МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электромеханика и промышленная автоматика»

А.А. ШКРОМАДО
Р.В. ШЕСТОВ
А.Н. БИРЮКОВ

ТЕХНИЧЕСКИЕ И ПРОГРАММНЫЕ СРЕДСТВА КОМПЛЕКСНОЙ АВТОМАТИЗАЦИИ

Лабораторный практикум

Самара
Самарский государственный технический университет
2017

Печатается по решению редакционно-издательского совета СамГТУ

УДК 004.4

Т **Технические и программные средства комплексной автоматизации:** Лабораторный практикум /Сост. *А.А. Шкромадо, Р.В. Шестов, А.Н. Бирюков.* – Самара: Самар. гос. техн. ун-т, 2017. – 58 с.: ил.

Рассмотрены основные теоретические положения разработки программ комбинационного управления с применением языков стандарта МЭК 61131, приведены методические рекомендации к выполнению лабораторных работ по изучению программных средств автоматизации в рамках дисциплины «Технические и программные средства комплексной автоматизации».

Для студентов высших технических учебных заведений, обучающихся по направлению 15.03.04 «Автоматизация технологических процессов и производств».

УДК 004.4

Р е ц е н з е н т ы: доцент кафедры «Вычислительная техника» ФГБОУ ВО «Самарский государственный технический университет» *к.т.н. Мартемьянов Б. В.*

заместитель директора института ракетно-космической Техники Самарского университета, к.т.н. *Алексеев А. В.*

ВВЕДЕНИЕ

Автоматизированные системы управления технологическими процессами (АСУТП) включают различные виды обеспечения: аппаратное, программное, информационное и др. Для разработки программного обеспечения используются разнообразные инструментальные комплексы и среды, как универсальные, так и поставляемые вместе с соответствующими аппаратными средствами и предназначенные именно для них. При этом все большее распространение на современном этапе развития систем промышленной автоматизации играет использование для написания прикладных программ управления языков стандарта МЭК 61131 как универсальных и общепризнанных.

Лабораторный практикум по дисциплине «Технические и программные средства комплексной автоматизации» является логическим продолжением курса лабораторных работ по дисциплине «SCADA-системы» и посвящен изучению программного обеспечения АСУТП в части дальнейшего развития навыков разработки человеко-машинного интерфейса оператора технологических процессов и разработки прикладных программ управления, начатого в рамках курса «SCADA-системы». В качестве среды разработки используются графический редактор и редактор программ инструментальной системы SCADA Trace Mode 6.

В пособии акцент сделан на изучении принципов и особенностей реализации комбинационного (логического) управления, а также фиксации событий и обработке алармов (тревог). Первая работа позволяет реализовать комбинационное управление с применением языков Ladder Diagram (LD) и Function Block Diagram (FBD). Вторая работа расширяет область профессиональных знаний сведениями об управлении последовательностями событий и принципами разработки программ на языке Sequential Function Chart (SFC). Помимо этого, используется также язык Structured Text (ST).

Обучающимся поэтапно предлагается ознакомиться с основными теоретическими сведения по рассматриваемому материалу, после чего выполнить практическую часть лабораторной работы с подробным описанием каждого шага и его значения. Для отработки и закрепления полученных навыков обучающимся требуется выполнить самостоятельное задание. Для контроля усвоения теоретических положений в конце каждой работы предусмотрены контрольные вопросы.

ЛАБОРАТОРНАЯ РАБОТА №1

КОМБИНАЦИОННОЕ УПРАВЛЕНИЕ НА РЕЛЕЙНО-КОНТАКТНЫХ СХЕМАХ

Цель работы: освоить принципы реализации комбинационного управления с применением языка релейно-контактных схем LD и функциональных блоков FBD.

Задача работы: разработать технологический экран (мнемосхему) буферной емкости и комбинированную программу управления подпиточным насосом и клапаном сброса воды на языке Ladder Diagram, совмещенную с программой-эмулятором уровня в емкости на языках Function Block Diagram и Structured Text.

Программное обеспечение: инструментальная система Trace Mode 6.

1.1. Порядок выполнения работы:

2. Изучите теоретический сведения, необходимые для выполнения лабораторной работы.
3. Выполните практические задания согласно методическим рекомендациям, описывающим содержание и порядок работы.
4. Составьте отчет по проделанной работе, включив в него результаты в соответствии с требованиями к содержанию отчета.
5. Подготовьте ответы на контрольные вопросы.

1.2. Теоретические сведения

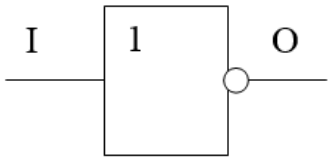
Комбинационное (логическое) управление – это управление устройствами и на основании данных от устройств, которые имеют только два возможных состояния – включено (логическая единица) или выключено (логический ноль). Это могут быть отдельные элементы (кнопки, контакты, различные реле, транзисторы в ключевом режиме работы и др.) или устройства на их основе (концевые или путевые выключатели, позиционные сигнализаторы, пускатели и др.).

В комбинационном управлении нет точного численного значения регулируемой величины, а есть только требуемое состояние исполнительных устройств, при котором контролируемый технологический параметр находится в допустимом диапазоне или имеет требуемое состояние, и которое достигается соответствующей логической комбинацией управляющих сигналов.

Ниже представлены основные логические операции, которые могут быть применены к обрабатываемым сигналам.

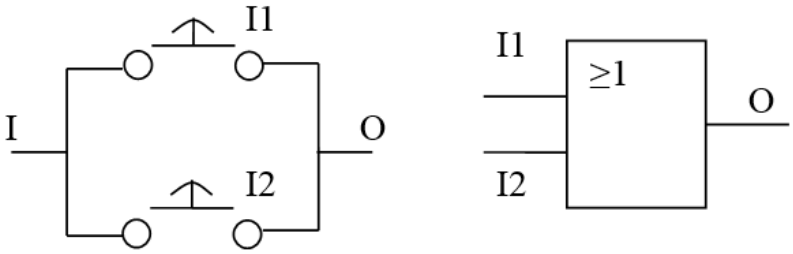
Простейшей логической операцией является ***отрицание*****,** или **инверсия**, которую выполняет элемент ***NOT («НЕ»)***. Этот элемент имеет только один вход и один выход – если входное значение *I=0*, то выходное *O=1*. Схематическое обозначение и таблица истинности элемента *NOT* показаны на рис. 1.1. При соединении с другими логическими элементами инверсия показывается маленькой окружностью на их входе либо выходе. Схематично инверсия обозначается $\overline{X}$. В языках программирования помимо оператора «*NOT*» инверсия может обозначаться как *«!»* (например, *O=NOT I* или *O=!I*; *I, O* – имя переменных входа и выхода соответственно).

Два нормально разомкнутых ключа *I1* и *I2*, соединенных параллельно, образуют элемент ***OR («ИЛИ»)***. Выполняемая ими операция называется **булевым (логическим) сложением** и обозначается ***O=I1+I2***. Результат на выходе *O=0* получается, если оба *I1* и *I2* равны 0, в противном случае результат *O=1* (рис. 1.2).



I	O
0	1
1	0

Рис. 1.1. Схематическое обозначение и таблица истинности элемента NOT («НЕ»)



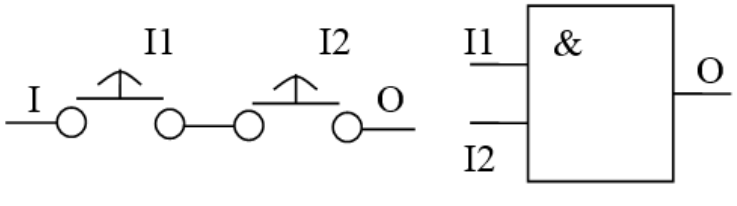
I1	I2	O
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 1.2. Обозначение и таблица истинности элемента OR («ИЛИ»)

Операция логического сложения в языках программирования помимо оператора *«OR»* может записываться как *«||»* (например, *O=I1 OR I2* или *O=I1||I2, I1* и *I2* – имена входных переменных, *O* – переменная, соответствующая выходу).

Элемент *OR* может иметь более двух входов, так как любое число ключей можно соединить параллельно. Расширение до трех ключей приводит к *O=I1+I2+I3*. Символы *«≥1»* на схематическом обозначении элемента *OR* указывает, что на его выходе будет единица при условии, что она есть хотя бы на одном входе.

Два нормально разомкнутых ключа *I1* и *I2*, соединенных последовательно, образуют элемент ***AND («И»)***, а выполняемая операция называется ***булевым умножением***. В этом случае выход *O=1* только если *I1* и *I2* оба равны 1, иначе *O=0*. Эта операция обозначается ***O=I1·I2***. Знак умножения в булевых выражениях часто опускается, также как в обычной алгебре. Аналогично элементу *OR*, элемент *AND* может иметь больше двух входов, так как любое число ключей можно соединить последовательно. Добавляя третий ключ получим *O=I1·I2·I3*. Схематическое обозначение и таблица истинности элемента *AND* показаны на рис. 1.3.



I1	I2	O
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 1.3. Обозначение и таблица истинности элемента AND («И»)

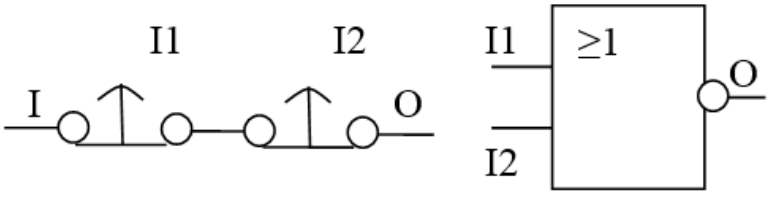
Операция логического умножения в языках программирования помимо оператора *«AND»* может записываться как *«&&»* (например, *O=I1 AND I2* или *O=I1&&I2, I1* и *I2* – имена входных переменных, *O* – переменная, соответствующая выходу).

При манипуляциях с булевыми выражениями полезны теоремы де Моргана:

$\overline{(X \cdot Y \cdot Z \cdot ...)} = \overline{X} + \overline{Y} + \overline{Z} + ...$ (Первая теорема де Моргана)

$\overline{(X + Y + Z + ...)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot ...$ (Вторая теорема де Моргана)

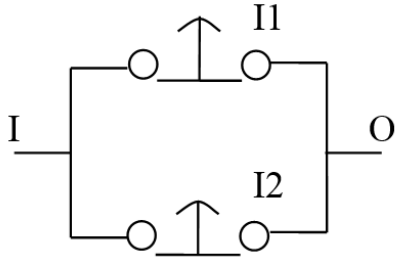
Два нормально замкнутых ключа образуют элемент ***NOR («НЕ ИЛИ»)***, т.е. цепь замкнута и проводит ток, если ни первый, ни второй ключ не приведены в действие. Согласно теореме де Моргана $O = \overline{I1} \cdot \overline{I2} = \overline{(I1 + I2)}$. Т.е. элемент *NOR* можно представить как комбинацию элементов *OR* и *NOT*, что отражено в его схематическом обозначении на рис. 1.4.

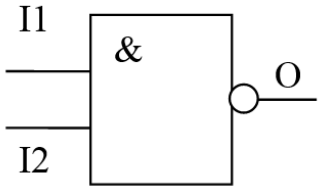


I1	I2	O
0	0	1
0	1	0
1	0	0
1	1	0

Рис. 1.4. Обозначение и таблица истинности элемента NOR («НЕ ИЛИ»)

Элемент ***NAND («НЕ И»)*** определяется так $O = \overline{(I1 \cdot I2)} = \overline{I1} + \overline{I2}$. Соответствующая цепь не проводит ток, если оба ключа *I1* и *I2* разомкнуты, если разомкнут только один ключ, то цепь остается замкнутой через другой (рис. 1.5).

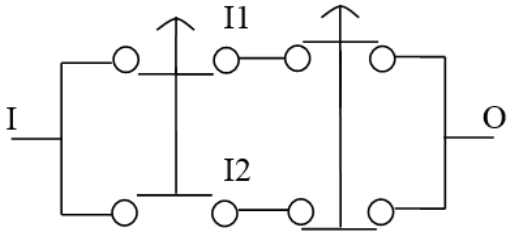


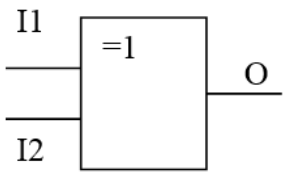


I1	I2	O
0	0	1
0	1	0
1	0	0
1	1	0

Рис. 1.5. Обозначение и таблица истинности элемента NAND («НЕ И»)

На рис. 1.6 показана цепь из двух ключей: каждый ключ состоит из двух контактов, один из которых нормально разомкнут, а другой нормально замкнут. Соответствующая операция называется ***XOR («исключающее ИЛИ»)***, а ее результат определяется выражением $O = I1 \cdot \overline{I2} + \overline{I1} \cdot I2$.





I1	I2	O
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 1.6. Обозначение и таблица истинности элемента XOR («исключающее ИЛИ»)

Эта цепь проводит ток в случае, только если либо *I1=1*, либо *I2=1*; если имеют одно и то же значение, то на выходе O=0.

Особое значение имеет сохранение значений сигналов и состояний и использование их позже в других операциях. Память реализуется на элементах с двумя устойчивыми состояниями, которые в русской терминологии называют триггерами. Выход триггера зависит не только от текущего состояния на входе, но и от предыдущего на выходе.

Простейшим и наиболее часто используемым элементом такого типа является ***SR-триггер*** (*Set-Reset* – установка-сброс). Два входа *S* и *R* (рис. 1.7) могут иметь логическое значение *«0»* либо *«1»*, однако им обоим нельзя принимать одно и то же значение одновременно. Выход обозначается *y (*может существовать и инвертированный выход $\overline{y}$). Если *S=1*, то выход изменяется на *y=1* ($\overline{y}$ *=0)* и триггер переходит в состояние «установка». Если затем вход *S* принимает значение *«0»*, то триггер «помнит», что до этого он имел значение *«1»* и

удерживает выходное значение *y=1*. Если теперь вход *R* примет значение *«1»*, то с учетом *S=0* триггер сбрасывается и на выходе *y=0* ($\bar{y}$ *=1)*. Аналогично, как и ранее, *R* может вернуться к *«0»* и состояние *y=0* останется до тех пор, пока не появится новый сигнал *S=1*.

На рис. 1.7 приведены три варианта реализации SR-триггера на двух логических элементах.

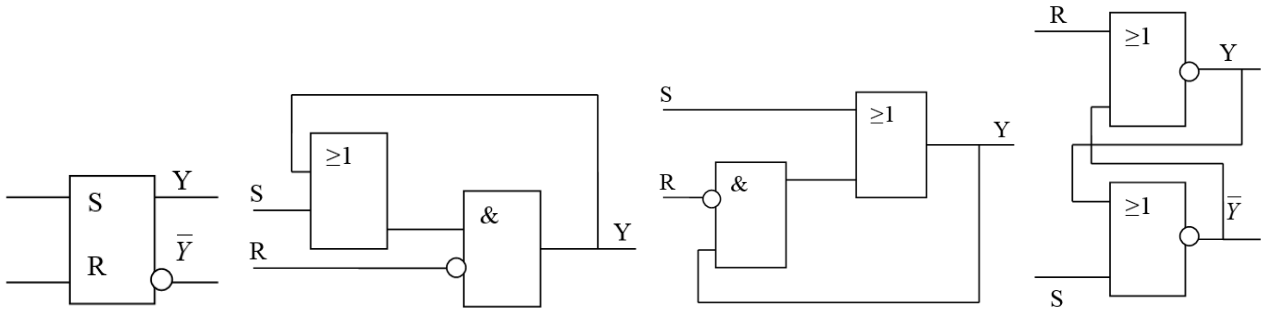


Рис. 1.7. SR-триггер и три варианта его реализации на двух элементах

Различные варианты комбинационного управления можно реализовывать, используя соответствующие логические блоки языка FBD или контакты и катушки языка LD.

1.3. Содержание и выполнение работы

Разработаем проект, в основе которого лежит комбинационное (логическое) управление парой исполнительных устройств – пускателем электродвигателя подпитывающего насоса и запорным электромагнитным клапаном сброса жидкости из емкости. Суть процесса заключается в необходимости поддержания уровня жидкости в емкости не ниже значения «минимум», но и не выше значения «максимум».

Минимум соответствует 35%, а максимум – 85% по уровню. Соответствующие данные о достижении пороговых значений обеспечиваются сигнализатором уровня (промежуточные значения не показываются) и отображаются на экране индикаторами типа «сигнальная лампа» оранжевого и красного цветов соответственно.

При снижении уровня ниже минимума включается подпитывающий насос (+2,5%/цикл); при повышении более максимума – открывается клапан сброса (-5%/цикл). Состояние устройств отображается цветом (включено – зеленый, выключено – серый). Номинально следует поддерживать уровень незначительно ниже максимума, чтобы максимально использовать объем емкости, но не превышать максимально допустимого значения. Из емкости осуществляется пере-

менный расход жидкости через регулирующий клапан отбора, для которого задается величина открытия в процентах. При 100% открытия регулирующего клапана расход составляет 3% за цикл пересчета.

Смена состояний клапана и насоса (пуск/стоп и открыто/закрыто) должны фиксироваться как событие с присвоение временной метки. Выход уровня за допустимые границы диапазона «минимум – максимум» должен фиксироваться как предупреждение и тревога соответственно в отчете тревог с фиксацией нормализации значения технологического параметра.

Возможен полный слив (сброс) жидкости из емкости через сливной запорный клапан, при котором должно блокироваться включение насоса при падении уровня ниже минимума и выдача сигнала «минимум» на экран, а в отчет тревог должна записываться информация о нормальном протекании процесса. Ручной слив инициируется двухпозиционной кнопкой на экране с указанием текущего статуса сброса.

Движение потоков жидкости при смене состояний исполнительных устройств должно отображаться поясняющей анимацией.

В качестве средства визуального контроля уровня в емкости (для облегчения контроля работы модели, в реальном процессе отсутствует) следует добавить поплавковый указатель уровня**.**

Создание проекта, узла, шаблона экрана. Откройте инструментальную систему Trace Mode 6 и создайте новый проект, нажав кнопку *«Создать новый проект»*  на панели инструментов *«Главная»*. Сохраните проект, нажав кнопку *«Сохранить текущий проект»*  на панели инструментов *«Главная»*. Выберите местоположение проекта и задайте его имя (см. Создание и сохранение проекта, ЛР№1 в [6]).

Создайте узел проекта типа RTM, нажав ПКМ на слое *«Система»* в навигаторе проекта и в контекстном меню выбрав *«Создать узел → RTM»*. (см. Создание узла проекта, ЛР№1 в [6]). Создайте шаблон экрана, аналогично нажав ПКМ на созданном узле RTM_1 и выбрав в контекстном меню *«Создать компонент → Экран»* (см. Создание шаблона экрана оператора, ЛР№1 в [6]).

Создание ГЭ «Емкость». Первоначально разместим на экране емкость (элемент 1). Для этого, выбрав *ГЭ «Емкость»*  из группы *«Объемные фигуры»,* разместите ее как показано на рисунке 1.7 (см. Создание ГЭ «Емкость», ЛР№1 в [6]). В свойствах *ГЭ «Емкость»*, используя пункты списков *«Верхний край»* и *«Нижний край»* задайте подходящую форму емкости и введите в свойство *«Высота краев (%)»* значение *«20»*.

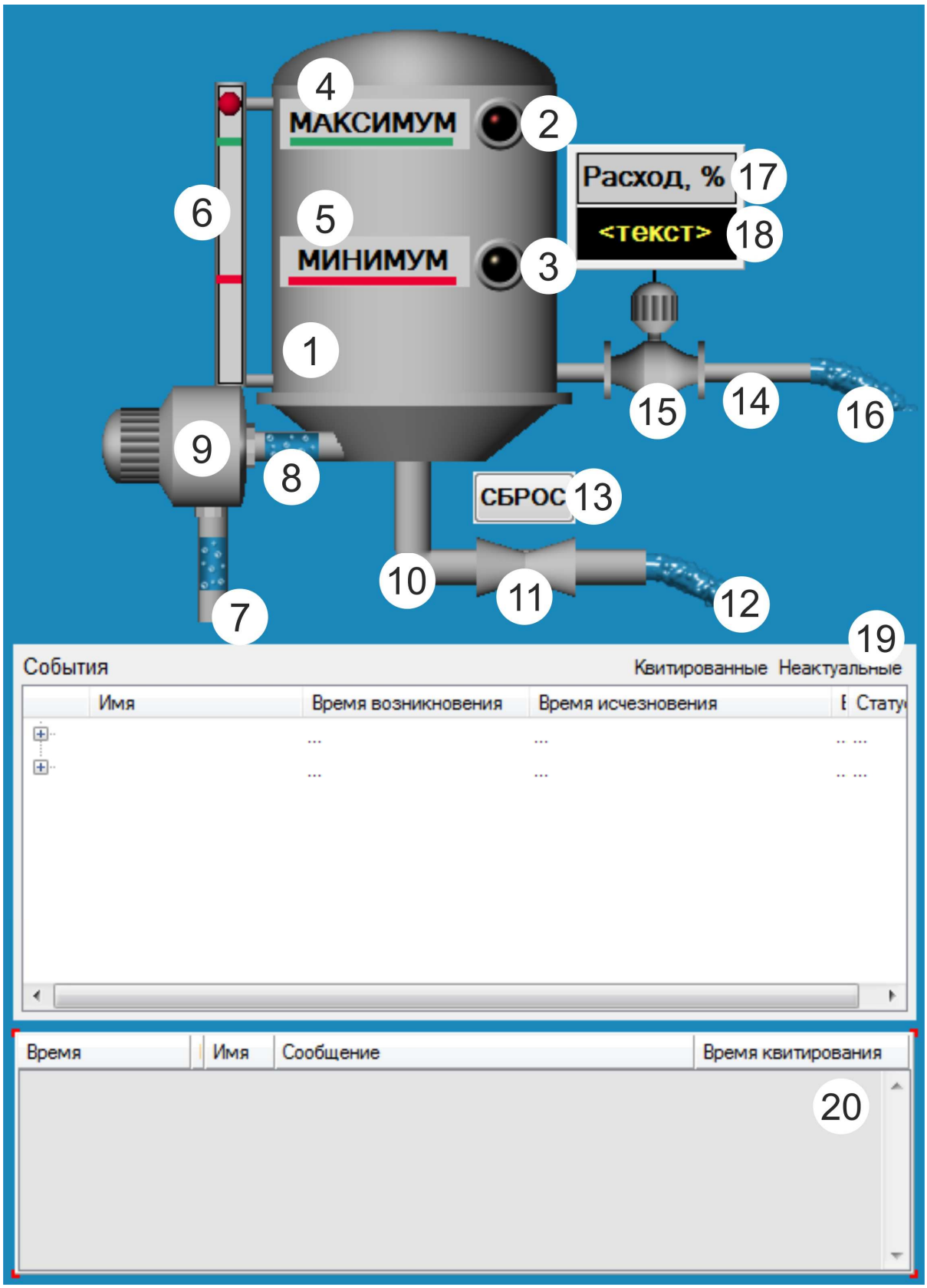


Рис. 1.7. Технологический экран оператора

Создание ГЭ «Труба». Далее покажем трубы, по которым осуществляется наполнение (элемент 7), опорожнение емкости (элемент 10) и отбор жидкости из нее (элемент 14).

Из группы *«Объемные фигуры»* аналогично *ГЭ «Емкость»* выберем *ГЭ «Труба»*  . Разместите соответствующим образом узлы труб (см. Создание ГЭ «Труба», ЛР№1 в [6]) так, чтобы трубы налива и слива имели Г-образную форму, а труба отбора была прямой (рисунок 1.7).

В свойствах *«Толщина» ГЭ «Труба»* установите толщину *«18»* для элемента 7, *«22»* для элемента 10 и *«12»* для элемента 14.

Создание ГЭ «Насос». Разместим на наливной трубе (элемент 7) насос (элемент 9). Для этого из группы *«Объемные фигуры»* выберем *ГЭ «Насос»*  (см. Создание ГЭ «Насос», ЛР№1 в [6]). Используя пункт свойств *«Форма насоса»* данного ГЭ, задайте форму, как показано на рисунке 1.7.

Для установки видимости взаимного расположения ГЭ (насос должен располагаться поверх трубы) можете воспользоваться пунктами *«Переместить наверх»*  или *«Переместить вниз»*  контекстного меню выделенного ГЭ.

Создание ГЭ «Клапан». Разместим на сливной трубе (элемент 10) запорный (электромагнитный) клапан (элемент 11). Для этого из группы *«Объемные фигуры»* выберем *ГЭ «Клапан»*  (см. Создание ГЭ «Клапан», ЛР№1 в [6]).

Аналогично разместите клапан (элемент 15) на трубе отбора (элемент 14). Для того, чтобы изменить внешний вид клапана, как показано на рисунке 1.7, и отобразить электропривод используйте значения из списков свойств *«Форма клапана»* и *«Форма привода»* соответственно.

Настройка отображения состояния насоса. Выберите ЛКМ *ГЭ «Насос»* на экране. На вкладке *«Общие свойства»*  в пункте свойств *«Базовый цвет → Вид индикации»* выберите из списка пункт *«Arg=const»* (см. Настройка отображения состояния насоса, ЛР№1 в [6])

В свойстве *«Привязка»* выполните привязку к вновь созданному аргументу *«Pump»* с типом *«IN»* и типом данных *«USINT»*.

В пункте свойств *«Если ЛОЖНО»* вместо красного цвета выберите цвет, соответствующий выключенному насосу – серый. В поле свойства *«Константа»* введите *«1»*.

Настройка отображения состояния запорного клапана. На вкладке *«Общие свойства»*  в пункте свойств *«Базовый цвет → Вид индикации»* выберите из списка пункт *«Arg=const»* (см. Настройка отображения состояния клапана, ЛР№1 в [6]).

В свойстве *«Привязка»* выполните привязку к вновь созданному аргументу *«Valve»* с типом *«IN»* и типом данных *«USINT»*.

В пункте свойств *«Если ЛОЖНО»* вместо красного цвета выберите цвет, соответствующий закрытому клапану – серый. В поле свойства *«Константа»* введите *«1»*.

Настройка отображения состояния регулирующего клапана. Цветом покажем состояние, соответствующее закрытому клапану. Поскольку закрытый клапан значит, что положение штока равно 0%, то в свойстве *«Базовый цвет → Вид индикации»* на вкладке *«Общие свойства»*  выберите из списка пункт *«Arg=const»* и выполните привязку к вновь созданному аргументу *«Flow»* с типом *«IN»* и типом данных *«REAL»* (т.к. в этом аргументе будет храниться численное значение положения штока).

В пункте свойств *«Если ЛОЖНО»* вместо красного цвета выберите зеленый цвет, а в пункте свойств *«Если ИСТИННО»* – серый. В поле свойства *«Константа»* введите *«0»*.

Создание задатчика положения штока регулирующего клапана. Статический и динамический текст. ГЭ «Текст».

Для задания (ввода) процента открытия регулирующего клапана воспользуемся ГЭ *«Текст»*. Для этого создайте статический текст *«Расход, %»* (элемент 17) с помощью инструмента *«Текст»*  (см. Создание статических надписей. ГЭ «Текст» ЛР№1 в [6]) и разместите его над клапаном (элемент 15). Установите жирное начертание, 12 кегль в свойстве *«Шрифт»* и черный цвет текста в одноименном свойстве. В свойстве *«Заливка»* выберите из списка подпункта *«Цвет заливки»* серый цвет.

Затем создайте динамическое поле для ввода значения положения штока, соответствующего расходу, (элемент 18) также с помощью инструмента *«Текст»*  , который разместите под созданной ранее подписью и выровняйте относительно статической надписи с помощью команд панели *«Топология экрана»* или воспользуйтесь командой меню *«Сервис → Тиражировать»*.

Для обеспечения индикации в свойствах динамического текста задайте привязку в подпункте *«Вид индикации»* свойства *«Текст»* к значению созданного ранее аргумента *«Flow»* (см. Создание динамических надписей. ГЭ «Текст» ЛР№1 в [6]). В свойстве *«Заливка»* выберите из списка подпункта *«Цвет заливки»* черный цвет. В свойстве *«Цвет текста»* выберите желтый цвет. Установите жирное начертание, 12 кегль в свойстве *«Шрифт»*.

Для организации ввода значения положения штока (расхода) в свойствах данного ГЭ перейдите на вкладку *«События»*  . Затем нажмите ПКМ на

свойстве *MousePress* (см. Создание кнопок управления исполнительными устройствами. ГЭ «Кнопка» ЛР№1 в [6]) и в контекстном меню выберите пункт *«Передать значение»*. В свойстве *«Тип передачи»* выберите из списка значение *«Ввести и передать»*. Свойства *«Значение»* оставьте пустым, т.к. значение будет вводиться с клавиатуры. В свойстве *«Результат»* нажмите на «…» и выберите в редакторе аргументов созданный ранее аргумент *«Flow»*.

Для повышения наглядности оба *ГЭ «Текст»* можно поместить на подложку – *ГЭ «Рамка»*  , находящуюся в группе *«Прямоугольники»*  . Для установки видимости взаимного расположения ГЭ воспользуйтесь пунктами *«Переместить наверх»*  или *«Переместить вниз»*  контекстного меню выделенного ГЭ.

Создание кнопки управления клапаном сброса. ГЭ «Кнопка».

Выберите на панели ГЭ инструмент *«Кнопка»*  . Разместите кнопку (элемент 13) над клапаном сброса (элемент 11), как показано на рисунке 1.7.

В поле свойства *«Текст»* на вкладке *«Общие свойства»*  введите надпись «СБРОС». В свойстве «*Шрифт*» установите жирное начертание, 10 кегль, выбрав их в списке (см. Создание кнопок управления исполнительными устройствами. ГЭ «Кнопка», ЛР№1 в [6]).

Оформим копку в виде двухпозиционной, т.е. и открытие, и закрытие клапана сброса будем осуществлять одной кнопкой. Для этого в свойстве *«Два состояния»* выберите значение *«TRUE»*. В свойстве *«Привязка»* выберите вновь созданный аргумент *«Empty»* с типом *«IN»* и типом данных *«USINT»*.

Таким образом, аргумент *«Empty»* хранит значение команды управления клапаном сброса (0 или 1), а значение аргумента *«Valve»* отображает фактическое состояние запорного клапана.

Переключитесь на вкладку *«События»*  . Нажмите ПКМ на свойстве «*MousePress*» (описывает действие, происходящее при нажатии оператором ЛКМ на кнопке) и в контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«1»* (команда открытия клапана на сброс). В свойстве *«Результат»* выберите в редакторе аргументов созданный ранее аргумент *«Empty»*.

Аналогично настроим противоположное действие – закрытие клапана сброса – и свяжем его с отпусканием левой кнопки мыши на кнопке сброса. Нажмите ПКМ на свойстве *MouseReleas* и в контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«0»* (команда закрытия клапана). В

свойстве *«Результат»* выберите в редакторе аргументов созданный ранее аргумент *«Empty»*.

Создание поплавкового уровнемера. ГЭ «Цилиндр». Несмотря на то, что сигнализатор уровня не сообщает само значение уровня, а только срабатывает при достижении пороговых значений минимума и максимума, для отображения уровня создадим аналог поплавкового уровнемера.

Для этого слева от *ГЭ «Емкость»* разместим *ГЭ «Цилиндр»* (составной элемент 6) из группы *«Объемные фигуры»* (данный ГЭ выбран в группе по умолчанию). Задание размеров и положения производится аналогично *ГЭ «Емкость»*. Соедините *ГЭ «Цилиндр»* с ГЭ *«Емкость»* с помощью пары таких же небольших цилиндров.

Создание поплавкового уровнемера. ГЭ «Прямоугольник». Динамическая заливка. Разместим над *ГЭ «Цилиндр»* ГЭ «Прямоугольник». Для этого в группе *«Прямоугольники»* выберите из раскрывающегося списка *ГЭ «Прямоугольник»* и разместите данный *ГЭ* так, как показано на рисунке 1.7 (составной элемент 6).

Реализуем динамическую заливку уровнемера, соответствующую уровню воды в емкости (см. Создание смотрового окна. ГЭ «Прямоугольник», ЛР№1 в [6]). Переключитесь на вкладку *«Динамическая заливка»*. Раскройте пункт свойств *«Слои → Слой»* и в подпункте *«Привязка»*, нажав на *«…»*, создайте и выберите в *табличном редакторе аргументов* аргумент *«Уровень» с типом «IN» и типом данных «REAL»*. Нажмите кнопку *«Готово»*.

Создание поплавкового уровнемера. ГЭ «Сфера». Одной из возможностей Trace Mode является динамическая трансформация объектов экрана: перемещение, изменение размеров или поворот. Разместим «поплавок», который будет перемещаться вместе с изменением уровня в смотровой трубке уровнемера.

Сначала разместим сам поплавок. Для этого «на дне» прямоугольника разместим *ГЭ «Сфера»* (составной элемент 6) из группы *«Объемные фигуры»*. В свойстве *«Базовый цвет»* сферы установите красный цвет.

Динамическая трансформация. Перемещение. Перейдите на вкладку *«Динамическая трансформация»* данного ГЭ, раскройте группу *«Перемещать»*, установив соответствующий флаг, и установите флаги *«Перемещать плавно»* и *«Использовать значения промежуточных узлов»*, что позволит перемещать «поплавок» плавно с изменением уровня. Перемещение происходит вдоль траектории, начало которой *(0)* отмечено красным маркером (см. экран)

при раскрытой группу «перемещать», а конец *(100)* – оранжевым маркером. Сама траектория показана белой линией. Следует учесть, что точкой отсчета перемещающейся фигуры является не ее центр, а левый верхний угол ограничивающего ее прямоугольника.

Граничные значения маркеров перемещения соответствуют выбранной в лабораторной работе системе отсчета в процентном представлении. Если же требуется изменить диапазон, то следует навести указатель мыши на соответствующий маркер, ввести численное значение узла в соответствующее поле и нажать кнопку *«Установить для узла»*. После этого обязательно нужно нажать кнопку *«Рассчитать значения узлов»* для расчета промежуточных значений при перемещении вдоль траектории.

Создание ГЭ «Линия». ГЭ «Линия» будем использовать для отрисовки отметок «Максимум» и «Минимум» на емкости. Выберите инструмент *«Линия»*  и разместите две горизонтальные линии, как показано на рисунке 1.7.

Линия представляет собой отрезок, у которого есть точка начала, устанавливаемая ЛКМ, и точка окончания, устанавливаемая ЛКМ или ПКМ. Удерживая CTRL при установке конечной точки можно провести прямую с углом наклона, кратным 45°.

В пункте *«Цвет»* свойства *«Контур»* ГЭ «Линия» выберите зеленый для линии максимума и красный для линии минимума. В пункт *«Толщина»* свойства *«Контур»* обеих линий введите «5». Скопируйте линии и разместите их на том же уровне на уровнемере, уменьшив их длину.

Создание подписей. Статические надписи. ГЭ «Текст». Создание статических надписей. ГЭ «Текст». Подпишем пороговые значения *«Максимум»* и *«Минимум»*.

Выберите на панели ГЭ инструмент *«Текст»*  . Разместите текстовые ГЭ как показано на рисунке 1.7. В свойствах верхнего *ГЭ «Текст»* (элемент 4) в поле свойства *«Текст»* на вкладке *«Общие свойства»* введите надпись *«Максимум»*. В свойстве *«Шрифт»* установите жирное начертание, 12 кегль, выбрав их в списке. Измените цвет текста в одноименном свойстве на черный.

Уберите рамку текста. Для этого в подпункте *«Стиль»* свойства *«Контур»* на вкладке *«Общие свойства»*  нажмите ЛКМ на изображении линии и выберите в списке пустой пункт (без линии).

Аналогично задайте свойства и измените надпись для нижнего *ГЭ «Текст» (элемент 5)*, отредактировав свойство *«Текст»* на вкладке *«Общие свойства»*  .

Добавление анимационных клипов в проект и размещение на экране. Видеоклип. Нажмите ПКМ на слое *«Ресурсы»* и в контекстном меню выберите *«Создать группу → Анимация»*. Затем нажмите ПКМ на группе *«Анимация»* и выберите в контекстном меню *«Создать компонент → Библиотека_видеоклипов»* (см. Добавление анимационного клипа в проект и размещение на экране. Видеоклип, ЛР№2 в [6]). Откройте библиотеку видеоклипов на редактирование. Нажмите ПКМ в рабочем поле библиотеки и в контекстном меню выберите пункт *«Импортировать»* .

Из папки «C:\Program Files (x86)\AdAstra Research Group\Trace Mode IDE 6 Base\Lib\Animation\Lamps» (на примере Windows 7) выберите файл lamp_orange_top и lamp_red_top, а затем из папки «…\Loading_fluid» файл fluid_loading_left_blue и из папки «…\Flows_of_fluid» файл fluid_flow_up_blue. Все анимационных клипа будут добавлены в библиотеку.

Чтобы использовать добавленные в библиотеку видеоклипы на экране на панели ГЭ в *РПД* раскройте группу *«Ресурсы»* и выберите ресурсную библиотеку видеоклипов . Перетащите анимационные клипы на экран, разместив их как показано на рисунке 1.7: элемент 2 – красная лампа, элемент 3 – оранжевая лампа, элемент 8 – поток голубой жидкости (получается поворотом на 90° – см. Создание ГЭ «Клапан», ЛР №1 в [6]), элементы 12 и 16 – голубая жидкость, выливающаяся слева.

Для облегчения позиционирования используйте кнопки панели *«Топология экрана»*. В случае, если один из элементов закрывает другой, используйте кнопки *«Переместить вниз»* и *«Переместить вверх»* для изменения порядка элементов.

В пункте свойств *«Привязка»* красной лампы (элемент 2) задайте привязку к вновь созданному аргументу *«Max»*, а оранжевой (элемент 3) – к вновь созданному аргументу *«Min»*.

В пункте свойств *«Привязка»* потока голубой жидкости (элемент 8 и его копия) задайте привязку к созданному ранее аргументу *«Pump»*. В раскрывающемся списке свойства *«Показывать при остановке»* выберите *«False»*, чтобы при выключенном насосе анимация текущей жидкости не отображалась.

В пункте свойств *«Привязка»* голубой жидкости, выливающейся слева (элемент 12) задайте привязку к созданному ранее аргументу *«Valve»*. В раскрывающемся списке свойства *«Показывать при остановке»* выберите *«False»*, чтобы при закрытом запорном клапане анимация выливающейся жидкости не отображалась. То же сделайте для элемента 16, установив привязку к аргументу *«Flow»*.

Динамическая трансформация. Изменение размеров. Изменение размеров служит хорошим способом обращения внимания оператора на количественное изменение значения технологического параметра.

Изменим размер анимации количества отбираемой из емкости жидкости в зависимости от величины ее расхода. Для этого в свойствах элемента 16 перейдите на вкладку *«Динамическая трансформация»* , раскройте группу *«Масштабировать»*, установив соответствующий флаг, и выберите в списке *«Привязка»* аргумент *«Flow»*. Масштабирование будем выполнять прямо пропорционально расходу, поэтому в поля *«Нач. размер»* и *«Значение»* введите *«1»*, а в *«Кон. размер»* и *«Значение»* введите «100». На экране красным маркером (при активном режиме редактирования масштабирования) отмечена точка, относительно которой изменяется размер объекта – центр масштабирования. В данном случае центр масштабирования практически совпадает с левым верхним углом ограничивающего ГЭ прямоугольника при размещении ГЭ так, как если бы он не менял размер (рисунок 1.7). Соответствующие флаги *«Горизонтально»* и *«Вертикально»* определяют направление изменения размеров элемента, поэтому при симметричном масштабировании, как в данном случае, остаются установленными.

Фиксация событий. ГЭ «Событие». Для отображения различных событий в реальном времени и фиксации их с помощью каналов класса *«Событие»* используется *ГЭ «Событие»* (элемент 19, рисунок 1.7). Для его размещения раскройте группу *«Таблицы»* и выберите соответствующий ГЭ . Размещение производится стандартным способом.

В качестве фиксируемых событий будем рассматривать включение подпитывающего насоса и открытие сливного запорного клапана.

На вкладке *«Осн. свойства»* в свойстве *«Цвета»* для пункта *«Событие возникло»* замените красный цвет на зеленый, т.к. контролируемые события не относятся к категории опасных.

Перейдите на вкладку *«Привязки»*. Нажмите ПКМ на свойстве *«Привязки»* и выберите в контекстном меню пункт *«Привязка»*. Создайте две привязки и, нажав на *«…»*, свяжите их с парой вновь созданных аргументов *«Pump_соб»* и *«Valve_соб»* с типом *«IN»* и типом данных *«REAL»* (аргументы, фиксирующие события, всегда создаются с такими типом и типом данных). Данные аргументы будут хранить события по включению/отключению насоса и открытию/закрытию клапана.

На экране путем перетаскивания границы измените ширину столбцов таким образом, чтобы скрыть столбцы *«Кодировка», «Статус», «Комментарий»*.

Данный ГЭ предусматривает фильтрацию выводимых событий: при нажатии кнопки *«Квитированные»* выводятся события, получение которых (прочтение) оператор подтвердил нажатием CTRL+ЛКМ на соответствующем событии; при нажатии кнопки *«Неактуальные»* выводятся события, уже не активные в данный момент.

Тревоги. Настройка узла. Для фиксации тревожных сообщений служит отчет тревог. В отчет тревог попадают данные каналов, для которых настроено архивирование в отчет тревог узла (см. Создание и настройка базы каналов). В качестве таковых будем рассматривать данные каналов *«Max»* (превышение максимального уровня) и *«Min»* (снижение уровня жидкости ниже минимума).

В первую очередь необходимо включить ведение отчета тревог для узла. Чтобы сделать это нажмите ПКМ на узле RTM_1 в навигаторе проекта и в контекстном меню выберите пункт  *«Редактировать»*. В открывшемся бланке свойств редактирования узла перейдите на вкладку *«Отчет тревог / Дамп / Параметры»* и в блоке *«Отчет тревог»* введите в поле *«Имя файла»* текст *«alarms.txt»* (имя текстового файла может быть любым), в *«максимум записей»* – *«50»*. Также установите состояние «TRUE», включив генерацию отчета. Если в списке «считывать при старте» выбрать «TRUE», то при следующем запуске профайлера в отчет на экран оператора будут выводиться тревожные сообщения с предыдущего запуска.

Тревоги. ГЭ «Отчет тревог узла».

Для отображения сообщений отчета тревог узла, удовлетворяющих определенным условиям, и их квитирования, применяется *ГЭ «Отчет тревог узла»*. Для размещения данного элемента выберите из группы *«Отчет тревог»*  *ГЭ «Отчет тревог узла»*  и разместите его стандартным способом. Настройте отображение столбцов как показано на рисунке 1.7 (элемент 20) путем перетаскивания их границы.

Изменим стандартные цвета выводимых сообщений. Для этого перейдите на вкладку *«Цвета»* и в пункте *«Цвета по умолчанию»* выберите *«False»*. Для пункта свойств *«Информация»* выберите зеленый цвет.

Создание словарей сообщений. Словари сообщений позволяют заменить техническую информацию, выводимую в отчет тревог, на более понятные оператору комментарии. Для каждого типа каналов создается свой тип словарей сообщений.

Для их создания нажмите ПКМ на узле в навигаторе проекта и в контекстном меню выберите пункт «*Создать группу → Словари сообщений*». Откройте данную группу двойным нажатием ЛКМ и, нажав ПКМ на самой группе

либо в правой части навигатора проекта, в контекстном меню выберите пункт *«Создать компонент → Словарь для HEX16»*, т.к. каналы *«Max»* и *«Min»* являются каналами именно этого класса.

Переименуйте созданный словарь в *«Максимум»* и откройте его на редактирование двойным нажатием ЛКМ. Обычно для каналов *HEX16*, если не выполнена привязка специально к конкретному биту, по умолчанию привязка выполняется к нулевому биту, поэтому для строки с комментарием «*0bit_Off»* изменим текст на *«Уровень в норме»* и выберем из списка категорию *««<I> Информация»* (тревога пропала, информирование), а для строки с комментарием «*0bit_On»* введем текст *«Превышен максимальный уровень»* и выберем категорию *«<A> Тревога»* (тревога активна). Редактирование осуществляется двойным нажатием ЛКМ на соответствующей строке словаря.

Аналогично создайте словарь для HEX16 с именем *«Минимум»* и для строки с комментарием «*0bit_Off»* измените текст на *«Уровень в норме»* и выберем из списка категорию *«<I> Информация»*, а для строки с комментарием «*0bit_On»* введем текст *«Достигнут минимальный уровень»* и выберем категорию *«<W> Предупреждение»*.

Направление передачи можно оставить без изменений, т.к. направление *«AR»* означает запись в отчет тревог, что и требуется.

Разработка комбинированной программы управления на языке LD и программы-эмулятора на языках FBD и ST.

Для реализации логического управления зачастую используется язык Ladder Diagram, заменивший аналогичные физические релейно-контактные схемы.

Язык содержит блоки – контакты и катушки. Контакты можно рассматривать как входные данные (датчики) системы управления. Это могут быть релейные выходы датчиков, сигналы сигнализаторов, кнопки и др. Катушки чаще всего соответствуют исполнительным устройствам, примерами которых могут служить реле, электродвигатель, лампочка и т.д. Наряду с блоками LD (контактами и катушками) возможно использование в программе и блоков FBD.

Контакты и катушки располагаются между парой основных вертикальных шин. Левая является аналогом шины питания, поэтому всегда имеет состояние «логическая единица», правая – «земля». Кроме того, в программу могут добавляться дополнительные вертикальные шины, на которые могут замыкаться выходы блоков, расположенных один над другим. Каждая управляющая цепь (ветвь) должна содержать хотя бы один выходной элемент, в противном случае источник напряжения будет накоротко замкнут на землю.

Логическое состояние правой шины, равно как и дополнительных шин, формируется как логическая сумма *(OR)* приходящих на нее значений сигналов на данном этапе пересчета.

Точно также, как и в случае с FBD-блоками, LD-блоки при их размещении в рабочем поле редактора последовательно получают номера, которые отображаются внизу блока. На блоке, который выполняется первым в программе, после его номера отображается символ «*B*»; на блоке, который выполняется последним, – символ *«E»*.

Номера следующих выполняемых блоков определяются автоматически при соединении входов и выходов блоков. Номер следующего выполняемого блока указывается внизу блока после двоеточия. При размещении блоков и задании связей номера следующих выполняемых блоков автоматически устанавливаются таким образом, чтобы при запуске программы первыми выполнялись блоки (в соответствии с их номерами), расположенные в первом (самом левом) столбце диаграммы, затем – во втором и т.д.

Таким образом, выполнение диаграммы происходит по столбцам «слева-направо» и «сверху-вниз», а если значение переменной претерпевало несколько изменений по ходу выполнения программы, то в аргумент будет выведено последнее ее состояние (в соответствии с порядком выполнения). Это свойство можно использовать для установления приоритетов, т.к. ветвь, расположенная ниже, будет задавать окончательное значение переменной, выводимое в аргумент, т.е. более высокий приоритет.

Напомним, что последовательное соединение блоков реализует операцию логического умножения *(«AND»)*, а параллельное – логического сложения (*«OR»)*. Примеры получения других логических комбинаций представлены в теоретических сведениях к данной лабораторной работе.

Все блоки LD имеют один вход (отрезок слева), один выход (отрезок справа) и одну связанную переменную. ***Связанной переменной*** называется переменная, от значения которой зависит выполняемое блоком действие или значение которой устанавливается в процессе выполняемого блоком действия. Связанная переменная задается пользователем. До тех пор, пока переменная не задана, над блоком отображаются три звездочки *«***»*.

Для соединения входов и выходов двух блоков, необходимо навести указатель на вход или выход блока и, когда он примет форму  , нажать ЛКМ и, не отпуская ее, протянуть указатель ко входу нужного блока и отпустить ЛКМ. Будет создана связь в виде линии. При перемещении блоков связи сохраняются.

Приступим непосредственно к созданию программы. Нажмите ПКМ на созданном узле RTM_1 и выберите в контекстном меню *«Создать компонент → Программа»*. Двойным нажатием ЛКМ на канале вызова программы, откройте ее на редактирование в *РШП*.

Нажмите ЛКМ на пункте *«Аргументы» РШП* и создайте в табличном редакторе аргументов программы семь аргументов (см., например, Разработка программы-эмулятора изменения уровня в емкости, ЛР№1 в [6]), с параметрами *Pump IN/OUT USINT, Valve IN/OUT USINT* («*IN/OUT*»=«*OUT*» в данном случае, т.к. это выходные аргументы программы, через которые осуществляется управление исполнительными устройствами); *Level OUT REAL* (тип *«OUT»*, т.к. это вычисленное программой-эмулятором значение уровня – выходной аргумент функции)*; Flow IN REAL* (тип «*IN*», т.к. это вводимое оператором значение расхода отбираемой жидкости – входной аргумент функции); *Empty IN USINT* (тип «*IN*», т.к. это вводимая оператором команда сброса жидкости, исходные данные для работы программы управления); *Max IN/OUT USINT, Min IN/OUT USINT (*тип «*IN/OUT*», т.к. *IN* – это ввод данных от сигнализатора уровня, исходные данные для работы программы управления; *OUT* – выходные данные сигнализатора уровня).

Нажмите ЛКМ на заголовке программы. В появившемся окне выбора языка программирования выберите язык графический язык *LD*. Нажмите в окне выбора языка кнопку *«Принять»*.

LD-программа может выступать в роли основной программы, функции и функции-блока. Написание программы представляет собой последовательное размещение библиотечных блоков, контактов и катушек, выполняющих определенные функции и их связывание между собой.

Для открытия библиотеки в *РШП* нажмите на панели кнопку «*Показать/скрыть палитру LD блоков»*  . Откроется библиотека функциональных блоков.

Напишем управляющую часть программы. Откройте в библиотеке вкладку *«Контакты»* и перетащите в *РШП*, удерживая ЛКМ, разомкнутый контакт | |. Нормально разомкнутый контакт («нормально» означает «в отсутствии управляющего сигнала») пропускает ток в состоянии ассоциированной переменной *«TRUE»* (контакт замкнут) и обесточивает цепь в состоянии *«FALSE»* (контакт разомкнут), т.е. реализует зависимость: если *var≠0* и *in≠0*, то *out=1*; если *var≠0*, а *in=0*, то *out=0*; если *var=0*, то *out=0*, где *var* – ассоциированный с LD-блоком аргумент, *in* – вход LD-блока, *out* – выход LD-блока.

Дважды нажмите на блок и выберите связанную переменную *«Min»* из списка (рис. 1.8). Скопируйте (*CTRL+C*) контакт и разметите копию *(CTRL+V)* ниже. Установите аналогично вышеописанному для этого контакта ассоциацию с переменной *«Max»*.

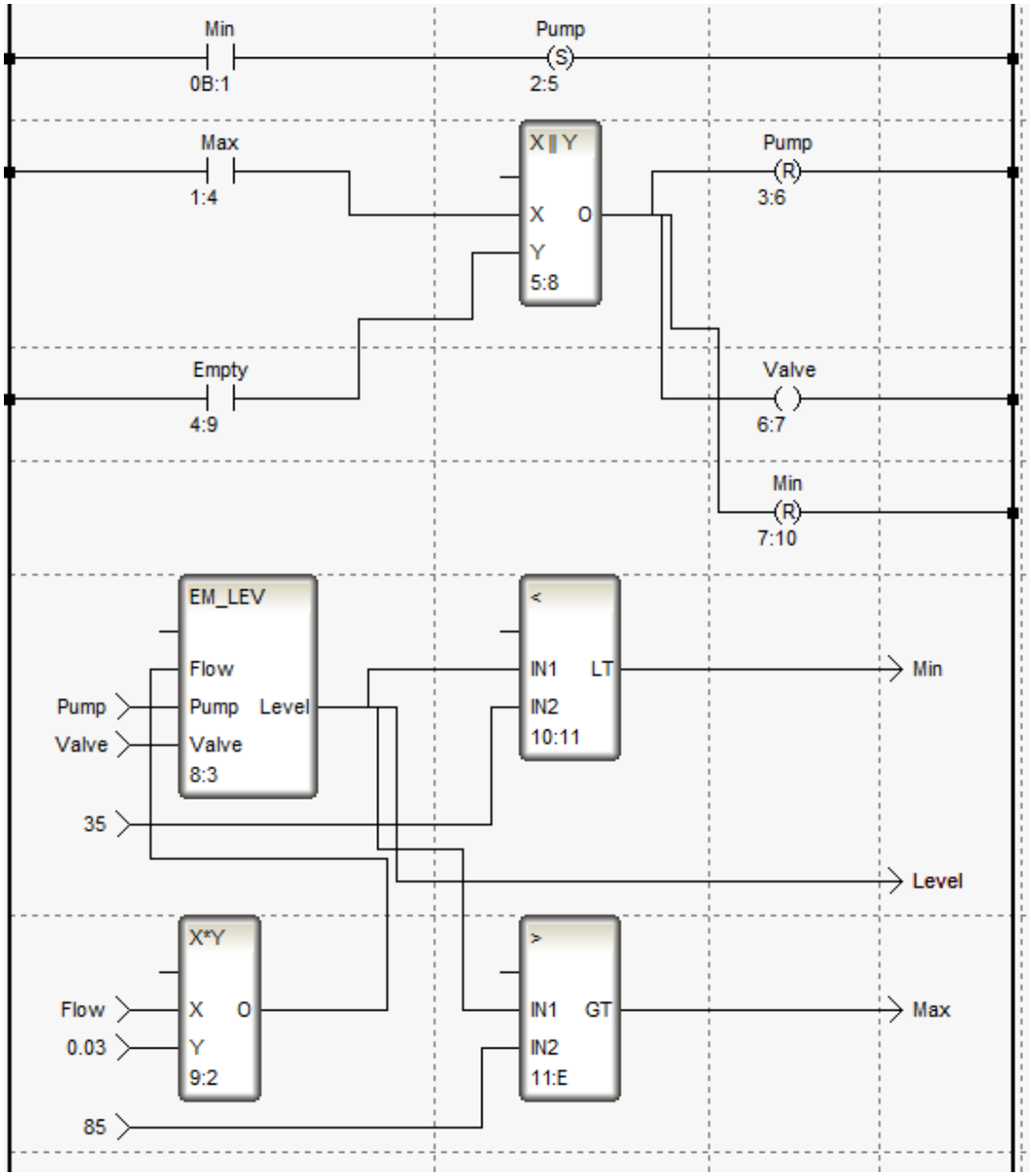


Рис. 1.8. Комбинированная программа управления подпиточным насосом и клапаном сброса воды на языке LD, совмещенная с программой-эмулятором уровня в емкости на языках FBD и ST

Переключитесь на вкладку *«Катушки»* и разместите в поле *РШП* катушку установки *(S)* в цепи контакта *«Min»*. Катушка установки соответствует обмотке установки *(S – Set)* поляризованного реле, переключаемого выдачей импульсов тока, что позволяет реализовать запоминание значения аргумента. Дважды нажмите на элемент и выберите связанную переменную *«Pump»* из списка.

Катушка установки реализует следующий алгоритм: если *in≠0,* то *Pump=out=1;* в дальнейшем аргумент *«Pump»* не зависит от значения входа, *out=1*.

Соедините контакт и катушку между собой, а также левую вертикальную шину с контактом, а катушку – с правой вертикальной шиной. Соединение осуществляется протягиванием с нажатой ЛКМ от выхода контакта ко входу катушки, от левой шины ко входу контакта, от выхода катушки к правой шине.

Аналогично перенесите в *РШП* катушку сброса *(R),* разместите ее в цепи контакта *«Max»* и свяжите с переменной *«Pump»*. Катушка сброса соответствует обмотке сброса *(R – Reset)* поляризованного реле, переключаемого выдачей импульсов тока, что позволяет реализовать сброс запомненного значения аргумента. Реализует следующий алгоритм: если *in≠0*, то *Pump=0*, а *out=1*; в дальнейшем аргумент *«Pump»* не зависит от значения входа, *out=1*.

Таким образом, срабатывание (замыкание) контакта *Min=1* (сигнал «минимум» от сигнализатора уровня) приводит к включению насоса (*Pump=1*) и насос остается включенным даже при исчезновении сигнала минимум (*Min=0*, размыкание контакта, уровень в норме) благодаря работе катушки установки вплоть до момента достижения максимального уровня (*Max=1,* замыкание контакта, сигнал «максимум» от сигнализатора уровня), при котором становится активной катушка сброса и происходит отключение насоса (*Pump=0*).

Поскольку отключение насоса должно происходить не только при достижении максимального уровня (*Max=1,* замыкание контакта, сигнал «максимум» от сигнализатора уровня), но и при открытии клапана слива по команде оператора (т.е. реализуется условие логического *«ИЛИ»*), то разместим для обнаружения подачи команды еще один нормально разомкнутый контакт | | (*Empty=1*, замыкание контакта, команда на слив; *Empty=0*, размыкание контакта, снятие команды на слив), создав третью горизонтальную цепь, и связав его с переменной *«Empty»*. Для выполнения логического сложения *«ИЛИ»* можно использовать вспомогательную вертикальную шину или логический блок X||Y. Используем второй вариант.

Откройте вкладку *«Логические функции»* и перенесите в *РШП* блок *«Логическое сложение»* X||Y, расположив его справа от контактов. Выходы контактов *«Max»* и *«Empty»* соедините со входами *(X)* и *(Y)* блока *X||Y*. Выход *(O)* блока *X||Y* соедините со входом катушки сброса *(R),* а выход катушки – с правой шиной.

Так как при достижении максимального порога (*Max=1,* сигнал «максимум» от сигнализатора уровня) необходимо не только отключить подпитываю-

щий насос (*Pump=0*), но и открыть сливной клапан (*Valve=1*), и этот же сливной клапан должен открываться соответственно при подаче команды на слив оператором *(Empty=1),* то реализуем это с помощью обычной катушки, расположенной после учитывающего эту пару условий блока логического сложения *X||Y*. Катушка реализует алгоритм: если *in≠0*, *var=out=1*; если *in=0*, *var=out=0.*

Перейдите на вкладку *«Катушки»* и разместите в *РШП* катушку () справа от блока логического условия *X||Y*. Свяжите данный LD-блок с переменной *«Valve»*. Свяжите выход *(O)* блока *X||Y* со входом катушки ( ), а выход катушки – с правой шиной (рис. 1.8).

По условию необходимо блокировать выдачу сигнала «минимум» от сигнализатора уровня *(Min=0)*, если потупила команда на слив от оператора *(Empty=1)*. Для блокировки удобно использовать катушку сброса *(R)*, поскольку данный тип катушки после срабатывания больше не зависит от изменения входного сигнала (как было бы при применении, например, инверсной катушки). Логично было бы соединить катушку сброса *(R)* с разомкнутым контактом *«Empty»*, но в этом случае изменение переменной *«Min»* данной катушки не будет последним в цикле выполнения программы и будет перезаписано в дальнейшем. Поэтому соединим вход катушки сброса *«Min»* с выходом FBD-блока *X||Y*. В этом случае данное значение переменной *«Min»* будет окончательным, т.к. это последний исполняемый блок программы (это видно по номеру блока *«7:E»*), и будет записано в соответствующий выходной аргумент программы.

Напишем эмуляционную часть программы. Для расчета уровня в зависимости от состояния исполнительных устройств, используем написанную ранее программу на языке ST с небольшими доработками (см. Редактирование шаблона программы «Регулятор». Комбинирование языков МЭК 61131-3. Функции, ЛР№3 в [6]).

Для этого выделите ЛКМ пункт *«Функции»* в дереве структуры программы. Аналогично тому, как мы создавали ранее аргументы (см. Настройка отображения состояния насоса ЛР№1 в [6]), создайте функцию с именем *«EM_LEV»*. В дереве структуры программы разверните функцию *«EM_LEV»* и нажмите ЛКМ на пункте *«Аргументы»* функции. Создайте четыре новых аргумента (Разработка программы-эмулятора изменения уровня в емкости ЛР№1): *Flow IN REAL, Pump IN USINT, Valve IN USINT, Level OUT REAL*. Аргументы функции имеют тоже назначение, что и аргументы программы.

Откройте функцию на редактирование, нажав на заголовке функции *«EM_LEV»* ЛКМ в структуре программы, и выберите язык ST в диалоге. Введите текст программы, как показано на рисунке 1.9.

Программа подробно рассматривалась в ЛР №1. В данном случае добавлено уменьшение уровня за счет отбора жидкости, значение которого хранится в аргументе функции *«Flow»*.

Кроме того, последняя пара операторов «*if-then*» заменяет условие *else,* необходимое для ограничения уровня диапазоном 0…100% в ранних условиях «*if-then*» программы в ЛР№1. Вынесение данной проверки отдельно необходимо для окончательной перезаписи результатов вычислений больших 100 значением 100, а отрицательных значений – нулем.

Снова нажмите на заголовок программы в структуре программы. Откройте палитру блоков LD-FBD кнопкой  на панели и перейдите в раздел *«Пользовательские»*. Перенесите в поле *РШП* пользовательский блок *«EM_LEV»* (блок получает имя функции) и разместите, как показано на рисунке 1.8. Ко входам *(Pump)* и *(Valve)* блока «*EM_LEV*» привяжите соответствующие одноименные аргументы программы. На выход *(Level)* привяжите аргумент *«Level»*, в который будет записываться рассчитанное блоком значение уровня в емкости.

Откройте вкладку *«Арифметические Функции»* палитры блоков и перенесите блок *«Умножение» X*Y* в рабочее поле. Ко входу *(X)* данного блока привяжите аргумент *«Flow»*, а на вход *(Y)* подайте константу «*0.03*» (корректирующий коэффициент – при 100% открытия клапана слива убыль жидкости составит 3%). Выход *(O)* блока «*X*Y*» привяжите ко входу *(Flow)* блока «*EM_LEV*».

Откройте вкладку *«Функции сравнения»* и перенесите в рабочее поле блоки *«Больше»* > и *«Меньше»* <. Разместите их, как показано на рисунке 1.8. На входы *(IN1)* обоих блоков подайте с выхода *(Level)* блока «*EM_LEV*». На вход *(IN2)* блока «<» подайте константу «35». На вход *(IN2)* блока «>» подайте константу «85» (по условию). Таким образом, пороговые значения будут сравниваться с вычисленным значением уровня, исходя из чего будут формироваться сигналы сигнализатора уровня *«Min»* и *«Max»,* равные единице. Поэтому на выход *(LT)* блока *«<»* привяжите аргумент *«Min»,* а на выход *(GT)* блока *«>»* – аргумент *«Max»*.

```
FUNCTION_BLOCK EM_LEV
    VAR_INPUT Flow : REAL; END_VAR
    VAR_INPUT Pump : USINT; END_VAR
    VAR_INPUT Valve : USINT; END_VAR
    VAR_OUTPUT Level : REAL; END_VAR

if Level<100 then
    case Pump of
    1: Level=Level+2.5;
    0: Level=Level+0;
    end_case;
    end_if;
if Level>0 then
    case Valve of
    1: Level=Level-5;
    0: Level=Level-0;
    end_case;
    end_if;
Level=Level-Flow;
if Level>100 then Level=100;
    end_if;
    if Level<0 then Level=0;
    end_if;

END_FUNCTION_BLOCK
```

Рис. 1.9. Программа эмуляции уровня на языке ST

Компиляция и отладка. Выполните

компиляцию программы клавишей *F7*. Откройте окно переменных, нажав кнопку *«Переменные»* на панели инструментов отладчика, и запустите программу на циклическое выполнение, нажав клавишу *F5* (см. Компиляция и отладка ЛР№1 в [6]).

Введите в аргумент *«Flow»* значение в диапазоне 0…100 и наблюдайте за формированием состояний сигнализатора уровня *«Min»* и *«Max»*, а также отработкой исполнительных устройств – насоса *«Pump»* и клапана *«Valve»*. Затем откройте сливной клапан, введя *«1»* в аргумент *«Empty»*. Наблюдайте за блокировкой запуска насоса *«Pump»* и состояния сигнализатора *«Min»*.

Создание и настройка базы каналов. Создадим каналы для организации обмена данными между экраном и программой (см. Создание базы каналов, ЛР№1 в [6]). Разверните узел RTM_1 в навигаторе проекта и нажмите ПКМ на канале вызова шаблона экрана. Выберите в контекстном меню пункт *«Свойства»*. В открывшемся окне переключитесь на вкладку *«Аргументы»*. Выделите ЛКМ с нажатой клавишей *CTRL* все каналы, кроме *«Pump_соб»* и *«Valve_соб»*. Далее нажмите ЛКМ на кнопку *«Создать по аргументам каналы с привязкой»* на панели работы с аргументами. Будет создано семь каналов соответствующего аргументам типа.

Теперь необходимо связать созданные каналы с программой. Аналогично нажмите ПКМ на канале вызова шаблона программы и выберите в контекстном меню пункт *«Свойства»*. В открывшемся окне переключитесь на вкладку *«Аргументы»*.

Перетащите каналы из навигатора проекта, удерживая ЛКМ, на строки аргументов с аналогичными именами в табличном редакторе привязок аргументов. Привязки будут созданы автоматически.

Двойным нажатием ЛКМ в навигаторе проекта откройте последовательно свойства каналов *«Max»* и *«Min»*, которые будут заноситься в отчет тревог узла. Переключитесь на вкладку *«Архивирование»* и установите флаг *«Отчет Тревог»* для обоих каналов. Для канала *«Max»* в поле «*Индекс аварийного словаря»* выберите созданный ранее словарь сообщений *«Максимум»*, нажав на *«…»* и выбрав путь к его расположению. Для канала *«Min»* аналогично выберите словарь сообщений *«Минимум»*.

Создадим каналы для фиксации событий. В навигаторе проекта перейдите в группу *«Каналы»* узла *RTM_1* (можно переименовать группу в *«События»*) и нажмите на ней ПКМ. В контекстном меню выберите пункт *«Создать компонент → Событие»*, создав тем самым канал класса Событие. Нажмите ПКМ на канале и, выбрав в контекстном меню пункт *«Переименовать»*, переименуйте

его в *«Насос»*. Точно также создайте еще один канал класса *«Событие»* и переименуйте его в *«Клапан_сброса»*.

Откройте последовательно оба созданных канала *«Событие»* на редактирование двойным нажатием ЛКМ и задайте размер стека аварий равным 10. Таким образом, в канале будет храниться информация о 10 последних изменениях состояния канала – событиях, что можно будет просмотреть в соответствующем графическом элементе.

Для того чтобы каналы класса *«Событие»* могли фиксировать события, их необходимо связать с соответствующими каналами – источниками данных. Это удобно сделать продублировав окно навигатора проекта кнопкой *«Открыть дополнительное окно навигатора проекта»*  . Сделав это, выделите ЛКМ и перетащите с зажатой ЛКМ канал класса *«HEX16»*  на канал класса *«Событие»*  . Повторите процедуру для пары каналов *«Valve»* и *«Клапан_сброса»*.

Сохранение проекта и запуск на исполнение в профайлере. Нажмите ЛКМ кнопку *«Сохранить»*  и затем *«Сохранить для МРВ»*  на главной панели инструментов (см. Сохранение и подготовка проекта к запуску, ЛР№1 в [6]). Откройте профайлер кнопкой *«Запустить профайлер»*  на главной панели инструментов. В профайлере еще раз нажмите ЛКМ кнопку *«Запуск/Останов»*  .

Проверьте работоспособность проекта и системы управления, вводя различные значения количества отбираемой из емкости жидкости (в процентах), и выполняя принудительный сброс в ручном режиме. Наблюдайте за выводом зафиксированных событий и информацией в отчете тревог узла. Итоговый результат разработки, запущенный на исполнение в профайлере, показан на рисунке 1.10.

Для останова профайлера нажмите ЛКМ кнопку *«Запуск/Останов»*  . После этого закройте профайлер.

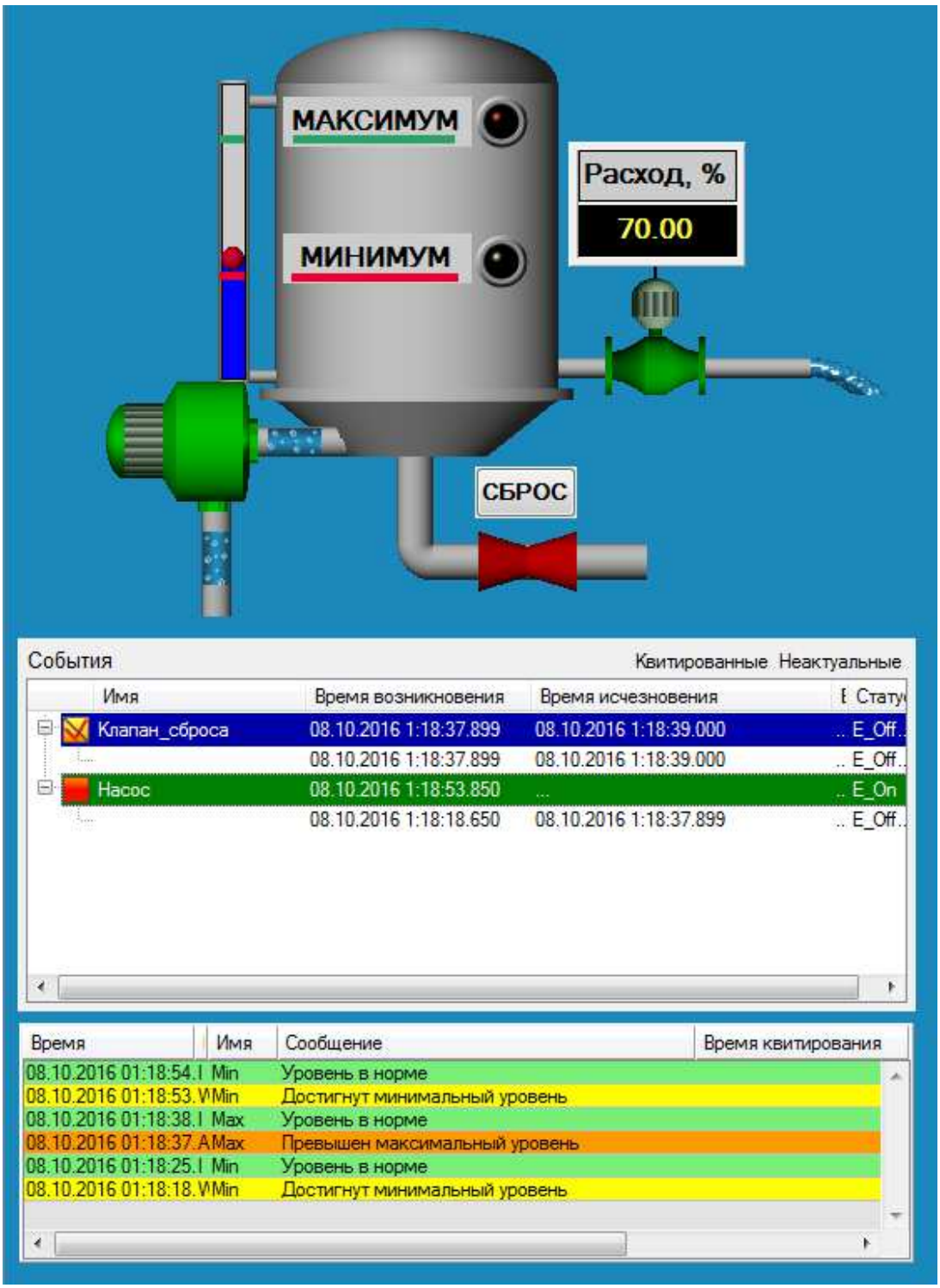


Рис. 1.10. Проект, запущенный на исполнение в профайлере

1.4. Содержание отчета

Отчет должен включать:

1) скриншот экрана проекта, запущенного на исполнение в профайлере, с указанием типов использованных ГЭ и настраиваемых свойств этих ГЭ;

2) разработанный алгоритм управляющей части программы комбинационного управления по ГОСТ 19.701-90 (без эмуляции и имитации);

3) скриншот программы комбинационного управления на языке LD, совмещенной с программой-эмулятором уровня в емкости и имитации работы сигнализатора уровня на языках FBD и ST с текстовым пояснением ее работы и описанием синтаксиса использованных операторов и функциональных блоков;

4) листинг программы-эмулятора уровня в емкости;

5) самостоятельное задание. Программу комбинационного управления на языке ST (замените управляющую часть программы на LD), а затем на FBD;

6) вывод по проделанной работе.

1.5. Контрольные вопросы

2. Дайте определение комбинационного управления.
3. Приведите примеры технических средств автоматизации, которые могут являться источниками данных (датчиками) и исполнительными устройствами при комбинационном управлении.
4. Какие основные логические операции вам известны?
5. Приведите схему, обозначение и таблицу истинности следующих элементов: OR («ИЛИ»), AND («И»), NOR («НЕ ИЛИ»), NAND («НЕ И»), XOR («исключающее ИЛИ»).
6. Для чего служат отчеты тревог? Что такое «квитирование»?
7. Каково назначение ГЭ «События»?
8. Какие шины присутствуют при написании программ на LD?
9. Что такое ассоциированная переменная?
10. Поясните работу таких LD-блоков как разомкнутый контакт | |, замкнутый контакт |/|, катушка ( ), инверсная катушка (/), катушка удержания (S), катушка сброса (R).
11. Каким образом реализовать приоритеты при написании программ на языке LD?

ЛАБОРАТОРНАЯ РАБОТА №2

УПРАВЛЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТЯМИ СОБЫТИЙ (КОМБИНАЦИОННОЕ УПРАВЛЕНИЕ)

Цель работы: освоить принципы реализации управления последовательностями событий с применением языка последовательных функциональных карт Sequential function chart (SFC).

Задача работы: разработать технологический экран (мнемосхему) химического реактора и программу управления последовательностями операций в реакторе на языке SFC, а также программу-эмулятор уровня в емкости на языке ST и программу-эмулятор температуры нагрева на языке FBD.

Программное обеспечение: инструментальная система Trace Mode 6.

2.1. Порядок выполнения работы:

1. Изучите теоретический сведения, необходимые для выполнения лабораторной работы.
2. Выполните практические задания согласно методическим рекомендациям, описывающим содержание и порядок работы.
3. Составьте отчет по проделанной работе, включив в него результаты в соответствии с требованиями к содержанию отчета.
4. Подготовьте ответы на контрольные вопросы.

2.2. Теоретические сведения

Существуют процессы, в которых важно не только соблюдение определенных комбинаций состояний исполнительных устройств, но и смена этих состояний в строго определенной последовательности. Чем сложнее процесс, чем больше в нем операций, тем выше необходимость структурирования последовательностного управления, что достигается с помощью функциональных карт. ***Функциональную карту*** можно рассматривать как специализированную для описания управляющих последовательностей графическую схему, которая в стандарте МЭК 61131-3 представлена языком SFC.

SFC описывает управляющие последовательности с помощью заранее определенных действий, которые необходимо произвести в определенной последовательности ***(шагов)*** и ***правил перехода между этими действиями (условий перехода)***, которые изображаются между шагами. Переход к следую-

щему по очереди шагу (действию) осуществляется только в том случае, если условие перехода истинно (возвращает «TRUE»). В противном случае переход не выполняется, и вся остальная часть действий после невыполненного условия в данной ветви (последовательности) остается также не исполненной.

Действия (шаги) могут быть написаны на любом языке стандарта МЭК 61131: ST, FBD, LD, IL и самом SFC. Условия перехода – на ST, FBD, LD и IL, т.к. запрограммировать функцию на языке SFC нельзя.

Шаг, который выполняется в данном цикле исполнения программы на SFC, называется ***активным***. Соответственно, при последовательном исполнении действий, остальные шаги ***неактивны***. Ветви, для которых это справедливо всегда называются ***простыми последовательностями (simple sequence)*** – в них существует только один переход после любого шага и только один шаг после любого перехода.

Однако существует также возможность создания ***альтернативных параллельных последовательностей (ветвлений)*** и ***синхронных параллельных последовательностей (расщеплений)***.

В альтернативной параллельной последовательности *(alternative parallel sequence)* существуют два или более переходов после одного шага. В этом случае исполнение может пойти по разным ветвям в зависимости от внешних условий. Обычно это условия типа «если-то-иначе», которые полезны при разграничении действий, например, при нормальном развитии процесса и в условиях аварийной ситуации.

В альтернативной параллельной последовательности очень важно удостовериться, что условие выбора одной из ветвей программы непротиворечиво и однозначно; другими словами, альтернативные ветви нельзя запускать одновременно. Каждая ветвь альтернативной параллельной последовательности должна всегда начинаться уникальным условием перехода (чаще, логическим).

В синхронной параллельной последовательности *(simultaneous parallel sequence)* после перехода предусматриваются два или более шагов, которые могут быть активными одновременно, т.е. при выполнении условия перехода обе ветви становятся активными одновременно и выполняются независимо и параллельно. На параллельное исполнение указывают двойные горизонтальные линии.

Следует учесть, что переход к шагу, расположенному после нижней двойной горизонтальной линии, может произойти только после завершения обеих выполнявшихся ветвей. При этом все параллельные шаги должны быть связаны с одним и тем же последующим условием.

Все три типа конструкций можно использовать вместе, но с известной осторожностью, чтобы избежать потенциальных конфликтов. Например, если ветви альтернативной последовательности оканчиваются как параллельно исполняемые (две горизонтальные черты), то дальнейшее развитие будет невозможно, так как контроллер будет ждать завершения работы обеих ветвей, хотя из-за альтернативного условия была запущена только одна из них. Возможна обратная ошибка: если параллельные ветви, которые должны закончиться одновременно, соединены завершаются как альтернативные (одна горизонтальная черта), то множество разных шагов могут остаться активными и дальнейшее развитие процесса может принять неуправляемый характер.

Конечно, компилятор обычно распознает такие несоответствия и выдает предупреждение пользователю до запуска программы. Но даже при использовании лучших компиляторов многие ошибки остаются скрытыми и труднораспознаваемыми. Структурный и методичный подход к программированию всегда является важным требованием.

2.3. Содержание и выполнение работы

Разработаем программу управления химическим реактором, реализующим этапы технологического процесса, описываемые следующим текстовым алгоритмом:

1. При условии поступления команды «Запуск» и отсутствии команды на слив продукта, открыть клапан компонента №1;
2. По достижении уровня в 40%, закрыть клапан компонента №1 и открыть клапан компонента №2;
3. По достижении уровня в 70%, закрыть клапан компонента №2, включить мешалку на 50 секунд, включить нагреватель и поддерживать температуру 75±1°C;
4. По истечении 50 секунд выключить мешалку, выключить нагреватель и открыть клапан выдачи продукта (сливной);
5. При опустошении емкости, закрыть клапан выдачи продукта (сливной). Инкрементировать количество циклов приготовления. Начать новый цикл приготовления.

При отсутствии команды «Запуск» или при поступлении команды «Стоп» при отсутствии команды на слив продукта закрыть клапаны компонентов №1 и №2, выключить мешалку и нагреватель, закрыть клапан выдачи продукта (сливной). При поступлении команды «Запуск» при отсутствии команды на

слив продукта возобновить приготовление с этапа, на котором была выполнена остановка процесса.

При поступлении команды на слив продукта со стороны оператора при наличии активной команды «Запуск» закрыть клапаны компонентов №1 и №2, выключить мешалку и нагреватель, установить таймер перемешивания заново на 50 секунд (100 циклов; связь числа циклов и секунд раскрывается далее по ходу работы), открыть клапан выдачи продукта (сливной). При деактивации команды на слив продукта при наличии активной команды «Запуск», принудительно выполнять слив до опустошения емкости (слив брака) – держать клапан слива открытым; после опустошения емкости закрыть клапан выдачи продукта (сливной) и начать новый цикл приготовления с п.1.

При деактивации команды на слив продукта при отсутствии активной команды «Запуск» закрыть клапан выдачи продукта (сливной). Ожидать дальнейших команд.

Уровень в емкости измеряется в процентах. Максимальная температура нагрева – 100^0C. Начальная температура - 10^0C. Время разогрева до максимальной температуры – около 3 минут.

На экране оператора должно быть представлено схематичное изображение реактора, снабженного цифровыми индикаторами температуры и уровня продукта в емкости, а также обратным счетчиком оставшихся циклов (времени) перемешивания. Также должны быть выполнены индикаторы работы нагревателя и мешалки; состояние клапанов должно отображаться цветовой индикацией. Для управления процессом – запуска и останова – предусмотреть соответствующие кнопки и индикатор состояния «запуск/останов». Для ручного слива продукта предусмотреть двухпозиционную кнопку управления сливным клапаном. Количество полных циклов приготовления должно отображаться соответствующим цифровым счетчиком.

Создание проекта, узла, шаблона экрана. Откройте инструментальную систему Trace Mode 6 и создайте новый проект, нажав кнопку *«Создать новый проект»*  на панели инструментов *«Главная»*. Сохраните проект, нажав кнопку *«Сохранить текущий проект»*  на панели инструментов *«Главная»*. Выберите местоположение проекта и задайте его имя (см. Создание и сохранение проекта, ЛР№1 в [6]).

Создайте узел проекта типа RTM, нажав ПКМ на слое *«Система»* в навигаторе проекта и в контекстном меню выбрав *«Создать узел → RTM»*. (см. Создание узла проекта, ЛР№1). Создайте шаблон экрана, аналогично нажав ПКМ

на созданном узле RTM_1 и выбрав в контекстном меню *«Создать компонент → Экран»* (см. Создание шаблона экрана оператора, ЛР№1 в [6]).

Добавление ресурса «Графические элементы». ГО «Емкость». Если это еще не было сделано ранее, добавьте библиотеку компонентов tmdevenv.tmul (см. Добавление ресурса «Графические элементы», ЛР№2 в [6]).

Затем создайте в проекте группу *«Графические_элементы»*. Для этого переключитесь в слой *«Ресурсы»* в навигаторе проекта, нажмите ПКМ на нем и выберите в контекстном меню *«Создать группу → Графические_элементы»*. Последовательно разверните пункты слоя *«Библиотека компонентов → Пользовательская → Library_1 → Object_1 → Resources → Tanks»*. Выберите шаблон графического объекта (ГО) *«Tank_1»* и перетащите его, зажав ЛКМ, в созданную группу «*Графические_элементы»*. Теперь данный объект добавлен в ваш проект.

Откройте шаблон экрана на редактирование в *РПД* и перетащите с нажатой ЛКМ ГО *«Графический_объект_1»* из группы *«Графические_элементы»* в навигаторе проекта на экран, не отпуская ее. Измените размер и, при необходимости, положение *ГО «Емкость»* (элемент 1), как показано на рисунке 2.1.

Трубы. ГЭ «Труба». Далее покажем трубы, по которым осуществляется наполнение (элемент 2, элемент 4) и опорожнение емкости (элемент 6).

Из группы *«Объемные фигуры»* выберем *ГЭ «Труба»*  . Разместите соответствующим образом узлы труб (см. Создание ГЭ «Труба», ЛР№1 в [6]) так как показано на рисунке 2.1.

Клапаны. ГЭ «Клапан». Разместим на трубах налива (элементы 2 и 4) и сливной трубе (элемент 6) запорные (электромагнитные) клапаны клапан компонента №2, компонента №1 и клапан выдачи продукта (сливной) (элементы 3, 5 и 6 соответственно). Для этого из группы *«Объемные фигуры»* выберем *ГЭ «Клапан»*  (см. Создание ГЭ «Клапан», ЛР№1 в [6]) и расположим, как на рисунке 2.1.

Настройка отображения состояния клапанов. Для каждого клапана (элементы 3, 5 и 6) в пункте свойств *«Базовый цвет → Вид индикации»* выберите из списка пункт *«Arg=const»* (см. Настройка отображения состояния клапана, ЛР№1 в [6]) и в свойстве *«Привязка»* выполните привязку к вновь созданным аргументам *«Клапан2», «Клапан1», «Слив»* с типом *«IN»* и типом данных *«USINT»*.

В пункте свойств *«Если ЛОЖНО»* вместо красного цвета выберите цвет, соответствующий закрытому клапану – серый. В поле свойства *«Константа»* введите *«1»*.

ГО «Электродвигатель». Отобразим электропривод мешалки. Аналогично тому, как выше был добавлен *ГО «Емкость»*, добавьте в группу *«Графические_элементы» ГО «Motor_1» (Электродвигатель)* из слоя *«Библиотека компонентов → Пользовательская → Library_1 → Object_1 → Resources → Motors»*. Разместите данный ГО на экране в соответствии с расположением элемента 8 на рисунке 2.1.

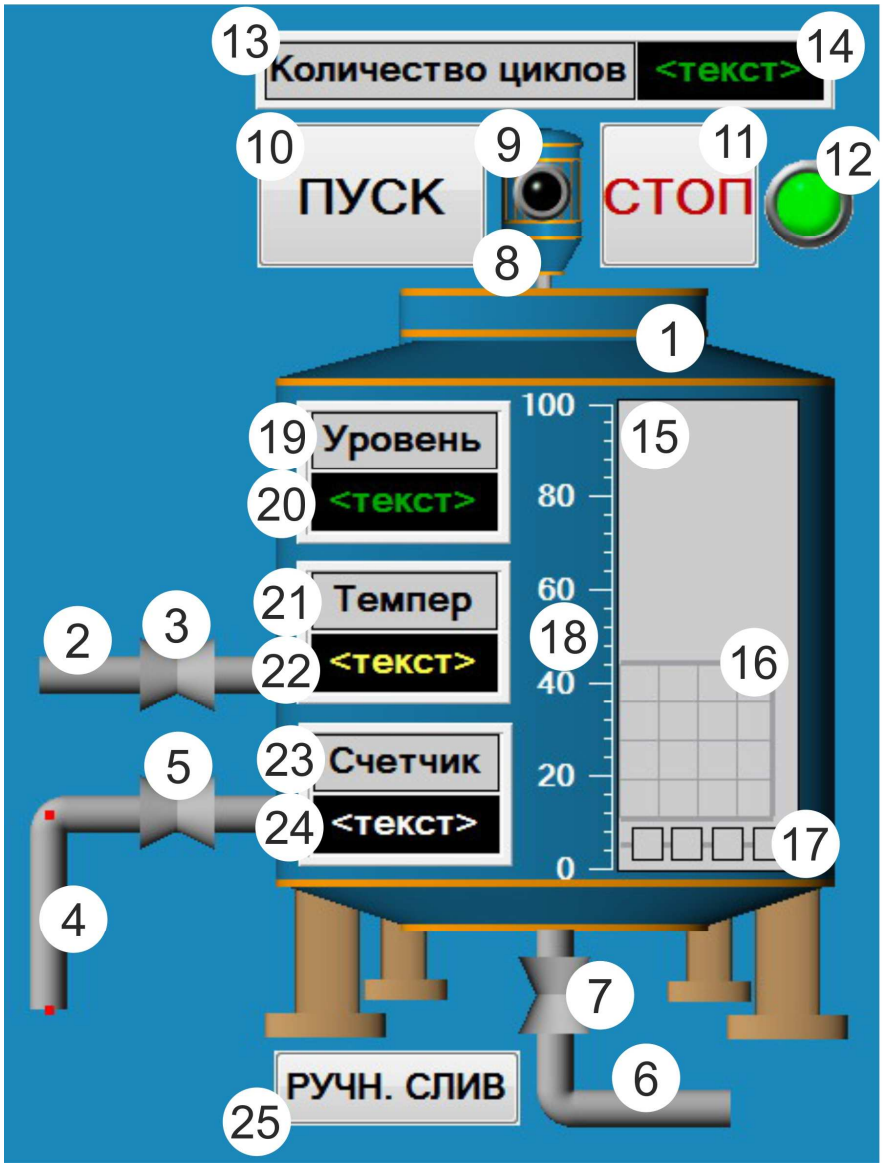


Рис. 2.1. Мнемосхема управления химическим реактором

Создание анимации работы электропривода мешалки. Анимация будет представлена зеленой сигнальной лампой. Нажмите ПКМ на слое *«Ресурсы»* и в контекстном меню выберите *«Создать группу → Анимация»*. Затем также нажмите ПКМ на группе *«Анимация»* и выберите в контекстном меню *«Создать компонент → Библиотека видеоклипов»* (см. Добавление анимационного клипа в проект и размещение на экране. Видеоклип, ЛР №2 в [6]).

Двойным нажатием ЛКМ откройте библиотеку видеоклипов на редактирование. Затем нажмите ПКМ в рабочем поле библиотеки и в контекстном меню выберите пункт *«Импортировать»* . Укажите путь к папке *«C:\Program Files (x86)\AdAstra Research Group\Trace Mode IDE 6 Base\Lib\Animation\Lamps\»* (на примере Windows 7) и затем выберите файл *«lamp_green_top»*. Переключитесь на вкладку *РПД* для разрабатываемого шаблона экрана. Откройте группу *«Ресурсы»* и выберите ресурсную библиотеку видеоклипов . Нажмите ЛКМ на видеоклипе в библиотеке и, удерживая ее, перетащите анимацию на экран, разместив как показано на рисунке 2.1 (элемент 9).

В свойствах видеоклипа *«Зеленая лампа»* выполните привязку (свойство *«Привязка»*) к созданному аргументу *«Миксер»* с типом «*IN*» и типом данных «*USINT*».

Кнопки управления процессом. ГЭ «Кнопка». Разместим кнопки для выдачи команд «Пуск» и «Стоп» (См. Создание кнопок управления исполнительными устройствами. ГЭ «Кнопка» ЛР №1 в [6]).

Выберите инструмент *«Кнопка»* . Разместите кнопку под емкостью (элемент 10), как показано на рисунке 2.1. В пустом поле свойства *«Текст»* на вкладке *«Общие свойства»* введите надпись «ПУСК» и нажмите *Enter*. В свойстве «*Шрифт*» установите жирное начертание, 20 кегль, выбрав их в списке.

Переключитесь на вкладку *«События»* . Нажмите ПКМ на свойстве *MousePress*. В контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«1»*. В свойстве *«Результат»* нажмите на «…» и установите привязку с вновь созданным аргументом *«Запуск»*.

Скопируйте кнопку и поместите справа (элемент 11). Измените значение, передаваемое в свойство *«Значение»* на вкладке *«События»* с *«1»* на *«0»*, а также измените значение в свойстве *«Текст»* на вкладке *«Общие свойства»* на «СТОП». Также измените цвет текста данной кнопки с черного на красный. Для этого выберите красный цвет в списке свойства *«Цвет текста»*.

Создание индикатора-выключателя. На панели ГЭ раскройте группу *«Выключатели»* и выберите ГЭ *«Выключатель 6»* . Расположите его справа от кнопки «Стоп» (элемент 12). В свойстве *«Привязка»* выберите созданный ранее аргумент *«Запуск»*. В свойстве *«Вид индикации»* выберите из списка *«Arg=Const»*. В поле свойства *«Константа»* введите значение *«0»*.

Создание счетчика числа циклов приготовления. ГЭ «Текст». Разместим над кнопками управления счетчик. Надпись «Количество циклов» выпол-

ним в виде статического текста, а сам цифровой индикатор – в виде динамического текста (см. Создание статических надписей. ГЭ «Текст» и Создание динамических надписей. ГЭ «Текст», ЛР №1 в [6]).

Выберите на панели ГЭ инструмент *«Текст»* . Разместите текст в позиции элемента 13. В свойствах *ГЭ «Текст»* в пустом поле свойства *«Текст»* на вкладке *«Общие свойства»* введите надпись «Количество циклов» и нажмите *Enter*. В параметрах свойства «*Шрифт*» установите жирное начертание, 14 кегль. Нажав на раскрывающийся список свойства *«Цвет текста»* измените цвет текста на черный.

Точно также выберите на панели ГЭ инструмент *«Текст»* . Разместите текст на емкости в позиции элемента 15, как показано на рисунке 2.1. Нажатием ЛКМ на параметрах шрифта в свойстве «*Шрифт*» установите жирное начертание, 14 кегль, выбрав их в списке. Измените цвет текста – выберите зеленый, нажав на раскрывающийся список свойства *«Цвет текста»*.

Настроим отображение численного значения данным ГЭ. Раскройте пункт свойств *«Текст → Вид индикации»* на вкладке *«Общие свойства»* и выберите из списка вместо *«Нет динамизации»* пункт *«Значение»*. В свойстве *«Привязка»* нажмите на «…» и выполните привязку к вновь созданному аргументу *«Цикл»* с типом *«IN»* и типом данных *«USINT»*.

Изменим формат отображения численного значения. Здесь же в пункте *«Формат»* свойства *«Текст»* выберите из раскрывающегося списка *«Integer»* (целочисленный).

Создание цифровых индикаторов параметров. ГЭ «Текст». Аналогично счетчику, создадим индикаторы температуры, уровня, и таймер времени (циклов) перемешивания.

Точно также, как и в предыдущем пункте, сделаем это с помощью ГЭ *«Текст»* . Разместите статический текст (элементы 19, 21, 23). В поле свойства *«Текст»* введите надписи, как показано на рисунке 2.1. В параметрах свойства «*Шрифт*» установите жирное начертание, 14 кегль. Нажав на раскрывающийся список свойства *«Цвет текста»* измените цвет текста на черный.

Создадим цифровые индикаторы, как динамический текст с помощью ГЭ *«Текст»* . Разместите текст в позициях элементов 20, 22, 24). Разверните пункт свойств *«Текст → Вид индикации»* на вкладке *«Общие свойства»* каждого элемента и выберите из списка вместо *«Нет динамизации»* пункт *«Значение»*. В свойстве *«Привязка»* выполните привязку к вновь созданным аргументам *«Уровень»* (элемент 20), *«Температура»* (элемент 22) и *«Таймер»* (элемент 24) с типом *«IN»* и типом данных *«REAL»*.

Для всех трех *ГЭ «Текст»* в пункте *«Формат»* свойства *«Текст»* выберите из раскрывающегося списка *«Float»* (с плавающей точкой) вместо *«Generic»* (общий, текстовый). Уменьшите количество знаков дробной части с трех до одного, исправив *«%.3f»* на *«%.1f»*.

Создание смотрового окна. ГЭ «Прямоугольник». Разместим на емкости «смотровое окно» (см. Создание смотрового окна. ГЭ «Прямоугольник», ЛР №1 в [6]) с помощью *ГЭ «Прямоугольник»*  , выбрав его из группы *«Прямоугольники»*  . Разместите *ГЭ «Прямоугольник»* так, как показано на рисунке 2.1 (элемент 15).

Выполним динамическую заливку «смотрового окна», соответствующую уровню жидкости в реакторе. На вкладке *«Динамическая заливка»*  раскройте пункт свойств *«Слои → Слой»* и в подпункте *«Привязка»*, нажав на *«…»*, выберите созданный ранее аргумент *«Уровень»*.

Создание шкалы. ГЭ «Ползунок». Разместим шкалу рядом со «смотровым окном» (см. Создание шкалы. ГЭ «Ползунок», ЛР №1 в [6]). Выберите *ГЭ «Ползунок»*  из группы *«Приборы»*  . Разместите *ГЭ «Ползунок»* рядом с *ГЭ «Прямоугольник»*, как показано на рисунке 2.1 (элемент 18).

Отредактируйте свойства ГЭ *«Ползунок»* следующим образом: в качестве значения свойства *«Ползунок»* выберите *«False»*, свойства *«Полоса»* – *«False»*. Раскройте свойство *«Рамка»* и в пункте *«Тип»* выберите значение *«Стандартная»*. В раскрывающемся списке пункта *«Стиль»* данного свойства выберите пустую строку. Раскройте свойство *«Фон»* и в раскрывающемся списке пункта *«Стиль»* данного свойства выберите *«Без заливки»*. Раскройте свойство *«Шкала»*. Нажав на раскрывающийся список пункта *«Цвет текста»,* выберите белый.

Раскройте пункт *«Уровень 2»* свойства *«Шкала»* и выберите значение *«False»* в подпункте *«Использовать»*. В подпункт *«Десятичные знаки»* введите значение «0». Измените размеры ползунка, чтобы сопоставить его размеры с размером «окна».

Индикация работы мешалки. ГЭ «Линия». Помимо индикации работы электропривода мешалки, введем анимацию «вращения» мешалки. Для этого воспользуемся *ГЭ «Линия»*  (см. Создание ГЭ «Линия», ЛР№1). Нарисуйте крайнюю правую границу мешалки (элемент 16).

Перейдите на вкладку *«Динамический контур»*  . В свойстве *«Привязка»* выберите созданный ранее аргумент *«Миксер»*. Цвет штриха установите зеленым, а цвет промежутка – серым. Установите значение свойства *«Длина штри-*

ха» равным *«5»*. Через команду меню *«Сервис → Тиражировать»* выполните тиражирование вертикальных линий мешалки. Аналогично нарисуйте горизонтальные линии.

Для крайних линий, формирующих контур мешалки, на вкладке *«Общие свойства»* введите в подпункт *«Толщина»* свойства *«Контур»* значение *«3»*.

Теперь при «вращении» мешалки контур будет анимироваться бегущим огнем.

Индикация работы нагревателя. ГЭ «Прямоугольник». Нагреватель схематично покажем группой прямоугольников. Соответственно, работу нагревателя будем индицировать изменением цвета прямоугольников.

Разместите прямоугольник (элемент 17), воспользовавшись *ГЭ «Прямоугольник»* из группы *«Прямоугольники»*, в пределах смотрового окна. Для облегчения позиционирования используйте кнопки панели *«Топология экрана»*. В случае, если один из элементов закрывает другой, используйте кнопки *«Переместить вниз»* и *«Переместить вверх»* для изменения порядка элементов.

В пункте *«Цвет заливки»* свойства *«Заливка»* в подпункте *«Вид индикации»* выберите из списка *«Arg=Const»*. Здесь же выполните привязку (свойство *«Привязка»*) к вновь созданному аргументу *«ТЭН»* с типом *«IN»* и типом данных *«USINT»*. В поле *«Константа»* введите значение *«1»*.

Теперь включенный нагреватель будет индицироваться зеленым цветом, а выключенный – красным. Скопируйте или растиражируйте прямоугольник, как показано на рисунке 2.1.

Через цепь прямоугольников проведите прямую линию, используя *ГЭ «Линия»*. На вкладке *«Общие свойства»* данного ГЭ введите в подпункт *«Толщина»* свойства *«Контур»* значение *«3»*.

Двухпозиционная кнопка управления сливным клапаном. ГЭ «Кнопка». Разместим кнопку для выдачи команды принудительного слива продукта (см. Создание кнопки управления клапаном сброса. ГЭ «Кнопка», ЛР №2).

Выберите инструмент *«Кнопка»*. Разместите кнопку (элемент 25), как показано на рисунке 2.1. В пустом поле свойства *«Текст»* на вкладке *«Общие свойства»* введите надпись *«РУЧН. СЛИВ»* и нажмите *Enter*. В свойстве *«Шрифт»* установите жирное начертание, 14 кегль, выбрав их в списке. В свойстве *«Два состояния»* выберите значение *«TRUE»*. В свойстве *«Привязка»* выберите вновь созданный аргумент *«РучнСлив»* с типом *«IN»* и типом данных *«USINT»*.

Переключитесь на вкладку *«События»*  . Нажмите ПКМ на свойстве *MousePress*. В контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«1»* (команда открытия клапана). В свойстве *«Результат»* нажмите на «…» и установите привязку с созданным ранее аргументом *«РучнСлив»*.

Аналогично настроим противоположное действие – закрытие клапана слива – и свяжем его с отпусканием левой кнопки мыши на двухпозиционной кнопке. Нажмите ПКМ на свойстве *MouseReleas* и в контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«0»* (команда закрытия клапана). В свойстве *«Результат»* выберите в редакторе аргументов созданный ранее аргумент *«РучнСлив»*.

Так как принудительный слив должен обнулять начальные условия, настроим сброс таймера оставшегося времени перемешивания (числа циклов). Под сбросом подразумевается установка начального числа циклов – 100. Для этого нажмите ПКМ на свойстве *MousePress*. В контекстном меню выберите пункт *«Передать значение»*. Оставьте в свойстве *«Тип передачи»* значение *«Прямая»*. В пустое поле свойства *«Значение»* введите *«100»* (число циклов). В свойстве *«Результат»* нажмите на «…» и установите привязку с созданным ранее аргументом *«Таймер»*.

Настройка времени пересчета узла. Выше уже упоминалось, что 100 циклов пересчета соответствуют 50 секундам. Это следует из свойства, называемого *«Пересчет узла»*.

Нажмите ПКМ на узле *«RTM_1»* в навигаторе проекта и выберите в контекстном меню пункт *«Редактировать»*. Откроется бланк редактирования свойств узла. Обратите внимание на группу *«Пересчет»*. Произведение чисел в полях *«Период»* и *«Разрешение»* определяет время пересчета узла. По умолчанию оно равно 550мс (0,55 с). Исправьте *«0,55»* в поле «Разрешение» на *«0,5»*. В результате за секунду узел будет пересчитываться дважды, т.е. 100 циклов пересчета равны 50 секундам. Время перечета – важный параметр, т.к. SCADA – это системы реального времени.

Разработка программы управления химическим реактором. Программу напишем на графическом языке SFC. Язык Техно SFC позволяет создавать программы в виде алгоритма, состоящего из шагов и переходов. Для шагов задаются выполняемые действия, для переходов – условия переходов между ша-

гами. SFC-программа может выступать в роли основной программы и функции-блока. Запрограммировать функцию на языке Техно SFC нельзя.

Нажмите ПКМ на созданном узле RTM_1 и выберите в контекстном меню *«Создать компонент → Программа»*. Двойным нажатием ЛКМ на канале вызова программы, откройте ее на редактирование в *РШП*.

Нажмите ЛКМ на пункте *«Аргументы» РШП* и создайте в табличном редакторе аргументов программы одиннадцать аргументов со следующими параметрами: *Клапан_1 IN/OUT USINT, Клапан_2 IN/OUT USINT, Уровень IN/OUT REAL, ТЭН IN/OUT USINT, Температура IN/OUT REAL*, *Миксер IN/OUT USINT, Слив IN/OUT USINT, Таймер IN/OUT REAL, Цикл IN/OUT USINT, Запуск IN USINT, РучнСлив IN USINT.*

Нажмите ЛКМ на заголовке программы. В появившемся окне выбора языка программирования выберите графический язык *SFC*. Нажмите в окне выбора языка кнопку *«Принять»*.

Выделите начальный шаг в дереве диаграммы и нажмите кнопку *«Создать шаг/переход»*  либо клавишу *«Insert»* на клавиатуре. Появится новый шаг *«Шаг 1»* и условие перехода между этими шагами – *«Переход 0»,* связанное с линией перехода, соединяющей шаги. Таким образом, при добавлении шага, условие перехода добавляется автоматически.

Для удаления шага достаточно выделить его и нажать кнопку *«Удалить элемент SFC-диаграммы»*  либо клавишу *«DEL»*. Удалить можно только последний созданный шаг в ветви, т.е. тот, который не имеет последующих связей (либо параллельный шаг). При этом удаляется и условие перехода к этому шагу (не действует для параллельных шагов).

Единственное условие перехода между двумя шагами удалить, не удаляя последующий шаг, нельзя. Однако можно создать новое (дополнительное) условие перехода между шагами, выделив один из шагов (первый), и с нажатой ЛКМ переместив указатель ко второму шагу. При этом одно из условий перехода для данной связи может быть удалено.

Каждый шаг соответствует определенному состоянию (этапу) процесса, поэтому в шагах задается состояние исполнительных устройств, которое к этому состоянию приводит.

Переход к первому шагу должен выполняться при условии поступления команды запуск, поэтому в элементе *«Переход 0»* необходимо это условие проверить. Сначала переименуем условие перехода *«Переход 0»*. Для этого дважды нажмите ЛКМ на этом переходе и введите *«Запуск»,* нажав после *«Enter»*.

Затем откройте этого откройте переход *«Запуск»* на редактирование, выделив его однократным нажатием ЛКМ и нажав кнопку *«Редактировать элемент SFC-диаграммы»* .

Также элемент можно открыть на редактирование, раскрыв пункт *«SFC-диаграмма»* в дереве программы и далее выбрав пункт *«Шаги»* или *«Переходы»*, и выделив нужный шаг или переход. Он будет открыт на редактирование. Также этот пункт удобен для быстрого переключения между элементами SFC-диаграммы.

Далее выберите язык, на котором будет написан переход. Выберите из диалога *«FBD диаграмма»* и нажмите в диалоге выбора кнопку *«Принять»*.

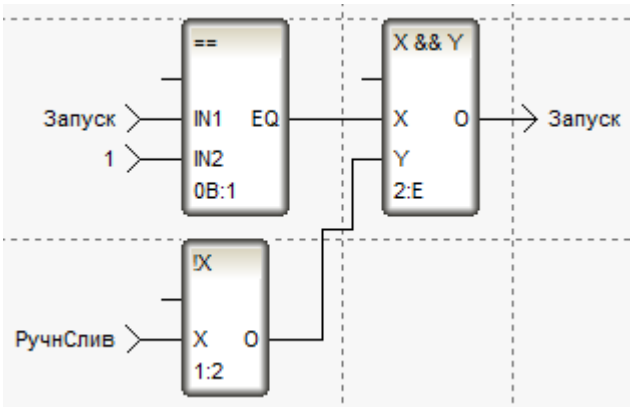


Рис. 2.2. Переход 0 на языке FBD

Ветвь диаграммы, которая начинается с шага 1 соответствует нормальному циклу протекания процесса, что возможно при условии поступления команды *«Запуск» (Запуск=1)* и отсутствии команды на слив от оператора *(РучнСлив=0)*. Одним из вариантов проверки сочетания данных условий является использование блока *«Равенство»* == для проверки равенства единице значения аргумента *«Запуск»* (рис. 2.2) и блока *«Инверсия»* *!X* для недопущения перехода при аргументе *РучнСлив=1*.

Поскольку переход к следующему шагу происходит только, если условие перехода истинно (*TRUE*, равно единице), то необходимо чтобы оба блока возвращали единицу и тогда проверку сочетания условий удобно выполнить с помощью блока логического умножения *«И»* *X&&Y*. Таким образом, при *Запуск=1* на выходе блока *«==»* будет сформирована единица (иначе – 0). Единица на выходе блока *«!X»* будет только при *РучнСлив=0*, что и требуется в данном случае, а переход к *шагу 1* будет выполнен только тогда, когда оба этих условия будут истинны (равны единице), иначе вся ветвь будет игнорироваться.

Обратите внимание, что в качестве привязываемых ко входам аргументов (рис. 2.2) выступают аргументы программы, а на выход последнего исполняемого блока, возвращающего результат обработки условия, привязывается переменная *«Запуск»*, соответствующая названию условия перехода, создавать которую не требуется.

В *РШП* нажмите на панели кнопку «*Показать/скрыть палитру FBD блоков»* . Откроется библиотека функциональных блоков. Перетащите соответ-

ствующие блоки в *РШП* и соедините как показано на рис. 2.2. Блоки *«Инверсия»* *!X* и *«И»* *X&&Y* находятся в разделе *«Логические»,* а *«Равенство»* == в разделе *«Сравнение»*.

Нажмите на заголовок программы *«Программа#1»*, чтобы увидеть SFC-диаграмму.

Откройте на редактирование *«Шаг 1»,* выделив его однократным нажатием ЛКМ и нажав кнопку *«Редактировать элемент SFC-диаграммы»*  . Далее выберите *«LD-диаграмма»* из диалога и нажмите кнопку *«Принять»*.

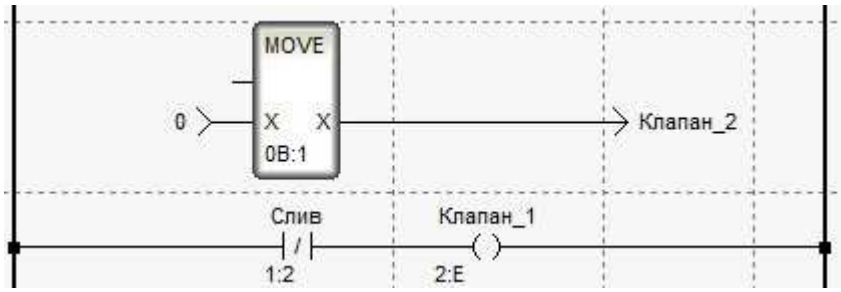


Рис. 2.3. Шаг 1 на языке LD

На данном шаге, соответствующем первому этапу процесса по заданию, необходимо открыть клапан компонента №1; клапан компонента №2 при этом следует держать закрытым. Для того чтобы записать *«0»* в аргумент *«Клапан_2»* воспользуемся блоком пересылки (присвоения) значения *«MOVE»* из раздела *«Ввод/вывод. Переходы»* (рис. 2.3), т.е. в результате *Клапан_2=0.*

Чтобы открыть клапан компонента №1 достаточно просто соединить катушку, связанную с переменной *«Клапан_1»*, с левой основной шиной, чье значение всегда равно логической единице, тем самым *Клапан_1=1*. Однако для того чтобы в дальнейшем не происходило открытие этого клапана при отпуске (сливе) продукта из реактора введем технологическую блокировку – открытие клапана №1 возможно только при закрытом клапане слива *(Слив=0)*, т.е. только на этапе наполнения емкости.

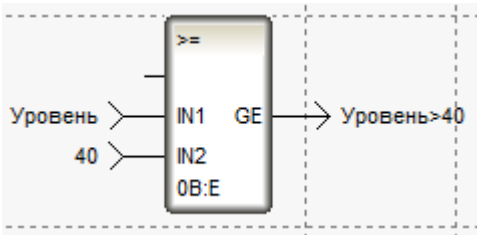


Рис. 2.4. Переход 1 на языке FBD

Для реализации этого соединим катушку «*Клапан_1» ( )* с замкнутым контактом *«Слив»* |/| (рис. 2.3), что обеспечивает следующее: если *Слив*=0, то *out=1* и *Клапан_1=1*; если *Слив*≠0, то *out=0* и *Клапан_1=0,* т.к. *in≠0* всегда (шина всегда имеет состояние TRUE). Контакт и катушка располагаются в разделах *«Контакты»* и *«Катушки»* палитры LD-блоков соответственно.

Выделите на SFC-диаграмме *«Шаг 1»* и нажмите кнопку *«Создать шаг/переход»*  .

Переход к следующему этапу процесса должен происходить при достижении уровнем в емкости значения *40%*, т.е. при условии «*Уровень≥40*». Переименуйте «*Переход 1*» в «*Уровень>40*».

```
SFC_STEP "ШАГ 2"
    VAR_INOUT Клапан_1 : USINT; END_VAR
    VAR_INOUT Клапан_2 : USINT; END_VAR
    VAR_INOUT Уровень : REAL; END_VAR
    VAR_INOUT ТЭН : USINT; END_VAR
    VAR_INOUT Температура : REAL; END_VAR
    VAR_INOUT Миксер : USINT; END_VAR
    VAR_INOUT Слив : USINT; END_VAR
    VAR_INOUT Таймер : REAL; END_VAR
    VAR_INOUT Цикл : USINT; END_VAR
    VAR_INPUT Запуск : USINT; END_VAR
    VAR_INPUT РучнСлив : USINT; END_VAR

    Клапан_1=0;
    Клапан_2=1;
    if Слив==1 then Клапан_2=0;
    end_if;

END_SFC_STEP
```

Рис. 2.5. Шаг 2 на языке ST

Откройте на редактирование условие перехода «*Уровень>40*». Выберите «*FBD-диаграмма*» из диалога и нажмите кнопку «*Принять*». Из раздела «*Сравнение*» палитры блоков перенесите в *РШП* блок «*Больше или равно*» ≥. Привяжите ко входу *(IN1)* аргумент «*Уровень*», ко входу *(IN2)* – константу «*40*». К выходу привяжите переменную «*Уровень>40*», соответствующую условию перехода (рис. 2.4). Таким образом, при *Уровень≥40* переменная «*Уровень>40*»=*TRUE* и будет выполнен переход к шагу 2.

Шаг напишем на языке ST. Выберите соответствующий язык, открыв «*Шаг 2*» на редактирование.

На данном шаге требуется закрыть клапан компонента №1 *(Клапан_1=0)* и отрыть клапан компонента №2 *(Клапан_2=1)*. Точно также, как и на первом шаге, потребуется ввести блокировку – при открытом клапане слива *(Слив=1)* клапан подачи второго компонента должен быть закрыт *(Клапан_2=0)*, что реализуется с помощью оператора условия «*if-then*» и оператора сравнения «*Равенство*» ==. Соответствующий данным операция листинг приведен на рисунке 2.5.

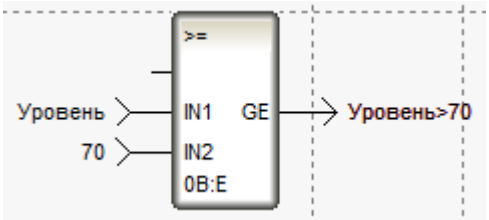


Рис. 2.6. Переход 2 на языке FBD

Выделите на SFC-диаграмме «*Шаг 2*» и нажмите кнопку «*Создать шаг/переход*» .

Выполним редактирование «*Переход 2*» и напишем его на языке FBD. Переименуйте его в «*Уровень>70*». Соответственно, данное условие перехода должно делать активным «*Шаг 3*» по достижении уровнем жидкости значения *70%*.

Все действия аналогичны редактированию «*Уровень>40*» (рис 2.6).

Для написания «*Шаг 3*» выберем язык *FBD*. На данном шаге в соответствии с условием необходимо закрыть клапан компонента №2, включить нагреватель и поддерживать температуру 75±1^{0}C, включить мешалку на 50 секунд.

Для закрытия клапана используем уже примененную ранее операцию присвоения значения с помощью блока *«MOVE»* (рис. 2.7) из раздела *«Ввод/вывод. Переходы»*. В результате *Клапан_2=0.*

Регулирование температуры и управление нагревателем будем выполнять с помощью рассмотренного в лабораторной работе №2 двухпозиционного регулятора – блока *«Гистерезис» HSTR* (см. Разработка программы эмулятора-регулятора температуры агента в емкости, ЛР №2 в [6]). На вход *(INP)* подадим значение температуры (аргумент *«Температура»*), полученное от программы-эмулятора температуры (см. далее); на вход *(PV)* – константу *«75»*, соответствующую уставке по условию; на вход *(DLT)* – константу *«1»,* допустимое отклонение (гистерезис) по условию (рис. 2.7). Выход *(Q)* блока инвертируем, чтобы получить характеристику обратного регулятора, применяемого для управления процессами нагрева, и свяжем с аргументом *«ТЭН»*, используемым для управления одноименным исполнительным устройством.

Для выполнения обратного отсчета циклов ожидания будем выполнять вычитание на каждом цикле пересчета единицы из значения аргумента *«Таймер»* с помощью блока *«X-Y»* из раздела *«Арифметические»* (рис. 2.7). Обновленное значение будет снова записываться в аргумент *«Таймер»* для индикации на экране оператора. Пока значение результата вычитания будет больше нуля, т.е. число циклов ожидания не истекло, значение аргумента *«Миксер»* будет равно единице и мешалка будет работать. Как только число оставшихся циклов сравняется с нулем *Таймер=0* или *Таймер<0*, мешалка выключится *–Миксер=0.* Проверка условия выполняется с помощью блока *«Больше»* > из радела *«Сравнение»*.

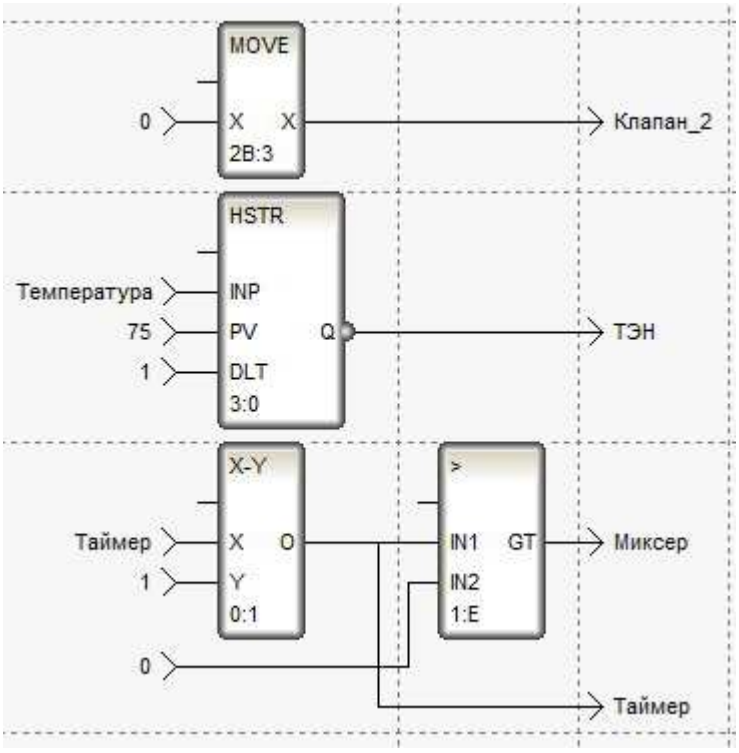


Рис. 2.7. Шаг 3 на языке FBD

Выделите на SFC-диаграмме *«Шаг 3»* и нажмите кнопку *«Создать шаг/переход»* .

Поскольку переход к следующему шагу *«Шаг 4»* выполняется после истечения циклов (времени) работы мешалки, т.е. когда *Миксер=0*, то именно это условие и можно проверить в условии *«Переход 3»*. Для этого используем уже рассмотренный ранее блок *«Равенство»* ==, сравнив с нулем значение аргумента *«Миксер»* (рис. 2.8). При выполнении условия (остановке миксера) выходная переменная примет значение *«TRUE»* и будет выполнен переход к четвертому шагу.

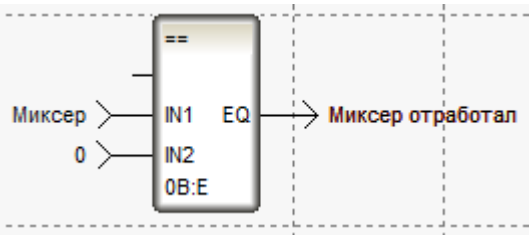


Рис. 2.8. Переход 3 на FBD

Переименуйте *«Переход 3»* в *«Миксер отработал»* и привяжите данную переменную к выходу блока «==».

Откройте на редактирование *«Шаг 4»* и выберите в диалоге язык *«FBD-диаграмма»*. На данном шаге необходимо отключить нагреватель (*ТЭН=0*), открыть клапан выдачи продукта (*Слив=1*) и инкрементировать счетчик циклов приготовления продукции (*Цикл=Цикл+1*). Для записи (присвоения) в аргументы соответствующих значений используем блоки пересылки значений *«MOVE»* из раздела *«Ввод/вывод. Переходы»*, с помощью которых запишем *«0»* в аргумент *«ТЭН»* и *«1»* в аргумент *«Слив»* (рис. 2.9).

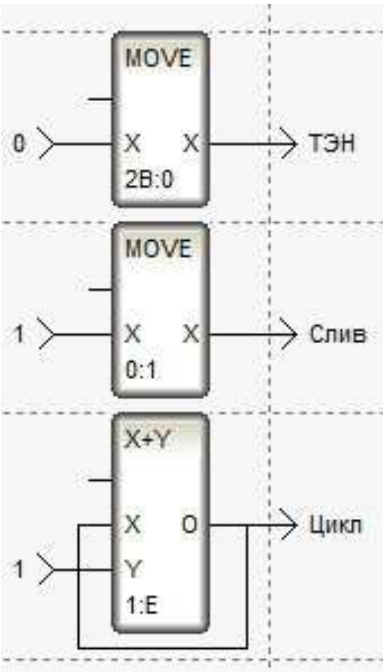


Рис. 2.9. Шаг 4 на языке FBD

Накопление циклов обеспечим использованием блока *арифметического сложения «X+Y»* из раздела *«Арифметические»,* заведя его выход, связанный с аргументом *«Цикл»,* на один из входов блока, а на второй подав константу *«1»* для обеспечения инкремента (рис 2.9).

Таким образом, все этапы нормального протекания технологического процесса отражены в управляющей программе. Осталось обеспечить закрытие клапана выдачи продукта *(Слив=0)* при опустошении емкости *(Уровень≤0)* и установку таймера отсчета числа циклов (времени) работы миксера *(Таймер=100)* в начале нового цикла приготовления продукта (новый цикл начнется только после обнуления таймера – *Таймер≤0)*. Это удобно отразить в начальных условиях.

Откройте на редактирование *«Начальный шаг»*. Выберите разработку на языке ST. Проверку условий и запись соответствующих значений в аргументы выполним с помощью оператора условия *«if-then»* (рис. 2.10).

```
SFC_STEP "Начальный шаг"
    VAR_INOUT Клапан_1 : USINT; END_VAR
    VAR_INOUT Клапан_2 : USINT; END_VAR
    VAR_INOUT Уровень : REAL; END_VAR
    VAR_INOUT ТЭН : USINT; END_VAR
    VAR_INOUT Температура : REAL; END_VAR
    VAR_INOUT Миксер : USINT; END_VAR
    VAR_INOUT Слив : USINT; END_VAR
    VAR_INOUT Таймер : REAL; END_VAR
    VAR_INOUT Цикл : USINT; END_VAR
    VAR_INPUT Запуск : USINT; END_VAR
    VAR_INPUT РучнСлив : USINT; END_VAR

    if Уровень<=0 then Слив=0;
    end_if;
    if Таймер<=0 then Таймер=100;
    end_if;

END_SFC_STEP
```

Рис. 2.10. Шаг 0 (начальный) на языке ST

Режимы процесса, соответствующие остановке и принудительному слива продукта в ручном режиме реализуем как альтернативные последовательности, т.е. представим их отдельными ветвями диаграммы. Для этого выделите ЛКМ начальный шаг на SFC-диаграмме и нажмите кнопку *«Создать шаг/переход»*  . Появится ветвь, параллельная ветви нормального протекания процесса, представленная единственным шагом – *«Шаг 5»* и предваряющим его условием перехода *«Переход 4»*. Переименуйте *«Переход 4»* в *«СТОП»*.

Данная ветвь диаграммы, которая начинается с шага 5, соответствует остановке технологического процесса, что возможно при неактивной команде *«Запуск» (Запуск=0)* и отсутствии команды на слив от оператора *(РучнСлив=0)*. Точно также, как и при проверке условия перехода *«Запуск»*, используем блок *«Равенство»* == для проверки равенства нулю значения аргумента *«Запуск»* (рис. 2.11) и блока *«Инверсия» !X* для недопущения перехода при аргументе *РучнСлив=1*. Сочетание логических условий также проверим с помощью блока логического умножения *«И» X&&Y*. Таким образом, при *Запуск=0* на выходе блока *«==»* будет сформирована единица (иначе – 0). Единица на выходе блока *«!X»* будет только при *РучнСлив=0*, а переход к *шагу 5* будет выполнен только тогда, когда оба этих условия будут истинны, иначе вся ветвь будет игнорироваться.

Обращаем ваше внимание на то, что условия перехода для параллельных ветвей должны быть уникальными, иначе программа будет выполнять ту ветвь, условие которой окажется истинным первым при проверке слева-направо (согласно порядку создания). При этом до проверки остальных ветвей, даже если их условия истинны, компилятор не дойдет. Это свойство можно использовать для реализации приоритетов, когда наиболее приоритетные ветви (при условии выполнения одного и того же условия) располагаются левее и, соответственно, будут выполняться первыми.

Таким образом, например, при поступлении команды *«Запуск»* при остановленном процессе (*Запуск=0*, исполняется вторая параллельная ветвь диаграммы), программа снова будет выполняться по первой ветви диаграммы *(Запуск=1)*, соответствующей режиму нормального протекания процесса.

Поскольку в режиме *«СТОП»* все исполнительные устройства должны быть отключены/закрыты, то при написании программы управления для пятого шага на языке ST, сделаем это простым присвоением значения *«0»* всем аргументам, соответствующим исполнительным устройствам (рис. 2.12).

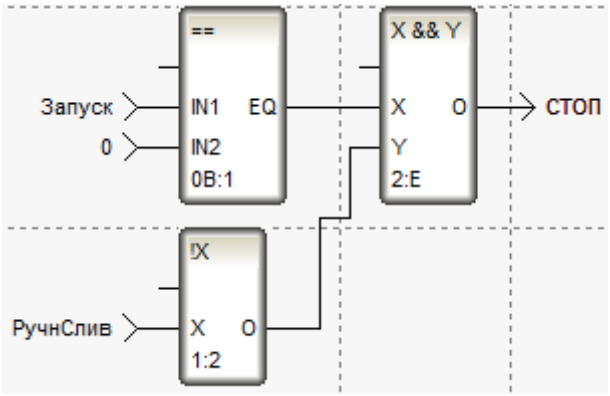


Рис. 2.11. Переход 4 на языке FBD

```
SFC_STEP "ШАГ 5"
    VAR_INOUT Клапан_1 : USINT; END_VAR
    VAR_INOUT Клапан_2 : USINT; END_VAR
    VAR_INOUT Уровень : REAL; END_VAR
    VAR_INOUT ТЭН : USINT; END_VAR
    VAR_INOUT Температура : REAL; END_VAR
    VAR_INOUT Миксер : USINT; END_VAR
    VAR_INOUT Слив : USINT; END_VAR
    VAR_INOUT Таймер : REAL; END_VAR
    VAR_INOUT Цикл : USINT; END_VAR
    VAR_INPUT Запуск : USINT; END_VAR
    VAR_INPUT РучнСлив : USINT; END_VAR

    Слив=0;
    Клапан_1=0;
    Клапан_2=0;
    ТЭН=0;
    Миксер=0;

END_SFC_STEP
```

Рис. 2.12. Шаг 5 на языке ST

Наконец, третья альтернативная ветвь SFC-диаграммы соответствует режиму ручного слива. Для создания ветви к альтернативной двум предыдущим выделите ЛКМ начальный шаг на SFC-диаграмме и нажмите кнопку *«Создать шаг/переход»*  . Переименуйте *«Переход 5»* в *«СБРОС»*.

Поскольку в условии перехода *«Запуск»* и *«СТОП»* уже была выполнена блокировка перехода при наличии команды ручного слива *(РучнСлив=1)* за счет инверсии, то в условии пятого перехода достаточно проверить наличие этой команды, что сделаем с помощью разомкнутого контакта, связанного с переменной *«РучнСлив»*. При *РучнСлив=1* (контакт проводит ток) значение правой шины также равно единице и наоборот (рис. 2.13). Поскольку при написании условий перехода на языке LD, если в переходе нет ассоциации с переменной условия (*«СБРОС»* в данном случае), то переход осуществляется по единичному состоянию правой шины.

Рис. 2.13. Переход 5 на LD

Действия в третьем альтернативном варианте возможного развития процесса фактически таковы же, как и при остановке системы (вторая альтернативная ветвь) с той лишь разницей, что необходимо удерживать открытым сливной клапан *(Слив=1)*. Листинг соответствующей программы на языке ST представлен на рисунке 2.14.

```
SFC_STEP "ШАГ 6"
    VAR_INOUT Клапан_1 : USINT; END_VAR
    VAR_INOUT Клапан_2 : USINT; END_VAR
    VAR_INOUT Уровень : REAL; END_VAR
    VAR_INOUT ТЭН : USINT; END_VAR
    VAR_INOUT Температура : REAL; END_VAR
    VAR_INOUT Миксер : USINT; END_VAR
    VAR_INOUT Слив : USINT; END_VAR
    VAR_INOUT Таймер : REAL; END_VAR
    VAR_INOUT Цикл : USINT; END_VAR
    VAR_INPUT Запуск : USINT; END_VAR
    VAR_INPUT РучнСлив : USINT; END_VAR

    Слив=1;
    Клапан_1=0;
    Клапан_2=0;
    ТЭН=0;
    Миксер=0;

END_SFC_STEP
```

Рис. 2.14. Шаг 6 на языке ST

Выбранный порядок расположения альтернативных ветвей обеспечивает при деактивации команды на слив продукта оператором *(РучнСлив=0)* при отсутствии активной команды запуска *(Запуск=0)* отработку второй альтернативной ветви диаграммы – закрытие клапана выдачи продукта. При деактивации команды на слив продукта оператором *(РучнСлив=0)* при наличии активной команды запуска *(Запуск=1)* будет выполняться первая альтернативная ветвь, что не позволит закрыть сливной клапан (это делается во второй ветви, а она не будет выполняться) вплоть до полного слива продукта, что и требуется по условию.

Таким образом, итоговый вид полученной программы управления химическим реактором на языке SFC представлен на рисунке 2.15.

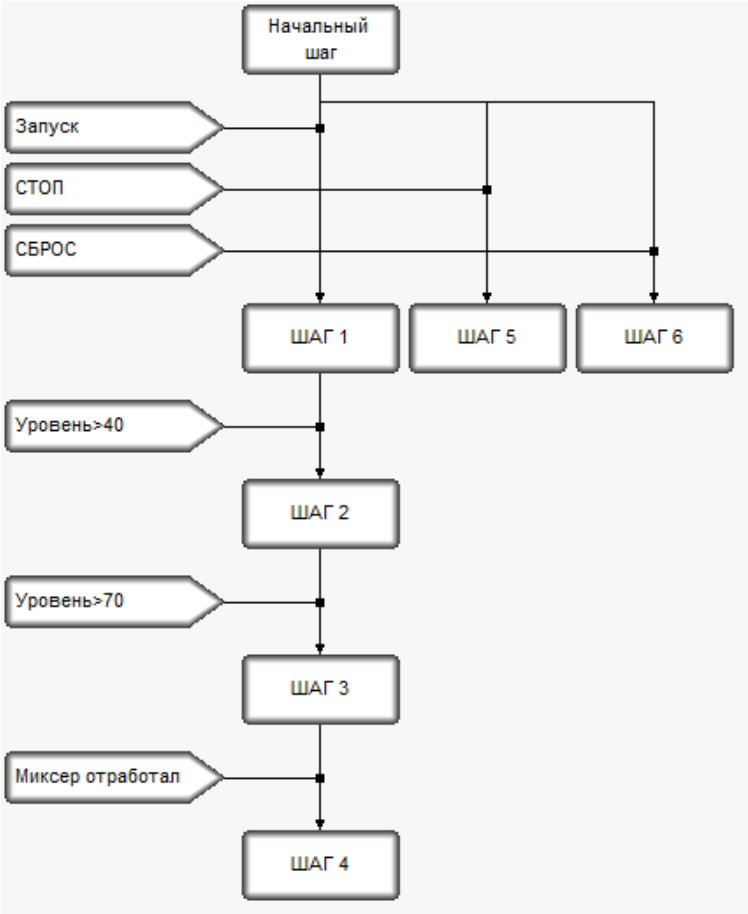


Рис. 2.15. Программа управления химическим реактором на языке SFC

Программа-эмулятор нагрева емкости реактора.

Программу-эмулятор температуры напишем на графическом языке FBD.

Нажмите ПКМ на созданном узле RTM_1 и выберите в контекстном меню *«Создать компонент → Программа»*. Двойным нажатием ЛКМ на канале вызова программы, откройте ее на редактирование в *РШП*.

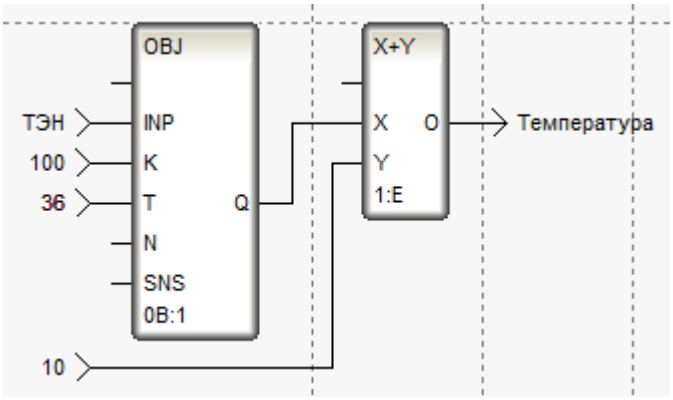


Рис. 2.16. Программа-эмулятор нагрева на FBD

Нажмите ЛКМ на пункте *«Аргументы» РШП* и создайте в табличном редакторе аргументов программы два аргумента со следующими параметрами: *ТЭН IN USINT, Температура OUT REAL*.

Нажмите ЛКМ на заголовке программы. В появившемся окне выбора языка программирования выберите графический язык *FBD*. Нажмите в окне выбора языка кнопку *«Принять»*.

В данном случае программа будет состоять из двух блоков *«Модель объекта» OBJ* и *«Сложение» X+Y* (рис. 2.16). Пример данной программы подробно разбирался ранее (см. Разработка программы эмулятора-регулятора температуры агента в емкости, ЛР №2 в [6]). Единственным отличием является то, что регулятор (блок *«Гистерезис» HSTR*) в данной работе является частью программы управления на языке SFC и в программе-эмуляторе не присутствует.

По сигналу управления (аргумент *«ТЭН»*), принимающему значения *«0»* либо *«1»* блоком модели объекта *«OBJ»* рассчитывается прирост (нагрев) температуры продукта в емкости при *ТЭН=1* или снижение температуры при *ТЭН=0*. Коэффициент *K=100* характеризует мощность нагревателя, т.е. максимально температура может быть увеличена на 100^0C. Постоянная времени *T* характеризует длительность (инерционность) тепловых процессов, протекающих в реакторе. Т.к. по условию нагрев производится за время около полутора минут, то в секундах это составит 90 секунд. Так как 100% от установившегося значения достигается за время *5T,* то *T=90/5=18* секунд. Учитывая, что узел пересчитывается дважды в секунду, то число циклов равно удвоенному времени в секундах, т.е. 36.

Константа «10», поданная на один из входов блока сложения *«X+Y»*, учитывает начальную температуру по условию, от которой осуществляется нагрев.

Программа-эмулятор уровня в емкости реактора.

Программу-эмулятор уровня напишем на языке ST.

Нажмите ПКМ на созданном узле RTM_1 и выберите в контекстном меню *«Создать компонент → Программа»*. Двойным нажатием ЛКМ на канале вызова программы, откройте ее на редактирование в *РШП*.

Нажмите ЛКМ на пункте *«Аргументы»* *РШП* и создайте в табличном редакторе аргументов программы четыре аргумента со следующими параметрами: *Клапан1 IN USINT, Клапан2 IN USINT, Сброс IN USINT, Уровень OUT REAL*.

Нажмите ЛКМ на заголовке программы. В появившемся окне выбора языка программирования выберите язык *ST*. Нажмите в окне выбора языка кнопку *«Принять»*.

```
PROGRAM
    VAR_OUTPUT Уровень : REAL; END_VAR
    VAR_INPUT Клапан1 : USINT; END_VAR
    VAR_INPUT Клапан2 : USINT; END_VAR
    VAR_INPUT Сброс : USINT; END_VAR

    if Уровень<100 then
    case Клапан1 of
    1: Уровень=Уровень+1;
    0: Уровень=Уровень+0;
    end_case;
    case Клапан2 of
    1: Уровень=Уровень+1;
    0: Уровень=Уровень+0;
    end_case;
    else Уровень=100;
    end_if;
    if Уровень>0 then
    case Сброс of
    1: Уровень=Уровень-2.5;
    0: Уровень=Уровень-0;
    end_case;
    else goto far;
    end_if;
    far: if Уровень<0 then Уровень=0;
    end_if;

END_PROGRAM
```

Рис. 2.17. Программа-эмулятор уровня на ST

Введите в редакторе текст, как показано на рисунке 2.17. Предполагается, что при открытии как первого, так и второго клапанов уровень увеличивается на 1% за каждый цикл пересчета, а при открытии клапана сброса – уменьшается на 2,5% за цикл пересчета.

Программа подробно рассматривалась в разделе «Разработка программы-эмулятора изменения уровня в емкости», ЛР№1. Небольшое отличие связано с добавлением второго клапана, который, как и первый, отвечает за наполнение емкости (в ЛР№1 эту функцию выполнял насос). Вместо аргумента *«Клапан»* (как в первой работе) за уменьшение уровня отвечает аргумент *«Слив»*.

Кроме того, для недопущения формирования отрицательных значений уровня на выходе программы, в части расчета уменьшения уровня используется переход к метке *«far:»*, где окончательно перезаписываются отрицательные значения нулевыми.

Компиляция и отладка. Выполните компиляцию программы-эмулятора уровня клавишей *F7*. Откройте окно переменных, нажав кнопку *«Переменные»*  на панели инструментов отладчика, и запустите программу на циклическое выполнение, нажав клавишу *F5* (см. Компиляция и отладка, ЛР№1 в [6]).

Вводите в аргументы *«Клапан1»* или *«Клапан2»* значения *«0»* либо *«1»* и наблюдайте за увеличением уровня. То же сделайте с аргументом *«Сброс»* и

наблюдайте за уменьшением уровня. Проверьте совместное функционирование клапанов и работу программы в крайних значениях 0 и 100.

Выполните компиляцию программы-эмулятора температуры клавишей *F7*. Откройте окно переменных, нажав кнопку *«Переменные»* и запустите программу на циклическое выполнение, нажав клавишу *F5* (см. Компиляция и отладка, ЛР№1 в [6]). Вводите в аргумент *«ТЭН»* значения *«0»* либо *«1»* и наблюдайте за изменением температуры.

Выполните компиляцию программы управления клавишей *F7*. Из-за особенностей исполнения программы на SFC, заключающейся в необходимости создавать для полноценной отладки глобальные переменные, работу программы управления лучше проверять непосредственно в профайлере.

Создание и настройка базы каналов. Создадим каналы для организации обмена данными между экраном и программами (см. Создание базы каналов, ЛР№1 в [6]). Разверните узел RTM_1 в навигаторе проекта и нажмите ПКМ на канале вызова шаблона экрана. Выберите в контекстном меню пункт *«Свойства»*. В открывшемся окне переключитесь на вкладку *«Аргументы»*. Далее нажмите ЛКМ на кнопку *«Создать по аргументам каналы с привязкой»* на панели работы с аргументами. Будет создано одиннадцать каналов соответствующего аргументам типа.

Теперь необходимо связать созданные каналы с программами. Нажмите ПКМ на канале вызова шаблона программы *«Программа#1»* (программа управления на SFC) и выберите в контекстном меню пункт *«Свойства»*. В открывшемся окне переключитесь на вкладку *«Аргументы»*.

Перетащите каналы из навигатора проекта, удерживая ЛКМ, на строки аргументов с аналогичными именами в табличном редакторе привязок аргументов. Привязки будут созданы автоматически.

Точно также откройте свойства и затем вкладку *«Аргументы»* программы *«Программа#2»* (эмулятор температуры на FBD). Перетащите каналы *«ТЭН»* и *«Температура»* из навигатора проекта, удерживая ЛКМ, на строки аргументов с аналогичными именами в табличном редакторе привязок аргументов.

Повторите процедуру для канала вызова *«Программа#3»*. На этот раз потребуется перетащить из навигатора проекта, удерживая ЛКМ, на соответствующие строки аргументов каналы *«Клапан1»*, *«Клапан2»*, *«Уровень»* и *«Слив»* для аргумента *«Сброс»*.

Сохранение проекта и запуск на исполнение в профайлере. Нажмите ЛКМ кнопку *«Сохранить»* и затем *«Сохранить для МРВ»* на главной панели инструментов (см. Сохранение и подготовка проекта к запуску ЛР№1 в [6]). Откройте профайлер кнопкой *«Запустить профайлер»* на главной панели инструментов. В профайлере еще раз нажмите ЛКМ кнопку *«Запуск/Останов»*.

Проверьте работоспособность проекта и системы управления, наблюдая за соответствием очередности и содержания фаз обработки техническому заданию. Проведите принудительную остановку и запуск процесса кнопками *«СТОП»* и *«ПУСК»*. Проверьте работоспособность режима ручного слива продукта из емкости как в рабочем состоянии, так и из состояния *«СТОП»*. Убедитесь, что в рабочем состоянии клапан слива остается открытым и слив осуществляется до полного опорожнения емкости, даже при команде закрытия сливного клапана от оператора.

Рис. 2.18. Проект, запущенный на исполнение в профайлере

Итоговый результат разработки, запущенный на исполнение в профайлере, показан на рисунке 2.18.

Для останова профайлера нажмите ЛКМ кнопку *«Запуск/Останов»* . После этого закройте профайлер.

2.4. Содержание отчета

Отчет должен включать:

1) скриншот экрана проекта, запущенного на исполнение в профайлере, с указанием типов использованных ГЭ и настраиваемых свойств этих ГЭ;

2) разработанный алгоритм программы последовательностного управления по ГОСТ 19.701-90;

3) скриншот диаграммы последовательностного управления на языке SFC, а также скриншоты всех шагов и переходов с текстовым пояснением их работы и описанием синтаксиса использованных операторов и функциональных блоков;

4) самостоятельное задание. Каждый из шагов в дополнение к рассмотренному в работе языку реализуйте также на языках ST, FBD и LD;

5) вывод по проделанной работе.

2.5. Контрольные вопросы

1. Что такое «функциональная карта»? Для чего они применяются?
2. Из каких элементов состоят SFC-диаграммы?
3. При каком условии выполняется переход?
4. На каких языках МЭК могут быть написаны шаги, а на каких - переходы?
5. Раскройте понятие «простых последовательностей».
6. Что характерно для альтернативных параллельных последовательностей?
7. Что такое «синхронные параллельные последовательности»?
8. Какие осложнения могут возникнуть при реализации альтернативных параллельных последовательностей? На что обратить внимание?
9. Каковы негативные последствия неправильной реализации синхронных параллельных последовательностей?
10. О чем свидетельствуют две горизонтальные черты в SFC-диаграмме?

ЗАКЛЮЧЕНИЕ

Программное обеспечение является одним из важнейших видов обеспечения автоматизированных систем управления технологическими процессами (АСУТП).

В данном лабораторном практикуме были продемонстрированы практические аспекты разработки программ комбинационного (логического) управления, а также управления последовательностями событий, используя для этой цели языки программирования промышленных контроллеров стандарта МЭК 61131 Ladder Diagram (LD), Function Block Diagram (FBD) и Sequential Function Chart (SFC). Разработка велась в редакторе программ инструментальной системы SCADA Trace Mode 6.

Должное внимание уделено совершенствованию навыков разработки интерфейса оператора, вопросам переключения (смены) режимов управления, организации фиксации событий, обработке алармов (тревог) и т.д.

Надеемся, что данное пособие будет полезным не только студентам направления «Автоматизация технологических процессов и производств», но и действующим специалистам АСУТП, занятым в сфере разработки прикладного программного обеспечения и работающим со SCADA-системами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Денисенко В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. / В.И. Денисенко. – М.: Горячая Линия-Телеком, 2009 – 608 с.: ил. - ISBN 978-5-9912-0060-8

2. Trace Mode 6. Руководство пользователя. Интегрированная SCADA/HMI-SOFTLOGIC-MES-EAM-HRM-система для разработки АСУ ТП, АСКУЭ и систем управления производством. 14-е изд. – М.: «AdAstra Research Group», Ltd, 2011. – 619 с. ил.

3. Андреев Е.Б. Программные средства систем управления технологическими процессами в нефтяной и газовой промышленности: учеб. пособие для вузов/ Е.Б.Андреев, В.Е.Попадько.-М.:ФГУП Изд-во "Нефть и газ" РГУ нефти и газа им. И.М.Губкина.Ч.1.-2005.-268с. - ISBN 978-5-8365-0316-1

4. Орлов С.А. Технологии разработки программного обеспечения: Разработка сложных программных систем: Учебник для вузов. - 3-е изд.-СПб.:Питер,2004.-527с.:ил.-(Учебник для вузов) - ISBN 5-94723-145-X

5. Харазов В.Г. Интегрированные системы управления технологическими процессами: учеб. пособие для вузов.-СПб.:Профессия, 2009.- 592с. :ил. - ISBN 978-5-93913-176-6

6. Шкромадо А.А. SCADA-системы / А.А. Шкромадо, Р.В. Шестов, А.Н.Бирюков. – Самара: Самар. гос. техн. ун-т, 2017. – 64 с.: ил.

СОДЕРЖАНИЕ

Практикум

ШКРОМАДО Антон Алексеевич
ШЕСТОВ Руслан Владимирович
БИРЮКОВ Алексей Николаевич

ТЕХНИЧЕСКИЕ И ПРОГРАММНЫЕ СРЕДСТВА КОМПЛЕКСНОЙ АВТОМАТИЗАЦИИ

Редакторы:

Е. С. Захарова
И. А. Назарова

Подписано в печать 27.02.17 г.
Формат 60x84 1/16. Бумага офсетная
Усл. п. л. 3,4 Уч.-изд. л. 3,4
Тираж 100 экз. Рег. № 2/17sf

Федеральное государственное бюджетное образовательное
Учреждение высшего образования
«Самарский государственный технический университет»
443100, г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
Филиал в г. Сызрани, 446001, г. Сызрань, ул. Советская 45