Министерство науки и высшего образования Российской Федерации

Амурский государственный университет

А.Н. Рыбалёв

ПРОГРАММИРОВАНИЕ ПЛК В CODESYS

Типовые системы управления

Учебное пособие

Благовещенск

Издательство АмГУ

2023

ББК 32.965.7 я73
Р 93

Рекомендовано
учебно-методическим советом университета

Рецензенты:

Ильченко А.В., исполнительный директор ООО «Ключель»;

Мясоедов Ю.В., декан энергетического факультета АмГУ, канд. техн. наук, профессор

Рыбалёв А.Н. Программирование ПЛК в CoDeSys. Типовые системы управления. Учебное пособие. – Благовещенск: Амурский гос. ун-т, 2023.

Пособие предназначено для студентов 3 и 4 курсов направления подготовки «Автоматизация технологических процессов и производств» и соответствует рабочим программам дисциплин «Средства автоматизации и управления», «Программное обеспечение систем управления», «Автоматизация технологических процессов и производств». Может также использоваться при проведении специализированных курсов повышения квалификации.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программируемые логические контроллеры (ПЛК) являются основным средством автоматизации технологических процессов во всех отраслях промышленности. На сегодняшний день существует большое разнообразие ПЛК, отличающихся по назначению, архитектуре, информационной мощности и другим параметрам. Многочисленны также и системы программирования ПЛК. Многие изготовители, особенно это касается крупных электротехнических компаний, самостоятельно разрабатывают программное обеспечение (компьютерные программы для персональных компьютеров), предназначенное для работы с их изделиями: программирования, отладки, мониторинга, сбора информации, визуализации технологического процесса и т.д. Другие производители ПЛК интегрируют свои решения с универсальными средами разработки, такими как CoDeSys, IsaGRAPH и др. Эти системы полностью удовлетворяют стандарту IEC 61131-3, предлагая пользователю типовые программные решения и пять языков программирования, рекомендованные стандартом. Необходимо отметить, что в последнее время и «фирменное» ПО, оставаясь «заточенным» на архитектуру конкретных ПЛК и их семейств, старается следовать этим рекомендациям, пусть не всегда и не в полном объеме. Это в принципе позволяет производить разработку в одних средах, а использовать полученные решения – в других. Наиболее перспективными в данном плане инструментами являются текстовый язык высокого уровня ST (Structured Text) и графические языки LD (Ladder Diagrams) и FBD (Functional Block Diagrams). В настоящей работе задействована бесплатно распространяемая среда разработки CoDeSys 2.3 [1], совместимая со всеми версиями ОС Windows, используемыми в настоящее время, и предъявляющая минимальные требования к аппаратным средствам компьютера. Программы апробируются и отлаживаются либо в режиме эмуляции, т.е. в самой среде разработки, либо на виртуальном контроллере PLCWinNT24, входящем в состав пакета. Второй вариант предпочтительнее для случаев, когда требуется тщательно выдерживать временны*е* параметры. В частности, это касается систем автоматического регулирования. Виртуальный контроллер является самостоятельной программой, предназначенной для работы в реальном времени (в той «степени реальности», с которой операционная система Windows ему это позволяет).

Определенную проблему при разработке программного обеспечения для ПЛК представляет отладка. Программисту, пишущему программу для персонального компьютера, ничего кроме компьютера не нужно, так как «объектом управления» для его программы является сам компьютер (здесь мы не принимаем во внимание вопросы совместимости). Объектом управления для программы ПЛК является технологическое оборудование, и проводить на нем отладку крайне нежелательно, если вообще возможно. Поэтому ее приходится проводить на моделях, и на программиста возлагается «двойная работа»: помимо собственно управляющей программы, он должен со-

здать и модель объекта управления (причем, как правило, – в первую очередь). Ситуацию несколько смягчает то обстоятельство, что модель должна реализовать только те аспекты состояния и поведения объекта, которые существенны для управляющей программы, но и это – достаточно сложная задача, часто превосходящая по сложности разработку самой этой программы.

Модель объекта управления может быть построена в таких системах имитационного моделирования, как Simulink Matlab, а межпрограммный обмен организован посредством доступных протоколов, в частности OPC [2]. При таком подходе в CoDeSys обязательно используется виртуальный контроллер, поскольку режим эмуляции не поддерживает взаимодействие с OPC-сервером. Преимущества подхода очевидны: во-первых, Simulink является мощным средством, позволяющим создать модель практически сколь угодно сложного объекта, во-вторых, сам «объект» отделяется от «системы управления», становясь самостоятельной сущностью, что дает возможность четко обозначить границы и связи между ними.

Однако применение внешних моделей имеет один существенный недостаток, а именно – сложность создаваемой программной конструкции. Помимо бесплатной среды CoDeSys требуется наличие системы имитационного моделирования, которая, как в случае с Simulink, входит в состав «тяжелого» и отнюдь не бесплатного математического пакета. Часто вызывает затруднения и установка соединения между программами.

В данной работе модели всех объектов управления реализованы непосредственно в CoDeSys, как часть программы ПЛК, что значительно упрощает разработку. «Объекты» живут в отдельных подпрограммах, обмениваясь «сигналами» с системой управления (другими подпрограммами) через общий список глобальных переменных. При переносе результатов разработки на целевую платформу потребуется всего лишь удалить подпрограммы-объекты и перенести часть глобальных переменных в конфигурационные списки, привязав их к реальным входам и выходам ПЛК (или сетевым переменным, если обмен производится по промышленной сети).

В пособии рассматриваются четыре системы программно-логического управления, шесть систем автоматического регулирования и одна система числового программного управления. Все системы, будучи весьма упрощенными, в то же время являются законченными в том смысле, что они полностью решают те задачи, которые на них возлагаются на практике. В частности, системы регулирования помимо собственно автоматического регулирования допускают и ручное управление исполнительными механизмами.

Пособие почти не содержит теоретических сведений и ориентировано исключительно на практический курс. Многие детали намеренно не раскрываются в расчете на интуицию лиц, выполняющих работы по воспроизведению систем. Это касается по большей части деталей настройки элементов визуализации. При проведении занятий возможен и такой подход, при котором часть проекта, например, модель объекта или экран визуализации, выдается в уже готовом виде и требуется воспроизвести оставшуюся часть.

1 ПРОГРАММНО-ЛОГИЧЕСКОЕ УПРАВЛЕНИЕ

1.1 Комбинационная схема. Управление вентиляцией и освещением

В комбинационных схемах выходы системы однозначно определяются входами в каждый момент времени. Практически это означает, что программа управления не нуждается в каких-либо дополнительных переменных.

Ниже рассмотрена система управления освещением и вентиляцией помещения [2]. Система осуществляет:

– автоматическое включение вентилятора по сигналу датчика загазованности;

– автоматическое включение освещения в присутствии человека по сигналу фотореле;

– ручное управление вентиляцией и освещением (является приоритетным по отношению к сигналам датчиков).

Все переменные программы являются входами и выходами контроллера. В программе для виртуального контроллера (или вообще при отсутствии контроллера) эти переменные логично сделать глобальными, поскольку в реальном ПЛК они действительно являются таковыми, хотя и объявляются отдельно. Список глобальных переменных показан на рис. 1.

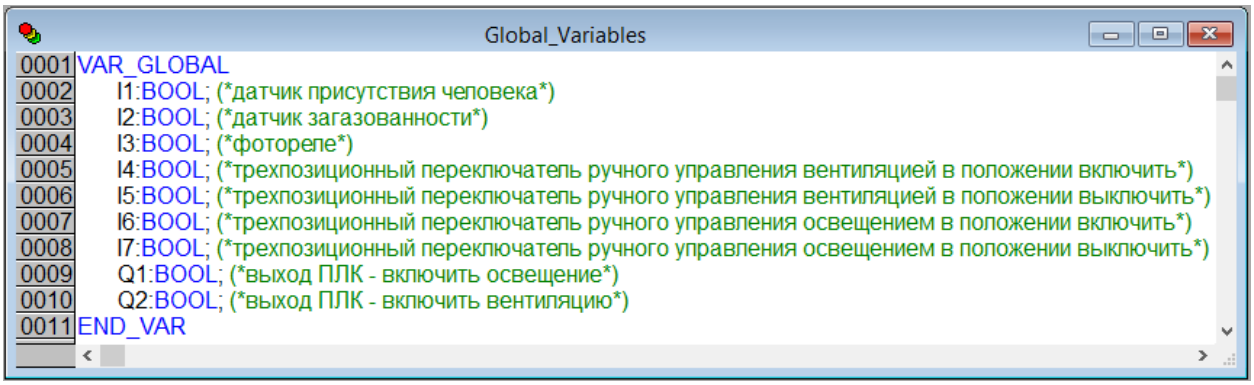


Рис. 1. Список глобальных переменных

Экран визуализации, предназначенный для отладки программы и демонстрации ее работы, показан на рис. 2.

Управление входами осуществляется с помощью кнопок с фиксацией и «подтверждается» изменением цвета отдельных элементов визуализации, а также изменением видимости элементов, составляющих фигурку человека. Включение освещения и вентиляции приводит к изменению цветов «помещения» в целом и «вентилятора» соответственно.

Отдельно рассмотрим реализацию трехпозиционных переключателей на базе трех кнопок с фиксацией: ON, AUTO и OFF. Нажатие на каждую из этих кнопок должно приводить к изменению сразу двух переменных, например I4 и I5. Поэтому кнопки привязываются к выполнению программ «внутреннего присваивания» INTERN ASSIGN, как показано на рис. 3 для кнопки перевода в автоматический режим системы управления вентиляцией.

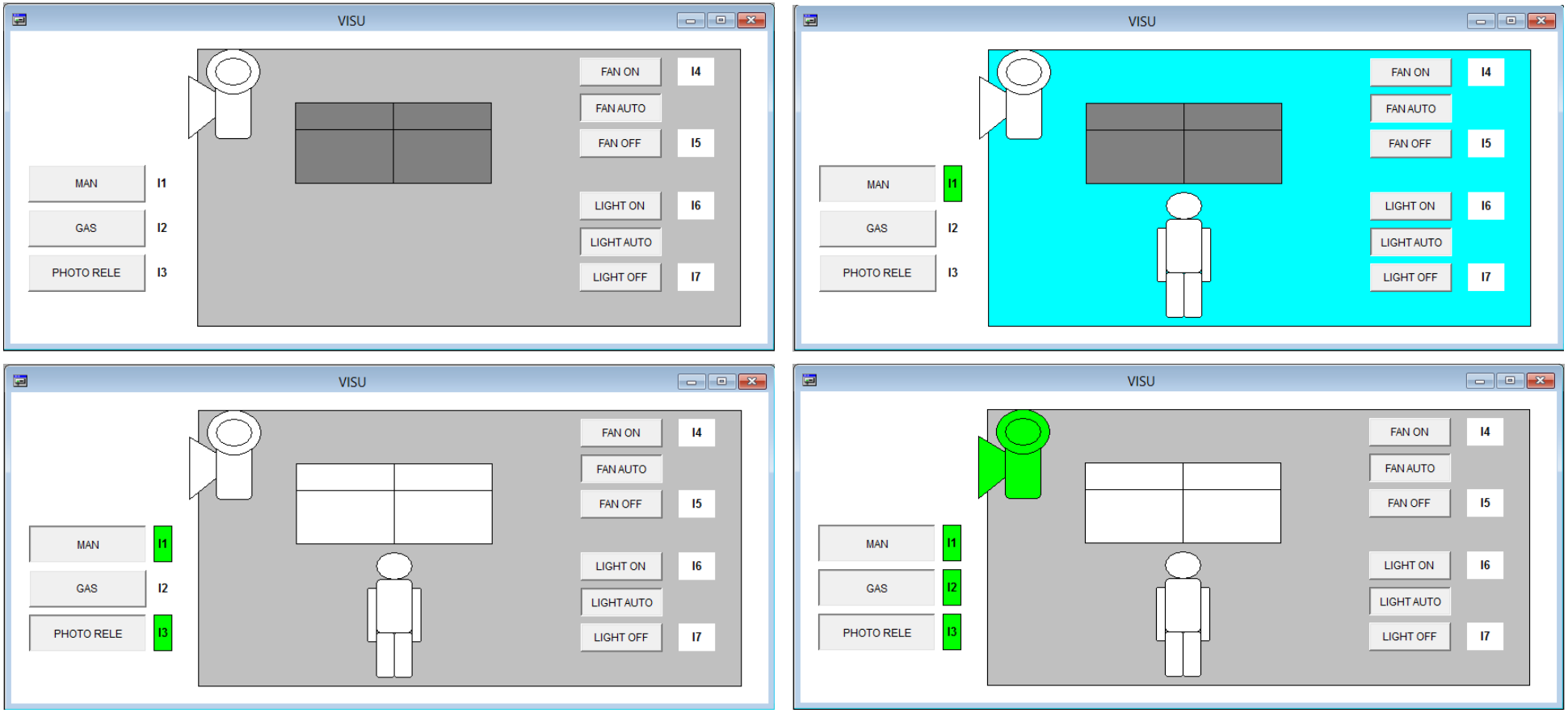


Рис. 2. Экран визуализации в работе

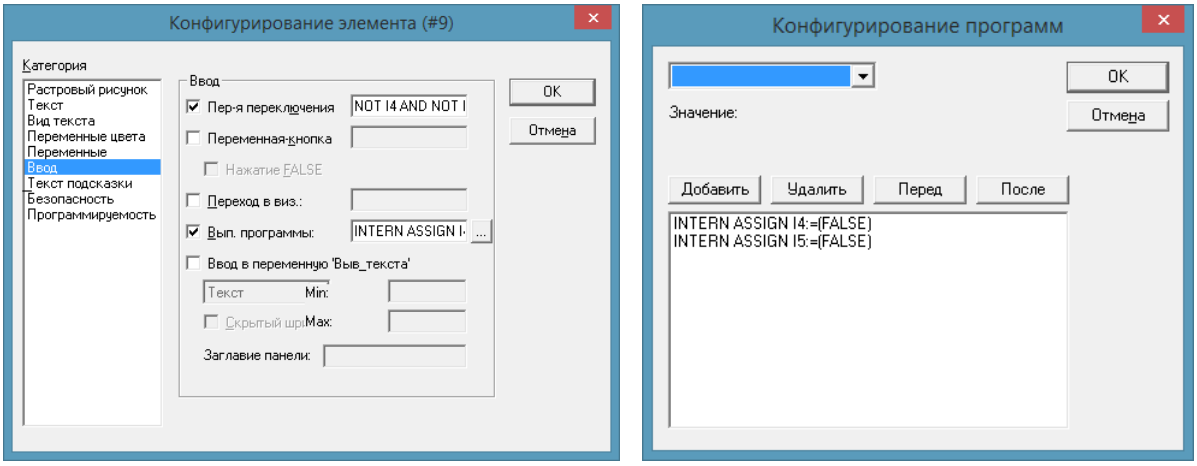


Рис. 3. Настройка кнопки для изменения значений нескольких переменных

Алгоритмы включения освещения и вентиляции представлены логическими выражениями [2]:

$$Q_1 = I_1 I_6 + I_1 \overline{I}_3 \overline{I}_7,$$

$$Q_2 = \overline{I}_1 I_2 + I_1 I_2 \overline{I}_5 + I_1 I_4.$$

Освещение *включается* (Q_1), если

[*человек находится в помещении* (I_1) **И** *дан приказ на включение освещения* (I_6)]

ИЛИ (+)

[*человек находится в помещении* (I_1) **И** *фотореле не сработало* ($\overline{I}_3$) **И** *нет приказа выключить освещение* ($\overline{I}_7$)].

Вентиляция *включается* (Q_2), если

[*человек не находится в помещении* ($\overline{I}_1$) **И** *помещение загазовано* (I_2)]

ИЛИ

[*человек находится в помещении* (I_1) **И** *помещение загазовано* (I_2) **И** *нет приказа выключить вентилятор* ($\overline{I}_7$)]

ИЛИ

[*человек находится в помещении* (I_1) **И** *есть приказ включить вентилятор* (I_6)].

Программа управления на языке ST формирует логические выражения при помощи операторов NOT, AND и OR, рис. 4.

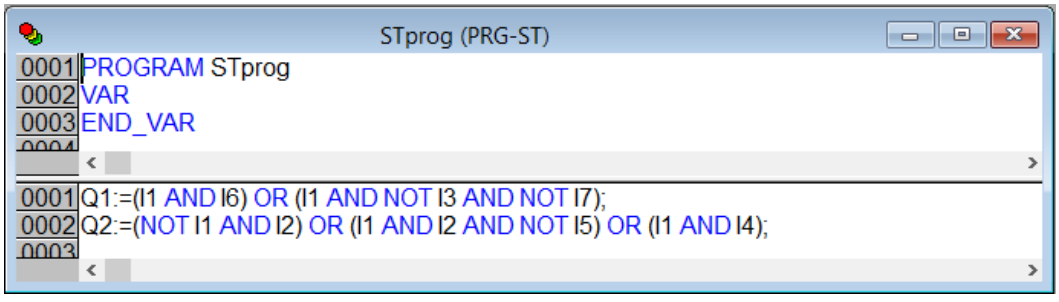


Рис. 4. Программа управления на языке ST

Программа управления на языке FBD задействует стандартные блоки-операторы AND и OR, а логическое отрицание производится путем инверсии на соответствующих входах, рис. 5.

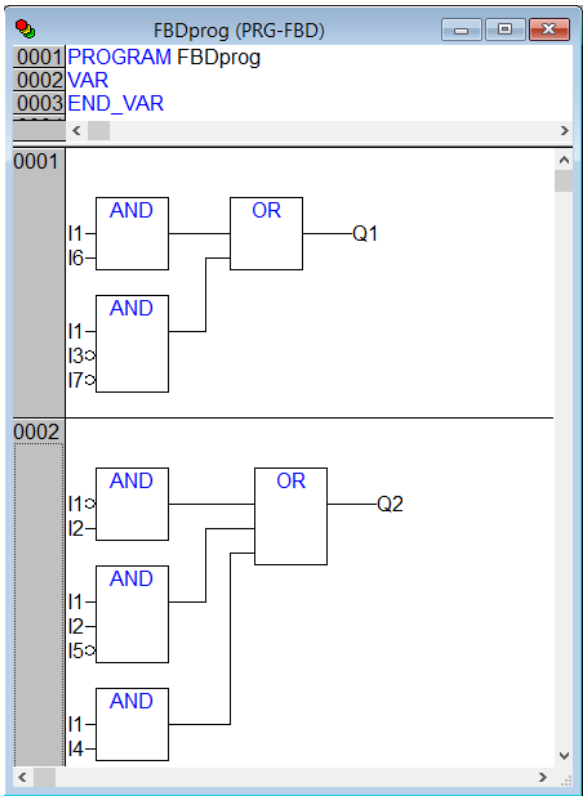


Рис. 5. Программа управления на языке FBD

Программа управления на языке LD выглядит как обычная релейно-контактная схема, в которой фигурируют «контакты», связанные с входами, и «катушки», связанные с выходами. Если логическая переменная участвует в выражении в инвертированном виде, задействуется нормально закрытый контакт. Логика «И» реализуется последовательным соединением контактов, а «ИЛИ» – параллельным, рис. 6.

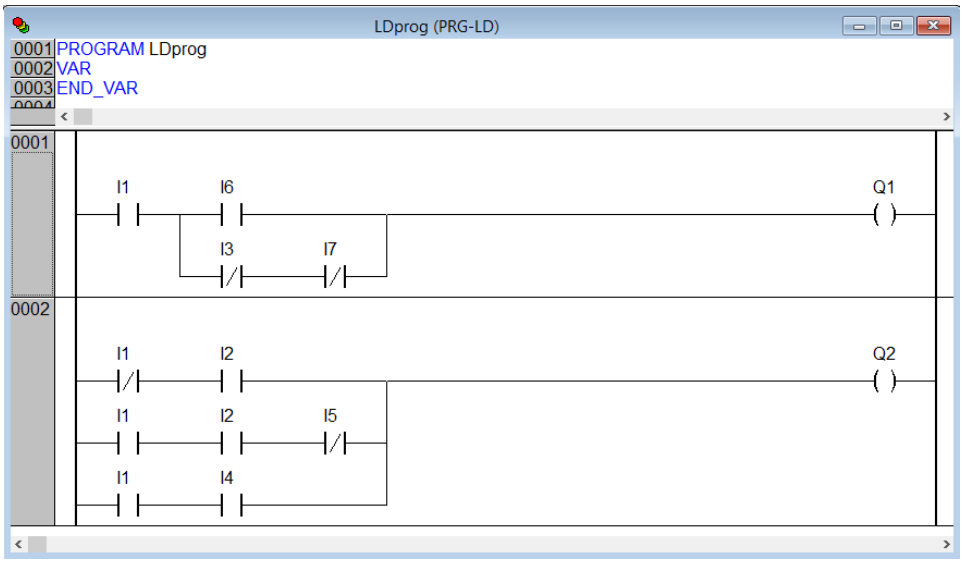


Рис. 6. Программа управления на языке LD

В «основной программе» PLC_PRG, которую лучше написать на языке ST, не делается ничего, кроме вызова на исполнение одной из трех представленных выше программ управления.

1.2 Многоточечное управление освещением. Простейший автомат

Рассмотрим сначала систему управления освещением длинного коридора с двумя дверями в двух его концах. Требуется, чтобы человек, войдя через одну из дверей, мог включить свет в коридоре, а, подойдя к другой двери, – выключить. Одно из решений показано на рис. 7 в виде электрической схемы.

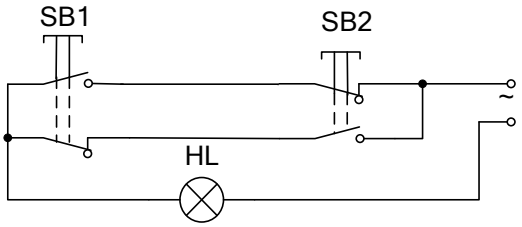


Рис. 7. Схема управления освещением коридора с двумя дверями

В схеме задействованы сдвоенные кнопки с фиксацией (или переключатели с перекидными контактами), установленные в разных концах коридора. Если обозначить их состояния переменным I_1 и I_2, а факт включения освещения – переменной Q, логику работу системы можно описать выражением:

$$Q = I_1\overline{I}_2 + \overline{I}_1 I_2.$$

Рассмотренный подход можно распространить и для общего случая, когда имеется одно большое помещение с множеством дверей, просто «копируя» показанные на рис. 7 цепи для каждой отдельной пары дверей. Таким образом, около каждой двери будет располагаться небольшой «пультик» с переключателями, число которых будет на единицу меньше, чем общее число дверей. Предполагается, что человек, заходя в помещение, заранее знает, через какую дверь он должен выйти, и, исходя из этого, выбирает соответствующий переключатель. Однако есть и более удачное решение, снимающее это ограничение и сводящее число переключателей на «пультике» до одного, правда, путем увеличения числа его полюсов.

Переключатели, задействованные в схеме, имеют два состояния: «кнопка нажата и зафиксирована» (обозначим его единицей) и «кнопка не нажата» (обозначим его нулем). Условия включения освещения для схемы, приведенной на рис. 7, могут быть представлены таблицей:

Таблица 1. Условия включения освещения для схемы на рис.6

Состояние переключателя 1	Состояние переключателя 2
1	0
0	1

Отметим, что строки таблицы полностью соответствуют «слагаемым»

из вышеприведенной формулы.

Для каждого переключателя имеется возможность «изменить ситуацию», т.е. включить или выключить свет, поскольку другой переключатель позволяет ему это сделать. Этот принцип может быть использован и в случае, если число переключателей более двух. Пусть у нас три переключателя (три двери). Тогда таблица будет иметь вид:

Таблица 2. Условия включения освещения для случая трех дверей

Состояние переключателя 1	Состояние переключателя 2	Состояние переключателя 3
1	1	1
1	0	0
0	1	0
0	0	1

Таблица составлена следующим образом. В первых двух столбцах представлены все комбинации состояний первых двух переключателей. Третий столбец построен так, чтобы для переключателей 1,2 (столбцов таблицы) оставшиеся два других переключателя (второй и третий, первый и третий) также представляли все возможные их состояния. Если мы исключим из таблицы первый столбец, оставшиеся второй и третий дадут все комбинации состояний второго и третьего переключателей. Исключая, аналогично, второй столбец, получим все комбинации состояний первого и третьего. Этим обеспечивается возможность включать и выключать освещение любым переключателем при любом состоянии всех других переключателей.

Соответствующее таблице логическое выражение:

$$Q = I_1 I_2 I_3 + I_1 \overline{I}_2 \overline{I}_3 + \overline{I}_1 I_2 \overline{I}_3 + \overline{I}_1 \overline{I}_2 I_3.$$

Добавим еще одну дверь и один переключатель, тогда таблица примет вид:

Таблица 3. Условия включения освещения для случая четырех дверей

Состояние переключателя 1	Состояние переключателя 2	Состояние переключателя 3	Состояние переключателя 4
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Соответствующее логическое выражение:

$$Q = I_1 I_2 I_3 I_4 + I_1 I_2 \overline{I}_3 \overline{I}_4 + I_1 \overline{I}_2 I_3 \overline{I}_4 + I_1 \overline{I}_2 \overline{I}_3 I_4 + \overline{I}_1 I_2 I_3 \overline{I}_4 + \\ + \overline{I}_1 I_2 \overline{I}_3 I_4 + \overline{I}_1 \overline{I}_2 I_3 I_4 + \overline{I}_1 \overline{I}_2 \overline{I}_3 \overline{I}_4.$$

Электрическая схема, построенная на основе данного выражения, показана на рис. 8.

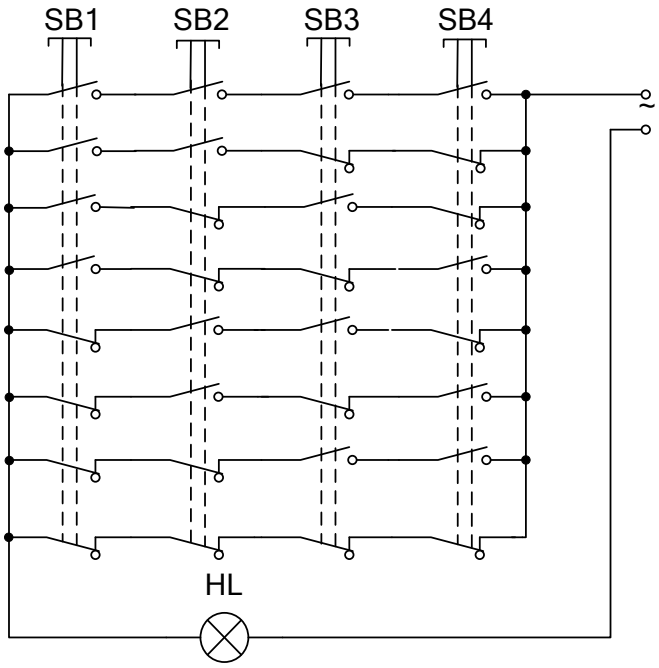


Рис. 8. Схема управления освещением помещения с четырьмя дверями

Рассмотрим программную реализацию системы. Список глобальных переменных ограничен входами и выходами контроллера, рис. 9.

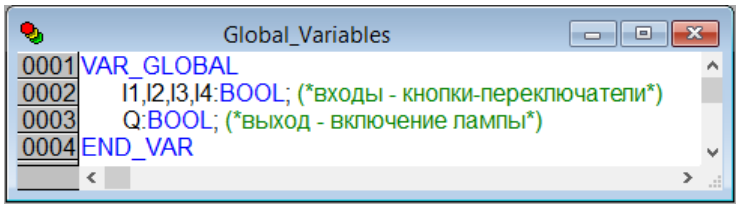


Рис. 9. Глобальные переменные

Программа на языке ST «один к одному» повторяет логическое выражение, приведенное выше, рис. 10.

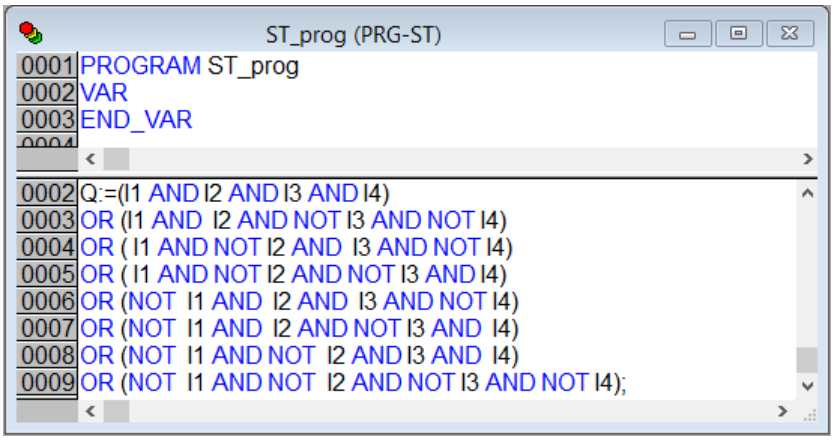


Рис. 10. Программа управления на языке ST

Программа на языке LD «воспроизводит» электрическую схему, рис. 11.

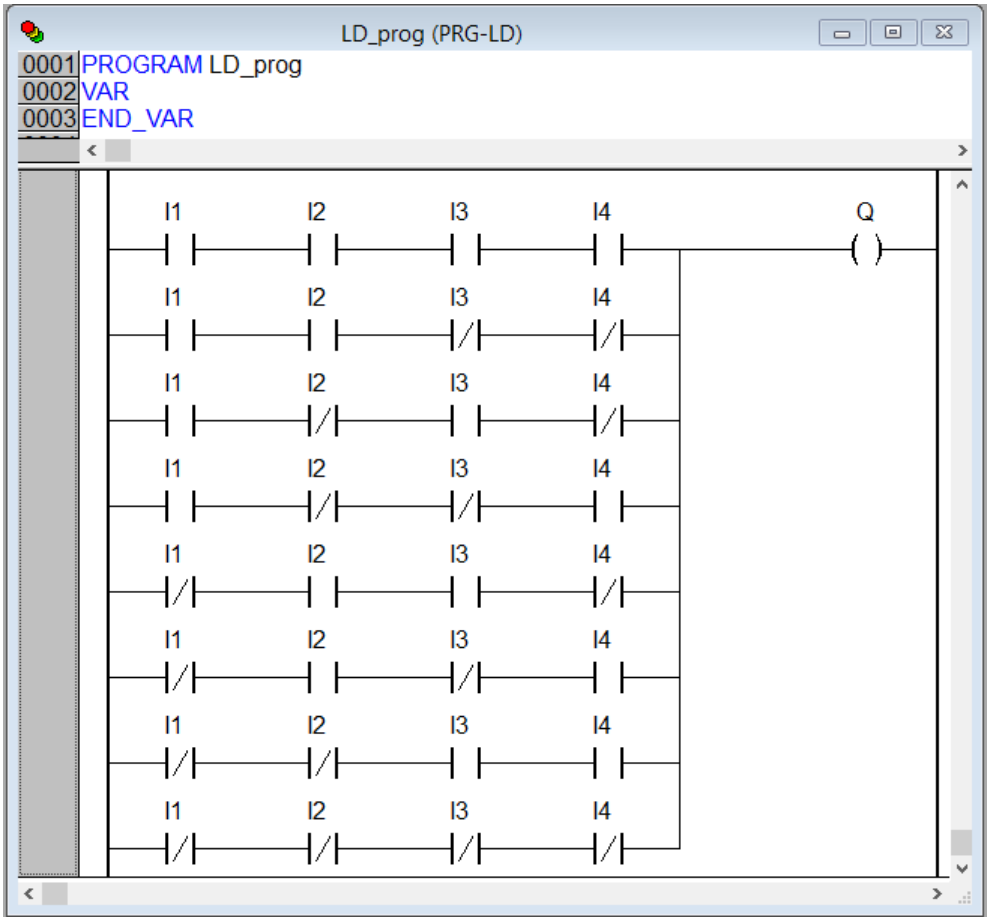


Рис. 11. Программа управления на языке LD

Окно визуализации, показанное на рис. 12, позволяет опробовать систему в действии. Оно содержит четыре кнопки с фиксацией, привязанные к входам, и «лампу», меняющую цвет при изменении выходной переменной.

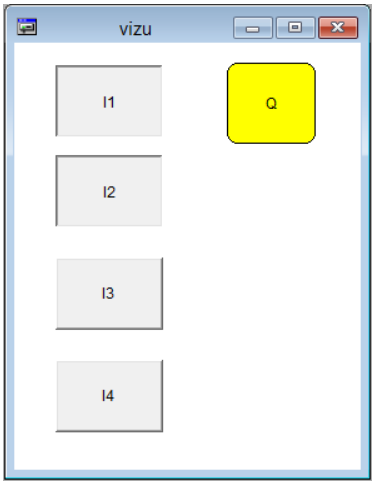


Рис. 12. Окно визуализации

В PLC_PRG производится вызов одной из двух программ: ST_prog или LD_prog.

Рассмотренное выше решение имеет очевидный недостаток: «лавинообразное» увеличение числа контактов переключателей при увеличении числа дверей. Если нам потребуется добавить пятую дверь, в схеме необходимо будет обеспечить полный набор всех возможных состояний уже четырех переключателей. Это означает, что схема будет содержать 16 участков коммутации, а каждый переключатель должен быть оснащен 16 контактами (8 нормально открытых и 8 нормально закрытых). Такую схему будет довольно сложно реализовать «в железе» просто потому, что вряд ли найдутся подходящие переключатели. Построить систему на базе ПЛК проще, так как подойдут обычные, одноконтактные, кнопки с фиксацией, а вся логика будет реализована программно. Усложнение программы не является большой проблемой, тем более что логические операции выполняются на ПЛК очень быстро. Но на самом деле и электрическую схему, и программный код можно существенно упростить, если использовать не комбинационную логику, а *автоматный* подход.

Систему следует рассматривать как *конечный автомат*, который может находиться в различных состояниях (число которых конечно). В каждом состоянии реализуется *своя* логическая зависимость выходов от входов и *своя* логика переходов в другие состояния. При этом автомат всегда знает (помнит), в каком он находится состоянии в текущий момент времени.

В нашем случае состояний всего два: «освещение выключено» и «освещение включено». Логика управления выходом (включения ламп):

– в состоянии «освещение выключено» – лампы не включать;

– в состоянии «освещение включено» – лампы включить.

Логика переходов:

– из состояния «освещение выключено» в состояние «освещение включено» – по нажатию одной из кнопок «Включить свет», установленных около дверей, сколько бы их ни было;

– из состояния «освещение включено» в состояние «освещение выключено» – по нажатию одной из кнопок «Выключить свет», установленных около дверей, сколько бы их ни было.

Электрическая схема, воспроизводящая автомат для случая четырех дверей, показана на рис. 13.

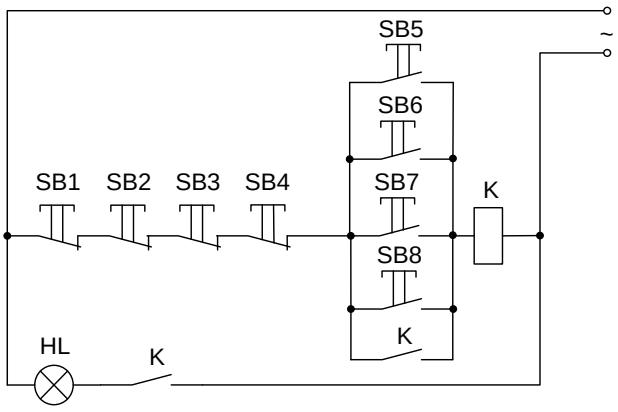


Рис. 13. Электрическая схема (автоматный подход)

Здесь применен некий коммутирующий аппарат (электромагнитное реле, магнитный пускатель), который подает напряжение на лампу и одновременно «запоминает» состояние схемы. SB1…SB4 – кнопки «Выключить свет», SB5…SB8 – кнопки «Включить свет». Все кнопки – без фиксации. «Запоминание» состояния производится с помощью дополнительного контакта аппарата K, шунтирующего кнопки SB5…SB8.

Очевидно, что эту схему можно «масштабировать» на любое количество дверей, просто добавляя кнопки.

Программная реализация системы включает список глобальных переменных (рис. 14), окно визуализации (рис. 15) и два варианта программы управления: на языке ST (рис. 16) и на языке LD (рис. 17), один из которых вызывается из PLC_PRG.

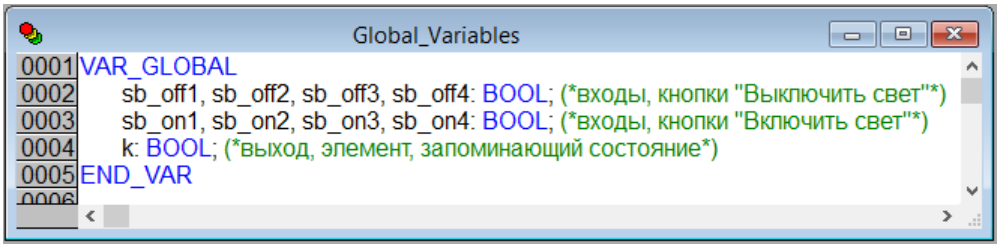


Рис. 14. Глобальные переменные

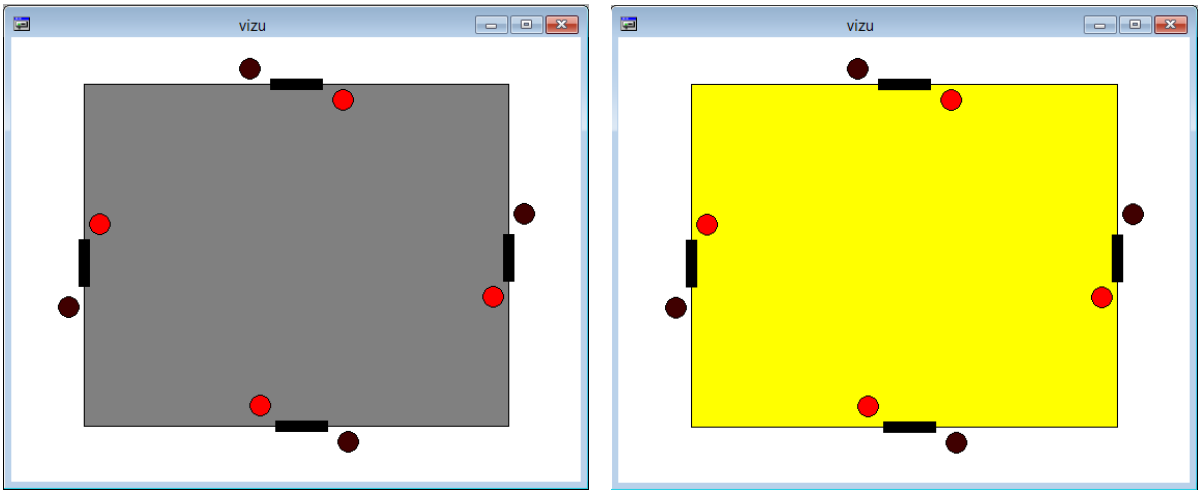


Рис. 15. Окно визуализации в работе

На окне визуализации около каждой двери размещены «кнопки». Черная кнопка предназначена для включения освещения, красная – для отключения.

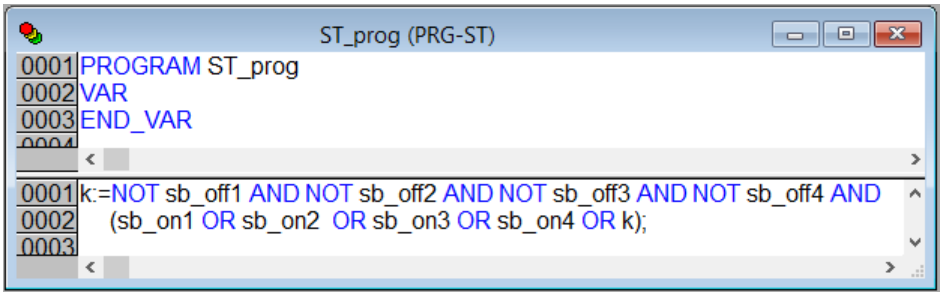


Рис. 16. Программа управления на языке ST

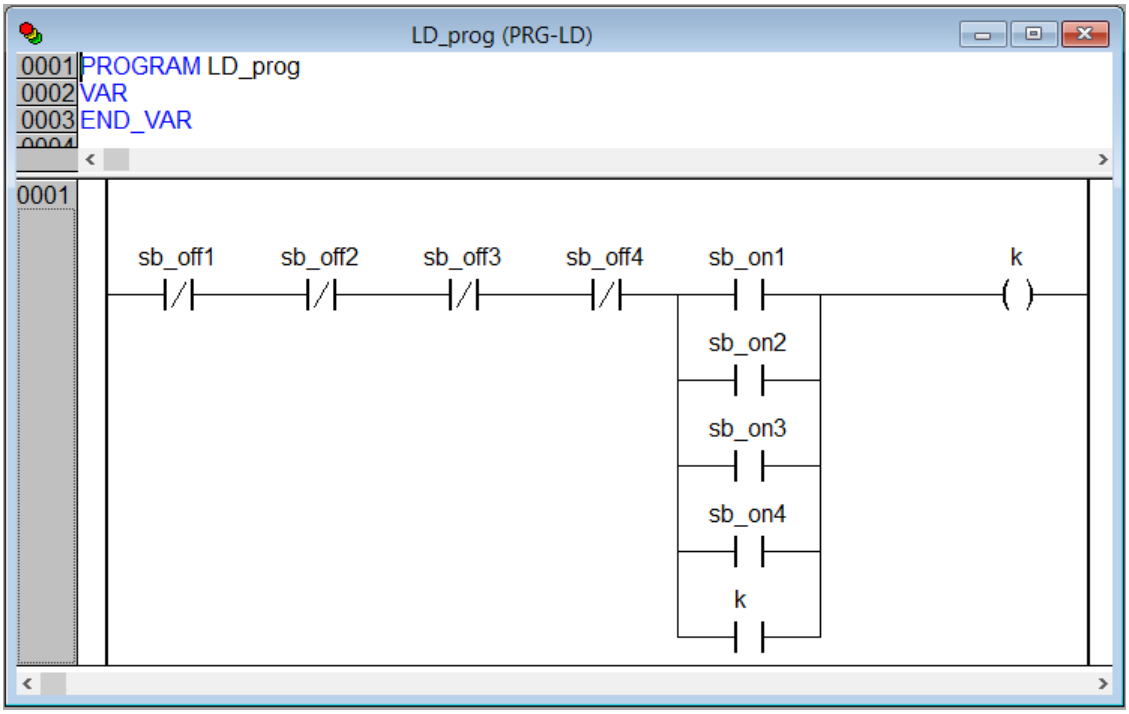


Рис. 17. Программа управления на языке LD

1.3 Управление смесительной установкой

Смесительная установка (рис. 18) предназначена для смешивания двух жидких ингредиентов. Управление подачей ингредиентов в емкость производится электромагнитными клапанами Y1 и Y2. Управление сливом смеси – электромагнитным клапаном Y3.

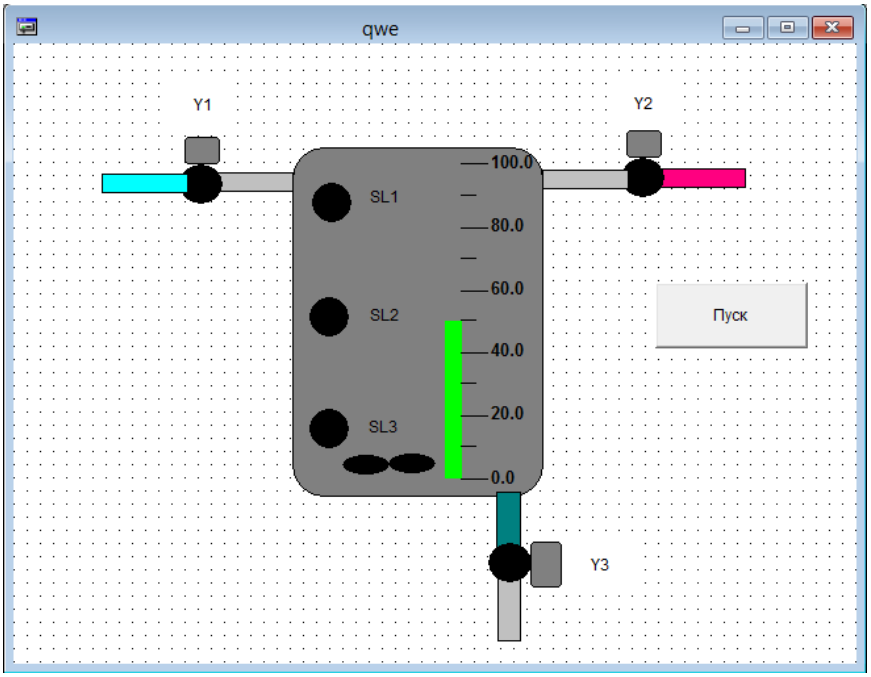


Рис. 18. Смесительная установка (окно визуализации)

Уровень в емкости контролируется тремя датчиками SL1, SL2, SL3 дискретного типа. В окне визуализации имеется также «измеритель» непре-

рывного типа, однако он – всего лишь инструмент моделирования и отладки, в реальной системе его нет.

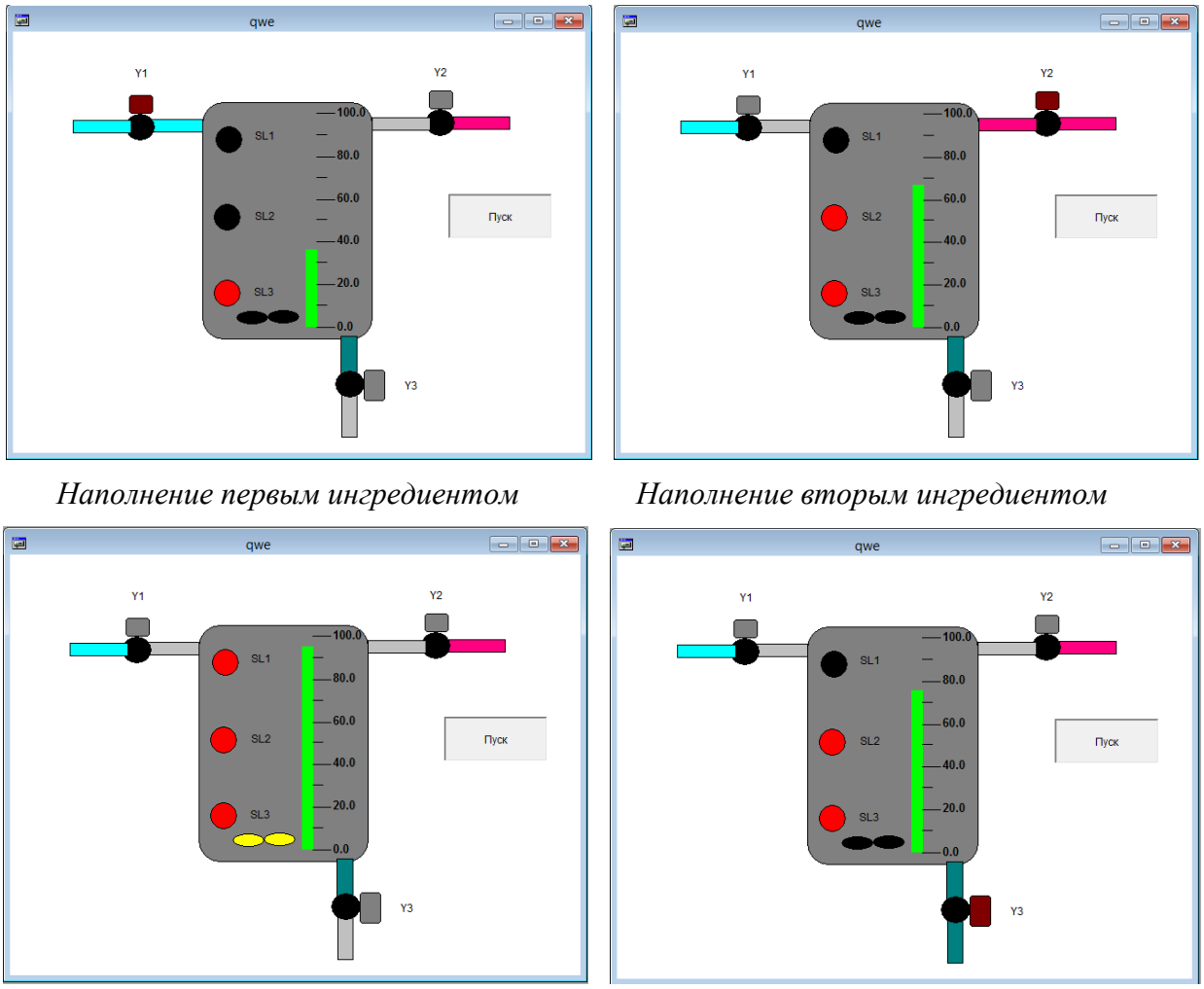
На рис. 19 показаны стадии процесса приготовления смеси:

1) подача первого ингредиента (заканчивается при срабатывании датчика SL2);

2) подача второго ингредиента (заканчивается при срабатывании датчика SL1);

3) перемешивание (мешалка работает в течение заданного времени);

4) слив смеси (заканчивается при отключении датчика SL3).



Наполнение первым ингредиентом *Наполнение вторым ингредиентом*

Перемешивание *Слив*

Рис. 19. Стадии процесса

Глобальные переменные проекта представлены на рис. 20.

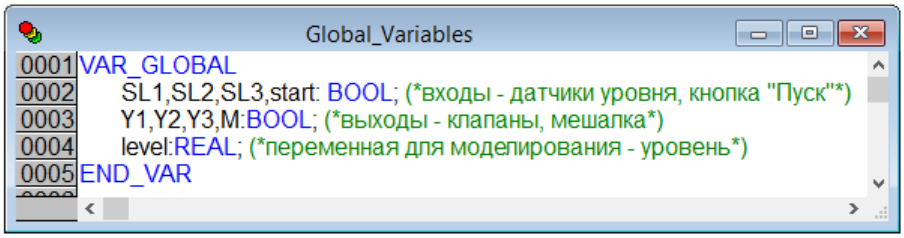


Рис. 20. Глобальные переменные

«Модель» объекта управления является частью программы PLC_PRG, рис. 21.

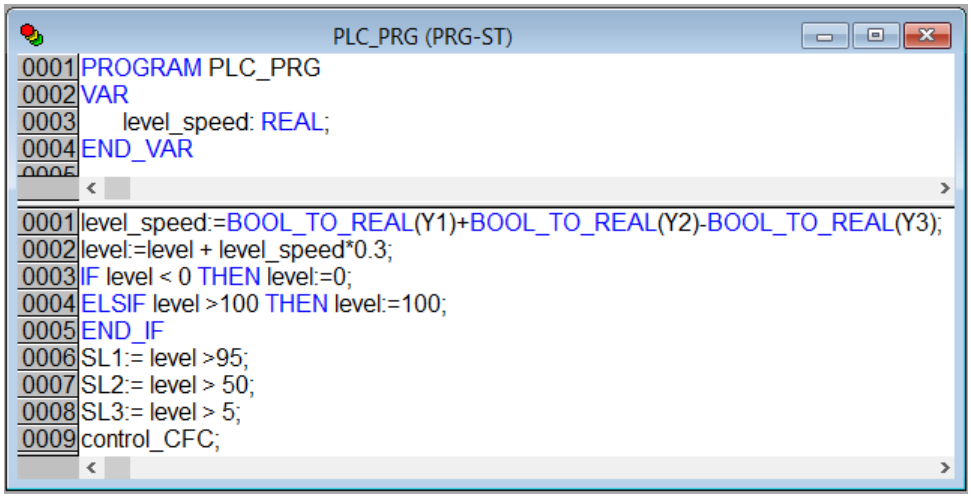


Рис. 21. Стадии процесса

В локальную переменную `level_speed` заносится результат вычисления «скорости» изменения уровня как функции состояния клапанов. Возможные значения: –1, 0, 1, 2. При правильно работающей программе управления значение 2 (одновременно открыты Y1 и Y2) появляться не должно. Глобальная переменная `level` «обновляется», если `level_speed` не равно нулю. Коэффициент 0.3 подобран «экспериментально», так, чтобы уровень изменялся не слишком быстро, но и не слишком медленно. Условными операторами диапазон изменения переменной `level` ограничен предельными значениями 0 и 100. Датчики SL1, SL2 и SL3 «срабатывают» при достижении уровнем значений 95, 50 и 5 соответственно.

Помимо «моделирования» объекта, в PLC_PRG вызывается одна из четырех программ управления, написанных на разных языках.

Программа управления, на каком бы языке она не была составлена, строится как конечный автомат. В первом приближении в качестве состояний автомата можно принять стадии процесса, показанные на рис. 19, добавив еще одно состояние – состояние паузы или ожидания.

На языке ST для построения автомата очень удобно использовать управляющую конструкцию CASE, в которой в зависимости от значения некоторой целочисленной переменной («номера» состояния автомата) выбирается на исполнение тот или иной фрагмент кода. В этих фрагментах находятся логика управления выходами и логика переходов в другие состояния. При таком подходе программа легко пишется и читается. Именно так и реализована ST-версия программы управления, рис. 22.

Для «нумерации» состояний используется локальная переменная `status`. Кроме нее задействуется также локальный экземпляр библиотечного функционального блока `TON` (таймер с задержкой включения) `timer`. Он предназначен для отчета времени в состоянии 3 – «перемешивание».

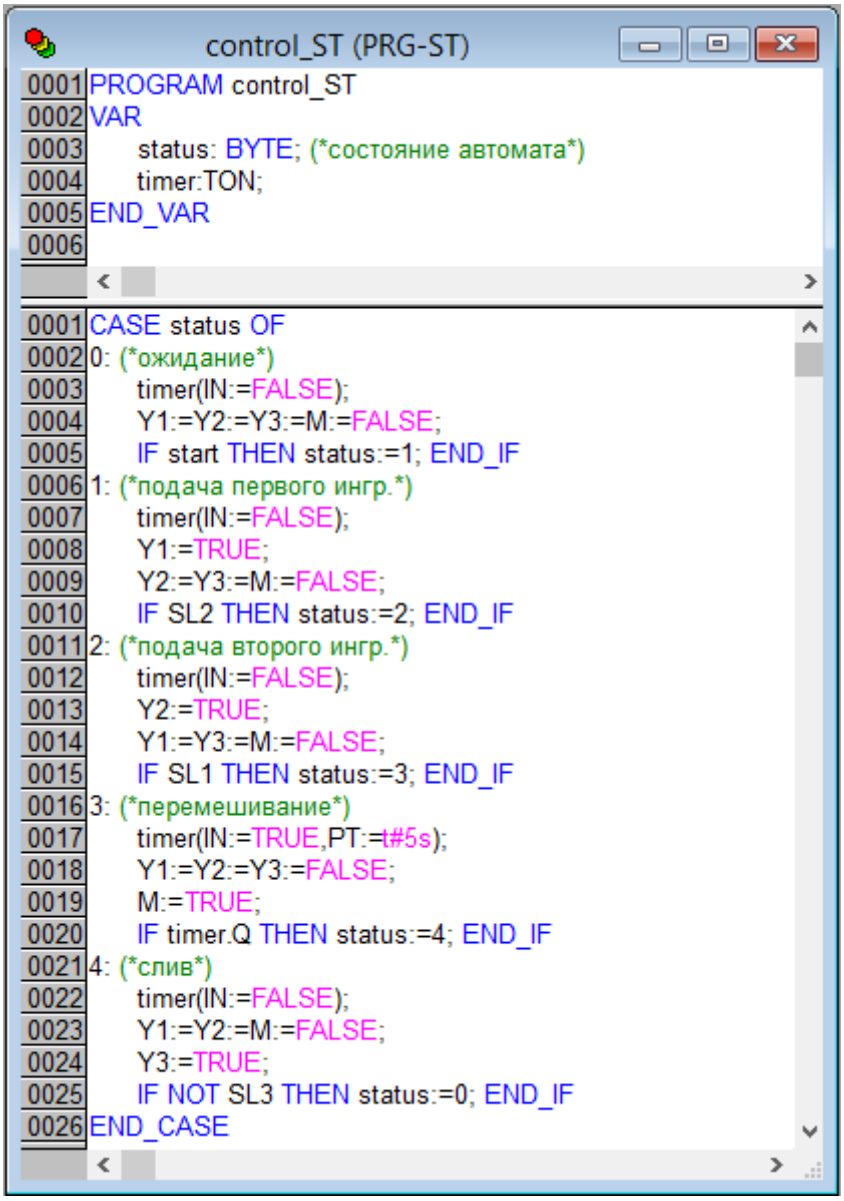


Рис. 22. Программа управления на языке ST

Программа демонстрирует «правило», согласно которому в каждом состоянии производится управление всеми выходами. Это может казаться избыточным, и часто так оно и есть, но зато позволяет исключить трудно обнаруживаемые ошибки, когда выходы, включенные в одном состоянии, забыли выключить в другом. То же самое касается и таймера, поскольку таймер для автомата – внешняя сущность, такая же, как объект управления. На неявный «вход» автомата поступает сигнал о срабатывании таймера (`timer.Q`). Через неявный «выход» автомат управляет таймером, запуская его или сбрасывая (`timer(…)`). Если забыть сбросить таймер перед его повторным запуском, выдержка времени не состоится.

Сам автомат с пятью состояниями, очевидно, является избыточным. Состояния 1 и 2 легко объединяются, поскольку управление клапанами Y1 и Y2 в них можно «поручить» датчику SL2, как это и сделано в управляющих программах на других языках. В результате объединения получилось состоя-

ние 12 – «наполнение». На рис. 23 показана программа управления на языке SFC.

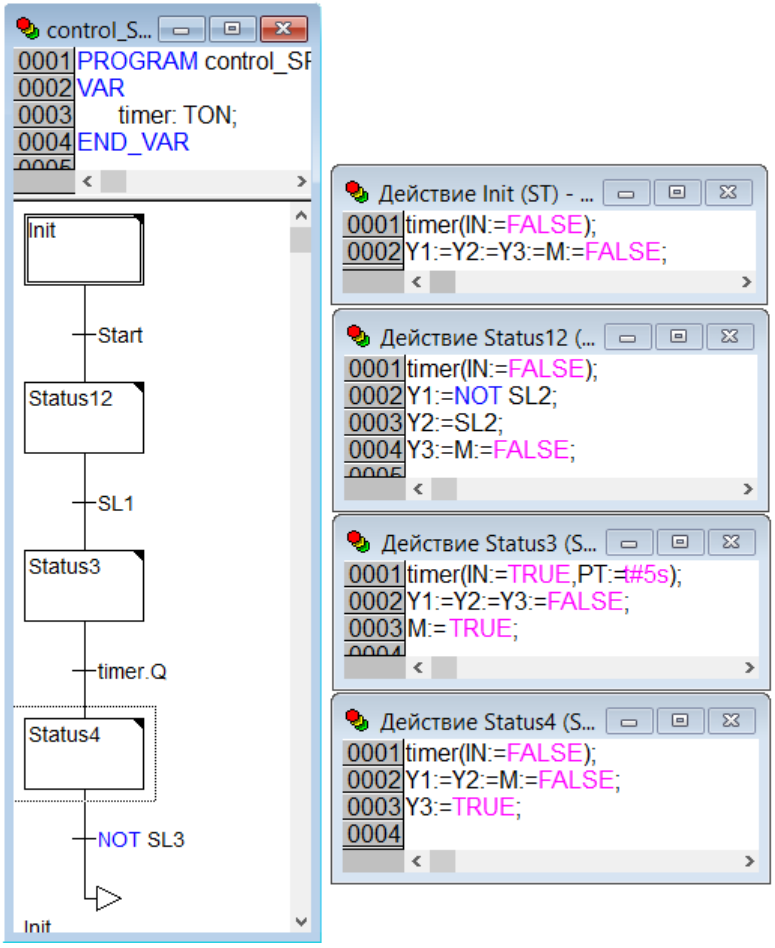


Рис. 23. Программа управления на языке SFC

Программа на SFC состоит из программируемых на других языках шагов (действий) и переходов. Переходы могут представлять обычные логические выражения, как в нашем случае, но могут быть и программами, результатом выполнения которых является логическое значение `FALSE` или `TRUE`. В любом случае переход «активируется», если его значение `TRUE`. Далее активируется шаг, к которому ведет переход, и начинают выполняться его операции (операции предыдущего шага выполняться перестают).

На самом деле язык SFC намного сложнее, чем описано выше. В частности, допускаются: ветвление, множество активных шагов, вызов дочерних подпрограмм, различные виды операций внутри шагов и многое другое. Но в нашем случае можно ограничиться упрощенным пониманием, согласно которому каждый шаг – это определенное состояние конечного автомата. Поэтому запоминать «номер» состояния не требуется, – программа это делает «сама».

Программа управления на языке LD показана на рис. 24. Здесь информацию о состоянии автомата удобнее сохранять в битовых переменных, чтобы можно было их использовать в виде «контактов». При этом необходимо гарантировать сброс одной битовой переменной при установке другой.

В цепях 1 – 4 представлена логика переходов, в остальных – логика управления выходами (включая управление таймером).

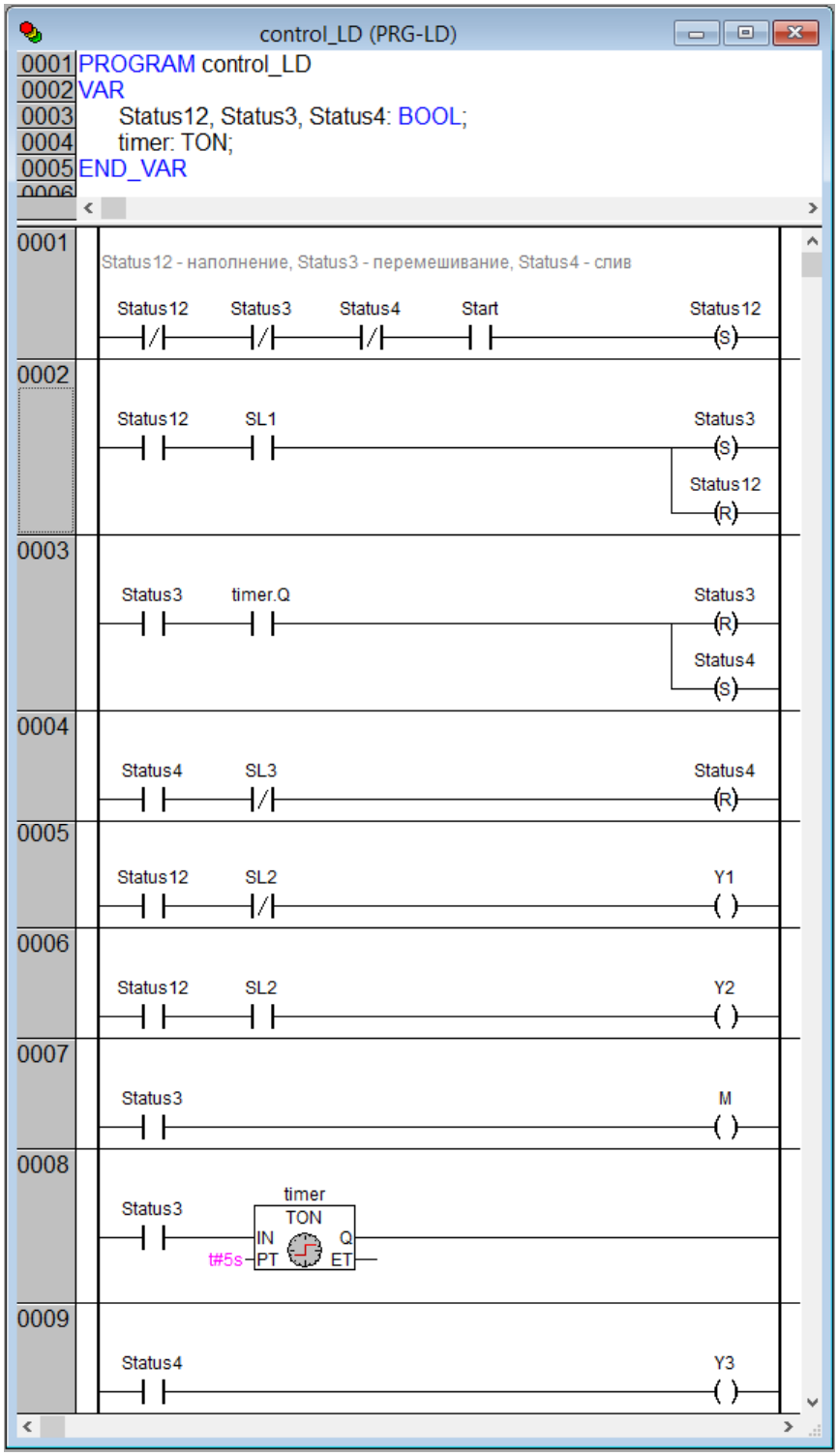


Рис. 24. Программа управления на языке LD

На рис. 25 приведена программа на языке CFC. В этой программе состояние автомата сохраняется в целочисленной переменной и «распознается» блоками сравнения (на равенство) EQ. Выходные сигналы блоков являются «опорными» для формирования как логики управления выходами, так и логики переходов (используется инструкция MOVE).

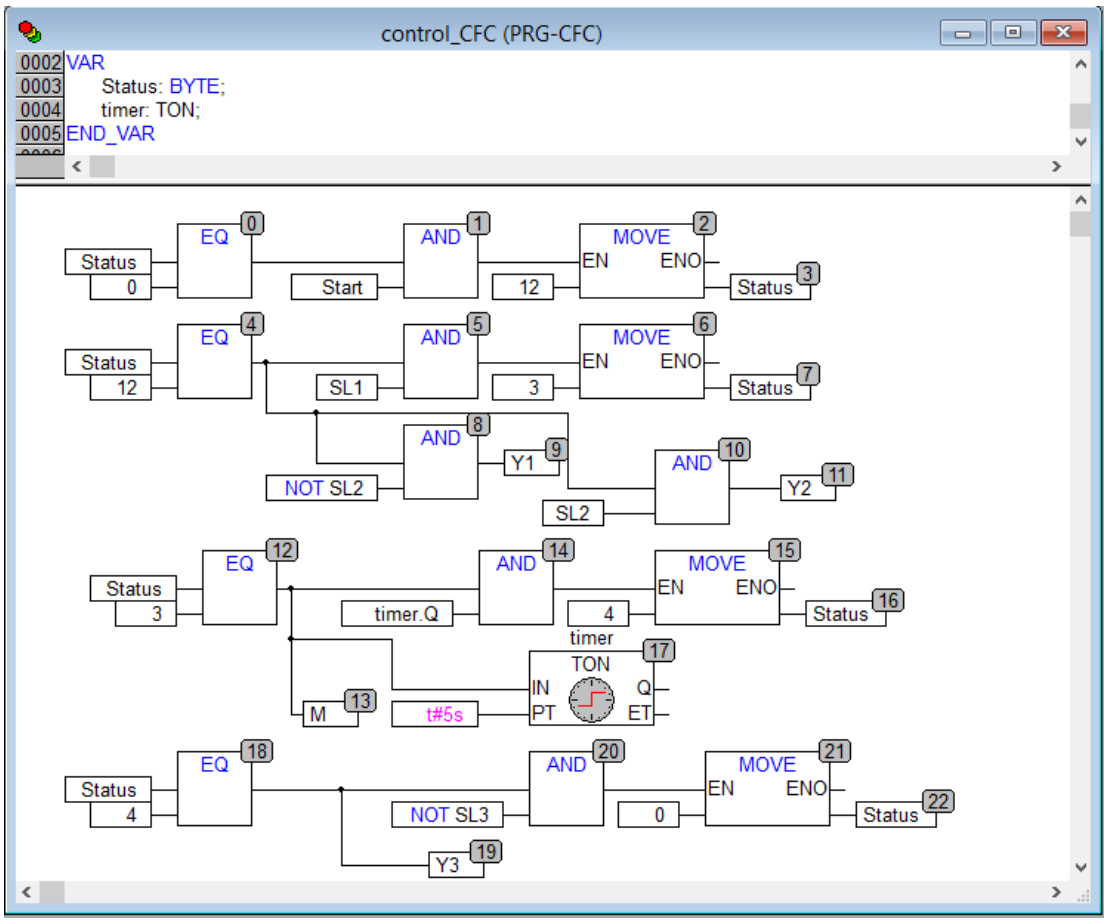


Рис. 25. Программа управления на языке CFC

В заключение отметим, что во всех приведенных программах реализован только «базовый» алгоритм управления, причем в неоптимизированном виде. В [2] приведены решения по реализации дополнительных функций (обнаружение отказов аппаратуры) и оптимизации алгоритма (сведение числа состояний автомата к двум).

1.4 Светофор для пешеходного перехода

В окне визуализации, показанном на рис. 26, размещены схематичные изображения светофоров для автомобилей и пешеходов и «пульта управления» для пешеходов (по одному экземпляру).

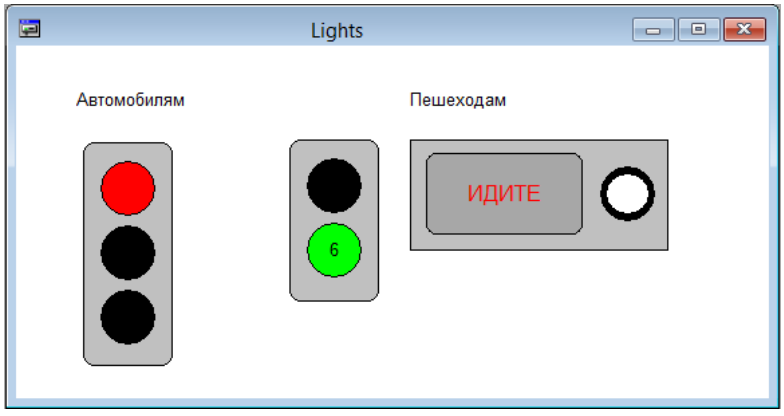


Рис. 26. Окно визуализации в действии

Зеленая лампа светофора для пешеходов ведет обратный отчет времени. На «пульте» располагаются кнопка и табло, выводящее в активном состоянии сообщения «ЖДИТЕ» и «ИДИТЕ».

Список глобальных переменных показан на рис. 27.

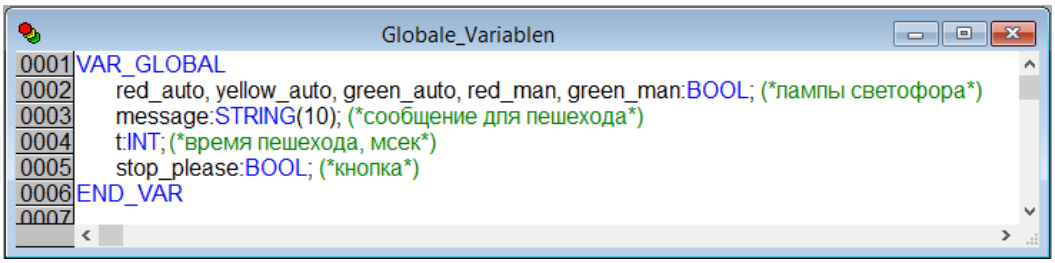


Рис. 27. Список глобальных переменных

Переменная `t` отсчитывает время в миллисекундах с момента загорания зеленого сигнала для пешеходов. Элемент визуализации (кружок) ведет обратный отчет в секундах, вычисляя значение выражения, показанного на рис. 28.

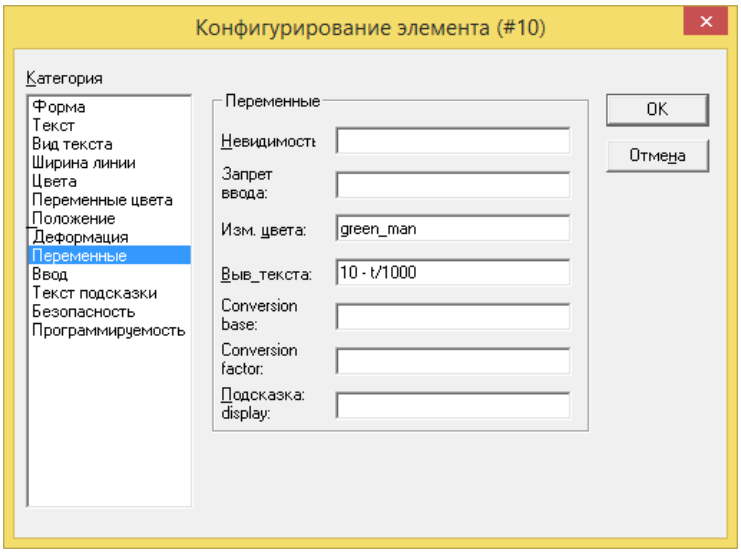


Рис. 28. Настройка элемента визуализации

Система может находиться в двух состояниях, представленных на рис. 29.

В состоянии 1, условно названном «переходом улицы пешеходами», помимо собственно перехода, производятся также необходимые действия, предшествующие ему, и последующие за ним. Длительность этого состояния фиксирована и равна 16 с. Все операции по управлению лампами светофоров и табло происходят строго по времени.

В состоянии 2, названном «движением автомобилей», действительно происходит только движение автомобилей: на светофоре для них горит зеленый свет, а на пешеходном светофоре – красный. Длительность этого состояния – не менее 10 с.

Переход из состояния 1 в состояние 2 происходит «автоматически» по истечению 16 секунд пребывания в состоянии 1. Переход из состояния 2 в

состояние 1 происходит по нажатию пешеходом кнопки, но не ранее 10 секунд пребывания в состоянии 2. Если пешеход нажал кнопку раньше, факт нажатия фиксируется системой (хотя сама кнопка – без фиксации), на табло выводится сообщение «ЖДИТЕ», но сам переход откладывается.

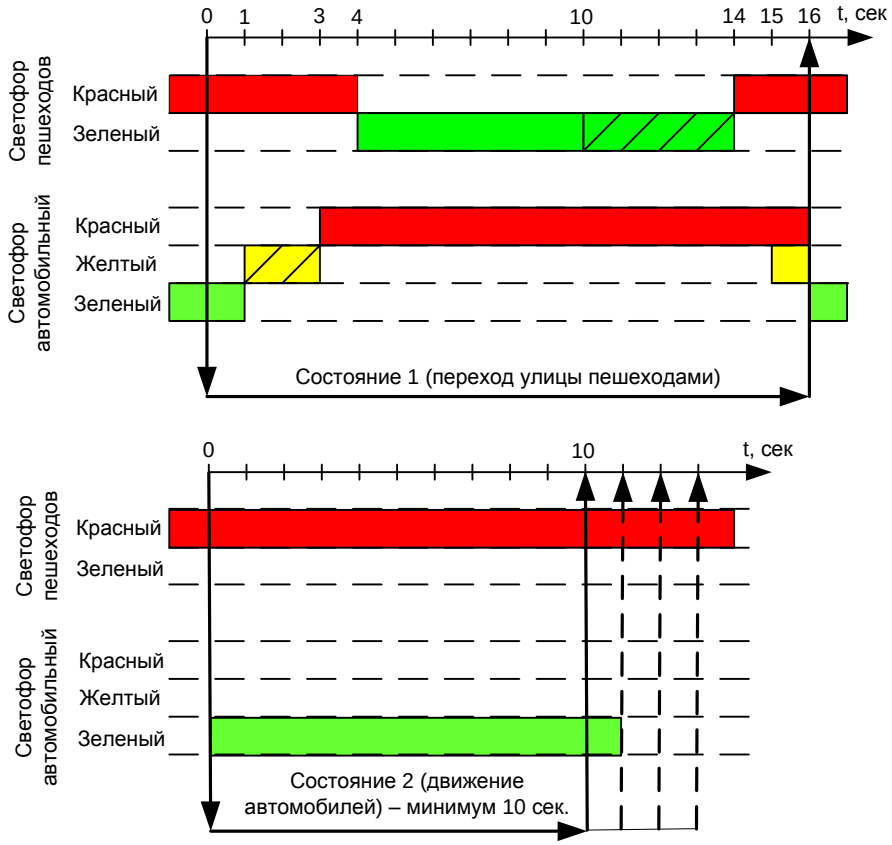


Рис. 29. Диаграммы работы системы

Заштрихованные области диаграмм означают мигание соответствующих ламп светофоров.

Программа управления на языке ST показана на рис. 30.

Поскольку автомат может находиться только в двух состояниях, для запоминания текущего состояния достаточно переменной типа BOOL, в нашем случае – это переменная `man`. Если `man=TRUE`, система находится в состоянии 1, иначе – в состоянии 2. Переменная `wait` фиксирует запрос пешехода на переход улицы. Таймер `timer` является основным средством организации работы по времени в обоих состояниях. В состоянии 1 его «уставка» равна 16 с, в состоянии 2 – 10 с. При переходе из одного состояния в другое таймер сбрасывается. Таймер `mig` используется для формирования мигающих сигналов на соответствующих временных участках диаграммы работы в состоянии 1. Его уставка задает период мигания ламп равным 800 мс, первые 400 мс из которых лампа светится. Для непрерывного контроля времени, отсчитываемого таймерами, задействуются значения на их выходах `ET` (типа `TIME`).

Программа управления на языке LD приведена на рис. 31-34.

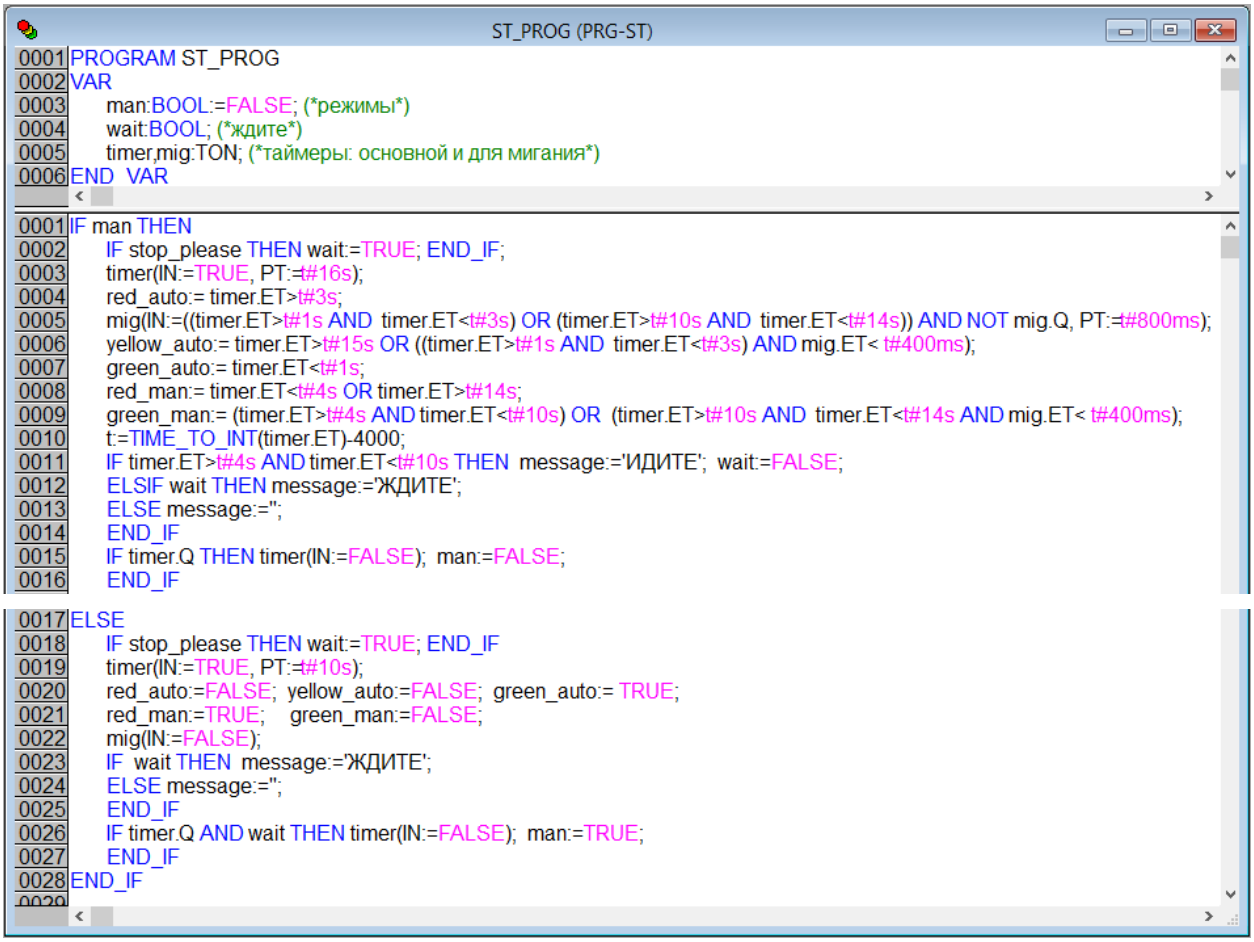


Рис. 30. Программа управления на языке ST

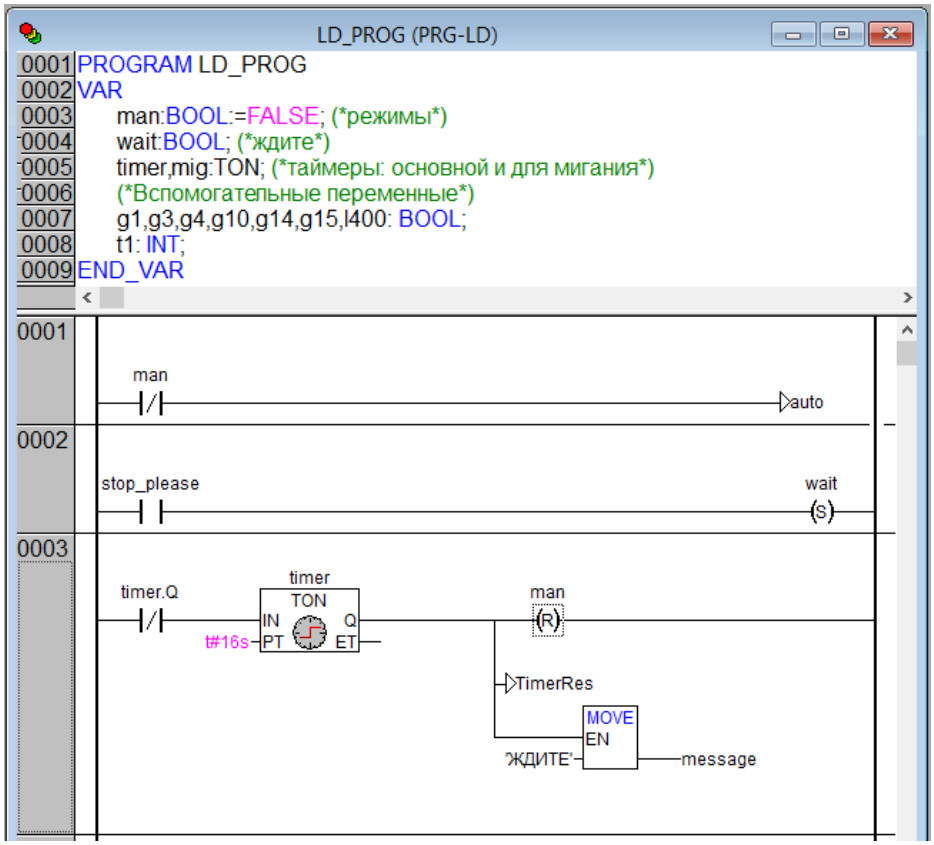


Рис. 31. Программа управления на языке LD, фрагмент 1

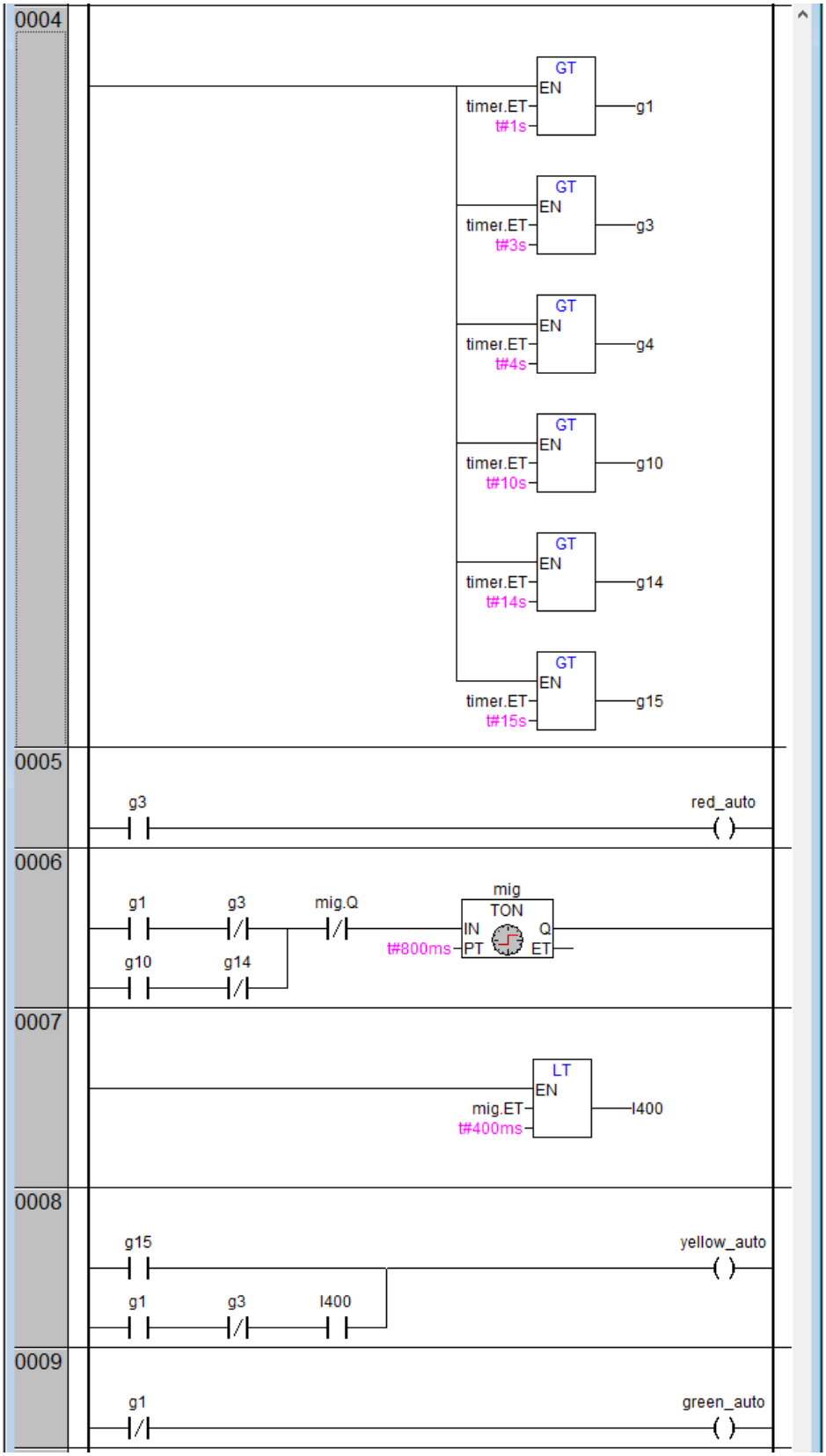


Рис. 32. Программа управления на языке LD, фрагмент 2

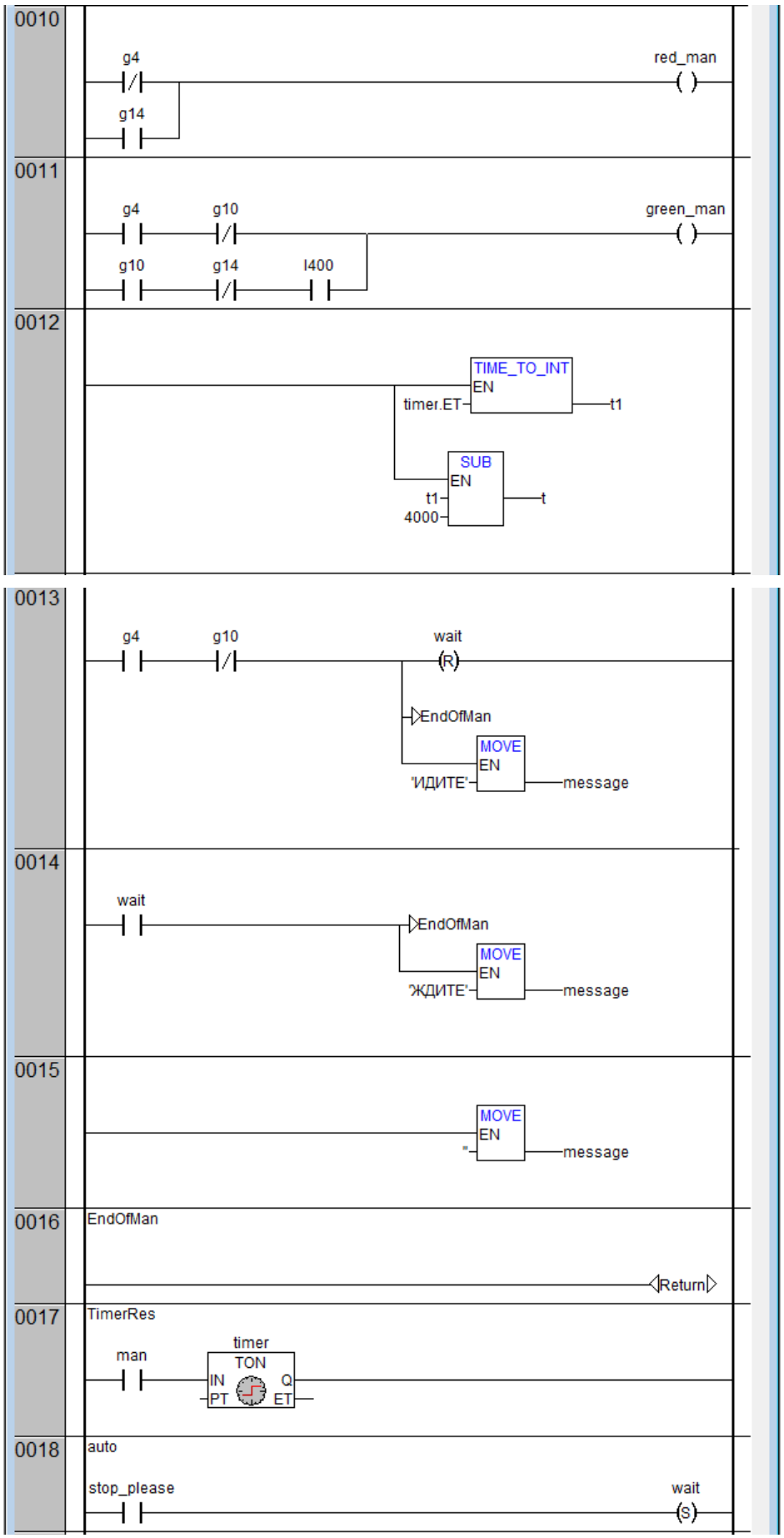


Рис. 33. Программа управления на языке LD, фрагмент 3

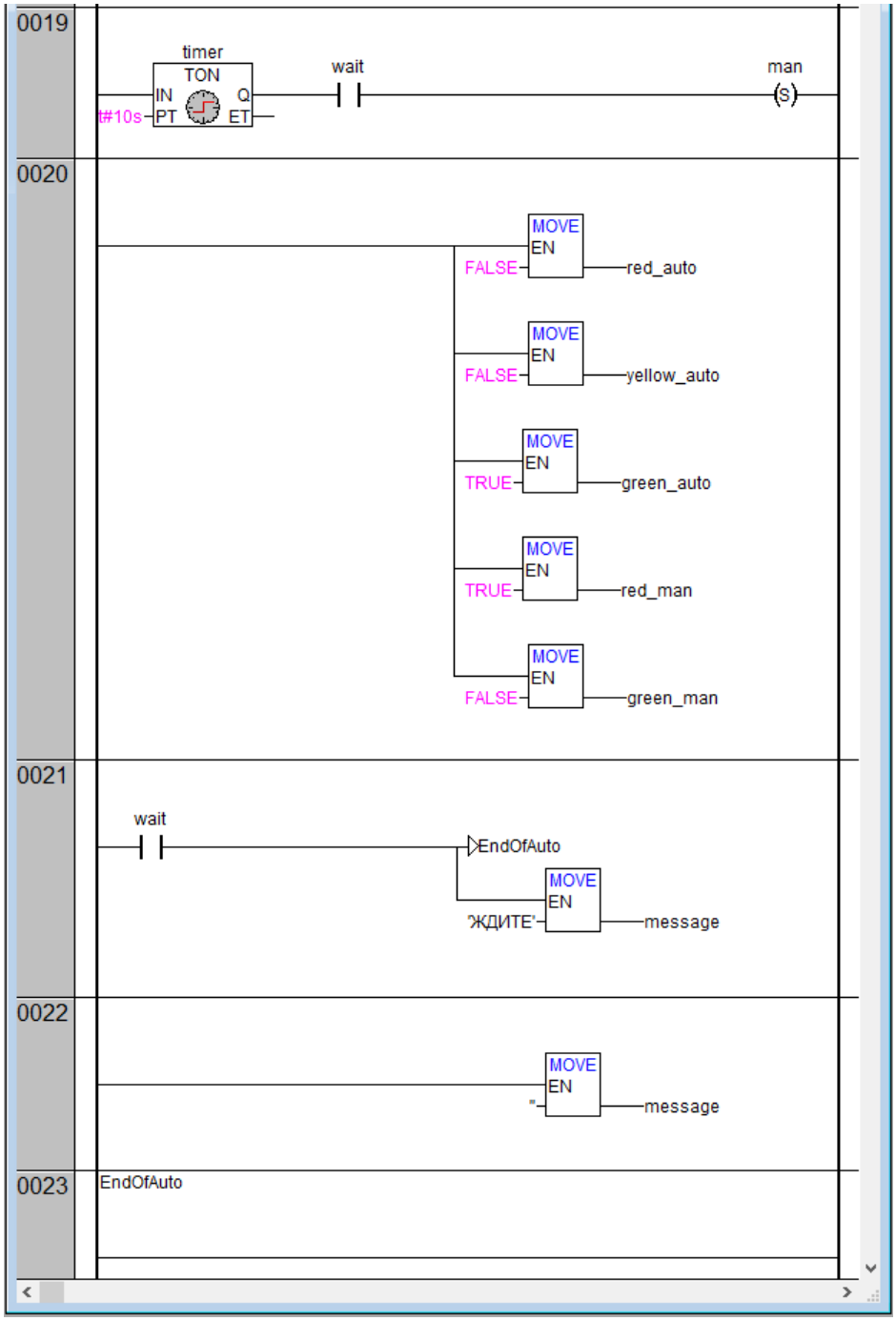


Рис. 34. Программа управления на языке LD, фрагмент 4

Программа в целом построена аналогично рассмотренной выше программе на языке ST, однако некоторые аспекты имеют свою специфику.

Условные операторы реализованы с помощью переходов (на метки) и оператора `Return`, осуществляющего выход из программы. Переходы позволяют исключить из выполнения ненужные в данный момент цепи.

Вся программа состоит из двух больших фрагментов. Цепи 2-16 выполняются в состоянии 1 (`man=TRUE`), цепи 18-23 – в состоянии 2 (`man=FALSE`). В первой цепи производится «выбор» фрагмента. Цепь 17 выполняется только при смене состояния с первого на второе. Управление в нее по-

падает по переходу из цепи 3 при срабатывании таймера. Назначение цепи 17 – сброс таймера `timer` с целью подготовки его к использованию в новом состоянии. Так как на момент перехода переменная `man` сброшена (еще в цепи 3), на вход `IN` таймера подается `FALSE` , что и приводит к его сбросу. После этого управление «своим ходом» попадает в первую цепь второго фрагмента. При смене состояния со второго на первое сброс таймера производится в цепи 3 собственным выходным сигналом `Q`, инверсия которого подается на вход `IN`. Отличия в порядке сброса таймера при различных переходах между состояниями связано со следующим. В состоянии 1 система находится фиксированное время, и выход таймера `Q` всегда равен `FALSE` (за исключением момента смены состояний). В состоянии 2 система может находиться сколь угодно большое время и после срабатывания таймера, поэтому он не может быть «самосбрасываемым».

Операции сравнения (>,<), которые в ST-программе являются составными частями выражений для управления лампами светофоров и запуска дополнительного таймера, в программе на языке LD реализуются отдельно блоками `GT` (greater than) и `LT` (less than). Результаты сравнения записываются в «промежуточные» битовые переменные, которые далее используются уже как контакты.

Операция присваивания переменной значения константы реализовано с помощью блока `MOVE`.

2.1 Релейная система регулирования. Программная реализация ШИМ

Структура системы управления в упрощенном виде показана на рис. 35. Система позволяет осуществлять:

– прямое ручное управление нагревательным элементом путем непосредственного включения/отключения;

– квазинепрерывное ручное управление нагревательным элементом посредством широтно-импульсной модуляции с возможностью плавного изменения скважности импульсов (скважность – это отношение ширины импульсов к периоду их следования);

– автоматическое регулирование температуры по релейному двухпозиционному закону.

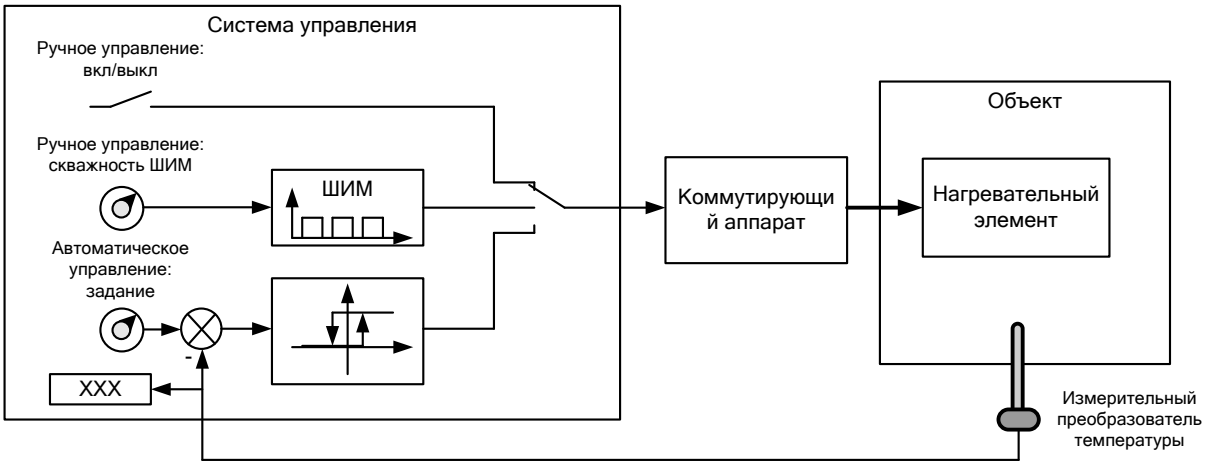


Рис. 35. Структура системы

Панель управления (экран визуализации) в действии показана на рис. 36. Список глобальных переменных приведен на рис. 37.

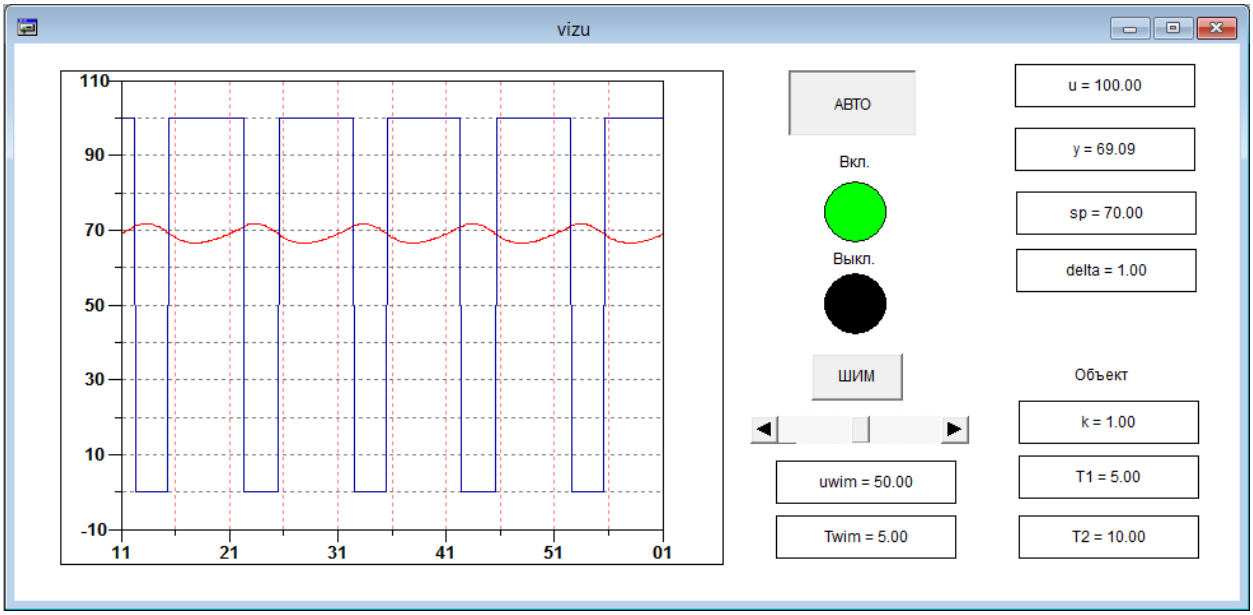


Рис. 36. Панель управления

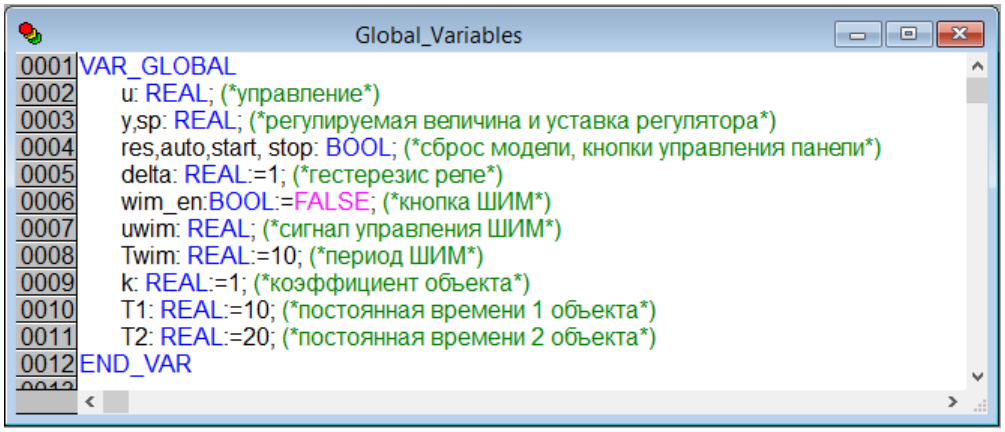


Рис. 37. Глобальные переменные

Объект управления описывается передаточной функцией второго порядка:

$$W_{об}(p)=\frac{k}{T_1T_2p^2+(T_1+T_2)p+1}=\frac{k}{T_1p+1}\times\frac{1}{T_2p+1}.$$

Коэффициент передачи и постоянные времени объекта подлежат настройке в соответствующих элементах визуализации (в учебных целях). Программная модель объекта строится по схеме, показанной на рис. 38.

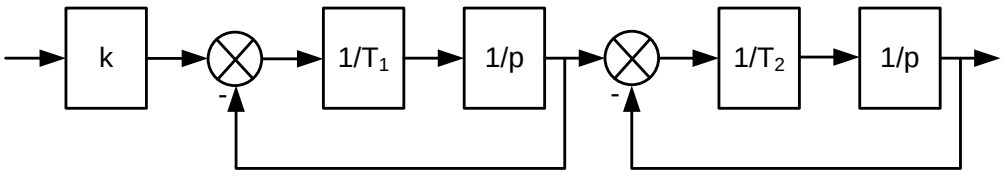


Рис. 38. Структура объекта управления

Модель оформлена в виде программы на языке CFC, рис. 39.

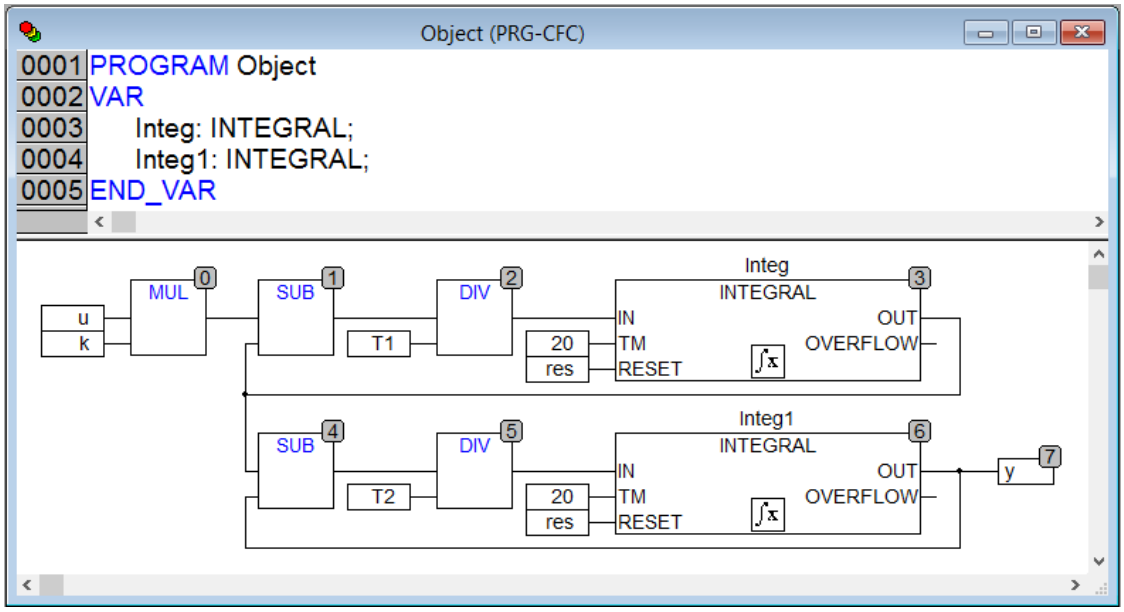


Рис. 39. Программная модель объекта управления

Основными элементами программной модели являются интеграторы – экземпляры библиотечного функционального блока `INTEGRAL` из библиотеки Util.lib. Эта библиотека должна быть подключена к проекту в Менеджере библиотек. Блок `INTEGRAL` выполняет простейший алгоритм численного интегрирования по методу трапеций и должен вызываться с определенной периодичностью, заданной на настроечном входе `TM` (в мс). Таким образом, программа `Object` должна выполняться циклически. Для того чтобы обеспечить циклический вызов, необходимо сконфигурировать задачи, как показано на рис. 40.

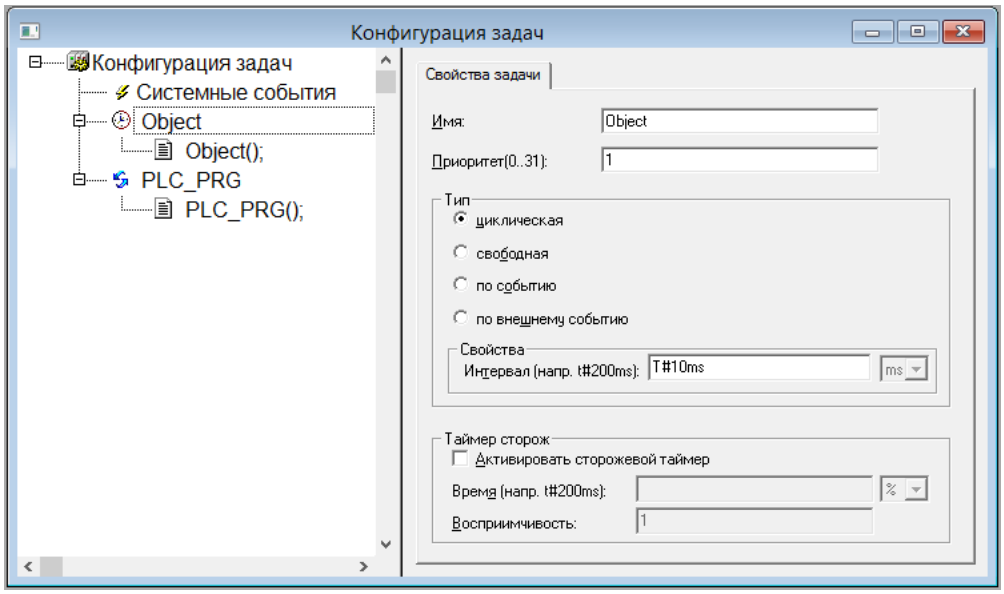


Рис. 40. Конфигурация задач

Программа `Object` включена в одноименную задачу, с интервалом вызова 10 мс. Как показывают результаты ранее проведенных экспериментов, для виртуального контроллера SP PLCWinNT интервал вызова программы, содержащей интегратор, должен быть вдвое меньше значения на его настроечном входе `TM`. (Интересно, что для реальных контроллеров интервал вызова должен быть строго равен этому значению, как это и должно быть).

Если в проекте присутствует хотя бы одна задача, «головная программа» `PLC_PRG` перестает автоматически вызываться, поэтому ее тоже нужно включить в какую-нибудь задачу. В нашем случае это задача `PLC_PRG`, вызываемая свободно.

Релейный регулятор и ШИМ-формирователь оформлены как функциональные блоки `Reg_RELE` и `WIM`. Они написаны на языке ST, рис. 41, 42.

На входы `Reg_RELE` подаются заданное и фактическое значения регулируемой величины, а также значение порога срабатывания/отпускания реле. Блок реализует функцию двухпозиционного реле с гистерезисом. Уровни выходного сигнала: 0, 100.

На входы `WIM` подаются заданная скважность (от 0 до 100%) и период следования импульсов.

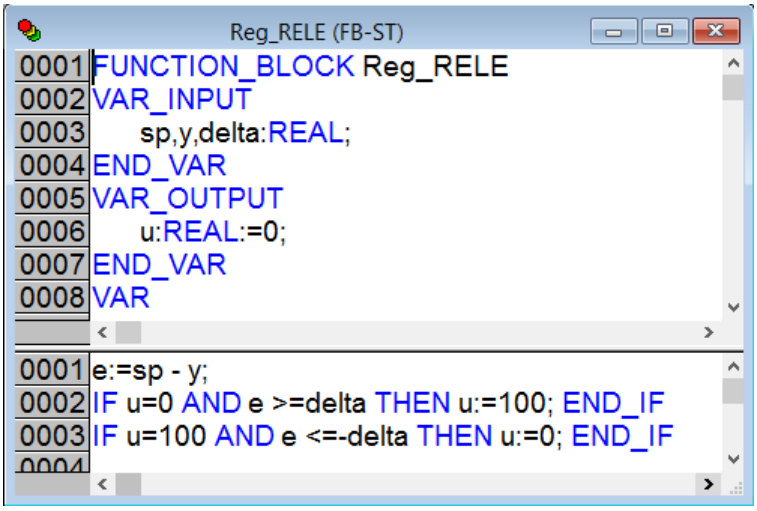


Рис. 41. Релейный регулятор

В блоке задействуется таймер, при срабатывании которого заданная скважность пересчитывается в заданную длительность импульса. Далее таймер сбрасывается и запускается снова. Фактическая длительность импульса контролируется на выходе таймера `ET` и постоянно сравнивается с заданной. Если фактическая длительность меньше заданной, выходной сигнал блока равен 100 (импульс продолжает формироваться), в противном случае – 0 (импульс прекращается).

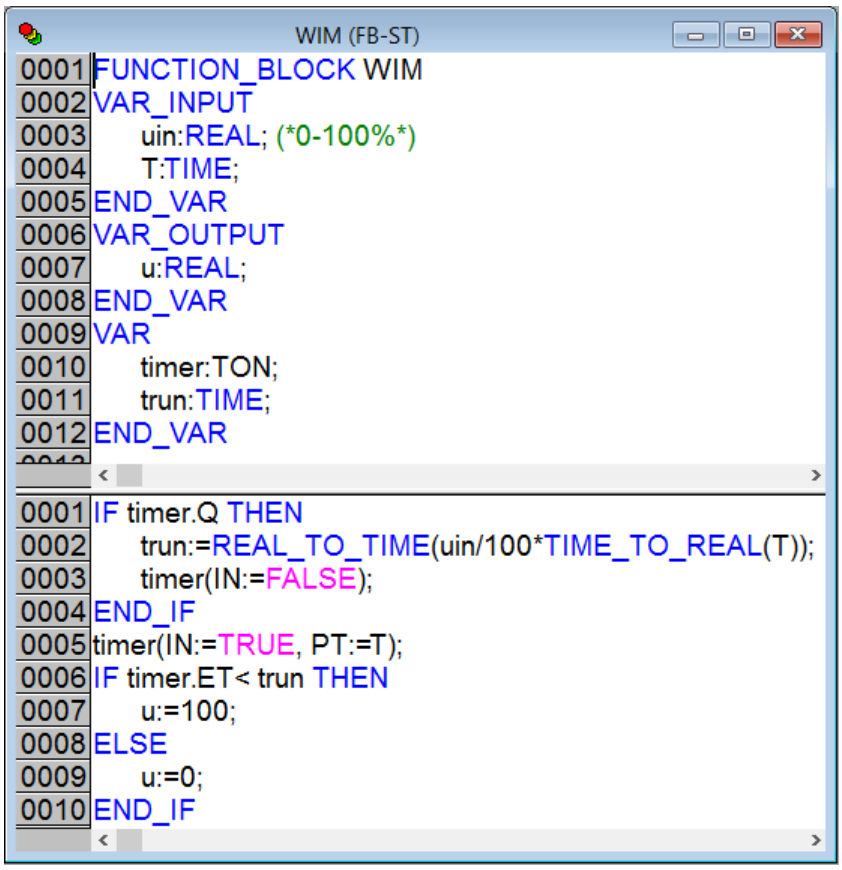


Рис. 42. Формирователь ШИМ

В программе `PLC_PRG`, показанной на рис. 43, «объединены» все части

проекта, кроме объекта управления, который функционирует самостоятельно, как отдельная задача. В программе задействуются локальные экземпляры функциональных блоков `Reg_RELE` и `WIM`, которые вызываются на исполнение в зависимости от режима. В автоматическом режиме управление формирует релейный регулятор. В ручном режиме, если нажата кнопка «ШИМ», управление формируется ШИМ-модулятором. Если кнопка не нажата, пользователь непосредственно включает/выключает «нагреватель», нажимая на круглые кнопки панели управления (см. рис. 36).

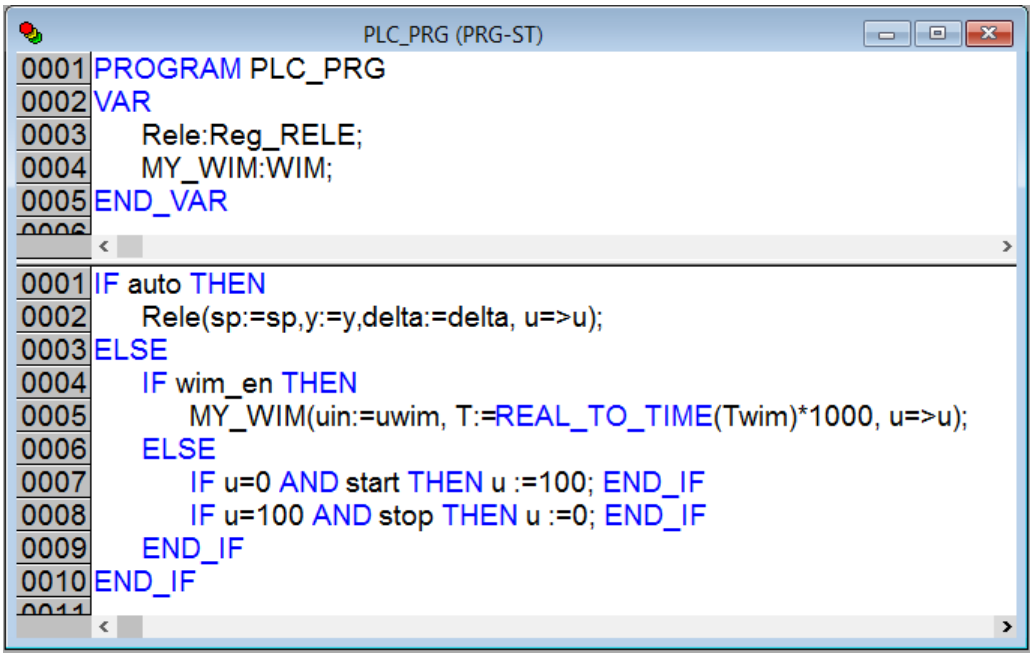


Рис. 43. Программа `PLC_PRG`

2.2 Система непрерывного регулирования (ПИ-регулятор)

Панель управления системы в действии показана на рис. 44.

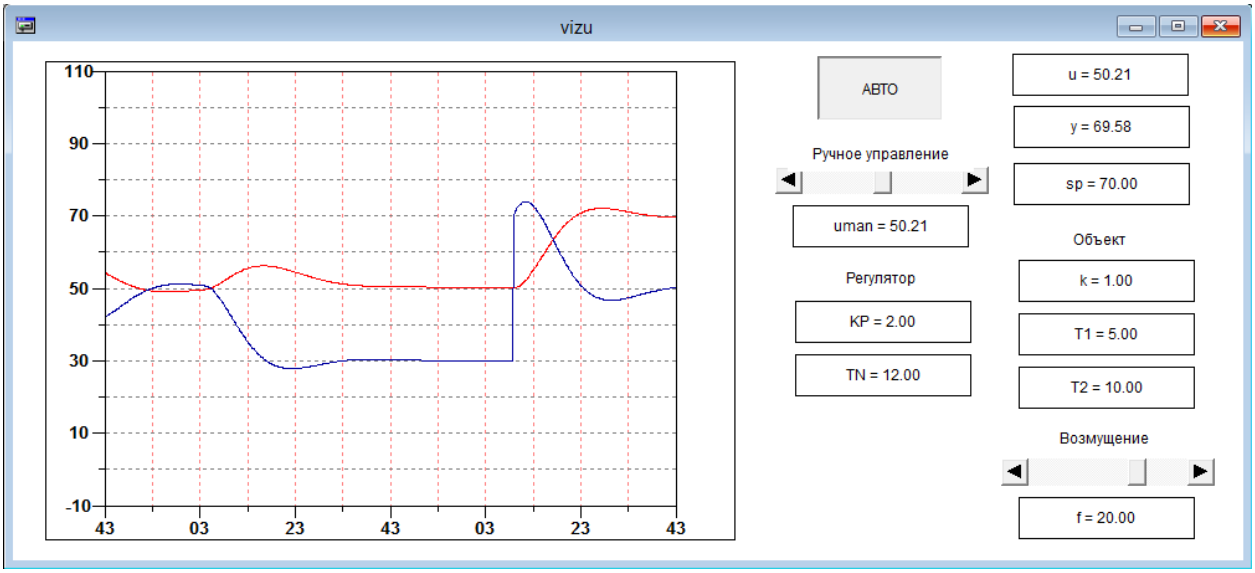


Рис. 44. Панель управления

Список глобальных переменных приведен на рис. 45.

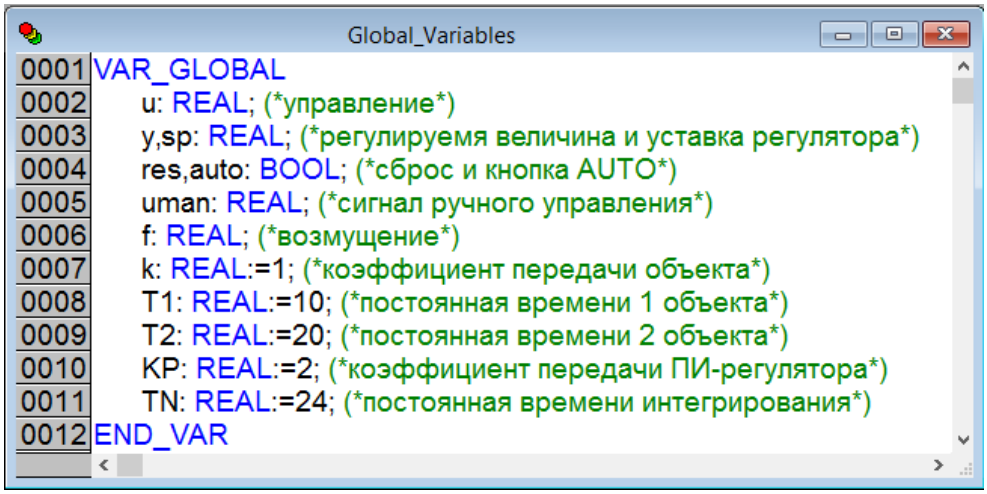


Рис. 45. Панель управления

Объект управления подвергся небольшой модификации: добавлен возмущающий сигнал, приложенный к той же самой точке, что и сигнал управления, рис. 46.

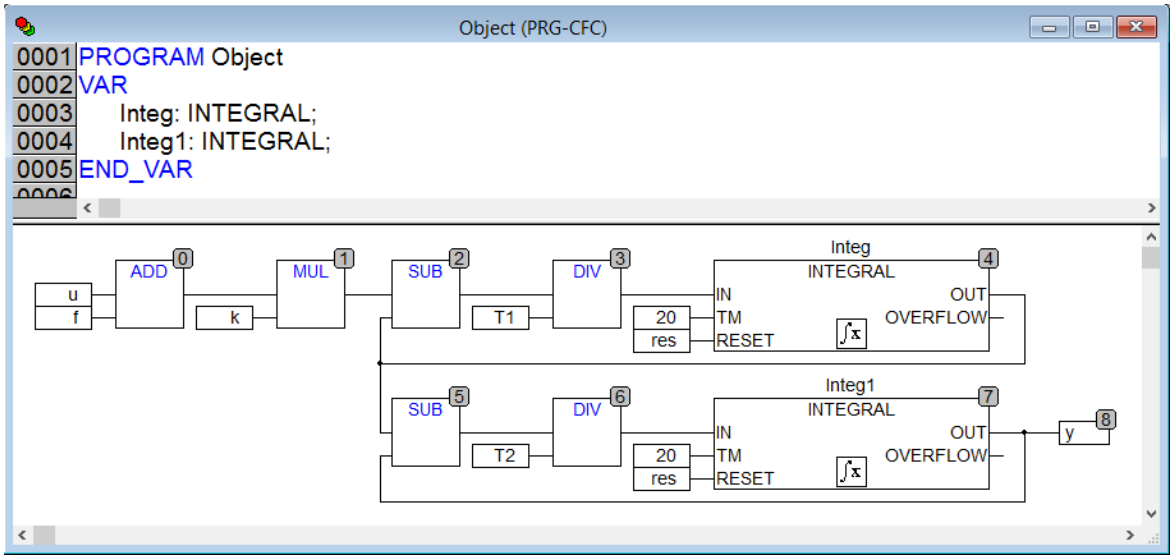


Рис. 46. Модель объекта

Как и в предыдущей работе, программа вызывается циклически, в составе *задачи*.

Программа непрерывного ПИ-регулирования, рис. 47, использует экземпляр библиотечного функционального блока `PID` из библиотеки Util.lib. Эта библиотека должна быть подключена к проекту в Менеджере библиотек. ПИ-регулятор получен из ПИД-регулятора путем обнуления постоянной времени дифференцирования `TV`. Коэффициент передачи и постоянная времени интегрирования подлежат настройке.

Регулятор переводится в режим ручного управления при подаче на его вход `MANUAL` логической единицы. В режиме ручного управления регулятор транслирует на свой основной выход сигнал с входа `Y_MANUAL`. Помимо этого «заряд» интегральной ячейки пересчитывается таким образом, чтобы буду-

щее переключение в автоматический режим происходило *безударно*, т.е. без скачкообразного изменения сигнала управления.

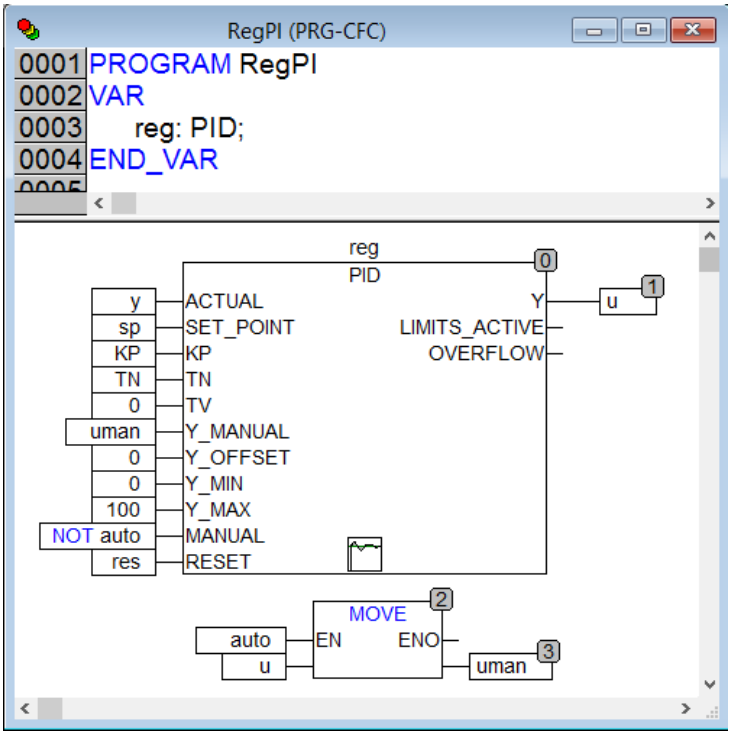


Рис. 47. Программа ПИ-регулирования

Чтобы безударно происходило также переключение на ручное управление, в программу добавлен блок `MOVE`, в автоматическом режиме «обновляющий» сигнал ручного управления.

Программа `PLC_PRG`, являющаяся «частью» свободно выполняемой задачи, в этой системе занимается исключительно вызовом программы `RegPI`.

2.3 Система импульсного регулирования (ПИ-регулятор + ШИМ)

Данная система является комбинацией предыдущей системы с технологией формирования управления по принципу ШИМ-модуляции, рассмотренной в п. 2.1. Непрерывный сигнал, формируемый ПИ-регулятором в автоматическом режиме или пользователем в ручном, подается на вход широтно-импульсного модулятора и превращается в импульсы соответствующей скважности. Такой подход на практике позволяет задействовать дешевую коммутирующую аппаратуру вместо дорогостоящей преобразовательной техники. Однако качество процесса регулирования при этом ухудшается. Для повышения качества требуется увеличение частоты модуляции, что, в свою очередь, влечет за собой ужесточение требований к коммутирующей аппаратуре.

На рис. 48 показана панель управления в работе. Список глобальных переменных, рис. 49, дополнен переменными, отвечающими за работу ШИМ.

В данном проекте используются программа `RegPI` и функциональный блок `WIM`, разработанные ранее. Они могут быть перенесены из предыдущих проектов процедурами «Экспорт» и «Импорт» (меню «Проект»).

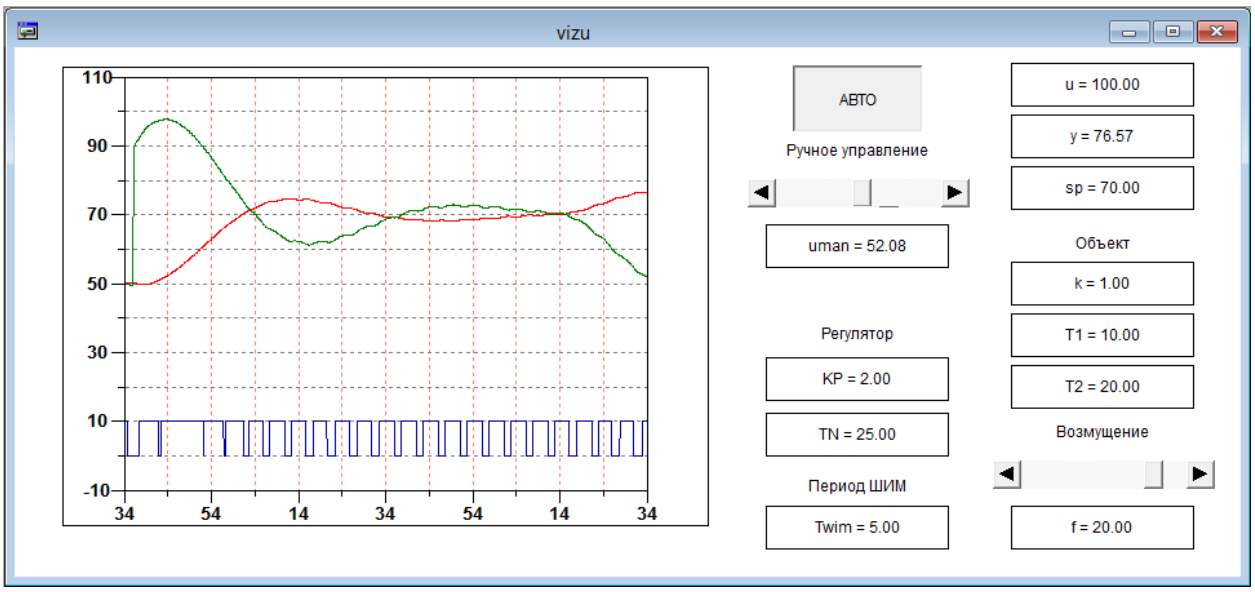


Рис. 48. Панель управления

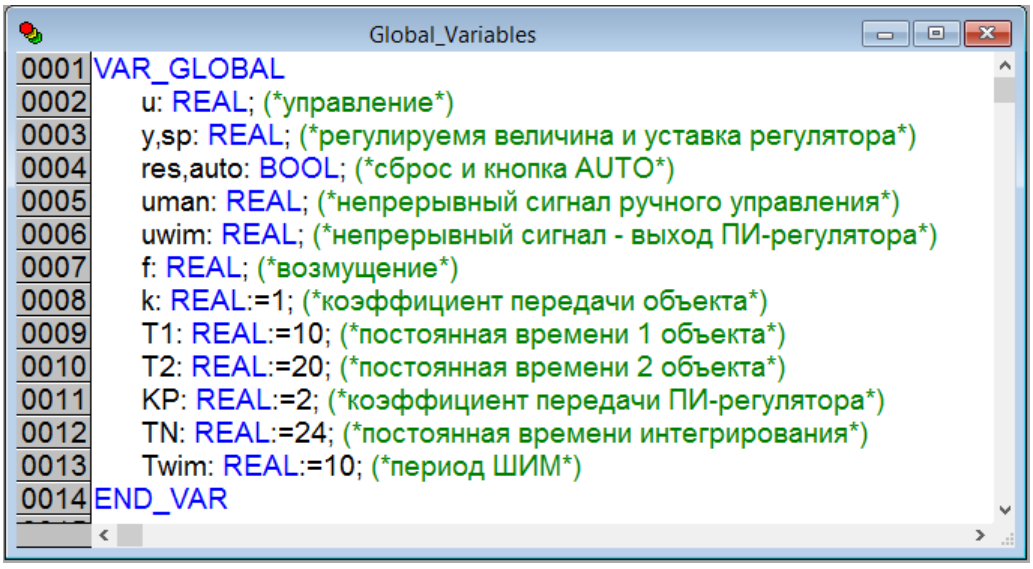


Рис. 49. Глобальные переменные

Программа регулирования (см. рис. 47) требует единственной модификации. Выходной сигнал регулятора должен быть теперь записан в переменную `uwim`. Функциональный блок `WIM` никаких изменений не требует.

В `PLC_PRG` объявляется экземпляр функционального блока `WIM`. Он вызывается на исполнение после вызова программы `RegPI`, рис. 50.

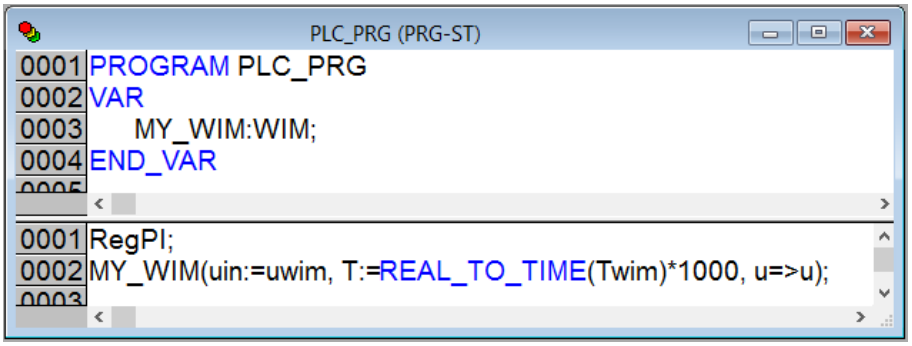


Рис. 50. Программа `PLC_PRG`

2.4 Система непрерывного регулирования (ПИ-регулятор) с компенсацией возмущений

Довольно часто основное возмущение, действующее на объект, может быть измерено. Тогда разумно завести сигнал по возмущению в систему управления и задействовать в формировании управляющего воздействия. Для этого требуется знать динамику канала «возмущение – выход». Однако даже если соответствующее описание не может быть получено аналитически, его всегда можно определить экспериментально, поскольку возмущение измеряется. В данной работе предполагается, что известны передаточные функции обоих каналов объекта: по управлению и возмущению. Тогда уравнение, описывающее объект имеет вид:

$$y = W_{ou}(p) \times u + W_{of}(p) \times f.$$

где $W_{ou}(p)$ и $W_{of}(p)$ – передаточные функции объекта по управлению и возмущению. В работе они приняты в следующем виде:

$$W_{ou}(p) = \frac{k}{T_1 p + 1} \times \frac{1}{T_2 p + 1},$$

$$W_{of}(p) = \frac{k_f}{T_{1f} p + 1} \times \frac{1}{T_{2f} p + 1}.$$

Таким образом, передаточная функция объекта по возмущению имеет ту же структуру, что и передаточная функция по управлению.

На рис. 51 приведен список глобальных переменных проекта.

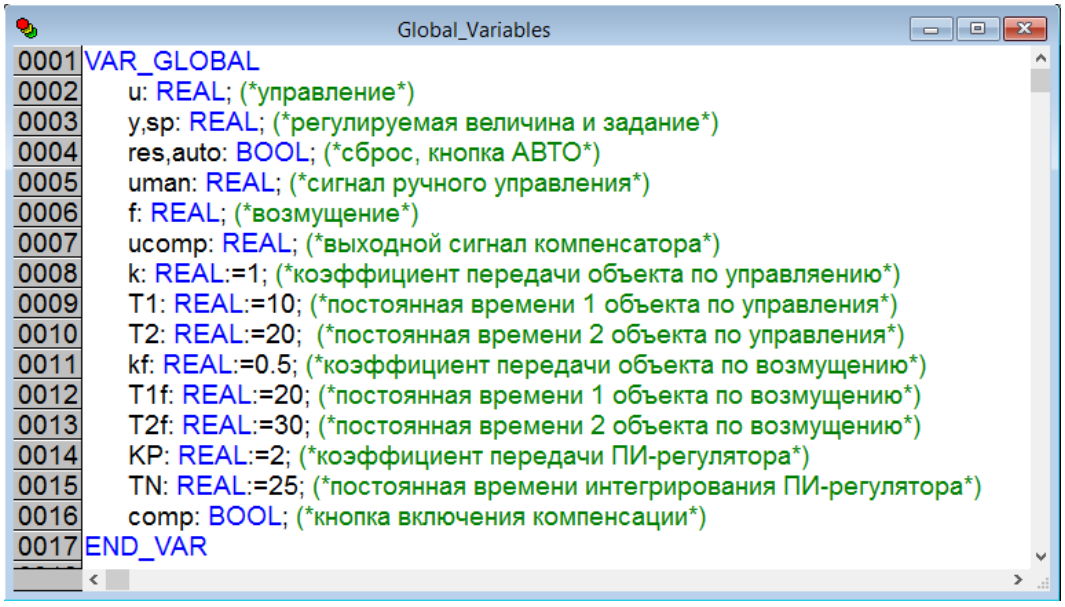


Рис. 51. Глобальные переменные проекта

На рис. 52 показана программная модель объекта управления.

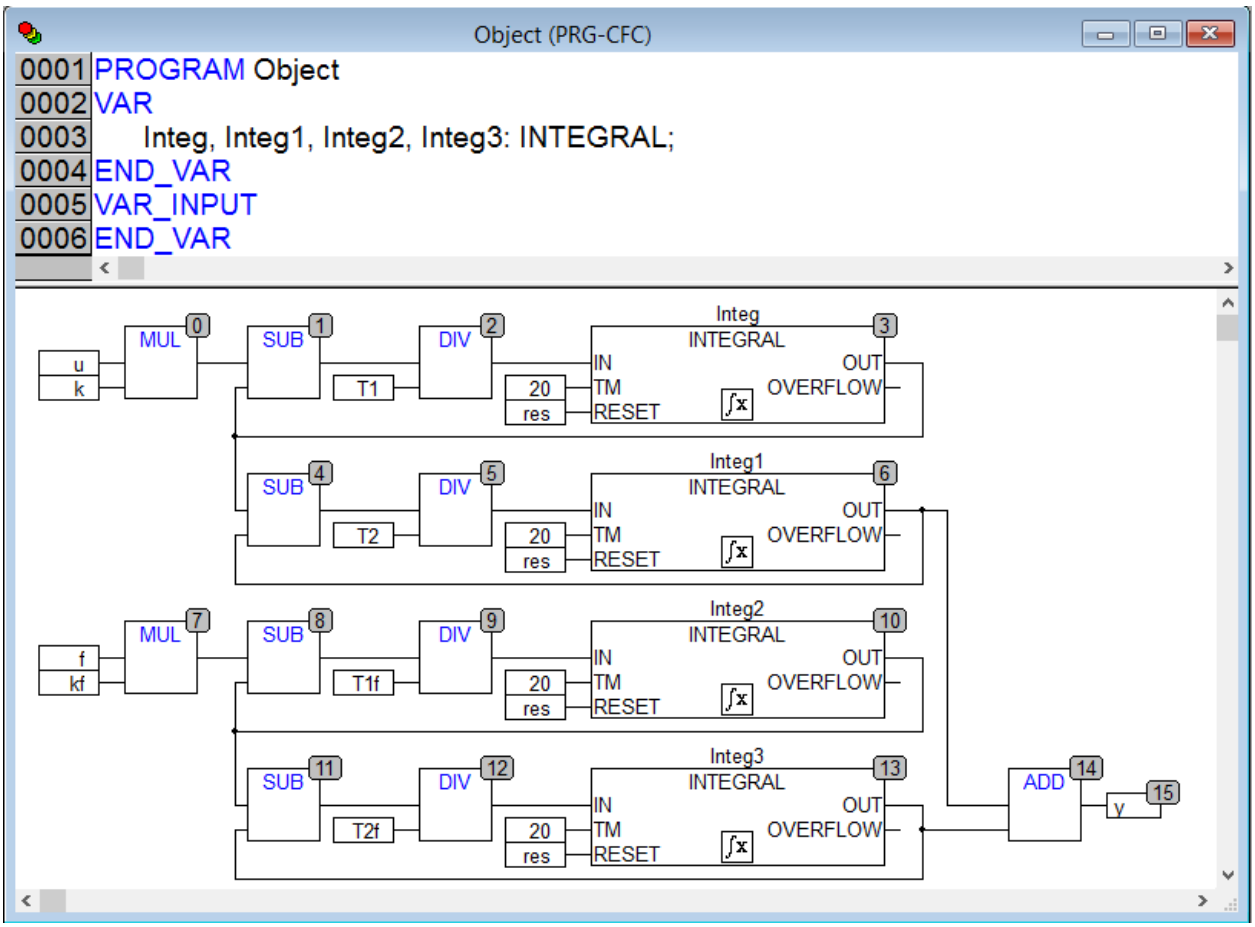


Рис. 52. Программная модель объекта управления

Сущность процесса компенсации возмущения демонстрируется на рис. 53.

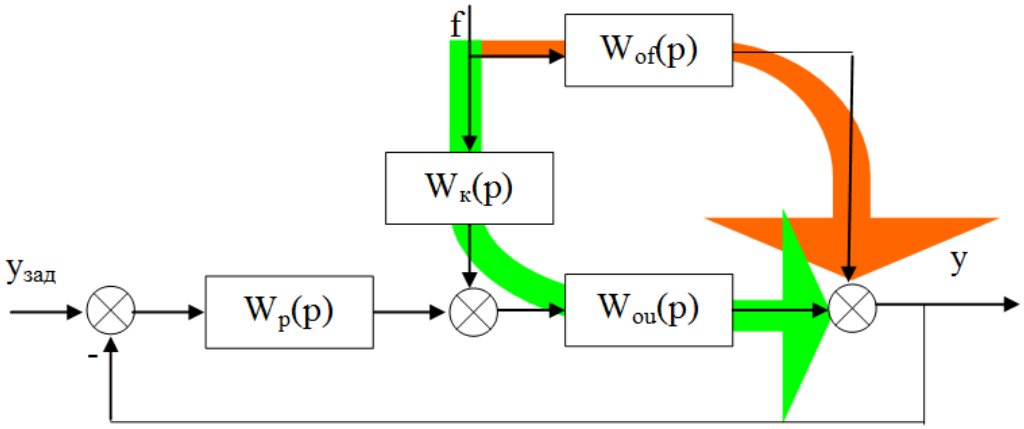


Рис. 53. Компенсация возмущения

В систему вводится дополнительное звено (компенсатор возмущения) с передаточной функцией $W_к(p)$, выбираемой таким образом, чтобы сигналы, показанные на рис. 53 большими стрелками, компенсировали друг друга:

$$f \times W_{of}(p) + f \times W_к(p) W_{ou}(p) = 0,$$

откуда

$$W_к(p) = -\frac{W_{of}(p)}{W_{ou}(p)}.$$

В нашем случае передаточная функция компенсатора принимает вид:

$$W_к(p) = -\frac{k_f}{k}\frac{T_1 p + 1}{T_{1f} p + 1} \times \frac{T_2 p + 1}{T_{2f} p + 1}.$$

Структура компенсатора показана на рис. 54.

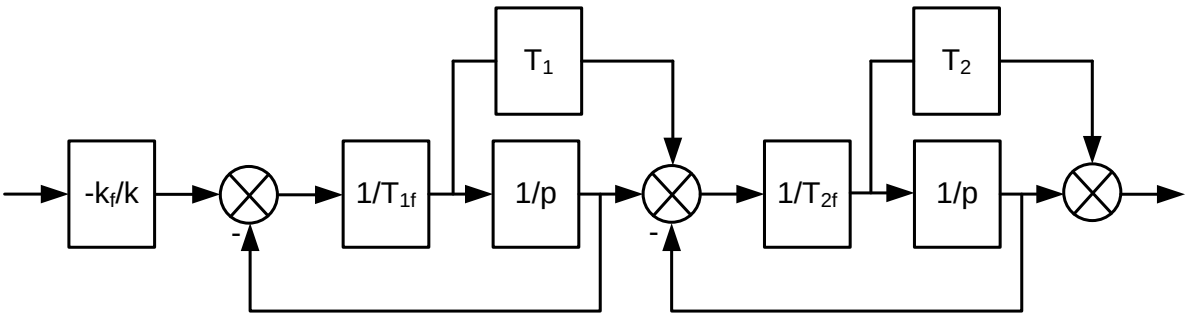


Рис. 54. Структурная схема компенсатора

Программная реализация компенсатора, приведенная на рис. 55, практически полностью воспроизводит эту структуру. Отсутствует только инверсия сигнала (она реализована в `PLC_PRG`).

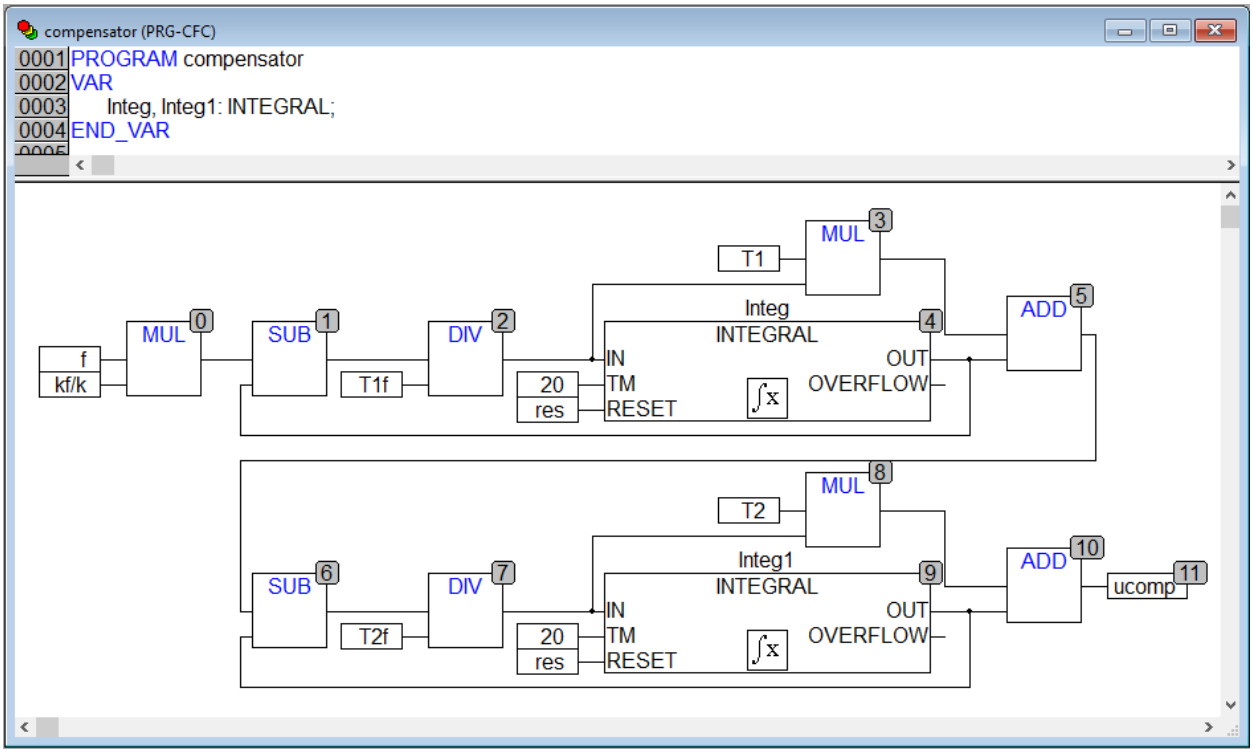


Рис. 55. Программная реализация компенсатора

Программы `Object` и `compensator` должны вызываться в рамках циклически выполняемой задачи, см. п. 2.1.

Кроме компенсатора система включает основной ПИ-регулятор, реализованный в виде отдельной программы `RegPI`, см. п. 2.2, рис. 47.

Панель управления показана на рис. 56.

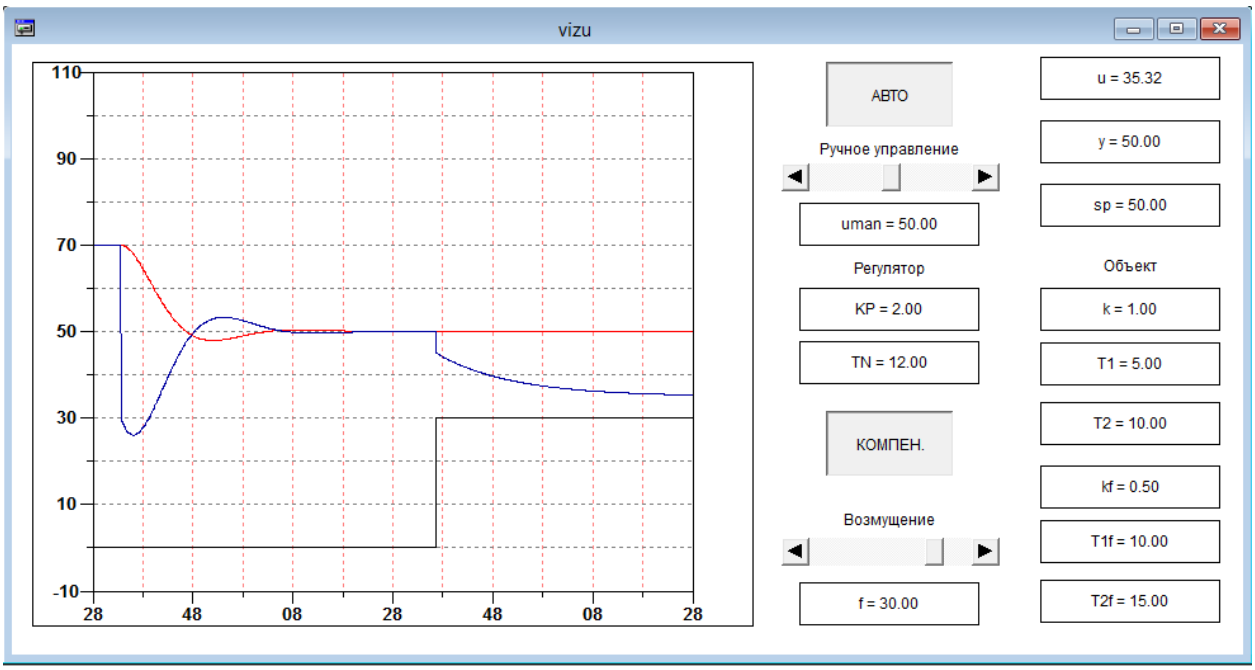


Рис. 56. Панель управления

В программе `PLC_PRG` вызывается ПИ-регулятор, и далее, если включена компенсация возмущения, из его выходного сигнала вычитается сигнал компенсатора, рис. 57.

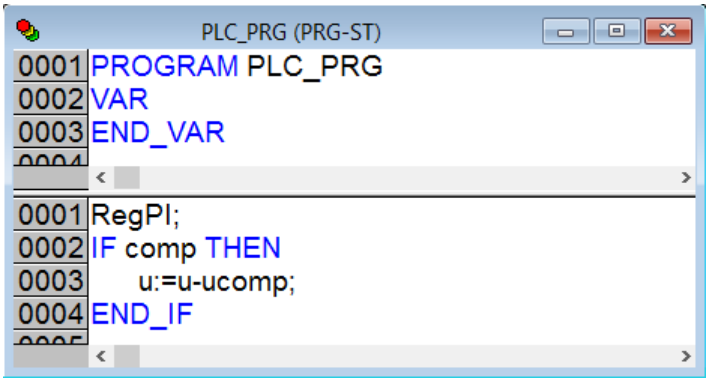


Рис. 57. Программа `PLC_PRG`

2.5 Система непрерывного регулирования (ПИ-регулятор) с исполнительным механизмом постоянной скорости

На практике в большинстве систем автоматического регулирования воздействие на объект управления осуществляется путем изменения подачи среды (например, теплоносителя) с помощью регулирующей арматуры (клапана, задвижки и т.д.). В большинстве случаев регулирующая арматура приводится в движение нерегулируемым электроприводом (электроприводом постоянной скорости). Как элемент системы такой привод управляется командами «включить на открытие», «включить на закрытие», «выключить». С

другой стороны, привод информирует систему управления о своем состоянии («открыт», «закрыт») и текущем положении. Состояния фиксируются концевыми выключателями, а положение измеряется измерительным преобразователем.

Структура системы, рассмотренной ниже, показана на рис. 58.

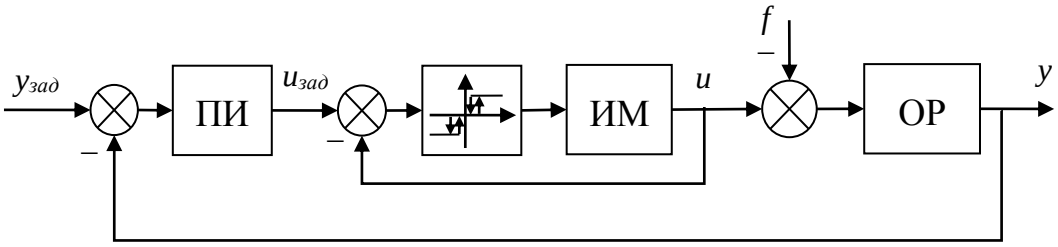


Рис. 58. Структура системы

Назначение внутреннего контура – воспроизведение исполнительным механизмом сигнала, формируемого ПИ-регулятором, в виде заданного перемещения рабочего органа регулирующей арматуры. (Предполагается, что «процент» открытия арматуры в точности равен «проценту» непрерывного управляющего сигнала, что в общем случае не так). Включение/отключение исполнительного механизма ИМ осуществляет трехпозиционный релейный регулятор. Точность воспроизведения задания определяется величиной зоны нечувствительности реле.

Список глобальных переменных приведен на рис. 59.

Global_Variables

```
0001 VAR_GLOBAL
0002     u: REAL; (*выход регулятора 0..100%*)
0003     y,sp: REAL; (*регулируемая величина и уставка регулятора*)
0004     res,auto: BOOL; (*сброс, автоматическое регулирование*)
0005     uman: REAL; (*сигнал ручного управления 0..100%*)
0006     open, close: BOOL; (*сигналы управления рег. клапаном*)
0007     full_time: REAL:=20; (*время полного хода ИМ, сек*)
0008     opened, closed: BOOL; (*сигналы концевых выключателей ИМ*)
0009     position: REAL; (*положение ИМ 0..100%*)
0010     KP: REAL:=2; (*коэффициент ПИ-регулятора*)
0011     TN: REAL:=25; (*постоянная времени интегрирования ПИ-регулятора*)
0012     delta: REAL:=0.4; (*зона нечувствительности релейного регулятора положения ИМ*)
0013     mintime: REAL:=100; (*минимальное время включения ИМ, мс*)
0014     otladka, open_BT  close_BT: BOOL; (*кнопки*)
0015     k: REAL:=1; (*коэффициент передачи объекта*)
0016     T1: REAL:=10; (*постоянная времени 1 объекта*)
0017     T2: REAL:=20; (*постоянная времени 2 объекта*)
0018     f: REAL; (*возмущение*)
0019 END_VAR
```

Рис. 59. Глобальные переменные

На рис. 60 показано основное окно визуализации. Система может находиться в трех состояниях:

– автоматическое управление (по схеме рис. 58);

– ручное управление, когда заданное положение исполнительного механизма задается вручную, а регулятор внутреннего контура его отрабатывает. ПИ-регулятор при этом не работает;

– отладка: прямое управление приводом, регуляторы не работают.

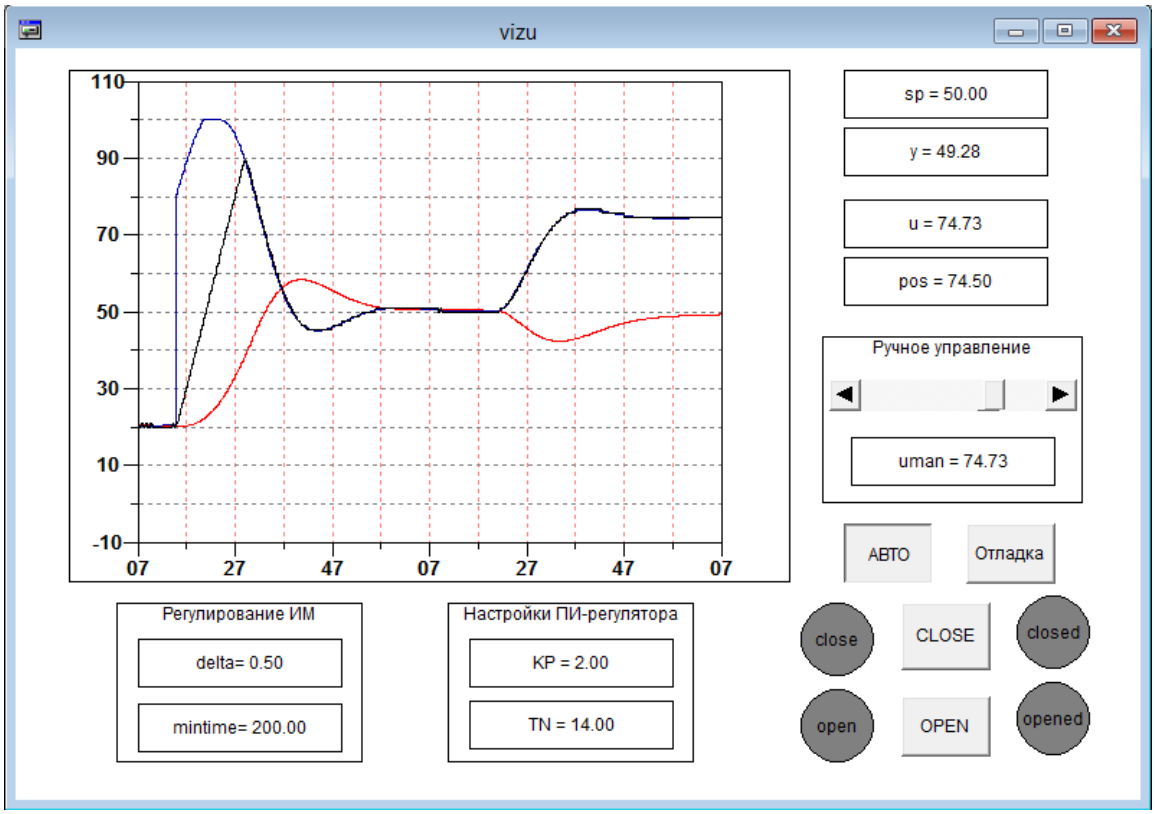


Рис. 60. Панель управления

В отличие от предыдущих систем «настройка» объекта и формирование возмущения вынесены в отдельное окно, показанное на рис. 61. Таким образом, основное окно максимально «приближено к реальности».

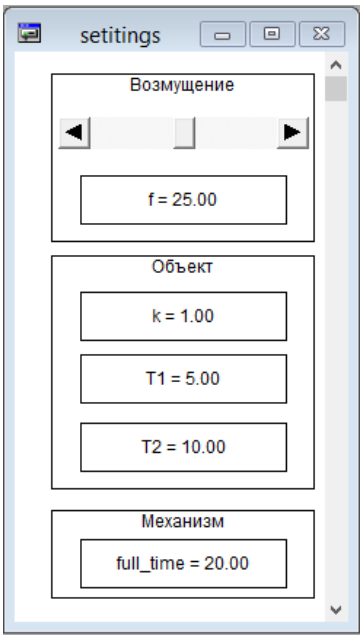


Рис. 61. Окно ввода параметров объекта

Программная модель исполнительного механизма оформлена в виде функционального блока, рис. 62. Входы блока: команды на открытие и закрытие и настроечный – время полного хода механизма. Выходы: дискретные сигналы концевых выключателей и «аналоговый» сигнал о положении механизма.

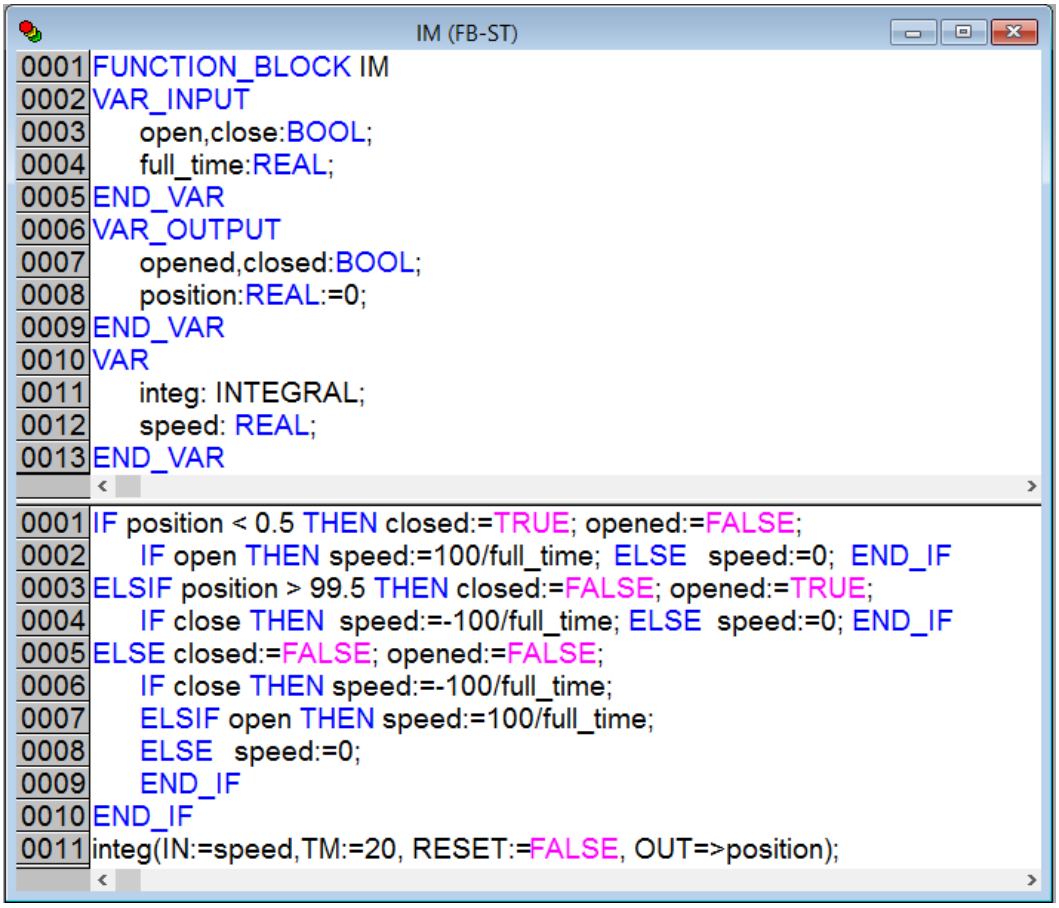


Рис. 62. Программная модель исполнительного механизма

Концевые выключатели «срабатывают», если положение меньше 0,5% или больше 99,5% полного хода. Скорость движения привода (локальная переменная `speed`) рассчитывается через время полного хода по результатам анализа состояния привода и поступающих команд. Полученное значение подается на вход интегратора `integ`, который рассчитывает положение механизма.

Программная модель объекта управления включает экземпляр функционального блока `IM` и собственно модель объекта по каналу управление – выход, рис. 63. Канал по возмущению вновь не детализирован, но само возмущение имеется и приложено к той же точке, что и управляющий сигнал. Программа `Object` вызывается в рамках циклически выполняемой задачи.

Релейный регулятор положения исполнительного механизма представлен функциональным блоком `RegIM`, рис. 64. На вход блока подаются непрерывный сигнал управления и настроечные сигналы: величина зоны нечувствительности реле и минимальное время включения/отключения привода.

Блок формирует дискретные сигналы на включение привода в направлении открытия и закрытия регулирующей арматуры.

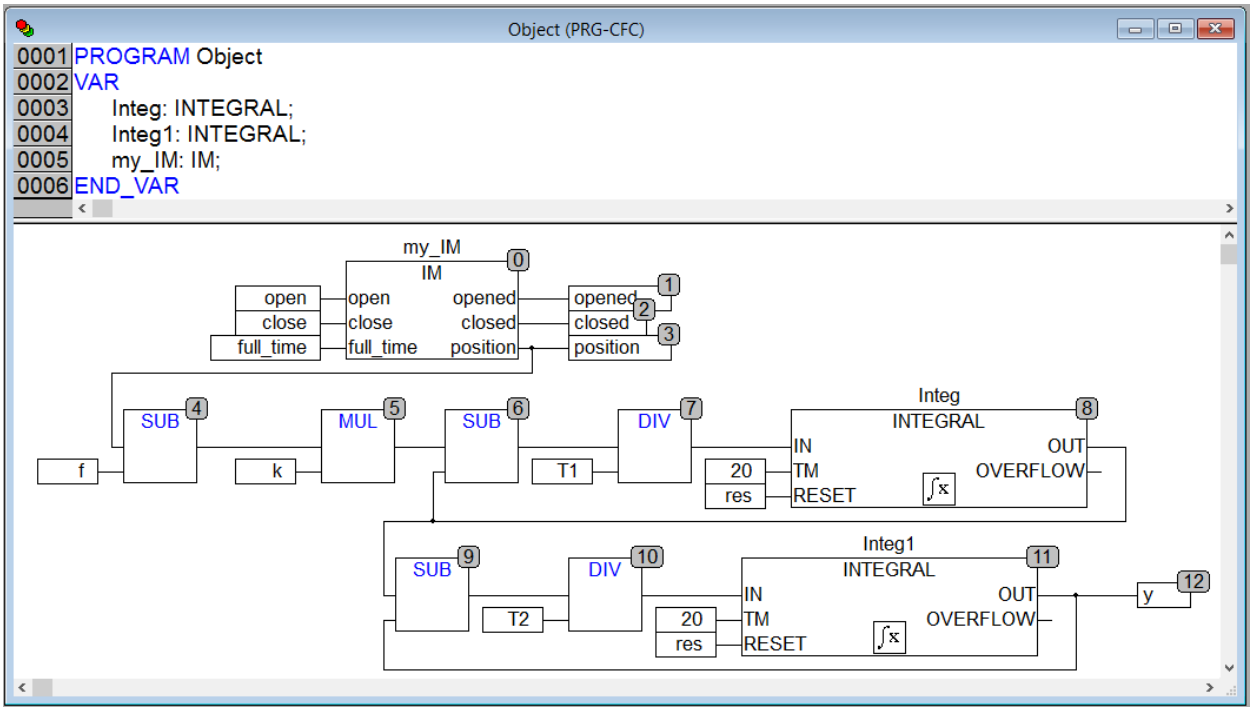


Рис. 63. Программная модель объекта управления

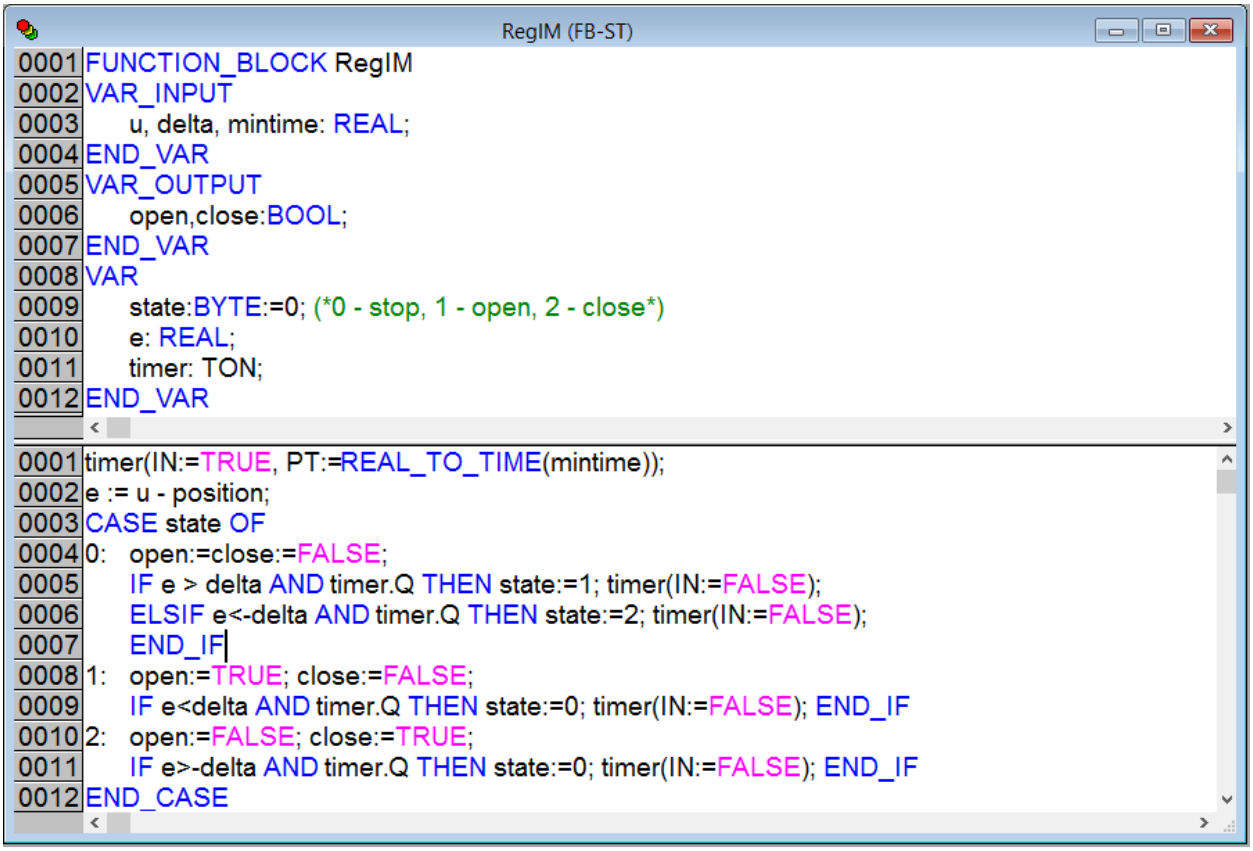


Рис. 64. Регулятор положения исполнительного механизма

Строго говоря, блок `RegIM` реализует трехпозиционный закон без гистерезиса, показанного на рис. 58. Защита от частых срабатываний осуществляется «напрямую» – путем запрета переводить привод из состояния «вклю-

чен» (точнее, двух состояний) в состояние «выключен» и обратно в течение времени `mintime`, отсчитываемого таймером. Таймер перезапускается при переходе из одного состояния в другое.

ПИ-регулятор по-прежнему оформлен в виде отдельной программы `RegPI` на языке CFC, см. рис. 47.

Программа `PLC_PRG`, свободно вызываемая в «своей» задаче, показана на рис. 65. Независимо от режима в `PLC_PRG` вызывается программа ПИ-регулирования. В ручном режиме и в режиме «Отладка» ПИ-регулятор «повторяет» сигнал ручного управления, в автоматическом – формирует управляющий сигнал самостоятельно. В режиме «Отладка» выход регулятора никак не используется, поскольку команды приводу выдает оператор с помощью кнопок панели. В ручном и автоматическом режимах выходной сигнал регулятора подается на вход экземпляра функционального блока `RegIM` (релейный регулятор положения привода).

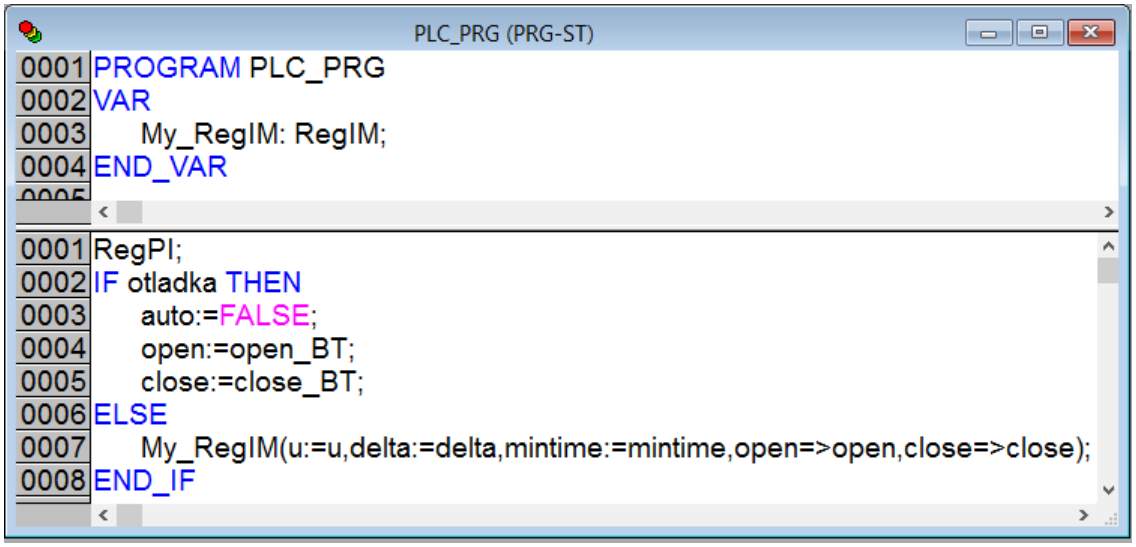


Рис. 65. Программа `PLC_PRG`

2.6 Система программного регулирования

Объектом управления является технологический процесс термической обработки некоторого жидкого продукта. Процесс осуществляется в емкости, в нижней части которой установлен трубчатый электрический нагреватель (ТЭН), мощность которого плавно регулируется при помощи тиристорного регулятора напряжения. Подачей и сливом продукта управляют электромагнитные клапаны, установленные на трубопроводах подачи и слива (в верхней части и на дне емкости). Уровень продукта контролируется тремя поплавковыми (дискретными) датчиками: максимального уровня, минимального уровня, уровня безопасного включения ТЭН (датчик установлен на отметке, гарантирующей нахождение ТЭН в жидкости).

Основное окно визуализации, имитирующее операторскую панель, показано на рис. 66. С помощью кнопок в нижнем левом углу вызываются дополнительные окна: для ввода параметров объекта (в учебных целях), рис. 67, и для вывода «инструкции» (задания), рис. 68.

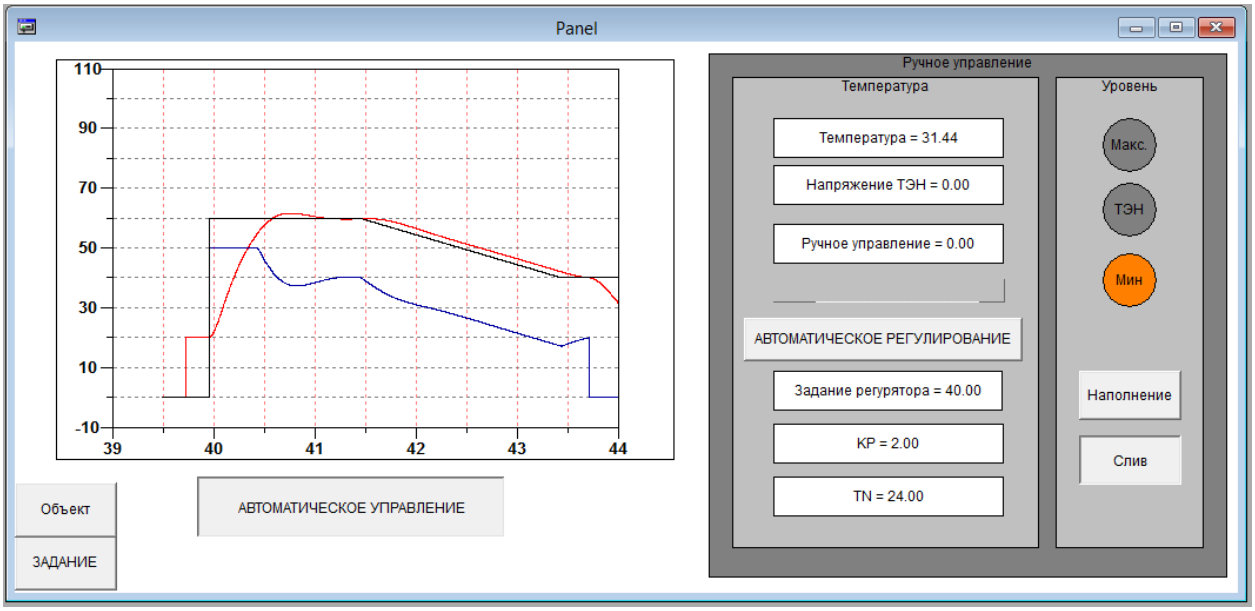


Рис. 66. Панель управления

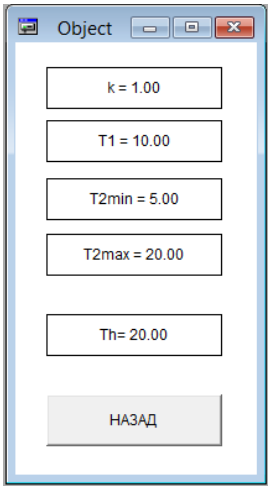


Рис. 67. Окно ввода параметров объекта

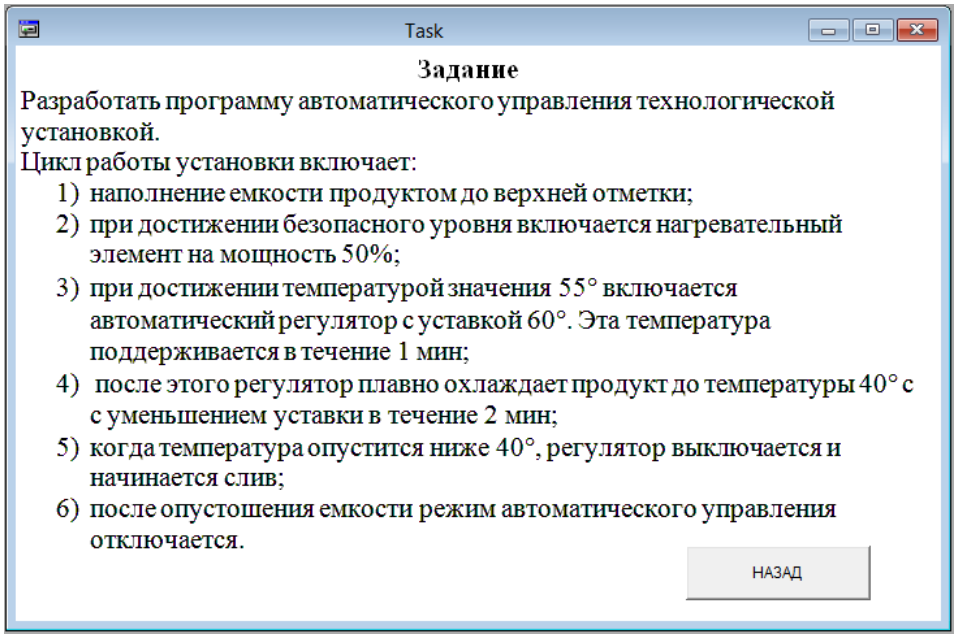


Рис. 68. Окно вывода «инструкции»

Список глобальных переменных проекта приведен на рис. 69.

Global_Variables

```
0001 VAR_GLOBAL
0002     (*ОБЪЕКТ*)
0003     level: REAL; (*уровень в емкости*)
0004     Th: REAL:=20; (*время полного наполнения емкости*)
0005     T1: REAL:=10; (*первая постоянная времени по нагреву - измеритель*)
0006     T2min: REAL:=5; (*вторая постоянная времени по нагреву, минимум*)
0007     T2max: REAL:=20; (*вторая постоянная времени по нагреву, максимум*)
0008     k: REAL:=1; (*коэффициент передачи по нагреву*)
0009     (*ВХОДЫ ПЛК*)
0010     minlevel: BOOL; (*минимальный уровень*)
0011     safelevel: BOOL; (*уровень безопасного включения ТЭН*)
0012     maxlevel: BOOL; (*максимальный уровень*)
0013     temp: REAL; (*температура продукта*)
0014     (*ВЫХОДЫ ПЛК*)
0015     valve_fill: BOOL; (*электромагнитный клапан на линии наполнения*)
0016     valve_drain: BOOL; (*электромагнитный клапан на линии стока*)
0017     u: REAL; (*сигнал управления  - напряжение ТЭН*)
0018     (*ВНУТРЕННИЕ ПЕРЕМЕННЫЕ ПЛК*)
0019     temp_sp: REAL; (*задание регулятору температуры*)
0020     res:BOOL; (*сброс всего*)
0021     auto_PI: BOOL; (*автоматический режим работы регулятора*)
0022     auto: BOOL; (*автоматический режим работы системы*)
0023     uman: REAL; (*сигнал ручного управления - напряжение ТЭН*)
0024     KP: REAL:=2; (*коэффициент передачи ПИ-регулятора*)
0025     TN: REAL:=24; (*постоянная времени интегрирования ПИ-регулятора*)
0026     fill_but:BOOL; (*кнопка Наполнение*)
0027     drain_but:BOOL;  (*кнопка Слив*)
0028 END_VAR
```

Рис. 69. Глобальные переменные проекта

Программная модель объекта управления показана на рис. 70.

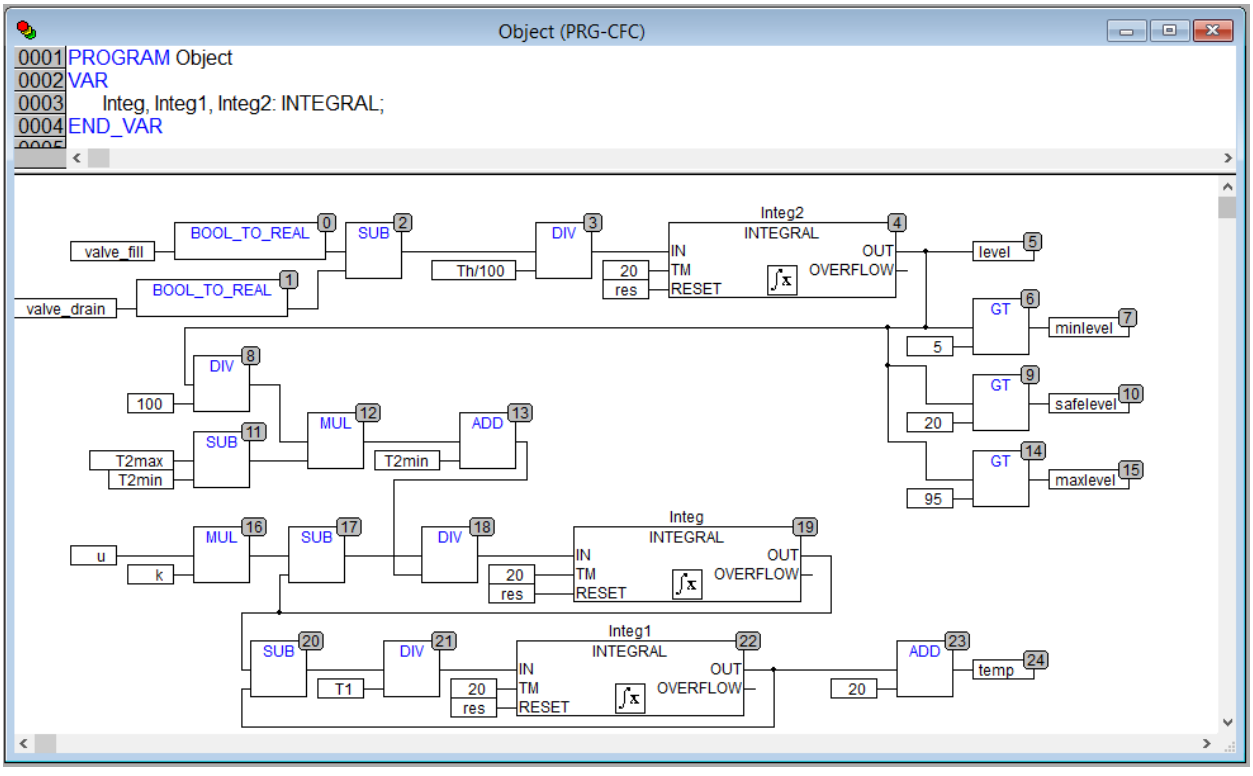


Рис. 70. Программная модель объекта управления

В верхней части диаграммы размещена модель объекта по уровню, построенная на интеграторе `integ2`. Выходные сигналы блоков преобразования типа `BOOL_TO_REAL` при активном состоянии входов принимают единичные значения. Уровень изменяется в диапазоне от 0 до 100%. Следовательно, постоянная времени интегрирования должна быть равна времени полного наполнения `Th`, деленному на 100. Действительно, если в начальный момент времени (t = 0) бак был пуст и клапан наполнения открыт, то в момент времени $t = T_h$ уровень станет равным

$$h(T_h) = \int_0^{T_h} \frac{100}{T_h} \times 1 \times dt = \frac{100}{T_h} t \Big|_0^{T_h} = \frac{100 T_h}{T_h} = 100.$$

Объект регулирования по температуре, как и ранее, описывается апериодическим звеном второго порядка, однако одна из его постоянных времени (вторая) изменяется в зависимости от уровня: чем больше уровень, тем больше эта постоянная и, следовательно, процесс нагрева происходит медленнее. В модели реализована упрощенная линейная зависимости постоянной времени от уровня:

$$T_2 = T_{2,\min} + \left(T_{2,\max} - T_{2,\min}\right) \frac{h}{100}.$$

Программа `Object` вызывается в рамках циклически выполняемой задачи.

ПИ-регулятор температуры оформлен в виде отдельной программы `RegPI` на языке CFC. В показанной на рис. 47 программе требуется лишь изменить имена входных переменных блоков `reg` и `MOVE`: `y -> temp`, `sp -> temp_sp`, `auto -> auto_PI`.

Программа `PLC_PRG` (вызывается свободно в «своей» задаче) показана на рис. 71.

Если система не находится в режиме автоматического управления, управление клапанами производится с помощью кнопок панели управления. Однако при достижении предельных уровней система автоматически выключает соответствующие кнопки, а, следовательно, наполнение или слив. Ручное и автоматическое (при ручном задании уставки) регулирование температуры запрещается, если уровень ниже безопасного.

В режиме автоматического управления выполняется пошаговый процесс, воспроизводящий «инструкцию», см. рис. 68.

На этапе 0 производится наполнение емкости продуктом до «безопасного уровня» без включения нагревательного элемента: регулятор находится в режиме ручного управления, сигнал управления равен нулю.

На этапе 1 нагреватель включается на 50% мощности. Сигнал задания устанавливается равным 60°, хотя регулятор по-прежнему остается в ручном режиме. «Подготовка» уставки позволит в дальнейшем безударно перейти в режим автоматического регулирования.

PLC_PRG (PRG-ST)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     state:BYTE:=0;
0004     timer: TON;
0005 END_VAR
```

```
0001 IF NOT AUTO THEN
0002     valve_fill:= fill_but AND NOT maxlevel;
0003     IF maxlevel THEN
0004         fill_but:=FALSE;
0005     END_IF
0006     valve_drain:=drain_but AND minlevel;
0007     IF NOT minlevel THEN
0008         drain_but:=FALSE;
0009     END_IF
0010     IF NOT safelevel THEN
0011         auto_PI:=FALSE;
0012         uman:=0;
0013     END_IF
0014     state:=0;
0015 ELSE
0016     CASE state OF
0017     0:
0018         auto_PI:=FALSE;
0019         uman:=0;
0020         valve_fill:= fill_but:=TRUE;
0021         valve_drain:=drain_but:=FALSE;
0022         IF  safelevel THEN state:=1; END  IF
0023     1:
0024         auto_PI:=FALSE;
0025         temp_sp:=60;
0026         uman:=50;
0027         valve_fill:= fill_but:=NOT maxlevel;
0028         valve_drain:=drain_but:=FALSE;
0029         IF temp > 55 THEN state:=2; END_IF
0030     2:
0031         auto_PI:=TRUE;
0032         temp_sp:=60;
0033         uman:=50;
0034         valve_fill:= fill_but:=NOT maxlevel;
0035         valve_drain:=drain_but:=FALSE;
0036         timer(IN:=TRUE, PT:=t#1m);
0037         IF timer.Q THEN state:=3; timer(IN:=FALSE); END_IF
0038     3:
0039         auto_PI:=TRUE;
0040         uman:=0;
0041         valve_fill:= fill_but:=NOT maxlevel;
0042         valve_drain:=drain_but:=FALSE;
0043         timer(IN:=TRUE, PT:=t#2m);
0044         temp_sp:=60 - TIME_TO_REAL(timer.ET)/1000*1/6;
0045         IF timer.Q AND temp < 40 THEN state:=4; timer(IN:=FALSE); END_IF
0046     4:
0047         auto_PI:=FALSE;
0048         uman:=0;
0049         valve_fill:= fill_but:=FALSE;
0050         valve_drain:=drain_but:=minlevel;
0051         IF NOT minlevel THEN auto :=FALSE; END_IF
0052     END_CASE
0053 END_IF
0054 RegPI;
0055
```

Рис. 71. Программа `PLC_PRG`

На этапе 2 в течение 1 минуты производится стабилизация температуры на уровне 60°.

На этапе 3 производится плавное охлаждение продукта, уставка регулятора в течение 2 минут снижается с 60° до 40°. Переход к следующему этапу происходит в момент, когда температура действительно достигнет 40°.

На этапе 4 осуществляется слив продукта с последующим отключением режима автоматического управления.

В режиме автоматического управления необходимо предусмотреть запрет пользователю вводить значения сигнала ручного управления, изменять режим работы регулятора и его уставку. Для полей ввода числовых значений и кнопки это можно сделать «напрямую», как показано на рис. 72.

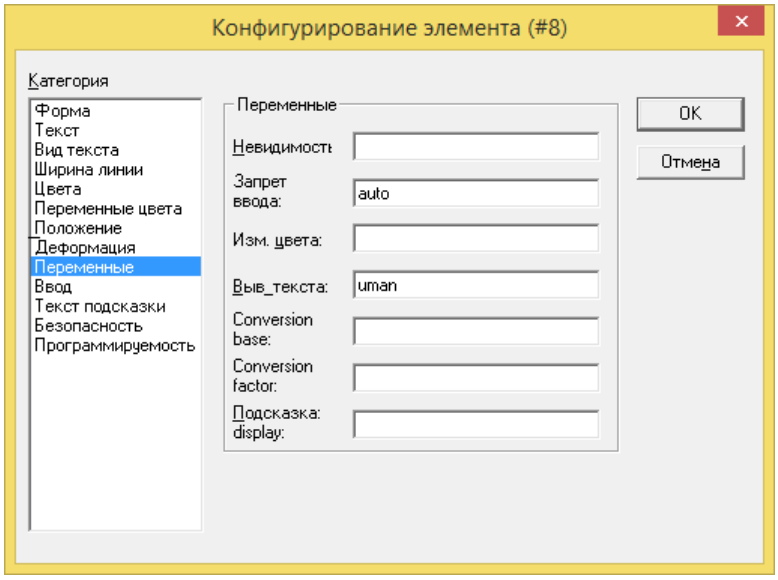


Рис. 72. Запрет ввода

«Ползунок» задания сигнала ручного управления в режиме автоматического управления можно сделать невидимым, рис. 73.

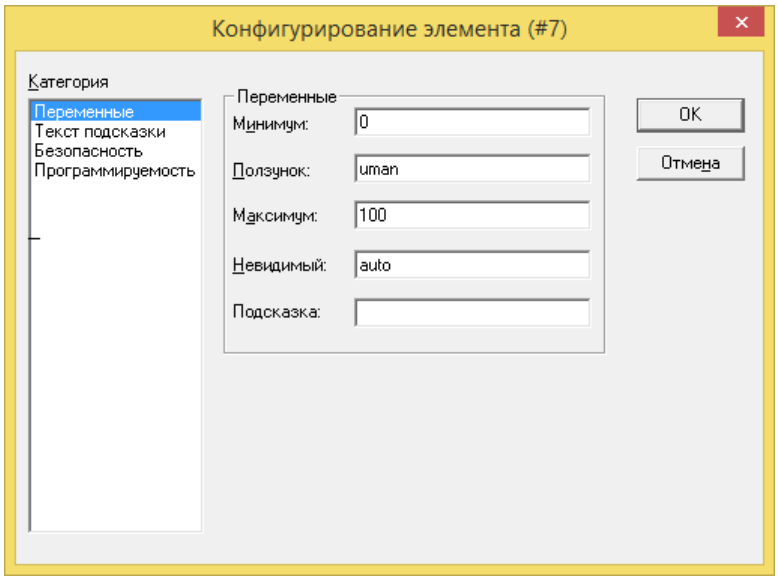


Рис. 73. Скрытие ползунка в автоматическом режиме

В системах числового программного управления программа изменения положения различных органов и приводов механизма отделена от «исполнительной системы» (часть которой – тоже программа) и выполняется ею. Таким образом, нужно разделять систему исполнения и «программу движения». Последняя в простейшем случае может быть представлена жестко структурированным массивом, содержащим задания регуляторам и другую необходимую информацию для каждого «шага» движения.

Далее рассматривается упрощенная система управления манипулятором, перемещающим цилиндрические металлические детали с падающего транспортера на «поддоны» с помощью электромагнитного схвата, рис. 74-78.

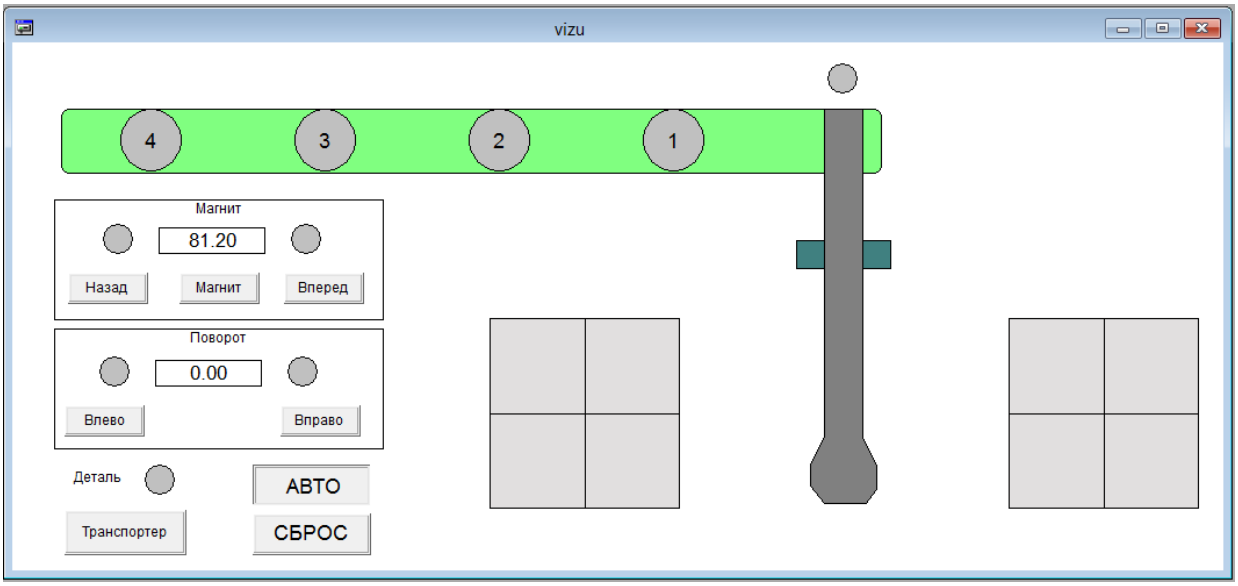


Рис. 74. Манипулятор в работе: ожидание первой детали

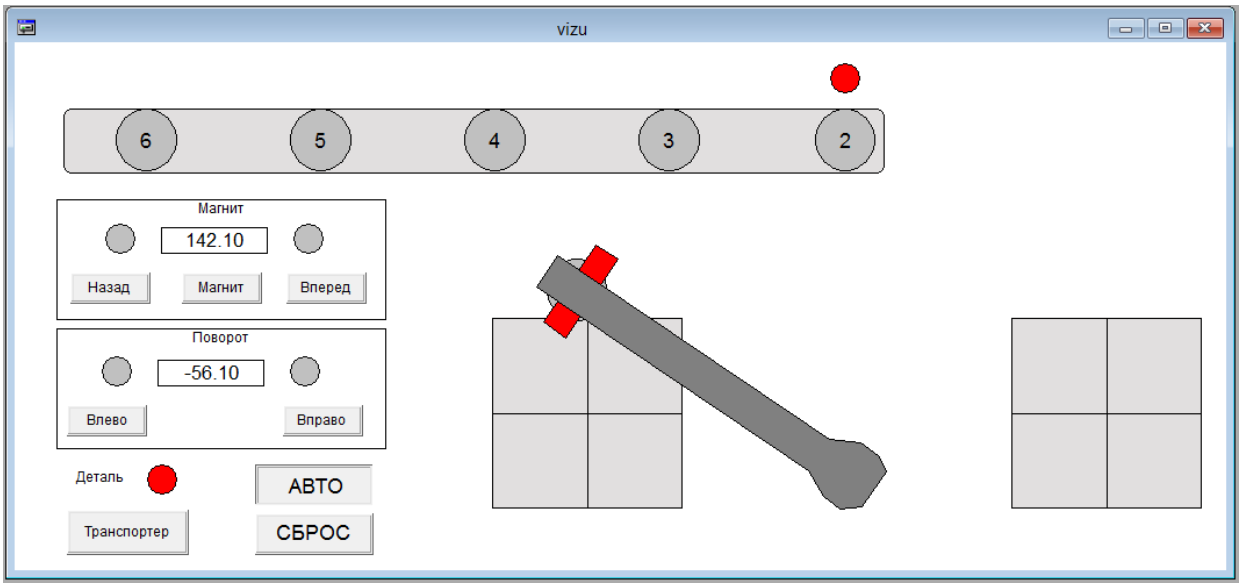


Рис. 75. Манипулятор в работе: перемещение первой детали

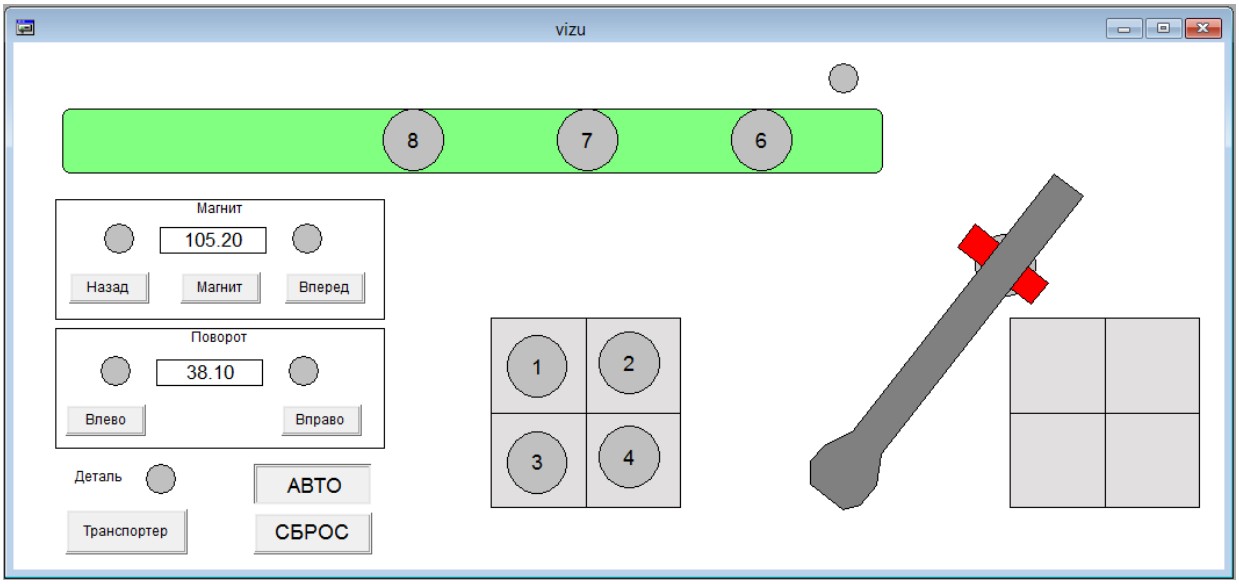


Рис. 76. Манипулятор в работе: перемещение пятой детали

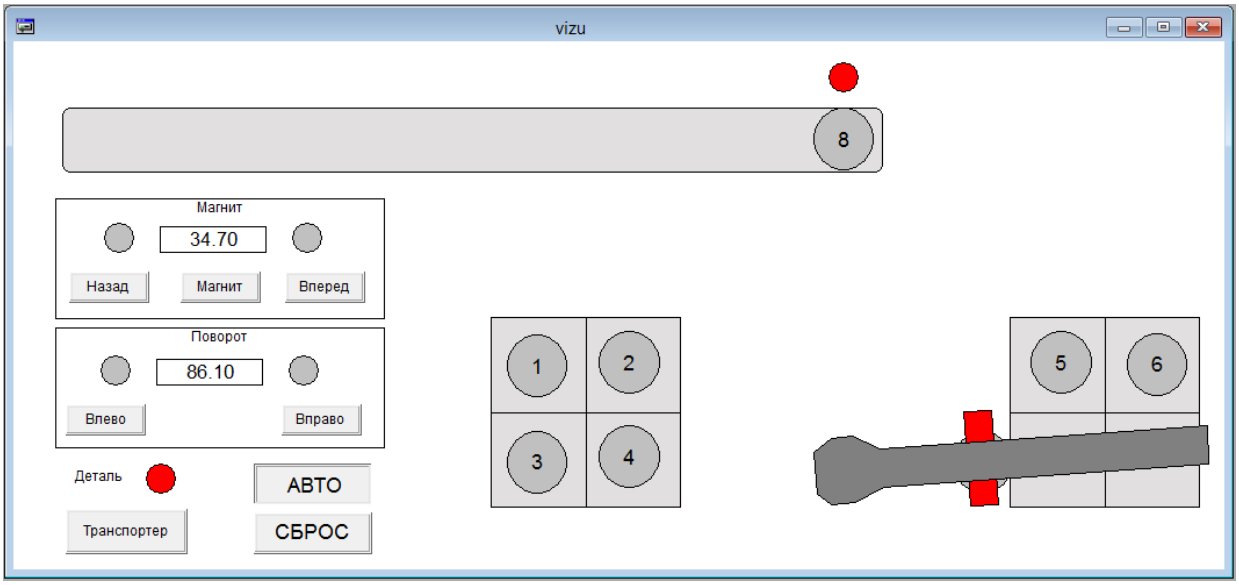


Рис. 77. Манипулятор в работе: перемещение седьмой детали

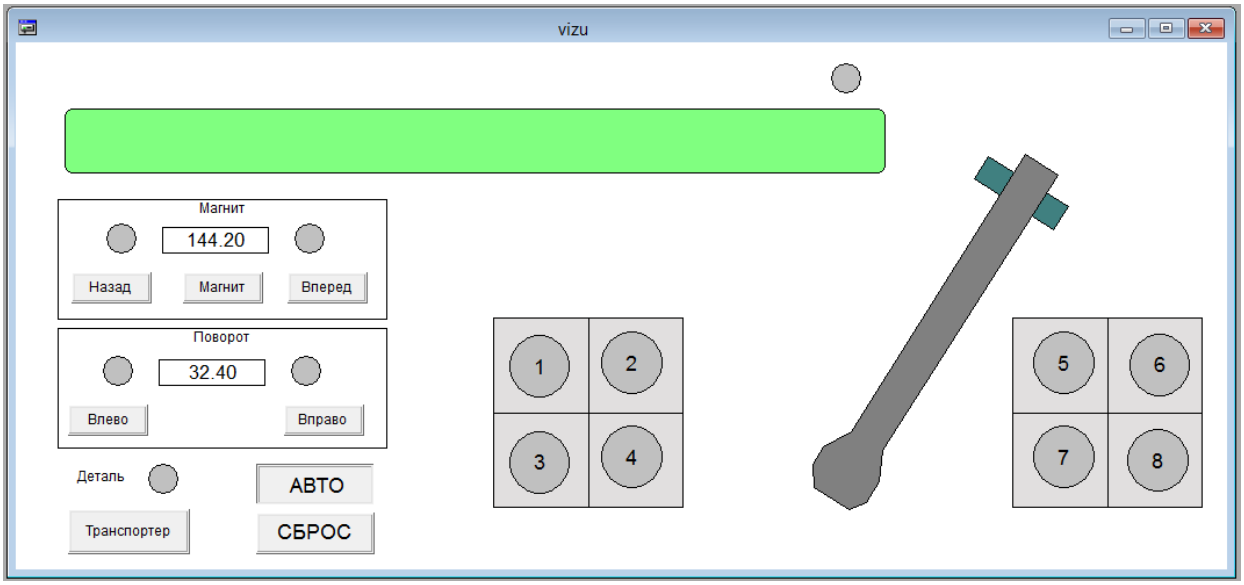


Рис. 78. Манипулятор в работе: возврат в исходное положение

В автоматическом режиме цикл работы манипулятора ограничен перемещением восьми деталей. Предполагается, что после этого погрузчик заменит заполненные поддоны на пустые, и цикл повторится сначала. Можно также организовать работу таким образом, чтобы во время заполнения одного поддона заменялся другой, тогда манипулятор будет работать «непрерывно», но в нашем случае такая задача не ставится, поскольку это «перегрузило» бы программную модель объекта, практически никак не затрагивая алгоритмов управления.

Список глобальных переменных приведен на рис. 79.

Global_Variables

```
0001 VAR_GLOBAL
0002     (*ВХОДЫ КОНТРОЛЛЕРА*)
0003     angle:REAL:=0; (*угол поворота манипулятора*)
0004     pos:REAL:=50; (*продольное перемещение магнита*)
0005     off_clock_KV, on_clock_KV:BOOL; (*концевые выключатели поворота*)
0006     magn_back_KV, magn_forv_KV:BOOL; (*концевые выключатели перемещения магнита*)
0007     on_place:BOOL:=FALSE; (*датчик наличия детали*)
0008
0009     (*ВЫХОДЫ КОНТРОЛЛЕРА*)
0010     transp: BOOL:=FALSE; (*транспортер включить*)
0011     magn_forv:BOOL:=FALSE; (*включить магнит вперед*)
0012     magn_back:BOOL:=FALSE; (*включить магнит назад*)
0013     off_clock:BOOL:=FALSE; (*поворот против часовой*)
0014     on_clock:BOOL:=FALSE; (*поворот по часовой*)
0015     magn: BOOL:=FALSE; (*включение магнита*)
0016
0017     (*ОРГАНЫ ВИЗУАЛИЗАЦИИ*)
0018     magn_forv_but:BOOL:=FALSE; (*кнопка "Магнит вперед"*)
0019     magn_back_but:BOOL:=FALSE; (*кнопка "Магнит назад"*)
0020     magn_but:BOOL:=FALSE; (*кнопка "Магнит включить"*)
0021     off_clock_but:BOOL:=FALSE; (*кнопка поворота против часовой*)
0022     on_clock_but:BOOL:=FALSE; (*кнопка поворота по часовой*)
0023     transp_but:BOOL:=FALSE; (*кнопка включения транспортера*)
0024     auto:BOOL:=FALSE; (*автоматический режим*)
0025
0026     (*НАСТРОЙКИ*)
0027     transp_speed: REAL:=0.3; (*"скорость" транспортера*)
0028     magn_speed: REAL:=0.3; (*"скорость" магнита*)
0029     rot_speed: REAL:=0.3; (*"скорость" поворота*)
0030
0031     (*ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ*)
0032     magnX:REAL:=0; (*положение магнита по x относительно исходного*)
0033     magnY:REAL:=0; (*положение магнита по y относительно исходного*)
0034     bodies: ARRAY [1..8] OF body:= (name:=1,x:=-130,y:=-190,a:=0,invisible:=TRUE, state:=0),
0035                                    (name:=2, x:=-230,y:=-190,a:=0,invisible:=TRUE, state:=0),
0036                                    (name:=3,x:=-330,y:=-190,a:=0,invisible:=TRUE, state:=0),
0037                                    (name:=4,x:=-430,y:=-190,a:=0,invisible:=TRUE, state:=0),
0038                                    (name:=5,x:=-530,y:=-190,a:=0,invisible:=TRUE, state:=0),
0039                                    (name:=6,x:=-630,y:=-190,a:=0,invisible:=TRUE, state:=0),
0040                                    (name:=7,x:=-730,y:=-190,a:=0,invisible:=TRUE, state:=0),
0041                                    (name:=8,x:=-830,y:=-190,a:=0,invisible:=TRUE, state:=0);
0042     res: BOOL; (*сброс*)
0043 END_VAR
```

Рис. 79. Глобальные переменные проекта

В списке вспомогательных переменных присутствует массив `bodies` из восьми структур типа `body`, описывающих перемещаемые детали. Объявление структурного типа показано на рис. 80.

body

```
0001 TYPE body :
0002 STRUCT
0003     name:BYTE:=0;
0004     x:REAL:=0;
0005     y:REAL:=0;
0006     a:REAL:=0;
0007     invisible:BOOL:=FALSE;
0008     state:BYTE:=0; (*0 - на ленте, 1 - в схвате, 2 - в корзине, 3 - на полу, 4 и более -  вне игры*)
0009 END_STRUCT
0010 END_TYPE
```

Рис. 80. Структурный тип `body`

На визуализации все восемь деталей (окружностей) «первоначально» располагаются в центре вращения манипулятора (элемент «полигон»). Эта точка является началом координат во всей системе.

Положением и состоянием деталей управляют переменные из массива `bodies`, как показано на рис. 81, 82 для восьмой детали.

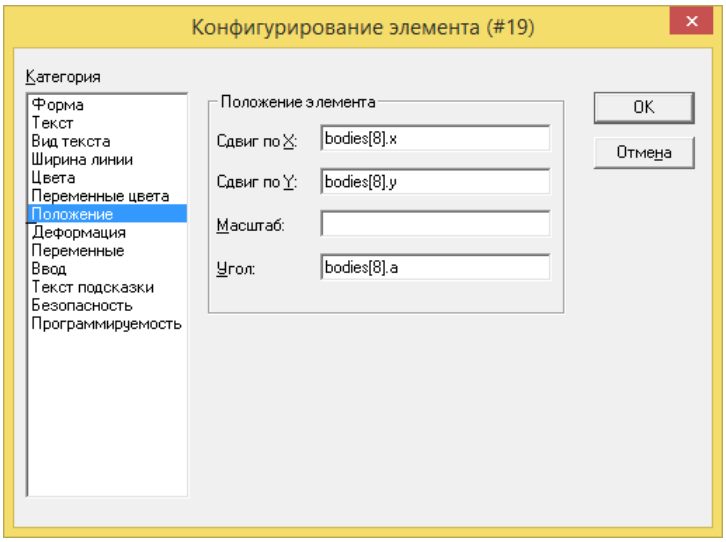


Рис. 81. Управление положением восьмой детали

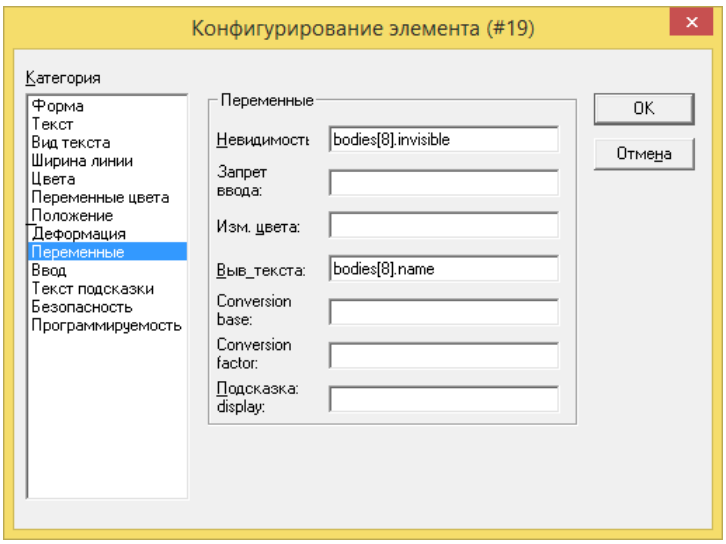


Рис. 82. Управление состоянием восьмой детали

Программа, моделирующая объект управления, показана на рис. 83.

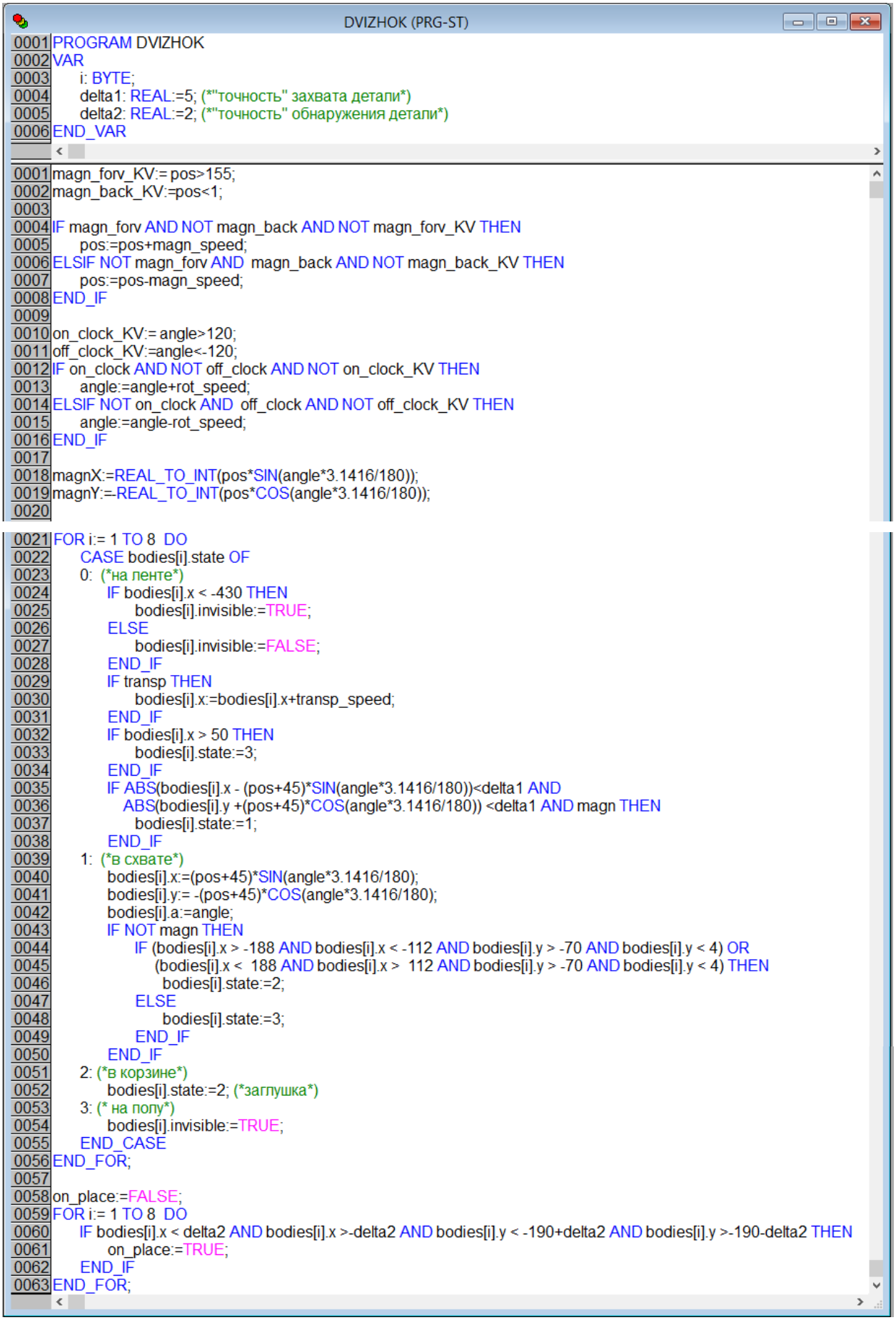


Рис. 83. Программная модель объекта управления

На основе анализа существующего состояния объекта и управляющих сигналов модель формирует для контроллера входные сигналы:

– концевых выключателей и измерителей положения приводов поворота манипулятора и перемещения магнита;

– датчика обнаружения детали в зоне захвата.

Кроме того, модель обслуживает визуализацию, рассчитывая:

– координаты магнита на визуализации. Магнит представлен элементом «полигон», его центр вращения совпадает с центром вращения манипулятора, поэтому он «автоматически» поворачивается вместе с «носителем», однако вследствие продольного перемещения координаты магнита требуют расчета;

– положение и состояние всех деталей. В цикле `FOR` анализируется состояние каждой детали и в зависимости от него выполняются соответствующие действия, включая возможную смену состояния. Если деталь находится на ленте, определяется ее видимость. Когда транспортер находится во включенном состоянии, пересчитывается положение детали. Если деталь находится под магнитом (с определенным допуском) и магнит включен, деталь переходит в состояние «в схвате». В этом состоянии координаты детали совпадают с координатами магнита. При отключении магнита определяется новое состояние детали. Если ее текущие координаты попадают в квадраты поддонов, деталь переходит в состояние «в корзине», иначе – в состояние «на полу». В состоянии «в корзине» с деталью не происходит ничего (это задел для возможной будущей модификации программы). В состоянии «на полу» деталь становится невидимой.

На данном этапе система является уже вполне работоспособной. Если в программе `PLC_PRG` предусмотреть «трансляцию» значений переменных, привязанных к кнопкам панели управления, в выходные переменные контроллера, можно управлять процессом переноса деталей вручную.

Автоматическое управление будет реализовано при помощи регуляторов положения приводов поворота манипулятора и перемещения магнита. Здесь будут задействованы простейшие релейные трехпозиционные регуляторы – экземпляры функционального блока `Regulator`, рис. 84.

Входными переменными блока являются заданное и фактическое положение привода, а также величина зоны нечувствительности. Выходные переменные – команды на включение привода в прямом и обратном направлениях, и сигнал `OK` об окончании процесса позиционирования.

Во внутреннюю локальную переменную `e` заносится значение ошибки регулирования. По результатам сравнения ошибки с порогами срабатывания принимаются решения о включении привода в том или ином направлении.

Собственно числовое программное управление осуществляется программой, показанной на рис. 85.

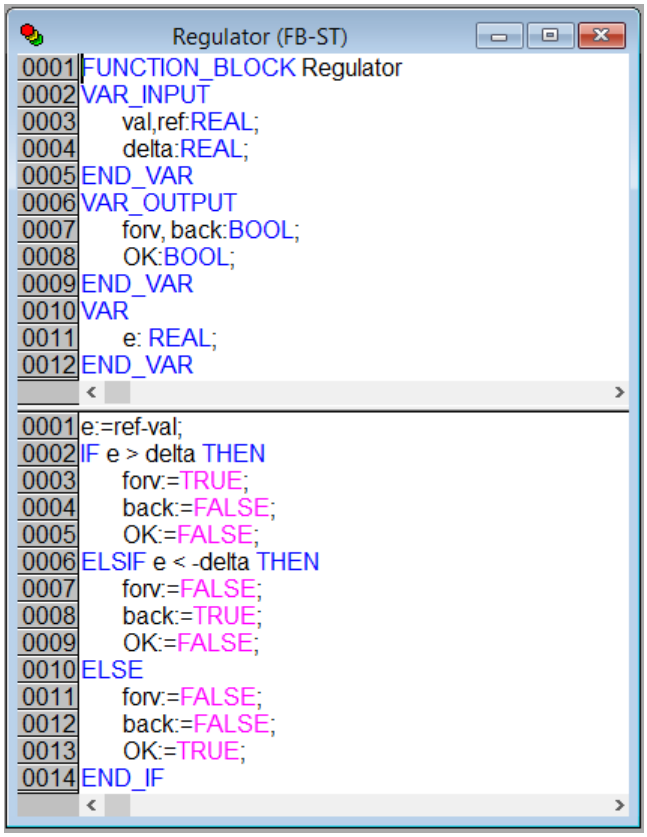


Рис. 84. Регулятор положения

Программа движения для простоты оформлена в виде двухмерного целочисленного массива `MOVE_PROG`. Массив может состоять из произвольного количества «строк», каждая из которых описывает позицию манипулятора и выполняемые в этой позиции действия и состоит из 4 чисел. Первые два числа – это заданные координаты манипулятора (перемещение магнита и угол поворота манипулятора), третье число – заданное состояние магнита (0 – выключен, 1 – включен), четвертое – выдержка времени (в мс) после отработки регуляторов до начала перехода в следующую позицию.

Массив объявлен в секции `VAR CONSTANT`, т.е., по сути, является константой. Поэтому при отладке программы движения каждая загрузка проекта в контроллер будет обновлять значения элементов массива без сброса контроллера, который потребовался бы при объявлении массива переменной. Это чрезвычайно упрощает отладку.

Программа вызывает на исполнение два экземпляра функционального блока `Regulator` (регуляторы положения магнита `RegP` и угла поворота манипулятора `RegA`), передавая им в качестве заданий соответствующие значения из массива `MOVE_PROG`. По окончанию отработки заданий регуляторы сообщают об этом установкой локальных переменных `ok1` и `ok2` в `TRUE`.

Включение магнита, если он еще не был включен, производится при наличии указания на это и срабатывании датчика наличия детали в зоне захвата. После включения магнита устанавливается в `TRUE` переменная `ok3`.

После того, как все механизмы отработали, запускается таймер. Выдержка времени берется из массива `MOVE_PROG`.

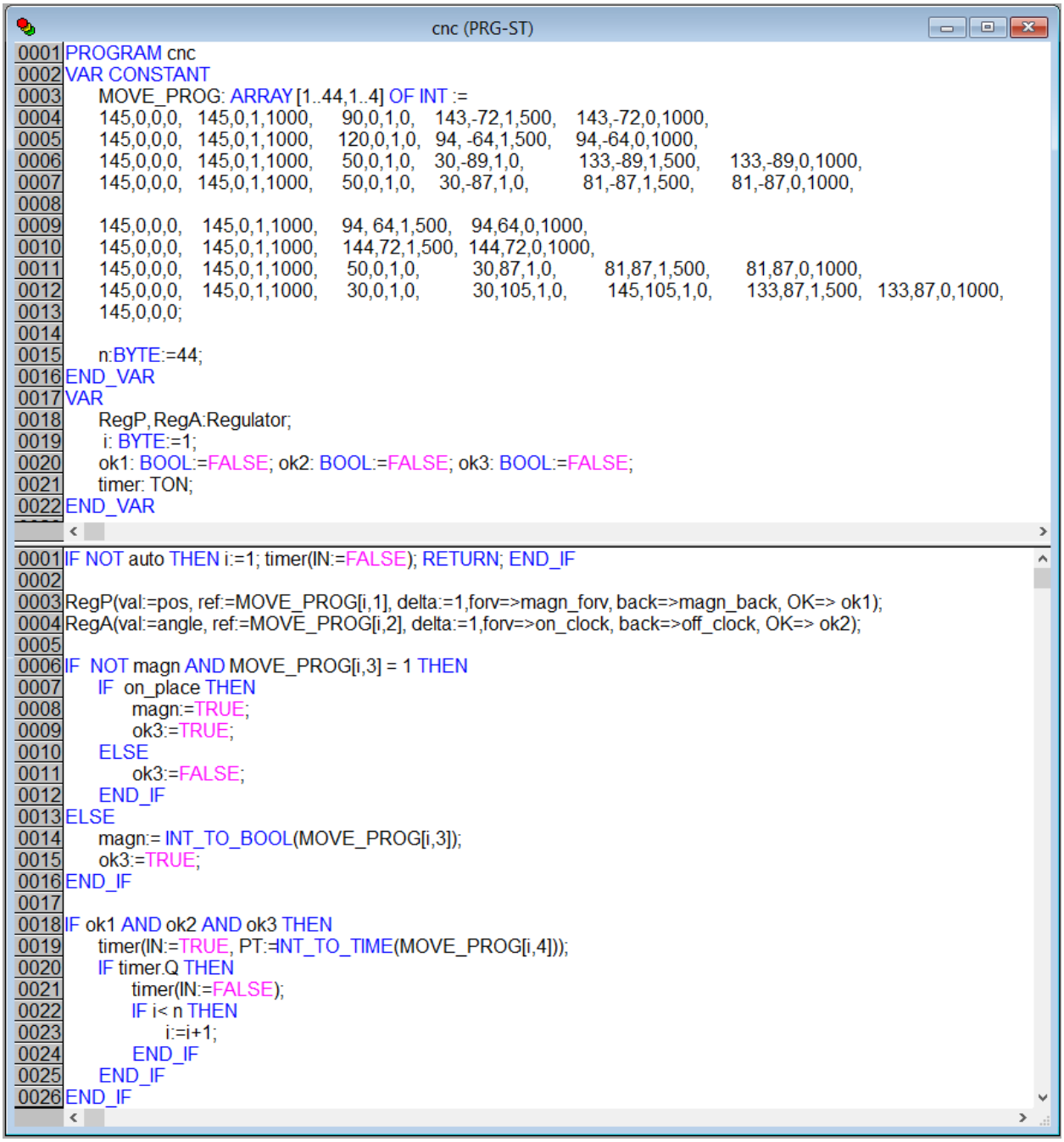


Рис. 85. Программа ЧПУ

При срабатывании таймера локальная переменная `i` увеличивается на единицу и программа переходит к выполнению следующего шага программы движения (если текущий шаг не был последним).

Программа PLC_PRG приведена на рис. 86.

В ручном режиме управление процессом производится при помощи кнопок. Имеется также возможность «сброса системы» вызовом программы `SBROS`. В автоматическом режиме производится включение транспортера при отсутствии детали в зоне захвата. Программа ЧПУ запускается вне зависимости от режима (это требуется для возобновления ее корректной работы после сброса), также как и модель объекта.

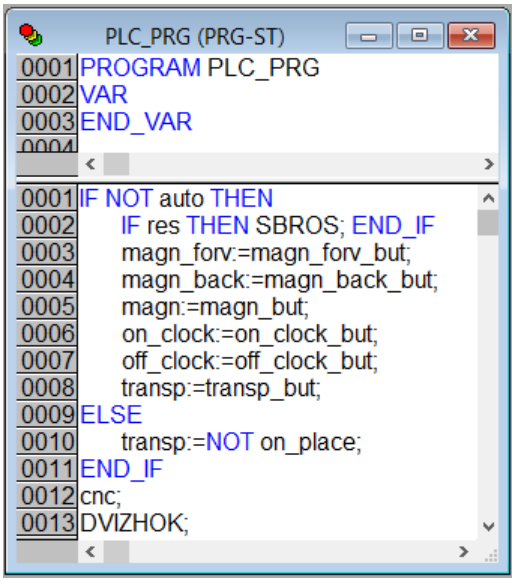


Рис. 86. Программа PLC_PRG

Программа сброса, рис. 87, предназначена для быстрого приведения системы в начальное состояние без сброса контроллера. Эта программа предназначена исключительно для отладки системы.

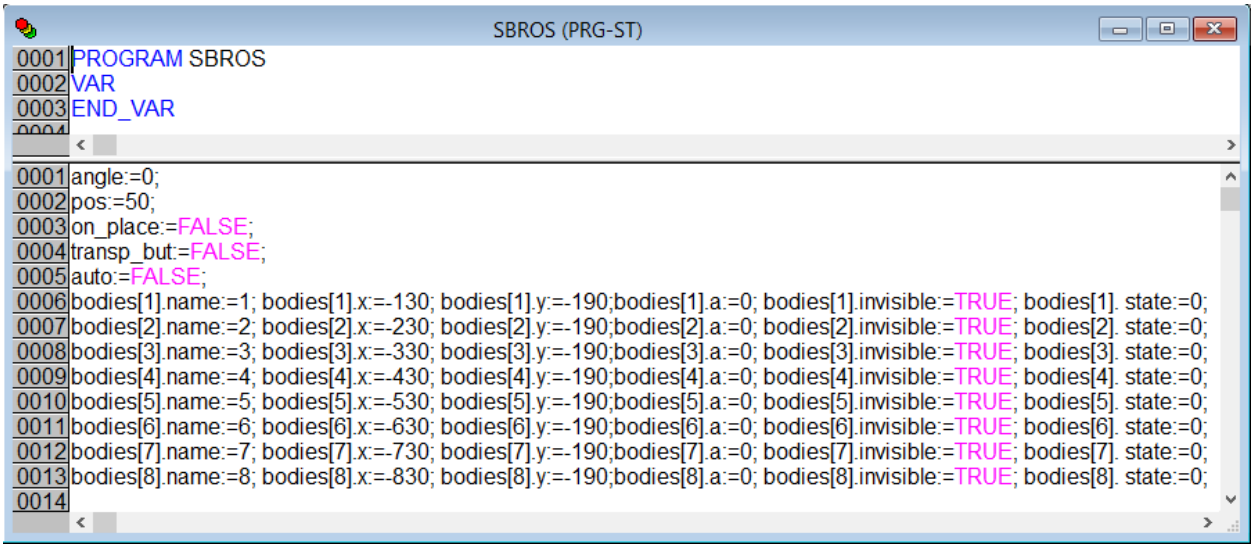


Рис. 87. Программа SBROS

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Руководство пользователя по программированию ПЛК в CoDeSys 2.3. – Режим доступа: https://www.owen.ru/product/ codesys_v2 /768.

2. Рыбалев А.Н. Имитационное моделирование АСУ ТП. – Благовещенск: Амурский гос. ун-т, 2019. – 408 с.

3. Лупал А.М. Теория автоматов: Учеб. пособие/СПбГУАП.СПб, 2000. 119 с.

Адрес редакции и издателя:
675027, г. Благовещенск, Игнатьевское шоссе, 21, комн. 406.

Адрес типографии:
675000, г. Благовещенск, ул. Мухина, 150а

Адрес учредителя:
ФБГОУ ВО «Амурский государственный университет»
675027, г. Благовещенск, Игнатьевское шоссе, 21.

Андрей Николаевич Рыбалёв,
доцент кафедры АППиЭ АмГУ, канд. техн. наук.

Имитационное моделирование АСУ ТП. *Монография.*

Издательство АмГУ. Подписано к печати __.03.2023. Дата выхода журнала в свет: __.03.2023. Редактор – *О.К. Мамонтова.* Компьютерная верстка – *Л.М. Пейзель*. Формат 60 х 84/16. Усл. печ. л. 3,68. Тираж __. Заказ ___. Бесплатно.

Отпечатано в типографии АмГУ.