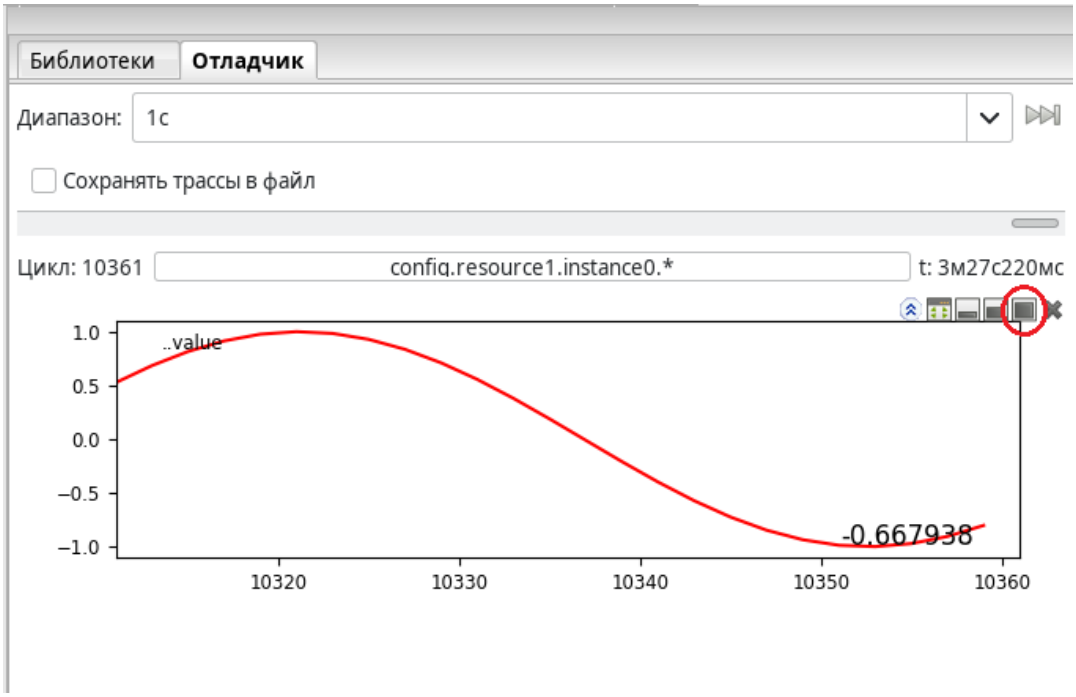


Рисунок 262 – График изменения переменной

9. ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Техническое задание на опытно-конструкторскую работу: «Создание программной платформы для разработки и отладки программного обеспечения программируемых логических контроллеров семейства Багет», шифр «ПП ПО ПЛК Багет».

2. Программные средства поддержки языков программирования высокого уровня стандарта IEC 61131-3 для ПЛК семейства СМ1820М. Руководство программиста - ЛЯЮИ.00540-01 33 01.

3. Си - компилятор для операционной системы реального времени Багет 3.3 (СКРВ Багет 3.3). Руководство программиста - ЮКСУ.90977-01 33 01.

4. Программное изделие «Операционная система реального времени Багет 2.7» (ОС РВ Багет 2.7). Описание применения.

5. [Электронный ресурс]: https://beremiz.org/

6. Программа «Конфигуратор модулей ввода-вывода для программируемого логического контроллера Багет-ПЛК1» (Конфигуратор Багет-ПЛК1) Руководство оператора ЮКСУ.91238-01 34 01.

7. ПЛК «БАГЕТ-ПЛК1» Руководство по эксплуатации ЮКСУ.421457.002РЭ

ПРИЛОЖЕНИЕ А

(информационное)

Установка программы

А.1 Установка программы СРиО для Linux Astra SE

Установка компонентов ПО производится в каталог `/opt/niisi/srio`. Для установки требуются права администратора и онлайн-доступ к официальным репозиториям ОС Linux Astra.

Исполняемым файлом является `/opt/niisi/srio/Beremiz/bagetide.sh`. В процессе установки создаётся файл-ссылка на исполняемый файл под названием `BagetIDE`. Она доступна в меню «Пуск» в разделе «Прочие», а также в поиске.

Порядок установки:

1) Обновить репозитории Astra Linux:

   – открыть файл `/etc/apt/sources.list` с правами администратора:

   ```
   sudo nano /etc/apt/sources.list
   ```

   – убедится, что репозитории из списка ниже перечислены в файле и что они не закомментированы (в начале строки не стоит символ решётки #):

   ```
   deb https://download.astralinux.ru/astra/stable/1.7_x86-64/repository-main/ 1.7_x86-64 main contrib non-free
   deb https://download.astralinux.ru/astra/stable/1.7_x86-64/repository-update/ 1.7_x86-64 main contrib non-free
   deb https://download.astralinux.ru/astra/stable/1.7_x86-64/repository-base/ 1.7_x86-64 main contrib non-free
   ```

   –добавить их при необходимости;

   – закомментировать:

   ```
   #deb cdrom[OS Astra Linux 1.7.5 1.7_x86-64 DVD ]/ 1.7_x86-64/ 1.7_x86-64 contrib main non-free
   ```

   – сохранить файл `sources.list`;

   – обновить репозитории командами:

   ```
   $sudo su
   #apt update
   ```

2) Установить «Установщик пакетов Python (PIP)» командами:

   ```
   $sudo su
   ```

```
#curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output
get-pip.py
#python2 get-pip.py
```

3) Распаковать архивный файл СРиО в рабочий каталог.
   - скопировать файл `srio<вер>.zip` в рабочий каталог.
   - ввести команду:

```
unzip srio<вер>.zip
```

   Где:

   - `<вер>` - версия СРиО;
   - рабочий каталог – каталог на инструментальной ЭВМ, из которого будет устанавливаться СРиО.

4) Установить СРиО:

   - перейти в подкаталог astra:

```
cd ./astra'
```

   - ввести команды установки:

```
$sudo su
#dpkg -i *.deb
#apt --fix-broken install
#cd /opt/niisi/srio/Beremiz
#./install.sh
#./bagetide.sh
```

Примечание: названия репозиториев и команды приведены для Astra Linux SE 1.7.5. версии «Орел».

A.1 Установка программы СРиО для ОС Linux Fedora

Установка компонентов ПО производится в каталог `/opt/niisi/srio`. Для установки требуются права администратора.

Исполняемым файлом является `/opt/niisi/srio/Beremiz/bagetide.sh`. В процессе установки создаётся файл-ссылка на исполняемый файл под названием BagetIDE (для оболочки Gnome). Она доступна в меню «Обзор» — в поиске или при нажатии кнопки «Показать приложения» на нижней панели.

Порядок установки:

1) Смонтировать привод чтения компакт-дисков:

```
sudo mount /dev/sr0 /media
```

2) Перейти в каталог: `cd /media/91292-01 12 01/fedora`

3) Ввести команду установки: `sudo yum localinstall -y *.rpm`

ПРИЛОЖЕНИЕ Б

(информационное)

Формат протокола отладки

Файл с протоколом отладки (трассой) записывается во временный каталог проекта: «/tmp/Beremiz_<XXXXX>/<имя_проекта>/build/» (где: <XXXXX> – случайные символы). Имя файла имеет вид: <временная_метка>.json (например: 2022-04-19T13:35:47.826858.json).

Файл с протоколом отладки формируется в формате JSON. Файл состоит из заголовка и массива полученных трасс.

JSON–схема заголовка:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "variables": {
      "type": "array",
      "items": [
        {
          "type": "string"
        }
      ]
    }
  },
  "required": [
    "variables"
  ]
}
```

Пример заголовка:

```
{
    "variables": [
        "CONFIG.RESOURCE1.PLC_TASK_INSTANCE.COUNTERST0.CNT"
    ]
}
```

JSON–схема массива полученных трасс:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "tick": {
      "type": "integer"
    },
    "values": {
      "type": "array",
```

```
      "items": [
        {
          "type": "string"
        }
      ]
    },
    "breakpoints": {
      "type": "array",
      "items": [
        {
          "type": "object",
          "properties": {
            "line": {
              "type": "string"
            },
            "values": {
              "type": "array",
              "items": [
                {
                  "type": "string"
                }
              ]
            }
          },
          "required": [
            "line",
            "values"
          ]
        },
        {
          "type": "object",
          "properties": {
            "line": {
              "type": "string"
            },
            "values": {
              "type": "array",
              "items": [
                {
                  "type": "string"
                }
              ]
            }
          },
          "required": [
            "line",
            "values"
          ]
        }
      ]
    }
  },
  "required": [
    "tick",
    "values"
  ]
}
```

Пример протокола отладки №1. Заголовок и массив полученных трасс для переменных `CounterST0.OUT` и `CounterFBD0.OUT`. Продолжительность 2 тика.

```
{
    "variables": [
        "CONFIG.RESOURCE1.PLC_TASK_INSTANCE.COUNTERST0.OUT",
        "CONFIG.RESOURCE1.PLC_TASK_INSTANCE.COUNTERFBD0.OUT"
    ]
}
[
    {
        "tick": 1,
        "values": [
            "1",
            "1"
        ]
    }
]
[
    {
        "tick": 2,
        "values": [
            "2",
            "2"
        ]
    }
]
```

Пример протокола отладки №2. Заголовок и массив полученных трасс для одной переменной `COUNTERST0.CNT`, с дополнительными точками снятия трасс:

```
{
    "variables": [
        "CONFIG.RESOURCE1.PLC_TASK_INSTANCE.COUNTERST0.CNT"
    ]
}
[
    {
        "tick": 2,
        "values": [
            "12"
        ],
        "breakpoints": [
            {
                "line": "BP2 - fin",
                "values": [
                    "12"
                ]
            },
            {
                "line": "BP1 - start",
                "values": [
                    "7"
                ]
            }
        ]
    }
]
```

ПРИЛОЖЕНИЕ В

(информационное)

Пример плагина для модуля MLCP

Плагин расширения для СРиО состоит из программного модуля на языке Python, отвечающего за корректное отображение настроек в пользовательском интерфейсе, и файла-шаблона на языке Си, на основании которого генерируется итоговый программный модуль, подлежащий компиляции и включению в состав итогового образа СПО ПЛК.

Чтобы СРиО инициализировала и подключила плагин, необходимо подключить его в файле: `/opt/niisi/srio/Beremiz/ide/bagetide.py` путем добавления новой записи в массив `beremiz_features.catalog` (рис. В.1).

```
beremiz_features.catalog.append(('MLCP' , 'MLCP расширение', 'Поддержка протокола
Monitoring Library Communication Protocol', 'baget.modules.mlcp.mlcp.mlcp_node'))
```

Рисунок В.1. Подключение плагина модуля MLCP.

Первое поле записи ‘MLCP’ — наименование узла в дереве проекта СРиО, затем идут краткое наименование для отображения в выпадающих подсказках и полное наименование плагина, последний параметр — Python пакет и имя класса реализующего функционал плагина.

Исходный текст плагина `mlcp.py` и файла-шаблона на языке Си `mlcp_node.c` находятся в каталоге: `/opt/niisi/srio/Beremiz/ide/baget/modules/mlcp`.

В плагине подключены модули `ConfigTreeNode` и `PLCController` (рис. В.2).

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from __future__ import absolute_import
import os

from ConfigTreeNode import ConfigTreeNode
from PLCControler import LOCATION_CONFNODE, LOCATION_VAR_MEMORY, LOCATION_GROUP
import util.paths as paths

mod_path = os.path.join(os.path.split(__file__)[0])

debug = False
```

Рисунок В.2. Подключение модулей `ConfigTreeNode` и `PLCController`.

Базовый класс плагина `mlcp_node` наследуется от базового класса `object`. Класс обязан содержать атрибут XSD, содержащий схему XML. Эта схема используется СРиО для отображения меню настроек модуля в пользовательском интерфейсе (рис. В.3)

```
class mlcp_node(object):
    XSD = """<?xml version="1.0" encoding="ISO-8859-1" ?>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="mlcpNodeObj">
        <xsd:complexType>
          <xsd:attribute name="Remote_IP_Address" type="xsd:string" use="optional"
          default="172.16.3.188"/>
          <xsd:attribute name="Url_Path_Read" type="xsd:string" use="optional"
          default="http://172.16.3.188:8888/request-r.html"/>
          <xsd:attribute name="Url_Path_Write" type="xsd:string" use="optional"
          default="http://172.16.3.188:8888/request-w.html"/>
          <xsd:attribute name="Controller_ID" type="xsd:integer" use="optional"
          default="0"/>
          <xsd:attribute name="Period" type="xsd:integer"  use="optional" default="1"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
    """

    CTNChildrenTypes = [("mlcp_tag", mlcp_tag, _("MLCP Tag"))]
```

Рисунок В.3. Класс `mlcp_node`, атрибут XSD, схема XML.

В рассматриваемом примере узел `mlcp_node` не имеет собственных переменных, доступных из языков МЭК, все теги объявляются в дочерних узлах. Поэтому объявлен атрибут `CTNChildrenTypes`, содержащий информацию о классе, реализующем дочерний узел (см. рис. В.3).

Класс `mlcp_tag` имеет также атрибут XSD (рис. В.4):

```
class mlcp_tag(object):

    XSD = """<?xml version="1.0" encoding="ISO-8859-1" ?>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="mlcpTagObj">
        <xsd:complexType>
          <xsd:attribute name="Tag_Name" type="xsd:string"  use="optional" default="tag"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
    """
```

Рисунок В.4. Атрибут XSD класса `mlcp_tag`.

Узел `mlcp_tag` описывает переменные, доступные из среды СРиО. Для этого реализован метод `GetVariableLocationTree` (рис. В.5).

Метод `Generate_Declaration` не является обязательным, в `mlcp_tag` его добавили для удобства генерации объявлений переменных (рис. В.5).

```
def GetVariableLocationTree(self):
    current_location = self.GetCurrentLocation()
    name = self.BaseParams.getName()
    tag_name = self.mlcpTagObj.getTag_Name()
    entries = []

    entries.append({
        "name": tag_name,
        "type": LOCATION_VAR_MEMORY,
        "size": 64,
        "IEC_type": "LREAL",
        "var_name": tag_name,
        "location": "L" + ".".join([str(i) for i in current_location]) + ".0",
        "description": "",
        "children": []})

    return {"name": name,
            "type": LOCATION_CONFNODE,
            "location": ".".join([str(i) for i in current_location]) + ".x",
            "children": entries}

def Generate_Declaration(self, var_name, index):
    return """
           IEC_LREAL * %s = &__values[%i];
    """ % (var_name, index)
```

Рисунок В.5. Методы `GetVariableLocationTree`, `Generate_Declaration`, `CTNGenerate_C` узла `mlcp_tag`.

Каждый узел должен реализовывать метод `CTNGenerate_C`, который вызывается каждый раз при генерации Си кода узла. Узлы `mlcp_tag` не имеют собственного кода, поэтому реализация метода возвращает пустую структуру (рис. В.5).

```
def Generate_Declaration(self, var_name, index):
    return """
           IEC_LREAL * %s = &__values[%i];
    """ % (var_name, index)
```

Рисунок В.6. Метод `CTNGenerate_C`.

В свою очередь класс `mlcp_node` реализует полноценный `CTNGenerate_C` (рис. В.6).

```
def CTNGenerate_C(self, buildpath, locations):
    loc_decl = []
    loc_vars = []
    tagnames = []
    loc_pub = []
    tags = 0
    current_location = self.GetCurrentLocation()
    prefix = "_".join(map(str, current_location))
    module_name = self.CTNType+ "_" + prefix;

        …


    loc_dict["Remote_IP_Address"] = self.mlcpNodeObj.getRemote_IP_Address();
    loc_dict["Url_Path_Read"] = self.mlcpNodeObj.getUrl_Path_Read();
    loc_dict["Url_Path_Write"] = self.mlcpNodeObj.getUrl_Path_Write();
    loc_dict["Controller_ID"] = self.mlcpNodeObj.getController_ID();
    loc_dict["Period"] = self.mlcpNodeObj.getPeriod();

    loc_dict["module_name"] = str(module_name)

    return self.DoGenerate("mlcp_node", loc_dict, buildpath, module_name);
```

Рисунок В.6. Метод `CTNGenerate_C` класса `mlcp_node`.

Метод `DoGenerate` размещает объявленные переменные в файлах-шаблонах на языке Си (`mlcp_node.c`). Их структура может быть произвольной, но обязательно должны быть объявлены функции в соответствии с табл. В.1.

Таблица В.1 Обязательные функции файла-шаблона.

№	Функция	Описание
1	`int __init_%(locstr)s (int argc, char **argv)`	Вызывается однократно при инициализации модуля.
2	`int __cleanup_%(locstr)s()`	Вызывается однократно при дкинициализации модуля.
3	`void __publish_%(locstr)s ()`	Вызывается каждый цикл исполнения.

Итоговый программный модуль на языке Си генерируется во временном каталоге проекта: «`/tmp/Beremiz_<XXXXX>/<имя_проекта>/build/MLCP_0.c`» (где: `<XXXXX>` – случайные символы).

ПРИЛОЖЕНИЕ Г

(информационное)

Сообщения программы

Основная работа в СРиО выполняется средствами графического интерфейса пользователя, см. п.6. Некоторые сообщения, выводимые через отладочную консоль, служебный терминал USB и всплывающие окна, приведены в табл. Г.1

Таблица Г.1 Сообщения о запуске проекта на Багет-ПЛК

№	Сообщение	Расшифровка
1	Подключаемся к <IP-адрес>:<порт> Где: <IP-адрес>: адрес Багет-ПЛК, например, 10.0.0.20 <порт>: порт, например, 443	Начало авторизации по нажатию кнопки «Логин» (см. п.8.2.3) или кнопки «Загрузить и запустить образ на ПЛК» (см.п.8.2.5).
2	Вход успешен	Успешная авторизация пользователя по нажатию кнопки «Логин» (см. п.8.2.3).
3	Сборка запущена в <временный каталог проекта> Где: <временный каталог проекта> - каталог на диске ИЭВМ: /tmp/Beremiz_<XXXXX>/<имя_проекта> <XXXXX> - случайные символы; <имя_проекта> - имя проекта.	Начало сборки (компиляции) проекта по нажатию кнопки «Сборка проекта в директории сборки» (см.п.8.1.2), или кнопки «Загрузить и запустить образ на ПЛК» (см.п.8.2.5). Далее выводятся промежуточные сообщения о ходе сборки проекта, о возможных ошибках или предупреждениях.
4	Сборка прошла успешно.	Сборка (компиляция) проекта по нажатию кнопки «Сборка проекта в директории сборки» (см.п.8.1.2), или кнопки «Загрузить и запустить образ на ПЛК» (см.п.8.2.5), прошла успешно.
5	Подключаемся к <IP-адрес>:<порт>	Выполняется автоматическая авторизация на ПЛК после сборки (компиляции) проекта.
6	Образ успешно отправлен на ПЛК	Образ загружен в ОЗУ Багет-ПЛК. Далее следует подключиться к целевому ПЛК (см. п.8.2.6)

7	JSONRPC connecting to URI: JSONRPC://<IP-адрес>:<порт1> Где: <IP-адрес>: адрес Багет-ПЛК, например, 10.0.0.20 <порт1>: порт, например, 1234	Подключение средств отладки и управления к Багет-ПЛК (см. п.6.5.1).
8	Последняя сборка проекта совпадает с программой в целевом ПЛК.	Собранный проект совпадает с загруженным в ПЛК.
9	Отладчик готов	В собранный проект включены средства отладки. Проект загружен в ОЗУ и готов к запуску.
10	ПЛК запускается	Сообщение о запуске проекта на ПЛК по нажатию кнопки «Запустить ПЛК» (см. п.8.2.7)

Таблица Г.2 Сообщения о запуске проекта для цели «simulation» (СС ИЭВМ)

№	Сообщение	Расшифровка
1	Удаляем старые файлы Считаем контрольную сумму проекта <путь к проекту> Готово	Подготовка к сборке проекта
2	Запуск симулятора	Запуск симулятора(СС ИЭВМ)
3	[PLC]:Waiting for connections...	Проект стартовал и ожидает подключения к симулятору (СС ИЭВМ) (см. п. 8.2.6).
4	JSONRPC connecting to URI: JSONRPC://127.0.0.1:1234	Подключение средств отладки и управления к СС ИЭВМ (см. п.6.5.1). Где: 127.0.0.1: IP-адрес СС ИЭВМ; 1234: порт.
5	Последняя сборка проекта совпадает с программой в целевом ПЛК.	Собранный проект совпадает с загруженным в ПЛК.
6	Отладчик готов	В собранный проект включены средства отладки. Проект загружен в ОЗУ и готов к запуску.
7	Последняя сборка проекта совпадает с программой в целевом ПЛК.	Собранный проект совпадает с загруженным в ПЛК.
8	Отладчик готов	В собранный проект включены средства отладки. Проект загружен в ОЗУ и готов к запуску.
9	ПЛК запускается	Сообщение о запуске проекта на симуляторе (СС ИЭВМ) по нажатию кнопки «Запустить ПЛК» (см. п.8.2.7)

10	<дата> <время> [LOG_INFO] ПЛК запущен Где: <дата>: системная дата; <время>: системное время.	Запуск проекта на симуляторе (СС ИЭВМ) после нажатия кнопки «Запустить ПЛК» (см. п.8.2.7)
11	ПЛК остановлен	Остановка проекта на симуляторе (СС ИЭВМ) после нажатия кнопки «Остановить запущенный ПЛК» (см. п.8.2.8)

Таблица Г.3 Сообщения о командах управления

№	Сообщение	Расшифровка
1	Отправлена команда перезагрузки	Сообщение об отправке команды перезагрузки по нажатию кнопки «Перезагрузка» (см. п.8.2.11)
2	Отправлена команда перезагрузки в технологический образ	Сообщение об отправке команды перезагрузки по нажатию кнопки «Перезагрузка ПЛК в технологический образ» (см. п.8.2.12)
3	Очистка директории сборки	Нажата кнопка «Очистить директорию сборки проекта» (см. п.8.1.1)
4	Удаляем старые файлы Считаем контрольную сумму проекта <путь к проекту> Готово	Сообщения при нажатии кнопки «Загрузить образ на ПЛК», в ПЗУ (см. п. 8.2.4).
5	Статус: <код> Запущен технологический образ: <техн.> Модули: <N> Тип модуля: <тип>, <инфо> Где: <код>: код статуса; <техн.>: True или False; <N>: число модулей УСО; <тип>: тип модуля УСО; <инфо>: информация о модуле УСО: адрес, статус, слово ошибок, версия прошивки.	Полученный статус ПЛК (см. п.8.2.19) Код статуса=0 соответствует отсутствию ошибок.

Таблица Г.4 Сообщения о сборке программы.

№	Сообщение	Расшифровка
1	Сборка запущена в <временный каталог проекта> Где: <временный каталог проекта> - каталог на диске ИЭВМ: /tmp/Beremiz_<XXXXX>/<имя_проекта> <XXXXX> - случайные символы; <имя_проекта> - имя проекта.	Начало сборки (компиляции) проекта по нажатию кнопки «Сборка проекта в директории сборки» (см.п.8.1.2), или кнопки «Загрузить и запустить образ на ПЛК» (см.п.8.2.5). Далее выводятся промежуточные сообщения о ходе сборки проекта, о возможных ошибках или предупреждениях.
2	(Пример промежуточных сообщений:) Генерация МЭК-61131 ST/IL/SFC кода ПЛК... Collecting data types Collecting POUs Generate Data Type datatype0 Generate Config(s) Generate POU program0 Компиляция МЭК-программы в С-код... Экспорт локальных переменных... С-код успешно сгенерирован. Запущена генерация конфигурации JSON	Пример промежуточных сообщений о ходе сборки проекта. Далее выводятся сообщения компилятора.
3	(Пример сообщений компилятора :) [MD5] <md5_сумма> ПЛК: [CC] plc_main.c -> plc_main.o [CC] plc_debugger.c -> plc_debugger.o <…> [CD] <временный каталог проекта> [SH] bash "./m_bin.sh" <…> bgtm27-gcc <…> <…> rm -f oc2000bsp.bin oc2000 bgtm27-gcc <…> -o oc2000 <…> bgtm27-objcopy <…> -g oc2000 oc2000bsp.bin <…> [DONE] Где: <…> - сообщения или параметры;	Пример сообщений компилятора.
4	Сборка прошла успешно.	Сборка (компиляция) проекта прошла успешно.

Таблица Г.5 Сообщения об ошибках

№	Сообщение	Расшифровка
1	Ошибка подключения	Введены неправильные логин и пароль при авторизации (см. п.8.2.3), либо неверно настроена связь с ПЛК (см. п.8.2.2).
2	Данная команда недоступна в текущем режиме ПЛК. Пожалуйста, перезагрузитесь в технологический образ.	Была нажата кнопка, доступная только при работе в технологическом образе (см. п.8.2.12). Требуется перезагрузка в этот образ.
3	В соединении отказано. Возможно, на ПЛК запущен технологический образ или боевой образ еще не инициализирован.	Технологический образ не включает в себя пользовательский проект. После загрузки проекта в ОЗУ Багет-ПЛК требуется время для его старта (см. п.8.2.7).
4	Текущая программа не совпадает с программой в целевом ПЛК. Работа в этом режиме может привести к непредсказуемым результатам!	Предупреждение о том, что загружаемый проект не совпадает с загруженным в ПЛК.
5	Не могу получить статус ПЛК	Не удается получить статус ПЛК. (см. п.8.2.19)

Таблица Г.6 Сообщения, выводимые в служебный USB-терминал

№	Сообщение	Расшифровка
1	CRC ok open bus as 10 starting threads	Процессорный модуль Багет-ПЛК стартовал (см.п.8.2.7), и готов к загрузке образа ОСРВ и к авторизации пользователя по кнопке «Логин».
2	[PLC]:Waiting for connections…	Проект стартовал и ожидает подключения к целевому ПЛК (см. п. 8.2.6).

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Подпись	Дата
	изменённых	заменённых	новых	аннулированных				
2	-	все	-	-	-	ЮКСУ.Н033-2024	Бухтеева	25.09.24