Министерство науки и высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

УСТРОЙСТВО РЕГИСТРАЦИИ И ОБРАБОТКИ ПОТОКОВ ВХОДНЫХ ДАННЫХ

по курсу «Микропроцессорные системы»

Студент гр. ИУ6-76Б _____ Р. М. Аксенов

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В. Я. Хартов

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ-6
_____А.В. Пролетарский
«_2_»_сентября_____2020 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Микропроцессорные системы

Студент___Аксенов Р. (ИУ6-76)_____
(фамилия, инициалы, индекс группы)

Направленность курсовой работы – учебная

График выполнения работы: 25% - 4 нед., 50% - 8 нед., 75% - 12 нед., 100% -16 нед.

Тема курсовой работы Устройство регистрации и обработки потоков входных данных

Разработать МК–систему из 2-х микроконтроллеров ATmega8515, один из которых представляет 2-канальный источник данных, а второй – приемник/регистратор с подключенной к нему внешней оперативной памятью для хранения данных, и пульта оператора (ПО) для ввода команд и отображения результатов обработки.
Разработать алгоритмы и программы для решения следующих задач:
а) по сигналам запроса от внешнего источника регистратор осуществляет прием побайтно входных потоков данных и запись их во внешнюю память, начиная с адреса 0x0000. Первый канал источника формирует 3 потока по 20 чисел, второй – 2 по 30 чисел. Потоки данных представить последовательностями фиксированных чисел, хранимых в постоянной памяти микроконтроллера, используемого как источник данных.
Передача каждого байта сопровождается подачей сигнала запроса от внешнего источника и возвращением сигнала подтверждения от регистратора по окончании записи во внешнюю память для сброса запроса и разрешения на продолжение передачи. Период формирования запросов от источника – 1 мс. Обслуживание каналов - поочередное. Каналы связи для передачи данных - с параллельными интерфейсами по 8 разрядов.
б) после приема данных для каждого потока данных вычисляется математическое ожидание. Результаты обработки запоминаются во внутреннем ОЗУ и пересылаются по последовательному каналу в ПЭВМ;
в) для заданных с пульта оператора номеров потоков (i=1,2,3,4,5) результаты обработки выводятся на жидкокристаллический дисплей.
Отладить модули программы с помощью симулятора.
Оценить потребляемую мощность устройства.

Оформление курсовой работы

1. Расчетно-пояснительная записка на___30_____ листах формата А4.
2. Перечень графического материала:

а) схема функциональная электрическая б) схема принципиальная электрическая

Дата выдачи задания 4_сентября 2020 г.
Руководитель курсовой работы _____/Хартов В.Я./
Задание получил _____/_____/ «__4__»_сентября_____2020 г.

Дата защиты - 20 декабря 2020 г.

Примечание: Задание оформляется в двух экземплярах; один выдаётся студенту, второй хранится на кафедре.

РЕФЕРАТ

РПЗ 42 стр., 2 таблицы, 26 рисунков, 11 источников, 2 приложения

МИКРОКОНТРОЛЛЕР, ATMEGA8515, ПРИЕМНИК, ПЕРЕДАТЧИК.

Целью данной работы является разработка устройства на базе ATMega8515 для регистрации входных данных, поступающих по двум параллельным шинам, их обработки и записи во внешнюю оперативную память. В качестве источника данных предлагается использовать другой микроконтроллер той же модели. Передача каждого байта сопровождается сигналом запроса от источника и сигналом подтверждения от приемника.

В ходе работы разработаны алгоритмы функционирования системы, функциональная и электрическая принципиальная схемы, написаны коды функционирования, соответствующие разработанным алгоритмам.

Для написания кодов был использован язык СИ AVR, работа осуществлялась в среде AVR Studio 4. Для проверки работоспособности разработанная схема была промоделирована в среде Proteus ISIS 8.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задача передачи, сохранения и обработки данных является актуальной во многих областях. Передача данных между микроконтроллерами — основа для создания связей между микроконтроллерами, для систем. Подключение большого числа внешних устройств необходимо для реализации потенциала микроконтроллера. Использование внешней оперативной памяти еще больше расширяет этот потенциал за счет увеличения объема хранимых и доступных для обработки данных.

Разработка устройства состоит из трех основных частей:

- Аппаратной части — коммутации микроконтроллеров, элемента внешней оперативной памяти и пульта управления;
- Программной части — написание программ для двух микроконтроллеров;
- Тестирование системы посредством симуляции в среде Proteus ISIS 8.

Во время выполнения курсовой работы необходимо построить структурную, функциональную и принципиальную схемы, разработать алгоритмы работы системы, на основе которых программируется микроконтроллер, а также рассчитать потребляемую мощность.

КОНСТРУКТОРСКАЯ ЧАСТЬ

1 Описание системы

Полученное задание можно решить при помощи системы, состоящей из следующих крупных элементов: два микроконтроллера ATMega8515, LCD-дисплей LM016L, драйвер MAX232, программатор AVR ISP500. Конечно же, в систему войдут и другие элементы, такие как кнопки и электрические компоненты, но в данном разделе опишем подробнее сложные блоки схемы. Обобщенная структурная схема приведена на рисунке 1.

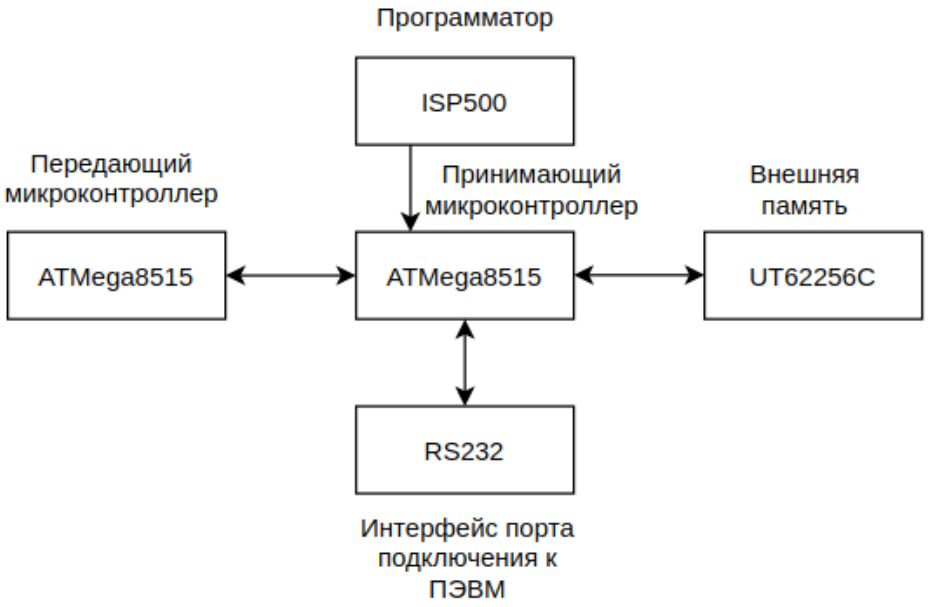

Рисунок 1 — Обобщенная структурная схема

Основной частью системы является микроконтроллер ATMega8515 семейства AVR. С целью повышения параллельности в AVR применяется гарвардская архитектура – для данных и программ используются разные шины и разные блоки памяти. Схема, наглядно демонстрирующая особенности такой архитектуры приведена на рисунке 2.

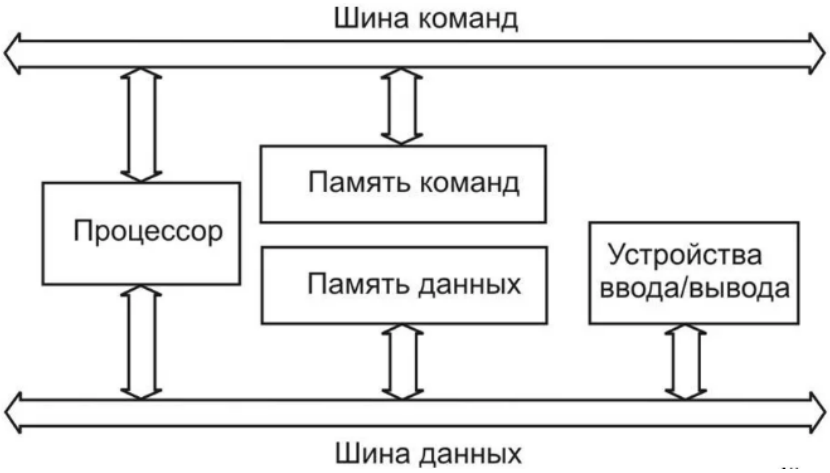

Рисунок 2 – Гарвардская архитектура

В программной памяти (памяти команд) применяется одноуровневая конвейеризация – во время выполнения инструкции следующая заранее выгружается, в итоге инструкции могут выполняться в каждый цикл.

Для решения поставленной задачи необходимо изучить состав микроконтроллера AVR, приведенный на рисунке 3.

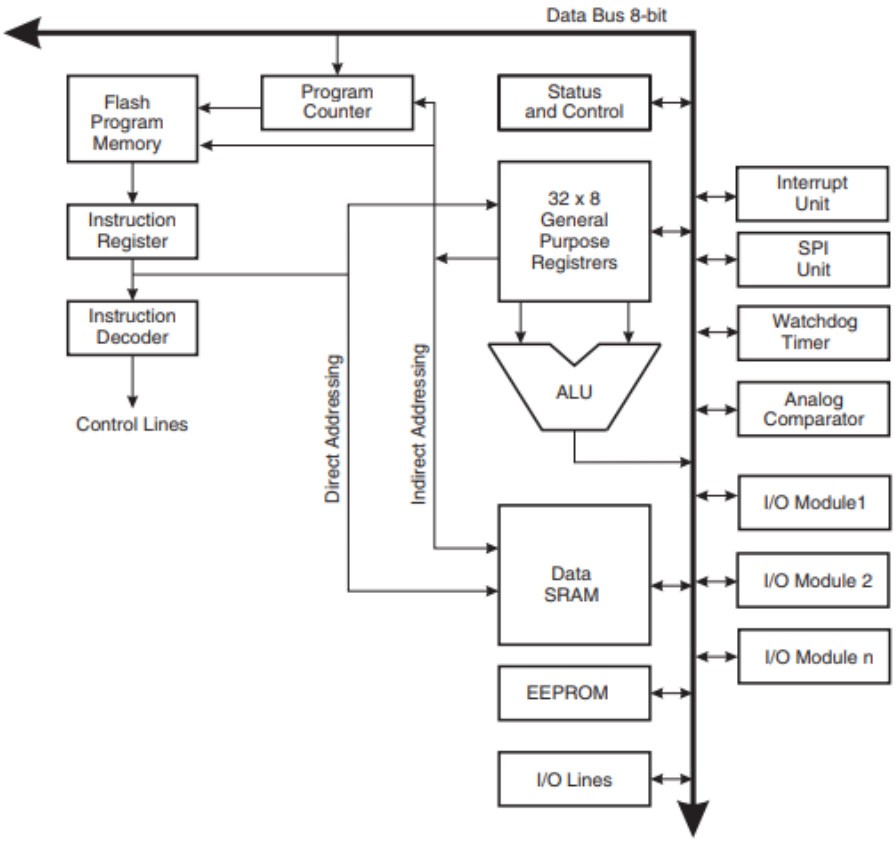

Рисунок 3 – Архитектура микроконтроллеров AVR

Расчет времени может выполнен при помощи счетчика-таймера, разные микроконтроллеры семейства имеют разное количество таймеров, поэтому данный элемент не отображен на рисунке 1.2. Изображенный на схеме сторожевой таймер «Watchdog Timer» является аппаратно-реализованной схемой, призванной осуществлять контроль над зависанием системы.

Интерфейс SPI позволяет осуществить связь с программатором AVR IPC500. Заметим, что при моделировании необходимости подключать программатор не возникнет, так как среда разработки Proteus ISIS 7 подразумевает программирование микроконтроллера напрямую путем загрузки в него HEX-файла программы.

Так как существует необходимость сохранения результатов во внешнюю память, рассмотрим структуру памяти микроконтроллера ATMega8515 в целом. На рисунке 4 приведена карта распределения памяти МК ATMega8515.

Рисунок 4– Карта распределения памяти ATMega8515

Ввиду невозможности использовать внешнюю SRAM способом, предусмотренном разработчиком микроконтроллера (подробнее об этом

далее), она будет подключена в ручном режиме, из-за это диапазон ее адресов будет начинаться не с $0260, а с начала, с $0000.

Еще один интересный момент в устройстве ATMega8515 – наличие подтягивающих резисторов на портах входа/выхода.

Все пины, задействованные в шинах, включены на ввод, без подключения подтягивающего резистора. Пины прерываний INT0 и INT1 включены так же на ввод, без подтягивающих резисторов. В случае, когда микроконтроллеру нужно отправить сигнал, порт переводится в режим «на вывод», на нем на короткое время устанавливается на короткое время сигнал логической единицы, а после переключается обратно в режим «на ввод».

Другой важный элемент системы – драйвер MAX232 для согласования уровня сигнала микроконтроллера ATMega8515 и разъема RS-232. Соответствие уровней напряжения на выходе узла ТТЛ и на входе RS-232 приведено на таблице 1.

Таблица 1 – Согласование напряжений в MAX232

Логический уровень	Напряжение RS-232	Напряжение от ТТЛ
«0»	От +3В до +15В	0В
«1»	От -3В до -15В	5В

Как видно из таблицы 1.3 напряжение на RS-232 не возвращается в нулевое значение, потому что используется способ линейного кодирования NRZ (No Return to Zero – без возвращения к нулю).

Связь через RS-232 осуществляется по стандарту UART, реализация которого в микроконтроллере ATMega8515

Соединение с программатором AVR ISP500 осуществляется по стандарту SPI (Serial Peripheral Interface – последовательный периферийный интерфейс). Схема подключения SPI приведена на рисунке 5.

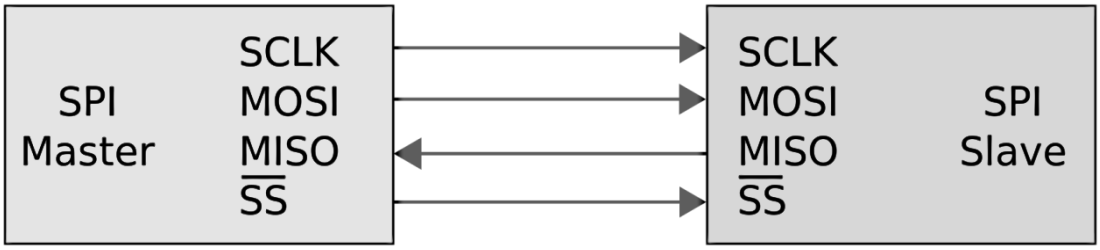

Рисунок 5 – Соединение двух устройств по стандарту SPI

2 Процесс работы и схемы алгоритмов

Привести единую схему алгоритма работы микроконтроллера не представляется возможным ввиду того, что большая часть функций инициируется через прерывания. Ввиду этого будут приведены схемы алгоритмов отдельных функций. После продолжительного времени проектирования удалось добиться разделения всего кода на простые и наглядные функции.

Ввиду того, что необходимо написать разработать программы отдельно для передающего и принимающего микроконтроллеров, рассматриваться они также будут раздельно. На рисунке 6 приведена схема инициализирующего алгоритма передающего микроконтроллера.

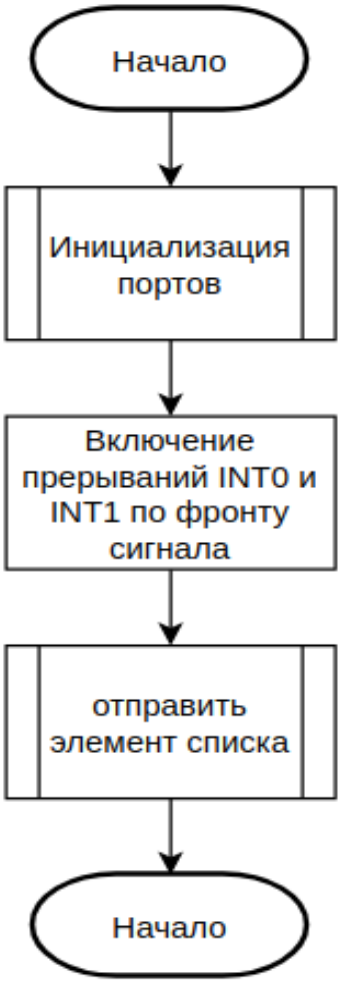

Рисунок 6 — Схема алгоритма инициализации передающего микроконтроллера

После передачи первого элемента первого потока микроконтроллер остается в бесконечном цикле. Дальнейшая работа выполняется при получении сигналов подтверждения от принимающего микроконтроллера, которые вызывают обработчик прерывания.

При обработке внешнего прерывания передающий микроконтроллер сверяет номер потока, который передается в данный момент, и осуществляет передачу следующего элемента. Схема этого алгоритма представлена на рисунке 7.

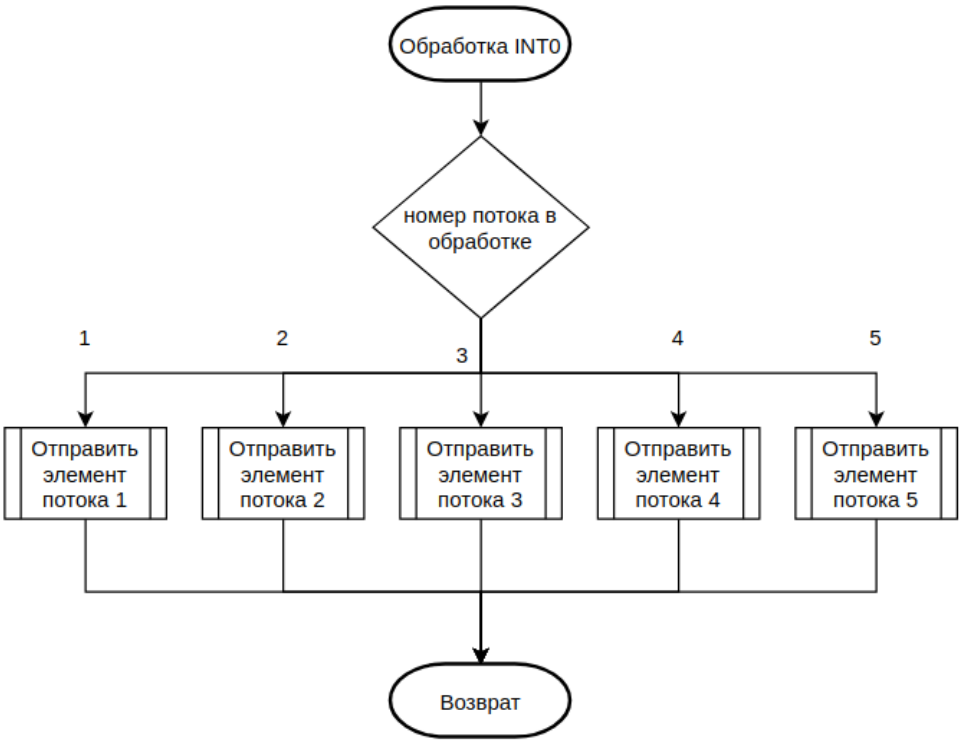

Рисунок 7 — Схема алгоритма обработки прерывания передающим микроконтроллером

При отправке элемента необходимо проверить, имеются ли еще элементы потока, ожидающие отправки. Если такой элемент находится, он выводится на параллельный порт, а микроконтроллер посылает сигнал запроса принимающему микроконтроллеру. Если такого элемента нет, фиксируется завершение передачи потока, что автоматически влечет за собой передачу следующего, так до тех пор, пока не будут переданы все потоки. Схема этого алгоритма представлена на рисунке 8.

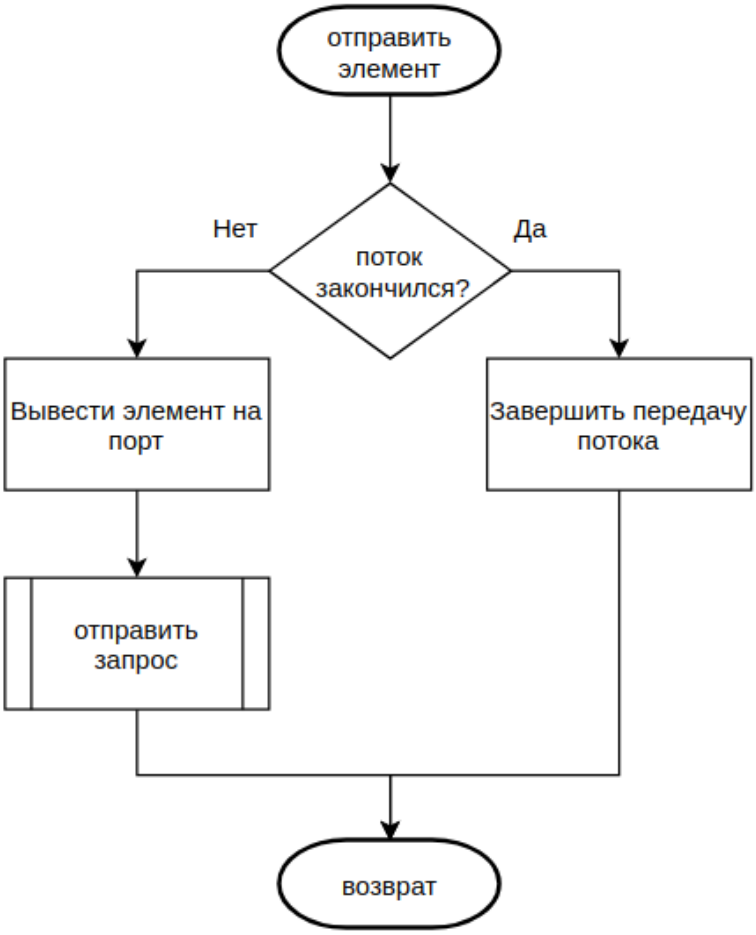

Рисунок 8 — Схема алгоритма передачи элемента передающим микроконтроллером

Рассмотрим процедуру отправки запроса. Эта процедура вызывает обработку прерывания у сопряженного микроконтроллера, и её реализация одинакова для обоих микроконтроллеров. Передающий таким образом отправляет запрос на прием, а принимающий — подтверждение приема.

Алгоритм этой процедуры проще описать текстом, чем схемой, ввиду отсутствия ветвления и необходимости комментариев к каждому действию. Сначала происходит отключение обработки прерывания INT0, чтобы не вызвать обработку прерывания на рассматриваемом микроконтроллере. После этого пин PD2 переключается на вывод. На нем устанавливается значение логической единицы. Это вызывает обработку прерывания у сопряженного микроконтроллера. Затем пин переключается обратно на прием, и обратно включается обработка прерывания INT0.

Рассмотрим принимающий микроконтроллер. Его инициализация не вызывает интереса, ввиду того, что в ней происходит только настройка портов шин на прием и порта внешней памяти на передачу, а так же включение обработки прерываний и пинов RXD и TXD для подключения пульта управления по UART. Больший интерес представляет обработка прерывания, вызванного запросом на прием данных от передающего микроконтроллера. При получении такого сигнала принимающий микроконтроллер считывает с соответствующего порта переданное число, записывает его во внешнюю память и передает сигнал о завершении приема. Схема этого алгоритма представлена на рисунке 9.

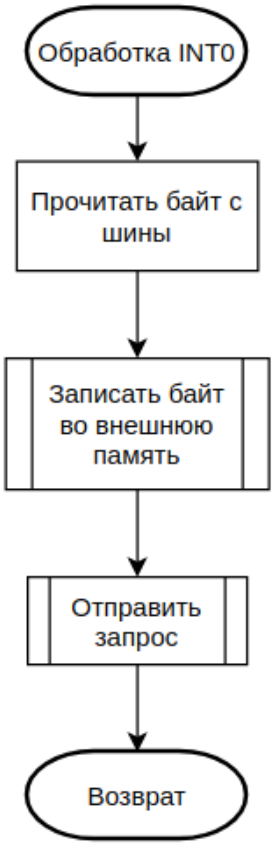

Рисунок 9 — Схема алгоритма обработки прерывания принимающим микроконтроллером

О технических ограничениях при подключении внешней оперативной памяти и итоговой схемотехнической реализации подробнее написано в следующем разделе этого документа. В результате проектирования были получены две функции для удобной и технологичной работой с внешней

памятью. Write для записи и read для чтения. Ввиду того, что байт адреса и байт данных передаются через один и тот же порт, необходимо сначала передать адрес, потом зафиксировать его сигналом ALE с пина PE1, после этого либо вывести на порт вместо адреса данные для записи и послать сигнал записи внешней памяти, либо перевести порт в режим приема и послать сигнал чтения внешней памяти, после чего зафиксировать значение, полученное это порте. Схемы этих алгоритмов представлены на рисунке 10.

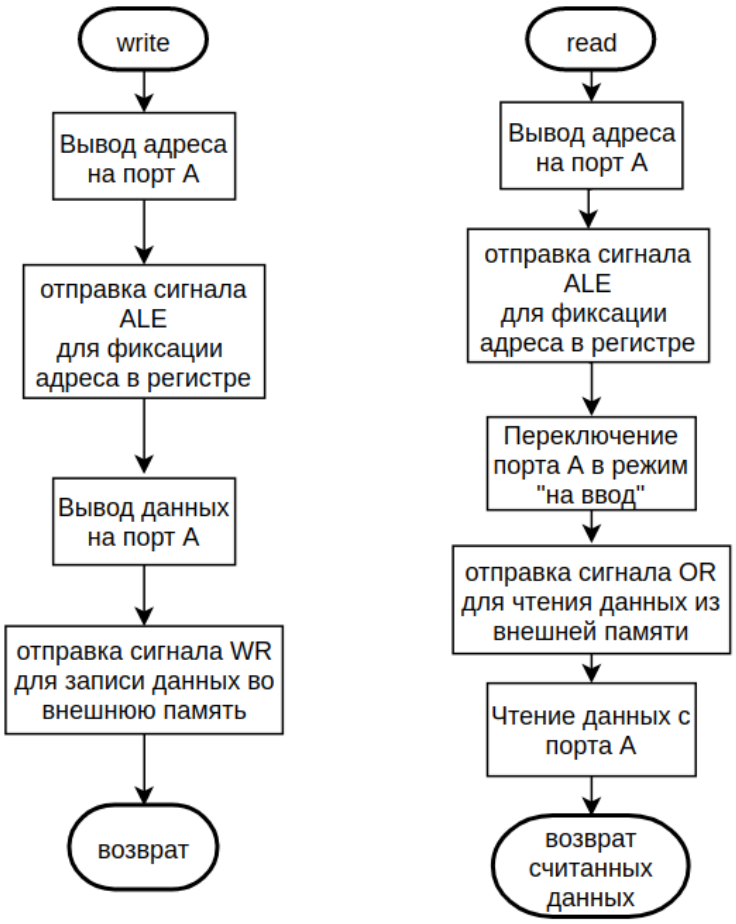

Рисунок 10 — схемы алгоритмов чтения и записи внешней SRAM

3 Описание функциональной схемы

В систему входят: два микроконтроллера ATMega8515, внешняя оперативная память UT62256C, D-триггер защелка 74HC573, разъем для подключения программатора и разъем RS-232, осуществляющий связь с компьютером.

Подключение к программатору происходит по протоколу SPI, для чего задействуется порт B микроконтроллера, конкретнее выходы PB5, PB6, PB7. PB5 используется для передачи сигнала MOSI (Master Out Slave In) и соединяется со входом MOSI программатора; PB6 – MISO (Master In Slave Out) – также соединяется с соответствующем ему входом программатора. PB7 зарезервирован под тактовый сигнал SCK.

В соответствие с устройством МК для связи UART необходимо задействовать порт D, а именно его выходы PD0/RXD и PD1/TXD, которые через MAX232 должны быть связаны с соответствующими входами RXD и TXD разъема RS-232.

Микроконтроллер ATMega8515 поддерживает автоматическую работу с внешней памятью. Для этого необходимо установить в 1 бит SRE регистра управленя MCUCR. Однако, в этом режиме порт A и порт C будут переведены в альтернативный режим работы. Из-за исключительно большого количества периферии, которую необходимо подключить, а так же малого количества используемых байт во внешней памяти, было принято решение от этого варианта отказаться. Вместо этого внешняя память была подключена только к порту A. Это снизило доступный для разыменования объем памяти с 32кБайт до 256 байт, но данное обстоятельство не является критичным в условиях поставленной задачи (необходимо всего 120 байт). Согласно рекомендуемому способу подключения, задействуется триггер-защелка для фиксации адреса. Для обращения к памяти на порт A выводится байт адреса, который затем фиксируется сигналом с мин P1/ALE, после чего байт адреса можно убрать и продолжить чтение, либо запись. Фрагмент подключения внешней оперативной памяти представлен на рисунке 11.

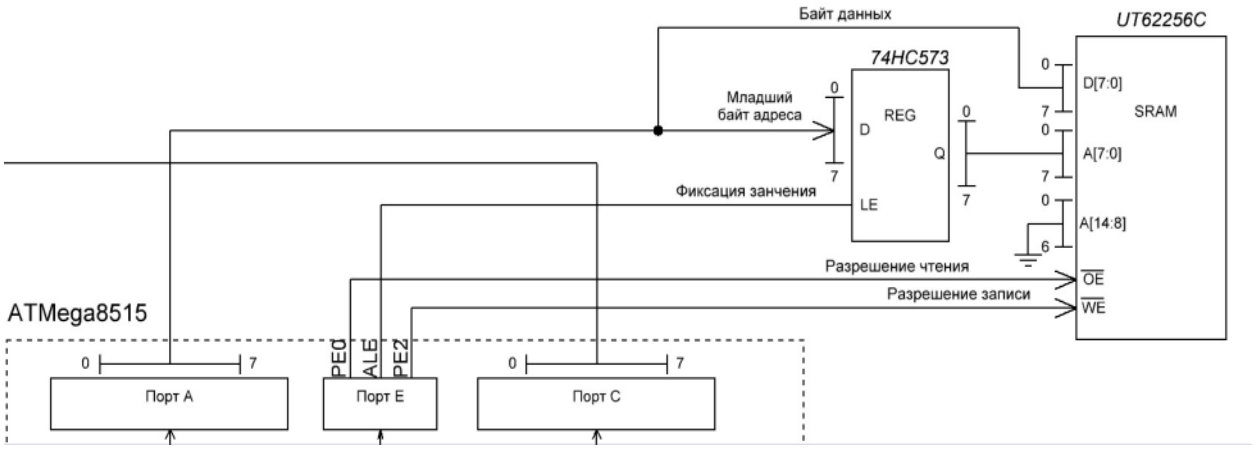

Рисунок 11 — Подключение внешней оперативной памяти

Разъем VDD соединяется с источником питания, VSS, VEE и RW замыкаются на землю. Разъем RS, отвечающий за переключение режима команда/символ дисплея, связывается с PC6. Пины PD2 и PD3, закрепленные за прерываниями INT0 и INT1 соответственно, задействованы у обоих микроконтроллеров для обмена сигналами запроса со стороны передающего и подтверждения со стороны принимающего микроконтроллера.

В изначальном варианте задания присутствовало требование вывода данных на LCD-дисплей. Однако, исследование показало, что у микроконтроллера ATMega8515 не хватает портов для подключения всего перечисленного оборудования. Отказаться от внешней памяти или от одной из параллельных шин передачи данных было невозможно. Ввиду получения управляющих сигналов через пульт управления, подключенный по UART, было принято решение выводить результат туда же. Вследствие этого от LCD-дисплея получилось отказаться.

Кнопка «RESET» подключается к соответствующему входу микроконтроллера.

4 Описание принципиальной схемы

Подключение отдельных устройств системы производилось в соответствии с информацией, представленной в их документации.

Выбранная внешняя оперативная память UT62256C была подключена в соответствии с рекомендациями производителя с использованием питания 5В. Конечный вариант подключения памяти, с использованием триггера-защелки для адреса, приведен на рисунке 12.

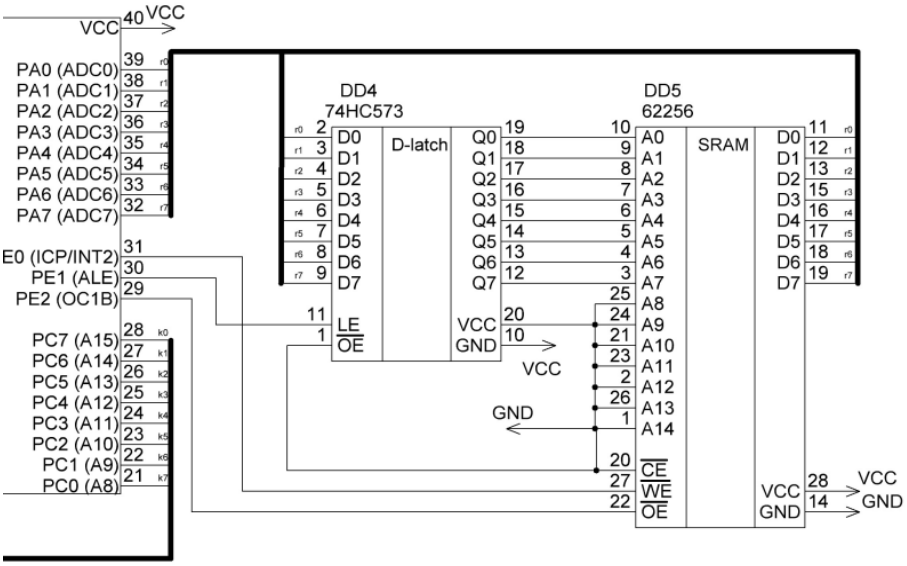

Рисунок 12 – Схема подключения внешней оперативной памяти

Для подключения MAX232 необходимо использовать 4 полярных конденсатора емкостью 1 мкФ каждый, здесь тоже используется питание 5 В, схема подключения приведена на рисунке 13.

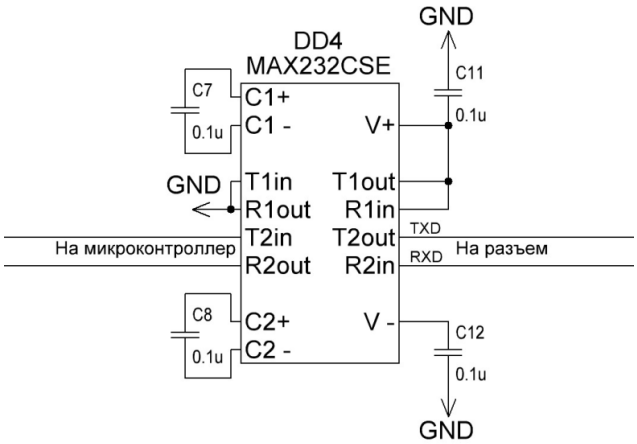

Рисунок 13 – Подключение драйвера MAX232

Подключение программатора осуществляется при помощи SPI, под что задействован порт B микроконтроллера ATMega8515. Подключено питание 5 В, резистор на 10 кОм и конденсатор на 0.1 мкФ позволяют погасить перепады напряжения. В этой же части схемы учтена кнопка сброса «RESET», подключаемая к соответствующему входу микроконтроллера. Итоговая схема подключения приведена на рисунке 14.

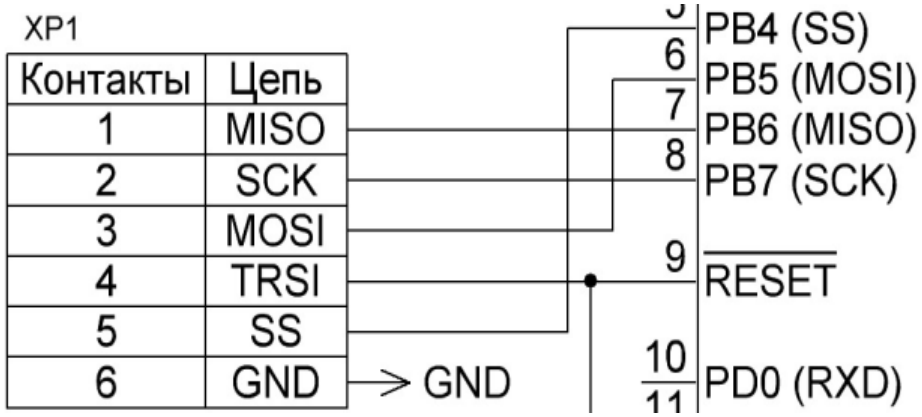

Рисунок 14 – Подключение программатора AVR ISP500

На рисунке 15 показано подключение к разъему питания с конденсаторами, позволяющими минимизировать негативное влияние перепадов напряжения.

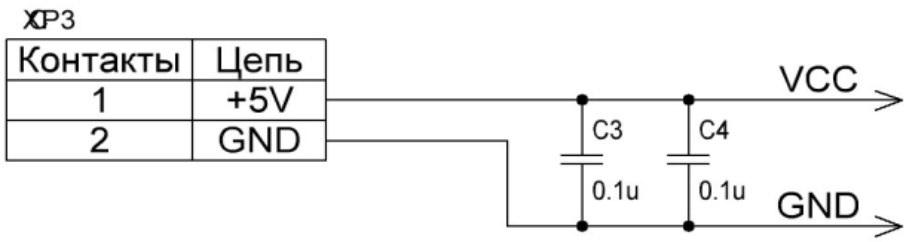

Рисунок 15 – Подключение разъема питания

5 Расчет потребляемой мощности

Общая мощность потребления устройства равна сумме мощностей его элементов. При нашей комплектации формула полной мощности в режиме работы – без учета программатора – выглядит так:

$$P_{общ.раб} = P_{ATMega8515(1)} + P_{ATMega8515(2)} + P_{UT62256C} + P_{74HC573} + P_{MAX232} + P_{резисторов}$$

Отдельные мощности рассчитываются следующим образом:

$$P = I_{max} * U_{num}$$

Для определения тока микроконтроллера возьмем график, изображенный на рисунке 5.1. Микроконтроллер работает при напряжении питания 5В на частоте 8 МГц.

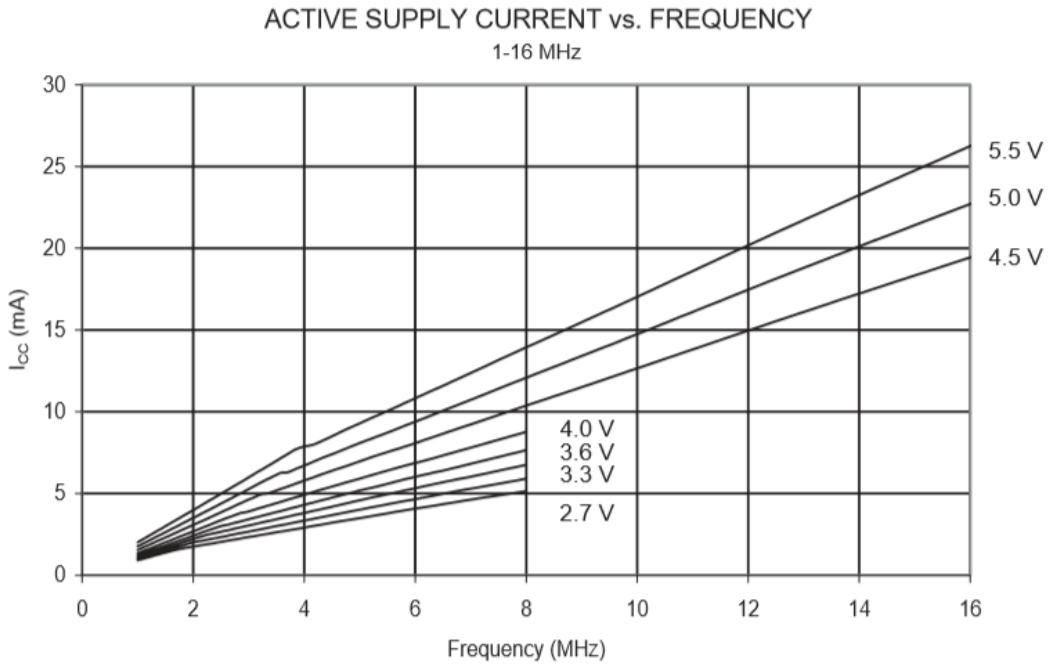

Рисунок 16 – Распределение тока по частотам

На таблице 16 приведены все значения необходимые для расчета полной потребляемой мощности. Ток микроконтроллера умножается на количество используемых входов.

Таблица 2 – Значения расчетных величин

Устройство	I_{max}	$U_{пит}$
ATMega8515(1)	12мА*32	5В
ATMega8515(2)	12мА*19	5В
MAX232	10мА	5В
LM016L	80мА	5В
ISP500	25мА/30мА	5В

Каждый резистор потребляет до 2.5 мВт. В схеме использовано три внешних резистора (SL-20V1-A10K), работают они на напряжении 5В.

$$P_{общ.раб} = 5\,В * (12\,мА * 32 + 12\,мА * 19 + 10\,мА + 80\,мА + 3 * 2{,}5\,мВт) = 3547{,}5\,мВт$$

Мощность, рассчитанная выше, не учитывает программатор, то есть считается, что он отключен.

Рассчитаем также потребляемую мощность при подключенном программаторе:

$$P_{общ} = P_{общ.раб} + P_{ISP\,500} = 3547{,}5\,мВт + 5\,В * 25\,мА = 3672{,}5\,мВт$$

ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

6 Описание системы разработки

При разработке была использована среда AVR Studio 4, позволяющая писать программы для микроконтроллера ATMega8515 на языке ассемблера и Си. Написанные программы можно компилировать в этой же среде, полученный в результате этого файл с расширением «.hex» загружается в модель микроконтроллера среды Proteus ISIS 8.

Схема, собранная в среде Proteus ISIS 7 приведена на рисунке 17.

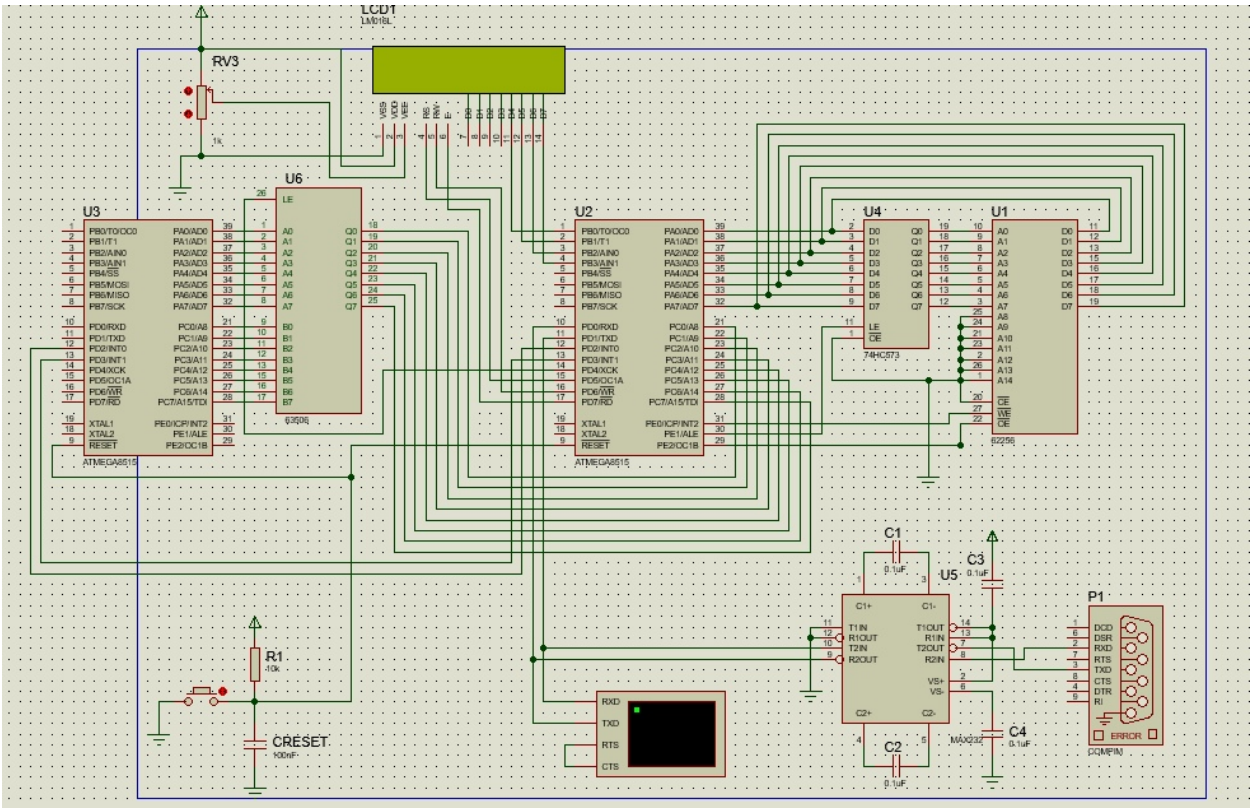

Рисунок 17 – Схема в среде Proteus ISIS 8

Оба микроконтроллера в среде Proteus ISIS настраиваются на частоту 8 МГц, а терминал, принимающий сообщения и передающий по UART – на скорость передачи 9600 бод, все эти настройки приведены на рисунке 18.

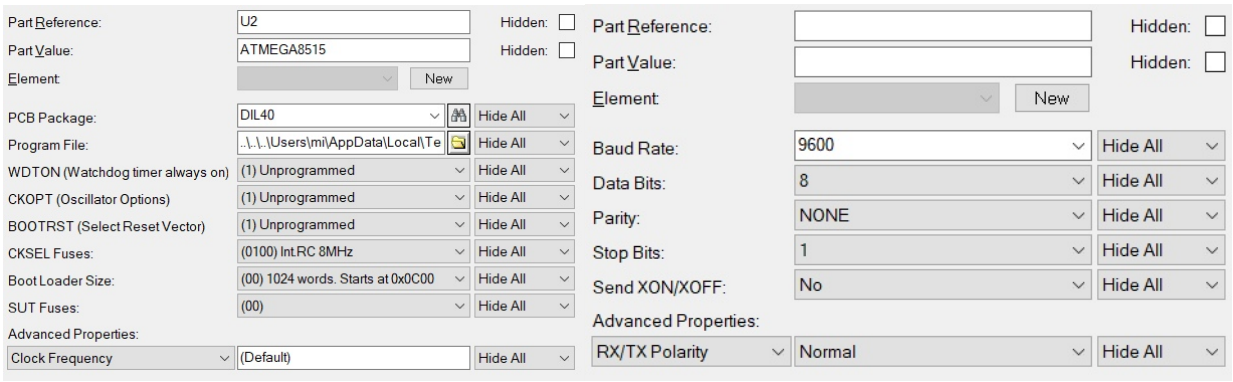

Рисунок 18 – Настройка устройств в Proteus ISIS 8

На рисунке 19 приведена настройка частоты микроконтроллера в среде AVR Studio 4, выбор актуальной частоты позволяет измерить время выполнения отдельной части кода при помощи сторожевого таймера.

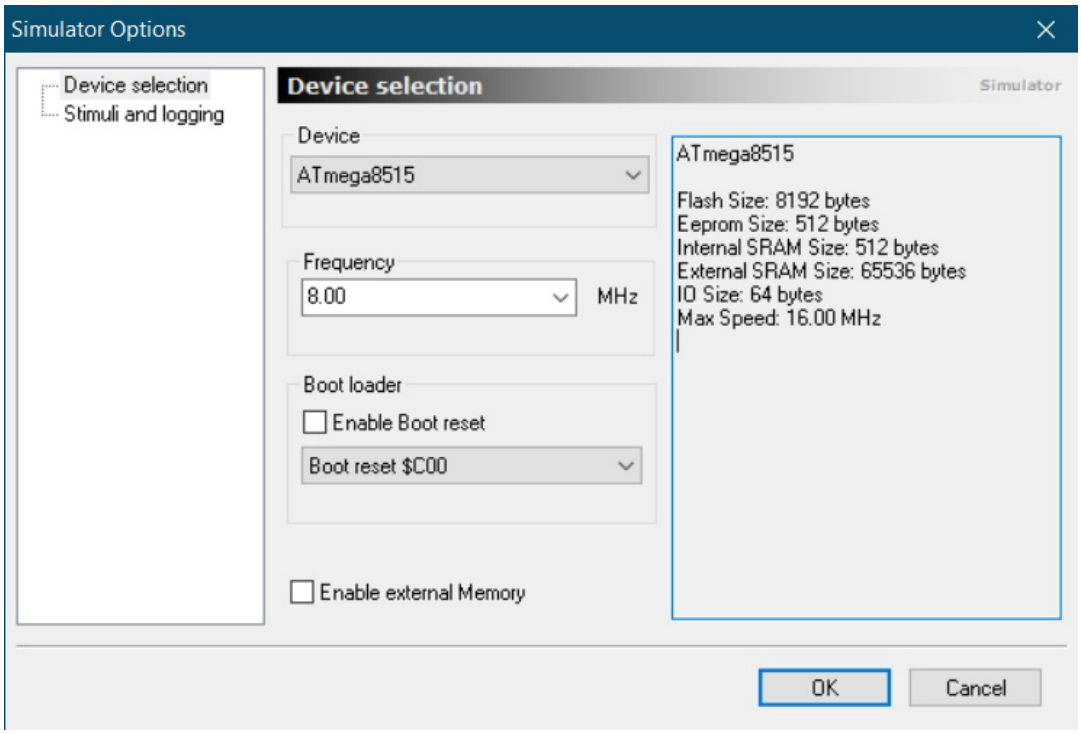

Рисунок 19 – Настройка частоты в среде AVR Studio 4

В ходе обработки данных на принимающем микроконтроллере необходимо посчитать математическое ожидание числового ряда. Результат этого вычисления, в общем случае, является дробным числом. При выводе результата необходимо преобразовать его в строку. Однако, микроконтроллер ATMega8515 не имеет встроенной возможности осуществлять подобное преобразование. Для включения такой возможности необходимо в AVR Studio 4 настроить проект следующим образом:

- в настройках проект подключить дополнительные библиотеки libprintf_flt.a и libm.a. Результат показан на рисунке 20.

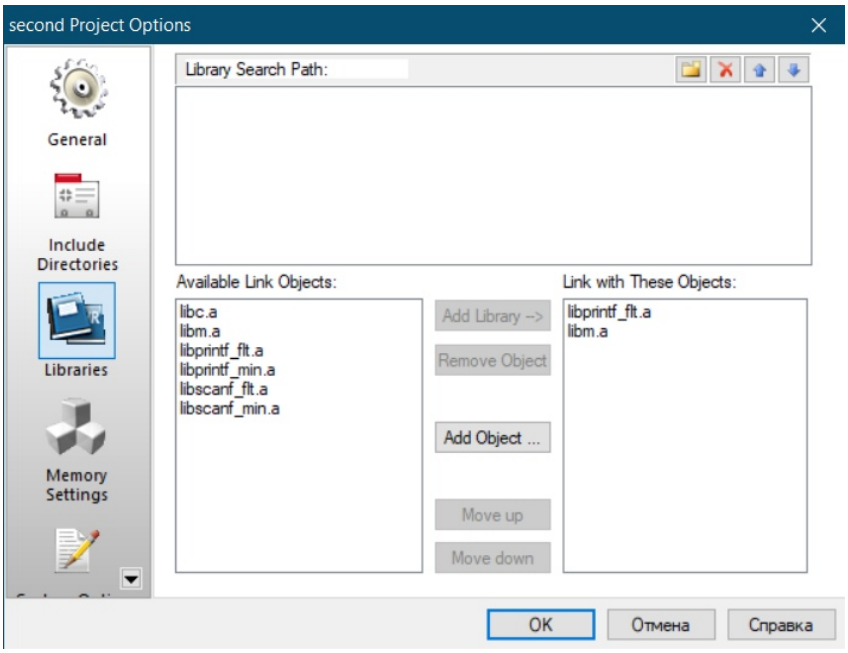

Рисунок *20* — Подключение дополнительных библиотек

- в настройках проекта добавить дополнительные опции линкера: -Wl, -u, vfprintf. Результат показан на рисунке 21.

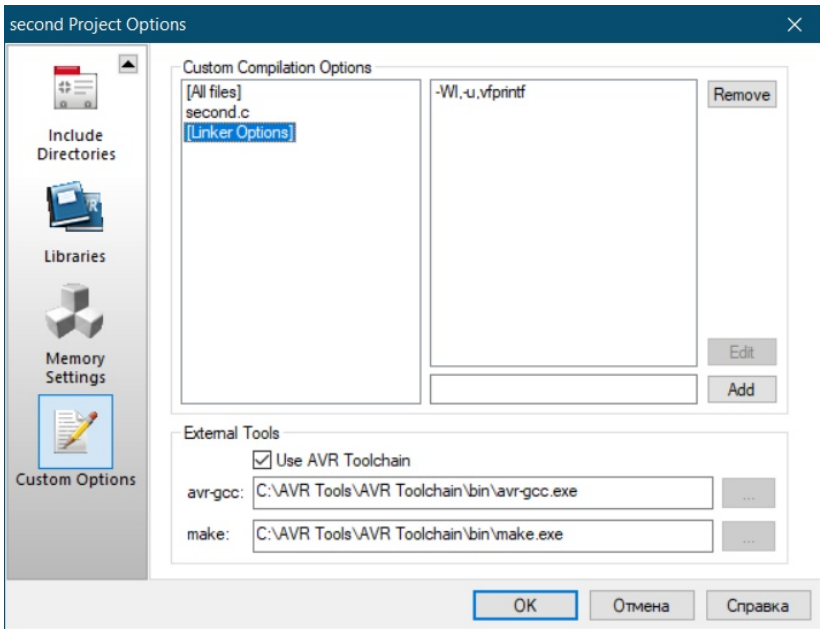

Рисунок *21* — Добавление ключей линкера

7 Разработка программы

Перед началом работы происходит инициализация микроконтроллеров ATMega8515, включающая в себя подготовку разных элементов логики системы к работе.

Происходит настройка портов ввода и вывода. Для передающего микроконтроллера это настройка портов A и C на вывод, настройка порта D на ввод. Для принимающего это настройка портов A и B на вывод и портова C на ввод. Так же в порте D все пины, кроме 2 и 3 настраиваются на вывод. Подтягивающие резисторы не используются нигде.

Для работы с дисплеем и UART кроме таймера применяется процедура задержки из стандартной библиотеки delay.h. Разные задачи – вывод на дисплей и UART – требуют задержку различной величины.

Управление скоростью передачи UART происходит при помощи контроллера скорости передачи, осуществляющего деление частоты, этот процесс подчиняется следующему закону:

$$BAUD = \frac{f_{CLK}}{16 \cdot (UBRR + 1)},$$

где BAUD – скорость передачи (в бодах),

f_{CLK} - тактовая частота микроконтроллера (Гц),

UBRR - содержимое регистра контроллера скорости передачи.

Текущая разработка, несмотря на виртуальную природу, проводилась с учетом всех возможностей микроконтроллера ATMega8515. Его максимальная частота работы составляет 16 МГц. Для целей задания была выбрана частота 8 МГц. Ввиду малого объема передаваемой по UART информации — команды по 1 символу — скорость работы модуля была выбрана 9600 бод. Это повысит надежность передачи данных.

Предполагается работа с дисплеем в четырехбитном режиме, с помощью сторонней библиотеки. Изначальная настройка происходит в отдельном файле lcd_lib.h. Инициализация дисплея производится вызовом

функции lcd_init(), после чего дисплей включается, и на нем начинает мигать курсоор. Таким способом дисплей сообщает о своем готовности к работе.

Оценим время работы программы. Поставив точку остановки на момент получения первого байта, запишем данные счетчика циклов. Это число 2662. Поставив точку остановки на момент окончания принятия последнего байта последней последовательности, получим число 25926.

25926 - 2662 = 23264

Получим число циклов. Разделим это число на значение тактовой частоты работы микроконтроллера и получим время работы программы в секундах.

8000000 / 23264 = 0.002908 с ~ 2.9 мс

Получим результат - 2.9 мс на передачу всех 120 чисел.

2.9 / 120 = 0.024 мс

Разделив время передачи всех чисел на их количество, получим примерное время передачи одного числа — 0.024 мс или 24 мкс.

8 Тестирование

В начале работы, как показано на рисунке 22, происходит инициализация дисплея и UART. После успешной инициализации на дисплее появляется курсор.

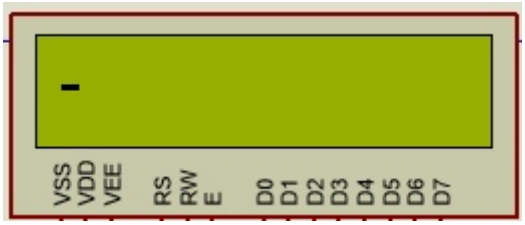

Рисунок 22 – Инициализированный дисплей

После этого автоматически начинается передача данных, сначала три пакета по 20 чисел, по одной шине, потом два пакета по 30 чисел, по другой шине. Все они последовательно записываются во внешнюю оперативную память начиная с адреса 0x00. Состояние памяти до передачи показано на рисунке 23.

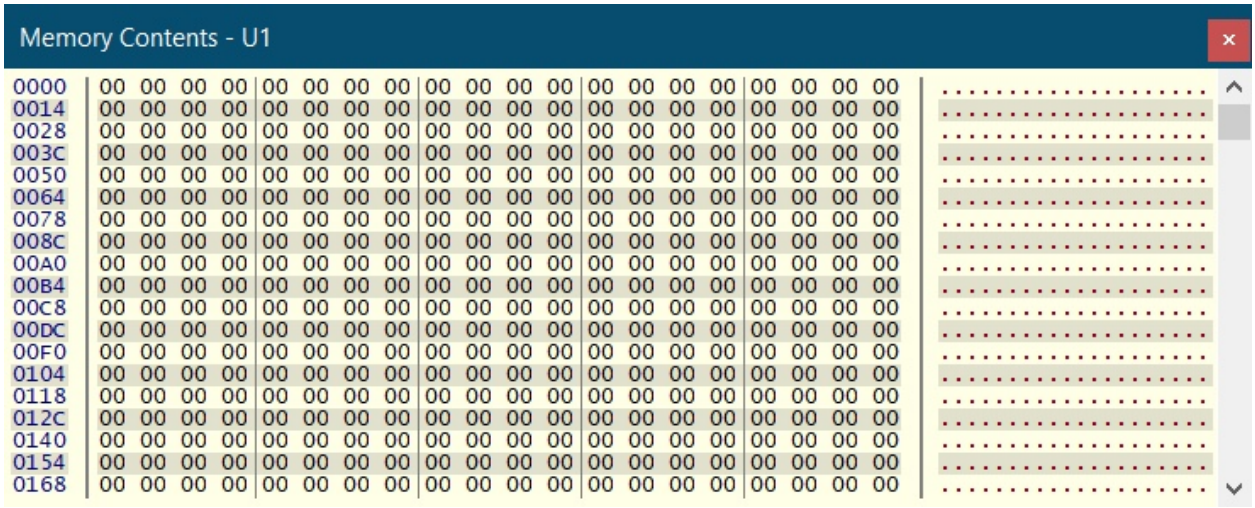

Рисунок 23 — Внешняя память до начала передачи

После завершение передачи во внешней памяти можно увидеть все переданные числа (рисунок 24).

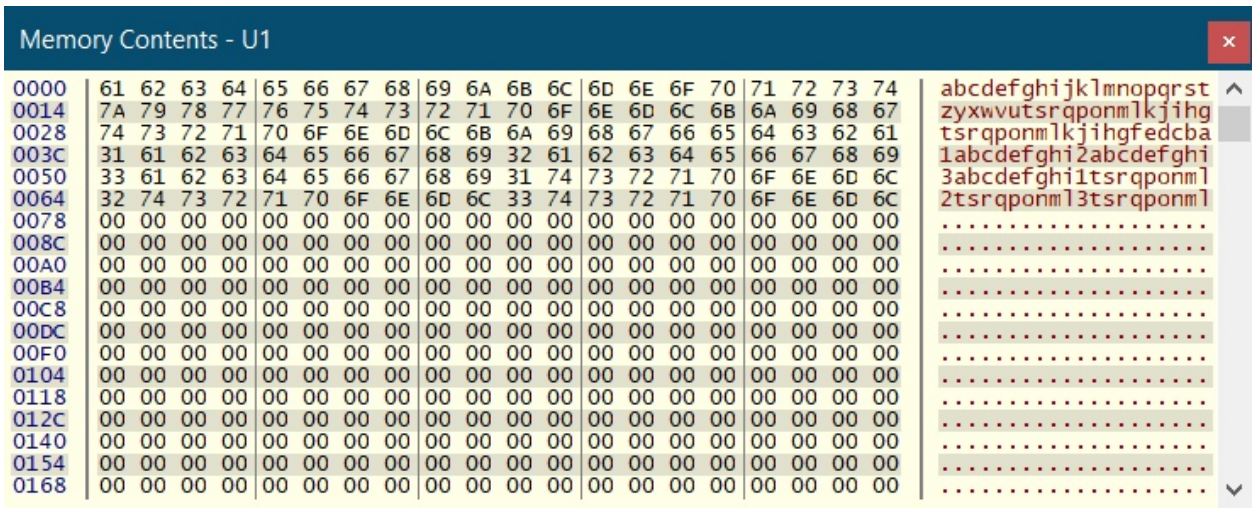

Рисунок 24 — Данные в памяти после завершения передачи

Таким образом можно убедиться, что данные действительно записываются во внешнюю оперативную память.

Представим другие данные на для передачи с целью упрощения вычисления результата обработки. Пять новых массивов чисел представлены на рисунке 25.

```
list_1[20] = {1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,4};
list_2[20] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};
list_3[20] = {255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255};
list_4[30] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
list_5[30] = {10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39};
```

Рисунок 25 — данные для передачи

Таким образом можно убедиться, что данные действительно записываются во внешнюю оперативную память.

Выведем поочередно результат обработки (математическое ожидание потока данных, оно же — среднее значение) на дисплей. Порядок соответствует порядку представления данных. Результат представлен на рисунке 26.

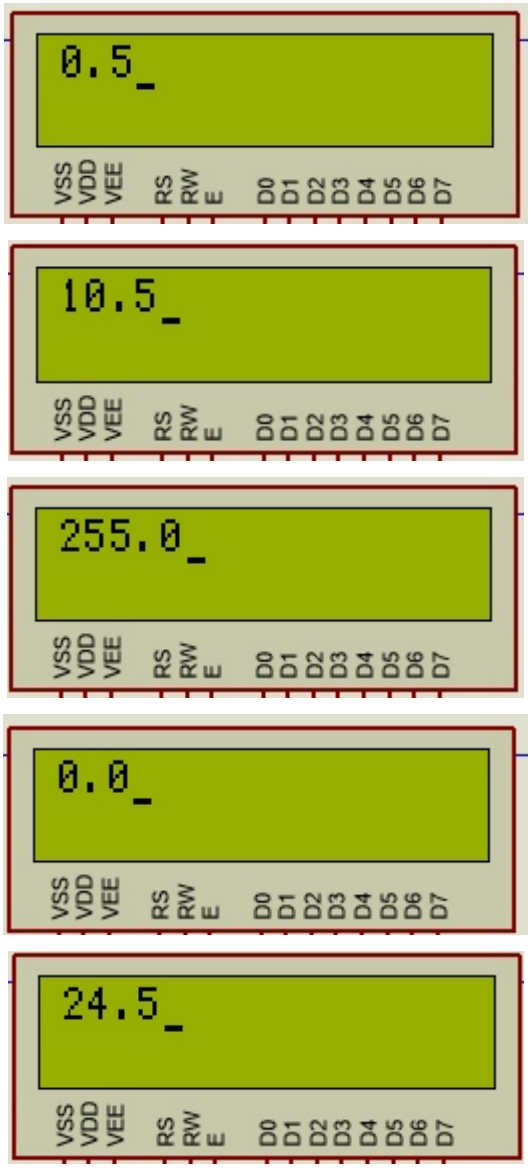

Рисунок 26 — Средние значения потоков данных

Кратко проверим вычисления. 1+2+3+4=10 и 20/10=0,5. Первый массив обработан верно. Сумма чисел в ряду от 1 до 20 по методу Гаусса равна (20+1)*10=210. 210/20=10,5. Второй массив обработан верно. Математическое ожидание из ряда повторяющихся чисел равно самому числу, третий массив обработан верно. Аналогично для 4 ряда. Он обработан верно и ряд из одних нулей не вызвал ошибки. Сумма чисел в ряду от 10 до 39 по методу Гаусса равна (10+39)*15=735. 735/30=24,5. Пятый ряд обработан верно.

Все переданные данные были обработаны корректно. Математическое ожидание вычислено правильно.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была спроектирована система, реализующая передачу данных между микроконтроллерами по двум восьмибитным шинам, запись данных во внешнюю оперативную память, вывод информации на дисплей, а так же подключение пульта управления по UART.

Программный код системы был написан на языке Си процессоров семейства AVR, для чего была использована среда разработки AVR Studio 4. Было произведено моделирование системы в среде Proteus ISIS 8.8, и выполнена ее симуляция.

Был подготовлен набор документации на объект разработки, состоящий из расчетно-пояснительной записки, функциональной схемы, принципиальной схемы, листинга программного кода, спецификации радиоэлементов, использованных в системе.

СПИСОК ИСТОЧНИКОВ

1. Документация микроконтроллера ATmega8515 - http://ww1.microchip.com/downloads/en/devicedoc/doc2512.pdf.

2. Документация ЖК-дисплея LM016L – http://pdf1.alldatasheet.com/datasheet-pdf/view/146552/HITACHI/LM016L.html

3. Документация драйвера MAX232 - http://www.ti.com/lit/ds/symlink/max232.pdf.

4. Документация программатора AVR-ISP500 – https://www.olimex.com/Products/AVR/Programmers/AVR-ISP500/resources/AVR-ISP500.pdf.

5. Документация регистра-защелки 74HC573 https://static.chipdip.ru/ lib/429/DOC005429812.pdf.

6. Документация внешней SRAM UT62256C https://static.chipdip.ru/lib/506/DOC001506992.pdf

7. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих. 2-е изд. М.: Из-во МГТУ им. Баумана, 2012.

8. Хартов В.Я. Микропроцессорные системы: учеб. пособие для студ. Учреждений высш. образования / В.Я.Хартов. – 2-е изд., испр. и доп. – М.: Издательский центр «Академия», 2014.

9. ГОСТ 2.743-91 Обозначения условные в графических схемах. Элементы цифровой техники.

10. ГОСТ 2.701-84 Правила выполнения схем.

11. ГОСТ 2.702-75 Правила выполнения электрических схем.

ПРИЛОЖЕНИЕ А. ТЕКСТ ИСХОДНОЙ ПРОГРАММЫ.

Исходный код принимающего микроконтроллера:

```
#include <avr/io.h>                     // Standard AVR header
#include <util/delay.h>         // Delay loop functions
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdio.h>
#include <string.h>

#define F_CPU 8000000L
#define lcd_on                  PORTD |= 0x80
#define lcd_off         PORTD &= ~0x80
#define lcd_data        PORTD |= 0x20
#define lcd_command PORTD &= ~0x20

char command = '0';
uint8_t SRAM_pointer = 0;
uint8_t sizes[10];
uint8_t cur_size = 0;

//delay function for multiple purpose
void delay_ms(int miliSec) //for 16 Mhz crystal
{
   int i,j;
   for(i=0;i<miliSec;i++)
        {
                for(j=0;j<1550;j++)
                {
                        asm("nop");
                        asm("nop");
                }
        }
}
//Send a command to the lcd
void lcd_send_command (unsigned char Command)
{
        lcd_command; // Set LCD in command mode
        PORTB = Command; // Load data to port
        lcd_on; // Write data to LCD
        asm("nop");
        asm("nop");
        lcd_off; // Disable LCD
        delay_ms(1); // wait for 1ms
}
//Send data to the lcd
void lcd_send_data (unsigned char Data)
{
        lcd_data; // Set LCD in data mode
        PORTB = Data; // Load data to port
```

```
        lcd_on; // Write data to LCD
        asm("nop");
        asm("nop");
        lcd_off; // Disable LCD
        delay_ms(1); // wait for 1ms
}
//Clear Display
void lcd_clear(void)
{
        lcd_send_command(0x01);//Clear
}
//Routine for lcd initialization
void lcd_init(void)
{
        // delay_ms(100); // wait for 100ms
        lcd_send_command (0x38); // 8 data lines
        lcd_send_command (0x06); // cursor setting
        lcd_send_command (0x0E); // display ON
        lcd_send_command (0x01); // clear LCD memory
        //delay_ms (10); // 10ms delay after clearing LCD
}
//set lcd  cursor position
void lcd_cursor (char row, char column)
{
 switch (row)
 {
   case 1: lcd_send_command (0x80 + column - 1); break;
   case 2: lcd_send_command (0xc0 + column - 1); break;
   default: break;
 }
}
// send string to lcd
void lcd_displaystring_f (char row, char column , char string[])
{
 lcd_cursor (row, column);
 while (*string)
   lcd_send_data(*string++);
}
// set chosen pins to 1
void on(volatile uint8_t *byte, uint8_t set_to)
{
        *byte |= set_to;
}
// set chosen pins to 0
void off(volatile uint8_t *byte, uint8_t set_to)
{
        *byte &= ~set_to;
}

void init_UART()
{
        UBRRH=0;
```

```
	UBRRL=51;
	UCSRA=0b00000000;
	UCSRB=0b10011000;
	UCSRC=0b10000110;
}

//	send one char via UART
void send_Uart(unsigned char c)
{
	while(!(UCSRA&(1<<UDRE)))
	{}
	UDR = c;
}
// Send string via UART
void send_Uart_str(unsigned char *s)
{
	while (*s != 0)
		send_Uart(*s++);
}
// read char from UART
unsigned char getch_Uart(void)
{
	while(!(UCSRA&(1<<RXC)))
	{}
	return UDR;
}

// UART interrupt
ISR(USART_RX_vect)
{
	command = getch_Uart();
}

// send INT0 interrupt
void send_0_int(void)
{
	GICR &= ~0x40;
	DDRD |= 0x04;
	on(&PORTD, 0x04);
	off(&PORTD, 0x04);
	DDRD &= ~0x04;
	GICR |= 0x40;
}
void write_to_SRAM(uint8_t address, uint8_t data)
{
	PORTA = address;
	on(&PORTE, 0x02);
	off(&PORTE, 0x02);
	PORTA = data;
	off(&PORTE, 0x01);
	on(&PORTE, 0x01);
}
```

```
uint8_t read_from_SRAM(uint8_t address)
{
	PORTA = address;
	on(&PORTE, 0x02);
	off(&PORTE, 0x02);
	DDRA = 0x00;
	off(&PORTE, 0x04);
	uint8_t res = PINA;
	on(&PORTE, 0x04);
	return res;
}
// compute averege for array in SRAM
float get_avg(uint8_t from, uint8_t count)
{
	int sum = 0;
	for (int i = from; i < from+count; i++)
		sum += read_from_SRAM(i);

	return (float)sum/count;
}
// find acerege for chosen flow of data
void perform_job(uint8_t list_num)
{
	uint8_t from = 0;
	for (uint8_t i = 0; i < list_num; i++)
		from += sizes[i];

	float avg = get_avg(from, sizes[list_num]);

	char avg_str[20];
	sprintf(avg_str, "%.1f", avg);
	//send_Uart_str(avg_str);
	lcd_displaystring_f(1,1,avg_str);
}

void ports_init(void)
{
	DDRE = 0xff;
	PORTE = 0x05;
   DDRB = 0xff;
	DDRC = 0x00;
   DDRD = 0xf2;
	DDRA = 0xff;
}
//common part of interrupt handling
void isr_int01_common(uint8_t max)
{
	uint8_t received_byte = PINC;

//	send_Uart(received_byte);
```

```
		write_to_SRAM(SRAM_pointer, received_byte);
		SRAM_pointer++;

		sizes[cur_size] += 1;
		if (sizes[cur_size] == max)
			cur_size++;

		send_0_int();
}
// INT1 interrupt handler
ISR(INT1_vect)
{

		off(&PORTD, 0x10);
		isr_int01_common(30);
}
// INT0 interrupt handler
ISR(INT0_vect)
{

		on(&PORTD, 0x10);
		isr_int01_common(20);
}
int main(void)
{
		SRAM_pointer = 0;
		ports_init();

		GICR|= 0xC0;
		GIFR = 0xC0;

		off(&GIFR, 0xC0);
   MCUCR= 0x0F;

		init_UART();
		asm("sei");



		lcd_init();

		while(1)
		{
			if(command > '0' && command < '6')
			{
				perform_job((uint8_t)command-48-1);
				//send_Uart(13);
				command = '0';
			}
			asm("nop");
		}
}
```

Исходный код передающего микроконтроллера:

```
#include <avr/io.h>                     // Standard AVR header
#include <util/delay.h>         // Delay loop functions
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdio.h>
#include <string.h>
//#include <cstdint>


#define F_CPU 8000000L

unsigned char command = '0';

int in_progress = 0;

// input flows of byte here
unsigned char list_1[20] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t'};
unsigned char list_2[20] = {'z','y','x','w','v','u','t','s','r','q','p','o','n','m','l','k','j','i','h','g'};
unsigned char list_3[20] = {'t','s','r','q','p','o','n','m','l','k','j','i','h','g','f','e','d','c','b','a'};
unsigned char list_4[30] =
{'1','a','b','c','d','e','f','g','h','i','2','a','b','c','d','e','f','g','h','i','3','a','b','c','d','e','f','g','h','i'};
unsigned char list_5[30] =
{'1','t','s','r','q','p','o','n','m','l','2','t','s','r','q','p','o','n','m','l','3','t','s','r','q','p','o','n','m','l'};

int cur_1 = 0, cur_2 = 0, cur_3 = 0, cur_4 = 0, cur_5 = 0;
// set chosen pins to 1
void on(volatile uint8_t *bit, uint8_t set_to)
{
        *bit |= set_to;
}
// set chosen pins to 0
void off(volatile uint8_t *bit, uint8_t set_to)
{
        *bit &= ~set_to;
}
// send INT0 interrupt
void send_0_int(void)
{
        off(&GICR, 0x40);
        on(&DDRD, 0x04);
        on(&PORTD, 0x04);

        off(&PORTD, 0x04);
        off(&DDRD, 0x04);
        on(&GICR, 0x40);
}
// send INT1 interrupt
void send_1_int(void)
{
        off(&GICR, 0x80);
```

```
	on(&DDRD, 0x08);
	on(&PORTD, 0x08);

	off(&PORTD, 0x08);
	off(&DDRD, 0x08);
	on(&GICR, 0x80);
}

void ports_init(void)
{
	DDRA = 0xff;
   DDRC = 0xff;
   DDRD = 0x02;
}
//send element from array [20]
void push_list_20(unsigned char *list, int *num)
{
	if (*num < 20)
	{
		PORTC = list[*num];
		*num += 1;
		send_0_int();
	}else{
		command = (uint8_t)in_progress + 48 + 1; // int to char
		in_progress = 0;
		*num = 0;
	}
}
//send element from array [30]
void push_list_30(unsigned char *list, int *num)
{
	if (*num < 30)
	{
		PORTA = list[*num];
		*num += 1;
		send_1_int();
	}else{
		command = (uint8_t)in_progress + 48 + 1; // int to char
		in_progress = 0;
		*num = 0;

	}
}
// handle INT0 interrup from Sender
ISR(INT0_vect)
{
	switch (in_progress)
	{
		case 1:
			push_list_20(&list_1, &cur_1);
			break;
		case 2:
```

```
                push_list_20(&list_2, &cur_2);
                break;
            case 3:
                push_list_20(&list_3, &cur_3);
                break;
            case 4:
                push_list_30(&list_4, &cur_4);
                break;
            case 5:
                push_list_30(&list_5, &cur_5);
                break;
        }
}



int main(void)
{
        ports_init();

        GICR |= 0xC0;
        MCUCR = 0x0F;

        asm("sei");


        _delay_ms(1000);

        command = '1';

        while(1)
        {

            switch (command)
            {
            case '1':
                command = '0';
                in_progress = 1;
                push_list_20(&list_1, &cur_1);
                break;
            case '2':
                in_progress = 2;
                push_list_20(&list_2, &cur_2);
                command = '0';
                break;
            case '3':
                in_progress = 3;
                push_list_20(&list_3, &cur_3);
                command = '0';
                break;
            case '4':
                in_progress = 4;
```

```
                push_list_30(&list_4, &cur_4);
                command = '0';
                break;
            case '5':
                in_progress = 5;
                push_list_30(&list_5, &cur_5);
                command = '0';
                break;
            }

            asm("nop");
        }
}
```

ПРИЛОЖЕНИЕ Б. СПЕЦИФИКАЦИЯ РАДИОЭЛЕМЕНТОВ СХЕМЫ.

Поз.	Обозначение	Наименование	Кол.	Примечание
		Микросхемы		
1	DD1, DD5	ATMega8515	2	
2	DD2	UT64506B	1	
3	DD3	LM032L	1	
4	DD4	MAX232CSE	1	
5	DD6	74HC573	1	
6	DD7	UT62256C	1	
		Конденсаторы		
7	C1, C3	NP0	2	
8	C2, C4-C14	C7R	12	
		Резисторы		
9	R1	CF-50, 10 кОм	1	
10	R2, R3, F4	SL-20V1-A10K, 1 кОм	3	
		Кварцевый резонатор		
11	Q1, Q2	HC-49U	2	
		Разъемы		
12	XP1, XP2	IDC-06F	2	
13	XP3	KF350	1	
14	XP4	IDC-09F	1	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм	Лист	№ докум.	Подпись	Дата
Разраб.		Аксенов Р.М.		
Пров.		Хартов В.Я.		
р.				
Утв.				

Устройство регистрации и обработки потоков входных данных

Спецификация радиоэлементов схемы

Лит.	Лист	Листов
	1	1

МГТУ им. Баумана, каф. Компьютерные системы и сети

Копировал: **Формат А4**