ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

С.А. Николаенко, Д.С. Цокур
Д.П. Харченко, А.П. Волошин

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

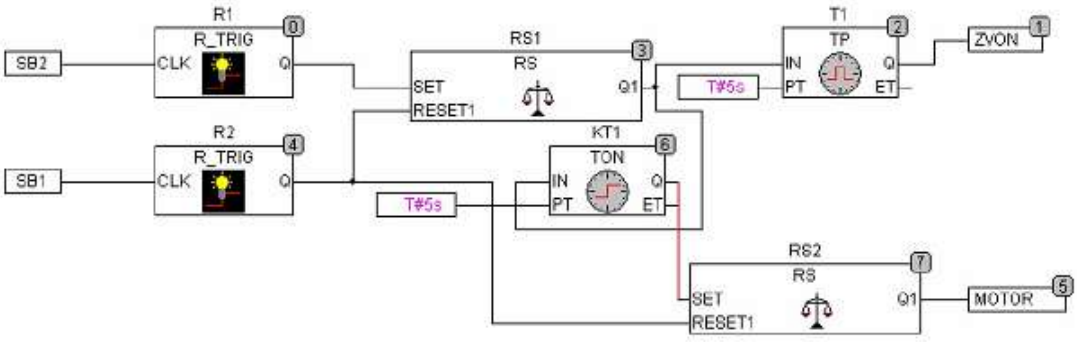
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

С.А.Николаенко, Д.С.Цокур
Д.П.Харченко, А.П.Волошин

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

■

Учебное пособие



Краснодар
2016

УДК 681.5
ББК 31.291

Р е ц е н з е н т ы :

Г.П. Стародубцева – профессор кафедры физики ФГБОУ ВО Ставропольского ГАУ, д-р с.-х наук, профессор;

И.В. Юдаев – заместитель директора по научной работе Азово-Черноморского инженерного института ФГБОУ ВО Донской ГАУ в г. Зернограде, д-р техн. наук, доцент

Николаенко С.А.
Автоматизация технологических процессов: учеб. пособие / С.А. Николаенко, Д.С. Цокур, Д.П. Харченко, А.П. Волошин – Краснодар: Изд-во ООО «КРОН», 2016. – 218 с.

В учебном пособии изложены теоретические основы, а также практические указания к выполнению лабораторных работ студентами по дисциплине «Автоматизация технологических процессов». Представленный материал может использоваться для изучения теории автоматического управления и методов составления алгоритмов управления технологическими процессами. Пособие предназначено для очной, заочной и дистанционной форм обучения для обучающихся всех трёх уровней (бакалавры, магистры и аспиранты) по направлениям подготовки «Агроинженерия» в соответствии с требованиями государственных общеобразовательных стандартов. Данное учебное пособие рекомендовано учёным советом факультета энергетики Кубанского ГАУ.

УДК 681.5
ББК 31.2

СОДЕРЖАНИЕ

1. АВТОМАТИЗАЦИЯ ПРОЦЕССОВ В СЕЛЬСКОХОЗЯЙСТВЕННОМ ПРОИЗВОДСТВЕ

Автоматизация создает техническую и научную основу для возникновения и последующего развития новейших направлений технического прогресса. В свою очередь, развитие микропроцессорной базы с применением новых радионавигационных систем и топоориентированных технологий, а также довольно быстрый рост технической оснащенности — все это создает необходимые предпосылки для проведения автоматизации деятельности сельскохозяйственного производства.

Таким образом, автоматизация производственных процессов является стратегическим направлением в развитии технологий и техники. Если учесть тот факт, что мировой уровень механизации самых важных в животноводстве и полеводстве процессов скоро достигнет 100%, то последующее развитие сельскохозяйственной техники наверняка будет характеризоваться более интенсивным использованием методов и средств автоматизации, робототехнических комплексов и информатизации.

Однако довольно часто стремление получить максимально высокое качество продукции и внедрение технологических интенсивных процессов ограничивается физиологическими возможностями обычного человека. Именно поэтому сегодня начинают широко использоваться высокоточные технологии, которые базируются на автоматическом управлении процессом. Следует отметить, что в течение последних десятилетий механизация сельского хозяйства сформировалась в отдельную, самостоятельную отрасль техники и науки, которая охватывает способы использования, теорию и принципы построения автоматизированных систем управления, действующих как с самым минимальным участием человека, так и без его участия.

Выделяют основные цели автоматизации технологического процесса:

- повышение эффективности производственного процесса;
- повышение безопасности производственного процесса.

Задачи автоматизации и их решение достигаются посредством решения следующих задач автоматизации технологического процесса:

- улучшение качества регулирования;

- повышение коэффициента готовности оборудования;

-улучшение эргономики труда операторов процесса;

- хранение информации о ходе технологического процесса и аварийных ситуациях.

Решение задач автоматизации технологического процесса осуществляется при помощи:

- внедрения современных методов автоматизации;

- внедрения современных средств автоматизации.

Автоматизация технологических процессов в рамках одного производственного процесса позволяет организовать основу для внедрения систем управления производством и систем управления предприятием. В связи с различностью подходов различают автоматизацию следующих технологических процессов:

- автоматизация непрерывных технологических процессов;

- автоматизация дискретных технологических процессов;

- автоматизация гибридных технологических процессов.

Непрерывные технологические процессы отличаются тем, что, как правило, сырье и полуфабрикаты подаются на переработку непрерывно в течение достаточно продолжительного времени, часто поступают с одного передела на другой без промежуточного хранения с задержкой только на время транспортировки. При этом машины и механизмы работают непрерывно во времени.

Дискретные технологические процессы характеризуются в основном следующими особенностями: наличием отдельных операций с четко выраженными началом и концом; наличием регламентированных перерывов с остановом и выключением различных групп технологического оборудования; относительной универсальностью единиц технологического обору-

дования, что обусловливает возможность выполнения на одном рабочем месте нескольких видов операций.

Гибридные технологические процессы объединяют в себе непрерывные и дискретные технологические процессы. Отличительная способность их заключается в объединении преимуществ разных технологий в одно единое целое.

Основной особенностью автоматизации сельскохозяйственного производства на современном этапе его развития является неразрывная связь биологических объектов с техникой, а следовательно, с непостоянными в различных промежутках времени параметрами (животных, растений, почвы) с непрерывностью процессов изготовления продукции и цикличностью получения продукции, свойственных только этим параметрам. В данных условиях системы автоматики должны учитывать:

- сложность и многообразие производственных процессов – это обуславливается разнообразием техники и технологических процессов;
- связь техники с различными биологическими объектами – при этом технику следует рассматривать в качестве человеко-машинной системы;
- распределенность регулируемых и контролируемых параметров большинства объектов по технологическому полю или объекту (теплицы или хранилища) в совокупности с возможными случайными возмущающими воздействиями;
- условия работы автоматизированных систем (в неотапливаемых помещениях, на открытом воздухе) с изменением в достаточно широких пределах влажности, температуры, запыленности, состава агрессивных газов, интенсивности солнечной радиации и др.;
- удаленность сельскохозяйственной техники от ремонтной базы, ее рассредоточенность по большим территориям.

1.1 Общие требования, предъявляемые к схемам управления технологическими процессами

В последнее время для успешной автоматизации технологических процессов предъявляют ряд требований, которые условно можно разделить на три группы [1]:

- технологические требования;
- требования безопасности;
- требования надежности.

Технологические требования приемлемы для трех типов ТП и формулируются следующим образом:

1) обеспечить запуск всех машин и механизмов в последовательности, направленной против движения продукта (зерна, дерти, корнеклубнеплодов, сена и др.);

2) обеспечить остановку всех машин и механизмов в последовательности, совпадающей с направлением движения продукта;

3) обеспечить остановку поточных линий по команде «рабочий стоп» с целью очистки тракта.

4) схемы должны иметь режим пусконаладочных работ. Для этого в схемах необходимо предусмотреть деблокировочные режимы, обеспечивающие возможность включения отдельного электрооборудования;

5) при необходимости предусмотреть изменение технологических параметров, например, угловой скорости;

6) учесть удобство и гибкость управления во всех режимах.

Требования безопасности:

1. Для безопасности обслуживающего персонала пуску сложных технологических установок должен предшествовать звуковой или световой сигнал.

2. Схемы должны обеспечивать невозможность неправильного включения или отключения электрических цепей.

3. Обеспечить защиту электрооборудования от аварийных режимов работы.

4. Предусмотреть аварийное отключение. При аварийном отключении одной из машин должны остановиться, без выдержки времени, все машины, работающие на ее загрузку, а с выдержкой времени– все машины, работающие на отгрузку.

Например: обеспечить последовательное включение размыкающих контактов защитных устройств в цепь катушки магнитного пускателя защищаемого электрооборудования.

Включение в общую цепь питания катушек не обеспечивает должной последовательности остановки при аварийных режимах. Замыкающие контакты магнитных пускателей защищаемого оборудования включить в цепь управления машин, работающих на загрузку.

Требования надежности:

1. Схемы управления должны быть относительно просты и состоять из однотипной аппаратуры, содержать минимальное число контактов.

2. Схемы должны быть включены с учетом удобства монтажа.

3. Схемы должны составляться таким образом, чтобы обеспечить контроль неисправностей. Необходимо наличие сигнализации о запуске отдельных машин.

4. Выбранная аппаратура должна соответствовать условиям работы электроустановок (число срабатываний, защита от воздействий окружающей среды).

5. Аппаратура должна быть доступна для проведения периодических технических осмотров и обслуживаний.

6. Схемы должны учитывать последствия отказов аппаратуры.

2. РЕЛЕЙНО-КОНТАКТНЫЕ СХЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

2.1 Общие сведения

Релейно-контактные схемы (их часто называют переключательными схемами) широко используются в технике автоматического управления.

К переключающим устройствам автоматики относятся:

- реле;
- контактные и бесконтактные устройства управления.

Реле – это коммутационное устройство, которое при воздействии каких-либо внешних факторов скачкообразно изменяет свое состояние.

По назначению реле бывают:

- управления (управляют электродвигателями, электромагнитными тормозами и т.п.);
- защиты (для включения и отключения аппаратов защиты, в схемах релейной защиты и т.п.);
- автоматики.

По характеру входной величины реле делятся на:

- электрические;
- оптические;
- тепловые;
- механические;
- акустические и т.п.

Электрические реле служат для включения и отключения электрических цепей, размножения контактов, блокировки, памяти и т.д.

Электрические реле различают:

по принципу действия

- электромагнитные;
- магнитоэлектрические;
- электронные;
- статические;

- электротепловые;

по способу коммутации:

- контактные;
- бесконтактные.

Рассмотрим некоторые из реле:

1) электрические реле;
2) реле на магнитоуправляемых контактах – герконовые реле;
3) электронные реле времени.

1.Электрические реле. Работа электромагнитных реле основана на использовании электромагнитных сил, возникающих в металлическом сердечнике при прохождении тока по виткам его катушки. Детали реле монтируются на основании и закрываются крышкой. Над сердечником электромагнита установлен подвижный якорь (пластина) с одним или несколькими контактами. Напротив них находятся соответствующие парные неподвижные контакты.

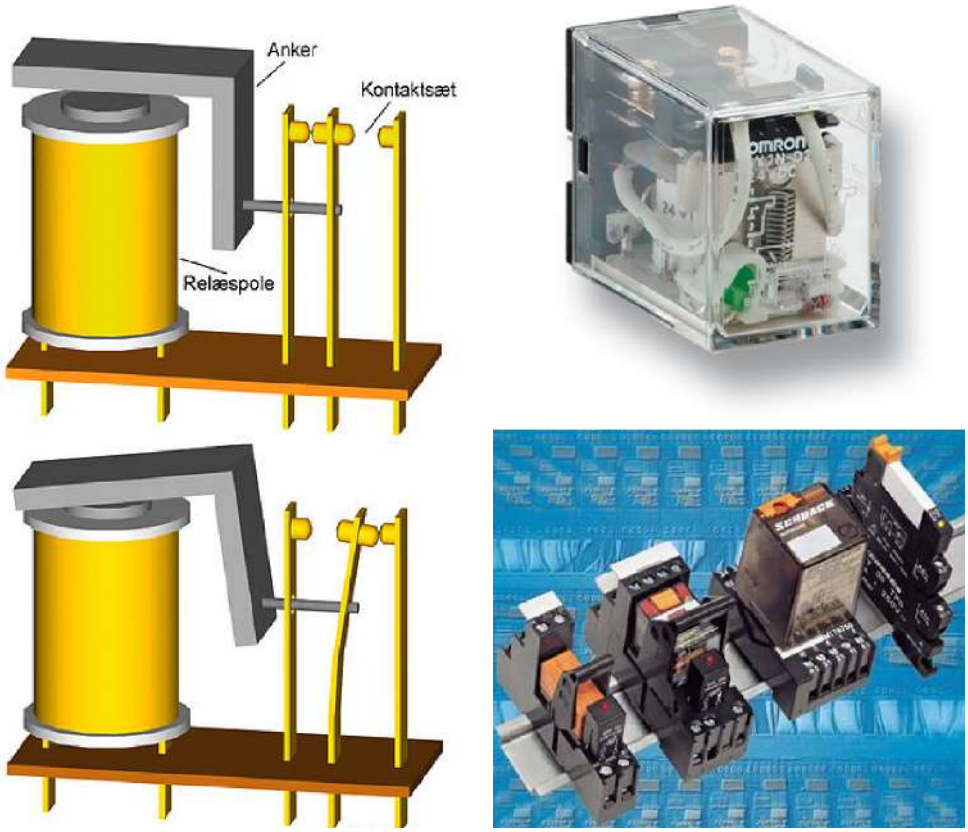


Рисунок 2.1– Внешний вид электрических реле.

2. Реле на магнитоуправляемых контактах –герконовые реле. Геркóн (сокращение от «герметичный [магнитоуправляемый] контакт») — электромеханическое устройство, представляющее собой пару ферромагнитных контактов, запаянных в герметичную стеклянную колбу. Принцип действия герконов основан на использовании сил взаимодействия, возникающих в магнитном поле между ферромагнитными телами. При этом силы вызывают деформацию и перемещение ферромагнитных токопроводов электронов. Магнитоуправляемый контакт (геркон) представляет собой электрический аппарат, изменяющий состояние электрической цепи посредством механического размыкания или замыкания ее при воздействии управляющего магнитного поля на его элементы, совмещающие функции контактов, пружин и участков электрической и магнитной цепей.

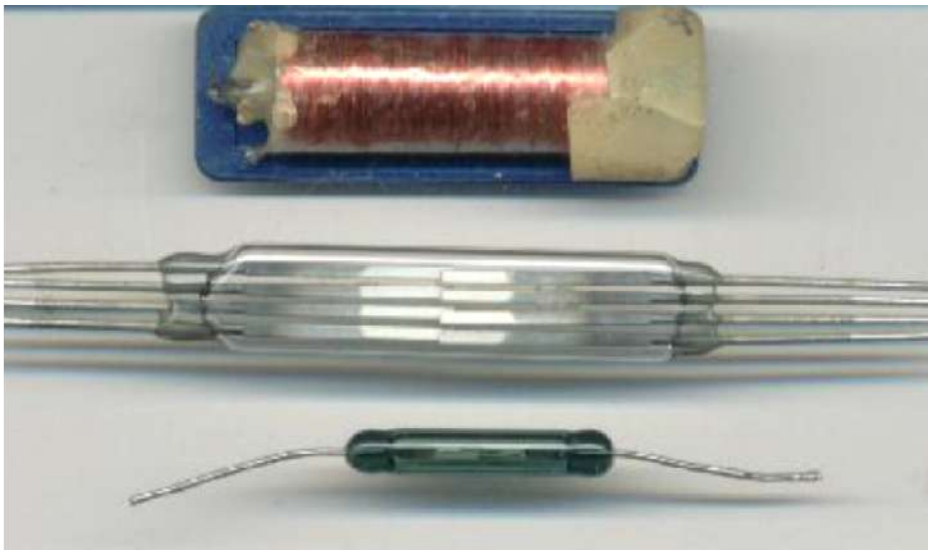


Рисунок 2.2 – Внешний вид реле на магнитоуправляемых контактах - герконовые реле.

3. Электронные реле времени. Реле времени – реле, предназначенное для создания независимой выдержки времени и обеспечения определенной последовательности работы элементов схемы. Реле времени применяется в случаях, когда необходимо автоматически выполнить какое-то действие не сразу после появления управляющего сигнала, а через установленный промежуток времени.

До появления недорогих микроконтроллеров работа электронных реле времени была основана на переходных процессах в разрядном контуре RC или RL.

Рисунок 2.3 – Внешний вид электронного реле времени.

Современные реле времени отрабатывают необходимую задержку времени в соответствии с программой, «зашитой» в микроконтроллер. При этом сам микроконтроллер может тактироваться с помощью встроенного кварцевого резонатора или RC-генератора.

Контактные устройства управления. Магнитный пускатель является коммутационным аппаратом и относится к семейству электромагнитных контакторов, позволяющих коммутировать мощные нагрузки постоянного и переменного тока, и предназначен для частых включений и отключений силовых электрических цепей [17].

Рисунок 2.4 – Внешний вид магнитного пускателя.

Магнитные пускатели применяются в основном для пуска, останова и реверсирования трехфазных асинхронных электродвигателей, но из-за своей неприхотливости они прекрасно работают в схемах дистанционного управления освещением, в схемах управления компрессорами, насосами, кран-балками, тепловыми печами, кондиционерами, ленточными конвейерами и т.д.

Принцип работы магнитного пускателя. Принцип работы очень простой: напряжение питания подается на катушку пускателя, в катушке возникает магнитное поле, за счет которого вовнутрь катушки втягивается металлический сердечник, к которому закреплена группа силовых (рабочих) контактов, контакты замыкаются, и через них начинает течь электрический ток. Управление магнитным пускателем осуществляется кнопками «Пуск», «Стоп», «Вперед» и «Назад».

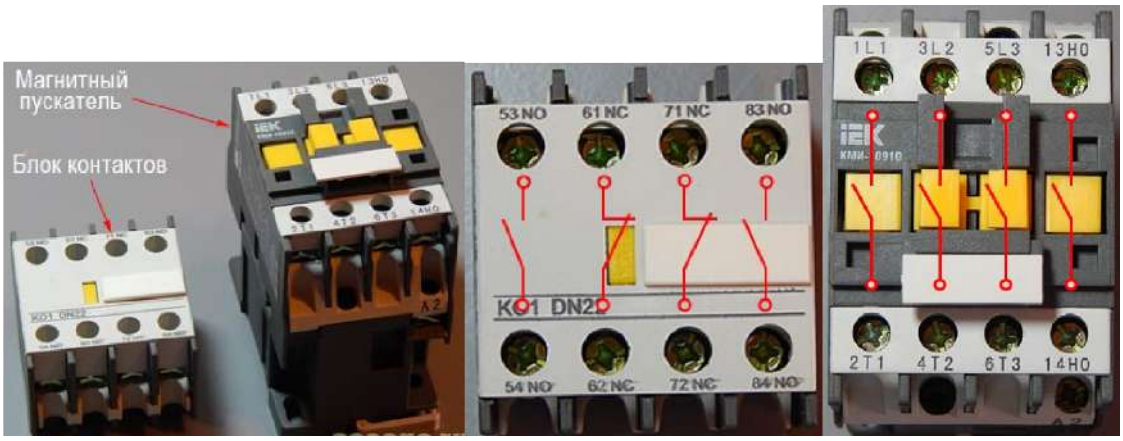


Рисунок 2.5 – Внешний вид магнитного пускателя и блока контактов с нанесенными обозначениями контактов.

Часто к магнитному пускателю устанавливают блок контактов. Хотя блок контактов и не является основной частью магнитного пускателя и не всегда он используется, но, если пускатель работает в схеме, где должны быть задействованы дополнительные контакты этого пускателя, например, реверс электродвигателя, сигнализация работы пускателя или включение дополнительного оборудования пускателем, то для размножения контактов как раз и служит блок контактов, или, как его еще называют, приставка контактная.

2.2 Типовые звенья схем автоматизации на базе релейно-контактных элементов

Для составления схем управления технологических процессов часто используют типовые звенья, реализованные в частном случае на базе релейно-контактных элементов. Выделяют четыре самых распространенных звена, первым, из которых является звено *нереверсивной схемы управления*. Состоит из двух кнопок управления и магнитного пускателя. Предназначено для ручного управления включением и отключением электроустановок (двигателей, нагревателей систем освещения и прочее) [11].

Принцип работы: при нажатии на кнопку пуска SB2 контакт кнопки замыкает цепь с катушкой магнитного пускателя КМ1, который в свою очередь силовыми контактами подаёт питание к потребителю (на схеме не обозначены силовые контакты), а дополнительным контактом КМ1.1 шунтирует кнопку пуска. При этом магнитный пускатель включает себя на «самоудержание». Для отключения потребителя необходимо нажать кнопку «стоп» SB1, что приведет к разрыву питающей цепи и размыканию катушки магнитного пускателя. Защита электродвигателя от перегрузки в представленной схеме управления реализуется посредством размыкающего контакта теплового реле КК1.1. В последующих схемах управления эта защита также будет реализовываться.

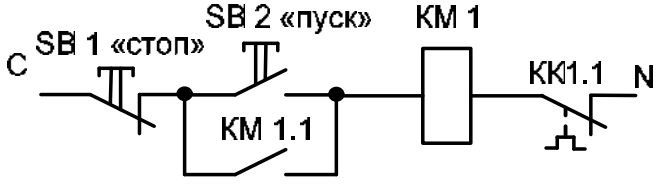


Рисунок 2.6 – Нереверсивная схема управления электродвигателя.

Звено *реверсивной схемы управления*. Состоит из трех кнопок управления и двух магнитных пускателей. В общем случае может использоваться для взаимоисключающего управления работой 2 механизмов. Реверсивная схема в первую очередь предназначена для реверсивного управ-

ления трехфазным асинхронным электродвигателем. Схема содержит 2 блокировки, исключающие одновременную работу 2 магнитных пускателей.

Принцип работы: при нажатии на кнопку пуска «вперед» SB2 один контакт кнопки замыкает цепь с катушкой магнитного пускателя КМ1, что приводит к включению электродвигателя, другой наоборот разрывает цепь с катушкой магнитного пускателя КМ2, тем самым не дает возможности одновременной подачи питания на две цепи управления. При этом дополнительным контактом КМ1.1 шунтирует кнопку пуска, тем самым обеспечивая питания первой катушки магнитного пускателя.

При нажатии на кнопку пуска «назад» SB3 нарушается целостность цепи питания катушки магнитного пускателя КМ1 и она отключается, и одновременно с этим замыкается цепь с КМ2. Это приводит к включению электродвигателя к сети только с другим чередованием фаз.

Для отключения электродвигателя от сети необходимо нажать кнопку «стоп» SB1, что приведет к разрыву питающей цепи и размыканию одной из катушек магнитного пускателя. Для защиты от одновременного включения двух катушек предусмотрены блокировочные контакты КМ2.2 и КМ1.2.

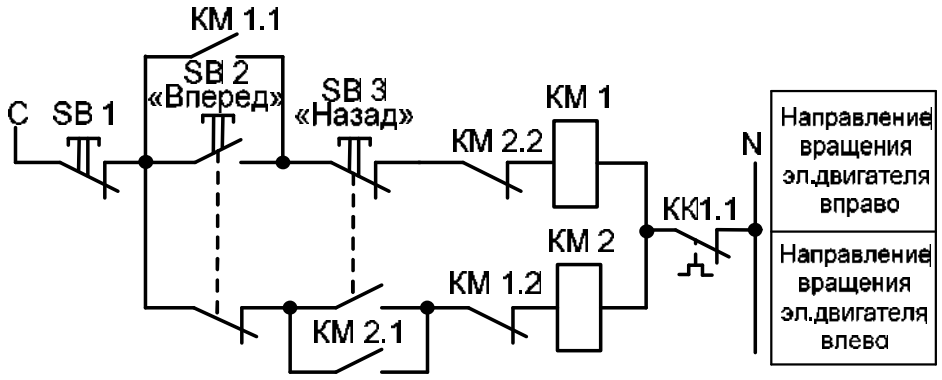


Рисунок 2.7 – Реверсивная схема управления электродвигателя.

Звено *пускосигнальное* предназначено для предупреждения персонала о запуске электроустановки либо электродвигателей технологической

линии. Состоит из двух кнопок управления, магнитного пускателя, промежуточного реле и реле времени.

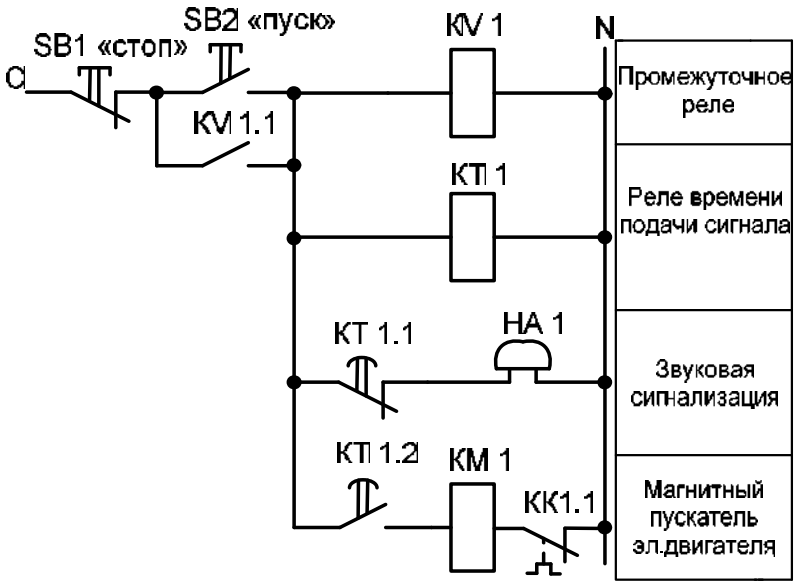


Рисунок 2.8 – Схема пускосигнального звена.

Представленное на рисунке звено работает следующим образом. При нажатии на кнопку пуска SB2 питание подается одновременно на катушку промежуточного реле KV1, реле времени КТ1 и через его замкнутый контакт КТ1.1 на звонок. При этом промежуточное реле служит только для шунтирования кнопки SB2 контактом KV1.1. Реле времени имеет два контакта, работа которых определяется временем установки. Так контакт КТ1.1 в первоначальный момент замкнут (КТ1.2 разомкнут), питание непрерывно подается на звонок НА1. После установленного времени реле срабатывает и замыкается контакт КТ1.2 (КТ1.1 разомкнут), что приводит к подаче питания на катушку магнитного пускателя КМ1, которая в свою очередь включает механизм.

Звено «*Рабочий стоп*» предназначено для обеспечения технологической остановки поточной линии с целью очистки тракта. Состоит из одной кнопки управления, промежуточного реле и реле времени. Работает следующим образом: при нажатии кнопки SB1 «Рабочий стоп» питание подается на промежуточное реле KV1 и реле времени КТ1. При этом одним

контактом промежуточного реле KV1.1 происходит шунтирование кнопки SB1, а другим KV1.2 происходит размыкание цепи питания магнитного пускателя головного механизма. Это приводит к отключению механизма, подающего продукт на линию, тем самым происходит очистка тракта.

Роль реле времени в данной схеме заключается в отключении питания всей схемы управления с задержкой по времени, необходимой для полной очистки механизмов от продукта. Так контакт КТ1.1 размещается в схеме управления, в зависимости от технологического процесса, либо в цепи с катушкой магнитного пускателя первого включенного механизма, либо в цепи с промежуточным реле KV1, контакт которого шунтирует кнопку питания SB2 схемы управления.

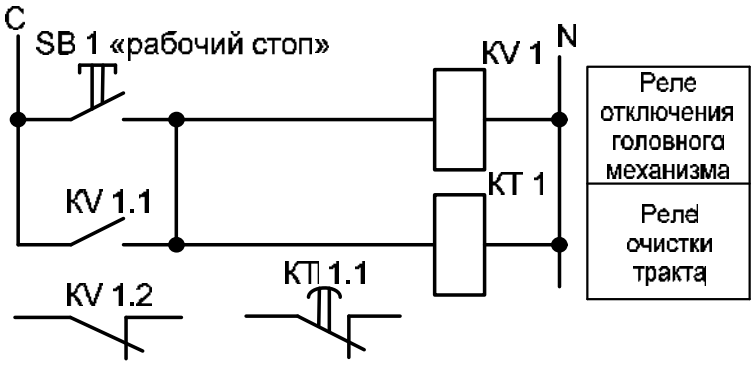


Рисунок 2.9 – Схема рабочего стопа.

2.3 Практическое изучение релейно-контактных схем на базе лабораторного стенда

Лабораторное оборудование, на котором будут собираться разработанные релейно-контактные схемы, представлено на рисунке 2.10. Питание лабораторного стенда осуществляется от сети переменного тока ~220В. При включении сетевой кнопки 1 загорается сигнальная лампа, на фазной и нулевой шинах появляется напряжение 24В. С этим безопасным напряжением происходит дальнейшая работа студентов на стенде. Лабораторная установка рассчитана на сборку различных релейно-контактных схем.

На передней панели лабораторного стенда размещены: 1) сетевая кнопка питания; 2) контакты реле времени; 3) лампа, сигнализирующая об аварийном режиме работы; 4) контакты магнитного пускателя; 5) контакты шины питания; 6) предохранитель; 7) контакты четырёх тумблеров на 2 положения; 8) контакты сигнальных ламп; 9) контакты пакетного переключателя на 3 положения; 10) контакты кнопок управления без фиксации; 11) контакты звонка; 12) контакты нулевой шины.

Для имитации работы механизмов линии на стенде предусмотрены сигнальные лампы EL в количестве 13 шт. Построение схем управления с задержкой времени осуществляется посредством четырех реле времени КТ. Особенность данных реле в том, что у них существует один перекидной контакт, поэтому и разрабатывать схемы управления необходимо с учетом этой особенности.

Для автоматического дистанционного выбора режимов работы используется пакетный переключатель SA. Особенность данного устройства заключается в том, что у него всего лишь три положения, а вот групп коммутации шесть.

Кнопки управления SB состоят из двух групп контактов (размыкающего и замыкающего), между которыми существует механическая связь.

Работу механизмов линии осуществляют с помощью магнитных пускателей КМ. На лабораторном стенде у магнитных пускателей имеется два замыкающих контакта, один размыкающий и катушка управления. Обозначение контактов представлено на информационных бирках на панели стенда.

При необходимости имитации работы различных датчиков перемещения, уровня, температуры в лабораторном стенде используют тумблера SQ, которые состоят из 4 контактов (2 – замыкающих, 2 – размыкающих), связанных между собой механической связью.

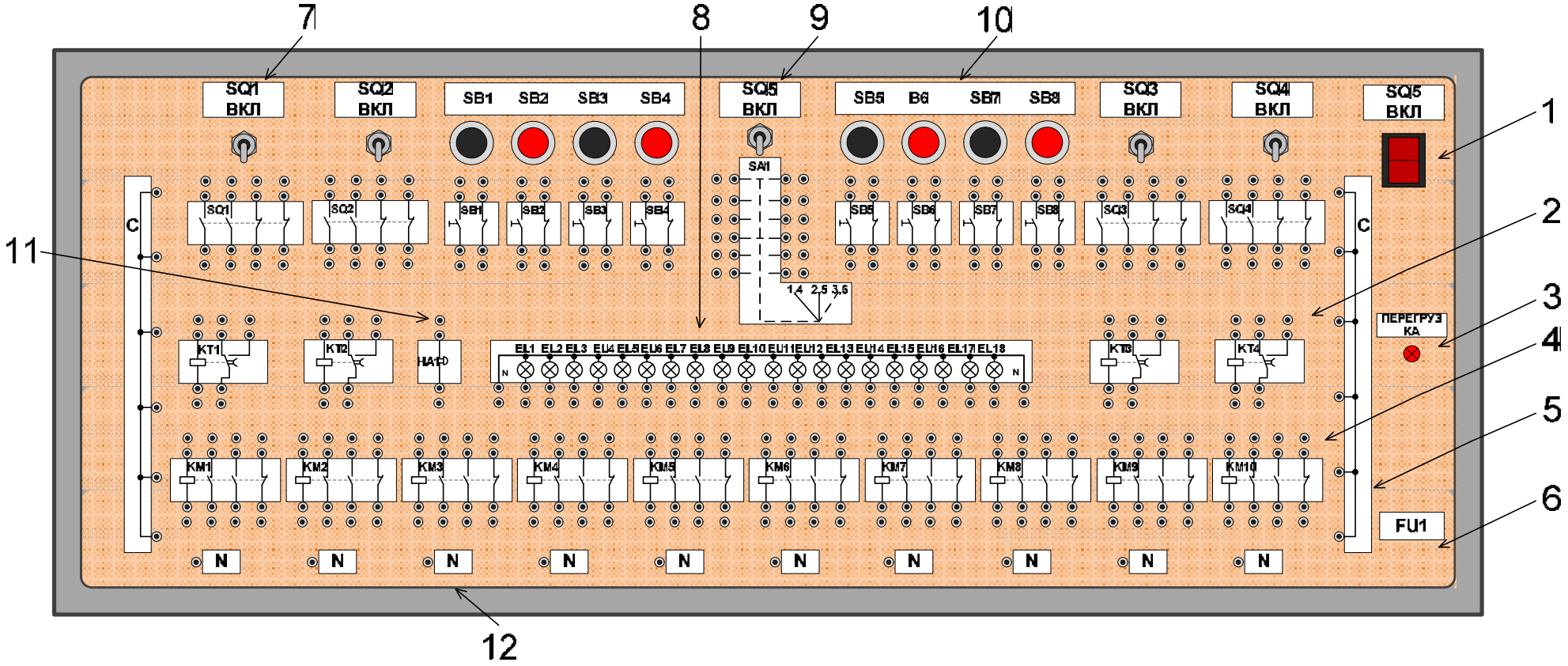


Рисунок 2.10 – Внешний вид лабораторной установки.

На лабораторном стенде студенты собирают разработанные по технологическому заданию релейно-контактные схемы. На рисунке 2.11 представлена схема сборки нереверсивного звена на панели стенда.

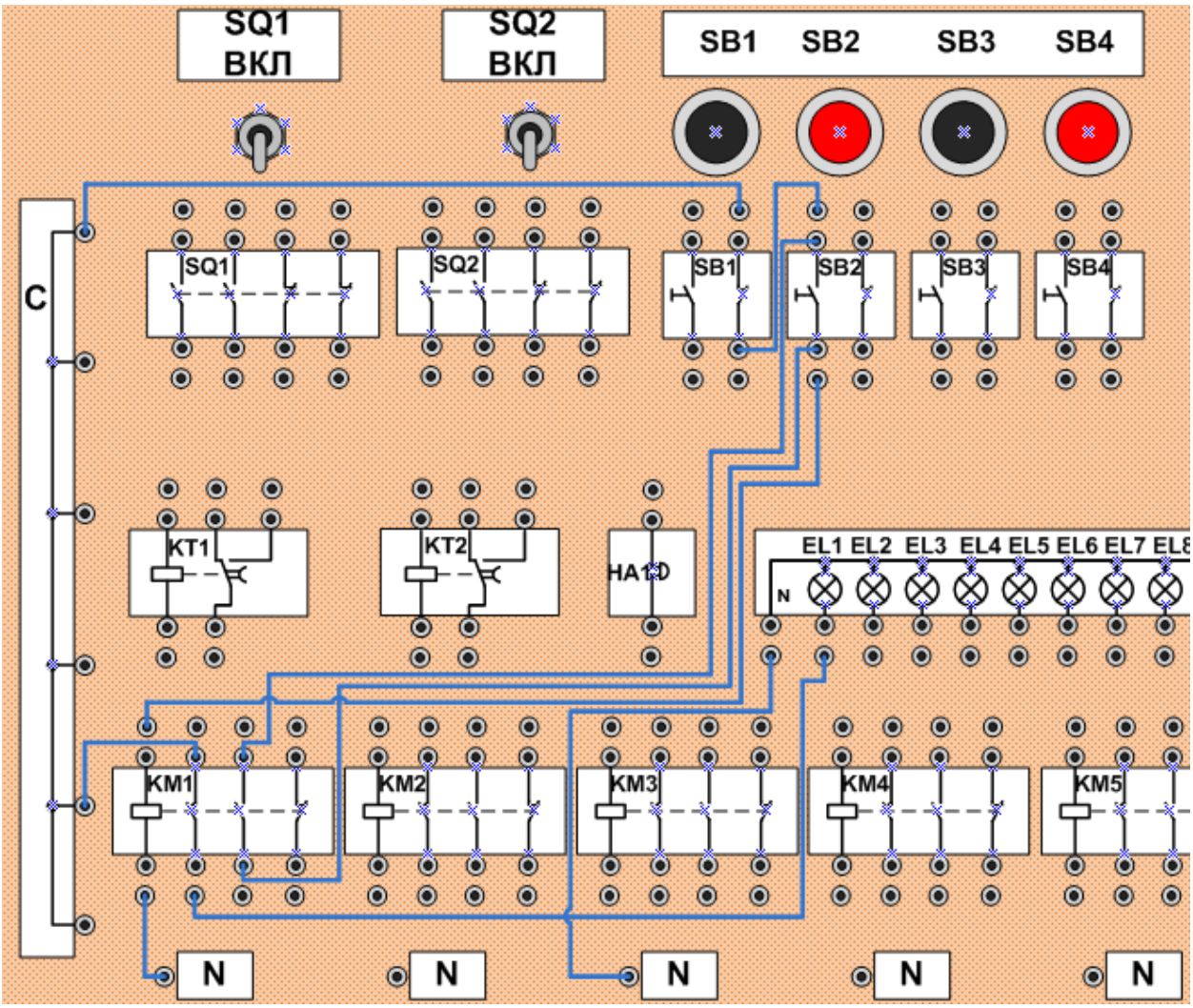


Рисунок 2.11 – Внешний вид сборки схемы на лабораторной установке.

2.4 Разработка схемы управления линии предварительной очистки зерна на базе релейно-контактных схем

Линия предварительной очистки зерна включает в себя: ковшовую норию, молотковую дробилку, скребковый транспортер и бункер для хранения зерна. Поточная линия производит перемещение продукта ковшовой норией на дробилку, в которой зерно измельчается и далее скребковым транспортером загружается в бункер для зерна. В бункере установлен датчик уровня, при срабатывании которого линия полностью отключается. На рисунке 2.12 представлена технологическая линия [12].

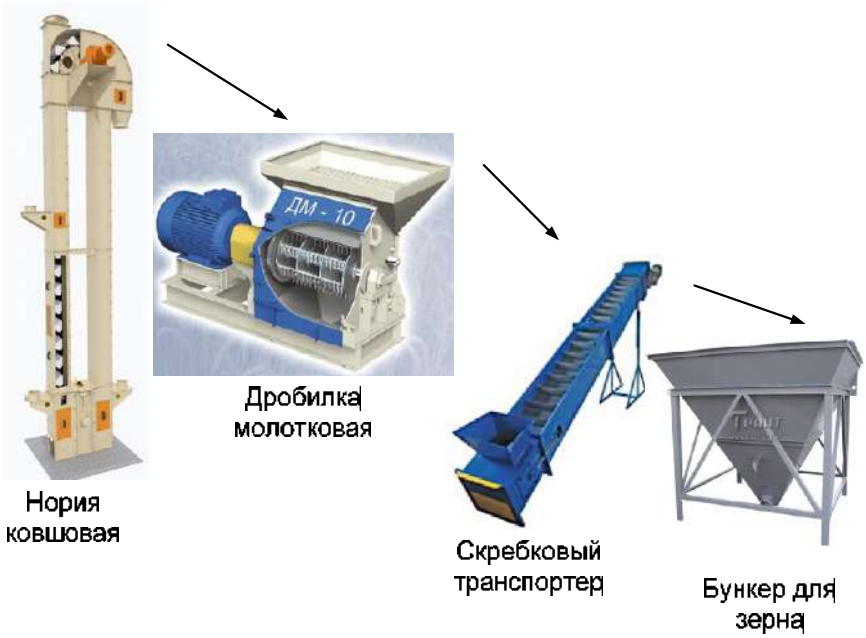


Рисунок 2.12 – Внешний вид линии предварительной обработки зерна.

Для правильного составления схемы управления линии необходимо сначала рассмотреть механизмы линии, изучить их особенности работы и только после этого приступать к составлению алгоритма управления.

Дробилка ДМ-10 обеспечивает измельчение как зерновых компонентов комбикорма (пшеница, ячмень, кукуруза, горох и т.д.), так и не зерновых (шрот, жмых, травяная мука, ракушка и т.д.). Дробилка ДМ-10 состоит из следующих основных сборочных единиц: рамы, корпуса, ротора, бункера, двух решет и привода. Дробилка комплектуется сменными решетами с диа-

метром отверстий 4,8мми 6,3мм.Рама предназначена для установки дробилки, представляет собой сварную конструкцию из швеллеров, пластин и косынок.

Корпус дробилки – сборная конструкция, включающая в себя переднюю и заднюю стенки, три перемычки, две опоры, вставку и полки. Для предохранения дробилки в случае взрыва пыли в камере дробления в передней и задней стенках выполнены по два предохранительных отверстия, закрываемых резиновыми крышками.

Ротор представляет собой сборную конструкцию из вала с приваренными пятью дисками. В шести отверстиях дисков при помощи втулок установлены оси, на которых в определенном порядке подвешены шарнирно 120 молотков.

Бункер служит для сглаживания пульсирующего потока зерна и очистки его от посторонних предметов. Бункер снабжен решеткой и магнитным сепаратором. На одной из боковых стенок бункера установлен датчик потока или уровня, сблокированный с приводами дробилки и загрузочного механизма, обеспечивающий возможность регулировки загрузки дробилки.

Привод включает в себя электродвигатель, втулочно-пальцевую муфту и защитный кожух.

Нория ленточного типа ковшовая предназначена для вертикального перемещения зерна и продуктов его переработки. Нория состоит из бесконечного тягового элемента с прикрепленными к нему ковшами, являющимися рабочим элементом. Тяговый элемент огибает приводной и натяжной барабаны. Тяговый элемент, приводной и натяжной барабаны заключены в металлическом кожухе, верхняя часть которого носит название головки, нижняя – башмака. Головка и башмак соединены норийными трубами. Сыпучий груз подается в башмак через загрузочный носок. При огибании тяговым элементом приводного барабана происходит разгрузка ковшей и груз через носок удаляется из головки.

Для предотвращения раскачивания тягового элемента, просыпания груза из ковшей и обеспечения необходимого натяжения тягового элемента нижний барабан связан с натяжным винтовым устройством. Барабан головки приводится в движение от электропривода через клиноременную и цепную передачи.

Взрыворазрядитель предназначен для отвода взрывной волны в случае ее возникновения. Аспирация нории осуществляется посредством аспирационной трубы, путем отсоса запыленного воздуха из нории в аспирационную сеть предприятия.

Дополнительно: для предотвращения завала башмака транспортируемым продуктом на норийной трубе устанавливается датчик подпора ВБЕ-Ц30-96-2111-3А на расстоянии 200–250мм выше фланца башмака. При превышении допустимого уровня продукта размыкаются контакты цепи управления электродвигателя, приводящего в движение транспортер, подающий продукт в элеватор. Транспортер останавливается, а элеватор продолжает работать и уменьшает подпор груза в носке.

Контроль скорости при пробуксовке или обрыве ленты осуществляется при помощи датчика скорости ВБИ-М30-49-2111-Л. При уменьшении частоты вращения барабана индуктируемый ток в датчике ВБИ-М30-49-2111-Л уменьшается, что приводит к срабатыванию реле и отключению механизмов, подающих продукт в элеватор.

Наша поточная линия не рассматривает механизмы, стоящие впереди нории, поэтому при составлении схемы особенности работы ковшовой нории мы учитывать не будем.

Скребковые транспортеры предназначены для транспортирования мелкокусковых и сыпучих материалов. Он состоит из приводной и натяжной секции, между которыми расположены промежуточные прямые и изогнутые секции, внутри которых расположена подвижная цепь со скребками. При запуске приводной станции транспортируемый продукт перемещается с одного

конца конвейера (транспортера) на противоположный конец при помощи подвижной цепи со скребками.

На основании рассмотренных механизмов линии и их работы выдвигают требования к схеме управления:

1) перед запуском линии дробления предусмотреть звуковую сигнализацию;
2) электродвигатели нории, дробилки, транспортера должны быть защищены от перегрузки;
3) предусмотреть задержку на включение ковшовой нории;
4) при достижении уровня в бункере линия должна отключаться. Контроль заполнения бункера осуществляется датчиком уровня;
5) схема управления должна иметь режим «Рабочий стоп»;
6) схема управления должна иметь режим пусконаладочных работ;
7) схема должна иметь световую индикацию работы механизмов.

Порядок выполнения работы:

1) внимательно изучить технологический процесс предварительной обработки зерна;
2) с учетом технологических требований, предъявляемых к схеме управления процессом обработки зерна, необходимо разработать электрическую релейно-контактную схему управления;
3) проверить разработанную схему управления на работоспособность на существующем стенде;
4) составить отчет о проделанной работе.

Схему управления рассмотренного технологического процесса можно реализовать с помощью типовых релейно-контактных блок-схем. Для реализации выдвинутых требований, используют блок-схемы: пускосигнального звена, именно с него и начнется схема управления, нереверсивного управления электродвигателем и звено рабочего стопа. Основной задачей остается

связать уже известные нам блок-схемы друг с другом и выстроить алгоритм управления.

Реализацию схемы управления начнем с переключателя режимов работы SA1. Именно он позволяет выбирать режим работы: автоматический – основной режим работы, ручной – режим пусконаладочных работ.

Режим пусконаладочных работ заключается в подаче питания через кнопки с фиксацией SB4-SB6 к катушкам магнитных пускателей механизмов линии в обход всей логики управления. В этом режиме оператор сам принимает решение по длительности работы линии или какого-то отдельного механизма, контроль заполнения бункера осуществляется только визуально.

Как правило, этот режим работы применяется либо при аварийных режимах работы, когда логика управления нарушена и необходимо завершить технологический процесс без утраты продукта на линии, либо при пусконаладочных работах, когда после ремонта какого-то механизма линии необходимо запустить только лишь его, а не все механизмы линии.

После переключателя режимов работы в схему управления включен блок пускосигнального звена, который позволяет, с задержкой по времени, одновременно отключить звонок и включить механизм скребкового транспортера.

При составлении релейно-контактных схем последовательность включения или отключения механизмов реализуется посредством замыкающих контактов магнитных пускателей. Так в нашем случае, если питание присутствует на катушке магнитного пускателя КМ1 (скребковый транспортер), то соответственно через контакт КМ1.1 питание также будет и на катушке магнитного пускателя КМ2 (молотковая дробилка).

Одновременно все механизмы линии запускать нецелесообразно, поскольку в процессе работы может возникнуть такой режим работы, когда электропривода двух механизмов линии еще не вышли на свой номинальный режим работы, а на них уже, через головной механизм, подается продукт, что приводит к аварийной остановке линии. Поэтому в схеме управления пита-

ние на катушку магнитного пускателя КМ3 головного механизма подается с временной задержкой, реализованной релем времени КТ2.

Механизмы линии все включены, осуществляется работа. Иногда во время работы наступает момент, когда бункер еще не полный, а линию необходимо отключить. В этом случае в схеме управления используют блок «рабочего стопа», который позволяет произвести отключение всех механизмов линии в правильной последовательности (по направлению движения продукта по линии). Так при нажатии на кнопку SB3 включается промежуточное реле KV2, размыкающий контакт которого KV2.2 разрывает цепь с катушкой КМ3, отключается головной механизм линии. При этом реле времени КТ3 производит отчет времени работы линии на очистку механизмов от продукта. После определенного времени контакт реле времени КТ3.1 разрывает цепь с промежуточным реле KV1, контакт которого является шунтирующим кнопку пуска. Это приводит к отключению всей схемы управления и, как следствие, остановке механизмов линии. Аналогичный алгоритм работы схемы управления при срабатывании датчика уровня в бункере SL1.

Защита электродвигателей линии от перегрузок в представленной схеме управления реализуется посредством размыкающих контактов тепловых реле КК1.1-КК3.1. На рисунке 2.13 изображена релейно-контактная схема управления линии предварительной очистки зерна.

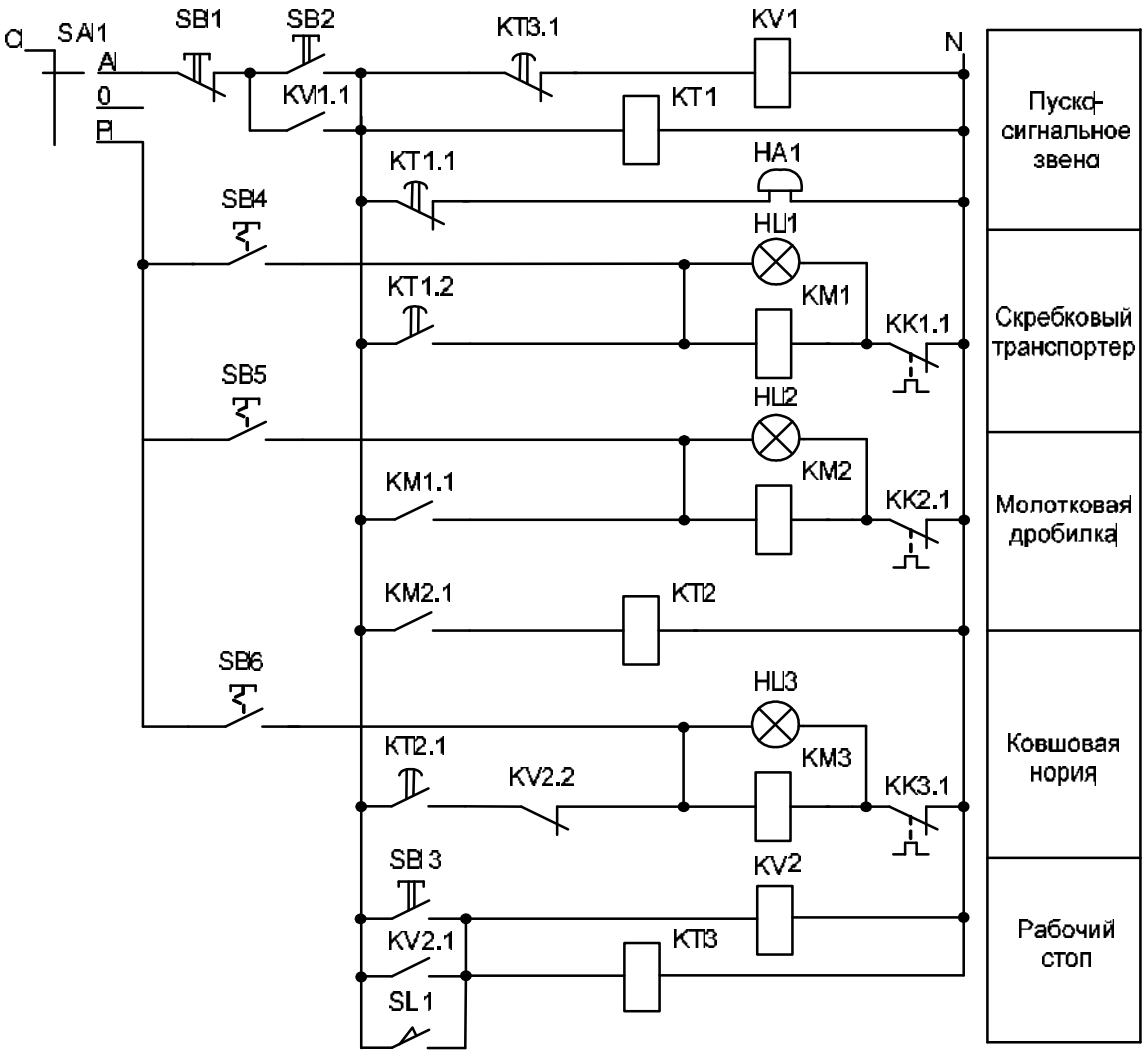


Рисунок 2.13 – Релейно-контактная схема управления линии предварительной очистки зерна.

3. ПРОГРАММИРУЕМЫЕ РЕЛЕ

Одним из основных путей повышения производительности применяемых в промышленности рабочих машин, улучшения качественных показателей выпускаемой продукции и снижения энергопотребления является применение средств автоматизации оборудования. Эти средства не только повышают эффективность производства, но также освобождают человека от утомляющей его работы по контролю состояния технологического процесса и формированию управляющих воздействий на исполнительные органы рабочих машин.

В ряду таких устройств особое место занимают электронные программируемые реле, или, как их еще называют, интеллектуальные реле.

Программируемые реле являются особым видом управляющих мини-ЭВМ, которые можно назвать управляющими логическими машинами последовательного действия. Программируемые реле имеют специфичные черты.

3.1 Программируемые реле серии EASY фирмы «Moeller»

3.1.1.Общие сведения о программируемых реле серии EASY.

Серия программируемых реле EASY довольно широка в применении в сельском хозяйстве и промышленности. Она представляет собой универсальную систему программируемых реле, устройств отображения и управления, компактных контроллеров. Управляющие реле EASY, многофункциональный дисплей MFD-Titanи новый контроллер easyControl в основе своей используют одну и ту же концепцию и позволяют решать широкий спектр задач автоматизации от элементарных схем до сложных технологических процессов. Базовые модули позволяют подключаться к шинам передачи данных easy-NET, CANopen и Ethernet. Также доступны для использования стандартные модули расширения (входы/выходы), а также моду-

ли с дисплеем и кнопками и без дисплея и кнопок для передачи данных по шинам ASInterface, DeviceNet, CANopen, ProfiBusи Ethernet [22].

Рисунок 3.1 – Внешний вид реле и его применение.

Рисунок 3.2 – Внешний вид реле EASY, панель оператора MFD Titan и Easy Control в технологическом процессе.

Любой человек, способный читать электрические схемы, сразу почувствует легкость в общении с реле EASY. Программирование представляет собой создание электрических схем в виде схем соединений. EASY500/700 и EASY800 предоставляют широкие возможности для решения задач управления в бытовых и промышленных приложениях.

Реле EASY – это:

• наиболее обширная гамма управляющих реле на рынке автоматизации;

• возможность использования для широкого круга задач автоматизации;

• оптимизация затрат для вашего приложения;

• возможность легкого и простого программирования и изменения параметров непосредственно с устройства.

В дополнение к функциям реле EASY MFD Titan предоставляет мощные возможности визуализации.

Дисплей MFD Titan – это:

• визуализация и управление в одном устройстве;

• мощный и недорогой;

• возможность расширения и объединения в сеть нескольких устройств;

• возможность использования в тяжелых промышленных условиях благодаря высокой степени защиты IP65.

Easy Control – это логическое продолжение линейки реле EASY. Программное обеспечение easy Soft, CoDeSys, соответствующее стандарту IEC61131-3, позволяет использовать новый контроллер easy Control для тех приложений, которые раньше были не доступны для EASY. Easy Control – это:

• мощный компактный контроллер с встроенным дисплеем или без него;

• возможность добавления модулей расширения и объединения в сеть.

Использование продуктов серии EASY можно применять для решения самых различных задач:

Освещение витрин и управление рекламами.

1. Автоматическая коммутация освещения и рекламы в зависимости от освещенности или по времени.

2. Активация освещения во время тревоги или по событию.

3. Управление рекламами согласно различным алгоритмам.

Освещение зданий.

1.Включение /выключение освещения централизовано и децентрализовано при помощи функции импульсного реле.

2. Использование функции реле времени и функции таймеров для управления по времени.

Теплицы.

1. Управление окнами в крышах, отоплением, вентиляторами, поливкой и освещением в зависимости от температуры, влажности и освещенности.

2. Настройка требуемых параметров прямо на дисплее прибора.

Управление насосными станциями.

•Управление насосными станциями питьевой воды и автоматизация обработки воды.

•Управление насосными станциями водоочистительного сооружения сточных вод.

•Увеличение или снижение количества работающих насосов в зависимости от высоты уровня.

•Возможность управления насосом при помощи аналогового выхода и частотного преобразователя.

•Возможность отправки SMS-сообщений в случае тревоги или неисправности.

Автоматический ввод резерва.

•Решение задачи автоматического ввода резервного питания двухвводной системы питания, при перебое в работе одного из них, при помощиуправления автоматическими выключателями с осуществлением их взаимной блокировки.

•Возможность автоматического возврата при возобновлении напряжения на неисправном приводе.

Управление конвейерами.

•Различные виды управления конвейерами, например, постепенный разгони остановка отдельных лент конвейера.

3.1.2 Особенности программируемых реле.

1.Облегчение программирования, которое выполняется, как правило, в форме предварительно составленной релейной электрической принципиальной схемы. Задание программы ведут на клавиатуре ручного ввода в символах принципиальной релейной схемы путем нажатия на кнопки и последовательного выбора замыкающего или размыкающего контакта, катушки аппарата и т.п.Здесь символы I без инверсии соответствуют замыкающим контактам, а с инверсией – размыкающим. Символ Q обозначает выходной сигнал управляющего устройства (рис.3.3).

2.Возможность использования непосредственно в цеховых условиях (большая помехозащищенность), гальваническая развязка от внешних цепей, расширенный диапазон допустимых условий эксплуатации.

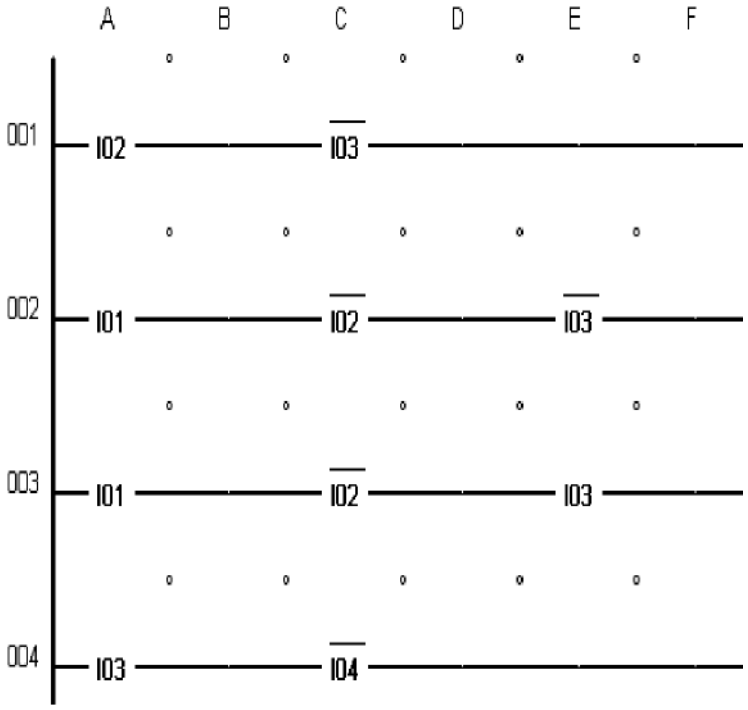


Рисунок 3.3 – Внешний вид релейной электрической схемы графического вида в программе EASY-SOFT6 Pro.

Примечание: реле, выполненные на переменное напряжение 115/240 В, уровень входного сигнала от 0 до 40 В воспринимают как нулевой, а от 79 до 246

В как единичный. Ток входных цепей не превышает 0,5 мА при напряжении 230В.

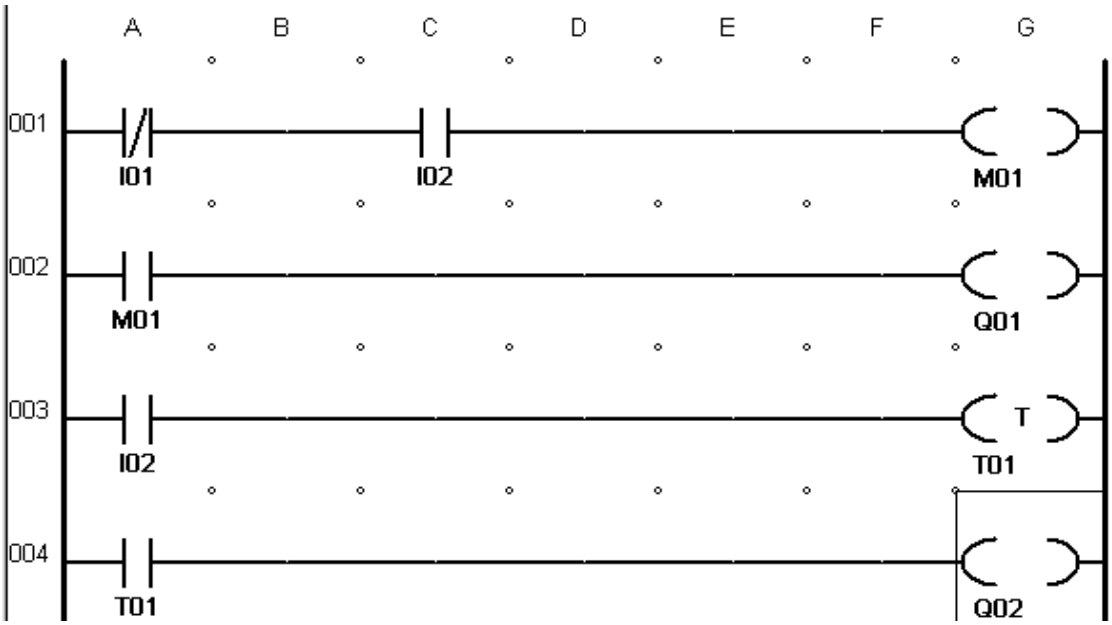


Рисунок 3.4 – Внешний вид релейной электрической схемы ANCI/CSA в программе EASY-SOFT6 Pro.

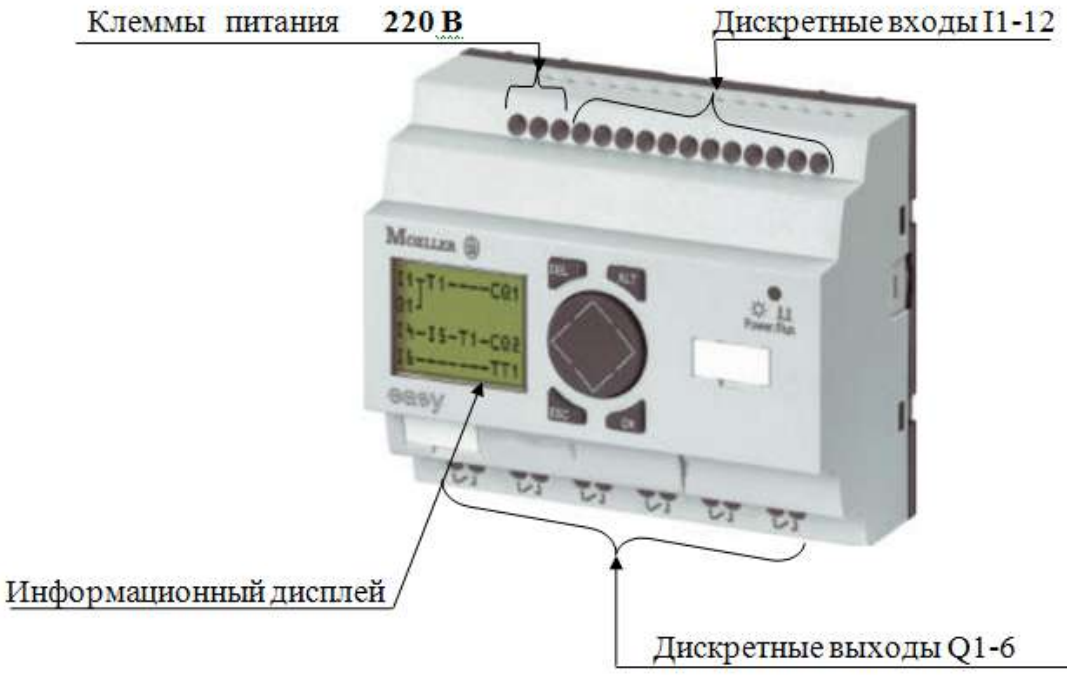


Рисунок 3.5 – Общий вид реле EASY-719.

Выходные реле и их контакты обозначаются на схемах буквой Q. У реле EASY и модулей расширения установлено по 4 выходных реле и, соответ-

ственно, 4 выходных контактов. Физически каждый контакт выходного реле является замыкающим (нормально разомкнутым). В программах можно использовать виртуальные размыкающие (нормально замкнутые) контакты этих реле. Можно также использовать общую точку для всех контактов, а напряжение питания для нагрузки использовать любое необходимое, как показано на рисунке 3.6.

На этом же рисунке показана нагрузочная способность контактов при управлении активной и индуктивной нагрузкой и допустимое за срок службы число включений. Нагрузка в виде электрических лампочек накаливания, хотя и является активной, имеет свои ограничения, связанные с наличием бросков пускового тока при включении.

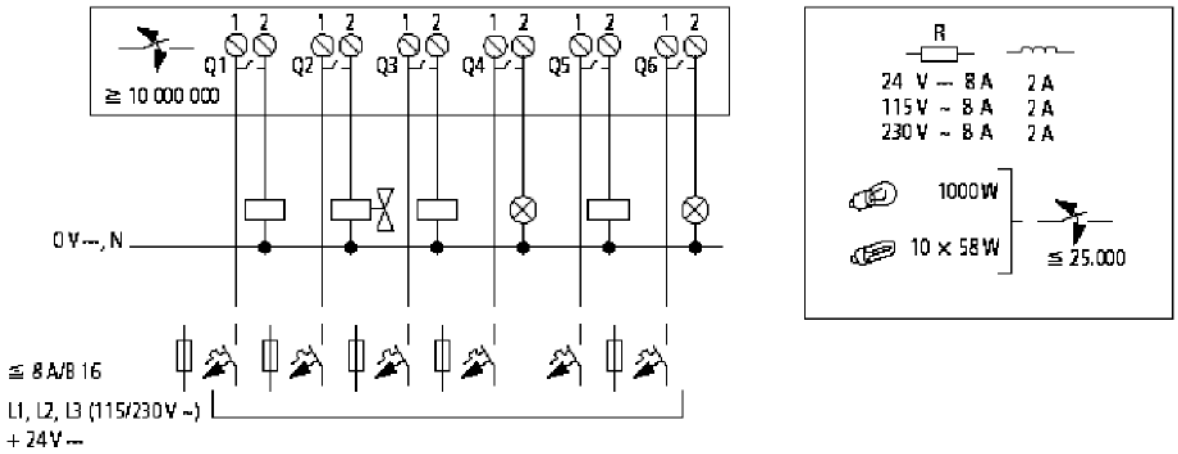


Рисунок3.6 – Пример схемы соединения выходных релейных цепей EASY-719.

3.1.3 Меню программируемого реле.

Меню программируемого устройства задается посредством операционных кнопок реле, внешний вид которых отображается на рисунке3.7.

Кнопка курсора, кроме этого, может выполнять функции Р кнопок (Р кнопки – это четыре дополнительных кнопки для управляющих команд в режиме RUN).

Рисунок 3.7 – Операционные кнопки реле.

На рисунке 3.7 обозначены: DEL – удалить объект в схеме;ALT – специальные функции в схеме соединений, состояние дисплея;ESC– следующий уровень меню, сохранить введенные данные; КНОПКИ КУРСОРА (вверх-вниз, влево-вправо) – движение курсора, выбор пункта меню, установка номера катушек, контакта, значение функции.

Нумерация кнопок начинается от левой кнопки по часовой стрелке:

- P1– левая кнопка;
- P2– верхняя кнопка;
- P3 – правая кнопка;
- P4– нижняя кнопка.

Основным экраном (дисплеем) является Status display, или дисплей состояния реле (рисунок3.8).В этом кадре показывается состояние входов и выходов реле, режим работы и время в одном из двух возможных форматов. Номера входов (I), на которые поданы управляющие сигналы, и номера включенных выходов (Q) подсвечиваются. Возможные режимы работы реле: STOP (СТОП) или RUN (РАБОТА). В режиме STOP производится ввод программы и настройка реле, в режиме RUN выполняются рабочие операции. Индикатор питания (POW) на лицевой панели реле в режиме STOP горит ровным светом, в режиме RUN –прерывистым.

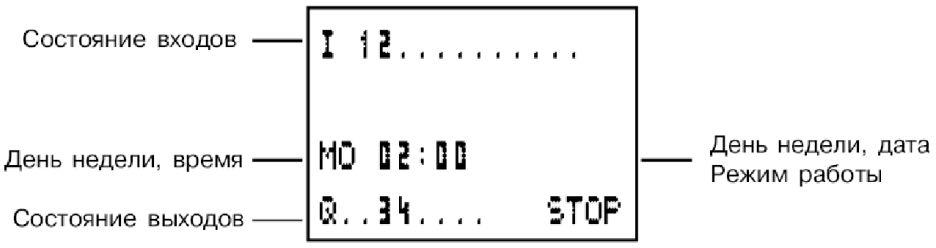


Рисунок 3.8 – Дисплей состояния реле.

3.1.4 Работа со схемой соединений.

Схема соединений размещается на специальном поле. Поле представляет собой скрытую сетку, которая содержит 256 строк и на которую можно наносить схему соединений. В каждой строке может быть размещено 4 контакта и 1 катушка. Схема должна состоять из коротких, логически завершенных строчек. Каждая строчка должна начинаться контактом, а заканчиваться катушкой какого-либо реле. Между контактами и катушкой прокладываются связывающие их линии. Линии могут ответвляться вверх и вниз и соединяться с линиями других строчек. Как указывалось в обзоре, ЖК-дисплей реле имеет ограниченные возможности по размещению информации, поэтому на дисплее показывается только часть поля, на котором изображается схема.

Для улучшения читабельности применяются крупные знаки, поэтому в пределах одного экрана на дисплее можно наблюдать три цепи схемы и строку состояния. В одной строке схемы одновременно может быть отображено два контакта или один контакт и катушка. Перемещение от начала до конца строки и по строчкам производится с помощью курсора. Контактам можно придавать прямое и инверсное значение. Например, контакт релейного выхода Q обозначает замыкающий контакт. Этот же контакт можно обозначить как $\bar{Q}$ (нажатием кнопки ALT), и в этом случае он начинает исполнять роль размыкающего контакта. Количество контактов одного и того же элемента и одного названия в схеме не ограничивается. Хотя физически, например, у выходного реле Q01, имеется всего один замыкающий контакт. К катушкам, наоборот, применяется строгое требование: одну катушку можно использовать только один раз и в одной цепи. Катушка здесь рассматривается как не-

который аналог исполнительного органа, рассматриваемого функционального элемента, и при программировании обозначается различными символами, например, {Q01. Многие функциональные элементы имеют по две и более катушек. Например, счетчик С имеет 4 катушки: счетную, изменения направления счета, установки начального значения и сброса в ноль. Каждому модулю или катушке соответствует один или несколько контактов, имеющих различное назначение. Например, у выходного реле и реле-маркера имеется только один одноименный контакт. У счетчика имеются четыре контакта, среди них, например, С01СУ, замыкающийся при переполнении счетчика и С012Е, замыкающийся, когда на выходе счетчика устанавливается ноль. По умолчанию в начале кадра устанавливается контакт цифрового входа I1 (на дисплее I01). В кадре присутствует также курсор в форме мерцающего черного прямоугольника. В нижней части кадра находится строка состояния, в которой обозначаются координаты курсора и емкость свободной памяти (в режиме работы на этом месте появляется RUN):

- L– номер строки схемы (от 1 до 256);
- С – номер позиции контакта или катушки в строке (от 1 до 5);
- В – количество свободных бит памяти (начальное значение 7944).

Номера катушек и их свойства выбираются из предлагаемого списка.

Программирование осуществляется «прокручиванием» списка и выбором требуемого наименования.

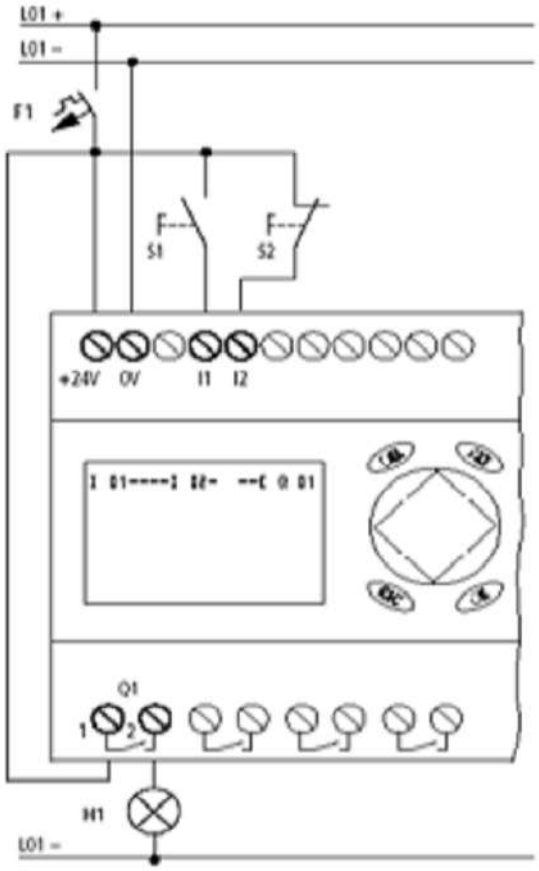


Рисунок 3.9 – Схема включения осветителя.

На экране реле данная схема принимает вид:

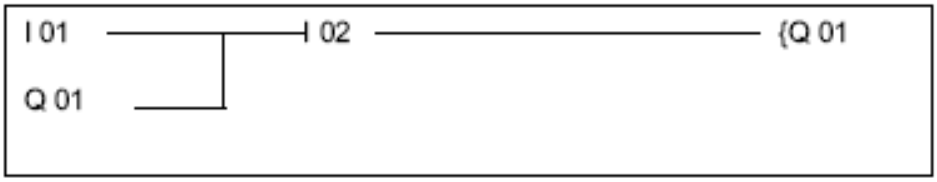


Рисунок 3.10 – Пример набора схемы управления осветителем.

Тестирование (проверка) собранной схемы производится в режиме RUN. Входим в меню ПРОГРАММИРОВАНИЕ и, например, при управлении осветителем, нажимая на кнопки S1 и S2, прослеживаем правильность срабатывания выходного реле Q1. Линии соединений, по которым в данный момент «протекает ток», выделяются двойной линией. При возврате к дисплею состояния можно проконтролировать поступление сигналов на входы I1 и I2, а также активирование выхода Q1. Кроме этого, можно услышать характер-

ный щелкающий звук при включении реле Q1. Применение Р-кнопок создает дополнительные удобства для наладки и тестирования системы.

Функциональные реле могут иметь одну или несколько катушек и, как минимум, один контакт. При отсутствии питания катушка находится в невозбужденном, не активированном состоянии, или состоянии 0. При подаче (имитации подачи) питания катушка переходит в возбужденное, активированное состояние, или состояние 1. Если нормальное состояние контакта при невозбужденной катушке открытое или разомкнутое (состояние 0), то при возбуждении катушки контакт замыкается, то есть переходит в состояние 1. В программной форме нормальное состояние катушки или контакта можно изменять на противоположное.

В программируемых реле EASY кроме обычных операций включения и выключения реле могут применяться более сложные функции. Каждой из катушек может быть присвоена одна из следующих семи функций (таблица3.1).

Таблица 3.1 – Обозначения и функции катушек.

Изображение на дисплее	Функция катушки	Примеры обозначения
[	Замыкатель (реле, контактор)	[Q01
∫	Импульсное реле, эффект триггера	∫ Q01
S,R	Установка в 1 (S) или в 0 (R). Реле с механической блокировкой	SQ01, RQ01
]	Инверсный замыкатель, негативная катушка	] Q01
⊓	Формирователь импульса по заднему фронту	⊓ M
⊔	Формирователь импульса по заднему фронту	⊔ M

Катушка с функцией замыкателя (реле, контактор) [.

Выходной сигнал следует немедленно после подачи входного сигнала, и реле работает как замыкатель (рисунок 3.11).

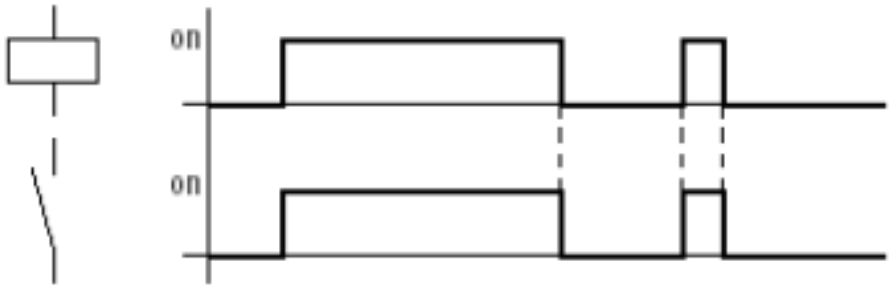


Рисунок 3.11 – Диаграмма сигналов функции замыкателя.

Импульсное реле  .

Реле переключается каждый раз, когда входной сигнал изменяется от 0 до 1.

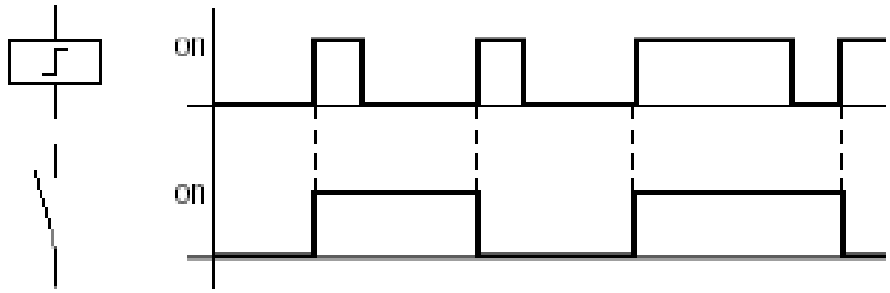
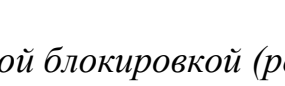


Рисунок 3.12 – Диаграмма работы импульсного реле.

Это реле выполняет функцию триггера или делителя на 2. При подаче на катушку 1-го импульса оно включается, при подаче 2-го импульса выключается.

Реле с механической блокировкой (реле с защелкой)  .

Для создания реле с такими свойствами используются функции S (SET) и R (RESET). Функция S в данном случае понимается как установка в 1, а функция R как сброс в 0 (рисунок 3.13).

Реле имеет следующий алгоритм работы:

- для включения реле достаточно короткого импульса на катушке S, далее реле остается включенным (становится на защелку), участок А диаграммы;
- выключение реле, происходит при подаче импульса на катушку R;

- если на катушке S сохраняется сигнал,то при включении R выходной контакт реле размыкается только на время действия сигнала R, участок B диаграммы;
- для выключения реле следует снять сигнал с катушки S и подать на катушку R.

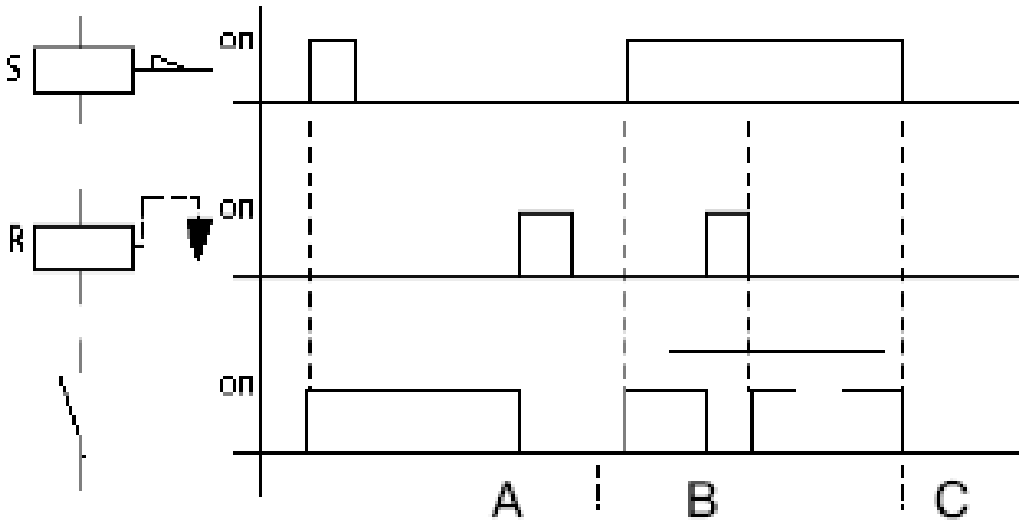


Рисунок 3.13 – Диаграмма работы реле с механической блокировкой.

Инверсная функция замыкателя (негативная катушка) .

Функция такого реле противоположная (негативная) по отношению к обычному реле. Диаграмма работы такого реле показана на рис. 3.14.

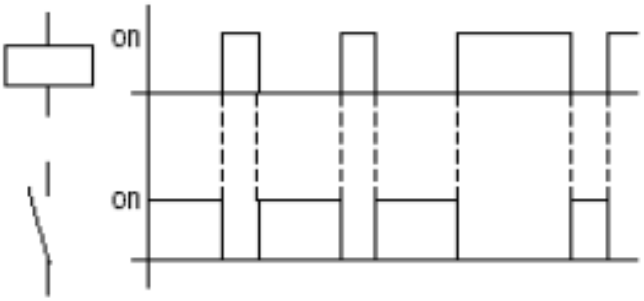


Рисунок3.14 – Диаграмма работы инверсного реле.

Реле-маркеры.

В реле EASY-719 имеются реле-маркеры, выполняющие роль вспомогательных, или промежуточных, реле, и имеющие обозначение М (таблица 3.2).

Таблица 3.2 – Характеристика модулей и элементов EASY-719.

Усл. обоз.	Название элемента	Наименование катушек и их функциональное назначение	Обозначение контактов и условие срабатывания
I	Цифровой вход	Нет	I01- внешний сигнал
Q	Цифровой выход	Q01- обычная катушка SQ01- с защелкой RQ01- сброс защелки	Q01 - релейный или транзисторный, от катушки Q01
M	Реле - маркер	M01- обычная катушка SM01- с защелкой RM01- сброс защелки	M01 - от катушки M01
C	Счетчик	CO1C- счетный вход CO1D- изменение направления счета C01SE- установка начального значения C01RE- сброс в ноль	C01OF- условие >, = C01FB- условие <, = C01CY- переполнение,C01ZE- ноль
T	Реле времени	T01EN- разрешение TO1ST- приостановка T01RE- сброс	T01Q1 - от катушки T01-по установленной функции

Реле делятся на несколько типов, в зависимости от объема информации, которой они управляют. Простейший вариант (реле М) выполняет функции обычного реле, имеющего два состояния контакта: замкнутое и разомкнутое. Имеется возможность использования до 32 реле-маркеров.

Реле времени (Т).

Для работы реле времени требуется 52 байта памяти. В реле EASY размещено 16 реле времени с различными настраиваемыми функциями. Реле времени применяется для изменения продолжительности замыкания и размыкания контактов. Регулирование выдержки времени возможно в диапазоне

от 5 мс до 99 час. 59 мин. При применении переменных величин выдержку времени можно увеличить до 596 час.

Реле времени устанавливается в схеме в форме катушек и контактов. Режим работы реле устанавливается через дисплей параметров реле. Реле запускается при включении катушки Т. Для предотвращения неопределенных состояний реле все катушки могут использоваться в схеме соединений только один раз.

Пример схемы соединения реле времени приведен на рисунке3.15.

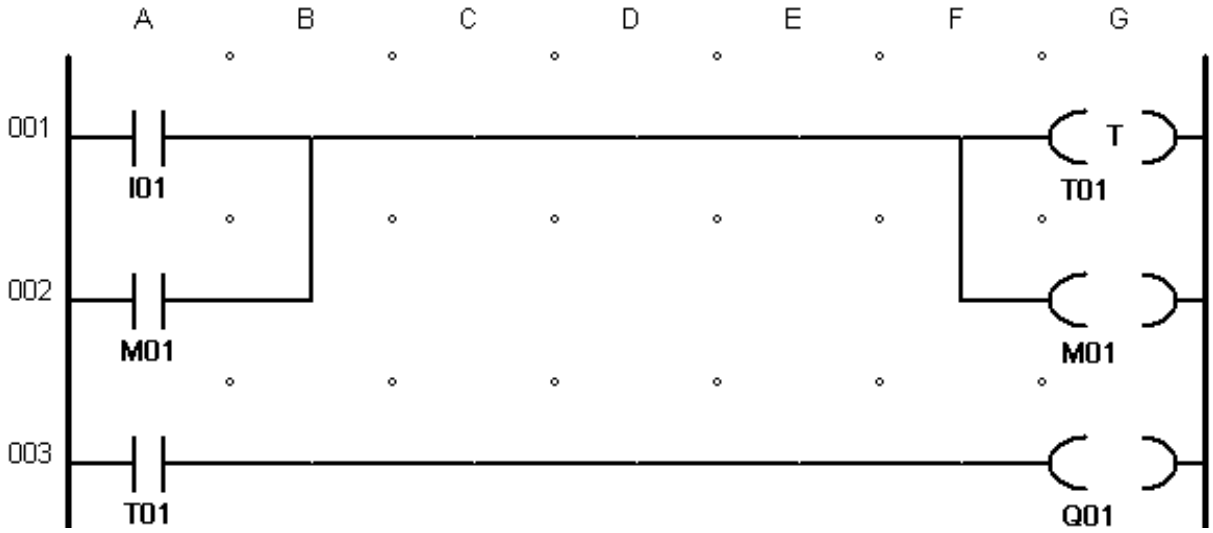


Рисунок 3.15 – Пример схемы соединения реле времени.

Таблица 3.3–Диапазоны установки времени.

Изображение на дисплее	Диапазон уставок	Точность
S 000.000	Секунды, от 0,005 до 2147483,645 с (596 ч) для констант и переменных величин	5 мс
M:S 00:00	Минуты: секунды, от 00,00 до 99,59 только для констант и переменных величин	1с
H:M 00:00	Часы: минуты, от 00,00 до 99,59 только для констант и переменных величин	1 мин

Таблица 3.4 – Режимы реле времени.

Обозначение	Выполняемая функция (режим работы)
X	Выдержка времени при включении
?X	Выдержка при включении, со случайным временем между нулем и уставкой
▮	Выдержка времени при выключении
?▮	Выдержка при выключении, со случайным временем между нулем и уставкой
X▮	Выдержка времени при включении и выключении
Л	Включение на заданное время и выключение (однопульсное включение). Запускается коротким импульсом
Ц	Пульсирующее включение реле с 2 настройками по времени. Время по входу >I1 задает работу, по входу >I2 паузу

Входы >I1 и >I2 могут быть представлены следующими операндами:

- константами;
- маркерами MD, MW, MB;
- текущей величиной ...QV другого функционального модуля.

Меню параметров функциональных модулей.

В отличие от простых релейных элементов, задание параметров которых производится непосредственно в схеме соединений, для функциональных модулей требуется вводить и редактировать большее количество параметров. Поэтому для настройки функциональных модулей и реле времени применяются специальные меню, располагаемые вне схемы соединений (рисунок3.16). Редактирование меню конкретного функционального модуля выполняется слева направо, сверху вниз. Перемещения по позициям дисплея и установка необходимых имен, переменных, констант и пр. осуществляются с помощью кнопок управления курсором и кнопки ОК. Знак « > », применяемый в меню, обозначает источник и направление передачи информации: >X

для входа и X> для выхода. Имена модулей, названия функций и входных/выходных величин будут рассмотрены при описании конкретных функциональных модулей.

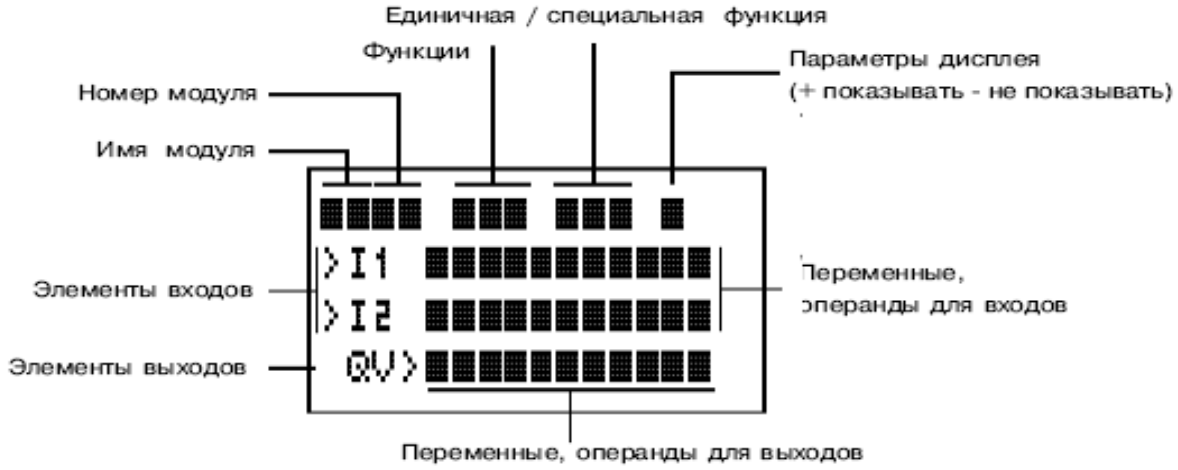


Рисунок 3.16 – Типовая структура меню для настройки функциональных блоков.

Войти в меню редактирования параметров функциональных модулей можно двумя путями. Первый раз в это меню мы попадаем автоматически из поля схемы соединений, как только установим первый контакт или катушку функционального модуля. Имя функции и номер в меню уже присутствуют, необходимо только закончить первую строчку меню, то есть указать выбранную функцию и доступ к параметрам дисплея. После нажатия ОК мы вновь возвращаемся к схеме соединений и продолжаем работу с ней.

3.1.5 Программирование реле EASY с использованием программного обеспечения «EASY-SOFT».

Конфигурация проекта.

В режиме «Проект» интерфейс EASY-SOFT (рисунок3.17) разделен на три части: панели инструментов 1, панели свойств 2, рабочего стола 3.

Проектом является комбинация из устройств и относящейся к ним схемы соединений. Выбрав необходимый тип программируемого реле, в панели

инструментов, необходимо перенести его на рабочий стол. Для этого щелкают левой кнопкой мыши на изображении необходимого устройства и, не отпуская кнопки, перетягивают его в рабочую область. Далее появится окно с выбором номера версии устройства. Номера версий устройств отличны друг от друга набором языков и шрифтов, используемых в устройстве [12].

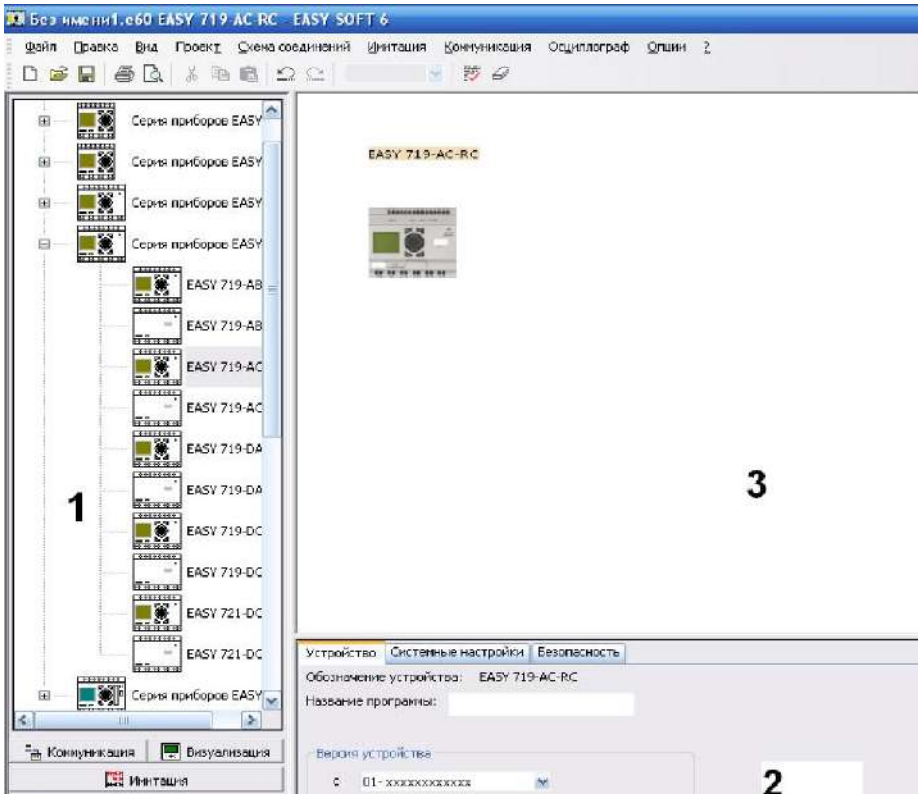


Рисунок – 3.17 – Интерфейс Easy Soft Pro в режиме «Проект».

Проектом является комбинация из устройств и относящейся к ним схемы соединений. Выбрав необходимый тип программируемого реле, в панели инструментов, необходимо перенести его на рабочий стол. Для этого щелкают левой кнопкой мыши на изображении необходимого устройства и, не отпуская кнопки, перетягивают его в рабочую область. Далее появится окно с выбором номера версии устройства. Номера версий устройств отличны друг от друга набором языков и шрифтов, используемых в устройстве.

В поле свойств находится перечень всех свойств выбранного устройства. К свойствам относятся, например, число входов и выходов, число маркеров, элементы времени и счета. Это поле необходимо для выбора устройства,

наиболее подходящего пользователю для создания схемы, в которой известно количество всех элементов, в том числе входов и выходов.

После помещения устройства на рабочую область в поле свойств будут находиться вкладки: режим работы, системные настройки, безопасность.

Схема соединений.

После выбора устройства необходимо создать схему соединений, для чего щелкнуть левой клавишей мыши на кнопке «Схема соединений» панели инструментов. В этом режиме интерфейс также состоит из панели инструментов, панели свойств и рабочего стола, называемого также окном схемы соединений (рисунок 3.18).

В области панели инструментов находятся все элементы, которые перетаскиванием при нажатой левой кнопке мыши можно поместить в рабочее окно схемы соединений.

I - вход основного устройства.

Q - выход основного устройства.

M - маркер, вспомогательное реле.

P - P кнопки.

Устройства EASY и MFD дают возможность запрограммировать четыре курсорные кнопки дополнительно в качестве контактов управления. В схеме соединений кнопки представлены как контакты P1 - P4.

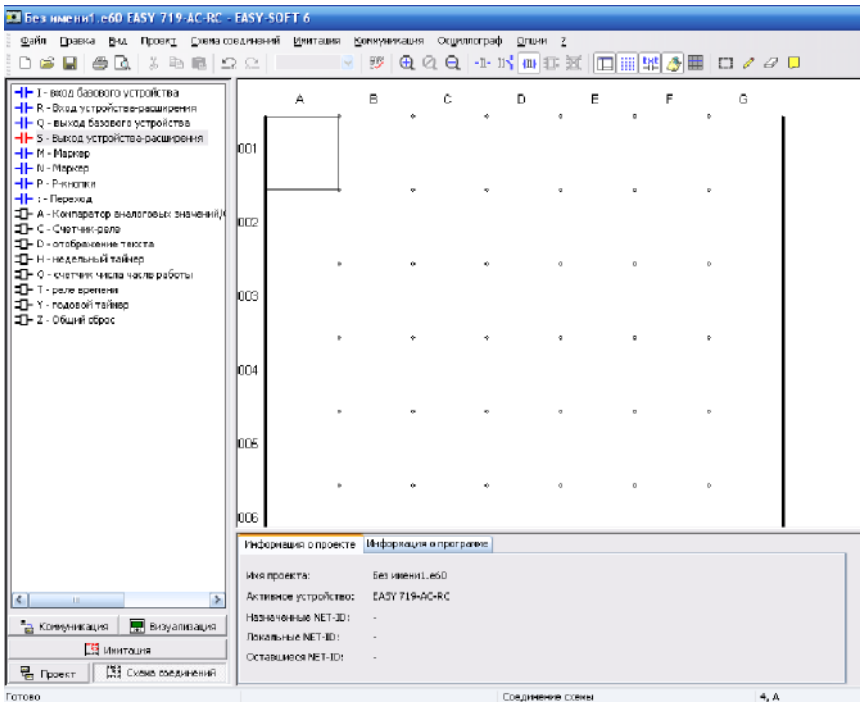


Рисунок 3.18 – Интерфейс Easy Soft Pro в режиме «Схема соединений».

При перетаскивании элементов в рабочее окно схемы соединений горизонтальные связи между ними формируются автоматически. Недостающие вертикальные связи могут быть добавлены с использованием инструмента «Карандаш» из панели инструментов, находящейся вверху окна (рисунок 3.19). Ненужные связи удаляются с помощью инструмента «Ластик». В каждой строке схемы можно располагать слева направо до четырех контактов и одну катушку.

Большинство элементов могут быть установлены в схему соединений как контакты и катушки, причем количество контактов каждого элемента не ограничено, а катушка должна быть установлена в схему только один раз. В окне схемы соединений каждый помещаемый туда элемент схемы остается помеченным квадратной рамкой, при этом в окне «Панель свойств» можно настраивать его параметры. К базовым параметрам можно отнести порядковый номер элемента, тип контакта (размыкающий или замыкающий), функцию катушки.

Можно также щелчком левой кнопки мыши выделить рамкой любой элемент схемы и настроить его параметры. Если элемент находится в области

контактов, то можно задать его номер (рисунок 3.20) и с помощью опциональной кнопки (в окне «Контакт») установить его как размыкающий или замыкающий. Если операнд находится справа в области катушек, то его настраивают через панель списка «Функция катушки».

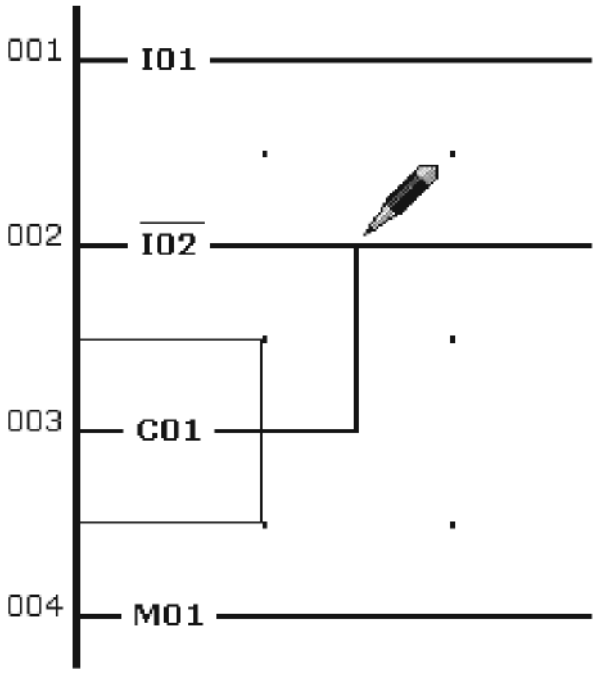


Рисунок 3.19 – Использование инструмента «Карандаш» для вычерчивания схемы соединений.

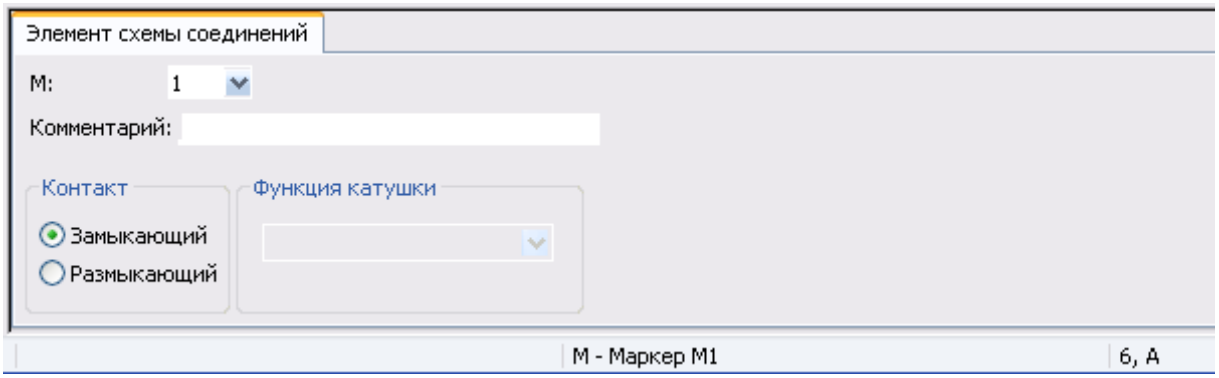


Рисунок 3.20 – Настройка параметров контактов.

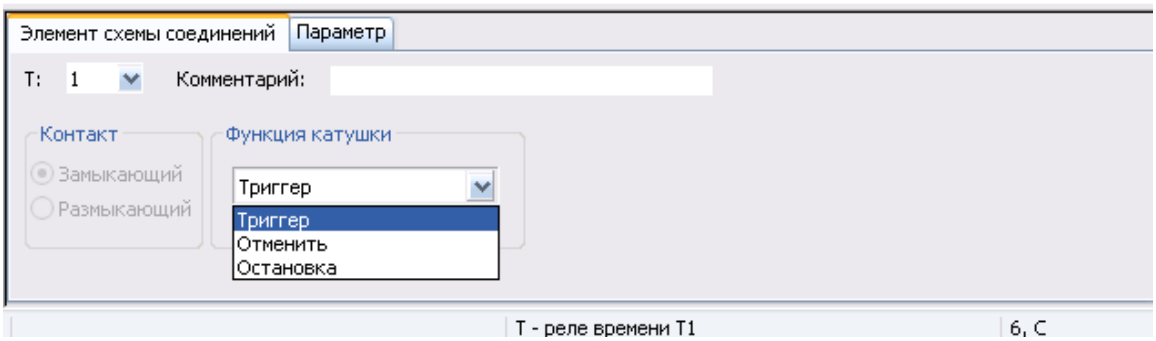


Рисунок 3.21 – Настройка параметров катушки реле времени.

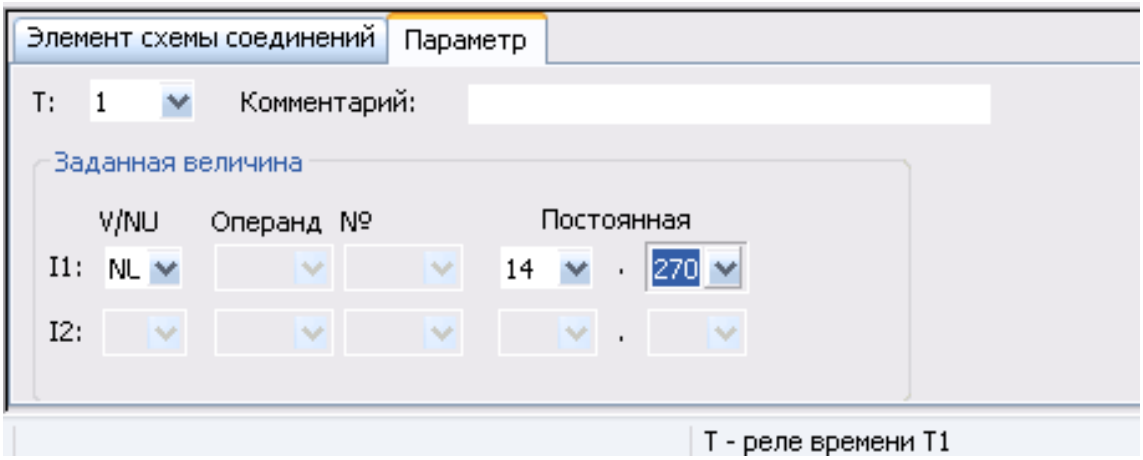


Рисунок 3.22 – Настройка диапазона времени реле времени.

Имитация.

Этот режим предназначен для проверки правильности работы созданной схемы управления. В этом режиме экран EASY Soft разделен на три части: панель инструментов, панель свойств, окно схемы соединений (рисунок3.23).

Панель свойств играет роль отображающего экрана, на котором можно наблюдать за состояниями входов, выходов, маркеров и дисплея.

Окно схемы соединений служит для иллюстрации состояния схемы во время ее работы, включаемые контакты и катушки подсвечиваются красным цветом.

В панели инструментов находятся следующие вкладки:

- принцип работы I; Входы I;
- аналоговые входы;

- цикл имитации;

- точка останова;

- показание.

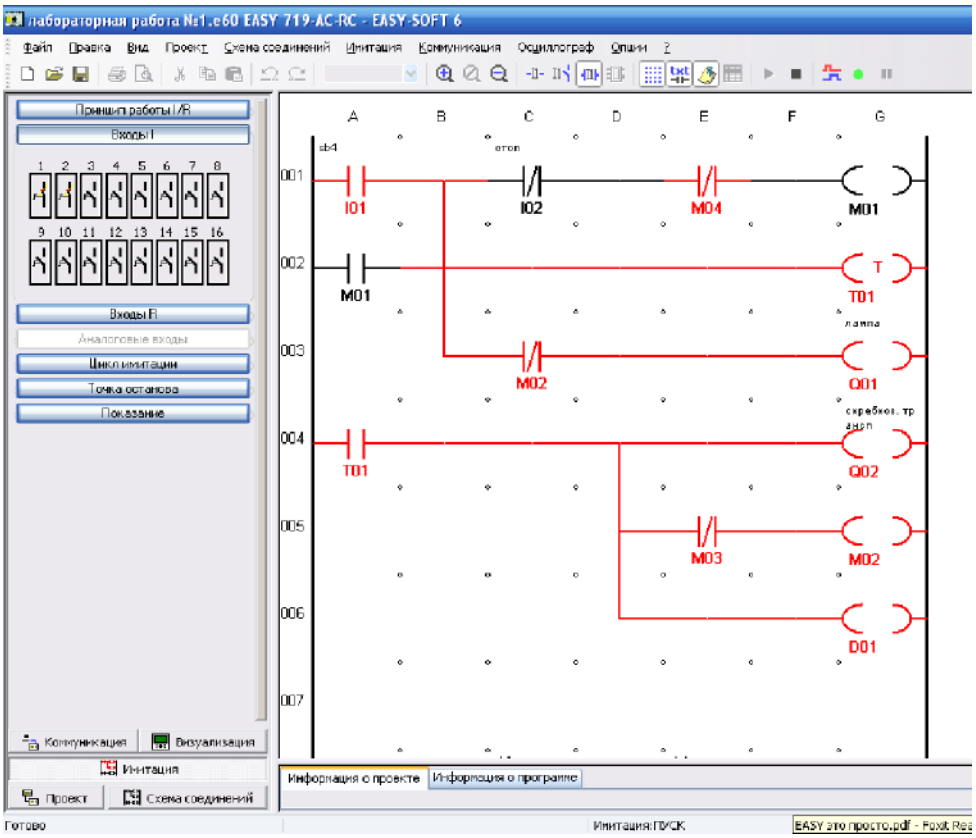


Рисунок 3.23 – Интерфейс Easy Soft Pro в режиме «Имитация».

Принцип работы I.

Вкладка дает возможность настройки состояний работы входов (I1...I16);а именно: размыкающий (нормально закрытый) с фиксацией или без, замыкающий (нормально открытый) с фиксацией или без. По умолчанию настройка всех входов установлена в положение замыкающий с фиксацией.

Установка связи с устройством (коммуникация).

После испытания схемы соединений ее нужно загрузить в устройство. Для этого необходимо подключить устройство с помощью кабеля EASY-PC-USB к ПК и нажатием кнопки «Коммуникация» на панели инструментов перейти к режиму установки связи с устройством.

Интерфейс EASY-SOFT и в этом режиме состоит из трех частей. Он очень похож на режим «Имитация» и содержит панель инструментов 1, поле свойств 2, а также схему соединений 3 (рис. 3.24).

При переходе к режиму «Коммуникация» EASY-SOFT пытается установить прямое соединение с устройством. Если попытка оказывается удачной, выполняется переход к режиму без сообщений об ошибках. Если же установить соединение не удается, выдается соответствующее сообщение об ошибке и также выполняется переход к режиму «Коммуникация».

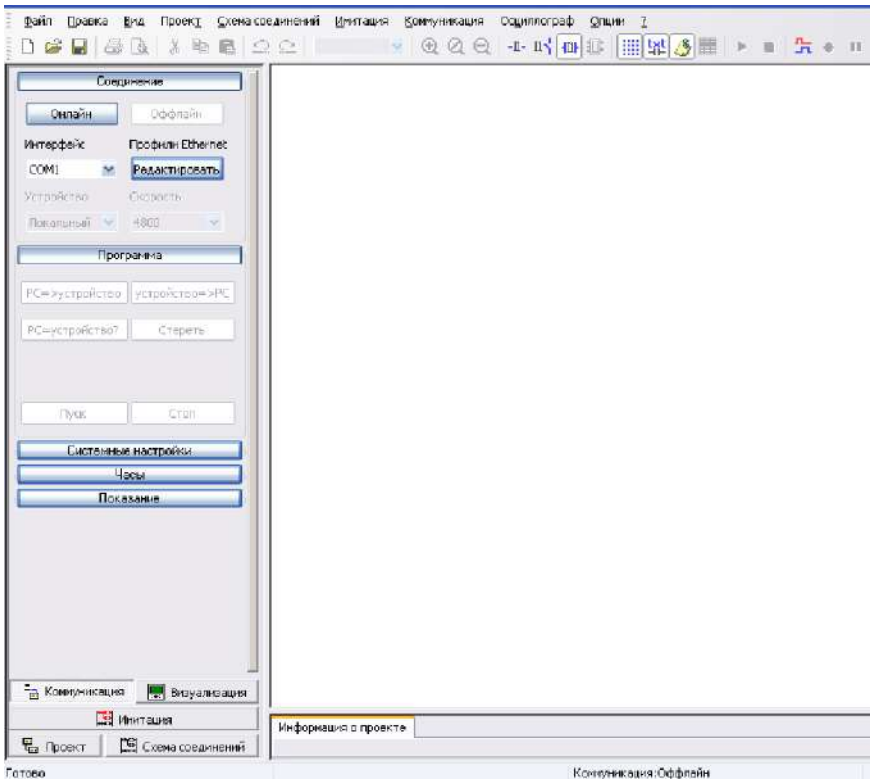


Рисунок 3.24 – Интерфейс Easy Soft 6Pro в режиме «Коммуникация».

Нажать на кнопку *Программа* на панели инструментов. Откроется диалоговое окно загрузки и передачи схемы соединений. Нажать на кнопку *PC-Устройство*. Перенос схемы соединений выполняется, когда устройство находится в режиме «Стоп». Если этого не происходит, на экране появляется диалоговое окно, для того чтобы остановить устройство и продолжить процесс загрузки. В окне *Панель свойств* находится индикатор процесса, отражающий текущее состояние передачи данных.

Запустить устройство нажатием кнопки *ПУСК* в открываемом из панели инструментов диалоговом окне *Программа.* Можно воспользоваться альтернативным вариантом через меню *Коммуникация, Устройство, Запустить.*

3.1.6 Разработка схемы управления линии предварительной очистки зерна на базе программируемого реле EASY.

Описание технологического процесса предварительной очистки зерна и требований к нему указаны в п. 2.4. Алгоритм выполнения работы:

1. В открывшемся окне «EASY-SOFT» на панели инструментов (вкладка «Проект») выбрать соответствующее реле EASY путем перетаскивания изображения устройства на рабочий стол.
2. Открыть вкладку «Схема соединений». На рабочем столе программы следует создать требуемую схему соединений с учетом технологических требований к схеме управления (см. выше). Элементы схемы соединений переносятся на рабочий стол перетаскиванием при зажатой левой кнопке мыши.
3. По окончании составления схемы соединений следует проверить работоспособность данной схемы. Это осуществляется во вкладке «Имитация» на панели инструментов. Здесь производится апробация набранной релейно-контактной схемы на наличие ошибок. При их наличии следует заново вернуться на вкладку «Схема соединений» для их дальнейшего исправления. При отсутствии ошибок в схеме переходим на вкладку «Коммуникация».
4. При переходе к режиму «Коммуникация» EASY-SOFT попытается установить прямое соединение с устройством. Если попытка оказывается удачной, выполняется переход к режиму без сообщений об ошибках. Если же возникли ошибки, следует проверить контактное соединения кабеля либо попробовать установить соединение через другой интерфейс (COM, USB, Ethernet) и снова попытаться установить соединение с программируемым реле нажатием на кнопку «Онлайн».

5. Включение управления схемой технологического процесса может производиться либо программно (при использовании интерфейса «EASY-SOFT»), либо при помощи элементов управления технологической схемой.

Алгоритм управления линии на базе программируемого реле представлен на рисунках 3.25, 3.26.

Рассмотрим работу представленного алгоритма управления. Как уже неоднократно было ранее оговорено, разработанный алгоритм состоит из двух явно выраженных типовых звеньев: пускосигнального звена и рабочего стопа. Отдельно они ранее рассматривались во 2 разделе.

Для простоты понимания разработанного алгоритма управления будем апеллировать представленными цепями. Так звено «пускосигнальное» занимает цепи 001-004. Рассмотрим, как оно работает. При нажатии на кнопку SB1 «пуск» сигнал появляется на первом дискретном входе программируемого реле I01. При этом внутри программы этот вход отображен в виде замыкающего контакта I01, который замыкает цепь и подает сигнал на катушку промежуточного реле KV1. В программе KV1 является маркером M01, имеющим как катушку управления, так и замыкающие/размыкающие контакты. Особенность маркера заключается в том, что он не имеет связи с выходами реле, служит лишь для взаимосвязи внутри программы. Поэтому его замыкающий контакт M01=KV1 шунтирует кнопку пуск.

Одновременно с появлением сигнала на промежуточном реле происходит включение катушку реле времени T01, один контакт которого, с выдержкой по времени, размыкает цепь со звонком (1 выход программируемого реле Q01), а другой – замыкает цепь 004 с Q02 (транспортер).

Если выход реле Q02 включен, работает транспортер, то только тогда может запуститься дробилка. Так через замкнутый контакт Q02 подается сигнал на замыкание третьего выходного реле Q03, к которому подключена дробилка.

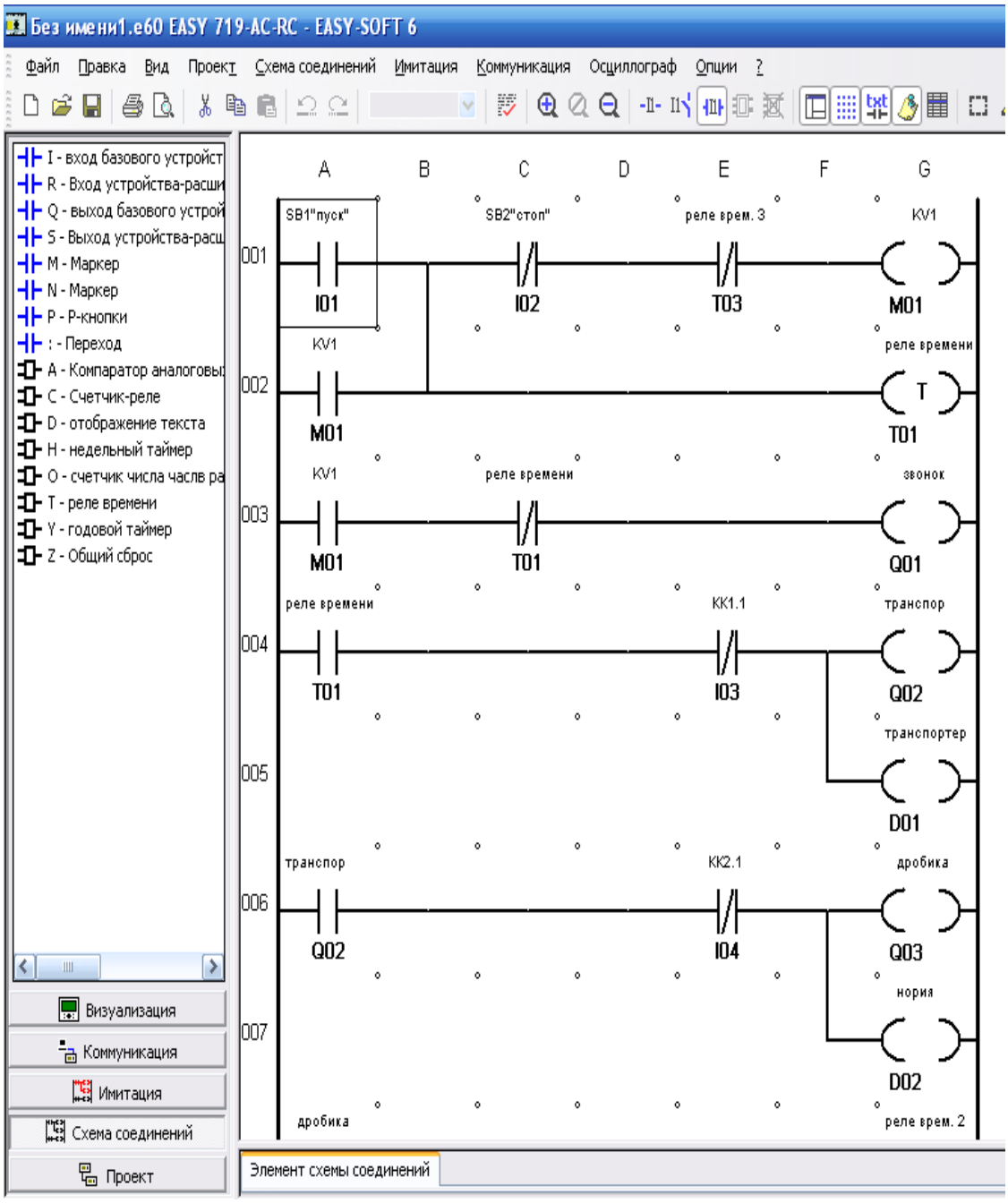


Рисунок 3.25 – Схема автоматизации линии на базе программируемого реле.

Для осуществления разгона 2 механизмов линии головной механизм необходимо включать с задержкой времени. Для этого используется реле времени T02, контакт которого произведет включение четвёртого выходного реле Q04 (нория).

Рассмотрим последнюю часть алгоритма управления, которая состоит из звена «рабочий стоп». Контакты кнопки «рабочий стоп» SB3 и контакт датчика уровня SL располагаются в программе параллельно друг другу, что

означает однотипность условия работы. Так при появлении сигнала с датчика уровня SL его контакт (I06) замыкается и происходит включение промежуточного реле KV2. Замыкающий контакт KV2 шунтирует датчик уровня, а размыкающий контакт KV2 разрывает цепь 006 с катушкой Q03 головного механизма, которым является нория. Это приводит к остановке подачи продукта на технологическую линию.

Одновременно, со срабатыванием датчика уровня, сигнал появляется на реле времени T03. Это приводит к отключению всех механизмов линии с временной задержкой, необходимой для очистки линии от продукта.

Для защиты электродвигателей механизмов линии в программе используются контакты теплового реле КК1.1…КК3.1, которые подключены к дискретным входам реле I03…I05. Как только происходит срабатывание теплового реле, на входе программируемого реле появляется сигнал, который внутри программы размыкает цепь с выходным устройством, что приводит к его отключению.

Для визуального контроля состояния работы механизмов в алгоритме управления предусмотрен блок отображения текста D01…D03. Он позволяет выводить информацию на внешний экран программируемого реле о состоянии входных и выходных сигналов.

На основании разработанного алгоритма управления технологического процесса получена принципиальная электрическая схема, внешний вид которой показан на рисунке 3.27. На нем совмещены цепи управления и силовая часть.

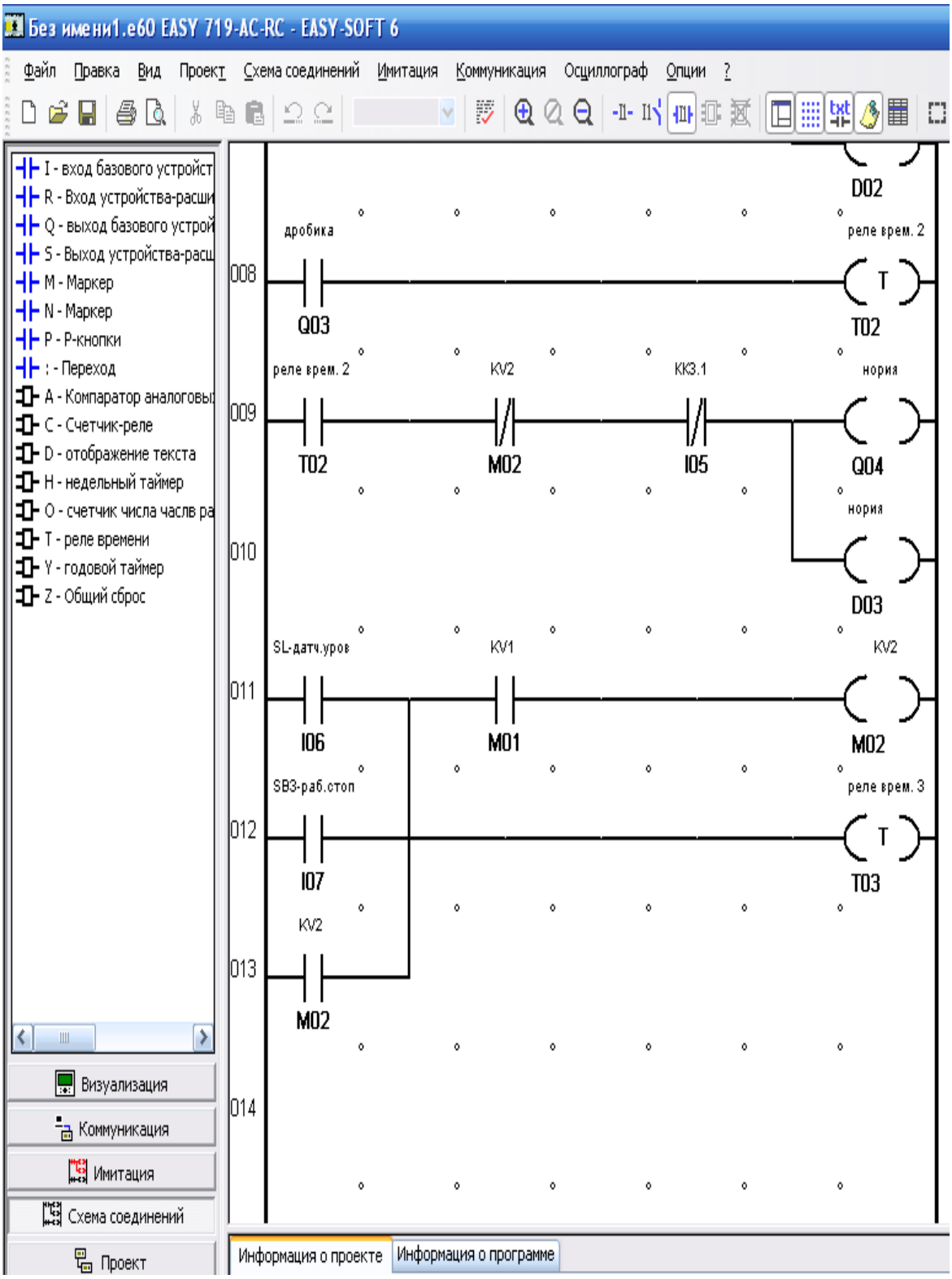


Рисунок 3.26– Схема автоматизации линии на базе программируемого реле (продолжение).

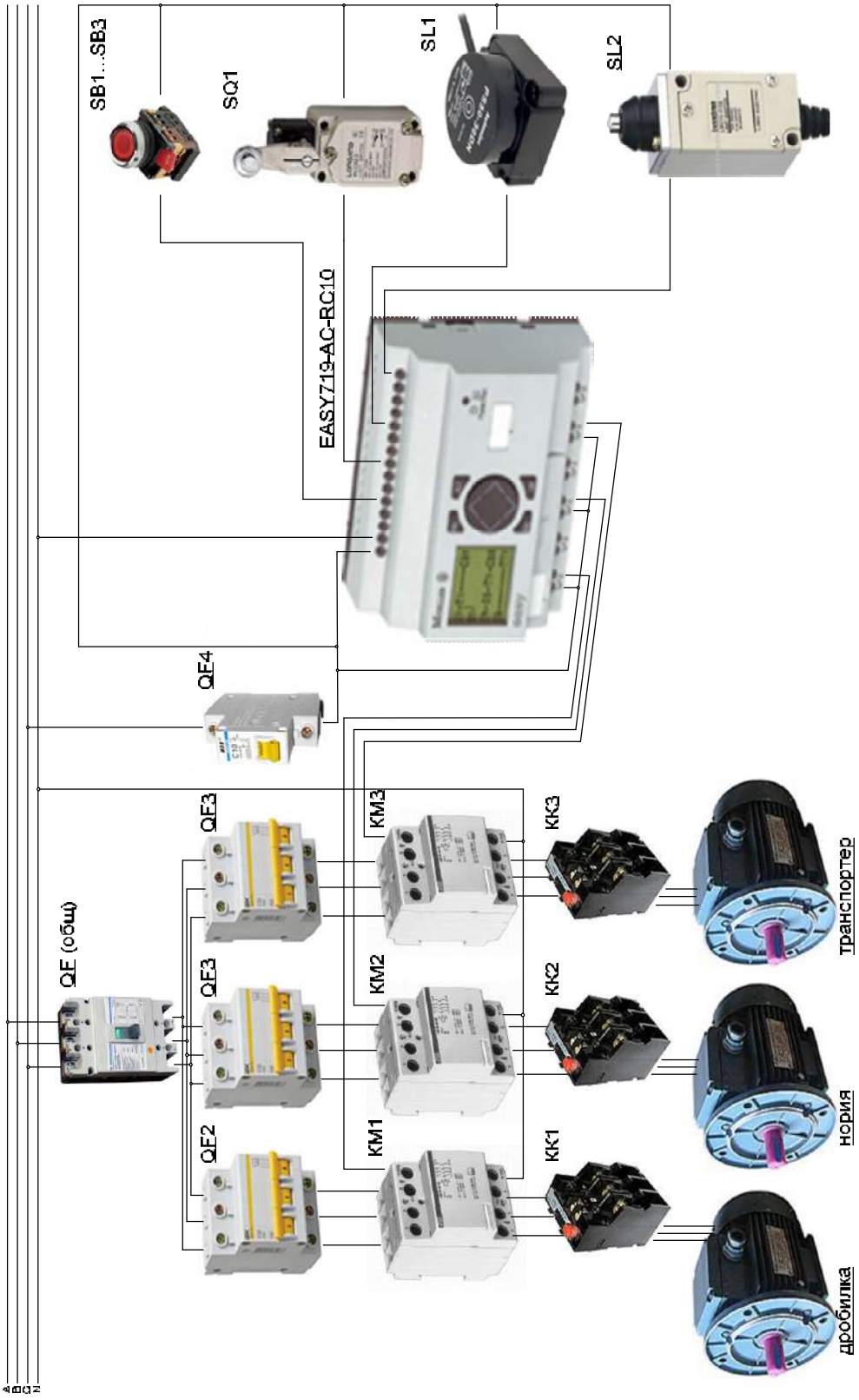


Рисунок 3.27 – Принципиальная схема управления линии предварительной очистки зерна.

3.2 Программируемые реле серии ПР114 фирмы «ОВЕН»

3.2.1 Общие сведения

ОВЕН ПР114 – это свободно программируемое устройство, которое не содержит в своей памяти заранее написанной программы. Алгоритм работы программируемого реле формируется непосредственно пользователем, что делает прибор универсальным и дает возможность широко использовать его в различных областях промышленности, сельском хозяйстве, ЖКХ и на транспорте.

Специалисты «ОВЕН» рекомендуют использовать приборы данной линейки при замене устаревших релейных систем защиты и контроля. За счет внутренней логики прибора можно значительно сократить количество коммутируемых электромагнитных устройств, что снизит затраты на проектирование и эксплуатацию систем, а также повысит их надежность.

Программирование ПР не требует специальных навыков, поскольку осуществляется с помощью простой и интуитивно понятной среды программирования [6].

Область применения:

1) управление наружным и внутренним освещением, освещением витрин;
2) управление технологическим оборудованием (насосами, вентиляторами, компрессорами, прессами);
3) конвейерные системы;
4) управление подъемниками и т. д.

Логика работы прибора ПР114 определяется пользователем в процессе программирования с помощью среды «OWEN Logic».

Общий вид прибора с указанными номерами клемм, разъема программирования и светодиодов, а также способы подключения дискретных датчиков с выходом типа «сухой контакт» представлены на рисунках 3.28 и 3.29.

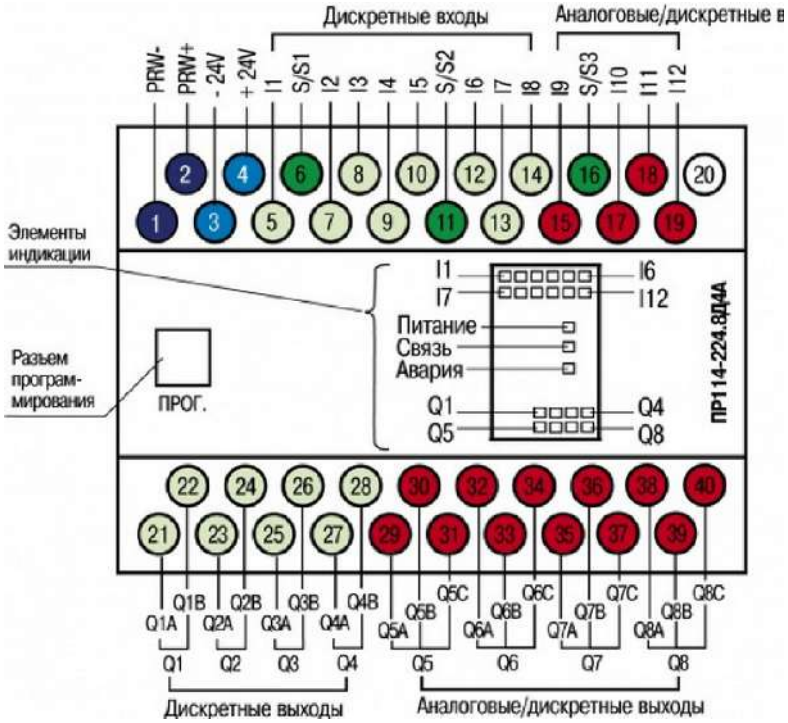


Рисунок 3.28 – Схема расположение контактов и элементов индикации в приборе ПР114

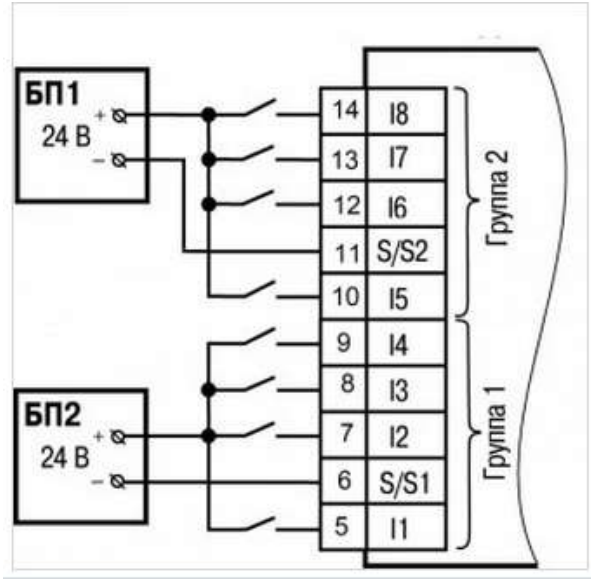


Рисунок 3.29 – Схема подключение к ПР114 дискретных датчиков с выходом Типа «сухой контакт»

3.2.2 Основные характеристики

Программа OWEN Logic позволяет сформировать схему автоматизации, соответствующую конкретному комплексу процедур, управление кото-

рыми должен выполнять ПР при подаче сигналов на его входы [19].

Управление внешними устройствами выполняют коммутационные элементы на выходах ПР по загруженной в его энергонезависимую память коммутационной программе (проекту).

Одновременно среда программирования может работать только с одним проектом. При создании проекта не требуется подключения ПР.

Последовательность операций при работе следующая:

1) запуск среды программирования на ПК (запуск программы OWEN Logic);
2) создание нового проекта (для конкретной модели ПР) или запуск существующего проекта для редактирования;
3) сохранение проекта в виде файла (с любым именем);
4) загрузка проекта в ПР.

В программе используется визуальный язык FBD, в котором заложены принципы построения электрических принципиальных схем, работающих с логическими сигналами.

В создаваемой при помощи программы схеме можно использовать функции (логические НЕ, И, ИЛИ, исключающее ИЛИ), а также специальные функциональные блоки (см. разделы 3.2.5, 3.2.6).

3.2.3 Особенности коммутационной программы.

Коммутационная программа для ПР составляется с учетом количества имеющихся у него входов, выходов и наличия часов реального времени. Общая структура для таких программ показана на рисунке 3.30 [8].

Работу ПР можно представить в виде последовательно выполняемых шагов:

Шаг 1 – логическое состояние входов автоматически записывается в ячейки памяти входов (количество ячеек равно числу входов, например, I1,…, I8).

Шаг 2 – коммутационная программа считывает значения из ячеек памяти входов и выполняет над ними логические операции, в соответствии со

схемой автоматизации.

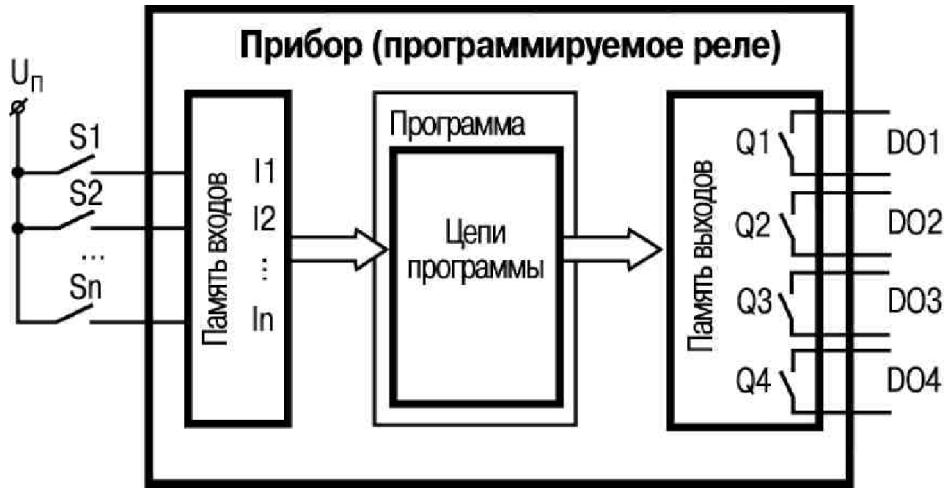


Рисунок 3.30 - Функциональная схема работы ПР (S1–Sn – переключатели или кнопки).

Шаг 3 – после обработки всей коммутационной программы производится запись результатов на физические выходы ПР (для включения контактов выходных реле Q1, …, Q4).

Шаг N– переход на Шаг 1 (после выполнения всех предыдущих шагов обработки коммутационной программы цикл работы ПР повторяется с первого шага). Скорость повторения определяется внутренней тактовой частотой работы ПР.

Время выполнения всех шагов называется рабочим циклом ПР (зависит от количества выполняемых операций в цепях схемы).

3.2.4 Работа с программой

Графический интерфейс

После запуска программы OWEN Logic появляется главное окно программы, приведенное на рисунке 3.31.

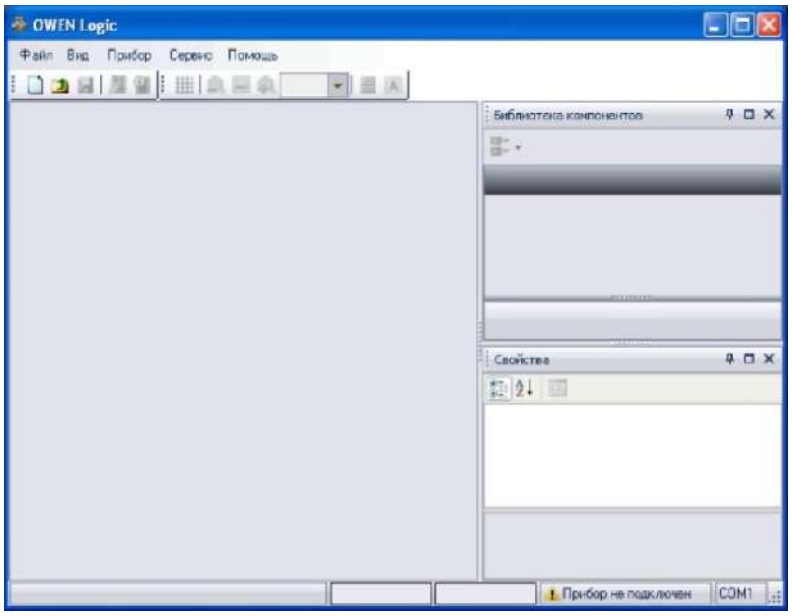


Рисунок 3.31– Главное окно программы (вид до открытия проекта).

Главное окно программы включает:

– заголовок окна (верхняя строка) – после создания проекта в заголовке автоматически выводится информация об имени файла проекта и его месте размещения на ПК;

– главное меню программы: Файл, Вид, Прибор, Сервис, Помощь;

– панели инструментов для быстрого вызова часто используемых действий

Создание нового проекта и его сохранение.

После запуска программы OWEN Logic следует нажать кнопку «Создать» на панели инструментов – при этом появится окно выбора типа ПР, для которого будет создаваться проект (рисунок 3.32) [4].

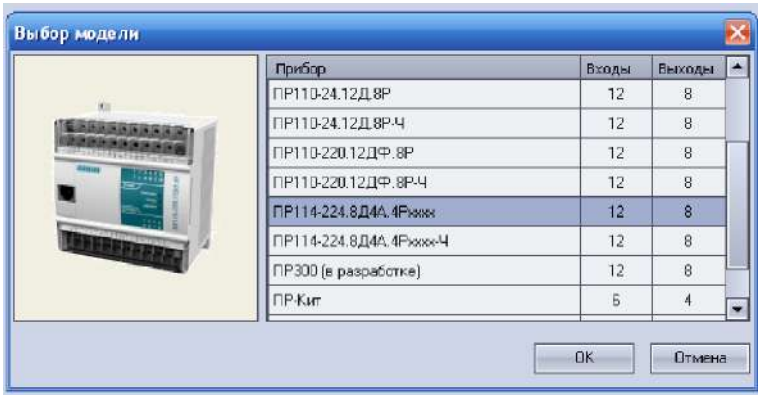


Рисунок 3.32– Окно «Выбор модели» ПР для проекта.

После выделения курсором нужной модели ПР, выбор подтверждается нажатием кнопки ОК, в рабочей области программы появится поле (холст), на краях которого слева размещены входы I1–I8, а справа – выходы Q1–Q4 ПР (рисунок 3.33).

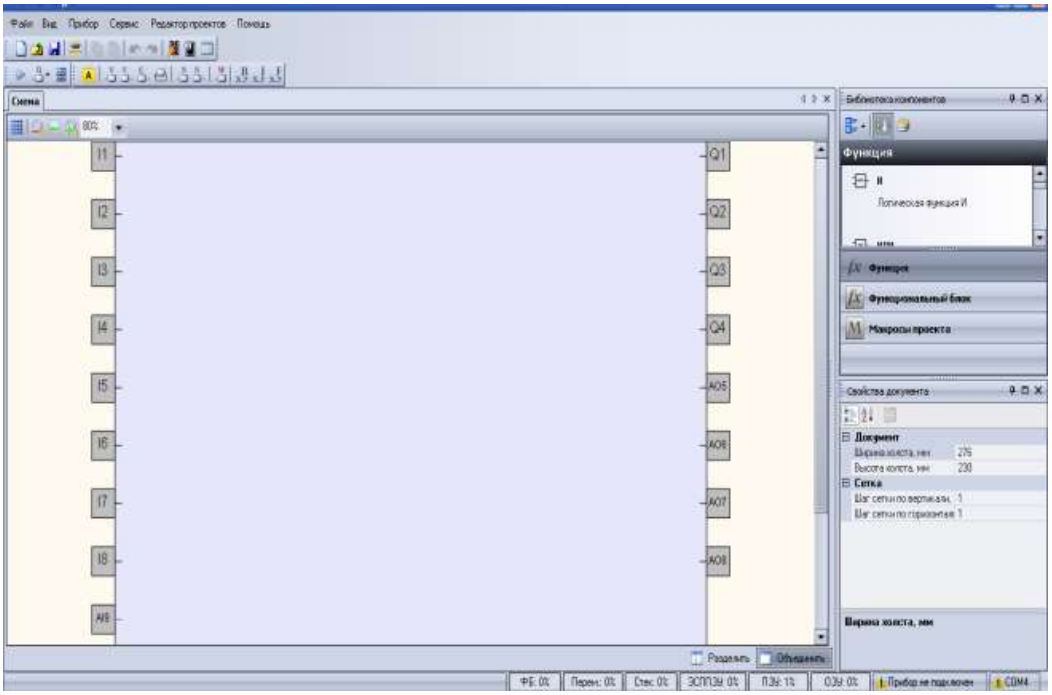


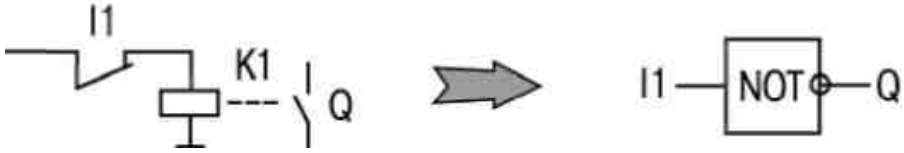
Рисунок 3.33–Вид окна программы после выбора модели ПР.

Проект создается путем перетаскивания выделенных курсором мыши блоков из вкладки «Библиотека компонентов» на рабочее поле (холст) и виртуального соединения цепей блоков между собой и с входами и выходами ПР. Для установки соединений цепей блоков пользователь курсором мыши указывает начальную (вход/выход) и конечную (вход/выход) точки привязки.

Примечание. Связь не может устанавливаться только между входами (или только между выходами) графических компонентов – допустимым является соединение выхода и входов.

3.2.5 Функции логических элементов программы

Функция «НЕ» (NOT)

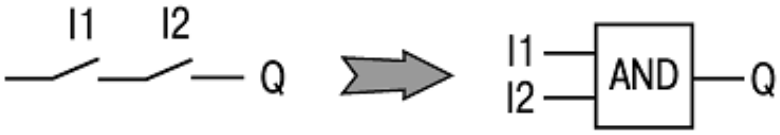


I1	Q
0	1
1	0

Рисунок 3.34– Электрическая принципиальная схема функции «НЕ»

Элемент используется для инвертирования сигнала. На выходе элемента логическая «1» (выход включен), если на входе логический «0» (контакты разомкнуты) и наоборот – инвертируется сигнал.

Функция «И» (AND)

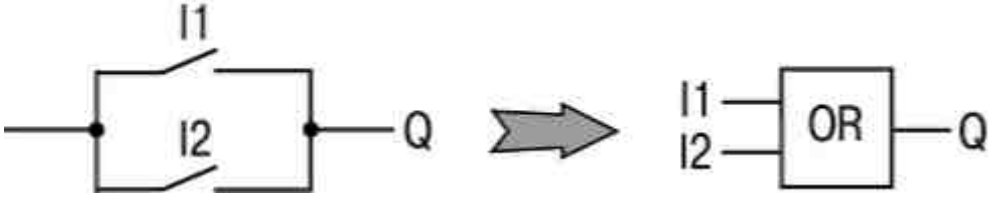


I1	I2	Q
0	0	0
0	1	0
1	0	0
1	1	1

Рисунок 3.35– Электрическая принципиальная схема функции «И»

Элемент используется для выполнения логических операций. На выходе элемента логическая «1» (выход включен), если на всех входах логическая «1» (все входы включены – контакты замкнуты). Работе соответствует приведенная таблица состояний.

Функция «ИЛИ» (OR)



I1	I2	Q
0	0	0
0	1	1
1	0	1
1	1	1

Рисунок 3.36 – Электрическая принципиальная схема функции «ИЛИ»

Элемент используется для выполнения логических операций. На выходе элемента логическая «1» (выход включен), только если на любом из входов логическая «1».При использовании функций **И** и **ИЛИ** следует учитывать, что не подключенные входы логических элементов к входам ПР или другим элементам в программе будут иметь следующие состояния:

– для элемента **И (AND) –** логическая «1»;

– для элемента **ИЛИ (OR) –** логический «0». В этом случае логические элементы выполняют функцию повторителя сигнала.

Для увеличения числа входов у логических элементов используется их каскадное включение, например, как это показано на рисунке 3.37.

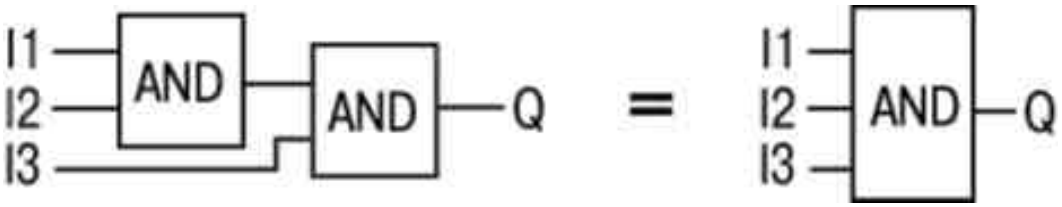


Рисунок 3.37– Пример каскадного включения логических элементов «И».

3.2.6 Функциональные блоки программы

RS-триггер с приоритетом выключения (RS).

Блок используется для переключения с фиксацией состояния при поступлении коротких импульсов на соответствующий вход. Работу поясняет приведенная на рисунке диаграмма.

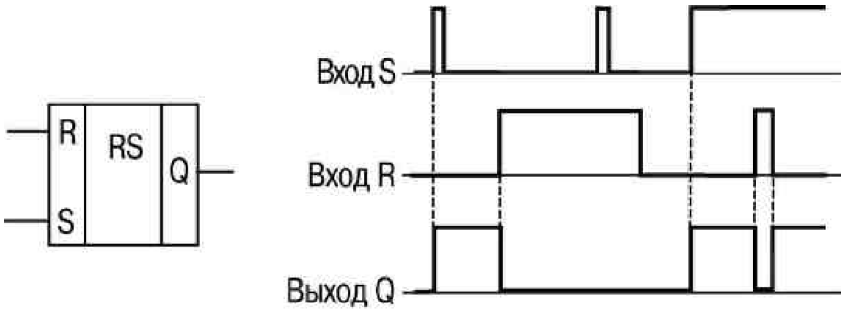


Рисунок 3.38 – Диаграмма работы RS-триггера с приоритетом выключения.

На выходе блока появится логическая «1» по фронту сигнала на входе S.При одновременном поступлении сигналов на входы приоритетным является сигнал входа R.

SR-триггер с приоритетом включения (SR).

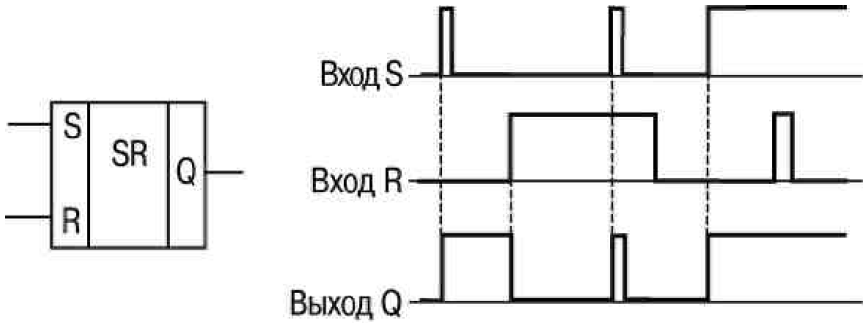


Рисунок 3.39 – Диаграмма работы SR-триггера с приоритетом включения.

Блок используется для переключения с фиксацией состояния при поступлении коротких импульсов на соответствующий вход. Работу поясняет приведенная на рисунке диаграмма. На выходе блока появится логическая «1» по фронту сигнала на входе S.При одновременном поступлении сигналов на входы приоритетным является сигнал входа S.

Импульс включения заданной длительности (TP).

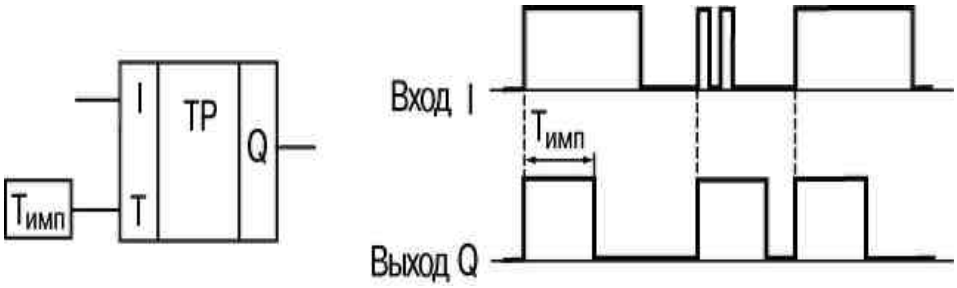


Рисунок 3.40 – Диаграмма работы блока «Импульс включения заданной длительности».

Блок используется для формирования импульса включения выхода на заданный интервал времени. Работу поясняет приведенная на рисунке диаграмма. На выходе Q блока появится логическая «1» по фронту входного сигнала (I). После запуска выход Q не реагирует на изменение значения входного сигнала на интервале $T_{имп}$, а по истечении этого интервала сбрасывается в «0».Допустимый диапазон значений $T_{имп}$=T – от 0 до 4147200000

мс, или 48 дней.

Генератор прямоугольных импульсов (BLINK).

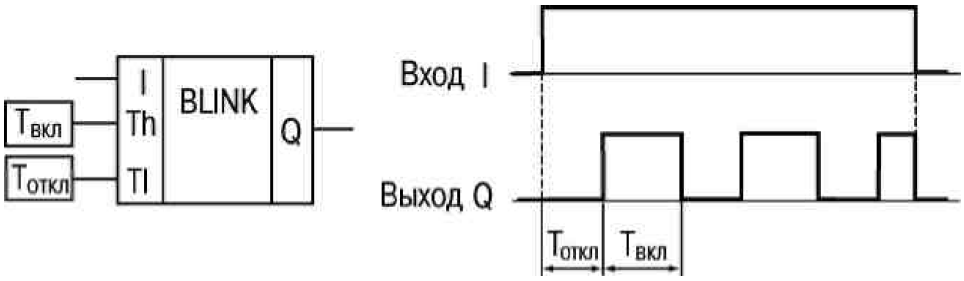


Рисунок 3.41 – Диаграмма работы блока «Генератор прямоугольных импульсов».

Блок используется для формирования прямоугольных импульсов пульсации. На выходе Q генератора формируются импульсы с заданными параметрами длительности включенного ($T_{вкл}$ – логическая «1») и отключенного ($T_{откл}$ – логический «0») состояния на время действия управляющего сигнала на входе I (логической «1»). Работу поясняет приведенная на рисунке диаграмма. Допустимый диапазон значений $T_{вкл}$ и $T_{откл}$ – от 0 до 4233600000 мс, или 49 дней.

Таймер с задержкой включения (TON).

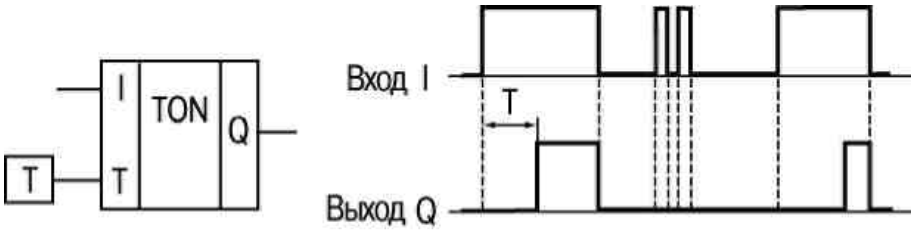


Рисунок 3.42 – Диаграмма работы блока «Таймер с задержкой включения».

Блок используется для операции задержки передачи сигнала. Работу поясняет приведенная на рисунке диаграмма. На выходе Q блока появится логическая «1» с задержкой относительно фронта входного сигнала (I). Выход включается логической «1» на входе продолжительностью не менее длительности T, а выключается по спаду входного сигнала. Допустимый диапа-

зон значений T – от 0 до 4147200000 мс, или 48 дней.

Таймер с задержкой отключения (TOF).

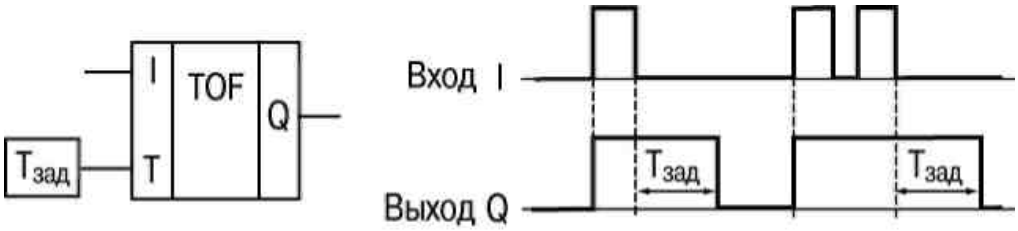


Рисунок 3.43– Диаграмма работы блока «Таймер с задержкой отключения».

Блок используется для задержки отключения выхода. Работу поясняет приведенная на рисунке диаграмма. На выходе блока появится логическая «1» по фронту сигнала на входе I, а начало отсчета времени задержки отключения ($T_{зад}$) происходит по каждому спаду входного сигнала. После отключения входного сигнала на выходе появится логический «0» с задержкой $T_{зад}$. Допустимый диапазон значений $T_{зад}$ – от 0 до 4147200000 мс, или 48 дней.

Инкрементный счетчик с автосбросом (CT).

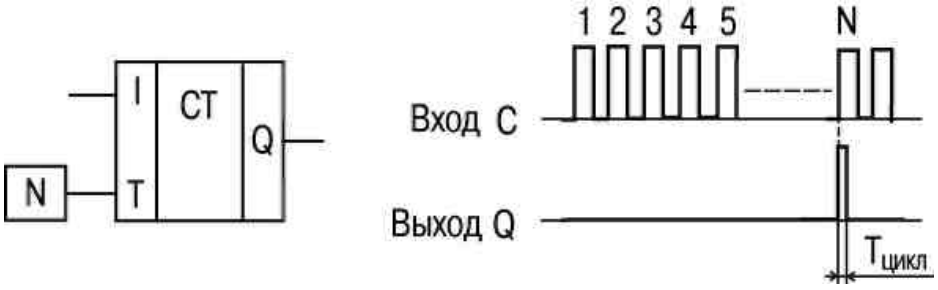


Рисунок 3.44 – Диаграмма работы блока «Инкрементный счетчик с автосбросом».

Блок используется для подсчета заданного числа импульсов. Работу поясняет приведенная на рисунке диаграмма. На выходе Q счетчика появится импульс логической «1» с длительностью рабочего цикла прибора ($T_{цикл}$), если число приходящих на вход импульсов достигнет установленного значения N.Допустимый диапазон значений числа импульсов N – от 0 до 65535.

Интервальный таймер (CLOCK).

Блок используется для формирования импульса включения выхода по часам реального времени. Работу поясняет приведенная на рисунке диаграмма. Время включения ($T_{вкл}$) и отключения ($T_{откл}$) выходов устанавливают в качестве параметров блока. Допустимый диапазон значений $T_{вкл}$ и $T_{откл}$ – от от 0,00 с до 24 ч.

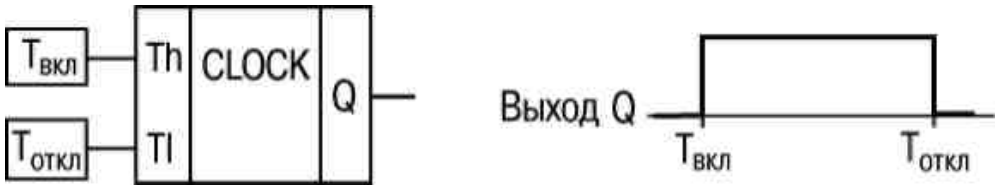


Рисунок 3.45– Диаграмма работы блока «Интервальный таймер».

В случаи если значение время отключения задано раньше времени включения (рисунок 3.46), диаграмма переключений будет иметь вид, приведенный на рисунке 3.47 (выход будет включен до момента времени отключения).

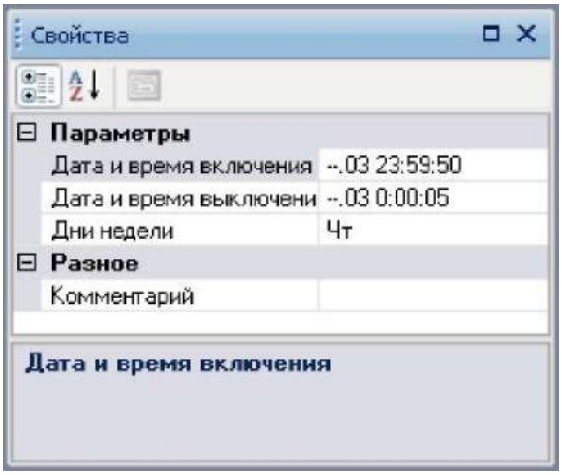


Рисунок 3.46– Вкладка «Свойства» для настройки временных параметров работы блока «Интервальный таймер».

Рисунок 3.47– Диаграмма включения выхода блока Интервальный таймер».

Интервальный таймер с недельным циклом (CLOCKWEEK)

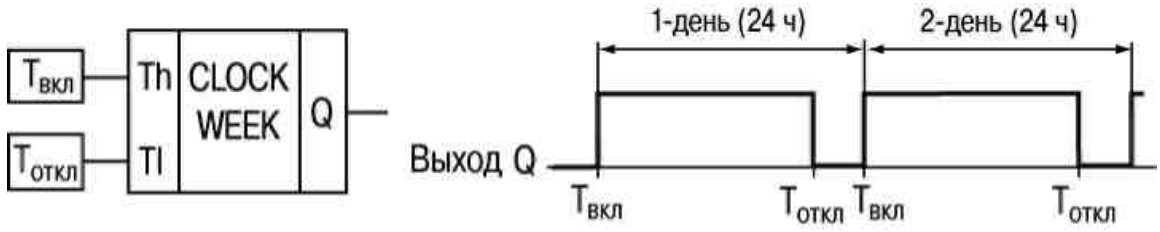


Рисунок 3.48 – Диаграмма работы блока «Интервальный таймер с недельным циклом».

Блок используется для формирования импульса включения выхода по часам реального времени, с учетом дней недели. Работу поясняет приведенная на рисунке диаграмма. Внутренняя структура блока имеет вид:

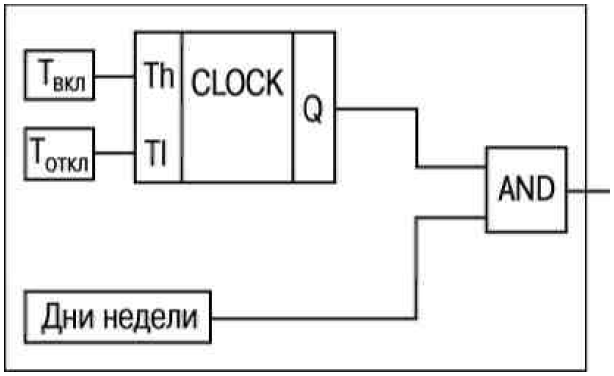


Рисунок 3.49 – Внутренняя структура блока «Интервальный таймер с недельным циклом».

Время включения ($T_{вкл}$), отключения ($T_{откл}$) выхода и дни недели работы устанавливают в качестве параметров блока. Допустимый диапазон значений $T_{вкл}$ и $T_{откл}$ – от 0,00 с до 24 ч.

3.2.7 Загрузка проекта в программируемый прибор.

После создания проекта его записывают в энергонезависимую память ПР. Для записи необходимо:

1) соединить ПР с COM или USB-портом ПК при помощи коммуникационного кабеля (см. приложение 2, рисунок 2.1);

2) подать питание на ПР согласно его руководству по эксплуатации;

3) настроить параметры соединения, если в этом есть необходимость;

4) записать проект в ПР при помощи соответствующей кнопки на панели инструментов или выбрать команду «Файл => Записать программу в прибор». **Примечание.** Сразу после записи ПР переходит в рабочий режим и коммутационная программа запускается автоматически.

Внимание! Если в подключенном ПР уже есть ранее записанная коммутационная программа, то при записи в него нового проекта она заменяется.

Кроме этого:

1) пользователь может добавить невидимые комментарии к компонентам схемы – комментарии будут отображаться в виде всплывающей подсказки только при установке курсора на элемент;

2) пользователь может добавить на схему проекта сектор с видимыми комментариями, а также выделить фрагмент поля у схемы цветом. Чтобы поместить на страницу какой-либо текст, следует щелкнуть мышкой (т. е. кратковременно нажать правую кнопку мыши) по кнопке  на панели инструментов, затем установить курсор на место, где должна располагаться надпись и при нажатой правой кнопке провести по диагонали в секторе размещения текста. В результате в том месте появится прямоугольная рамка текстовой области с рабочим текстом (рисунок 3.50).

Рисунок 3.50– Сектор холста для внесения комментариев.

3.2.8 Последовательность работы над проектом

Составление коммутационной программы рекомендуется начинать с планирования. План должен описывать все возможные состояния ПР при функционировании (в виде диаграммы режимов, таблицы состояний, электрической или функциональной схемы и/или др.).

После того, как продуманы все задачи, которые должны выполняться, необходимо составить программу на основе функций (логических элементов) и функциональных блоков(см. разделы3.2.5, 3.2.6).

Работа над проектом включает:

1) открытие нового проекта – весь проект будет храниться в одном файле, которому следует присвоить идентификационное имя.

2) формирование структуры текущего проекта рекомендуется выполнять в следующем порядке:

а) из «Библиотеки компонентов» на холст добавляются необходимые блоки-путем перетаскивания их мышью при нажатой на ней левой кнопке (из соответствующей вкладки «Функции» или «Функциональные блоки»);

б) последовательно выделяя курсором блоки схемы, на закладке «Свойства»установить их параметры, например, как это показано на рисунке 3.51;

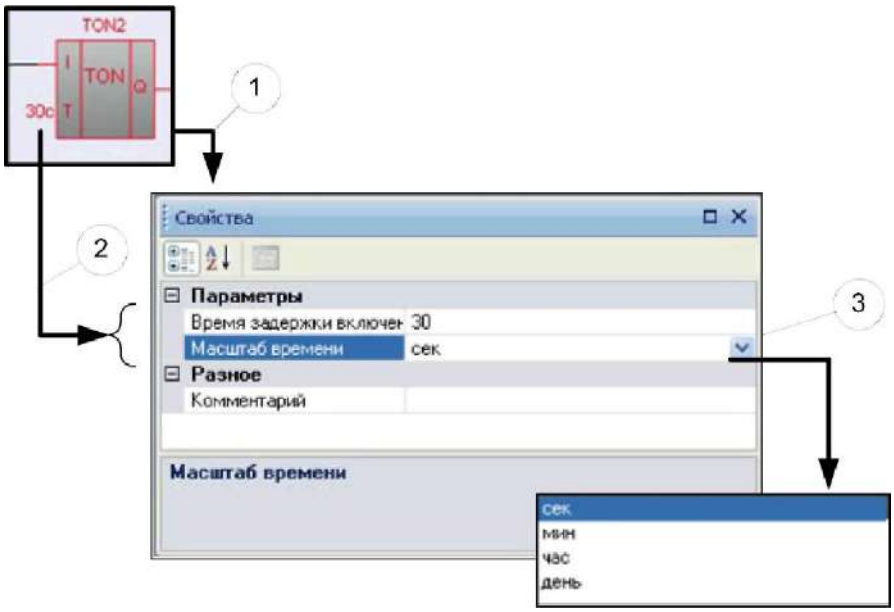


Рисунок 3.51– Порядок действий при настройке параметров функционального блока.

в) соединяются блоки программы между собой, а также с нужными входами и выходами ПР. При этом допускается передвигать квадраты входов и выходов в вертикальной плоскости для расположения соединительных линий по кратчайшей длине;

3) загружается проект в ПР и проверяется его работа на макете. При проверке

правильности работы коммутационной программы в ПР последовательно включают и выключают каждый вход, контролируя состояние выходов на соответствие нужным условиям;

4) по результатам макетирования производится редактирование проекта с целью устранения ошибок (добавление новых элементов и программных цепей, или их удаления, редактирование параметров работы функциональных блоков и т. д.);

5) после устранения всех ошибок подготовка проекта завершается сохранением его в файле;

6) загрузка проекта в ПР114 и проверка его работы на макете.

Процесс создания коммутационной управляющей программы иллюстрируется на конкретном примере, приведенном в разделе 3.2.10.

3.2.9 Типовые звенья схем автоматизации, реализованных в программе OWEN Logic

На основании ранее рассмотренных звеньев (пункте 2.2) были разработаны типовые звенья, реализованные в программе OWEN Logic с помощью функциональных блоков и логических функций, представлены на рисунках 3.52, 3.53 и 3.54.

Рассмотрим типовое звено *нереверсивной схемы управления*. Состоит из двух функциональных блоков детектора переднего фронта «RTRIG», одного блока «SR» триггера с приоритетом включения, логической функции ИЛИ «OR»и логической функции НЕ «NOT». Предназначено для ручного управления включением и отключением электроустановок (двигателей, нагревателей систем освещения и прочее).

Принцип работы: при нажатии на кнопку «пуск» сигнал появляется на входе I2, к которому подключен функциональный блок детектора переднего фронта «RTRIG1». Функциональный блок пропускает через себя сигнал им-

пульсом, после чего на его выходе будет всегда «0». Пройденного кратковременного импульса хватает для того, чтобы «SR1» триггера сработал и на выходе его появилась и запомнилась «1». Это приводит к появлению сигнала на выходе реле «Q2», а следовательно, и к включению магнитного пускателя механизма.

При нажатии на кнопку «стоп» сигнал появляется на входе I3, к которому подключен функциональный блок детектора переднего фронта «RTRIG2». Он пропускает через себя сигнал импульсом, который, проходя через логическую функцию ИЛИ «OR», подается на вход сброса «SR1» триггера, что приводит к его отключению и, как следствие, остановке работы механизма.

Для защиты электродвигателя от аварийных режимов работы в схеме управления предусмотрена установка теплового реле. В случае возникновения перегрузки сигнал, проходящий через размыкающий контакт теплового реле, пропадает на входе реле I5, что приводит к появлению сигнала на входе сброса «SR1» триггера. На рисунке 3.52 представлена нереверсивная схема управления.

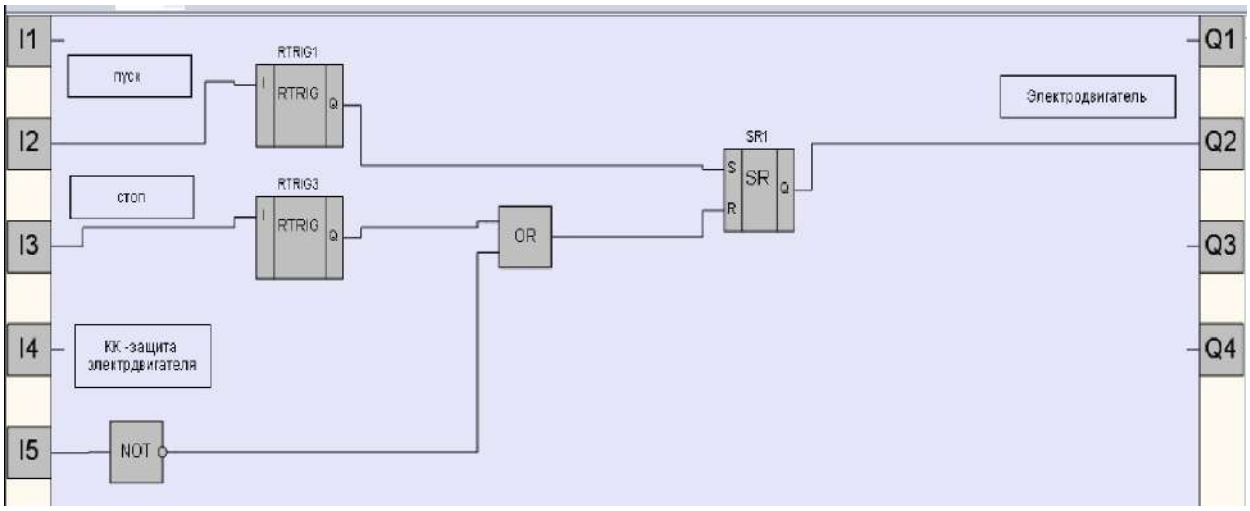


Рисунок 3.52– Нереверсивная схема управления электродвигателя.

Рассмотрим типовое звено *реверсивной схемы управления*. Состоит из трех функциональных блоков детектора переднего фронта «RTRIG», двух блоков «SR» триггера с приоритетом включения, трех логических функций ИЛИ «OR»и логической функции НЕ «NOT». Предназначено для ручного управления реверсом электродвигателя.

Принцип работы: при нажатии на кнопку «пуск влево» сигнал появляется на входе I2, к которому подключен функциональный блок детектора переднего фронта «RTRIG1». Функциональный блок пропускает через себя сигнал, который подается на вход установки «SR1» триггера. Это приводит к появлению сигнала на выходе реле «Q2», а следовательно, и к включению первого магнитного пускателя, который приводит во вращение электродвигатель против часовой стрелки.

При нажатии на кнопку «пуск вправо» сигнал появляется на входе I4, к которому подключен функциональный блок детектора переднего фронта «RTRIG2». Функциональный блок пропускает через себя сигнал, который подается на вход установки «SR2» триггера. Это приводит к появлению сигнала на выходе реле «Q4», а следовательно, и к включению второго магнитного пускателя, который приводит во вращение электродвигатель по часовой стрелке.

Во избежание одновременной работы двух магнитных пускателей в схеме предусмотрены деблокировочные режимы, которые реализуются посредством двух логических функций ИЛИ «OR». Если на выходе «SR1» триггера будет сигнал, то на входе сброса «SR2» триггера также будет сигнал, что исключает одновременного наличия сигнала на выходах «SR1» и«SR2» триггеров.

При нажатии на кнопку «стоп» сигнал появляется на входе I3, к которому подключен функциональный блок детектора переднего фронта «RTRIG3». Сигнал подается через него на вход обоих «SR1» и «SR2» триггеров, что приводит к их отключению и, как следствие, остановке работы механизма. На рисунке 3.53 представлена нереверсивная схема управле-

ния.

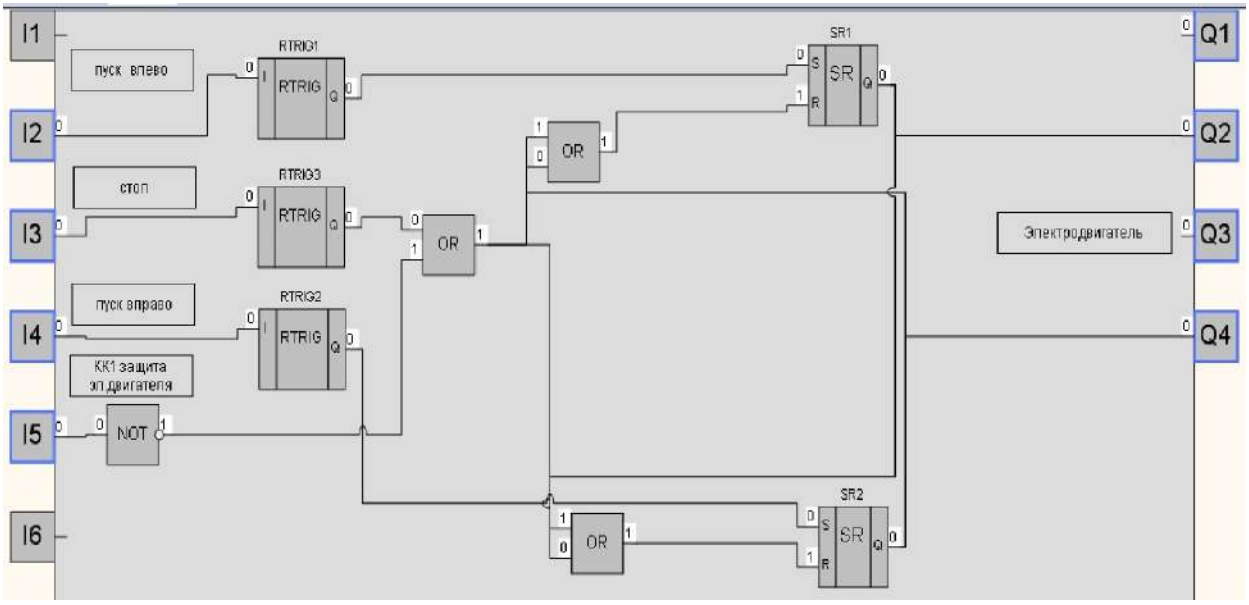


Рисунок 3.53 – Нереверсивная схема управления.

Типовое звено *пускосигнальное*. Состоит из двух функциональных блока детектора переднего фронта «RTRIG», двух блока «SR» триггера с приоритетом включения, таймера с задержкой включения «TON», блока импульса включения заданной длительности «TP», логических функций ИЛИ «OR»и логической функции НЕ «NOT». Предназначено для ручного управления электродвигателем с задержкой его включения.

Принцип работы: при нажатии на кнопку «пуск» сигнал появляется на входе I2, к которому подключен функциональный блок детектора переднего фронта «RTRIG1». Функциональный блок пропускает через себя сигнал, который подается на вход установки «SR1» триггера. Это приводит к появлению сигнала на выходе триггера, который одновременно подается на два временных блока. Блок импульса включения заданной длительности «TP» сразу формирует сигнал на выходе определенное время, в нашем случае равное 5 сек., что приводит к появлению сигнала на выходе реле «Q2» – звенит звонок. Одновременно с этим на входе блока таймера с задержкой включения «TON» присутствует сигнал, который, с задержкой времени, равной 5 сек., подаст сигнал на управление «SR2» триггера механизма.

При нажатии на кнопку «стоп» сигнал появляется на входе I3, к которому подключен функциональный блок детектора переднего фронта «RTRIG2». Сигнал подается через него на вход обоих «SR1» и «SR2» триггеров, что приводит к их отключению и, как следствие, остановке работы механизма. На рисунке 3.54 представлена схема пускосигнального звена.

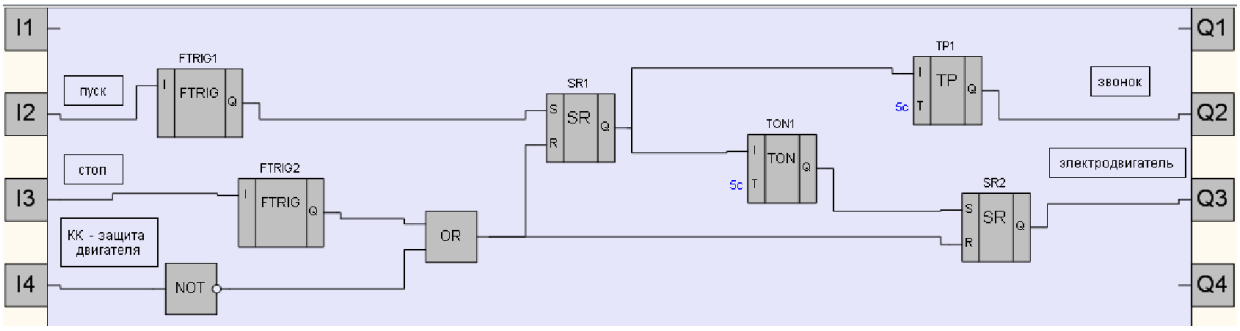


Рисунок 3.54 – Пускосигнальное звено.

Типовое звено «*рабочий стоп*». Состоит из функционального блока детектора переднего фронта «RTRIG1», двух блоков «SR» триггера с приоритетом включения, таймера с задержкой включения «TON1». Предназначено для ручного управления электродвигателями с задержкой их отключения.

Принцип работы: при нажатии на кнопку «пуск» сигнал появляется на входе I1, к которому подключен функциональный блок детектора переднего фронта «RTRIG1». Функциональный блок пропускает через себя сигнал, который подается на вход установки «SR1» триггера. Это приводит к появлению сигнала на выходе триггера, который одновременно подается на временной блок таймера с задержкой включения «TON1» и «SR2» триггера.
При возникновении сигнала на входе сброса «SR2» триггера происходит его отключение. При этом отключается головной механизм линии.

Одновременно с этим на входе блока таймера с задержкой включения «TON» присутствует сигнал, который, с задержкой времени, равной 5 сек., подаст сигнал на отключение всех «SR» триггеров механизмов линии. На рисунке 3.55 представлена схема звена «рабочего стопа».

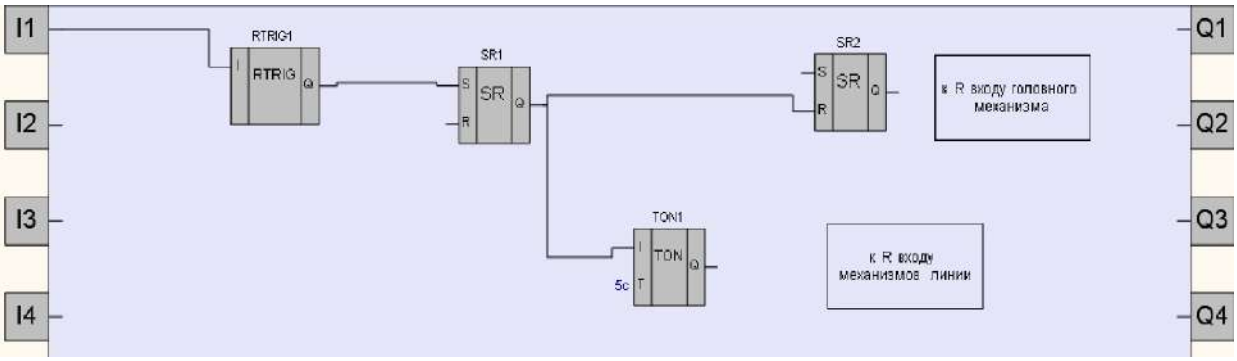


Рисунок 3.55 – Звено «Рабочий стоп».

3.2.10 Разработка схемы управления линии предварительной очистки зерна на базе ПР 114

На основании разработанных бесконтактных схем управления необходимо реализовать алгоритм управления линии предварительной очистки зерна (см. п. 2.4). Для проверки работоспособности алгоритма управления используется лабораторное оборудование, внешний вид которого изображен на рисунке 3.56. На нем обозначены:

1. Кнопка включения/выключения питания стенда.
2. Панель оператора СП270.
3. Блок питания 24В.
4. Программируемое реле ПР114.
5. Кнопки управления.
6. Концевые выключатели.
7. Аналоговые входы.
8. Аналоговые выходы.
9. Твердотельные реле с токовым управлением.
10. Лампы индикации работы механизмов.
11. Предохранитель.

Панель оператора представляет собой программируемый терминал, выполняющий функции интерфейса оператора в системе, включающей опре-

деленную совокупность технологических процессов (например, в системах вентиляции, тепло- и водоснабжения зданий), ПЛК (программируемый логический контроллер), выполняющий контрольные и управляющие функции относительно этих процессов, и оператора, контролирующего прохождение этих процессов и, при необходимости, корректирующего деятельность ПЛК.

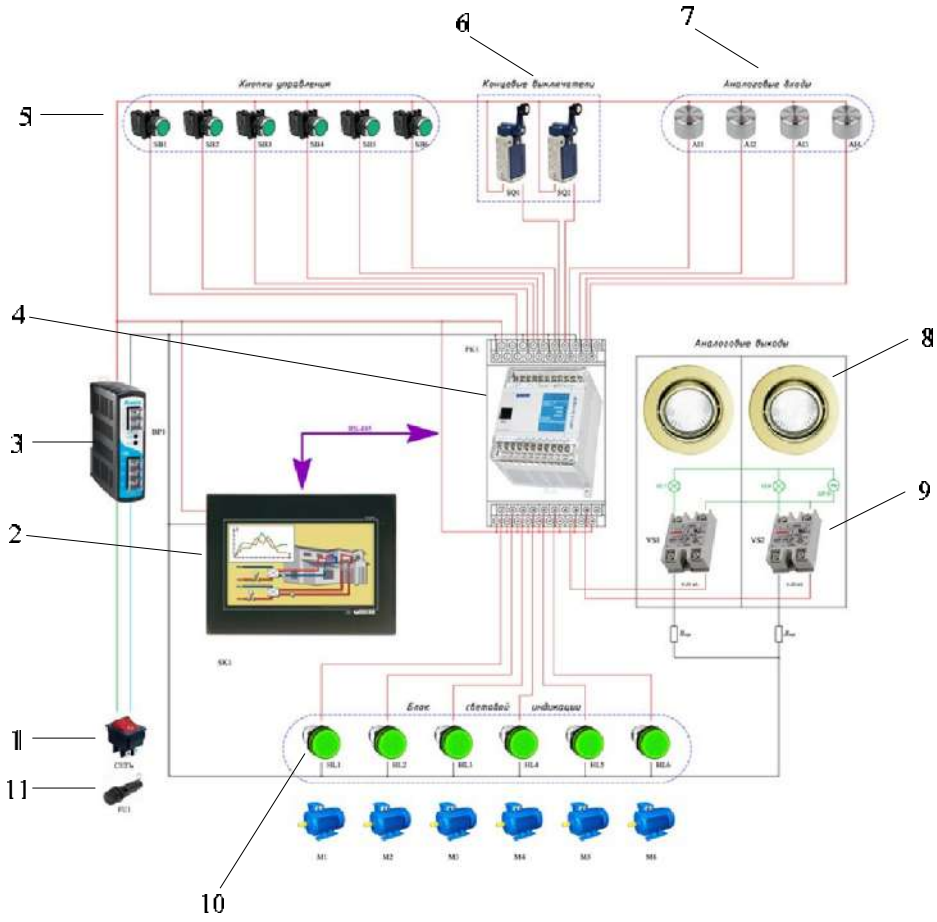


Рисунок 3.56 – Внешний вид лабораторной установки.

Функции интерфейса выполняются сенсорным экраном панели оператора, на котором в виде определенных графических символов и знаков автоматически, в режиме реального времени, отображается информация, получаемая панелью от ПЛК. Этот же экран обеспечивает взаимодействие оператора с ПЛК и контролируемой им системой. Путем сенсорного воздействия на определенные области экрана (то есть касания их пальцем) оператор может запустить или остановить требуемые процедуры (например, включить или выключить определенный агрегат). Необходимо отметить, что все со-

единения уже сделаны внутри стенда, студенту остается по техническому заданию написать программу работы для ПР114 и создать визуализацию автоматизации линии, которая выводиться на панели оператора СП270.

Для правильности составления схемы в программе необходимо пользоваться материалом, изложенным в методическом указании выше. Для образца правильного составления схемы для данного тех. задания, с учетом общепринятых норм и правил автоматизации технологических процессов, представлена функционально-структурная схема (рисунок 3.59).После написания программы на компьютере необходимо произвести загрузку ее в ПР114. Для этого сначала необходимо настроить связь компьютера с ПР114 по методике, изложенной в приложении 2. После этого необходимо записать разработанную программу в прибор. Для этого в меню «Прибор» выбрать пункт «Записать программу в прибор» (рисунок 3.57).

OWEN Logic выдаст окно «Программирование прибора», отображающее статус загрузки (рисунок 3.58). После того, как оно автоматически закроется по окончании записи программы, вы можете проверять работу вашего алгоритма непосредственно на приборе. Не забудьте сохранить ваш алгоритм!

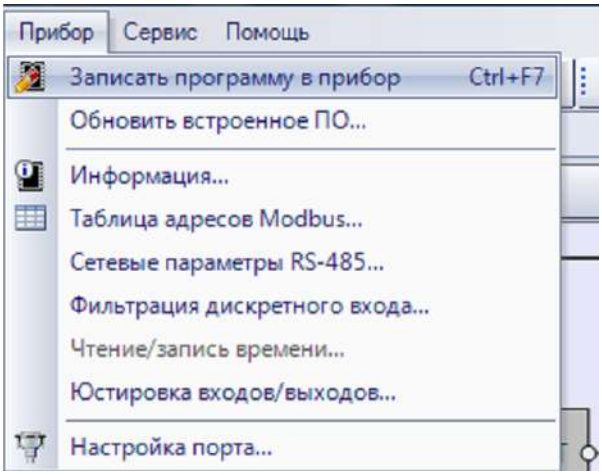


Рисунок 3.57 – Внешний вид записи программы в прибор.

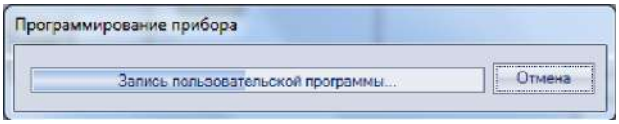


Рисунок 3.58 – Внешний вид записи программы в прибор.

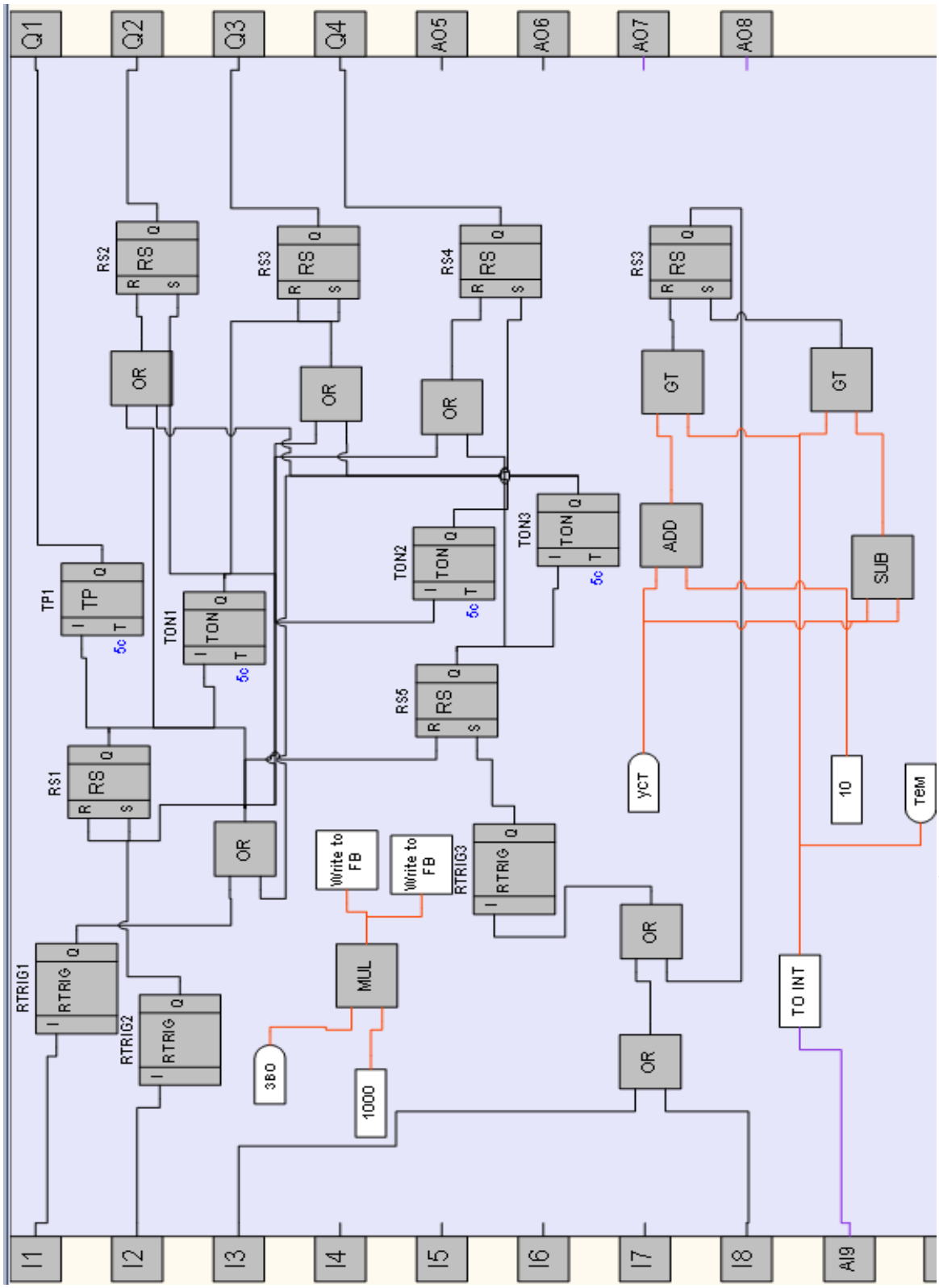


Рисунок 3.59–Структурно-функциональная схема автоматизации предварительной обработки зерна.

Более детально функционально-структурная схема представлена на рисунках 3.60–3.61.

На рисунке 3.60 представлена блок-схема, реализующая выдержку на включения первого механизма и одновременно предупреждающая о начале работы механизмов линии. Она позволяет изменять временные установки на пускосигнальном звене с графической панели оператора, при этом нет необходимости к изменению времени в программе.

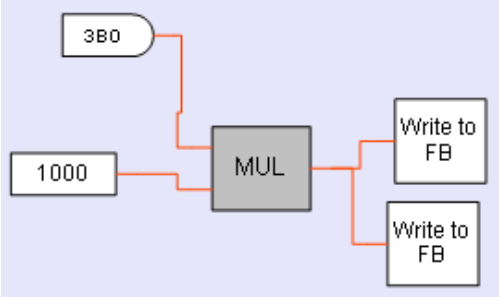


Рисунок 3.60 – Схемная реализация звена уставки времени на пускосигнальном звене.

В последнее время наблюдается тенденция к усложнению техпроцессов. Уже практически не встретишь техпроцесс, в котором для контроля параметров используют только датчики с дискретными сигналами. Зачастую используются датчики с аналоговыми сигналами, позволяющие повысить точность и быстродействие регулирования, динамику процесса. Так для контроля заполнения бункера используется датчик уровня с аналоговым сигналом.

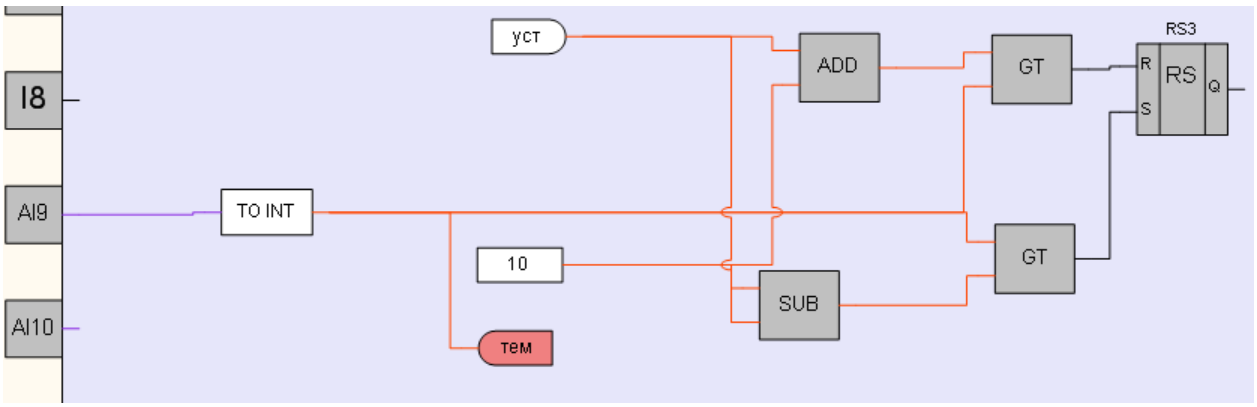


Рисунок 3.61 – Схемная реализация блока аналогового входа.

На рисунке 3.61 представлена схема, реализующая работу аналогового входа, который может быть представлен в виде датчика уровня с токовым

выходом. На рисунке показан алгоритм двухпозиционного регулирования, когда сигнал, приходящий с датчика, сравнивается с заданным значением. Если, например, такой сигнал больше чем заданное значение, то на выходе RS триггера появляется сигнал для отключения механизма, если меньше, то состояние RS на выходе неизменно от первоначального.

Рассмотрим более детально работу блока аналогового входа. С датчика уровня на вход AI9 программируемого реле приходит аналоговый сигнал. Для дальнейшей его обработки и сравнивания его с заданным значением необходимо его преобразовать, а именно, изменить его тип данных в блоке «TOINT» в целочисленный. Это необходимо делать для того, чтобы можно было сравнивать сигнал от датчика уровня с установленным значением.

Для того чтобы расширить зону сравнения сигнала с уставкой, в представленной схеме используют два блока. В блоке «ADD» происходит суммирования значения уставки с числом 10, и на выходе формируется результирующее число. В блоке «SUB» наоборот происходит вычитание из значения уставки числа 10.

Преобразованный сигнал с датчика поступает в блок «GT». В нем осуществляется операция сравнения на большее значение. Если сигнал с датчика больше значения с блока «ADD», то на выходе RS триггера появляется сигнал, что приводит к отключению механизма линии.

Для графического отображения работы линии используется графическая панель СП-270. Для более детального изучения конфигурации панели оператора необходимо обратиться на сайт изготовителя [19], в разделе «Графические панели» изучить материал представленный в файле «Программа – конфигуратор панели оператора СП270. Руководство пользователя».

Разработанный алгоритм работы линии, по технологическому заданию реализован в программе OWEN Logic, позволяет управлять механизмами посредством ПР114. Отображение работы механизмов реализовано в СП270, внешний вид экранов представлен на рисунках 3.62–3.63.

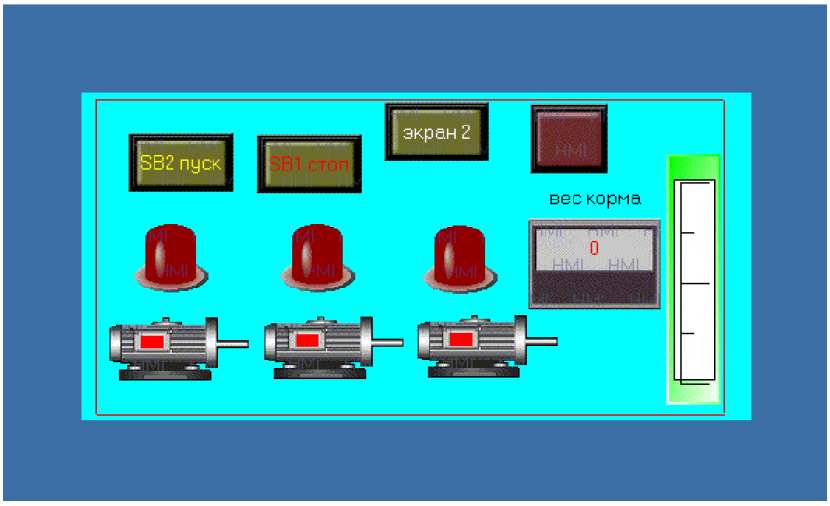


Рисунок 3.62 – Копия основного экран 1 панели СП-270.

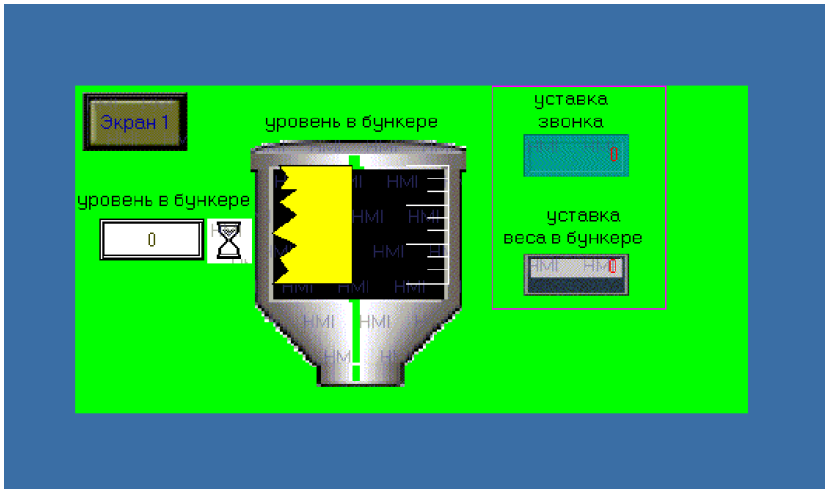


Рисунок 3.63 – Копия экрана 2 панели СП-270.

Описание работы схемы:

При нажатии кнопки «ПУСК» (I2 на схеме, рис. 3.59) сигнал подается на блок RTRIG2, на его выходе лог. «1» появляется на мгновение и пропадает. Этого приводит к появлению на входе «S» RS-триггера «RS1» лог. «1» – на выходе «Q» RS-триггера «SR1» установится лог. «1». Этот сигнал поступает дальше на вход блока «TP1» (блок используется для формирования импульса включения выхода на заданный интервал времени), на выходе лог. «1» подается на выход **Q1**. К выходу **Q1** подключена лампа HL1, имитирующая световую сигнализацию начала работы линии.

Время работы выхода **Q1** определяет блок «TP1». Одновременно с подачей лог. «1» на вход блока «TP1», лог. «1» подается на таймера с задерж-

кой включения «TON1». Через определенное время на выходе блока «TON1» появляется лог. «1», которая приходя на входы RS-триггеров «RS2» и «RS3», производит включение выходов **Q2** и **Q3**. Загораются лампы индикации HL2 и HL3, включаются два механизма линии: скребковый транспортер и дробилка.

Одновременно с этим лог. «1» подается на таймер с задержкой включения «TON2», который через время производит включение выхода **Q4** (загорается лампа индикации HL4, имитирующая работу ковшовой нории).

Все механизмы линии работают, происходит технологический процесс дробления продукта и загрузка его в накопительный бункер. В бункере установлен датчик веса, сигнал которого поступает на аналоговый вход **AI9**. Также в бункере установлен датчик уровня, сигнал с которого приходит на вход **I8.**

Как только вес продукта достигает своего заданного значения, либо срабатывает датчик уровня, либо оператор сам решил прекратить загрузку бункера, появляется лог. «1» на входе блока «OR», сигнал от которого через блок «RTRIG3» поступает на «RS5».C выхода триггера лог. «1» сразу подается на вход «R» RS - триггера «RS4», что приводит к отключению выхода **Q4** и одновременно лог. «1» приходит на вход блока «TON2», который отключает RS-триггеры «RS2» и «RS3» через выдержку по времени.

Цепь входа I1 реализует функцию «Общий стоп» или «аварийный стоп»: при нажатии кнопки «SB1» (I1 на схеме, рис. 3.59) лог. «1» подается на все RS-триггеры: «RS2», «RS3», «RS4», выключающие механизмы линии.

3.3 Программируемые реле фирмы Siemens LOGO 230 RC

3.3.1 Общие сведения

Логические модули LOGO являются компактными функционально законченными универсальными изделиями. Они предназначены для построения простейших устройств автоматики с логической обработкой информации. Алгоритм функционирования модулей задается программой, составленной из набора встроенных функций. Программирование модулей может производиться с их клавиатуры без использования дополнительного программного обеспечения. Стоимостные показатели модулей настолько низки, что их применение может оказаться экономически целесообразным даже в случае замены устройств, включающих в свой состав 2 многофункциональных реле времени или 2 таймера и 3-4 промежуточных реле [21].

Область применения:

1. Управление наружным и внутренним освещением, освещением витрин.
2. Управление коммутационной аппаратурой (АВР, АПВ и т.д.).
3. Управление технологическим оборудованием (насосами, вентиляторами, компрессорами, прессами).
4. Системы отопления и вентиляции.
5. Системы управления дорожным движением.
6. Конвейерные системы.
7. Управление подъемниками.

На рисунке 3.64 представлен внешний вид программируемого реле LOGO 230RC, на котором обозначены функциональные возможности данного оборудования.

Рисунок 3.64 – Внешний вид программируемого реле LOGO 230RC.

На рисунке 3.65 представлена принципиальная схема управления включения и отключения сигнальной лампы E1. Как видим из схемы управления, сигнальная лампа будет гореть только в том случае, когда выключатели S1, S2 и S3 замкнуты.

Попытаемся реализовать представленную принципиальную схему управления с помощью *LOGO.* Схема создается посредством соединения друг с другом блоков и соединительных элементов. В результате получится схема управления, реализованная в *LOGO* с помощью программного продукта LOGO! Soft Comfort на языке функциональных блок-схем FBD.

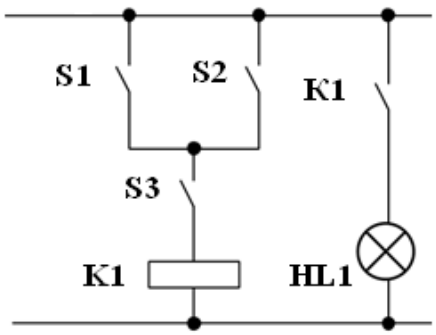


Рисунок 3.65 – Принципиальная схема управления включения сигнальной лампы.

На рисунке 3.66 показана схема подключения к реле выключателей S1, S2, S3 и сигнальной лампы E1, а также реализация логики управления с помощью функциональных блоков.

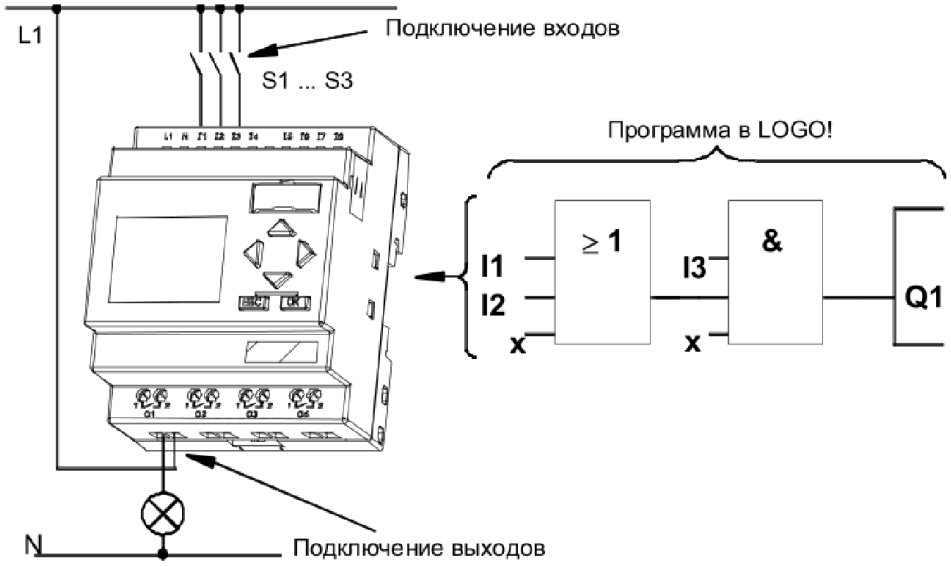


Рисунок 3.66 – Схема подключения реле с входными и выходными устройствами и логика управления.

Правила работы с *LOGO.*

1. Коммутационная программа создается в режиме программирования.
2. Коммутационная программа всегда вводится от выхода к входу.
3. Можно соединить выход с несколькими входами, но не несколько выходов с одним входом.
4. Нельзя соединять выход с предшествующим входом в пределах одного программного пути. Для образования таких внутренних обратных связей (рекурсий) включайте промежуточные флаги или выходы.

В режиме RUN *LOGO* выполняет работу. Для этого сначала считывается состояние входов, а затем с учетом написанной программы, происходит управление выходами. На рисунке 3.67 представлена работа реле с выводом на дисплей состояний дискретных входов/выходов.

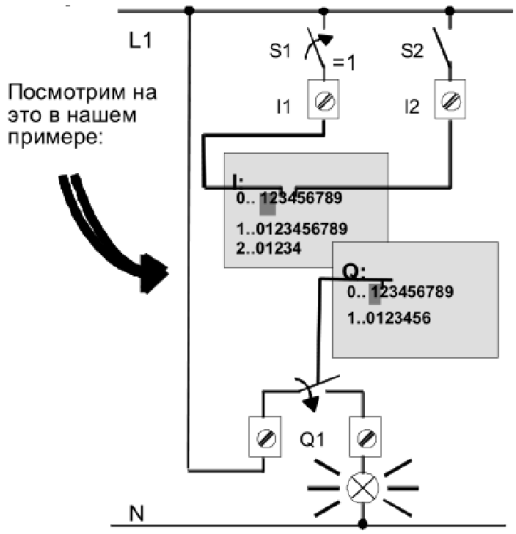


Рисунок 3.67 – Отображение состояния входов/выходов на дисплее.

При замыкании ключа S1 к первому входу реле приложено питание, на дисплее высвечивается 1. По определенной логике программы внутри реле, в результате воздействия на 1 дискретный вход, появляется сигнал на выходе Q 1, при этом на дисплее появляется 1 на выходе.

Классификация функций LOGO!

LOGO! в режиме программирования предоставляет в ваше распоряжение различные элементы. Чтобы при этом не потерять общего представления, эти элементы разделяются на списки. Этими списками являются:

Co: список соединительных элементов (Connector [элемент])

GF: список основных функций AND [И], OR [ИЛИ]

SF: список специальных функций

BN: список готовых к использованию в коммутационной программе блоков.

Входы:

1) цифровые входы

Цифровые входы обозначаются буквой I. Номера цифровых входов (I1, I2,...) соответствуют номерам входных клемм на LOGO! Basic и на подключенных цифровых модулях в том порядке, в котором они установлены.

2) аналоговые входы

У вариантов LOGO! 24, LOGO! 24o, LOGO! 12/24RC и LOGO! 12/24RCo имеются входы I7 и I8, которые могут быть также запрограммированы для использования в качестве входов **AI1** и **AI**2. Если эти входы используются как I7 и I8, то входной сигнал интерпретируется как цифровая величина. Если они используются как AI1 и AI2, то сигналы интерпретируется как аналоговые величины. При подключении аналогового модуля его входы получают номера, следующие за существующими аналоговыми входами.

В случае специальных функций, которые на стороне входов имеет смысл соединять только с аналоговыми входами, при выборе в режиме программирования входного сигнала предлагаются только аналоговые входы AI1...AI8, аналоговые флаги AM1...AM6, номера блоков функции с аналоговым выходом или аналоговые выходы AQ1 и AQ2.

Выходы:

1) цифровые выходы

Цифровые выходы обозначаются буквой Q. Номера выходов (Q1, Q2,... Q16) соответствуют номерам выходных клемм на LOGO! Basic и на подключенных модулях расширения в том порядке, в котором они установлены.

Кроме того, имеется возможность использования 16 неподключенных к блокам выходов. Они обозначены символом **x** и не могут повторно использоваться в коммутационной программе (отличие, например, от флагов). В списке появляются все запрограммированные неподключенные выходы, а также один еще не запрограммированный неподключенный выход. Использование неподключенного выхода имеет смысл, например, у специальной функции «Тексты сообщений », если только текст сообщения имеет значение для коммутационной программы.

2) аналоговые выходы

Аналоговые выходы обозначаются буквами **A**Q. Имеются в распоряжении два аналоговых выхода, а именно, AQ1 и AQ2. К аналоговому выходу можно подключать только аналоговую величину, т.е функцию с аналоговым выходом или аналоговый флаг AM.

Флаги

Флаги обозначаются буквами **M** или **AM.** Это виртуальные выходы, которые имеют на своем выходе такое же значение, как и на своем входе. В LOGO! имеется 24 цифровых флага M1... M24 и 6 аналоговых флагов AM1... AM6.

Флаг запуска

Флаг M8 устанавливается в первом цикле работы программы пользователя и, следовательно, может использоваться в вашей коммутационной программе как флаг запуска. Он автоматически сбрасывается после первого цикла обработки программы. Во всех последующих циклах флаг M8 может использоваться таким же образом, как и другие флаги, для операций установки, удаления и анализа.

Биты регистра сдвига

LOGO! предоставляет в распоряжение биты регистра сдвига S1.S8, которые в коммутационной программе могут только считываться. Содержимое битов регистра сдвига может быть изменено только с помощью специальной функции «Регистр сдвига».

Клавиши управления курсором

В распоряжении пользователя имеется четыре клавиши управления курсором: C ↑, C →, C ↓, C ←, («C» означает «Cursor»). Клавиши управления курсором программируются в коммутационной программе таким же образом, как и другие входы. Клавиши управления курсором можно активизировать на предусмотренном для этого дисплее, когда система находится в режиме RUN, и в активном тексте сообщения(ESC + желаемая клавиша). Использование клавиш управления курсором позволяет экономить выключатели и входы и делает возможным ручное вмешательство в работу коммутационной программы.

Уровни

Уровни напряжения обозначаются **hi** и **l**o. Если на блоке должно постоянно иметь место состояние «1» = hi или «0» = lo, то на вход подается фиксированный уровень или постоянное значение hi или lo.

Открытые соединительные элементы

Если соединительный элемент блока не используется, то его можно обозначить символом **x**.

Основные функций - GF

Основные функции − это простые логические элементы булевой алгебры. Список GF содержит блоки основных функций, которые вы можете использовать в своей коммутационной программе. Имеются следующие основные функции, представленные в таблице 1.

Функция AND (И).

На рисунке 3.68 показано представление функции И на коммутационной схеме и в LOGO!

Рисунок 3.68 − Представление функции «И» в LOGO! и на коммутационной схеме.

Выход «И» принимает состояние 1 только тогда, когда **все** входы имеют состояние 1 (т.е. все контакты замкнуты).Если какой-то вход этого блока не подключен (x), то для этого входа x =1.

Таблица 3.5 – Основные функции LOGO!

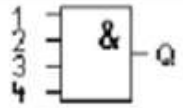
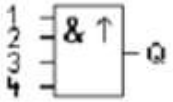
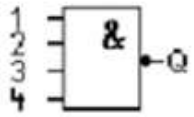
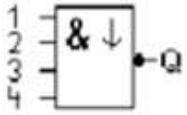
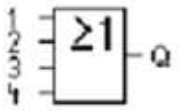
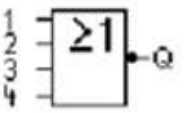
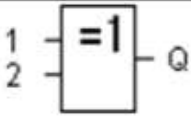
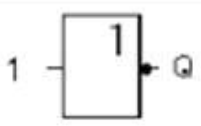
Представление на коммутационной схеме	Представление в LOGO!	Наименование основной функции
Последовательное соединение замыкающих контактов		AND (И)
		AND с анализом фронта
Параллельное соединение размыкающих контактов		NAND (И-НЕ)
		NAND с анализом фронта

Таблица 3.5 (продолжение) – Основные функции LOGO!

Представление на коммутационной схеме	Представление в LOGO!	Наименование основной функции
Параллельное соединение замыкающих контактов		OR (ИЛИ)
Последовательное соединение размыкающих контактов		NOR (ИЛИ-НЕ)
Двойной перекидной контакт		XOR (исключающее ИЛИ)
Размыкающий контакт		NOT (отрицание, инверсия)

AND с анализом фронта.

На рисунке 3.69 показано представление функции «AND» с анализом фронта.

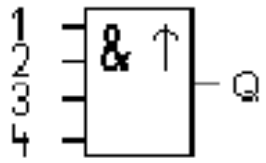


Рисунок 3.69 − Функции AND с анализом фронта.

Выход функции «AND» с анализом фронта принимает состояние 1 только тогда, когда всевходы имеют состояние 1 и хотя бы одинвход в предыдущем цикле имел состояние 0. Если какой-то вход этого блока не используется (x), то для этого входа x = 1.

NAND (И-НЕ)

На рисунке 3.70 показано представление функции «И-НЕ» на коммутационной схеме и в LOGO!

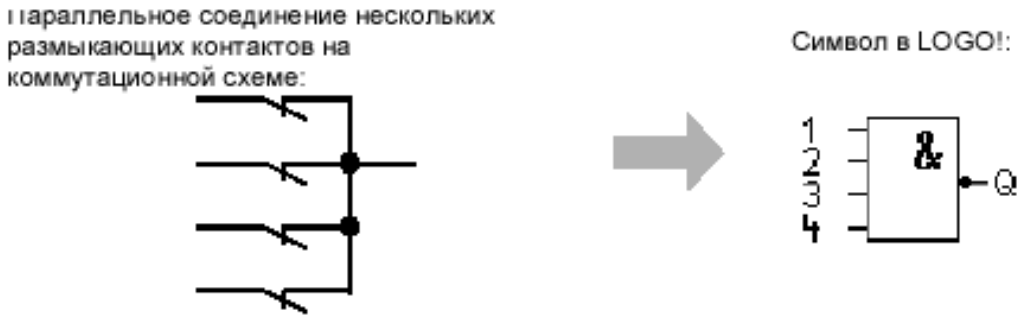


Рисунок 3.70 − Функция И-НЕ на коммутационной схеме и в LOGO!

Выход функции «И-НЕ» принимает состояние 0 только тогда, когда на **все** входы подан сигнал 1 (коммутационной схеме все контакты разомкнуты). Если какой-то вход этого блока не подключен (x), то для этого входа x = 1.

OR (ИЛИ)

На рисунке 3.71 показано представление функции ИЛИ на коммутационной схеме и в LOGO!

Рисунок 3.71 − Представление функции ИЛИ на коммутационной схеме и в LOGO!

Выход функции «ИЛИ» принимает состояние 1, если хотя бы одинвход имеет состояние 1 (т.е. замкнут).

Если какой-то вход этого блока не используется (x),то для этого входа x = 0.

NOR (ИЛИ – НЕ)

На рисунке 3.72 показано представление функции «ИЛИ – НЕ» на коммутационной схеме и в LOGO!

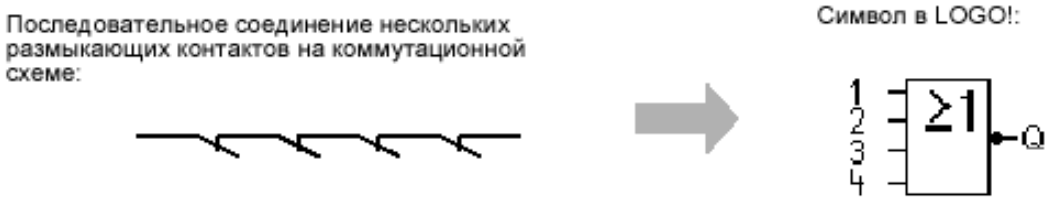


Рисунок 3.72 – Представление функции ИЛИ – НЕ на коммутационной схеме в LOGO!

Выход функции «ИЛИ» принимает состояние 1 только тогда, когда все входы имеют состояние 0 (т. е. они выключены). Как только любой из входов включается (1), выход «И» устанавливается в 0. Если какой-то вход этого блока не используется (x), то для этого входа x = 0.

Основные сведения о специальных функциях

Специальные функции включают в свой состав функции времени, обладают свойством сохраняемости и различными возможностями параметризации, чтобы приспособить программу к вашим индивидуальным потребностям.

Задержка включения

При задержке включения выход включается только по истечении параметризуемого интервала времени.

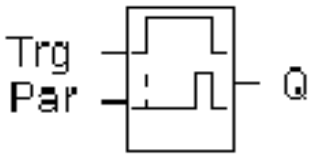


Рисунок 3.73− Изображение в LOGO! функции «задержка включения».

Вход Trg. Через вход **Trg** (trigger = запустить) производится запуск отсчета времени задержки включения. **Параметр T** представляет время, по истечении которого включается выход (сигнал переключается с 0 на 1).

Выход Q включается по истечении заданного времени T, если Trg все еще установлен.

Параметр T. Заданием времени для параметра T может также служить фактическое значение другой, уже запрограммированной функции. Вы можете использовать фактические значения следующих функций:

Задержка выключения.

При задержке выключения выход сбрасывается только по истечении заданного интервала времени. На рисунке 3.74 показано изображение в LOGO! функции «задержка выключения».

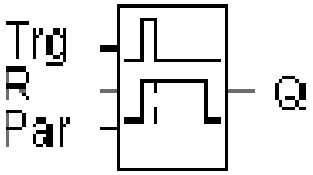


Рисунок 3.74− Изображение в LOGO! функции «задержка выключения».

Вход Trg. Задержка выключения запускается отрицательным фронтом сигнала (с 1 на 0) на входе Trg (trigger = запустить). **Вход R**. Сигнал на входе R сбрасывает время задержки выключения и устанавливает выход в 0. **Параметр T** − это время, через которое выключается выход (сигнал переключается с 1 на 0). **Выход Q** устанавливается сигналом на входе Trg. Он сохраняет это состояние, пока не истечет время T.

Параметр T. Заданием времени для параметра T может также служить фактическое значение другой, уже запрограммированной функции.

Задержка включения и выключения

При задержке включения и выключения выход устанавливается по истечении заданной задержки включения и сбрасывается по истечении заданной задержки выключения. На рисунке 3.75 представлено изображение функции «задержка включения и выключения» в LOGO!

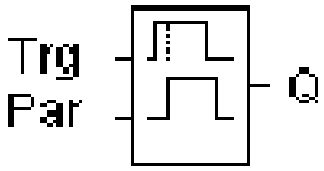


Рисунок 3.75 − Изображение в LOGO! функции «задержка включения и выключения».

Вход Trg. Нарастающий фронт (с 0 на 1) на входе Trg (trigger = запустить) запускает время TH для задержки включения. Падающий фронт (с 1 на 0) запускает время TL для задержки выключения. **Параметр TH**− это время, по истечении которого выход включается (сигнал переключается с 0 на 1). **TL** − это время, по истечении которого выход выключается (сигнал переключается с 1 на 0). **Выход Q** включается по истечении заданного времени TH, если Trg еще установлен, и выключается по истечении времени TL, если Trg не будет тем временем снова установлен.

Интервальное реле (ввод импульса)

Входной импульс вызывает появление сигнала заданной длительности на выходе. На рисунке 3.76 представлено изображение в LOGO! функции «интервальное реле».

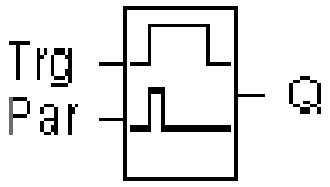


Рисунок 3.76− Изображение в LOGO! функции «интервальное реле».

Вход Trg. Сигнал на входе Trg (trigger = запустить) запускает отсчет времени для интервального реле. **Параметр T**− это время, через которое выключается выход (сигнал переключается с 1 на 0). **Выход Q** включается одновременно с Trg и остается включенным в течение времени, если входной сигнал остается равным 1.

Самоблокирующееся реле.

Вход S устанавливает выход Q, вход R снова сбрасывает выход Q. На рисунке 3.77 представлена временная диаграмма функции «самоблокирующееся реле».

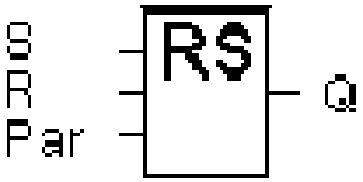


Рисунок 3.77 − Временная диаграмма функции «самоблокирующееся реле».

Пользовательский интерфейс.

Для осуществления интерфейса между реле и человеком существует программа LOGO! Soft Comfort V6.0. Она позволяет производить написание программы работы программируемого реле в интуитивно понятной форме

для человека. Функциональные возможности программы огромны, что позволяет создавать сложные алгоритмы управления.

Основная особенность программного продукта LOGO! Soft Comfort V6.0 состоит в том, что она позволяет конвертировать алгоритм управления процессом из одного вида в другой. При этом присутствует универсальность. Если разработчику близки релейно-контактные схемы, то реализация алгоритма управления будет осуществляться на языке релейно-контактных схем LAD. Если же разработчику близки бесконтактные элементы схем управления, то реализация алгоритма управления будет осуществляться на языке функциональных блок-схем FBD. Внешний вид программного продукта LOGO! Soft Comfort V6.0 представлен на рисунке 3.78.

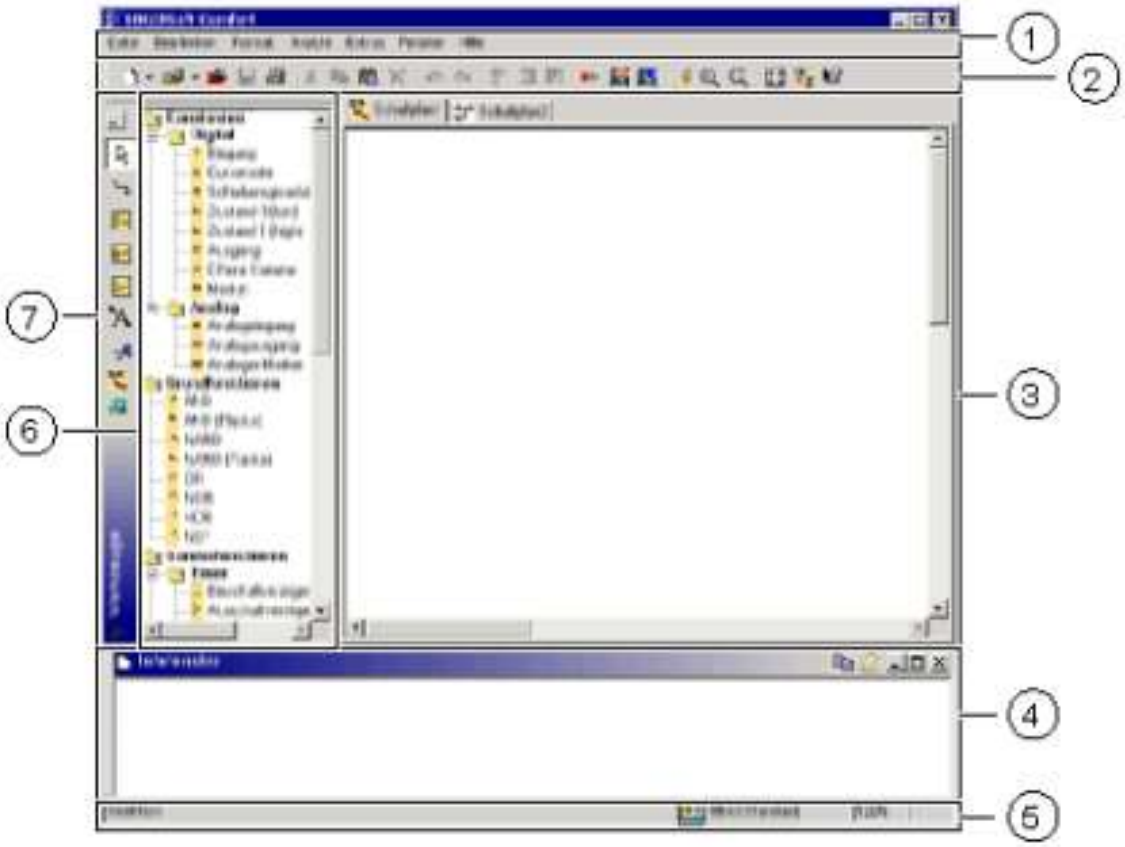


Рисунок 3.78 –Внешний вид программной оболочки LOGO! Soft Comfort.

На рисунке 3.78 обозначаются:

1- Панель меню

2- Стандартная панель инструментов

3- Интерфейс программирования

4- Окно информации

5- Строка состояния

6- Постоянные и соединительные: базовые и специальные функции

7- Панель инструментов программирования

Функциональные клавиши и клавиши быстрого выбора

В программе реализован целый ряд функциональных клавиш и клавиш быстрого выбора часто используемых функций с целью упрощения работы с программой LOGO! Soft Comfort. Функциональные клавиши в программе LOGO! Soft Comfort: [F1] Вызывает контекстно-зависимую оперативную справку; [F2] Сервис-> Определить LOGO!; [F3] Запуск/выход из режима эмуляции; [F4] открытие/закрытие Вид -> Окно информации; [F5] Инструмент соединитель; [F6] Инструмент постоянные и клеммы; [F7] Инструмент базовые функции; [F8] Инструмент специальные функции; [F9] Инструмент текст; [F10] Открывает панель меню; [F11] Инструмент разрезать/связать; [F12] Инструмент эмуляция.

Редактирование блоков.

Нажатие правой кнопкой мыши над объектом открывает меню быстрого выбора, предлагающее различные параметры редактирования объекта. Параметры выбора зависят от выбранного объекта. Объекты состоят не только из блоков и соединительных линий, но и интерфейса программирования и панелей инструментов. Вы также можете вызвать из меню быстрого выбора справку по выбранному объекту. На рисунке 3.79 представлен внешний вид редактирования блока задержки отключения.

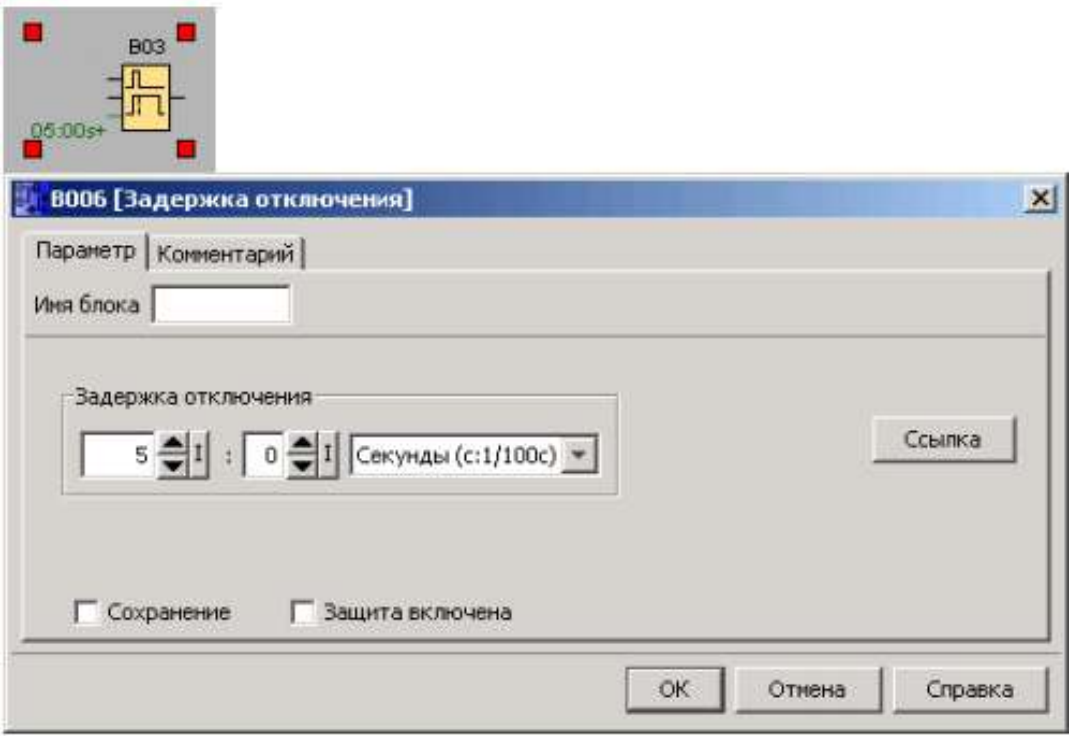


Рисунок 3.79 – Внешний вид редактирования блока задержки отключения.

Соединение блоков.

Для выполнения электрической схемы необходимо выполнить соединение блоков. На панели инструментов программирования выберите значок соединения блоков. На рисунке 3.80 показан значок функции «соединение блоков».

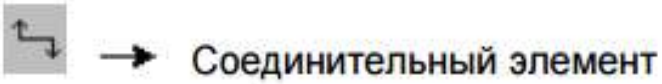


Рисунок 3.80 – Внешний вид функции «соединение блоков».

Для соединения функциональных блоков производят следующий алгоритм действий. Необходимо навести указатель мыши на соединительный элемент блока. Нажать левую кнопку мыши и удерживать ее нажатой. Переместить указатель мыши от исходного соединительного элемента к целевому соединительному элементу. Отпустить кнопку мыши. Программа LOGO! Soft Comfort выполняет соединение двух клемм. Пример для релейно-контактных схем показан на рисунке 3.81.

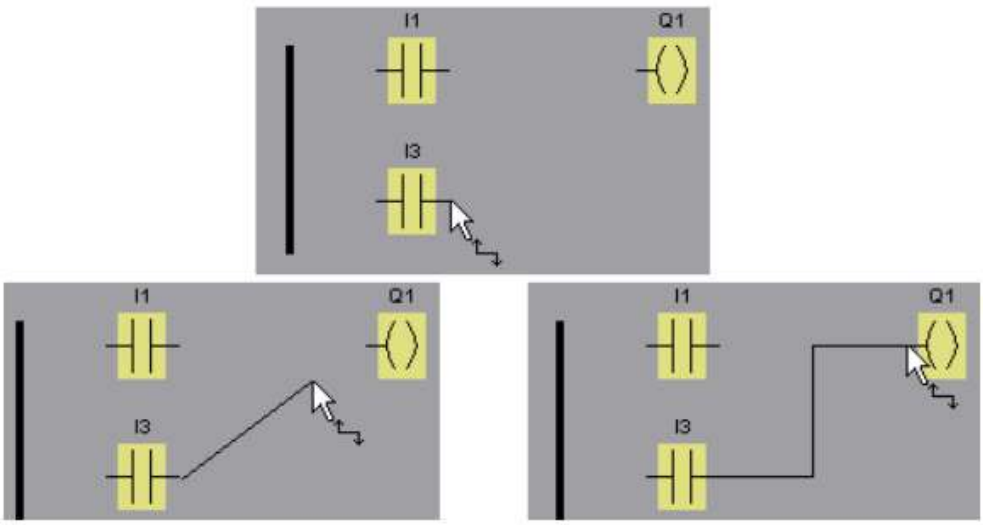


Рисунок 3.81 – Внешний вид соединения элементов для релейно-контактных схем.

Разрезание соединений.

Компоновка большой схемы может вызывать трудности при ее интерпретации, особенно при наличии в ней большого числа пересечений линий.

Вы можете оптимизировать расположение соединений при помощи инструмента «разрезать/связать» на панели инструментов программирования. На рисунке 3.82 представлен значок функции «разрезать/связать».

Рисунок 3.82 – Внешний вид функции «разрезать/связать».

Алгоритм работы данной функции следующий: нажмите на соединение после вызова этого инструмента. Происходит графическое разделение выбранной соединительной линии. Тем не менее соединение между блоками продолжает оставаться активным. Открытые концы разрезанного соединения теперь отображаются со значками в виде стрелок, указывающих на направление протекания сигнала. Над значками теперь отображаются перекрестные ссылки, включая номер страницы электрической схемы, номер блока и номер клеммы блока, соединенного с открытым соединением. Нажмите правую

кнопку на линии, соединяющей два блока, которую вы хотите разрезать, после чего выберите команду резки. Вы можете также разрезать группу соединений, воспользовавшись для этого командой меню Правка -> Разрезать соединения. Перед тем, как разрезать какие-либо соединения, существует возможность выбора критериев резки, например, можно разрезать все соединения, проходящие через блоки. Пример для редактора функциональных блок-схем показан на рисунке 3.83.

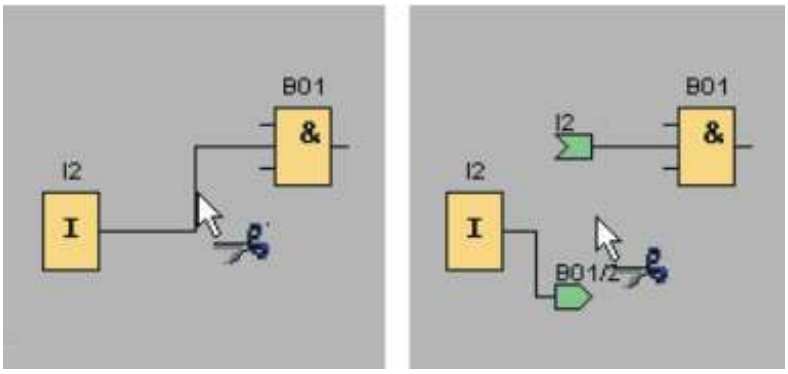


Рисунок 3.83 – Внешний вид использования функции «разрезать/связать» для функциональных блок-схем FBD.

3.3.2 Пример выполнения схемы автоматизации линии предварительной обработки зерна на базе Siemens LOGO 230RC

Реализовывать схему автоматизации линии (см. п. 2.4) будем на лабораторном оборудовании, внешний вид которого изображен на рисунке 3.84. Питание стенда осуществляется от однофазной цепи переменного напряжения 220 В. Лабораторный стенд выполнен в пластиковом коробе, лицевая панель выполнена из изоляционного материала, на поверхность которой нанесена пленка с информационным материалом.

Все токоведущие части оборудования располагаются внутри стенда, что позволяет избежать воздействия электрического тока на людей. Подключение дискретных входов и выходов программируемого реле происходит через специальные изоляционные гильзы.

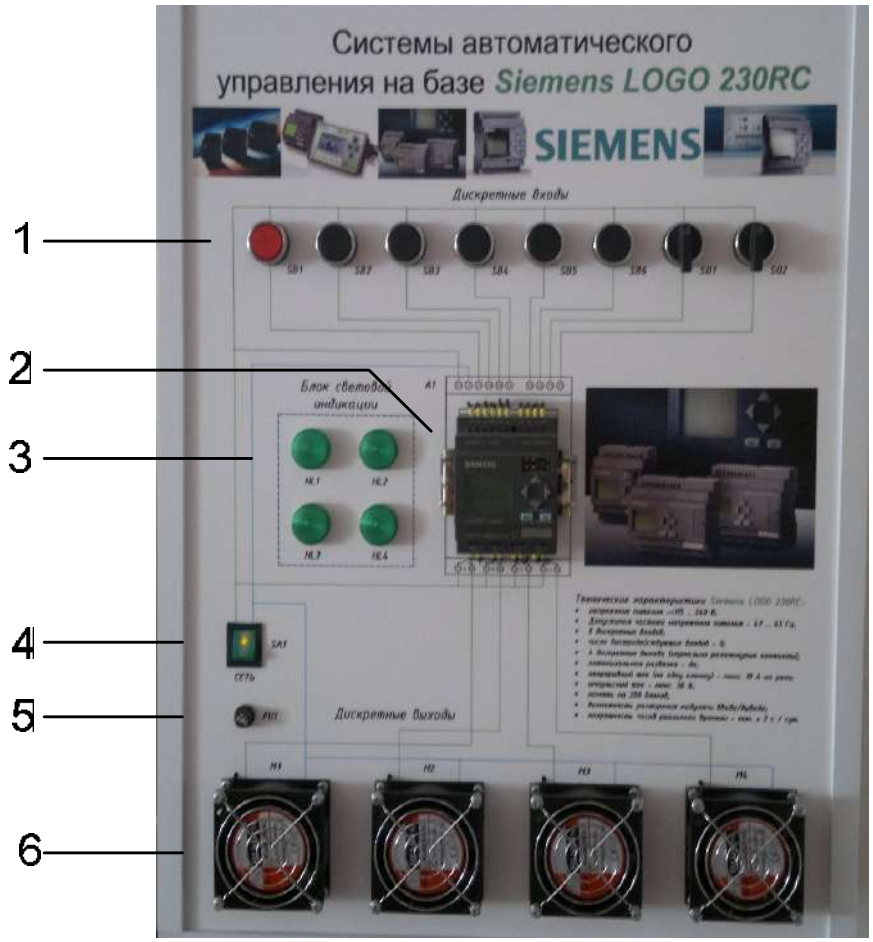


Рисунок 3.84 – Внешний вид лабораторного стенда «Системы автоматического управления на базе Siemens LOGO 230RC».

На рисунке 3.84 обозначены: 1 – дискретные входы реле (кнопки управления); 2 – программируемое реле Siemens LOGO 230RC; 3 – блок световой индикации; 4 – кнопка включения/отключения питания стенда; 5 – предохранитель; 6 – дискретные выходы (вентиляторы, имитирующие работу механизмов линии).

Обозначения сигналов управления, используемые при составлении программы управления, указаны в таблицах 3.6 и 3.7.

Таблица 3.6 − Входные дискретные сигналы.

№ п/п	Наименование сигнала	Имя источника	Разряд-ряд-ность	Сигнал
1	2	3	4	5
1	Кнопка «Общий стоп»	I2	bit	DI 220 V AC
2	Кнопка «Пуск»	I1	bit	DI 220 V AC
3	Кнопка «Рабочий стоп»	I3	bit	DI 220 V AC
4	Датчик уровня	I4	bit	DI 220 V AC

Таблица 3.7− Перечень выходных сигналов и данных.

№ п/п	Наименование сигнала	Имя источника	Диапа-зон из-мене-ния	Пользователь-ская информация
1	2	3	4	5
1	Включение звонка	Q1	1 Bit	DO 1
2	Включение скребкового транспортера	Q2	1 Bit	DO 2
3	Включение молотковой дробилки	Q3	1 Bit	DO 3
4	Включение ковшовой нории	Q4	1 Bit	DO 4

При написании программы для конкретного технологического процесса воспользуемся типовыми блоками схем автоматизации [16].

Нереверсивная схема управления электродвигателем реализована в программе LOGO! Soft Comfort на языке FBD (рисунок 3.85) и LAD (рисунок 3.86). На рисунке 3.85 представлена функциональная блок-схема, на которой I1 − входной блок, на который приходит сигнал с кнопки «пуск»; I2 – входной блок, на который приходит сигнал с кнопки «стоп»; B004 и B005 – интервальные реле с запуском по фронту; B004 (RS) – реле с блокировкой и Q1 – выходной блок, катушка выходного реле (магнитного пускателя).

При нажатии на кнопку «пуск» появляется сигнал на выходе блока I1, который, через интервальное реле с запуском по фронту, подается на реле с блокировкой RS. При появлении сигнала на входе S реле с блокировкой, на выходе сигнал будет присутствовать всегда, т.е. он запомнил состояние сигнала на входе S. Если на выходе RS присутствует сигнал, значит, он существует и на выходном блоке Q1, а, следовательно, и замкнут выходной контакт реле 1 LOGO.

Для отключения выходного блока Q1 необходимо снять сигнал с выхода реле с блокировкой RS. Для этого необходимо нажать кнопку «стоп», в результате чего появится сигнал на входе R реле с блокировкой RS, что приведет к сбросу.

Для избежание пагубного влияния работы механических частей коммутирующей аппаратуры (кнопок, тумблеров, концевых выключателей) рекомендуется в схеме управления ставить последовательно с входными блоками сигналов I1 интервальные реле с запуском по фронту B004. Эти реле в схеме управления позволяют пропускать через себя сигнал определенное время, после чего на их выходе сигнал пропадает. Использование интервальных реле с запуском по фронту повышает работоспособность схемы управления.

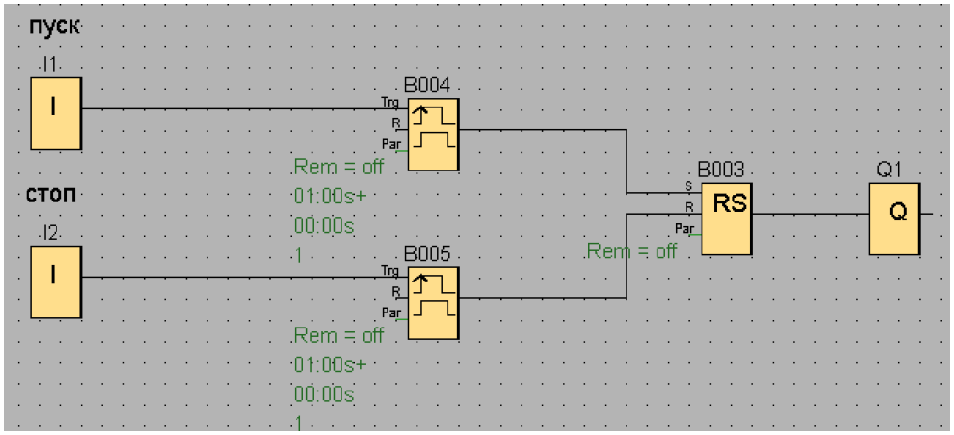


Рисунок 3.85 – Нереверсивная схема управления электродвигателем на языке функциональных блок-схем FBD.

Универсальность программы LOGO! Soft Comfort заключается в том, что она позволяет перевод уже разработанной схемы управления в другой

язык программирования. В результате схема нереверсивного управления электродвигателем, разработанная на языке функциональных блок-схем FBD, трансформируется в схему управления на языке релейно-контактных схем LAD (рисунок 3.86).

Как видим из рисунка, основные блоки схемы управления остались прежними, только изменилась конфигурация, в частности появились замыкающие контакты с катушкой.

При нажатии на кнопку «пуск» замыкается контакт I1, который подает сигнал на интервальное реле с запуском по фронту T004, которое уже через свой замыкающий контакт T004 подается на реле с блокировкой RS. При появлении сигнала на входе S реле с блокировкой его контакт SF003 замыкает цепь с выходной катушкой реле Q1, что приводит к замыканию 1 выходного контакта реле LOGO.

Для отключения выходной катушки реле Q1 необходимо разомкнуть контакт SF003 реле с блокировкой RS. Для этого необходимо нажать кнопку «стоп», в результате чего появится сигнал на входе R реле с блокировкой RS, что приведет к сбросу.

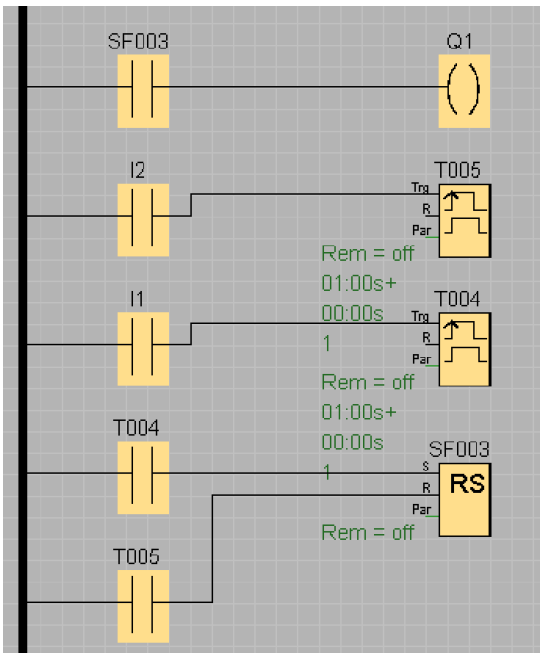


Рисунок 3.86 – Нереверсивная схема управления электродвигателем на языке релейно-контактных схем LAD.

Звено «Рабочий стоп» предназначено для правильного отключения механизмов линии. Позволяет отключить механизмы линии с полной их очисткой от продукта.

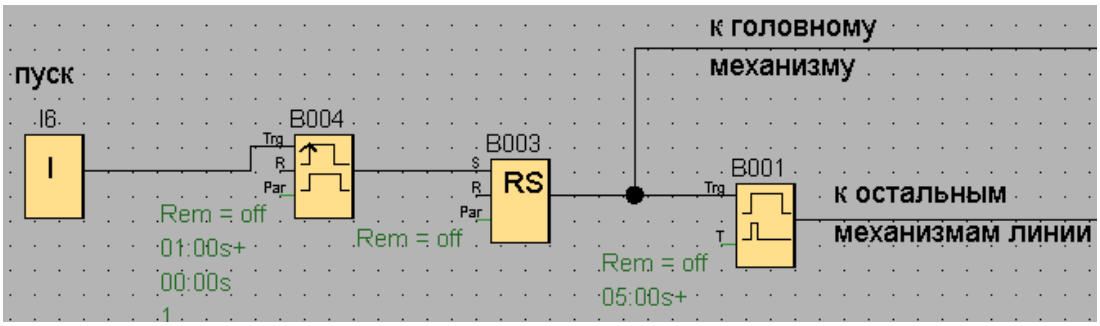


Рисунок 3.87 – Схема звена «Рабочий стоп» на языке функциональных блок-схем FBD.

Представленная на рисунке 3.87 схема работает следующим образом. При нажатии на кнопку пуск «Рабочий стоп» замыкается контакт I6, который подает через интервальное реле с запуском по фронту B004 сигнал на реле с блокировкой (RS) B003. Сигнал на выходе реле запоминается и далее подается на отключение механизмов линии. При этом отключение головного механизма происходит мгновенно, а вот остальные механизмы линии отключаются либо все одновременно с помощью интервального реле B001, либо ступенчато в направлении хода продукта.

На рисунке 3.88 представлена схема управления звена «Рабочий стоп», реализованная на языке релейно-контактных схем LAD. Принцип построения уже рассмотрен нами ранее. Поэтому подробно на этом вопросе останавливаться не будем.

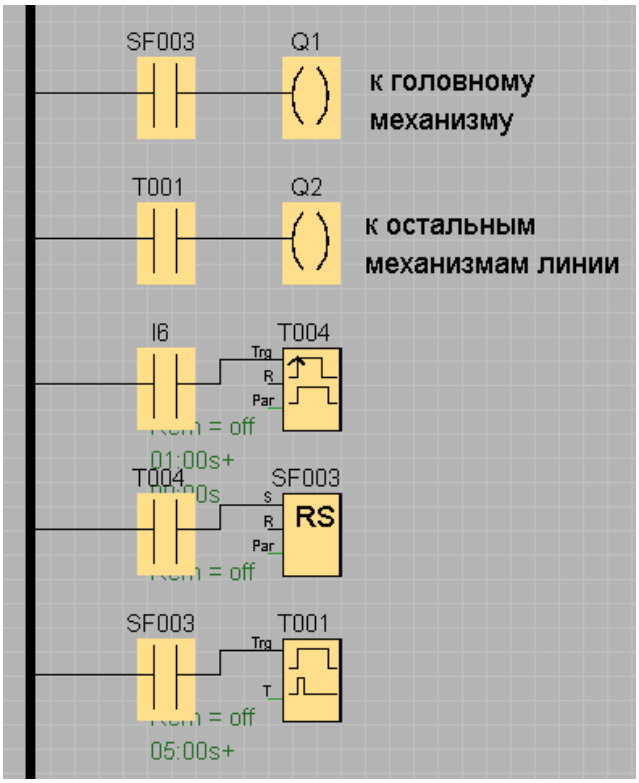


Рисунок 3.88 – Схема звена «Рабочий стоп» на языке релейно-контактных схем LAD.

Пускосигнальное звено предназначено для предупреждения персонала о запуске механизма. Позволяет включить электродвигатель с задержкой по времени, при этом звучит звуковая сигнализация.

На рисунке 3.89 представлена схема пускосигнального звена управления электродвигателем. Помимо уже известных нам входных I и выходных Q блоков, реле с блокировкой RS и интервальных реле с запуском по фронту B004 и B005, появились новые элементы: B001 – интервальное реле (импульсный вход), служащее для отключения с задержкой по времени выходного блока Q1 (звонка); B007 – блок задержки включения (таймер с задержкой включения), производит включение выходного блока Q2 (механизма) с задержкой времени.

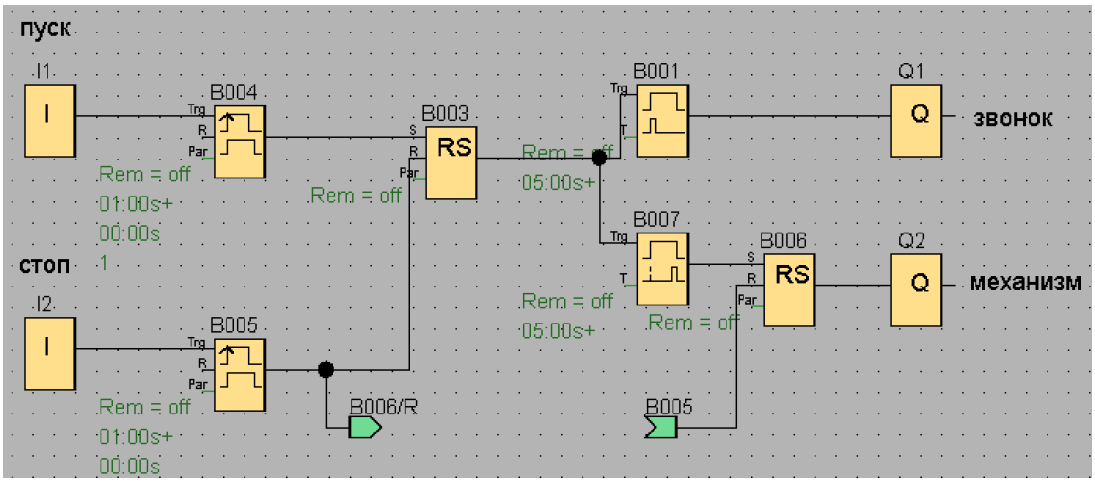


Рисунок 3.89 – Схема пускосигнального звена на языке функциональных блок-схем FBD.

Все временные задержки настраиваются в программе. В данном примере временные задержки составляют 5 с.

Для упрощения чтения разработанных графических алгоритмов работы системы используют так называемые разрывные линии. При этом происходит графическое разделение выбранной соединительной линии. Тем не менее соединение между блоками продолжает оставаться активным. Открытые концы разрезанного соединения теперь отображаются со значками в виде стрелок, указывающих направление протекания сигнала. Так на нашем примере связь между блоками B005 и B006 представлена в виде стрелок, название которых указывают адрес присоединения.

Представленная схема пускосигнального звена работает следующим образом. При нажатии на кнопку «пуск» появляется сигнал на выходе блока I1, который, через интервальное реле с запуском по фронту, подается на реле с блокировкой RS. При появлении сигнала на входе S реле с блокировкой, на выходе сигнал запоминается и подается на B001 – интервальное реле и B007 – блок задержки включения. Реле B001 мгновенно пропускает сигнал на определенное время, в нашем случае на 5 с., при этом выходной блок Q1 включен, а следовательно, звенит звонок. Блок задержки включения B007 наоборот пропустит через себя сигнал лишь с выдержкой времени (в нашем случае

5 с.). В результате выходной блок Q2 получит сигнал через реле с блокировкой RS и включит механизм.

Для отключения механизма необходимо наличие сигнала на реле с блокировкой RS (B006) на его входе сброса R. Этот сигнал появится лишь при нажатии кнопки «стоп» входного блока I2.

Схема управления пускосигнального звена, реализованная на языке релейно-контактных схем LAD, изображена на рисунке 3.90. При нажатии на кнопку «пуск» замыкается контакт I1, который подает сигнал на интервальное реле с запуском по фронту T004, которое уже через свой замыкающий контакт T004 подается на реле с блокировкой RS (SF003). При появлении сигнала на входе S реле с блокировкой его контакт SF003 замыкает цепь с интервальным реле T001, замыкающий контакт которого подаст сигнал через свой контакт T001 на катушку Q1 на время 5с., после которого разомкнется. Это приведет к включению звонка.

Одновременно с появлением сигнала на интервальным реле T001 сигнал поступает на триггер задержки включения T007, контакт которого T007, с задержкой по времени, подаст сигнал на реле с блокировкой RS (SF006), которое, в свою очередь, замкнет цепь с катушкой Q2. Это приведет к включению механизма.

Для отключения механизма необходимо снять питание с катушки Q2. Для этого необходимо нажать на кнопку «стоп» (контакт I2). При этом появится сигнал на входах R реле с блокировкой SF003 и SF006, что приведет к их сбросу.

На основании разработанных типовых схем составим программу управления линии предварительной обработки зерна, внешний вид которой изображен на рисунке 3.91. Как видим из рисунка, представленная схема управления состоит из звеньев, рассмотренных нами ранее.

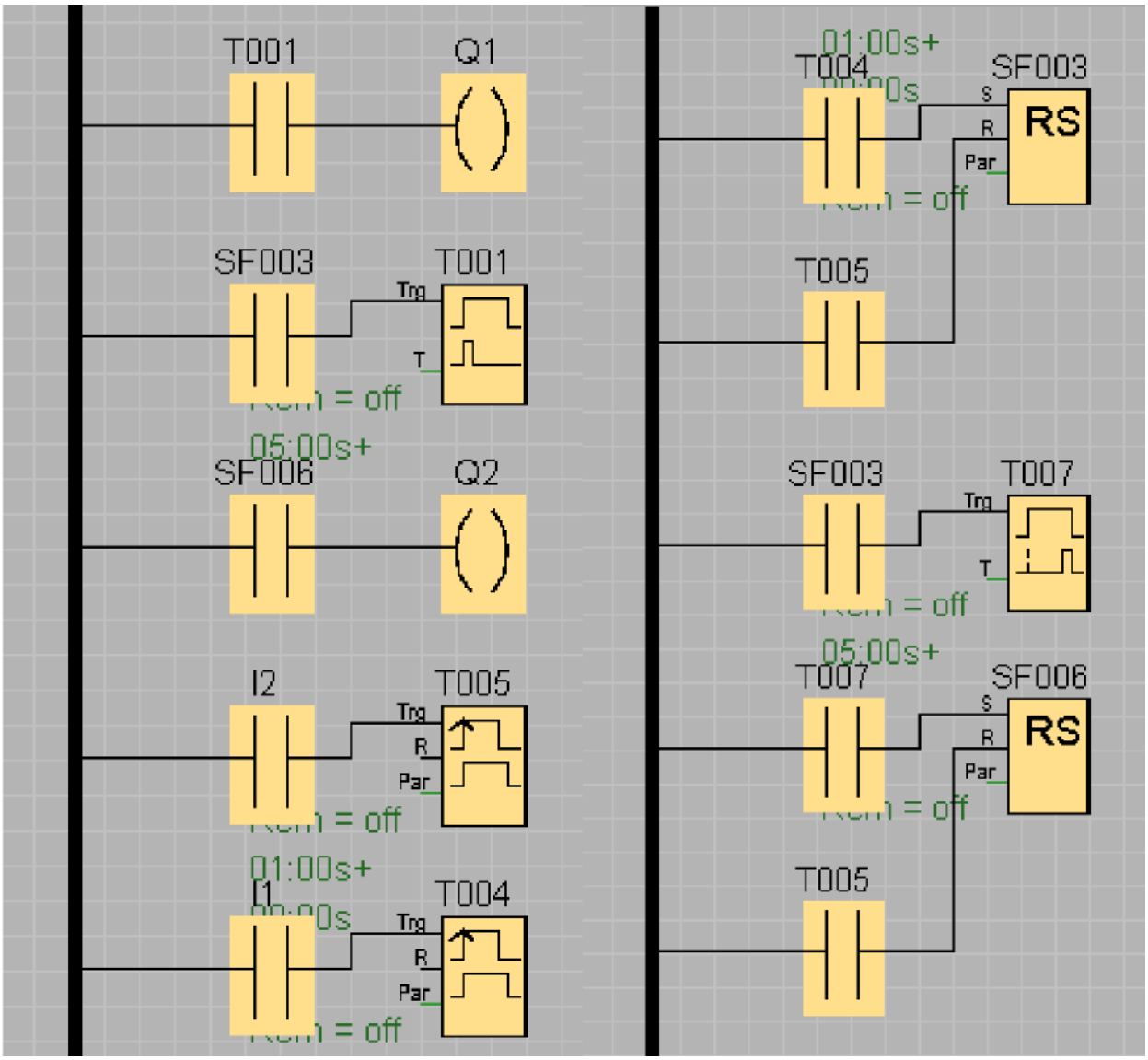


Рисунок 3.90 – Схема пускосигнального звена на языке релейно-контактных схем LAD.

Представленная схема рассчитана на управление тремя механизмами, поскольку, к сожалению, технически реализовать управление большим количеством механизмов невозможно.

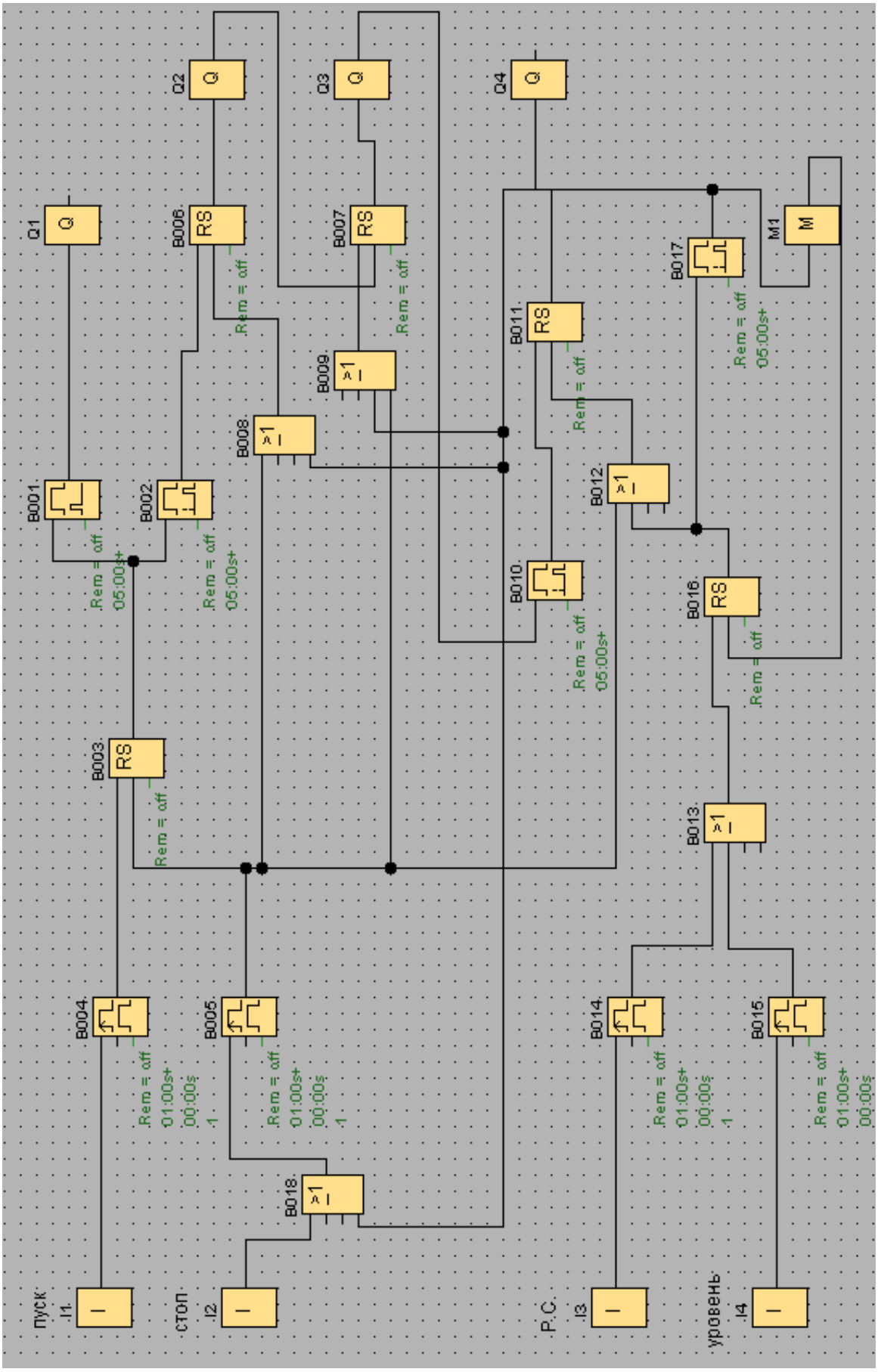


Рисунок 3.91 – Программа управления линией обработки зерна.

4. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ

ПЛК – программируемый логический контроллер, представляющий собой микропроцессорное устройство, предназначенное для сбора, преобразования, обработки, хранения информации и выработки команд управления, имеющий конечное количество входов и выходов, подключенных к ним датчиков, ключей, исполнительных механизмов к объекту управления, и предназначенный для работы в режимах реального времени [2].

Принцип работы ПЛК несколько отличается от «обычных» микропроцессорных устройств. Программное обеспечение универсальных контроллеров состоит из двух частей. Первая часть − это системное программное обеспечение. Проводя аналогию с компьютером, можно сказать, что это операционная система, т.е. управляет работой узлов контроллера, взаимосвязи составляющих частей, внутренней диагностикой. Системное программное обеспечение ПЛК расположено в постоянной памяти центрального процессора и всегда готово к работе. При включении питания ПЛК готов взять на себя управление системой уже через несколько миллисекунд. ПЛК работает циклически по методу периодического опроса входных данных.

Рабочий цикл ПЛК включает 4 фазы:

1. Опрос входов.

2. Выполнение пользовательской программы.

3. Установка значений выходов.

4. Некоторые вспомогательные операции (диагностика, подготовка данных для отладчика, визуализации и т. д.).

Выполнение 1 фазы обеспечивается системным программным обеспечением. После чего управление передается прикладной программе, той программе, которую вы сами записали в память, по этой программе контроллер делает то, что вы пожелаете, а по ее завершении управление опять передается системному уровню. За счет этого обеспечивается максимальная простота построения прикладной программы – ее создатель не должен знать, как про-

изводится управление аппаратными ресурсами. Необходимо знать, с какого входа приходит сигнал и как на него реагировать на выходах.

Очевидно, что время реакции на событие будет зависеть от времени выполнения одного цикла прикладной программы. Определение времени реакции – времени от момента события до момента выдачи соответствующего управляющего сигнала – поясняется на рисунке.

Обладая памятью, ПЛК в зависимости от предыстории событий, способен реагировать по-разному на текущие события. Возможности перепрограммирования, управления по времени, развитые вычислительные способности, включая цифровую обработку сигналов, поднимают ПЛК на более высокий уровень в отличие от простых комбинационных автоматов.

Рассмотрим режим входа и выхода ПЛК. Существует три вида входов: дискретные, аналоговые и специальные.

Один дискретный вход ПЛК способен принимать один бинарный электрический сигнал, описываемый двумя состояниями – включен или выключен. Все дискретные входы (общего исполнения) контроллеров обычно рассчитаны на прием стандартных сигналов с уровнем 24 В постоянного тока. Типовое значение тока одного дискретного входа (при входном напряжении 24 В) составляет около 10 мА.

Аналоговый электрический сигнал отражает уровень напряжения или тока, соответствующий некоторой физической величине в каждый момент времени. Это может быть температура, давление, вес, положение, скорость, частота и т. д.

Поскольку ПЛК является цифровой вычислительной машиной, аналоговые входные сигналы обязательно подвергаются аналого-цифровому преобразованию (АЦП). В результате образуется дискретная переменная определенной разрядности. Как правило, в ПЛК применяются 8−12-разрядные преобразователи, что в большинстве случаев, исходя из современных требований по точности управления технологическими процессами, является достаточным. Кроме этого, АЦП более высокой разрядности не оправдывают

себя, в первую очередь из-за высокого уровня индустриальных помех, характерных для условий работы контроллеров.

Практически все модули аналогового ввода являются многоканальными. Входной коммутатор подключает вход АЦП к необходимому входу модуля.

Стандартные дискретные и аналоговые входы ПЛК способны удовлетворить большинство потребностей систем промышленной автоматики. Необходимость применения специализированных входов возникает в случаях, когда непосредственная обработка некоторого сигнала программно затруднена, например, требует много времени.

Наиболее часто ПЛК оснащаются специализированными счетными входами для измерения длительности, фиксации фронтов и подсчета импульсов.

Например, при измерении положения и скорости вращения вала очень распространены устройства, формирующие определенное количество импульсов за один оборот – поворотные шифраторы. Частота следования импульсов может достигать нескольких мегагерц. Даже если процессор ПЛК обладает достаточным быстродействием, непосредственный подсчет импульсов в пользовательской программе будет весьма расточительным по времени. Здесь желательно иметь специализированный аппаратный входной блок, способный провести первичную обработку и сформировать необходимые для прикладной задачи величины.

Вторым распространенным типом специализированных входов являются входы, способные очень быстро запускать заданные пользовательские задачи с прерыванием выполнения основной программы – входы прерываний.

Дискретный выход также имеет два состояния – включен и выключен. Они нужны для управления: электромагнитных клапанов, катушек, пускателей, световых сигнализаторов и т.д. В общем, сфера их применения огромна и охватывает почти всю промышленную автоматику.

При создании системы управления технологического процесса всегда существует проблема по взаимопониманию программиста и технологов. Технолог скажет «нам надо немного подсыпать, чуть подмешать, еще подсыпать и чуть нагреть». И мало когда следует ждать от технолога формализованного описания алгоритма. И получалось так, что программисту нужно долго вникать в технологический процесс, потом писать программу. Зачастую при таком подходе программист остается единственным человеком, способным разобраться в своем творении, со всеми вытекающими отсюда последствиями. Такая ситуация породила стремление к созданию технологических языков программирования, доступных инженерам и технологам и максимально упрощающих процесс программирования.

За последнее десятилетие появилось несколько технологических языков. Более того, Международной Электротехнической Комиссией разработан стандарт МЭК-61131-3, концентрирующий все передовое в области языков программирования для систем автоматизации технологических процессов. Этот стандарт требует от различных изготовителей ПЛК предлагать команды, являющиеся одинаковыми и по внешнему виду, и по действию.

Стандарт специфицирует 5 языков программирования [5]:

1. Sequential Function Chart (SFC) – язык последовательных функциональных блоков;
2. Function Block Diagram (FBD) – язык функциональных блоковых диаграмм;
3. Ladder Diagrams (LAD) – язык релейных диаграмм;
4. Statement List (STL) – язык структурированного текста, язык высокого уровня. Напоминает собой Паскаль;
5. Instruction List (IL) – язык инструкций, это типичный ассемблер с аккумулятором и переходом по метке.

Язык LAD или KOP (с немецкого Kontaktplan) похожи на электрические схемы релейной логики. Поэтому инженерам, не знающим мудреных языков программирования, не составит труда написать программу. Язык FBD

напоминает создание схем на логических элементах. В каждом из этих языков есть свои минусы и плюсы. Поэтому при выборе специалисты основываются в основном на личном опыте. Хотя большинство программных комплексов дают возможность переконвертировать уже написанную программу из одного языку в другой. Так некоторые задачи изящно и просто решаются на одном языке, а на другом придется столкнуться с некоторыми трудностями.

Наибольшее распространение в настоящее время получили языки LAD, STL и FBD.

Большинство фирм-изготовителей ПЛК традиционно имеют собственные фирменные наработки в области инструментального программного обеспечения. Например такие, как «Concept» Schneider Electric, «Step 7» Siemens.

Открытость МЭК-стандартов привела к созданию фирм, занимающихся исключительно инструментами программирования ПЛК.

Наибольшей популярностью в мире пользуется комплекс CoDeSys. CoDeSys разработан фирмой 3S. Это универсальный инструмент программирования контроллеров на языках МЭК, не привязанных к какой-либо аппаратной платформе и удовлетворяющих всем современным требованиям.

Основные особенности:

- полноценная реализация МЭК-языков;

- встроенный эмулятор контроллера позволяет проводить отладку проекта без аппаратных средств. Причем эмулируется не некий абстрактный контроллер, а конкретный ПЛК с учетом аппаратной платформы;

- встроенные элементы визуализации дают возможность создать модель объекта управления и проводить отладку, т.е. дает возможность создавать человеко-машинный интерфейс (HMI);

- очень широкий набор сервисных функций, ускоряющих работу программиста;

- существует русская версия программы и русская документация.

4.1 Системы автоматического управления на базе ПЛК DVP-14SS2

4.1.1 Общие сведения

Рассмотрим вопрос автоматизации линии на базе промышленного контроллера ПЛК DVP-14SS2. Универсальный лабораторный стенд «Изучение систем автоматического управления технологическими процессами на базе Дельта ПЛК DVP-14SS2» предназначен для изучения малых и средних систем автоматического управления технологическими процессами на базе программируемого логического контроллера DVP-14SS2, который позволяет изучить навыки программирования контроллера на языках LD, IL и SFC в программном комплексе WPL Soft, а также моделировать работу систем автоматического управления технологическими процессами [18].

Лабораторный стенд «Изучение систем автоматического управления технологическими процессами на базе Дельта ПЛК DVP-14SS2» выполняется в настольном исполнении, внешний вид которого представлен на рисунке 4.1.

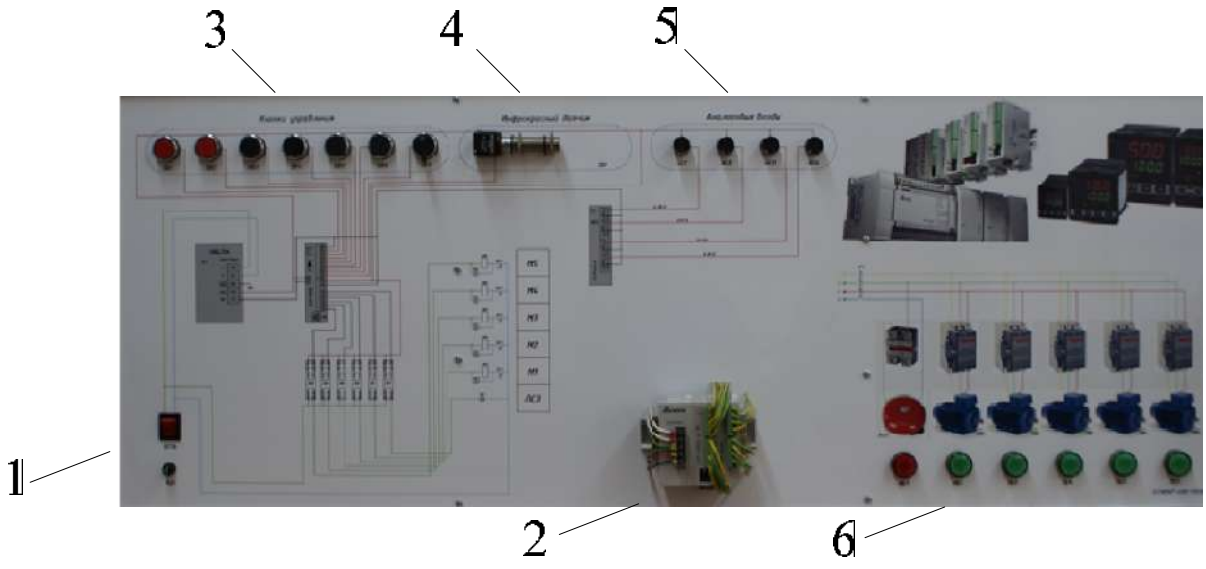


Рисунок 4.1 – Внешний вид лабораторного стенда.

На рисунке 4.1обозначены: 1 – кнопка вкл./выкл. лабораторного стенда, 2 – блок питания совместно с промышленным контроллером DVP-14SS2 и модулем аналогового входа DVP04DA-S, 3 – кнопки управления (дискрет-

ные), 4 − бесконтактный оптический датчик ВБО-М18-76К-5123-СА, 5 – переменные резисторы для имитации аналоговых сигналов.

Данное оборудование позволяет реализовывать системы управления линий производств. Отличительной особенностью представленного лабораторного оборудования является использование программированного контроллера совместно с модулем аналоговых входов, позволяющим расширить область применения по автоматизации технологического процесса. При этом общение с ПЛК реализовывается на базе 3 языков программирования.

Программируемые логические контроллеры Дельта DVP выполнены в полном соответствии со стандартом ГОСТ Р 51840-2001 (IEC 61131-2), что обеспечивает высокую аппаратную надежность.

По электромагнитной совместимости контроллеры соответствуют классу А по ГОСТ Р 51522-99 (МЭК 61326-1-97) и ГОСТ Р 51841-2001, что подтверждено неоднократными испытаниями изделия [13].

Технические характеристики DVP-14SS2:

- дискретных входов 8;
- дискретных выходов 6;
- напряжение питания 24 В (20,4-28,8) постоянного тока;
- пиковый ток 7,5 А;
- предохранитель электропитания 1,85 А/30 В постоянного тока, самовосстанавливающийся предохранитель;
- потребляемая мощность 1,8 Вт;
- защита от неправильной полярности источника постоянного тока;
- сопротивление изоляции 5 Мом;
- встроенный аккумулятор, позволяющий «пережидать» пропадание питания − выполнять программу при пропадании питания и переводить выходные элементы в «безопасное состояние». Время «пережидания» настраивается пользователем при создании проекта;
- встроенные часы реального времени;
- возможность создавать и сохранять архивы на Flash контроллера[15].

Условия эксплуатации:

- расширенный температурный рабочий диапазон окружающего воздуха: от 0°С до +55 °С;
- закрытые взрывобезопасные помещения или шкафы электрооборудования без агрессивных паров и газов;
- верхний предел относительной влажности воздуха − 95 % при 25 °С и более низких температурах без конденсации влаги;
- атмосферное давление от 84 до 106,7 кПа;
- хранение -25 °С… 70 °С

Все дискретные входы контроллера измеряют сигнал 24В.

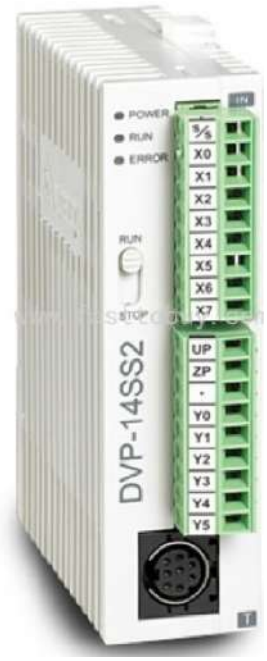


Рисунок 4.2 – Программируемый логический контроллер DVP-14SS2.

Помимо самого ПЛК, лабораторный стенд содержит модуль аналоговых входов (рис. 4.3).

Рисунок 4.3 – Модуль аналогового входа DVP04DA-S.

- Напряжение питания 24 В постоянного тока.
- На каждом модуле 4 канала.
- Диапазон аналоговый +/- 10 VDC +/- 20 мА.
- Разрядность АЦП 14 бит (мин. шаг 1,25 мВ) 13 бит (мин. шаг 5 мкА).
- Время отклика 3 мс на каждый канал.
- Аналоговая и цифровая части между собой изолированы.
- Аналоговые каналы между собой не изолированы.
- Абсолютный диапазон входа +/- 15 VDC +/- 32 мА.
- Доступные протоколы обмена данными по RS485. Modbus RTU/ASCII, скорость обмена 9600 ~ 115200. Формат данных для ASCII: 7 бит данных, четно, 1 стоповый (7, E, 1). Формат данных для RTU: 8 бит данных, четно, 1 стоповый (8, E, 1). Когда модуль подключен по внутренней шине непосредственно к ПЛК, порт RS485 недоступен.
- Непосредственно к ПКЛ по внутренней шине можно подключить до 8 аналоговых модулей. На дискретные входы/выходы это никак не влияет. Нумерация аналоговых модулей будет 0 ~ 7, начиная с самого ближнего к ПКЛ и далее по порядку по мере удаления от ПЛК;
- Работа: 0°C ~ 55°C , 50% ~ 95% относительной влажности, степень за-

грязнения 2.

- Хранение: -25°C ~ 70°C, 5% ~ 95% относительной влажности.

Оценка работоспособности модуля по состоянию индикаторов:

1. При подаче питания должен загореться индикатор POWER, а также на 0,5 сек. индикатор ERROR, после чего оно должен погаснуть.
2. В случае нормального напряжения питания индикатор POWER должен гореть постоянным зеленым светом, а индикатор ERROR не должен светиться. При снижении напряжения питания ниже 19,5 VDC индикатор ERROR начнет непрерывно мигать до тех пор, пока не восстановится должный уровень напряжения питания.
3. Если модуль подключен по внутренней шине к контроллеру, то при переводе ПЛК в режим РАБОТА на модуле должен загореться индикатор A/D.
4. При получении первой команды по RS485 на модуле должен загореться индикатор A/D.
5. Если какой-либо из рабочих параметров АЦП выйдет за допустимый диапазон, то начнет мигать индикатор ERROR.

ПЛК и модуль аналоговых входов запитываются с помощью блока питания DVPPS01 (рис. 4.4).

Мощные источники питания Delta серии CliQ (до 480 Вт) выпускаются на номинальное выходное напряжение 24 В постоянного тока, предназначены для эксплуатации при температуре воздуха от -20 °C до +70 °C.

- Высокий уровень защиты: защита от перегрузки, защита от перенапряжения, защита от короткого замыкания, тепловая защита.
- Широкий диапазон входного напряжения 85−264В переменного тока или 120−375 В постоянного тока.
- Соответствие RoHS.
- Возможность ручной корректировки выходного напряжения.

Рисунок 4.4 – Блок питания DVPPS01.

- Габаритные размеры: 90x36,5x60 мм.
  Интерфейсы и протоколы
- Базовый коммуникационный протокол MODBUS.
- Все ПЛК в сети должны иметь одинаковый коммуникационный формат (D1120) режим ASCII или RTU.
- Один ведущий ПЛК (серии SA/SX) может иметь до 16 ведомых ПЛК, а один ведущий ПЛК (серия EH) может иметь до 32 ведомых. Для работы с более чем 16 ведомыми ПЛК в ведущем ПЛК должно быть включено специальное реле M1353=1.
- Адреса (ID) в сети не должны повторяться, каждый ведомый ПЛК должен иметь уникальный ID (1 - 32).
- Для связи одного ведущего ПЛК с одним ведомым можно использовать интерфейсы: RS-232, RS-485, RS-422.
- Для связи одного ведущего ПЛК с нескольким ведомыми можно использовать только интерфейс RS-485.

4.1.2 Описание программного продукта WPL Soft

Для программирования всех контроллеров DVP можно использовать портативный программатор DVPHPP02 или компьютер с программным обес-

печением WPL Soft.

WPL Soft позволяет программировать, редактировать и отлаживать программу всех контролеров DVP, а также конфигурировать модули ЦПУ и периферийное оборудование.

Основные характеристики WPL Soft:

1. Работает под Windows, имеет интерфейс на английском языке и развитую систему помощи.

2. Позволяет писать комментарии на русском языке (комментарии к устройствам, строкам и блокам в режиме LAD).

3. Поддерживает работу с проектами и использует иерархический метод отображения внутренней системной информации ПЛК (включая системные параметры подключенных периферийных устройств).

4. Автоматически определяет параметры и скорость коммуникации подключенного ПЛК.

5. Устанавливает значения календаря и часов реального времени.

6. Поддерживает два варианта соединения с ПЛК: прямое соединение и через модем.

7. Возможна отладка программы в режиме ONLINE с отображением текущего состояния всех внутренних устройств.

8. Для программирования всех типов центральных процессоров могут быть использованы три языка программирования: LAD (диаграммы релейно-контактной логики), IL (список инструкций) и SFC (последовательные функциональные схемы). Редактор позволяет выполнять конвертацию программы с одного языка на другой и обратно. Интерфейс редактора способен отображать программу одновременно во всех трех языковых режимах.

9. Возможность редактирования значений всех типов внутренних устройств ПЛК (включая M, S, T, C, D и файловые регистры).

10. Много полезных функций для режима on-line, таких как:

− удобная установка протокола коммуникации, который будет сохранен в регистре D1120;

− LRC/CRC генератор для расчета контрольных сумм используемых в режиме MODBUS;

− чтение внутренней системной информации ПЛК;

− мастер, помогающий написать сложные инструкции: ПИД-регулятор, быстродействующий счетчик, импульсный выход и др.

Рисунок 4.5 – Языки программирования.

После установки программы установщик создаст на рабочем столе иконку быстрого запуска редактора WPL Soft.

Рисунок 4.6 – Иконка быстрого запуска WPL Soft.

Двойным щелчком мыши по иконке WPL Soft запустите редактор WPL Soft. Появится окно редактора.

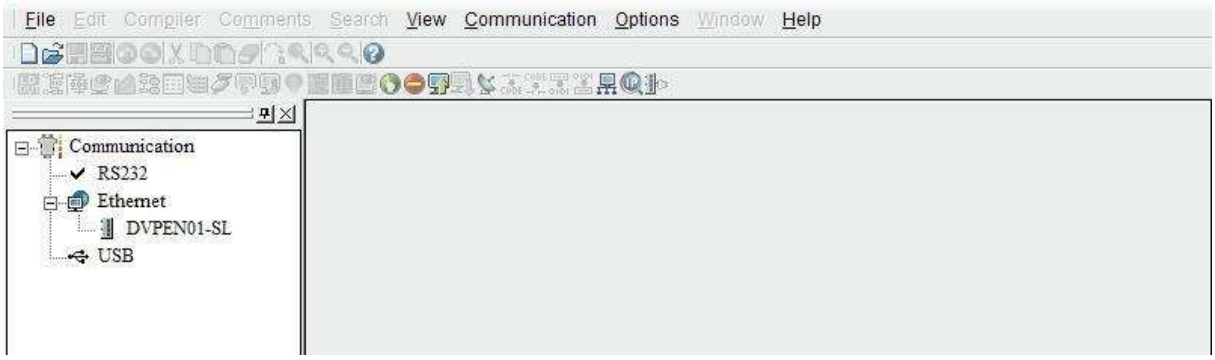


Рисунок 4.7 – Окно редактора WPL Soft.

В верхней части окна расположено главное меню приложения, чуть ниже – панель кнопок быстрого запуска. Серым отмечены неактивные кнопки и пункты меню. Они будут активированы только после создания или открытия проекта.

Рабочая область окна разделена на две части. Слева расположено окно коммуникации, оно нам пока не понадобится. Справа – пустая серая область, в ней после создания проекта появятся окна редактирования логики проекта.

Чтобы создать новый проект, выберите пункт меню File → New ( ). Появится окно конфигурирования проекта (рисунок 4.8).

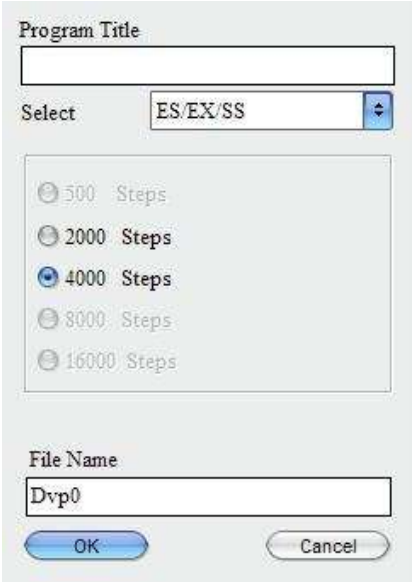


Рисунок 4.8 – Окно конфигурирования проекта.

В верхней части окна Вы можете задать название проекта (Program Title). Чуть ниже расположена конфигурация PLC (оставьте настройки по

умолчанию). В самом низу– область ввода имени файла проекта, ее стоит изменить, задав более-менее осмысленное значение (например, Demo).

Нажмите кнопку ОК. Появятся три окна, каждое из которых по-своему отражает логику созданного проекта:

Instruction List Mode – отображает логику проекта в виде последовательности инструкций контроллера. Если Вы привыкли программировать логику в текстовом режиме, для вас это окно может оказаться полезным. Всем остальным заглядывать в него понадобится крайне редко.

Ladder Diagram Mode – отображает логику проекта в виде лестничных диаграмм. Это окно мы рекомендуем использовать в качестве основного окна редактирования. Примеры, рассмотренные в данном руководстве, будут использовать именно в это окно.

SFC Mode – отображает логику проекта в виде графов переходов. Если для Вас этот метод отображения более привычен, используйте его. Однако в данном руководстве редактирование в этом режиме не рассматривается.

Обратите внимание, что нижняя часть панели кнопок быстрого запуска изменяется в зависимости от выбранного Вами окна отображения. Эта часть панели содержит кнопки редактирования логики проекта, они будут рассмотрены нами на соответствующих этапах создания логики демонстрационного проекта.

Любое выбранное отображение логики может быть преобразовано в набор инструкций (Instruction List Mode). В свою очередь набор инструкций может быть преобразован в любой другой тип отображения. Все эти операции осуществляются вызовом команд меню Compiler (в скобках показана кнопка быстрого запуска команды):

Ladder =>Instruction – преобразует лестничные диаграммы в набор инструкций.

Instruction =>Ladder – преобразует набор инструкций в лестничные диаграммы.

SFC =>Instruction – преобразует граф переходов в набор инструкций

Instruction => SFC – преобразует набор инструкций в граф переходов.

Чтобы сохранить созданный проект, выберете пункт меню File → Save ().

Появится стандартное диалоговое окно сохранения файла системы Windows. Задайте имя файла, которое считаете нужным, или оставьте предлагаемое редактором по умолчанию и нажмите кнопку «Сохранить». Проект будет сохранен в указанном месте.

Чтобы закрыть проект, выберете пункт меню File → Close Project. Чтобы выйти из приложения, выберете пункт меню File → Exit или нажмите «Alt+X» с клавиатуры.

Для открытия созданного ранее проекта выберете пункт меню File → Open (). Появится стандартное диалоговое окно открытия файла системы Windows. В этом окне выберете имя файла проекта, сохраненного на одном из предыдущих этапов, и нажмите кнопку «открыть».

Как и в случае создания нового проекта, после открытия сохраненного проекта неактивные кнопки быстрого запуска и пункты меню станут активными, а в правой части рабочей области появятся окна редактирования логики проекта.

Так как ранее в этом проекте мы не создавали никакой логики, все окна, кроме окна редактирования набора инструкций, будут пустыми. По умолчанию вся память PLC заполнена командами NOP (no operation) – нет операции.

Начнем с простого примера. Пусть входы 0 и 1 должны зажигать выход 0, а вход 2 управлять работой схемы: если вход 2 установлен, выход 0 всегда выключен.

Реализуем эту логику на языке лестничных диаграмм. Для этого активируйте окно редактирования лестничных диаграмм, выбрав пункт меню View→ Ladder Diagram. Все лестничные диаграммы «растут» слева на право. Левая граница лестничной диаграммы отмечена вертикальной чертой.

Окно редактирования лестничных диаграмм разбито на клетки, в каждой из таких клеток может быть размещено не более одного элемента лестничной диаграммы. Синий квадрат представляет собой область ввода. Перемещать область ввода можно с клавиатуры клавишами «вверх», «вниз», «влево», «вправо» или мышкой, щелкая по интересующей Вас части лестничной диаграммы.

Установите область ввода в левый верхний угол лестничной диаграммы, как показано на рисунке 4.9.

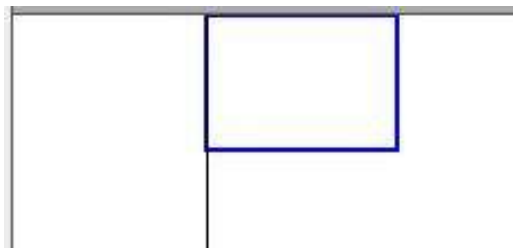


Рисунок 4.9 – Внешний вид исходное положение области ввода в программе.

Выбор команд редактирования осуществляется нажатием на соответствующие кнопки панели быстрого запуска. Кроме того, все команды редактирования имеют горячие клавиши (они изображены на нижней части кнопок редактирования).

Входы и маркеры на лестничной диаграмме представлены в виде контактов различного типа. Всего существует четыре типа контактов, каждому из которых соответствует кнопка панели быстрого запуска:

- Нормально разомкнутый контакт (

).
- Нормально замкнутый контакт (

).
- Контакт, замыкаемый фронтом сигнала (

).
- Контакт, замыкаемый спадом сигнала (

).

Нажмите на кнопку  панели быстрого запуска или кнопку «F1» клавиатуры. Появится окно редактирования устройства (рисунок 4.10).

Рисунок 4.10 – Окно редактирования устройства.

В левом верхнем углу окна указан тип редактируемого устройства − нормально разомкнутый контакт (Constantly opened contact).

В окне редактирования устройства необходимо указать тип (имя) устройства (Device Name) и номер устройства (Device Number).

В редакторе WPL Soft существуют следующие основные типы (имена) устройств (операндов):

- X – входы. Входные реле. Определяют состояние внешних битовых устройств, подключенных к входным клеммам ПЛК. Могут принимать одно из двух состояний: 0 или 1. Адресация ведется в восьмеричной системе: X0, X1, … X7, X10, X11, …
- Y – выходы. Выходные реле. Определяют состояние выходных клемм ПЛК, к которым подключается нагрузка. В программе могут быть как контактами, так и катушками, и принимать одно из двух состояний: 0 или 1. Адресация ведется в восьмеричной системе: Y0, Y1, … Y7, Y10, Y11, …
- M – маркер. Внутренние (вспомогательные) реле. Память для двоичных промежуточных результатов. В программе могут быть как контактами, так и катушками, и принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: M0, M1 M7, M8, M9, …

- S – маркер. Управляющие шаговые реле. Используются для программирования последовательного управляющего процесса. Могут принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: S0, S1, … , S1023.
- T – таймер. Реле времени. В программе могут использоваться для хранения текущего значения таймера и иметь 16-битный формат, а также могут быть контактами и принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: T0, T1, …, T255.
- C – счетчик. Используются для реализации счета. В программе могут использоваться для хранения текущего значения счетчика и иметь 16- или 32-битный формат, а также могут быть контактами и принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: C0, C1, …, C255.

Диапазон доступных номеров устройств указан красными буквами рядом с надписью Range. При этом входы и выходы нумеруются в восьмеричной системе, а маркеры, таймеры и счетчики – в десятичной.

Поле Comment позволяет задать комментарий к устройству. Комментарии сделают Вашу лестничную диаграмму более понятной и простой для восприятия.

Итак, задайте имя устройства X, номер – 0, в поле комментария (Comment) введите «Вход 0». После этого нажмите кнопку ОК.

Сместите поле ввода вниз и проделайте все тоже самое для устройства X1. В поле комментария введите «Вход 1». В результате у Вас должно получиться:

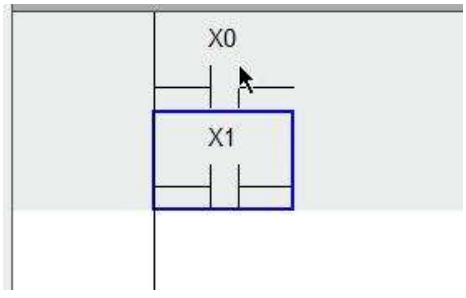


Рисунок 4.11 – Входы 0 и 1 на лестничной диаграмме.

Так как оба входа 0 и 1 должны зажигать выход 0, нам необходимо объединить X0 с X1. Поместите поле ввода справа от X0 и нажмите на кнопку «F9», появится вертикальная черта, соединяющая X0 с X1.

Выходы на лестничных диаграммах отображаются в виде катушек (Coils). В отличие от контактов, катушка на лестничной диаграмме всегда находится в крайнем правом положении. Катушка устанавливает состояние соответствующего ей устройства как результат работы участка диаграммы (последовательности контактов) слева от нее.

Помимо выходов (Y), катушками могут быть маркеры (M, S), это позволяет сохранять результаты работы участков диаграммы и использовать их на следующих циклах работы PLC.

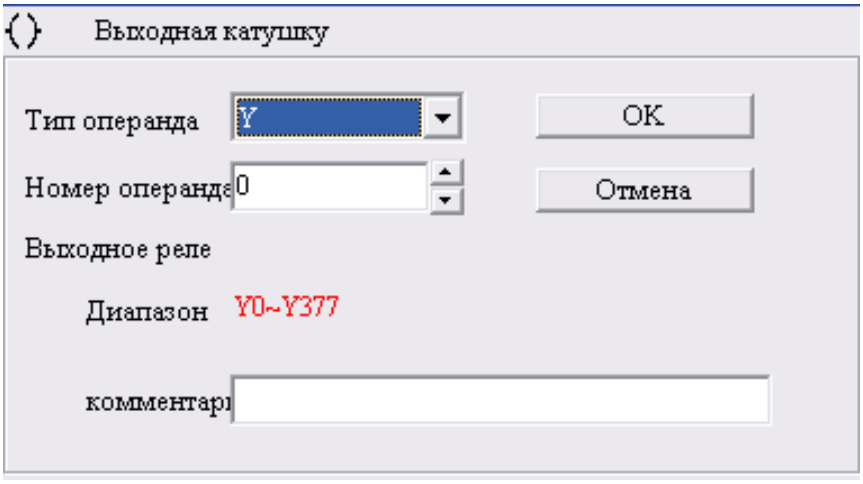


Рисунок 4.12 – Входы 0, 1 и выход 0 на лестничной диаграмме.

Добавьте на диаграмму выход 0. Для этого нажмите на кнопку «F7». Появится окно редактирования устройства (рисунок 4.12). В левом верхнем углу, как обычно, указан тип редактируемого устройства – выходная катушка (Output Coil). Задайте имя устройства – Y, номер устройства – 0, комментарий – «Выход 0» и нажмите OK.

Катушка появится не в выбранной Вами области ввода, как Вы можете ожидать, а в крайнем правом положении на лестничной диаграмме, где она и должна находиться по правилам. В результате у вас должно получиться:

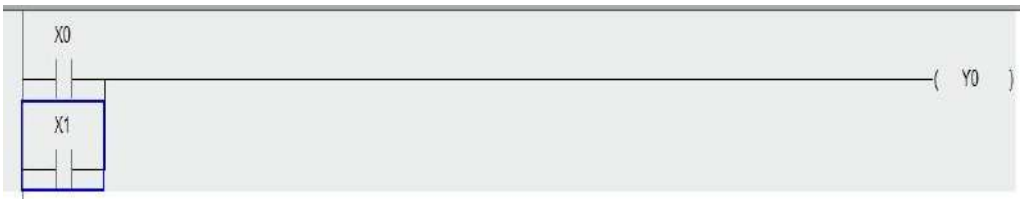


Рисунок 4.13 – Входы 0, 1 и выход 0 на лестничной диаграмме.

Осталось добавить вход 2. В соответствии с формулой (1) вход 2 должен быть нормально замкнутым реле. Поместите область ввода справа от X0 и нажмите на кнопку «F2». Вновь появится окно редактирования устройства (рисунок 4.10), типом устройства на этот раз будет нормально замкнутый контакт (constantly closed contact). Введите имя и номер устройства (X2) и задайте комментарий («Вход 2»). В итоге у Вас должна получиться следующая картинка:

Рисунок 4.14 – Входы 1..3, выход 0 на лестничной диаграмме.

Теперь перейдем к устройствам чуть более сложным, чем катушки и контакты – к таймерам. Таймеры служат для введения задержки распространения сигналов. Длительность задержки устанавливается в периодах работы PLC (10 мс для ЛИР-986). У каждого таймера есть вход и выход. Вход таймера управляет работой таймера, а выход отражает его текущее состояние.

При наличии положительного сигнала на входе таймера таймер начинает работать. Как только сигнал пропадает, таймер сбрасывается. По достижению таймером установленного значения задержки включается выход таймера.

В лестничных диаграммах работой таймеров управляют специальные инструкции. Инструкции могут управлять не только таймерами, во многом они похожи на катушки: они так же располагаются в крайнем правом поло-

жении на диаграмме и используют в качестве условия активации результат работы последовательности контактов слева. Но операции, выполняемые инструкциями, значительно более сложные, чем простое включение/выключение некоторого устройства, к тому же инструкции могут принимать конфигурирующие их работу параметры.

Добавим задержку на установку выхода 0. Для этого поместите область ввода справа от контакта X2 и нажмите на кнопку «F6». Появится окно редактирования инструкций.

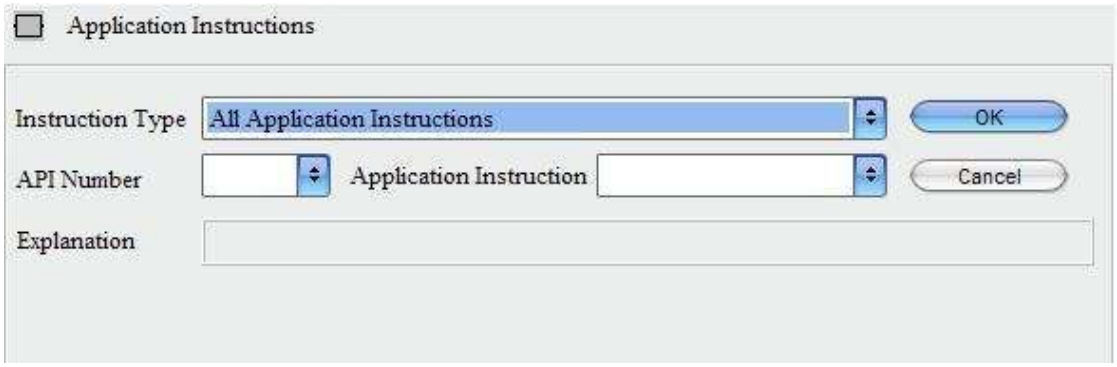


Рисунок 4.15 – Окно редактирования инструкций.

В списке Instruction Type (Тип Инструкции) выберите Timer & Counter (Таймер и Счетчик). В списке Application Instruction выберите TMR. Окно приобретет вид, представленный на рисунке 4.16, появятся параметры S и D, а также Reference (справка) внизу окна.

Справка состоит из двух окон. В верхнем окне звездочками отмечены допустимые для каждого из параметров типы (имена) устройств. В нижнем окне размещена справочная информация по каждому из параметров: S – Timer number (номер таймера), D – Timer Setting Value (значение задержки таймера).

Выберете из списка параметра S значение T, затем в появившемся числовом поле ввода задайте номер таймера – 0. Затем аналогично для параметра D – тип устройства «K» (константа), значение 100 (задержка на 1 с) и нажмите OK.

Катушка Y0 заменится символом таймера 0. В этом символе указано

слева на право: имя инструкции, конфигурирующей таймер (TMR), имя таймера (T0) и величина задержки таймера (K100 – константа 100).

Теперь, чтобы таймер 0 устанавливал выход 0, нужно добавить на диаграмму выход таймера и выход 0. Выход таймера добавьте в виде нормально разомкнутого контакта, а выход 0, как и прежде, в виде катушки. Обратите внимание на то, что редактор WPL Soft запомнил комментарий к устройству Y0 («Выход 0»). Результат представлен на рисунке 4.16.

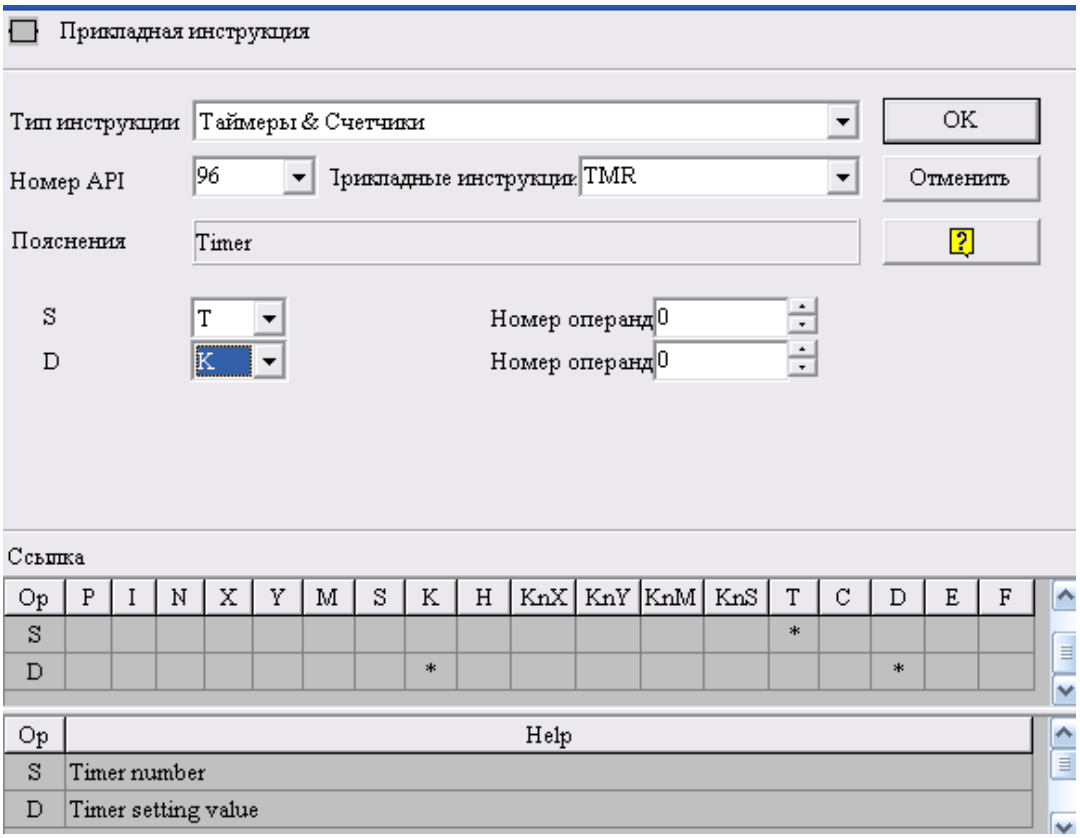


Рисунок 4.16 – Окно редактирования инструкций.

Результат добавления таймера времен в программу отображен на рисунке 4.17. Название таймера T0 работает с задержкой на включение или отключение своих контактов. Время срабатывания определяется величиной K100. В конкретном случае замыкающий контакт таймера T0 сработает через задержку времени, равной 10 с.

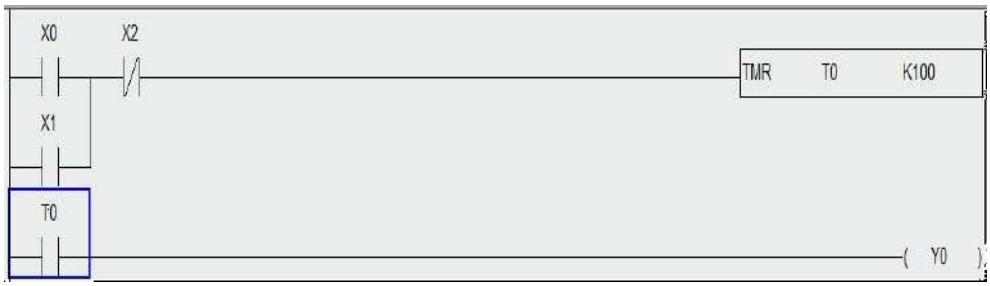


Рисунок 4.17 – Результат добавления таймера T0.

Счетчики предназначены для подсчета импульсов. Так же, как и таймеры, счетчики имеют собственный вход и выход. Счет импульсов осуществляется по входу счетчика. Как только значение счетчика достигает заданного значения, счетчик устанавливает свой выход в активное состояние.

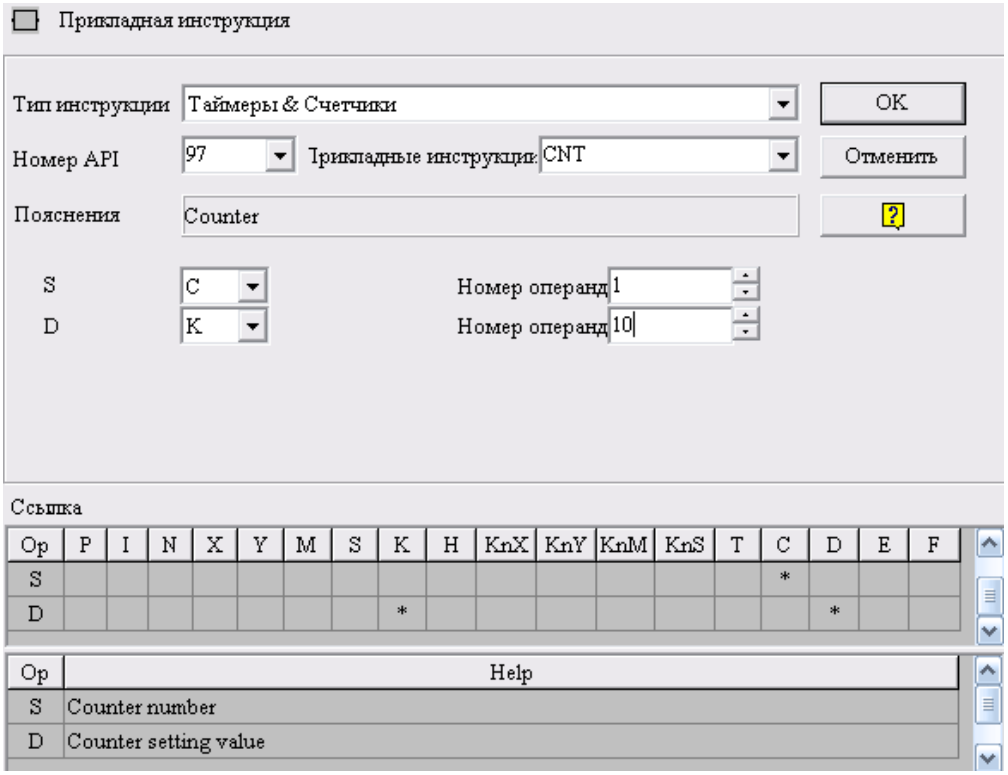


Рисунок 4.18 – Окно редактирования инструкций для счетчика.

Добавим счетчик импульсов по входам X0, X1. Для этого установите область ввода справа от X1 и нажмите на кнопку «F6». В появившемся окне редактирования инструкций (рисунок 4.18) выберите Instruction Type (Тип Инструкции) –Timer & Counter (Таймер и Счетчик), Application Instruction – CNT. Окно изменится подобно тому, как это было при добавлении таймера.

Задайте следующие параметры: S – C1 (в ЛИР-986 таймеры и счетчики − это одни и те же устройства, поэтому следите, чтобы номера устройств C и T в Ваших проектах не совпадали), D – K10.

Выход счетчика 1 отправим также на выход 0. Для этого добавьте нормально разомкнутый контакт C1 и катушку Y0. Задайте комментарий к контакту C1 – «Счетчик импульсов».

Сброс счетчика осуществляется инструкцией RST. Установим сброс счетчика 1 по установке выхода таймера 0. Для этого установите область ввода справа от контакта T0 и нажмите на кнопку «F6». В появившемся окне редактирования инструкций (рисунок 4.19) выберите Instruction Type (Тип Инструкции) - Output (Выход), Application Instruction – RST. Единственный появившийся параметр S установите равным C1 и нажмите OK.

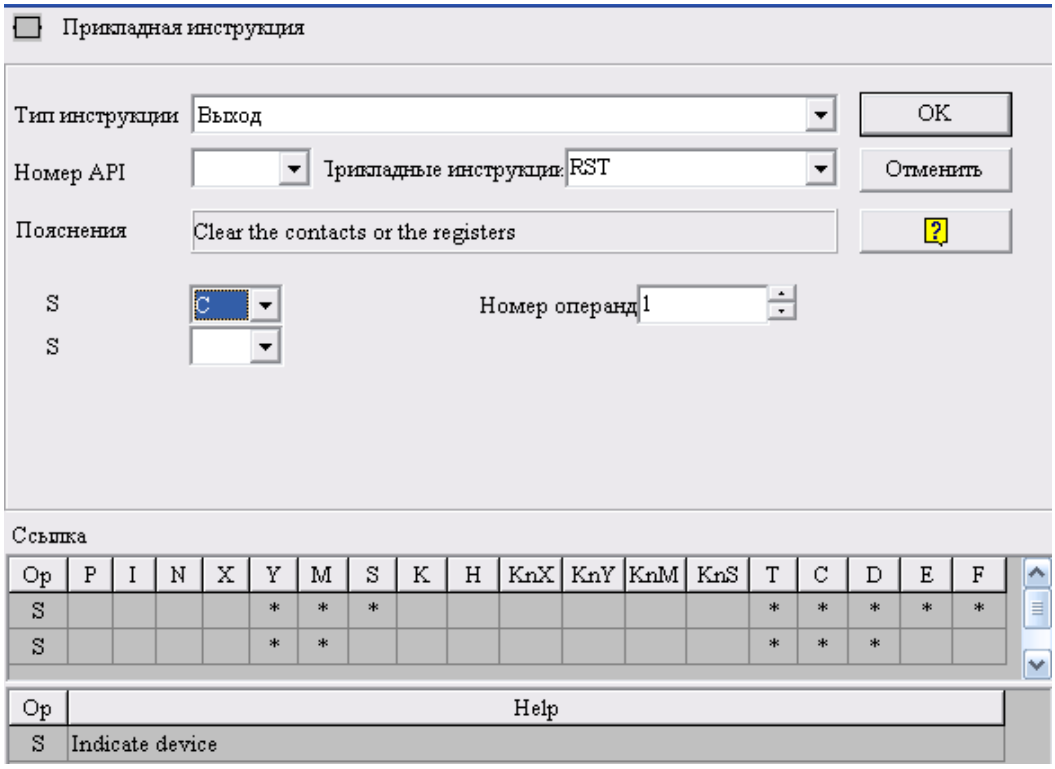


Рисунок 4.19 – Окно редактирования инструкций для вывода.

Теперь можно добавить осмысленный комментарий к контакту T0. Для этого щелкните дважды мышью по контакту T0 или, установив на него область ввода, нажмите клавишу «Enter» с клавиатуры. Появится окно Input Instruction (ввод инструкции), в нем нажмите на кнопку «…», чтобы вызвать

окно редактирования устройства. Введите комментарий «Сброс счетчика импульсов» и нажмете ОК. Результат добавления счетчика представлен на рисунке 4.20.

Рисунок 4.20 – Результат добавления счетчика 1.

Чтобы перевести лестничную диаграмму в набор инструкций, необходимо скомпилировать проект. Это обязательная операция, без нее вся созданная Вами логика – это не более чем картинка.

За компиляцию проекта отвечает раздел меню Compiler (Компилятор). Он содержит различные варианты преобразования отображений логики в набор инструкций и обратно. Нам понадобится пункт меню Ladder => Instruction. Если у Вас все еще активно окно отображения набора инструкций, то этот пункт меню будет неактивным (серым цветом). Перейдете к окну редактирования лестничных диаграмм и вызовите требующийся пункт меню.

Если в процессе компиляции появится ошибка, информация об этом отразиться в возникшем снизу окне (рисунок 4.21).

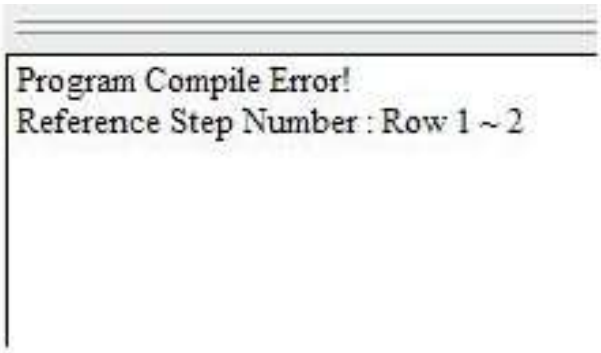


Рисунок 4.21 – Ошибка компиляции.

4.1.3 Пример выполнения схемы автоматизации линии предварительной обработки зерна на базе ПЛК DVP-14SS2

Реализовывать схему автоматизации линии (см. п. 2.4) будем на лабораторном оборудовании, внешний вид которого изображен на рисунке 4.1.

Таблица 4.1− Входные дискретные сигналы.

№ п/п	Наименование сигнала	Имя источника	Разряд-ряд-ность	Сигнал
1	2	3	4	5
1	Кнопка«Общий стоп»	SB1	bit	DI 24 V DC
2	Кнопка«Пуск»	SB2	bit	DI 24 V DC
3	Кнопка «Рабочий стоп»	SB3	bit	DI 24 V DC
4	Датчик уровня	SL1	bit	DI 24 V DC

Таблица 4.2− Перечень выходных сигналов и данных.

№ п/п	Наименование сигнала	Имя источника	Диапазон изменения	Пользовательская информация
1	2	3	4	5
1	Включение звонка	km1	1 Bit	DO 1
2	Включение скребкового транспортера	km2	1 Bit	DO 2
3	Включение молотковой дробилки	km3	1 Bit	DO 3
4	Включение ковшовой нории	km4	1 Bit	DO 4

При написании программы для конкретного технологического процесса воспользуемся типовыми блоками смех автоматизации.

Нереверсивная схема управления электродвигателем реализована в программе WPL Soft на языке LD. На рисунке 4.22 представлена графически схема, на которой X1 − кнопка «пуск» с замыкающим контактом; X0 – кнопка «стоп» с размыкающим контактом; катушка выходного реле (магнитного пускателя) Y0, являющегося механизмом и шунтирующим замыкающим

контактом Y0.

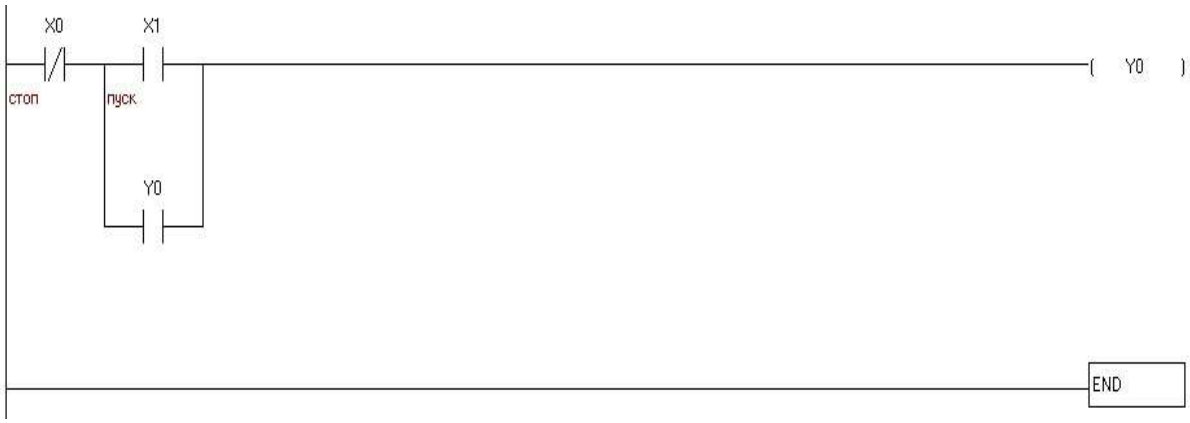


Рисунок 4.22 – Нереверсивная схема управления электродвигателем на языке LD.

Для **реверсивного управления** электродвигателей необходимо наличие двух магнитных пускателей. В нашем случае, применительно к программе управления, необходимо наличие двух выходных реле.

Разработанная схема управления позволяет использовать ее для управления механизмами, обладающими явно выраженными инерционными свойствами. Для этого реверс необходимо производить только с использованием общей кнопки «стоп», имитирующей размыкающим контактом X0. На рисунке 4.23 представлена реверсивная схема управления электродвигателем.

При нажатии на кнопку пуск «вперед» замыкается контакт X1, который подает сигнал на вход катушки Y0 выходного реле. При этом контакты выходного реле Y0 одновременно шунтируют кнопку пуска «вперед» и размыкают цепь с катушкой выходного реле Y1. Благодаря этому избегается одновременное включение двух выходных реле, а в частности, магнитных пускателей электродвигателя. При нажатии на кнопку пуск «назад», замыкается контакт X2, который подает сигнал на вход катушки Y1 выходного реле. Происходит изменение состояния работы выходных реле, что приводит к реверсу электродвигателя.

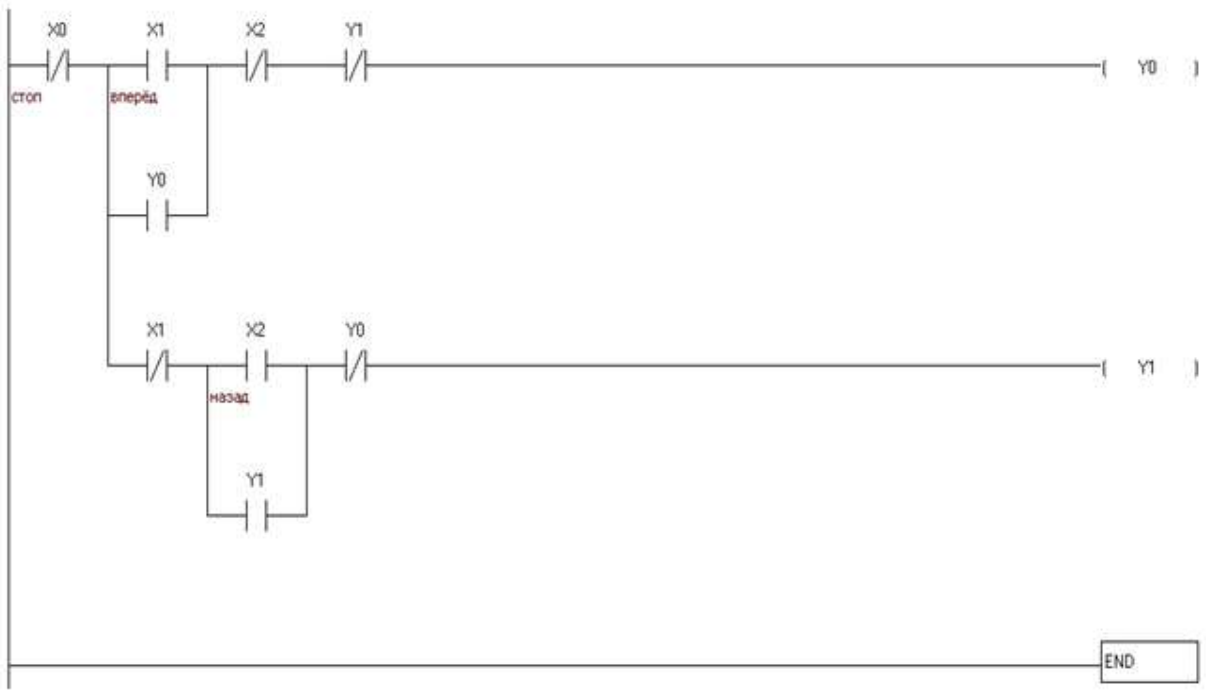


Рисунок 4.23 – Реверсивная схема управления электродвигателем на языке CFC.

Для дополнительной деблокировки работы выходных реле используют размыкающие контакты кнопок X1 и X2.

Пускосигнальное звено предназначено для предупреждения персонала о запуске механизма. Позволяет включить электродвигатель с задержкой по времени, при этом звучит звуковая сигнализация. Пускосигнальное звено реализовано в программе WPL Soft на языке LD.

На рисунке 4.24, помимо уже известных нам кнопок управления, появились новые элементы: M0 – промежуточное реле, служащее в одном случае для шунтирования кнопки пуск, в другом − для правильной последовательности включения механизмов в программе; TMR – реле времени (таймер с задержкой включения); T0 – контакты таймера времени.

При нажатии на кнопку пуск X1 включается реле M0, контакты которого замыкаются, тем самым происходит шунтирование кнопки пуск и подачи питания на цепи управления. В цепи 0003 контакт таймера времени T0 замкнут, что приводит к работе выходного реле «звонок». Контакт таймера времени T0 пропустит через себя сигнал с задержкой по времени, равной 5с, после которой замыкается второй контакт таймера времени T0. Одновременно

с этим контактом происходит отключение выходного реле «звонок» и включение выходного реле «механизм». Для остановки механизма используют общую кнопку стоп X0.

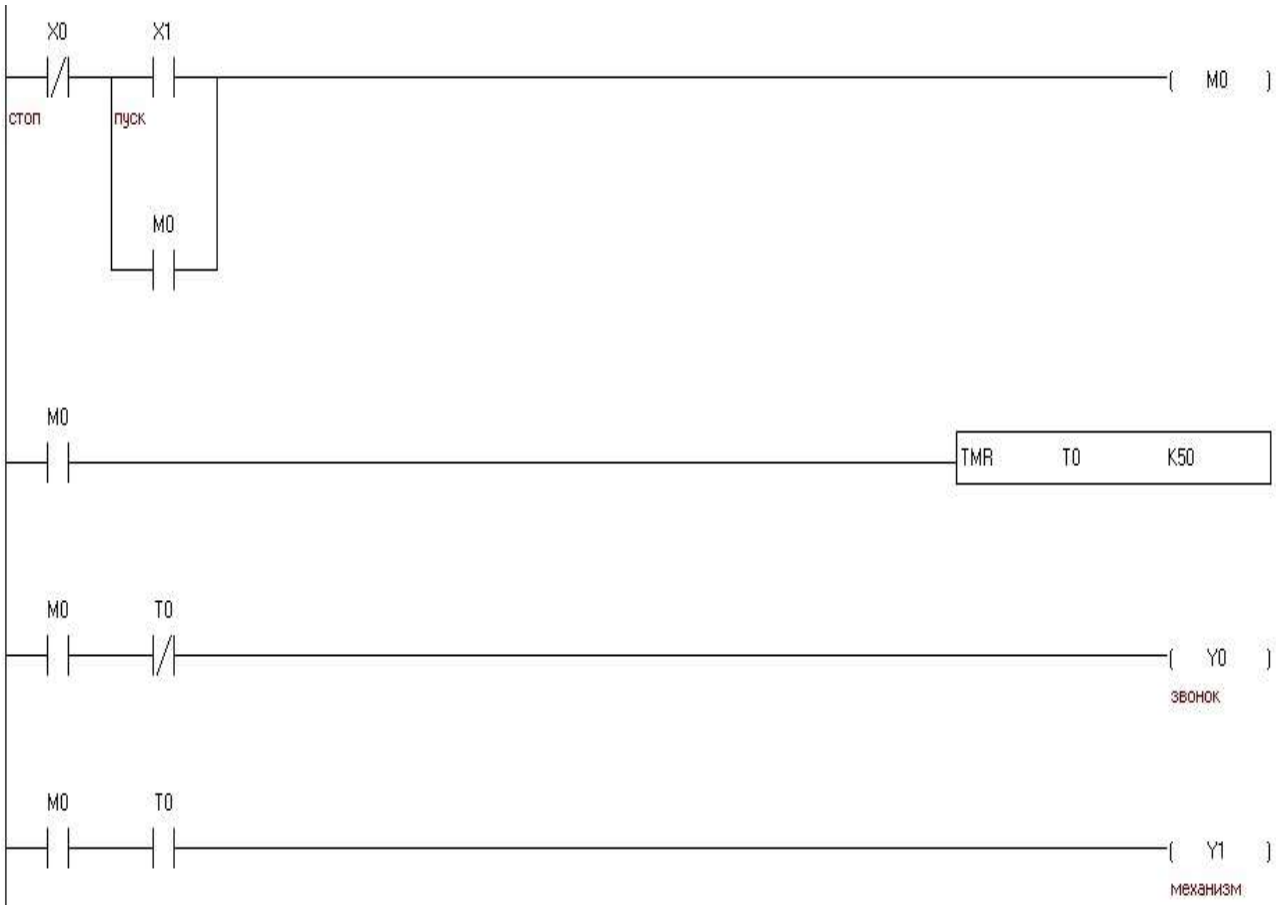


Рисунок 4.24 – Пускосигнальное звено на языке LD.

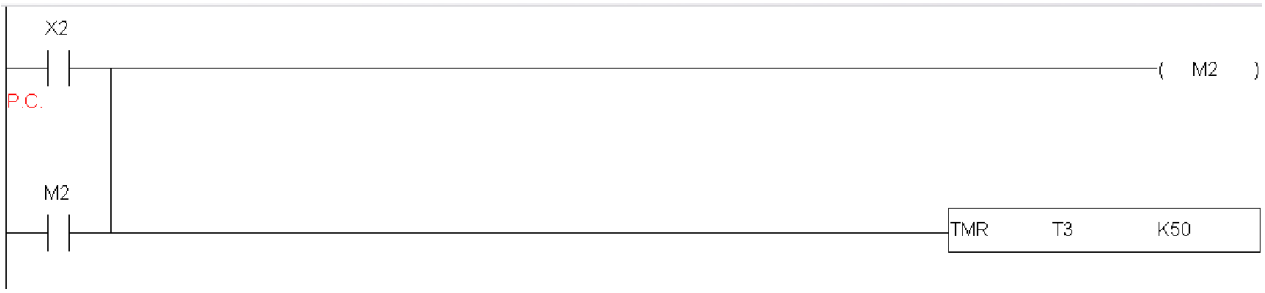


Рисунок 4.25 –Звено «Рабочий стоп» на языке LD.

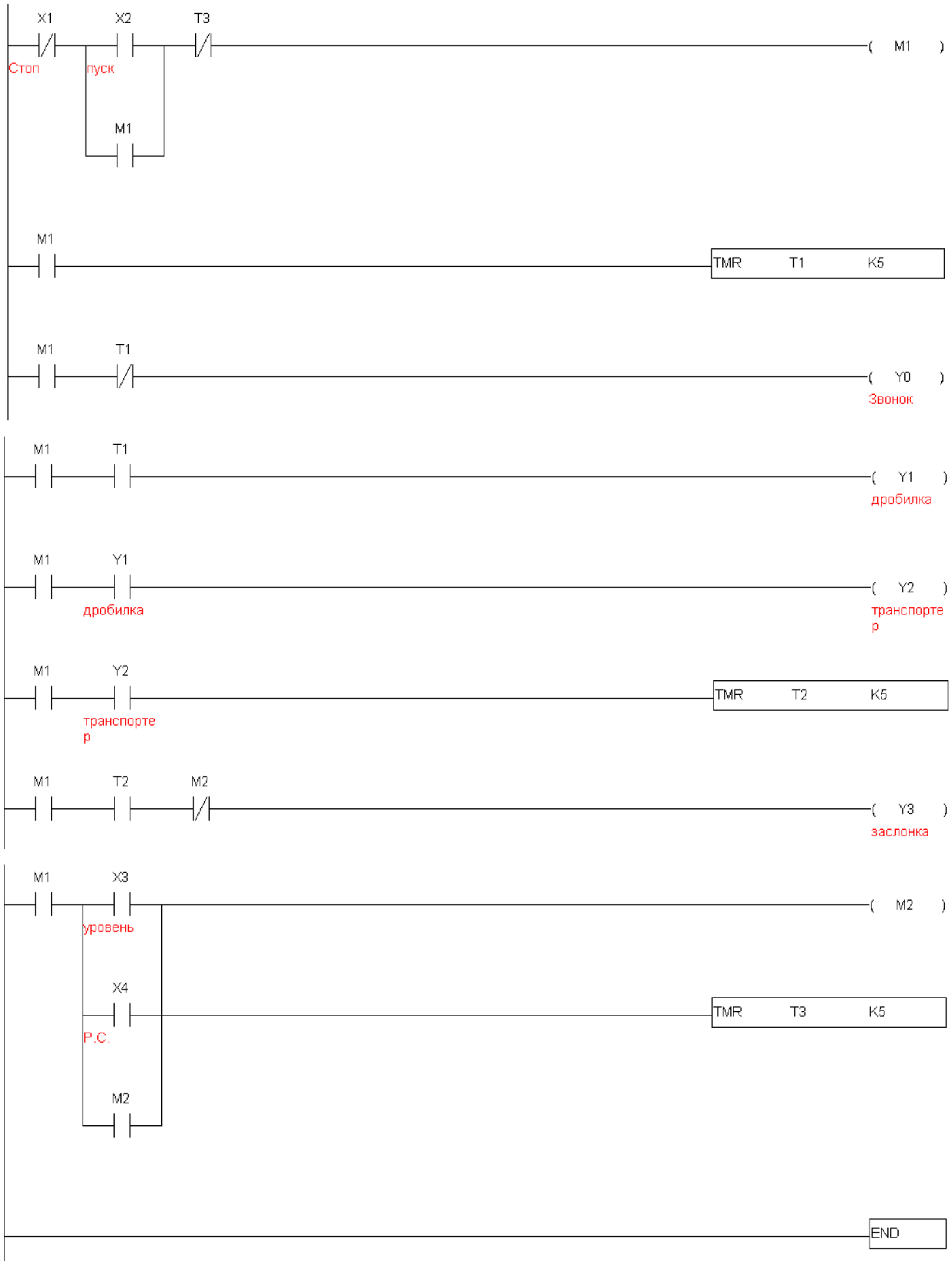


Рисунок 4.26 – Алгоритм управления линии предварительной обработки зерна на языке LD.

Звено «Рабочий стоп» предназначено для правильного отключения механизмов линии. Позволяет отключить механизмы линии с полной их очисткой от продукта. Звено «Рабочий стоп»реализовано в программе WPL Soft на языке LD (рис. 4.25).

При нажатии на кнопку пуск X2 включается промежуточное реле M2, контакт которого замыкается, тем самым происходит шунтирование кнопки пуск X2 и подача питания на цепь управления с блоком таймера времени TRM E3.

При этом происходит размыкание контакта M2, стоящего в цепи с реле головного механизма (на рисунке 4.25 не показан), и включение таймера, контакт T3 которого, с выдержкой по времени, произведет отключение всей линии одновременно либо ступенчато.

На основании разработанных типовых схем составим программу управления линии предварительной обработки зерна, внешний вид которой изображен на рисунке 4.26. Разработанный алгоритм управления реализуется в программном продукте WPL Soft на базе языка программирования LD.

Как видим, в написании программы управления линии предварительной обработки зерна использовались типовые блоки, из которых, как в конструкторе, собирается алгоритм управления. Каждый участок линии начинается с замыкающего контакта M1. Это сделано для того, чтобы питание на цепи подавалось только после включения кнопки пуск X2.

4.2 Системы автоматического управления на базе ОВЕН ПЛК-160

4.2.1 Общие сведения

Программируемый логический контроллер ОВЕН ПЛК160 (рис. 4.27) предназначен для [20]:

- измерения аналоговых сигналов тока или напряжения и преобразования их к выбранной пользователем физической величине;

- измерения дискретных входных сигналов;
- управления дискретными (релейными) выходами;
- управления аналоговыми выходами;
- приема и передачи данных по интерфейсам RS-485, RS-232, Ethernet;
- выполнения пользовательской программы по анализу результатов измерения;
- дискретных и аналоговых входов, управления дискретными входами и выходами, передачи и приема данных по интерфейсам RS-485, RS-232, Ethernet.

Рисунок 4.27 – Внешний вид программируемого логического контроллера ПЛК160-24.А-М.

Логика работы контроллера определяется потребителем в процессе программирования контроллера. Программирование осуществляется с помощью программного обеспечения CoDeSys 2.3 (версии 2.3.9.9). При этом поддерживаются все языки программирования, указанные в МЭК 61131-3.

Контроллер может быть использован как:

- специализированное устройство управления выделенным локализованным объектом;
- устройство мониторинга локализованного объекта в составе комплекс-

ной информационной сети;

- специализированное устройство управления и мониторинга группой локализованных объектов в составе комплексной информационной сети.

Контроллеры выполнены в полном соответствии со стандартом ГОСТ Р 51840-2001 (IEC 61131-2), что обеспечивает высокую аппаратную надежность. По электромагнитной совместимости контроллеры соответствуют классу А по ГОСТ Р 51522-99 (МЭК 61326-1-97) и ГОСТ Р 51841-2001, что подтверждено неоднократными испытаниями изделия.

Технические характеристики ОВЕН ПЛК160-24.А-М:

В контроллере изначально заложены мощные вычислительные ресурсы при отсутствии операционной системы:

- высокопроизводительный процессор RISC архитектуры ARM9 с частотой 180МГц компании Atmel;
- большой объем оперативной памяти − 8МБ;
- большой объем постоянной памяти −Flash память, 4МБ;
- объем энергонезависимой памяти для хранения значений переменных − до 16КБ;
- время цикла по умолчанию составляет 1мс при 50 логических операциях, при отсутствии сетевого обмена;
- широкие возможности самодиагностики контроллера;
- встроенный аккумулятор, позволяющий «пережидать» пропадание питания − выполнять программу при пропадании питания и переводить выходные элементы в «безопасное состояние». Время «пережидания» настраивается пользователем при создании проекта;
- встроенные часы реального времени;
- возможность создавать и сохранять архивы на Flash контроллера.

Условия эксплуатации:

- расширенный температурный рабочий диапазон окружающего воздуха: от-10 °С до +50 °С;

- закрытые взрывобезопасные помещения или шкафы электрооборудования без агрессивных паров и газов;
- верхний предел относительной влажности воздуха − 80 % при 25 °С и более низких температурах без конденсации влаги;
- атмосферное давление от 84 до 106,7 кПа.

Конструктивные особенности:

- контроллеры выполнены в компактном DIN-реечном корпусе. Габаритные и установочные размеры отличаются в зависимости от модификации и приведены в конце раздела;
- расширение количества точек ввода/вывода осуществляется путем подключения внешних модулей ввода/вывода по любому из встроенных интерфейсов.

4.2.2 Начало работы с ОВЕН ПЛК 160 в программе CoDeSys v.2.3.

Для создания нового проекта (пользовательской программы ПЛК) следует:

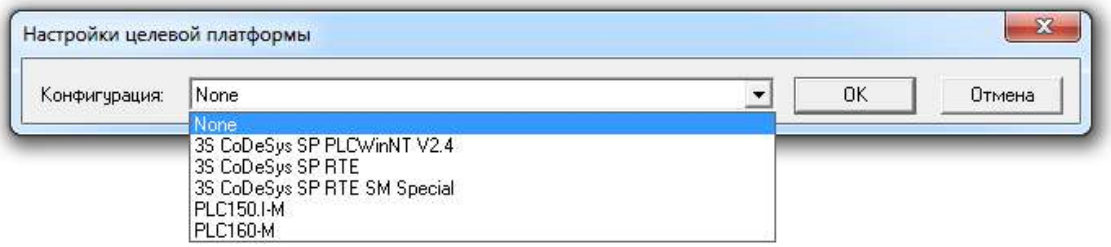
1. Выбрав команду Пуск → Программы → 3S Software → CoDeSys V.2.3 → CoDe-Sys V.2.3, запустить ПО CoDeSys.
2. В открывшемся главном окне ПО CoDeSys вызвать команду «Файл → Новый (File → New)» главного меню или нажать кнопку «Новый (New)» () панели инструментов.
3. В открывшемся окне «Настройки целевой платформы (TargetSettings)» (рисунок 4.28, а) нажатием на кнопку у правого края поля «Конфигурация (Configuration)» раскрыть список предварительно установленных на ПК Target-файлов. В списке выделить требуемый файл (рисунок 4.28, б) и щелкнуть на его названии левой кнопкой мыши.
4. В открывшихся вкладках окна «Настройки целевой платформы (TargetSetting)» отображаются установленные производителем значения

параметров целевой платформы (рисунок 4.28, в). Как правило, установленные производителем значения параметров не требуют изменения.

5. Нажать кнопку «ОК» окна «Настройки целевой платформы (TargetSetting)».
6. В открывшемся окне «Новый программный компонент (New POU)» (см. рисунок 4.29), в поле «Имя нового POU » – отображается заданное по умолчанию имя новой главной программы проекта (PLC_PRG); его не следует изменять. В группе переключателей «Тип POU» отображается заданный по умолчанию тип новой главной программы проекта (Программа (Program)); его также не следует изменять [10].

а)

б)



в)

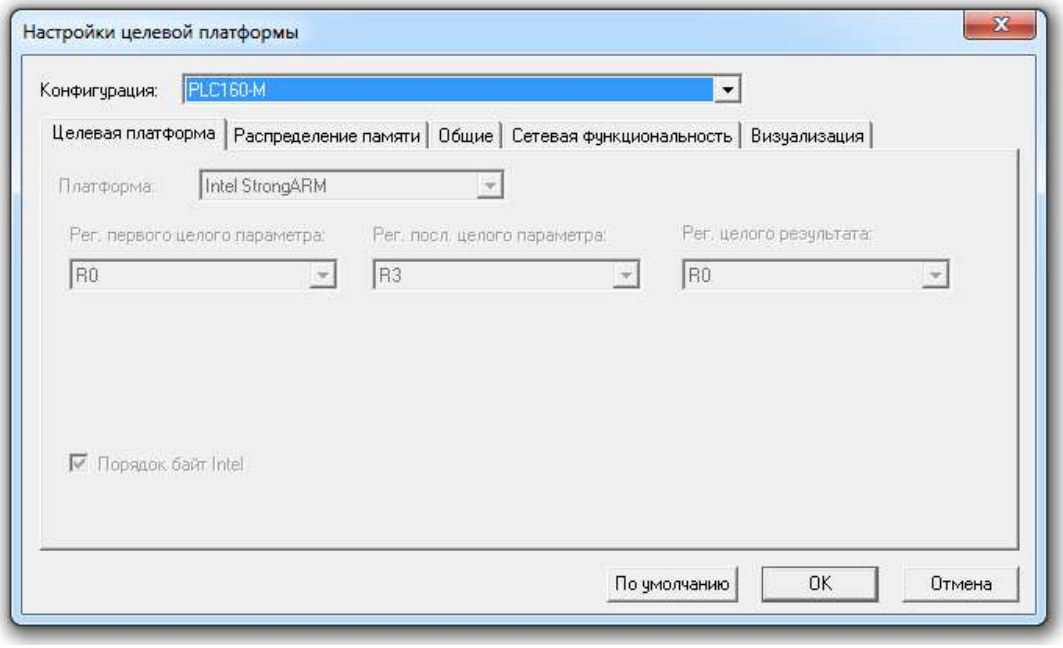


Рисунок 4.28 – Окно «Настройки целевой платформы (TargetSetting)».

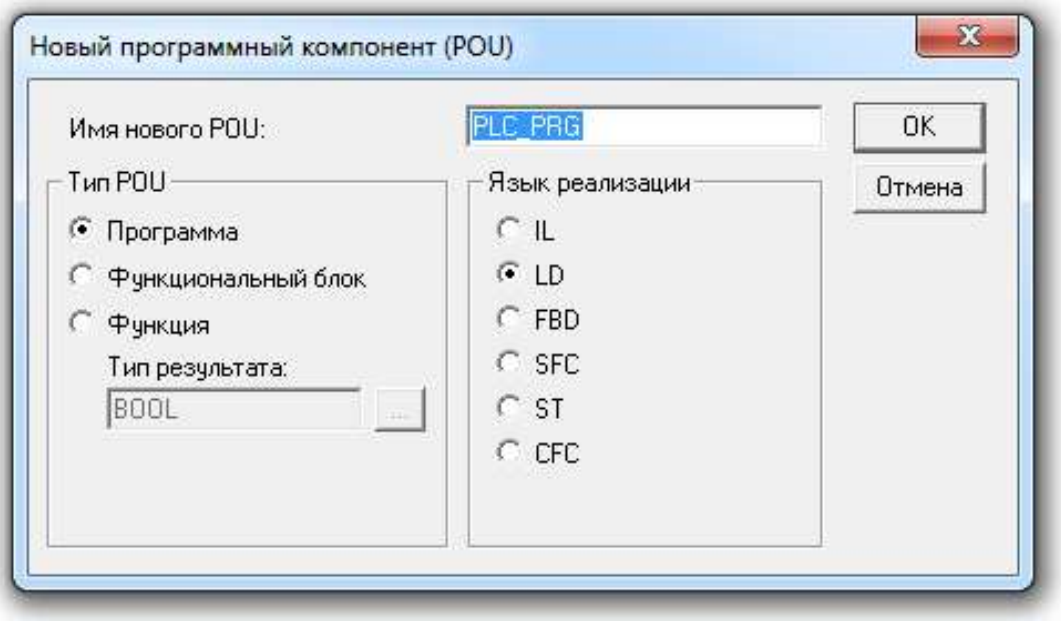


Рисунок 4.29 – Окно «Новый программный компонент (New POU)».

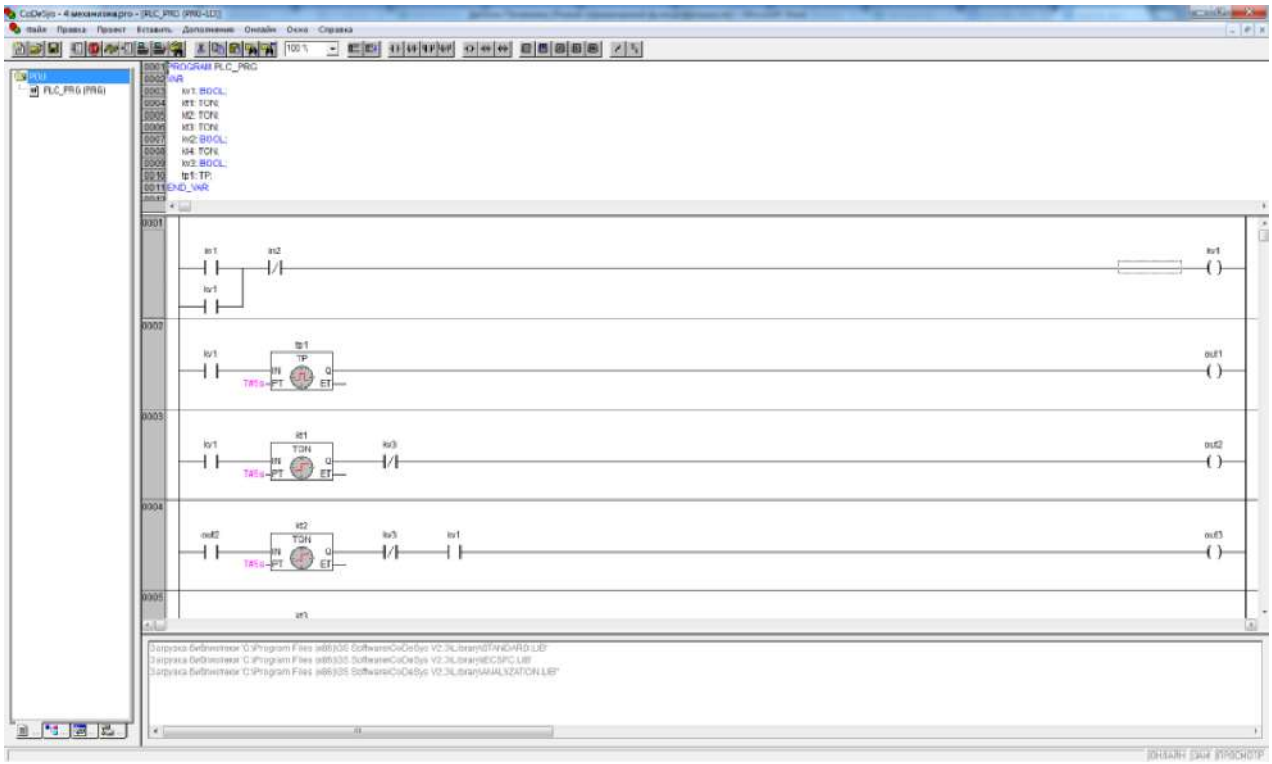


Рисунок 4.30 – Окно создания проекта PLC_PRG (PRG-LD) на языке LD (LadderDiagram – язык релейных диаграмм).

7. В группе переключателей «Язык реализации (Languageofthe POU)» следует выбрать требуемый язык программирования. В правой верхней области главного окна программы откроется окно редактора, в котором создается

программа, исполняемая контроллером. В зависимости от выбранного языка программирования это окно выглядит по-разному (на рисунке 4.30 – пример для языка LD (LadderDiagram – язык релейных диаграмм).В верхней части этого окна отображается область объявления переменных – «Редактор объявлений», в нижней – область редактора собственно программы. Одновременно главное меню программы (команда «Вставить (Insert)») дополняется командами, специфичными для выбранного языка.

Кроме того, панель инструментов дополняется локальной панелью, содержащей кнопки, соответствующие этим командам.

4.2.3 Общие сведения о языке «LD» – релейные диаграммы.

Графический язык, реализующий структуры электрических цепей; программа на языке LD состоит из схем с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата. Сложен в использовании для работы с аналоговыми типами данных [7].

Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать на нем и сложные цепи – как в FBD. Кроме того, LD достаточно удобен для управления другими компонентами POU. Используется для программирования большинства ПЛК. Допустимо переключение между языками FBD и LD.

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

Редактор LD – это графический редактор. Наиболее важные команды находятся в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>.

С помощью перетаскивания мышкой элементы (контакт, обмотку или функциональный блок) или их наименования в LD можно перемещать в другие позиции. Для этого выбирается нужный элемент (контакт, обмотка или функциональный блок) и перетаскивается, удерживая нажатой клавишу мышки. В процессе этого все допустимые места для помещения элемента будут показаны серыми прямоугольниками. Перетащите элемент в одну из этих позиций и отпустите клавишу. Элемент будет перемещен.

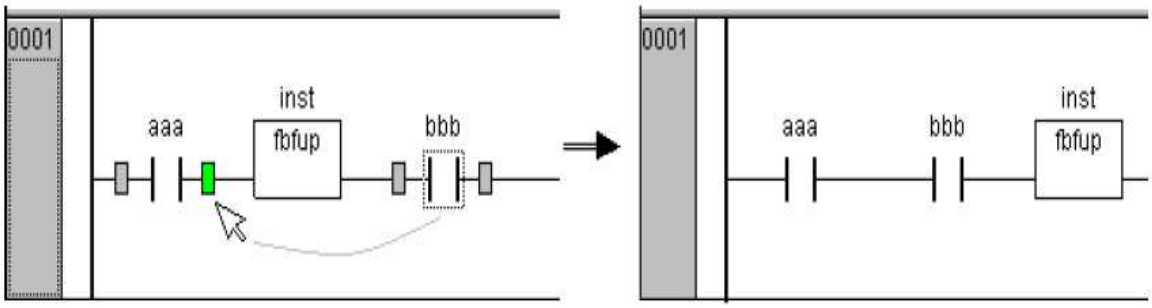


Рисунок 4.31 – Перемещение элементов в редакторе LD.

Если перетащить элемент в поле имени другого элемента, то данное поле будет подсвечено зеленым цветом. Если теперь отпустить клавишу мышки, то имя в поле будет заменено «перетаскиваемым» именем. Если включено отображение адреса и комментария (опция), то они также будут скопированы.

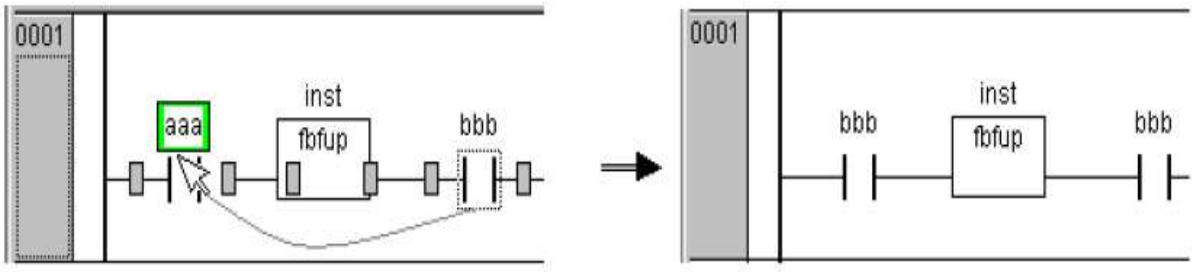
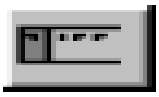
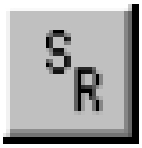


Рисунок 4.32 – Перемещение наименований в редакторе LD.

В таблице 4.3 представлены распространенные команды языка LD.

Таблица 4.3 – Назначение команд на языке LD

Название команды (быстрый ввод)	Обозн. команды	Назначение команды
'Insert' 'Network (before)'		команда используется для вставки цепи, выше выбранной в редакторе LD.
'Insert' 'Network (after)'		команда используется для вставки цепи, ниже выбранной в редакторе LD.
'Insert' 'Contact' (<Ctrl>+<K>)		команда используется для вставки контакта перед выбранной позицией в цепи.
'Insert' 'Contact (negated)' (<Ctrl> + <G>)		команда используется для вставки инверсного контакта. Она заменяет последовательность команд 'Insert' 'Contact' и 'Extras' 'Negate'.
'Insert' 'Parallel Contact' (<Ctrl>+<R>)		команда используется для вставки контакта, параллельного выделенной позиции схемы.
'Insert' 'Parallel Contact (negated)' (***<Ctrl>*** + ***<O>***)		команда используется для вставки инверсного контакта. Она заменяет последовательность команд 'Insert' 'ParallelContact' и 'Extras' 'Negate'.
'Insert' 'Coil' (<Ctrl>+<L>)		команда используется для вставки обмотки, параллельной выбранной.
'Insert' "Set' coil' (<Ctrl> + <I>)		команда используется для вставки 'Set' обмотки, параллельной выбранной. Она заменяет последовательность команд 'Insert' 'Coil' и 'Extras' 'Set/Reset'.
'Insert' 'Reset' coil'		команда используется для вставки 'Reset' обмотки, параллельной выбранной. Она заменяет последовательность команд 'Insert' 'Coil' и 'Extras' 'Set/Reset'.
'Insert' 'Function Block' (<Ctrl>+<B>)		команда используется для вставки оператора, функционального блока, функции или программы.

'Insert' 'Box with EN'		команда используется для вставки функционального блока, оператора, функции или программы с EN-входом в схему LD.
'Insert' 'Rising edge detection'		команда вставляет в цепь функциональный блок R_TRIG, который служит для выделения переднего фронта импульса (FALSE -> TRUE) сигнала.
'Insert' 'Falling edge detection'		команда вставляет в цепь функциональный блок F_TRIG, который служит для выделения заднего фронта импульса (TRUE -> FALSE) сигнала.
'Insert' 'Timer (TON)'		команда вставляет в цепь функциональный блок таймер TON, который служит формирования задержки сигнала.
'Extras' 'Negate' (<Ctrl>+<N>)		Эта команда используется для инвертирования выбранного контакта, обмотки, инструкции перехода или возврата, входа или выхода POU (позиция курсора 2 или 3). При этом в символе обмотки или контакта появляется слеш ((/) или |/|). При инвертировании инструкции перехода или возврата, входов или выходов POU появляется кружок в точке соединения, как и в редакторе FBD. Инверсная обмотка записывает в соответствующую логическую переменную значение, обратное своему. Инвертированный контакт замыкает схему, если соответствующая логическая переменная имеет значение False. Если инвертирована инструкция возврата или перехода, то она выполняется, когда соединенная с ней линия передает значение Off. Снять инвертирование с элемента можно, переинвертировав этот элемент.
'Extras' 'Set/Reset'		Если выделить обмотку и выполнить эту команду, то можно получить Set-обмотку. Такая обмотка записывает в соответствующую логическую переменную значение True, когда на входе этой обмотки имеется сигнал On, и сохраняет значение этой переменной, когда на входе сигнал Off. Такая обмотка обозначается буквой “S”.

		Выполнив эту команду еще раз, вы получите Reset-обмотку. Такая обмотка записывает в соответствующую логическую переменную значение False, когда на входе этой обмотки имеется сигнал On, и сохраняет значение этой переменной, когда на входе сигнал Off. Такая обмотка обозначается буквой “R”. Выполнив эту команду несколько раз, можно получить Set-, Reset- и обыкновенную обмотку.

В режиме Online контакты и обмотки, которые находятся в состоянии «On», изображаются синим цветом. Кроме того, все линии, передающие состояние «On», также окрашиваются синим. Указываются значения всех входов и выходов функциональных блоков. В режиме Online можно устанавливать точки останова и выполнять программу по шагам. Если вы переместите указатель мыши на переменную, то в подсказке появятся тип, комментарии и адрес переменной.

4.2.4 Настройка и установка связи с ПЛК

Перед установкой связи ПО CoDeSys с контроллером следует однократно настроить канал связи (интерфейс и настройки обмена), по которому будет осуществляться связь. В дальнейшем, при отладке программы, настройка интерфейса связи может потребоваться только при переходе на связь по другому интерфейсу.

Установка связи с контроллером возможна по интерфейсам Ethernet, Debug RS-232, USB Device или через модем, подключенный к последовательному порту RS-232 или Debug RS-232.

Для установки связи по интерфейсу USB Device следует на ПК установить драйвер, создающий виртуальный COM-порт, через который будут передаваться данные в ПО CoDeSys. Драйвер виртуального COM-порта (для операционной системы Windows 2000 и более поздних) находится на компакт-диске, входящем в комплект поставки. Для установки драйвера необхо-

димо подключить включенный ПЛК к USB-порту ПК стандартным кабелем типа A-B. После отключения питания или перезагрузки ПЛК для установки связи может потребоваться повторное отключение и подключение кабеля USB-порта для повторной инициализации драйвера.

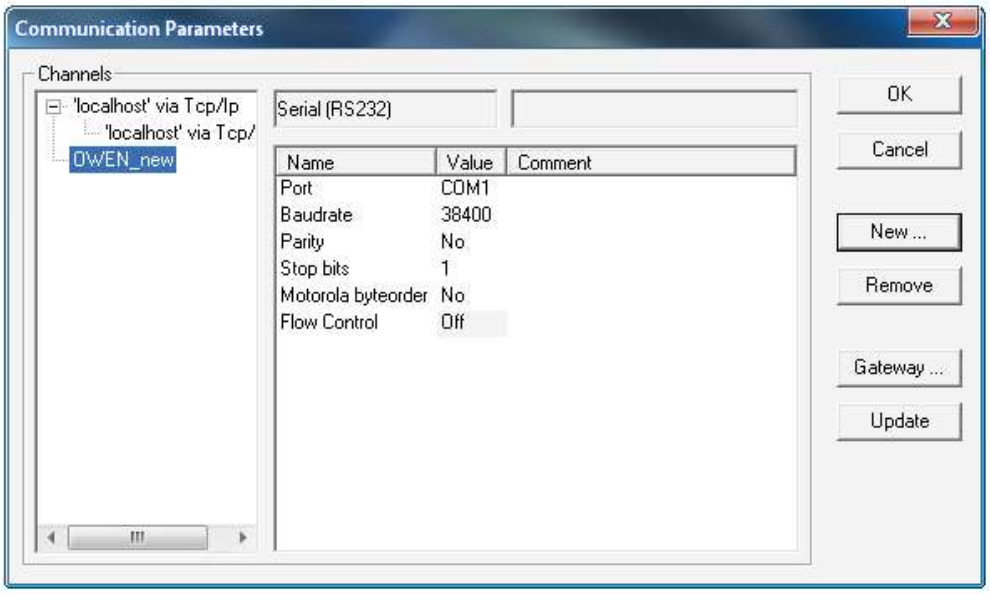
Для настройки интерфейса соединения с контроллером следует:

1. Выбрать команду «Онлайн → Параметры связи (Online → Communication parameters)» главного меню ПО CoDeSys. Откроется окно «Communication parameters», см. рисунок 4.33,а.
2. Нажать кнопку«New» окна «Communication parameters». Откроется окно «Communication parameters: New Channel» (см. рисунок 4.33,б). В этом окне задается имя нового соединения (например, Owen) и выбирается из перечня интерфейс соединения:

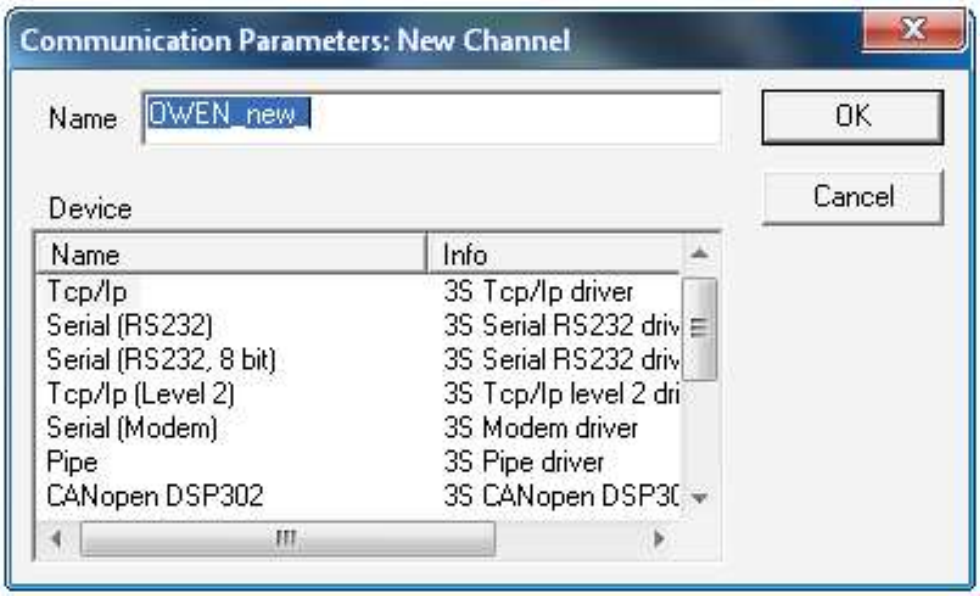
- «Serial (RS232)» для связи по интерфейсу Debug RS-232 или USB Device;
- «Tcp/Ip (Level 2)» для связи по интерфейсу Ethernet;
- «Serial (Modem)» для связи через модем, подключенный к последовательному портуRS-232 или Debug RS-232.

При выборе соединения «Serial (RS232)» в настройках параметров следует задать:

- COM-порт (параметр Port), по которому ПЛК подключается к ПК;
- скорость соединения (параметр Baudrate) 115200 бит/с;
- бит четности (параметр Parity) «No».



а)



б)

Рисунок 4.33 – Настройка интерфейса для соединения с ПЛК. Окна «Communication parameters» (а) и «Communication parameters: New Channel» (б).

Для установки связи необходимо, чтобы предварительно была создана программа пользователя. Примеры программ на языках FBD, LD и ST, которые можно использовать для проверки связи с контроллером, приведены на рисунке 4.34. Простейшей программой на языке ST является символ «:» (двоеточие). Такой программы достаточно для проверки связи с ПЛК.

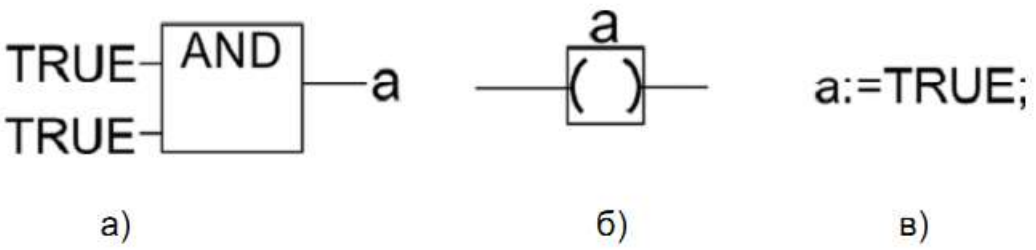


Рисунок 4.34 – Примеры программ на языках FBD (а), LD (б) и ST (в).

Для загрузки программы в контроллер следует: выбрать команду «Онлайн → Подключение (Online → Login)» главного меню, тем самым установить связь с ПЛК. При этом должен быть снят флаг перед строкой меню «Онлайн → Режим эмуляции (Online → SimulationMode)» (установка и снятие флага производится последовательными щелчками левой кнопкой мыши на строке). Перед установкой связи ПО скомпилирует проект и в случае наличия в нем ошибок прервет установку связи. Сразу после установки связи среда программирования предложит загрузить (см. рисунок 4.35) или обновить код пользовательской программы в оперативной памяти контроллера.

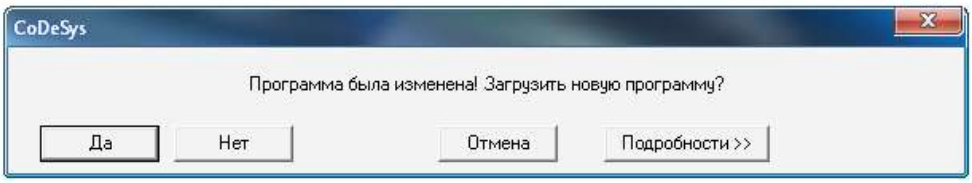


Рисунок 4.35 – Окно предложения загрузки программы.

Конфигурирование области ввода-вывода ПЛК

Перед созданием программы необходимо настроить конфигурацию входов, выходов и интерфейсов связи ПЛК с внешними устройствами (модулями ввода-вывода, устройствами индикации и т.д.), обмен данными с которыми будет производиться по сети. Перечисленные устройства обмениваются данными с пользовательской программой через специальную область памяти: Память ввода-вывода. Конфигурация ее задается в окне режима («Ресурса») «Конфигурация ПЛК (PLC Configuration)» ПО CoDeSys.

4.2.5 Пример выполнения схемы автоматизации линии предварительной обработки зерна на базе ОВЕН ПЛК 160

Реализовывать схему автоматизации линии будем на лабораторном оборудовании, внешний вид которого изображен на рисунке 4.36.

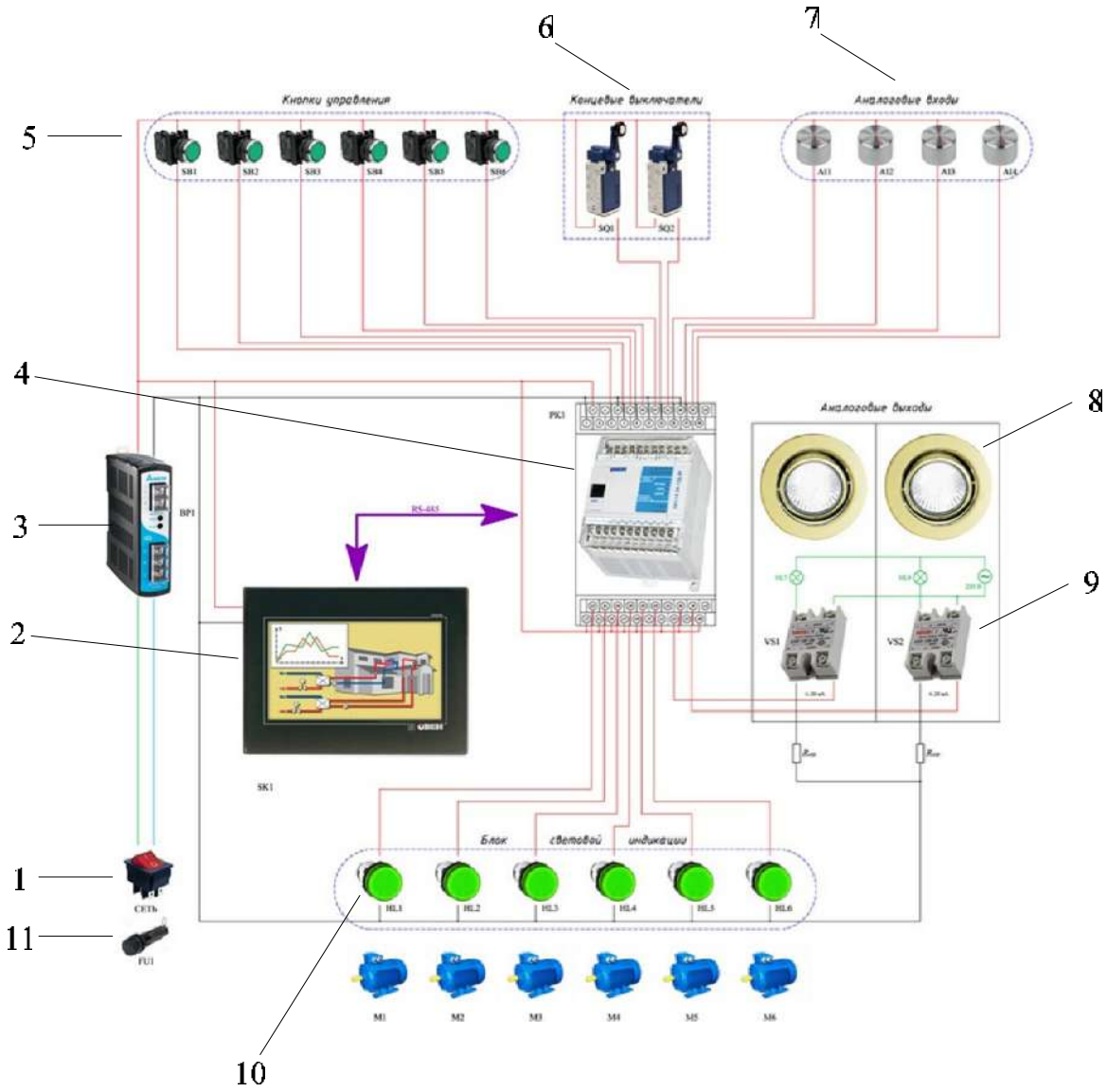


Рисунок 4.36 – Внешний вид лабораторной установки.

В состав лабораторного оборудования входит:

1. Кнопка включения/выключения питания стенда.
2. Панель оператора СП270.
3. Блок питания 24 В.

4. Программируемый контроллер ПЛК160.
5. Кнопки управления.
6. Концевые выключатели.
7. Аналоговые входы.
8. Аналоговые выходы.
9. Твердотельные реле с токовым управлением.
10. Лампы индикации работы механизмов.
11. Предохранитель.

Обозначения сигналов управления представлены в таблицах 4.4 – 4.6.

Таблица 4.4− Входные дискретные сигналы

№ п/п	Наименование сигнала	Имя источника	Разряд-ряд-ность	Сигнал
1	2	3	4	5
1	Кнопка«Общий стоп»	SB1	bit	DI 24 V DC
2	Кнопка«Пуск»	SB2	bit	DI 24 V DC
3	Кнопка «Рабочий стоп»	SB3	bit	DI 24 V DC
4	Датчик уровня	SL1	bit	DI 24 V DC

Таблица 4.5− Перечень выходных сигналов и данных

№ п/п	Наименование сигнала	Имя источни-ка	Диапа-зон из-мене-ния	Пользова-тельская ин-формация
1	2	3	4	5
1	Включение звонка	km1	1 Bit	DO 1
2	Включение скребкового транспортера	km2	1 Bit	DO 2
3	Включение молотковой дро-билки	km3	1 Bit	DO 3
4	Включение ковшовой нории	km4	1 Bit	DO 4

Таблица 4.6− Входные аналоговые сигналы

№ п/п	Наименование сигнала	Имя источника	Разряд-ряд-ность	Сигнал
1	2	3	4	5

1	Датчик веса	urov	bit	0-10 В

При написании программы для конкретного технологического процесса воспользуемся типовыми блоками схем автоматизации.

Нереверсивная схема управления электродвигателем реализована в программе CoDeSys на языке LD и CFC (рис. 4.37 и 4.38). На рисунке 4.37 представлена графически схема, на которой SB2–кнопка «пуск» с замыкающим контактом; SB1 – кнопка «стоп» с размыкающим контактом; катушка реле (магнитного пускателя) KM1, являющегося механизмом и шунтирующим замыкающим контактом KM1.

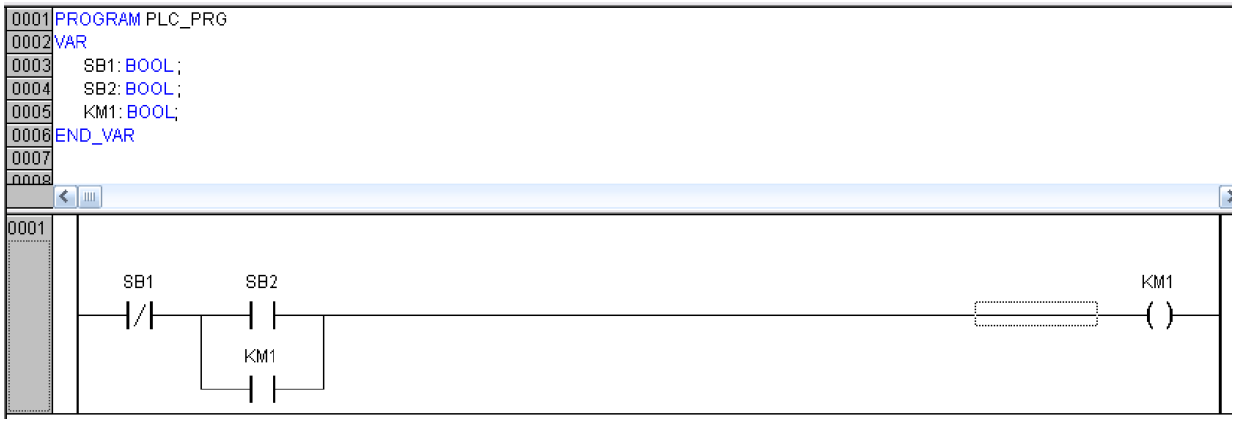


Рисунок 4.37 – Нереверсивная схема управления электродвигателем на языкеLD.

На рисунке 4.38 представлена графически схема, на которой SB1 − кнопка «стоп»; SB2 – кнопка «пуск»; R1, R2 – триггеры переднего фронта (генерируют импульс по переднему фронту входного сигнала); RS1 – триггер с приоритетом выключения; KM1 – механизм.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     SB1: BOOL;
0004     SB2: BOOL;
0005     RS1: RS;
0006     KM1: BOOL;
0007     R1: R_TRIG;
0008     R2: R_TRIG;
0009 END_VAR
```

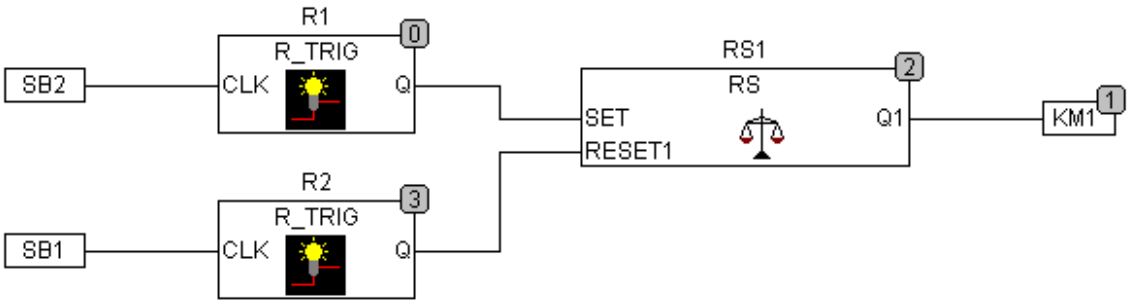


Рисунок 4.38 – Нереверсивная схема управления электродвигателем на языке CFC.

При нажатии на кнопку SB2 сигнал подается на вход R_TRIG. Триггер переднего фронта пропускает через себя сигнал импульсом, после чего на его выходе будет всегда «0». Пройденного кратковременного импульса хватает для того, чтобы RS триггер сработал, и на выходе его появилась и запомнилась «1». Это приводит к включению КМ1 – механизма. При нажатии на кнопку SB1 происходит сброс RS триггера, что приводит к отключению механизма.

Пускосигнальное звено реализовано в программе CoDeSys на языке LD и CFC (рис. 4.39 и 4.40). На рисунке 4.39, помимо уже известных нам кнопок управления, появились новые элементы: KV1 – промежуточное реле, служащее в одном случае для шунтирования кнопки пуск, в другом − для правильной последовательности включения механизмов в программе; КТ1 – реле времени (таймер с задержкой включения); катушка реле с названием MOTOR – электродвигатель механизма.

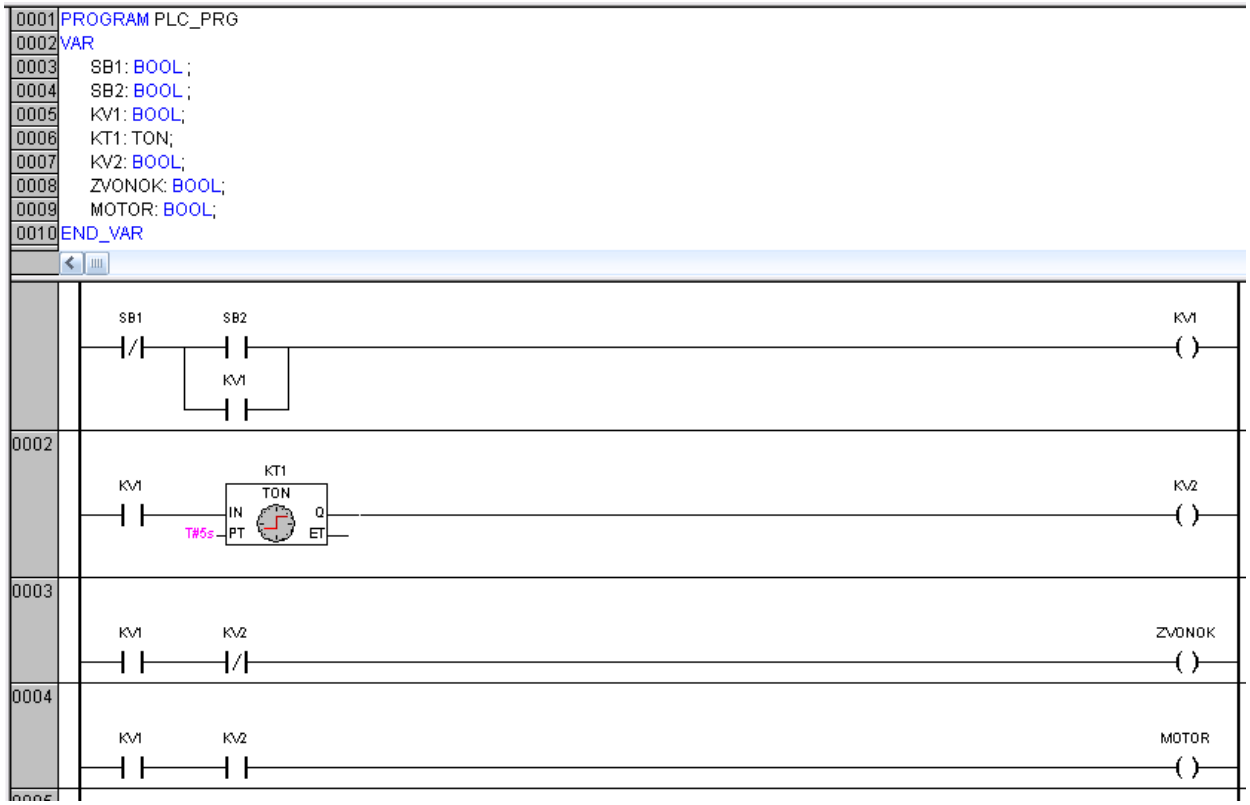


Рисунок 4.39 – Пускосигнальное звено на языке LD.

При нажатии на кнопку «пуск» SB2 включается реле KV1, контакты которого замыкаются, тем самым происходит шунтирование кнопки «пуск» и подачи питания на цепи управления (0002…0004). В цепи 0003 контакт реле KV2 замкнут, что приводит к работе катушки реле ZVONOK. Реле времени KT1 пропустит через себя сигнал с задержкой по времени, равной 5с, после которой включится промежуточное реле KV2. Одновременно с этим контакт реле KV2 в цепи 0003 отключит катушку реле ZVONOK, а в цепи 0004 включит катушку реле MOTOR. Для остановки механизма используют кнопку SB1.

На рисунке 4.40 представлена электрическая схема пускосигнального звена, реализованная на бесконтактных элементах. На ней обозначены: R1, R2 – триггеры переднего фронта (генерирует импульс по переднему фронту входного сигнала); RS1, RS2 – триггеры с приоритетом выключения; TP – таймер длительности включения (пропускает через себя сигнал определенное время, после чего на выходе «0»); zvon, motor – механизмы.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     SB1: BOOL;
0004     SB2: BOOL;
0005     RS1: RS;
0006     R1: R_TRIG;
0007     R2: R_TRIG;
0008     T1: TP;
0009     KT1: TON;
0010     RS2: RS;
0011     ZVON: BOOL;
0012     MOTOR: BOOL;
0013 END_VAR
```

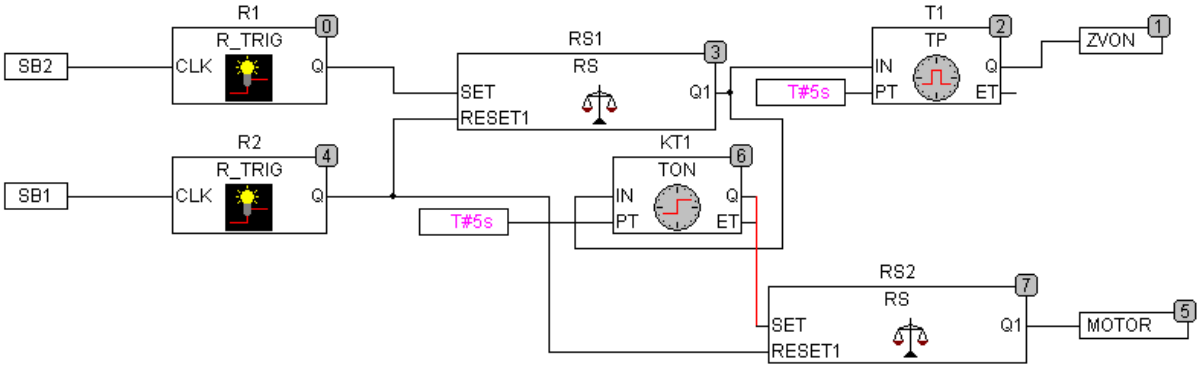


Рисунок 4.40 – Пускосигнальное звено на языке CFC.

При нажатии на кнопку SB2 сигнал подается на вход R_TRIG. Триггер переднего фронта пропускает через себя сигнал импульсом, после чего на его выходе будет всегда «0». Пройденного кратковременного импульса хватает для того, чтобы RS1 триггер сработал, и на выходе его появилась и запомнилась «1». Далее сигнал подается одновременно на два временных блока: TP сразу пропускает сигнал на выход «ZVON» ровно 5 с., после чего сигнал пропадает, «ZVON» отключается; TON пропускает через себя сигнал с задержкой, равной 5 с., после чего через RS2 триггер включается механизм «motor». При нажатии на кнопку SB1 происходит сброс RS1 и RS2 триггеров, что приводит к отключению механизма «motor».

Звено «Рабочий стоп» реализовано в программе CoDeSys на языке LD и CFC (рис.4.41 и 4.42).На рисунке 4.41 при нажатии на кнопку «пуск» SB включается реле KV, контакты которого замыкаются, тем самым происходит шунтирование кнопки пуск и подачи питания на цепь управления (0002).

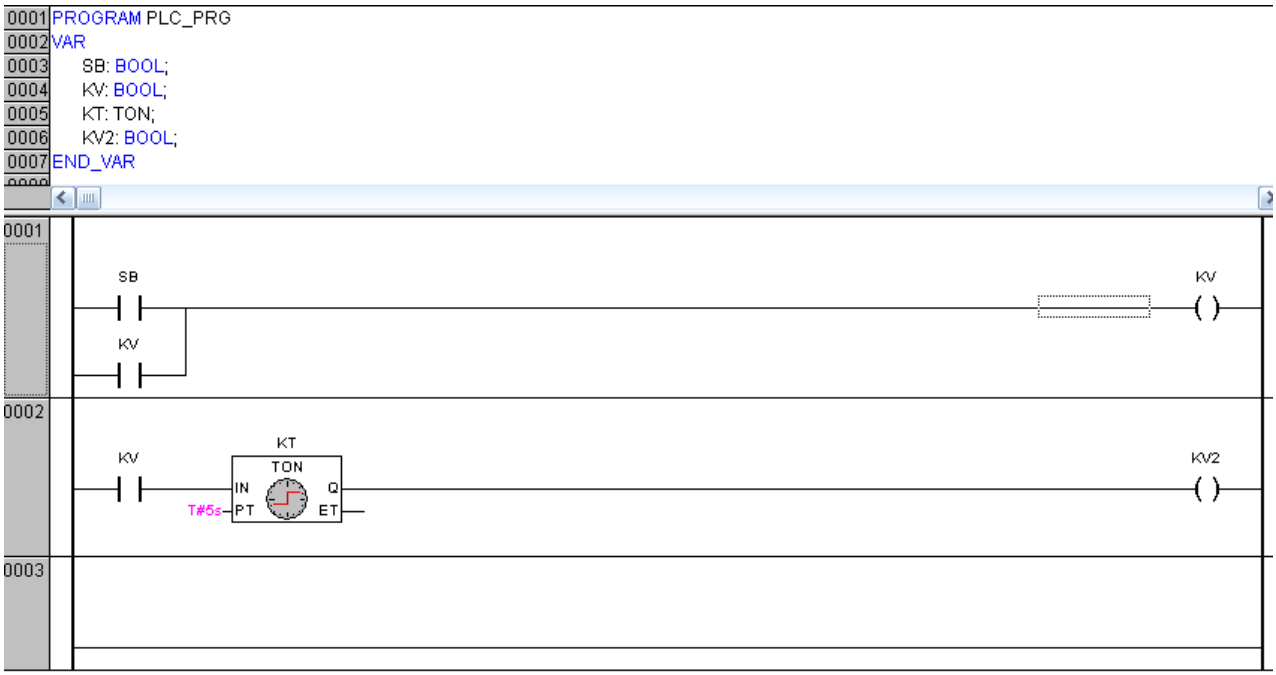


Рисунок 16 –Звено «Рабочий стоп» на языке LD.

При этом размыкающий контакт KV, стоящий в цепи с катушкой головного механизма, размыкается, тем самым приводит к отключению механизма, подающего компоненты на линию. В цепи 0002 контакт реле KV замкнут, что приводит к работе таймера. Таймер TON с задержкой по времени, необходимой на очистку линии от продукта, подаст сигнал на реле KV2, размыкающие контакты которого разомкнут цепи с катушками всех механизмов линии.

На рисунке 4.42 представлена электрическая схема звена «Рабочего стопа», реализованная на бесконтактных элементах. При нажатии на кнопку «пуск» SB сигнал подается на вход R_TRIG. Триггер переднего фронта пропускает через себя сигнал импульсом на RS1 триггер, на выходе которого записывается в память «1». Далее сигнал подается сразу на сброс RS3 триггера, который отключает головной механизм линии и через таймер TON с задержкой отключает все остальные триггеры механизмов линии, в данном случае RS4.

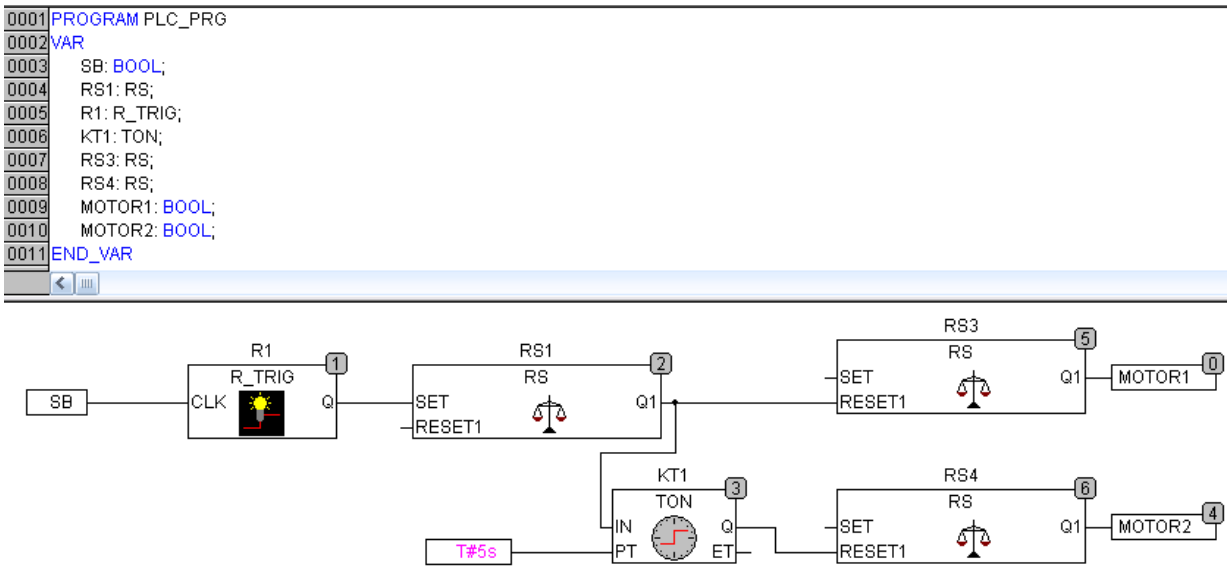


Рисунок 4.42 – Звено «Рабочий стоп» на языке CFC.

На основании разработанных типовых схем составим программу управления линии предварительной обработки зерна (см. п. 2.4), внешний вид которой изображен на рисунке 4.43. Разработанный алгоритм управления реализуется в программном продукте CoDeSys на базе языка программирования LD (релейных диаграмм).

Как видим, в написании программы управления линии предварительной обработки зерна использовались типовые блоки, из которых, как в конструкторе, собирается алгоритм управления. Каждый участок линии 0001….0009 начинается с замыкающего контакта kv1. Это сделано для того, чтобы питание на цепи подавалось только после включения кнопки «пуск» SB2.

В данной программе на участке 0009 представлено новое звено, которое отвечает за контроль веса в бункере. В этом участке происходит слияние релейно-контактных схем с их дискретными сигналами и бесконтактных схем с аналоговыми сигналами. В блоке «dwq» учитывается аналоговый сигнал, приходящий с аналогового датчика, сравнение его с текущей уставкой (значение физической величины заданное) и на выходе результат сравнения.

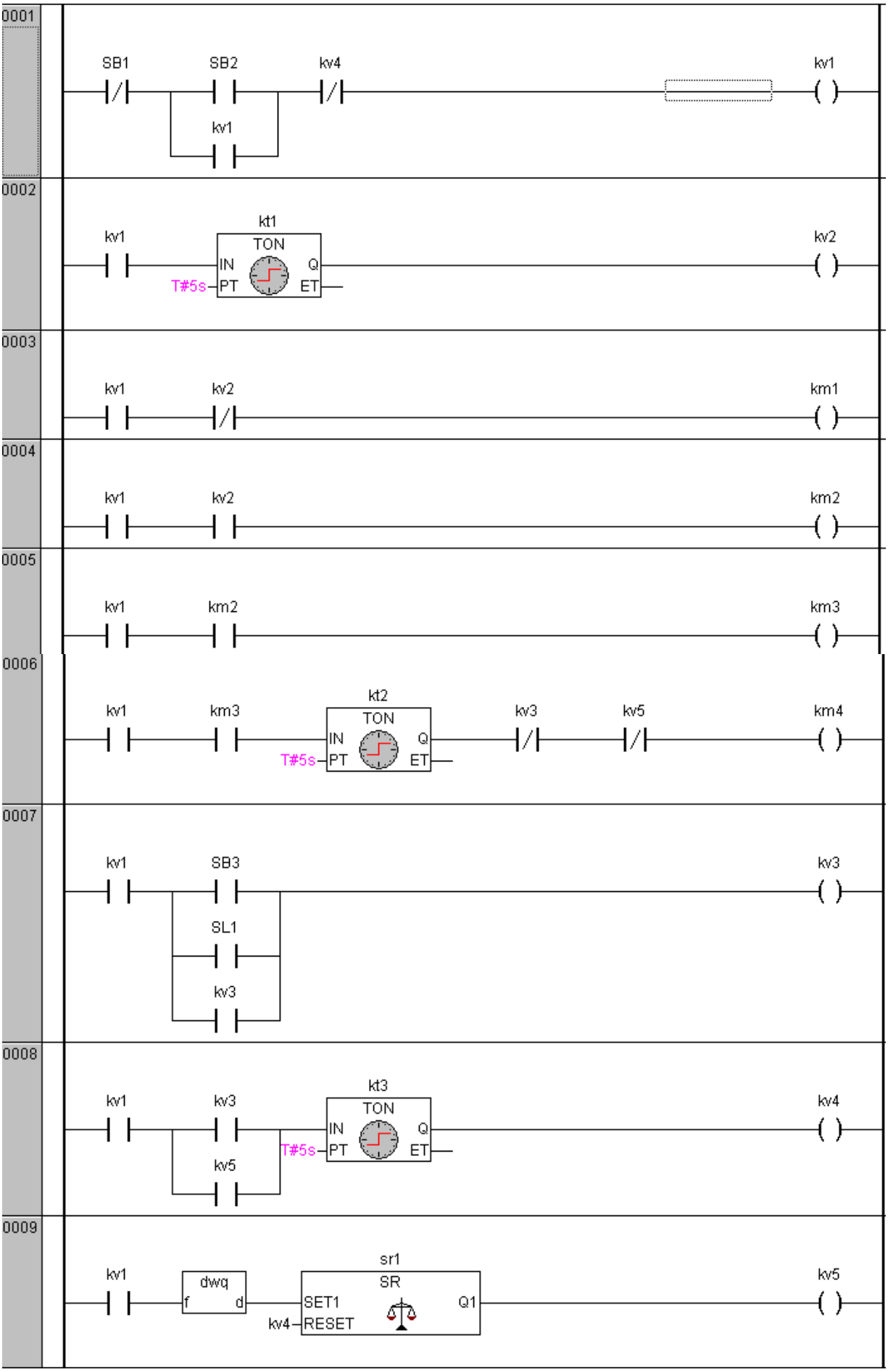


Рисунок 4.43 – Алгоритм управления линией на языке LD.

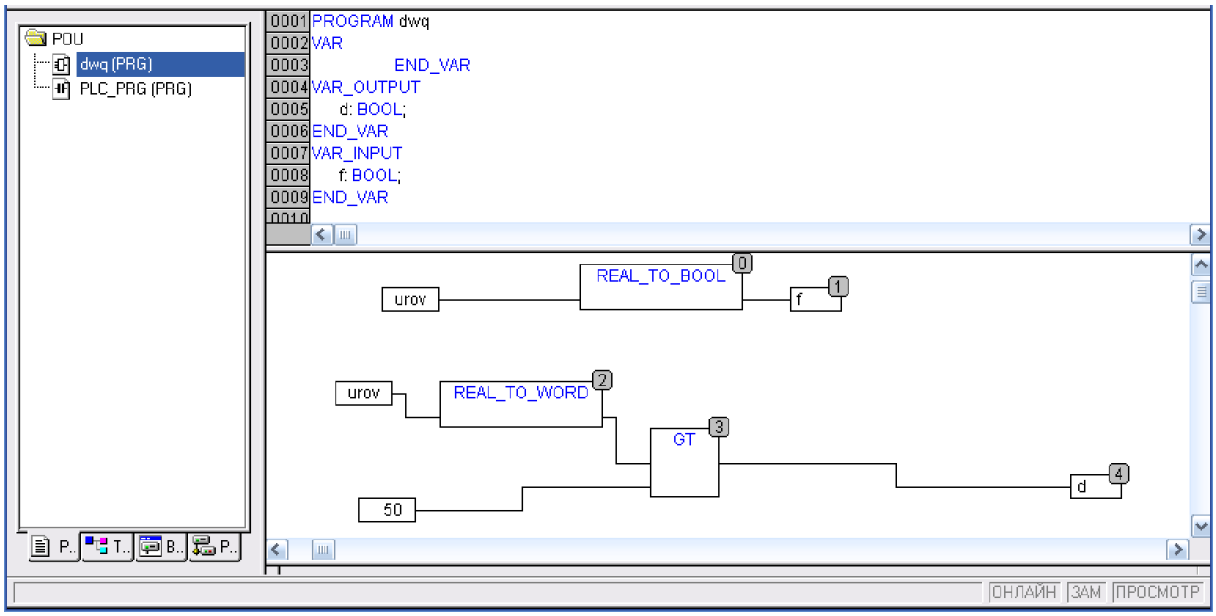


Рисунок 4.44 – Реализация блока «dwq» в CoDeSys на базе языка программирования CFC.

Для более детального рассмотрения блока «dwq» представим его в развернутом виде на рисунке 4.44. Функциональный блок «dwq» состоит из входной величины urov и выходной d. Именно величина urov отвечает за аналоговый сигнал, приходящий на вход контроллера. Этот сигнал сравнивается со значением уставки, равной 50. Сравнение происходит в блоке GT. В этом блоке сравнение может осуществляться только между величинами, имеющими одинаковые типы данных. Число 50, целочисленное значение, сравнивается с величиной urov – вещественной переменной, поэтому необходимо преобразовать тип данных величины urov из вещественной переменной в целочисленное значение. Для этого служит блок REAL_TO_WORD. Результат сравнения величин формируется в выходную величину d.

Для правильной привязки глобальных переменных к реальным входам выходам контроллера используют вкладку «Ресурсы» с конфигурацией ПЛК. На рисунке 4.45 представлен внешний вид вкладки конфигурации ПЛК с прописанными глобальными величинами.

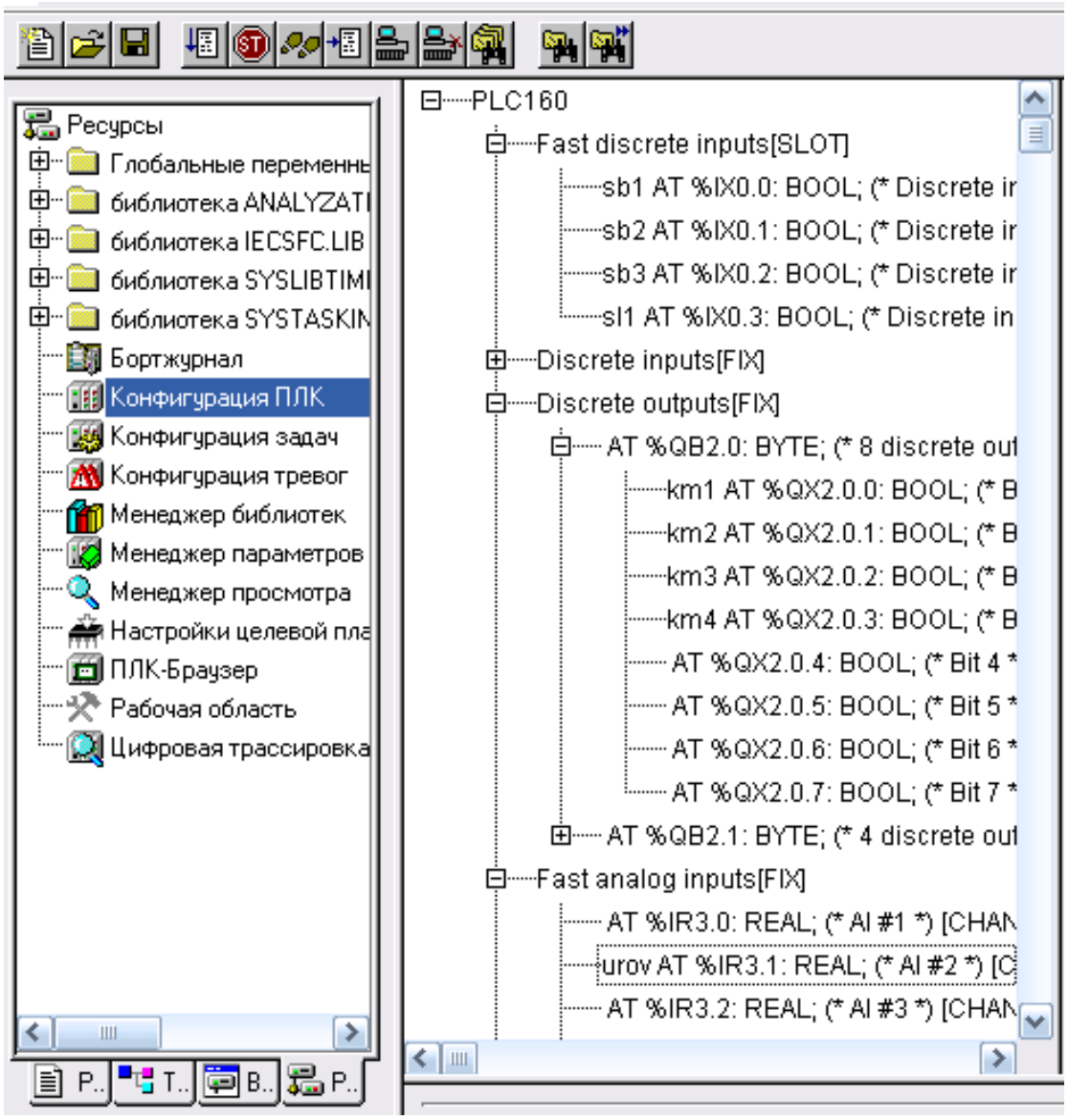


Рисунок 4.45 – Внешний вид вкладки конфигурации ПЛК.

На рисунке 4.45 показана конфигурация контроллера ПЛК160, у которого имеется 4 быстрых дискретных входа, где прописаны наши глобальные переменные sb1, sb2, sb3, sl1, которые участвуют в написании программы управления линии предварительной обработки зерна (рисунок 4.43). К этим дискретным входам контроллера подключены кнопки управления и датчик уровня.

У контроллера имеются дискретные выходы, где прописаны наши глобальные переменные km1, km 2, km 3, km 4. К этим дискретным выходам контроллера подключены катушки магнитных пускателей механизмов линии.

К аналоговому входу контроллера подключен датчик, глобальная пе-

ременная, описывающая сигнал которого является величиной urov.

Для имитации работы аналогового датчика в лабораторном стенде используется переменный резистор, который формирует управляющий сигнал в виде напряжения 0…10 В. В связи с этим необходимо настраивать параметр канала входа. На рисунке 4.46 представлен внешний вид настройки параметра канала входа аналогового датчика.

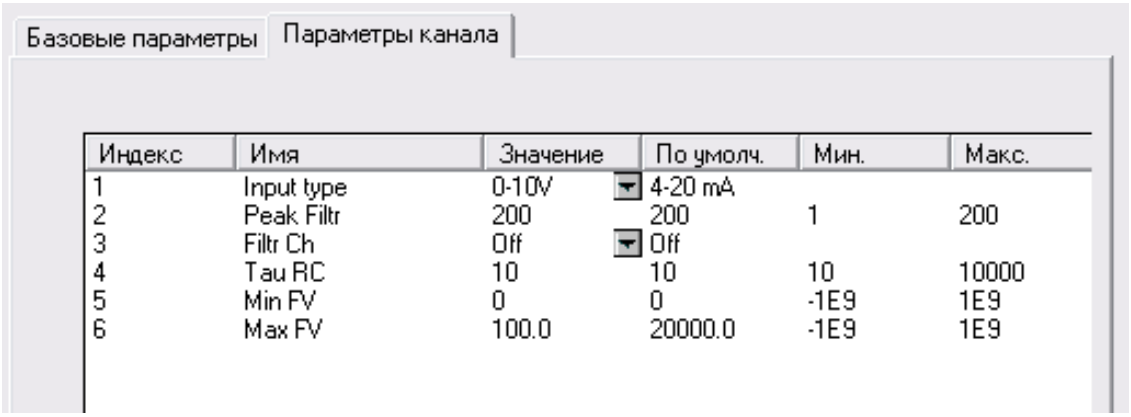


Рисунок 4.46 – Внешний вид вкладки конфигурации ПЛК.

В параметрах канала настраивается: индекс 1 – выбирается значение приходящего сигнала от датчика. В индексе 6 – устанавливается верхняя граница измеряемой физической величины. Все остальные настройки параметра канала можно оставить по умолчанию.

Аналогично с учетом требований, предъявляемых к линии предварительной обработки зерна, разрабатывается программа управления в CoDeSys на базе языка программирования CFC (рисунок 4.47). Для нее характерно использование функциональных блоков совместно с элементами логики.

Как видно из представленной схемы, здесь также использовался принцип конструктора. Все входные величины расположены в левой части схемы, выходные − в правой. На таймерах времени КТ1…КТ3 установлены временные задержки на уровне 5 сек.

Для осуществления контроля уровня заполнения бункера используется аналоговый датчик, благодаря которому появилась возможность координировать уровень заполнения. В связи с этим возникла необходимость в появлении звена,

отвечающего за контроль аналогового сигнала. На рисунке 4.48 представлено звено контроля уровня заполнения бункера.

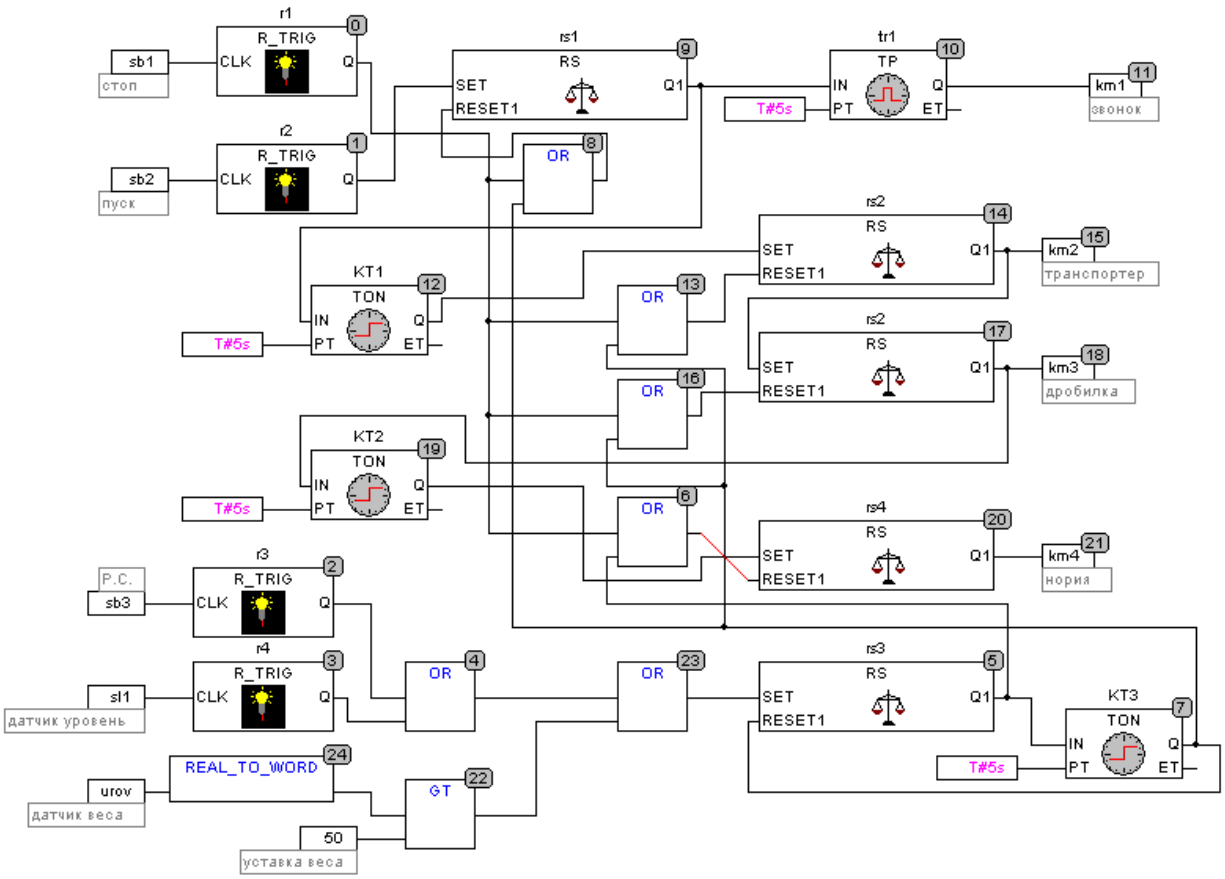


Рисунок 4.47 – Программа управления механизмов линии на языке CFC.

Звено контроля практически аналогично звену, изображенному в блоке «dwq», поэтому описывать подробно его не будем [9].

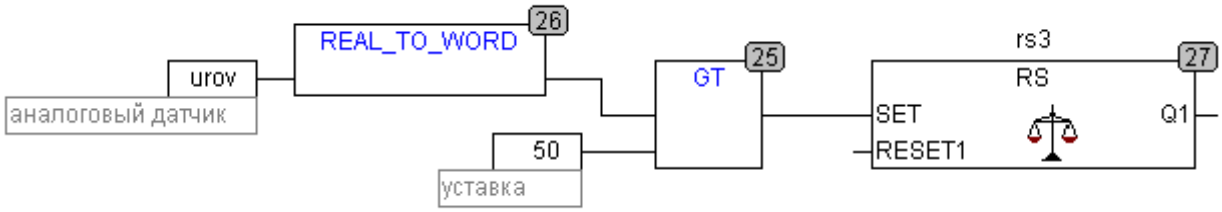


Рисунок 4.48 – Внешний вид звена, отвечающего за контроль аналогового сигнала датчика.

Зачастую бывает такая ситуация, когда внутри программы алгоритм управления работает правильно и соответственно правильно работают выходные устройства контроллеров, а на деле нет контроля за работой самих

механизмов. На рисунках 4.44 и 4.48 алгоритмы управления линей составлены правильно и удовлетворяют требованиям, только нет обратной связи с реально действующим электрооборудованием. В связи с этим необходимо дополнить разработанные нами схемы элементами, которые позволят нам контролировать работоспособность электродвигателей.

Рассмотрим доработку схемы управления, реализованную на графическом языке программирования LD. Для этого необходимо в цепи 0001…0009 с катушками управления выходных устройств km2…km4 поместить замыкающие контакты глобальных переменных km11…km31.

Если катушка управления внутри программы km1 получила питание, то соответственно дискретный выход 1 включится и подаст питание на магнитный пускатель 1, который включит звонок. Если звонок включен, то тогда на дискретный вход котроллера придет питание через дополнительный контакт магнитного пускателя 1 и соответственно тогда только замкнется контакт km11, что приведет к включению механизма транспортера, в противном случае транспортер работать не будет.

Сказанное выше реализуется довольно-таки просто в программе. Для этого необходимо добавить глобальные переменные km11, km21, km31.

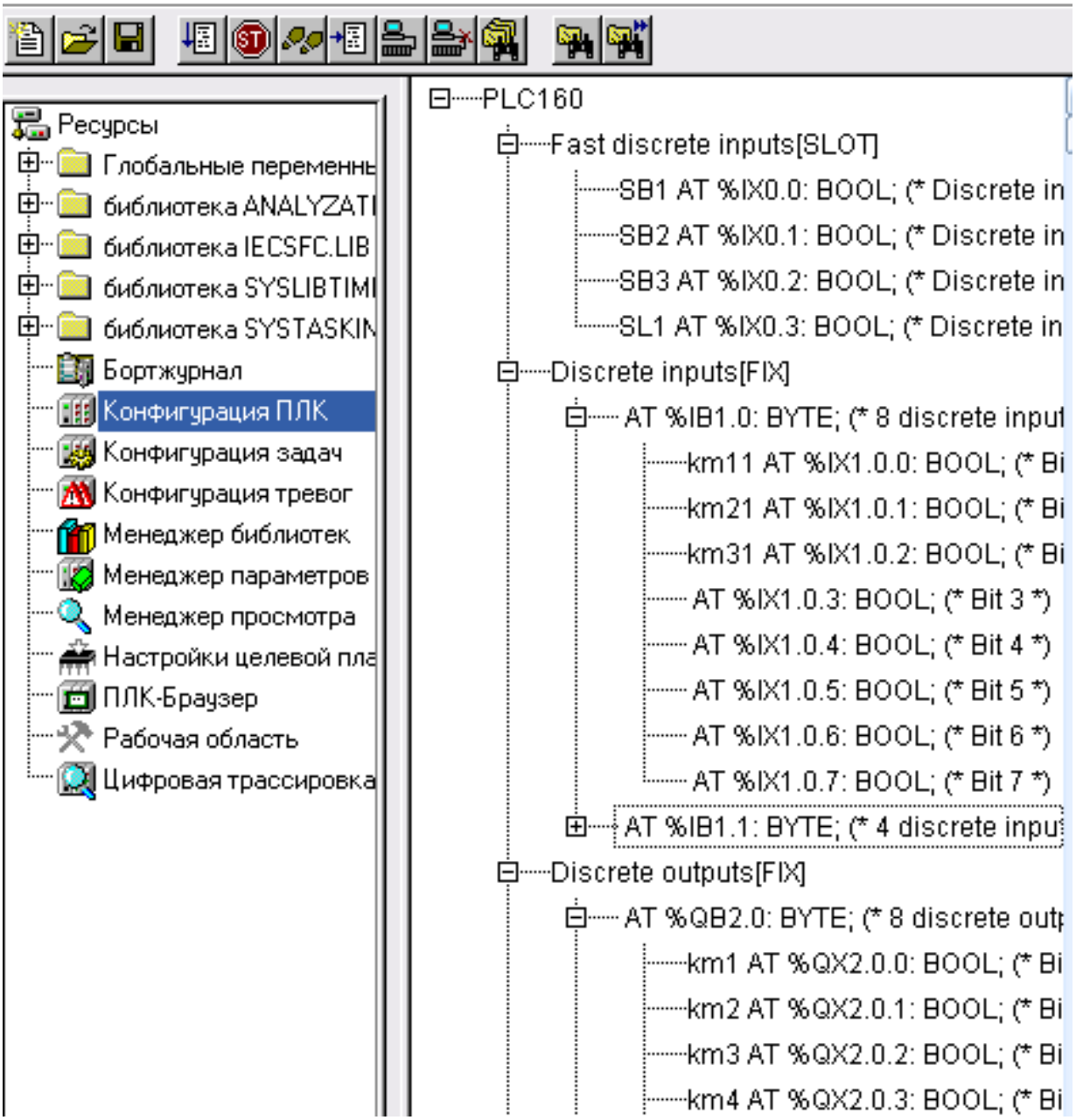


Рисунок 4.49 – Внешний вид вкладки конфигурации ПЛК.

На рисунке 4.49 показана конфигурация контроллера ПЛК160, у которого, помимо уже существующих 4 быстрых дискретных входов, где прописаны наши глобальные переменным sb1, sb2, sb3, sl1, добавляются новые km11, km21, km31.

Разработана новая программа управления линии с контролем состояния работы механизмов. Алгоритм управления реализуется в программном продукте CoDeSys на базе языка программирования LD (релейных диаграмм).

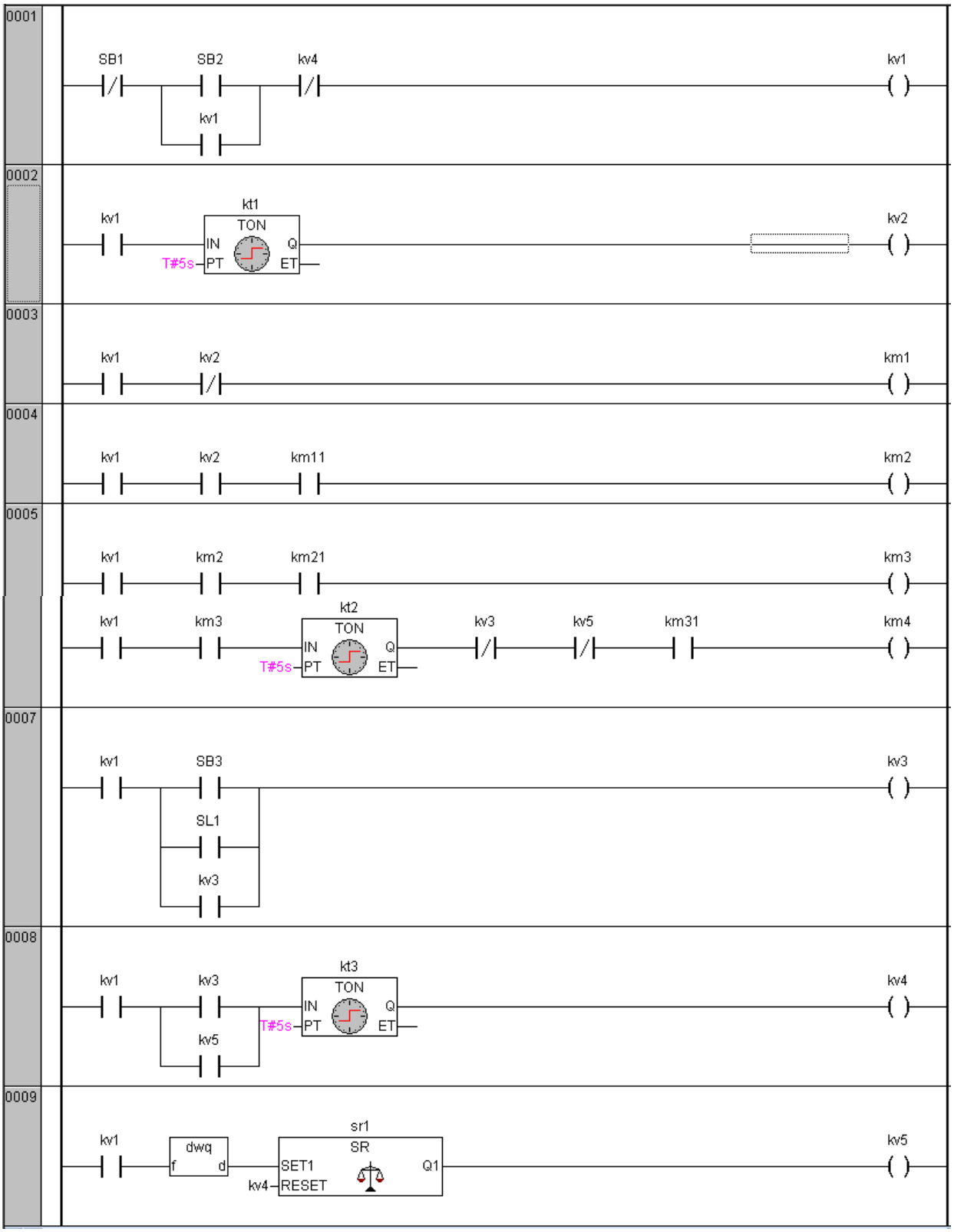


Рисунок 4.49 – Внешний вид программы управления линией

5. СИСТЕМЫ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ НА БАЗЕ ПЛАТФОРМЫ ARDUINO

5.1 Общие сведения

Arduino − это плата с микроконтроллером и памятью. Существует большое множество различных вариаций рассматриваемой платы, но наибольшее распространение получила версия Arduino Uno (рис. 5.2), построенная на базе микроконтроллера ATmega328p с тактовой частотой 16 МГц (рис. 5.1) и обладающая памятью 32 кБ [23].

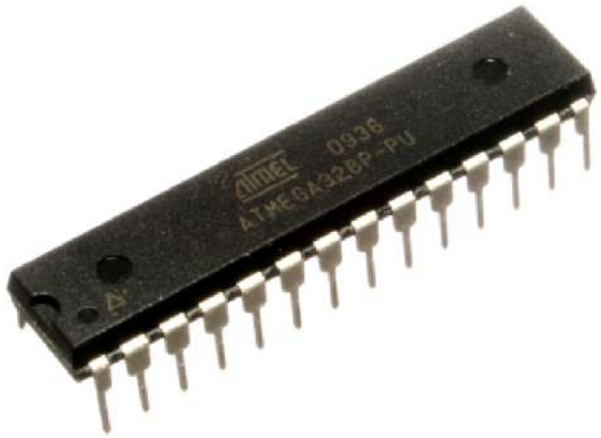


Рисунок 5.1−Микроконтроллер ATmega328p.

На данной плате расположены контакты (или так называемые пины), к которым можно подключать всевозможные компоненты: светодиоды, датчики, моторы, светильники, любое электрооборудование. Всего таких пинов 20. Из них 14 (от 0 до 13) используются как цифровые входы/выходы, остальные 6 как аналоговые входы (пронумерованы на плате от A0 до A5) (рис. 5.2). Среди цифровых пинов встречаются контакты с тильдой (~). Этим знаком обозначаются пины, поддерживающие широтно-импульсную модуляцию (ШИМ), о чем будет рассказано ниже. GND – пин заземления.VIN – пин для подключения питания при отсутствии батарейки или USB.

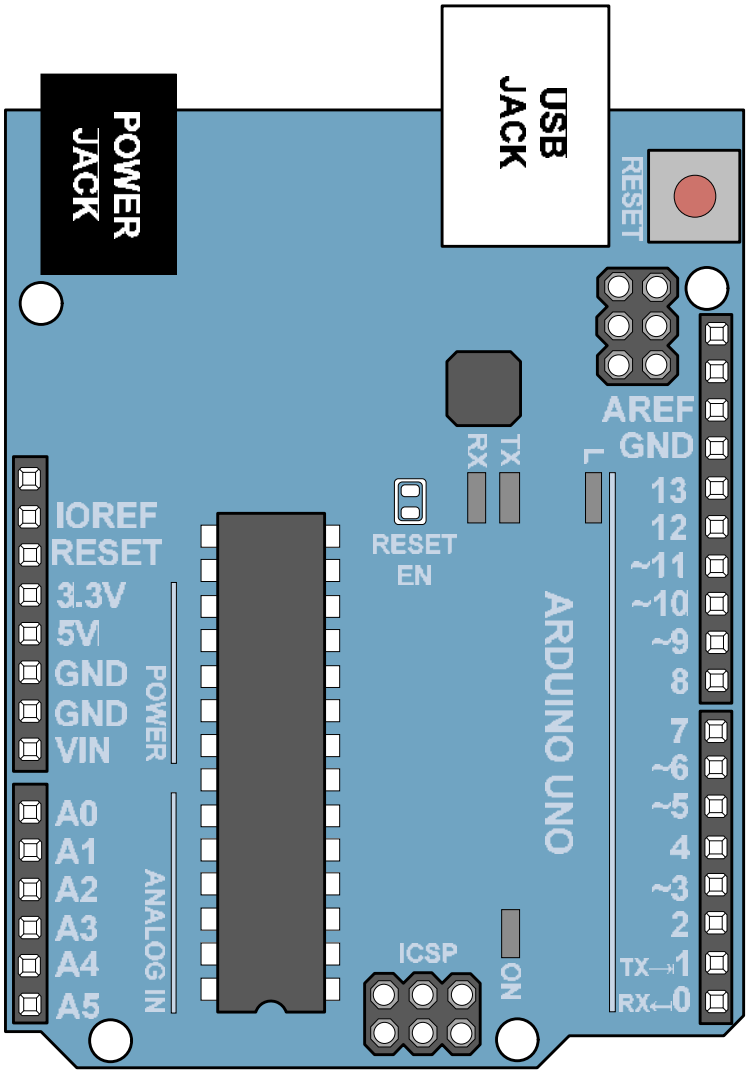


Рисунок 5.2 – Расположение контактов Arduino Uno.

Стоит пояснить, что такое цифровые и аналоговые входы/выходы.

Использование цифрового сигнала подразумевает подачу на любой из 14 пинов либо 0В, либо напряжения своего питания — 5В (рис. 5.3). То есть либо логического нуля (LOW – 0В) либо логической единицы (HIGH – 5В). Промежуточных значений нет. Если напряжение не равно 5В или 0В, микро-контроллер округляет показания датчика в ближайшую сторону.

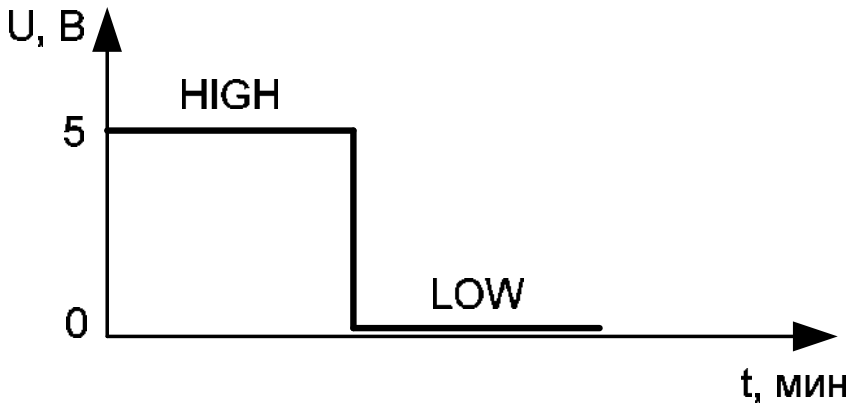


Рисунок 5.3 – Цифровой сигнал.

Достоинством цифрового сигнала является его простая реализация в Arduino, а также возможность использования длинных проводников для подключения датчиков к плате без боязни исказить сигнал за счет воздействия внешних электромагнитных воздействий [13, 3].

При подключении аналоговых датчиков к аналоговым входам Arduino показания датчика передаются в виде переменного напряжения. То есть напряжение может принимать значение от 0В до напряжения питания – 5В. Между измеряемой величиной и возвращаемым обратно напряжением установлена определенная зависимость. Например, фоторезистор − элемент, изменяющий свое сопротивление от количества света, падающего на него (рис. 5.4). При полной темноте его сопротивление максимально, а при появлении света сопротивление снижается.

Таким образом, в зависимости от количества света, падающего на наш датчик (фоторезистор), на входе A0 будут подаваться различные значения напряжения от 0 до 5 В.

Недостатком аналогового сигнала является чувствительность к внешним шумам. Если провод от датчика до платы имеет большую длину, то он начнет улавливать внешние воздействия, искажая показания. Рациональной считается длина проводника от аналогового датчика к микроконтроллеру не более 0,5 м.

Рисунок 5.4– Фоторезистор.

Как и в любой микроконтроллер, в ATmega328p можно загрузить программу, которая будет управлять подключенными к пинам устройствами. Для программирования используется упрощенная версия языка C++, (Wiring). Разработка программы ведется в бесплатной среде Arduino IDE (рис.5.5).

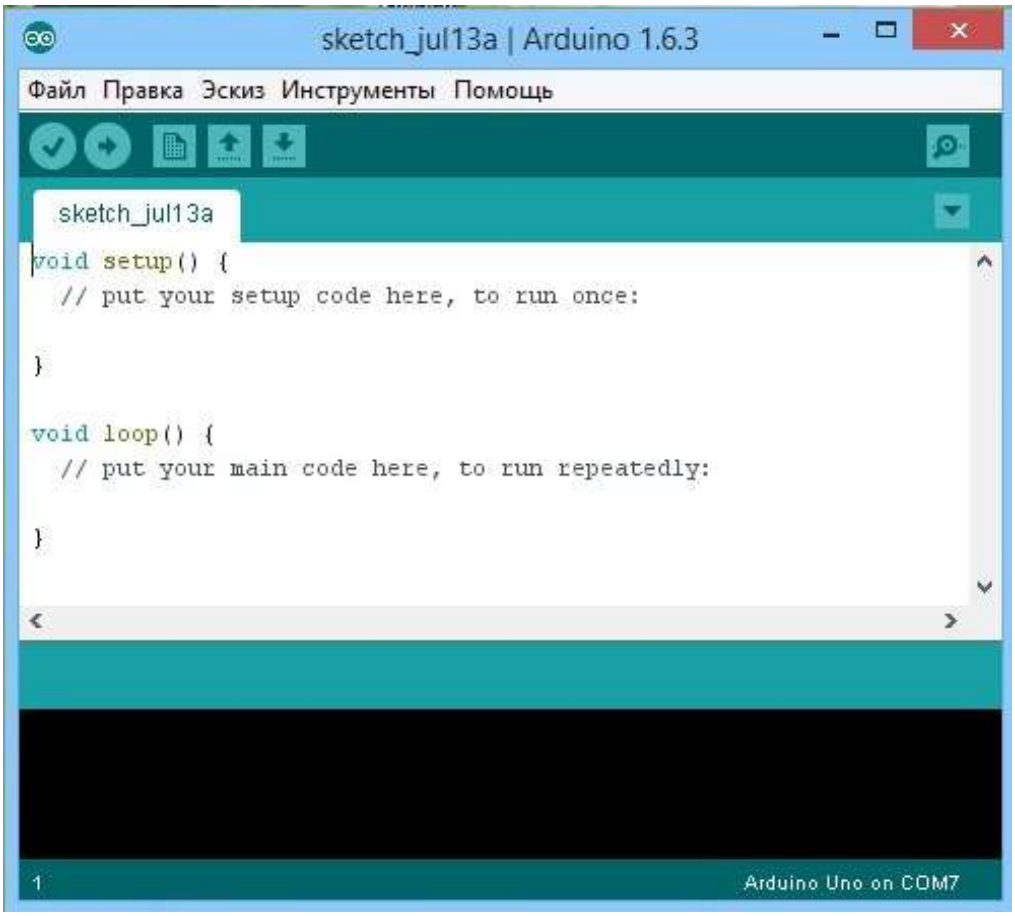


Рисунок 5.5−Arduino IDE.

Поддерживаются операционные системы Windows, MacOS X и Linux. Для программирования используется USB-кабель.

Питание платы может осуществляться как через USB, так и от батарейки на 9В или источника питания постоянного тока на 5−12В. Питание определяется автоматически.

5.2 Программирование Arduino

Программа в Arduino IDE состоит из двух главных процедур: **«void setup()»** и **«void loop()»** (рис. 5.5). Начало процедуры начинается левой фигурной скобкой − {, а конец процедуры – правой фигурной скобкой − }. Две данные процедуры обязательно должны быть в каждой программе. Без них процесс компиляции не произойдет.

Процедура **«void setup()»** необходима для того, чтобы сконфигурировать плату и обозначить, какой из пинов является выходом, а какой входом. Чтобы это сделать, необходимо воспользоваться функцией **«pinMode()»**. Обозначим, например, 5 пин нашей платформы как выход [15]:

```
voidsetup ( )
{
pinMode(5, OUTPUT);
}
```

Как видно из листинга программы, мы заключили функцию **«pinMode()»** в процедуре **«void setup()»**. Причем сама функция внутри скобок имеет два так называемых аргумента: 5 и OUTPUT. Первый аргумент – это номер пина, который мы хотим сконфигурировать, второй – это статус, который мы ему задаем (в нашем случае выход – OUTPUT, если же мы хотим сделать пин входом, то необходимо написать INPUT). Процедура **«void setup()»** выполняется только один раз при запуске программы.

Обратите внимание на знак «;» после скобки функции **«pinMode()»**. Дело в том что язык С++ таким образом понимает, где заканчивается выра-

жение нашей функции. Данный знак необходимо ставить в конце функции всегда, иначе при компиляции программа выдаст ошибку.

Далее необходимо задать действие для электрооборудования (им может быть все, что угодно: светодиоды, вентиляторы, сервоприводы, двигатели и т.д.), которое подключено к нашему 5 пину. Разберем пример, в котором нам необходимо включать на 5 секунд и выключать на 10 секунд вентилятор, подключенный к 5 пину. Для этого необходимо, чтобы на входе вначале появлялся сигнал (5В – HIGH) на 5 секунд, а затем пропадал (0В – LOW) на 10 секунд. Эта задача решена в листинге программы, приведенной ниже:

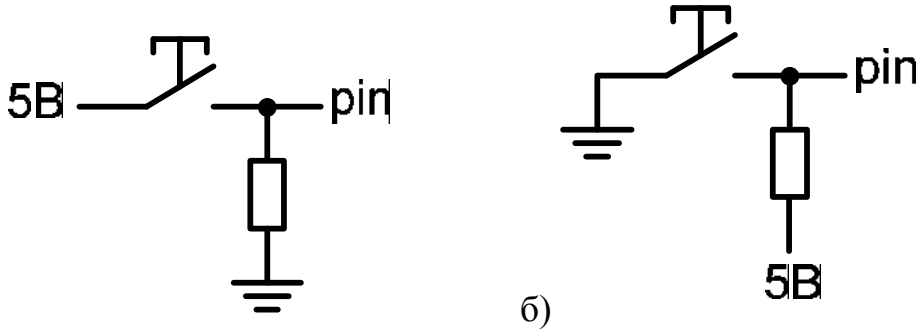
```
void setup ( )
{
pinMode(5, OUTPUT);          // объявляем 5 пин выходом
}

/* Процедура void loop ( )используется для того, чтобы осуществлять
выполнение действий над пинами */
voidloop ( )
{
digitalWrite(5, HIGH);       // включаем вентилятор
delay(5000);                 // на 5 секунд
digitalWrite(5, LOW);        // выключаем вентилятор
delay(10000);                // на 10 секунд
}
```

Очень полезно при написании программы писать комментарии, поясняющие ее работу. Это делается с помощью двойного слеша «//» вначале комментария. Такой комментарий может занимать одну строчку, но если он занимает несколько строчек, то его необходимо заключить в символы /* … */, как показано в листинге программы выше.

Все действия над пинами в нашей программе выполняются в процедуре **«void loop()»**. Для того чтобы запустить или выключить наш вентилятор, мы используем функцию **«digitalWrite()»**. Дословный перевод функции «цифровая запись», то есть мы «записываем» значение 0 (выключено) или 1 (включено) в пин 5. Она также содержит два аргумента. Первый аргумент нам хорошо знаком – это номер нашего пина, к которому подключен вентилятор, второй аргумент отвечает за подачу сигнала. Запись: **digitalWrite(5, HIGH);** означает, что в первоначальный момент времени работы программы на 5 пин будет подан высокий сигнал (HIGH), равный 5 В (логическая единица) и наш вентилятор включится. Далее ниже записана функция **«delay()»**, отвечающая за остановку программы. Данная функция содержит всего один аргумент, в который мы должны записать время задержки выполнения программы. Временные промежутки в программе записываются в миллисекундах, поэтому наши 5 секунд, в которые должен работать вентилятор, представлены как 5000 мс. Затем мы снова используем функцию **«digitalWrite()»**, но уже с другим аргументом – LOW, который подаст сигнал, равный 0В (логический ноль) на 5 пин и тем самым выключит наш вентилятор. Отключение происходит на 10 секунд с помощью функции **«delay()»**. Далее вентилятор вновь включается и процесс повторяется вновь, пока не исчезнет питание. То есть, по сути, процедура **«void loop()»**− это цикл. Важно отметить, что функция **«delay()»** останавливает выполнение программы на указанное в скобках время, то есть любые другие действия в программе невозможны, пока выполняется данная функция. Например, в ситуации, когда необходимо остановить поочередно запускающиеся с некоторым интервалом двигатели, функция **«delay()»** не подходит и вместо нее необходимо использовать функцию **«millis()»**, которая показывает, какое время прошло с момента запуска программы. О работе этой функции будет рассказано ниже.

При подключении кнопок к платам Arduino необходимо помнить о таком явлении, как «дребезг контактов». Данное явление связано с тем, что при нажатии на кнопку между ее контактами образуются десятки микроразрядов-

за несколько миллисекунд. Это явление необходимо учитывать при написании программы, в которой фиксируется момент нажатия на кнопку. Для решения проблемы дребезга контактов используют схемы со стягивающим или с подтягивающим резистором [15].



а) б)

Рисунок 5.6 – Схемы со стягивающим резистором (а) и с подтягивающим (б).

Схема со стягивающим резистором (рис. 5.6,а) будет подавать питание на пин микроконтроллера при нажатии на кнопку, а схема с подтягивающим резистором (рис. 5.6,б) наоборот, будет выдавать питание на пин, пока кнопка не нажата. На цифровыхпинах платформы Arduino уже имеются подтягивающие резисторы. Для того чтобы их задействовать, при объявлении кнопки в процедуре **«void setup()»** необходимо написать:

pinMode(номер пина, INPUT_PULLUP)

Для того чтобы прочитать значение с кнопки, необходимо воспользоваться функцией **«digitalRead()»**.

Нажатие на кнопку предусматривает совершение какого-либо действия после него. Чтобы задать выполнение такого действия, необходимо использовать функцию **«if()»** – «если значение в скобках истина, то выполнить …».

Нереверсивная схема управления двигателем

Разберем работу функции **«if()»** на примере нереверсивной схемы управления двигателем.

```
#define BUTTON_ONE 4  // объявляем пин кнопки Пуск
#define BUTTON_TWO 2  // объявляем пин кнопки Стоп
#define KM_ONE 5 // объявляем пин, к которому подключен двигатель
boolean keyPuskUp = false;  // флаг «нажата ли кнопка Пуск»

void setup()
{
// объявляем кнопку Пуск как вход с подтяжкой
pinMode(BUTTON_ONE, INPUT_PULLUP);
// объявляем кнопку Стоп как вход с подтяжкой
pinMode(BUTTON_TWO, INPUT_PULLUP);
// объявляемпин 5 выходом
pinMode(KM_ONE, OUTPUT);
}

void loop()
{
/* объявляем переменную keyPUSK, в которую записывается состояние кнопки Пуск*/
boolean keyPUSK = digitalRead(BUTTON_ONE);
/* объявляем переменную keySTOP, в которую записывается состояние кнопки Стоп */
boolean keySTOP = digitalRead(BUTTON_TWO);

if(!keyPUSK)  // если кнопка Пуск нажата, то
{
keyPuskUp = true;  // устанавливаем флаг «нажата ли кнопка Пуск»
digitalWrite(KM_ONE, HIGH);  // включаем двигатель
}
if(!keySTOP)  // если кнопка Стоп нажата, то
```

```
{
keyPuskUp = false;  // снимаем флаг
digitalWrite(KM_ONE, LOW);  // выключаем двигатель
}
}
```

Для того чтобы не объявлять номер пина в каждой функции по нескольку раз, удобнее объявить пин заранее и записать его в переменную, что и сделано в первых трех строчках кода с помощью выражения **#define**. То есть мы с самого начала прописали наши пины как константы и присвоили им имена: **BUTTON_ONE**, **BUTTON_TWO** и **KM_ONE**. Имена пинов можно использовать любые, но желательнее для эстетичности кода записывать их большими буквами. После имени идет номер пина, к которому подключены наши устройства (кнопки и пускатель). Обратите внимание, что при объявлении пинов с помощью выражения **#define** в конце строки ставить «;» нет необходимости. Объявление **#define** необходимо делать в самом начале программы перед процедурой **«void setup()»**. Также можно объявить любую переменную и задать ей начальное значение. Для этого вначале строки необходимо указать тип данных этой переменной, указать ее имя и через знак равенства присвоить ей необходимое значение. Так, например, мы объявляем переменную **keyPuskUp,** которая имеет булевый тип данных (два состояния: ложь −false или истина −true), что и говорит запись boolean перед переменной. Помимо булевых переменных используются также и другие типы данных. Наиболее распространенные их них приведены в таблице 1.

Таблица 5.1 – Типы данных

char	Переменная типа char хранит один алфавитно-цифровой символ. При объявлении символ записывают в одиночных кавычках: 'A' .
byte	Тип данных byte 8-ми битное беззнаковое целое число, в диапазоне 0..255.
int	Тип данных int (целое число). int занимает 2 байта памяти и может хранить числа от -32 768 до 32 767.
word	Тип данных word хранит 16-битное, не содержащее знака, число

	от 0 до 65535.
long	Тип данных long используется для хранения целых чисел в расширенном диапазоне от -2,147,483,648 до 2,147,483,647.
Unsigned long	Unsigned long используется для хранения положительных целых чисел в диапазоне от 0 до 4,294,967,295.
float	Тип данных float служит для хранения чисел с плавающей запятой. Этот тип часто используется для операций с данными, считываемыми с аналоговых входов. Диапазон значений — от -3.4028235E+38 до 3.4028235E+38.
void	Ключевое слово void используется при объявлении процедуры, если она не возвращает никакого значения при ее вызове. Помимо процедур setup и loop пользователь может объявлять и свои, в которых будет прописан тот или иной код.

Переменная **keyPuskUp** будет являться так называемым флагом, то есть переменной, которая говорит о состоянии процесса. В данном случаи о состоянии кнопки Пуск, нажата она или нет. Если нажата, то ей присваивается состояние истина – true. В первоначальный момент времени работы программы она не нажата, поэтому ей присвоено значение ложь – false.

В процедуре **«void setup()»** пины с именами **BUTTON_ONE** и **BUTTON_TWO,** к которым подключены кнопки, объявлены как входы с подтяжкой, а пин с именем **KM_ONE** как выход.

В процедуре **«void loop()»** вначале объявляем две булевых переменных **keyPUSK** и **keySTOP,** в которых мы записываем состояние кнопок Пуск и Стоп с помощью функции **«digitalRead()»**.

Далее стоит условие **if(!keyPUSK)** – «если кнопка Пуск нажата». Восклицательный знак в С++ означает инверсию и ставится перед переменной, которую мы хотим инвертировать. В данном случае инверсия нам необходима, потому что мы используем схему с подтягивающими резисторами и при нажатии на кнопку питание на пин не поступает (то есть микроконтроллер видит это как логический ноль – ложь). Поэтому мы инвертируем состояние нажатия и как бы «обманываем» микроконтроллер, говоря ему, что питание на пине есть (логическая единица – истина).

Действие, которое необходимо выполнить при соблюдении условия **«if()»,** обязательно заключается в фигурные скобки. В них мы «поднимаем» флаг, устанавливая его в состояние true, и включаем двигатель KM_ONE с помощью функции **«digitalWrite()»**.

То же самое проделываем с кнопкой Стоп, но при выполнении условия снимаем флаг (переводим его в состояние false) и выключаем двигатель.

Примечание. В данном случае можно было не использовать прием с установкой флага. Но в более сложных программах очень удобно отслеживать состояние работы устройств с помощью флагов.

Реверсивная схема управления двигателем

Разберем теперь пример с реверсивной схемой управления.

```
// объявляем пин, к которому подключена кнопка Влево
#define BUTTON_ONE 4
// объявляем пин, к которому подключена кнопка Вправо
#define BUTTON_TWO 2
// объявляем пин, к которому подключена кнопка Стоп
#define BUTTON_THREE 1
// объявляем магнитный пускатель, отвечающий за вращение Влево
#define KM_ONE 5
// объявляем магнитный пускатель, отвечающий за вращение Вправо
#define KM_TWO 6
boolean keyLeftUp = false; // флаг «нажата ли кнопка Влево»
boolean keyRightUp = false; // флаг «нажата ли кнопка Вправо»

void setup()
{
pinMode(BUTTON_ONE, INPUT_PULLUP);
pinMode(BUTTON_TWO, INPUT_PULLUP);
```

```
pinMode(BUTTON_THREE, INPUT_PULLUP);
pinMode(KM_ONE, OUTPUT);
pinMode(KM_TWO, OUTPUT);
}

voidloop()
{
/* объявляем переменную keyLEFT, в которую записывается состояние кнопки Влево */
boolean keyLEFT = digitalRead(BUTTON_ONE);

/* объявляем переменную keyRIGHT, в которую записывается состояние кнопки Вправо */
boolean keyRIGHT = digitalRead(BUTTON_TWO);
/* объявляем переменную keySTOP, в которую записывается состояние кнопки Стоп */
boolean keySTOP = digitalRead(BUTTON_THREE);

/* Далее функция if() −  если кнопка Влево нажата, то выполнить алгоритм между фигурных скобок { }. Восклицательный знак − инверсия, необходима потому, что кнопка с подтяжкой) */
if(!keyLEFT)
{
keyLeftUp = true;  // устанавливаем флаг «Влево»
keyRightUp = false;  // оставляем опущенным флаг «Вправо»
digitalWrite(KM_ONE, HIGH);  // включаем двигатель «Влево»
digitalWrite(KM_TWO, LOW);  // блокируем двигатель «Вправо»
}

if(!keyRIGHT)  // если кнопка Вправо нажата, то
```

```
{
keyRightUp = true;  // устанавливаем флаг «Вправо»
keyLeftUp = false;  // опускаем флаг «Влево»
digitalWrite(KM_TWO, HIGH);  // включаем двигатель «Вправо»
digitalWrite(KM_ONE, LOW);  // блокируем двигатель «Влево»
}

if(!keySTOP)  // если кнопка Стоп нажата, то
{
keyLeftUp = false;  // снимаем флаг «Влево»
keyRightUp = false;  // снимаем флаг «Вправо»
digitalWrite(KM_ONE, LOW);  // блокируем двигатель «Влево»
digitalWrite(KM_TWO, LOW);  // блокируем двигатель «Вправо»
}
}
```

Как и в схеме с нереверсивным управлением, мы вначале объявили пины для кнопок (влево, вправо и стоп) и двух магнитных пускателей, а также указали два флага для двух состояний: влево и вправо. В процедуре **«void setup()»** указали, какой из пинов является выходом, а какой входом. В **«void loop()»** указали алгоритм действий при нажатии на каждую из кнопок с помощью функции **«if()»** [14].

Пускосигнальное звено

Рассмотрим пример пускосигнального звена.

```
#define BUTTON_ONE 4    // Пуск
#define BUTTON_TWO 2    // Стоп
#define KM_ONE 9  // магнитный пускатель двигателя
#define ZVONOK 3    // Звонок
/* таймер задержки на запуск (имеет тип данных long, как и все остальные временные переменные в данной программе) */
```

```
long prev_time = 5000;
long startTime;     // время нажатия на кнопкуПуск
/* переменная, в которую записывается общее время работы технологического процесса */
long currentTime;
boolean keyPuskUp = false;   // флаг "нажата ли кнопка Пуск"
/* переменная, в которую записывается время срабатывания магнитного пускателя и отключения звонка */
long time;

void setup()
{
pinMode(BUTTON_ONE, INPUT_PULLUP);
pinMode(BUTTON_TWO, INPUT_PULLUP);
pinMode(KM_ONE, OUTPUT);
pinMode(ZVONOK, OUTPUT);
}

voidloop()
{
currentTime = millis();  // записываем время начала выполнения программы
booleankeyPUSK = digitalRead(BUTTON_ONE);
booleankeySTOP = digitalRead(BUTTON_TWO);

if(!keyPUSK)  // если кнопка Пуск нажата, то
{
keyPuskUp = true;  // устанавливаем флаг
startTime = millis();  // записываем время нажатия на кнопку Пуск
digitalWrite(ZVONOK, HIGH); // включаем Звонок
}
```

```
if(keyPuskUp)  // если установлен флаг кнопки Пуск, то
{
start();  // выполнить код в процедуре voidstart()
}

if(!keySTOP)  // если кнопка Стоп нажата, то
{
digitalWrite(ZVONOK, LOW);  // выключить звонок
keyPuskUp = false;  // убрать флаг кнопки Пуск
digitalWrite(KM_ONE, LOW);  // выключить двигатель
}
}

void start()
{
/* в переменную time записываем время, когда должен выключиться звонок и включиться двигатель, то есть время, которое настанет через 5 секунд (prev_time = 5000) после времени нажатия на кнопку Пуск (startTime) */
time = startTime +  prev_time;
/* таким образом, когда общее время после начала выполнения программы превысит время, записанное в переменной time, произойдет отключение звонка и включение двигателя */
if(currentTime>= time)
{
digitalWrite(ZVONOK, LOW); // выключить звонок
digitalWrite(KM_ONE, HIGH); // включить двигатель
}
}
```

Для того чтобы задать задержку на включение двигателя, в программе можно было использовать функцию **«delay()»**. Но как уже ранее упоминалось функция **«delay()»** останавливает выполнение программы на указанный интервал времени. Поэтому если нам по какой-либо причине понадобиться выключить технологический процесс в момент времени, когда звенит звонок, программа не отреагирует на нажатие кнопки Стоп, так как она как бы «заснет» на время выполнения функции **«delay()»**. Для решения данной проблемы в программе была использована другая временная функция −**«millis()»**. Данная функция возвращает количество миллисекунд с момента начала выполнения программы. Это количество сбрасывается на ноль вследствие переполнения значения, приблизительно через 50 дней.

При запуске программы начинается запись времени работы программы в переменную **currentTime**. Считываются состояния кнопок Пуск и Стоп, как и в предыдущих программах. Если нажата кнопка Пуск, то в переменную **ketPuskUp** записываем состояние **true** (поднимаем флаг). Записываем время нажатия на кнопку Пуск с помощью присвоения переменной **startTime** значения функции **«millis()»** и включаем звонок.

Далее создаем условие с помощью функции **«if()»**: если флаг поднят (значение переменной **keyPuskUp** в скобках должно быть = true), то выполнить код в процедуре **«void start()»**. Таким образом, на данном примере показано, что, программист может сам вводить свои процедуры (название может быть любым, для наглядности мы назвали нашу процедуру **start**). Можно было бы и не вводить новую процедуру, а выполнить те же самые действия в условии **if(keyPuskUp)**, но в более сложных программах удобнее разделять текст программы на процедуры, что позволяет легче отслеживать ошибки при ее наладке.

В процедуре **«void start()»** мы создаем отчет времени с помощью переменной **time,** в которую записываем время, когда должен отключиться звонок и включиться двигатель. Сравниваем это время с временем с момента запус-

ка программы (**currentTime**) и, как только последнее превысит значение первого, отключаем звонок и включаем двигатель.
При нажатии на кнопку Стоп отключаем звонок и двигатель, а также опускаем флаг.

Работа с потенциометрами

Помимо управления различными устройствами, платформа Arduino позволяет подключать различные цифровые и аналоговые датчики. Подобных совместимых с Arduino датчиков великое множество. Разберем пример подключения потенциометра. Схема подключения потенциометров к платформе показана на рисунке 5.7. В схеме стоят ограничивающие резисторы R1 ... 6, для того чтобы ограничить напряжение от источника питания с 12В до 5В. Подача большего питания, чем 5В, на аналоговые пины A0 ... 5 грозит выходом их из строя.

Для того чтобы объявить подключенный к пинуA0 потенциометр, необходимо вначале программы его прописать с помощью уже известного нам выражения **#define**. Например:

```
#define POT_NULL A0
```

Вместо POT_NULL может быть любое другое название, главное что теперь везде при встрече в коде записи POT_NULL наш микроконтроллер будет знать, что мы имеем ввиду потенциометр, подключенный к пину A0.

В процедуре **«void setup()»** аналоговые пины как входы можно не объявлять, так как они таковыми считаются по умолчанию.

Далее в процедуре «voidloop()» для чтения показаний потенциометра необходимо ввести переменную, например **potnull,** и присвоить ей показания потенциометра, которые поступают на пин A0 с помощью функции **«analogRead()»**. Выглядеть это будет, например, так:

```
int potnull = analogRead(POT_NULL);
```

То есть в скобках функции **«analogRead()»** прописывается имя того пина, с которого мы читаем значение. В данном случае под **POT_NULL** мы записали пин A0 в самом начале программы.

Функция **«analogRead()»** возвращает целочисленные значения (именно поэтому переменная **potnull** имеет тип данный int) в диапазоне от 0 до 1023. Таким образом, в зависимости от поворота ручки потенциометра в переменную potnull будут записываться числа 0 до 1023. Опираясь на эти значения в коде программы, можно реализовывать различные действия, например, создавать уставки срабатывания тех или иных механизмов по достижению определенного значения в переменной potnull.

Широтно-импульсная модуляция (ШИМ)

Платформа Arduino не может выдавать на свои пины произвольное напряжение. Только 5 В либо 0В. Поэтому, для того чтобы управлять частотой вращения двигателей или яркостью освещения, необходимо использовать ШИМ − процесс получения изменяющегося аналогового значения посредством цифровых устройств. Иными словами ШИМ − это эмуляция аналогового сигнала, для которой мы подаем на пины сигналы с напряжением 5В с различной частотой в зависимости от того, какую яркость или частоту вращения хотим получить (рис. 5.8) [23].

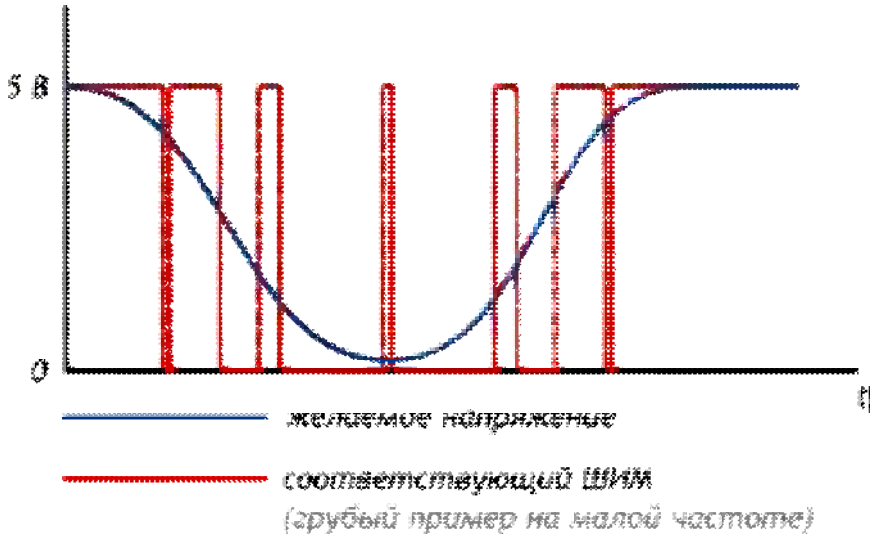


Рисунок 5.8 – Рисунок широтно-импульсной модуляции.

При ШИМ наш микроконтроллер переключается между землей и питанием тысячи раз в секунду. Поэтому, например, подключенный к платформе светодиод будет мерцать с частотой этих переключений, но наблюдающему человеку будет казаться, что он горит не в полную силу.

Таким образом, изменяя отношение времени включения и отключения, мы можем управлять яркостью освещения, частотой вращения двигателей, звукового сигнала и т.д. Это отношение называют скважностью (рис. 5.9).

Номера пинов, поддерживающих ШИМ на платформе Arduino UNO: 3, 5, 6, 9, 10, 11. Для получения ШИМ на любом из этих пинов необходимо использовать функцию **«analogWrite()»**. Например, подключив к 5 пину платформы светодиодную ленту, мы хотим, чтобы она горела в полсилы:

```
analogWrite(5, 127);
```

Первое число в скобках − номер пина, второе отвечает за скважность и задается в диапазоне от 0 до 255. То есть 127 − это эквивалент 2,5В (50%) и наша светодиодная лента будет гореть в полсилы.

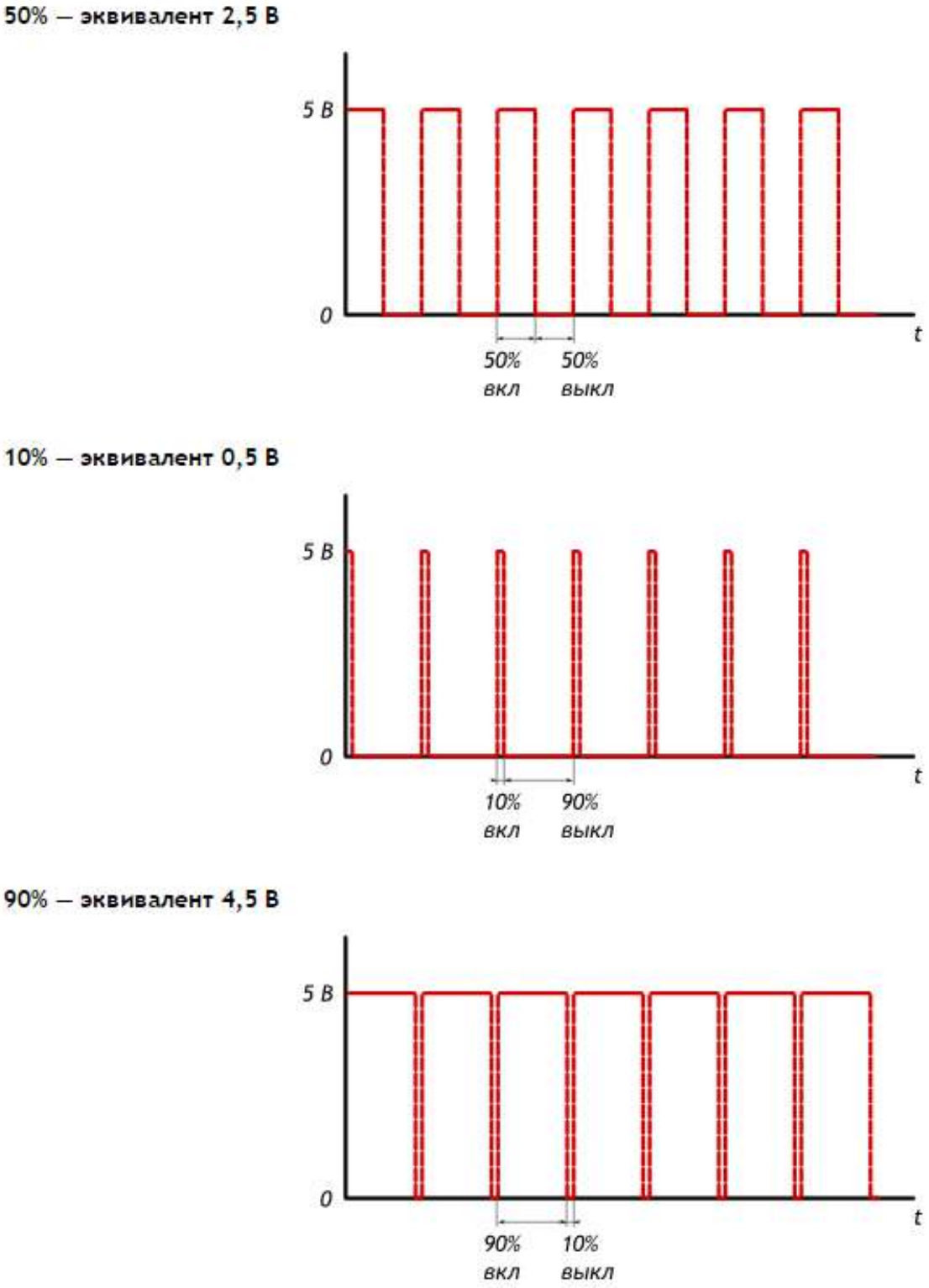


Рисунок 5.9 – Рисунок скважности.

Также яркостью светодиодной ленты или скоростью вращения вентиляторов можно управлять с помощью потенциометров. Рассмотрим это на примере из пункта 5.2.5:

```
int potnull = analogRead(POT_NULL);
int shim = potnull / 4;
int light = analogWrite(5, shim);
```

Во второй строке мы разделили показания потенциометра на 4 и таким образом привели их из диапазона 0−1023 (см. п. 2.4) к диапазону 0−255. Полученное число записываем в функцию **«analogWrite()»**. Таким образом, поворачивая ручку потенциометра, мы сможем регулировать яркость светодиодной ленты.

Запись информации на текстовый экран

К платформе Arduino можно подключать текстовые экраны и записывать на них информацию. Рассмотрим на примере экрана фирмы Мэлт MT-16S2H (рис. 5.10).

Схема подключения данного экрана представлена на рисунке 5.12.

Рисунок 5.10 – Рисунок текстового экрана MT-16S2H.

Для работы с экраном проще всего воспользоваться уже готовой библиотекой. Библиотеки − это готовый набор функций, которые облегчают пользователю работу по программированию, особенно когда необходимо управлять экранами, сервоприводами, различными датчиками. Для работы с

экраном в Arduino IDE существует стандартная библиотека Liquid Crystal (рис. 5.11).

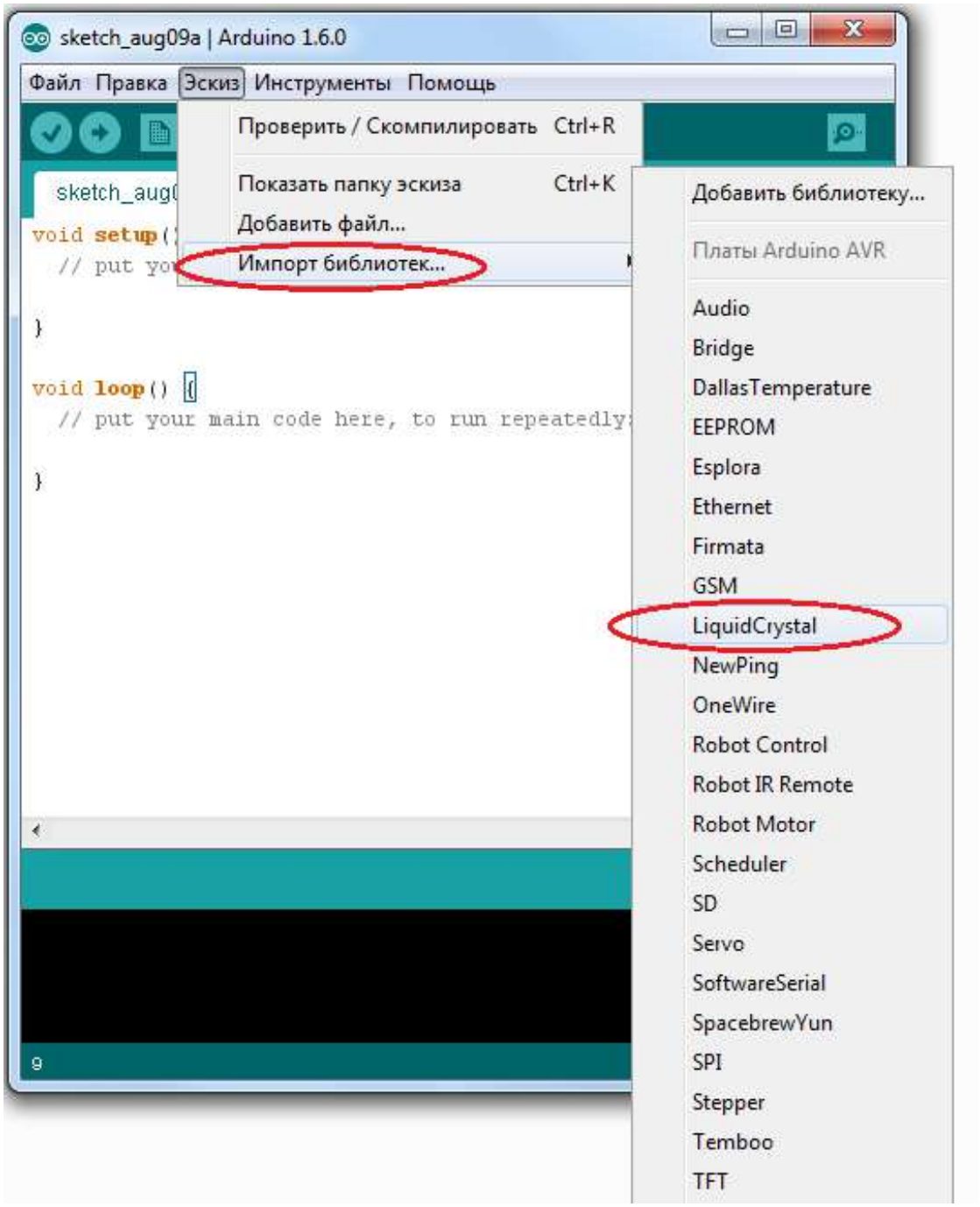


Рисунок 5.11− Библиотека Liquid Crystal.

При ее выборе в пункте меню в начале программы появится строчка

#include<LiquidCrystal.h>

Это значит, что вы успешно добавили библиотеку в программу. Далее можно воспользоваться следующим кодом:

```
#include<LiquidCrystal.h>
/* Инициализируем объект-экран, передаем использованные для подключения контакты на Arduino в необходимом порядке (согласно схеме подключения на рисунке 5.12*)/
LiquidCrystallcd(7, 8, 10, 11, 12, 13);

void setup()
{
/* устанавливаем размер (количество столбцов и строк) экрана, в данном экране две строки по 16 символов */
lcd.begin(16, 2);
// печатаем первую строку
lcd.print("Helloworld!");
/* устанавливаем курсор в колонку 0, строку 1. То есть на
самом деле это вторая строка, т.к. нумерация начинается с нуля */
lcd.setCursor(0, 1);
// печатаемвторуюстроку
lcd.print("How are you?");
}

void loop()
{
}
```

Как видно из кода программы, все действия выполняются в процедуре **«void setup()»**. В результате на экране должна появиться надпись:

Hello World!
How are you?

Для вывода русских букв необходимо знать таблицу символов с кодами кодировки, которую можно найти в документации к экрану (см. приложение 3 и 4). Например, букве Ц соответствует код E1, для ее вывода необходимо написать этот код после последовательности **x**:

```
lcd.print("\xE1");
```

Надпись в первой строке «Привет» будет выглядеть так:

```
lcd.print("\xA8\x70\xB8\xB3\x65\xBF");
```

5.3 Пример выполнения схемы автоматизации линии предварительной обработки зерна на базе платформы Arduino

Технологическая схема примера уже была подробно рассмотрена ранее. Реализация примера выполняется на лабораторном стенде (рис. 5.12). Стенд содержит:

1. Предохранитель
2. Кнопку питания SA1
3. Блок питания 12В
4. 4 кнопки SB1 … SB4
5. Текстовый экран MT-16S2H
6. 4 силовых ключа
7. 4 вентилятора M1 … M4
8. 1 светодиодный светильник HL1
9. 6 потенциометров
10. ПлатформуArduino UNO

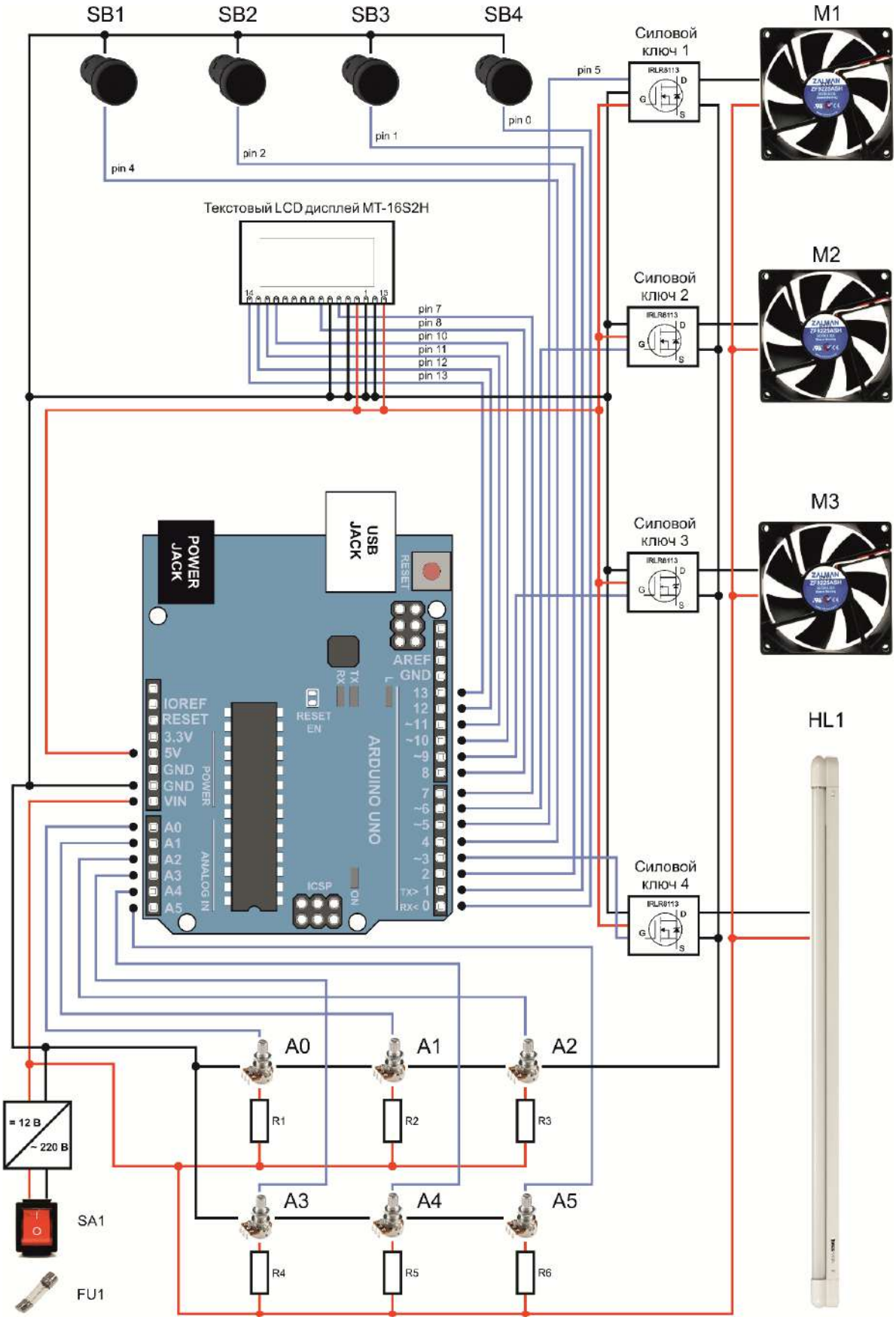


Рисунок 5.12 – Внешний вид лабораторного стенда.

Код для реализации технологического процесса:

```
#include <LiquidCrystal.h>

LiquidCrystallcd(7, 8, 10, 11, 12, 13);

#define BUTTON_ONE 4    // Пуск
#define BUTTON_TWO 2    // Стоп
#define BUTTON_THREE 1  // Рабочий стоп
#define KM_ONE 5    // Третий механизм
#define KM_TWO 6    // Второй механизм
#define KM_THREE 9  // Головной механизм
#define ZVONOK 3    // Звонок
long prev_time_1 = 5000; // Таймер задержки 1 на запуск
long prev_time_2 = 5000; // Таймер задержки 2 на запуск
long prev_time_3 = 5000; // Таймер задержки 3 на запуск
long prev_time_4 = 5000; // Таймер задержки 4 на рабочий стоп
long prev_time_5 = 5000; // Таймер задержки 5 на рабочий стоп
long startTime;     // время нажатия на кнопку Пуск
long startTime_1;   // время нажатия на кнопку Рабочий стоп
long currentTime;   // время работы технологического процесса
boolean keyPuskUp = false;   // Флаг "нажата ли кнопка Пуск"
boolean keyWorkStopUp = false;  // Флаг "нажата ли кнопка Рабочий стоп"
long time_1;  // время включения KM_ONE
long time_2;  // времявключения KM_TWO
long time_3;  // времявключения KM_THREE
long time_4;  // времявыключения KM_THREE
long time_5;  // времявыключения KM_TWO

void setup()
{
```

```
lcd.begin(16, 2);
lcd.print("Nikolaenko");
lcd.setCursor(0, 1);
lcd.print("Cokur");

pinMode(BUTTON_ONE, INPUT_PULLUP);
pinMode(BUTTON_TWO, INPUT_PULLUP);
pinMode(BUTTON_THREE, INPUT_PULLUP);
pinMode(KM_ONE, OUTPUT);
pinMode(KM_TWO, OUTPUT);
pinMode(KM_THREE, OUTPUT);
pinMode(ZVONOK, OUTPUT);
}

void loop()
{
currentTime = millis();
boolean keyPUSK = digitalRead(BUTTON_ONE);
boolean keySTOP = digitalRead(BUTTON_TWO);
boolean keyWORKSTOP = digitalRead(BUTTON_THREE);

if(!keyPUSK)
{
keyPuskUp = true;
keyWorkStopUp = false;
startTime = millis();  // записываем время нажатия накнопк Пуск
digitalWrite(ZVONOK, HIGH); // включаем Звонок
}

if(keyPuskUp)  // если установлен флаг кнопки Пуск
```

```
{
start();
}

if(!keySTOP)
{
digitalWrite(ZVONOK, LOW);
keyPuskUp = false;
fullstop();
}

if(!keyWORKSTOP)
{
keyPuskUp = false;
digitalWrite(KM_THREE, LOW);
startTime_1 = millis();
keyWorkStopUp = true;
}

if(keyWorkStopUp)
{
workstop();
}
}

void start()
{
time_1 = startTime+  prev_time_1;
if(currentTime>= time_1)
{
```

```
digitalWrite(ZVONOK, LOW);
digitalWrite(KM_ONE, HIGH);
}

time_2 = time_1 +  prev_time_2;
if(currentTime>= time_2)
{
digitalWrite(KM_TWO, HIGH);
}

time_3 = time_2 +  prev_time_3;
if(currentTime>= time_3)
{
digitalWrite(KM_THREE, HIGH);
}
}

void fullstop()
{
digitalWrite(KM_ONE, LOW);
digitalWrite(KM_TWO, LOW);
digitalWrite(KM_THREE, LOW);
}

void workstop()
{
time_4 = startTime_1 +  prev_time_4;
if(currentTime>= time_4)
{
digitalWrite(KM_TWO, LOW);
```

```
}

time_5 = time_4 +  prev_time_5;
if(currentTime>= time_5)
{
digitalWrite(KM_ONE, LOW);
}
}
```

Приложение 1

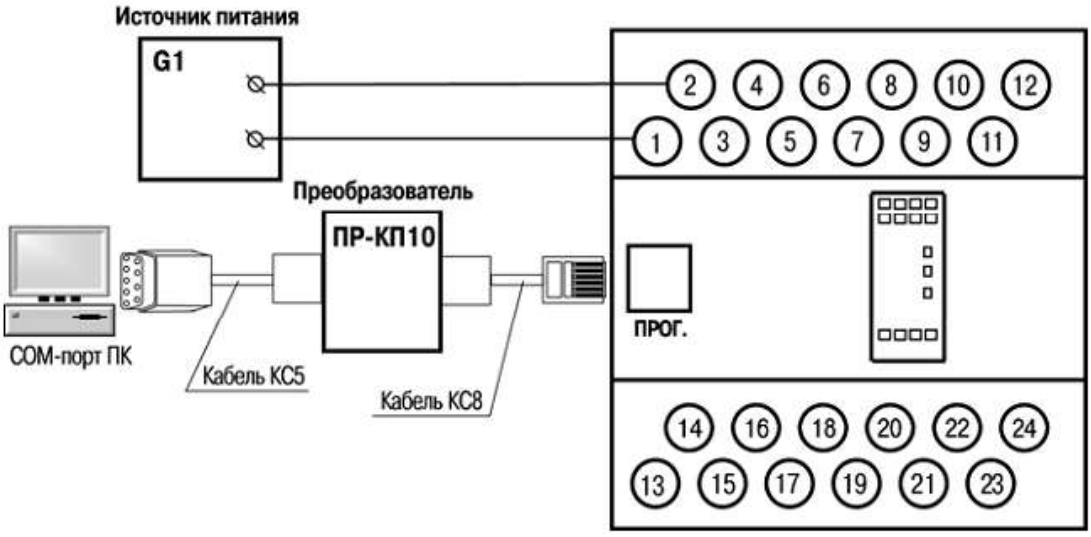


Рисунок 1 − Схема подключения программируемого реле ПР114 (через преобразователь ПР-КП10), G1 − источник питания с номинальным напряжением, зависящим от исполнения прибора.

Приложение 2

Настройка подключения ПР к компьютеру.

Есть два способа подключения ПР к вашему компьютеру. Вы можете использовать комплект ПР-КП10, если ваш компьютер располагает Com-портом. В этом случае вам просто необходимо будет указать в настройках OWEN Logic номер этого порта.

Вероятнее всего, вы все-таки выберете вариант подключения через USB. Такой порт есть практически у любого современного компьютера, включая ноутбуки. В этом случае вам необходимо использовать комплект ПР-КП20 и провести для него несколько настроек.

Драйвер для работы ПР-КП20 поставляется в комплекте с лабораторным стендом. В противном же случае необходимо скачать его с официального сайта изготовителя. Для этого вам необходимо перейти на сайт компании ОВЕН по следующей ссылке.

http://www.owen.ru/catalog/36879114

На этой странице вам необходимо скачать драйвер для работы ПР-КП20 с вашим компьютером. Найдите ссылку «Драйвер ПР-КП20», загрузите файл на ваш компьютер. После того, как вы разархивируете его, откройте папку и запустите программу установки (см. *рис. Б.1*).

Установку необходимо производить при отключенном от компьютера ПР-КП20.

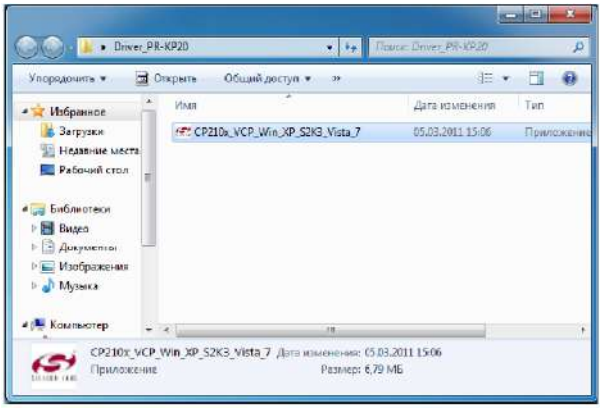


Рисунок Б.1 – Установка драйвера ПР-КП20 на компьютер.

В последующих трех окнах (*рис.Б.2*) просто нажимайте кнопку «Next». Таким образом, вы соглашаетесь на обычные настройки по умолчанию. В окне, изображенном на *рис.Б.3*, необходимо нажать кнопку «Install». После этого начнется установка, необходимо немного подождать.

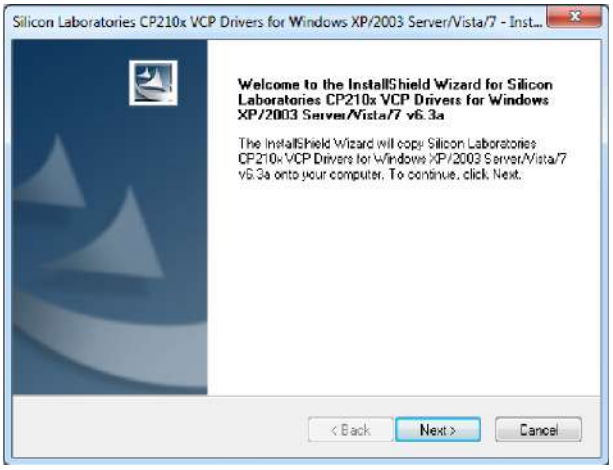


Рисунок Б.2 – Установка драйвера ПР-КП20 на компьютер.

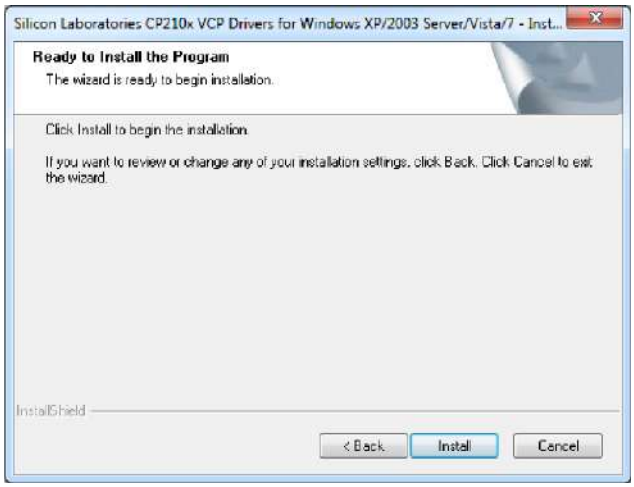


Рисунок Б.3 – Установка драйвера ПР-КП20 на компьютер.

После окончания установки в последнем окне полезно поставить галочку, как показано на *рис.Б.4*. Затем можно нажать кнопку «Finish».

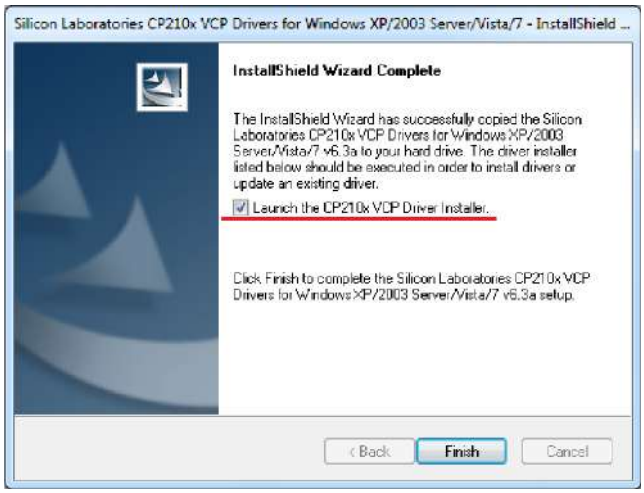


Рисунок Б.4 – Установка драйвера ПР-КП20 на компьютер.

Откроется окно добавления драйвера на ваш компьютер (*рис.Б.5*). Здесь также нажимаем кнопку «Install» и ждем.

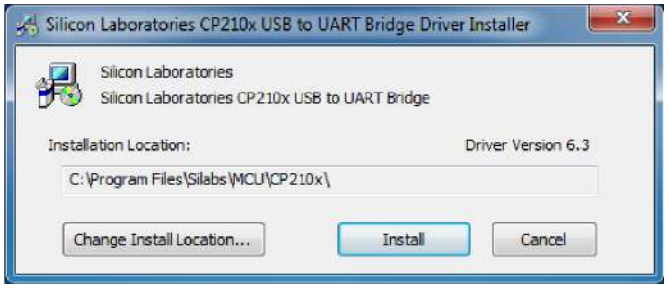


Рисунок Б.5 – Установка драйвера ПР-КП20 на компьютер.

После установки драйвера система предложит вам перезагрузить ваш компьютер (*рис.Б.6*). Если вам необходимо сделать сохранения в других открытых программах, нажмите «Перезагрузить позже». Проделайте все необходимые действия для завершения работы и перезагрузите компьютер вручную.

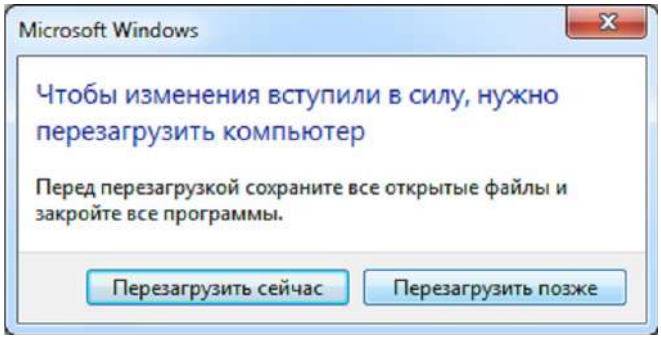


Рисунок Б.6 – Установка драйвера ПР-КП20 на компьютер.

После перезагрузки вам необходимо уточнить, какой номер порта получил ваш ПР-КП20. На рабочем столе откройте «Компьютер». Нажмите кнопку «Свойства системы» (см. *рис.Б.7*). Другой способ: нажать кнопку «Пуск» ( ), открыть «Панель управления» и выбрать «Система».

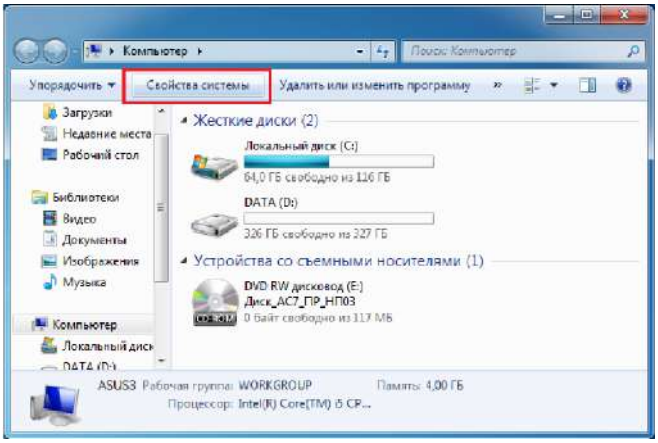


Рисунок Б.7 – Установка драйвера ПР-КП20 на компьютер.

В открывшемся окне (*рис.Б.8*) выберете слева пункт «Диспетчер устройств».

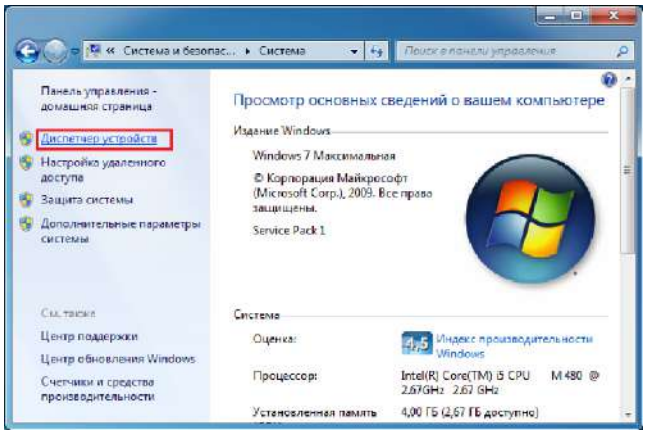


Рисунок Б.8 – Установка драйвера ПР-КП20 на компьютер.

На *рис.Б.9* показано, как операционная система определяет ПР-КП20 без предварительно установленного драйвера. Подключите Ваш ПР-КП20 к компьютеру. Система должна автоматически его определить и выдать сообщение (*рис.Б.10*).

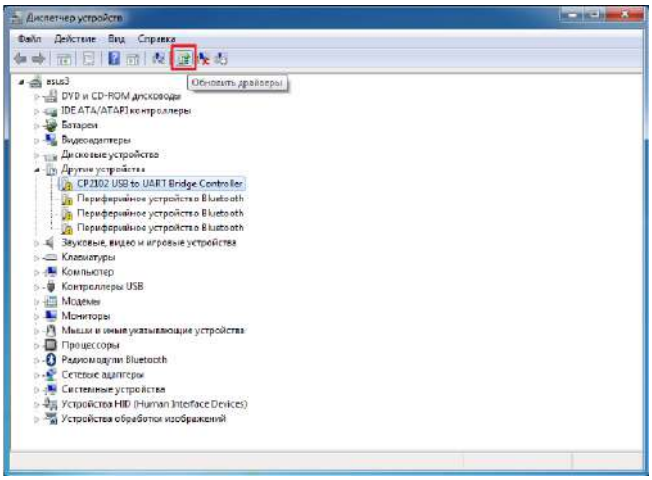


Рисунок Б.9 – Установка драйвера ПР-КП20 на компьютер.

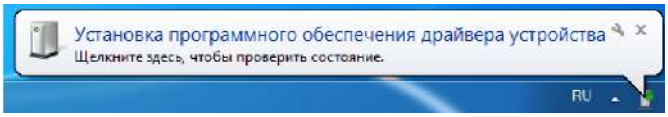


Рисунок Б.10 –Установка драйвера ПР-КП20 на компьютер.

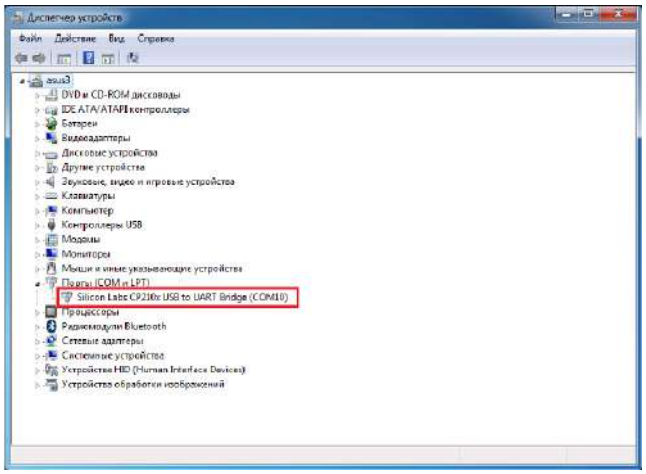
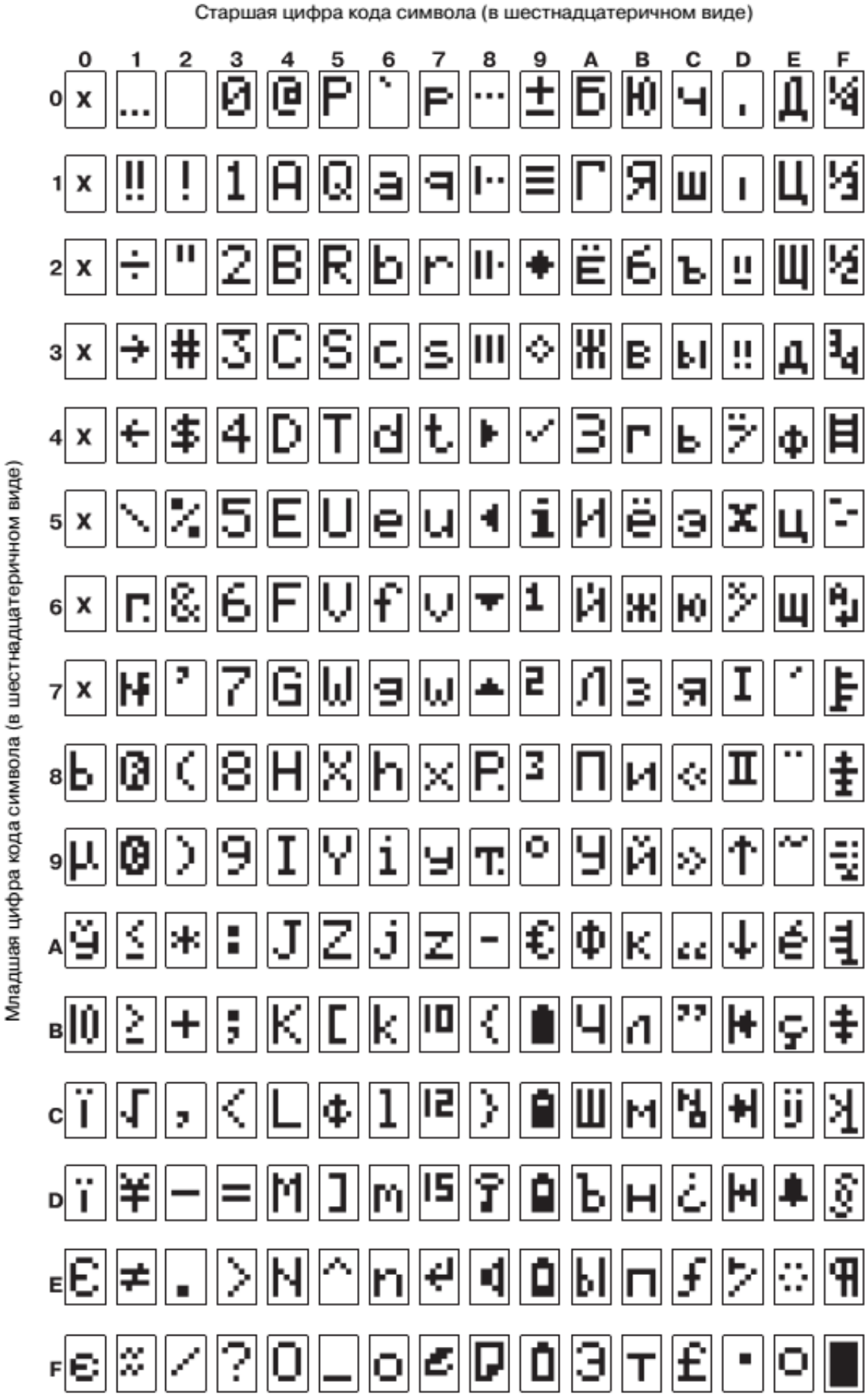


Рисунок Б.11 –Установка драйвера ПР-КП20 на компьютер.

После успешного определения системой ПР-КП20 в Диспетчере устройств оно будет отображаться так, как показано на *рис.Б.11.* ПР-КП также будет присвоен номер порта. Например, Com10, как в рассматриваемом здесь случае. Этот номер порта мы будем использовать при настройке связи с нашим программируемым реле. На этом этот технический этап настройки закончен.

Приложение 3 - Страница 0 встроенного знакогенератора



Приложение 4 - Страница 1 встроенного знакогенератора

СПИСОК ЛИТЕРАТУРЫ

1. Бородин И.Ф., Кирилин И.И. Основы автоматики и автоматизации производственных процессов. - М.: Колос, 1987.
2. Карташов Б.А., Привалов А.С., Самойленко В.В., Татамиров Н.И. Компьютерные технологии и микропроцессорные средства в автоматическом управлении. Под ред. Б.А. Карташова. - Ростов н/Д: Феникс, 2013. - 540 с: ил.
3. Николаенко С.А. Исследования влияния параметров электроозонирования на выживаемость тест-микроорганизмов / С.А. Николаенко, Д.С. Цокур // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета (Научный журнал КубГАУ) [Электронный ресурс]. – Краснодар: КубГАУ, 2014. – №09(103). С. 737 – 752. – IDA [article ID]: 1031409045. – Режим доступа: http://ej.kubagro.ru/2014/09/pdf/45.pdf, 1 у.п.л.
4. Оськин С.В. Автоматизация технологических процессов: методические указания по выполнению контрольных работ для студентов з/о обучения / С.В. Оськин, Д.А. Овсянников, Д.П. Харченко, С.А. Николаенко, В.А. Дидыч. – Краснодар, РИО КубГАУ, 2009. – 42 с.: ил.
5. Оськин С.В. Автоматизированные системы управления технологическими процессами. Часть 1. Языки программирования. Лабораторный практикум /С.В. Оськин, Д.А. Овсянников, С.А. Николаенко, В.А. Дидыч, М.А. Карпов. – Краснодар, РИО КубГАУ, 2010. – 32 с.: ил.
6. Оськин С.В. Автоматизированный электропривод / С.В. Оськин, Н.И. Богатырев, С.М. Моргун. – Краснодар, ОАО «Кубанское полиграфическое объединение», 2014. – 212 с.: ил.
7. Николаенко С.А. Учебное пособие для выполнения лабораторных работ по дисциплине «Автоматика» для студентов по направлению «Агроинженерия» / С.А. Николаенко, Д.С. Цокур, А.П. Волошин. – Краснодар, РИО КубГАУ, 2014. – 99 с.: ил.
8. Оськин С.В. Автоматизированный электропривод: учеб. пособие / С.В. Оськин. - Краснодар: Изд-во ООО « Крон», 2015. - 489 с.
9. Оськин С.В. Автоматика. Практикум по выполнению курсовой работы «Анализ и синтез линейной системы автоматического регулирования» / С.В. Оськин, Д.А. Овсянников, Д.П. Харченко, С.А. Николаенко, В.А. Дидыч. – Краснодар, РИО КубГАУ, 2009. – 32 с.: ил.
10. Оськин С.В. Компьютерное моделирование систем автоматического управления: практикум / С.В. Оськин, Д.А. Овсянников, С.А. Николаенко, А.П. Волошин. – Краснодар, РИО КубГАУ, 2010. – 43 с.: ил.
11. Оськин С.В. Лабораторный практикум по дисциплине «Автоматизация технологических процессов» Часть 1 /С.В, Оськин, С.А, Николаенко, А.П. Волошин, Д.С. Цокур. – Краснодар, РИО КубГАУ, 2013. – 87 с.: ил.

12. Оськин С.В. Программируемое реле EASY-719: учебн. Пособие / С.В. Оськин, Д.А. Овсянников, С.А. Николаенко, А.П. Волошин. – Краснодар, РИО КубГАУ, 2010. – 40 с.: ил.
13. Николаенко С.А. Исследование качества регулирования концентрации озона в улье с использованием системы стабилизированного электроозонирования пчелиных семей / С.А. Николаенко // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета (Научный журнал КубГАУ) [Электронный ресурс]. – IDA [article ID]: 1031409046. – Краснодар: КубГАУ, 2014. – № 09(103).
14. Николаенко С.А. Повышение продуктивности пчеловодства / С.А. Николаенко, Д.С. Цокур // «Механизация и электрификация сельского хозяйства», №10 – 2015. – С. 13-16.
15. Харченко Д.П. Схемотехника. Внутреннее устройство и программирование PIC-микроконтроллеров: учебн. Пособие / Д.П. Харченко, С.А. Николаенко, Д.С. Цокур, А.П. Волошин. – Краснодар, РИО КубГАУ, 2014. – 98 с.: ил.
16. Николаенко С.А. Автоматизация систем управления: учебн. Пособие / С.А. Николаенко, Д.С. Цокур. – Краснодар, РИО КубГАУ, 2015. – 120 с.: ил.
17. http://sesaga.ru/naznachenie-ustrojstvo-i-rabota-magnitnogo-puskatelya.html
18. http://kipservis.ru/delta/logicheskie_kontrollery_programmiruemye_dvp_ss.htm
19. http://www.owen.ru/catalog/programmiruemoe_rele_pr114/opisanie
20. http://www.owen.ru/catalog/programmiruemij_logicheskij_kontroller_oven_plk160/opisanie
21. http://dfpd.siemens.ru/products/automation/simatic/logo/
22. http://www.elec.ru/viewer?url=/files/127/000000156/attfile/02.pdf
23. http://arduino.ru/Hardware/ArduinoBoardUno

Учебное издание

Николаенко Сергей Анатольевич

Цокур Дмитрий Сергеевич

Харченко Дмитрий Павлович

Волошин Александр Петрович

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

Учебное пособие

Подписано в печать 01.09.2016 г.

Формат 60х84 . Усл. печ. л. 14,7

Тираж 200 экз.

Заказ 108

Типография ООО «Крон»

350044, г. Краснодар, ул. Алма-Атинская 57, оф.3