Министерство науки и высшего образования Российской Федерации
Филиал федерального государственного автономного образовательного учреждения высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)» в г. Миассе
Факультет «Электротехнический»
Кафедра «Автоматика»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой

_____ С.С. Голощапов

_____ 2018 г.

Комплекс программно-технических средств для изучения семейства протоколов передачи данных Modbus

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

ЮУрГУ-270304.2018.348.00 ПЗ ВКР

Руководитель проекта
Старший преподаватель

_____Е. А. Канашев

«____» _____2018 г.

Автор работы
студент группы МиЭт-482
_____ Е.А. Щипакина

«____» _____2018 г.

Нормоконтролер
_____ Т.А. Барбасова

«____» _____2018 г.

Миасс 2018

АННОТАЦИЯ

Щипакина Е. А. Комплекс программно-технических средств для изучения семейства протоколов Modbus. – Миасс: филиал «ФГАО ВО ЮУрГУ (НИУ)», Эт, 2018, 89 с., 27 ил., библиогр. список – 52 наим., 3 прил., 3 листа схем ф. А3, 4 листа ф. А4

В выпускной квалификационной работе представлена разработка комплекса ПТС для изучения семейства протоколов передачи данных Modbus.

Необходимость создания такого вида установки с реализацией на ней программного обеспечения, позволяющего имитировать работу термоконтроллера, объясняется тем, что уже имеющиеся разработки сложны и не могут подстроиться под весь спектр задач, которые необходимо выполнять.

В результате работы произведен анализ функций лабораторного стенда, синтез электрической схемы стенда, а также был создан алгоритм работы установки. Результатом работы стало тестирование на разработанной установке спроектированного программного обеспечения, которое по всем параметрам соответствует требованиям технического задания.

Оценив данную работу, приходим к выводу, что тема является актуальной в настоящее время, так как используются современные методы реализации поставленной задачи.

					270304.2018.348.00 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Щипакина Е.А.			Комплекс программно-технических средств для изучения семейства протоколов передачи данных Modbus	Лит. Д	Лист 4 / Листов 89
Провер.		Канашев Е.А.					
Н. Контр.		Барбасова Т.А.				ЮУрГУ (НИУ) Кафедра «Автоматика»	
Утв.		Голощапов С.С					

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В рамках выпускной квалификационной работы рассматривается комплекс программно-технических средств для изучения семейства протоколов передачи данных Modbus.

В общем случае обмен информацией между устройствами, которые входят в состав автоматизированной системы, осуществляется посредством промышленной сети [15]. В основе любой промышленной сети будет лежать тот или иной протокол передачи данных.

Протокол передачи данных – это набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Такие соглашения устанавливают единообразный способ передачи сообщений и поиск ошибок при взаимодействии программного обеспечения рассредоточенной в пространстве аппаратуры, которая соединена каким-либо интерфейсом [16].

В работе рассматривается связь между объектами посредством семейства протоколов пере дачи данных Modbus. Протокол передачи данных Modbus был разработан компанией Modicon и опубликован в 1979 году. Протокол являлся открытым стандартом, который описывает формат сообщений, способ передачи этих сообщений в сети, состоящей из разных электронных устройств [32].

В настоящее время рассматриваемый протокол передачи данных используется в различных промышленных сетях, подстраиваясь под специфику сети, а именно, существуют следующие разновидности протокола: Modbus RTU, Modbus ASCII, Modbus TCP [18].

Для реализации всех пунктов технического задания в состав проектируемой установки необходимо включить микроконтроллер, который будет выполнять задачи управления электронными устройствами.

Микроконтроллер – это микросхема, предназначенная для управления электронными устройствами, однокристальный компьютер, сочетающий в себе

функции как процессора, так и периферийных устройств. Микроконтроллер будет принимать данные на вход, сохранять в оперативной или постоянной памяти, обрабатывать данные согласно заданным алгоритмам, передавать данные через

выход, выполнять установленные ему команды с использованием встроенных внутрь таймеров, последовательных интерфейсов, аналоговых преобразователей и прочего. Большой функционал, заключенный в одной микросхеме, позволяет уменьшить целое устройство со сложной принципиальной схемой до размеров единственной микросхемы, что значительно снижает размеры, энергопотребление и стоимость всего устройства, выполненного на базе микроконтроллера [14].

Микроконтроллеры великолепно подходят для мелкосерийного производства, реализации проектов, требующих особой гибкости выполняемых функций и так далее [13].

Целью работы стала разработка комплекса программно-технических средств для изучения семейства протоколов передачи данных Modbus, где управляющим элементом является микроконтроллер.

Для достижения цели необходимо решить следующие задачи:

1) ознакомиться с литературой о протоколах передачи данных;

2) рассмотреть отечественные технологии в рассматриваемой области знаний;

3) изучить структуру и параметры передачи данных в рассматриваемом протоколе передачи данных Modbus;

4) разработать алгоритм работы лабораторной установки и программного обеспечения;

5) реализовать алгоритм работы лабораторного стенда;

6) произвести проверку программного обеспечения.

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ. АКТУАЛЬНОСТЬ РАЗРАБОТКИ

1.1 Анализ технического задания

Темой выпускной квалификационной работы является разработка комплекса программно-технических средств для изучения семейства протоколов передачи данных Modbus. Данный лабораторный стенд необходим для ознакомления студентов с протоколом передачи данных Modbus, а также для приобретения практических навыков в данной области знаний.

Передача данных в условиях промышленности представляет собой огромный интерес и сложность, прежде всего, ввиду объемного набора контролируемых величин. Для корректного съема значений с датчиков температуры, давления и других типов и дальнейшей их передачи сейчас разработано множество протоколов передачи данных, которые позволяют корректно, безошибочно передавать данные благодаря встроенным алгоритмам проверки [27].

Для ознакомления на занятиях с самыми распространенными протоколами передачи данных было получено задание на разработку комплекса программно-технических средств для изучения одного из видов протокола передачи данных, а именно Modbus.

При проектировании разных систем разработчики часто сталкиваются с задачей, которую в реальных условиях создать невозможно. Для этого программно реализуется модель необходимого устройства в необходимых условиях и тестируется. В нашем случае необходимо реализовать имитацию термоконтроллера, другими словами, смоделировать процесс нагрева до определенной температуры и дальнейшее остужение. Для передачи в каждый момент времени данных о значении текущей температуры используется протокол передачи данных Modbus [37].

Для реализации управления электронными элементами, которые можно будет подключить к устройству, будем использовать микроконтроллер. Для него необходимо написать ПО, которое позволит корректно отражать состояние

имитируемого датчика температуры, а протокол передачи данных Modbus в это время будет передавать данные на устройство, где визуально можно будет пронаблюдать за процессом нагрева с помощью временной диаграммы.

На самой установке необходимо поставить светодиоды, чтобы визуально наблюдать степень нагрева имитируемого объекта.

Согласно техническому заданию необходимо разработать устройство, которое должно отвечать следующим требованиям:

1) поддержание режима термоконтроллера;
2) возможность связи с другими устройствами посредством интерфейса rs-485;
3) переключение между протоколами передачи данных modbus rtu и modbus ascii;
4) изменение скорости передачи данных: 9600, 19200, 38400;
5) изменение значения бита паритета: odd, even, no parity;
6) обеспечение режима дискретного ввода.

При разработке устройства для его правильного функционирования важно подобрать необходимые элементы. Это обеспечивается обоснованным выбором компонентов: интерфейс связи с электрическим устройством, микроконтроллер, блок питания, светодиодная панель и другие элементы, входящие в состав установки.

1.2 Обзор имеющихся разработок

В настоящее время имеется множество разработок, а точнее лабораторных стендов, которые позволяют заниматься изучением протокола передачи данных Modbus студентам. В учебной лаборатории ЮУрГУ установлен лабораторный стенд «Интерфейсы RS-485/422 в микроконтроллерных и промышленных сетях», разработанная организацией «Учтех-Профи». [19]

Стенд предназначен для проведения лабораторно-практических работ для студентов высших, средних и профессионально-технических учебных заведений с целью получения знаний, опыта и навыков работы с устройствами в

микроконтроллерных и промышленных сетях по интерфейсам RS485 и RS422. Стенд позволяет повысить опыт в программировании микроконтроллеров, пополнить знания в языках программирования C, C++ и Assembler, на практике изучить физические принципы передачи данных через изучаемые интерфейсы, а также получить знания и опыт работы с протоколами MODBUS ASCII и MODBUS RTU. Во время лабораторных работ изучается влияние различных параметров на передачу данных по интерфейсам и потребление тока линией связи. Во время написания программ для микроконтроллера повышается не только уровень знания языков программирования, но и знания в области работы с периферийными устройствами [19].

На рисунке представлен общий вид лабораторного стенда.

Рисунок 1 – Подключенные стенд к ПК

Рисунок 2 – Общий вид лабораторного стенда

В состав этой установки входит следующее [19]:

1.Моноблок (1 шт.)

1) персональный компьютер;
2) блок микроконтроллера;
3) блок периферии rs-422 и rs-485;
4) блок связи;
5) блок анализаторов.

2.Соединительные, приборные провода (20 шт.).

3.Комплект учебно-методических пособий (2 шт.).

4.Программное обеспечение (1 компакт-диск).

Рассмотрим подробнее каждый из блоков лабораторного стенда.

1 Блок микроконтроллера

Модуль предназначен для изучения микроконтроллера и синхронных/асинхронных приемопередатчиков USART в составе

микропроцессорных систем. Модуль представлен следующими функциональными частями:

1) микроконтроллер atmega32, с программатором для программирования микроконтроллера через интерфейс spi (1 шт.);
2) жидкокристаллический двухстрочный цифробуквенный дисплей (1 шт.);
3) семисегментный индикатор (1 шт.);
4) светодиоды для индикации логических уровней (8 шт.);
5) потенциометр для генерации аналоговых сигналов (1 шт.);
6) генератор импульсов на 50 гц (1 шт.);
7) фильтры низких частот (2 шт.);
8) 10-канальный генератор логических уровней (1 шт.).

2 Блок периферии RS-422 и RS-485

Модуль предназначен для изучения устройств с интерфейсами RS422 и RS-485, с протоколами MODBUS ASCII, MODBUS RTU и другими протоколами. Модуль представлен следующими

функциональными частями:

1) ультразвуковой дальномер с интерфейсом rs-485 (1 шт.);
2) сервопривод с интерфейсом rs-485 и протоколом modbus rtu (1 шт.);
3) датчик освещенности с интерфейсом rs-422 и протоколом modbus ascii (1 шт.);
4) датчик температуры с интерфейсом rs-422 и семисегментным индикатором температуры (1 шт.);
5) резисторы защитного смещения (5 шт.);
6) резисторы согласования линии (4 шт.);
7) блок имитации шины rs-485/rs-422 с индикацией активности линий (1 шт.);
8) блок связи.

Модуль предназначен для реализации интерфейса RS-485 и RS-422 в микропроцессорных системах и промышленных сетях. Модуль представлен следующими функциональными частями [19]:

1) виртуальный com-порт (1 шт.);
2) миллиамперметр (1 шт.);
3) транслятор уровней rs-485-ттл (2 шт.);
4) светодиоды различного цвета (2 шт.);
5) кнопочные переключатели (2 шт.).

3 Блок анализаторов

Модуль предназначен для снятия логических временных диаграмм и для визуализации обмена на шине при помощи специализированного анализатора трафика RS-485/RS-422. Модуль представлен следующими функциональными частями [19]:

1) 16-и канальный логический анализатор (1 шт.);
2) цифровой двухканальный осциллограф (1 шт.);
3) транслятор уровней rs-485-ттл (2 шт.);
4) анализатор трафика rs-485/rs-422 (1 шт.).

Все блоки стенда можно коммутировать между собой приборными проводами.

Приведем основные характеристики рассматриваемого стенда в таблице

Таблица 1 – Характеристики лабораторного стенда

Напряжение питания, В	220
Частота питающего напряжения, Гц	50
Диапазон рабочих температур, ℃	+10…+40
Габаритные размеры, ШхВхГ, мм	410 х 357 х 168
Количество мест для обучаемых	3

Распространение стенда осуществляется с помощью интернета, то есть его можно найти на сайте «Учтех-Профи», где и приведены все характеристики

лабораторной установки. Стоимость представленного стенда составляет 223180 рублей.

Огромным упущением разработчиков стало следующее: каждый из элементов в данном модуле связывается с устройствами по разным протоколам передачи данных, что затрудняет работу студентов с данным стендом.

Рассмотрев имеющиеся разработки для изучения протокола передачи данных Modbus, а также изучив необходимую документацию и ознакомившись с основными понятиями промышленных сетей, ещё раз убедились в актуальности и нужности разработки комплекса ПТК для изучения семейства протоколов передачи данных Modbus. Разрабатываемый стенд должен быть более компактным в своих размерах, а также выполнять основные задачи, которые могут быть поставлены перед студентами, учитывать стоимость элементов (сделать более дешевую версию). Лабораторная установка должна соответствовать всем требованиям государственного стандарта, быть гибкой к выполнению ряда специфических задач, например: поддерживать режим имитатора термоконтроллера.

2 ОПИСАНИЕ УСТАНОВКИ

2.1 Основные понятия о протоколе передачи данных Modbus

Для передачи данных от проектируемого устройства к другим электронным устройствам, согласно техническому заданию, используем протокол передачи данных Modbus.

Modbus – это открытый коммуникационный протокол, который основан на архитектуре ведущий-ведомый. Он применяется в промышленности, а точнее для организации связи между электронными устройствами. Этот протокол используется для передачи данных через последовательные линии связи, а точнее через следующие интерфейсы: RS-485, RS-422, RS-232, а также через Ethernet в сети TCP/IP [45].

Одним из преимуществ Modbus является отсутствие необходимости в специальных контроллерах, что существенно снижает затраты разработчиков контроллерного оборудования и других системных стандартов. Открытость протокола обеспечивается бесплатными текстами стандартов, которые можно скачать [35].

Популярность протокола обосновывается тем, что он совместим с огромным количеством оборудования. Несмотря на открытость, Modbus осуществляет высокую достоверность передачи данных, которая прежде всего зависит от функции контроля ошибок. Так же удобство данного протокола заключается в унификации команд обмена, которые представлены в виде стандартных регистров и функций их чтения-записи [46].

В данном протоколе принята следующая специфическая терминология. Ознакомимся с ней, приведя необходимые определения.

1 PDU (Protocol Data Unit) – код функции и данные пакета. Это общая для всех физических уровней часть пакета Modbus [27].

2 ADU (Application Data Unit) – полный пакет Modbus, который включает в себя спецификацию для физического уровня, часть пакета и PDU [27].

В протоколе передачи данных Modbus выделяют четыре типа данных:

1 Discrete Inputs – это однобитный тип данных, который доступен только для чтения [27].

2 Coils – это однобитный тип данных, который доступен для чтения и записи данных [27].

3 Input Registers – шестнадцатибитовый знаковый или беззнаковый тип данных, который доступен только для чтения [31].

4 Holding Registers – шестнадцатибитовый знаковый или беззнаковый тип данных, который доступен для записи и чтения [44].

В рассматриваемом стандарте Modbus введены следующие требования: обязательные и опционные. Существует также три степени соответствия стандарту:

1) полностью соответствует – протокол соответствует всем обязательным и всем опционным требованиям;

2) условно соответствует – протокол соответствует только обязательным требованиям и не соответствует опционным;

3) не соответствует [42].

Разработана специальная сетевая модель OSI (open systems interconnection basic reference model) [27].

Сетевая модель OSI – сетевая модель стека сетевых протоколов. На основе данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель включает в себя различные уровни взаимодействия систем: физический, канальный, сетевой, транспортный, сеансовый, уровень представления, прикладной. Каждый уровень, в свою очередь, выполняет определенные функции [33].

Модель OSI для протокола Modbus выглядит следующим образом.

Таблица 2 – Модель OSI для Modbus

Номер уровня	Название уровня	Реализация
7	Прикладной	Modbus Application Protocol
6	Уровень представлений	-
5	Сеансовый	-
4	Транспортный	-
3	Сетевой	-
2	Канальный	Протокол «master-slave» Режимы RTU и ASCII
1	Физический	RS-485, RS-232

Протокол Modbus предполагает, что может быть только одно ведущее устройство, то есть контроллер, а также до 247 ведомых устройств, то есть модулей ввода вывода. Все эти устройства могут быть объединены в компьютерную сеть [31].

Передача данных может быть инициирована только ведущим устройством, причем различают два вида передачи данных [38]:

1) всем устройствам сети (широковещательный режим – это режим, когда сообщение посылается одновременно всем устройствам);

2) только одному устройству.

Для удобства, широковещательному режиму зарезервирован адрес «0», то есть если в команде используется это число, то сообщение принимается всеми устройствами.

Modbus RTU (Remote Terminal Unit) – это коммуникационный протокол, который основан на архитектуре ведущий-ведомый (то есть master-slave). Он используется для передачи данных с использованием интерфейсов RS-485, RS-422, RS-232, а также через Ethernet в сети TCP/IP [31].

В режиме Modbus RTU существует особый вид передачи сообщений, называемый кадром. Он имеет следующий вид.

Рисунок 3 – Формат кадра для режима Modbus RTU

Сообщение начинает восприниматься как новое после определенной паузы длительностью 3,5 символа (чтобы рассчитать величину паузы в секундах необходимо знать скорость передачи данных) [34].

Поле адреса фрейма всегда содержит только адрес ведомого устройства, даже в ответах на команду. Поэтому master-устройство всегда знает, от какого модуля пришел ответ.

Поле «Код функции» содержит информацию о действии, которое необходимо выполнить [32].

Поле «Данные» может содержать произвольное количество байт. В состав этого поля может входить информация о параметрах, которые используются в запросах контроллера или ответах модуля [38].

Поле «Контрольная сумма» включает в себя контрольную сумму CRC длинной 2 байта.

Modbus RTU предусматривает использование 8 бит данных в 11-битовом символе, который позволят передавать по байту на символ. Формат символа в данном режиме представляется следующим образом: 1 стартовый бит, 8 бит данных, причем младший бит передается первым, 1 бит паритета, 1 стоповый бит [16].

Modbus ASCII – это спецификация сетевого протокола обмена данных для обеспечения сетевой связи между интеллектуальными устройствами. Данный

режим использует шинную топологию, с использованием сетевых интерфейсов RS-232 и RS-485 [32].

В режиме Modbus ASCII каждый байт сообщения передается как два ASCII символа, а точнее их шестнадцатеричное представление. Значение байта будет передаваться как ASCII-код. Другими словами, байты данных, код функции и байт поля проверки будут передаваться символами 0-9 и A-F.

Формат символа в ASCII-режиме выглядит следующим образом: 1 стартовый бит, 7 бит данных, 1 бит паритета, 1 стоповый бит.

Представим на рисунке формат кадра сообщения.

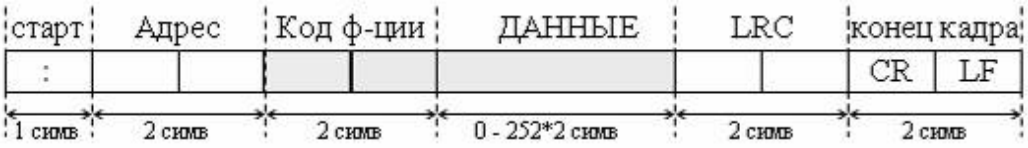

Рисунок 4 – Формат кадра для ASCII режима обмена

Для разграничения кадров используется стартовый символ «:», а конец кадра обозначается стоповой последовательностью «CR LF». Для контроля ошибок введен LRC алгоритм, то есть подсчет контрольной суммы, которая рассчитывается над всеми байтами кадра, кроме стартовой и стоповой последовательности [31].

В свою очередь, режим Modbus ASCII накладывает меньшие требования на оборудование за счет использования стартовой и стоповой последовательности, нечувствительности к значительным паузам между символами.

2.2 Интерфейс для подключения к электронным устройствам

Как было сказано ранее для передачи данных на физическом уровне будем использовать интерфейс TIA/EIA-485 (далее будем использовать название RS-485, которое будет более удобно в употреблении) Данный интерфейс приобрел

большую популярность и стал основой для создания огромного количества промышленных сетей [20].

В стандарте RS-485 для передачи и приёма используется одна витая пара проводов, которая сопровождается оплеткой или общим проводом [19].

В основе построения RS-485 лежит дифференциальный способ передачи сигнала. Он состоит в следующем: когда напряжение, соответствующее уровню логической единицы или нуля, отсчитывается не от «земли», а измеряется как разность потенциалов между двумя передающими линиями: DATA+ и DATA- . При этом напряжение каждой линии относительно «земли» может быть произвольным, но не должно выходить за диапазон -7…+12 В [28].

Приемники сигнала являются дифференциальными, то есть воспринимают только разность между напряжениями на линии DATA+ и DATA- . При разности напряжение более 200 мВ, до +12 В считается, что на линии установлено значение логической единицы. При напряжении менее -200 мВ, до -7 В установлено значение логического нуля. Дифференциальное напряжение на выходе передатчика в соответствии со стандартом должно быть не менее 1,5 В, поэтому при пороге срабатывания приемника 200 мВ помеха может иметь размах 1,3 В над уровнем 200 мВ. Такой большой запас необходим для работы на длинных линиях с большим омическим сопротивлением. Физически, именно этот запас по напряжению и определяет максимальную длину линии связи при низких частотах. Для стандарта RS-485 максимальная длина линии составляет 1200м, а скорость передачи до 10 Мбит/с [20].

Благодаря симметрии линий относительно «земли» в них наводятся помехи, близкие по форме и величине. В приемнике с дифференциальным входом сигнал выделяется путем вычитания напряжений на линиях, поэтому после вычитания напряжение помехи оказывается равным нулю [28].

Для минимизации чувствительности линии передачи к электронной наводке используется витая пара проводов [19].

Перечислим основные свойства интерфейса [30]:

1) формирователь обеспечивает тристабильный режим сети: 1, 0 и высокоомное состояние;

2) каждый узел сети имеет уникальный адрес для исключения коллизий;

3) удобство для подключения нескольких датчиков к одному ПЛК с топологией «шина»;

4) возможна 2-х и 4-х проводная система подключения узлов.

2.3 Управляющее устройство

Для реализации устройства необходимо определиться с аппаратной частью. Основные функции управления установкой будет выполнять микроконтроллер. Зададим следующие критерии, которые должно реализовывать разрабатываемое устройство [41]:

1) поддержание взаимодействия с устройствами сети по интерфейсу двухпроводному RS-485;

2) имитация режима работы термоконтроллера;

3) реализация протоколы передачи данных: MODBUS RTU и MODBUS ASCII, а также возможность переключения между ними;

4) для протоколов передачи данных нужно обеспечить изменения бита паритета;

5) обеспечение изменение скорости передачи данных в режиме UART.

В настоящее время на рынке представлено огромное количество модулей для программирования [1].

В данный момент в имеющейся элементной базе представлены следующие микроконтроллеры: PIC16F684, STM32F303VC, ATMega328. Рассмотрим основные характеристики представленных микроконтроллеров и выберем наиболее подходящий [14].

1) Микроконтроллер Pic Microchip [52]

Рисунок 5 – Внешний вид микроконтроллера Pic Microchip

Рассмотрим микроконтроллер PIC16F684, который удовлетворяет требованиям технического задания.

Перечислим основные особенности микроконтроллера:

1. flash-память программ объемом 256 кбайт;
2. оперативная память объемом 128;
3. количество выводов/портов 14/12;
4. встроен аналоговый компаратор;
5. встроен 10-разрядный ацп;
6. встроен модуль захвата/сравнения;
7. встроен модуль шим
8. поддерживается режим энергосбережения sleep.

2) Микроконтроллер STM32 [51]

Рассмотрим микроконтроллер STM32F303VC, который удовлетворяет требованиям технического задания.

Рисунок 6 – Внешний вид микроконтроллера STM32

Перечислим основные характеристики микроконтроллера STM32F303VC:

1. flash-память программ объемом 256 кбайт;
2. оперативная память объемом 48 кбайт;
3. ядро arm cortex-m4;
4. модель защиты памяти;
5. содержит до 13 таймеров;
6. 4 модуля ацп;
7. 7 аналоговых компараторов.

3) Микроконтроллер фирмы AVR [14]

Микроконтроллеры AVR фирмы Atmel семейства Mega предназначены для использования во встраиваемых приложениях. Они изготавливаются по малопотребляющей КМОП-технологии, которая позволяет достичь наилучших соотношений в стоимости, быстродействии и энергопотреблении.

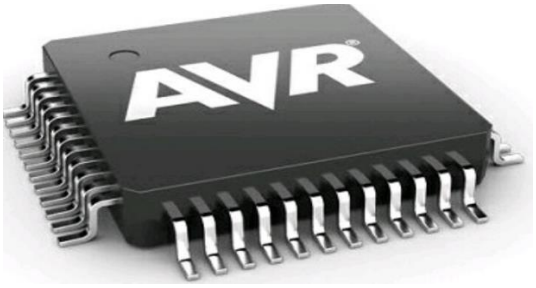

Рисунок 7 – Внешний вид микроконтроллера AVR

Рассмотрим микроконтроллер ATMega328, который удовлетворяет требованиям технического задания.

Перечислим основные особенности микроконтроллера:

1. flash-память программ объемом 32 кбайт;
2. оперативная память объемом 2 кбайт;
3. память данных на основе eeprom объемом 1кбайт;
4. возможность защиты от чтения и модификации памяти программ и данных;
5. возможность программирования непосредственно через последовательные интерфейсы;
6. возможность самопрограммирования;
7. разнообразные способы синхронизации;
8. наличие нескольких режимов пониженного энергопотребления.

Выбирая из этих трех микроконтроллеров, остановимся на микроконтроллере ATMega 328, который по всем характеристикам удовлетворяет требованиям технического задания, а также к его преимуществам следует отнести доступность и приемлемую цену.

Далее подробнее рассмотрим архитектуру, структуру, свойства выбранного микроконтроллера.

Процессор микроконтроллера обладает следующими свойствами: полностью статическая архитектура, минимальная тактовая частота равна нулю; АЛУ подключено к регистрам общего назначения; большинство команд выполняются за один период тактового сигнала; векторная система прерываний, а также поддержка очереди прерываний [16].

Подсистема ввода/вывода поддерживает следующие свойства [14]:

1) программное конфигурирование и выбор портов ввода/вывода;
2) выводы микроконтроллера могут быть запрограммированы как входные или как выходные независимо друг от друга;
3) выходные буферы содержат триггер шмидта на всех выводах;

4) на всех входах имеется индивидуально отключаемые внутренние подтягивающие резисторы сопротивлением 20…50 ком.

Микроконтроллера ATMega поддерживают подключение разнообразных периферийных устройств, в том числе и тех, которые необходимо будет подключить нам [41].

Ядро микроконтроллера ATMega имеет следующую архитектуру, представленную на рисунке [14].

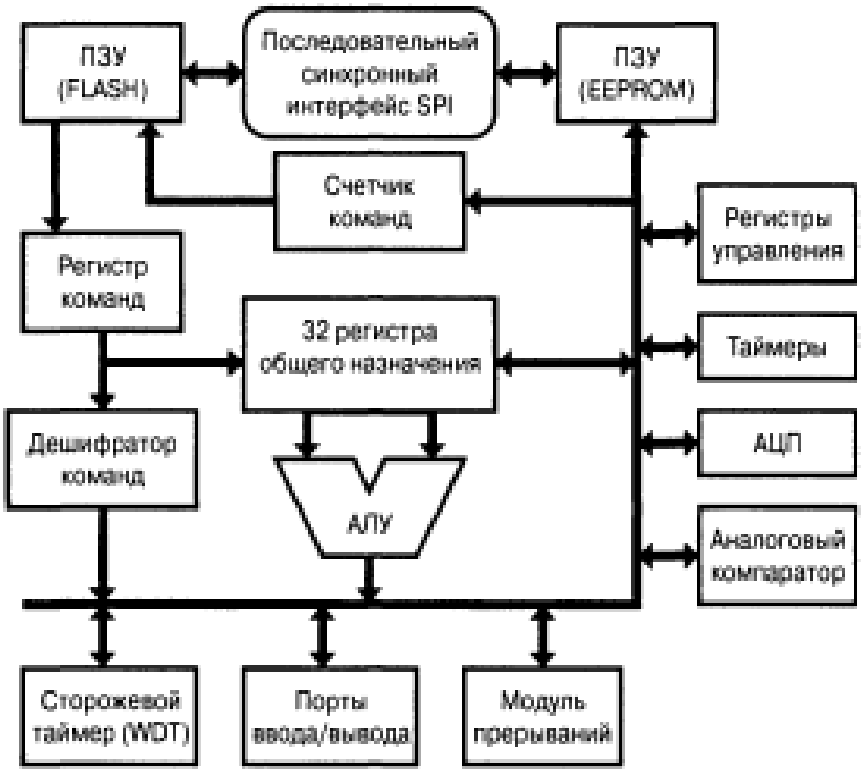

Рисунок 8 – Архитектура ядра микроконтроллера ATMega

В этом ядре АЛУ подключено к 32 рабочим регистрам, которые объединены в регистровый файл. Практические каждая команда занимает одну ячейку памяти программ [38].

В данном типе микроконтроллеров организована Гарвардская архитектура, которая характеризуется разделением памяти программ и данных. Такая

организация архитектуры позволяет одновременно работать с памятью программ и данных [24].

Для моделирования устройства будем использовать микроконтроллер ATMega 32. Перечислим некоторые его характеристики: имеет FLASH-память программ 32Кбайта, ОЗУ объемом 2 Кбайта, EEPROM-память 1 Кбайт [14].

На рисунке представлено расположение выводов выбранного микроконтроллера [41].

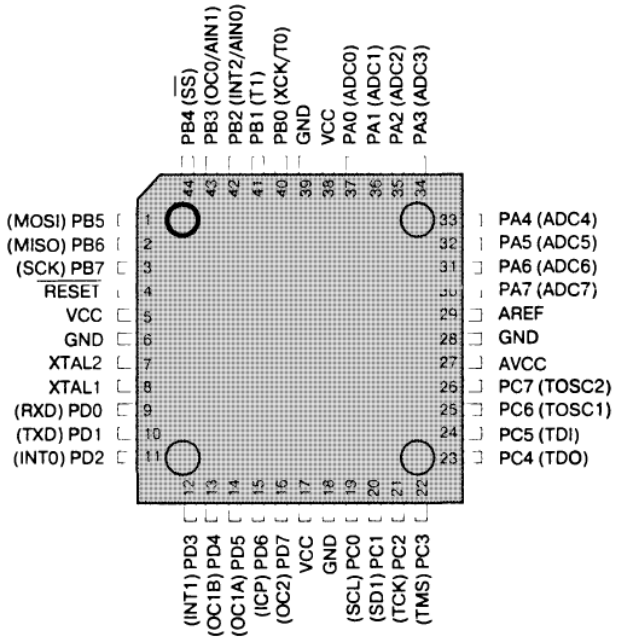

Рисунок 9 – Расположение выводов микроконтроллера

Отметим, что набор периферийных устройств достаточно разнообразен. Следует выделить особенности модели ATMega 328 [14]:

1) четыре 8-битных порта ввода/вывода;
2) два 8-битных (т0 и т2) и один 16-битный (т1) таймер/счетчик;
3) 4 канала шим;
4) по одному интерфейсу usart, spi и twi;
5) 8-канальный 10-битный ацп;
6) интерфейс jtag.

Приведем структурную схему микроконтроллера.

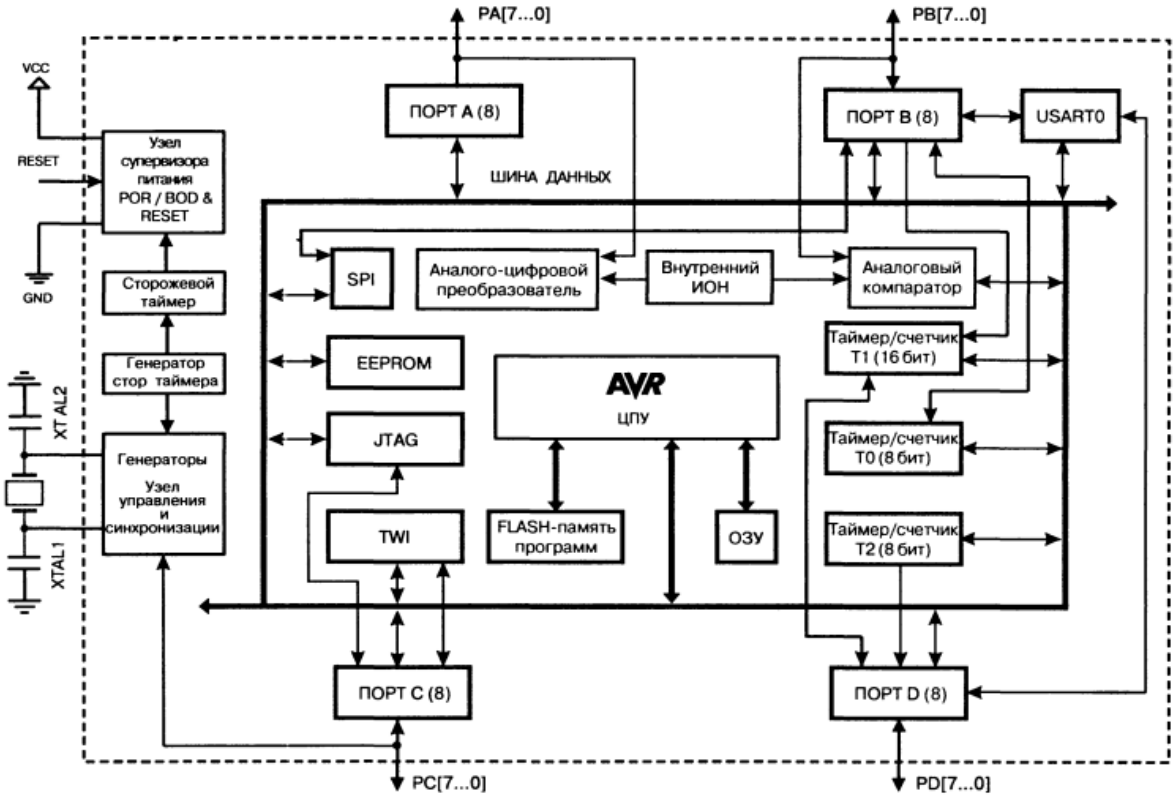

Рисунок 10 – Структурная схема микроконтроллера

Далее рассмотрим систему команд микроконтроллера. Она весьма разнообразна. Несмотря на то, что микроконтроллер имеет RISC-архитектуру, набор реализованных инструкций, а также разнообразие близко к микроконтроллерам с CISC-архитектурой. Почти каждая команда занимает одну ячейку памяти программ [17].

Все множество команд микроконтроллера AVR разделяют на некоторые группы [26]:

1) команды логических операций;
2) команды арифметических операций и команды сдвига;
3) команды операций с битами;
4) команды пересылки данных;
5) команды передачи управления;
6) команды управления системой.

3 ЭЛЕМЕНТНАЯ БАЗА ЛАБОРАТОРНОЙ УСТАНОВКИ ДЛЯ ИЗУЧЕНИЯ СЕМЕЙСТВА ПРОТОКОЛОВ ПЕРЕДАЧИ ДАННЫХ MODBUS

3.1 Обоснование и выбор элементной базы устройства

Перейдем к реализации устройства на плате. Для этого необходимо привести перечень необходимых элементов.

В пункте 2.2 произведен выбор микроконтроллера, который по всем параметрам удовлетворяет требованиям технического задания. В работе будем использовать микроконтроллер ATMega 32, который абсолютно идентичен выбранному выше микроконтроллеру. Данная замена аргументируется прежде всего тем, что микроконтроллер ATMega 32 уже имеется на руках [14].

Приведем основные характеристики микроконтроллера серии ATMega 32 в таблице.

Таблица 3 – Характеристики микроконтроллера ATMega 32

Название	ATMega 32
Объем Flash-памяти программ, кБ	32
Объем памяти данных (ОЗУ, SRAM), кБ	2
Энергонезависимая память данных (EEPROM), кБ	1
Последовательный интерфейс SPI	+
Напряжение питания, В	4.0…5.5
Максимальный ток порта, мА	40
Максимальный ток по выводам VCC и GND, мА	200
Количество входов/выходов, шт.	32
Тактовая частота процессора, МГц	0 – 16

При разработке устройства необходимо учесть возможность внутрисхемного программирования [14]. Для этого необходимо на схеме предусмотреть разъем для подключения программатора. Из массы представленных разъемов выберем BH-06 для интерфейса SPI, который полностью удовлетворяет всем параметрическим и электрическим характеристикам, требуемым для реализации. На рисунке приведен внешний вид необходимого разъема [44].

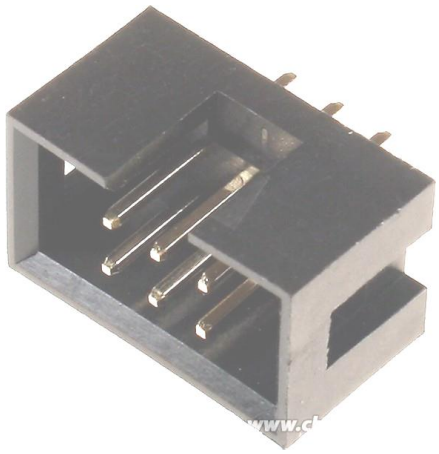

Рисунок 11 – Внешний вид разъема BH-06 для интерфейса SPI

Приведем характеристики вилки разъема в таблице.

Таблица 4 – Вилка на плату DS1013-06S [44]

Название	BH-06 (IDC-06MS) (DS1013-06S)
Тип разъема	вилка на плату, прямая, 6 контактов с шагом 2.54 мм
Предельный ток, А	1 А
Предельное напряжение	500 В в течение 1 минуты
Диапазон температур, ˚С	-40…+105
Обозначение на схеме	ST VP
Количество, шт.	1

Для данного устройства необходимо обеспечить питание 4,0…5,5В. Для этого следует продумать схему питания особым образом. От источника питания на микросхему прямо поступает 24В. Таким образом используем схему делителя напряжения. Используем схему из одного предохранителя PTF76, диода SK36A, двух керамических конденсаторов SM/C_0805 и двух электролитических алюминиевых конденсаторов SR-50-100, резистора SM/R_0805 номиналом 1кОм, катушка индуктивности CW68, а также схема импульсного регулятора напряжения LM2575S. [14].

Приведем характеристики выбранных элементов.

Таблица 5 – Характеристика керамических конденсаторов SM/C_0805

Название	SM/C_0805
Номинальная ёмкость, мкФ	0.1
Рабочее напряжение, В	25
Диапазон температур, ˚С	-25…+85
Количество, шт.	2

Таблица 6 – Характеристика электролитических конденсаторов SR-50-100

Название	SR-50-100
Номинальная ёмкость, мкФ	100
Рабочее напряжение, В	25
Диапазон температур, ˚С	-40…+85
Количество, шт.	2

Регулятор напряжения LM2575S имеет максимальное входное напряжение 40 В, а выходное 4,0…5,5 В, выходной ток 5 мА. Такой элемент полностью удовлетворяет требованиям к питанию микроконтроллера.

Таблица 7 – Характеристики стабилизатора

Название	LM2575S
Максимальное входное напряжение,В	40
Выходное напряжение, В	5 ± 0.25
Выходной ток, мА	до 5
Диапазон температур, ˚С	-40 … +125
Количество, шт.	1

На рисунке представим стабилизатор LM2575S.

Рисунок 12 – Внешний вид регулятора импульсного напряжения LM2575S

А также представим схему подключения регулятора импульсного напряжения.

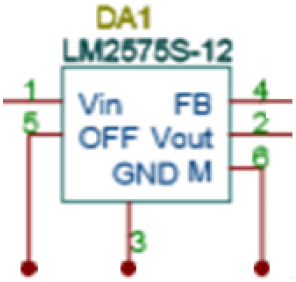

Рисунок 13 – Схема импульсного напряжения с обозначением выводов

Таблица 8 – Характеристики резисторов SM/C_0805

Название	SM/C_0805
Номинал сопротивления, кОм	1
Рассеиваемая мощность, Вт	0.125
Количество, шт	1

Катушка индуктивности необходима для ограничения высокочастотного тока, поступающего в схему импульсного напряжения. Выберем катушку индуктивности CW68, которая по всем характеристикам удовлетворяет требованиям технического задания.

Таблица 9 – Характеристики катушки индуктивности CW68

Название	CW68
Номинал индуктивности, мкГн	100, 5%
Рассеиваемая мощность, Вт	0.125
Максимальный ток, мА	710
Количество, шт	1

Для ограничения обратного тока, который может возникнуть в цепи, необходим диод. Для данной схемы выберем диод SK36A, который полностью подходит по электрическим параметрам, а также полностью удовлетворяет требования технического задания. В таблице приведем некоторые характеристики.

Таблица 10 – Характеристики диода SK36A

Название	SK36A
Прямой ток, А	3
Прямое напряжение, В	60
Обратный ток, мкА	600
Обратное напряжение, В	0,75
Рабочая температура, ℃	125
Количество, шт	1

Подключение питания к устройству будет осуществляться через специальное гнездо питания, расположенное на схеме.

Таблица 11 – Гнездо питания с клеммами DS-201 [6]

Название	DS-201
Тип разъема	Jack 5.5
Длина контактной части, мм	9
Обозначение на схеме	X1
Количество, шт.	1

Для индикации контролируемых величин установим панель, содержащую светодиоды.

Для этой задачи отлично подходит панелька RGB V2 Keyes 5050. Выбранный модуль содержит в своем составе 8 светодиодов, которые, при необходимости и в зависимости от конфигурации, могут загораться тремя цветами: красным, зеленым, синим.

Для управления выбранной панелью конфигурируют управляющие биты, которые соответствуют выбранному светодиоду. То есть в зависимости от введенного бита в разных ситуациях будет загораться светодиод определенного и выбранного цвета

Такой модуль полностью совместим с выбранным микроконтроллером AVR, а также удовлетворяет всем требованиям технического задания.

Таблица 12 – Характеристики модуля RGB V2 Keyes 5050

Название	модуль RGB V2 Keyes 5050
Тип светодиодов	RGB 5050
Количество светодиодов	8
Напряжение питания, В	5
Количество, шт.	1

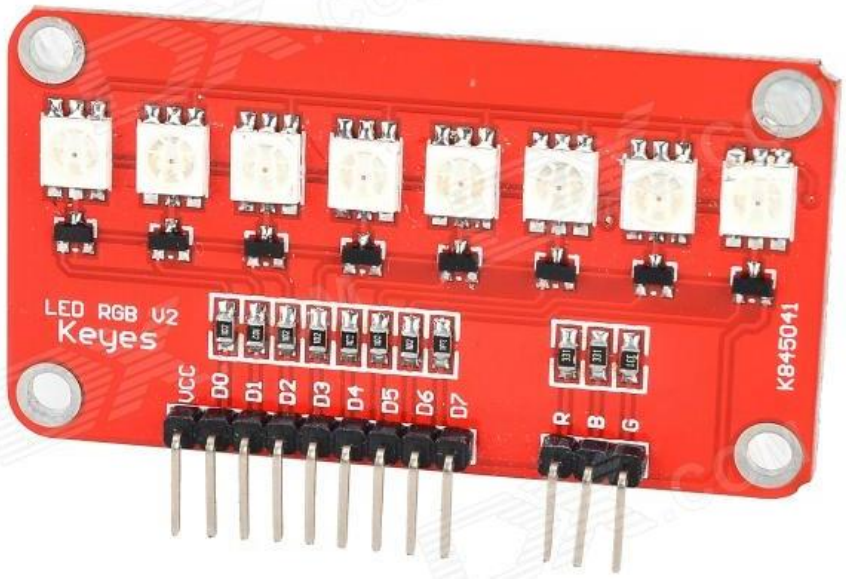

Рисунок 14 – Внешний вид модуля RGB V2 Keyes 5050

Для управления светодиодами потребуются набор защитных резисторов, номинал которых найдем как:

$$R = \frac{U_{пит} - U_{раб}}{I_{пр}}, \quad (1)$$

где R – номинал защитного резистора;

$U_{пит}$ – максимальное напряжение на выходе контроллера;

$I_{пр}$ – прямой диодный ток;

$U_{раб}$ – рабочее напряжение.

Определим рабочее напряжение по диаграмме из спецификации на индикатор [14]:

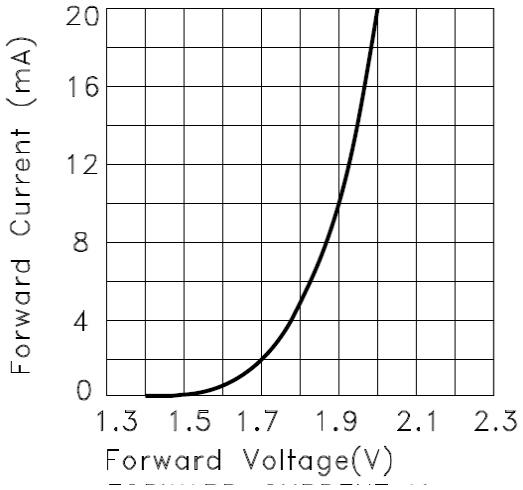

Рисунок 15 – Диаграмма «прямой ток – прямое напряжение»

Таким образом, для номинального прямого тока 10 мА, номинальное рабочее напряжение равно 1.9 В. Рассчитаем по формуле (1):

$$R = \frac{5\text{-}1.9}{0.010} = 310 \text{ (Ом)}.$$

Из ряда E24 выберем резистор номиналом 330 Ом.

Таблица 13 – Характеристики резисторов SM/C_0805 360 Ом

Название	SM/C_0805
Номинал сопротивления, Ом	360
Рассеиваемая мощность, Вт	0.125
Диапазон температур, ˚C	-60 … +125
Срок работы, тыс. часов	25
Количество, шт.	8

Управление переключения светодиодов осуществляется при помощи биполярных транзисторов. Главное требование к выбору транзисторов является выполнение условий по протекающему току: ток коллектора должен быть равным прямому току светодиода.

Для данной схемы прямой ток будет равен 0,01 А. Поэтому выбор транзистора будем выполнять исходя из этого условия.

Выберем транзистор 2TY с максимальным током коллектора 50 мА.

Таблица 14 – Характеристики транзисторов 2TY

Название	2TY
Структура	p-n-p
Максимально допустимое напряжение коллектор-база, В	40
Максимально допустимое напряжение коллектор-эмиттер, В	25
Максимально допустимый постоянный (импульсный) ток коллектор, мА	50 (500300)
Обратный ток коллектора, мкА	до 0.05
Статический коэффициент передачи тока в схеме с общим эмиттером	100-250
Количество, шт.	8

Между выходом микроконтроллера и базой транзистора необходимо установить защитный резистор для правильного функционирования светодиодов.

Произведем расчет защитного резистора:

$$R_з = \frac{U_{к\ вых}}{I_б} = \frac{5}{0{,}00005} = 100\ (кОм). \quad (2)$$

Выберем из ряда E24 резистор номиналом 100 кОм [5]. Для схемы выберем резистор SM/C_0805 номиналом 100 кОм.

Выбранные резистор полностью удовлетворяет требованиям по питанию схемы, а также соответствует требованиям технического задания.

Таблица 15 – Характеристики резисторов SM/C_0805 100 кОм

Название	SM/C_0805
Номинал сопротивления, кОм	100
Рассеиваемая мощность, Вт	0.125
Количество, шт.	8

Определимся с аппаратной реализацией интерфейса RS-485. Аппаратная реализация основывается на микросхеме приемопередатчиков с дифференциальными вводами/выводами, которые подключаются к линии с цифровыми портами, подключаемыми к портам UART микроконтроллера. Приведем структурную схему этого элемента [20].

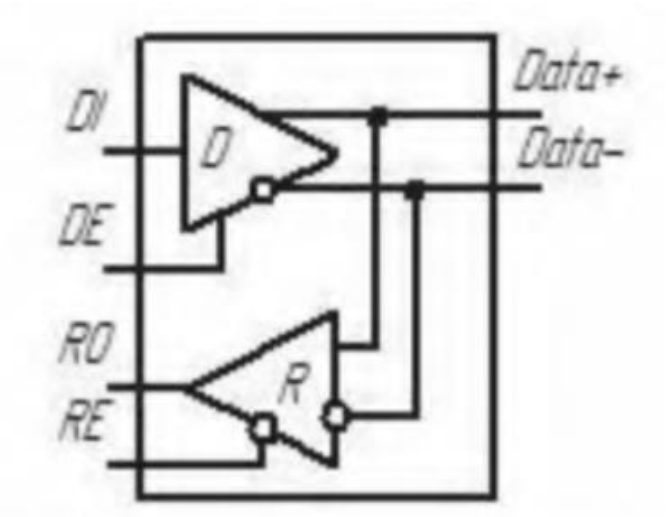

Рисунок 16 – Структурная схема приемопередатчика интерфейса RS-485

Поясним обозначения на рисунке.

D – передатчик,

R – приемник,

DI – цифровой вход передатчика,

RO – цифровой выход приемника,

DE – разрешение работы передатчика,

RE – разрешение работы приемника.

Приведем схему включения интерфейса RS-485 к UART микроконтроллера на рисунке.

Для корректной работы интерфейса RS-485 с микроконтроллером цифровой выход приемника (RO) подключается к порту приемника UART, который обозначается буквами TX, а цифровой вход контроллера (DI) подключается к порту передатчика UART, который обозначается буквами RX. Так как на дифференциальной стороне передатчик и приемник соединены, то следует иметь в виду то, что во время приема следует отключать передатчик, а во время передачи, соответственно, отключать приемник. Именно с этой целью разработаны управляющие входы: разрешение приема данных (RE), разрешение передачи данных (DE). Вход RE инверсный, поэтому его подключают прямо к DE, причем переключение приемника и передатчика происходит одним сигналом с любого выбранного порта микроконтроллера. Условились к следующим обозначениям:

1) логический ноль – прием данных;
2) логическая единица – передача данных.

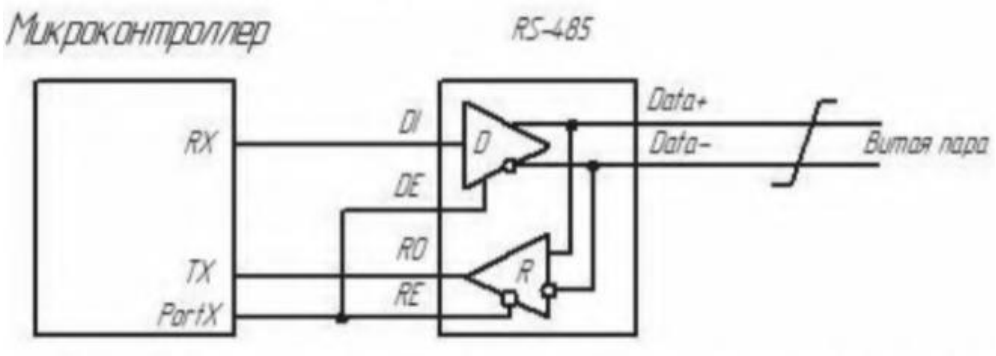

Рисунок 17 – Схема подключения RS-485 к UART микроконтроллера

4 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ УСТРОЙСТВА

4.1 Алгоритм работы разрабатываемого программного обеспечения

Составим алгоритм работы программного обеспечения для лабораторной установки для изучения семейства протоколов передачи данных Modbus.

Программа должна начинаться с описания переменных, используемых в коде. Также и будет начинаться проектируемое ПО.

Сначала происходит описание переменный, используемых в программе. Далее необходимо произвести инициализацию портов ввода/вывода.

Произведем инициализацию протокола передачи данных Modbus, а далее произведем его запуск. Эти действия организуем с помощью специальных библиотек.

После произведем запуск таймера T0, в котором будет инициализирована функция, которая в полной мере опишет процессы, происходящие на имитируемом датчике температуры.

Произведем запись исходных данных в регистры хранения, а также в регистры ввода.

Проинициализировав необходимые для обмена по протоколу передачи данных Modbus, приступим к основному циклу программы.

Изначально произведем проверку параметров, записанных в регистрах. Если обнаружены ошибки, то необходимо вывести их коды на идентификатор, если ошибок не обнаружено, то продолжаем работу дальше.

Произведем расчет значение текущей температуры и текущего времени. Далее произведем обработку прерываний по переполнению счетчика T0.

Проверив прерывания по переполнению, переходим к вводу слова управления, которое зададим с помощью цифр:

1) 0 – исходное состояние;

2) 1 – сброс ошибок;

3) 2 – запуск моделирования.

После выполнения необходимого количества пунктов переходим к завершению программы.

4.2 Описание используемых библиотек

Для работоспособности проектируемого модуля необходимо запрограммировать контроллер согласно техническому заданию.

Для связи установки с другими электронными устройствами, как говорилось ранее, используется протокол передачи данных Modbus. Для функционирования этого протокола передачи данных существуют определенные библиотеки, в которых содержатся коды стандартных функций для реализации протокола передачи данных на любом устройстве [49].

Существует несколько вариантов выбора библиотек: написать самому необходимые для программирования библиотеки, выбрать библиотеку из ранее созданных. Чтобы не выполнять работу два раза, так как сроки выполнения задания ограничены, то будет правильнее взять уже разработанное ПО, а в него уже добавлять новые конфигурации, исходя их технического задания.

Рассмотрим варианты уже разработанных библиотек для реализации slave-устройства для Modbus.

1) Библиотека EasyModbus представляет собой быстрый и безопасный доступ к ПК или встроенных систем во многие ПЛК-системы, а также другие компоненты для автоматизации промышленности. Для чтения или записи данных из или в ПЛК требуется всего несколько строк кода, что делает данную библиотеку понятной и простой в использовании [48].

С помощью рассматриваемой библиотеки можно реализовать Modbus slave для микроконтроллеров ATMEL ATMega, что полностью удовлетворяет требованиям технического задания.

2) Библиотека nModbus представляет собой большую библиотеку, включающую в себя реализацию всех режимов работы с протоколом Modbus.

Модель реализации классов этой библиотеки позволяет работать с любым типом Modbus-устройством. Данная библиотека является открытым ресурсом, что делает её простой в распространении [50].

Рассматриваемая библиотека полностью удовлетворяет требованиям настоящего технического задания, то есть её можно использовать как реализацию slave-устройства в работе.

3) Библиотека FreeModbus – это реализация ведомого устройства (slave). Данная библиотека поддерживает необходимые режимы, требуемые в техническом задании: Modbus RTU и Modbus ASCII [48] . Огромным плюсом данной библиотеки является то, что она находится в свободном доступе для пользователей, проста в использовании и адаптируется под ряд задач, которые необходимо реализовать в ходе ВКР.

Рассмотрев ряд библиотек, которые позволяют реализовать slave-устройство с использованием протокола передачи данных Modbus, выбираем библиотеку FreeModbus, так как она наиболее широко охватывает перечень задач, которые необходимо решить в ходе работы.

Воспользуемся библиотекой FreeModbus, которая доступна для скачивания и представляет собой бесплатное ПО.

Для выбранного микроконтроллера ATMega 32 необходимые библиотеки находятся в папке AVR.

Продемонстрируем её составляющие.

В библиотеке представлены файлы, описывающие алгоритм режима работы протокола передачи данных: Modbus RTU и Modbus ASCII.

Файлы mbascii.c и mbascii.h содержат конфигурацию режима Modbus ASCII, задается фрейм сообщения. Приведем листинг программы [49].

Листинг 1

```
/* Первым байтом в сообщении задается адрес ведомого устройства. */
        pucSndBufferCur = ( UCHAR * ) pucFrame - 1;
        usSndBufferCount = 1;
 /* Формируется новая копия Modbus-PDU в Modbus-Serial-Line-PDU. */
        pucSndBufferCur[MB_SER_PDU_ADDR_OFF] = ucSlaveAddress;
        usSndBufferCount += usLength;
```

```
    /* Рассчитывается LRC контрольная сумма для Modbus-Serial-Line-PDU.*/
            usLRC = prvucMBLRC( ( UCHAR * ) pucSndBufferCur,
usSndBufferCount );
            ucASCIIBuf[usSndBufferCount++] = usLRC;
        /* Передача данных включена */
            eSndState = STATE_TX_START;
            vMBPortSerialEnable( FALSE, TRUE );
```

А также прописываются параметры режима передачи данных.

Файлы mbrtu.c и mbrtu.h содержат конфигурацию режима Modbus RTU, задается фрейм сообщения. Приведем листинг программы [49].

Листинг 2

```
    /* Длина сообщения и CRC контрольной суммы */
        if( ( usRcvBufferPos >= MB_SER_PDU_SIZE_MIN )
            && ( usMBCRC16( ( UCHAR * ) ucRTUBuf, usRcvBufferPos ) ==
0 ) )
        {
            *pucRcvAddress = ucRTUBuf[MB_SER_PDU_ADDR_OFF];
        /* Общая длина Modbus-PDU состоит из Modbus-Serial-Line-PDU за вычетом поля
адреса и поля CRC контрольной суммы */
            *pusLength = ( USHORT )( usRcvBufferPos -
MB_SER_PDU_PDU_OFF - MB_SER_PDU_SIZE_CRC );
        /* Возвращается начальный Modbus PDU */
            *pucFrame = ( UCHAR * ) & ucRTUBuf[MB_SER_PDU_PDU_OFF];
            xFrameReceived = TRUE;
```

А также прописываются параметры режима передачи данных.

В файлах mbcrc.c и mbcrc.h прописывается алгоритм вычисления контрольной суммы, который применяется для определения подлинности отправленного и доставленного фрейма сообщения. Другими словами, контрольная сумма – алгоритм проверки на ошибки [48].

Файл mbfuncoils.c описывает тип данных Coils. Здесь прописываются основные функции чтения и записи этого типа данных.

Файл mbfuncdisc.c описывает тип данных Discrete Inputs. Здесь прописываются основные функции чтения для этого типа данных.

Файл mdfuncholding.c описывает тип данных Holding Registers. Здесь прописываются основные функции чтения и записи для этого типа данных.

Файл mbfuncinput.c описывает тип данных Input Registers. Здесь прописываются основные функции чтения для этого типа данных.

Файл mbfuncother.c описывает задание SlaveID. Здесь прописываются параметры ввода и считывания поля SlaveID.

Файл mbutils.c описывает порядок установки и считывания бит данных при приеме и передаче.

Приведем иерархию каталогов конфигурации порта:

1) файл port.h;
2) файл portevent.c;
3) файл portserial.c;
4) файл porttimer.c.

Приведем иерархию каталогов заголовочных файлов Modbus:

1) файл mb.h;
2) файл mbconfig.h;
3) файл mbframe.h;
4) файл mbfunc.h;
5) файл mbport.h;
6) файл mbproto.h;
7) файл mbutils.h.

4.3 Создание проекта

Проектирование кода будем выполнять в программе Programmer’s Notepad, которая является свободным текстовым редактором для Windows. Выбранная среда очень удобна в использовании со своим интеллектуально понятным интерфейсом.

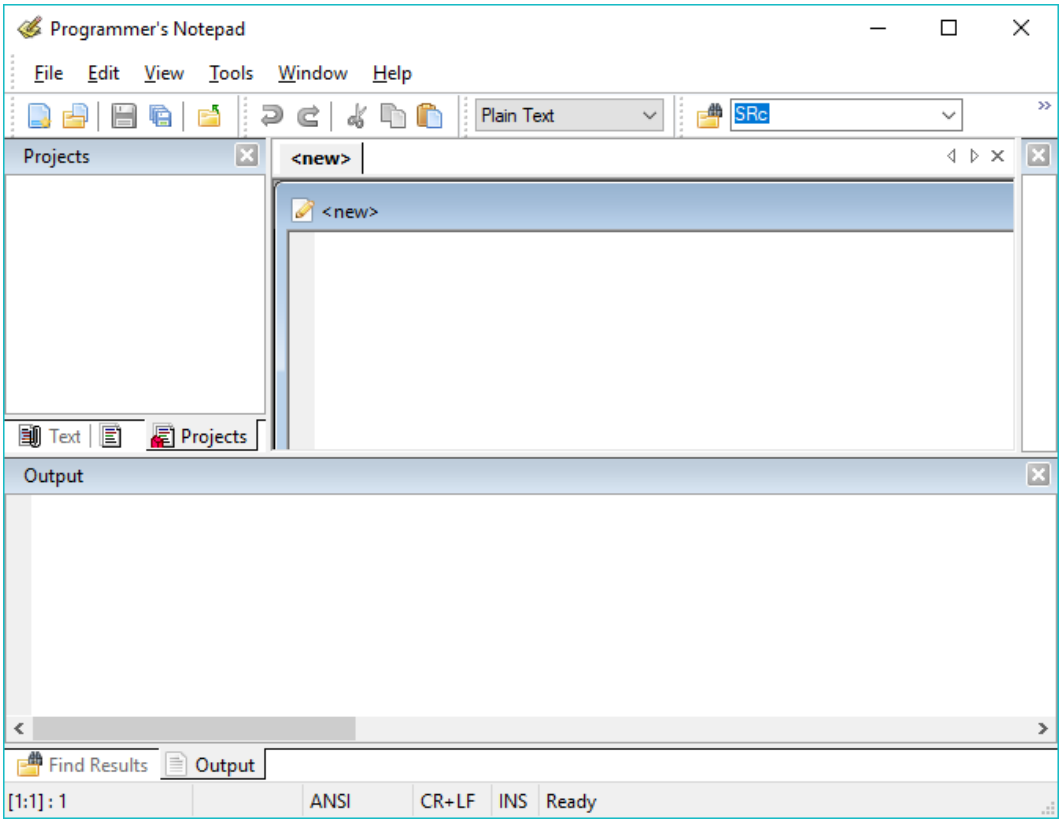

Рисунок 18 – Стартовое окно выбранного редактора кода

Создадим новый проект в выбранной среде разработки кода.

Далее прогрузим туда уже известные файлы и вложения из библиотеки FreeModbus.

Теперь всё готово для начала проектирования кода.

4.4 Описание работы программного обеспечения разрабатываемого стенда

Начнем с реализации кода главного загрузочного файла. По ходу распишем необходимые действия и произведем вычисление понадобившихся величин.

Рабочая программа микроконтроллера должна содержать следующие части: объявление регистров, описание таймера, конфигурация работы светодиодов, основное тело программы (функция main()), функции, отвечающие за

конфигурацию порта передачи данных, а также самого протокола передачи данных.

Объявление регистров происходит в самом начале программы. В нашем случае необходимо задать регистры, несущий следующий смысл:

1) регистр, содержащий состояние датчика температуры;
2) регистр, содержащий код ошибок;
3) регистр, содержащий относительное значение температуры;
4) регистр, содержащий максимальное значение температуры;
5) регистр, содержащий относительное значение времени.

Здесь были описаны так называемые Input Registers, которые используются только для чтения данных.

Необходимо задать так же Holding Registers, которые будут позволять считывать и записывать данные в память микроконтроллера. Такие регистры необходимы со следующим смыслом:

1) регистр, содержащий номер устройства;
2) регистр, содержащий параметры сетевого обмена;
3) регистр, содержащий слово управления;
4) регистр, содержащий задание температуры.

Таким образом, необходимо задать пять Input Registers, четыре Holding Registers, а также добавить регистр, который будет содержать значение старта.

Приведем листинг программы, где описаны необходимые регистры.

Листинг 3

```
#define REG_INPUT_START 1
#define REG_INPUT_NREGS 5
#define REG_HOLDING_START 1
#define REG_HOLDING_NREGS 4

enum {TMP_STA = 0, TMP_ERR, TMP_tmp, TMP_tmp_m, TMP_time};
enum {TMP_ID = 0, TMP_NW, TMP_CTRL, TMP_SP};
```

Далее необходимо описать моделирование переходного процесса нагрева, то есть считывание имитируемым датчиком температуры. Опишем этот процесс передаточной функцией, содержащей устойчивое апериодическое звено. Такая передаточная функция в полной мере отражает все особенности

переходного процесса, а также самой модели датчика. Приведем математическую модель:

$$W(s) = \frac{1}{0{,}01s+1}. \quad (2)$$

Для использования этой передаточной функции в программе необходимо преобразовать из пространства состояний в дискретную модель.

Запишем общую формулу z-преобразования.

$$G(z) = \sum_{n=0}^{\infty} x(n)z^{-n}. \quad (3)$$

Сейчас существует множество программных обеспечений, позволяющих без каких-либо затруднений произвести z-преобразования любой, даже самой сложной функции. Воспользуемся ПО MathCad для проведения данного преобразования. В итоге получим:

$$G(z) = W(s) \rightarrow ztrans; \quad (4)$$

$$G(z) = \frac{0{,}0009995}{z-0{,}999}. \quad (5)$$

При значении постоянной времени T=0,01 с.

Для записи закона изменения температуры запишем в виде конечно-разностного уравнения преобразованную функцию. В результате получаем следующее:

$$y(k+1) = 0{,}999 \cdot y(k) + 0{,}0009995 \cdot u(k). \quad (6)$$

Теперь в виде конечно-разностного уравнения можно включать закон изменения температуры в код программы.

Полученное уравнение реализуем в таймере-счетчике, причем каждый отсчет этого таймера будет происходить через каждые 0,01 с, то есть временной интервал между отсчетами будет равным величине постоянной времени Т.

Приведем листинг моделирования передаточной функции.

Листинг 4

```
SIGNAL (TIMER0_OVF_vect)
{
  TCNT0 = 0xB8;
  sei();
  if (usRegInputBuf[TMP_STA] != 0 || (usRegInputBuf[TMP_STA] == 0
&& usRegHoldingBuf[TMP_CTRL] >=4) )

{
      if (usRegHoldingBuf[TMP_CTRL] > 0 &&
usRegHoldingBuf[TMP_CTRL] < 5)
      {
          freq = 0.999 * freq + 0.0009995 *
(((float)usRegHoldingBuf[TMP_SP] / 32767.0 ) *
usRegInputBuf[TMP_tmp_m]);
      }
      else if (usRegHoldingBuf[TMP_CTRL] == 5)
      {
          freq = 0.998 * freq;
      }
  }
}
```

Далее необходимо задать режим работы светодиодов. Они работают следующим образом: при получении команды старт, то есть при начале нагрева, должны загораться светодиоды, причем в процентном соотношении от заданного значения уставки. В панели имеется 8 светодиодов, то есть рассчитаем разницу между загоранием светодиодов относительно друг друга в процентах от уставки. Обозначим искомое значение за x:

$$x = \frac{100}{8} = 12{,}5\ (\%). \qquad (5)$$

В итоге получилось следующее: через каждые 12,5 % от заданного значения уставки будет загораться один светодиод и так до того момента, пока не достигнем заданного значения температуры.

Также имеется возможность в зависимости от того, происходит ли нагрев или охлаждение, изменять цвет светодиодов.

В программе эти функции отразим следующим образом.

Листинг 5

```
void SetLedColor(BOOL red, BOOL green, BOOL blue)
{
    if ( red == 0)
        PORTC |= (1<<PORTC0);
    else
        PORTC &= ~(1<<PORTC0);

    if ( green == 0)
        PORTC |= (1<<PORTC1);
    else
        PORTC &= ~(1<<PORTC1);

    if ( blue == 0 )

PORTC |= (1<<PORTC3);
    else
        PORTC &= ~(1<<PORTC3);
}

inline
unsigned char GetDI()
{
    unsigned char DI = 0;
    DI |= ((PINC >> 7) & 0x01) << 0;
    DI |= ((PINC >> 6) & 0x01) << 1;
    DI |= ((PINC >> 5) & 0x01) << 2;

    DI |= ((PINC >> 4) & 0x01) << 3;
    DI = ~DI;
    return (DI & 0x0F);
}
```

В основной функции main() пропишем инициализацию используемых таймеров, произведем инициализацию портов, задействованных в работе, зададим начальные значения регистрам ,отвечающим за вывод температуры. Также произведем инициализацию светодиодной панели индикации.

Листинг окончательной версии разработанного программного обеспечения приведен в ПРИЛОЖЕНИИ А.

Программно необходимо реализовать переключение между следующим режимами протокола передачи данных: Modbus ASCII и Modbus RTU. В разрабатываемом ПО реализуем это следующими образом. Приведем листинг части программы.

Листинг 6

```
//Выбираем режим работы. RTU режим по умолчанию.
   unsigned short mode;
   mode = eeprom_read_word((uint16_t*)MODE);

   switch ( mode )
   {
      case 1:
        mode = MB_RTU;
        break;

      case 2:
        mode = MB_ASCII;
        break;

      default:
        mode = MB_RTU;
        eeprom_write_word( (uint16_t*)MODE, 1 );

 }

  //Записываем значение в регистр 1003.
   usRegHoldingBuf[3] = mode;
```

Также, исходя из технического задания, необходимо реализовать выбор бита паритета в передаваемом сообщении. Существует несколько вариантов бита паритета:

1) Odd parity;
2) Even parity;
3) No parity.

Значение бита паритета будем записывать в регистр 1002. По умолчанию зададим его значение равным No parity. Если по каким-либо причинам появится необходимость изменить бит паритета, то содержимое регистра 1002 следует изменить. Такие изменения выполним программно.

То есть, с помощью оператора switch() реализуем необходимый выбор. Приведем листинг программы.

Листинг 7

```
//Выбираем паритет. Even Parity по умолчанию.
  unsigned short parit;

parit = eeprom_read_word((uint16_t*)PARITY);
  switch ( parit )
  {
     case 1:
       parit = MB_PAR_NONE;
       break;

     case 2:
       parit = MB_PAR_ODD;
       break;

     case 3:
       parit = MB_PAR_EVEN;
       break;

     default:
       parit = MB_PAR_EVEN;
       eeprom_write_word((uint16_t*)PARITY, 3 );
   }

  //Записываем значение в регистр 1002.
    usRegHoldingBuf[2]=parit;
```

Исходя из технического задания, реализуем переключение скорости передачи данных. Скорость передачи данных при передаче через интерфейс RS-485 по витой паре часто зависит от длинны линии, то есть чем больше дистанция, а соответственно длиннее провод, тем меньше скорость передачи данных. В нашем случае техническим заданием ограничен разброс скоростей передачи данных, а именно:

1) 9600 бит/с;
2) 19200 бит/с;
3) 38400 бит/с.

Значение скорости передачи данных будем записывать в регистр 1001. По умолчание его значение равно 9600 бит/с. Если по каким-либо причинам появится

необходимость изменить бит паритета, то содержимое регистра 1001 следует изменить. Такие изменения выполним программно.

Произведем программную реализацию необходимого выбора скорости.

Листинг 8

```
//Выбираем скорость. 9600 бод по умолчанию.
  unsigned short boudRateInit;
  boudRateInit = eeprom_read_word( BOUD_SETTING_ADRESS );
  switch ( boudRateInit )
  {
     case 1:
       boudRateInit = 9600;
       break;

     case 2:

 boudRateInit = 19200;
       break;

     case 3:
       boudRateInit = 38400;

break;

     default:
       boudRateInit = 9600;
       eeprom_write_word(BOUD_SETTING_ADRESS, 1 );
  }

 //Записываем значение в регистр 1001.
 usRegHoldingBuf[1]=boudRateInit;
```

Для разрабатываемого ПО составим таблицы для более понятного пояснения содержания регистров, а также их адресов.

Таблица 16 – Digital Inputs

Номер	Описание	Обозначение
0x0000	регистр состояния дискретного входа	DI0
0x0001	регистр состояния дискретного входа	DI1
0x0002	регистр состояния дискретного входа	DI3
0x0003	регистр состояния дискретного входа	DI4

Таблица 17 – Coils

Номер	Описание	Обозначени
0x0000	Управление дискретным выходом	DO0
0x0001	Управление дискретным выходом	DO1
0x0002	Управление дискретным выходом	DO2
0x0003	Управление дискретным выходом	DO3
0x0004	Управление дискретным выходом	DO4
0x0005	Управление дискретным выходом	DO5
0x00060	Управление дискретным выходом	DO6
0x0007	Управление дискретным выходом	DO7
0x1000	Управление дискретным выходом	Red
0x1001	Управление дискретным выходом	Green
0x1002	Управление дискретным выходом	Blue

Таблица 18 – Input Registers

Номер	Описание	Обозначение
0x0000	Состояние дискретных входов	DI as WORD
0x0020	Tcpu	-
0x0021	Tps	-

Таблица 19 – Holding Registers

Номер	Описание	Обозначение
0x0000	Управление дискретным выходами DO0...DO7	(DI as WORD)
0x0001	цвет светодиодов	-
0x0002	цвет светодиодов (резерв)	-
0x0003	цвет светодиодов (резерв)	-
0x0004	режим работы (DO, бегущий огонь, ...)	-
0x0005	скорость "бега"	
0x42DB	Control register	(if 0xA5E0 - restart)

4.5 Алгоритм работы лабораторной установки

Составим алгоритм работы лабораторной установки для изучения семейства протоколов передачи данных Modbus.

В самом начале работы на установку необходимо подать питание, которое реализуется с помощью специального блока от розетки прямо на саму установку. Далее произойдет инициализация микроконтроллера, находящегося на плате самого устройства. Подключив интерфейс RS-485 к ПК, установка готова к использованию.

Теперь, когда программное обеспечение загружено в контроллер, то установка может свободно выполнять функции, предъявляемые к ней в техническом задании.

Для обмена данными с другими электронными устройствами используется преобразователь интерфейса RS-485, который в свою очередь подключается через разъем USB к любому желаемому устройству.

Рассмотрим алгоритм работы имитатора термоконтроллера, реализованного в разработанном стенде.

В режиме термоконтроллера данная установка работает следующим образом. Изначально задается уставка температуры, то есть то значение, до которого должен нагреться объект исследования. Эта температура записывается в регистр TMP_SP. При подаче сигнала «СТАРТ» происходит начало имитации считывания нагрева исследуемого предмета. При этом с течением времени температура увеличивается, причем закон изменения записан в виде передаточной функции, рассматриваемой выше. Через определенные промежутки времени значение температуры считывается и записывается в регистр TMP_tmp. Для индикации температуры нагрева используется светодиодная панель. Через каждые 12,5 % загорается ещё один светодиод и так до того момента, пока температура не достигнет значения уставки, то есть пока не загорятся все светодиоды. Все это время к значению TMP_tmp добавляются 12,5 % от значения, записанного в регистр TMP_SP.

4.6 Тестирование программного обеспечения для лабораторной установки для изучения семейства протоколов передачи данных Modbus

Для реализации пунктов технического задания произведем тестирование разработанного программного обеспечения на установке, которая была спроектирована в ходе выполнения выпускной квалификационной работы.

С помощью командной строки произведем сборку и отладку проекта. Загрузку ПО осуществим с помощью программатора USBAsp, а также среды AVRDUDE, которая является свободно распространяемой оболочкой для программирования.

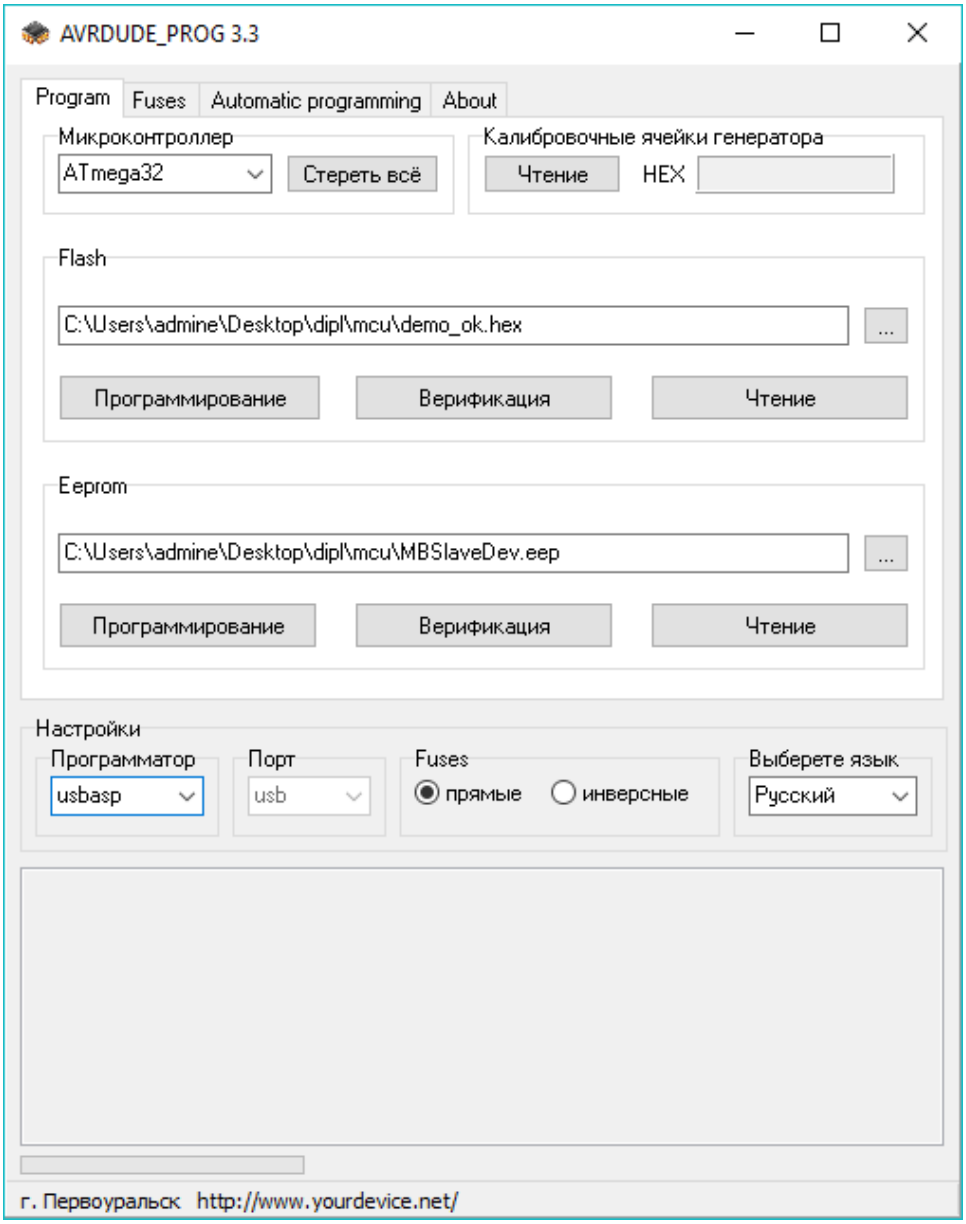

Рисунок 19 – Загрузочное окно программы AVRDUDE

После загрузки программы в контроллер на практике произведем реализацию пункта 3.2 настоящего технического задания, а именно имитацию термоконтроллера. На рисунках 20 и 21 представлена визуализация нагрева на светодиодах.

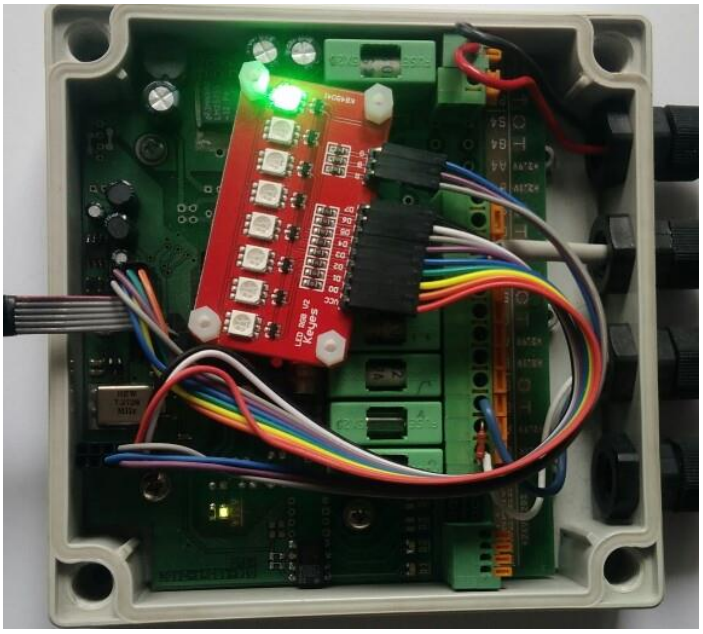

Рисунок 20 – Визуализация нагрева на 12,5 % от заданной уставки

Рисунок 21 – Визуализация нагрева на 100 % от заданной уставки

При имитации нагрева светодиодная панель выводит индикацию температуры, что соответствует требованиям технического задания.

С помощью программного обеспечения Serial Port Monitor произведем тестирование пунктов 3.4 и 3.5 настоящего технического задания.

Изменив бит паритета, произведем отправку запроса на чтение Holding Register (команда 0x03).

Изначально бит паритета установлен в значение No parity. Зададим это значение в программе Serial Port Monitor и отправим запрос. Результат запроса представлен на рисунке 22.

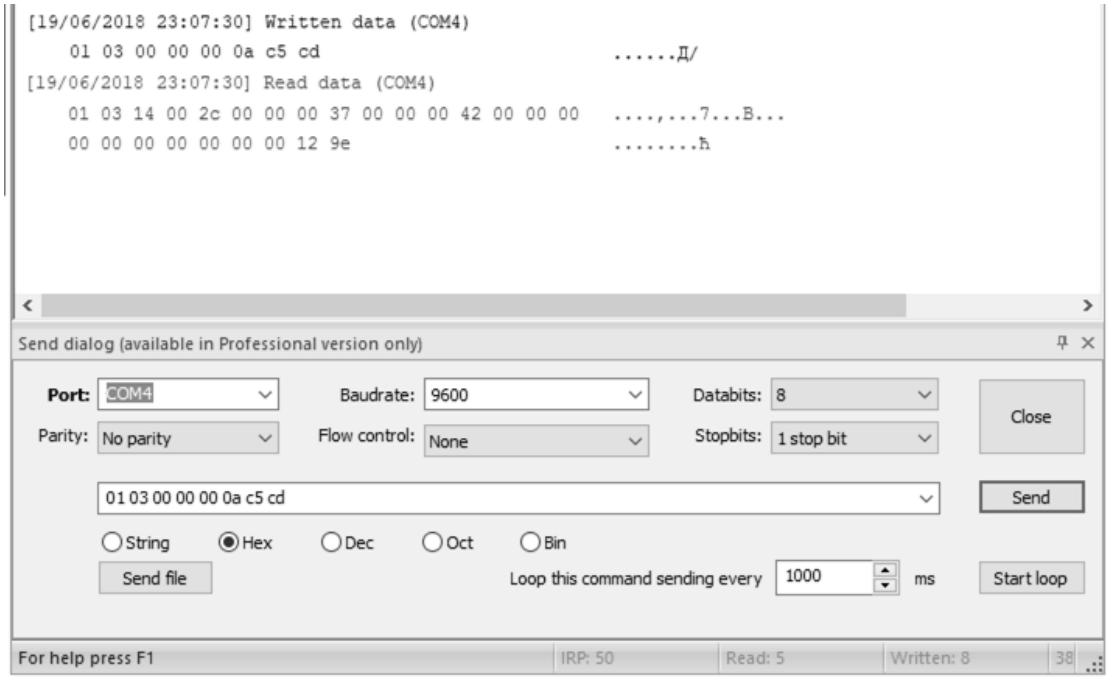

Рисунок 22 – Передача данных при бите паритета установленном в No parity

Изменим содержимое регистра 1002, отвечающего за значение паритета, на Even parity. Произведем отправку запроса с новым значением бита паритета. Результат запроса представлен на рисунке 23.

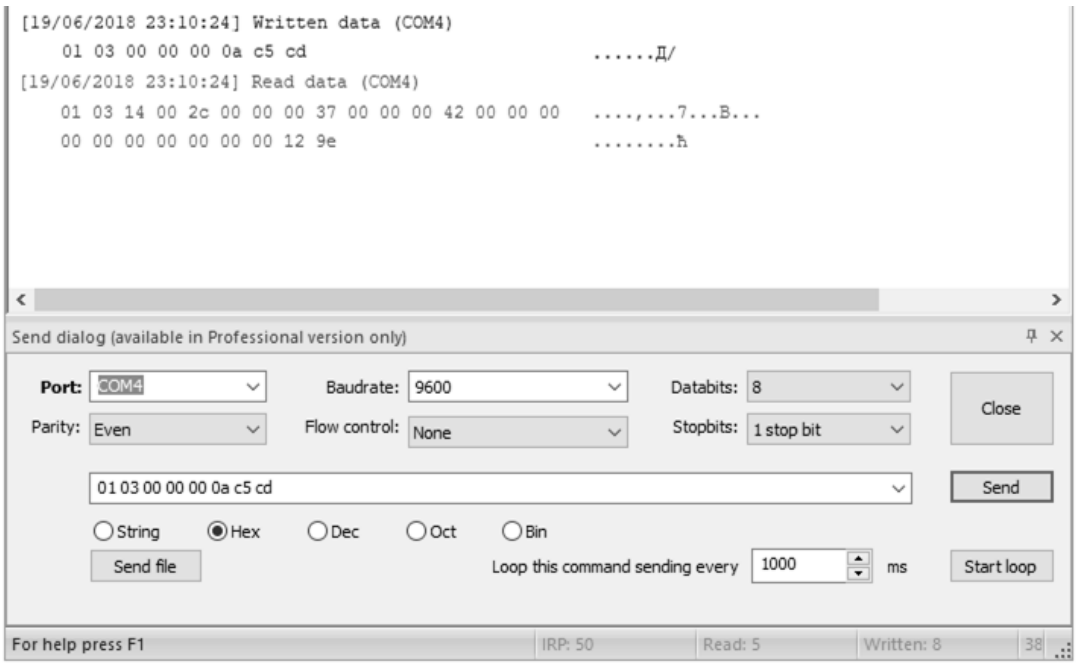

Рисунок 23 – Передача данных при бите паритета установленном в Even parity

Произведем ещё одно изменение бита паритета, а именно изменим содержание регистра 1002 на Odd parity. Произведем ещё одну отправку запроса с новым значением бита паритета. Результат запроса представлен на рисунке 24.

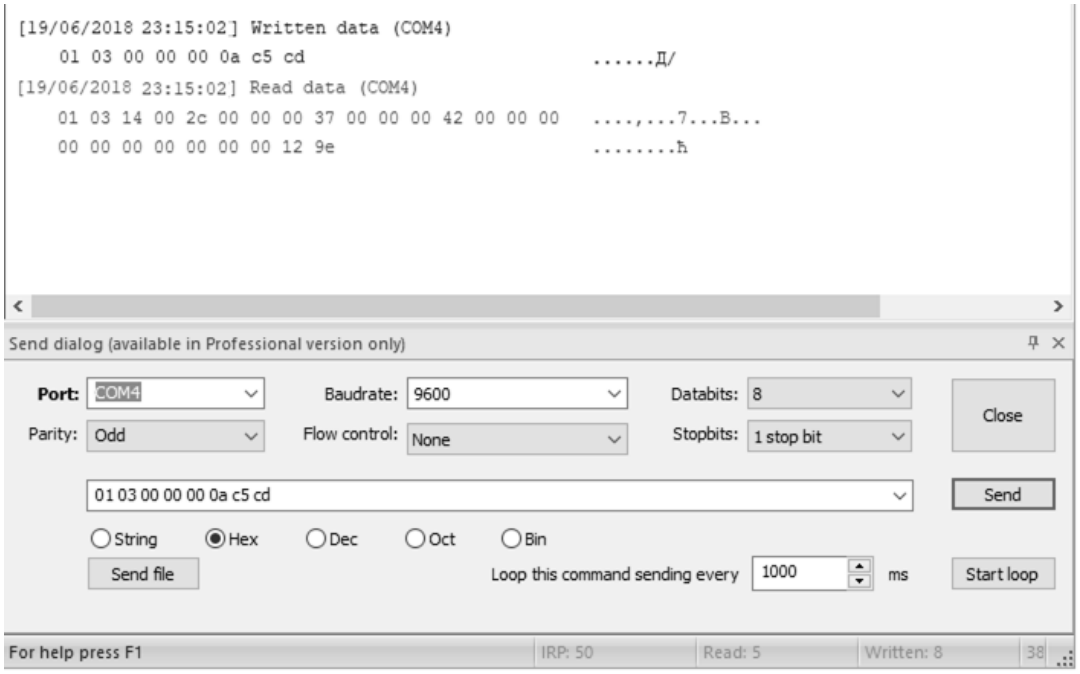

Рисунок 24 – Передача данных при бите паритета установленном в Odd parity

Таким образом, из рисунков 22-24 видно, что при изменении значения регистра, отвечающего за бит паритета, на запрос был получен корректный ответ, что подтверждает работоспособность разработанного ПО.

Изменив значение скорости передачи, произведем отправку запроса на чтение Holding Register (команда 0x03).

По умолчанию скорость передачи данных равна 9600 бит/с. Зададим это значение в программе Serial Port Monitor и отправим запрос. Результат запроса представлен на рисунке 25.

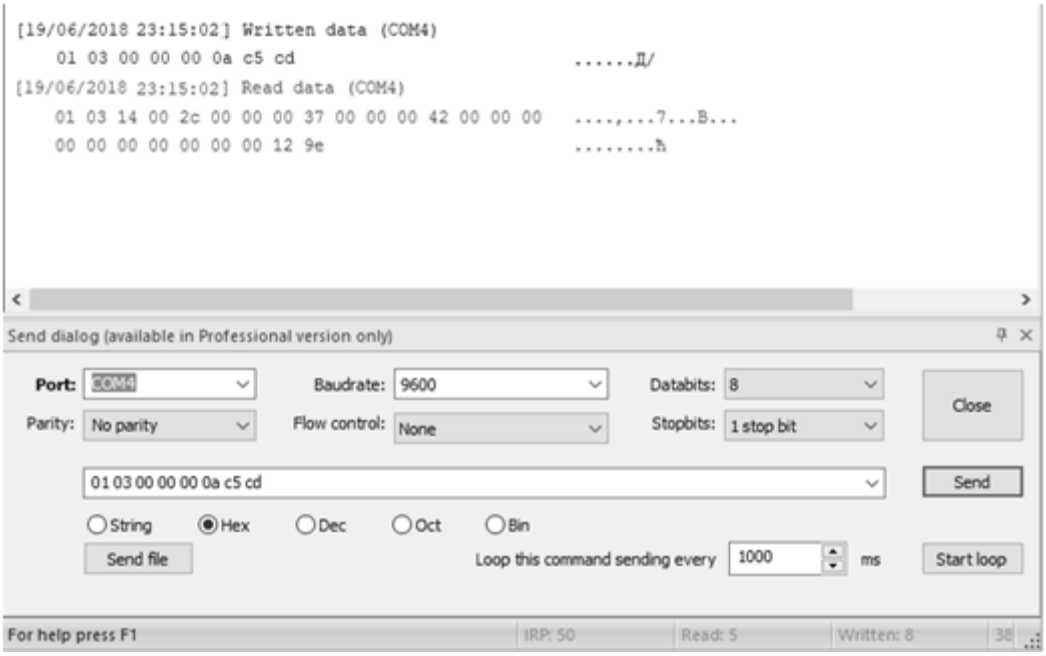

Рисунок 25 – Передача данных при значении скорости 9600 бит/с

Изменим содержимое регистра 1001, где поставим значение скорости передачи данных 19200 бит/с. Зададим это значение в программе Serial Port Monitor и отправим запрос. Результат запроса представлен на рисунке 26.

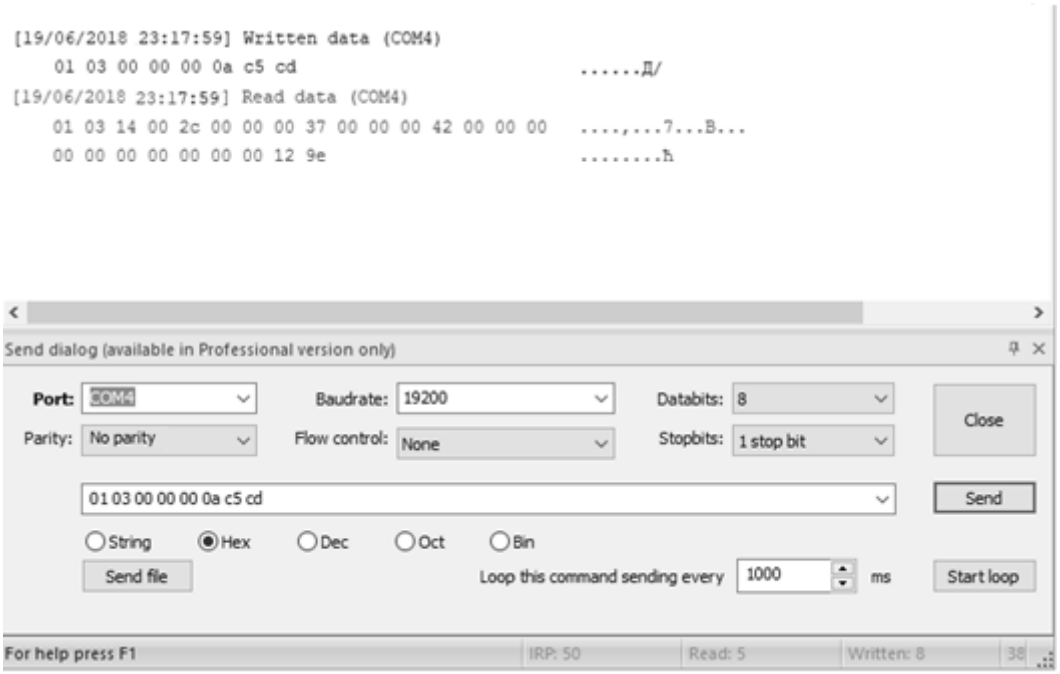

Рисунок 26 – Передача данных при значении скорости 19200 бит/с

Изменим содержимое регистра 1001, где поставим значение скорости передачи данных 38400 бит/с. Зададим это значение в программе Serial Port Monitor и отправим запрос. Результат запроса представлен на рисунке 27.

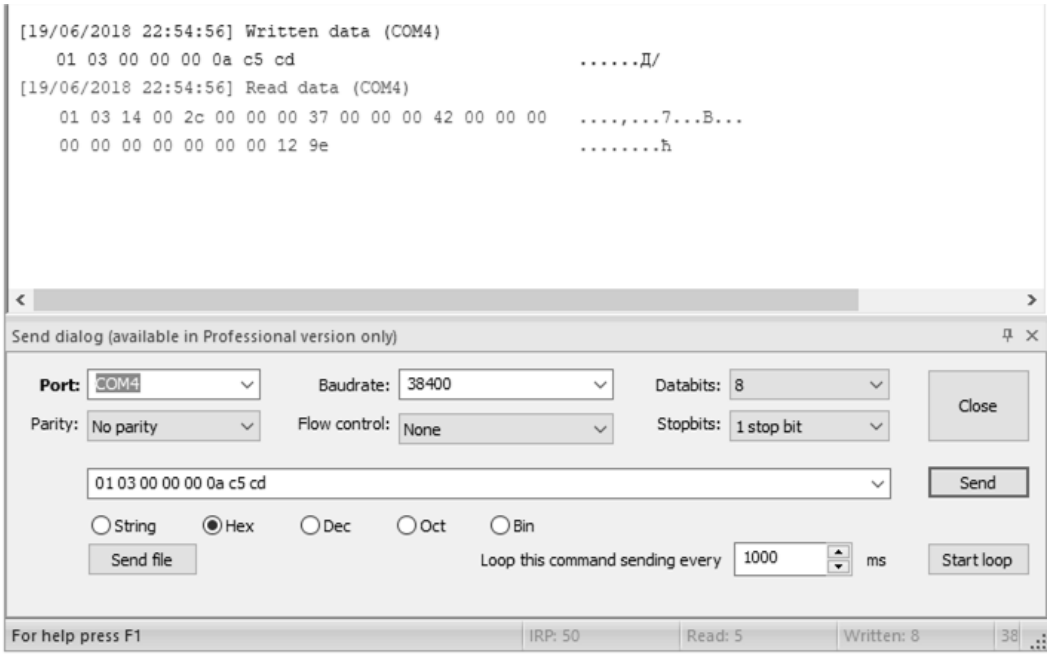

Рисунок 27 – Передача данных при значении скорости 38400 бит/с

Таким образом, из рисунков 25-27 видно, что при изменении скорости передачи на запрос был получен корректный ответ, что подтверждает работоспособность разработанного ПО.

5 МЕТРИКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ЛАБОРАТОРНОЙ УСТАНОВКИ

Программная метрика – мера, позволяющая получить численное значение некоторого свойства программного обеспечения [21].

Измерение ПО необходимо для определения, прогнозирования и улучшения качества существующего продукта или процесса.

Метрики программ принято разделять на три основные группы:

1) размерно-ориентированные метрики;

2) метрики сложности потока управления программ;

3) метрики сложности потока данных программ.

5.1 Метрики сложности потока управления программ

Метрики сложности потока управления программ:

1) метрики Мак-Кейба или цикломатическая сложность;

2) метрики Хансена;

3) метрики Вудворда;

4) метрик Харрисона и Мейджела;

5) метрики Пивоварского;

6) метрики граничных значений;

7) метрики Шнейдевинда.

С помощью вышеперечисленных метрик можно оперировать либо плотностью управляющих переходов внутри программ, либо взаимосвязями этих переходов.

Из перечисленных метрик рассмотрим наиболее популярные метрики: Мак-Кейба, Хансена, Харрисона и Мейджела.

5.1.1 Цикломатическая сложность, или метрика Мак-Кейба

Цикломатическая сложность части программного кода – количество линейно независимых маршрутов через программный код [21]. Математически

цикломатическая сложность структурированной программы определяется с помощью ориентированного графа, в узлах которого находятся неделимые группы команд, а ребрами соединяются такие блоки, которые могут быть выполнены сразу друг за другом. Таким образом, показатель цикломатической сложности определяется по формуле:

$$C = E - N + P = 22 - 17 + 4 = 9, \quad (7)$$

где: C – цикломатическая сложность;

E – количество рёбер в графе;

N – количество узлов в графе;

P – количество компонент связности.

Для вычисления формулы (7), необходимо построить ориентированный граф. Для этого воспользуемся некоторыми правилами для построения:

1) если исходный код не содержит никаких указаний if или циклы for, то цикломатическая сложность равна единице;

2) если код имеет единственный оператор if, содержащий простое условие, то существует два пути через код: один если условие оператора if имеет значение true и один – если false.

3) если код содержит в себе подпрограммы, то каждая из низ может быть представлены как независимая часть графа, в таком случае, P будет равно числу подпрограмм.

Основными достоинствами данной метрики являются: простота вычисления, наглядность и содержательность [26]. Основным недостатком является нечувствительность к размеру и изменению структуры ПО.

5.1.2 Метрика Хансена

Сущность данной метрики состоит в том, что сложность программы оценивается парой: цикломатическая сложность и число операторов.

Значение метрики Хансена вычисляется по формуле:

$$\{n, N\} = \{9, 155\}, \quad (8)$$

где n – цикломатическая сложность;

N – число операторов в программном коде.

Программное обеспечение для стенда контроля еще находится на стадии разработки, то предварительное число операторов равно 100.

Использование в качестве оценки сложности программы пары, представленной в формуле (8), повышается чувствительность метрики к структурированности программы.

Метрика Хансена может быть применена для оценки сложности анализа бинарного кода статическими методами, когда аналитику известен размер всего приложения и число инструкций в составляющих его функциях.

5.1.3 Метрики Харрисона и Мейджела

Метрики Харрисона и Мейджела учитывают уровень вложенности и размер программы. Данная метрика включает в себя следующие компоненты:

- функциональное число;
- функциональное отношение.

Функциональное число рассчитывается по формуле:

$$F_1 = \sum \chi_l = 1, \quad (9)$$

где χ_l – приведенные сложности всех вершин ориентированного графа.

Функциональное отношение рассчитывается по формуле:

$$F^* = \frac{N \cdot \chi_l}{F_1} = 1, \quad (10)$$

где $N \cdot \chi_l$ – число вершин графа;

F_1 – функциональное число.

5.2 Метрики сложности потока данных программ

В отличие от метрик сложности управления, метрики сложности потока данных в большей степени ориентированы на оценку исходного текста программ на языках высокого уровня, соответственно, с высокоуровневыми типами данных.

Метрики сложности потока данных программ:

1) метрика Чепина;
2) метрика спена;
3) метрика обращений к глобальным переменным;
4) метрика Кафура.

Рассмотрим наиболее популярную метрику – метрику Чепина.

5.2.1 Метрика Чепена

Метрика Чепина состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных разбивается на 4 группы:

1) множество «P» – вводимые переменные для расчетов и для обеспечения вывода;
2) множество «M» – модифицируемые или создаваемые внутри программы переменные;
3) множество «C» – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
4) множество «T» – не используемые в программе (“паразитные”) переменные.

Тогда значение метрики Чепина вычисляется по формуле:

$$Q = a1P + a2M + a3C + a4T = 30+10+45+2=87, \quad (11)$$

где a1, a2, a3, a4 – весовые коэффициенты.

Автор метрики предлагает следующие значения весовых коэффициентов: a1=1, a2=2, a3=3, a4=0.5.

Таким образом, любые рассмотренные метрические характеристики анализируемых программ должны либо использоваться совместно и в зависимости друг от друга, либо применяться в зависимости от конкретной задачи.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы разработан комплекс программно-технических средств для изучения семейства протоколов передачи данных Modbus.

Произведен синтез структурных схем лабораторного стенда и его основных конструктивных частей. Осуществлен выбор оборудования: микроконтроллер, интерфейс для передачи данных, индикационная панель и другие элементы, необходимые для работоспособности установки.

Разработан алгоритм работы лабораторной установки исходя из требований технического задания. По алгоритму работы стенда реализовано программное обеспечение, которое протестировано на разработанной установке. В ПО реализовано следующее: передача данных по UART со скоростями 9600 бит/с, 19200 бит/с, 34800 бит/с; изменение бита паритета; переключение между Modbus RTU Modbus и ASCII; имитация режима работы термоконтроллера.

Произведен анализ качества разработанного программного обеспечения с помощью различных видов метрик. Такая процедура позволила дать количественную и качественную оценку разработанного программного обеспечения.

Разработанное устройство подлежит дальнейшему анализу и разработке. К числу рассматриваемых вопросов можно отнести разработку дополнительных имитационных режимов, добавление других протоколов передачи данных, усовершенствование уже имеющихся элементов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Баранов В.Н. Применение микроконтроллеров AVR: Схемы, алгоритмы, программы. – М.: Издательский дом «ДОДЭКА-XXI». – 2004, 287 с.

2 Белов, А.В. Программирование микроконтроллер для начинающих и не только: учебник / А.В. Белов. – Москва: Изд-во: НиТ, 2016. – 352 с.

3 Борисов А.М. Основы построения промышленных сетей автоматики: учебное пособие. – Челябинск: Изд. Центр ЮУрГУ. – 2012.

4 Вставская Е.В. Микропроцессорные устройства систем управления. Конспект лекций. – Челябинск: Издадельство ЮУрГУ. – 2009, 92 с.

5 ГОСТ 2.701-2008. ЕСКД. Схемы. Виды и типы. Общие требования к выполнению. – М.: Изд-во стандартов, 2009. – 16 с.

6 ГОСТ 19.002-80. ЕСПД. Схемы алгоритмов и программ. Правила выполнения. – М.: Изд-во стандартов, 1981. – 9 с.

7 ГОСТ 19.003-80. ЕСПД. Схемы алгоритмов и программ. Обозначение условные графические. – М.: Изд-во стандартов, 1981. – 9 с.

8 ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. – М.: Изд-во стандартов, 1992. – 22 с.

9 ГОСТ 2.743–1991. ЕСКД. Элементы цифровой техники. – М. Изд-во стандартов, 1991. – 22 с.

10 ГОСТ 2.759–1982. ЕСКД. Элементы аналоговой техники. – М. Изд-во стандартов, 1982. – 23 с

11 ГОСТ 2.764–1986. ЕСКД. Интегральные оптоэлектронные элементы индикации. – М. Изд-во стандартов, 1986. – 10 с.

12 Денисенко, В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. – М.: Горячая линия-Телеком, 2009. – 608 с.

13 Дорф Р., Бишоп.Р Современные системы управления. – М.: Лаборатория базовых знаний. – 2002. – 832 с.

14 Евстифеев, А.В. Микроконтроллеры AVR семейства Mega: руководство пользователя/ А.В. Евстифеев. – М.: издательский дом «ДОДЭКА-XXI», 2007.

15 Зимин В.В. Промышленные сети. – Н. Новгород: НГТУ. – 2006, 252 с.

16 Кангин В.В, Аппаратные и программные средства управления. Промышленные сети и контроллеры: учебное пособие. – М.: Бином. Лаборатория знаний. – 2010.

17 Классификация и выбор микроконтроллеров [Электронный ресурс]. – Режим доступа: https://prog-cpp.ru/select-micro/ (дата обращения 18.05.18).

18 Кругляк К. Промышленные сети: Цели и средства Современные технологии автоматизации. – 2002. №4. с. 6-17.

19 Лабораторный стенд: «Интерфейсы RS-485/422 в микроконтроллерных и промышленных сетях» [Электронный ресурс]. – Режим доступа: http://labstand.ru/ – Лабораторный стенд. (дата обращения 07.06.2018)

20 Локотков А., Интерфейсы последовательной передачи данных, https://www.cta.ru/cms/f/326702.pdf (дата обращения 07.05.2018).

21 Метрики кода программного обеспечения. – https://www.viva64.com/ru/a/0045/#ID0E2C (дата обращения 17.04.2018).

22 Микушин, А.В. Цифровые устройства и микропроцессоры: учеб. пособие / А.В. Микушин, А.М. Сажнев, В.И. Сединин. – СПб.: БХВ-Петербург, 2010. – 832 с.

23 Мир микроконтроллеров –: http://microkontroller.ru/praktikum-mikrokontrollershhika – (дата обращения 08.02.2018).

24 Мортон, Д. Микроконтроллеры AVR: ввод. курс: пер. с англ. / Д. Мортон. – М.: ДОДЭКА-21, 2006.

25 Новиков, Ю.В. Основы микропроцессорной техники: учебное пособие/ Ю.В. Новиков, П.

26 К. Скоробогатов. – 4-е изд., испр. – М.: Интернет-Университет Информационных Технологий; Бином. Лаборатория знаний, 2009. – 357 с.

27 Организация памяти микроконтроллера [Электронный ресурс]. – Режим доступа: https://prog-cpp.ru/micro-memory/ (дата обращения 20.04.18).

28 Основные понятия о промышленных сетях и интерфейсах - http://www.bookasutp.ru/Chapter2_1.aspx (дата обращения 01.04.2018).

29 Парк Дж., Маккей С., Райт Э. Передача данных в системах контроля и управления: Практическое руководство. – М.: Группа ИДТ. – 2007.

30 Перрин Б., Интерфейс RS-485 - https://emag.ru/pdf/teldor.pdf (дата обращения 28.05.2018).

31 Предко М. Руководство по микроконтроллерам – М.: Постмаркет. – 2001, 415 с.

32 Протокол передачи данных Modbus - https://ru.wikipedia.org/wiki/Modbus.

33 Реализация протокола Modbus на электронных устройствах - https://hwcenter.ru (дата обращения 21.05.18).

34 Сетевая модель OSI – https://ru.wikipedia.org/wiki/setevaya _model_OSI .

35 Серф В., Дей Д., Протоколы и методы управления в сетях передачи данных. – М.: Радио и связь. – 1985, 480с.

36 СТО ЮУрГУ 04–2008 Стандарт организации. Курсовое и дипломное проекти-рование. Общие требования к содержанию и оформлению / составители: Т.И. Парубочая, Н.В. Сырейщикова, В.И. Гузеев, Л.В. Винокурова. – Челябинск: Изд-во ЮУрГУ, 2008. – 56 с.

37 Таймеры-счетчики [Электронный ресурс]. – Режим доступа: https://prog-cpp.ru/micro-timers/ (дата обращения 25.04.18).

38 Термоконтроллер - https://www.honeywellprocess.com/library/tech-specs .

39 Томас Дж., Введение в протокол Modbus – И.: СТА, 2009 г. 57 с.

40 Трамперт В. AVR микроконтроллеры – К.: МК-Пресс. – 2006, 458 с.

41 Хусаинов Р.З., Садов В.Б. Программирование микроконтроллеров семейства ATmega: методические указания. – Челябинск: Издательство ЮУрГУ. – 2007, 123 с.

42 Черников Б.В., Поклонов Б.Е. Оценка качества программного обеспечения, -М.: ИД «ФОРУМ»-ИНФА-М, 2012 – 22с.

43 Чип и Дип: электронные компоненты и приборы. – Режим доступа: https://www.chipdip.ru/product/ds-201 – Гнездо питания на плату (дата обращения 09.02.2018).

44Чип и Дип: электронные компоненты и приборы. – Режим доступа: https://www.chipdip.ru/product/idc-06ms-bh-06 – Вилка на плату прямая 6 конт. (дата обращения 09.02.2018).

45Чип и Дип: электронные компоненты и приборы. – Режим доступа: https://www.chipdip.ru/product/kls7-ts3601-4.3-180-tc-0120 – (дата обращения 09.02.2018).

46Энциклопедия АСУ ТП [Электронный ресурс]. – Режим доступа: http://www.bookasutp.ru/Chapter2_8.aspx . – Modbus. – (Дата обращения: 19.11.2016).

47 myROBOT: Подключение atmel avr, стабилизация работы микроконтроллера - http://myrobot.ru/articles/mc_stab.php – (дата обращения 08.02.2018).

48 EasyModbus - http://easymodbustcp.net/en/ .

49FreeModbus – A Modbus ASCII/RTU and TCP implementation [Электронный ресурс]. – Режим доступа - https://www.freemodbus.org/. – (дата обращения 20.05.18).

50 nModbus - http://openbooks.ifmo.ru/ru/file/980/980.pdf .

51 STM32F303VC - http://www.datasheetspdf.com/datasheet/stm32f303vc.html.

52 Pic16f684-i/sl - http://datasheet.su//microchip/pic16f684-i .

ПРИЛОЖЕНИЕ А

Листинг разработанного ПО

```
/* ----------------------- AVR INCLUDES ----------------------*/
#INCLUDE "AVR/IO.H"
#INCLUDE "AVR/INTERRUPT.H"
#INCLUDE <UTIL/DELAY.H>


/* ---------------------- MODBUS INCLUDES --------------------*/
#INCLUDE "MB.H"
#INCLUDE "MBPORT.H"

/* ---------------------- DEFINES ----------------------------*/
#DEFINE REG_INPUT_START 1
#DEFINE REG_INPUT_NREGS 5
#DEFINE REG_HOLDING_START 1
#DEFINE REG_HOLDING_NREGS 4

ENUM {TMP_STA = 0, TMP_ERR, TMP_TMP, TMP_TMP_M, TMP_TIME};
ENUM {TMP_ID = 0, TMP_NW, TMP_CTRL, TMP_SP};


/* ---------------------- STATIC VARIABLES -------------------*/
STATIC USHORT   USREGINPUTBUF[REG_INPUT_NREGS];
STATIC USHORT   USREGHOLDINGBUF[REG_HOLDING_NREGS];

FLOAT FREQ = 0;
UNSIGNED CHAR LEDS = 0;
UNSIGNED CHAR DI;

SIGNAL( TIMER0_OVF_VECT )
{
  // REINITIALIZE TIMER 0 VALUE
  TCNT0 = 0XB8;
  SEI();

  // МОДЕЛИРОВАНИЕ ПЕРЕДАТОЧНОЙ ФУНКЦИИ
  //
  //              1                        0.0009995
  //  W(S) = ----------    ==>   G(Z) = -----------, ДЛЯ TS =
0.01S
  //          10S + 1                   Z - 0.999
  //
  //
  //  Y(K+1) = 0.999 * Y(K) + 0.0009995 * U(K)

  IF (USREGINPUTBUF[TMP_STA] != 0 || (USREGINPUTBUF[TMP_STA] == 0
&& USREGHOLDINGBUF[TMP_CTRL] >=4) )
  {
      IF (USREGHOLDINGBUF[TMP_CTRL] > 0 &&
USREGHOLDINGBUF[TMP_CTRL] < 5)
```

```
      {
          FREQ = 0.999 * FREQ + 0.0009995 *
(((FLOAT)USREGHOLDINGBUF[TMP_SP] / 32767.0 ) *
USREGINPUTBUF[TMP_TMP_M]);
      }
      ELSE IF (USREGHOLDINGBUF[TMP_CTRL] == 5)
      {
          FREQ = 0.998 * FREQ;
      }
  }
}

INLINE
VOID OUTPUTLEDS(UNSIGNED CHAR LEDS)
{
    LEDS  = ~LEDS;
    PORTA = (PORTA & 0XC0) | (LEDS & 0X3F);
    PORTD = (PORTD & 0X3F) | (LEDS & 0XC0);
}

INLINE
VOID SETLEDCOLOR(BOOL RED, BOOL GREEN, BOOL BLUE)
{
    IF ( RED == 0)
        PORTC |= (1<<PORTC0);
    ELSE
        PORTC &= ~(1<<PORTC0);

    IF ( GREEN == 0)
        PORTC |= (1<<PORTC1);
    ELSE
        PORTC &= ~(1<<PORTC1);

    IF ( BLUE == 0 )
        PORTC |= (1<<PORTC3);
    ELSE
        PORTC &= ~(1<<PORTC3);
}

INLINE
UNSIGNED CHAR GETDI()
{
    UNSIGNED CHAR DI = 0;
    DI |= ((PINC >> 7) & 0X01) << 0;
    DI |= ((PINC >> 6) & 0X01) << 1;
    DI |= ((PINC >> 5) & 0X01) << 2;
    DI |= ((PINC >> 4) & 0X01) << 3;
    DI = ~DI;
    RETURN (DI & 0X0F);
}
```

```
/* ---------------------- START IMPLEMENTATION -----------------
*/
INT MAIN( VOID )
{
    DDRA = (0<<DDA7) | (0<<DDA6) | (1<<DDA5) | (1<<DDA4) |
(1<<DDA3) | (1<<DDA2) | (1<<DDA1) | (1<<DDA0);
    DDRD = (1<<DDD7) | (1<<DDD6) | (0<<DDD5) | (0<<DDD4) |
(0<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);
    // PORT C INITIALIZATION
    // FUNCTION: BIT7=IN BIT6=IN BIT5=IN BIT4=IN BIT3=OUT BIT2=IN
BIT1=OUT BIT0=OUT
    DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (1<<DDC3)
| (0<<DDC2) | (1<<DDC1) | (1<<DDC0);
    // STATE: BIT7=T BIT6=T BIT5=T BIT4=T BIT3=0 BIT2=T BIT1=0
BIT0=1
    PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) |
(1<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (1<<PORTC0);

    // TIMER/COUNTER 0 INITIALIZATION
    // CLOCK SOURCE: SYSTEM CLOCK
    // CLOCK VALUE: 7,200 KHZ
    // MODE: NORMAL TOP=0XFF
    // OC0 OUTPUT: DISCONNECTED
    // TIMER PERIOD: 10 MS
    TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) |
(1<<CS02) | (0<<CS01) | (1<<CS00);
    TCNT0=0XB8;
    OCR0=0X00;
    TIMSK |= (1<<TOIE0);
    CONST UCHAR     UCSLAVEID[] = { 0XAA, 0XBB, 0XCC };
    EMBERRORCODE    ESTATUS;
    /ЗДЕСЬ ПРОИСХОДИТ ИНИЦИАЛИЗАЦИИ ДАННЫХ ИЗ ЭНЕРГОНЕЗАВИСИМОЙ
ПАМЯТИ. ПРИ ОТСУТСТВИИ ДАННЫХ В НЕЙ, ЗАПИСЫВАЕМ ЗНАЧЕНИЯ ПО
УМОЛЧАНИЮ.
  //ВЫБИРАЕМ СКОРОСТЬ. 9600 БОД ПО УМОЛЧАНИЮ.
  UNSIGNED SHORT BOUDRATEINIT;
  BOUDRATEINIT = EEPROM_READ_WORD( BOUD_SETTING_ADRESS );
  SWITCH ( BOUDRATEINIT )
  {
     CASE 1:
       BOUDRATEINIT = 9600;
       BREAK;

     CASE 2:
       BOUDRATEINIT = 19200;
       BREAK;

     CASE 3:
       BOUDRATEINIT = 38400;
       BREAK;

     DEFAULT:
```

```
        BOUDRATEINIT = 9600;
        EEPROM_WRITE_WORD(BOUD_SETTING_ADRESS, 1 );
    }

    //ЗАПИСЫВАЕМ ЗНАЧЕНИЕ В РЕГИСТР 1001.
    USREGHOLDINGBUF[1]=BOUDRATEINIT;
    //ВЫБИРАЕМ ПАРИТЕТ. EVEN PARITY ПО УМОЛЧАНИЮ.
    UNSIGNED SHORT PARIT;
    PARIT = EEPROM_READ_WORD((UINT16_T*)PARITY);
    SWITCH ( PARIT )
    {
       CASE 1:
         PARIT = MB_PAR_NONE;
         BREAK;

       CASE 2:
         PARIT = MB_PAR_ODD;
         BREAK;

       CASE 3:
         PARIT = MB_PAR_EVEN;
         BREAK;

       DEFAULT:
         PARIT = MB_PAR_EVEN;
         EEPROM_WRITE_WORD((UINT16_T*)PARITY, 3 );
     }

     //ЗАПИСЫВАЕМ ЗНАЧЕНИЕ В РЕГИСТР 1002.
     USREGHOLDINGBUF[2]=PARIT;
     //ВЫБИРАЕМ РЕЖИМ РАБОТЫ. RTU РЕЖИМ ПО УМОЛЧАНИЮ.
     UNSIGNED SHORT MODE;
     MODE = EEPROM_READ_WORD((UINT16_T*)MODE);
     SWITCH ( MODE )
     {
        CASE 1:
          MODE = MB_RTU;
          BREAK;

        CASE 2:
          MODE = MB_ASCII;
          BREAK;

        DEFAULT:
          MODE = MB_RTU;
          EEPROM_WRITE_WORD( (UINT16_T*)MODE, 1 );
     }
     //ЗАПИСЫВАЕМ ЗНАЧЕНИЕ В РЕГИСТР 1003.
     USREGHOLDINGBUF[3] = MODE;
     //ВЫБИРАЕМ НОМЕР ПОДЧИНЕННОГО УСТРОЙСТВА. 1-ОЕ УСТРОЙСТВО ПО
УМОЛЧАНИЮ.
     UNSIGNED SHORT IDSLV = EEPROM_READ_WORD((UINT16_T*)IDSLAVE);
```

```
   IF ( IDSLV < 1 || IDSLV > 247 )
   {
      EEPROM_WRITE_WORD((UINT16_T*)IDSLAVE, 1 );
      IDSLV = 1;
   }
   //ЗАПИСЫВАЕМ ЗНАЧЕНИЕ В РЕГИСТР 1004.
    USREGHOLDINGBUF[4] = IDSLV;

    EMBERRORCODE    ESTATUS;
    //ЗДЕСЬ СООТВЕТСТВЕННО ПРОИСХОДИТ САМА ИНИЦИАЛИЗАЦИЯ.
    ESTATUS = EMBINIT( (EMBMODE)MODE, IDSLV, 0, BOUDRATEINIT,
(EMBPARITY)PARIT );
    SEI(  );

    /* ENABLE THE MODBUS PROTOCOL STACK. */
    ESTATUS = EMBENABLE(  );
    USREGINPUTBUF[ TMP_STA ] = 0;
    USREGINPUTBUF[ TMP_ERR ] = 0;
    USREGINPUTBUF[ TMP_TMP ] = 0;
    USREGINPUTBUF[ TMP_TIME ] = 0;
    USREGINPUTBUF[ TMP_TMP_M ] = 0X029A;
    SETLEDCOLOR(1, 0, 0);
    OUTPUTLEDS( 0XFF );
    FOR( ;; )
    {
        ( VOID )EMBPOLL(  );

        /* HERE WE SIMPLY COUNT THE NUMBER OF POLL CYCLES. */
        DI = GETDI();

        IF (DI != 0)
            USREGINPUTBUF[TMP_ERR] = DI;

        USREGINPUTBUF[DRV_TMP] = (SHORT) ((FREQ * 0X3FFF) /
USREGINPUTBUF[DRV_TMP_M]);
        USREGINPUTBUF[DRV_TIME] = USREGINPUTBUF[DRV_TMP] /4;
        SWITCH (USREGHOLDINGBUF[TMP_CTRL])
        {
        CASE 0:
            USREGINPUTBUF[ TMP_TMP ] = 0;
            USREGINPUTBUF[ TMP_TIME ] = 0;
            BREAK;
        CASE 1:
            USREGINPUTBUF[ TMP_ERR ] = 0;
            BREAK;
        CASE 2:
            USREGINPUTBUF[ TMP_STA ] = 1;
            BREAK;
        CASE 3:
            USREGINPUTBUF[ TMP_STA ] = 2;
            BREAK;
        CASE 4:
```

```
            USREGINPUTBUF[ TMP_STA ] = 0;
            BREAK;
        }

        IF (USREGINPUTBUF[TMP_ERR] == 0)
        {
            IF ((USREGINPUTBUF[TMP_STA] == 0) ||
(USREGINPUTBUF[TMP_STA] == 1 && USREGINPUTBUF[TMP_CTRL] == 5))
SETLEDCOLOR (0, 1, 1);
            ELSE IF (USREGINPUTBUF[TMP_STA] == 1 &&
USREGINPUTBUF[TMPDRV_CTRL] != 5) SETLEDCOLOR (0, 1, 0);
            ELSE IF (USREGINPUTBUF[TMP_STA] == 2) SETLEDCOLOR (0,
0, 1);
            ELSE
            {
                SETLEDCOLOR (1, 0, 0);
            }

            SWITCH ( USREGINPUTBUF[TMP_TMP] / 2047 )
            {
                CASE 0:
                    IF (USREGINPUTBUF[TMP_TMP] < 100 ) LEDS =
0X00;
                    ELSE LEDS = 0X01;
                    BREAK;
                CASE 1:
                    LEDS = 0X03;
                    BREAK;
                CASE 2:
                    LEDS = 0X07;
                    BREAK;
                CASE 3:
                    LEDS = 0X0F;
                    BREAK;
                CASE 4:
                    LEDS = 0X1F;
                    BREAK;
                CASE 5:
                    LEDS = 0X3F;
                    BREAK;
                CASE 6:
                    LEDS = 0X7F;
                    BREAK;
                DEFAULT:
                    LEDS = 0XFF;
            }
            OUTPUTLEDS( LEDS );
        }
        ELSE
        {
            SETLEDCOLOR (1, 0, 0);
            OUTPUTLEDS( 0XFF );
```

```
        }
    }
}

EMBERRORCODE
EMBREGINPUTCB( UCHAR * PUCREGBUFFER, USHORT USADDRESS, USHORT
USNREGS )
{
    EMBERRORCODE    ESTATUS = MB_ENOERR;
    INT             IREGINDEX;
    IF( ( USADDRESS >= REG_INPUT_START ) && ( USADDRESS + USNREGS
<= REG_INPUT_START + REG_INPUT_NREGS ) )
    {
        IREGINDEX = ( INT )( USADDRESS - REG_INPUT_START );
        WHILE( USNREGS > 0 )
        {
            *PUCREGBUFFER++ =
                ( UNSIGNED CHAR )( USREGINPUTBUF[IREGINDEX] >> 8
);
            *PUCREGBUFFER++ =
                ( UNSIGNED CHAR )( USREGINPUTBUF[IREGINDEX] & 0XFF
);
            IREGINDEX++;
            USNREGS--;
        }
    }
    ELSE
    {
        ESTATUS = MB_ENOREG;
    }

    RETURN ESTATUS;
}

EMBERRORCODE
EMBREGHOLDINGCB( UCHAR * PUCREGBUFFER, USHORT USADDRESS, USHORT
USNREGS, EMBREGISTERMODE EMODE )
{
    EMBERRORCODE    ESTATUS = MB_ENOERR;
    INT             IREGINDEX;
    IF( ( USADDRESS >= REG_HOLDING_START ) && ( USADDRESS +
USNREGS <= REG_HOLDING_START + REG_HOLDING_NREGS ) )
    {
        IREGINDEX = ( INT )( USADDRESS - REG_HOLDING_START );
   SWITCH ( EMODE )
   {
   /* PASS CURRENT REGISTER VALUES TO THE PROTOCOL STACK. */
   CASE MB_REG_READ:
       WHILE( USNREGS > 0 )
       {
           *PUCREGBUFFER++ = ( UCHAR ) (
USREGHOLDINGBUF[IREGINDEX] >> 8 );
```

```
            *PUCREGBUFFER++ = ( UCHAR ) (
USREGHOLDINGBUF[IREGINDEX] & 0XFF );
            IREGINDEX++;
            USNREGS--;
        }
        BREAK;

        /* UPDATE CURRENT REGISTER VALUES WITH NEW VALUES FROM THE
PROTOCOL STACK. */
        CASE MB_REG_WRITE:
            WHILE( USNREGS > 0 )
            {
                 USREGHOLDINGBUF[IREGINDEX] = *PUCREGBUFFER++ << 8;
                 USREGHOLDINGBUF[IREGINDEX] |= *PUCREGBUFFER++;
                 IREGINDEX++;
                 USNREGS--;
            }
   }
    }
    ELSE
    {
         ESTATUS = MB_ENOREG;
    }
    RETURN ESTATUS;
}


EMBERRORCODE
EMBREGCOILSCB( UCHAR * PUCREGBUFFER, USHORT USADDRESS, USHORT
USNCOILS, EMBREGISTERMODE EMODE )
{
    RETURN MB_ENOREG;
}

EMBERRORCODE
EMBREGDISCRETECB( UCHAR * PUCREGBUFFER, USHORT USADDRESS, USHORT
USNDISCRETE )
{
    RETURN MB_ENOREG;
}
```

ПРИЛОЖЕНИЕ Б

Алгоритм работы лабораторной установки