МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ И ОБОРУДОВАНИЯ В ОТРАСЛИ. Программируемый логический контроллер: язык ST

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ
для студентов специальности
6-05-0713-04 «Автоматизация технологических процессов и производств (компьютерная мехатроника)»

Витебск
2026

УДК 681.5

Составители:

К. Н. Ринейский, А. М. Самусев, Д. А. Тёмкин

Одобрено кафедрой «Автоматизация производственных процессов»
УО «ВГТУ», протокол № 9 от 16.04.2026.

Рекомендовано к изданию редакционно-издательским советом
УО «ВГТУ», протокол № 9 от 29.05.2026.

Автоматизация технологических процессов и оборудования в отрасли. Программируемый логический контроллер: язык ST : методические указания / сост. К. Н. Ринейский, А. М. Самусев, Д. А. Тёмкин. – Витебск : УО «ВГТУ», 2026. – 35 с.

Методические указания являются руководством по выполнению лабораторных работ по дисциплине «Автоматизация технологических процессов и оборудования в отрасли» для студентов специальности 6-05-0713-04 «Автоматизация технологических процессов и производств (компьютерная мехатроника)», содержат перечень лабораторных работ, освещают теоретические вопросы подготовки к их выполнению, приводят примеры построения автоматизированных систем управления технологическим процессом на основе использования программируемого логического контроллера, программного обеспечения и средств автоматизации.

УДК 681.5

Содержание

Введение

Основная цель лабораторных работ – это развитие инженерных навыков по разработке систем автоматизации производственных процессов на основе современных технических средств контроля, управления на основе блочно-модульной автоматики и по созданию прикладного программного обеспечения на основе промышленных языков программирования стандарта МЭК 61131-3.

Разработка программного обеспечения имеет большое значение, так как является одной из важных составляющих в этапах разработки систем управления и формирования комплексных знаний инженера по автоматизации.

Полученные знания должны подготовить студента к выполнению дипломного проекта.

Лабораторная работа 1.
«Основы выражений и присваивания в языке Structured Text»

Цель работы: изучение базового синтаксиса языка ST, правил формирования выражений, использования операторов и порядка их вычисления.

Теоретическая часть

Structured Text – это текстовый язык программирования высокого уровня, стандартизированный в IEC 61131-3. Его синтаксис напоминает язык Pascal. ST широко используется для реализации сложных алгоритмов, математических вычислений и логических условий, которые трудно описать графическими языками (CFC, FBD, LD).

Основные преимущества ST:

- Читаемость кода.
- Возможность использования сложных конструкций (циклы, условия).
- Удобство работы с массивами и структурами.

Типы данных для арифметических операций

Для вычислений с дробными числами необходимо использовать тип данных REAL или LREAL. Целочисленные типы (INT, WORD, DWORD) также поддерживают арифметику, но при делении целых чисел результат может быть округлён.

Основой ST-программы служат выражения. Результат вычисления выражения присваивается переменной при помощи оператора «:=», как и в Паскале. Каждое выражение обязательно заканчивается точкой с запятой «;». Выражение состоит из переменных констант и функций, разделенных операторами. Стандартные операторы в выражениях ST имеют символьное представление, например, математические действия: +, -, *, /, операции сравнения и т. д. (табл. 1)

Таблица 1 – ST-операторы

Операция	Обозначение
Умножение	*
Деление	/
Сложение	+
Сравнение	< , > ,<=, >=
Неравенство	<>
Равенство	=

Помимо операторов, элементы выражения можно отделять пробелами и табуляциями для лучшего восприятия. В текст могут быть введены комментарии. Везде, где допустимы пассивные разделители, можно вставлять и комментарии.

Несколько выражений можно записать подряд в одну строку. Но хорошим стилем считается запись одного выражения в строке. Длинные выражения можно перенести на следующую строку. Перенос строки равноценен пассивному разделителю.

Выражение может включать другое выражение, заключенное в скобки. Выражение, заключенное в скобки, вычисляется в первую очередь.

Вычисление выражения происходит в соответствии с правилами приоритета операций. Первыми выполняются операции с наивысшим приоритетом. В порядке уменьшения приоритета операции располагаются так: выражение в скобках; вызов функции; степень EXPT; замена на знака (–); отрицание NOT; умножение, деление и деление по модулю MOD; сложение и вычитание (+, –); операции сравнения (<, >, <=, >=); равенство (=); неравенство (<>); логические операции AND, XOR и OR.

Пустое выражение состоит из точки с запятой «;». Для точки с запятой транслятор не генерирует никакого кода. Если случайно поставить лишнюю «;», это не вызовет ошибки. Единственное осмысленное применение пустого выражения – это обеспечение правильности языковых конструкций».

Примеры

1. Базовое присваивание и функции:

```
iVarl := 1 + iVar2 / ABS(iVar2);
iVarl := 1 + (*получить знак*) iVar2 / ABS(iVar2);
(*проверка на 0 была выше*)
```

2. Логические выражения:

```
bAlarm := bylnpl + bylnp2 AND bylnpl + bylnp2 <> 0 OR bAlarm2;
```

3. Влияние скобок на результат:

```
X := 2 + 2 * 2; (* = 6*) X := (2 + 2) * 2; (* = 8*)
```

4. Работа с переполнением и типами (SINT):

```
siVarl := 120 - siVar2 + 20;  (*120 - 120 = 0, 0 + 20 = 20*)
siVarl := 120 - (siVar2 + 20); (*120 + 20 = -116, 120 + 116 = -20*)
```

5. Пустое выражение в конструкции выбора:

```
IF x = Threshold THEN ; (*все хорошо*)
ELSIF x > Threshold THEN
   bMarker := bMarker - 1; (*шаг вниз*)
ELSE
   bMarker := bMarker + 1; (*шаг вверх*)
END_IF
```

Практическая часть

Ход работы

Задание 1. Написать программу для расчета переменной R5, значение которой определяется по формуле (рис. 1). Переменным R1–R4 задать следующие начальные значения: R1= 60; R2 = 89; R3 = 20; R4 = 4.

Предусмотреть возможность пользователем менять параметры R1–R4 с экрана визуализации во время выполнения программы. При попытке пользователем установить значение R2 = -11 меняет свое световое состояние на 1 секунду out1, значение R2 остается прежним.

Рисунок 1 – Математическая задача

Задание 2. Пользователь имеет возможность менять значение переменной setPoint, которая изменяется в пределах от 0 до 100 (рис. 2). Начальное значение setPoint задать равным 50. При значении setPoint меньше 20 меняет свое световое состояние lowLevel(out1). При значении setPoint больше 90 меняет свое световое состояние highLevel(out2).

Дополнительное задание:
При нажатии на панели кнопок управления приводит к следующему:

- in1 – значение переменной setPoint увеличивается на 1;
- in2 – значение переменной setPoint увеличивается на 10;
- in3 – значение переменной setPoint уменьшается на 1;
- in4 – значение переменной setPoint уменьшается на 10.

Рисунок 2 – Изменение уровня

Задание 3. Пользователь может включить светофор с помощью кнопки in1 и выключить – с помощью in2 (рис. 3). При включении светофора загорается и горит 5 секунд красный сигнал (out1). После того как погаснет красный, загорается на 3 секунды желтый (out2). После того как погаснет желтый, загорается на 5 секунд зеленый (out3). После того как погаснет зеленый, вновь загорается красный и т. д. Как только пользователь нажмет на кнопку стоп, все три сигнала гаснут. Пользователь может заново включить светофор, нажав кнопку запуска.

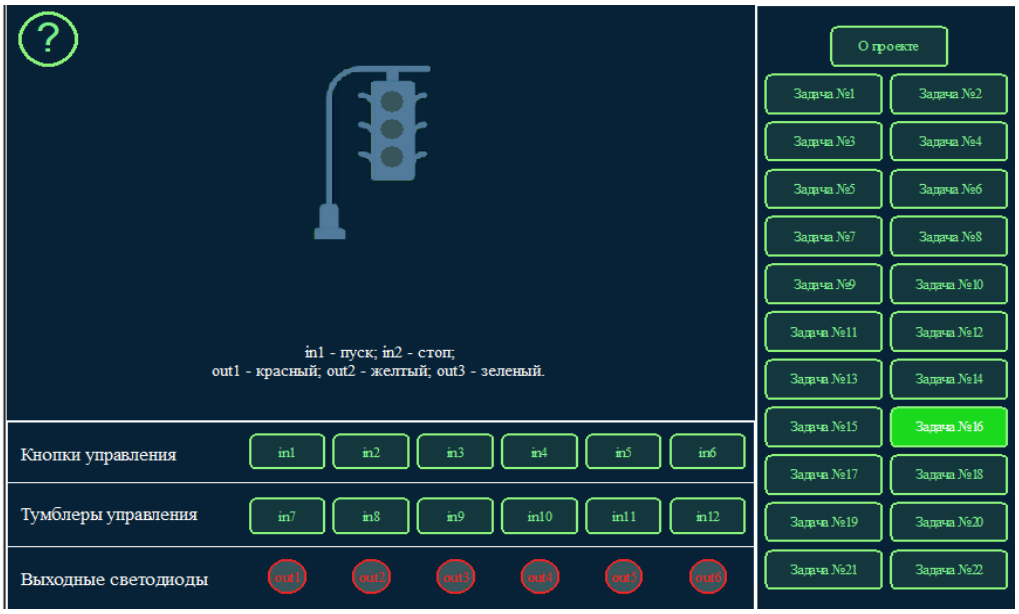

Рисунок 3 – Управление светофором

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Описание графического языка программирования ST.
4. Арифметические операторы CoDeSys.
5. Найти решение уравнения.
6. Вывод.

Контрольные вопросы

1. Что является основой программы на языке ST?
2. Какой символ используется в качестве оператора присваивания?
3. Каким знаком должно заканчиваться каждое выражение?
4. Как влияют комментарии, пробелы и переносы строк на выполнение кода?
5. В каком порядке вычисляются элементы выражения, если часть из них заключена в скобки?
6. Перечислите порядок приоритета операций от высшего к низшему.

Лабораторная работа 2.
«Условные операторы выбора IF и CASE в языке Structured Text»

Цель работы: Изучение синтаксиса и принципов работы операторов выбора IF и CASE, а также правил их использования для управления ходом выполнения программы.

Теоретическая часть

1. Оператор выбора IF «Оператор выбора позволяет выполнить различные группы выражений в зависимости от условий, выраженных логическими выражениями

Полный синтаксис оператора IF (если) выглядит так:

```
IF <логическое выражение IF>
THEN
   <выражения IF> ;
[
ELSIF <логическое выражение ELSEIF 1>
THEN
   <выражения ELSEIF 1> ;
...
ELSIF <логическое выражение ELSEIF n>
THEN
   <выражения ELSEIF n> ;
]
[
ELSE
   <выражения ELSE> ;
]
END_IF
```

Если <логическое выражение IF> ИСТИНА, то выполняются выражения первой группы – <выражения IF>. Прочие выражения пропускаются, альтернативные условия не проверяются. Часть конструкции в квадратных скобках является необязательной и может отсутствовать.

Если <логическое выражение IF> ЛОЖЬ, то одно за другим проверяются условия ELSIF. Первое истинное условие приведет к выполнению соответствующей группы выражений. Прочие условия ELSIF анализироваться не будут. Групп ELSIF может быть несколько или не быть совсем.

Если все логические выражения дали ложный результат, то выполняются выражения группы ELSE, если она есть. Если группы ELSE нет, то не выполняется ничего».

2. Оператор множественного выбора CASE

«Оператор множественного выбора CASE позволяет выполнить различные группы выражений в зависимости от значения одной целочисленной переменной или выражения. Синтаксис:

```
CASE <целочисленное выражение> OF
   <значение 1>:
      <выражения 1> ;

   <значение 2> , <значение 3> :
      <выражения 3> ;

   <значение 4>..<значение 5> :
      <выражения 4> ;
   ...
[
ELSE
   <выражения ELSE> ;
]
END_CASE
```

Если значение выражения совпадает с заданной константой, то выполняется соответствующая группа выражений. Прочие условия не анализируются. Если несколько значений констант должны соответствовать одной группе выражений, их можно перечислить через запятую. Диапазон значений можно определить через двоеточие. Группа выражений ELSE является необязательной. Она выполняется при несовпадении ни одного из условий.

Значениями выбора CASE могут быть только целые константы, переменные использовать нельзя. Одинаковые значения в альтернативах выбора задавать нельзя, даже в диапазонах».

Примеры:

1. Простейший оператор IF:

```
IF bReset THEN
   iVarl := 1;
   iVar2 := 0;
END_IF
```

2. Конструкция IF с несколькими группами ELSIF:

```
IF bReset THEN
   iVarl := 1;
ELSIF byLeft < 16 THEN
   iVarl := 2;
ELSIF byLeft < 32 THEN
   iVarl := 3;
ELSIF byLeft < 64 THEN
   iVarl := 4;
ELSE
   bReset := TRUE;
END_IF
```

3. Использование оператора CASE для выбора режима:

```
CASE byLeft/2 OF
```

```
0, 127:
   bReset := TRUE;
   Varl := 0;
16..24:
   Varl := 1;
ELSE
   Varl := 2;
END_CASE
```

4. Пример некорректного использования CASE (вызовет ошибку):

```
CASE byLeft OF
   20: Varl := 0;
   16..24: Varl := 1; (*Ошибка: 20 входит в диапазон 16..24*)
END_CASE
```

Практическая часть

Ход работы

Задание 1. Для запуска двигателя engine (out1) пользователю необходимо нажать на кнопку запуска push (in1) (рис. 4). Для останова двигателя engine (out1), пользователю необходимо нажать на кнопку останова stop (in2). При нештатной ситуации (срабатывает датчик аварии accident (in7) или пожара fire (in8)) меняет свое световое состояние лампочка lamp (out2). Двигатель engine (out1) останавливается при нештатной ситуации и не может быть запущен заново, пока срабатывает датчик аварии accident (in7) или пожара fire (in8).

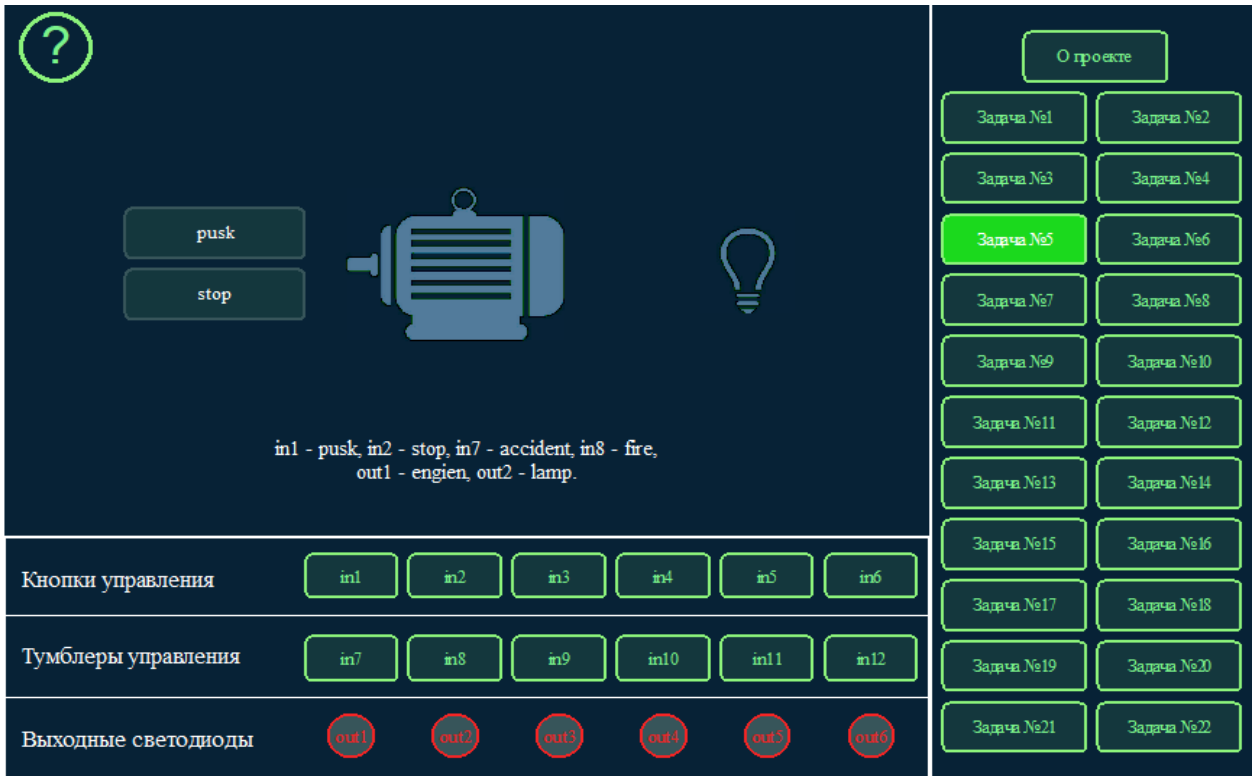

Рисунок 4 – Управление двигателем

Задание 2. Пользователь с помощью кнопок старт (in1) и стоп (in2) запускает и останавливает работу фонтанов (рис. 5). Фонтаны работают в следующей последовательности тактов:
0 – все фонтаны выключены;
1 – работает только 1-й фонтан(out1);
2 – работает только 2-й фонтан(out2);
3 – работает только 3-й фонтан(out3);
4 – работает только 4-й фонтан(out4);
5 – работает только 5-й фонтан(out5);
6 – работает только 6-й фонтан(out6);
7 – работает только 1-й фонтан;
8 – работает только 2-й фонтан;
9 – работает 1-й, 2-й и 3-й фонтан;
10 – работает 1-й, 2-й, 3-й и 4-й фонтан;
11 – работает 1-й, 2-й, 3-й, 4-й и 5-й фонтан;
12 – работает 1-й, 2-й, 3-й, 4-й, 5-й и 6-й фонтан;
13 – все фонтаны выключены;
14 – все фонтаны выключены;
15 – все фонтаны выключены;
16 – все фонтаны выключены.

Процесс циклический. Длительность одного такта 1 секунда.

Рисунок 5 – Управление фонтанами

Задание 3. На объекте (рис. 6) установлен датчик температуры tempSensor и вентилятор fan. Значение температуры задается пользователем целочисленно в

диапазоне от 0 до 100. Вентилятор имеет возможность работы в пяти режимах, режим работы зависит от значения температуры:
fan = 0, при tempSensor = [0, 20]; fan = 1, при tempSensor = (20, 40];
fan = 2, при tempSensor = (40, 60]; fan = 3, при tempSensor = (60, 80];
fan = 4, при значении tempSensor = (80, 100].

Дополнительное задание

В зависимости от режима работы fan, переключать свое световое состояние выходных светодиодов: out1 горит только при fan = 0; out2 горит только при fan = 1; out3 горит только при fan = 2; out4 горит только при fan = 3; out5 горит только при fan = 4.

Рисунок 6 – Управление скоростью вентилятора

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Привести листинг программы на языке ST (должен быть написан студентом самостоятельно) с комментариями.
4. Пояснить, как в коде реализована проверка диапазонов температуры.
5. Пояснить, каким образом с помощью оператора CASE обеспечивается работа только одного светодиода в каждый момент времени.
6. Вывод.

Контрольные вопросы

1. Какое назначение имеет оператор выбора IF в языке ST?
2. Что происходит с остальными условиями в конструкции IF, если основное выражение оказалось истинным?
3. Для чего используется ключевое слово ELSIF?
4. В каком случае будут выполнены команды, находящиеся в блоке ELSE?
5. Каким ключевым словом должна заканчиваться любая конструкция IF?
6. В чем заключается основное отличие оператора CASE от оператора IF?

Лабораторная работа 3.
«Операторы циклов WHILE, REPEAT и FOR в языке Structured Text»

Цель работы: изучение синтаксиса и принципов работы операторов циклов в языке ST, а также способов управления итерациями и правил их безопасного использования.

Теоретическая часть

Циклы WHILE и REPEAT

Циклы WHILE и REPEAT обеспечивают повторение группы выражений, пока верно условное логическое выражение. Если условное выражение всегда истинно, то цикл становится бесконечным.

Синтаксис WHILE:

WHILE <Условное логическое выражение> DO
  <Выражения – тело цикла>
END_WHILE

Условие в цикле WHILE проверяется до начала цикла. Если логическое выражение изначально имеет значение ЛОЖЬ, тело цикла не будет выполнено ни разу.

Синтаксис REPEAT:

REPEAT
  <Выражения – тело цикла>
UNTIL <Условное логическое выражение>
END_REPEAT

Условие в цикле REPEAT проверяется после выполнения тела цикла. Если логическое выражение изначально имеет значение ЛОЖЬ, тело цикла будет выполнено один раз».

Цикл FOR

Цикл FOR обеспечивает заданное количество повторений группы выражений. Синтаксис:

FOR <Целый счетчик> := <Начальное значение>
TO <Конечное значение>
[BY <Шаг>] DO

<Выражения – тело цикла>
END_FOR

Перед выполнением цикла счетчик получает начальное значение. Далее тело цикла повторяется, пока значение счетчика не превысит конечного значения. Счетчик увеличивается в каждом цикле. Начальное и конечное значения и шаг могут быть как константами, так и выражениями. В качестве счетчика можно использовать переменную любого целого типа.

Прерывание итераций EXIT и RETURN

Оператор EXIT, помещенный в теле циклов WHILE, REPEAT и FOR, приводит к немедленному окончанию цикла. Оператор RETURN осуществляет немедленный возврат из POU. Это единственный способ прервать вложенные итерации без введения дополнительных проверок условий.

Примеры:

1. Использование цикла WHILE для деления:

```
ci := 64;
WHILE ci > 1 DO
   Varl := Varl + 1;
   ci := ci/2;
END_WHILE
```

2. Использование цикла FOR для инкремента:

```
Varl := 0;
FOR cw := 1 TO 10 DO
   Varl := Varl + 1;
END_FOR (*Данный цикл будет выполнен 10 раз*)
```

3. Поиск элемента в массиве с оператором EXIT:

```
bObtained:= FALSE;
FOR cN := 1 TO Maxlndex DO
   IF x = aX[cN] THEN
      Index := cN;
      bObtained := TRUE;
      EXIT;
   END_IF
END_FOR
```

4. Оптимизация цикла по скорости:

```
iMax := 5+x;
iPoly := 2*x*x + 1;
WHILE ci < iMax DO
   Var := Varl + iPoly;
   ci := ci + 1;
END_WHILE
```

Практическая часть

Ход работы

Задание 1. Пользователь управляет движением шахматной фигуры согласно шахматным правилам (рис. 7). На доске расположено четыре клетки. На начальном этапе фигура расположена на левой клетке. С помощью кнопок управления in1 (движение по часовой стрелке), in2 (движение против часовой стрелки) и in3 (движение вверх/вниз или влево/вправо) Пользователь может перемещать фигуру в разные клетки. При нахождении фигуры в правой клетке горит светодиод out1, верхней клетке – out2, левой клетке – out3, нижней клетке – out4.

Дополнительное задание

Предусмотреть возможность выбора шахматной фигуры. При нажатии на кнопку in4 выбирается слон, in5 – ладья, in6 – ферзь. Движение фигуры все так же ограничены правилами шахмат.

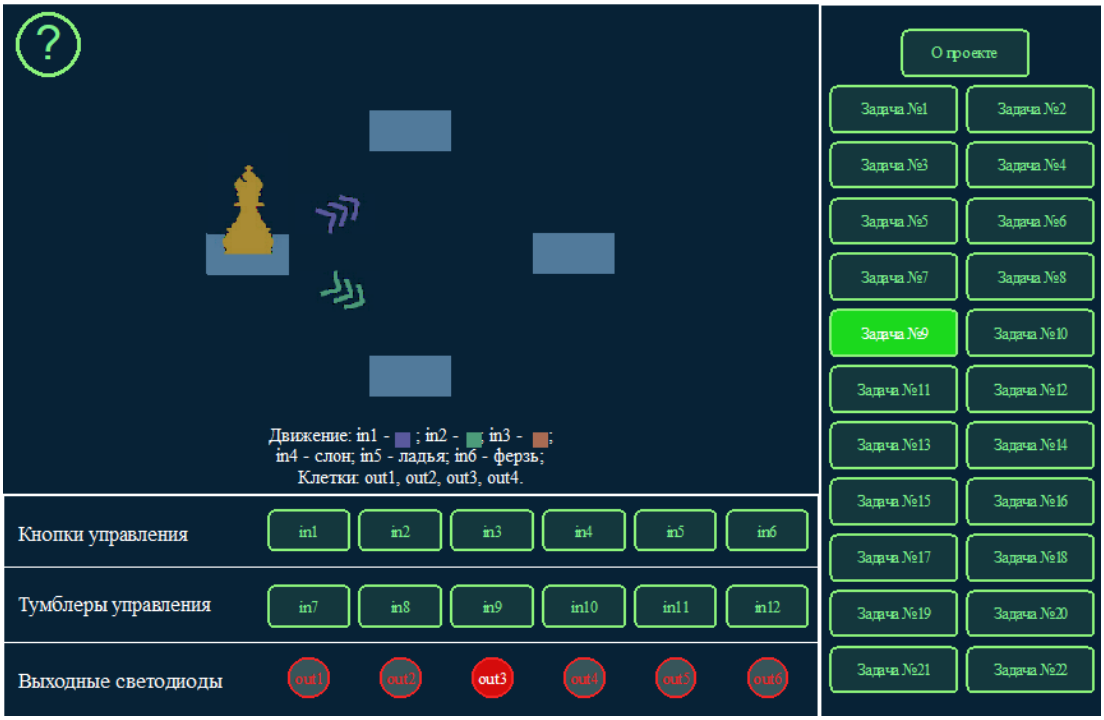

Рисунок 7 – Управление шахматной фигурой

Задание 2. Система (рис. 8) имеет возможность работать в двух режимах – фильтрации и промывки. Одновременная работа в двух режимах недопустима. Переход в необходимый режим осуществляется пользователем только в ручном режиме. Для перехода в ручной режим необходимо нажать тумблер in7, при этом загорается индикатор ручного режима out1 и открывается доступ к кнопкам управления «Фильтрация» (in2) и «Промывка» (in3), нажатие на которые приводит к установке соответствующего режима работы. Индикатор out2 сигнализирует о режиме фильтрации, out3 – промывки. Выйдя из ручного режима, пользователь теряет возможность переключать режимы работы. При

необходимости остановить фильтрацию или промывку в любой момент пользователю следует нажать на кнопку «Стоп» (in1).

Дополнительное задание

При режиме фильтрации открыт клапан, установленный до фильтра (out4) и клапан, установленный после фильтра (out5). При режиме промывки открыт клапан, установленный после фильтра (out5) и клапан слива (out6).

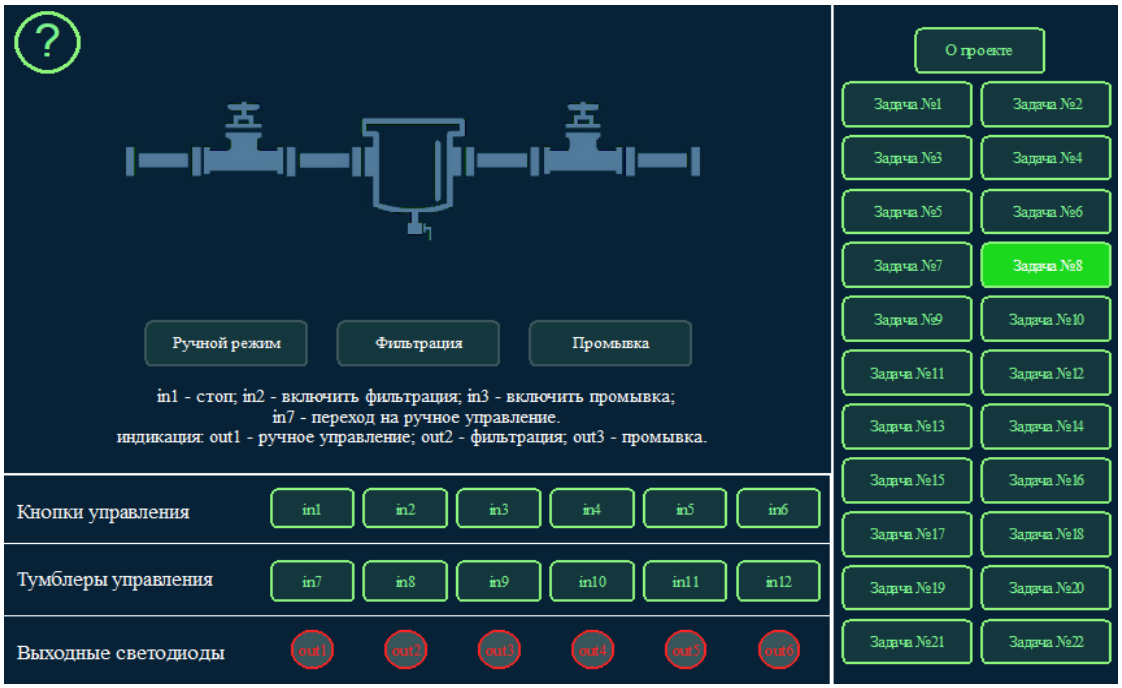

Рисунок 8 – Система фильтрации

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Описание методики подключения и загрузки проекта в ПЛК.
4. Работа с глобальными переменными.
5. Настройка связи между CoDeSys и ПЛК.
6. Загрузка проекта в ПЛК.
7. Вывод.

Контрольные вопросы

1. Какое основное назначение операторов цикла в языке ST?
2. В чем заключается главное различие между циклами `WHILE` и `REPEAT`?
3. Что произойдет, если условие в операторе `REPEAT` будет изначально ложным?
4. Какой синтаксис имеет оператор цикла `FOR`?
5. Можно ли использовать переменные в качестве начальных, конечных значений и шага в цикле `FOR`?

Лабораторная работа 4.
«Типы данных и адресация переменных в языке Structured Text»

Цель работы: изучение стандартных и пользовательских типов данных, правил объявления массивов и структур, а также механизмов прямой и поразрядной адресации в языке ST.

Теоретическая часть

1. Типы данных. Тип данных переменной определяет род информации, диапазон представления и множество допустимых операций. Языки МЭК используют идеологию строгой проверки типов данных. Это означает, что любую переменную можно использовать только после ее объявления. Присваивать значение одной переменной другой можно, только если они обе одного типа. Допускается также присваивание значения переменной совместимого типа, имеющей более широкое множество допустимых значений. В этом случае происходит неявное преобразование типа без потерь. Неявные преобразования типов данных с потерями запрещены.
2. Пользовательские типы данных: массивы и структуры

Массивы представляют собой множество однотипных элементов с произвольным доступом. Массивы могут быть многомерными. Размерность массива и диапазоны индексов задаются при объявлении. Индексы должны быть целого типа и только положительные, отрицательные индексы использовать нельзя.

Структуры предназначены для создания новых типов данных на основе элементов разных базовых типов. С переменной типа структура можно обращаться как с единым элементом, передавать в качестве параметра, создавать указатели, копировать и т. д. Описание структуры происходит глобально, на уровне проекта. Объявление структуры должно начинаться с ключевого слова STRUCT и заканчиваться END_STRUCT.

3. Прямая и поразрядная адресация.

Для создания прямо адресуемой переменной используется следующее объявление: имя переменной AT% прямой адрес тип;. Прямой адрес начинается с буквы, определяющей область памяти: I – область входов, Q – область выходов, M – прямо адресуемая память. Далее следует символ, определяющий тип прямого адреса: X или нет – Бит, B – Байт, W – Слово, D – Двойное слово, L – Длинное слово.

В стандарте предусмотрена удобная форма работы с отдельными битами переменных типа битовых строк – поразрядная адресация. Необходимый бит указывается через точку после идентификатора. Младшему биту соответствует нулевой номер.

Примеры:

1. **Объявление и инициализация массивов:**

```
Mass1: ARRAY [1..6] OF SINT := 1,1,2,2,2,2;
```

Mass2d: ARRAY [1..2,1..4] OF SINT := 1,2,3,4,5,6,7,8;.

2. Работа со структурами:

```
TYPE Trolley:
STRUCT
   Start: TIME;
   Distance: INT;
   Load_On: BOOL;
END_STRUCT
END_TYPE
Telegal.On := True; (*Доступ к элементу структуры*)
```

3. Прямая адресация (связь с входами/выходами):

```
dwHeat AT %MD1: BYTE; wbHeat AT %MW2: BYTE;
byHeat AT %MB4: BYTE; (*Все три объявления адресуют один и тот же байт*)
IF %IW4 > 1 THEN ... (*Использование прямого адреса в коде*).
```

4. Поразрядная адресация:

```
a : WORD; a.3 := 1; (*Установка 3-го бита переменной 'a'*)
bStop AT %IX64.3 : BOOL; (*Адресация бита во входной области*)
```

Практическая часть

Ход работы

Задание 1. Пользователь управляет переключением сигналов колонны (рисунок 9) с помощью ключа(in7). В начальном состоянии красный(out1), желтый(out2) и зеленый(out3) сигналы выключены. Ключ имеет два состояния. Поворот ключа – переход из одного состояния в другое.

1-й поворот – мигает красный сигнал (1 секунду горит, 1 секунду не горит).

2-й поворот – стабильно горит красный сигнал.

3-й поворот – мигает желтый сигнал (1 секунду горит, 1 секунду не горит).

4-й поворот – стабильно горит желтый сигнал.

5-й поворот – мигает зеленый сигнал (1 секунду горит, 1 секунду не горит).

6-й поворот – стабильно горит зеленый сигнал.

7-й поворот – стабильно горят все три сигнала.

8-й поворот – возвращает сигналы в начальное состояние.

Процесс повторяется. При необходимости сбросить сигналы в начальное состояние, необходимо нажать кнопку in1.

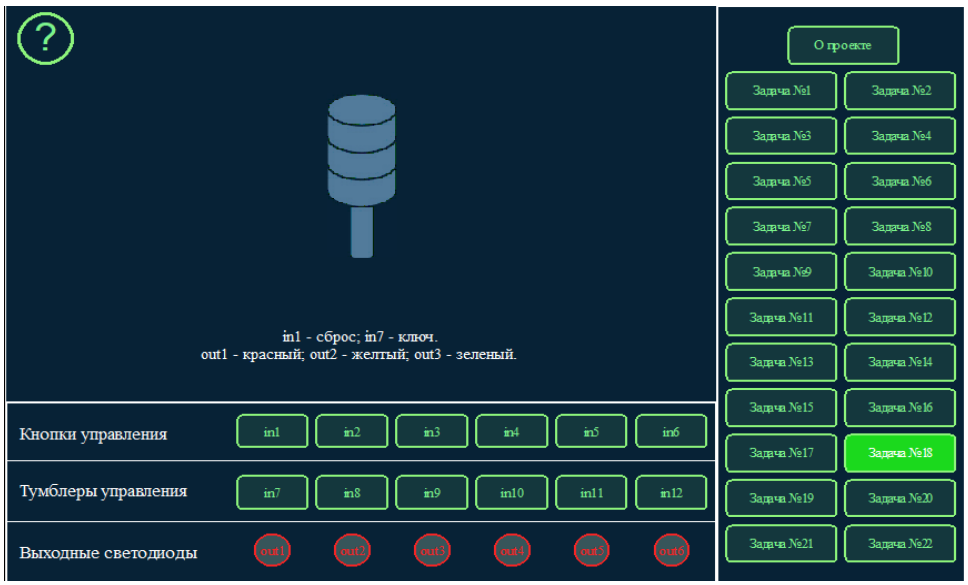

Рисунок 9 – Световая колонна

Задание 2. На начальном этапе двери ворот закрыты (рис. 10), горит светодиод out1. Для открытия ворот пользователю необходимо нажать на кнопку in1 – запускается режим открытия ворот: в течение 5 секунд мигает светодиод out1, далее запускается привод открытия/закрытия ворот (out3). По достижении верхнего положения привод останавливается и загорается светодиод out2. Режим открытия ворот пользователь может остановить, пока мигает светодиод out1, нажав на кнопку in2. Для закрытия ворот пользователю необходимо нажать на кнопку in2 – запускается режим закрытия ворот: в течение 5 секунд мигает светодиод out2, далее запускается привод открытия/закрытия ворот (out3). По достижении нижнего положения привод останавливается и загорается светодиод out2. Режим закрытия ворот пользователь может остановить, пока мигает светодиод out2, нажав на кнопку in2.

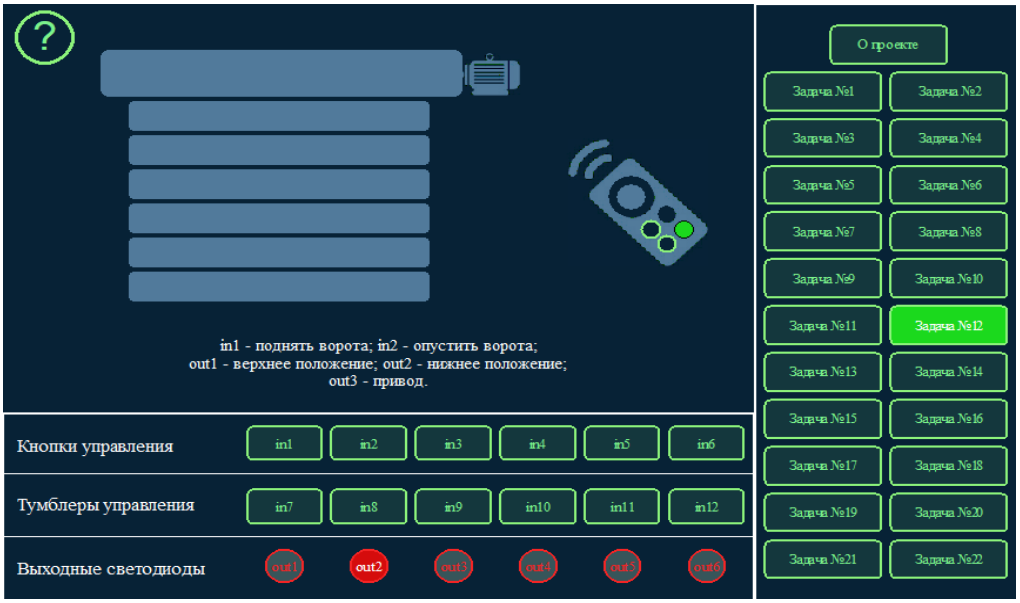

Рисунок 10 – Управление воротами

Задание 3. В каждой из комнат установлены датчики дыма (рис. 11). В первой комнате пользователь управляет датчиками с помощью тумблеров in7, in8, in9, во второй – с помощью тумблеров in10, in11, in12. При активации одного из датчиков в комнате срабатывает соответствующий сигнал предупреждения out1 (для первой комнаты) и out2 (для второй комнаты). При одновременной работе более 2 датчиков в одной комнате срабатывает общий сигнал тревоги out3. Установить сигнал тревоги пользователь может вручную, нажав на кнопки in1 (для первой комнаты) и in3 (для второй комнаты). Для сброса общего сигнала тревоги пользователю необходимо нажать на кнопку in2 (для первой комнаты) и in4 (для второй комнаты).

Дополнительное задание

Если сигнал тревоги был установлен и в первой, и во второй комнате, то пользователю необходимо сбросить сигнал и в первой, и во второй.

При срабатывании сигнала тревоги в первой комнате срабатывает спринклер out4 и работает до тех пор, пока пользователь не сбросит сигнал в этой комнате.

При срабатывании сигнала тревоги во второй комнате срабатывает спринклер out5 и работает до тех пор, пока пользователь не сбросит сигнал в этой комнате.

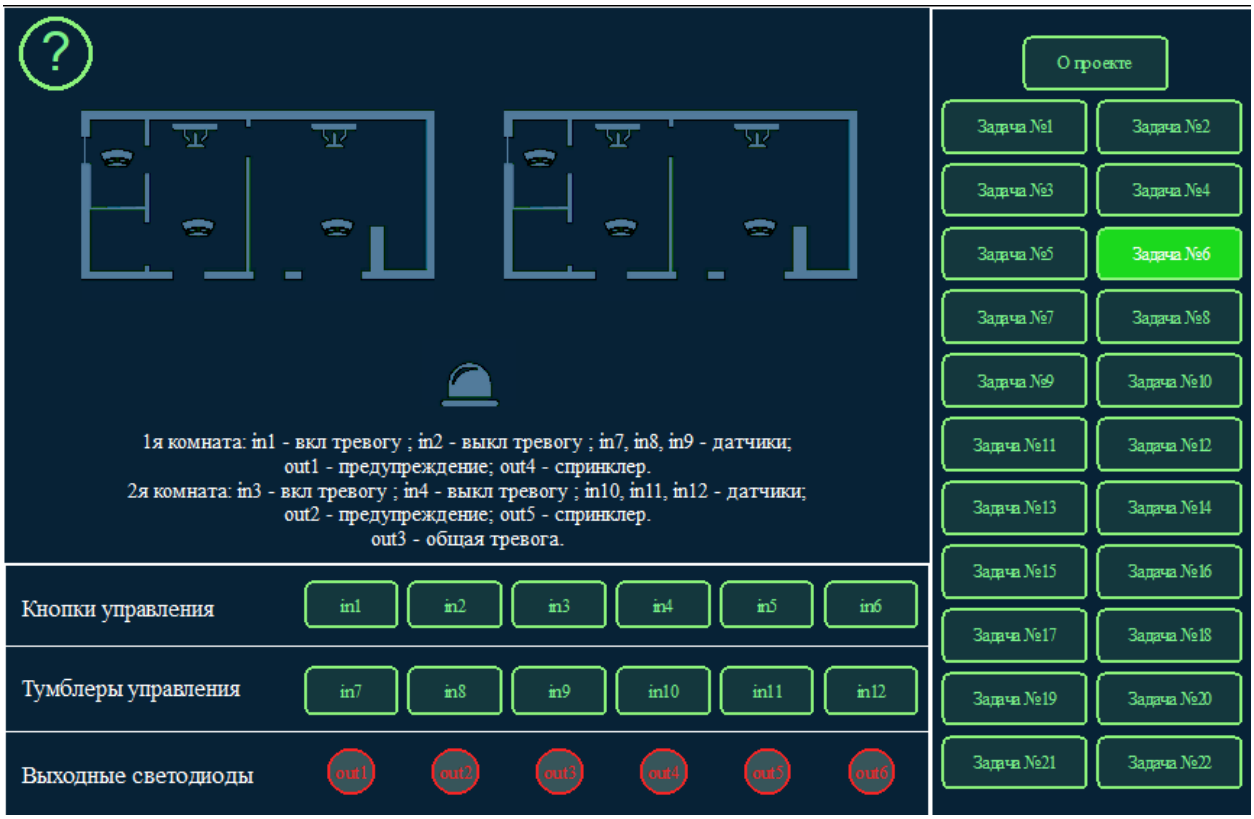

Рисунок 11– Пожарная сигнализация

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Описание графического языка программирования CFC.
4. Логические и арифметические операторы CoDeSys.
5. Реализация изменения состояния объекта управления с использованием нескольких устройств управления.
6. Найти решение уравнения.
7. Вывод.

Контрольные вопросы

1. В чем заключается суть идеологии «строгой проверки типов» в МЭК-языках?
2. Можно ли присвоить значение переменной типа INT переменной типа SINT без специальных операторов?
3. Какие требования предъявляются к индексам массивов в языке ST?
4. В чем основное отличие структур от массивов?
5. Какими ключевыми словами ограничивается объявление структуры?

Лабораторная работа 5.
«Стандартные функциональные блоки в языке Structured Text: таймеры и счетчики»

Цель работы: изучение принципов работы, правил вызова и особенностей применения стандартных функциональных блоков таймеров (TP, TON, TOF) и счетчиков (CTU, CTD, CTUD) в программах на языке ST.

Теоретическая часть

1. Особенности таймеров ПЛК

Таймеры ПЛК принципиально отличаются от таймеров, применяемых в языках общего применения. В языках программирования компьютеров существуют функции задержки (delay, sleep), которые приводят к приостановке выполнения программы на заданное время. Таймера, способного приостановить работу ПЛК, в стандарте МЭК нет. Представьте себе, что на один вход контроллера поступает некоторый сигнал. На второй вход поступает тот же сигнал, но через аппаратный модуль задержки. Именно так работают стандартные таймеры. Временная задержка влияет только на формирование выходных сигналов и не вызывает никакого замедления в программе.

Для правильной работы таймеров необходима аппаратная поддержка. Все экземпляры функциональных блоков таймеров «засекают» время, пользуясь общими часами. Нельзя полагаться на то, что повторный вызов экземпляра функционального блока в одном рабочем цикле даст различные результаты.

Значения программных таймеров могут обновляться при вызове экземпляра функционального блока или синхронно с обновлением входов.

2. Типы стандартных таймеров

TP (генератор импульса): запуск таймера происходит по фронту импульса на входе IN. Вход PT задает длительность формируемого импульса. После запуска таймер не реагирует на изменение значения входа IN. Выход ET отсчитывает прошедшее время. При достижении ET значения PT счетчик останавливается, и выход Q сбрасывается в 0.

TOF (таймер с задержкой выключения): по фронту входа IN выход Q устанавливается в TRUE. Сброс счетчика ET и начало отсчета времени происходит по каждому спаду входа IN. Выход Q будет сброшен через заданное PT время после спада входного сигнала. Если во время отсчета вход IN будет установлен в TRUE, то отсчет приостанавливается.

TON (таймер с задержкой включения): по фронту входа IN выполняется обнуление счетчика и начинается новый отсчет времени. Выход Q будет установлен в TRUE через заданное PT время, если IN будет продолжать оставаться в состоянии TRUE. Спад входа IN останавливает отсчет и сбрасывает выход Q в FALSE.

3. Стандартные счетчики

CTU (инкрементный счетчик): по каждому фронту на входе CU значение счетчика (выход CV) увеличивается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет или превысит заданный PV порог. Логическая единица на входе сброса (RESET = TRUE) останавливает счет и обнуляет счетчик (CV := 0).

CTD (декрементный счетчик): по каждому фронту на входе CD счетчик (выход CV) уменьшается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет нуля. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE.

CTUD (инкрементный/декрементный счетчик): по значению входа RESET = TRUE счетчик CV сбрасывается в 0. По значению входа LOAD = TRUE счетчик CV загружается значением равным PV. По фронту на входе CU счетчик увеличивается на 1. По фронту на входе CD счетчик уменьшается на 1 (до 0).

Примеры

1. Вызов экземпляра счетчика (CTU) с передачей параметров:

```
ctuTimeMeter (RESET := FALSE, CU := Inp1, CV => x);
```

2. Доступ к переменным экземпляра таймера через точку:

```
`X := ctuTimeMeter.PV - ctuTimeMeter.CV + 1;.
```

3. Реализация генератора импульсов на базе таймера TP:

на рисунке 12 показано простейшее применение блока TP в качестве генератора коротких прямоугольных импульсов. Длительность паузы задается таймером. В первом цикле bx получит значение 1 благодаря инвертору NOT. Так

формируется фронт запуска, который поступает на вход IN таймера в третьем цикле. Инвертор формирует фронт запуска по каждому спаду выхода таймера.

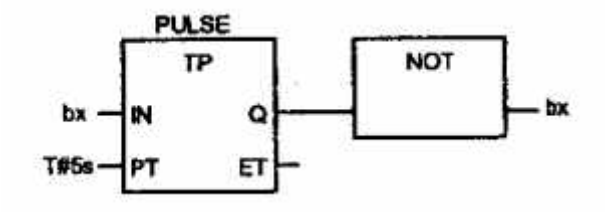

Рисунок 12 – Использование блока TP

4. Использование таймера TON в алгоритме управления приводом:

```
ENTRY_ACTION
   Tm(IN := FALSE, PT:= tStart); (*запуск таймера*)
   Tm.IN := TRUE;
   Starting := TRUE;
   Power := TRUE;
   Reversal := Direction;
END_ACTION
```

Практическая часть

Ход работы

Задание 1. Необходимо посчитать количество людей в комнате (рис. 13) и если в комнате находятся люди, то необходимо включить свет (out1). На входе установлены два датчика: внешний (in7) и внутренний (in8). Когда человек заходит в комнату, срабатывает внешний датчик, затем внутренний – количество людей увеличивается на 1. Далее перестает работать внешний датчик, затем внутренний. Когда человек выходит из комнаты, срабатывает внутренний датчик, затем внешний – количество людей уменьшается на 1. Далее перестает работать внутренний датчик, затем внешний. Рассматриваем идеальную ситуацию: человек проходит без остановок и разворотов, в один момент проходит только 1 человек.

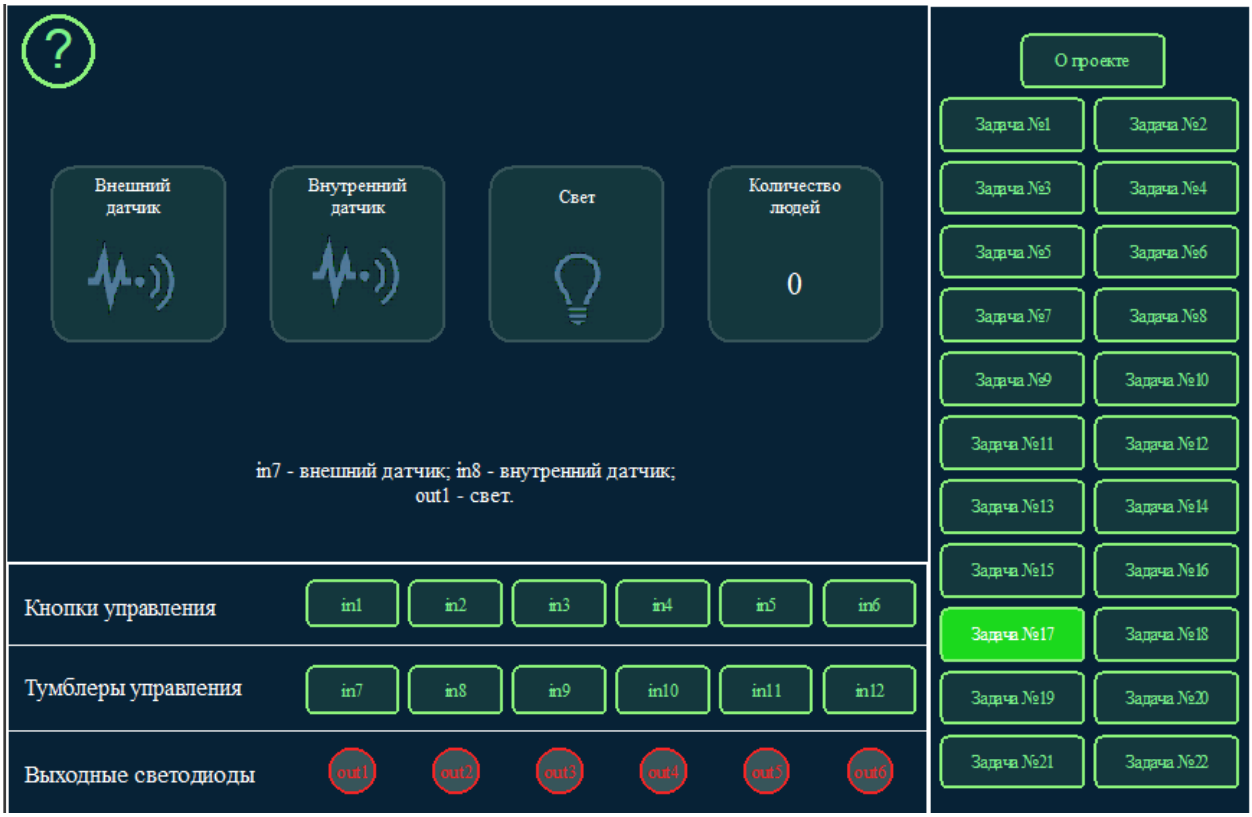

Рисунок 13 – Управление автоматическим освещением

Задание 2. Пользователь вручную задает значение (рис. 14), поступаемое с датчика давления. Если значение давления меньше 100 кПа, то включается насос подкачки (out1). Когда давление установится больше 200 кПа, насос автоматически выключается. Спустя 5 секунд после сигнала на включение насоса, система оценивает рост давления – если каждые 4 секунды, при включенном насосе, не происходит увеличение давления, то срабатывает сигнал тревоги (out2). Сигнал тревоги может отключить только пользователь, нажав на кнопку сброса тревоги (in1). Пока не отключен сигнал тревоги, насос остается в выключенном состоянии.

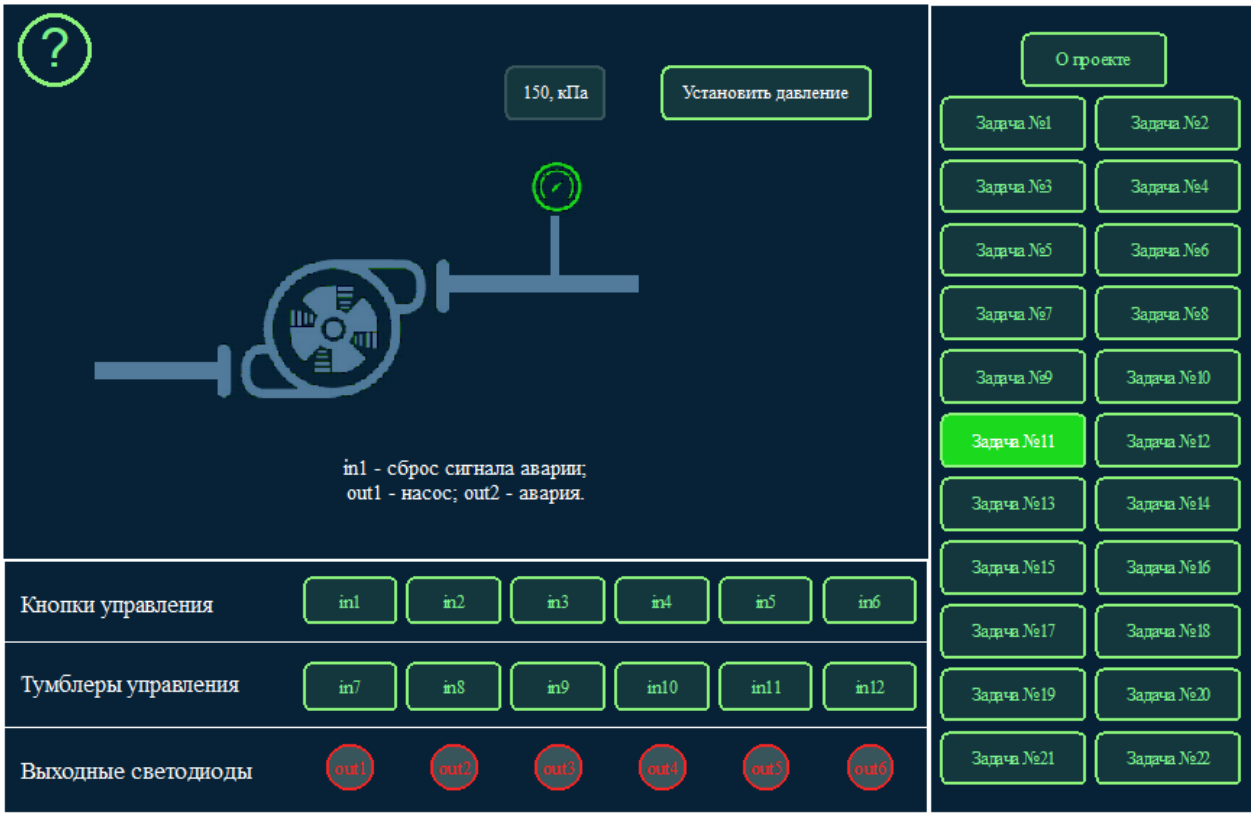

Рисунок 14 – Управление насосом подкачки

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Описание графического языка программирования CFC.
4. Операторы выбора и ограничения CoDeSys.
5. Операторы сравнения CoDeSys.
6. Реализация конфигурирования объектом управления (вентилятором) в зависимости от влияющих факторов (температуры) с использованием операторов выбора и ограничения, сравнения.
7. Вывод.

Контрольные вопросы

1. В чем заключается принципиальное отличие таймеров ПЛК от функций задержки в языках программирования для персональных компьютеров?
2. Почему результат повторного вызова одного и того же экземпляра таймера в одном рабочем цикле ПЛК может быть неизменным.
3. Опишите поведение выхода `Q` таймера `TP` при поступлении сигнала на вход `IN` во время уже запущенного отсчета времени.
4. Какое событие на входе инициирует начало отсчета времени в блоке `TOF`?
5. При каком условии выход `Q` таймера `TON` примет значение `TRUE`?
6. Каким образом осуществляется сброс текущего значения (`CV`) в инкрементном счетчике `CTU`?

Лабораторная работа 6.
«Стандартные операторы выбора, ограничения и сравнения в языке Structured Text»

Цель работы: изучение синтаксиса и особенностей применения стандартных операторов выбора (SEL, MUX), функций поиска экстремумов (MIN, MAX), ограничителя (LIMIT), а также операторов сравнения в программах на языке ST.

Теоретическая часть

1. Операторы выбора и ограничения представлены в таблице 1

Таблица 1. – Операторы выбора и ограничения

Текстовый формат	Действие	Типы параметров
OUT:- SEL(G, INO, INI)	Бинарный выбор: OUT:- INO при G - FALSE OUT:- INI при G - TRUE	INO, INI: ANY G: BOOL
OUT :- MAX(IN0, INI)	Наибольшее из значений	ANY
OUT := MIN(IN0, IN1)	Наименьшее из значений	ANY
OUT := LIMIT(Min, IN, Max)	Ограничитель: OUT:=MIN(MAX(IN,Min), Max)	-
OUT := MUX(K, IN0, ..., INn-1)	Мультиплексор: OUT :=IN_K	IN_0 , … , $IN_{(K-1)}$:ANY K:ANY_INT

2. Операторы сравнения

Операторы реализуют операции сравнения:

- **GT:** > (больше)
- **GE:** >= (больше или равно)
- **EQ:** = (равно)
- **LE:** <= (меньше или равно)
- **LT:** < (меньше)
- **NE:** <> (не равно)

Операторы сравнения принимают 2 параметра любого типа, тип возвращаемого значения – **BOOL**. Действие описывается по отношению к первому параметру. Так, bOUT := IN1 > IN2; будет иметь значение TRUE, если IN1 больше IN2.

Если необходимость сравнения нескольких величин все же возникает, необходимо использовать несколько операторов сравнения, объединенных по И. Сравнение трех переменных на ST можно записать так: bOut := iX1 > iX2 AND iX1 > iX3 AND iX2 > iX3;.

Сравнение текстовых строк производится на основании значений кодов символов.

Примеры

1. Использование функции бинарного выбора (в реализации функции):

```
IF ABS(in1 - pattern) < ABS(in2 - pattern)
THEN Nearby_int := in1;
ELSE Nearby_int := in2;
END_IF
```

Эквивалентная запись с использованием SEL:

```
Nearby_int := SEL(ABS(in1 - pattern) < ABS(in2 - pattern), in2, in1);
```

2. Применение ограничителя и мультиплексора в логике «удовлетворенности»:

```
OUT := LIMIT(1, Drinking, 3);
Satisfaction := MUX(OUT, 'Мало', 'Норма', 'Много');
```

3. Сложное условие сравнения:

```
bAlarm := byInp1 > byInp2 AND byInp1 + byInp2 <> 0 OR bAlarm2;
```

Практическая часть

Ход работы

Задание 1. Система включает в себя два исполнительных механизма: вентилятор (out1) и котел (out2). Запуск осуществляется Пользователем путем нажатия на кнопку in1, при этом сразу включается в работу вентилятор, а спустя 5 секунд включается в работу сам котел. При необходимости остановить рабочий режим, пользователю следует нажать на кнопку останова in2. При нажатии на in2 выключается котел, а насос продолжает работать дополнительно 3 секунды. Для повторного запуска системы необходимо заново нажать кнопку запуска in1.

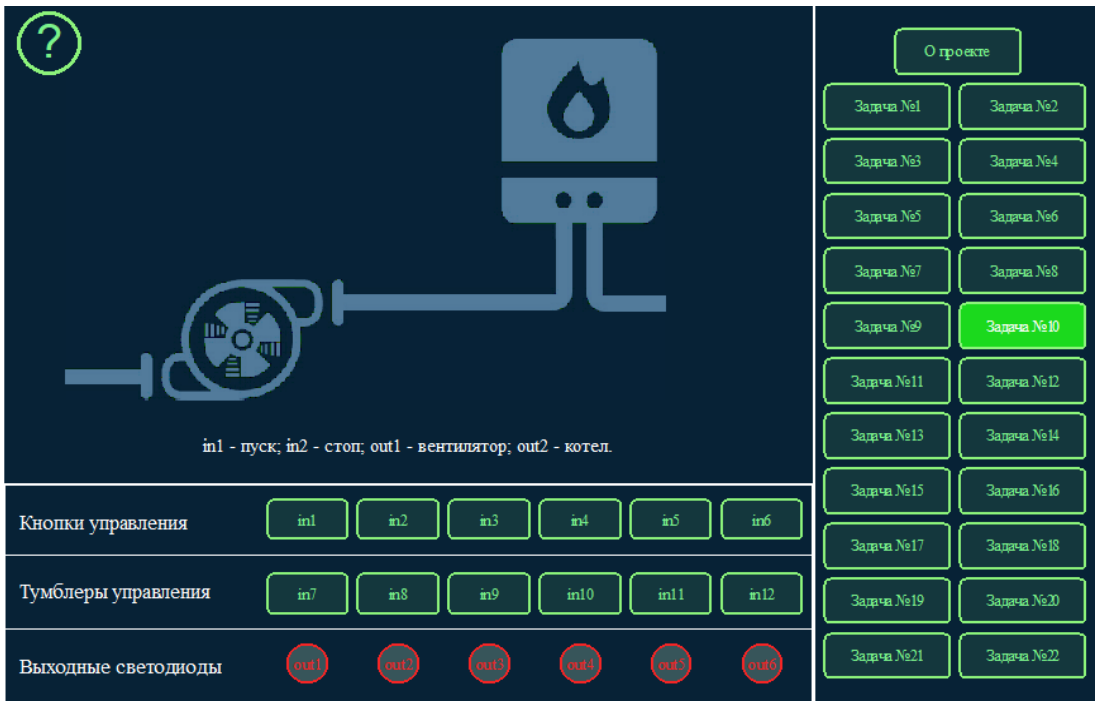

Рисунок 15 – Управление системой запуска котла

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.
3. Операторы выбора и ограничения CoDeSys.
4. Операторы сравнения CoDeSys.
5. Детекторы фронта CoDeSys.
6. Реализация изменения состояния объекта управления при помощи органов управления.
7. Вывод.

Контрольные вопросы

1. Какую логику реализует оператор SEL и какого типа должен быть его управляющий параметр G?
2. В чем разница между функциями MIN и MAX?
3. Каким образом функция LIMIT обрабатывает входной сигнал, если он выходит за границы установленного диапазона?
4. Через какую комбинацию функций MIN и MAX можно выразить действие ограничителя LIMIT?
5. Какой тип данных всегда возвращают операторы сравнения?

Лабораторная работа 7.
«Работа с массивами и строковыми переменными в языке Structured Text »

Цель работы: изучение синтаксиса объявления и способов обращения к элементам массивов, а также правил работы со строковыми данными и стандартными строковыми функциями.

Теоретическая часть

Массивы

Массивы представляют собой множество однотипных элементов с произвольным доступом. Массивы могут быть многомерными. Размерность массива и диапазоны индексов задаются при объявлении. Индексы должны быть целого типа и только положительные, отрицательные индексы использовать нельзя.

Для доступа к элементам массива применяется следующий синтаксис: <Имя_массива>[Индекс1, Индекс2, Индекс3]. Для двухмерного массива используются два индекса. Для одномерного, очевидно, достаточно одного. Если это не принципиально, используйте в массивах нумерацию с 0. В этом случае вычисление физического адреса элемента при исполнении проще. В результате код получается несколько короче.

Строки (Тип STRING)

Тип строковых переменных STRING определяет переменные, содержащие текстовую информацию. Размер строки задается при объявлении. Если начальное значение не задано, то при инициализации будет создана пустая строка. Количество необходимой памяти определяется заданным при объявлении размером строки. Для типа STRING каждый символ занимает 1 байт (WSTRING 2 байта). Строковые константы задаются между одинарных кавычек. Если длина строки STRING при объявлении не указана, то принимается значение по умолчанию – 80 символов.

В CoDeSys используются нультерминированные (как в С-компиляторах) строки. То есть под строку всегда заранее выделяется область памяти заданного максимального размера. Любая строка оканчивается нулевым байтом, который не входит в состав строки, а служит исключительно для определения конца строки функциями, оперирующими со строками. При объявлении строки необходимо задавать размер на единицу больше необходимого для символа «конец строки».

Стандартные строковые функции

Строковые функции представлены следующими инструкциями:

- **LEN(STR):** возвращает длину строки.
- **LEFT(STR, SIZE):** возвращает левую часть STR размером SIZE.
- **RIGHT(STR, SIZE):** возвращает правую часть STR размером SIZE.
- **MID(STR, LEN, POS):** возвращает часть STR с позиции POS длиной LEN.
- **CONCAT(STR1, STR2):** возвращает конкатенацию строк STR := STR1 + STR2.
- **INSERT(STR1, STR2, POS):** возвращает STR1 со вставленной STR2 в позицию POS.
- **REPLACE(STR1, STR2, LEN, POS):** возвращает STR1, заменив LEN символов с позиции POS на STR2.
- **FIND(STR1, STR2):** возвращает позицию STR2 в строке STR1. Если STR2 не найдена, возвращает 0. Нумерация позиций в строке начинается с 1.

Примеры

1. **Объявление и инициализация массивов:**

```
 XYbass: ARRAY [1..10,1..20] OF INT;
Mass1: ARRAY [1..6] OF SINT := 1,1,2,2,2,2;
 Mass2: ARRAY [1..6] OF SINT := 1,1,4(2); (*повторить последовательность 2 раза*)
Mass2d: ARRAY [1..2,1..4] OF SINT := 1,2,3,4,5,6,7,8; (*построчно*)
```

2. **Доступ к элементам и циклы:**

```
XYbass := 1;. siSample[ci] := -1; (*ci – индекс переменная*).
```

3. **Работа со строками:**

```
str2: STRING(60) := 'Протяжка';
str1 := 'Полет нормальный';
stHello := CONCAT(STR1:='Добрый ', STR2:='день');
```

Практическая часть

Ход работы

Управление освещением в доме (рис. 16). В зависимости от текущего периода суток меняется принцип работы освещения. Пользователь переключает период суток с помощью тумблера in7 (выключенное состояние тумблера соответствует ночному периоду суток). Индикатор out1 включается только в дневной период.

Дневной режим работы.

Свет в гостиной (out2) переключается с помощью тумблера in8.

Свет на кухне (out3) переключается с помощью тумблера in9, причем, если не включен свет в гостиной, светом на кухне управлять невозможно.

Свет в спальне (out4) и детской (out5) в дневное время выключен.

Ночной режим работы.

Свет в гостиной (out2) переключается с помощью тумблера in8.

Свет на кухне (out3) в ночное время выключен.

Свет в спальне (out4) можно включить с помощью тумблера in10 или in11.

Свет в детской (out5) можно включить с помощью тумблера in12, причем, если не горит свет в спальне (out4), свет в детской тоже не горит.

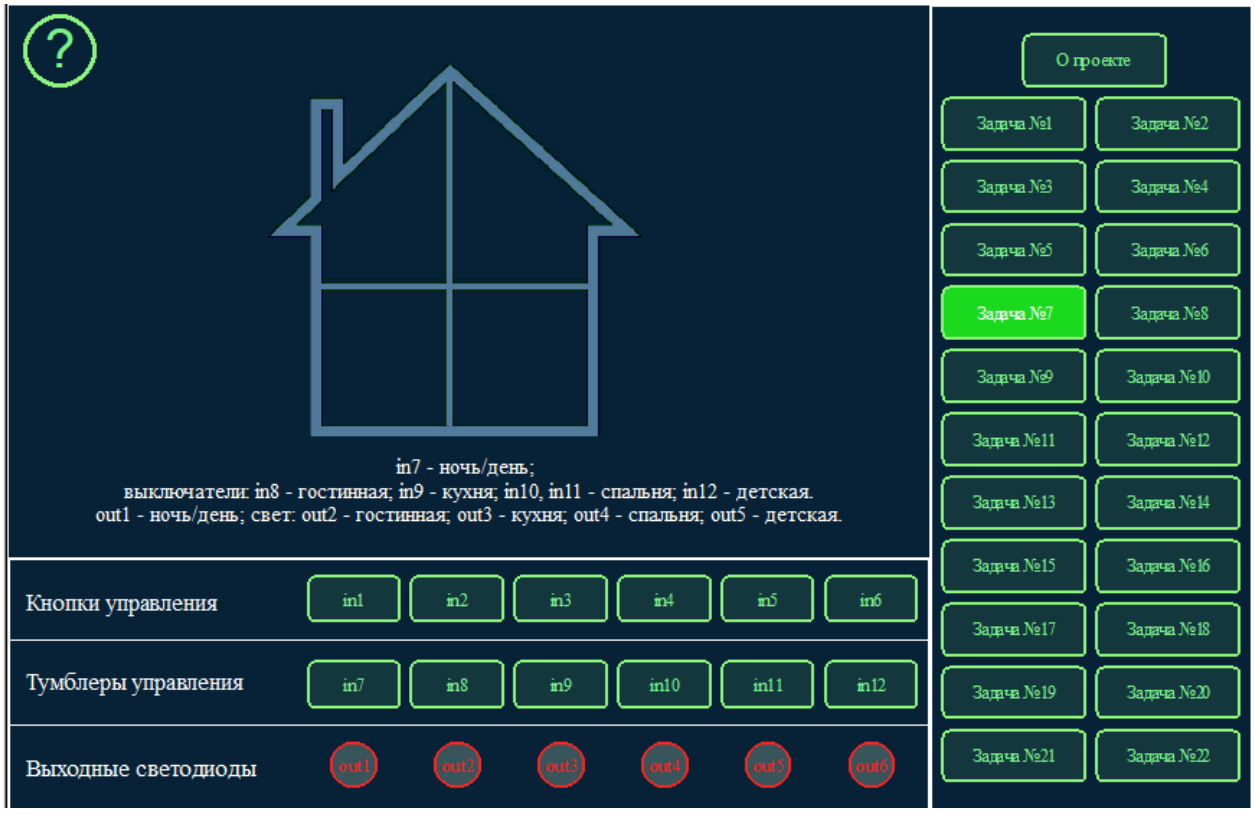

Рисунок 16 – Управление освещением в доме

Требования к содержанию отчёта

1. Название лабораторной работы.
2. Цель работы.

3. Детекторы фронтов.
4. SR-триггер: описание, структура, принцип действия.
5. RS-триггер: описание, структура, принцип действия.
6. Реализация изменения состояния объекта управления (двигателя) с использованием органов управления при возникновении нештатных происшествий (пожара либо аварии).
7. Реализация управлением системой фильтрации.
8. Вывод.

Контрольные вопросы

1. Что такое массив и какие типы элементов он может объединять?
2. Какие формальные требования предъявляются к индексам массивов в языке ST?
3. Почему при работе с массивами рекомендуется использовать нумерацию с нуля?
4. Как осуществляется инициализация многомерного массива при его объявлении?
5. Какую информацию содержат переменные типа STRING?
6. Какая длина строки принимается по умолчанию в CoDeSys, если она не указана явно?

Литература

1) Петров, И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / И. В. Петров ; под ред. В. П. Дьяконова. – Москва : СОЛОН-Пресс, 2004. – 256 с.
2) Денисенко В. В. Компьютерное управление технологическим процессом, экспериментом, оборудованием / В. В. Денисенко. – М. : Горячая линия – Телеком, 2008. – 608 с.
3) Денисенко В. В. Компьютерное управление технологическим процессом, экспериментом, оборудованием / В. В. Денисенко. – М. : Горячая линия – Телеком, 2009. – 606 с.
4) Промышленные сети передачи данных / Р. К. Нургалиев, Р. Н. Зарипов, Д. Б. Флакс [и др.] // Вестник Казанского технологического университета. – 2013. – Т. 16, № 1. – С. 252–255.
5) SCADA-системы: взгляд изнутри / Андреев Е. Б., Кунцевич Н. А., Синенко О. В. – Москва: РТСофт, 2004. – 240 с.
6) Пьявченко, Т. А. Проектирование АСУТП в SCADA-системе / Т. А. Пьявченко. – Таганрог : ТТИ ЮФУ, 2007. – 84 с.

Учебное издание

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ И ОБОРУДОВАНИЯ В ОТРАСЛИ. Программируемый логический контроллер: язык ST

Методические указания по выполнению лабораторных работ

Составители:
Константин Николаевич Ринейский
Артем Михайлович Самусев
Даниил Александрович Тёмкин

Редактор Р.А. Никифорова
Корректор А.С. Прокопюк
Компьютерная верстка Д.А. Темкин

Подписано к печати 25.06.2026. Усл. печ. листов 2,2.
Уч.-изд. листов 2,8. Заказ № 178.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 3/1497 от 30 мая 2017 г.