МИНИСТЕРСТВО НАУКИ И ВЫСЩЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ имени
В.И.ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

Сосновский Ю.В.

Учебно-методическое пособие по проведению практических занятий по курсу «Автоматизированные системы на встроенных контроллерах»

для студентов
направления подготовки 09.03.01 «Информатика и вычислительная техника»
образовательно-квалификационного уровня «бакалавр»

Симферополь, 2020

Рекомендовано к печати заседанием кафедры

от 15.01.2020 г. № _5__

Рекомендовано к печати:

Учебно-методическим советом Физико-технического института (структурное подразделение) ФГАОУ ВО «КФУ им. В.И.Вернадского»

протокол № __5__ от 24.01.2020

Составитель (автор): Сосновский Юрий Вячеславович, к.т.н., доцент кафедры компьютерной инженерии и моделирования

Введение

Автоматизация объектов и процессов в последнее десятилетие перешла из сугубо технологического использования в пользовательскую сферу применения. На сегодняшний день существует большое число автоматизированных систем управления в промышленности, сельском хозяйстве, производстве и даже в быту. При этом сложность подобных систем постоянно растет, повышаются требования к обслуживающему персоналу и создателям систем.

Изучение дисциплины «Автоматизированные системы на встроенных контроллерах» позволяет дать студентам базовые знания по современным технологиям, применяемым в системах управлении сложными объектами и процессами, дать практические навыки по использованию языков программирования стандарта МЭК 61131.

Данное методическое пособие не является исчерпывающим источником всех необходимых сведений, а дает лишь требуемый минимум материала для выполнения заданий. Пособие подразумевает самостоятельную работу студента со справочными материалами по стандарту МЭК 61131-3, с руководством по программированию в системе CODESYS v.2 и другими материалами, которые доступны в электронном виде и являются частью методического обеспечения курса «Автоматизированные системы на встроенных контроллерах».

Список условных обозначений

АСУП – автоматизированная система управления производством
АСУТП – автоматизированная система управления технологическим процессом
ЛКМ – левая кнопка мыши
ОПЛК – ПЛК со встроенной операторской панелью
ОС – операционная система
ПКН – правая кнопка мыши
ПЛК – программируемый логический контроллер, промышленный компьютер
ИАС – интегрированная автоматизированная система
ИП – информационное поле
ЧМИ – человеко-машинный интерфейс
FBD – аббр. от англ. *Function Block Diagram,* функциональные блоковые диаграммы
IL – аббр. от англ. *Instruction List,* список инструкций
LD – аббр. от англ. *Ladder Diagram*, лестничная диаграмма или релейно-контактная схема
POU – аббр. от англ. *Program Unit*, программный модуль
SCADA-система – аббр. от англ. *supervisory control and data acquisition*, диспетчерское управление и сбор данных
SFC – аббр. от англ. *Sequential Function Chart,* последовательные функциональные диаграммы
ST – аббр. от англ. *Structured Text,* структурированный текст

Содержание

Системы управления сложными процессами

Чаще всего под системами управления сложными процессами подразумевают автоматизированные системы управления, которые применяются в производственном процессе даже малых предприятий, а также активно проникают в бытовую сферу как т.н. системы «умного дома».

Автоматизированная система управления технологическим процессом (АСУТП) – система, имеющая непосредственное отношение к управлению теми устройствами, которые обеспечивают выполнение технологического процесса. В тоже время, на современном этапе развития автоматизации АСУТП принято интегрировать в систему более высокого уровня – интегрированную автоматизированную систему (ИАС).

Интегрированная автоматизированная система [1] – совокупность двух или более информационных систем, в которых функционирование одной зависит от результатов функционирования другой. Таким образом, данные системы совместно следует рассматривать как единую информационную систему.

Таким образом, ИАСУ является результатом интеграции АСУТП и АСУП (автоматизированной систему управления производством).

В зарубежной литературе встречаются термины «компьютерно-интегрированное производство» и «компьютерно-интегрированный процесс», отражающий интеграцию ИАС и остального информационного поля предприятия в единую систему посредством локальной компьютерной сети.

Аппаратно-программные средства ИАС

Интегрированная автоматизированная система кроме типовых рабочих станций может включать в свой состав ряд специфических аппаратных средств, таких как:

- средства человеко-машинного интерфейса (ЧМИ, *Human Machine Interface – HMI*);

- программируемые логические контроллеры (ПЛК, Programming Logic Controller – PLC);
- исполнительные механизмы и датчики.

А также ряд специальных программных средств, таких как:

- специализированная система сбора данных и управления в реальном режиме времени – SCADA-система (аббр. от англ. *supervisory control and data acquisition*, диспетчерское управление и сбор данных);
- специализированные СУБД.

В простом случае в виде элементов человеко-машинного интерфейса могут выступать индикаторные лампы, цифровые индикаторы и органы управления – кнопки, джойстики. Широкое применение находят т.н. панели оператора – специализированные дисплеи, выводящие информацию о технологическом процессе в удобной для оператора форме. Могут иметь дополнительные кнопки для задания параметров, управления. Все чаще встречаются в исполнении с сенсорным экраном, что позволяет использовать мини-SCADA системы даже технологических процессах малой степени сложности.

Исполнительные механизмы – широкий класс электрических, электропневматических, электромеханических и электро-гидравлических устройств. Предназначены для превращения команд управления, получаемых от ПЛК или его блока ввода-вывода, в то или иное физическое действие. Условно можно выделить простые классы исполнительных устройств, имеющих булево управление по принципу включено/выключено, и устройств, требующих аналоговых сигналов управления. Подобные оконечные устройства часто имеют свои микроконтроллеры, генерируют сигналы обратной связи, возможных ошибок и т.п.

Датчики – также широкий класс устройств, предоставляющих информацию о текущем состоянии контролируемого объекта. Номенклатура датчиков чрезвычайно широка. Среди них также можно выделить датчики, выходной сигнал которых подчиняется булевой логике и датчики, имеющие аналоговый выходной сигнал.

Программируемый логический контроллер

Программируемый логический контроллер (ПЛК) представляет 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 (Рис 1).

Важно понимать, что ПЛК это тщательно проработанное устройство, удовлетворяющее жестким стандартам производства и требованиям эксплуатации.

Рисунок 1 – Схематическая диаграмма работы ПЛК

На сегодняшний день отечественные авторы [1] выделяют три категории ПЛК – собственно, сами ПЛК, IBM-PC совместимые контроллеры а также ОПЛК – ПЛК со встроенной операторской панелью.

Области применения ПЛК – управление сложными (распределенными) системами в условиях внешней среды различной степени агрессивности.

Особенности ПЛК, отличающие их от микроконтроллеров:

- стандартизация языков программирования;
- стандартизация подключаемых датчиков и оконечных устройств.

ПЛК работают циклически по методу периодического опроса входных данных. Рабочий цикл ПЛК включает 4 фазы:

1. опрос входов;
2. выполнение пользовательской программы;

3. установку значений выходов;

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

Выполнение первой фазы обеспечивается системным программным обеспечением. После чего управление передается прикладной программе – той программе, которую разработчик записал в память программ ПЛК. Контроллер выполняет алгоритм, заложенный в программу, по завершение очередного цикла управление опять передается системному уровню. За счет этого обеспечивается максимальная простота построения прикладной программы – разработчик не должен знать, как производится 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 управляющего сигнала поясняется на рисунке 2.

Рисунок 2 – цикл работы ПЛК

Выводы ПЛК делятся на несколько категорий. Самые простые – т.н. дискретные входы и дискретные выходы. У большинства промышленных ПЛК функция входа или выхода является жестко заданной, что диктуется специфическими и разными схемами защиты входов и выходов. В промышленной среде применяются стандартные

значения напряжений и токов для информационных линий, наиболее часто применяемые показаны в таблице 1.

Аналоговые входы ПЛК могут воспринимать плавно меняющийся сигнал в том или ином заданном диапазоне. Аналоговые выходы могут генерировать напряжение или ток в соответствии с заданными в управляющей программе значениями. Конкретные значения напряжений, токов и ряд других параметров всегда указываются в спецификации на устройство или модуль.

Таблица 1. Типовые значения напряжений и токов систем автоматизации

Параметр	Максимальное значение	Примечания
Постоянное напряжение	±5В или 0-5В	Чаще всего используется 0-5В как TTL уровни для непосредственного согласования с логическими элементами электронных схем
	±10В или 0-10В	Системы малой автоматики, а также входные значения для аналоговых модулей ввода
	24В	Стандарт напряжения в малых, средних системах автоматизации, где нет специальных требований к помехоустойчивости
	48 В	Высокая помехоустойчивость при линиях большой длины без использования специальных интерфейсов
Постоянный ток	4-20мА 0-20мА	Зависит от схемотехники

Некоторые выходы могут иметь специальные альтернативные функции, такие как высокоскоростной счетчик импульсов, фиксатор

длительности, фронтов импульсов и т.д. Часто на отдельные обособленные разъемы выведены интерфейсы связи ПЛК.

Существует различные принципы классификации ПЛК, один из них – классификация по назначению:

- логические реле (Zelio Logic, ОВЕН Логик);
- ПЛК для машин и механизмов (Modicom M238, M258);
- ПЛК общего назначения (Modicom M340, TSX Premium);
- ПЛК распределенного ввода-вывода (TSX Quantum);
- ПЛК для агрессивных внешних условий.

Альтернативная классификация – по числу входов/выходов:

- микроПЛК или программируемые реле до 32 I/O;
- малые ПЛК 32-12 I/O;
- средние ПЛК 64-1024 I/O;
- большие ПЛК 512-4096 I/O;
- огромные ПЛК системы 2048-8192 I/O.

Выбор аппаратной платформы для автоматизации – сложный процесс, при котором требуется учитывать не только текущие потребности в числе входов выходов, наличие интерфейсов связи, но и учесть возможности масштабирования системы, горизонтальной и вертикальной интеграции.

Контрольные вопросы для самопроверки

1. Назовите аппаратные и программные средства интегрированных автоматизированных систем.
2. Перечислите функции SCADA-систем.
3. Опишите основные особенности, отличающие принципы функционирования ПЛК от обычных микроконтроллеров.
4. Назовите возможные критерии и классификации ПЛК.
5. Опишите рабочий цикл и назовите 4 фазы работы ПЛК.
6. Какие типовые значения уровней напряжений и токов в системах промышленной автоматизации.
7. В каких типах аппаратно-программных систем находит применение операционная система Linux.

Стандарт МЭК 61131

Различные ПЛК могут выполнять одинаковые функции, также как и одинаковые ПЛК могут решать абсолютно различные задачи управления. С целью минимизации времени внедрения, облегчения перехода на другие версии контроллеров в среде производителей и интеграторов принят промышленный международный стандарт МЭК 61131 (*IEC 61131*), который стандартизирует сферу разработки программируемых логических контроллеров. Стандарт МЭК 61131 состоит из следующих разделов:

раздел 1 – Общая информация;
раздел 2 – Требования к оборудованию и тестированию;
раздел 3 – Языки программирования ПЛК;
раздел 4 – Рекомендации для пользователей;
раздел 5 – Коммуникации;
раздел 6 – Функциональная безопасность;
раздел 7 – Программирование на основе нечеткой логики;
раздел 8 – Руководящие указания по применению и реализации языков программирования.

Основная часть, касающаяся программирования ПЛК – раздел 3 «Языки программирования ПЛК» именуется стандартом МЭК 61131-3 (*IEC 61131-3*). Данный подраздел описывает пять языков программирования логических контроллеров – IL, LD, FBD, SFC, ST.

Язык IL (*Instruction List*, рус. – Список инструкций) – язык, похожий на ассемблер. Язык является аппаратно-независимым текстовым и низкоуровневым. В настоящее время применяется крайне ограниченно ввиду постоянного роста сложности управляющих программ и тенденции к повышению уровня абстракции в любых системах программирования.

Язык LD (*Ladder Diagram*, рус. – Лестничная диаграмма или Релейно-контактные схемы). Графический язык, позволяющий представить логику работу посредством реализации логики в виде схемы на электромагнитных реле. Простой, интуитивно понятный язык, оперирующий ограниченным набором исходных элементов.

Язык FBD (*Function Block Diagram*, рус. – Функциональные блоковые диаграммы). Графический язык с высокой степенью абстракции. Позволяет представить логику работы программы в виде функциональных логических блоков.

Язык SFC (*Sequential Function Chart*, рус. – Последовательные функциональные диаграммы). Высокоуровневый графический язык, представляющий логику программы управления с помощью графической модели сетей Петри. Оперирует событиями и переходами. Чрезвычайно удобен при описании последовательного процесса, переходы между состояниями которого происходят по внешним или внутренним событиям.

Язык ST (*Structured Text*, рус. – Структурированный текст). Подобный классическому Паскалю язык, отличается от него строгой типизацией и несколько более бедным синтаксисом.

Для создания управляющих программ ПЛК используются специализированные системы программирования. Это могут быть коммерческие продукты, выпускаемые производителем аппаратных решений ПЛК, например, система программирования Unity Pro компании Schneider Electric. Подобные системы, как правило, ориентированы на производимый компанией технологический ряд ПЛК и не поддерживает контроллеры сторонних производителей.

Особняком стоит среда разработки CODESYS, которая является инструментальным программным комплексом промышленной автоматизации. CODESYS позволяет в одном программном проекте использовать все пять языков программирования, определяемых стандартом МЭК 61131-3, а также в дополнение к ним язык CFC (*Continuous Function Chart*).

Инструментальный программный комплекс промышленной автоматизации CODESYS

Из множества сред разработки систем автоматизации рассмотрим программную систему CODESYS. В настоящее время активно применяется ветка CODESYS v2.x, V3.x. Версии ветки v2.x предоставляют базовый функционал, просты в изучении. Версии ветки v3.x обладают рядом дополнительных функций. В любом случае, применение той или иной ветки среды разработки диктуется типом и моделью используемого ПЛК.

Среда разработки CODESYS v2.3

Создание проекта программной системы автоматизации в CODESYS v2.3 начинается с окна выбора используемого ПЛК или встроенного в среду разработки симулятора. Процесс выбора, стартовое окно и необходимый тип стимулятора показан на рис . 3.

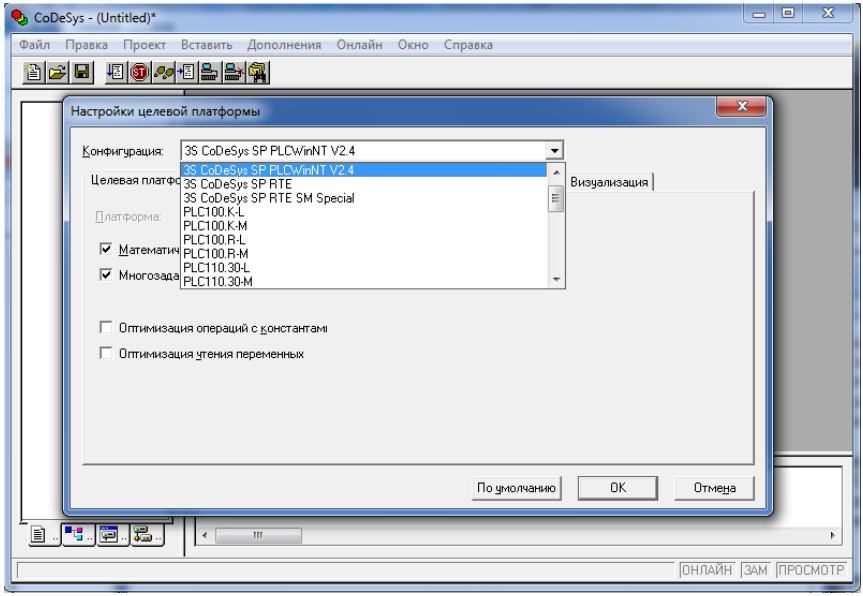

Рисунок 3 – Стартовое окно CODESYS и выбор ПЛК

Минимальными программными компонентами в CODESYS являются «Программа», «Функциональный блок» и «Функция». Компонент «Программа» не требует особых пояснений, следует отметить, что основной модуль программы должен соответствовать этому типу. Тип «Функциональный блок» является типичным объектом по классификации объектно-ориентированного подхода в программировании. Блок данного типа может выполнять некоторые действия, а также обладает собственной памятью. В этом заключается основное отличие от блока типа «Функция». Окно создания нового программного блока приведено на рисунке 4.

Программа PLC_PRG – это специальный POU, который должен быть в каждом проекте. Эта программа вызывается один раз за цикл управления.

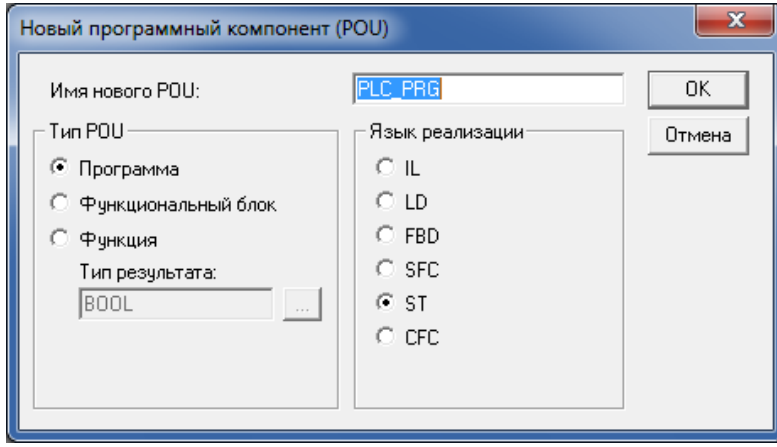

Рисунок 4 – создание нового программного блока

При создании нового программного блока выбирается язык реализации, который будет использован в текущем блоке. В одной программе можно использовать практически любые комбинации языков стандарта МЭК 61131-3.

Минимальная программа для ПЛК на LD

Рассмотрим решение простейшей задачи управления. С подобных задач начиналось знакомство с микроконтроллерами серии AT Mega в рамках курса «Микропроцессорные системы». Задача состоит в необходимости программной реализации выполнения простейшего алгоритма, приведенного на рисунке 5.

В отличие от обычного микроконтроллера, для которого такой алгоритм является интуитивно понятным, в ПЛК он работать не будет. Причина в том, что данный алгоритм содержит бесконечный цикл, если его реализовать для ПЛК, то последний просто зациклится.

Даже в случае, если убрать переход на начало после второй задержки, ПЛК все равно состояние выходов не изменит. Причина в том, что обычный микроконтроллер изменит состояние выхода после обработки текущей команды. А ПЛК обрабатывает весь цикл выполнения программы и, только потом, обрабатывает выходы. Таким образом, подобная программа для ПЛК не приведет к изменению состояния выходов.

Алгоритм подобной программы для ПЛК показан на рисунке 6. Важно отметить, что настройка и запуск аппаратного таймера осуществляется в разделе конфигурации ПЛК, поэтому на блок-схеме рисунка 6 не показана. Также, не показан блок начала программы – для разных языков стандарта МЭК 61131-3 подобный блок может иметь свои особенности.

Для реализации алгоритма, приведенного на рисунке6X на языке LD, следует создать новую программу, выбрав соответствующий язык. Описать три переменных следующего вида: X:BOOL – переменная выхода, в терминологии LD – контакт. T1:TP и T2:TP – экземпляры (таймеры) класса TP.

Класс TP предоставляет пользователю таймеры т.н. одиночного импульса, позволяя производить отсчет промежутков времени, задаваемых переменной на входе PT таймера. Перечень типов переменных в CODESYS приведен в обширной справочной системе программы.

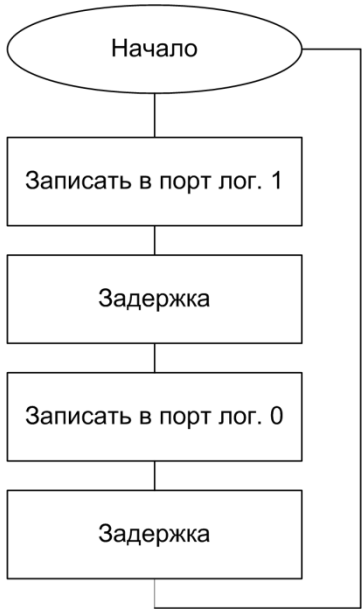

Рисунок 5 – алгоритм для МК

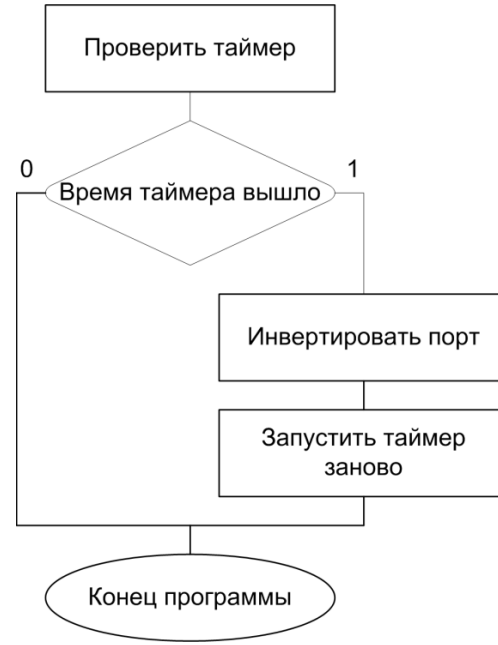

Рисунок 6 – алгоритм для ПЛК

.

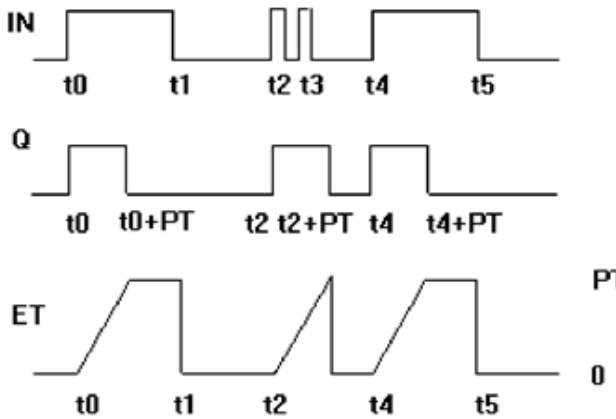

Рисунок 7 – временные диаграммы ТР

На рисунке 7 показаны временные диаграммы таймера ТР. Используемые входы: IN – вход импульса начала отсчета времени, PT – значение времени импульса задержки. Выходы: ET – текущий

остаток времени до включения, Q – выход, на который спустя время, заданное в PT подается импульс.

Произведем небольшую модификацию программы, заключающуюся в добавлении кнопки, при нажатии на которую система начинает работу (последовательную смену состояний с периодом 1+1с), при отпускании кнопки выполнение алгоритма останавливается (рисунок 8). Верхний блок показывает схему соединения контакта A и обмотки B. После запуска эмуляции и старта контроллера двойной щелчок ЛКМ на A позволяет изменить состояния контакта.

Однако состояние меняется отложено. Это значит, что требуемое изменение будет показано напротив соответствующей переменной (средний блок рисунка 8), добавлено в список изменений и пользователь может продолжать вносить изменения в значения переменных проекта. Для применения изменений требуется нажать комбинацию Ctrl+F7. Такой подход позволяет производить *синхронизированные во времени* действия по изменению состояний входов, выходов и значений переменных.

Результат применения изменений показан на нижнем блоке рисунка 8. Видно, что переменные A, B изменили свои значения как в списке переменных, так и на самой диаграмме LD. Подробнее работа с диаграммами LD описана в [2, с. 33]

Понимать работу контакта A и обмотки реле B необходимо следующим образом: при замыкании контакта A течет ток от одного проводника (левая вертикальная линия, синяя) к другому (правая вертикальная линия). Проходя через обмотку B, ток вызывает активацию обмотки, которая, в свою очередь, обеспечивает срабатывание всех остальных контактов, обозначаемых символом B.

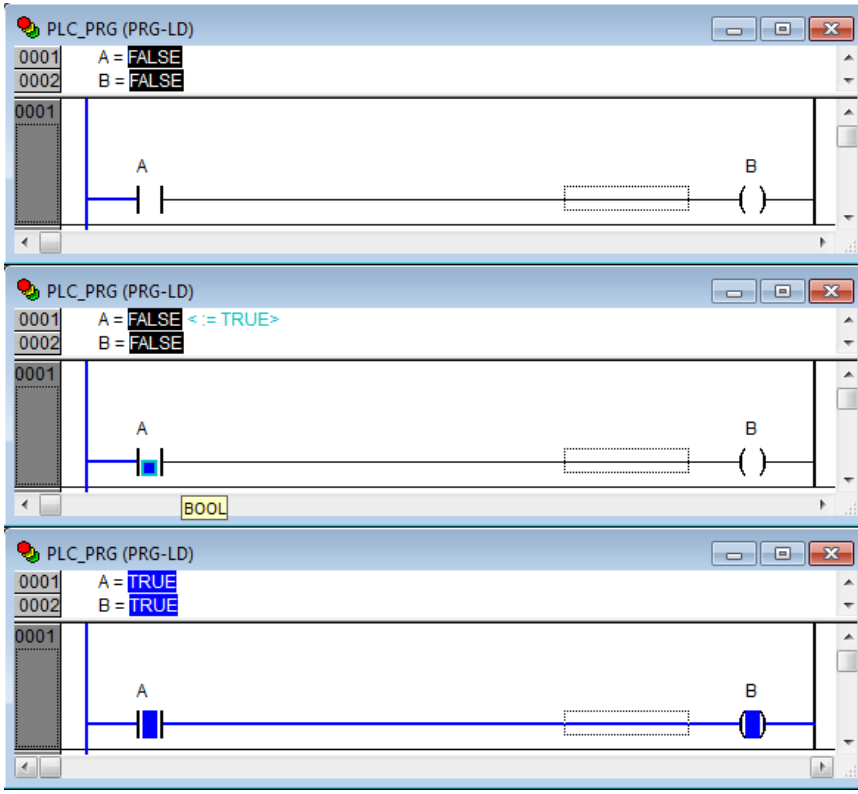

Рисунок 8 – управление кнопкой

Полный вид программы на LD для решения поставленной задачи приведен на рисунке 10, секция переменных – на рисунке 9.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     A:BOOL;
0004     B: BOOL;
0005     X:BOOL;
0006     T1:TP;
0007     T2:TP;
0008 END_VAR
```

Рисунок 9 – секция переменных

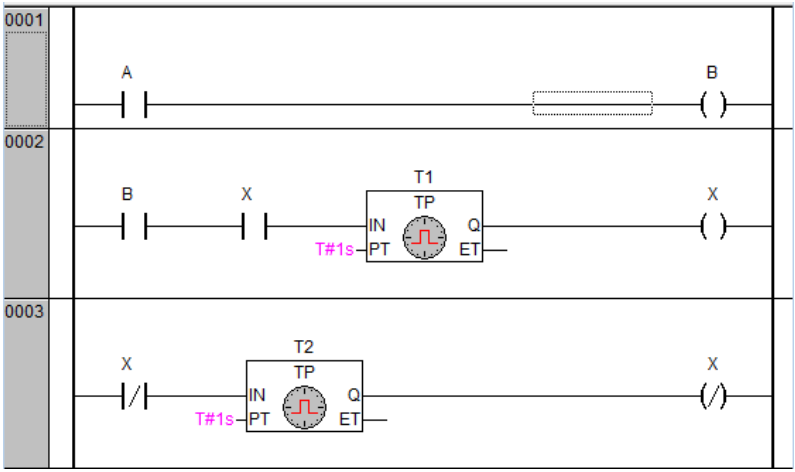

Рисунок 10 – программа на LD

Для визуализации работы лестничной диаграммы (смены состояний логических переменных) используется цифровая трассировка, запустить которую можно выбрав четвертую вкладку проекта «Ресурсы» – «Цифровая трассировка». В появившемся окне (рисунок 11) нажать ПКН, выбрать пункт «Настройки трассировки», с помощью пункта «Менеджер» выбрать переменные для отображения, выбрать пункт «Автоматическое чтение» и запустить автотрассировку.

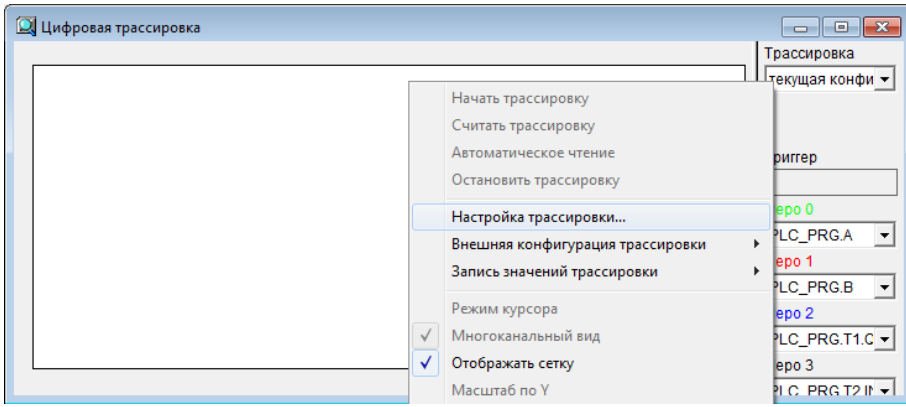

Рисунок 11 – подготовка цифровой трассировки

Результат описанной последовательности действий представлен на рисунке 12.

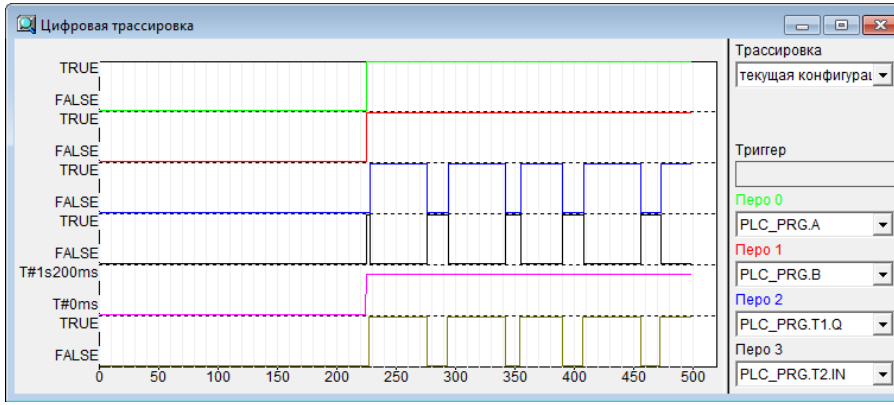

Рисунок 12 – цифровая трассировка работы лестничной диаграммы

Режимы функционирования ПЛК и системы CODESYS

В отличие от привычных микропроцессоров и микроконтроллеров можно выделить несколько основных подходов к обеспечению выполнения программного кода аппаратными средствами ПЛК. Один из них состоит в том, что в ПЛК встроена операционная система реального времени, например операционная система Unity Pro компании Schneider Electric в соответствующих ПЛК. При компиляции программы создаваемый транслятором программный код загружается в ПЛК и передается ОС на выполнение. При этом за взаимодействие с оборудованием ПЛК ответственна его ОС, конфигурация оборудования осуществляется настройками файла проекта. Подобное решение часто применяется для производительных контроллеров, имеющих значительные вычислительные мощности и сложные программные модули.

Система промышленной автоматизации CODESYS использует альтернативный подход. Целевые ПЛК не имеют своей собственной операционной системы, при компиляции программы создается программный код, который «оборачивается» исполнительным кодом для каждого конкретного ПЛК в его собственной конфигурации. Таким образом, для успешного создания программного проекта и загрузки в ПЛК необходимо скачать и установить соответствующие

библиотечные файлы для создания исполняемого кода для данного ПЛК.

В последнее время наблюдается тенденция к распространению ПЛК и устройства отображения информации в одном устройстве – ОПЛК (операторская панель и ПЛК). Часто подобные устройства имеют на борту операционную систему Linux, специальный редактор графического интерфейса и коммуникационные порты под те или иные интерфейсы.

Загрузка программы в ПЛК

Основные команды по взаимодействию системы программирования с аппаратным обеспечением ПЛК собраны во вкладке «Онлайн». Для подключения к эмулятору ПЛК требуется поставить соответствующую галочку в пункте «Эмуляция», выбрать пункт «Подключение». При этом выполняется трансляция программы в программный код. Если она выполняется успешно, система CODESYS переходит в состояние «Онлайн», «Эмуляция», как показано на рисунке 13.

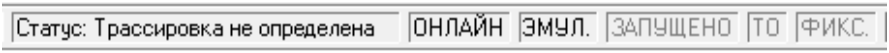

Рисунок 13 – вид строки состояния

Если бы вместо эмулятора был подключен реальный ПЛК, при подключении осуществлялась бы проверка разницы между исполняемым кодом в ПЛК и вновь сгенерированным исполняемым кодом CODESYS. В случае, если разница есть, но она не затрагивает системные параметры ПЛК и не требует обновления его конфигурации, тогда можно загрузить в память ПЛК только изменения в программе.

Если в ПЛК исполняемый код отсутствует, или различия затрагивают область аппаратной конфигурации, требуется полностью обновить исполняемый код, что занимает больше времени. Для того, чтобы при последующих перезагрузках ПЛК использовал новую

программу, требуется выбрать пункт «Создание загрузочного проекта».

После всех этих действий в памяти программ ПЛК будет исполняемый программный код новой программы. В тоже время, как видно из рисунка Х, ПЛК не находится в режиме «Запущено», т.е. он остановлен и ждет команды начать работать по программе. Это сделано для недопущения неконтролируемого старта ПЛК и тех исполнительных устройств, которые к нему могут быть подключены. Для начала исполнения программы следует выбрать пункт «Старт».

Визуализация в системе CODESYS

Система автоматизации CODESYS предоставляет широкие возможности для контроля и отладки программы в режиме онлайн. Таким образом, при подключении к реальному ПЛК есть возможность наблюдать в реальном режиме времени состояния его входов, выходов и внутренних переменных. В тоже время при усложнении управляющей программы оперировать логическими значениями переменных человеку – программисту, отладчику становится труднее. В системах управления сложными объектами применяются специальные системы человеко-машинного интерфейса SCADA-системы, предназначенные для сбора данных, отображения их, коммуникации с человеком и управления технологическим процессом.

В CODESYS содержится встроенное средство для создания операторского интерфейса, который некоторые разработчики даже используют в качестве малой SCADA системы. Таким образом, для создания визуализации следует выбрать третью вкладку, нажав ПКН, создать новый объект-визуализацию. Перечень инструментов для рисования интерфейса представлен на панели инструментов. Основные элементы показаны на рисунке 14, это разнообразные геометрические фигуры, растровый рисунок, кнопки, таблицы тревог, индикаторы, различные графики (тренды).

Рисунок 14 – панель инструментов визуализации

Рассмотрим простой пример – создание кнопки и индикатора состояния переменной. Алгоритм создания управляющего элемента (кнопки) следующий:

1. нарисовать графический символ объекта (прямоугольник или иная фигура);
2. щелкнуть ПКН, «Конфигурировать»;
3. выбрать пункт «Ввод», поставить галочку в «Переменная переключения»;
4. ввести название программного модуля, через точку – имя переменной в нем.

Рисунок 15 поясняет требуемые действия для выполнения данного алгоритма.

Алгоритм создания индикатора, меняющего цвет при изменении логического значения переменной следующий:

1. нарисовать графический символ объекта (прямоугольник или иная фигура);
2. щелкнуть ПКН, «Конфигурировать»;
3. выбрать пункт «Цвета». Обычный цвет заливки будет отображаться при значении связанной переменной FALSE. Тревожный цвет заливки будет отображаться при значении связанной переменной TRUE;
4. выбрать пункт «Переменные», «Изменение цвета» и аналогично п.4 предыдущего алгоритма вписать имя переменной, которая будет изменять цвет объекта.

В результате будет создан объект визуализации, имеющий условную кнопку, при нажатии на которую связанная логическая переменная будет изменять свое значение, а объект-индикатор будет изменять свой цвет при изменении логического значения своей связанной переменной.

Существующие особенности функционирования диаграмм согласованы с принципами работы ПЛК. Изменение состояния выводов ПЛК устанавливаются только после завершения очередного

цикла работы программы. Поэтому конструкция, приведенная на рисунке 16 работать не будет.

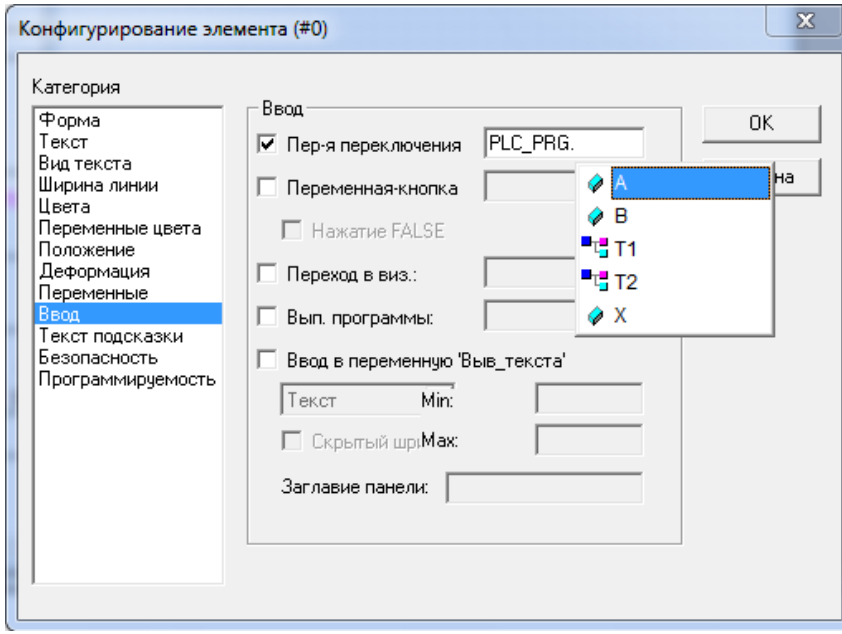

Рисунок 15 – конфигурирование объекта, ввод

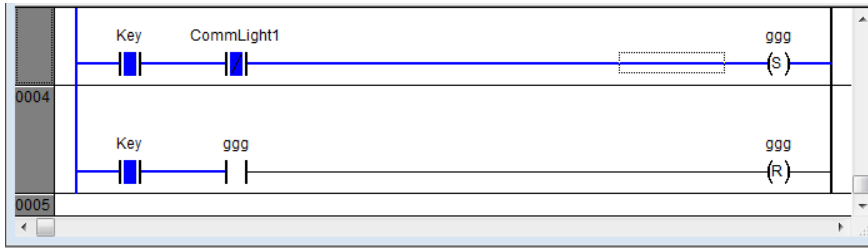

Рисунок 16 – некорректный вариант программного переключателя

Для корректной реализации переключения состояния переменной с помощью последовательных нажатий на кнопку следует использовать функциональный блок, способный идентифицировать ниспадающий фронт импульса кнопки. Таким блоком является F_TRIG, при переходе сигнала с лог. 1 в лог. 0 на входе он генерирует на выходе короткий импульс. Корректная схема переключателя

состояния переменной по нажатию кнопки и результаты трассировки представлены на рисунках 17-18.

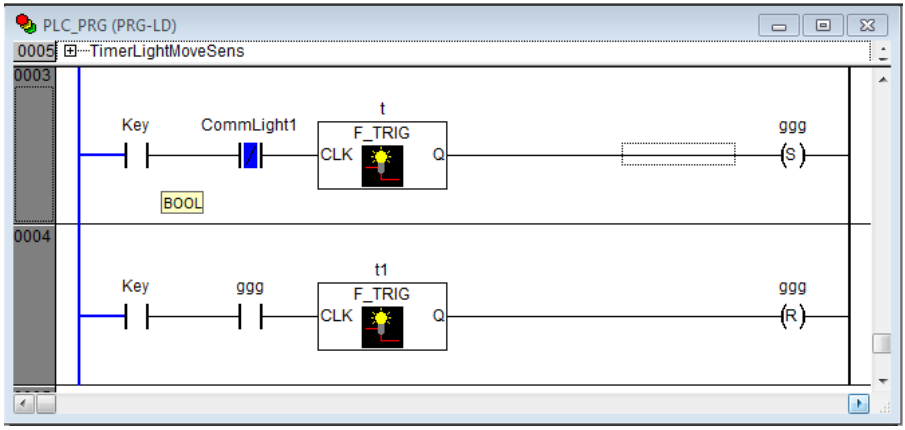

Рисунок 17 – переключатель с блоком F_TRIG

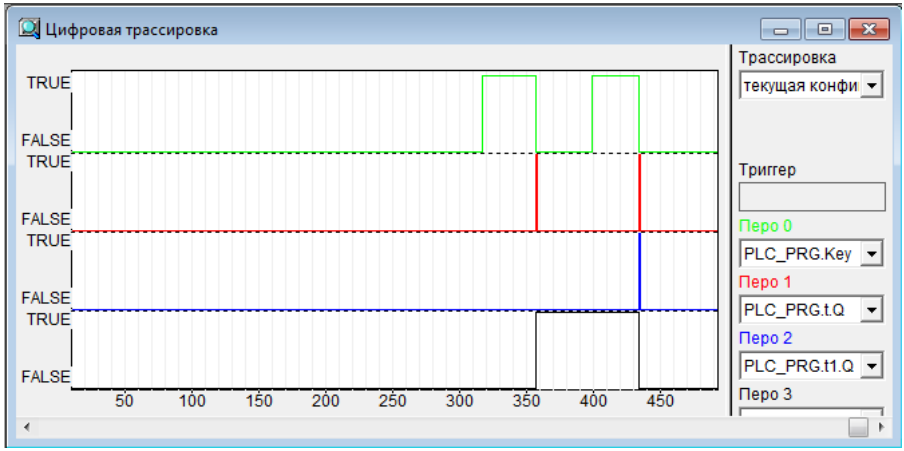

Рисунок 18 – трассировка переключателя

Использование цифровой трассировки позволяет наглядно представить моменты изменения сигналов – значений логических переменных и является удобным инструментом при отладке программ.

Язык программирования ST

Язык ST (Структурированный текст) является адаптированным к требованиями программирования промышленной автоматизации языком Pascal. В качестве процедур, привычных в Pascal, применяются компоненты программ стандарта МЭК.

По аналогии с другими языками МЭК, ST допускает вызов другого функционального блока. К примеру, вызов таймера с параметрами IN, PT осуществляется следующим образом:

```
CMD_TMR(IN := %IX3, PT := 10);
A := CMD_TMR.Q;
```

Переменной A присваивается значение выходной переменной таймера Q.

Существует большое число проектов, в которых основной программный модуль удобно создавать на ST, демонстрирующем большую гибкость в использовании, а функциональные блоки различных режимов создавать на том языке стандарта МЭК, который в наибольшей мере подходит для решаемой задачи.

Язык программирования SFC

SFC – графический язык, который позволяет описать хронологическую последовательность различных действий в программе. Для этого действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами [5]. Пример SFC диаграммы приведен на рисунке X.

Шаг SFC. SFC POU состоит из набора шагов, связанных переходами. Существуют 2 вида шагов:

Шаг простого типа (упрощенный SFC) может включать единственное действие. Графический флажок (небольшой треугольник в верхнем углу шага) показывает, пустой шаг или нет.

МЭК шаг (стандартный SFC) связан с произвольным числом действий или логических переменных. Связанные действия располагаются с правой стороны от шага.

Действие. Действие может содержать список инструкций на IL или ST, схемы на FBD или LD, или снова схемы на SFC.

При использовании простых шагов действие всегда связывается с этим шагом. Для того чтобы редактировать действие, необходимо дважды щелкнуть левой клавишей мышки на шаге. Или выделить шаг и выбрать команду меню «Дополнения» «Открыть действие/Переход».

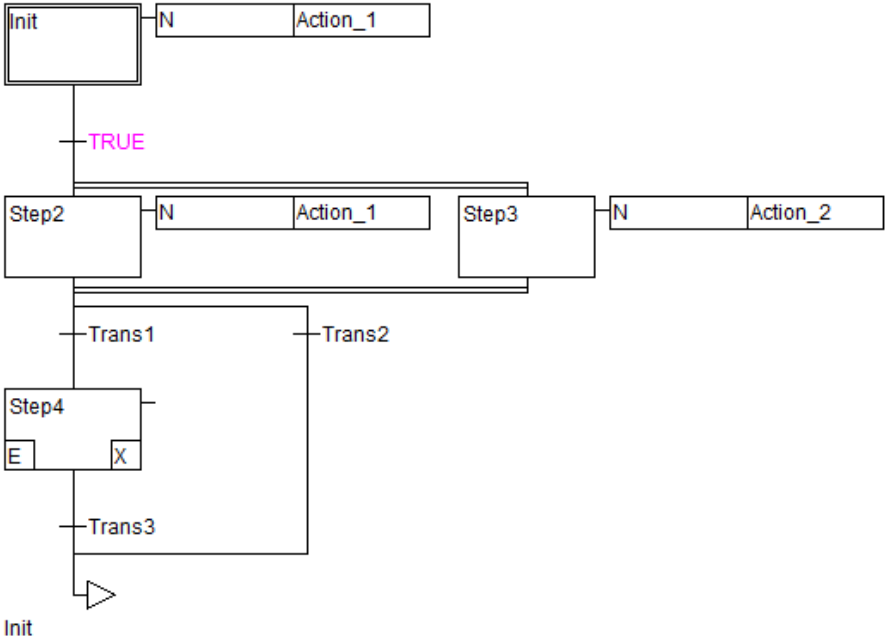

Рисунок 19 – диаграмма SFC

Помимо основного действия, шаг может включать одно входное и одно выходное действие. Действия МЭК шагов показаны в Организаторе объектов, непосредственно под вызывающей их POU. Редактирование действия запускается двойным щелчком мыши или клавишей <Enter>. Новые действия добавляются командой главного меню «Проект» «Добавить действие». Одному шагу можно сопоставить до 9 действий.

Входное действие выполняется один раз при активизации шага, выходное – при деактивизации.

Шаг, который имеет входное действие, обозначается буквой «E» в левом нижнем углу, шаг с выходными действиями – буквой «X» в правом нижнем углу. Пример такого шага приведен на рисунке X – «Step 4».

Входные и выходные действия могут описываться на любом языке. Для того чтобы отредактировать входное или выходное действие, нужно дважды щелкнуть мышкой в соответствующем углу шага.

Переход/условие перехода. Между шагами находятся так называемые переходы. Условием перехода может быть логическая переменная или константа, логический адрес или логическое выражение, описанное на любом языке. Условие может включать несколько инструкций, образующих логический результат, в виде ST выражения (т.е. (i<= 100) AND b) либо на любом другом языке. Но условие не должно содержать присваивания, вызов программ и экземпляров функциональных блоков!

В редакторе SFC условие перехода можно записать непосредственно около символа перехода либо в отдельном окне редактора для ввода условия.

Порядок выполнения шагов и действий в SFC

Активный шаг. После вызова SFC POU начальный шаг (шаг, выделенный двойной рамкой) выполняется первым.

Шаг, выполняемый в данный момент, называется активным. Действия, связанные с активным шагом, выполняются один раз в каждом управляющем цикле. В режиме онлайн активные шаги выделяются синим цветом. Следующий за активным шагом шаг станет активным, только когда условие перехода к этому шагу примет значение TRUE.

В каждом управляющем цикле будут выполнены действия, содержащиеся в активных шагах. Далее проверяются условия перехода, и, возможно, уже другие шаги становятся активными, но выполняться они будут уже в следующем цикле.

Замечание: выходное действие выполняется однократно в следующем цикле, после того, как условие перехода станет истинным.

Кроме действий, с шагом можно связывать логические переменные.

С помощью так называемых классификаторов, действия и логические переменные могут активироваться и деактивироваться, в том числе и с задержкой времени. Например: действие может продолжать работу, даже если запустивший его шаг утратил активность. С помощью классификатора S (установка) можно программировать параллельные процессы и т.д.

Таблица 2. Классификаторы действий в CODESYS

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

Логическая переменная <StepName>.x, связанная с шагом, получает значение ИСТИНА при каждой активации шага.

Действие, связанное с МЭК шагом, описывается справа от него в блоке, состоящем из двух полей. Левая часть этого блока содержит

классификатор, возможно, с константой времени, а правая часть содержит имя действия или логической переменной.

Замечание: Если действие деактивируется, то оно выполняется еще один раз. Это означает, что каждое действие выполняется хотя бы два раза.

При выполнении шага сначала производится деактивация действий, затем выполняются активные действия в алфавитном порядке.

Классификаторы действий приведены в таблице 2. Классификаторы L, D, SD, DS, SL требуют указания временной константы (например “L T#5s”) или переменной типа в формате TIME (например “L t_var”).

Замечание: В процессе деактивации действие выполняется еще один раз. Это относится и к действиям с классификатором P.

Неявные переменные в SFC

В SFC существуют неявно объявленные переменные, которые могут быть полезны для определения состояния шагов, действий и контроля времени активности шагов. Все они устанавливаются в начале каждого рабочего цикла.

Для МЭК шагов определены две переменные: <StepName>.x и <StepName>._x. Переменная <StepName>.x содержит признак активности шага в текущем цикле. Переменная <StepName>._x содержит признак активности шага в следующем цикле. Если <StepName>.x=TRUE, то шаг будет выполняться в текущем цикле. Если <StepName>._x=TRUE и <StepName>.x=FALSE, то шаг будет выполняться в следующем цикле. Соответственно значение <StepName>._x будет скопировано в <StepName>.x в начале цикла.

Для МЭК шагов также определена переменная <StepName>.t, которая показывает время активности соответствующего шага.

Альтернативная ветвь. Две и более ветви SFC могут быть альтернативными. Каждая альтернативная ветвь должна начинаться и заканчиваться переходом. Альтернативные ветви могут содержать параллельные ветви и другие альтернативные ветви. Альтернативная

ветвь начинается горизонтальной линией (начало альтернативы), а заканчивается горизонтальной линией (конец альтернативы) или переходом на произвольный шаг (jump). Если шаг, который находится перед линией альтернативного начала, активен, то первые переходы альтернативных ветвей начинают оцениваться слева направо.

Таким образом, первым активируется тот шаг, который следует за первым слева истинным переходом.

Параллельные ветви. Две и более ветви SFC могут быть параллельными. Каждая параллельная ветвь должна начинаться и заканчиваться шагом. Параллельные ветви могут содержать альтернативные ветви и другие параллельные ветви. Параллельная ветвь наносится двойной горизонтальной линией (параллельное начало) и заканчивается двойной горизонтальной линией.

Если шаг активен, условие перехода после этого шага истинно и за этим переходом следуют параллельные ветви, то активируются первые шаги этих ветвей. Эти ветви выполняются параллельно друг другу. Шаг, находящийся после параллельных ветвей, становится активным только тогда, когда все предыдущие шаги активны и условие перехода истинно.

Более подробно работа с SFC описана в [5].

Отладка программ ПЛК

На сегодняшний день все большая ответственность за качество работы автоматизированных систем управления возлагается на прикладное программное обеспечение ПЛК. В тоже время, языки программирования МЭК611131-3 ориентированы не столько на профессиональных программистов, сколько на специалистов в прикладной области. Сами языки МЭК содержат ряд свойств, позволяющих сводить вероятность ошибок к минимуму. Это строгий контроль типов, специальные программные скобки, отказ от указателей, неявного преобразования типов и языковых конструкций, имеющих побочные эффекты. Высококачественные системы прикладного МЭК программирования помимо традиционных отладочных средств имеют немало специфических инструментов. Некоторые рекомендации специалистов по отладке ПО не применимы для ПЛК физически.

Для полной проверки необходимо выполнить компиляцию проекта. Воспользуйтесь командой меню «Проект», «Компилировать» или клавишей F11. Не нужно пытаться написать всю программу целиком и затем откомпилировать. Компилируйте проект после написания каждого более или менее целостного фрагмента. Ошибки лучше исправлять сразу.

CODESYS выводит результаты компиляции в специальном окне сообщений, расположенном в нижней части рабочего экрана. Сообщения об ошибках всегда нужно анализировать и исправлять, начиная с первого. Игнорировать предупреждения может только очень опытный программист, в особых случаях, при четком понимании того, что он делает. В общем случае, к предупреждениям нужно относится с тем же вниманием, как и к ошибкам.

Если в программе есть обращения к несуществующим (пока не реализованным) POU, достаточно сделать пустые заглушки для них. В ST компонентах достаточно поставить «;» в тексте, чтобы компилятор не выдавал предупреждения. Необходимо добиться полного отсутствия ошибок и предупреждений.

Работа в режиме «Стоп». В режиме «Стоп» система исполнения CODESYS опрашивает входы/выходы контроллера и производит все обычные вспомогательные действия с единственным исключением – прикладная программа не выполняется. В режиме «стоп» все данные переменных «заморожены». С помощью отладчика CODESYS можно остановить программу. Для этого существует команда «Онлайн» – «Стоп» и соответствующая иконка на панели управления. Остановка программы выполняется не мгновенно в отличие от точек останова, а по окончании рабочего цикла ПЛК. В реальном ПЛК можно получить доступ к значениям входов и выходов ПЛК. Это дает возможность проверить монтаж оборудования. Также для изучения работы датчиков возможно наблюдение внешнего воздействия в числовом формате или в виде графиков (цифровая графическая трассировка). Для этого достаточно создать и загрузить в ПЛК пустую программу, содержащую только конфигурацию входов/выходов.

Выполнение по циклам. В режиме «Стоп» нажмите «Ctrl-F5» (команда «Онлайн», «Один цикл»). Контроллер отработает один очередной рабочий цикл и вновь остановится. Выполняя код в реальном времени, есть возможность воздействовать на входы и наблюдать результат. Но увидеть каким путем контроллер пришел в данное состояние часто невозможно ввиду быстрого протекания процесса. Каждый рабочий цикл контроллера это один такт его работы. В каждом такте программа получает значения входов и вычисляет значения выходов. Выполнение по циклам дает возможность детально отследить все промежуточные действия программы.

Обратите внимание – значения переменных, показанные справа от инструкций в режиме «Останов» получены в конце полного рабочего цикла, а не сразу после выполнения данной команды!

Изменение и контроль значений переменных. Для изменения значений переменных во время выполнения программы требуется дважды щелкнуть на переменной – для типа BOOL значение поменяется на противоположное, как показано на рисунке 20,

переменная «btTvLeft». Для остальных типов при щелчке на переменной появляется окно, к котором можно ввести новое значение.

Обратите внимание – при подобных действиях сразу значения переменных не изменяются, а рядом с описанием переменной появляется надпись со знаком <:= X>, где «X» – новое значение. Чтобы фактически записать новые значение переменных, требуется нажать комбинацию «Ctrl + F» – записать значения. Это сделано для того, чтобы можно было выполнять синхронные (т.е. одномоментные) изменения нескольких переменных сразу.

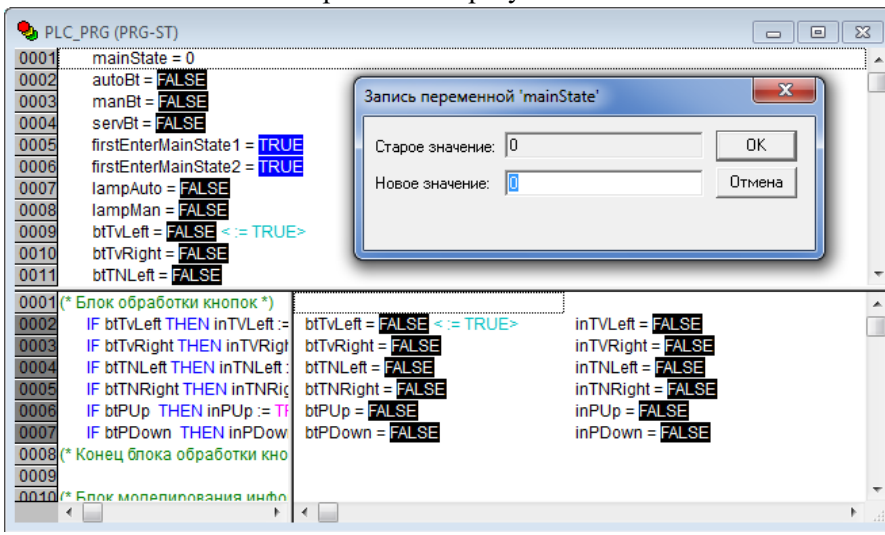

Рисунок 20 – Изменение значений переменных

Фиксация значений переменных. Фиксация значения переменной означает невозможность ее изменения программой ПЛК. Для этого требуется выполнить команду «Онлайн», «Фиксировать» (F7). Значения переменной на экране будут выделены красным цветом. Это признак фиксированных значений переменных.

Подробно и с примерами вопросы написания и отладки прикладных программ для ПЛК описаны в [8], а также в электронном виде в сопроводительной электронной документации курса.

Моделирование информационного поля программы для ПЛК

Одним из основных отличий отладки прикладных программ для персонального компьютера и для ПЛК является наличие информационного поля у программы в ПК и его отсутствие у программы в ПЛК. Под *информационным полем* понимают совокупность входных сигналов ПЛК, поступающих от внешней среды, которая, в свою очередь, тем или иным образом может реагировать на выходные сигналы ПЛК, а также на какие-либо внешние события.

В сложных системах для отладки программы создаются отдельные аппаратно-программные комплексы, в которых реализовываются алгоритмы поведения объекта управления.

При программировании систем управления не высокой сложности достаточно реализовывать локальные программные блоки, которые будут выполнять моделирование информационного поля для программы ПЛК.

Располагать такой программный блок следует в начале программного файла PLC_PRG CODESYS. Рассмотрим следующий пример: требуется моделировать срабатывание концевых датчиков гидроцилиндра, см. рисунок 21.

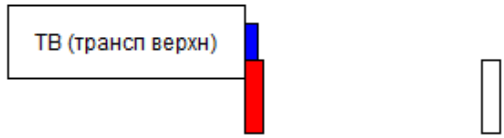

Рисунок 21 – пример визуализации

В графическом примере, показанном на рисунке 21, активен левый датчик (выделен красным), шток гидроцилиндра находится в крайнем левом положении. Для визуализации перемещения штока и моделирования срабатывания датчиков следует создать переменную, значение которой будет инкрементироваться при выполнении действия «движение вправо» и декрементироваться при выполнении действия «движение влево». Важно понимать, что действия по

инкрементированию и декрементированию выполняются в программе ПЛК при поступлении той или иной команды.

Таким образом, для «внешней» активации датчика в блок моделирования ИП требуется добавить проверку значения связанной переменной и, если она превышает заданное значение (например, *tv > 100*), в таком случае следует активировать правый датчик. Аналогично, если связанная переменная меньше граничного значения, требуется активировать левый датчик. Пример подобного кода приведен на рисунке 22. Из рисунка видно, что рассматриваемый код находится в начале программного файла перед автоматом состояний.

Обратите внимание – проверять равенство переменной какому-либо значению не рекомендуется, т.к. может произойти ситуация, когда момент равенства будет пропущен.

```
(* Блок обработки кнопок *)
    IF btTvLeft THEN inTVLeft := TRUE; ELSE inTVLeft := FALSE; END_IF;
    IF btTvRight THEN inTVRight := TRUE; ELSE inTVRight := FALSE; END_IF;
    IF btTNLeft THEN inTNLeft := TRUE; ELSE inTNLeft := FALSE; END_IF;
    IF btTNRight THEN inTNRight := TRUE; ELSE inTNRight := FALSE; END_IF;
    IF btPUp  THEN inPUp := TRUE; ELSE inPUp := FALSE; END_IF;
    IF btPDown  THEN inPDown := TRUE; ELSE inPDown := FALSE; END_IF;
(* Конец блока обработки кнопок *)

(* Блок моделирования информационного поля программы *)
    IF TV_poz > 0   THEN dTVPozRight := TRUE; ELSE dTVPozRight := FALSE; END_IF;
    IF TV_poz < -120   THEN dTVPozLeft := TRUE;   ELSE dTVPozLeft := FALSE; END_IF;
    IF TN_poz > 0 THEN dTNPozRight := TRUE; ELSE dTNPozRight := FALSE; END_IF;
    IF TN_poz < -120   THEN dTNPozLeft := TRUE;   ELSE dTNPozLeft := FALSE; END_IF;
    IF P_poz > 0 THEN dPDown := TRUE; ELSE dPDown := FALSE; END_IF;
    IF P_poz < -120   THEN dPUp := TRUE;   ELSE dPUp := FALSE; END_IF;
(* Конец блока моделирования информационного поля программы *)

CASE mainState OF   (* Автомат *)
0:                  (* Начальное состояние *)
    IF autoBt THEN mainState := 1;
    ELSIF manBt THEN mainState :=3;
    ELSIF servBt THEN mainState := 4;
    END_IF;
```

Рисунок 22 – Пример фрагмента кода моделирования ИП

Использование описанного метода позволяет в одном проекте программы для ПЛК объединить и, собственно, функционал программы и модель внешних воздействий на программу ПЛК.

Контрольные вопросы для самопроверки

1. Дайте определения языкам программирования систем автоматизации стандарта МЭК 61131-3.
2. Какие минимальные программные компоненты представлены в CODESYS v2.3?
3. В чем заключается принципиальное различие алгоритмов программы для микроконтроллера общего применения и программируемого логического контроллера?
4. Назовите основные блоки графического языка LD.
5. В каком виде может быть вызван функциональный блок «Таймер» в LD?
6. Что позволяет наблюдать цифровая трассировка в проекте CODESYS?
7. Опишите особенности языка ST стандарта МЭК 61131-3.
8. Какие особенности программирования связаны использованием switch-технологии?
9. В каком виде может быть вызван функциональный блок «Таймер» в ST?
10. Приведите графический пример простой диаграммы состояний и псевдокод цифрового автомата, описывающий данную диаграмму.
11. Опишите особенности языка FBD. Приведите графический пример.
12. Опишите особенности языка SFC.
13. Опишите понятия «Шаг» и «Действие» в SFC.
14. В какой последовательности осуществляется проверка логических условий переходов на диаграммах SFC?
15. Что представляет собой входное, выходное действие шага SFC?
16. Перечислите основные классификаторы действий шагов SFC в CODESYS.

17. С помощью какого метода можно измерять время выполнения (активности) шага SFC?
18. Может ли оставаться какое-либо действие SFC активным после того, как шаг, с которым связано это действие перестал быть активным?
19. Каким образом можно задать одновременные (параллельные) действия в SFC? Приведите графический пример.
20. Что представляет собой моделирование информационного поля автоматизированной системы?

Практическая работа №1.
Минимальная система управления на LD

Цель: Создание модели простой автоматической системы управления с помощью языка стандарта МЭК 61131-3 LD. Все сигналы в системе – дискретные. Управление процессом с помощью встроенной системы визуализации CODESYS ветки v2.3.

Теоретическая часть в достаточном объеме описана выше. Данная система демонстрирует возможности языка LD по логическому преобразованию данных, поступающих на вход системы.

Практическая часть.

1. Оформить работу согласно установленным требованиям.
2. В системе CODESYS создать модель автоматической системы управления, имеющей входы In1-In6, выходы Q1-Q6.
3. Логическое описание выходов:
   Q1 = In1 AND In2
   Q2 = NOT (In1 OR In2)
   Q3 = In3 OR {(In1 AND In2) AND NOT In4}
   Q4 = In4 {в течение первых 10с.}, = NOT In4 {после отсчета 10с.}
   Q5 = In5 OR Q4
   Q6 = Q2 AND Q4 AND NOT In6
4. Создать систему визуализации с кнопками In1-In6 и выходами Q1-Q6. Индикаторы выходов должны быть зеленого цвета при значении FALSE и красного цвета при значении TRUE соответствующей переменной.
5. Описать функционирование системы. Завершить оформление работы.

Практическая работа №2.
Система управления освещением на LD

Цель: Создание модели автоматической системы управления освещением с помощью языка стандарта МЭК 61131-3 LD. Все сигналы в системе – дискретные. Управление процессом с помощью встроенной системы визуализации CODESYS ветки v2.3.

Теоретическая часть в достаточном объеме описана выше. Данная система является некоторым упрощением реальных систем управления освещением, входящим в комплексы т.н. «Умный дом».

Практическая часть.

1. Оформить работу согласно установленным требованиям.
2. В системе CODESYS создать модель автоматической системы управления, освещением, имеющей входы InLightSens, InMoveSens, InMicrofoneSens, InKey, выходы QLightOut, QLightHall, QLightNear.
3. Логическое описание выходов:
   QLightOut = NOT InLightSens
   QLightHall = (NOT InLightSens AND {InMoveSens + 10с.})
   QLightHall{0->1} = InKey {нечетное нажатие}, QLightHall{1->0} = InKey{четное нажатие} – доп. условия с высшим приоритетом
   QLightNear{0->1} = InMicrofoneSens{нечетный сигнал}
   QLightNear{1->0} = InMicrofoneSens{четный сигнал}
4. Создать систему визуализации с интерактивными (управляемыми ЛКМ) датчиками и световыми приборами. Предусмотреть изменение цвета датчика и светового прибора при активации.
5. Описать функционирование системы. Завершить оформление работы.

Практическая работа №3. Программный модуль на ST. Моделирование информационного поля системы управления

Цель: Создание средствами языка ST основного POU, моделирующего внешнее информационное поле системы управления.

Теоретическая часть. Все существующие микропроцессорные системы управления функционируют в среде внешнего информационного поля. Это понятие есть результат работы многих датчиков, информации, передаваемой по интерфейсам связи и т.п.

При отладке программной части системы управления остро стоит необходимость создания внешнего информационного поля, взаимодействующего с программой управления. Существуют примеры, когда для создания подобного информационного поля создаются отдельные специальные стенды с ПЛК, единственный функционал которых – генерирование сигналов, которые в реальной системе будут сниматься с аппаратных датчиков.

Примером подобного подхода является процесс создания управляющей программы корабля, выполняемый заводом «Фиолент».

В тоже время, при создании относительно простых систем возможно моделирование информационного поля с помощью самой среды CODESYS. Данный поход особенно удобен, когда основной блок PLC_PRG написан на языке ST. Рисунок 23 поясняет изменения в функционировании программы. Чтение физических входов не производится, т.е. программным переменным значения входов не присваиваются. Хотя, сам цикл чтения ПЛК все равно выполняет. Но вместо необходимого цикла присваивания программным переменным значений входов внедряется собственный блок, который по требуемому алгоритму рассчитывает значения входных переменных и передает в программу.

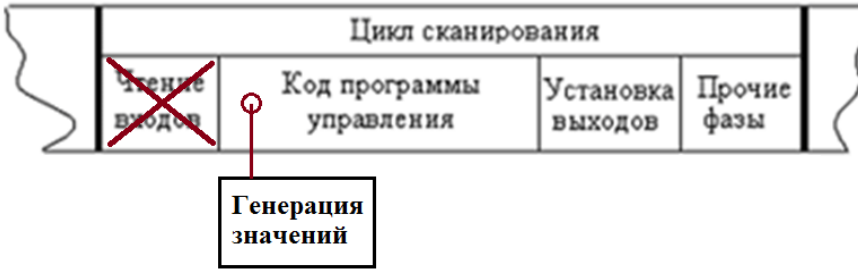

Рисунок 23 – моделирование информационного поля

Подобный подход позволяет выполнять отладку программы в эмуляторе так, как будто она работает на реальном ПЛК, интегрированном в работающую систему управления.

Практическая часть.

1. Оформить работу согласно установленным требованиям.
2. В системе CODESYS создать главный POU PLC_PRG на ST.
3. В основном блоке программы реализовать вызов программного модуля автоматического управления освещением (созданного в л.р. №2).
4. Создать секцию моделирования работы датчиков InLightSens, InMoveSens. Датчик InMicrofoneSens оставить активируемым вручную посредством визуализационного экрана.
5. Создать секцию обработки выходов, которая по значениям внутренних переменных программы будет устанавливать значения выходных переменных QLightOut, QLightHall, QLightNear.
6. Описать функционирование системы. Завершить оформление работы.

Практическая работа №4. Программный модуль на ST. Реализация с помощью автоматного подхода

Цель: Создание средствами языка ST основного программного блока POU, созданного с помощью автоматного подхода к программированию.

Работа является логическим продолжением проекта, выполненного в работе №3.

Теоретическая часть. Switch-технология – технология для поддержки автоматного программирования (технология автоматного программирования), была предложена А.А. Шалыто в 1991 году [3]. Она охватывает спецификацию, проектирование, реализацию, отладку, документирование и сопровождение программ. При этом под термином «автоматное программирование» понимается не только построение и реализация конечных автоматов, но и проектирование и реализация программ в целом, поведение которых описывается автоматами.

Поведение автоматов задается графами переходов (диаграммами состояний), на которых для их компактности входные и выходные воздействия обозначаются символами, а слова используются только для названий пронумерованных состояний. Расшифровка символов выполняется на схеме связей. Применение символов позволяет изображать сложные графы переходов весьма компактно — так, что человек может в большинстве случаев охватить каждый из них одним взглядом. Это обеспечивает когнитивное восприятие указанных графов [4].

Пример основного блока на ST с использованием автоматного подхода показан на рисунке 24.

Данный фрагмент не является полным описанием состояний автомата, а отражает только его часть.

```
(* Конец присвоения внутренним переменным значений входных переменных (кнопки, датчики) *)
CASE mainState OF
    0:  (*состояние полной остановки, начальное состояние*)
        IF firstEnterMainState0 THEN
            init();
            outNFinProduct := 0; (*число готовой продукции = 0 *)
            outFlagError := 0; (*обнуление ошбок *)
            firstEnterMainState0 := FALSE;
            outShowState := 0;    (* Вывод на ИП320 состояния программы - Ключ*)
        END_IF
        IF keySelectAndStart THEN
            firstEnterMainState0 := TRUE;
            mainState := 1;
        END_IF
    1:  (* состояние после поворота ключа (кн.2)  *)
        IF firstEnterMainState1 THEN
            outShowState := 1;    (* Вывод на ИП320 состояния программы - Выбор режима *)
            output0_2 := 0;       (* Выключаем реле активации блока клапанов РЭС9 *)
            firstEnterMainState1 := FALSE;
        END_IF
```

Рисунок 24 – реализация цифрового автомата на ST

Практическая часть. Работа является продолжением работы №3 и использует созданный на предыдущих этапах программный проект.

1. Оформить работу согласно установленным требованиям.
2. В системе CODESYS создать главный POU PLC_PRG на ST.
3. В основном блоке программы реализовать такие состояния, как «Останов», «Автоматическое управление», «Ручное управление».
   В состоянии «Останов» система не должна реагировать на внешние сигналы управления, приходящие как с блока автоматической генерации, так и с блока визуализации при управлении вручную. Кроме того, при переходе системы в состояние «Останов» предусмотреть блок кода, исполняемый один раз при переходе в это состояние – т.н. входное действие. Иллюстрация – на рисунке 24. Данное действие должно принудительно выключать все световые приборы (соответствующие переменные переводить в FALSE)
   В режиме «Автоматическая работа» система управляется значениями входных переменных, которые моделируются как описано в л.р. №3.
   В режиме «Ручная работа» управление происходит только с помощью интерфейса визуализации.

Переход между состояниями осуществляется с помощью дополнительных кнопок, отображаемых на панели визуализации. Также текущее состояние требуется визуализировать цветом, номером или надписью

4. Описать функционирование системы. Завершить оформление работы.

Практическая работа №5. Модель системы автоматизированного управления объектом с использованием программных модулей на языке SFC

Цель: Создание модели системы автоматизированного управления объектом технологического процесса.

Теоретическая часть в достаточном объеме описана выше.

Практическая часть. Требуется создать модель системы автоматизированного управления станком, включающей в себя три гидроцилиндра (транспортер верхний, транспортер нижний и пресс), шесть датчиков положения (по два на каждый гидроцилиндр), вибрационный двигатель. Общий вид визуализации модели, включая пульт управления, показан на рисунке 25.

Отдельное требование – создание подсистемы моделирования информационного поля программы. Данная подсистема должна независимо от основной программы в зависимости от значений переменных, связанных с положениями гидроцилиндров, моделировать активацию датчиков положения. Так, к примеру, на рисунке 25 присутствуют 6 датчиков, красным выделены те датчики, которые являются активными на данный момент.

Пульт управления содержит кнопки выборы режима функционирования системы. При выборе режима «Автомат» система ожидает нажатия кнопки «Старт», после чего автоматически переводит гидроцилиндры в безопасное состояние (ТВ и ТН – в левое крайнее положение, Пуансон – в верхнее, вибратор выключен). Кнопка «Стоп» прерывает выполнение цикла в автоматическом режиме, останавливает все движения и приводит гидроцилиндры в безопасное состояние, выключает вибратор. Кнопка «Пауза»

выключает вибратор и мгновенно останавливает все движения. Повторное нажатие на кнопку «Пауза» позволяет продолжить выполнение программы в авто режиме.

Режим, активируемый кнопкой «Ручной» позволяет управлять всеми движениями вручную с помощью нажатия соответствующих кнопок на правой панели. При этом однократное короткое нажатие запускает соответствующее движение, которое выполняется и оканчивается автоматически по достижению датчика. Время выполнения движений – не менее 5 секунд. При этом, в данном режиме система управления не позволяет выполнять запрещенные движения (к примеру при нижнем положении пуансона двигать вправо верхний гидроцилиндр).

Сервисный режим, активируемый кнопкой «Сервисн» аналогичен ручному с той разницей, что движение активируется на все время нажатия соответствующей кнопки и запрещенных движений нет. Подобные режимы в реальных условиях позволяют проводить сервисные манипуляции с оборудованием высококвалифицированным персоналом, в остальное время они заблокированы, т.к. при неумелом использовании они могут вывести физическое оборудование из строя.

Каждый режим содержит индикатор, меняющий цвет при активации режима.

Требуемая циклическая последовательность работы системы следующая:

1. выполняется приведение гидроцилиндров в начальное состояние, показанное на рисунке 25, вибратор выключен;
2. первое движение осуществляет ТВ вправо до датчика, после чего возвращается обратно;
3. второе движение выполняет пуансон вниз до датчика и обратно;
4. после этого включается вибрационный двигатель (индикатор – изменение цвета) и начинается движение ТН вправо до датчика. По достижению ТН правого датчика вибрационный двигатель выключается, ТН движется влево до датчика.

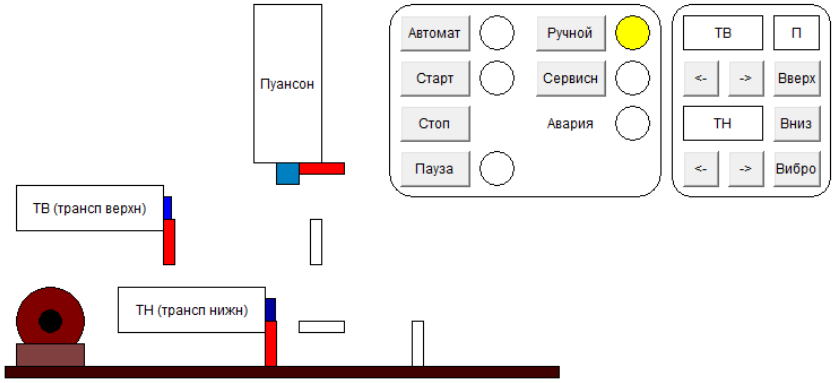

Рисунок 25 – общий вид моделируемой системы, начальное состояние

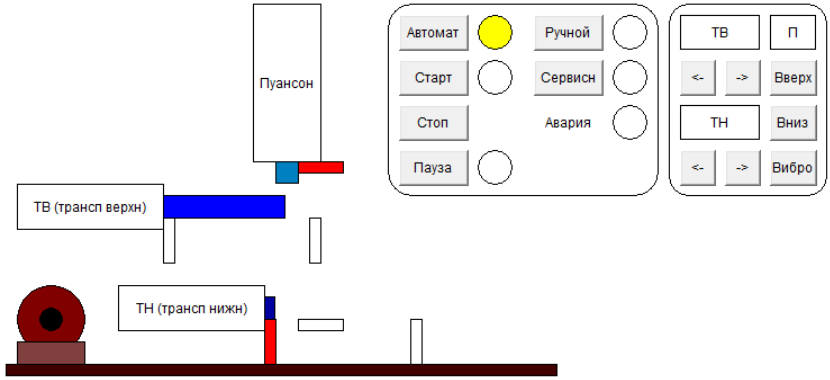

Рисунок 26 – авто режим, движение ТВ

1. Оформить работу согласно установленным требованиям.
2. В системе CODESYS создать модель автоматической системы управления согласно технического задания и описания выше. Основной программный блок должен быть выполнен с применением switch-технологии на языке стандарта МЭК 61131-3 ST, программные блоки ручного и автоматического режима – на языке SFC.
3. Создать систему визуализации.

4. Описать функционирование системы. Завершить оформление работы.

Учитывая близость проекта, выполняемого в ходе данной лабораторной работы к практическому приложению и, следовательно, сложность программного проекта, выполнение работы может быть разделено на следующие уровни:

Уровень 1, оценка 60 баллов. Создан проект в CODESYS, корректно выбрана целевая платформа симуляции. Создана программа PLC_PRG на ST, программный блок ручного режима на SFC, создана визуализация, аналогичная показанной на рисунке X.

Функционал: программа компилируется без ошибок, переходит в ручной режим с активацией соответствующего индикатора и выполняет движения штоками гидроцилиндров при нажатии соответствующих кнопок управления.

Уровень 2, оценка 61-85 баллов. Кроме выполненных заданий 1-го уровня, дополнительно реализован функционал:

- движение гидроцилиндров в ручном режиме начинается по нажатию кнопки, останавливается автоматически при активации датчика. Датчики управляются подсистемой моделирования ИП программы ПЛК;

- реализованы проверки условий движения гидроцилиндров для недопущения повреждения оборудования;

- отдельной кнопкой включается двигатель вибратора (меняет цвет), при этом данное действие может выполняться одновременно с движением гидроцилиндров;

Уровень 3, оценка 86-100 баллов. Кроме выполненных заданий 1-го и 2-го уровней, дополнительно реализован функционал:

- реализован автоматический режим с соответствующей индикацией и корректным началом работы;

- реализован сервисный режим (без проверки условий движения);

- реализован функционал кнопок «Старт», «Стоп», «Пауза»;

- реализована возможность создания аварии пользователем путем вмешательства в систему (на визуализации), индикации данного состояния и корректного прекращения работы в авто режиме.

Приложения

Приложение А. Наиболее часто используемые операторы и функции

Операция	ST
Присваивание	:=
Алгебраические операции	+, -, *, /
Замена знака	-
Остаток от деления	MOD
Абсолютное значение	ABS
Сравнения	< , > ,<=, >=
Равенство, неравенство	=, <>
И, ИЛИ, НЕ	AND, OR, NOT
Исключающее ИЛИ	XOR
Побитный сдвиг влево, вправо	SHL(in,n), SHR(in,n)
Циклический сдвиг влево, вправо	ROL(in,n), ROR(in,n)
Выбор их двух значений наибольшего, наименьшего	MAX(in0, in1), MIN(in0, in1)
Бинарный выбор	SEL(g, in0, in1)
Мультиплексор	MUX(k, in0,...,inN)

Приложение Б. Примеры работы с переменными типа TIME

Операция	Описание
t1 := t2 - T#2m;	0мин – 2мин = -2мин
t1 := t1 + T#5m;	-2мин + 5мин = 3мин
t1 := t1/2;	3мин/2 = 1мин30сек
t3 := LIMIT(T#2s, t1, T#30s);	t3 = 30сек

Библиографический список

1. Пупена О.М. Промислові мережі та інтеграційні технології: курс лекцій для студ. напряму 6.050202 «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання. – К.: НУХТ, 2011. – 67 с.
2. Руководство пользователя по программированию ПЛК в CoDeSys 2.3 / Пролог, 2008
3. Туккель Н. И., Шалыто А. А. SWITCH-технология − автоматный подход к созданию программного обеспечения «реактивных» систем //Программирование. 2001. № 5, с.45-62.
4. Switch-технология, электронный ресурс. Режим доступа http://ru.wikipedia.org/wiki/Switch-технология
5. Руководство пользователя по программированию ПЛК в CODESYS 2.3 /Пролог, 2008. – 452с.
6. Бурькова Е.В. Проектирование микропроцессорных систем: методические ука-зания к курсовому проектированию / Е.В. Бурькова – Оренбург, ГОУ ОГУ, 2008. – 32 с.
7. Илюхин В.Н. Программирование промышленных логических контроллеров «ОВЕН» в системе «CoDeSys». Конспект лекций по дисциплине «Средства электроавтоматики пневмо- и гидросистем». Самара, 2012. – 84 с.
8. Петров И.В. Отладка прикладных программ в CODESYS / И.В. Петров // Промышленные АСУ и контроллеры N2. – 2006г.
9. Трухин М.П. Моделирование сигналов и систем. Дифференциальные, дискретные и цифровые модели динамических систем: учебное пособие / М.П. Трухин; под научной редакцией С.В. Поршнева. – Санкт-Петербург: Лань, 2019. – 228 с.

Учебно-методическое пособие
по проведению практических занятий
по курсу «Автоматизированные системы на встроенных контроллерах»

для студентов
направления подготовки 09.03.01 «Информатика и вычислительная техника»
образовательно-квалификационного уровня «бакалавр»

Автор-составитель Сосновский Ю.В.

Редактор Ю.В. Сосновский

295007, Симферополь, пр-т Академика Вернадского, 4
ФГАОУ ВО «Крымский федеральный университет
им. В.И. Вернадского»
Физико-технический институт