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)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 **ИФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине «Микропроцессорные системы»

НА ТЕМУ:

Устройство для измерения скорости чтения

Студент	ИУ6-73Б		А.А. Третьяков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель			И.Б. Трамов
		(Подпись, дата)	(И.О. Фамилия)

2024 г.

ЛИСТ ЗАДАНИЯ

РЕФЕРАТ

РПЗ 51 страница, 21 рисунок, 7 таблиц, 11 источников, 3 приложения.

МИКРОКОНТРОЛЛЕР, СИСТЕМА, ТАБЛИЦА ШУЛЬТЕ, ЖК-ДИСПЛЕЙ

Объектом разработки является устройство для измерения скорости чтения.

Основная цель курсовой работы состоит в формировании навыков разработки и проектирования микропроцессорных систем путём освоения современных технологий проектирования систем на основе микроконтроллеров, а также программируемых систем на кристалле.

Цель работы – создание функционального устройства ограниченной сложности, модель устройства и разработка необходимой документации на объект разработки.

Поставленная цель достигается посредством использования Proteus 8.

В процессе работы над курсовым проектом решаются следующие задачи: выбор МК и драйвера обмена данных, создание функциональной и принципиальной схем системы, расчет потребляемой мощности устройства, разработка алгоритма управления и соответствующей программы МК, а также написание сопутствующей документации.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

МК – микроконтроллер.

ТЗ – техническое задание.

Proteus 8 — пакет программ для автоматизированного проектирования (САПР) электронных схем.

ASCII – таблица кодировки символов.

UART – Universal asynchronous receiver/transmitter – последовательный универсальный синхронный/асинхронный приемопередатчик.

SPI – Serial Peripheral Interface – интерфейс для связи МК с другими внешними устройствами.

ЖК-дисплей (LCD) – жидкокристаллический дисплей.

ВВЕДЕНИЕ

В данной работе производится разработка устройства для измерения скорости чтения.

В процессе выполнения работы проведён анализ технического задания, создана концепция устройства, разработаны электрические схемы, построен алгоритм и управляющая программа для МК, выполнено интерактивное моделирование устройства.

Устройство для измерения скорости чтения, согласно ТЗ, предлагается реализовать на базе МК семейства AVR.

Система состоит из МК и 4 блоков светофоров, каждый из которых включает в себя 1 светофор для автомобилей и 2 для пешеходов. На каждом пешеходном светофоре имеется кнопка для вызова пешеходной фазы вне очереди. Также можно в МК загружать два расписания и имеется возможность переключения между ними.

Микроконтроллеры семейства AVR представляют собой одни из наиболее распространённых и широко используемых микропроцессорных устройств в области встроенных систем. Отличительной чертой AVR является их RISC-архитектура (Reduced Instruction Set Computer), обеспечивающая высокую производительность при низком энергопотреблении. Эти микроконтроллеры нашли своё применение в таких областях, как автоматизация процессов, управление бытовой и промышленной техникой, разработка измерительных приборов, управление робототехникой и создание обучающих устройств.

Одним из ключевых преимуществ AVR-микроконтроллеров является их простой в освоении и использовании набор инструкций, состоящий из 120–130 команд. Это позволяет оптимизировать разработку программного обеспечения и сокращает затраты на обучение персонала. К другим важным особенностям относятся:

– наличие встроенной памяти программ и данных различных типов (Flash, EEPROM, SRAM);

– поддержка работы с широким набором периферийных устройств, включая таймеры/счётчики, интерфейсы UART, SPI, I2C, АЦП и ШИМ;

– возможность гибкой настройки тактирования с использованием как внутреннего, так и внешнего генераторов;

– низкое энергопотребление благодаря режимам сна и управления питанием;

– доступность инструментов разработки, таких как Atmel Studio, AVR-GCC и симуляторы.

Актуальность разрабатываемой модели измерителя скорости чтения заключается в необходимости объективной оценки и развития когнитивных способностей человека в современных условиях. Скорость чтения играет важную роль в образовательной и профессиональной деятельности, так как позволяет человеку быстрее обрабатывать и усваивать информацию. Тренировка навыков быстрого и осознанного чтения особенно востребована в эпоху информационной перегрузки, когда эффективность работы с текстами напрямую влияет на результаты обучения и трудовой деятельности.

1 Конструкторская часть

1.1 Анализ требований и принцип работы системы

Исходя из требований, изложенных в техническом задании, можно сделать вывод, что целью системы является обеспечение отображения, обработки и передачи данных, связанных с экспериментами, а также поддержка таймеров и взаимодействие через Bluetooth.

Согласно ТЗ, устройство для измерения скорости чтения должно быть реализовано с помощью отображения на ЖК-дисплее таблицы Шульте.

Таблицы Шульте – таблицы со случайно расположенными объектами (обычно числами или буквами), служащие для проверки и развития быстроты нахождения этих объектов в определённом порядке. Упражнения с таблицами позволяют улучшить периферическое зрительное восприятие, что важно, например, для скорочтения.

Для выполнения данного задания было решено использовать таблицу Шульте 5x5 со случайно размещенными в ней числами от 1 до 25. Ниже приведен пример такой таблицы (рисунок 1).

20	13	16	9	17
7	10	14	8	4
15	24	11	6	23
22	25	18	2	1
12	21	3	19	5

Рисунок 1 – Выбранная для реализации таблица Шульте

При выполнении задания требуется учесть и реализовать следующие пункты:

– многофункциональность (необходимо реализовать работу с несколькими независимыми LCD-дисплеями, отображающими таблицы и результаты),

– взаимодействие с внешними устройствами (требуется передавать данные через Bluetooth),

– энергонезависимость памяти (сохранение рекордов в EEPROM),

– управляемость (реализация управления с помощью кнопок).

Принцип работы системы заключается в следующем:

1) при нажатии на кнопку на 3 ЖК-дисплея (2 строки на первом, 2 строки на втором и 1 строка на третьем) выводится случайно сгенерированная таблица Шульте 5x5, состоящая из чисел от 1 до 25;

2) при нажатии на другую кнопку запускается отсчет времени чтения (последовательного поиска чисел) таблицы;

3) при повторном нажатии второй кнопки отсчет времени останавливается;

4) на те же 3 ЖК-дисплея выводятся результаты временных замеров пользователя, а также в последней строке третьего ЖК-дисплея выводится среднее время чтения пользователя;

5) установка нового рекорда сопровождается звуковым сигналом;

6) вся информация (оповещение о новом рекорде, время замеров, среднее время чтения пользователя) передается на телефон.

Разработанная структурная схема модели контроллера беговой дорожки представлена на рисунке 1.

Рисунок 1 – Структурная схема устройства

1.2 Проектирование функциональной схемы

В этом разделе приведено функциональное описание работы системы и проектирование функциональной схемы.

1.2.1 Микроконтроллер ATmega16

Основным элементом разрабатываемого устройства является микроконтроллер (МК). Существует множество семейств МК, для разработки выберем из тех, что являются основными:

- 8051 – это 8-битное семейство МК, разработанное компанией Intel.
- PIC – это серия МК, разработанная компанией Microchip;
- AVR – это серия МК разработанная компанией Atmel;
- ARM – одним из семейств процессоров на базе архитектуры RISC, разработанным компанией Advanced RISC Machines.

Сравнение семейств показано в таблице 1.

Таблица 1 – Сравнение семейств МК

Критерий	8051	PIC	AVR	ARM
Разрядность	8 бит	8/16/32 бит	8/32 бит	32 бит, иногда 64 бит
Интерфейсы	UART, USART, SPI, I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, иногда CAN, USB, Ethernet	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI, IrDA

Продолжение таблицы 1

Критерий	8051	PIC	AVR	ARM
Скорость	12 тактов на инстру-кцию	4 такта на инструкцию	1 такт на инструкцию	1 такт на инструкцию
Память	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
Энергопо-требление	Среднее	Низкое	Низкое	Низкое
Объем FLASH памяти	До 128 Кб	До 512 Кб	До 256 Кб	До 192 Кб

Согласно ТЗ, было выбрано семейство AVR.

Семейство AVR – это группа 8-битных микроконтроллеров, разработанных компанией Atmel (в настоящее время часть Microchip Technology). Эти микроконтроллеры получили широкую популярность благодаря своей простоте, надежности и высокой производительности. AVR-микроконтроллеры используются в различных приложениях, от простых устройств до сложных систем управления [1].

AVR в свою очередь делятся на семейства [2]:

1. TinyAVR, имеющие следующие характеристики:
   - Flash-память до 16 Кбайт;
   - RAM до 512 байт;
   - ROM до 512 байт;
   - число пинов (ножек) ввода-вывода 4–18;
   - небольшой набор периферии.
2. MegaAVR, имеющие следующие характеристики:
   - FLASH до 256 Кбайт;
   - RAM до 16 Кбайт;
   - ROM до 4 Кбайт;
   - число пинов ввода-вывода 23–86;

– расширенная система команд (ассемблер и C) и периферии.

3. XMEGA AVR, имеющие следующие характеристики:

– FLASH до 384 Кбайт;

– RAM до 32 Кбайт;

– ROM до 4 Кбайт;

– четырехканальный контроллер DMA (для быстрой работы с памятью и вводом/выводом).

Выберем подсемейство MegaAVR, так как оно имеет больше пинов и объем памяти, чем TinyAVR, а также поддерживает C. XMEGA AVR не был выбран так как они лучше по характеристикам, чем необходимо, то есть весь их потенциал не будет раскрыт.

Для реализации проекта был выбран микроконтроллер ATmega16 из подсемейства MegaAVR семейства AVR, обладающий всем необходимым функционалом:

– интерфейс SPI для программирования микроконтроллера;

– интерфейс UART для обмена данными;

– 1 Кбайт ОЗУ, что вдвое больше по сравнению с ATmega8515, обеспечивая больше возможностей для работы с данными;

– две пары 8-разрядных и 16-разрядных таймеров/счетчиков, что позволяет увеличить функциональность и точность измерений;

– 16 Кбайт FLASH памяти, что также вдвое больше, обеспечивая больший объем памяти для программного кода;

– три внешних источника прерываний, что идеально подходит для управления внешними событиями;

– частота работы до 16 МГц, обеспечивающая высокую производительность.

Кроме того, с данным микроконтроллером уже есть опыт работы, что упростит разработку и не потребует времени на изучение нового функционала.

ATmega16 – это экономичный 8-разрядный микроконтроллер, основанный на архитектуре AVR RISC, который обеспечивает

производительность до 1 миллиона операций в секунду на 1 МГц синхронизации. Это достигается благодаря выполнению большинства инструкций за один машинный цикл. Микроконтроллер также позволяет оптимизировать энергопотребление за счет изменения частоты синхронизации, что делает его подходящим для использования в энергоэффективных системах. Структурная схема МК показана на рисунке 2 и УГО на рисунке 3.

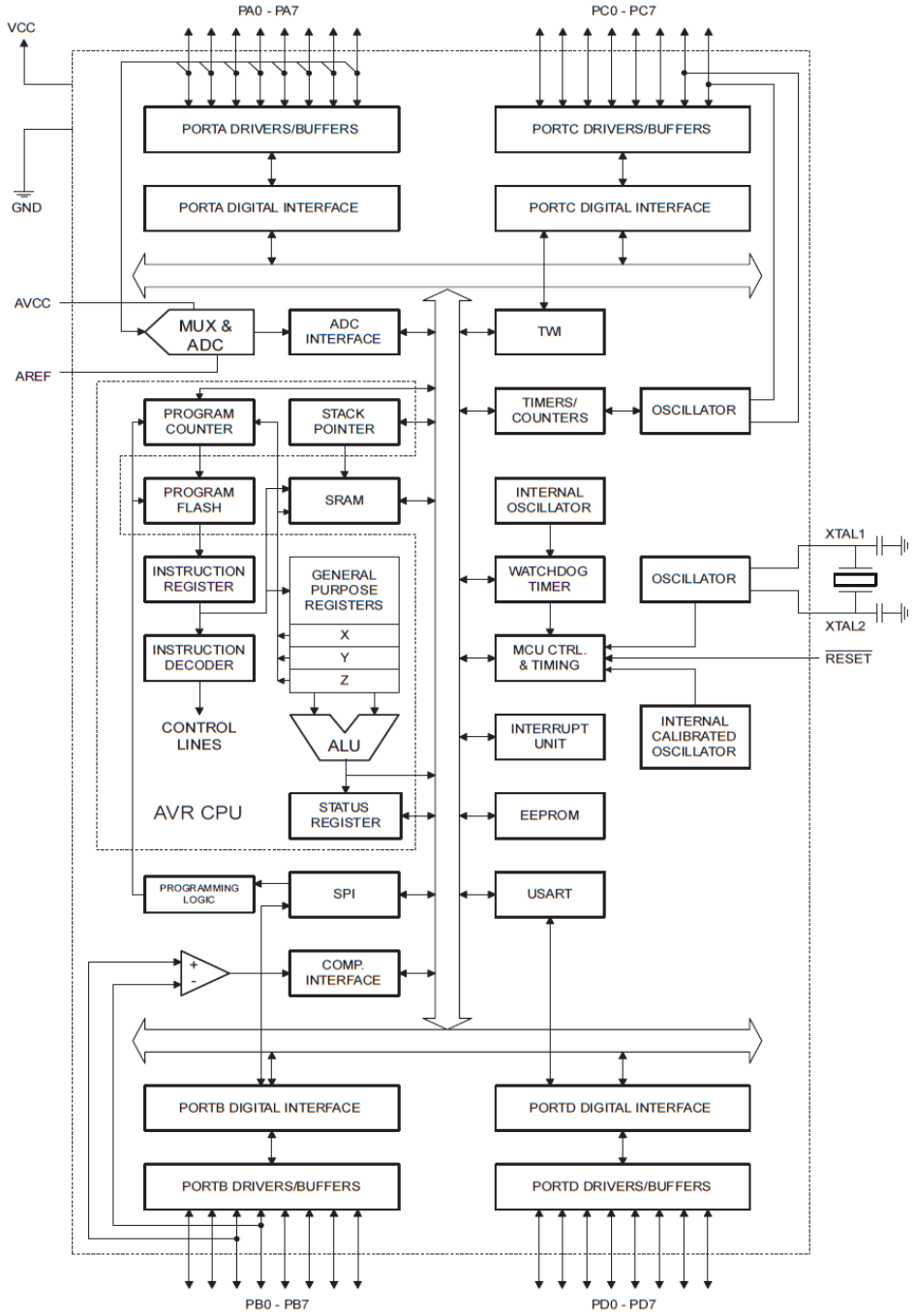


Рисунок 2 – Структурная схема МК ATmega16

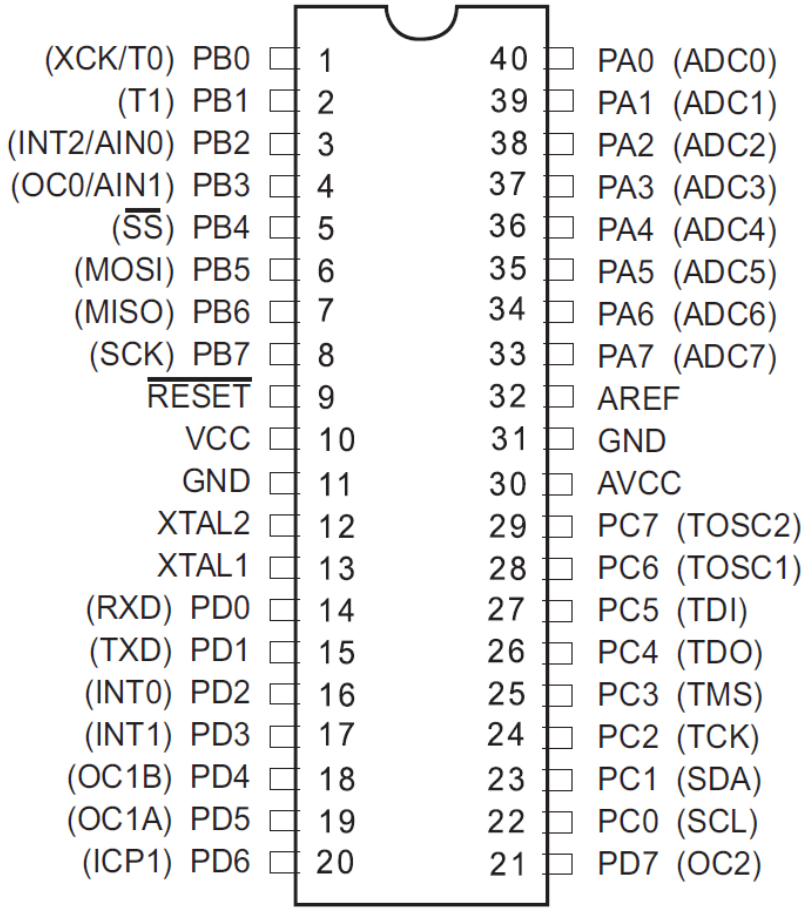


Рисунок 3 – УГО МК ATmega16

Он обладает следующими характеристиками [3]:

1) RISC архитектура:

– мощная система команд с 130 инструкциями, большинство из которых выполняются за один машинный цикл;

– 32 восьмиразрядных рабочих регистров общего назначения;

– производительность до 16 миллиона операций в секунду при частоте 16 МГц;

– встроенное умножение за 2 цикла.

2) Энергонезависимые память программ и данных:

– 8 Кб внутрисхемно-программируемой flash-памяти с возможностью самозаписи;

– возможность внутрисхемного программирования программой во встроенном секторе начальной загрузки;

– возможность считывания во время записи;

– 512 байт ЭППЗУ (EEPROM);

– 512 байт внутреннего статического ОЗУ;

– возможность организации внешней области памяти размером до 64 Кб;

– программирование битов защиты программного обеспечения.

3) Периферийные устройства:

– один 8-разрядный таймер-счетчик с отдельным предделителем и режимом компаратора;

– один 16-разрядный таймер-счетчик с отдельным предделителем и режимом компаратора;

– три канала ШИМ (широтно-импульсная модуляция);

– программируемый последовательный UART (устройство синхронной или асинхронной приемопередачи);

– последовательный интерфейс SPI с режимами главный и подчиненный;

– программируемый сторожевой таймер с отдельным встроенным генератором;

– встроенный аналоговый компаратор.

4) Специальные функции микроконтроллера:

– сброс при подаче питания и программируемый супервизор питания;

– встроенный калиброванный RC-генератор;

– внутренние и внешние источники запросов на прерывание;

– три режима управления энергопотреблением: холостой ход (Idle), пониженное потребление (Power-down) и дежурный (Standby).

5) Ввод-вывод:

– 35 программируемых линий ввода-вывода;

– 47 регистров ввода/вывода.

6) Напряжение питания: 4.5 – 5.5В.

7) Рабочая частота: 1 – 16 МГц.

1.2.2 Используемые элементы

Для функционирования устройства для измерения скорости чтения в МК ATmega16 задействованы не все элементы его архитектуры. Выделим и опишем те, что используются во время функционирования схемы [4].

Порты A, B, C, D – назначения каждого из них описано в разделе 1.2.3.

Указатель стека – используется для работы со стеком, при вызове подпрограмм. В коде они присутствуют.

SRAM – статическая память МК, где хранятся объявленные переменные.

Регистры общего назначения – предназначены для хранения операндов арифметико-логических операций, а также адресов или отдельных компонентов адресов ячеек памяти.

АЛУ – блок процессора, который под управлением устройства управления служит для выполнения арифметических и логических преобразований над данными.

SREG – регистр состояния, содержит набор флагов, показывающих текущее состояние работы микроконтроллера.

SPI – интерфейс для связи МК с другими внешними устройствами. В проекте используется только для прошивки МК.

Программный счетчик – используется для указания следующей команды выполняемой программы.

Память Flash – память МК, в котором хранится загруженная в него программа.

Регистры команды – содержит исполняемую в текущий момент (или следующий) команду, то есть команду, адресуемую счетчиком команд.

Блок расшифровки команд – выделяет код операции и операнды команды, а затем вызывает микропрограмму, которая исполняет данную команду.

Сигналы управления – синхронизируют обработку данных.

Логика программирования – устанавливает логику того, как программа будет вшита в МК.

Таймеры/счетчики – включает в себя 8-разрядные таймеры T0, T2 и 16-разрядный таймер T1. Во время работы программы используется только таймер T1. Он считает время, с которой пользователь прочитает элементы таблицы Шульте.

Управление синхронизацией и сбросом – в этом блоке обрабатываются тактовые сигналы и принимается сигнал сброса.

Прерывания – контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств МК (таймеров, портов ввода/вывода). В данном проекте широко используется система прерываний. Обрабатываются прерывания при переполнении счетчика T1 и внешние прерывания от управляющих кнопок.

USART – через этот интерфейс в МК передается информация из ПЭВМ. В регистр USART информация попадает через порт PD0 (RXD). Также с выхода PD1 (TXD) информация о тренировке подается на Bluetooth-модуль.

Генератор – генератор тактовых импульсов. Необходим для синхронизации работы МК.

1.2.3 Распределение портов

МК ATmega16 содержит четыре порта – A, B, C и D. Опишем назначение каждого из них.

Порт A:

– PA0, PA2, PA4 – отвечают за RS-сигналы первого, второго и третьего ЖК-дисплеев соответственно;

– PA1, PA3, PA5 – отвечают за E-сигналы первого, второго и третьего ЖК-дисплеев соответственно;

– PA6 – отвечает за подачу сигнала на Buzzer в случае установления нового рекорда.

Порт B:

– PA5 (MOSI) – участвует при программировании МК;

– PA6 (MISO) – участвует при программировании МК;

– PA7 (SCK) – участвует при программировании МК;

Порт C: все выходы этого порта (PC0-PC7) отвечают за данные, передаваемые на ЖК-дисплеи.

Порт D:

– PD0 – отвечает за принятие вводимой информации (RXD);

– PD1 – отвечает за выдачу выходных данных по USART (TXD);

– PD2 – отвечает за запуск и остановку таймера отсчета времени, по остановке времени информация о временных замерах и среднем времени скорости чтения выводятся на ЖК-дисплеи;

– PD3 – отвечает за генерацию новой таблицы Шульте;

1.2.4 Организация памяти

Схема организации памяти МК ATmega16 показана на рисунке 4.

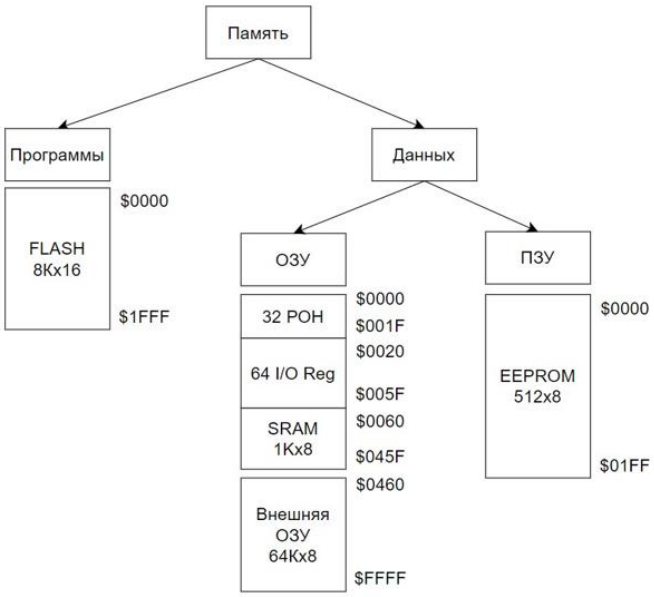


Рисунок 4 – Организация памяти МК ATmega

Память программы предназначена для хранения последовательности команд, управляющих функционированием микроконтроллера, и имеет 16-ти битную организацию.

Все AVR имеют Flash-память программ, в выбранном МК её объем 8Кх16, то есть длина команды 16 разрядов. Информацию в flash-память можно мгновенно стереть и записать. В память микроконтроллера программа записывается с помощью программатора.

Память данных делится на три части:

1) Регистровая память – 32 регистра общего назначения и служебные регистры ввода/вывода.

2) Оперативная память (ОЗУ) – МК ATmega16 имеет объем внутреннего SRAM 1 Кбайт (с адреса $0060 до $045F). Число циклов чтения и записи в RAM не ограничено, но при отключении питания вся информация теряется. Также есть возможность подключить к МК ATmega16 внешнее ОЗУ объемом до 64Кбайт.

3) Энергонезависимая память (EEPROM) – эта память доступна МК в ходе выполнения, она предназначена для хранения промежуточных результатов. В выбранном МК ее объем составляет 512 байт. Также в неё могут быть загружены данные через программатор.

1.2.5 Прием данных от ПЭВМ

Приём данных от ПЭВМ происходит через драйвер MAX232. MAX232 – интегральная схема, преобразующая сигналы последовательного порта RS-232 в цифровые сигналы.

RS-232 – стандарт физического уровня для синхронного и асинхронного интерфейса (USART и UART). Обеспечивает передачу данных и некоторых специальных сигналов между терминалом и устройством приема. Сигнал, поступающий от интерфейса RS-232, через преобразователь передается в микроконтроллер на вход RXD.

К внешнему устройству MAX232 подключен через разъем DB9. Условно обозначен на схеме как XS1.

Внутреннее устройство MAX232 показано на рисунке 5. Назначение пинов описано в таблице 2 [5].

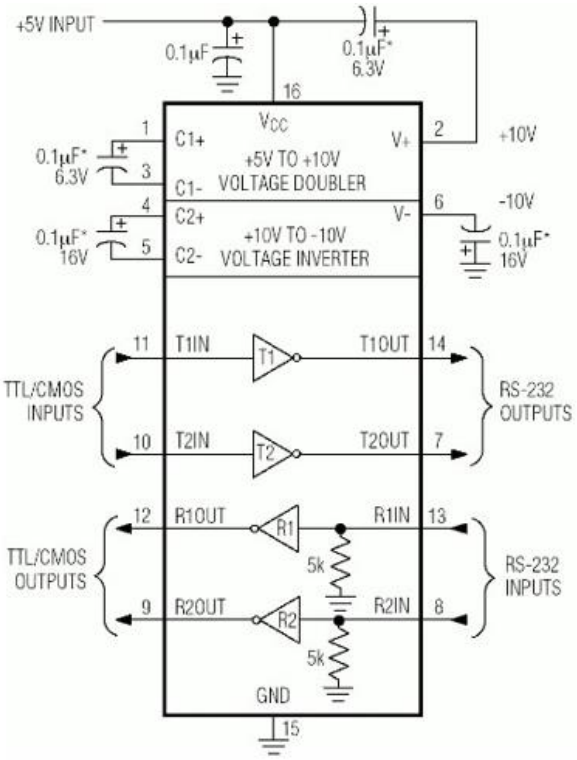


Рисунок 5 – Преобразователь MAX232

Таблица 2 - Назначение пинов MAX232

Номер	Имя	Тип	Описание
1	C1+	–	Положительный вывод C1 для подключения конденсатора
2	VS+	O	Выход положительного заряда для накопительного конденсатора
3	C1-	–	Отрицательный вывод C1 для подключения конденсатора
4	C2+	–	Положительный вывод C2 для подключения конденсатора
5	C2-	–	Отрицательный вывод C2 для подключения конденсатора
6	VS-	O	Выход отрицательного заряда для накопительного конденсатора
7, 14	T2OUT, T1OUT	O	Вывод данных по линии RS232
8, 13	R2IN, R1IN	I	Ввод данных по линии RS232

Продолжение таблицы 4

Номер	Имя	Тип	Описание
9, 12	R2OUT, R1OUT	O	Вывод логических данных
10, 11	T2IN, T1IN	I	Ввод логических данных
15	GND	–	Земля
16	Vcc	–	Напряжение питания, подключение к внешнему источнику питания 5 В

Когда микросхема MAX232 получает на вход логический "0" от внешнего устройства, она преобразует его в положительное напряжение от +5 до +15В, а когда получает логическую "1" - преобразует её в отрицательное напряжение от -5 до -15В. Обратные преобразования от RS-232 к внешнему устройству происходят аналогично.

1.2.6 Настройка канала передачи

Для передачи информации на Bluetooth-модуль в МК используется последовательный интерфейс USART. Для его работы необходимо настроить регистры управления UCSRA, UCSRB и UCSRC [6].

UCSRA. Биты регистра UCSRA показаны в таблице 3.

Таблица 3 – Биты регистра UCSRA

Номер	7	6	5	4	3	2	1	0
Название	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
Доступ	R	R/W	R	R	R	R	R	R

Во время работы системы используются только биты RXC и UDRE [6].

RXC – флаг завершения приема. установлен, когда USART-модуль принимает данные в буфер. Используется при чтении данных из MAX232. Если бит RXC установлен, то в регистре UDR есть данные, которые нужно из него считать.

UDRE – флаг опустошения регистра данных и готовности регистра данных UDRE. Устанавливается в 1, когда буфер передатчика пуст и готов принимать новые данные. Используется при выводе данных из МК.

UCSRB. Биты регистра UCSRB показаны в таблице 4.

Таблица 4 – биты регистра UCSRB

Номер	7	6	5	4	3	2	1	0
Название	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Доступ	R	R/W	R/W	R/W	R/W	R/W	R	R/W

Во время работы системы, для управления приемом и передачей информации, будут использоваться только биты TXEN и RXEN [6].

TXEN – разрешение передачи. Если разряд сбросится во время передачи, передача закончит свою работу до конца, и тогда передатчик выключится.

RXEN – при установке в 1 разрешается работа приемника. При сбросе разряда работа приемника запрещается, буфер сбрасывается.

UCSRC. Биты регистра UCSRC показаны в таблице 5.

Таблица 5 – Биты регистра UCSRC

Номер	7	6	5	4	3	2	1	0
Название	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Доступ	R	R/W	R/W	R/W	R/W	R/W	R	R/W

Во время работы системы будут использоваться только биты URSEL, UCSZ0 и UCSZ1 [6].

URSEL – регистры UCSRC и UBRRH находятся в одном адресном пространстве, и чтобы понять, куда записывать данные используется бит URSEL. При URSEL равным 1 в UCSRC, при 0 в UBRRH.

UCSZ0 и UCSZ1 – формат посылки. Каждый из них принимает значение 1 – для 8 битной посылки данных.

Скорость передачи определяется выражением 1:

$$BAUD = \frac{f_{osc}}{16(UBRR+1)} \quad (1)$$

где BAUD — скорость передачи (бод);

f_{osc} — тактовая частота микроконтроллера (Гц);

UBRR — содержимое регистров UBRR0H, UBRR0L контроллера скорости передачи.

Зададим скорость 9600 бод при $fosc$ = 8МГц. Получится:

$$UBRR = \frac{f_{osc}}{16*BAUD} - 1 = \frac{8*10^6}{16*9600} - 1 = 51_{10} \quad (2)$$

UBRRL будет принимать значение 51.

1.2.7 Генератор тактовых импульсов и сброс

Для работы МК необходим тактовый генератор.

Внешний резонатор подключается к микроконтроллеру для обеспечения стабильной и точной работы его тактового генератора. Вот несколько причин, почему это важно:

1) Точность частоты: внешние резонаторы могут обеспечить более высокую точность и стабильность частоты по сравнению с встроенными генераторами. Это особенно важно в приложениях, где требуется точное время или синхронизация;

2) Настройка частоты: внешние резонаторы позволяют выбрать нужную частоту тактирования для конкретного приложения. Это может быть полезно, если требуется изменить скорость работы микроконтроллера;

3) Устойчивость к температурным изменениям: Внешние резонаторы часто имеют лучшие характеристики стабильности частоты при изменении температуры, что делает их более надежными в различных условиях эксплуатации;

4) Снижение помех: внешние резонаторы могут помочь уменьшить шум и помехи, которые могут влиять на работу микроконтроллера, что особенно важно в чувствительных приложениях;

5) Долговечность: внешние резонаторы могут иметь более длительный срок службы по сравнению с встроенными генераторами, что может быть критично в некоторых приложениях;

Таким образом, использование внешнего резонатора может значительно улучшить производительность и надежность системы на основе микроконтроллера. Подключим к системе внешний кварцевый резонатор с частотой 8 МГц – этого хватит для стабильной работы устройства.

В МК AVR, таких как ATmega, вывод $\overline{RESET}$ используется для инициализации устройства и его сброса. Когда питание включается, конденсатор разряжен, и $\overline{RESET}$ находится на уровне близком к нулю, что приводит к сбросу микроконтроллера. Затем конденсатор начинает заряжаться через резистор, и напряжение на $\overline{RESET}$ постепенно возрастает. Когда оно достигает порогового значения, микроконтроллер выходит из состояния сброса и начинает выполнение программы.

Время зарядки конденсатора определяет задержку перед запуском микроконтроллера, что позволяет обеспечить стабильность питания и минимизировать вероятность случайных сбросов.

Использование $\overline{RESET}$ также позволяет реализовать ручной сброс микроконтроллера, если на вывод $\overline{RESET}$ подать низкий уровень с помощью кнопки. Это может быть полезно для перезапуска программы или для инициализации устройства в определенном состоянии.

Подключим этот выход к питанию через резистор номиналом 10 кОМ и параллельно подключим конденсатор емкостью 100мкФ, а к нему землю.

1.2.8 Построение функциональной схемы

С учетом всех ранее отмеченных особенностей была спроектирована функциональная схема контроллера устройства для измерения скорости чтения, представленная на рисунке 6 и в приложении Б [7, 8].

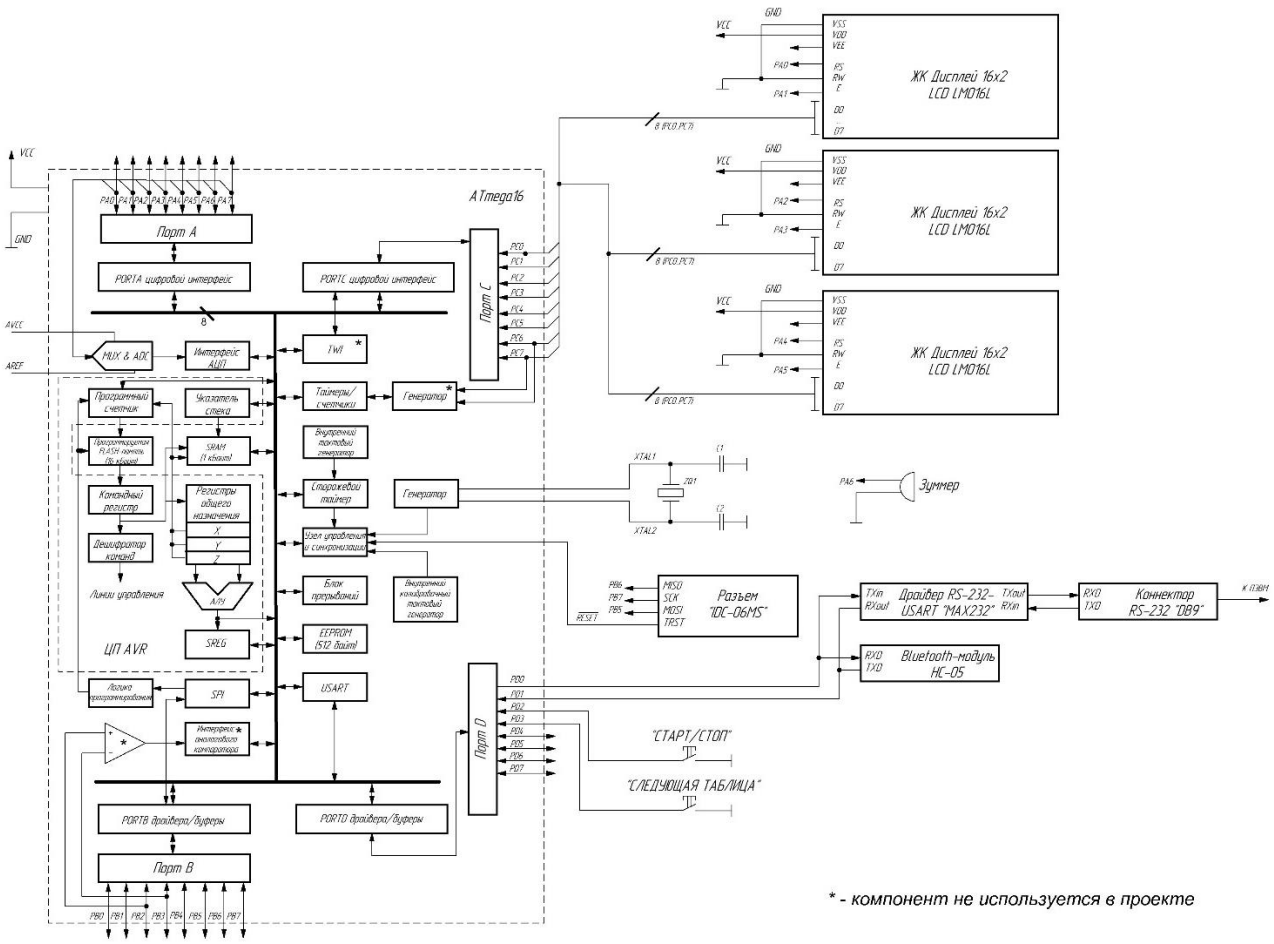


Рисунок 6 – Функциональная схема

1.3 Проектирование принципиальной схемы

1.3.1 Выбор устройства отображения данных

Устройство отображения данных должно предоставить пользователю таблицу Шульте и статистику замеров его скорости чтения, а именно:

– номер теста скорости чтения (от 1 до 8);

– время прочтения таблицы для каждого теста;

– среднее время прочтения таблицы.

В качестве такого устройства был выбран жидкокристаллический дисплей LCD LM016L, характеристики которого представлены в таблице 6 [9].

Таблица 6 – Характеристики ЖК-дисплея LM016L

Параметр	Значение
Напряжение питания	4.5 - 5.5 В
Ток потребления	15 - 25 мА
Потребляемая мощность	75 - 137.5 мВт
Входное сопротивление	10 кОм
Входной ток	1 мА
Напряжение логического уровня	2.4 - 5.5 В
Частота тактового сигнала	1 - 4 МГц
Время доступа	100 - 200 нс

Для отображения информации на данном LCD отведено лишь 2 строки по 16 символов. Управление LCD-дисплеем LM016L осуществляется с помощью нескольких сигналов, которые обеспечивают передачу данных и команд. Вот краткое описание основных сигналов управления:

– RS (Register Select) - определяет, будет ли передаваемая информация интерпретироваться как команда (RS = 0) или как данные (RS = 1).

– RW (Read/Write) - указывает направление передачи данных. Если RW = 0, данные записываются в дисплей (операция записи). Если RW = 1, данные считываются из дисплея (операция чтения).

– E (Enable) - активирует передачу данных. При переходе сигнала E из низкого в высокий уровень (положительный фронт) данные, находящиеся на шинах данных, будут считаны или записаны.

– Данные (D0-D7) - 8-битная шина данных, по которой передаются команды и данные. Дисплей может работать в 4-битном или 8-битном режиме. В 4-битном режиме используются только D4-D7.

Для того, чтобы поместить таблицу Шульте 5x5 на ЖК-дисплеи, необходимо применить 3 ЖК-дисплея (для размещения 5 строк). При выводе восьми результатов тестирования и среднего времени прочтения таблиц используется следующее распределение информации по строкам ЖК-дисплеев:

– первая строка первого ЖК-дисплея используется для вывода заголовков граф таблицы тестов (номер и время теста);

– вторая строка первого ЖК-дисплея, первая и вторая строки второго ЖК-дисплея, а также первая строка третьего ЖК-дисплея используются для вывода номера теста и времени прочтения таблицы Шульте;

– вторая строка третьего ЖК-дисплея используется для вывода среднего времени прочтения таблицы Шульте.

1.3.2 Разъем программатора

Для подключения программатора необходим специальный разъем IDC-06MS. Подключение осуществляется при помощи интерфейса SPI. Под него на МК ATmega16 задействованы последние 3 контакта порта PB, и контакт $\overline{RESET}$ микроконтроллера. На принципиальной схеме, которая показана в разделе 1.3.5, условное обозначение – XP1.

Он имеет следующие разъемы для подключения к МК [10]:

– MISO (Master Input Slave Output) – для передачи данных от микроконтроллера в программатор;

– SCK – тактовый сигнал;

– MOSI (Master Output Slave Input) – для передачи данных от программатора в микроконтроллер;

– $\overline{RESET}$ – вводит контроллер в режим программирования.

1.3.3 Подключение цепи питания

Разрабатываемой микроконтроллерной системе необходимо питание. В данном проекте потребуется 1 ключ питания на 5В. Питание будем подавать через разъем DS-201, условно обозначенный на схеме как XP2.

1.3.4 Расчет потребляемой мощности

Потребляемая мощность – это мощность, потребляемая интегральной схемой, которая работает в заданном режиме соответствующего источника питания.

Чтобы рассчитать суммарную мощность, рассчитаем мощность каждого элемента. На все микросхемы подается напряжение +5В. Мощность, потребляемая один устройством, в статическом режиме, рассчитывается формулой:

$$P = U * I \quad (3)$$

где U – напряжение питания (В);

I – ток потребления микросхемы (мА).

Также в схеме присутствуют резисторы. Одно из предназначений резисторов на схеме – уменьшить значение тока, подаваемого на устройство. В устройстве для измерения скорости чтения установлены подтягивающие резисторы у кнопок.

Для подтягивающих резисторов используем формулу (11):

$$P_{R_i} = \frac{U^2}{R_i} \quad (4)$$

$$P = \frac{5^2\ B}{10*10^3\ \text{Ом}} = 2{,}5\ \text{мВт} \quad (5)$$

Таких резисторов на схеме размещено 3, поэтому их суммарная потребляемая мощность будет равна 7,5 мВт.

Расчет потребляемого напряжения для каждого элемента проекта показан в таблице 7.

Таблица 7 – Потребляемая элементами мощность

Микросхема	Ток потребления, мА	Потребляемая мощность, мВт	Количество устройств	Суммарная потребляемая мощность, мВт
ATmega16	25	125	1	125
MAX232	8	40	1	40
LM016L	25	125	3	375
HC-05	50	250	1	250
TMB12A05	30	150	1	150

$$P_\Sigma = P_{ATmega16} + P_{MAX232} + 3 * P_{LM016L} + P_{HC-05} + P_{\text{TMB12A05}} + P_R =$$

$$= 125 + 40 + 375 + 250 + 150 + 7{,}5 = 947{,}5\ \text{мВт} \quad (6)$$

Суммарная потребляемая мощность системы равна 947,5 мВт.

1.3.5 Построение принципиальной схемы

На основе всех вышеописанных сведений была спроектирована принципиальная схема разрабатываемой системы, показанная на рисунке 7 и в приложении Б [7, 8].

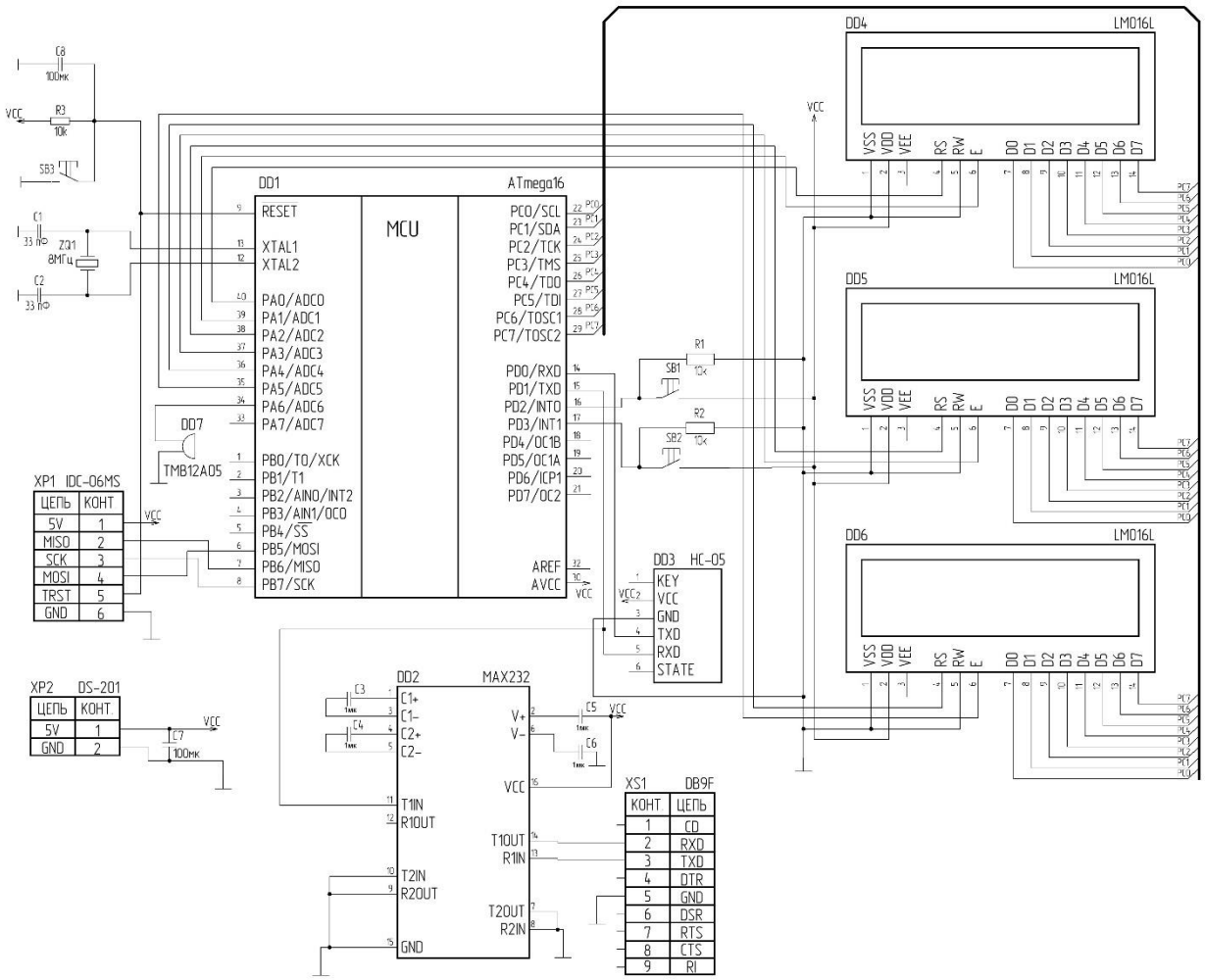


Рисунок 7 – Принципиальная схема

1.4 Алгоритмы работы системы

1.4.1 Функция Main

Для устройства измерения скорости чтения была разработана программа, схема алгоритма которой представлена на рисунке 8.

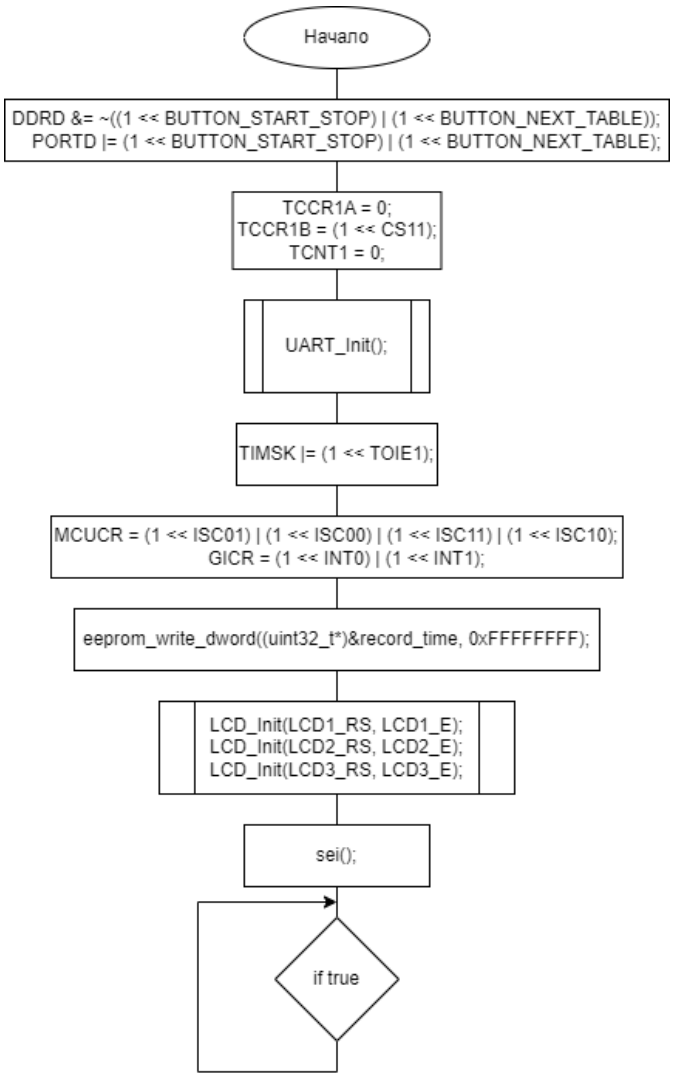


Рисунок 8 – Схема алгоритма программы

Работа начинается с функции main, в которой производится первичная настройка портов и из которой вызываются функции инициализации UART и ЖК-дисплеев. Пины BUTTON_START_STOP (PD2) (запуск/остановка таймера) и BUTTON_NEXT_TABLE (PD3) (генерация и отображение таблицы Шульте) настраиваются как входы. Таймер 1 настроен на работу в режиме нормального счёта, предделитель установлен на 8 (TCCR1B = (1 << CS11)), что задаёт шаг таймера (таймер будет увеличиваться каждую микросекунду (формула 7)).

$$F_{timer} = \frac{F_{cpu}}{prescaler} = \frac{8\ \text{МГц}}{8} = 1\ \text{МГц} \quad (7)$$

Прерывание по переполнению таймера включено (TIMSK |= (1 << TOIE1)). Прерывания INT0 и INT1 настраиваются для реагирования на нажатия на кнопки.

1.4.2 Используемые при работе подпрограммы

В разработанной программе используется несколько процедур и функций, которые решают определенные задачи. Рассмотрим их подробнее.

Процедурой LCD_Init() настраиваются ЖК-дисплеи. Эта процедура:

1) задаёт порты для управления дисплеями как выходы,

2) посылает команды инициализации (установка 8-битного режима, включение дисплея, очистка экрана).

В программе предусмотрено три независимых дисплея, которые управляются через отдельные управляющие линии RS, E и общий порт данных PORTC. Для каждого ЖК-дисплея вызывается LCD_Init().

LCD_Command() – процедура, предназначенная для отправки команды на подключенный ЖК-дисплей. В ЖК-дисплеях команды используются для выполнения различных операций. В разрабатываемом устройстве LCD_Command() используется для:

1) очистки экрана,

2) установки курсора,

3) включение дисплея,

4) задания режима работы.

Процедура LCD_Data() предназначена для отправки данных на подключенный ЖК-дисплей. В отличие от команд, данные передаются для отображения на экране, например, для вывода символов, текста или чисел.

LCD_Clear() – отдельная процедура, при вызове которой на ЖК-дисплей посылается команда очистки.

LCD_Print() необходима для посимвольной передачи строк.

Процедура generate_schulte_table() создает и выписывает на три ЖК-дисплея случайную таблицу Шульте.

Процедуры ISR(INT1_vect) и ISR(INT0_vect) обрабатывают прерывания по нажатию на кнопки.

Процедура ISR(TIMER1_OVF_vect) обрабатывает прерывание по переполнению таймера 1. При выбранной настройке таймера время до переполнения составляет:

$$t_{\text{переполнения}} = \frac{2^{16}}{F_{timer}} = \frac{65536}{10^6} = 65536 \text{ мкс} \quad (8)$$

что недостаточно ни для одного замера времени.

Поэтому была добавлена процедура ISR(TIMER1_OVF_vect), которая считает количество переполнений таймера (overflow_count) для дальнейшего корректного расчета времени прочтения таблицы Шульте.

start_timer() запускает таймер, stop_timer() останавливает таймер и вызывает функции для вывода результата. Также процедура stop_timer() производит расчет времени прочтения таблицы Шульте по формуле 9.

elapsed_time = (overflow_count * 65536) + finish_time - start_time (9)

Данная формула корректно рассчитывает время прочтения таблицы при любых значениях finish_time и start_time. Переменная elapsed_time является 32-х разрядной, что задает лимит по времени для прочтения таблицы в 2^{32} - 1 мкс (4294967295 мкс), что равняется примерно 71,5 минуте, чего точно будет достаточно для замера времени чтения пользователя.

Функция calculate_average_time() возвращает среднее значение по последим восьми замерам времени.

Процедура out_times() выводит на ЖК-дисплеи все 8 временных замеров, а также среднее значение по этим замерам.

Функция custom_utoa() – это написанный аналог функции utoa(). Необходимость написания данной функции обусловлена тем, что функция utoa() явно преобразовывает входные параметры к знаковому типу, что сказывается на выводе результатов замеров времени при использовании данной функции (результаты могли быть отрицательными).

Процедура UART_Init() инициализирует модуль USART. UART_Transmit() отправляет один байт данных через USART. UART_SendString() вызывает процедуру UART_Transmit() для каждого символа передаваемой строки. send_test_times_via_bluetooth() вызывает UART_SendString() для всей необходимой для передачи информации (массив замеров времени прочтения таблицы, среднее время прочтения, оповещение о рекорде).

Ниже приведены схемы алгоритмов всех упомянутых выше подпрограмм (рисунки 9-15).

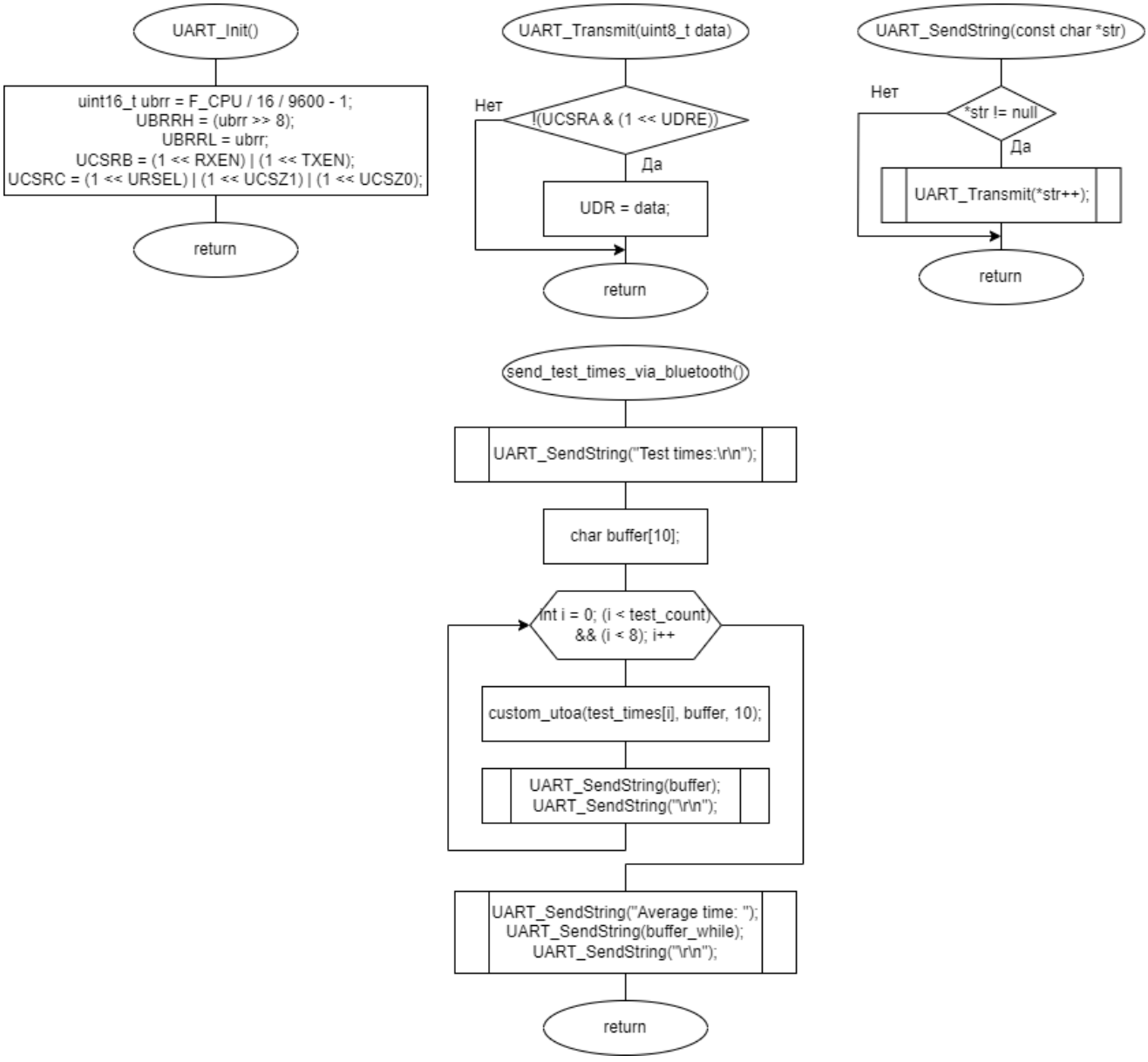


Рисунок 9 – Схемы алгоритмов функций работы с USART

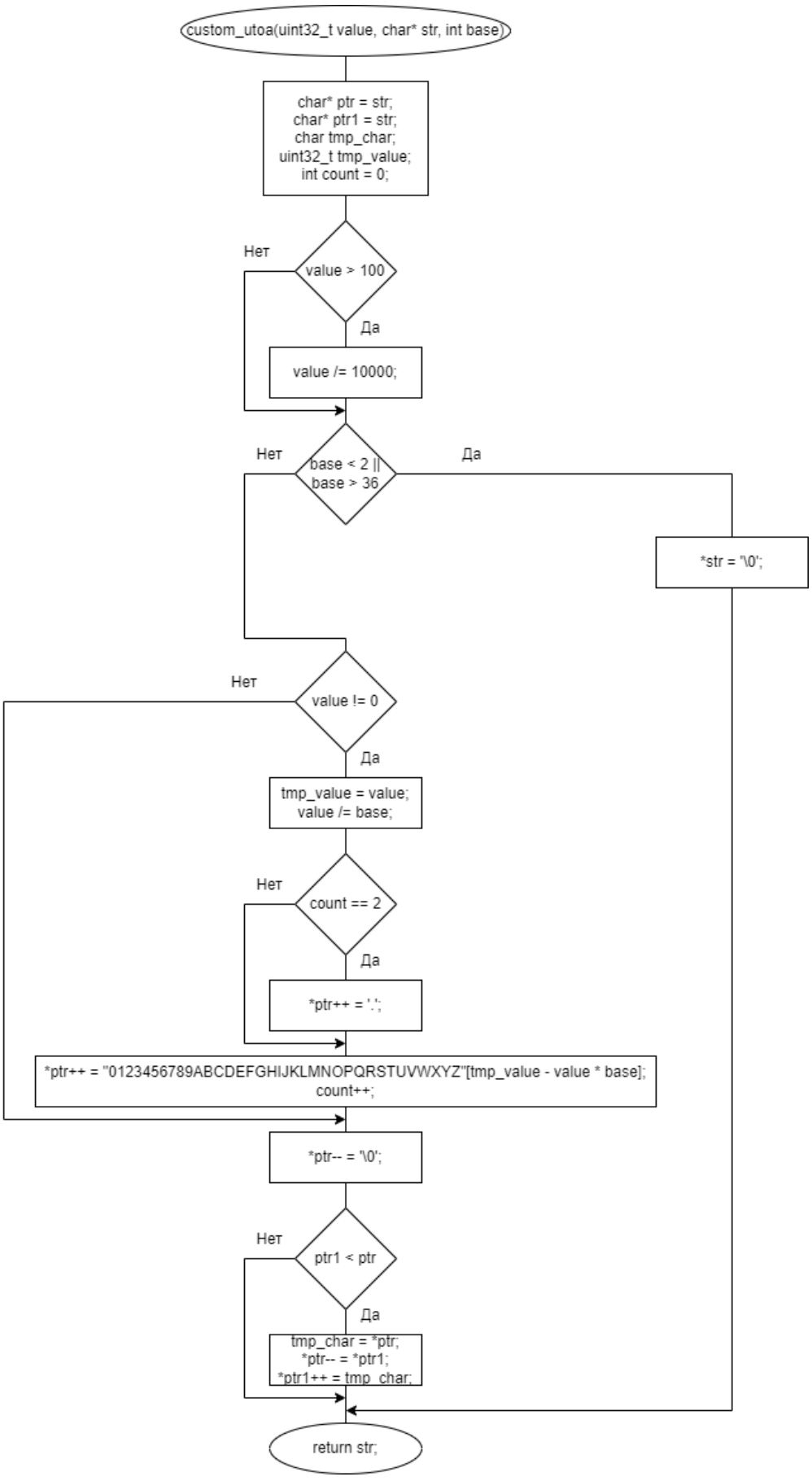


Рисунок 10 – Схема алгоритма custom_utoa()

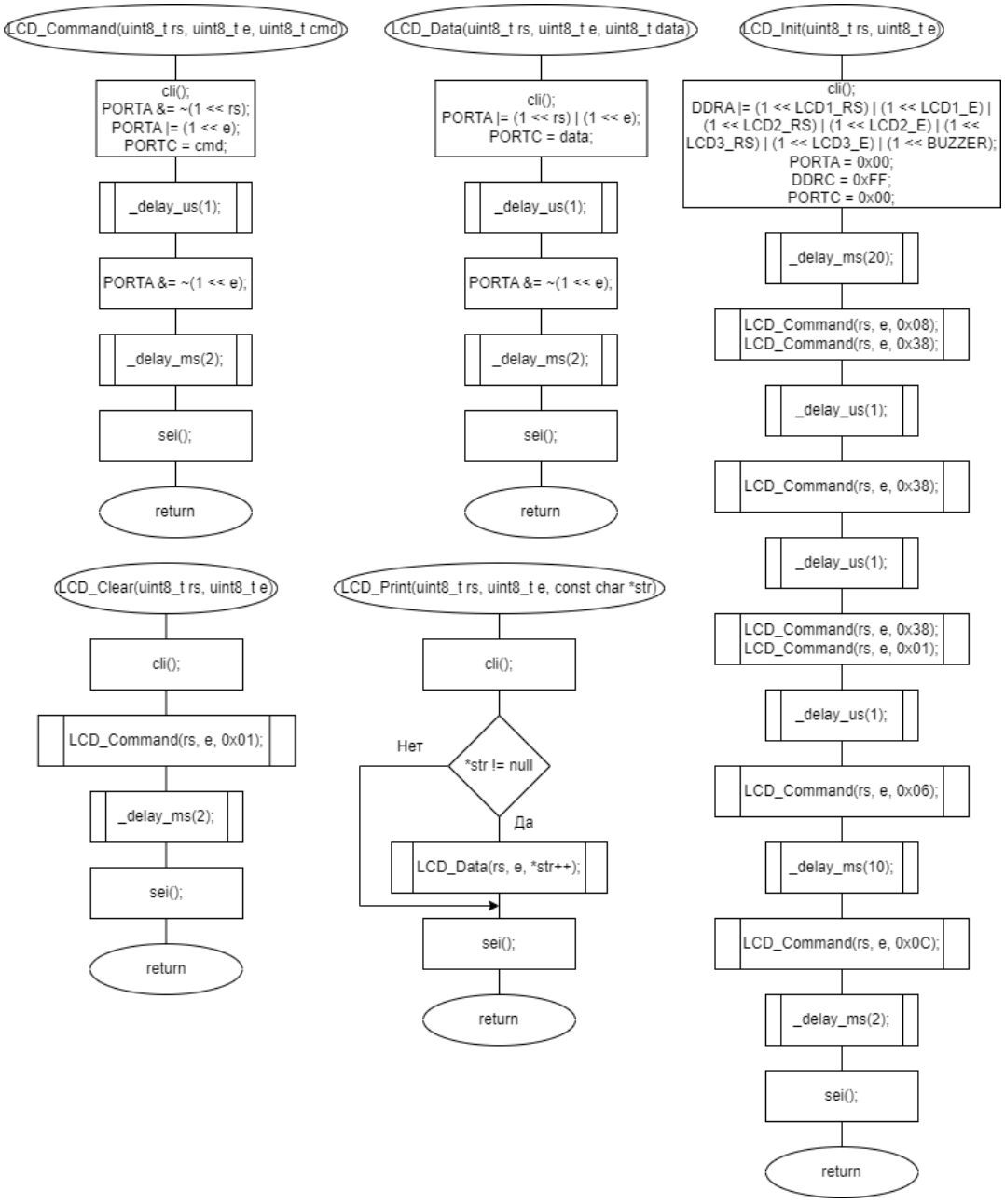


Рисунок 11 – Схемы алгоритмов функций работы с ЖК-дисплеем

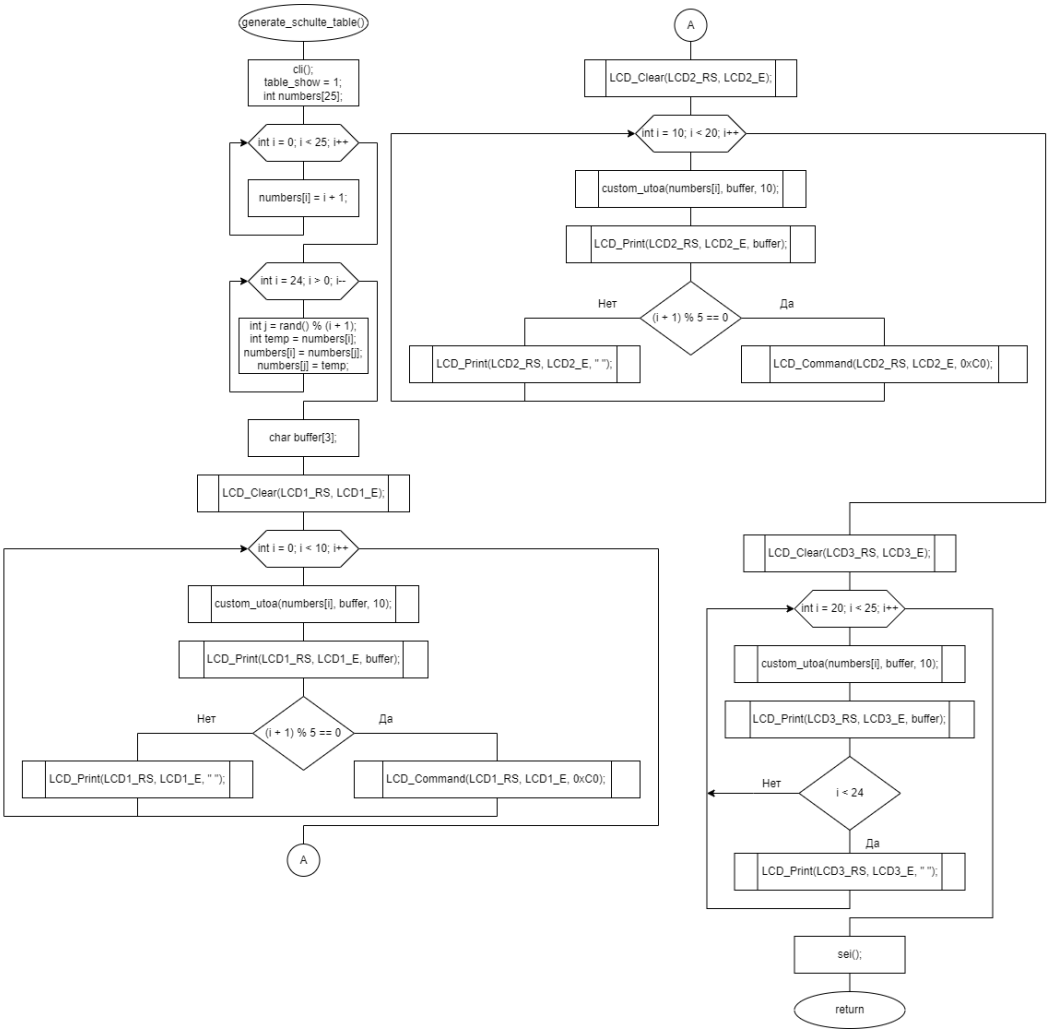


Рисунок 12 – Схема алгоритма generate_schulte_table()

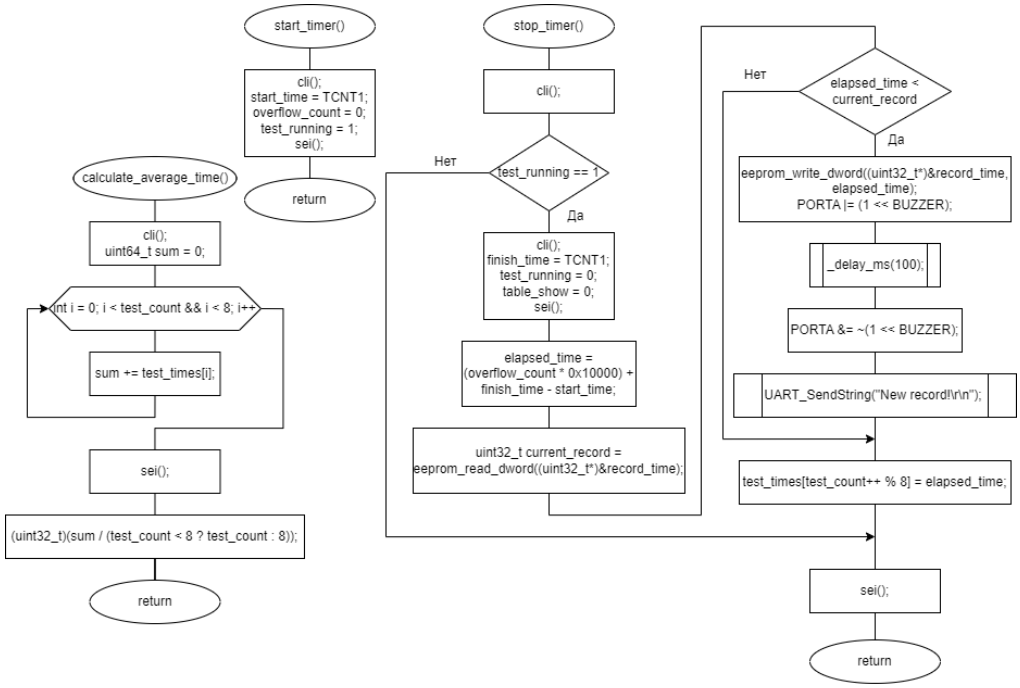


Рисунок 13 – Схема алгоритма процедур start_timer(), stop_timer(), calculate_average_time()

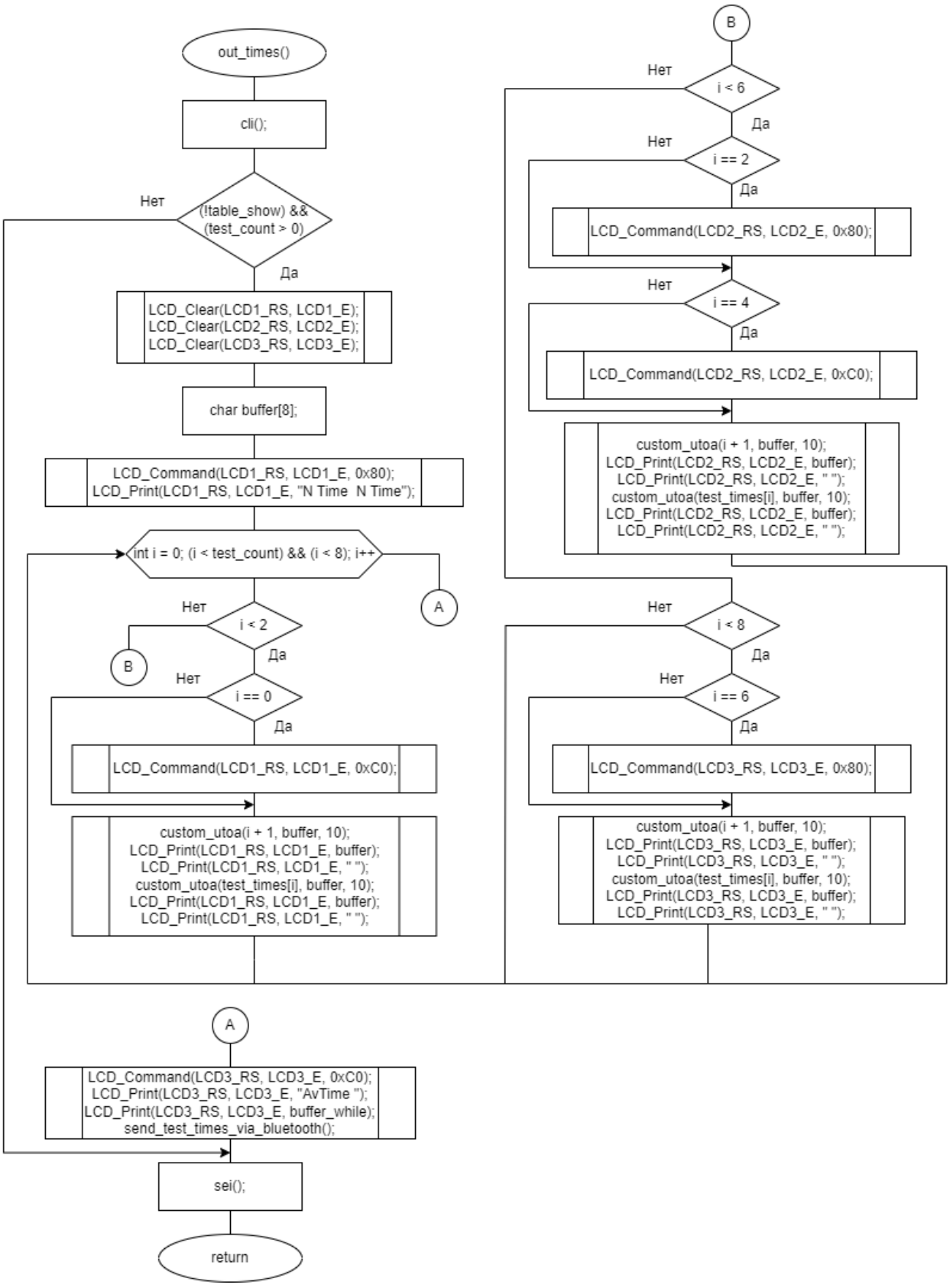


Рисунок 14 – Схема алгоритма out_times()

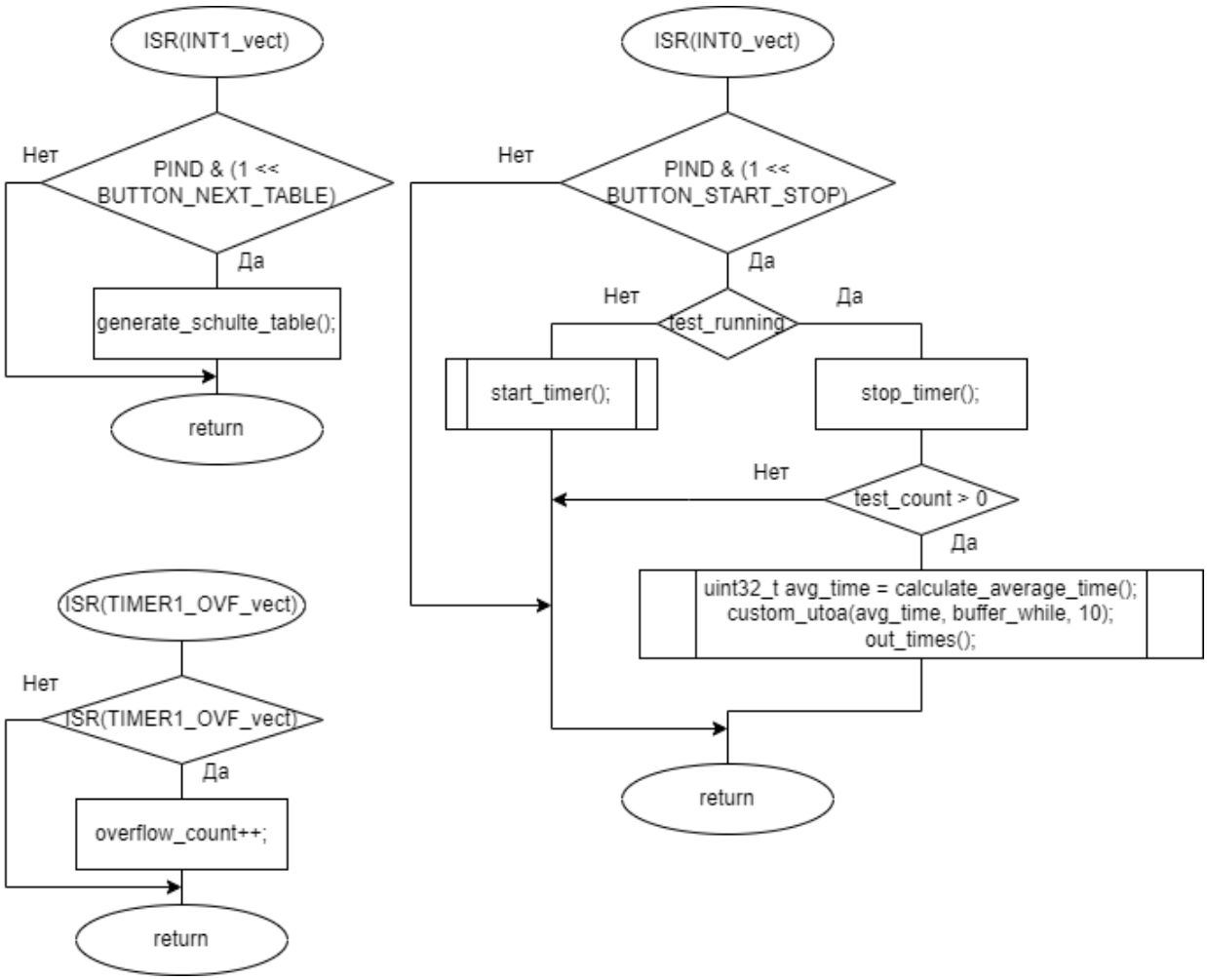


Рисунок 15 – Схема алгоритма процедур, отвечающих за прерывания

Ниже приведено краткое описание работы программы:

Программа предназначена для работы с тремя LCD-дисплеями, кнопками, таймером и сохранением результатов в EEPROM. Она генерирует таблицу Шульте, измеряет время выполнения теста, отображает результаты и отправляет их через Bluetooth. Этапы работы программы:

1) инициализация оборудования:

а) LCD-дисплеи (используются три LCD-дисплея, подключенных к разным пинам микроконтроллера, инициализируются в 8-битном режиме с отключением курсора);

б) USART (настраивается на скорость 9600 бод для передачи данных через Bluetooth);

в) кнопки (кнопка BUTTON_START_STOP (PD2) для запуска/остановки таймера, кнопка BUTTON_NEXT_TABLE (PD3) для генерации новой таблицы Шульте);

г) таймер (используется таймер 1 в нормальном режиме с предделителем 8, прерывание срабатывает при переполнении таймера);

д) EEPROM (в EEPROM записывается и считывается рекордное время выполнения теста).

2) работа с таблицей Шульте:

а) генерация (создается массив чисел от 1 до 25, перемешивается случайным образом, числа распределяются по трем дисплеям в формате таблицы 5x5);

б) отображение (числа отображаются на LCD-дисплеях по строкам, используется задержка для корректного отображения символов);

3) измерение времени:

а) запуск таймера (нажатие кнопки BUTTON_START_STOP запускает таймер, начальное значение считывается из регистра TCNT1, обнуляется счетчик переполнений overflow_count);

б) остановка таймера (повторное нажатие кнопки останавливает таймер, рассчитывается общее время выполнения с учетом количества переполнений таймера);

в) сохранение времени (если время меньше рекордного, оно сохраняется в EEPROM, добавляется в массив последних 8 измерений).

4) работа с результатами:

а) среднее время (вычисляется среднее время из последних 8 измерений);

б) отображение результатов (результаты отображаются на LCD-дисплеях, среднее время и результаты измерений передаются через Bluetooth);

в) рекордное время (если установлено новое рекордное время, включается зуммер).

5) Bluetooth-вывод (результаты измерений отправляются через USART в текстовом формате, передаются значения всех измерений и среднее время);

6) обработка прерываний:

а) прерывание кнопок (прерывания INT0 и INT1 обрабатывают нажатия кнопок, INT0 запускает или останавливает таймер, INT1 генерирует новую таблицу Шульте);

б) прерывание таймера (увеличивает счетчик overflow_count при переполнении таймера).

7) основной цикл программы (в основном цикле программа не выполняет активных действий, так как все операции управляются через прерывания).

На основе составленных схем алгоритмов и разработанных этапов работы программы был написан код, который показан в Приложении А (Текст программы).

2 Технологическая часть

Для реализации работы модели микроконтроллера устройства для измерения скорости чтения была написана программа на языке Си, после загруженная в МК. Симуляция проводилась в программе Proteus 8.

2.1 Отладка и тестирование программы

Программа была отлажена в среде разработки AVR Studio, макет модели был собран в приложении Proteus 8. Proteus 8 предназначен для выполнения различных видов моделирования аналоговых и цифровых устройств. В этом приложении была собрана модель устройства, основанная на принципиальной схеме, то есть содержащая МК ATmega16, три ЖК-дисплея, управляющие кнопки, Bluetooth-модуль.

Все эти средства помогают убедиться в правильности написанной программы и управления реальными физическими устройствами.

При разработке алгоритма и написании кода программы были использованы следующие библиотеки:

– avr/io.h – библиотека ввода/вывода для микроконтроллеров AVR, предоставляющая доступ к регистрам и функциональным модулям микроконтроллера, таким как порты ввода-вывода, таймеры, АЦП, и другие аппаратные ресурсы. Используется для низкоуровневого управления периферийными устройствами и настройкой пинов;

– avr/eeprom.h – библиотека для работы с EEPROM (энергонезависимой памятью) микроконтроллеров AVR. Предоставляет функции для чтения и записи данных в память EEPROM, которая сохраняет данные даже при выключении питания устройства. Используется в программах для хранения настроек, параметров или других данных, требующих долгосрочного сохранения;

– avr/interrupt.h – заголовочный файл библиотеки AVR, предоставляющий макросы и функции для работы с прерываниями. Позволяет настроить обработчики прерываний и управлять глобальным

включением/выключением прерываний в микроконтроллере. Используется для асинхронного выполнения операций, таких как обработка событий от кнопок, таймеров или внешних сигналов;

– util/delay.h – библиотека, предоставляющая функции для создания задержек (например, _delay_ms() или _delay_us()), реализованных через программные циклы. Используется для временного ожидания между операциями, например, при настройке ЖК-дисплея или синхронизации действий микроконтроллера;

– stdlib.h – стандартная библиотека языка программирования C, предоставляющая функции для управления памятью, преобразования данных, генерации случайных чисел и других операций общего назначения. В программе используется для выполнения таких операций, как преобразование чисел в строковый формат (например, itoa()), выделение памяти, сортировка массивов и других задач.

Так как в работе активно используется система прерываний, можно также заметить системную функцию ISR, которая обозначает прерывание таймера. Она вызывается с разными параметрами. Поясним каждый из них:

– INT0_vect – вектор внешнего прерывания по нажатию кнопки «старт/стоп» (изменение сигнала на входе INT0);

– INT1_vect – вектор внешнего прерывания по нажатию кнопки «следующая таблица» (изменение сигнала на входе INT1);

– TIMER1_OVF_vect – вектор прерывания таймера T1 в результате его переполнения [11].

В результате компиляции программы был создан файл с расширением “.hex” объемом 21 килобайт. Далее была проведена отладка разработанной программы в среде AVR Studio, проведено тестирование и симуляция работы модели беговой дорожки на виртуальном макете в программе Proteus 8 версии. Модель работает корректно и отвечает всем требованиям, заявленным в ТЗ.

2.2 Симуляция работы системы

Программа Proteus позволяет собрать модель системы и запустить ее в окружении, приближенном к реальным рабочим условиям. Схема проекта модели микроконтроллера беговой дорожки в Proteus показана на рисунке 16. Работа схемы представлена на рисунке 17.

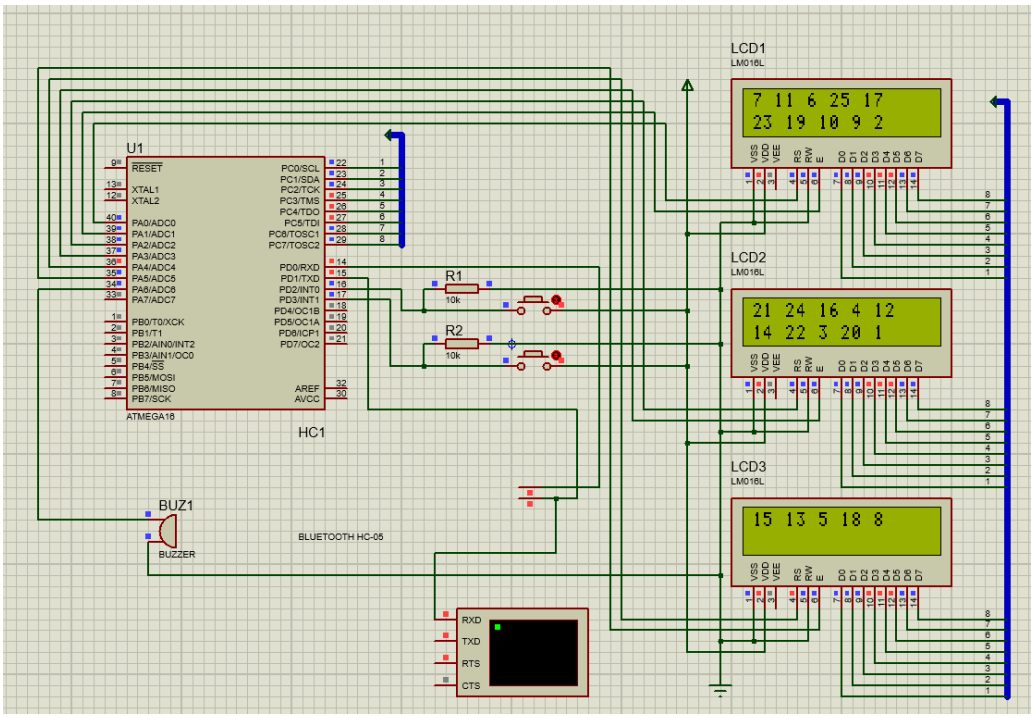


Рисунок 16 – Модель устройства для измерения скорости чтения

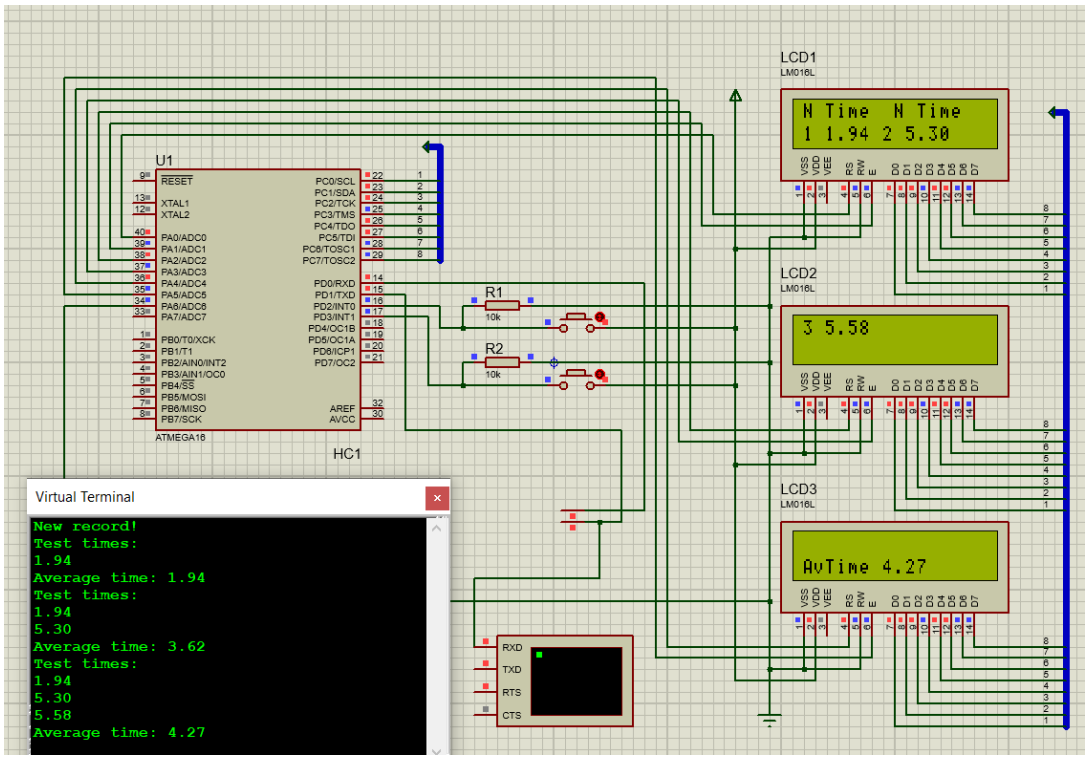


Рисунок 17 – Работа схемы

Для моделирования ввода данных с ПЭВМ используется инструмент системы – Virtual Terminal. С его помощью можно эмулировать терминал для взаимодействия с МК через интерфейс USART по портам TXD и RXD.

Запустив симуляцию и выбрав в верхнем меню Debug – Virtual Terminal, откроем виртуальный терминал. Для проверки работы канала передачи данных USART посчитаем теоретическое время прочтения таблицы со временем, выводимым на терминал (для реализации проверки значение переменной overflow_count выводится на порт B). На рисунке 18 представлен начальный момент отсчета времени прочтения таблицы, а на рисунке 19 конечный момент.

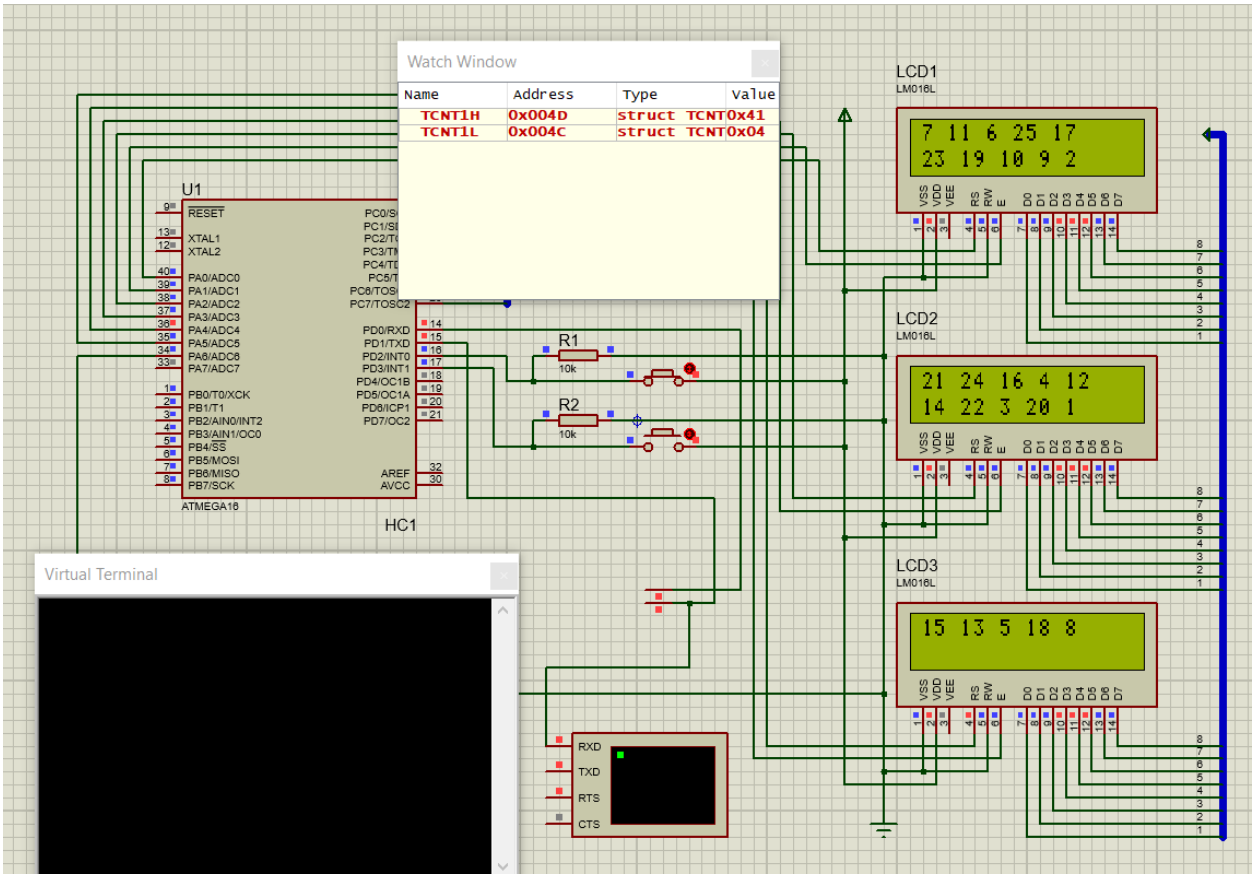


Рисунок 18 – Начальный момент отсчета времени прочтения таблицы

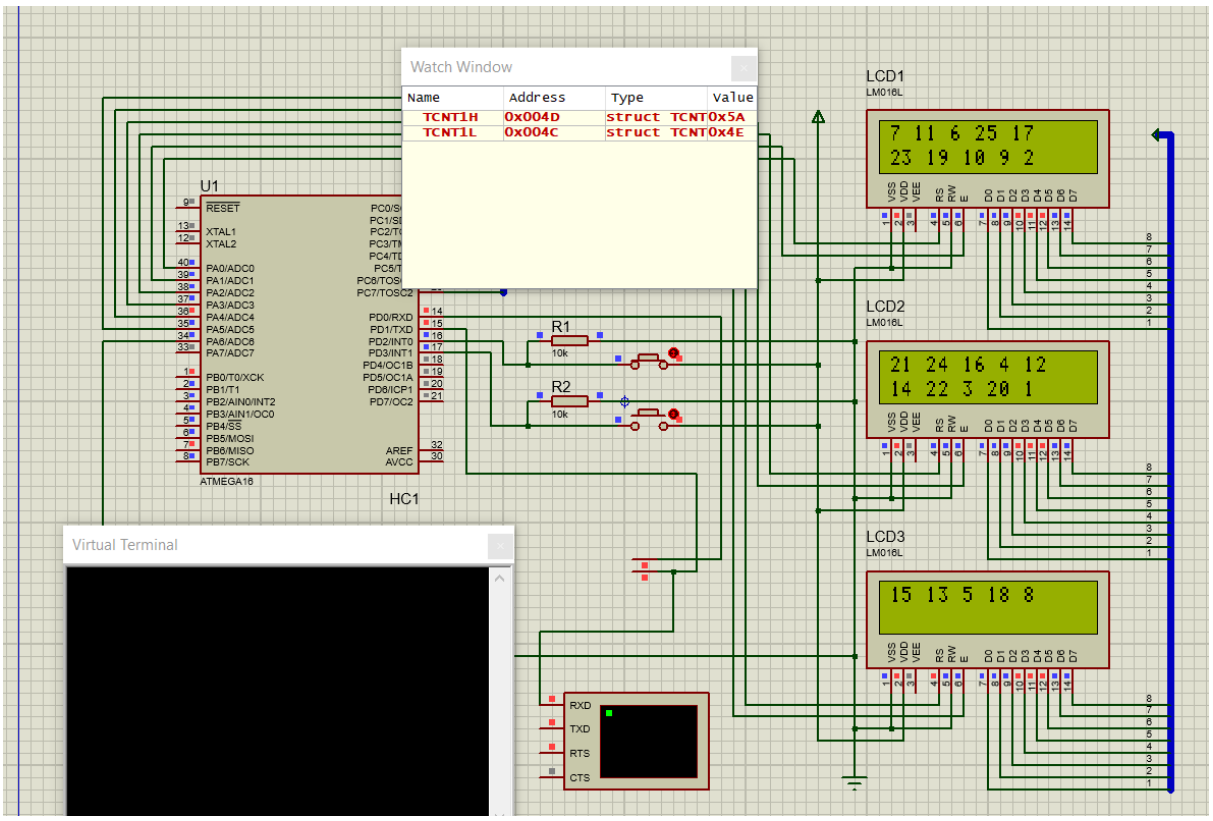


Рисунок 19 – Конечный момент отсчета времени прочтения таблицы

Результат представлен на рисунке 20.

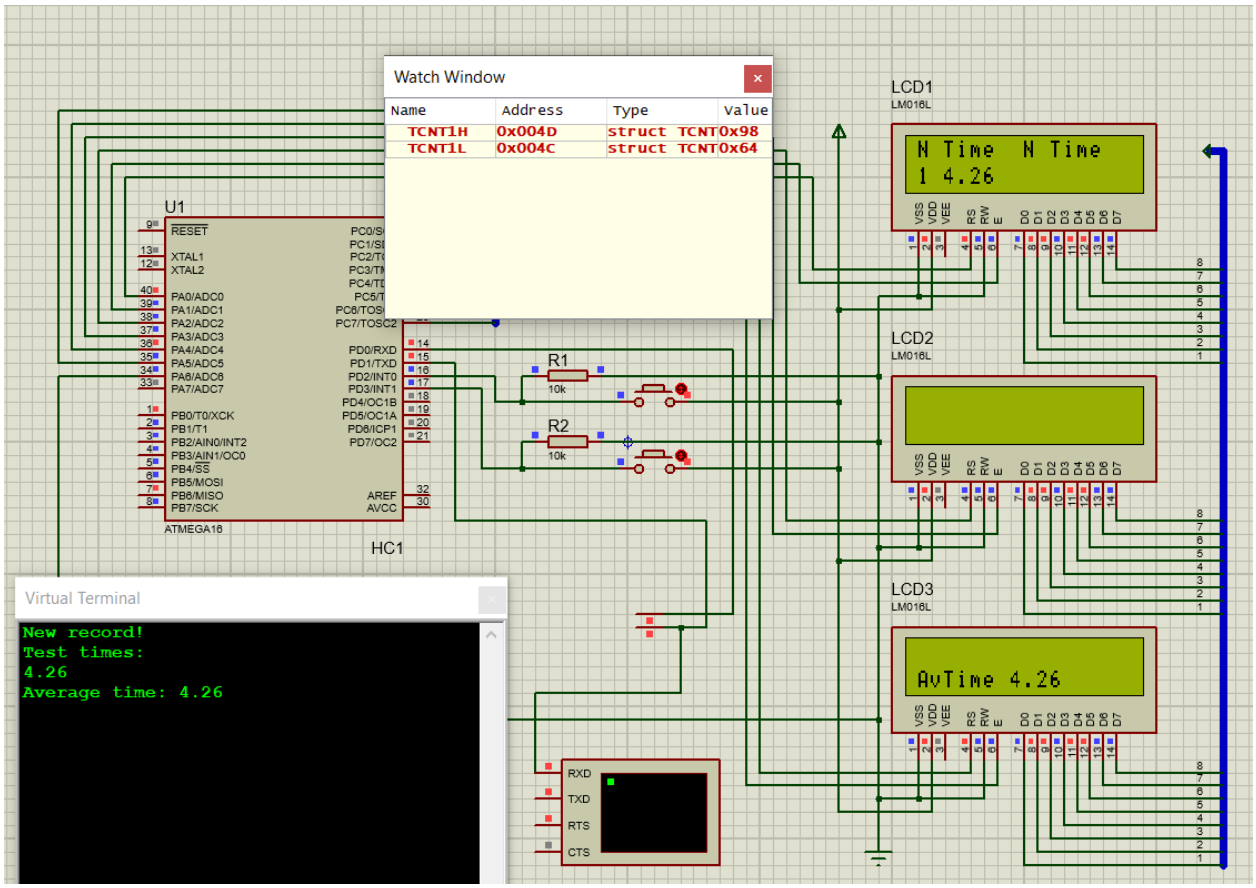


Рисунок 20 – Результат проверки USART

Рассчитаем время прочтения таблицы по полученным данным и сравним его со временем, переданным по USART.

По формуле 9:

elapsed_time = (overflow_count * 65536) + finish_time - start_time

Из полученных данных видно, что overflow_count = $01000001_2 = 65_{10}$, finish_time = $9864_{16} = 39012_{10}$ (что при F_{timer} = 1 МГц соответствует микросекундам), а start_time = $5A4E_{16} = 23118_{10}$ мкс.

Рассчитаем elapsed_time:

elapsed_time = (65 * 65536) + 39012 – 23118 = 4275734 мкс

Так как ответ в программе округляется и выводится в секундах, видно, что значение 4275734 мкс (4,27 с) соответствует выведенному значению на терминал (4,26 с).

2.3 Способы программирования МК

После реализации алгоритма программы в виде кода и его отладки получаем файл с расширением .hex. Это бинарный файл с программой, который можно загрузить в микроконтроллер несколькими способами и использовать [10]:

– внутрисхемное программирование (ISP);

– параллельное высоковольтное программирование;

– через JTAG;

– через Bootloader;

– Pinboard II.

Выполним прошивку через канал SPI. Для записи программы в память микроконтроллера требуется программатор и одноименный разъем. Схема подключения представлена на рисунке 21.

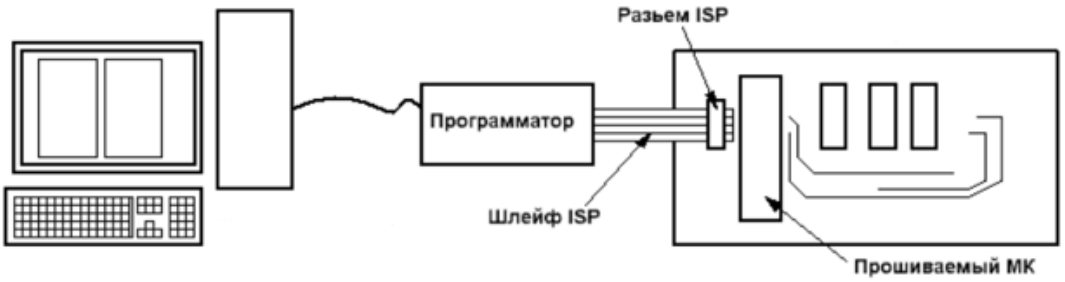


Рисунок 21 – Программирование МК

Прошивка проходит по интерфейсу SPI, для работы программатора нужно 4 контакта и питание (достаточно только земли, чтобы уравнять потенциалы земель программатора и устройства):

– MISO – Master-Input/Slave-Output – данные, идущие от контроллера;

– MOSI – Master-Output/Slave-Input – данные идущие в контроллер;

– SCK – тактовые импульсы интерфейса SPI;

– RESET – сигналом на RESET программатор вводит контроллер в режим программирования;

– GND – земля;

– VCC – питание.

Взаимодействие устройств по интерфейсу SPI требует установки одного из устройств в режим ведущего, а остальных – в режим ведомого. При этом ведущее устройство отвечает за выбор ведомого и инициализацию передачи. Передача по SPI осуществляется в полнодуплексном режиме (в обе стороны), по одному биту за такт в каждую сторону. По возрастающему фронту сигнала SCK ведомое устройство считывает очередной бит с линии MOSI, а по спадающему – выдает следующий бит на линию MISO.

Таким образом в МК передается бинарный файл скомпилированной ранее программы.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана модель устройства для измерения скорости чтения, основу которой составляет МК ATmega16 из серии Mega семейства AVR. Устройство разработано в соответствии с ТЗ.

В процессе работы над курсовой работой были разработаны электрическая, функциональная и принципиальная схемы, перечень элементов и документация к устройству. Исходный код программы был разработан в среде AVR Studio на языке C, отлажен и протестирован при помощи симулятора Proteus 8.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих: учебное пособие. – 2-е изд., испр. и доп. – М.: Издательство: МГТУ им. Н.Э. Баумана, 2012. – 280с.
2. Микроконтроллеры фирмы "ATMEL" семейства AVR [Электронный ресурс] – URL: https://www.promelec.ru/articles/73/ (дата обращения: 10.10.2024)
3. Документация к МК ATmega16 [Электронный ресурс] – URL: http:// www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr/atmega16.htm (дата обращения: 11.10.2024)
4. Даташит ATmega16 [Электронный ресурс] – URL: https:// www.alldatasheet.com/datasheet-pdf/view/78532/ATMEL/ATMEGA16.html (дата обращения: 13.10.2024)
5. MAX232 даташит [Электронный ресурс] – URL: https:// www.alldatasheet.com/datasheet-pdf/pdf/73047/MAXIM/MAX232.html (дата обращения: 27.10.2024)
6. Использование интерфейса USART микроконтроллеров AVR - Микроконтроллеры и Технологии [Электронный ресурс] – URL: https://radioparty.ru/prog-avr/program-c/307-lesson-usart-avr (дата обращения: 15.11.2024)
7. ГОСТ 2.743-91 ЕСКД ОБОЗНАЧЕНИЯ БУКВЕННО-ЦИФРОВЫЕ В ЭЛЕКТРИЧЕСКИХ СХЕМАХ [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200001985 (дата обращения: 16.11.2024)
8. ГОСТ 2.721-74 ЕСКД ОБОЗНАЧЕНИЯ УСЛОВНЫЕ ГРАФИЧЕСКИЕ В СХЕМАХ [Электронный ресурс]. – Режим доступа: https:// docs.cntd.ru/document/1200007058 (дата обращения: 16.11.2024)
9. LM016L Datasheet [Электронный ресурс] – URL: https:// www.alldatasheet.com/datasheet-pdf/pdf/146552/HITACHI/LM016L.html (дата обращения: 18.11.2024)

10. Трактат о программаторах [Электронный ресурс] – URL: https://easyelectronics.ru/avr-uchebnyj-kurs-traktat-o-programmatorax.html (дата обращения: 18.11.2024)
11. AVR. Учебный курс. Таймеры. [Электронный ресурс] – URL: https://easyelectronics.ru/avr-uchebnyj-kurs-tajmery.html?ysclid=m4658l907r176153014 (дата обращения: 26.10.2024)

Приложение А

Текст программы

Объем исполняемого кода – 486 строк.

```
#include <avr/io.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL  // Частота 8 МГц
#include <util/delay.h>
#include <stdlib.h>


// Определения пинов для LCD
#define LCD1_RS PA0
#define LCD1_E PA1
#define LCD1_DATA PORTC

#define LCD2_RS PA2
#define LCD2_E PA3
#define LCD2_DATA PORTC

#define LCD3_RS PA4
#define LCD3_E PA5
#define LCD3_DATA PORTC

#define BUTTON_START_STOP PD2 // Кнопка запуска/остановки
#define BUTTON_NEXT_TABLE PD3 // Кнопка переключения таблицы
#define BUZZER PA6              // Пин для управления зуммером

// Переменные для измерения времени
volatile uint16_t start_time = 0;  // Начальное время
volatile uint32_t elapsed_time;    // Общее время
volatile uint16_t finish_time = 0; // Конечное время
volatile uint16_t overflow_count = 0;   // Счетчик переполнений
таймера
volatile uint8_t test_running = 0; // Флаг состояния теста
volatile uint8_t table_show = 0;   // Флаг показа таблицы
uint32_t record_time EEMEM = 0xFFFFFFFF; // EEPROM для рекорда
volatile uint32_t test_times[8];   // Хранение результатов
тестов
volatile uint8_t test_count = 0;   // Количество тестов
char buffer_while[8] = "";    // Буфер для строковых операций

// Функция преобразования числа в строку
char* custom_utoa(uint32_t value, char* str, int base) {
    char* ptr = str;         // Указатель на текущую позицию в
строке
    char* ptr1 = str;        // Указатель для разворота строки
    char tmp_char;           // Временная переменная для обмена
символов
```

```
    uint32_t tmp_value;      // Временная переменная для
хранения значения
     int count = 0;
     if (value > 100) value /= 10000;

    if (base < 2 || base > 36) {
        *str = '\0';
        return str;
    }

    do {
        tmp_value = value;
        value /= base;
          if (count == 2) *ptr++ = '.';
        *ptr++ =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"[tmp_value - value *
base];
          count++;
    } while (value);

    *ptr-- = '\0';

    // Разворот строки
    while (ptr1 < ptr) {
        tmp_char = *ptr;
        *ptr-- = *ptr1;
        *ptr1++ = tmp_char;
    }
    return str;
}


// Настройка UART на скорости 9600 бод (частота 8 МГц)
void UART_Init() {
    uint16_t ubrr = F_CPU / 16 / 9600 - 1;
    UBRRH = (ubrr >> 8);
    UBRRL = ubrr;
    UCSRB = (1 << RXEN) | (1 << TXEN); // прием и передача
    UCSRC = (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0); //
Формат: 8 бит данных, 1 стоп-бит
}

// Процедура передачи одного байта
void UART_Transmit(uint8_t data) {
    while (!(UCSRA & (1 << UDRE))); // Ожидание освобождения
регистра данных
    UDR = data;
}

// Процедура для передачи строки
void UART_SendString(const char *str) {
    while (*str) {
        UART_Transmit(*str++);
```

```
    }
}

// Отправка результатов тестов через UART
void send_test_times_via_bluetooth() {
    UART_SendString("Test times:\r\n");  // \r\n для новой
строки
    char buffer[10];
    for (int i = 0; (i < test_count) && (i < 8); i++) {
        custom_utoa(test_times[i], buffer, 10);
        UART_SendString(buffer);
        UART_SendString("\r\n");  // \r\n после каждого
показателя
    }

    UART_SendString("Average time: ");
    UART_SendString(buffer_while);
    UART_SendString("\r\n");
}


// Процедура для отправки команды на LCD
void LCD_Command(uint8_t rs, uint8_t e, uint8_t cmd) {
     cli();
     PORTA &= ~(1 << rs);
    PORTA |= (1 << e);  // импульс на E
    PORTC = cmd;        // команду на вывод данных
    _delay_us(1);
    PORTA &= ~(1 << e);
    _delay_ms(2);       // Задержка для выполнения команды
     sei();
}

// Процедура для отправки данных на LCD
void LCD_Data(uint8_t rs, uint8_t e, uint8_t data) {
     cli();
    PORTA |= (1 << rs) | (1 << e);     // RS = 1 (режим данных)
     PORTC = data;           // установка данных на вывод
    _delay_us(1);
    PORTA &= ~(1 << e);
    _delay_ms(2);           // Задержка для выполнения команды
     sei();
}

// Процедура для инициализации LCD
void LCD_Init(uint8_t rs, uint8_t e) {
     cli();
     DDRA |= (1 << LCD1_RS) | (1 << LCD1_E) | (1 << LCD2_RS) |
(1 << LCD2_E) | (1 << LCD3_RS) | (1 << LCD3_E) | (1 << BUZZER);
     PORTA = 0x00;
    DDRC = 0xFF;        // Порт C как выход (данные)
     PORTC = 0x00;
```

```
    _delay_ms(20);       // Задержка после подачи питания

    // Инициализация в 8-битном режиме
     LCD_Command(rs, e, 0x08);
     LCD_Command(rs, e, 0x38);
     _delay_us(1);
     LCD_Command(rs, e, 0x38);
     _delay_us(1);
     LCD_Command(rs, e, 0x38);
     LCD_Command(rs, e, 0x01);
     _delay_us(1);
     LCD_Command(rs, e, 0x06);
     _delay_ms(10);
    LCD_Command(rs, e, 0x0C); // Включение дисплея и отключение
курсора
    _delay_ms(2);
     sei();
}

// Процедура для очистки экрана
void LCD_Clear(uint8_t rs, uint8_t e) {
     cli();
    LCD_Command(rs, e, 0x01); // Очистка дисплея
    _delay_ms(2);         // Задержка для очистки
     sei();
}

// Процедура для печати строки на LCD
void LCD_Print(uint8_t rs, uint8_t e, const char *str) {
     cli();
    while (*str) {
        LCD_Data(rs, e, *str++); // Отправка каждого символа на
LCD
    }
     sei();
}

// Генерация таблицы Шульте
void generate_schulte_table() {
     cli();
     table_show = 1;
    int numbers[25];
    for (int i = 0; i < 25; i++) numbers[i] = i + 1;
    for (int i = 24; i > 0; i--) {
        int j = rand() % (i + 1);
        int temp = numbers[i];
        numbers[i] = numbers[j];
        numbers[j] = temp;
    }

    // Отправка на LCD дисплеи
    char buffer[3];
```

```
    // Первый LCD
    LCD_Clear(LCD1_RS, LCD1_E);
    for (int i = 0; i < 10; i++) {
        custom_utoa(numbers[i], buffer, 10);
        LCD_Print(LCD1_RS, LCD1_E, buffer);
        if ((i + 1) % 5 == 0) LCD_Command(LCD1_RS, LCD1_E,
0xC0);
        else LCD_Print(LCD1_RS, LCD1_E, " ");
    }

    // Второй LCD
    LCD_Clear(LCD2_RS, LCD2_E);
    for (int i = 10; i < 20; i++) {
        custom_utoa(numbers[i], buffer, 10);
        LCD_Print(LCD2_RS, LCD2_E, buffer);
        if ((i + 1) % 5 == 0) LCD_Command(LCD2_RS, LCD2_E,
0xC0);
        else LCD_Print(LCD2_RS, LCD2_E, " ");
    }

    // Третий LCD
    LCD_Clear(LCD3_RS, LCD3_E);
    for (int i = 20; i < 25; i++) {
        custom_utoa(numbers[i], buffer, 10);
        LCD_Print(LCD3_RS, LCD3_E, buffer);
        if (i < 24) LCD_Print(LCD3_RS, LCD3_E, " ");
    }
     sei();
}

// Процедура запуска таймера
void start_timer() {
     cli();
    start_time = TCNT1;
     overflow_count = 0;
    test_running = 1;
     sei();
}

// Процедура остановки таймера
void stop_timer() {
     cli();
    if (test_running) {

          cli();
          finish_time = TCNT1;
        test_running = 0;
          table_show = 0;
          sei();

        // Рассчет времени прочтения таблицы
        elapsed_time = (overflow_count * 0x10000) + finish_time
- start_time;
```

```
        uint32_t current_record =
eeprom_read_dword((uint32_t*)&record_time);

        if (elapsed_time < current_record) {
            eeprom_write_dword((uint32_t*)&record_time,
elapsed_time);
            PORTA |= (1 << BUZZER);  // Подача сигнала о рекорде
на зуммер
            _delay_ms(100);
            PORTA &= ~(1 << BUZZER); // Выключение зуммера
               UART_SendString("New record!\r\n");
        }

        test_times[test_count++ % 8] = elapsed_time;
    }
     sei();
}

// Функция вычисления среднего времени прочтения таблиц
uint32_t calculate_average_time() {
     cli();
    uint64_t sum = 0;
    for (int i = 0; i < test_count && i < 8; i++) {
        sum += test_times[i];
    }
     sei();
    return (uint32_t)(sum / (test_count < 8 ? test_count : 8));
}

void out_times() {
     cli();
     if   ((!table_show) && (test_count > 0)) {
          LCD_Clear(LCD1_RS, LCD1_E);
         LCD_Clear(LCD2_RS, LCD2_E);
         LCD_Clear(LCD3_RS, LCD3_E);

         char buffer[8];

          LCD_Command(LCD1_RS, LCD1_E, 0x80);
         LCD_Print(LCD1_RS, LCD1_E, "N Time  N Time");

          for (int i = 0; (i < test_count) && (i < 8); i++){
               if (i < 2){
                    if (i == 0) LCD_Command(LCD1_RS, LCD1_E,
0xC0);
                    custom_utoa(i + 1, buffer, 10);
                    LCD_Print(LCD1_RS, LCD1_E, buffer);
                    LCD_Print(LCD1_RS, LCD1_E, " ");
                    custom_utoa(test_times[i], buffer, 10);
                    LCD_Print(LCD1_RS, LCD1_E, buffer);
                    LCD_Print(LCD1_RS, LCD1_E, " ");
               }
```

```
                else if (i < 6){
                     if (i == 2) LCD_Command(LCD2_RS, LCD2_E,
0x80);
                     else if (i == 4) LCD_Command(LCD2_RS,
LCD2_E, 0xC0);
                     custom_utoa(i + 1, buffer, 10);
                     LCD_Print(LCD2_RS, LCD2_E, buffer);
                     LCD_Print(LCD2_RS, LCD2_E, " ");
                     custom_utoa(test_times[i], buffer, 10);
                     LCD_Print(LCD2_RS, LCD2_E, buffer);
                     LCD_Print(LCD2_RS, LCD2_E, " ");
                }
                else if (i < 8){
                     if (i == 6) LCD_Command(LCD3_RS, LCD3_E,
0x80);
                custom_utoa(i + 1, buffer, 10);
                     LCD_Print(LCD3_RS, LCD3_E, buffer);
                     LCD_Print(LCD3_RS, LCD3_E, " ");
                     custom_utoa(test_times[i], buffer, 10);
                     LCD_Print(LCD3_RS, LCD3_E, buffer);
                     LCD_Print(LCD3_RS, LCD3_E, " ");
                }
           }
           LCD_Command(LCD3_RS, LCD3_E, 0xC0);
           LCD_Print(LCD3_RS, LCD3_E, "AvTime ");
           LCD_Print(LCD3_RS, LCD3_E, buffer_while);

           send_test_times_via_bluetooth();  // Отправка
результатов на Bluetooth
      }
      sei();
}

ISR(INT1_vect) {
    if (PIND & (1 << BUTTON_NEXT_TABLE)) {
        generate_schulte_table();
    }
}
ISR(INT0_vect) {
    if (PIND & (1 << BUTTON_START_STOP)) {
        if (test_running) {
                stop_timer();
                if (test_count > 0){
                     uint32_t avg_time =
calculate_average_time();
                  custom_utoa(avg_time, buffer_while, 10);
                     out_times();
                }
           }
        else start_timer();
    }
}
```

```
ISR(TIMER1_OVF_vect) {
     if (test_running){
     overflow_count++;  // Счетчик переполнений
          // PORTB = overflow_count;
     }
}

// Основная функция
int main() {

    DDRD &= ~((1 << BUTTON_START_STOP) | (1 <<
BUTTON_NEXT_TABLE));
    PORTD |= (1 << BUTTON_START_STOP) | (1 <<
BUTTON_NEXT_TABLE);

     // DDRB = 0xFF;
    // PORTB = 0x00;

     // Настройка таймера 1 в нормальном режиме
     TCCR1A = 0; // Режим нормального счётчика
     TCCR1B = (1 << CS11); // Предделитель 8
     TCNT1 = 0; // Сброс счётчика

     UART_Init();
     TIMSK |= (1 << TOIE1); // Разрешение прерывания по
переполнению таймера 1


    // Настройка прерываний
    MCUCR = (1 << ISC01) | (1 << ISC00) | (1 << ISC11) | (1 <<
ISC10); // Прерывание на спадающем фронте
    GICR = (1 << INT0) | (1 << INT1);

     eeprom_write_dword((uint32_t*)&record_time, 0xFFFFFFFF);

    // Инициализация LCD
    LCD_Init(LCD1_RS, LCD1_E);
    LCD_Init(LCD2_RS, LCD2_E);
    LCD_Init(LCD3_RS, LCD3_E);

    sei();  // Включаем глобальные прерывания

    while (1) {}
}
```

Приложение Б

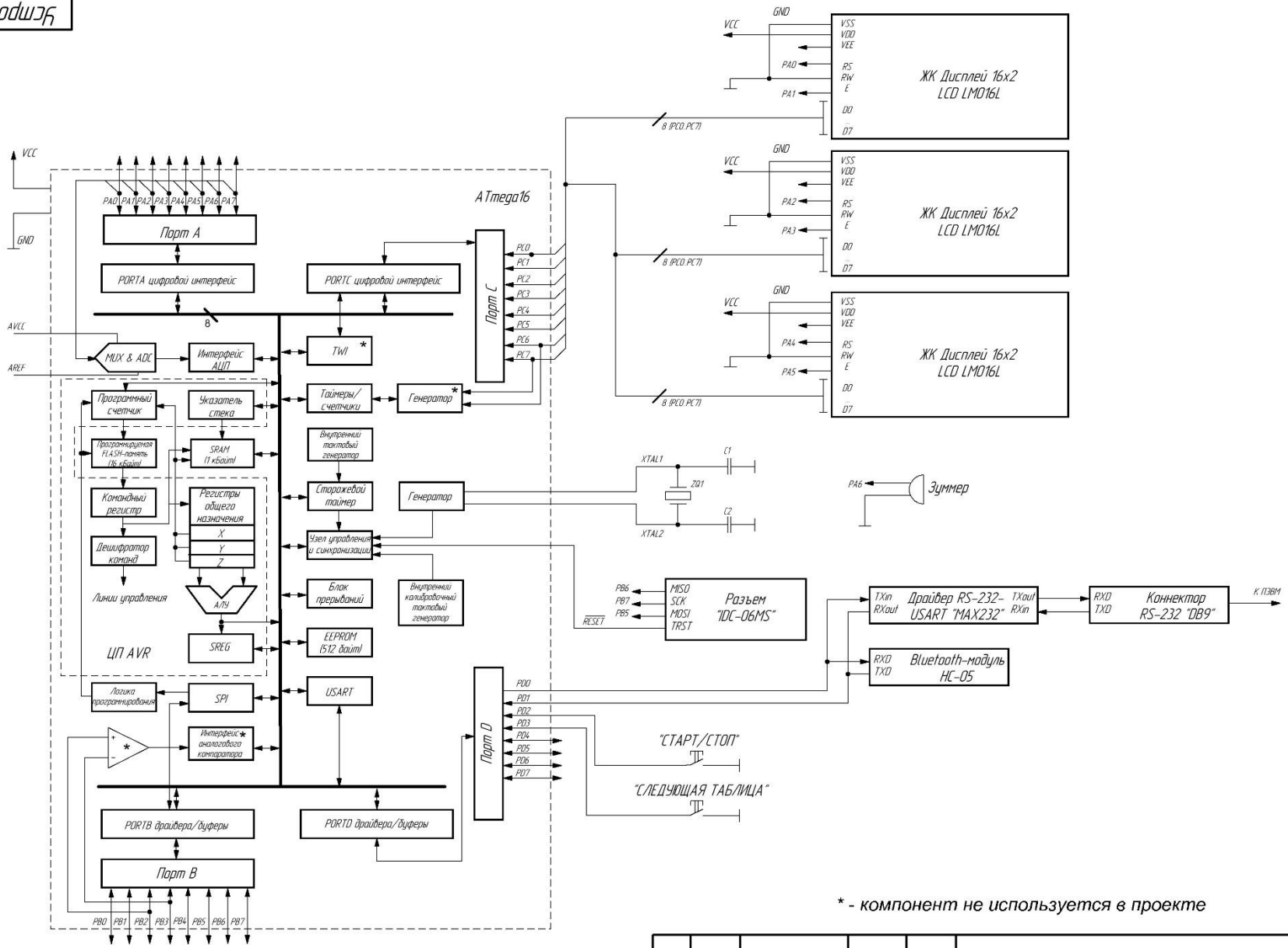
Графическая часть

На 2 листах

Электрическая схема функциональная

Электрическая схема принципиальная

Устройство для измерения скорости чтения



* - компонент не используется в проекте

Изм	Лист	№ докум.	Подпись	Дата
Разраб.		Третьяков А.А.		
Провер.		Трамов И.Б.		
Т. контр.				
Реценз.				
Н. контр.		Трамов И.Б.		
Утверд.				

Устройство для измерения скорости чтения

Схема электрическая функциональная

Лит	Масса	Масштаб

Лист 1 Листов 1

МГТУ им. Н.Э.Баумана
Группа ИУ6-73Б

Копировал

Формат

Перв. примен

Справ. №

Подп. и дата

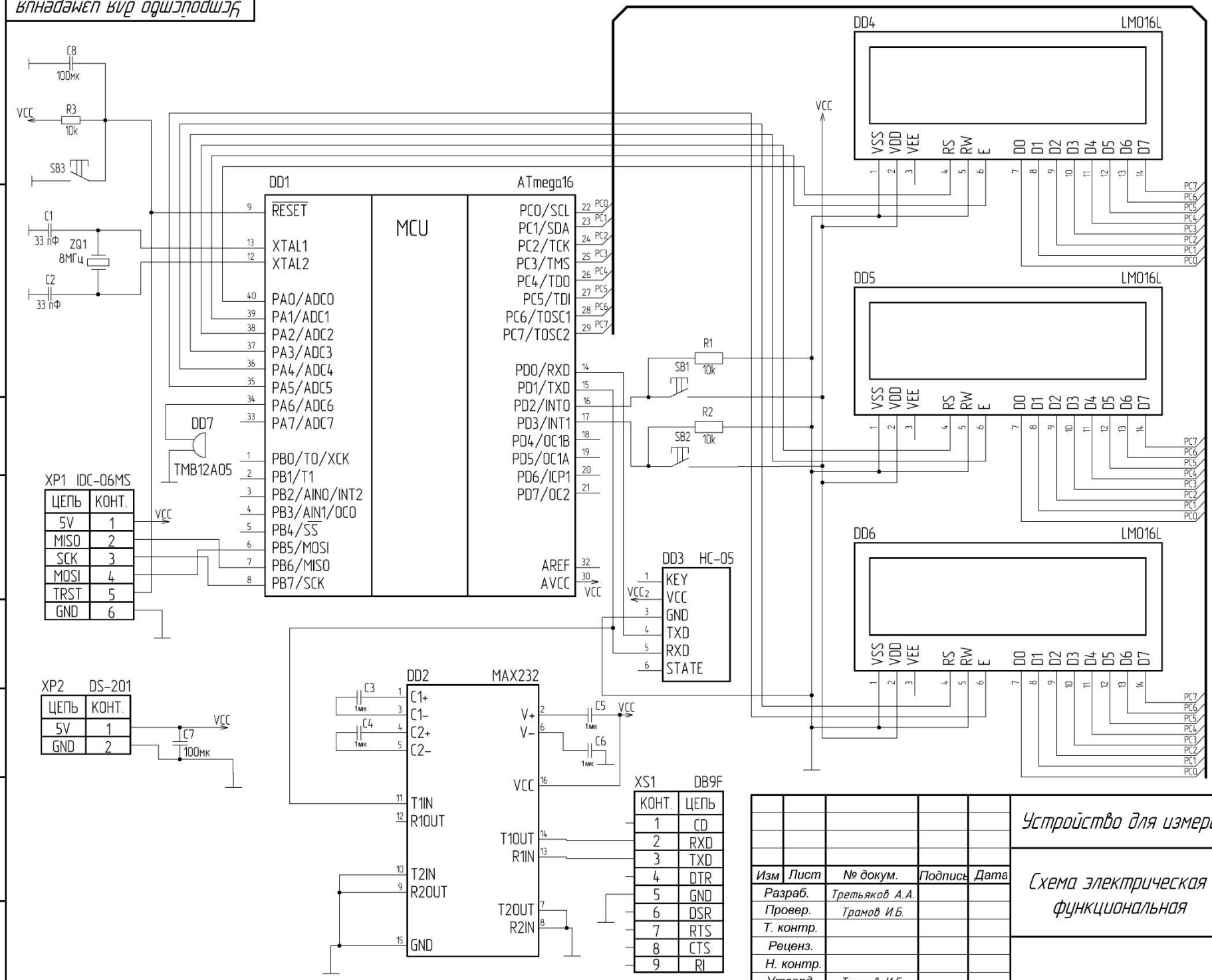
Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Устройство для измерения скорости чтения



Изм	Лист	№ докум.	Подпись	Дата
Разраб.		Третьяков А.А.		
Провер.		Трамов И.Б.		
Т. контр.				
Реценз.				
Н. контр.				
Утверд.		Трамов И.Б.		

Схема электрическая функциональная

Лит	Масса	Масштаб

Лист 1 Листов 1

МГТУ им. Н.Э.Баумана
Группа ИУ6-73Б

Копировал

Формат

Перв. примен

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Приложение В

Перечень элементов

Зона	Поз. обозначение	Наименование	Кол.	Примечание
		Конденсаторы		
	C1, C2	FKP-2 – 33 пФ	2	
	C3 – C6	K73-17 МПО – 1 мкФ	4	
	C7, C8	K50-35 – 100 мкФ	2	
		Схемы интегральные		
	DD1	ATmega16	1	
	DD2	MAX232	1	
	DD3	HC-05	1	
	DD4-DD6	LM016L	3	
	DD7	TMB12A05	1	
		Резисторы		
	R1 – R3	МЛТ-0.125 – 10 кОм	3	
		Коммутационные устройства		
	SB1-SB3	K-1-1 M20	3	
		Разъемы		
	XP1	IDC-06MS	1	
	XP2	DB9F	1	
	XS1	DS-201	1	
		Резонатор кварцевый		
	ZQ1	HC-49S – 8 МГц	1	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата	Устройство для измерения скорости чтения		
Разраб.		Третьяков А.А.			Перечень элементов	Лит	Лист 1 / Листов 1
Пров.		Трамов И.Б.					
Утв.		Трамов И.Б.				МГТУ им. Н.Э.Баумана Группа ИУ6-73Б	

Копировал

Формат А4