|  |  |
| --- | --- |
|  | **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 **09.03.01 Информатика и вычислительная техника**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***к курсовой работе***

***по дисциплине «Микропроцессорные системы»***

***на тему:***

**Тренажер для оператора**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | С.В. Астахов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Руководитель |  |  |  | С.А. Хохлов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |

*2022 г.*



**РЕФЕРАТ**

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

МИКРОКОНТРОЛЛЕР, ATMEGA8535, ПУЛЬТ ОПЕРАТОРА, ТРЕНАЖЕР, СКОРОСТЬ РЕАКЦИИ.

Объектом разработки является устройство, измеряющее скорость реакции оператора на движущуюся мишень.

Цель работы – создание программного обеспечения, функциональной и принципиальной схем описанного устройства.

Задачи, решенные в процессе выполнения курсовой работы:

* анализ объекта разработки на функциональном уровне;
* разработка функциональной схемы;
* выбор элементной базы для реализации устройства;
* разработка принципиальной схемы;
* расчет потребляемой мощности;
* разработка алгоритмов работы программного обеспечения;
* реализация программного обеспечения устройства на языке С;
* отладка программного обеспечения в среде симуляции Proteus 8.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**АЦП** — аналого-цифровой преобразователь.

**МК** — микроконтроллер.

**RISC** — (reduced instruction set computer) архитектурный подход к проектированию процессоров, в которой быстродействие увеличивается за счёт такого кодирования инструкций, чтобы их декодирование было более простым, а время выполнения — меньшим.

**UART** — (Universal Asynchronous Receiver-Transmitter) узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству.

**SPI** — (Serial Peripheral Interface) последовательный периферийный интерфейс, шина SPI) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии.

**I2C** — (Inter-Integrated Circuit) последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов.

**USB** — (Universal Serial Bus) последовательный интерфейс для подключения периферийных устройств к вычислительной технике.

**RS-232** — (Recommended Standard 232) стандарт физического уровня

для асинхронного интерфейса (UART).

SRAM, ERAM, EEPROM, (FLASH), ССИ (7-сегментный индикатор)**Содержание**

[1 Конструкторская часть 7](#_Toc1)

[1.1 Разработка функциональной схемы 7](#_Toc2)

[1.1.1 Разработка обобщенной функциональной схемы 7](#_Toc3)

[1.1.2 Описание архитектуры и технические характеристики микроконтроллера 8](#_Toc4)

[1.1.3 Детализация функциональной схемы 10](#_Toc5)

[1.2 Разработка принципиальной схемы 10](#_Toc6)

[1.3 Алгоритм основной программы 11](#_Toc7)

[2 Технологическая часть 14](#_Toc8)

[2.1 Тестирование программы 14](#_Toc9)

Введение

В данной работе производится разработка тренажера для оператора. Основная функция устройства – измерение времени реакции оператора на одну движущуюся мишень и среднего времени реакции на K (K=15) мишеней. Мишени представлены горящими светодиодами, ввод информации происходит с матричной клавиатуры, вывод – на дисплей из 7-сегментных индикаторов. Кроме того, устройство должно издавать звук при неправильном «захвате мишени», иметь возможность изменения скорости движения мишеней и возможность передачи результатов испытаний в ПЭВМ.

В устройстве использован микроконтроллер ATmega8535 семейства AVR. Микроконтроллеры AVR имеют гарвардскую архитектуру (программа и данные находятся в разных адресных пространствах) и систему команд, близкую к идеологии RISC. Процессор AVR имеет 32 8-битных регистра общего назначения, объединённых в регистровый файл. В отличие от «идеального» RISC, регистры не абсолютно ортогональны.

Целевой МК имеет четыре 8-разрядных порта ввода-вывода, 10-разрядный АЦП, два 8-разрядных и один 16-разрядный таймер, EEPROM и RAM объемом по 512 байт, встроенные интерфейсы I2C, SPI, UART. МК способен работать на частоте до 16 МГц.

# **1 Конструкторская часть**

## 1.1 Разработка функциональной схемы

### 1.1.1 Разработка обобщенной функциональной схемы

На основе текста задания можно определить, что система будет состоять из следующих элементом:

* дисплей из 7-сегментных индикаторов;
* матричная клавиатура;
* светодиодная матрица;
* зуммер (звукоизвлекатель).

Кроме того, необходимо осуществлять передачу данных к ПЭВМ, для чего понадобится драйвер, преобразующий сигналы интерфейса UART/I2C/SPI в сигналы интерфейса USB/RS-232.

Для управления дисплеем и светодиодной матрицей так же будем использовать вспомогательные драйвера, чтобы сократить количество задействованных контактов МК.

На основе результатов первичного анализа требований была составлена обобщенная функциональная схема, представленная на рисунке 1.



Рисунок 1 — обобщенная функциональная схема

### 1.1.2 Описание архитектуры и технические характеристики микроконтроллера

Семейство микроконтроллеров AVR включает tinyAVR, megaAVR и XMEGA AVR.

В контроллерах типа tinyAVR максимальное число линий ввода-вывода составляет 18, чего будет недостаточно для подключения 5 периферийных устройств. Возможности контроллеров типа XMEGA AVR наиболее широки, но, при использовании контроллеров данного типа большинство их возможностей не будет использовано, поэтому для реализации устройства было решено использовать контроллер типа megaAVR.

Среди микроконтроллеров типа megaAVR был выбран ATmega8535, который обладает следующими характеристиками:

* 8-битная шина данных;
* тактовая частота до 16 МГц;
* четыре 8-разрядных порта ввода-вывода (32 контакта);
* 8 КБайт памяти программ;
* 10-разрядный АЦП;
* встроенные интерфейсы I2C, SPI, UART;
* два 8-разрядных и один 16-разрядный таймер/счетчик.

Так как в микроконтроллерах AVR используется гарвардская архитектура, в соответствии с которой разделены адресные пространства памяти программ и памяти данных. Такая структура позволяет процессору одновременно работать с памятью программ и памятью данных, что позволяет повысить производительность.

В памяти данных регистровая и оперативная память (SRAM) образуют единое адресное пространство, которое может быть расширено за счет подключения внешнего запоминающего устройства ERAM.

Для долговременного хранения данных, которые могут изменяться в процессе работы МК используется энергозависимая память EEPROM.

Карта памяти ATmega8535 представлена на рисунке 2.



Рисунок 2 — карта памяти ATmega8535

Структурная схема ATmega8535 представлена на рисунке 3.



Рисунок 3 — структурная схема ATmega8535

### 1.1.3 Детализация функциональной схемы

Для детализации функциональной схемы были подобраны цифровые схемы, дисплей на основе 7-сегментных индикаторов и детализировано устройство светодиодной матрицы. Кроме того, описано к каким линиям портов МК подключаются те или иные устройства.

Примечание: в данном разделе используемые схемы описаны лишь поверхностно, подробнее см. раздел «Проектирование принципиальной схемы».

В качестве устройства ввода используется матричная клавиатура 4х4, активная строка в которой выбирается с помощью линий PA0-PA3, а столбцы считываются с помощью линий PA4-PA7 порта A.

В качестве дисплея используется схема FYQ-5641AUG-21 с общим катодом. Выбор активного индикатора при динамической индикации осуществляется с помощью линий PB4-PB7 порта B. Выбор сегментов осуществляется с помощью драйвера семейства 7448, подключенного к линиям PB0-PB3 порта B.

Передача данных на ПЭВМ осуществляется с помощью интерфейса UART через драйвер USB-UART, подключенный к линиям PD0(Rx), PD1(Tx) порта D.

Звуковой сигнал вырабатывается с помощью зуммера со встроенным генератором, подключаемого к линии PD7 порта D.

Светодиодная матрица реализована на базе светодиодных полос по 4 светодиода. Так как на основе требований задания можно заключить, что одновременно может быть активен всего один светодиод, матрица управляется с помощью дешифратора 5-32, реализованного на базе дешифратора 3-8 и восьми дешифраторов 2-4. Линии PC0-PC4 подключены к адресным разрядам дешифраторов, а линия PC7 — к разрешающему входу.

Полученная функциональная схема содержится в Приложении Х.

## 1.2 Разработка принципиальной схемы

## 1.3 Алгоритм основной программы

Схема алгоритма представлена на рисунках Х и Х, а код основной программы — в приложении Х.

Основная идея алгоритма состоит в том, что система может находиться либо в режиме прохождения серии испытаний, либо в режиме паузы.

В активном режиме (режиме прохождения испытаний) можно ускорять или замедлять скорость движения мишени. В случае правильного «захвата цели» на дисплей будет выведено время реакции в данном испытании в миллисекундах, а горящий светодиод «переместится» на другую строку матрицы. Иначе — система издаст звуковой сигнал. После окончания заданного числа испытаний система выведет среднее время реакции и перейдет в режим паузы. Также испытания можно прервать нажатием кнопки «Restart» (в этом случае дисплей будет погашен).

В режиме паузы светодиоды погашены. Если серия испытаний была пройдена до конца, можно отправить на ПЭВМ через драйвер UART-USB информацию о времени реакции на каждую мишень и о среднем времени реакции. Можно начать новое испытание кнопкой «Restart», переменные для подсчета времени и числа пройденных испытаний обнулятся, как и показания дисплея.

Назначение некоторых используемых переменных и функций:

* active — флаг режима прохождения испытаний;
* timer\_ms — время с начал испытания (обновляется по прерыванию от Т1);
* tries\_counter — cчетчик числа завершенных испытаний в серии;
* display\_set\_long(long target); — вывести на дисплей значение, либо специальные символы, если значение не умещается;
* leds\_random\_line(); — «перемещает» мишень в другую строку и возвращает номер строки.

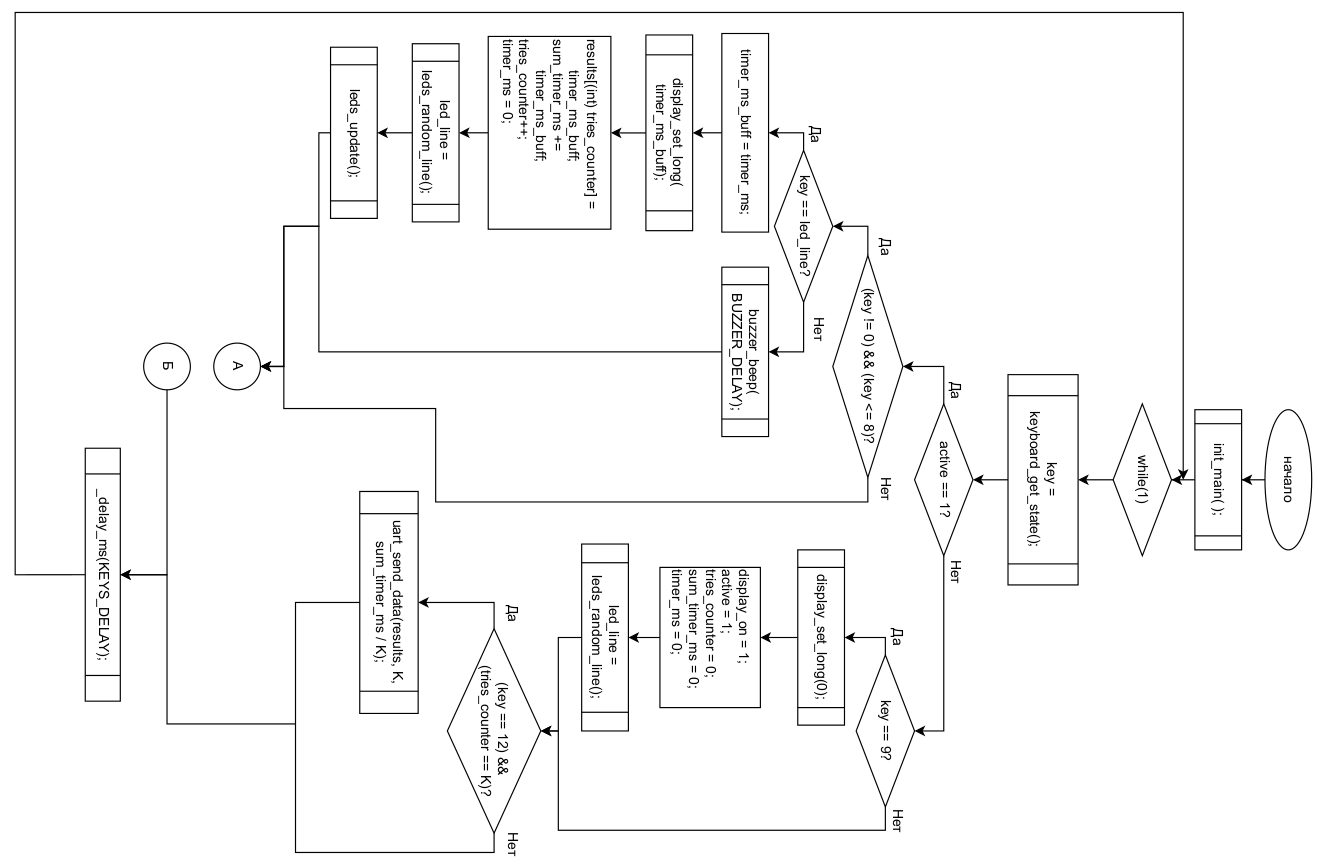


Рисунок Х — алгоритм основной программы



Рисунок Х — алгоритм основной программы

1.4 Управление динамической индикацией

Для отображения скорости реакции используется дисплей из четырех ССИ. На нем осуществляется динамическая индикация числа с переключением разрядов по переполнению таймера Т0.

Минимальная частота при которой человек воспринимает мигающее изображение как статическое — 24 Гц. Так как используется четыре индикатора, их необходимо переключать с частотой 24\*4 = 96 Гц. При тактовой частоте 4 МГц и размерности таймера в 8 бит коэффициент предделителя можно определить из уравнения:

4 000 000 / (28 \* K) = 24 \* 4

K = 4 000 000 / (28 \* 24 \* 4) = 162.76

Наиболее близкие значения предделителя к полученному — 64 и 256. Но выбор значения 256 приведет к тому, что частота индикации будет ниже предельной. Поэтому было выбрано значение 64. Для установки этого значения в соответствии с таблицей Х были установлены биты CS01 и CS0 регистра TCCR0.

Таблица Х — выбор источника тактового сигнала для таймера/счётчика Т0

|  |  |  |  |
| --- | --- | --- | --- |
| **CS02** | **CS01** | **CS00** | **Источник тактового сигнала** |
| 0 | 0 | 0 | Таймер/счётчик остановлен |
| 0 | 0 | 1 | CK (тактовый сигнал микроконтроллера) |
| 0 | 1 | 0 | CK/8 |
| 0 | 1 | 1 | CK/64 |
| 1 | 0 | 0 | CK/256 |
| 1 | 0 | 1 | CK/1024 |
| 1 | 1 | 0 | Вывод Т0, инкремент счётчика производится  по спадающему фронту импульсов |
| 1 | 1 | 1 | Вывод Т0, инкремент счётчика производится  по нарастающему фронту импульсов |

1.5 Отсчет времени реакции

Отсчет времени реакции реакции производится за счет переменной timer\_ms, инкрементируемой каждую миллисекунду по прерыванию таймера Т1, работающего в режиме сравнения.

Кроме того, в обработчике этого прерывания можно осуществлять переключение горящего светодиода в светодиодной матрице, чтобы организовать задержку переключения светодиодов независимо от задержки считывания кнопок.

Для повышения точности возьмем значение предделителя К=1. Рассчитаем значение регистра сравнения OCR1A:

(F\_CPU / K) / OCR1A = △t

Где △t = 1с = 1000 мс, К = 1

(F\_CPU / 1) / OCR1A = 1000

OCR1A = F\_CPU / 1000

Код обработчика прерываний Т1 в режиме сравнения приведен в листинге Х.

Листинг Х — код обработчика прерываний Т1 в режиме сравнения

|  |
| --- |
| ISR (TIMER1\_COMPA\_vect)  {  timer\_ms++;  if((active == 1) && (timer\_ms % delay\_amount == 0)){  leds\_move\_column();  leds\_update();  }  TCNT1=0; //clear ticks  } |

С помощью режима отладки и показателя Stop Watch было определено время обработки прерывания при выполнении условий ветвления и при их невыполнении.

Данный процесс проиллюстрирован на рисунках Х-Х.



Рисунок Х — начало обработки прерывания



Рисунок Х — конец обработки прерывания при несоблюдении условий ветвления



Рисунок Х — конец обработки прерывания при соблюдении условий ветвления

Время обработки прерывания в случае несоблюдения и соблюдения условий ветвления составило соответственно:

tfalse = (737.75 - 722.00) = 15.75 мкс

ttrue = (881.75 - 722.00) = 159.75 мкс

Очевидно, что при минимальном времени задержки между переключениями светодиодов будет получено максимальное значение времени обработки прерывания, а значит и максимальная погрешность значения timer\_ms.

Минимальное время задержки между переключениями светодиодов (MIN\_DELAY) составляет 100 мс = 100 000 мкс. Отсюда было найдено значение относительной погрешности timer\_ms.

t = (t / t) \* 100% = ( (99 \* tfalse + ttrue) / t ) \* 100% = ( (99 \* 15.75 + 159.75) / 100 000) \* 100% = 1.719%.

При скорости реакции в пределах 10 с (что крайне много) погрешность не превысит 18 мс. По сравнению с задержкой считывания с клавиатуры и временем выполнения основной программы это число незначительно.

# 2 Технологическая часть

## 2.1 Тестирование программы

Для тестирования программы была построена упрощенная схема системы в среде Proteus 8 (рисунок Х).

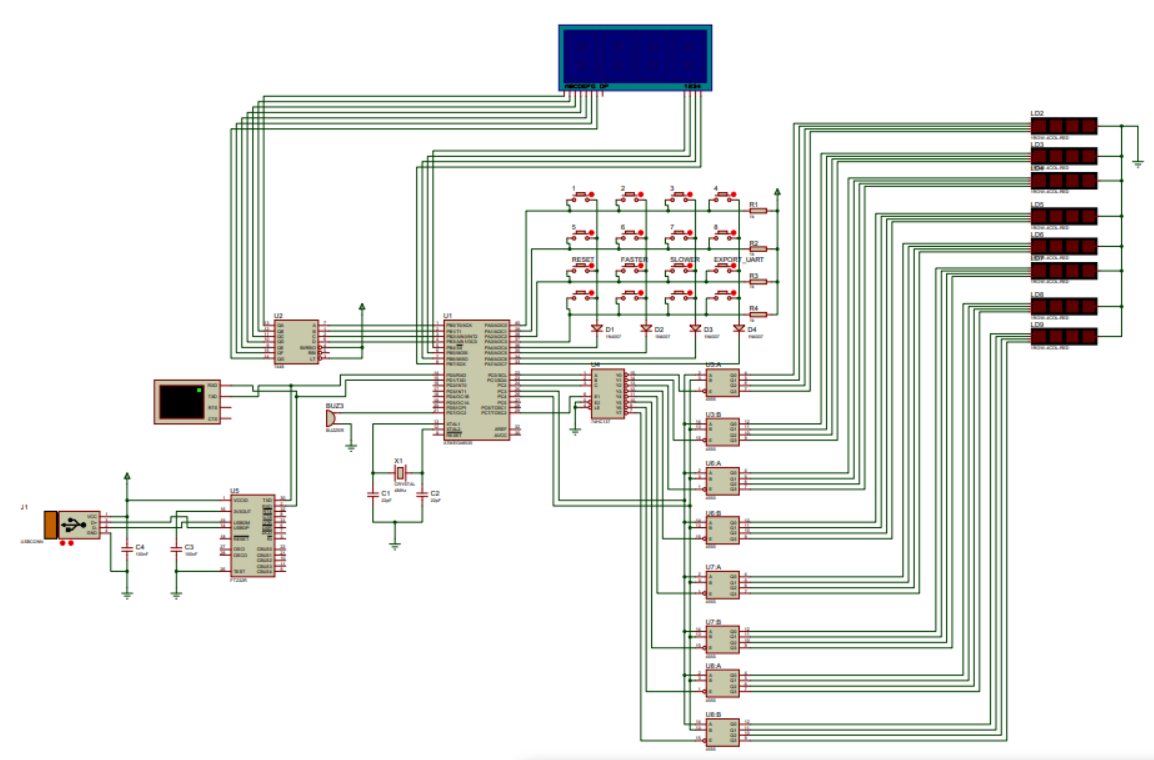


Рисунок Х — упрощенная схема тренажера оператора в Proteus

Ввод информации систему осуществляется посредством матричной клавиатуры 4х4. Вывод информации осуществляется на светодиодную матрицу 8х4, дисплей из ССИ. Для проверки работы UART используется виртуальный терминал.

Между серий испытаний экран и светодиодная матрица погашены. В начале каждой новой серии испытаний дисплей отображает «0000». После каждого правильного выбора строки виртуальный дисплей отображает в миллисекундах время, за которое оператор отреагировал на мишень.

Если время реакции >9999 мс, то дисплей отображает специальные символы, напоминающие литеру «E» (что подразумевает «Error» — ошибка отображения). При этом испытание все равно считается выполненным и его результаты будут сохранены в памяти МК.

Пример отображения времени реакции <10000 мс показан на рисунке Х.



Рисунок Х — отображение времени реакции

Состояние дисплея при превышении его разрядности (время реакции >9999 мс) показано на рисунке Х.



Рисунок Х — состояние дисплея при превышении разрядности

Пример передачи данных на ПЭВМ показан на рисунке Х. Для удобства тестирования было установлено число испытаний в серии К=3.



Рисунок Х — передача данных по UART

Также было проверено, что МК издает звуковой сигнал при неправильном нажатии клавиши. Кроме того, проверено, что нельзя передать данные по UART пока не пройдена до конца серия испытаний, а также проверено выключение дисплея при прерывании серии испытаний кнопкой «RESTART» и установка «0000» при запуске новой серии испытаний. Проверена работоспособность кнопок регулирования скорости движения светодиодов.