Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра информационных систем и технологий

СПЕЦИАЛИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

**Методические указания по курсовому проектированию
по одноименной дисциплине для студентов специальности
1-40 05 01-03 «Информационные системы и технологии»
(издательско-полиграфический комплекс)**

Минск 2014

УДК 004.915 : 655(075.8)
ББК 32.97я73
С 71

Рассмотрены и рекомендованы к изданию редакционно-издательским советом Белорусского государственного технологического университета.

Составители:
С. И. Акунович, Т. П. Брусенцова

Рецензент
кандидат технических наук, доцент кафедры автоматизации производственных процессов и электротехники БГТУ
Н. П. Коровкина

По тематическому плану изданий учебно-методической литературы университета на 2013 год. Поз. 147.

Предназначены для студентов специальности 1-40 05 01-03 «Информационные системы и технологии» (издательско-полиграфический комплекс).

ПРЕДИСЛОВИЕ

Представленные методические указания могут быть использованы студентами специальности «Информационные системы и технологии» при выполнении курсового проекта по дисциплине «Специализированные информационные системы».

Цель курсового проекта – закрепление знаний, полученных при изучении дисциплины «Специализированные информационные системы», а также приобретение практических навыков разработки систем логического управления (СЛУ) с использованием современных технологий и инструментальных средств.

Проектирование любой управляющей системы всегда начинается с алгоритмического описания процесса управления. Лишь после четкого описания алгоритма, его проверки и отладки можно перейти к конструированию технической системы, реализующей этот алгоритм.

Ключевой задачей в решении проблемы логической отработки СЛУ является моделирование и проверка процессов ее функционирования на стадии проектирования, до начала программной или аппаратной реализации. Если в проекте СЛУ допущены и не выявлены ошибки, дальнейшая реализация может оказаться крайне затруднительной.

В настоящее время разработан и используется мощный арсенал средств для логической отработки исключительно широкого разнообразия СЛУ методами компьютерного моделирования. Выбор наиболее адекватных из этих средств, их адаптация и доработка применительно к конкретному классу СЛУ являются сложной практической задачей.

В настоящих методических указаниях решение этой задачи рассматривается в рамках совокупности курсовых проектов по следующим направлениям:

- Структурный анализ и логическое моделирование СЛУ в среде EXCEL;
- Разработка проектов СЛУ на языке LD;
- Разработка проектов СЛУ на языке ST;
- Разработка проектов СЛУ на языке SFC.

Данные методические указания представляют собой практический обзор наиболее важных тем и содержат ссылки на подробные руководства или материалы, в которых соответствующие вопросы проработаны более детально.

ВВЕДЕНИЕ

Перед началом курсового проектирования студент получает индивидуальное задание в виде темы курсового проекта и бланка задания. Темы курсовых проектов выдает преподаватель. В исключительных случаях возможен самостоятельный выбор темы по согласованию с преподавателем.

В процессе выполнения задания студент должен:

- провести поиск в литературных и интернет-источниках информации по теме проекта;
- выполнить обзор существующих решений задач по данному направлению, перечислить возможные пути решения;
- провести анализ программного средства (ПС) по заданию;
- выполнить проектирование ПС;
- оформить пояснительную записку к курсовому проекту, в которой описать (задокументировать) результаты работы по каждому из вышеперечисленных пунктов.

По окончании работы студент представляет отчет (пояснительную записку) и ПС, записанное на электронный носитель, с необходимыми для работы компонентами на проверку руководителю курсового проекта. После предварительной проверки студент допускается (или не допускается) к защите. На защите он докладывает преподавателю о проделанной работе и демонстрирует разработанное ПС, отвечает на вопросы, поясняет и обосновывает сущность своих решений. По результатам защиты выставляется оценка.

1. СТРУКТУРНЫЙ АНАЛИЗ И МОДЕЛИРОВАНИЕ СЛУ В СРЕДЕ EXCEL

Моделирование СЛУ выполняется для проверки правильности логических формул, обеспечивающих обработку входных сигналов (например от датчиков космического аппарата (КА)) и выдачу выходных сигналов на исполнительные механизмы КА.

Для этой цели разработчик переключает в нужной последовательности входные элементы. Excel вычисляет все логические формулы по событийному принципу и отображает новые состояния всех элементов, анализируя которые разработчик делает выводы о правильности логических формул или необходимости их исправления.

Рассмотрим описанный процесс на примере моделирования переключения функций памяти ZРОБИВК и ZЗТРЕВ в книгах Excel ИМС_КО и ЦПМ. Процесс моделирования состоит из следующих этапов:

- в книге Excel ИМС_КО определяем структурные связи между элементами (логическими формулами). Для включения ZРОБИВК нужно включить входной контакт Х011_17 (рис. 1);

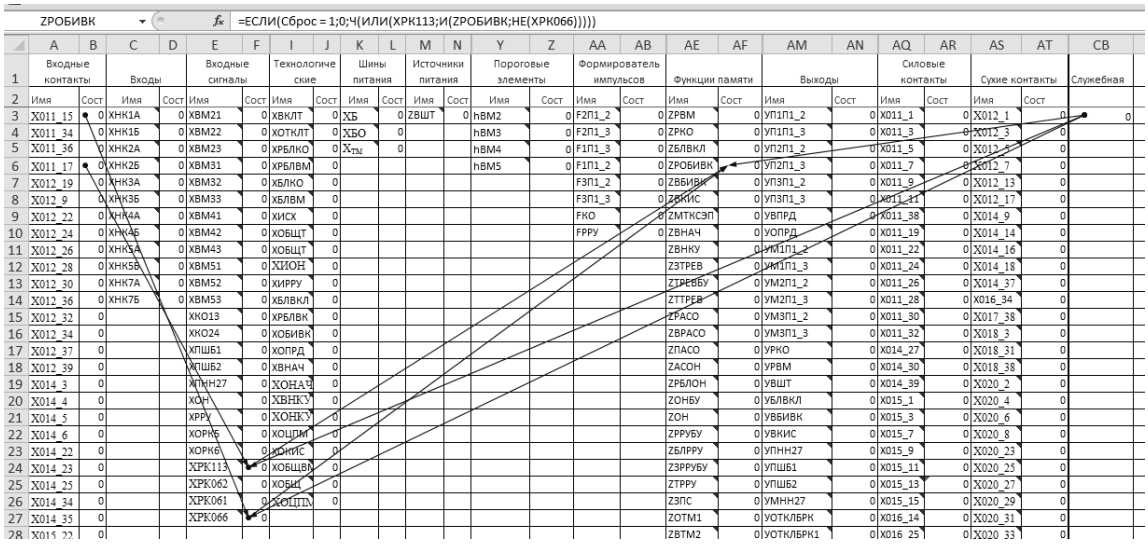
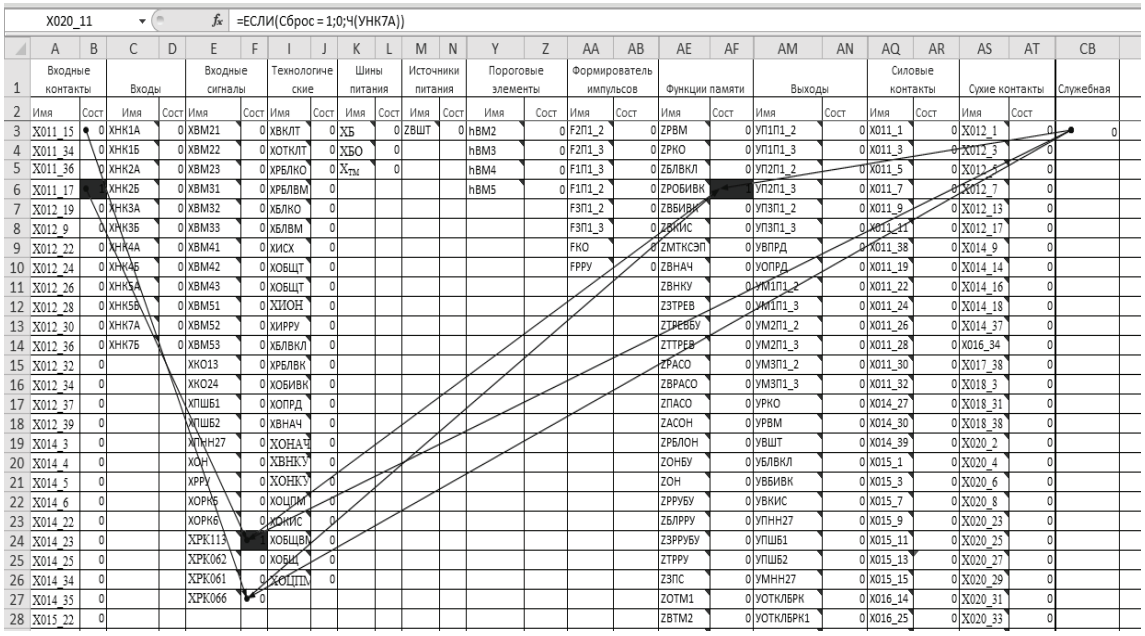


Рис. 1. Вид экрана для включения ZРОБИВК

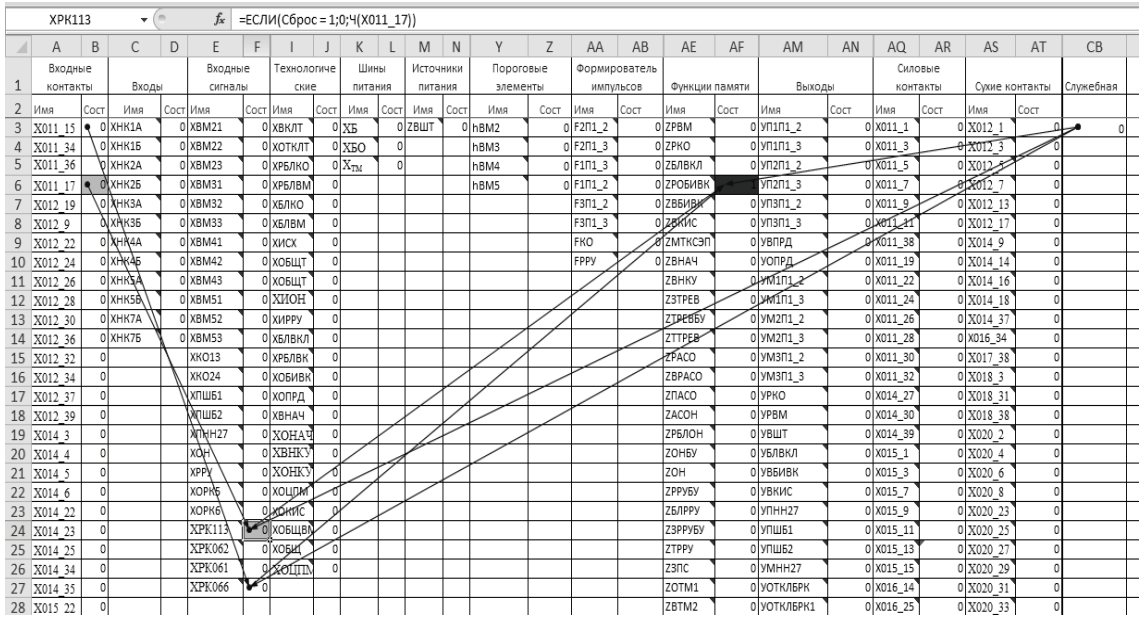
- включаем Х011_17, в результате чего включается ХРК113 и ZРОБИВК. Все переключения выделяем красным цветом (рис. 2);



	Входные контакты		Входы		Входные сигналы		Технологические		Шины питания		Источники питания		Пороговые элементы		Формирователь импульсов		Функции памяти		Выходы		Силовые контакты		Сухие контакты		Служебная
	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	
3	X011_15	0	XHK1A	0	XBM21	0	XBKЛТ	0	XБ	0	ZBШТ	0	hBM2	0	F2П1_2	0	ZPBM	0	УП1П1_2	0	X011_1	0	X012_1	0	0
4	X011_34	0	XHK1Б	0	XBM22	0	XOTKЛТ	0	XБО	0			hBM3	0	F2П1_3	0	ZPKO	0	УП1П1_3	0	X011_3	0	X012_3	0	
5	X011_36	0	XHK2A	0	XBM23	0	XPБЛКО	0	$X_{тм}$	0			hBM4	0	F1П1_3	0	ZБЛВКЛ	0	УП2П1_2	0	X011_5	0	X012_5	0	
6	X011_17		XHK2Б	0	XBM31	0	XPБЛВМ	0					hBM5	0	F1П1_2	0	ZPOБИВК		УП2П1_3	0	X011_7	0	X012_7	0	
7	X012_19	0	XHK3A	0	XBM32	0	XБЛКО	0							F3П1_2	0	ZBБИВК	0	УП3П1_2	0	X011_9	0	X012_13	0	
8	X012_9	0	XHK3Б	0	XBM33	0	XБЛВМ	0							F3П1_3	0	ZBKИС	0	УП3П1_3	0	X011_11	0	X012_17	0	
9	X012_22	0	XHK4A	0	XBM41	0	XИСХ	0							FKO	0	ZMTKCЭП	0	УВПРД	0	X011_38	0	X014_9	0	
10	X012_24	0	XHK4Б	0	XBM42	0	XOБЩТ	0							FPPУ	0	ZBHAЧ	0	УОПРД	0	X011_19	0	X014_14	0	
11	X012_26	0	XHK5A	0	XBM43	0	XOБЩТ	0									ZBHKУ	0	УМ1П1_2	0	X011_22	0	X014_16	0	
12	X012_28	0	XHK5Б	0	XBM51	0	XИОН	0									Z3TPEB	0	УМ1П1_3	0	X011_24	0	X014_18	0	
13	X012_30	0	XHK7A	0	XBM52	0	XИРРУ	0									ZTPEBБУ	0	УМ2П1_2	0	X011_26	0	X014_37	0	
14	X012_36	0	XHK7Б	0	XBM53	0	XБЛВКЛ	0									ZTTPEB	0	УМ2П1_3	0	X011_28	0	X016_34	0	
15	X012_32	0			XKO13	0	XPБЛВК	0									ZPACO	0	УМ3П1_2	0	X011_30	0	X017_38	0	
16	X012_34	0			XKO24	0	XOБИВК	0									ZBPACO	0	УМ3П1_3	0	X011_32	0	X018_3	0	
17	X012_37	0			XПШБ1	0	XOПРД	0									ZПACO	0	УРКО	0	X014_27	0	X018_31	0	
18	X012_39	0			XПШБ2	0	XBHAЧ	0									ZACOH	0	УРВМ	0	X014_30	0	X018_38	0	
19	X014_3	0			XПНН27	0	XOHAЧ	0									ZPБЛОН	0	УВШТ	0	X014_39	0	X020_2	0	
20	X014_4	0			XOH	0	XBHKУ	0									ZOHБУ	0	УБЛВКЛ	0	X015_1	0	X020_4	0	
21	X014_5	0			XPPУ	0	XOHKУ	0									ZOH	0	УВБИВК	0	X015_3	0	X020_6	0	
22	X014_6	0			XOPK5	0	XOЦПМ	0									ZPPУБУ	0	УВКИС	0	X015_7	0	X020_8	0	
23	X014_22	0			XOPK6	0	XOKИС	0									ZБЛРРУ	0	УПНН27	0	X015_9	0	X020_23	0	
24	X014_23	0			XPK113		XOБЩВМ	0									Z3PPУБУ	0	УПШБ1	0	X015_11	0	X020_25	0	
25	X014_25	0			XPK062	0	XOБЩ	0									ZTPPУ	0	УПШБ2	0	X015_13	0	X020_27	0	
26	X014_34	0			XPK061	0	XOЦПМ	0									Z3ПС	0	УМНН27	0	X015_15	0	X020_29	0	
27	X014_35	0			XPK066	0											ZOTM1	0	УОТКЛБРК	0	X016_14	0	X020_31	0	
28	X015_22	0															ZBTM2	0	УОТКЛБРК1	0	X016_25	0	X020_33	0	

Рис. 2. Вид экрана включения переключателей

• выключаем входной контакт X011_17, но ZPOБИВК остается включенным (рис. 3);



	Входные контакты		Входы		Входные сигналы		Технологические		Шины питания		Источники питания		Пороговые элементы		Формирователь импульсов		Функции памяти		Выходы		Силовые контакты		Сухие контакты		Служебная
	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	
3	X011_15	0	XHK1A	0	XBM21	0	XBKЛТ	0	XБ	0	ZBШТ	0	hBM2	0	F2П1_2	0	ZPBM	0	УП1П1_2	0	X011_1	0	X012_1	0	0
4	X011_34	0	XHK1Б	0	XBM22	0	XOTKЛТ	0	XБО	0			hBM3	0	F2П1_3	0	ZPKO	0	УП1П1_3	0	X011_3	0	X012_3	0	
5	X011_36	0	XHK2A	0	XBM23	0	XPБЛКО	0	$X_{тм}$	0			hBM4	0	F1П1_3	0	ZБЛВКЛ	0	УП2П1_2	0	X011_5	0	X012_5	0	
6	X011_17	0	XHK2Б	0	XBM31	0	XPБЛВМ	0					hBM5	0	F1П1_2	0	ZPOБИВК	1	УП2П1_3	0	X011_7	0	X012_7	0	
7	X012_19	0	XHK3A	0	XBM32	0	XБЛКО	0							F3П1_2	0	ZBБИВК	0	УП3П1_2	0	X011_9	0	X012_13	0	
8	X012_9	0	XHK3Б	0	XBM33	0	XБЛВМ	0							F3П1_3	0	ZBKИС	0	УП3П1_3	0	X011_11	0	X012_17	0	
9	X012_22	0	XHK4A	0	XBM41	0	XИСХ	0							FKO	0	ZMTKCЭП	0	УВПРД	0	X011_38	0	X014_9	0	
10	X012_24	0	XHK4Б	0	XBM42	0	XOБЩТ	0							FPPУ	0	ZBHAЧ	0	УОПРД	0	X011_19	0	X014_14	0	
11	X012_26	0	XHK5A	0	XBM43	0	XOБЩТ	0									ZBHKУ	0	УМ1П1_2	0	X011_22	0	X014_16	0	
12	X012_28	0	XHK5Б	0	XBM51	0	XИОН	0									Z3TPEB	0	УМ1П1_3	0	X011_24	0	X014_18	0	
13	X012_30	0	XHK7A	0	XBM52	0	XИРРУ	0									ZTPEBБУ	0	УМ2П1_2	0	X011_26	0	X014_37	0	
14	X012_36	0	XHK7Б	0	XBM53	0	XБЛВКЛ	0									ZTTPEB	0	УМ2П1_3	0	X011_28	0	X016_34	0	
15	X012_32	0			XKO13	0	XPБЛВК	0									ZPACO	0	УМ3П1_2	0	X011_30	0	X017_38	0	
16	X012_34	0			XKO24	0	XOБИВК	0									ZBPACO	0	УМ3П1_3	0	X011_32	0	X018_3	0	
17	X012_37	0			XПШБ1	0	XOПРД	0									ZПACO	0	УРКО	0	X014_27	0	X018_31	0	
18	X012_39	0			XПШБ2	0	XBHAЧ	0									ZACOH	0	УРВМ	0	X014_30	0	X018_38	0	
19	X014_3	0			XПНН27	0	XOHAЧ	0									ZPБЛОН	0	УВШТ	0	X014_39	0	X020_2	0	
20	X014_4	0			XOH	0	XBHKУ	0									ZOHБУ	0	УБЛВКЛ	0	X015_1	0	X020_4	0	
21	X014_5	0			XPPУ	0	XOHKУ	0									ZOH	0	УВБИВК	0	X015_3	0	X020_6	0	
22	X014_6	0			XOPK5	0	XOЦПМ	0									ZPPУБУ	0	УВКИС	0	X015_7	0	X020_8	0	
23	X014_22	0			XOPK6	0	XOKИС	0									ZБЛРРУ	0	УПНН27	0	X015_9	0	X020_23	0	
24	X014_23	0			XPK113	0	XOБЩВМ	0									Z3PPУБУ	0	УПШБ1	0	X015_11	0	X020_25	0	
25	X014_25	0			XPK062	0	XOБЩ	0									ZTPPУ	0	УПШБ2	0	X015_13	0	X020_27	0	
26	X014_34	0			XPK061	0	XOЦПМ	0									Z3ПС	0	УМНН27	0	X015_15	0	X020_29	0	
27	X014_35	0			XPK066	0											ZOTM1	0	УОТКЛБРК	0	X016_14	0	X020_31	0	
28	X015_22	0															ZBTM2	0	УОТКЛБРК1	0	X016_25	0	X020_33	0	

Рис. 3. Вид экрана при выключенном входном контакте X011_17

• для выключения ZPOБИВК включаем входной контакт X011_15 (рис. 4);

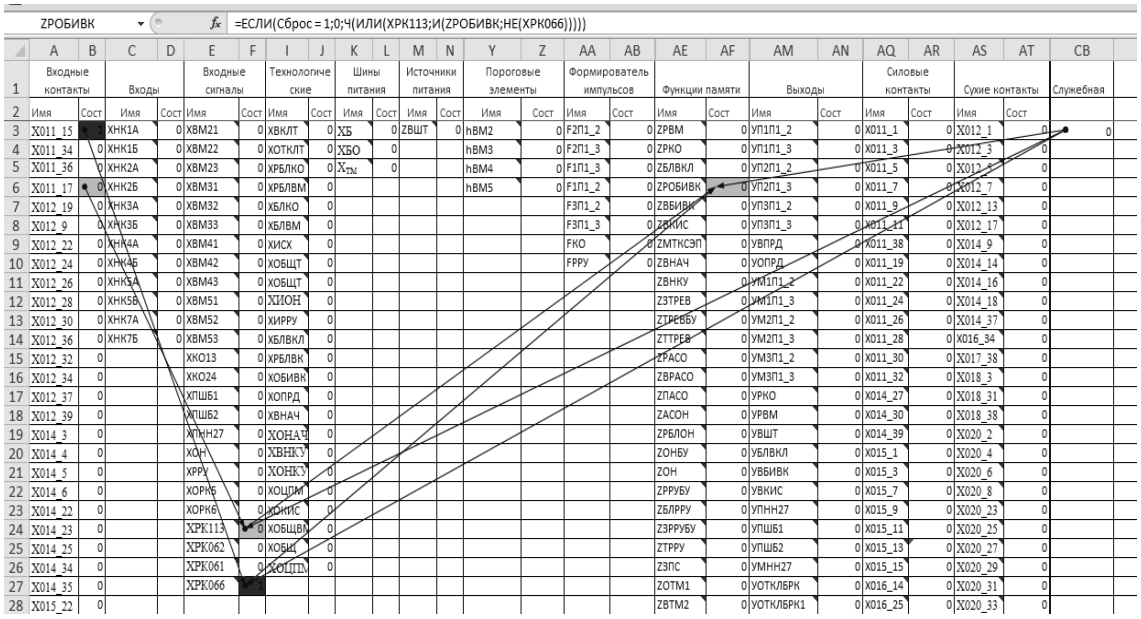


Рис. 4. Вид экрана выключения ZPOБИВК

- выключаем входной контакт X011_15 (рис. 5);

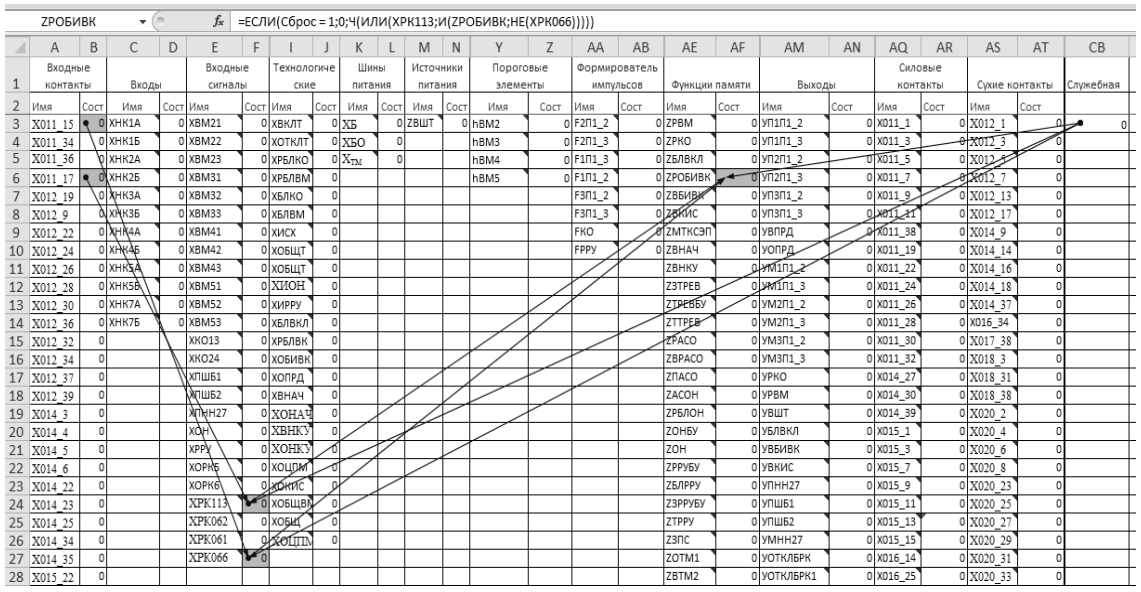


Рис. 5. Вид экрана при выключенном
входном контакте X011_15

- функция памяти Z3TPEB имеет внешнюю ссылку, по которой переходим в книгу ЦПМ (рис. 6);

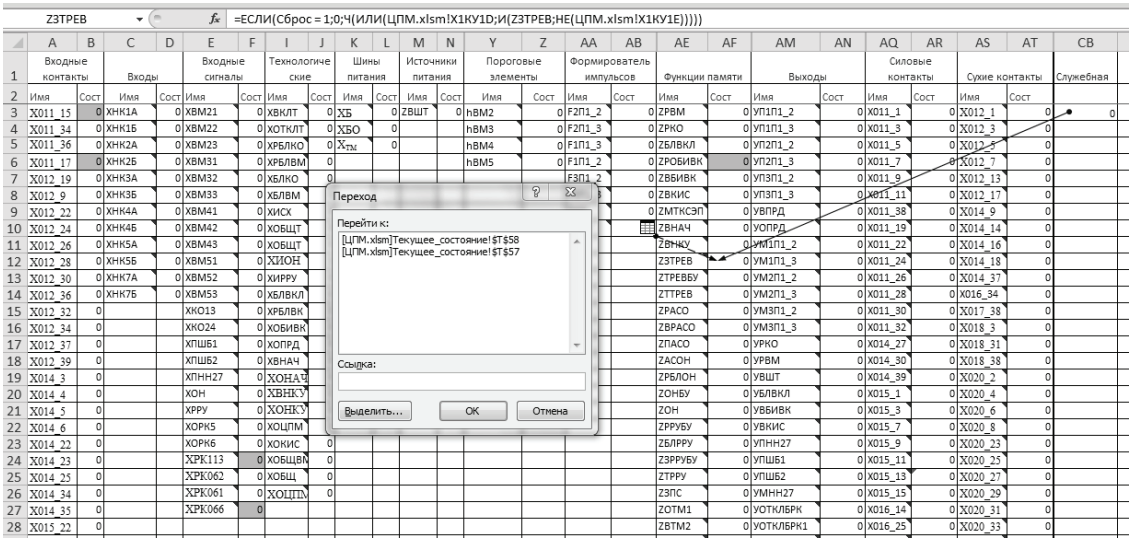


Рис. 6. Вид экрана перехода в книгу ЦПМ

- В ЦПМ включаем X1KУ1D (рис. 7);

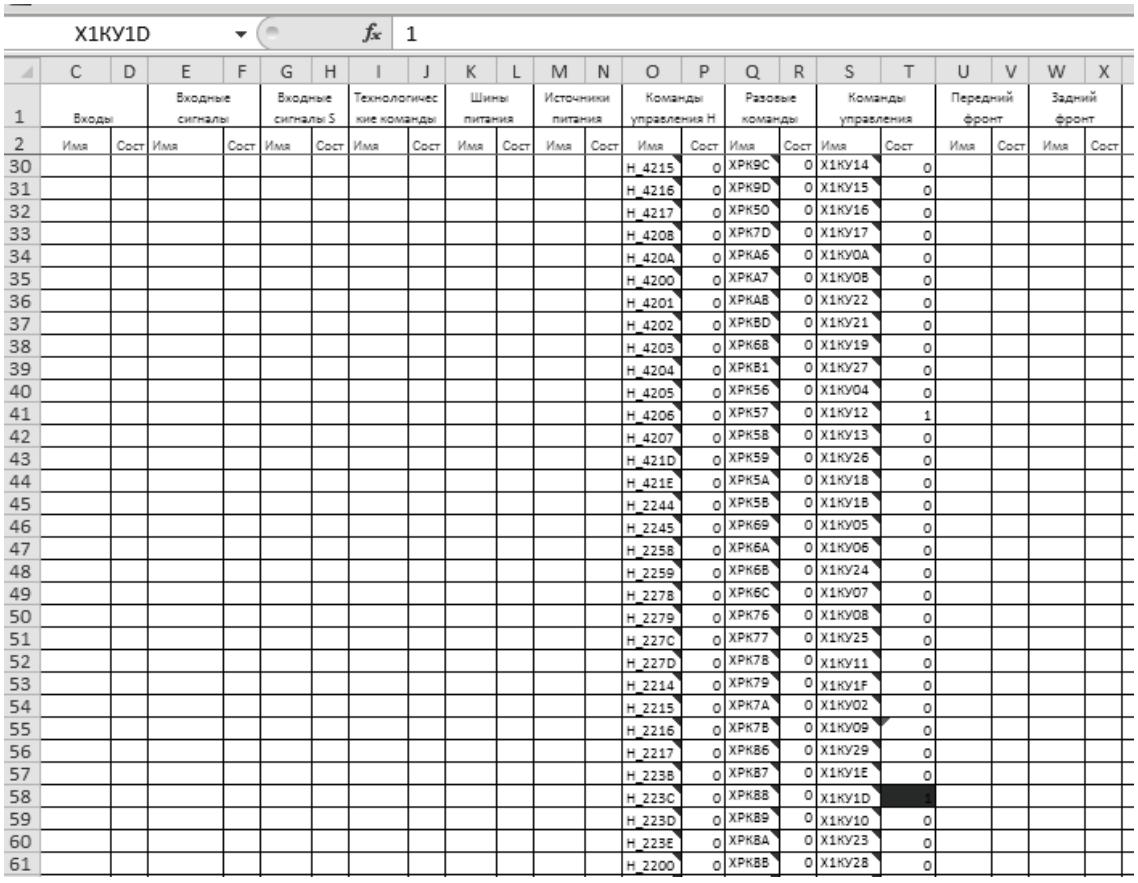
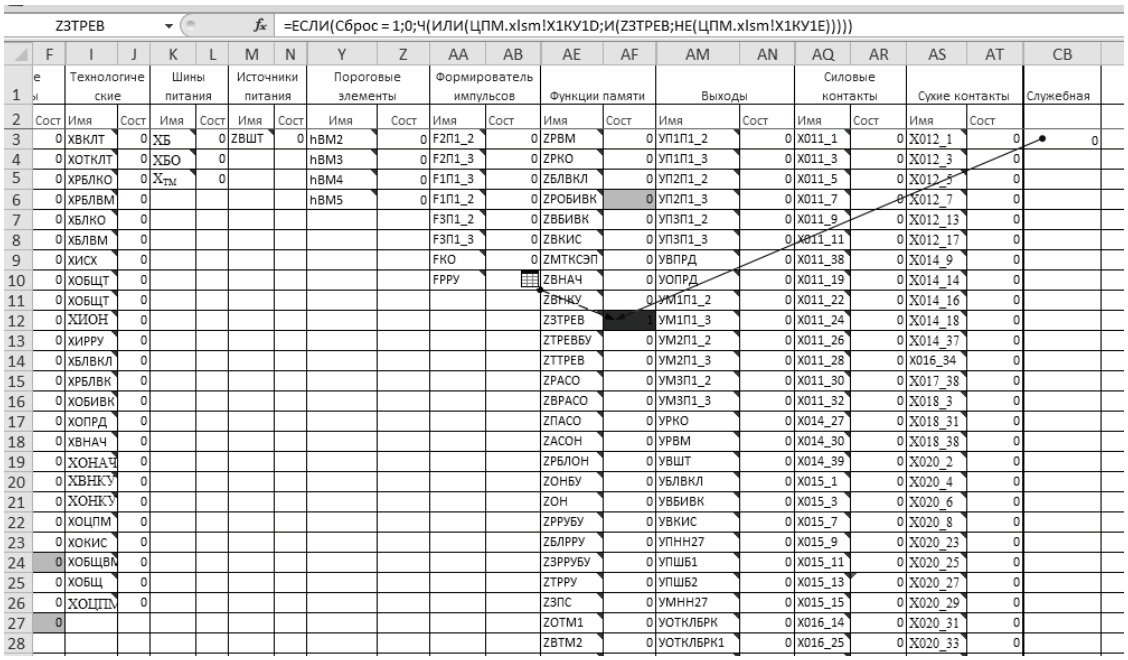


Рис. 7. Вид экрана при включенном X1KУ1D

- В результате этого в ИМС_КО включается Z3TPEB (рис. 8).



	F	I	J	K	L	M	N	Y	Z	AA	AB	AE	AF	AM	AN	AQ	AR	AS	AT	CB
1	е	Технологические		Шины питания		Источники питания		Пороговые элементы		Формирователь импульсов		Функции памяти		Выходы		Силовые контакты		Сухие контакты		Служебная
2	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	
3	0	ХВКЛТ	0	ХБ	0	ZBШТ	0	hBM2	0	F2П1_2	0	ZPBM	0	УП1П1_2	0	X011_1	0	X012_1	0	0
4	0	ХОТКЛТ	0	ХБО	0			hBM3	0	F2П1_3	0	ZPKO	0	УП1П1_3	0	X011_3	0	X012_3	0	
5	0	ХРБЛКО	0	$X_{тм}$	0			hBM4	0	F1П1_3	0	ZБЛВКЛ	0	УП2П1_2	0	X011_5	0	X012_5	0	
6	0	ХРБЛВМ	0					hBM5	0	F1П1_2	0	ZPOБИВК	0	УП2П1_3	0	X011_7	0	X012_7	0	
7	0	ХБЛКО	0							F3П1_2	0	ZBБИВК	0	УП3П1_2	0	X011_9	0	X012_13	0	
8	0	ХБЛВМ	0							F3П1_3	0	ZBKИС	0	УП3П1_3	0	X011_11	0	X012_17	0	
9	0	ХИСХ	0							FKO	0	ZMTKСЭП	0	УВПРД	0	X011_38	0	X014_9	0	
10	0	ХОБЩТ	0							FPPУ		ZBНАЧ	0	УОПРД	0	X011_19	0	X014_14	0	
11	0	ХОБЩТ	0									ZBНКУ	0	УМ1П1_2	0	X011_22	0	X014_16	0	
12	0	ХИОН	0									Z3TPEB	0	УМ1П1_3	0	X011_24	0	X014_18	0	
13	0	ХИРРУ	0									ZTPEBБУ	0	УМ2П1_2	0	X011_26	0	X014_37	0	
14	0	ХБЛВКЛ	0									ZTTPEB	0	УМ2П1_3	0	X011_28	0	X016_34	0	
15	0	ХРБЛВК	0									ZPACO	0	УМ3П1_2	0	X011_30	0	X017_38	0	
16	0	ХОБИВК	0									ZBPACO	0	УМ3П1_3	0	X011_32	0	X018_3	0	
17	0	ХОПРД	0									ZПАСО	0	УРКО	0	X014_27	0	X018_31	0	
18	0	ХВНАЧ	0									ZACOH	0	УРВМ	0	X014_30	0	X018_38	0	
19	0	ХОНАЧ	0									ZPБЛОН	0	УВШТ	0	X014_39	0	X020_2	0	
20	0	ХВНКУ	0									ZOHБУ	0	УБЛВКЛ	0	X015_1	0	X020_4	0	
21	0	ХОНКУ	0									ZOH	0	УВБИВК	0	X015_3	0	X020_6	0	
22	0	ХОЦПМ	0									ZPPУБУ	0	УВКИС	0	X015_7	0	X020_8	0	
23	0	ХОКИС	0									ZБЛРРУ	0	УПНН27	0	X015_9	0	X020_23	0	
24	0	ХОБЩВМ	0									Z3PPУБУ	0	УПШБ1	0	X015_11	0	X020_25	0	
25	0	ХОБЩ	0									ZTPPУ	0	УПШБ2	0	X015_13	0	X020_27	0	
26	0	ХОЦПМ	0									Z3ПС	0	УМНН27	0	X015_15	0	X020_29	0	
27	0											ZOTM1	0	УОТКЛБРК	0	X016_14	0	X020_31	0	
28												ZBTM2	0	УОТКЛБРК1	0	X016_25	0	X020_33	0	

Рис. 8. Вид экрана ИМС_КО

Таким способом выполняется проверка правильности логических формул.

2. РАЗРАБОТКА ПРОЕКТОВ СЛУ НА ЯЗЫКЕ LD

2.1. Описание языка LD

***Язык* LD** – *графический* язык, применяется для описания логических выражений различного уровня сложности. Он содержит *контакты* (входные аргументы) и *катушки* (выходные переменные). Элементы организуются в *сеть* релейно-контактных схем. При необходимости можно реализовывать более сложную логику, используя функции и функциональные блоки.

Каждому контакту ставится в соответствие логическая переменная, определяющая его состояние. Ее имя ставится над контактом и служит его названием. Если контакт замкнут, то переменная имеет значение *true*, если разомкнут – *false*. *Последовательное соединение* контактов или цепей соответствует логической операции И/AND, *параллельное* – ИЛИ/OR. Нормально *замкнутый* (*инверсный*) контакт равнозначен логической операции НЕ/NOT.

Графические представления элементов языка LD и их текстовые аналоги в среде разработки IsaGraf 6 приведены на рис. 9–25.

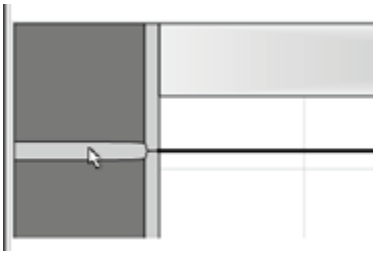


Рис. 9. Цепи

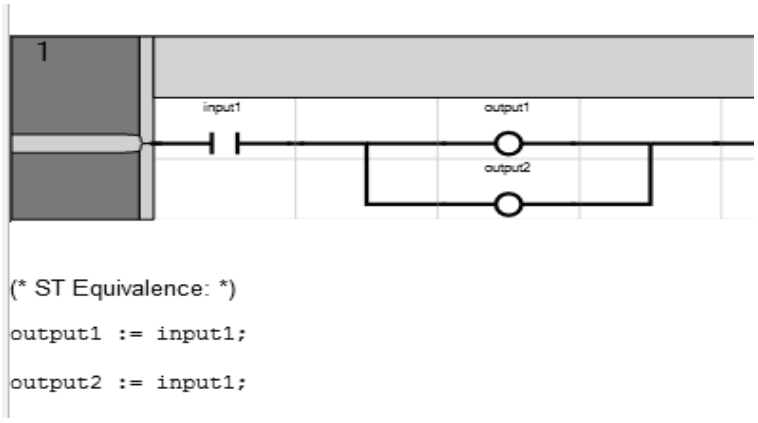
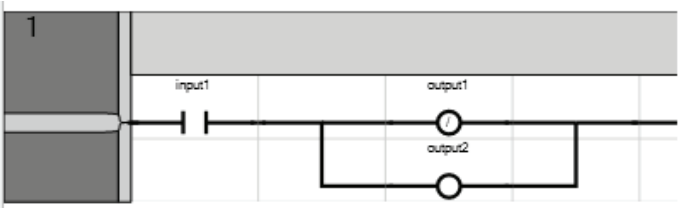
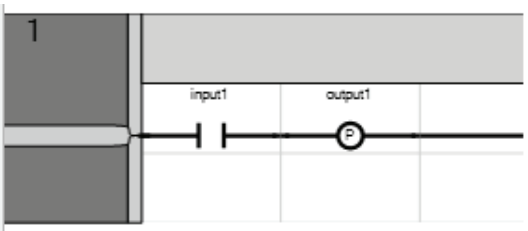


Рис. 10. Прямая обмотка



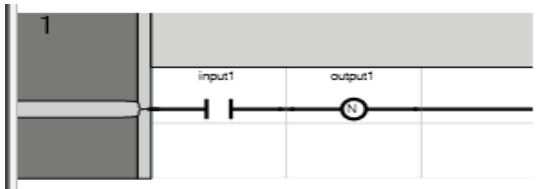

```
(* ST Equivalence: *)
output1 := NOT (input1);
output2 := input1;
```

Рис. 11. Инверсная обмотка



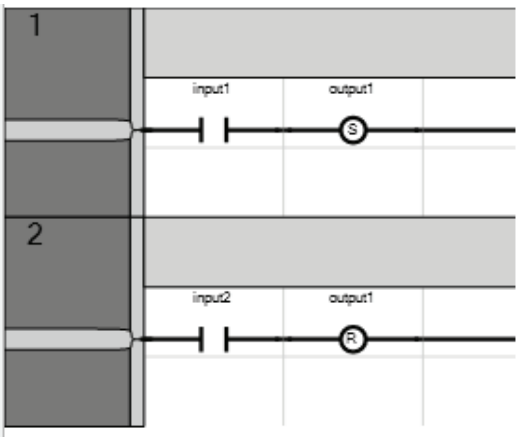

```
(* ST Equivalence: *)
IF (input1 and NOT(input1prev)) THEN
output1 := TRUE;
ELSE
output1 := FALSE;
END_IF;
(* input1prev is the value of input1 at the previous cycle *)
```

Рис. 12. Импульсная обмотка (Передний фронт)



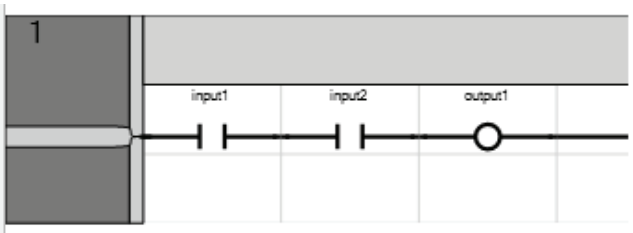

```
(* ST Equivalence: *)
IF (NOT(input1) and input1prev) THEN
output1 := TRUE;
ELSE
output1 := FALSE;
END_IF;
(* input1prev is the value of input1 at the previous cycle *)
```

Рис. 13. Импульсная обмотка (Задний фронт)



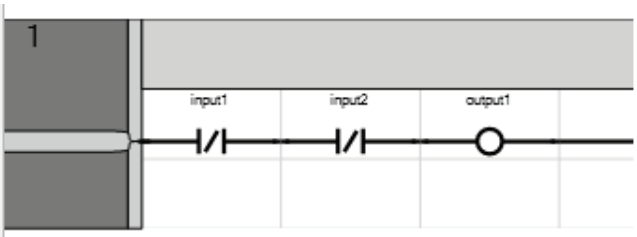

```
(* ST Equivalence: *)
IF input1 THEN
output1 := TRUE;
END_IF;
IF input2 THEN
output1 := FALSE;
END_IF;
```

Рис. 14. Включающая обмотка (Set) и Выключающая обмотка (Reset)



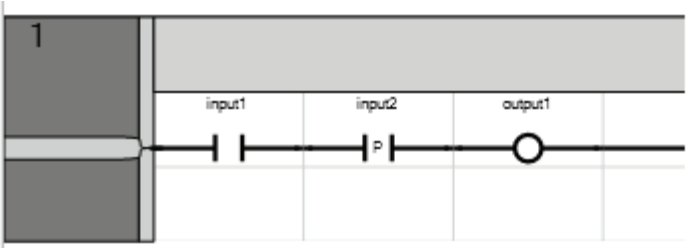

```
(* ST Equivalence: *)
output1 := input1 AND input2;
```

Рис. 15. Прямой контакт



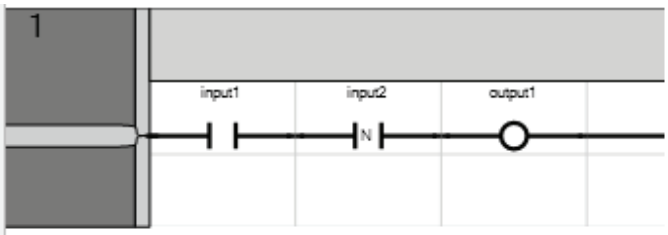

```
(* ST Equivalence: *)
output1 := NOT (input1) AND NOT (input2);
```

Рис. 16. Инверсный контакт



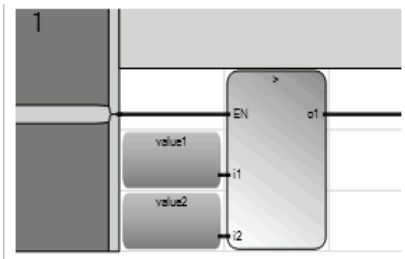

```
(* ST Equivalence: *)
output1 := input1 AND (input2 AND NOT (input2prev));
(* input2prev is the value of input2 at the previous cycle *)
```

Рис. 17. Импульсный контакт (Передний фронт)



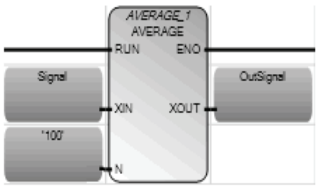

```
(* ST Equivalence: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(* input2prev is the value of input2 at the previous cycle *)
```

Рис. 18. Импульсный контакт (Задний фронт)




```
IF rung_state THEN
q := (value1 > value 2);
ELSE
q := FALSE;
END_IF;
(* continue rung with q state *)
```

Рис. 19. Блоки. EN вход




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(* continue rung with eno state *)
```

Рис. 20. Блоки. ENO выход

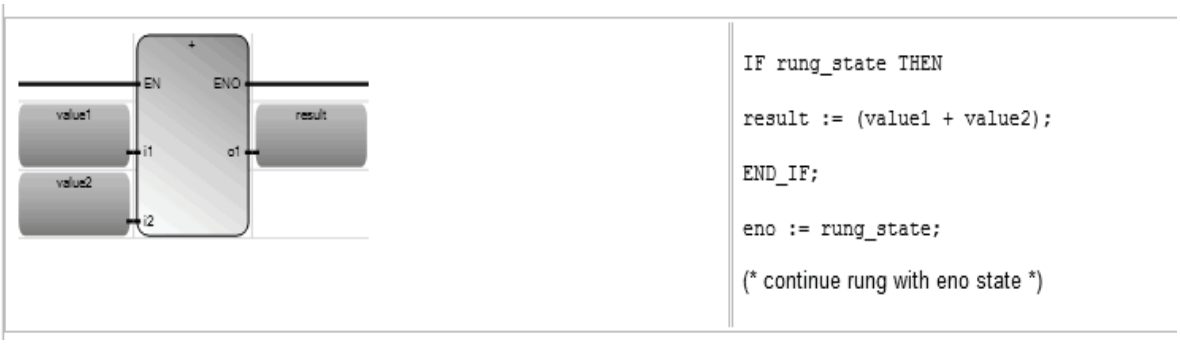


Рис. 21. Блоки. EN и ENO параметры

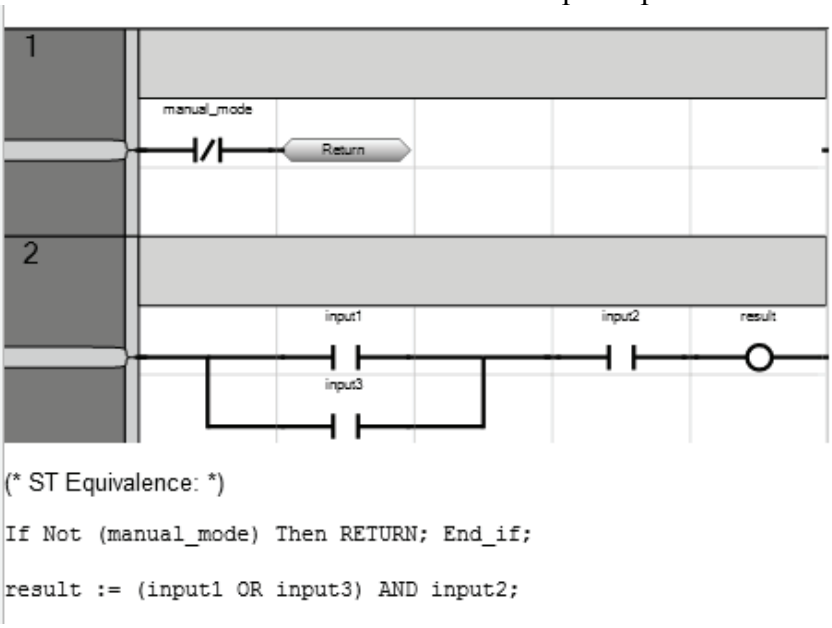


Рис. 22. Возвраты

Если линия подключения (слева от символа Return) имеет состояние TRUE, программа завершается – никакая дальнейшая часть диаграммы не выполняется.

Никакое подключение не может быть произведено справа от символа RETURN.

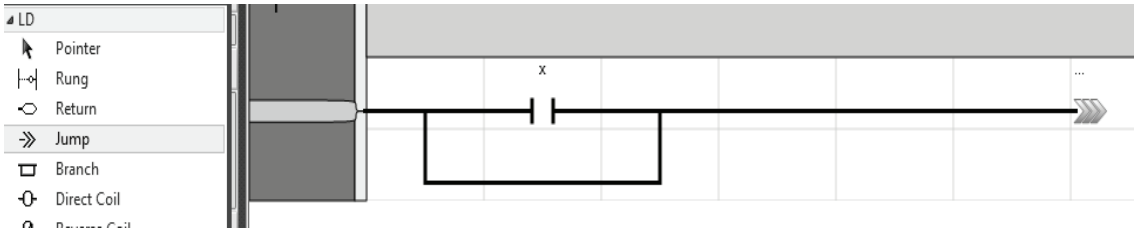


Рис. 23. Переходы и метки

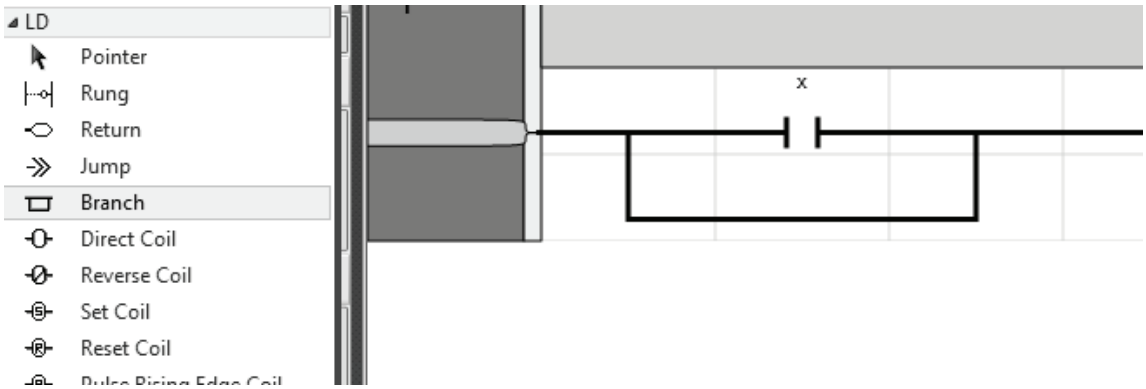


Рис. 24. Разветвления

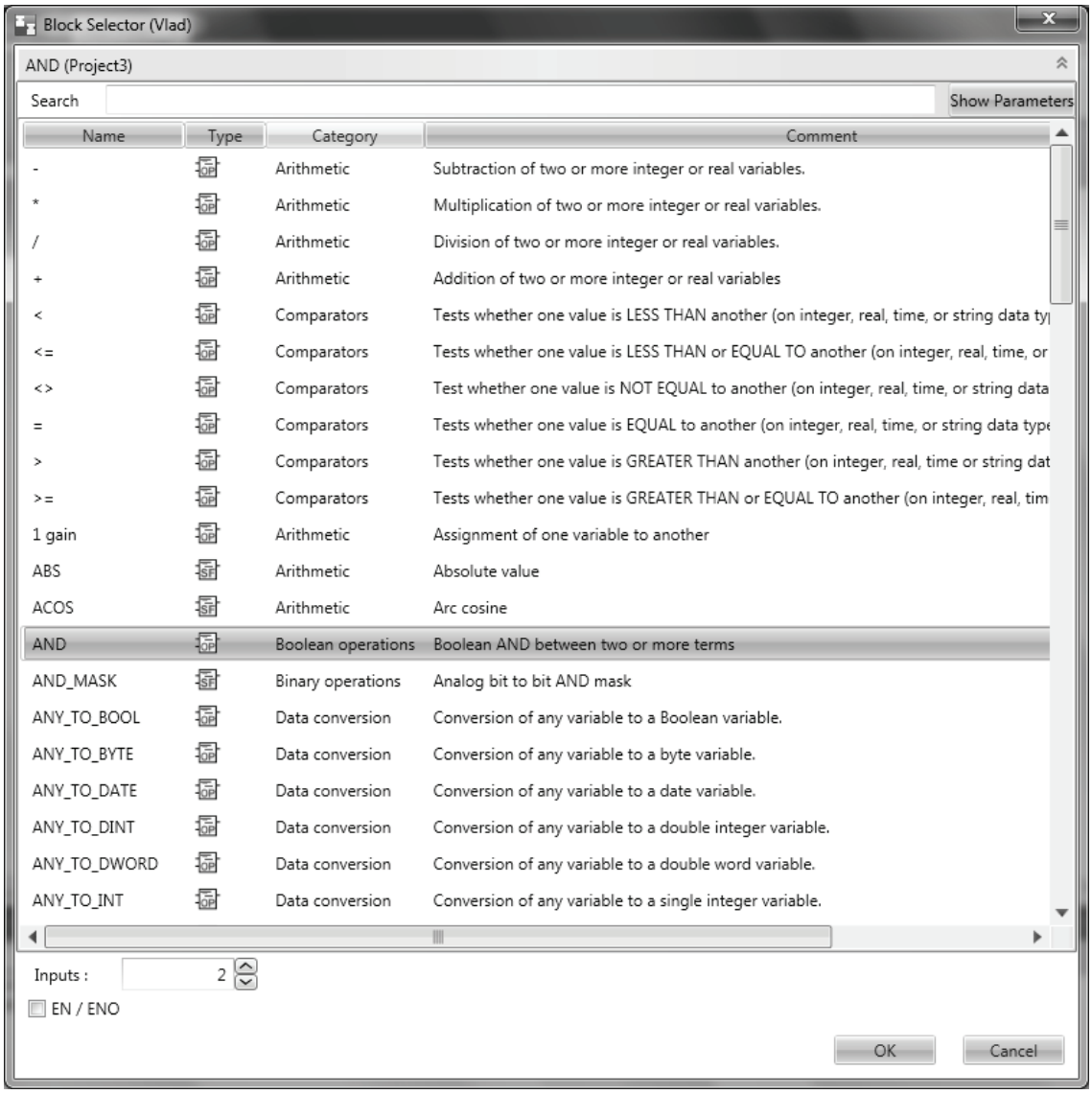


Рис. 25. Функции и функциональные блоки

2.2. Разработка программ на языке LD

2.2.1. Описание задания

Разработать программу на языке LD для моделирования логических функций, представленных в таблице Excel и зависящих от входного контакта X012_28. Выявим ячейки, зависимые от ячейки X012_28 (рис. 26).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Входные контакты		Технологические команды		Функции памяти		Выходы		Выходные контакты		Силовые контакты		Сухие контакты		Электронные ключи	
2	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост	Имя	Сост
3	X011_15	0	ХВКЛТ	0	ZPBM	0	УП1П1_2	0			X011_1	0	X012_1	0		
4	X011_34	0	ХОТКЛТ	0	ZPKO	0	УП1П1_3	0			X011_3	0	X012_3	0		
5	X011_36	0	ХРБЛКО	0	ZБЛВКЛ	0	УП2П1_2	0			X011_5	0	X012_5	0		
6	X011_17	0	ХРБЛВ	0	ZPOБИВК	0	УП2П1_3	0			X011_7	0	X012_7	0		
7	X012_19	0	ХБЛКО	0	ZBБИВК	0	УП3П1_2	0			X011_9	0	X012_13	0		
8	X012_9	0	ХБЛВМ	0	ZBKИС	0	УП3П1_3	0			X011_11	0	X012_17	0		
9	X012_22	0	ХИСХ	0	ZMTKСЭП	0	УВПРД	0			X011_38	0	X014_9	0		
10	X012_28	0	ХИОН	0	Z3TPEB	0	УМ1П1_3	0			X011_24	0	X014_18	0		
11	X012_32	0	ХРБЛВК	0	ZPACO	1	УМ3П1_2	0			X011_30	0	X017_38	0		
12	X014_4	0	ХВНКУ	0	ZOНБУ	0	УБЛВКЛ	0			X015_1	0	X020_4	0		
13	X014_5	0	ХОНКУ	0	ZOH	0	УВБИВК	0			X015_3	0	X020_6	0		
14	X016_10	0					УМТКСЭП	0			X018_18	0	X022_28	0		
15	X016_30	0					УРАСО	0			X021_22	0	X022_24	0		
16	X020_30	0					УБИВКП	0								
17							УРОБИВКТМ	0								
18							УВБИВКТМ	0								

Рис. 26. Моделирование логических функций

2.2.2. Структура проекта

Создадим проект типа Simulator, в котором будет располагаться две программы типа Ladder Diagram (рис. 27).

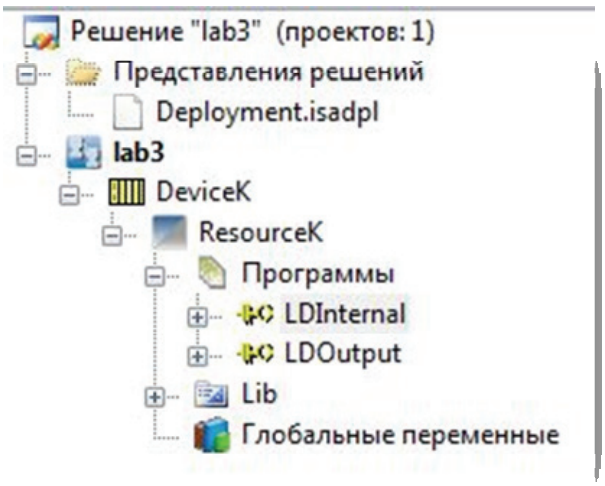


Рис. 27. Структура проекта

2.2.3. Разработка программ

В программе LD Internal (рис. 28) отображены внутренние функции. Функции памяти представлены двумя витками: S – включение, R – выключение.

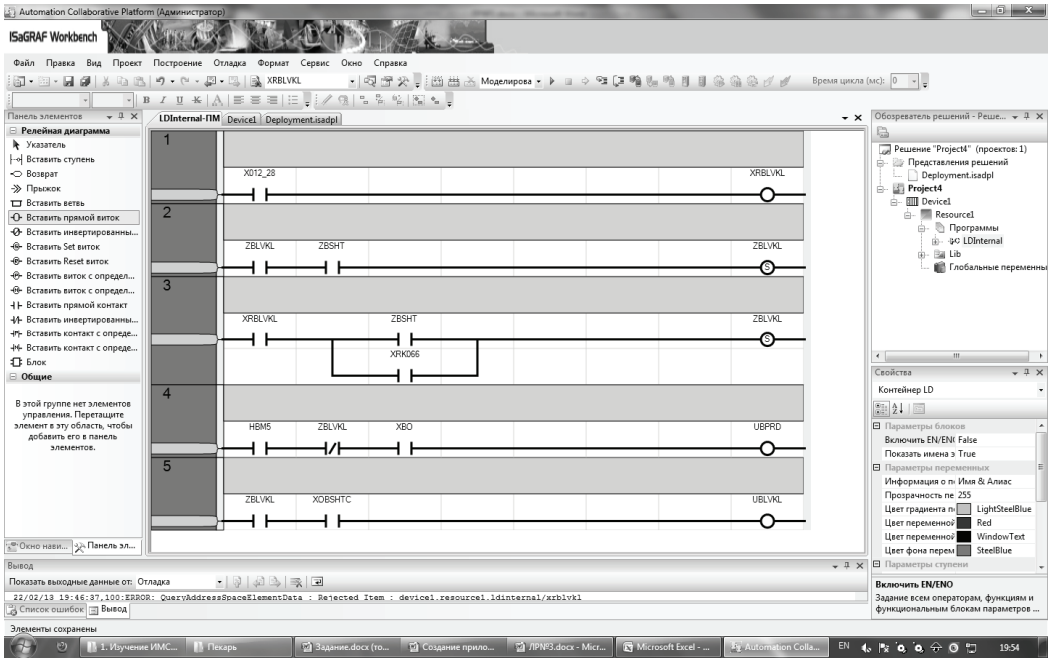


Рис. 28. Программа LDInternal

Программа LDOutput (рис. 29) отображает выходные функции:

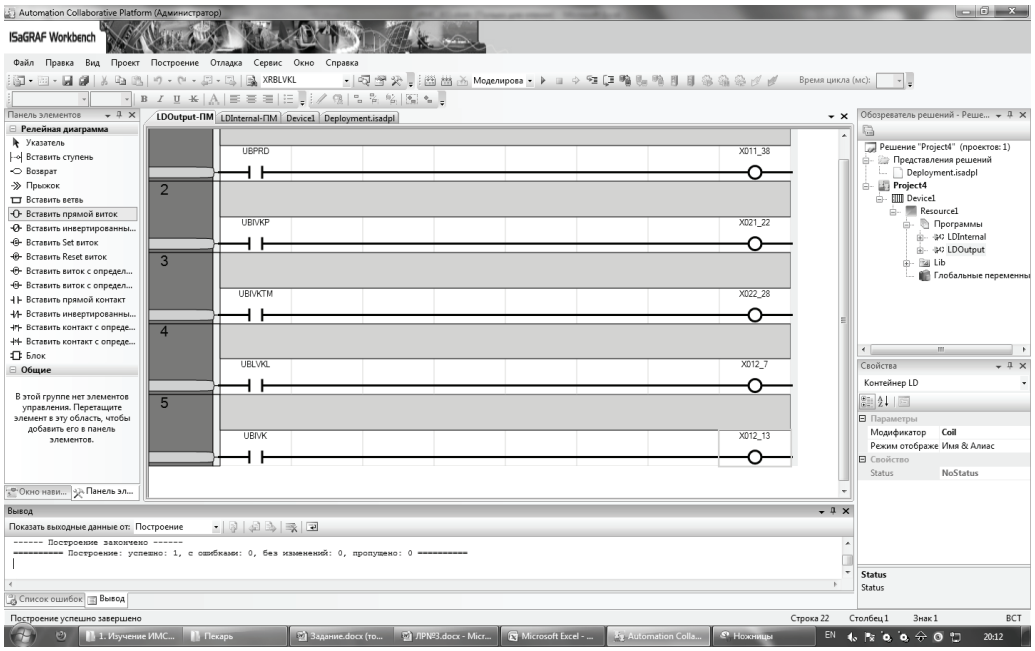


Рис. 29. Подпрограмма LDOutput

2.2.4. Монтирование переменных

Окно устройства ввода-вывода отображает добавленные устройства, на нем также можно добавить новые устройства (рис. 30).

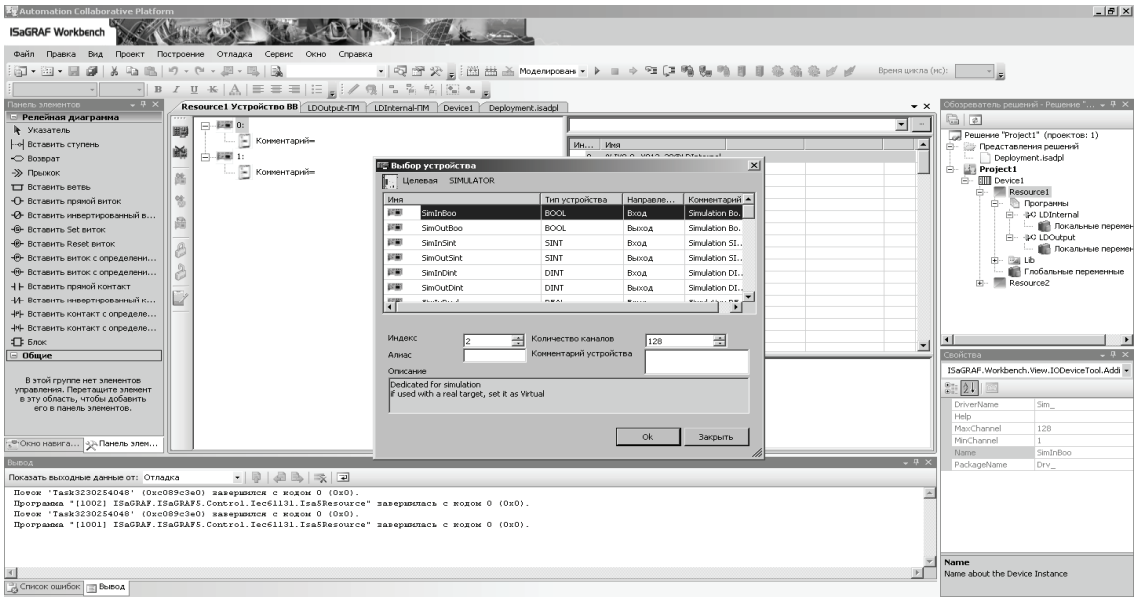


Рис. 30. Добавление устройства

На рис. 31 отображены добавленные виртуальные устройства.

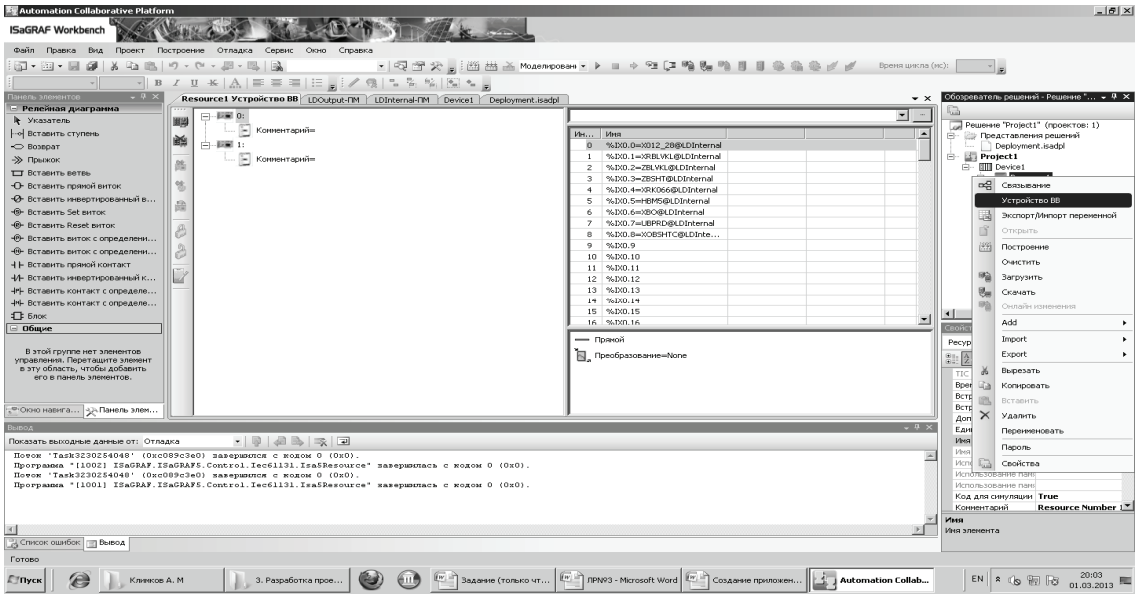
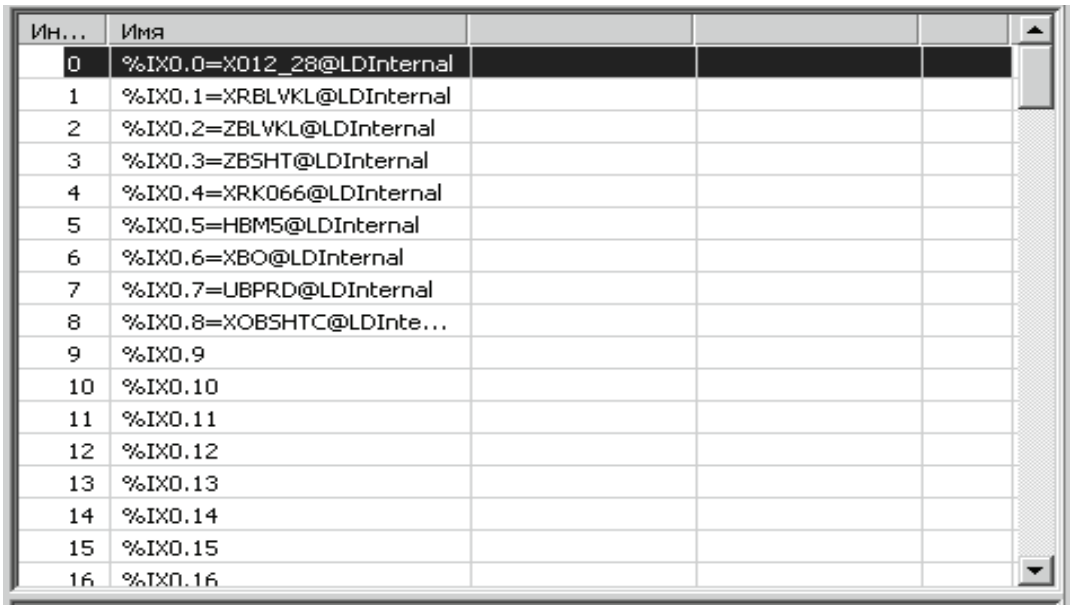


Рис. 31. Добавленные устройства

В области справа от таблицы с устройствами отображаются переменные, смонтированные на входы (рис. 32).



Ин...	Имя
0	%IX0.0=X012_28@LDInternal
1	%IX0.1=XRBLVKL@LDInternal
2	%IX0.2=ZBLVKL@LDInternal
3	%IX0.3=ZBSHT@LDInternal
4	%IX0.4=XRK066@LDInternal
5	%IX0.5=HBM5@LDInternal
6	%IX0.6=XBO@LDInternal
7	%IX0.7=UBPRD@LDInternal
8	%IX0.8=XOBSHTC@LDInte...
9	%IX0.9
10	%IX0.10
11	%IX0.11
12	%IX0.12
13	%IX0.13
14	%IX0.14
15	%IX0.15
16	%IX0.16

Рис. 32. Переменные, смонтированные на входы

Монтирование осуществляется двойным кликом на строку входа и выбором переменной во всплывающем окне (рис. 33).

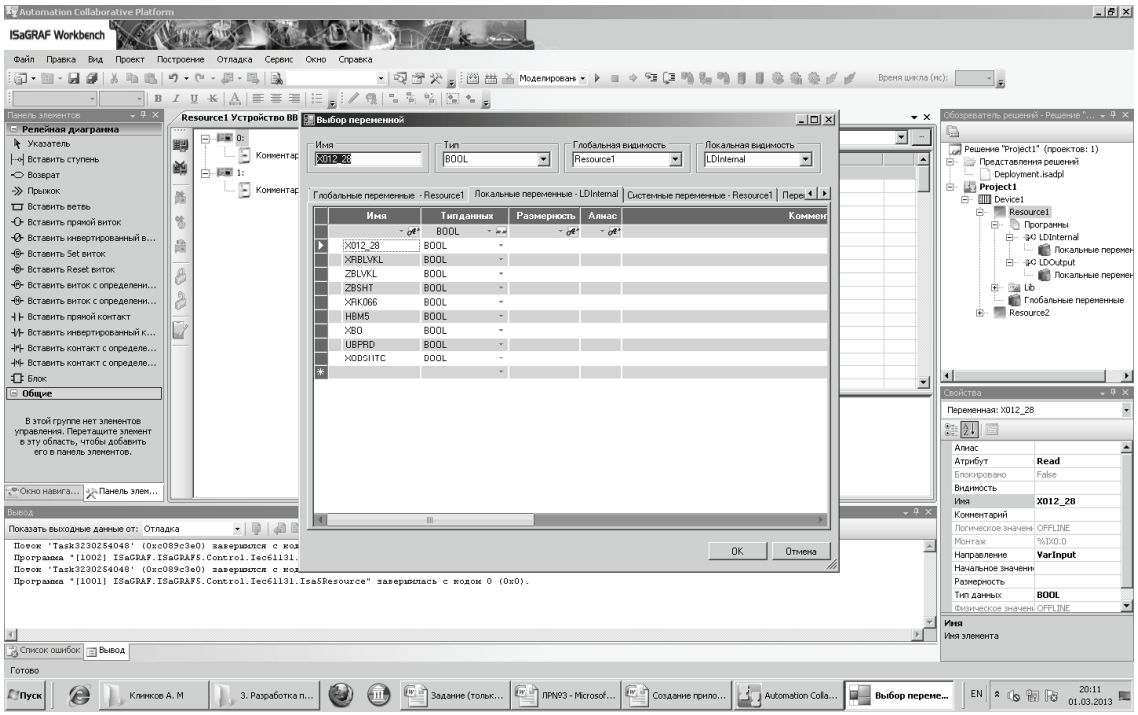


Рис. 33. Монтирование переменной

После монтирования задается направление каждой переменной (рис. 34).

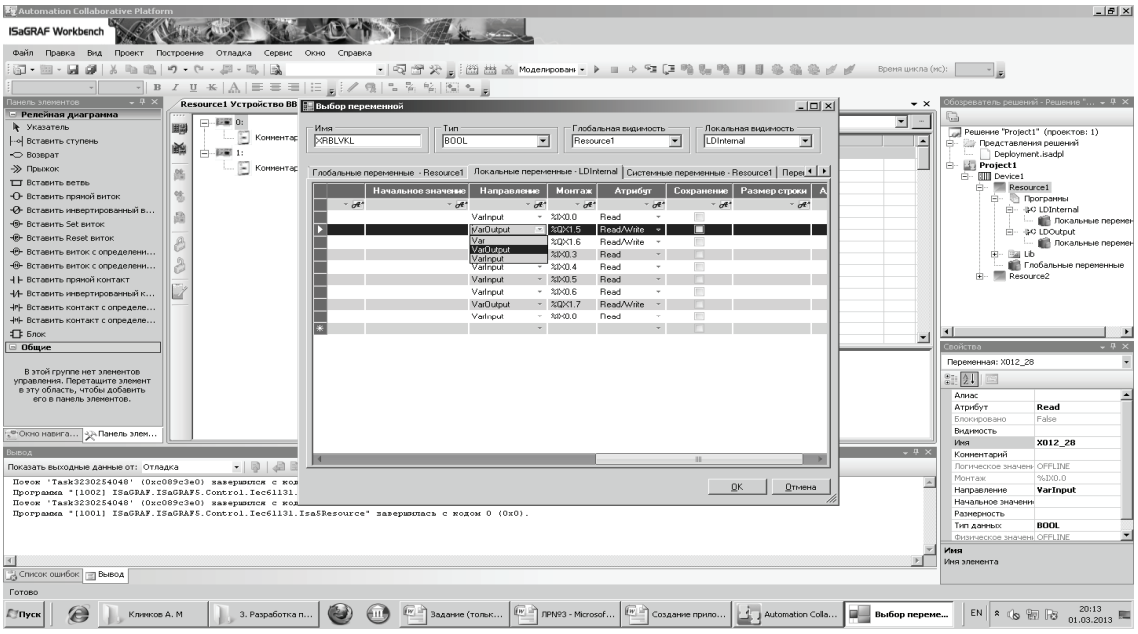


Рис. 34. Задание направления переменной

2.2.5. Отладка программ

По завершении действий по монтированию производится отладка, во время которой на входы подаются значения из окна Устройств (рис. 35) или в программе LD (рис. 36–37).

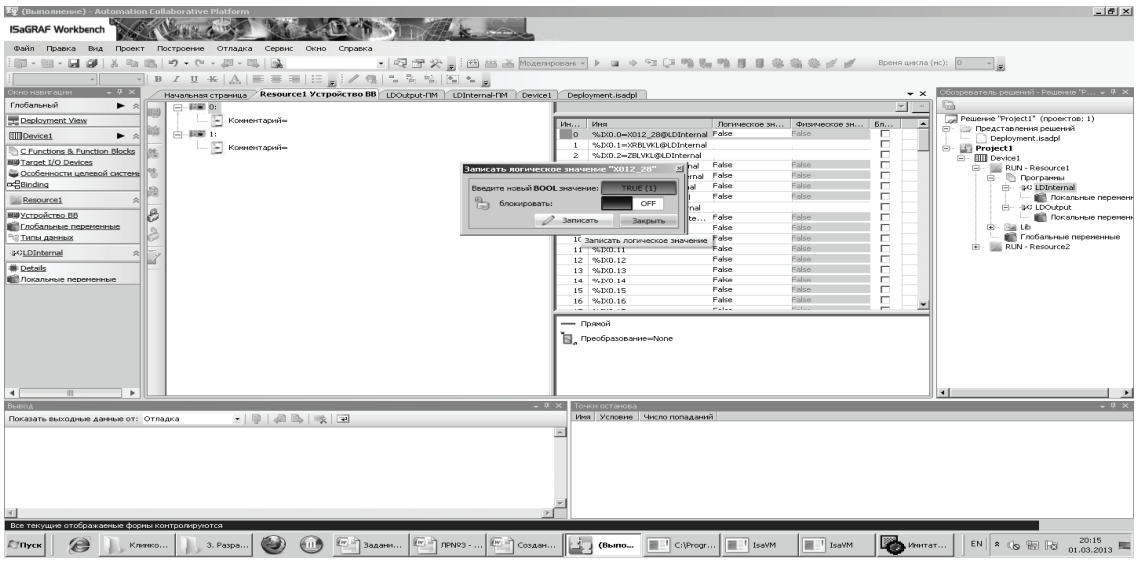


Рис. 35. Запись значения переменной через устройство

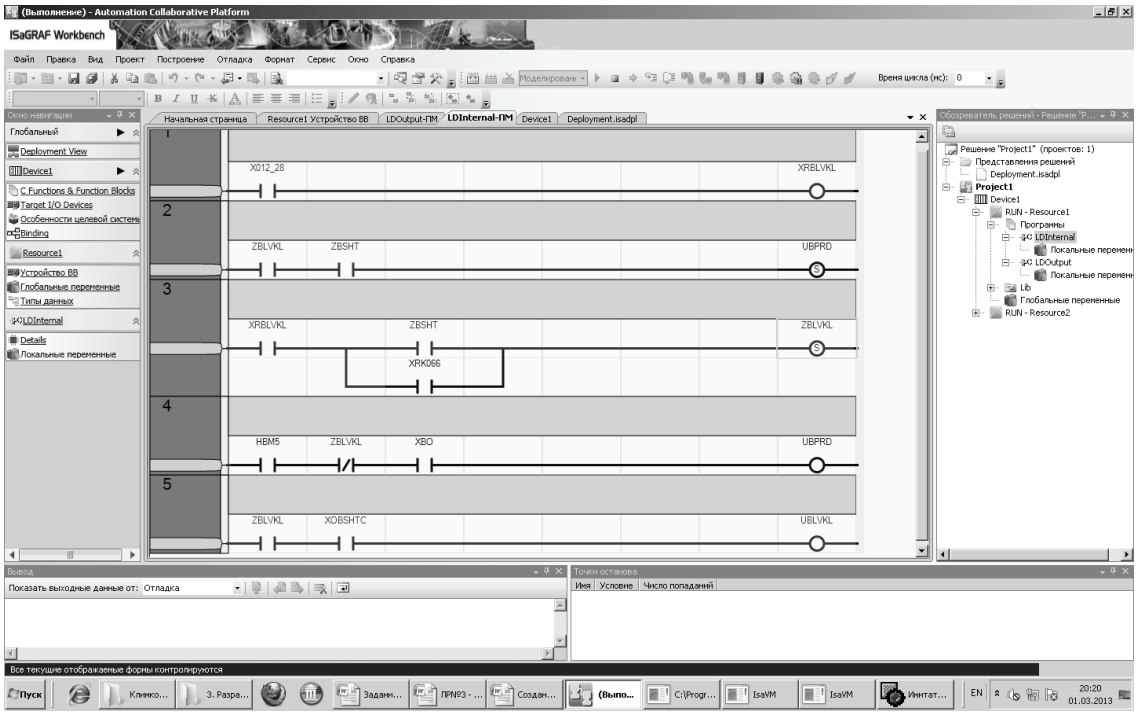


Рис. 36. Результат ввода значений переменных через устройство вода на диаграмме LD Internal

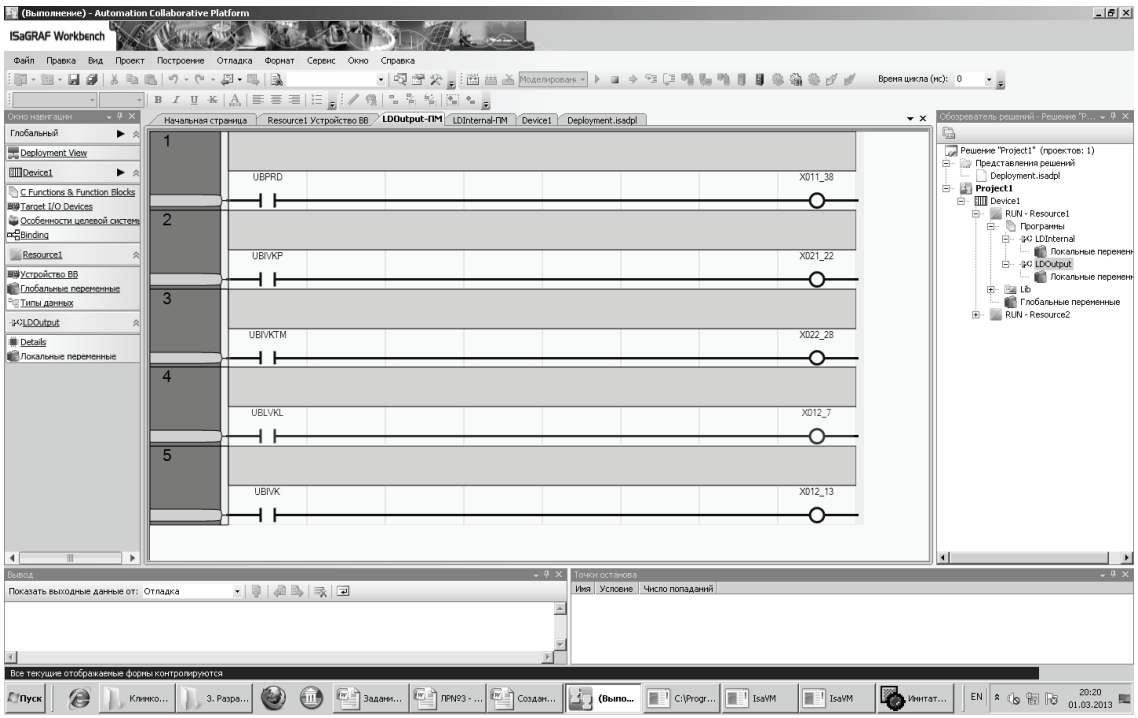


Рис. 37. Результат ввода значений переменных через устройство вода на диаграмме LDOutput

3. РАЗРАБОТКА ПРОЕКТОВ СЛУ НА ЯЗЫКЕ ST

3.1. Описание языка ST

Язык структурированного текста – ST (Structured Text) – это текстовый язык высокого уровня с инструкциями и синтаксисом уровня адаптированного языка Паскаль. Он позволяет программировать сложные алгоритмы обработки данных – последовательности команд с использованием: переменных, вызовов функций и функциональных блоков (ФБ), операторов повторения и т. д., а также для описания действий внутри шагов и условий языка **SFC**. В основном используется в тех случаях, когда алгоритм трудно описать с помощью графических языков.

Логика функциональных блоков создается в C++ и не может быть изменена в ST-редакторе.

Основой ST-программы служат выражения, которые состоят из операндов (констант и переменных) и операторов.

Операторы являются «командами» языка программирования ST. Они должны заканчиваться точкой с запятой. Одна строка может содержать несколько операторов (отделяемых точками с запятой).

Результат вычисления выражения присваивается переменной при помощи оператора присваивания «:=».

Имена, используемые в исходном коде (идентификаторы переменных, константы, ключевые слова), разделены неактивными разделителями (пробелами, символами окончания строки и табуляции) или активными разделителями, которые имеют заранее определенное значение (например, символ-разделитель «>» означает сравнение больше чем, а символ «+» – операцию сложения и т. д.).

В текст могут быть введены комментарии, которые должны начинаться символами «(*» и заканчиваться ими же «)*».

Тип всех операндов выражения должен быть одинаковым. Для изменения типов можно использовать функции преобразования типов: BOO, ANA, REAL, TMR и MSG.

Для того чтобы отделить части выражения и явно определить приоритетность операций, используются скобки. Когда в сложном выражении нет скобок, приоритетность ST-операторов задана неявно.

Рассмотрим операторы языка ST:

1. Вызов функций из ST

Имя: имя вызываемой функции, написанной на языках IEC 61131 или «C»

Назначение: вызывает ST, IL, LD или FBD Функции или «C» функцию и получает ее возвращаемое значение

Синтаксис: <**variable**> **:=** <**funct**> **(**<**par1**>**,** ... <**parN**> **);**

Операнды: тип возвращаемого значения и параметров вызова должен соответствовать интерфейсу, определенному для функции.

Возвращаемое значение: значение, возвращаемое функцией.

Вызовы функций могут быть использованы в любом выражении.

2. Вызов функциональных блоков из ST

Имя: имя экземпляра функционального блока

Назначение: вызывает функциональный блок из стандартной библиотеки или библиотеки пользователя и обращается к его возвращаемым параметрам.

Синтаксис: (* вызвать функциональный блок *)

<**blockname**> **(** <**p1**>**,** <**p2**> ... **);**

(* получить возвращаемые параметры *)

<**result**> **:=** <**blockname**>**.** <**ret_param1**>**;**

...

<**result**> **:=** <**blockname**>**.** <**ret_paramN**>**;**

Операнды: параметры являются выражениями, которые соответствуют типу параметров, специфицированных для этого ФБ

Возвращаемое значение: см. синтаксис получения возвращаемых параметров

Назначение

Имя: :=

Назначение: í означает переменной выражение

Синтаксис: <variable> := <any_expression> ;

Операнды: переменная должна быть Internal или Output и выражение должны иметь тот же тип.

Выражение может быть вызовом функции

3. Предложение IF-THEN-ELSIF-ELSE

Имя: IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF

Назначение: выполняет один из нескольких списков предложений ST. Выбор осуществляется в соответствии со значением булевского выражения

Синтаксис: IF <**Boolean_expression**> **THEN**

<statement> ;

<statement> ;

...

ELSIF <Boolean_expression> THEN
<statement> ;
<statement> ;
...
ELSE
<statement> ;
<statement> ;
...
END_IF;

Предложения ELSE и ELSIF являются необязательными. Если предложение ELSE опущено и условие равно FALSE, то никакие инструкции не выполняются. Предложение ELSIF может использоваться многократно. Предложение ELSE, если используется, должно появляться только один раз в конце последовательности ‘IF, ELSIF ...’.

4. Предложение CASE

Имя: CASE ... OF ... ELSE ... END_CASE

Назначение: выполняет один из нескольких списков предложений ST.

Выбор осуществляется в соответствии с целочисленным выражением

Синтаксис: CASE <integer_expression> OF
<value> : <statements> ;
<value> , <value> : <statements> ;
...
ELSE
<statements> ;
END_CASE;

Значениями Case должны быть целые константные выражения. Несколько значений, разделенных запятыми, могут предшествовать одному и тому же списку предложений. Предложение ELSE является необязательным.

5. Предложение FOR

Имя: FOR ... TO ... BY ... DO ... END_FOR

Назначение: выполняет ограниченное число итераций, используя целую индексную переменную

Синтаксис: FOR <index> := <mini> TO <maxi> BY <step> DO
<statement> ;
<statement> ;
END_FOR;

Операнды: index: внутренняя целая переменная, увеличивающаяся в каждом цикле итерации;

mini: начальное значение для индекса (перед первой итерацией);

maxi: максимально допустимое значение для индекса;

step: приращение индекса в каждом цикле итерации.

Предложение [BY step] является необязательным. Если оно не приведено, то шаг приращения равен 1.

Предупреждение: поскольку ядро является **синхронной** системой, входные переменные не обновляются в течение итераций FOR.

6. Предложение WHILE

Имя: WHILE ... DO ... END_WHILE

Назначение: итерационная структура для группы ST предложений. Условие "продолжения" оценивается ПЕРЕД любой итерацией.

Синтаксис: WHILE <Boolean_expression> DO

<statement> ;

<statement> ;

...

END_WHILE

Предупреждение: поскольку ядро является **синхронной** системой, входные переменные не обновляются в течение итераций WHILE. Изменение состояния входной переменной не может быть использовано для описания условия предложения WHILE.

7. Предложение REPEAT

Имя: REPEAT ... UNTIL ... END_REPEAT

Назначение: итерационная структура для группы ST предложений. Условие "продолжения" оценивается ПОСЛЕ любой итерации.

Синтаксис: REPEAT

<statement> ;

<statement> ;

...

UNTIL <Boolean_condition>

END_REPEAT ;

Предупреждение: поскольку ядро является синхронной системой, входные переменные не обновляются в течение итераций REPEAT. Изменение состояния входной переменной не может быть использовано для описания условия предложения REPEAT.

8. Предложение EXIT

Имя: EXIT

Назначение: выход из итерационных предложений FOR, WHILE или REPEAT.

Синтаксис: EXIT

EXIT обычно используется в предложении IF внутри блока FOR, WHILE или REPEAT.

9. Предложение RETURN

Имя: RETURN

Назначение: прекращает выполнение текущей программы.

Синтаксис: RETURN

Операнды: (нет).

В блоке действий SFC предложение RETURN обозначает конец выполнения только данного блока.

10. GSTART

Имя: GSTART

Назначение: запускает дочернюю программу SFC, устанавливая маркер на все ее начальные шаги.

Синтаксис: GSTART (<**child_program**>);

Операнды: указываемая SFC программа должна быть дочерней по отношению к программе, в которой написано предложение.

Возвращаемое значение: (нет).

Потомки дочерней программы предложением GSTART автоматически не запускаются.

11. GKILL

Имя: GKILL

Назначение: убивает дочернюю SFC программу, удаляя маркеры, существующие на ее шагах.

Синтаксис: GKILL (<**child_program**>);

Операнды: указываемая SFC программа должна быть дочерней по отношению к программе, в которой написано предложение.

Возвращаемое значение: (нет).

12. GFREEZE

Имя: GFREEZE

Назначение: замораживает дочерний модуль SFC (программу или функциональный блок); приостанавливает его выполнение. Приостановленный ПМ SFC может затем быть перезапущен с использованием предложения GRST.

Синтаксис: GFREEZE (<**child_program**>);

Операнды: указываемая SFC программа должна быть дочерней по отношению к программе, в которой написано предложение.

Возвращаемое значение: (нет).

Потомки дочерней программы замораживаются автоматически совместно с указанной программой.

13. GRST перезапускает замороженную SFC программу

Имя: GRST

Назначение: перезапускает дочернюю программу, замороженную предложением GFREEZE: все маркеры, удаленные GFREEZE, восстанавливаются.

Синтаксис: GRST (<**child_program**>);

Операнды: указываемая SFC программа должна быть дочерней по отношению к программе, в которой написано предложение.

Возвращаемое значение: (нет).

Потомки дочерней программы предложением GRST перезапускаются автоматически.

14. GSTATUS получает текущее состояние SFC программ

Имя: GSTATUS

Назначение: возвращает текущее состояние SFC программы.

Синтаксис: <**var**> := **GSTATUS** (<**child_program**>);

Операнды: указываемая SFC программа должна быть дочерней по отношению к программе, в которой написано предложение.

Возвращаемое значение:

0 = программа неактивна (убита);

1 = программа активна (запущена);

2 = программа заморожена.

3.2. Структура проекта

Проект имеет название Kursovoi. Проект Kursovoi написан на языке ST. Структура проекта приведена на рис. 38.

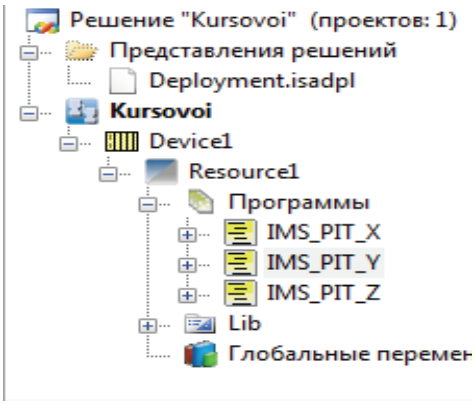


Рис. 38. Структура проекта

3.3. Словарь проекта

Словарь проекта (рис. 39) представляет собой набор глобальных переменных.

В ISAGRAF переменные могут иметь одно из следующих направлений:

- Var. Внутренняя переменная, обновляемая программами.
- VarInput. Переменная, подключенная к устройству ввода (регенерируется системой).
- VarOutput. Переменная, подключенная к устройству вывода.

Имя	Тип данных	Алиас	Направление	Атрибут
X063_12	BOOL		VarInput	Read
X063_1	BOOL		VarInput	Read
X000_1	BOOL		VarInput	Read
X000_12	BOOL		VarInput	Read
X067_10	BOOL		VarInput	Read
XBPIT	BOOL		Var	Read/Write
XBOPIT	BOOL		Var	Read/Write
XBO	BOOL		Var	Read/Write
XB	BOOL		Var	Read/Write
XTM	BOOL		Var	Read/Write
ZPZV1	BOOL		Var	Read/Write
ZPZV2	BOOL		Var	Read/Write
ZPZV3	BOOL		Var	Read/Write
ZPOZ1	BOOL		Var	Read/Write
ZPOZ2	BOOL		Var	Read/Write
ZPOC1	BOOL		Var	Read/Write
ZPOC2	BOOL		Var	Read/Write
ZGK1	BOOL		Var	Read/Write
ZGK2	BOOL		Var	Read/Write
ZGK3	BOOL		Var	Read/Write

Рис. 39. Словарь проекта

3.4. Разработка программ

В данном проекте все переменные являются глобальными переменными булевского типа. Каждая имеет имя, заданное по принципу транслитерации переменных из исходной таблицы Excel.

Программа IMS_PIT_X, представленная ниже, содержит коды, определяющие значения переменных «Шины питания», «Пороговые элементы».

```
(*Шины питания*)
(*ХБПИТ=Х063_12*)
ХВРІТ :=Х063_12;
(*ХБОПИТ=Х063_1*)
ХВОРІТ := Х063_1;
```

```
(*Пороговые элементы*)
(*Рднс1 = 1*)
Rdns1 := TRUE;
(*Рднс2 = 1*)
Rdns2 := TRUE;
```

Программа IMS_PIT_Y содержит коды (фрагменты), определяющие значения переменных «Разряды слов», «Выходы», «Силовые контакты», «Сухие контакты», «Электронные ключи».

```
(*Разряды слов*)
(*СД18_00=ZПО31*)
CD18_00 := ZPOZ1;
(*СД18_01=ZПО32*)
CD18_01 := ZPOZ2;
(*СД18_02=ZГК1*)
CD18_02 := ZGK1;
```

```
(*Выходы*)
(*УПЗВ1П=ZПЗВ1*ХБОПИТ*)
YPZV1P := ZPZV1 AND XBOPIT;
(*УПЗВ1=ХБПИТ*)
YP3V1 := XBPIT;
(*УПЗВ2П=ZПЗВ2*ХБОПИТ*)
YPZV2P := ZPZV2 AND XBOPIT;
```

```
(*Силовые контакты*)
(*Х056_1=УПЗВ1П*)
X056_1 := YPZV1P;
(*Х056_33=УПЗВ1*)
X056_33 := YP3V1;
(*Х056_5=УПЗВ2П*)
X056_5 := YPZV2P;
```

```
(*Сухие контакты*)
(*Х067_1=УРДНС_ТМ*)
X067_1 := YRDNS_TM;
(*Х067_3=УДНС1_ТМ*)
X067_3 := YDNS1_TM;
```

```
(*Электронные ключи*)
(*Х067_26=УРУБИС_ТМ*)
X067_26 := YRYBIS_TM;
(*Х067_27=УРZБИС_ТМ*)
X067_27 := YRZBIS_TM;
```

Программа IMS_PIT_Z содержит коды, определяющие значения переменных «Функции памяти».

```
(*Функции памяти*)
(*ZПЗВ1=Х3КУ00+ZПЗВ1*^Х3КУ02*)
ZPZV1 :=XZKU00 OR ZPZV1 AND NOT XZKU02;
(*ZПЗВ2=Х3КУ01+ZПЗВ2*^Х3КУ03*)
ZPZV2 :=XZKU01 OR ZPZV2 AND NOT XZKU03;
(*ZГК2=Х3КУ0F+ZГК2*^Х3КУ12*)
ZGK2 :=XZKU0F OR ZGK2 AND NOT XZKU12;
(*ZГК3=Х3КУ10+ZГК3*^Х3КУ13*)
ZGK3 :=XZKU10 OR ZGK3 AND NOT XZKU13;
```

3.5. Монтирование переменных

После написания программы на языке ST мы должны смонтировать переменные. Ниже приведены примеры смонтированных входных и выходных переменных (рис. 40–41). Переменные монтируем по двум типам: входные булевские переменные и выходные булевские переменные.

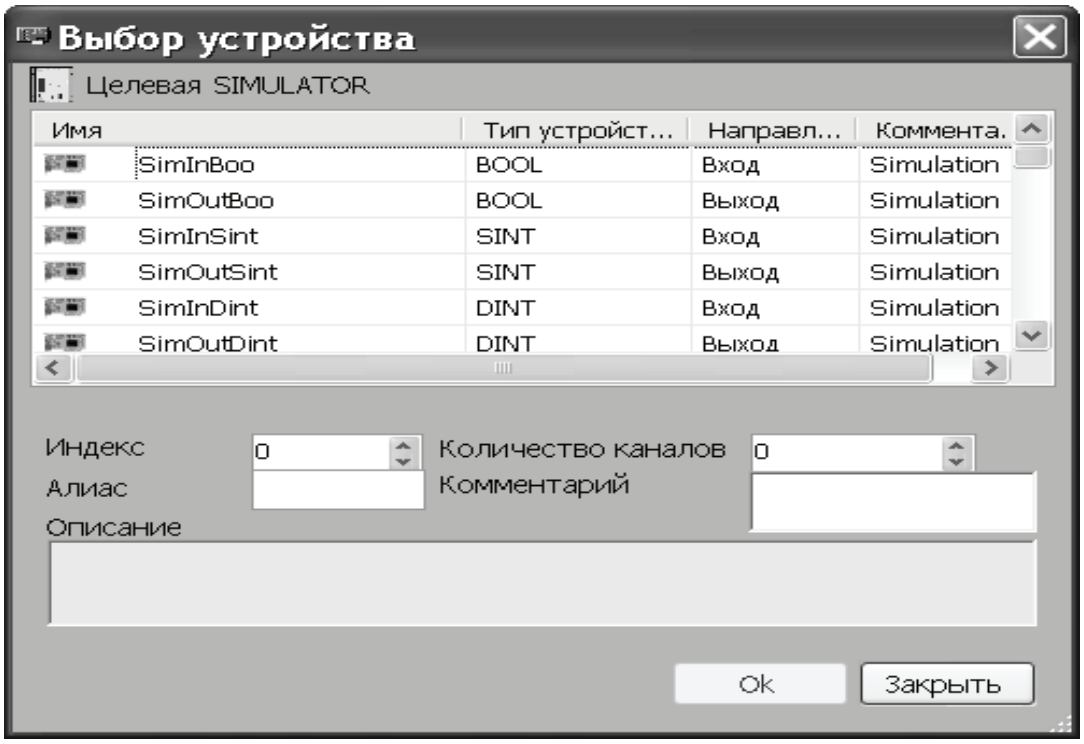
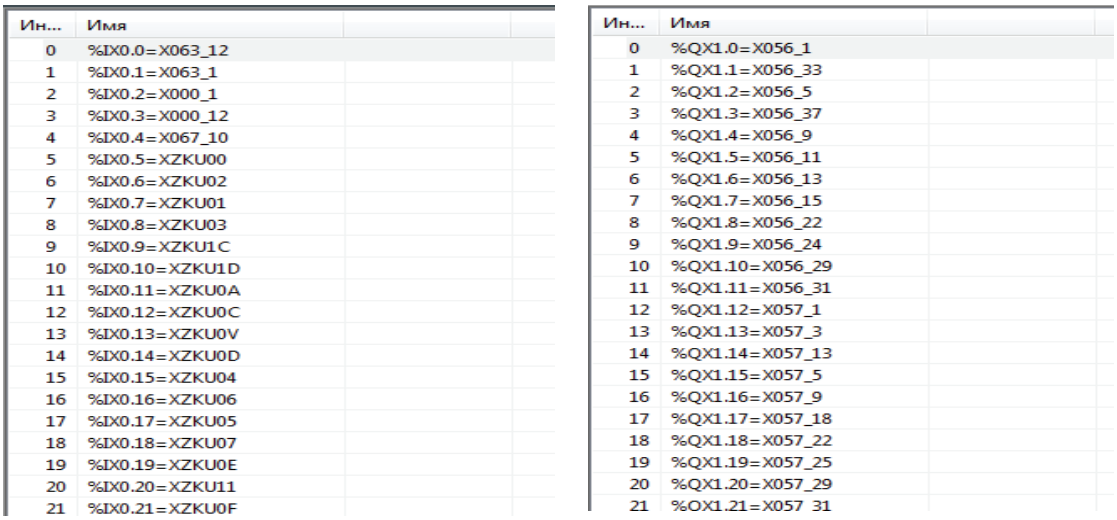


Рис. 40. Вид экрана выбора устройства



Ин...	Имя
0	%IX0.0=X063_12
1	%IX0.1=X063_1
2	%IX0.2=X000_1
3	%IX0.3=X000_12
4	%IX0.4=X067_10
5	%IX0.5=XZKU00
6	%IX0.6=XZKU02
7	%IX0.7=XZKU01
8	%IX0.8=XZKU03
9	%IX0.9=XZKU1C
10	%IX0.10=XZKU1D
11	%IX0.11=XZKU0A
12	%IX0.12=XZKU0C
13	%IX0.13=XZKU0V
14	%IX0.14=XZKU0D
15	%IX0.15=XZKU04
16	%IX0.16=XZKU06
17	%IX0.17=XZKU05
18	%IX0.18=XZKU07
19	%IX0.19=XZKU0E
20	%IX0.20=XZKU11
21	%IX0.21=XZKU0F

Ин...	Имя
0	%QX1.0=X056_1
1	%QX1.1=X056_33
2	%QX1.2=X056_5
3	%QX1.3=X056_37
4	%QX1.4=X056_9
5	%QX1.5=X056_11
6	%QX1.6=X056_13
7	%QX1.7=X056_15
8	%QX1.8=X056_22
9	%QX1.9=X056_24
10	%QX1.10=X056_29
11	%QX1.11=X056_31
12	%QX1.12=X057_1
13	%QX1.13=X057_3
14	%QX1.14=X057_13
15	%QX1.15=X057_5
16	%QX1.16=X057_9
17	%QX1.17=X057_18
18	%QX1.18=X057_22
19	%QX1.19=X057_25
20	%QX1.20=X057_29
21	%QX1.21=X057_31

Рис. 41. Смонтированные входные и выходные булевские переменные

3.6. Отладка программ

Изменяя значения входных переменных, наблюдаем и оцениваем изменение выходных переменных. На рис. 42–43 приведены примеры результатов выполнения кода программы IMS_PIT.

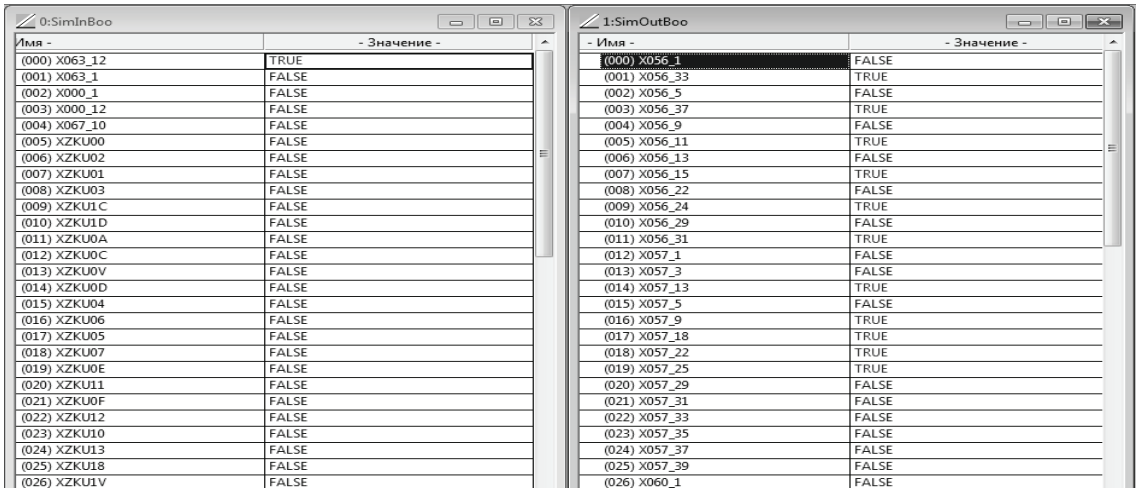


Рис. 42. Пример результата выполнения сценария программы IMS_PIT

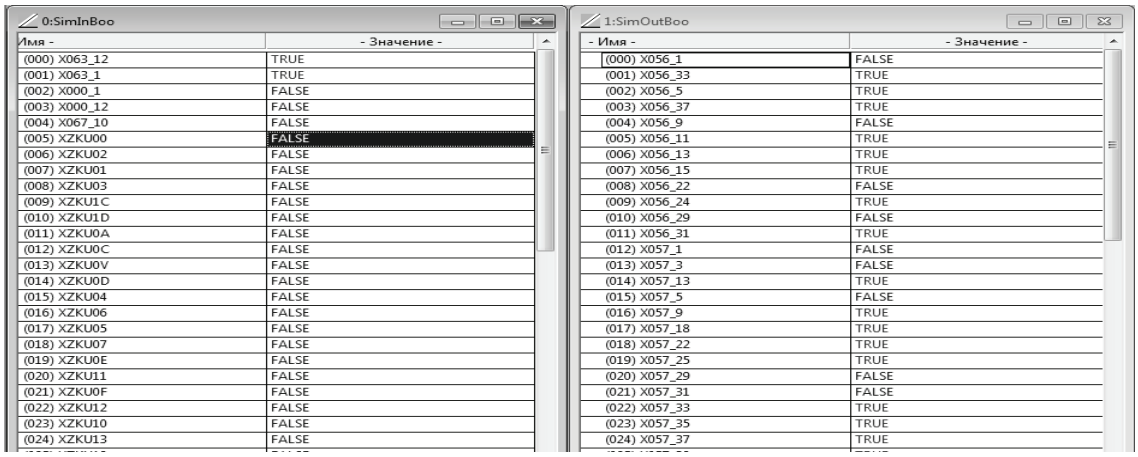


Рис. 43. Пример результата выполнения сценария программы IMS_PIT

Изменяя значения входных переменных, наблюдаем и оцениваем изменение выходных переменных.

4. РАЗРАБОТКА ПРОЕКТОВ СЛУ НА ЯЗЫКЕ SFC

4.1. Описание языка SFC

Язык SFC предназначен для использования на этапе проектирования ПО и позволяет описать блок-схему программы, т. е. логику ее работы на уровне последовательных шагов и условных переходов. Он обеспечивает общую структуризацию и координацию функций управления последовательными процессами или машинами и механизмами.

SFC-программа состоит из элементов двух типов: шагов (steps) и переходов (transitions), которые могут включать в себя элементы других языков.

Логические структуры, связанные с шагом, обрабатываются до тех пор, пока не произойдет событие, предписывающее перейти к обработке другого шага. Каждому переходу сопоставлено логическое условие, а шагу – совокупность действий.

SFC-программа – это графически представленная совокупность шагов и переходов, соединенных направленными связями. Основное правило при построении схем: шаги не могут следовать подряд; переходы тоже не могут следовать подряд.

Программирование на SFC обычно разделяется на два различных уровня:

уровень 1 – показывает графически блок-схемы, номера ссылок на шаги, переходы и комментарии, присоединенные к ним;

уровень 2 – программирование действий внутри шага или условий, присоединенных к переходу, на языке ST или IL.

Начальная ситуация описывается начальным шагом; после запуска программы автоматически активизируются (выделяются) все начальные шаги.

У шага имеются атрибуты, которые могут быть использованы в любом другом языке:

GSnnn.x – характеризует его активность (логическая переменная);

GSnnn.t – характеризует продолжительность (время) его активного состояния (таймер),

здесь nnn – номер шага.

На втором уровне программирования осуществляется детальное описание действий, которые выполняются во время активности шага,

и условий, которые соответствуют переходам. По умолчанию языком программирования второго уровня является язык ST.

Расхождения – это множественные связи от одного шага или перехода ко многим.

Схождения – это множественные связи от более чем одного шага или перехода к одному другому.

При обозначении схождений и расхождений используются одиночные или двойные линии.

Альтернативные расхождения и схождения обозначаются одиночными горизонтальными линиями.

Расхождение альтернативное (альтернативные ветви) – это множественная связь от одного шага к нескольким переходам. Активной становится одна из ветвей (в зависимости от активности того или иного перехода). Проверка активности переходов осуществляется слева направо.

Каждая альтернативная ветвь начинается и заканчивается собственным условием перехода.

Проверка альтернативных условий выполняется слева направо. Если верное условие найдено, то прочие альтернативы не рассматриваются. В таких ветвях всегда работает только одна из них, поэтому ее окончание и будет означать переход к следующему за альтернативной группой шагу. При создании альтернативных ветвей желательно задавать взаимоисключающие условия.

Схождение альтернативное – используется для того, чтобы объединить несколько ветвей SFC, начавшихся из альтернативного расхождения.

Параллельные расхождения (параллельные ветви) и схождения обозначаются двойными горизонтальными линиями.

Каждая параллельная ветвь начинается и заканчивается шагом. То есть условие входа в параллельность всегда одно, условие выхода тоже всегда одно на всех.

Параллельные ветви выполняются теоретически одновременно. Практически – в одном рабочем цикле, слева направо.

Условие перехода, завершающее параллельность, проверяется только в случае, если в каждой параллельной ветви активны последние шаги.

Иерархия программы SFC. В системе ISaGRAF каждая SFC-программа может управлять (запускать, уничтожать и т. д.) другими программами на этом же языке (SFC), которые в таком случае называют дочерними программами той программы, которая ими управляет.

Элементы языка SFC в системе ISaGRAF-6 представлены ниже на рис. 44–58.

Рис. 44. Начальный шаг

Рис. 45. Шаг

Рис. 46. Активный и неактивный шаг

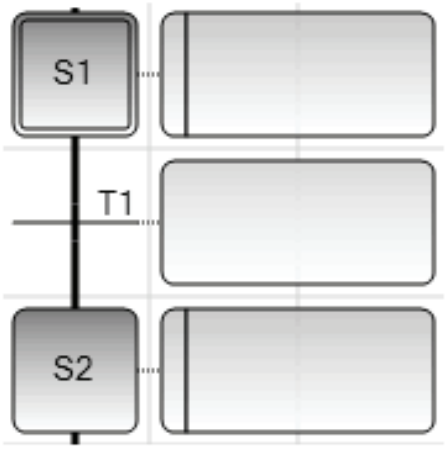


Рис. 47. Переход

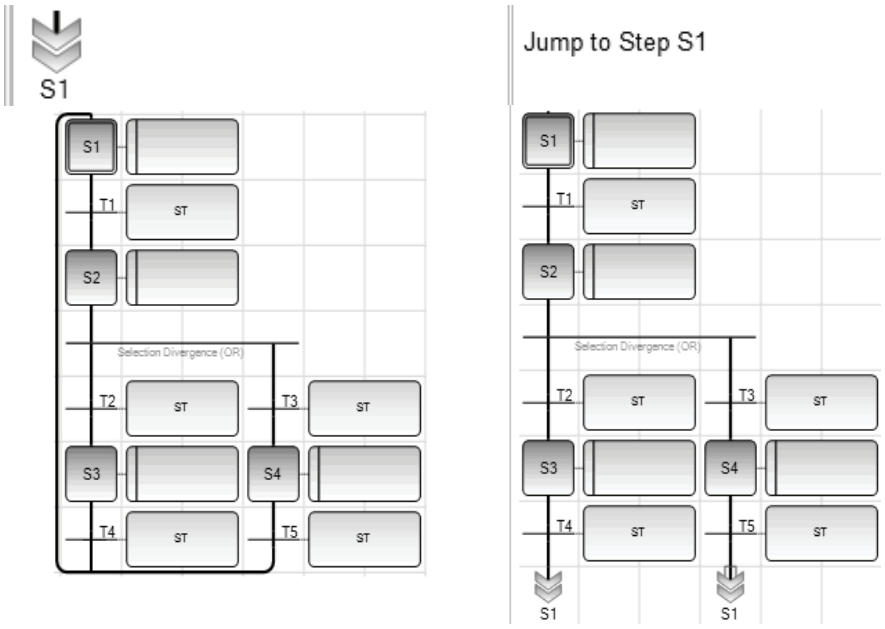


Рис. 48. Длинный переход

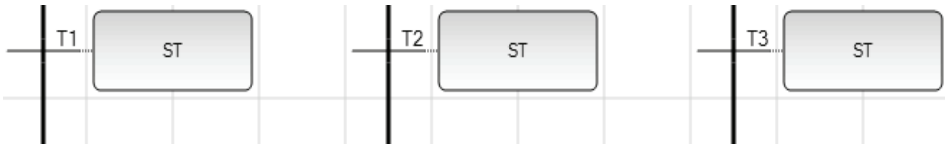
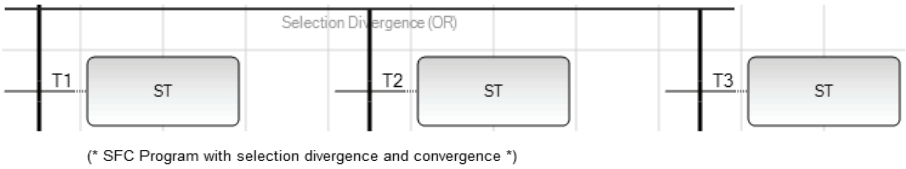


Рис. 49. Альтернативное схождение



(* SFC Program with selection divergence and convergence *)

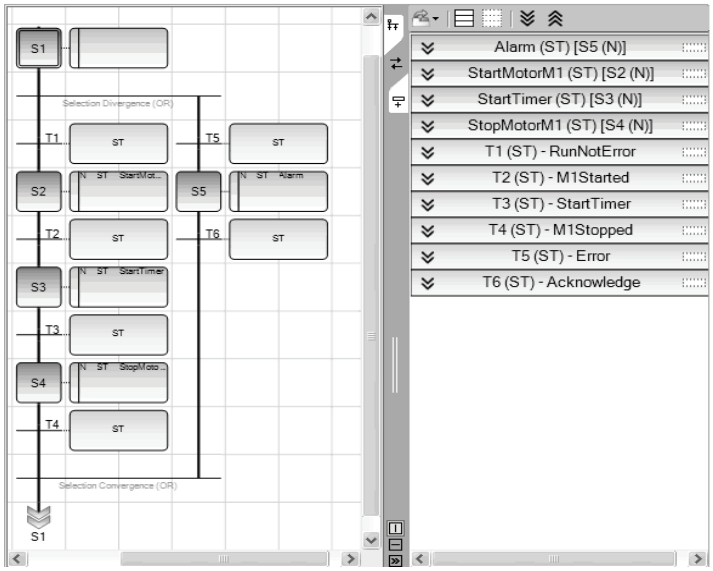


Рис. 50. Альтернативное расхождение

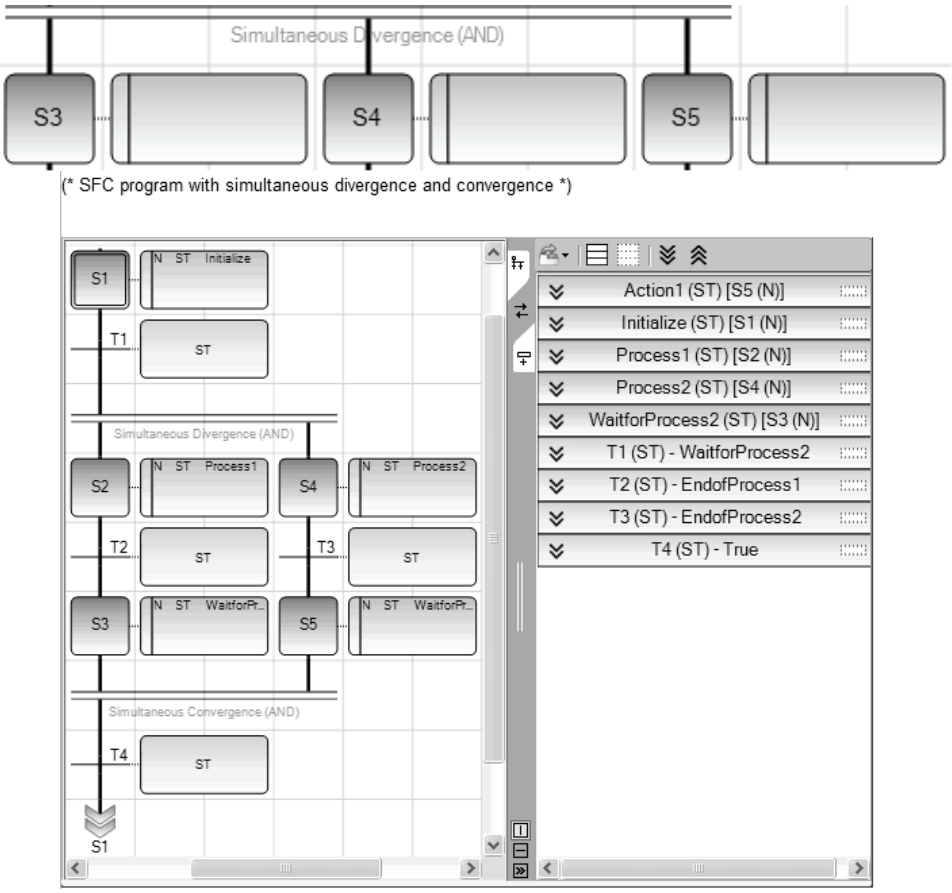


Рис. 51. Параллельное расхождение

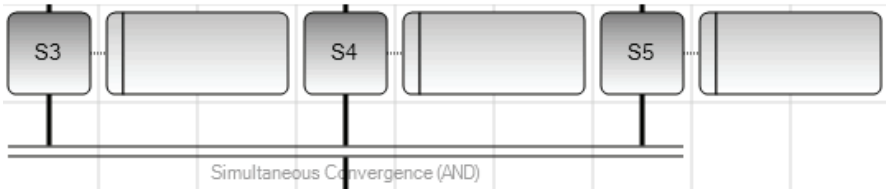


Рис. 52. Параллельное схождение

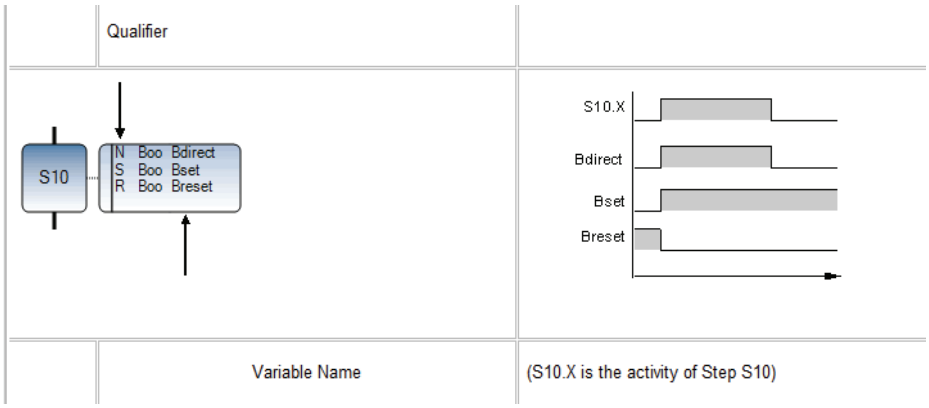


Рис. 53. Булевские действия

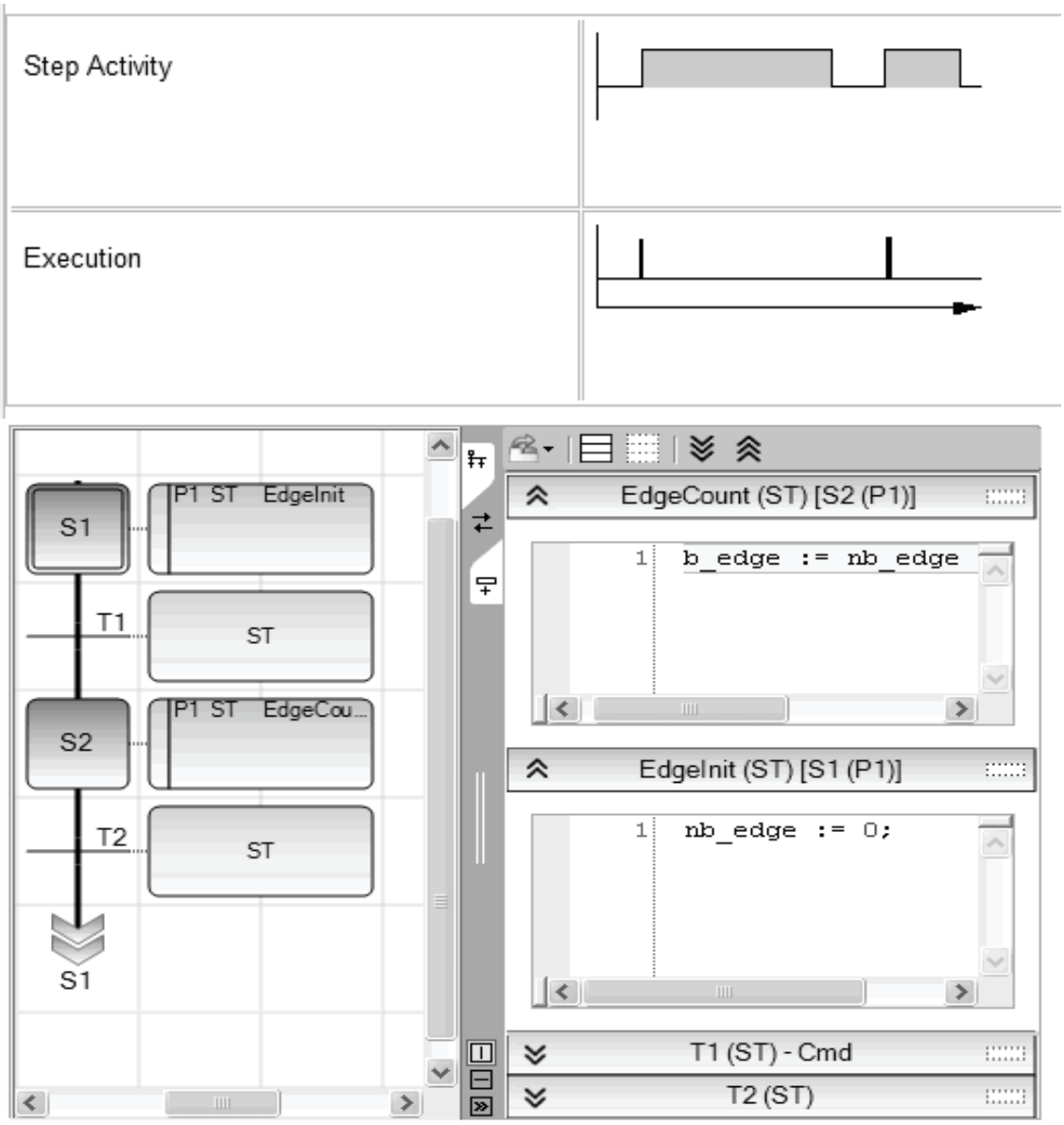


Рис. 54. Импульсные действия

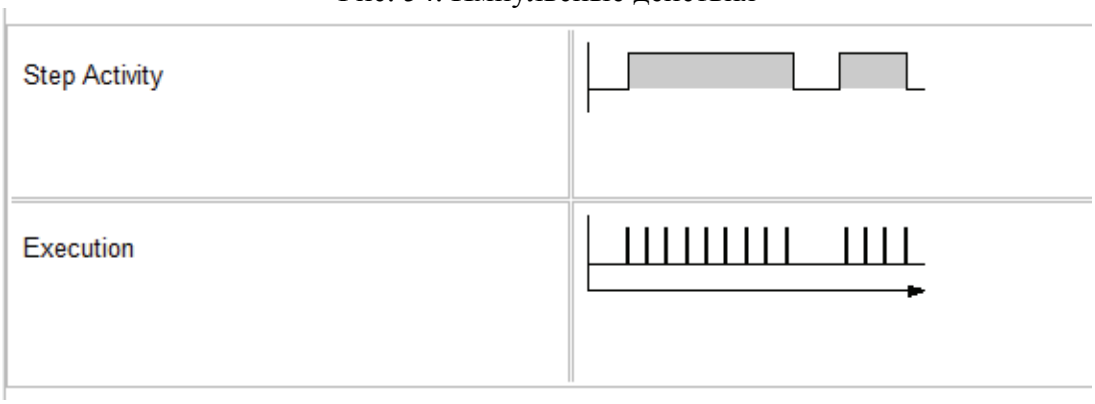


Рис. 55. Не сохраняемые действия (начало, окончание см. на с. 38)

Рис. 55. Окончание (начало см. на с. 37)

```
(* SFC Program with ST programming for Transitions *)
```

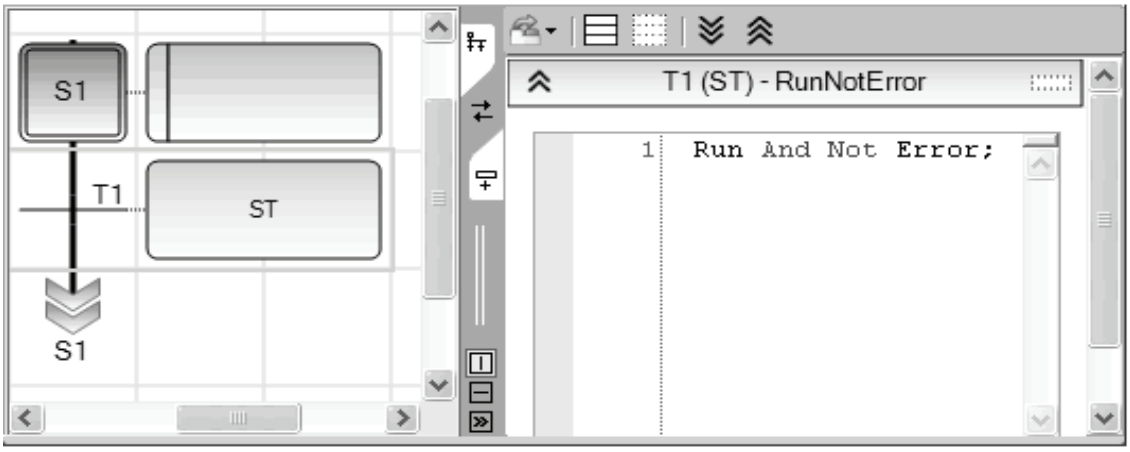


Рис. 56. Программирование условий
переходов на ST

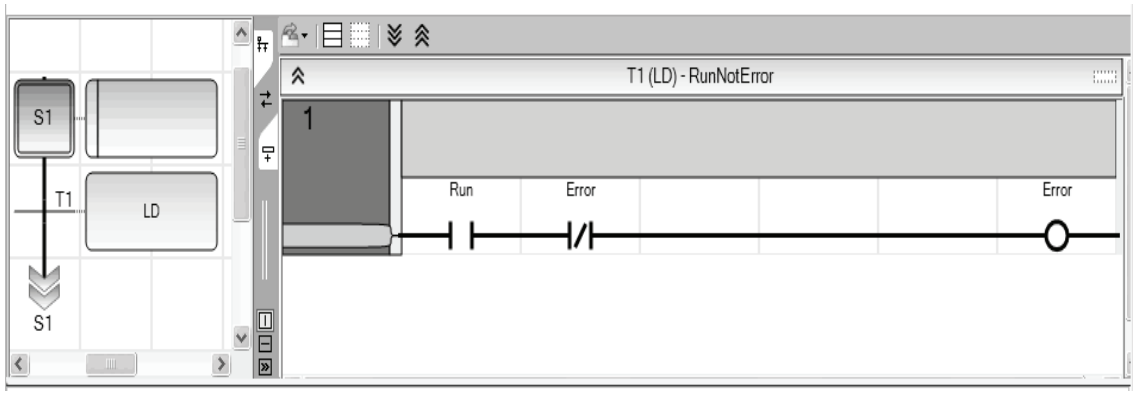


Рис. 57. Программирование условий переходов на LD

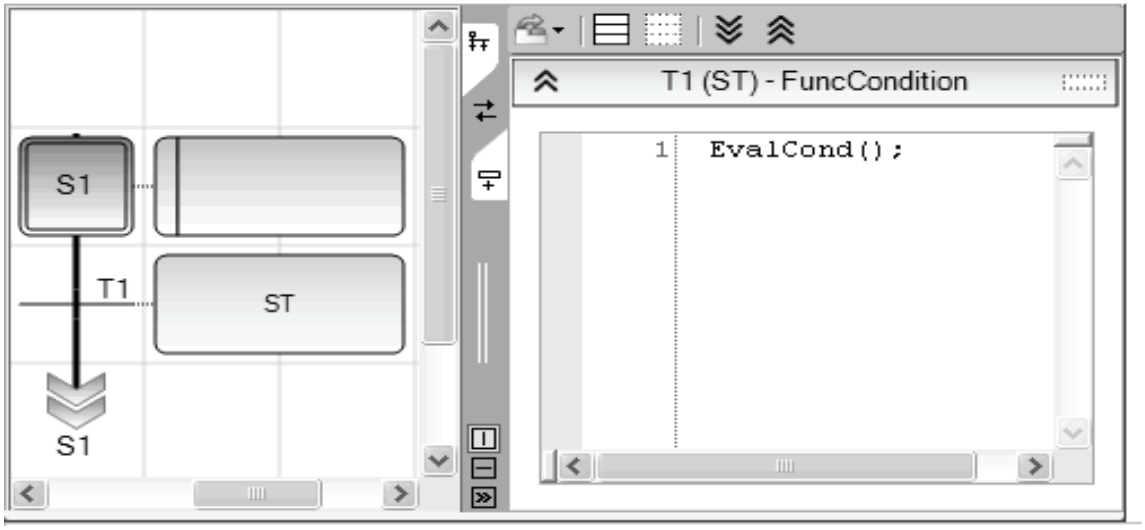


Рис. 58. Вызов функций из переходов

5. ПРОЕКТ УПРАВЛЕНИЯ ПРОЦЕССОМ ОБРАБОТКИ СИГНАЛА ТРЕВОГА

Алгоритмы управления приведены на рис. 59.

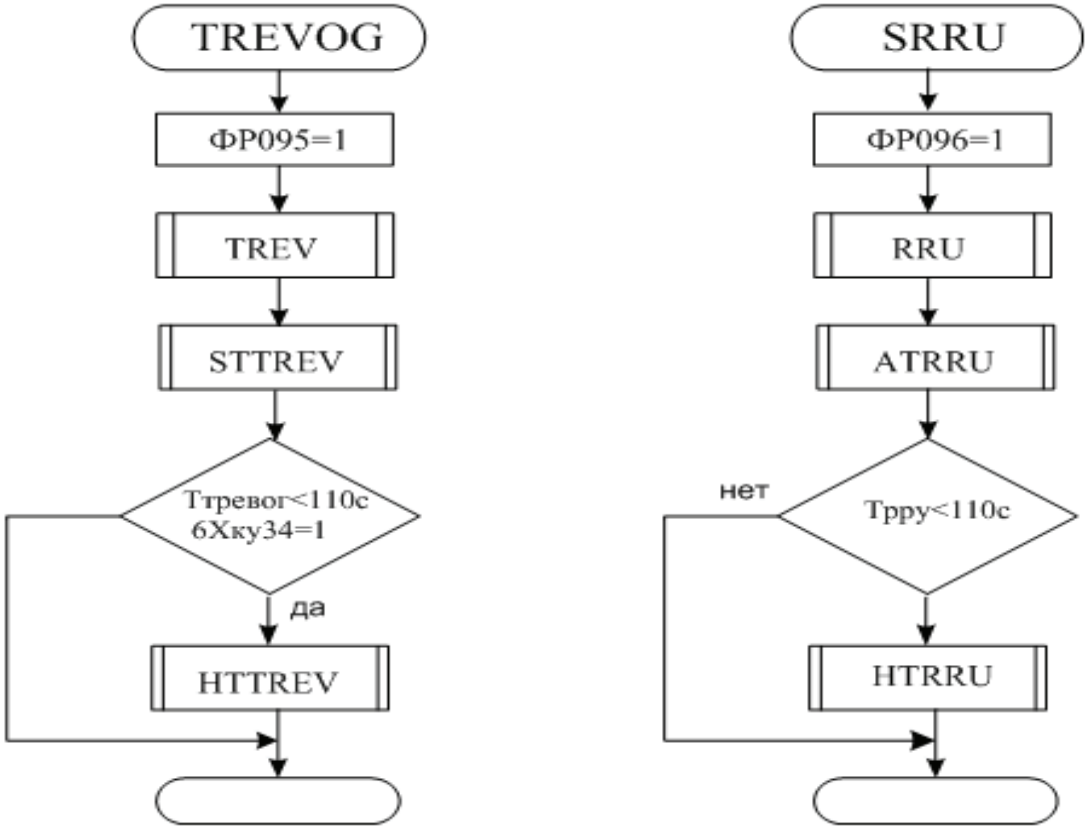


Рис. 59. Алгоритмы управления

5.1. Структура проекта

На рис. 60 изображена первая ступень структуры проекта.

После запуска ISaGraf, на главной странице выбираем пункт «Создать проект». В открывшемся окне выбираем тип проекта ISAGRAF-5, шаблон проекта Simulator, задаем название нашего проекта – Algoritm15.

Проект состоит из двух программ «TREVOG» и «SRRU». Для их создания необходимо в обозревателе решения для проекта выбрать меню Устройство – Ресурсы – Программы – Добавить – Создать SFC (рис. 61).

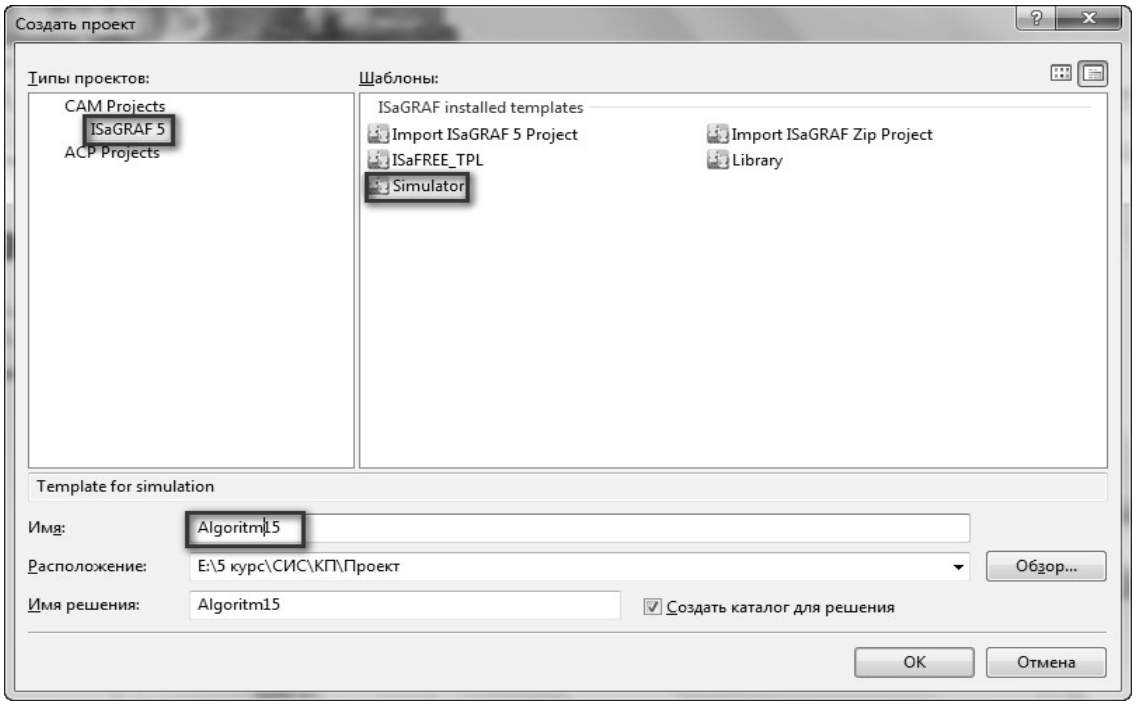


Рис. 60. Создание проекта

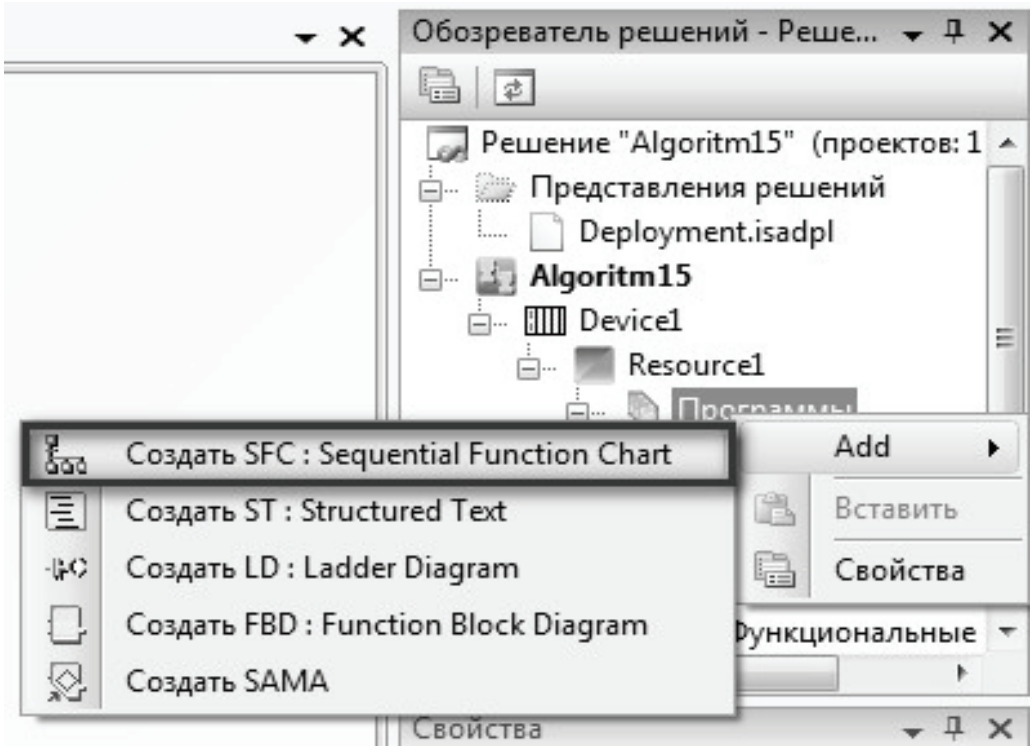
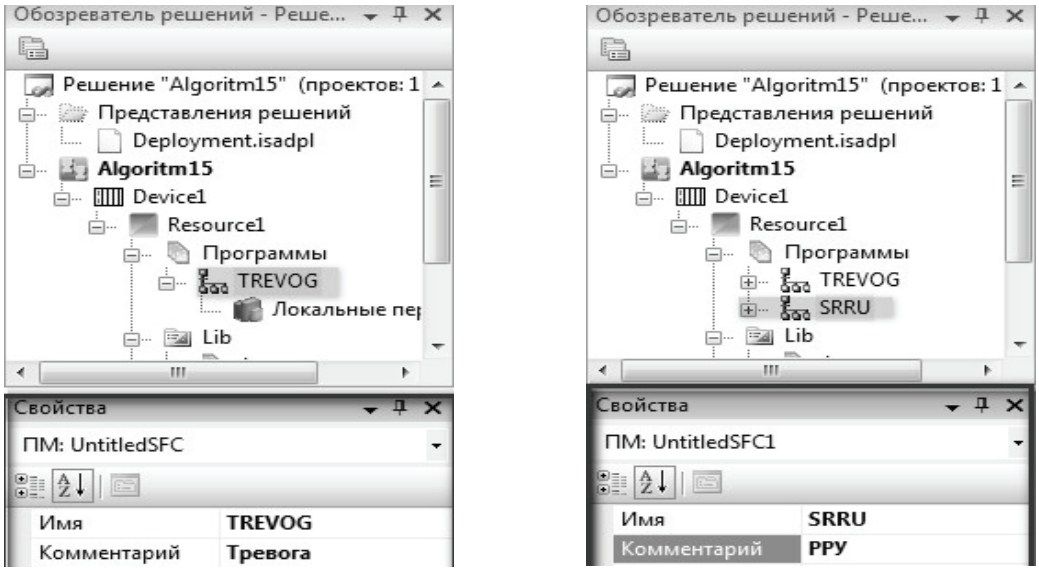


Рис. 61. Создание программы SFC

Далее в свойствах программы устанавливаем соответствующее имя и комментарий (рис. 62, *а*, *б*).



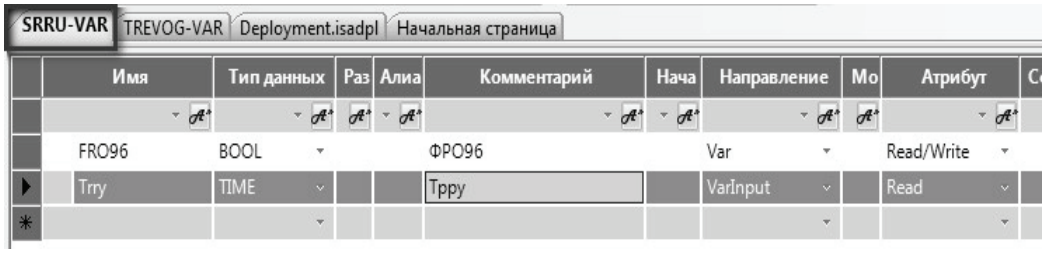
а *б*

Рис. 62. Свойства программы:
а – TREVOG; *б* – SRRU

5.2. Словарь проекта

Переменные проекта могут быть глобальными (т. е. видимыми в любой программе проекта) или локальными для выделенной программы. Записываем все переменные, которые будут необходимы для работы программ в локальные переменные. Для каждой программы используется свой словарь (рис. 63, *а*, *б*).

а



б

Рис. 63. Переменные программы:
а – TREVOG; *б* – SRRU

Для TREVOG необходимо три переменные:

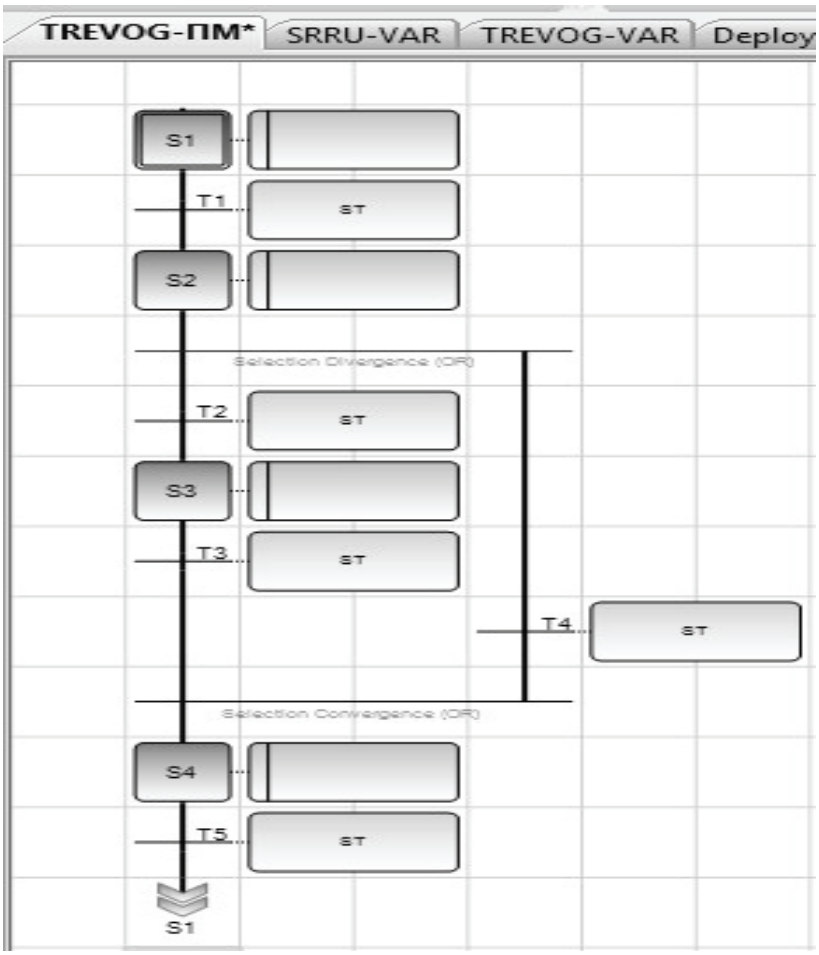
- ФРО95 (булева переменная),
- Х6ку34 (булева переменная),
- Ттревог (временной тип).

Для программы SRRU:

- ФРО96 (булева переменная),
- Трру (временной тип).

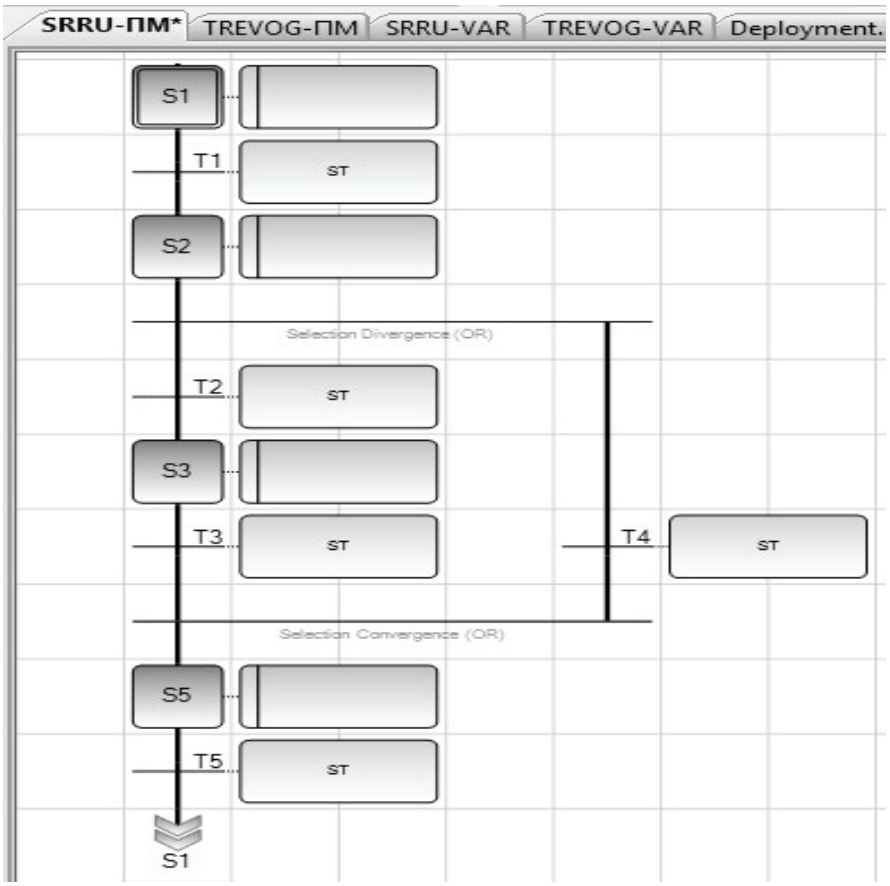
5.3. Разработка программ

В соответствии со схемой алгоритма, изображенной на рис. 63, в SFC-редакторе требуется построить блок-схемы для каждой программы (рис. 64, *а*, *б*).



а

Рис. 64. Блок-схема программы:
а – *а* – TREVOG; *б* – SRRU (начало, окончание см. на с. 44)



б

Рис. 64. Окончание (начало см. на с. 43)

Далее необходимо добавить действие на языке ST для шагов на которых выполняются какие-либо действия. Для этого необходимо щелкнуть правой кнопкой мыши по блоку перехода и выбрать меню Добавить – Создать действие на языке ST (рис. 65).

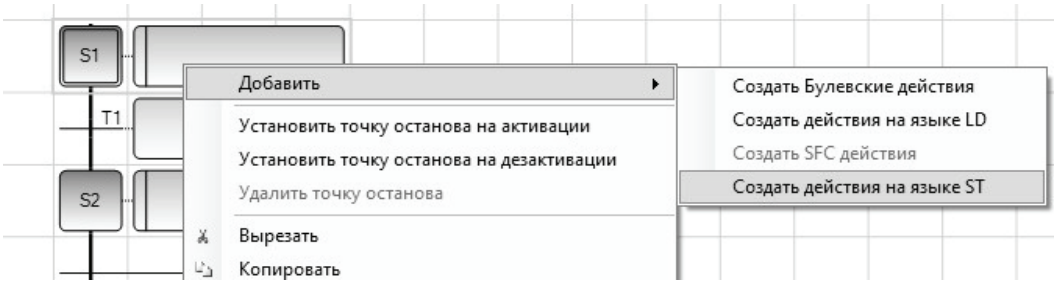
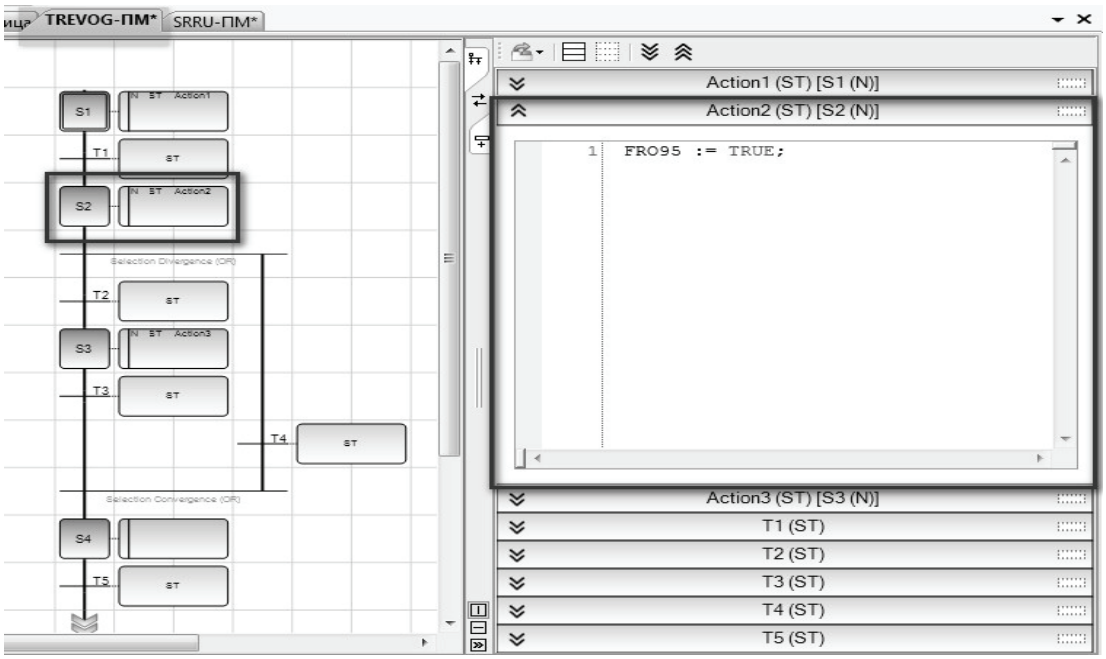
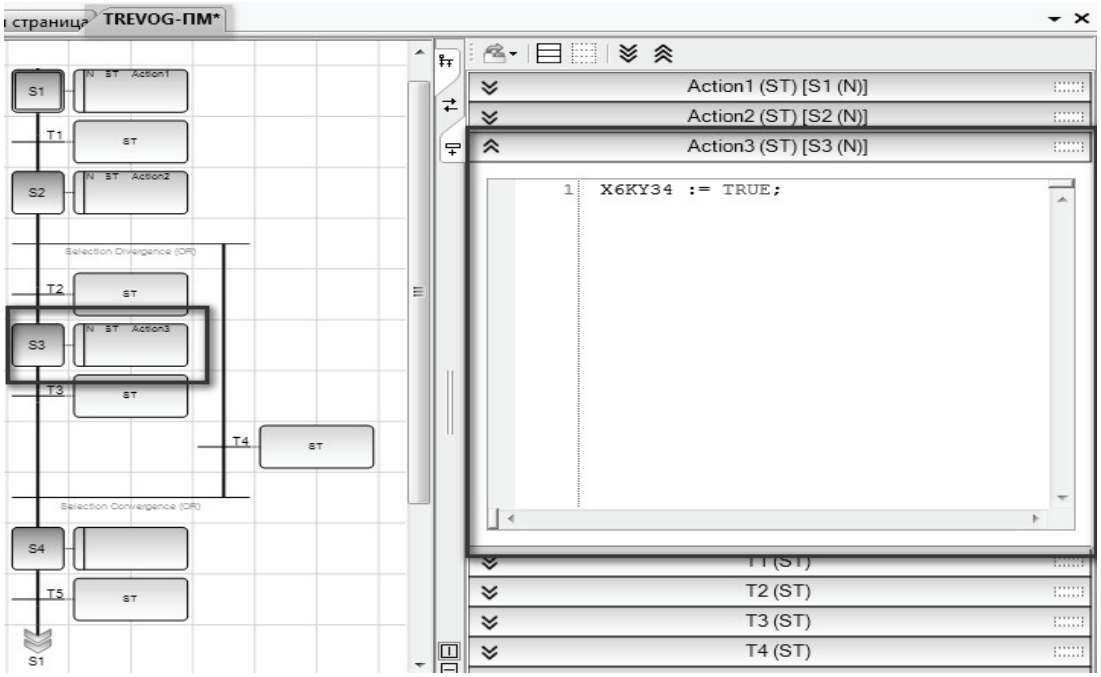


Рис. 65. Добавление действий на языке ST

В соответствии с алгоритмом программы TREVOG переменным ФРО95 и Х6ку34 присваивается единица, что было задано в действии для соответственных блоков (рис. 66, *а*, *б*).



а



б

Рис. 66. Действие на языке ST для блока:
а – S2; *б* – S3

В соответствии с алгоритмом программы SRRU переменной ФРО96 присваивается единица, что было задано в действии для соответствующего блока (рис. 67).

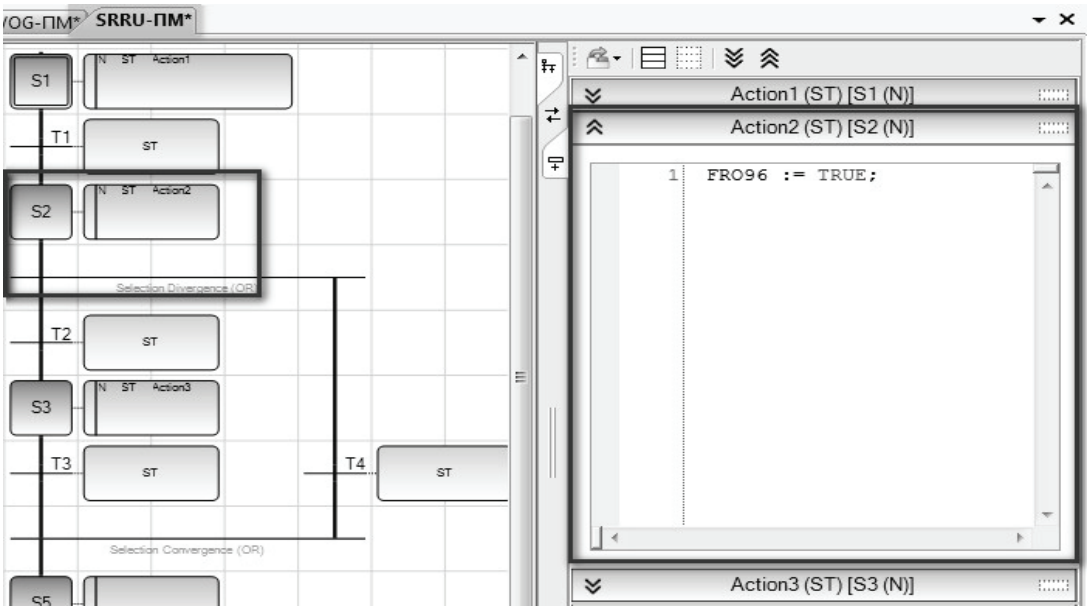
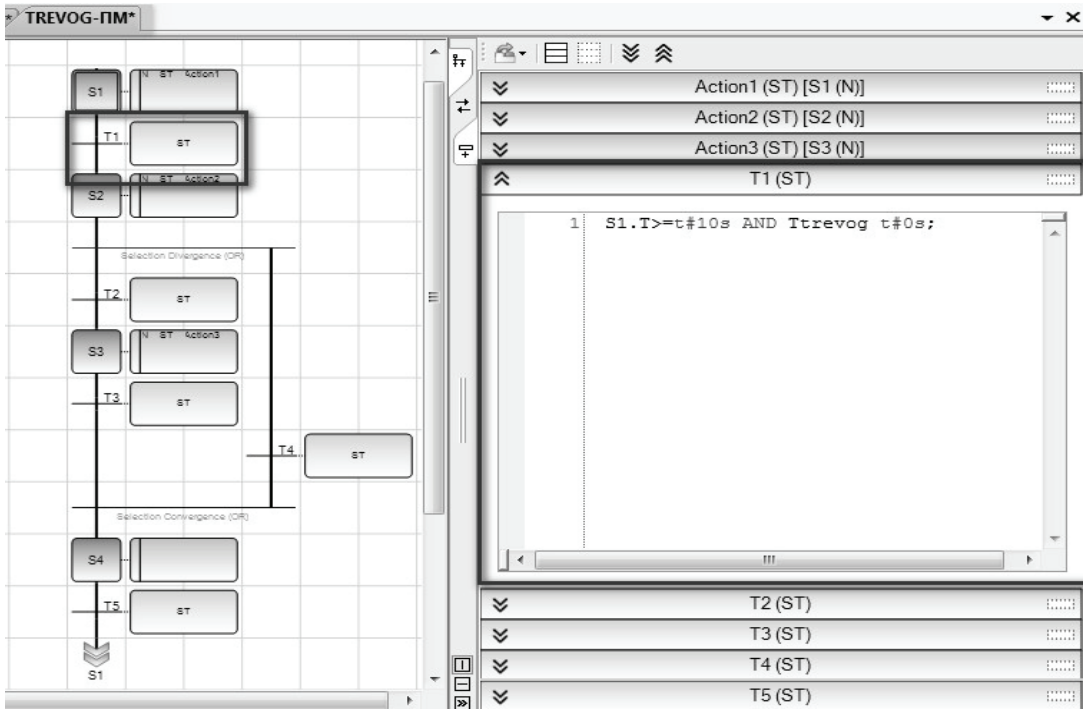


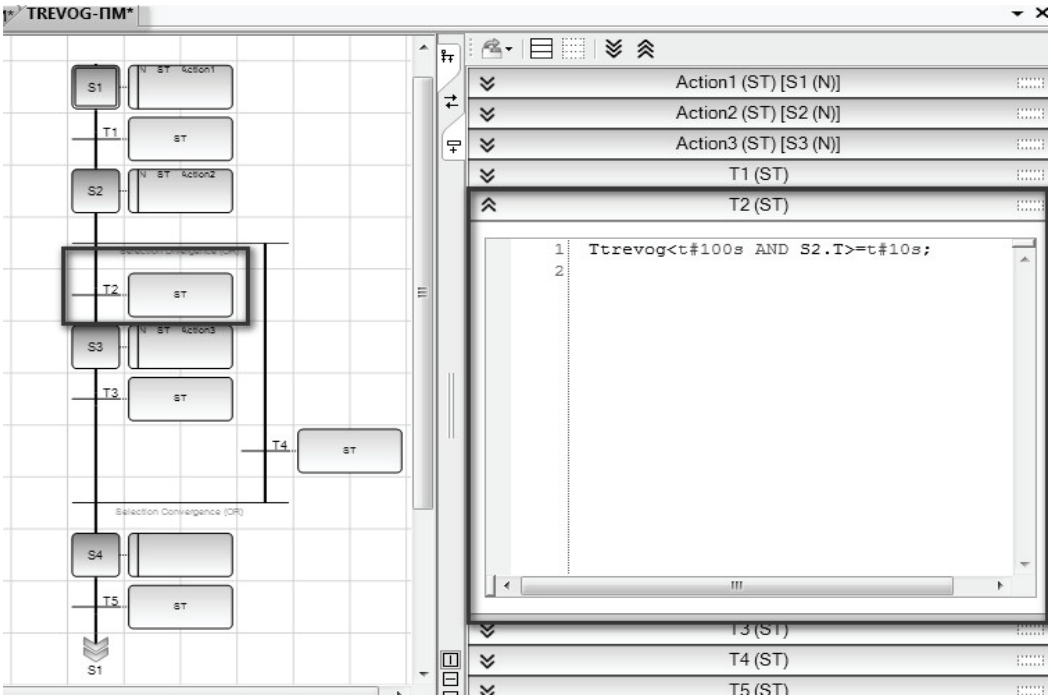
Рис. 67. Действие на языке ST для блока S2

Далее создаем условия для переходов. Устанавливаем таймеры с задержкой на 10 секунд. Условия для переходов программы TREVOG показаны на рис. 68, *а*, *б*.



а

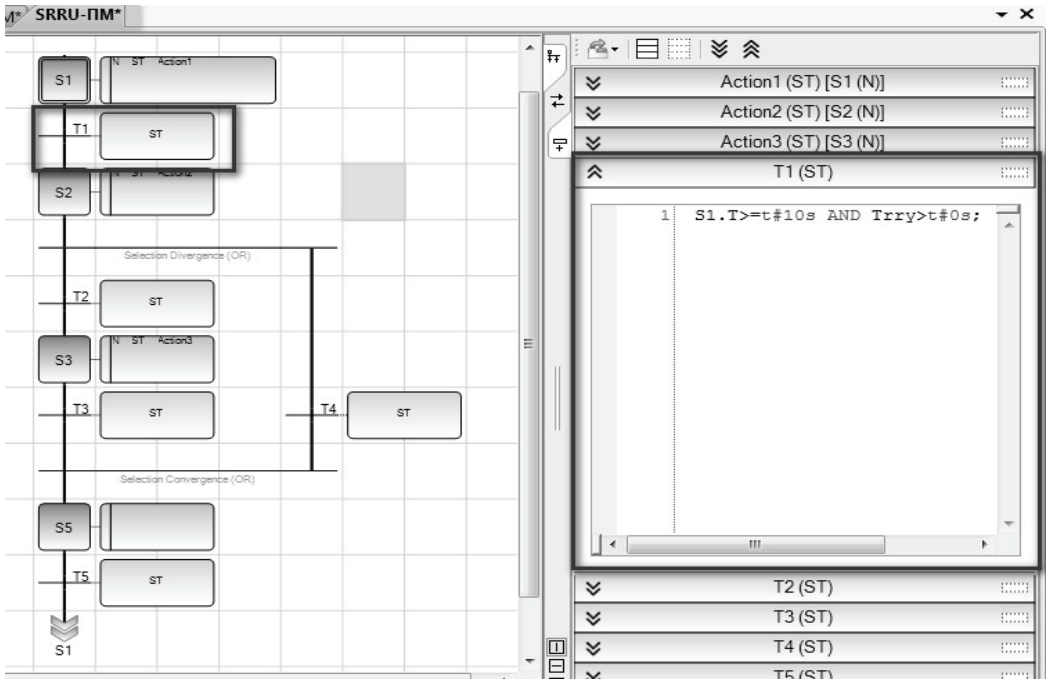
Рис. 68. Условие для перехода:
а – T1; *б* – T2 (начало, окончание см. на с. 47)



б

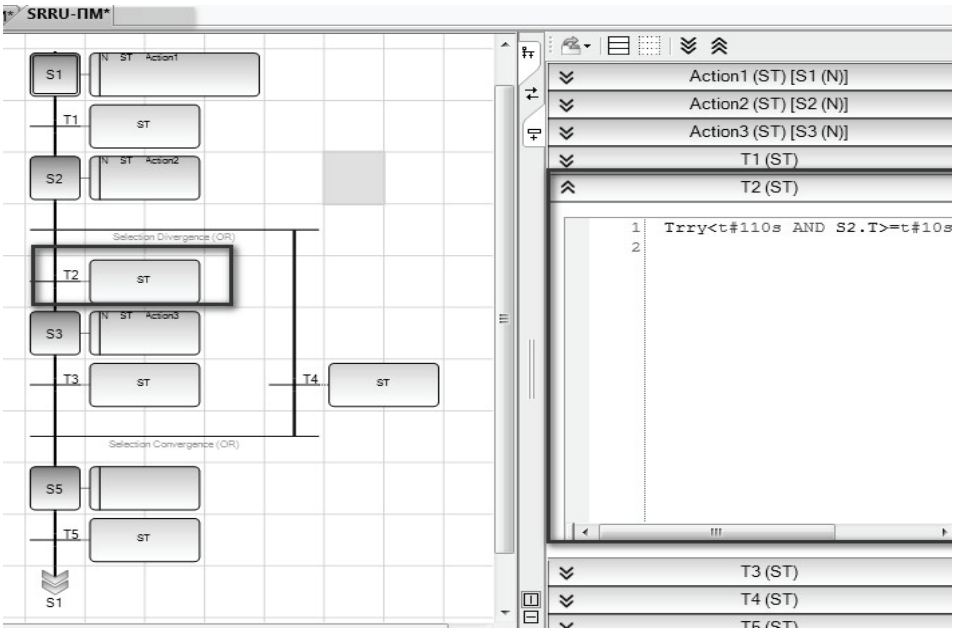
Рис. 68. Окончание (начало см. на с. 46)

Для программы SRRU условия для переходов показаны на рис. 69, *а*, *б*.



а

Рис. 69. Условие для перехода:
а – T1; *б* – T2 (начало, окончание см. на с. 48)



б

Рис. 69. Окончание (начало см. на с. 47)

5.4. Монтирование переменных

Алгоритм содержит две входные переменные, которые необходимо смонтировать, т. е. назначить их виртуальному устройству, чтобы успешно промоделировать процесс. Таким образом, можно назначить переменные Trry (программа SRRU) и Trevog (программа TREVOG) устройству SimInTime (рис. 70).

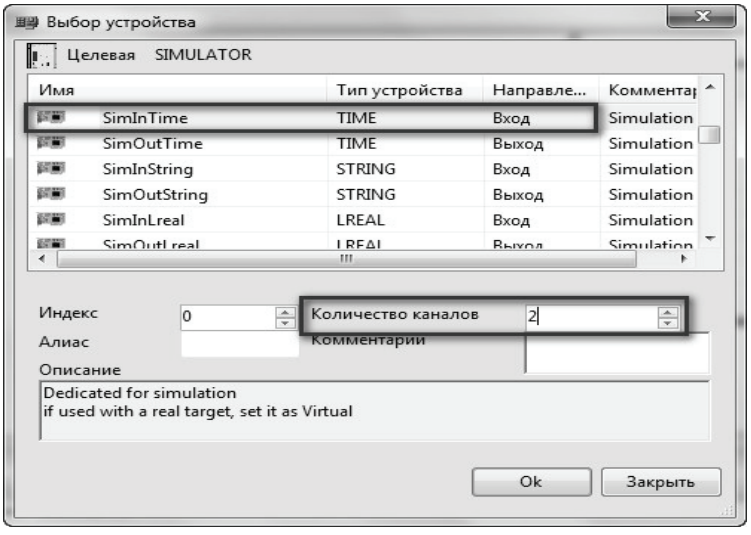


Рис. 70. Монтирование переменных
(начало, окончание см. на с. 49)

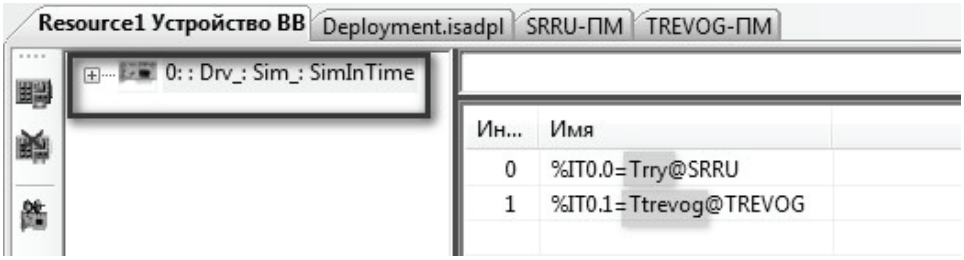


Рис. 70. Окончание (начало см. на с. 48)

После этого необходимо построить проект (рис. 71) и в случае отсутствия ошибок можно приступить к отладке.

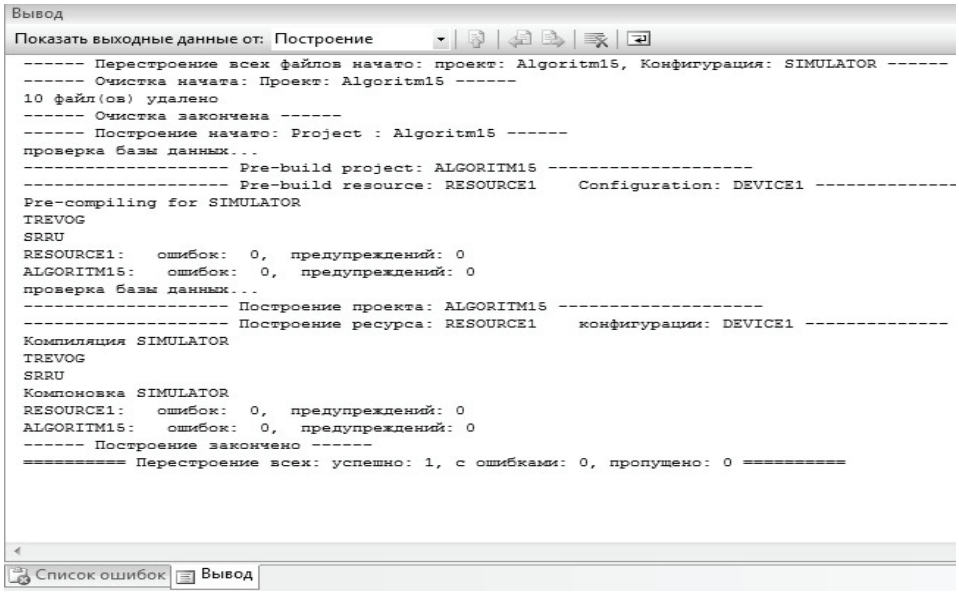


Рис. 71. Вывод при построении проекта

5.5. Отладка проекта

На работу алгоритма влияют две переменные (рис. 72) временного типа.

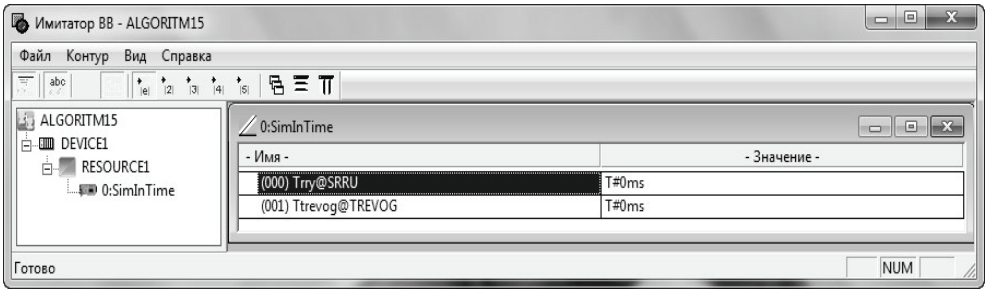
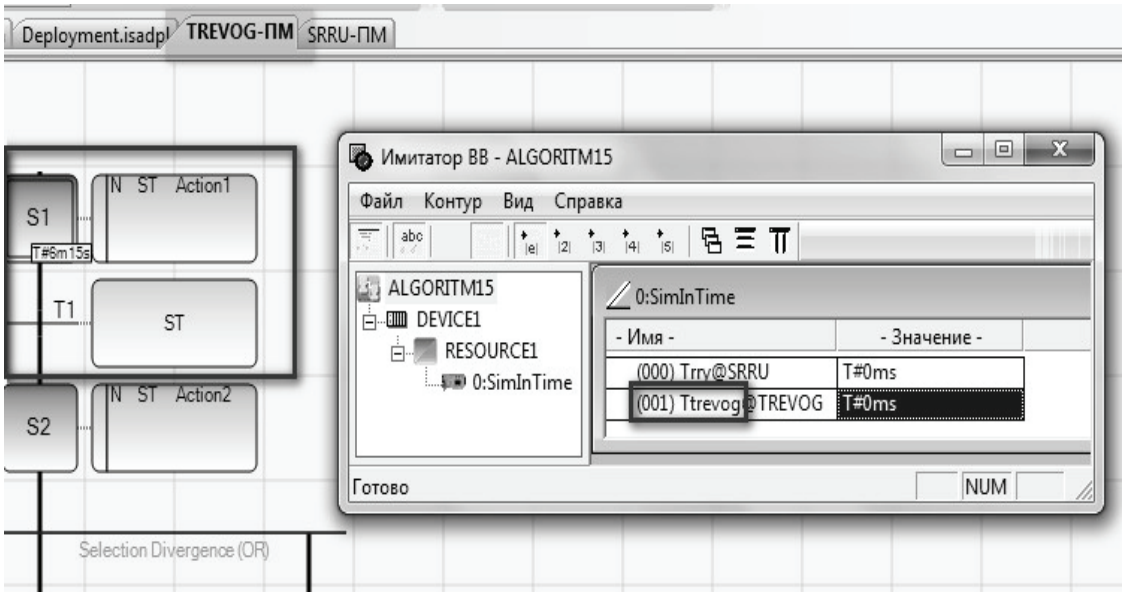


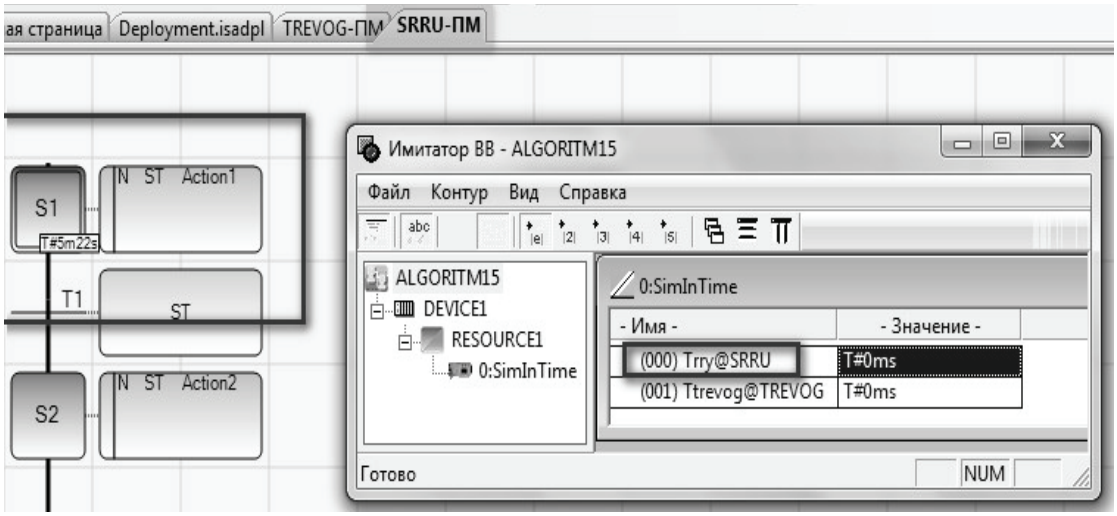
Рис. 72. Отладка программы

Первоначально переменные равны нулю и поэтому программы не выполняются (рис. 73, *а*, *б*).

После того, как мы зададим переменным значение, начнется выполнение алгоритма.



а



б

Рис. 73. Программа:
а – TREVOG; *б*– SRRU

Если мы зададим переменной Trevog время 10 с, то увидим, что каждый шаг программы TREVOG будет выполняться 10 с (рис. 74).

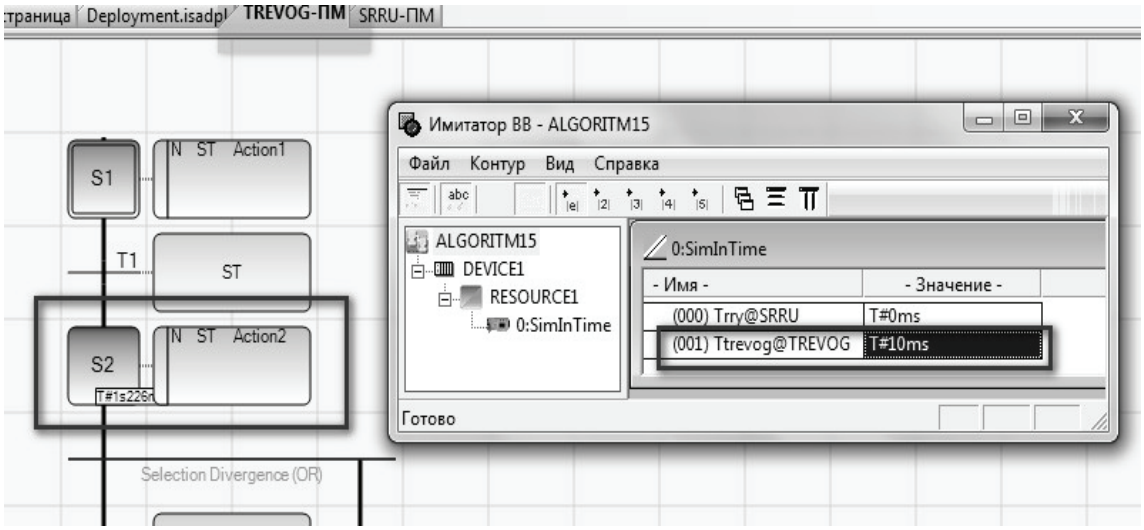


Рис. 74. Выполнение программы TREVOG

Теперь зададим время 15 с переменной Trry и увидим работу программы SRRU (рис. 75).

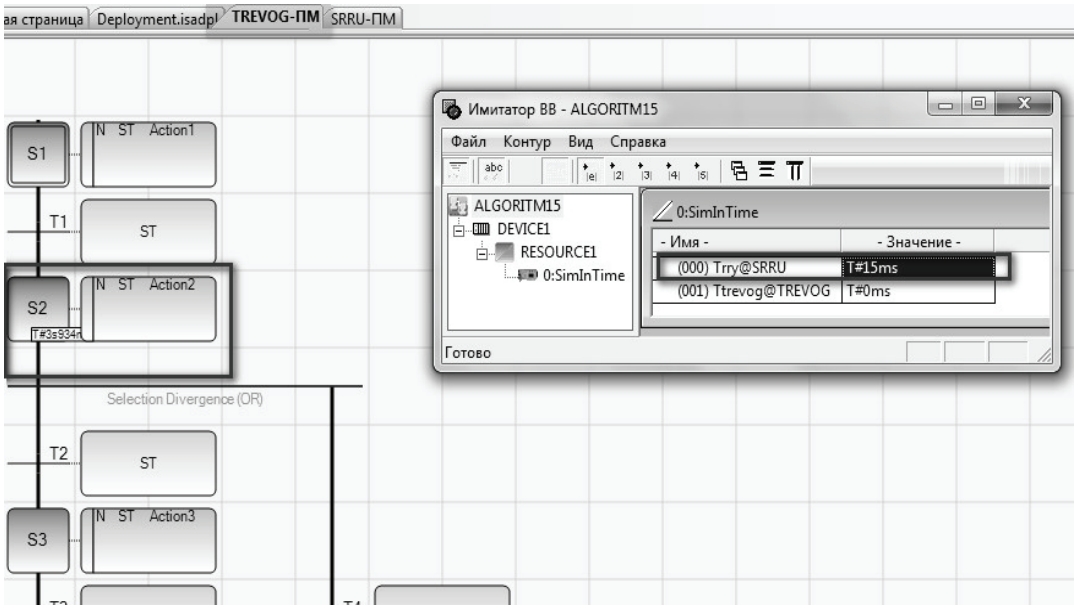


Рис. 81. Отладка программы SRRU

6. СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ КУРСОВОГО ПРОЕКТА

Курсовой проект должен включать программное средство и пояснительную записку.

Структура пояснительной записки следующая:

1. Титульный лист.
2. Задание на курсовой проект.
3. Реферат.
4. Содержание.
    - 4.1.Введение.
    - 4.2.Основные разделы:
    - 4.2.1. Исходные данные.
    - 4.2.2. Модель предметной области.
    - 4.2.3. Структура проекта.
    - 4.2.4. Словарь проекта.
    - 4.2.5. Исходные коды программ.
    - 4.2.6. Сценарий и результаты моделирования.
    - 4.3.Заключение.
5. Перечень графического, иллюстрационного материала.
    - 5.1.Структурная схема блока управления.
    - 5.2.Функциональная схема блока управления.
    - 5.3.Модель предметной области модуля управления.
    - 5.4.Модель модуля управления в Excel.
    - 5.5.Исходный код модуля управления.
6. Список использованных источников.
7. Приложения.

Введение должно содержать оценку современного состояния решаемой проблемы, ее актуальность и практическую значимость, описание основной цели и подчиненные ей более частные задачи.

В заключении приводятся краткие выводы по результатам работы, разработка рекомендаций и исходных данных по конкретному использованию, перспективы дальнейшего развития.

Список использованных источников включает все источники, записанные в порядке появления ссылок на них в тексте пояснительной записки. Ссылки в тексте на литературные источники обязательны. Сведения об источниках, включенных в список, следует приводить в соответствии с требованиями ГОСТ 7.1.

Приложения содержат:

- листинги программ;
- примеры работы;
- копии экрана;
- концептуальную модель, логическую модель и т. д.

К оформлению основного текста пояснительной записки предъявляются следующие требования.

Титульный лист следует оформлять в соответствии с приложением 1.

Задание на курсовой проект оформляется в соответствии с приложением 2.

Титульный лист и задание на курсовой проект не нумеруются, но в общую нумерацию входят.

Пояснительная записка оформляется на стандартных листах формата А4 на одной стороне листа. Поля страницы: правое – 10 мм; верхнее – 20 мм; левое – 23 мм; нижнее – 15 мм. Цвет шрифта – черный. Текст пояснительной записки следует печатать шрифтом Times New Roman, размером 14 пт, через одинарный междустрочный интервал. В случае вставки в строку формул допускается увеличение междустрочного интервала. Размер шрифта символов в формулах и уравнениях, заголовках разделов, заголовках и подрисуночных подписях иллюстраций, заголовках и тексте таблиц должен соответствовать размерам основного шрифта текста.

Допускается использование сокращений и аббревиатур.

Текст пояснительной записки набирается с абзацного отступа, равного 15 мм. Размеры полей и абзацных отступов должны быть одинаковыми на протяжении всего текста пояснительной записки. Номера страниц ставятся в правом нижнем углу без точки. В общую нумерацию страниц включают титульный лист, бланк задания, все листы работы, иллюстрации и таблицы, расположенные на отдельных листах, а также приложения.

Каждый раздел начинается с новой страницы. Заголовки элементов записки «Содержание», «Введение», «Заключение», «Список использованных источников», «Приложение» следует записывать в начале соответствующих страниц строчными буквами, кроме первой прописной, полужирным шрифтом с выравниванием по центру.

Заголовки структурных элементов и разделов основной части следует располагать с абзацного отступа, набирать полужирным шрифтом, строчными буквами, кроме первой прописной буквы.

В конце заголовков точку не ставят. Перенос слов в заголовках запрещен. Если заголовок занимает более одной строки, то последующие его строки должны быть записаны без абзацного отступа. Если заголовок состоит из двух предложений, то их разделяют точкой.

Нумерация заголовков выполняется арабскими цифрами. Подразделы должны иметь порядковую нумерацию внутри раздела, нумерация подраздела состоит из номеров раздела и подраздела, разделенных точкой. В конце точка не ставится. Номер пункта состоит из номеров раздела, подраздела и пункта, разделенных точками.

Заголовки разделов должны быть отделены от текста интервалом 18 пт, заголовки подразделов и пунктов: сверху – интервалом 18 пт, снизу – интервалом 12 пт. Соседние, последовательно записанные заголовки раздела и подраздела следует отделять друг от друга интервалом 12 пт, а подраздела и пункта – интервалом 6 пт.

Запрещено переносить заголовки подразделов и пунктов с листа на лист, а также записывать их в конце текста, если после указанных заголовков на листе размещается меньше двух строк текста.

В тексте не допускается употреблять обороты разговорной речи, применять для одного и того же понятия различные термины, сокращения слов, кроме установленных правилами орфографии. Перечень допускаемых сокращений русских слов оговорен в ГОСТ 2.316 и ГОСТ 7.12, белорусских – в СТБ 7.12.

Пункты перечисления записываются после двоеточия с абзацного отступа, после каждого пункта ставится точка с запятой, после последнего – точка. Перед каждым пунктом перечисления следует ставить тире.

Таблицу следует располагать в записке непосредственно после текста, в котором она упоминается. На все таблицы документа в тексте должны быть приведены ссылки. Таблица при необходимости может иметь заголовок, который выполняется строчными буквами (кроме первой прописной).

При переносе части таблицы на другую страницу слово «Таблица» и ее название помещают только над первой частью таблицы, над последующими частями таблицы пишут слева «Продолжение таблицы» с указанием ее номера.

В таблице допускается применять шрифт на 1–2 пт меньший, чем в тексте.

В формулах и уравнениях в качестве символов следует применять обозначения, установленные стандартами или принятые

в данной отрасли. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него.

Формулы и уравнения выравниваются по центру.

Все формулы и уравнения нумеруются арабскими цифрами сквозной нумерацией или в пределах раздела. В случае нумерации в пределах раздела номер формулы состоит из номера раздела и порядкового номера формулы, разделенных точкой. Номер указывают в круглых скобках с правой стороны листа на уровне формулы. Ссылки в тексте на порядковые номера формул дают в скобках, например «…в формуле 1.1)…».

Иллюстрации (фотографии, рисунки, чертежи, схемы, диаграммы, графики, карты и др.) служат для наглядного представления характеристик объектов исследования, полученных теоретических и (или) экспериментальных данных и выявленных закономерностей. Не допускается одни и те же результаты представлять в виде иллюстраций и таблицы.

Иллюстрации следует располагать непосредственно после абзаца, содержащего ссылку на них. Если для размещения иллюстраций недостаточно места на соответствующей странице, необходимо разместить ее в начале следующей страницы. На все иллюстрации должны быть даны ссылки в тексте.

Иллюстрации следует нумеровать сквозной нумерацией или в пределах каждого раздела арабскими цифрами. При нумерации в пределах раздела номер рисунка включает в свой состав номер раздела и порядковый номер рисунка по разделу, разделенных точкой, например «Рисунок 1.2». Иллюстрации отделяются от текста интервалом 14 пт.

Ссылки на источники в тексте осуществляются путем приведения порядкового номера в соответствии с библиографическим списком. Номер источника по списку заключается в квадратные скобки, например [2].

Приложения должны иметь общую с остальной частью записки нумерацию страниц. В тексте документа на все приложения должны быть ссылки. Приложения располагаются в порядке ссылок на них. Все приложения должны быть перечислены в содержании документа с указанием их номера и заголовка.

Каждое приложение следует начинать с нового листа с указанием в правом верхнем углу слова «Приложение», напечатанного прописными буквам. Приложение должно иметь содержательный заголовок, который размещается с новой строки по центру листа с прописной буквы.

Приложения обозначаются прописными буквами русского алфавита, начиная с А (за исключением Е, З, Й, О, Ч, Ь, Ы, Ъ); например ПРИЛОЖЕНИЕ А. Допускается обозначать приложения буквами латинского алфавита, за исключением I и O.

7. ТЕМЫ КУРСОВЫХ ПРОЕКТОВ

1. Разработка проекта модуля управления (ИМС УУ) на языке LD.
2. Разработка проекта модуля питания (ИМС ПИТ) на языке LD.
3. Разработка проекта модуля контакта отделения (ИМС КО) (Входы-Выходы) на языке LD.
4. Разработка проекта модуля контакта отделения (ИМС КО) (Память) на языке LD.
5. Разработка проекта модуля раскрывающихся элементов (ИМС РЭК) (Входы-Выходы) на языке LD.
6. Разработка проекта модуля раскрывающихся элементов (ИМС РЭК) (Память) на языке LD.
7. Разработка проекта модуля коррекции (ИМС СК) (Входы-Выходы) на языке LD.
8. Разработка проекта модуля коррекции (ИМС СК) (Память) на языке LD.
9. Разработка проекта модуля управления (ИМС УУ) на языке ST.
10. Разработка проекта модуля питания (ИМС ПИТ) на языке ST.
11. Разработка проекта модуля контакта отделения (ИМС КО) на языке ST.
12. Разработка проекта модуля раскрывающихся элементов (ИМС РЭК) на языке ST.
13. Разработка проекта модуля системы коррекции (ИМС СК) на языке ST.
14. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 2.
15. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 5.
16. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 6.
17. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 7.
18. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 8.
19. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 9.
20. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 10.

21. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 11.
22. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 12.
23. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 13.
24. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 14.
25. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 15.
26. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 16.
27. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 17.
28. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 18.
29. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 19.
30. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 20.
31. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 21.
32. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 22.
33. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 23.
34. Разработка программного обеспечения приборного модуля (ЦПМ) на языке SFC по алгоритму 24.

ПРИЛОЖЕНИЕ 1

ПРИМЕР ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ КУРСОВОГО ПРОЕКТА

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет издательского дела и полиграфии
Кафедра информационных систем и технологий
Специальность 1-40 01 02-03
Специализация «Издательско-полиграфический комплекс»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Специализированные информационные системы»
Тема_____
_____

Исполнитель
студент(ка) 4 курса группы 8 _____ А. С. Иванов
подпись, дата

Руководитель
доцент, канд. техн. наук _____ С. И. Акунович
подпись, дата

Курсовой проект защищен с оценкой_____
Руководитель _____ С. И. Акунович
подпись, дата

Минск 2014

ПРИЛОЖЕНИ 2

ПРИМЕР ЗАДАНИЯ НА КУРСОВОЙ ПРОЕКТ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет издательского дела и полиграфии
Кафедра информационных систем и технологий
Специальность 1-40 01 02-03
Специализация «Издательско-полиграфический комплекс»

«Утверждаю»
Заведующий кафедрой ИСиТ
_____ П. П. Урбанович
подпись
«___»_____201__г.

ЗАДАНИЕ
на курсовой проект

студенту(тке) _____

1. Тема _____
   _____
   _____
2. Срок защиты _____
3. Исходные данные _____
   _____
   _____
   _____
   _____
   _____

4. Содержание расчетно-пояснительной записки курсового проекта (перечень вопросов, подлежащих разработке)

_____
_____
_____
_____
_____
_____
_____

5. Перечень графического, иллюстрационного материала

_____
_____
_____
_____
_____

6. Консультанты (с указанием разделов)

_____
_____
_____
_____

7. Календарный график работы

_____
_____
_____
_____

8. Дата выдачи задания _____

Руководитель _____ С. И. Акунович
подпись

Задание принял к исполнению _____
дата и подпись

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Акунович, С. И. Композиционное конструирование моделей систем дискретного управления в среде DELPHI: методические указания / С. И. Акунович. – Минск: БГТУ, 2009. – 78 с.

2. Акунович, С. И. Дискретные системы логического управления технологических машин / С. И. Акунович, А. А. Гончаров, Ю. Н. Петренко. – Минск: Юнипак, 2006. – 142 с.

3. Уокенбах, Дж. Excel 2007. Профессиональное программирование на VBA / Дж Уокенбах; пер. с англ. – М.: Издательский дом «Диалектика», 2008. – 928 с.

4. Микропроцессорные и цифровые устройства полиграфического оборудования [Электронный ресурс] / Режим доступа: http://text-totext.ru/metodichka/metodichka-1.html. – Дата доступа: 08.05.2013.

5. Леоненков, А. Самоучитель UML / А. Леоненков. – СПб.: BHV-Санкт-Петербург, 2002. – 304 с.

6. Семенов, Е. М. Проектирование систем управления в среде технологического программирования ISaGRAF / Е. М. Семенов. [Электронный ресурс] – 2009. Режим доступа: http://www.spb-lta-kafapp.narod.ru/Metoda3.pdf. – Дата доступа: 08.05.2013.

7. ISaGRAF. Version 5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ. [Электронный ресурс]. – Режим доступа: http://mosinv.ru/Documentation/XT-PRO/ISaGRAF.pdf. – Дата доступа: 08.05.2013.

ОГЛАВЛЕНИЕ

СПЕЦИАЛИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Методические указания

Составители: **Акунович** Станислав Иванович
Брусенцова Татьяна Палладьевна

Редактор *Ю. Д. Нежикова*
Компьютерная верстка *Я. Ч. Болбот*
Корректор *Ю. Д. Нежикова*

Издатель:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.