МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра информационных систем и технологий

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Микропроцессорные средства и системы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Проектирование микропроцессорных устройств на базе однокристальных микроконтроллеров»

|  |  |
| --- | --- |
| Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | Н.А. Бурмистров |
|  |  |
| Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | В.Г. Иоффе |
|  |  |

САМАРА 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра информационных систем и технологий

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Обучающемуся Бурмистрову Никите Александровичу группы 6403-090301D

Тема проекта: ***«***Проектирование микропроцессорных устройств на базе однокристальных микроконтроллеров» в соответствии заданием:

Разработать контроллер, реализующий следующие функции: опрос состояния матрицы 4Х4 («сухие» контакты), включение соответствующего канала АЦП (номер канала определяется номером контакта), разрядность которого 10 бит, быстродействие – не менее 100 мкс, входное напряжение не более 2,56 В. В каждом канале необходимо контролировать Nmini≤Ni≤Nmaxi, где Nmini, Nmaxi – соответственно минимальное и максимальное значение i-того канала, а Ni –текущее значение. Nmini, Nmaxi хранятся в ПЗУ или EEPROM. При выходе за указанные пределы включить соответствующие светодиоды. Окончание измерения индицируется сигналом (отрицательный импульс 10 мкс).

По внешнему запросу прерывания данные передать в приемник через интерфейс RS-232 на скорости 19200 б/с. Данные передаются в следующем формате – номер канала, результат измерения в двоичной форме. Если результат измерения не соответствует заданному диапазону, то вместо значения передать код $С000.

|  |  |  |
| --- | --- | --- |
| Планируемые результаты освоения образовательной программы (компетенции) | Планируемые результаты курсового проектирования | Содержание задания |
| ПК-1 Способен обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности  ПК-1.4. Проектирует информационные системы, разрабатывает программу работ по экспериментальной проверке ее эффективности | Знать: технические характеристики микроконтроллеров и инструментальных средств.  Уметь: проектировать устройства на микроконтроллерах.  Владеть: навыками  решения задач интегрирования в систему микропроцессорных устройств и их комплексной отладки с целью повышения эффективности | 1.Анализ полученного задания  2. Разработка структурной организации  3.Анализ алгоритмов обработки и управления  4.Разработка принципиальной электрической схемы  5.Выбор инструментальных средств и языка программирования  6. Разработка программы  7.Тестирование разработанного контроллера  8. Разработка инструкции по эксплуатации  9. Заключение |

Дата выдачи задания 5 сентября 2022 г.

Срок представления на кафедру пояснительной записки 24 декабря 2022 г.

Руководитель курсового проекта

доцент каф. ИСТ к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Г. Иоффе

*(подпись)*

Задание принял к исполнению

обучающийся группы № 6403-090301D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Бурмистров

*(подпись)*

РЕФЕРАТ

Пояснительная записка к курсовому проекту: N c., N рисунков, N таблица, N источника, 3 приложения.

МИКРОКОНТРОЛЛЕР, ДАТЧИК, СВЕТОДИОД, ГЕНЕРАТОР, ПРИЕМОПЕРЕДАТЧИК RS-232, ИЗМЕРЕНИЕ ЧАСТОТЫ, ДВОИЧНО-ДЕСЯТИЧНАЯ ФОРМА ЧИСЛА.

В соответствии с техническим заданием спроектирована система на основе ОМК ATmega8535 семейства AVR фирмы Atmel, предназначенная для принятия сигналов от кнопок, принятие данных с портов SPI и RS232, для сигнализации о неисправности в случае срыва приема данных. Разработаны алгоритмы функционирования, структурная схема, а также программное обеспечение.

В соответствии с техническим заданием спроектирована система на основе ОМК ATmega8535 семейства AVR фирмы Atmel, предназначенная для измерения напряжения с каналов, для сигнализации выхода напряжений за пределы минимальных и максимальных значений, для передачи данных через интерфейс RS232. Разработаны алгоритмы функционирования, структурная схема, а также программное обеспечение.

Система реализована и отлажена средствами Proteus 8 Professional.

Выполнена отладка и проверка системы на работоспособность

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧНИЙ, ЕДИНИЦ И ТЕРМИНОВ

МПС – микропроцессорная система;

МПУ – микропроцессорное устройство;

ОМК – однокристальный микроконтроллер;

ПБ – процессорный блок;

ПЗУ (ROM) – постоянное запоминающее устройство;

ОЗУ (RAM) – оперативно запоминающее устройство;

ГСА – граф-схема алгоритма;

Рг – регистр;

ШД – шина данных;

КГ – кварцевый генератор;

ИОН – источник опорного напряжения

СТ – счетчик таймер.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc122262711)

[1 Структурная организация 8](#_Toc122262712)

[2 Алгоритмы обработки и управления 11](#_Toc122262713)

[3 Выбор элементной базы 15](#_Toc122262714)

[3.1 Выбор ОМК 15](#_Toc122262715)

[3.2 Выбор приемопередатчика 16](#_Toc122262716)

[3.3 Выбор мультиплексора 16](#_Toc122262717)

[5 Программное обеспечение 20](#_Toc122262718)

[6 Инструкция по эксплуатации контроллера 22](#_Toc122262719)

[7 Алгоритм тестирования 23](#_Toc122262720)

[ЗАКЛЮЧЕНИЕ 39](#_Toc122262721)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 40](#_Toc122262722)

[ПРИЛОЖЕНИЕ А – Принципиальная схема 41](#_Toc122262723)

[ПРИЛОЖЕНИЕ Б – Спецификация элементов 42](#_Toc122262724)

[ПРИЛОЖЕНИЕ В – Листинг программы 43](#_Toc122262725)

# ВВЕДЕНИЕ

В рамках курсового проекта необходимо реализовать систему, позволяющую получать данные с четырех источников в последовательном коде и формирующую сигнал о неисправности системы, а также произвести тестирование программы и создать пакет необходимой документации.

В рамках курсового проекта необходимо реализовать систему, позволяющую измерять напряжение с одного из 15 каналов по нажатию на соответствующую кнопку на клавиатуре, с последующим выводом результата на терминал и формировать сигнал при завершении измерения.

Для решения поставленной задачи эффективно использовать МПС, реализуемую на базе однокристального микроконтроллера (ОМК).

ОМК обладают многими преимуществами – небольшая стоимость и энергопотребление, а также высоким функционалом, достаточным для поставленной задачи. ОМК могут содержать в своем составе порты ввода/вывода, аналого-цифровые и цифро-аналоговые преобразователи, счетчики-таймеры, аналоговые компараторы и другие периферийные устройства, что избавляет от необходимости их отдельной покупки. Высокая скорость обработки данных позволяет использовать их для решения инженерных задач широкого спектра. В то время как разработка МПС на базе микропроцессоров требует применения большого числа микросхем различного назначения, что существенно увеличивает стоимость, габариты, энергопотребление и снижает надежность контроллера.

# 1 Структурная организация

Для рассмотренного выше задания необходимо ПБ, ПЗУ, ОЗУ, блок обработки прерываний, счетчики, индикаторы, мультиплексор, регистры ввода/вывода, интерфейс RS-232.

Для измерения напряжения с определенного канала требуется матричная клавиатура, вмещающая в себе 15 кнопок, по нажатию на которые будут идентифицировать определенный канал. Для постоянного опроса матричной клавиатуры потребуется счетчик таймер и блок прерываний.

Для измерения напряжений с 8 каналов и более потребуется мультиплексор. Управление мультиплексором может осуществляться с помощью восьмиразрядного регистра выхода.

Управление индикаторами может выполняться восьмиразрядным выходным регистром.

Передача информации по интерфейсу RS-232 может быть реализована асинхронным приемопередатчиком UART(USART) и преобразователем сигнала MAX-232.

Практически требуемый объем ОЗУ и ПЗУ может быть определен экспериментально с помощью симулятора после выбора ОМК и разработки программного обеспечения. Тогда можно выбрать новую конфигурацию ОМК с целью сокращения его стоимости.

Для обработки внешних управляющих сигналов контроллер должен включать в себя блок обработки прерываний.

Объем ПЗУ требуемый для хранения констант может быть определен на основании следующих расчетов:

- хранение минимальных значений напряжений на каналах – 15x8 = 120 байт;

- хранение максимальных значений напряжений на каналах – 15x8 = 120 байт;

- хранение измерений напряжений с каналов– 15x8 = 120 байт;

Параметры процессорного блока ПБ определяются особенностями алгоритмов обработки и требуемым быстродействием.

Структура контроллера для рассмотренного примера представлена на рисунке 1.

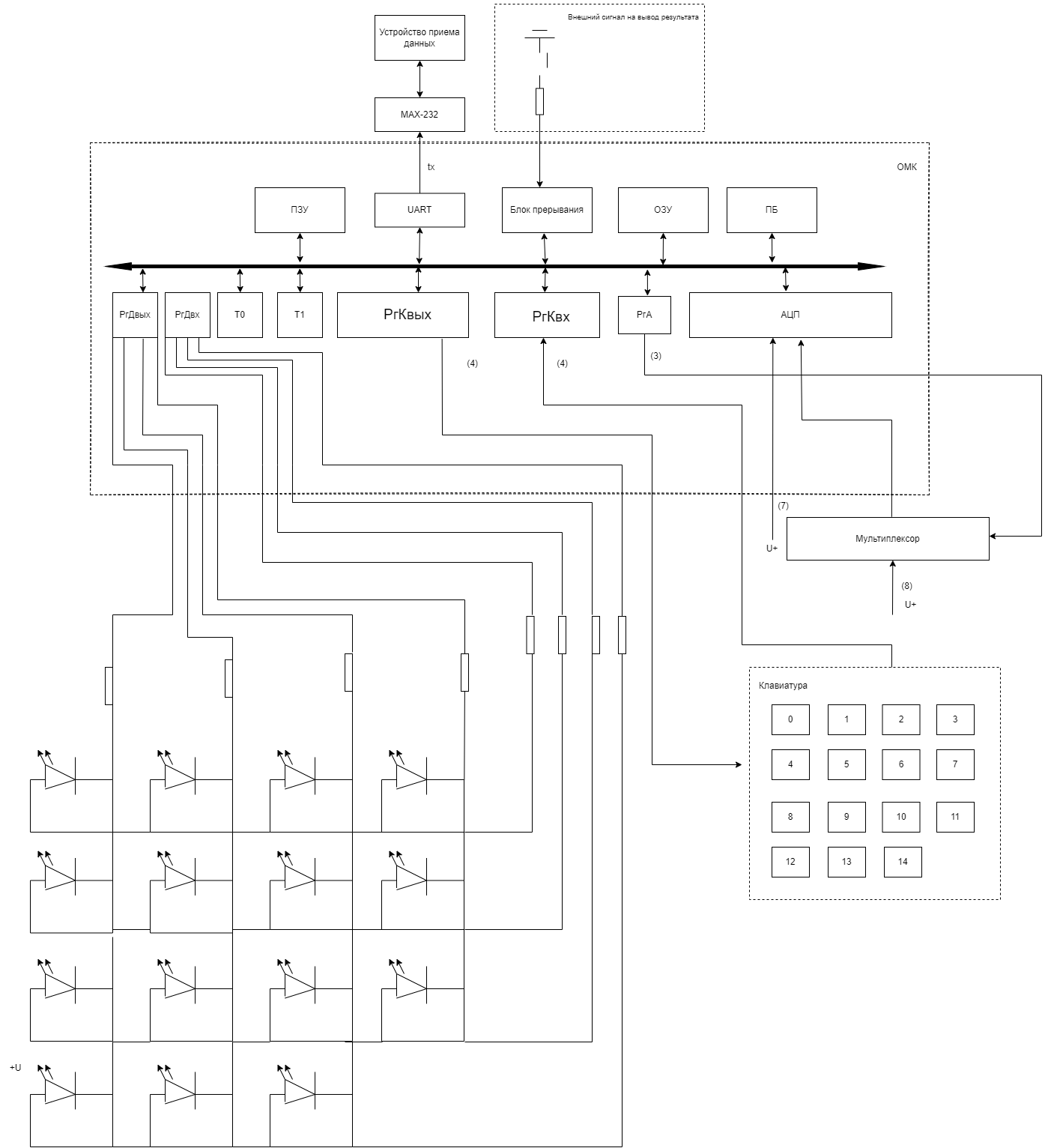


Рисунок 1 – Структурная схема контроллера

Схема состоит из следующих элементов:

* ПБ (процессорный блок) ­­–­­­­­­­ выполняет основные функции по обработку и управлению;
* ОЗУ (оперативное запоминающее устройство) – устройство, хранящее входных и выходных данных, а также промежуточных результатов;
* ПЗУ (постоянное запоминающее устройство) – устройство, хранящее программы обработки данных и запуска устройства.
* Блок обработки прерывания – элемент, который позволяет прервать обработку программы для выполнения подпрограммы;
* T0 (счетчик таймер) – устройство, предназначенное для опроса клавиатуры по переполнению;
* T1 (счетчик таймер) – устройство, предназначенное для генерации прерывания по захвату;
* РгДВых (регистр выхода диодов) – регистр, предназначенный для генерации выходных сигналов на матрицу диодов ­
* РгДВх (регистр входа диодов) – регистр, предназначенный программной реализации включения диодной матрицы
* Устройство приема данных ­ – устройство, которое принимает данные по MAX232
* USART(UART) – последовательный порт, через который с помощью MAX232 осуществляется отправка данных на устройство приема данных
* Мультиплексор – устройство, позволяющая увеличить количество линий для измерения на АЦП
* РгКвх (регистр входа клавиатуры) – регистр, получить адрес нажатой клавиши на клавиатуре
* РгКвых (регистр выхода клавиатуры) – регистр, позволяющий программно реализовать опрос клавиатуры
* РгА (регистр адреса) – регистр, позволяющий указывать мультиплексору какой канал нужно пропускать
* Клавиатура – устройство, предназначенное для выбора нужного канала для измерения на нем напряжения.

# 2 Алгоритмы обработки и управления

Разработка алгоритма функционирования МПУ выполняется на основе технического задания и структурной схемы. Одним из наиболее распространенных средств разработки алгоритмов является проектирование на основе обобщенной граф-схемы алгоритма ГСА.

Граф-схема алгоритма представляется совокупностью последовательно исполняемых операторов: так называемых операторных отображающих преобразование данных, и условных вершин, отражающих проверки входов и результатов исполнения предыдущих шагов с целью выбора пути продолжения исполнения.

Алгоритм, работающий в соответствии с заданием по данному курсовому проекту, выглядит следующим образом:

1. Указание диапазона между минимальным и максимальным напряжением для каждого из 15 каналов напряжения;
2. Инициализация всех блоков микроконтроллера;
3. Опрос клавиатуры 4x4;
4. Измерение напряжение на соответствующем канале;
5. Подача отрицательного импульса по завершению измерения;
6. При выходе за диапазон измерения включить соответствующий светодиод;
7. По сигналу передать значение через интерфейс RS-232;

На рисунке 2 представлена обобщенная граф-схема алгоритма.



Рисунок 2 – Обобщённая граф схема алгоритма

Алгоритм «бегущий ноль» реализован следующим образом:

1. Обход матрицы по ширине и подача сигнала на соответствующую колонку
2. Проверка нажатия клавиши в колонке
3. Установление индекса в переменную, соответствующего клавише

На рисунке 3 представлена граф-схема алгоритма получения индекса нажатой клавиши.



Рисунок 3 – Алгоритм опроса клавиатуры

На схеме использовались обозначения:

width – ширина матричной клавиатуры;

length – высота матричной клавиатуры;

i – вспомогательная переменная для нахождения индекса нажатой кнопки;

j – вспомогательная переменная для нахождения индекса нажатой кнопки;

channelNumber – номер нажатой клавиши;

canReadChannel – была ли нажата клавиша на клавиатуре;

Разрядность данных:

i [7:0], j[7:0], channelNumber[7:0], canReadChannel[1:0].

Алгоритм передачи данных по интерфейсу USART реализован следующим образом:

1. Запись данных в переменную data;
2. Ожидание возможности записи данных в сдвиговый регистр;
3. Передача значения из data в сдвиговый регистр;
4. Ожидание завершения передачи ;

На рисунке 4 представлена граф-схема алгоритма передача данных по USART.



Рисунок 4 – Алгоритм передачи данных по USART

На схеме использовались обозначения:

data – передаваемые данные.

Разрядность данных:

data [7:0].

# 3 Выбор элементной базы

## 3.1 Выбор ОМК

Выбранный в соответствии с техническим заданием ОМК должен обладать функциональной полнотой, чтобы минимизировать число внешних компонентов, необходимых для реализации требуемого МПУ, иметь минимальное количество неиспользуемых линий ввода-вывода, минимальное энергопотребление и стоимость.

По заданию требуется подобрать ОМК, в состав которого должны входить следующие элементы:

* ПБ;
* ОЗУ;
* ПЗУ;
* 31 линия ввода-вывода
* 10 разрядный АЦП
* Интерфейс USART
* Блок управления прерывания

Для данного проекта выбран ОМК ATMEGA8535, который обладает следующими параметрами:

* тактовой частотой до 16 МГц;
* объемом памяти программ 8 кбайт;
* наличием EEPROM 512 байт;
* наличием встроенного интерфейса USART;
* напряжением питания в диапазоне от 4.5 до 5.5 В;
* количеством 32 выходов/входов.
* наличием 16 битного таймера;
* наличием INT0, INT1 и INT2.

Данный ОМК соответствует требованиям технического задания, обладает функциональной полнотой, минимизирует число внешних компонентов, необходимых для реализации МПУ.

Таблица 1 – Распределение линий ОМК

|  |  |
| --- | --- |
| Наименование линии | Назначение |
| PC0 – PC7 | Линии подключения клавиатуры |
| PA0 – PA7 | Входы АЦП |
| PB0 – PB7 | Линии подключения светодиодов |
| PD0 – PD1 | Линии USART |
| RESET | Линия сброса |
| PD2 | Вход внешнего запроса прерывания |
| PD4 | Выход отрицательно импульса |
| PD5-PD7 | Линии управления мультиплексором |

Настройки периферийных устройств:

UART – формат обмена данных 8 бит и бит паритета, режим работы асинхронный, скорость передачи данных 19200 бит/c, способ вывода – программный.

## 3.2 Выбор приемопередатчика

Приемопередатчик должен осуществлять приме данных по интерфейсу UART и передавать полученные данные по интерфейсу RS-232. Таким приемником является MAX-232.

Режим работы данного приёмопередатчика является дуплекс, напряжение питания находится в диапазоне от 4.5 до 5.5 В.

## 3.3 Выбор мультиплексора

Одной из основных характеристик при выборе мультиплексора является его разрядность, так как в задании указано, что измерение производится с 15 каналов , но 7 из них идут напрямую в АЦП, то соответственно разрядность мультиплексора должна быть равна 8, чтобы можно было измерить со всех каналов напряжение. Также важно чтобы напряжение работы мультиплексора совпадало с напряжением питания микроконтроллера.

Данным требованиям удовлетворяет восьмиканальный мультиплексор 74HC4051.

## 3.4 Выбор ИОН

Для корректного измерения входного напряжения с помощью АЦП не всегда подходит внутренний источник, поэтому нужно иметь внешний стабильный источник опорного напряжения. По заданию максимальным измеряем напряжением является 2.55 B, следовательно нужно брать источник, который работает в данном диапазоне.

Выберем ИОН AD780, выходное напряжение которого составляет 2,5 или 3,0 B и программируется коммутацией внешних выводов. Погрешность данного источника опорного напряжения составляет ±0.03 %, что указывает на высокую стабильность элемента.

# 4. Принципиальная электрическая схема

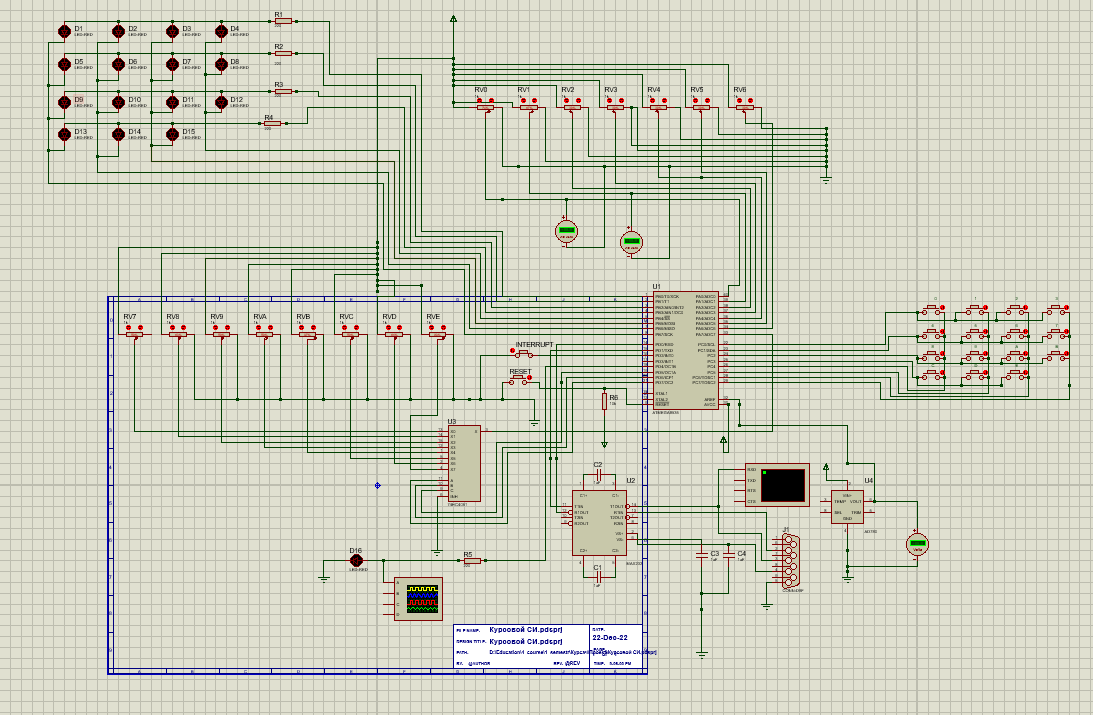


Рисунок 5 – Принципиальная схема

Центральным блоком на схеме является ОМК ATMEGA8535, который работает на частоте 4МГц. Выбор частоты обусловлен высокой частой опроса клавиатуры для того, чтобы ни одно из нажатий не было пропущено.

При включении питания автоматически выполняется сброс микроконтроллера U1, выполненного на микросхеме ATmega8535, что приводит к инициализации всех основных структур программного обеспечения и началу работы микроконтроллера.

В случае некорректного запуска работы контроллера предусмотрена внешняя схема сброса, состоящая из резистора R5 и кнопки RESET.

После инициализации всех устройств начинается опрос матричной клавиатуры, который реализован с помощью алгоритма «бегущий ноль», предназначенный для идентификации нажатой кнопки на клавиатуре.

После нажатия на кнопку на клавиатуре подается сигнал микроконтроллеру с какого канала нужно измерить напряжение, которое регулируется с помощью переменного резистора на каждом из каналов.

Если была нажата клавиша с индексом больше 7, то линиях PD4 – PD7 поддается сигнал на мультиплексор U3 для передачи напряжение с соответствующего канала на PA7 линию приема напряжения АЦП.

По завершению измерения на линию PD4 поддается отрицательный сигнал в 10мкс. В случае если результат измерения выходит за пределы между минимальным и максимальным значением, указывающее в программе, то на линии PB0 – PB7 подается напряжение для включение соответствующего светодиода.  
 Для вывода значения нужно поддать сигнал микроконтроллеру с помощью нажатия кнопки INTERRUPT, вызывающее программное внутреннее прерывание INT0. Стоит отметить то, что программно реализовано так, что если не было завершено измерение, то прерывание не будет обрабатываться.

Данные передаются по USART через линию PD1 на скорости 19200 бит/c. Так как у ОМК нет аппаратной реализации RS-232, данные передаются по USART, а потом преобразуются в RS-232 посредством интерфейса MAX-232.

# 5 Программное обеспечение

В качестве языка программирования выбран язык C, так как он в себе сочетает черты языков низкого и высокого уровня. На данный момент он является одним из популярных и распространённых языков программирования для микроконтроллеров.

В качестве инструментального средства для разработки и отладки программного обеспечения ОМК был выбран Proteus 8 Этот пакет программ автоматизированного проектирования электронных схем. Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов.

При разработке программ под МК AVR на СИ, может использоваться любая среда разработки, на выходе которой формируются файлы с расширениями, поддерживаемыми Proteus 8 (HEX, ASM, COF, ELF/DWARF2,

В качестве компилятора использовался WinAVR поддерживающего язык C, обеспечивая полный цикл разработки для AVR.

Листинг программы приведен в приложении В.

Память программы занимает 2341 байт из 8192 байт.

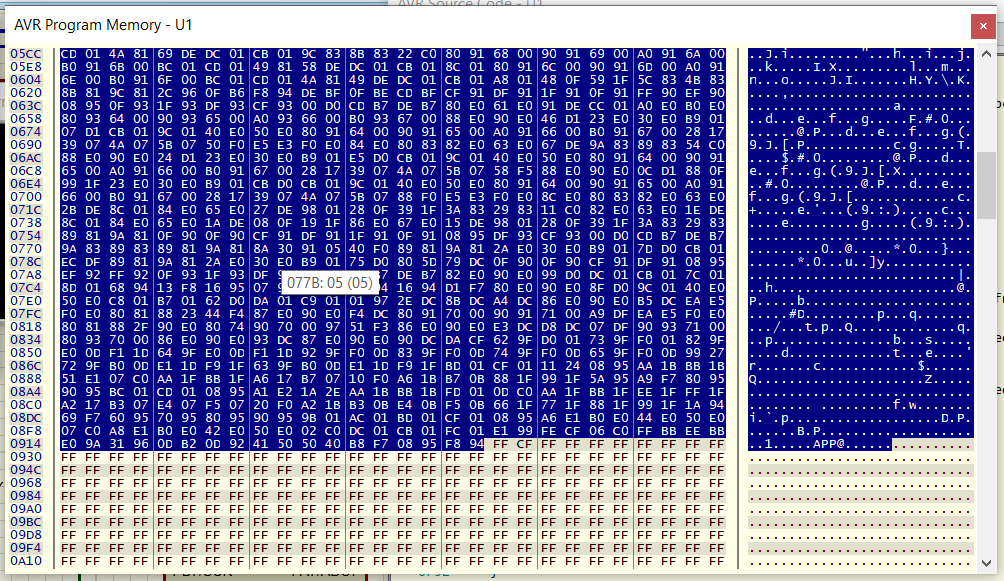


Рисунок 6 – Занимая память программ

Память EEPROM занимает 71 байт из 512 байт.

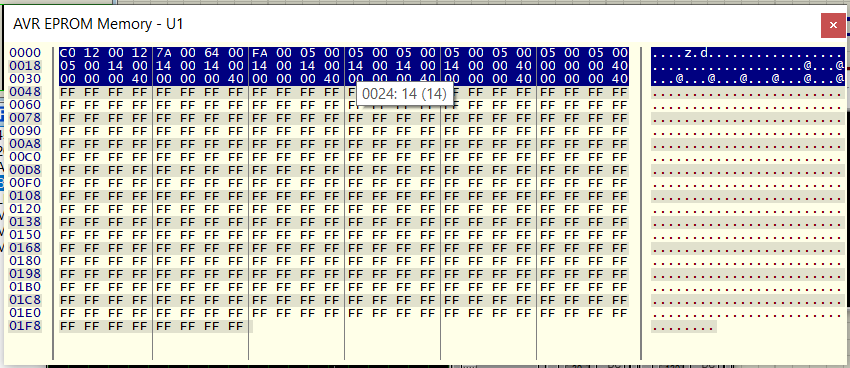


Рисунок 7 – Занимая память EEPROM

# 6 Инструкция по эксплуатации контроллера

Перед запуском контроллера необходимо установить значения датчиков Sensor0 – Sensor7, задать коэффициенты датчиков A и B, и Qдоп.

В момент первого включения контроллера все каналы будут в открытом состоянии, для считывания более достоверных значений с датчиков.

Чтобы произвести считывание значения расхода топлива с датчиков необходимо нажать на кнопку SB1. После ее нажатия в память устройства запишется значение расхода топлива и откроется/закроется нужный канал топливной системы, это будет видно по свечению диодов D1 – D3. Стоит отметить, что при повторном нажатии на кнопку SB1 заново откроет все каналы и произведется повторное измерение значения расхода топливной системы, после измерения откроет/закроет нужный канал.

Для вывода значения результата расхода топлива необходимо нажать на кнопку SB2. Результат передастся по интерфейсу RS-485. Если значение расхода топлива еще не считано, то кнопка SB2 будет не активной.

В случаи зависания контроллера или других ошибках в его работе, предусмотрена кнопка SB3 – сброс. После ее нажатия устройство перезапуститься.

# 7 Алгоритм тестирования

Тестовый пример 1. Пусть расход топлива на входе будет меньше 1/3Qдоп, тогда после считывания результата будет активен только первый канал.

При подсчете значений с датчиков используются следующие константы: f\_mcu = 8000000 (значение частоты работы ОМК), accuracy = 100 (коэффициент необходимый для достижения требуемой точности).

Необходимо установить следующие данные: q\_opt = 2500 (максимально-возможное значение расхода эквивалентен значению 250.0 Гц), массив значений коэффициента mass\_a заполнить значениями 5 (это эквивалентно коэффициенту 1/2), массив значений коэффициентов mass\_b заполнить значениями 20 (это эквивалентно коэффициенту 2), данные с значениями датчиков представлены в таблице 1.

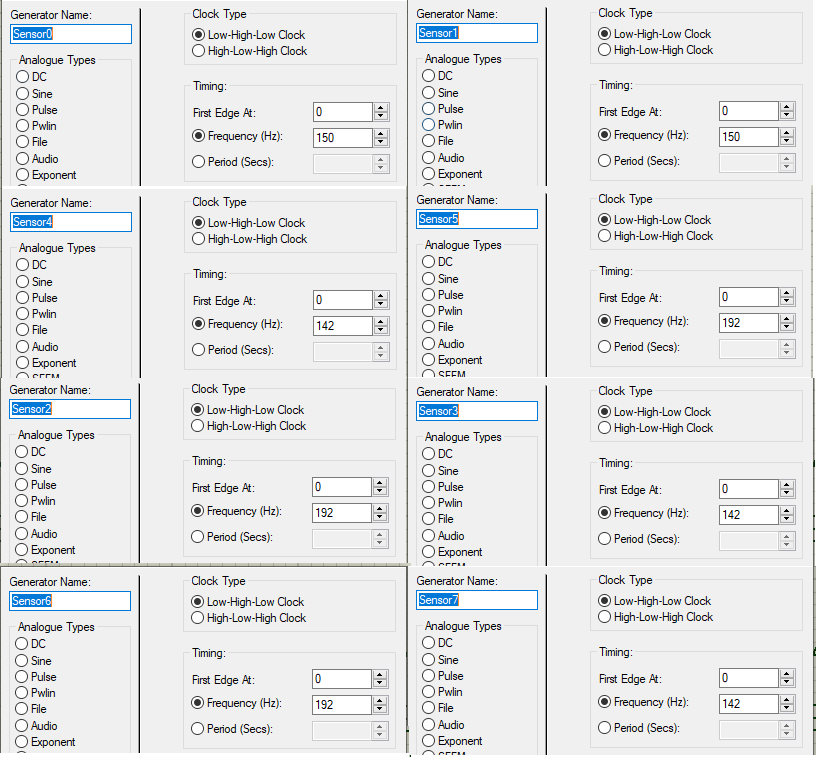


Рисунок 8 – Значения датчиков

Таблица 1 – Значения датчиков

|  |  |
| --- | --- |
| Датчик | Значение |
| Sensor0 | 150 Гц |
| Sensor1 | 150 Гц |
| Sensor2 | 192 Гц |
| Sensor3 | 142 Гц |
| Sensor4 | 142 Гц |
| Sensor5 | 192 Гц |
| Sensor6 | 192 Гц |
| Sensor7 | 142 Гц |

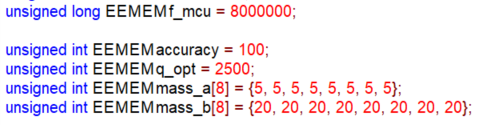


Рисунок 9 – Значение коэффициентов и Qдоп

По нажатию кнопки SB1 найдем значение расхода на входе (с датчиков Sensor0 и Sensor1).

Сначала контроллер найдет значение периода Sensor0, на рисунке 10 представлено значение периода Sensor0.



Рисунок 10 – Значение периода Sensor0

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 11.



Рисунок 11 – Значение частоты Sensor0

Аналогично найдем значение периода Sensor1, на рисунке 12 представлено значение периода Sensor1.



Рисунок 12 – Значение периода Sensor1

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 13.



Рисунок 13 – Значение периода Sensor1

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 14.



Рисунок 14 – Значение value\_opt

Так как , значение на входе будет равно:

Посмотри какое значение определил контроллер, рисунок 15.



Рисунок 15 – Преобразованное значение частоты входного потока.

После нахождение значения , определим какие из каналов нужно закрыть для более точного измерения расхода. Так как:

Таким образом необходимо закрыть второй и третий канал, а результатом расхода топлива будет значение с первого канала (с датчиков Sensor2 и Sensor3).

Получим период Sensor2, представлено на рисунке 16.



Рисунок 16 – Период Sensor2.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 17.



Рисунок 11 – Значение частоты Sensor2

Получим период Sensor3, представлено на рисунке 16.



Рисунок 16 – Период Sensor3.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 17.



Рисунок 17 – Значение частоты Sensor3

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 18.



Рисунок 18 – Значение value\_opt

Так как , значение на первом канале будет равно:

Посмотри какое значение определил контроллер, рисунок 19.



Рисунок 19 – Значение расхода топлива

Таким образом значение расхода топлива будет . Выведем результат расхода топлива на терминал, нажав кнопку SB2 (рисунок 20).

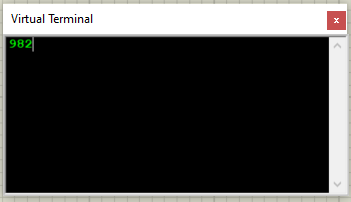


Рисунок 20 – Результат значения расхода топлива

В консоли видно значение 982, что равно 98.2, таким образом контроллер определил верное значение расхода топлива с допустимой погрешностью.

Тестовый пример 2. Пусть расход топлива на входе будет больше 1/3Qдоп и меньше 2/3Qдоп, тогда после считывания результата будут активны каналы первый и второй.

Данные с значениями датчиков представлены в таблице 2.

Таблица 2 – Значения датчиков

|  |  |
| --- | --- |
| Датчик | Значение |
| Sensor0 | 192 Гц |
| Sensor1 | 142 Гц |
| Sensor2 | 192 Гц |
| Sensor3 | 142 Гц |
| Sensor4 | 142 Гц |
| Sensor5 | 192 Гц |
| Sensor6 | 192 Гц |
| Sensor7 | 142 Гц |

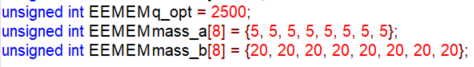


Рисунок 21 – Значение коэффициентов и Qдоп

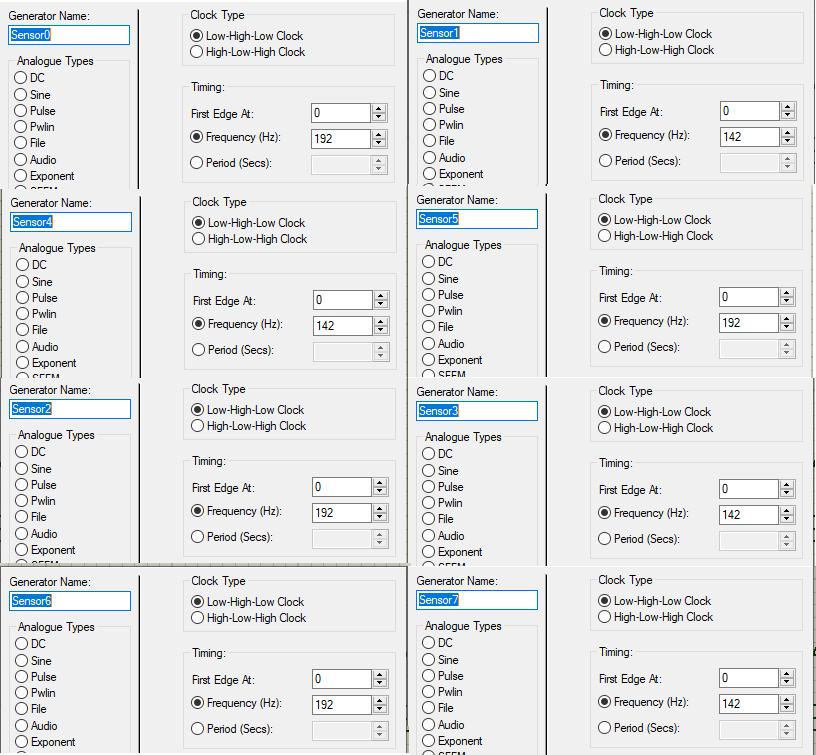


Рисунок 22 – Значения датчиков

По нажатию кнопки SB1 найдем значение расхода на входе (с датчиков Sensor0 и Sensor1).

Сначала контроллер найдет значение периода Sensor0, на рисунке 23 представлено значение периода Sensor0.



Рисунок 23– Значение периода Sensor0

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 24.



Рисунок 24 – Значение частоты Sensor0

Аналогично найдем значение периода Sensor1, на рисунке 25 представлено значение периода Sensor1.



Рисунок 25 – Значение периода Sensor1

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 26.



Рисунок 26 – Значение периода Sensor1

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 27.



Рисунок 27 – Значение value\_opt

Так как , значение на первом канале будет равно:

Посмотри какое значение определил контроллер, рисунок 28.



Рисунок 28 – Значение

После нахождение значения , определим какие из каналов нужно закрыть для более точного измерения расхода. Так как:

Таким образом необходимо закрыть третий канал, а результатом расхода топлива будет значение с первого и второго каналов (с датчиков Sensor2 и Sensor3, Sensor4 и Sensor5).

Определим значение расхода на первом канале.

Получим период Sensor2, представлено на рисунке 29.



Рисунок 29 – Период Sensor2.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 30.



Рисунок 30 – Значение частоты Sensor2

Получим период Sensor3, представлено на рисунке 31.



Рисунок 31 – Период Sensor3.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 32.

  
Рисунок 32 – Значение частоты Sensor3

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 33.



Рисунок 33 – Значение value\_opt

Так как , значение на первом канале будет равно:

Определим расход на втором канале:

Получим период Sensor4, представлено на рисунке 34.



Рисунок 34 – Период Sensor4

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 35.



Рисунок 35 – Значение частоты Sensor4

Получим период Sensor5, представлено на рисунке 36.



Рисунок 36 – Период Sensor5

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 37.



Рисунок 37 – Значение частоты Sensor5

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 38.



Рисунок 38 – Значение value\_opt

Так как , значение на первом канале будет равно:

Таким образом значение расхода топлива будет .

Выведем результат расхода топлива на терминал, нажав кнопку SB1 (рисунок 39).

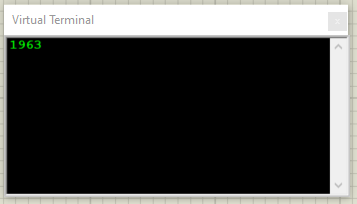


Рисунок 39 – Результат значения расхода топлива

В консоли видно значение 1963, что равно 196.3, таким образом контроллер определил верное значение расхода топлива с допустимой погрешностью.

Тестовый пример 3. Пусть расход топлива на входе будет больше 2/3Qдоп, тогда после считывания результата будут активны все каналы.

Данные с значениями датчиков представлены в таблице 3.

Таблица 3 – Значения датчиков

|  |  |
| --- | --- |
| Датчик | Значение |
| Sensor0 | 360 Гц |
| Sensor1 | 290 Гц |
| Sensor2 | 192 Гц |
| Sensor3 | 142 Гц |
| Sensor4 | 142 Гц |
| Sensor5 | 192 Гц |
| Sensor6 | 192 Гц |
| Sensor7 | 142 Гц |

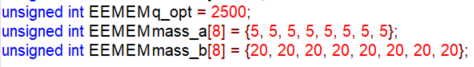


Рисунок 40 – Значение коэффициентов и Qдоп

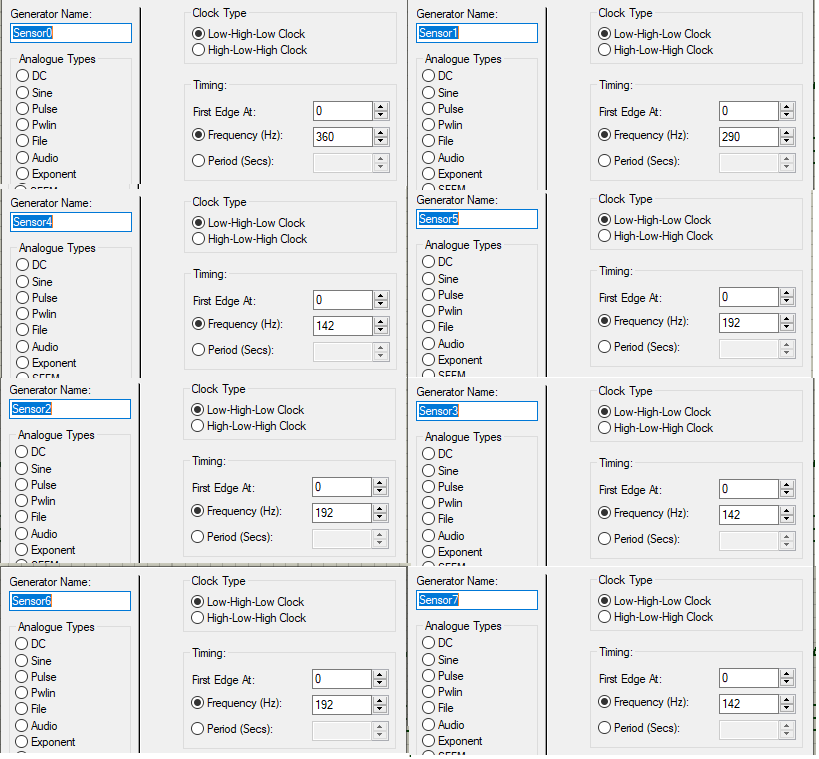


Рисунок 41 – Значения датчиков

По нажатию кнопки SB0 найдем значение расхода на входе (с датчиков Sensor0 и Sensor1).

Сначала контроллер найдет значение периода Sensor0, на рисунке 42 представлено значение периода Sensor0.



Рисунок 42– Значение периода Sensor0

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 43.



Рисунок 43 – Значение частоты Sensor0

Аналогично найдем значение периода Sensor1, на рисунке 44 представлено значение периода Sensor1.



Рисунок 44 – Значение периода Sensor1

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 45.



Рисунок 45 – Значение периода Sensor1

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 46.



Рисунок 46 – Значение value\_opt

Так как , значение на первом канале будет равно:

Посмотри какое значение определил контроллер, рисунок 47.



Рисунок 47 – Значение

После нахождение значения , определим какие из каналов нужно закрыть для более точного измерения расхода. Так как:

Таким образом все каналы остаются открыты, а результатом расхода топлива будет значения всех каналов (с датчиков Sensor2 и Sensor3, Sensor4 и Sensor5, Sensor6 и Sensor7).

Определим значение расхода на первом канале.

Получим период Sensor2, представлено на рисунке 48.



Рисунок 48 – Период Sensor2.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 49.



Рисунок 49 – Значение частоты Sensor2

Получим период Sensor3, представлено на рисунке 50.



Рисунок 50 – Период Sensor3.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 51.



Рисунок 51 – Значение частоты Sensor3

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 52.

  
Рисунок 52 – Значение value\_opt

Так как , значение на первом канале будет равно:

Определим расход на втором канале:

Получим период Sensor4, представлено на рисунке 53.



Рисунок 53 – Период Sensor4

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 54.



Рисунок 54 – Значение частоты Sensor4

Получим период Sensor5, представлено на рисунке 55.



Рисунок 55 – Период Sensor5

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 56.



Рисунок 56 – Значение частоты Sensor5

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 57.



Рисунок 57 – Значение value\_opt

Так как , значение на первом канале будет равно:

Определим значение расхода на третьем канале.

Получим период Sensor6, представлено на рисунке 58.



Рисунок 58 – Период Sensor6.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 59.



Рисунок 59 – Значение частоты Sensor6

Получим период Sensor3, представлено на рисунке 60.



Рисунок 60 – Период Sensor7.

Из полученного периода контроллер получит частоту датчика, по следующей формуле:

Посмотрим какое значение нашел контроллер, представлено на рисунке 61.



Рисунок 61 – Значение частоты Sensor7

Определим погрешность между датчиками :

Просмотри значение value\_opt определенное контроллером, рисунок 62.

  
Рисунок 62 – Значение value\_opt

Так как , значение на первом канале будет равно:

Таким образом значение расхода топлива будет .

Выведем результат расхода топлива на терминал, нажав кнопку SB1 (рисунок 63).

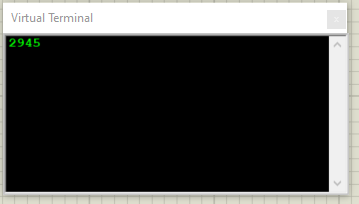


Рисунок 63 – Результат значения расхода топлива

В консоли видно значение 2945, что равно 294.5, таким образом контроллер определил верное значение расхода топлива с допустимой погрешностью.

# ЗАКЛЮЧЕНИЕ

Входе выполнения курсового проекта был разработан контроллер подсчета расхода топлива, алгоритм его работы, структурная и принципиальная схема.

Контроллер расхода топлива построен на микроконтроллере фирмы AVR ATMega 8535. Программа контроллера написана на языке С и скомпилирована с помощью среды WinAVR, симуляция проводилась в среде Proteus.

Частота работы микроконтроллера 8 МГц, интерфейс RS-485 работает на скорости 4800 бит/c

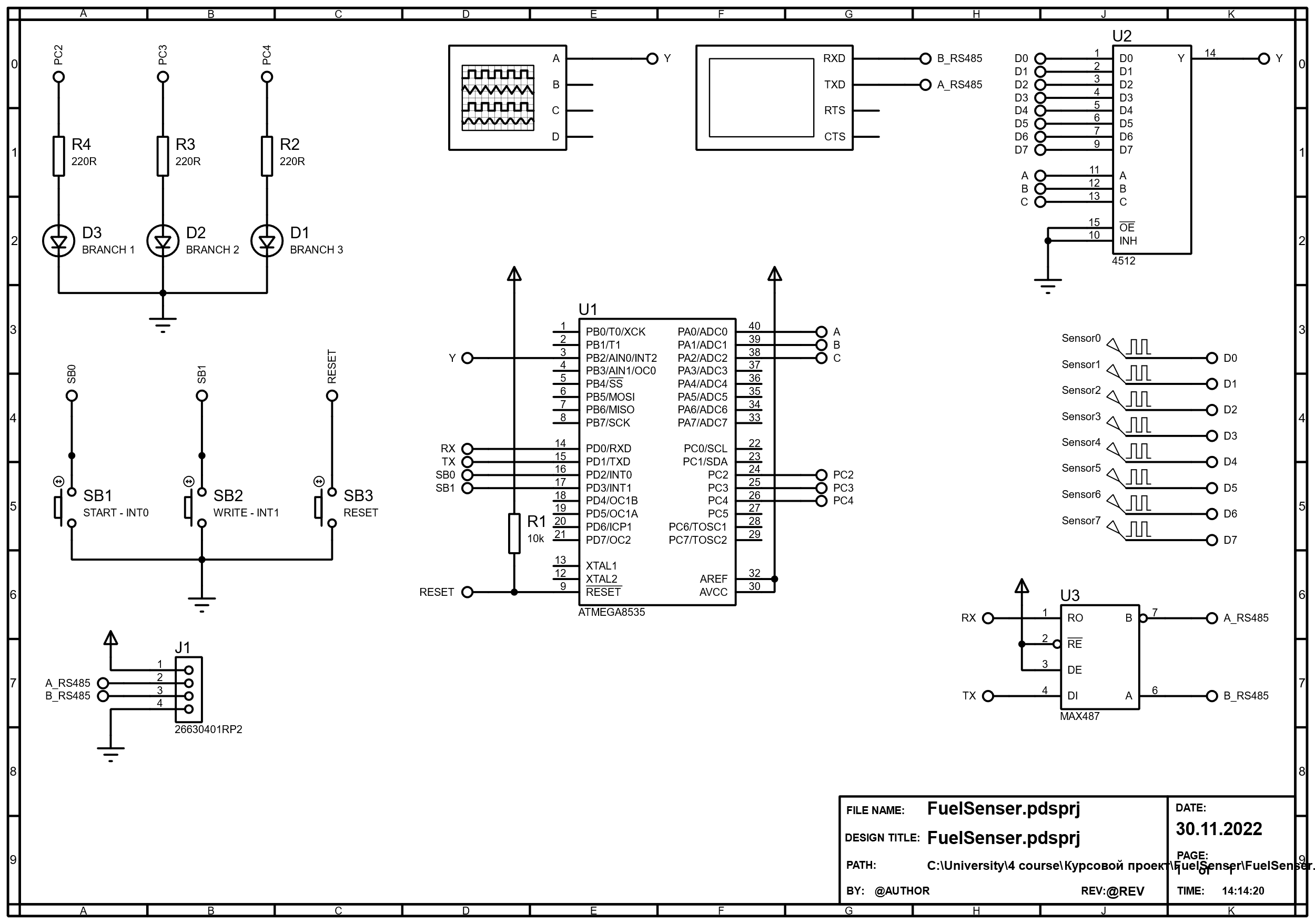
# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Иоффе, В. Г. Структурная организация однокристальных микроконтроллеров. [Электронный ресурс] : [учеб. пособие] / В. Г. Иоффе ; М-во образования и науки Рос. Федерации, Самар. нац. исслед. ун-т им. С. П. Королева (Самар. ун-т). - Самара : Изд-во Самар. ун-та, 2017. URL: http://repo.ssau.ru/handle/Metodicheskie-materialy/Strukturnaya-organizaciya-odnokristalnyh-mikrokontrollerov-Elektronnyi-resurs-ucheb-posobie-70944

2. Проектирование микропроцессорных устройств на базе однокристальных микроконтроллеров [Электронный ресурс] : [метод. указания] сост. В. Г. Иоффе / М-во образования и науки Рос. Федерации, Самар. нац. исслед. ун-т им. С. П. Королева (Самар.ун-т). - Самара : Изд-во Самар. ун-та, 2015. URL:. http://repo.ssau.ru/handle/Metodicheskie-materialy/Proektirovanie-mikroprocessornyh-ustroistv-na-baze-odnokristalnyh-mikrokontrollerov-Elektronnyi-resurs-metod-ukazaniya-70959

3. Разработка и отладка микропроцессорных устройств в виртуальной среде моделирования Proteus [Электронный ресурс] : [метод. указания], сост. В. Г. Иоффе / М-во образования и науки Рос. Федерации,;. Самар. нац. исслед. ун-т им. С. П. Королева (Самар. ун-т). - Самара : Изд-во Самар. ун-та, 2017. URL: http://repo.ssau.ru/handle/Metodicheskie-materialy/Razrabotka-i-otladka-mikroprocessornyh-ustroistv-v-virtualnoi-srede-modelirovaniya-Proteus-Elektronnyi-resurs-metod-ukazaniya-70958

# ПРИЛОЖЕНИЕ А – Принципиальная схема



# ПРИЛОЖЕНИЕ Б – Спецификация элементов

|  |  |  |
| --- | --- | --- |
| Обозначение | Наименование | Кол-во |
|  | Резисторы |  |
| R1 | CF-100, 3 кОм | 1 |
| R2…R4 | CF-100, 220 Ом | 3 |
|  | Диоды |  |
| D1…D3 | GNL-3012GD | 3 |
|  | Кнопки |  |
| SB1…SB3 | KLS7-TS3601 | 3 |
|  | Микросхемы |  |
| U1 | ATMega 8535 | 1 |
| U2 | 4512 | 1 |
| U3 | MAX-487 | 1 |

# ПРИЛОЖЕНИЕ В – Листинг программы

#include <avr/io.h>

#include <stdbool.h>

#include <avr/eeprom.h>

unsigned int EEMEM baud = 4800;

unsigned long EEMEM f\_mcu = 8000000;

unsigned int EEMEM accuracy = 100;

unsigned int EEMEM q\_opt = 2500;

unsigned int EEMEM mass\_a[8] = {5, 5, 5, 5, 5, 5, 5, 5};

unsigned int EEMEM mass\_b[8] = {20, 20, 20, 20, 20, 20, 20, 20};

unsigned long value\_opt = 0;

unsigned long freq\_chanel\_1 = 0;

unsigned long first\_value\_senser = 0;

unsigned long second\_value\_senser = 0;

unsigned int flow = 0;

unsigned int value = 0;

bool read\_impulse = false;

//================= Интерфейс USART ===============

/\*

Инициализация интерфейса USART

\*/

void init\_usart(unsigned int baud)

{

UBRRH = (unsigned char) (baud>>8);

UBRRL = (unsigned char) baud;

UCSRB = (1<<TXEN);

UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1) | (1 << UPM1);

}

/\*

Отправка данных по интерфейсу USART

\*/

void transmit\_usart( unsigned char data )

{

while (!(UCSRA & (1<<UDRE)));

UDR = data;

while (!(UCSRA&(1<<TXC)));

}

//================================================

//===================== Таймер ====================

/\*

Запустить таймер1

\*/

void start\_timer1 ()

{

TCCR1B |= (0 << CS12) | (0 << CS11) | (1 << CS10);

}

/\*

Остановить таймер1

\*/

void stop\_timer1 ()

{

TCCR1B = 0;

TCNT1 = 0;

}

//================================================

//===================== Порты ====================

/\*

Инициализация портов ввода/вывода

\*/

void init\_ports()

{

PORTA = 0;

PORTD = (1 << PD2) | (1 << PD3) | (1 << PD0);

PORTC = (1 << PC2) | (1 << PC3) | (1 << PC4);

DDRA = (1 << DDA0) | (1 << DDA1) | (1 << DDA2);

DDRC = (1 << DDC2) | (1 << DDC3) | (1 << DDC4);

}

//================================================

//=================== Прерывания ===================

/\*

Инициализация прерывания

\*/

void init\_interupt ()

{

MCUCR |= (1 << ISC11 ) | (1 << ISC01);

MCUCSR |= (0 << ISC2);

}

/\*

Активировать прерывания <interupt>

<interupt> - номер прерывания

\*/

void enable\_interupt (unsigned int interupt)

{

GICR |= (1 << interupt);

}

/\*

Деактивировать прерывания <interupt>

<interupt> - номер прерывания

\*/

void disable\_interupt (unsigned int interupt)

{

GICR &= ~(1 << interupt);

}

/\*

Сбросить все прерывания

\*/

void reset\_interupt ()

{

GICR = 0;

}

/\*

Сбросить флаг прерывания <interupt>

<interupt> - номер прерывания

\*/

void reset\_state\_interupt (unsigned int interupt)

{

GIFR |= (1 << interupt);

}

//================================================

/\*

Чтение кол-во импульсов на датчике <number\_sensor>

<number\_sensor> - номер датчика

\*/

unsigned int read\_number\_impulse(unsigned char number\_sensor)

{

read\_impulse = true;

PORTA = number\_sensor;

enable\_interupt(INT2);

while(true)

{

if (GIFR & (1 << INTF2))

{

if (read\_impulse)

{

read\_impulse = false;

reset\_state\_interupt(INTF2);

start\_timer1();

}

else

{

value = TCNT1;

stop\_timer1();

disable\_interupt(INT2);

reset\_state\_interupt(INTF2);

break;

}

}

}

return value;

}

/\*

Получение преобразовоного значения с датчика

<value\_freq> - значение с датчика

<number\_senser> - номер датчика

\*/

unsigned long setup\_value\_senser (unsigned long value\_freq, unsigned char number\_senser)

{

return (value\_freq \* eeprom\_read\_word(mass\_a + number\_senser) + (eeprom\_read\_word(mass\_b + number\_senser) \* eeprom\_read\_word(&accuracy))) / (eeprom\_read\_word(&accuracy));

}

/\*

Определение значения канала <first\_number\_senser> и <second\_number\_senser>

<first\_number\_senser> - номер первого датчика

<second\_number\_senser> - номер второго датчика

\*/

unsigned int read\_freq\_chanel(unsigned char first\_number\_senser, unsigned char second\_number\_senser)

{

first\_value\_senser = eeprom\_read\_dword(&f\_mcu) \* eeprom\_read\_word(&accuracy) / read\_number\_impulse(first\_number\_senser);

second\_value\_senser = eeprom\_read\_dword(&f\_mcu) \* eeprom\_read\_word(&accuracy) / read\_number\_impulse(second\_number\_senser);

value\_opt = (first\_value\_senser > second\_value\_senser ? first\_value\_senser : second\_value\_senser) \* 5 / 100;

if (first\_value\_senser > second\_value\_senser && first\_value\_senser - second\_value\_senser > value\_opt)

{

return setup\_value\_senser(first\_value\_senser, first\_number\_senser);

}

else if (second\_value\_senser > first\_value\_senser && second\_value\_senser - first\_value\_senser > value\_opt)

{

return setup\_value\_senser(second\_value\_senser, second\_number\_senser);

}

else

{

return (setup\_value\_senser(first\_value\_senser, first\_number\_senser) + setup\_value\_senser(second\_value\_senser, second\_number\_senser))/2;

}

}

/\*

Определение значения расхода

\*/

unsigned int processing\_flow(){

freq\_chanel\_1 = read\_freq\_chanel(0, 1);

if (freq\_chanel\_1 <= eeprom\_read\_word(&q\_opt) / 3)

{

PORTC = (1 << PC2) | (0 << PC3) | (0 << PC4);

return (unsigned int) read\_freq\_chanel(2, 3);

}

else if ((eeprom\_read\_word(&q\_opt) / 3) < freq\_chanel\_1 && freq\_chanel\_1 <= ((eeprom\_read\_word(&q\_opt) << 1) / 3))

{

PORTC = (1 << PC2) | (1 << PC3) | (0 << PC4);

return (unsigned int) (read\_freq\_chanel(2, 3) + read\_freq\_chanel(4, 5));

}

else

{

return (unsigned int) (read\_freq\_chanel(2, 3) + read\_freq\_chanel(4, 5) + read\_freq\_chanel(6, 7));

}

}

/\*

Отправка результата

\*/

void write\_flow(unsigned int value)

{

if (value > 9)

{

write\_flow(value / 10);

}

transmit\_usart((value % 10) + 48);

}

/\*

Выполнение программы

\*/

int main()

{

init\_usart(eeprom\_read\_dword(&f\_mcu)/16/eeprom\_read\_word(&baud)-1);

init\_ports();

init\_interupt();

enable\_interupt(INT0);

while (1)

{

if (GIFR & (1 << INTF1)){

reset\_state\_interupt(INTF1);

write\_flow(flow);

}

if (GIFR & (1 << INTF0)){

reset\_state\_interupt(INTF0);

reset\_interupt();

flow = processing\_flow();

enable\_interupt(INT0);

enable\_interupt(INT1);

}

}

return 0;

}