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

ПРОЕКТИРОВАНИЕ ТРЕНАЖЕРА

по курсу «Микропроцессорные системы»

Студент гр. ИУ6-76 _____ _____Т.А. Тищенко___

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____В. Я Хартов___

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

РЕФЕРАТ

РПЗ 30 стр., 5 таблиц, 24 рисунка, 9 источников, 2 приложения

МИКРОКОНТРОЛЛЕР, ATMEGA8515, ДИСПЛЕЙ, ТРЕНАЖЕР.

Настоящая работа посвящена разработке тренажера проверки психофизиологических характеристик человека в виде МК-системы на базе микроконтроллера ATMega8515. Пользователю предлагается выбрать временной диапазон, в котором далее генерируется псевдослучайное число. Пользователь нажимает кнопку и самостоятельно засекает такое число секунд, после чего останавливает отсчет, после чего система сообщает, насколько он отклонился от требуемой величины.

Требования к разработанному тренажеру:

- Возможность смены диапазона генерации времени, которое пользователь должен засечь, возможные диапазоны времени в секундах: от 5 до 20, от 20 до 40, от 30 до 60, от 5 до 60;

- Генерация псевдослучайного числа в заданном диапазоне;

- Сравнение сгенерированного эталонного времени и времени, которое засек пользователь, вывод разницы;

- Накопление результатов пяти испытаний;

- Вывод результатов испытаний по UART.

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

Для написания кодов был использован язык ассемблера AVR, работа осуществлялась в среде AVR Studio 4. Для проверки работоспособности разработанная схема была промоделирована в среде Proteus ISIS 7.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача тестирования психофизиологических характеристик человека является актуальной в областях, где успешность работы или даже жизнь рабочего зависят от этих самых характеристик. Один из возможных примеров применения такой системы тестирования – подготовка дайверов, запасы кислорода которых рассчитаны на определенное количество времени, а поэтому ощущение прошедшего с момента погружения времени становится важной способностью в данной специальности.

Для выполнения поставленной задачи было решено использовать минимальный набор из двух кнопок, подключенных к микроконтроллеру. Одна кнопка будет отвечать за смену диапазона генерации, другая – за переход к следующему шагу испытания. Первый шаг – выбор диапазона, второй шаг – генерация псевдослучайного числа из диапазона, третий шаг – окончание генерации и вывод сгенерированного числа, четвертый шаг – непосредственно тестирование, в течение которого пользователь мысленно засекает выведенное число секунд, пятый шаг – вывод результатов.

Разработка была разбита на следующие части:

- Разработка программной части;
- Разработка аппаратной части;
- Тестирование системы посредством симуляции в среде Proteus ISIS 7.

КОНСТРУКТОРСКАЯ ЧАСТЬ

1 Описание системы

Полученное задание можно решить при помощи системы, состоящей из следующих крупных элементов: микроконтроллер 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 1.1.

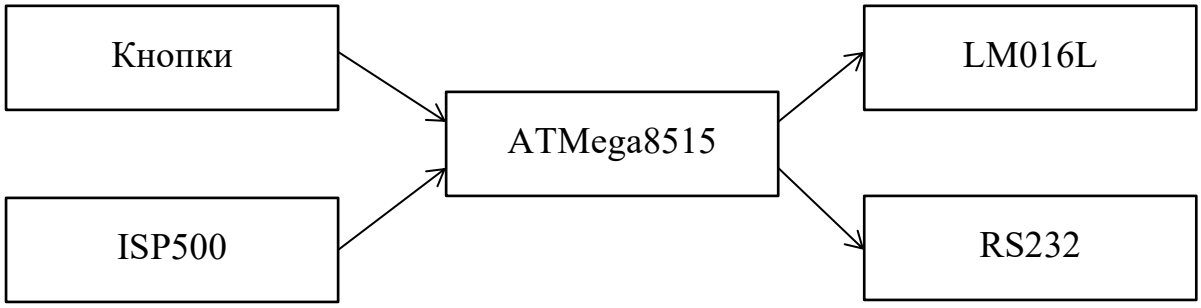

Рисунок 1.1 – Структурная схема системы

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 1.1.

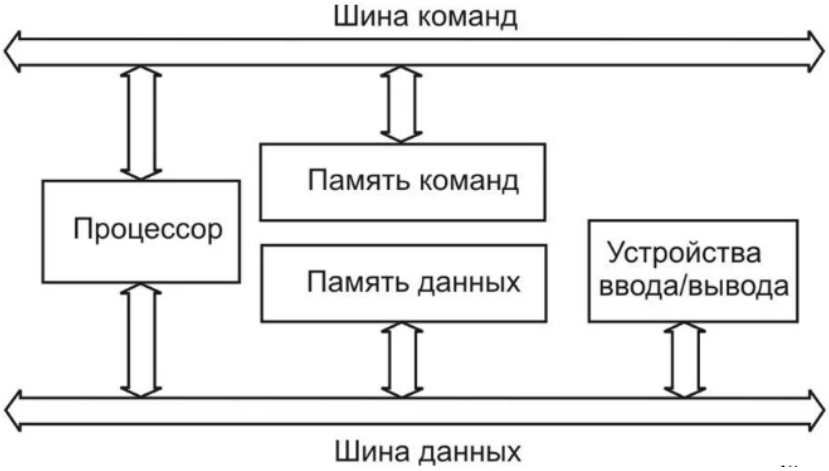

Рисунок 1.2 – Гарвардская архитектура

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 процессора AVR, приведенный на рисунке 1.2.

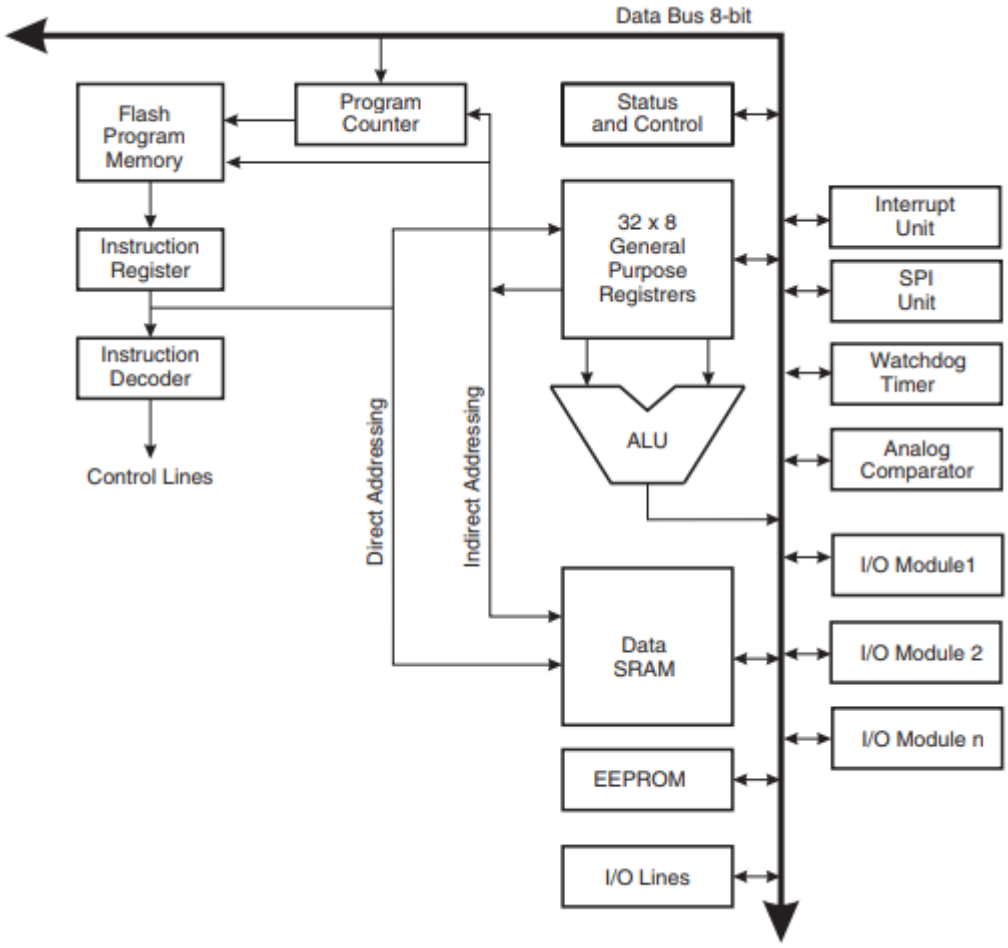

Рисунок 1.3 – Архитектура процессора AVR

Расчет времени может выполнен при помощи счетчика-таймера, разные процессоры 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 в память, рассмотрим структуру памяти микроконтроллера ATMega8515. На рисунке 1.3 приведена карта распределения памяти МК ATMega8515.

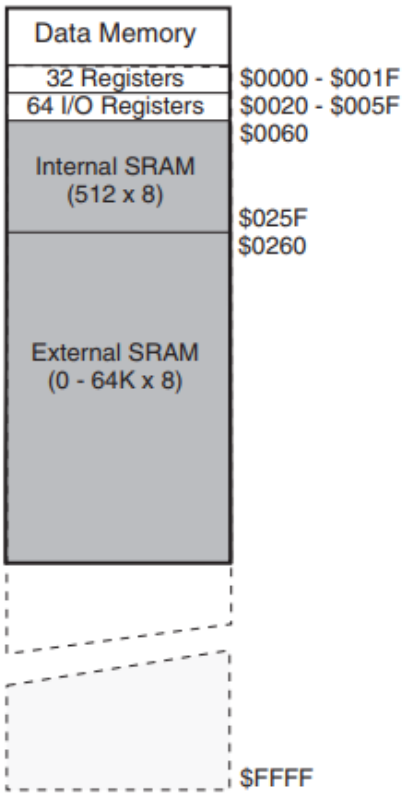

Рисунок 1.4 – Карта распределения памяти ATMega8515

Для записи пяти значений выберем пять адресов внутренней памяти SRAM, при разработке были использованы адреса $0240 – $0245.

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

Рассмотрим внешний LCD-дисплей LM016L. В целях экономии портов входа/выхода микроконтроллера дисплей настроен на работу в четырехбитном режиме. Команды и символы дисплея кодируются восемью разрядами, но в четырехбитном режиме вместо одной пересылки восьми бит происходит две пересылки четырех. Для работы дисплея в таком режиме

нужно в два раза меньше портов микроконтроллера, но каждое обращение к дисплею состоит из двух частей.

Дисплей работает в режиме команды или вывода символа, что регулируется при помощи входа RS.

На таблице 1.1 приведены две основные команды дисплея вместе с сигналами, которые необходимо подать на входы дисплея для их вызова.

Таблица 1.1 – Используемые команды дисплея LM016L

Команда	RS	D7	D6	D5	D4	D3	D2	D1	D0
Очистка дисплея	0	0	0	0	0	0	0	0	1
Установка курсора в начало	0	0	0	0	0	0	0	1	0

В режиме символа используются коды, представленные на таблице 1.2. Как видно, для цифр вторые четыре бита совпадают с двоичным представлением цифры.

Таблица 1.2 – Коды символов дисплея

Символ	Первые 4 бита	Вторые 4 бита	RS
« »	0010	0000	1
«!»	0010	0001	1
«0»	0011	0000	1
«1»	0011	0001	1
«2»	0011	0010	1
•••			
«:»	0011	1010	1

Таким образом для вывода цифры на дисплей необходимо при установленном RS = 1 передать сначала 4 бита 0011, а затем передать

двоичный код цифры. Появляется проблема вывода чисел, имеющих более одного десятичного разряда. Для решения проблемы решено было хранить числа в двоично-десятичном формате, для чего при выводе результата необходимо реализовать двоично-десятичную арифметику. Будет разумно написать функцию вывода символа на дисплей, принимающую первые и вторые 4 бита независимо из основной программы.

Третий 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 1.3.

Таблица 1.3 – 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 1.4.

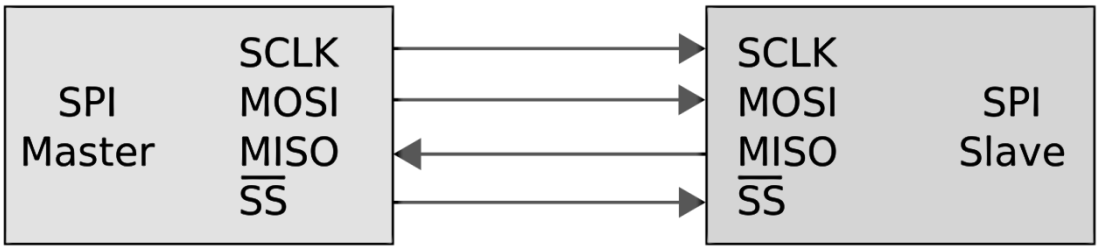

Рисунок 1.5 – Соединение двух устройств по стандарту SPI

2 Процесс работы и схемы алгоритмов

Схема основного алгоритмы программы МК приведена на рисунке 2.1. Основной алгоритм заключается в определении нажатой кнопки и выборе соответствующих действий.

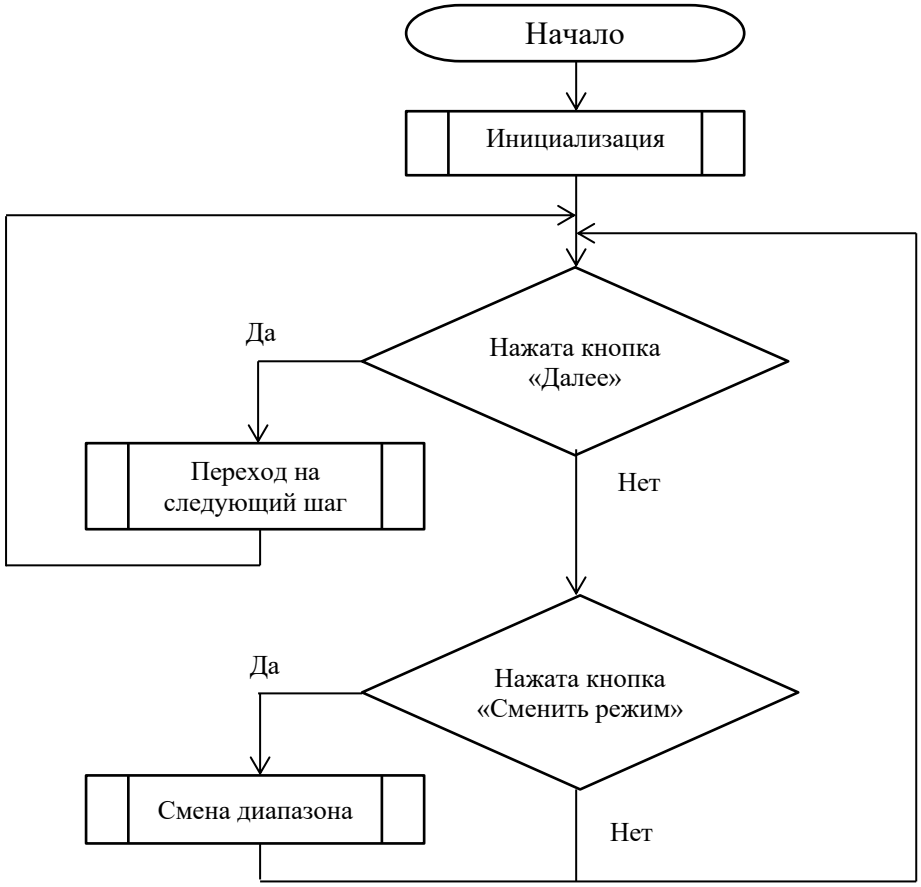

Рисунок 2.1 – Основной алгоритм

Смена диапазона по нажатию соответствующей кнопки не отличается сложностью, важно только контролировать граничное значение – после четвертого режима происходит возвращение в первый. Соответствующий алгоритм представлен на рисунке 2.2.

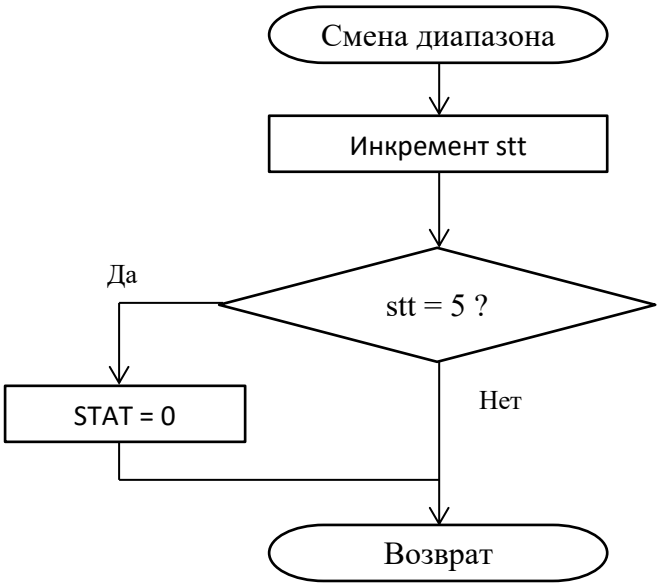

Рисунок 2.2 – Алгоритм смены диапазона

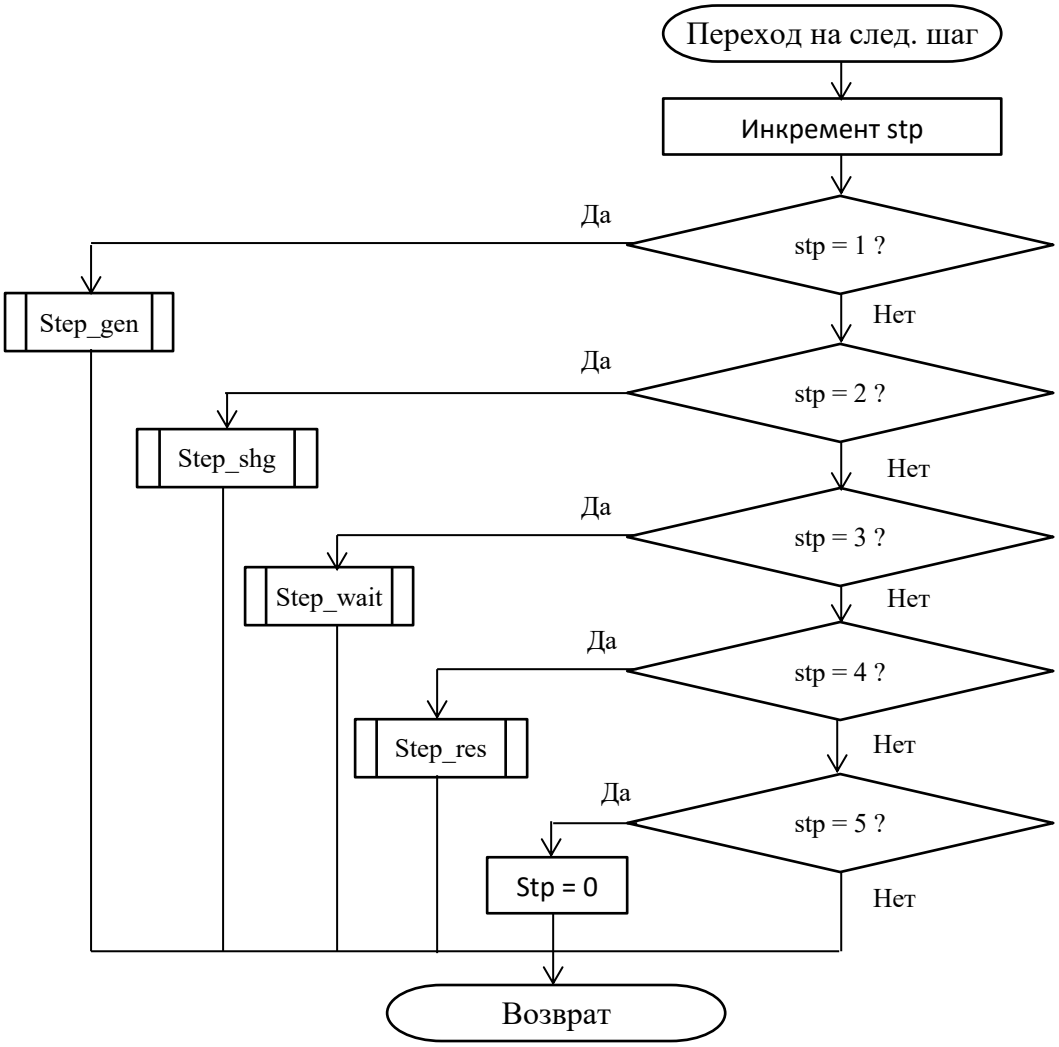

Рисунок 2.3 – Алгоритм перехода на следующий шаг

Алгоритм перехода к следующему шагу приведен на рисунке 2.3, где показаны наименования участков программы, соответствующих каждому шагу, а также сброс счетчика шага в 0 по достижению границы его значений.

Действие кнопки перехода к следующему шагу зависит от шага, от выбранного диапазона и от истории испытаний, ведь результаты 5 испытаний хранятся в памяти, при накоплении 5 результатов происходит их вывод вовне при помощи протокола UART.

Алгоритм для первого шага заключается в ожидании нажатия кнопки.

На втором шаге начинается генерация случайного числа в диапазоне, выбранном пользователем. При нажатии кнопки в момент перехода на второй шаг запускается таймер, который «бежит» по диапазону до момента прерывания отсчета. Так как пользователь может остановить отсчет в любой момент, и таймер задерживается на каждом значении диапазона одинаковое время, то вероятность каждого значения из диапазона зависит только от времени нажатия кнопки. Высокая скорость пробега значительно усложняет возможность предсказания сгенерированного числа. Таким образом, число можно считать псевдослучайным. Дисплей на этом шаге очищается.

На третьем шаге отсчет останавливается, и сгенерированное псевдослучайное число выводится на LCD-дисплей.

На четвертом шаге снова запускается таймер, но уже настроенный для отсчета по секундам. Здесь на дисплей после сгенерированного числа выводится символ «!», показывающий пользователю, что именно такое время ему необходимо засечь.

На пятом шаге происходит остановка таймера. Замеренное время выводится на дисплей, также выводится разность этого времени и времени, которое необходимо было засечь. Разница сохраняется в память МК. Если в этот момент было выполнено 5 испытаний, что фиксируется в специальной переменной, то происходит передача 5 записей из памяти по UART. Сообщение, передаваемое по UART, начинается со строки «res:». Алгоритм пятого шага приведен на рисунке 2.4.

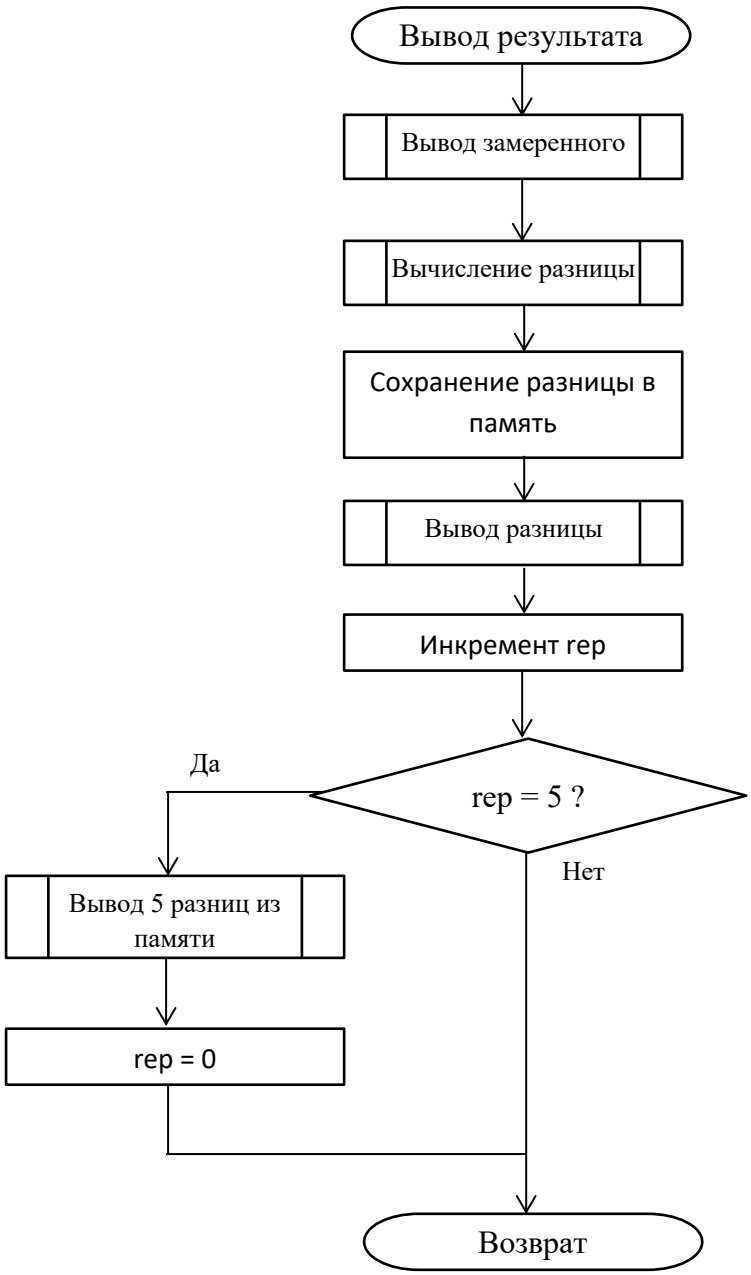

Рисунок 2.4 – Шаг вывода результатов

При нажатии кнопки на пятом шаге происходит переход к первому шагу, но, если вывода 5 значений не происходило, увеличивается на единицу переменная, отвечающая за кол-во выполненных испытаний. Если вывод 5 значений из памяти по UART произошел, то начинается новая пятерка испытаний, и соответственно счетчик количества испытаний равен 0.

Для расчета разности засеченного времени применяется двоично-десятичные числа переводятся в двоичный формат, в котором выполняется вычитание, после чего результат переводится в двоично-десятичную форму.

Так как цифры в двоично-десятичном представлении хранятся отдельно, перевод в двоичную форму проводится по следующей формуле:

$$10 * < 1 \text{ цифра } A > + < 2 \text{ цифра } A > - 10 * < 1 \text{ цифра } B > - < 2 \text{ цифра } B >,$$

где A – уменьшаемое, а B – вычитаемое.

Было решено назвать эталон wit (сокращенное wait), причем старший десятичный разряд хранится в переменной with, младший – в witl. Время, которое засекает пользователь, сохраняется в переменные resh и resl по тому же правилу.

Алгоритм вычитания показан на рисунке 2.5, а перевода из двоичной формы в двоично-десятичную представлен на рисунке 2.6.

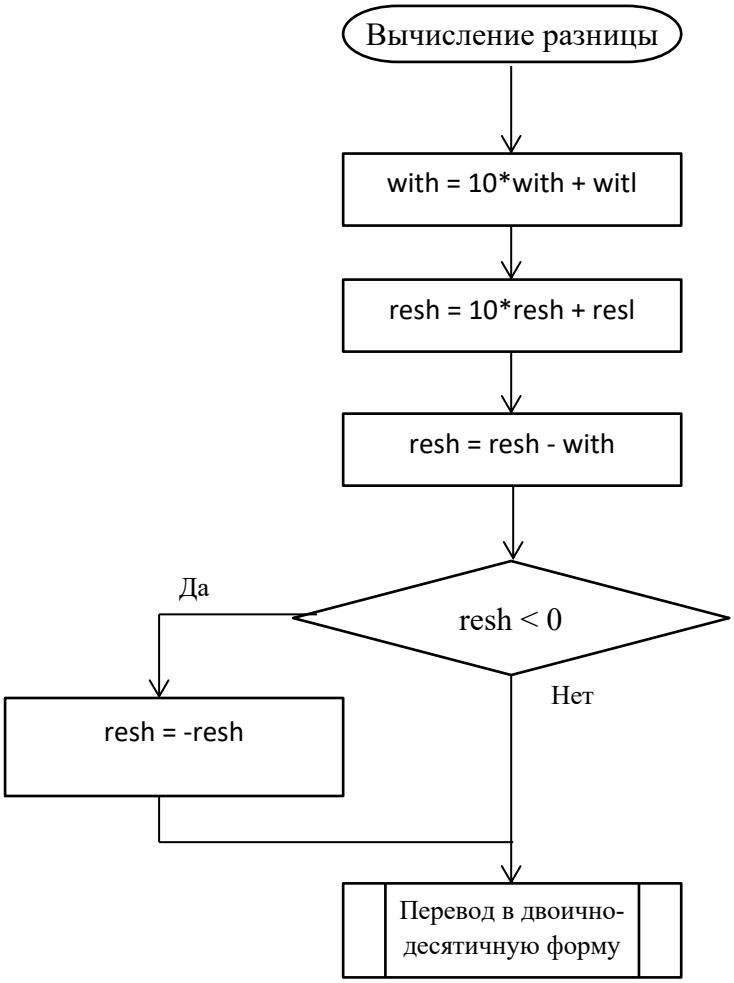

Рисунок 2.5 – Вычитание

С точки зрения реализации при resh < 0 организуем перевод resh в прямой код. Разряды числа будем хранить в разных регистрах, 4 бита будут содержать число, остальные 4 бита будут функциональными. 2 бита из четырех можно будет использовать, чтобы сразу посылать сообщения в формате понятном дисплею.

Для перевода из двоичной в двоично-десятичную форму запускается цикл, цель которого определить, сколько десяток может уместить двоичное число. Количество десяток становится старшим разрядом двоично-десятичного числа, а остаток – младшим.

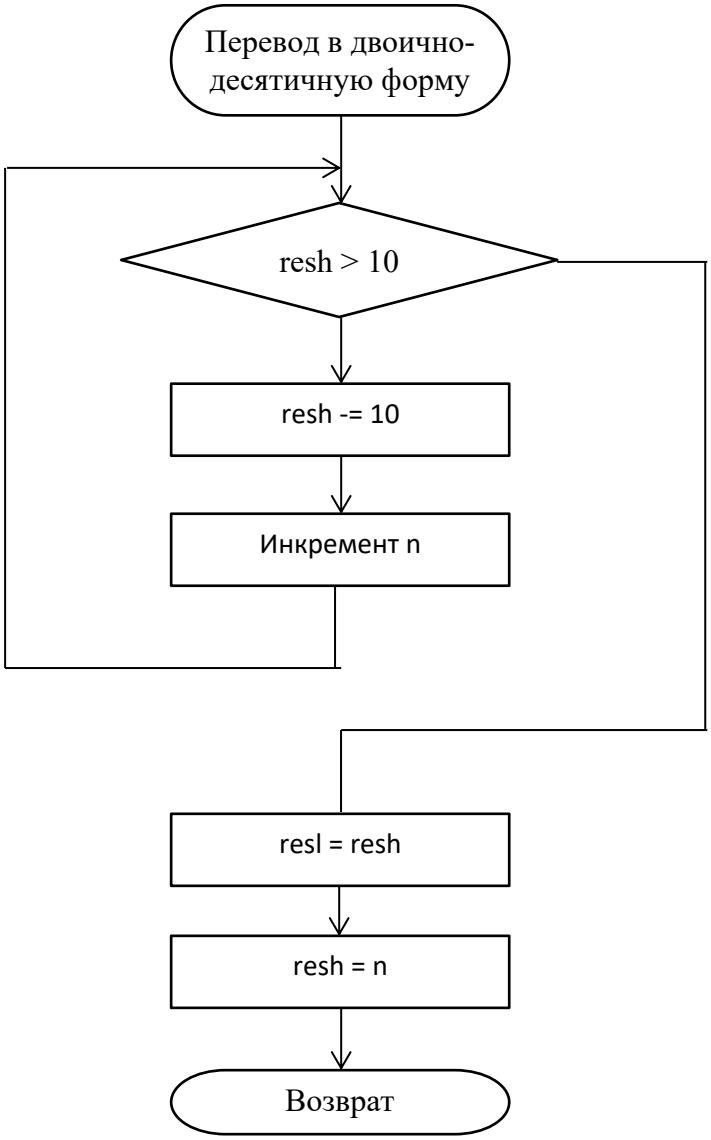

Рисунок 2.6 – Перевод в двоично-десятичную форму

3 Описание функциональной схемы

Помимо самого микроконтроллера ATMega8515 в систему входят две кнопки, LCD-дисплей, программатор и вход 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.

Порт C отведен под управление LCD-дисплеем. Разъем VDD соединяется с источником питания, VSS, VEE и RW замыкаются на землю. Разъем для стробирующего сигнала E соединяется с PC7 МК, а разъем RS, отвечающий за переключение режима команда/символ дисплея, связывается с PC6. Разъемы PC0-PC3 подключены к входам D4-D7 дисплея для обеспечения четырехбитного режима работы.

Система управляется двумя кнопками, подключенными к порту A. Кнопка «STEP» подключается к PA0 и отвечает за переход к следующему шагу тестирования тренажера. Кнопка «STAT», подключенная к PA1, переключает диапазон генерации времени. Кнопка «RESET» подключается к соответствующему входу микроконтроллера.

4 Описание принципиальной схемы

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

Выбранный LCD-дисплей LM016L был подключен в соответствие с рекомендациями производителя с использованием питания 5 В. Так как предполагается работа в четырехбитном режиме, некоторые входы дисплея не понадобились, а другие оказались замкнуты на землю. Конечный вариант подключения дисплея приведен на рисунке 4.1.

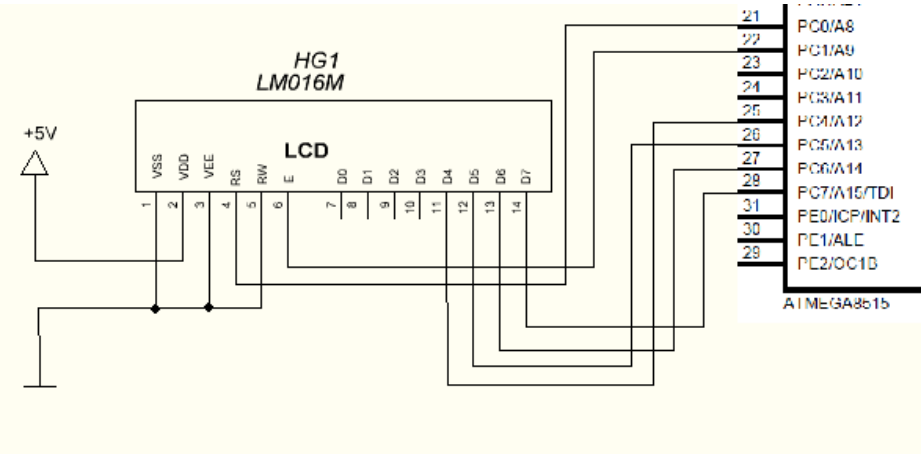

Рисунок 4.1 – Схема подключения дисплея

Для подключения MAX232 необходимо использовать 4 полярных конденсатора емкостью 1 мкФ каждый, здесь тоже используется питание 5 В, схема подключения приведена на рисунке 4.2.

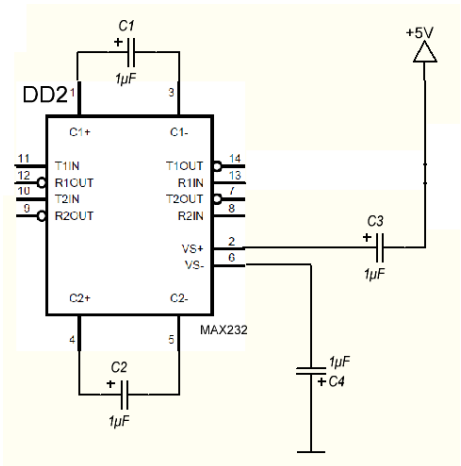

Рисунок 4.2 – Подключение драйвера MAX232

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 4.3.

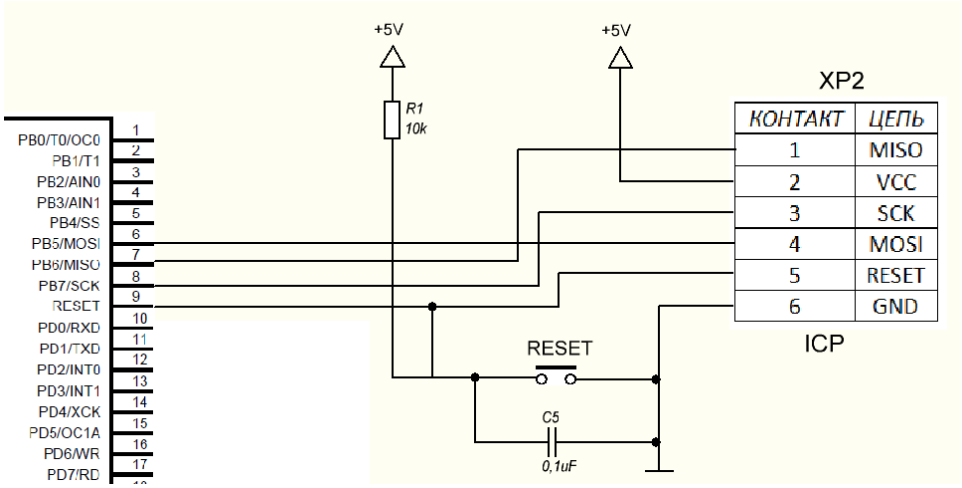

Рисунок 4.3 – Подключение программатора AVR ISP500

На рисунке 4.4 показано подключение к разъему питания с конденсаторами, позволяющими минимизировать негативное влияние перепадов напряжения.

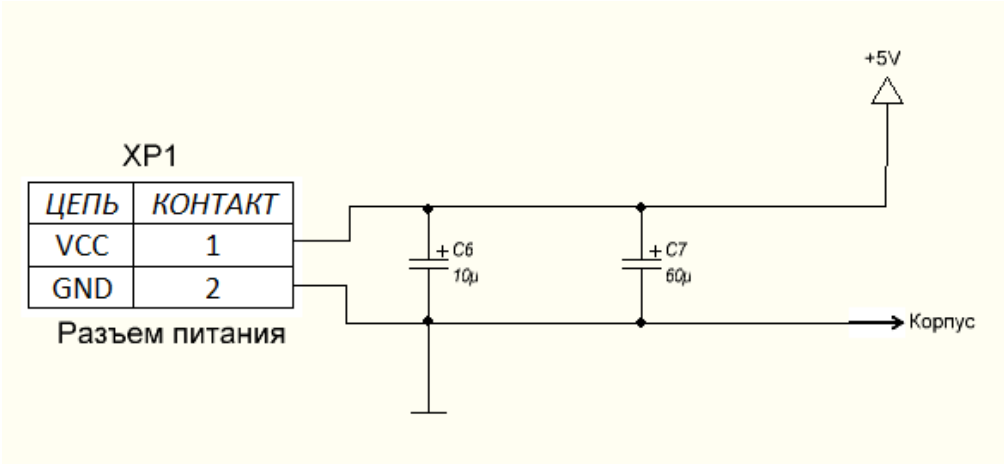

Рисунок 4.4 – Подключение к разъему питания

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:

$$P_{\text{общ.раб}} = P_{ATMega8515} + P_{LM016L} + P_{MAX232} + P_{\text{резисторов}}$$

Отдельные мощности рассчитываются следующим образом:

$$P = I_{max} * U_{\text{пит}}$$

Для определения тока микроконтроллера возьмем график, изображенный на рисунке 5.1. Микроконтроллер работает при напряжении питания 5В на частоте 3,69 МГц.

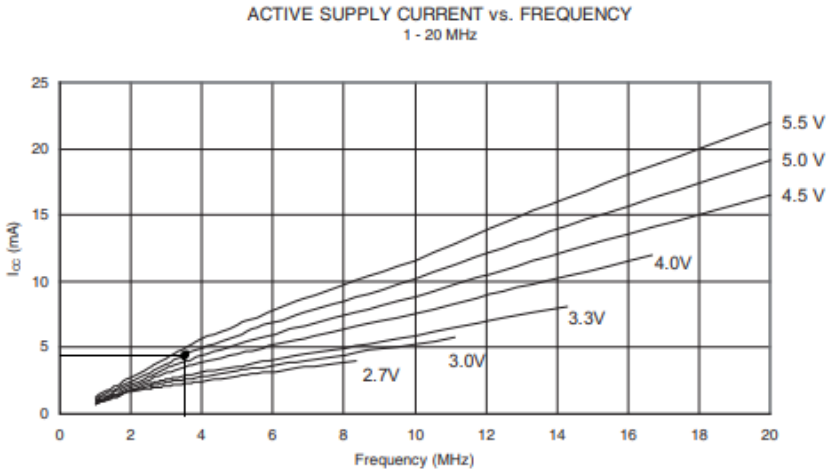

Рисунок 5.1 – Распределение тока по частотам

На таблице 5.1 приведены все значения необходимые для расчета полной потребляемой мощности. Ток микроконтроллера умножается на количество используемых входов.

Таблица 5.1 – Значения расчетных величин

Устройство	I_{max}	$U_{\text{пит}}$
ATMega8515	4,93мА*16	5В
MAX232	10мА	5В
LM016L	80мА	5В
ISP500	25мА/30мА	5В

Каждый резистор потребляет до 2.5 мВт. В схеме использован только один внешний резистор при подключении программатора, он работает на напряжении 5В.

$$P_{\text{общ.раб}} = P_{ATMega8515} + P_{LM016L} + P_{MAX232} + P_{\text{резисторов}} =$$
$$= 5\text{В} * 4{,}93\text{мА} * 16 + 10\text{мА} * 5\text{В} + 80\text{мА} * 5\text{В} + 2{,}5\text{мВт} =$$
$$= 846{,}9\text{мВт}$$

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

Рассчитаем также потребляемую мощность при подключенном программаторе:

$$P_{\text{общ}} = P_{ATMega8515} + P_{LM016L} + P_{MAX232} + P_{\text{резисторов}} + P_{ISP500} =$$
$$= 846{,}9\text{мВт} + 5\text{В} * 25\text{мА} = 971{,}9\text{мВт}$$

В режиме программирования контроллера программатором мощности еще увеличится:

$$P_{\text{общ.прог}} = P_{ATMega8515} + P_{LM016L} + P_{MAX232} + P_{\text{резисторов}} + P_{ISP500} =$$
$$= 846{,}9\text{мВт} + 5\text{В} * 30\text{мА} = 996{,}9\text{мВт}$$

ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

6 Описание системы разработки

При разработке была использована среда AVR Studio 4, позволяющая писать программы для микроконтроллера ATMega8515 на языке ассемблера AVR. Написанные программы можно компилировать в этой же среде, полученный в результате этого файл с расширением «.hex» загружается в модель микроконтроллера среды Proteus ISIS 7.

Размер полученного файла с расширением «.hex» составил 3КБ.

Схема, собранная в среде Proteus ISIS 7 приведена на рисунке 6.1.

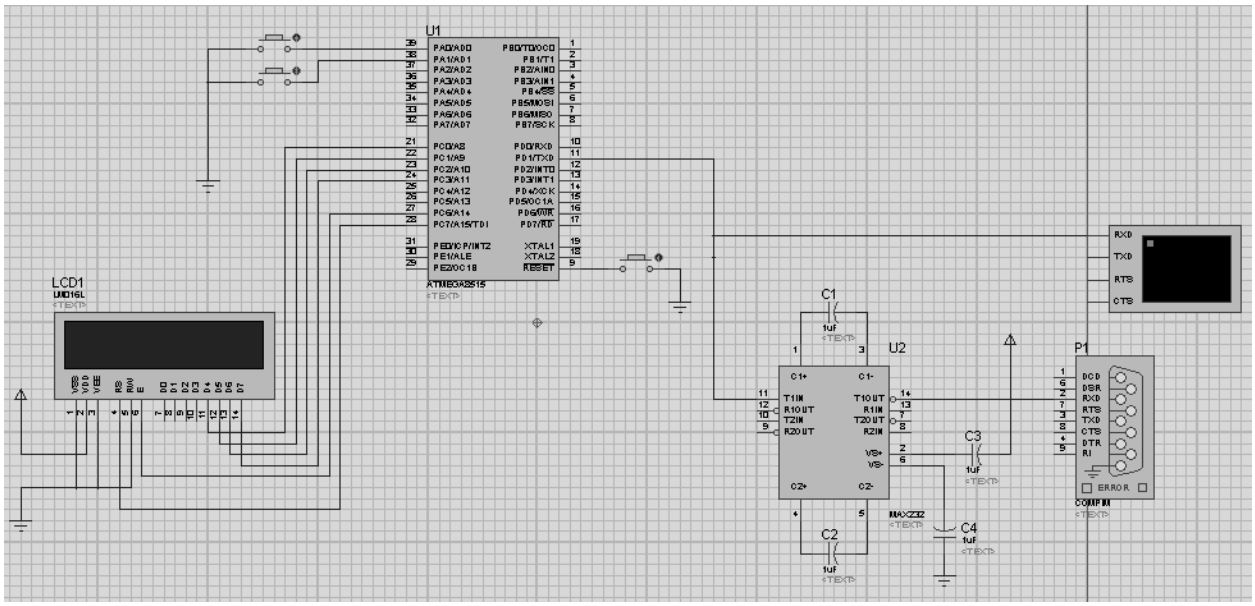

Рисунок 6.1 – Схема в среде Proteus ISIS 7

Микроконтроллер в среде Proteus ISIS настраивается на частоту 3,69 МГц, а терминал, принимающий сообщение по UART – на скорость передачи 19200 бод, все эти настройки приведены на рисунке 6.2.

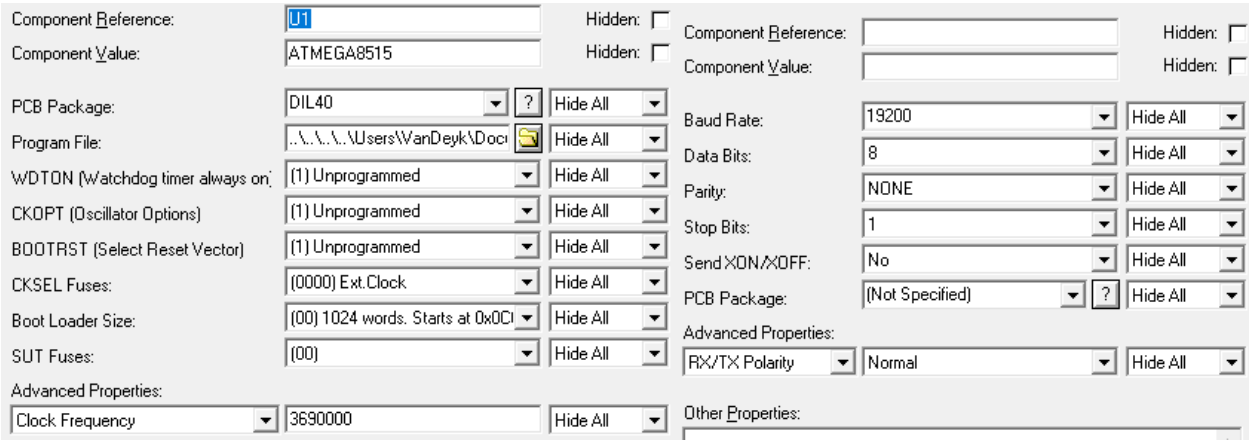

Рисунок 6.2 – Настройка устройств в Proteus ISIS 7

На рисунке 6.3 приведена настройка частоты микроконтроллера в среде AVR Studio 4, выбор актуальной частоты позволяет измерить время выполнения отдельной части кода при помощи сторожевого таймера.

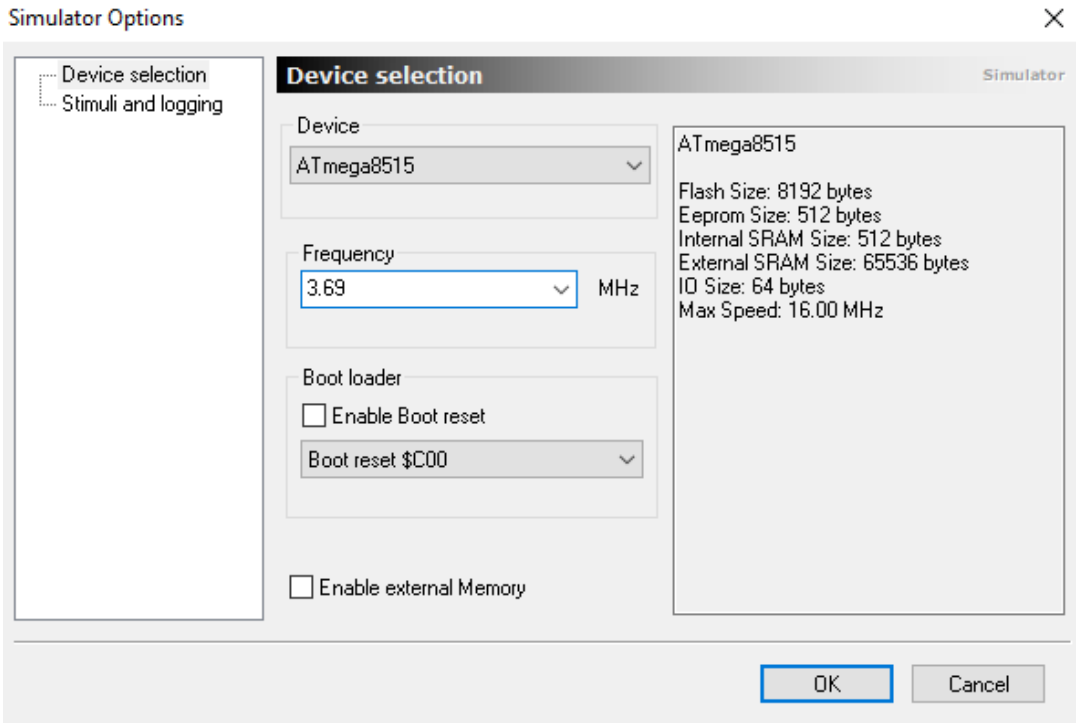

Рисунок 6.3 – Настройка частоты в среде AVR Studio 4

7 Разработка программы

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

Происходит настройка портов ввода и вывода. Порт A настраивается на прием сигналов с кнопок, подключаются встроенные в микроконтроллер подтягивающие резисторы. Выходы портов C и D, задействованные в схеме, настраиваются на вывод.

Происходит настройка таймера. Вариант заполнения регистра TIMSK, представленный на рисунке 7.1, разрешает прерывание по переполнению шестнадцатиразрядного таймера T1. Шестнадцатиразрядный таймер позволяет отсчитывать большие промежутки времени, в нашем случае – секунды.

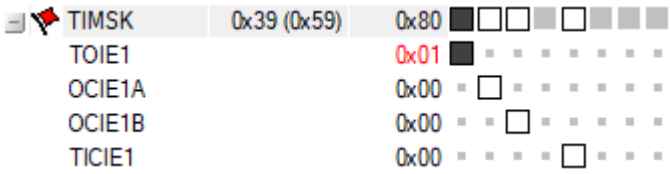

Рисунок 7.1 – Регистр TIMSK

При запуске таймера для отсчета секунд используется предделитель 64, то есть тактовый сигнал делится на 64, для этого в регистр TCCR1B записывается значение 0x03.

Отсчет времени ограничен значением, записанным в регистр TCNT1.

$$T = N \cdot Tcnt = N / Fcnt = N \cdot K / Fск = (65536 - TCNT1) \cdot K / Fск,$$

$$1\text{сек} = (65536 - TCNT1) * 64 \ / \ 3{,}69\text{МГц}$$

$$65536 - \frac{3{,}69\text{МГц}}{64} = TCNT1$$

$$TCNT1 = 7879{,}75$$

Внесем в TCNT1 десятеричное число 7880, которое в шестнадцатеричной системе счисления будет иметь вид 1EC8.

Заметим, что генерация псевдослучайного числа происходит при помощи того же таймера, но в этом случае убирается предделитель и в TCNT1 записывается число 0xFFFF.

Для работы с дисплеем и UART кроме таймера применяется задержка, реализованная в виде вложенного цикла, величины этой задержки рассчитывались экспериментально с такой целью, чтобы одна фиксированная функция задержки работала для разных устройств. Разные задачи – вывод на дисплей и UART – требуют задержку различной величины, но в целях универсальности была написана функция задержки delay.

Рассчитаем итоговую длительность задержки в среде AVR Studio 4. Вход в функцию задержки произошел на отметке 1,63мкс.

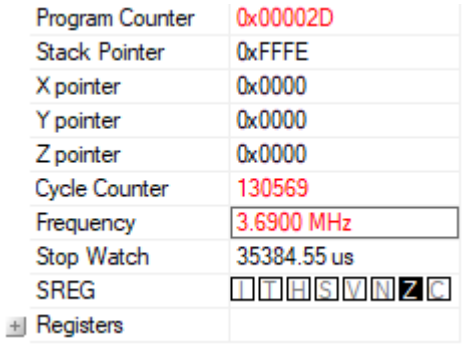

Рисунок 7.2 – Расчет времени программной задержки

Таким образом, величина программной задержки равна 35382,92 микросекунд или 0,035 сек. Меньшие длительности задержки приводили к некорректной работе системы в среде Proteus ISIS 7.

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.

В таблице 7.1 приведены значения регистра контроллера скорости для разных частот и скоростей передачи.

Текущая разработка, несмотря на виртуальную природу, проводилась с учетом ограничений платы STK500, имеющей в своем составе микроконтроллер ATMega8515. Так, программатор AVR ISP500 заточен для работы с этой платой. Максимальная тактовая частота, устанавливаемая с помощью STK500, равна 3,69 МГц, значения частот выше этой величины нас не интересуют. Так как при увеличении ошибки помехозащищённость линии передачи снижается, скорости передачи, имеющие ошибку установки более 1%, использовать не рекомендуется. В итоге была выбрана скорость 19200 бод.

Таблица 7.1 – Значения UBBR

Скорость (бод)	f_{CLK} **=1 МГц**	**Ошибка, %**	f_{CLK} **=2.4576 МГц**	**Ошибка, %**	f_{CLK}**=3.6864 МГц**	**Ошибка, %**
2400	25	0.2	63	0	95	0
4800	12	0.2	31	0	47	0
9600	6	7.5	15	0	23	0
14400	3	7.8	10	3.1	15	0
19200	2	7.8	7	0.0	11	0
28800	1	7.8	4	6.3	7	0
38400	1	22.9	3	0	5	0
57600	0	7.8	2	12.5	3	0
76800	0	22.9	1	0	2	0
115200	0	84.3	0	25.0	1	0

Предполагается работа с дисплеем в четырехбитном режиме, инициализация в котором производится следующей последовательностью сигналов, посылаемых на DL: 3, 3, 3, 2, 2, 1, 0, 15. Вывод одного числа приведен ниже, импульс на стробирующий сигнал подается при помощи команды sbi установки соответствующего бита в «1» и cli в «0»:

```
ldi tmp, 0b00000011 ; Выводим тройку на D4-D7
out PORTC, tmp
rcall delay
```

После вывода всей последовательности 3, 3, 3, 2, 2, 1, 0, 15 таким способом дисплей сообщает о своем готовности к работе мигающим курсором в левом верхнем углу экрана.

Расчет разницы засеченного и эталонного времени включает в себя перевод из двоично-десятичной формы записи в двоичную и обратно, приведем соответствующую часть программы ниже:

```
sub_s1:   ldi tmp, 10
          mul with, tmp
          mov with, r0
          add with, witl
sub_s2:
          mul resh, tmp
          mov resh, r0
          add resh, resl
          sub resh, with
          brmi sub_c2
          rjmp sub_s3
sub_c2: ; Переводим в прямой код, если разница < 0
          ldi dr1, 0xff
          eor resh, dr1
          inc resh
sub_s3:
          ldi dr1, 0
          ldi tmp, 0b00001010
sub_loop: ; Считаем, сколько десяток влезет в число
          cp resh, tmp
          brlo sub_out
          inc dr1
          sub resh, tmp
          rjmp sub_loop
```

Измерим максимальное время выполнения этой операции в «худшем» случае, пусть было необходимо замерить 60 секунд, а пользователь замерил 0 секунд. При таких входных данных мы посетим каждый участок кода и проведем больше всего времени в цикле. Замеры приведены на рисунке 7.3. Итоговое время 17,88 микросекунд.

Рисунок 7.3 – Замеры времени выполнения перевода

8 Тестирование

В начале работы, как показано на рисунке 8.1, происходит инициализация дисплея, и он начинает показывать номер исходного диапазона – «0», пи этом по UART посылается первое сообщение «hi».

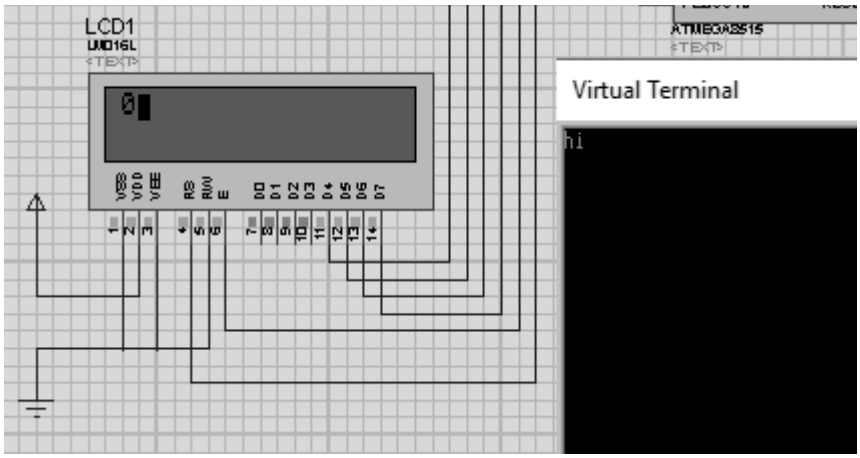

Рисунок 8.1 – Начало работы системы

После одного нажатия кнопки «STEP» дисплей отчищается, и до тех пор, пока пользователь не нажмет кнопку повторно, происходит генерация псевдослучайного числа. Сгенерированное число выводится на дисплей. Следующее нажатие кнопки сообщает системе, что пользователь готов засекать время, система сообщает о начале этого этапа выводом восклицательного знака после сгенерированного числа, как показано на рисунке 8.2.

Рисунок 8.2 – Пользователь начал засекать время

По следующему нажатию кнопки отсчет заканчивается и время, засеченное пользователем, сравнивается со сгенерированным эталоном. На дисплей выводится, сколько на самом деле прошло секунд с начала замера, и какова разница между замеренным временем и эталоном. Дисплей на этом этапе приведен на рисунке 8.3.

Рисунок 8.3 – Вывод измеренного пользователем времени и разницы

По завершении пяти проверок система выводит разницы, которые были посчитаны в течение этих пяти проверок, на выход RS-232 по стандарту UART, что и показано на рисунке 8.4.

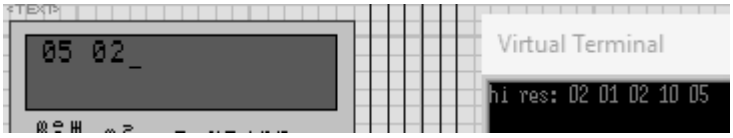

Рисунок 8.4 – Вывод сообщения по завершению пяти испытаний

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был спроектирован тренажер психофизических характеристик человека, позволяющий оценить, насколько точно пользователь способен засекать время.

Программный код системы был написан на языке ассемблера процессоров семейства AVR, для чего была использована среда разработки AVR Studio 4. Было произведено моделирование системы в среде Proteus ISIS 7.6, 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.

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

5. 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.

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

7. ГОСТ 2.701-84 Правила выполнения схем.

8. ГОСТ 2.702-75 Правила выполнения электрических схем.

9. Документация платы STK500 – http://www.mouser.com/ds/2/36/doc1925-476973.pdf.

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ.

ПРИЛОЖЕНИЕ Б. СПЕЦИФИКАЦИЯ.