МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ
Директор ИЭиАС
_____ В.Р. Храмшин

10.02.2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

ЦИФРОВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Направление подготовки (специальность)
27.04.04 Управление в технических системах

Направленность (профиль/специализация) программы
Цифровые системы управления технологическими комплексами

Уровень высшего образования - магистратура

Форма обучения
очная

Институт/ факультет	Институт энергетики и автоматизированных систем
Кафедра	Автоматизированных систем управления
Курс	2
Семестр	3

Магнитогорск
2023 год

Программа практики/НИР составлена на основе ФГОС ВО - магистратура по направлению подготовки 27.04.04 Управление в технических системах (приказ Минобрнауки России от 11.08.2020 г. № 942)

Программа практики/НИР рассмотрена и одобрена на заседании кафедры Автоматизированных систем управления
25.01.2023 протокол №7

Зав. кафедрой _____ С.М. Андреев

Программа практики/НИР одобрена методической комиссией ИЭиАС
10.02.2023 г. Протокол № 7

Председатель _____ В.Р. Храмшин

Программа составлена:
доцент кафедры АСУ, канд. техн. наук _____ Е.С. Рябчикова

Рецензент:
зам. директора ЗАО "КонсОМ СКС" , канд. техн. наук

Ю.Н. Волщуков

ЗАО "КонсОМ СКС"

Лист актуализации рабочей программы

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2024 - 2025 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от __ _____ 20__ г. № _
Зав. кафедрой _____ С.М. Андреев

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2025 - 2026 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ С.М. Андреев

1 Цели освоения дисциплины (модуля)

- формирование у обучающегося способности определять общую схему системы автоматизированного управления согласно заданной структуре АСУ ТП и выполняет её реализацию с помощью цифровых систем управления;
- формирование у обучающегося способности выбирать цифровые средства контроля и регулирования технологических факторов согласно требованиям.

2 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Цифровые системы управления входит в часть учебного плана формируемую участниками образовательных отношений образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения дисциплин/ практик:

Автоматизированное проектирование систем управления

Аппаратные средства и программное обеспечение микропроцессорных технологических контроллеров

Технологические контроллеры и средства диспетчерского управления

Математическое моделирование объектов и систем управления

Встраиваемые системы управления

Современные проблемы теории управления

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения дисциплин/практик:

Подготовка к защите и процедура защиты выпускной квалификационной работы

Производственная-преддипломная практика

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Цифровые системы управления» обучающийся должен обладать следующими компетенциями:

Код индикатора	Индикатор достижения компетенции
ПК-2 Способен применять средства контроля и регулирования технологических факторов при разработке и реализации системы автоматизированного управления особо сложными технологическими процессами термической и химико-термической обработки	
ПК-2.1	Определяет общую схему системы автоматизированного управления согласно заданной структуре АСУ ТП и выполняет её реализацию
ПК-2.2	Выбирает средства контроля и регулирования технологических факторов согласно требованиям
ПК-2.3	Определяет эффективность реализованной системы автоматизированного и автоматического управления технологическим процессом

4. Структура, объём и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 6 зачетных единиц 216 акад. часов, в том числе:
– контактная работа – 63,05 акад. часов:
– аудиторная – 60 акад. часов;
– внеаудиторная – 3,05 акад. часов;
– самостоятельная работа – 117,25 акад. часов;
– в форме практической подготовки – 24 акад. час;
– подготовка к экзамену – 35,7 акад. час

Форма аттестации - экзамен

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа студента	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код компетенции
		Лек.	лаб. зан.	практ. зан.				
1. Основы управления в технических системах с использованием цифровых систем управления								
1.1 Понятие цифровых систем управления. Классификация цифровых систем управления	3	2			5	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками.	Устный опрос	ПК-2.1, ПК-2.2
1.2 Основные методы решения задач управления в технических системах с использованием цифровых систем управления. Основные этапы разработки САиУ		2			5	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками.	Устный опрос	ПК-2.1, ПК-2.2
1.3 Виды обеспечения САиУ. Наука как объект компьютеризации		2			5	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками.	Устный опрос	ПК-2.1, ПК-2.2
1.4 Анализ и выбор архитектуры САиУ		2			5	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками	Устный опрос	ПК-2.1, ПК-2.2

1.5 Цифровые системы при создании SCADA-систем		2			5	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками	Устный опрос	ПК-2.1, ПК-2.2
Итого по разделу		10			25			
2. Современные методы разработки систем автоматизации с применением цифровых систем управления								
2.1 Основы программирования ПЛК ОВЕН	3	2			10	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками	Устный опрос	ПК-2.1, ПК-2.2
2.2 Изучение языков программирования CoDeSys		3		6	10	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками. Подготовка практической работы	Устный опрос по практической работе.	ПК-2.1, ПК-2.2
2.3 Реализация системы управления простыми объектами в CoDeSys				14	20	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками. Подготовка практической работы	Устный опрос по практической работе.	ПК-2.1, ПК-2.2
2.4 Реализация системы управления технологическим процессом в CoDeSys				10		Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками. Подготовка практической работы	Устный опрос по практической работе.	ПК-2.1, ПК-2.2
2.5 Визуализация системы управления технологическим процессом в CoDeSys				10	2	Самостоятельное изучение учебной и научной литературы. Работа с электронными библиотеками. Подготовка практической работы	Устный опрос по практической работе.	ПК-2.1, ПК-2.2

2.6 Реализация системы управления сложным технологическим объектом				5	50,25	Выполнение творческого задания по индивидуальном у варианту	Доклад по творческому заданию с презентацией проекта	ПК-2.1, ПК-2.2
Итого по разделу		5		45	92,25			
Итого за семестр		15		45	117,25		экзамен	
Итого по дисциплине		15		45	117,25		экзамен	

5 Образовательные технологии

Для реализации предусмотренных видов учебной работы в качестве образовательных технологий в преподавании дисциплины «Компьютерные технологии управления в технических системах» используются:

Традиционные образовательные технологии – информационная лекция (вводная лекция, где дает первое представление о предмете и знакомство студентов с назначением и задачами курса); лекции – консультации, изложение нового материала сопровождается постановкой вопросов и дискуссией в поисках ответов на эти вопросы; практические работы.

Технологии проблемного обучения – практическое занятие в форме семинара и творческое домашнее задание, направленное на решение комплексной учебно-познавательной задачи, требующей от студента применения как научно-теоретических знаний, так и практических навыков.

Интерактивные технологии: семинар-дискуссия – коллективное обсуждение какого-либо спорного вопроса, проблемы, выявление мнений в группе. Изложение проблем и их совместное решение.

Информационно-коммуникационные образовательные технологии – в ходе проведения лекционных занятий предусматривается использование электронного демонстрационного материала (лекции-визуализации), использование Интернет ресурсов для промежуточных аттестаций и проверки остаточных знаний.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Представлено в приложении 1.

7 Оценочные средства для проведения промежуточной аттестации

Представлены в приложении 2.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Шишов, О. В. Программируемые контроллеры в системах промышленной автоматизации : учебник / О.В. Шишов. — Москва : ИНФРА-М, 2020. — 365 с. + Доп. материалы [Электронный ресурс]. — (Среднее профессиональное образование). - ISBN 978-5-16-015321-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1025245 (дата обращения: 13.05.2023). – Режим доступа: по подписке.

2. Свободно программируемые устройства в автоматизированных системах управления: Учебное пособие / Минаев И.Г., Самойленко В.В., Ушкур Д.Г. - Москва :СтГАУ - «Агрус», 2016. - 168 с.: ISBN 978-5-9596-1222-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/975920 (дата обращения: 13.05.2023). – Режим доступа: по подписке.

б) Дополнительная литература:

1. Затонский, А. В. Информационные технологии: разработка информационных моделей и систем : учебное пособие / А. В. Затонский. - Москва : РИОР : ИНФРА-М, 2020. - 344 с. - (Высшее образование: Бакалавриат). - ISBN 978-5-369-01183-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1043096 (дата обращения: 13.05.2023). – Режим доступа: по подписке.

2. Советов, Б. Я. Информационные технологии : учебник для вузов / Б. Я. Советов, В. В. Цехановский. — 7-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2020. — 327 с. — (Высшее образование). — ISBN 978-5-534-00048-1. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/449939 (дата

обращения: 13.05.2023).

3. Шишов, О. В. Технические средства автоматизации и управления : учеб. пособие / О.В. Шишов. — Москва : ИНФРА-М, 2018. — 396 с. + Доп. материалы [Электронный ресурс; Режим доступа: https://new.znanium.com]. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-010325-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/973005 (дата обращения: 13.05.2023). – Режим доступа: по подписке.

4. Юсупов, Р. Х. Основы автоматизированных систем управления технологическими процессами: Учебное пособие / Юсупов Р.Х. - Москва :Инфра-Инженерия, 2018. - 132 с. ISBN 978-5-9729-0229-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/989081 (дата обращения: 13.05.2023). – Режим доступа: по подписке.

5. Страшун, Ю. П. Технические средства автоматизации и управления : учебно-методическое пособие / Ю. П. Страшун. — Москва : МИСИС, 2015. — 154 с. — ISBN 978-5-87623-910-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/116695 (дата обращения: 13.05.2023). — Режим доступа: для авториз. пользователей.

в) Методические указания:

1. Методические указания по выполнению практических заданий представлены в приложении 3.

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение

Наименование ПО	№ договора	Срок действия лицензии
7Zip	свободно распространяемое	бессрочно
MS Office 2003 Professional	№ 135 от 17.09.2007	бессрочно
CoDeSys	свободно распространяемое	бессрочно
FAR Manager	свободно распространяемое	бессрочно

Профессиональные базы данных и информационные справочные системы

Название курса	Ссылка
Электронная база периодических изданий East View Information Services, ООО «ИВИС»	https://dlib.eastview.com/
Национальная информационно-аналитическая система – Российский индекс научного цитирования	URL: https://elibrary.ru/project_risc.asp
Поисковая система Академия Google (Google Scholar)	URL: https://scholar.google.ru/
Информационная система - Единое окно доступа к информационным ресурсам	URL: http://window.edu.ru/
Федеральное государственное бюджетное учреждение «Федеральный институт промышленной собственности»	URL: http://www1.fips.ru/
Российская Государственная библиотека. Каталоги	https://www.rsl.ru/ru/4readers/catalogues/
Электронные ресурсы библиотеки МГТУ им. Г.И. Носова	https://magtu.informsystema.ru/Marc.html?locale=ru

Университетская информационная система РОССИЯ	https://uisrussia.msu.ru
Международная база полнотекстовых журналов Springer Journals	http://link.springer.com/
Международная реферативная и полнотекстовая справочная база данных научных изданий «Springer Nature»	https://www.nature.com/siteindex
Архив научных журналов «Национальный электронно-информационный концорциум» (НП НЭИКОН)	https://archive.neicon.ru/xmlui/

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

1.Учебные аудитории для проведения занятий лекционного типа (ауд. 437)

Мультимедийные средства хранения, передачи и представления информации

2. Учебная аудитория для проведения практических занятий: компьютерный класс (ауд. 448)

Персональные компьютеры с пакетом MS Office, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета

3. Помещения для самостоятельной работы обучающихся (ауд. 448)

Персональные компьютеры с пакетом MS Office, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета

4. Учебные аудитории для групповых и индивидуальных консультаций, текущего контроля и промежуточных консультаций (ауд. 448)

Доска, мультимедийный проектор, экран

5. Помещение для хранения и профилактического обслуживания учебного оборудования (ауд. 445)

Стеллажи для хранения учебно-методический документации

Приложение 1

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Цифровые системы управления»

По дисциплине «Цифровые системы управления» предусмотрена аудиторная и внеаудиторная самостоятельная работа обучающихся.

Внеаудиторная самостоятельная работа предполагает выполнение творческого домашнего задания (индивидуально или в составе группы). Аудиторная самостоятельная работа предполагает подготовку к выполнению практических работ, а также выступление на семинаре-дискуссии, где проводится обсуждение выполненного творческого задания.

При выполнении творческого задания на основании заданной технологической схемы и описания технологического процесса обучающийся должен разработать:

- технологические требования к схеме управления;
- таблицу сигналов;
- прикладную программу для ПЛК;
- визуализацию работы системы управления;
- дать описание работы прикладной программы и визуализации.

В таблицу сигналов вносятся:

- порядковый номер переменной;
- имя переменной (не должно содержать пробелов и кириллицы);
- тип переменной (дискретный, аналоговый);
- класс переменной (локальная, глобальная);
- адрес (для внутренних переменных не заполняется).

Пример заполнения таблицы параметров:

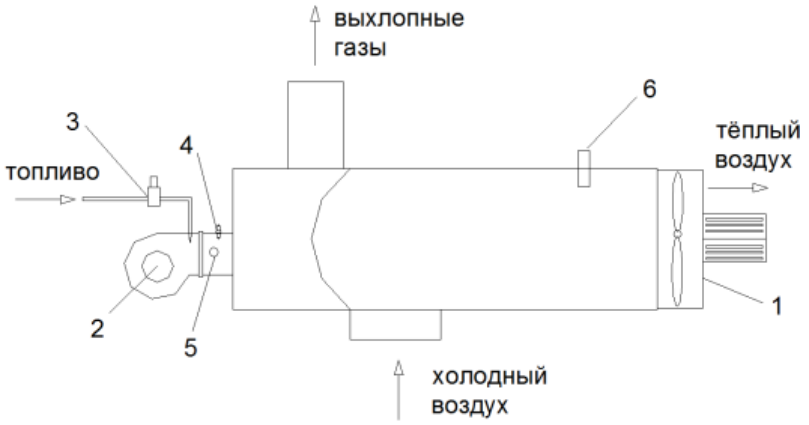
№ п/п	Наименование параметра	Имя	Тип	Класс	Адрес
1.	Температура воздуха	Temp_1	аналог.	глобал.	%IB0
2.	Кнопка «Пуск»	SB1	дискр.	локал.	-

В тексте программы необходимо учесть все технологические требования, предъявляемые к системе управления.

Работоспособность проекта обучающийся должен проверить в режиме эмуляции, а затем продемонстрировать на семинаре-дискуссии.

Примеры творческих заданий

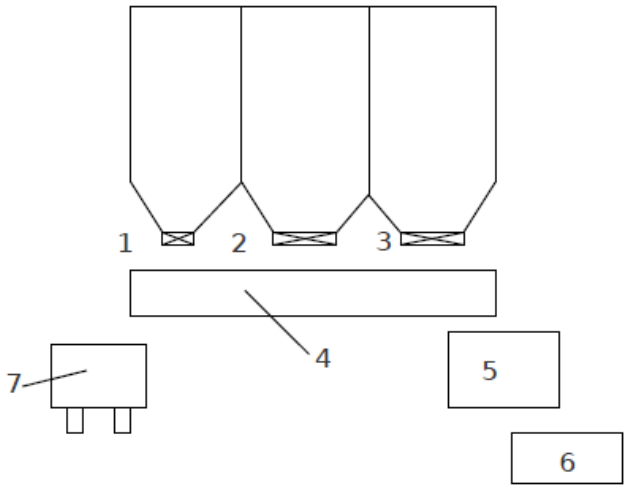
1. ***Система управления теплогенератором***



При нажатии на кнопку ПУСК, звучит предупредительная сигнализация и запускается основной вентилятор теплого воздуха 1. После запуска основного вентилятора, включается топливный вентилятор 2 для продувки (10 с). Затем включается топливный соленоидный клапан 3 и топливная смесь закачивается в камеру сгорания (5 с). Срабатывает запальная свеча 4 (4 с). Реле пламени 5 контролирует наличие пламени.

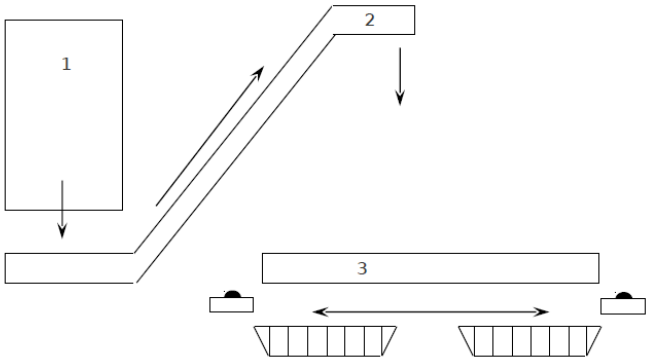
Если пламя не появилось в течение 5 с., процесс розжига выполняется еще раз (с продувки воздухом 15с.). При повторном незапуске агрегата включается продувка 1мин. и аварийная сигнализация. При нормальном запуске агрегата, система должна контролировать температуру воздуха на выходе термопреобразователем 6 и изменять скорость вращения топливного вентилятора 2. При остановке агрегата, продувка должна осуществляться до тех пор, пока температура не упадет ниже T_{min}.

2. Система управления бункерами и транспортером



Зерно поступает на транспортер 4 через одну из задвижек 1,2 или 3 или все вместе (выбор задвижки производится оператором) и далее либо в тележку 7 либо на дробилку 5 и далее в бункер 6. Схема должна отключаться при срабатывании датчика уровня в бункере 6 или при срабатывании датчика давления под тележкой.

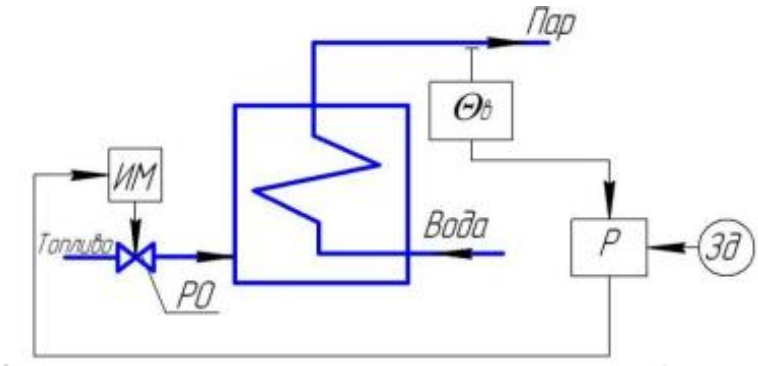
3. Система управления раздатчиком корма



Продукт на платформенный раздатчик корма 3 подается загрузочным транспортером 2 и шнековым дозатором корма из бункера 1. Платформенный раздатчик начинает движение после того, как на него падает первая порция корма. При этом транспортер 3 движется вправо. При наезде на конечный выключатель SQ1 корм сбрасывается в кормушки и транспортер останавливается. Обратное движение платформенного раздатчика начинается через одну-две секунды, при этом происходит заполнение второй половины платформенного раздатчика.

Через выдержку времени должно произойти отключение шнекового дозатора корма, а остатков корма на загрузочном транспортере 2 должно хватить для заполнения оставшейся части фронта кормления. При наезде на конечный выключатель SQ2 происходит сбрасывание корма во вторую половину кормушек и отключение всей схемы. Сброс корма в кормушки производится плужковыми сбрасывателями.

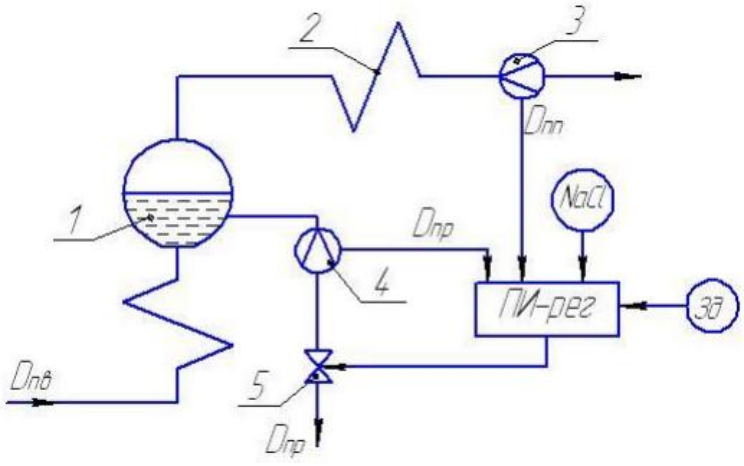
4. Система автоматического регулирования нагрузки водогрейного котла



Регулятор нагрузки котла получает импульс по температуре воды за котлом и воздействует на изменение подачи топлива к котлу. Закон управления можно выбрать любой, например, ПИ-закон управления.

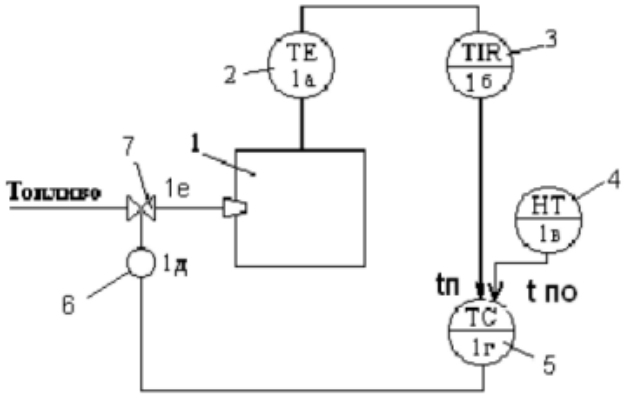
5. Система автоматического регулирования непрерывной продувки барабанного парового котла

Непрерывная продувка котла служит для удаления избытка соли NaCl и оксидов кремния SiO2, скапливающихся в котловой воде в процессе парообразования. Регулирование непрерывной продувки осуществляют воздействием регулятора продувки на регулировочный клапан на линии продувки. На вход ПИ-регулятора поступают сигналы по расходу пара Dnn и расходу продувочной воды Dnp, а также корректирующий сигнал по солесодержанию солей NaCl.



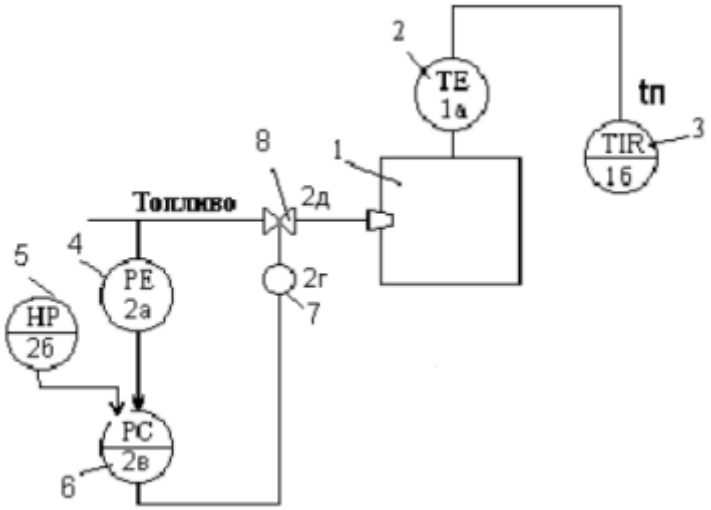
1 – барабан к/а, 2 – пароперегреватель, 3 – датчик расхода свежего пара Dпп, 4 – датчик расхода продувочной воды Dпр, 5 – регулирующий клапан продувки, ПИ-рег – регулятор продувки, Зд – задатчик ручного управления

6. *Замкнутая САР температуры в печи*



Чувствительным элементам – датчиком температуры служит термопара 2 (поз. обозн. 1а) Информация о значении температуры в печи поступает на показывающий и регистрирующий прибор 3 (поз. обозн. 1б), а с него в регулятор 5 (поз. обозн. 1г). В регулятор с задатчика 4 (поз. обозн. 1в) поступает сигнал о заданном значении температуры $t_{по}$, в состав которого входит сравнивающий элемент. Сравнивающий элемент вырабатывает отклонение $\varepsilon = t_{по} - t_{п}$, и в соответствии с алгоритмом управления, регулятор формирует управляющее воздействие. Это воздействие в виде управляющего сигнала передаётся на исполнительный механизм 6 (поз. обозн. 1д), обеспечивающий перемещение регулирующего органа 7 (поз. обозн. 1е). В качестве регулирующего органа используется поворотная заслонка в трубопроводе. Если температура в печи меньше заданной, то расход топлива увеличивается, а если больше - то уменьшается. Предусмотреть ручной и автоматический режим работы системы управления.

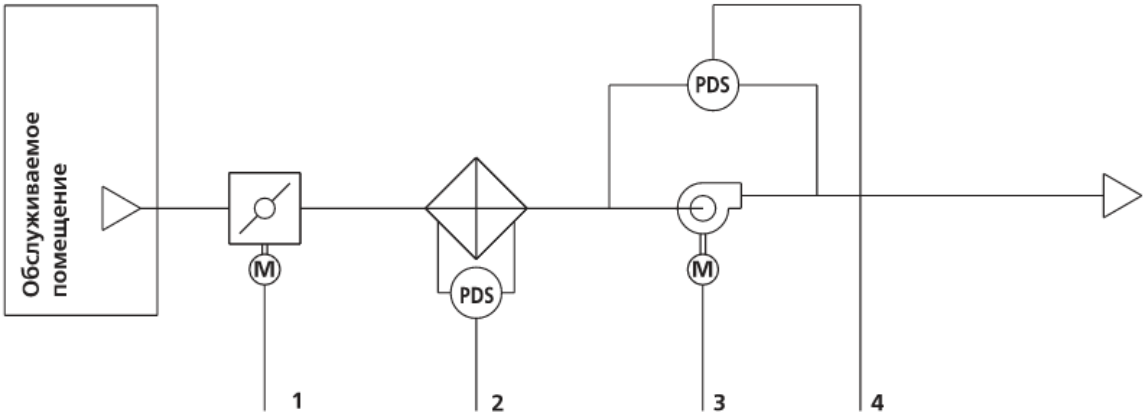
7. *Разомкнутая САР температуры в печи с регулированием по возмущению*



Регулируемой величиной является температура $t_п$ в печи 1. Основным возмущением является изменение давления газа в газопроводе, которое вызывает изменение расхода топлива и изменение температуры в печи, т.е. изменение регулируемой величины. Для компенсации влияния возмущения на значение выходной величины применяют регулятор 6 (поз. обозн. 2в), называемый компенсатором возмущений. Регулятор получает информацию о значении давления газа от датчика давления 4 (поз. обозн. 2а) и заданном значении давления от ручного задатчика 5 (поз. обозн. 2б). Затем по заранее заданной программе с помощью исполнительного механизма 7 (поз. обозн. 2г) регулятор изменяет положение регулирующего органа 8 (поз. обозн. 2д). Давление перед горелкой при правильно выбранной структуре и законе действия компенсатора не будет зависеть от давления в газопроводе и, следовательно, не будет сказываться на расходе топлива и значении температуры в печи. В этом заключается принцип компенсации возмущений. В рассмотренном примере регулируемая величина – температура в печи измеряется термопарой 2 (поз. обозн. 1а) и регистрируется прибором 3 (поз. обозн. 1а). Но эта текущая информация не используется системой регулирования, т.е. отсутствует обратная связь по результатам работы системы.

Контур компенсации возмущения разомкнут, т.е. выходная величина контура не оказывает влияния на входную величину – изменение давления в газопроводе.

8. Система управления вентиляцией (вытяжкой)



Система вентиляции (вытяжки) содержит следующие элементы:

1 — привод воздушной заслонки;
2 — датчик-реле перепада давления на фильтре (PDS);
3 — вентилятор;
4 — датчик-реле перепада давления на вентиляторе (PDS).

Система имеет два режима запуска:

- местный
- (с электрического щита);
- дистанционный
- (по команде оператора из диспетчерской с выносного пульта управления).

Режим выбирается переключателем «Вкл / Выкл / ДУ» на лицевой панели щита.

Режимы работы вытяжной вентиляции:

- автономный режим, когда включение системы происходит непосредственно со щита;
- сблокированный режим, когда включение системы происходит от приточной вентиляции.

При срабатывании внешнего датчика сигнализации «Пожар» система выключается.

Система предусматривает управление и контроль следующих параметров:

1) контроль засорения фильтра по датчику-реле перепада давления воздуха;
2) контроль работоспособности вентилятора по датчику$реле перепада давления воздуха;
3) контроль работоспособности вентилятора по токам короткого замыкания;
4) управление воздушной заслонкой электроприводом.

Описание работы системы

Запуск производится переключателем «Пуск» в положение «Вкл», загорается индикатор «Пуск».

1) Если система настроена на автономную работу (на сухих контактах установлена перемычка), происходит запуск двигателя вентилятора 3, привод 1открывает воздушную заслонку, при открытии заслонки загорается индикатор «Заслонка», работает датчик-реле 2 перепада давления на фильтре. Через определенный интервал времени включается датчик-реле 4 перепада давления на вентиляторе. При выходе вентилятора на рабочий режим загорается индикатор «Вентилятор».

2) Если система сблокирована с включением приточной вентиляции, то она переходит в режим ожидания. При запуске приточной вентиляции происходит запуск и вытяжной вентиляции. Дальнейшая работа системы аналогична автономному режиму работы.

Воздух из обслуживаемого помещения, проходя через открытую воздушную заслонку, попадает на воздушный фильтр. Если перепад давления на фильтре слишком велик, что определяется по датчику-реле 2, то на щите загорается индикатор «Фильтр». Отключение системы при этом не предусмотрено.

Датчик-реле 4 контролирует перепад давления воздуха на вентиляторе 3.

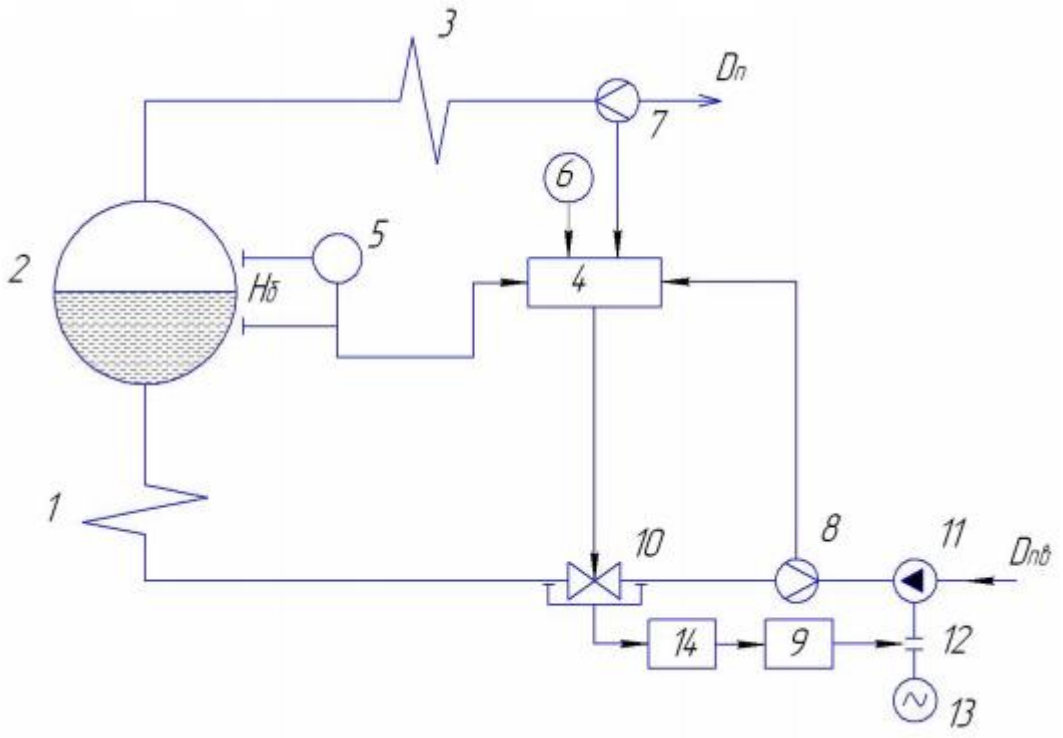
Если при запуске системы через определенный интервал времени заданный перепад давления не появляется, система останавливается. То же происходит, если указанный перепад давления исчезает во время работы системы. При этом загорается индикатор «Авария», индикатор «Вентилятор» гаснет.

9. Система автоматического регулирования питания барабана парового котла

В САР питания котла водой должен быть реализован принцип комбинированного регулирования по возмущению – при изменении расхода пара или питательной воды и отклонению – при изменении уровня воды в барабане котла. Регулятор питания должен обеспечить постоянство среднего уровня воды независимо от нагрузки котла и возмущающих воздействий.

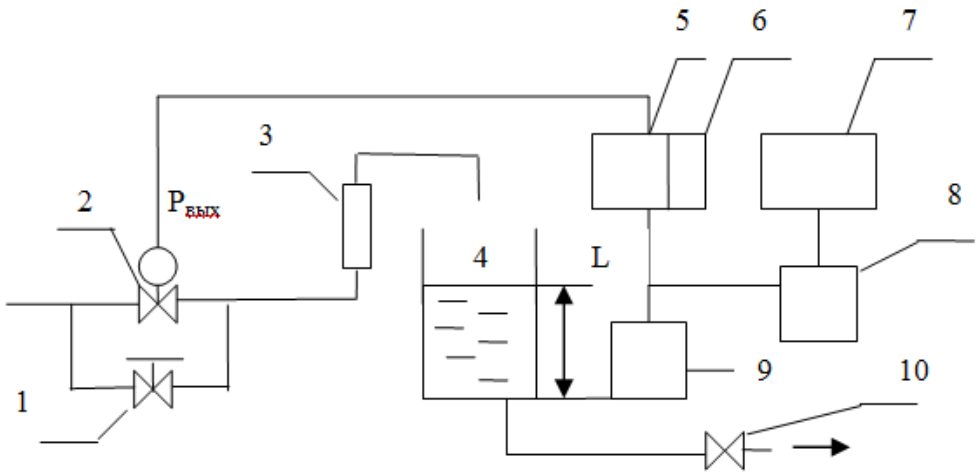
В САР питания используют для этих целей трехимпульсный регулятор питания. Сигналы по возмущению: расход свежего пара $D_п$, расход питательной воды $D_{пв}$. Сигнал по отклонению: уровень в барабане котельного агрегата $H_б$. Сигнал по расходу питательной воды используется как выключающий для снятия в статике сигнала по расходу пара. Регулятор питания перемещает регулировочный орган на линии питательной воды при появлении сигнала небаланса между расходами питательной воды и перегретого пара. Помимо этого он воздействует на положение клапана при отклонении уровня воды в барабане котельного агрегата от заданного значения. Использование сигналов $D_п$ и $D_{пв}$ обеспечивают быстродействие САР питания, сигнал $H_б$ – заданную точность поддержания уровня в барабане. В схеме измерительного блока регулятора питания датчики $D_п$, $D_{пв}$ и $H_б$ включены таким образом, что при понижении уровня воды в барабане котлоагрегата, увеличении расхода пара, уменьшении расхода питательной воды, они действуют в одном направлении – в сторону открытия питательного клапана, а при повышении уровня, уменьшении расхода пара и увеличении расхода питательной воды в сторону закрытия питательного клапана.

В качестве регулировочных органов питания используются шиберные клапаны и клапаны золотникового типа.



1-экономайзер, 2-барабан котла, 3-пароперегреватель, 4-регулятор питания, 5-датчик уровня, 6-задатчик, 7-датчик расхода пара, 8-датчик расхода питательной воды, 9-регулятор производительности, 10-питательный клапан, 11-питательный насос, 12-гидромуфта, 13- электродвигатель, 14 – дифференциальный манометр

10. Система автоматического регулирования уровня жидкости в резервуаре



В системе управления должно быть реализовано одноконтурное регулирование уровня жидкости в резервуаре.

Вода поступает в резервуар 4 (объект регулирования) по длинному трубопроводу, на котором установлены параллельно вентиль 1 и пневматический регулирующий клапан 2. Расход воды контролируют по показаниям ротаметра 3. Уровень воды в резервуаре (регулируемую величину) измеряют гидростатическим дифманометром-уровнемером 9. Пневматический сигнал с его выхода поступает па вторичный прибор со станцией управления 5 и работающий с ним в комплекте автоматический регулятор 6, а также па манометр 8 с электрическим выходом, подаваемым на регистрирующий прибор 7. Выход с регулятора направляется на исполнительное устройство - клапан 2. Вентиль 1 предназначен для подачи на объект и в систему регулирования возмущающих воздействий, вентиль 10 — для установки номинального расхода жидкости через объект. Заданное значение уровня жидкости в резервуаре устанавливают задатчиком станции управления прибора 5.

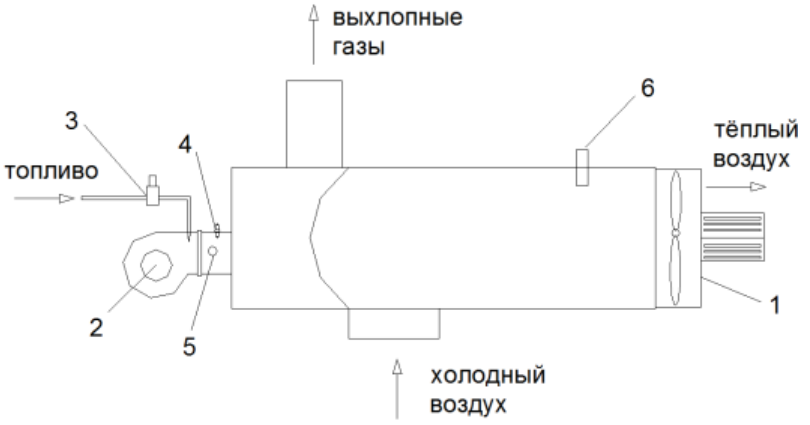
Приложение 2

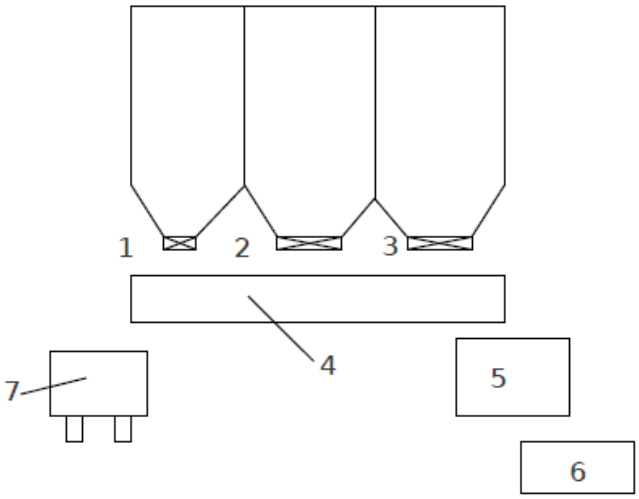
Оценочные средства для проведения промежуточной аттестации по дисциплине «Цифровые системы управления»

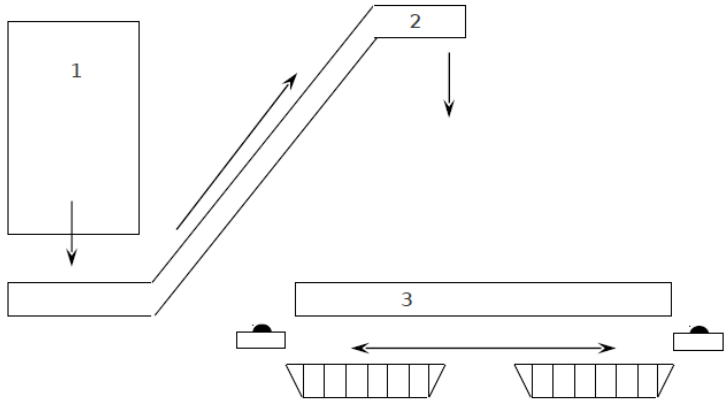
а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
ПК-2: Способен применять средства контроля и регулирования технологических факторов при разработке и реализации системы автоматизированного управления особо сложными технологическими процессами термической и химико-термической обработки		
ПК-2.1	Определяет общую схему системы автоматизированного управления согласно заданной структуре АСУ ТП и выполняет её реализацию	***Теоретические вопросы для проведения экзамена:*** 1. Общее определение промышленных контроллеров 2. Структурные компоненты контроллеров 3. Системное и прикладное программное обеспечение 4. Функциональные ресурсы, предоставляемые прикладной программе 5. Классификация контроллеров 6. Требования к языкам программирования промышленных контроллеров. Стандарт МЭК 61131-3 7. Особенности программирования ПЛК 8. Общие элементы языков стандарта МЭК 61131-3 9. Язык релейных диаграмм LD: общие сведения и примеры реализации 10. Язык релейных диаграмм LD: триггеры 11. Язык релейных диаграмм LD: таймеры 12. Язык релейных диаграмм LD: счётчики 13. Язык функциональных блоковых диаграмм FBD (общие сведения и примеры реализации) 14. Язык инструкций IL (общие сведения и примеры реализации) 15. Язык структурированного текста ST (общие сведения и примеры реализации) 16. Язык последовательных функциональных блоков SFC (общие сведения и примеры реализации)

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		17. Язык непрерывной потоковой схемы CFC (общие сведения и примеры реализации) 18. Функция и структура систем подготовки проектов 19. Пакеты создания проектов различных компаний (общий список и подробнее пакет CoDeSys) 20. Пакеты создания проектов различных компаний (общий список и подробнее пакет TIA Portal) 21. Роль и место контроллеров в структуре системы управления 22. Характеристики программно-технических комплексов (ПТК) для построения систем автоматизации 23. Классификация ПТК 24. Особенности выбора ПТК для конкретного объекта 25. Средства организации человеко-машинного интерфейса: операторные панели 26. Средства организации человеко-машинного интерфейса: панельные контроллеры 27. Цифровые промышленные сети (ЦПС): требования, общая классификация и принципы построения 28. Типовые стандартные ЦПС 29. Беспроводные локальные сети для промышленного применения 30. Классификация устройств связи с объектами (УСО) 31. УСО: нормирующие преобразователи 32. УСО: устройства удаленного сбора данных и управления 33. УСО: интеллектуальные датчики и исполнительные устройства 34. Программируемые реле ***Примеры творческих практических заданий*** 1. Разработать программу системы управления теплогенератором, предусмотреть ее визуализацию

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		 При нажатии на кнопку ПУСК, звучит предупредительная сигнализация и запускается основной вентилятор теплого воздуха 1. После запуска основного вентилятора, включается топливный вентилятор 2 для продувки (10 с). Затем включается топливный соленоидный клапан 3 и топливная смесь закачивается в камеру сгорания (5 с). Срабатывает запальная свеча 4 (4 с). Реле пламени 5 контролирует наличие пламени. Если пламя не появилось в течение 5 с., процесс розжига выполняется еще раз (с продувки воздухом 15с.). При повторном незапуске агрегата включается продувка 1мин. и аварийная сигнализация. При нормальном запуске агрегата, система должна контролировать температуру воздуха на выходе термопреобразователем 6 и изменять скорость вращения топливного вентилятора 2. При остановке агрегата, продувка должна осуществляться до тех пор, пока температура не упадет ниже T_{min}. 2. Разработать программу системы управления бункерами и транспортером, предусмотреть ее визуализацию

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		 Зерно поступает на транспортер 4 через одну из задвижек 1,2 или 3 или все вместе (выбор задвижки производится оператором) и далее либо в тележку 7 либо на дробилку 5 и далее в бункер 6. Схема должна отключаться при срабатывании датчика уровня в бункере 6 или при срабатывании датчика давления под тележкой. 3. Разработать программу системы управления распределением шихтового материала, предусмотреть ее визуализацию Продукт на платформенный раздатчик шихтового материала подается загрузочным транспортером 2 и шнековым дозатором шихты из бункера 1. Платформенный раздатчик начинает движение после того, как на него падает первая порция корма. При этом транспортер 3 движется вправо. При наезде на конечный выключатель SQ1 корм сбрасывается в бункеры и транспортер останавливается. Обратное движение платформенного раздатчика начинается через одну-две секунды, при этом происходит заполнение второй половины платформенного раздатчика.

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		 Через выдержку времени должно произойти отключение шнекового дозатора шихты, а остатков шихты на загрузочном транспортере 2 должно хватить для заполнения оставшейся части фронта шихты. При наезде на конечный выключатель SQ2 происходит сбрасывание корма во вторую половину бункеров и отключение всей схемы. Сброс корма в бункеры производится плужковыми сбрасывателями.
ПК-2.2	Выбирает средства контроля и регулирования технологических факторов согласно требованиям	***Практические задания к экзамену*** Реализовать заданный алгоритм управления на языке CFC и разработать визуализацию: 1. При срабатывании системы защиты от концевого выключателя необходимо мгновенно открыть 1-й клапан, а через 15 с закрыть 2-й и 3-й клапаны. Кроме того, в случае превышения концентрации выше некоторого заданного значения необходимо закрывать клапаны с задержкой 10 с. 2. В автоматической системе защиты имеется три датчика, установленные в различных местах производственного участка. В случае завышения значения технологического параметра выше допустимого на одном из датчиков необходимо открыть 1-й клапан на 10 с, через 3 с после его открытия закрыть 2-й клапан, а через 4 секунды после закрытия 2-го открыть 3-й на 20 с. 3. В автоматической системе имеется два дискретных датчика и четыре клапана, установленных на трубопроводах подачи сырья. В случае срабатывания первого датчика необходимо закрыть последовательно три первых клапана с задержкой 5 с. При срабатывании второго датчика необходимо закрыть третий клапан на 15 с, после чего открыть его. При одновременном срабатывании двух датчиков закрыть все клапаны на 20

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		с, после чего произвести их открытие в обратном порядке с интервалом 3с. 4. В автоматической системе имеется два дискретных датчика и шесть клапанов, установленных на трубопроводах подачи сырья. При одновременном срабатывании двух датчиков закрыть все клапаны на 20 с, после чего произвести их открытие в обратном порядке с интервалом 3 с. 5. В автоматической системе имеется дискретный датчик и пять клапанов. После пятого срабатывания датчика необходимо закрыть все клапаны с интервалом 5 с, после чего выдержать еще 4 с и произвести открытие клапанов в обратной последовательности с интервалом 1 с. 6. В автоматической системе имеется два аналоговых датчика и три клапана. В случае, если суммарное значение с датчиков превышает некоторое заданное, требуется закрыть клапаны с интервалом 3с. Если суммарное значение с датчиков равно заданному – закрыть все клапаны на 20 с. 7. В автоматической системе защиты имеется три аналоговых датчика. В случае равенства значений с датчиков некоторому заданному значению, необходимо открыть 1-й клапан на 10 с, через 3с после его открытия закрыть 2-й клапан. 8. В автоматической системе имеется два дискретных датчика. При двукратном срабатывании первого датчика и однократном срабатывании второго необходимо закрывать клапан с задержкой 10 с. 9. В автоматической системе после четвертого срабатывания дискретного датчика необходимо закрыть клапан на 8 с, после чего открыть его на 2 с. 10. В автоматической системе защиты имеется три аналоговых датчика. В случае равенства значений с двух датчиков некоторому заданному значению, необходимо открывать 1-й клапан на 3 с, с периодом 5 с. 11. В автоматической системе имеется два дискретных датчика и четыре клапана. При срабатывании второго датчика необходимо закрыть третий клапан на 15 с, после чего открыть его. При одновременном срабатывании двух датчиков закрыть все клапаны с задержкой 25с. 12. В автоматической системе при шестом срабатывании концевого выключателя необходимо последовательно

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		закрыть четыре клапана с задержкой 2с. Через 5с после закрытия последнего клапана мгновенно открыть все клапаны. 13. В системе имеется два дискретных датчика и четыре клапана. В случае срабатывания первого датчика необходимо закрыть последовательно три первых клапана с задержкой 5 с. При двукратном срабатывании второго датчика необходимо закрыть третий клапан на 15 с, после чего открыть его. 14. В автоматической системе имеется три дискретных датчика и четыре клапана. При срабатывании второго датчика необходимо закрыть третий клапан на 5с, после чего открыть его. При одновременном срабатывании двух датчиков закрывать все клапаны на 3с с периодом 5с. 15. В системе защиты имеется три датчика. В случае, когда суммарное значение с датчиков превышает некоторое заданное значение необходимо открыть 1-й клапан на 10 с, через 3 с после его открытия закрыть 2-й клапан. 16. В автоматической системе имеется два аналоговых датчика. В случае если разность между показаниями превышает некоторое заданное значение, необходимо закрыть последовательно три клапана с задержкой 5 с. Через 20с после закрытия последнего произвести их открытие в обратном порядке с интервалом 3с. 17. В автоматической системе имеется два аналоговых датчика и три клапана. В случае если частное от показаний датчиков больше, либо равно некоторому заданному значению необходимо открывать клапан на 5с с периодом 3с. 18. В автоматической системе имеется три аналоговых датчика и один концевой выключатель. При шестом срабатывании концевого выключателя или в случае превышения значения с одного из аналоговых датчиков выше некоторого заданного значения необходимо последовательно закрыть четыре клапана с задержкой 2 с. Через 5 с после закрытия последнего клапана мгновенно открыть все клапаны. 19. В автоматической системе установлены аналоговый датчик и один переключатель, имеющий положение “включено” и “выключено”. В случае превышения значения аналогового датчика выше некоторого заданного значения необходимо закрыть первый клапан

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		на 10с, после чего открыть его. В случае изменения переключателя из положения “выключено” в положение “включено” необходимо с задержкой 5с закрыть клапан на 15с. 20. В автоматической системе установлены два переключателя, имеющих фиксированные положения “включено” и “выключено”. При переходе первого переключателя из положения “выключено” во “включено” закрыть клапан на 15 с, после чего открыть его. При переходе второго переключателя из положения “включено” в “выключено” закрыть клапан на 10с, с задержкой 2с. 21. В автоматической системе установлен переключатель, имеющий фиксированные положения “включено” и “выключено”. При переходе переключателя из положения “включено” в “выключено” необходимо закрыть последовательно три первых клапана с задержкой 5с. При возвращении переключателя в исходное положение закрыть четвертый на 20с. 22. В автоматической системе установлен переключатель, имеющий фиксированные положения “включено” и “выключено”. При переходе переключателя из положения “включено” в “выключено” необходимо закрыть последовательно два клапана с задержкой 6с, после закрытия последнего открыть все клапаны с задержкой 10с. При возвращении переключателя в исходное положение закрыть третий клапан на 20 с. 23. В системе имеется дискретный датчик и переключатель. В случае срабатывания первого датчика необходимо закрыть первый клапан с задержкой 5с. При переходе переключателя из положения “включено” в “выключено” необходимо закрыть три клапана на 15 с, после чего открыть их. 24. В автоматической системе защиты имеется два аналоговых датчика. В случае равенства значений с датчиков некоторому заданному значению, необходимо открыть 1-й клапан на 10 с. В случае открытия клапана необходимо закрыть второй клапан, а при закрытии первого закрыть третий с задержкой 4с на 10с. 25. В автоматической системе имеется три дискретных датчика. При срабатывании второго датчика необходимо закрыть первый клапан на 5с, после чего открыть его. После третьего закрытия клапана закрывать еще два

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		клапана на 10с каждый. 26. При срабатывании концевого выключателя в автоматической системе необходимо мгновенно открыть 1-й клапан, а через 15 с закрыть 2-й. В случае закрытия первого клапана более трех раз необходимо закрыть третий клапан на 30с. 27. В автоматической системе имеется два дискретных датчика. При одновременном срабатывании двух датчиков закрыть первый клапан на 2с. В случае закрытия первого клапана более трех раз необходимо закрыть еще два с интервалом 5с. 28. В автоматической системе имеется пять аналоговых датчиков. В случае, если суммарное значение с двух датчиков не превышает частного от трех оставшихся датчиков требуется закрыть первый клапан на 10с. В случае двукратного закрытия первого клапана закрывать второй не более чем на 3с с периодом 2с. 29. В автоматической системе имеется два дискретных датчика. При одновременном срабатывании двух датчиков закрыть клапан на 20с. При срабатывании одного из датчиков закрыть второй клапан на 3с, а после пятого закрытия второго закрыть третий на 7с. 30. В автоматической системе имеется дискретный датчик. При пятом срабатывании датчика необходимо закрыть клапан на 6с, а в случае шестикратного закрытия клапана закрыть еще четыре клапана с интервалом 4с.

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация по дисциплине «Цифровые системы управления» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме экзамена.

Экзамен по данной дисциплине проводится в устной форме по экзаменационным билетам, каждый из которых включает 2 теоретических вопроса и одно практическое задание.

Показатели и критерии оценивания экзамена:

– на оценку **«отлично»** (5 баллов) – обучающийся демонстрирует высокий уровень сформированности компетенций, всестороннее, систематическое и глубокое знание учебного материала, свободно выполняет практические задания, свободно оперирует знаниями, умениями, применяет их в ситуациях повышенной сложности.

– на оценку **«хорошо»** (4 балла) – обучающийся демонстрирует средний уровень сформированности компетенций: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.

– на оценку **«удовлетворительно»** (3 балла) – обучающийся демонстрирует пороговый уровень сформированности компетенций: в ходе контрольных мероприятий допускаются ошибки, проявляется отсутствие отдельных знаний, умений, навыков, обучающийся испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

– на оценку **«неудовлетворительно»** (2 балла) – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач.

– на оценку **«неудовлетворительно»** (1 балл) – обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения простых задач.

Приложение 3

Методические указания к выполнению практических работ по дисциплине «Цифровые системы управления»

Практическая работа №1

Изучение языков программирования CoDeSys

Цель работы: приобретение и закрепление навыков программирования в среде программного комплекса CoDeSys на трех языках программирования (Список Инструкций (IL), Диаграммы Функциональных блоков (FBD) и Релейно-контактные схемы (LD)), а также обучение основам визуализации.

1. Теоретическое введение

В CoDeSys имеются следующие языки программирования:

1. Список Инструкций (IL). Простой машинно-независимый ассемблер.
2. Структурированный текст (ST) Высокоуровневый 'Паскаль-подобный' язык.
3. Функциональные блоковые диаграммы (FBD). Графический язык описания логических и аналоговых вычислений в очень простой и выразительной форме. CoDeSys автоматизирует составление FBD диаграмм самостоятельно размещая программные компоненты и соединения.
4. Релейно-контактные схемы (LD). Графический язык, описывающий логику работы программы в форме соединения контактов и обмоток реле. Как и в FBD, редактор LD автоматически размещает и проводит соединения компонентов схемы.
5. Последовательные функциональные схемы (SFC). Графический язык, ориентированный на описание взаимосвязанных состояний и действий системы. CoDeSys поддерживает все без исключения типы действий предусмотренные стандартом.
6. Непрерывные функциональные схемы (CFC). Редактор CFC аналогичен FBD, но в отличие от него не разделяет диаграмму на цепи, а оперирует со свободно размещаемыми компонентами. Диаграммы могут иметь обратные связи и настраиваемый порядок выполнения.

Непосредственно в CoDeSys есть возможность создать произвольное визуальное отображение. Атрибутами (цвет, размер, положение и т.д.) графических объектов управляют значения переменных проекта.

Графическая трассировка позволяет наглядно отслеживать изменения значений переменных во времени. Одновременно можно контролировать до 20 переменных и синхронизировать запуск трассировки с определенным событием.

1.1 Язык IL (Instruction list)

Язык IL (Instruction list) дословно – список инструкций. Каждая инструкция начинается с новой строки и содержит оператор и, в зависимости от типа операции, один и более операндов, разделенных запятыми.

Перед операндом может находиться метка, заканчивающаяся двоеточием **(:)**. Комментарий должен быть последним элементом в строке. Между инструкциями могут находиться пустые строки.

Пример:

```
LD 17
    ST lint (* комментарий*)
GE 5
   JMPC next
   LD idword
   EQ istruct.sdword
   STN test
next:
```

Модификаторы и операторы IL

В IL можно использовать следующие операторы и модификаторы. Модификатор **C** используется с операторами JMP, CAL, RET. Инструкция выполняется только тогда, когда результат аккумулятора ИСТИНА.

Модификатор **N** используется с операторами JMP, CAL, RET. Инструкция выполняется тогда, когда результат аккумулятора ЛОЖЬ. Модификатор **N** в других случаях означает отрицание операнда.

Ниже приведена таблица всех операторов IL с пояснениями и допустимыми модификаторами:

Таблица – Операторы IL

Оператор	Модификатор	Значение
LD	N	Присвоение аккумулятору значения оператора
ST	N	Присвоение значения аккумулятора операнду
S		Присвоить логическому операнду значение ИСТИНА, если значение аккумулятора ИСТИНА
R		Присвоить логическому операнду значение ЛОЖЬ
AND	N	Побитное И
OR	N	Побитное ИЛИ
XOR	N	Побитное исключающее ИЛИ
ADD		Сложение
SUB		Вычитание
MUL		Умножение
DIV		Деление
GT		>
GE		>=
QE		=
NE		< >
LE		<=
LT		<
JMP	CN	Переход к метке
CAL	CN	Вызов функционального блока
RET	CN	Выход из POU и возврат в вызывающую программу

Пример IL программы с использованием некоторых модификаторов:

```
LD TRUE        (*загрузить значение ИСТИНА в аккумулятор*)
ANDN BOOL1 (*выполнить И с инверсным значением переменной BOOL1*)
JMPC mark     (*если значение аккумулятора ИСТИНА, то перейти к метке «mark»*)
LDN BOOL2 (*сохранить инверсное значение BOOL2 в аккумуляторе*)
ST ERG         (*сохранить значение аккумулятора в ERG*)
```

После оператора можно поставить скобки, тогда значение выражения внутри скобок рассматривается как операнд.

Например:

```
LD 2
MUL 2
ADD 3
ST ERG
```

Здесь значение ERG равно 7. Если поставить скобки, то порядок вычислений изменится:

```
LD 2
MUL ( 2
ADD 3
    )
ST ERG
```

Теперь значение переменной ERG равно 10.

Операция MUL выполняется только тогда, когда программа доходит до «)». В качестве операнда MUL использует значение 5.

1.2 Язык релейно-контактных схем (LD)

Язык релейно-контактных схем – графический язык, реализующий структуры электрических цепей. Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи - как в FBD. Кроме того, LD достаточно удобен для управления другими компонентами POU.

Диаграмма LD состоит из ряда цепей. Слева и справа схема ограничена вертикальными линиями - шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями. Слева любая цепь начинается набором контактов, которые посылают слева направо состояние «ON» или «OFF», соответствующие логическим значениям ИСТИНА или ЛОЖЬ. Каждому контакту соответствует логическая переменная. Если переменная имеет значение ИСТИНА, то состояние передается через контакт. Иначе правое соединение получает значение выключено («OFF»).

Контакт

Контакты обозначаются двумя параллельными линиями и могут иметь состояния «ON» или «OFF». Эти состояния соответствуют значениям ИСТИНА или ЛОЖЬ. Каждому контакту соответствует логическая переменная. Если значение переменной ИСТИНА, то контакт замкнут.

Контакты могут быть соединены параллельно, тогда соединение передает состояние «ON», когда хотя бы одна из ветвей передает «ON». Если контакты соединены последовательно, то для того, чтобы соединение передало «ON», необходимо, чтобы оба контакта передавали «ON». Это соответствует электрической параллельной и последовательной схеме. Контакт может быть инвертируемым. Такой контакт обозначается с помощью символа |/| и передает состояние «ON», если значение переменной ЛОЖЬ.

Обмотка

В правой части схемы может находиться любое количество обмоток (реле), которые обозначаются круглыми скобками **().** Они могут соединяться только параллельно. Обмотка передает значение соединения слева направо и копирует его в соответствующую логическую переменную.

В целом цепь может быть либо замкнутой (ON), либо разомкнутой (OFF). Это как раз и отражается на обмотке и соответственно на логической переменной обмотки (ИСТИНА/ЛОЖЬ).

Обмотки также могут быть инверсными. Если обмотка инверсная (обозначается символом **(/)**), тогда в соответствующую логическую переменную копируется инверсное значение.

Функциональные блоки в LD

Кроме контактов и обмоток, в LD можно использовать функциональные блоки и программы. Они должны иметь логические вход и выход и могут использоваться так же, как контакты.

SET и RESET обмотка

Обмотки могут быть с «самофиксацией» типов SET и RESET. Обмотки типа SET обозначаются буквой «S» внутри круглых скобок (S). Если соответствующая этой обмотке переменная принимает значение ИСТИНА, то она навсегда (до сброса R) сохраняет его.

Обмотки типа RESET обозначаются буквой R. Если соответствующая переменная принимает значение ЛОЖЬ, то она навсегда (до установки S) сохраняет его.

LD в качестве FBD

Весьма вероятно, что при работе с LD вы захотите с помощью контакта управлять другими POU.

Во-первых, можно использовать обмотку для передачи значения глобальной переменной, которая будет использоваться в другом месте. Кроме того, можно вставить вызов прямо в схему LD.

Такой POU может быть оператором, функцией, программой или функциональным блоком, который имеет добавочный вход, обозначаемый EN. Вход EN всегда логического типа, и POU выполняется, только когда значение EN=ИСТИНА. POU встраивается в схему параллельно обмоткам, и вход EN соединяется ответвлением. Использование таких POU делает LD схему похожей на FBD схему.

1.3 Язык функциональных блоковых диаграмм (FBD)

FBD – это графический язык программирования. Он работает с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата.

Редактор FBD - графический редактор, который работает со списком цепей, каждая из которых состоит из логических или арифметических выражений, вызовов функций, программ или функциональных блоков, инструкций возврата и перехода.

Наиболее важные функции вы можете найти в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>.

Позиция курсора в FBD

Текстовый курсор может устанавливаться в любую часть FBD цепи, содержащую текст. Выбранный текст выделяется синим и может быть изменен. Текущую позицию графического курсора можно увидеть по прямоугольнику с пунктирной границей.

Установка позиции курсора

Позицию курсора можно установить с помощью левой кнопки мыши или с помощью клавиатуры. При использовании клавиш перемещения вы будете менять текущую позицию курсора на соседнюю в заданном направлении. При использовании этого способа можно выбрать любую позицию курсора, в том числе и текстовое поле. Клавиши вверх и вниз позволяют выбрать предыдущую и следующую позицию курсора.

Пустая схема содержит только три знака вопроса «???». Такую схему можно выбрать, щелкнув на них мышью.

“Insert” “Assign”

Быстрый ввод:<Ctrl>+<A>

Эта команда вставляет инструкцию присваивания в схему.

В зависимости от позиции курсора присваивание будет вставлено прямо перед выбранным входом, перед выходом или в конце схемы. После вставки присваивания появятся три знака вопроса, выделив которые, можно вводить имя переменной. Имя переменной удобно вводить с помощью Ассистента ввода (F2).

Обратите также внимание на возможность ввода адресов вместо имен переменных.

Чтобы ввести дополнительное присваивание к существующему, используйте команду “Insert” “Output”.

“Insert” “Jump”

Быстрый ввод:**<Ctrl>+<L>**

Эта команда вставляет инструкцию перехода.

В зависимости от позиции курсора инструкция перехода будет вставлена прямо перед выбранным входом, перед выходом или в конце схемы.

После вставки инструкции перехода появятся три знака вопроса, выделив которые, можно вводить имя метки.

“Insert” “Return”

Быстрый ввод:**<Ctrl>+<R>**

Эта команда вставляет инструкцию возврата Return.

В зависимости от позиции курсора инструкция возврата будет вставлена прямо перед выбранным входом, перед выходом или в конце схемы.

“Insert” “Box”

Быстрый ввод:**<Ctrl>+<B>**

С помощью этой команды в схему можно вставлять операторы, функции, функциональные блоки и программы. Сразу после выполнения этой команды в схеме появляется оператор “AND”. Выбрав текстовое поле, где написано “AND”, этот оператор можно превратить в любой другой объект (функцию, функциональный блок, программу, оператор), написав имя желаемого объекта. Это имя удобно выбирать, используя Input Assistant (<F2>). Если новый блок имеет другое число входов, чем оператор AND, то будут добавлены новые входы или удалены ненужные.

В функциях и функциональных блоках изображаются формальные входные и выходные параметры.

Над функциональными блоками находится поле, в котором нужно ввести имя экземпляра функционального блока. Если тип функционального блока введен не корректно (функциональный блок не описан), то появляется блок, имеющий два входа. Если выбрано поле

ввода имени экземпляра функционального блока, то с помощью клавиши <F2> можно вызвать Input Assistant.

Новый POU вставляется в выбранную позицию:

1. Выбран вход блока. В этом случае POU вставляется в позицию перед входом. Первый вход этого POU соединяется с ветвью, ранее соединенной с выбранным входом. Выход POU соединяется с выбранным входом.
2. Выбран выход, тогда POU вставляется после этого выхода. Первый вход этого POU соединяется с выбранным выходом. Выход вставленного POU соединяется с ветвью, ранее соединенной с выбранным выходом.
3. Выбран POU, тогда старый блок будет заменен на новый. Насколько это возможно, новый блок будет присоединен к схеме так же, как и старый. Если новый элемент имеет меньше входов, чем старый, то ненужные ветви будут удалены. То же верно и для выходов.
4. Выбрана инструкция перехода или возврата, тогда POU будет вставлен перед ней. Первый вход этого POU соединяется с ветвью, ранее соединенной слева с выбранным элементом. Первый выход этого POU соединяется с ветвью, ранее соединенной справа с выбранным элементом.
5. Выбрана последняя позиция схемы. Новый POU соединяется с последним блоком схемы.

Все входы POU, которые не удалось соединить автоматически, соединяются с тремя знаками вопроса. Этот текст можно заменить на имя переменной или константу.

Если справа от вставленного POU находится ветвь, то она будет соединена с первым выходом этого POU.

“Insert” “Input”

Быстрый ввод:**<Ctrl>+<U>**

Добавляет вход оператора. Некоторые операторы могут иметь переменное число входов (например, ADD может иметь два и более входа). Для того чтобы добавить вход, выберите уже существующий вход, перед которым вы хотите вставить новый и выполните команду “Insert” “Input”. Есть другой способ: выберите оператор и выполните команду “Insert” “Input”, тогда новый вход будет самым нижним.

Слева от вставленного входа появится строка “???”. Вместо нее нужно ввести имя переменной или константу, для чего можно воспользоваться Input Assistant.

“Insert” “Output”

Добавляет новое присваивание к уже существующему. Это позволяет передать одно значение сразу нескольким переменным. Если вы выберите пересечение линий над присваиванием или выход прямо перед ним, то после уже существующего присваивания будет вставлено новое.

В случае, если линии пересекаются прямо перед выбранным присваиванием, то новое присваивание будет вставлено перед выбранным.

Слева от вставленного присваивания появится строка “???”. Вместо нее нужно ввести имя переменной или константу, для чего можно воспользоваться Input Assistant.

“Extras” “Negate”

Быстрый ввод:**<Ctrl>+<N>**

С помощью этой команды можно инвертировать входы, выходы, инструкции перехода или возврата.

Символ отрицания – небольшая окружность на месте соединения.

Если выбран вход, то этот вход будет инвертирован. То же верно и для выхода.

При инвертировании инструкций перехода или возврата они выполняются, если ветвь, к которой они присоединены, передает FALSE.

Снять отрицание можно через повторное отрицание.

“Extras” “Set/Reset”

При помощи этой команды вы можете определить Set- и Reset-выходы. Set-выход обозначается буквой S, а Reset-выход – буквой R.

Set-выход принимает значение TRUE, а Reset-выход – значение FALSE, если ветвь, к которой они присоединены, передает TRUE. Если эта ветвь передает FALSE, то переменные сохраняют свои значения.

При многократном выполнении этой команды можно получить Set-выход, Reset-выход и обычный выход.

“Extras“ “View“

Используя эту команду, можно использовать редактор LD или FBD для программных компонентов (POU) созданных в FBD редакторе. Это возможно как в Offline так и в Online режимах.

Open instance

Команда аналогична команде ‘Project’ ‘Open instance’ . Она присутствует в контекстном меню (<F2>) и в меню 'Extras' , если курсор установлен на имени функционального блока в графическом или текстовом редакторе.

Команды вырезать (Cut), копировать (Copy), выделить (Paste) и удалить (Delete) в FBD

Эти команды можно найти в меню Edit.

Если выбрано пересечение линий, то присваивания, инструкции перехода или возврата, расположенные под пересекающимися линиями, будут удалены, вырезаны или скопированы.

Когда выбрано POU, то эти действия будут выполнены над выбранным объектом и всеми ветвями, которые соединяют этот объект со схемой.

Кроме того, ветви, полностью расположенные перед позицией курсора, будут вырезаны, удалены или скопированы.

Скопированные или вырезанные части схемы находятся в буфере и могут быть вставлены в нужное место, которое перед этим нужно выбрать. Можно выбирать входы и выходы.

Если POU вставляется из буфера (не забудьте, что в этом случае все соединяющие ветви, кроме первой, хранятся в буфере как единое целое), первый вход соединяется с ветвью перед выбранной точкой.

В другом случае (из буфера вставляется не POU), ветвь, находящаяся перед выбранной точкой, полностью заменяется на содержимое буфера.

В обоих случаях последний вставляемый элемент соединяется с ветвью, расположенной справа от выбранной точки.

С помощью вырезания и вставки решается следующая проблема: новый оператор вставляется в середину схемы; ветвь, расположенная справа от оператора, теперь соединяется с первым входом, но может быть соединена со вторым. Вы должны выбрать первый вход и выполнить команду “Edit” ”Cut”. Затем, выделите второй вход и выполните команду “Edit” ”Paste”. Теперь ветвь соединится со вторым входом**.**

FBD диаграмма в режиме Online

В режиме Online в редакторе FBD можно устанавливать точки останова. Если в цепи была установлена точка останова, то номер соответствующей цепи станет синим. Выполнение программы останавливается перед цепью, в которой установлена точка останова. В этом случае номер цепи становится красным.

Используя команду “Step in” или ”Step over”, можно последовательно выполнять цепи, останавливаясь после каждой. На экран выводится текущее значение каждой переменной. Исключение составляет тот случай, когда вход функционального блока – это выражение. Тогда выводится только значение первой переменной в выражении.

Двойной щелчок мышью по переменной выводит диалоговое окно для ввода нового значения переменной. Если переменная является логической, то диалоговое окно не выводится, а значение переменной просто переключается. Для записи значения переменных в контроллер используется команда “Online” ”Write values”. После этого переменные снова становятся черными.

Контроль потока выполнения программы запускается с помощью команды “Online” ”Display Flow Control”. Использую эту команду, вы можете просмотреть значения, передаваемые по линиям соединения.

Если линии соединения передают не логические значения, то эти значения изображаются в отдельных полях. Поля для переменных, которые не используются, изображаются серым цветом.

Если линия передает значение TRUE, то она изображается синим. Эта команда позволяет наблюдать за потоком информации во время выполнения программы.

В режиме Online, если вы переместите указатель мыши на переменную, то в подсказке появится тип, комментарии и адрес этой переменной.

1.4 Визуализация

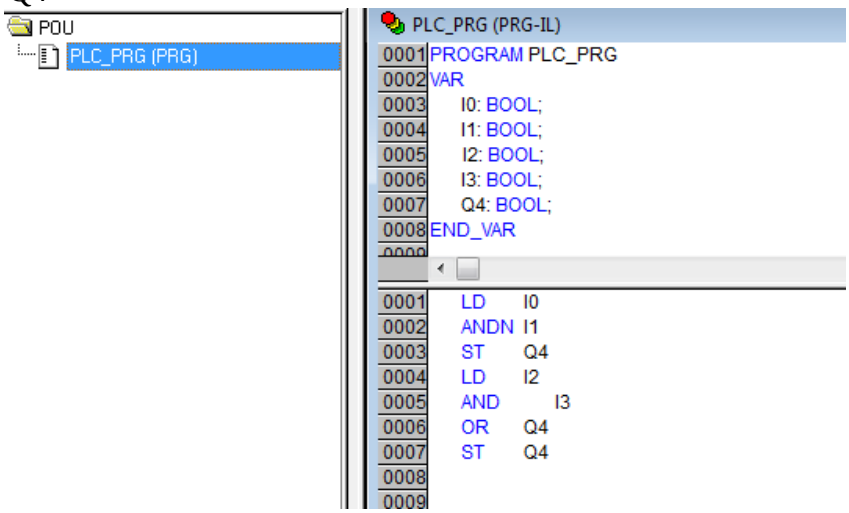
Под термином «визуализация» будем понимать графический человеко-машинный интерфейс, представляющий собой набор условных изображений технологического процесса и его стадий, называемых «мнемосхемами», с отображением значимых технологических параметров и ситуаций, а также средств взаимодействия с автоматизированной системой управления.
На мнемосхемах изображаются (условно) агрегаты технологической линии. Чаще всего создается основная мнемосхема, на которой изображается вся технологическая цепочка и наиболее важные технологические параметры, а также набор мнемосхем для каждого агрегата, на которых можно отследить и задать технические параметры его работы. Также могут быть созданы отдельные мнемосхемы для энергоучета, формирования различного рода отчетности и т.д.
Визуализация на платформе контроллера имеет существенно меньше возможностей по сравнению с визуализацией SCADA-системы. Кроме того, дополнительные возможности контроллера (формирование отчетности, хранение данных и т.д.) на порядок ниже возможностей промышленного компьютера. Однако, для нужд небольших систем управления их может быть вполне достаточно, и тогда визуализация, реализуемая в контроллере может использоваться для отображения и управления с помощью графический панелей оператора.
Основное назначение визуализации – организация рационального и эргономичного взаимодействия оператора и системы управления. Поэтому организация ее объектов должна быть предельна проста и информативна.
Для создания визуализации в CoDeSys нужно перейти на вкладку Визуализация в организаторе проекта и в контекстном меню выбрать ***Добавить объект***.
Для вставки элементов в окно визуализации следует воспользоваться панелью элементов, расположенных над окном. При двойном щелчке на созданном объекте (или выборе пункта контекстного меню Конфигурировать) появляется окно ***Конфигурирование элемента***. На вкладках этого окна задаются свойства выбранного объекта и привязываются переменные проекта, управляющие и управляемые этим объектом.
На вкладке ***Форма*** определяется форма фигуры. На вкладке ***Текст*** задается текст связанный с элементом, а также параметры этого текста. На вкладке ***Цвета*** можно определить цвет объекта, например, кнопки, в выключенном состоянии (верхняя панель) и во включенном состоянии (нижняя панель, тревожный цвет).
Для того, чтобы элементы меняли цвет в соответствии с состоянием переменных, нужно на вкладке ***Переменные*** окна **Конфигурирование элемента** соответствующей лампы/кнопки в поле *Изменение цвета* ввести имя переменной из конфигурации ПЛК, которая отвечает за включение аналогичного выхода ПЛК (переменную нужно выбрать из списка после нажатия кнопки F2).
В визуализации можно предусмотреть возможность изменения состояния переменной при нажатии на соответствующую кнопку. Для этого в окне **Конфигурирование элемента** на вкладке ***Ввод*** нужно поставить галочку на пункте *Переменная кнопка* и в текстовом поле ввести имя соответствующей глобальной переменной.

2. Порядок выполнения работы

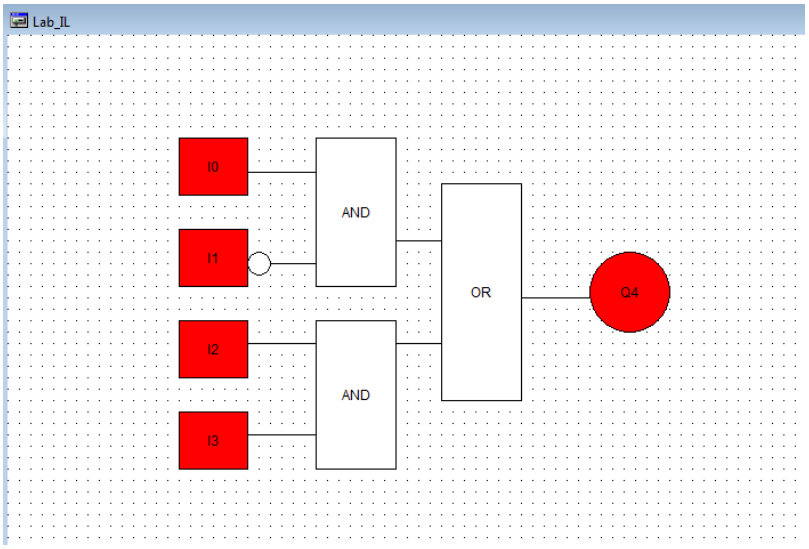
2.1. Исследование языка IL

1. Запустить CoDeSys V2.3.
2. Создать новый проект. Выполнить команду: Файл, Создать.
3. Выбрать целевую платформу (Target Settings) 3S CoDeSys SP PLCWinNT V2.4.
4. Определить тип первого программного компонента (New POU) – PLC_PRG. Выбрать язык программирования – IL и тип компонента – программа. В однозадачных проектах система исполнения циклически вызывает программу PLC_PRG.
5. Объявить переменные I0, I1, I2, I3, Q4. В диалоге определения переменных определить тип переменных – BOOL и класс переменных – VAR GLOBAL.
6. В редакторе набрать текст программы согласно своему варианту. Например,

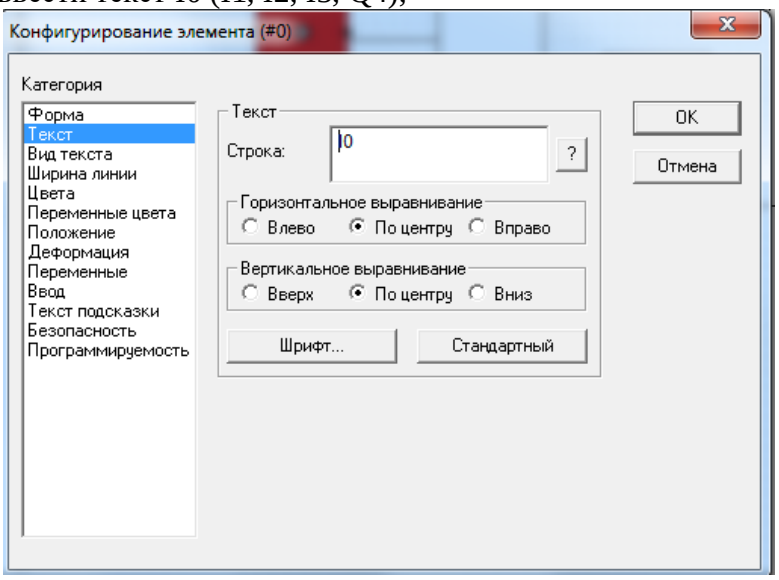
```
        LD      I0
ANDN    I1
ST      Q4
LD      I2
AND     I3
OR      Q4
ST      Q4
```



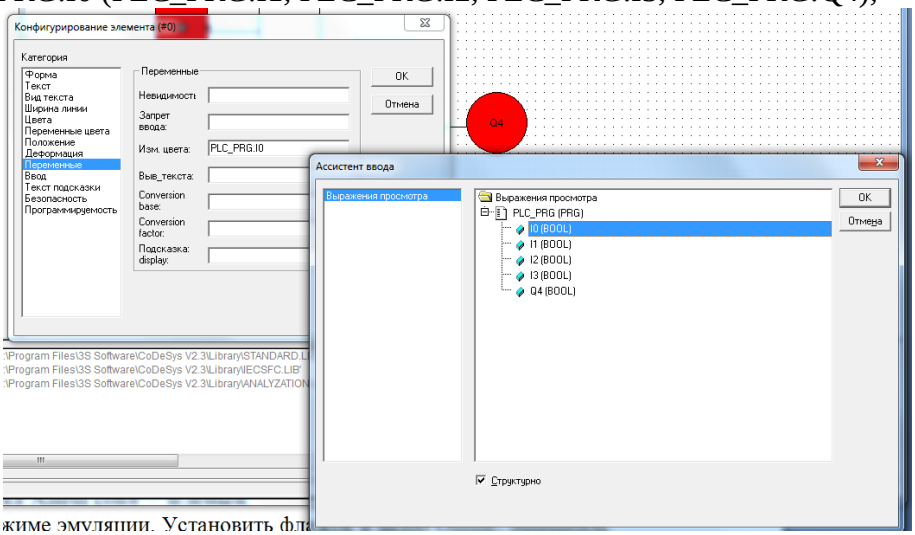
7. Откомпилировать проект. Выполнить команду: Проект, Компилировать или нажать «F11».
8. Исправить ошибки, если таковые имеются.
9. Создать визуализацию проекта. В левой части окна CoDeSys внизу выбрать страницу визуализации (Visualization). На Visualization щелкнуть правой клавишей мыши. В контекстном меню ввода задать команду ***Добавить объект…*** Присвоить новому объекту имя Lab_IL.
10. Нарисовать элементы визуализации в виде прямоугольников для переменных I0, I1, I2, I3, логических элементов AND, OR. Нарисовать элемент визуализации в виде окружности для переменной Q4.



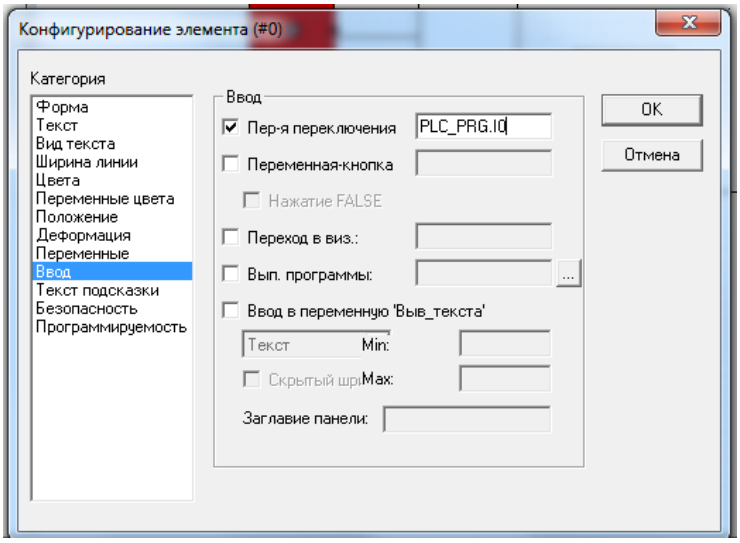
11. Настроить элементы визуализации – переменные I0, I1, I2, I3, Q4. Для них задать следующие настройки:
    - Текст, строка – ввести текст I0 (I1, I2, I3, Q4);



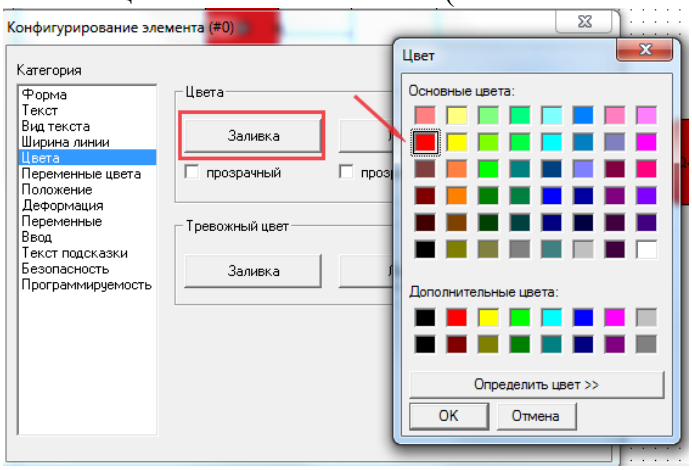
    - Переменные, Изм.цвета – выбрать переменную из списка, нажав кнопку F2, например PLC_PRG.I0 (PLC_PRG.I1, PLC_PRG.I2, PLC_PRG.I3, PLC_PRG.Q4);



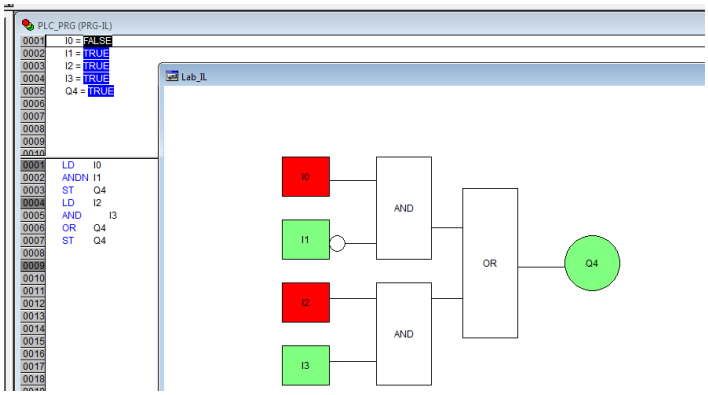
    - Ввод, флажок Переменная переключения включить, выбрать переменную из списка, нажав кнопку F2, например PLC_PRG.I0 (PLC_PRG.I1, PLC_PRG.I2, PLC_PRG.I3);



- Цвета, верхняя панель Цвета – заливка красным (выключенное состояние переменной), нижняя панель Тревожный цвет – заливка зеленым (включенное состояние переменной).



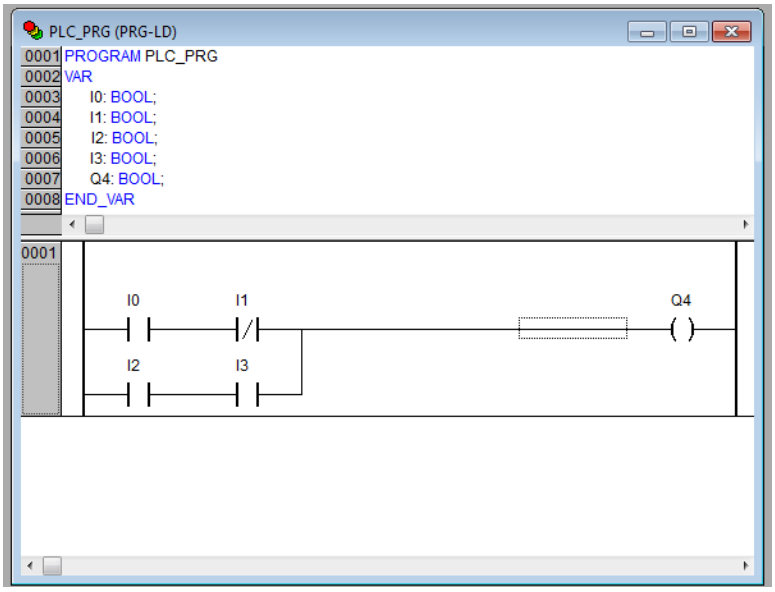
12. Перейти в режиме эмуляции. Установить флажок в меню Онлайн, Режим эмуляции.
13. Запустить проект. Выполнить команду Онлайн, Подключение. Выполнить команду Онлайн, Пуск (или кнопкой F5).
14. Изменяя значения входных переменных, нажимая на элементы визуализации, наблюдать за изменением выходной переменной. Зафиксировать результаты вычислительного эксперимента в виде таблицы, где отобразить состояние входов и состояние выхода.
15. Остановить проект. Выполнить команду Онлайн, Стоп.



2.2. Исследование языка LD

1. Создать новый проект. Выполнить команду: Файл, Создать.
2. Выбрать целевую платформу (Target Settings) 3S CoDeSys SP PLCWinNT V2.4.
3. Определить тип первого программного компонента (New POU) – PLC_PRG. Выбрать язык программирования – LD и тип компонента – программа. В однозадачных проектах система исполнения циклически вызывает программу PLC_PRG.

4. Объявить переменные I0, I1, I2, I3, Q4. В диалоге определения переменных определить тип переменных – BOOL и класс переменных – VAR GLOBAL.
5. В графическом редакторе нарисовать схему из параллельных и последовательных контактов I0, I1, I2, I3 и обмотки Q4 согласно своему варианту.

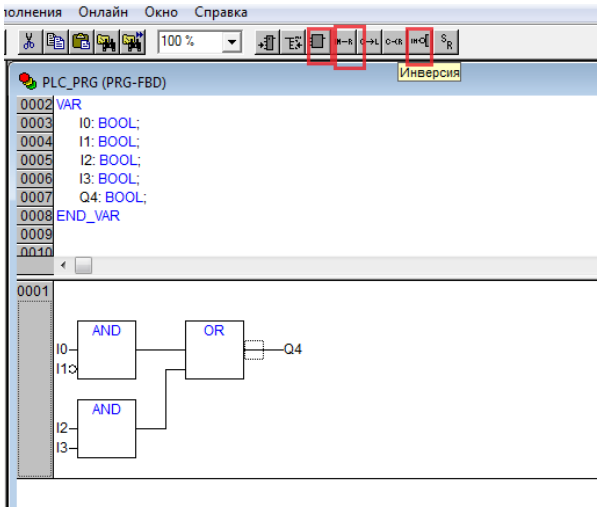


6. Откомпилировать проект. Выполнить команду: Проект, Компилировать или нажать «F11».
7. Исправить ошибки, если таковые имеются.
8. Создать визуализацию проекта. В левой части окна CoDeSys внизу выбрать страницу визуализации (Visualization). На Visualization щелкнуть правой клавишей мыши. В контекстном меню ввода задать команду ***Добавить объект…*** Присвоить новому объекту имя Lab_IL.
9. Нарисовать элементы визуализации в виде прямоугольников для переменных I0, I1, I2, I3, логических элементов AND, OR. Нарисовать элемент визуализации в виде окружности для переменной Q4.
10. Настроить элементы визуализации – переменные I0, I1, I2, I3, Q4 – аналогично п.п. 11 предыдущего задания.
11. Перейти в режиме эмуляции. Установить флажок в меню Онлайн, Режим эмуляции.
12. Запустить проект. Выполнить команду Онлайн, Подключение. Выполнить команду Онлайн, Пуск (или кнопкой F5).
13. Изменяя значения входных переменных, нажимая на элементы визуализации, наблюдать за изменением выходной переменной. Убедиться, что реализованная схема работает также, как в предыдущем задании.
14. Остановить проект. Выполнить команду Онлайн, Стоп.

2.3. Исследование языка FBD

1. Создать новый проект. Выполнить команду: Файл, Создать.
2. Выбрать целевую платформу (Target Settings) 3S CoDeSys SP PLCWinNT V2.4.
3. Определить тип первого программного компонента (New POU) – PLC_PRG. Выбрать язык программирования – FBD и тип компонента – программа. В однозадачных проектах система исполнения циклически вызывает программу PLC_PRG.
4. Объявить переменные I0, I1, I2, I3, Q4. В диалоге определения переменных определить тип переменных – BOOL и класс переменных – VAR GLOBAL.

5. В графическом редакторе нарисовать схему согласно своему варианту, используя команды Элемент, Присваивание. При этом переменные и команды блоков присваивать из списка (кнопка F2).



6. Откомпилировать проект. Выполнить команду: Проект, Компилировать или нажать «F11».
7. Исправить ошибки, если таковые имеются.
8. Создать визуализацию проекта. В левой части окна CoDeSys внизу выбрать страницу визуализации (Visualization). На Visualization щелкнуть правой клавишей мыши. В контекстном меню ввода задать команду ***Добавить объект…*** Присвоить новому объекту имя Lab_IL.
9. Нарисовать элементы визуализации в виде прямоугольников для переменных I0, I1, I2, I3, логических элементов AND, OR. Нарисовать элемент визуализации в виде окружности для переменной Q4.
10. Настроить элементы визуализации – переменные I0, I1, I2, I3, Q4 – аналогично п.п. 11 предыдущего задания.
11. Перейти в режиме эмуляции. Установить флажок в меню Онлайн, Режим эмуляции.
12. Запустить проект. Выполнить команду Онлайн, Подключение. Выполнить команду Онлайн, Пуск (или кнопкой F5).
13. Изменяя значения входных переменных, нажимая на элементы визуализации, наблюдать за изменением выходной переменной. Убедиться, что реализованная схема работает также, как в предыдущих заданиях.
14. Остановить проект. Выполнить команду Онлайн, Стоп.

3. Варианты индивидуальных заданий

1. Q0 = I0 & I1 V I2 & I3
2. Q1 = I4 & (I4 V I6) & I7
3. Q2 = I0 V I1 & I2 & I3
4. Q3 = I5 V I4 V I6 & I7
5. Q4 = I0 & I1 V I2 V I3
6. Q5 = I4 & I5 & I6 V I7
7. Q6 = I0 & I1 & I2 & I3
8. Q7 = I4 V I1 V I2 V I3
9. Q0 = I4 & I5 V I2 & I3
10. Q1 = I0 & (I1 V I2) & I3
11. Q2 = I5 & I6 V I2 & I3

12. Q3 = I0 & I1 & I2 & I3
13. Q4 = I0 & I1 & I4 V I5
14. Q5 = I0 & I1 V I2 & I5
15. Q6 = I0 & I1 V I2 V I3

& - логическое «И»
V – логическое «ИЛИ»

4. Содержание практической работы и требования к отчету

В результате выполнения практической работы должны быть разработаны:

- главная программа PLC_PRG на трех языках программирования (Список Инструкций (IL), Диаграммы Функциональных блоков (FBD) и Релейно-контактные схемы (LD));
- визуализация проекта для каждого языка программирования;
- отчет.

Отчет должен содержать следующие разделы:

- задание;
  - введение, в котором описывается решаемая задача, кратко описываются разделы лабораторной работы;
  - разработка главной программы PLC_PRG на трех языках программирования (Список Инструкций (IL), Диаграммы Функциональных блоков (FBD) и Релейно-контактные схемы (LD));
  - разработка визуализации проекта для каждого языка программирования;
  - результаты вычислительного эксперимента в виде таблицы истинности;
  - выводы.

Практическая работа №2
Примеры создания в CoDeSys систем управления простыми объектами

Цель работы: реализация различных систем управления простыми объектами на языках программирования релейно-контактных схем (LD) и CFC, а также разработка визуализации для данных объектов.

1 УПРАВЛЕНИЕ ОСВЕЩЕНИЕМ В КОМНАТЕ

Цель - свет должен быть выключен, когда в комнате никого нет.

На входе установлены два дискретных датчика: один снаружи комнаты, другой внутри. Когда срабатывает сначала внешний датчик, затем внутренний, это означает, что человек зашел в комнату. Когда срабатывает сначала внутренний датчик, затем внешний, это означает, что человек вышел из комнаты.

Если человек вошел – включить свет, если человек вышел – выключить свет.

Необходимо считать количество людей, заходящих и выходящих из комнаты. Пока в комнате остается хотя бы один человек, свет должен быть включен.

Пример визуализации управления освещением в комнате представлен на рисунке 1.1.

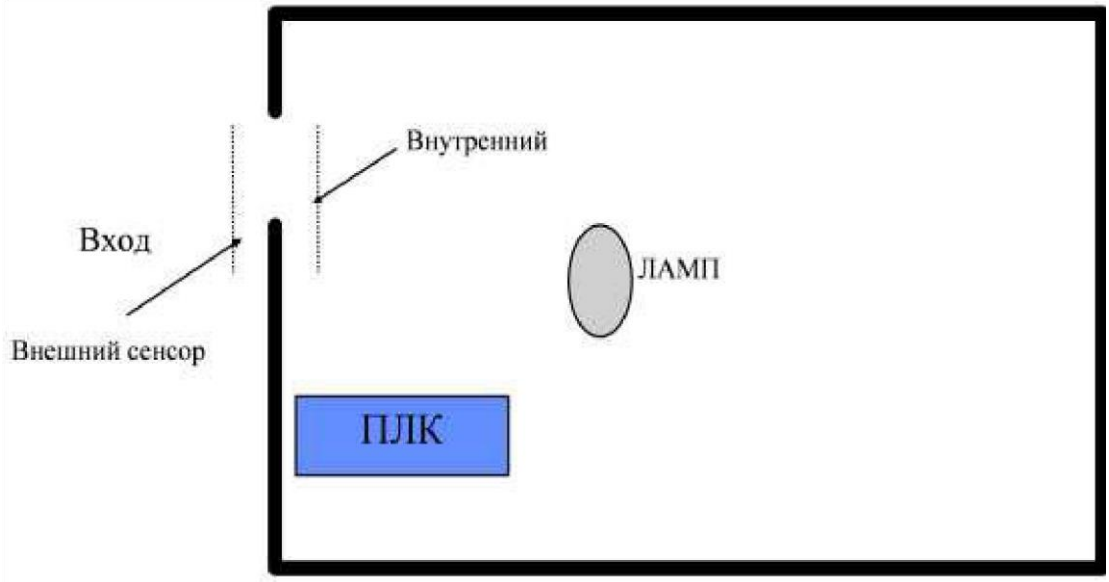


Рисунок 1.1 – Пример визуализации управления освещением в комнате

1.1 Разработка главной программы для управления освещением в комнате на языке программирования релейно-контактных схем (LD) и её визуализации

В соответствии с данной задачей выполним программирование ПЛК для управления освещением в комнате. Для этого объявим локальные переменные, на основании которых разработаем главную программу на языке программирования релейно-контактных схем (LD). Локальные переменные с разработанной главной программой на языке программирования релейно-контактных схем LD для управления освещением в комнате представлены на рисунке 1.2.

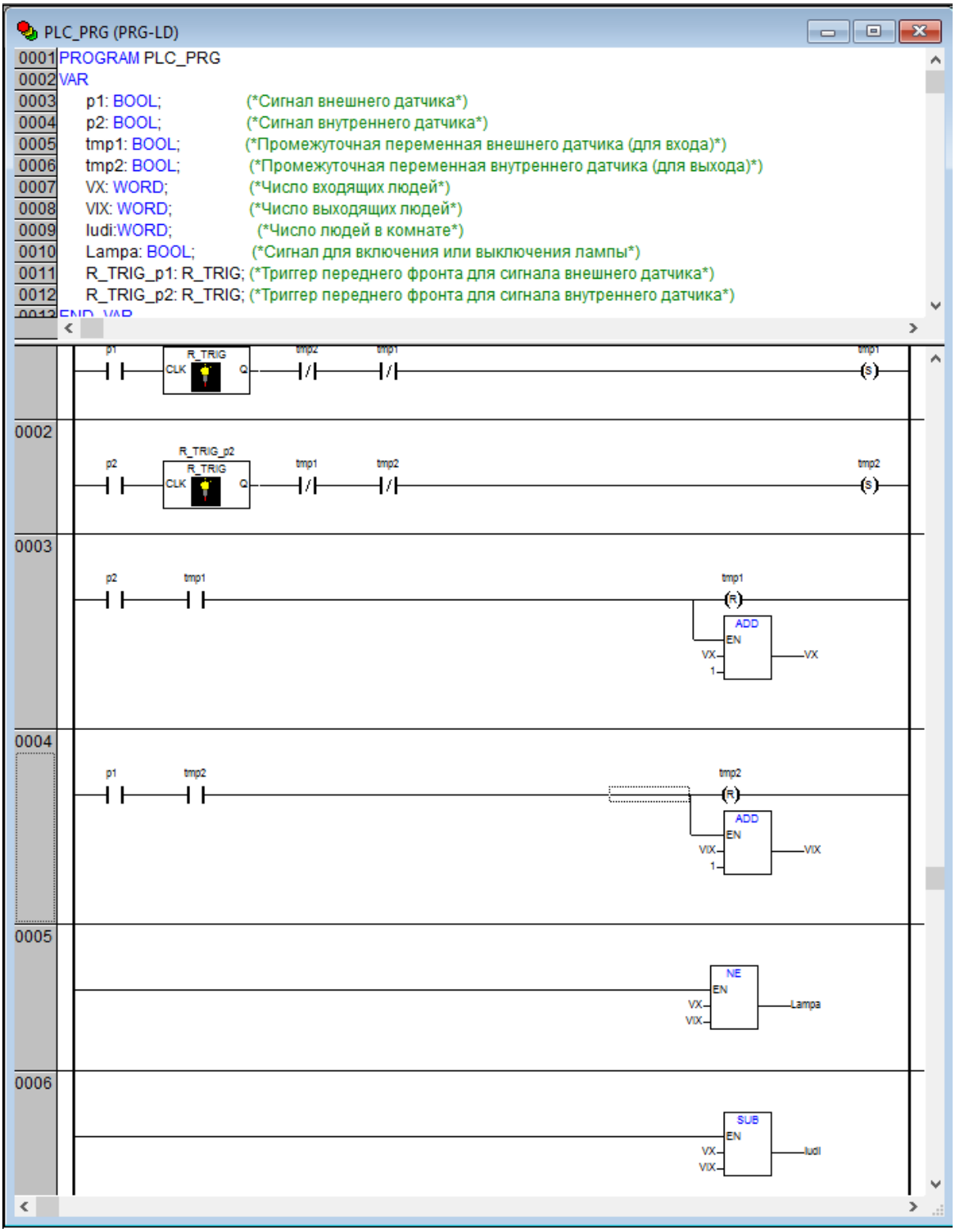


Рисунок 1.2 – Локальные переменные с разработанной главной программой на языке программирования релейно-контактных схем LD для управления освещением в комнате

В приведенной программе применяются промежуточные переменные **(tmp1)** и **(tmp2)** для анализа ситуации, есть ли человек на входе в помещение или на выходе из помещения между датчиками, а также предусмотрено использование триггеров **(R_TRIG_p1)** и **(R_TRIG_p2)** для выделения переднего фронта.

В первой ветке при срабатывании внешнего датчика **(p1)** триггер по переднему фронту **(R_TRIG_p1)** меняет состояние с 0 на 1 и одновременно с этим производится проверка состояния промежуточных переменных, есть ли человек на входе в помещение или выходе из помещения между датчиками. Если нет никаких сигналов, то взводится переменная **(tmp1)**.

В третьей ветке при срабатывании внутреннего датчика **(p2)** происходит сброс промежуточной переменной **(tmp1)** и генерируется инкремент в переменную **(VX)** блока **(ADD)**.

Во второй ветке при срабатывании внутреннего датчика **(p2)** триггер по переднему фронту **(R_TRIG_p2)** меняет состояние с 0 на 1 и одновременно с этим производится проверка состояния промежуточных переменных, есть ли человек на входе в помещение или выходе из помещения между датчиками, если нет никаких сигналов, то взводится переменная **(tmp2)**.

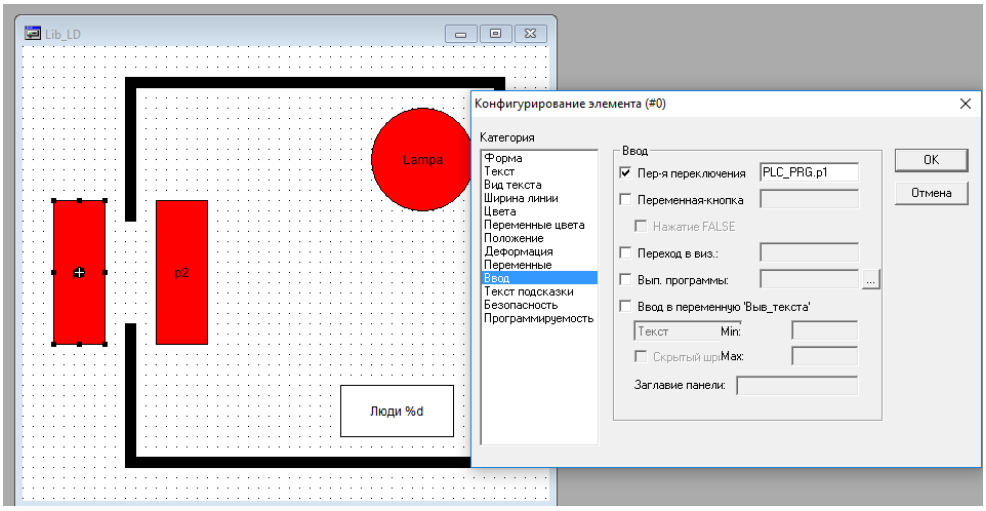
В четвертой ветке при срабатывании внешнего датчика **(p1)** происходит сброс промежуточной переменной **(tmp2)** и генерируется декремент в переменную **(VIX)** блока **(ADD)**.

В пятой ветке вызывается блок **(NE)**, реализующий подсчёт людей по переменным **(VX)** и **(VIX),** приходящих с блоков **(ADD),** и включение лампы в комнате при выходном значении блока 1 или более. Когда переменные **(VX)** и **(VIX)** равны, происходит выключение лампы в комнате.

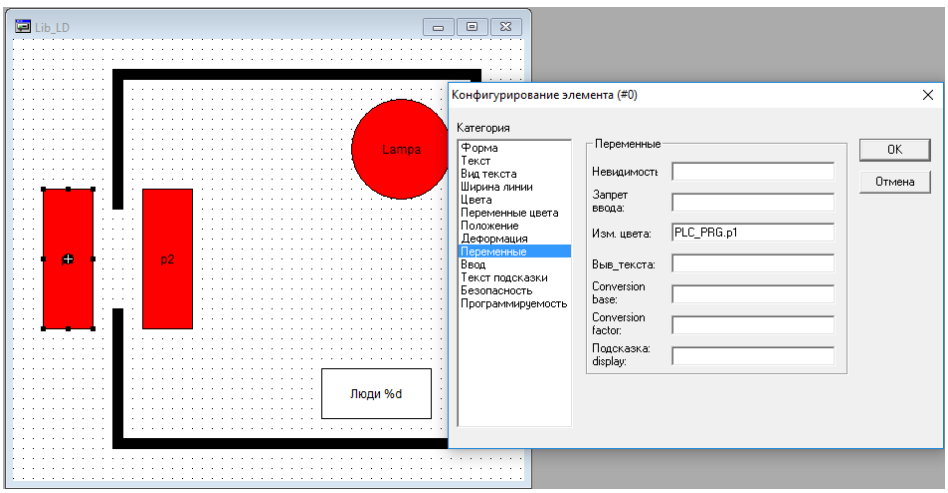
В шестой ветке вызывается блок вычитания **(SUB)** для подсчета количества людей, находящихся в комнате. Количество людей выводится в переменную **(ludi)**.

Таким образом, для решения данной задачи необходимо было считать количество входящих в комнату людей (VX) и выходящих людей (VIX). Когда срабатывает сначала внешний датчик (p1), затем внутренний (p2), это означает, что человек зашел в комнату, и счетчик считает входящих людей (VX).Когда срабатывает сначала внутренний датчик (p2), затем внешний (p1), это означает, что человек вышел из комнаты, и пошел счет выходящих людей (VIX). Затем, сравнив результаты подсчетов, можно сделать вывод: если количество входящих в комнату людей равно количество выходящих, то лампочка гаснет, если не равно (NE), то горит.

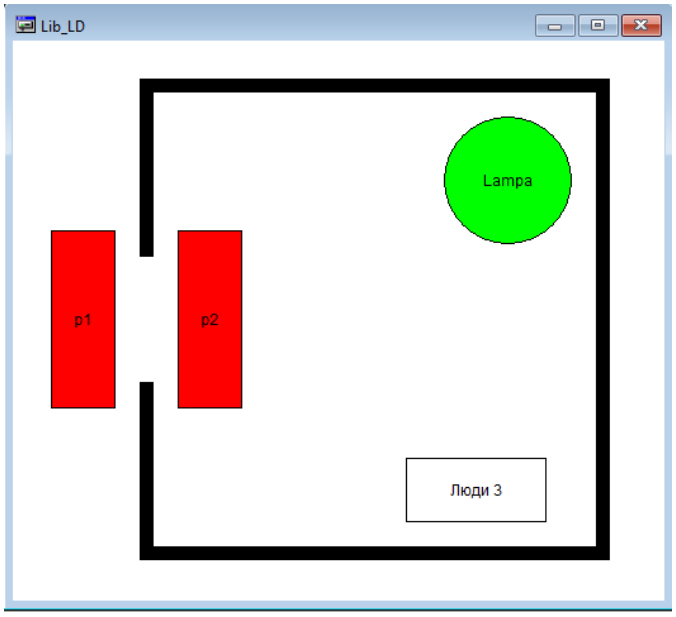
Реализованная визуализация для управления освещением в комнате представлена на рисунках 1.3 и 1.4.



а)



б)



в)

Рисунок 1.3 – Реализованная визуализация для управления освещением в комнате: а) и б) пример визуализации для датчика; в) визуализация при входе человека в комнату (сигналы с датчиков уже отсутствуют)

При изменении состояния датчиков **(p1)** и **(p2)** цвет с красного меняется на зеленый. Если в комнате есть люди, то лампа меняет цвет с красного на зеленый. Счетчик «Люди» показывает количество людей, находящихся в комнате, с учетом числа зашедших и вышедших.

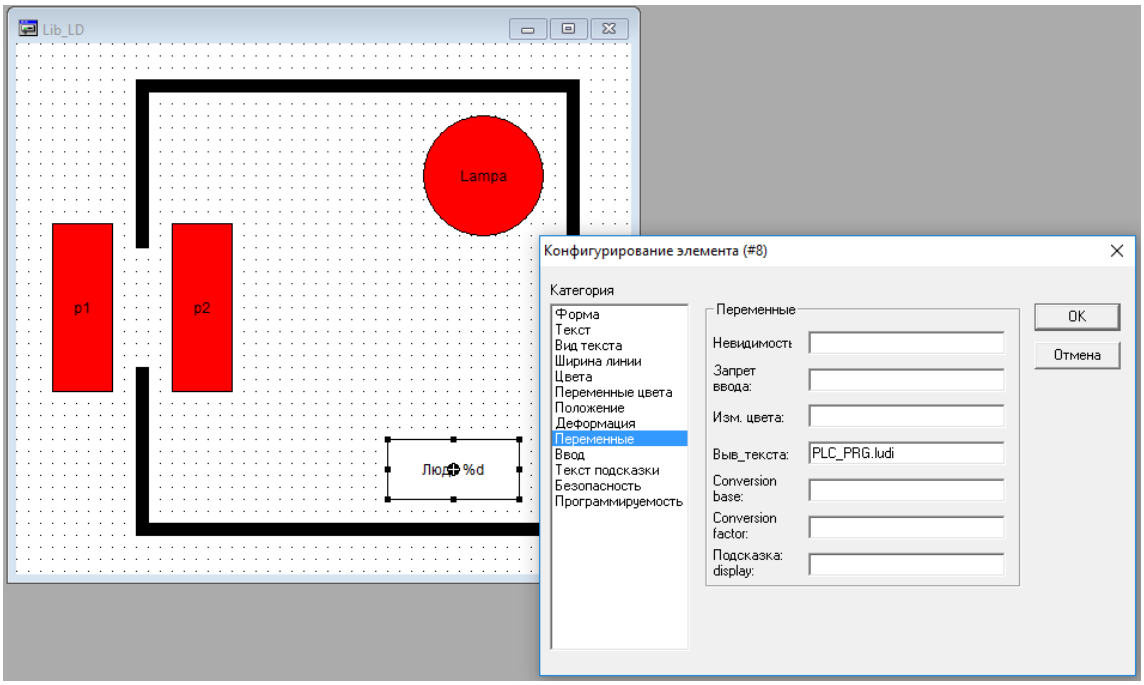


Рисунок 1.4 – Реализация отображения аналогового значения для счетчика «Люди» (%d – означает значение в десятичном коде)

1.2 Разработка главной программы для управления освещением в комнате на языке программирования CFC и её визуализации

В соответствии с данной задачей выполним программирование ПЛК для управления освещением в комнате с использованием конфигурации контроллера. Для этого необходимо объявить глобальные и локальные переменные, на основании которых разработаем главную

программу на языке программирования CFC. Глобальные переменные представлены на рисунке 1.5.

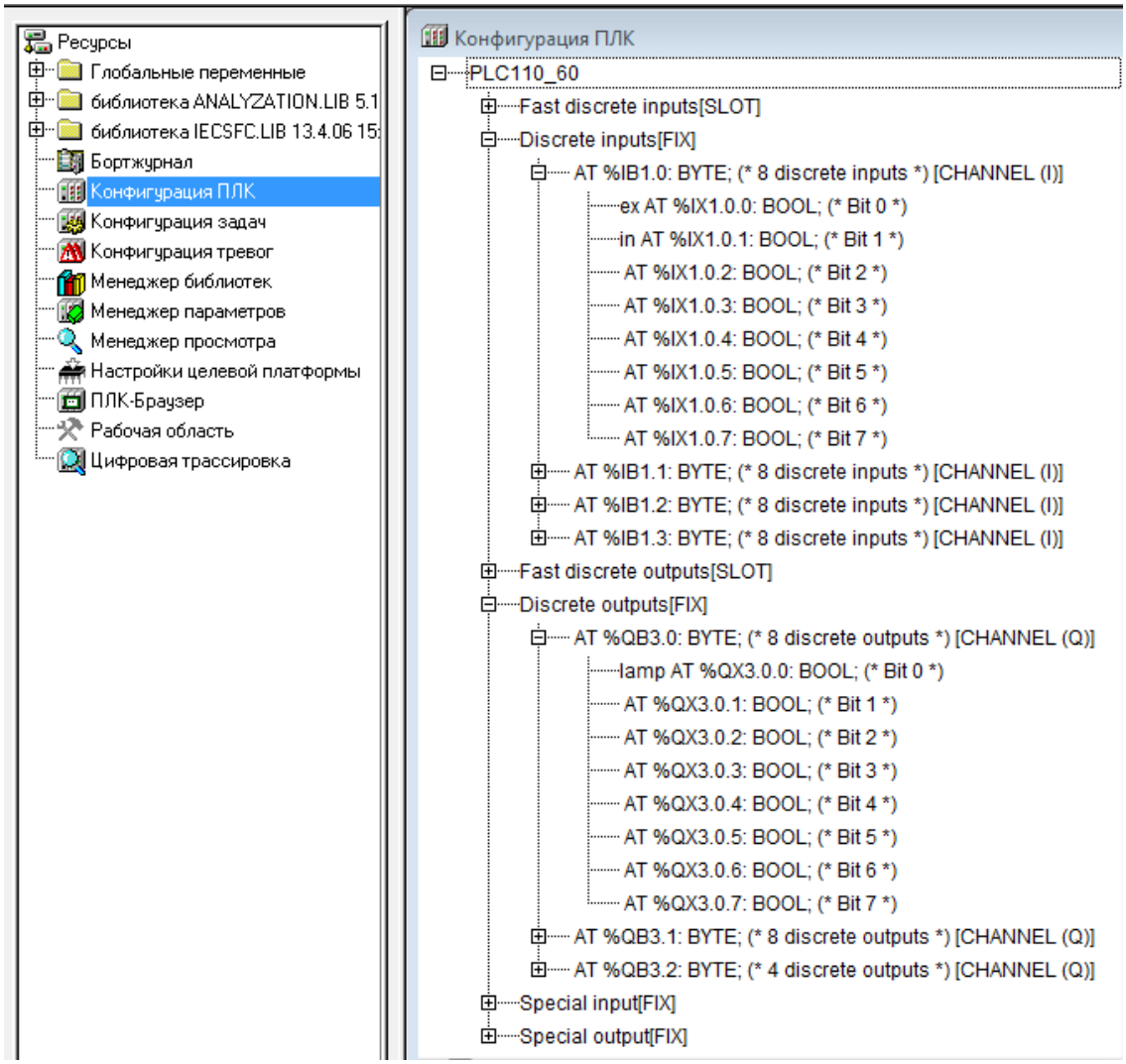


Рисунок 1.5 – Глобальные переменные

Локальные переменные с разработанной главной программой на языке программирования CFC для управления освещением в комнате представлены на рисунке 1.6.

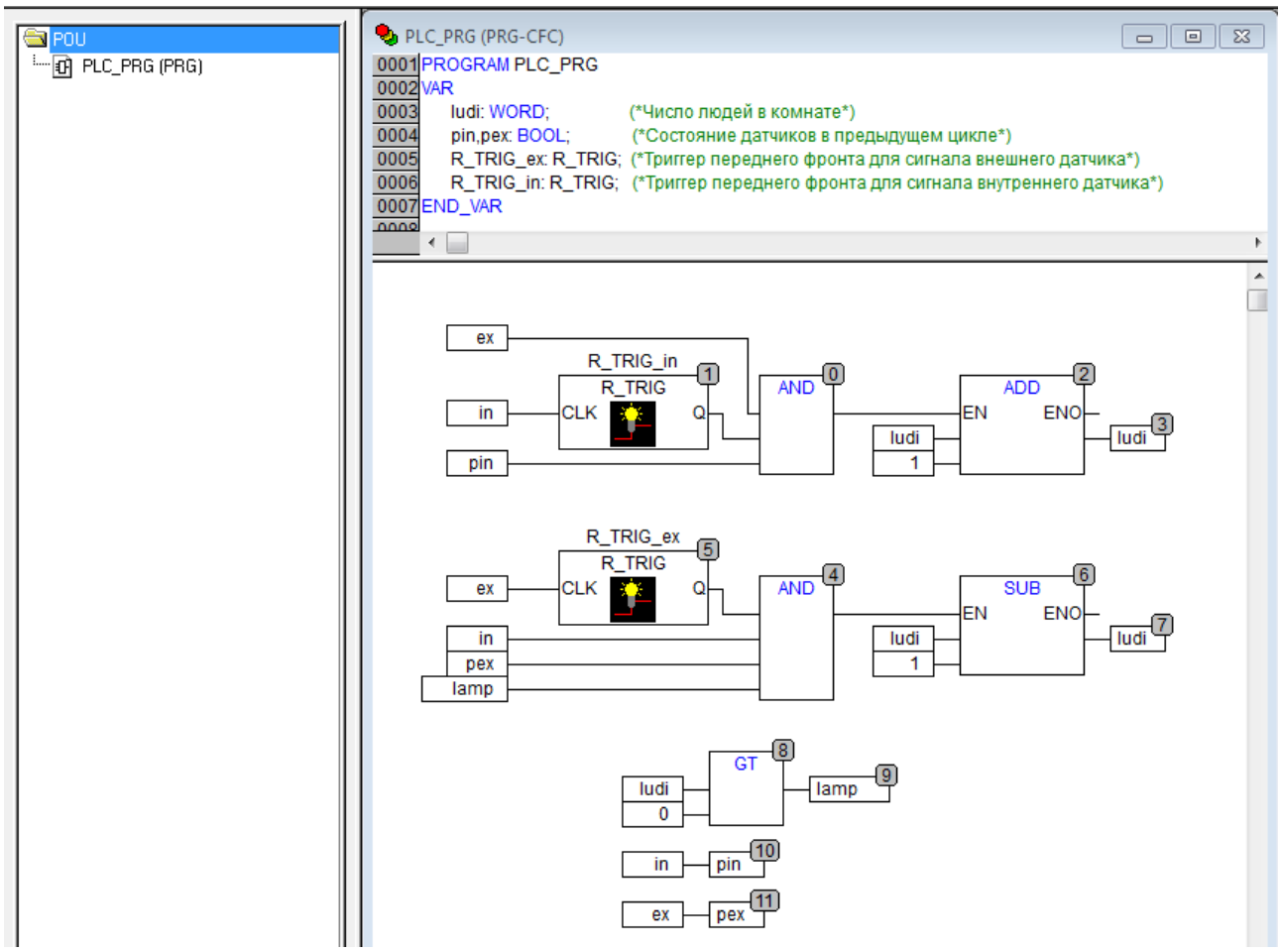


Рисунок 1.6 – Локальные переменные с разработанной главной программой на языке программирования CFC для управления освещением в комнате

В приведенной программе применяются триггеры **(R_TRIG_ex)** и **(R_TRIG_in)** для выделения переднего фронта.

При срабатывании датчиков **(ex)** и **(in)** в последовательности для входа или выхода, происходит проверка состояния датчиков в предыдущем цикле с помощью переменных **(pex)** и **(pin).** После проведения проверки в той или иной ситуации триггеры по переднему фронту **(R_TRIG_ex)** и **(R_TRIG_in)** меняют свое состояние с 0 на 1 и с помощью блоков суммирования или вычитания происходит запись выходного значения в переменную **(ludi)** и хранение данного значения в переменных **(pex)** и **(pin).**

Счётчик реализован с помощью блока **(GT)**, в котором происходит сравнение выходного значения с блоков суммирования и вычитания с нулевой константой, вследствие чего если выходное значение больше нуля, то в комнате загорается лампа, если равно нулю, то лампа в комнате выключается.

ВНИМАНИЕ!!! Порядок выполнения блоков в главной программе должен быть таким же, как на рис.1.6. Изменить порядок выполнения можно через контекстное меню блока, команда «Порядок».

Визуализацию можно организовать аналогично рис. 1.3.

1.3 Разработка главной программы для управления освещением в комнате на языке программирования CFC с использованием таймера и её визуализации

Необходимо реализовать систему управления освещением на языке программирования CFC с использованием таймера, предусмотрев, что если вышел последний человек – выключить свет (**lamp**) с задержкой 5 сек.

Глобальные переменные представлены на рисунке 1.5.

Локальные переменные с разработанной главной программой на языке программирования CFC с использованием таймера для управления освещением в комнате представлены на рисунке 1.7.

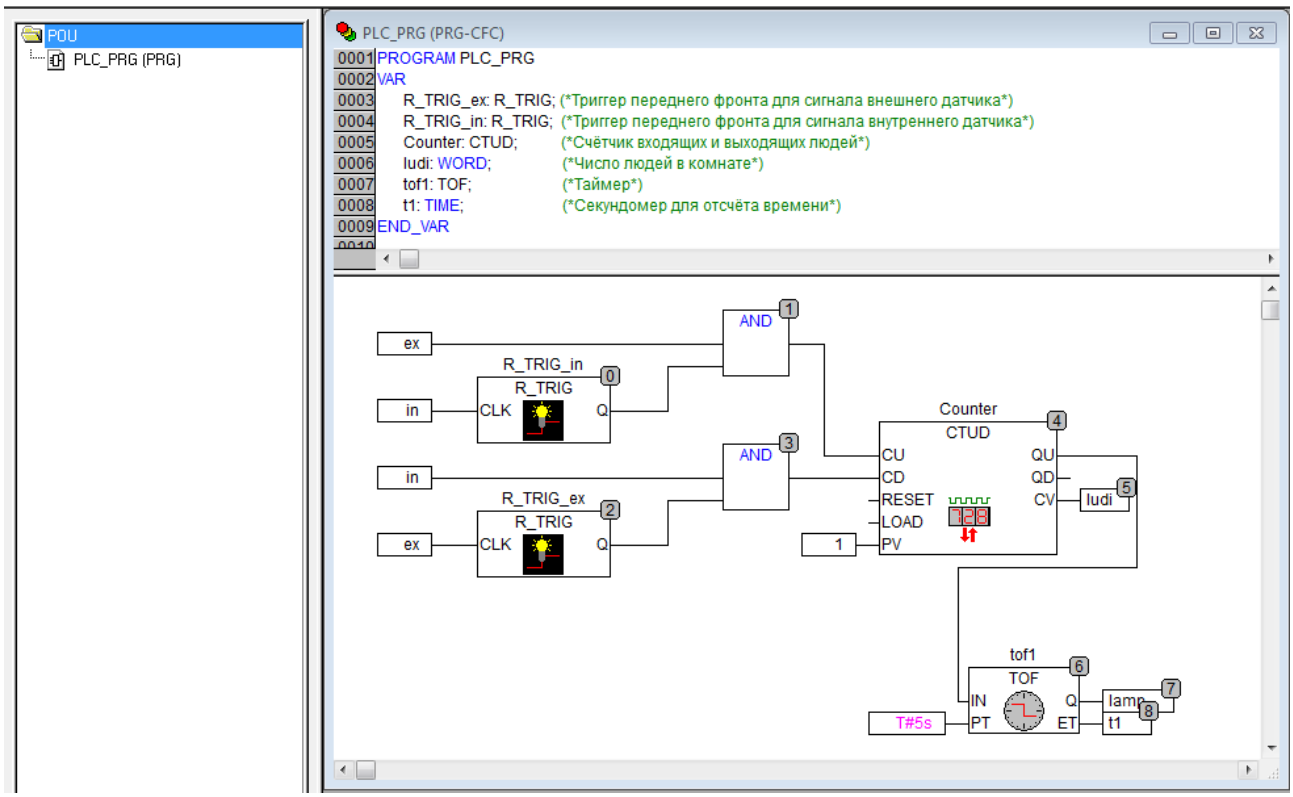


Рисунок 1.7 – Локальные переменные с разработанной главной программой на языке программирования CFC с использованием таймера для управления освещением в комнате

При срабатывании датчиков **(ex)** и **(in)** в последовательности для входа или выхода, триггеры по переднему фронту **(R_TRIG_ex)** и **(R_TRIG_in)** меняют свое состояние с 0 на 1 и сигнал через блоки **(AND)** поступает на счётчик, при этом одновременно происходит запись выходного значения в переменную **(ludi)**. Счётчик реализован с помощью функционального блока **(CTUD),** в котором происходит сравнение выходных значений с блоков **(AND)**, вследствие чего если выходное значение счётчика больше нуля, то сигнал поступает на таймер и в комнате загорается лампа, если же выходное значение счётчика равно нулю, то лампа выключается в комнате через 5 секунд.

Визуализацию можно организовать аналогично рис. 1.3.

2 УПРАВЛЕНИЕ КОТЛОМ

Необходимо реализовать с использованием языка программирования CFC:

- Включение котла с кнопки (pusk), при условии отсутствия аварий.
- Отключение котла (kotel) при возникновении любой из аварий.
- Включение сигнализации (lamp) при возникновении любой из аварий (avar или pojar).
- Отключение котла с кнопки (stop).

В конфигурацию контроллера добавим дискретные входные сигналы:

- pojar – имитация датчика, возникновения пожара
- avar – имитация датчика, возникновения нештатной ситуации
- pusk - сигнал нажатия кнопки «ПУСК»

- stop – сигнал нажатия кнопки «СТОП» Добавим также дискретные выходные сигналы:
  - lamp – включение лампы при аварии или пожаре
  - kotel – работа котла.

Конфигурация дискретных сигналов показана на рисунке 2.1.

Реализация управления работы котла выполнена на языке программирования CFC. Включение и отключение котла осуществляется с помощью SR триггера. Идентификатор триггера указан в окне локальных переменных. Программный код работы управления котлом показан на рисунке 2.2.

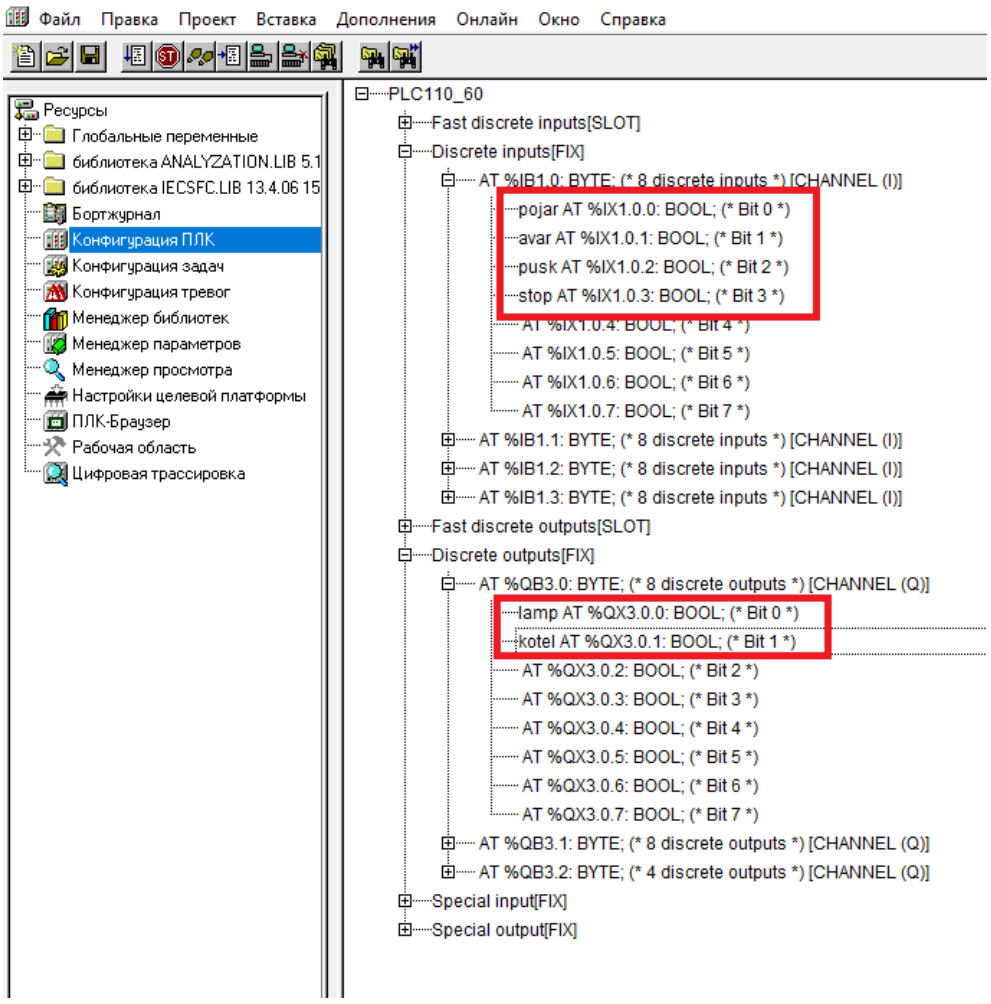


Рисунок 2.1 – Конфигурация параметров для управления котлом

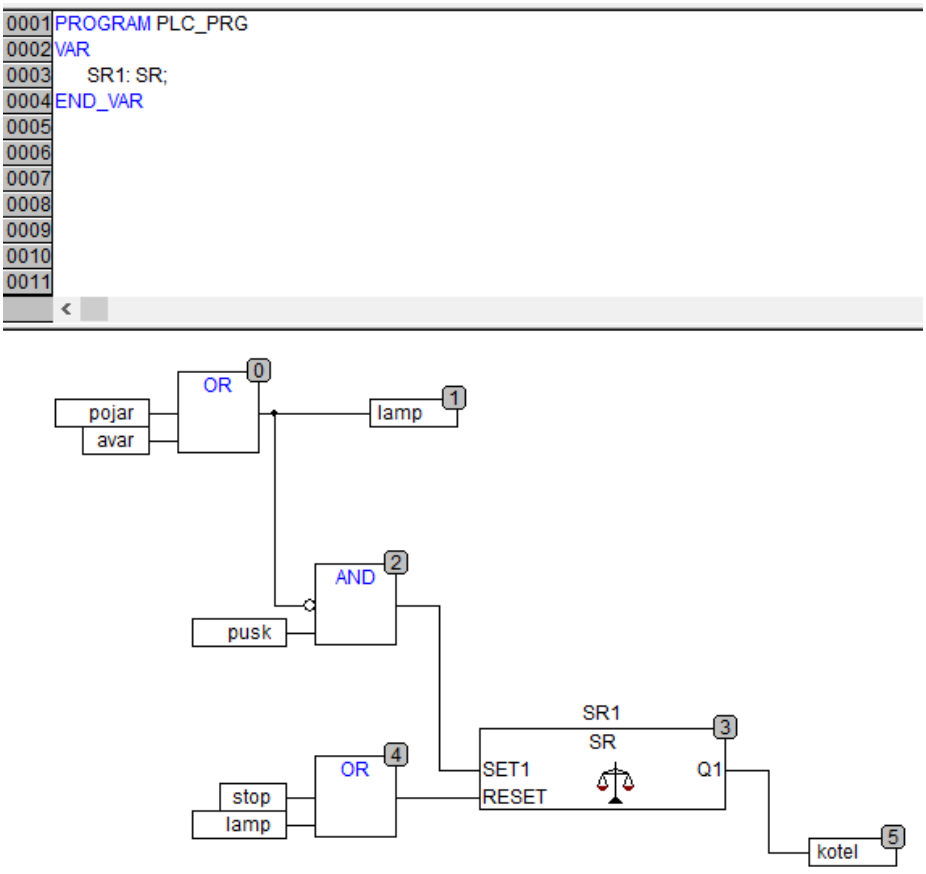


Рисунок 2.2 – Программный код управления котлом на языке CFC

Далее перейдем к визуализации проекта. Создадим проект «kotel». Конфигурация сигналов «avar» и «pojar» будет производиться аналогичным способом, как и в предыдущих работах для дискретных входных сигналов.

Сигналы «pusk» и «stop» являются кнопками, поэтому при нажатии они не должны фиксироваться. Для этого в категории «Ввод» выбирается пункт «переменная-кнопка» с указания тега «.pusk». Аналогичным образом и производится конфигурация кнопки «stop». Изменения цвета производится аналогичным способом, что и в предыдущих работах. Пример представлен на рисунке 2.3.

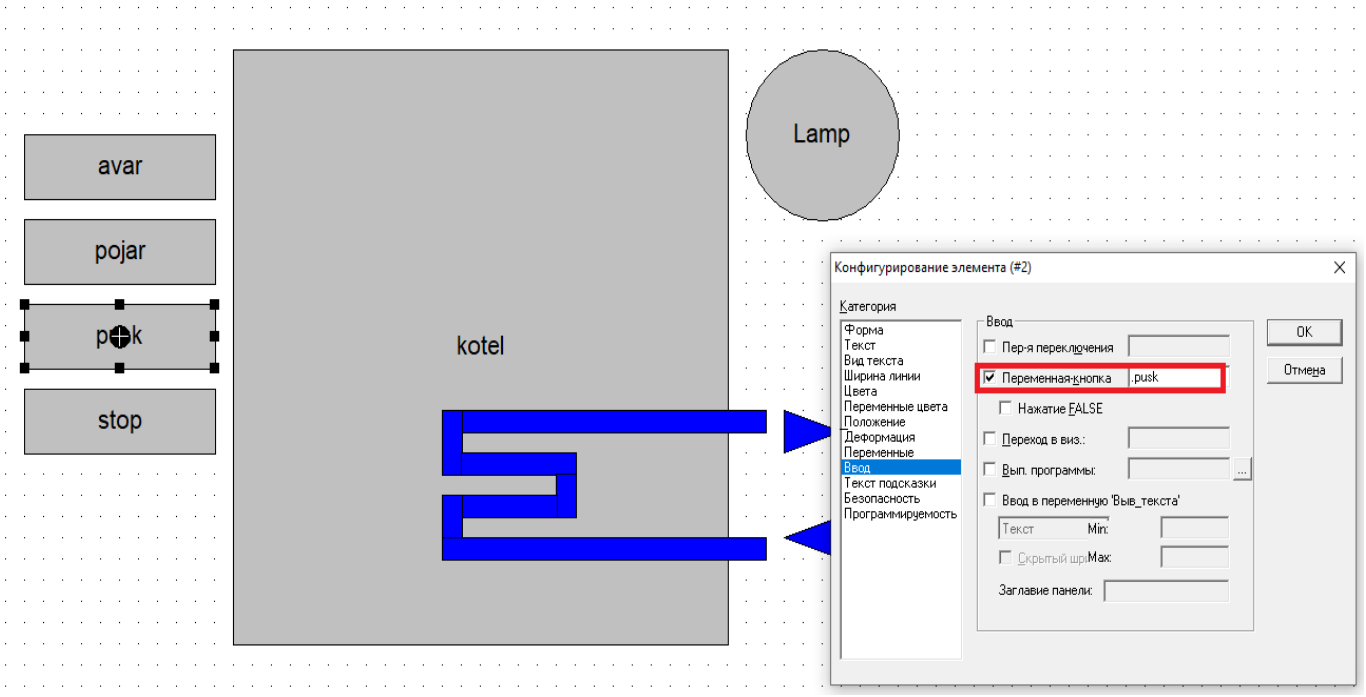
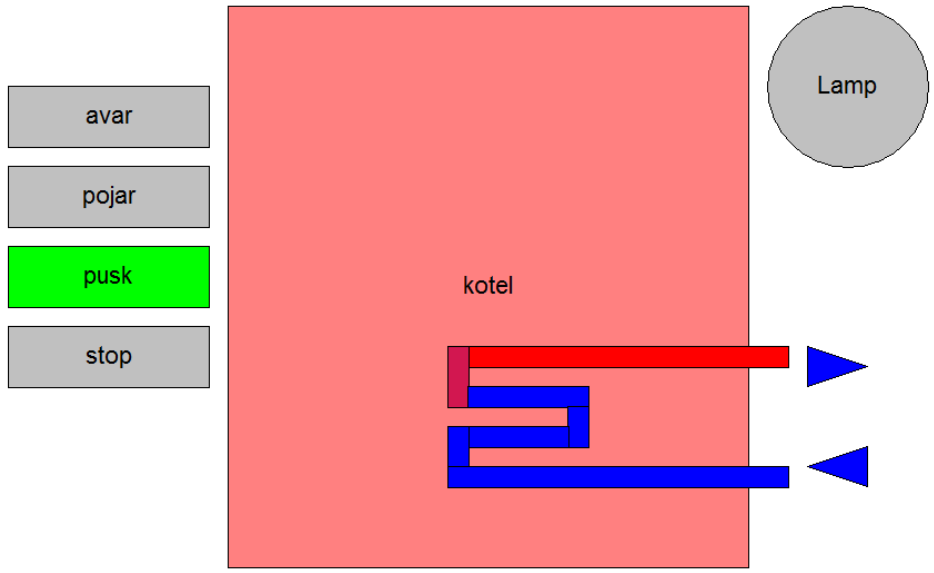
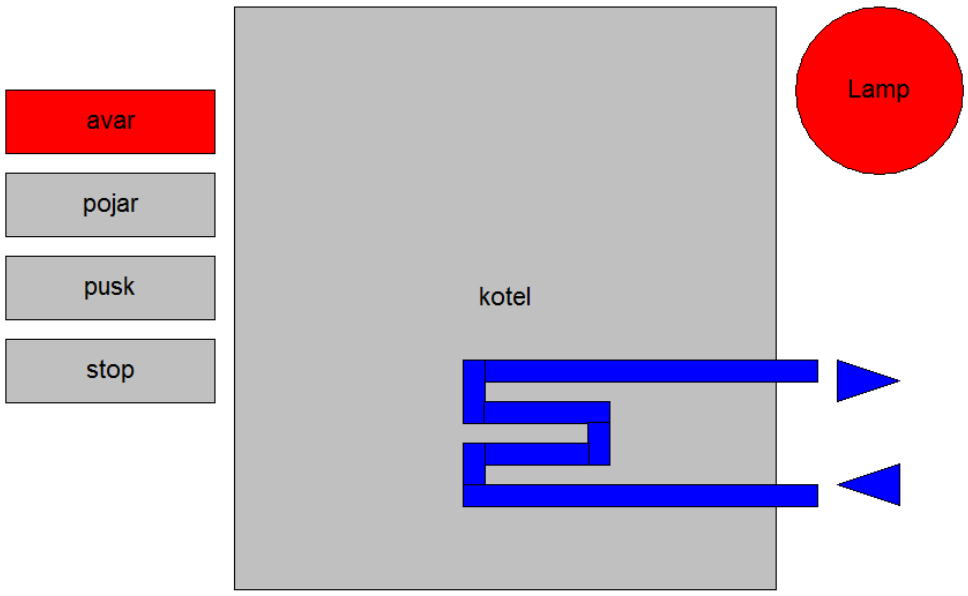


Рисунок 2.3 – Настройка визуализации элемента кнопки пуск

Для наглядности работы котла элементы трубопровода воды представлены на визуализации при включении котла и успешной дальнейшей его работе меняет оттенок на бордовый цвет. Аналогичным образом и меняется рабочее пространство котла. В соответствии с программой при аварийных ситуациях котел не будет работать, после нажатия кнопки «pusk». По нажатию кнопки «stop» котел останавливается. На рисунке 2.3 показана визуализации работа котла.



а)



б)

Рисунок 2.3 – Визуализация работы котла: а) в обычном режиме работы; б) при наличии аварии

3 УПРАВЛЕНИЕ НАСОСОМ

Необходимо реализовать управление включением насоса с задержкой по времени. При нажатии на кнопку **«ПУСК»**, насос должен включиться и проработать 5 секунд, затем автоматически отключиться. Необходимо также подсчитывать количество включений двигателя.

Добавим дискретный входной сигнал:

- pusk – сигнал нажатия кнопки «ПУСК» для включения насоса Дискретный выходной сигнал:
  - nasos – работа насоса

Конфигурация дискретных сигналов показана на рисунке 3.1.

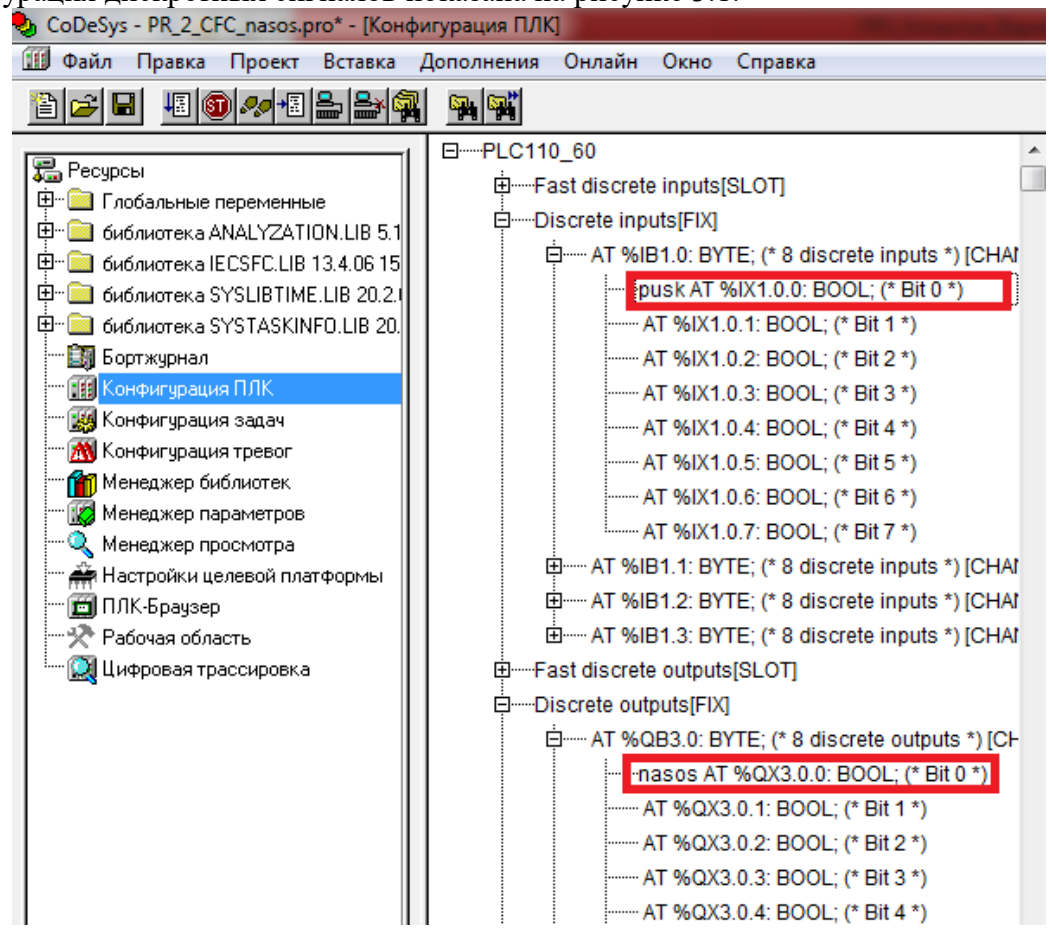


Рисунок 3.1 – Конфигурация дискретных сигналов ПЛК

Программный код реализован на языке CFC. В качестве элементов используются: положительный фронт дискретного сигнала **(r_tr1)** с кнопки «pusk»; счётчик **(CTU1)** для вычисления количества включений насоса и таймер задержки **(tp1)** отключения насоса. Выдержка для отключения насоса составляет 5 секунд.

В окне локальных переменных объявим количество включений насоса переменной типа WORD **(c1)**, а также идентификаторы названий для других элементов.

Полностью программный код на языке CFC представлен на рисунке 3.2.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     r_tr1 :R_TRIG;(*передний фронт*)
0004     CTU1:CTU;(*счётчик*)
0005     tp1:TP;(*Таймер*)
0006     c1: WORD;(*количество включений*)
0007 END_VAR
0008
0009
0010
0011
0012
0013
```

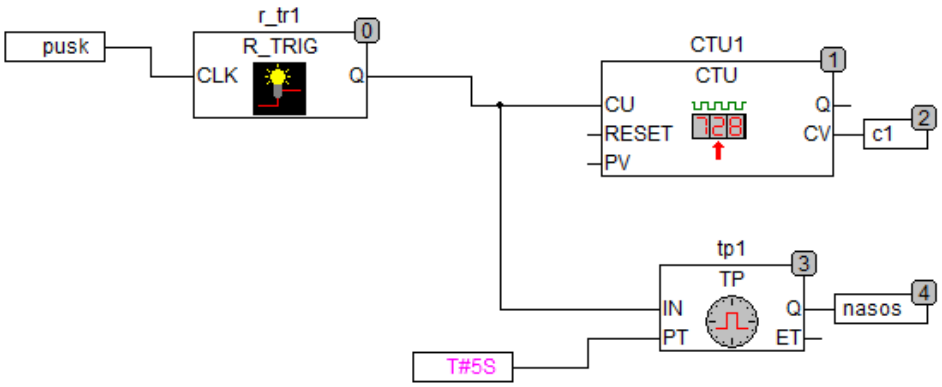
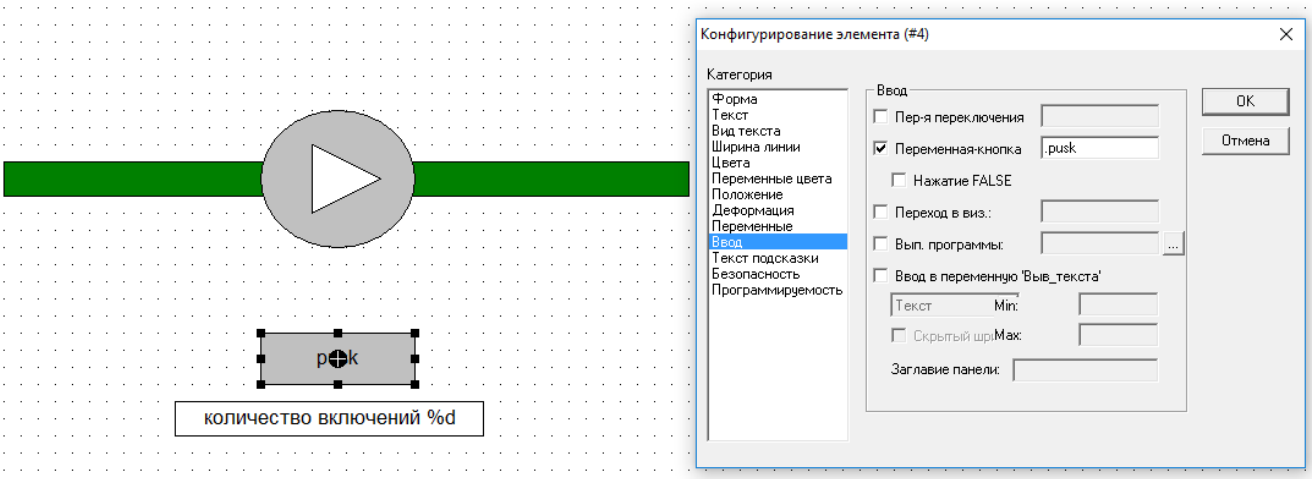


Рисунок 3.2 – Программный код управления включением насоса на языке CFC

Создадим проект визуализации «nasos» для управления насосом. Визуализация работы кнопки «pusk» представлена на рисунке 3.3. Визуализация счётчика количества включений насоса представлена на рисунке 3.4.

Насос будет менять состояние, когда будет формироваться выходное значение «включения насоса» и отключаться по истечения времени, установленном на таймере задержки отключения. Визуализация управления работой насоса в целом представлена на рисунке 3.5.



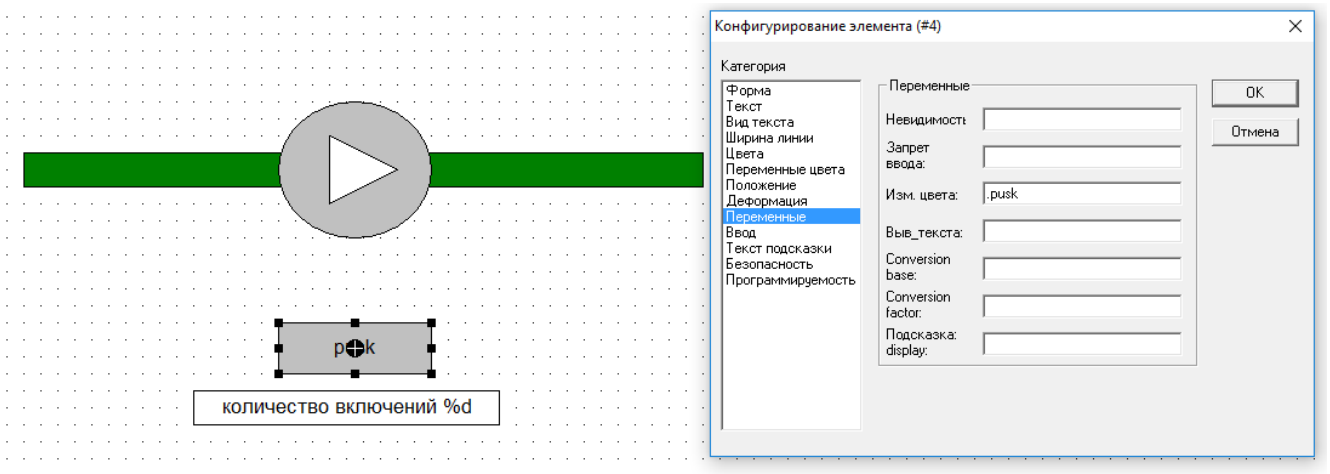


Рисунок 3.3 – Настройка визуализации работы кнопки «pusk»

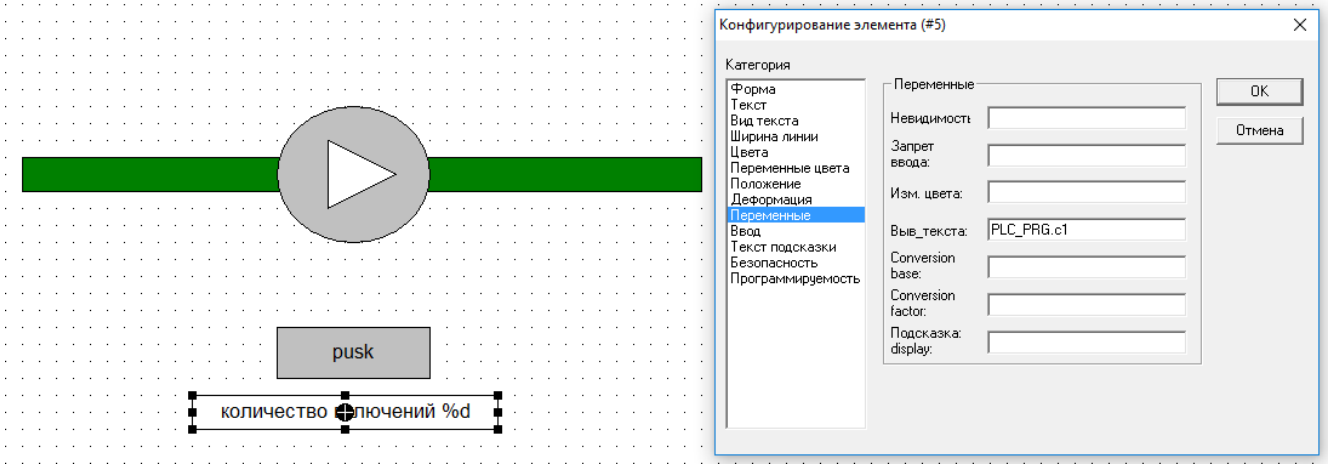


Рисунок 3.4 - Настройка визуализации счётчика количества включений насоса

а)

б)

Рисунок 3.5 – Визуализация управления работой насоса:
а) насос включен; б) насос выключен

4 СИСТЕМА ПОЖАРНОЙ СИГНАЛИЗАЦИИ ЗДАНИЯ

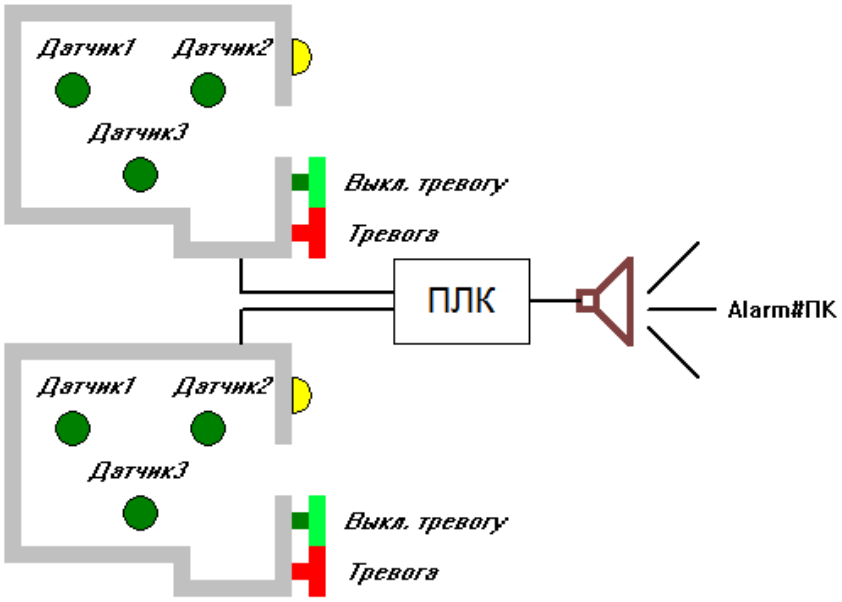


Рисунок 4.1 – Техническое задание

В здании две одинаковые комнаты. В каждой комнате установлено три пожарных датчика (d11, d12, d13 и d21, d22, d23), кнопка ручного включения сигнализации (tr1 и tr2) и кнопка ручного отключения сигнализации (sb1иsb2). Для каждой комнаты предусмотрена сигнальная лампа (lamp1,lamp2). Сигнализация пожара (alarm) является общей для обеих комнат. Если в комнате срабатывает хотя бы один из датчиков, то загорается сигнальная лампа для соответствующей комнаты. Лампа гаснет, если все датчики в комнате отключены. Если в комнате срабатывает любые два из трех датчиков, то включается пожарная сигнализация. Сигнализация работает до тех пор, пока ее не отключат соответствующей кнопкой. Сигнализация может быть включена кнопкой проверки вне зависимости от состояния датчиков.

Добавим дискретные входные сигналы (кнопки включения тревоги и сброса аварии) и выходные дискретные сигналы (лампа перед входом в комнаты и звуковой сигнал общей тревоги). Конфигурации дискретных сигналов контроллера представлены на рисунке 4.2.

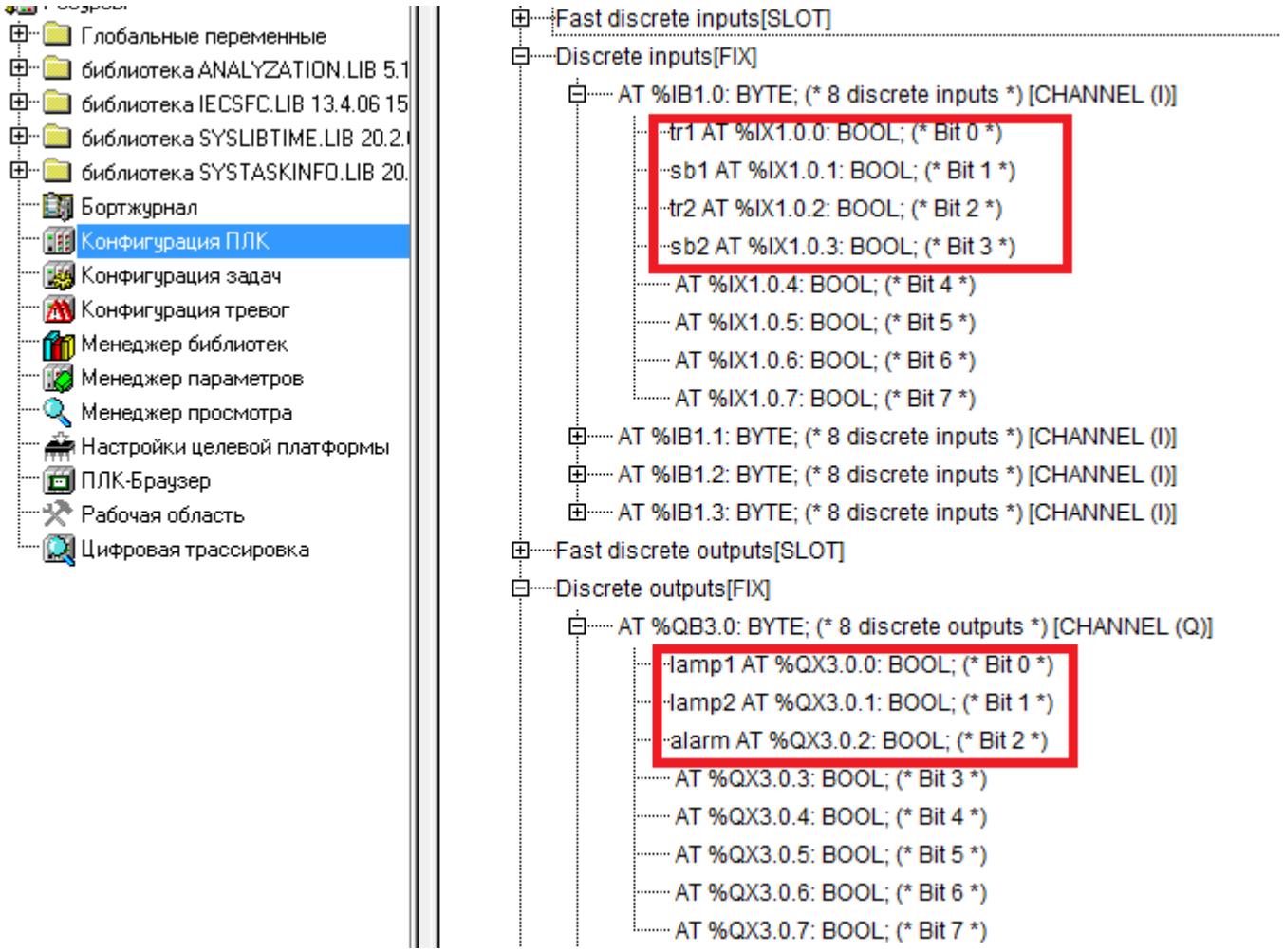


Рисунок 4.2 – Конфигурация дискретных сигналов ПЛК

Далее приступим к написанию программного кода на языке CFC. Так как у нас используется две одинаковых комнаты и работа пожарной сигнализации в этих комнатах предусматривается идентичной, то целесообразно использовать одну функцию управления для двух комнат. Для этого правой кнопкой мыши на позиции «POU» нажимаем кнопку «Добавить объект». В предложенном меню, выбираем функциональный блок, язык программирования CFC (рисунок 4.3).

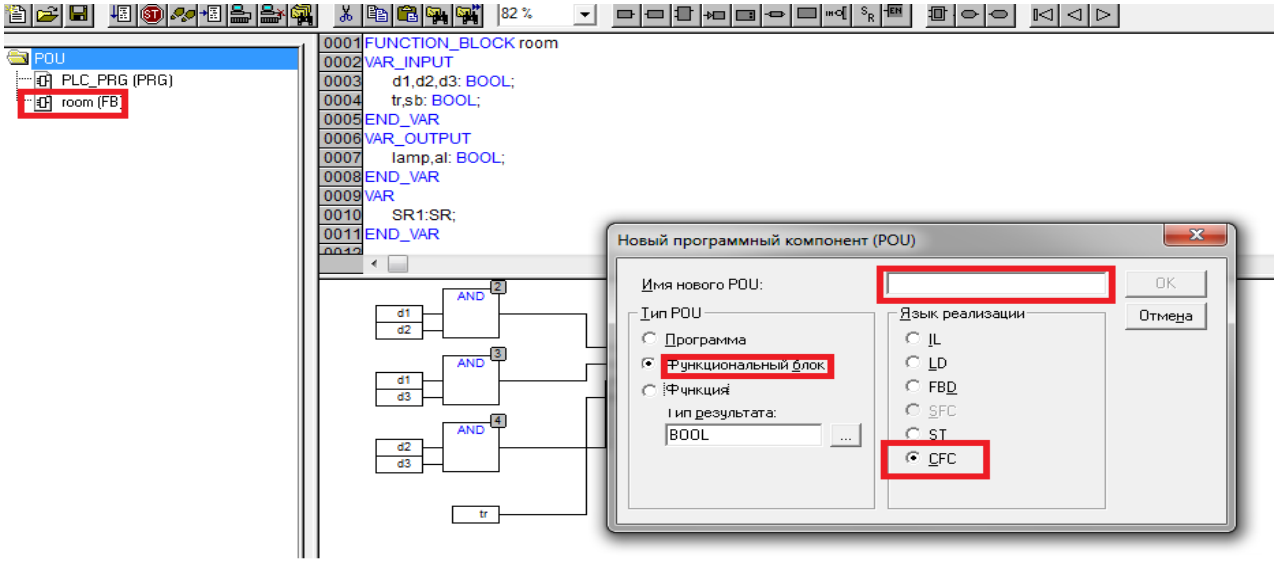


Рисунок 4.3 – Создание функционального блока в среде разработке CoDeSyS

Перед созданием блока в окне локальных переменных добавим сигналы для управления пожарной сигнализацией в комнате. Дискретные входные переменные сигналы d1,d2,d3 – для подключения трех пожарных датчиков. К выходным дискретным сигналам lamp подключаются

лампы на вход в комнату, к al подключается аварийная сигнализация. Реализация работы функционального блока управлением пожарной сигнализации показана на рисунке 4.4.

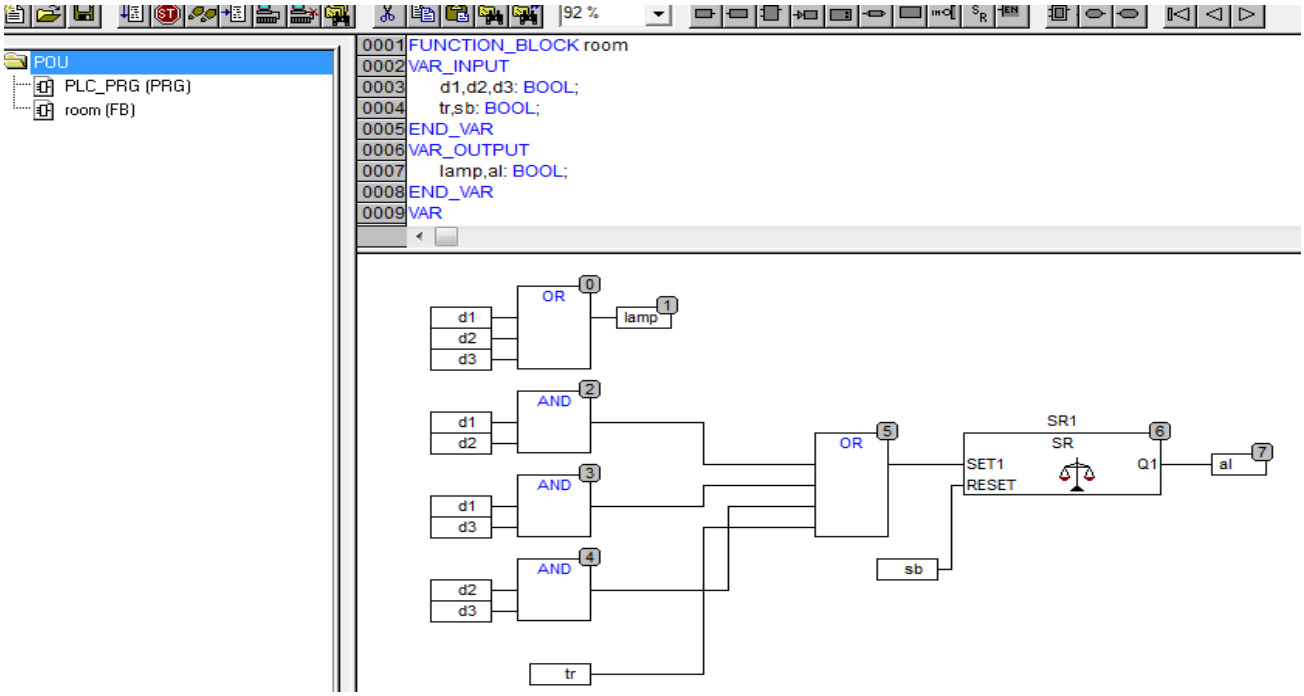


Рисунок 4.4 – Реализация работы функционального блока управлением пожарной сигнализацией

Из программы PLC_PRG вызовем функциональный блок реализованный выше.

Так как у нас две комнаты, соответственно и функциональный блок будет вызываться дважды. В окне локальных переменных добавляем идентификацию для функциональных блоков «room» r1 и r2. Пожарные датчики в комнате также будут локальными переменными d11,d12,d13,d21,d22,d23. Реализация управления системой пожарной сигнализацией представлена на рисунке 4.5.

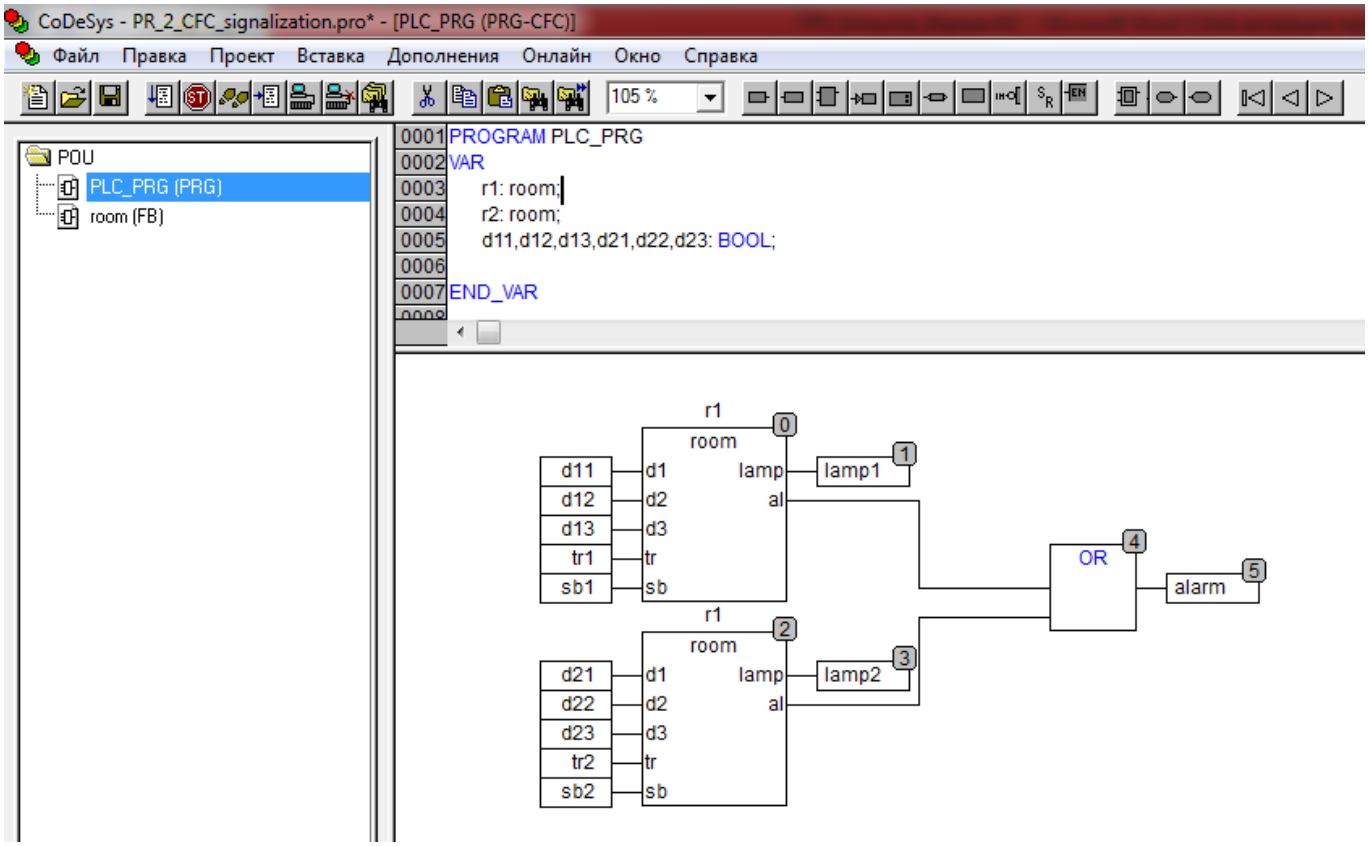


Рисунок 4.5 – Реализация программы управлением пожарной сигнализации

Скрин работы системы визуализации представлен на рисунке 4.5.

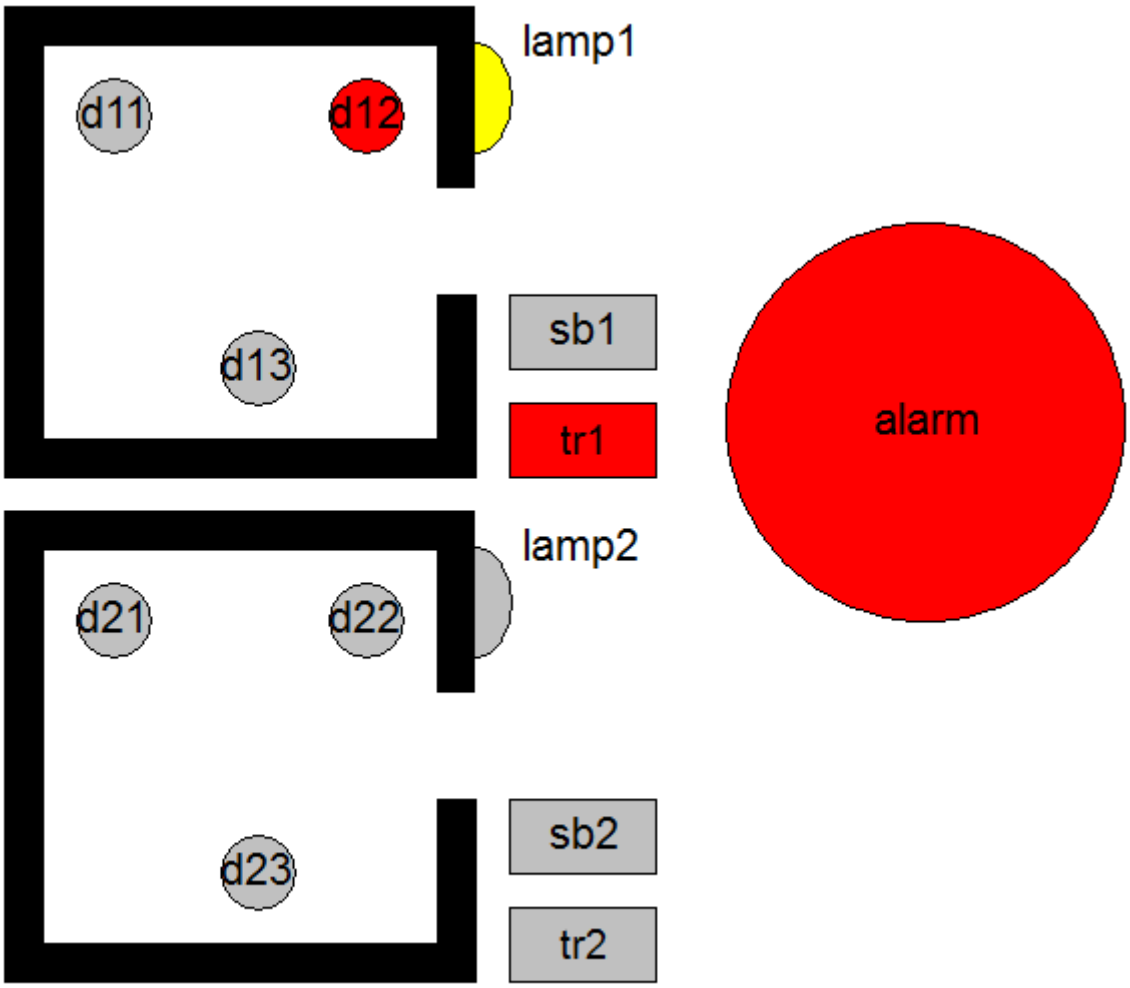


Рисунок 4.5 – Визуализация системы управления пожарной сигнализацией

Практическая работа №3
Система автоматического контроля температуры объекта

Имеется некоторая технологическая установка, в которой необходимо контролировать температуру. Также необходимо отслеживать, чтобы эта температура не превышала некоторого заданного значения (уставки). Если температура поднимается выше уставки и держится на этом уровне больше 3 секунд, то срабатывает сигнализация (например, должна замигать лампа), и запускается в работу некая охладительная установка. Когда температура падает, сигнализация и охладительная установка продолжают работать до тех пор, пока их работа не будет сброшена специальной командой (например, от кнопки «Сброс»). Необходимо предусмотреть возможность изменения уставки в системе.

Порядок выполнения работы

1. Создаём новый проект с целевой платформой для PLC150.I-M (рис. 1). Для главной программы проекта PLC_PRG выбираем язык реализации CFC (рис. 2).

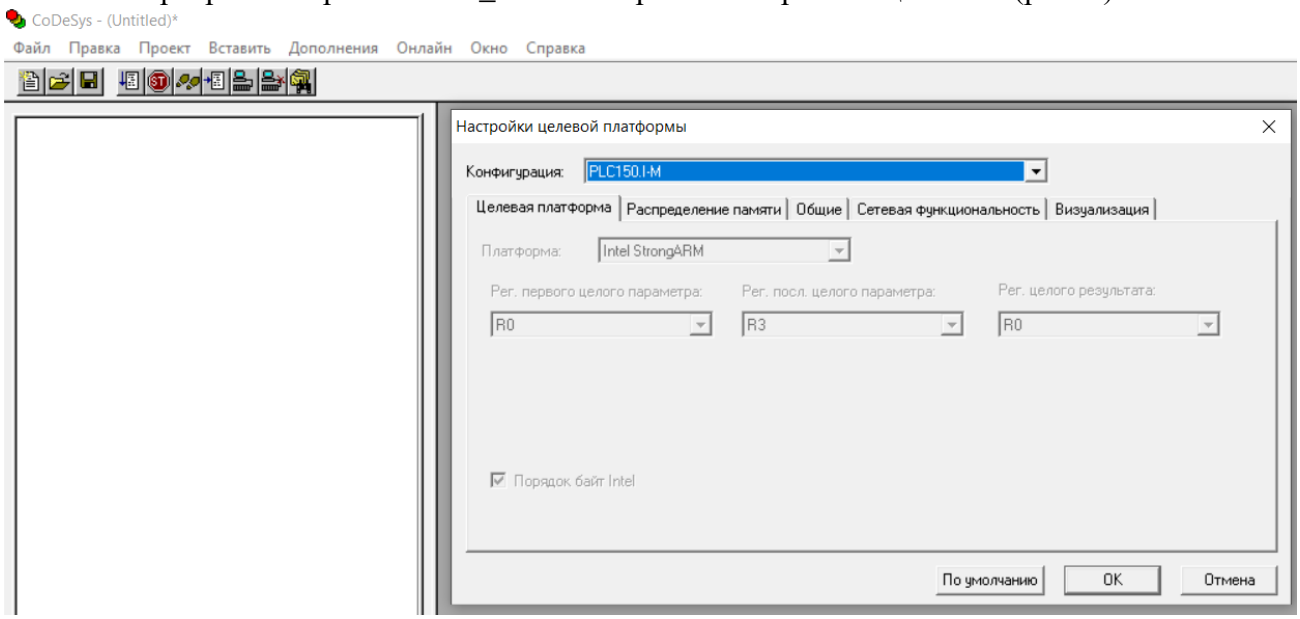


Рис.1

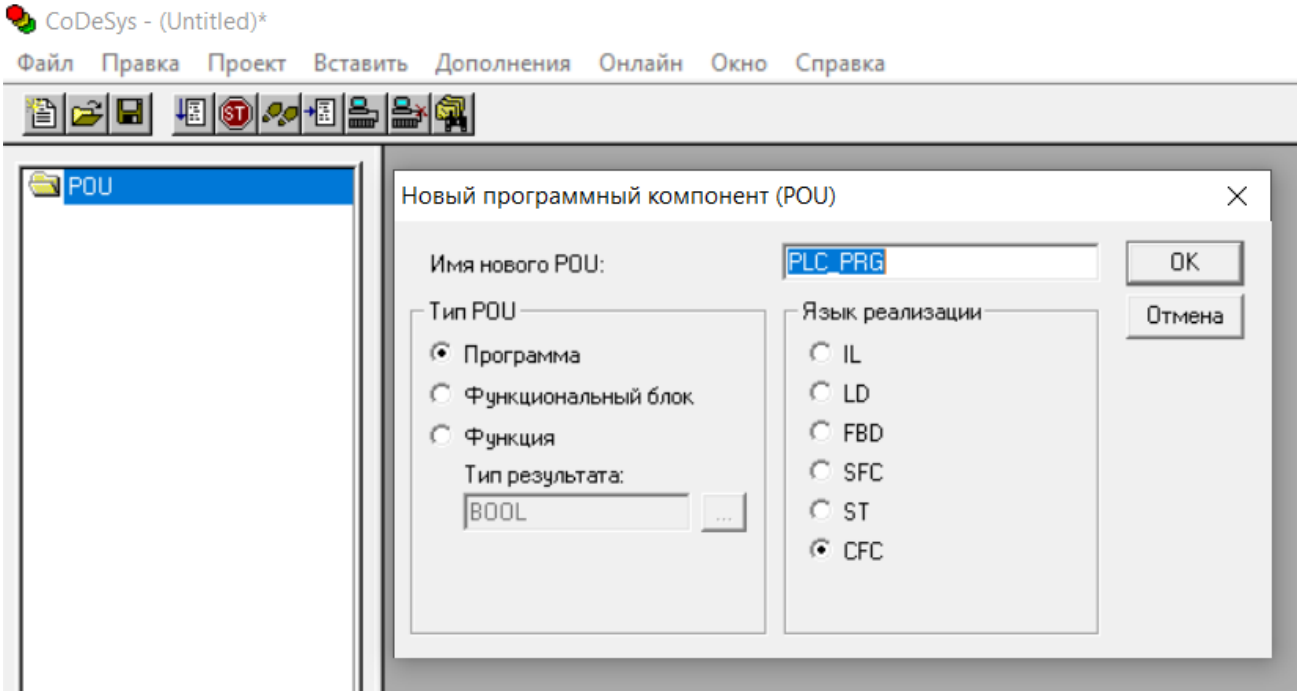


Рис. 2

2. Во вкладке проекта «Ресурсы» выбираем раздел «Конфигурация ПЛК» и устанавливаем сначала во вкладке «Параметры модуля» время цикла 10 ms (рис. 3). Затем устанавливаем в конфигурации один дискретный вход для сброса сигнализации (переменная «sbr») и два

дискретных выхода для включения лампы сигнализации (переменная «lamp») и для включения системы охлаждения (переменная «ohl») (рис. 4). Далее настраиваем аналоговый вход на измерение температуры. Для этого выделяем первый аналоговый вход и в контекстном меню через правую кнопку выбираем пункт «Заменить элемент», а далее выбираем тип элемента «RTD sensor» (рис. 5). Предполагаем, что работать мы будем с термометром сопротивления. В параметрах модуля для этого сенсора в поле «Тип сенсора» выбираем вариант r385_500 (рис. 6). Таким образом мы предполагаем, что к контроллеру будет подключен датчик Pt500. Далее указываем переменную, которая будет отвечать за измеренную температуру (рис. 7).

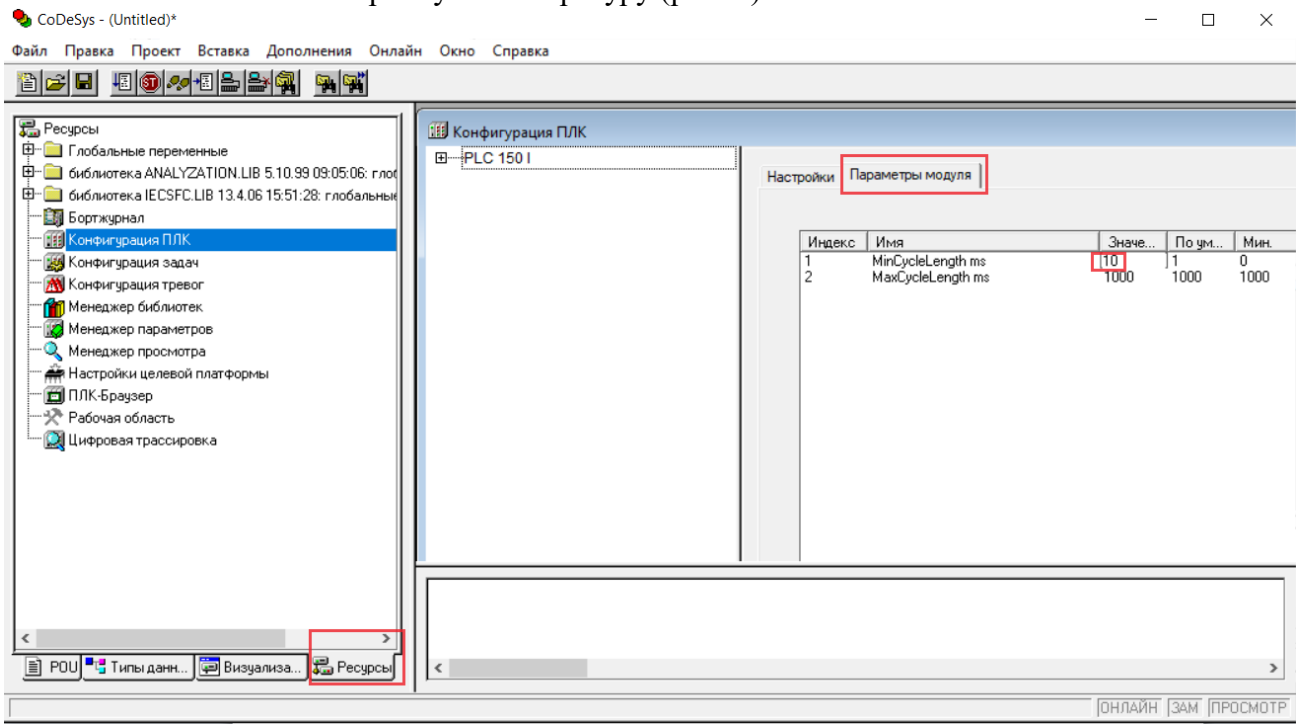


Рис. 3

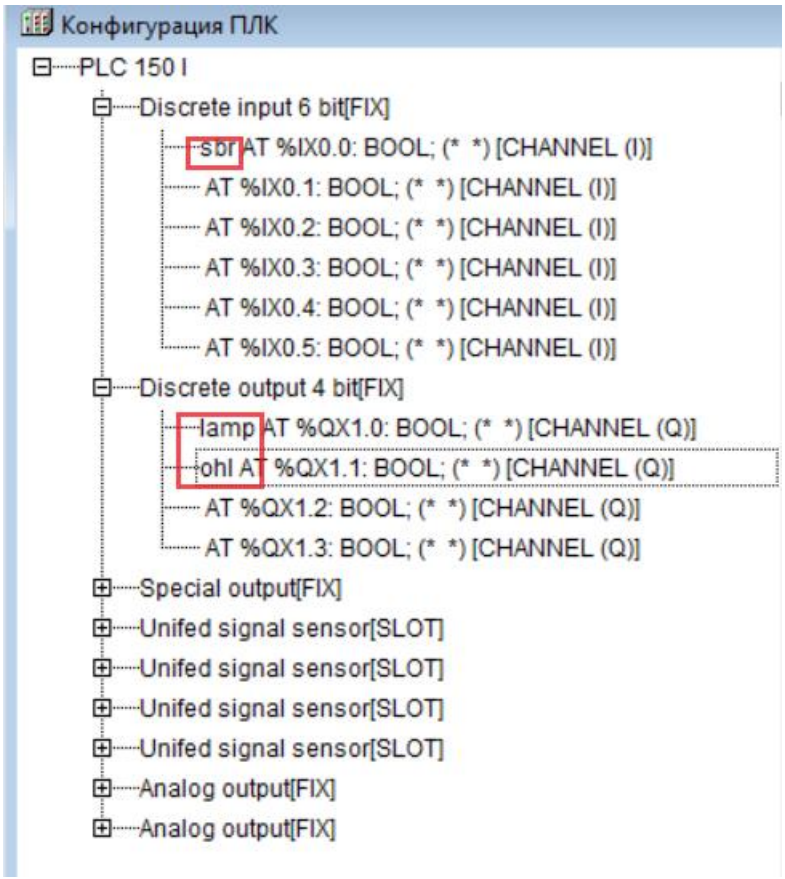


Рис. 4

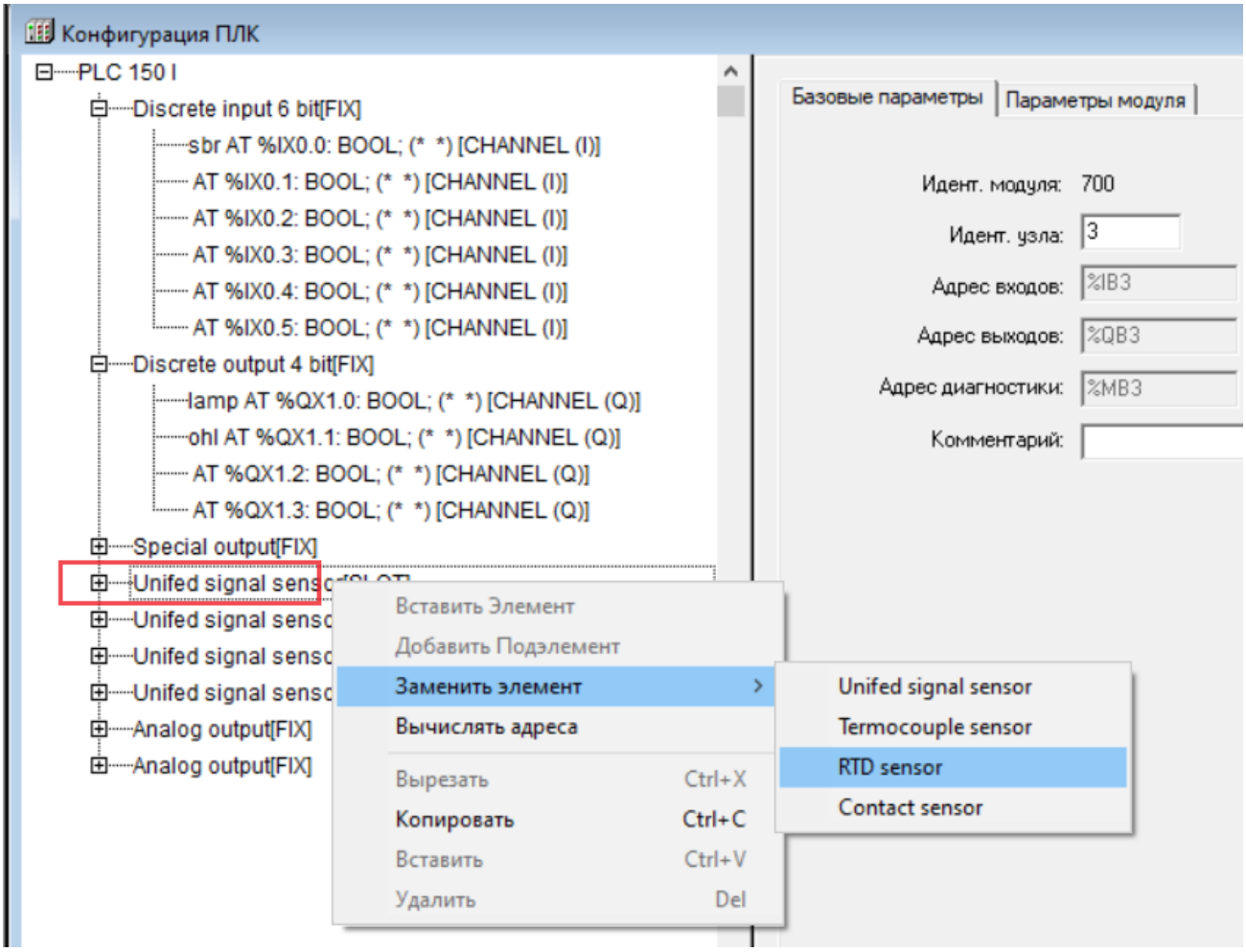

Конфигурация ПЛК
PLC 150 I
Discrete input 6 bit[FIX]
sbr AT %IX0.0: BOOL; (* *) [CHANNEL (I)]
AT %IX0.1: BOOL; (* *) [CHANNEL (I)]
AT %IX0.2: BOOL; (* *) [CHANNEL (I)]
AT %IX0.3: BOOL; (* *) [CHANNEL (I)]
AT %IX0.4: BOOL; (* *) [CHANNEL (I)]
AT %IX0.5: BOOL; (* *) [CHANNEL (I)]
Discrete output 4 bit[FIX]
lamp AT %QX1.0: BOOL; (* *) [CHANNEL (Q)]
ohl AT %QX1.1: BOOL; (* *) [CHANNEL (Q)]
AT %QX1.2: BOOL; (* *) [CHANNEL (Q)]
AT %QX1.3: BOOL; (* *) [CHANNEL (Q)]
Special output[FIX]
Unifed signal sens
Unifed signal sensc
Unifed signal sensc
Unifed signal sensc
Analog output[FIX]
Analog output[FIX]
Базовые параметры
Параметры модуля
Идент. модуля: 700
Идент. узла: 3
Адрес входов: %IB3
Адрес выходов: %QB3
Адрес диагностики: %MB3
Комментарий:
Вставить Элемент
Добавить Подэлемент
Заменить элемент
Вычислять адреса
Вырезать Ctrl+X
Копировать Ctrl+C
Вставить Ctrl+V
Удалить Del
Unifed signal sensor
Termocouple sensor
RTD sensor
Contact sensor

Рис.5

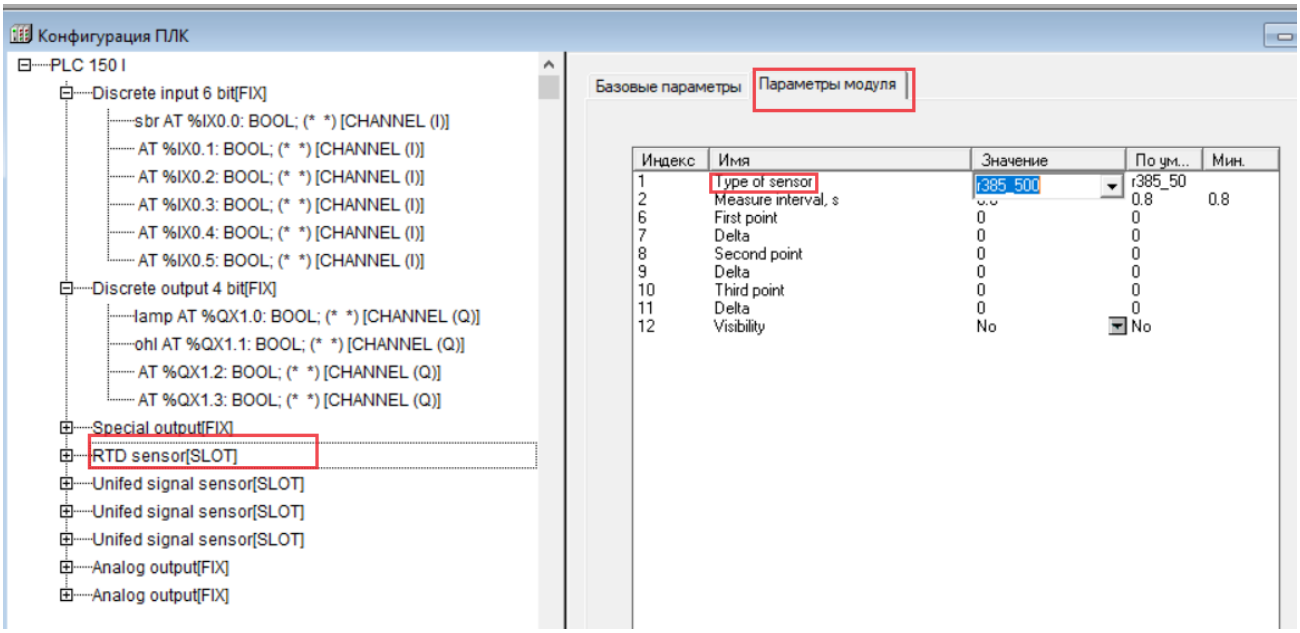

Конфигурация ПЛК
PLC 150 I
Discrete input 6 bit[FIX]
sbr AT %IX0.0: BOOL; (* *) [CHANNEL (I)]
AT %IX0.1: BOOL; (* *) [CHANNEL (I)]
AT %IX0.2: BOOL; (* *) [CHANNEL (I)]
AT %IX0.3: BOOL; (* *) [CHANNEL (I)]
AT %IX0.4: BOOL; (* *) [CHANNEL (I)]
AT %IX0.5: BOOL; (* *) [CHANNEL (I)]
Discrete output 4 bit[FIX]
lamp AT %QX1.0: BOOL; (* *) [CHANNEL (Q)]
ohl AT %QX1.1: BOOL; (* *) [CHANNEL (Q)]
AT %QX1.2: BOOL; (* *) [CHANNEL (Q)]
AT %QX1.3: BOOL; (* *) [CHANNEL (Q)]
Special output[FIX]
RTD sensor[SLOT]
Unifed signal sensor[SLOT]
Unifed signal sensor[SLOT]
Unifed signal sensor[SLOT]
Analog output[FIX]
Analog output[FIX]
Базовые параметры
Параметры модуля
Индекс
Имя
Значение
По ум...
Мин.
1 Type of sensor r385_500 r385_50
2 Measure interval, s 0.8 0.8
6 First point 0 0
7 Delta 0 0
8 Second point 0 0
9 Delta 0 0
10 Third point 0 0
11 Delta 0 0
12 Visibility No No

Рис. 6

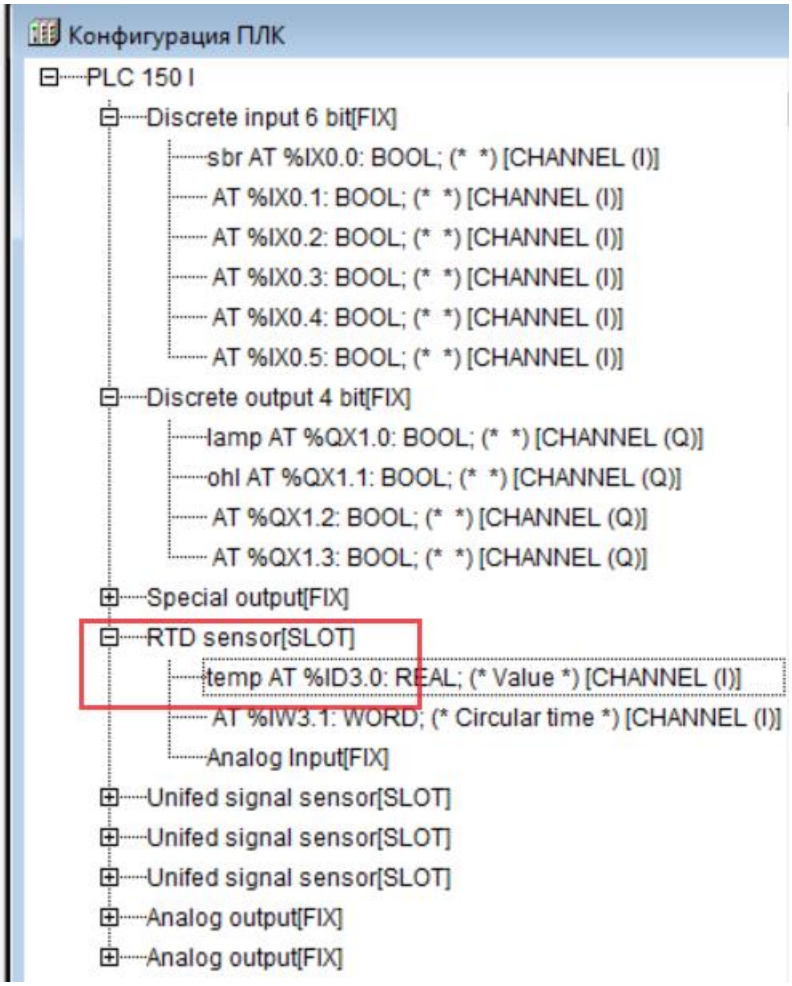


Рис.7

3. В главной программе PLC_PRG задаём нужный набор промежуточных переменных.

Для этого с помощью команды Shift+F2 вызываем окно объявления переменных, и в нём по очереди указываем нужные имена переменных и их типы (рис. 8).

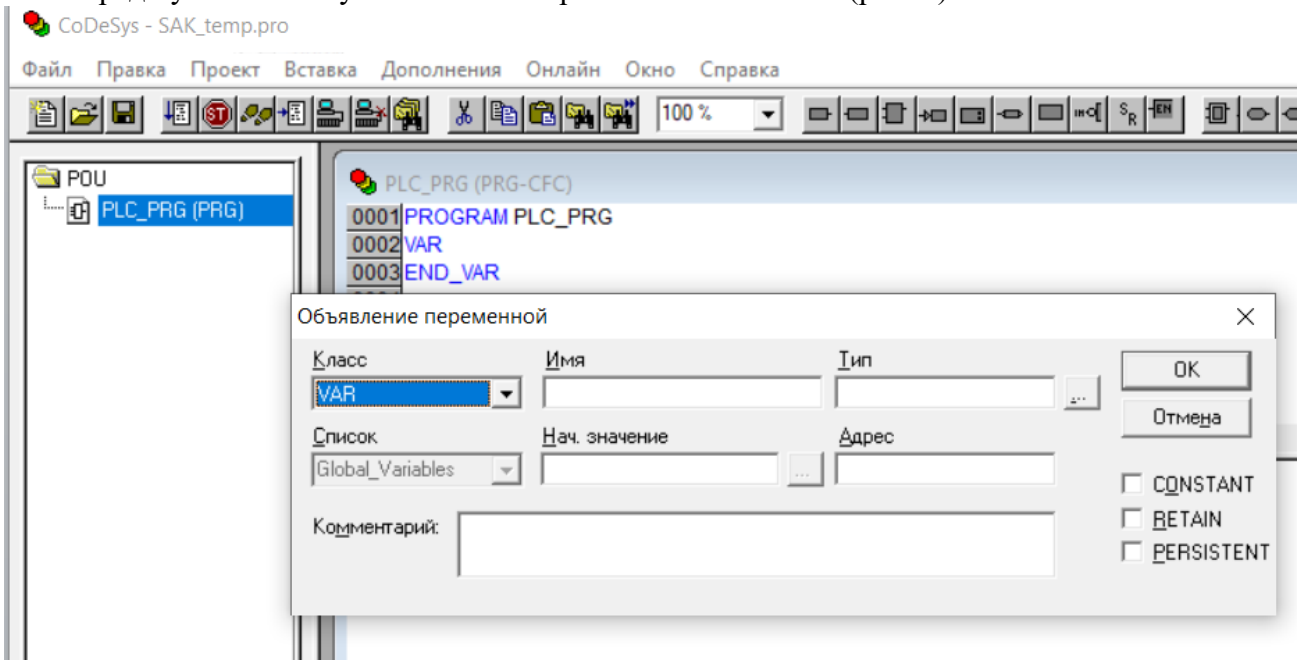


Рис. 8

Перечень промежуточных переменных:

- alarm – для сохранения текущего статуса «Авария», то есть либо есть авария, либо её нет (рис.9);

- ust – заданное значение температуры (уставка), обязательно того же типа, что и текущее значение температуры, то есть типа REAL, при этом можно сразу указать начальное значение задания, например, 80 (рис. 10);
- tim – время, через которое должна сработать сигнализация, если текущее значение температуры превысит уставку (рис. 11).

Итоговый список промежуточных переменных представлен на рис.12.

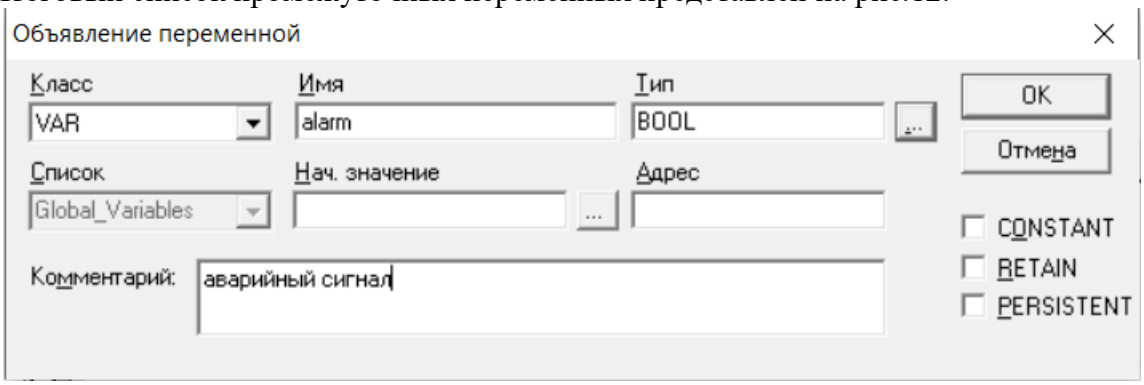


Рис.9

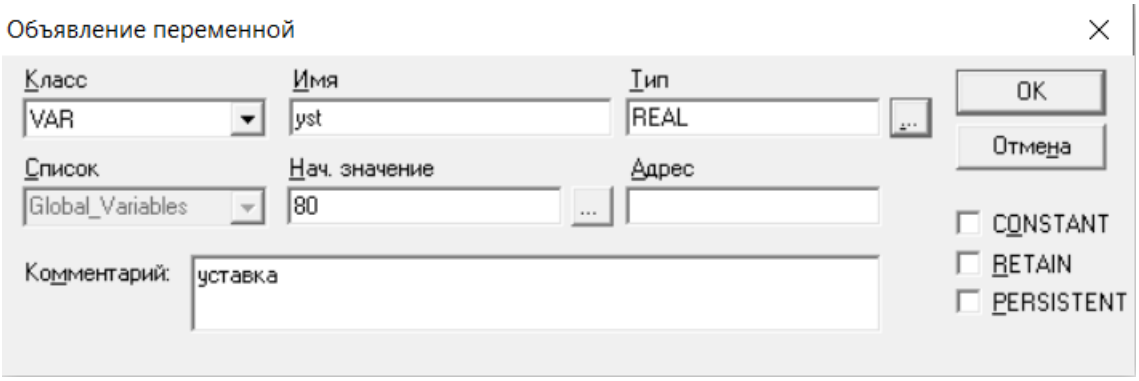


Рис. 10

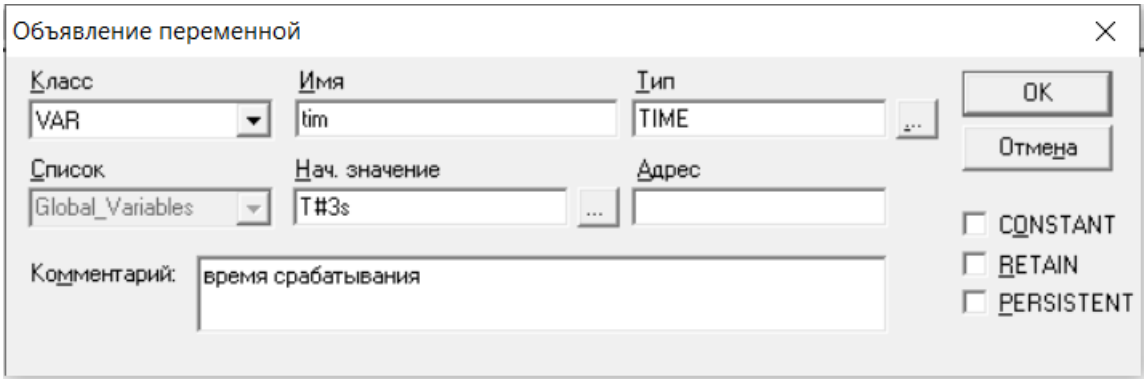


Рис. 11

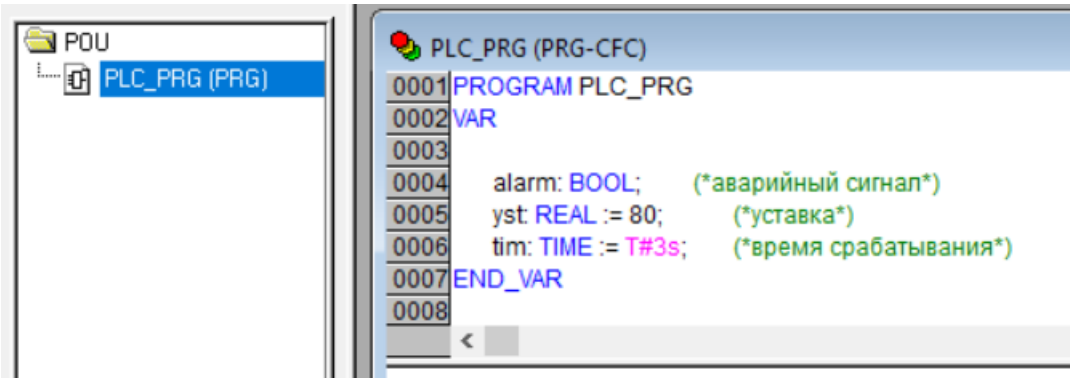


Рис.12

4. Для того, чтобы лампа сигнализации мигала, понадобится библиотечный компонент – блок «BLINK». Подключаем библиотеку Util.lib, в которой он находится. Для этого во вкладке проекта «Ресурсы» выбираем раздел «Менеджер библиотек» и в свободном месте

поля со списком подключенных библиотек через правую кнопку вызываем контекстное меню и выбираем пункт «Добавить библиотеку» (рис. 13). Затем в перечне библиотек выбираем файл Util.lib и добавляем его в проект его через кнопку «Открыть» (рис. 14). Нужный нам блок «BLINK» находится в разделе «signals» (рис. 15). Можно изучить его структуру.

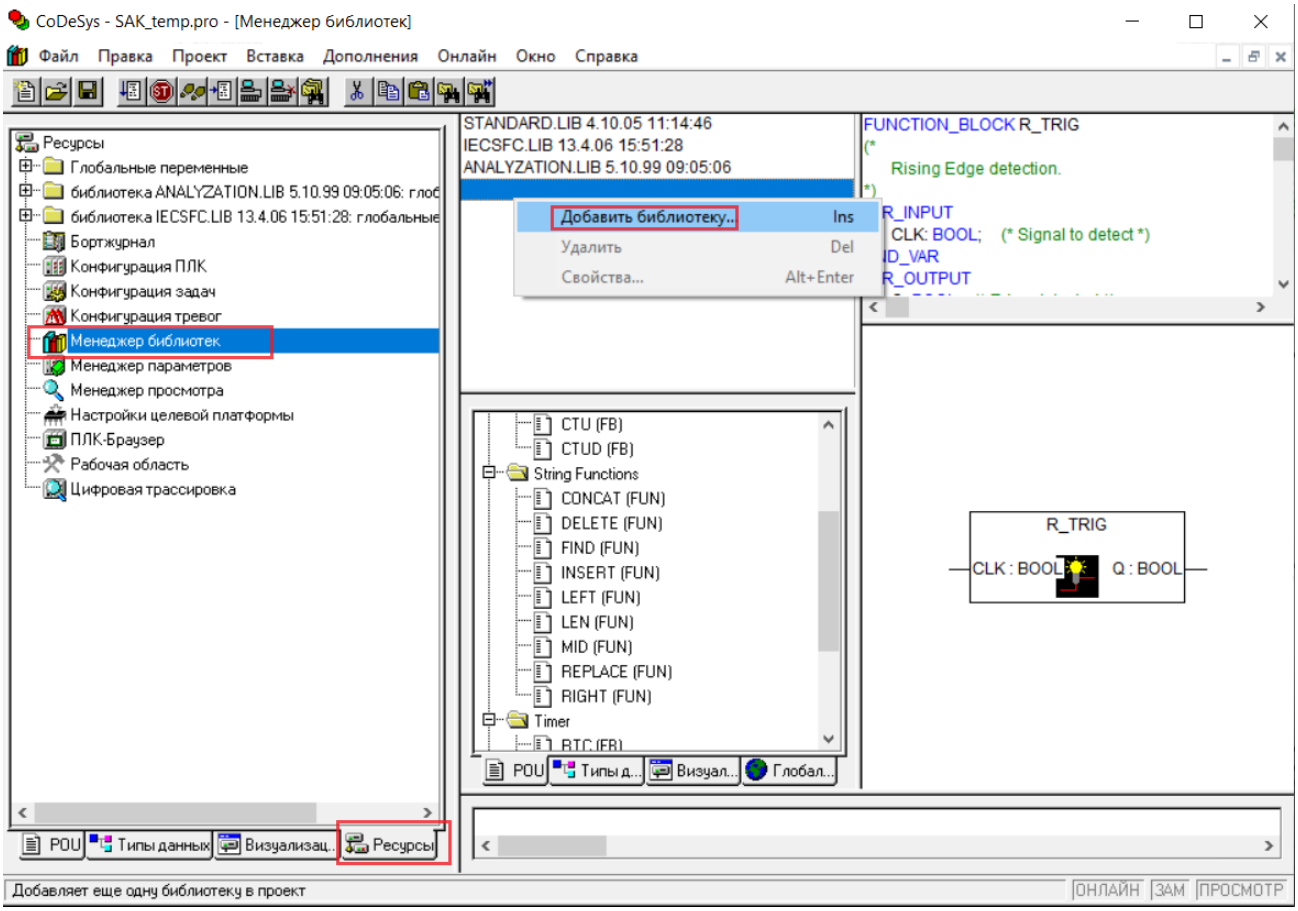


Рис. 13

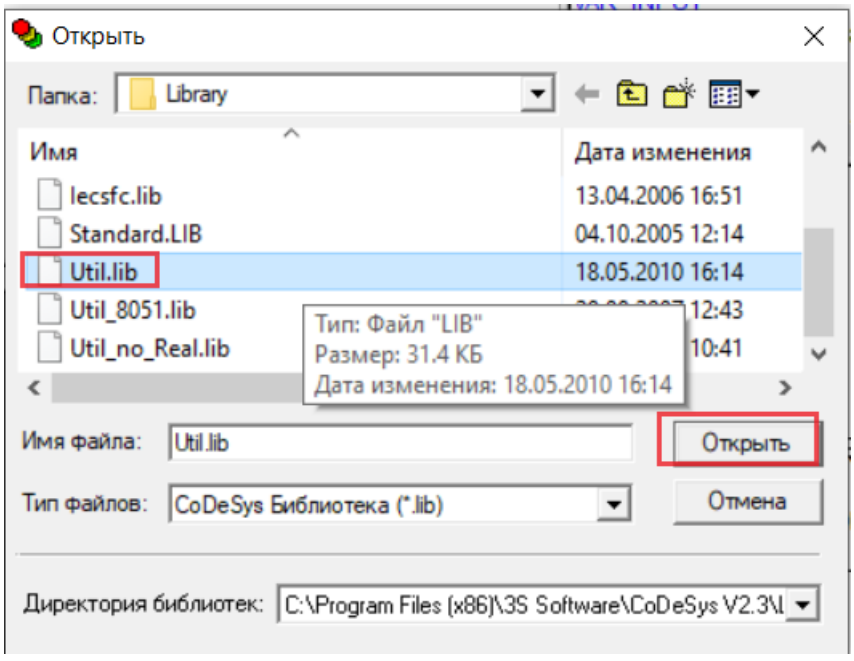


Рис. 14

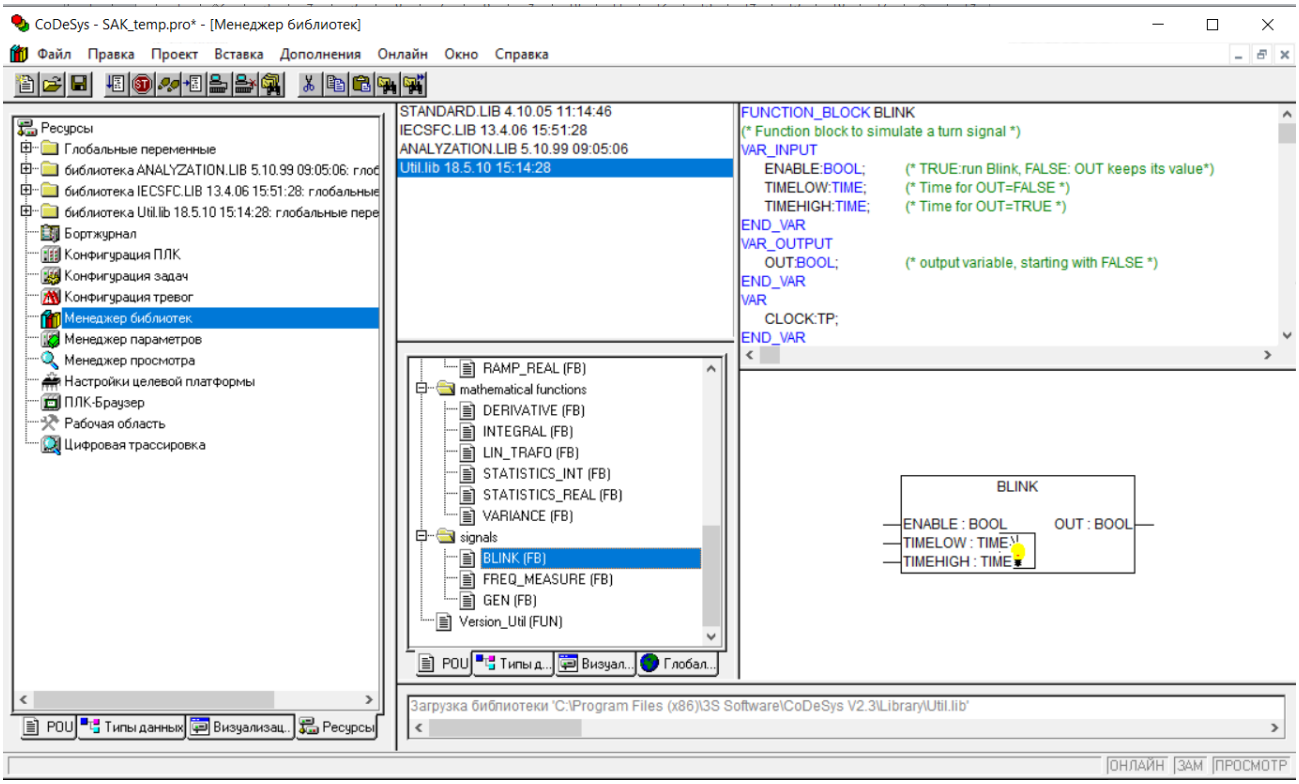


Рис. 15

5. В главной программе PLC_PRG пишем на языке CFC код программы (рис. 16).

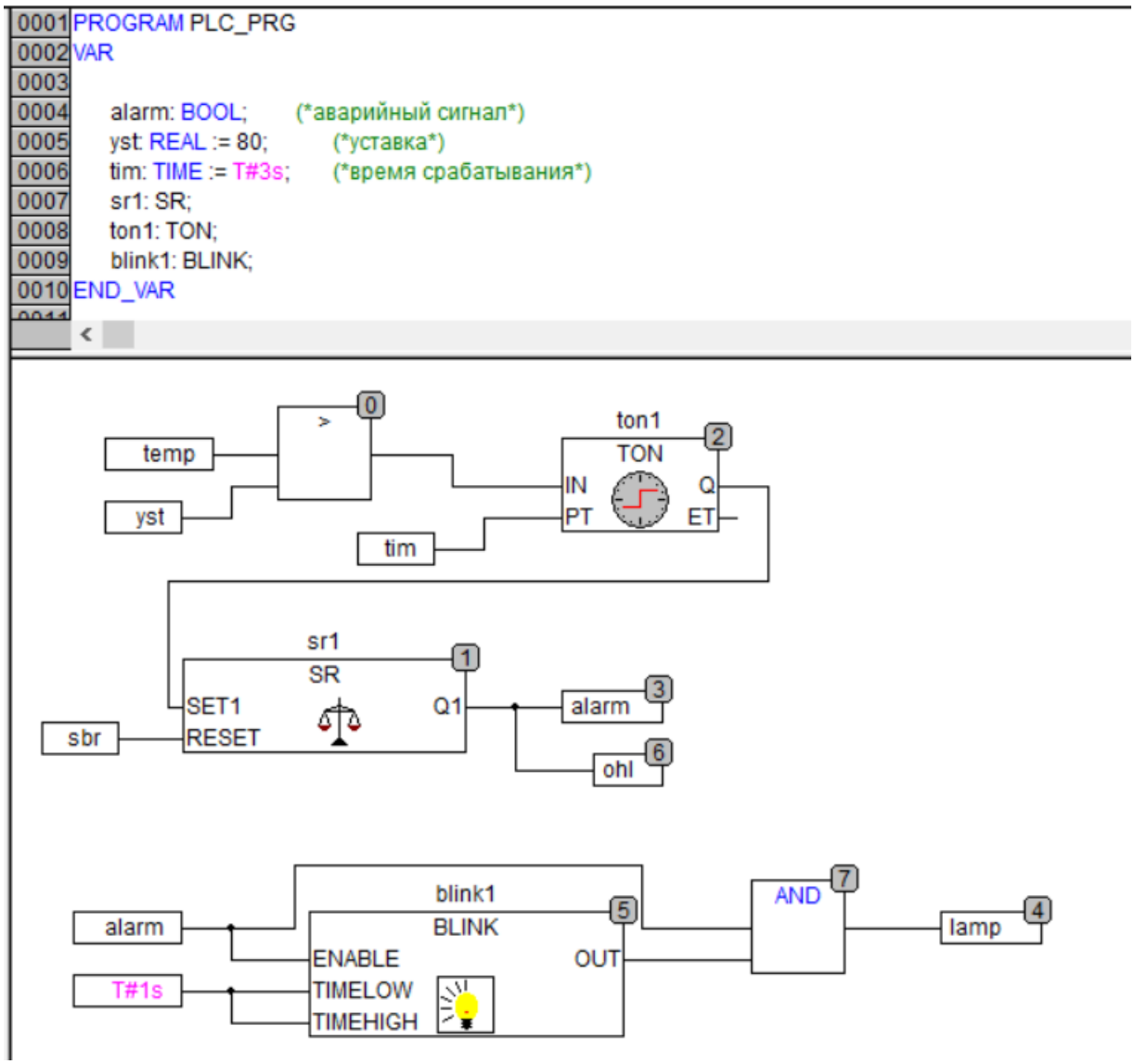
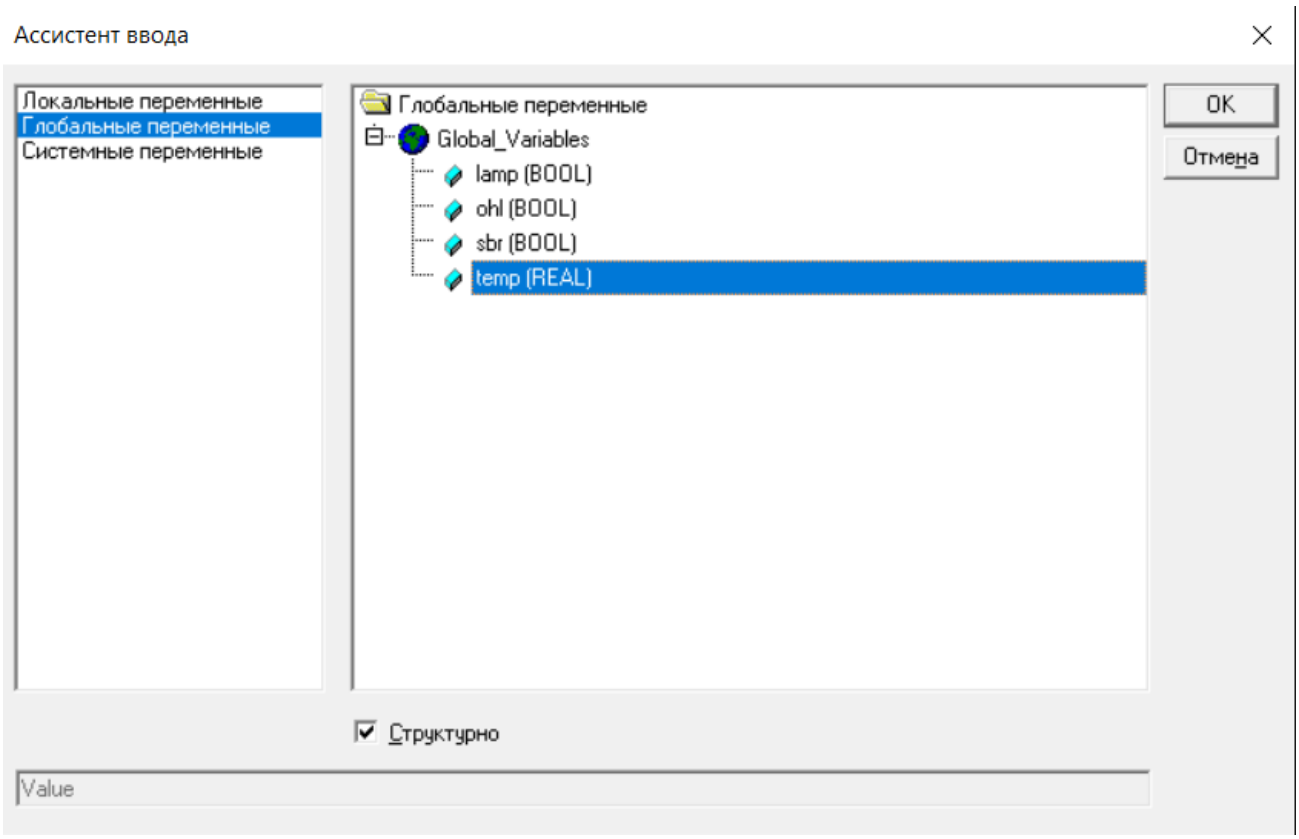


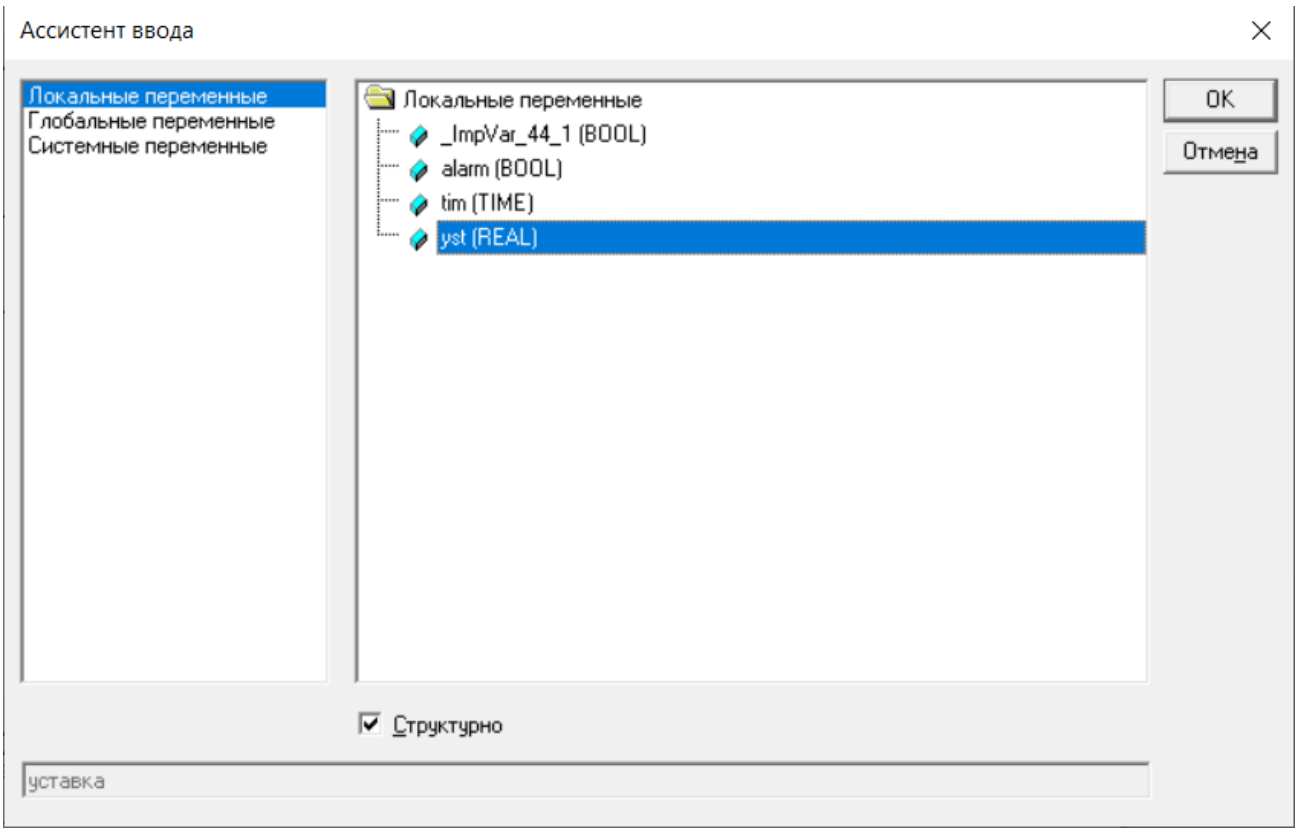
Рис. 16

ВАЖНО!!!

Все глобальные и локальные переменные для входов и выходов блоков указываем через ассистент ввода с помощью команды F2 (рис. 17, а, б).



а)



б)

Рис. 17

Сравнение текущего значения температуры с заданным (уставкой) реализуем в блоке сравнения «<>».

Далее добавляем таймер задержки включения – блок TON с временем задержки 3 секунды. Так как таймер является библиотечным алгоритмом, то для него необходимо придумать имя экземпляра, например, ton1. Вход IN таймера соединяем с выходом оператора сравнения заданного

и текущего значений температуры. На второй вход таймера подаём переменную «tim», в которой мы задали ранее время задержки 3 секунды.

Затем добавим библиотечный алгоритм, реализующий триггер с предустановкой - SR-триггер. Так как SR-триггер является библиотечным алгоритмом, то для него необходимо придумать имя экземпляра, например, sr1. На вход SR-триггера подаём сигнал с таймера задержки включения. На выход SR-триггера подключаем сигнализацию – локальную переменную «alarm». На выход SR-триггера также подключаем охладитель, то есть значение глобальной переменной «ohl».

Если текущая температура держится выше заданной больше трёх секунд, то на выходе таймера появляется сигнал «True», который включает SR-триггер, и сигнализация срабатывает. Одновременно включится охладитель.

Сброс сигнализации и отключение охладителя должны происходить при условии, что текущая температура стала ниже заданной, но только при нажатой кнопке «Сброс». Для этого на вход RESET SR-триггера подключаем сигнал с этой кнопки, то есть значение глобальной переменной «sbr».

Таким образом, когда значение переменной «temp» станет меньше уставки «yst», и будет нажата кнопка «sbr», сигналы в переменных «alarm» и «ohl» пропадут.

Для реализации мигания лампы сигнализации во время аварии вызываем библиотечный алгоритм «BLINK», даём для его экземпляра имя blink1. Далее на временные входы блока «BLINK» задаём отрезки для мигания – 1 секунда для выключенного и включенного состояния (можно использовать одну константу).

Мигать лампа будет тогда, когда есть аварийный режим, поэтому на первый вход блока «BLINK» подаём значение переменной «alarm».

Далее используем логический блок «AND», чтобы сформировать условие включения лампы (глобальная переменная «lamp»): лампа будет включена при наличии сигнала с блока «BLINK» и при наличии сигнала аварийного режима с переменной «alarm».

После добавления всех необходимых блоков обязательно сохраняем проект, а далее желательно указать порядок работы блоков в соответствии с основным потоком данных. Для этого в свободном месте поля проекта вызываем контекстное меню и указываем «Порядок» - «Показать порядок» - «В соответствии с потоком данных» (рис. 18).

Итоговый программный код показан на рис. 19.

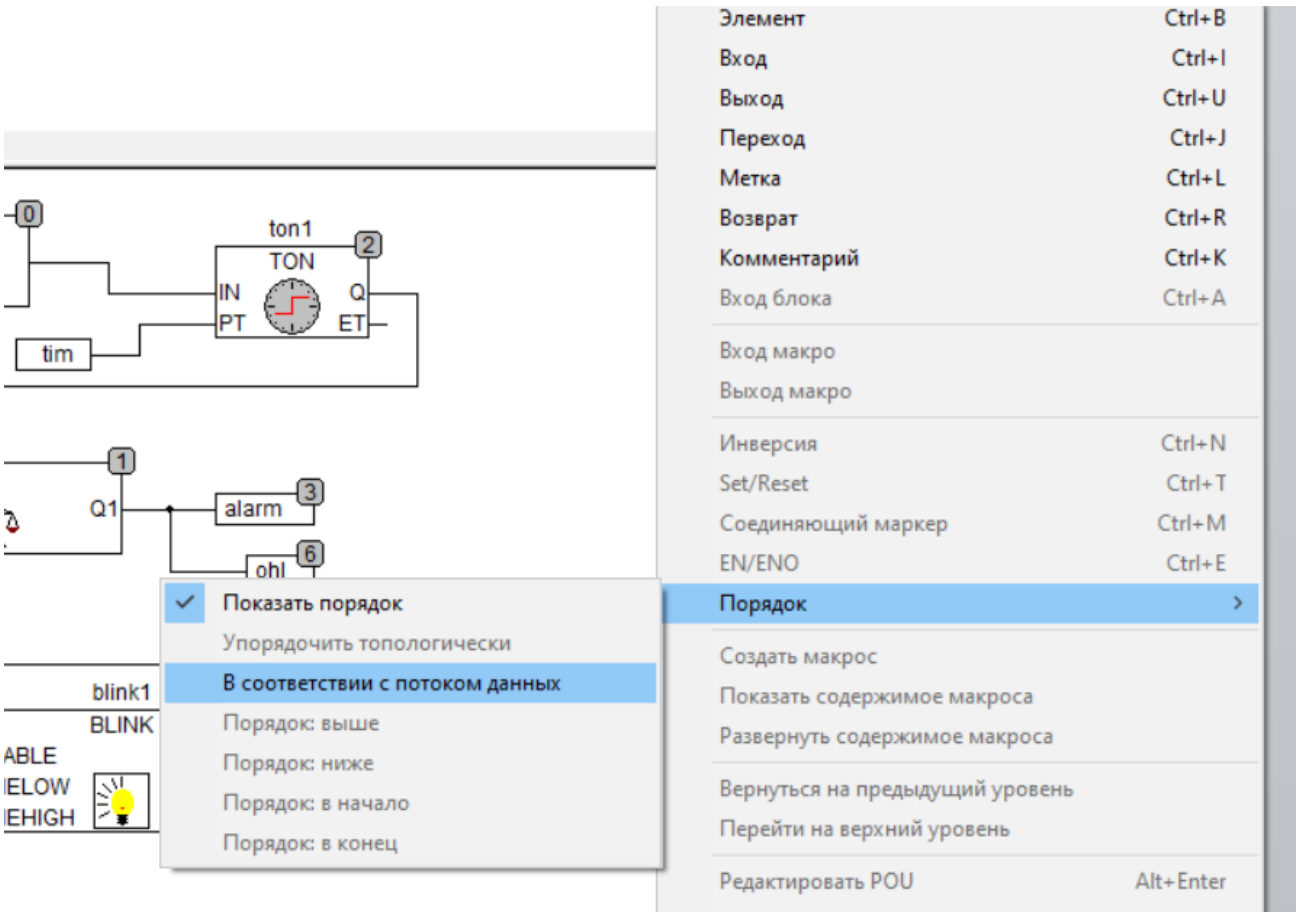

0
ton1
TON
2
IN
PT
Q
ET
tim
1
Q1
alarm
3
ohl
6
blink1
BLINK
ABLE
ELOW
EHIGH
Показать порядок
Упорядочить топологически
В соответствии с потоком данных
Порядок: выше
Порядок: ниже
Порядок: в начало
Порядок: в конец
Элемент Ctrl+B
Вход Ctrl+I
Выход Ctrl+U
Переход Ctrl+J
Метка Ctrl+L
Возврат Ctrl+R
Комментарий Ctrl+K
Вход блока Ctrl+A
Вход макро
Выход макро
Инверсия Ctrl+N
Set/Reset Ctrl+T
Соединяющий маркер Ctrl+M
EN/ENO Ctrl+E
Порядок
Создать макрос
Показать содержимое макроса
Развернуть содержимое макроса
Вернуться на предыдущий уровень
Перейти на верхний уровень
Редактировать POU Alt+Enter

Рис. 18

```
0001 PROGRAM PLC_PRG
0002 VAR
0003
0004     alarm: BOOL;       (*аварийный сигнал*)
0005     yst: REAL := 80;       (*уставка*)
0006     tim: TIME := T#3s;     (*время срабатывания*)
0007     sr1: SR;
0008     ton1: TON;
0009     blink1: BLINK;
0010 END_VAR
```

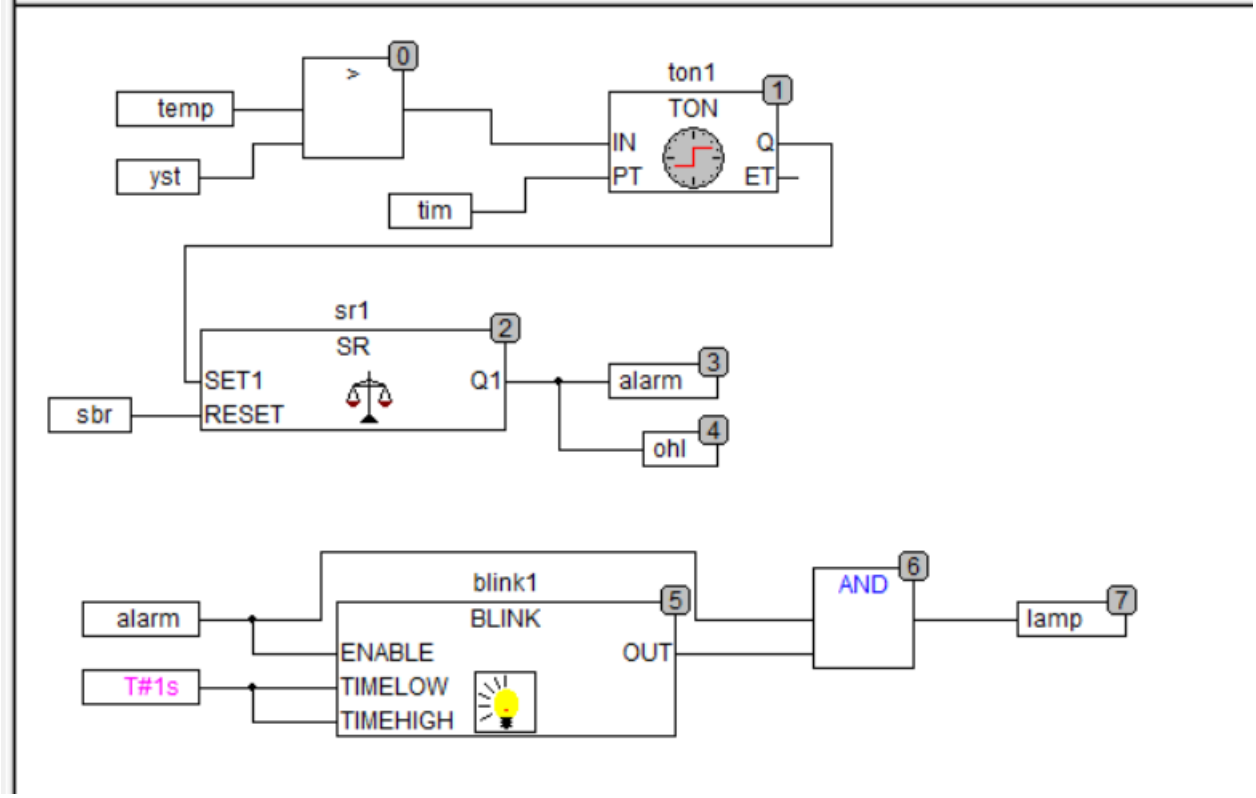


Рис. 19

6. Через клавишу F11 проверяем наличие ошибок. Если ошибок нет, то приступаем к созданию визуализации проекта.
7. Во вкладке проекта «Визуализация» через контекстное меню выбираем команду «Добавить объект», пишем имя визуализации, например «SAK_viz».
8. Для создания визуализации изменения текущего значения температуры используем элемент визуализации «Ползунок» (рис. 20).

Кликнув дважды левой кнопкой мыши по созданному элементу, вызываем окно его конфигурирования и в категории «Переменные» для поля «Ползунок» с помощью клавиши F2 выбираем значение глобальной переменной «temp». Здесь же указываем диапазон изменения текущего значения температуры, например, от 20 до 120 °С (рис. 21).

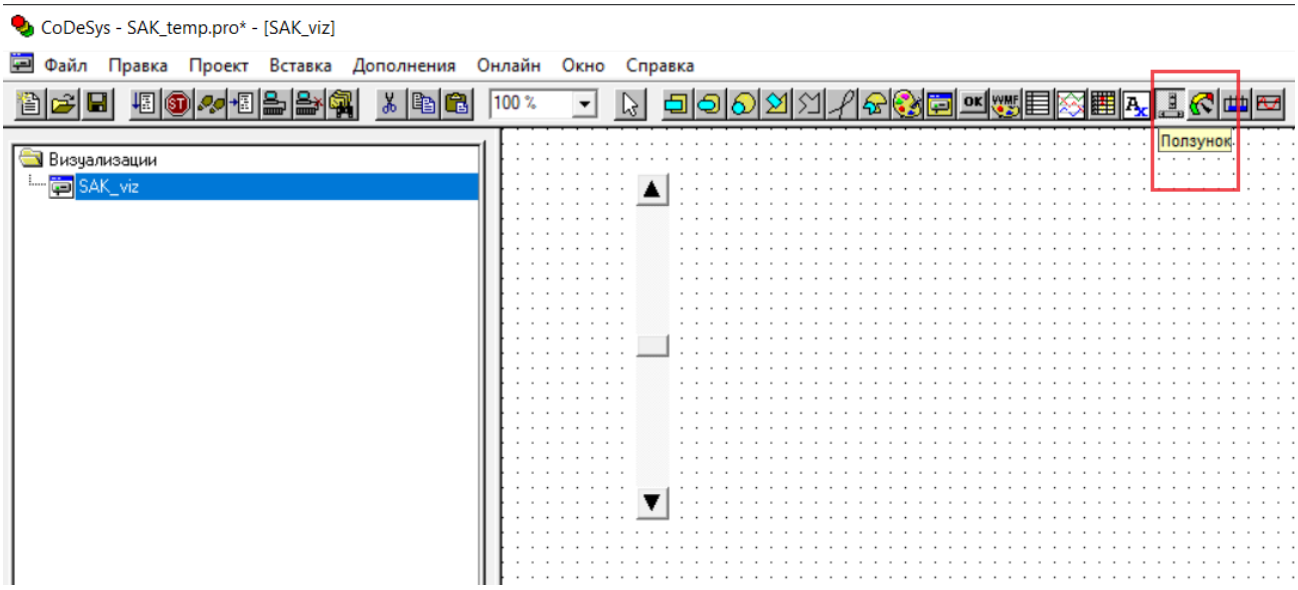


Рис. 20

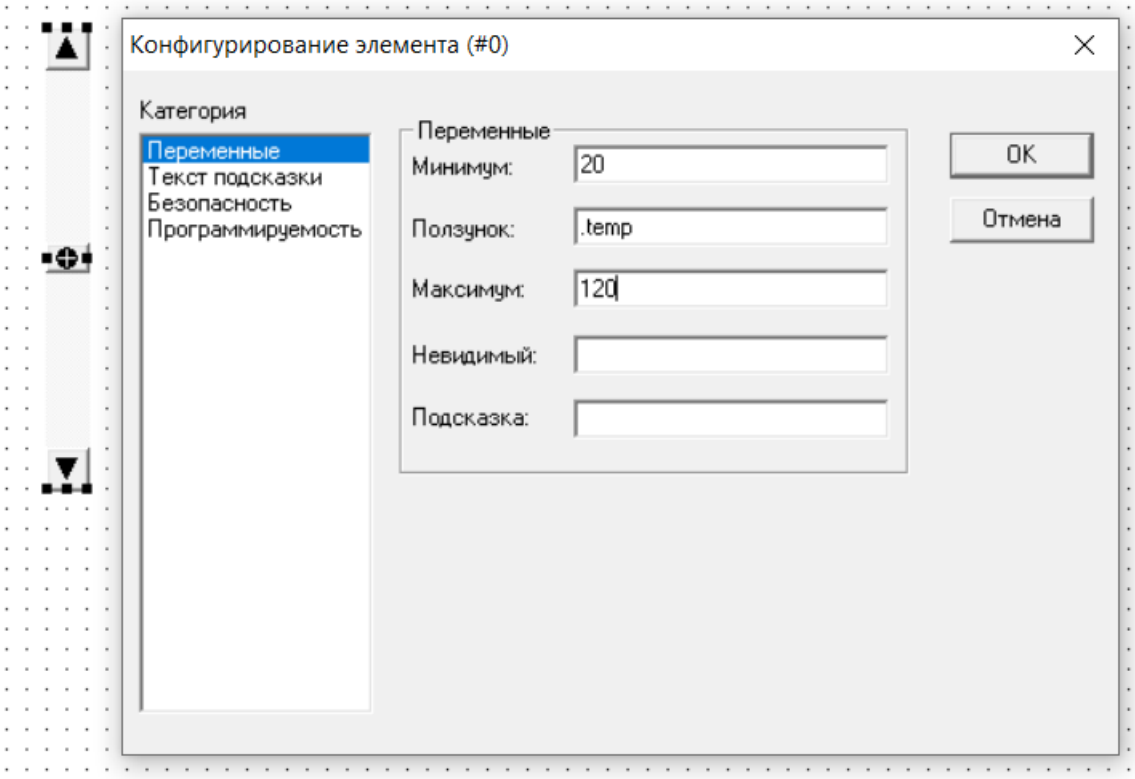


Рис. 21

9. Для того, чтобы видеть текущее значение температуры в числовом формате, используем элемент визуализации «Прямоугольник» (рис. 22). В окне конфигурирования элемента в категории «Текст» сначала выбираем шрифт с кирриллической раскладкой (через пункт Шрифт, рис. 23). Далее привязываем этот элемент к значению переменной «temp». Для этого используем специальный формат и в поле «Строка» пишем команду (рис. 24): temp = %3.1f.

Далее выбираем категорию «Переменные» и в поле «Выв_текста» с помощью клавиши F2 вводим значение глобальной переменной «temp» (рис. 25).

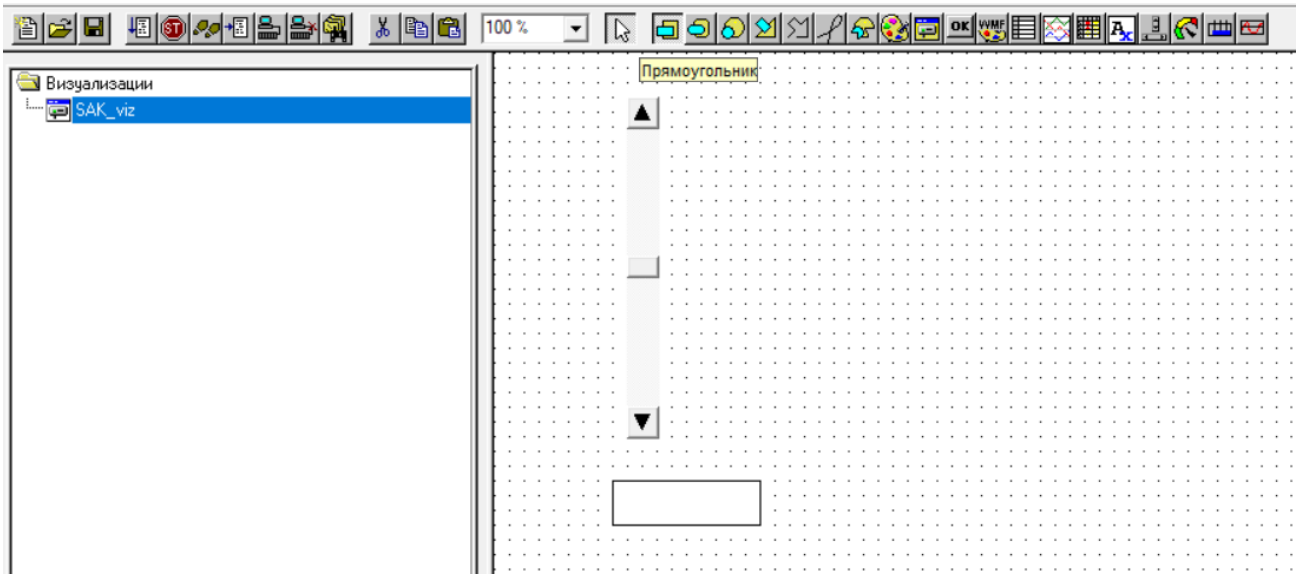

100 %
Визуализации
SAK_viz
Прямоугольник

Рис. 22

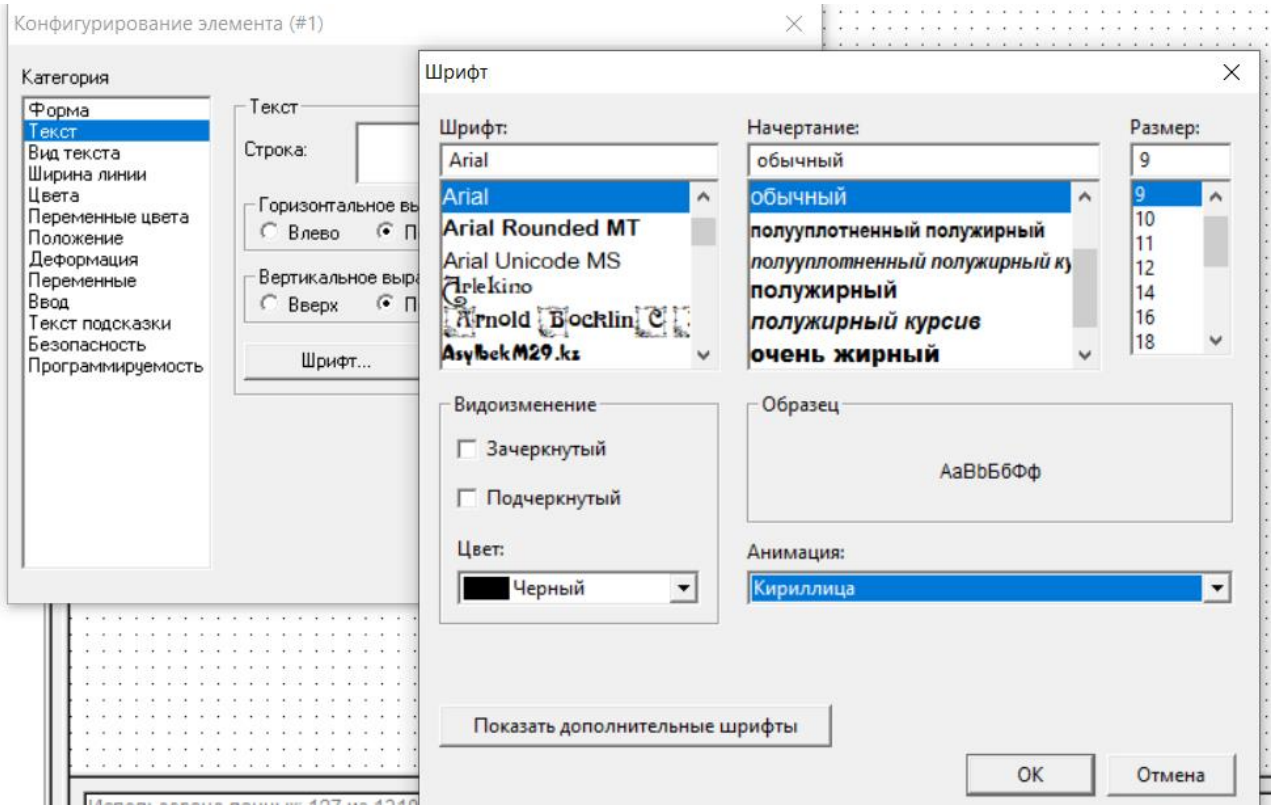

Конфигурирование элемента (#1)
Категория
Форма
Текст
Вид текста
Ширина линии
Цвета
Переменные цвета
Положение
Деформация
Переменные
Ввод
Текст подсказки
Безопасность
Программируемость
Текст
Строка:
Горизонтальное вы
Влево
Вертикальное выра
Вверх
Шрифт...
Шрифт
Шрифт:
Arial
Arial
Arial Rounded MT
Arial Unicode MS
Harlekino
Arnold Boecklin C
AsylbekM29.kz
Начертание:
обычный
обычный
полууплотненный полужирный
полууплотненный полужирный ку
полужирный
полужирный курсив
очень жирный
Размер:
9
9
10
11
12
14
16
18
Видоизменение
Зачеркнутый
Подчеркнутый
Цвет:
Черный
Образец
AaBbБбФф
Анимация:
Кириллица
Показать дополнительные шрифты
OK
Отмена

Рис. 23

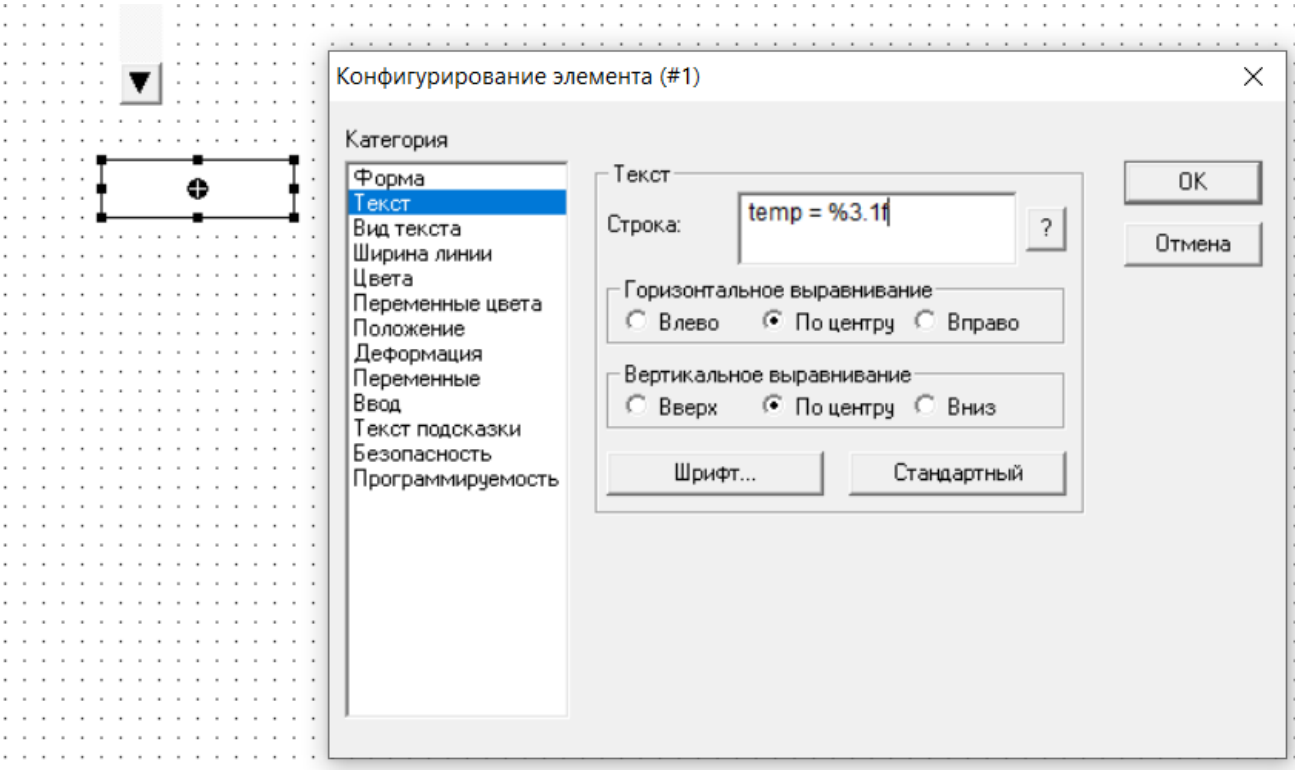


Рис. 24

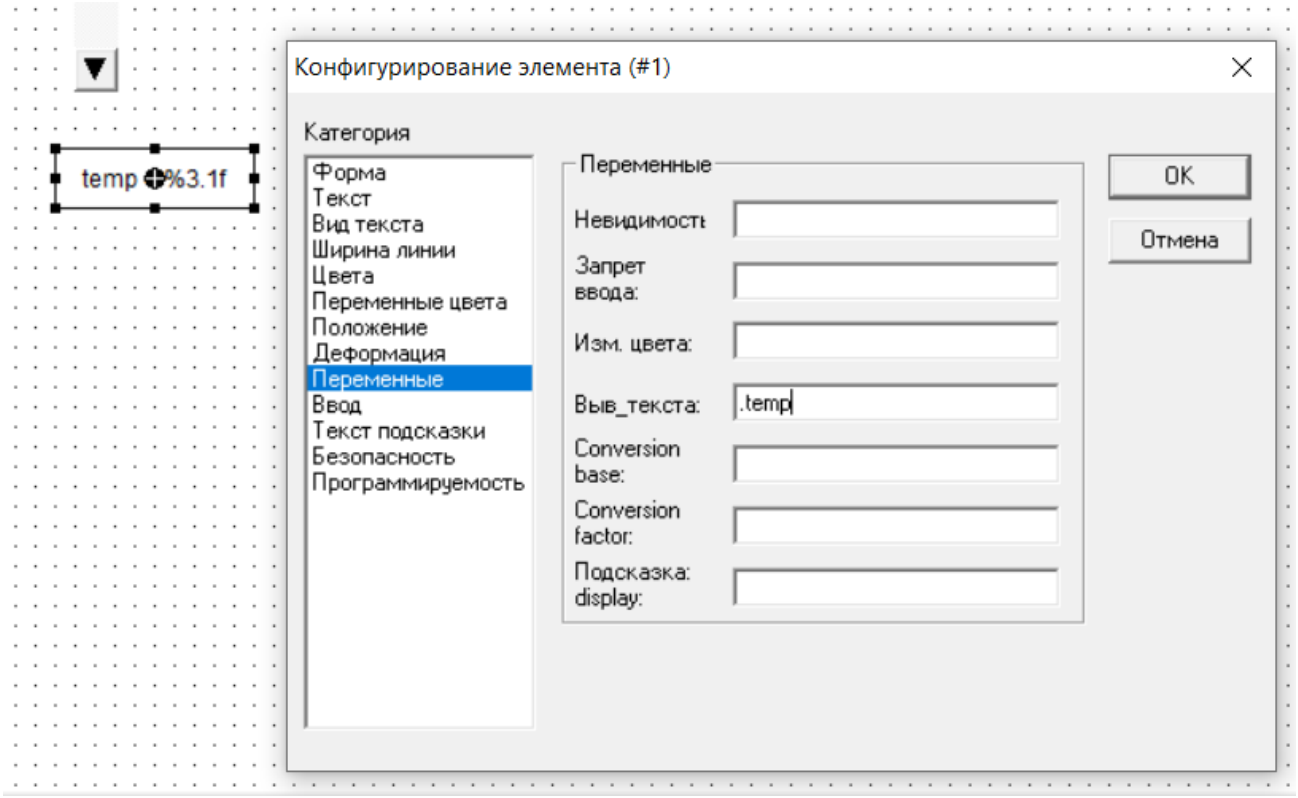


Рис. 25

10. Аналогичным образом реализуем визуализацию для того, чтобы видеть значение уставки температуры в числовом формате. Можно скопировать элемент визуализации для текущего значения температуры в числовом формате и заменить в окне конфигурации элемента значение переменной «temp» на значение переменной «yst» (рис. 26, 27).

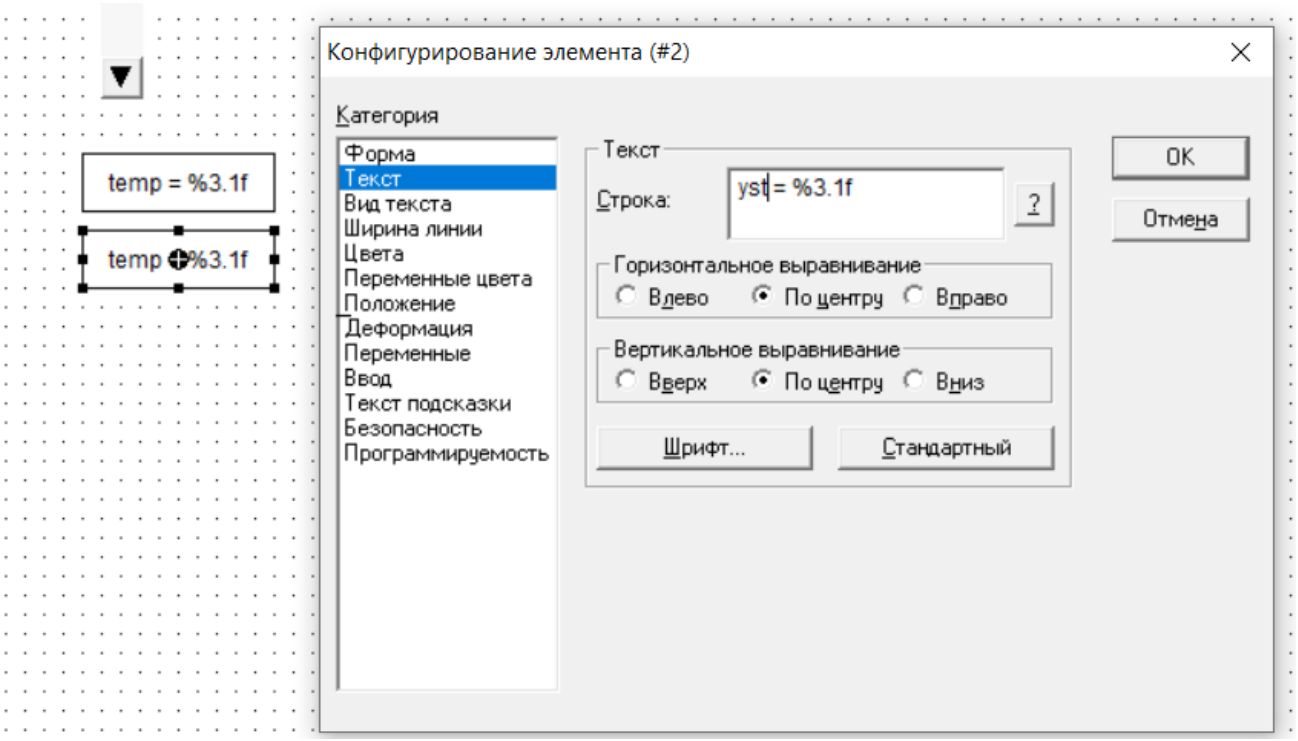


Рис. 26

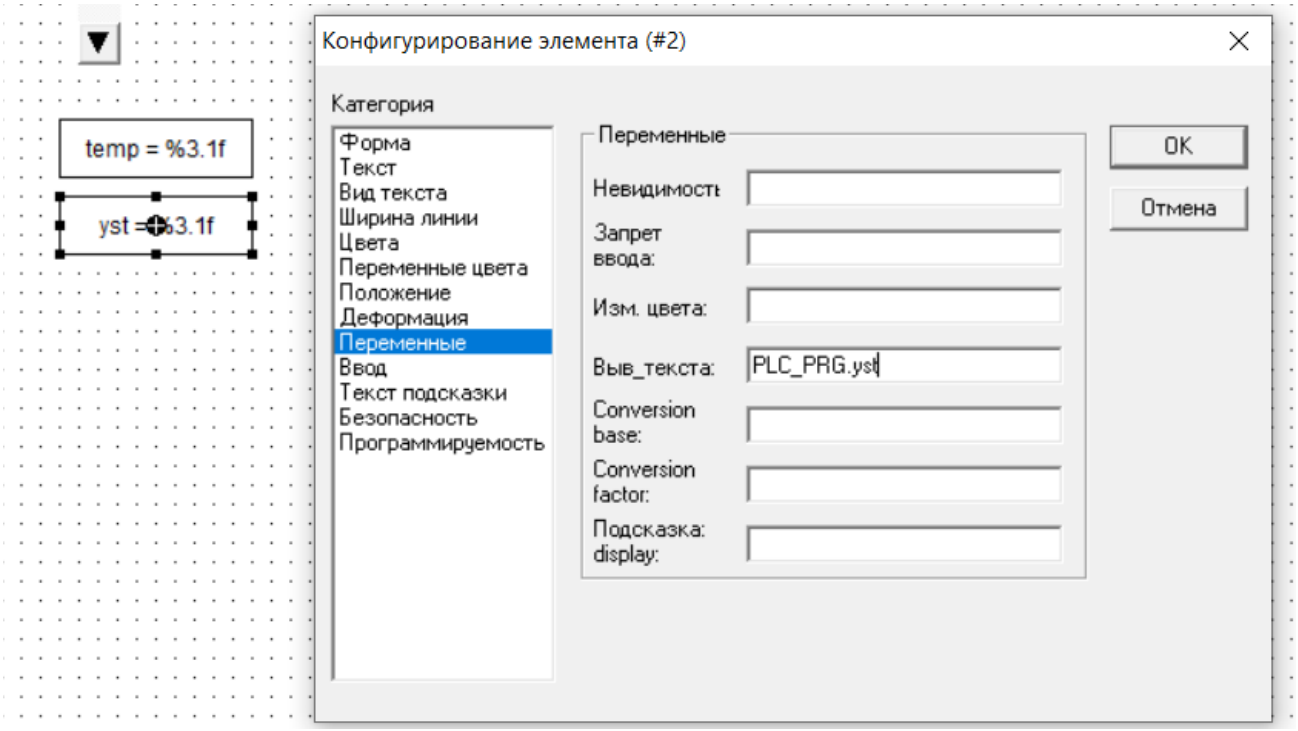


Рис. 27

11. Для того, чтобы реализовать в визуализации возможность изменения значения уставки, в окне конфигурирования этого элемента выбираем категорию «Ввод», далее выбираем элемент, который будет появляться при клике на кнопку уставки – «Цифровая панель», и указываем диапазон значений уставки, например, от 50 до 100 (рис. 28).

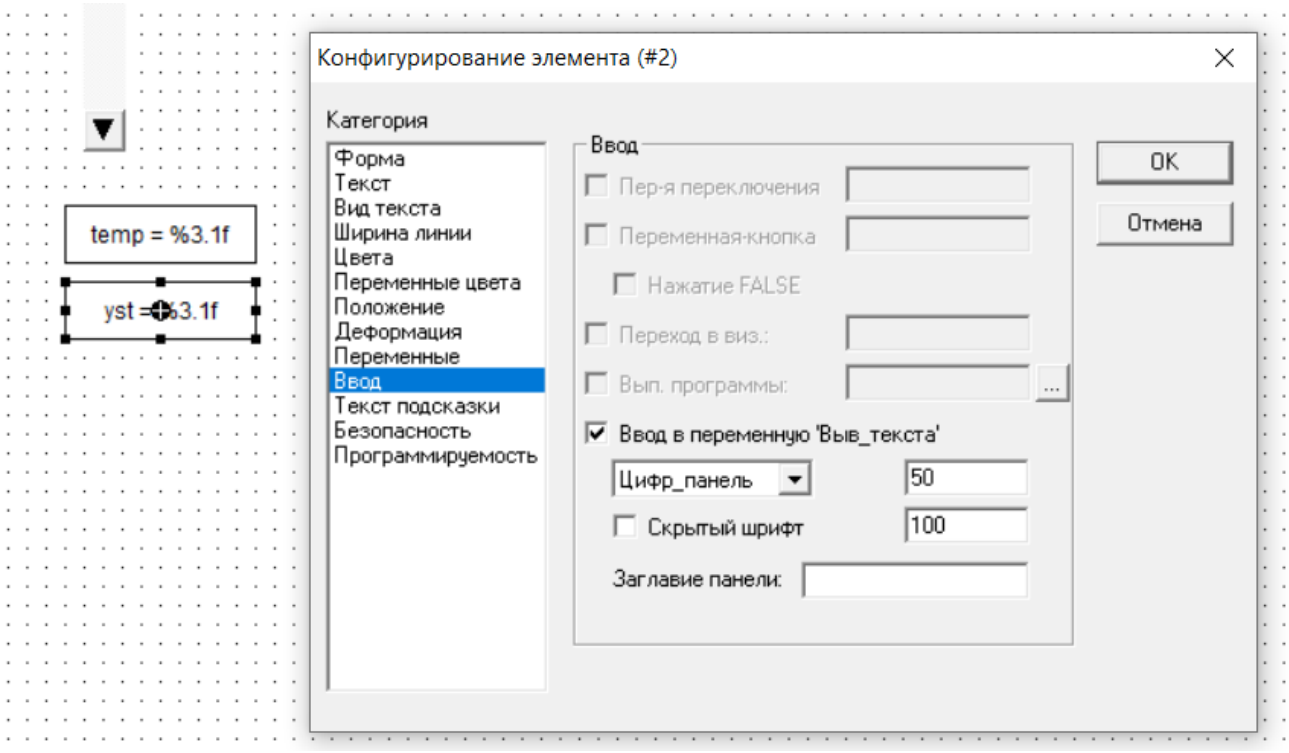
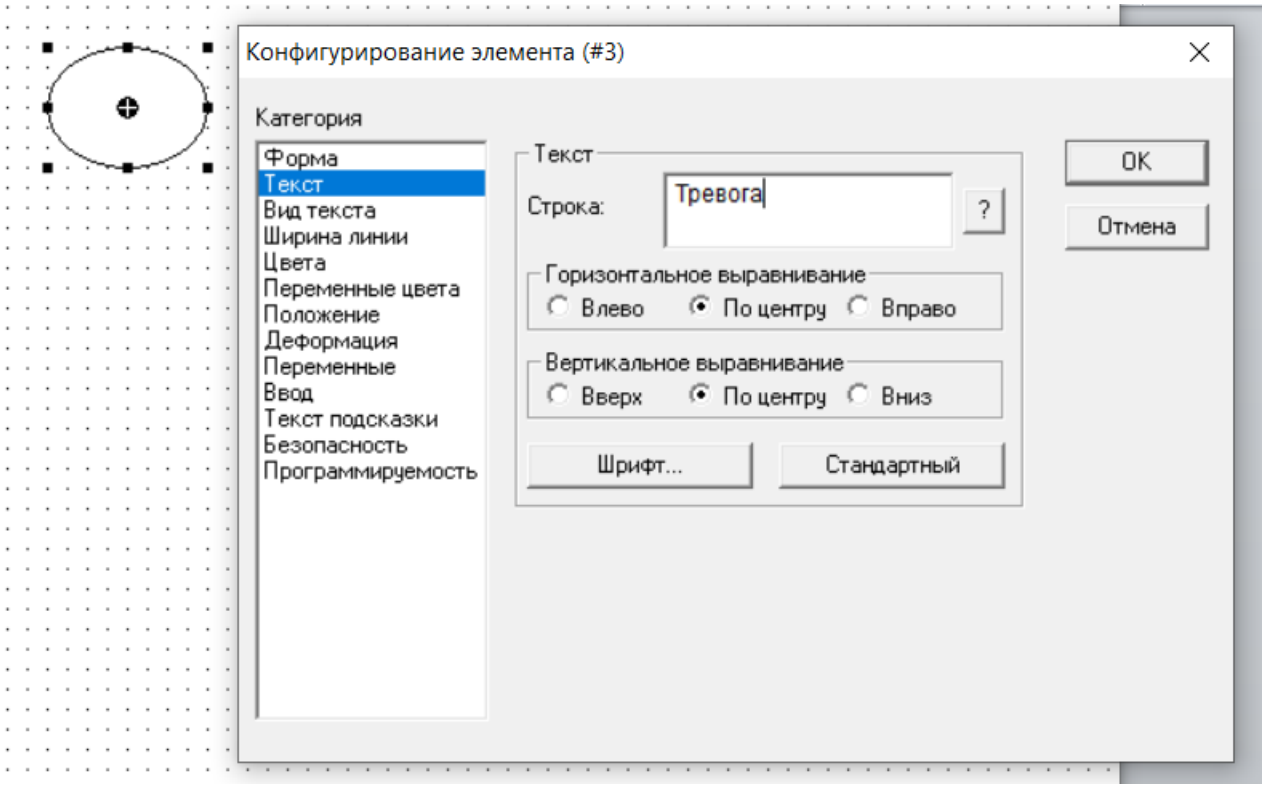
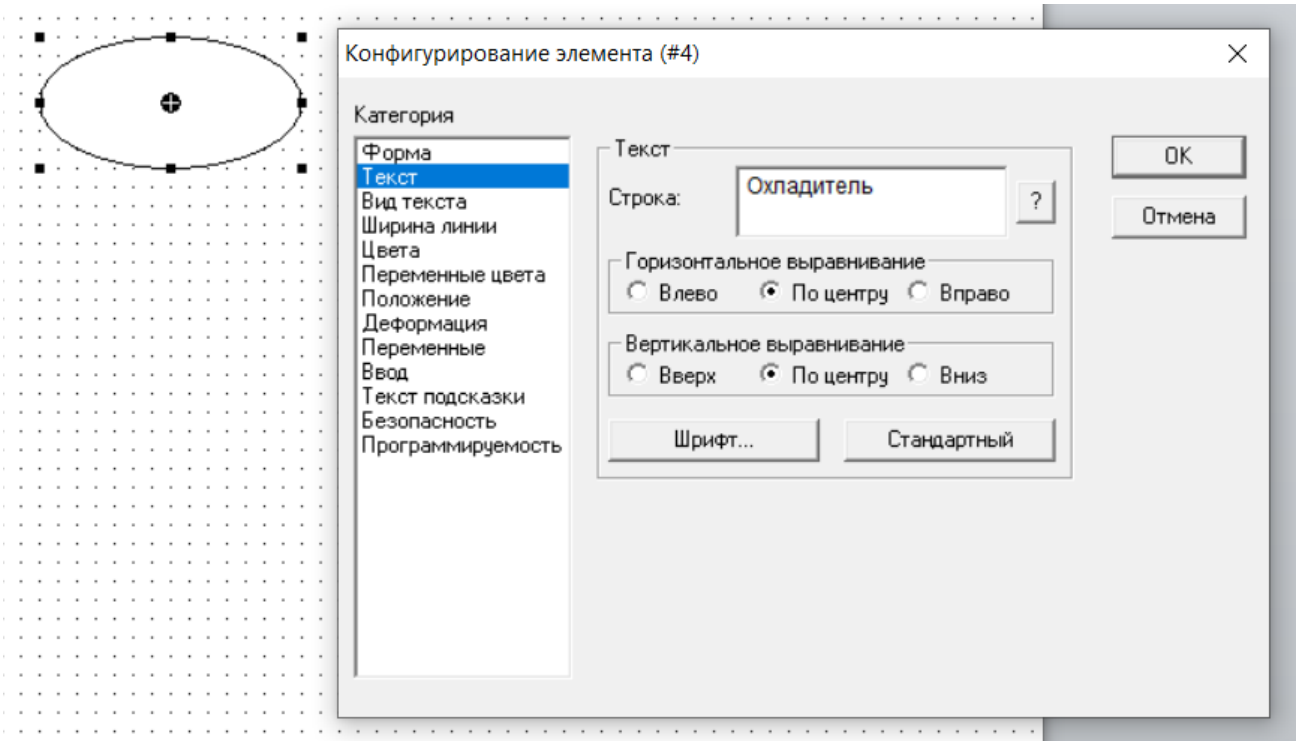


Рис. 28

12. Для отображения на визуализации включенного и выключенного состояний лампы сигнализации и охладителя используем элемент «Эллипс». В категории «Текст» указываем наименование элемента (рис. 29, а, б). В категории «Цвета» выбираем для выключенного состояния элементов серый цвет, а для тревожного (включенного) состояния – красный цвет (рис. 30). В категории «Переменные» в поле «Изм. цвета» с помощью клавиши F2 указываем значения переменных «lamp» и «ohl» соответственно (рис. 31, а, б)



а)


Конфигурирование элемента (#4)
Категория
Форма
Текст
Вид текста
Ширина линии
Цвета
Переменные цвета
Положение
Деформация
Переменные
Ввод
Текст подсказки
Безопасность
Программируемость
Текст
Строка:
Охладитель
?
Горизонтальное выравнивание
Влево
По центру
Вправо
Вертикальное выравнивание
Вверх
По центру
Вниз
Шрифт...
Стандартный
OK
Отмена

б)

Рис. 29

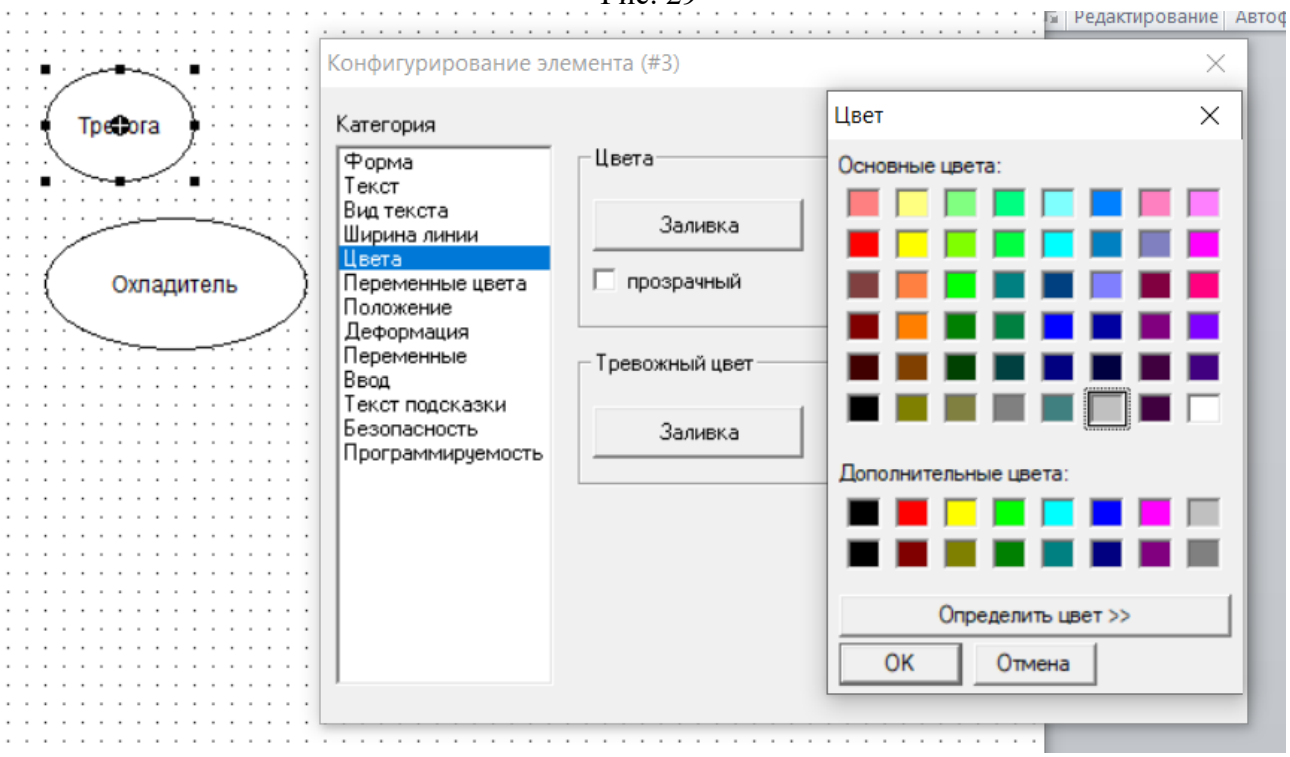
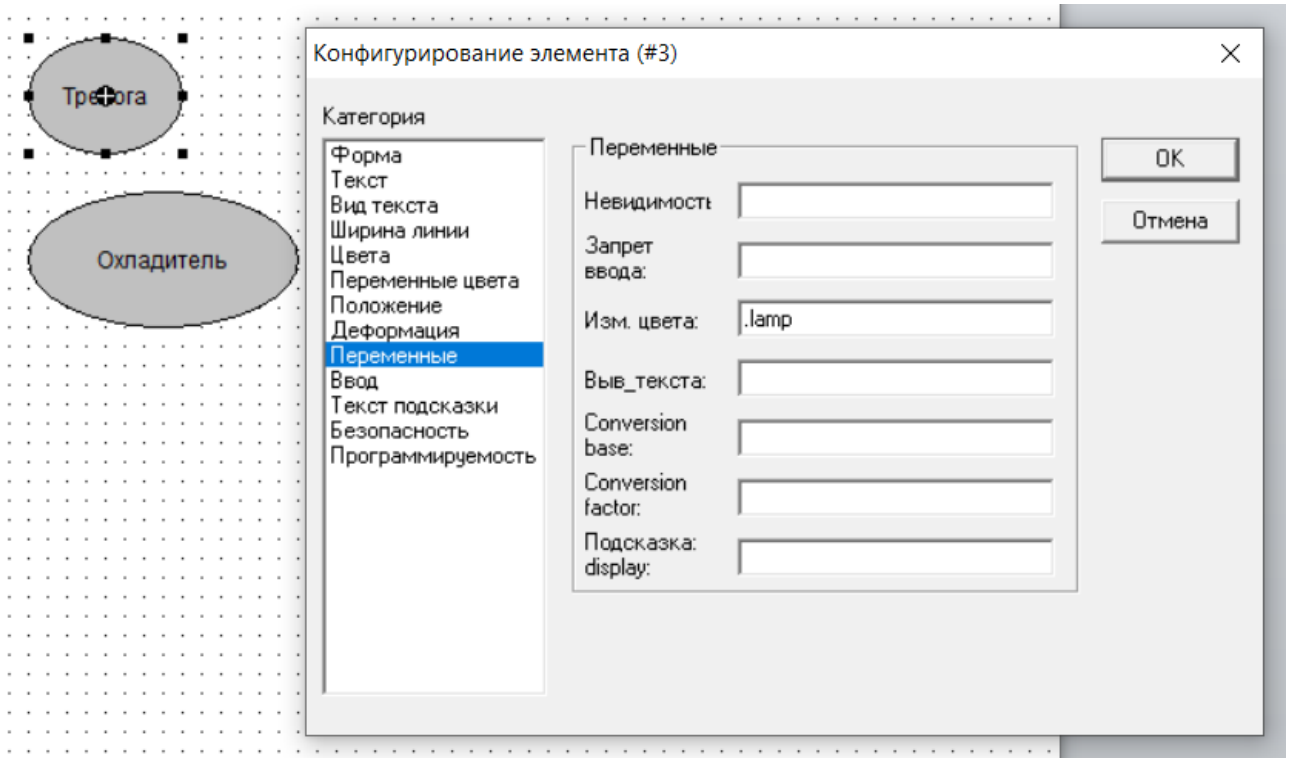
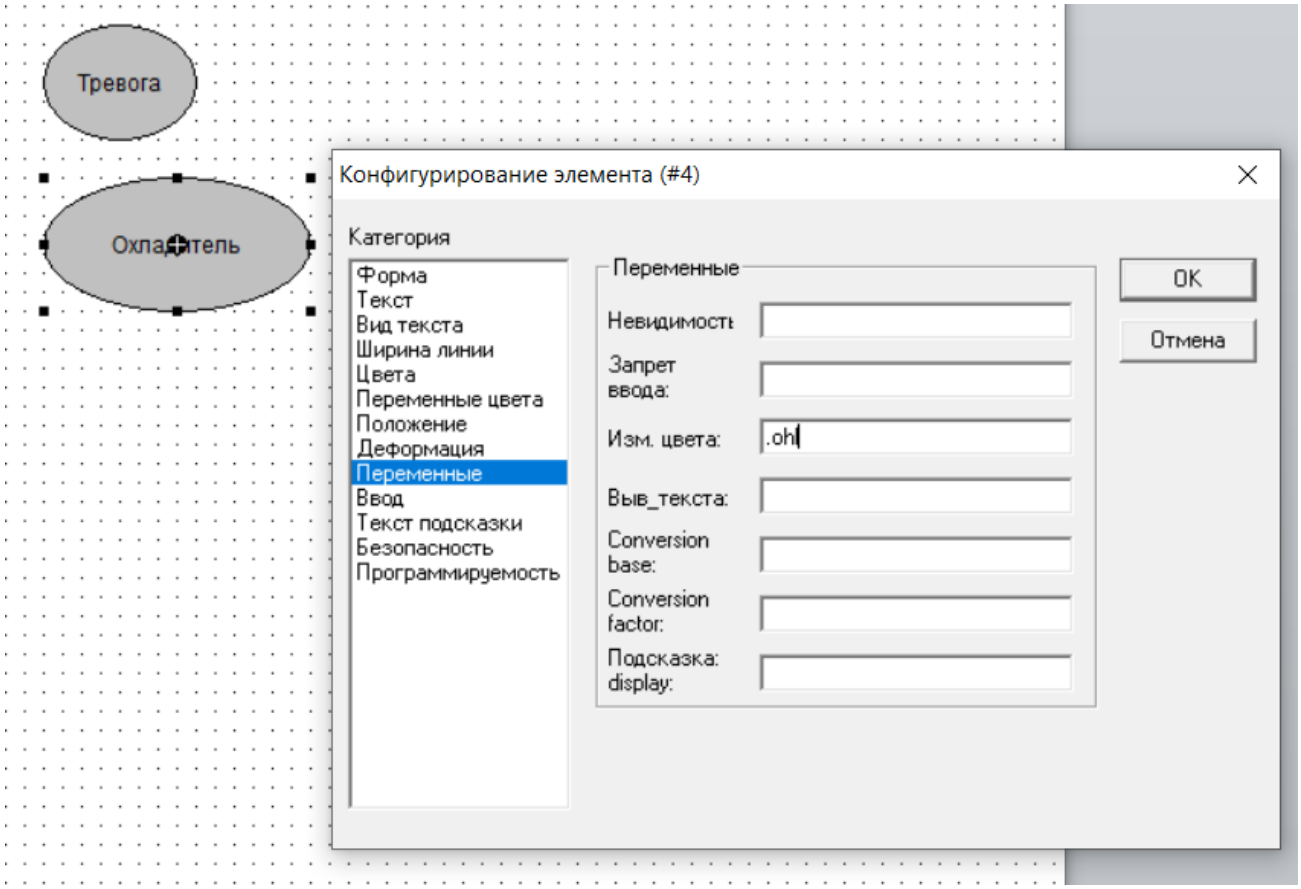

Редактирование
Конфигурирование элемента (#3)
Тревога
Охладитель
Категория
Форма
Текст
Вид текста
Ширина линии
Цвета
Переменные цвета
Положение
Деформация
Переменные
Ввод
Текст подсказки
Безопасность
Программируемость
Цвета
Заливка
прозрачный
Тревожный цвет
Заливка
Цвет
Основные цвета:
Дополнительные цвета:
Определить цвет >>
OK
Отмена

Рис. 30



а)



б)

Рис. 31

13. Для визуализации кнопки сброса используем элемент «Кнопка» (рис. 32). В окне конфигурирования элемента в категории «Текст» пишем «Сброс» (рис. 33). В категории «Ввод» для элемента «Переменная-кнопка» с помощью клавиши F2 выбираем глобальную переменную «sbr» (рис. 34).

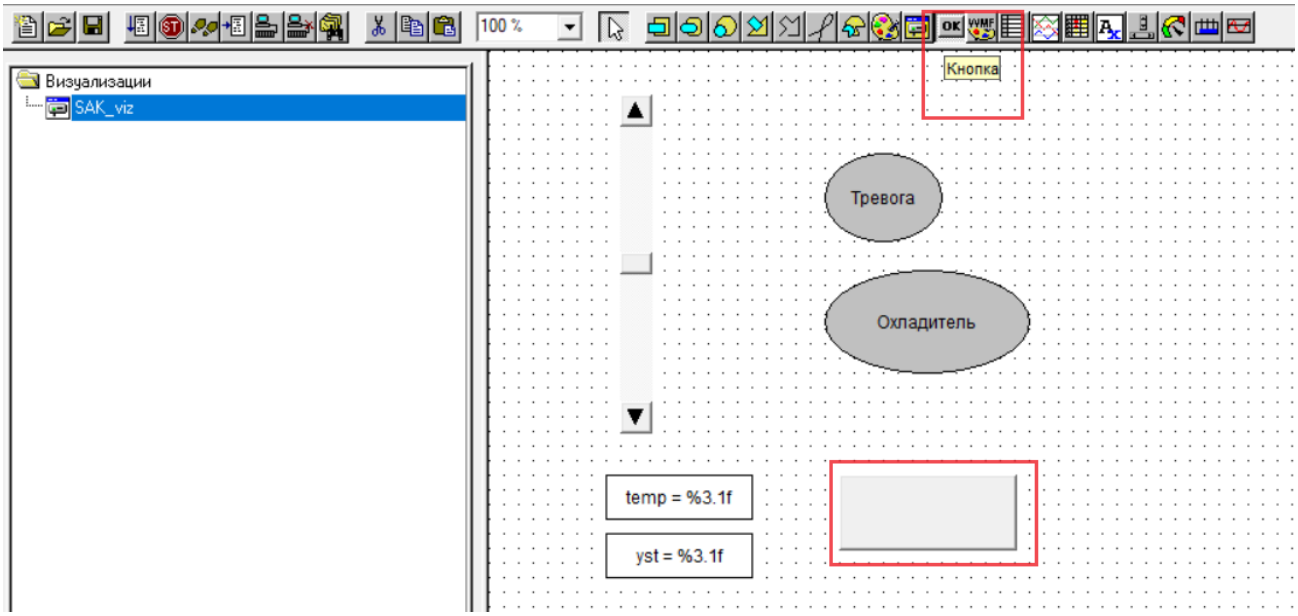

100 %
Кнопка
Визуализации
SAK_viz
Тревога
Охладитель
temp = %3.1f
yst = %3.1f

Рис.32

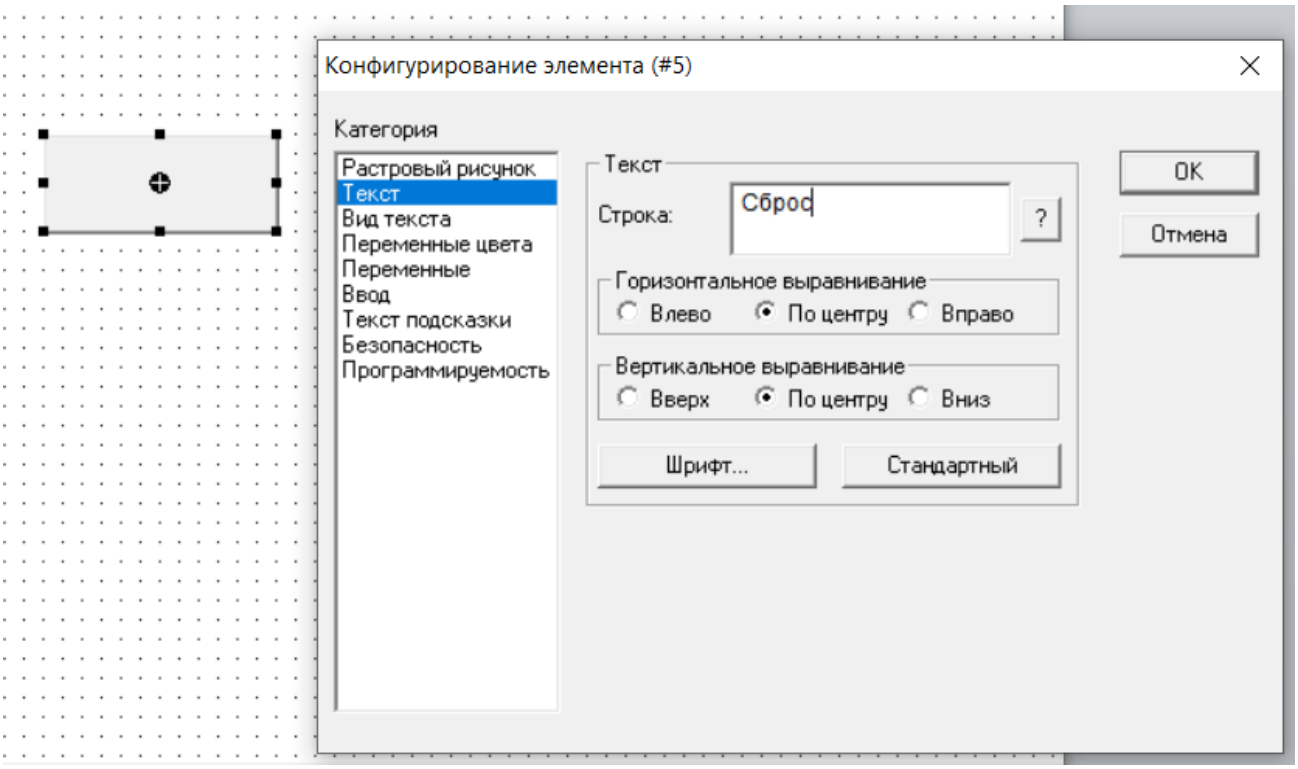

Конфигурирование элемента (#5)
Категория
Растровый рисунок
Текст
Вид текста
Переменные цвета
Переменные
Ввод
Текст подсказки
Безопасность
Программируемость
Текст
Строка:
Сброс
?
Горизонтальное выравнивание
Влево
По центру
Вправо
Вертикальное выравнивание
Вверх
По центру
Вниз
Шрифт...
Стандартный
OK
Отмена

Рис. 33

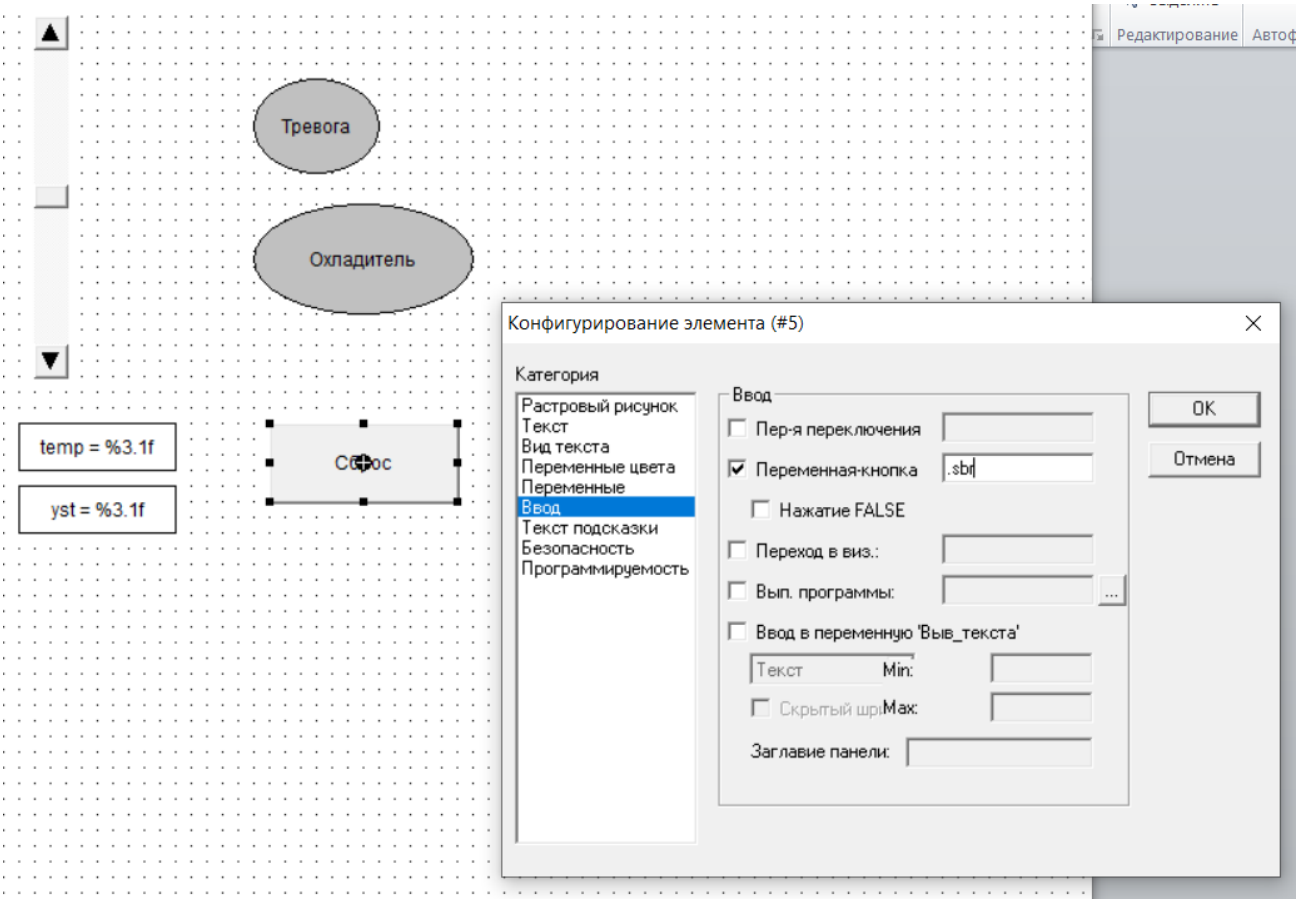


Рис. 34

14. Сохраняем проект. В меню «Онлайн» выбираем «Режим эмуляции», далее «Подключение» и «Старт» (рис. 35).

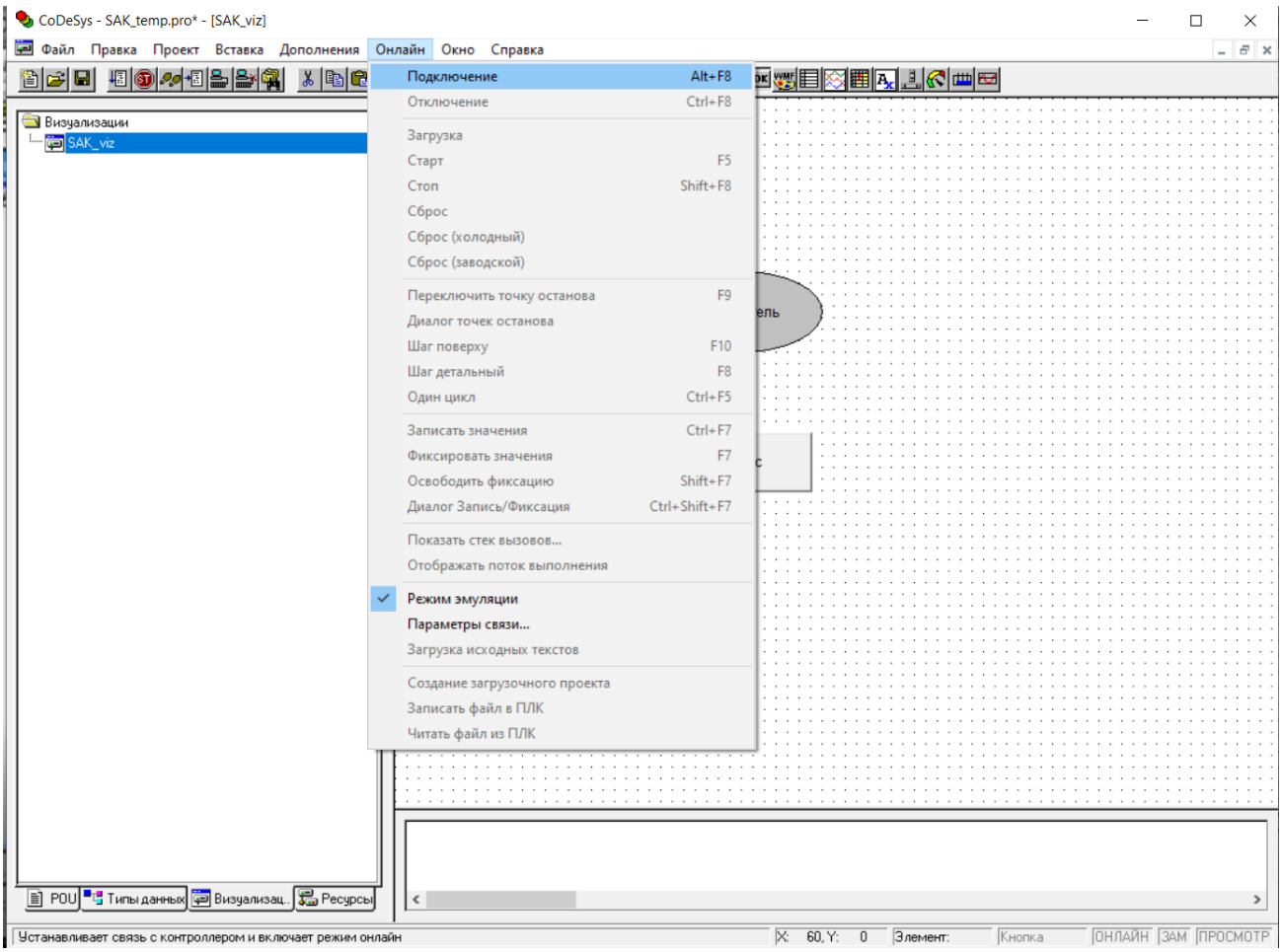


Рис. 35

15. Проверяем работу проекта. Перемещаем движок ползунка на значение выше уставки. Через 3 секунды лампа сигнализации начинает мигать и запускается в работу охладитель (рис. 36). Если текущее значение температуры выше заданного, то нажатие на кнопку «Сброс» не приводит к отключению сигнализации и охладителя. Если температура стала ниже уставки, но кнопка «Сброс» не нажата, то сигналазиация и охладитель продолжают работу. Если же при этом нажать на кнопку «Сброс», то лампа сигнализации прекращает мигать, гаснет, а охладитель выключается (рис. 37).

Рис. 36

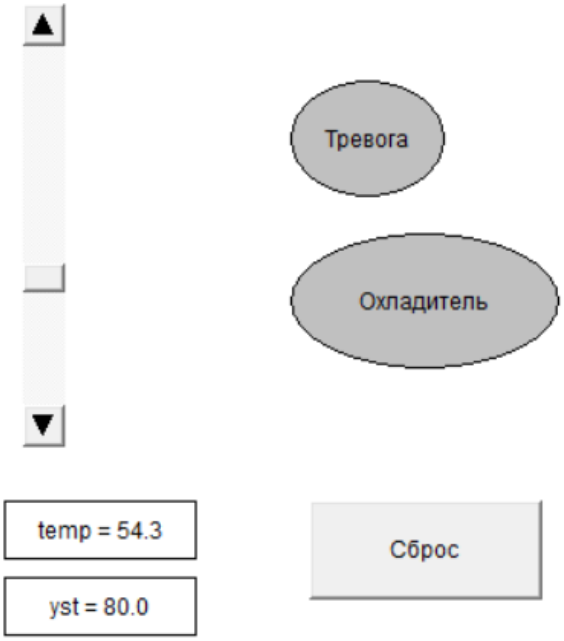


Рис. 37

Содержание отчета

1. Скрин окна главной программы PLC_PRG с реализованной системой автоматического контроля температуры установки, аналогично рис. 19.
2. Описание работы системы автоматического контроля температуры установки.
3. Скрины визуализации проекта, аналогично рис. 36 и 37.

Практическая работа № 4
Изучение основ языка программирования SFC в среде CoDeSys

Порядок выполнения

1. Создадим новый проект без конфигурации контроллера (выбрать тип None). Выберем язык программирования SFC (рис. 1).

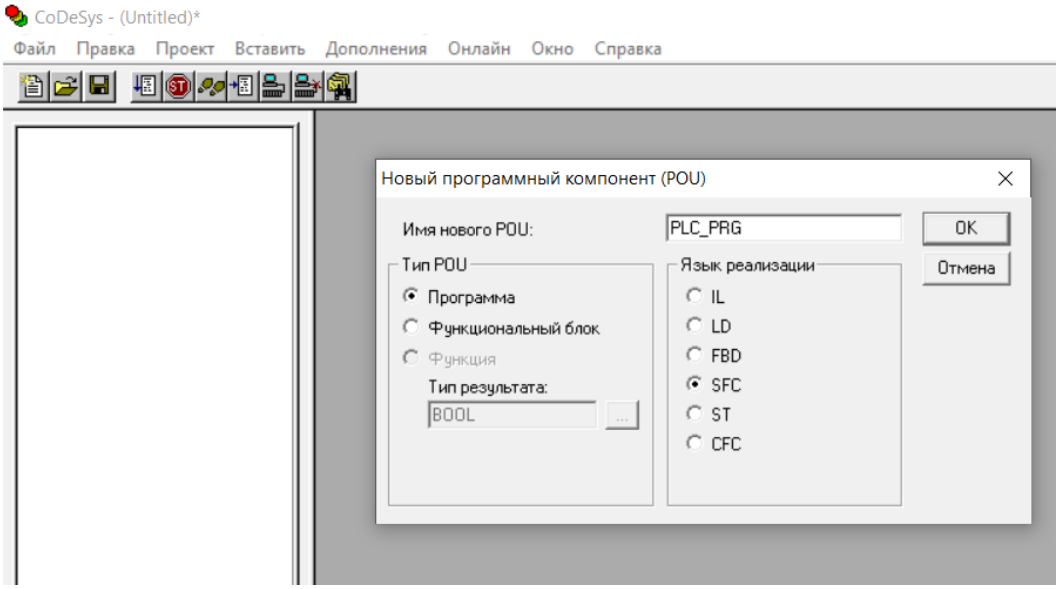
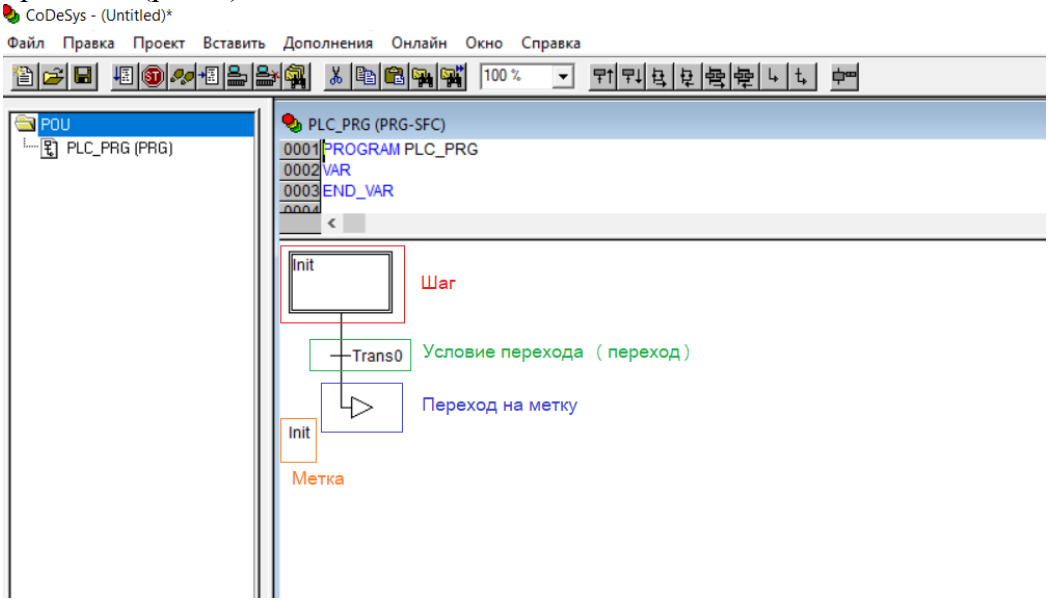


Рис. 1

2. Разберём основные элементы окна программирования SFC при начале работы с новым проектом (рис. 2).



Изначально программа зациклена сама на себя.

Панель инструментов языка SFC представлена на рис. 3. Её возможности:

- добавление шага перехода (сверху и снизу);
- добавление альтернативной ветви (справа и слева);
- добавление параллельной ветви (справа и слева);
- добавление безусловного перехода;
- добавление условного перехода;
- использование МЭК-шагов.

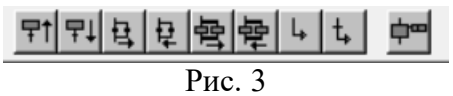


Рис. 3

3. Добавим новые шаги относительно перехода Trans0. Для этого переместим курсор на переход Trans0 и выберем на панели инструментов сначала команду Шаг- переход (сверху)

(добавится шаг Step2), а затем команду Шаг- переход (снизу) (добавится шаг Step3) (рис. 4). В SFC шаги всегда строятся относительно какого-либо перехода.

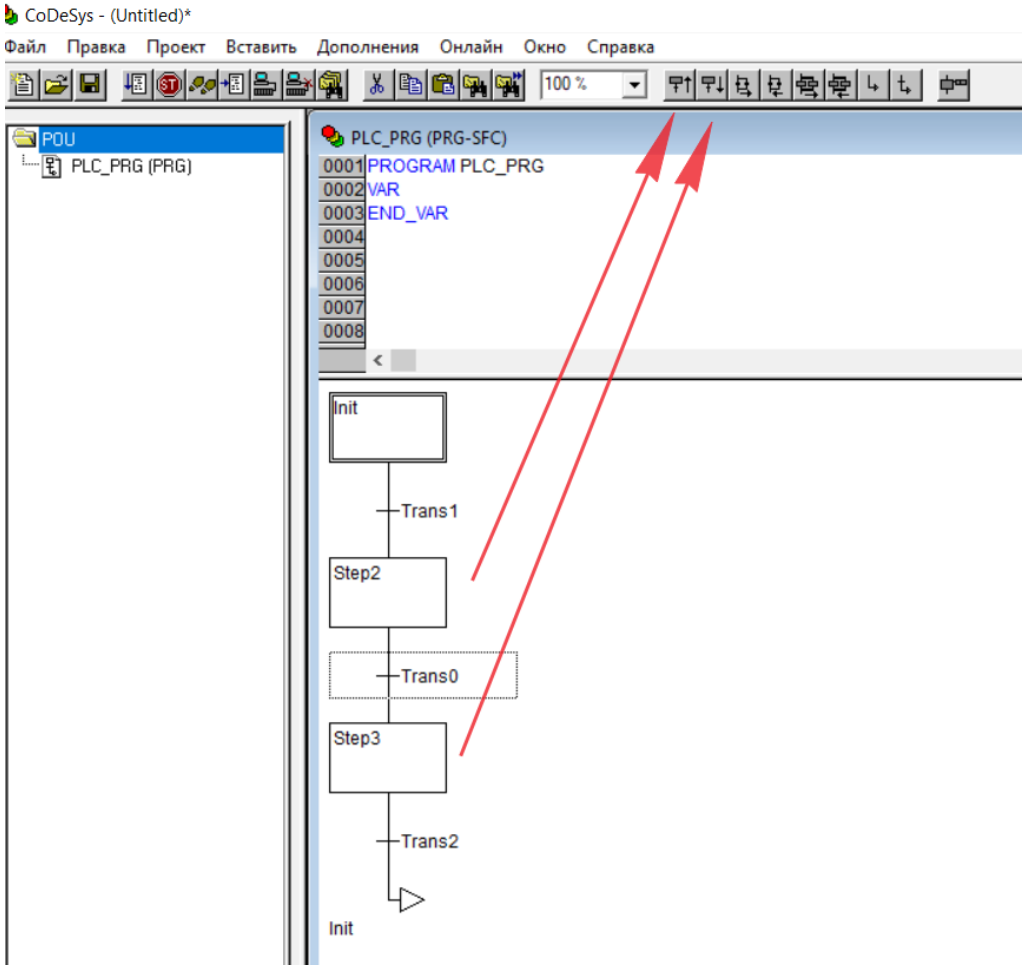


Рис. 4

Каждый шаг имеет своё имя, которое можно изменять по своему усмотрению, например, как показано на рис. 5.

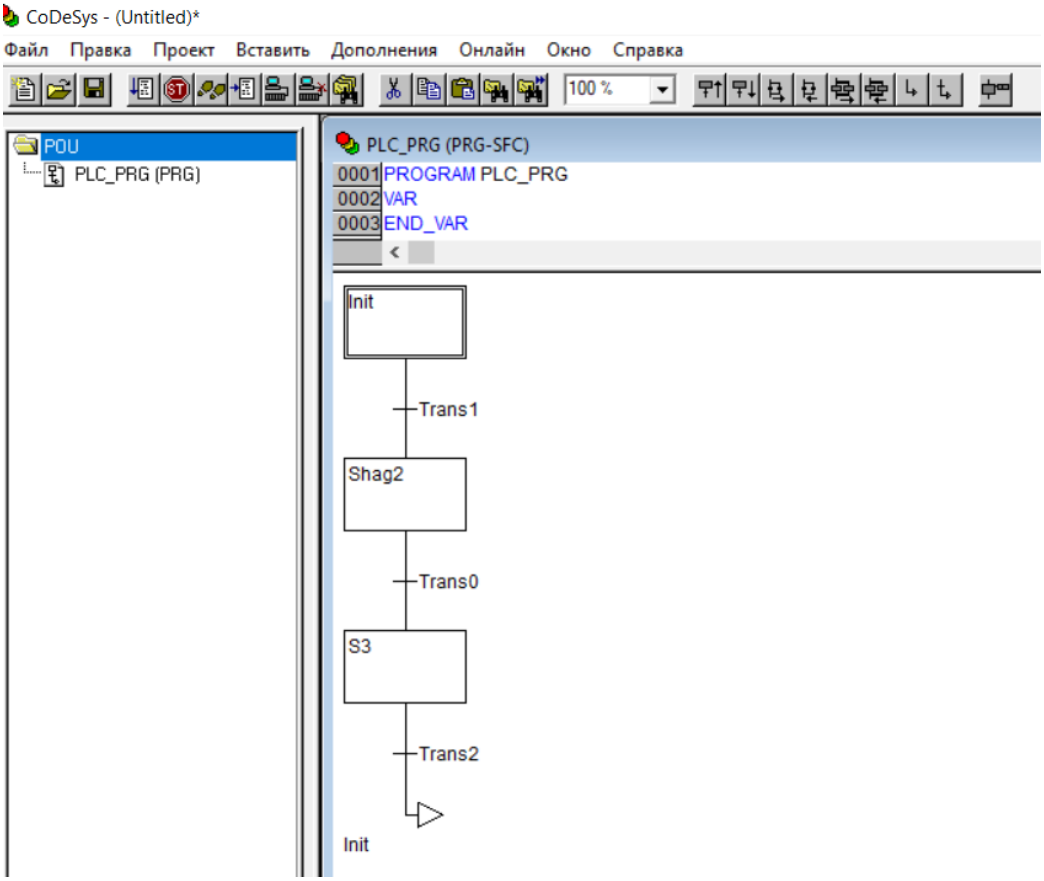


Рис. 5

4. Разберём возможности построения различных логических структур на языке SFC.
   4.1. Добавим альтернативную ветвь справа от перехода Trans1. Для этого выделяем его, перемещая курсор, и выбираем команду Альтернативная ветвь (справа). Справа от перехода Trans1 появится еще один переход Trans3. Затем относительно Trans3 можно добавить вниз шаг Step4 и относительно перехода Trans1 вниз добавим шаг Step5 (рис. 6).

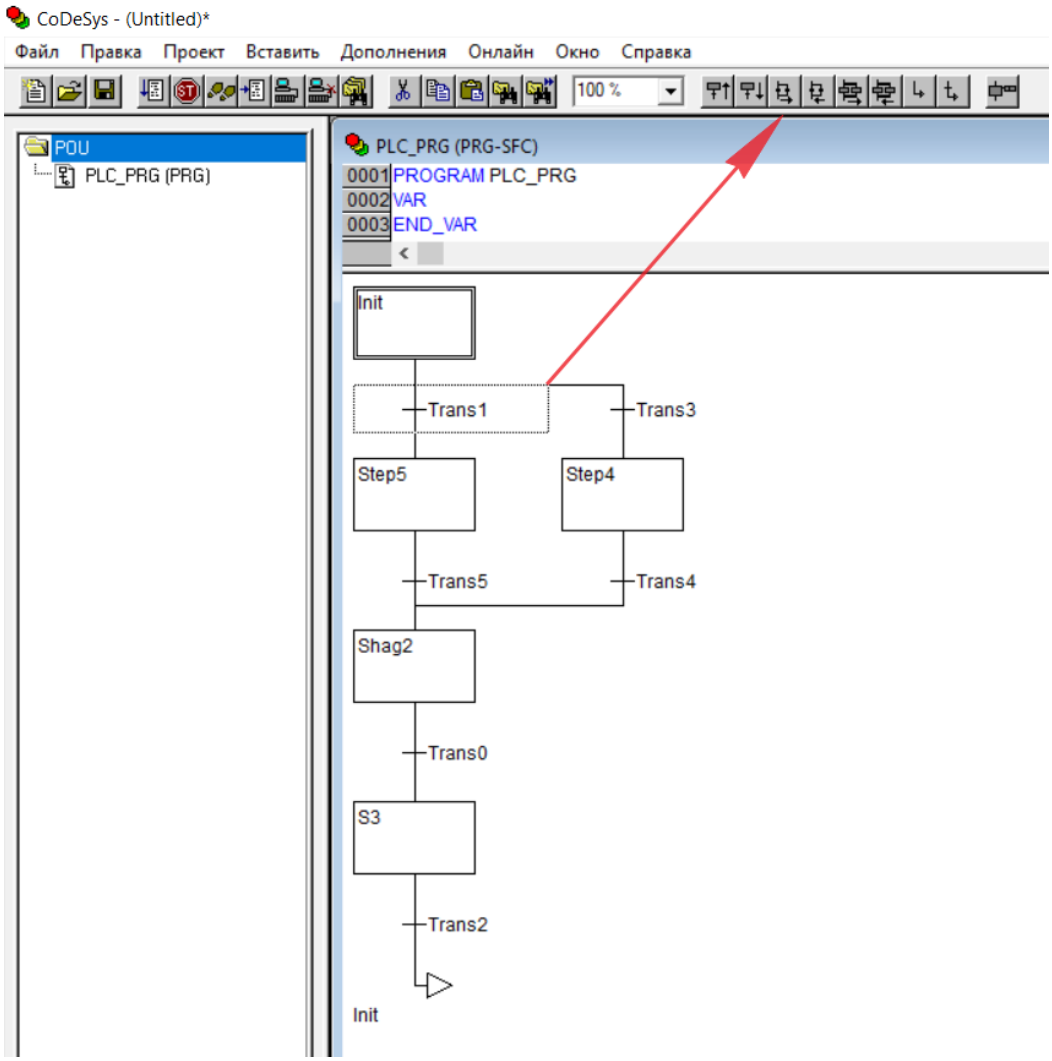


Рис. 6

4.2. Добавим альтернативную ветвь между переходами Trans1 и Trans0. Для этого нужно нажать клавишу Shift и по очереди кликнуть на эти переходы, а затем выбрать команду Альтернативная ветвь (справа) (рис. 7).

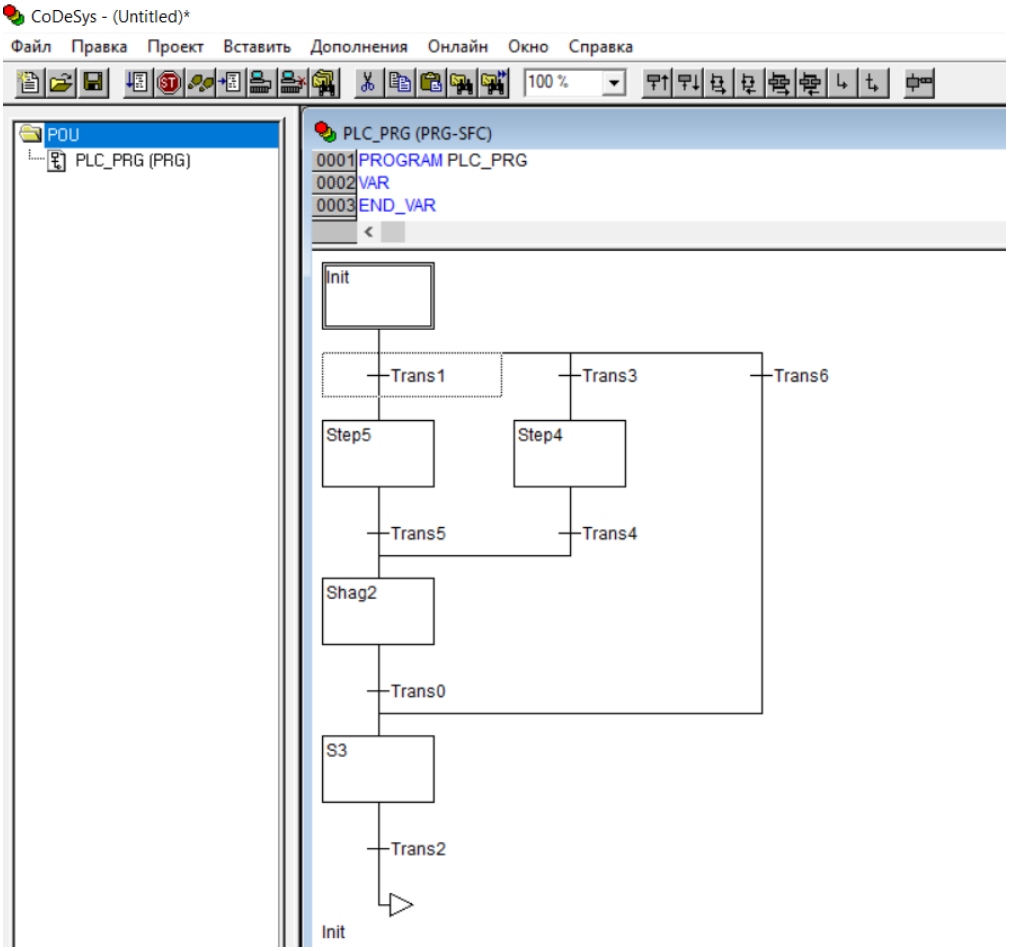


Рис. 7

4.3. Добавим шаг, параллельный шагу Step4. Для этого переведем курсор на шаг Step4 и выберем команду Параллельная ветвь (справа) (рис. 8). Шаги Step4 и Step6 будут выполняться параллельно в зависимости от условия Trans3.

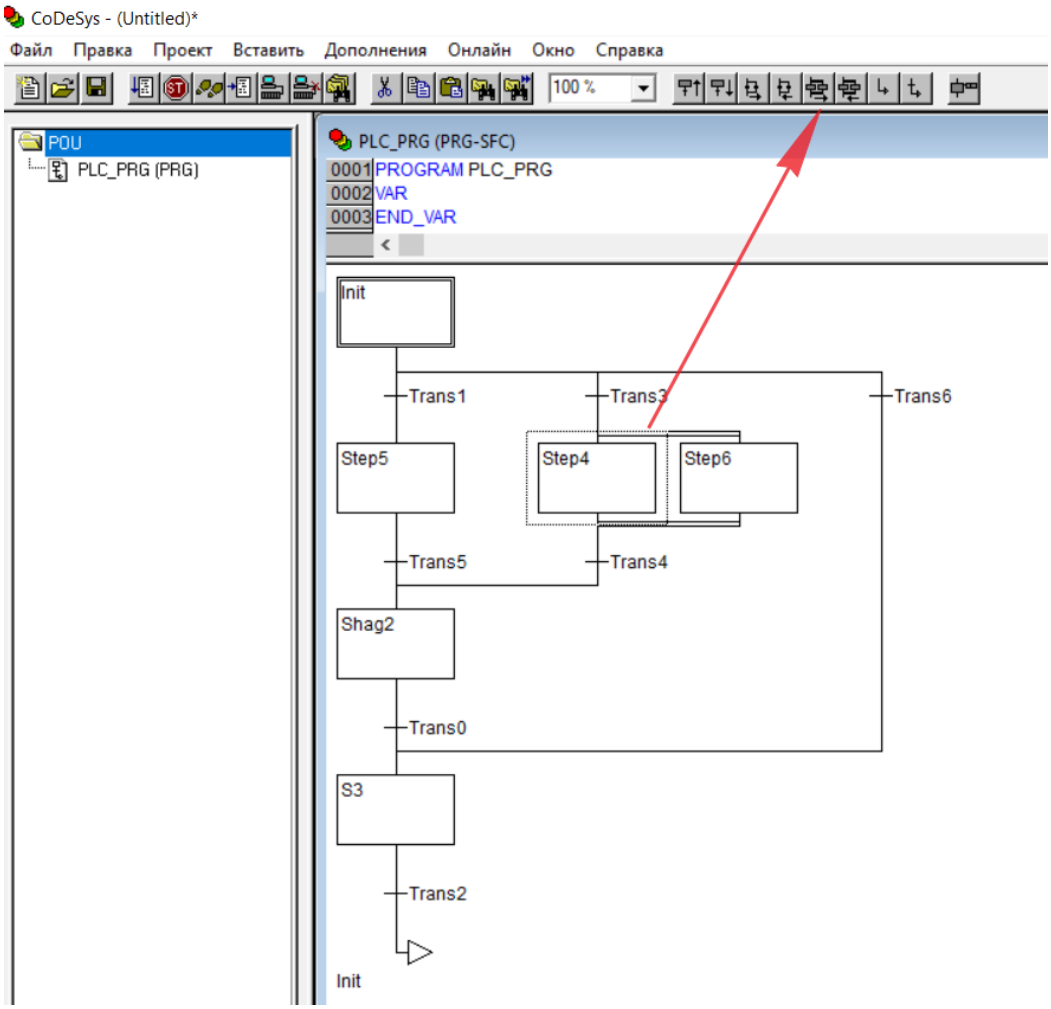


Рис. 8

4.4. Добавим после шага Step6 условие. Для этого выделяем его курсором и выбираем команду условного перехода (рис. 9). Теперь после шага Step6 будет проверяться условие Trans7, и в случае его истинности, будет осуществлен переход на шаг S3.

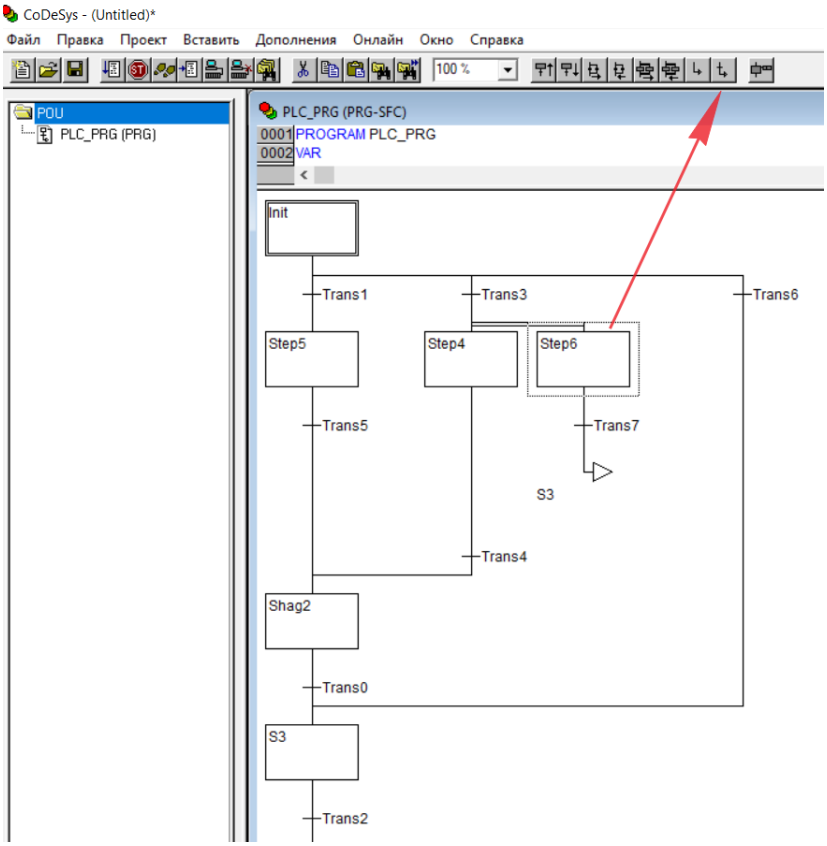


Рис. 9

4.5. Добавим после перехода Trans6 шаг Step7, а затем для этого шага выберем команду Безусловный переход, чтобы добавился переход Trans8. И сделаем так, чтобы этот переход вёл на метку Step4 (изменим название метки). Результат показан на рис. 10. Теперь после выполнения шага Step7 будет проверено условие Trans8, и в случае его истинности, будет осуществлен переход на шаг Step4.

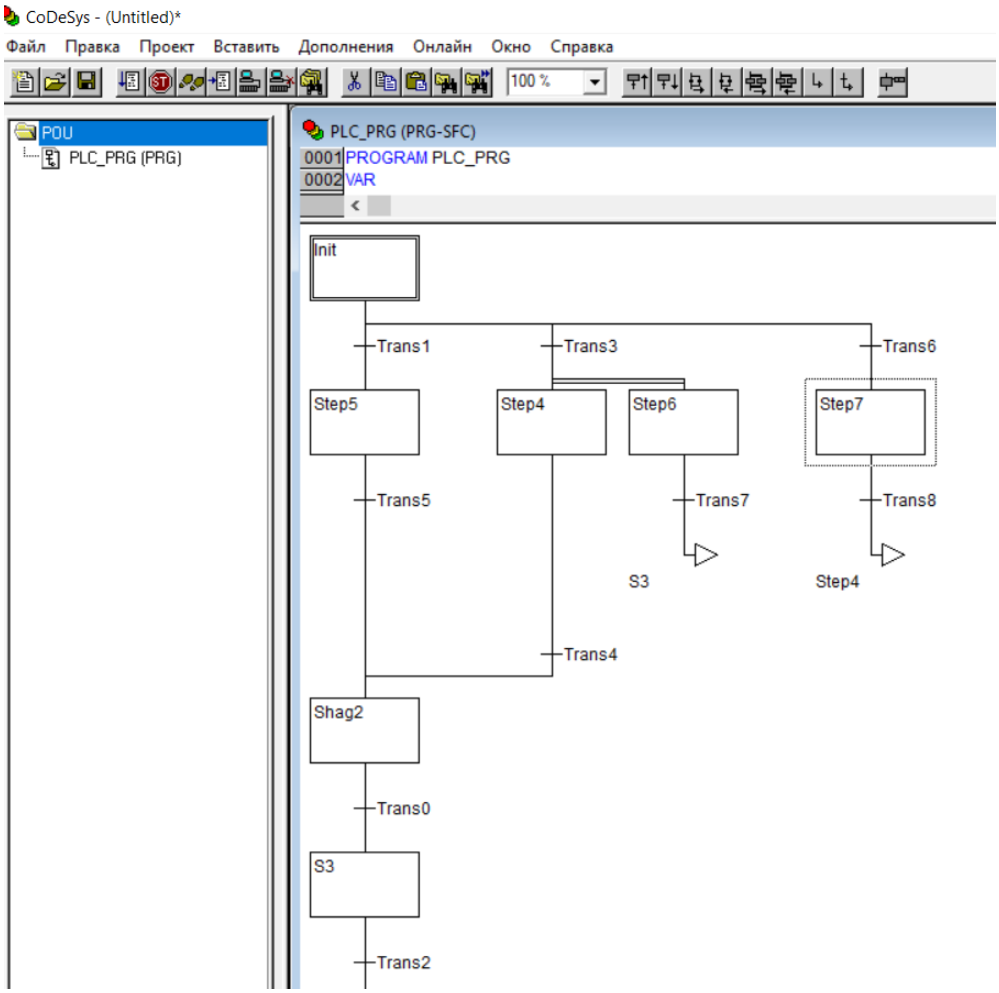


Рис. 10

4.6. Удалим все наши логические построения, чтобы схема имела вид как на рис. 11. Если в параллельной или альтернативной ветви нет условия перехода, то ее можно удалять клавишей Delete, выбирая курсором соответствующий шаг. Если же после шага есть условие перехода, то их нужно сначала выделить вместе с помощью клавиши Shift, и только после этого можно будет удалить через клавишу Delete. Если после перехода есть метка, то их тоже нужно сначала вместе выделить, а потом только удалить.

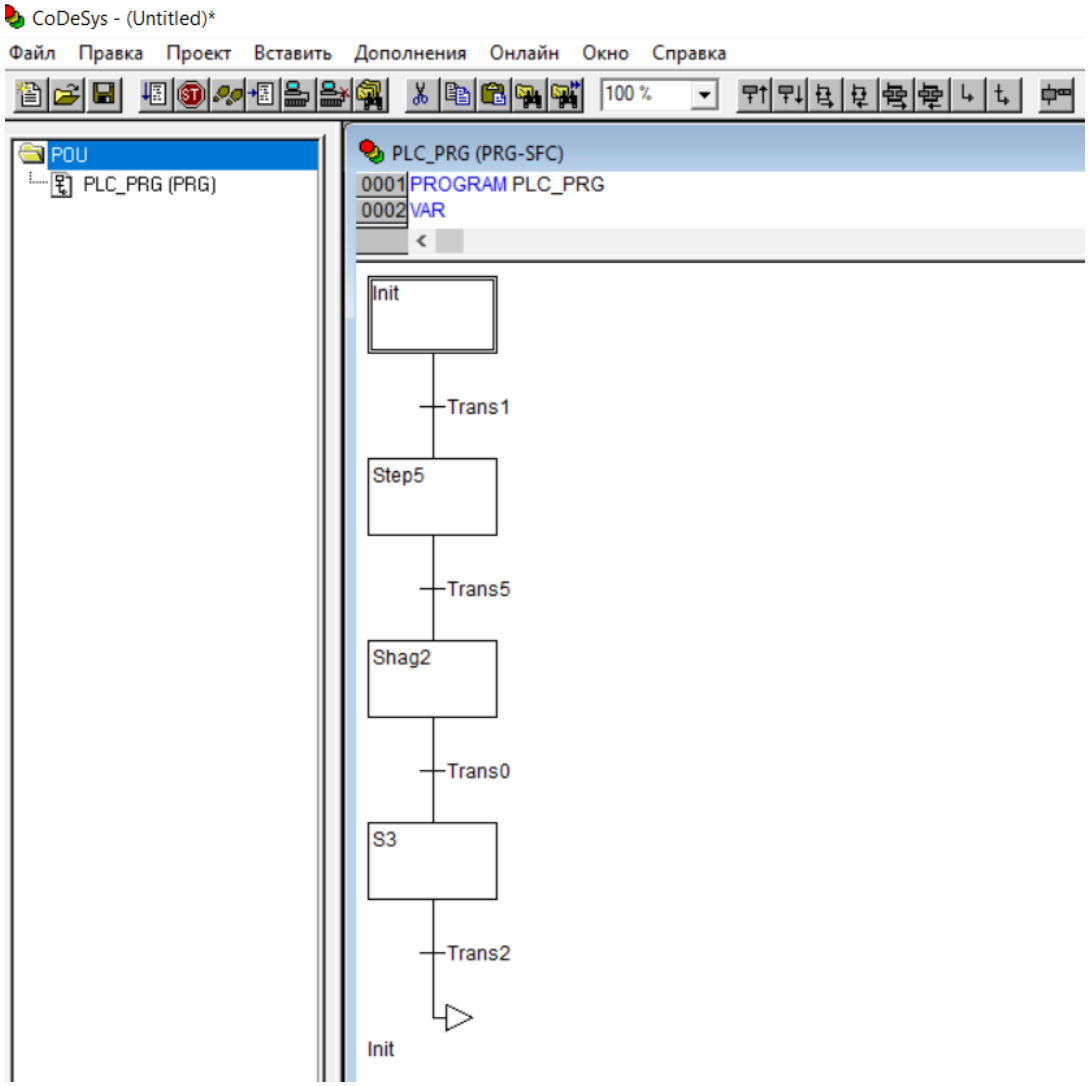


Рис. 11

5. Разберём, как создаются действия внутри шагов и условия перехода.
   5.1. Для перехода Trans1 запишем простое логическое условие «X>10». При этом появится окно объявления переменной, выберем тип INT для X (рис. 12). Теперь шаг Step5 будет выполнен, только если X>10.

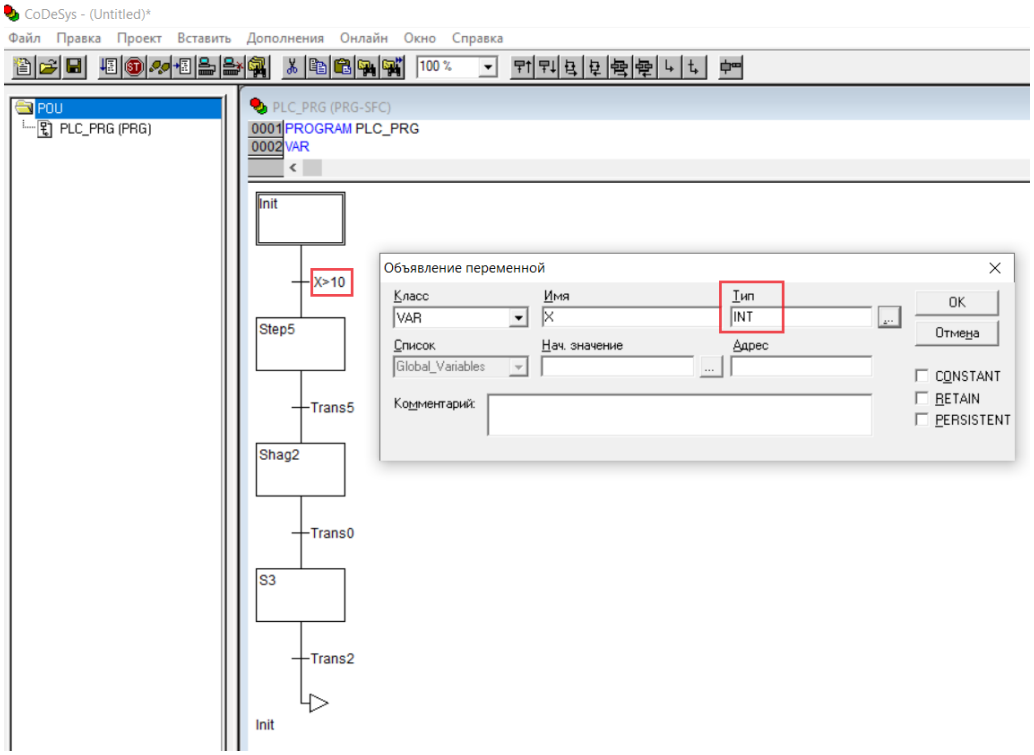


Рис. 12

5.2. Чтобы прописать действие внутри конкретного шага, нужно дважды кликнуть по нему и выбрать язык, на котором будет прописано это действие. Например, для шага Step5 создадим простое действие на языке LD (рис. 13). Новые переменные объявляем сразу в процессе их создания, тип выбираем BOOL.

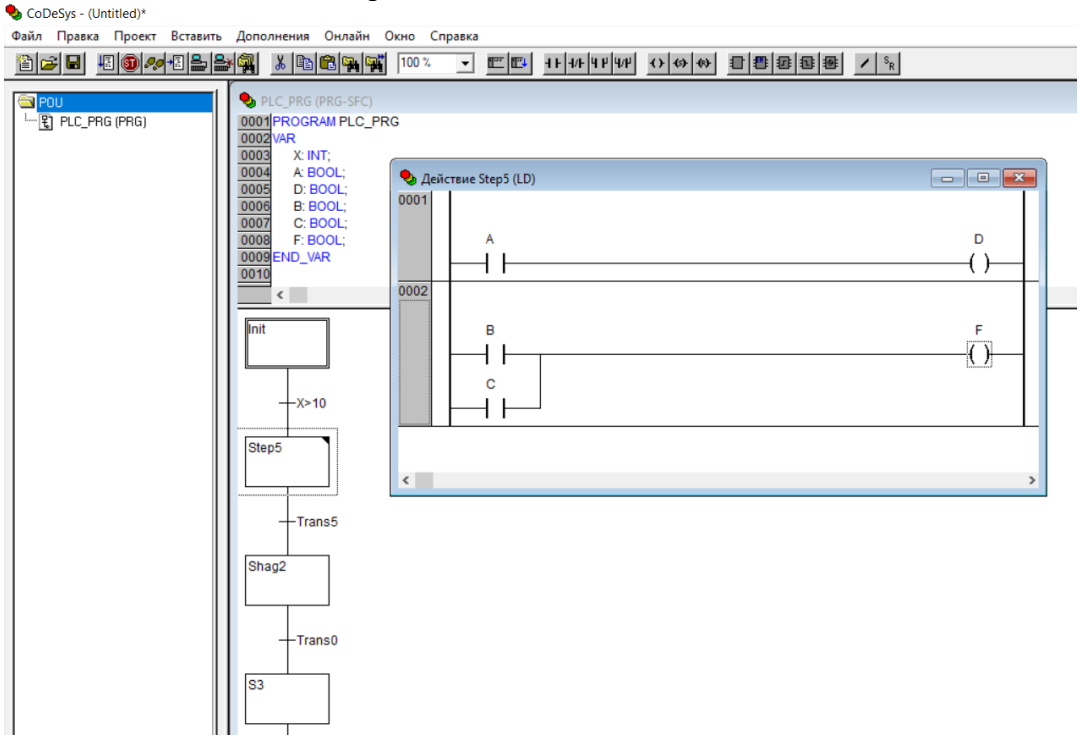


Рис. 13

Действия являются частью программы, поэтому переменные A, B, C, D и F для действия шага Step5 будут объявлены внутри основной программы PLC_PRG.

После создания действия для шага его можно закрыть. Если нужно будет что-то изменить в действии, то оно также вызывается двойным кликом по соответствующему шагу. Если для шага задано какое-то действие, то в блоке шага появляется небольшой треугольник (справа в верхней части) (рис. 14).

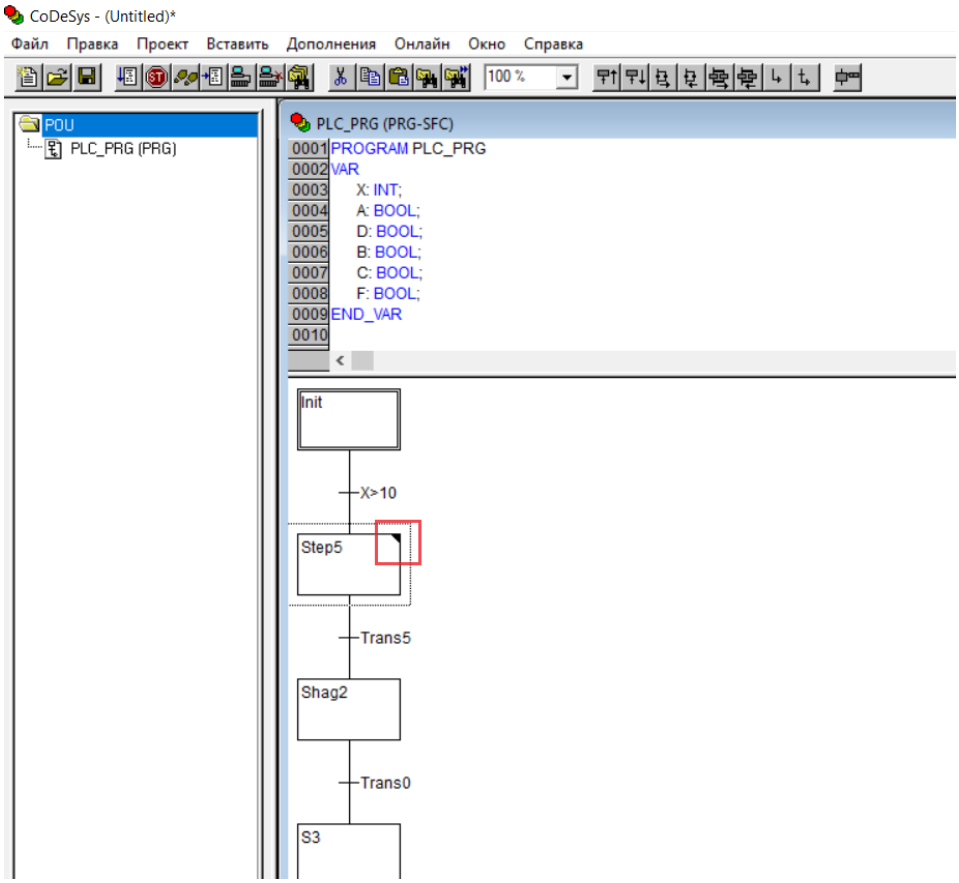


Рис. 14

5.3. Для шага Shag2 создадим действие на языке ST (рис. 15).

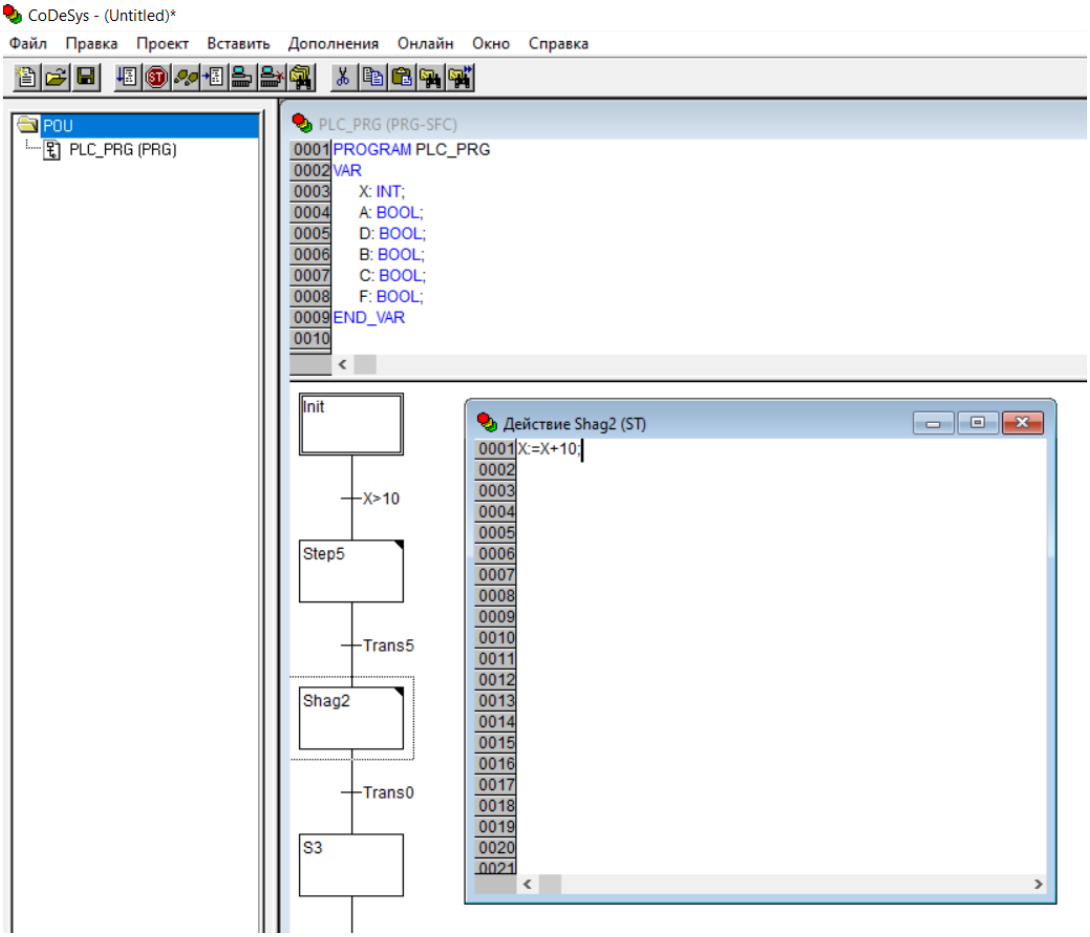


Рис. 15

5.4. Управлять действиями шагов можно с помощью вызова контекстного меню правой кнопкой мыши (рис. 16). В частности, можно посмотреть содержимое действия шага или же полностью его очистить. Например, сейчас очистим действие для Shag2, и создадим для него же новое действие на языке FBD (рис. 17). Потом можно снова очистить шаг Shag2.

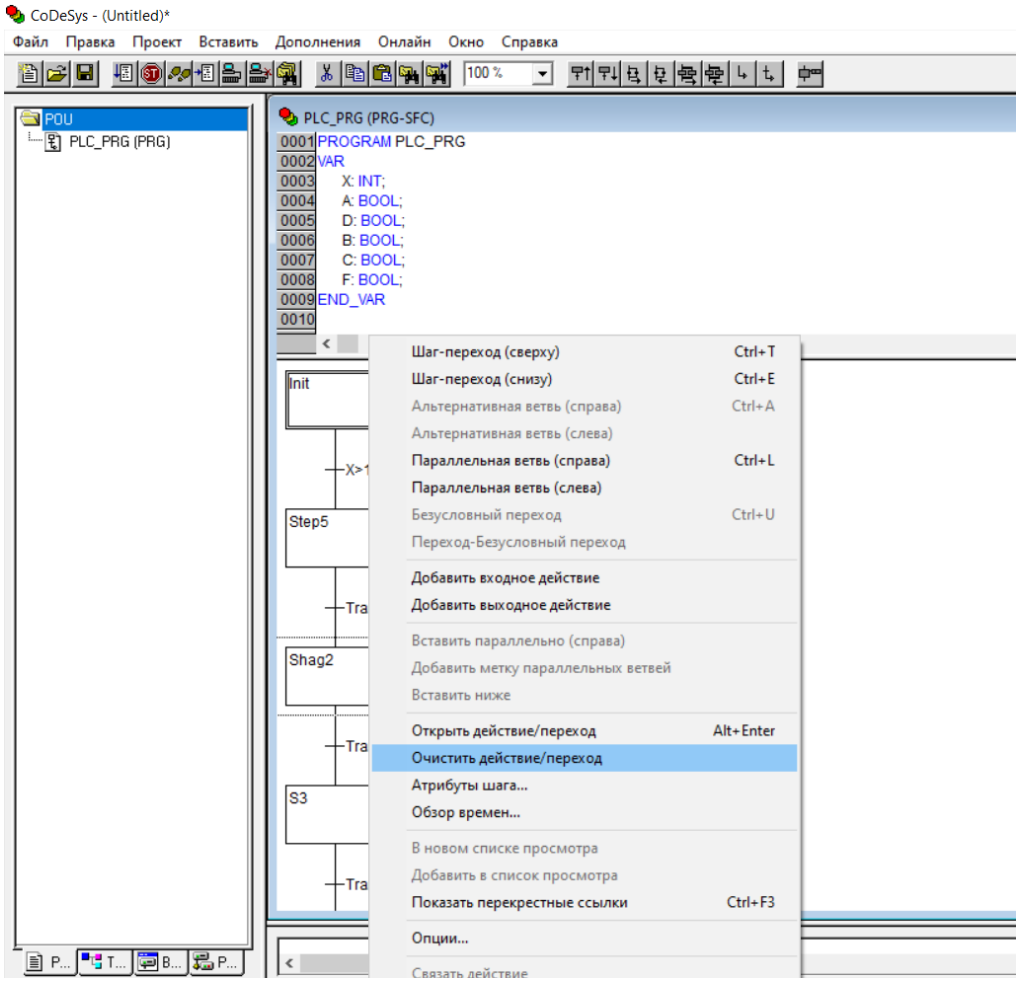


Рис. 16

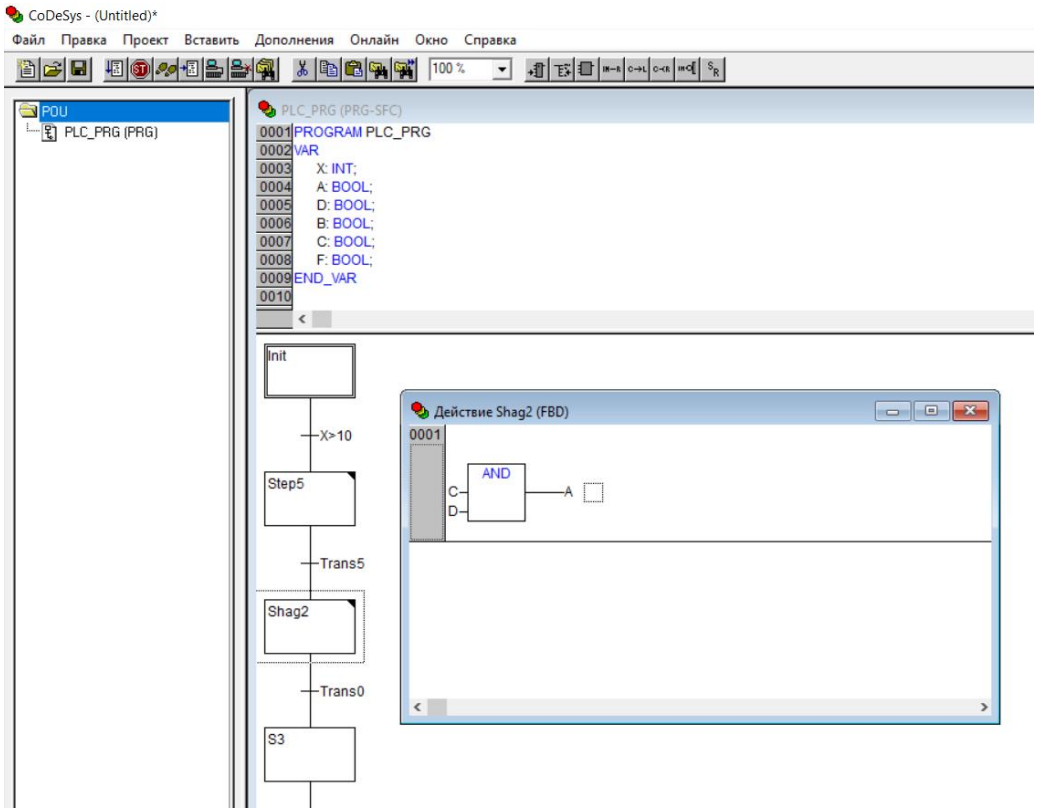


Рис. 17

5.5. Для каждого шага мы можем задать действие, которое будет выполняться внутри него (см. п.5.2, 5.3), а также можем задать с помощью контекстного меню так называемое входное действие для шага, и выходное действие для него же (рис. 18).

Входное действие – это действие, которое выполняется при переходе на данный шаг. Выходное действие – это действие, которое выполняется перед тем, как перейти на следующий шаг.

Добавим входное и выходное действия для шага Step5, выберем язык реализации ST (рис. 19, 20).

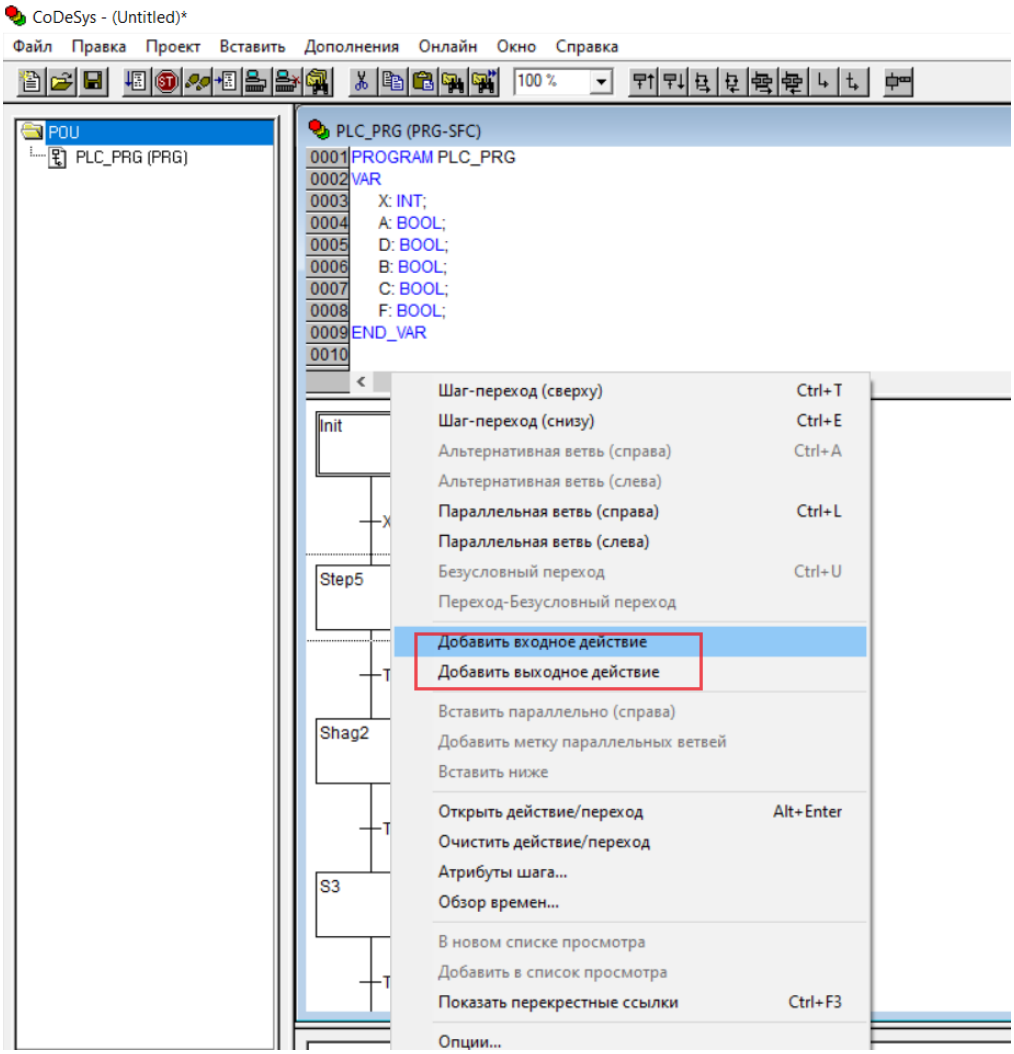

CoDeSys - (Untitled)*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 END_VAR
0010
Init
Step5
Shag2
S3
Шаг-переход (сверху) Ctrl+T
Шаг-переход (снизу) Ctrl+E
Альтернативная ветвь (справа) Ctrl+A
Альтернативная ветвь (слева)
Параллельная ветвь (справа) Ctrl+L
Параллельная ветвь (слева)
Безусловный переход Ctrl+U
Переход-Безусловный переход
Добавить входное действие
Добавить выходное действие
Вставить параллельно (справа)
Добавить метку параллельных ветвей
Вставить ниже
Открыть действие/переход Alt+Enter
Очистить действие/переход
Атрибуты шага...
Обзор времен...
В новом списке просмотра
Добавить в список просмотра
Показать перекрестные ссылки Ctrl+F3
Опции...

Рис. 18

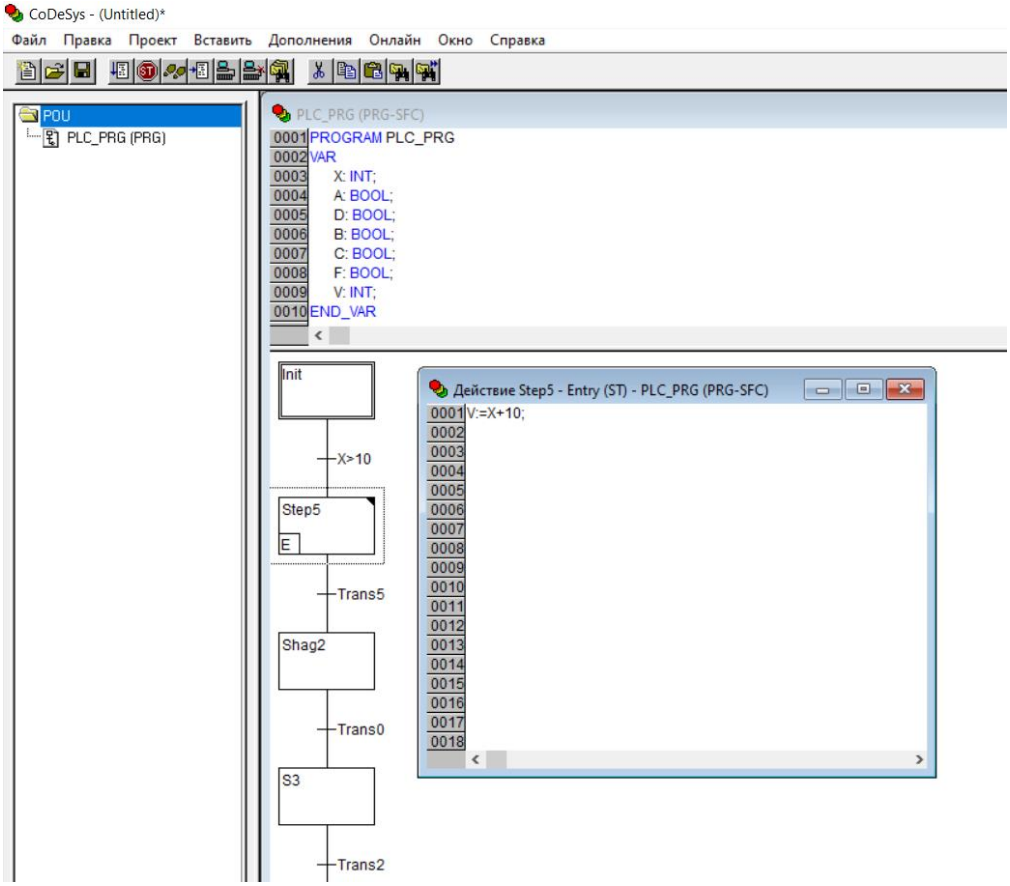

CoDeSys - (Untitled)*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT;
0010 END_VAR
Init
X>10
Step5
E
Trans5
Shag2
Trans0
S3
Trans2
Действие Step5 - Entry (ST) - PLC_PRG (PRG-SFC)
0001 V:=X+10;

Рис. 19

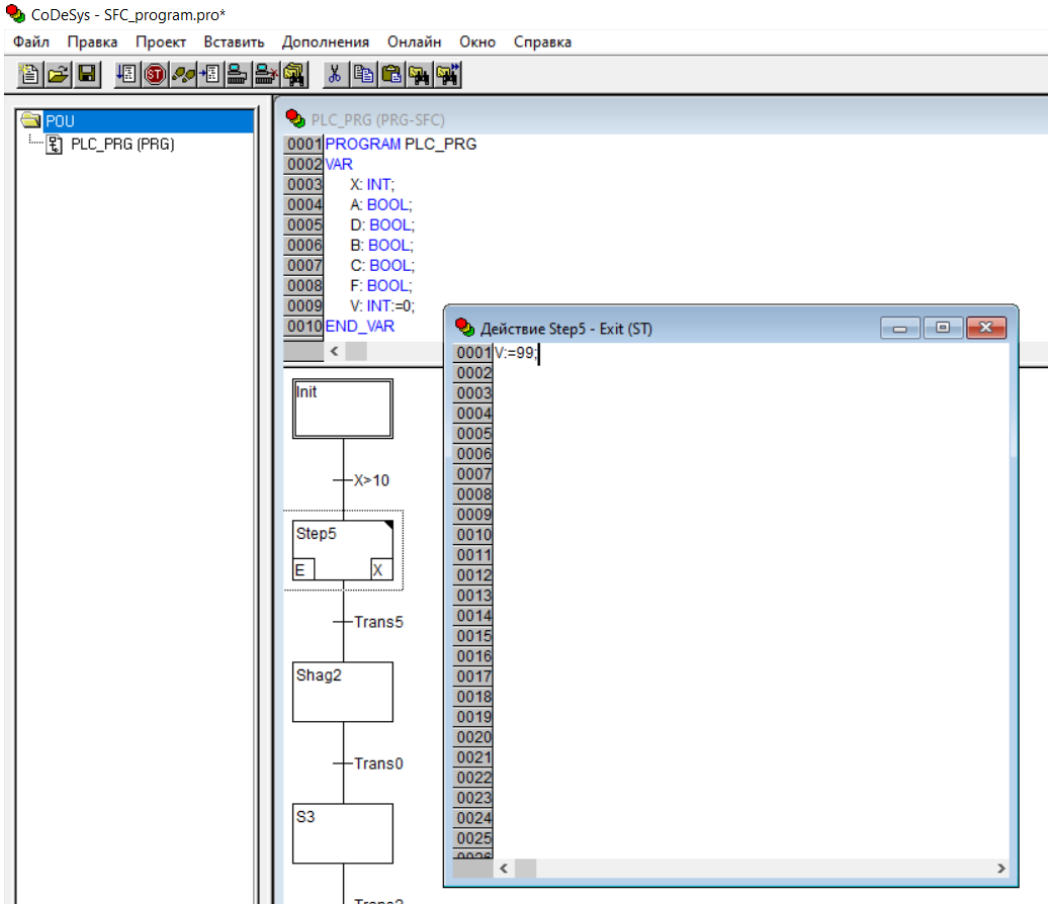


Рис. 20

Послед добавления входного и выходного действия для шага меняется его внешний вид – появляются буквы E (вход) и X (выход).

5.6. Зададим условием перехода для шага Shag2 значение переменной F (рис. 21), которое будет изменяться в результате выполнения действия внутри шага Step5 (см. рис. 13).

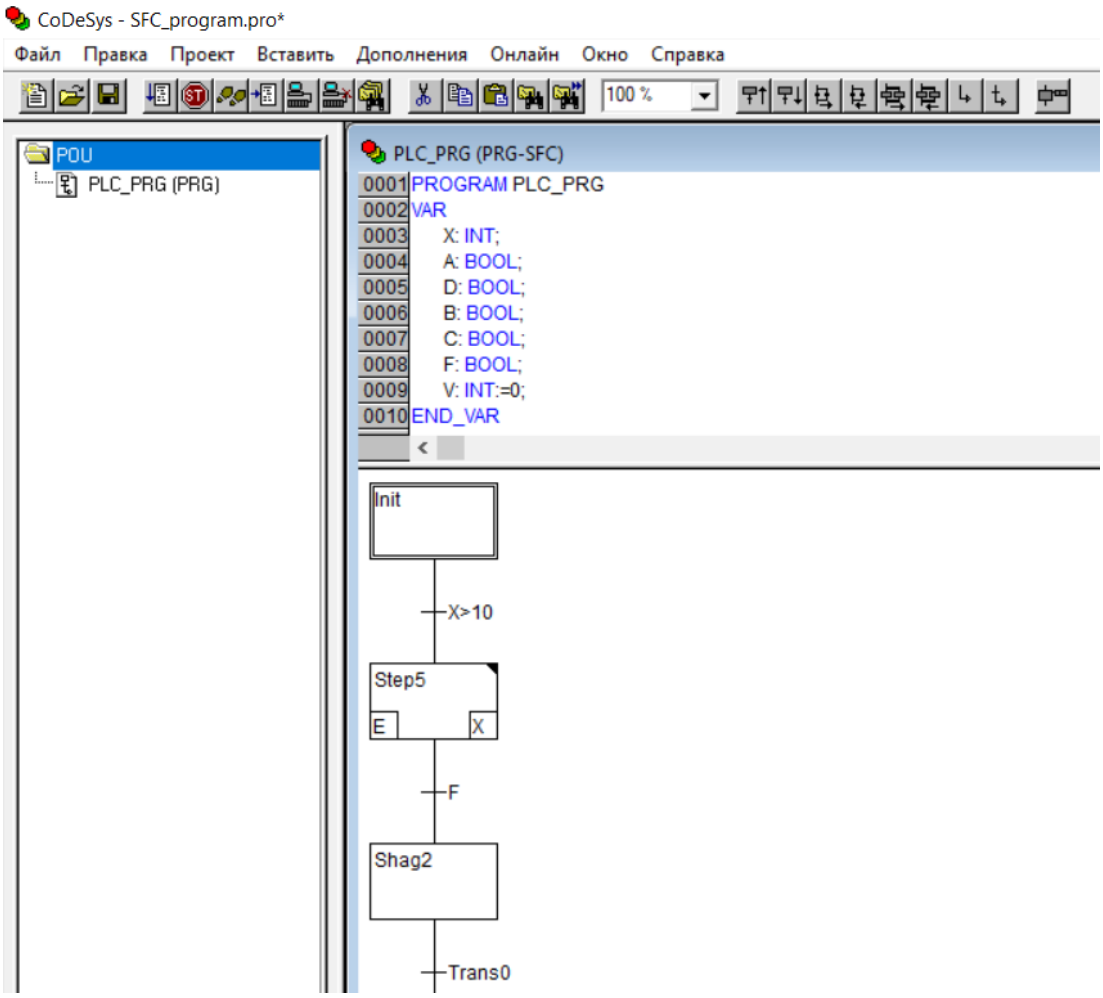


Рис. 21

5.7. Создадим для шага Shag2 действие на языке FBD (рис. 22), а условие перехода от него пропишем как X=0 (рис. 22).

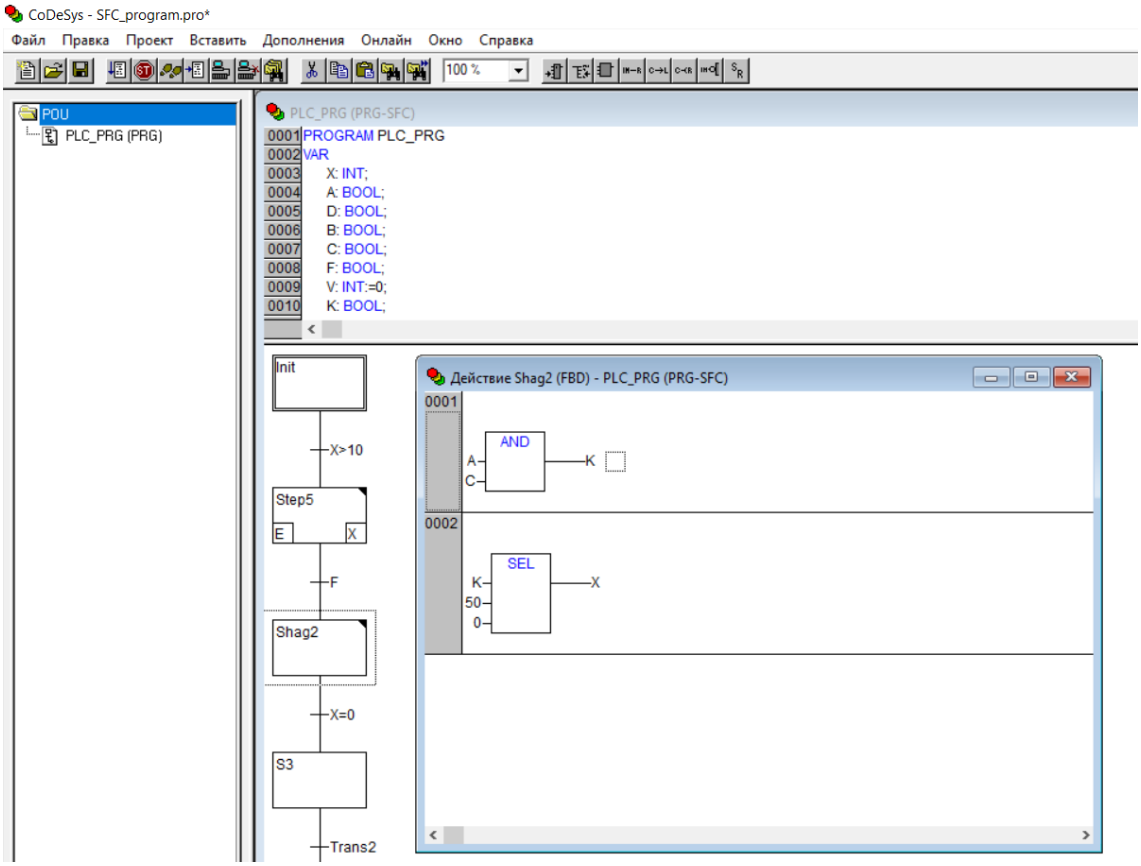


Рис. 22

5.8. Удалим шаг S3 и переход Trans2 (рис. 23)

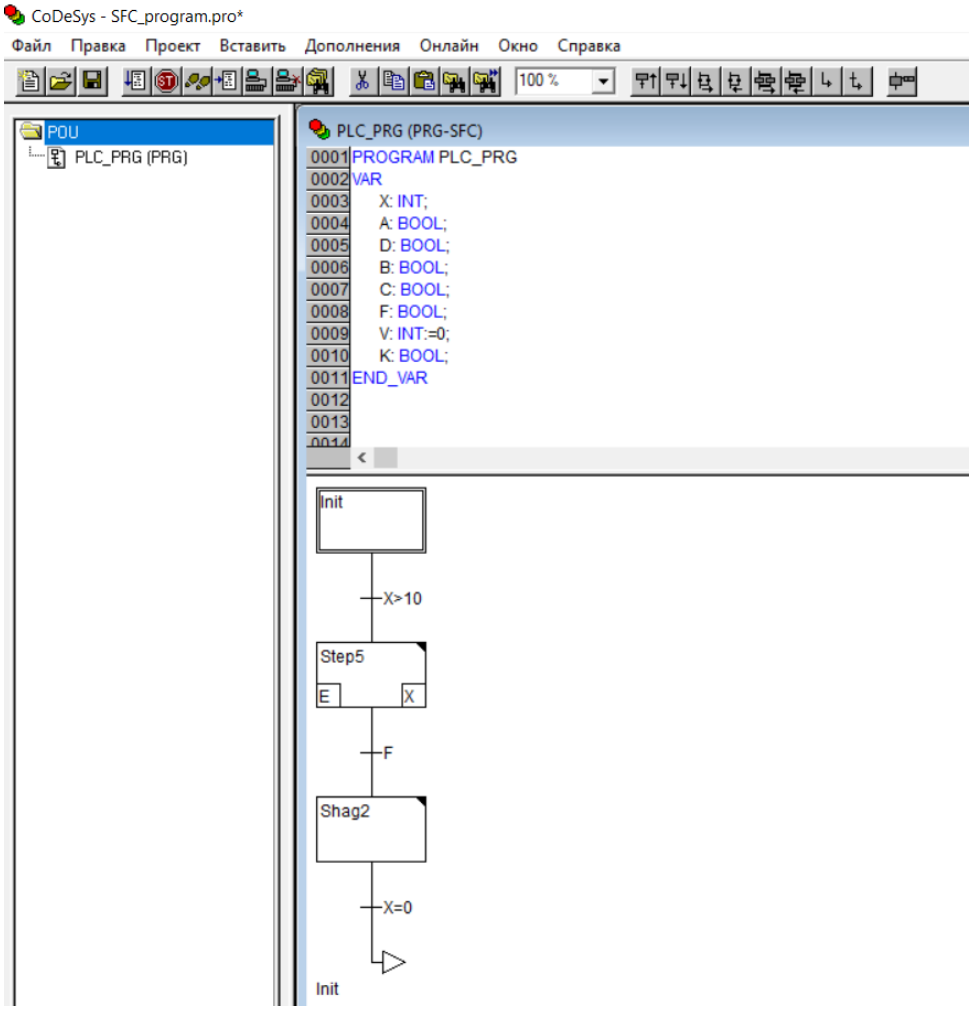


Рис. 23

6. Запускаем программу на исполнение: Онлайн – Подключение; Онлайн – Старт. В языке SFC всегда подсвечивается синим цветом та область, на которой сейчас находится выполнение программы. Так как в начальный момент времени X=0, то программа не может перейти к следующему шагу из-за условия X>10 (рис. 24).

   Если ввести новое значение для переменной X=11, то будет осуществлен переход на следующий шаг Step5 (рис. 25). При этом переменная V приняла значение 21, так как осуществилось входное действие для шага Step5 (рис. 26). Но если после этого в переменную V принудительно записать значение 30, то оно измениться уже независимо от алгоритма входного действия (рис. 27).

   Примечание: чтобы новое значение переменной записалось в память программы, нужно после изменения значения нажать сочетание клавиш CTRL+F7, либо выбрать пункт меню Онлайн – Записать значения.

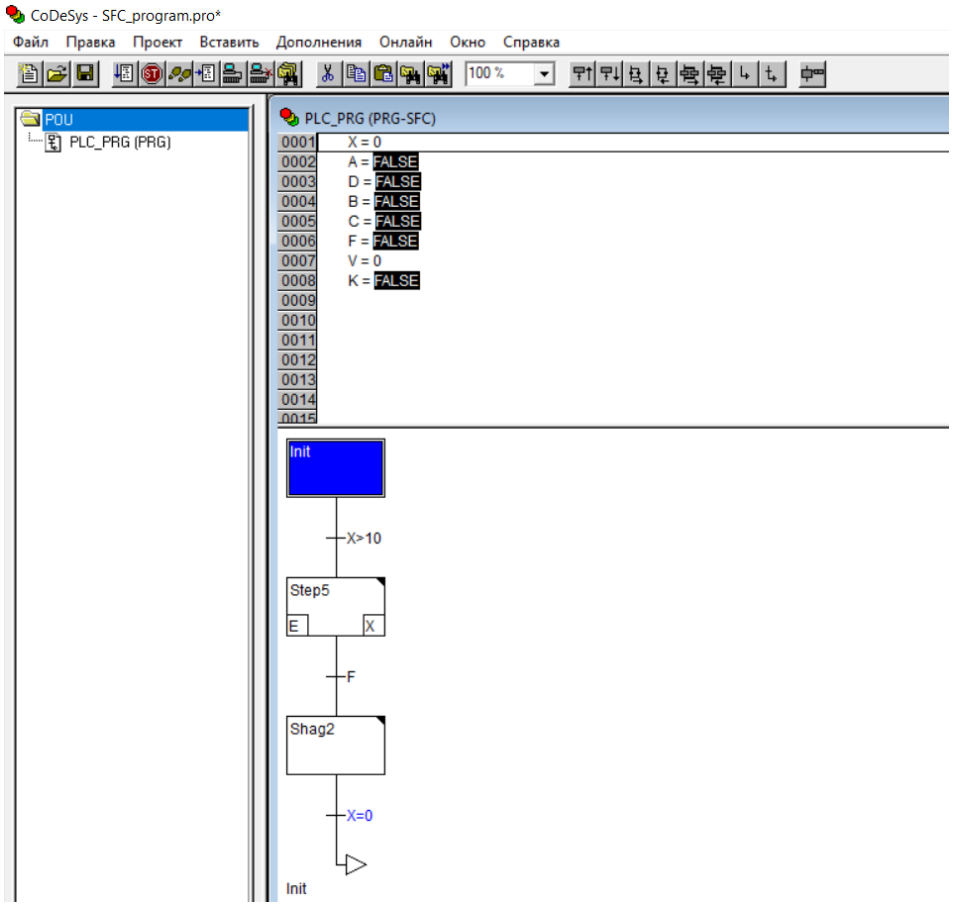

CoDeSys - SFC_program.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 X = 0
0002 A = FALSE
0003 D = FALSE
0004 B = FALSE
0005 C = FALSE
0006 F = FALSE
0007 V = 0
0008 K = FALSE
0009
0010
0011
0012
0013
0014
0015
Init
X>10
Step5
E
X
F
Shag2
X=0
Init

Рис. 24

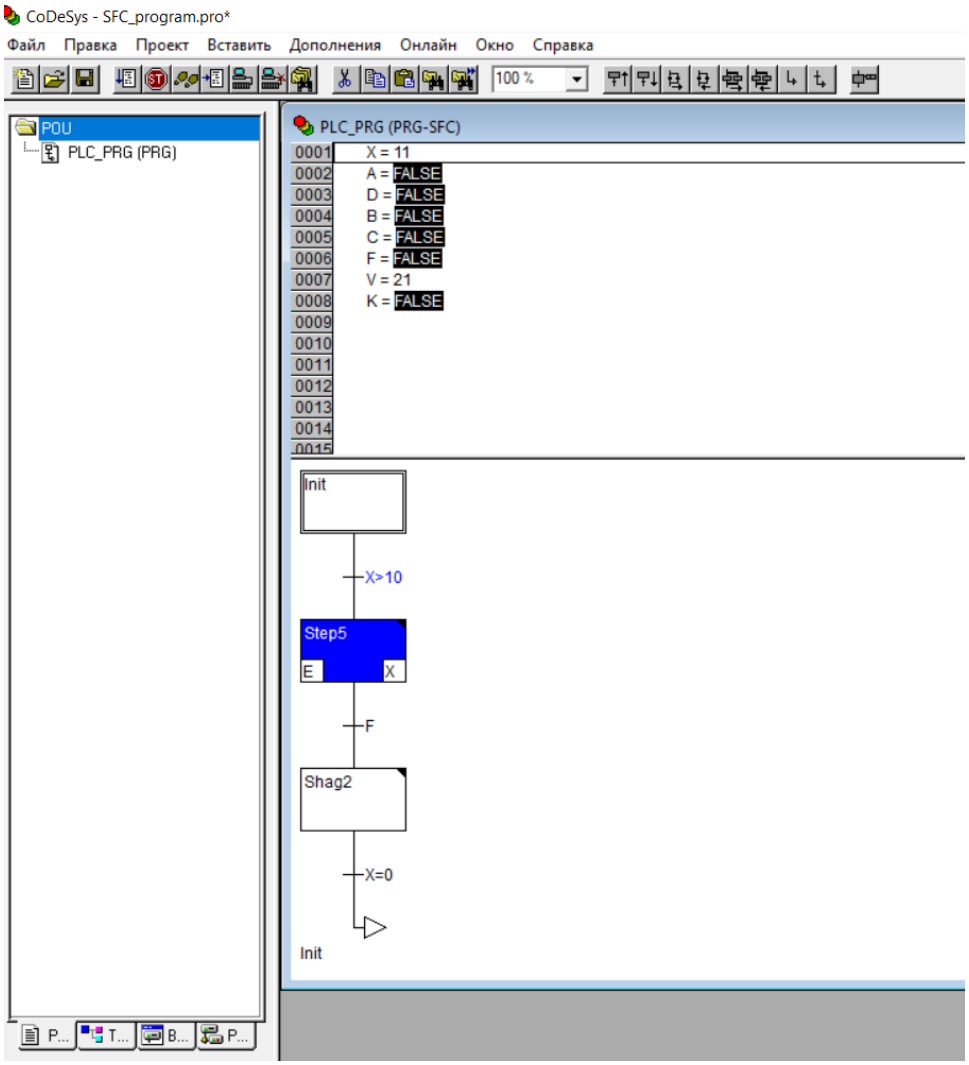

CoDeSys - SFC_program.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 X = 11
0002 A = FALSE
0003 D = FALSE
0004 B = FALSE
0005 C = FALSE
0006 F = FALSE
0007 V = 21
0008 K = FALSE
Init
X>10
Step5
E
X
F
Shag2
X=0
Init

Рис. 25

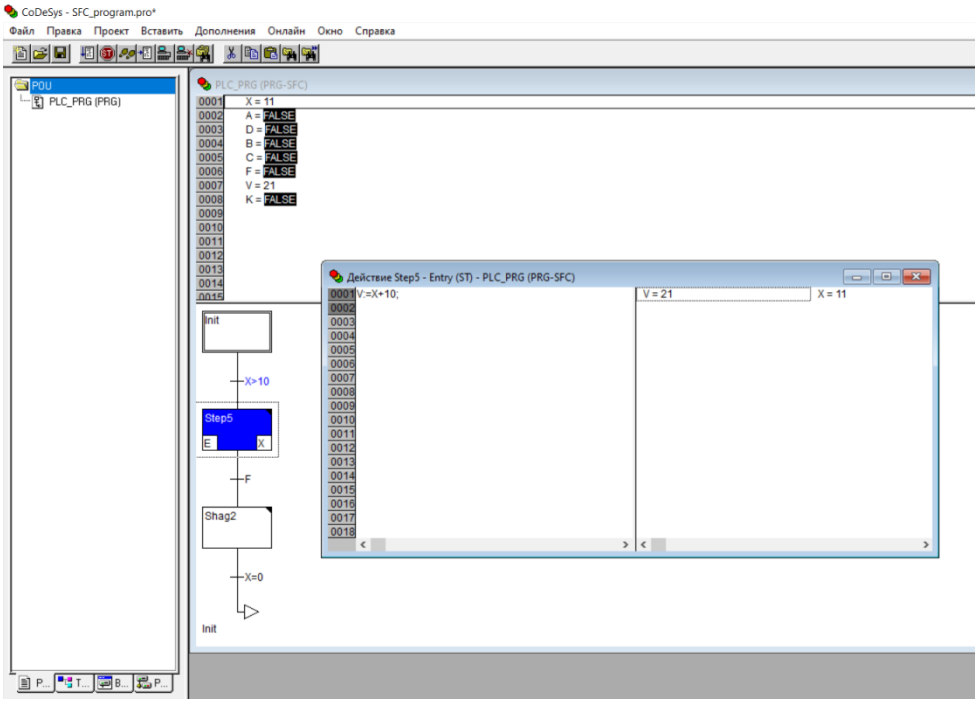

CoDeSys - SFC_program.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 X = 11
0002 A = FALSE
0003 D = FALSE
0004 B = FALSE
0005 C = FALSE
0006 F = FALSE
0007 V = 21
0008 K = FALSE
Действие Step5 - Entry (ST) - PLC_PRG (PRG-SFC)
0001 V:=X+10;
V = 21
X = 11
Init
X>10
Step5
E
X
F
Shag2
X=0
Init

Рис. 26

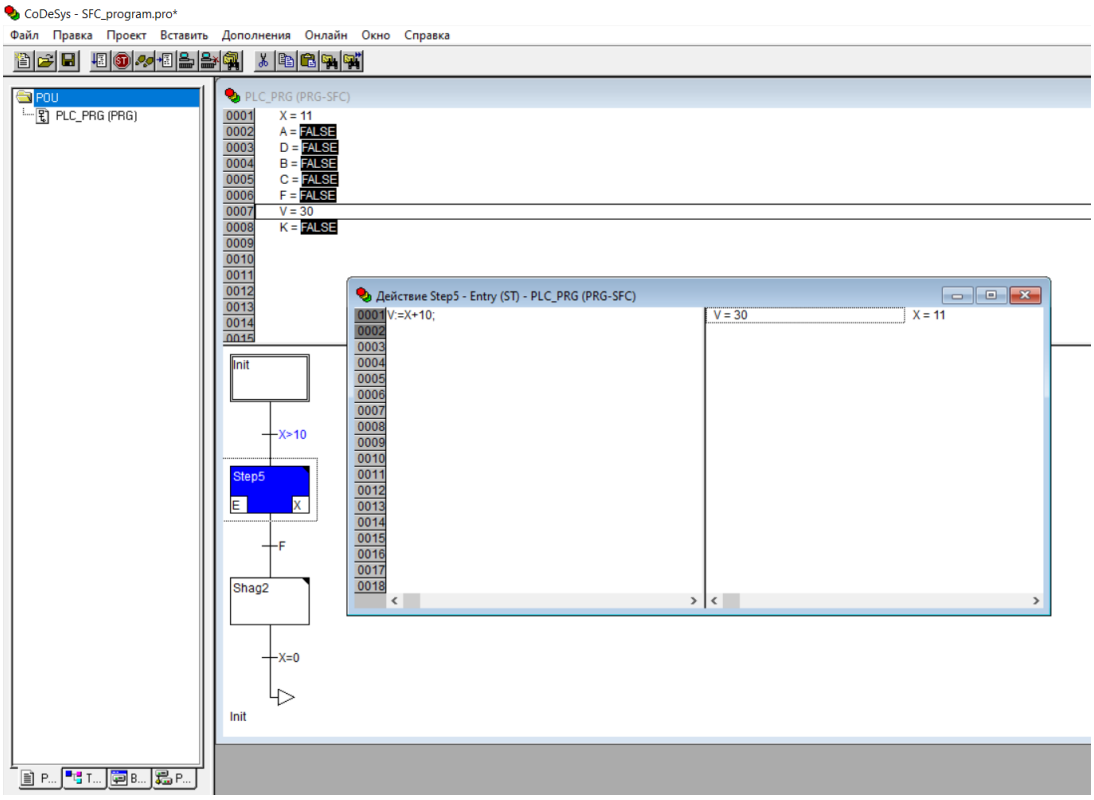


Рис. 27

Чтобы осуществился переход на Shag2, переменная F должна принять значение True (см. рис. 13). Для этого запишем значение True в переменную B (рис. 28). Так как произошел этот переход, то у шага Step5 выполнилось выходное действие, и значение переменной V стало равным 99.

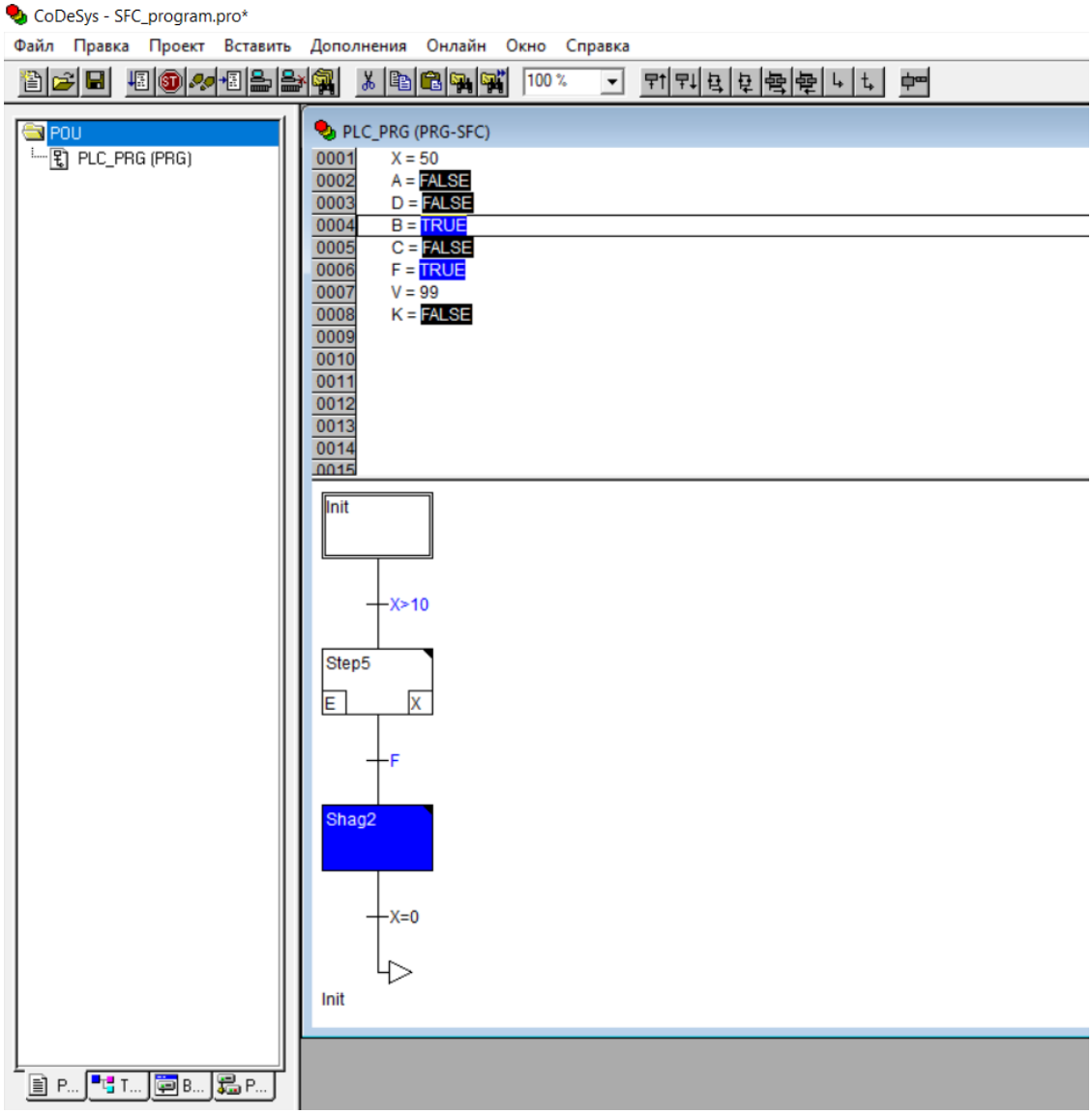


Рис. 28

Чтобы далее осуществился переход на метку Init, необходимо, чтобы X=0. Согласно логике действия шага Shag2 (см. рис. 22) для этого нужно, чтобы переменная K приняла значение True (тогда в X запишется 0), а для этого переменные A и C должны быть True. В результате видим, что выполнение программы снова вернулось на начальный шаг Init (рис. 29).

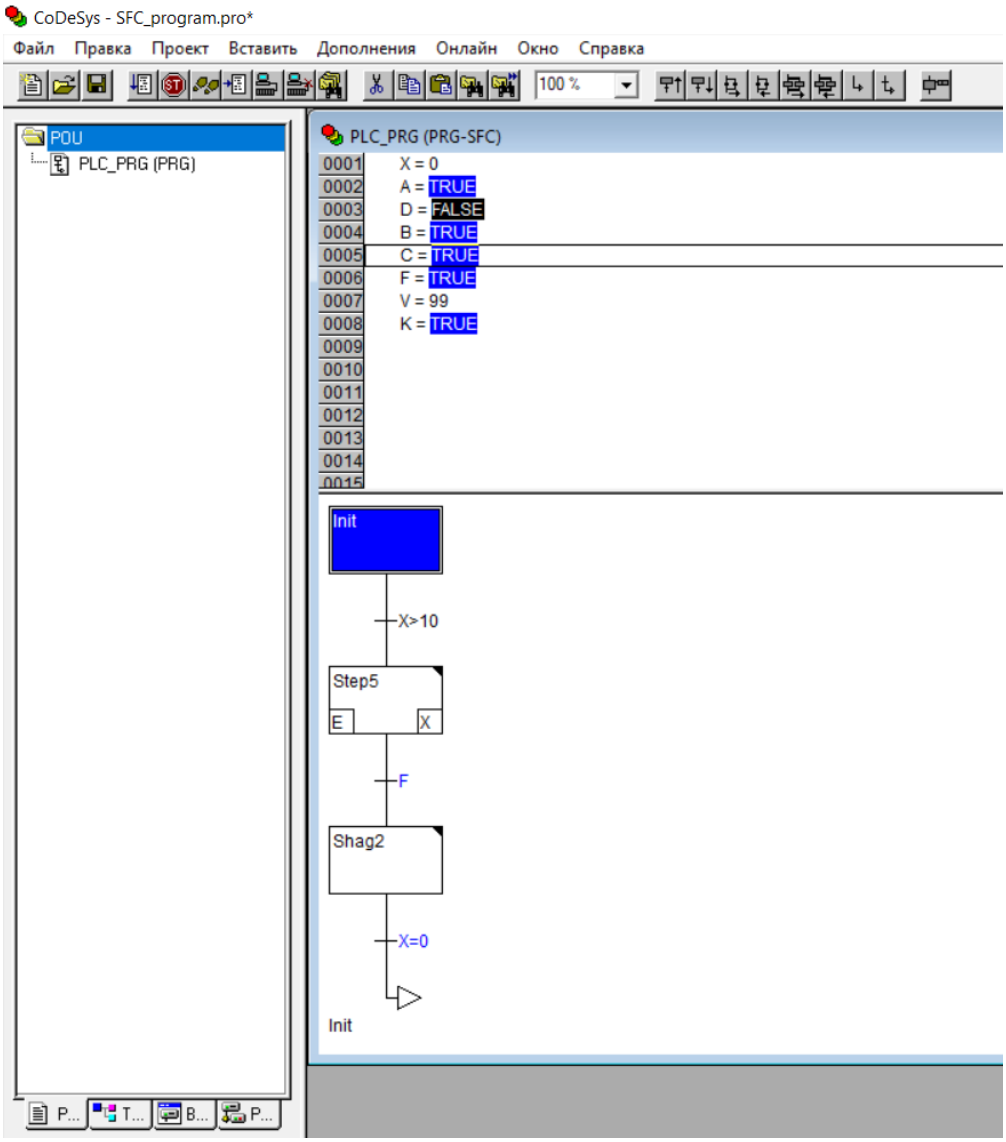


Рис. 29

7. Далее можно попробовать снова добиться пошагового выполнения программы, меняя значения нужных переменных, но не забывая, что после выполнения предыдущего шага у них уже присвоены какие-то значения. То есть, если сейчас задать для X значение 11, то переход к шагу Step5 произведен не будет. Предлагаю вам самостоятельно проделать нужные операции и посмотреть, как будет вести себя программа в том или ином случае. При этом всегда смотрите логику выполнения действий внутри шагов, а также входного и выходного действий.
8. После изучения работы программы отключим её выполнение и удалим входное и выходное действия для шага Step5. Для этого используем контекстное меню, команду Очистить действие(переход), ставим галочки на входное и выходное действия шага (рис. 30). Очистить действие можно только таким образом. То есть если вы просто удалите логические цепочки внутри действий, то программа всё равно будет считать, что для шага есть входное и выходное действие.

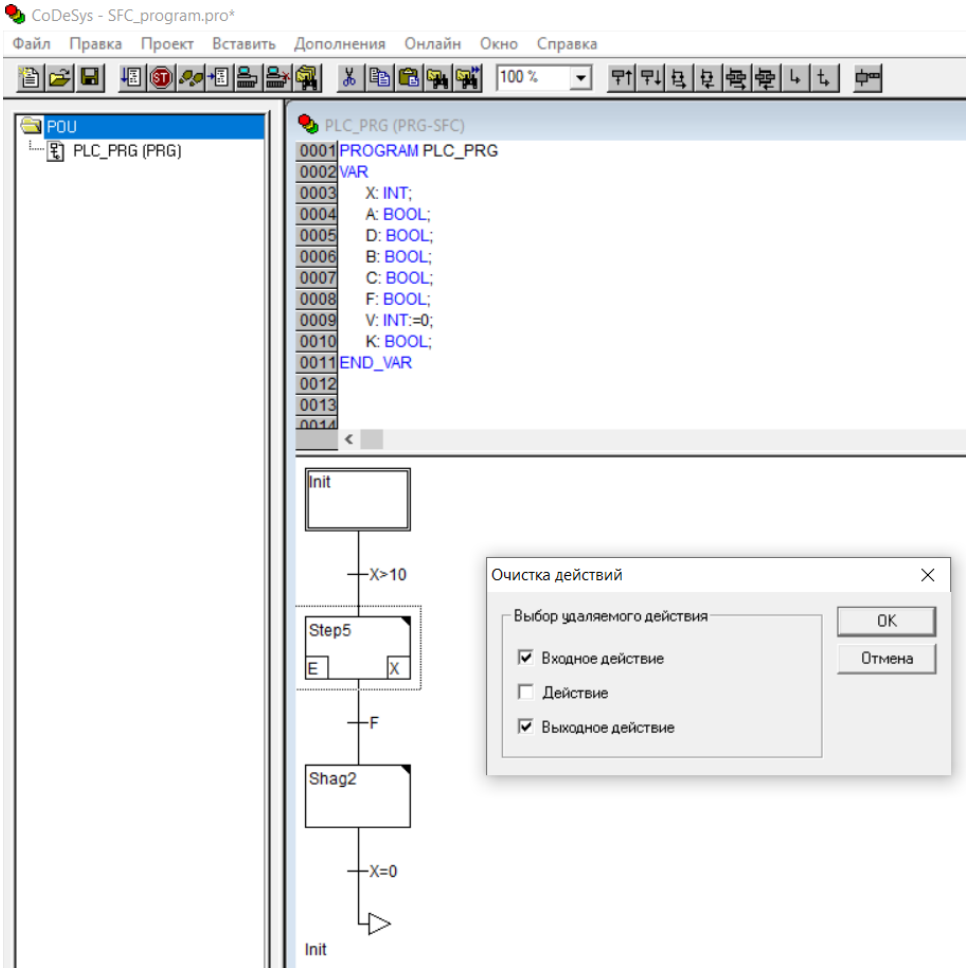


Рис. 30

Очистим аналогичным образом и действие внутри шага Step5.

9. Добавим теперь более сложное условие перехода на шаг Step5, напишем его на языке FBD. Для этого кликаем на переход дважды и выбираем нужный язык реализации, а затем пишем программу (рис. 31, 32). При этом ни в коем случае нельзя для выхода логического блока использовать операцию присваивания (при компиляции сразу же будет ошибка). Связано это с тем, что логический переход не должен включать побочных операций. В данном примере выход блока AND и будет являться сигналом выполнения или невыполнения данного перехода.

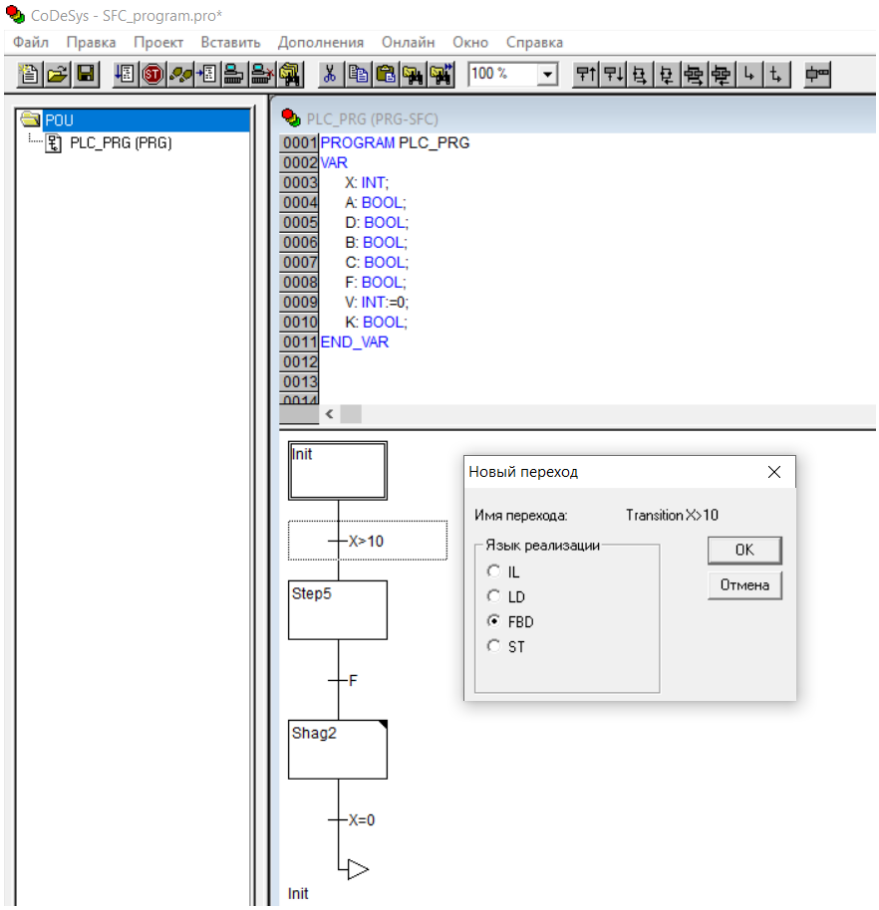

CoDeSys - SFC_program.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Init
X>10
Step5
F
Shag2
X=0
Init
Новый переход
Имя перехода: Transition X>10
Язык реализации
IL
LD
FBD
ST
OK
Отмена

Рис. 31

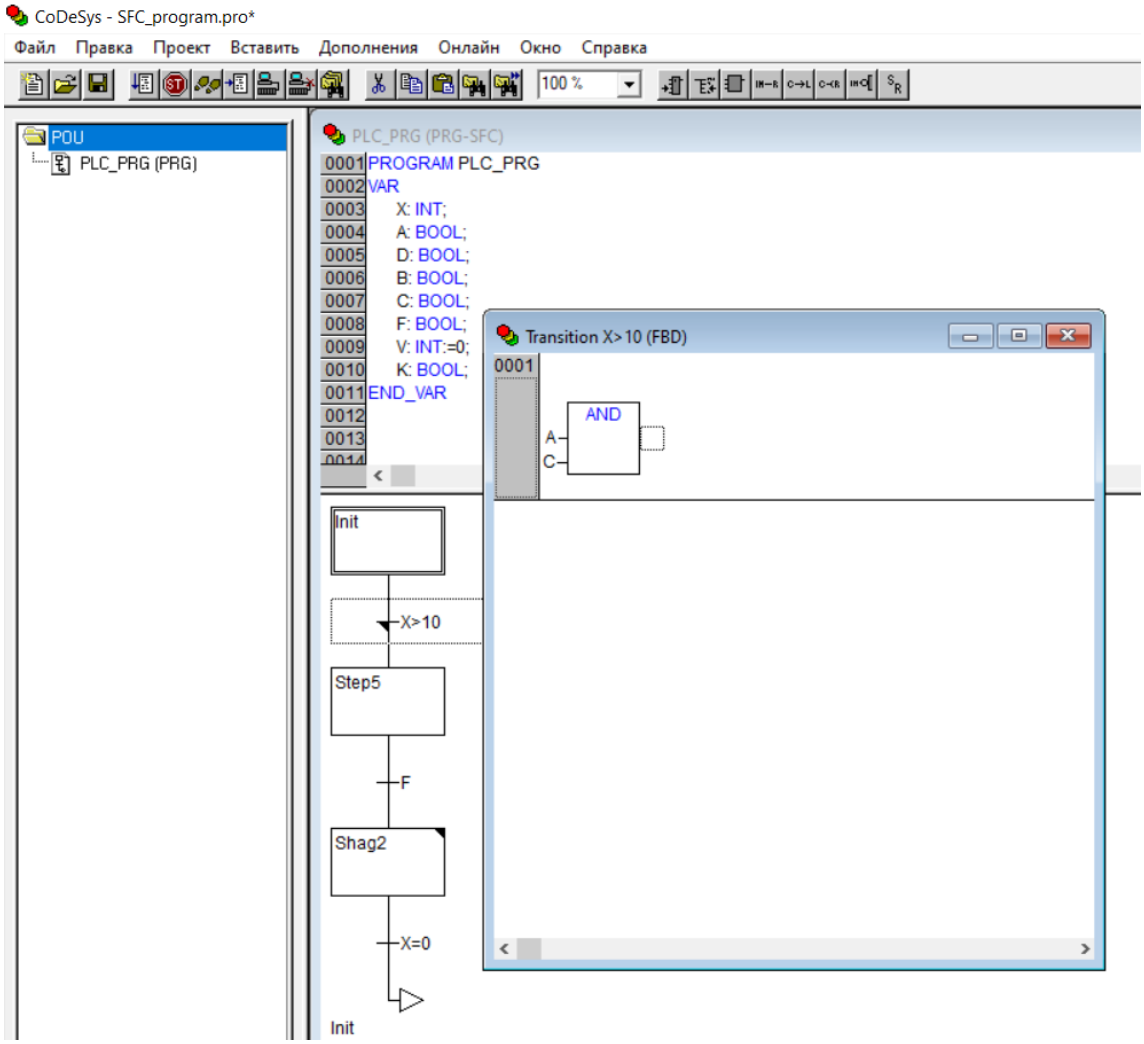

CoDeSys - SFC_program.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Transition X>10 (FBD)
0001
AND
A
C
Init
X>10
Step5
F
Shag2
X=0
Init

Рис. 32

После того, как была задана внутренняя логика перехода с помощью языка FBD, выражение X>10 перестало являться условием перехода и теперь стало вновь именем перехода. Правильнее всего будет сразу же изменить это выражение на какое-то осмысленное имя перехода, чтобы не вводить себя в заблуждение. Вы можете запустить программу на выполнение, задать X=11 и убедиться, что не произойдет переход на шаг Step5, пока не будет соблюдено внутренне логическое условие этого перехода (см. рис. 32).

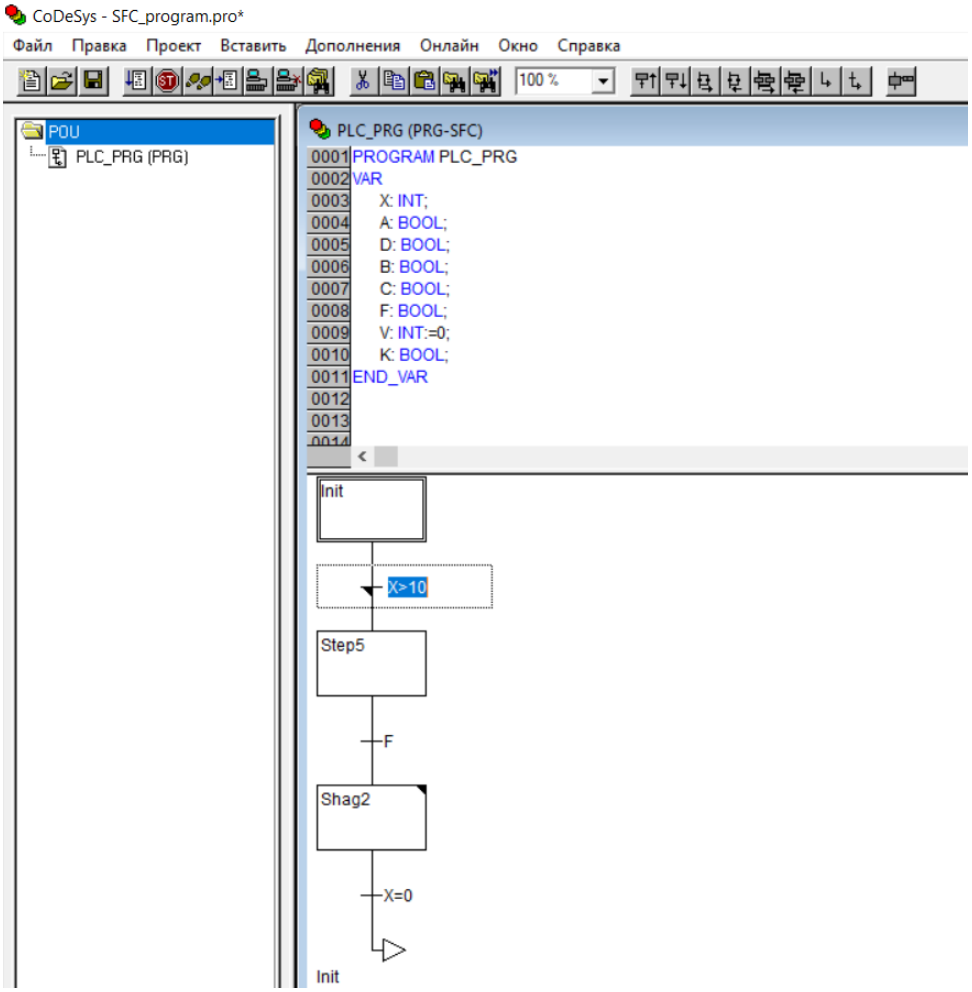


Рис. 33

Задание на самостоятельную работу

Необходимо в среде CoDeSys эмулировать работу сверлильного станка на языке SFC с визуализацией процесса сверления (аналогично рис. 34). Реализация логики работы программы представлена на рис. 35. Логика действий внутри шагов и названия переменных представлены в таблице 1. В ней же приведена настройка элементов визуализации.

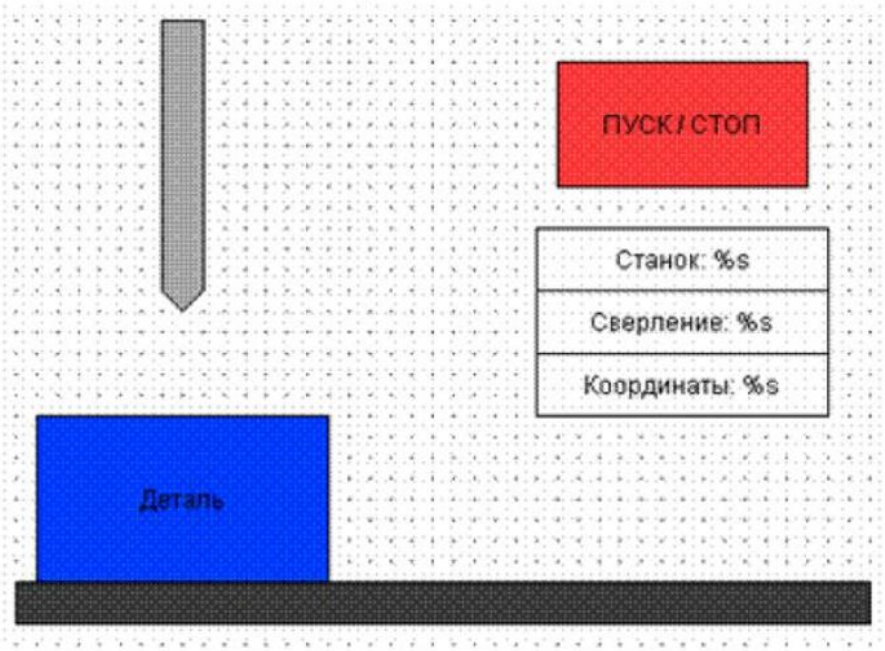


Рис. 34. Визуализация процесса сверления

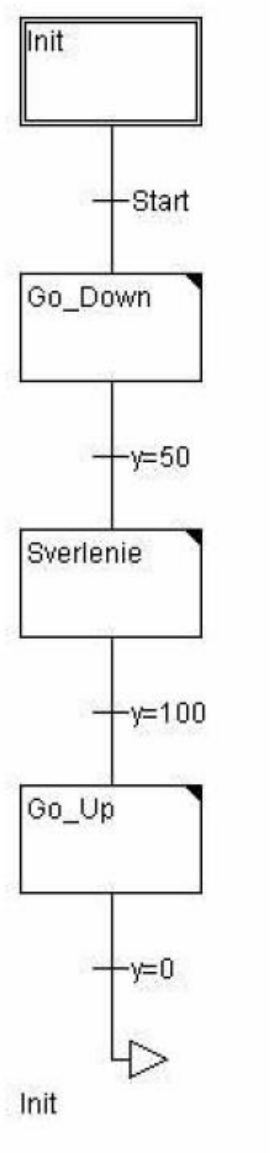


Рис. 35. Программа на языке SFC

Таблица 1 – Шаги программы и настройка элементов визуализации

Категория	Подкатегория / Значение
Шаги программы	
Шаг Start	
Шаг Go_Down	y:=y+1;
Шаг Sverlenie	Rejim_Sverlenia:=TRUE; y:=y+1;
Шаг Go_Up	Rejim_Sverlenia:=FALSE; y:=y-1;
Деталь	
Текст	Деталь
Цвета	Обычный: Синий
Сверло	
Цветаа	Обычный: Серый Тревожный: Темно-серый
Положение	Сдвиг по Y: PLC_PRG.y
Переменные	Изменение цвета: PLC_PRG.Rejim_Sverlenia
Кнопка	
Текст	ПУСК / СТОП
Цвета	Обычный: Зеленый Тревожный: Красный
Переменные	Изменение цвета: PLC_PRG.Start
Параметр - Станок	
Текст	Станок: %s
Переменные	Вывод текста: PLC_PRG.Start
Параметр - Сверление	
Текст	Сверление: %s
Переменные	Вывод текста: PLC_PRG.Rejim_Sverlenia
Параметр - Координатысверла	
Текст	Координаты сверла: %s
Переменные	PLC_PRG.y

Содержание отчета

1. Скрины окна PLC_PRG с готовыми программами на языке SFC.
2. Скрины действий шагов.
3. Скрины визуализации (настройка элементов и итоговый результат работы).

Практическая работа № 5
Использование МЭК шагов в SFC

Порядок выполнения

1. Загрузим проект, полученный при выполнении практической работы №4 (часть до самостоятельной работы), и удалим все переходы и действия, а затем включим в проекте опцию «Использовать МЭК шаги» (рис. 1).

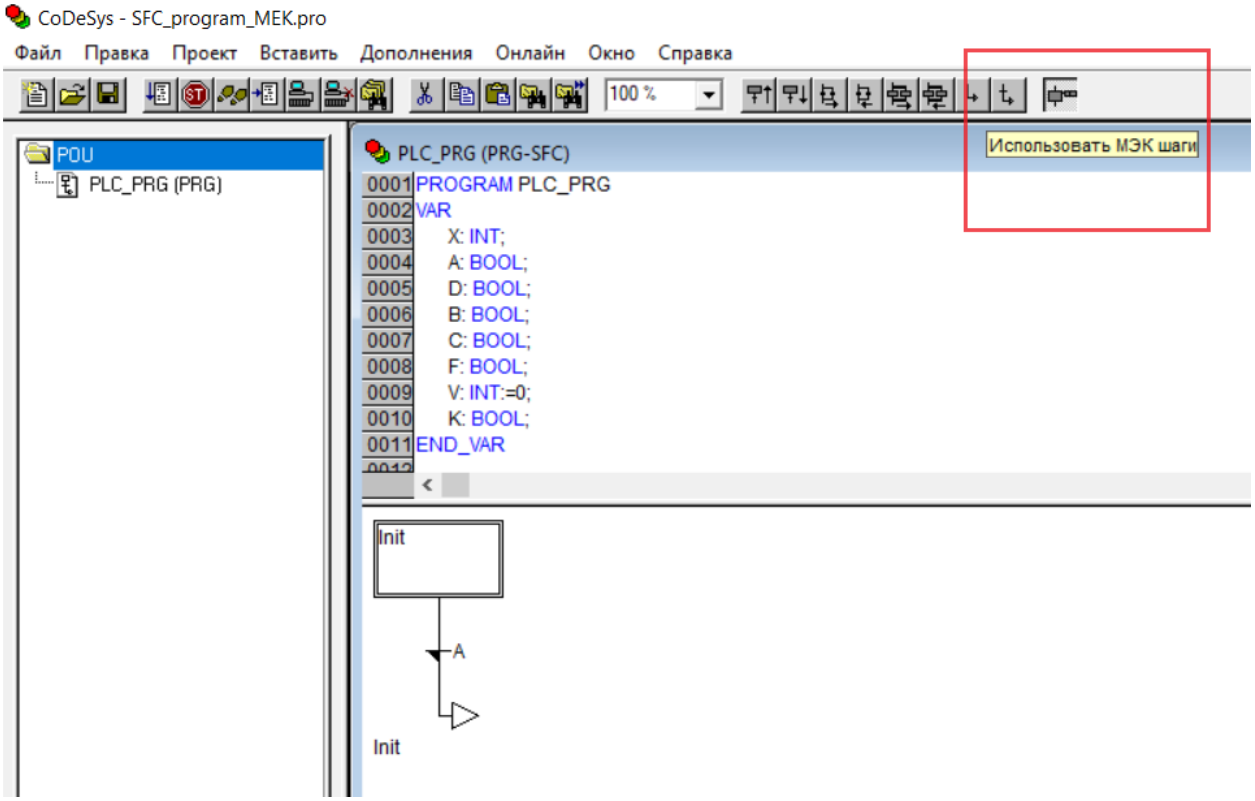


Рис. 1

2. МЭК-шаг отличается от простого шага тем, что внутри шага можно добавлять несколько действий одновременно. Для добавления шага выделим переход A и выберем команду «Шаг-переход снизу» (рис. 2), а затем через контекстное меню выберем команду «Связать действия» и добавим несколько действий внутри шага (рис. 3).

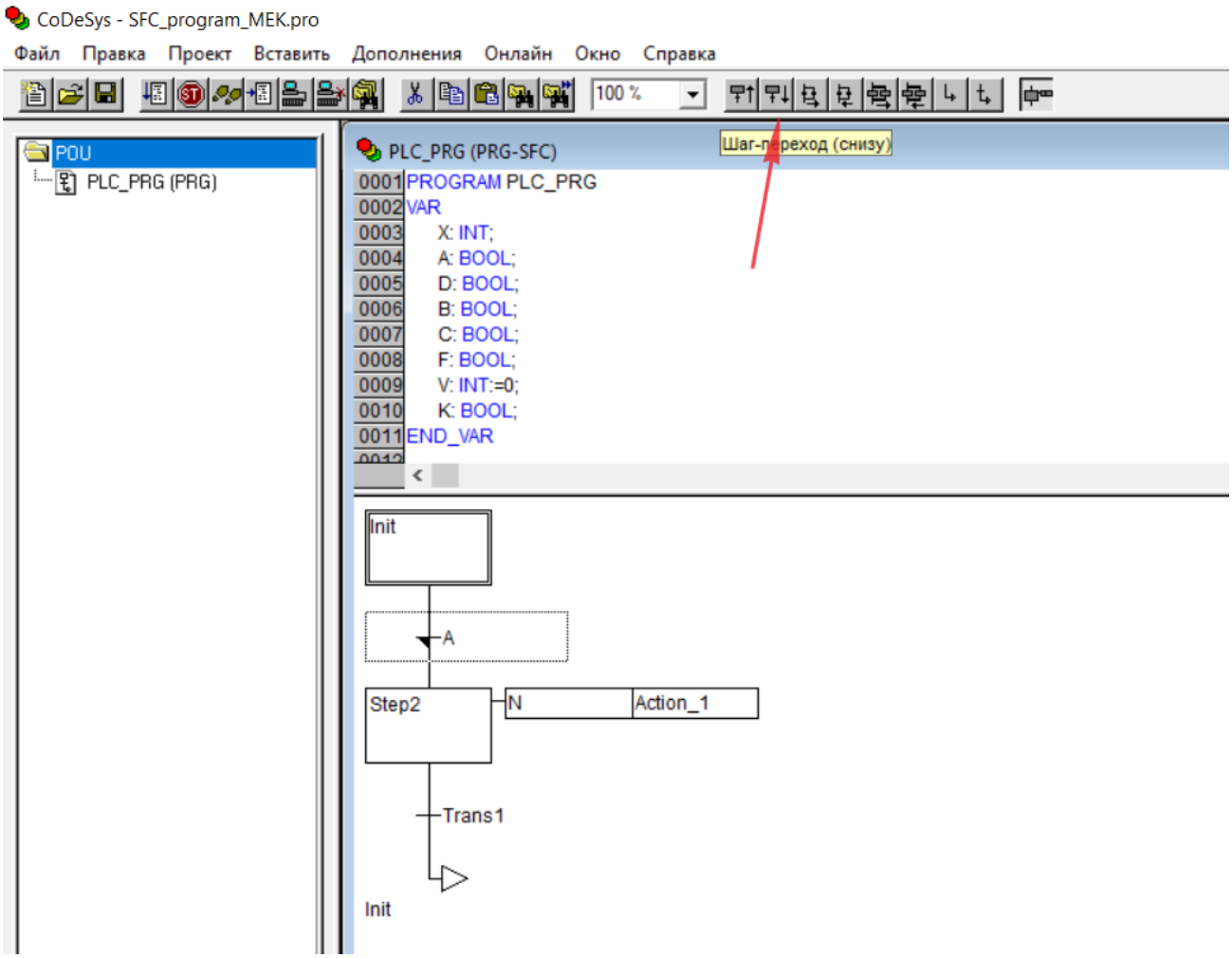

CoDeSys - SFC_program_MEK.pro
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
Шаг-переход (снизу)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Init
A
Step2
N
Action_1
Trans1
Init

Рис. 2

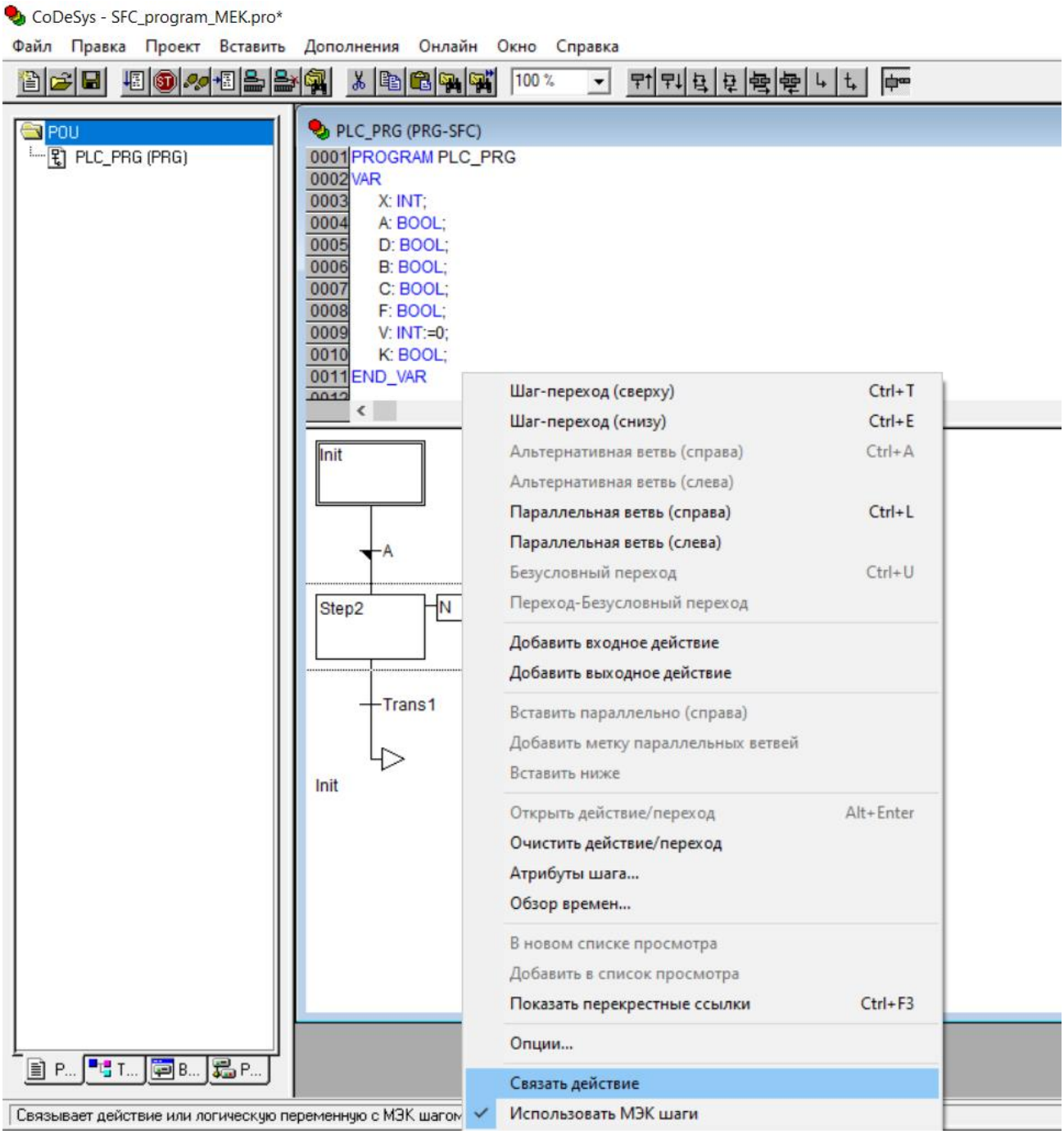


Рис. 3

Каждое действие может иметь классификатор. С помощью классификаторов действия и логические переменные могут активироваться и деактивироваться, возможно, с задержкой времени. Чтобы выбрать нужный классификатор, можно выделить классификатор по умолчанию и вызвать ассистент ввода через клавишу F2 (рис. 4). Список классификаторов представлен на рис. 5. Выберем три классификатора – N, P, SD (рис. 5).

Справа от классификаторов указываются имена для действий, которые будут выполняться для этих классификаторов на данном шаге.

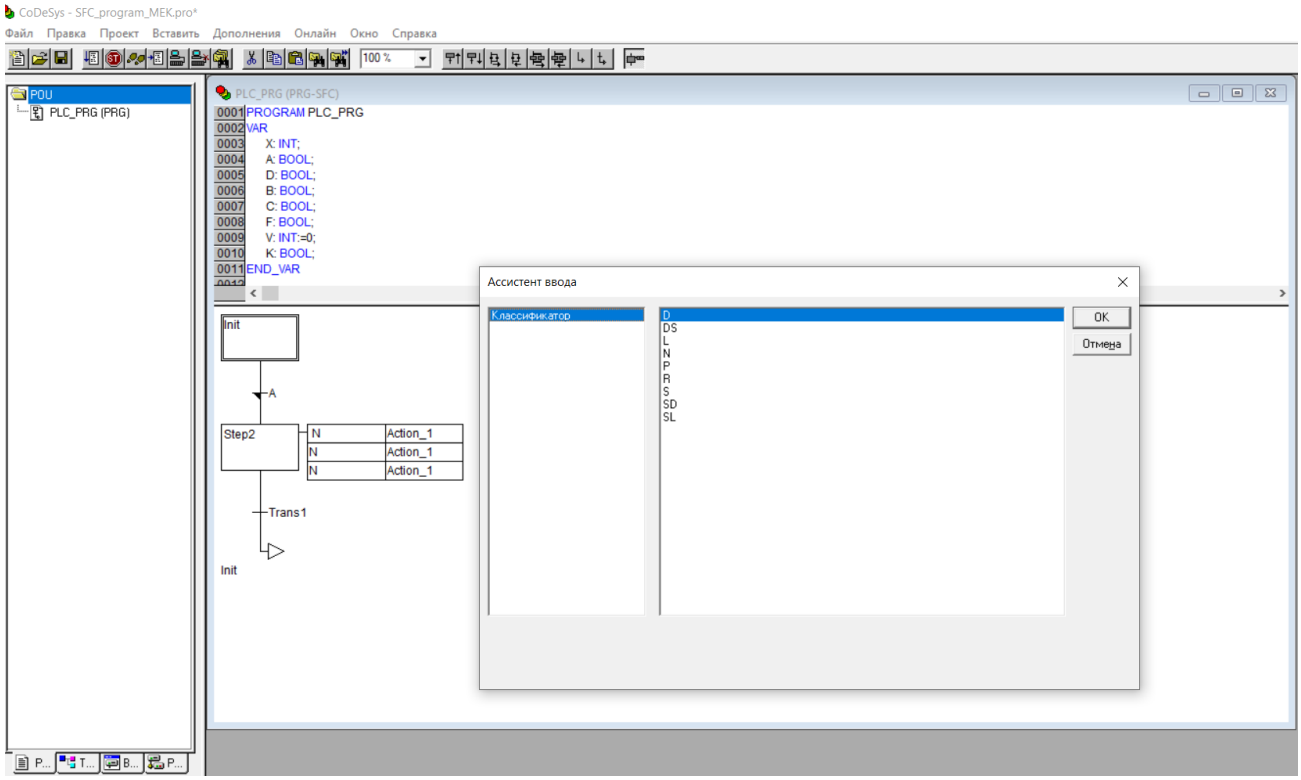


Рис. 4

N	Не сохраняемое	Действие активно в течение активности шага
R	Внеочередной Сброс	Деактивация действия
S	Установка	Действие активно вплоть до сброса
L	Ограниченное по времени	Действие активно в течение указанного времени, но не дольше времени активности шага
D	Отложенное	Действие активируется по прошествии указанного времени, если шаг еще активен и продолжает быть активным
P	Импульс	Действие выполняется один раз, если шаг активен
SD	Сохраняемое и отложенное	Действие активно после указанного времени до сброса
DS	Отложенное и сохраняемое	Действие активно после указанного времени, если шаг еще активен, вплоть до сброса
SL	Сохраняемое и ограничен-ное по времени	Активно после указанного времени.

Рис. 5

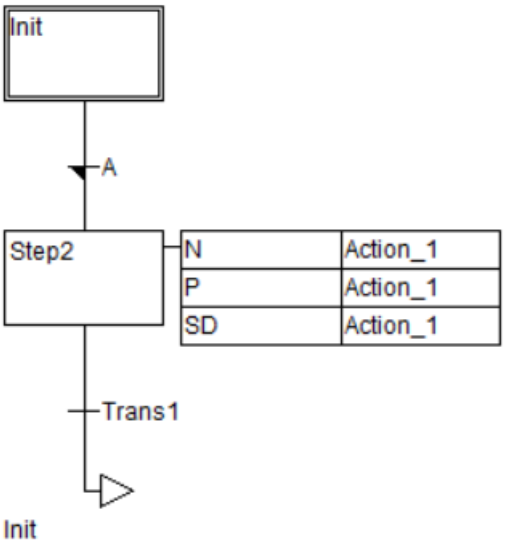


Рис. 6

3. При использовании режима «МЭК шаги» действия добавляются через окно с перечнем POU. Для добавления действия нужно вызвать контекстное меню и выбрать команду «Добавить действие» (рис. 7). При этом нужно обязательно выделить главную программу проекта PLC_PRG.
   Создадим действие Action_1 на языке программирования FBD (рис. 8), запишем для него простую логическую программу (рис. 9). Так как действие Action_1 имеет классификатор N, то оно будет выполняться постоянно, пока активен данный шаг Step2.

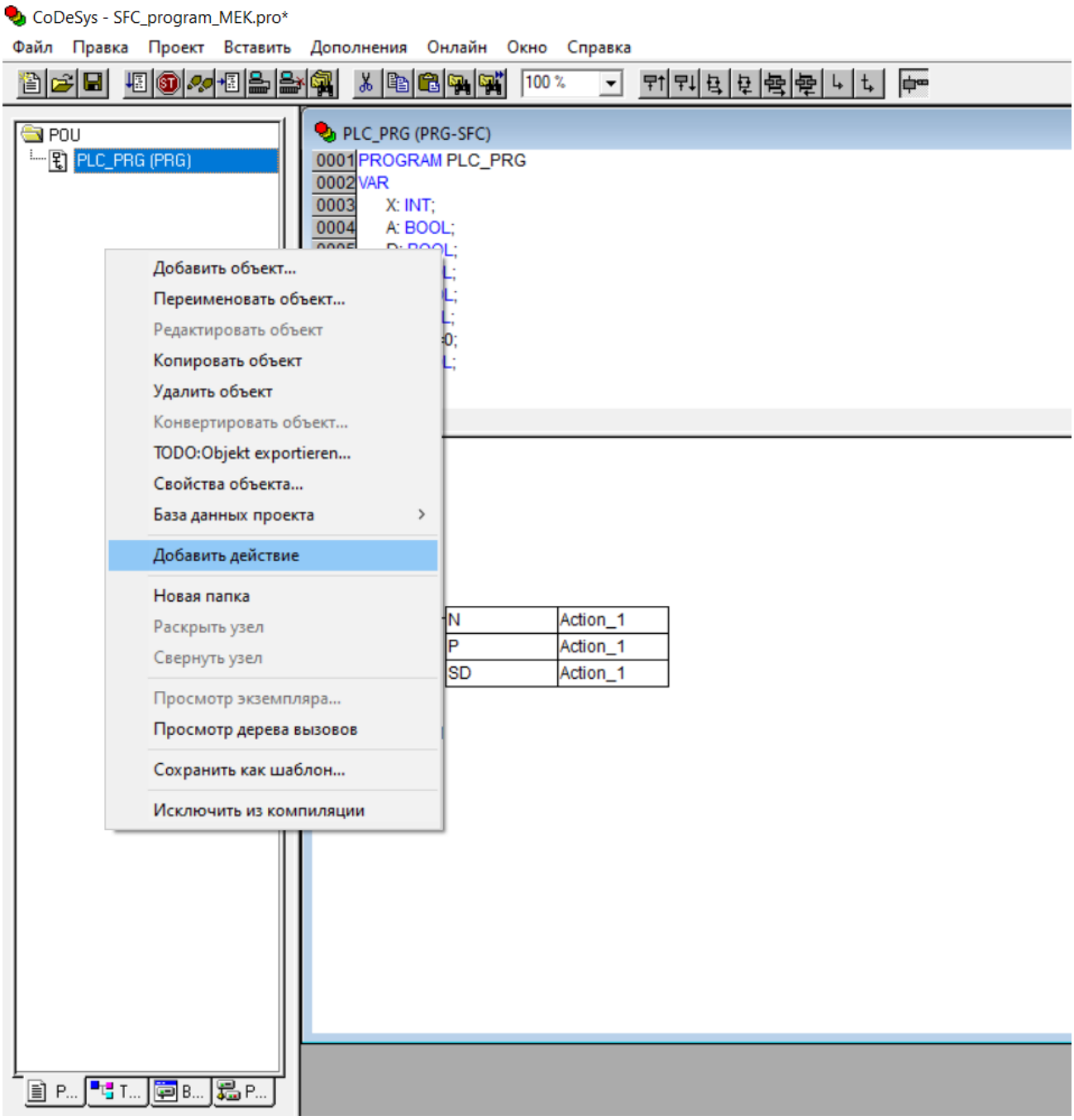

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
Добавить объект...
Переименовать объект...
Редактировать объект
Копировать объект
Удалить объект
Конвертировать объект...
TODO:Objekt exportieren...
Свойства объекта...
База данных проекта
Добавить действие
Новая папка
Раскрыть узел
Свернуть узел
Просмотр экземпляра...
Просмотр дерева вызовов
Сохранить как шаблон...
Исключить из компиляции
N Action_1
P Action_1
SD Action_1
P...
T...
B...
P...

Рис. 7

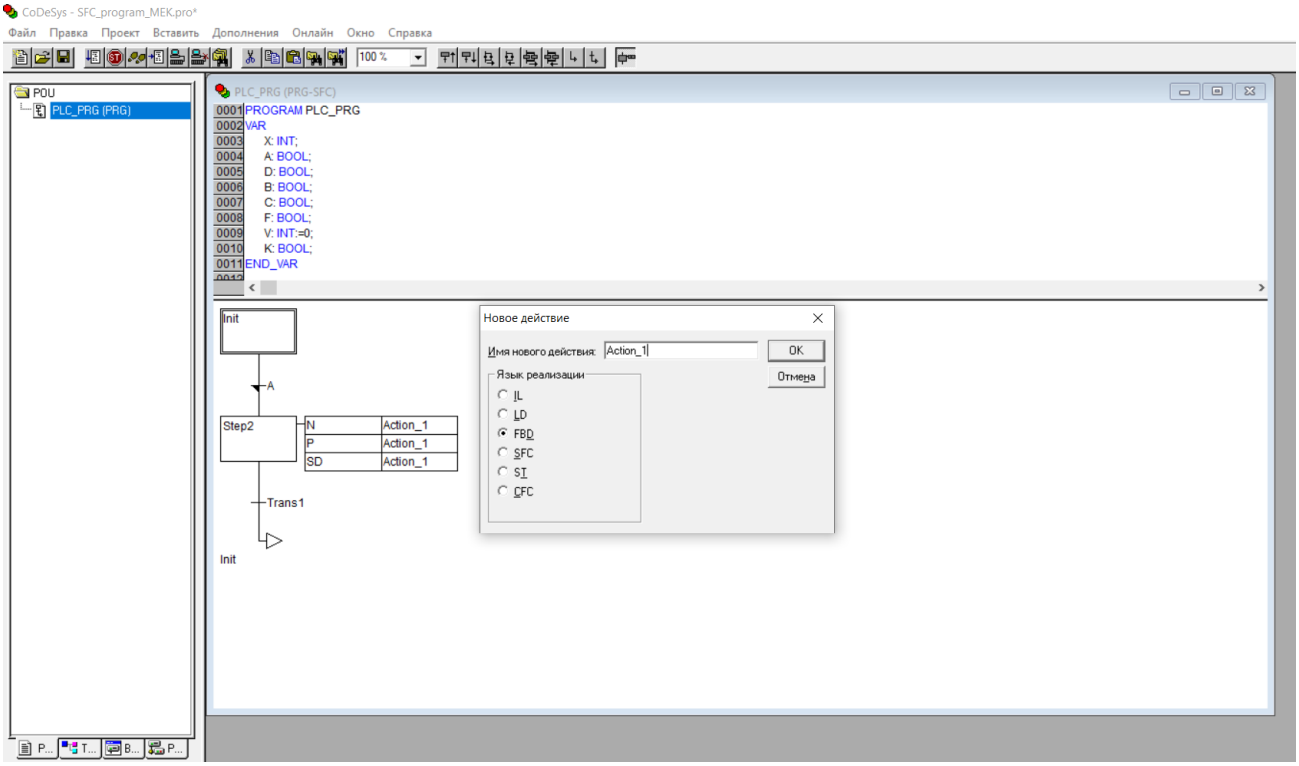


Рис. 8

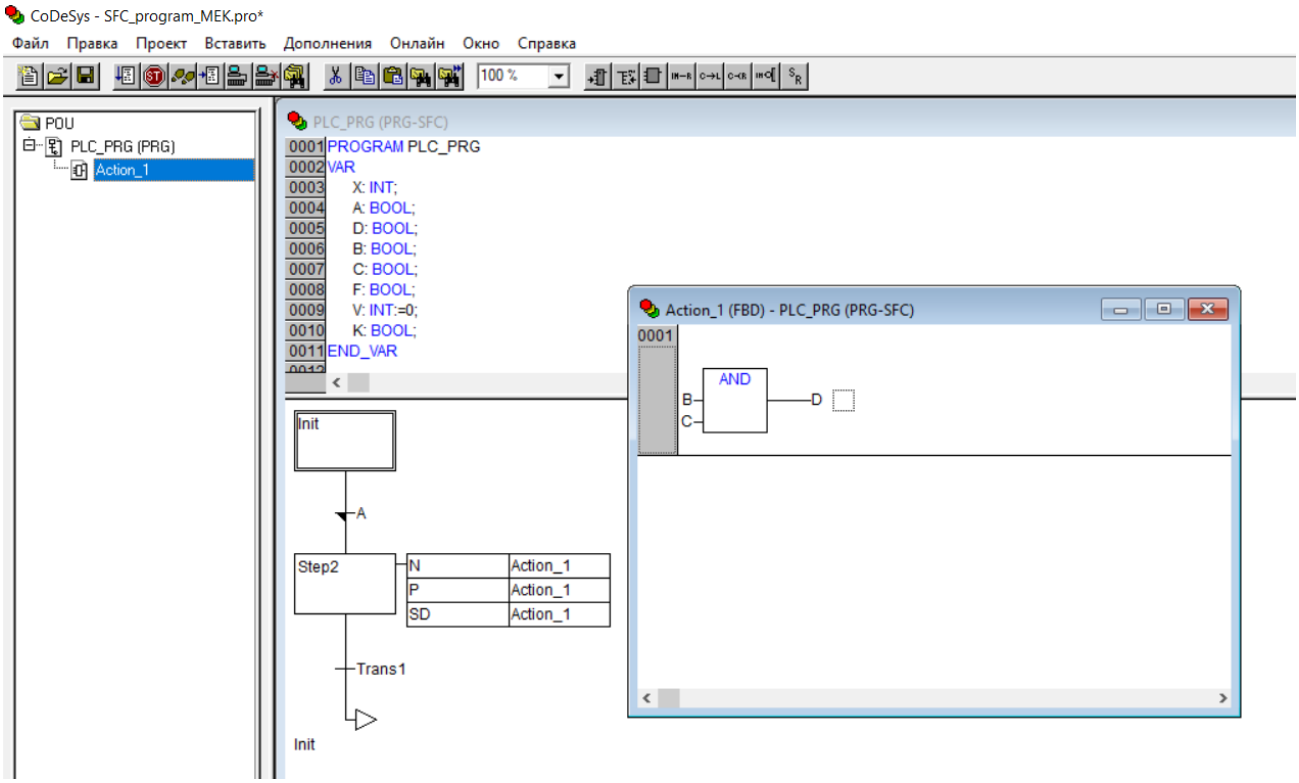


Рис. 9

4. Для классификатора P создадим действие с именем «Dejstvie2» на языке программирования LD (рис. 10, 11). Имя действия в шаге можно написать либо выбрать через F2(рис. 12).

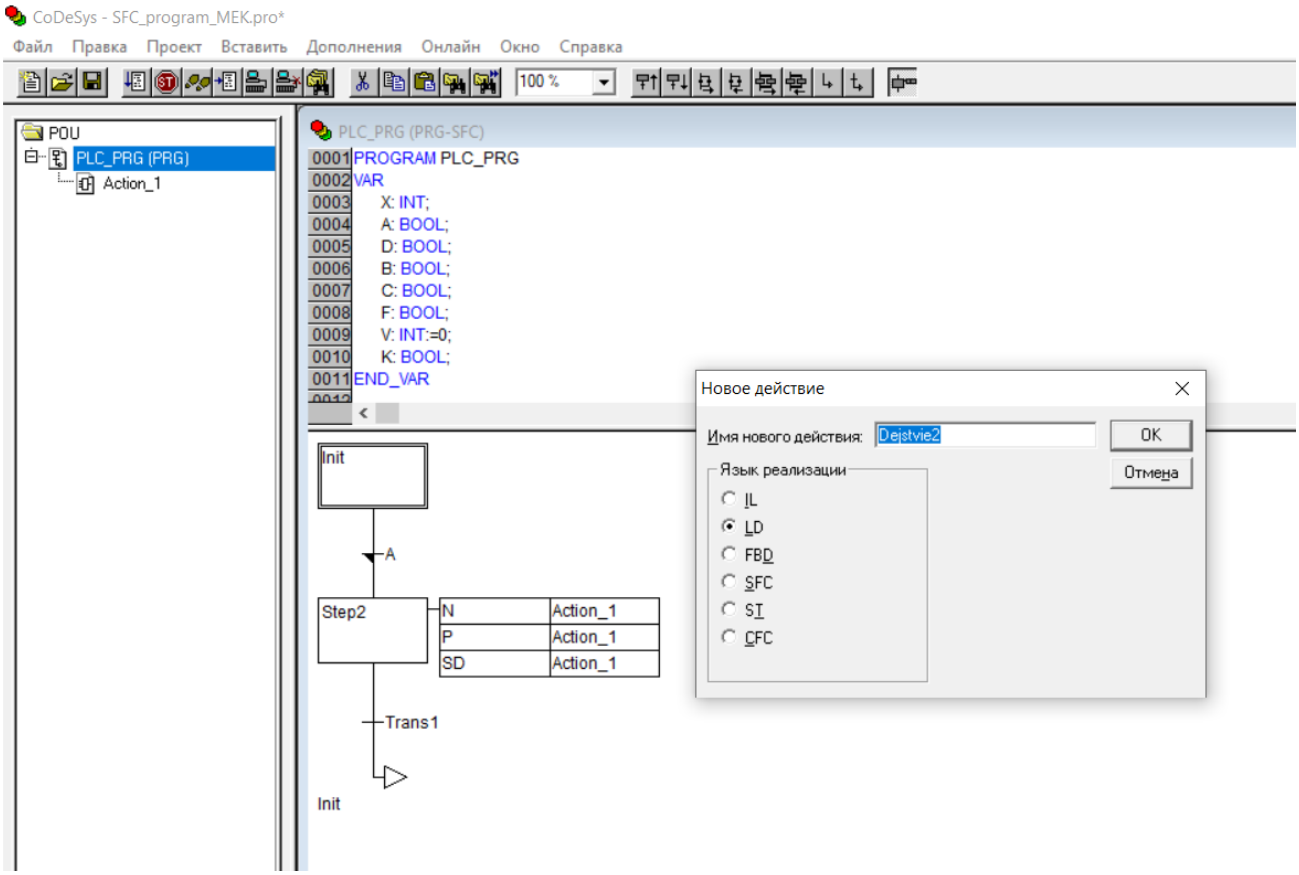

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
Action_1
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Init
A
Step2
N Action_1
P Action_1
SD Action_1
Trans1
Init
Новое действие
Имя нового действия: Dejstvie2
OK
Отмена
Язык реализации
IL
LD
FBD
SFC
ST
CFC

Рис. 10

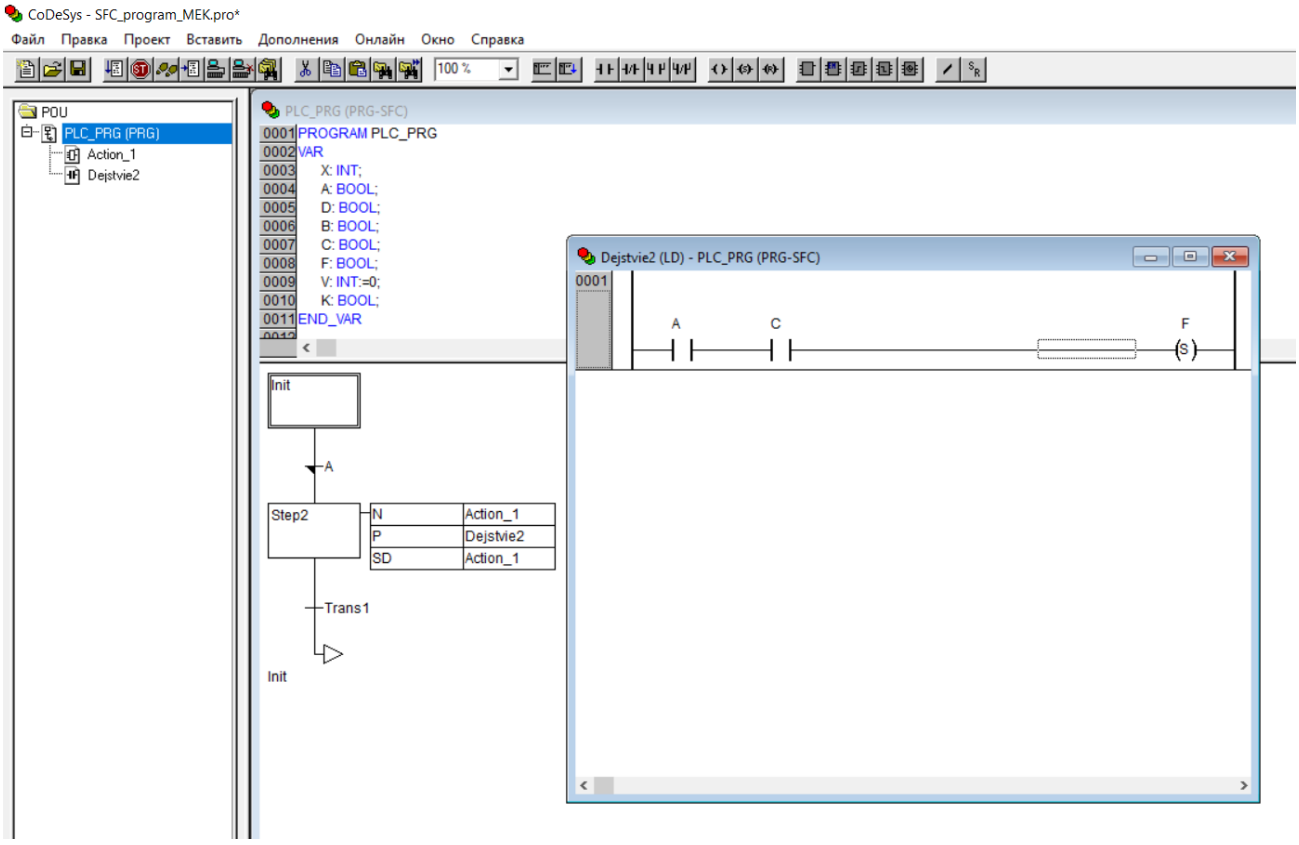

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
Action_1
Dejstvie2
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Init
A
Step2
N Action_1
P Dejstvie2
SD Action_1
Trans1
Init
Dejstvie2 (LD) - PLC_PRG (PRG-SFC)
0001
A
C
F
S

Рис. 11

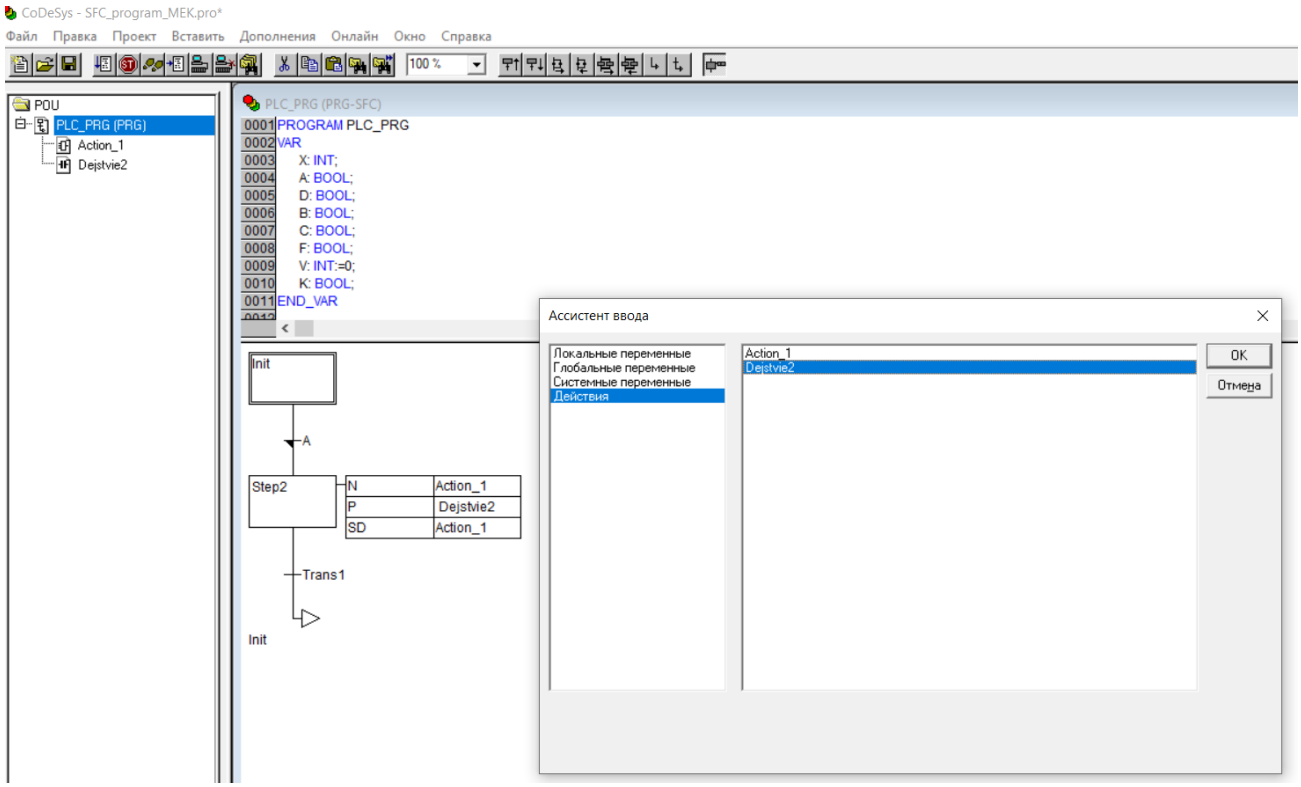


Рис. 12

5. Для классификатора SD создадим действие с именем «SUM» на языке программирования ST (рис. 13, 14).

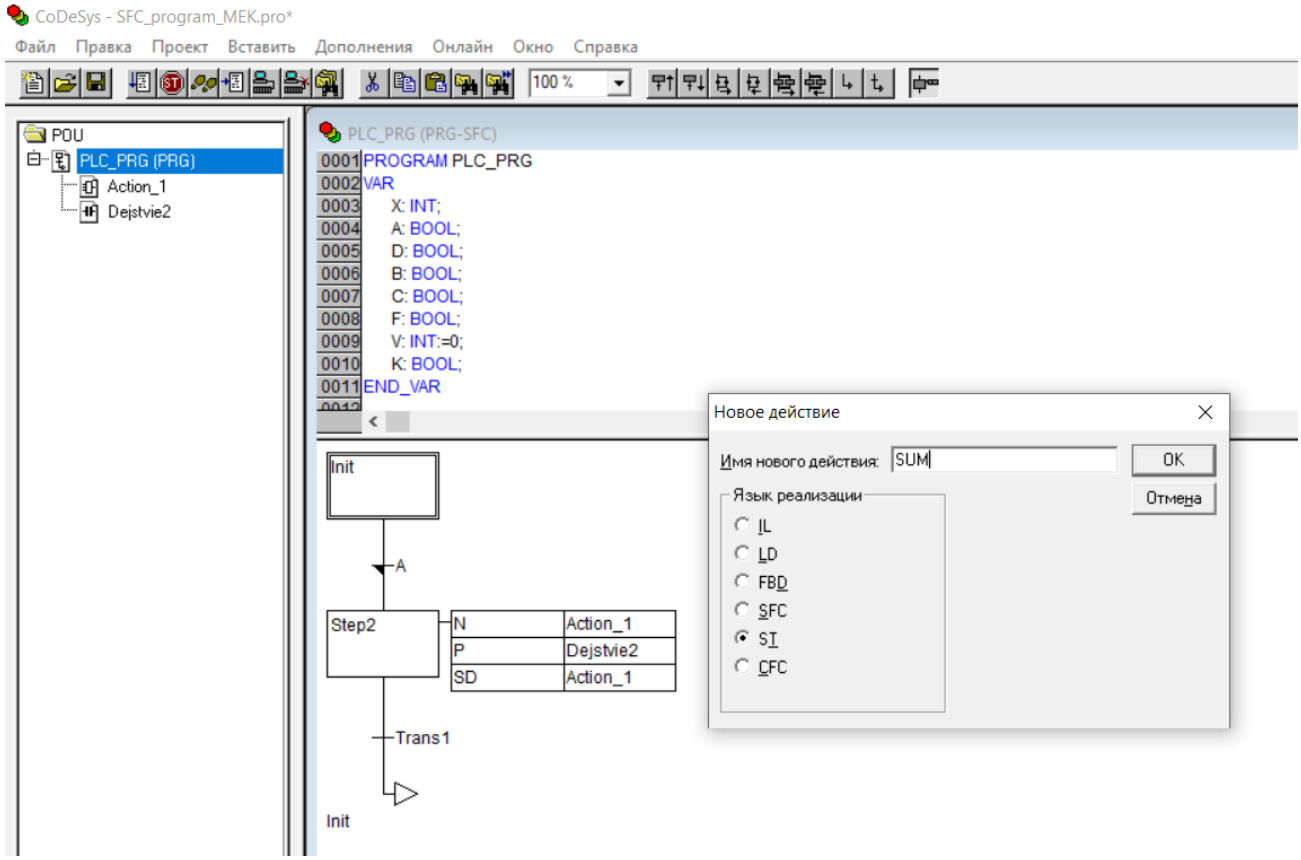


Рис. 13

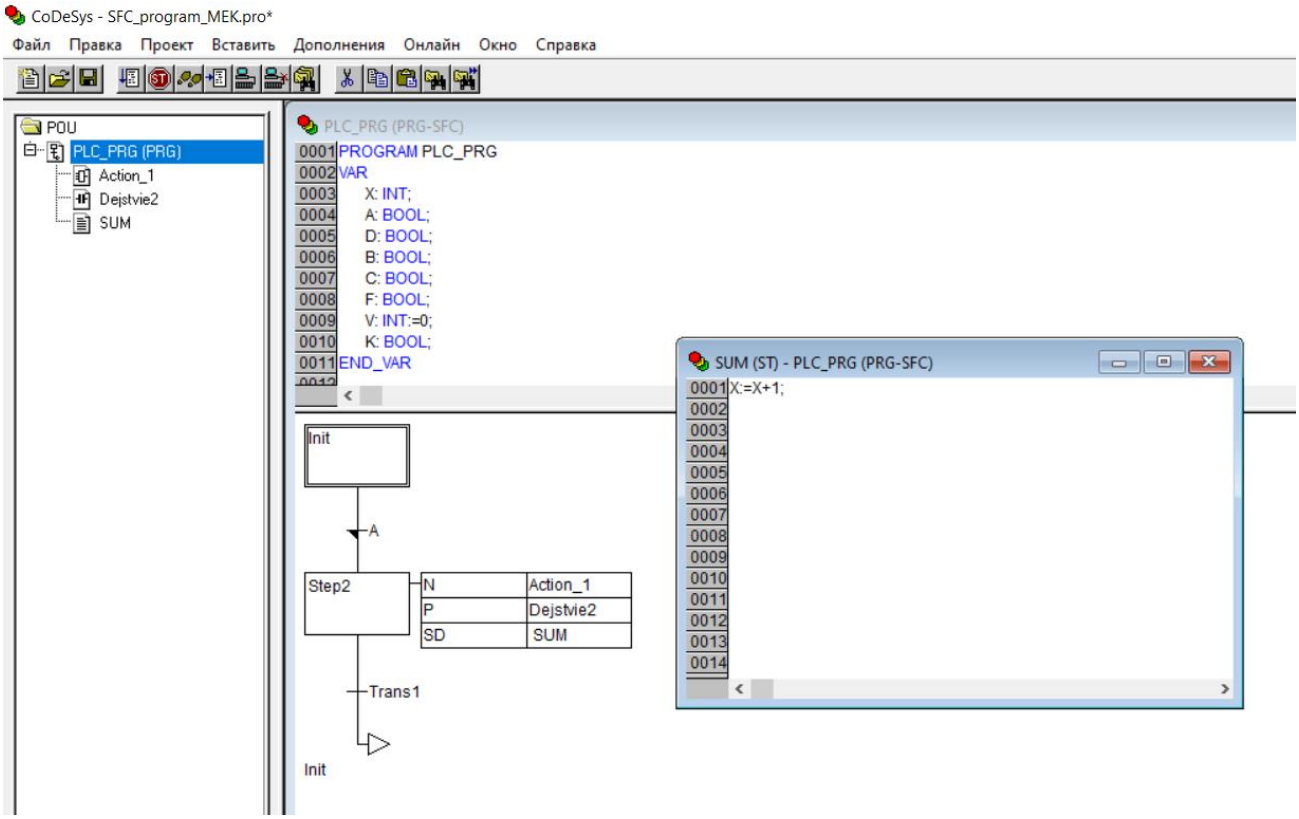


Рис. 14

При работе с классификатором SD есть один нюанс. Так как это классификатор отложенного действия, то он требует специальную переменную – время, после истечения которого данное действие будет активно. Укажем эту переменную в поле после классификатора SD, формат переменной – временной (рис. 15). Теперь в программе через 5 секунд после запуска шага Step2 будет выполняться действие SUM.

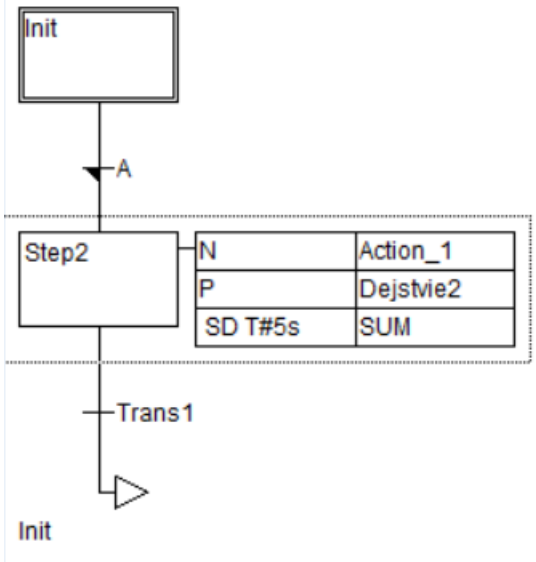


Рис. 15

6. Добавим условие перехода на метку Init (рис. 16).

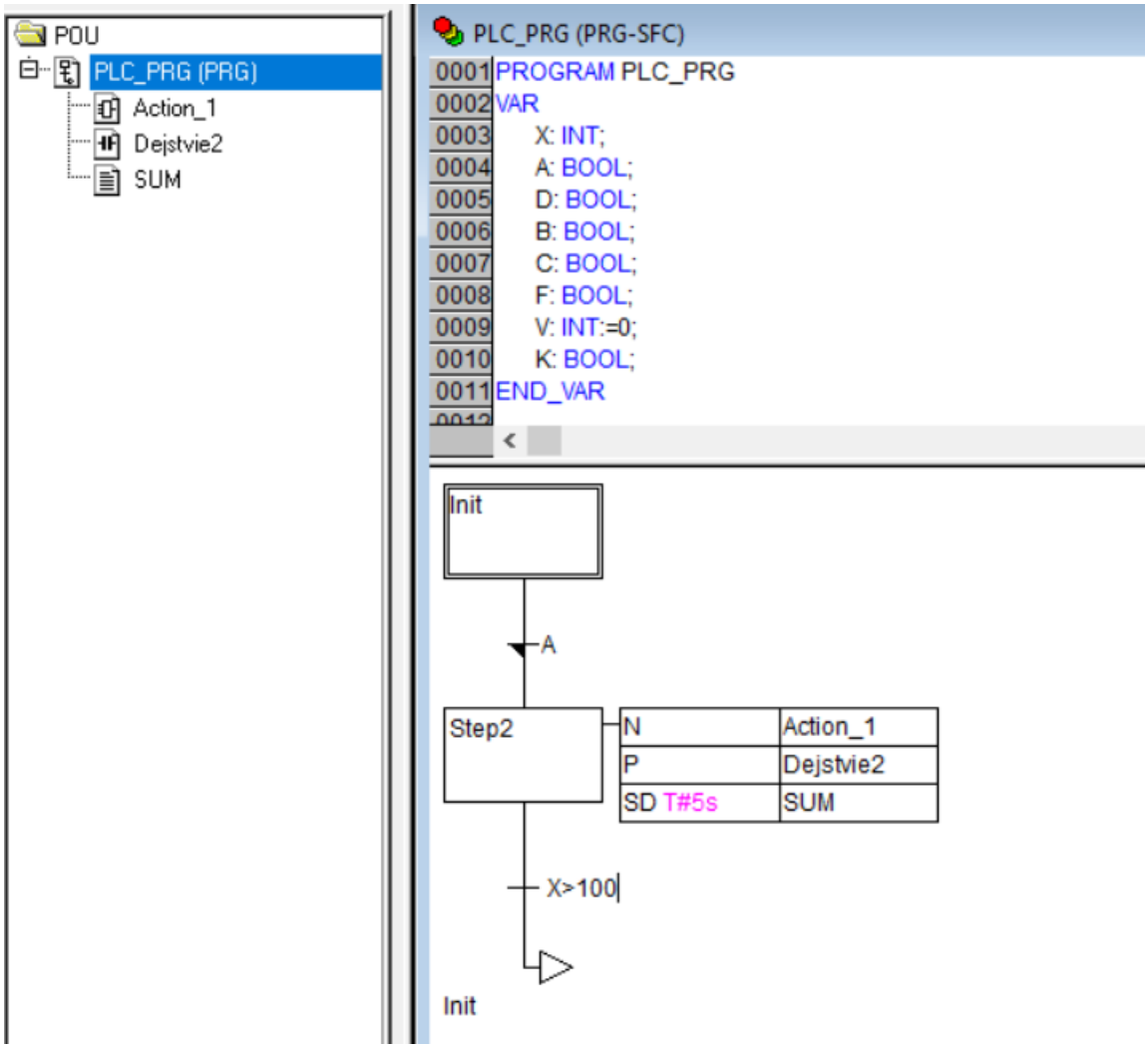


Рис. 16

7. Запустим проект на исполнение – Онлайн-Подключение. И увидим, что проект создан некорректно (рис. 17). Дело в том, что для работы с МЭК-шагами нужна специальная IEC-библиотека, которую необходимо подключить. Во вкладке проекта «Ресурсы» выбираем раздел «Менеджер библиотек» и через контекстное меню выбираем команду «Добавить библиотеку» (рис. 18). В списке библиотек выберем Iecsfc.lib и добавим её в проект (рис. 19). Вид этой библиотеки показан на рис. 20.

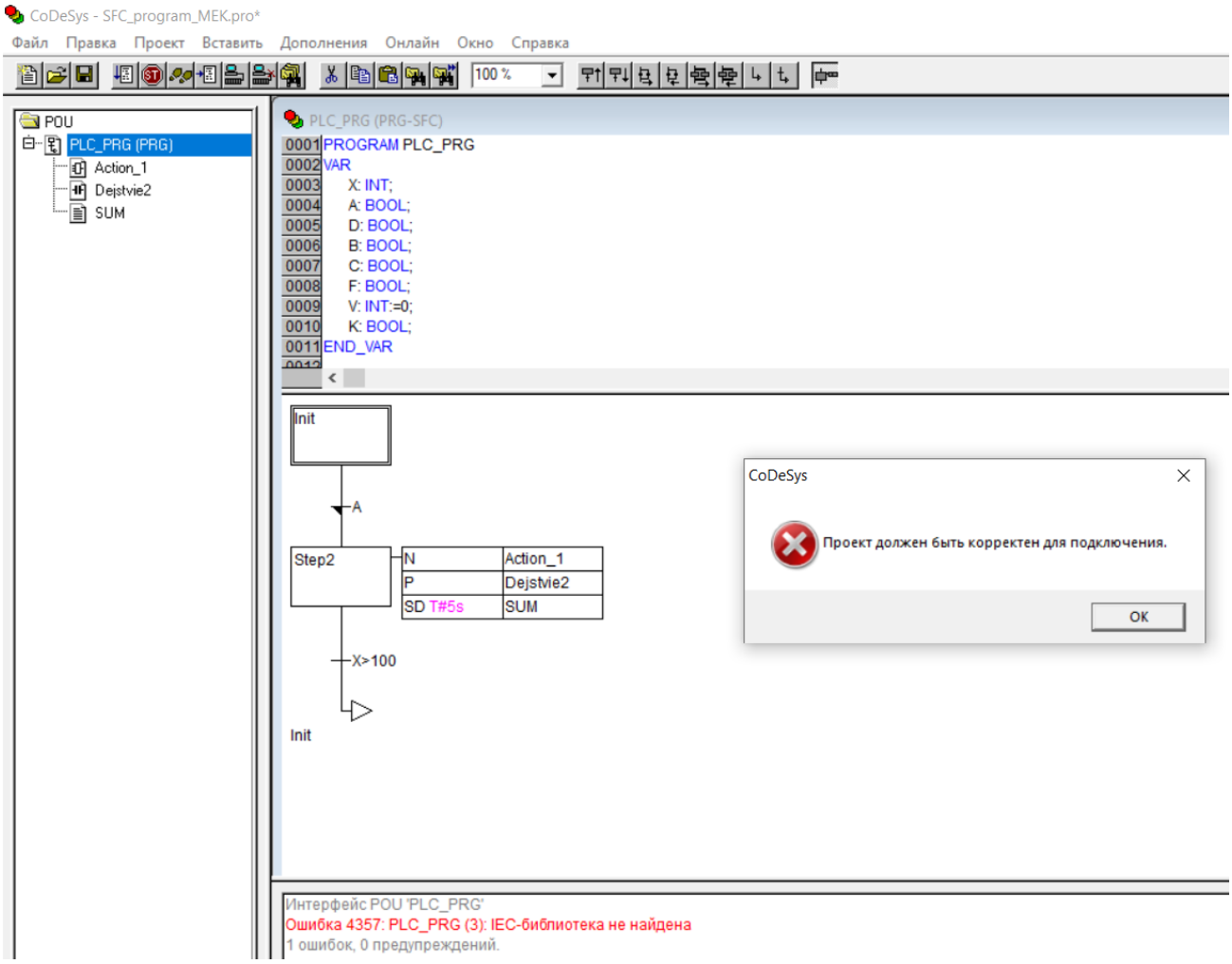

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
Action_1
Dejstvie2
SUM
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
0004 A: BOOL;
0005 D: BOOL;
0006 B: BOOL;
0007 C: BOOL;
0008 F: BOOL;
0009 V: INT:=0;
0010 K: BOOL;
0011 END_VAR
Init
A
Step2
N Action_1
P Dejstvie2
SD T#5s SUM
X>100
Init
CoDeSys
Проект должен быть корректен для подключения.
OK
Интерфейс POU 'PLC_PRG'
Ошибка 4357: PLC_PRG (3): IEC-библиотека не найдена
1 ошибок, 0 предупреждений.

Рис. 17

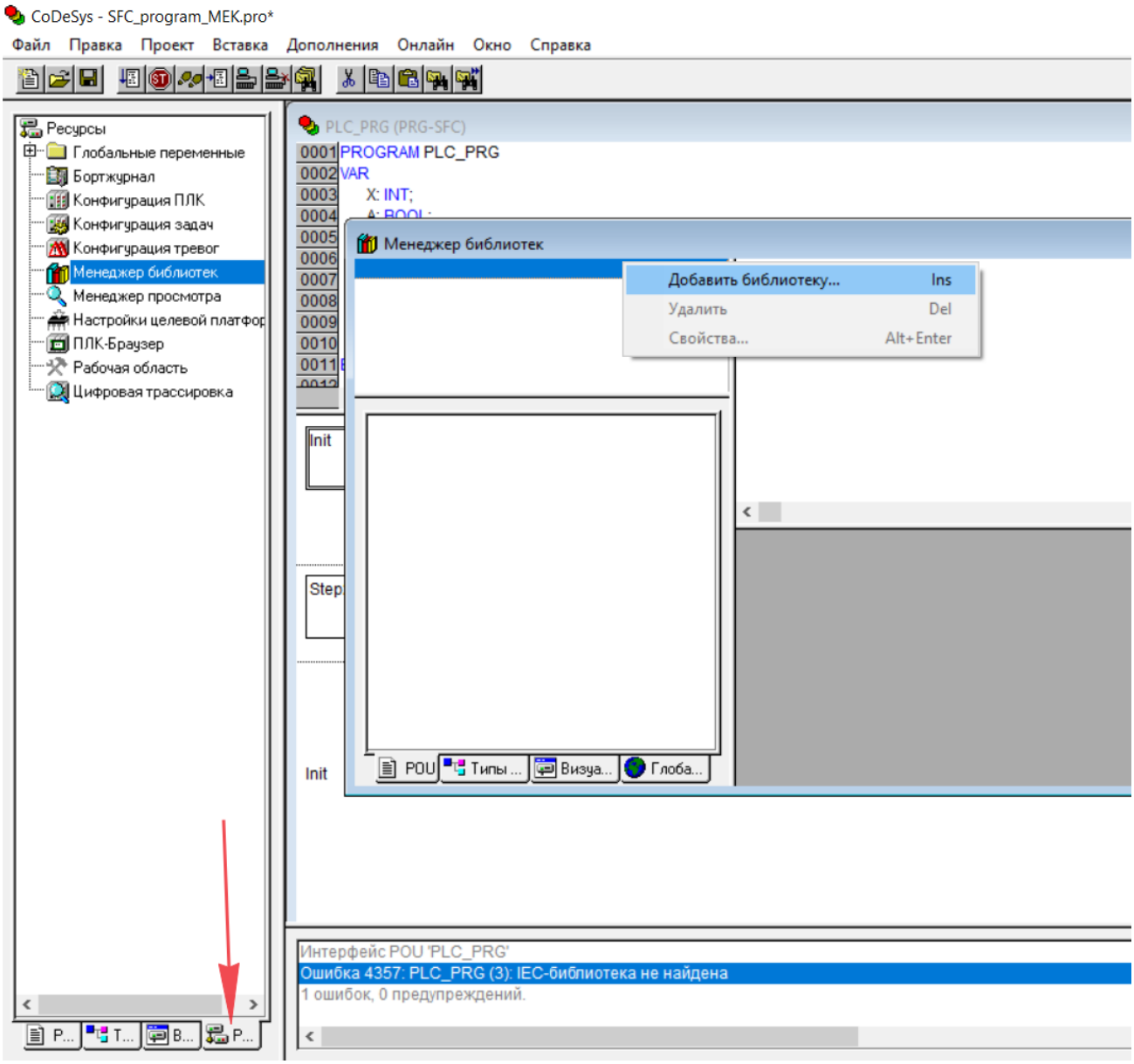

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставка Дополнения Онлайн Окно Справка
Ресурсы
Глобальные переменные
Бортжурнал
Конфигурация ПЛК
Конфигурация задач
Конфигурация тревог
Менеджер библиотек
Менеджер просмотра
Настройки целевой платфор
ПЛК-Браузер
Рабочая область
Цифровая трассировка
PLC_PRG (PRG-SFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003 X: INT;
Менеджер библиотек
Добавить библиотеку... Ins
Удалить Del
Свойства... Alt+Enter
Init
Step
Init
POU
Типы ...
Визуа...
Глоба...
Интерфейс POU 'PLC_PRG'
Ошибка 4357: PLC_PRG (3): IEC-библиотека не найдена
1 ошибок, 0 предупреждений.

Рис. 18

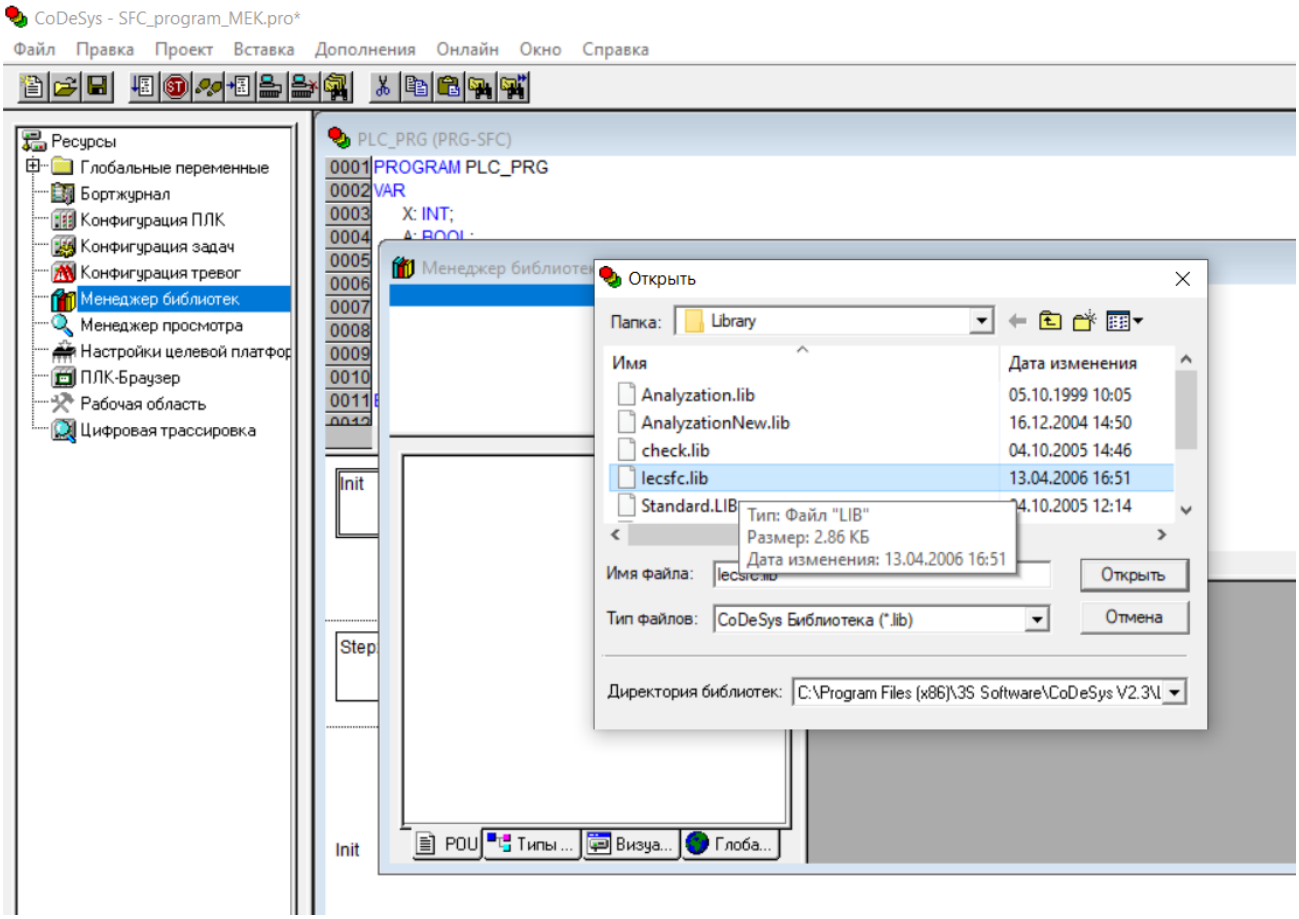


Рис. 19

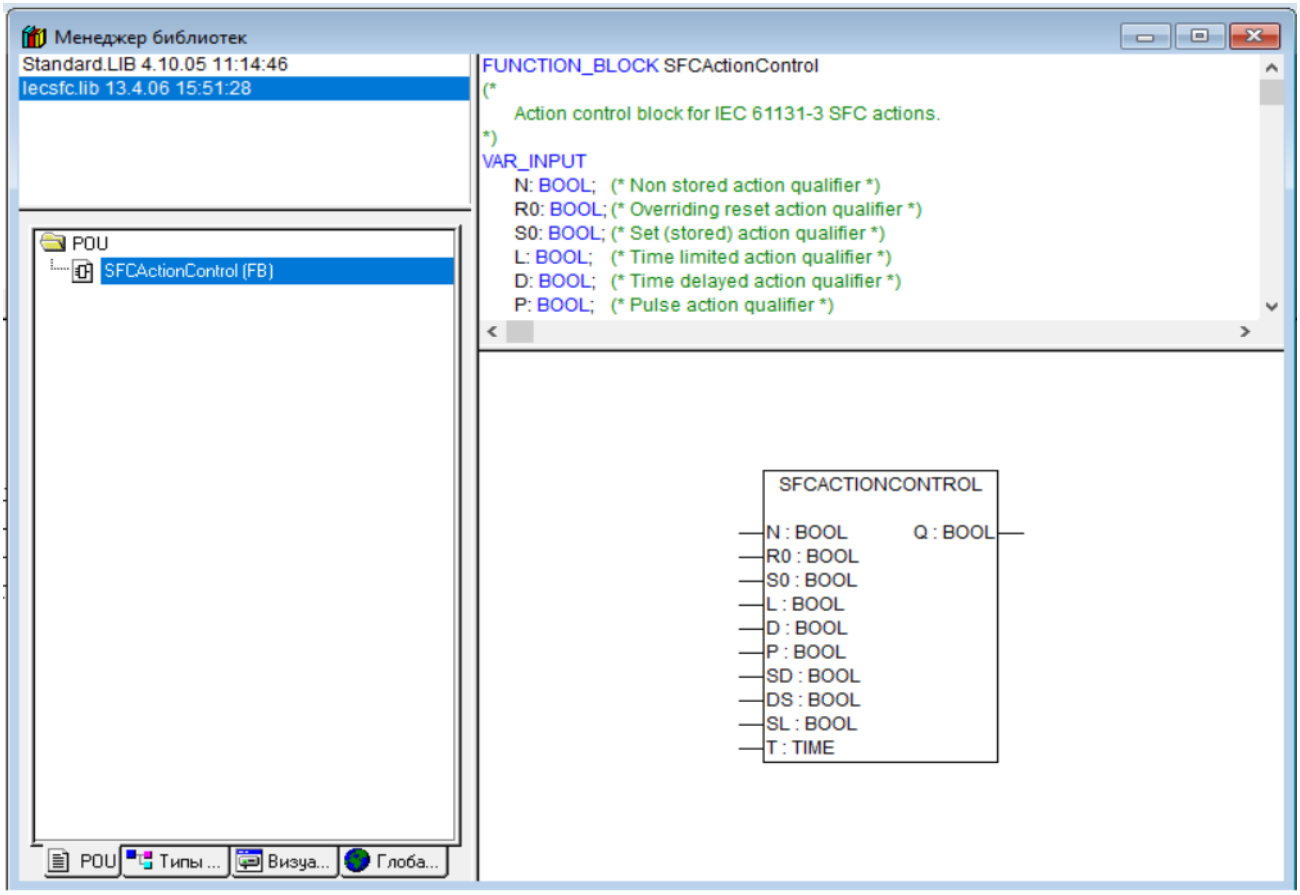


Рис. 20

8. Заново запустим проект на подключение, должен запуститься без ошибок. Так как для запуска шага Step2 требуется значение True для переменных A и C, то установим их в это значение (рис. 21). Не забываем ввести изменение значения переменной через команду

Ctrl+F7. При этом сразу выполнится действие Action_1 и будет активно в течение всей работы шага, а через 5 секунд начнет выполняться действие SUM (рис. 22). Как только X станет больше 100, запустится импульсный режим выполнения действий Action_1 и Dejstvie2 (будет «мигание») , а действие SUM будет постоянно активно (рис. 23). Выполнение действия SUM сейчас не зависит от того, работает ли шаг Step2, или не работает, так как у него стоит идентификатор SD – «сохраняемое и отложенное». То есть оно будет выполняться до тех пор, пока не сработает действие, помеченное идентификатором R – «внеочередной сброс». Но для этого нужно в программе заранее предусмотреть наличие такого идентификатора.

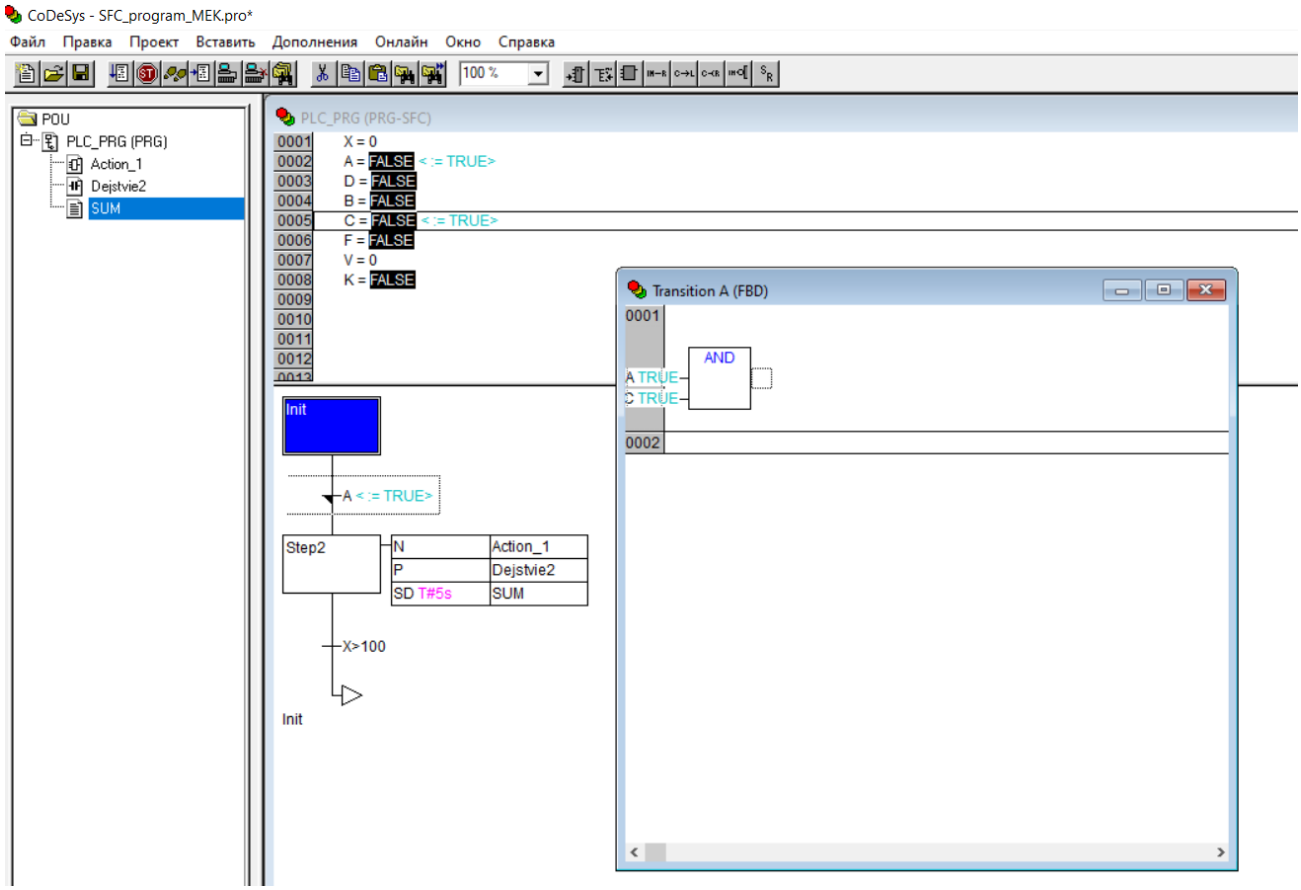


Рис. 21

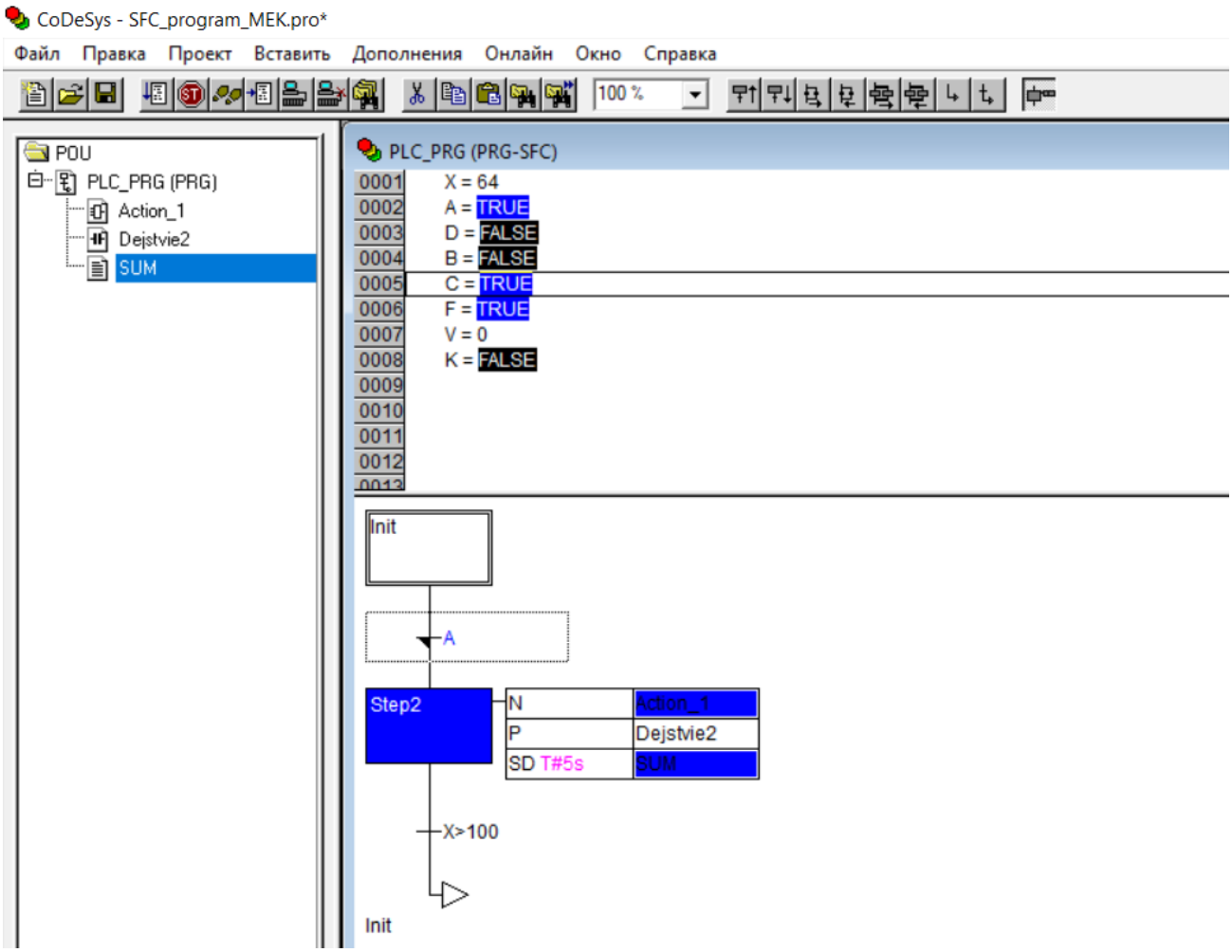

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
Action_1
Dejstvie2
SUM
PLC_PRG (PRG-SFC)
0001 X = 64
0002 A = TRUE
0003 D = FALSE
0004 B = FALSE
0005 C = TRUE
0006 F = TRUE
0007 V = 0
0008 K = FALSE
Init
A
Step2
N Action_1
P Dejstvie2
SD T#5s SUM
X>100
Init

Рис. 22

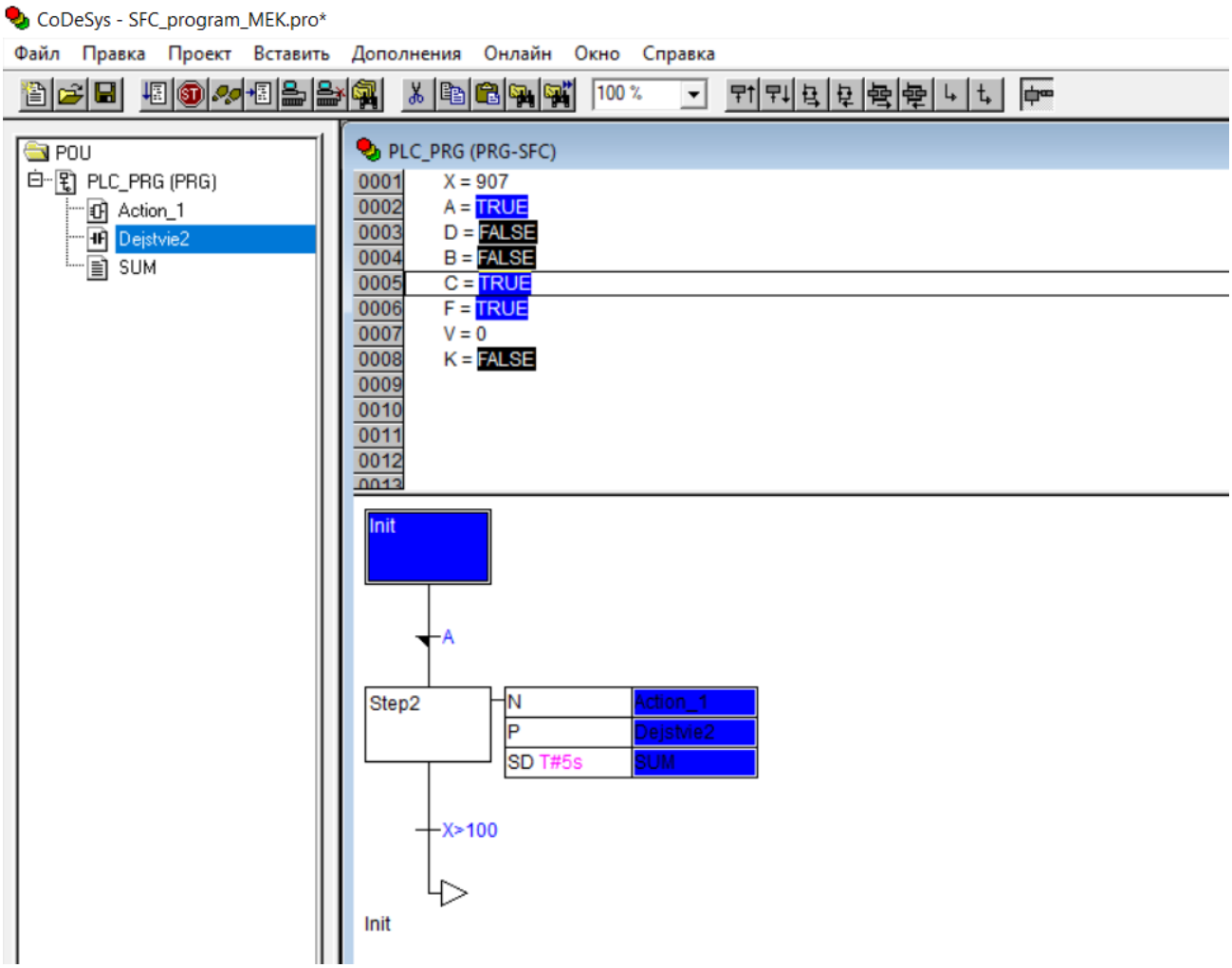

CoDeSys - SFC_program_MEK.pro*
Файл Правка Проект Вставить Дополнения Онлайн Окно Справка
100 %
POU
PLC_PRG (PRG)
Action_1
Dejstvie2
SUM
PLC_PRG (PRG-SFC)
0001 X = 907
0002 A = TRUE
0003 D = FALSE
0004 B = FALSE
0005 C = TRUE
0006 F = TRUE
0007 V = 0
0008 K = FALSE
Init
A
Step2
N Action_1
P Dejstvie2
SD T#5s SUM
X>100
Init

Рис. 23

Пример применения языка SFC с использованием МЭК шагов

Рассмотрим пример составления программы на языке SFC с использованием МЭК шагов для управления водоотливной установкой. Для этого создадим новый проект и сразу для него подключим библиотеку Iecsfc.lib (рис. 24).

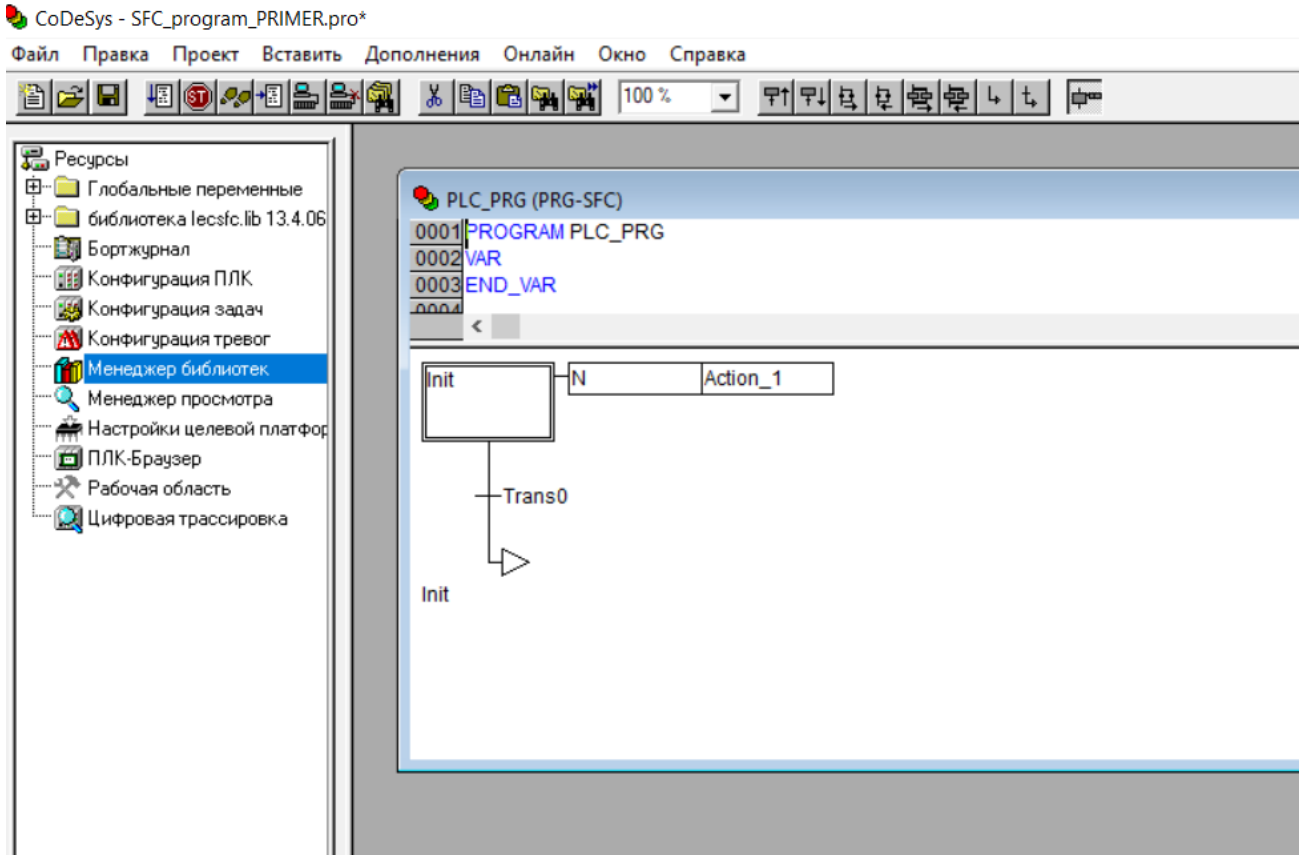


Рис. 24

В шаге Init нам не нужны будут никакие действия, поэтому их удалим через команду «Очистить действие» (рис. 25).

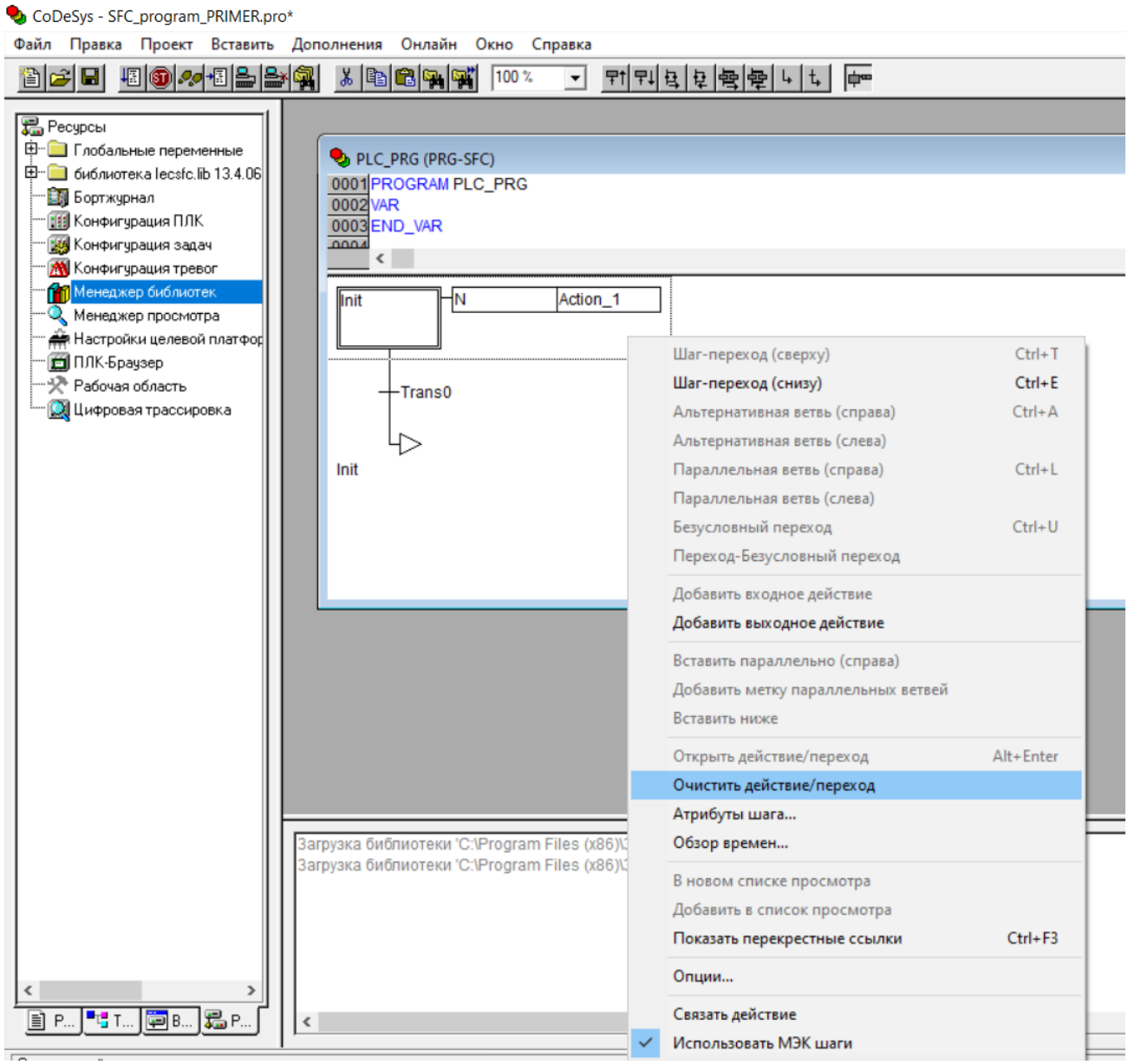


Рис. 25

Зададим условие перехода для нижнего уровня установки. За сигнал с датчика нижнего уровня будет отвечать переменная N_Y (рис. 26).

Если сигнал с датчика нижнего уровня отсутствует, то должны отключаться все насосы. Для этого создаём шаг Step2 и в нём прописываем действие Action_1 с идентификатором P (будет работать импульсно), язык реализации выбираем LD. В программе прописываем условия отключения трёх насосов N1, N2, N3 при отсутствии сигнала с датчика нижнего уровня (рис. 27). Кроме того, здесь же сразу предусмотрим сброс переменных TN2 и TN3, которые будут отвечать за отключение насосов, если истекло время их работы.

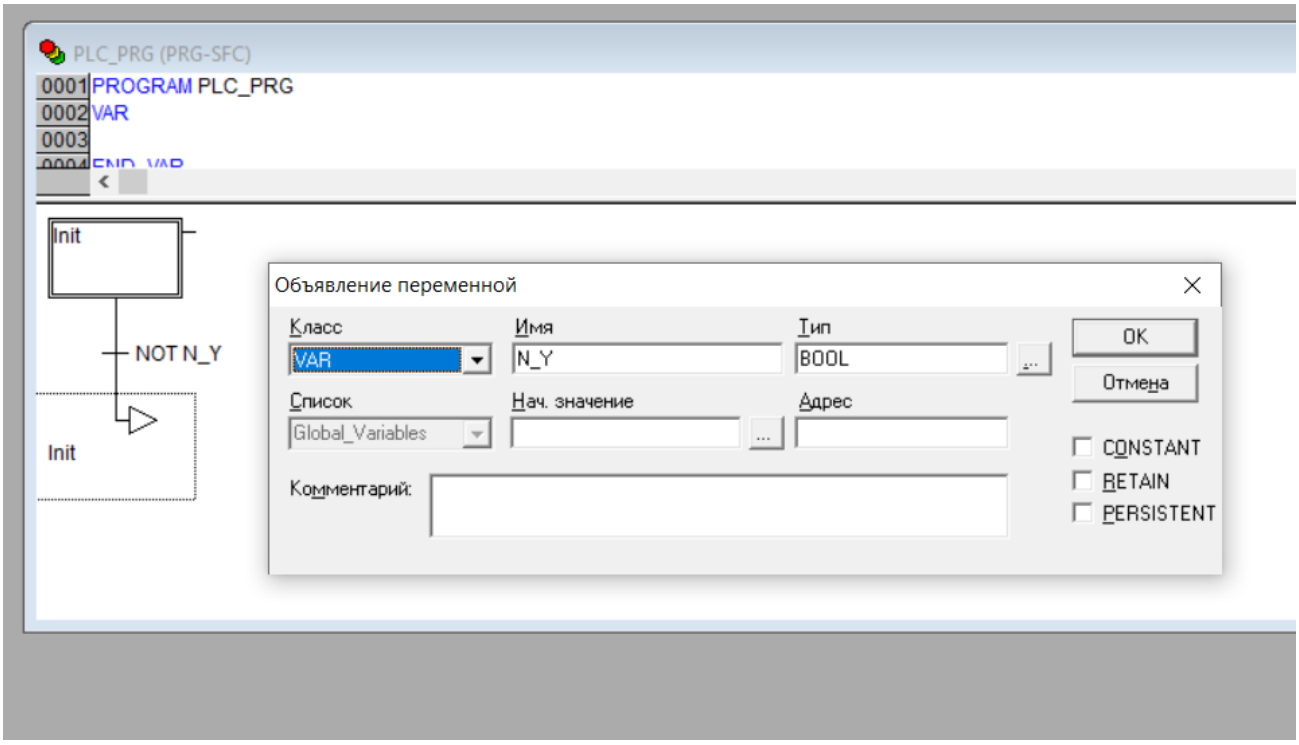


Рис. 26

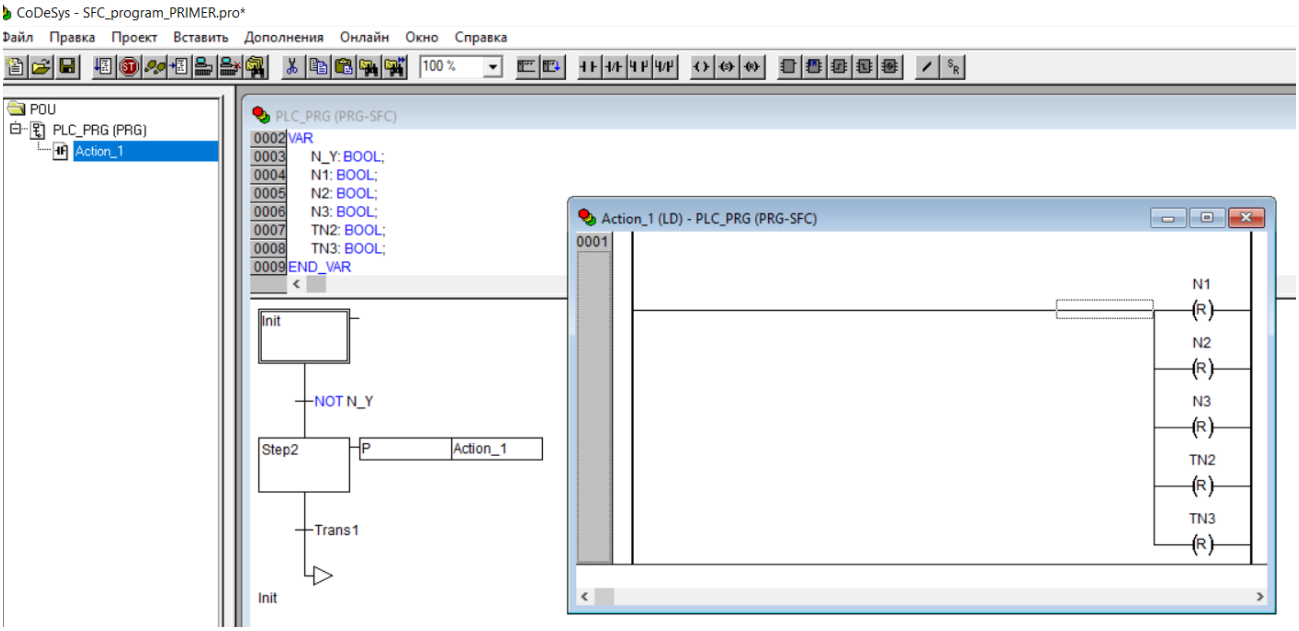


Рис. 27

Добавим переход на следующий шаг как условие появления сигнала с датчика нижнего уровня. Создадим шаг Step3, для него пропишем действие ACT2 с идентификатором N (то есть оно будет выполняться постоянно), язык реализации выберем ST и пропишем на нём основную логику работы системы управления водоотливной станцией.

Первая часть программы будет реализовывать условия включения насосов (рис. 28).

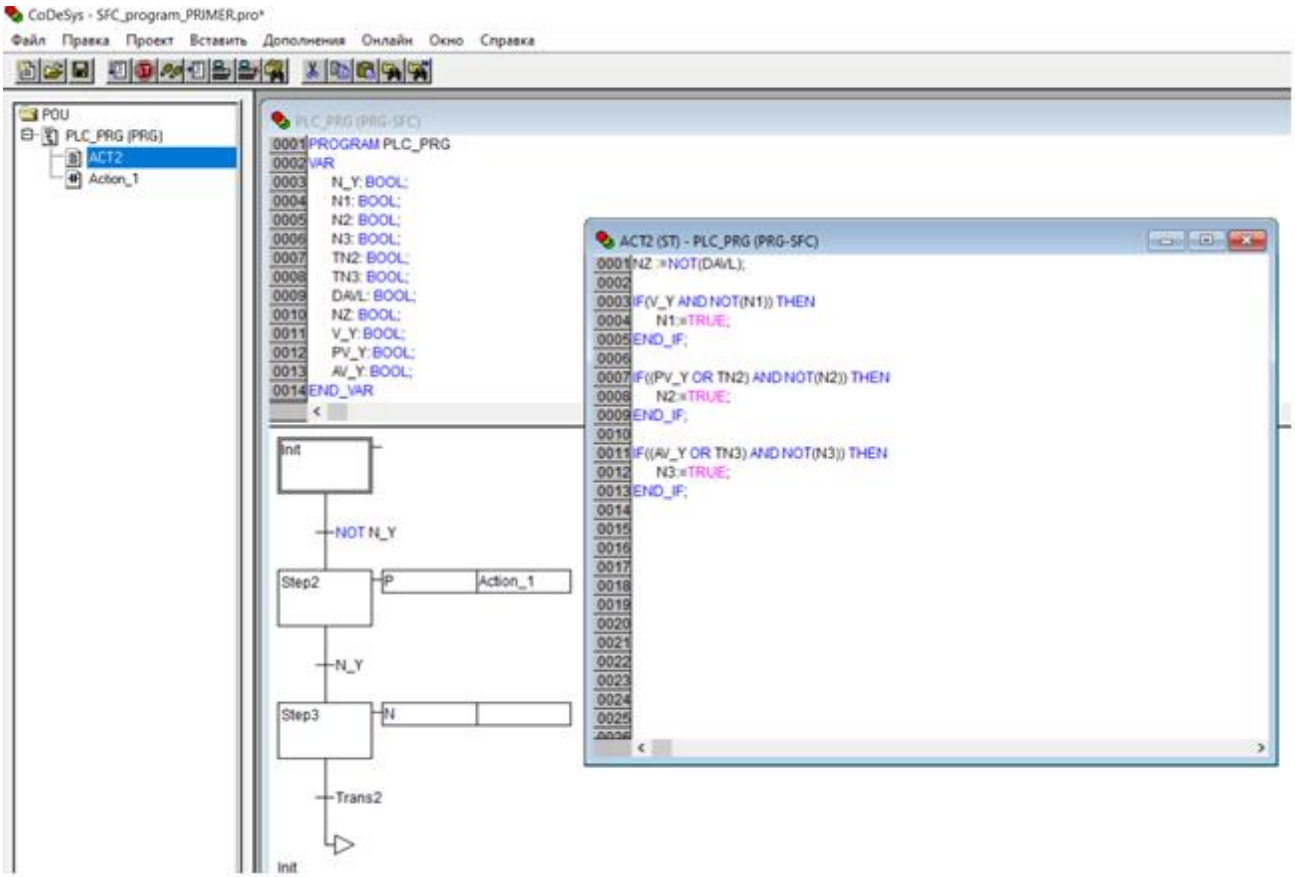


Рис. 28

Заливочный насос NZ включается, когда нет давления DAVL.

Далее проверяем условие: если есть сигнал с датчика верхнего уровня V_Y и насос N1 не включен, то нужно его запустить в работу.

Если появляется сигнал с датчика верхнего предупредительного уровня PV_Y и не работает насос N2, то нужно его также запустить в работу.

Если появляется сигнал с датчика уровня AV_Y и не работает насос N3, то нужно его также запустить в работу.

Далее создаём таймер TON1 типа TON через команду «Авто объявление» (вызов через контекстное меню) (рис. 29).

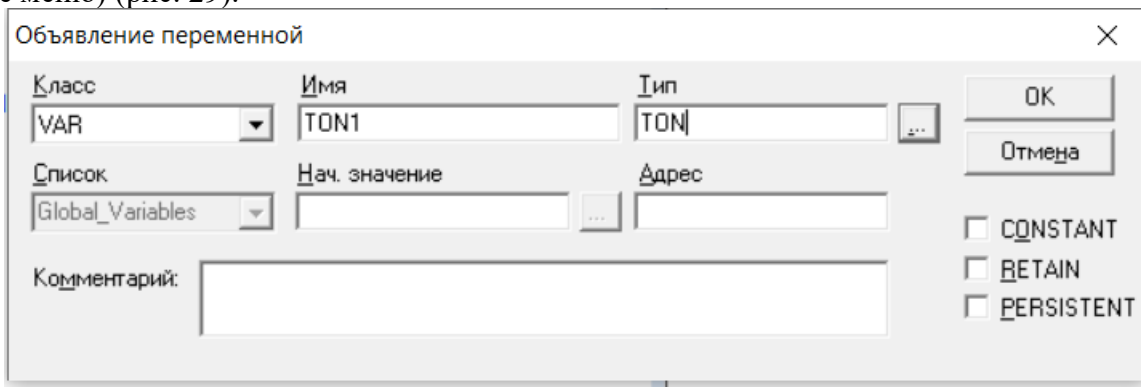


Рис. 29

Обратимся к таймеру TON1, указав его параметры: срабатывание по сигналу от насоса N1 через время T1, которое сразу задаём при создании этой переменной (рис. 30).

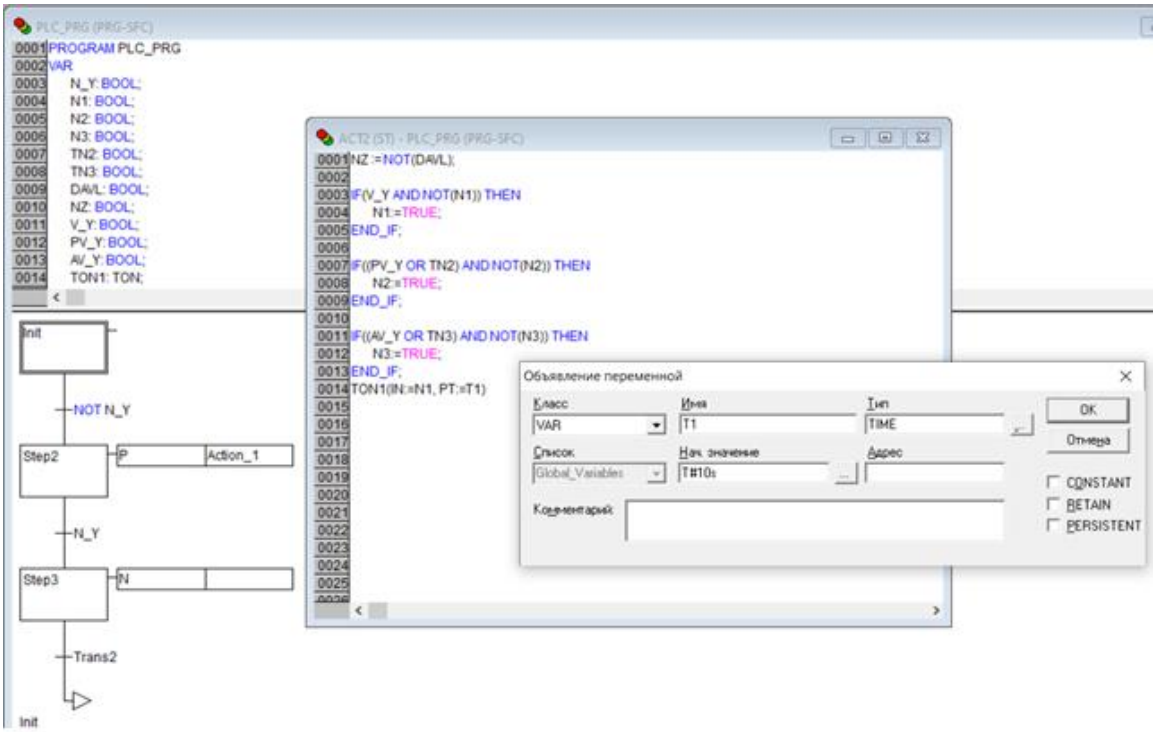


Рис. 30

Вызвать таймер можно и через ассистент ввода по кнопке F2: выбираем пункт «Локальные переменные» и в нём указываем таймер TON (рис. 31). При этом в программе на ST будет сразу сформирована конструкции вызова таймера, в которой нужно будет только привязать к его входам и выходам необходимые сигналы.

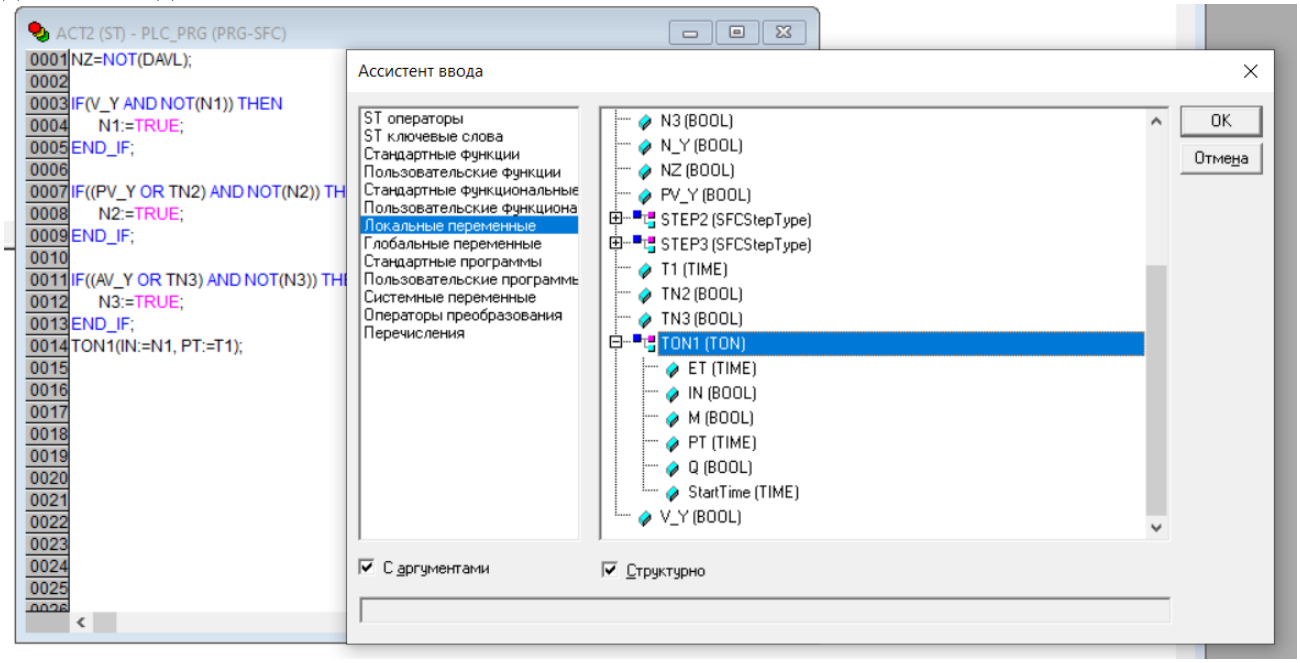


Рис. 31

Выход таймера TON1 свяжем с переменной отключения второго насоса TN2.

Аналогично вызовем таймер TON2, который будет срабатывать по сигналу от насоса N2 через время T2, также равное 10 секундам. И свяжем выход этого таймера с переменной отключения второго насоса TN3.

На этом описание действия ACT2 будет завершено (рис. 32).

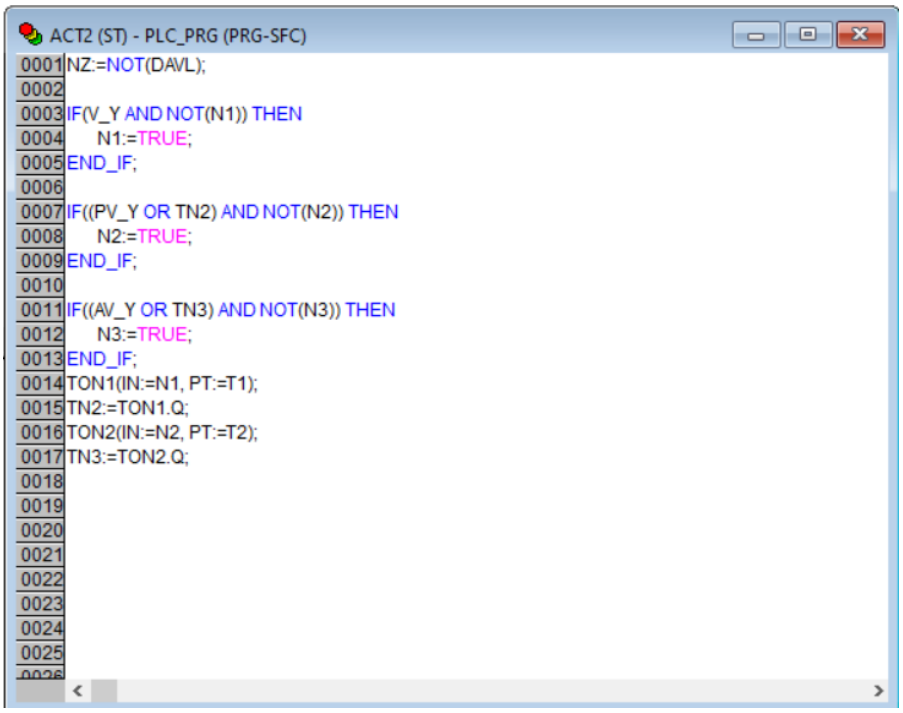


Рис. 32

Далее остаётся только прописать условие перехода после шага Step3 на шаг Step2 (рис. 33).

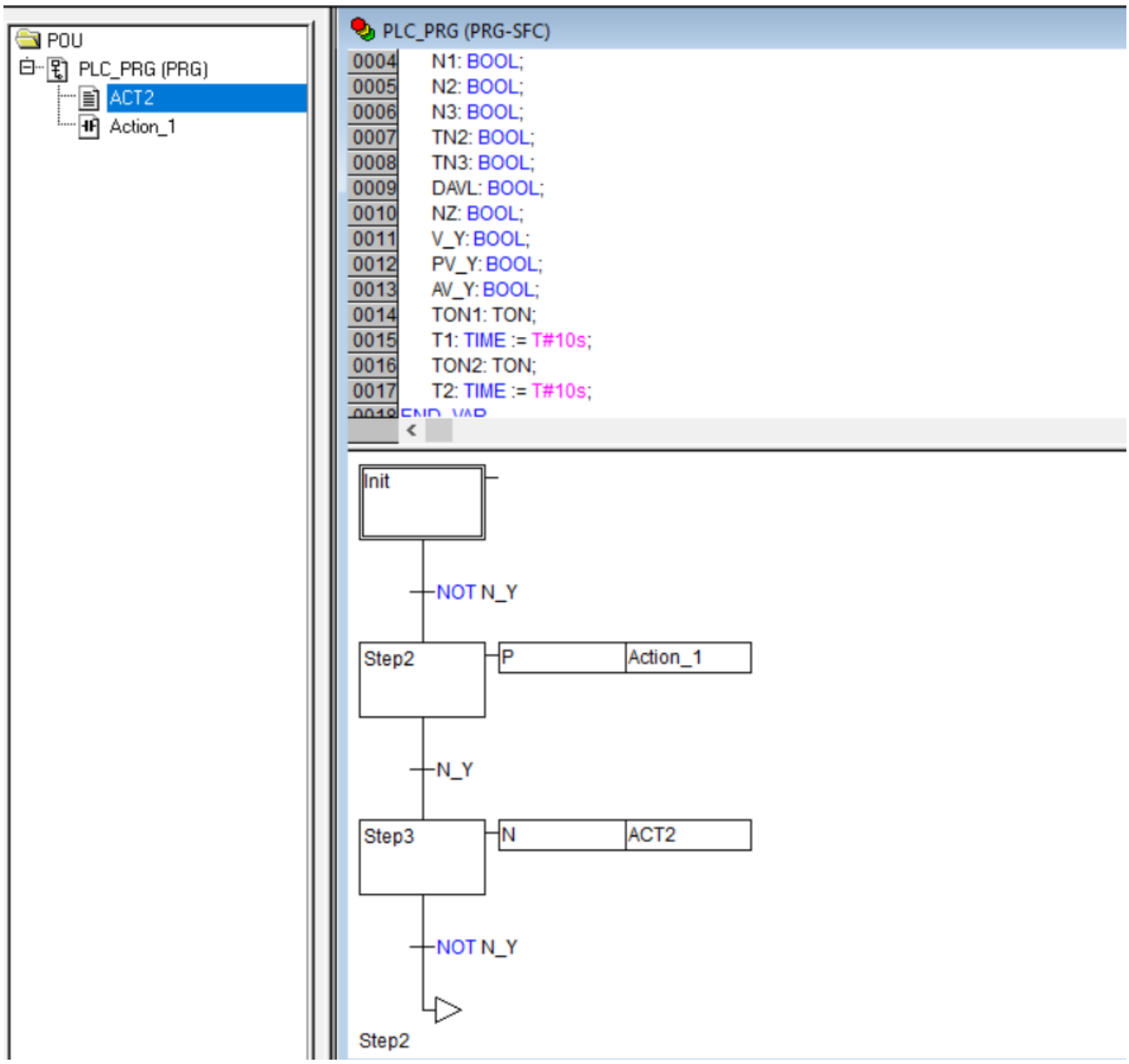


Рис. 33

Запустим проект на исполнение в режиме эмуляции. При подключении сразу же осуществляется переход на шаг Step2, так как N_Y:=FALSE. То есть сбрасываются все насосы.

Активируем сигнал с датчика нижнего уровня N_Y (рис. 34).

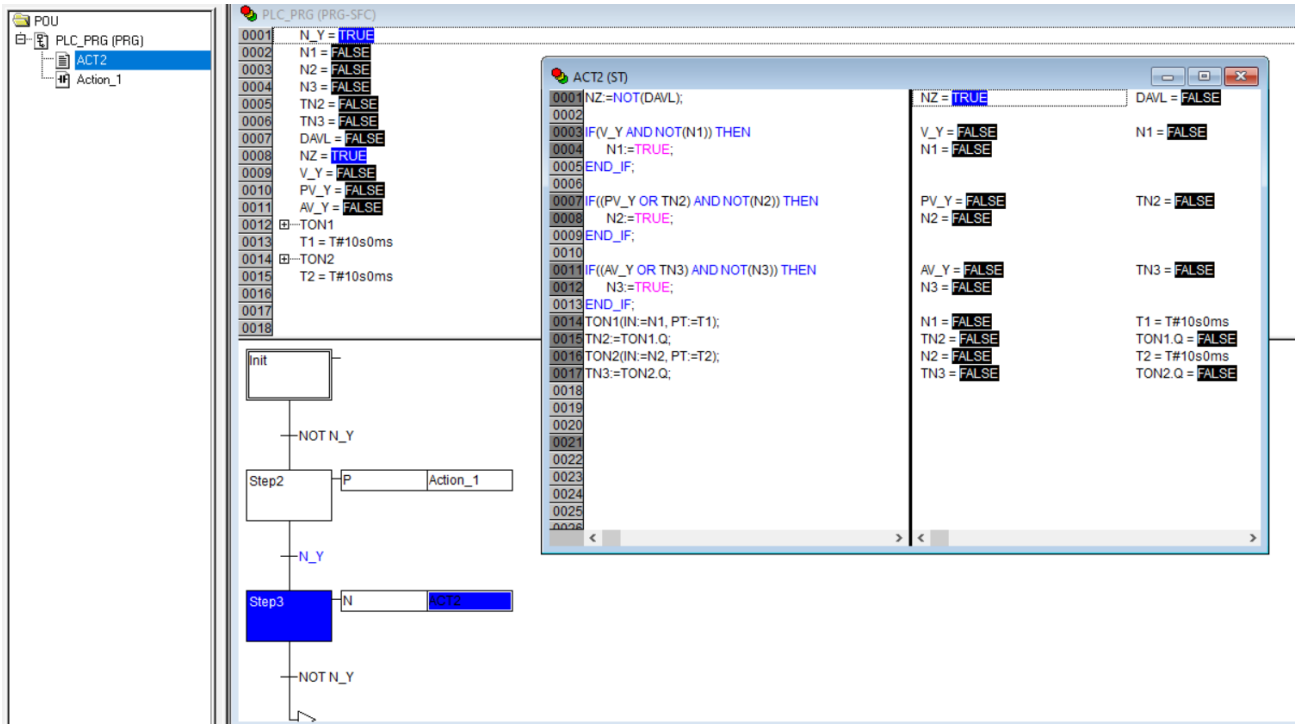


Рис. 34

Далее необходимо проверить работу всей системы управления:

- отключить сигнал давления (должен отключиться насос NZ);
- включить сигнал с датчика верхнего уровня (должен включиться насос N1);
- если верхний уровень не откачивается, то через 10 секунд должен подключиться насос N2;
- если же и второй насос не справился с задачей, то еще через 10 секунд должен включиться третий насос N3;
- все три насоса должны работать, пока не откачают дренажную яму – отключаем датчики N_Y и V_Y;
- после этого все насосы сбрасываются, так как произойдет переход на шаг Step2;
- проверить правильность включения насосов (не дожидаясь времени таймеров), если появятся последовательно сигналы с датчиков N_Y, V_Y, PV_Y, AV_Y;
- как только пропадут сигналы со всех датчиков уровня (произошла откачка воды), все насосы должны отключиться.

Кроме того, необходимо разработать визуализацию работы системы управления.

Содержание отчета

1. Код программы на языке SFC с использованием МЭК шагов для управления водоотливной установкой (скрины главного окна и действий).
2. Визуализация работы данной системы управления с подробным описанием всех элементов.

Практическая работа №6

Реализация системы управления ленточным конвейером в среде CoDeSys

1. Создание проекта

Для того, чтобы создать новый проект, нужно запустить среду программирования CoDeSys и выбрать *Файл – Создать* или нажать кнопку **Создать** на панели инструментов (рис.1).

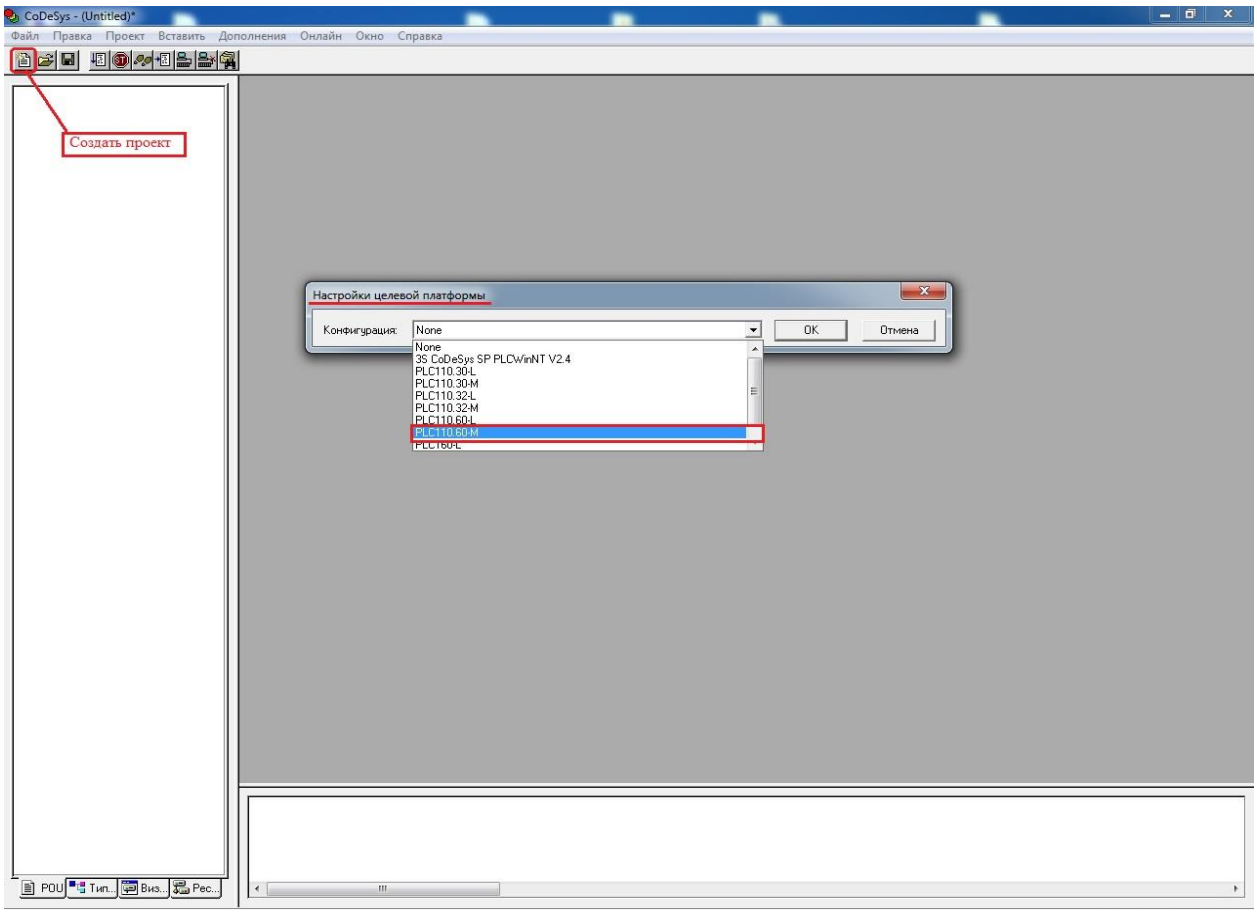


Рис.1. Создание проекта в CoDeSys и определение целевой платформы

После выбора соответствующей целевой платформы (выделена красным на рис.1) среда программирования предложит настроить конфигурацию выбранного контроллера. Зачастую используются параметры по умолчанию. Наибольший интерес представляет параметр *Загрузить символьный файл* на вкладке **Общие** (рис.2). Этот параметр позволяет активировать импорт/экспорт данных для связи с OPC-сервером SCADA-системы верхнего уровня, если таковой имеется.

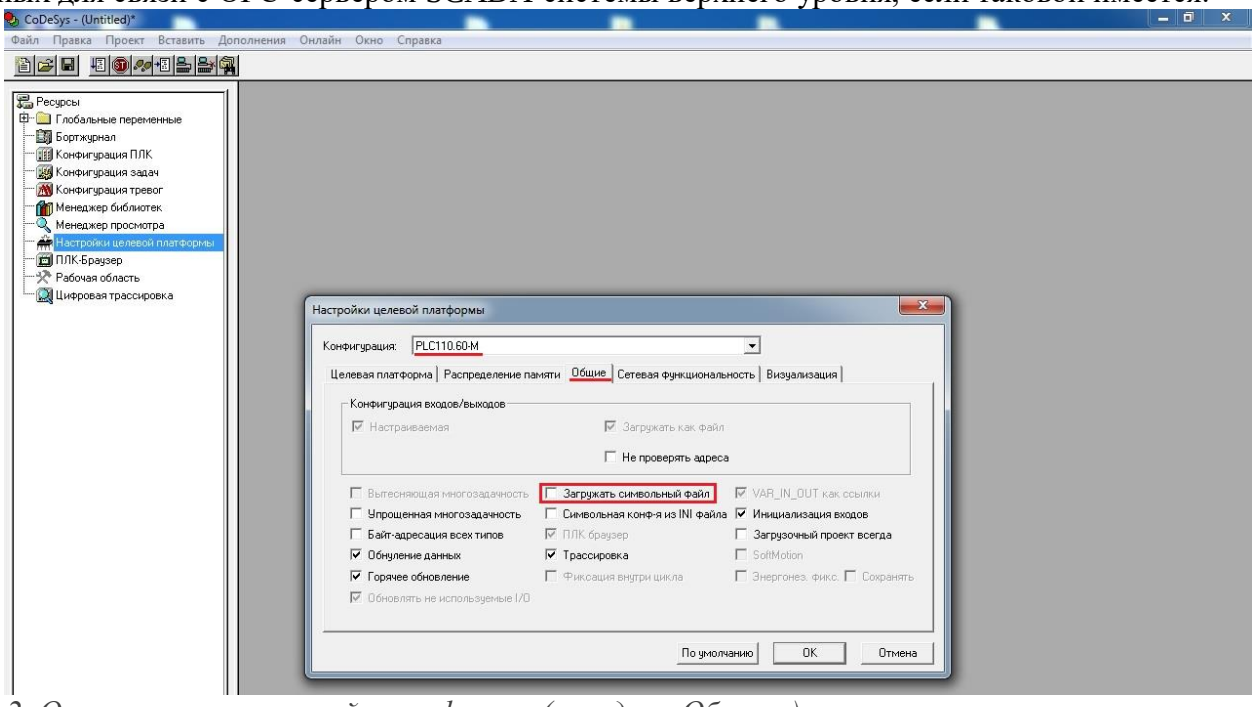


Рис.2. Окно настроек целевой платформы (вкладка «Общие»)

После выбора настроек целевой платформы среда программирования предложит вам создать *Программу* **PLC_PRG** и выбрать один из языков программирования логических контроллеров (рис.3). В данном проекте мы будем создавать программу на **LD** – языке линейных схем.

ВНИМАНИЕ: ЗАПРЕЩЕНО ИЗМЕНЯТЬ ИМЯ ОБЪЕКТА PLC_PRG.
Программа **PLC_PRG** является основной программой контроллера при однозадачной конфигурации, она запускается на исполнение при старте контроллера и из нее вызываются все остальные функции, функциональные блоки и подпрограммы. В случае отсутствие объекта типа *Программа* с именем **PLC_PRG** при запуске контроллера НИЧЕГО НЕ ПРОИЗОЙДЕТ. Исключение составляют случаи, когда программист определяет многозадачную систему управления и самостоятельно задает программу, запускающуюся при старте контроллера, или же если проект не предназначается для запуска на контроллере (к примеру, будет скомпилирован как библиотека функциональных блоков).

В программе **PLC_PRG** мы будем вызывать созданные в процессе программирования функциональные блоки и создавать логику работы системы управления.

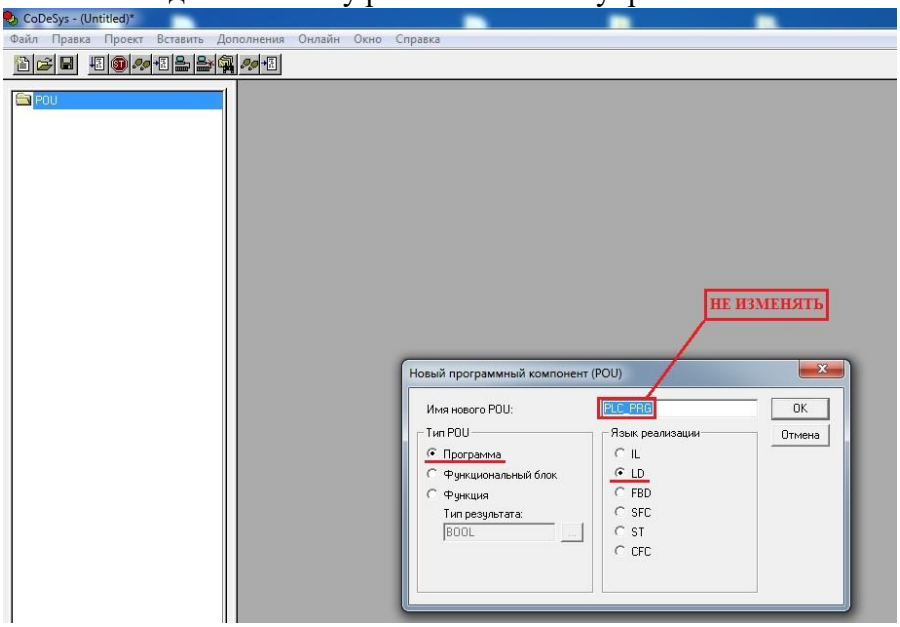


Рис.3. Создание основной рабочей программы ПЛК – PLC_PRG

Теперь, после создания основной программы проекта, можно приступить непосредственно к программированию. Но прежде стоит посмотреть, что же представляет из себя ПЛК с точки зрения среды программирования, т.е. каким образом конкретный ПЛК с указанной целевой платформой представлен в среде программирования и каким образом можно использовать имеющиеся аппаратные средства. Для этого нужно на вкладке **Ресурсы** окна *Менеджер объектов* (слева от рабочей области) выбрать пункт ***Конфигурация ПЛК***.

В появившемся окне (рис.4) вы увидите 36 дискретных входов, 4 их которых быстрые (например, для подключения энкодера); 24 дискретных выхода, 4 из которых быстрые (например, для импульсной модуляции); а также специальные входы и выходы для самодиагностики ПЛК.

Кроме того, в *Конфигурации ПЛК* можно добавить необходимые объекты (подэлементы ПЛК), которые реализуют взаимодействие с последовательными портами ПЛК. Это могут быть, в частности, устройства типа Modbus Master с подключением соответствующего интерфейса (RS485 или RS232) для опроса дополнительных модулей или других устройств по последовательной шине. Таким образом, в окне *Конфигурация ПЛК* мы можем задать параметры работы с аппаратными средствами контроллера и подключенных к нему устройств, а также задать переменные для ввода данных в программу.

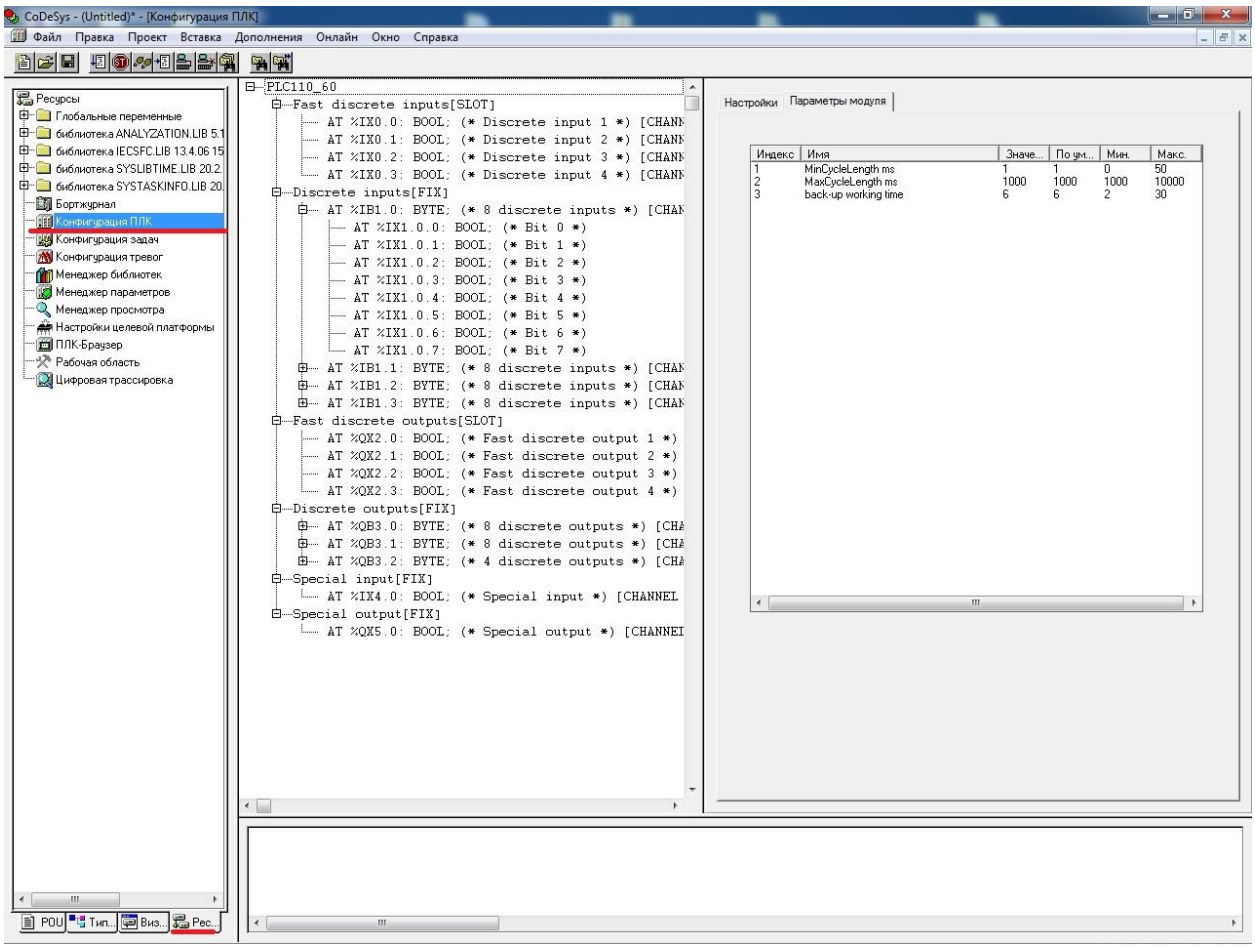


Рис.4. Начальная конфигурация ПЛК110.60

Еще один момент, важный для программирования логических контроллеров, да и вообще программирования в целом – это использование библиотек, как стандартных (созданных официальными разработчиками, в частности, компаниями 3S Software и ОВЕН), так и пользовательских (созданных самим программистом).

Для работы с библиотеками на вкладке **Ресурсы** предусмотрен пункт *Менеджер библиотек* (рис.5).

В данном окне можно подключать и отключать библиотеки, просматривать элементы библиотек, просматривать список переменных функциональных блоков и их внешний вид в графических редакторах программного кода (LD, FBD, CFC и SFC).

Из окна *Менеджер библиотек* **нельзя** вставить функциональный блок в программный код. *Менеджер библиотек* предназначен ***только для просмотра содержимого библиотек и работы с файлами библиотек (*.lib)***. Для вставки требуемого функционального блока в программу нужно выбрать в контекстном меню ***Вставить – Элемент***.

Чтобы создать пользовательскую (т.е. свою) библиотеку, нужно создать обычный проект. При этом, в данном проекте может отсутствовать целевая платформа, а также основная программа – сама по себе библиотека должна быть универсальна, и она не будет запускаться на реальном контроллере. Создание пользовательских библиотек довольно удобная функция, однако следует иметь ввиду, что редактирование объектов библиотеки из проекта, к которому она привязана, невозможно.

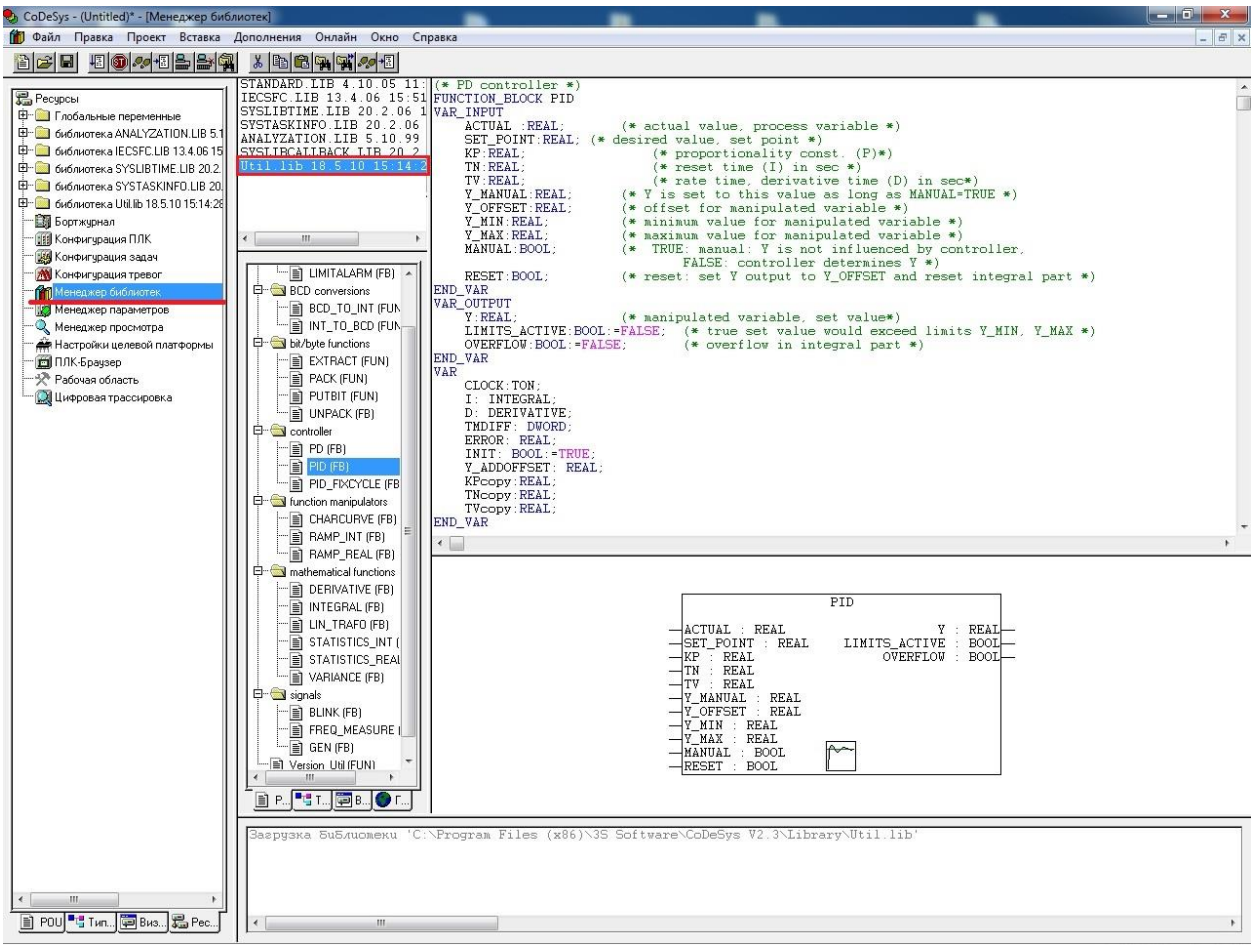


Рис.5. Менеджер библиотек

Для дальнейшей работы нам понадобится библиотека ***Util.lib***, которая выделена на рис.5 красным цветом, однако отсутствует в стандартном проекте. Чтобы ее подключить, нужно щелкнуть правой кнопкой мыши на области имен библиотек и в контекстном меню выбрать пункт ***Добавить библиотеку***. Откроется стандартный диалог выбора файла в каталоге по умолчанию, где и находится нужная нам библиотека. Остается только выбрать файл с именем *Util*. Теперь, когда библиотека подключена, мы можем просмотреть ее элементы. Опять-таки, чтобы добавить нужный элемент в программный код нужно воспользоваться контекстным меню в окне редактора или же кнопкой на панели инструментов, добавить элемент из менеджера библиотек нельзя.

Еще одним полезным, точнее необходимым элементом в программировании контроллеров, да и программировании в целом, являются глобальные переменные. Для них на вкладке **Ресурсы** выделена отдельная папка с соответствующим названием (рис.6).

Глобальные переменные – это переменные, видимые во всем проекте, и существующие все время исполнения программы. В папке *Глобальные переменные* можно создавать списки глобальных переменных, группируя эти переменные в соответствии с логикой их использования. У глобальных переменных могут быть модификаторы ***RETAIN***, который говорит о том, что данная переменная или группа переменных хранится в энергонезависимой памяти контроллера (соответственно не сбрасывается при пропадании питания), и ***PERSISTANT***, который говорит о том, что переменная устойчива к холодному сбросу. Эти модификаторы могут использоваться совместно.

Использование переменных с этими модификаторами в объявлении объектов также допускается, однако не рекомендуется. Правила работы с этими модификаторами, а также работы с модификаторами прав доступа ({noread}, {nowrite}) можно найти в справке среды программирования CoDeSys.

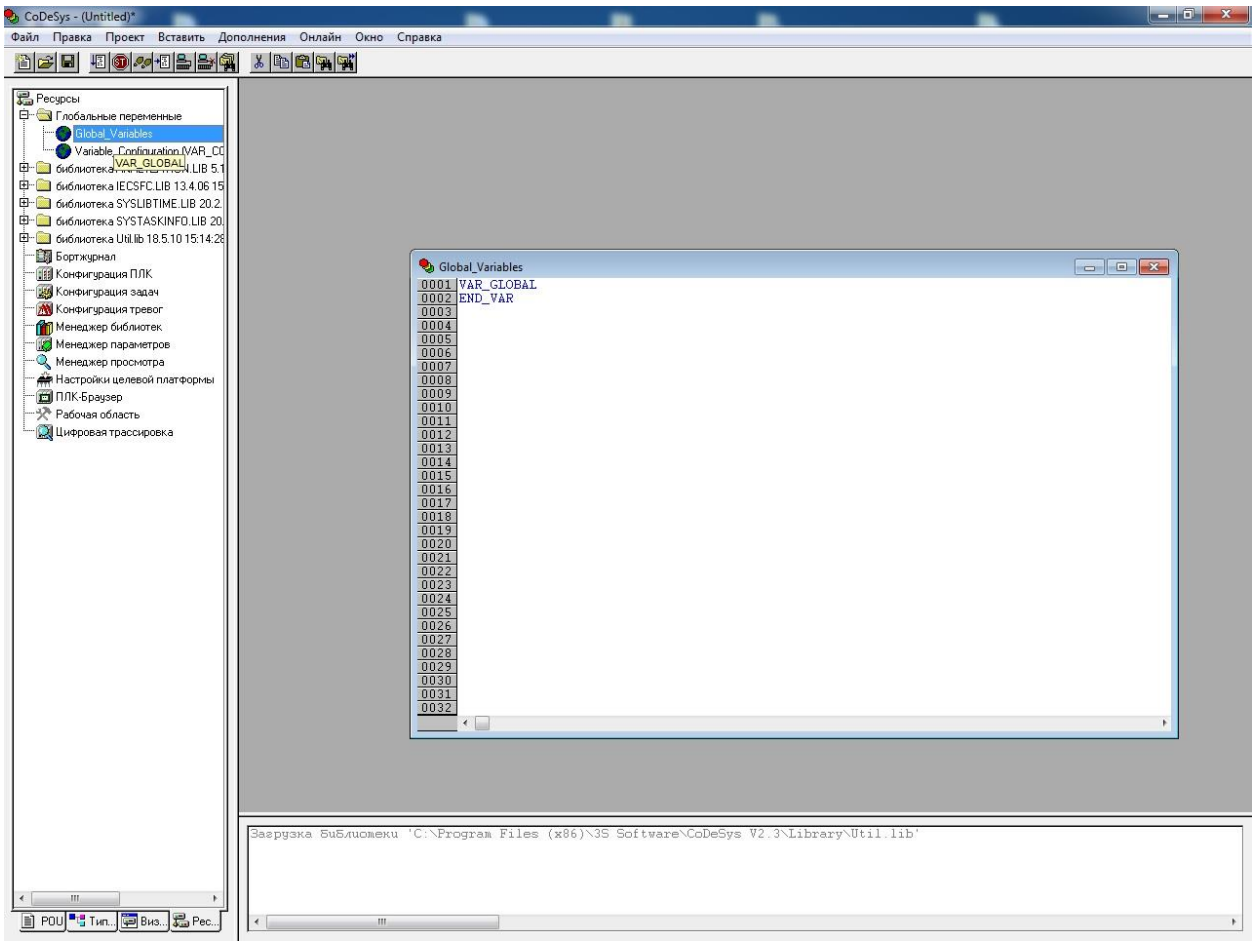


Рис.6. Глобальные переменные проекта

Итак, после освоения базовых практических основ можно перейти к созданию пользовательского функционального блока.

Функциональный блок – это структурная единица программного кода, которая на основании входных данных посредством заложенных алгоритмов формирует выходные данные. Отличием функционального блока от функции является множество выходных параметров (функция возвращает только одно значение).

Функциональный блок может быть написан на языке, отличном от основного языка проекта (на котором написана программа PLC_PRG). Создавая функциональный блок, мы, по аналогии с программированием в C++, создаем пользовательский класс или, если проще, тип данных. Хотя в реальности функциональный блок более похож именно на класс.

У функционального блока имеются входные параметры (VAR_INPUT), выходные параметры (VAR_OUTPUT) и локальные переменные (VAR). Также могут использоваться переменные, являющиеся одновременно и входными, и выходными, однако использование таких параметров нежелательно, т.к. по сути является переопределением внешней переменной, что может привести к сложно выявляемым ошибкам работы программы.

Можно отметить, что локальные переменные могут объявляться с модификатором RETAIN, при этом весь функциональный блок будет помещен в энергонезависимый сегмент. Однако, это тоже нежелательно, т.к. объем энергонезависимой памяти ограничен.

Каждый функциональный блок должен выполнять определенные логические функции, причем создание функционального блока оправдано, если эти функции необходимо выполнить в программе неоднократно. Таким образом, функциональный блок, как и класс, должен обладать достаточной универсальностью.

2. Создание блока управления трехфазным асинхронным электродвигателем с прямым пуском

Создадим функциональный блок с именем **PRIVOD** на языке программирования **LD** (рис.7).

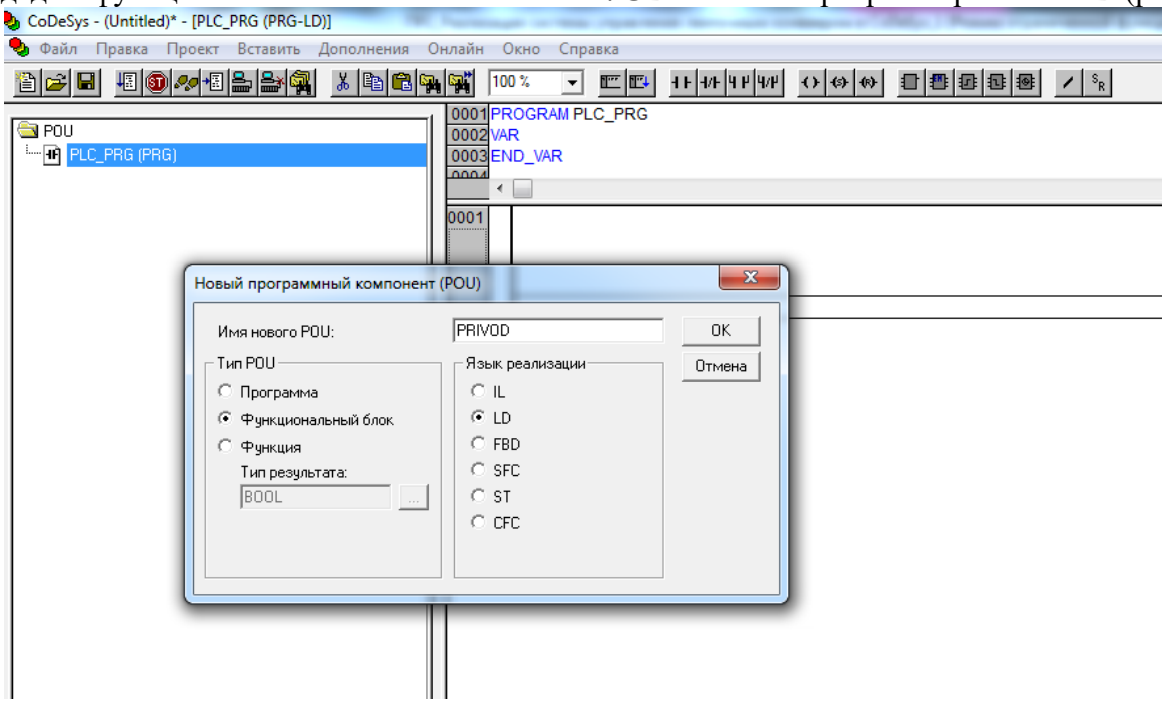


*Рис. 7. Создание функционального блока с именем **PRIVOD***

Пример принципиальной электрической схемы включения электродвигателя с прямым пуском приведен на рис.8.

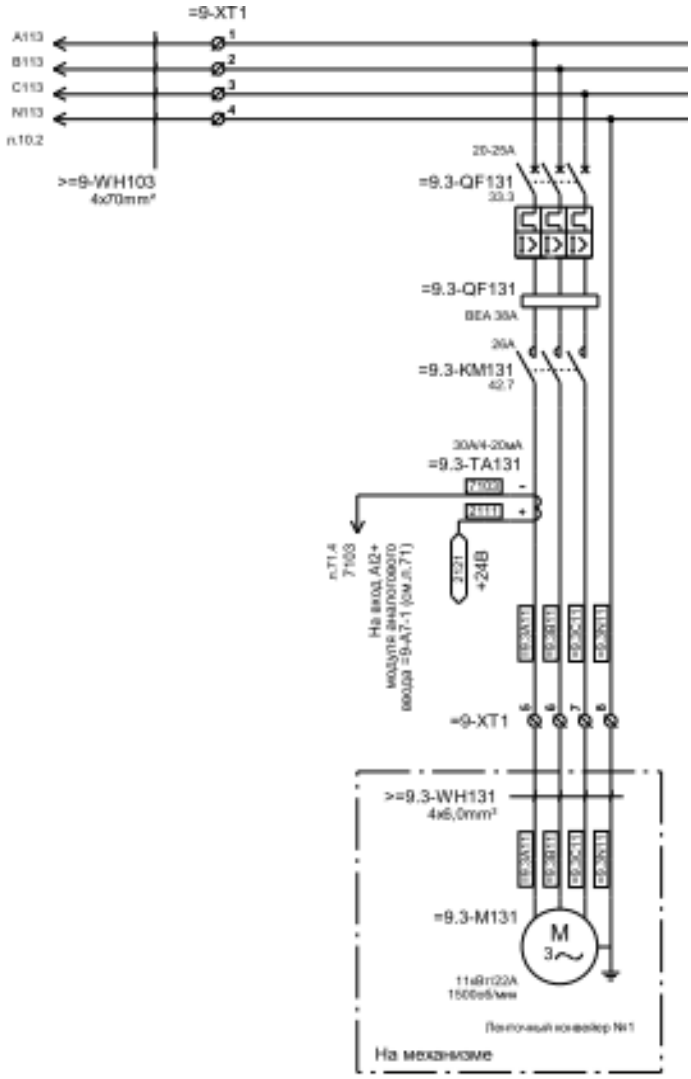


Рис.8. Пример принципиальной электрической схемы неуправляемого электропривода

Для включения трехфазного асинхронного электродвигателя с короткозамкнутым ротором (далее «двигателя», позиция на рисунке =9.3-М131) по схеме по схеме «прямого пуска» требуются автоматический выключатель для защиты двигателя (позиция =9.3-QF131), выполняющий функции защиты от токов короткого замыкания и перегрузки двигателя (т.е. выполняет еще и функцию теплового реле), а также контактор (позиция =9.3-КМ131). С дополнительных контактов автомата и контактора подаются сигналы на вход ПЛК для контроля состояния пусковой аппаратуры. Катушка контактора подключается к выходу ПЛК.

Сигнал с автоматического выключателя свидетельствует о том, что автоматический выключатель включен (т.е. нет аварийного отключения по току короткого замыкания или току перегрузки). Таким образом, отсутствие этого сигнала следует отнести к аварийной ситуации.

Сигнал с контактора свидетельствует о том, что на электродвигатель подано напряжение, т.е. что двигатель включен и соответствующий технологический агрегат запущен. Данный сигнал используется в цепи «самоподхвата» (включается параллельно с пусковой кнопкой).

Включается контактор сигналом с выхода ПЛК, или же через промежуточное реле, если параметры выходов контроллера не позволяют включить катушку контактора напрямую. Логика включения контактора определяется исходя из технологии автоматизируемого объекта.

Помимо силовой пусковой аппаратуры необходимы управляющие сигналы – сигнал запуска и сигнал останова. Эти сигналы могут приходить от разных устройств, их может быть несколько. Как правило, необходимо реализовать как минимум два источника сигналов: сигналы с местного поста управления и сигналы с поста оператора. Это могут быть также сигналы из SCADA-системы. Таким образом, необходимо реализовать логику управления для всех источников сигналов с учетом режима их работы (наладочный, ручной или автоматический).

Также нужно реализовать отработку аварийных сигналов, которые приходят от соответствующих первичных преобразователей, датчиков и вторичных приборов. Количество этих сигналов зависит от конкретного технологического агрегата и его технического оснащения. Как упоминалось выше, к аварийным сигналам также можно отнести отсутствие сигнала с автоматического выключателя двигателя.

Для того, чтобы электродвигатель включался в соответствии с технологией, нужно предусмотреть сигнал готовности, который будет свидетельствовать о том, что по технологической цепочке привод можно запускать в работу. Аналогично нужно реализовать блокировку отключения, чтобы остановка агрегатов происходила в обратном порядке.

Кроме того, можно предусмотреть сигналы включения сигнальных ламп, которые зачастую устанавливаются на местных постах управления и совмещаются с кнопками запуска и останова. Чаще всего, лампа запуска работает в двух режимах: «Мигание» - привод готов к запуску, «Свечение» - привод запущен. Лампа останова также работает в двух режимах: «Свечение» - привод останавливается (ждет сигнала останова в соответствии с технологией), «Мигание» - авария технологического агрегата. Если ни одна лампа не горит, значит привод остановлен и запуск его запрещен. Можно также предусмотреть шифровку аварий посредством изменения частоты мигания лампы останова.

Дополнительно можно предусмотреть выход для формирования звукового сигнала при запуске и аварии привода.

Исходя из вышесказанного, объявление функционального блока **PRIVOD** будет выглядеть следующим образом:

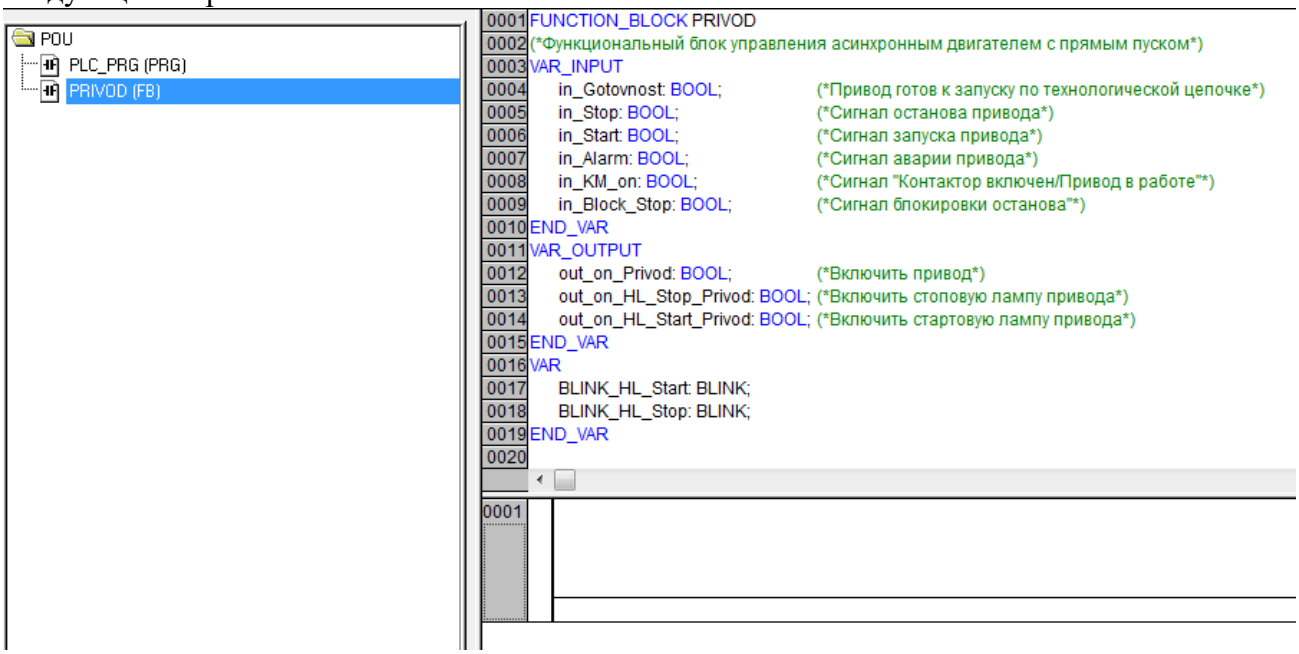


Рис.9. Список входных и выходных переменных функционального блока PRIVOD

!!! По мере написания функционального блока в раздел VAR будут добавлены локальные переменные для выполнения промежуточных операций !!!

Итак, приступим к написанию кода блока PRIVOD.

Для ***включения привода*** нам необходимо выполнение следующей логической цепочки:

- должен быть сигнал разрешения запуска по технологической цепочке;
- не должно быть сигнала аварии;
- не должно быть сигнала останова;
- есть сигнал запуска или самоподхвата.

При этом, при наличии сигнала останова и сигнала блокировки останова, в случае отсутствия аварийного сигнала, привод не должен отключаться.

Включение стартовой лампы осуществляется следующим образом: если нет аварийного и стопового сигналов, то в случае, когда привод запущен, лампа горит, а в случае, когда привод не запущен, но готов к запуску, лампа мигает.

Включение стоповой лампы осуществляется следующим образом: лампа горит, когда есть сигнал останова, и мигает, когда есть сигнал аварии.

Мигание осуществляется с помощью функционального блока **BLINK** из библиотеки ***Util.lib***. Для вставки функционального блока нужно щелкнуть правой кнопкой мыши на цепи, в которую его нужно вставить, и в контекстном меню выбрать ***Функциональный блок***, или нажать сочетание клавиш ***Ctrl+B***, или нажать кнопку ***Элемент*** на панели инструментов. При этом появится окно выбора функциональных блоков, где нужно выбрать соответствующую библиотеку и в ней найти требуемый функциональный блок (**BLINK** находится в папке *signals*).

В результате, код функционального блока **PRIVOD** будет выглядеть следующим образом:

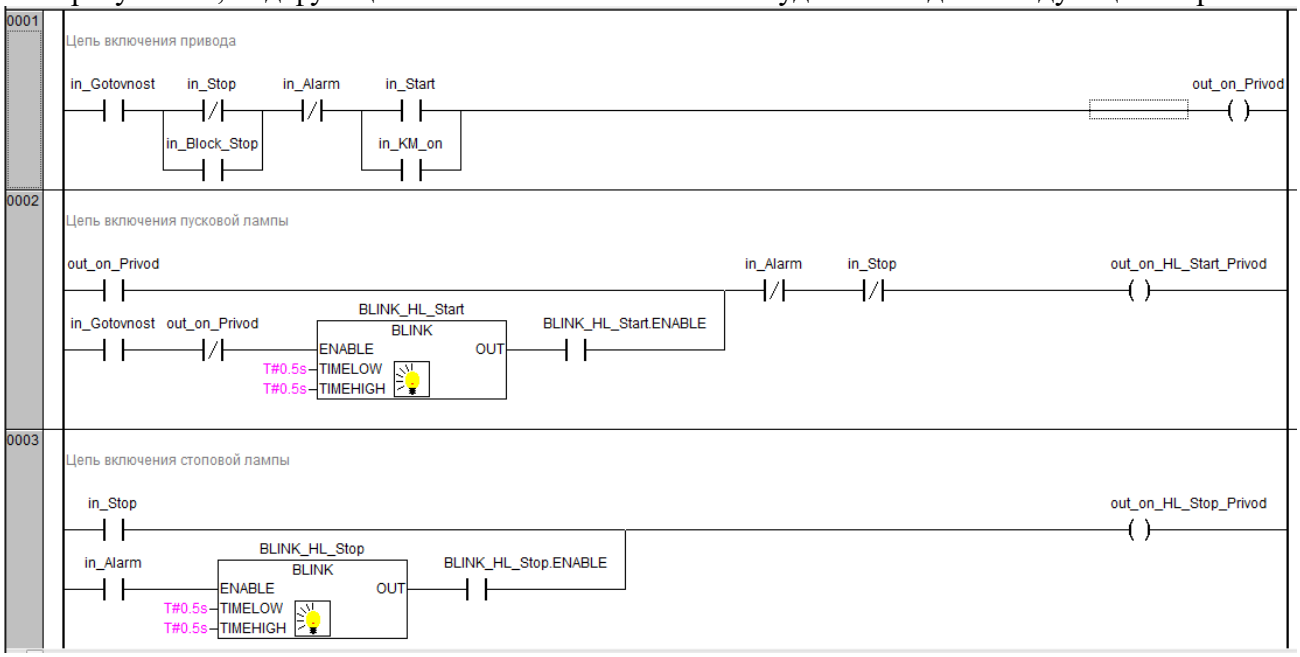


Рис.10. Код функционального блока PRIVOD

3. Вызов функционального блока PRIVOD и определение конфигурации контроллера

Для начала поставим перед собой простую задачу. Предположим, что у нас есть шлюзовой питатель, который подает сыпучий материал из бункера на ленточный конвейер, который в свою очередь транспортирует этот материал в конечную точку назначения. Нам нужно управлять запуском приводов этих агрегатов, осуществляя элементарную технологическую цепочку. В дальнейшем будем расширять рамки системы управления, но на начальном этапе реализуем следующие функции:

- контроль последовательности включения и отключения приводов;
- отработку аварийных сигналов;
- запуск и останов приводов по входным сигналам «Пуск» и «Стоп».

Начнем с ленточного конвейера. Мы уже знаем, что у нас должно быть три сигнала от пусковой аппаратуры:

- «Автоматический выключатель включен» (вход);
- «Контактор включен» (вход);
- «Включить контактор» (выход).

Кроме того, у нас должно быть четыре сигнала для светосигнальной аппаратуры: кнопка «Пуск» (вход) и лампа, совмещенная с кнопкой «Пуск» (выход); кнопка «Стоп» (вход) и лампа, совмещенная с кнопкой «Стоп» (выход).

Аналогично, такие сигналы будут использованы и для других агрегатов с асинхронными трехфазными двигателями, в том числе и для шлюзового питателя.

Также, для ленточного конвейера используются датчики противоаварийной защиты: трос безопасности (зачастую совмещенный с кнопкой аварийного останова), датчик схода ленты, а также датчик движения ленты.

Трос безопасности (и/или кнопка аварийного останова) предназначен для останова конвейера технологическим персоналом в случае возникновения аварийной ситуации (рис.11). При натяжении троса размыкается контакт концевого выключателя и разрывается цепь аварийного останова. При этом может разрываться питание оперативных цепей управления, что приведет к мгновенному пропаданию напряжения на катушке контакторов и останову технологической линии. Возможно формирование аварийного сигнала в контроллере; при этом останов должен быть безусловным, т.е. однозначно и мгновенно останавливать технологическую линию.

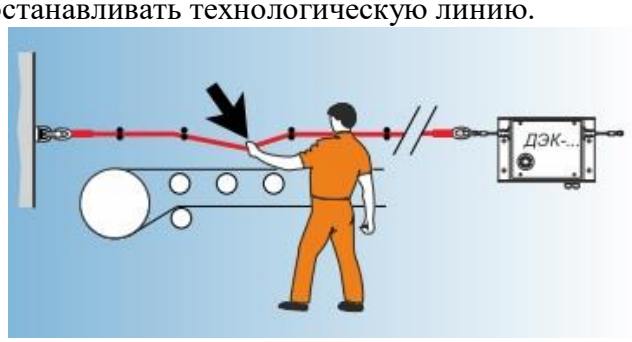


Рис.11. Принцип действия троса безопасности

Датчик схода ленты предназначен для предотвращения схода ленты с опорных роликов. Сход ленты – серьезная аварийная ситуация. Даже если при этом не произойдет повреждения ленты, восстановление работы конвейера после схода – сложная и трудоемкая задача. Поэтому практически все ленточные конвейеры оснащаются датчиками схода ленты, иногда (когда конвейер длинный, например, более 30 метров) несколькими парами. Принцип действия датчиков схода ленты можно уяснить из рис.12.

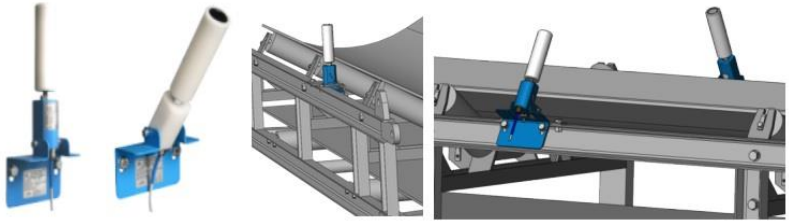


Рис.12 Принцип действия датчиков схода ленты

Датчик движения вала устанавливается на ведомый барабан ленточного конвейера. Он представляет собой индукционный датчик наличия материала, мимо которого проходит металлическая метка, наваренная на барабан. Таким образом, каждый оборот метка проходит мимо датчика, в результате чего возникает импульс. Наличие импульсов с заданной частотой говорит о нормальной работе конвейера. Если же конвейер включен, а частота импульсов равна нулю, значит, имеет место пробуксовка ленты, что нередко бывает в зимний период времени. В результате лента может быть повреждена, что повлечет немалые материальные и трудовые затраты. Поэтому необходимо контролировать наличие изменяющегося сигнала с датчика движения. Принцип действия датчика движения приведен на рис.13.

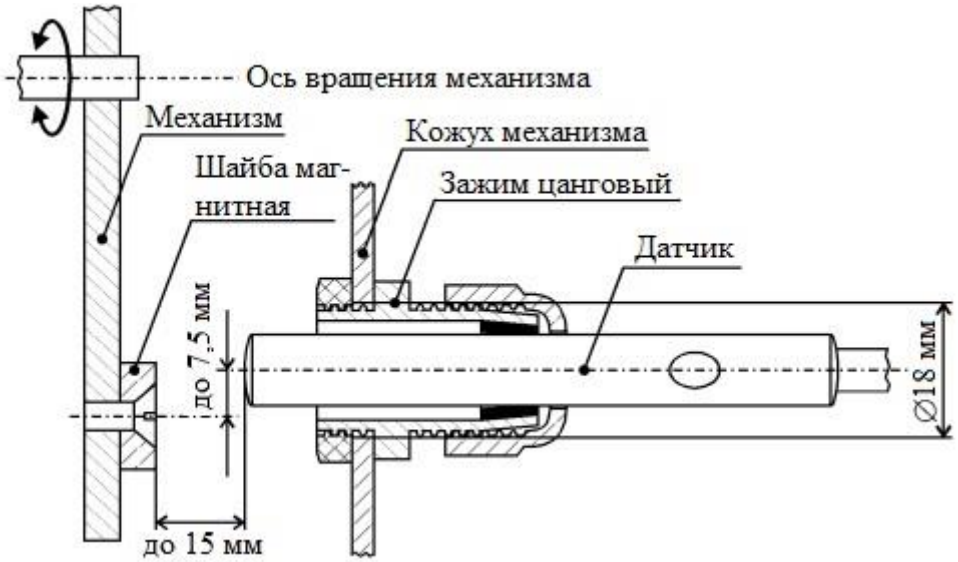


Рис.13. Принцип работы датчика движения

Таким образом, для ленточного конвейера нужно предусмотреть еще как минимум три сигнала с аварийных датчиков.

Итак, в общей сложности для ленточного конвейера нужно выделить 7 входов и 3 выхода.

Практика показывает, что пространство входов и выходов ПЛК целесообразно организовывать, основываясь на логических критериях их использования. К примеру, можно условиться, что первые 8 входов и 8 выходов будут отведены под общие нужды системы управления (например, сигналов «Наладочный режим», «Сброс аварии», «Звуковая аварийно-предупредительная сигнализация» и т.д.). Далее группами по 8 входов и 4 выхода (по 1 резервному входу и выходу на агрегат) будут идти сигналы технологических агрегатов.

При этом переменные, относящиеся к определенному агрегату должны содержать его условное обозначение (например, постфикс или префикс ***LK***). Это существенно повышает наглядность кода.

Теперь, основываясь на вышеприведенных соображениях, создадим в конфигурации ПЛК переменные для управления ленточным конвейером. Вид конфигурации приведен на рис.14.

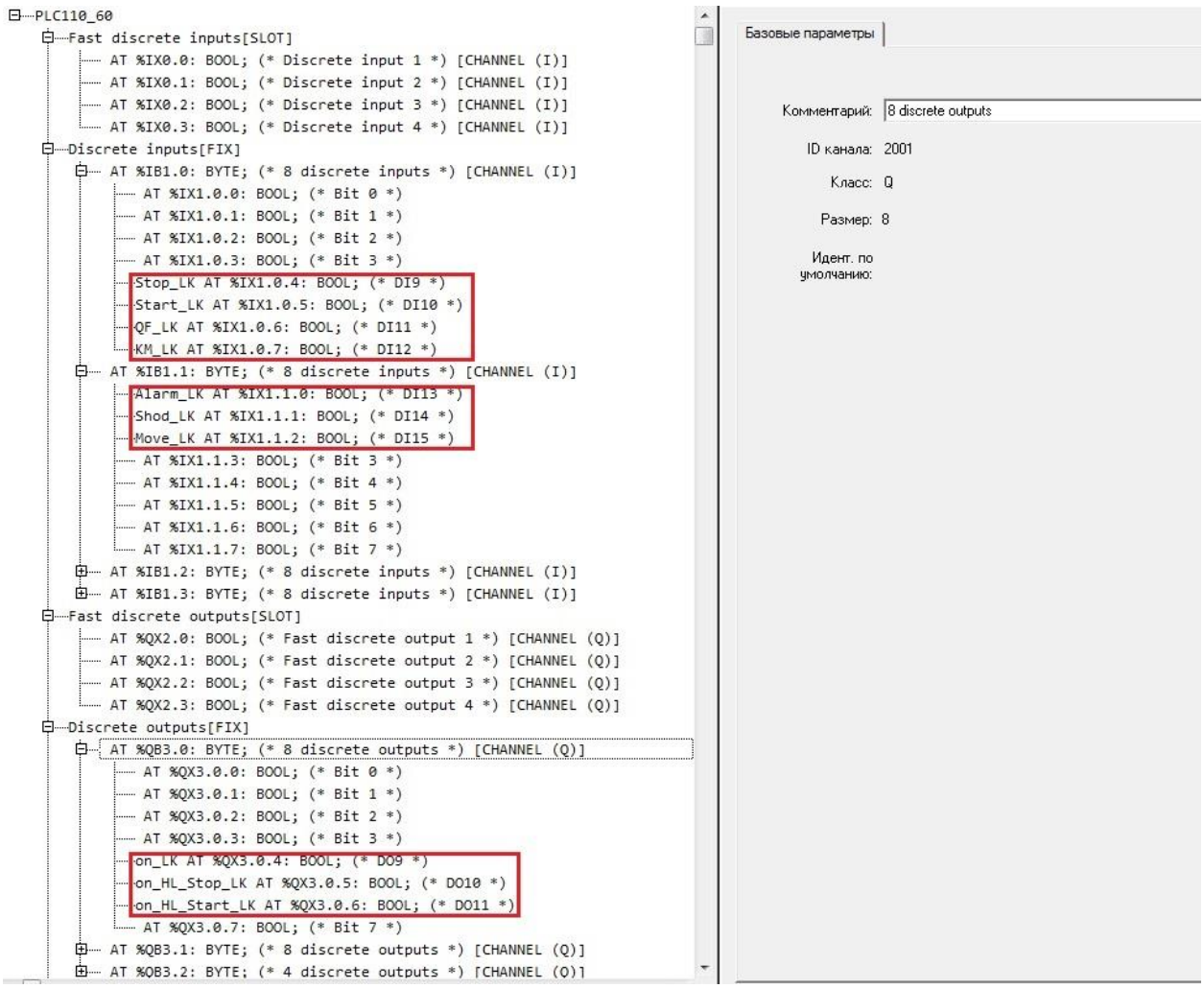


Рис.14. Конфигурация ПЛК с сигналами для ленточного конвейера

Теперь, когда создан блок PRIVOD и объявлены переменные для работы с ленточным конвейером, мы можем создать экземпляр блока PRIVOD для ленточного конвейера. Вызов блока управления ленточным конвейером (***LK***) приведен на рис.15.

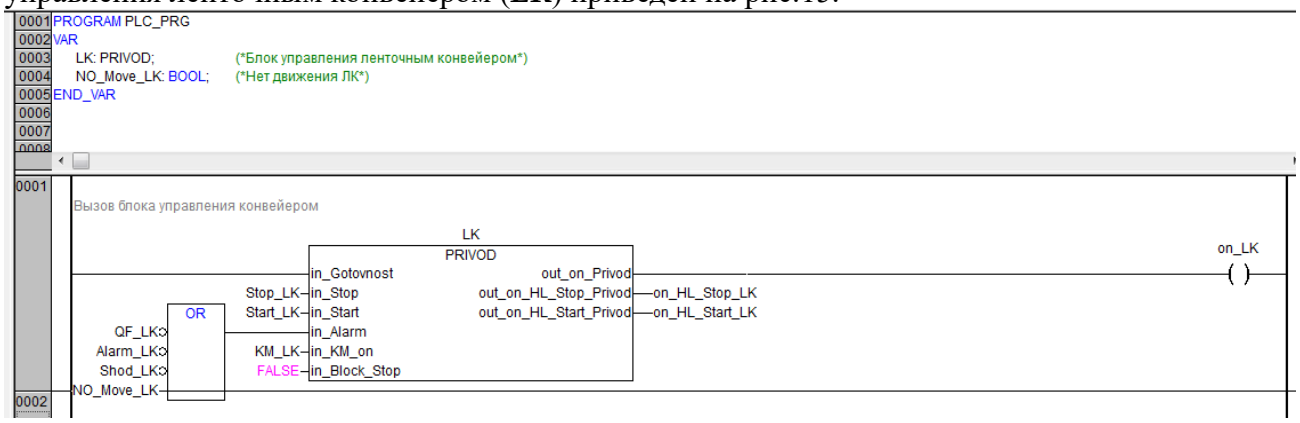


Рис.15. Вызов блока управления ленточным конвейером (LK)

При компиляции проекта ошибок быть не должно. В случае их возникновения внимательно проверьте правильность написания имен всех переменных. После подключения (***Онлайн – Подключение***) в режиме эмуляции (должна быть установлена метка ***Режим эмуляции*** в меню ***Онлайн***) и выставления в конфигурации ПЛК значений переменных ***QF_LK***, ***Alarm_LK*** и ***Shod_LK*** в значение **TRUE,** вы сможете включать и отключать ленточный конвейер (DO9 в конфигурации ПЛК) с помощью кнопок (сигналов) ***Stop_LK*** и ***Start_LK***.

По аналогии с блоком ленточного конвейера создадим экземпляр блока шлюзового питателя, назовем его ***PIT***, и вызовем его в ***PLC_PRG***. Для шлюзового питателя будет использоваться кнопка аварийного останова (вместо троса безопасности), сход ленты использоваться не будет (вход необходимо оставить под резерв), датчик движения функционирует аналогичным образом.

Соответственно, нужно объявить переменные для питателя (с постфиксом Pit) в конфигурации ПЛК с резервным входом (точней двумя, с учетом отсутствия сигнала схода ленты) и резервным выходом (рис. 17). А затем необходимо сделать вызов блока с учетом того, что шлюзовой питатель может быть запущен только спустя 3 секунды после запуска ленточного конвейера (рис. 18).

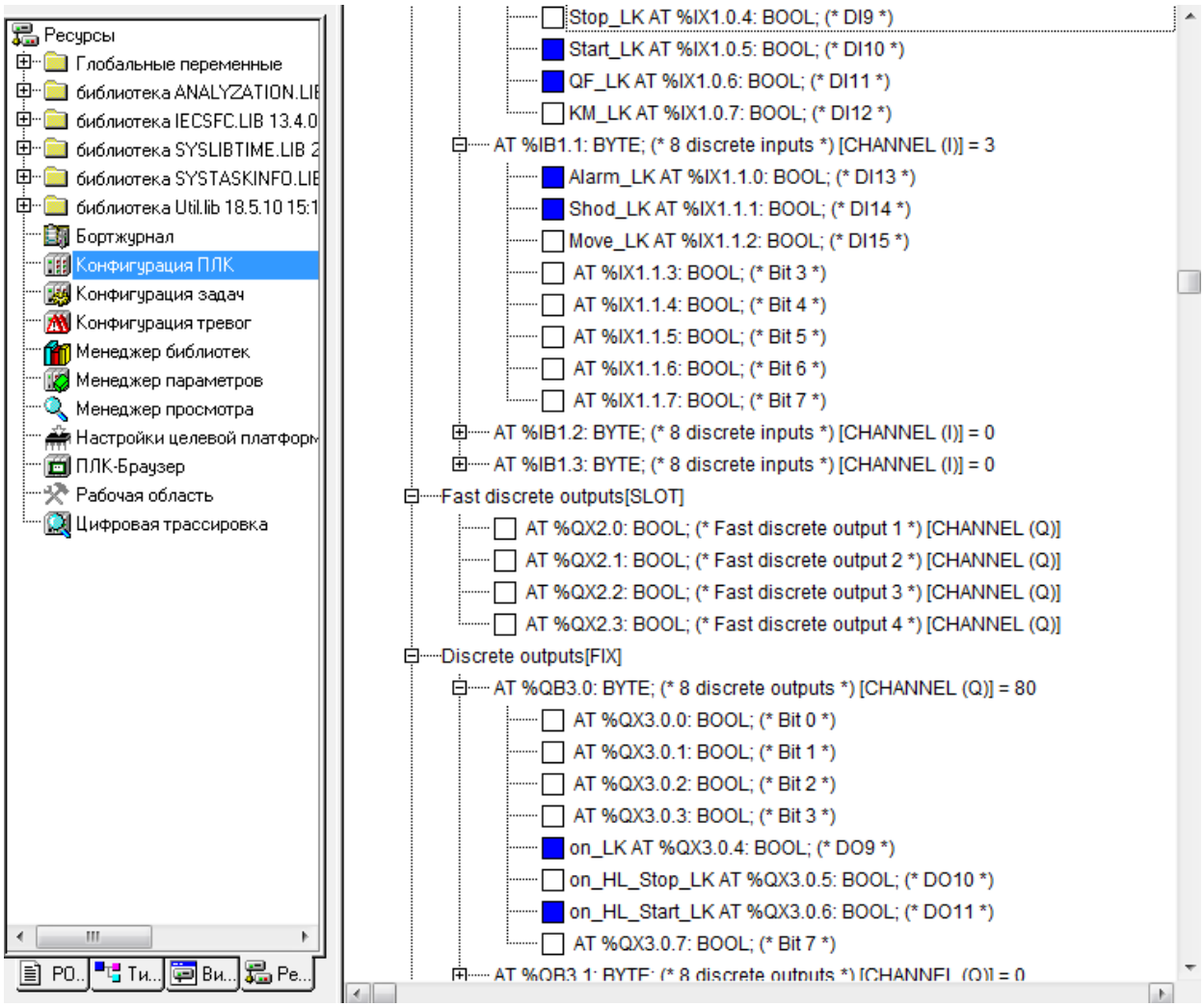


Рис. 16. Управление ленточным конвейером через ресурс «Конфигурация ПЛК»

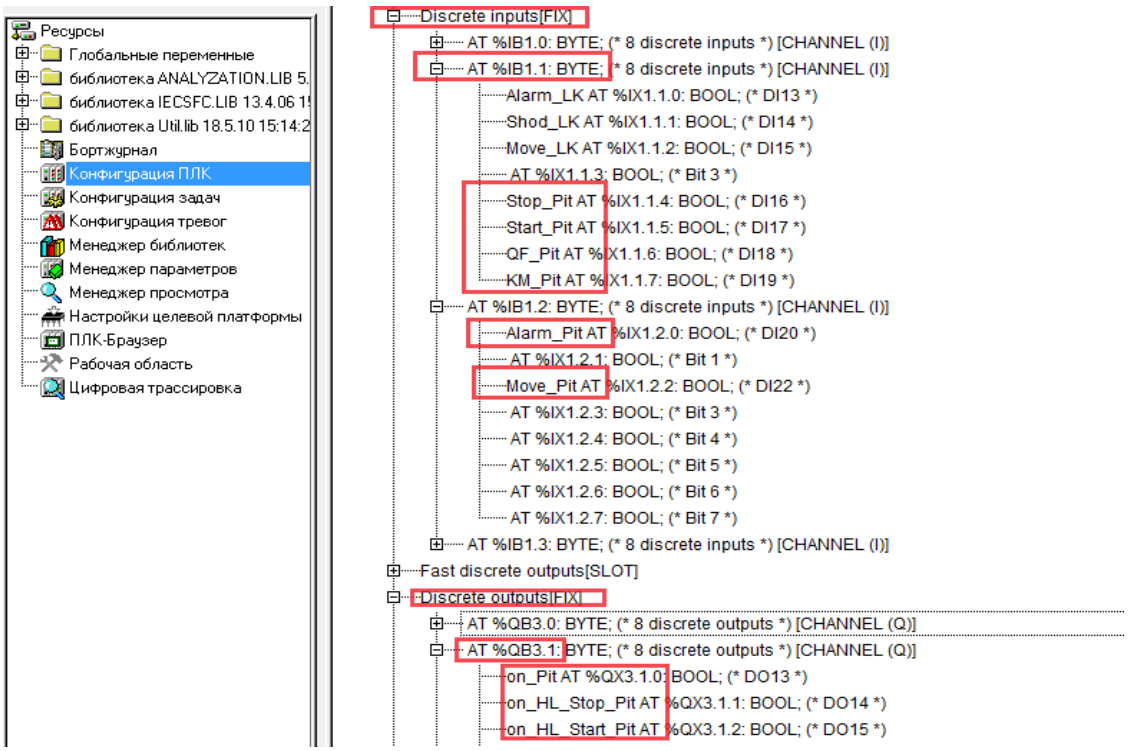


Рис.17. Конфигурация ПЛК с сигналами для шлюзового питателя

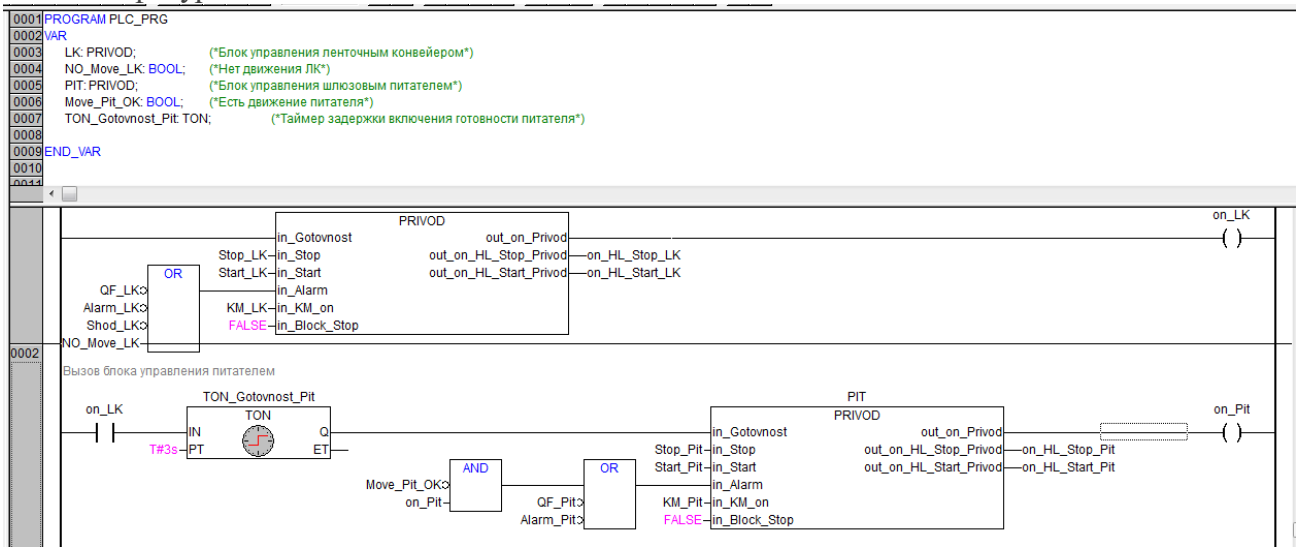


Рис.18. Вызов блока управления шлюзовым питателем (Pit)

Так как входные сигналы ПЛК мы не можем изменять из программы, цепь самоподхвата будет работать некорректно, пока мы не подключим реальный ПЛК и реальный контактор (или хотя бы реле). Поэтому в программе ***PLC_PRG*** нужно сделать имитацию срабатывания контактора (вставить после каждого из блоков). Логика проста: если есть сигнал на включение соответствующего агрегата, то наш «виртуальный контактор» срабатывает. Эти сигналы нужно подать через блок **OR** вместе с физическим сигналом на соответствующий вход блоков ***LK*** и ***PIT***. В результате цепь самоподхвата станет работать аналогично реальной ситуации (рис. 19).

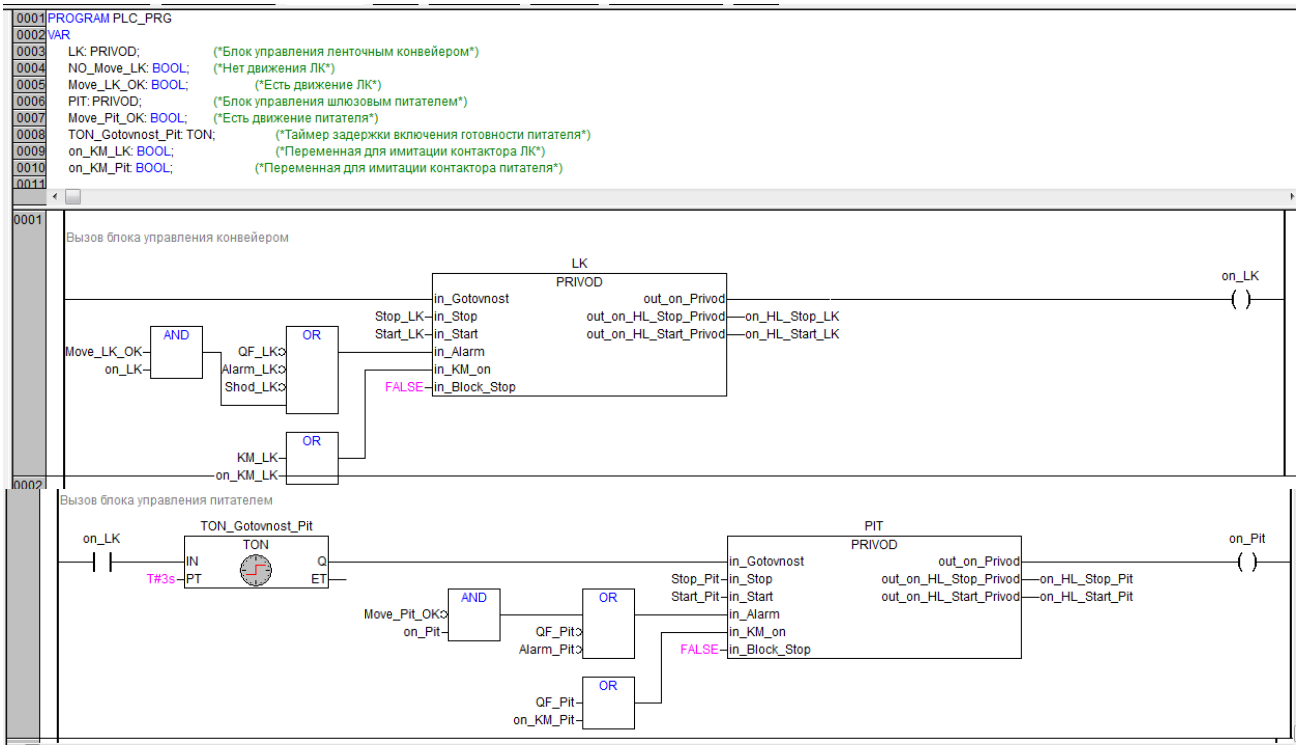


Рис. 19. Имитация срабатывания контакторов ***LK*** и ***PIT***

Кроме того, необходимо реализовать имитацию включения катушки и дополнительного контакта ленточного конвейера и шлюзового питателя (рис. 20).

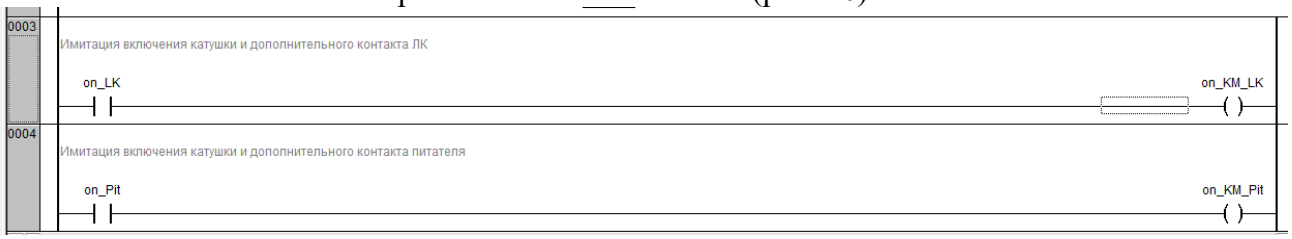


Рис. 20. Имитация включения катушки и дополнительного контакта ***LK*** и ***PIT***

Примечание: если мы просто включим сигнал ***in_KM*** на входе ПЛК для соответствующего привода, то он будет запускаться без кнопки старт.

Кроме того, не будет и адекватной отработки сигнала с датчика движения, потому что он должен появляться и пропадать с определенной частотой. Таким образом, необходимо реализовать имитацию датчика движения и отработку аварии по отсутствию движения. Имитацию можно осуществить простым блоком **BLINK**, на вход которого будет приходить сигнал датчика с входа ПЛК. При этом время импульса и время паузы в реальной ситуации соответствуют времени появления сигнала и времени отсутствия, и в сумме должны составлять приблизительно один период вращения вала агрегата, на котором установлен датчик. Отработка этого сигнала также довольно проста: если при включенном приводе в течении 2-3 периодов вращения вала нет изменения сигнала датчика, то необходимо сформировать сигнал аварии и отключить привод (рис.

21, 22).

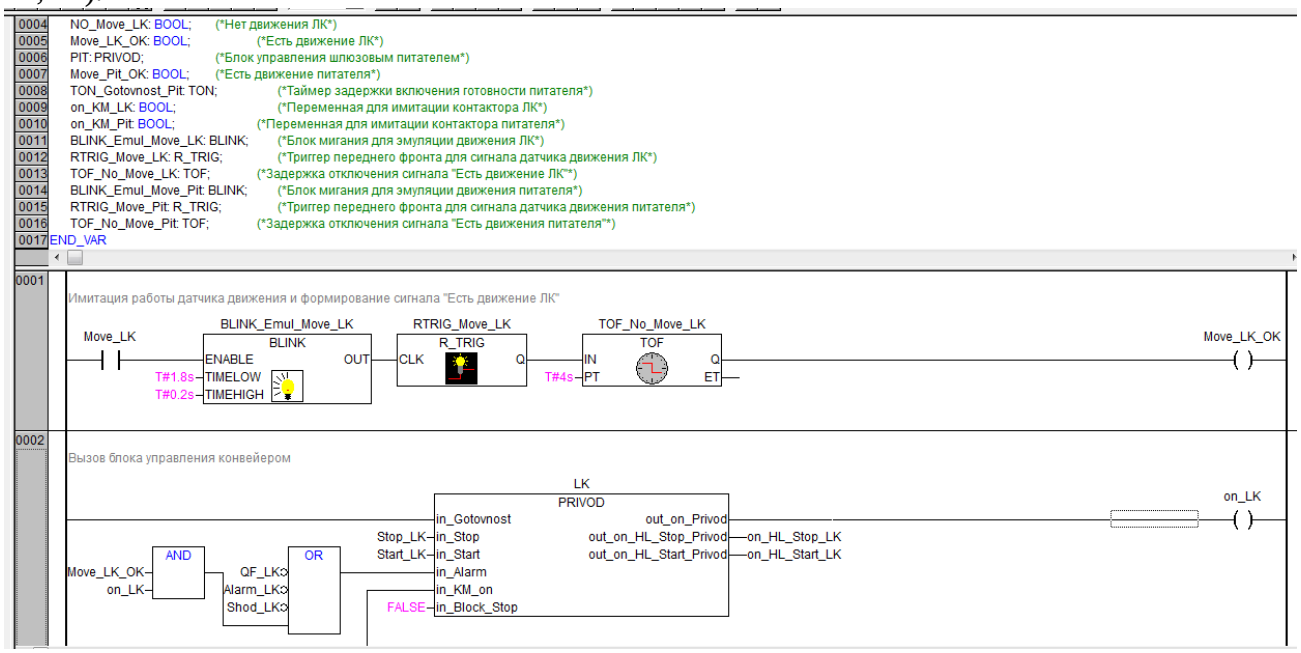


Рис. 21. Имитация датчика движения и формирование сигнала «Есть движение ЛК»

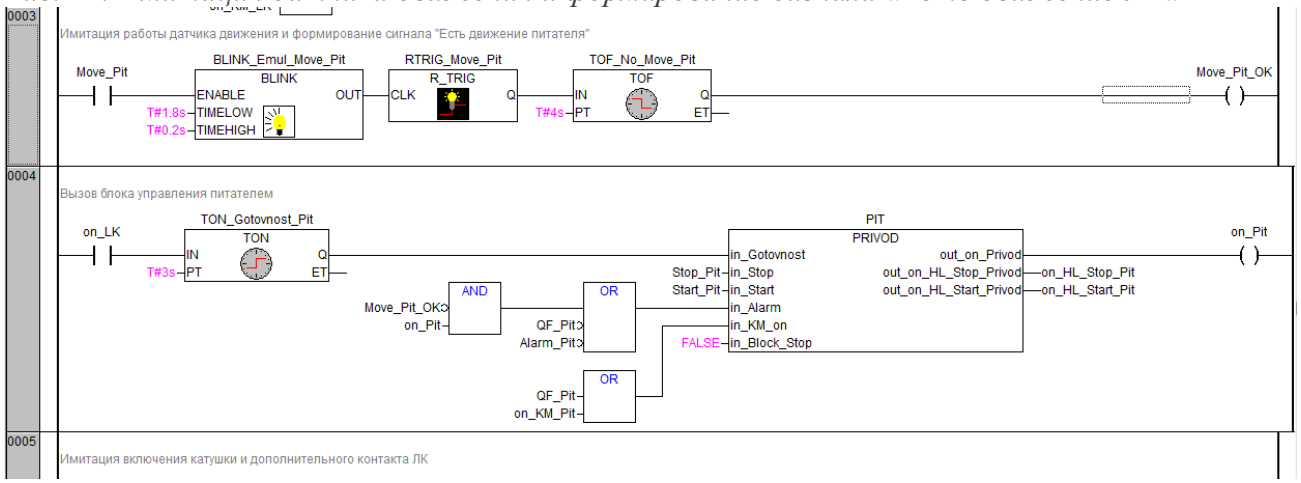


Рис. 22. Имитация датчика движения и формирование сигнала
«Есть движение шлюзового питателя»

4. Реализация фиксации аварии и квитирования аварийной ситуации

Фиксация аварии подразумевает, что при возникновении аварийной ситуации сигнал аварии фиксируется и не сбрасывается автоматически после устранения причин аварии, а сбрасывается только по специальному сигналу. Это может быть кнопка на посте управления оператора или сигнал из SCADA-системы.

Предположим, что для сброса аварии используется кнопка, нормально открытый контакт которой подает сигнал на 8 вход контроллера (назовите переменную ***Reset_Alarm***, рис. 22). Таким образом, нужно зафиксировать сигнал аварии и сбрасывать его только после устранения всех неполадок ***и нажатия кнопки «Сброс аварии»***. Отсутствие движения при этом в учет не брать – данный сигнал может появиться только после запуска агрегата.

Кроме того, нужно известить персонал о наличии аварии включением звуковой сигнализации, которая будет включаться выходом 8 (назовите переменную ***on_Sound_Signal***, рис. 22).

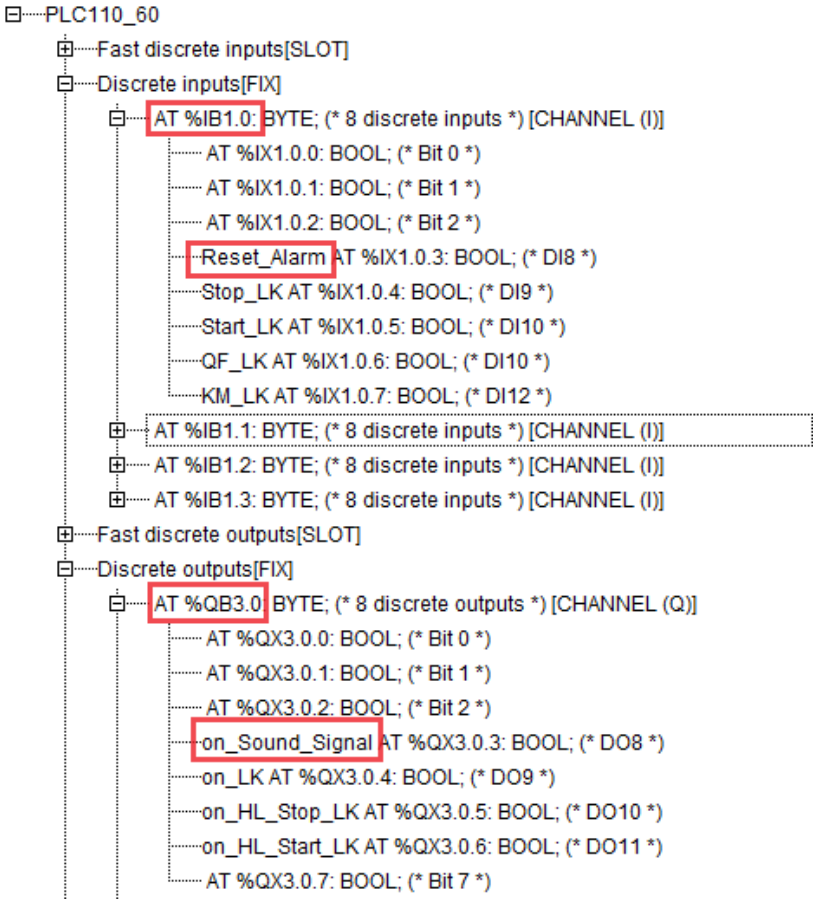


*Рис. 22. Введение переменных **Reset_Alarm** и **on_Sound_Signal** в конфигурацию контроллера*

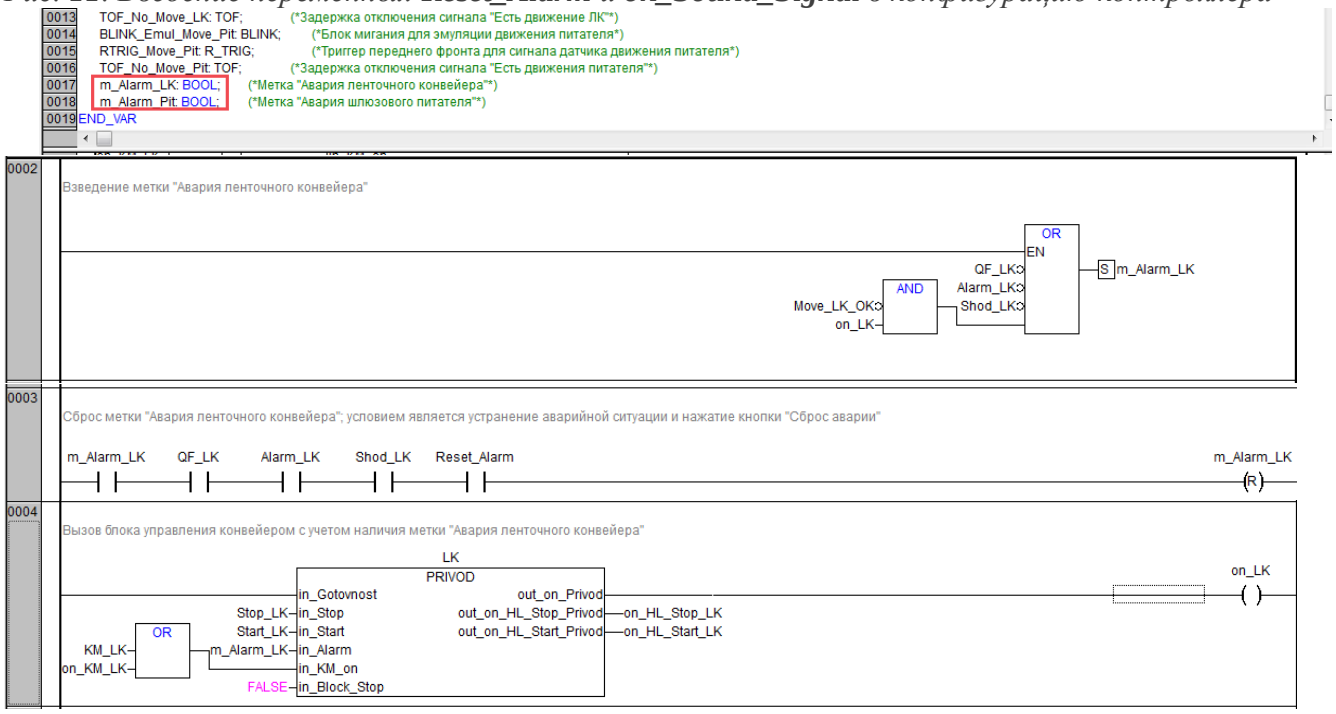


Рис. 23. Фиксация аварии ленточного конвейера

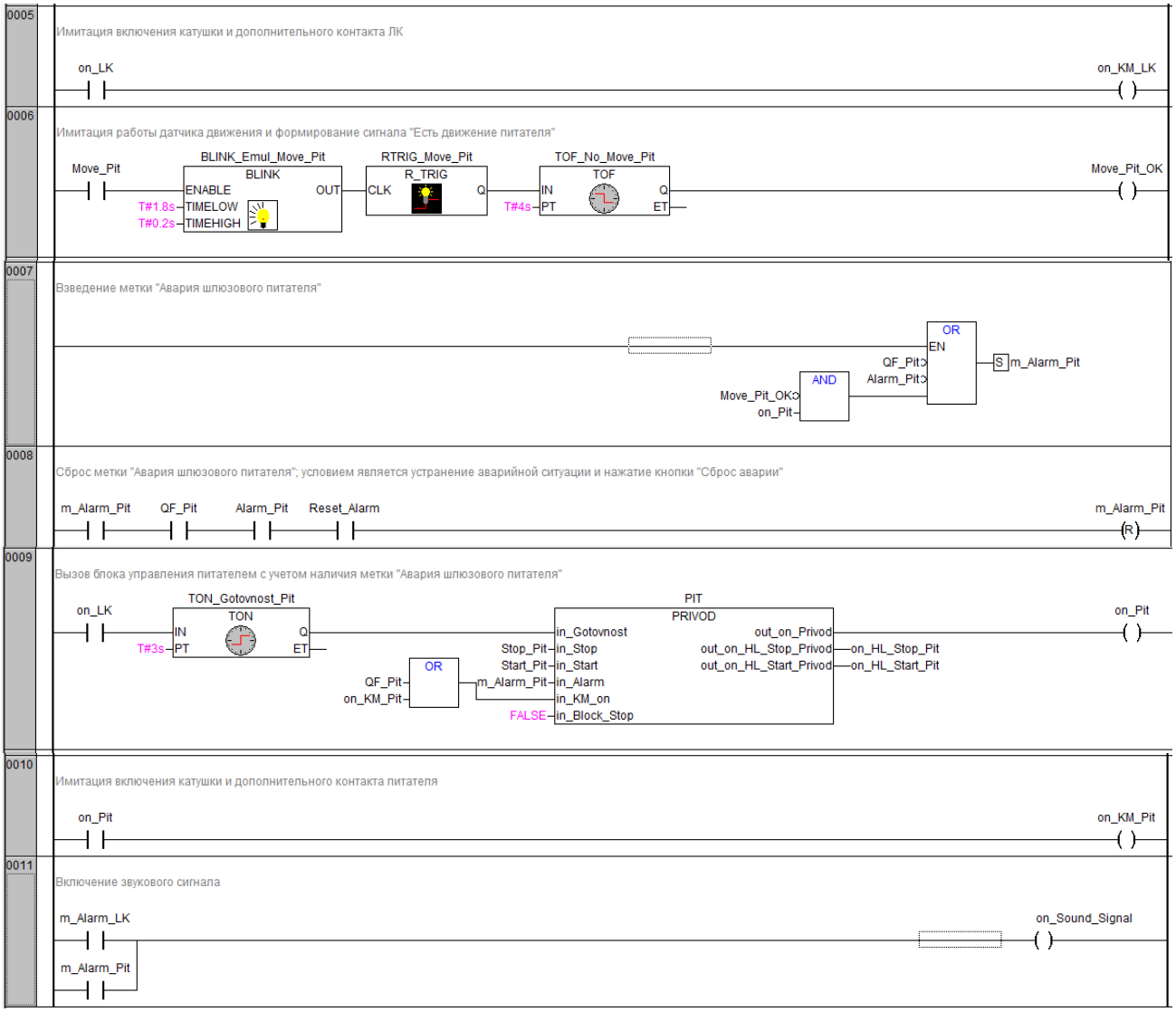


Рис. 24. Фиксация аварии шлюзового питателя и реализация включения звукового сигнала

5. Реализация предпусковой сигнализации

Предпусковая сигнализация является необходимым условием запуска технологической линии. Первый агрегат можно запустить только после проведения предпусковой сигнализации. Таким образом, на вход готовности первого привода сигнал будет идти не напрямую, а с соответствующим логическим условием.

Предположим, что для включения предпусковой сигнализации используется кнопка, нормально открытый контакт которой подает сигнал на 7 вход контроллера (назовите переменную ***Start_Warn***). Таким образом, ***при нажатии кнопки «Предпусковая сигнализация»*** нужно включить предпусковую сигнализацию на определенное время (примем его равным 5 секунд) и по окончании предпусковой сигнализации дать готовность к запуску первому приводу (в нашем случае, ленточному конвейеру). Пусть с кнопкой предпусковой сигнализации совмещена лампа предпусковой сигнализации, которая включается выходом 7 ПЛК (назовите переменную ***on_HL_Start_Warn***). Эта лампа должна мигать, если предпусковая сигнализация не было проведена, и гореть – если предпусковая сигнализация уже прошла.

Кроме того, нужно известить персонал о начале запуска линии включением звуковой сигнализации, которая будет включаться выходом 8 (переменная ***on_Sound_Signal*** была введена ранее).

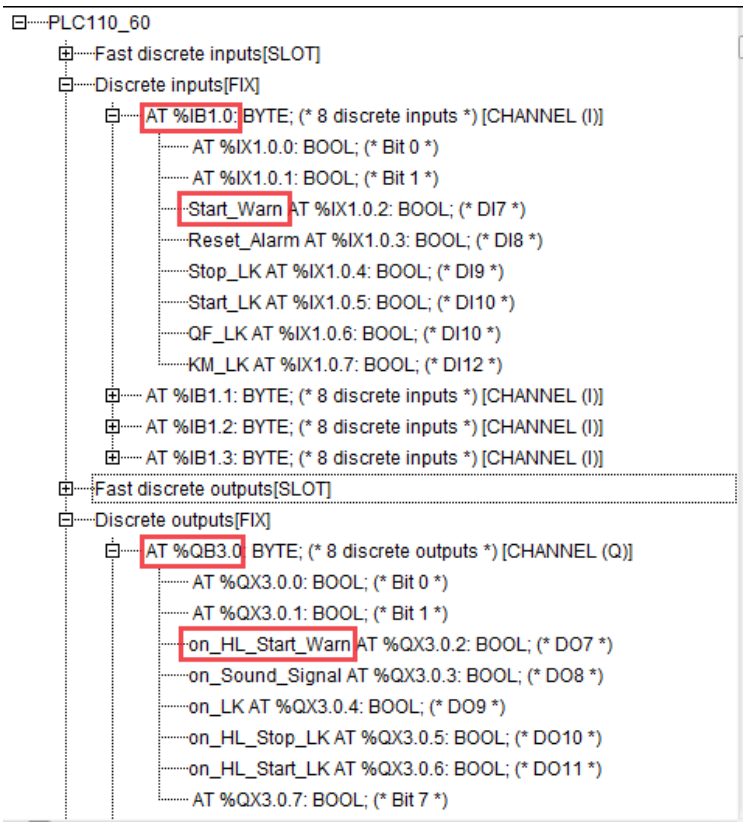
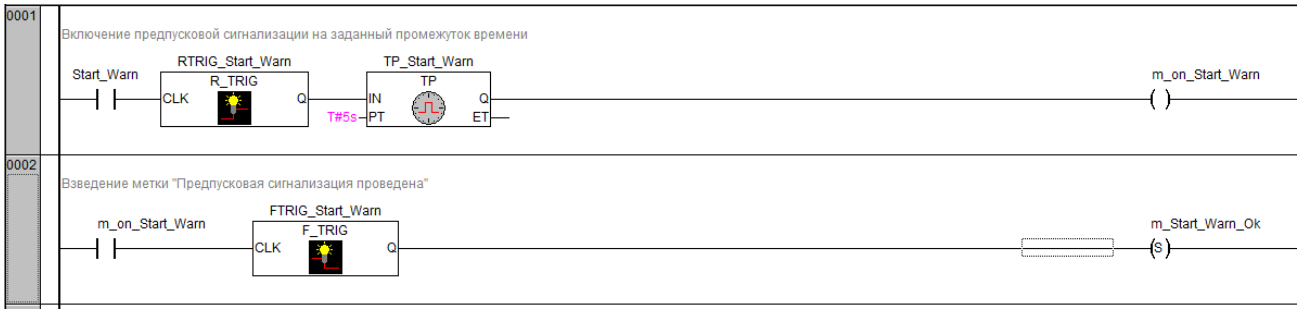


Рис. 25. Введение переменных ***Start_Warn*** *и* ***on_HL_Start_Warn*** *в конфигурацию контроллера*

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     LK: PRIVOD;                  (*Блок управления ленточным конвейером*)
0004     NO_Move_LK: BOOL;            (*Нет движения ЛК*)
0005     Move_LK_OK: BOOL;                (*Есть движение ЛК*)
0006     PIT: PRIVOD;                 (*Блок управления шлюзовым питателем*)
0007     Move_Pit_OK: BOOL;           (*Есть движение питателя*)
0008     TON_Gotovnost_Pit: TON;              (*Таймер задержки включения готовности питателя*)
0009     on_KM_LK: BOOL;                      (*Переменная для имитации контактора ЛК*)
0010     on_KM_Pit: BOOL;                 (*Переменная для имитации контактора питателя*)
0011     BLINK_Emul_Move_LK: BLINK;       (*Блок мигания для эмуляции движения ЛК*)
0012     RTRIG_Move_LK: R_TRIG;           (*Триггер переднего фронта для сигнала датчика движения ЛК*)
0013     TOF_No_Move_LK: TOF;         (*Задержка отключения сигнала "Есть движение ЛК"*)
0014     BLINK_Emul_Move_Pit: BLINK;      (*Блок мигания для эмуляции движения питателя*)
0015     RTRIG_Move_Pit: R_TRIG;          (*Триггер переднего фронта для сигнала датчика движения питателя*)
0016     TOF_No_Move_Pit: TOF;        (*Задержка отключения сигнала "Есть движения питателя"*)
0017     m_Alarm_LK: BOOL;        (*Метка "Авария ленточного конвейера"*)
0018     m_Alarm_Pit: BOOL;       (*Метка "Авария шлюзового питателя"*)
0019     RTRIG_Start_Warn: R_TRIG;    (*Триггер переднего фронта кнопки "Пусковая сигнализация"*)
0020     TP_Start_Warn: TP;           (*Импульс предпусковой сигнализации*)
0021     m_on_Start_Warn: BOOL;       (*Сигнал включения предпусковой сигнализации*)
0022     FTRIG_Start_Warn: F_TRIG;    (*Триггер заднего фронта включения предпусковой сигнализации*)
0023     m_Start_Warn_Ok: BOOL;       (*Метка "Предпусковая сигнализация проведена"*)
0024     FTRIG_on_LK: F_TRIG;         (*Триггер отключения ЛК*)
0025     BLINK_HL_Start_Warn: BLINK;  (*Мигание лампы предпусковой сигнализации*)
0026 END_VAR
```

Рис. 26. Введение дополнительных локальных переменных для реализации предпусковой сигнализации



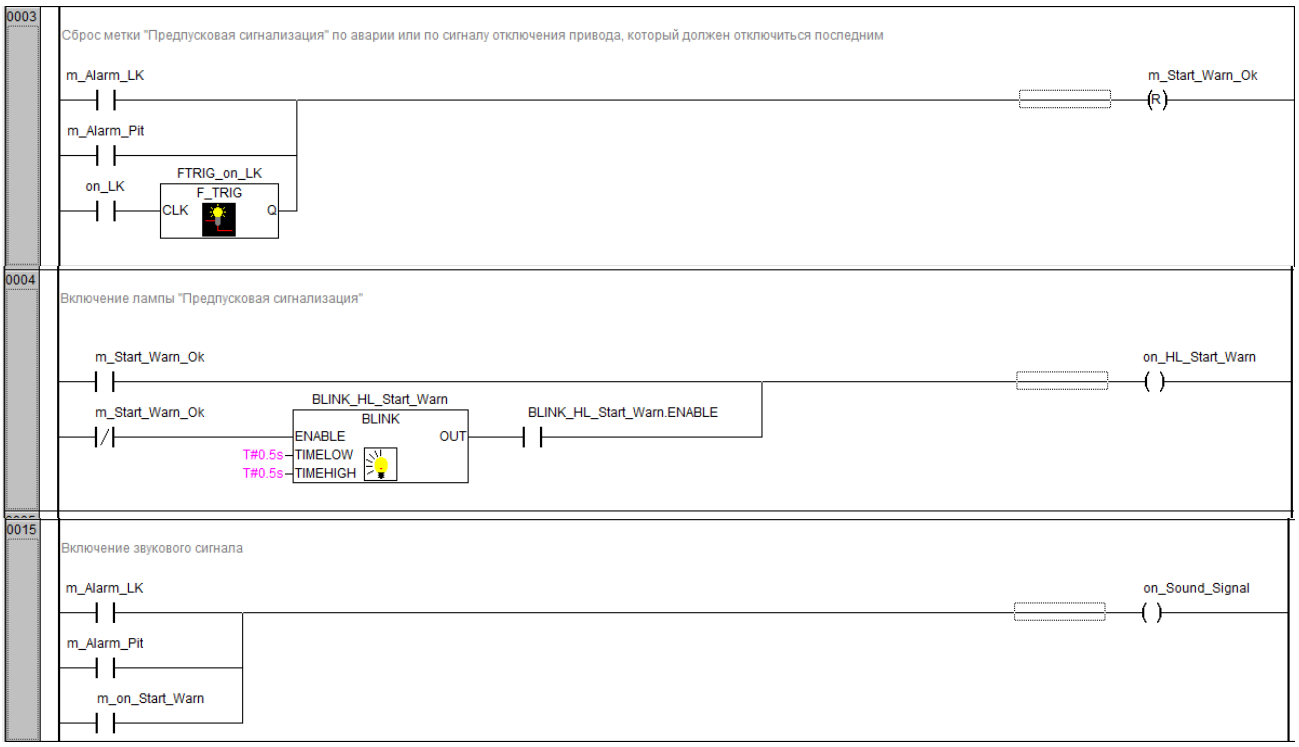


Рис. 27. Реализация предпусковой сигнализации
(цепи 0005-0015 аналогичны цепям 0002-0010 рис. 23, 24)

6. Реализация наладочного режима работы

Наладочный режим работы предусмотрен для отладки и ремонта оборудования с местных постов управления. При этом работы осуществляются без материала и последовательность запуска и останова игнорируется. Включается данный режим работы как правило ключ-биркой, которую ремонтный персонал уносит с собой – чтобы исключить возможность запуска оборудования с поста оператора. Подчеркнем еще раз: в наладочном режиме запуск возможен только с местных постов управления. Предположим, нормально открытый сигнал с ключ-бирки ***«Наладка»*** подает сигнал на 6 вход ПЛК (назовите переменную ***Naladka***).

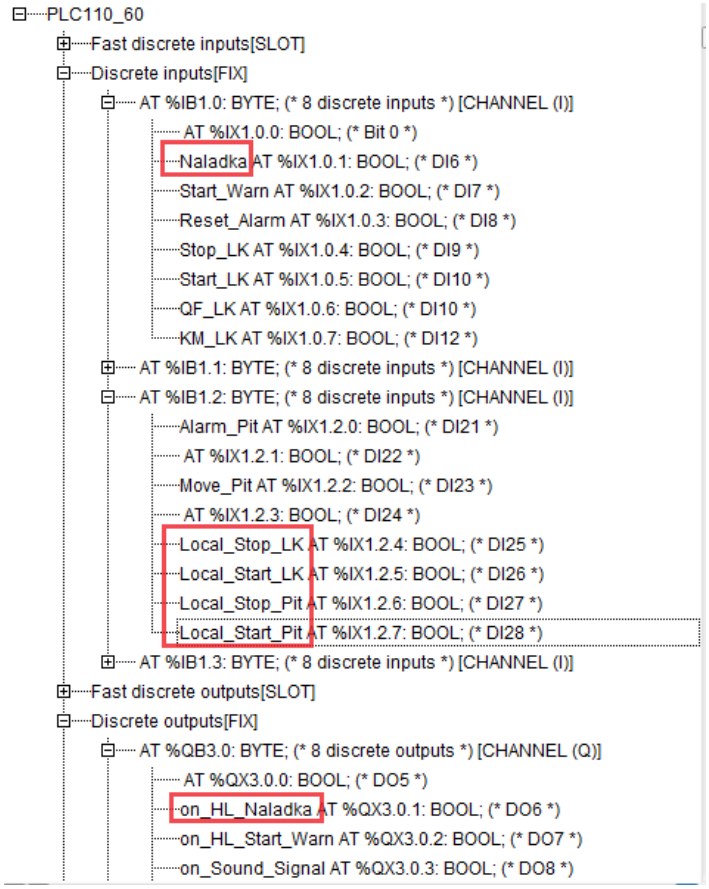


Рис. 28. Введение новых переменных в конфигурацию контроллера для реализации наладочного режима

Предположим, что местный пост подключен к входам 25-28 и состоит из четырех кнопок (назовите переменные аналогично используемым именам пусковых и стоповых кнопок с префиксом ***Local_***). Таким образом, ***при переходе в режим «Наладка»*** нужно заблокировать сигналы с поста оператора и передать управления на местный пост, а также снять ограничения на последовательности останова и запуска. Если при технической реализации местного поста управления используются кнопки с подсветкой, то сигналы для ламп зачастую берутся параллельно аналогичным сигналам ламп на посте оператора, поэтому отдельно программировать выходы не будем. Однако для оператора нужно предусмотреть лампу «Наладка», которая будет включаться выходом 6 ПЛК (назовите переменную ***on_HL_Naladka***).

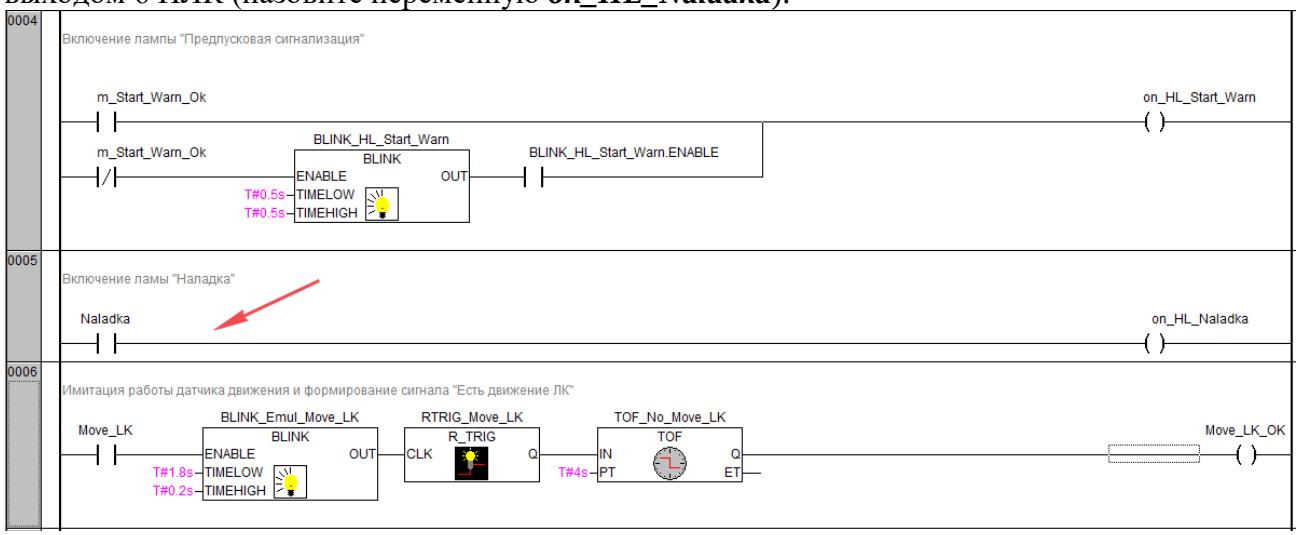


Рис. 29. Включение лампы «Наладка»

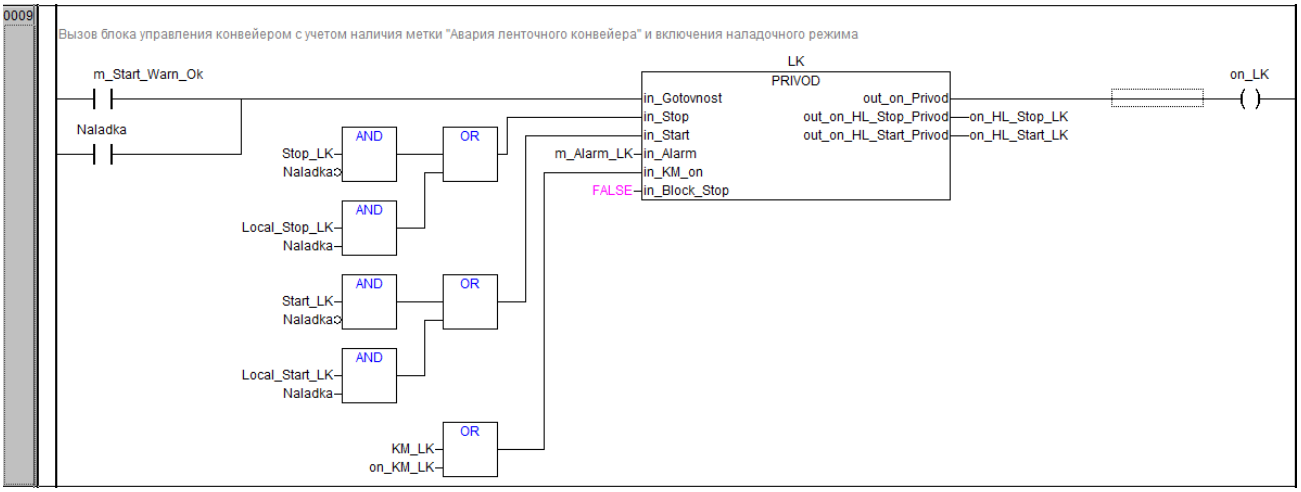


Рис. 30. Вызов блока управления конвейером с учетом наличия метки «Авария ленточного конвейера» и включения наладочного режима (цепи перед ним без изменений)

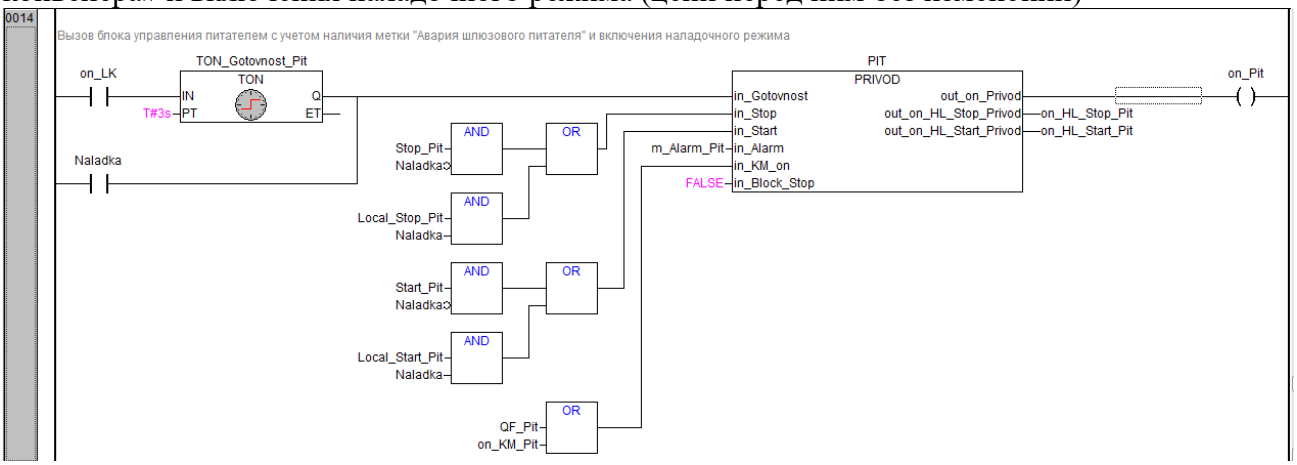


Рис. 31. Вызов блока управления питателем с учетом наличия метки «Авария ленточного конвейера» и включения наладочного режима (цепи перед ним и после без изменений)

7. Реализация автоматического режима работы

Автоматический режим работы предусмотрен для автоматического запуска всего оборудования в соответствии технологией. Включается данный режим работы как правило переключателем или ключ-биркой. Предположим, нормально открытый сигнал с переключателя ***«Авто»*** подает сигнал на 5 вход ПЛК (назовите переменную ***Auto_Start***).

Запуск и останов линии при этом будет осуществляться кнопками «Общий ПУСК» и «Общий «СТОП», которые будут подавать соответствующие сигналы на входы 1 и 2 ПЛК (назовите переменные ***Start_All*** и ***Stop_All***). Таким образом, ***при переходе в режим «Авто»*** нужно организовать запуск и останов приводов по технологической цепочке. При этом нужно учесть, что ленточный конвейер может быть остановлен только после останова ленточного питателя.

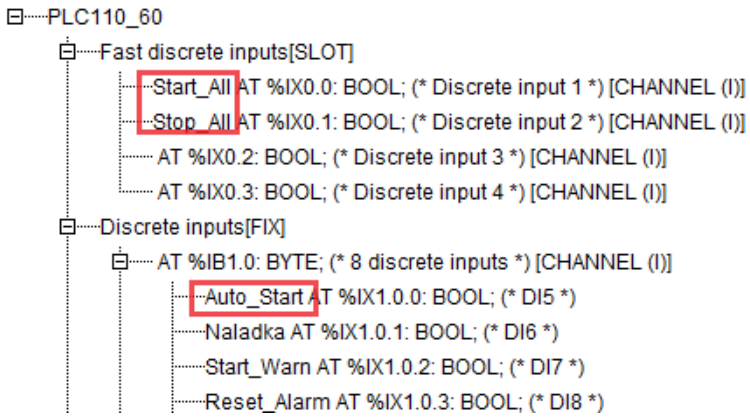


Рис. 32. Введение новых переменных в конфигурацию контроллера для реализации автоматического режима

```
0002 VAR
0003     LK: PRIVOD;                (*Блок управления ленточным конвейером*)
0004     NO_Move_LK: BOOL;          (*Нет движения ЛК*)
0005     Move_LK_OK: BOOL;              (*Есть движение ЛК*)
0006     PIT: PRIVOD;               (*Блок управления шлюзовым питателем*)
0007     Move_Pit_OK: BOOL;         (*Есть движение питателя*)
0008     TON_Gotovnost_Pit: TON;            (*Таймер задержки включения готовности питателя*)
0009     on_KM_LK: BOOL;                    (*Переменная для имитации контактора ЛК*)
0010     on_KM_Pit: BOOL;           (*Переменная для имитации контактора питателя*)
0011     BLINK_Emul_Move_LK: BLINK;         (*Блок мигания для эмуляции движения ЛК*)
0012     RTRIG_Move_LK: R_TRIG;             (*Триггер переднего фронта для сигнала датчика движения ЛК*)
0013     TOF_No_Move_LK: TOF;       (*Задержка отключения сигнала "Есть движение ЛК"*)
0014     BLINK_Emul_Move_Pit: BLINK;        (*Блок мигания для эмуляции движения питателя*)
0015     RTRIG_Move_Pit: R_TRIG;            (*Триггер переднего фронта для сигнала датчика движения питателя*)
0016     TOF_No_Move_Pit: TOF;      (*Задержка отключения сигнала "Есть движения питателя"*)
0017     m_Alarm_LK: BOOL;          (*Метка "Авария ленточного конвейера"*)
0018     m_Alarm_Pit: BOOL;         (*Метка "Авария шлюзового питателя"*)
0019     RTRIG_Start_Warn: R_TRIG;  (*Триггер переднего фронта кнопки "Пусковая сигнализация"*)
0020     TP_Start_Warn: TP;         (*Импульс предпусковой сигнализации*)
0021     m_on_Start_Warn: BOOL;     (*Сигнал включения предпусковой сигнализации*)
0022     FTRIG_Start_Warn: F_TRIG;  (*Триггер заднего фронта включения предпусковой сигнализации*)
0023     m_Start_Warn_Ok: BOOL;     (*Метка "Предпусковая сигнализация проведена"*)
0024     FTRIG_on_LK: F_TRIG;       (*Триггер отключения ЛК*)
0025     BLINK_HL_Start_Warn: BLINK;(*Мигание лампы предпусковой сигнализации*)
0026     m_Alarm: BOOL;             (*Метка аварии, включается при наличии любой аварии в системе*)
0027     m_Stop_All: BOOL;          (*Метка общего останова*)
0028     m_Start_All: BOOL;         (*Метка общего пуска*)
0029     FTRIG_Stop_All: F_TRIG;    (*Триггер отключения ленточного конвейера*)
0030     TOF_Reset_Stop_All: TOF;   (*Задержка отключения сброса метки общего останова*)
0031     TOF_Block_Stop_LK: TOF;    (*Задержка отключения ЛК*)
0032 END_VAR
```

Рис. 33. Введение дополнительных локальных переменных для реализации автоматического режима

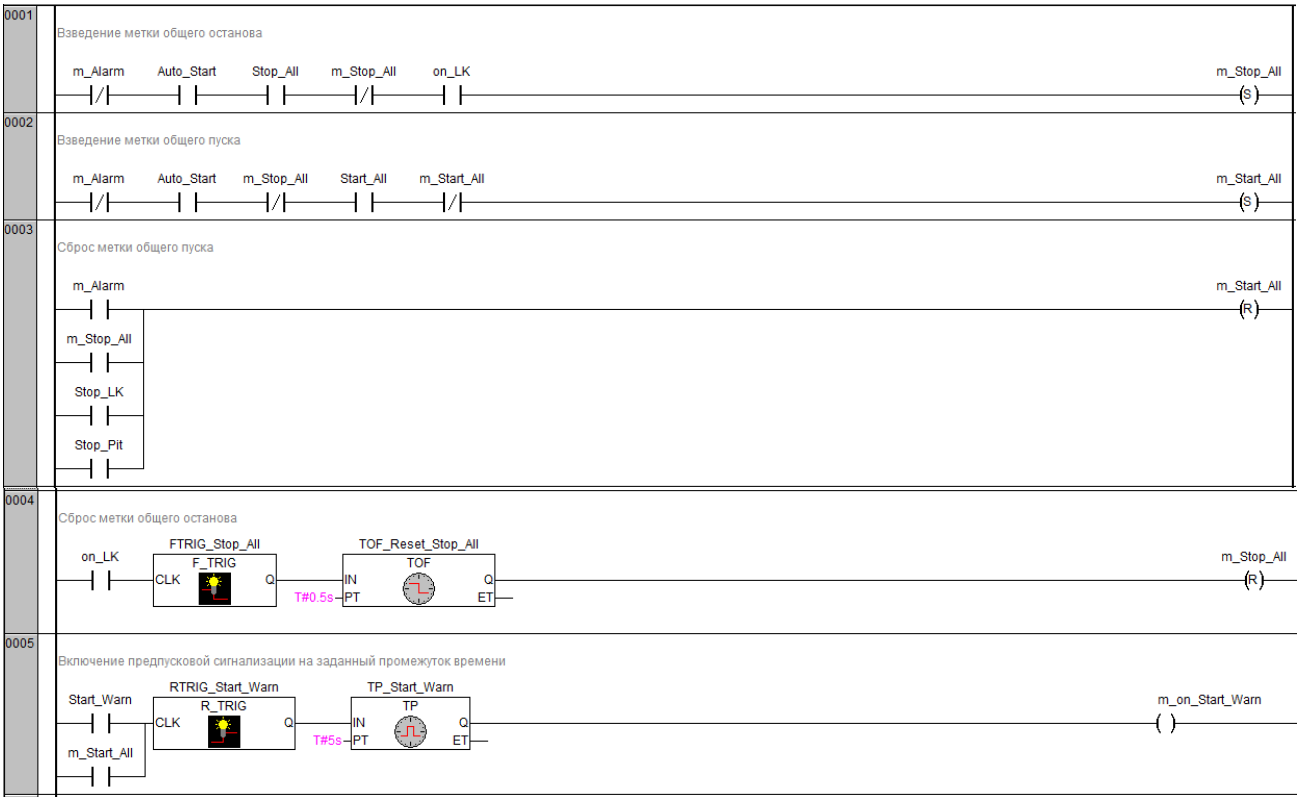


Рис. 34. Введение и сброс меток общего пуска и останова

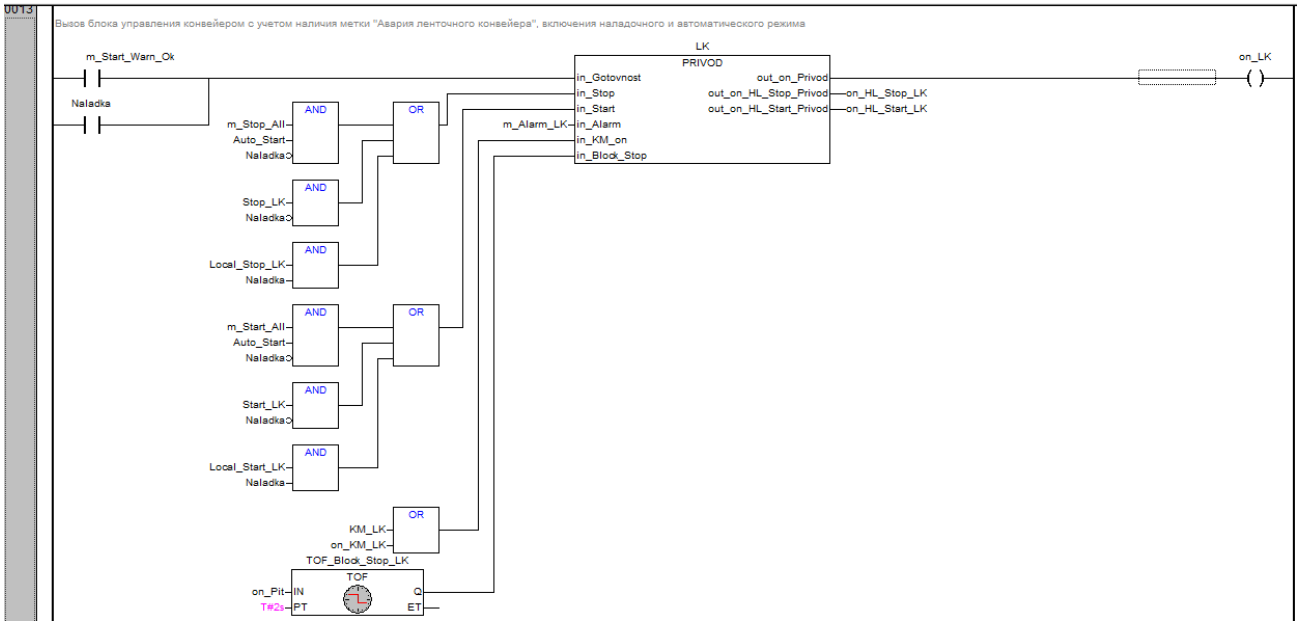


Рис. 35. Вызов блока управления конвейером с учетом наличия метки «Авария ленточного конвейера», включения наладочного и автоматического режима

Рис. 36. Вызов блока управления питателем с учетом наличия метки «Авария шлюзового питателя», включения наладочного и автоматического режима

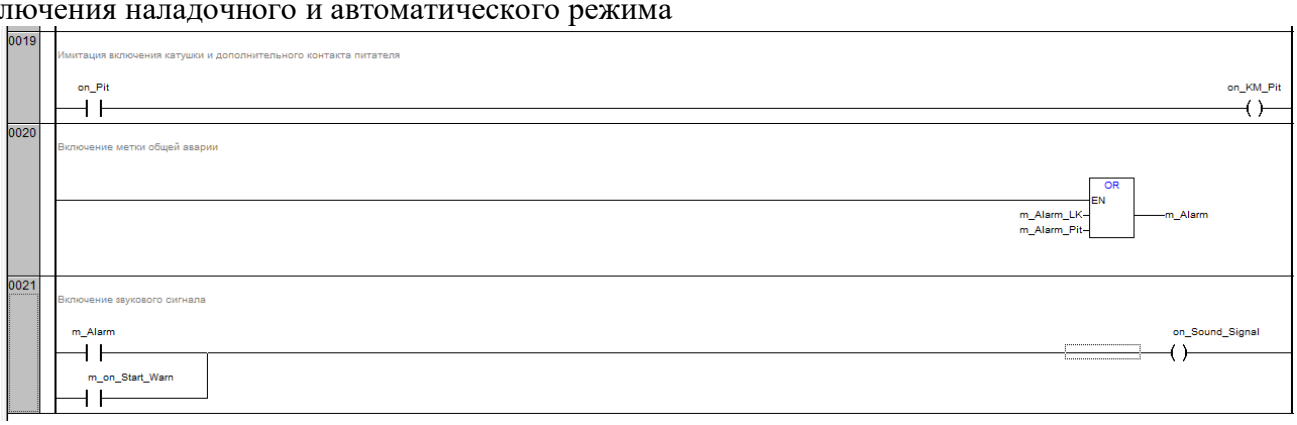


Рис. 37. Включение метки общей аварии и звукового сигнала с ее учетом

На основе конечного проекта в дальнейшем будет построена визуализация, позволяющая управлять технологической линией.

Практическая работа №7
Визуализация системы управления ленточным конвейером в CoDeSys

Внимание: в этой части практикума в качестве основы используется полный проект, выполненный в результате работы над первой частью (ПР №6).
Под термином «визуализация» будем понимать графический человеко-машинный интерфейс, представляющий собой набор условных изображений технологического процесса и его стадий, называемых «мнемосхемами», с отображением значимых технологических параметров и ситуаций, а также средств взаимодействия с автоматизированной системой управления.
На мнемосхемах изображаются (условно) агрегаты технологической линии. Чаще всего создается основная мнемосхема, на которой изображается вся технологическая цепочка и наиболее важные технологические параметры, а также набор мнемосхем для каждого агрегата, на которых можно отследить и задать технические параметры его работы. Также могут быть созданы отдельные мнемосхемы для энергоучета, формирования различного рода отчетности и т.д.
Визуализация на платформе контроллера имеет существенно меньше возможностей по сравнению с визуализацией SCADA-системы. Кроме того, дополнительные возможности контроллера (формирование отчетности, хранение данных и т.д.) на порядок ниже возможностей промышленного компьютера. Однако, для нужд небольших систем управления их может быть вполне достаточно, и тогда визуализация, реализуемая в контроллере может использоваться для отображения и управления с помощью графический панелей оператора.
Основное назначение визуализации – организация рационального и эргономичного взаимодействия оператора и системы управления. Поэтому организация ее объектов должна быть предельна проста и информативна.
Для создания визуализации в CoDeSys нужно перейти на вкладку ***Визуализация*** в организаторе проекта и в контекстном меню выбрать *Добавить объект*. Создайте визуализацию и назовите ее ***PLC_VISU*** (это имя визуализации по умолчанию, которая будет запускаться при старте контроллера). В свойствах созданного объекта на вкладке ***Визуализация*** вы можете указать тип визуализации. ***Управляющая панель*** – это визуализация, автоматически отображаемая на всех других визуализациях. Это может быть меню, критически важные технологические параметры и т.д. ***Визуализация без управляющей панели*** – это визуализация, в которой не отображается имеющаяся в проекте управляющая панель. И наконец, ***Визуализация*** – это стандартный объект визуализации. Для PLC_VISU оставляем это свойство без изменений.
Для вставки элементов в окно визуализации следует воспользоваться панелью элементов, расположенных над окном. Изобразим ленточный конвейер, для чего используем элемент ***Прямоугольник со скругленными углами***. При двойном щелчке на созданном объекте (или выборе пункта контекстного меню ***Конфигурировать***) появляется окно **Конфигурирование элемента**. На вкладках этого окна задаются свойства выбранного объекта и привязываются переменные проекта, управляющие и управляемые этим объектом.
На вкладке ***Форма*** определяется форма фигуры (рис.1).

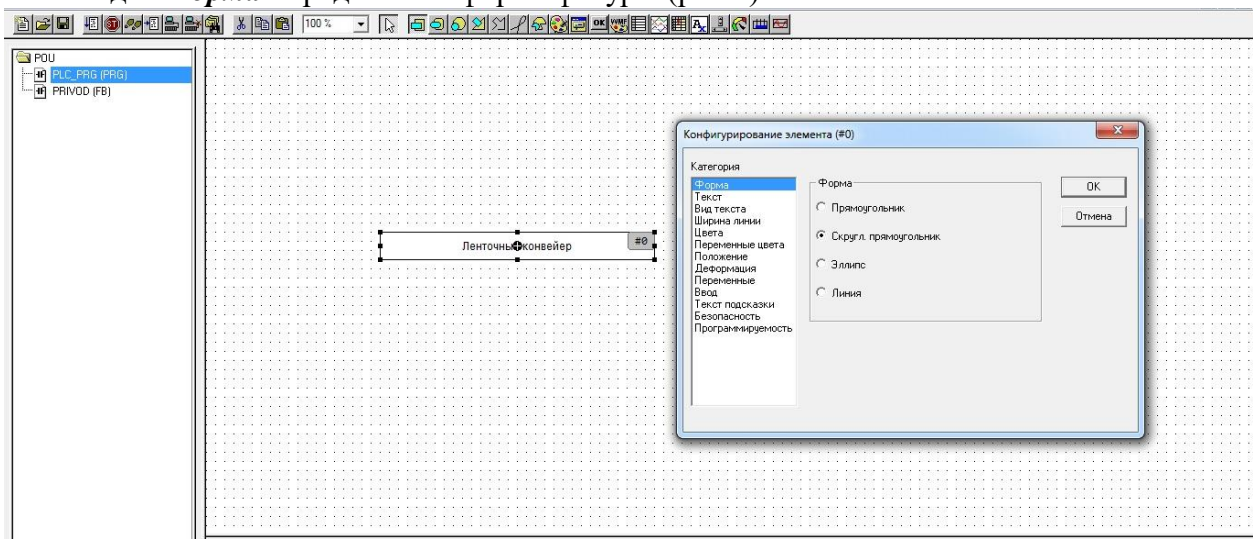


Рис.1. Вкладка «Форма» окна «Конфигурирование элемента»

На вкладке ***Текст*** задается текст, связанный с элементом, а также параметры этого текста. Для данного элемента нужно написать текст «Ленточный конвейер» (рис.2).

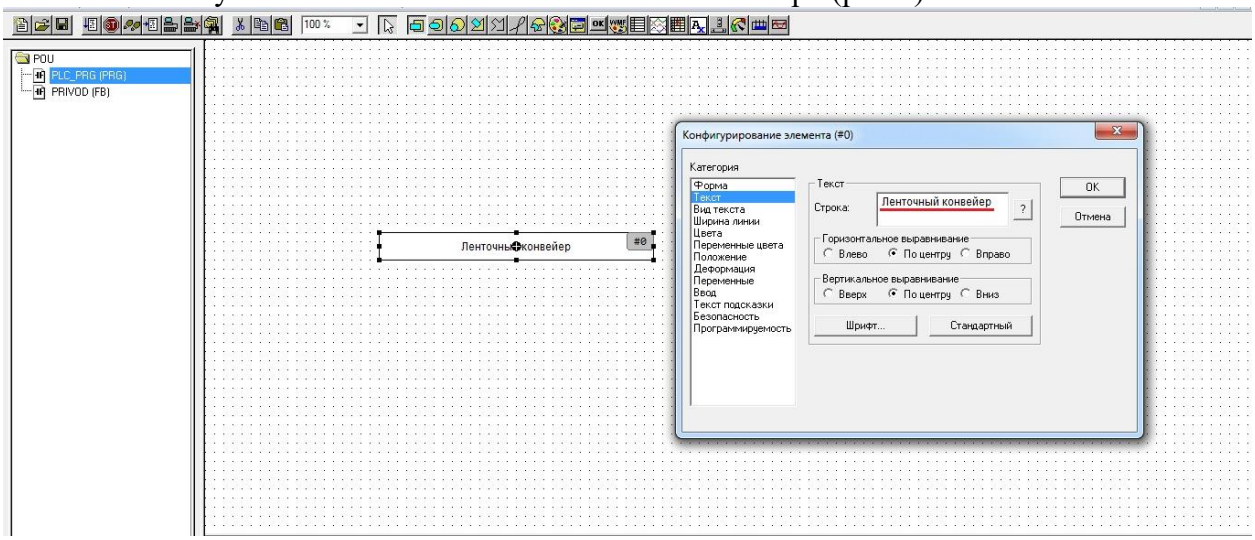


Рис.2. Вкладка «Текст» окна «Конфигурирование элемента»

Далее на вкладке ***Цвета*** нужно изменить цвет заливки на серый (рис.3). Тревожный цвет для объекта «Ленточный конвейер» сейчас изменять не нужно.

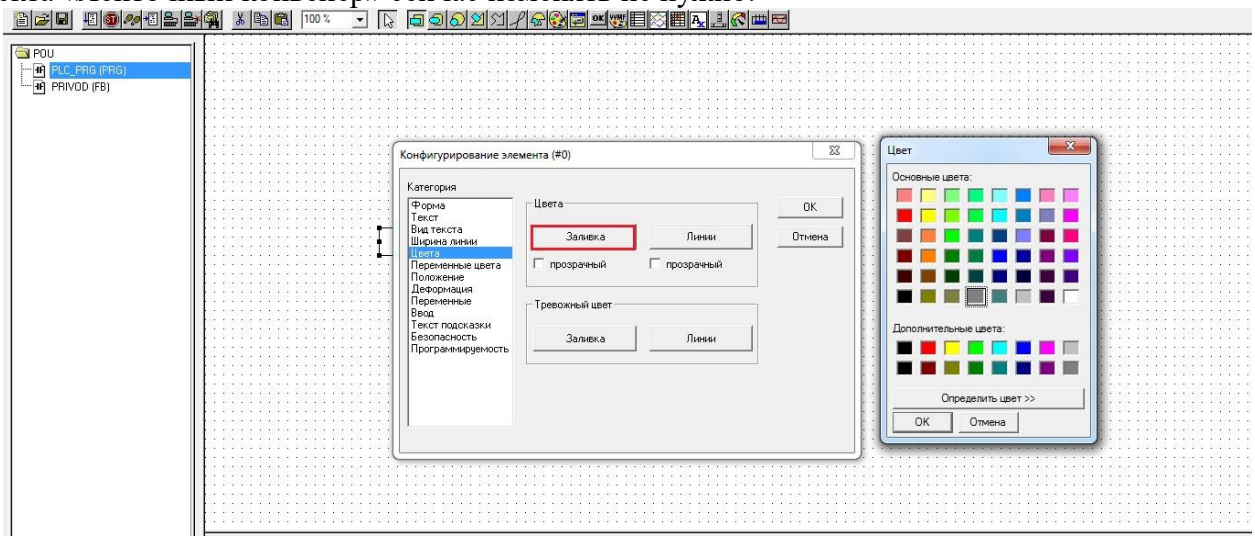


Рис.3. Вкладка «Цвета» окна «Конфигурирование элемента»

Аналогичным образом создайте кнопки «Пуск» и «Стоп» для ленточного конвейера (два круга, один темно зеленый, второй темно красный). Для того чтобы сделать надписи просто добавьте прямоугольники, и на вкладке ***Текст*** введите соответствующие данные (рис.4).

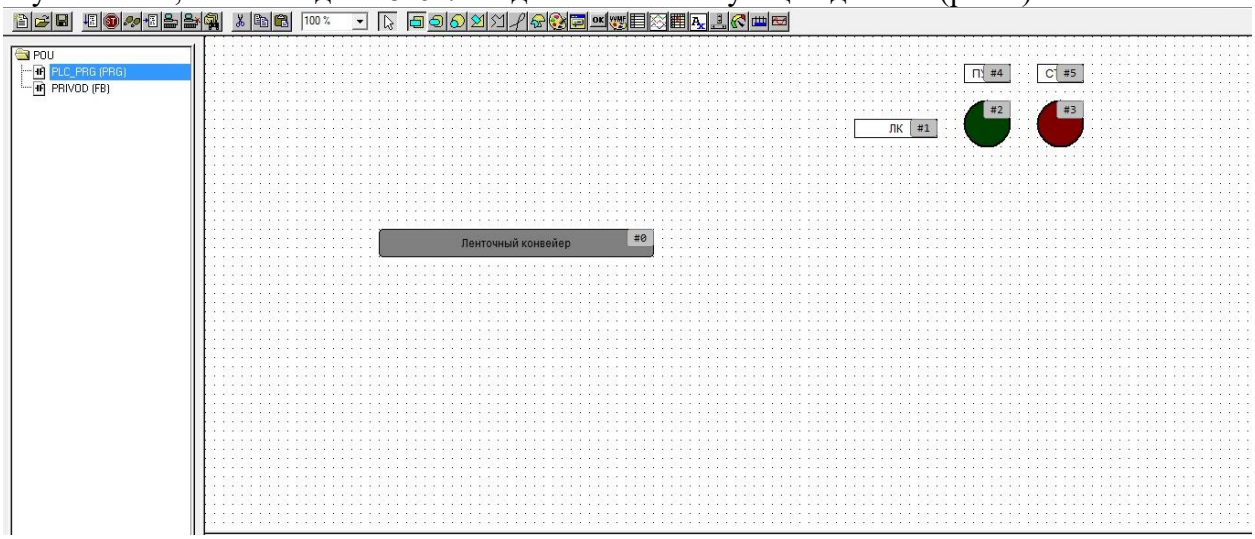


Рис.4. Окно основной визуализации после создания ленточного конвейера и кнопок

Для кнопок измените тревожный цвет. С помощью тревожного цвета можно имитировать лампу, совмещенную с кнопкой. Для кнопки «Пуск» выберите тревожный цвет ярко-зеленым, а для кнопки «Стоп» - ярко-красным.

Для того, чтобы лампы загорались, нужно на вкладке ***Переменные*** окна **Конфигурирование элемента** соответствующей лампы/кнопки в поле *Изменение цвета* ввести имя переменной из конфигурации ПЛК, которая отвечает за включение аналогичного выхода ПЛК (рис.5).

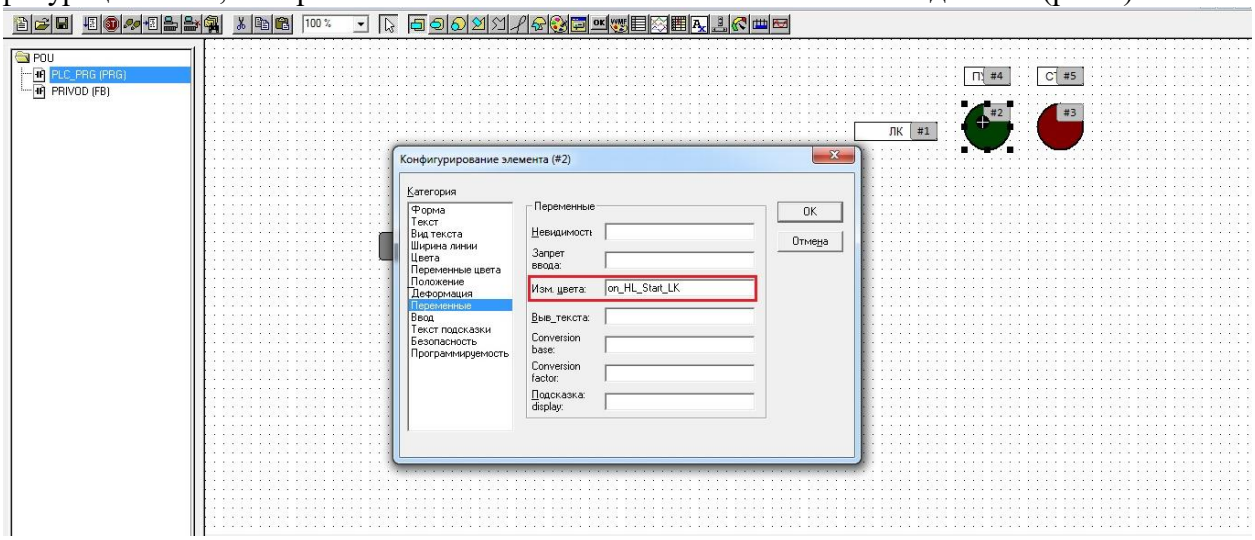


Рис.5. Задание переменной для изменения цвета кнопки/лампы

Задайте переменные для кнопок/ламп, скомпилируйте проект и проверьте корректность работы визуализации. Если лампы не горят, обратитесь к преподавателю.

Теперь нужно добавить функционал кнопок. Допустим, что визуализация нужна нам для отладочных целей и кнопки визуализации включены параллельно кнопкам управления оператора (сигналам с входов ПЛК). Объявим глобальные переменные viz_Start_LK и viz_Stop_LK в отдельно созданной вкладке глобальных переменных (рис.6).

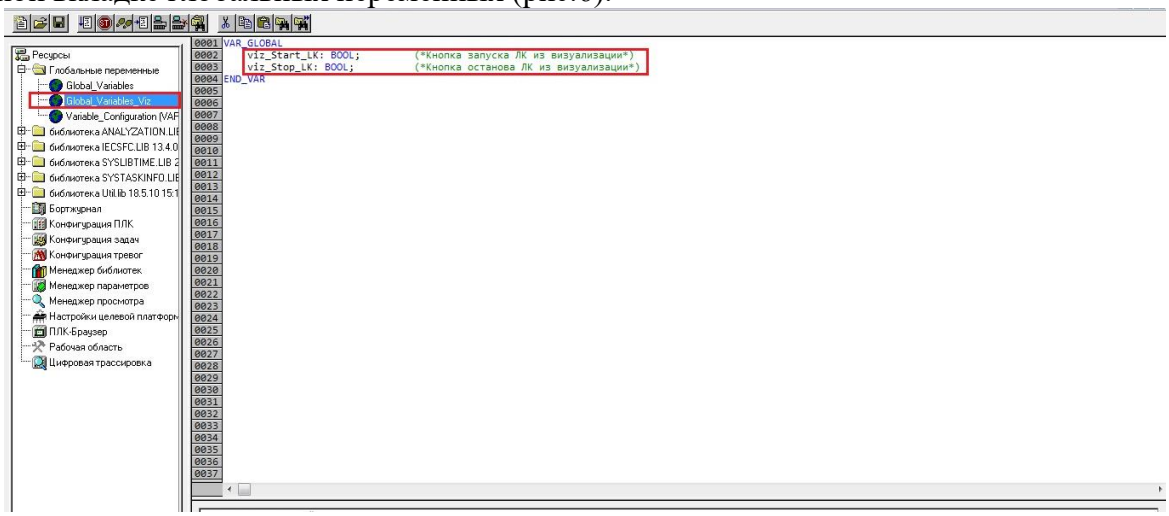


Рис.6. Объявление глобальных переменных визуализации

После того, как глобальные переменные объявлены, нужно осуществить их изменение при нажатии на соответствующую кнопку. Для этого в окне **Конфигурирование элемента** на вкладке ***Ввод*** нужно поставить галочку на пункте *Переменная кнопка* и в текстовом поле ввести имя соответствующей глобальной переменной (рис.7).

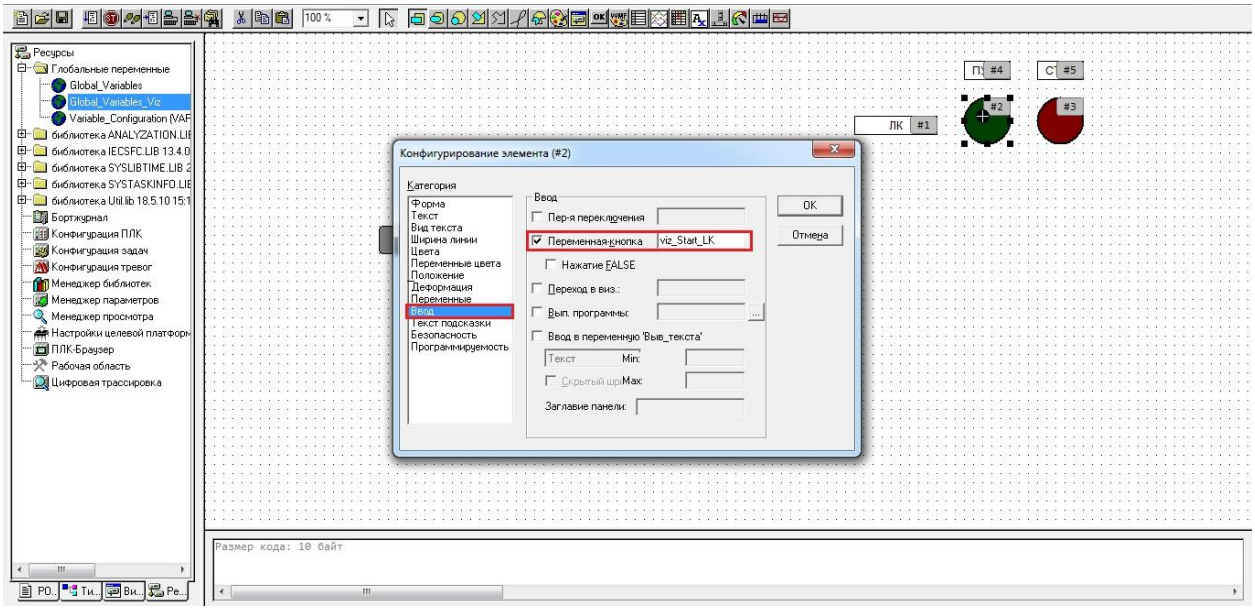


Рис.7. Выбор переменной для ввода при нажатии на кнопку

Теперь при нажатии на кнопку в режиме онлайн, указанная нами переменная будет изменять значение с нуля на единицу. Осталось включить созданные переменные параллельно с входами контроллера. Для этого нужно в пусковой и стоповой цепях блока ленточного конвейера в ***PLC_PRG*** с сигналами запуска и останова с входов ПЛК через блок ИЛИ добавить сигналы из визуализации (рис.8).

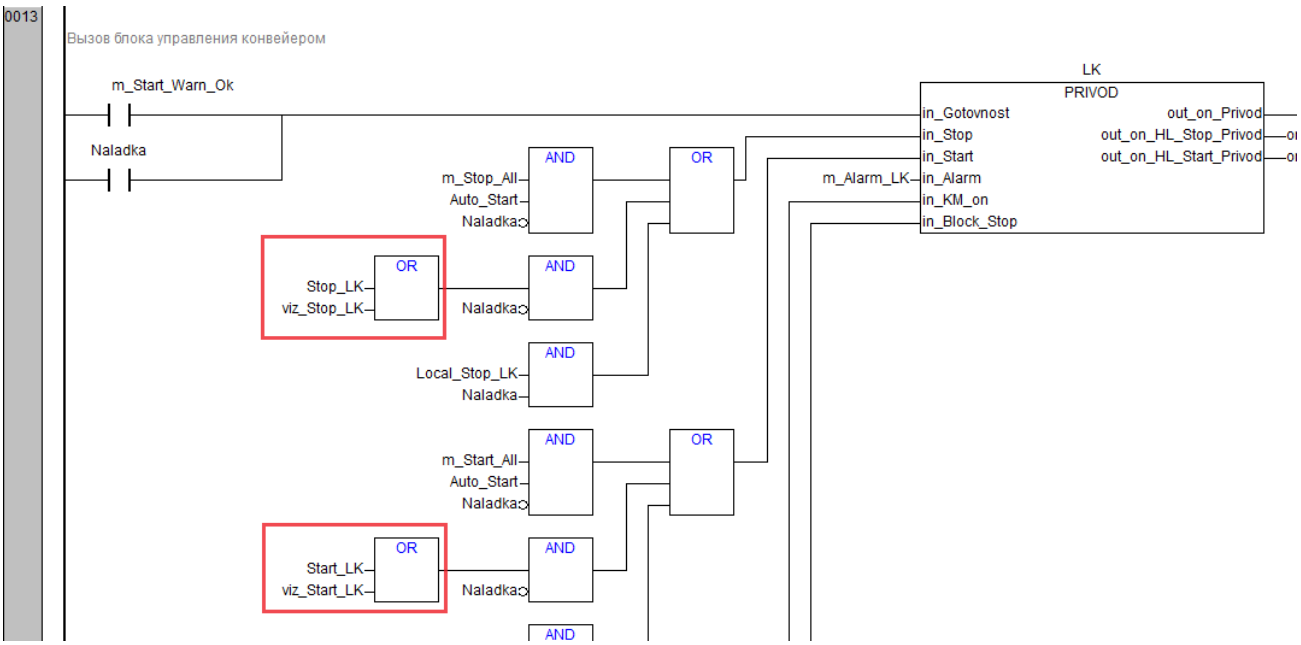


Рис.8. Добавление запуска и останова из визуализации в цепочку пуска и останова ленточного конвейера

Теперь осуществим изменение цвета изображения конвейера таким образом, чтобы он отображал одновременно и сигналы, соответствующие пусковой лампе, и сигналы, соответствующие стоповой лампе. Для этого создадим глобальные переменные viz_Change_Color_LK типа BOOL (переменная для разрешения изменения цвета) и viz_Color_LK типа DWORD (значение цвета), с помощью которых будет реализовано изменение цвета изображения ленточного конвейера (рис.9).

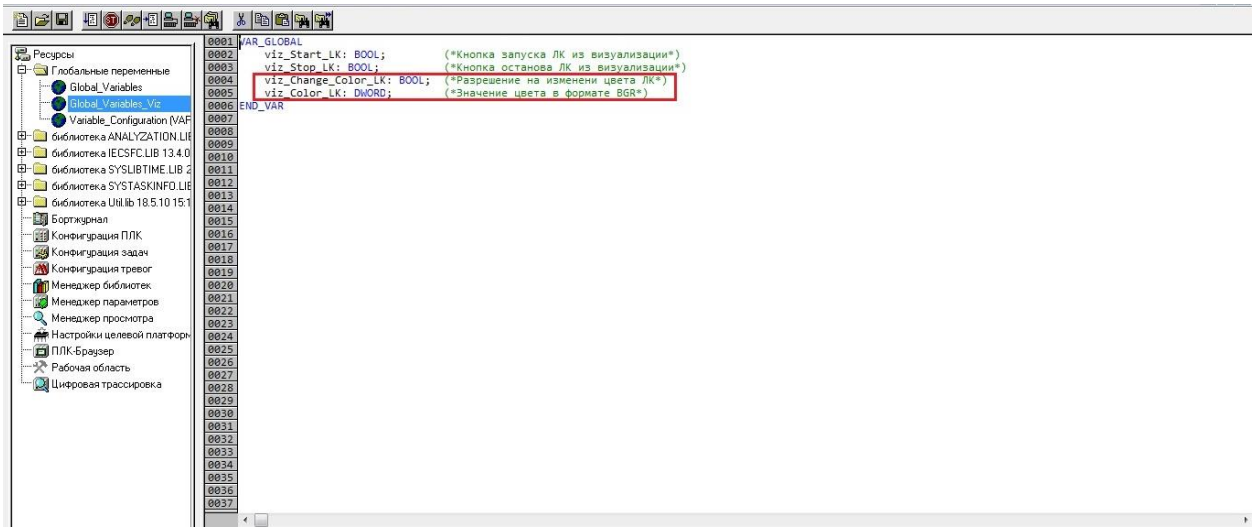


Рис.9. Добавление глобальных переменных для управления цветом изображения ленточного конвейера

Переменную viz_Change_Color_LK нужно указать в поле *Изменение цвета* на вкладке ***Переменные*** окна **Конфигурирование элемента** изображения ленточного конвейера, а переменную viz_Color_LK в поле *Тревожный* вкладки ***Переменные цвета*** (рис.10 и 11).

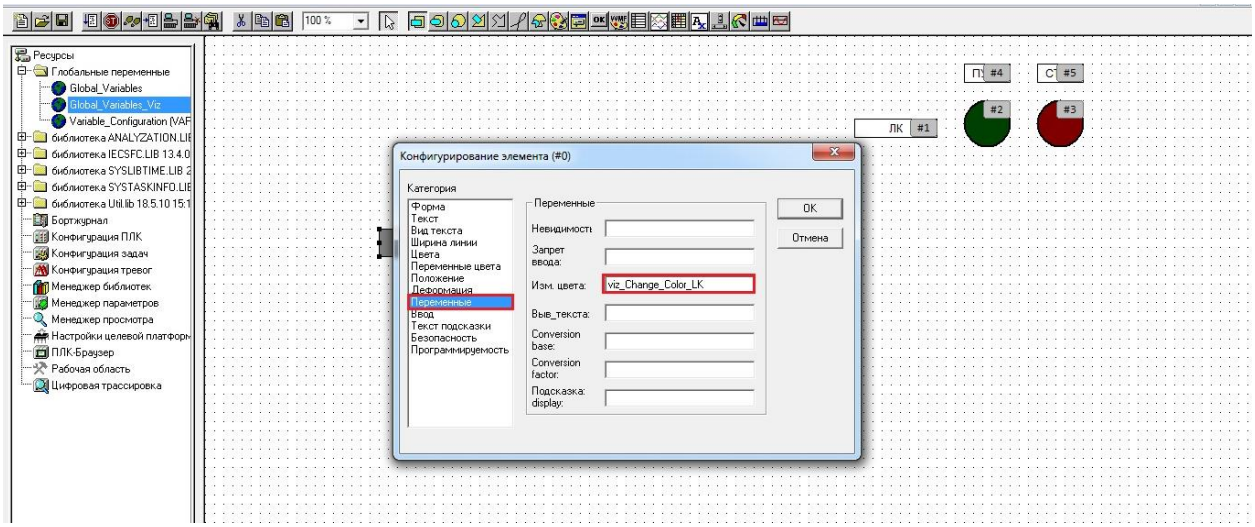


Рис.10. Указание переменной viz_Change_Color_LK

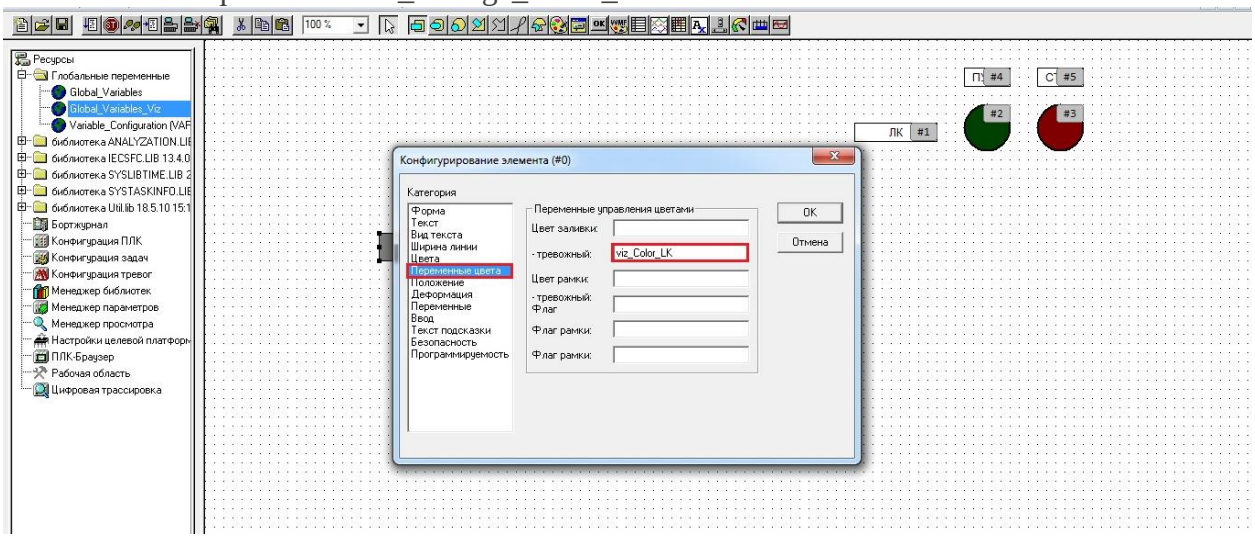


Рис.11. Указание переменной viz_Color_LK

Теперь нужно создать функциональный блок на языке ST для задания значений созданных переменных (назовите его ***Change_Color_Privod***). Логика его работы довольно проста: на вход принимаются сигналы включения сигнальных ламп; если есть сигнал хотя бы одной из ламп, то

выдается выходной сигнал «Разрешение на изменение цвета», а значение принимает цвет в соответствии с имеющимся сигналом (для стартовой лампы – зеленый, для стоповой – красный). Объявление переменных и код функционального блока приведены на рис.12.

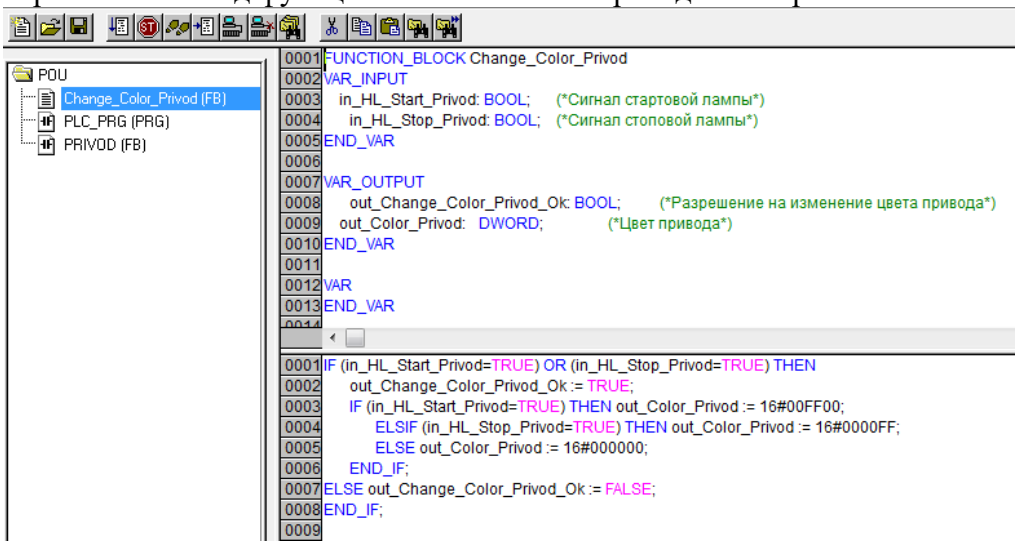


Рис.12. Функциональный блок изменения цвета привода

Чтобы изменять цвет изображения ленточного конвейера, нужно создать для него экземпляр блока *Change_Color_Privod* (назовите его ***FB_Change_Color_LK***). Вызовем этот блок сразу после вызова блока привода ленточного конвейера (**LK**). На вход блока нужно подать сигналы включения ламп запуска и останова ленточного конвейера, а значения выходов присвоить созданным глобальным переменным (рис.13).

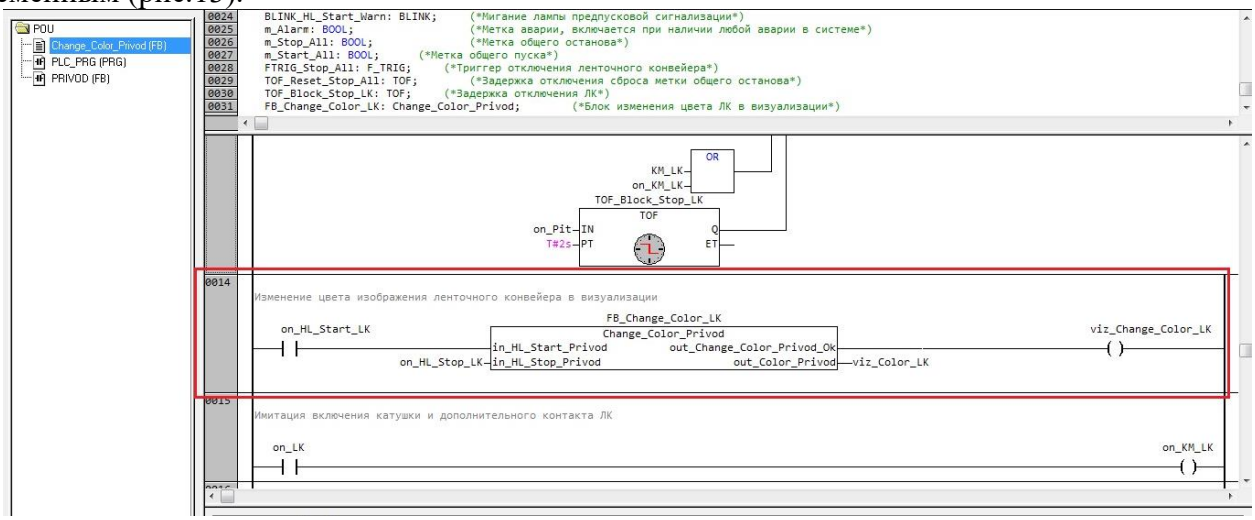


Рис.13. Вызов функционального блока изменения цвета ленточного конвейера

Скомпилируйте проект и запустите его. Проверьте правильность работы визуализации. В случае успешного выполнения переходите к самостоятельной работе.

Самостоятельная работа (общее задание).

По аналогии с ленточным конвейером необходимо создать элементы визуализации для шлюзового питателя (изобразите его многогранником). Результат выполнения задания сверить с преподавателем.

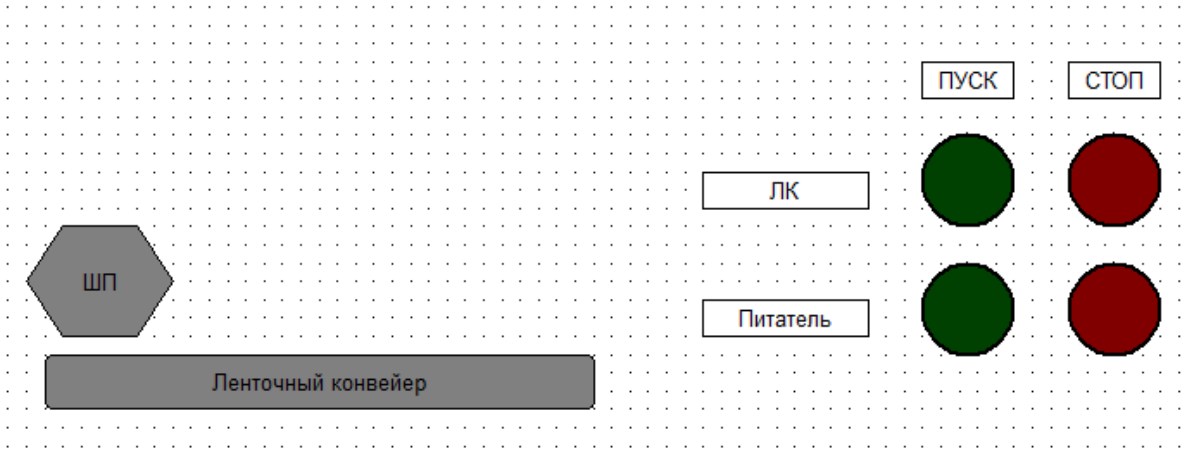


Рис. 14. Добавление элементов для визуализации работы шлюзового питателя

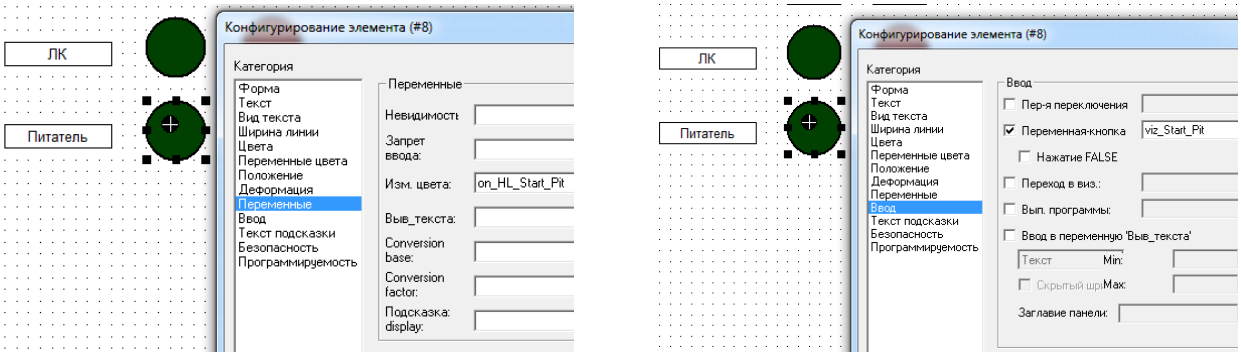


Рис. 15. Задание переменной для изменения цвета кнопки/лампы «Старт» питателя и выбор переменной для ввода при нажатии на кнопку «Старт» питателя

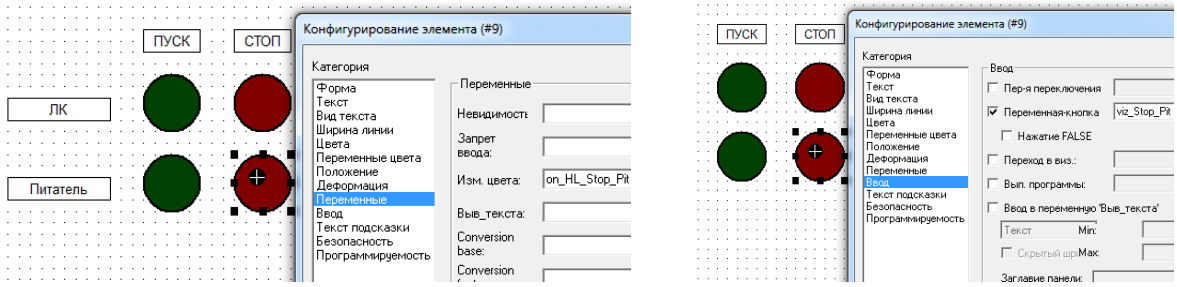


Рис. 16. Задание переменной для изменения цвета кнопки/лампы «Стоп» питателя и выбор переменной для ввода при нажатии на кнопку «Стоп» питателя

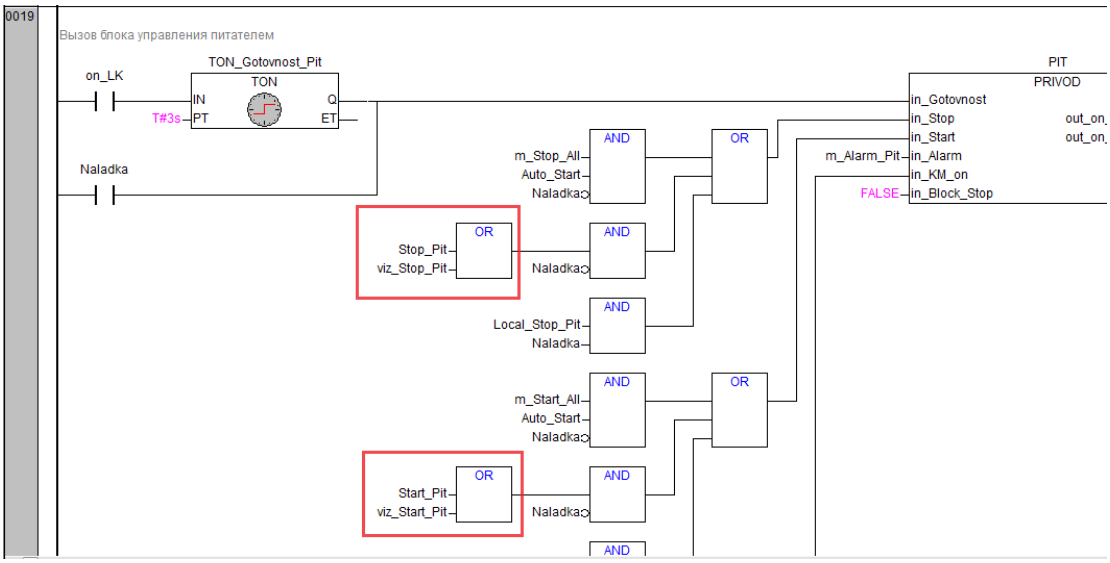


Рис.17. Добавление запуска и останова из визуализации в цепочку пуска и останова шлюзового питателя

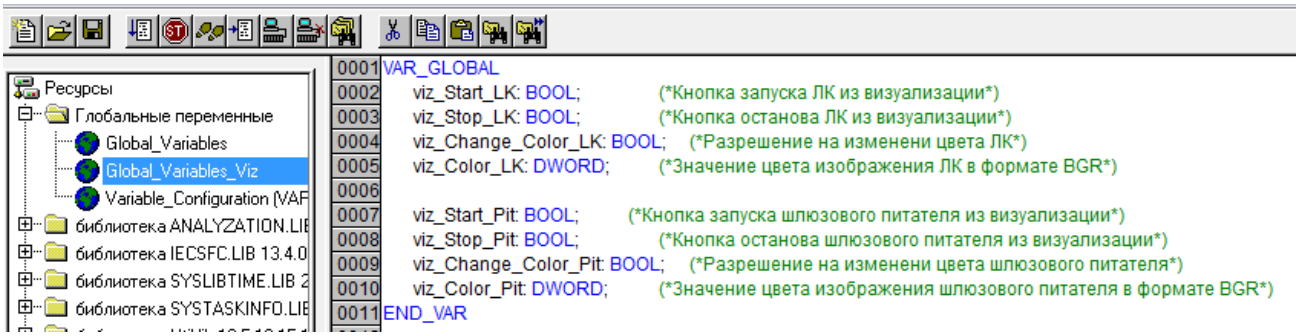


Рис.18. Добавление глобальных переменных для управления цветом изображения шлюзового питателя

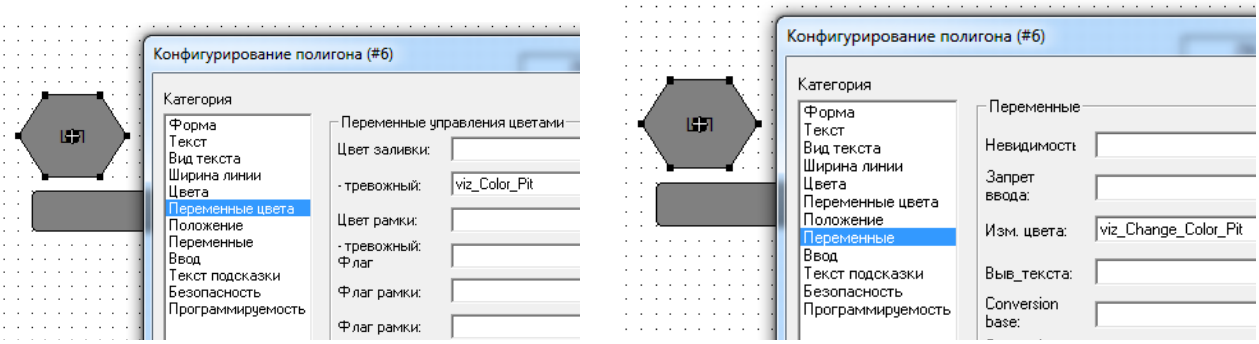


Рис. 19. Привязка глобальных переменных для управления цветом изображения шлюзового питателя

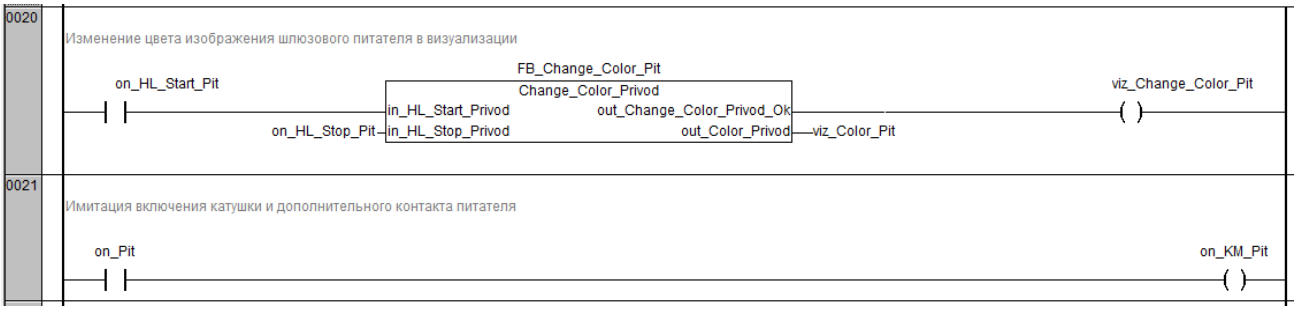


Рис.20. Вызов функционального блока изменения цвета шлюзового питателя

Далее необходимо:

1. Реализовать визуализацию аварийной сигнализации и квитирование аварийной ситуации из визуализации.
2. Реализовать визуализацию предпусковой сигнализации и включение ее из визуализации.
3. Реализовать визуализацию наладочного режима работы (сделать в визуализации местный пост управления и его работу в наладочном режиме).
4. Реализовать визуализацию автоматического режима работы (сделать в визуализации кнопки «Общий Пуск» и «Общий Стоп» и их работу в автоматическом режиме).

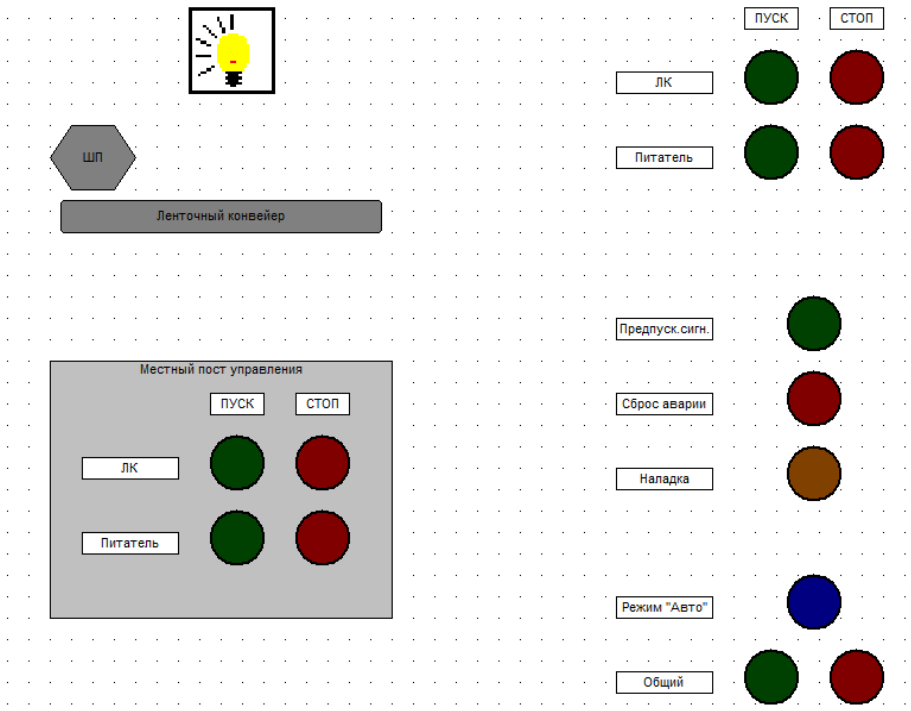


Рис. 21. Добавление элементов для визуализации аварийной и предпусковой сигнализации, наладочного и автоматического режимов работы

```
0001 VAR_GLOBAL
0002     viz_Start_LK: BOOL;            (*Кнопка запуска ЛК из визуализации*)
0003     viz_Stop_LK: BOOL;             (*Кнопка останова ЛК из визуализации*)
0004     viz_Change_Color_LK: BOOL;     (*Разрешение на изменени цвета ЛК*)
0005     viz_Color_LK: DWORD;           (*Значение цвета изображения ЛК в формате BGR*)
0006
0007     viz_Start_Pit: BOOL;         (*Кнопка запуска шлюзового питателя из визуализации*)
0008     viz_Stop_Pit: BOOL;            (*Кнопка останова шлюзового питателя из визуализации*)
0009     viz_Change_Color_Pit: BOOL;    (*Разрешение на изменени цвета шлюзового питателя*)
0010     viz_Color_Pit: DWORD;          (*Значение цвета изображения шлюзового питателя в формате BGR*)
0011
0012     viz_Reset_Alarm: BOOL;         (*Кнопка сброса аварии в визуализации*)
0013     viz_Start_Warn: BOOL;          (*Кнопка предпусковой сигнализации в визуализации*)
0014
0015     viz_on_Sound_Signal: BOOL;     (*Звуковая сигнализация в визуализации*)
0016
0017     viz_Naladka: BOOL;             (*Включение налодочного режима из визуализации*)
0018     viz_Local_Start_LK: BOOL;      (*Кнопка местного запуска ЛК из визуализации*)
0019     viz_Local_Stop_LK: BOOL;       (*Кнопка местного останова ЛК из визуализации*)
0020     viz_Local_Start_Pit: BOOL;     (*Кнопка местного запуска шлюзового питателя из визуализации*)
0021     viz_Local_Stop_Pit: BOOL;      (*Кнопка местного останова шлюзового питателя из визуализации*)
0022
0023     viz_Auto_Start: BOOL;      (*Включение автоматического режима из визуализации*)
0024     viz_Start_All: BOOL;       (*Кнопка "Общий Пуск" в визуализации*)
0025     viz_Stop_All: BOOL;            (*Кнопка "Общий Стоп" в визуализации*)
0026     viz_on_HL_Start_All: BOOL;  (*Ключение лампы "Общий Старт" в визуализации*)
0027     viz_on_HL_Stop_All: BOOL;   (*Ключение лампы "Общий Стоп" в визуализации*)
0028 END_VAR
```

Рис.22. Добавление глобальных переменных для визуализации аварийной и предпусковой сигнализации, наладочного и автоматического режимов работы

```
0013    BLINK_Emul_Move_Pit: BLINK;      (*Блок мигания для эмуляции движения питателя*)
0014    RTRIG_Move_Pit: R_TRIG;          (*Триггер переднего фронта для сигнала датчика движения питателя*)
0015    TOF_No_Move_Pit: TOF;          (*Задержка отключения сигнала "Есть движения питателя"*)
0016    m_Alarm_LK: BOOL;              (*Метка "Авария ленточного конвейера"*)
0017    m_Alarm_Pit: BOOL;             (*Метка "Авария шлюзового питателя"*)
0018    RTRIG_Start_Warn: R_TRIG;      (*Триггер переднего фронта кнопки "Пусковая сигнализация"*)
0019    TP_Start_Warn: TP;             (*Импульс предпусковой сигнализации*)
0020    m_on_Start_Warn: BOOL;           (*Сигнал включения предпусковой сигнализации*)
0021    FTRIG_Start_Warn: F_TRIG;      (*Триггер заднего фронта включения предпусковой сигнализации*)
0022    m_Start_Warn_Ok: BOOL;           (*Метка "Предпусковая сигнализация проведена"*)
0023    FTRIG_on_LK: F_TRIG;           (*Триггер отключения ЛК*)
0024    BLINK_HL_Start_Warn: BLINK;      (*Мигание лампы предпусковой сигнализации*)
0025    m_Alarm: BOOL;                 (*Метка аварии, включается при наличии любой аварии в системе*)
0026    m_Stop_All: BOOL;              (*Метка общего останова*)
0027    m_Start_All: BOOL;         (*Метка общего пуска*)
0028    FTRIG_Stop_All: F_TRIG;      (*Триггер отключения ленточного конвейера*)
0029    TOF_Reset_Stop_All: TOF;       (*Задержка отключения сброса метки общего останова*)
0030    TOF_Block_Stop_LK: TOF;        (*Задержка отключения ЛК*)
0031    FB_Change_Color_LK: Change_Color_Privod;      (*Блок изменения цвета ЛК в визуализации*)
0032    FB_Change_Color_Pit: Change_Color_Privod;     (*Блок изменения цвета питателя в визуализации*)
0033    RTRIG_Naladka: R_TRIG;         (*Триггер включения ключа "Наладка"*)
0034    RTRIG_viz_Naladka: R_TRIG;         (*Триггер включения кнопки "Наладка" в визуализации*)
0035    m_Naladka: BOOL;         (*Метка наладочного режима работы*)
0036    FTRIG_Naladka: F_TRIG;     (*Триггер отключения ключа "Наладка"*)
0037    FTRIG_viz_Naladka: F_TRIG;     (*Триггер отключения кнопки "Наладка" в визуализации *)
0038    m_Auto_Start: BOOL;      (*Метка автоматического режима работы*)
0039    RTRIG_Auto_Start: R_TRIG;      (*Триггер включения ключа "Авто"*)
0040    RTRIG_viz_Auto_Start: R_TRIG;      (*Триггер включения автоматического режима из визуализации*)
0041    FTRIG_Auto_Start: F_TRIG;      (*Триггер отключени ключа "Авто"*)
0042    FTRIG_viz_Auto_Start: F_TRIG;      (*Триггер отключения автоматического режима из визуализации*)
0043END_VAR
```

Рис.23. Добавление локальных переменных для визуализации аварийной и предпусковой сигнализации, наладочного и автоматического режимов работы

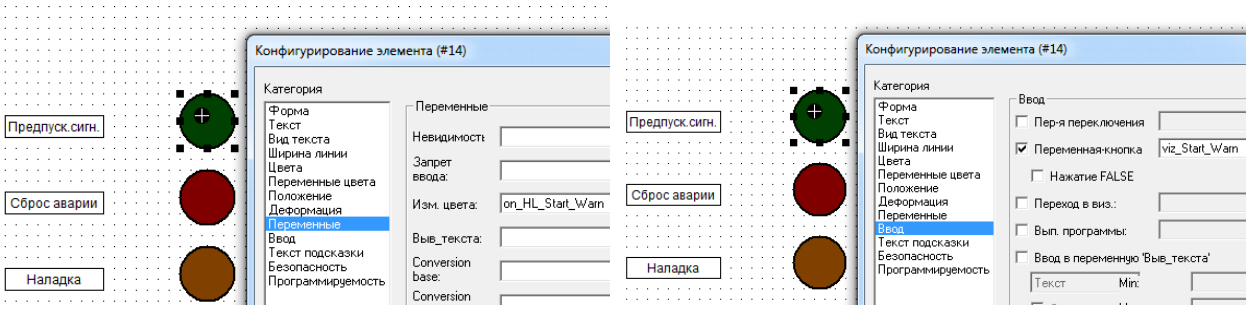


Рис. 24. Задание переменной для изменения цвета кнопки/лампы «Предпусковая сигнализация» и выбор переменной для ввода при нажатии на кнопку «Предпусковая сигнализация»

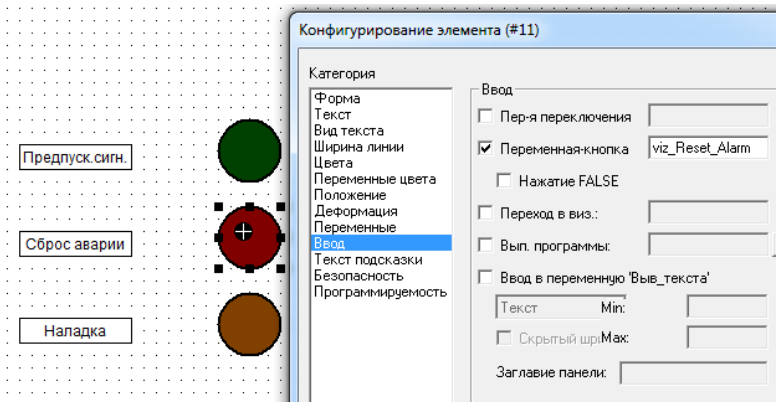


Рис. 25. Выбор переменной для ввода при нажатии на кнопку «Сброс аварии»

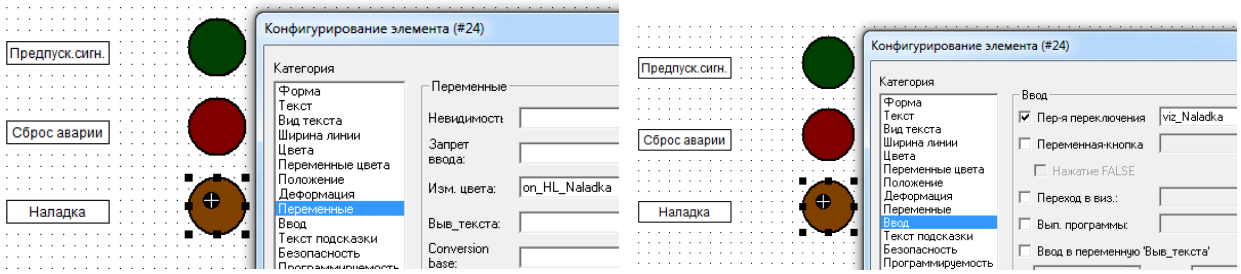


Рис. 26. Задание переменной для изменения цвета кнопки/лампы «Наладка» и выбор переменной переключения для ввода при нажатии на кнопку «Наладка»

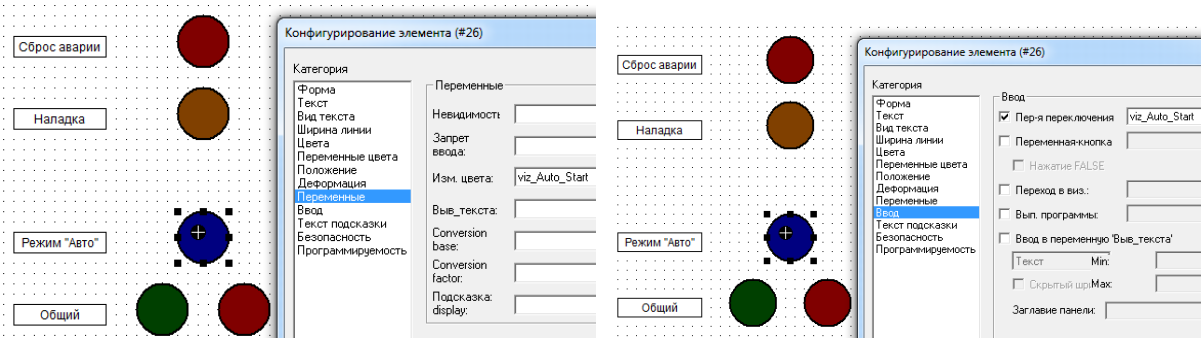


Рис. 27. Задание переменной для изменения цвета кнопки/лампы «Авто» и выбор переменной переключения для ввода при нажатии на кнопку «Авто»

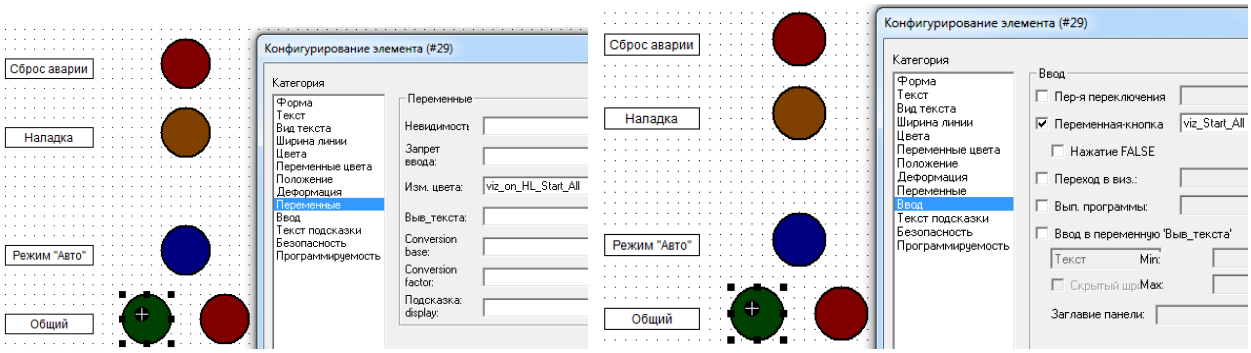


Рис. 28. Задание переменной для изменения цвета кнопки/лампы «Общий ПУСК» и выбор переменной переключения для ввода при нажатии на кнопку «Общий ПУСК»
(в автоматическом режиме)

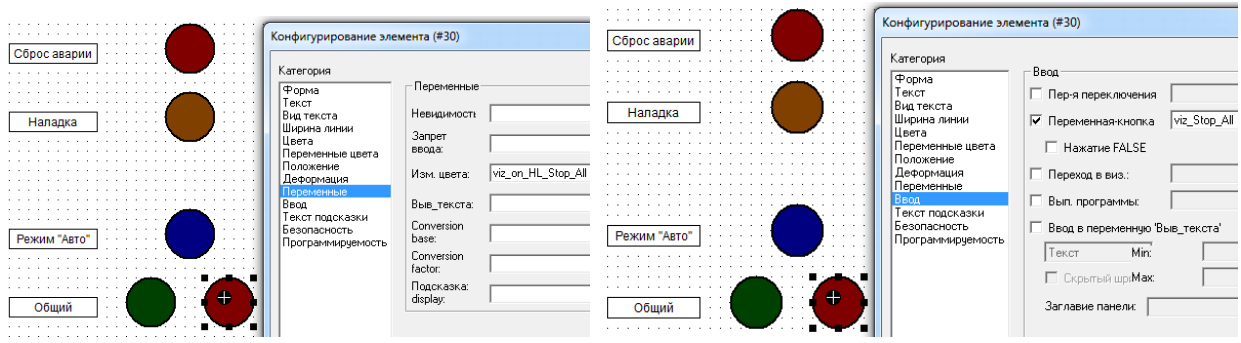


Рис. 29. Задание переменной для изменения цвета кнопки/лампы «Общий СТОП» и выбор переменной переключения для ввода при нажатии на кнопку «Общий СТОП»
(в автоматическом режиме)

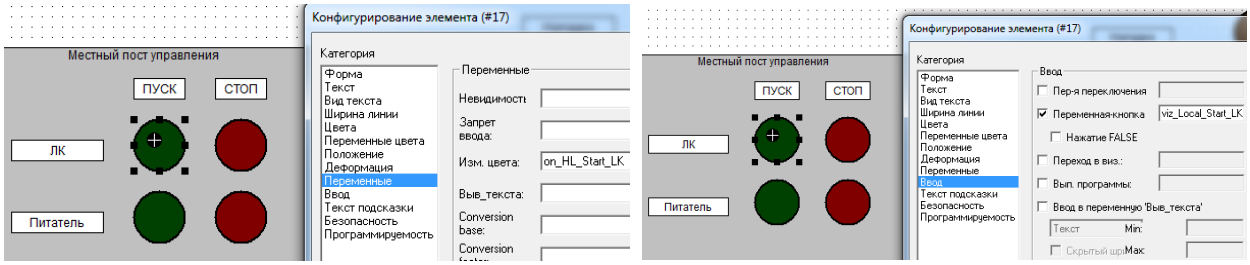


Рис. 30. Задание переменной для изменения цвета кнопки/лампы «Пуск» ленточного конвейера и выбор переменной переключения для ввода при нажатии на кнопку «Пуск» ленточного конвейера (при управлении с местного поста)

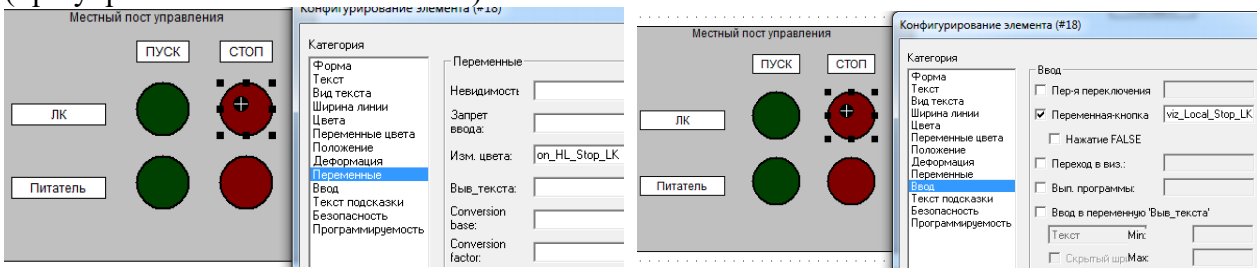


Рис. 31. Задание переменной для изменения цвета кнопки/лампы «Стоп» ленточного конвейера и выбор переменной переключения для ввода при нажатии на кнопку «Стоп» ленточного конвейера (при управлении с местного поста)

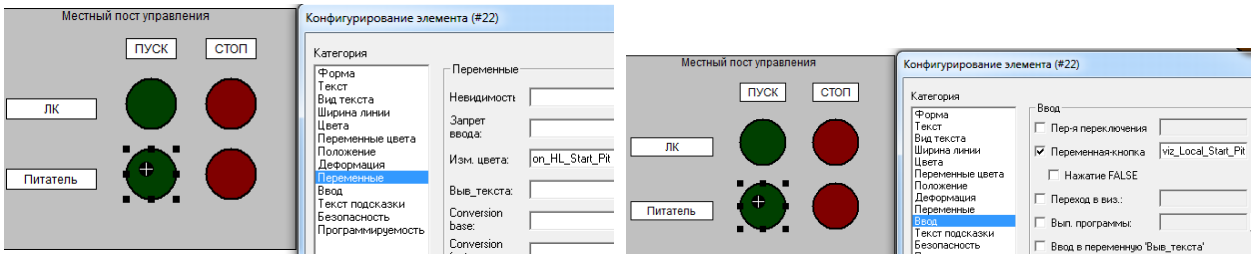


Рис. 32. Задание переменной для изменения цвета кнопки/лампы «Пуск» шлюзового питателя и выбор переменной переключения для ввода при нажатии на кнопку «Пуск» шлюзового питателя (при управлении с местного поста)

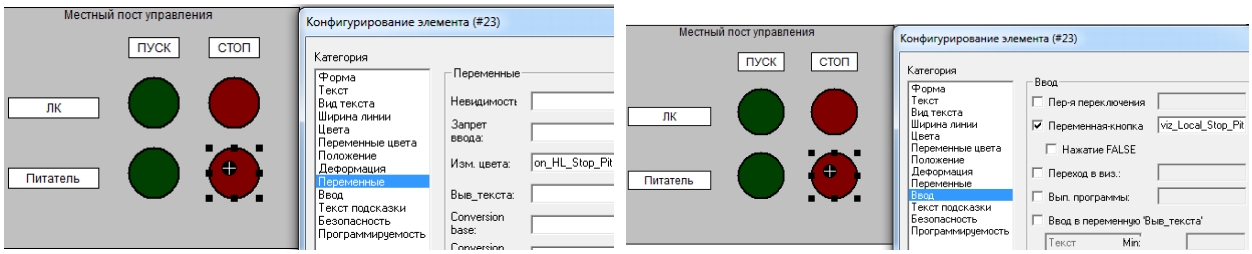


Рис. 33. Задание переменной для изменения цвета кнопки/лампы «Стоп» шлюзового питателя и выбор переменной переключения для ввода при нажатии на кнопку «Стоп» шлюзового питателя (при управлении с местного поста)

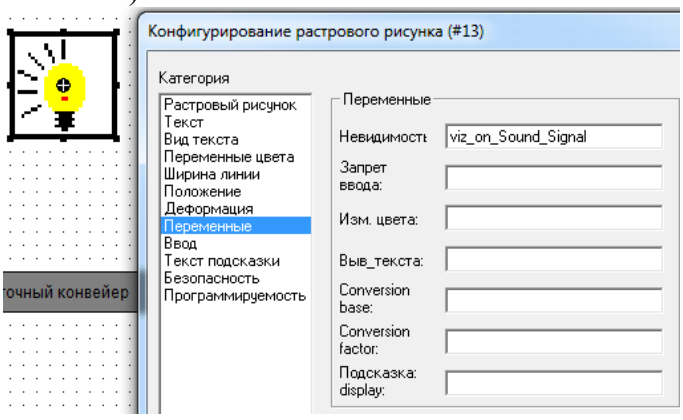
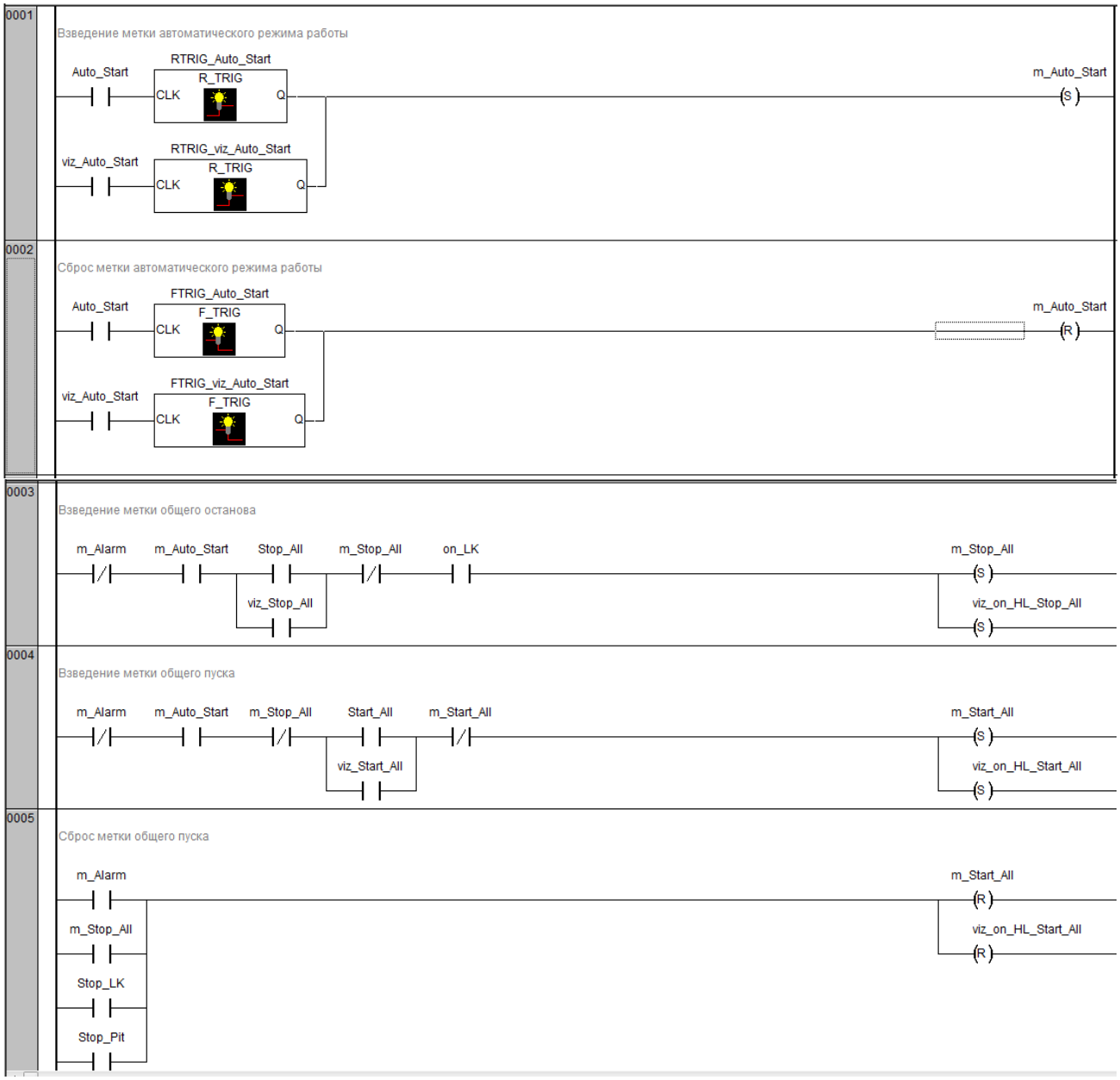


Рис. 34. Задание переменной для изменения видимости сигнальной лампы

(имитация звукового сигнала)



0006

Сброс метки общего останова

0007

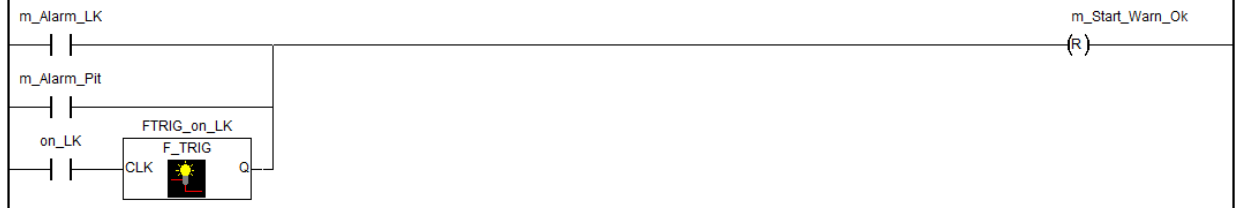
Включение предпусковой сигнализации на заданный промежуток времени

0008

Взведение метки "Предпусковая сигнализация проведена"

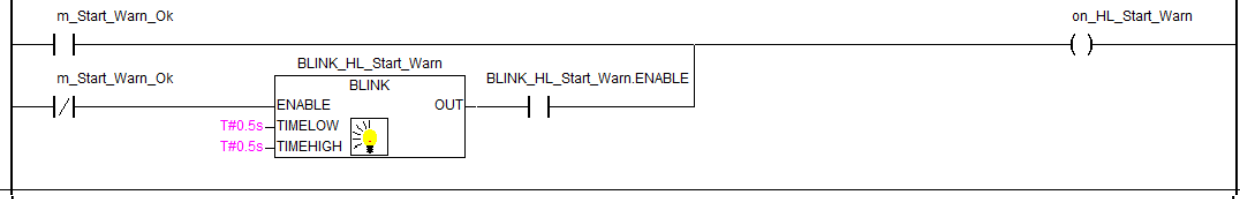
0009

Сброс метки "Предпусковая сигнализация" по аварии или по сигналу отключения привода, который должен отключиться последним



0010

Включение лампы предпусковая сигнализация

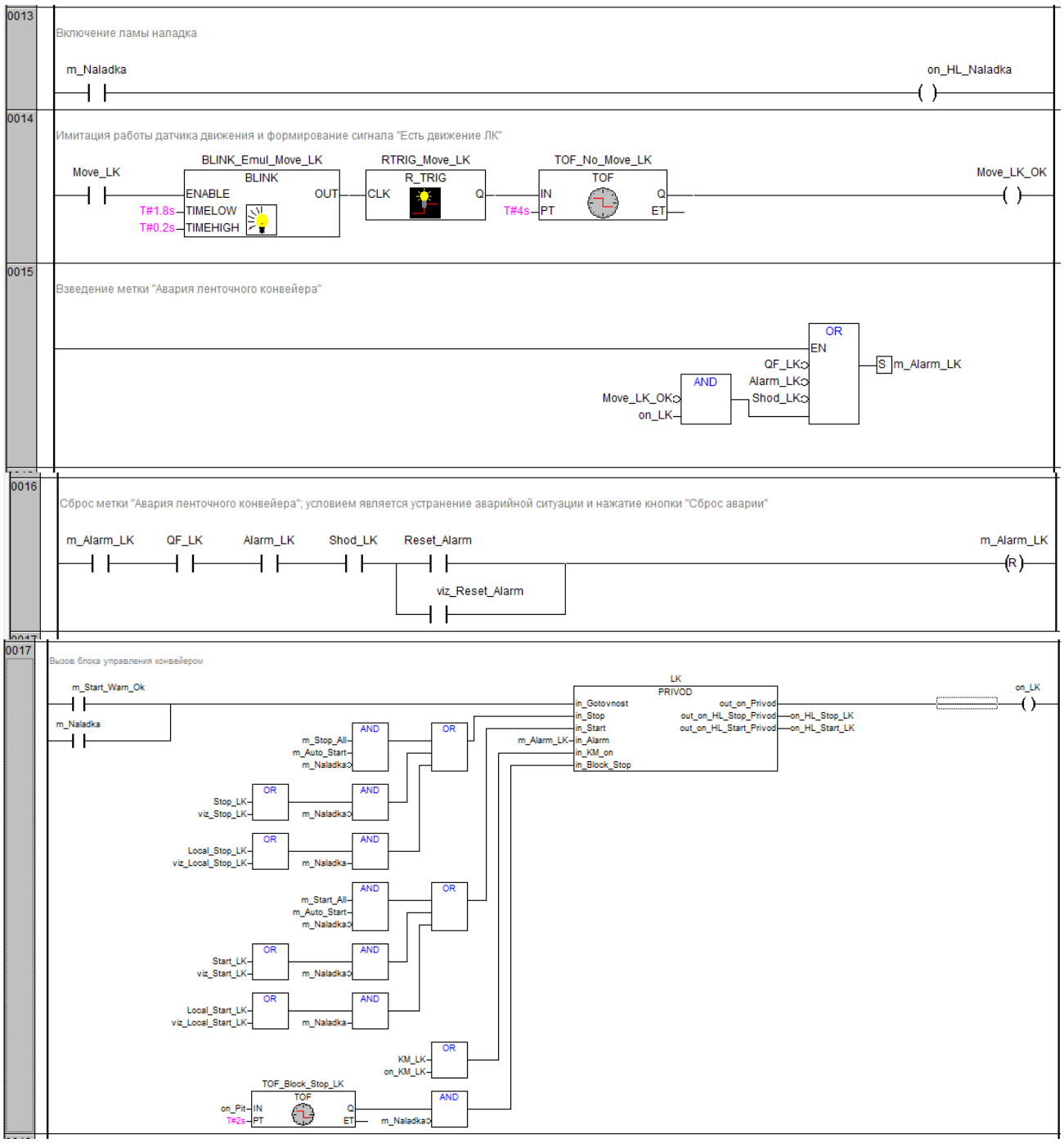


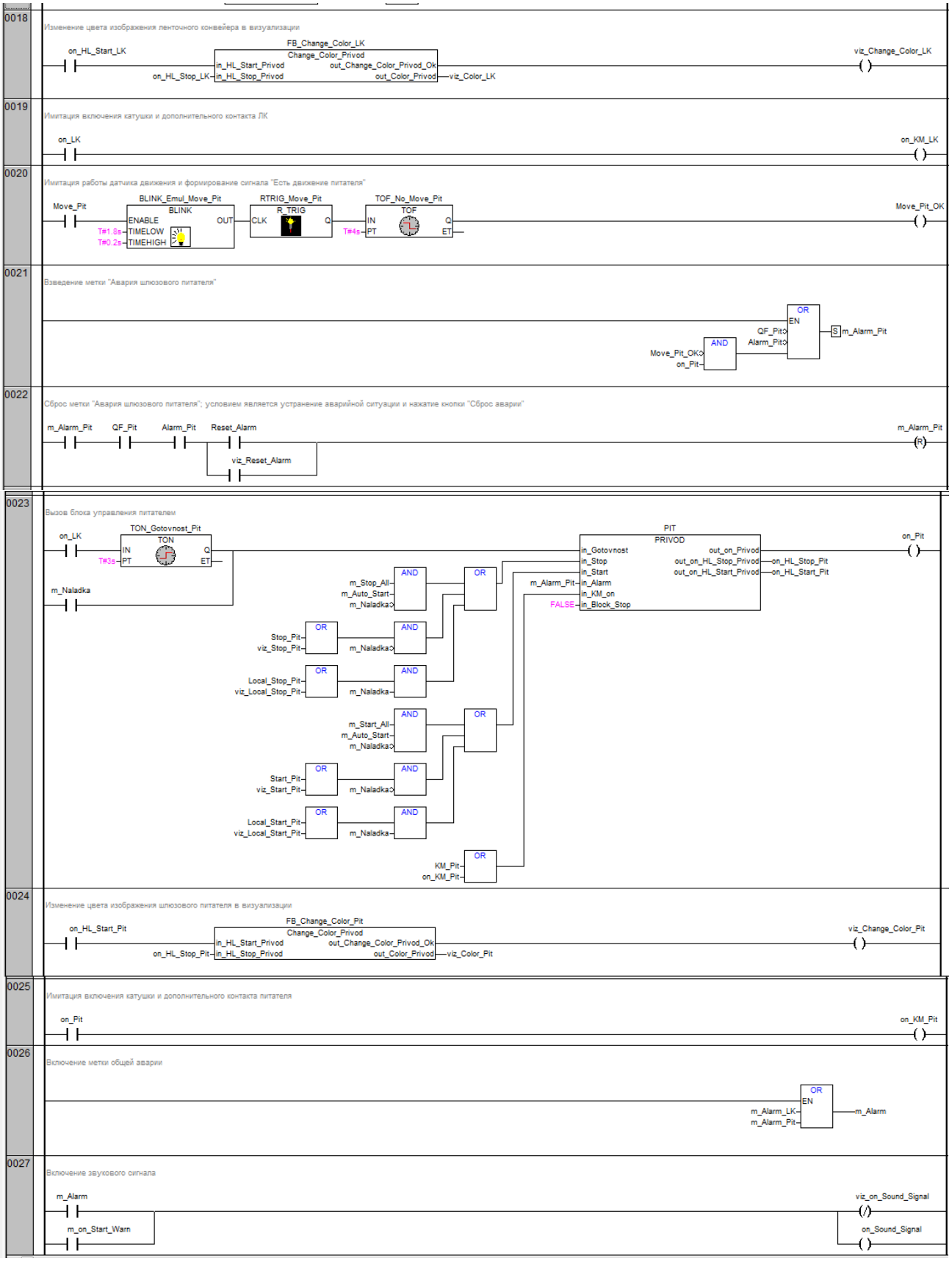
0011

Взведение метки наладочного режима работы

0012

Сброс метки наладочного режима работы


0013
Включение ламы наладка
m_Naladka
on_HL_Naladka
0014
Имитация работы датчика движения и формирование сигнала "Есть движение ЛК"
Move_LK
BLINK_Emul_Move_LK
BLINK
ENABLE
OUT
T#1.8s
TIMELOW
T#0.2s
TIMEHIGH
RTRIG_Move_LK
R_TRIG
CLK
Q
TOF_No_Move_LK
TOF
IN
Q
T#4s
PT
ET
Move_LK_OK
0015
Введение метки "Авария ленточного конвейера"
OR
EN
QF_LK
Alarm_LK
Shod_LK
AND
Move_LK_OK
on_LK
S
m_Alarm_LK
0016
Сброс метки "Авария ленточного конвейера"; условием является устранение аварийной ситуации и нажатие кнопки "Сброс аварии"
m_Alarm_LK
QF_LK
Alarm_LK
Shod_LK
Reset_Alarm
viz_Reset_Alarm
m_Alarm_LK
R
0017
Вызов блока управления конвейером
m_Start_Warn_Ok
m_Naladka
LK
PRIVOD
in_Gotovnost
in_Stop
in_Start
in_Alarm
in_KM_on
in_Block_Stop
out_on_Privod
out_on_HL_Stop_Privod
out_on_HL_Start_Privod
on_HL_Stop_LK
on_HL_Start_LK
on_LK
m_Alarm_LK
m_Stop_All
m_Auto_Start
Stop_LK
viz_Stop_LK
Local_Stop_LK
viz_Local_Stop_LK
m_Start_All
Start_LK
viz_Start_LK
Local_Start_LK
viz_Local_Start_LK
KM_LK
on_KM_LK
TOF_Block_Stop_LK
on_Pit
T#2s
AND
OR



ЗАПУСК В РАБОТУ: Онлайн – Подключение – Ресурсы – Конфигурация ПЛК (установить входные сигналы контроллера, см. рис. 35) – Онлайн – Старт – Визуализация.
При правильной работе при нажатии «Старт» должна мигать лампа «Предпусковая сигнализация». При ее нажатии включается звуковой сигнал, через 5 секунд начинает мигать лампа «Пуск» ЛК и сам ЛК, информируя о готовности ЛК к работе. После запуска ЛК через 3 секунды начинает мигать лампа «Пуск» питателя и сам питатель, информируя о готовности питателя к работе. Затем можно

нажать кнопку «Пуск» питателя, запуская тем самым всю линию в работу. Если линия работает в ручном режиме, то чтобы ее остановить, нужно сначала остановить питатель (кнопка «Стоп», при этом замигает лампа «Пуск» питателя), и только после этого отключить ЛК. До отключения питателя ЛК отключить не получится.

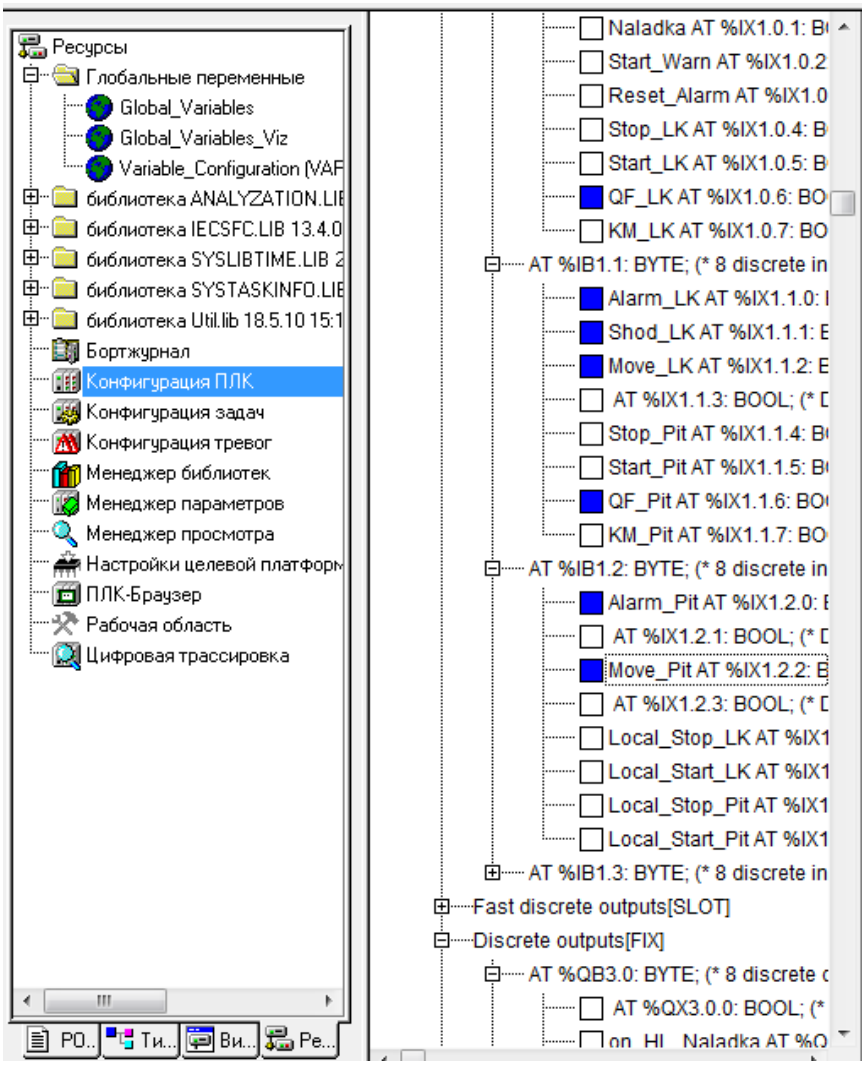


Рис. 35. Установка входных сигналов контроллера

АВАРИЯ - убрать сигнал Shod_LK. При этом ЛК замигает красным цветом, замигает красным цветом кнопка «Стоп» ЛК и будет мигать лампа предпусковой сигнализации, информируя о том, что для последующего запуска после устранения аварии нужно снова нажать кнопку предпусковой сигнализации. Оператор должен устранить аварию (снова установить сигнал Shod_LK) и нажать кнопку «Сброс аварии», подтверждая, что авария устранена.

НАЛАДОЧНЫЙ РЕЖИМ: при нажатии кнопки «Наладка» мигают ЛК, питатель, все пусковые лампы, и предпусковая сигнализация. Запуск агрегатов возможен только с местного поста управления. Можно поочередно запускать и проверять правильность работы ЛК и питателя. Когда наладка завершена, нужно снова нажать кнопку «Наладка», при этом агрегаты останавливаются, и снова загорается лампа предпусковой сигнализации.

АВТОМАТИЧЕСКИЙ РЕЖИМ: запускаем переключатель «Авто» и даем общий «Пуск», при этом линия начинает автоматически работать в соответствии с технологией: подается звуковой сигнал, запускается ЛК, запускается питатель. При нажатии общего «Стоп» агрегаты останавливаются, и затем загорается предпусковая сигнализация.

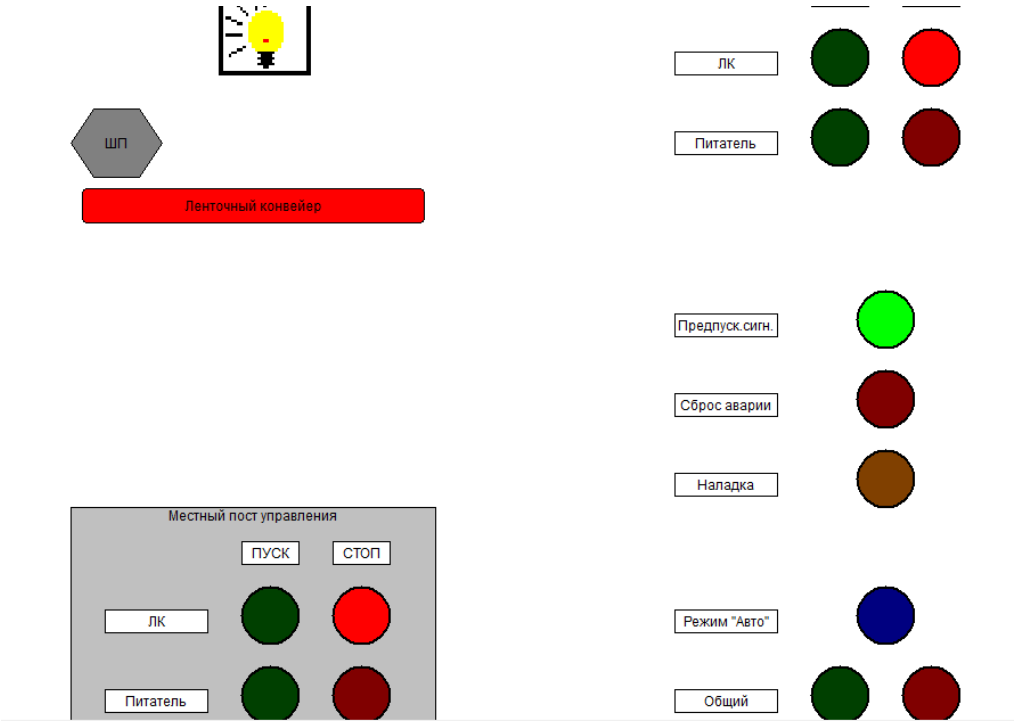


Рис. 36. Визуализация аварийной ситуации

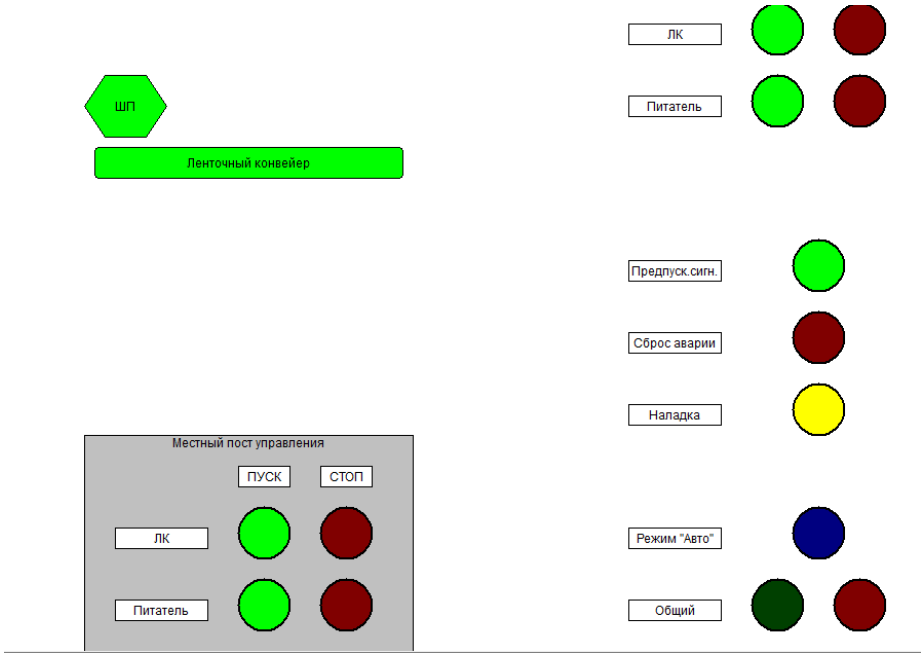


Рис. 37. Визуализация наладочного режима