МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Утверждаю
Директор ИШЭ
_____ А.С. Матвеев
«16» 11 2018 г.

А.С. Гирник

СИСТЕМЫ АВТОМАТИКИ И УПРАВЛЕНИЯ НА БАЗЕ ПРОГРАММИРУЕМЫХ ЭЛЕКТРОННЫХ АППАРАТОВ ПРОИЗВОДСТВА EATON

Методические указания к выполнению лабораторных работ
по профилям «Электромеханические и электротехнические системы автономных объектов», «Электрооборудование летательных аппаратов», «Электропривод и автоматика», а также по дисциплинам «Микропроцессорные средства систем автоматики, управления и диагностики», «Микропроцессорные устройства в электрооборудовании автономных объектов», «Мехатронные системы летательных аппаратов» для студентов, обучающихся по направлению 13.03.02 «Электроэнергетика и электротехника»

Издательство
Томского политехнического университета
2018

УДК 681.584(076.5)
ББК 32.965.6я73
Г82

Гирник А.С.

Г82 Системы автоматики на базе программируемых электронных аппаратов производства Eaton : методические указания к выполнению лабораторных работ по профилям «Электромеханические и электротехнические системы автономных объектов», «Электрооборудование летательных аппаратов», «Электропривод и автоматика», а также по дисциплинам «Микропроцессорные средства систем автоматики, управления и диагностики», «Микропроцессорные устройства в электрооборудовании автономных объектов», «Мехатронные системы летательных аппаратов» для студентов, обучающихся по направлению 13.03.02 «Электроэнергетика и электротехника» / А.С. Гирник ; Томский политехнический университет. – Томск : Изд-во Томского политехнического университета, 2018. – 90 с.

УДК 681.584(076.5)
ББК 32.965.6я73

Методические указания рассмотрены и рекомендованы
к изданию методическим семинаром отделения электроэнергетики
и электротехники ИШЭ «16» ноября 2018 г.

Руководитель ООЭ
кандидат технических наук _____ Ю.Н. Дементьев

Председатель учебно-методической
комиссии _____ Н.М. Космынина

Рецензент
Доктор технических наук, профессор
руководитель профиля «Электрооборудование
летательных аппаратов» ИШЭ ООЭ ТПУ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С появлением программируемых реле, предназначенных для реализации алгоритмов логического управления путем замены релейно-контактных схем, собранных на дискретных компонентах, можно реализовать схему, эквивалентную десяткам логических элементов, при этом изменяется подход и даже идеология процесса проектирования. Эти средства не только повышают эффективность производства, но также освобождают человека от работы по контролю состояния технологического процесса и формированию управляющих воздействий на исполнительные органы рабочих машин. Применение программируемых реле в системах управления настолько повысило их потенциальные возможности, что в настоящее время их считают обладающими элементами интеллекта человека.

Современные автоматические системы управления производством разделяются на три уровня:

1. Верхний уровень – управление технологическим процессом из панели оператора, реализуемой на SCADA платформе.
2. Средний уровень – программно-электрическая часть, реализуемая с помощью программируемых реле и контроллеров с имеющимися информационными цифровыми или аналоговыми входами и выходами.
3. Низкий уровень – физические модели, исполнительные органы, которые управляются от элементов среднего уровня.

В совокупности все три уровня автоматизации представляют собой интеллектуальную систему управления технологическими процессами. В данном учебном пособии рассматривается реализация низкого и среднего уровней на базе программного обеспечения и оборудования производства Eaton (Moeller).

В число программного обеспечения, используемого в учебном пособии, входят такие продукты, как EasySoft v6.20 Pro и CodeSys v2.3.9 SP4, имеющиеся в распоряжении отделения электроэнергетики и электротехники Национального исследовательского Томского политехнического университета.

1. ОПИСАНИЕ УЧЕБНОГО СТЕНДА

На рис. 1 показан внешний вид учебного стенда для изучения систем автоматики на базе программируемых электронных аппаратов, таких как реле, контроллеры, HMI панель, и другие компоненты.

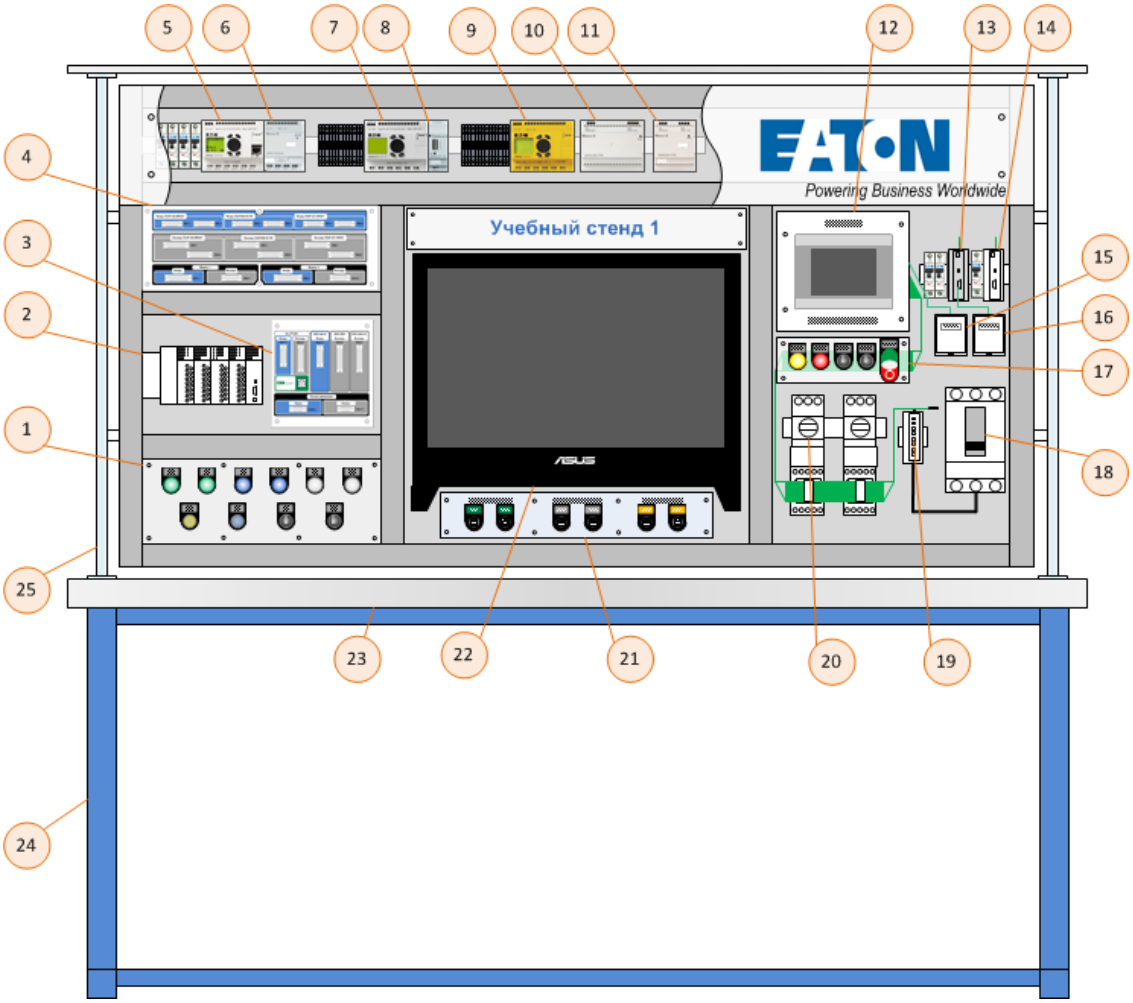

Рис. 1. Внешний вид учебного стенда:
1 – кнопочный пост; 2 – модульный ПЛК XC-201 с набором модулей; 3 – интерфейсная панель разъёмов 2; 4 – интерфейсная панель разъёмов 1; 5 – ПЛК EC4P-222-MRAD1; 6 – модуль расширения EASY411-DC-ME; 7 – программируемое реле Easy 820-DC-RC; 8 – модуль Ethernet EASY209-SE; 9 – реле безопасности Easy-Safety ES4P-221-DRXD1; 10 – блок питания Easy 600 POW; 11 – блок питания Easy 400 POW; 12 – HMI Панель XV-102-E6-57TVRC-10; 13 – концетратор EU5C-SWD-CAN; 14 – концентратор easy800 with SmartWire-DT EASY806-DC-SWD; 15 – розетка подключения к CANopen HMI Панели; 16 – розетка подключения к CANopen концетратора EU5C-SWD-CAN; 17 – SWD Кнопочный пост; 18 – автомат NZMN2-ME90; 19 – SWD-модуль NZM-XSWD-704 соединения для NZM; 20 – пусковая сборка MSC-DEA-12-M7; 21 – панель сетевых коммуникаций; 22 – персональный компьютер; 23 – стол; 24 – стойки; 25 – система подвески оборудования

На рис. 2–3 показаны интерфейсные панели разъёмов для электрического соединения:

1) кнопок управления с входами аппаратов (реле, контроллеры);
2) сигнальных ламп с выходами аппаратов;
3) датчиков физических моделей с входами аппаратов;
4) исполнительных элементов физических моделей (контакторы пуска двигателя, соленоиды и т. д.) с выходами аппаратов.

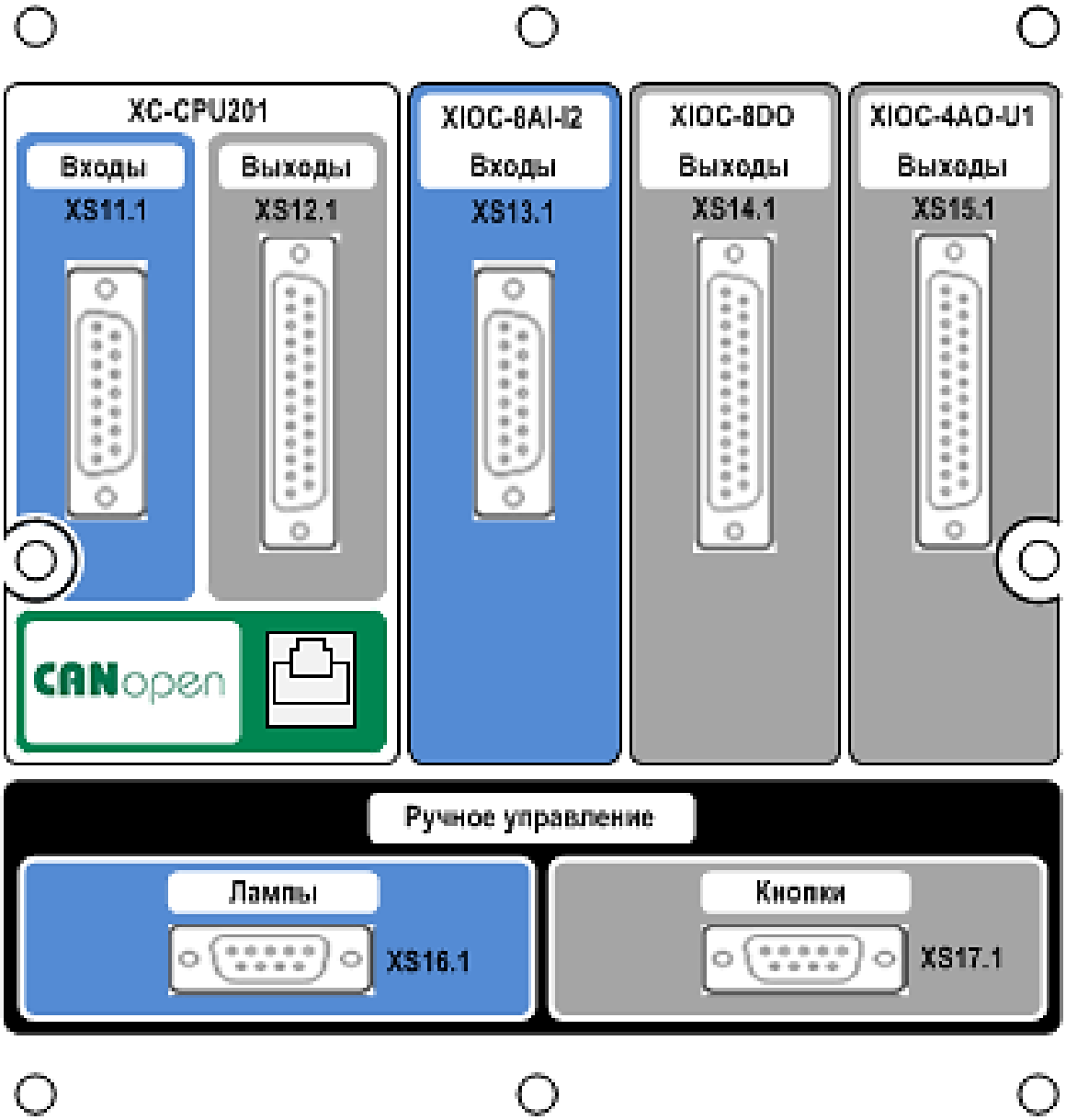

Рис. 2. Интерфейсная панель разъёмов 2

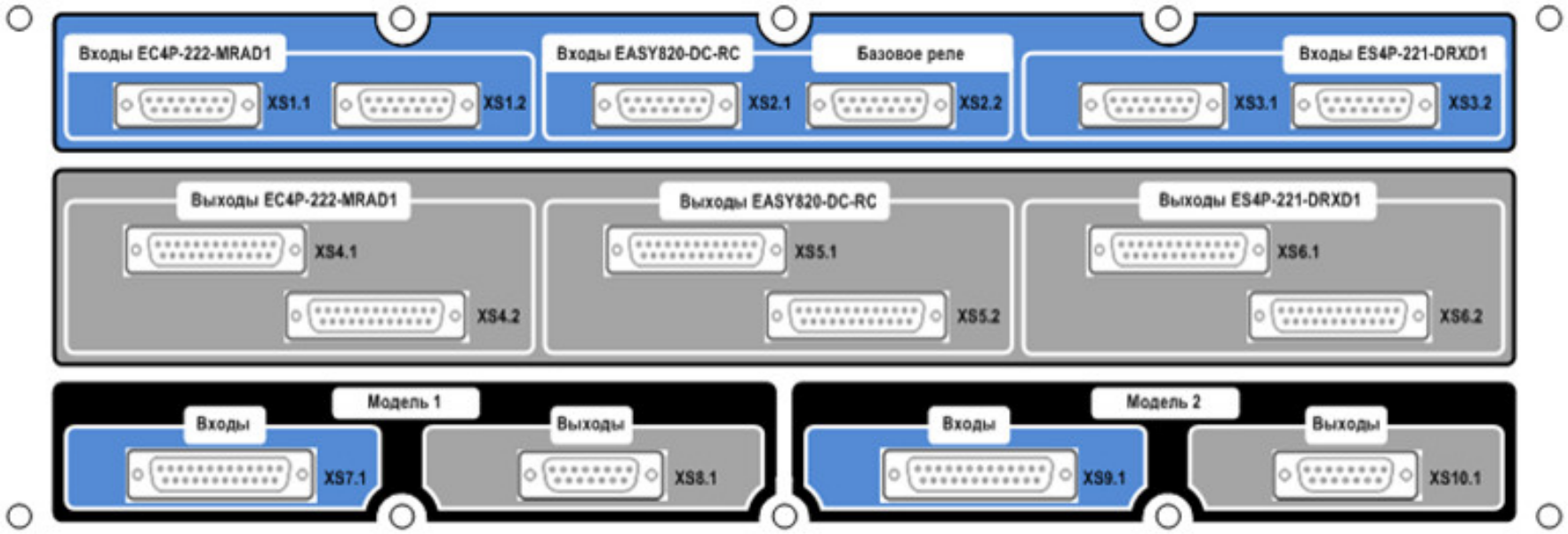

Рис. 3. Интерфейсная панель разъёмов 1

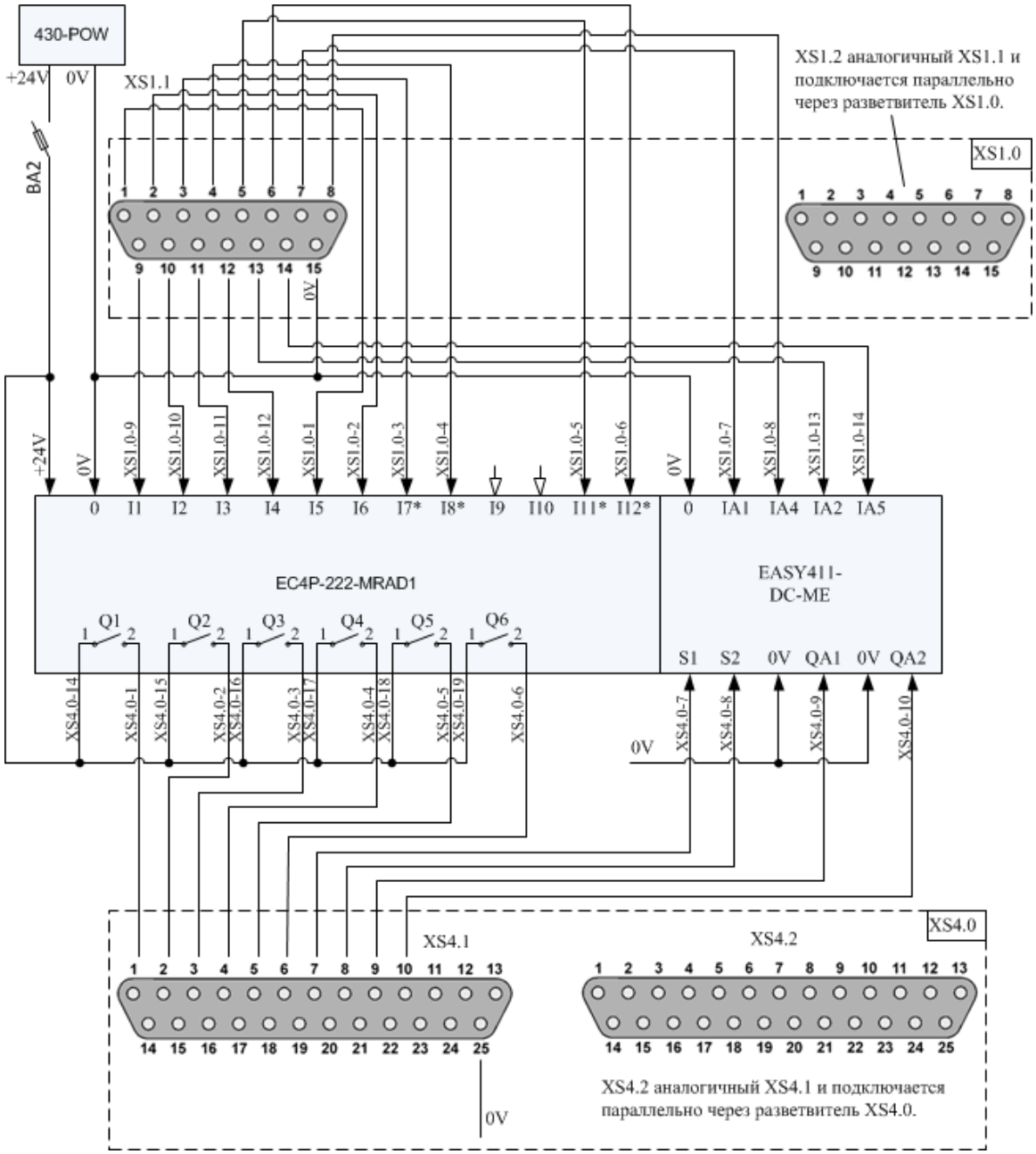

Рис. 4. Подключение контроллера EC4P-222-MRAD1 к панели разъёмов 1

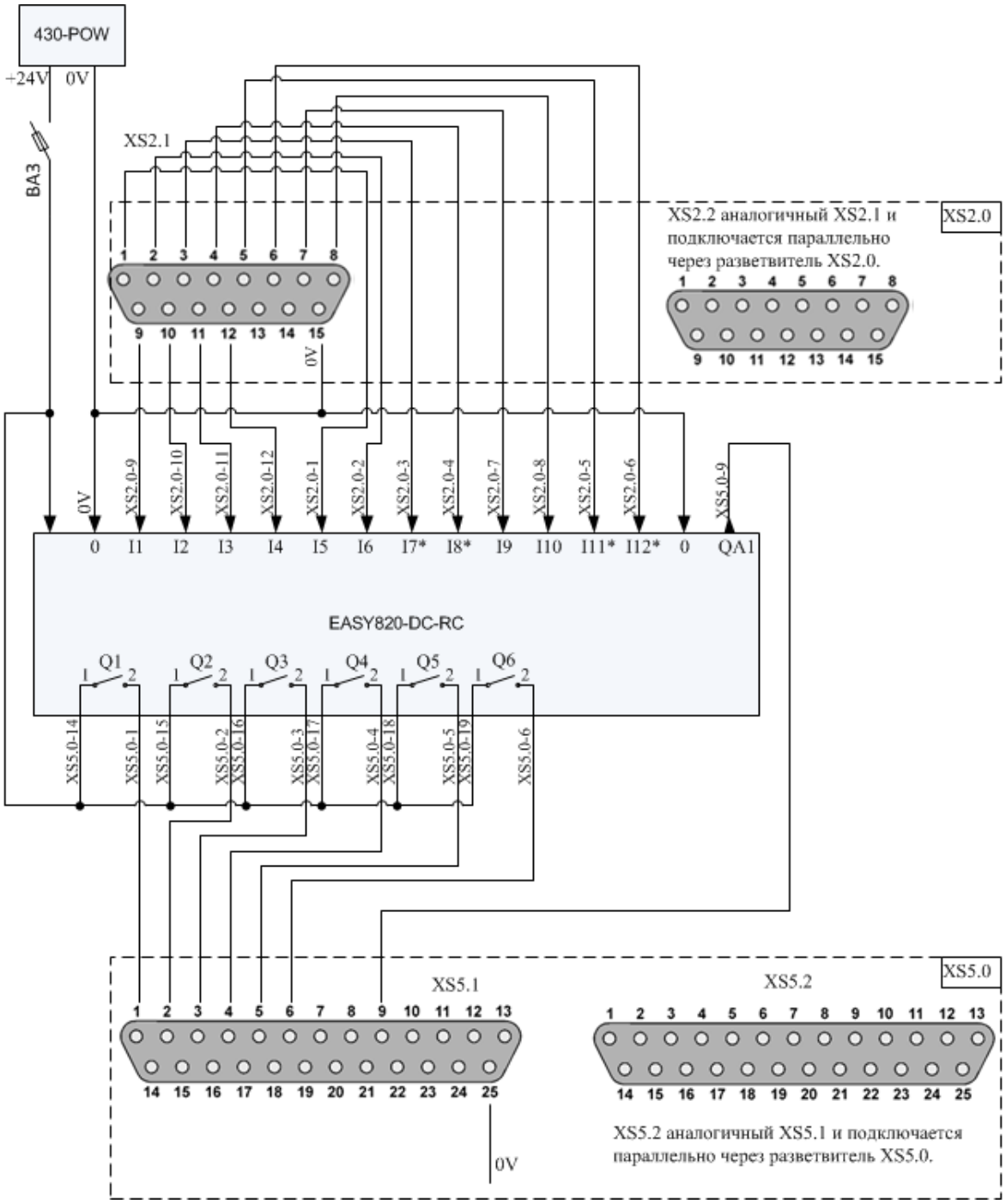

Рис. 5. Подключение реле EASY820-DC-RC к панели разъёмов 1

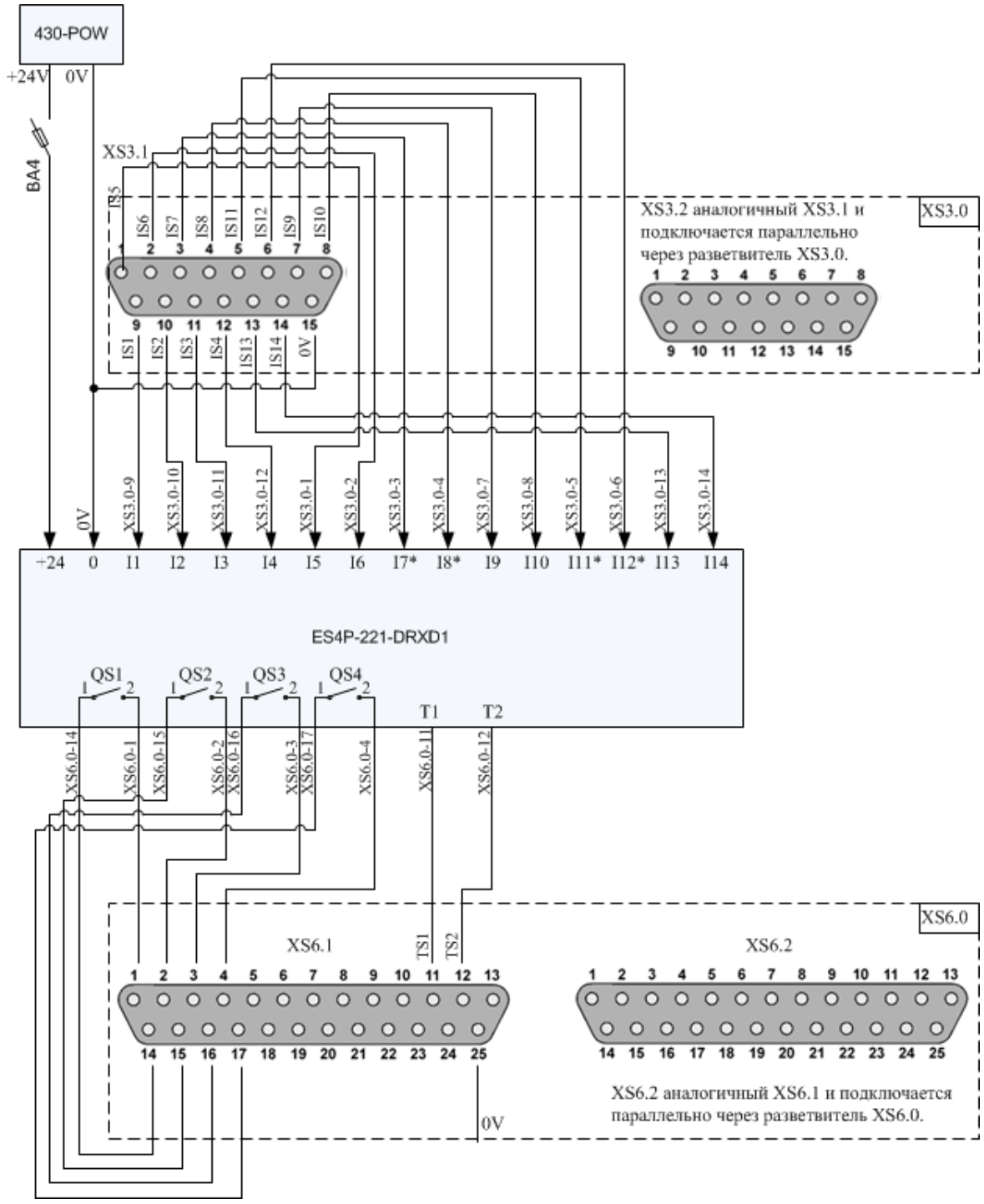

Рис. 6. Подключение реле ES4P-221-DRXD1 к панели разъёмов 1

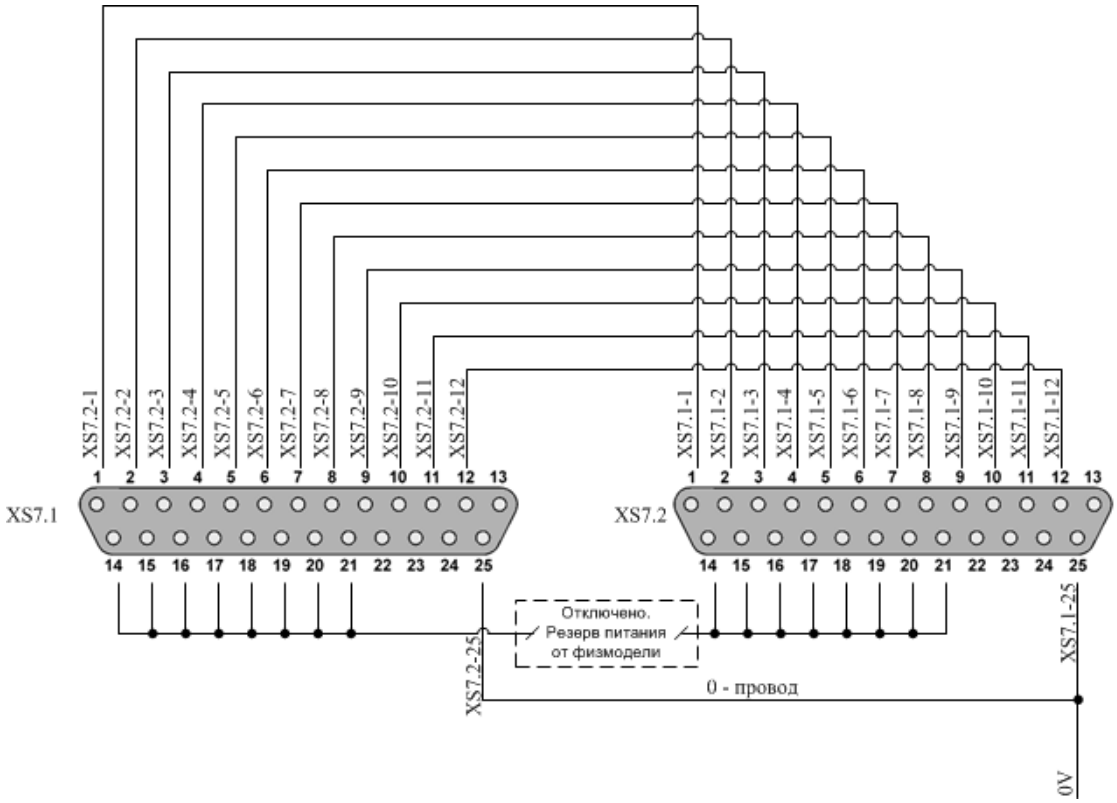

Рис. 7. Входы физической модели

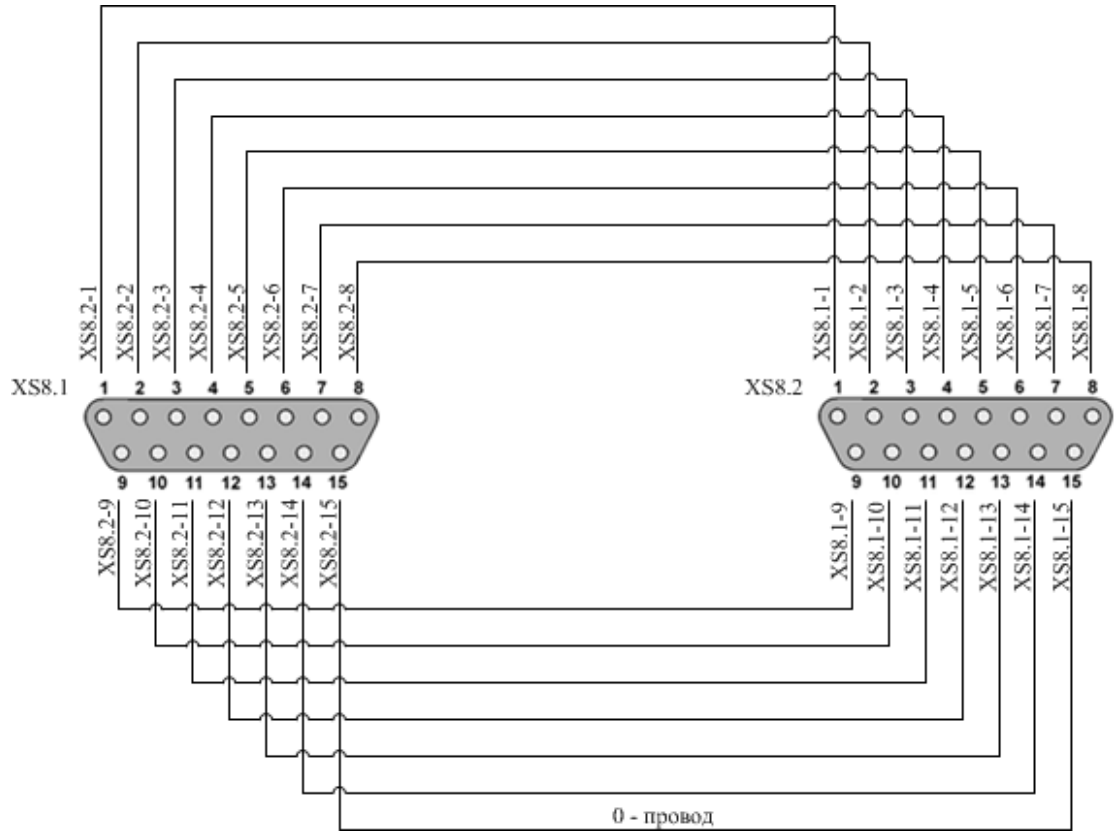

Рис. 8. Выходы физической модели

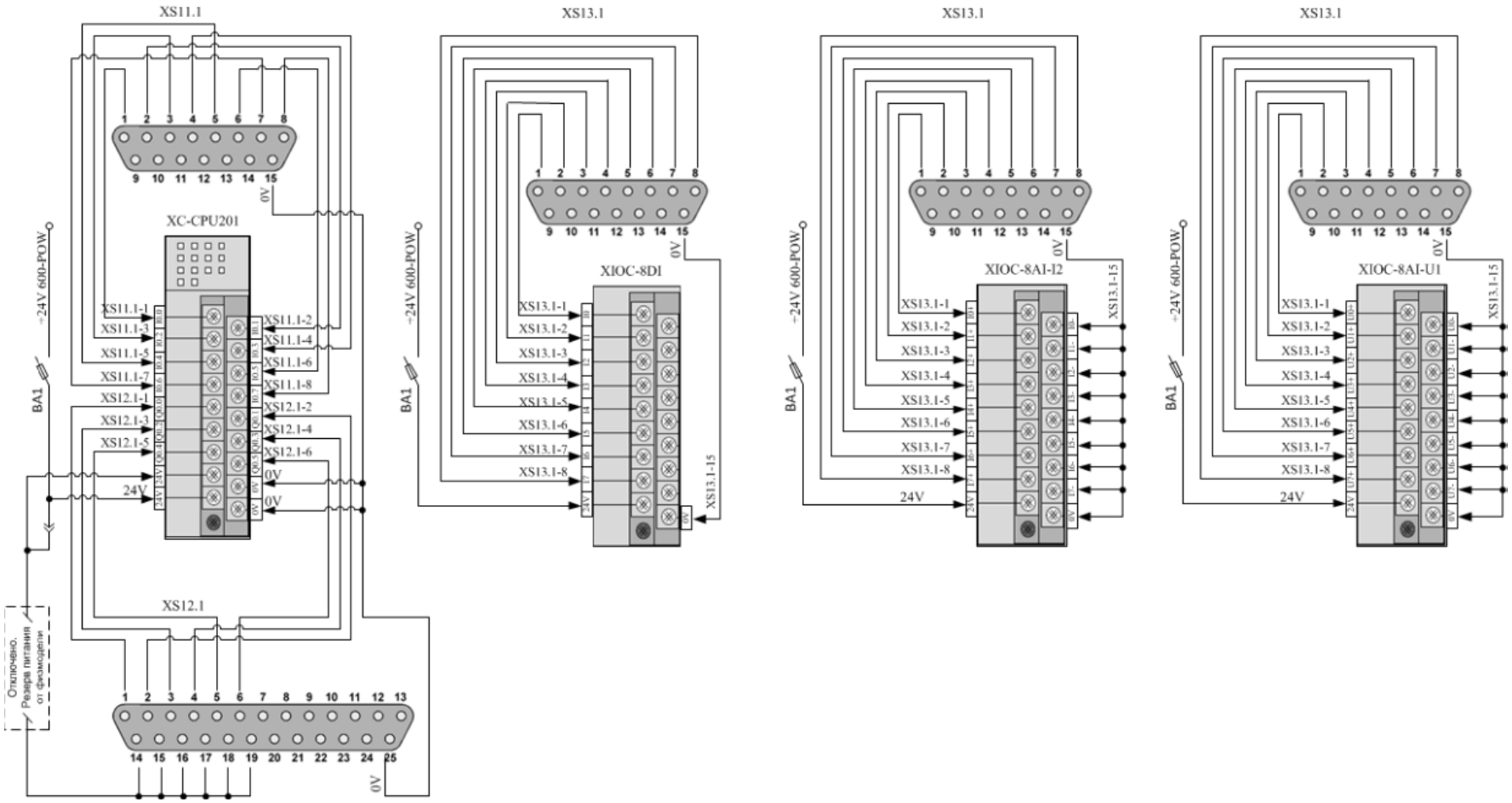

Рис. 9. Подключение модулей контроллера XC-CPU201 к панели разъёмов 2

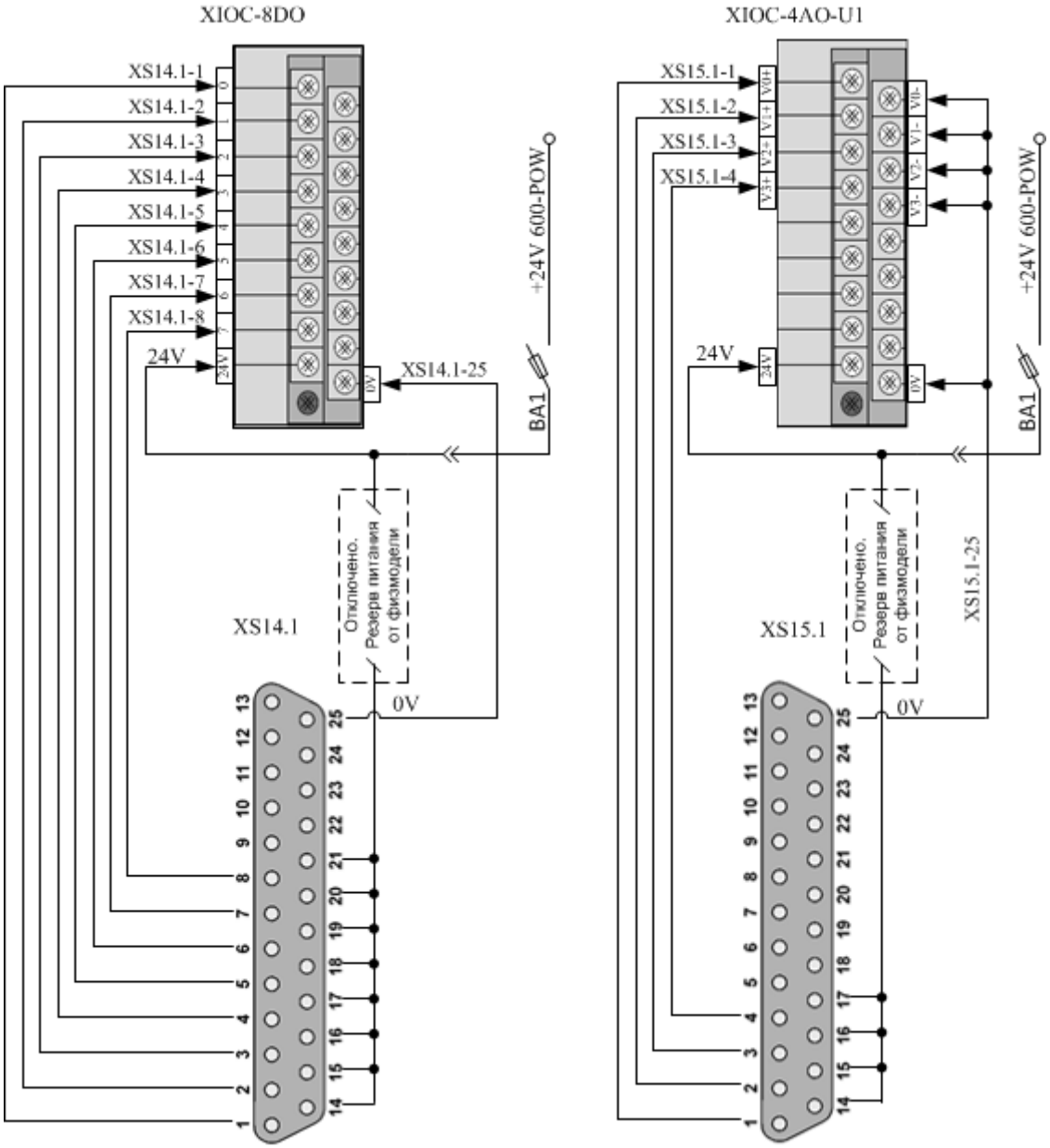

Рис. 10. Подключение выходов модулей контроллера XC-CPU201 к панели разъёмов 2

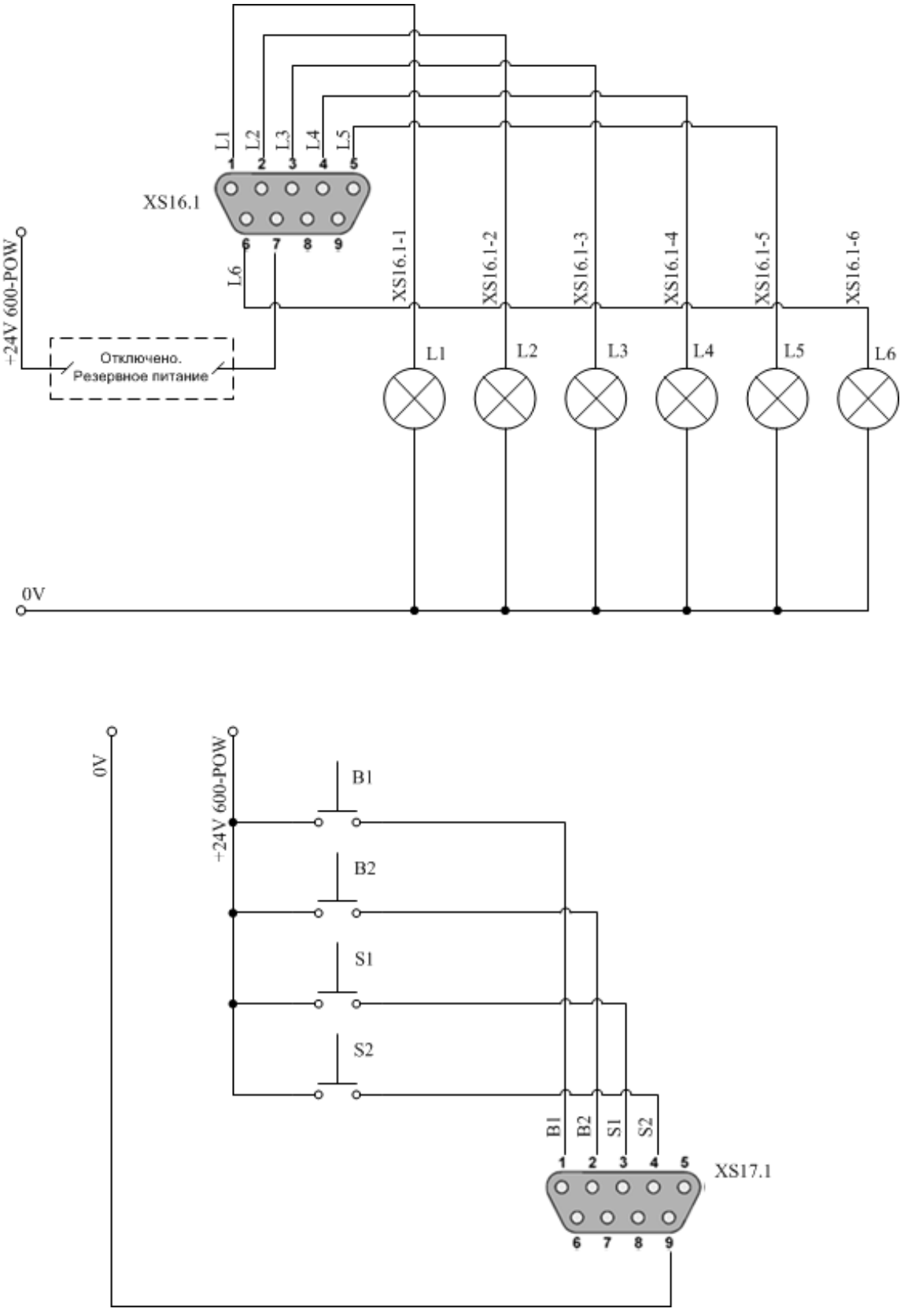

Рис. 11. Элементы ручного управления панели разъёмов 2

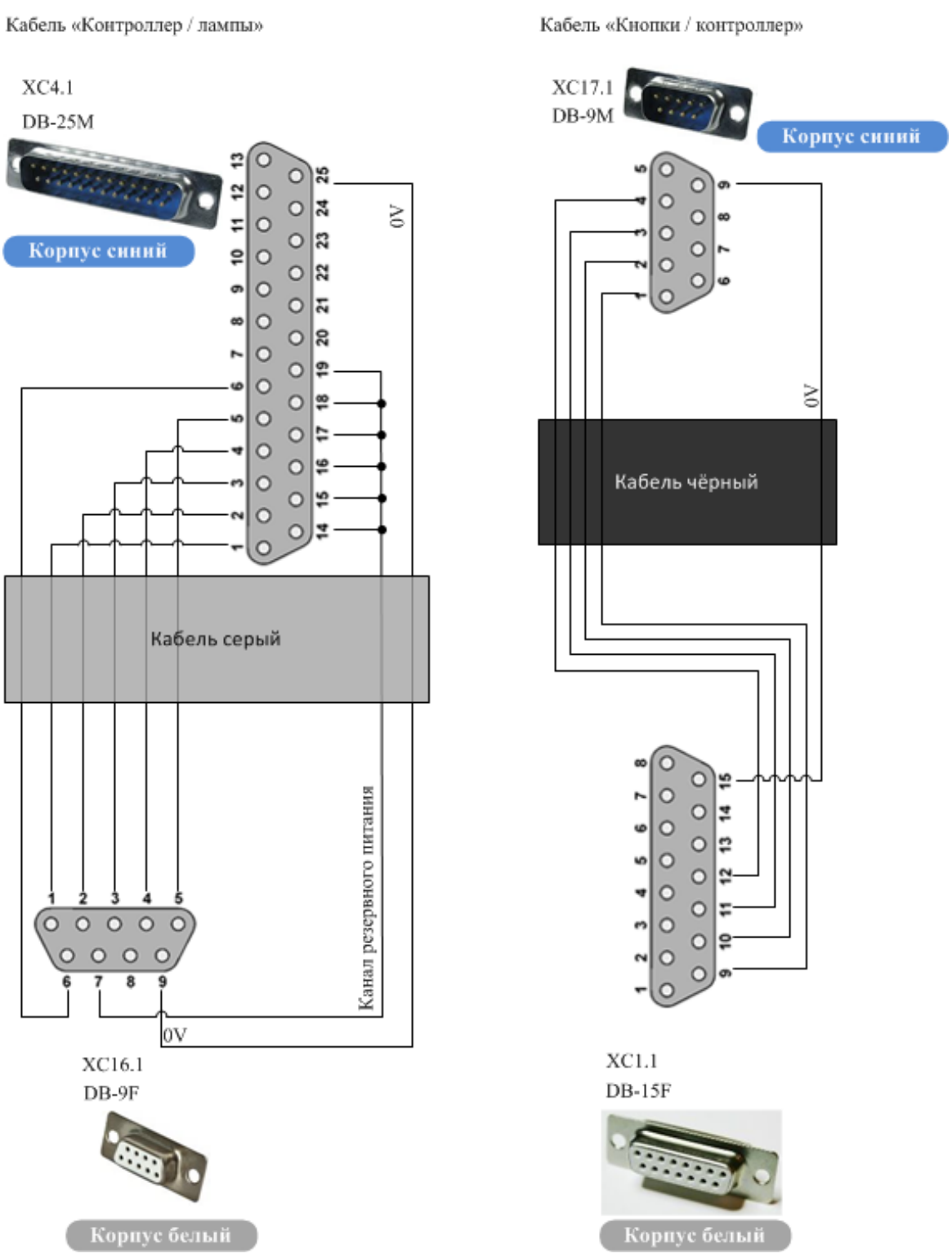

Рис. 12. Кабели ручного управления

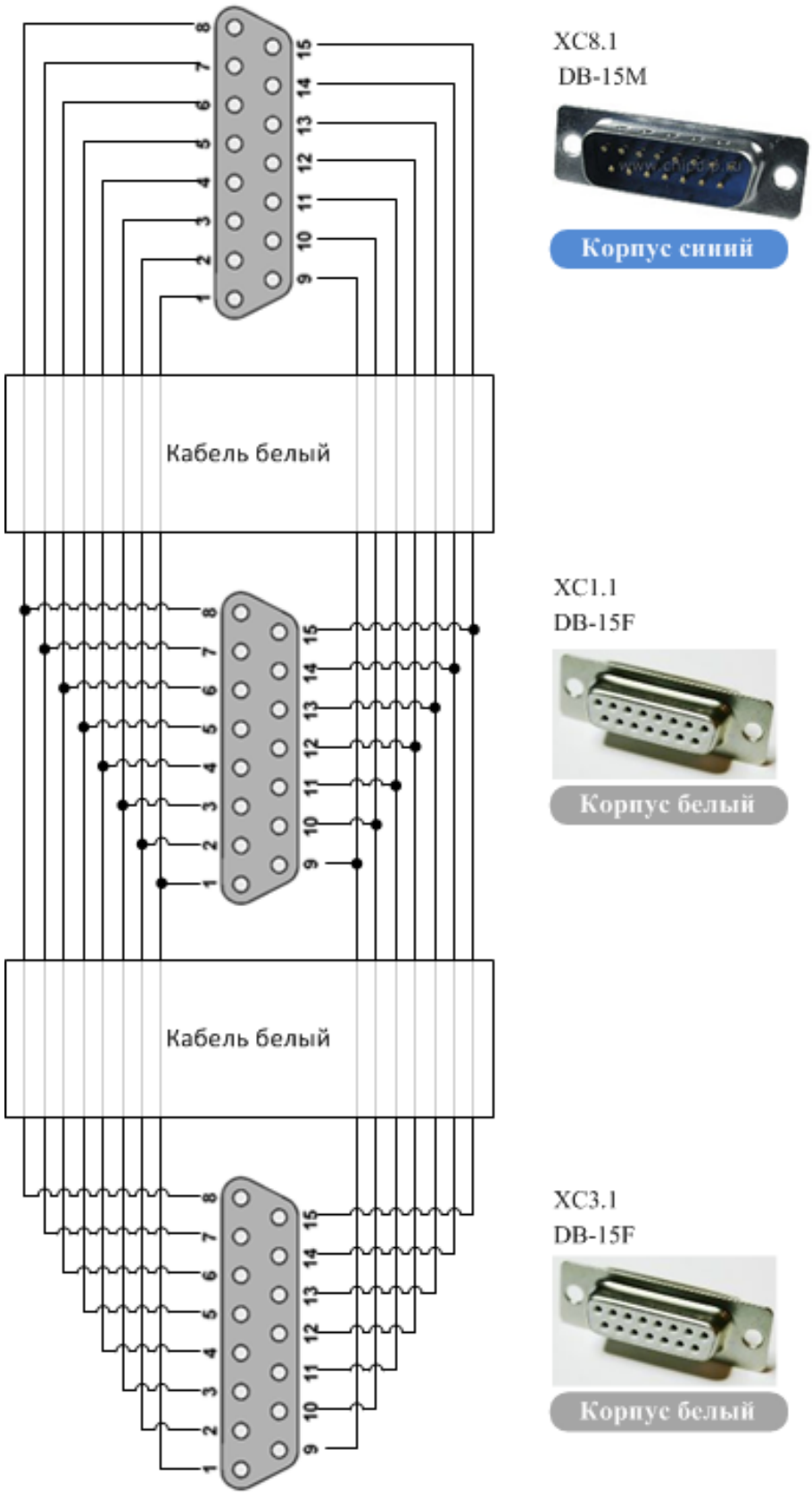

Рис. 13. Кабель «Физическая модель / Вход-контроллер»

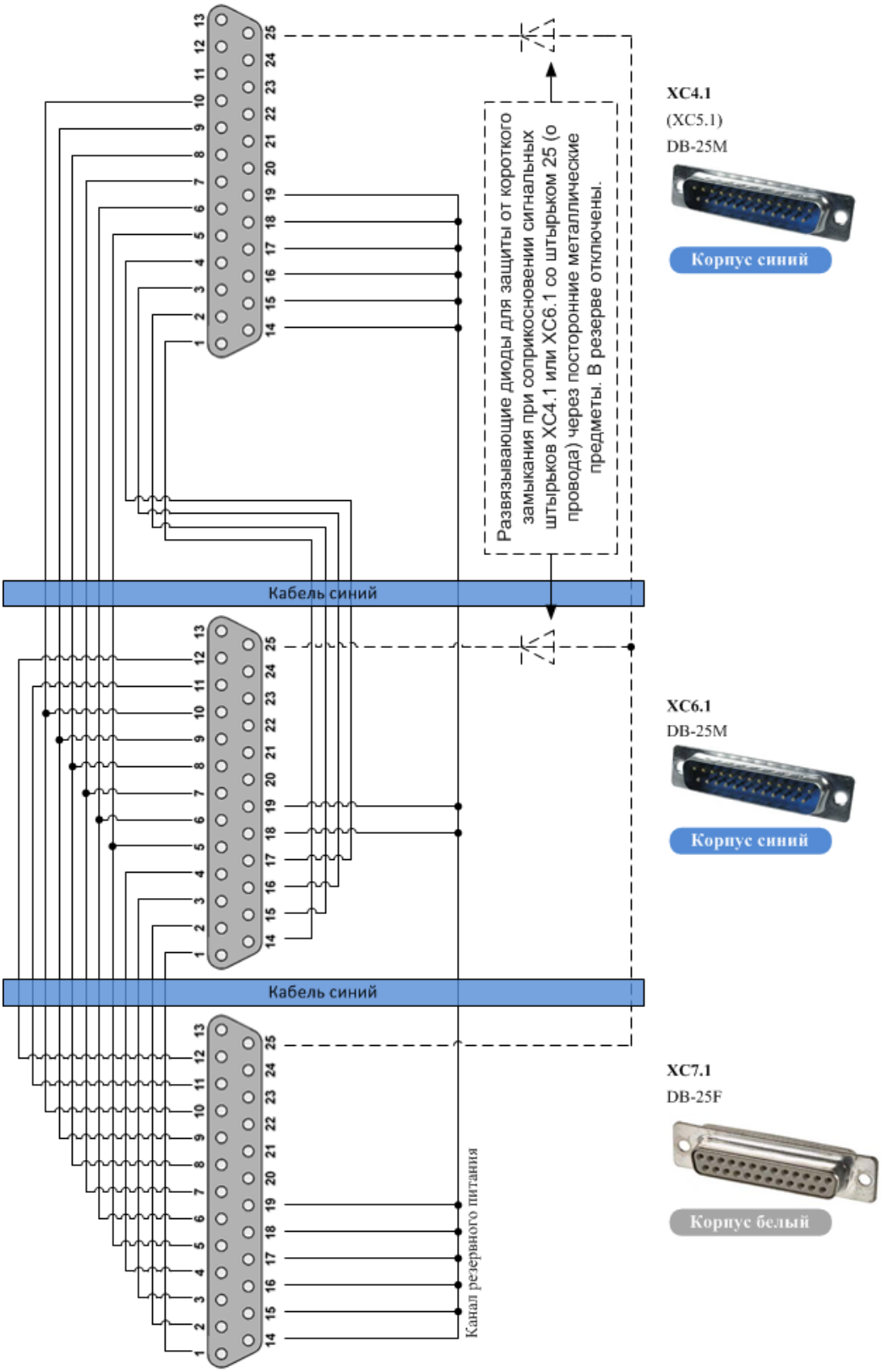

Рис. 14. Кабель «Выход-контроллер / Физическая модель»

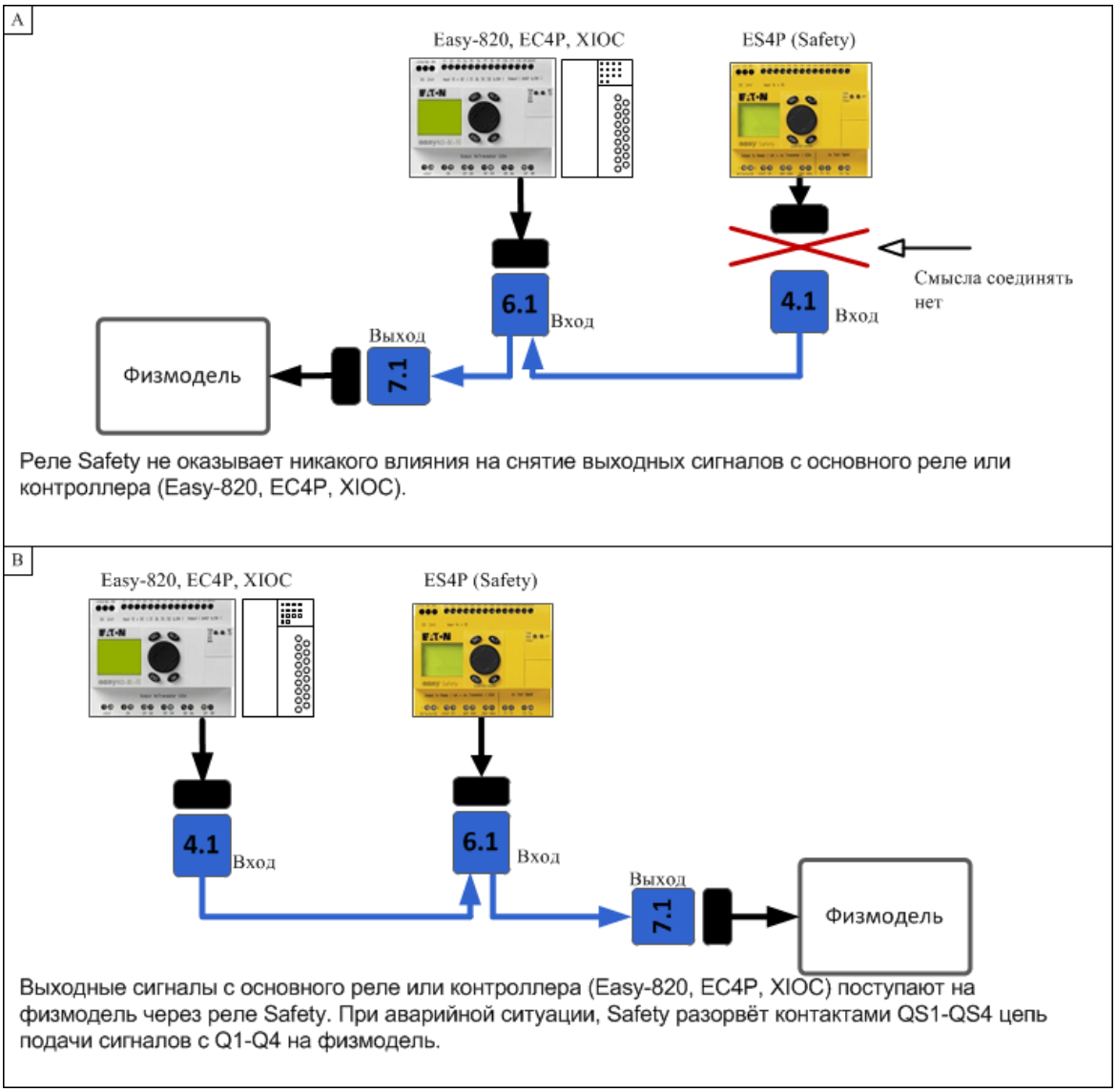

Рис. 15. Варианты подключения кабеля «Выход-контроллер / Физическая-модель»

2. ПРОСТАЯ ПРОГРАММА ДВИЖЕНИЯ МЕХАНИЗМА

Цель: познакомиться с основой программирования контроллера EC4P и визуализацией программного алгоритма.

2.1. Порядок работы

2.1.1. Ознакомиться с программной средой программирования CoDeSys.

2.1.2. Написать программный алгоритм движения механизма.

2.1.3. Настроить визуализацию программного алгоритма.

2.1.4. Апробация созданной программы.

2.2. Создание программного алгоритма

Задача программы – задание траектории движения геометрической фигуры. Лабораторная работа разработана на базе [7.4].

Этапы создания программы:

2.2.1. Запуск программной среды CoDeSys. Запустите программу в Windows через Пуск/Программы/Moeller Software/easy Soft CoDesys V2.3.9.

2.2.2. Создайте новый проект командой File/New.

2.2.3. Настройка целевой платформы (Target Settings). Проект является машинно-независимым, поэтому его можно опробовать в режиме эмуляции. Но для определённости лучше выбрать определённый контроллер. На странице диалогового окна «Configuration» установите 3S CoDeSys SP PLCWinNT V2.4 и подтвердите ввод – OK.

2.2.4. Главная программа PLC_PRG POU. Следующее диалоговое окно определяет тип первого программного компонента (New POU). Выберите язык реализации (language of POU) SFC и сохраните предложенные по умолчанию тип компонента – программа (Type of the POU Program) и имя – Name PLC_PRG. PLC_PRG – это особый программный компонент (POU). В однозадачных проектах он циклически вызывается системой исполнения.

По умолчанию в окне созданной SFC программы имеется пустая диаграмма, содержащая начальный шаг «Init» и соответствующий переход «Trans0», заканчивающийся возвратом к Init (рис. 16).

2.2.5. Последовательность работы механизма. Каждой фазе работы соответствует определённый этап (шаг). Выделите переход (Trans0) так, чтобы он оказался окружён пунктирной рамкой. В контекстном меню дайте команду вставки шага и перехода под выделенным: Step-Transition (after). Аналогично повторите попытку ещё 4 раза. Включая Init, должно получиться 6 шагов с переходами. Щёлкая мышью по именам переходов и шагов, вы заметите, что они выделяются цветом. Таким способом вы можете определить новые наименования.

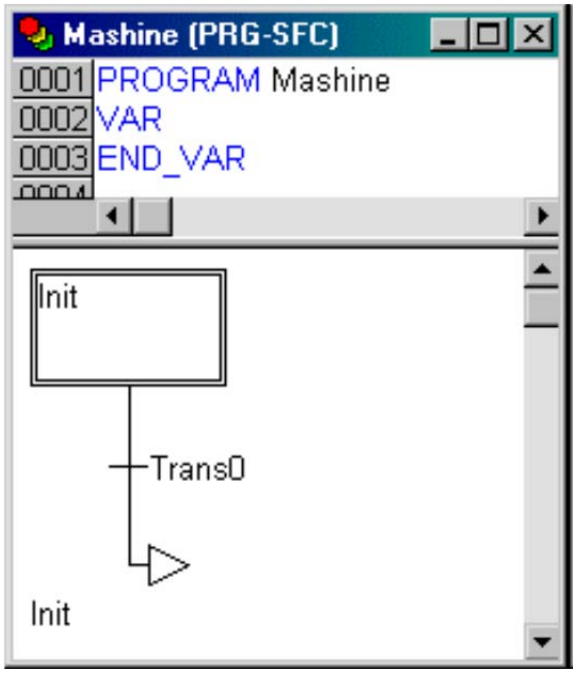

Рис. 16. Исходный вид диаграммы SFC

Первый после Init шаг должен назваться Go_Right. Под ним Go_Down, Go_Left, Go_Up и Count.

2.2.6. Программирование первого шага. Щёлкните дважды на шаге Go_Right. CoDeSys начнёт определение действия шага и попросит выбрать язык его реализации (Language). Выберите ST (structured text) и перейдите в автоматически открытое окно текстового редактора. В этом шаге рабочий орган нашего механизма должен перемещаться по оси X вправо. Программа должна выглядеть так:

X_pos := X_pos + 2;

Завершите ввод клавишей Return, и определите переменную X_pos типа INT (целое). Теперь верхний уголок шага должен быть закрашен. Это признак того, что действие этого шага определено.

2.2.7. Программирование следующих шагов. Повторите описанную последовательность для всех оставшихся шагов. Переменные Y_pos и Counter должны быть типа INT.

Шаг Go_Down – программа: Y_pos := Y_pos + 2;

Шаг Go_Left – программа: X_pos := X_pos – 2;

Шаг Go_Up – программа: Y_pos := Y_pos – 2;

Шаг Count – программа: Counter := Counter + 1;

2.2.8. Определение переходов. Переход должен содержать условие, разрешающее переключение на следующий шаг. Переход после шага Init назовите Start и определите новую логическую переменную Start типа BOOL. При единичном значении этой переменной начинается цикл работы механизма.

Следующий переход должен содержать условие X_pos = 180, так при значении позиции X включается следующая фаза движения.

Условие третьего шага Y_pos = 180,

четвёртого X_pos = 0,

пятого Y_pos = 0 и

шестого TRUE (переход разрешён сразу же, после однократного выполнения).

В результате должна получиться программа, имеющая вид как на рис. 17, где справа открыты действия каждого шага, реализованные на языке ST.

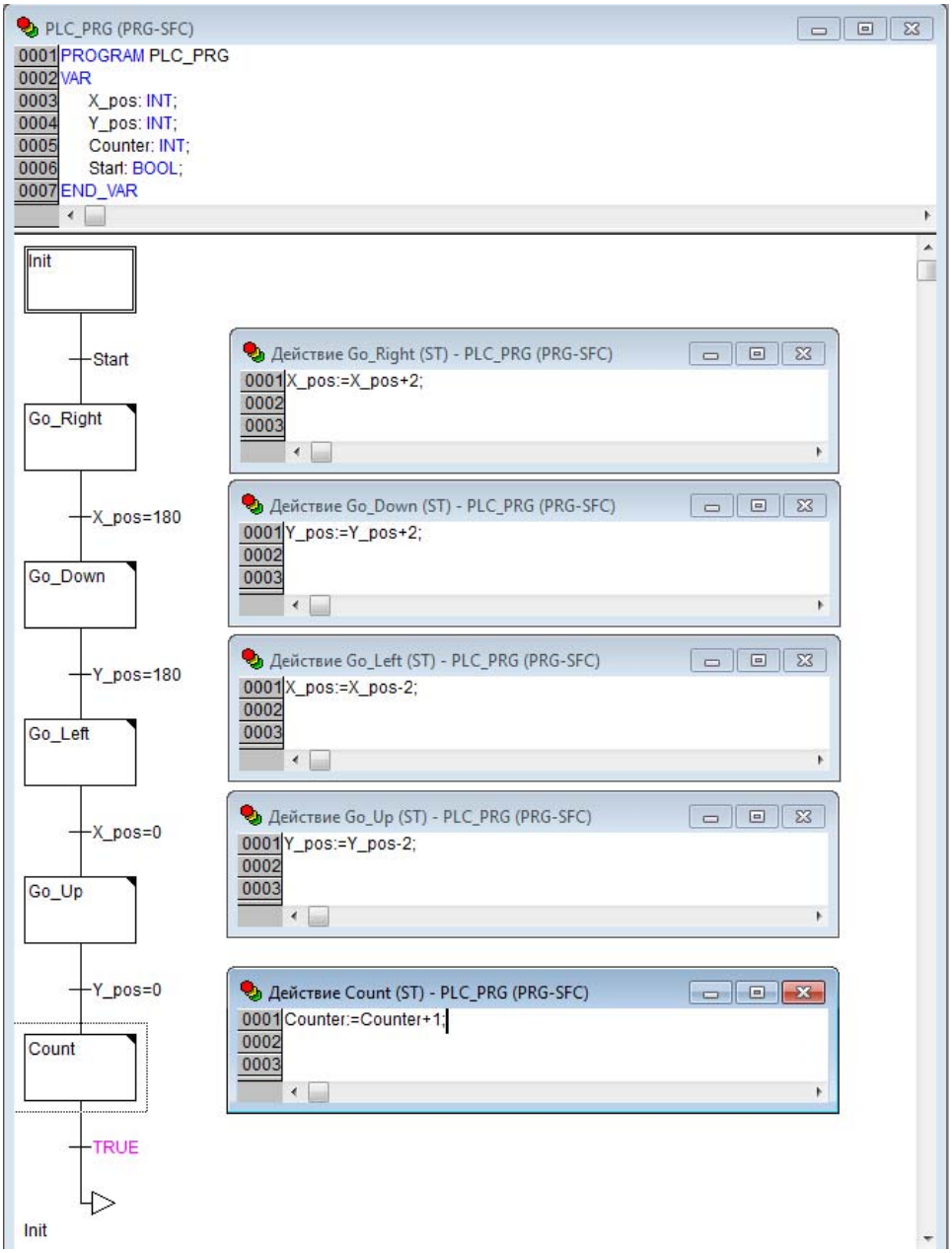

Рис. 17. Программный алгоритм

2.2.9 Компиляция проекта. Откомпилируйте проект целиком командой меню Project/Rebuild all, либо клавишей F11. Если вы все сделали верно, то в нижней части окна должно появиться сообщение «0 errors» (0 ошибок). В противном случае необходимо исправить допущенные ошибки. В этом помогут развёрнутые сообщения об ошибках.

2.3. Визуализация

2.3.1. Создание визуализации. Третья страничка организатора объектов CoDeSys называется визуализация (Visualization). Перейдите на страничку визуализации. В контекстном меню введите команду добавления объекта Add object. Присвойте новому объекту PLC_VISU.

2.3.2. Рисуем элементы визуализации. На панели инструментов выберите прямоугольник (Rectangle). В окне редактора нажмите левую клавишу мыши и растяните прямоугольник до нужно ширины и высоты, отпустите клавишу. Таким образом, нарисуйте вертикальный прямоугольник «общую рамку». В этой рамке далее нарисуйте следующие элементы:

- Клавишу «Пуск», имеющую следующие настройки:

– Кликните два раза левой клавишей мыши по элементу, вызовите диалоговое окно и в окошке «содержимое» (Contents) категории «текст» (Text Category) слово ОК.

– В категории переменных (Variables Category) щёлкните мышью в поле «изменение цвета» (Change Color) и воспользуйтесь ассистентом ввода «F2». Вставьте переменную «Start», это будет выглядеть так – PLC_PRG.Start, т. е. используется переменная «Start», которая относится к PLC_PRG. Эта переменная будет менять цвет данного элемента визуализации (клавиша «Пуск»).

– В категории цвета (Colors) задайте цвет закраски элемента (Inside), например, красный. Для возбуждённого состояния необходимо определить другой цвет (Alarm color), например, зелёный.

– В категории ввода (Input Category) необходимо ещё раз ввести переменную PLC_PRG.Start и поставить флажок «Кнопка-переключатель» (Toggle variable).

В итоге, прямоугольник будет отображаться красным при значении переменной Start равной FALSE и зелёным, при значении TRUE. Её значение будет изменяться при каждом нажатии нашей клавиши.

- Прямоугольник для счётчика циклов с настройками:

– Text Category, Contents текст «Счётчик: %s» (%s – заместитель для отображения значения переменной).

– Variable Category, Textdisplay переменная PLC_PRG.Counter.

- Квадрат – область движения рабочего инструмента механизма с настройками:
  - Colors Category, Color закраска Inside жёлтым цветом.
- Маленький квадрат, обозначающий рабочий инструмент механизма с настройками:
  - Absolute movement Category, X-Offset переменная PLC_PRG.X_pos.
  - Absolute movement Category, Y-Offset переменная PLC_PRG.Y_pos.
  - Colors Category, Color закраска Inside синим цветом.
- Настройка слоёв размещения элементов. Если какие-то элементы перекрывают остальные не нужным образом, то необходимо поместить их на задний план, щёлкнув правой клавишей мыши и выбрав команду «на задний план» (Send to back).

В итоге должно получиться окно как на рис. 18.

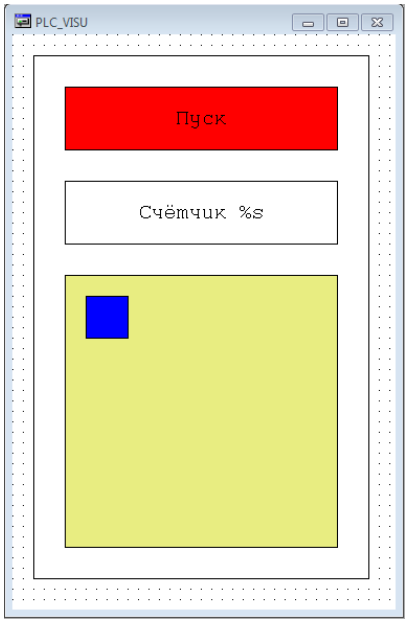

Рис. 18. Окно визуализации

В итоге должно получиться окно как на рис. 18.

2.4. Запуск целевой системы

Откомпилируйте проект, вкладка «Проект», команда «Компилировать» (Project/Build). Предварительно поставьте во вкладке «Онлайн» галочку «режим симуляции» (Online/Simulation Mode). Здесь же установите соединение с контроллером командой «Онлайн» (Online) и запустите проект командой «старт» (Run).

При нажатии на кнопку «Пуск» в окне визуализации, программа начнёт свою работу и элемент рабочего механизма (синий квадратик) начнёт своё движение. При повторном нажатии на кнопку «Пуск», по завершении текущего цикла, движение элемента остановится.

3. ПРОГРАММА УПРАВЛЕНИЯ УЛИЧНЫМ СВЕТОФОРОМ

Цель: освоить принципы построения объединённой интеллектуальной системы управления на разных языковых реализациях.

3.1. Порядок работы

3.1.1. Ознакомиться с различными языковыми реализациями программной среды CoDeSys (языки CFC, FBD, SFC, ST).

3.1.2. Создать систему на базе программных языков CFC, FBD, SFC, ST.

3.1.3. Настроить визуализацию системы.

3.1.4. Апробация созданной системы.

3.2 Создание системы

Задача системы – управление уличным светофором для двух взаимно-перпендикулярных направлений движения. Лабораторная работа разработана на базе [7.2, 7.3].

Этапы создания системы:

3.2.1. Запуск программной среды CoDeSys. Запустите программу в Windows через Пуск/Программы/Moeller Software/easy Soft CoDesys V2.3.9.

3.2.2. Создайте новый проект командой File/New.

3.2.3. Настройка целевой платформы (Target Settings). На странице диалогового окна «Configuration» выберите тип контроллера EC4P-200 V2.3.9 и подтвердите ввод – OK.

3.2.4. Главная программа PLC_PRG POU. Следующее диалоговое окно определяет тип первого программного компонента (New POU). Выберите язык реализации (language of POU) CFC и сохраните предложенные по умолчанию тип компонента – программа (Type of the POU Program) и имя – Name PLC_PRG. PLC_PRG – это особый программный компонент (POU). В однозадачных проектах он циклически вызывается системой исполнения.

Назначение PLC_PRG – в этой программе вводится входной сигнал включения, разрешающий начало работы светофора и «цветовые команды» каждой лампы связаны с соответствующими выходами аппаратуры.

3.2.5. Создание дополнительных программных и функциональных блоков.

- Командой «Проект/Добавить объект» (Project/Object Add) добавьте новый объект (делается это в первой системной вкладке, где список POU). Объект должен быть программой (в появившемся диалоговом меню выбрать тип POU – «Program») на языке Sequential Function Chart (SFC) с именем SEQUENCE. Назначение «SEQUENCE» – назначать стадию работы для каждого из светофоров через заданный промежуток.

- Далее добавьте новый объект – функциональный блок на языке Function Block Diagram (FBD) с именем TRAFFICSIGNAL. Назначение «TRAFFICSIGNAL» – сопоставить определённые стадии работы светофора соответствующим цветам.
- Добавьте функциональный блок на языке FBD с именем WAIT. Назначение WAIT – таймер, который на вход получает длину стадии в миллисекундах и на выходе выдаёт состояние «ИСТИНА» (TRUE) по истечении заданного периода времени.

3.2.6. Программирование TRAFFICSIGNAL. В редакторе объявлений определите входную переменную (между ключевыми словами VAR_INPUT и END_VAR) по имени STATUS типа INT. STATUS будет иметь четыре возможных состояния, определяющие соответствующие стадии – зелёная, жёлтая, жёлто-красная и красная. Определите ещё три выходных переменных RED, YELLOW и GREEN. Раздел объявления будет выглядеть как на рис. 19.

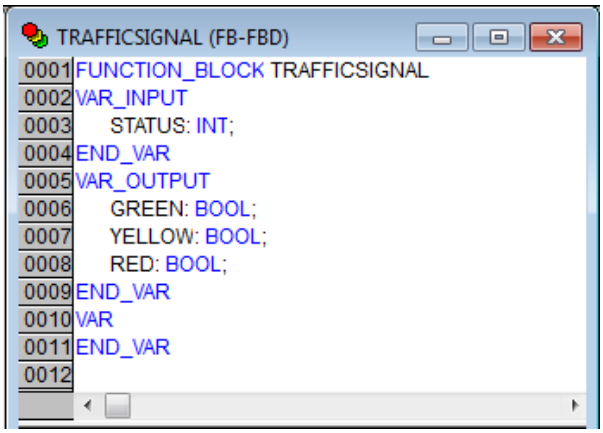

Рис. 19. Объявления блока TRAFFICSIGNAL

Опишем, что связывает вход STATUS с выходными переменными. Для этого перейдите в раздел кода POU (Body). Щёлкните на поле слева от первой цепи (серая область с номером 1). Выберите команду «Вставить оператор» (Insert/Operator). В первой цепи будет прямоугольник с оператором AND и двумя входами. Щёлкните мышкой на тексте AND и замените его на EQ. Три знака вопроса на верхнем входе замените на имя переменной STATUS. Для нижнего входа поставить 1.Щёлкните на месте позади блока EQ и вставьте «присвоение» (Assignment). Измените три вопроса после присвоения на GREEN. Таким образом, STATUS будет сравниваться с 1, результат присваивается GREEN, который будет включён, когда STATUS равен 1.

Для других цветов TRAFFICSIGNAL вставьте ещё две цепи. Сделать это нужно с помощью команды «вставка новой цепи» (Insert/Network(after)). Законченный POU TRAFFICSIGNAL должен вы-

глядеть как на рис. 20. При этом, чтобы вставить оператор перед входом другого оператора, нужно выделить сам вход, а не текст, которые выделяется прямоугольником. Далее использовать команду «Вставка оператора». Если на платформе CoDeSys не установлена стандартная библиотека standart.lib, то её необходимо установить следующим образом. Откройте менеджер библиотек командами «Window»/«Library Manager», выберите «Insert»/«Additional library». Должно открыться диалоговое окно выбора файлов. Выберите standart.lib из списка файлов.

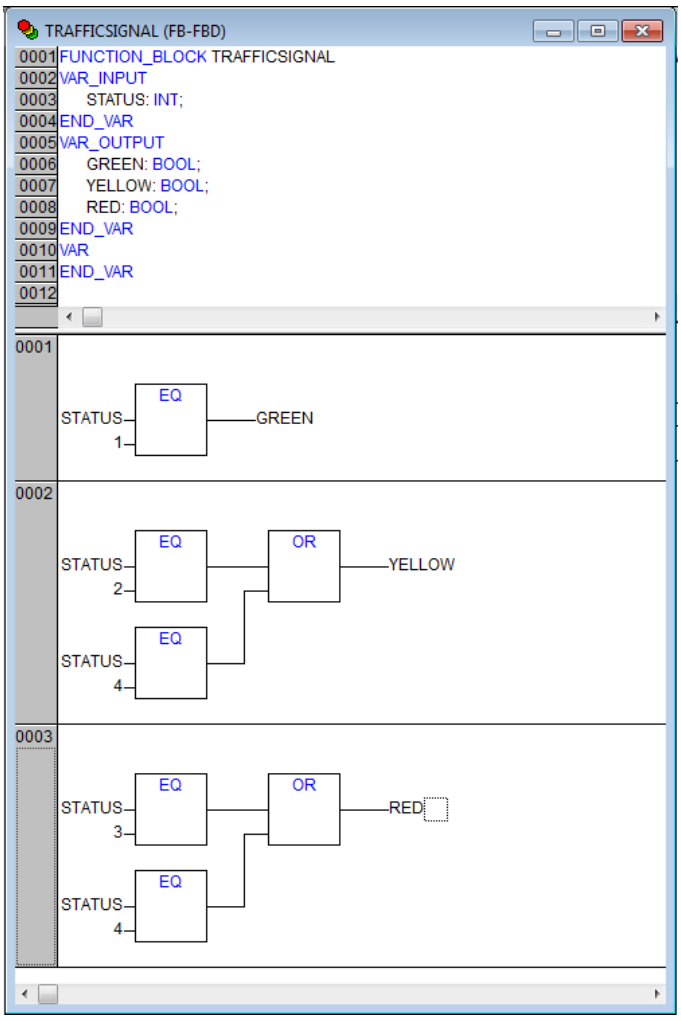

Рис. 20. Функциональный блок TRAFFICSIGNAL

3.2.7. Программирование WAIT. Сделайте объявления переменных как на рис. 21. Поставьте оператор AND и на верхний вход подайте «run» – это запускающий вход, при значении которого TRUE, начнётся отсчёт времени оператором TON. Оператор «TON» – это таймер, который по сигналу TRUE на входе IN начинает отсчёт заданного на входе PT времени. ПО истечении заданного времени, выход Q выдаёт сигнал TRUE. На нижний вход оператора AND через инвертор (маленькая

окружность на входе, которая ставится командой «вставить инверсию») подайте переменную OK.

Суть блока WAIT заключается в следующем. В исходном состоянии на выходе Q снимается FALSE и передаётся в переменную OK, которая инвертируется в TRUE и подаётся на нижний вход оператора AND. Это состояние готовности. При подаче управляющего сигнала run в оператор AND, последний выдаёт на своём выходе сигнал TRUE, который подаётся на вход IN оператора TON. После этого начинается отсчёт времени, по истечении которого, с Q выдаётся TRUE. Этот сигнал через переменную OK подаётся на инвертор оператора AND и преобразуется в FALSE. Далее, так как на AND подаются два сигнала TRUE и FALSE, то с его выхода теперь уже выдаётся сигнал FALSE, который сбрасывает оператор TON в исходное состояние. Таким образом, блок WAIT формирует короткий импульс через заданный промежуток времени. И если на вход run продолжает подаваться сигнал TRUE, то таймер WAIT снова начинает отсчёт времени и т. д.

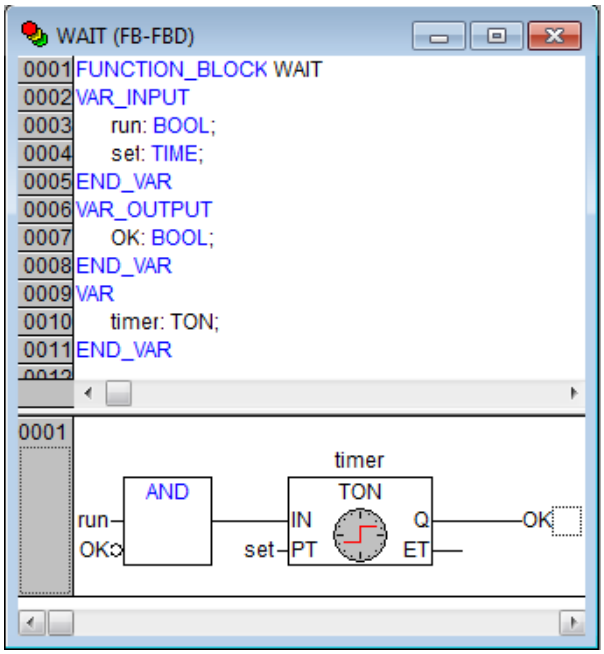

Рис. 21. Функциональный блок WAIT

3.2.8. Программирование SEQUENCE. Первоначально граф SFC состоит из этапа «Init» перехода «Trans0» и возврата назад к Init. Сначала необходимо сделать этапы для каждой стадии TRAFFICSIGNAL. Вставьте их, отмечая «Trans0» и выбирая команды «Insert»/«Step transition (after)». Повторите эту процедуру ещё три раза. Для редактирования названия перехода или этапа нужно просто щёлкнуть мышкой на нужном тексте. Назовите первый переход после Init – «START», а все прочие переходы «DELAY.OK».

Первый переход разрешается, когда START устанавливается в TRUE, все прочие – когда DELAY в OK станет TRUE, т. е. когда заданный период таймера WAIT закончится. Кстати для разнообразия можно запись DELAY.OK заменить локальной в SEQUENCE переменной «waited», которая будет принимать также значение OK из функционального блока WAIT, обращение к которому будет осуществляться через специальную инструкцию обращения. Делаться это будет в действиях, назначаемых в каждом из этапов – Switch1, Green2, Switch2, Green1, а также Init. Подробно это будет показано далее. Этап «Switch» должен включать жёлтую фазу, в Green1 TRAFFICSIGNAL1 будет зелёным, в Green2 TRAFFICSIGNAL2 будет зелёным. Наконец, измените адрес возврата Init на Switch1. Можно задать адрес возврата и на Init, тогда будет возможность управлять включением/выключением светофора после первого запуска.

При этом задайте переменные: входную START типа BOOL; выходные TRAFFICSIGNAL1 и TRAFFICSIGNAL1 типа INT; DELAY, относящаяся к блоку WAIT, waited типа BOOL.

В результате должна получиться архитектура стадий как на рис. 22.

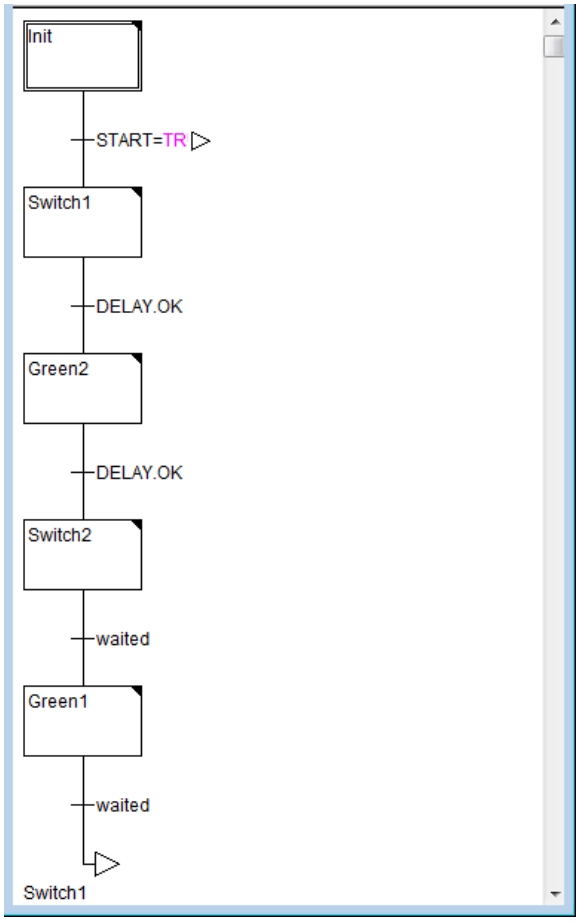

Рис. 22. Архитектура стадий SEQUENCE

Далее необходимо запрограммировать этапы и переходы. Если сделать двойной щелчок мыши на изображении этапа, то откроется диалог определения нового действия. Для программирования действия мы будем использовать язык ST. Назначьте в каждом назначаемом действии переменные TRAFFICSIGNAL1 и TRAFFICSIGNAL2 как на рис. 23.

Там же сделайте вызовы блока DELAY с помощью инструкции DELAY(run:=TRUE, set:=T#2s, OK=>waited); Где, в скобках через запятую первые две переменные run и set – это входы функционального блока WAIT, OK – это его выход. В случае если на переходах (рис. 4) используется запись DELAY.OK, то OK=> можно не назначать ничему. Если на переходах используется запись waited, то OK нужно назначить переменной waited, т. е. OK=>waited.

В результате должна получиться программа, как на рис. 23.

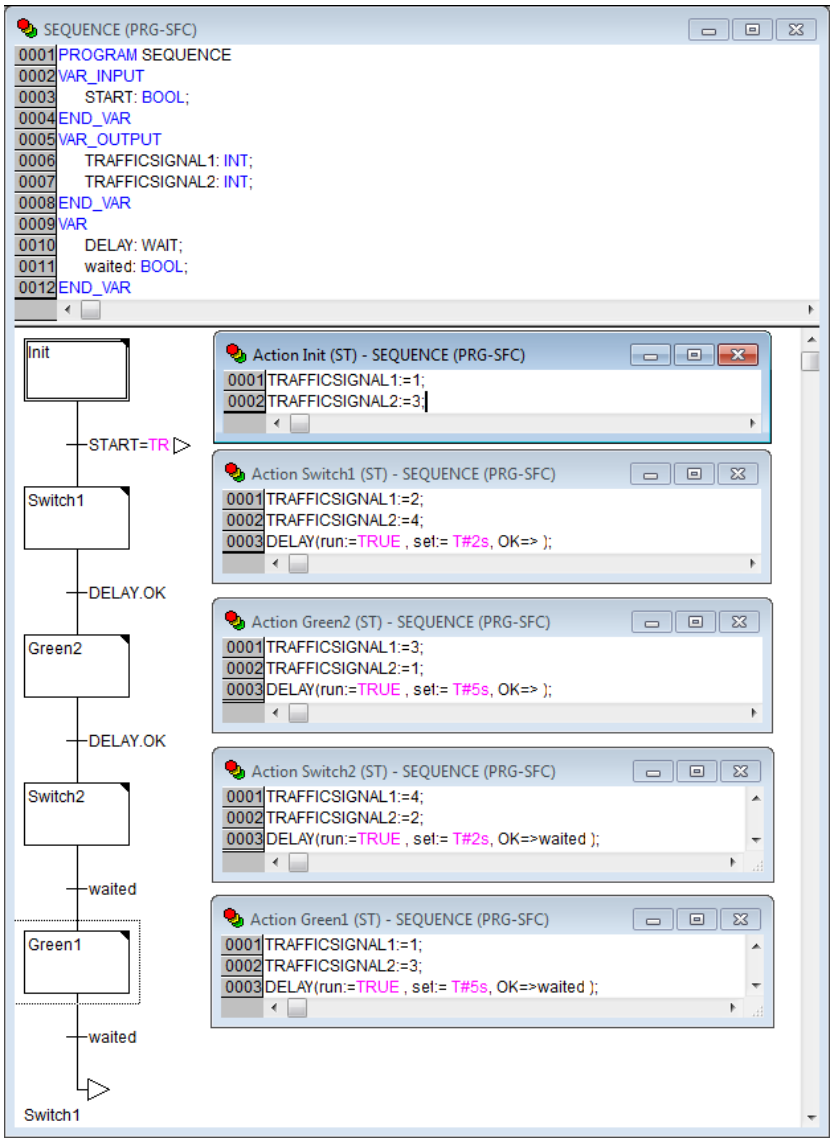

Рис. 23. Программа SEQUENCE

3.2.9. Программирование PLC_PRG. Здесь необходимо входные и выходные переменные в блоке PLC_PRG. Нужно дать возможность запустить систему выключателем IN и обеспечить переключение всех шести ламп (2 светофора) путём передачи команд переключения на каждом шаге SEQUENCE.

Объявите переменные LIGHT1 и LIGHT2 типа TRAFFICSIGNAL в редакторе объявлений. Для объявления переменных на все 6 ламп светофора воспользуйтесь глобальными переменными (Global Variables) из ресурсов (Resources). Двоичная входная переменная IN, необходимая для установки переменной START блока SEQUENCE в TRUE, будет определена таким же образом. Объявление глобальных переменных будет выглядеть как на рис. 24.

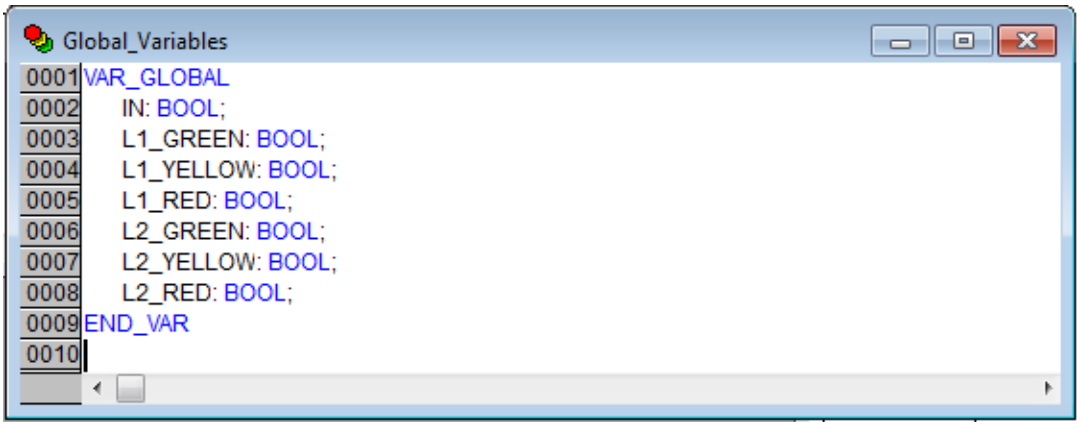

Рис. 24. Объявление глобальных переменных

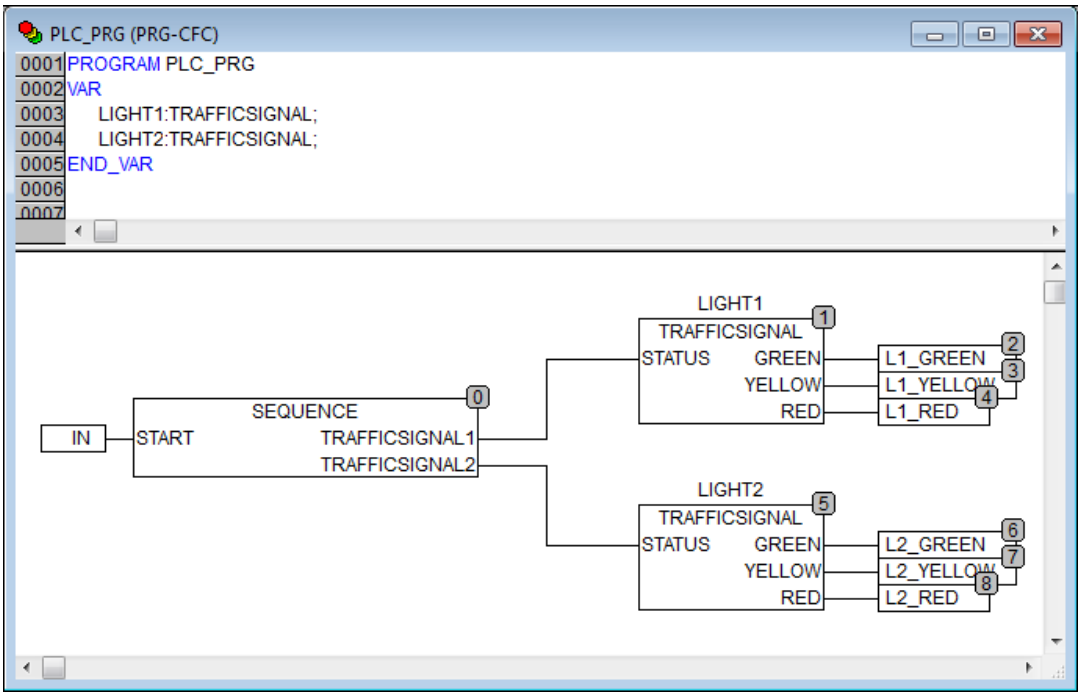

Рис. 25. Главная программа PLC_PRG

Завершим создание PLC_PRG. Щёлкните правой клавишей мыши в окне редактора и выберите элемент «Box». Щёлкните на тексте AND и напишите «SEQUENCE». Элемент автоматически преобразуется в SEQUENCE с уже определёнными входными и выходными переменными.

Вставьте два элемента и назовите их TRAFFICSIGNAL, которые являются функциональными блоками. Сверху их три знака вопроса нужно заменить на уже объявленные локальные переменные LIGHT1 и LIGHT2.

Создайте элемент типа «Input», который получит название IN и шесть элементов типа Output, которым нужно дать следующие имена: L1_green, L1_yellow, L1_red, L2_green, L2_yellow, L2_red.

Теперь все элементы можно соединить линиями, для этого необходимо щёлкнуть мышью на короткой линии входа/выхода и тянуть её, не отпуская клавишу мыши, к входу/выходу другого элемента.

Программа PLC_PRG должна иметь вид как на рис. 25.

3.3. Визуализация работы системы

3.3.1. Создание визуализации. Третья страничка организатора объектов CoDeSys называется визуализация (Visualization). Перейдите на страничку визуализации. В контекстном меню введите команду добавления объекта Add object. Присвойте новому объекту имя Lights.

3.3.2. Рисуем элементы визуализации.

- На панели инструментов выберите «Эллипс» (Ellipse) и начертите окружность диаметром около 2 см. Дважды щёлкните по нарисованной окружности и задайте следующие настройки:

– В категории переменных (Variables Category) щёлкните мышью в поле «изменение цвета» (Change Color) и воспользуйтесь ассистентом ввода «F2». Вставьте переменную «L1_red», это будет выглядеть так – .L1_red, с точкой перед самим именем. Эта переменная будет менять цвет данного элемента визуализации.

– В категории «цвета» (Colors) задайте нейтральный цвет закраски элемента (Inside), например чёрный. Для возбуждённого состояния необходимо определить другой цвет (Alarm color) – красный.

- Вызовите команду копирования (Edit/Copy) и дважды выполните команду вставки (Edit/Paste). Вы получите две новых окружности. Переместите окружности с помощью мышки одна под другой, чтобы они представляли собой вертикальный ряд. Настройте свойства для обеих новых окружностей. В поле «изменение цвета» Change Color: для средней окружности – .L1_yellow, для нижней – .L1_green. В категории «цвета» (Colors) в области «возбуждённого цвета» (Alarm color) установите цвета (жёлтый и зелёный).

- Нарисуйте корпус светофора с помощью команды «прямоугольник» (Insert/Rectangle). Вставьте прямоугольник так чтобы введённые ранее окружности находились внутри его. Выберите цвет прямоугольника и переместите эту фигуру на задний план командой Extras/Send to back. После этого окружности будут снова видны.
- Нарисуйте второй светофор путём копирования первого. Замените имена переменных, управляющими цветами (.L1_red на .L2_red; .L1_yellow на .L2_yellow; .L1_green на .L2_green).
- Нарисуйте переключатель, необходимый для старта светофора. Вставьте прямоугольник, в его настройках в поле «изменение цвета» (Change Color) категории переменных (Variables Category) введите переменную .IN. В поле «содержимое» (Content) категории «текст» (Text) введите имя «Включить». Для того, чтобы нажатие на эту кнопку влияло на состояние переменной IN, в поле Toggle variable категории Input введите переменную .IN. Для отображения включенного состояния необходимо вписать переменную .IN в поле «изменение цвета» (Change Color) категории «переменные» (Variables).

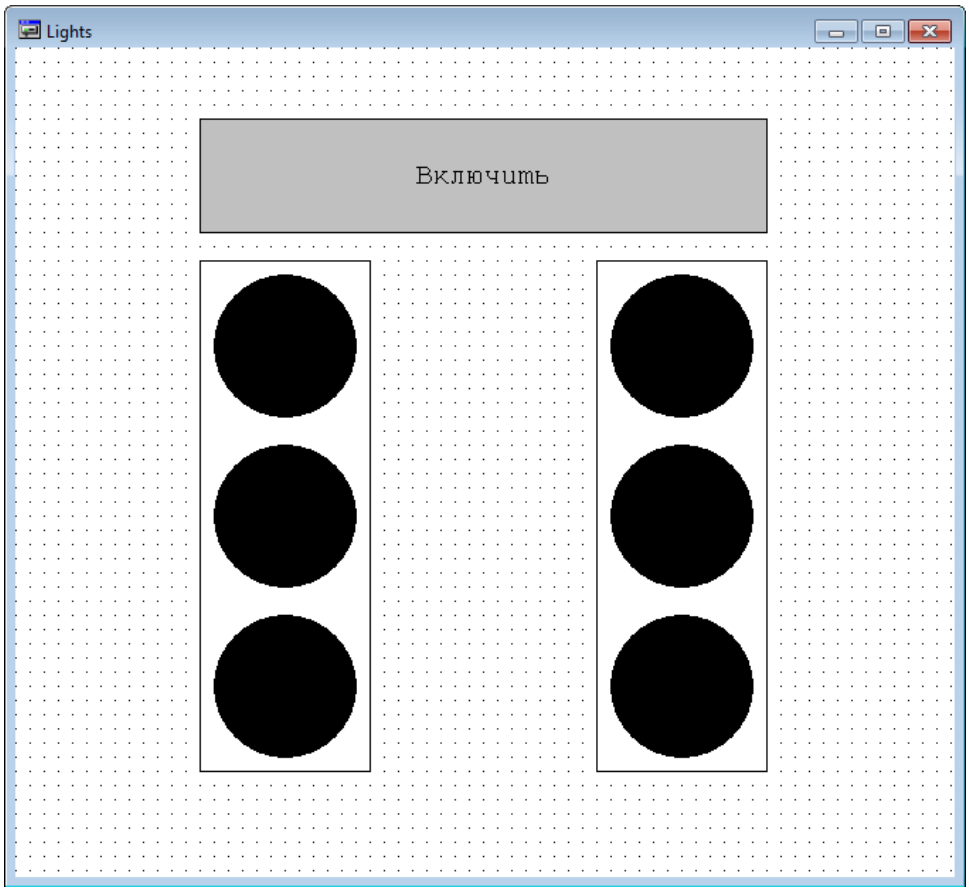

Рис. 26. Окно визуализации светофоров в исходном состоянии

В результате должно получиться окно визуализации как на рис. 26.

3.4. Запуск системы

3.4.1. Откомпилируйте проект командой «проект/компилировать» (Project/Rebuild all).

3.4.2. Установите соединение с контроллером «Связь/соединение» (Online/Login).

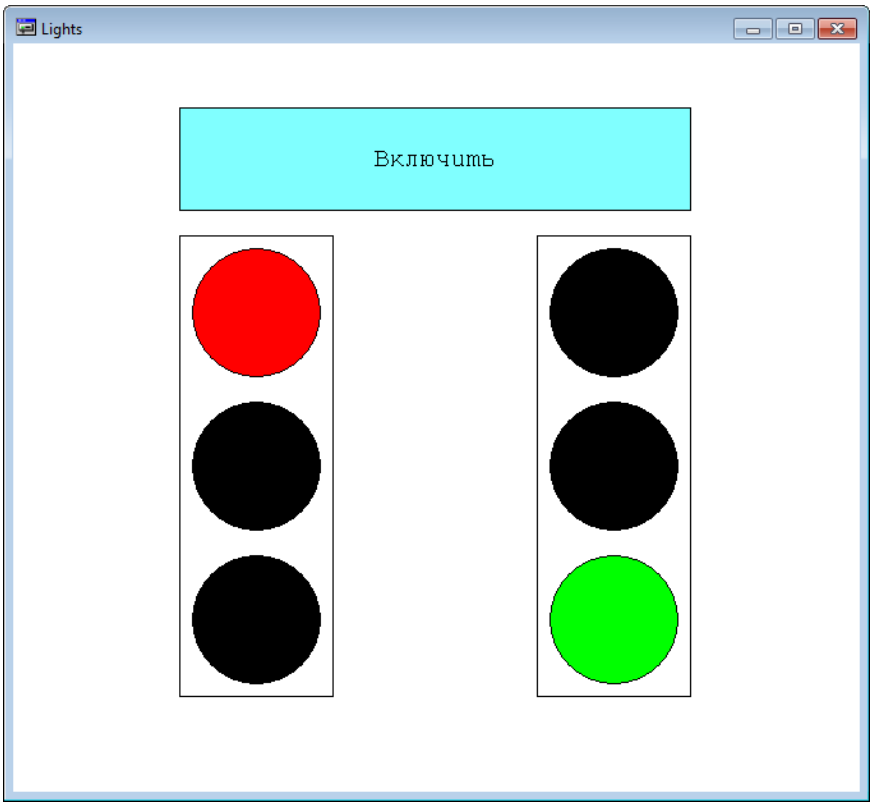

Рис. 27. Окно визуализации светофоров в рабочем состоянии

3.4.3. Запустите систему командой «Связь/запуск» (Online/Run). Нажмите на кнопку «Включить» в окне визуализации, которое в рабочем состоянии должно иметь вид как на рис. 27.

4. СИСТЕМА УПРАВЛЕНИЯ КОНВЕЙЕРОМ

Цель: освоить принципы построения объединённой интеллектуальной системы для управления физической моделью конвейера (ленточного транспортёра).

4.1. Порядок работы

4.1.1. Ознакомиться с лабораторной установкой конвейера.

4.1.2. Создать систему управления физической моделью на базе программных языков CFC и ST (среда CoDeSys).

4.1.3. Первая апробация созданной системы.

4.1.4. Создать систему управления физической моделью на базе программного языка LD (среда EasySoft).

4.1.5. Апробация созданной системы в среде Easy Soft.

4.2. Описание лабораторной установки

Физическая модель ленточного транспортёра представляет собой установку с транспортной лентой, приводимой в движение ведущим барабаном, который в свою очередь вращается от приводного двигателя. Над лентой расположена оптическая система для контроля габаритов транспортируемых грузов и их высоты. В конце транспортного пути расположены две приёмные корзины, которые предназначены для приёма условно как высоких, так и низких грузов, которые сбрасываются поворотной площадкой. Выбор направления для сброса в ту или иную корзину определяется системой управления. Также под каждой корзиной имеются датчики веса грузов. Ознакомительный эскиз лабораторной установкой ленточного транспортёра приведён на рис. 28.

Система управления физической моделью использует следующие реализации контроля:

1. Контроль позиционирования грузов по ширине ленты. Реализуется с помощью лазерных лучей по бокам вдоль ленты.
2. Контроль заданной высоты транспортируемых грузов за счёт использования двух световых лучей на двух разных высотах поперёк направления движения ленты.
3. Контроль наполненности корзин грузами с помощью 2 датчиков перемещения.
4. Контроль скорости ленты, устанавливается оптодатчик скорости на ведомом барабане. Датчик подсчитывает число светоимпульсов через вращающееся зеркало, измеряя также и длительность между импульсами.
5. Контроль проскальзывания ленты относительно ведущего барабана, реализуется путём сравнения скоростей ведущего барабана и ленты. При этом на ведущем барабане также установлено зеркало для отражения света на оптодатчик.

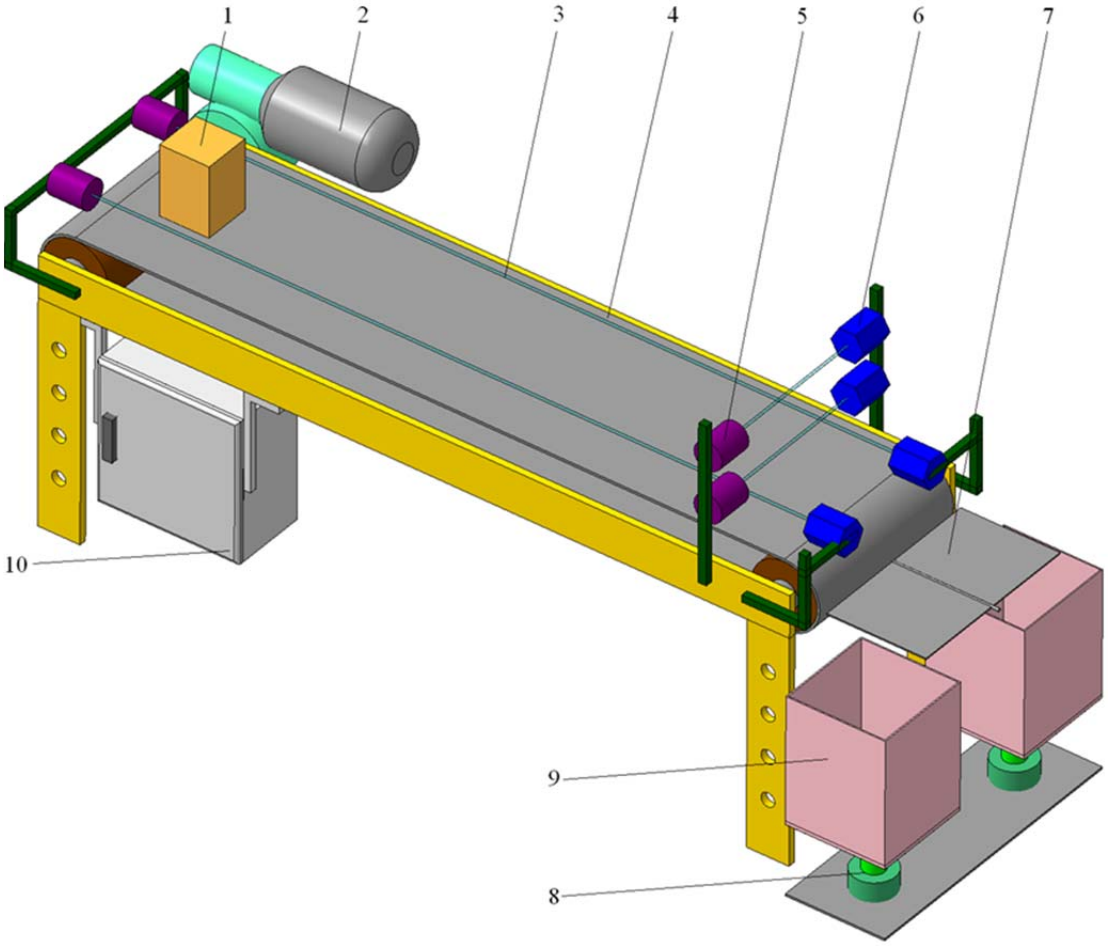

Рис. 28. Физическая модель конвейера: 1 – транспортируемый груз; 2 – электродвигатель асинхронный; 3 – световой луч; 4 – лента; 5 – источник света; 6 – оптодатчик; 7 – направляющая поворотная площадка для спуска грузов в корзины 9; 8 – датчики перемещения для измерения веса грузов; 9 – приёмные корзины для сбора грузов; 10 – силовой шкаф с электрооборудованием

Таблица 1

Подключение датчиков к различным аппаратам

Назначение	Контакт на DB разъёме	Поступление на входы					
		Easy-820	ES4P	EC4P	XC-CPU201	XIOC-8DI	XIOC-8AI-U1
Левый габарит	1	I5	I5	I5	I0.0	I0	U0+
Правый габарит	2	I6	I6	I6	I0.1	I1	U1+
Высокий груз	3	I7*	I7*	I7*	I0.2	I2	U2+
Низкий груз	4	I8*	I8*	I8*	I0.3	I3	U3+
Вес левой корзины	5	I11*	I11*	I11*	I0.4	I4	U4+
Вес правой корзины	6	I12*	I12*	I12*	I0.5	I5	U5+
Ведущий вал	7	I9	I9	I9	I0.6	I6	U6+
Ведомый вал	8	I10	I10	I10	I0.7	I7	U7+

Таблица 2

Подключение исполнительных элементов к различным аппаратам

Назначение	Контакт на DB разъёме	Управление с выходов				
		Easy-820	ES4P	EC4P	XC-CPU201	XIOC-8DO
Прямой пуск	1	Q1	QS1	Q1	Q0.0	0
Сброс груза влево	2	Q2	QS2	Q2	Q0.1	1
Сброс груза вправо	3	Q3	QS3	Q3	Q0.2	2
Двоичное задание частоты 001	4	Q4	QS4	Q4	Q0.3	3
Двоичное задание частоты 010	5	Q5	–	Q5	Q0.4	4
Двоичное задание частоты 100	6	Q6	–	Q6	Q0.5	5
Включение ПЧ дистанционно	7	–	–	S1	–	6
–	8	–	–	–	–	–
Аналоговое задание частоты	9	QA1	–	QA1	–	–
Аналоговое ПИД управление	10	–	–	QA2	–	–

4.3. Создание системы управления в среде CoDeSys

4.3.1. Запуск программной среды CoDeSys. Запустите программу в Windows через Пуск/Программы/Moeller Software/easy Soft CoDesys V2.3.9.

4.3.2. Создайте новый проект командой File/New.

4.3.3. Настройка целевой платформы (Target Settings). На странице диалогового окна «Configuration» выберите тип контроллера EC4P-200 V2.3.9 и подтвердите ввод – ОК.

4.3.4. Главная программа PLC_PRG POU. Следующее диалоговое окно определяет тип первого программного компонента (New POU). Выберите язык реализации (language of POU) CFC и сохраните предложенные по умолчанию тип компонента – программа (Type of the POU Program) и имя – Name PLC_PRG. PLC_PRG – это особый программный компонент (POU). В однозадачных проектах он циклически вызывается системой исполнения. Эта программа будет объединять все составляющие функциональные блоки, каждый из которых отвечает за тот или иной контроль работы всей системы.

4.3.5. Создание дополнительных функциональных блоков.

- *Блок запуска двигателя.* Командой «Проект/Добавить объект» (Project/Object Add) добавьте новый объект (делается это в первой системной вкладке, где список POU). Объект должен быть функциональным блоком (в появившемся диалоговом меню выбрать тип POU – «Function Block») на языке Structured Text (ST) с именем «motor».

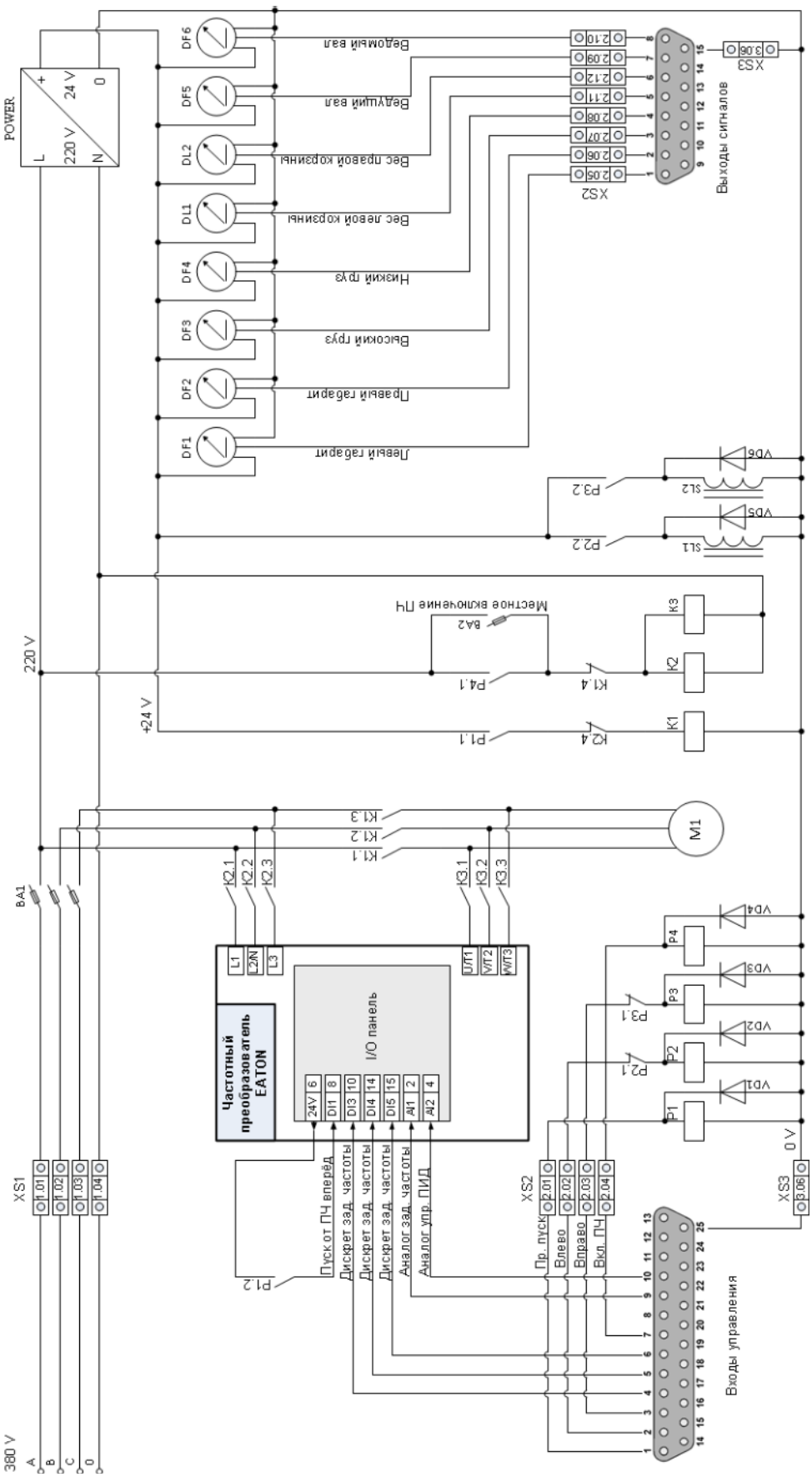

Рис. 29. Транспортёр ленточный. Схема электрическая управления

Объявите следующие входные переменные в редакторе объявлений:

– start типа BOOL – для запуска двигателя;

– stop типа BOOL – для остановки двигателя;

– glide_isset типа BOOL – переменная наличия проскальзывания ленты конвейера;

– left_dimension типа BOOL – переменная наличия выхода транспортируемого груза за левую границу (перекрытие светового луча левого габарита).

– right_dimension типа BOOL – переменная наличия выхода транспортируемого груза за правую границу (перекрытие светового луча правого габарита).

– full_weight типа BOOL – переменная полного заполнения грузами любой из приёмных корзин.

Объявите выходную переменную в редакторе объявлений «work» типа BOOL – переменная, которая в состоянии TRUE подаёт сигнал на запуск двигателя.

В рабочей области редактора блока «motor» пропишите следующий программный код:

```
IF glide_isset = FALSE AND left_dimension = FALSE AND
right_dimension = FALSE AND full_weight = FALSE
THEN
  IF start = TRUE THEN work:=TRUE; END_IF;
  IF stop = TRUE THEN work:=FALSE; END_IF;
  IF start = TRUE AND stop = TRUE THEN work:=FALSE;
END_IF;
ELSE
work:=FALSE;
END_IF;
```

Данный программный код в начале проверяет, если переменные наличия каких-либо нарушений равны FALSE, то далее назначить переменной «work» – TRUE, если «start» равна TRUE (нажата кнопка пуска конвейера), и назначить FALSE, если переменная «stop» равна TRUE (нажата кнопка стопа конвейера).

Добавление нового функционального блока показано на рис. 30.

Окно блока «motor» показано на рис. 31.

- *Блок контроля проскальзывания ленты.* Добавьте новый функциональный блок и назовите его «glide» с языковой реализацией ST.

Объявите следующие входные переменные в редакторе объявлений:

– velocity1 и velocity2 типа REAL – переменные скоростей вращения соответственно ведомого и ведущего барабанов конвейера.

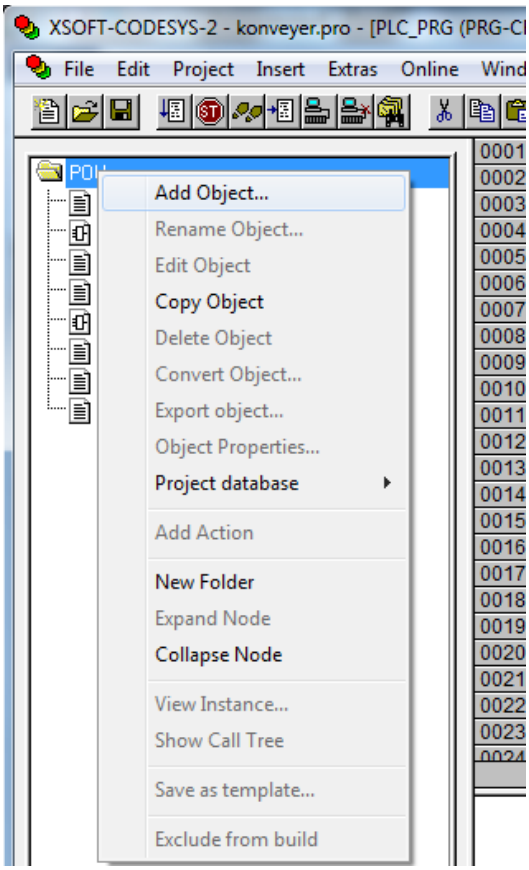

Рис. 30. Добавление нового функционального блока

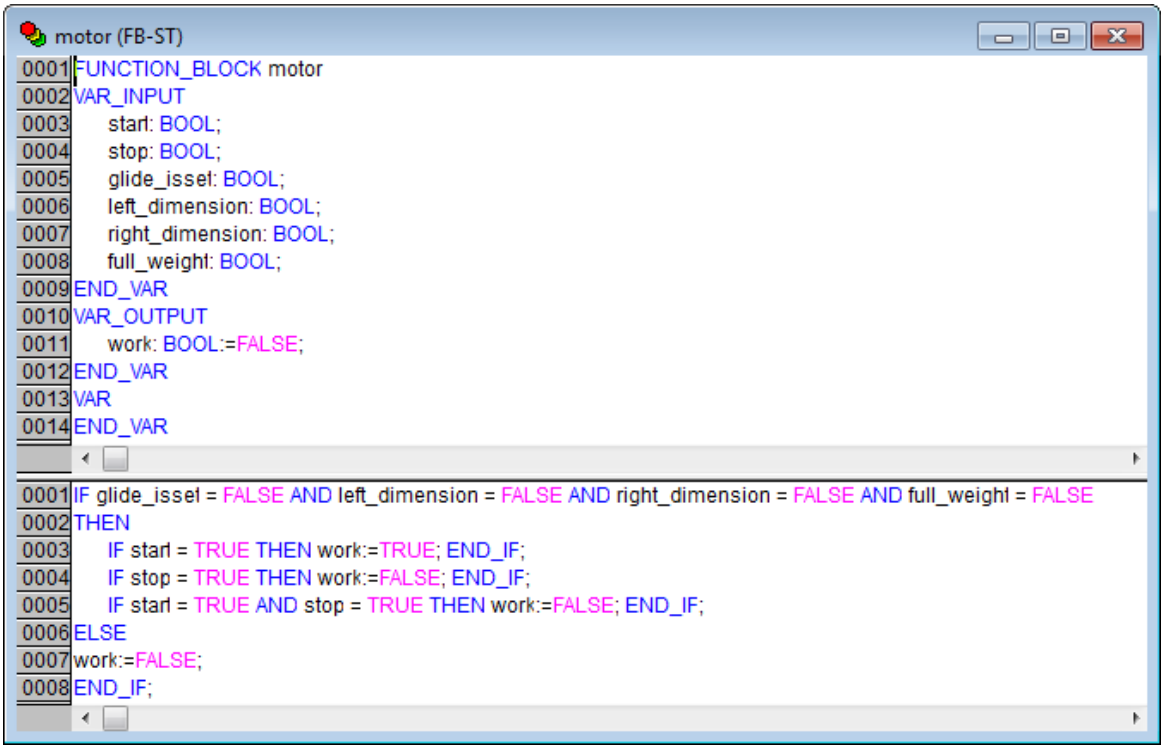

Рис. 31. Окно функционального блока «motor»

Объявите выходную переменную в редакторе объявлений «glide» – переменная типа BOOL, которая сигнализирует о наличии проскальзывания ленты конвейера.

Объявите константу в редакторе объявлений «glide_limit» – константа типа REAL, которая задаёт допустимую разницу между скоростями вращения ведущего и ведомого барабанов конвейера.

Вставьте в рабочую область блока «glide» следующий код:

IF (velocity2-velocity1) > glide_limit
THEN glide:=**TRUE**; **ELSE** glide:=**FALSE**;
END_IF;

Окно функционального блока «glide» представлено на рис. 32.

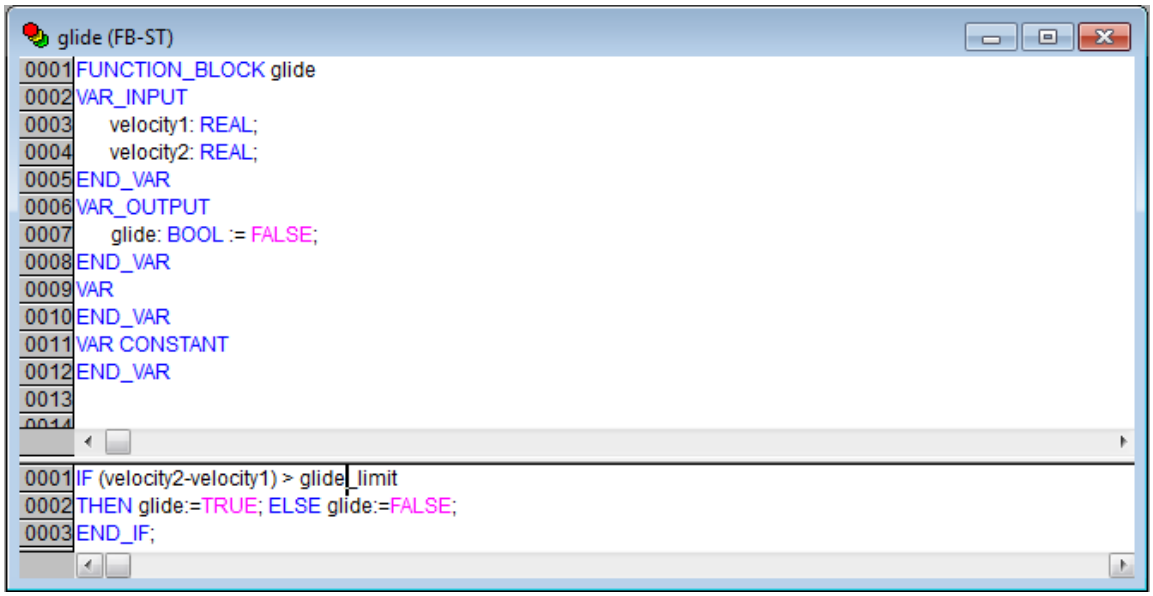

Рис. 32. Окно функционального блока «glide»

- *Блок измерения скорости вращения барабанов конвейера.* Добавьте новый функциональный блок и назовите его «velocity» с языковой реализацией ST.

Объявите входную переменную в редакторе объявлений «sensor» типа BOOL – переменная наличия импульса от фотодатчика, принимающего свет от зеркала, которое вращается вместе с барабаном.

Объявите выходную переменную в редакторе объявлений «velocity» типа REAL – переменная, выдающая значение окружной скорости вращения барабана в см/сек.

Объявите следующие локальные переменные в редакторе объявлений:

– strob – переменная типа BOOL, необходимая для фиксации момента замера времени появления импульса от фотодатчика;

– time1 – переменная типа TIME, принимающая значение момента времени появления «предыдущего» импульса от фотодатчика;

– time2 – переменная типа TIME, принимающая значение момента времени появления «следующего» импульса от фотодатчика;

– interval_time – переменная типа TIME, принимающая значение интервала между «предыдущим» и «следующим» импульсами от фотодатчиков.

– interval_dword – переменная типа DWORD (двойное слово), т. е. это то же время interval_time, только в формате двойного слова;

– interval_real – переменная типа REAL, это то же время interval_dword, только измеряется в миллисекундах.

Объявите константу в редакторе объявлений «sensor_way» – это геометрический путь по длине окружности между зеркалами на барабане конвейера.

Вставьте в рабочую область блока «velocity» следующий код:

```
IF sensor = TRUE AND strob = TRUE
THEN
interval_time:=time2-time1;
time1:=TIME();
strob:=FALSE;
END_IF;

IF sensor = FALSE
THEN
strob:=TRUE;
END_IF;

time2:=TIME();
interval_dword:=TIME_TO_DWORD(interval_time);
interval_real:=(DWORD_TO_REAL(interval_dword)) / 1000; (*Secund*)

velocity:=sensor_way / interval_real; (*Santimeter / Secund*)
```

Принцип работы кода состоит в следующем. При подачи на вход блока «velocity» в переменную «sensor» значения TRUE, так как «strob»=TRUE, начальное время time1 принимает какой-то начальный момент времени, затем блокиратор «strob» принимает FALSE. Теперь даже повторном чтении данного кода вычислительной машиной (ПК), уже при «strob»=FALSE, первое «IF» условие уже не будет выполняться, а значит переменная «time1» далее будет сохранять своё ранее принятое значение. Разница «interval_time:=time2-time1» пока для нас не представляет особого значения, кроме того, что она в начале прочитывания компьютером кода будет равна 0. При снятии сигнала «sensor», т. е. при переводе его в FALSE, блокиратор «strob» принимает значение

TRUE и тем самым производится подготовка к следующему запуску первого условия «IF» для последующей стадии расчёта временного интервала. Пока «sensor» находится в FALSE, переменная «time2» набирает свою величину реального времени. Как только «sensor» станет TRUE, сработает первый оператор «IF» и в нём произойдёт расчёт разности времени между конечным значением «time2» и начальной временной точкой «time1», которая до этого момента сохранялась постоянной. Таким образом, происходит вычисление времени между «предыдущим» импульсом «sensor» и «последующим». После этого «time1» принимает новое «стартовое» время, и, когда блокиратор «strob» станет FALSE, «time1» останется постоянным до следующего такого же импульса «sensor». После каждого расчёта разности времени между импульсами «sensor» производится вычисление скорости движения ленты конвейера путём деления расстояния, пройденного лентой от одного момента подачи «sensor» до следующего, на время между этими подачами сигнала «sensor». Причём это время и есть та разница «interval_time:=time2-time1». Подача импульсов «sensor» напрямую связана с отражением света от зеркал на барабанах конвейера и попаданием его на фотодатчики.

Окно функционального блока «velocity» представлено на рис. 33.

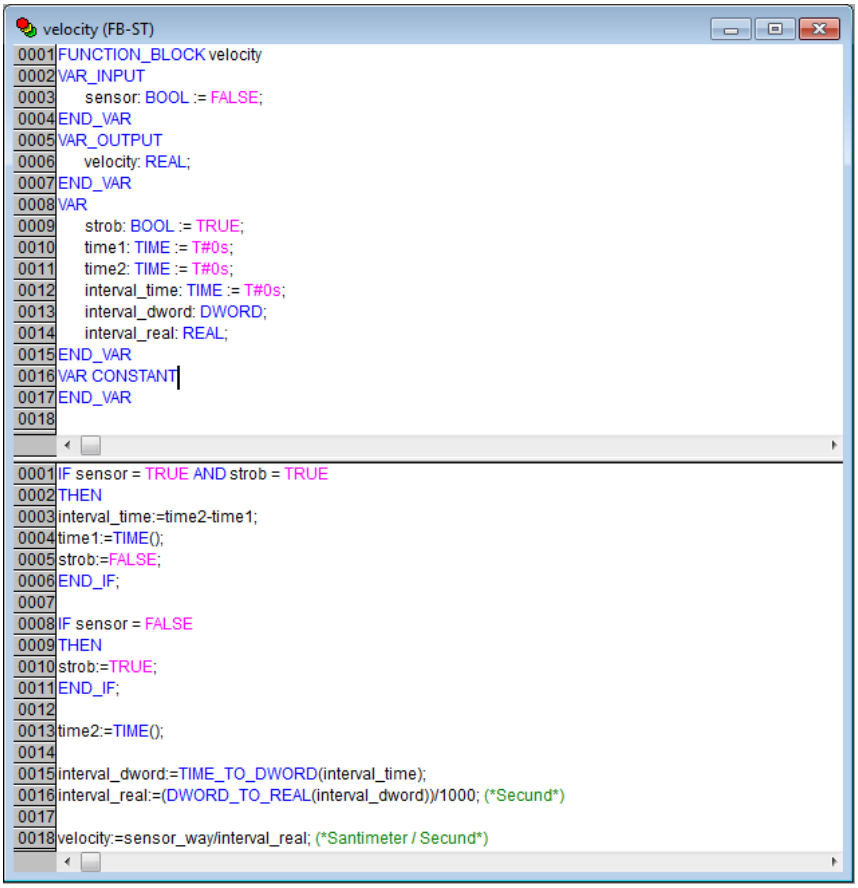

Рис. 33. Окно функционального блока «velocity»

- *Блок сигнала сброса груза.* Добавьте новый функциональный блок и назовите его «stage» с языковой реализацией ST.

Объявите входную переменную в редакторе объявлений «sensor» типа BOOL – переменная наличия импульса, когда перекрывается световой луч на фотодатчик, контролирующий высокий или низкий груз.

Объявите выходную переменную в редакторе объявлений «movie» типа BOOL – переменная, выдающая сигнал на поворот площадки для сброса груза в приёмную корзину.

Объявите следующие локальные переменные в редакторе объявлений:

– time1 – переменная типа TIME;
– time2 – переменная типа TIME;
– interval_time – переменная типа TIME.
– interval_dword – переменная типа DWORD (двойное слово), т. е. это то же время interval_time, только в формате двойного слова;
– interval_real – переменная типа REAL, это то же время interval_dword, только измеряется в миллисекундах.

Объявите константу в редакторе объявлений «interval_init» – это постоянная времени, в течение которой поворотная площадка остаётся ещё в наклонённом состоянии для того, чтобы после перекрытия грузом светового луча, груз ещё успел сброситься в корзину.

Вставьте в рабочую область блока «stage» следующий код:

```
IF sensor = TRUE
THEN
movie:=TRUE;
END_IF;

IF sensor = FALSE
THEN
        time2:=TIME();
        interval_time:=time2-time1;
        interval_dword:=TIME_TO_DWORD(interval_time);
        interval_real:=DWORD_TO_REAL(interval_dword);
        IF ((interval_real/1000) > interval_init)
        THEN
        movie:=FALSE;
        END_IF;
ELSE
time1:=TIME();
END_IF;
```

Окно функционального блока «stage» представлено на рис. 34.

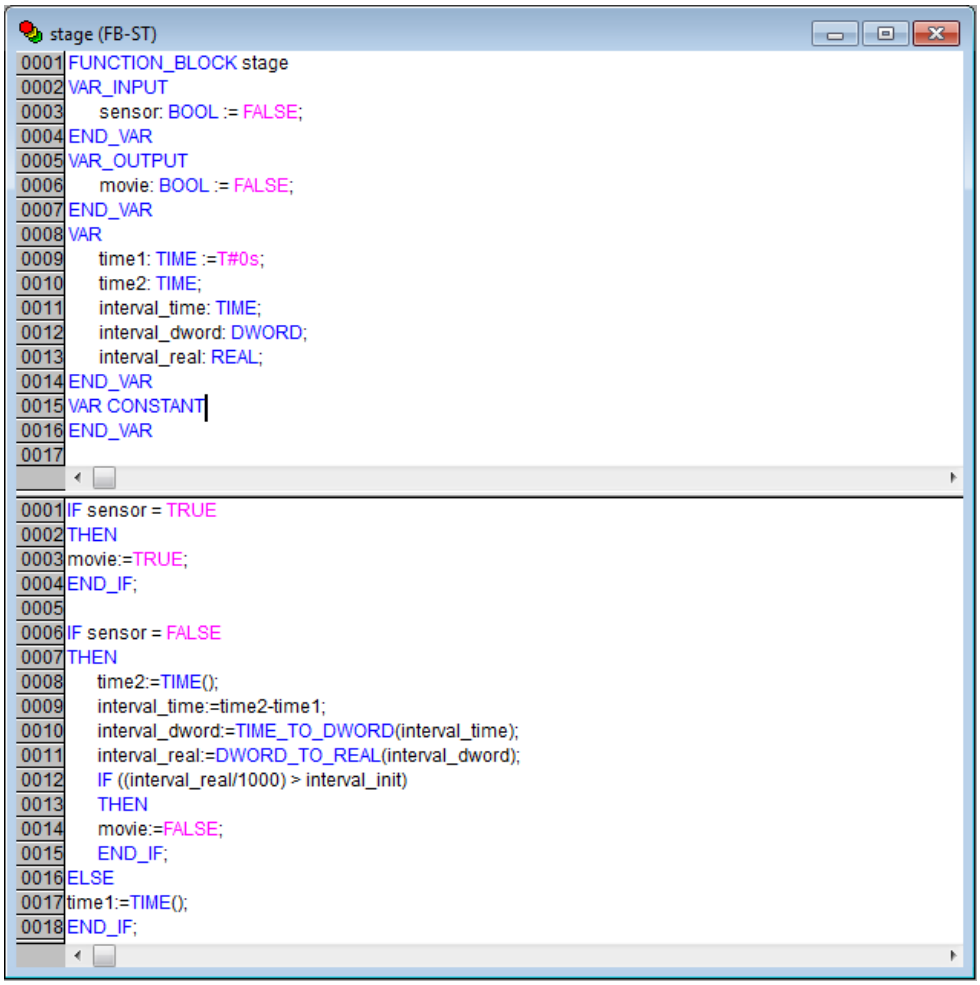

Рис. 34. Окно функционального блока «stage»

- *Блок координации поворотной площадки.* Добавьте новый функциональный блок и назовите его «stage_prove» с языковой реализацией ST.

Объявите следующие входные переменные в редакторе объявлений «stage_prove»:

– high_input – переменная типа BOOL, принимающая сигнал при перекрытии светового луча высоким грузом;

– low_input – переменная типа BOOL, принимающая сигнал при перекрытии светового луча низким грузом;

– Объявите следующие выходные переменные в редакторе объявлений «stage_prove»:

– move_high – переменная типа BOOL, выдающая сигнал поворота площадки в сторону корзины для высоких грузов;

– move_low – переменная типа BOOL, выдающая сигнал поворота площадки в сторону корзины для низких грузов;

Вставьте в рабочую область блока «stage_prove» следующий код:

```
IF high_input = TRUE AND low_input = TRUE
THEN
movie_high:=TRUE;
movie_low:=FALSE;
ELSE
        IF high_input = FALSE THEN movie_high:=FALSE;
END_IF;
        IF high_input = TRUE THEN movie_high:=TRUE; END_IF;
        IF low_input = FALSE THEN movie_low:=FALSE; END_IF;
        IF low_input = TRUE THEN movie_low:=TRUE; END_IF;
END_IF;
```

Смысл блока «stage_prove» – предотвратить ложную подачу сигналов на поворот площадки сброса грузов одновременно в одно и в другую стороны, когда высокий груз перекрывает два световых луча, контролирующих соответственно высокие и низкие грузы. Окно функционального блока «stage_prove» представлено на рис. 35.

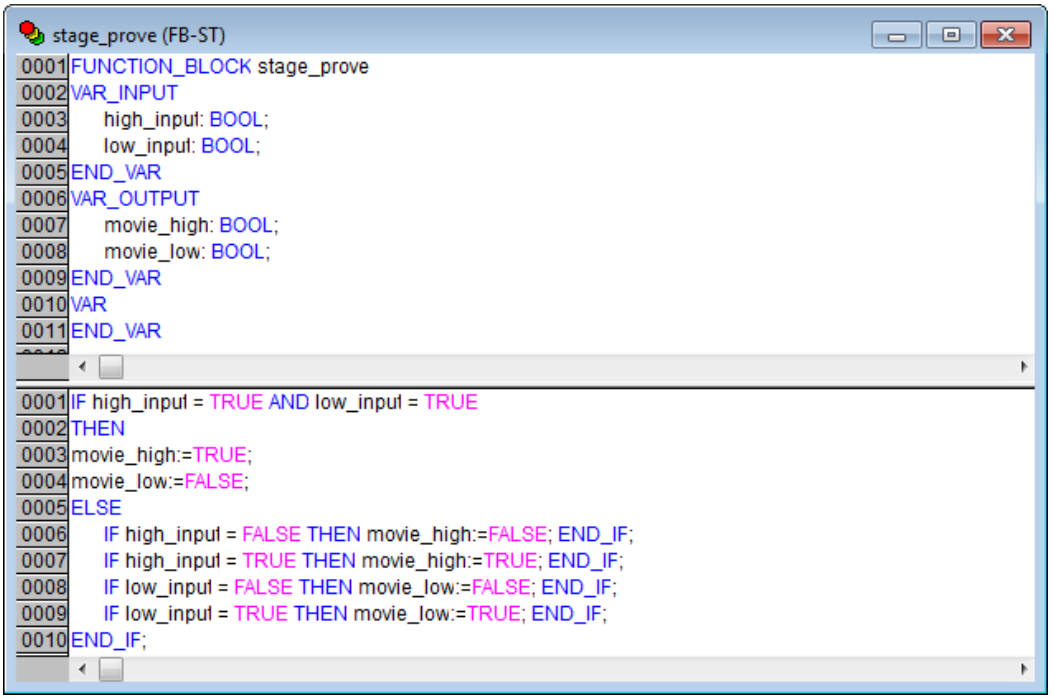

Рис. 35. Окно функционального блока «stage_prove»

- *Блок инерционной задержки сигнала.* Добавьте новый функциональный блок и назовите его «internal» с языковой реализацией FBD.

Необходимость в этом блоке обусловлена вероятностью ложных импульсов с датчиков веса, что могло бы привести к остановке конвейера за счёт ложного поступления сигнала на вход «full_weight» блока «motor».

Окно функционального блока «internal» представлено на рис. 36.

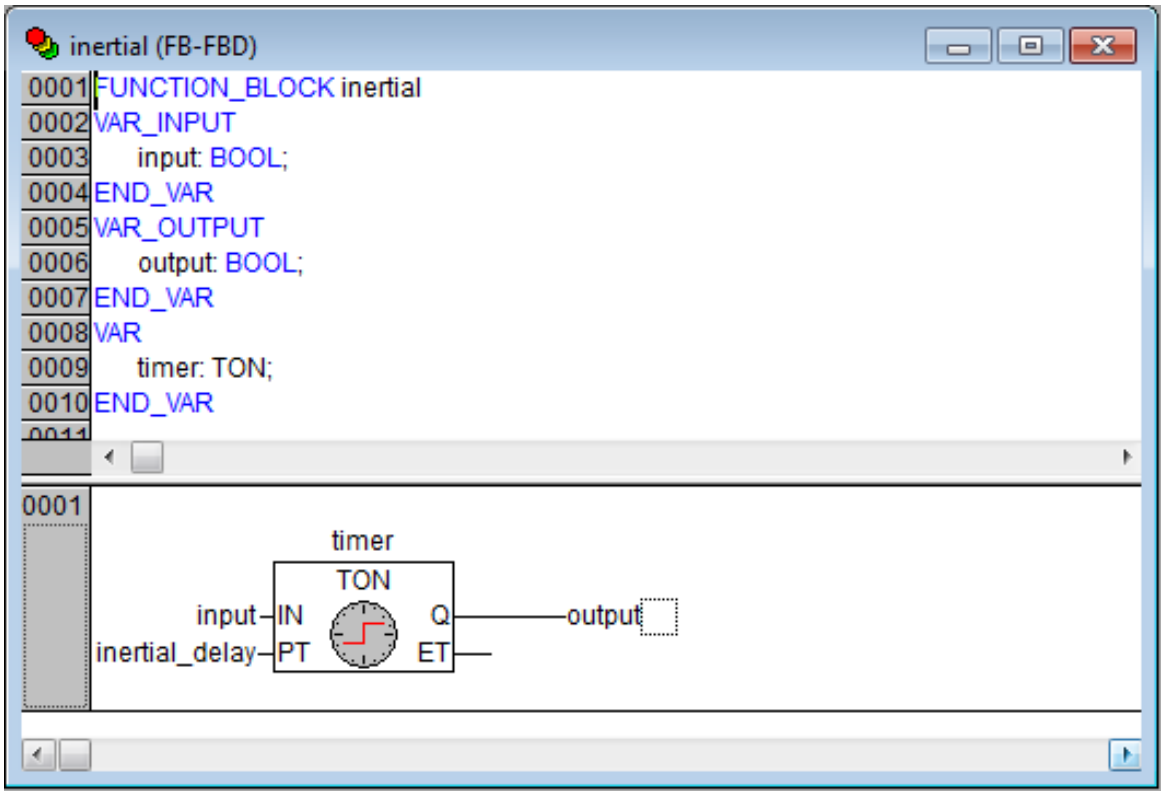

Рис. 36. Окно функционального блока «internal»

- *Блок задания скорости вращения двигателя.* Необходим для случая с использованием частотного преобразователя, когда по рис. 29 запитаны контакторы К2 и К3. Добавьте новый функциональный блок и назовите его «Manage_motor» с языковой реализацией ST.

Задайте переменные и создайте программный код, как показано на рис. 37.

У данного блока есть две входные переменные «rotation_up» и «rotation_down» для увеличения или уменьшения внутренней переменной «CV» счётчика «CTUD_1». Эта переменная передаёт своё значение на внутреннюю переменную «pv» функционального блока «Manage_motor».

С помощью условных операторов производится проверка значения переменной «pv» и на основании него формируется двоичный 3-х разрядный код. Этот код поразрядно подаётся на выходные переменные «Q4», «Q5», «Q6» функционального блока «Manage_motor».

Последние значения двоичного кода в виде трёх отдельных сигналов подаются через электрические выходы контроллера на входы «DI3», «DI4», «DI5» преобразователя частоты (рис. 29).

Окно функционального блока «Manage_motor» представлено на рис. 37.

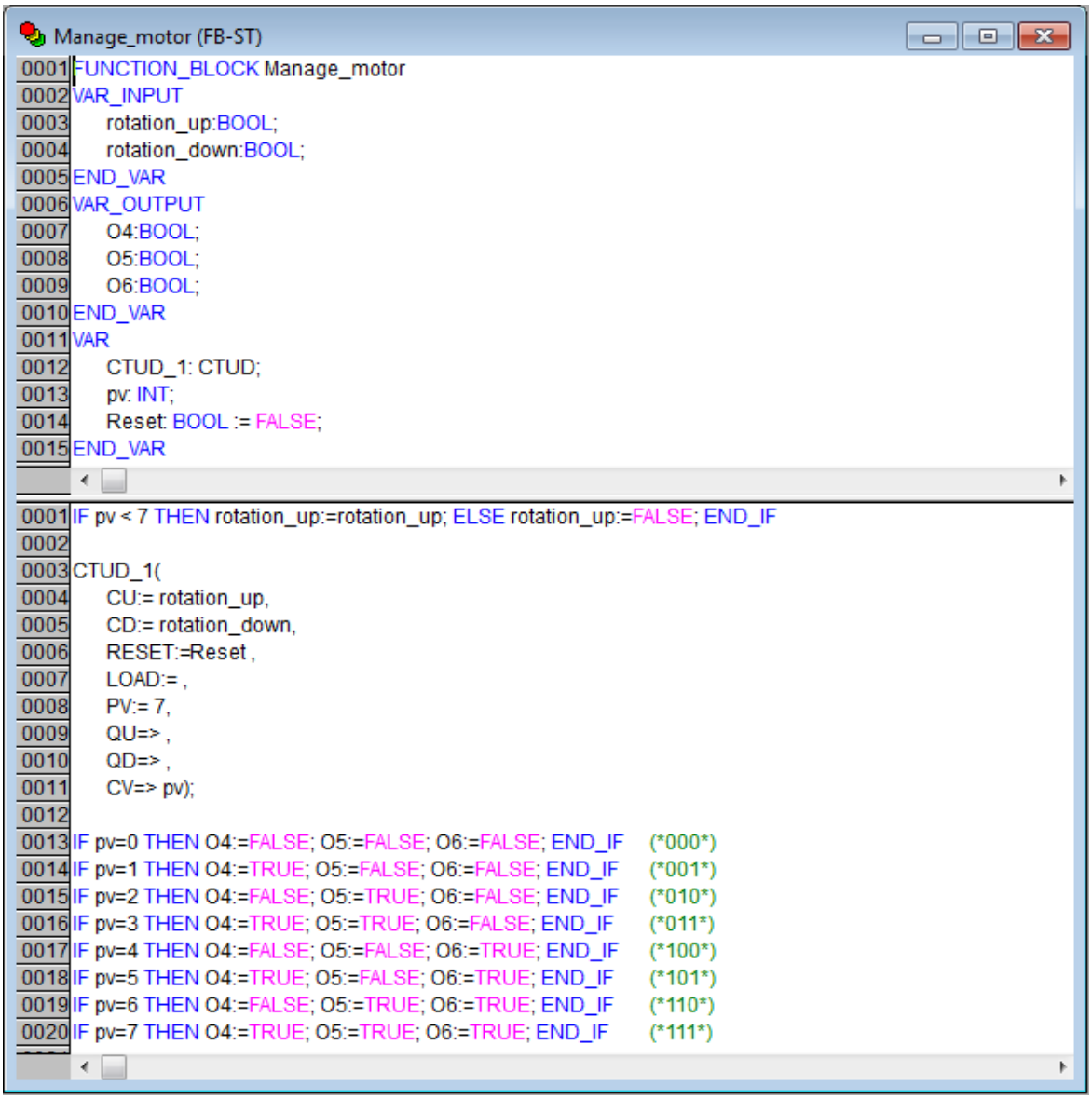

Рис. 37. Окно функционального блока «Manage_motor»

4.3.6. Завершение главной программы PLC_PRG и объединение всех блоков в систему.

Вернитесь в редактор программы PLC_PRG. Далее будет необходимо объявить переменные, причём некоторые из них будут напрямую связаны с физическими входами и выходами контроллера EC4P-200. Делаться это должно с помощью синтаксиса: «имя переменной» AT %IX0.0: BOOL; Здесь приведён пример для булевой переменной, связанной с первым по счёту «I1» входом контроллера с помощью записи 0.0 после IX. Обозначение релейного выхода контроллера обозначается QX. Например «имя переменной» AT %IX0.1: BOOL – переменная связанная со вторым «Q2» релейным выходом.

Вообще, чтобы определить как правильно прописывать в такой ситуации физические входы и выходы, необходимо открыть в разделе «ресурсы» (Resources) вкладку «конфигурация контроллера» (PLC Configuration), и в диалоговом окне открыть опцию «Configuration EC4P-200», а ней «Local I/O[SLOT]». Там будет всё отчётливо прописано для каждого входа и выхода аппарата.

Объявите следующие переменные в редакторе объявлений «PLC_PRG»:

- high_sensor AT %IX0.6: BOOL – переменная, принимающая сигнал от датчика высоких грузов;
- low_sensor AT %IX0.7: BOOL – переменная, принимающая сигнал от датчика низких грузов;
- velo_sensor1 AT %IX10.1: BOOL – переменная, принимающая импульсы от фотодатчика к ведомому барабану;
- velo_sensor2 AT %IX10.2: BOOL – переменная, принимающая импульсы от фотодатчика к ведущему барабану;
- left_sensor AT %IX0.4: BOOL – переменная, принимающая сигнал перекрытия светового луча левого габарита;
- right_sensor AT %IX0.5: BOOL – переменная, принимающая сигнал перекрытия светового луча правого габарита;
- weight_high AT %IX1.2: BOOL – переменная, принимающая сигнал от датчика веса под корзиной для высоких грузов;
- weight_low AT %IX1.3: BOOL – переменная, принимающая сигнал от датчика веса под корзиной для низких грузов;
- start_button AT %IX0.0: BOOL – переменная, принимающая сигнал от кнопки пуска двигателя;
- stop_button AT %IX0.1: BOOL – переменная, принимающая сигнал от кнопки остановки двигателя;
- rotation_up AT %IX0.2: BOOL – переменная, принимающая сигнал от кнопки увеличения частоты вращения двигателя;
- rotation_down AT %IX0.3: BOOL – переменная, принимающая сигнал от кнопки уменьшения частоты вращения двигателя;
- work_motor AT %QX0.0: BOOL – переменная, выдающая сигнал для пуска двигателя;
- movie_high AT %QX0.1: BOOL – переменная, выдающая сигнал для поворота площадки в сторону корзины для высоких грузов;
- movie_low AT %QX0.2: BOOL – переменная, выдающая сигнал для поворота площадки в сторону корзины для низких грузов;
- Q4 AT %QX0.3: BOOL – переменная, выдающая сигнал первого регистра для задания частоты вращения двигателя через ПЧ;

- Q5 AT %QX0.4: BOOL – переменная, выдающая сигнал второго регистра для задания частоты вращения двигателя через ПЧ;
- Q6 AT %QX0.5: BOOL – переменная, выдающая сигнал третьего регистра для задания частоты вращения двигателя через ПЧ;
- velocity_mesure1: velocity – локальное в «PLC_PRG» имя функционального блока «velocity»; velocity_mesure2 – то же;
- glide_check: glide – локальное в «PLC_PRG» имя функционального блока «glide»;
- movie_stage: stage – локальное в «PLC_PRG» имя функционального блока «stage»; movie_stage2 – то же;
- stage_check: stage_prove – локальное в «PLC_PRG» имя функционального блока «stage_prove»;
- start_motor: motor – локальное в «PLC_PRG» имя функционального блока «motor»;
- weight_height_internal: internal – локальное в «PLC_PRG» имя функционального блока «internal».
- weight_low_internal: internal – локальное в «PLC_PRG» имя функционального блока «internal».
- M_M: Manage_motor – локальное в «PLC_PRG» имя функционального блока «Manage_motor».
- PVM: INT – локальное в «PLC_PRG».

На рис. 38 показаны переменные корневой программы «PLC_PRG».

Создайте в рабочем поле редактора новые элементы и назовите их также, как все составляющие функциональные блоки. Сверху знаки вопросов замените на объявленные в PLC_PRG локальные переменные, причём так, чтобы эти имена соответствовали предназначению функционального блока как в поле объявлений. То есть например блок с названием «velocity» имеет сверху имя «velocity_mesure1», как в объявлениях velocity_mesure1: velocity. Создайте также в рабочем поле редактора входы и выходы в соответствии с именами объявленных переменных для сенсоров, кнопок и переменных управления поворотной площадкой и двигателем. В результате должно получиться окно программы как на рис. 39.

Запуск двигателя производится с помощью блока «start_motor» путём подачи от кнопки пуска на вход «start» сигнала TRUE при условии, что на контрольные входы «glide_isset», «left_dimension», «right_dimension» и «full_weight» поступают сигналы FALSE.

Разберём остальные блоки, подающие сигналы на контрольные входы «start_motor». Блок «glide_check» вычисляет разность между скоростями вращения ведущего и ведомого барабанами. На входы этого

блока поступают сигналы о величинах скорости этих барабанов от блоков «velocity_mesure1» и «velocity_mesure2». Последние блоки вычисляют скорости вращения барабанов по интервалам между импульсами от датчиков «velo_sensor1» и «velo_sensor2».

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 (*Physic inputs*)
0004     high_sensor AT %IX0.6: BOOL; (*I07*)
0005     low_sensor AT %IX0.7: BOOL; (*I08*)
0006     velo_sensor1 AT %IX10.1: BOOL; (*IA1-R2*)
0007     velo_sensor2 AT %IX10.2: BOOL; (*IA4-R3*)
0008     left_sensor AT %IX0.4: BOOL; (*I05*)
0009     right_sensor AT %IX0.5: BOOL; (*I06*)
0010     weight_high AT %IX1.2: BOOL; (*I11*)
0011     weight_low AT %IX1.3: BOOL; (*I12*)
0012     start_button AT %IX0.0: BOOL; (*I01*)
0013     stop_button AT %IX0.1: BOOL; (*I02*)
0014     rotation_up AT %IX0.2: BOOL;
0015     rotation_down AT %IX0.3: BOOL;
0016 (*Physic outputs*)
0017     work_motor AT %QX0.0: BOOL; (*Q01*)
0018     movie_high AT %QX0.1: BOOL; (*Q02*)
0019     movie_low AT %QX0.2: BOOL; (*Q03*)
0020     Q4 AT %QX0.3:BOOL; (*Q04*)
0021     Q5 AT %QX0.4:BOOL; (*Q05*)
0022     Q6 AT %QX0.5:BOOL; (*Q06*)
0023 (*Vars*)
0024     velocity_mesure1: velocity;
0025     velocity_mesure2: velocity;
0026     glide_check: glide;
0027     movie_stage: stage;
0028     movie_stage2: stage;
0029     stage_check: stage_prove;
0030     start_motor: motor;
0031     weight_high_inertial: inertial;
0032     weight_low_inertial: inertial;
0033     M_M: Manage_motor;
0034     PVM: INT;
0035 END_VAR
```

Рис. 38. Переменные корневой программы «PLC_PRG»

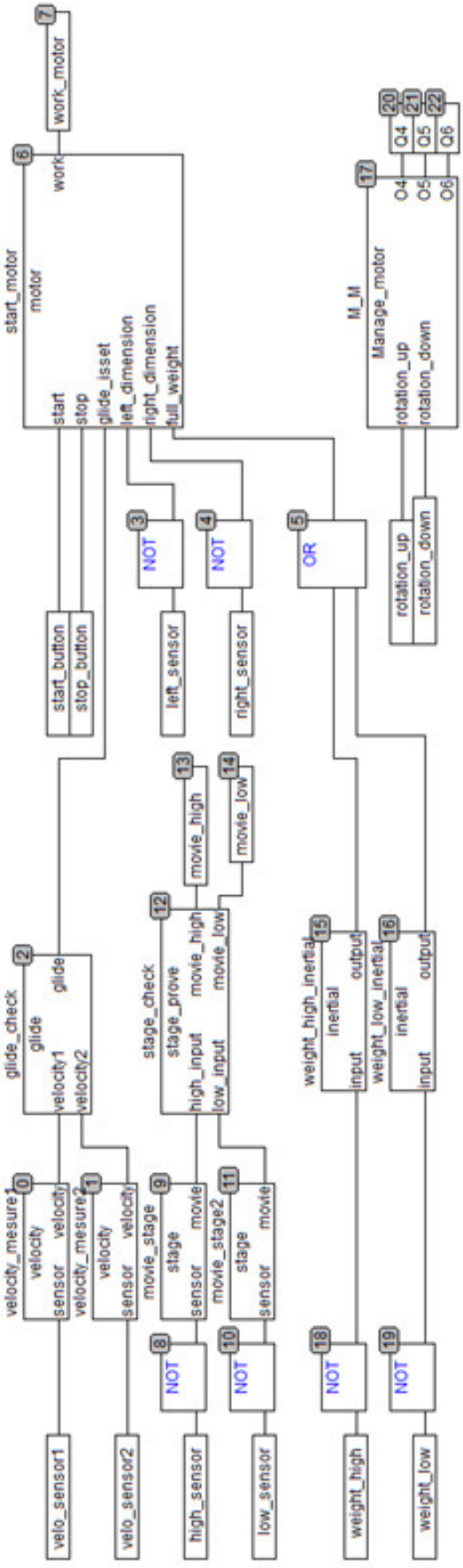

Рис. 39. Корневая программа «PLC_PRG»

Принцип работы данной программы состоит в следующем.

Сигнал о наполнении любой из корзин поступает от переменных «weight_high» и «weight_low» через промежуточные блоки инерционной задержки «weight_high_internal» и «weight_low_internal» на входы аварийного контроля в блок «start_motor». Приведение в движение поворотной площадки осуществляется с помощью блока «stage_check» совместно с блоками «movie_stage» и «movie_stage2». При этом подаются сигналы в переменные «movie_high» и «movie_low», далее с физических выходов %QX0.1 и %QX0.2, связанных с этими переменными, поступают сигналы на привод поворотной площадки сброса грузов.

Сигнал запуска двигателя поступает из переменной «work_motor», связанной с физическим выходом «%QX0.0».

Из переменных «Q4», «Q5», «Q6», связанных с физическими выходами %QX0.3, %QX0.4, %QX0.5, подаётся сигнал на специальные входы преобразователя частоты для задания скорости вращения.

Часть окна программы в верхней левой части показана на рис. 40.

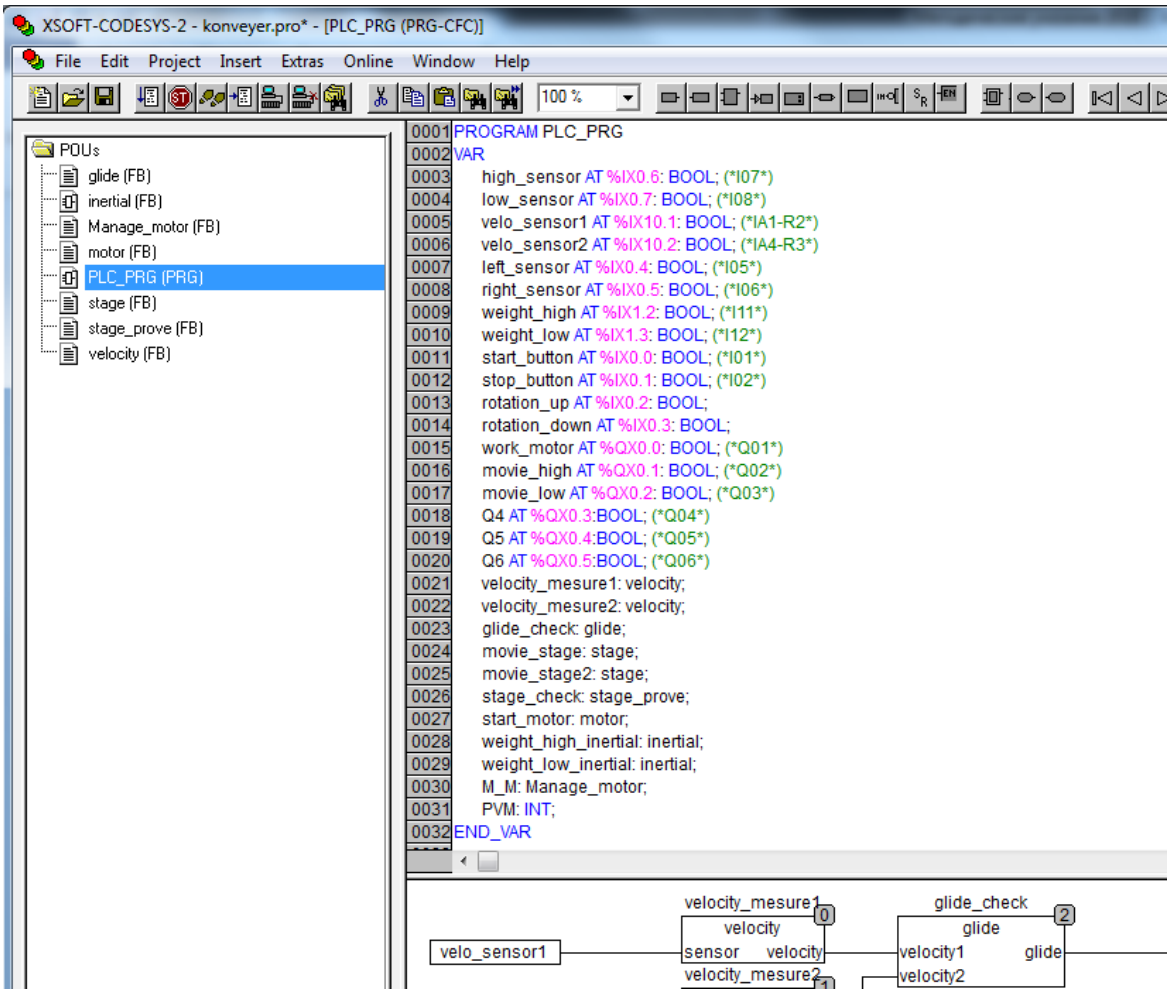

Рис. 40. Окно программы со списком функциональных блоков

4.4. Апробация созданной системы в среде CoDeSys

4.4.1. Проверка работоспособности в режиме эмуляции. Для запуска проекта в режиме эмуляции установите флажок в меню Online/Simulation. Откомпилируйте проект командой Project/Rebuild All. Установите соединение с контроллером Online/Login. Запустите проект Online/Run.

4.4.2. Предварительная подготовка лабораторной установки конвейера.

Соедините необходимые провода между датчиками физической модели и входами «I» контроллера. Также соедините необходимые провода между входами исполнительных элементов физической модели (поворотная площадка, двигатель) и выходами «Q» контроллера. Данные соединения должны производиться в соответствии с назначением переменных физическим входам/выходам в редакторе объявлений PLC_PRG. Естественно объявленные переменные соответствуют определённым датчикам и исполнительным элементам физической модели.

4.4.3. Настройка канала и соединение с реальным контроллером. В меню Online откройте диалог Communication parameters. Нажмите клавишу New для настройки нового соединения. Присвойте ему некоторое осмысленное имя. В простейшем случае CoDeSys SP RTE работает на той же машине (компьютере) что и среда программирования CoDeSys. Это означает, что можно применить способ соединения посредством разделяемой памяти (Shared memory (Kernel)). Если контроллер расположен на другой машине сети, необходимо изменить параметр «localhost» на имя машины или задать соответствующий IP адрес. Настройку нужно подтвердить клавишей OK.

Соединение с контроллером устанвливается командой Online/Login из среды программирования CoDeSys. Если используется удалённое соединение, CoDeSys попросит вас подтвердить загрузку (download) кода проекта.

Команда Online/Run запускает проект.

4.5. Создание системы управления в среде Easy Soft

Разработка программной системы управления в среде Easy Soft – это разработка уже совсем на другом языке программирования и среде проектирования. Общее описание конвейерной установки уже имеется в начале данного раздела. Язык программирования в данном случае **«LD»** – лестничная диаграмма. Чем-то напоминающий релейную логику.

4.5.1. Создайте новый проект и добавьте в него устройство – программируемое реле EASY серии 800. В данном случае это EASY-820-DC-RC. Версия устройства 7.

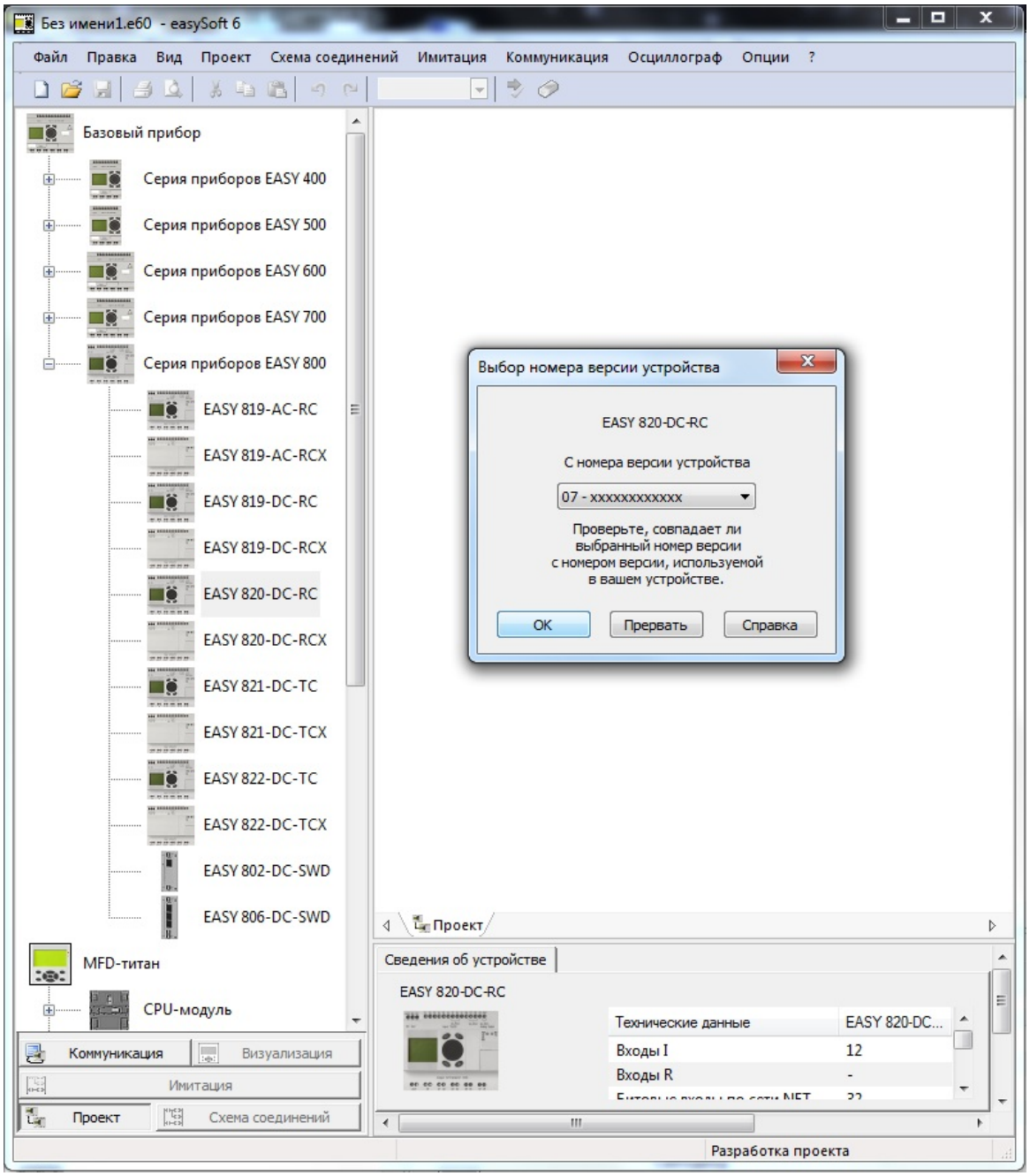

Рис. 41. Добавление нового устройства EASY 820-DC-RC в проект

4.5.2. После добавления устройства, справа внизу откройте его свойства и вкладку «системные настройки». Поставьте галочку «Р-кнопки» (рис. 42). Это необходимо для того, чтобы можно было управлять работой программы с помощью кнопок, расположенных непосредственно на самом реле.

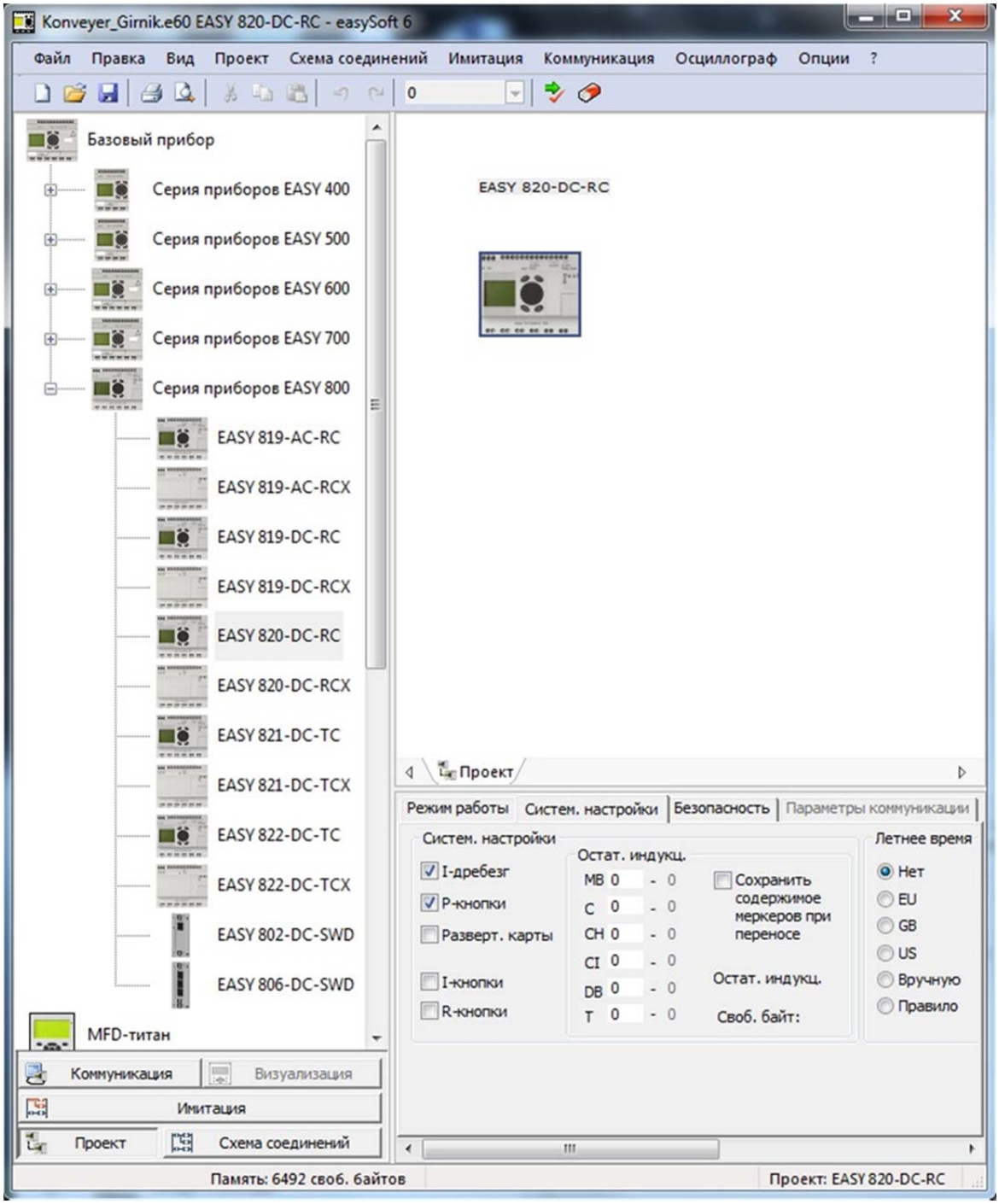

Рис. 42. Системные настройки устройства в проекте

4.5.3. Нажмите слева внизу кнопку «Схема соединений» и вы перейдёте в окно, как на рис. 43, для рисования схемы управления конвейером. Слева расположена панель с модулями для проектирования

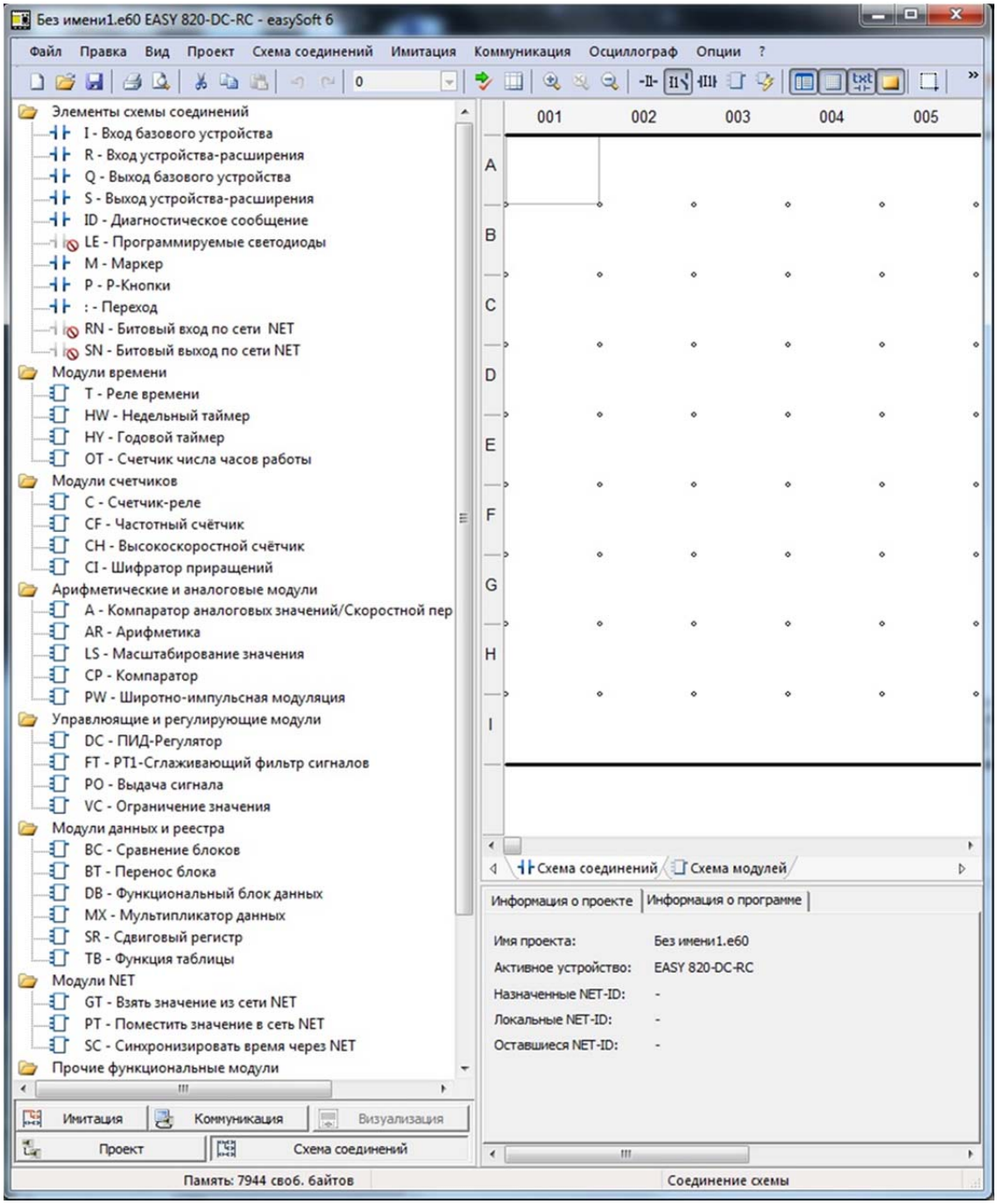

Рис. 43. Рабочая область для схемы соединений

4.5.4. Используя модули, нарисуйте схему запуска и сортировки, как на рис. 44–45.

В схеме рис. 44, пуск и останов двигателя конвейера производится с помощью подачи сигналов на входы реле «I01» и «I02», либо с помощью P-кнопок «P03» и «P01». Такая неравномерная последовательность номеров P-кнопок обусловлена их спецификой расположения и заводской настройкой по номерам. Дело в том, что в данном проекте подра-

зумевается, что за пуск отвечает кнопка со стрелкой «вверх», а за останов кнопка со стрелкой «вниз».

При перекрытии сигналов датчиков левого или правого габаритов, данные сигналы, а именно их отсутствие, размыкают цепь питания контактора «Q01» с помощью контактов «I05» или «I06».

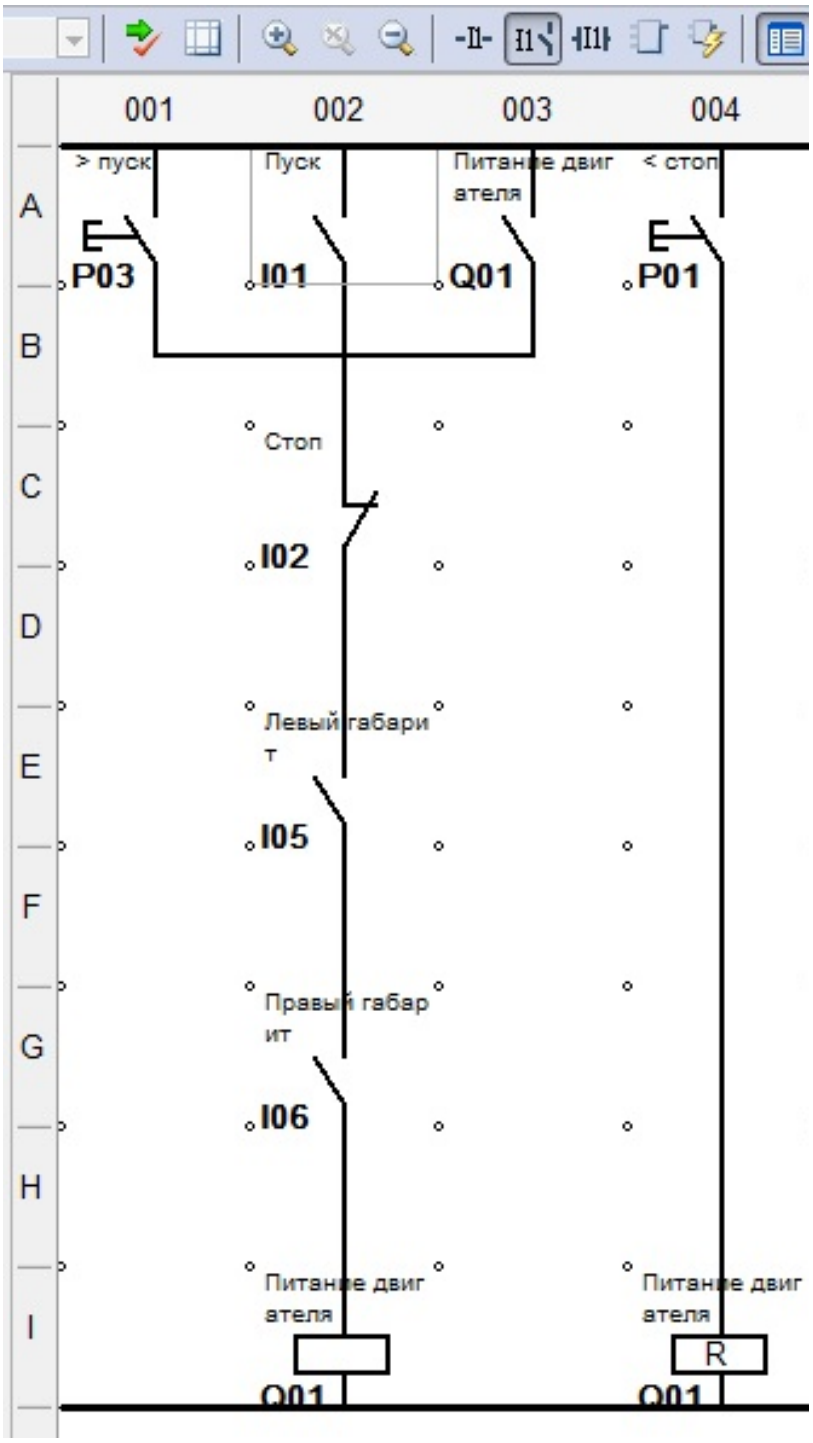

Рис. 44. Схема запуска

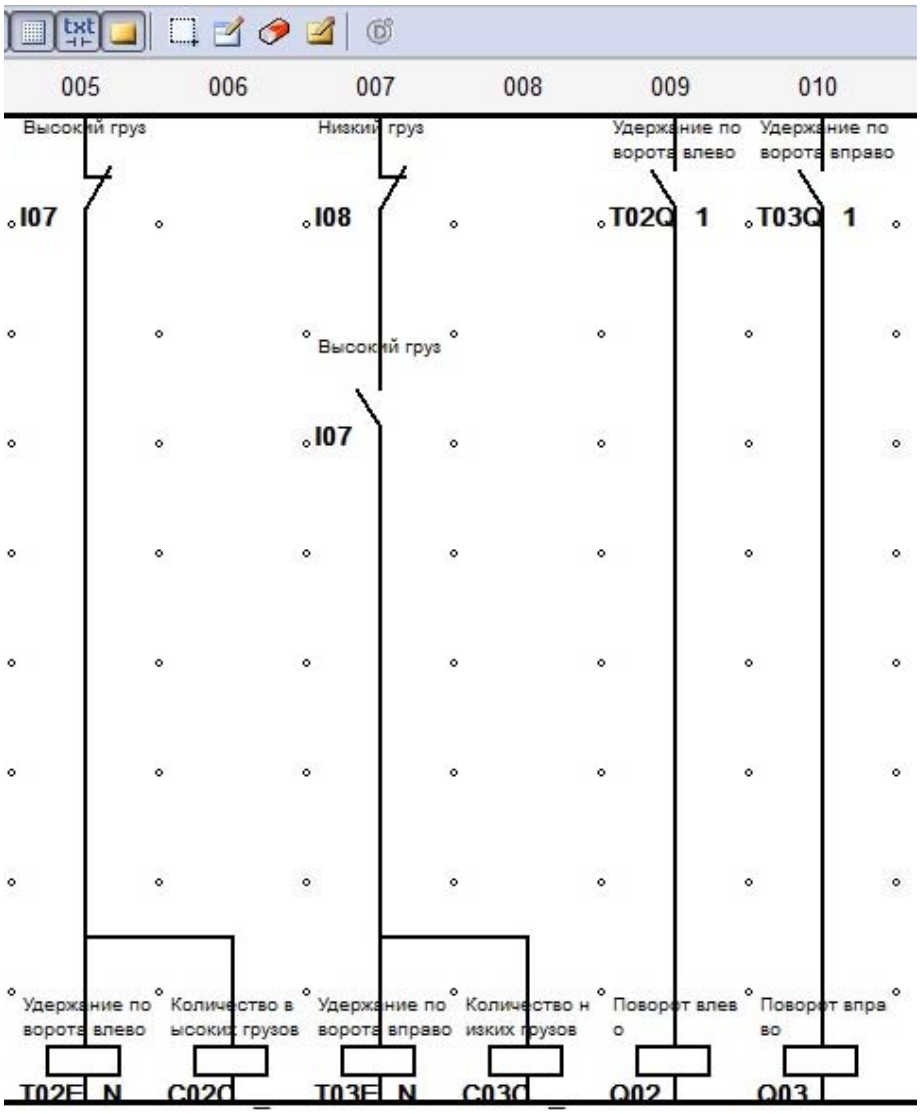

Рис. 45. Схема сортировки

В схеме рис. 45 сигналы перекрытия уровней высокого и низкого грузов поступают на входы «I07» и «I08». При этом, данные сигналы запускают таймеры задержки отключения «T02E» и «T03E» чтобы удержать питание контакторов «Q02» и «Q03», которые в свою очередь подают питание на соленоиды привода поворотной площадки для сброса грузов по корзинам. Это нужно, чтобы пока зафиксированный груз проходит датчики уровней и начинает падать на площадку, последняя ещё должна какое-то время быть повёрнута в нужную корзину, чтобы груз успел упасть. Ведь здесь будет присутствовать такой момент времени, когда падающий груз уже не перекрывает датчики уровня и их сигналы на входы «I07» и «I08» снова поступают.

Кроме того, вместе с таймерами, срабатывают счётчики «C02C» и «C03C» для отдельного подсчёта количества высоких и низких грузов.

Настройки модулей, применённых в схеме на рис. 44 и 45, показаны на рис. 46–53.

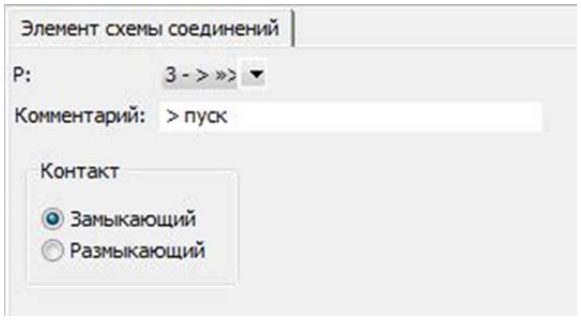

Рис. 46. Настройки P-кнопки для пуска

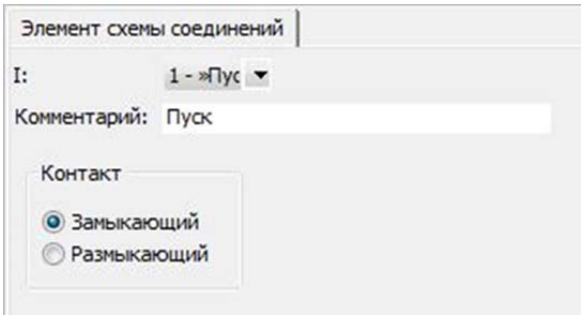

Рис. 47. Настройки 1-го входа «I»

Остальные входы «I» настраиваются подобным образом, за исключением того, что в некоторых местах схемы это будут размыкающие контакты.

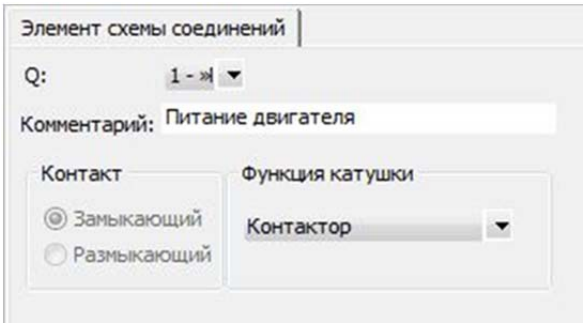

Рис. 48. Настройки контактора «Q1»

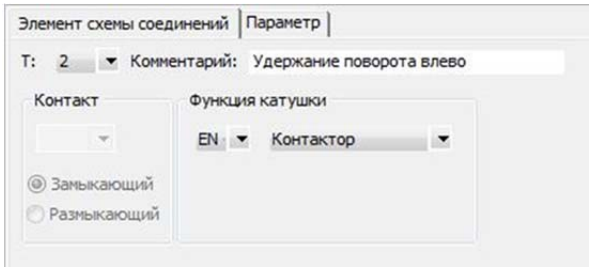

Рис. 49. Настройки 2-го таймера «T02E»

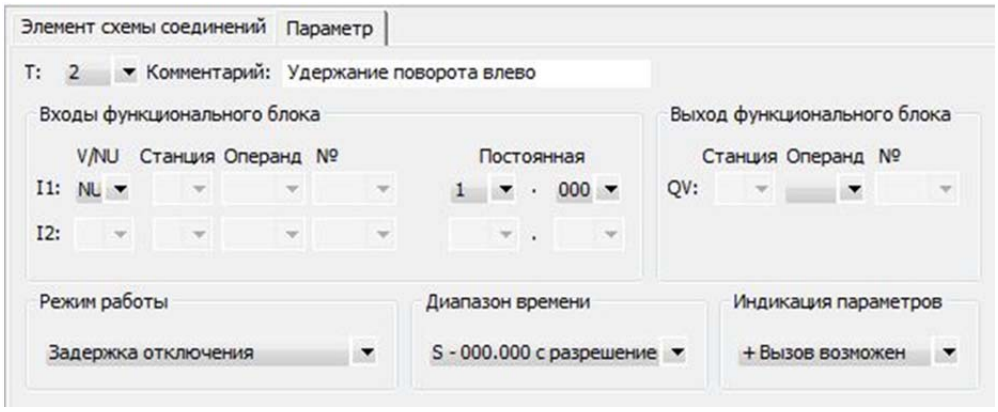

Рис. 50. Параметры 2-го таймера «T02E»

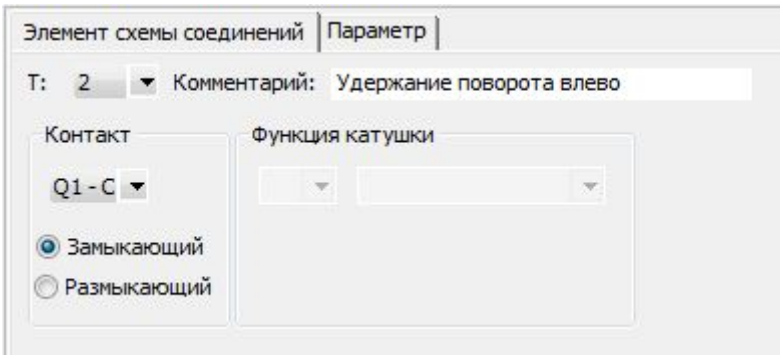

Рис. 51. Настройки контакта «T02Q1» 2-го таймера

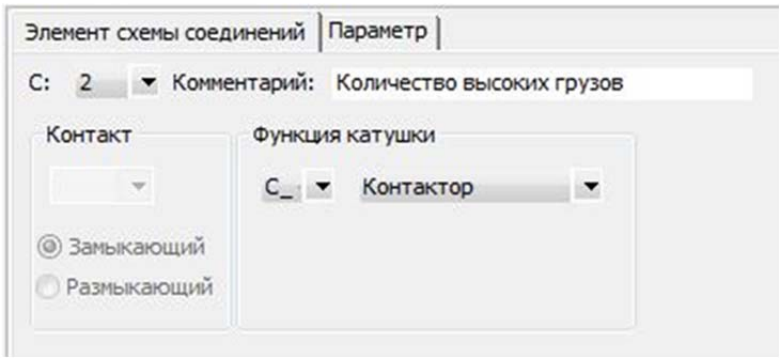

Рис. 52. Настройки счётчика «C02C» подсчёта высоких грузов

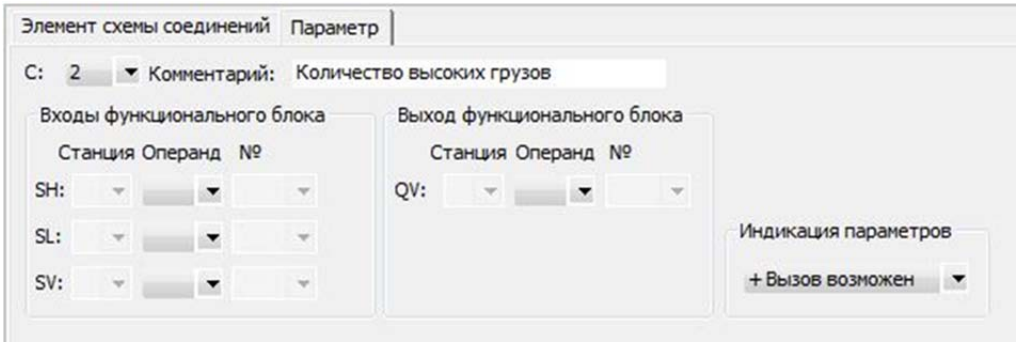

Рис. 53. Параметры счётчика «C02C» подсчёта высоких грузов

Настройки и параметры счётчика «C03C» для подсчёта низких грузов, делаются аналогично.

4.5.5. Нарисуйте схему управления скоростью двигателя, как на рис. 54 (11 и 12 поля схемы пустые).

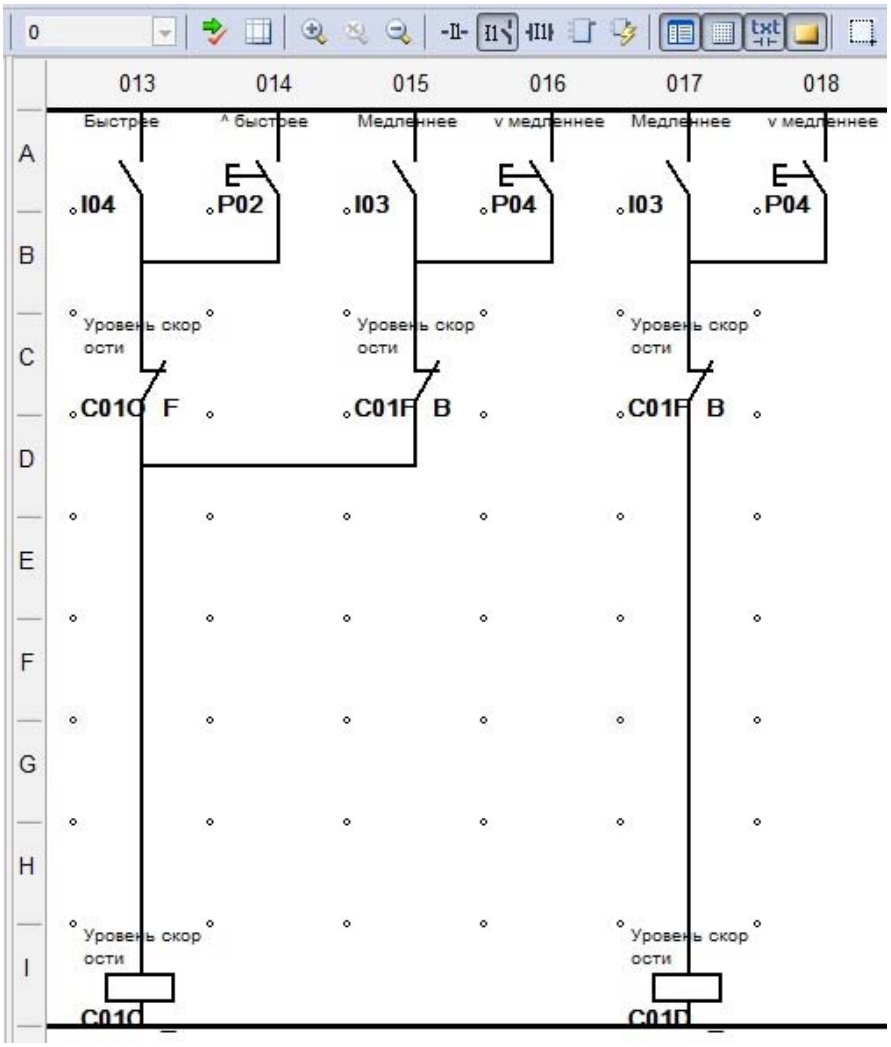

Рис. 54. Схема управления скоростью двигателя конвейера

Настройки модулей, применённых в схеме на рис. 54, показаны на рис. 55–59.

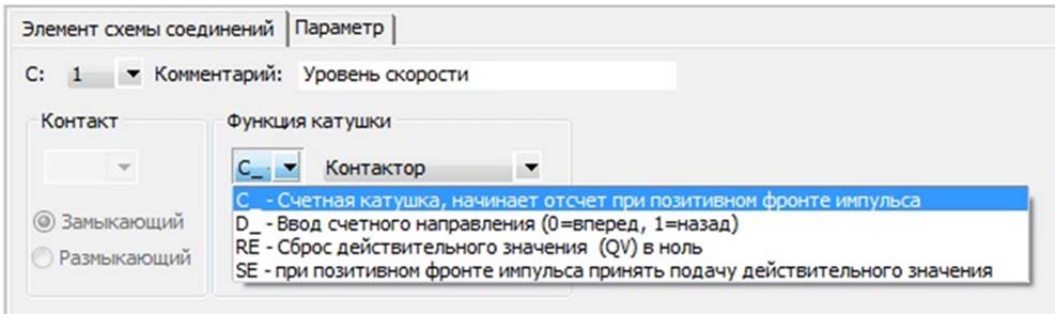

Рис. 55. Настройки счётчика «C01C» для подсчёта в прямом направлении

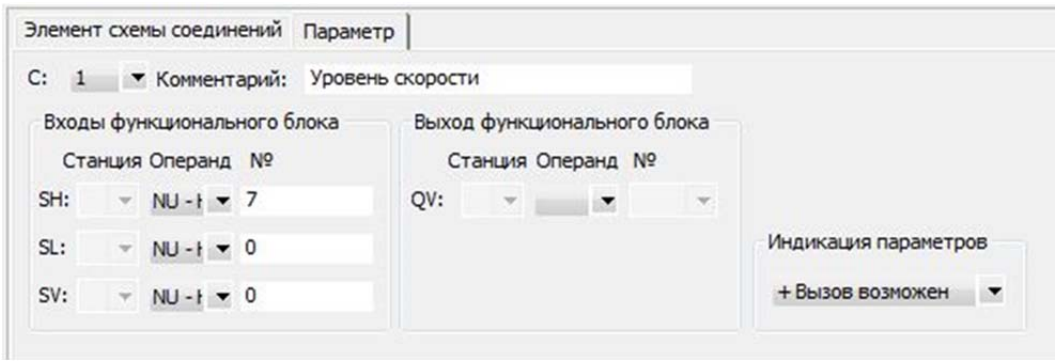

Рис. 56. Параметры счётчика «C01C»

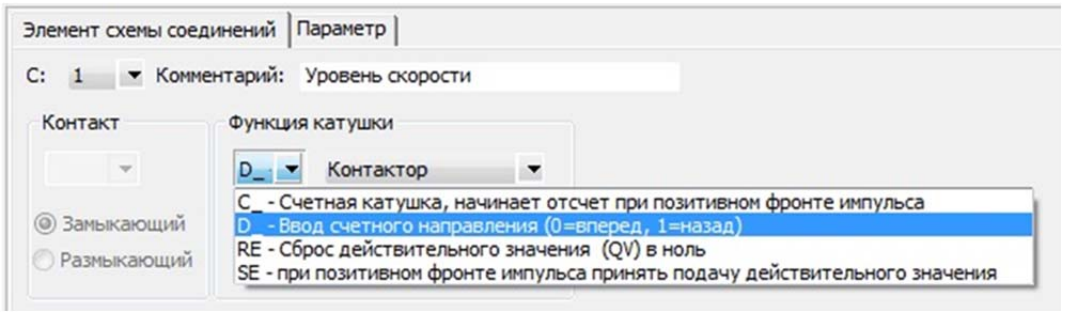

Рис. 57. Настройки счётчика «C01D» для задания счёта в обратном направлении

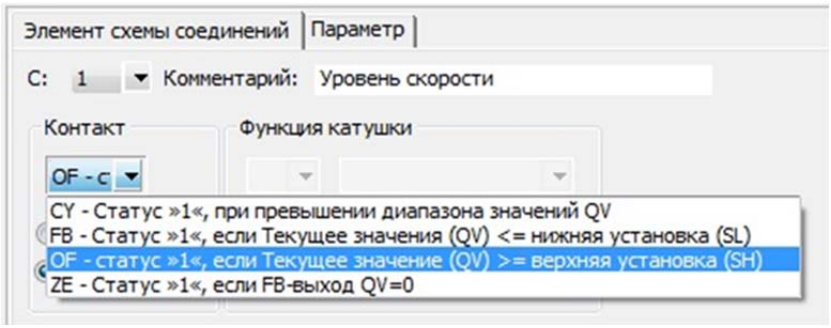

Рис. 58. Настройки ограничителя счёта для контакта «OF» счётчика «C01C»

При достижении текущего значения счётчика «C01C» равного пределу 7, контакт «OF» размыкает цепь питания самого счётчика прямого счёта «C01C».

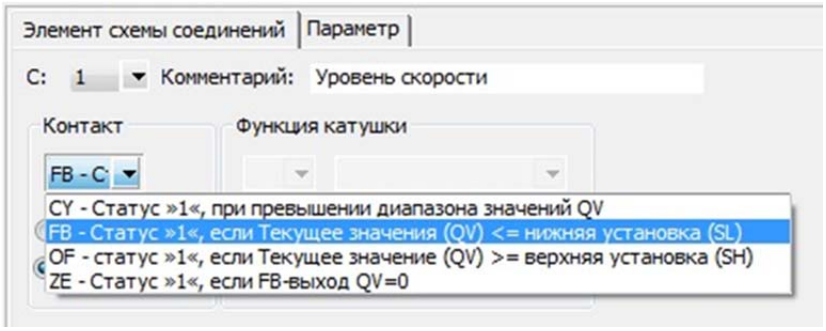

Рис. 59. Настройки ограничителя счёта для контакта «FB» счётчика «C01C»

При снижении текущего значения счётчика «C01C» до минимального значения 0, контакт «FB» блокирует питание основного счётчика «C01C» и его инвертора направления счёта «C01D».

Текущее значение счётчика «C01C» необходимо преобразовать в двоичный вид из 3-х разрядов. Для этого будем использовать компараторы «CP», которые в зависимости от значения счётчика будут включать нужные контакторы «Q4», «Q5» и «Q6». Каждый контактор соответствует своему разряду для формирования двоичного кода, который в виде трёх отдельных сигналов подаётся на входы преобразователя частоты (ПЧ), управления скоростью вращения двигателя (рис. 29).

На рис. 60–62 показана схема преобразования уровня скорости в электрические сигналы в виде 3-х разрядного двоичного кода.

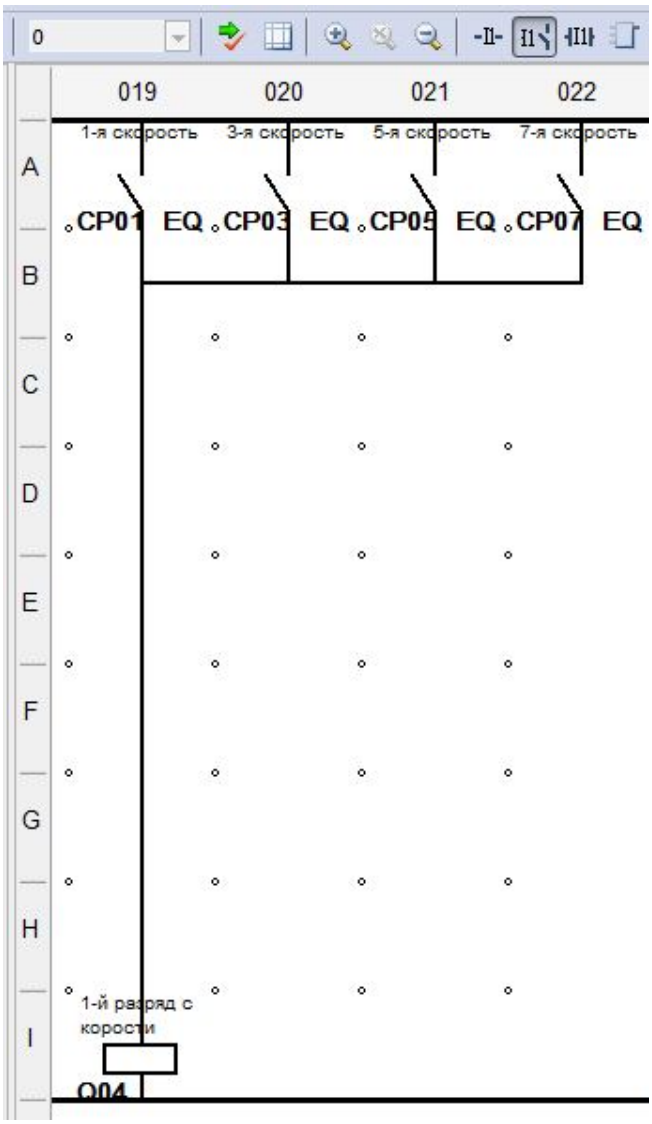

Рис. 60. Схема получения 1-го разряда двоичного кода скорости на базе счётчика номера скорости

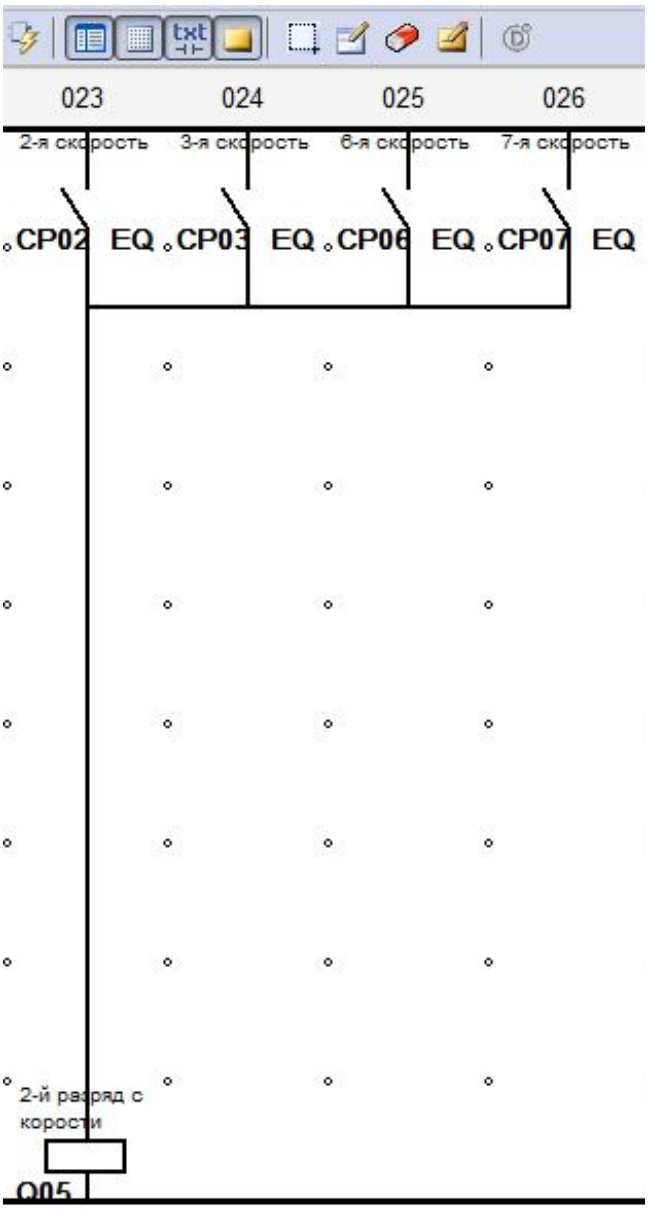

Рис. 61. Схема получения 2-го разряда двоичного кода скорости на базе счётчика номера скорости

Примеры настроек и параметров для компаратора «CP01» показаны на рис. 63, 64. Этот компаратор замыкает свои контакты при условии, если уровень скорости равен 1. Это достигается с помощью задания параметров:

«I1» – сравниваемый сигнал, который получает выходное значение «QV» от счётчика «C01»;

«I2» – эталон, с которым сравнивается «I1», при равенстве с которым компаратор замыкает контакты. В данном случае равен 1 – номеру первой скорости.

Компараторы «CP02-CP07» настраиваются аналогично, но условие срабатывания у них задаётся для «I2» со значением их номера скорости.

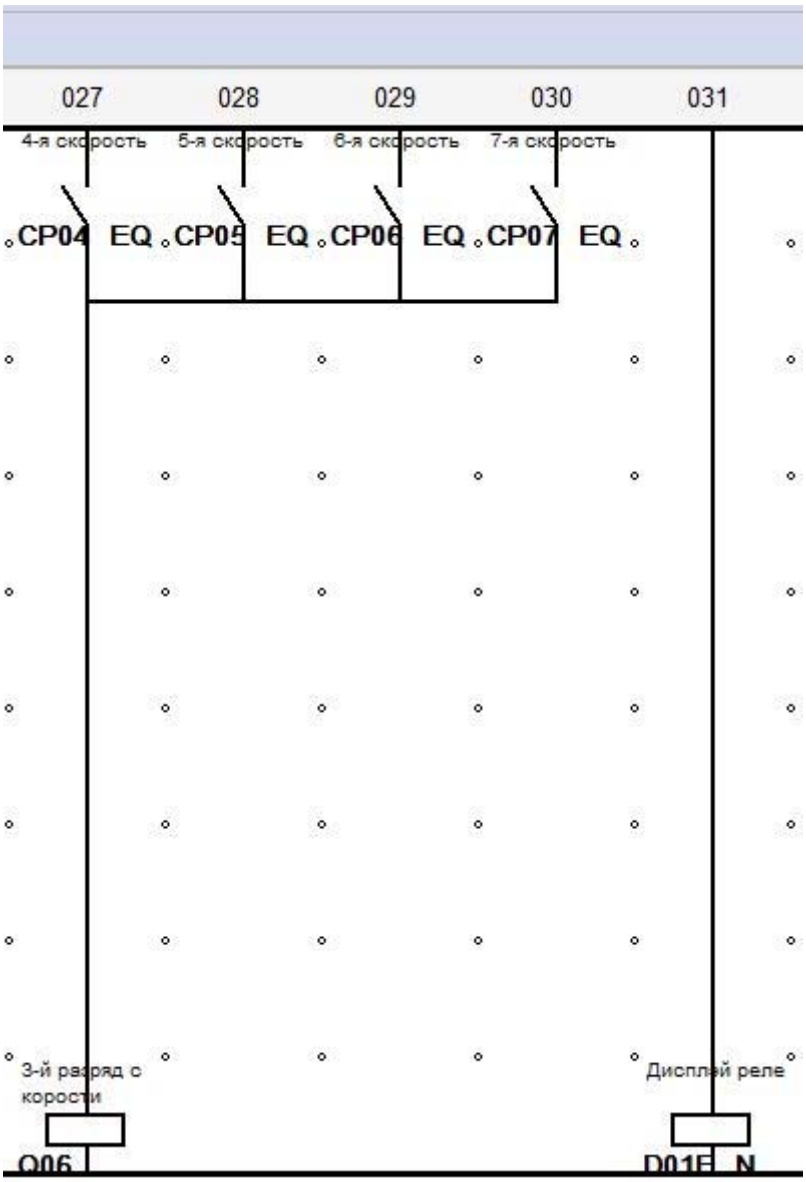

Рис. 62. Схема получения 3-го разряда двоичного кода скорости на базе счётчика номера скорости

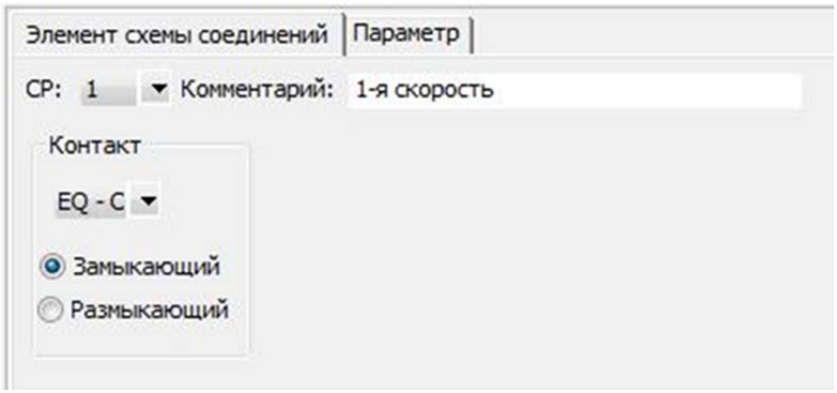

Рис. 63. Настройки компаратора «CP01»

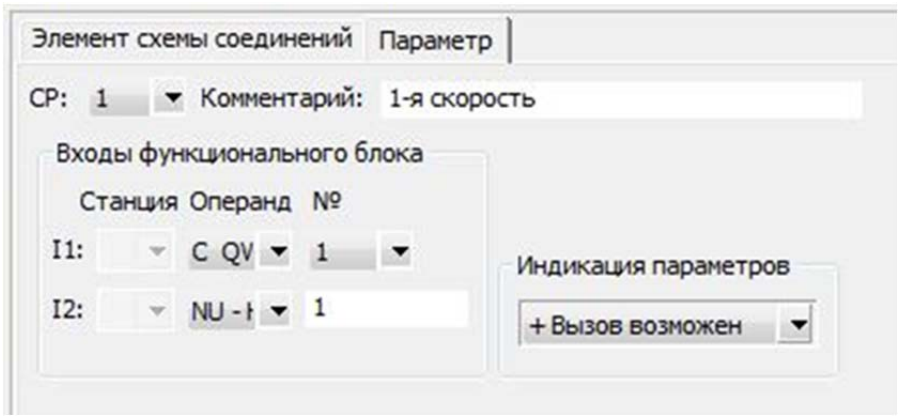

Рис. 64. параметры компаратора «CP01»

Для отображения некоторых параметров и их величин сразу на живом дисплее реле, существует специальный модуль «D». На рис. 62 это модуль «D01». Его настройки и параметры показаны на рис. 65–68. Данный модуль настроен для отображения количества высоких и низких грузов, а также их веса.

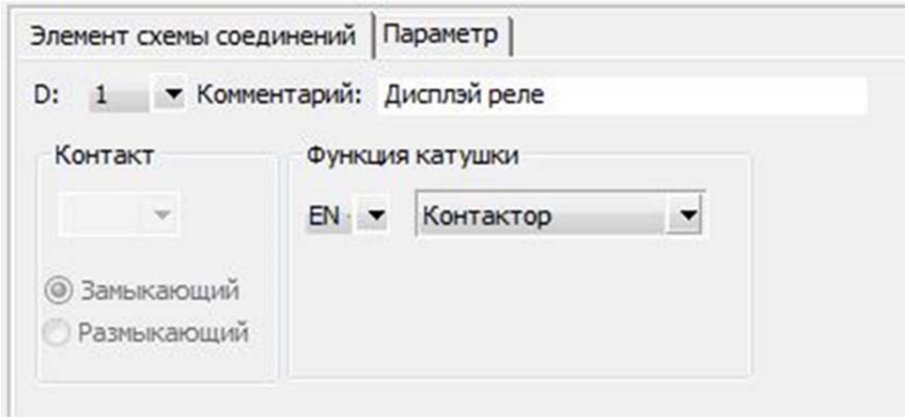

Рис. 65. Настройки модуля «D01»

Рис. 66. Параметры модуля «D01»

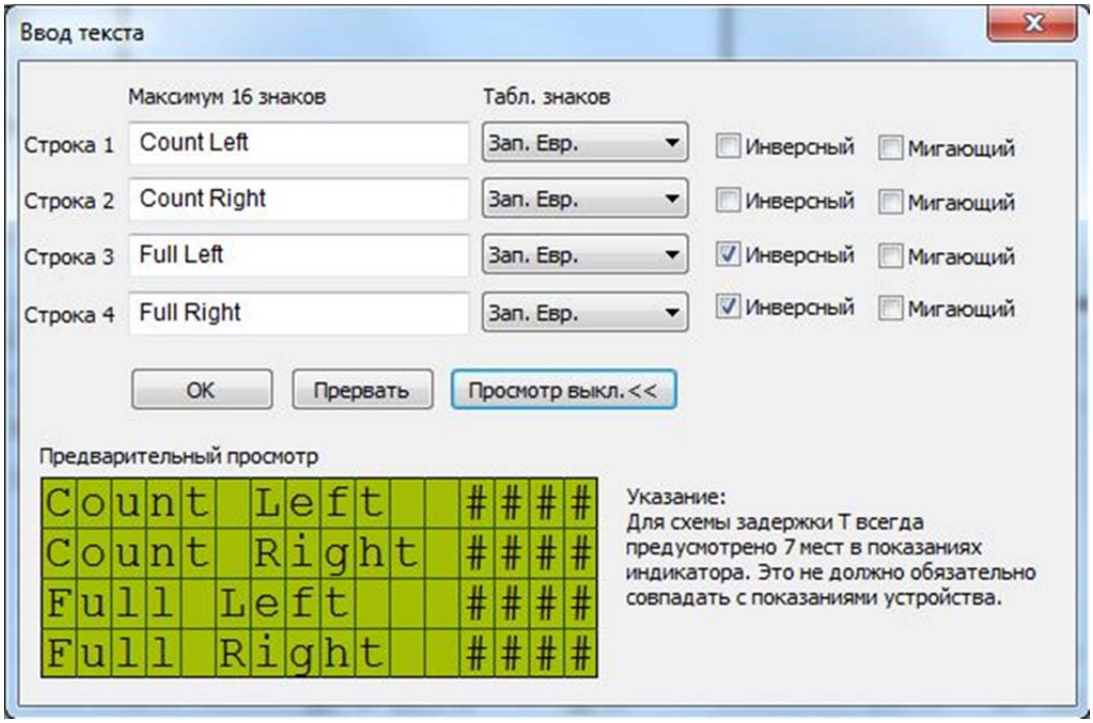

Рис. 67. Ввод текста модуля «D01»

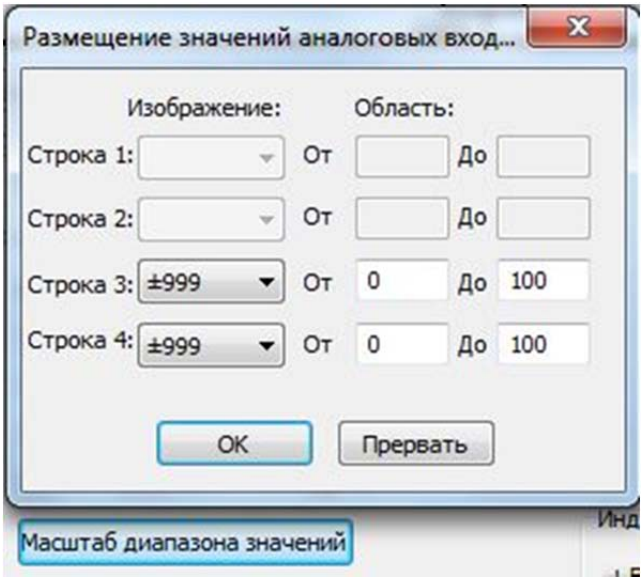

Рис. 68. Масштаб диапазона значений

4.6. Апробация созданной системы в среде Easy Soft

После создания программы в среде Easy Soft, программный код нужно записать внутрь процессора электронного реле. Чтобы это сделать, нужна коммуникация компьютера с электронным реле. Для этого предусмотрено устройство расширения Easy209-SE, которое физически подсоединено к самому реле на стенде. А уже компьютер подсоединяется с помощью витой пары к Easy209-SE.

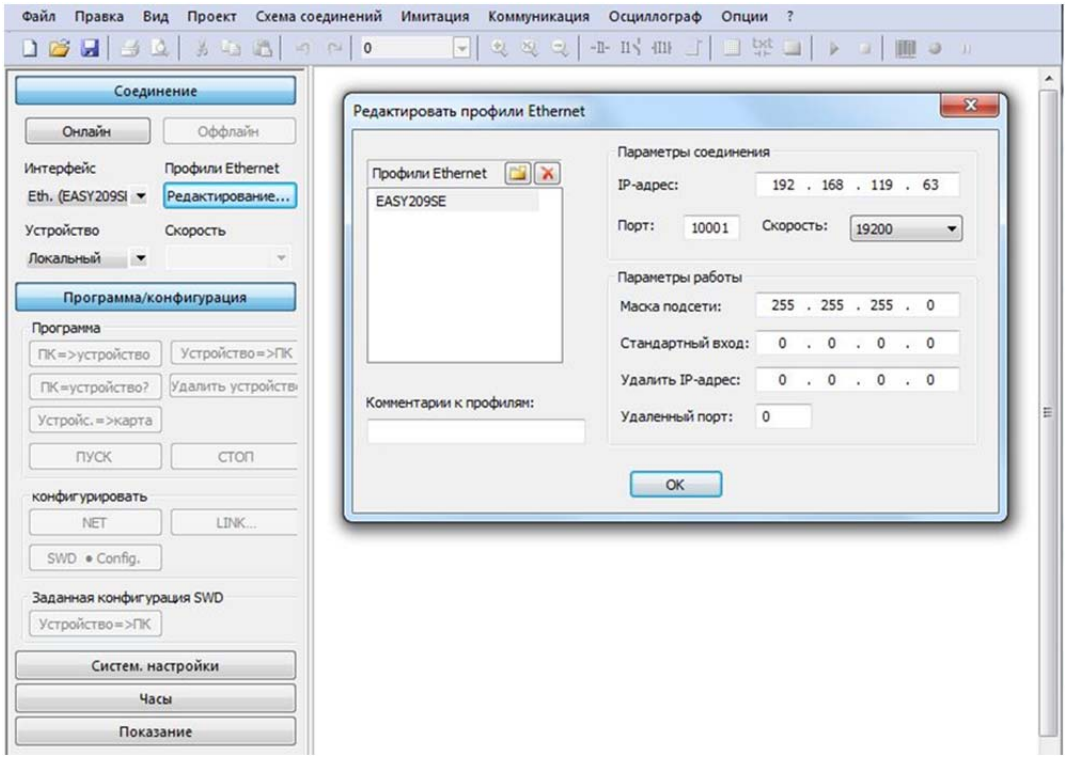

Рис. 69. Настройки коммуникации с реле

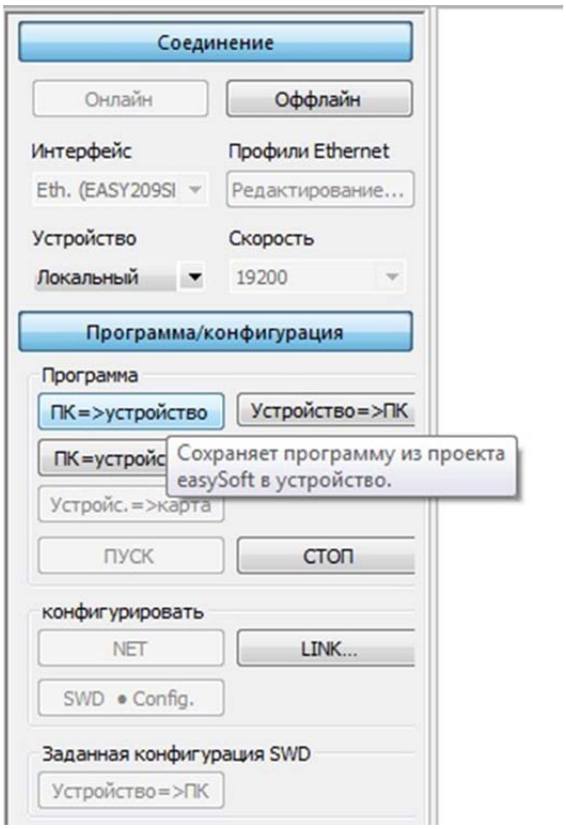

Рис. 70. Сохранение программы из персонального компьютера в процессор электронного реле

Чтобы настроить коммуникацию по протоколу Ethernet с электронным реле через его устройство расширения, нажмите внизу слева кнопку «Коммуникация» среды Easy Soft и перейдите в окно, вид которого показан на рис. 69. Откройте окно редактирования интерфейса, создайте новую настройку и назовите её например «EASY209SE». Чтобы узнать какие данные подключения вводить (ip адрес, маска подсети), нужно воспользоваться программой «Easy-209SE Configurator». Программа интуитивно проста. Она автоматически определит нужные параметры.

Когда настроены параметры подключения, нажмите кнопку «Онлайн». Далее, кнопку «ПК => устройство» чтобы записать программу в процессор реле, рис. 70. Можно дистанционно запустить реле, нажав кнопку «Пуск».

5. СИСТЕМА УПРАВЛЕНИЯ ГАРАЖНОЙ ЖАЛЮЗИ

Цель: освоить принципы построения интеллектуальной системы для управления физической моделью гаражной жалюзи (роль-ставни).

5.1. Порядок работы

5.1.1. Ознакомиться с лабораторной установкой жалюзи.

5.1.2. Создать систему управления физической моделью на базе программного языка LD (среда EasySoft).

5.1.3. Апробация созданной системы в среде EasySoft.

5.2. Описание лабораторной установки

Лабораторная установка гаражной жалюзи представляет собой конструкцию с поднимаемой лентой вдоль вертикальных направляющих с помощью асинхронного двигателя (рис. 71).

На пути поднятия лента встречает два индукционных датчика положения ED1 и ED2. Эти датчики замыкают свои контакты при подходе к ним специальной металлической пластины, расположенной на самой ленте.

Для контроля крайних положений ленты, т. е. закрытого и открытого состояний, предусмотрены конечные выключатели KV1 и KV2.

Для обеспечения безопасности при закрытии (наличие посторонних предметов или человека под лентой) предусмотрен датчик движения DD1, который может выдавать сигнал для остановки закрывающейся ленты.

Для автоматического закрывания при наступлении тёмного времени суток, предусмотрен датчик освещения DO1.

Для того чтобы предотвратить несанкционированный доступ к управлению, предусмотрен поворотный ключ KM1. Этот ключ представляет собой замочную скважину, в которую необходимо вставить ключ для поворота, чтобы тем самым замкнулись специальные контакты, разрешающие работу всей установки.

Принципиальная электрическая схема лабораторной установки приведена на рис. 72. Эта модель соединяется с лабораторным стендом посредством специальных кабелей через разъёмы типа «DB-15» – для выходов сигналов и «DB-25» – для входов управления. Само управление производится с помощью кнопок стенда, сигналы которых поступают через специальные соединительные кабели на входы электронного реле. На такие же входы (только с другим номером) поступают и сигналы с датчиком модели. Выходные сигналы управления с реле с выходов Q поступают на входы модели.

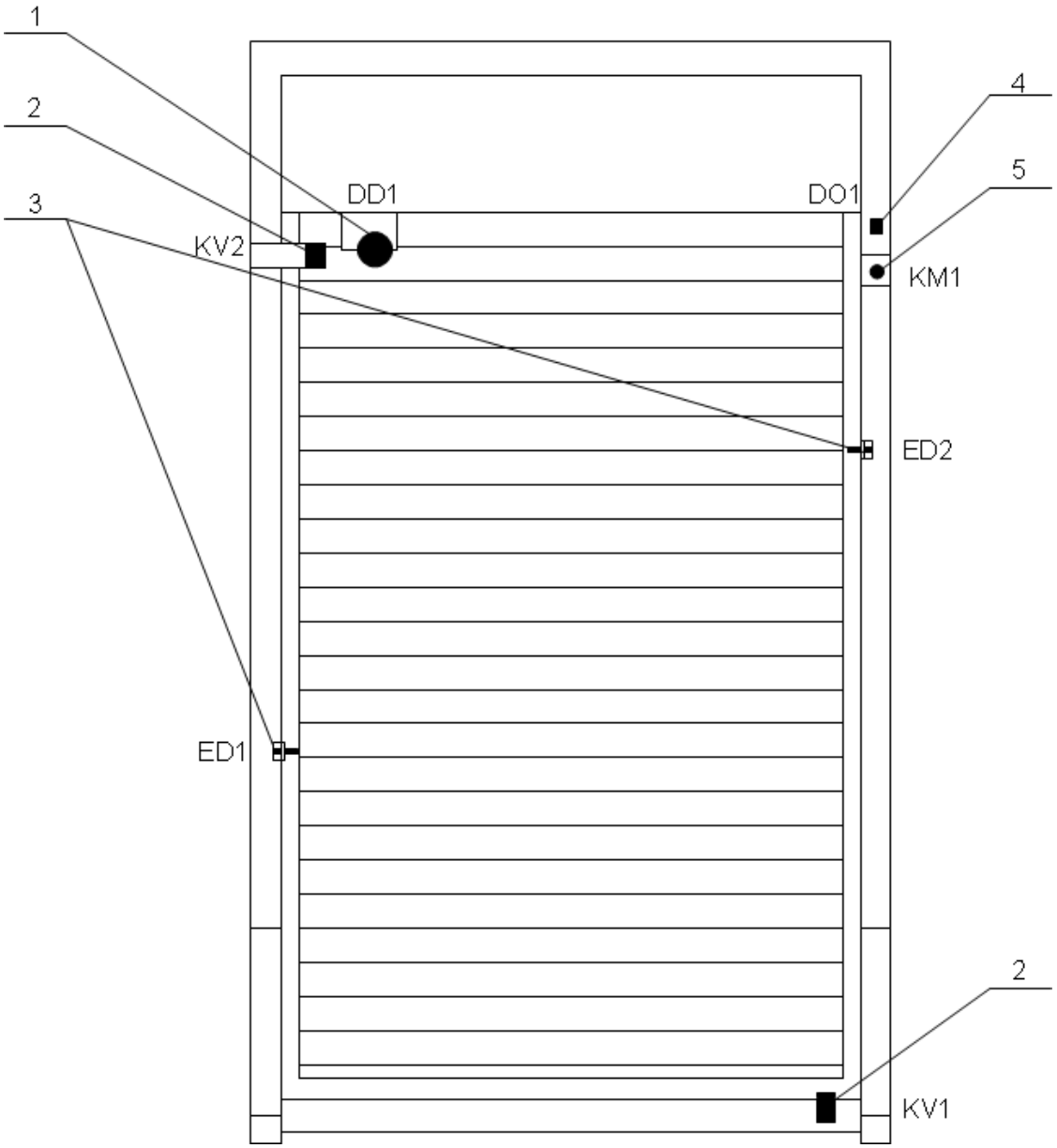

Рис. 71. Вид модели «Жалюзи» с расположением навесного оборудования: 1 – датчик движения; 2 – конечные выключатели; 3 – датчики промежуточного положения; 4 – датчик освещения; 5 – поворотный ключ

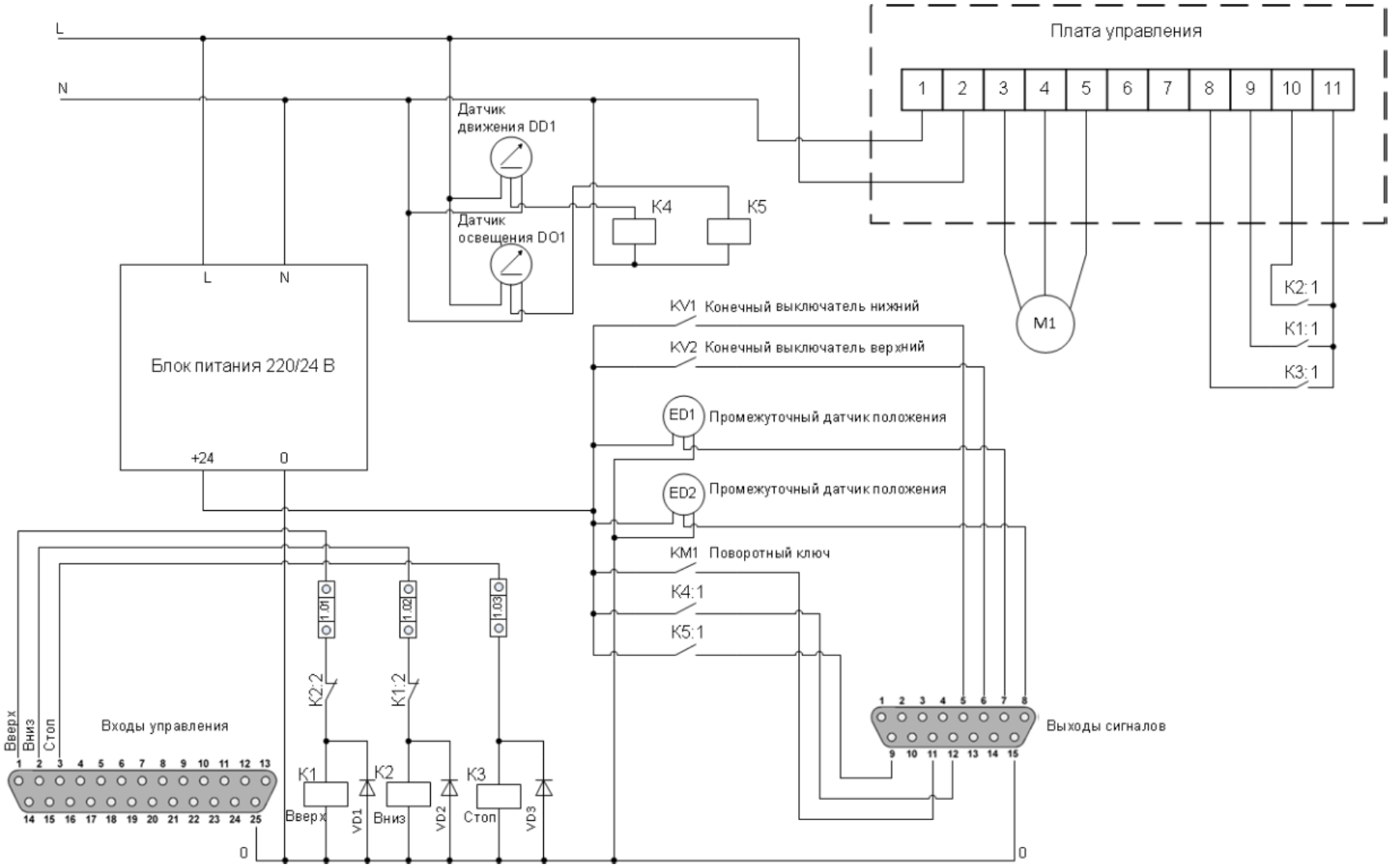

Рис. 72. Принципиальная электрическая схема модели «Жалюзи»

5.3. Создание системы управления в среде Easy Soft

5.3.1. Создайте новый проект и добавьте в него устройство – программируемое реле EASY серии 800. В данном случае это EASY-820-DC-RC. Версия устройства 7.

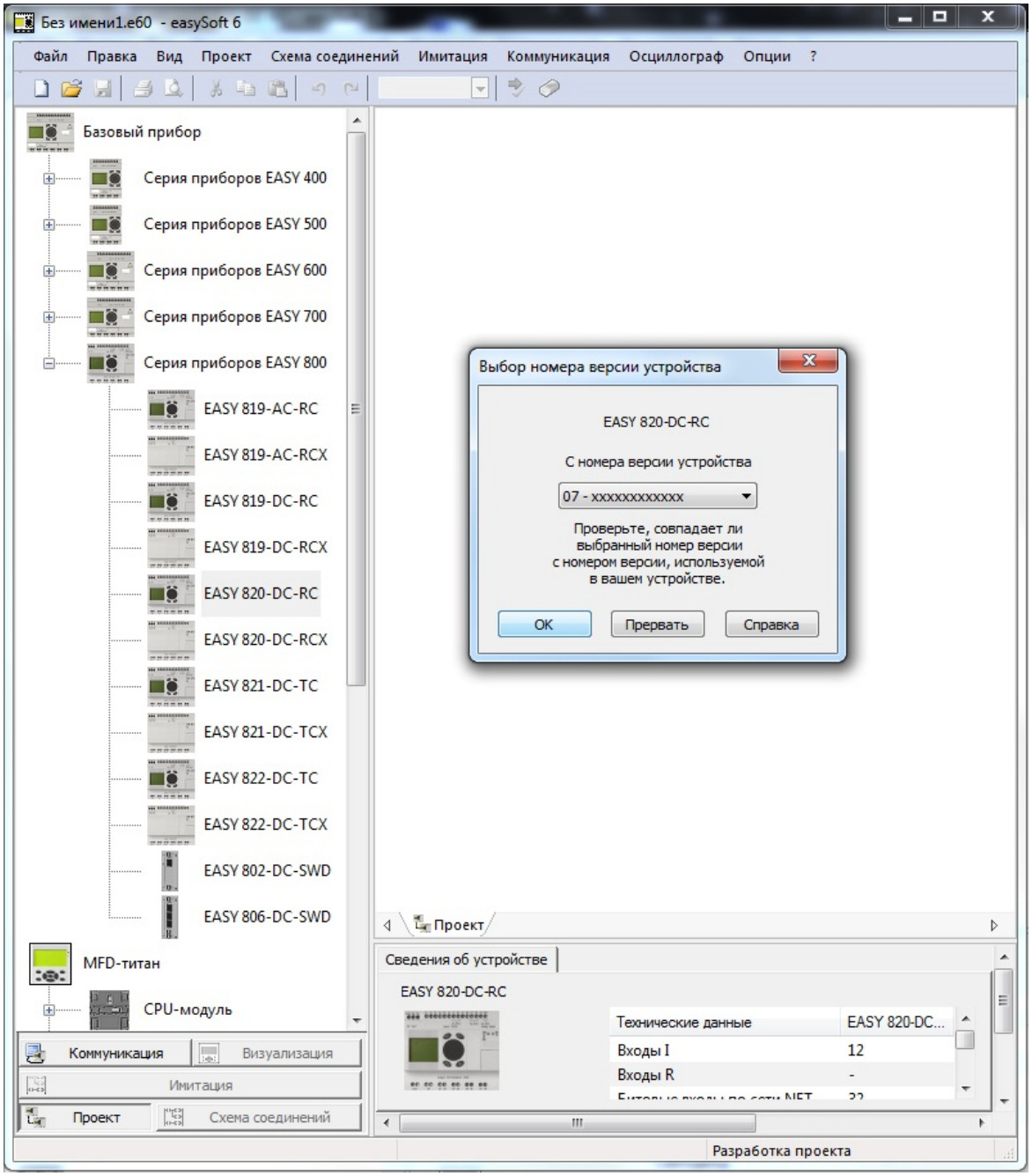

Рис. 73. Добавление нового устройства EASY 820-DC-RC в проект

5.3.2. Нажмите слева внизу кнопку «Схема соединений» и вы перейдёте в окно, как на рис. 74, для рисования схемы управления жалюзи. Слева расположена панель с модулями для проектирования.

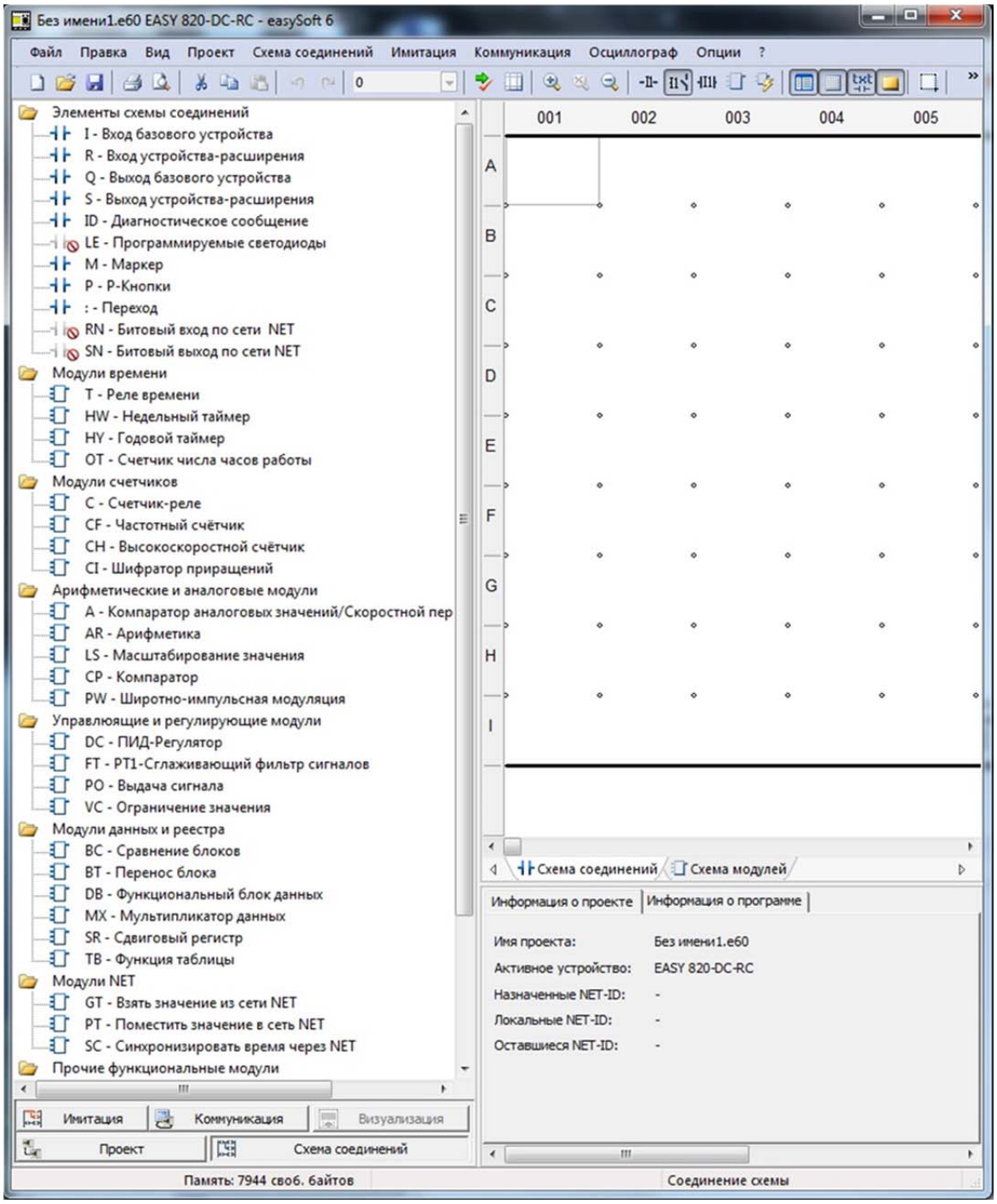

Рис. 74. Рабочая область для схемы соединений

5.3.3. Используя модули, нарисуйте схему управления открытием и закрытием жалюзи, как на рис. 75. А также схему управления остановом и открытием ночью, как на рис. 76.

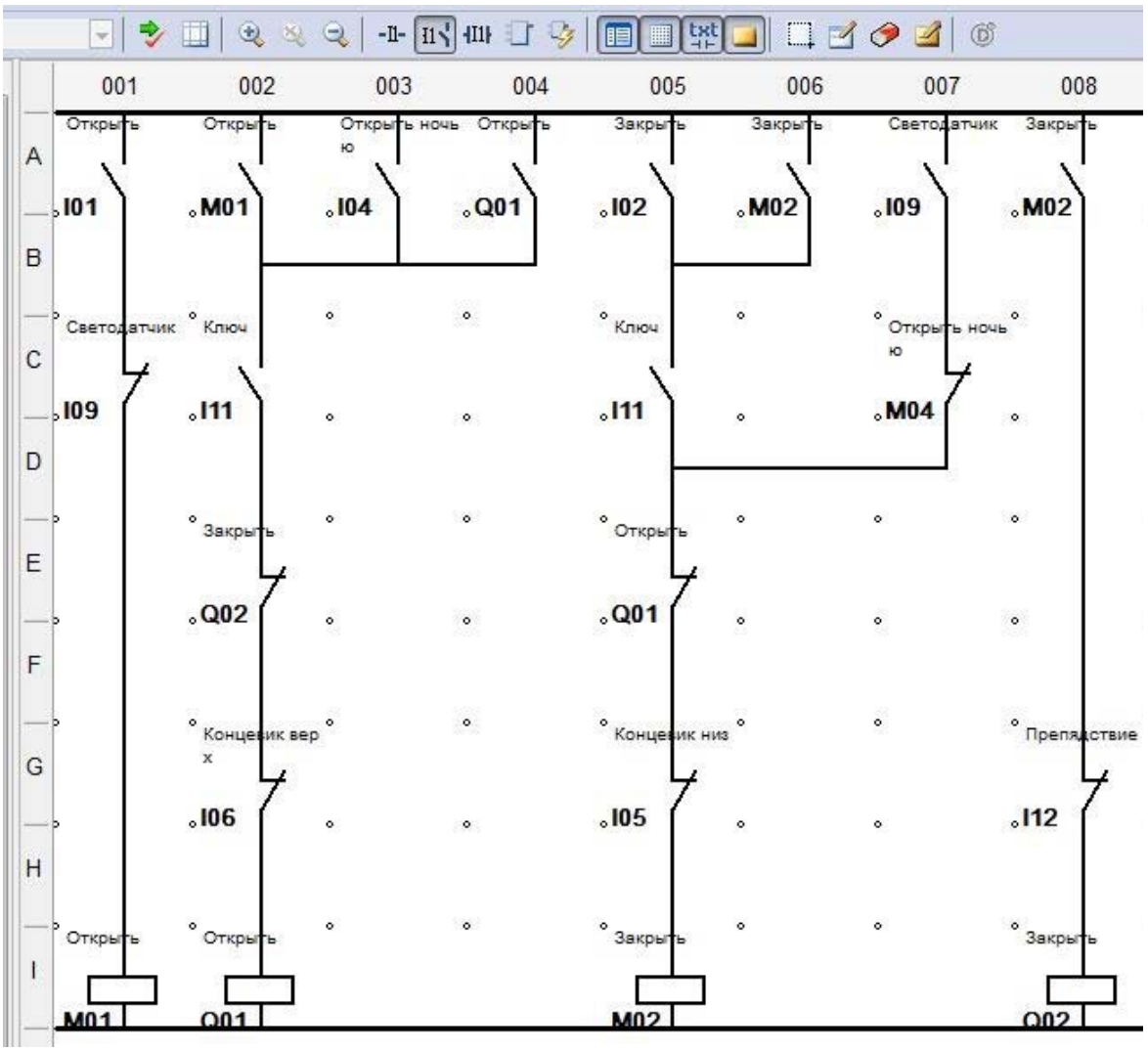

Рис. 75. Схема управления открытием и закрытием жалюзи

На рис. 75 контакторы Q1 и Q2 управляются через промежуточные маркеры и имеют взаимную блокировку с помощью своих перекрёстных размыкающих контактов Q01 и Q02. В роли поворотного ключа «KM1» (рис. 71) служат замыкающие контакты I11.Нижний и верхний конечные выключатели – соответственно I05 и I06. Сигнал с датчика движения «DD1» (рис. 71) поступает на вход I12 в качестве сигнала препятствия.

Положение контактов I09 светодатчика показано для случая, когда светлое время суток.

При наступлении тёмного времени суток контакты I09 переключаются. В цепи питания M02 подготавливают его питание. И если лента открыта, то нижний конечный выключатель I05 замкнут. Таким образом, создаётся цепь питания для маркера M02. Он срабатывает и своими контактами питает цепь Q02.

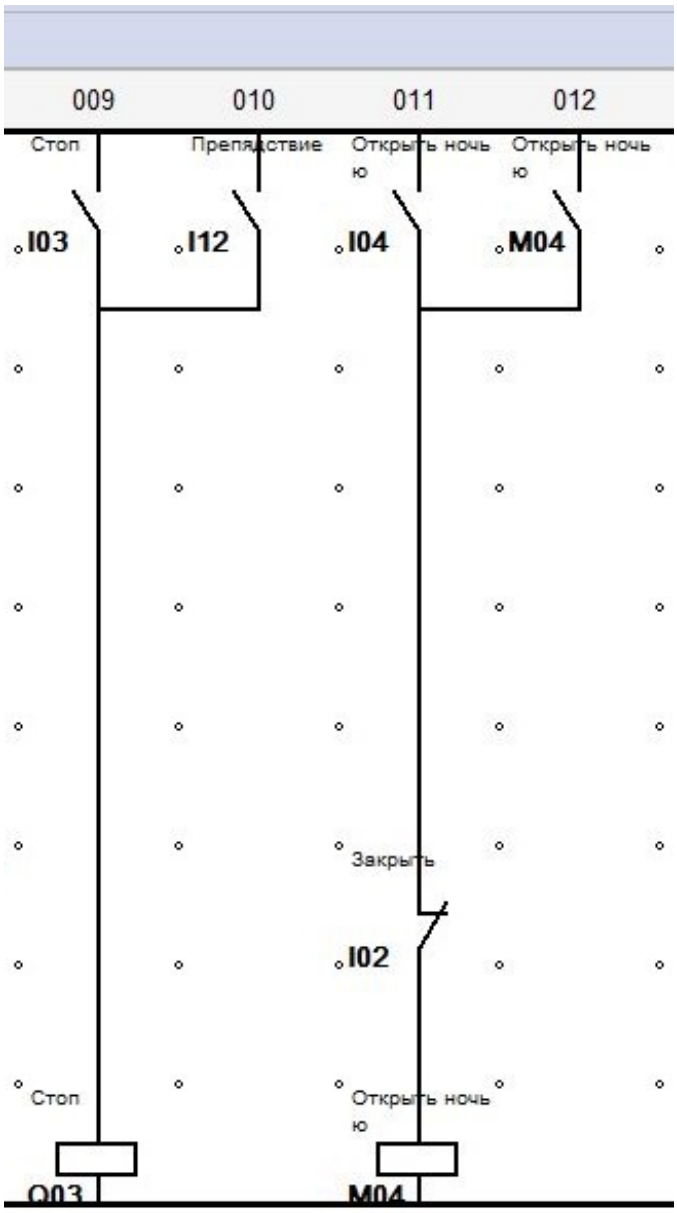

Рис. 76. Схема управления остановом и открытием ночью

На рис. 76 предусмотрен специальный вход реле I04 для открытия ленты жалюзи в тёмное время суток. С помощью стандартного сигнала от кнопки на вход I01 открыть ленту не получится, так как в данном случае цепь питания маркера M01 от входа I01 разорвана размыкающим контактов светодатчика I09. А суть заключается в том, что сигнал с I04 запустит на само-подхват маркер M04, который разомкнёт цепь питания от I09 маркера закрытия M02. Это нужно, чтобы когда жалюзи откроется, не сработало моментально её обратное закрытие. Что было бы бессмысленным. Поэтому создана такая блокирующая цепь. Эта цепь подхвата M04 будет сброшена после обычного закрытия жалюзи с помощью подачи сигнала на вход I02.

На рис. 77–79 показаны примеры настроек входов I, маркеров M и контакторов Q.

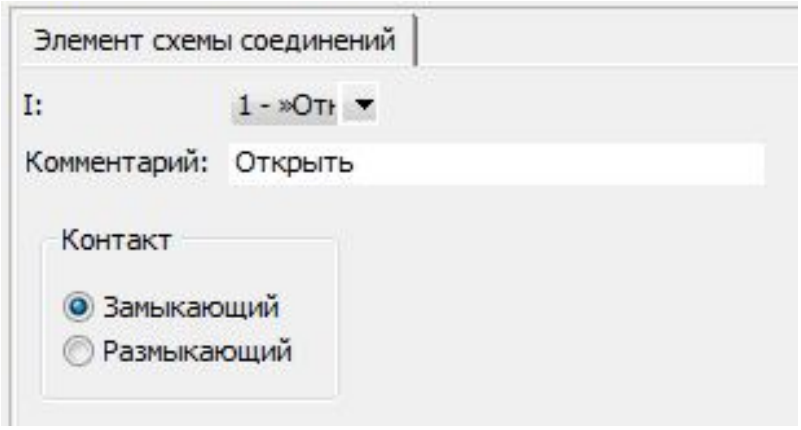

Рис. 77. Настройки замыкающего контакта в хода «I1»

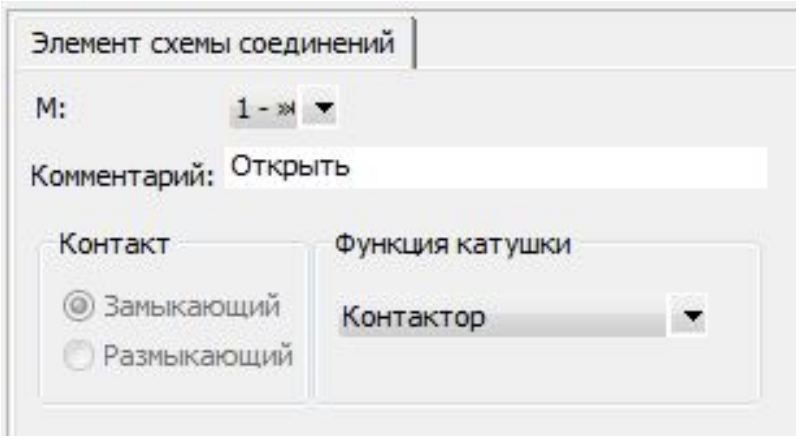

Рис. 78. Настройки маркера «M1»

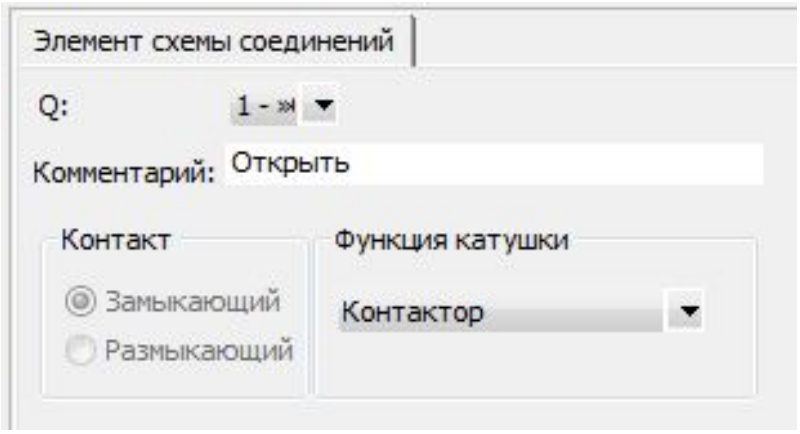

Рис. 79. Настройки замыкающего контакта выхода «Q1»

Для всех остальных входов, маркеров и контакторов настройки делаются аналогичным образом.

5.4. Апробация созданной системы в среде Easy Soft

После создания программы в среде Easy Soft, можно провести имитацию её работы. Это удобно для тестирования в локальной среде разработки на локальном компьютере. Перед имитацией желательно настроить принципы работы каждого из входов «I», т. е. например это кнопка с удержанием или нет (рис. 80).

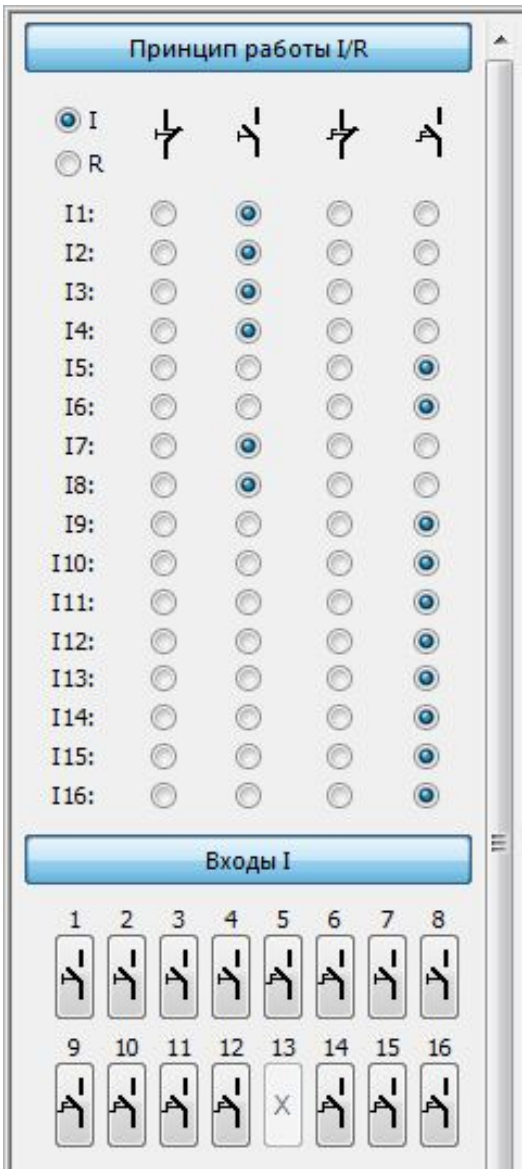

Рис. 80. Настройка принципов работы каждого входа «I»

Также желательно выбрать параметры для отображения (рис. 81). При клике на нужные типы параметров (Входы I или Маркер) откроются окна (рис. 82–83). Либо можно даже настроить осциллограф для наблюдения за изменением назначенных параметров во времени (рис. 84).

Чтобы запустить имитацию, нужно нажать кнопку «зелёная стрелка вправо» как на рис. 85.

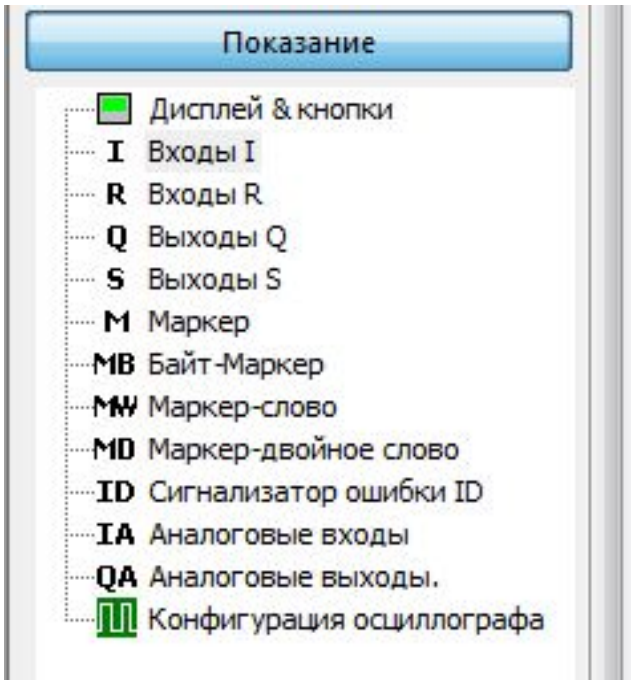

Рис. 81. Окно выбора просмотра показаний параметров

Рис. 82. Показания входов «I»

Рис. 83. Показания маркеров «M»

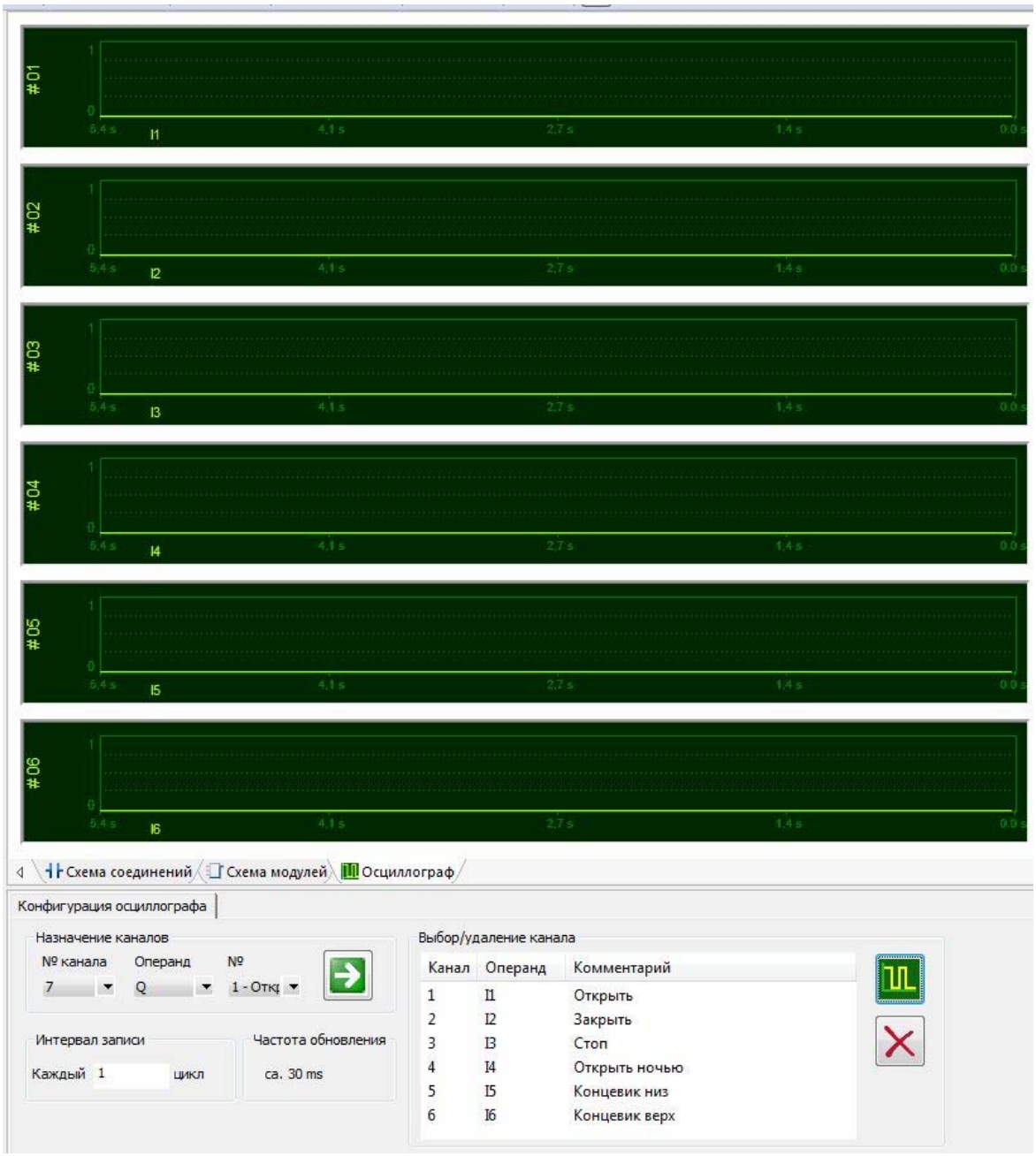

Рис. 84. Каналы осциллографа для просмотра изменений назначенных параметров во времени

Рис. 85. Кнопка запуска режима имитации

После создания программы в среде Easy Soft, программный код нужно записать внутрь процессора электронного реле. Чтобы это сделать, нужна коммуникация компьютера с электронным реле. Для этого предусмотрено устройство расширения Easy209-SE, которое физически

подсоединено к самому реле на стенде. А уже компьютер подсоединяется с помощью витой пары к Easy209-SE.

Чтобы настроить коммуникацию по протоколу Ethernet с электронным реле через его устройство расширения, нажмите внизу слева кнопку «Коммуникация» среды Easy Soft и перейдите в окно, вид которого показан на рис. 86.

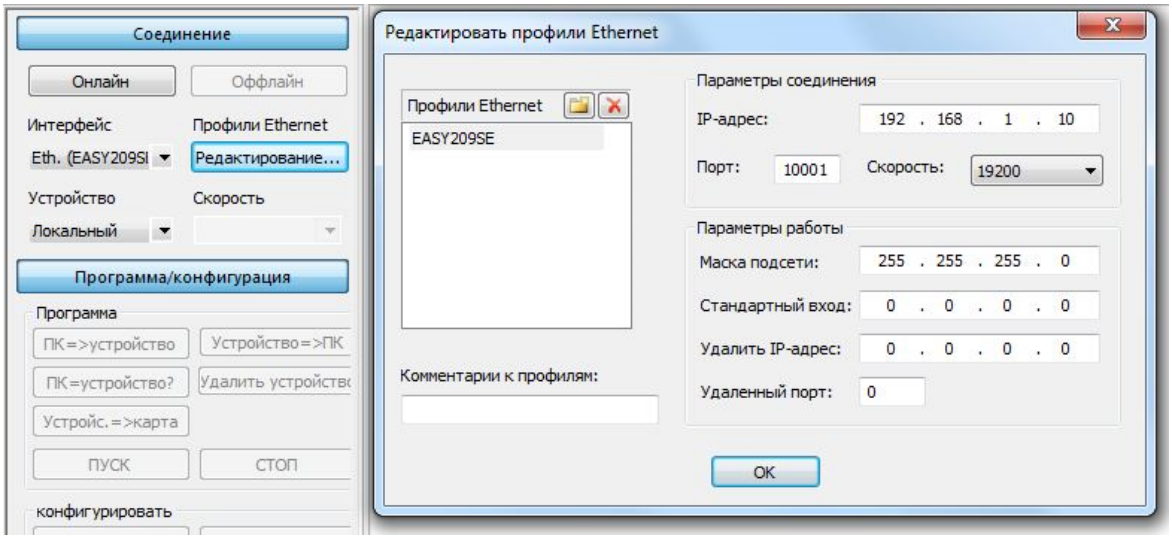

Рис. 86. Настройки подключения к реле

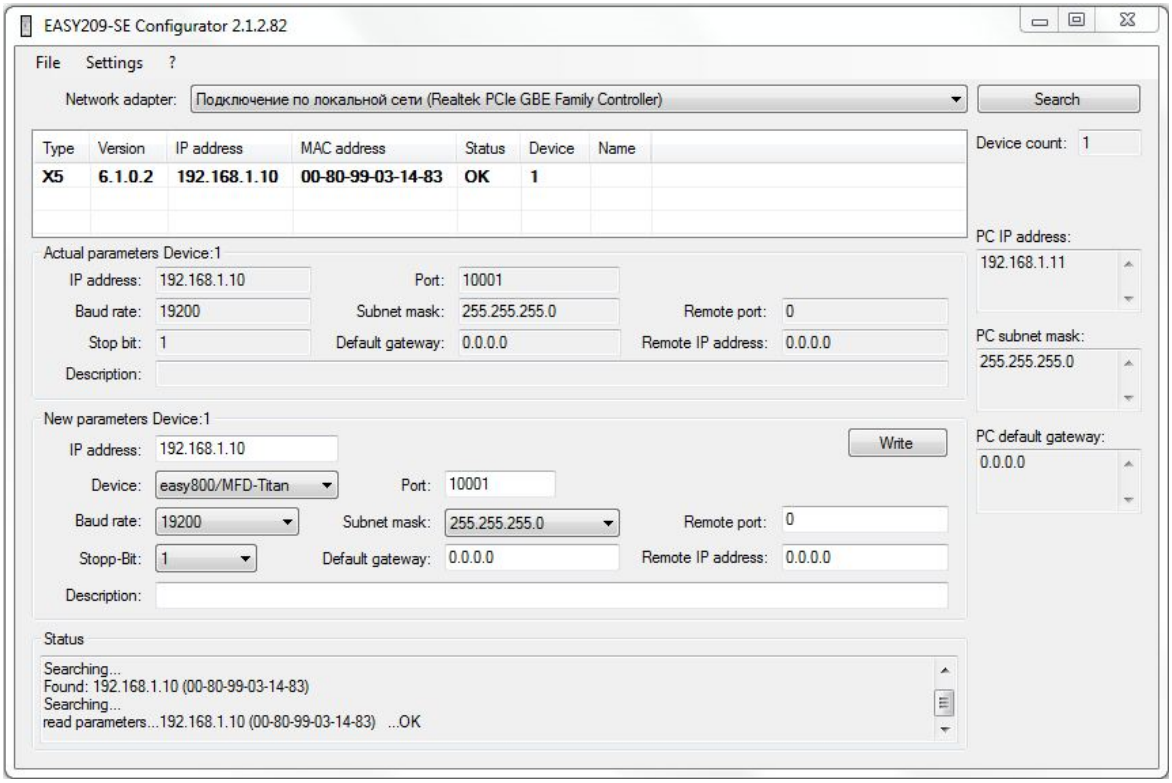

Рис. 87. Определение ip адреса реле с помощью конфигуратора

Откройте окно редактирования интерфейса, создайте новую настройку и назовите её например «EASY209SE». Чтобы узнать какие данные подключения вводить (ip адрес, маска подсети), нужно воспользоваться программой «Easy-209SE Configurator» (рис. 87). Программа интуитивно проста. Она автоматически определит нужные параметры.

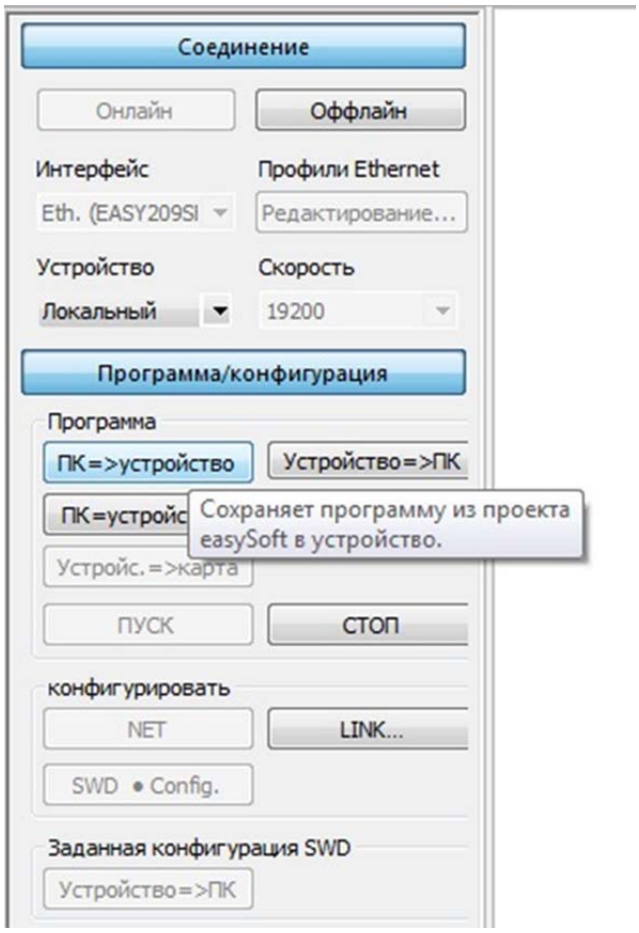

Рис. 88. Сохранение программы из персонального компьютера в процессор электронного реле

Когда настроены параметры подключения, нажмите кнопку «Онлайн». Далее, кнопку «ПК => устройство» чтобы записать программу в процессор реле, рис.88. Можно дистанционно запустить реле, нажав кнопку «Пуск».

6. СИСТЕМА УПРАВЛЕНИЯ ВЫПУСКА ШАССИ САМОЛЁТА

Цель: освоить принципы построения интеллектуальной системы для управления выпуска и уборки шасси самолёта.

6.1. Порядок работы

6.1.1. Создать систему управления на базе программного языка LD (среда EasySoft).

6.1.2. Апробация созданной системы в среде EasySoft в режиме имитации.

6.2. Создание системы управления в среде Easy Soft

6.1.1. Создайте новый проект и добавьте в него устройство – программируемое реле EASY серии 800. В данном случае это EASY-820-DC-RC. Версия устройства 7.

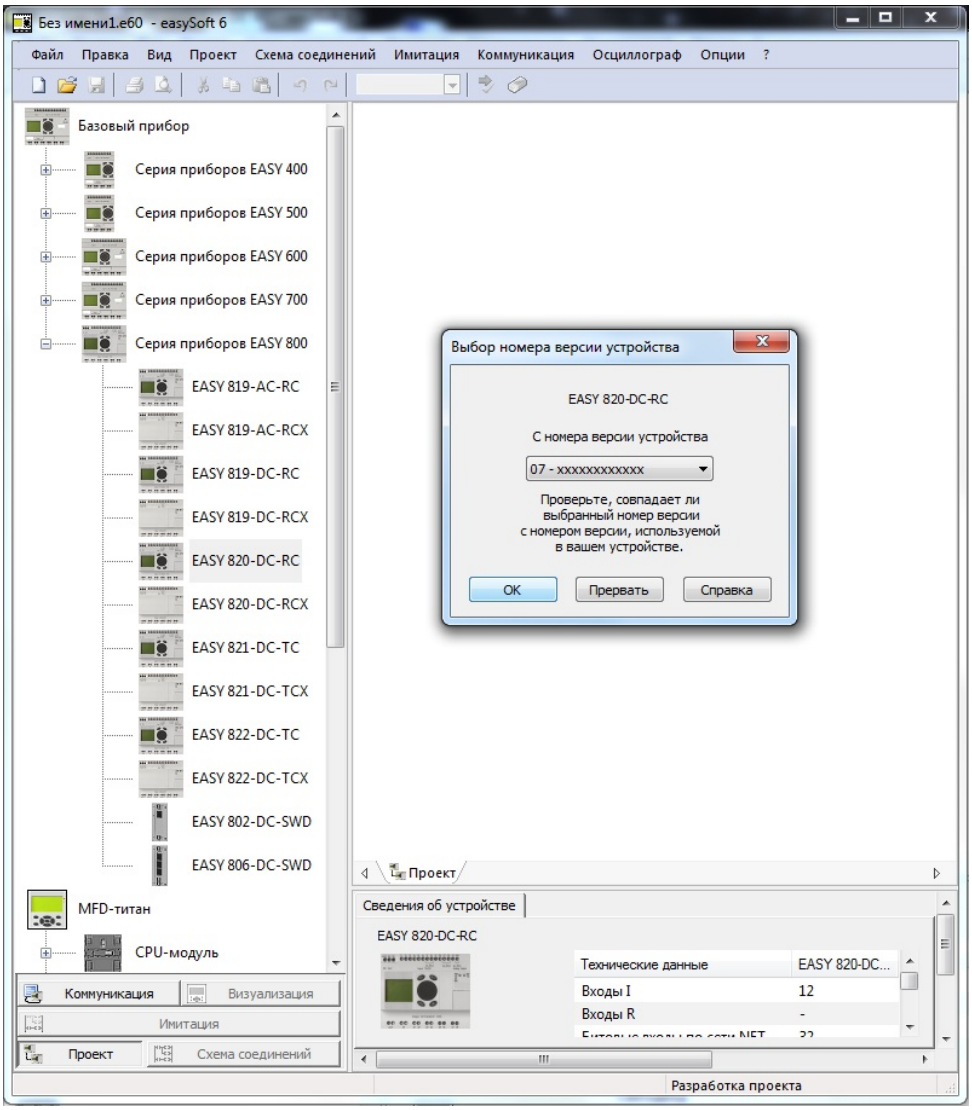

Рис. 89. Добавление нового устройства EASY 820-DC-RC в проект

6.2.2. Нажмите слева внизу кнопку «Схема соединений» и вы перейдёте в окно для рисования схемы управления выпуском шасси. Слева расположена панель с модулями для проектирования.

6.2.3. Используя модули, нарисуйте схему управления выпуском и уборкой шасси, как на рис. 90.

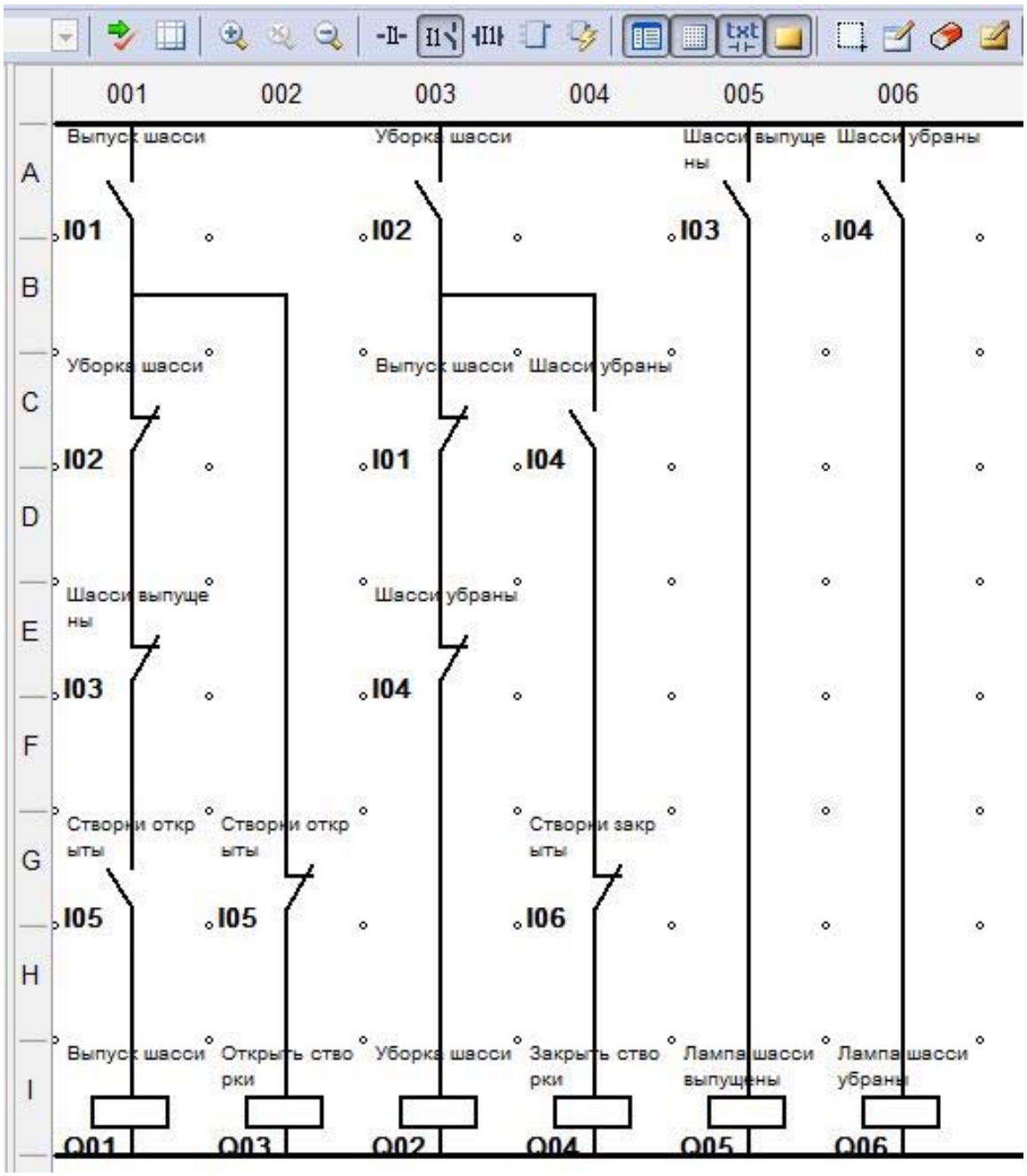

Рис. 90. Схема управления выпуском и уборкой шасси самолёта ТУ-4

Схема разработана на базе [7.6–7.7].

6.2.4. Разберём подробно схему рис. 90 для двух состояний.

1. Шасси убраны. Створки закрыты.

В этом состоянии пары контактов концевого выключателя I04 в ветках схемы 004 и 006 замкнуты. В ветке схемы 003 контакты I04 разомкнуты, предотвращая питания контактора Q02 уборки шасси. При этом запитана лампа сигнализации «убранных шасси» через контактор Q06. Схема готова сначала к открытию створок, а затем к выпуску шасси.

При включении контактов переключателя I01 для выпуска шасси, сработает контактор Q03, который запустит двигателя привода открытия створок. Когда створки откроются, во первых, в ветке схемы 002 контакты концевого выключателя I05 разомкнутся, тем самым обесточив питание двигателя открытия створок через Q03. Во вторых, в ветке схемы 001 замыкающие контакты I05 замкнутся, тем самым запитав контактор Q01, который запустит двигателя выпуска шасси (рис. 91). Когда шасси будут выпущены, сработает концевой выключатель I03. Он разорвёт питание Q01 в ветке схемы 001 и запитает контактор Q05 в ветке схемы 005, который в свою очередь запитает лампу сигнализации «выпущенных шасси». В этот момент в ветке 006 контакты I04 уже будут разомкнуты и лампа сигнализации «убранных шасси» гореть не будет.

2. Шасси выпущены. Створки открыты.

В этом состоянии в ветке схемы 003 контакты концевого выключателя I04 замкнуты, которые подготавливают цепь питания контактора Q02 для уборки шасси. Пока шасси не убраны, в ветке схемы 004 контакты I04 разомкнуты, тем самым блокируя питание контактора Q04 закрытия створок в нужный момент. При включении контактов переключателя I02, получает питание контактор уборки шасси Q02, который питает двигатель уборки шасси (рис. 91). Когда шасси будут убраны, в ветке схемы 003 контакты I04 разомкнутся, потеряет питание Q02 и двигатель уборки шасси остановится. В этот момент в ветке схемы 004 замкнутся контакты I04, которые запитают контактор закрытия створок Q04. Когда створки закроются, в ветке схемы 004 контакты концевого выключателя I06 разомкнутся и разорвут питание контактора Q04 привода двигателя закрытия створок. Также в ветке схемы 006 включатся контакты I04, которые запитают через контактор Q06 лампу сигнализации «убранных шасси».

На рис. 91 приведена схема, которая поясняет принцип питания электроприводов механизмов выпуска и уборки шасси (M1), а также открытия и закрытия заслонок (M2). Каждый двигатель имеет по две обмотки последовательного возбуждения, которые могут создавать магнитные поля разных направлений при их соответствующем питании. Таким образом обеспечивается изменение направления вращения двигателей M1 и M2.

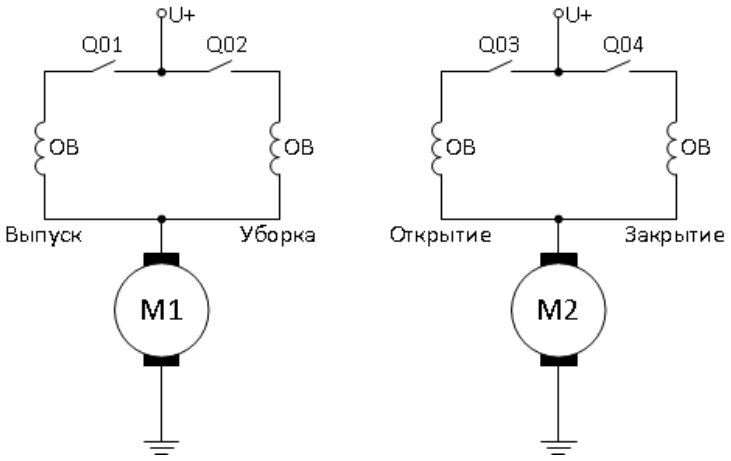

Рис. 91. Принцип питания двигателей привода шасси и створок

На рис. 92, 93 показаны настройки модулей входа «I01» и базового выхода «Q01». Остальные модули схемы рис. 90 настраиваются аналогично.

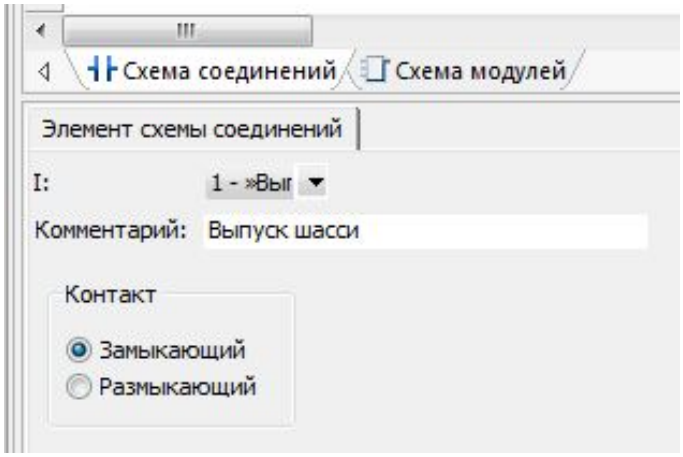

Рис. 92. Настройки модуля входа I01

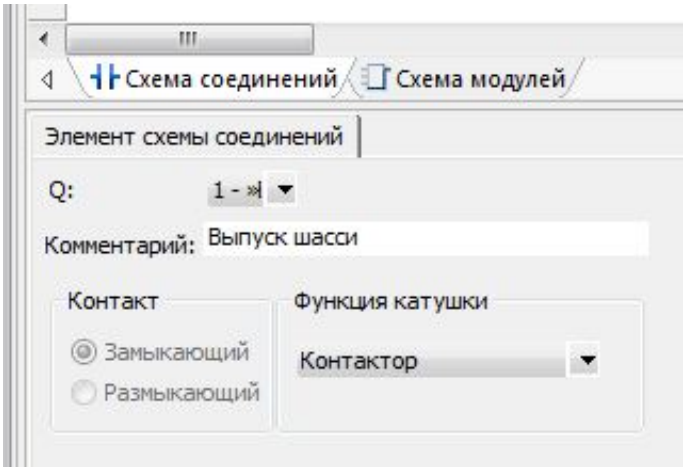

Рис. 93. Настройки модуля контактора Q01

6.3. Апробация созданной системы в среде EasySoft в режиме имитации

6.3.1. Для перехода в режим имитации нажмите кнопку «Имитация» внизу слева на панели, рис. 94.

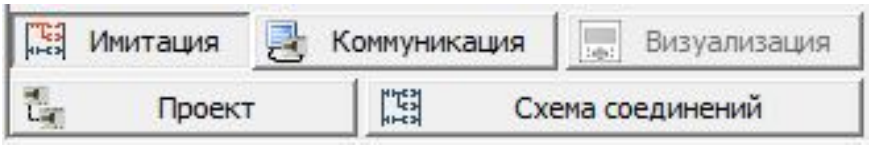

Рис. 94. Кнопка перехода в режим имитации

6.3.2. Настройте принципы работы входов «I», рис. 95.

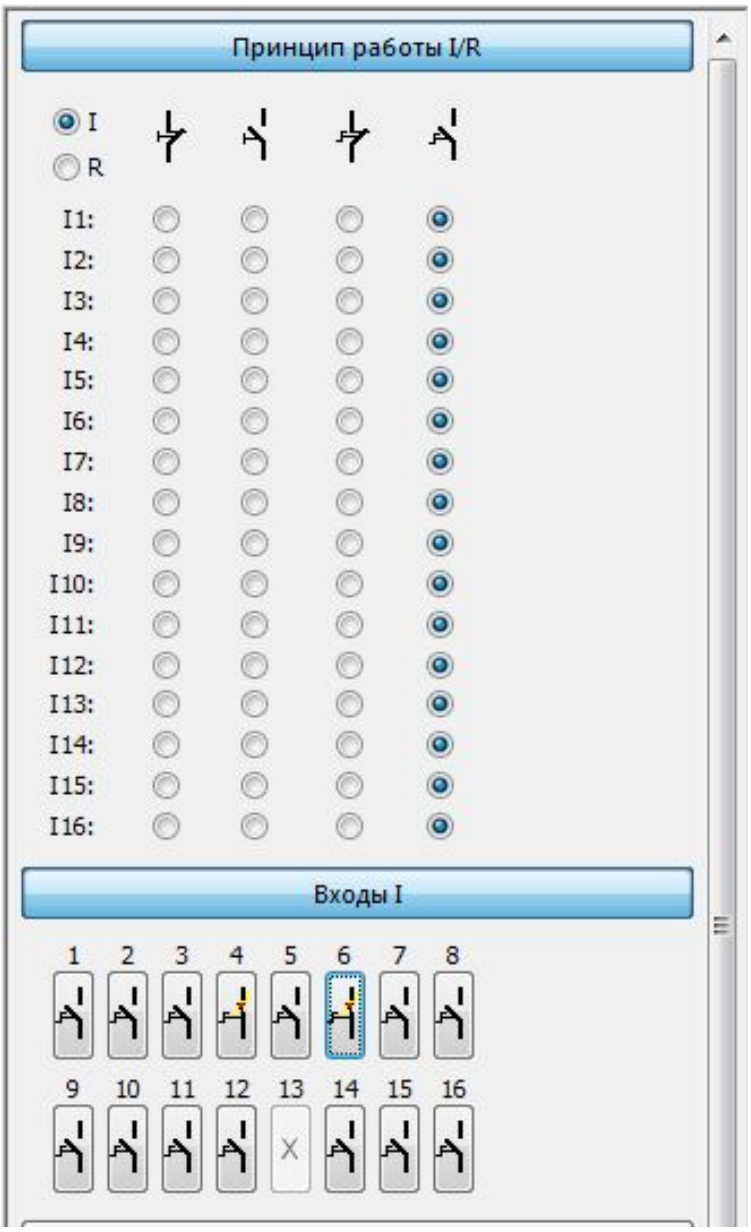

Рис. 95. Настройка принципов работы входов I

6.3.3. Выберите необходимые параметры для визуального отображения в процессе имитации, рис. 96.

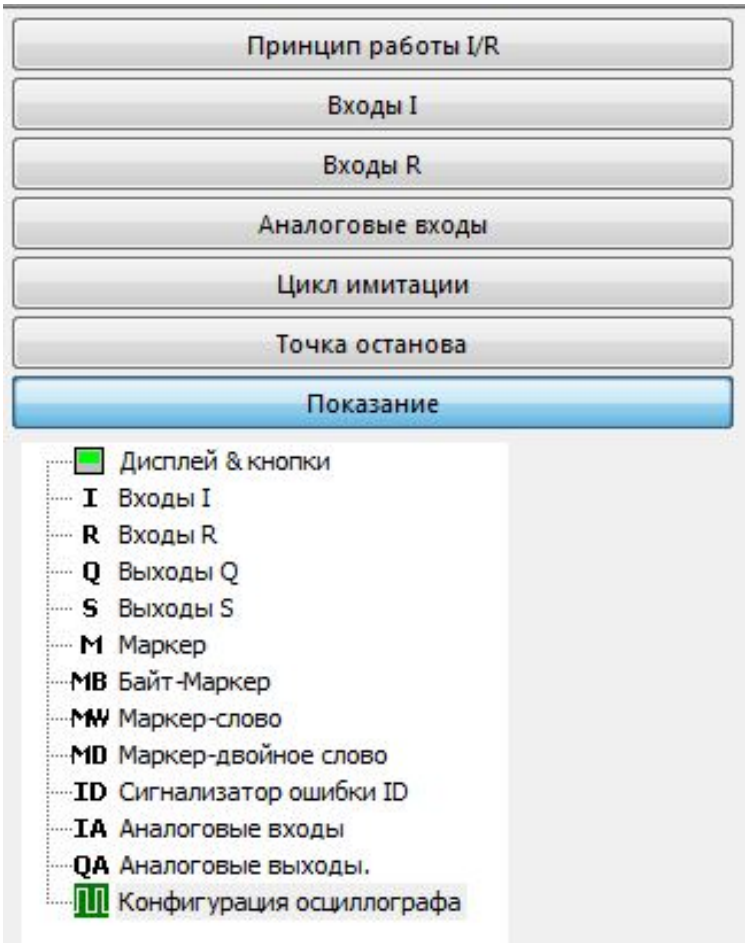

Рис. 96. Окно выбора отображаемых параметров в режиме имитации

Рис. 97. Кнопка запуска режиме имитации

Чтобы запустить режим имитации, нужно нажать на панели верхнего меню, кнопку в виде стрелки, рис. 97.

СПИСОК ЛИТЕРАТУРЫ

1. Электронные программируемые реле EASY и MFD-Titan. – Одесса: Издательство, 2006. – 223 с.

2. Руководство пользователя по программированию ПЛК в CoDeSys 2.3. 3S – Smart Software Solutions GmbH 2006.

3. Руководство пользователя по программированию ПЛК в CoDeSys V2.3.-Смоленск: ПК «Пролог», 2004. – 423 с.

4. Первые шаги с CoDeSys. 3S – Smart Software Solutions GmbH 2004.

5. Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК CoDeSys 2.3. 3S – Smart Software Solutions GmbH 2008.

6. Электрооборудование летательных аппаратов : учебник для вузов. Т. 2. / под ред. С. А. Грузкова. – Москва : Издательский дом МЭИ, 2016. Т. 2.: Элементы и системы электрооборудования – приемники электрической энергии. – 552 с.: ил.

7. Электрооборудование летательных аппаратов : учебник для вузов в 2 т. / под ред. С. А. Грузкова. – Москва : Издательский дом МЭИ , 2018. Т. 1 : Системы электроснабжения летательных аппаратов. – 568 с.

Учебное издание

ГИРНИК Андрей Сергеевич

СИСТЕМЫ АВТОМАТИКИ И УПРАВЛЕНИЯ НА БАЗЕ ПРОГРАММИРУЕМЫХ ЭЛЕКТРОННЫХ АППАРАТОВ ПРОИЗВОДСТВА EATON

Методические указания к выполнению лабораторных работ
по профилям «Электромеханические и электротехнические системы автономных объектов», «Электрооборудование летательных аппаратов», «Электропривод и автоматика», а также по дисциплинам «Микропроцессорные средства систем автоматики, управления и диагностики», «Микропроцессорные устройства в электрооборудовании автономных объектов», «Мехатронные системы летательных аппаратов» для студентов, обучающихся по направлению 13.03.02 «Электроэнергетика и электротехника»

Компьютерная верстка *Д.В. Сотникова*

Зарегистрировано в Издательстве ТПУ
Размещено на корпоративном портале ТПУ

Издательство
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ