

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

«УТВЕРЖДАЮ»
Декан АВТФ
_____ С.А. Гайворонский
_____ “ ” _____ 2008г.

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО
ПРОМЫШЛЕННЫМ КОНТРОЛЛЕРАМ**

методические указания по выполнению лабораторных работ

Издательство
Томского политехнического университета
2008

УДК 681.3.06

Лабораторный практикум предназначен для выполнения студентами циклов лабораторных работ по дисциплине инновационной образовательной программы магистерской подготовки «Управление в технических (мехатронных) системах»: средства автоматизации гибких автоматизированных производств / сост. С.В. Леонов, В.А. Рудницкий– Томск: Изд-во Томского политехнического университета, 2008. – 120 с.

Методические указания рассмотрены и рекомендованы к изданию методическим семинаром кафедры интегрированных компьютерных систем управления факультета автоматики и вычислительной техники «_____»_____ 2008г.

Зав. кафедрой

проф., доктор техн. наук

_____ А.М. Малышенко

СОДЕРЖАНИЕ

1. СИСТЕМА ПРОГРАММИРОВАНИЯ ПЛК OPENPCS	4
2. СОЗДАНИЕ ПРОГРАММ И ИХ ОТЛАДКА В КОНТРОЛЛЕРЕ ELSY-TM	23
3. ОБЩИЕ СВЕДЕНИЯ О ПЛК SIMATIC	30
4 ЗАПУСК SIMATIC MANAGER И СОЗДАНИЕ ПРОЕКТА	33

1. СИСТЕМА ПРОГРАММИРОВАНИЯ ПЛК OPENPCS

Цель работы: Создание ресурса, задач, программ на языках стандарта IEC 6 1131-3 и их отладка в PLC-симуляторе OpenPCS 2004. Все программы выполняют одну и ту же задачу, хорошо известную по лабораторным работам верхнего уровня SCADA-системы – реализация алгоритма «Пуск-Стоп».

О стандарте IEC 6 1131-3

Стандарт IEC 6 1131-3 описывает синтаксис и семантику пяти языков программирования ПЛК, - языков, ставших широко известными за более чем 30-летнюю историю их применения в области автоматизации промышленных объектов:

1. SFC (Sequential Function Chart) - графический язык, используемый для описания алгоритма в виде набора связанных пар: шаг (step) и переход (transition). Шаг представляет собой набор операций над переменными. Переход - набор условных логических выражений, определяющий передачу управления к следующей паре шаг-переход. По внешнему виду описание на языке SFC напоминает хорошо известные логические блок-схемы алгоритмов. SFC имеет возможность распараллеливания алгоритма. Однако SFC не имеет средств для описания шагов и переходов, которые могут быть выражены только средствами других языков стандарта. Происхождение: Grafset (Telemecanique-Groupe Schneider).

2. LD (Ladder Diagram) - графический язык программирования, являющийся стандартизованным вариантом класса языков релейно-контактных схем. Логические выражения на этом языке описываются в виде реле, которые широко применялись в области автоматизации в 60-х годах. Ввиду своих ограниченных возможностей язык дополнен привнесенными средствами: таймерами, счетчиками и т.п. Происхождение: различные варианты языка релейно-контактных схем (Allen-Bradley, AEG Schneider Automation, GE-Fanuc, Siemens).

3. FBD (Functional Block Diagram) - графический язык по своей сути похожий на LD. Вместо реле в этом языке используются функциональные блоки, по внешнему виду - микросхемы. Алгоритм работы некоторого устройства на этом языке выглядит как функциональная схема электронного устройства: элементы типа "логическое И", "логическое ИЛИ" и т.п., соединенные линиями. Корни языка выяснить сложно, однако большинство специалистов сходятся во мнении, что это не что иное, как перенос идей языка релейно-контактных схем на другую элементную базу.

4. ST (Structured Text) - текстовый высокоуровневый язык общего назначения, по синтаксису ориентированный на Паскаль. Происхождение: Grafset (Telemecanique-Groupe Schneider).

5. IL (Instruction List) - текстовый язык низкого уровня. Выглядит как типичный язык Ассемблера, что объясняется его происхождением: для некоторых моделей ПЛК фирмы Siemens является языком Ассемблера. В рамках стандарта IEC 6 1131-3 к архитектуре конкретного процессора не привязан. Происхождение - STEP 5 (Siemens).

Перечисленные языки IEC 6 1131-3 используются ведущими фирмами изготовителями ПЛК, имеют длительную историю применения, достаточно распространены и известны пользователям по тем или иным модификациям. Несмотря на то, что во многих случаях такие модификации незначительны, это влечет определенные неудобства при работе с ПЛК различных фирм-изготовителей. С этой точки зрения, стандарт IEC 6 1131-3, несомненно, прогрессивен, поскольку позволяет привести бесчисленное число различных вариантов и интерпретаций языков ПЛК к единому знаменателю. OpenPCS представлен в виде двух частей: набора средств разработки и исполняемого на целевом ПЛК ядра-интерпретатора. Набор средств разработки выполняется на компьютере проектировщика, например, компьютере типа IBM PC, и состоит из редактора, отладчика и препроцессора, который подготавливает описанный проектировщиком алгоритм к формату, "понятному" ядру-интерпретатору. Этот набор имеет современный пользовательский интерфейс, позволяет тестировать алгоритм в режиме эмуляции и получать листинг алгоритма на языках его описания. После создания, пользовательская программа совместно с ядром-интерпретатором загружается в целевой ПЛК для исполнения. Ядро-интерпретатор, как следует уже из его названия, транслирует пользовательский алгоритм во время исполнения. Это позволяет сконцентрировать машинно-зависимый код и таким образом снизить накладные расходы при переходе на другой ПЛК. Неплохой подход, однако, сразу необходимо отметить, что интерпретационная модель имеет недостаток - она всегда снижает показатели эффективности исполнения программы.

Для исполняющей системы контроллер с загруженной программой представляется приведенным на рисунке образом:

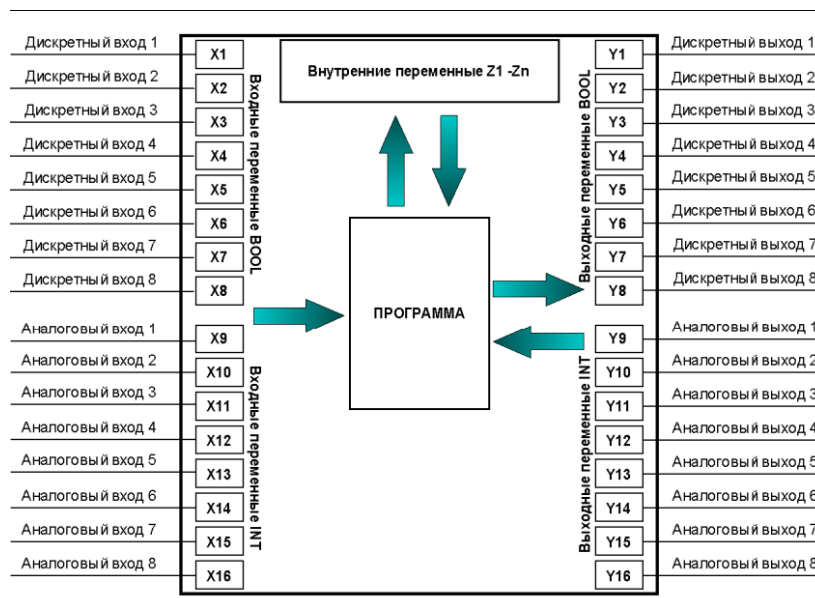



Рис.1 Обобщенная внутренняя структура контроллера

Программа, исполняющаяся в контроллере, получает информацию из внешней среды через переменные X, определенные как входные переменные.

На основе полученных данных исполняющая система производит определенные программой действия и выводит результат через переменные Y, определенные как выходные. Для промежуточных вычислений служат внутренние переменные Z.

УПРАЖНЕНИЕ 1

Реализация программы «Старт-стоп» на языке ST

1. Запустите OpenPCS (ярлык на рабочем столе  или программная группа infoteam openpcs 2004 в главном меню). OpenPCS требует файл с расширением VAR – файл проекта. Возможна показанная на рис. 2 ситуация, когда по умолчанию предлагается загрузить демонстрационный проект.

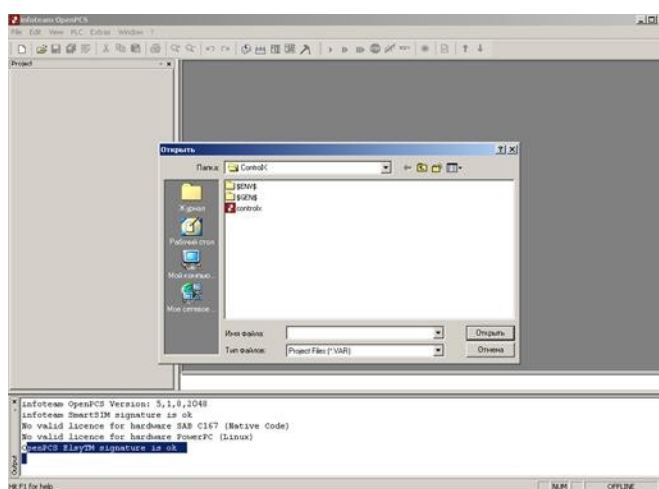


Рис.2. Начало работы с программным пакетом

2. Если рабочий проект в OpenPCS не создавался, выберите предлагаемый проект (Рис.3) либо загрузите его по указанному ниже пути. Самостоятельно ознакомьтесь с демонстрационными проектами, находящимися в директории C:\Program Files\OpenPCS2004\Samples\

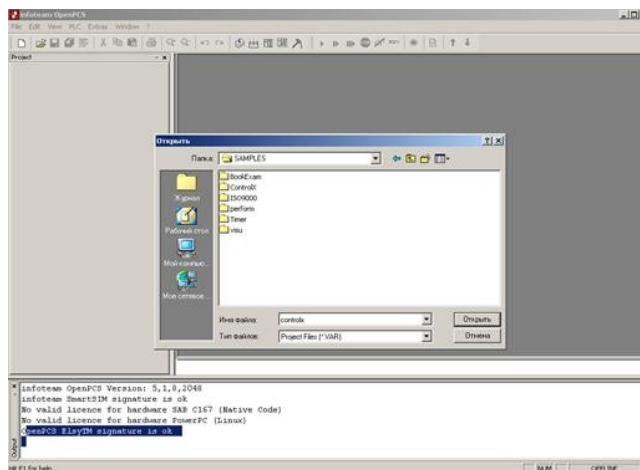


Рис. 2. Загрузка рабочего проекта

3. Создайте новый проект как показано на рис.4.: (File-> Project -> New) с именем Familiya (Ваша фамилия) в директории C:\ws 326- xx

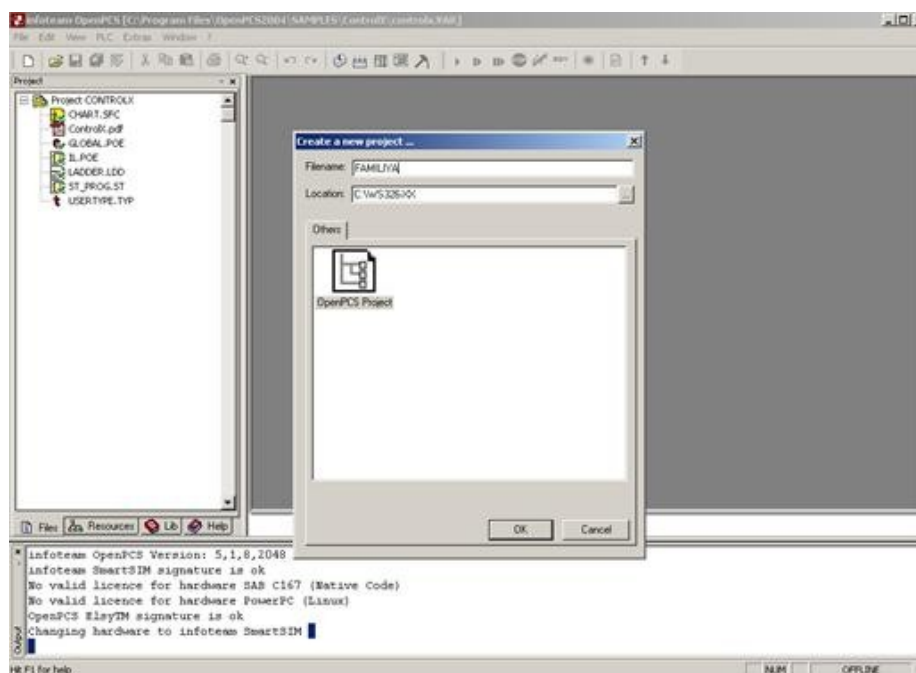


Рис.4. Создание нового проекта

Путь к проекту не должен содержать имен файлов и папок с кириллицей, личная директория в работах с OpenPCS будет рассматриваться как контейнер для хранения проектов, но не для работы с ними.

4. Создайте новую программу на ST (File->New) с именем ST1, на предложение о добавлении созданной программы к активному ресурсу ответить отказом (Рис.5.). Ознакомьтесь с ключевой для дальнейшей работы информацией:

- **Input** – переменная, подлежащая только считыванию из пользовательского кода, должна быть описана как входная переменная. Она не должна модифицироваться пользовательским кодом. Ключевое слово: **VAR_INPUT**;
- **Output** – переменная, которая является результатом какого-либо пользовательского кода, вычисляется этим кодом и должна подвергаться вызовам кода, должна быть описана как выходная переменная. Ключевое слово: **VAR_OUTPUT**;
- **Global** – переменная, которая должна быть доступной для более чем одного пользовательского кода и при этом не передаваться как аргумент, следует описывать как глобальную. Это возможно только в блоках типа PROGRAM . Другой пользовательский код, желающий иметь доступ к таким переменным, необходимо описывать как внешние (см.ниже). Ключевое слово: **VAR_GLOBAL**;
- **Extern** – для доступа из пользовательского кода к глобальной переменной. Эта переменная должна быть описана в пользовательском коде как внешняя. Ключевое слово: **VAR_EXTERNAL**;

- **АТ** – переменная, которая будет представлять физический вход или выход, может быть отражена на этот физический адрес.

Пример: булевская (BOOL) выходная (Q) переменная Motor имеет физический адрес (0.0) в адресном пространстве некоторого устройства.

VAR

Motor AT %Q0.0:BOOL;

END_VAR

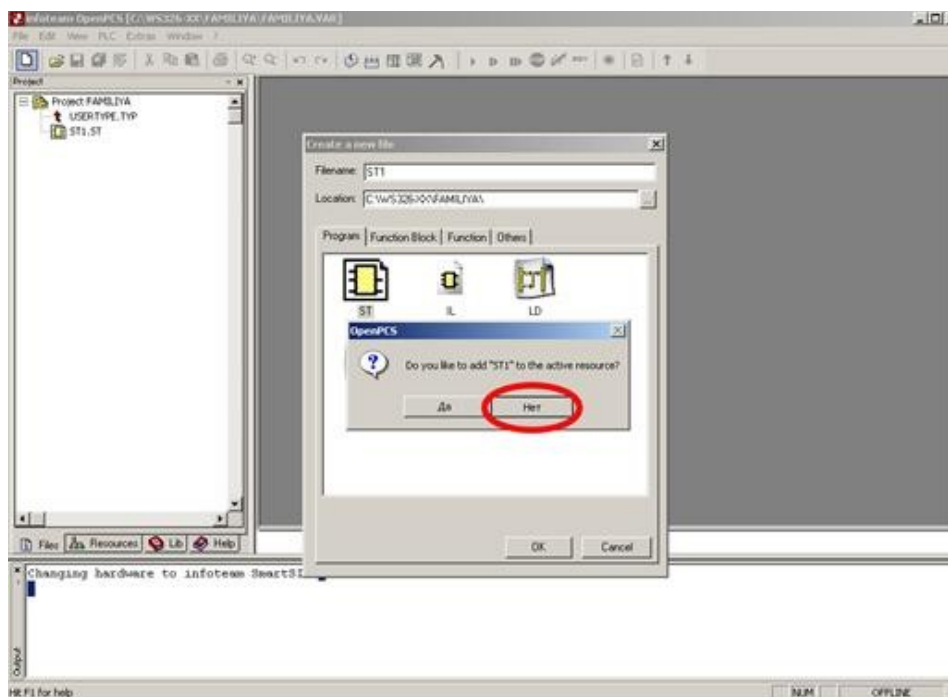


Рис. 5

Секция	Функция	Функциональный блок	Программа
VAR_INPUT	(1)	(1)	
VAR_OUTPUT		(1)	
VAR_GLOBAL			(1)
VAR_EXTERNAL		(1)	

Примечание: (1) – может использоваться, но лишь однократно.

На рис 6. представлен скриншот основных полей:

1. МЕНЮ И ПАНЕЛЬ ИНСТРУМЕНТОВ
2. МЕНЕДЖЕР ПРОЕКТОВ
3. РЕДАКТОР ПЕРЕМЕННЫХ
4. РЕДАКТОР КОДА
5. ОКНО ДИАГНОСТИКИ И ТЕСТИРОВАНИЯ

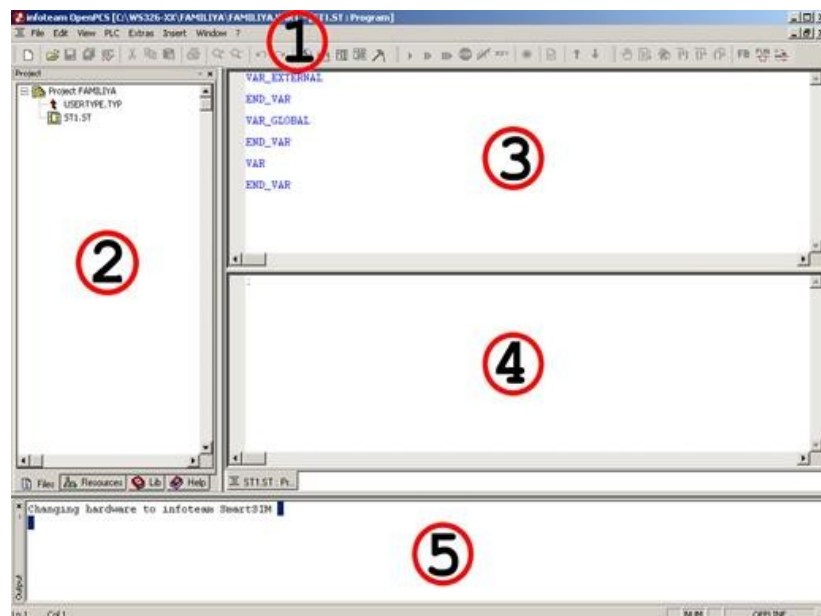


Рис. 6. Основные поля программного пакета OpenPCS

5. В редакторе переменных VAR опишите локальные переменные:

Название сигнала	Аппаратный адрес	Тип переменной
Valve_In_ST	AT%I0.0	Bool
Reset_ST	AT%I0.1	Bool
Pump_In_ST	AT%I0.2	Bool
And1_ST	нет	Bool
And2_ST	нет	Bool
Valve_Control_ST	AT%Q0.0	Bool
Pump_Control_ST	AT%Q0.2	Bool

Примечание: Valve_In_ST – это говорящее имя переменной (задвижка_входной_программа на языке ST).

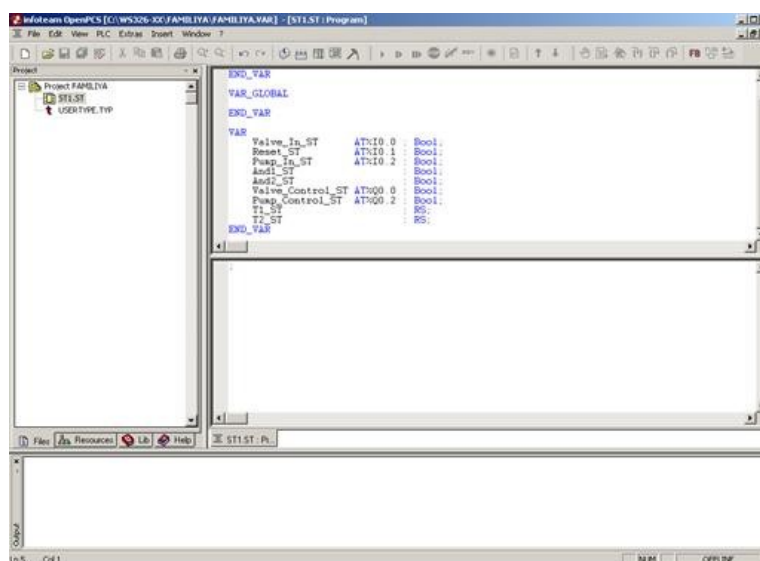


Рис. 7. Описание функциональных блоков RS –триггеров

6. Опишите функциональные блоки (Рис.7.) RS -триггеров (в сущности, это подпрограммы-процедуры) T1_ST:RS и T2_ST:RS: T1_ST – имя функционального блока, RS – его тип.

7. Введите текст программы:

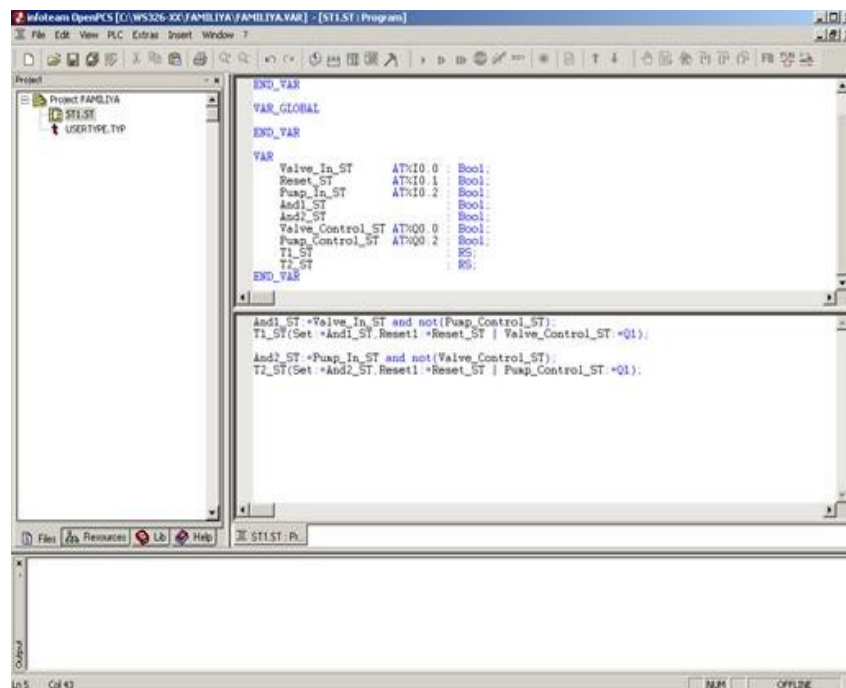


Рис. 7. Текст программы на языке ST

Комментарий к фрагменту программы, поясняющий ее работу:

А) And1_ST:=Valve_In_ST and not (Pump_Control_ST) - переменной And1_ST присваивается результат логического умножения (and) переменной Valve_in_ST и инверсного (not) значения переменной Pump_Control_ST.

Б) T1_ST (Set:=And1_ST, Reset1:=Reset_ST | Valve_Control_ST:=Q1) - входу установки триггера T1_ST в единичное состояние Set присваивается переменная And1_ST, входу сброса триггера Reset1 присваивается переменная Reset_ST. Это входы триггера. После разделителя | идет описание выхода Q1, которому присвоена переменная Valve_Control_ST.

Примечание: 3 и 4 строки выполняют аналогичную функцию, но для других переменных.

8. Теперь созданную программу необходимо проверить на синтаксические ошибки (File->Check Syntax(Alt+F10)) при отсутствии ошибок будет выдано следующее сообщение, рисунок 9.

9. Проверьте, что у вас существует активный ресурс (признак активного ресурса – ярко-зеленый цвет) и он не содержит программ, рисунок 10.



Рис. 9. Результаты проверки программы



Рис.10

10. Настройте ресурс для работы с симулятором промышленного контроллера.

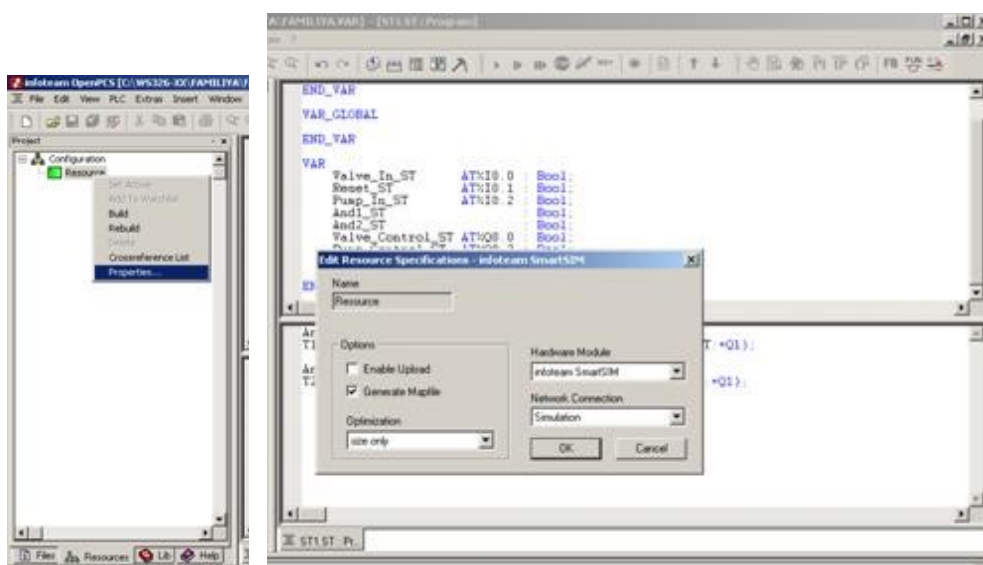


Рис. 11. Настройка ресурса для работы с симулятором

11. Свяжите написанную программу с активным ресурсом:



Рис. 12

12. Откомпилируйте активный ресурс, связанный с вашей задачей (программой):

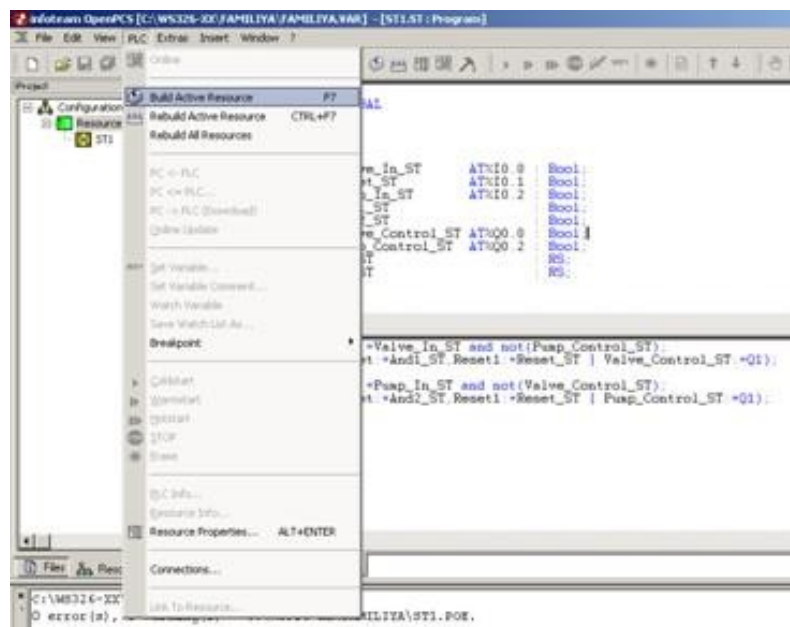


Рис. 13. Компиляция активного ресурса

При компиляции будет создан исполняемый код Start_Stop.PCD и сгенерированы сведения о наличии ошибок и предупреждений, как показано на рисунке 14.



Рис. 14

Свяжитесь с «контроллером» (PLC -симулятором), рисунок 15.

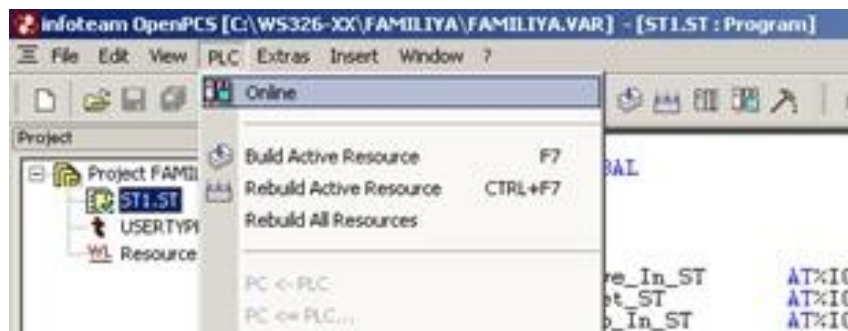


Рис. 15

13. Загрузите в симулятор код:



Рис. 16

14. Запустите симулятор (run). Из дерева сигналов добавьте двойным кликом необходимые входные и выходные переменные в окно мониторинга и проверьте работу программы, для этого изменяйте входные переменные (двойной клик по переменной) с соответствующими адресами и следите за значением выходных переменных в симуляторе и окне мониторинга OpenPCS.

15. На рисунке 18 показано распределение адресного пространства симулятора.

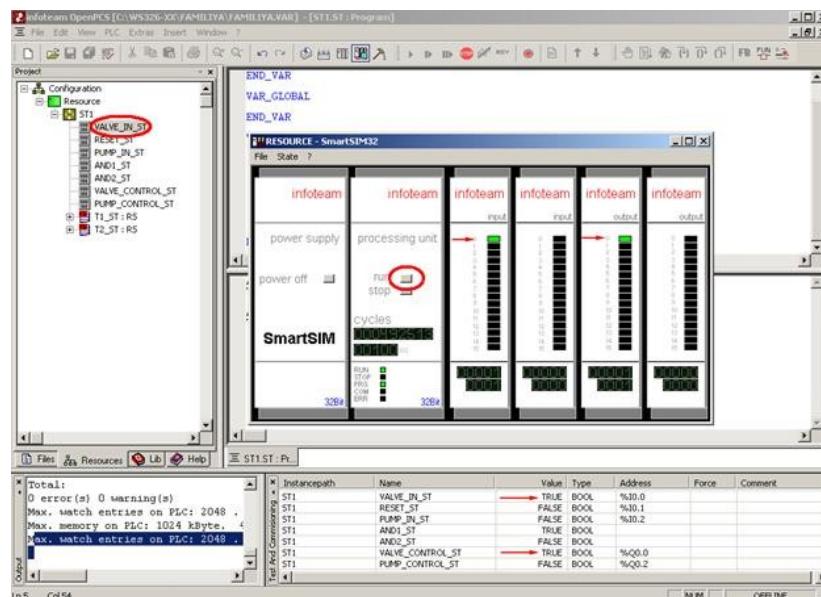


Рис. 17. Окно мониторинга OpenPCS

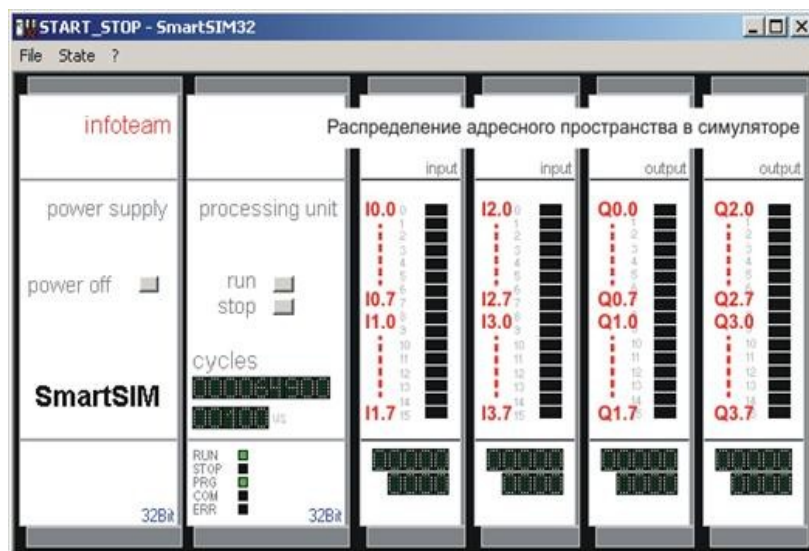


Рис. 18. Распределение адресного пространства симулятора

УПРАЖНЕНИЕ 2

Реализация программы «Старт-стоп» на языке IL

1. Остановите работу симулятора, закройте его и разорвите соединение. Создайте новый файл IL с именем IL1. Поскольку ресурс для работы с симулятором уже создан, то все последующие задачи будут добавляться к нему (рис.19).

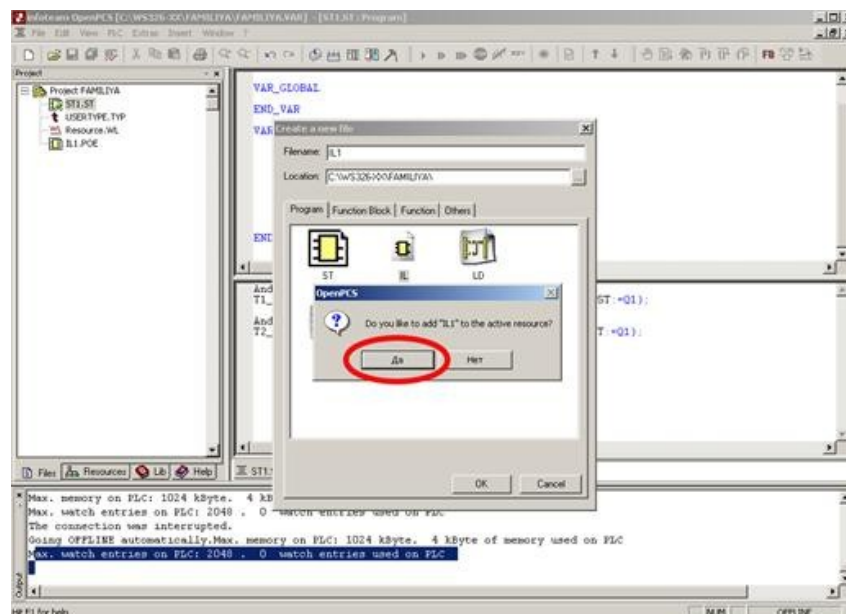


Рис. 19

2. Опишите переменные. Описание переменных носит такой же характер, как и в редакторе ST, изменим только имена переменных и их адреса, а так же имена функциональных блоков T1_IL и T2_IL.

Название сигнала	Аппаратный адрес	Тип переменной
Valve_In_IL	AT%I1.0	Bool
Reset_IL	AT%I1.1	Bool
Pump_In_IL	AT%I1.2	Bool
And1_IL	нет	Bool
And2_IL	нет	Bool
Valve_Control_IL	AT%Q1.0	Bool
Pump_Control_IL	AT%Q1.1	Bool

3. Напишите программу «Старт-стоп»:

LDN Pump_control_IL – инвертированная загрузка (LDN) переменной Pump_control_IL.

AND Valve_in_IL – ее логическое умножение на переменную Valve_in_IL.

ST AND1_IL – сохранение (ST) результата умножения в переменную AND1_IL.

CAL T1_IL (Set:=AND1_IL, Reset1:=Reset_IL | Valve_control_IL:=Q1) – вызов (CAL) процедуры (функционального блока в терминах OpenPCS), реализующей RS-триггер и присвоение его входам-выходам переменных.

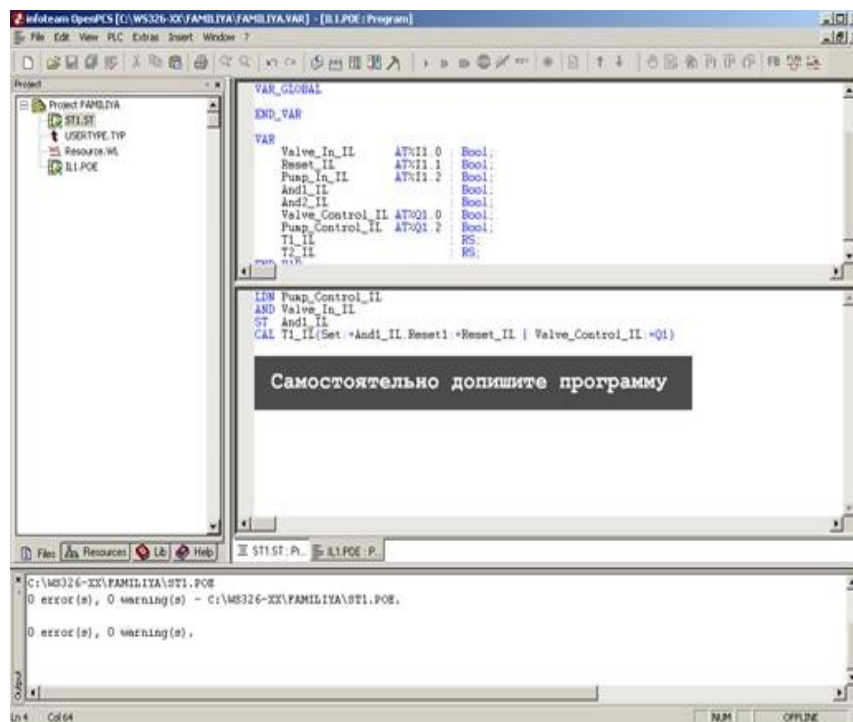


Рис. 20. Программа «Старт-стоп» на языке IL

4. Если вдруг программа не добавлена к ресурсу на этапе создания новой программы, то сделайте это принудительно согласно рисунку 21.

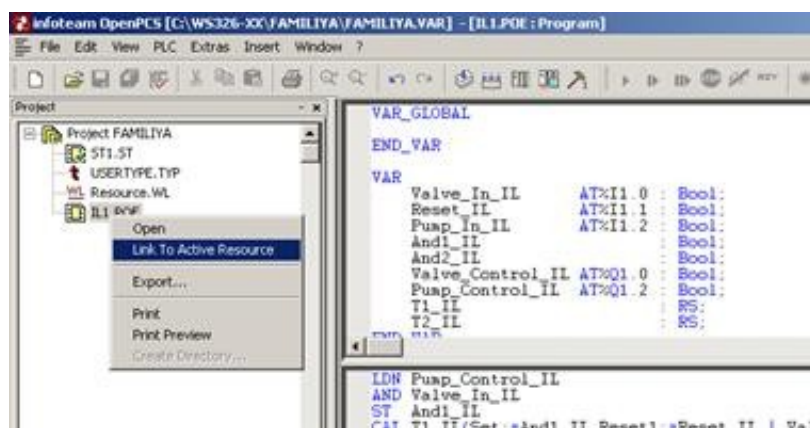


Рис. 21

5. Откомпилируйте программу, загрузите код в симулятор (теперь будут выполняться обе программы ST1 и IL1 в разном адресном пространстве симулятора), запустите его, добавьте необходимые переменные в окно мониторинга и проверьте работу по аналогии с предыдущим упражнением (предыдущая программа так же должна работать). Для идентификации адресов пользуйтесь рисунком 18.

УПРАЖНЕНИЕ 3

Реализация программы «Старт-стоп» на языке LD

1. Остановите работу симулятора и создайте новый файл LD с именем LD1 . Поскольку ресурс для работы с симулятором уже создан, то все последующие задачи будут добавляться к нему, если это не оговорено особо. Окно редактора LD (рис.22.) выглядит довольно специфично.

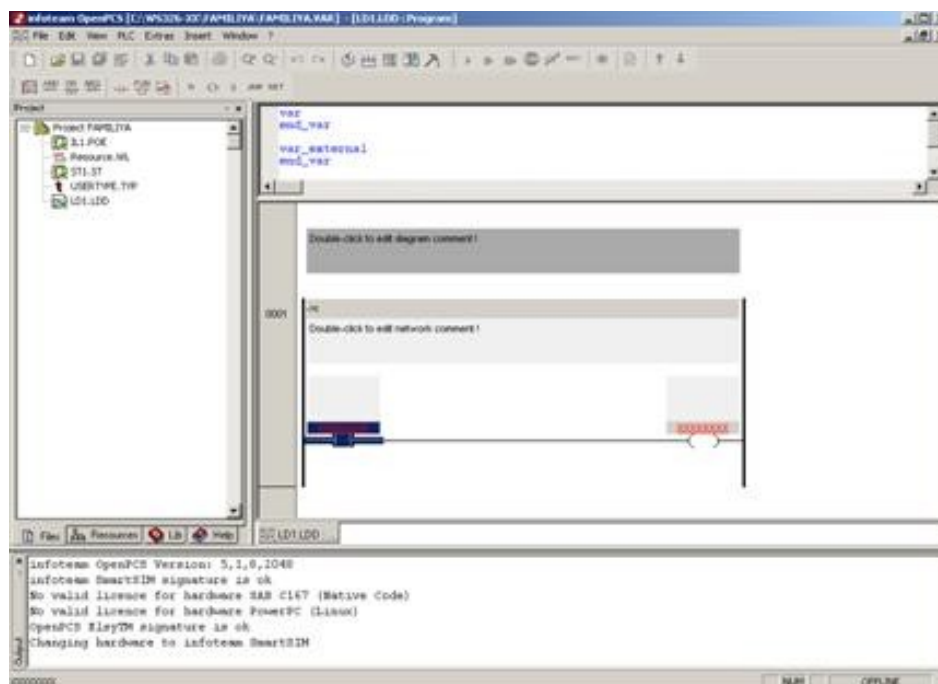


Рис. 22. Окно редактора LD

2. Элемент, показанный на рисунке 23, носит название элементарная сеть, это заготовка для создания элемента программы.



Рис. 23. Элементарная сеть

3. Опишите переменные. Описание переменных не отличается от редактора ST. Функциональные блоки - T1_LD и T2_LD. Обратите внимание на адреса сигналов.

Название сигнала	Аппаратный адрес	Тип переменной
Valve In LD	AT%I2.0	Bool
Reset LD	AT%I2.1	Bool
Pump In LD	AT%I2.2	Bool
And1 LD	нет	Bool
And2 LD	нет	Bool
Valve Control LD	AT%Q2.0	Bool
Pump Control LD	AT%Q2.2	Bool

4. Дополните сеть до элемента «И» и проинвертируйте один из «контактов».

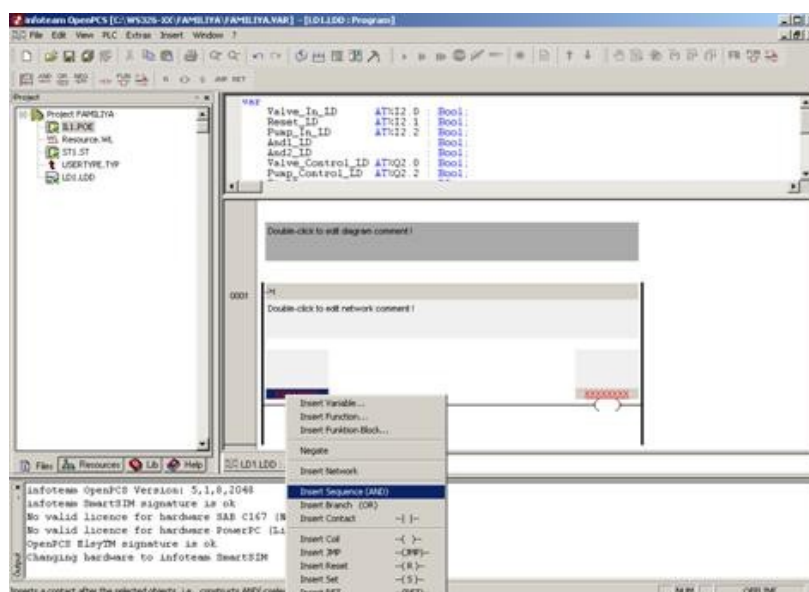


Рис. 24

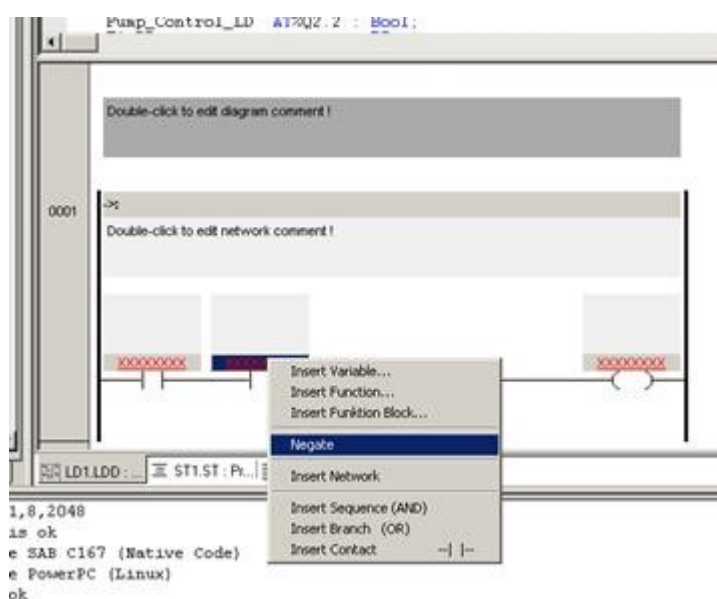


Рис. 25

5. Присвойте элементам сети переменные.

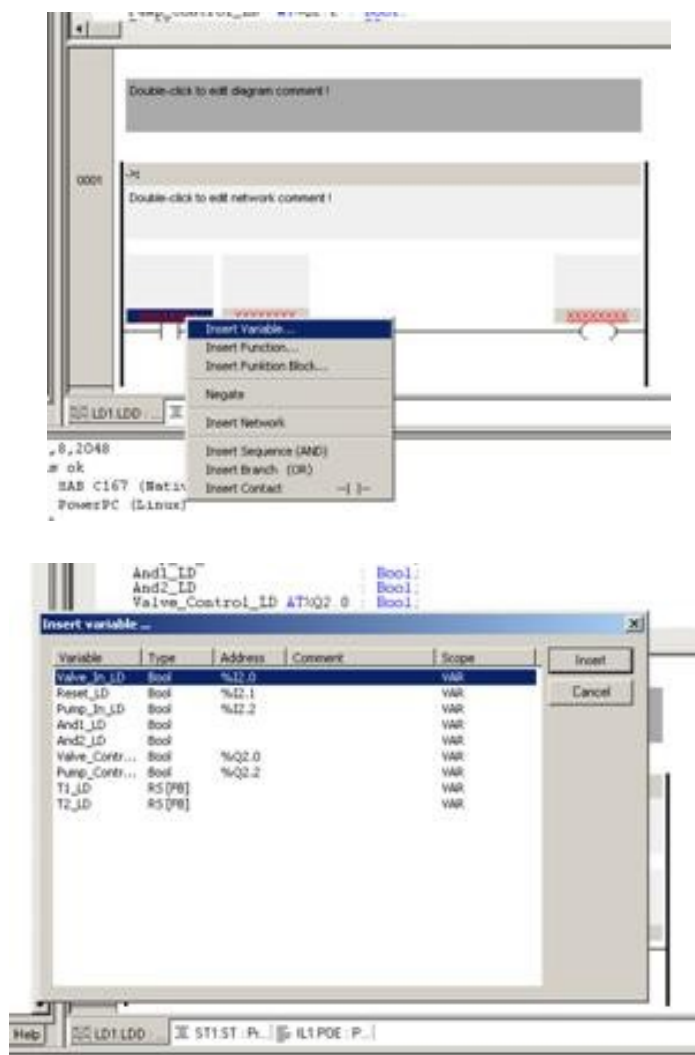


Рис. 26

6. Добавьте новую сеть.



Рис. 27

7. Вставьте функциональный блок RS -триггер:

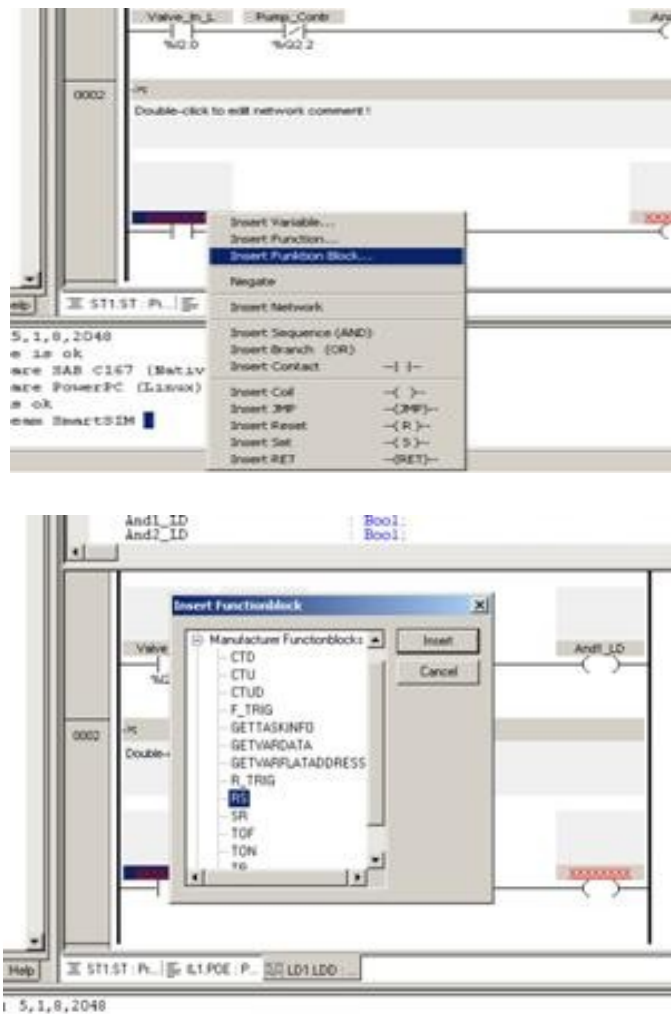


Рис. 28

8. Присвойте входам и выходу триггера соответствующие переменные, после чего фрагмент программы будет иметь вид.

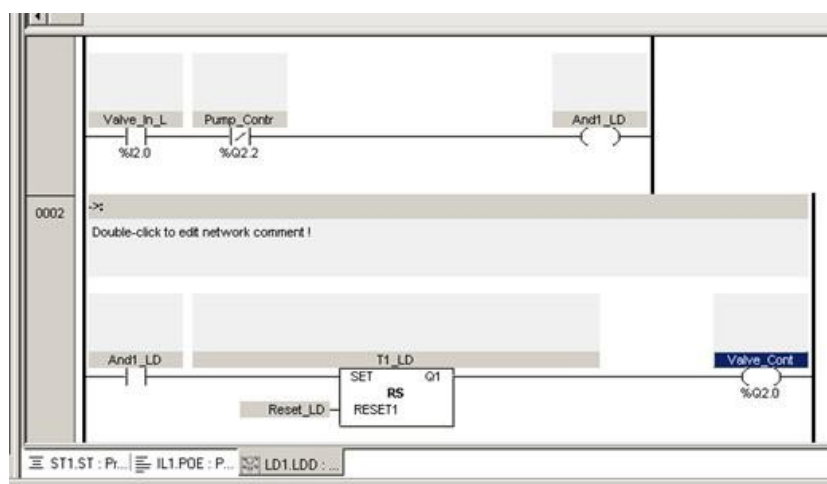


Рис. 29

9. Допишите остальную часть программы, откомпилируйте ее, загрузите в симулятор, запустите его, добавьте необходимые переменные в окно мониторинга и проверьте работу программы. На этом этапе в симуляторе должны выполняться программы ST1 , IL1 и LD1, каждая в своем адресном пространстве, убедитесь в этом.

УПРАЖНЕНИЕ 4

Реализация программы «Старт-стоп» на языке FBD

Так как этот графический язык ориентирован на создание программ по виду похожих на принципиальную блочную схему, то будет уместным привести задачу именно в подобном графическом виде.

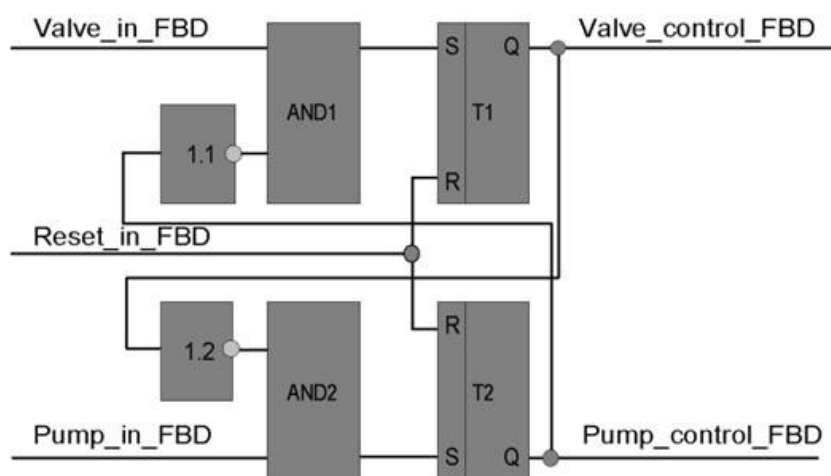


Рис. 30

1. Остановите симулятор и создайте новую программу FBD 1 на языке FBD, добавив ее к активному ресурсу. Включите сетку (View -> Grid (CFC)).

2. Опишите переменные. Описание переменных не отличается от редактора ST. Функциональные блоки - T 1_FBD и T 2_FBD . Обратите внимание на адреса сигналов.

Название сигнала	Аппаратный адрес	Тип переменной
Valve_In_FBD	AT%I3.0	Bool
Reset_FBD	AT%I3.1	Bool
Pump_In_FBD	AT%I3.2	Bool
And1_FBD	Net	Bool
And2_FBD	Net	Bool
Valve_Control_FBD	AT%Q3.0	Bool
Pump_Control_FBD	AT%Q3.2	Bool

3. В панели инструментов выберите Insert -> Block, NOT(*BOOL*) (логическое отрицание) и вставьте блок правым кликом мыши в левый верхний квадрат и добавьте остальные блоки.

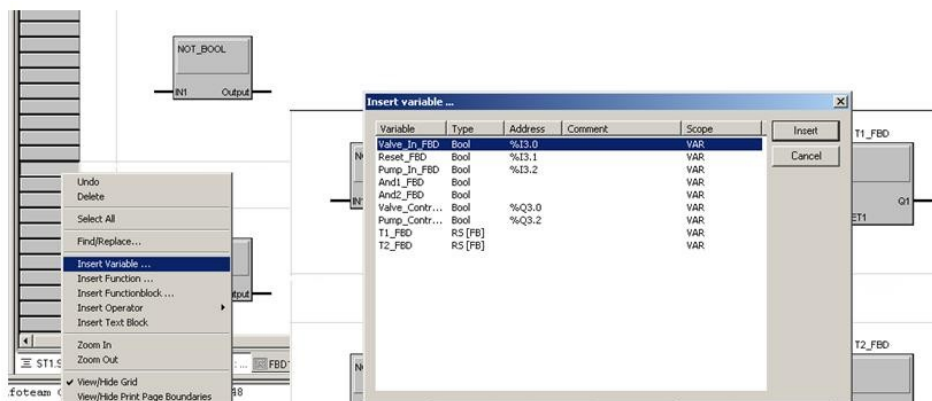


Рис. 31

4. Назначьте входам – входные, а выходам – выходные переменные:

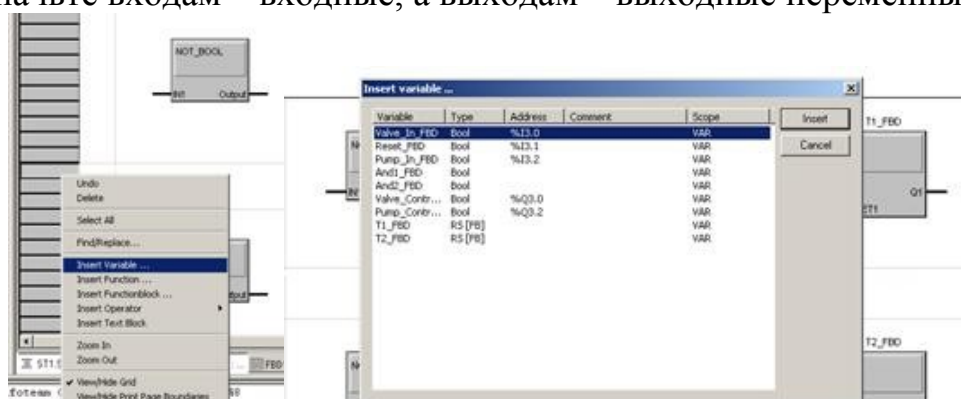


Рис. 32

5. Соедините блоки, выбрав мышкой необходимые выходы и применив инструмент Connection.

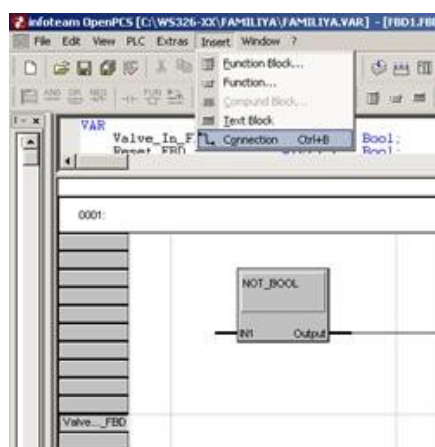


Рис. 33

6. Откомпилируйте программу, загрузите ее в симулятор и убедитесь в работоспособности всех четырех программ в адресном пространстве PLC -симулятора, защитите вашу работу. Закройте OpenPCS и переместите ваш проект директорию. На этом лабораторная работа закончена.

Контрольные вопросы

1. Почему для сигнала Valve_In_ в программах тип переменной указан как Bool?
2. Каким образом можно откомпилировать активный ресурс?
3. Сколько устройств, аналогичных рассмотренному в лабораторной работе, можно подключить к данному симулятору?
4. В каком случае удобнее использовать язык ST, а в каком FBD?

Требования по содержанию отчета

В отчете студент должен перечислить цели лабораторной работы, описать ход работы, ответить на контрольные вопросы, сделать вывод о проделанной работе.

2. СОЗДАНИЕ ПРОГРАММ И ИХ ОТЛАДКА В КОНТРОЛЛЕРЕ ELSY-TM


Цель работы: Создание программ на языках стандарта IEC 6 1131-3 и их отладка в контроллере Elsy-TM.

В данной лабораторной работе требуется создать, новый ресурс, функциональный блок (другими словами это процедура, которая может иметь сложную структуру внутри и выглядеть очень простой внешне, имеющей входы и выходы). Кроме этого необходимо выполнить программу при помощи функционального блока (ФБ) с указанием этого блока в программе, при этом отладка будет производиться не в PLC-симуляторе OpenPCS 2006, а непосредственно в контроллере Elsy-TM.

1. Скопируйте папку проекта FAMILIYA из личной директории в директорию ws143-xx .



Путь к проекту не должен содержать имен файлов и папок с кириллицей, личная директория в работах с OpenPCS будет рассматриваться как контейнер для хранения проектов, но не для работы с ними!

2. Запустите OpenPCS (ярлык на рабочем столе  или программная группа Infoteam OpenPCS 2006 в главном меню).

3. Откройте проект (File->Project->Open), путь C:\ws143-xx\FAMILIYA\FAMILIYA.VAR

4. Создайте новый ресурс с именем startstop (“File->New...”, и выберите Resource). Сделайте его активным (рис.34).

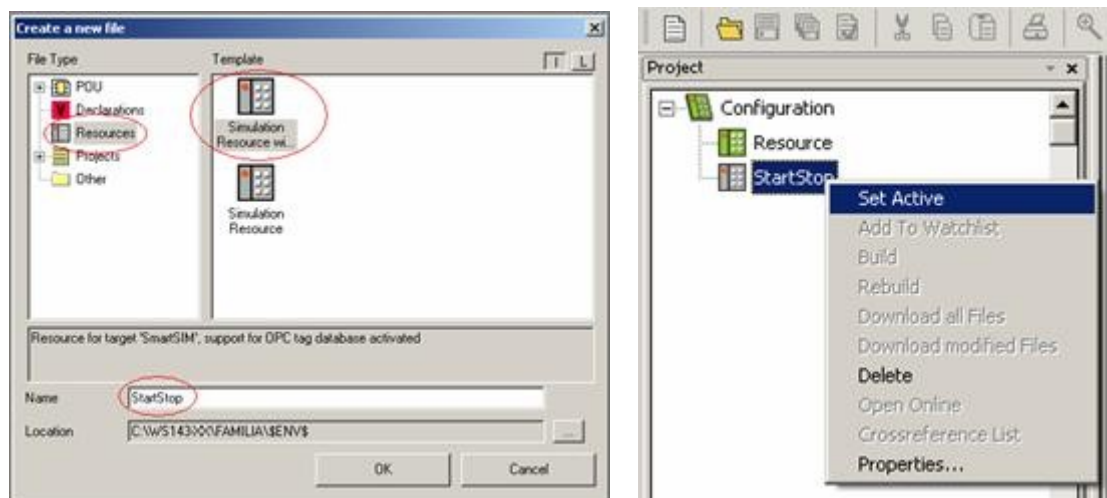


Рис. 34. Создание нового ресурса

5. Проверьте и при необходимости настройте параметры соединения. Для этого перейдите в главное меню OpenPCS “PLC->Connections...”. Активным должно быть выбрано соединение PLCXX, где XX -номер вашего лабораторного стенда, обязательно 2 цифры (например, для стенда №7 активным должно быть соединение PLC 07). Далее проверьте параметры этого соединения. Для этого нажмите кнопку Edit справа. Появится окно настроек. В строке Name должно быть PLCXX, в строке Driver – TCP (если драйвер другой, выберите TCP с помощью кнопки Select), далее нажмите Settings и проверьте номер порта (9988) и IP-адрес.

Номер порта и IP-адрес выбираются по табличным данным.

Рабочее место №	IP-адрес контроллера	Рабочее место №	IP-адрес контроллера
1	192.168.0.111	7	192.168.0.171
2	192.168.0.121	8	192.168.0.181
3	192.168.0.131	9	192.168.0.191
4	192.168.0.141	10	192.168.0.201
5	192.168.0.151	11	192.168.0.211
6	192.168.0.161	12	192.168.0.221

6. Проверьте (при необходимости настройте) Resource Properties, вашего ресурса (меню OpenPCS “ PLC-> Resource Properties...”). В качестве аппаратного модуля (Hardware Module) должен быть выбран ElsyTM, сетевое подключение (Network Connection) - настроенное п.5 подключение PLCXX (Рис.35). Остальное оставить без изменений.



Рис. 35 Настройка ресурса

УПРАЖНЕНИЕ 1

Создание ФБ и программы с физическими адресами контроллера и загрузка её в ПЛК.

1. Создайте ФБ с именем Start_Stop на языке программирования ПЛК–ST (рис.36).

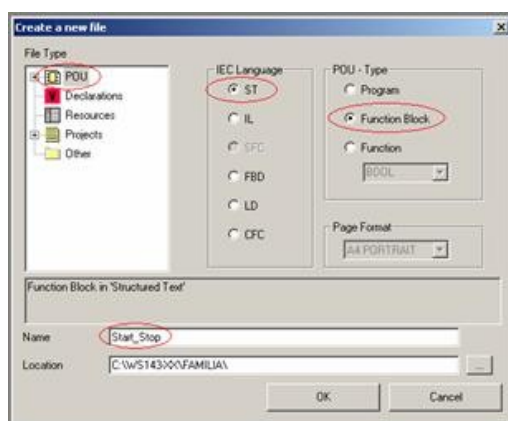


Рис. 36 Создание функционального блока

2. Опишите переменные исходя из табличных данных.

Название сигнала	Вид переменной	Тип переменной
Valve_ I n	входной	Bool
Reset	входной	Bool
Pump_ In	входной	Bool
And1	внутренний	Bool
And2	внутренний	Bool
Valve_ Control	выходной	Bool
Pump_ Control	выходной	Bool

3. Опишите функциональные блоки T1_PLC:RS и T2_PLC:RS (рис.37).

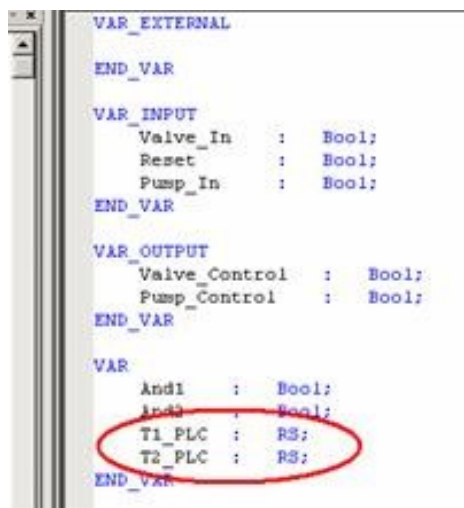


Рис.37 Описание функциональных блоков

4. Создайте код ФБ который реализует логику старт-стоп (аналогичный код был создан в предшествующей лабораторной работе).

5. Проверьте ФБ на синтаксические ошибки: при отсутствии ошибок в ресурсе ФБ появится в меню вставки ФБ. В редакторе переменных вызовите меню (нажав правой кнопкой мыши на поле) как на рисунке 38, и удостоверьтесь в наличии вашего блока.

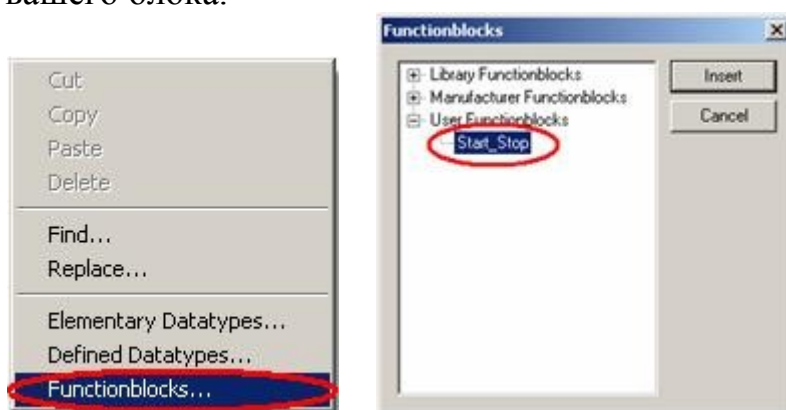


Рис.38. Проверка наличия созданного блока

6. Импортируйте 2 файла Variables. POE и Transport.ST из папки Методические указания (рис.39).

Откройте код импортированных файлов и проверьте синтаксические ошибки в них, для создания POE файлов (Program Organization Element – элемент программной организации), который необходим для создания PCD файла (Project Compiling Data – компилированные данные проекта).

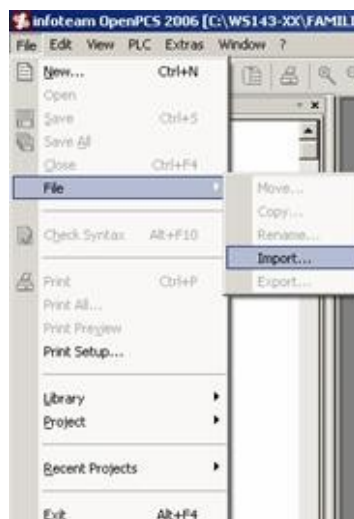


Рис.39. Импорт файлов

7. Добавьте импортированные файлы к активному ресурсу (рис.40).

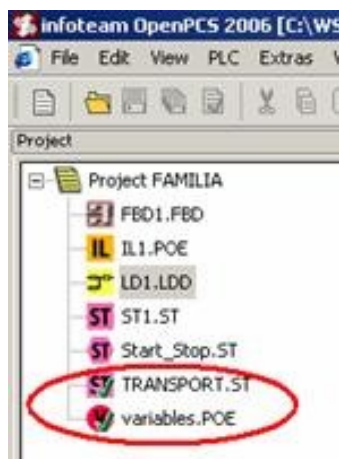


Рис. 40. Добавление файлов

8. Создайте новую программу с именем Familia1 на языке FBD. Откройте Variables.POE – здесь описаны глобальные переменные, которые переключаются в физические адреса ПЛК Elsy-TM в Transport.ST. Скопируйте переменные из Variables.POE в вашу программу.

Название сигнала	Вид переменной	Тип переменной
D_In_3_1	внешняя	Bool
D_In_3_2	внешняя	Bool
D_In_3_3	внешняя	Bool
D_Out_1_1	внешняя	Bool
D_Out_1_2	внешняя	Bool
D_Out_2_1	внешняя	Bool

D_Out_2_2	внешняя	Bool
-----------	---------	------

Опишите функциональный блок Start_stop 1: Start_stop ; во внутренние переменные. Вставьте функциональный блок Start_stop (вход D_In_3_2 необходимо проинвертировать):

Соедините входы ФБ: Valve_IN с D_In_3_3,

Reset с D_In_3_2,

Pump_In с D_In_3_1,

выхода: Valve_Control с D_Out_1_1 и D_Out_2_1,

Pump_Control с D_Out_1_2 и D_Out_2_2.

9. Проверьте программу на синтаксические ошибки. Откомпилируйте (plc-> build), свяжитесь (plc-> online) и прошейте программу в ПЛК Elsy-TM. Запустите программу в контроллере при помощи Open PCS 2006 PLC -> Coldstart или соответствующей кнопкой на панели инструментов. Признаком того, что программа запущена в контроллере служит мигающий индикатор на модуле TC505 (рис.41).



Рис.41. Вид индикатора на модуле

10. При помощи пульта расположенного на учебном лабораторном стенде (рис.42) проверьте правильность работы программы и ФБ. При правильной работе, реакция на кратковременное включение нижнего тумблера - откачка из емкости, на средний – сброс, верхний – набор в емкость.

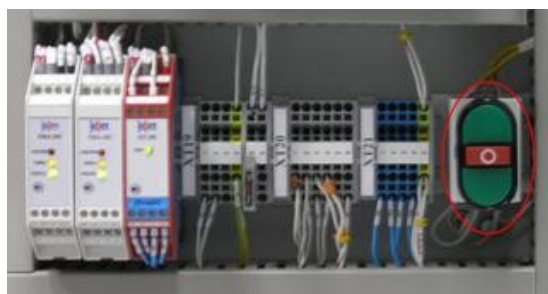


Рис.42. Вид пульта на стенде

Контрольные вопросы

1. Как создать и активизировать новый ресурс?
2. Как проверить функциональные блоки на наличие в них синтаксических ошибок?
3. Почему необходимо добавить к активному ресурсу файлы Variables.POE и Transport.ST?

Требования по содержанию отчета

В отчете студент должен перечислить цели лабораторной работы, описать ход работы, ответить на контрольные вопросы, сделать вывод о проделанной работе.

3. ОБЩИЕ СВЕДЕНИЯ О ПЛК SIMATIC

Новая система автоматизации SIMATIC объединяет отдельные частные решения системной автоматизации на основе однородной архитектуры в единое целое от аппаратуры "полевого" уровня непосредственно до управления процессом. Это достигается с помощью интегрированных в систему средств конфигурирования и программирования, с помощью управления данными в системе коммуникаций с программируемыми контроллерами (SIMATIC S7), специализированными компьютерами (SIMATIC M7) и системами управления (SIMATIC C7).

С помощью программируемых контроллеров трех выпускаемых серий перекрываются все области их применения при решении задач автоматизации процессов в целом и в производственной сфере в частности. При этом изделия серии S7-200 используются как компактные контроллеры ("микро-PLC"), изделия серий S7-300 и S7-400 используются как модульные функционально расширяемые контроллеры для применения в системах низкой и высокой производительности.

Система STEP 7, представляющая собой дальнейшее развитие STEP 5, является программным обеспечением для программирования в новой системе SIMATIC. Система Windows Microsoft была выбрана в качестве операционной среды, чтобы пользователь STEP 7 мог в полной мере использовать знакомый ему интерфейс пользователя для стандартных ПК (оконная система, работа с манипулятором "мышь").

Для программирования блоков STEP 7 предназначены языки программирования, соответствующие международному стандарту DIN EN 6.1131-3: STL ("statement list" - список мнемоник, Assembler-подобный язык), LAD ("ladder diagram" - "контактный план", представление в виде логических схем), FBD ("function block diagram" - "функциональный план", язык функциональных блок-схем) и поставляемый по отдельному заказу пакет SCL ("Structured Control Language" - "структурированный язык управления", Pascal-подобный язык высокого уровня). Кроме того по специальным заказам могут быть также поставлены дополнительные пакеты ПО, предоставляющие следующие языки программирования: S7-GRAPH (для графической разработки программ систем автоматизации SIMATIC в виде последовательности шагов и переходов между ними), S7-HiGraph (для графической разработки программ систем автоматизации SIMATIC в виде графа состояний системы и переходов между ними) и CFC ("continuous function chart" - план соединений программных блоков; при этом проектирование на CFC похоже на проектирование с FBD). Пользователю предоставляется полное право выбора из этого набора различных методов представления для описания функций при решении его задачи управления.

Программируемый контроллер SIMATIC S7-300/400 имеет модульную конструкцию. Модули, из которых составляется требуемая конфигурация контроллера, могут быть центральными (располагаться по соседству с CPU)

или распределенными. В системах SIMATIC S7 распределенные входы/выходы (I/O) являются составной частью системы. CPU, имеющий различные области памяти, составляет основу оборудования системы для обработки программ пользователя. Загрузочная память (load memory) целиком содержит пользовательскую программу: части программы, выполняемые в любое заданное время (исполняемый модуль программы), находятся в рабочей памяти (work memory), обеспечивающей малое время доступа к данным, что предопределяет высокую скорость обработки программы.

STEP 7 – это программное обеспечение для программирования S7-300/400. Для организации работы по конфигурированию, программированию и тестированию программной части системы автоматического управления процессами служит утилита SIMATIC Manager. SIMATIC Manager – это приложение, работающее под управлением Windows и содержащее все функции, необходимые для создания проекта. При необходимости SIMATIC Manager инициирует запуск других утилит, например, для конфигурирования станций, для инициализации модулей или для написания и тестирования программ.

Пользователь должен создать свое программное решение для автоматизированной системы, используя языки программирования STEP 7. Программа SIMATIC S7 является структурированной программой, что означает, что она состоит из блоков, обладающих определенными функциями, соответствующими их положению в сетевой и иерархической структуре системы. Различные классы приоритетов позволяют располагать в определенном порядке прерывания исполняемой программы пользователя.

STEP 7 работает с переменными различных типов, начиная с переменных двоичного типа (BOOL), с переменных численных форматов (INT или REAL) и заканчивая сложными типами, такими как массивы или структуры (комбинации переменных различных типов в форме единой переменной).

Базовый пакет STEP 7 (STEP 7 Basic Package) содержит следующие языки программирования: STL ("statement list" - список мнемоник), LAD ("ladder diagram" - контактный план), FBD ("function block diagram" – функциональный план). В дополнение к базовому пакету возможна поставка по специальному заказу пакетов S7-SCL ("Structured Control Language" – структурированный язык управления), S7-GGRAPH (для графической разработки программ систем автоматизации SIMATIC в виде последовательности шагов и переходов между ними), S7-HiGraph (для графической разработки программ систем автоматизации SIMATIC в виде графа состояний системы и переходов между ними).

SIMATIC Manager является главной утилитой STEP 7. При первом запуске активизируется программа "мастер проекта" (Project Wizard). Эта программа может быть использована для быстрого создания новых проектов. Тем не менее, Вы можете выключить эту программу с помощью элемента управления Check box "Display Wizard on starting the SIMATIC Manager" ("Отображать мастер-программу при запуске SIMATIC Manager"). Мастер-программа может быть вызвана при необходимости с помощью команд меню: *File (Файл) -> "New Project" Wizard*.

Процесс программирования начинается при открытии или запуске проекта ("project"). Примеры проектов представляют собой хороший материал для ознакомления.

При открытии примера проекта ZEn01_09_S7_ZEBRA с помощью команд меню: *File (Файл) -> Open (Открыть)*, Вы увидите разделенное окно проекта: слева будет структура открытого объекта (иерархическая), а справа – выбранный объект (рис.43).

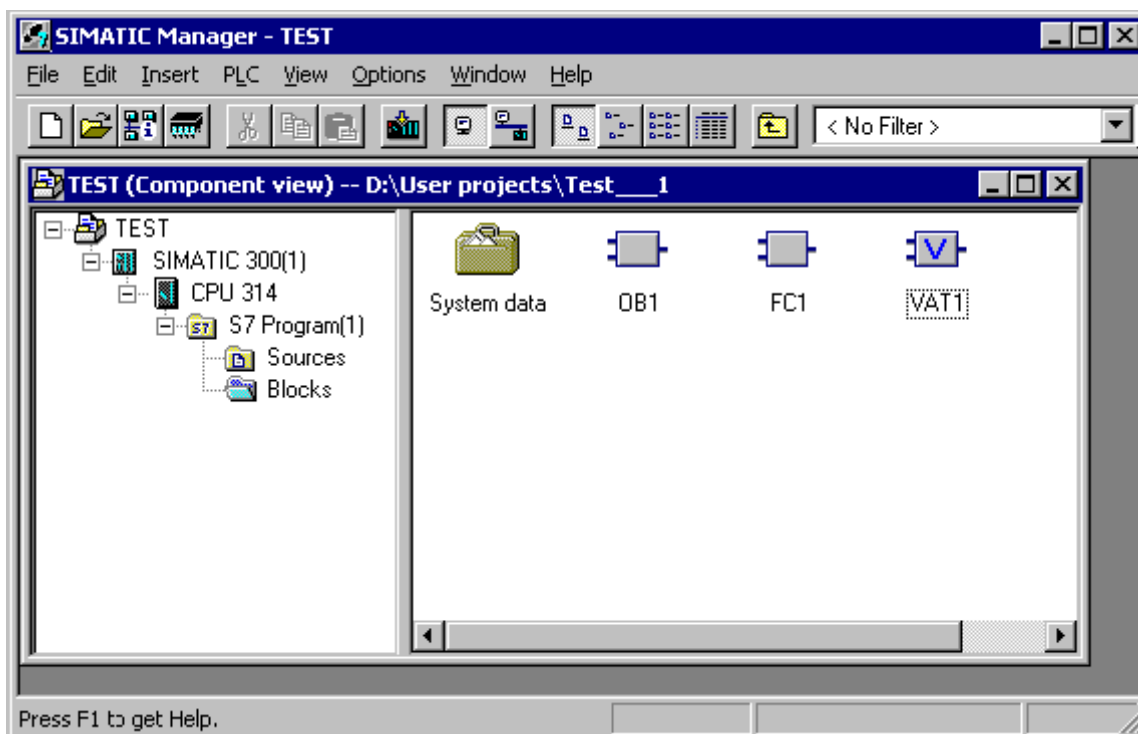


Рис.43 Пример открытого окна утилиты SIMATIC Manager

Щелчок на значке квадрата со знаком "+" позволяет открыть вложенные уровни структуры объекта; выбор объекта в левой части окна всегда вызывает отображение его содержания в правой части окна. С помощью SIMATIC Manager Вы сможете работать в среде STEP 7. "Логические" объекты, отображаемые в окнах SIMATIC Manager, соответствуют "реальным" объектам Вашей установки (процесса). Проект включает в себя установку (процесс) в целом, тогда как станция (station) соответствует программируемому контроллеру (PLC).

Проект может содержать несколько станций, связанных друг с другом, например, посредством подсети MPI. Станция содержит CPU, а CPU содержит S7-программу. В свою очередь программа включает в себя другие объекты, такие как объект *Blocks* (блоки), содержащий среди прочего скомпилированные блоки.

Объекты STEP 7 объединяются в древовидную структуру. Объекты, выделенные жирным шрифтом, содержат другие объекты. В автономном режиме (offline view) все показанные на экране объекты доступны пользователю. Эти объекты расположены на жестком диске программатора PG. Если Ваш PG находится в интерактивной связи (online) с CPU (обычная система управления с

PLC), Вы можете включить интерактивный режим (online view), выбрав опции меню: *View -> Online (Режим -> Интерактивный)*. Эта опция вызывает другое окно проекта, содержащее объекты назначенного устройства; при этом объекты, выделенные на рисунке, более не отображаются.

Вы можете видеть на панели заголовка окна активного проекта, работаете ли Вы в интерактивном (online) или в автономном (offline) режиме. Для более четкого разделения для панели заголовка и заголовка окна этих режимов могут быть установлены различные цвета. Для этого выберите опции меню: *Options -> Customize (Опции -> Установки пользователя)* и измените соответствующие параметры на вкладке "View" ("Режим"). Выбрав опции меню: *Options -> Customize (Опции -> Установки пользователя)*, можно изменить базовые установки SIMATIC Manager, такие как session language (язык), архив программы и место расположения для проектов, библиотек и конфигурирование архива программы.

В STEP 7 "главные объекты", находящиеся на верхнем уровне структурной иерархии, это проекты (project) и библиотеки (library). *Проекты (projects)* используются для систематического хранения данных и программ для решения задачи автоматизации. Важнейшие из них:

- данные конфигурации оборудования;
- параметры для модулей;
- данные конфигурации сетевых коммуникаций;
- программы (коды и данные, символы, исходные программы).

Объекты в проекте организованы в виде иерархической системы. Первым шагом для редактирования всех объектов проекта является открытие проекта. В следующих разделах обсуждается процесс редактирования этих объектов.

Библиотеки (library) используются для хранения многократно используемых компонентов программы. Библиотеки организованы в виде иерархической системы. Они могут содержать STEP 7 программы, которые в свою очередь могут содержать программы пользователя (скомпилированные блоки), исходные тексты программ и таблицы символов. За исключением возможности интерактивной (online) связи (не возможна отладка программы), создание программ или частей программ в библиотеке обеспечивает такие же функциональные возможности как и у объекта.

4 ЗАПУСК SIMATIC MANAGER И СОЗДАНИЕ ПРОЕКТА

SIMATIC Manager [Администратор SIMATIC] – это центральное окно, которое становится активным при запуске STEP 7. По умолчанию запускается мастер STEP 7 (STEP 7 Wizard), который оказывает вам помощь при создании проекта STEP 7. Структура проекта используется для надлежащего хранения и размещения всех данных и программ.

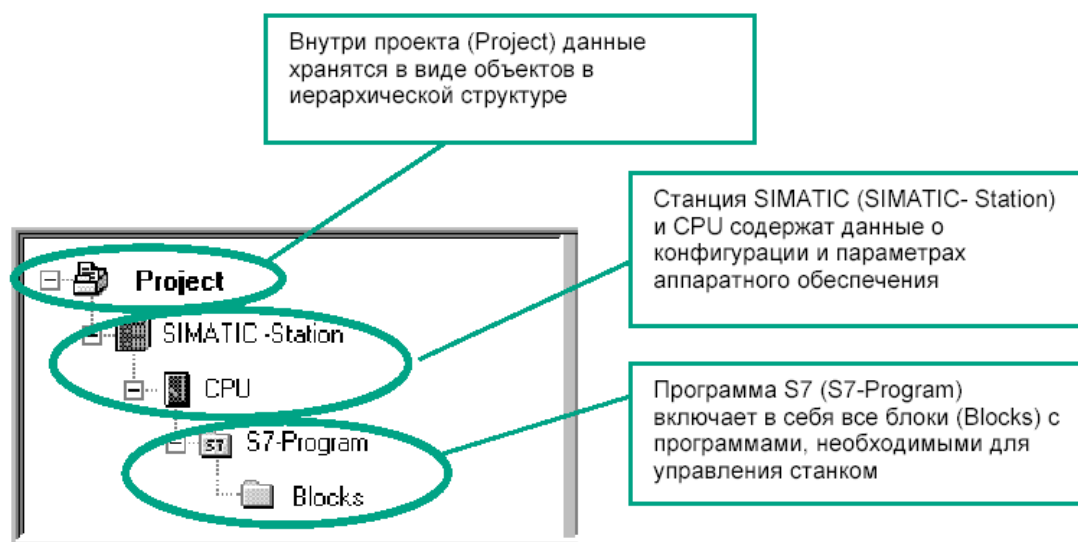


Рис.44 Создание проекта в SIMATIC Manager

Дважды щелкните на пиктограмме **SIMATIC Manager**. Активизируется мастер STEP 7 (STEP 7 Wizard). В **предварительном обзоре (preview)** вы можете включать и выключать отображение структуры создаваемого проекта. Чтобы перейти к следующему диалоговому окну, щелкните на кнопке **Next** (рис.45).

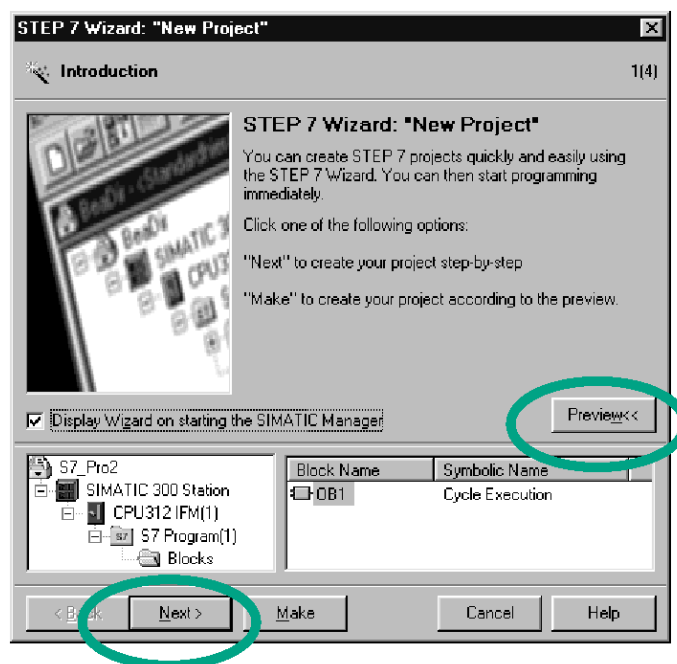


Рис.45

Для примера реализации проекта выберите CPU 314. Установка по умолчанию для адреса MPI равна 2.

Щелкните на **Next (Дальше)**, чтобы подтвердить настройки и перейти к следующему диалоговому окну. Каждый CPU обладает определенными свой-

ствами; например, относительно конфигурации его памяти или адресных областей. Поэтому пользователь должен выбрать CPU, прежде чем начать программирование. Адрес MPI (многоточечный интерфейс) нужен, чтобы CPU мог обмениваться информацией с устройством программирования или PC (рис.46).

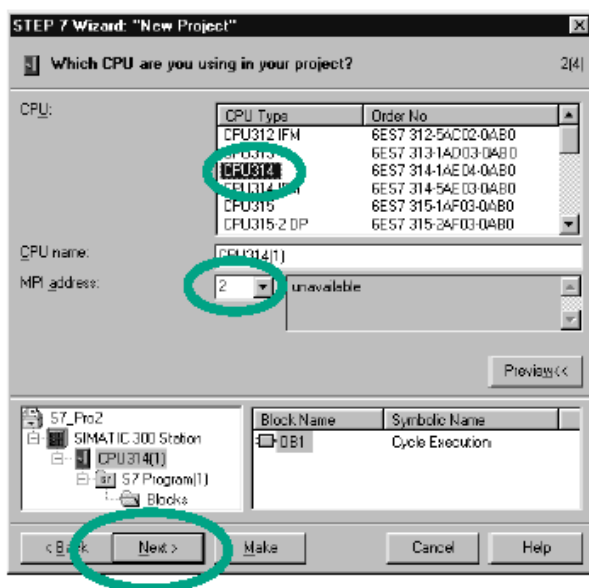


Рис.46

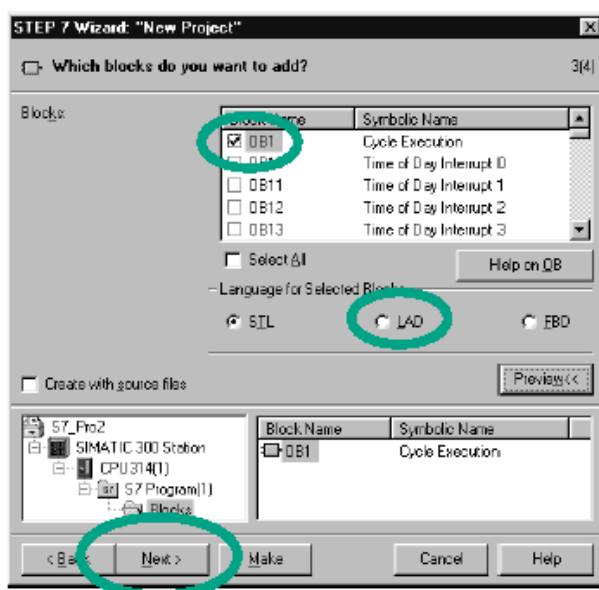


Рис.47

Выберите организационный блок **OB1** (если он еще не выбран), рисунок 47. Выберите один из языков программирования: контактный план (**LAD**), список операторов (**STL**) или функциональный план (**FBD**). Далее подтвердите настройки кнопкой **Next** [Дальше]. OB1 представляет самый высокий уровень программирования и организует другие блоки в программе S7. Позднее можно выбрать другой язык программирования. После нажатия кнопки **Make** [Co-

здать], SIMATIC Manager откроет окно для проекта "Getting Started", который вы создали.

Далее приведено для каких целей нужны созданные файлы и папки и как с ними можно эффективно работать. Мастер STEP 7 активизируется каждый раз, когда запускается эта программа. Вы можете деактивировать эту установку по умолчанию в первом диалоговом окне для мастера (Wizard). Однако если проект создается без мастера STEP 7, то необходимо реализовать каждый каталог внутри проекта самостоятельно. Дополнительную информацию можно найти, используя команду меню [**Help > Contents** [**Помощь > Содержание**] в разделе "Setting Up and Editing the Project [Создание и редактирование проекта]".

Начиная с версии STEP 7 V3.2 программа STEP 7 Wizard помогает пользователю при создании новых проектов. Пользователь должен задать тип используемого CPU, и программа-мастер создаст проект с S7-станцией и выбранным CPU, а также каталог для S7-программы, каталог для исходных программ и каталог блоков с выбранными организационными блоками.

Создание проекта с S7-станцией

При необходимости студент может создать новый проект "вручную". Для создания нового проекта выберите опции меню: *File -> New (Файл -> Создать)*, введите имя в диалоговом окне, измените тип и место расположения, если это необходимо, и подтвердите выбор щелчком на кнопке "ОК" или нажатием клавиши "Enter".

Для создания новой станции в проекте выберите проект и вставьте станцию с помощью опций меню: *Insert -> Station -> Simatic 300 Station (Вставка -> Станция -> Станция S7-300)* (в данном случае станция S7-300).

Конфигурирование станции производится следующим образом. Щелкните на прямоугольнике со значком плюса, следующем за объектом *project* в левой части окна проекта и выберите станцию; SIMATIC Manager отображает объект Hardware (оборудование) в правой части окна. Двойным щелчком по *Hardware* запускается утилита конфигурирования оборудования Hardware Configuration, с помощью которой осуществляется редактирование таблиц конфигурации.

Если каталог модулей не показан на экране, то вызовите его с помощью опций меню: *View -> Catalog (Вид -> Каталог)*. Конфигурирование начинается с выбора несущей шины (rail), например, в "SIMATIC 300" и "RACK 300" и переносом методом "drag-n-drop" посредством мыши на свободное место в верхней половине окна станции (station window). При этом студент может наблюдать таблицу, в которой показаны слоты на шине. На следующем этапе необходимо выбрать требуемые модули из каталога модулей и, используя процедуру "drag-n-drop", перенести эти модули в соответствующие слоты. Для дальнейшего редактирования структуры проекта требуется установить по крайней мере один CPU, например, CPU 314 в слот 2. Остальные необходимые модули можно добавлять позже.

Далее необходимо сохранить и скомпилировать станцию, после чего закрыть ее и вернуться в SIMATIC Manager. Кроме конфигурации оборудования открытая станция показывает также CPU. При конфигурировании CPU утилита SIMATIC Manager также создает S7-программу со всеми объектами. Создание структуры проекта при этом завершается.

Для просмотра содержания S7-программы необходимо открыть CPU; в правой части окна проекта. Здесь можно видеть символы для S7-программы (*S7-program*) и для таблицы соединений (*connection table*). Откройте *S7-program* – SIMATIC Manager отображает символы для скомпилированной программы пользователя (*Blocks - Блоки*), каталог для исходных программ и таблицу символов в правой части окна.

Откройте программу пользователя (*Blocks - Блоки*) – SIMATIC Manager отображает символы для скомпилированных данных конфигурации (*System data - Системные данные*) и пустой организационный блок для основной (*main*) программы (OB1) в правой части окна.

Далее приступим к редактированию объектов программы пользователя. На этом этапе достигнут нижний уровень иерархической структуры объектов. При первом открытии OB 1 отображается окно свойств объекта и запускается редактор для редактирования организационного блока. Студент может добавлять другие пустые блоки для инкрементного редактирования посредством выбора пунктов: **Insert -> S7 Block -> .** (*Blocks* должно быть выделено) и выбором требуемого типа из представленного списка.

Создание проекта без S7-станции

При необходимости можно создать программу без предварительного конфигурирования станции. Для этого нужно самостоятельно создать каталог для программы, т.е. выбрать проект и сгенерировать S7-программу, используя опции меню: **Insert -> Program -> S7 Program** (*Вставить -> Программу -> S7-программу*). В данной S7-программе SIMATIC Manager создает объект **Symbols** (*Символы*) и каталоги объектов **Sources** (*Исходные файлы*) и **Blocks** (*Блоки*). Каталог **Blocks** (*Блоки*) содержит пустой блок OB 1.

Также студент может создать программу в объекте *library* (*библиотека*), если необходимо использовать ее больше, чем один раз. При этом такая стандартная программа будет всегда доступна, и студент может ее копировать полностью или по частям в свою текущую программу. Необходимо помнить, что у пользователя нет возможности интерактивной (*online*) связи с библиотекой, и поэтому отладить S7-программу можно только в составе проекта.

Создание S7-программ

Программа пользователя создается в каталоге (в объекте) *S7 Program*. Вы можете назначать этот объект в объекте CPU в структурной иерархии проекта, или вне зависимости от CPU. В свою очередь объект *S7 Program* включает в себя объект *Symbols* (*Символы*) и каталоги *Source Files* (*Исходные файлы*) и *Blocks* (*Блоки*) (см. рис. 48).

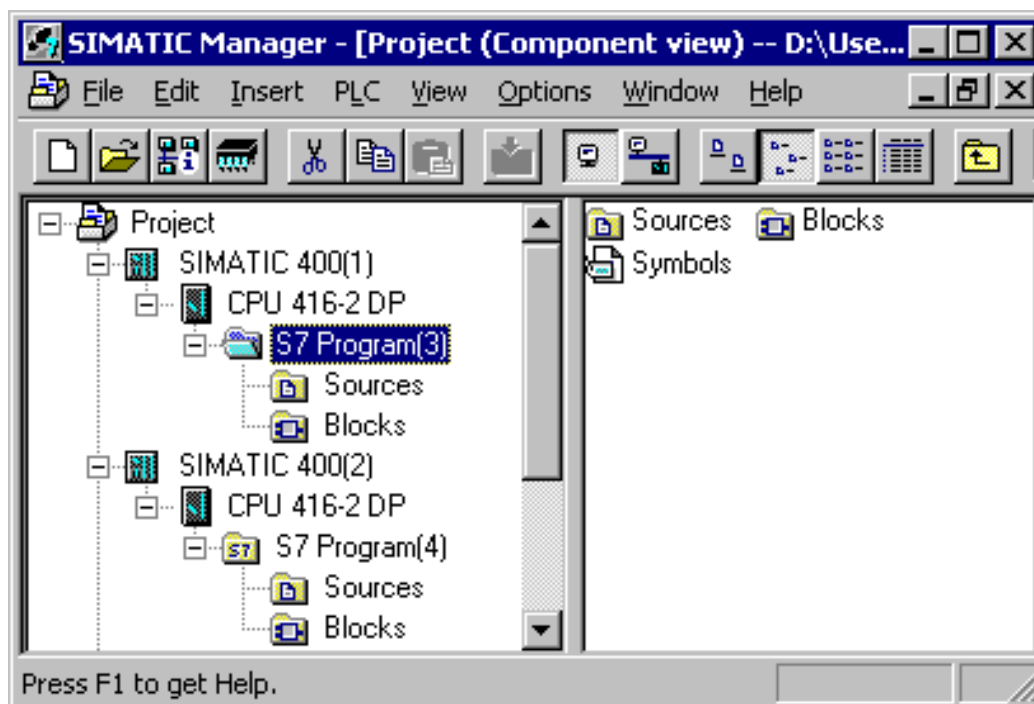


Рис. 48 Объекты, участвующие при генерации программы

В случае создания программы путем написания исходных файлов ("**source-oriented**") Вы должны создать одну или несколько исходных программ и сохранить их в виде файлов в каталоге *Source Files (Исходные файлы)*. Исходные программы - это текстовые файлы формата ASCII, которые содержат операторы программы для одного или нескольких блоков, возможно даже целиком всю программу. Необходимо скомпилировать исходные программы; далее скомпилированные блоки программы помещаются в каталог *Blocks (Блоки)*. Скомпилированные блоки содержат код MC7 и выполняются в S7 CPU.

В случае создания программы "инкрементным" путем ("**incremental**"), - методом добавления, требуется вводить программу блок за блоком. Вводимые блоки немедленно проверяются на наличие синтаксических ошибок. При поступлении команды на сохранение блок сначала компилируется, затем сохраняется в каталоге *Blocks (Блоки)*. При создании программы данным методом можно также редактировать блоки в интерактивном (online) режиме в CPU, даже во время рабочего режима.

В программе обрабатываются значения сигналов или значения адресов. Адрес - это, например, вход I1.0 (абсолютная адресация). С помощью таблицы символов **Symbol Table** в объекте Symbols, можно назначить адресу символьное имя, например, "Switch motor on" ("Включение мотора") и после этого обращаться к этому адресу, используя данное символьное имя (символьная адресация). В свойствах автономного объекта *Blocks (Блоки)* можно определить, каким способом будут адресоваться переменные в таблице символов (Symbol Table) после корректировки - абсолютным или символьным в уже скомпилированных блоках, согласно приоритету адресации (*address priority*).

Таблица символов (Symbol Table)

В управляющей программе работа производится с адресами, т.е. с входами, с выходами, таймерами и блоками. Студент может назначить абсолютные адреса (например, I1.0) или символьные адреса (например, Start signal [сигнал запуска]). При символьной адресации используются символьные имена. Это делает программу легко читаемой, благодаря тому, что символьные имена несут смысловую нагрузку. При использовании символьной адресации различаются локальные (*local*) и глобальные (*global*) символы (символьные имена). Локальный (*local*) символ распознается только в блоке, в котором они определены. Поэтому при необходимости можно использовать одинаковые локальные символьные имена в различных целях в разных блоках. Глобальный символ распознается в любом месте программы и имеет одинаковое значение во всех блоках программы. Студент должен определить глобальный символ в таблице символов (объект *Symbols* в каталоге *S7 Program*). Глобальный символ начинается с символа алфавита и может иметь в длину до 24 символов. Глобальный символ может также содержать пробелы, специальные символы и национальные символы.

Исключения составляют символы 00 hex, FF hex и кавычки ("). При программировании необходимо заключать спецсимволы в кавычки. В скомпилированном блоке программный редактор отображает все глобальные символы в кавычках. Комментарий к символу может составлять в свою очередь запись из 80 символов. В таблице символов можно назначать имена следующим адресам и объектам:

- Входам I, выходам Q, периферийным входам PI и выходам PQ;
- Маркерам M, таймерам T и счетчикам C;
- Блокам кодов OB, FB, FC, SFC, SFB и блокам данных DB;
- Типам данных, определенным пользователем, UDT;
- Таблице переменных VAT.

Адреса данных в блоках данных находятся среди локальных адресов, связанные символы определяются в разделе описаний (*declaration section*) блоков данных в случае глобальных блоков данных и в разделе описаний (*declaration section*) функциональных блоков в случае экземплярных блоков данных.

При создании S7-программ SIMATIC Manager создает также пустую таблицу символов *Symbols*. Студент может открыть эту таблицу и определить глобальные символы и назначить их абсолютным адресам (рис. 49).

The screenshot shows the 'Symbol Editor' window for a SIMATIC 300 project. It contains a table with the following data:

	Symbol	Address	Data type	Comment
1	stop	I 0.0	BOOL	Остановка конвейера
2	start	I 0.1	BOOL	Запуск конвейера
3	PartNO	MW 10	INT	Номер партии
4				
5				

Рис. 49 Пример таблицы символов Symbol Table

В S7-программе может быть только одна таблица символов *Symbols*. Тип данных является частью определения символа. Он определяет особые свойства данных, в частности представление содержимого данных. Например, тип данных **BOOL** идентифицирует двоичную переменную, а тип данных **INT** обозначает переменную в цифровой форме, содержание которой определяется 16-битным целым числом.

В случае "инкрементного" программирования создается таблица символов до ввода программы; здесь можно также добавить или скорректировать отдельные символы во время ввода программы. При создании программы путем, ориентированным на создание исходных текстов программы готовая таблица символов должна быть доступна к моменту компиляции программы.

Для выполнения практических упражнений по STEP 7 в этом руководстве потребуется следующее:

- □ Устройство программирования фирмы Siemens или PC;
- □ Пакет программного обеспечения STEP 7 и авторизационная дискета;
- □ Программируемый контроллер SIMATIC S7-300 или S7-400.

Дополнительная документация по STEP 7:

- Базовая информация по STEP 7;
- □ Справочная информация по STEP 7.

Объединение аппаратного и программного обеспечения

С помощью программного обеспечения STEP 7 вы можете создать свою программу S7 внутри проекта. Программируемый контроллер S7 состоит из источника питания, CPU и модулей ввода и вывода. Программируемый логический контроллер (ПЛК) контролирует установку и управляет ею с помощью программы S7. К модулям ввода/ вывода в программе S7 обращаются через адреса (рис.50).

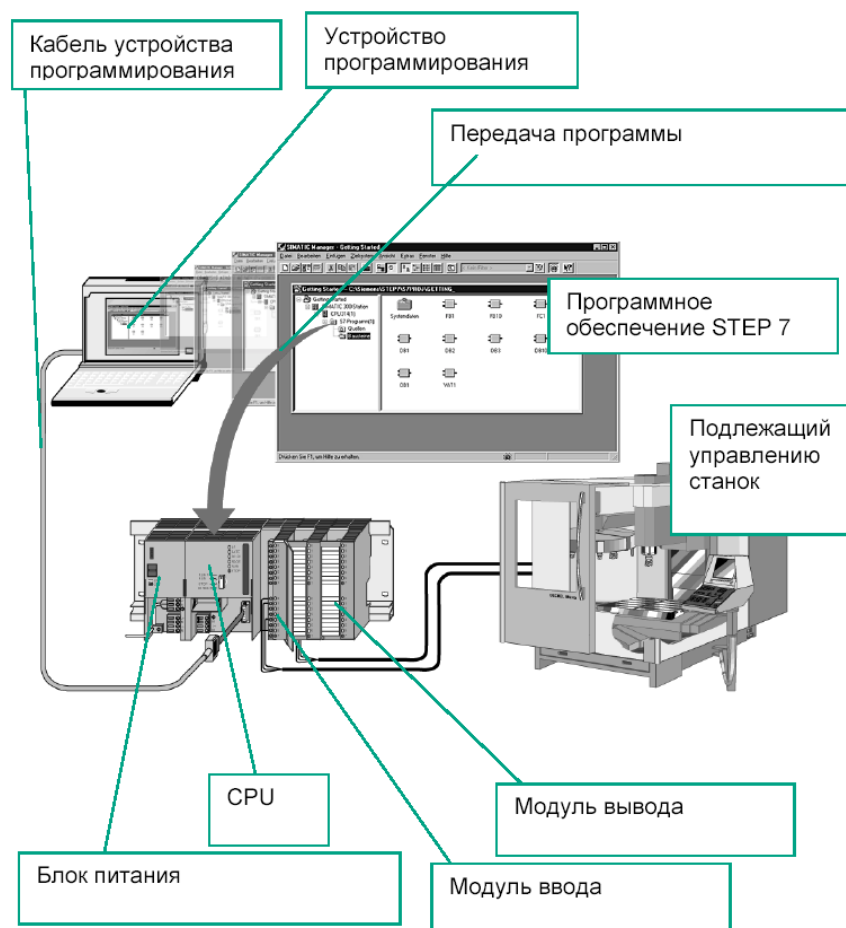


Рис. 50 Аппаратное и программное обеспечение

Основная последовательность действий при использовании STEP 7. Если пользователь создает большие программы со многими входами и выходами, то рекомендуется сначала сконфигурировать аппаратные средства. Преимущество этого состоит в том, что STEP 7 отображает возможные адреса в редакторе конфигурирования аппаратуры.

Если был выбран второй вариант, то нужно определить каждый адрес самостоятельно, в зависимости от выбранных пользователем компонентов, и соответственно нельзя будет вызывать эти адреса через STEP 7 (рис.51).

При конфигурировании аппаратуры вы не только можете определять адреса, но и можете также изменять параметры и свойства модулей. Например, если вы хотите работать с несколькими CPU, то вы должны согласовывать адреса MPI этих CPU.

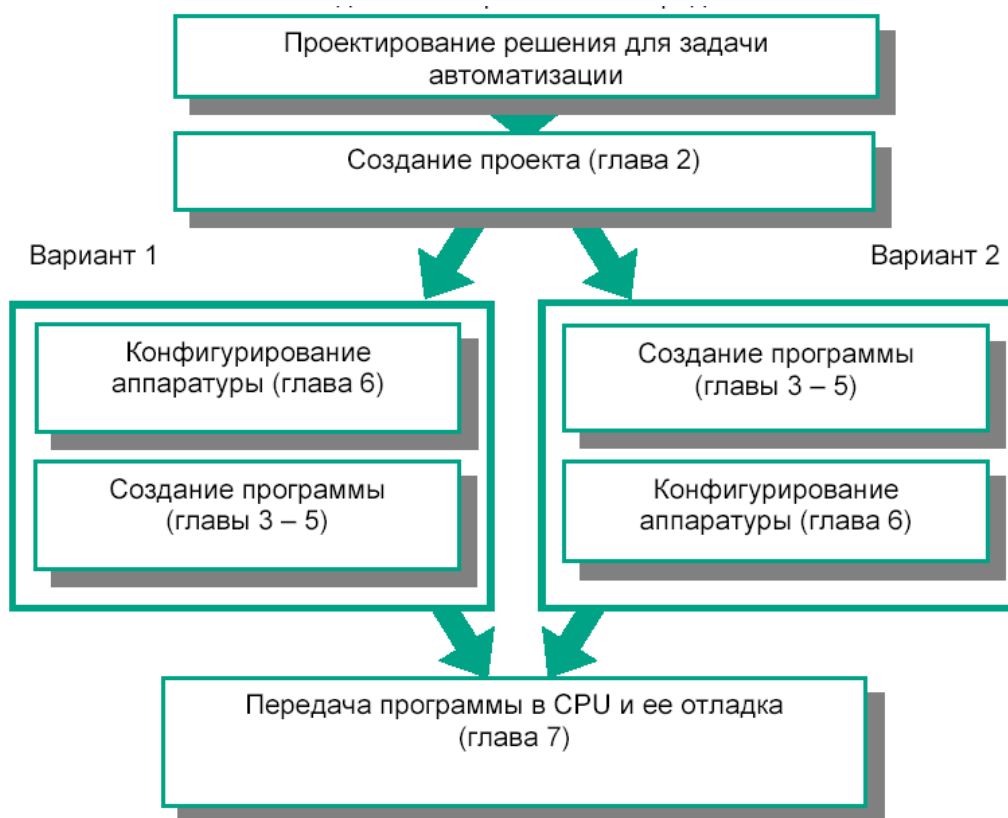


Рис. 51 Последовательность действий при использовании STEP 7

Структура проекта в SIMATIC Manager. Оперативная справка

После закрытия мастер STEP 7 появляется SIMATIC Manager с открытым окном проекта "Getting Started". Отсюда можно запустить все функции и окна STEP 7.

Проект отображается с выбранной станцией S7 и CPU. Щелкните на знаке + или –, чтобы открыть или закрыть папку (рис.52). Позднее можно запускать другие функции, щелкая на символах, отображаемых на правой панели. Щелкните на папке **Program (1)**. Она содержит все необходимые компоненты программы. Компонент Source Files [Исходные файлы] используется для хранения программ в виде исходных файлов. Щелкните на папке **Blocks** [Блоки]. Она содержит **OB 1**, который уже создан, а позднее и все другие блоки. Отсюда можно запускать программирование в контактном плане, списке операторов или функциональном плане. Щелкните на папке **SIMATIC 300 Station**. Здесь хранятся все данные проекта, относящиеся к аппаратуре. В последнем можно использовать компонент Hardware [Аппаратура] для указания параметров программируемого контроллера.

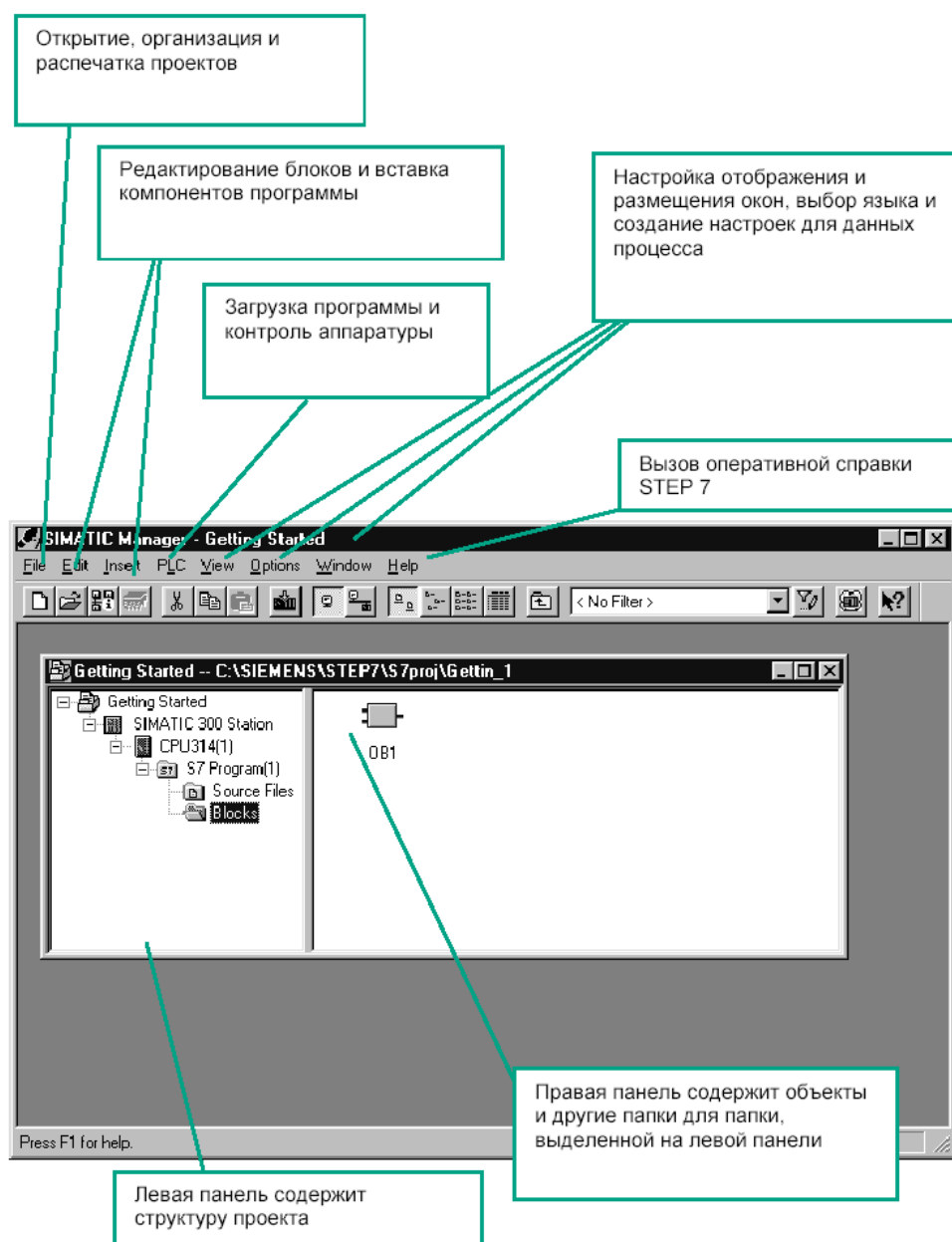


Рис. 52 Окно SIMATIC Manager

Если для решения задачи автоматизации нужно другое программное обеспечение SIMATIC, например, дополнительные пакеты PLCSIM (программа имитации аппаратных средств) или S7 Graph (графический язык программирования), то они тоже встраиваются в STEP 7. С помощью SIMATIC Manager можно, например, непосредственно открывать соответствующие объекты, такие как функциональный блок S7 Graph.

Дополнительную информацию можно найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Working Out the Automation Concept [Разработка концепции автоматизации]" и "Basics of Designing the Program Structure [Основы проектирования структуры программы]". Информацию о дополнительных пакетах студент может найти в каталоге ST 70 "Com-

ponents for Completely Integrated Automation [Компоненты для полностью встроенной автоматизации]".

5. ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ LAD, STL И FBD

Каждый вход и выход имеет абсолютный адрес, predetermined конфигурацией аппаратуры. Этот адрес указывается непосредственно, например: I 1.5 (вход-байт 1-бит 5). Абсолютный адрес может быть заменен символическим именем по вашему выбору (рис.53).

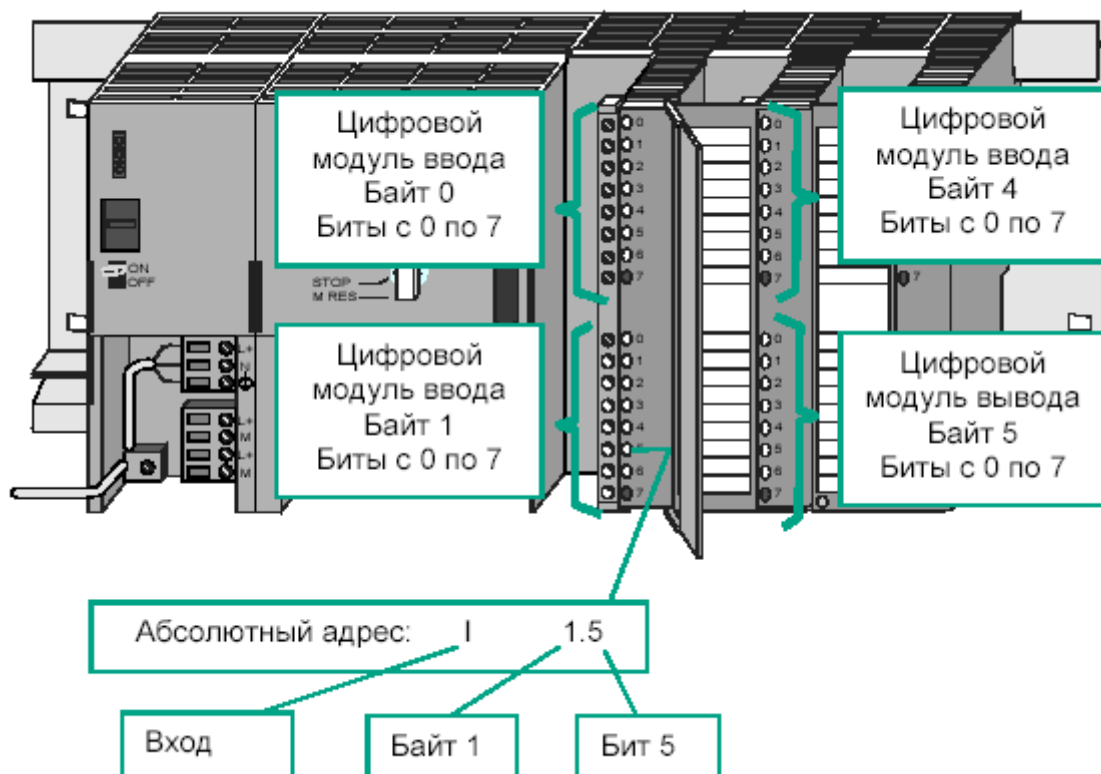


Рис. 53 Распределение адресов

Лучше использовать только абсолютное программирование, если в программе S7 не нужно обращаться ко многим входам и выходам.

Создание программы в OB1

В STEP 7 программы S7 создаются на стандартных языках программирования: контактный план (LAD), список операторов (STL) или функциональный план (FBD), рисунок 54. На практике необходимо решить, какой язык использовать.

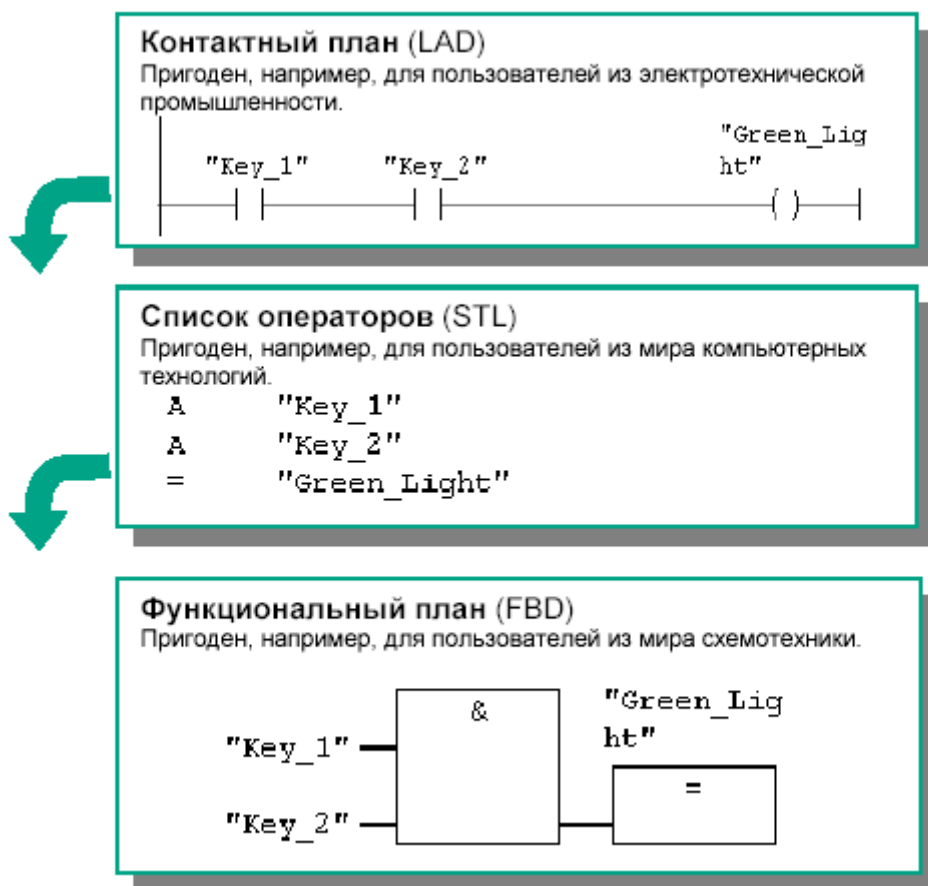


Рис. 54 Стандартные языки программирования

Блок OB1 теперь откроется в соответствии с языком, который вы выбрали при создании блока в мастере проекта. Однако вы можете в любое время изменить язык программирования, установленный по умолчанию.

Копирование таблицы символов и открытие OB 1

Для копирования таблицы символов необходимо открыть свой проект "Getting Started". Для этого щелкните на кнопке **Open [Открыть]** на панели инструментов, выберите проект "Getting Started", который был создан, и подтвердите с помощью **OK**.

В зависимости от того, какой язык программирования решено использовать, откройте один из следующих проектов:

- □zEn01_06_STEP7__**LAD**_1-9;
- □zEn01_02_STEP7__**STL**_1-9;
- □zEn01_04_STEP7__**FBD**_1-9.

В методических указаниях приведены все три примера проектов. Необходимо открыть „zEn01_XXX“ и достигнуть компонент **Symbols [Символы]**. Далее студент может скопировать его с помощью буксировки в папку **S7 Program** в окне своего проекта "Getting Started". Затем окно „zEn01_XXX“ закрывается. Дважды щелкнув на **OB 1** в проекте "Getting Started" и откроется окно для программирования LAD/STL/FBD.

Буксировка означает, что вы щелкаете мышью на любом объекте и перемещаете его, удерживая кнопку мыши нажатой. Когда вы отпускаете кнопку мыши, объект вставляется в выбранной позиции. В STEP 7 OB 1 обрабатывается CPU циклически. CPU читает и исполняет строку за строкой команды программы. Когда CPU возвращается к первой строке программы, он завершает ровно один цикл. Время, необходимое для этого, называется временем цикла сканирования. Все блоки программируются в окне LAD/STL/FBD.

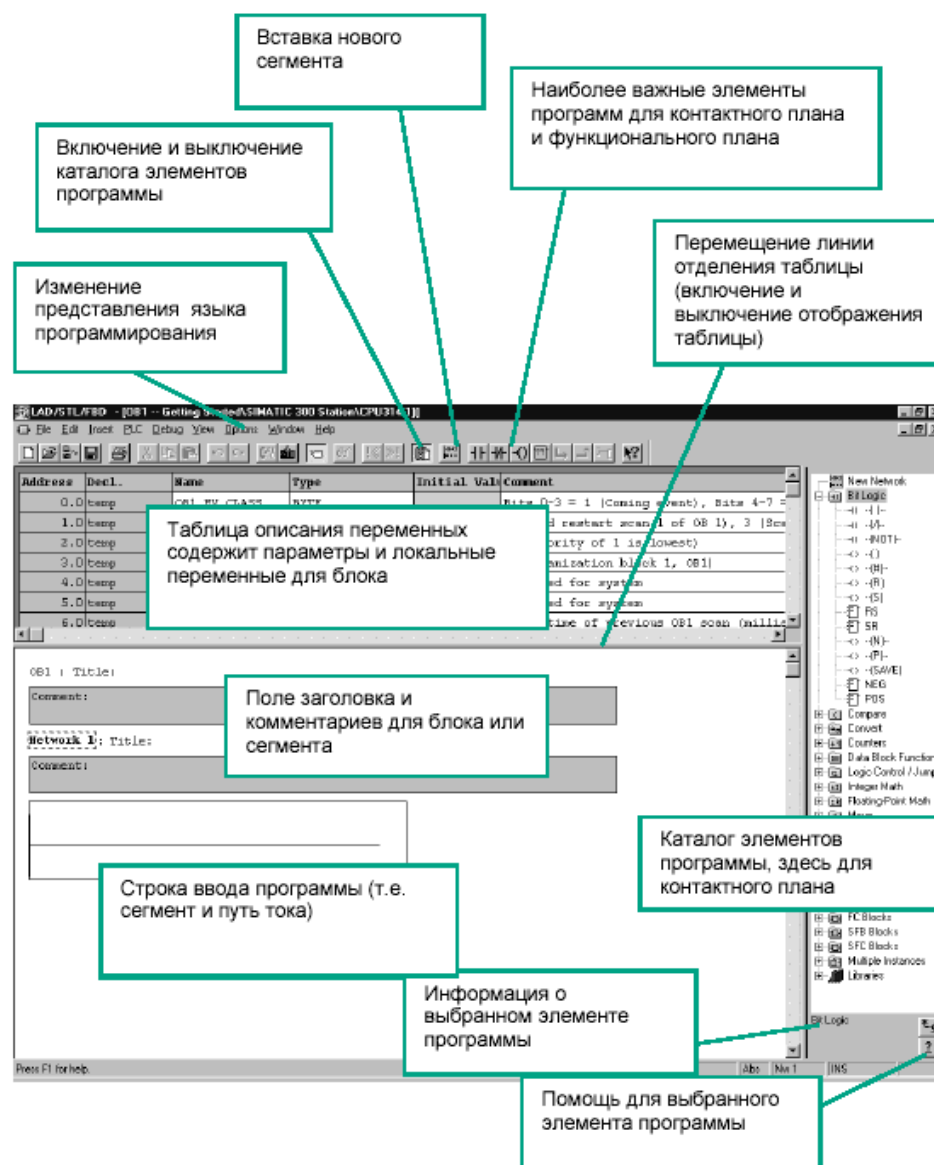


Рис. 55 Окно для программирования LAD/STL/FBD

Программирование OB 1 в виде контактного плана

Далее рассмотрим примеры программирования последовательной, параллельной цепи и функцию памяти SR (установка / сброс) в виде контактного плана (LAD).

Программирование последовательной цепи в контактном плане. Если необходимо, установите **LAD** в качестве языка программирования в меню **View [Вид]**. Щелкните в области **заголовка (title)** OB1 и введите, например, "Циклически обрабатываемая главная программа". Выберите путь тока для своего первого элемента. Щелкните на этой кнопке на панели инструментов и вставьте нормально открытый контакт (рис.56).

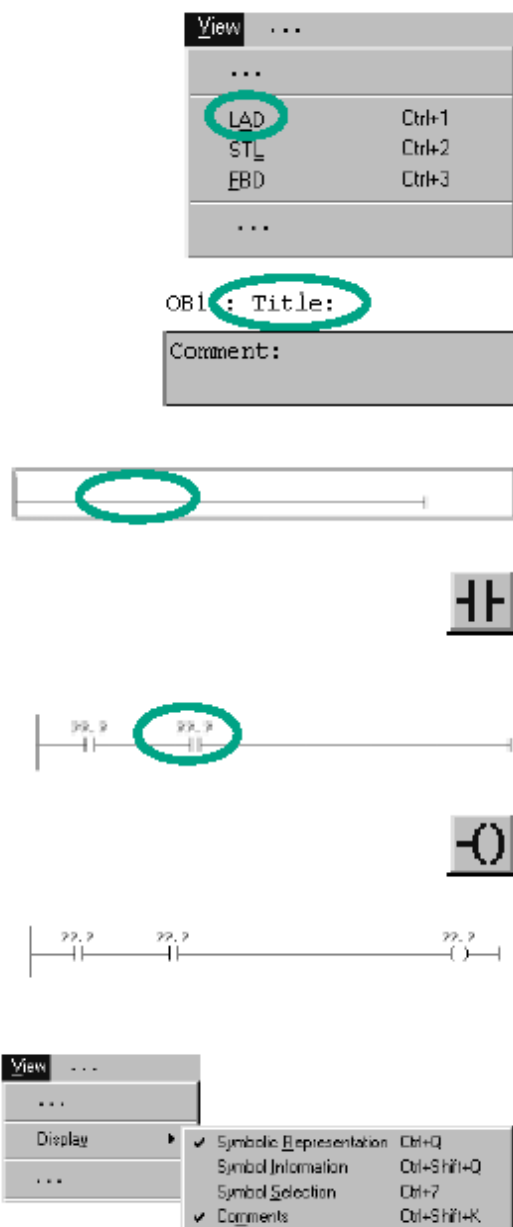


Рис. 56 Последовательная цепь в контактном плане

Таким же образом вставьте второй нормально открытый контакт. Вставьте катушку у правого конца пути тока. В этой последовательной цепи пока отсутствуют адреса нормально открытых контактов и катушки. Проверьте, активизировано ли символическое представление (Symbolic Representation).

Щелкните на знаке **???** и введите символическое имя "Key_1 [Ключ_1]" (в кавычках). Подтвердите, нажав **Enter**. Введите символическое имя "Key_2 [Ключ_2]" для второго нормально открытого контакта. Введите имя

"Green_Light [Зеленый_свет]" для катушки. Теперь запрограммирована вся последовательная цепь. Сохраните блок, если отсутствуют символы, выделенные красным цветом. Символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка (рис.57).

Вы можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке **??.**, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически.

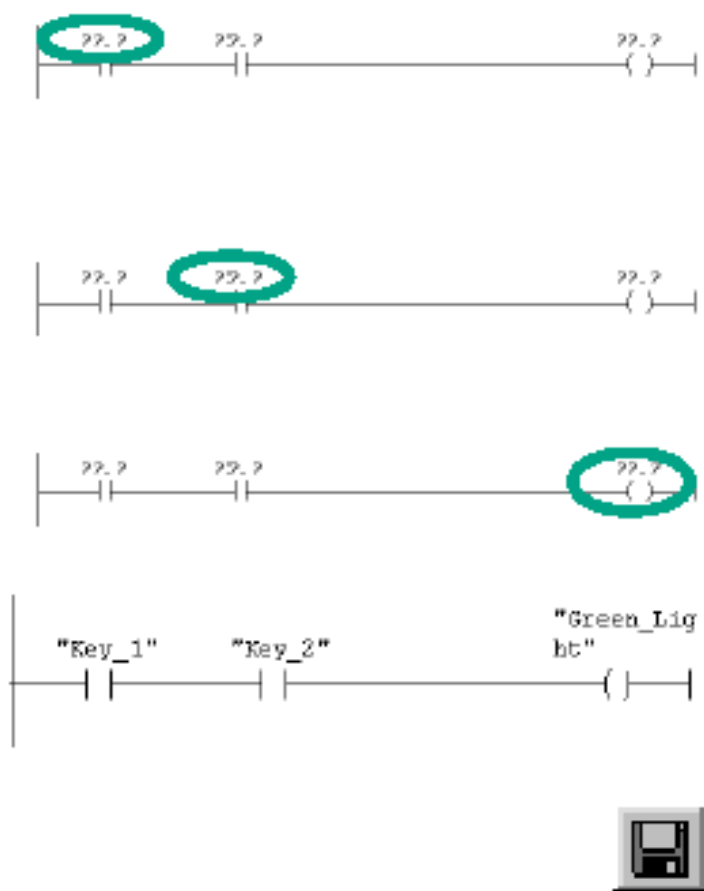


Рис. 57

Для программирования параллельной цепи в контактном плане выделите **Network 1 [Сегмент 1]**. Вставьте новый сегмент. Снова выберите путь тока. Вставьте нормально открытый контакт и катушку. Выделите вертикальную линию в пути тока. Вставьте параллельную ветвь. Добавьте еще один нормально открытый контакт в параллельной ветви. Закройте ветвь (если необходимо, выберите нижнюю стрелку). В параллельной цепи все еще отсутствуют адреса. Для назначения символических адресов действуйте так же, как и для последовательной цепи.

Напишите у верхнего нормально открытого контакта "Key_3 [Ключ_3]", у нижнего контакта "Key_4 [Ключ_4]", а у катушки "Red_Light [Красный_свет]" Сохраните блок.

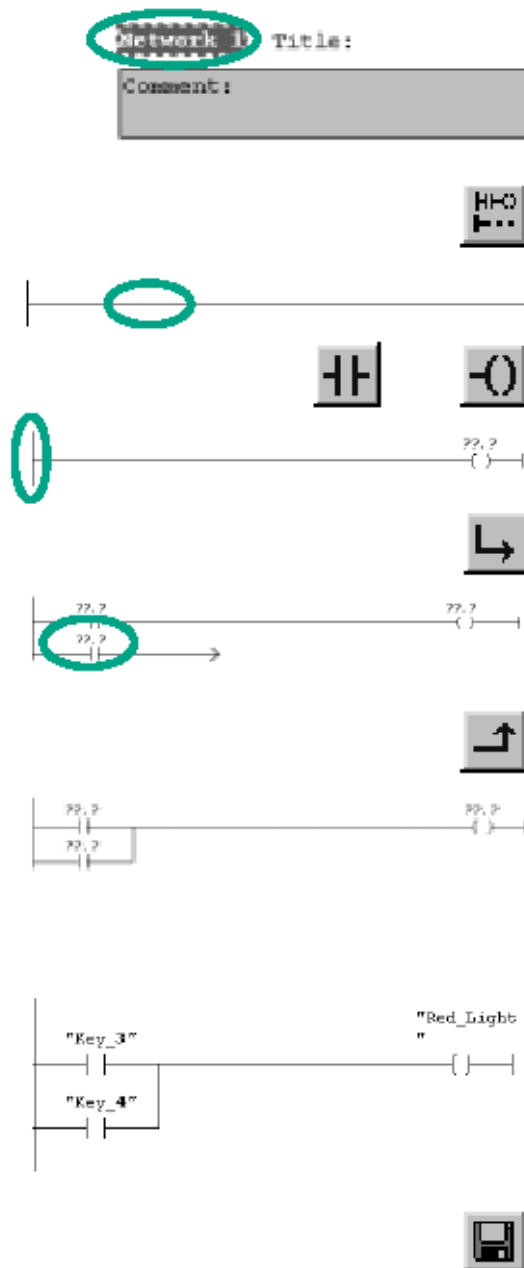


Рис. 58 Параллельная цепь в контактном плане

Для программирования функции памяти в контактном плане выделите Network 2 [Сегмент 2] и вставьте еще один сегмент (рис.59). Снова выделите путь тока. Перемещайтесь в каталоге элементов программы в разделе **Bit Logic [Двоичная логика]**, пока не достигнете элемента **SR**. Дважды щелкните, чтобы вставить этот элемент. Вставьте нормально открытый контакт перед каждым из входов S и R. Введите следующие символические имена перед элементом SR:

- Верхний контакт "Automatic_On
- [Автоматический_режим_включен]"
- Нижний контакт "Manual_On
- [Ручной_режим_включен]"

- Элемент SR "Automatic_Mode
 - [Автоматический_режим]".
- Сохраните блок и закройте окно.

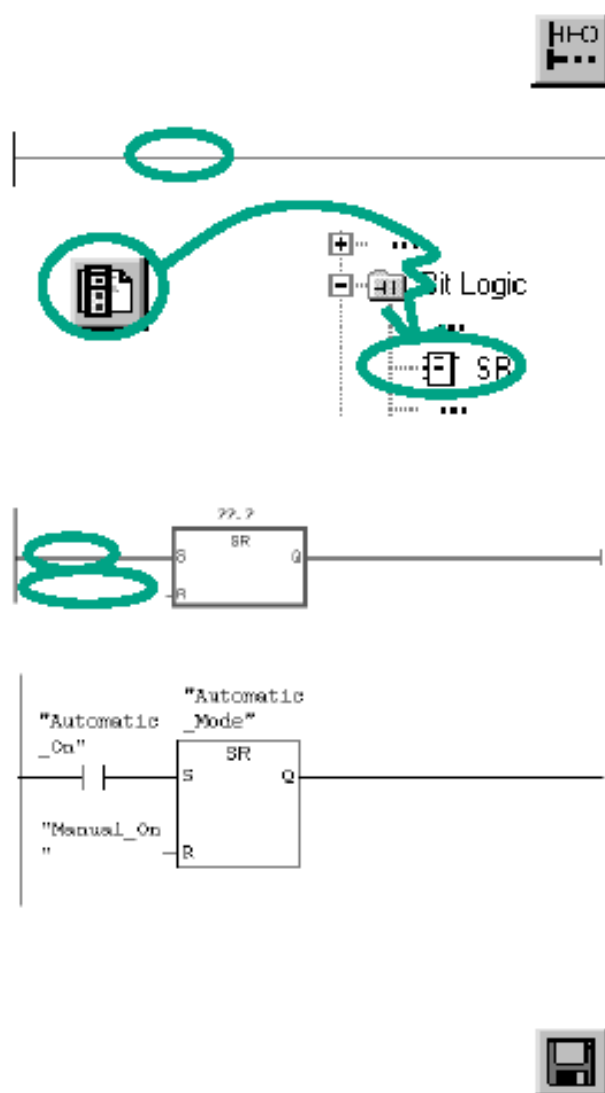


Рис. 59 Функция памяти в контактном плане

Программирование ОВ1 в виде списка операторов

Для программирования команды AND [И] в списке операторов необходимо, установите **STL** в качестве языка программирования в меню **View [Вид]**. Проверьте, активизировано ли символическое представление (Symbol Representation). Щелкните в области **заголовка (title)** ОВ 1 и введите, например, "Циклически обрабатываемая главная программа". Выберите область для своего первого оператора. Напечатайте A (AND) в первой строке программы, пробел, а затем символическое имя "Key_1 [Ключ_1]" (в кавычках). Завершите строку нажатием **Enter**. Курсор переходит на следующую строку (рис.60).

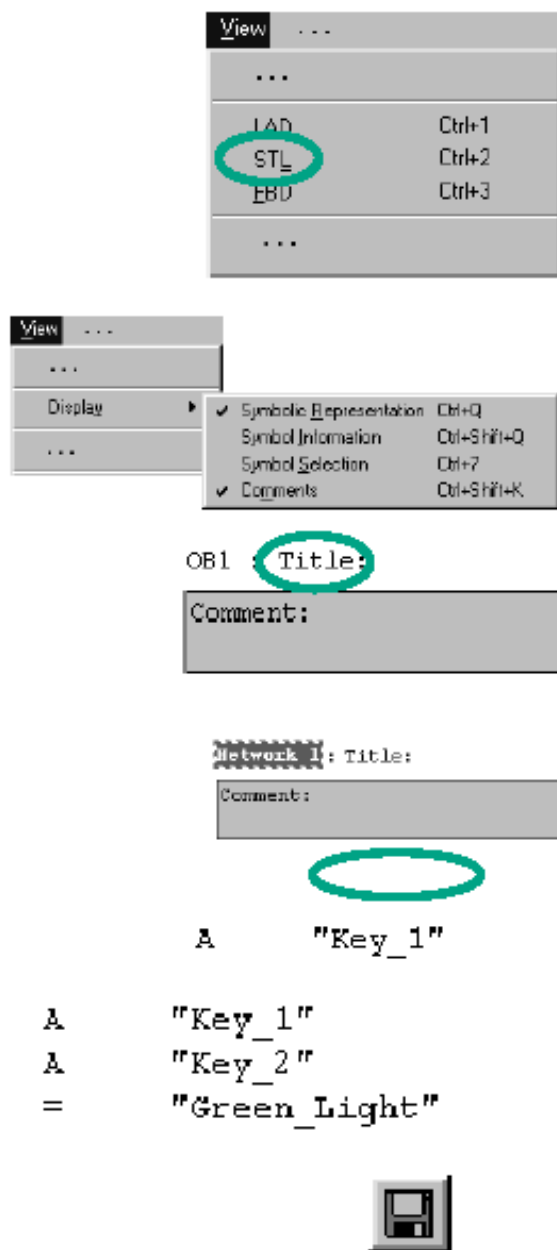


Рис. 60 Программирование команды AND

Таким же образом завершите команду AND [И], как показано слева. Теперь вы запрограммировали всю команду AND. Сохраните блок, если в нем больше нет символов, выделенных красным цветом.

Для программирования команды OR [ИЛИ] в списке операторов символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка. Вы можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке ???.?, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически (рис.61).

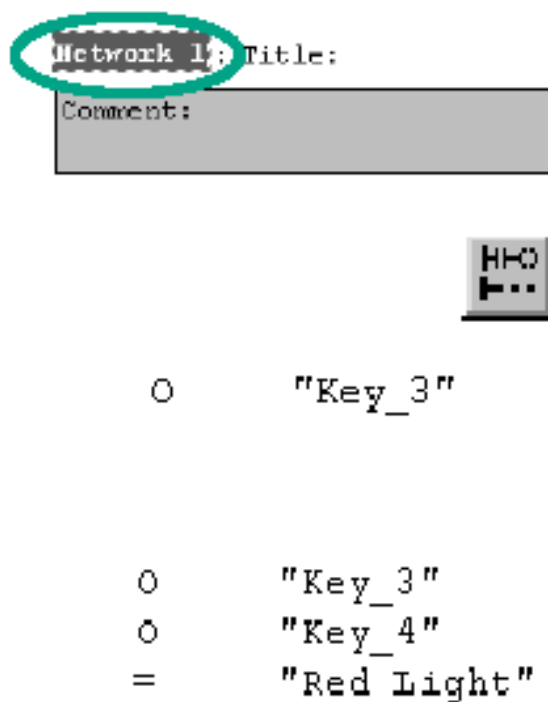


Рис. 61 Программирование команды OR

Выделите **Network 1 [Сегмент 1]**. Вставьте новый сегмент и снова выберите область ввода. Введите O (OR) и символическое имя "Key_3 [Ключ_3]" (так же, как для команды AND). Закончите команду OR и сохраните ее.

Для программирования функции памяти в списке операторов выделите **Network 2 [Сегмент 2]** и вставьте еще один сегмент. В первой строке напечатайте команду A с символическим именем "Automatic_On [Автоматический_режим_включен]". Завершите функцию памяти и сохраните ее. Закройте блок (рис.62).

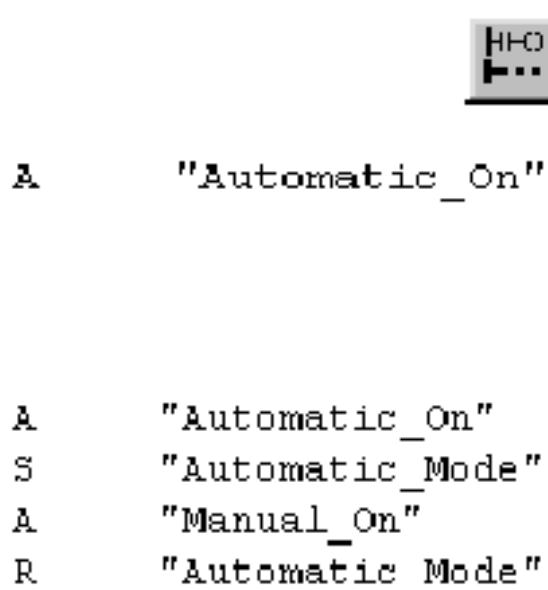


Рис. 62 Программирование функции памяти

Если вы хотите увидеть разницу между абсолютной и символической адресацией (рис.63), деактивизируйте команду меню **View > Display > Symbolic Representation** [**Вид > Отображение > Символическое представление**].

A	"Key_1"
A	"Key_2"
=	"Green_Light"
A	I 0 . 1
A	I 0 . 2
=	Q 4 . 0

Рис. 63

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents** [**Помощь > Содержание**] в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing STL Statements [Редактирование операторов STL]".

Программирование OB1 в виде функционального плана

Для программирование функции AND [И] в функциональном плане необходимо, установите **FBD** в качестве языка программирования в меню **View** [**Вид**]. Щелкните в области заголовка (title) OB 1 и введите, например, "Циклически обрабатываемая главная программа". Выберите область ввода для функции AND (под полем комментария). Вставьте блок AND (&) и присваивание (=). Адреса элементов в функции AND все еще отсутствуют (рис.64). Проверьте, активизировано ли символическое представление (Symbol Representation).

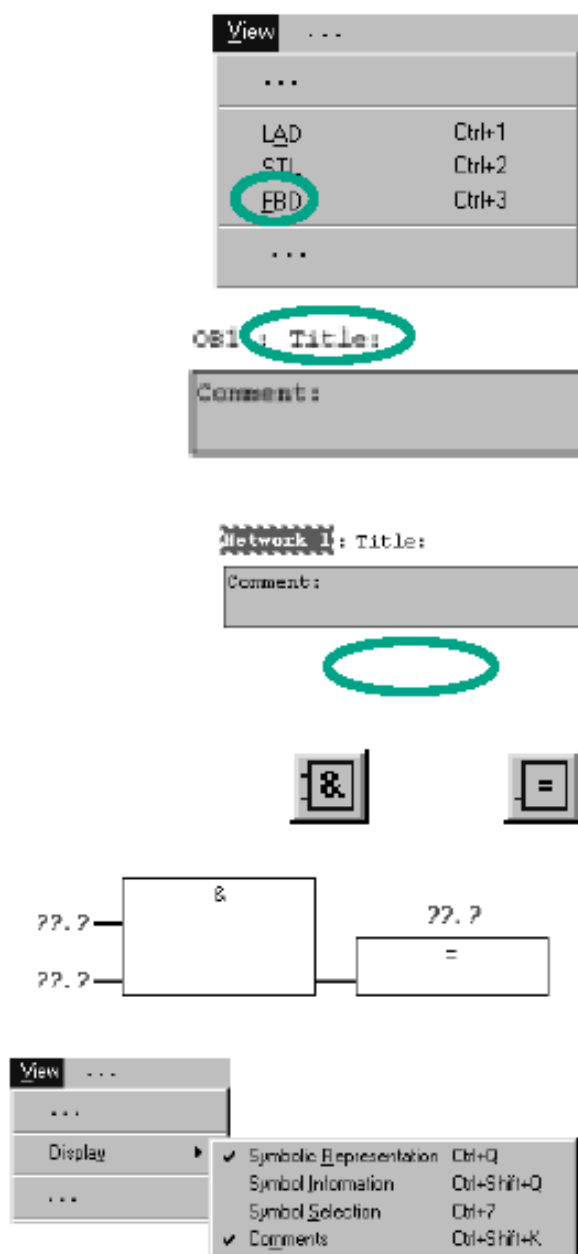


Рис. 64 OB1 в виде функционального плана

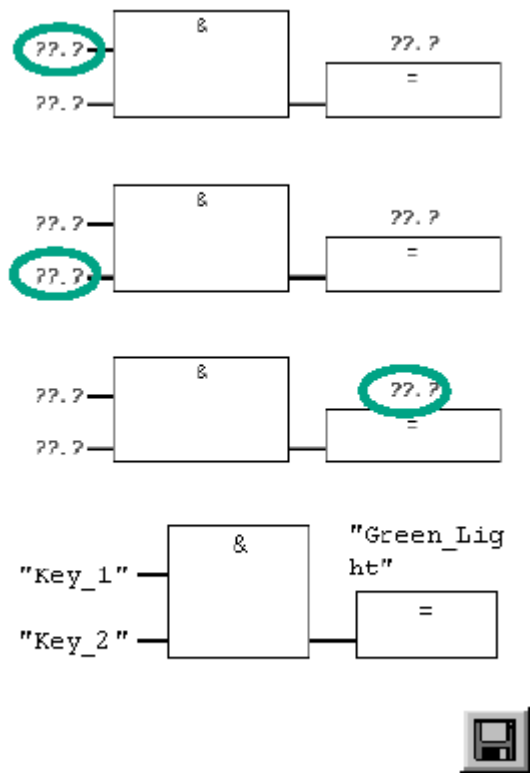


Рис. 65

Щелкните на знаке **??.** и введите символическое имя "Key_1 [Ключ_1]" (в кавычках). Подтвердите, нажав **Enter**. Введите символическое имя "Key_2 [Ключ_2]" для второго входа. Введите имя "Green_Light [Зеленый_свет]" для присваивания. Теперь запрограммирована вся функция AND. Если отсутствуют символы, выделенные красным цветом, можете сохранить блок. Символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка. Можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке **??.**, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически.

Для программирования функции OR [ИЛИ] в функциональном плане вставьте новый сегмент. Снова выделите область ввода для функции OR. Вставьте блок OR (👉1) и присваивание (=). В функции OR все еще отсутствуют адреса. Действуйте так же, как и для функции AND. Введите "Key_3 [Ключ_3]" для верхнего входа, "Key_4 [Ключ_4]" для нижнего входа и "Red_Light [Красный_свет]" для присваивания. Сохраните блок.

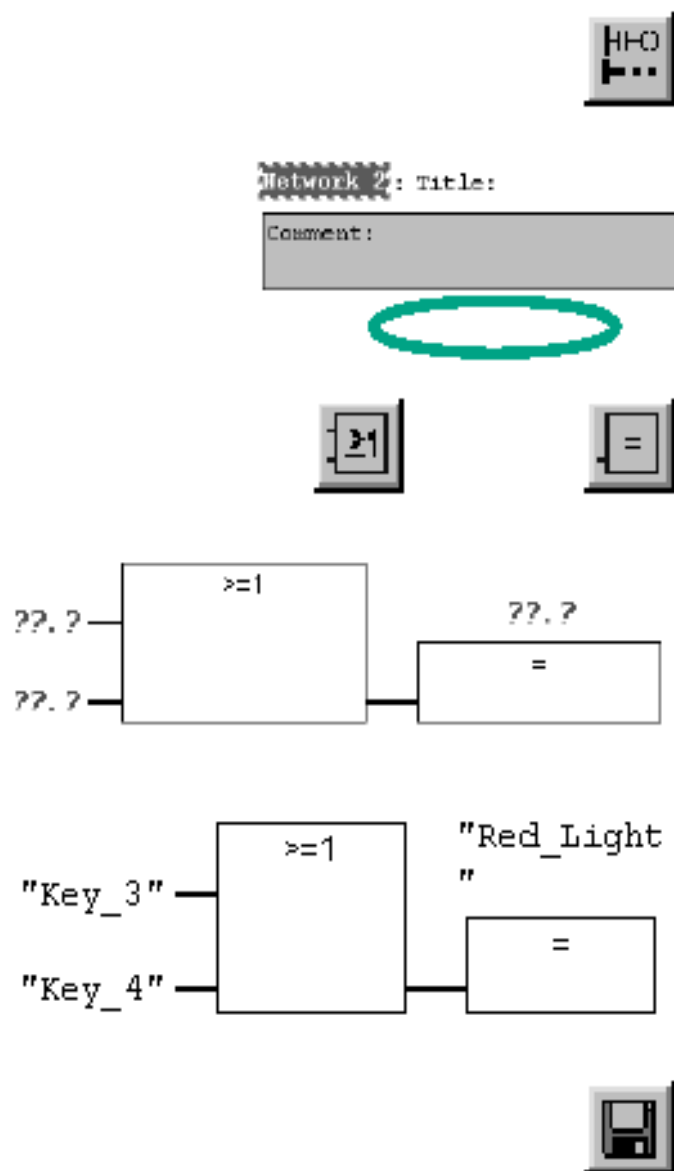


Рис. 66

Для программирования функции памяти в функциональном плане выделите Network 2 [Сегмент 2] и вставьте еще один сегмент. Снова выберите область ввода (под полем комментария). Перемещайтесь в каталоге элементов программы в разделе **Bit Logic [Двоичная логика]**, пока не достигнете элемента **SR**. Дважды щелкните, чтобы вставить этот элемент. "Automatic Mode", "Automatic on", "Manual on".

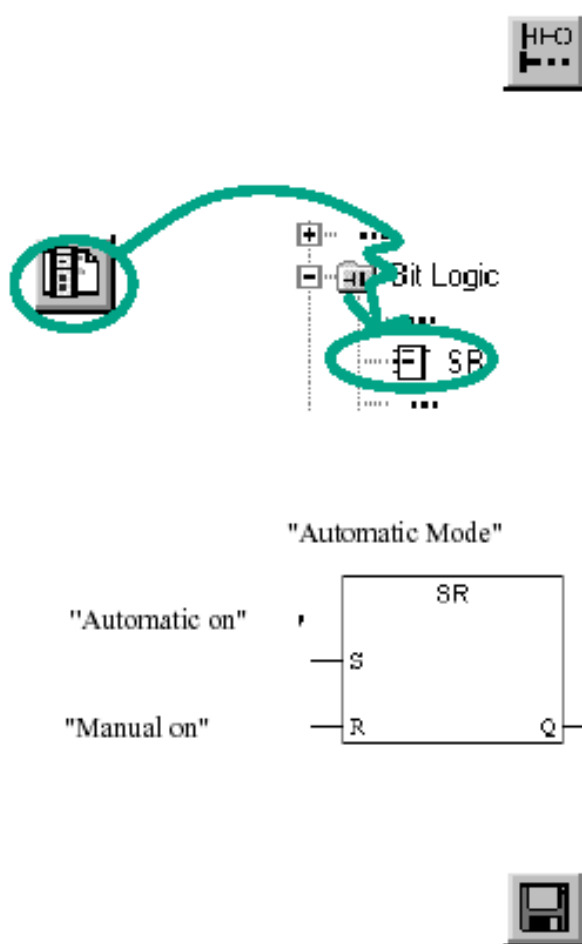


Рис. 67

Введите следующие символические имена для элемента SR:

- Установить (S) Automatic_On;
- [Автоматический_режим_включен];
- Сбросить (R) "Manual_On";
- [Ручной_режим_включен];
- Бит памяти "Automatic_Mode";
- [Автоматический_режим]".

Сохраните блок и закройте окно. Если вы хотите увидеть разницу между абсолютной и символической адресацией, деактивизируйте команду меню **View > Display > Symbolic Representation** [Вид > Отображение > Символическое представление].

Вы можете изменить разрыв строки в символической адресации в окне программирования LAD/STL/FBD с помощью команды меню **Options > Customize** [Параметры > Настроить], выбрав во вкладке "LAD/FBD" "Address Field Width [Ширина поля адреса]". Здесь вы можете установить разрыв строки между 10 и 24 символами.

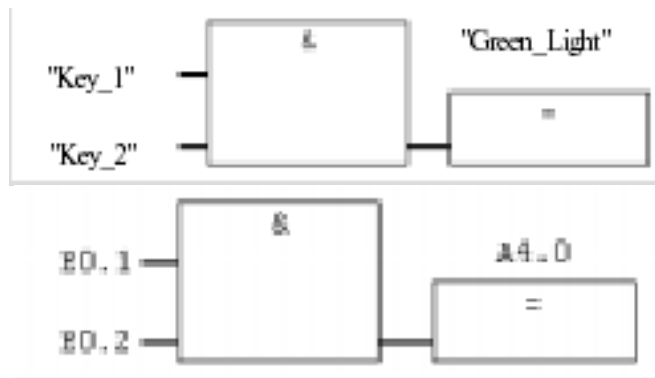


Рис. 68

Дополнительную информацию можно найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing FBD Statements [Редактирование операторов FBD]".

Контрольные вопросы




4. Поясните принцип программирования на языках STL, LAD, FBD?
5. Составьте программу с ячейкой памяти в функциональном плане?
6. Объясните, что значит программирование по абсолютным адресам и символьным переменным?

Требования по содержанию отчета

В отчете студент должен перечислить цели лабораторной работы, описать ход работы, ответить на контрольные вопросы, сделать вывод о проделанной работе.

методические указания по выполнению лабораторных работ по курсу: средства
автоматизации гибких автоматизированных производств

Составители: Леонов Сергей Владимирович
Рудницкий Владислав Александрович

Подписано к печати . Формат 60х84/16. Бумага «Классика». Печать RISO. Усл.печ.л. . Уч.-изд.л. . Заказ . Тираж экз.		
	Томский политехнический университет Система менеджмента качества Томского политехнического университета сертифицирована NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2000	
ИЗДАТЕЛЬСТВО  634050, г. Томск, пр. Ленина, 30.		