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

УТВЕРЖДАЮ
Декан ЭФФ
_____Г.С. Евтушенко
«17» ноября 2008 г.

А.В. Юрченко

ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ

Методические указания к выполнению лабораторных работ
по курсу «Информационно-измерительные системы»
для студентов IV курса, обучающихся по направлению
200100 «Приборостроение»

Издательство
Томского политехнического университета
2008

УДК 681.518.3(07)
ББК 32.973.202:301я73
Ю83

Юрченко А.В.

Ю83 Информационно-измерительные системы: методические указания к выполнению лабораторных работ по курсу «Информационно-измерительные системы» для студентов IV курса, обучающихся по направлению 200100 «Приборостроение» / А.В. Юрченко. – Томск: Изд-во Томского политехнического университета, 2008. – 56 с.

ISBN 5-98298-360-8

УДК 681.518.3(07)
ББК 32.973.202:301я73

Методические указания рассмотрены и рекомендованы
к изданию методическим семинаром кафедры
информационно-измерительной техники ЭФФ
«11» октября 2007 г.

Зав. кафедрой ИИТ
кандидат технических наук *А.Е. Гольдштейн*

Председатель учебно-методической
комиссии *А.Н. Гормаков*

Рецензент

Заведующий кафедрой ИИТ ТУСУРа, профессор
А.А. Светлаков

ISBN 5-98298-360-8

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 1

ПРОГРАММИРУЕМЫЙ ЛОГИЧЕСКИЙ КОНТРОЛЛЕР ELSY-TM

ВВЕДЕНИЕ

Лабораторно-практическая работа посвящена созданию тестовой станции солнечных батарей на основе логического контроллера Elsy-TM. Работа закрепляет на практике знания полученные в разделе курса лекции «Информационно-измерительные системы» раздел стандартные модульные интерфейсы для построения информационно-измерительных систем.

Работа выполняется в 2 этапа. Первый этап посвящен изучению структуры контроллера и системы его программирования в среде OpenPCS.Отработка навыков производится на эмуляторе контролера Elsy-TM. Второй этап посвящен созданию измерительной схемы на базе контроллера, позволяющая проводить тестирования солнечных батарей с использованием эталонного образца солнечной батарей.

1. ЦЕЛЬ РАБОТЫ

1.1. Изучить систему программирования стандарта IEC61131 в среде OpenPCS на базе контроллера Elsy-TM.1.2. Изучить структуру тестовой станции солнечных батарей. Создать программу управления станцией позволяющей проводить калибровку станции и измерения характеристик солнечных батарей.

1.3. Провести измерения характеристик солнечных батарей.

2. ПРОГРАММА РАБОТЫ

2.1. 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», Упражнения 1-4. Проводится на эмуляторе контроллера Elsy-TM

2.2. Создание программ на языках стандарта IEC 6 1131-3 и их отладка в контроллере Elsy-TM. Упражнения 5-7. Проводится на контроллере Elsy-TM.

2.3. Создание программы калибровки тестовой станции солнечных батарей. Упражнения 7-8. Проводится на эмуляторе контроллера Elsy-TM.

2.4. Проведение калибровки тестовой станции и проведение измерений параметров солнечных батарей. Упражнение 9.

2.5. Сделать выводы и подготовить отчет по работе.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ И СРЕДСТВА ИЗМЕРЕНИЯ

Объектами исследования являются кремниевые солнечные батареи, подключенные к нагрузке, обеспечивающий режим тока короткого замыкания.

Для исследования в работе используются следующие приборы:

- контроллер Elsy-TM в составе модулей центрального процессора, аналогового ввода вывода, дискретного ввода вывода, питания и интерфейсного модуля.
- имитатор солнечного излучения, выполненный на базе светодиодов.

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

4.1. Общий обзор и особенности контроллера ЭЛСИ-ТМ

Контроллер предназначен для измерения непрерывных сигналов, представленных

напряжением постоянного тока и (или) постоянным током, сбора и обработки информации с первичных датчиков, формирования сигналов управления по заданным алгоритмам, приема и передачи информации по последовательным каналам связи в системах измерения, контроля и управления объектами нефтяной и газовой промышленности, энергетики и других отраслей.

Контроллер является восстанавливаемым, многоканальным, многофункциональным изделием с переменным составом функциональных модулей и возможностью резервирования источника питания и центрального процессора.

Алгоритм работы контроллера определяется прикладной программой, разрабатываемой пользователем в соответствии с требованиями к системе управления, создаваемой с использованием контроллера.

Основная область применения – системы автоматического и автоматизированного управления технологическими процессами.

Преимущества:

- высокая производительность выполнения пользовательских программ;
- надежность:
- использование элементной базы лучших мировых производителей;

- возможность резервирования источников питания и модулей центрального процессора;
- исполнение модулей в индивидуальном прочном корпусе облегчает их транспортировку и эксплуатацию;
- развитая система самодиагностики;
- «горячая замена» модулей контроллера позволяет менять модули «на ходу» : без переинициализации;
- устойчивость к перегрузкам входных цепей, повышающая живучесть контроллера в аварийных ситуациях;
- ・универсальность:
- гибкая модульная архитектура контроллера с применением разнообразных коммуникационных модулей позволяет создавать оптимальную систему управления, удовлетворяющую самым строгим критериям;
- поддержка различных интерфейсов и протоколов обеспечивает возможность
- совместной работы с оборудованием различных производителей;
- разработка пользовательской программы в среде OpenPCS, поддерживающей 5 языков IEC61131-3. Эта комбинация языков создает универсальную среду программирования, обладающую уникальной производительностью;
- расширенный диапазон рабочих температур позволяет эксплуатировать контроллер практически на всей территории России.

Коммутационная панель контроллера предназначена для механического объединения модулей контроллера ЭЛСИ-ТМ, организации электрических соединений между модулями, а также для монтажа контроллера на месте установки. В зависимости от числа устанавливаемых модулей панель имеет несколько исполнений.

Для обеспечения функции резервирования источника питания и центрального процессора предусмотрены специальные варианты исполнения.

Основные интерфейсные модули.

1. Центральный процессор обеспечивает управление работой контроллера и выполняет проверку работоспособности функциональных модулей, обмен данными между модулями, логическую обработку данных и выдачу сигналов управления в соответствии с прикладной программой. При подаче питания или возникновении сбоя в работе центральный процессор производит перезапуск контроллера.

2. Источники питания используются для обеспечения питанием от сети постоянного или переменного тока модулей контроллера, установленных на коммутационной панели.

Предусмотрена возможность резервирования источников для обеспечения бесперебойного питания контроллера.

3. Модули ввода/вывода обеспечивают получение и преобразование входных сигналов с первичных средств измерения и датчиков и формирование выходных сигналов на исполнительные механизмы.

4. Интерфейсные модули. Связь со SCADA, контроллерами внутри сети и другим технологическим оборудованием обеспечивается интерфейсными модулями по стандартным промышленным протоколам передачи информации – MODBUS RTU, MODBUS TCP/IP, IEC870.5.

Прикладные программа для контроллера ЭЛСИ-ТМ разрабатывается в среде *OpenPCS* фирмы Infoteam. Разработка программ производится на базе языков стандарта IEC61131-3.

Среда позволяет:

- разрабатывать прикладную программу и загружать ее в контроллер;
- производить отладку программы и мониторинг процесса выполнения.

Подробно контроллер и процесс разработки программного обеспечения контроллера описан в технической документации по контроллеру.

4.2. Система программирования ПЛК OpenPCS

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: Grafcet (Telemechanique-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 функциональна 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.

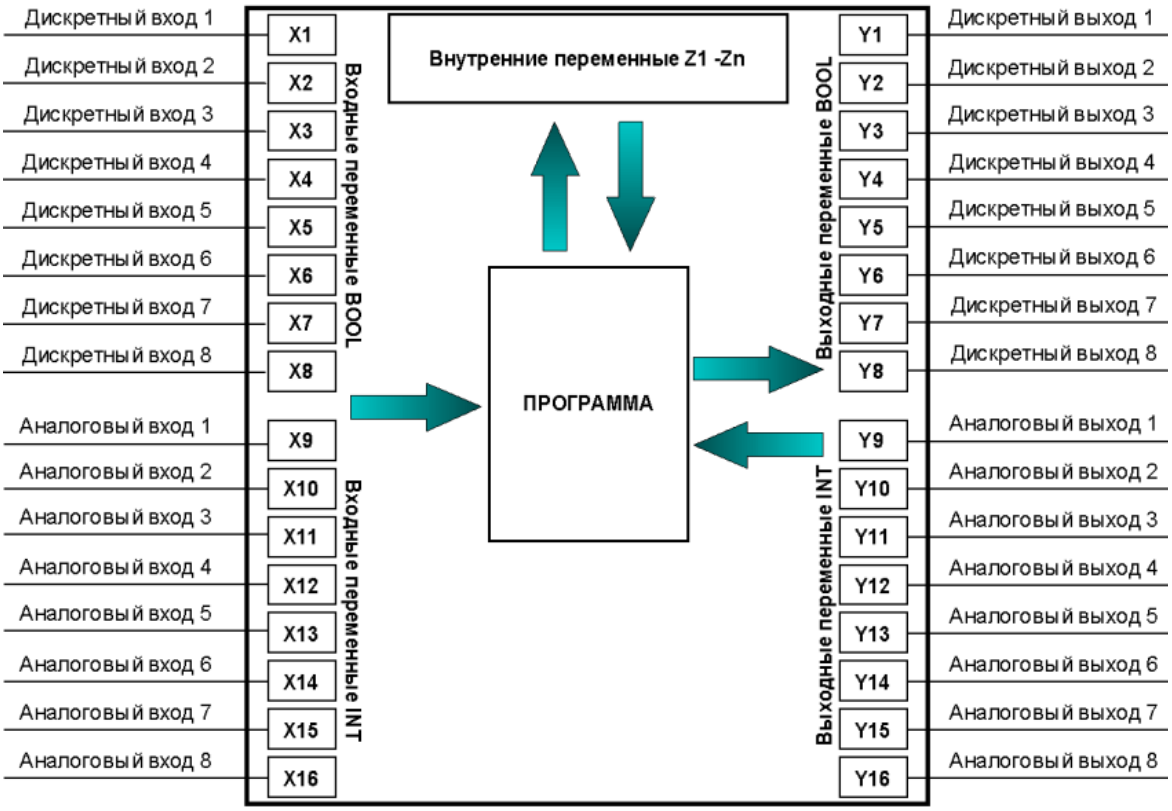
4. **ST** (Structured Text) – текстовый высокоуровневый язык общего назначения, по синтаксису ориентированный на Паскаль. Происхождение: Grafcet (Telemechanique-Groupe Schneider).

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 несущественны, 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 исполняется 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:



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 результат выводит через переменные **Y**, определенные как выходные. Для промежуточных вычислений служат внутренние переменные **Z**.

Упражнение 1: OpenPCS

Запустите OpenPCS (ярлык на рабочем столе  или программная группа infoteam openpcs 2004 в главном меню):

1. OpenPCS требует файл с расширением VAR – файл проекта. Возможна показанная на рис. 1-1 ситуация, когда по умолчанию предлагается загрузить демонстрационный проект.

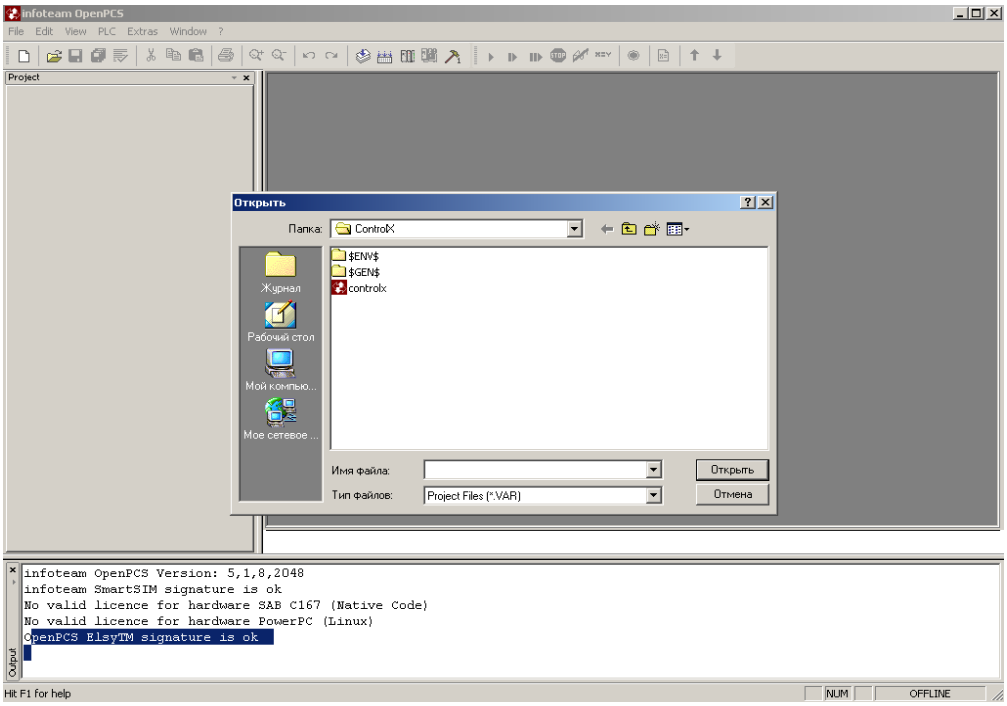


Рис. 1-1

Если рабочий проект в OpenPCS не создавался, выберите предлагаемый проект либо загрузите его по указанному ниже пути.

Факультативно: Ознакомьтесь с демонстрационными проектами, находящимися в директории

C:\Program Files\OpenPCS2004\Samples

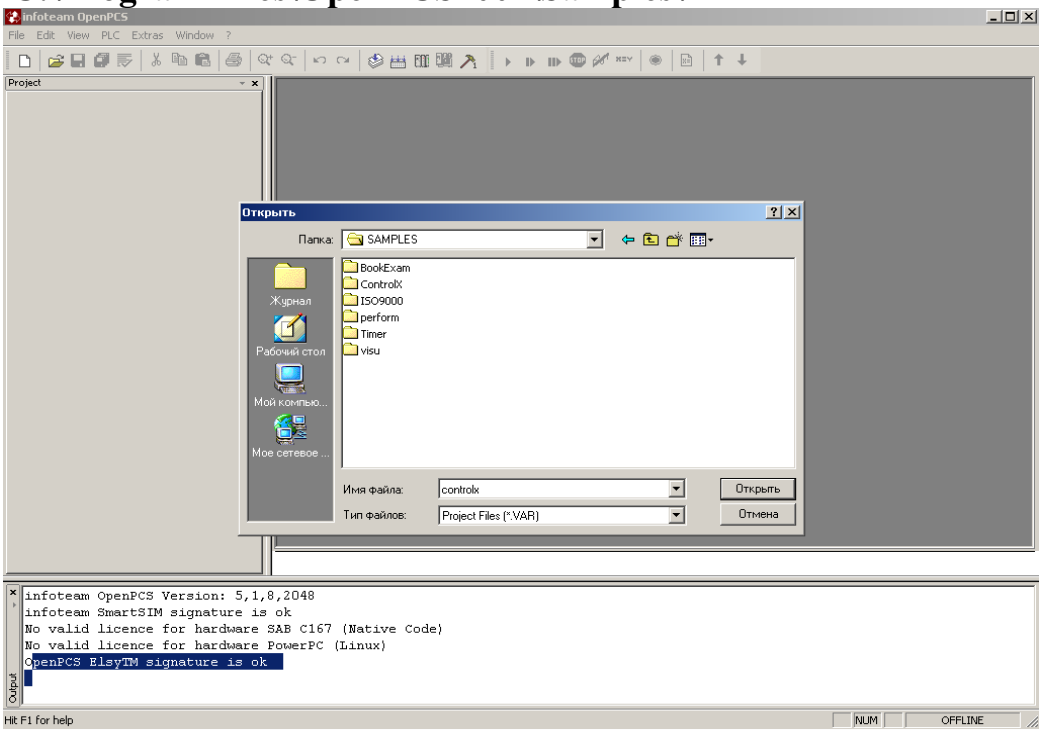


Рис. 1-2

2. Создайте новый проект (**File-> Project-> New**) с именем **Familiya (Ваша фамилия, конечно)** в директории **C:\ws326-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!**).

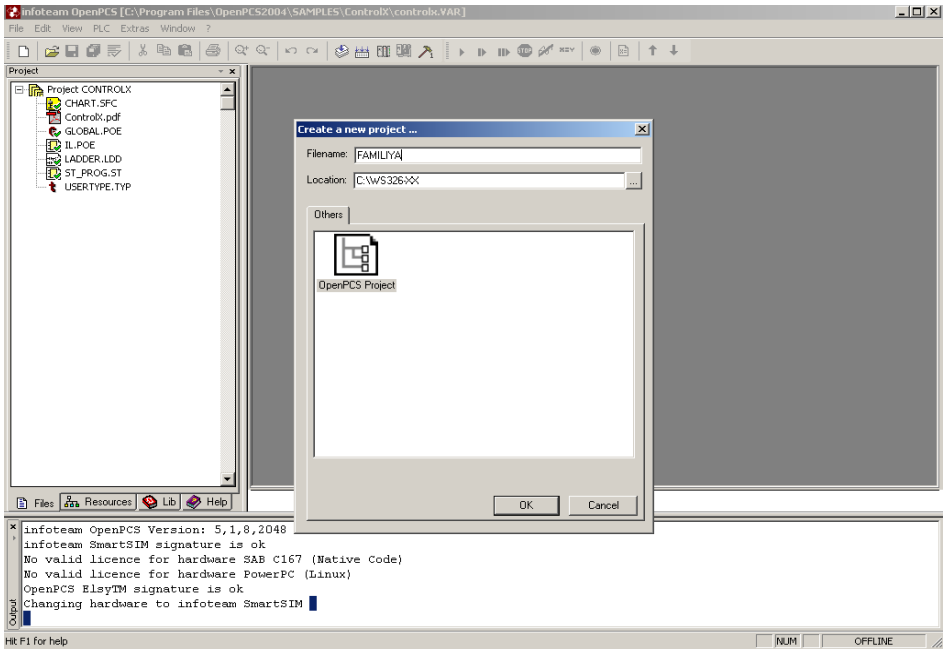


Рис. 1-3

3. Создайте новую программу на ST (**File->New**) с именем **ST1**, на предложение о добавлении созданной программы к активному ресурсу ответить отказом:

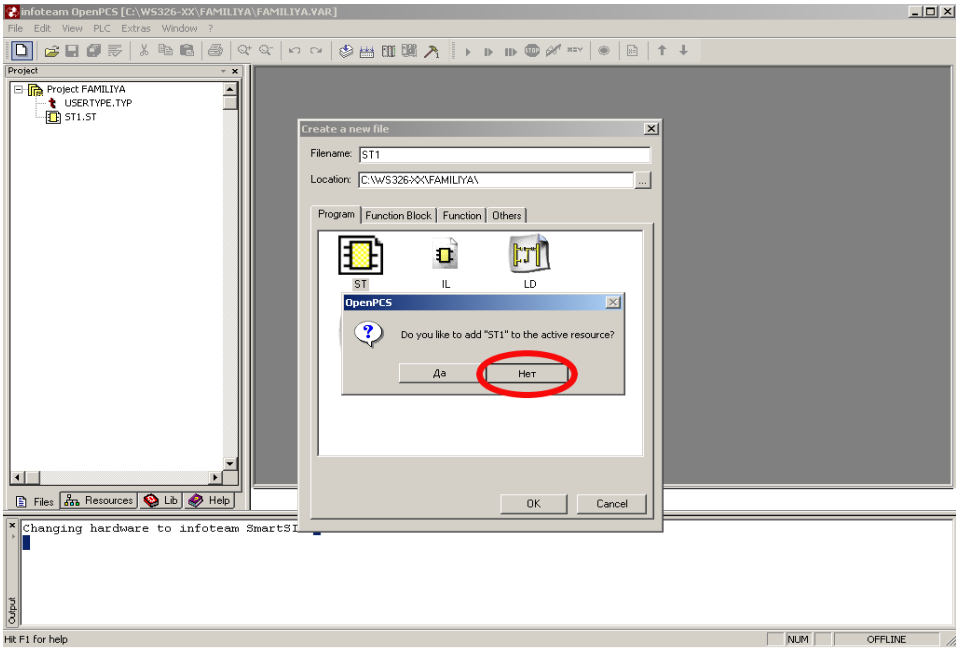


Рис. 1-4

4. Ознакомьтесь с ключевой для дальнейшей работы информацией:

Различные секции (разделы) описания:

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 Переменную, которой необходимо 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, должны описывать их 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**

AT 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
```

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

(1) Может использоваться, но лишь однократно.

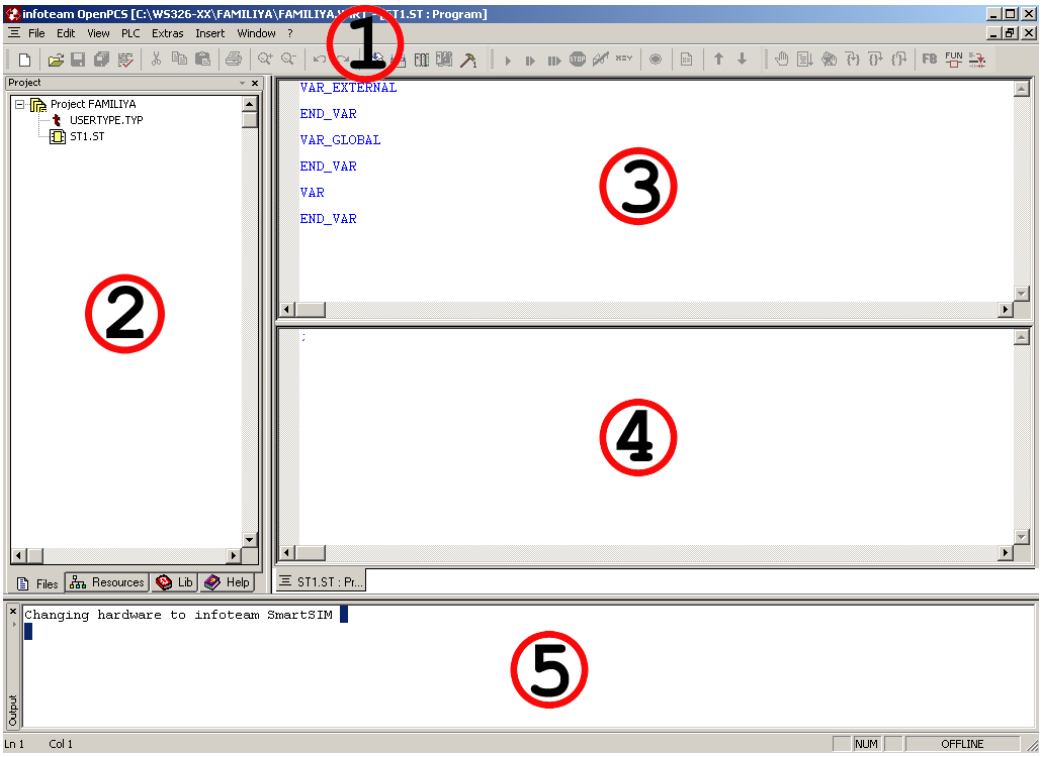


Рис. 1-5
1 – меню и панель инструментов: 2 – менеджер проектов;
3 – редактор переменных; 4 – редактор кода;
5 – окно диагностики и тестирования

5. В редакторе переменных VAR опишите локальные переменные.

Таблица 1-1

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

- **Valve_In_ST** – это говорящее имя переменной (задвижка, входной, программа на языке ST)

Опишите функциональные блоки (это термин OpenPCS)RS-триггеров (в сущности, это подпрограммы-процедуры)**T1_ST:RS** и **T2_ST:RS:**

- **T1_ST** – имя функционального блока, **RS** – его тип.

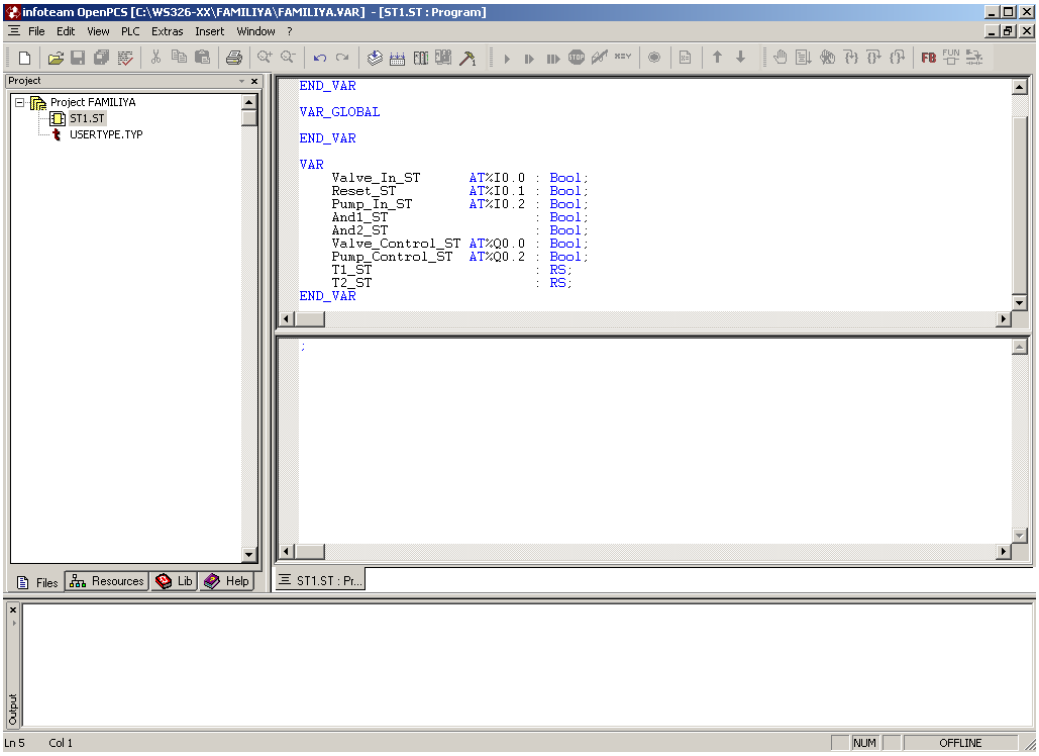


Рис. 1-6

6. Введите текст программы:

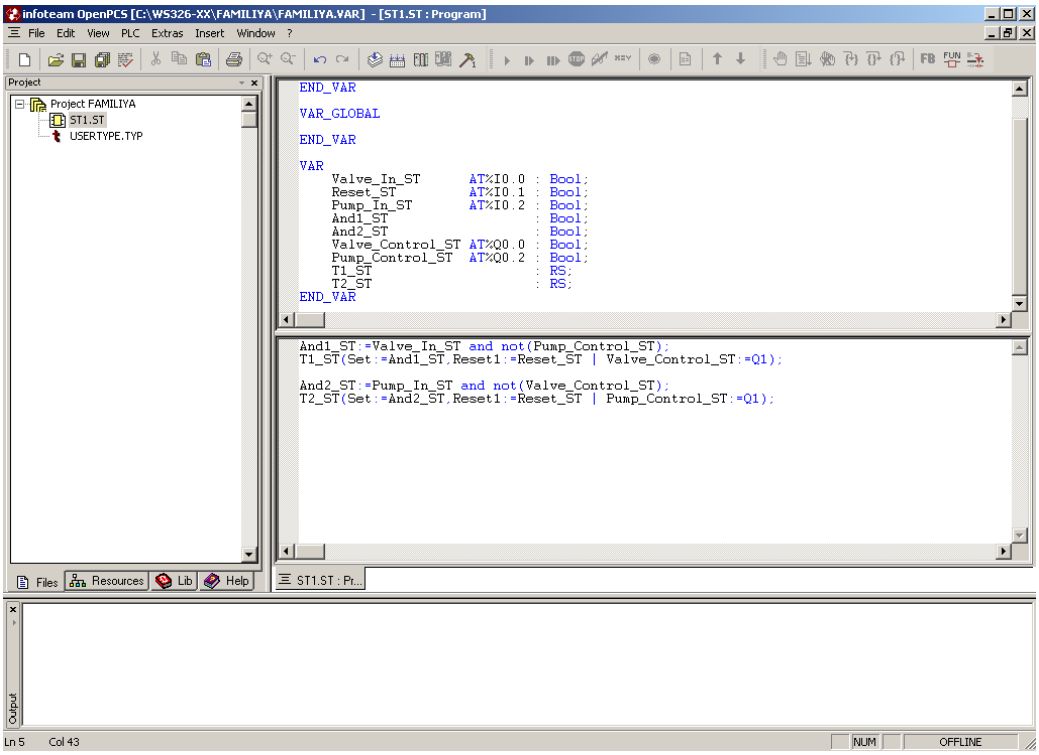


Рис. 1-7

Комментарий к фрагменту программы, поясняющий ее работу

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.**

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**.

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

7. Теперь созданную программу необходимо проверить на синтаксические ошибки (**File-> Check Syntax(Alt+F10)**) при отсутствии ошибок будет выдано следующее сообщение:

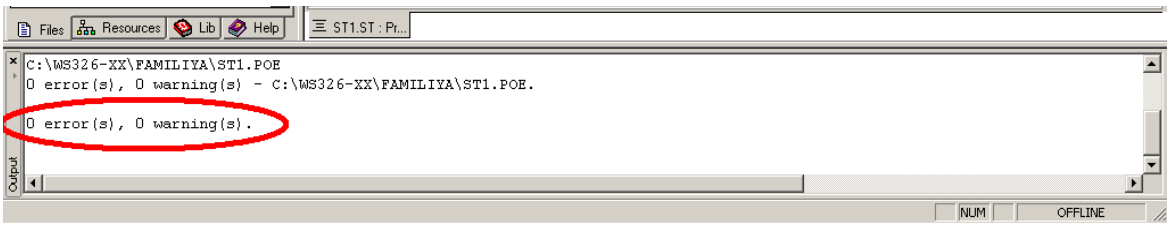


Рис. 1-8

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

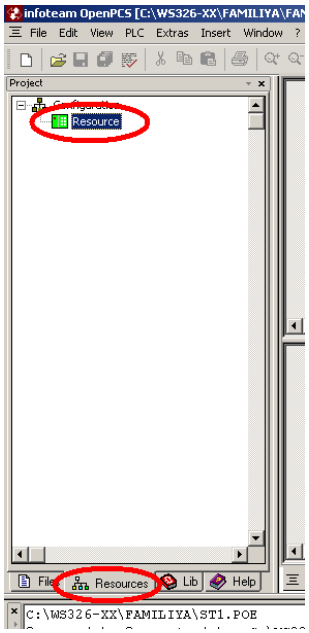


Рис. 1-9

9. Настройте ресурс для работы с симулятором промышленного контроллера:

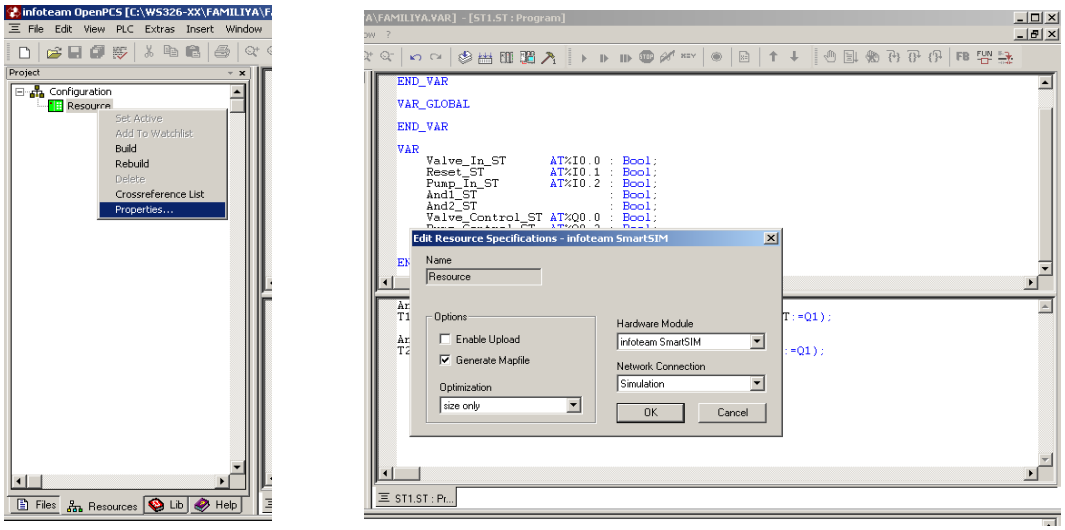


Рис. 1-10

10. Свяжите написанную программу с активным ресурсом:

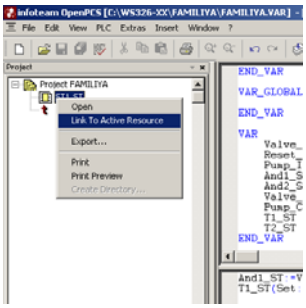


Рис. 1-11

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

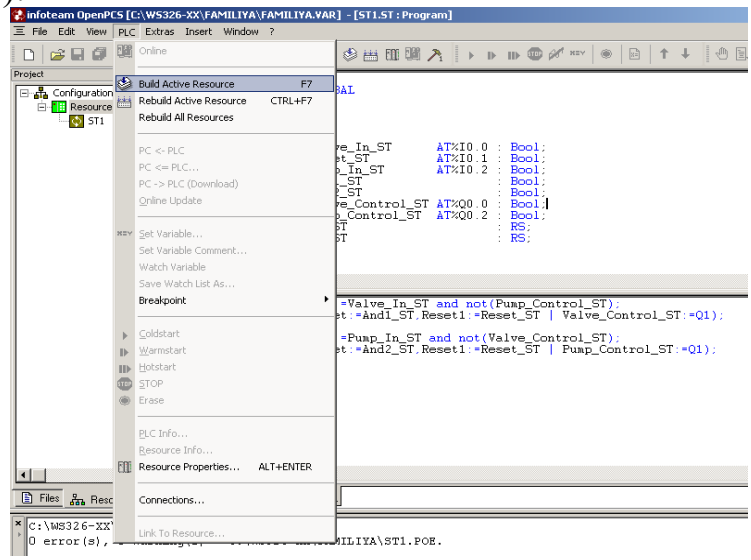


Рис. 1-12

При компиляции будет создан исполняемый код **Start_Stop.PCD** и сгенерированы сведения о наличие ошибок и предупреждений, как показано на следующем рисунке:

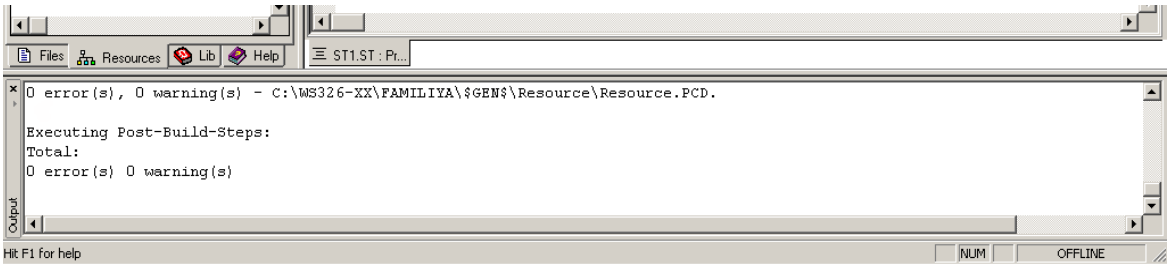


Рис. 1-13

12. Свяжитесь с «контроллером» (PLC-симулятором):

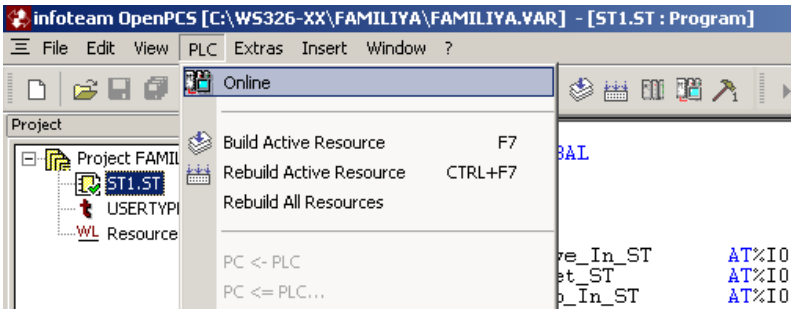


Рис. 1-14

13. Загрузите в симулятор код:

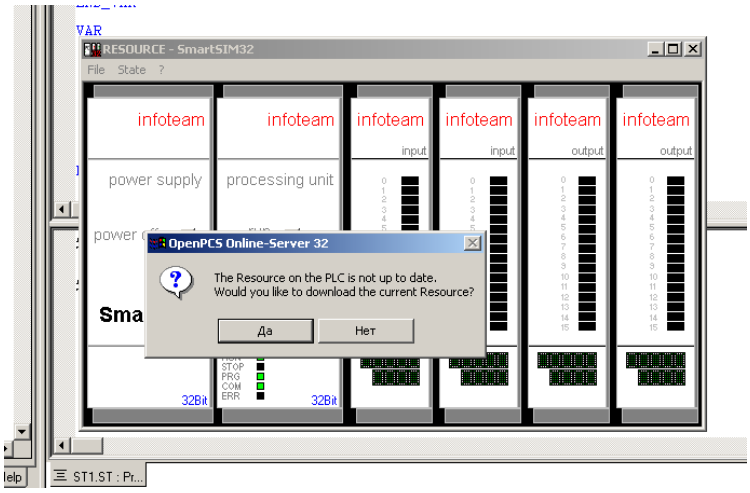


Рис. 1-15

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. Алгоритм работы программы «Старт-стоп»)

описан неоднократно. Напомним еще раз: представьте лифт, в котором три кнопки: вверх, стоп, вниз. Можете себе представить механизм, который едет одновременно и верх и вниз?

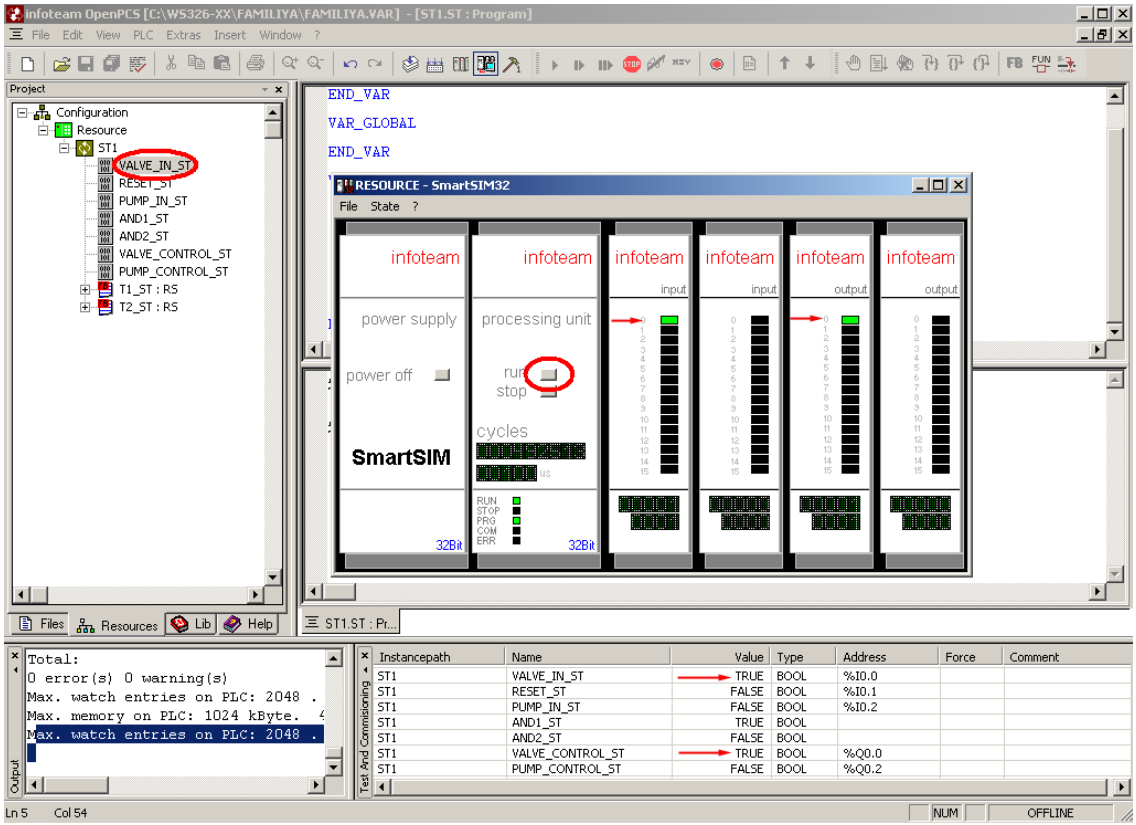


Рис. 1-16

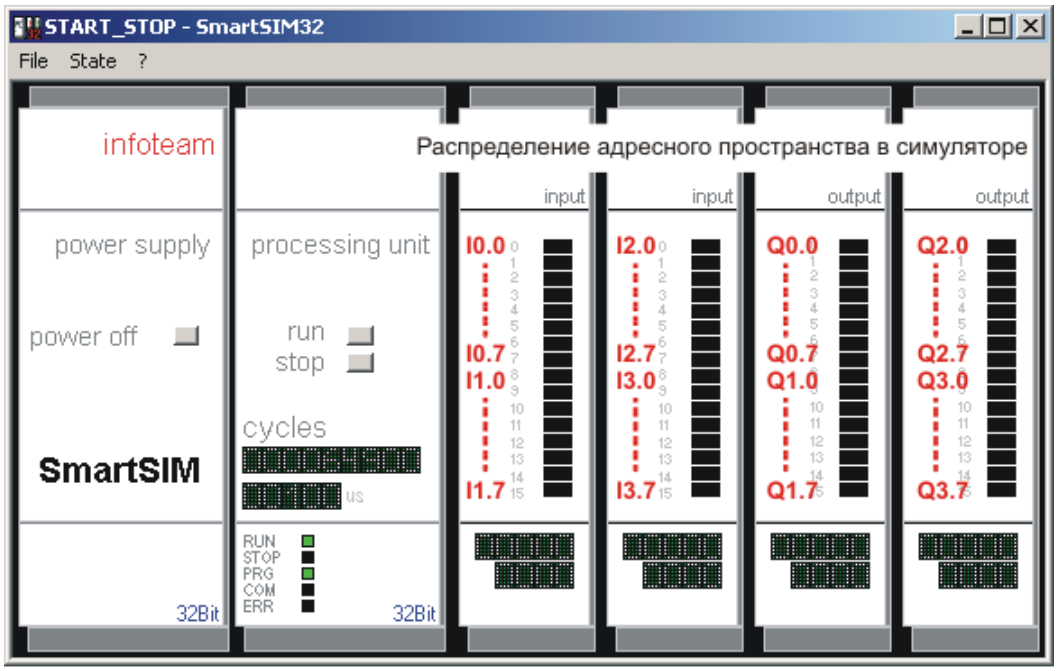


Рис. 1-17

На рис. 1-17 показано 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:

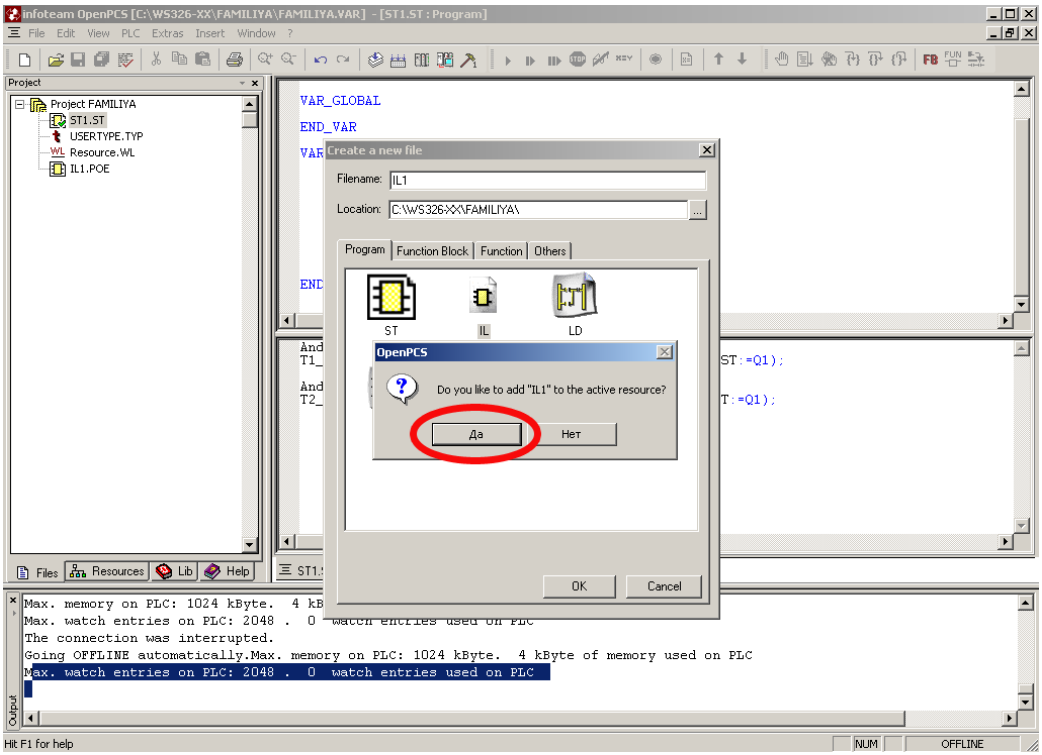


Рис. 2-1

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**:

Таблица 2-1

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.

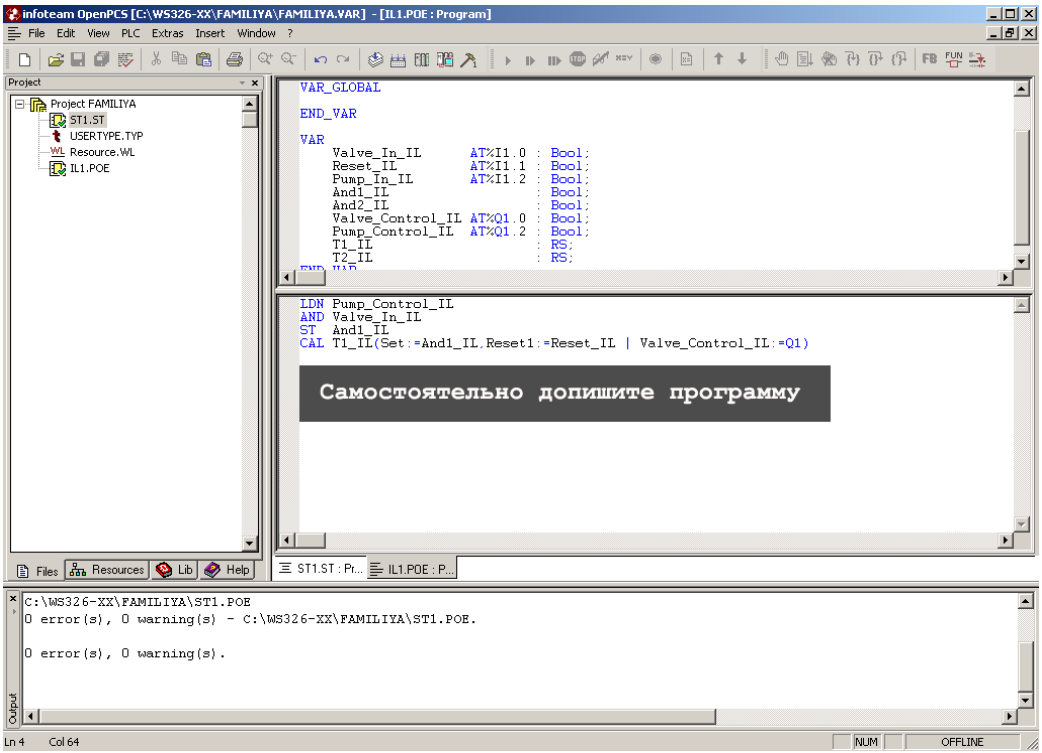
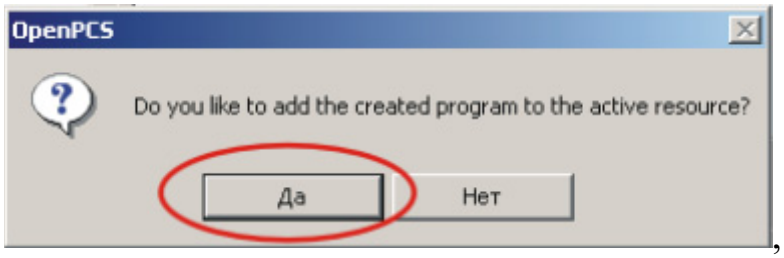


Рис. 2-2

4. Если вдруг программа не добавлена к ресурсу на этапе создания новой программы



то сделайте это принудительно:

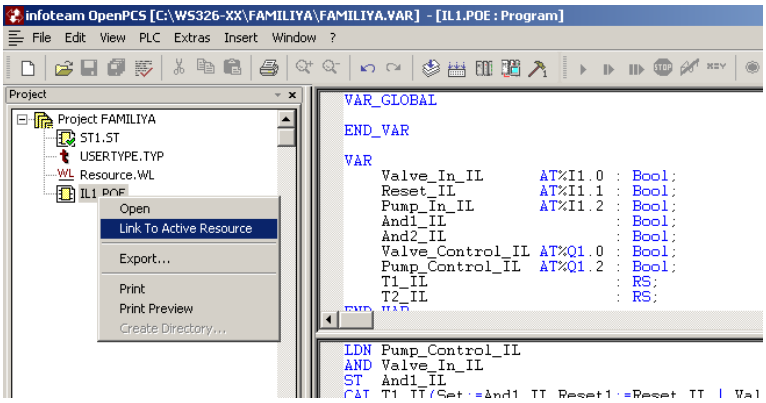


Рис. 2-3

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 рис. 1-17.

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 выглядит, как показано на рисунке: поскольку язык графический, редактор выглядит довольно специфично.

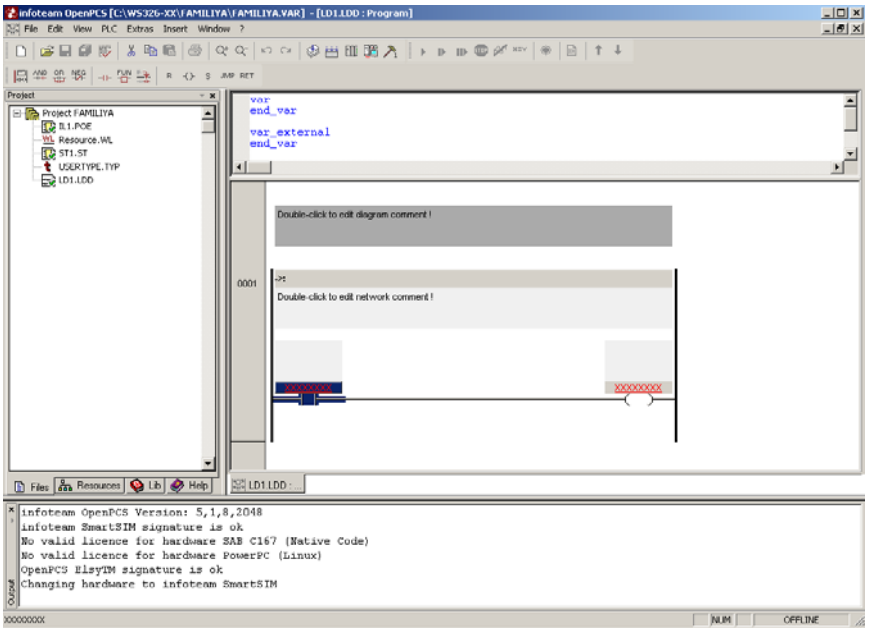


Рис. 3-1

Элемент, показанный на рис. 3-2, носит название элементарная сеть, это заготовка для создания элемента программы:

Рис. 3-2

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

Таблица 3-1

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

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

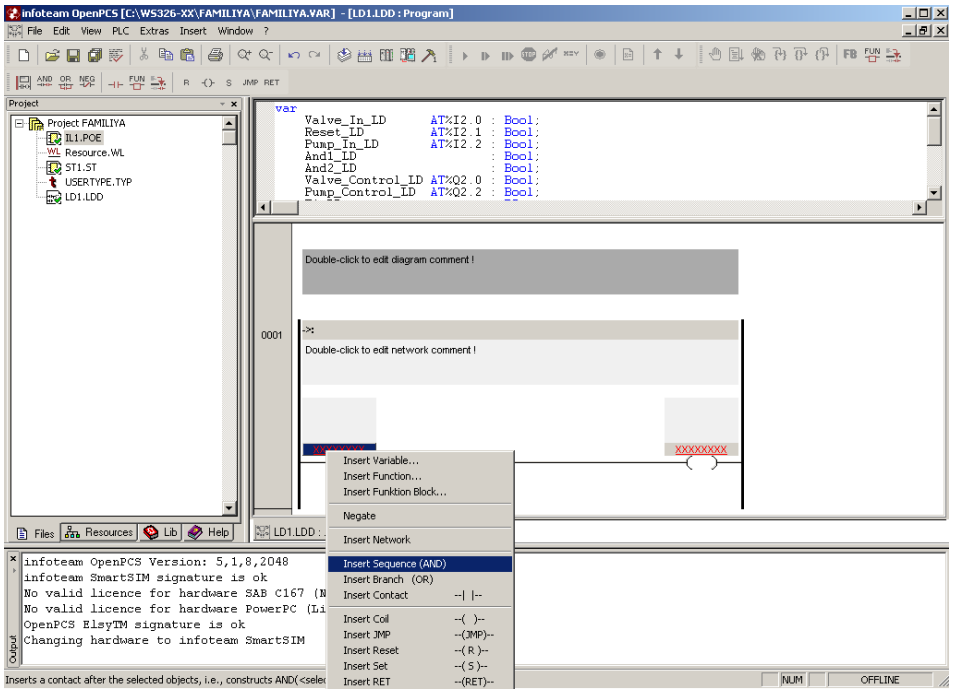


Рис. 3-3

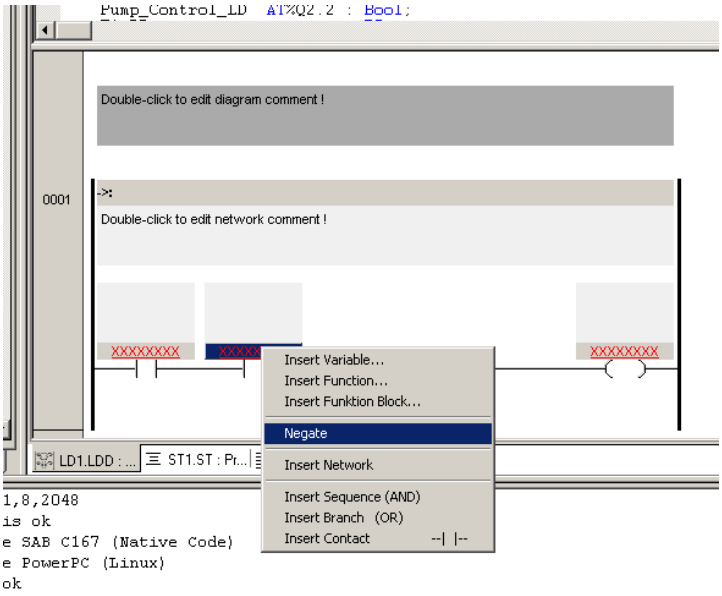


Рис. 3-4

3. Присвойте элементам сети переменные:

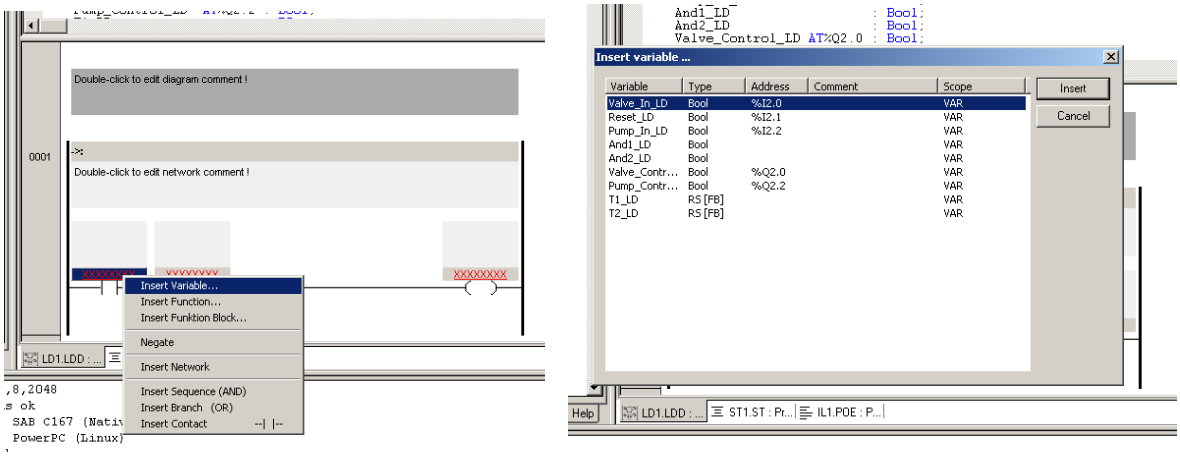


Рис. 3-5

4. Добавьте новую сеть:

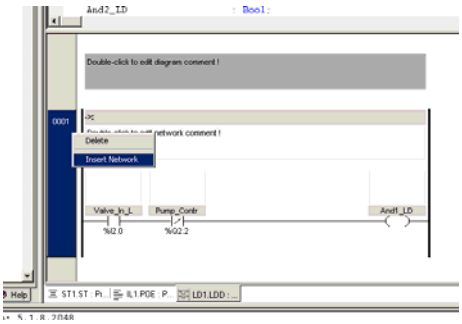


Рис. 3-6

5. Вставьте функциональный блок RS-триггер:

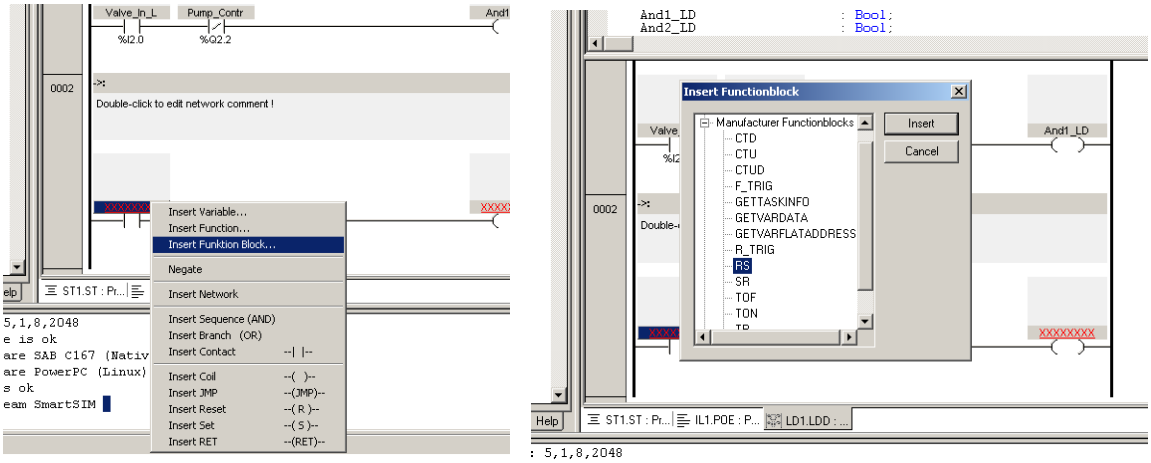


Рис. 3-7

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

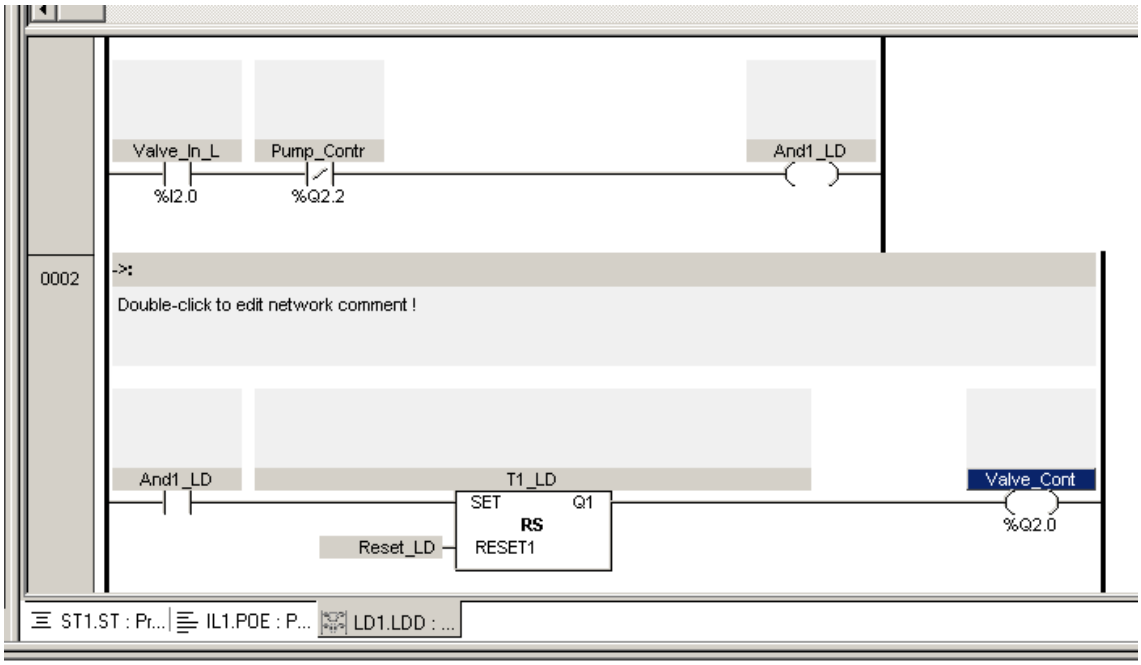


Рис. 3-8

7. 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.

Примечание. Так как этой графический язык ориентирован на создание программ по виду похожих на принципиальную блочную схему, то будет уместным привести задачу именно в подобном графическом виде, тем более что эта схема уже хорошо знакома по работам в GWX32:

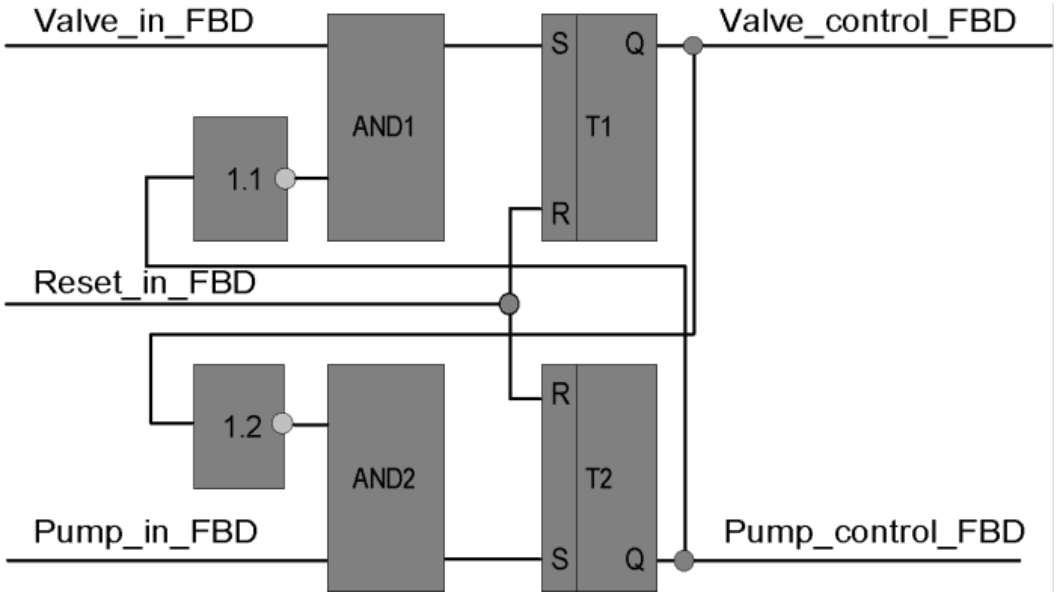


Рис. 4-1

1. Остановите симулятор и создайте новую программу **FBD1** 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 – **T1_FBD** и **T2_FBD**. Обратите внимание на адреса сигналов:

Таблица 4-1

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	Нет	Bool
And2_FBD	Нет	Bool
Valve_Control_FBD	AT%Q3.0	Bool
Pump_Control_FBD	AT%Q3.2	Bool

3. В панели инструментов выберите (см. рис. 4-2) **Insert -> Block, NOT(*BOOL*)** (логическое отрицание) и вставьте блок правым кликом мыши в левый верхний квадрат и добавьте остальные блоки:

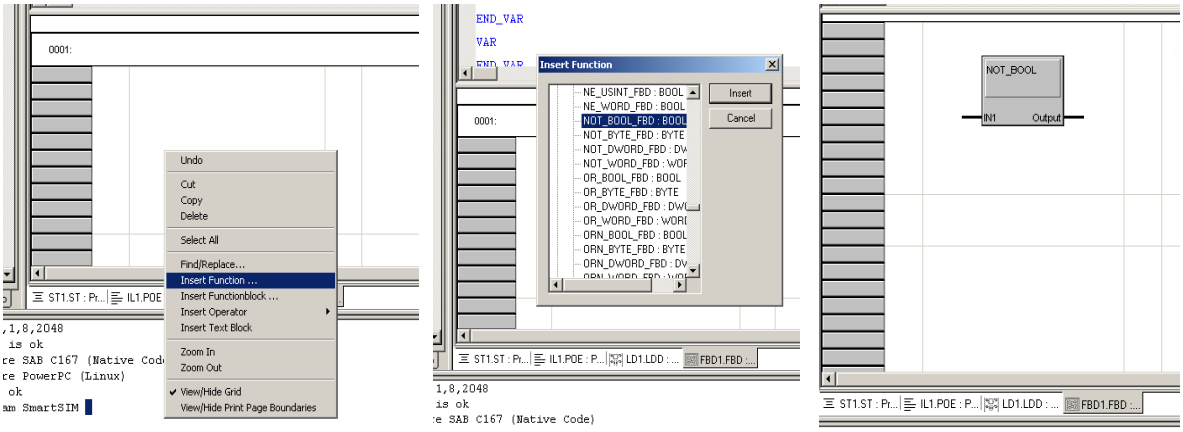


Рис. 4-2

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

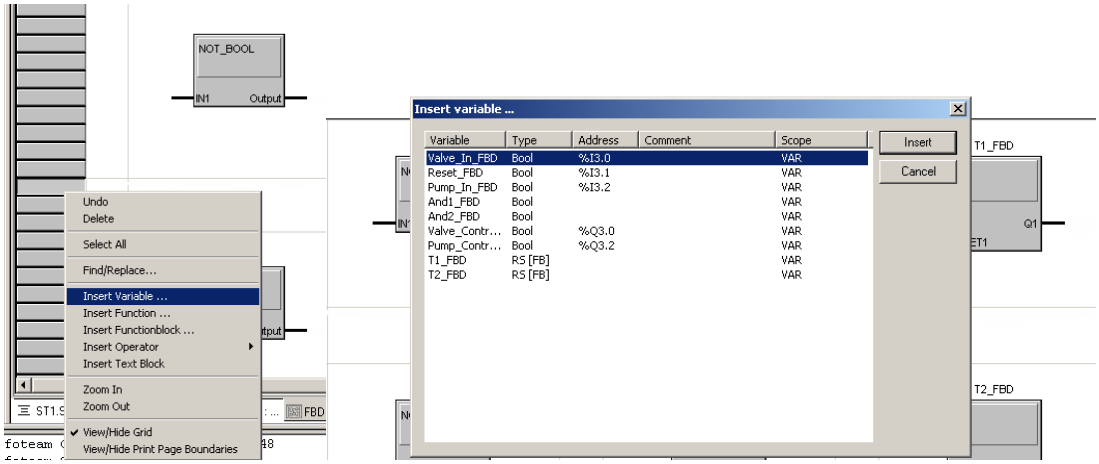


Рис. 4-3

5. Соедините блоки, выбрав мышкой необходимые выводы и применив инструмент **Connection**:

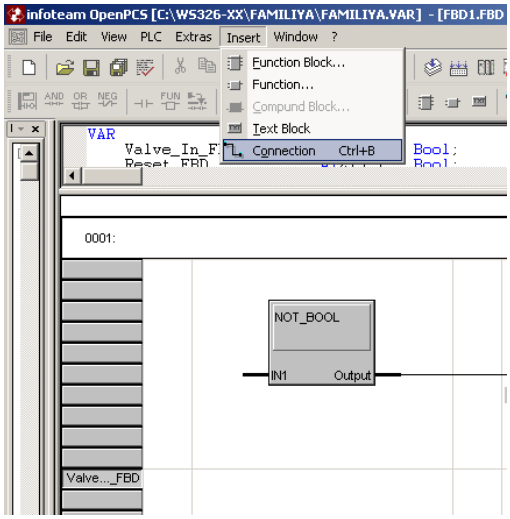


Рис. 4-4

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.

Упражнение 5: Создание программ на языках стандарта IEC 6 1131-3 и их отладка 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. 5-1.

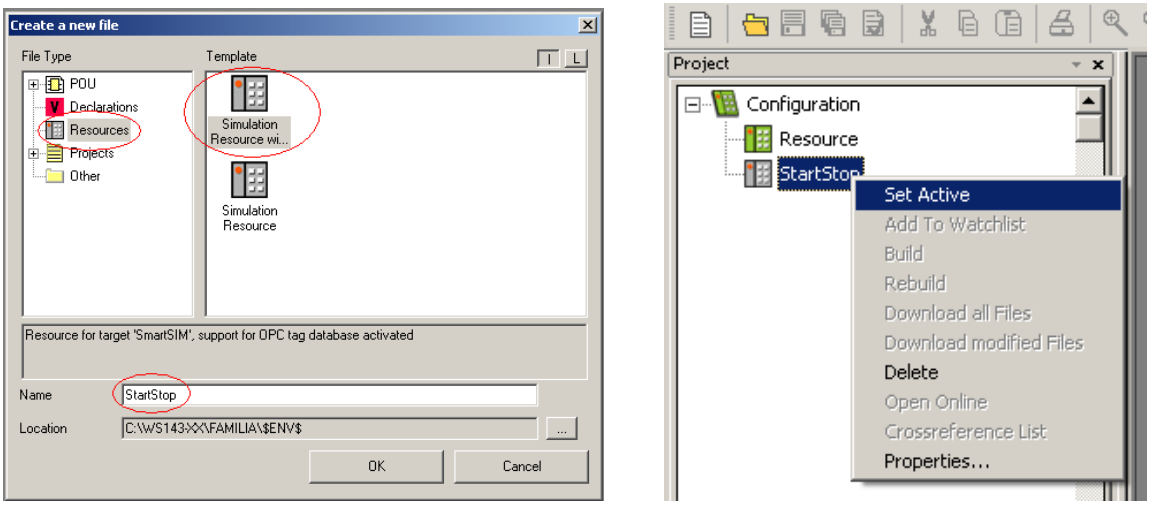


Рис. 5-1

5. **Проверьте** и при необходимости **настройте** параметры соединения. Для этого перейдите в главном меню OpenPCS **«PLC-> Connections...»**. Активным должно быть выбрано соединение **PLCXX**, где **XX**-номер вашего лабораторного стенда, обязательно 2 цифры (например, для стебнда №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 выбираются по табличным данным:

Таблица 1

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.** Остальное оставить без изменений. Рис. 5-2.

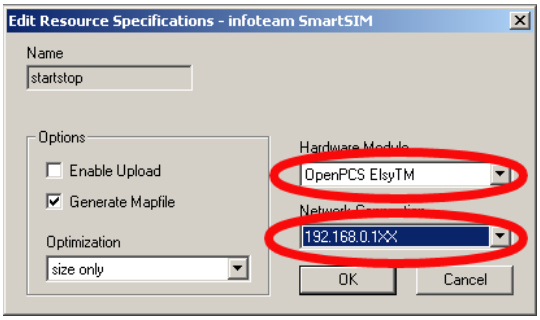


Рис. 5-2

Упражнение 6: 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**.

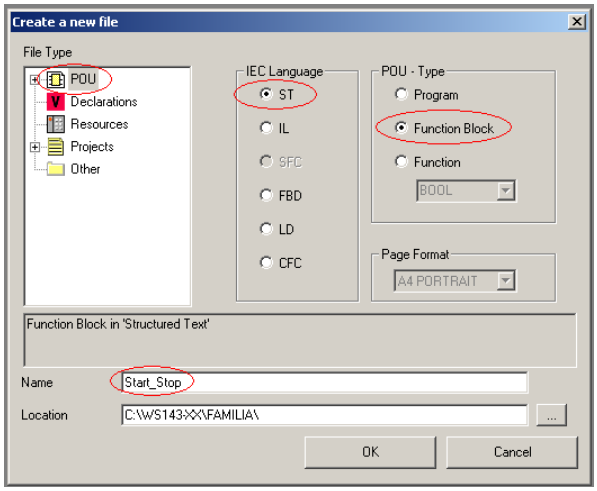


Рис. 6-1

Опишите переменные исходя из табличных данных:

Таблица 6-1

Название сигнала	Вид переменной	Тип переменной
Valve_In	входной	Bool
Reset	входной	Bool
Pump_In	входной	Bool
And1	внутренний	Bool
And2	внутренний	Bool
Valve_Control	выходной	Bool
Pump_Control	выходной	Bool

7. Опишите функциональные блоки **T1_PLC:RS** и **T2_PLC:RS**

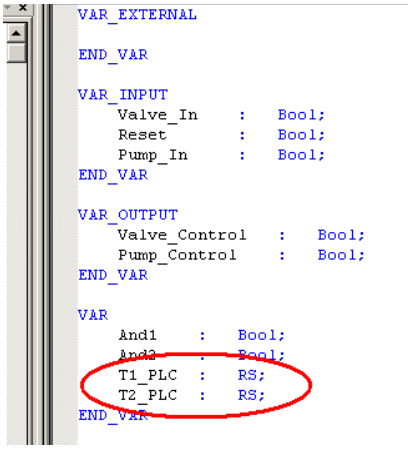


Рис. 6-2

8. Создайте код ФБ который реализует логику старт-стоп (аналогичный код был создан в 5 упражнении).

9. 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 рис. 6-3, и удостоверьтесь в наличии вашего блока:

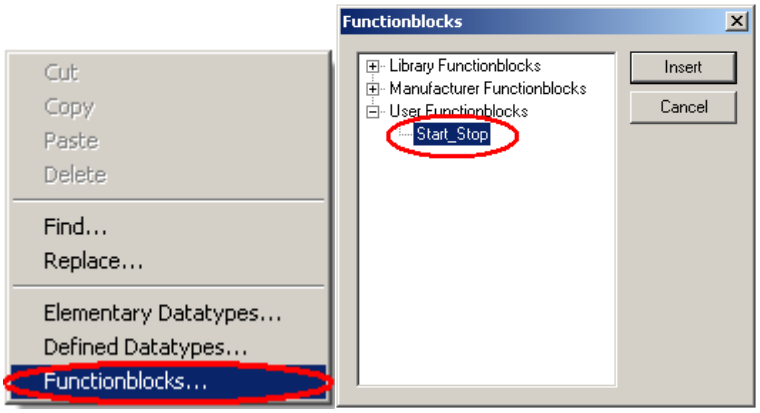


Рис. 6-3

10. Импортируйте 2 файла **Variables.POE** и **Transport.ST** из папки **Методические указания:**

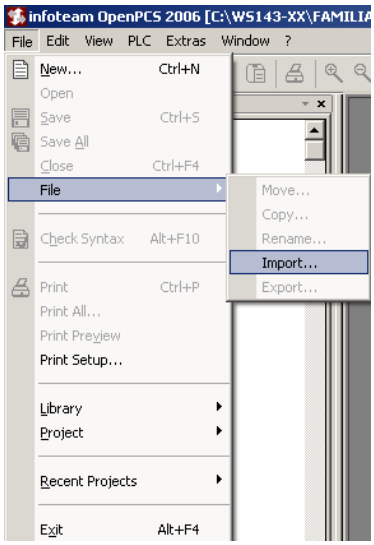


Рис. 6-4

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).

11. Добавьте импортированные файлы к активному ресурсу.

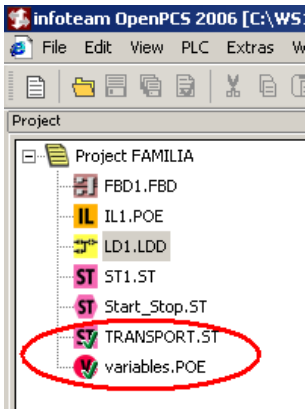


Рис. 6-5

12. Создайте новую **программу** с именем **Familia1** на языке **FBD.** Откройте **Variables.POE** – здесь описаны глобальные переменные, которые перекладываются в физические адреса ПЛК Elsy-TM в Transport.ST. Скопируйте переменные, приведенные в таблице 6-2, из **Variables.POE** в вашу программу.

Таблица 6-2

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

13. Опишите функциональный блок **Start_stop1:Start_stop;** во внутренние переменные **.** Вставьте функциональный блок **Start_stop** (вход **D_In_3_2** необходимо **проинвертировать**)**:**

Соедините входа 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.

14. Проверьте программу на синтаксические ошибки. Откомпилируйте (plc->bild), свяжитесь (plc->online) и прошейте программу в ПЛК Elsy-TM. Запустите программу в контроллере при помощи OpenPCS2006 **PLC-> Coldstart** или соответствующей кнопкой на панели инструментов. Признаком того, что программа запущенна в контроллере служит мигающий индикатор на модуле TC505 (рис. 6-6).

Рис. 6-6

15. При помощи пульта расположенного на учебном лабораторном стенде (рис. 6-7) 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.

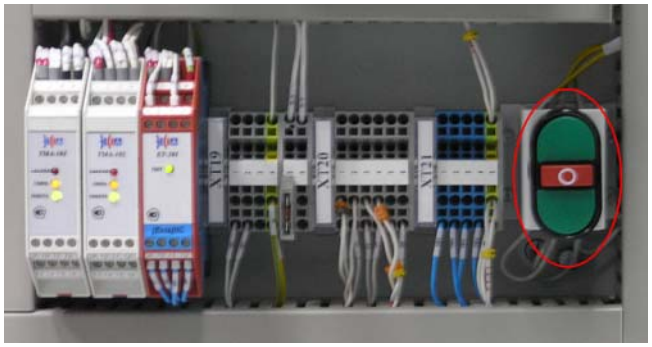


Рис. 7-7

Упражнение 7: Создание ФБ, обеспечивающего ввод десятичного числа с помощью 3 кнопок управления.

1. Создайте новый ресурс под именем DC_1 (как создавать ресурс, описано в предыдущей лабораторной работе). Сделайте новый ресурс активным, настройте его соединение так же как и в предыдущей лабораторной работе.

2. Создайте новый ФБ на языке ST с именем DC1, который будет выполнять функцию преобразования двоичного кода в десятичное.

3. Алгоритм работы программы:

Вариант №1

Верхняя кнопка отвечает за увеличение по единице, если значение выхода достигает 255, то выход обнуляется и продолжается с нуля.

Нижняя кнопка отвечает за увеличение по десятке, если значение выхода достигает 255, то выход обнуляется и продолжается с нуля.

Средняя кнопка сбрасывает число в 0.

При длительном удержании, через 2 секунды должно начать автоматически увеличиваться выход, если удерживается верхняя, то по единице, если нижняя, то по десятке, с периодом полсекунды.

Пример

Допустим нам надо ввести число 33

Вводится следующим образом: нажимаем верхнюю кнопку, вводится число 1 если будем удерживать больше 2 секунд должно выводиться 2, 3, 4… и т. д. до 33 или кратковременными нажатиями (33 раза) на верхнюю кнопку, или 3 раза на нижнюю и 3 раза на верхнюю.

Вариант №2

Алгоритм работы программы:

Верхние кнопки отвечают за позицию бита в байте, признак завершения ввода числа служит восьмое нажатие кнопки.

Нижние кнопки отвечают за значение бита, если кнопка была нажата, то в бит записывается 1, если не нажимается то 0.

Средняя кнопка сбрасывает число в 0.

Затем полученное двоичное число переводим в десятичное.

Пример

Допустим нам надо ввести число 33

1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	1

Вводится следующим образом: вверх, вверх, вниз и не отпуская кнопки нажимаем вверх, вверх, вверх, вверх, вверх, вниз и не отпуская кнопки нажимаем вверх. 8-ой бит это 2 в нулевой степени, соответственно равен единице. 3-ий бит это 2 в пятой степени и соответственно равен 32, что в сумме с 8-ым битом дает 33.

Вариант №3

Алгоритм работы программы:

Верхняя кнопка отвечает за увеличение числа, когда значение достигает 255, сбрасывается в 0 и начинает увеличиваться дальше.

Нижняя кнопка отвечает за уменьшение числа, когда значение достигает 0, сбрасывается в 255 и начинает уменьшаться дальше.

Средняя кнопка сбрасывает число в 0.

При длительном удержании, через 2 секунды должно начать автоматически увеличиваться или уменьшаться значение выход, если удерживается верхняя, то по единице увеличивается, если нижняя, то по единице уменьшается, с периодом полсекунды.

Пример

Допустим нам надо ввести число 33

Вводится следующим образом: нажимаем вверх 33 раза, или удерживаем верхнюю кнопку пока счетчик не досчитает до 33 (если перескочили, нажимаем вниз или обнуляем и повторяем).

Вариант №4:

Алгоритм работы программы:

Верхние кнопки отвечают за позицию бита в байте, признак завершения ввода числа служит девятое нажатие кнопки.

Нижние кнопки отвечают за значение бита, если кнопка была нажата, то в бит записывается 1, если не нажимается то 0.

Средняя кнопка сбрасывает число в 0.
Затем полученное двоичное число переводим в десятичное.

Пример
Допустим нам надо ввести число 33

1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	1

Вводится следующим образом: вверх, вверх, вверх, вниз, вверх, вверх, вверх, вверх, вниз, вверх. 8 бит это 2 в нулевой степени, соответственно равен единице. 3 бит это 2 в пятой степени и соответственно равен 32, что в сумме с 8 битом дает 33.

5. ФБ должен иметь 3 входных (In1, In2, In3 типа Bool) и 1 выходную (Out1 типа Uint) переменную, количество внутренних переменных и их название не имеет значение. Код ФБ должен выполнять описанный алгоритм. Для правильной работы блока, вам понадобятся стандартные ФБ **R_TRIG** или **F_TRIG**, **TP, TON** или **TOF,** а также функции **If** и/или **Case** (описание ФБ и функций можно в «Руководство пользователя Система программирования OpenPCS» (файл с названием OpenPCS 35E2 Rus.pdf) ст 96, 189, также можно воспользоваться встроенным справочником-помощью в программу OpenPCS). Проверьте ФБ на синтаксические ошибки.

6. Откройте файл Variables.POE и скопируйте следующие переменные, которые будут использоваться в программе как внешние:

Таблица 1

Название сигнала	**Вид переменной**	**Тип переменной**
D_In_3_1	внешняя	Bool
D_In_3_2	внешняя	Bool
D_In_3_3	внешняя	Bool

Опишите ваш ФБ DC1 . И сопоставьте им входа и выхода. Для DC на вход подаются D_In_3_X на выход Out_Uint. После того как программа будет готова, проверьте ее на синтаксические ошибки и откомпилируйте. После того как код не будет содержать ошибок, прошейте его в контроллер и запустите программу.

7. Для ввода чисел используйте пульт, для проверки правильности работы блока используйте окно мониторинга и вставьте туда переменную Out_Uint.

4.2. Тестовая станция солнечных батарей

Упражнение 8: Пошаговая калибровка солнечной батареи

Лабораторный стенд состоит из светодиодного источника света и двух солнечных батарей (далее СБ). Яркость источника регулируется ПЛК ElsyTM путем подачу управляющих воздействий на устройство аналогового вывода TMA-102. Данные с СБ поступают на один из аналоговых входов ПЛК. Переключение между тестовой и эталонной СБ происходит посредством реле.

Для выполнения работы необходимы следующие параметры:

Эталонное значение тока КЗ СБ (Iэ)	
Сопротивление нагрузки СБ	

Погрешность измерения примите p=0,05.

1. Создайте новый ресурс с названием «SB». Сделайте его активным.
2. Создайте программу на языке ST, которая бы реализовывала алгоритм, представленный на рис. 8-1.
3. Замерьте время проведения калибровки.

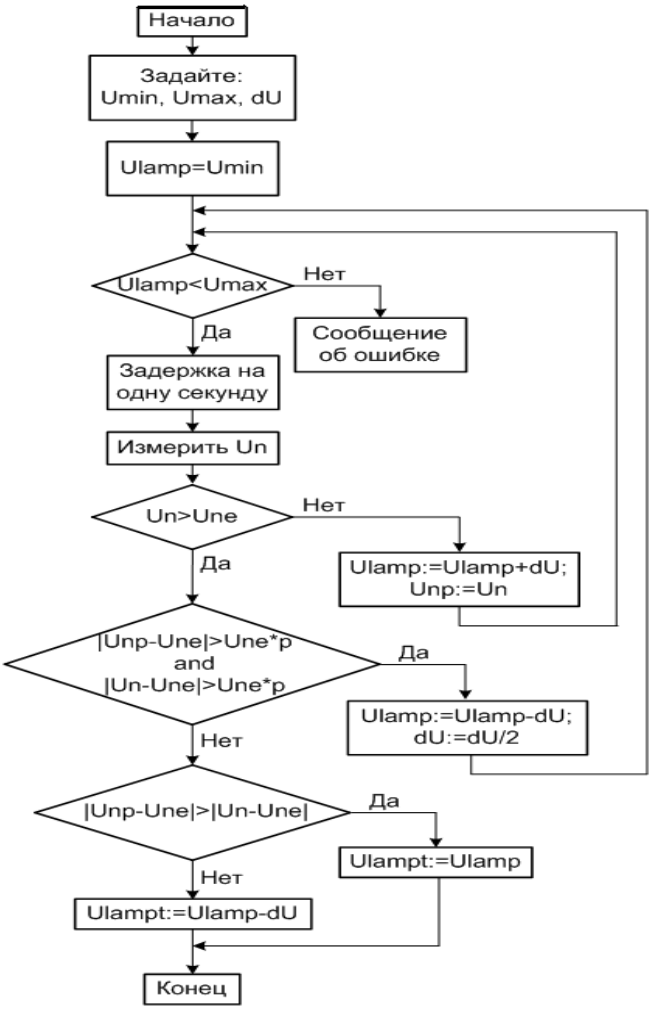


Рис 8-1

Упражнение 9: Калибровка солнечной батареи по двум точкам

Реализуйте алгоритм, представленный на рис. 9-1.

1. Сравните полученное значение Un с эталонным Une. Если они не будут совпадать в пределах погрешности – скорректируйте Ulampt.
2. Сравните времена калибровок по обоим способам.

Упражнение 10: Измерение параметров солнечной батареи

1. Включите СБ при помощи реле.
2. Включите лампу с Ulampt и измерьте Un.
3. Рассчитайте Iсб.

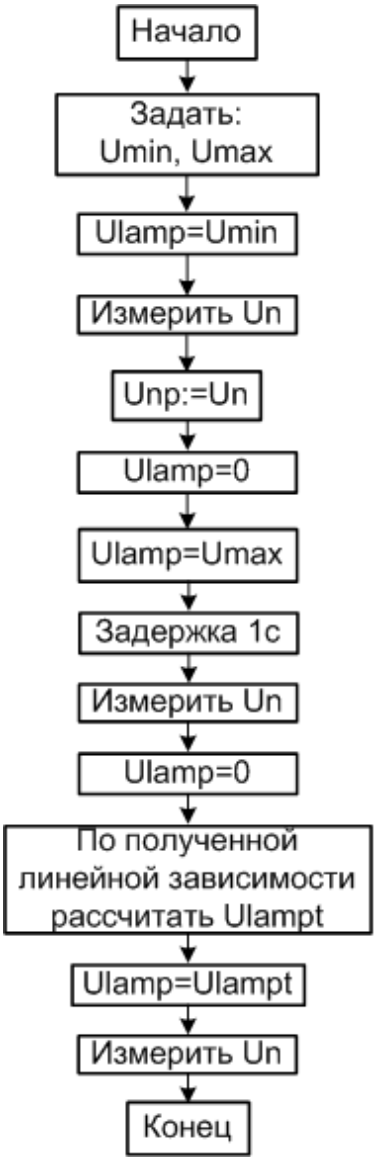


Рис 9-1

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Название и цель работы
2. Структурная схема измерения
3. Таблица экспериментальных данных. График зависимости тока короткого замыкания солнечной батареи от мощность излучения.
4. Анализ двух методов калибровки.
5. Выводы по работе.

6. СПИСОК ЛИТЕРАТУРЫ

1. Конспект лекций по курсу «Информационно-измерительные системы» (Юрченко А.В.)
2. Техническое описание к контроллеру.
3. Руководство к применению контроллера Элеси-ТМ.

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА №2

СТАНЦИЯ МОНИТОРИНГА РАБОТЫ СОЛНЕЧНЫХ БАТАРЕЙ НА ОСНОВЕ ПЛАТЫ СБОРА ДАННЫХ LA20USB

ВВЕДЕНИЕ

Лабораторно-практическая работа посвящена созданию станции мониторинга работы солнечных батарей на основе платы сбора данных LA20USB. Работа закрепляет на практике знания полученные в разделе курса лекции «Информационно-измерительные системы» раздел автоматизированные системы научных исследований.

В ходе работы студент с использованием системы программирования Labview разработает программу управления станцией и проведет исследования влияния климатических факторов на работу солнечной батареи.

Работа выполняется в 2 этапа. Первый этап посвящен изучению структуры станции, электронной нагрузки солнечных батарей и платы сбора данных. В рамках первого этапа студент подключает плату сбора данных и разрабатывает программу ручного измерения вольтамперной характеристики солнечной батареи. Второй этап посвящен созданию программы управления комплексов в автоматическом режиме и проведения анализа результатов работы станции и данных полученных из TOR-станции работающей в ИОА СО РАН.

1. ЦЕЛЬ РАБОТЫ

1.1. Изучить возможности платы сбора данных по организации научного эксперимента

1.2. Изучить структуру станции, электронной нагрузки солнечных батарей и платы сбора данных. Создать программу управления станцией на Labview позволяющей проводить ручное измерения вольтамперной характеристики солнечной батареи и рассчитывать ее основные характеристики.

1.3. Создание программы управления комплексом в автоматическом режиме и проведения анализа результатов работы станции и данных полученных из TOR-станции работающей в ИОА СО РАН.

2. ПРОГРАММА РАБОТЫ

2.1. Изучение структуры станции, принципа работы ее элементов.

2.2. Подключение инициализация плана сбора данных в среде Labview.

2.3. Разработка программы измерения вольтамперной характеристики солнечной батареи и произвести расчет КПД, тока КЗ, напряжения ХХ и определить рабочую точку.

2.4. Разработка алгоритма и программы управления комплексом в автоматическом режиме, обеспечивающий расчет основных характеристик солнечной батарей.

2.5. Сбор данных о работе солнечных батарей в 10корп. ТПУ и в ИОА СОРАН.

2.6. Анализа результатов работы станции и данных полученных из TOR-станции работающей в ИОА СО РАН и определением факторов влияющих на работу солнечных батарей.

2.7. Сделать выводы и подготовить отчет по работе.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ И СРЕДСТВА ИЗМЕРЕНИЯ

Объектами исследования являются кремниевые солнечные батареи, подключенные к электронной нагрузке, обеспечивающий измерение вольтамперной характеристики.

Для исследования в работе используются следующие приборы:

- плата сбора данных LA20USB.
  - солнечные батареи 240Вт и 25Вт.

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Станция мониторинга работы солнечных батарей состоит из трех основных частей:

1. Плата сбора данных LA20USB, обеспечивающая управление всеми элементами станции и сбор данных.

2. Электронная нагрузка, обеспечивающая измерение вольтамперной характеристики солнечной батареи посредством изменения своего сопротивления по команде с АЦП.

3. Кремниевая солнечная батарея мощностью 25 Вт, установленная на крыше 10корп.ТПУ.

Также, для анализа работы солнечных батареи используется TOR-станция мониторинга состояния атмосферы с контролем параметров 25 Вт солнечной батареи, размещенная в Институте оптики атмосферы СО РАН. Доступ к данным, полученных посредством станции по адресу www.meteo.iao.ru

4.1. Общая информация по плате сбора данных LA20USB

4.1.1. Устройство и работа прибора

Структурная схема взаимодействия составных частей прибора показана на рис. 1.

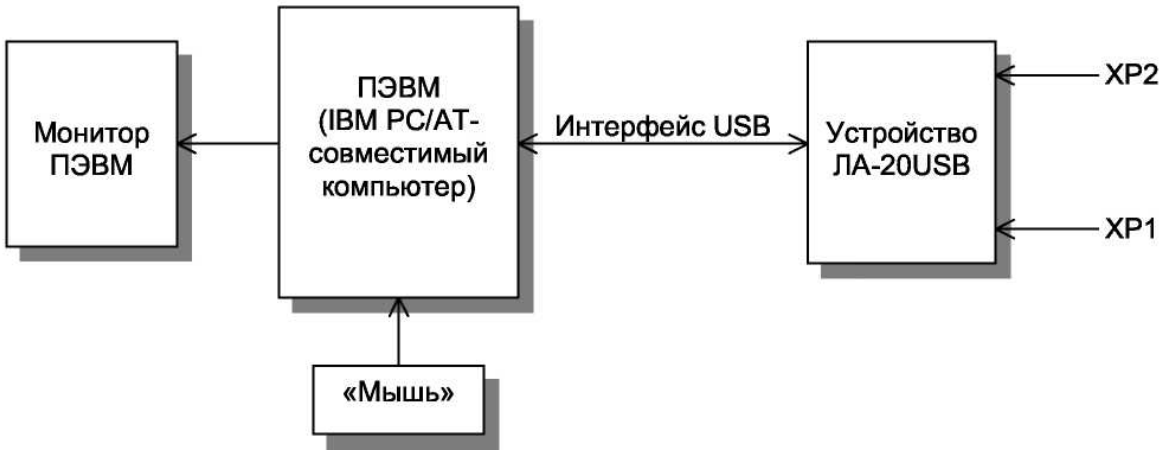


Рис. 1

Исследуемый аналоговый сигнал подается на входы каналов 0-31 устройства ЛА-20USB (более подробно о функциональной схеме ЛА-20USB см. в техническом описании на плату). Устройство ЛА-20USB осуществляет преобразование входного аналогового сигнала в цифровую форму, удобную последующей обработки ПЭВМ.

Обмен данными аналого-цифрового преобразования между ПЭВМ и устройством осуществляется через интерфейс USB ПЭВМ.

ПЭВМ при помощи специальной программы, входящей в комплект поставки (например ADCLab) или программы, разработанной самим пользователем в среде Labview, осуществляет обработку поступающих от устройства данных аналого-цифрового преобразования и управление устройством через интерфейс USB.

Функциональная схема устройства ЛА-20USB показана на рис. 2. Устройство содержит следующие независимые узлы: аналогово-цифровой канал (АЦК), опорный кварцевый генератор, цифровой порт ввода/вывода, внутренний интерфейс управления и конфигурации, интерфейс USB 2.0 и вторичный источник питания.

Основное назначение АЦК – преобразование исследуемого аналогового сигнала в цифровую форму. АЦК состоит из входного мультиплексора, полного инструментального усилителя, программируемого усилителя, АЦП с УВХ. Режим работы АЦК (однополюсный или дифференциальный) задаётся программно. С помощью программируемого

усилителя можно программно задать входной диапазон АЦП. При задании для работы одновременно нескольких каналов (группы каналов) – коэффициент усиления в каждом канале может быть задан раздельно на каждый канал.

Источник тактовой частоты АЦП может быть внешний ТТЛ-совместимый сигнал, импульсная последовательность отрицательной полярности; длительность импульсов не менее 100 нс (частота от 1 до 50 кГц), подаваемый на соответствующий контакт разъема XP2 или XP1. или внутренний – 1 МГц с 16 разрядным счетчиком-делителем ($2^{16} – 1 = 65535$), таким образом, минимальная задаваемая частота для платы может быть приблизительно 15,26 Гц. Внутренним источником тактовой частоты АЦП служит кварцевый генератор. Сетка задаваемых частот образуется так: максимальная частота 50 кГц, образуется как 1 МГц/20 = 50 кГц, следующая частота 1 МГц/21, еще следующая 1 МГц/22 и т. д. до 1 МГц/65535.

Рис. 2

Выбор режима работы аналогово-цифрового преобразователя – от кварцевого генератора или внешнего источника тактовой частоты, задается программно.

Цифровой порт ввода/вывода содержит 16 линий – 8 на ввод (порт PB) и 8 на вывод (порт PA). Линии ввода и вывода независимы. Ввод может стробироваться сигналом STR_DIO с соответствующего контакта разъёма XP1. Стробирование осуществляется уровнем логического нуля. Кроме режима программного ввода/вывода возможен ввод 4-х битов данных синхронно с данными АЦП. Это может использоваться в

задачах, где необходимо фиксировать состояние цифровых линий параллельно с оцифровкой аналоговых входов. Также в задачах синхронизации собираемых данных, при объединении нескольких устройств в единую систему

4.1.2. Интерфейсы управления и ввода вывода

Внутренний интерфейс управления и конфигурации представляет собой набор регистров и управляющей логики, необходимый для программного задания всех параметров работы преобразователя, таких, например, как:

- режим работы АЦК;
- число опрашиваемых каналов;
- режим работы цифрового порта ввода/вывода и другие параметры.

Интерфейс USB версии 2.0 осуществляет управление обменом данными между прибором и ПЭВМ.

Устройством можно управлять при помощи любого языка программирования, который имеет возможность работать с портами ввода/вывода ПЭВМ. Например: Basic, Visual Basic, C, C++ и другие. Более подробную информацию о характеристиках платы сбора данных можно получить в руководстве пользователя по плате сбора данных.

4.2. Структурная схема станции мониторинга и описание работы электронной нагрузки

На рис. 3 представлена структурная схема станции мониторинга работы солнечной батареи. Основным элементом, которой является электронная нагрузка (ЭН) которая изменяет свое сопротивление в зависимости от подаваемого на нее напряжения с ЦАП. Это позволяет измерять посредством АЦП напряжение на изменяющейся нагрузки солнечной батареи.

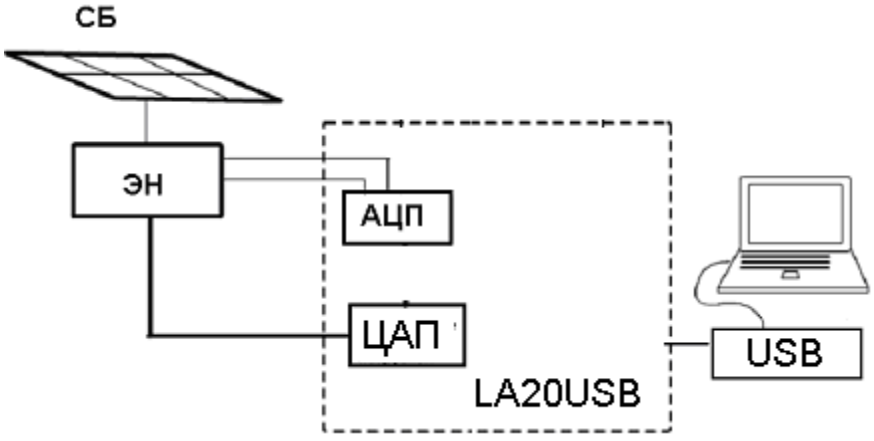


Рис. 3. Блок схема мобильной станции

Принцип действия электронной нагрузки основан на изменении сопротивления затвора полевого транзистора VT1. Принципиальная схема электронной нагрузки представлена на рис. 4. Управление затвором транзистора обеспечивается микросхемой DA1 через операционный усилитель.

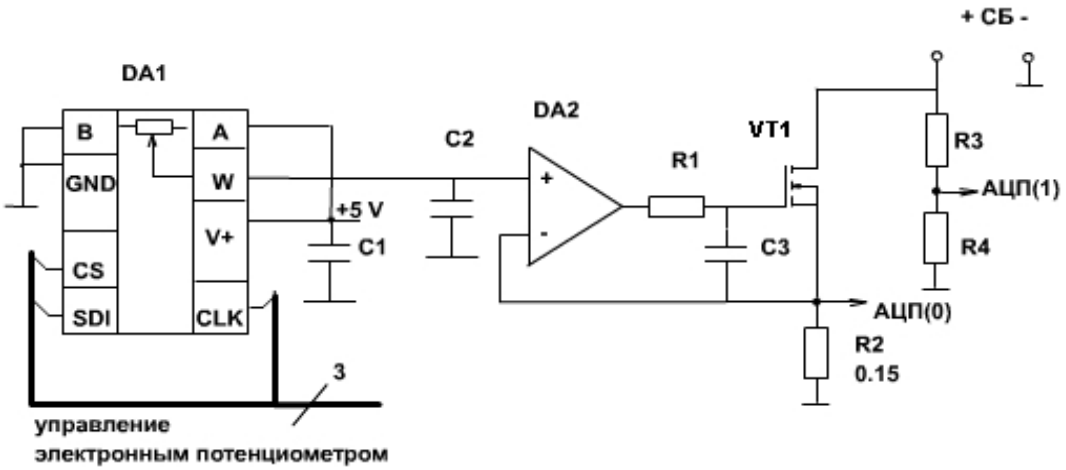


Рис. 4

4.3. Упражнение 1. Разработка программы измерения вольтамперной характеристики солнечной батареи в ручном режиме в среде Labview

Запустите пакет LabVIEW. Во время запуска в диалоговом окне LabVIEW щелкните мышью на опции **Новый ВП (New VI).** На экране появится лицевая панель с названием **Untitled1.**

Перейдите на окно блока диаграммы ВП. Если блок диаграмма не видна то выберите **Окно (Window) → показать блок диаграмму (show Block Diagram).**

На панели функций выберите **структуры (structures) →структура стековая последовательность (Stacked Sequence Structure).**

Переместите выбранную структуру на блок диаграмму.

Структура **Последовательность** используется для управления порядком ввыполнения узлов данных, которые не зависят друг от друга. Эта структура выглядит как набор кадров и обеспечивает последовательное выполнение размещенных в ее кадрах фрагментов программы. Сейчас структура имеет один кадр. Создадим еще один. Нажмите правой кнопкой мышки над верхним краем стуктуры и в выпавшем меню выберите добавить кадр после(add frame after) рис. 6.

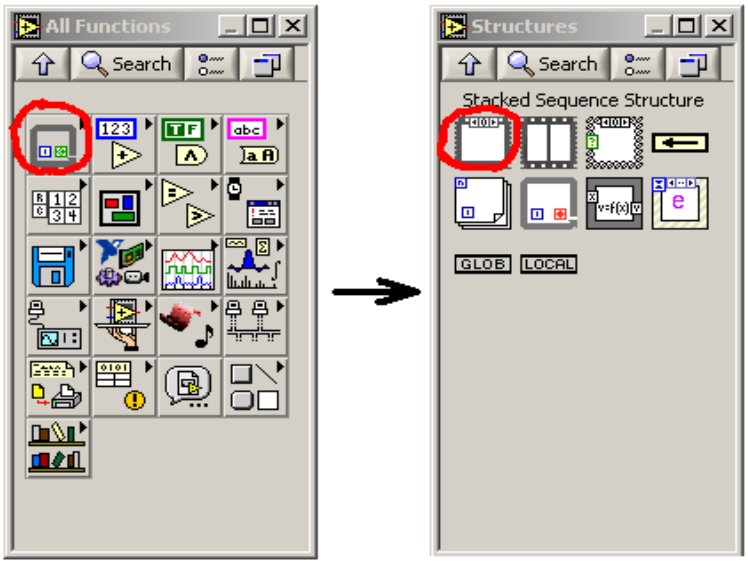


Рис. 5

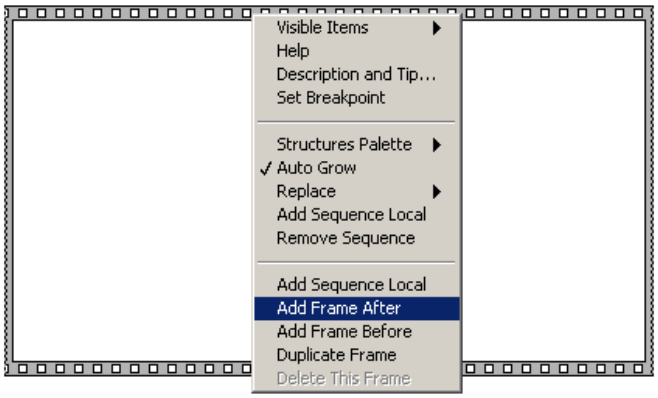


Рис. 6

В результате в верхней части структуры появится элемент управления кадрами (рис. 7).

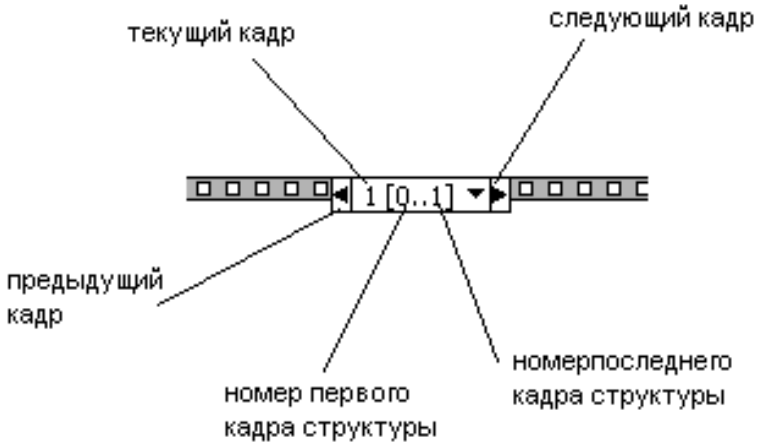


Рис. 7

После проделанных действий получилась структура состоящая из двух кадров.

При запуске программы сначала будет выполнено содержимое нулевого кадра затем первого и т. д. пока не закончатся кадры. Пусть основная программа будет располагаться в кадре с номером нуль.

Переключите текущий кадр на нулевой. Для этого на элементе управления кадрами структуры левой кнопкой мышки нажмите на соответствующий черный треугольник. Так чтобы в поле текущий кадр была цифра нуль.

Внутри нулевого кадра создадим структуру **цикл пока (While loop)** (рис. 8).

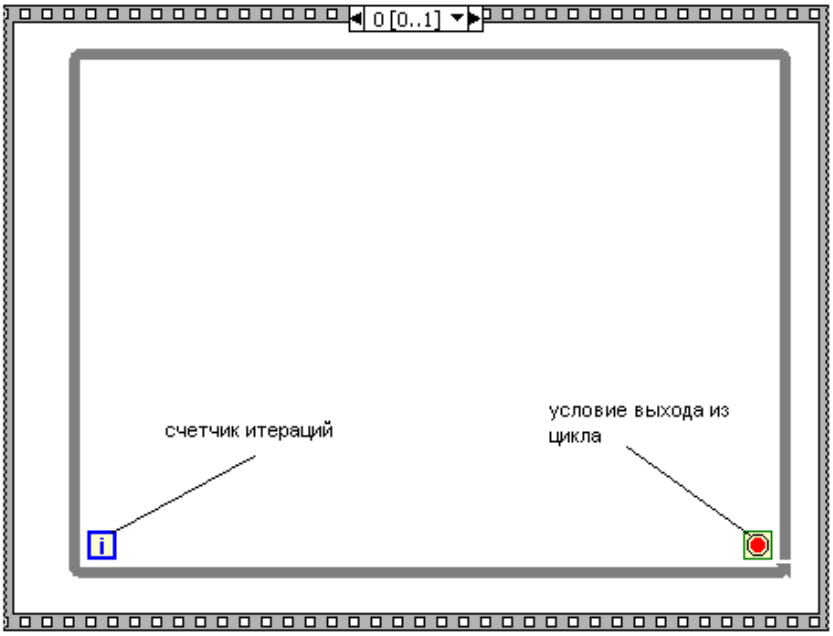


Рис. 8

Цикл пока будет выполняться до тех пор пока не будет выполнено условие выхода. В нашем случае условие выхода пока не определено. Счетчик итераций показывает, сколько раз уже было выполнено тело цикла. Под телом цикла следует понимать все то, что находится внутри цикла. Для правильного функционирования цикла необходимо задать условие выхода. В нашем случае это будет нажатие кнопки выход. Для этого создадим кнопку выход.

Перейдите в окно **передняя панель (Front panel).** Из **набора инструментов (controls)** выберите **логические (boolean)** затем **кнопка стоп (stop buttion)** (рис. 5) и переместите её в область передней панели.

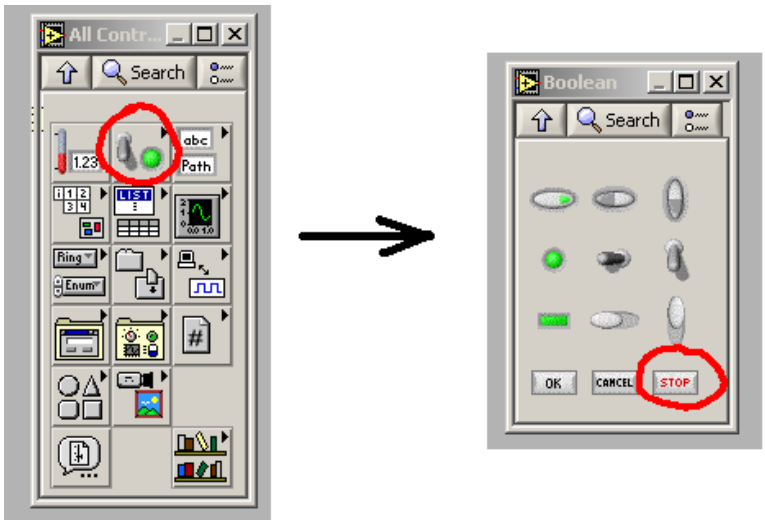


Рис. 9

В результате проделанных действий на передней панели ВП должна появиться кнопка stop при нажатии на которую программа должна заканчивать работу. Вообще любая кнопка это переменная логического типа которая принимает всего лишь два значения нажата – не нажата или TRUE и FALSE. Подключив кнопку к условию выхода из цикла WHILE получим что цикл будет выполняться до тех пор пока не будет нажата кнопка STOP. Практически каждому элементу на передней панели соответствует терминал на блоке диаграмме. В нашем случае создав кнопку на передней панели на блоке диаграмме автоматически появится терминал который соответствует только этой кнопке. терминал принимает значение кнопки на передней панели. Т. е. если кнопка не была нажата то соответствующий ей терминал имеет значение FALSE (ЛОЖЬ), а если нажата – TRUE (истина). Теперь необходимо чтобы созданная кнопка играла роль выхода из цикла. Для этого в окне блока диаграммы необходимо подключить ее к условию выхода из цикла. Перейдите в окно блока диаграммы и на панели инструментов (не панели функций!!!) (рис 10) выберите connect Wire.

Рис. 10

Если терминал кнопки находится за пределами тела цикла то перетащите его внутрь цикла и расположите рядом с условием выхода из цикла. Для перетаскивания терминала необходимо выбрать стрелочку на панели инструментов. Соедините терминал кнопки с условием выхода (рис. 11).

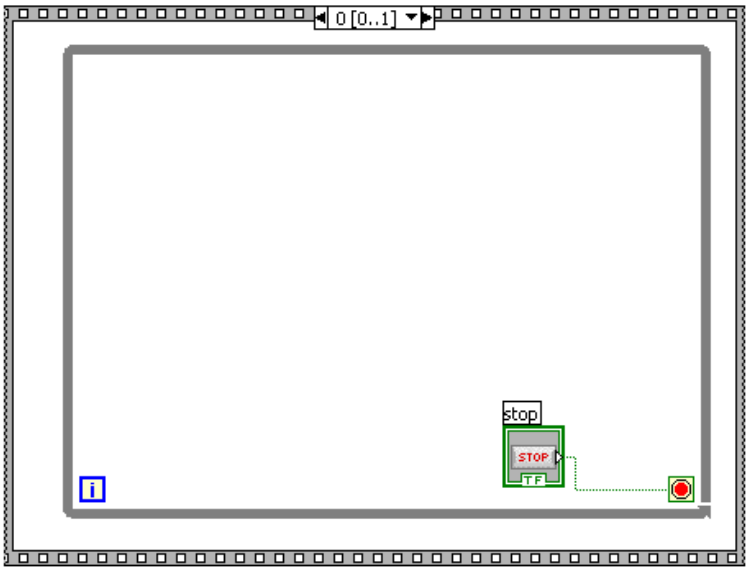


Рис. 11

Теперь запустим программу. Если все сделано правильно, то в верхней части передней панели появится белая стрелочка слева на право. Это кнопка запуска программы. Если стрелочка серого цвета и разорвана значит программа не может быть запущена т. к. были допущены ошибки. Перейдите на окно передней панели и запустите программу. Нажмите на кнопку стоп. После запуска программа переходит в нулевой кадр последовательной структуры в которой находится цикл WHILE. Начинает работать цикл. Цикл будет работать до тех пор пока не будет выполнено условие выхода, т. е. пока не будет нажата кнопка стоп. После нажатия кнопки работа цикла прекращается и программа переходит в следующий кадр. Так как он пока пустой работа программы завершается. Теперь необходимо инициализировать плату сбора данных к LabVIEW. Для этого воспользуемся стандартной библиотекой поставляемой вместе с платой. Перейдите на блок диаграмму и из панели функций выберите **user libraries → init_la20**. Поместите этот виртуальный подприбор (ВПП) на блок диаграмму так чтобы он был за пределами последовательной структуры. Это связано с тем, что инициализация платы должна производиться один раз в начале программы. Перейдите в следующий кадр и поместите внутри него ВПП закрытия драйвера Close. Соедините выход **Device out** ВПП

init_la20 со входом **Close.** Перейдите на нулевой кадр и в тело цикла WHILE поместите ВПП GET_volts. Соедините выход **Device out** ВПП **init_la20** со входом **Device inp** ВПП GET_volts. Над выходом volts ВПП GET_volts нажмите правую кнопку мышки и выберете пункт **create → indicator.** В результате на блоке диаграмме появится терминал массива а на передней панели соответствующий ему индикатор. В этом массиве будут хранится результаты работы ВПП GET_volts. Т. е. значения напряжений на каналах платы сбора данных.

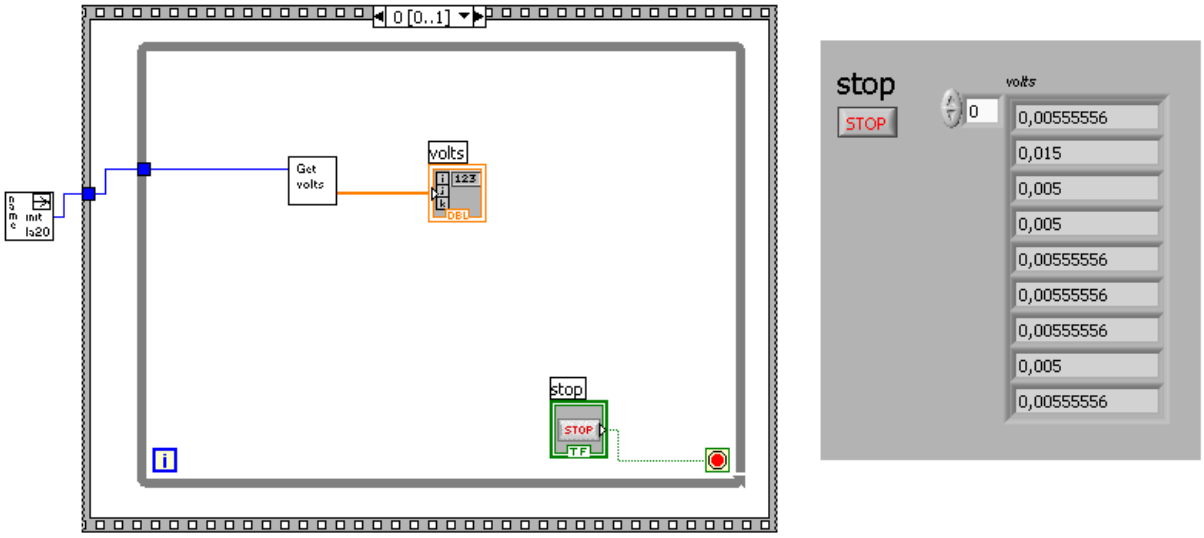


Рис. 12

Теперь необходимо организовать управление электронной нагрузкой. Для этого выберите из панели функций **user libraries → set load.vi** . Соедините выход **Device out** ВПП **Get_volts** со входом **Device inp** ВПП **set load.** Создайте орган управления нагрузкой. Для этого над другим входом ВПП set LOAD нажмите правую кнопку мыши и выберите **create → control.** На диаграмме появится терминал регулятора нагрузки а на передней панели сам регулятор (рис. 13). Теперь крутя виртуальный регулятор на вход электронного потенциометра будет подаваться соответствующий цифровой код а значит меняться сопротивление между выходами микросхемы. Изменяя сопротивление электронного потенциометра меняется напряжение на резисторе R2 а значит и ток через нагрузку (транзистор VT1).

Как было сказано выше в массиве volts хранятся значения напряжений на каналах АЦП. К нулевому каналу АЦП подключено напряжение солнечной батареи, а к первому – напряжение которое характеризует ее ток. Поэтому необходимо выделить из массива нулевой и первый элемент. Для работы с массивами на панели функций есть специальный набор **array**. Из этого набора необходимо выбрать функцию i**ndex array**

которая выбирает указанный элемент из массива. Поместите на блок диаграмму эту функцию рядом с массивом **volts**. Соедините массив volts со входом функции выбора элемента. Наведите мышку на эту функцию. Сверху и снизу функции должны появится черные квадратики. Наведите мышку на нижний квадратик и не отпуская левую кнопку растяните терминал функции на одну позицию вниз. Подсоедините ко входам index функции выбора элемента массива цифры 0 и 1. Т. е. на первом выходе этой функции будет нулевой элемент массива а на втором первый. Создайте для этих элементов индикаторы (рис. 13).

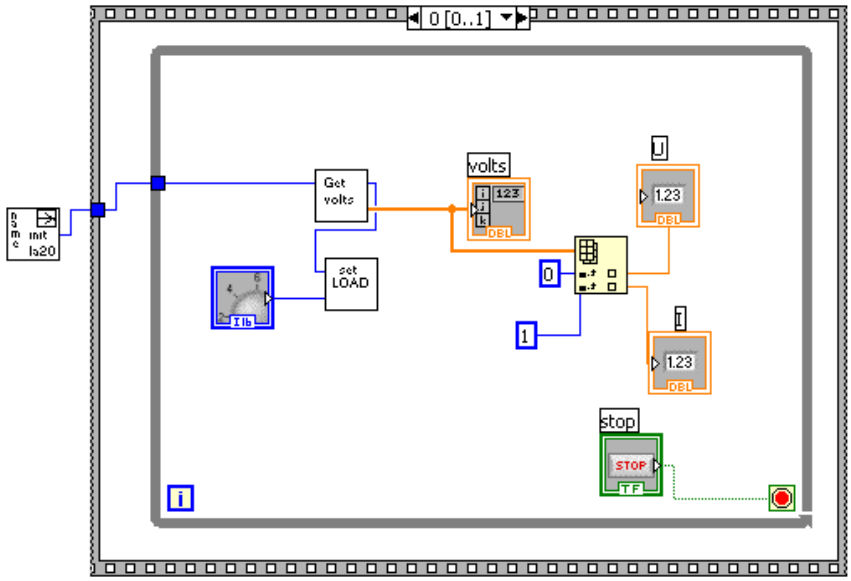


Рис. 12

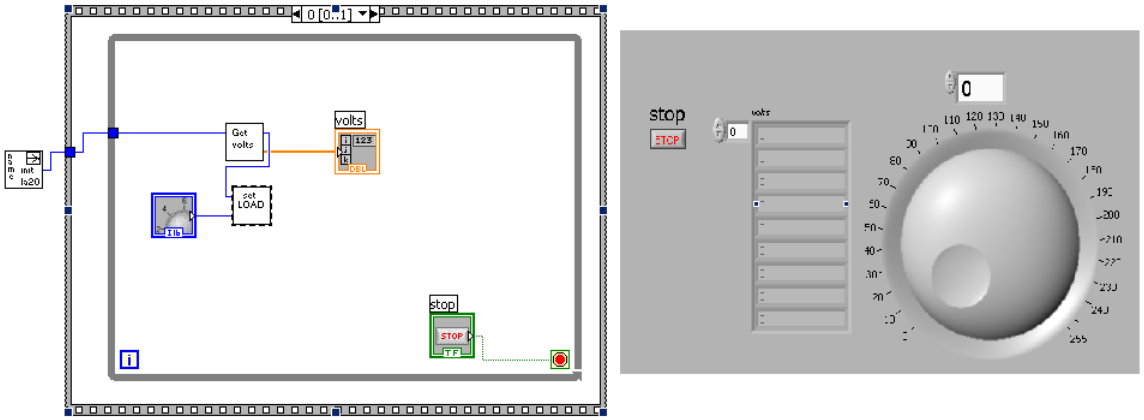


Рис. 13

Запустите программу. Изменяя электронную нагрузку следите за изменением тока и напряжения СБ. Измерите вольтамперную характеристику солнечной батареи. И определите ее КПД, ток короткого замыкания, напряжение холостого хода и рабочую точку.

4.4. Упражнение 2. Создание программы управления комплексом в автоматическом режиме и проведения анализа результатов работы станции и данных полученных из TOR-станции работающей в ИОА СО РАН

1. Используя навыки полученные при выполнении 1 упражнения создайте программу позволяющая ежеминутно измерять вольтамперную характеристику солнечной батареи и определять ее основные характеристики.

2. Обеспечьте сохранение результатов измерения в файле в текстовом формате в виде следующей таблицы:

Время и дата измерений	**КПД**	**Iкз**	**Vxx**	**Ip**	**Vp**

3. Используя интернет-ресурс TOR-станции (www.meteo.iao.ru) получите данные о состоянии атмосферы на данный момент. Также получите данные о результатах испытания солнечной батареи в течении последнего месяца.

4. Проведите сравнение результатов полученных вами и результатов полученных на TOR-станции. Постройте зависимости основных характеристик солнечной батареи от основных климатических параметров, измеряемых станцией:

№	Описание	Обозначение	Единица измерения
1	Дата	DATE	–
2	Время	TIME	час
3	Напряжение на нагрузке СБ	SUN	В
4	Солнечная радиация	SR	Ватт/$м^2$
5	Давление	PRESS	мм.рт.ст
6	Температура	TEMP	C^o
7	Влажность	HUMID	%
8	Скорость ветра	WINDS	м/сек
9	Макс. скорость ветра	MAXWINDS	м/сек
10	Направление ветра	WINDD	м/сек
11	Концентрация CO	CO	мгр/ $м^3$
12	Концентрация CO_2	CO2	мгр/ $м^3$
13	Концентрация O_3	O3	мгр/ $м^3$
14	Концентрация NO_2	NO2	мгр/ $м^3$
15	Концентрация NO	NO	мгр/ $м^3$
16	Радиационный фон	RF	Микроренген
17	Сумма всех аэрозолей	AEROS	частиц/ $м^3$

5. Проведите анализ эффективности использования солнечной энергии в г. Томске и определите факторы влияющие на работы солнечных батарей в натурных условиях г. Томска.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Название и цель работы
2. Структурная схема измерения
3. Таблица экспериментальных данных. Вольтамперная характеристика солнечной батареи, измеренная в ручном режиме. Расчет основных характеристик солнечной батареи
4. Графики зависимости основных характеристик солнечных батарей от климатических параметров.
5. Анализ эффективности применения солнечный батарей в г.Томске и описание фаеторов влияющий на ее работу.
6. Выводы по работе.

6. СПИСОК ЛИТЕРАТУРЫ

1. Конспект лекций по курсу «Информационно-измерительные системы» (Юрченко А.В.)
2. Руководство к применению контроллера платы сбора данных LA20USB.

Приложение 1

ПРИНЦИП ДЕЙСТВИЯ СОЛНЕЧНЫХ БАТАРЕЙ И ИХ ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Солнечные батареи представляют собой последовательно и параллельно соединенные солнечные элементы, помещенные в защитный каркас. Солнечные элементы представляют собой p-n переход, который получен путем диффузии и представляет собой структуру (рис. 1).

Принцип действия солнечных элементов (СЭ): Оптическое излучение, падающее на поверхность такой структуры создает в объеме полупроводника электронно-дырочные пары, причем скорость генерации пар убывает по экспоненциальному закону от поверхности в глубь полупроводника. Когда на СЭ падает солнечный свет, фотоны с энергией меньшей ширины запрещенной зоны E_g не дают вклада в выходную мощность прибора (в пренебрежении примесным поглощением света). Каждый фотон с энергией большей E_g, дает вклад пропорциональный E_g в выходную мощность СЭ, а остальная часть энергии фотона ($h\gamma$-E_g) рассеивается в виде тепла (фононов)

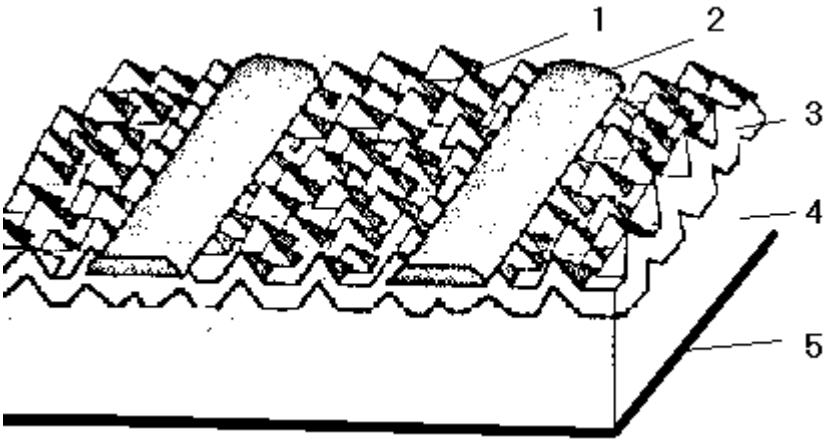


Рис. 1. Конструкция промышленно изготовляемых СЭ
1 – текстурированная поверхность с просветляющим покрытием SiO_2,
2 – лицевой омический контакт, 3 – n^+-Si, 4 – подложка p-Si,
5 – тыловой омический контакт

Образовавшиеся, за счет поглощения света, носители, участвуя в тепловом движении, перемещаются в разных направлениях, в том числе, и по направлению к p-n переходу (рис. 2).

Неосновные, сгенерированные оптическим излучением и подошедшие к p-n переходу, носители заряда будут разделяться полем перехода. Для основных носителей переход представляет собой потенциальный барьер. Если поглощение кванта света (фотона) произошло в

n-области на расстоянии меньшем, чем диффузионная длина носителей заряда от p-n перехода, то неосновные носители заряда могут дойти до перехода и разделится полем перехода.

Аналогично, если фотон поглотился в p-области, то неосновные носители заряда, находящиеся на расстоянии диффузионной длины от p-n перехода, переносятся в n-область. Если поглощение произошло в области пространственного заряда, то электроны и дырки уходят из нее в n и p области соответственно (рис. 2). Таким образом, потенциальный барьер играет роль разделителя носителей заряда. Процесс разделения носителей заряда приводит к накоплению основных носителей заряда в n и p областях. Избыточные дырки заряжают p-область положительно, а избыточные электроны заряжают n-область отрицательно. Между n- и p-областями возникает фотоЭДС с полярностью противоположной барьерному полю p-n перехода.

Появление фотоЭДС приводит к уменьшению барьерного поля p-n перехода. Это приводит к уменьшению потока электронов из p-области и дырок из n-области. Когда фотоЭДС уменьшит потенциальный барьер до величины порядка kT, дальнейшее увеличение фотоЭДС прекращается. Отсюда следует, что фотоЭДС не может превышать контактную разность потенциалов. Уменьшение контактной разности потенциалов обнаруживается во внешней цепи, как появление напряжения на выходных контактах СЭ.

Если цепь СЭ разомкнута (сопротивление нагрузки равно бесконечности), то все, разделенные p-n переходом, носители заряда скапливаются у p-n перехода и компенсируют потенциальный барьер, создавая фотоЭДС равное напряжению холостого хода V_{xx}.

Если СЭ замкнут накоротко (сопротивление нагрузки равно нулю), то избыточные, разделенные p-n переходом, сгенерированные носители заряда будут иметь возможность циркулировать через эту короткозамкнутую цепь, создавая максимально возможное значение тока – ток короткого замыкания $I_{кз}$. В этом случае никакого скопления избыточного заряда у p-n перехода не возникает. Потенциальный барьер будет иметь ту же высоту, что и при отсутствии оптического излучения. ФотоЭДС будет равна нулю.

Если фотоэлемент замкнут на конечное сопротивление $R_н$, то часть, разделенных переходом, носителей заряда затратят свою энергию на снижение потенциального барьера, т. е. создадут напряжение нагрузки $V_н$, а оставшаяся часть носителей создаст ток в нагрузке $I_н$.

Вольтамперная характеристика СЭ при отсутствии освещения представляет собой ВАХ p-n перехода:

$$I = I_S \cdot (e^{\frac{qV}{kT}} - 1)$$

где I_S – ток насыщения, создаваемый свободными носителями, сгенерированными за счет теплового возбуждения.

При освещении p-n перехода через него помимо темнового тока будет протекать ток, обусловленный генерацией носителей заряда оптическим излучением $I_{кз}$.

Таким образом, можно записать вольтамперная характеристика СЭ, как сумму токов (для идеального случая):

$$I = I_S \cdot (e^{\frac{qV}{kT}} - 1) - I_{кз}$$

Графически зависимость тока от напряжения для СЭ имеет вид: (рис. 3).

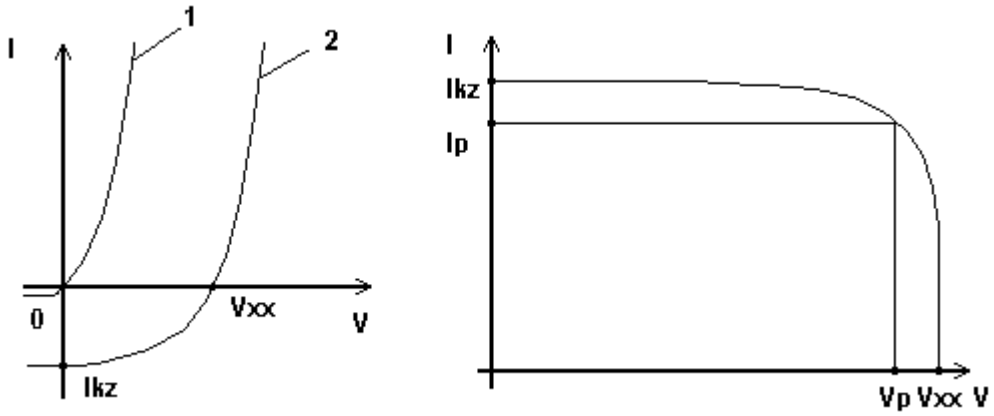


Рис. 3

Поскольку вольтамперная характеристика СЭ проходит через четвертый квадрант, то это означает, что прибор, описываемый такой вольтамперной характеристикой, является источником энергии.

Выражение для напряжения холостого хода:

$$V_{xx} = \frac{kT}{q} \cdot \ln\left(\frac{I_{кз}}{I_S} + 1\right)$$

Для нахождения тока короткого замыкания необходимо решать уравнения переноса с учетом положения перехода относительно лицевого или тылового омического контакта и функции генерации носителей заряда, в которую входят: спектр падающего излучения (спектр солнца), коэффициент поглощения оптического излучения полупроводником, коэффициент отражения от поверхности полупроводниковой структуры

Важнейшей характеристикой СЭ является коэффициент полезного действия. Он определяется, как отношение максимальной мощности, отдаваемой в нагрузку СЭ к мощности солнечного излучения, падающего перпендикулярно рабочей поверхности.

$$\eta = \frac{V_P \cdot I_P}{P_C} \cdot 100\,\%,$$

где: V_p и I_p – напряжение и ток в рабочей точке ФЭП (при которых достигается максимальная мощность, отдаваемая в нагрузку рис. 4); P_c – мощность падающего на СЭ излучения.

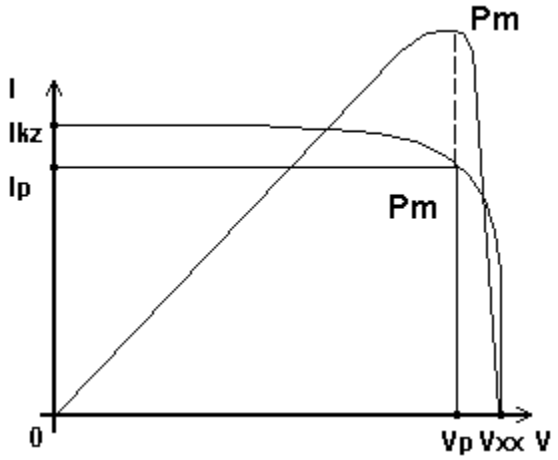


Рис. 4. ВАХ СЭ и зависимость выделяемой мощности на нагрузке от напряжения

СОДЕРЖАНИЕ

Учебное издание

ЮРЧЕНКО Алексей Васильевич

ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ

Методические указания к выполнению лабораторных работ
по курсу «Информационно-измерительные системы»
для студентов IV курса, обучающихся по направлению
200100 «Приборостроение»

Научный редактор
доктор технических наук,
профессор — *А.Е. Гольдштейн*

Верстка — *В.П. Аршинова*

Дизайн обложки — *О.Ю. Аршинова*
О.А. Дмитриев

Подписано к печати 17.11.2008. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. 3,26. Уч.-изд. л. 2,94.
Заказ 858. Тираж 100 экз.

Томский политехнический университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2000

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.