



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

\_\_\_\_\_  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

С.В. Астахов

\_\_\_\_\_  
(И.О. Фамилия)

Руководитель

\_\_\_\_\_  
(Подпись, дата)

С.А. Хохлов

\_\_\_\_\_  
(И.О. Фамилия)

2022 г.

УТВЕРЖДАЮ  
Заведующий кафедрой ИУ-6  
А.В. Пролетарский  
« 2 » сентября 2022 г.

## ЗАДАНИЕ на выполнение курсовой работы

по дисциплине Микропроцессорные системы

Студент Астахов С.В. (ИУ6-72Б)

(фамилия, инициалы, индекс группы)

Направленность курсовой работы – учебная

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

**Тема курсовой работы** Тренажер для оператора

Разработать устройство для измерения времени реакции оператора на движущуюся мишень (светящийся светодиод) по одной из 8-и линеек светодиодной матрицы 8x4. После появления мишени испытуемый должен “захватить цель”, нажав кнопку с номером активной линейки на матричной клавиатуре 4x4 пульта оператора. Время с момента появления мишени до ее захвата вывести на дисплей из 7-сегментных индикаторов. В случае промаха сформировать короткий звуковой сигнал для SPEAKER. Подсчитать среднее время реакции оператора за K испытаний и также вывести на дисплей.

Разработать алгоритм запуска мишеней, выбирая линейку траектории по случайному закону. Предусмотреть регулировку скорости перемещения мишени вдоль линейки. Результаты испытаний переслать в ПЭВМ по команде с пульта оператора.

Микроконтроллер - ATmega8535, K=15.

Отладить модули разработанной программы с помощью симулятора.

Оценить потребляемую мощность устройства.

### Оформление курсовой работы

1. Расчетно-пояснительная записка на 30 листах формата А4.

2. Перечень графического материала

а) схема функциональная электрическая

б) схема принципиальная электрическая

Дата выдачи задания 4 сентября 2022 г.

Руководитель курсовой работы

Задание получил Астахов С.В. / « 4 » сентября 2022 г.

Дата защиты - 20 декабря 2022 г.

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

## РЕФЕРАТ

РПЗ X стр., X таблицы, X рисунков, X источников, X приложения.

МИКРОКОНТРОЛЛЕР, АТМЕГА8535, ПУЛЬТ ОПЕРАТОРА, ТРЕНАЖЕР, СКОРОСТЬ РЕАКЦИИ.

Объектом разработки является тренажер оператора.

Тренажер оператора представляет собой устройство, измеряющее скорость реакции оператора на движущуюся мишень. Устройство может выводить информацию о результатах испытаний на дисплей из ССИ и пересылать эти данные на ПЭВМ.

Цель работы – создание программного обеспечения, функциональной и принципиальной схем, спецификации 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 C;
- отладка программного обеспечения в среде симуляции Proteus 8.

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**АЦП** — аналого-цифровой преобразователь.

**ССИ** — семисегментный индикатор.

**МК** — микроконтроллер.

**ПЭВМ** — персональная электронная вычислительная машина.

**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.

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

**SRAM** — (static random access memory) статическая память с произвольным доступом, полупроводниковая оперативная память, в которой каждый разряд хранится в схеме с положительной обратной связью, позволяющей поддерживать состояние без регенерации.

**EEPROM** — (Electrically Erasable Programmable Read-Only Memory) электрически стираемое перепрограммируемое постоянное запоминающее устройство.

**FLASH** — разновидность полупроводниковой технологии электрически перепрограммируемой памяти

**ERAM, RS-232, SPI, I2C**

## Содержание

Введение.....	6
1 Конструкторская часть.....	7
1.1 Разработка функциональной схемы.....	7
1.1.1 Разработка обобщенной функциональной схемы.....	7
1.1.2 Описание архитектуры и технические характеристики микроконтроллера.....	8
1.1.3 Детализация функциональной схемы.....	10
1.2 Разработка принципиальной схемы.....	11
1.2.1 Драйвер дисплея (SN74LS48).....	11
1.2.2 Расчет транзисторных ключей.....	12
1.2.3 Драйвер USB-UART (FT232RL).....	14
1.3 Расчет потребляемой мощности.....	16
1.4 Разработка программной части.....	17
1.4.1 Используемые библиотеки и подпрограммы.....	17
1.4.2 Алгоритм основной программы.....	19
1.4.3 Управление динамической индикацией.....	22
1.4.4 Отсчет времени реакции.....	23
1.4.5 Работа с UART.....	25
2 Технологическая часть.....	30
2.1 Тестирование программы.....	30
ЗАКЛЮЧЕНИЕ.....	32
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	33
ПРИЛОЖЕНИЕ А. Исходный текст программы.....	34
ПРИЛОЖЕНИЕ Б. Функциональная электрическая схема.....	35
ПРИЛОЖЕНИЕ В. Принципиальная электрическая схема.....	36
ПРИЛОЖЕНИЕ Д. Перечень элементов.....	37

## **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 элементов:

- 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.

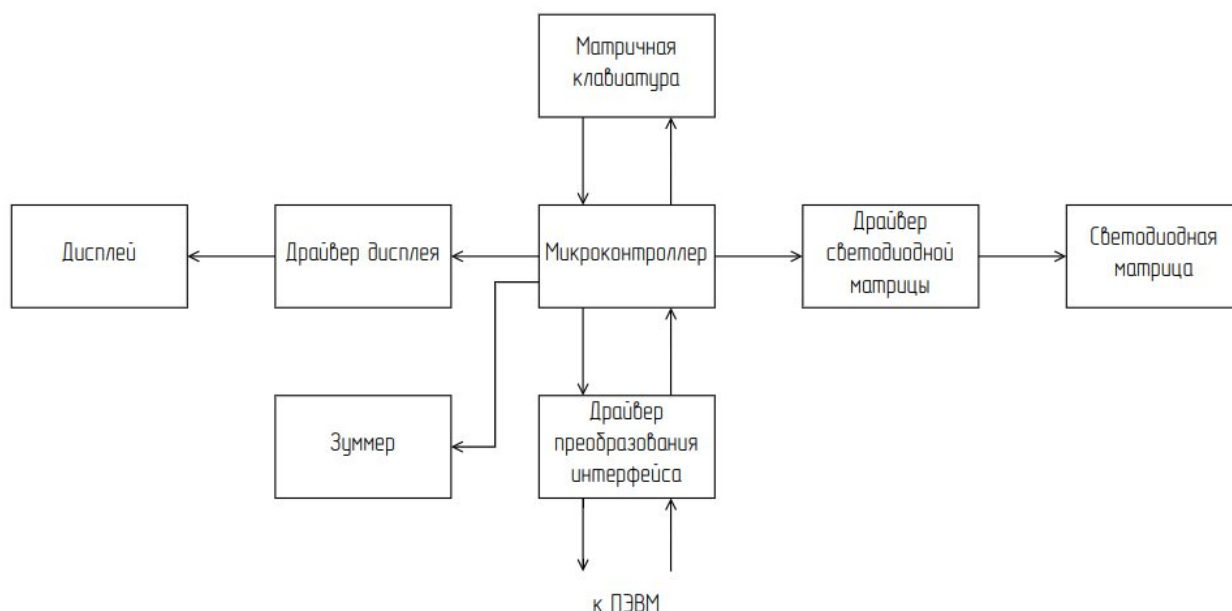


Рисунок 1 — обобщенная функциональная схема

### **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 АТmega8535

Структурная схема АТmega8535 представлена на рисунке 3.

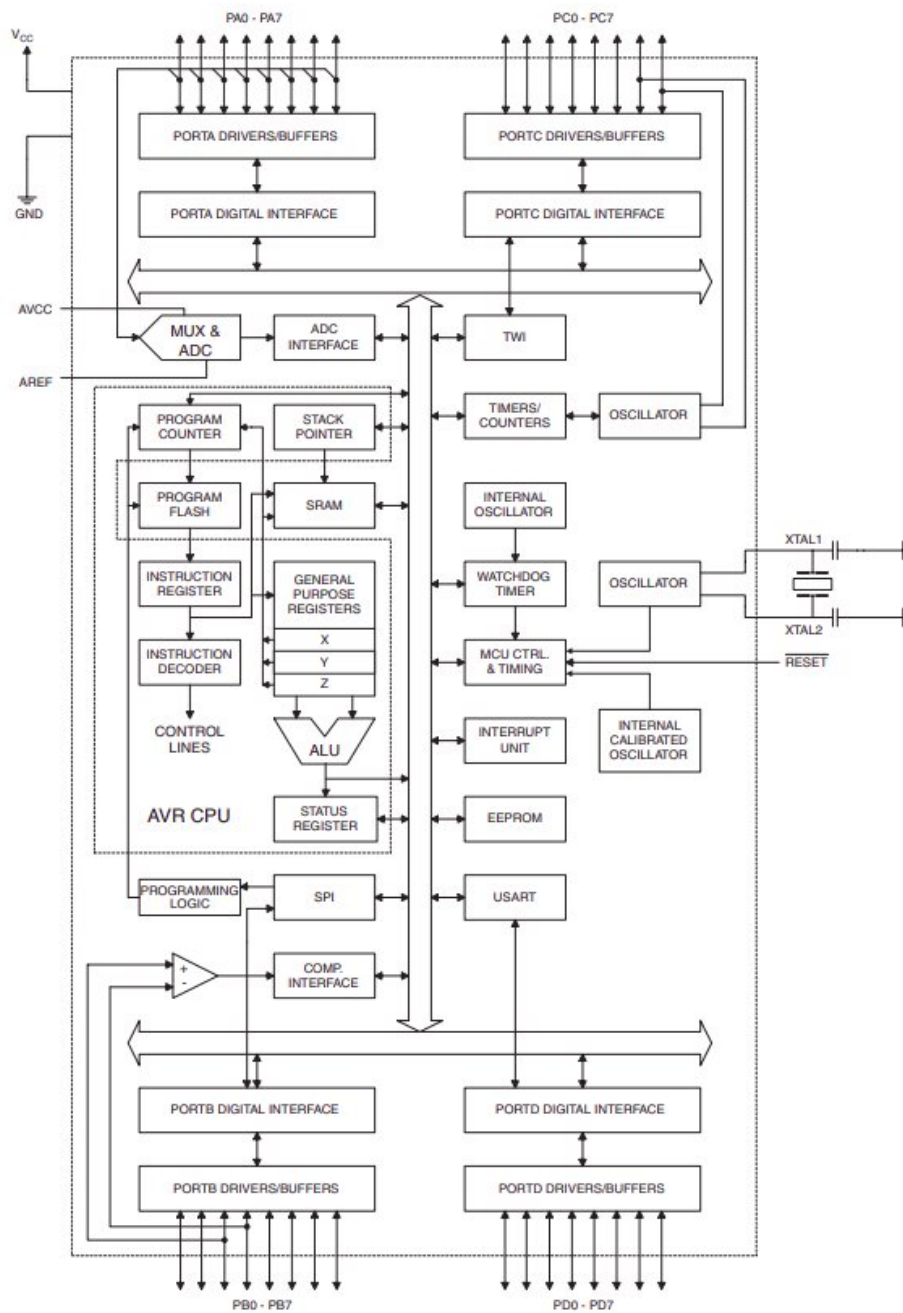


Рисунок 3 — структурная схема АТmega8535

### 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 4x4, 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 X.**

## 1.2 Разработка принципиальной схемы

### 1.2.1 Драйвер дисплея (SN74LS48)

Для упрощения управления ССИ в составе дисплея, а также для сокращения числа использованных контактов микроконтроллера был использован драйвер SN74LS48, преобразующий 4 бита двоично-десятичного кода (один десятичный разряд) в сигналы для сегментов ССИ.

Условное графическое обозначение драйвера представлено на **рисунке X**. (на изображении устройство отражено относительно вертикальной оси, как это сделано для удобства расположения на принципиальной схеме)

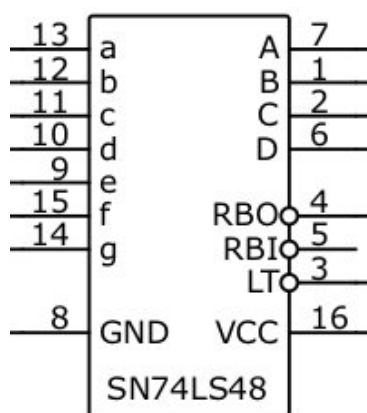


Рисунок X — условное графическое обозначение SN74LS48

Назначение контактов микросхемы:

Входы A, B, C, D — разряды двоично-десятичного числа.

Выходы a, b, ..., g — сигналы для сегментов ССИ.

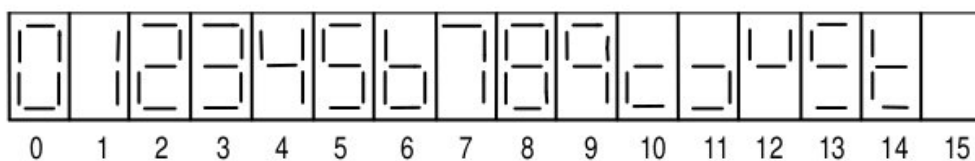
Вход LT — тест светодиодов (активация всех светодиодов).

**Вход RBI — не отображать 0.**

**Вход BI — не отображать 0.**

**Выход RBO — выключено отображение 0.**

Состояние ССИ в зависимости от входного числа показано на **рисунке X**.



**Рисунок X** — состояния ССИ

### 1.2.2 Расчет транзисторных ключей

Так как дешифраторы, используемые для выбора активного светодиода в матрице имеют малые значения предельного выходного тока, для питания светодиодов необходимо использовать транзисторные ключи.

Схема такого с использованием n-p-n транзистора приведена на рисунке X.

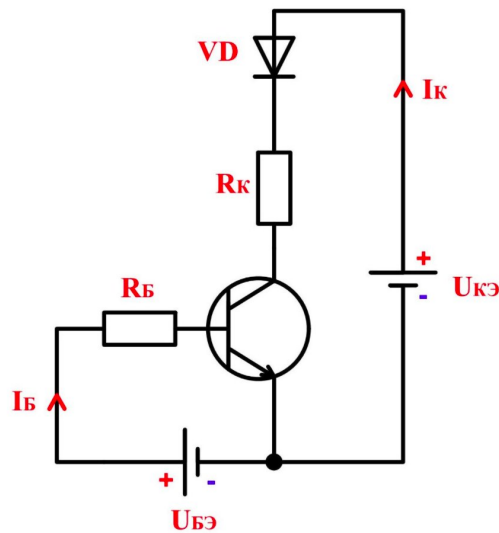


Рисунок X — транзисторный ключ

Резистор со стороны коллектора ограничивает ток, протекающий в светодиоде, его номинал рассчитывается по формуле (падением напряжения на светодиоде и переходе коллектор-эмиттер можно пренебречь):

$$R_k = (U_{vcc} - U_{vd} - U_{кэ}) / I_k = (5 - 2.5 - 0.4) / 0.02 = 105 \text{ Ом}$$

где  $U_{vcc}$  — напряжение питания;

$U_{vd}$  — падение напряжения на светодиоде;

$U_{кэ}$  — падение напряжения на переходе коллектор-эмиттер;

$I_k$  — предельный ток в цепи коллектора (ток через светодиод).

Ближайшие стандартные номиналы резисторов — 100 и 110 Ом. Так как резистор 100 Ом дает ток выше требуемого, что может вывести из строя светодиод, был выбран номинал 110 Ом.

$$\text{Тогда } I_k = (U_{vcc} - U_{vd} - U_{кэ}) / R_k^* = 2.1 / 110 = 19 \text{ мА}$$

Номинал резистора со стороны базы рассчитывается по формуле:

$$R_b = (U_{vcc} - U_{бэ}) / I_b = U_{бэ} / (I_{кэ} / \beta) = 4.4 / (0.019 / 200) = 46315 \text{ Ом}$$

где  $U_{бэ}$  — падение напряжения на переходе база-эмиттер;

$I_б$  — предельный ток в цепи базы;

$\beta$  — коэффициент усиления транзистора

Ближайшее номинальное значение — 47 кОм.

Так как светодиоды, используемые в ССИ имеют схожие характеристики, для управления ССИ были использованы транзисторные ключи с теми же номиналами резисторов.

Расчет транзисторного ключа для зуммера:

$$R_k = (U_{vcc} - U_{ha} - U_{кэ}) / I_k = (5 - 1.5 - 0.4) / 0.012 = 258 \text{ Ом}$$

Ближайшие стандартные номиналы резисторов — 252 и 270 Ом. Так как резистор 252 Ом дает ток выше требуемого, что может вывести из строя зуммер, был выбран номинал 270 Ом.

$$\text{Тогда } I_k = (U_{vcc} - U_{ha} - U_{кэ}) / R_k^* = 3.1 / 270 = 11.5 \text{ мА}$$

$$R_б = (U_{vcc} - U_{бэ}) / I_б = U_{бэ} / (I_{кэ} / \beta) = 4.4 / (0.0115 / 200) = 76521 \text{ Ом}$$

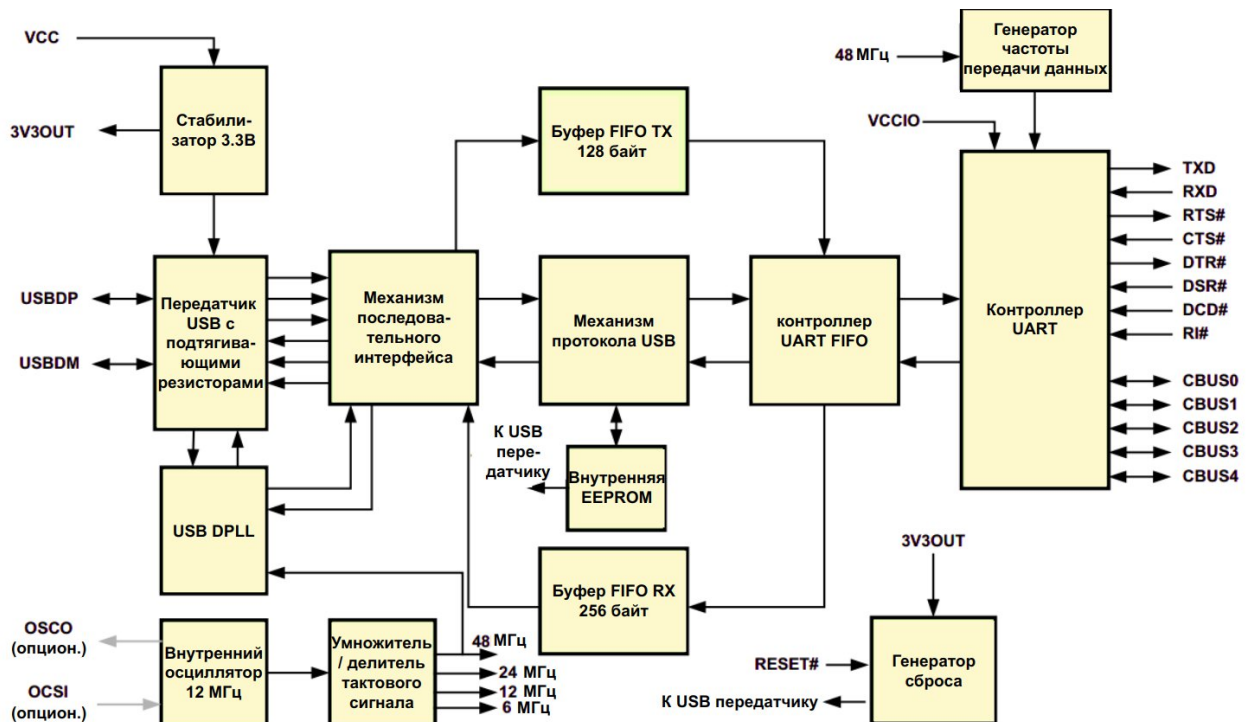
...

### 1.2.3 Драйвер USB-UART (FT232RL)

Так как USB является одним из наиболее популярных интерфейсов подключения периферийных устройств к ПЭВМ, было решено осуществлять пересылку данных на ПЭВМ через него.

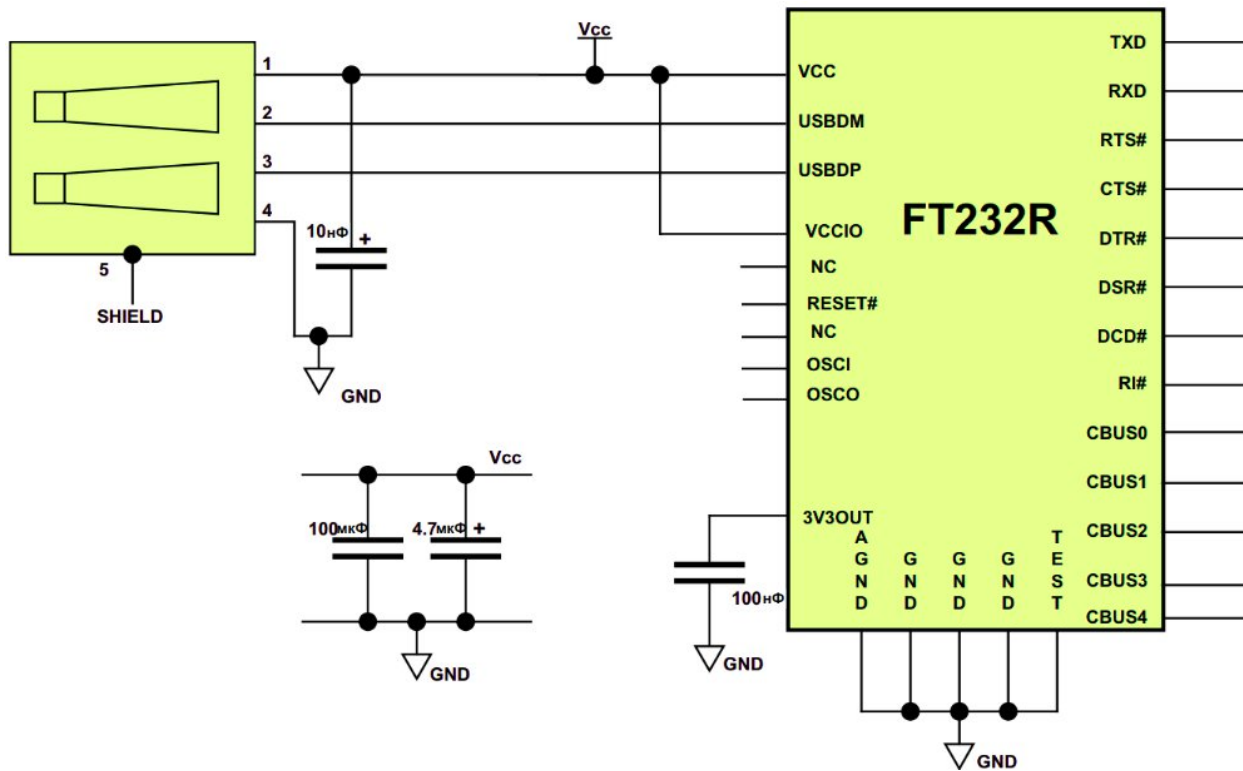
Так как МК ATmega8535 не имеет встроенной аппаратной поддержки USB, пересылка данных осуществляется через драйвер USB-UART. В качестве конкретной модели выбран драйвер FT232RL.

Структурная схема данного драйвера показана на рисунке X.



**Рисунок X** — структурная схема FT232RL.

Драйвер был подключен по схеме с питанием от шины, приведенной в документации. Схема представлена также на **рисунке X**.



**Рисунок X** — схема подключения драйвера с питанием от шины

### 1.3 Расчет потребляемой мощности

Таблица X — потребляемая мощность

Элемент	Ток потребления, мА	Потребляемая мощность, мВт	Количество, шт	Суммарная потребляемая мощность, мВт
ATmega8535	5.2	26.0	1	26.0
FT232RL	15.0	75.0	1	75.0
SN74LS48	25.0	125.0	1	125.0
74HC137	15.0		1	
CD4555			4	



## **1.4 Разработка программной части**

### **1.4.1 Используемые библиотеки и подпрограммы**

При написании программы была использована библиотека `avr-libc`, которая является адаптацией стандартной библиотеки Си под архитектуру микроконтроллеров AVR.

В проекте были указаны следующие заголовочные файлы из `avr-libc`:

- `avr/io.h` — содержит основную информацию, такую как имена регистров и констант;
- `avr/interrupt.h` — содержит адреса векторов прерываний;
- `util/delay.h` — позволяет управлять задержками;
- `util/setbaud.h` — упрощает управление скоростью передачи по UART.

Кроме того, был написан ряд собственных модулей и функций, позволяющих упростить управление подключенными к МК устройствами. Ниже приведены главные из них:

- `uart.h` — работа с UART:
  - `void uart_init()` — инициализировать настройки UART;
  - `void uart_send_byte(char c)` — отправить по UART один байт;
  - `void uart_send_long(long data)` — преобразовать целое число в ASCII и отправить по UART;
  - `void uart_send_data(long* res_array, char range, long average)` — отправить по UART данные о результатах испытаний.
- `display.h` — работа с дисплеем из ССИ:
  - `void display_set_bytes(char t1, char t2, char t3, char t4)` — сохранить значения разрядов в буферные переменные дисплея;
  - `void display_set_long(long target)` — разбить целое число на разряды и сохранить в буферных переменных дисплея;

- `void display_flash_once()` — переключить активный индикатор дисплея;
- `void display_off()` — выключить дисплей;
- `leds.h` — управление светодиодной матрицей:
  - `char leds_random_line()` — присвоить переменной, хранящей номер активной строки случайное значение из интервала `[0;8)`;
  - `void leds_move_column()` — изменить номер активного столбца;
  - `void leds_off()` — погасить светодиоды.
- `keyboard.h` — работа с клавиатурой:
  - `char keyboard_get_state()` — получить номер нажатой клавиши.

### 1.4.2 Алгоритм основной программы

Схема алгоритма основной программы представлена на **рисунках X и X**, а исходный код основной программы — в **приложении X**.

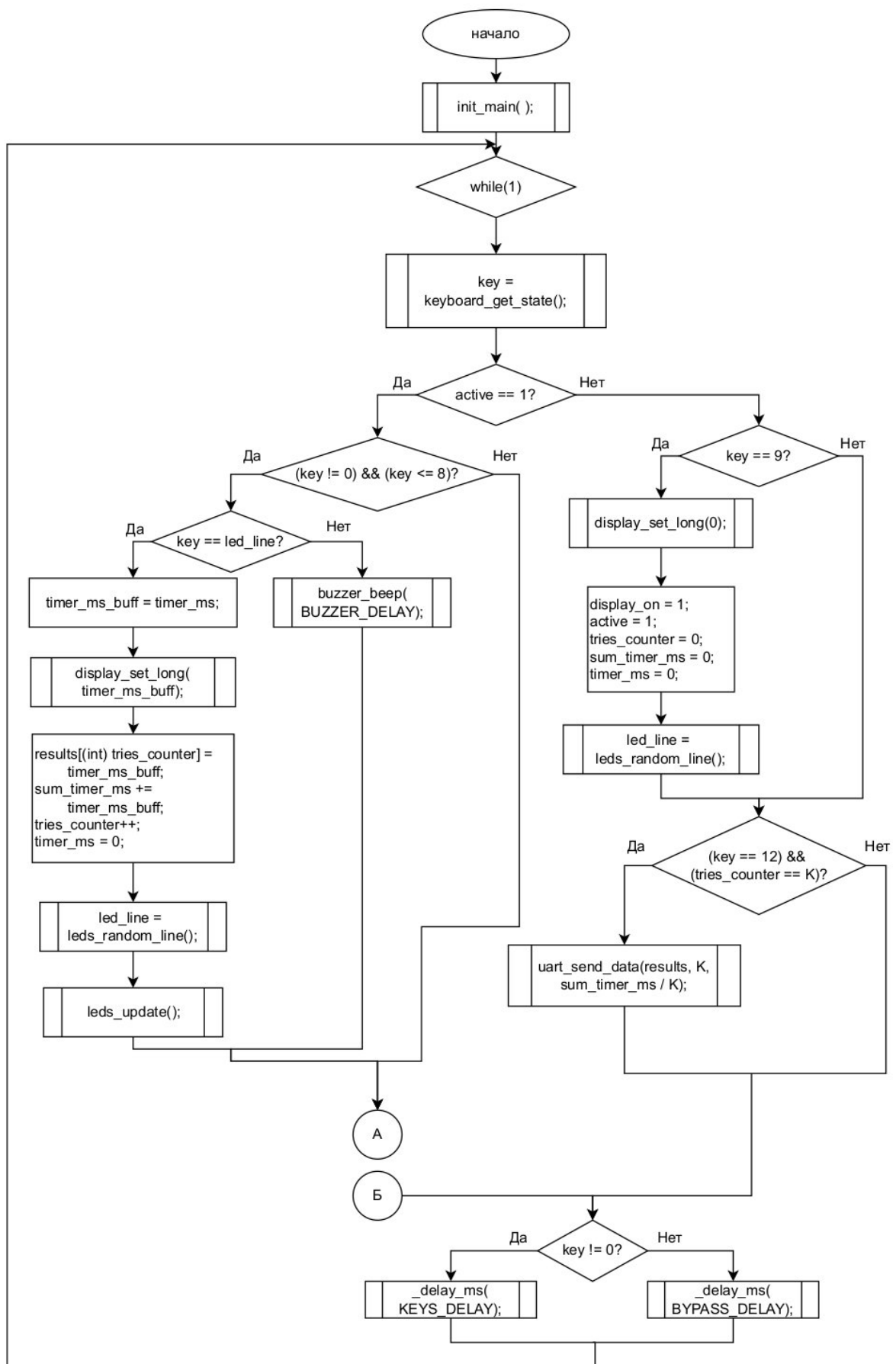
Основная идея алгоритма состоит в том, что система может находиться либо в режиме прохождения серии испытаний, либо в режиме паузы.

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).

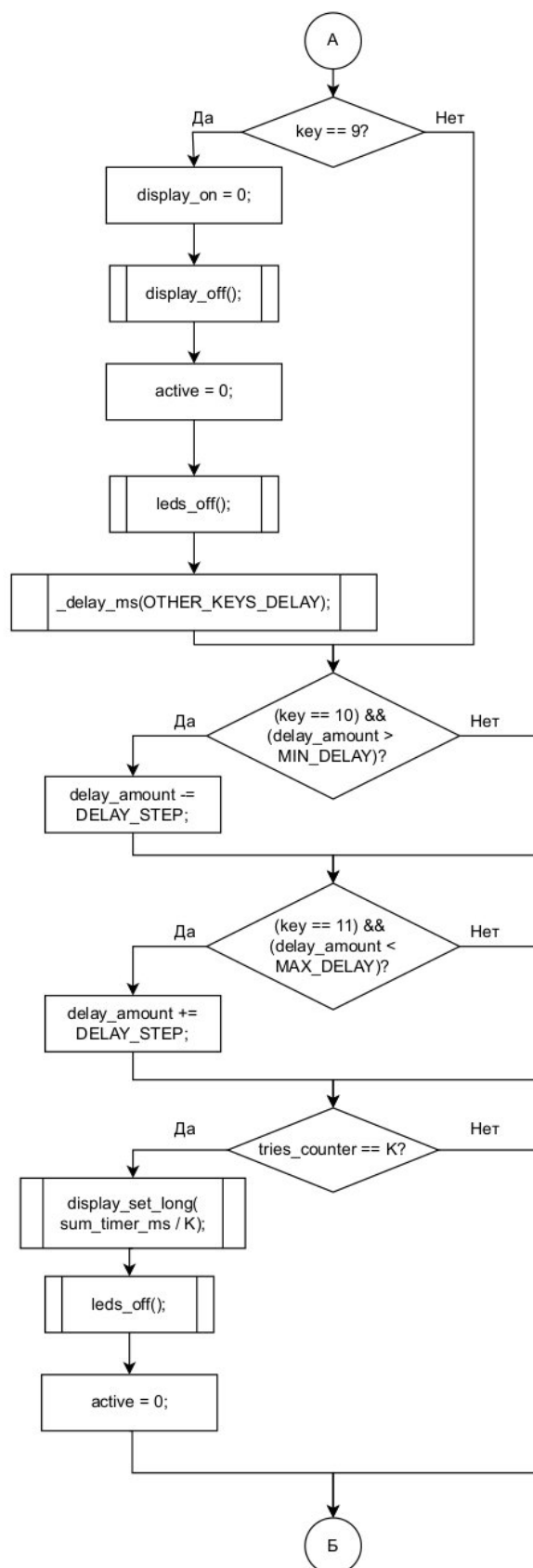
В режиме паузы светодиоды погашены. Если серия испытаний была пройдена до конца, можно отправить на ПЭВМ через драйвер UART-USB информацию о времени реакции на каждую мишень и о среднем времени реакции. Можно начать новое испытание кнопкой «Restart», переменные для подсчета времени и числа пройденных 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 T1);
- `tries_counter` — счетчик числа завершенных испытаний в серии;
- `void display_set_long(long target);` — вывести на дисплей значение, либо специальные символы, если значение не умещается;
- `char leds_random_line();` — «перемещает» мишень в другую строку и возвращает номер строки.



**Рисунок X** — алгоритм основной программы



**Рисунок X** — алгоритм основной программы

### 1.4.3 Управление динамической индикацией

Для отображения скорости реакции используется дисплей из четырех ССИ. На нем осуществляется динамическая индикация числа с переключением разрядов по переполнению таймера T0.

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

$$4\,000\,000 / (2^8 \cdot K) = 24 \cdot 4$$

$$K = 4\,000\,000 / (2^8 \cdot 24 \cdot 4) = 162.76$$

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 X** были установлены биты CS01 и CS0 регистра TCCR0.

**Таблица X** — выбор источника тактового сигнала для таймера/счётчика T0

CS02	CS01	CS00	Источник тактового сигнала
0	0	0	Таймер/счётчик остановлен
0	0	1	СК (тактовый сигнал микроконтроллера)
0	1	0	СК/8
0	1	1	СК/64
1	0	0	СК/256
1	0	1	СК/1024
1	1	0	Вывод T0, инкремент счётчика производится по спадающему фронту импульсов
1	1	1	Вывод T0, инкремент счётчика производится по нарастающему фронту импульсов

#### 1.4.4 Отсчет времени реакции

Отсчет времени реакции реакции производится за счет переменной `timer_ms`, инкрементируемой каждую миллисекунду по прерыванию таймера T1, 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  $K=1$ .  
Рассчитаем значение регистра сравнения OCR1A:

$$(F\_CPU / K) / OCR1A = \Delta t$$

$$\text{Где } \Delta t = 1\text{с} = 1000 \text{ мс}, K = 1$$

$$(F\_CPU / 1) / OCR1A = 1000$$

$$OCR1A = F\_CPU / 1000$$

Код обработчика прерываний T1 в режиме сравнения приведен в листинге X.

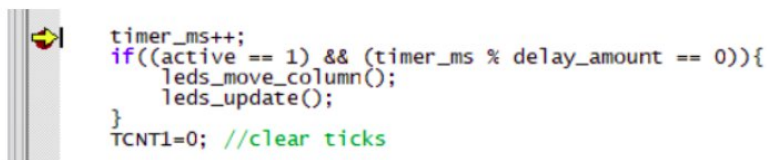
Листинг X — код обработчика прерываний T1 в режиме сравнения

```
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.

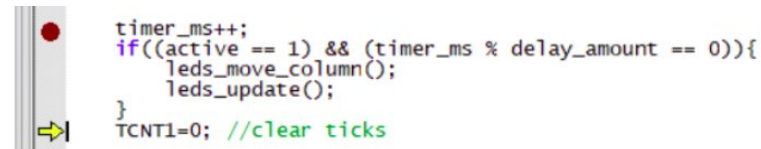
Данный процесс проиллюстрирован на рисунках X-X.

Z pointer	0x16D0
Cycle Counter	2888
Frequency	4.0000 MHz
Stop Watch	722.00 us
SREG	□□□□□□□□



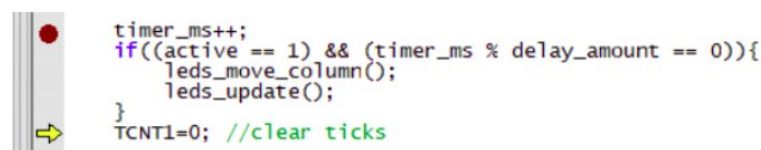
**Рисунок X** — начало обработки прерывания

Cycle Counter	2951
Frequency	4.0000 MHz
Stop Watch	737.75 us
SREG	□□□□□□□□



**Рисунок X** — конец обработки прерывания при несоблюдении условий ветвления

Z pointer	0x0000
Cycle Counter	3527
Frequency	4.0000 MHz
Stop Watch	881.75 us
SREG	□□□□□□□□



**Рисунок X** — конец обработки прерывания при соблюдении условий ветвления

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

$$t_{\text{false}} = (737.75 - 722.00) = 15.75 \text{ мкс}$$

$$t_{\text{true}} = (881.75 - 722.00) = 159.75 \text{ мкс}$$

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.

$$\delta t = (\Delta t / t) * 100\% = ( (99 * t_{\text{false}} + t_{\text{true}}) / t ) * 100\% = ( (99 * 15.75 + 159.75) / 100\,000 ) * 100\% = 1.719\%.$$

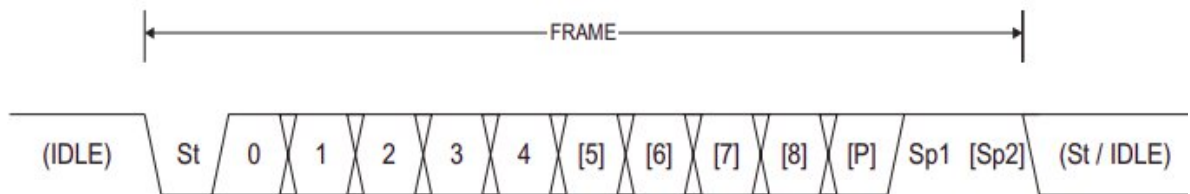
При скорости реакции в пределах 10 с (что крайне много) погрешность не превысит 18 мс. По сравнению с задержкой считывания с клавиатуры и временем выполнения цикла основной программы это число незначительно



### 1.4.5 Работа с UART

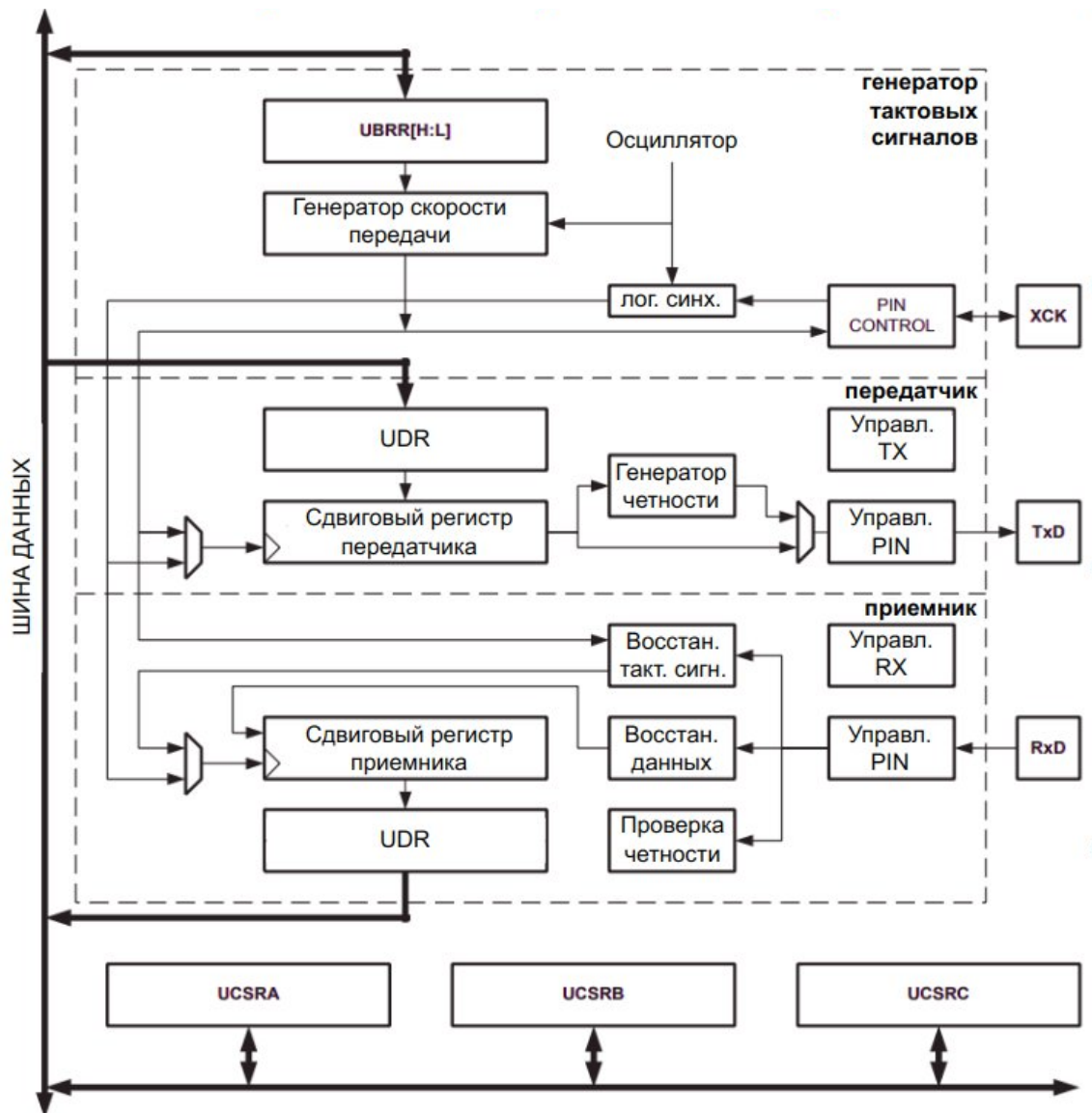
В качестве способа передачи информации на ПЭВМ в разработанном устройстве используются интерфейсы UART и USB, связь которых осуществляется через соответствующий драйвер.

Кадр UART имеет формат, приведенный на **рисунке X**.



**Рисунок X** — формат кадра UART

Структура модуля UART показана на **рисунке X**.



**Рисунок X** — структура модуля UART

St — стартовый бит (всегда на уровне логического 0);

0-8 — биты данных;

P — бит четности;

Sp1, Sp2 — стоповые биты (всегда на уровне логической 1);

IDLE — передачи данных не происходит (уровень логической 1 на линии).

Управление работой приёмопередатчика осуществляется с помощью регистров управления UCSRB и UCSRC. Текущее состояние приёмопередатчика определяется с помощью регистра состояния UCSRA. Формат этих регистров приведен в таблицах X, X, X.

**Таблица X** — Формат регистра управления UCSRB

№ разряда	7	6	5	4	3	2	1	0
Имя	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCZ2	RXB8	TXB8

Назначение управляющих битов:

RXCIE — разрешение прерывания по завершении приёма;

TXCIE — разрешение прерывания по завершении передачи;

UDRIE — разрешение прерывания при опустошении регистра данных UART;

RXEN — разрешение приёма;

TXEN — разрешение передачи;

UCZ2 — (вместе с UCZ0 и UCZ1) — количество полезных информационных битов;

RXB8 - 8-й разряд принимаемых данных.

TXB8 - 8-й разряд передаваемых данных.

**Таблица X** — Формат регистра управления UCSRC

№ разряда	7	6	5	4	3	2	1	0
Имя	URSEL	UMSEL	UPM1	UPM0	USBS	UCZ1	UCZ0	UCPOL

URSEL — выбор между UBRRH и UCSRC;

UMSEL — выбор синхронного режима работы;

UPM1 и UPM0 — режим проверки чётности;

USBS — определяет количество стоповых битов;

UCZ0 и UCZ1 (вместе с UCZ2) — количество полезных информационных битов;

UCPOL — выбор фронта синхросигнала (для синх. режима).

Выбор размера кадра осуществляется согласно **таблице X**.

**Таблица X** — выбор размера кадра

UCZ2	UCZ1	UCZ0	Размер кадра
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
100 — 110			зарезервировано
1	1	1	9 бит

**Таблица X** — Формат регистра состояния UCSRA

№ разряда	7	6	5	4	3	2	1	0
Имя	RXC	TXC	UDRE	FE	DOR	PE	U2X	MCPM

RXC — флаг завершения приёма. Данный флаг устанавливается в «1» при пересылке принятого слова из сдвигового регистра приёмника в регистр данных UDR. Сбрасывается флаг аппаратно при чтении регистра UDR;

TXC — флаг завершения передачи. Данный флаг устанавливается в «1» после передачи всех разрядов слова, включая стоп-бит, из сдвигового регистра передатчика, при условии, что в регистр данных UDR не было загружено новое значение.

UDRE – регистр данных пуст. Данный флаг устанавливается в «1» после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика. Установка этого флага означает, что передатчик готов к получению нового значения для передачи. Сбрасывается флаг аппаратно при записи в регистр UDR;

FE – флаг ошибки формата. Данный флаг устанавливается в «1», если стоп-бит принятого слова равен «0». Флаг сбрасывается при приёме стоп-бита, равного «1»;

DOR — флаг переполнения. Данный флаг устанавливается в «1», если в сдвиговом регистре приёмника находится новое принятое слово, а старое содержимое регистра UDR не прочитано. Флаг остаётся установленным до тех пор, пока не будет прочитано содержимое регистра UDR;

PE — флаг ошибки четности;

U2X — двояние скорости передачи;

MPCM — режим мультипроцессорного взаимодействия;

Управление скоростью приёма и передачи данных осуществляется контроллером скорости передачи, который является обыкновенным делителем частоты. Скорость передачи зависит от содержимого регистров UBRRH:UBRRL.

Скорость передачи определяется следующим выражением:

$$BAUD = f_{clk} / (16 * (UBRR + 1))$$

На основе описанного выше механизма настройки UART был написан программный драйвер, фрагмент исходного кода которого приведен в листинге X.

**Листинг X** — программный драйвер UART

```
include <avr/io.h>
#define BAUD 9600 /* частота 9600 */
#include <util/setbaud.h> /* упрощенная настройка частоты */

void uart_init() {
    UBRRH = UBRRH_VALUE; /* упрощенная настройка частоты */
    UBRRL = UBRRL_VALUE; /* упрощенная настройка частоты */

    #if USE_2X
        UCSRA |= (1 << U2X);
    #else
        UCSRA &= ~(1 << U2X);
    #endif

    UCSRC = (1 << UCSZ1) | (1 << UCSZ0); /* 8-битный кадр */
    UCSRB = (1 << RXEN) | (1 << TXEN); /* Разрешить получение и передачу */
}

void uart_send_byte(char c) {
    loop_until_bit_is_set(UCSRA, UDRE); /* Ждать опустошения регистра данных */
    UDR = c; /* Записать c в регистр данных (передать c на ПЭВМ).
}
```

## 2 Технологическая часть

### 2.1 Тестирование программы

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

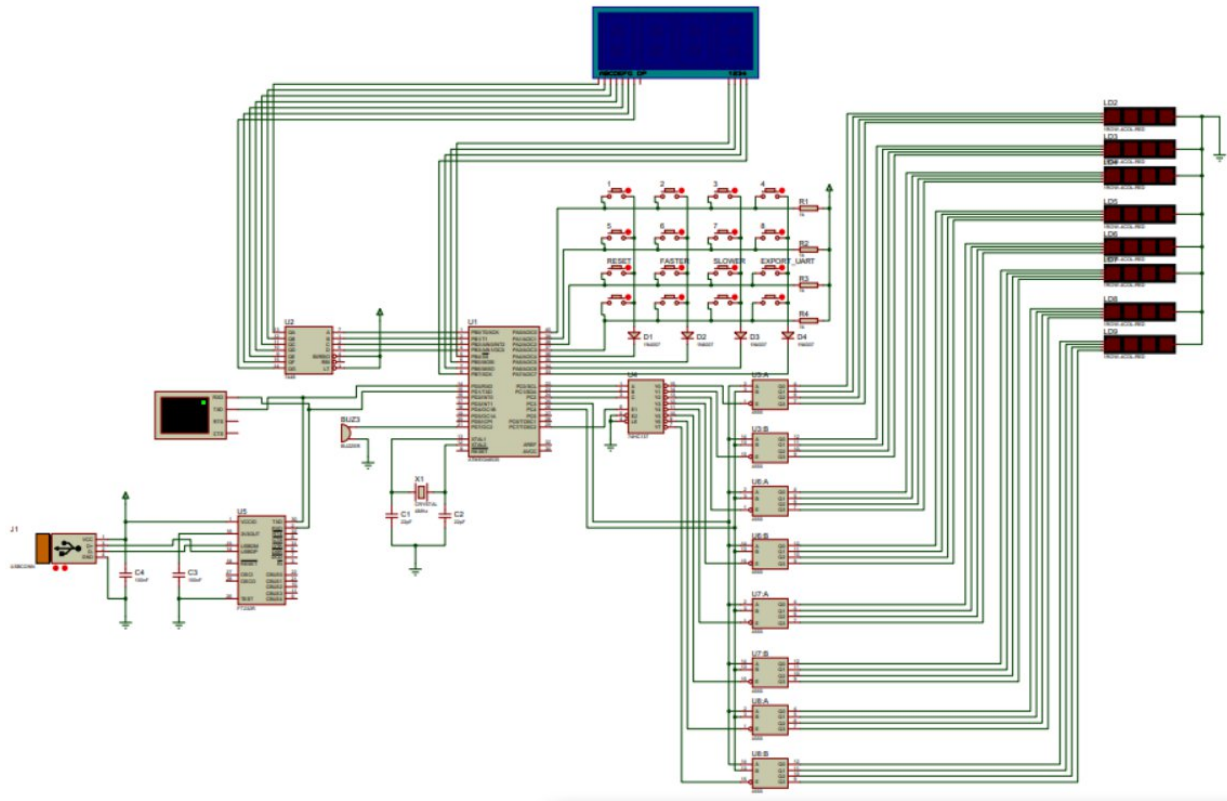


Рисунок X — упрощенная схема тренажера оператора в Proteus

Ввод информации систему осуществляется посредством матричной клавиатуры 4x4. Вывод информации осуществляется на светодиодную матрицу 8x4, 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 «Е» (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

X.



Рисунок X — отображение времени реакции

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



Рисунок X — состояние дисплея при превышении разрядности

Пример передачи данных на ПЭВМ показан на рисунке X. Для удобства тестирования было установлено число испытаний в серии  $K=3$ .

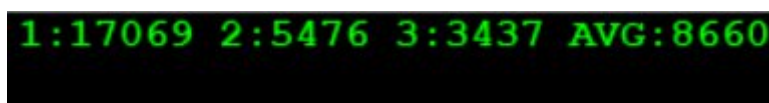


Рисунок X — передача данных по UART

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.

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы было спроектировано устройство для измерения времени реакции оператора на движущуюся мишень. Устройство может выводить информацию о результатах испытаний на дисплей из ССИ и пересылать эти данные на ПЭВМ через интерфейс USB.

Для написания исходного кода была использована среда AVR Studio 4, компилятор GCC, а так же библиотека AVR Libc. Кроме того, было произведено моделирование системы в среде Proteus 8.

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 ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

## **ПРИЛОЖЕНИЕ А**

### **Исходный текст программы**

**ПРИЛОЖЕНИЕ Б**  
**Функциональная электрическая схема**  
**Листов 1**

**ПРИЛОЖЕНИЕ В**  
**Принципиальная электрическая схема**  
**Листов 1**

**ПРИЛОЖЕНИЕ Д**  
**Перечень элементов**  
**Листов X**