##### Содержание

[Содержание 4](#_Toc484890732)

[Введение 6](#_Toc484890733)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 9](#_Toc484890734)

[1.1. Общие сведения 9](#_Toc484890735)

[1.2. Назначение и цели создания системы 11](#_Toc484890736)

[1.3. Характеристики объекта автоматизации 11](#_Toc484890737)

[1.4. Требования к системе 12](#_Toc484890738)

[1.5. Порядок контроля и приемки программного обеспечения 16](#_Toc484890739)

[1.6. Требования к документации 16](#_Toc484890740)

[2. ОПИСАНИЕ СРЕДСТВ АВТОМАТИЗАЦИИ 18](#_Toc484890741)

[2.1. Simatic S7-300 18](#_Toc484890742)

[2.2. Simatic Step 7 20](#_Toc484890743)

[2.3. Языки программирования контроллеров 21](#_Toc484890744)

[2.4. Загрузка программы в контроллер 23](#_Toc484890745)

[2.5. S7-PLCSIM 25](#_Toc484890746)

[2.6. Конфигурация оборудования 25](#_Toc484890747)

[3. ОПИСАНИЕ ПРОГРАММЫ 27](#_Toc484890748)

[3.1. Общие сведения 27](#_Toc484890749)

[3.2. Функциональное назначение 27](#_Toc484890750)

[3.3. Описание логической структуры 27](#_Toc484890751)

[3.4. Используемые технические средства 36](#_Toc484890752)

[3.5. Вызов и загрузка 37](#_Toc484890753)

[3.6. Входные данные 37](#_Toc484890754)

[3.7. Выходные данные 37](#_Toc484890755)

[4. ТЕКСТ ПРОГРАММЫ 38](#_Toc484890756)

[4.1. Главный модуль программы 38](#_Toc484890757)

[4.2. Подсистема управления секциями вентиляторов 39](#_Toc484890758)

[4.3. Подсистема управления насосами 50](#_Toc484890759)

[5. РУКОВОДСТВО ПРОГРАММИСТА 58](#_Toc484890760)

[5.1. Назначение и условия применения программы 58](#_Toc484890761)

[5.2. Характеристика программы 58](#_Toc484890762)

[5.3. Обращение к программе 60](#_Toc484890763)

[5.4. Сообщения 60](#_Toc484890764)

[6. РУКОВОДСТВО ОПЕРАТОРА 61](#_Toc484890765)

[6.1. Назначение программы 61](#_Toc484890766)

[6.2. Условия выполнения программы 61](#_Toc484890767)

[6.3. Выполнение программы 62](#_Toc484890768)

[6.4. Сообщения оператору 62](#_Toc484890769)

[7. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 63](#_Toc484890770)

[7.1. Объект испытаний 63](#_Toc484890771)

[7.2. Цель испытаний 63](#_Toc484890772)

[7.3. Требования к программе 63](#_Toc484890773)

[7.4. Требования к программной документации 64](#_Toc484890774)

[7.5. Средства и порядок испытаний 64](#_Toc484890775)

[7.6. Методы испытаний 65](#_Toc484890776)

[Заключение 67](#_Toc484890777)

[Список использованных источников 68](#_Toc484890778)

##### Введение

В середине 90-х гг. прошлого века микропроцессорная техника достигла состояния надежности, обеспечивающего возможность управления технологическим процессом с помощью контроллеров. Препятствиями на пути более активного применения микропроцессорных средств стали: человеческий фактор (привычка пользоваться индивидуальными приборами и ключами), нехватка финансирования для ввода новых энергоблоков в период с 1995 г., а также неприспособленность технологического оборудования к автоматизации.

Построение автоматизированных систем управления технологическим процессом (АСУ ТП) на базе микропроцессорной техники позволило значительно повысить качество систем управления, открыло новые перспективы автоматизации. Под АСУ ТП обычно понимается целостное решение, обеспечивающее автоматизацию основных операций технологического процесса на производстве в целом или каком-то его участке, выпускающем относительно завершённое изделие.

Понятие «автоматизированный», в отличие от понятия «автоматический», подчёркивает необходимость участия человека в отдельных операциях, как в целях сохранения контроля над процессом, так и в связи со сложностью или нецелесообразностью автоматизации отдельных операций.

Темой выпускной квалификационной работой является – «Программная автоматизация управления системой охлаждения производственного объекта».

**Объект –** программное обеспечение.

**Предмет** – программное средство для управления системой охлаждения производственного объекта.

**Цель** – реализация программного обеспечения автоматизации управления системой охлаждения производственного объекта.

Система охлаждения представляет собой аппараты воздушного охлаждения («сухие градирни») – 4 шт., каждый аппарат имеет 3 секции, в которых установлены по 4 вентиляторных агрегата (всего 48 шт.). Управление происходит посекционно. Система в автоматическом режиме контролирует температуру теплоносителя (водный раствор), который поступает для охлаждения, включая или выключая секции по заданному алгоритму работы. Подача теплоносителя осуществляется с помощью насосов, которые удерживают определенное давление.

Входными данными являются показания с датчиков – входные сигналы: температуры, давления, текущие состояния секций и насосов. На основе них алгоритм программы предпринимает действие для управления объектом автоматизации. Действия являются выходными данными – выходными сигналами: включение или выключение секции, насоса.

Реализация алгоритма автоматизации системы управления охлаждения происходит с помощью программируемого логического контроллера (ПЛК) Siemens SIMATIC S7-300, который включает в себя сигнальные и функциональные модули для работы с входными и выходными сигналами.

Для конфигурации ПЛК используется проект в среде разработки Siemens Simatic Step 7, включающий в себя конфигурацию необходимого оборудования и реализацию алгоритма на понятных для контроллера языках программирования.

**Задачи**:

1. изучить предметную область;
2. изучить основные принципы устройства оборудования автоматизации;
3. изучить инструмент разработки (Siemens Simatic Step 7);
4. разработать алгоритм.

Задачей программного обеспечения является улучшение качества автоматизации системы управления объекта, уменьшение количества ошибок при работе на объекте, повышение качества труда, уменьшение затрат на обслуживание.

Пояснительная записка выпускной квалификационной работы состоит из 6 документов. Коды документов приведены ниже в таблице 1.

Таблица 1

|  |  |  |
| --- | --- | --- |
| Обозначение | Наименование | Примечание |
| 4217.02067988.13 – 558 12 | Текст программы |  |
| 4217.02067988.10 – 558 13 | Описание программы |  |
| 4217.02067988.10 – 558 33 | Руководство программиста |  |
| 4217.02067988.10 – 558 34 | Руководство оператора |  |
| 4217.02067988.10 – 558 51 | Программа и методика испытаний |  |
| 4217.02067988.10 – 558 90 | Техническое задание |  |
| 4217.02067988.13 – 558 91 | Описание средств автоматизации |  |

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## Общие сведения

### Полное наименование программного обеспечения и его условное обозначение

Полное наименование – «Программная автоматизация управления системой охлаждения производственного объекта».

Условное обозначение – «Программа автоматизации».

### Шифр темы

Шифр темы – выпускная квалификационная работа.

Код работы – 4217.02067988.13 – 558.

### Наименование предприятий разработчика и заказчика системы

Разработчик – студент группы 3ВТб-1 ФГБОУ ВО «КнАГТУ» Р.В. Смирнов.

Заказчик – ФГБОУ ВО «КнАГТУ», кафедра МОП ЭВМ, в лице заведующего кафедрой МОП ЭВМ, кандидата технических наук, профессора В.А. Тихомирова.

### Перечень документов, на основании которых создается программное обеспечение

Перечень документов, на основании которых создается программное обеспечение:

* задание на выпускную квалификационную работу;
* приказ на закрепление темы № 1578-Д от 23.11.2016 г.;
* техническое задание;
* календарный план-график выполнения этапов работы (см. п. 1.5 данного технического задания).

### Плановые сроки начала и окончания работы по созданию программы

Плановый срок начала работ: 08.05.2017 г.

Плановый срок окончания работ: 25.06.2017 г.

### Сведения об источниках и порядке финансирования работ

Данная работа является инициативной, финансирование не проводится.

### Порядок оформления и предъявления заказчику результатов работы по созданию программного обеспечения

Оформление и предъявление заказчику результатов работ по созданию программного обеспечения «Программная автоматизация» производится в соответствии с календарным планом-графиком выполнения работ (см. п. 1.5 данного технического задания).

Порядок взаимодействия разработчика с заказчиком:

* создание рабочей программы;
* предъявление программного обеспечения заказчику;
* выявление заказчиком недостатков и недоработок в рамках настоящего технического задания;
* устранение выявленных недостатков и недоработок;
* создание лазерного диска с программным обеспечением;
* создание пакета технической документации на программное обеспечение, оформленного в соответствии с ГОСТ 19.101 и ЕСПД.

## Назначение и цели создания системы

### Назначение системы

Данное программное обеспечение предназначено для автоматизации управления системой охлаждения производственного объекта.

### Цели создания системы

Цель создания программного обеспечения – автоматизация управления производственным объектом.

## Характеристики объекта автоматизации

### Краткие сведения об объекте автоматизации

Объектом автоматизации является система управления охлаждением производственного объекта.

Результатом работы является программное обеспечение для программируемого логического контроллера Siemens Simatic S7-300.

### Сведения об условиях эксплуатации объекта автоматизации и характеристик окружающей среды

Условия эксплуатации программного обеспечения «Программная автоматизация управления системой охлаждения производственного объекта» определяются условиями эксплуатации использующегося аппаратного обеспечения. Программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» не имеет ограничения по времени эксплуатации.

Физические условия эксплуатации и характеристики окружающей среды определяются соответствующими характеристиками окружающей среды используемого аппаратного обеспечения, указанными в документации на него.

## Требования к системе

### Требование к системе в целом

#### Требования к структуре и функционированию системе

Система должна состоять из подсистемы управления насосами и подсистемы управления секциями, а также включая общую подсистему, которая управляет ими.

#### Требования к численности и квалификации персонала

С программным обеспечением «Программная автоматизация управления системой охлаждения производственного объекта» должен работать квалифицированный работник, обладающий базовыми навыками программирования логических контроллеров, и ознакомленный с предметной областью применения программного обеспечения.

#### Требования к надежности

Программное обеспечение системы должно функционировать и не приводить к зависанию системы при любых ошибочных действиях операторов в среде данного программного обеспечения.

#### Требования к эргономике и технической эстетике

Программное обеспечение и технические средства должны иметь эстетически привлекательный вид и быть удобными в использовании.

#### Требования к эксплуатации и техническому обслуживанию

Условия эксплуатации программного обеспечения «Программная автоматизация управления системой охлаждения производственного объекта» определяются условиями эксплуатации использующегося аппаратного обеспечения. Программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» не имеет ограничения по времени эксплуатации.

Характеристики окружающей среды определяются соответствующими характеристиками окружающей среды для выбранного типа носителя данных и используемого компьютерного оборудования.

Требования к допустимым площадям размещения ПЭВМ с установленным программным обеспечением «Программная автоматизация управления системой охлаждения производственного объекта», к параметрам сети энергоснабжения, уровню звукоизоляции, используемым в помещении отделочным материалам и т.п., определяются в соответствии с СанПиНом 2.2.2.542-96, с учетом требований использующегося аппаратного обеспечения.

Требования к эксплуатации и техническому обслуживанию технических средств и дополнительного программного обеспечения определяются в соответствии с предъявляемыми к ним соответствующими требованиями.

#### Требования к защите информации от несанкционированного

#### доступа

Защита системы и информации, с которой она работает, от несанкционированного доступа возлагается частично на операционную систему и специализированные программные и программно-аппаратные продукты, частично на систему аутентификации, реализованную в компонентах системы. Доступ к данным предоставляется только легальным пользователям.

#### Требования по сохранности информации при авариях

Сохранность данных должна быть обеспеченна:

* при сбоях в работе оборудования сети в любой момент времени;
* при сбоях электропитания или программного обеспечения не ведущих к физическому уничтожению или повреждению информации.

#### Требования к защите от влияния внешних воздействий

Автоматизированная система должна быть устойчивой к наличию и параллельной с ней работе на той же ЭВМ другого программного обеспечения.

#### Требования к стандартизации и унификации

Автоматизированная система должна предоставлять пользователю привычный, общепринятый в операционной системе Microsoft Windows интерфейс. Программная документация, поставляемая с программой, должна быть оформлена в соответствии со стандартом ЕСПД.

### Требования к функциям, выполняемым системой

Функции подсистемы графического интерфейса:

* отображение на экране информационных слов в шестнадцатеричном, десятичном и двоичном виде;
* отображение наличия обмена по заданным адресам.

Функции подсистемы управления секциями:

* опрос секций вентиляторов;
* анализ состояний секций;
* управление секциями на основании данных.

Функции подсистемы управления насосами:

* опрос насосов;
* анализ состояний насосов;
* управление насосами на основании данных.

Функции подсистемы общего назначения:

* хранение информации об объекте автоматизации;
* организация управления другими подсистемами.

### Требования к видам обеспечения

#### Требования к математическому обеспечению

В основе разрабатываемого ПО должна лежать модульное программирование. Организация программы выполняется как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

#### Требования к информационному обеспечению

Для разработки данного программного обеспечения рекомендуется следующее информационное обеспечение, информацию о котором можно посмотреть в разделе «Список использованных источников»:

* программирование с использованием среды разработки Siemens Simatic Step 7 [2, 3];
* разработка программного обеспечения для ПЛК с использованием промышленных языков программирования [1, 4].

#### Требования к лингвистическому обеспечению

В качестве языков программирования используются – LAD, SCL, STL. Взаимодействие пользователя и ПО осуществляется через среду разработки Siemens Simatic Step 7.

#### Требования к программному обеспечению

Для разработки данного программного обеспечения рекомендуется следующее программное обеспечение:

* операционная система Windows XP версии SP2 или SP3;
* среда разработки Siemens Simatic Step 7.

#### Требования к техническому обеспечению

Для использования программного обеспечения рекомендуется:

* оперативная память 512 Мб и выше;
* процессор Intel Pentium IV 3.200 МГц и выше;
* существование логических и/или физических дисков со свободным дисковым пространством 2048 Мб и выше;
* клавиатура
* манипулятор «мышь».

## Порядок контроля и приемки программного обеспечения

Контроль системы и ее приемка осуществляются в следующей последовательности:

1. тестирование системы по следующим параметрам:

* выдача корректных результатов при любом наборе входных данных;
* стабильность работы системы при выполнении любых доступных функций на каждом шаге ее работы;

1. исправление недостатков и ошибок системы, выявленных в результате тестирования;
2. повторное тестирование и исправление ошибок;
3. при устранении всех выявленных ошибок сдача системы в эксплуатацию.

## Требования к документации

Пояснительная записка оформляется в соответствии с ГОСТ 19.106-78 от 01.01.80 «Требования к программным документам, выполненным печатным способом», ГОСТ 19.101-77 от 01.01.80 «Виды программ и программных документов», ГОСТ 2.304-81 от 01.01.82 «Шрифты чертежные», ГОСТ 2.301-68 от 01.01.71 «Форматы ЕСКД», ЕСПД ГОСТ 19.001-77 от 1.01.80.

# ОПИСАНИЕ СРЕДСТВ АВТОМАТИЗАЦИИ

## Simatic S7-300

Для автоматизации промышленных объектов используются программируемые логические контроллеры.

Программируемый логический контроллер – это микропроцессорное устройство, предназначенное для управления технологическими процессами в промышленности. Принцип работы ПЛК заключается в обработке по прикладной программе пользователя данных с модулей входов (например, сигналов от подключенных датчиков) и последующей выдачей управляющих сигналов, посредством модулей выходов и модулей связи, обеспечивающих подключение исполнительных устройств.

Для реализации программного обеспечения используется ПЛК Siemens SIMATIC S7-300 (рисунок 2.1). Это модульный программируемый контроллер, предназначенный для построения систем автоматизации низкой и средней степени сложности.

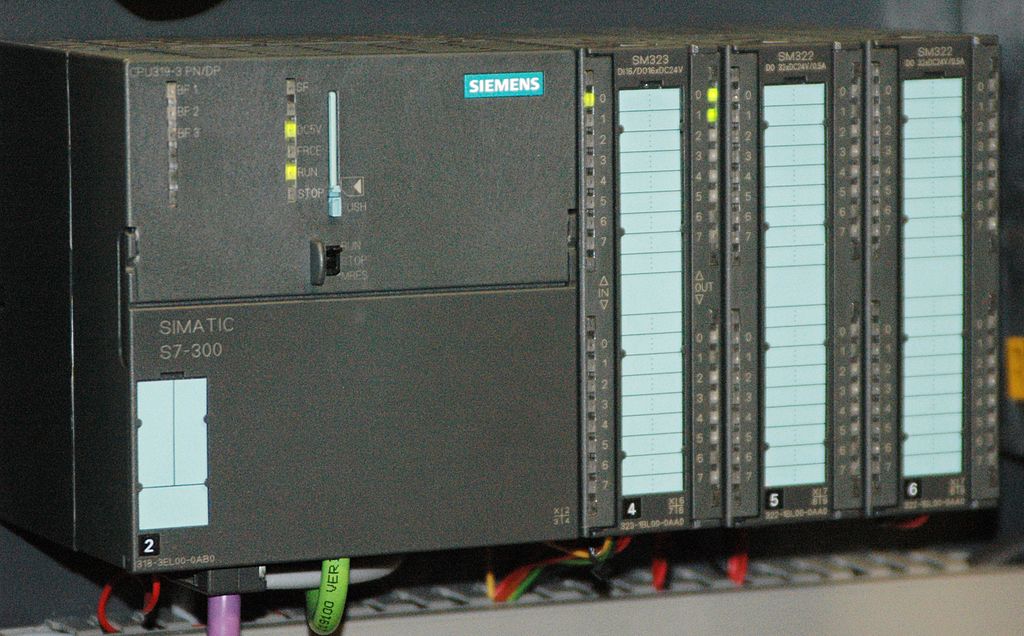
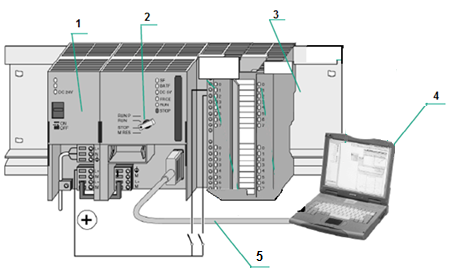


Рисунок 2.1 – ПЛК SIMATIC S7-300

Модульная конструкция, работа с естественным охлаждением, возможность применения структур локального и распределенного ввода-вывода, широкие коммуникационные возможности, множество функций, поддерживаемых на уровне операционной системы, удобство эксплуатации и обслуживания обеспечивают возможность получения рентабельных решений для построения систем автоматического управления в различных областях промышленного производства.

Программируемый контроллер состоит из источника питания, центрального процессора и модулей ввода и вывода. Программируемый логический контроллер контролирует установку и управляет ею с помощью программы S7. К модулям ввода/вывода в программе обращаются через адреса (рисунок 2.2).

Центральный процессор с его различными областями памяти формирует аппаратную основу для обработки пользовательских программ. Загрузочная память содержит программу пользователя в полном объеме: части программы в соответствии с ее исполнением в любой данный момент времени находятся в рабочей памяти, малое время доступа к которой необходимо для быстрой обработки программы.



1 – блок питания; 2 – переключатель режимов работы; 3 – стойка;

4 – программатор с ПО Step 7; 5 – кабель программатора

Рисунок 2.2 – Схема ПЛК SIMATIC S7-300

Эффективному применению контроллеров способствует возможность использования нескольких типов центральных процессоров различной производительности, наличие широкой гаммы модулей ввода-вывода дискретных и аналоговых сигналов, функциональных модулей и коммуникационных процессоров.

## Simatic Step 7

Simatic Step 7 — программное обеспечение фирмы Siemens (рисунок 2.3) для разработки систем автоматизации на основе программируемых логических контроллеров Simatic S7-300/S7-400/M7/C7 и WinAC.

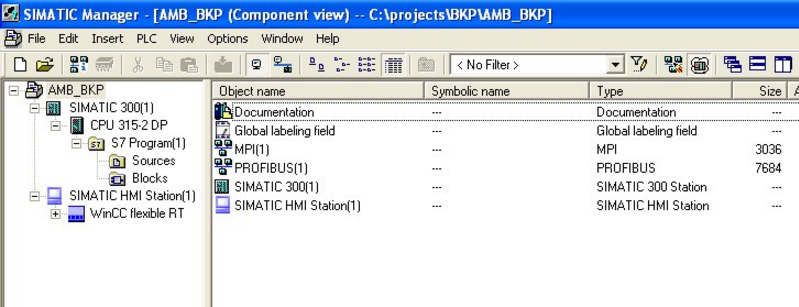
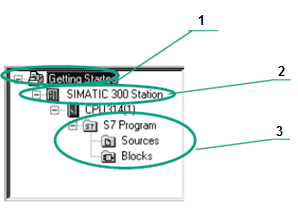


Рисунок 2.3 – ПЛК SIMATIC S7-300

С помощью этой программы выполняется комплекс работ по созданию и обслуживанию систем автоматизации на основе программируемых логических контроллеров Simatic S7-300 и Simatic S7-400 фирмы Siemens. В первую очередь это работы по программированию контроллеров.

При необходимости менеджер запускает дополнительный инструментарий, например, для конфигурирования станций, инициализации модулей, написания и тестирования программ.

Проект в среде Simatic Step 7 представляет следующую иерархическую структуру (рисунок 2.4).



1 –объекты структуры; 2 –данные о конфигурации;

3 – блоки с программами

Рисунок 2.4 – Иерархия проекта

## Языки программирования контроллеров

Программирование контроллеров производится редактором программ, обеспечивающим написание программ на трех языках: LAD, SCL и STL.

LAD – язык релейно-контактной логики (Ladder Diagram). Предназначен для программирования промышленных контроллеров. Синтаксис языка удобен для замены логических схем, выполненных на релейной технике. Ориентирован на инженеров по автоматизации, работающих на промышленных предприятиях. Обеспечивает наглядный интерфейс логики работы контроллера, облегчающий не только задачи собственно программирования и ввода в эксплуатацию, но и быстрый поиск неполадок в подключаемом к контроллеру оборудовании.

Программа на языке релейной логики имеет наглядный и интуитивно понятный инженерам-электрикам графический интерфейс (рисунок 2.5), представляющий логические операции, как электрическую цепь с замкнутыми и разомкнутыми контактами. Протекание или отсутствие тока в этой цепи соответствует результату логической операции (истина – если ток течёт; ложь – если ток не течёт).

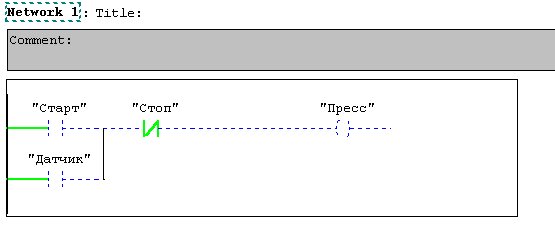


Рисунок 2.5 – Пример релейно-контактной логики

Язык SCL – структурированный язык управления (Structured Control Language). Является языком программирования высокого уровня для SIMATIC S7 (рисунок 2.6). Язык SCL оптимизирован для программирования программируемых контроллеров. Содержит в себе элементы языка Паскаль (Paskal) наряду с типичными для ПЛК элементами, такими, например, как «вход» и «выход». Язык SCL особенно подходит для программирования сложных алгоритмов или для задач, относящихся к области управления данными. Поддерживает характерную для STEP 7 блочную структуру, а также позволяет создавать программы, включающие в себя фрагменты на базовых языках программирования STL, LAD.

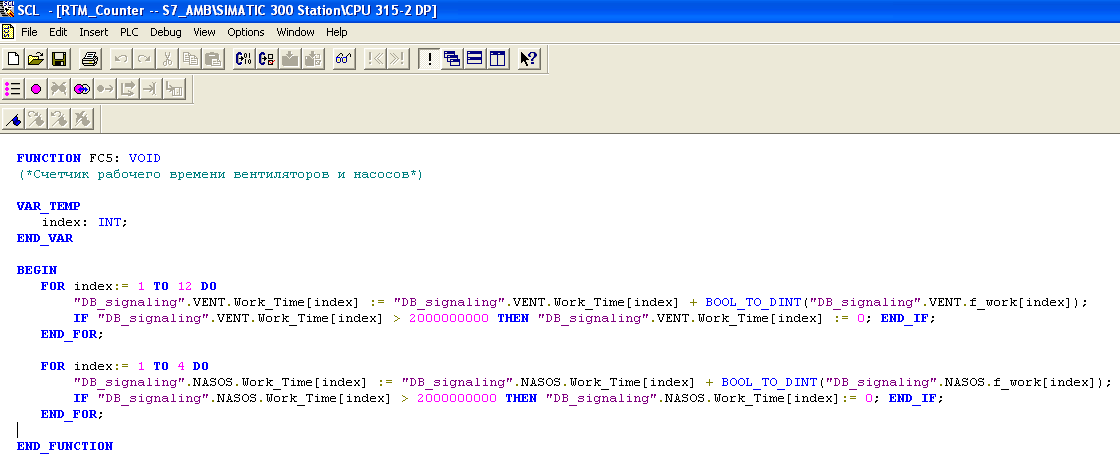


Рисунок 2.6 – Пример программы SCL

Язык STL – набор списка инструкций (от англ. Statement List). Это текстовый язык, подобный машинному коду (рисунок 2.7). Каждая команда соответствует шагу работы CPU при обработке программы. Несколько команд могут быть связаны друг с другом, образуя сегменты.

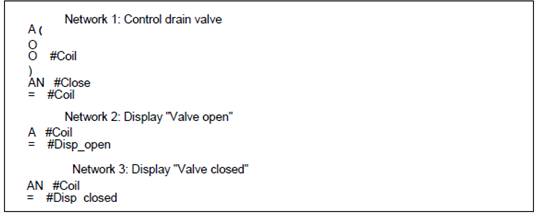


Рисунок 2.7 – Пример программы STL

## Загрузка программы в контроллер

Для того, чтобы контроллер начал использовать программу, разработанную в среде Simatic Step 7, ее необходимо загрузить в ПЛК (рисунок 2.8).

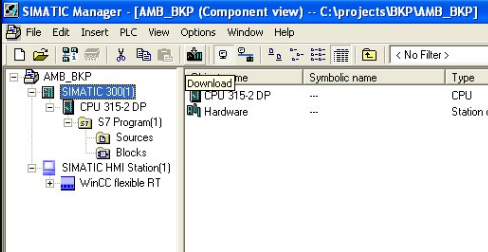


Рисунок 2.8 – Загрузка программы

Контроллер необходимо предварительно подключить ко внутренней сети для того, чтобы иметь возможность управлять программой (рисунок 2.9).

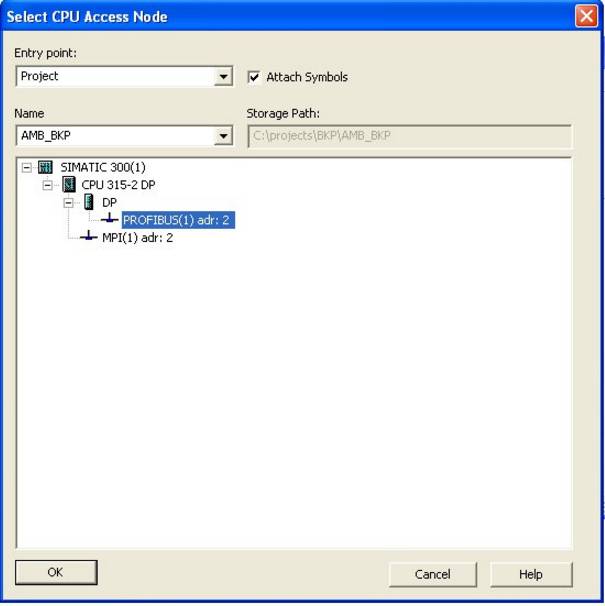


Рисунок 2.9 – Установка соединения

Кроме этого, необходимо убедиться в правильности настройки сети (рисунок 2.10).

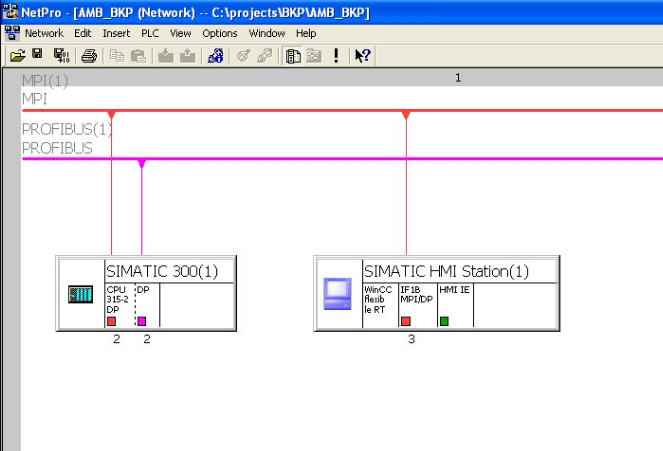


Рисунок 2.10 – Проверка настроек сети

## S7-PLCSIM

Для тестирования программного обеспечения используется специализированная утилита S7-PLCSIM (рисунок 2.11), представляющая собой симулятор контролера Simatic, с помощью которого имеется возможность задавать входные и выходные сигналы или же изменять значения в памяти ПЛК.

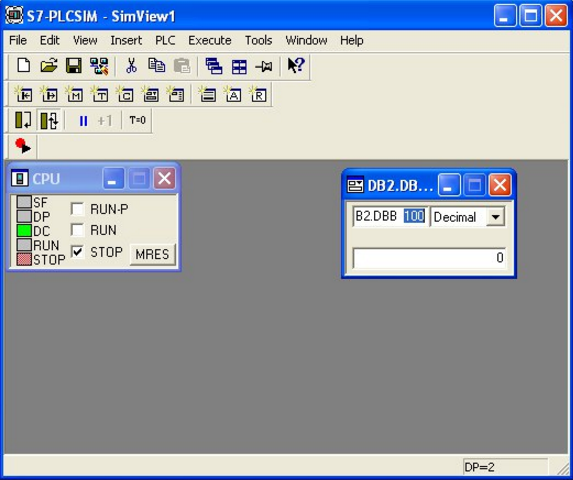


Рисунок 2.11 – Утилита S7-PLCSIM

## Конфигурация оборудования

Перед написанием программы произвелась конфигурация оборудования системы автоматизации и настройка параметров всех модулей с помощью встроенной утилиты Hardware Configuration (рисунок 2.12). Конфигурирование выполняется автономно без подключения к центральному процессору.

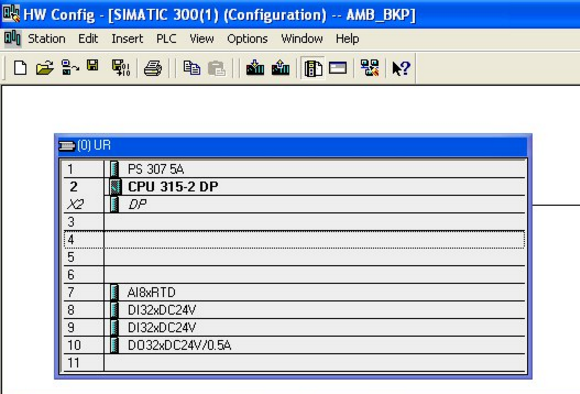


Рисунок 2.12 – Конфигурация оборудования

В данной работе использовалась следующая конфигурация оборудования:

* аналоговый модуль ввода SM 331 (AI 8 x RTD) для считывая показаний температуры и давления;
* два цифровых модуля ввода SM 321 (DI 32 x DC 24 V) для считывания флагов состояний секций, установки поддержания давления, насосов и технического состояния оборудования;
* цифровой модуль вывода SM 322 (DO 32 x 24 V/ 0.5 A) для управления вентиляторами и насосами.

Были сконфигурированы следующие параметры модулей (рисунок 2.13).

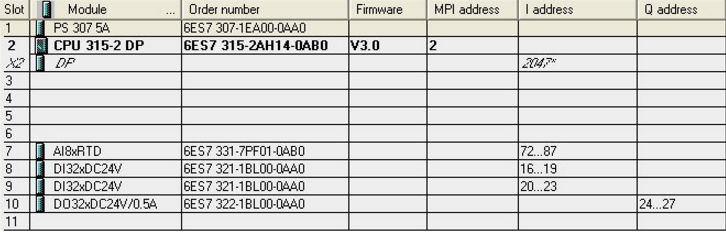


Рисунок 2.13 – Параметры модулей

Здесь задается для каждого модуля свой диапазон входных или выходных адресов.

# ОПИСАНИЕ ПРОГРАММЫ

## Общие сведения

Наименование программного обеспечения: «Программная автоматизация управления системой охлаждения производственного объекта».

Для функционирования данной программы необходима операционная система Windows XP версии SP2 или SP3.

Исходными языками программирования являются LAD, SCL, STL. Среда разработки – Siemens Simatic Step 7.

## Функциональное назначение

Программное обеспечение предназначено для автоматизации управления системой охлаждения производственного объекта.

## Описание логической структуры

Программная реализация данной работы имеет следующую структуру в среде Simatic Step 7 (рисунок 3.1).

В целом программное обеспечение для центрального процессора состоит из операционной системы и пользовательской программы.

Операционная система – это совокупность всех инструкций и описаний, которые осуществляют управление всеми системными ресурсами и процессами, использующими эти ресурсы. Она включает в себя такие функции, как резервирование данных в случае сбоя электропитания, активация приоритетных классов и так далее. Операционная система является компонентом центрального процессора, к которому у пользователя нет доступа в режиме записи. Однако, имеется возможность перезагружать операционную систему с карты памяти в случае, к примеру, обновления программы.

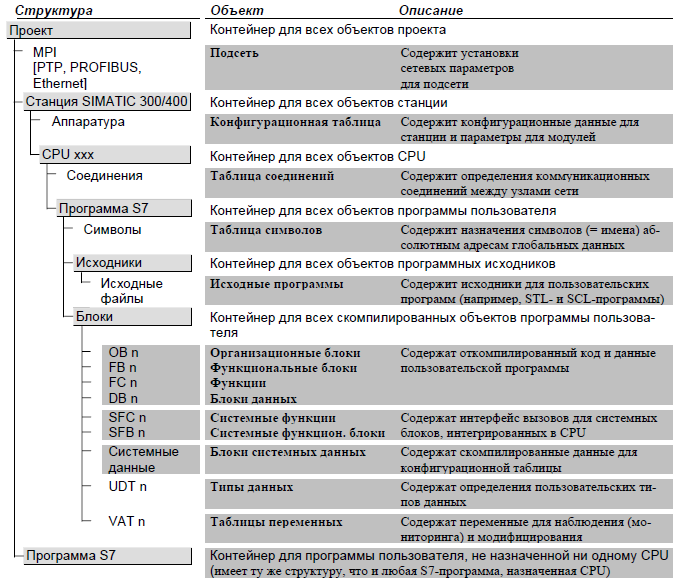


Рисунок 3.1 – Структура программы

Пользовательская программа представляет собой совокупность всех инструкций и описаний для обработки сигналов, с помощью которых осуществляется управление предприятием (процессом) в соответствии с определенной задачей автоматизации.

Пользовательская программа может состоять из программных разделов, которые обрабатываются центральным процессором в зависимости от определенных событий. Таким событием может быть запуск системы автоматизации, прерывание или обнаружение программной ошибки (рисунок 3.2).

Программы, назначенные для обработки событий, разделяются на приоритетные классы, которые определяют порядок обработки программы (система взаимных прерываний), когда происходит несколько событий.

Программой с низшим приоритетом является главная программа (main program), циклически обрабатываемая центральным процессором. События могут прервать главную программу в любом месте, после чего центральный процессор выполнит связанную с прерыванием обслуживающую программу (процедуру) или программу (процедуру) обработки ошибки и вернет управление главной программе.

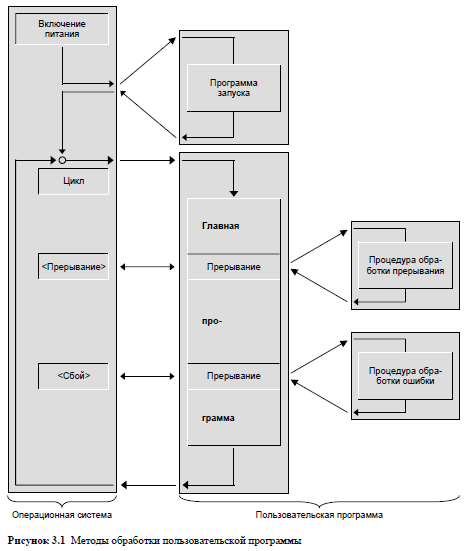


Рисунок 3.2 – Методы обработки пользовательской программы

Каждому событию соответствует специальный организационный блок. Организационные блоки в программе пользователя реализуют механизм приоритетных классов. При возникновении события центральный процессор активизирует назначенный организационный блок. Организационный блок – это часть пользовательской программы, которую вы можете сами написать. Организационные блоки - представляют собой интерфейс между операционной системой и программой пользователя. Также программа может быть разделена и сохранена в нескольких блоках (структурированная программа).

Перед началом обработки главной программы центральный процессор выполняет программу запуска. Эта программа может быть запущена включением питания, поворотом переключателя режимов на передней панели центрального процессора или с помощью программирующего устройства.

Главная программа располагается в организационном блоке ОВ1, который всегда обрабатывается центральным процессором. Начало пользовательской программы идентично первому сегменту в ОВ1. По завершению обработки ОВ1 (конец программы) центральный процессор передает управление операционной системе, и после вызова различных функций операционной системы, таких как обновление образа процесса, центральный процессор снова вызывает ОВ1.

Событиями, которые могут вмешиваться в работу программы, являются прерывания и ошибки.

Источником прерываний может быть процесс (аппаратные прерывания), или они могут исходить от центрального процессора (циклические прерывания, прерывания по времени суток и другие).

Программа ПЛК строится на основе блоков и функций: организационные блоки ОВ; функции FC, SFC; функциональные блоки FB, SFB; блоки данных DB. Синтаксис вводимой информации немедленно проверяется. Блок компилируется одновременно с записью и сохранением его в контейнере. С помощью пошагового программирования имеется возможность редактировать блоки центрального процессора в онлайновом режиме, даже во время выполнения операций. Количество блоков определенного блочного типа и размер блоков зависит от типа центрального процессора. Число организационных блоков и их номера фиксированы; они назначаются операционной системой центрального процессора.

Функция FC (пользовательская) содержит решение отдельной части программы. SFC (системная) - встроенные в операционную систему.

В основном функциональные блоки имеют те же возможности, что и функции. Кроме того, функциональные блоки имеют свою собственную область памяти в виде экземпляры блоков данных. Поэтому функциональные блоки можно использовать в часто повторяющихся сложных вычислениях, например, в задачах управления с обратной связью. Системные функциональные блоки SFB - это параметрируемые блоки, встроенные в операционную систему.

Блоки данных DB – область данных программы, в которых данные пользователя организованы в определенную структуру.

По существу, кодовые блоки состоят из трех частей (рисунок 3.4):

* заголовок блока, который содержит свойства блока, например, имя блока;
* раздел описаний (объявлений), в котором описаны (определены) локальные переменные блока (внутриблочные переменные);
* раздел программы, который содержит программу и комментарии к ней.

Переменная – это величина определенного формата (рисунок 3.5). Простые переменные состоят из адреса (например, вход 5.2, где 5 – номер байта, 2 – номер бита в нем). Также можно осуществить доступ к адресу или переменной символически, присвоив адресу имя (символ) в таблице символов.

Адреса, содержащие один, два или четыре байта, или переменные соответствующих типов называются численными операндами: 1 бит – BOOL, 8 бит – BYTE, 16 бит – WORD, 32 бита - DWORD.

Переменные, которые объявляются внутри блока, называются локальными (внутриблочными) переменными. К ним относятся параметры блока, статические и временные локальные данные и даже адреса данных в глобальных блоках данных.

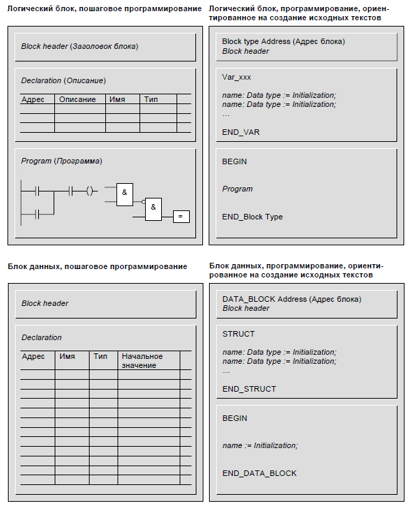


Рисунок 3.4 – Описание блока

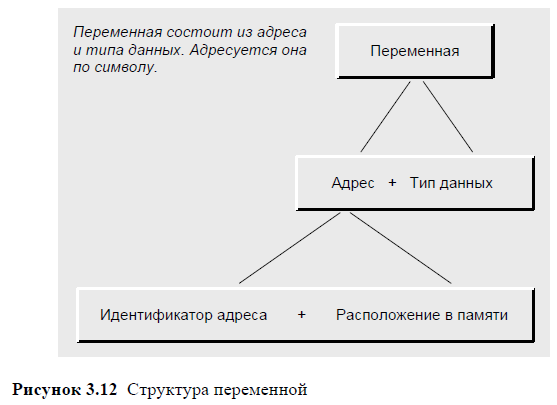


Рисунок 3.5 – Описание переменной

Когда эти переменные являются переменными простого типа данных, они также могут быть доступны как операнды (например, статические локальные данные – как DI-операнды, временные локальные данные – как L-операнды, а данные в глобальных блоках данных – как DB-операнды).

Наряду с этим, локальные переменные могут быть также сложных типов данных, таких как структуры или массивы.

Циклическая работа центрального процессора состоит из трех основных шагов (рисунок 3.6):

1) считывает состояние входных сигналов и обновляет таблицу входной области отображения процесса (I-область).

2) выполняет программу пользователя с соответствующими инструкциями.

3) записывает значения из таблицы области отображения выходов (Q-область в выходные модули).

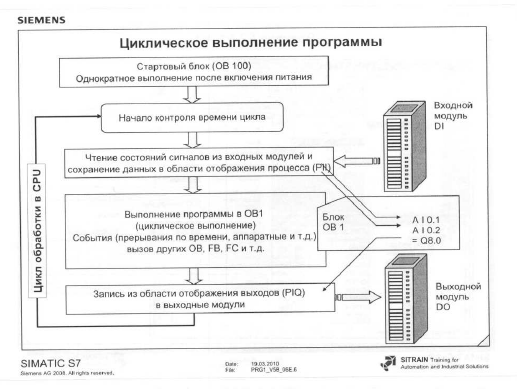


Рисунок 3.6 – Цикл программы ПЛК

Программное обеспечение состоит из следующих подсистем (рисунок 3.7):

* подсистема управления секциями;
* подсистема управления насосами;
* подсистема общего назначения.

Подсистема управления секциями вентиляторов состоит из следующих структур, функций и блоков:

* VentSection (UDT1) – пользовательский тип данных, с помощью которого описываются характеристики секции;
* ArrayVent (DB2) – блок данных хранит информацию о состояниях секций;
* ventArrayControl (FC4) – функция для управления секциями вентиляторов;
* ventArrayGetCountWorking (FC5) – функция для подсчета количества секций в работе;
* ventArrayDiagnostics (FC6) – функция для проверки рабочего состояния секций;
* ventArrayRun (FC7) – функция для запуска секции;
* ventArrayDisable (FC8) – функция для выключения секции;
* ventArrayEvents (FC9) – функция для обработки событий;
* ventArraySignalSend (FC23) – функция для отправки управляющих сигналов секциям;
* ventArraySignalRead (FC24) – функция для чтения текущего состояния секций.

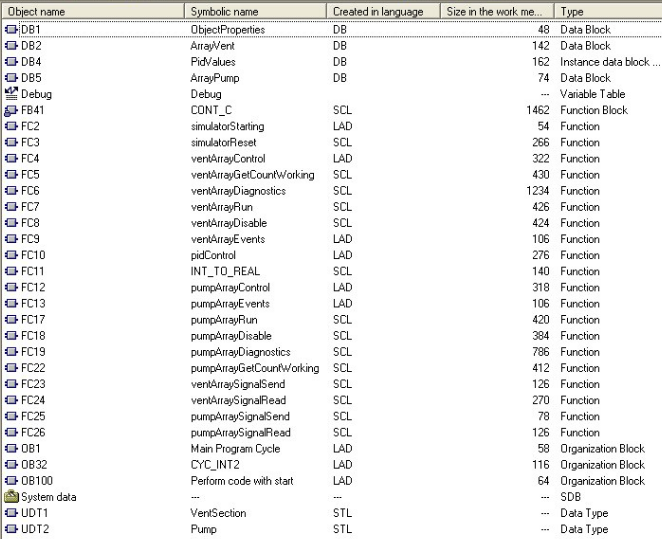


Рисунок 3.7 – Объекты программы

Подсистема управления насосами состоит из следующих структур, функций и блоков:

* Pump (UDT2) – пользовательский тип данных, с помощью которого описываются характеристики насоса;
* ArrayPump (DB5) – блок данных, служит для хранения информации о состояниях насосов;
* pumpArrayControl (FC12) – функция для управления насосами;
* pumpArrayEvents (FC13) – функция для обработки событий;
* pumpArrayRun (FC17) – функция для запуска насоса;
* pumpArrayDisable (FC18) – функция для выключения насоса;
* pumpArrayDiagnostics (FC19) – функция для проверки рабочего состояния насосов;
* pumpArrayGetCountWorking (FC22) – функция для подсчета количества насосов в работе;
* pumpArraySignalSend (FC25) – функция для отправки управляющих сигналов секциям;
* pumpArraySignalRead (FC26) – функция для чтения текущего состояния секций.

В подсистему общего назначения входят следующие блоки и функции:

* ObjectProperies (DB1) – блок данных для хранения информации об объекте автоматизации;
* PidValues (DB4) – блок данных для хранения рабочих значений управляющей функции температуры;
* pidControl (FC10) – функция управления температурой;
* Main Program Cycle (OB1) – организационный блок, вызывающийся каждый цикл работы программы контроллера;
* CYC\_INT2 (OB32) – организационный блок, который вызывается каждую секунду;
* Perform code with start (OB100) – организационный блок, работа которого выполняется только при старте контроллера.

## Используемые технические средства

Техническое обеспечение необходимое для работы программы:

* оперативная память 512 Мб и выше;
* процессор Intel Pentium IV 800 МГц и выше;
* существование логических и/или физических дисков со свободным дисковым пространством 2 Мб и выше;
* клавиатура
* манипулятор «мышь».

## Вызов и загрузка

Для функционирования программного обеспечения на персональном компьютере должна быть установлена операционная система Windows XP версии SP2 или SP3 и среда разработки Siemens Simatic Step 7.

Вызов программы осуществляется из среды Siemens Simatic Step 7.

## Входные данные

Входными данными для программного обеспечения «Программная автоматизация управления системой охлаждения производственного объекта» являются:

* входные сигналы с датчика SM 331 (AI 8 x RTD) для считывая показаний температуры и давления;
* входные сигналы с датчиков SM 321 (DI 32 x DC 24 V) для считывания флагов состояний секций, установки поддержания давления, насосов и технического состояния оборудования.

## Выходные данные

Выходными данными программного обеспечения «Программная автоматизация управления системой охлаждения производственного объекта» являются: выходные сигналы с датчика SM 322 (DO 32 x 24 V/ 0.5 A) для управления вентиляторами и насосами.

# ТЕКСТ ПРОГРАММЫ

## Главный модуль программы

OB1 "Main Program Sweep (Cycle)"

|  |
| --- |
| **Block: OB1 "Main Program Sweep (Cycle)"**  Главный цикл программы. Выполняется каждый такт.  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.jjwy3wyy8nus8dy8mgj2qyxjd.tmp  Network: 1  L "ID\_PVTEMPERATURE"ID72 -- Текущая температура  T "ObjectProperties".pvTemperatureDB1.DBD4 -- Текущая температура  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.8h7tnh1l1ose81rc9xp9_h6ge.tmp  Network: 2  L "ID\_PRESSURE"ID78 -- Текущее давление  T "ArrayPump".pressureDB5.DBD30 -- Давление  NOP 0 |

OB32 "Cyclic Interrupt"

|  |
| --- |
| **Block: OB32 "Cyclic Interrupt"**  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.6h0p47yfqa6u_geocahvfyerg.tmp  Network: 1 VentArray Control  Управление секциями  A "ObjectProperties".enableDB1.DBX0.1 -- Флаг выкл (0) / вкл (1) объекта (участка)  JNB \_001 CALL "ventArrayControl"FC4 -- Управление массивом из секций вентиляторов  \_001: NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.bps4o9g7goh07i2hg3_ddvzmc.tmp  Network: 2 pid control  Управление пид-регулятором  A "ObjectProperties".enableDB1.DBX0.1 -- Флаг выкл (0) / вкл (1) объекта (участка)  JNB \_002  CALL "pidControl"FC10 -- Управление пид регулятором  \_002: NOP 0  Network: 3 pump control  Управление насосами  A "ObjectProperties".enableDB1.DBX0.1 -- Флаг выкл (0) / вкл (1) объекта (уч астка)  JNB \_003  CALL "pumpArrayControl"FC12 -- Управление насосами \_  003: NOP 0 |

OB100 "Complete Restart"

|  |
| --- |
| **Block: OB100 "Complete Restart"**  Выполняет код при первом запуске контроллера  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.e06dphuko3j58r6150an1h8vc.tmp  Network: 1 Simulator for ArrayVentSection  Запускаем симулятор для массива из секций вентилятора  A "ObjectProperties".simulateDB1.DBX0.0 -- Флаг выкл (0) / вкл (1) симулятора  JNB \_001  CALL "simulatorStarting"FC2 -- Запуск симулятора  \_001: NOP 0 |

FC2"simulatorStarting" Запуск симулятора

|  |
| --- |
| Network: 1  CALL "simulatorReset"FC3 -- Сброс конфигурации симулятора в начальные на стройки  NOP 0 |

FC3 simulatorReset

|  |
| --- |
| (\*  Функция симулятора для сброса секций в начальные настройки  \*)  FUNCTION simulatorReset : VOID  VAR  i : INT;  size : INT;  END\_VAR  //Вентиляторы  i := 0;  size := "ArrayVent".size;  FOR i := 1 TO size DO  "ArrayVent".items[i].status := 3;  END\_FOR;  //Насосы  i := 0;  size := "ArrayPump".size;  FOR i := 1 TO 4 DO  "ArrayPump".items[i].status := 3;  END\_FOR;  END\_FUNCTION |

## Подсистема управления секциями вентиляторов

FC4 "ventArrayControl" Управление массивом из секций вентиляторов

|  |
| --- |
| **Block: FC4 VentArray Control**  Управление массивом секций  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.2wufqzscoopvuc625hsk_7nbh.tmp  Network: 1  Чтение входных сигналов  CALL "ventArraySignalRead"FC24 -- Получение сигналов от объектов автоматизации в программу контроллера  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.9gqg9nm3r5e7vxdrc7uhh9f5.tmp  Network: 2 VentArray Diagnostics  Диагностика секций  CALL "ventArrayDiagnostics"FC6 -- Диагностика секций  NOP 0  Network: 3 GetCount Enable items  Получение количества работающих секций и прибавление к часам работы +1с  CALL "ventArrayGetCountWorking"FC5 -- Получить количество работающих ве нтиляторов  RET\_VAL:="ArrayVent".itemsActiveDB2.DBW96 -- Количество секций в работе  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.zee_xt9f431wfawtsy9gc67re.tmp  Network: 4 VentArray Events  Запуск любого события с определенным промежутком времени  A(  A(  A(  L "ArrayVent".statusDB2.DBB100 -- Статус установки охлаждения  T #arrayStatus  SET  SAVE  CLR  A BR  )  JNB \_001  L 0  T "ArrayVent".statusDB2.DBB100 -- Статус установки охлаждения  SET  SAVE  CLR  \_001: A BR  )  A(  L #arrayStatus  L 0  <>I  )  A( L "ArrayVent".timerDisableDB2.DBW104 -- Таймер установки  L 0  ==I  )  JNB \_002  L "ArrayVent".timeDisableDB2.DBW102 -- Время между действиями  T "ArrayVent".timerDisableDB2.DBW104 -- Таймер установки  SET  SAVE  CLR  \_002: A BR  )  JNB \_003  CALL "ventArrayEvents"FC9 -- Ответы на события в установке - управления секциями event:=#arrayStatus  \_003: A BR  S #isDone |

FC9 "ventArrayEvents" Ответы на события в установке - управления секциями

|  |
| --- |
| **Block: FC9 VentArray Events**  Управляет событиями  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.cjhsqm2mvbq9j59jx9k6168pe.tmp  Network: 1 VentArray Run  Запуск секции  A(  L #event  L 51  ==I  )  JNB \_001  CALL "ventArrayRun"FC7 -- Запуск секции с минимальной наработкой  \_001: NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.fj7xwkn74s4d07ovdx2rnbsde.tmp  Network: 2 VentArray Disable  Выключение секции  A(  L #event  L 50  ==I  )  JNB \_002  CALL "ventArrayDisable"FC8 -- Выключение секции с максимальной наработкой  \_002: NOP 0 |

FC5 ventArrayGetCountWorking

|  |
| --- |
| (\*  Функция получения количества работающих секций  \*)  FUNCTION ventArrayGetCountWorking : INT  CONST  (\*  \*\*\*\*\* СОБЫТИЯ СЕКЦИИ \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 - перегрев  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  NEGATIVE\_TIME\_WAITING := BYTE#102; //отрицательное время ожидания  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  count : INT;  size : INT;  END\_VAR;  i := 0;  count := 0;  size := "ArrayVent".size;  //поиск секций: в состоянии работы  // статус = работа  // или в статусе выключения  // с положительным временем  FOR i := 1 TO size DO  IF (("ArrayVent".items[i].working) AND ("ArrayVent".items[i].status = NORMAL\_MODE))  OR  (("ArrayVent".items[i].status = DISABLING) AND (0 < "ArrayVent".items[i].timeWaiting))  THEN  count := count + 1;  "ArrayVent".items[i].hoursWorked := "ArrayVent".items[i].hoursWorked + 1;  END\_IF;  END\_FOR;  ventArrayGetCountWorking := count;  END\_FUNCTION |

FC6 ventArrayDiagnostics

|  |
| --- |
| (\*  Функция для диагностики секций  \*)  FUNCTION ventArrayDiagnostics : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ СЕКЦИИ \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 - перегрев  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  NEGATIVE\_TIME\_WAITING := BYTE#102; //отрицательное время ожидания  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  size : INT;  END\_VAR  i := 0;  size := "ArrayVent".size;  FOR i := 1 TO size DO  //проверка на ошибки  IF ("ArrayVent".items[i].enable) AND ("ArrayVent".items[i].hoursWorked < 0) //101 - отрицательные часы работы  THEN  "ArrayVent".items[i].status := NEGATIVE\_WORKING\_HOURS;  END\_IF;    IF ("ArrayVent".items[i].enable) AND ("ArrayVent".items[i].timeWaiting < 0) //102 - отрицательное время ожидания  THEN  "ArrayVent".items[i].status := NEGATIVE\_TIME\_WAITING;  END\_IF;  IF NOT ("ArrayVent".items[i].enable) AND ("ArrayVent".items[i].working) //103 - потеря питания при работе  THEN  "ArrayVent".items[i].status := POWER\_LOSS;  END\_IF;  IF ("ArrayVent".items[i].status = DISABLED) //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  AND ("ArrayVent".items[i].working)  THEN  "ArrayVent".items[i].status := CRASH\_MODE;  END\_IF;  IF (WORD\_TO\_INT(BYTE\_TO\_WORD("ArrayVent".items[i].status)) >= 100) //обнаружение ошибки  THEN  "ArrayVent".items[i].enable := false;  "ArrayVent".items[i].working := false;  "ArrayVentOutput".items[i].working := false;  CONTINUE;  END\_IF;  //выключение секции (уменьшаем времяподавление)  IF ("ArrayVent".items[i].autoMode)  AND ("ArrayVent".items[i].status = DISABLING)  AND ("ArrayVent".items[i].timeWaiting > 0)  THEN  "ArrayVent".items[i].timeWaiting := "ArrayVent".items[i].timeWaiting - 1;    END\_IF;  //выключение секции (если времяподавление уже нулевое или отрицательное)  IF ("ArrayVent".items[i].autoMode)  AND ("ArrayVent".items[i].status = DISABLING)  AND ("ArrayVent".items[i].timeWaiting <= 0)  THEN  "ArrayVent".items[i].working := false;  "ArrayVent".items[i].status := DISABLED;  END\_IF;  END\_FOR;  END\_FUNCTION |

FC7 ventArrayRun

|  |
| --- |
| (\*  Запуск секции с минимальной наработкой  \*)  FUNCTION ventArrayRun : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ СЕКЦИИ \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 - перегрев  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  NEGATIVE\_TIME\_WAITING := BYTE#102; //отрицательное время ожидания  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i :INT;  enum : INT;  size : INT;  hoursWorked : DINT;  END\_VAR  i := 0;  enum := -1;  size := "ArrayVent".size;  hoursWorked := 2147483647;  //поиск секции: в автоматическом режиме  // статус выключен  // с минимальной наработкой  FOR i := 1 TO size DO  IF ("ArrayVent".items[i].autoMode)  AND ("ArrayVent".items[i].status = DISABLED)  AND ("ArrayVent".items[i].hoursWorked < hoursWorked)    THEN    hoursWorked := "ArrayVent".items[i].hoursWorked;  enum := i;    END\_IF;  END\_FOR;  //включение найденной секции  IF (enum <> -1)  THEN  "ArrayVent".items[enum].working := true;  "ArrayVent".items[enum].status := NORMAL\_MODE;    END\_IF;  END\_FUNCTION |

FC8 ventArrayDisable

|  |
| --- |
| (\*  Выключение секции с максимальной наработкой  \*)  FUNCTION ventArrayDisable : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ СЕКЦИИ \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 - перегрев  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  NEGATIVE\_TIME\_WAITING := BYTE#102; //отрицательное время ожидания  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  enum : INT;  size : INT;  hoursWorked : DINT;  END\_VAR  i := -1;  enum := -1;  size := "ArrayVent".size;  hoursWorked := -2147483648;  //поиск секции: в автоматическом режиме  // статус = работает  // с максимальными часами работы  FOR i := 1 TO size DO  IF ("ArrayVent".items[i].autoMode)  AND ("ArrayVent".items[i].status = NORMAL\_MODE)  AND ("ArrayVent".items[i].hoursWorked >= hoursWorked)  THEN  enum := i;  hoursWorked := "ArrayVent".items[i].hoursWorked;  END\_IF;  END\_FOR;  //выключение секции  IF (enum <> -1)  THEN  "ArrayVent".items[enum].timeWaiting := "ArrayVent".timeDisable;  "ArrayVent".items[enum].status := DISABLING;  END\_IF;  END\_FUNCTION |

FC10"pidControl" Управление пид регулятором

|  |
| --- |
| **Block: FC10 PID Controller**  Управление пид регулятором  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.bv1m0ed2ziuw_2s0k8ro45p4d.tmp  Network: 1  L "ArrayVent".statusDB2.DBB100 -- Статус установки охлаждения  T #status  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.qln4dx6ken57is245w0zcyqnc.tmp  Network: 2  CALL "CONT\_C" , "PidValues"FB41 / DB4 -- Continuous Control  COM\_RST :=  MAN\_ON :=  PVPER\_ON:=  P\_SEL :=  I\_SEL :=  INT\_HOLD:=  I\_ITL\_ON:=  D\_SEL :=  CYCLE := SP\_  INT :="ObjectProperties".spTemperatureDB1.DBD8 -- Заданная температура  PV\_IN :="ObjectProperties".pvTemperatureDB1.DBD4 -- Текущая температура  PV\_PER :=  MAN :=  GAIN :=  TI :=  TD :=  TM\_LAG :=  DEADB\_W :=  LMN\_HLM :=1.200000e+001  LMN\_LLM :=0.000000e+000  PV\_FAC :=  PV\_OFF :=  LMN\_FAC :=  LMN\_OFF :=  I\_ITLVAL:=  DISV :=  LMN :=  LMN\_PER :=  QLMN\_HLM:=  QLMN\_LLM:=  LMN\_P :=  LMN\_I :=  LMN\_D :=  PV :=  ER :=  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.jn_4ewi4jvj0m63xnqzlnhmmb.tmp  Network: 3 INT\_TO\_REAL  Конвертирование int в real  A(  L "ArrayVent".itemsActiveDB2.DBW96 -- Количество секций в работе  ITD  T #countIA  SET  SAVE  CLR  A BR  )  JNB \_001  L #countIA  DTR  T #countItemsActive  \_001: NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.zjm8ff7quyaqz_vft_llxiqec.tmp  Network: 4  A(  L "PidValues".LMNDB4.DBD72 -- manipulated value  RND  T #tempLMN  AN OV  SAVE  CLR  A BR  )  JNB \_002 |

FC23 ventArraySignalSend

|  |
| --- |
| FUNCTION ventArraySignalSend : VOID  BEGIN  "Q\_VS\_WORKING[1]" := "ArrayVent".items[1].working;  "Q\_VS\_WORKING[2]" := "ArrayVent".items[2].working;  "Q\_VS\_WORKING[3]" := "ArrayVent".items[3].working;  "Q\_VS\_WORKING[4]" := "ArrayVent".items[4].working;  "Q\_VS\_WORKING[5]" := "ArrayVent".items[5].working;  "Q\_VS\_WORKING[6]" := "ArrayVent".items[6].working;  "Q\_VS\_WORKING[7]" := "ArrayVent".items[7].working;  "Q\_VS\_WORKING[8]" := "ArrayVent".items[8].working;  "Q\_VS\_WORKING[9]" := "ArrayVent".items[9].working;  "Q\_VS\_WORKING[10]" := "ArrayVent".items[10].working;  "Q\_VS\_WORKING[11]" := "ArrayVent".items[11].working;  "Q\_VS\_WORKING[12]" := "ArrayVent".items[12].working;  END\_FUNCTION |

FC24 ventArraySignalRead

|  |
| --- |
| FUNCTION ventArraySignalRead : VOID  BEGIN  "ArrayVent".items[1].enable := "I\_VS\_ENABLE[1]";  "ArrayVent".items[1].working := "I\_VS\_WORKING[1]";  "ArrayVent".items[1].autoMode := "I\_VS\_AUTOMODE[1]";  "ArrayVent".items[2].enable := "I\_VS\_ENABLE[2]";  "ArrayVent".items[2].working := "I\_VS\_WORKING[2]";  "ArrayVent".items[2].autoMode := "I\_VS\_AUTOMODE[2]";  "ArrayVent".items[3].enable := "I\_VS\_ENABLE[3]";  "ArrayVent".items[3].working := "I\_VS\_WORKING[3]";  "ArrayVent".items[3].autoMode := "I\_VS\_AUTOMODE[3]";  "ArrayVent".items[4].enable := "I\_VS\_ENABLE[4]";  "ArrayVent".items[4].working := "I\_VS\_WORKING[4]";  "ArrayVent".items[4].autoMode := "I\_VS\_AUTOMODE[4]";  "ArrayVent".items[5].enable := "I\_VS\_ENABLE[5]";  "ArrayVent".items[5].working := "I\_VS\_WORKING[5]";  "ArrayVent".items[5].autoMode := "I\_VS\_AUTOMODE[5]";  "ArrayVent".items[6].enable := "I\_VS\_ENABLE[6]";  "ArrayVent".items[6].working := "I\_VS\_WORKING[6]";  "ArrayVent".items[6].autoMode := "I\_VS\_AUTOMODE[6]";  "ArrayVent".items[7].enable := "I\_VS\_ENABLE[7]";  "ArrayVent".items[7].working := "I\_VS\_WORKING[7]";  "ArrayVent".items[7].autoMode := "I\_VS\_AUTOMODE[7]";  "ArrayVent".items[8].enable := "I\_VS\_ENABLE[8]";  "ArrayVent".items[8].working := "I\_VS\_WORKING[8]";  "ArrayVent".items[8].autoMode := "I\_VS\_AUTOMODE[8]";  "ArrayVent".items[9].enable := "I\_VS\_ENABLE[9]";  "ArrayVent".items[9].working := "I\_VS\_WORKING[9]";  "ArrayVent".items[9].autoMode := "I\_VS\_AUTOMODE[9]";  "ArrayVent".items[10].enable := "I\_VS\_ENABLE[10]";  "ArrayVent".items[10].working := "I\_VS\_WORKING[10]";  "ArrayVent".items[10].autoMode := "I\_VS\_AUTOMODE[10]";  "ArrayVent".items[11].enable := "I\_VS\_ENABLE[11]";  "ArrayVent".items[11].working := "I\_VS\_WORKING[11]";  "ArrayVent".items[11].autoMode := "I\_VS\_AUTOMODE[11]";  "ArrayVent".items[12].enable := "I\_VS\_ENABLE[12]";  "ArrayVent".items[12].working := "I\_VS\_WORKING[12]";  "ArrayVent".items[12].autoMode := "I\_VS\_AUTOMODE[12]";  END\_FUNCTION |

## Подсистема управления насосами

FC12"pumpArrayControl" Управление насосами

|  |
| --- |
| Block: FC12 PumpArray control  Управление насосами  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.9up_puqik4uwxo78udle71gjg.tmp  Network: 1  CALL "pumpArraySignalRead"FC26 -- Получение сигналов от объектов автомати зации в программу контроллера  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.32k9y5kw_h82b92gbtv8zpi3b.tmp  Network: 2 pumpDiagnostics  Диагностика секций  CALL "pumpArrayDiagnostics"FC19 -- Диагностика секций  NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.y70szs21qsoctej7vm_n31l0e.tmp  Network: 3  Получение количество работающих насосов в данный момент  CALL "pumpArrayGetCountWorking"FC22 -- Получить количество работающих на сосов  RET\_VAL:="ArrayPump".itemsActiveDB5.DBW24 -- Количество работающих насосов  NOP 0  Network: 4  Управление действиями и событиями в установке  A(  A(  A(  L "ArrayPump".statusDB5.DBB28 -- Статус установки насосов  T #arrayStatus  SET  SAVE  CLR  A BR  )  JNB \_001  L 0  T "ArrayPump".statusDB5.DBB28 -- Статус установки насосов  SET  SAVE  CLR  \_001: A BR  )  A(  L #arrayStatus  L 0  <>I  )  A(  L "ArrayPump".timerDisableDB5.DBW36 -- Таймер установки  L 0  ==I  )  JNB \_002  L "ArrayPump".timeDisableDB5.DBW34 -- Время между действиями  T "ArrayPump".timerDisableDB5.DBW36 -- Таймер установки  SET  SAVE  CLR  \_002: A BR  )  JNB \_003  CALL "pumpArrayEvents"FC13 -- Ответы на события в установке - упра вления насосами  arrayStatus:=#arrayStatus  \_003: NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.p4km61hp_9rbaxgl4nvy_flae.tmp  Network: 5  Управление таймером: уменьшение времени работы таймера  A(  L "ArrayPump".timerDisableDB5.DBW36 -- Таймер установки  L 0  <>I  )  JNB \_004  L "ArrayPump".timerDisableDB5.DBW36 -- Таймер установки  L 1  -I  T "ArrayPump".timerDisableDB5.DBW36 -- Таймер установки  \_004: NOP 0  Network: 6  Проверка на количество работающих насосов >= 2  A(  L "ArrayPump".itemsActiveDB5.DBW24 -- Количество работающих насосов  L 2  <I  )  JNB \_005  L 51  T "ArrayPump".statusDB5.DBB28 -- Статус установки насосов  \_005: NOP 0  C:\Users\Ghost Writter\AppData\LocalLow\Temp\Microsoft\OPC\DDT.jnvv2tjnj3pemlvv0h_y3sqkd.tmp  Network: 7  CALL "pumpArraySignalSend"FC25 -- Отправка сигналов к объектам автоматиза ции из программы контроллера  NOP 0 |

FC17 pumpArrayRun

|  |
| --- |
| (\*  Функция включения насоса  \*)  FUNCTION pumpArrayRun : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ НАСОСА \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 -  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  enum : INT;  hoursWorked : DINT;  size : INT;  END\_VAR;  i := 1;  enum := -1;  hoursWorked := 2147483647;  size := "ArrayPump".size;  //включение насоса: в автоматическом режиме  // в выключенном состоянии  // с минимальной наработкой  FOR i := 1 TO 4 DO  IF ("ArrayPump".items[i].autoMode)  AND ("ArrayPump".items[i].status = DISABLED)  AND (hoursWorked > "ArrayPump".items[i].hoursWorked)  THEN    hoursWorked := "ArrayPump".items[i].hoursWorked;  enum := i;    END\_IF;  END\_FOR;  IF enum <> -1  THEN  "ArrayPump".items[enum].working := true;  "ArrayPump".items[enum].status := NORMAL\_MODE;    END\_IF;  END\_FUNCTION |

FC18 pumpArrayDisable

|  |
| --- |
| (\*  Функция выключения насоса  \*)  FUNCTION pumpArrayDisable : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ НАСОСА \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 -  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  enum : INT;  size : INT;  hoursWorked : DINT;  END\_VAR;  i := 1;  enum := -1;  size := "ArrayPump".size;  hoursWorked := -2147483648;  //выключение насоса: в автоматическом режиме  // в состоянии работы  // с максимальным числом рабочих часов  FOR i := 1 TO size DO  IF ("ArrayPump".items[i].autoMode)  AND ("ArrayPump".items[i].status = NORMAL\_MODE)  AND ("ArrayPump".items[i].hoursWorked >= hoursWorked)  THEN  enum := i;  hoursWorked := "ArrayPump".items[i].hoursWorked;  END\_IF;  END\_FOR;  IF (enum <> -1)  THEN  "ArrayPump".items[enum].status := DISABLING;  END\_IF;  END\_FUNCTION |

FC19 pumpArrayDiagnostics

|  |
| --- |
| (\*  Функция диагностики насосов  \*)  FUNCTION pumpArrayDiagnostics : VOID  CONST  (\*  \*\*\*\*\* СОБЫТИЯ НАСОСА \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 -  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  size : INT;  END\_VAR  i := 0;  size := "ArrayPump".size;  FOR i := 1 TO size DO  //проверка на ошибки  IF ("ArrayPump".items[i].enable) AND ("ArrayPump".items[i].hoursWorked < 0) //101 - отрицательные часы работы  THEN  "ArrayPump".items[i].status := NEGATIVE\_WORKING\_HOURS;  END\_IF;  IF NOT ("ArrayPump".items[i].enable) AND ("ArrayPump".items[i].working) //103 - потеря питания при работе  THEN  "ArrayPump".items[i].status := POWER\_LOSS;  END\_IF;  IF ("ArrayPump".items[i].status = DISABLED) AND ("ArrayPump".items[i].working) //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  THEN  "ArrayPump".items[i].status := CRASH\_MODE;  END\_IF;  IF (WORD\_TO\_INT(BYTE\_TO\_WORD("ArrayPump".items[i].status)) >= 100) //обнаружение ошибки  THEN  "ArrayPump".items[i].enable := false;  "ArrayPump".items[i].working := false;  CONTINUE;  END\_IF;  //выключение если нет никаких ошибок    IF ("ArrayPump".items[i].autoMode)  AND ("ArrayPump".items[i].status = DISABLING)  THEN  "ArrayPump".items[i].working := false;  "ArrayPump".items[i].status := DISABLED;  END\_IF;    END\_FOR;  END\_FUNCTION |

FC22 pumpArrayGetCountWorking

|  |
| --- |
| (\*  Функция подсчета количества работающих насосов  \*)  FUNCTION pumpArrayGetCountWorking : INT  CONST  (\*  \*\*\*\*\* СОБЫТИЯ НАСОСА \*\*\*\*\*  \*\*\*\*\* РАБОЧИЙ СТАТУС \*\*\*\*\*  0 - нормальная работа  1 - выключение  2 - включение  3 - выключен  \*\*\*\*\* ОШИБКА РАБОТЫ \*\*\*\*\*  100 -  101 - отрицательные часы работы  102 - отрицательное время ожидания  103 - потеря питания при работе  104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)    \*)  NORMAL\_MODE := BYTE#0; //нормальная работа  DISABLING := BYTE#1; //выключение  ENABLED := BYTE#2; //включен  DISABLED := BYTE#3; //выключен  NEGATIVE\_WORKING\_HOURS := BYTE#101; //отрицательные часы работы  POWER\_LOSS := BYTE#103; //потеря питания при работе  CRASH\_MODE := BYTE#104; //104 - сбой в режиме работы (статус - выключен (3), а флаг working = true)  END\_CONST  VAR  i : INT;  count : INT;  size : INT;  END\_VAR;  i := 0;  count := 0;  size := "ArrayPump".size;  //подсчет количества насосов: исправных  // в работе  // включёных или выключающихся  FOR i := 1 TO size DO  IF ("ArrayPump".items[i].enable)  AND ("ArrayPump".items[i].working)  AND (("ArrayPump".items[i].status) = NORMAL\_MODE)  OR  (("ArrayPump".items[i].status) = DISABLING)  THEN  count := count + 1;  "ArrayPump".items[i].hoursWorked := "ArrayPump".items[i].hoursWorked + 1;  END\_IF;  END\_FOR;  pumpArrayGetCountWorking := count;  END\_FUNCTION |

FC25 pumpArraySignalSend

|  |
| --- |
| FUNCTION pumpArraySignalSend : VOID  BEGIN  "Q\_PUMP\_WORKING[1]" := "ArrayPump".items[1].working;  "Q\_PUMP\_WORKING[2]" := "ArrayPump".items[2].working;  "Q\_PUMP\_WORKING[3]" := "ArrayPump".items[3].working;  "Q\_PUMP\_WORKING[4]" := "ArrayPump".items[4].working;  END\_FUNCTION |

FC26 pumpArraySignalRead

|  |
| --- |
| FUNCTION pumpArraySignalRead : VOID  BEGIN  "ArrayPump".items[1].enable := "I\_PUMP\_ENABLE[1]";  "ArrayPump".items[1].working := "I\_PUMP\_WORKING[1]";  "ArrayPump".items[1].autoMode := "I\_PUMP\_AUTOMODE[1]";  "ArrayPump".items[2].enable := "I\_PUMP\_ENABLE[2]";  "ArrayPump".items[2].working := "I\_PUMP\_WORKING[2]";  "ArrayPump".items[2].autoMode := "I\_PUMP\_AUTOMODE[2]";  "ArrayPump".items[3].enable := "I\_PUMP\_ENABLE[3]";  "ArrayPump".items[3].working := "I\_PUMP\_WORKING[3]";  "ArrayPump".items[3].autoMode := "I\_PUMP\_AUTOMODE[3]";  "ArrayPump".items[4].enable := "I\_PUMP\_ENABLE[4]";  "ArrayPump".items[4].working := "I\_PUMP\_WORKING[4]";  "ArrayPump".items[4].autoMode := "I\_PUMP\_AUTOMODE[4]";  END\_FUNCTION |

# РУКОВОДСТВО ПРОГРАММИСТА

## Назначение и условия применения программы

### Назначение программы

Программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» предназначено для автоматизации производственного объекта.

В данном программном обеспечении были реализованы следующие функции:

* обработка и анализ входных данных объекта автоматизации;
* принятие решения на основе данных;
* отправка выходных данных к объекту автоматизации.

### Условия применения программы

Техническое обеспечение необходимое для функционирования программы:

* операционная система Windows XP версии SP2 или SP3;
* оперативная память 512 Мб и выше;
* процессор Intel Pentium IV 800 МГц и выше;
* существование логических и/или физических дисков со свободным дисковым пространством 2 Мб и выше;
* клавиатура
* манипулятор «мышь»;

## Характеристика программы

Исходными языками программирования являются LAD, SCL, STL. Среда разработки – Siemens Simatic Step 7.

Программное обеспечение состоит из ряда подсистем:

1. подсистема общего назначения для управления другими подсистемами;
2. подсистема управления секциями вентиляторов;
3. подсистема управления насосами.

Для просмотра результатов работы программы используется среда разработки Siemens Simatic Step 7. Для этого используется специальный редактор.

Для подсистемы общего назначения используется блок данных DB1, который содержит общую информацию об объекте автоматизации (рисунок 5.1).



Рисунок 5.1 – Вывод блока DB1

Для подсистемы управления секциями вентиляторов используется блок данных DB2, который содержит информацию о текущей работе секций вентиляторов (рисунок 5.2).

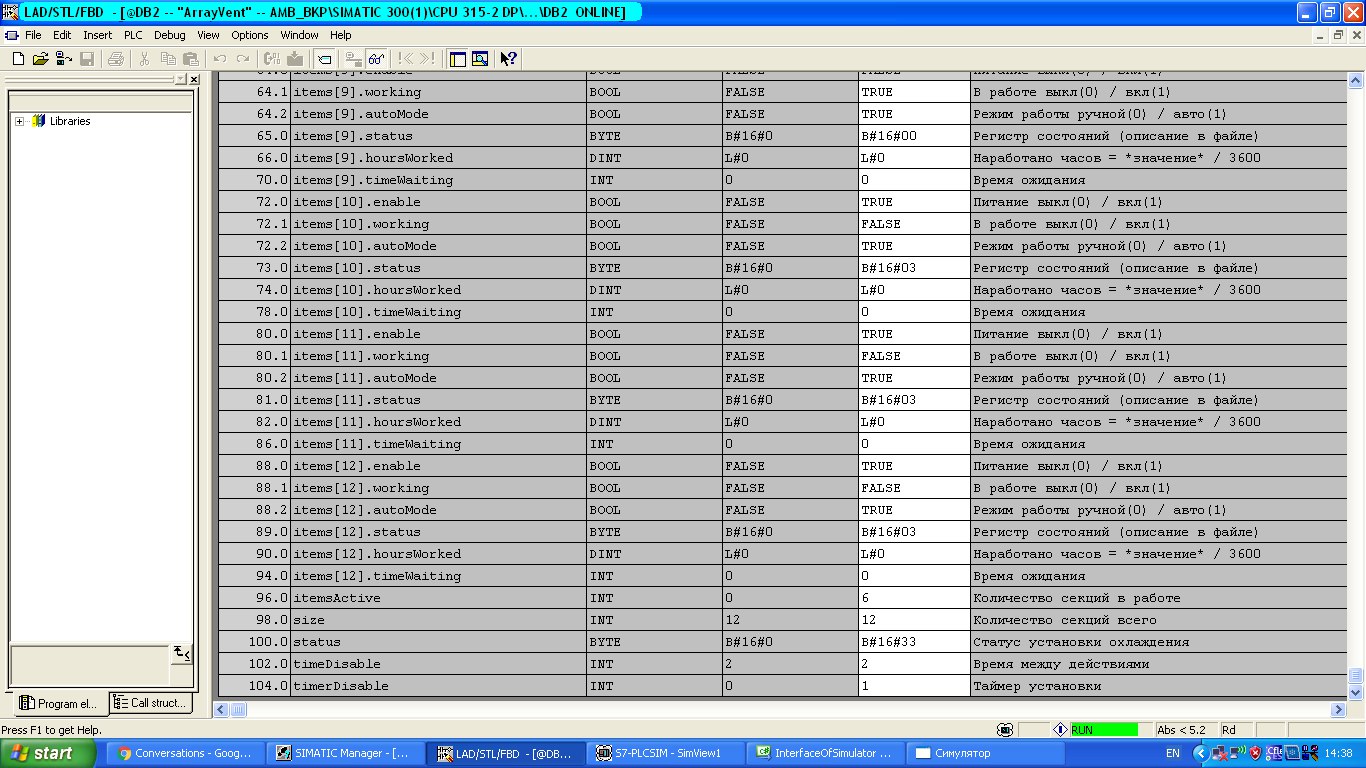


Рисунок 5.2 – Вывод блока DB2

Для подсистемы управления насосами используется блок данных DB3, который содержит общую информацию о текущей работе насосов (рисунок 5.3).



Рисунок 5.3 – Вывод блока DB3

Главной функцией программы является организационный блок OB1, которая выполняется с каждым тактом процессора ПЛК.

## Обращение к программе

Для вызова программы необходимо загрузить программу в память контроллера, после чего запустить его. Программа работает в автономном режиме.

## Сообщения

Данное программное обеспечение обрабатывает все возникающие ошибки и записывает их в соответствующие организационные блоки для секций вентиляторов и насосов.

# РУКОВОДСТВО ОПЕРАТОРА

## Назначение программы

Программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» предназначено для автоматизации управления производственного объекта.

В данном программном обеспечении были реализованы следующие функции:

* считывание входных сигналов с датчиков;
* анализ входных данных;
* принятие решения на основании данных;
* отправка управляющих (выходных) сигналов к объекту автоматизации.

## Условия выполнения программы

Техническое обеспечение на стороне пользователя, необходимое для функционирования программы:

* оперативная память 512 Мб и выше;
* процессор Intel Pentium IV 800 МГц и выше;
* существование логических и/или физических дисков со свободным дисковым пространством 2 Мб и выше;
* клавиатура
* манипулятор «мышь».

Для функционирования данной программы необходимо, чтобы у пользователя была установлена операционная система Windows XP версии SP2 или SP3.

## Выполнение программы

Для того, чтобы контроллер начал использовать программу, разработанную в среде Simatic Step 7, ее необходимо загрузить в ПЛК (см. рисунок 2.8).

Контроллер необходимо предварительно подключить ко внутренней сети для того, чтобы иметь возможность управлять программой (см. рисунок 2.9).

Кроме этого, необходимо убедиться в правильности настройки сети (рисунок 2.10).

Для тестирования программного обеспечения используется специализированная утилита S7-PLCSIM (см. рисунок 2.11), представляющая собой симулятор контролера Simatic, с помощью которого имеется возможность задавать входные и выходные сигналы или же изменять значения в памяти ПЛК.

Перед написанием программы произвелась конфигурация оборудования системы автоматизации и настройка параметров всех модулей с помощью встроенной утилиты Hardware Configuration. Конфигурирование выполняется автономно без подключения к центральному процессору.

## Сообщения оператору

Данное программное обеспечение обрабатывает все возникающие ошибки и записывает их в соответствующие организационные блоки для секций вентиляторов и насосов.

# ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

## Объект испытаний

Наименование программного обеспечения: «Программная автоматизация управления системой охлаждения производственного объекта».

Программное обеспечение предназначено для автоматизации управления системой охлаждения производственного объекта.

Объектом автоматизации является система охлаждения производственного объекта.

Возможными областями применения программного обеспечения являются производственные предприятия.

## Цель испытаний

Целью данных испытаний являются проверка работоспособности программного обеспечения и соответствие программы и результатов ее работы требованиям корректности, надежности и правильности, предъявленными в техническом задании.

## Требования к программе

Программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» выполняет в режиме реального времени управление системой охлаждения производственного объекта.

Программное обеспечение состоит из модулей, обеспечивающих гибкую настройку под конкретные нужды пользователя. Каждый модуль выполняет определенную функцию. Добавление или удаление какого-либо модуля не влияет на выполнение основной задачи и не приводит к отказам программы.

Во время эксплуатации программного обеспечения при условии соблюдения всех требований, перечисленных в пункте 1.4 «Технического задания», не было зафиксировано ни одного случая отказа программы. Все возможные ошибки, не обрабатываемые программой, обрабатываются операционной системой.

В процессе функционирования программное обеспечение «Программная автоматизация управления системой охлаждения производственного объекта» не влияет на работу любых других программных средств и не приводит к сбоям в работе компьютера и операционной системы.

Оценку надежности программы необходимо производить в результате многочасового тестирования с использованием всех функциональных возможностей визуализатора.

Диагностирование программного обеспечения должно производиться при соблюдении условий эксплуатации указанных в разделе 1.4 «Технического задания».

## Требования к программной документации

При испытании программы «Программа автоматизации» использовались следующие документы:

* «Техническое задание»;
* «Описание средств автоматизации»;
* «Описание программы»;
* «Руководство программиста»;
* «Руководство оператора».

## Средства и порядок испытаний

Технические средства, используемые во время испытаний:

* оперативная память 512 Мб и выше;
* процессор Intel Pentium IV 800 МГц и выше;
* существование логических и/или физических дисков со свободным дисковым пространством 2 Мб и выше;
* клавиатура
* манипулятор «мышь».

Во время испытаний использовалась операционная система Windows XP версии SP2.

Испытания программы проводились в следующем порядке:

1. испытание на корректность (адекватно ли программа реагирует на ввод-вывод информации);
2. испытание на правильность (соответствуют ли полученные результаты ожидаемым);
3. испытание на надежность (процент отказов и сбоев системы).

## Методы испытаний

### Проверка на корректность

Был произведен запуск программы.

На первом шаге тестирования была проведена проверка на корректность обработки входных сигналов с заданной конфигурацией оборудования.

На втором шаге была проведена проверка корректности принятия решения на основании входных данных.

На третьем шаге был произведен ввод некорректной информации.

После чего был выполнен четвертый шаг – проверка отправки корректных управляющих сигналов объекту автоматизации.

По результатам выполнения этих шагов, можно сделать вывод, что работа программы удовлетворяет необходимым требованиям и не отличаются от ожидаемых, следовательно, программа работает корректно.

### Проверка на правильность

При работе программы с любыми корректными данными, программа должна отображать на экране данные из блоков данных и входных модулей.

Программа тестировалась при различных алгоритмах генерации входных данных.

Для любых входных данных программа выдает корректную информацию.

Во время тестирования тестовый файл был запущен более 100 раз на различных компьютерах, программа реагировала правильно на все действия пользователя, результат работы программы соответствовал заявленными требованиями.

При проверке на правильность результат не отличался от ожидаемого, поэтому можно сказать, что программа работает правильно.

### Проверка на надежность

При обработке различных наборов данных процент погрешностей и ошибок не должен превышать допустимого значения. При проведении серии экспериментов превышение допустимого предела по проценту ошибок и погрешностей не зафиксировано.

Программа тестировалась в течение месяца различными пользователями. В процессе тестирования глобальных сбоев системы обнаружено не было. Все замечания пользователей были рассмотрены и исправлены в процессе тестирования.

Нарушений в работе операционной системы и параллельных программ не зафиксировано.

Поскольку в ходе работы программы не было зафиксировано ошибок и погрешностей, то можно сделать вывод, что программа работает надежно.

##### Заключение

Целью данной выпускной квалификационной работы является автоматизация управления системой охлаждения производственного объекта.

В ходе данной выпускной квалификационной работы был реализован алгоритм автоматизации для ПЛК Siemens Simatic S7-300, с помощью которого происходит эффективное управление оборудованием и ресурсами.

После запуска программное обеспечение работает в автономном режиме без участия пользователя. Информация об ошибках и сбоях, а также о текущих действиях алгоритма хранится в соответствующих блоках данных подсистем.

Разработанное программное обеспечение, позволяет повысить качество труда на рабочем месте производственного предприятия.

Для разработки программного обеспечения использовались языки промышленного программирования: LAD, SCL, STL. Для реализации проекта использовалась среда разработки Siemens Simatic Step 7. Тестирование производилось на компьютере с процессорам Intel Pentium IV 800 с установленной операционной системой Windows XP версии SP2.

##### Список использованных источников

1. Кривченко, И.В. AVR - микроконтроллеры: очередной этап на пути развития / И. В. Кривченко // Компоненты и технологии. – 2002. – № 3. – С. 5-14.
2. Википедия – свободная энциклопедия [Электронный ресурс] / Wikimedia Foundation – режим доступа: http://www.wikipedia.org/wiki/Simatic\_Step\_7, свободный. – Загл. с экрана. Яз. рус., англ.
3. Siemens – Global WebSite [Электронный ресурс] / Siemens – режим доступа: http://www.siemens.com, свободный. – Загл. с экрана. Англ.
4. Бергер, Г. Автоматизация посредством STEP 7 с использованием STL и SCL и програмируемых контроллеров SIMATIC S7-300/400 / Г. Бергер. – Нюрнберг : Siemens AG, – 2001. – 776 с.