|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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:***

***\_\_\_\_\_Электронный калькулятор\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

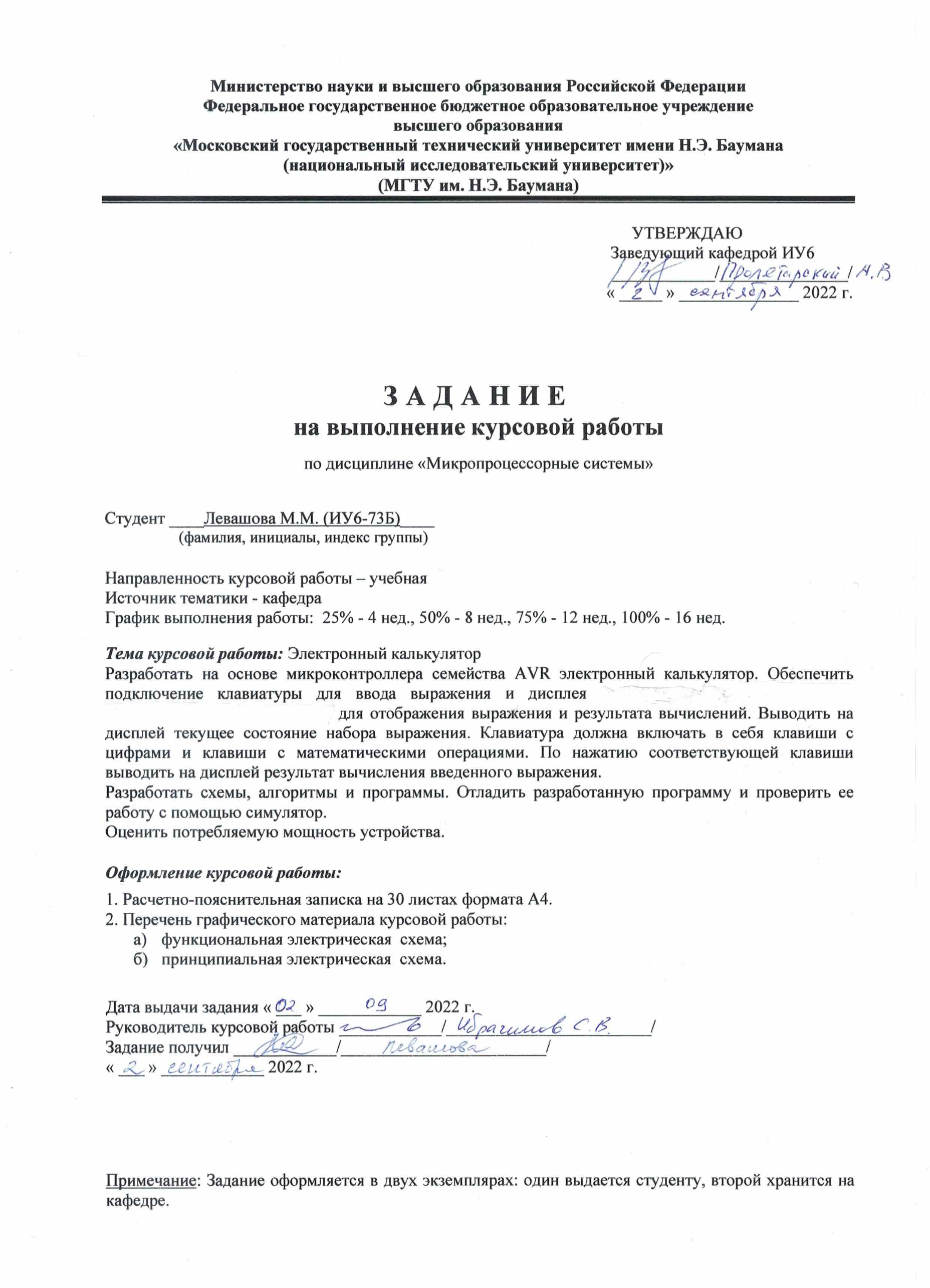
***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-73Б |  |  | М.М. Левашова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Руководитель |  |  |  | С.В. Ибрагимов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |

*2022 г.*



РЕФЕРАТ

РПЗ 46 страниц, 14 рисунков, 9 таблиц, 10 источников, 3 приложения.

МИКРОКОНТРОЛЛЕР, КАЛЬКУЛЯТОР, ATMEGA8515, USART, ДИСПЛЕЙ, КЛАВИАТУРА.

Объектом разработки является электронный калькулятор.

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.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ТЗ – техническое задание

МК – микроконтроллер

УГО – условное графическое обозначение

ATmega8515 – используемый микроконтроллер

USART – (англ. Universal Synchronous receiver/transmitter) Универсальный синхронный приемопередатчик

Datasheet – документация на устройство

Proteus 8 – среда моделирования

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc121476889)

[1 КОНСТРУКТОРСКАЯ ЧАСТЬ 7](#_Toc121476890)

[1.1 Анализ требований и принцип работы системы 7](#_Toc121476891)

[1.2 Проектирование функциональной схемы 8](#_Toc121476892)

[1.2.1 Микроконтроллер АТmega8515 8](#_Toc121476893)

[1.2.2 Клавиатура 16](#_Toc121476894)

[1.2.3 Передача данных в ПЭВМ 16](#_Toc121476895)

[1.2.4 Настройка канала передачи 18](#_Toc121476896)

[1.2.5 Дисплей 19](#_Toc121476897)

[1.2.6 Генератор тактовых импульсов и сброс 20](#_Toc121476898)

[1.2.7 Построение функциональной схемы 21](#_Toc121476899)

[1.3 Проектирование принципиальной схемы 21](#_Toc121476900)

[1.3.1 Разъем программатора 21](#_Toc121476901)

[1.3.2 Подключение цепи питания 23](#_Toc121476902)

[1.3.3 Расчет потребляемой мощности 23](#_Toc121476903)

[1.3.4 Построение принципиальной схемы 24](#_Toc121476904)

[1.4 Алгоритмы работы системы 25](#_Toc121476905)

[1.4.1 Main и подпрограмма вычислений 25](#_Toc121476906)

[1.4.2 Используемые при работе подпрограммы 28](#_Toc121476907)

[2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 29](#_Toc121476908)

[2.1 Отладка и тестирование системы 29](#_Toc121476909)

[2.2 Симуляция работы системы 29](#_Toc121476910)

[2.3 Программирование микроконтроллера 30](#_Toc121476911)

[ЗАКЛЮЧЕНИЕ 33](#_Toc121476912)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 34](#_Toc121476913)

[ПРИЛОЖЕНИЕ А 35](#_Toc121476914)

[ПРИЛОЖЕНИЕ Б 44](#_Toc121476915)

[ПРИЛОЖЕНИЕ В 46](#_Toc121476916)

# ВВЕДЕНИЕ

В данной работе производится разработка на основе микроконтроллера AVR ATmega8515 электронного калькулятора, позволяющего выполнять базовые арифметические операции (сложение, вычитание, умножение, деление), операции для нахождения целой части от деления и остатка от деления, вычисление процента от числа. Также предусмотрена возможность использовать полученный ответ в следующем вычислении, передавать ответ по протоколу UART в компьютер, очищать, включать и выключать дисплей. Калькулятор работает с вещественными числами.

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.

Система состоит из МК и клавиатуры, дисплея и блока передачи данных.

Проектирование контроллера с заданными функциями состоит из двух основных частей: конструкторская часть и технологическая часть.

Конструкторская часть включает в себя:

* проектирование структурно-функциональной схемы;
* описание архитектуры, используемого микроконтроллера и описание назначения функциональных элементов схемы;
* описание принципиальной электрической схемы МК-системы с обоснованием выбора используемых радиоэлементов;
* описание алгоритмов функционирования МК-системы;
* расчет потребляемой мощности устройства.

Технологическая часть включает в себя:

* характеристику использованных систем разработки и отладки программ;
* тестирование и отладку программы;
* описание и моделирование работы системы;
* описание способа программирования МК.

# 1 КОНСТРУКТОРСКАЯ ЧАСТЬ

## **1.1 Анализ требований и принцип работы системы**

Согласно техническому заданию, необходимо разработать на основе микроконтроллера семейства AVR электронный калькулятор. Обеспечить подключение клавиатуры для ввода выражения и дисплея для отображения выражения и результата вычислений. Выводить на дисплей текущее состояние набора выражения. Клавиатура должна включать в себя клавиши с цифрами и клавиши с математическими операциями. По нажатию соответствующей клавиши выводить на дисплей результат вычисления введенного выражения.

Клавиатура калькулятора состоит из 23 клавиш, их обозначения и функции представлены в таблице 1.

Таблица 1 – Функции клавиш клавиатуры калькулятора

|  |  |
| --- | --- |
| Клавиши | Функция |
| «0» … «9» | Цифры для ввода числа |
| «.» | Точка для ввода вещественных чисел |
| «+», «-», «\*», «/» | Арифметические операции двух переменных |
| «=» | «Равно» выводит результат вычисления введенного выражения на дисплей |
| «MOD» | Остаток от деления |
| «DIV» | Целочисленное деление |
| «%» | Процент от числа (первый аргумент – сам процент, второй - число) |
| «ANS» | Результат последнего вычисления используется вместо аргумента текущей операции (для второго аргумента) |
| «AC» | Очищение дисплея |
| «ON/OFF» | Включение / выключение дисплея |
| «UART» | Передача результата вычисления в компьютер |

Для реализации данного функционала устройство должно содержать следующие структурные блоки [1]:

* микроконтроллер;
* программатор;
* клавиатура;
* дисплей;
* блок передачи данных из ПЭВМ.

Разработанная структурная схема устройства представлена на рисунке 1.

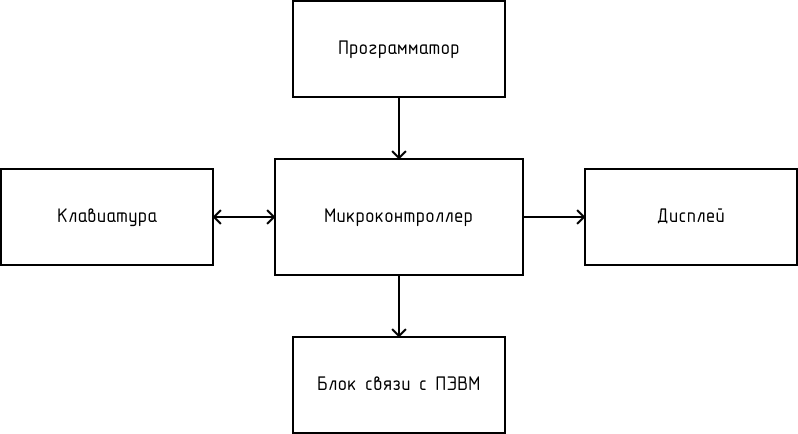


Рисунок 1 – Структурная схема устройства

## **1.2 Проектирование функциональной схемы**

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

### **1.2.1 Микроконтроллер АТmega8515**

Микроконтроллер является основным элементом разрабатываемого устройства. В соответствие с техническим заданием для реализации системы необходимо использовать микроконтроллер AVR.

Микроконтроллеры AVR делятся на семейства:

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 С) 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 С. [2]

В подсемействе MegaAVR семейства AVR был выбран МК – ATmega8515, обладающий всем необходимым функционалом для реализации проекта:

* интерфейс SPI для программирования МК;
* интерфейс UART для обмена данными;
* 512 байт ОЗУ;
* 1 8-разрядный счетчик;
* 8 Кбайт FLASH памяти;
* 3 внешних источника прерываний;
* частота работы до 16 МГц;

В разрабатываемом устройстве используется микроконтроллер в прямоугольном пластиковом корпусе с 40 выводами (40-pin PDIP). Конфигурация выводов микроконтроллера показана на рисунке 2.

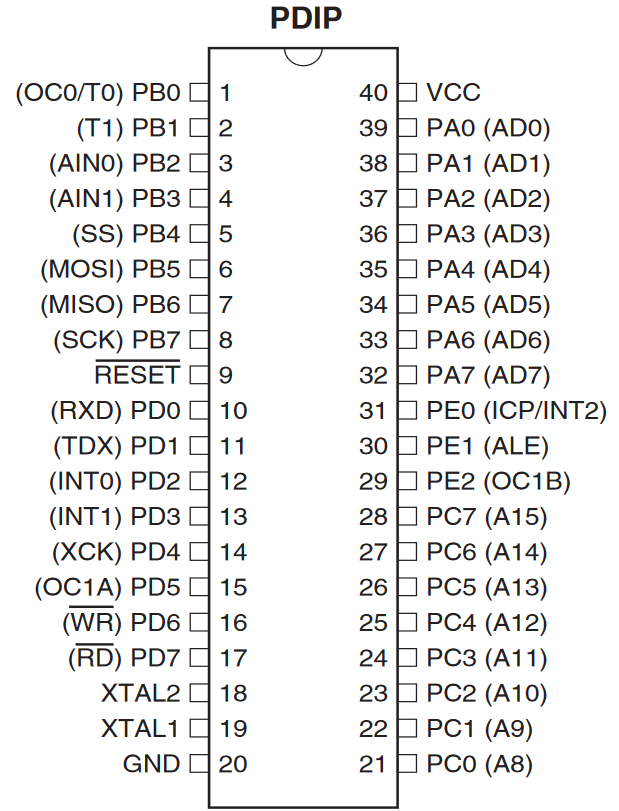


Рисунок 2 – Конфигурация выводов Atmega8515

Для подачи питания и связи с внешними цепями микросхема МК имеет 40 контактов, в том числе 32 контакта, которые могут быть сконфигурированы для реализации конкретных задач и требований разработчика.

ATmega8515 — это 8-разрядный КМОП-микроконтроллер с низким энергопотреблением, основанный на улучшенной RISC-архитектуре AVR. Выполняя мощные инструкции за один такт, ATmega8515 достигает пропускной способности, приближающейся к 1 MIPS на МГц, что позволяет разработчику системы оптимизировать энергопотребление в зависимости от скорости обработки.

Структурная схема микроконтроллера представлена на рисунке 3.

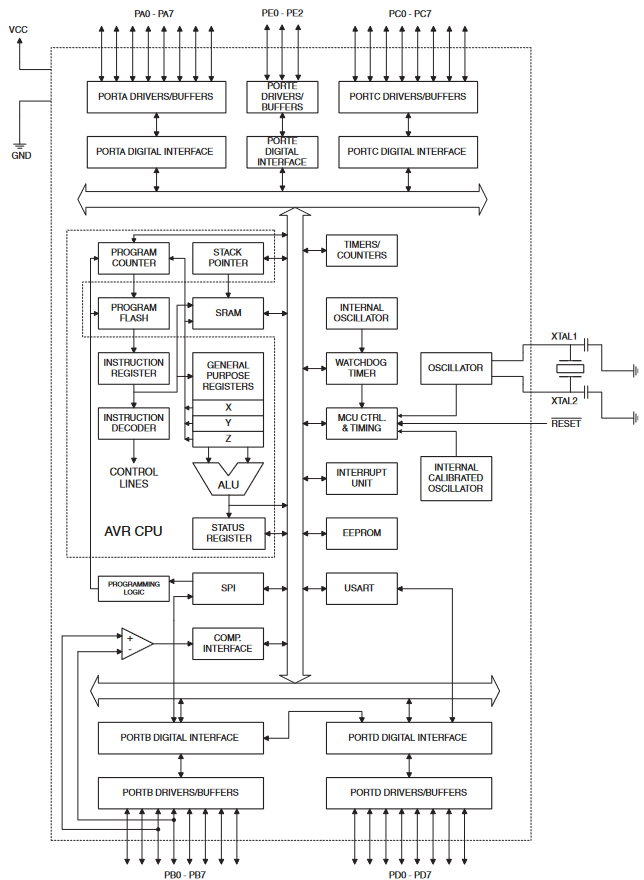


Рисунок 3 – Структурная схема ATmega8515

Ядро AVR сочетает в себе богатый набор инструкций с 32 рабочими регистрами общего назначения. Все 32 регистра напрямую подключены к арифметико-логическому устройству (АЛУ), что позволяет получить доступ к двум независимым регистрам в одной инструкции, выполняемой за один такт. Полученная в результате архитектура более эффективна в коде, обеспечивая при этом пропускную способность в десять раз выше, чем у обычных микроконтроллеров CISC.

ATmega8515 обеспечивает следующие функции: 8 КБ внутрисистемной программируемой флэш-памяти с возможностью чтения во время записи, 512 байт EEPROM, 512 байт SRAM, интерфейс внешней памяти, 35 линий ввода-вывода общего назначения, 32 рабочих регистра общего назначения, два гибких таймера/счетчика с режимами сравнения, внутренними и внешними прерываниями, последовательный программируемый USART, программируемый сторожевой таймер с внутренним генератором, последовательный порт SPI и три программно выбираемых режима энергосбережения. [3]

**1.2.1.1 Используемые элементы**

Для функционирования электронного калькулятора в МК ATmega8515 задействованы не все элементы его архитектуры. Выделим и опишем те, что используются во время функционирования схемы.

Порты А, В, C, D – назначения каждого из них описано в разделе 1.2.1.2.

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.

Декодер – блок, выделяющий код операции и операнды команды, а затем вызывающий микропрограмму, которая исполняет данную команду.

Сигналы управления – синхронизируют обработку данных.

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

Управление синхронизацией и сбросом (MCU CTRL. & Timing) – в этом блоке обрабатываются тактовые сигналы и принимается сигнал сброса.

UART – через этот интерфейс из МК передается информация в ПЭВМ. В регистр UART информация попадает через порт PD1 (TxD).

Генератор – генератор тактовых импульсов. Необходим для синхронизации работы МК. [1]

**1.2.1.2 Распределение портов**

МК ATmega8515 содержит пять портов – А, В, С, D и Е. Опишем назначение тех, что используются в данной системе для её функционирования.

Порт А:

РА0..PA1 – вывод информации на линии данных дисплея.

Порт В:

РВ5 – MOSI (SPI Master Output/Slave Input);

РВ6 – MISO (SPI Master Input/Slave Output);

РВ7 – SCK (SPI Serial Clock).

Порт C:

PC0…PC3 – подключены на вывод к строкам клавиатуры для установки сигнала проверки строки;

PC4 – RS – выбор регистра: 0 – команды, 1 – данные;

PC5 – RW – выбор операции: 1 – чтение, 0 – запись;

PC6 – E – разрешение на выполнение операции, по заднему фронту на этом выводе происходит запись в дисплей.

Порт D:

РD0 – RXD, Receive Data (Data input pin for USART) – вход приемника;

PD1 – TXD, Transmit Data (Data output pin for USART) – выход передатчика;

PD2…PD6 – подключены на ввод к столбцам клавиатуры для считывания зажатой кнопки.

**1.2.1.3 Организация памяти**

Структура памяти МК ATmega8515 приведена на рисунке 4.

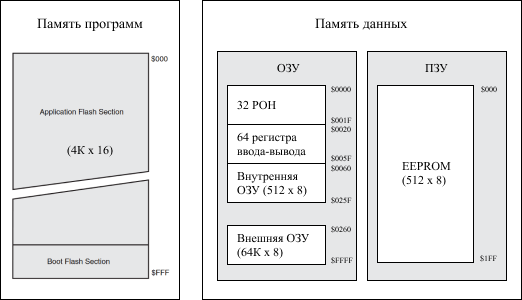


Рисунок 4 – Структура памяти

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

Все AVR имеют Flash-память программ, которая может быть различного размера - от 1 до 256 КБайт. В МК Atmega8515 объем Flash-памяти программ 4Кх16 (длина команды 16 разрядов, следовательно каждая команда находится в одной ячейке памяти). Информацию в flash-память можно мгновенно стереть и записать. Заносится с помощью программатора.

Память построена на принципе электрической перепрограммируемости, т. е. допускает многократное стирание и запись информации.

Память данных разделена на три части:

1. регистровая память: 32 регистра общего назначения (РОН) и служебные регистры ввода/вывода;

Структура 32 регистров общего назначения представлена на рисунке 5.

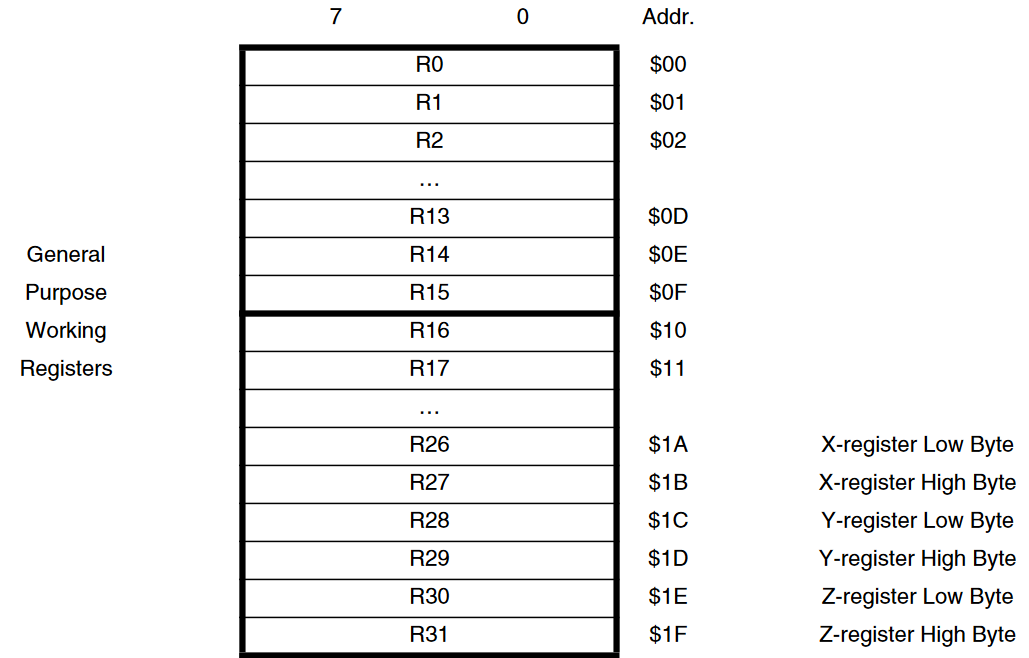


Рисунок 5 – Структура регистров общего назначения

1. оперативная память (ОЗУ или RAM) – МК ATmega8515 имеет объем внутреннего SRAM 512 байт (с адреса $0060 до $025F). Число циклов чтения и записи в RAM не ограничено, но при отключении питания вся информация теряется. Для МК ATmega8515 возможна организация подключения внешнего статического ОЗУ объемом до 64К;
2. энергозависимая память (ЭСППЗУ или EEPROM) – эта память доступна МК в ходе выполнения, для хранения промежуточных результатов. В МК ATmega8515 ее объем 512 байт. Также в неё могут быть загружены данные через программатор. [3]

### **1.2.2 Клавиатура**

Клавиатура состоит из 23 клавиш: 10 клавиш с цифрами («0»..«9»), клавиша «.», 7 клавиш с операторами («+», «-», «\*», «.», «%», «div», «mod»), а также клавиша включения-выключения («ON/OFF»), очищения дисплей («AC») и передачи результата в компьютер («UART»).

Клавиатура – матричная , то есть каждая кнопка, при нажатии на нее, замыкает контакты между конкретным столбцом и конкретным рядом, создавая цепь, которую можно программно обнаружить. Для реализации 23 клавиш потребуется 10 ножек контроллера (4 строки и 6 столбцов).

### **1.2.3 Передача данных в ПЭВМ**

Передача данных в 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 из выхода TxD микроконтроллера.

К внешнему устройству MAX232 подключен через разъем DB-9. На схеме условное обозначение – ХS1.

Внутреннее изображение MAX232 показано на рисунке 6.

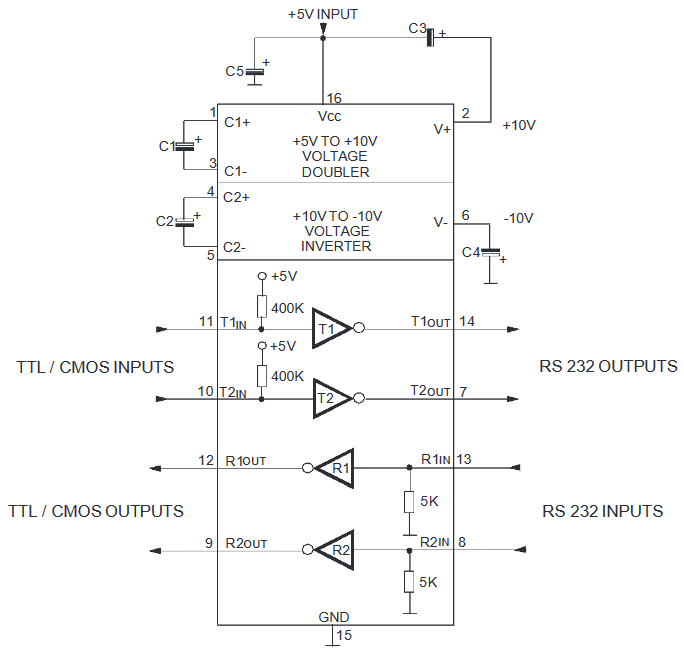


Рисунок 6 – Преобразователь MAX232

Микросхема-драйвер фирмы Maxim серии MAX232 используется в качестве преобразователя уровней напряжения при связи устройства с ПЭВМ. Она содержит преобразователь напряжения +5В в напряжение +10В, инвертор, преобразующий напряжение +10В в напряжение -10В, и преобразователь уровней сигналов последовательного интерфейса. [4]

### **1.2.4 Настройка канала передачи**

UART (универсальный асинхронный приёмопередатчик) – одна из старейших и самых распространенных на сегодняшний день технологий передачи данных. Слово «асинхронный» означает, что интерфейс не использует линию для синхросигнала, приемник и передатчик заранее настраиваются на одну частоту.

Для корректной работы необходимо настроить регистры управления UCSRA, UCSRB и UCSRC.

UCR (UCSRB) – регистр управления. Биты регистра UCSRB показаны в таблице 2. Во время работы системы, для управления передачей информации, будет использоваться бит TXEN.

TXEN – разрешение передачи. Если разряд сбросится во время передачи, выключение передатчика произойдет только после завершения передачи.

Таблица 2 – Биты регистра UCSRB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Название | RXCIE | TXCIE | UDRIE | RXEN | TXEN | CHR9 | RXB8 | TXB8 |

USR (UCSRA) – регистр состояния. Биты регистра UCSRA показаны в таблице 3. Во время работы системы используются биты TXC и UDRE.

TXC – флаг завершения передачи. Устанавливается в 1 при завершении передачи.

UDRE – флаг опустошения регистра данных. Устанавливается в 1 при пустом буфере передатчика. Используется при выводе данных из МК. Когда USRE = 1, то есть регистр UDR пуст, в него можно загружать новые данные для передачи.

Таблица 3 – Биты регистра UCRSA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Название | RXC | TXC | UDRE | FE | OR | - | - | - |

UCSRС. Биты регистра UCSRС показаны в таблице 4.

Таблица 4 – Биты регистра UCSRC

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Название | RXCIE | TXCIE | UDRIE | RXEN | TXEN | UCSZ | RXB8 | TXB8 |
| 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.

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. [1]

Настройка скорости обмена USART вычисляется по формуле:

где XTAL – частота работы МК (1 МГц), baudrate – требуемая скорость передачи (2400 бод).

.

### **1.2.5 Дисплей**

Среди множества различных LCD дисплеев наиболее распространенными являются текстовые дисплеи на основе контроллера hd44780, которые стали стандартом на символьные ЖКИ.

Контроллер LCD hd44780 имеет встроенный знакосинтезатор: для вывода символа необходимо записать его код в соответствующую область (DDRAM) памяти контроллера. Также есть возможность загружать в память знакосинтезатора (CGRAM) произвольные символы. Контроллер обладает параллельным интерфейсом, имеет 4 или 8 линии данных, три управляющих линии.

ЖК-дисплей 16x2 означает, что есть две строки, в каждой строке которых может отображаться 16 символов, и каждый символ занимает пространство матрицы 5X7 на ЖК-дисплее.

14 контактов можно разделить на пять категорий: контакты питания, контрастный контакт, контакты управления и контакты данных. 14-пиновый интерфейс дисплеев на HD44780 представлен в таблице 5. [5]

Таблица 5 – Выводы дисплея HD44780

|  |  |  |
| --- | --- | --- |
| Номер пина | Название | Функция |
| 1 | VSS | Земля, общий провод, GND |
| 2 | VCC | Напряжение питания (+5 V) |
| 3 | Vo | Настройка контрастности |
| 4 | R/S | Выбор регистра |
| 5 | R/W | Чтение/запись |
| 6 | (Enable) | Строб по спаду |
| 7 | Bit 0 | младший для 8-битного интерфейса |
| 8 | Bit 1 |  |
| 9 | Bit 2 |  |
| 10 | Bit 3 |  |
| 11 | Bit 4 | младший для 4-битного интерфейса |
| 12 | Bit 5 |  |
| 13 | Bit 6 |  |
| 14 | Bit 7 | старший для 8-(4-)битного интерфейса |

### **1.2.6 Генератор тактовых импульсов и сброс**

Для работы МК необходим тактовый генератор. Внутри МК есть собственный, но для надежности подключим внешний резонатор. Он подключается к внутреннему через входы XTAL1 и XTAL2. Частота резонатора равна 1 МГц для соответствующей тактовой частоты МК.

В AVR есть внутренняя схема сброса, и сигнал RESET изнутри уже подтянут резистором 10кОм к Vcc. Чтобы сброс мог происходить вручную, подключим выход RESET к питанию, через резистор в 10кОм. При включении схемы напряжение на RESET равняется логической единице – МК запускается.

### **1.2.7 Построение функциональной схемы**

На основе всех вышеописанных сведений была спроектирована функциональная схема разрабатываемой системы, показанная на рисунке 7 и в приложении Б [6-8].

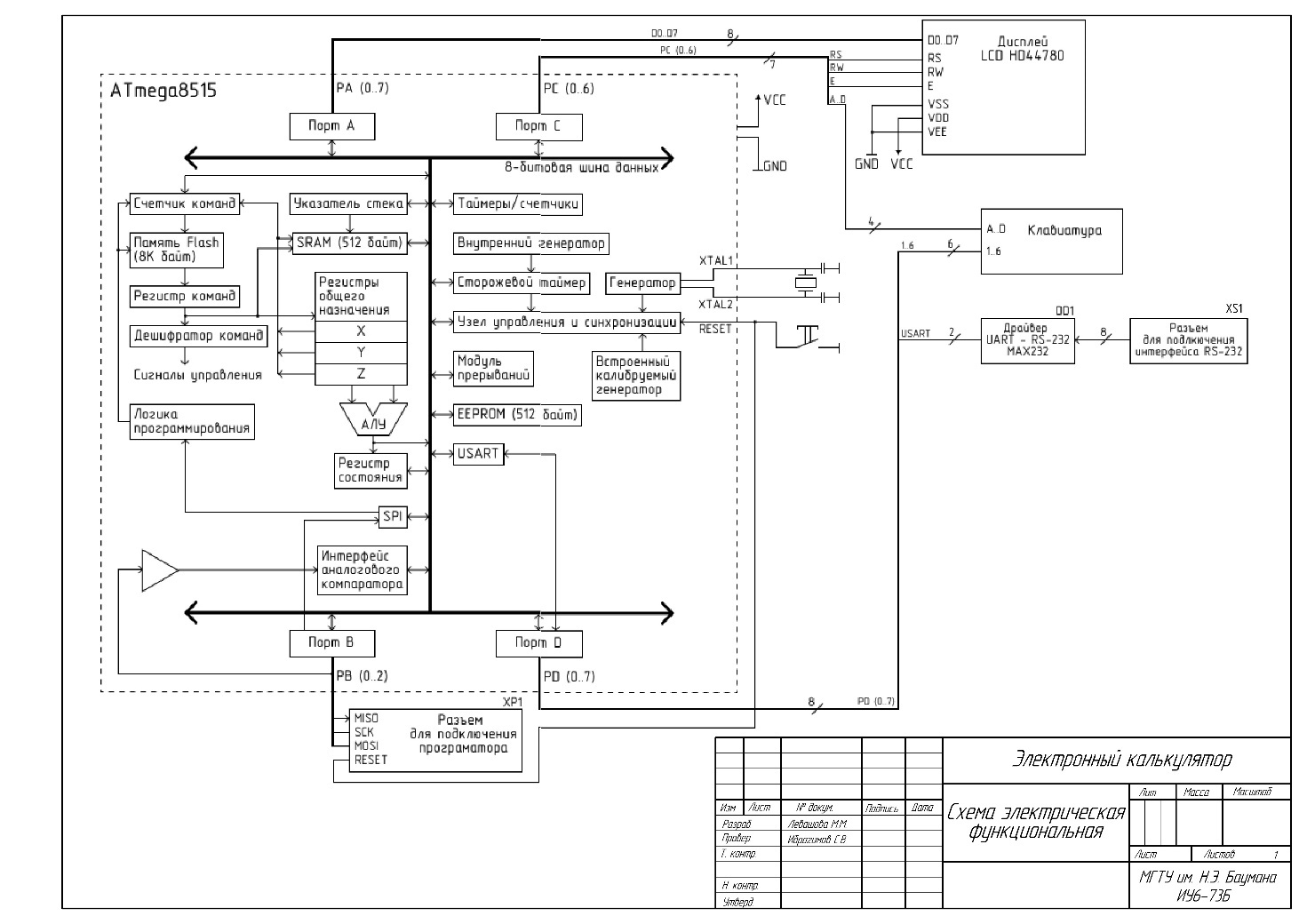


Рисунок 7 – Функциональная схема

## **1.3 Проектирование принципиальной схемы**

### **1.3.1 Разъем программатора**

Для программирования МК используется программатор, для его подключения необходим специальный разъем. Будет использован разъем IDC-06MS. Расположение выводов вилки представлено на рисунке 8, выводы разъема описаны в табоице 6. Подключение программатора осуществляется при помощи интерфейса SPI, под что на МК ATmega8515 задействован порт РВ. На принципиальной схеме, которая показана в разделе 1.3.5, условное обозначение – ХР1.

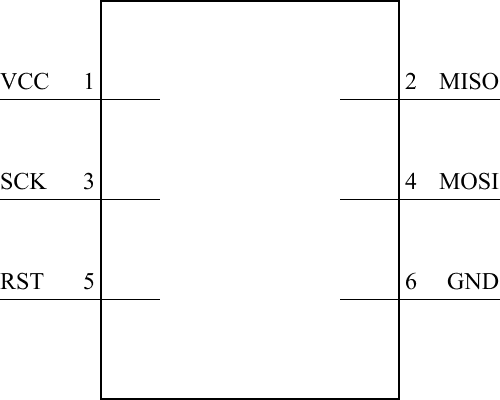


Рисунок 8 – Расположение выводов IDC-06MS

Таблица 6 – Выводы IDC-06MS

|  |  |  |
| --- | --- | --- |
| Номер вывода | Название | Функция |
| 1 | VCC | Питание. |
| 2 | MISO | Передача данных от микроконтроллера в программатор. |
| 3 | SCK | Тактовый сигнал. |
| 4 | MOSI | Передача данных от программатора в микроконтроллер. |
| 5 | RST | Сигналом на RESET программатор вводит контроллер в режим программирования. |
| 6 | GND | Земля. |

### **1.3.2 Подключение цепи питания**

Для питания основных функциональных блоков разрабатываемого устройства необходимы различные напряжения: 5В и 12В. Поэтому в разрабатываемой схеме был использован стабилизатор напряжения LM7805, который преобразует входные 12 В в 5 В, а также двухконтактная розетка DS-201 для подачи 12 В.

LM7805 – стабилизатор напряжения с интегральной схемой с фиксированным напряжением. У стабилизатора 3 вывода, на первый вывод подается входное напряжение (+12В), второй вывод – это земля, а третий вывод — это выходное напряжение (+5В).

### **1.3.3 Расчет потребляемой мощности**

Для определения потребляемой мощности МК воспользуемся формулой:

Где, I – ток потребления МК равный 10 мА (взято из даташита), U – напряжение питания равное 5В. Рассчитаем потребляемую мощность МК:

Также в схеме присутствуют резисторы. Мощность для резисторов рассчитывается по формуле:

где R – сопротивление резистора;

I – ток, проходящий через резистор.

Далее рассчитаем суммарную потребляемую мощность остальных устройств разрабатываемой системы (таблица 7).

Таблица 7 – Потребляемая мощность

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Микросхема | Ток потребления, мА | Потребляемая мощность, мВт | Количество устройств | Суммарная потребляемая мощность |
| MAX232 | 8 | 40 | 1 | 40 |

Продолжение таблицы 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HD44780 | 1 | 5 | 1 | 5 |
| LM7805 | 5 | 25 | 1 | 25 |
| R1, R2 (10к) | - | 1,25 | 2 | 2,5 |

### **1.3.4 Построение принципиальной схемы**

На основе всех вышеописанных сведений была спроектирована принципиальная схема разрабатываемой системы, показанная на рисунке 9 и в приложении Б [6-8].

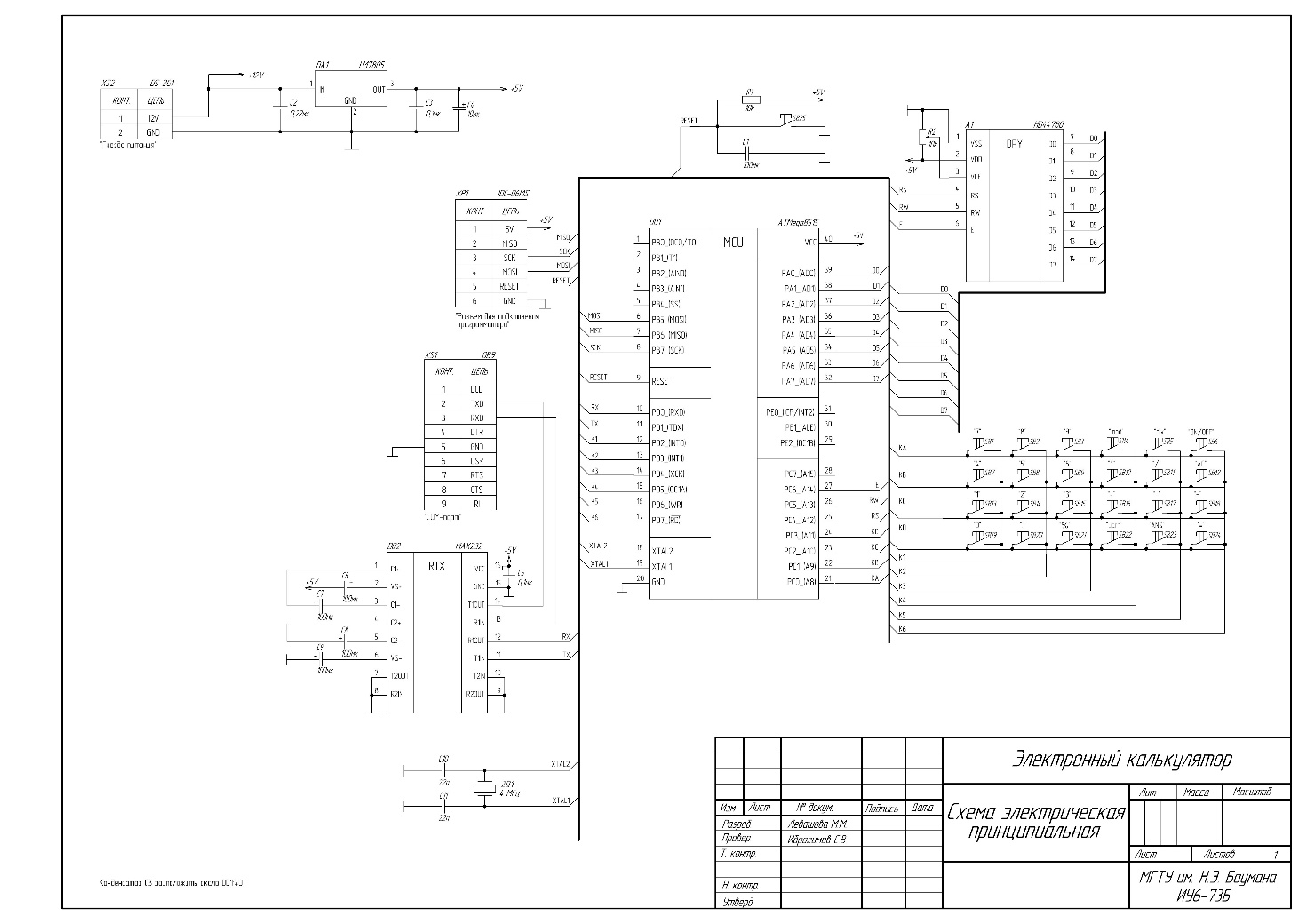


Рисунок 9 – Принципиальная схема

## **1.4 Алгоритмы работы системы**

### **1.4.1 Main и подпрограмма вычислений**

Работа начинается с функции main, из которой вызывается функция инициализации калькулятора. Эта функция в свою очередь вызывает функции инициализации дисплея, клавиатуры и модуля передачи UART и запускает работу калькулятора. Схема алгоритма главной функции и функции инициализации калькулятора показана на рисунке 10.

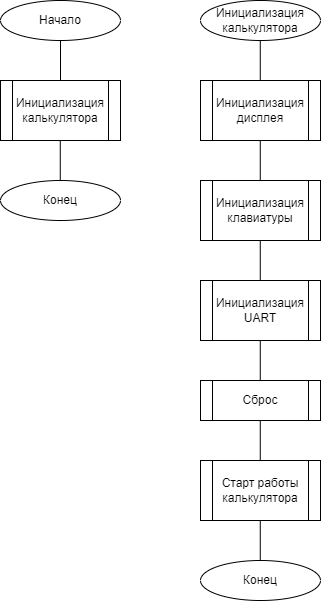


Рисунок 10 – Схема алгоритма

Для представления выполнения ввода и вычисления выражения используется конечный автомат. Конечный автомат можно представить в виде графа, вершины которого являются состояниями, а ребра — переходы между ними. Описание состояний автомата представлено на рисунке 11 и в таблице 8. Каждое ребро имеет метку, информирующую о том, когда должен произойти переход. Например, на рисунке видно, что автомат сменит состояние «start\_disp» на состояние «a\_input» при условии ввода цифры. Схема алгоритма подпрограммы вычислений представлена на рисунке 12.

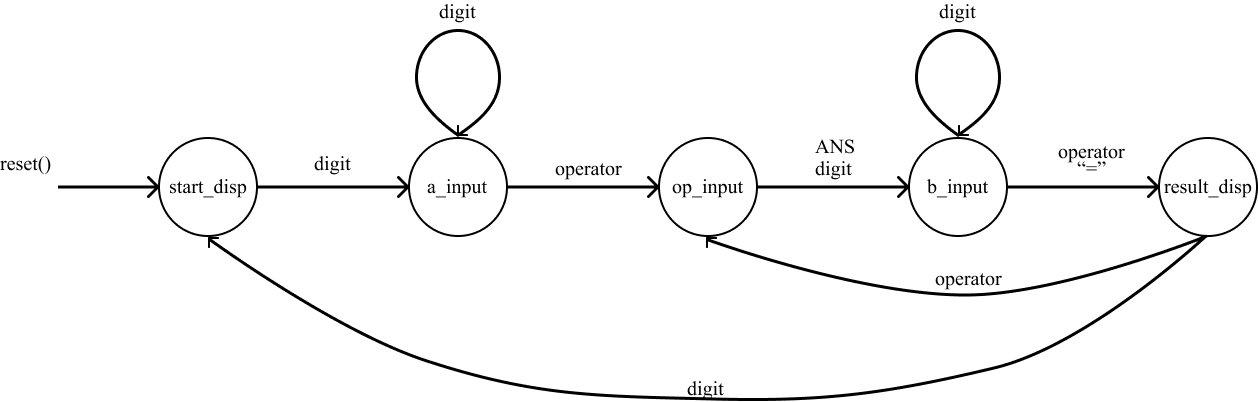


Рисунок 11 – Описание состояний автомата

Таблица 8 – Состояния автомата

|  |  |  |
| --- | --- | --- |
| Номер состояния | Обозначение | Описание |
| 0 | start\_disp | Начало работы. На дисплее ничего нет. |
| 1 | a\_input | Ввод первого числа. |
| 2 | op\_input | Ввод оператора. |
| 3 | b\_input | Ввод второго числа. |
| 4 | result\_disp | Отображение результата вычисления. |



Рисунок 12 – Схема алгоритма подпрограммы вычислений

### **1.4.2 Используемые при работе подпрограммы**

Программа состоит из 4 подпрограмм в соответствии с функциональными блоками:

1. Главная подпрограмма;
2. Подпрограмма калькулятора;
3. Подпрограмма дисплея;
4. Подпрограмма клавиатуры.

Главная подпрограмма и подпрограмма калькулятора описаны выше.

Описание функций подпрограммы дисплея представлено в таблице 9.

Таблица 9 – Функции подпрограммы дисплея

|  |  |
| --- | --- |
| Имя функции | Действия |
| init\_display | Инициализация дисплея |
| send\_command | Запись байта (символа) команды в lcd контроллер |
| send\_data | Запись байта (символа) данных в lcd контроллер |
| trigger | Генерирование строба на линии E. По этому стробу производится запись команды/данных или чтение. |
| send\_string | Вывод строки |
| move\_to | Изменение текущей позиции курсора |
| clear | Очищение дисплея |
| disp\_on\_off | Включение-выключение дисплея |

В подпрограмме клавиатуры две функции: инициализации и сканирования клавиши. В подпрограмме UART так же две функции: инициализации и передачи данных в ПЭВМ.

# 2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

Для реализации работы модели перекрестка была написана программа на языке Си, после загруженная в МК. Симуляция проводилась в программе Proteus 8.

## **2.1 Отладка и тестирование системы**

Программа была отлажена с использованием программы Proteus 8. Это многофункциональная программа для автоматизированного проектирования электронных схем.

При написании кода были использованы следующие библиотеки:

* avr/io.h – в этом файле находятся определения констант, имен регистров и прочего для выполнения базового ввода-вывода;
* util/delay.h – это библиотека, позволяющая вызывать задержки (delay). При вызове \_delay\_ms в качестве параметра передается время в миллисекундах. В программе используется после чтения данных, перед их выводом, чтобы МК успел обработать полученную информацию;
* stdbool.h;
* string.h;
* stdlib.h.

По итогу отладки и тестирования, результатом стала функционирующая модель системы перекрестка, работающая в соответствии с ТЗ. Симуляция системы описана в разделе 2.2.

## **2.2 Симуляция работы системы**

Данный проект был промоделирован в Proteus. Были протестированы следующие программные модули:

* ввод данных с клавиатуры;
* вычисление математических операций;
* отображение информации на дисплее;
* передача данных в ПЭВМ.

Для моделирования передачи данных в ПЭВМ используется инструмент системы – Virtual Terminal. Он позволяет эмулировать простейший терминал, который даёт возможность передавать данные по порту TxD через интерфейс UART.

На рисунке 13 изображена упрощенная принципиальная схема устройства в симуляторе Proteus.

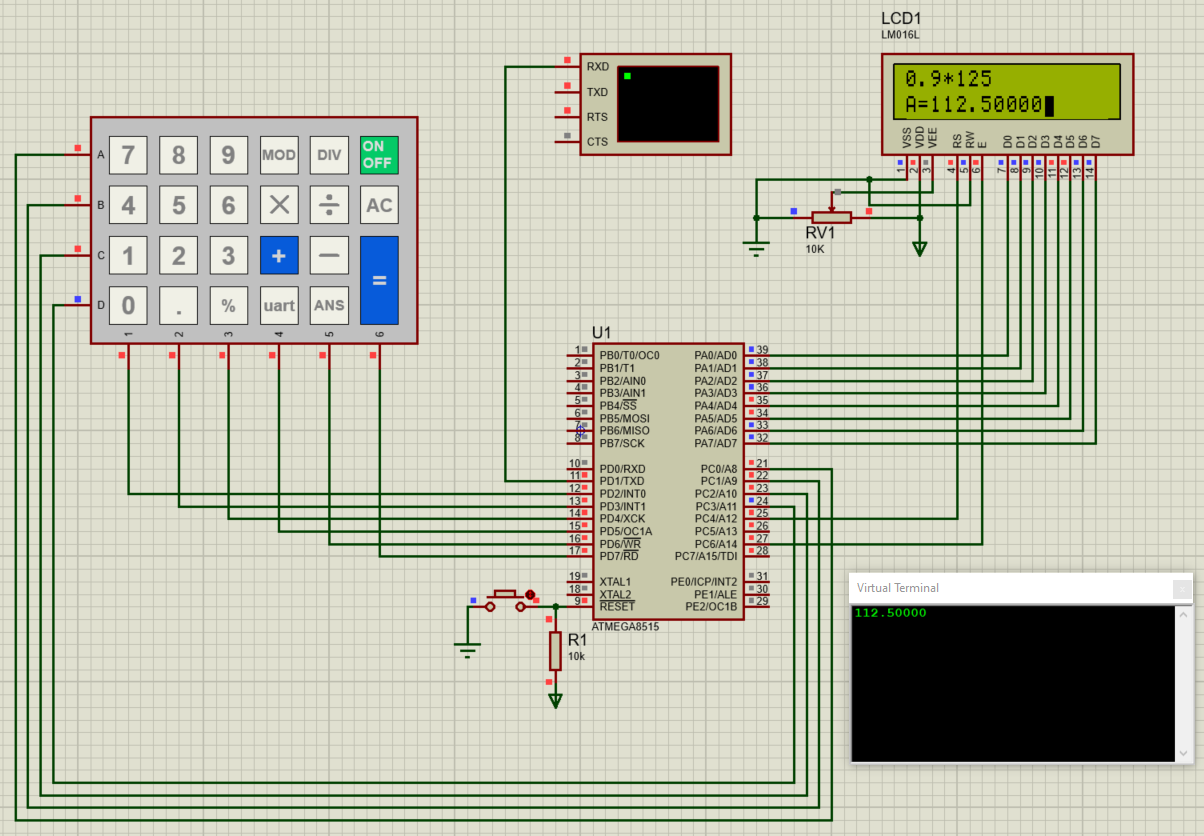


Рисунок 13 – Схема устройства в симуляторе Proteus

## **2.3 Программирование микроконтроллера**

Программирование МК через канал SPI происходит через программатор и одноименный разъем, о котором было рассказано в разделе 1.3.1.

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.

При программировании AVR программатор всегда функционирует как ведущее устройство, а микроконтроллер как ведомое. SPI является синхронным интерфейсом: все операции синхронизированы фронтами тактового сигнала (SCK), который вырабатывается ведущим устройством.

Передача по SPI осуществляется в полнодуплексном режиме, 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.

В МК передается бинарный файл с расширением “.hex” с скомпилированной программой.

Для работы с модулем SPI используются три регистра:

* SPDR (SPI Data Register) – регистр данных. В этот регистр заносится байт для последующей его передачи и из него же считывается пришедший байт информации;
* SPSR (SPI Status Register) – статусный регистр. Предназначен для контроля состояния SPI модуля, содержит дополнительный бит управления скоростью обмена;
* SPCR (SPI Control Register) – управляющий регистр. С помощью данного регистра устанавливается конфигурация модуля SPI.

Программирование микроконтроллера по SPI осуществляется путем посылки 4-байтовых команд на вывод MOSI МК, в который один или два байта определяют тип операции, остальные – адрес, записываемый байт, установочные биты и биты защиты, пустой байт. При выполнении операции чтения считываемый байт снимается через вывод MISO. Так же можно запрограммировать память данных EEPROM. В каждой команде указывается адрес записываемой ячейки и записываемое значение.

Форматы байтов команд для программирования микроконтроллера ATmega8515 представлены на рисунке 14.

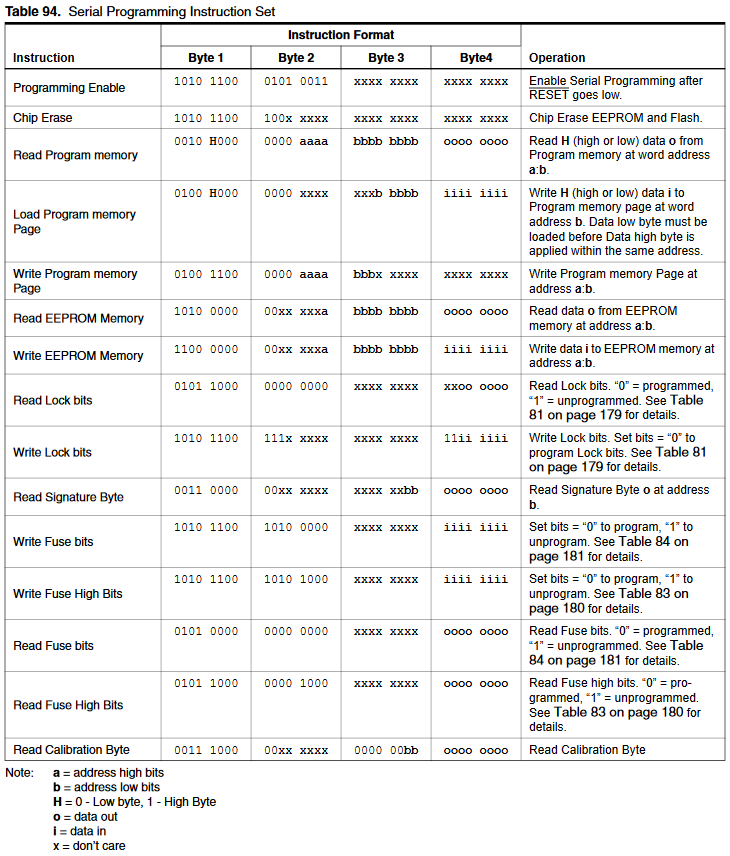


Рисунок 14 – Набор инструкций последовательного программирования через SPI

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был спроектировано и реализован электронный на основе микроконтроллера Atmega8515. Устройство позволяет выполнять базовые арифметические операции (сложение, вычитание, умножение, деление), операции для нахождения целой части от деления и остатка от деления, вычисление процента от числа.

В результате проектирования были разработаны принципиальная и функциональная электрические схемы для аппаратной части устройства. Также были разработаны коды модулей для программы на языке Си.

Разработанное устройство удовлетворяет требованиям, предъявленным в задании на курсовую работу, и может использоваться для произведения вычислений.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Хартов, В.Я. Микропроцессорные системы: учеб. пособие для студ. учреждений высш. проф. образования, Академия, М., 2014. – 368с.

2. Хартов, В.Я. Микроконтроллеры AVR. Практикум для начинающих: 2-е издание, Издательство МГТУ им. Н.Э. Баумана, 2012. – 278с.

3. ATmega8515 Datasheet [Электронный ресурс]. – Режим доступа: <https://static.chipdip.ru/lib/059/DOC000059786.pdf> (дата обращения: 10.12.2022)

4. Документация на драйвер MAX232 [Электронный ресурс]. – URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/397176/MAXIM/MAX232.html> (дата обращения 10.12.2022).

5. HD44780U Datasheet [Электронный ресурс]. – Режим доступа: <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf> (дата обращения: 10.12.2022)

6. ГОСТ 2.702-2011 Правила выполнения электрических схем

7. ГОСТ 2.710-81 Обозначения буквенно-цифровые в электрических схемах

8. ГОСТ 2.721-74 Обозначения условные графические в схемах. Обозначения общего применения

9. ГОСТ 2.102-68 ЕСКД. Виды и комплектность конструкторских документов

10. ГОСТ 2.105-95 ЕСКД. Текстовые документы

# ПРИЛОЖЕНИЕ А

**Текст программы**

**Main.c**

# define F\_CPU 1000000UL

#include "calculator.h"

int main(void){

init\_calculator();

return 0;

}

**Calculator.h**

#ifndef CALCULATOR\_H

#define CALCULATOR\_H

#include <stdbool.h>

void init\_calculator(void);

void run(void);

void reset(void);

double calculate(float, char, float);

void show\_result(void);

void decide(unsigned char, bool\*, int\*);

void send\_digit(unsigned char);

void uart(void);

float add\_digit(float, unsigned char, bool\*, int\*);

#endif

**Calculator.c**

#include <stdlib.h>

#include <util/delay.h>

#include "calculator.h"

#include "keyboard.h"

#include "lcd.h"

#include "uart.h"

enum exp\_states

{

start\_disp, //0 Displaying banner.

a\_input, //1 Input first operand, a.

op\_input, //2 Middle Operator pressed, op.

b\_input, //3 Input second operand, b.

result\_disp, //4 Result displayed

};

float r, a, b; // a and b are operands, r is the result.

float ans; // for "ANS" operator

int count; // Number of characters displaying on first line.

enum exp\_states state; // Calculator and Monitor state.

unsigned char op; // Operator: [/, \*, +, -, ...].

static char line[] = " "; // Single line, 16 characters long.

static char buffer[16];

static char overflow[] = " Overflow ";

/////////////////////////////////////////////////////////////////////////////////////////////

void init\_calculator() {

\_delay\_ms(500);

init\_display();

init\_keyboard();

UsartInit();

reset();

run();

}

void reset() {

clear();

move\_to(0, 0);

r = a = b = op = count = 0;

state = start\_disp;

}

void run() {

bool \*is\_dec\_p;

bool is\_decimal = 0;

is\_dec\_p = &is\_decimal;

int \*paw\_p;

int paw = 1;

paw\_p = &paw;

while (1) {

decide(scan\_key(), is\_dec\_p, paw\_p);

}

}

void send\_digit(unsigned char digit) {

send\_data(digit + '0');

}

double calculate(float m, char operator, float n) {

switch (operator) {

case '+':

return r = m + n;

case '-':

return r = m - n;

case '\*':

return r = m \* n;

case '/':

return r = m / n;

case 'p'://%

return r = m \* n / 100;

case 'm'://mod

return r = (int)m % (int)n;

case 'v'://div

return r = (int)m / (int)n;

}

return r = m;

}

void show\_result() {

move\_to(0, 1); //bottom line start

send\_string("A=");

char\* data = buffer;

if (r <= 999999999999)

dtostrf(r, 9, 5, buffer); //float -> char\* (r -> buffer)

else

data = overflow;

send\_string(data);

state = result\_disp;

}

void uart() {

char uart\_ans[64];

dtostrf(ans, 9, 5, uart\_ans); //float -> char\* (ans -> uart\_ans)

Transmit(uart\_ans);

}

void decide(unsigned char key, bool\* is\_dec\_p, int\* paw\_p) {

// Check if it is a digit.

if (key >= '0' && key <= '9')

{

unsigned char digit = key - '0'; // '2' --> 2

switch (state) {

case start\_disp:

if (digit) {

state = a\_input;

}

case a\_input:

if (count == 14) return; // You cannot fillup the screen with a single operand.

if (\*is\_dec\_p == 0) {

a = a \* 10 + digit; // append to a

}

if (\*is\_dec\_p == 1) {

int d = 1;

for (int i=0; i<\*paw\_p; i++) {

d = d \* 10;

}

a = (a \* d + digit) / d;

\*paw\_p = \*paw\_p + 1;

}

send\_data(key);

count++;

break;

case op\_input:

if (digit) {

state = b\_input;

}

case b\_input:

if (\*is\_dec\_p == 0) {

b = b \* 10 + digit; // append to b

}

if (\*is\_dec\_p == 1) {

int d = 1;

for (int i=0; i<\*paw\_p; i++) {

d = d \* 10;

}

b = (b \* d + digit) / d;

\*paw\_p = \*paw\_p + 1;

}

send\_data(key);

count++;

break;

case result\_disp:

if (digit) {

line[0] = ' ';

line[1] = ' ';

move\_to(0, 0); // top string

send\_string(line); // Clear 1st line.

move\_to(0, 0);

a = b = op = count = 0;

state = a\_input;

decide(key, is\_dec\_p, paw\_p); // Recursively capture digit.

return;

}

break;

}

}

else if (key == '.') {

\*is\_dec\_p = 1;

send\_data(key);

}

else {

\*is\_dec\_p = 0;

\*paw\_p = 1;

switch (key) {

case '/':

case '\*':

case '+':

case '-':

case 'p':

case 'm':

case 'v':

switch (state) {

case op\_input:

move\_to(0,0); // Modify the operand displayed.

case a\_input:

send\_data(key);

count++;

state = op\_input;

break;

case b\_input:

calculate(a, op, b);

show\_result();

ans = calculate(a, op, b);

case result\_disp:

move\_to(0, 0);

line[0] = 'A'; // 'A' represents current result.

line[1] = key; // Operator of the operation.

send\_string(line);

move\_to(2,0);

count = 2;

a = r; // Put result into a.

b = 0; // Clear b.

state = op\_input;

break;

}

op = key;

break;

case '=':

if(state == b\_input){

calculate(a, op, b);

show\_result();

ans = calculate(a, op, b);

}

break;

case 'C': // "C" button.

reset();

break;

case 'A': // "ANS" button.

send\_data('A');

b = ans;

state = b\_input;

break;

case 'O':// "ON/OFF" button.

disp\_on\_off();

break;

case 'u':// "uart" button.

uart();

break;

}

}

}

**Lcd.h**

#ifndef LCD\_H

#define LCD\_H

void init\_display(void);

void send\_command(unsigned char);

void send\_data(unsigned char);

void send\_string(const char\*);

void clear(void);

void disp\_on\_off(void);

void move\_to(unsigned char, unsigned char);

void trigger(void);

#endif

**Lcd.c**

#include <avr/io.h>

#include <util/delay.h>

#include <stdbool.h>

#include "lcd.h"

bool on\_flag;

void init\_display(){

DDRC |= (1<<PC4)|(1<<PC6); // PC 4 is output for RS, 6 is output for Enable.

DDRA = 0xFF; // PA 0-7 is 8 bit data output bus.

PORTC &=~(1<<PC6); // Reset PC6 (E).

\_delay\_ms(15);

send\_command(0x38); // 2 line mode.

\_delay\_ms(5);

send\_command(0x38); // 2 line mode.

\_delay\_ms(100);

send\_command(0x38); // 2 line mode.

send\_command(0x0F); // LCD ON, cursor ON

\_delay\_ms(10);

send\_command(0x01); // Clear display screen

\_delay\_ms(10);

send\_command(0x81);

\_delay\_ms(10);

PORTC |= (1<<PC6); // Set PC6 (E).

on\_flag = 1;

}

void send\_command(unsigned char command){

PORTA = command;

trigger();

\_delay\_us( 40 );

}

void send\_data(unsigned char data){

PORTA = data;

trigger();

\_delay\_us( 40 );

}

void trigger(){

PORTC |= (1<<PC6); // Set PC6 (E).

\_delay\_us(1);

PORTC &=~(1<<PC6); // Reset PC6 (E).

\_delay\_us(1);

}

//

void send\_string(const char \*str){

while(\*str) send\_data(\*str++); // Send characters one by one.

}

void move\_to(unsigned char x, unsigned char y){

unsigned char address = x;

if ( y ) {

address += 0x40;

}

PORTC &=~(1<<PC4); // Reset PC4 (RS).

send\_command(1<<7 | address);

PORTC |=(1<<PC4); // Set PC4 (RS).

}

void clear(){

PORTC &=~(1<<PC4); // Reset PC4 (RS).

send\_command(0x01);

PORTC |=(1<<PC4); // Set PC4 (RS).

}

void disp\_on\_off(){

if (on\_flag == 1) {

PORTC &=~(1<<PC4); // Reset PC4 (RS).

send\_command(0x08);

PORTC |=(1<<PC4); // Set PC4 (RS).

on\_flag = 0;

}

else {

PORTC &=~(1<<PC4); // Reset PC4 (RS).

send\_command(0x0F); // LCD ON, cursor ON

\_delay\_ms(10);

send\_command(0x01); // Clear display screen

\_delay\_ms(10);

send\_command(0x81);

\_delay\_ms(10);

PORTC |=(1<<PC4); // Set PC4 (RS).

on\_flag = 1;

}

}

**Keyboard.h**

#ifndef KEYBOARD\_H

#define KEYBOARD\_H

void init\_keyboard(void);

unsigned char scan\_key(void);

#endif

**Keyboard.c**

#include <avr/io.h>

#include <util/delay.h>

#include "keyboard.h"

unsigned char keys[4][6] = {

{'7', '8', '9', 'm', 'v', 'O'},

{'4', '5', '6', '\*', '/', 'C'},

{'1', '2', '3', '+', '-', '='},

{'0', '.', 'p', 'u', 'A', '='}

};

void init\_keyboard() {

DDRC |= 0x0F; // PB 0-3 output for rows.

DDRD &= 0x03; // PC 2-7 input for columns.

}

unsigned char scan\_key() {

while (1) {

PORTD |= 0xFC;

for(int i = 0;i<4;i++){

PORTC = ~(1 << i);

for(int j = 2;j < 8;j++){

if(bit\_is\_clear(PIND, j)){

while(bit\_is\_clear(PIND, j)) \_delay\_ms(50);

return keys[i][j-2];

}

}

}

}

}

**Uart.h**

#ifndef UART\_H

#define UART\_H

void UsartInit(void);

void Transmit(char\* data);

#endif

**Uart.c**

#include <avr/io.h>

#include <util/delay.h>

#include <string.h>

#include "uart.h"

void UsartInit()

{

UBRRL=25; //1 000 000 / (2400 \* 16) - 1 = 25

UCSRB=(1<<TXEN);

UCSRC=(1<<URSEL)|(1<<UCSZ0|(1<<UCSZ1));

}

void Transmit(char\* data)

{

for (int i=0; i<strlen(data); i++)

{

while(!(UCSRA&(1<<UDRE))) {} // wait ready of port

UDR = data[i];

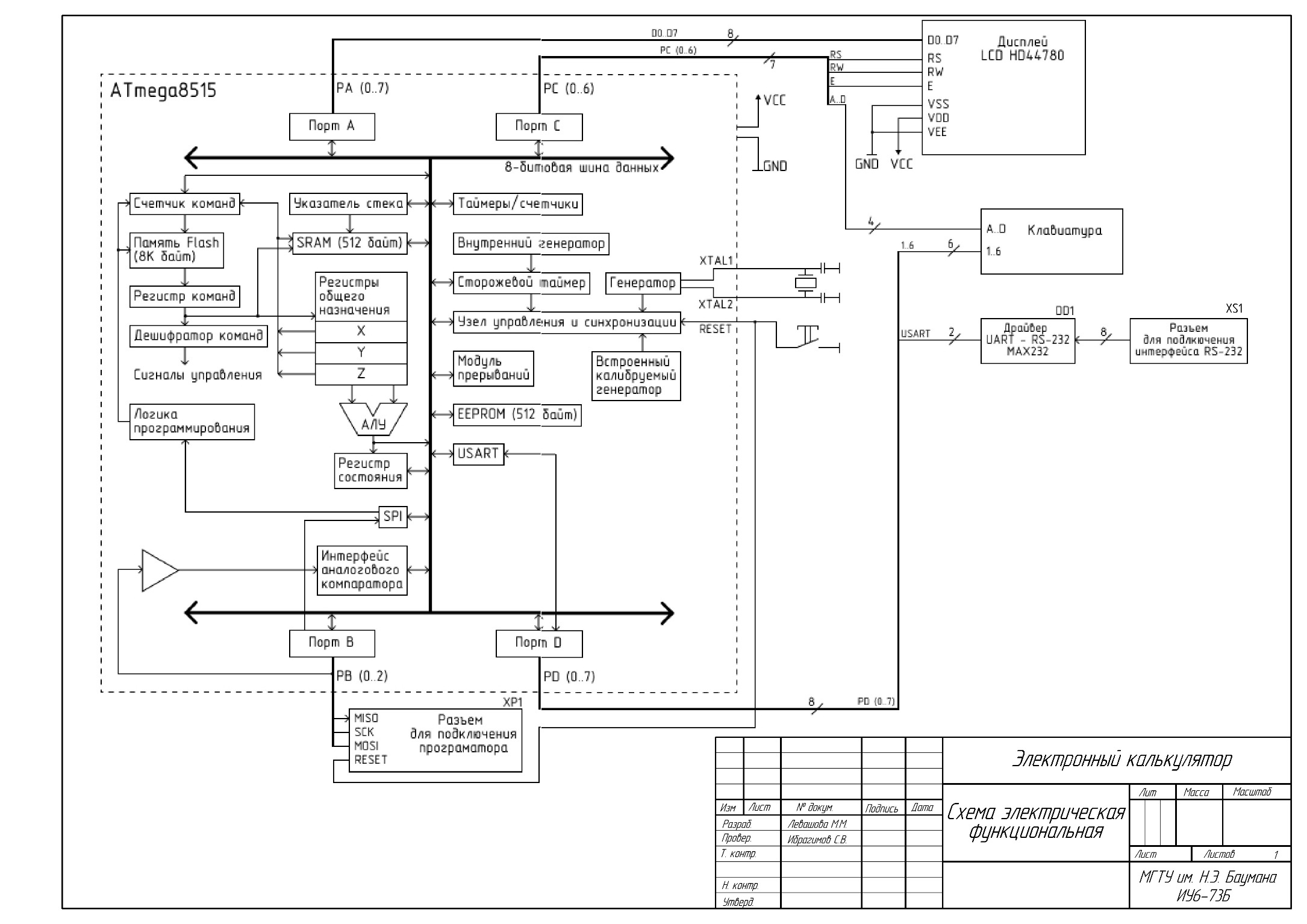
}

}

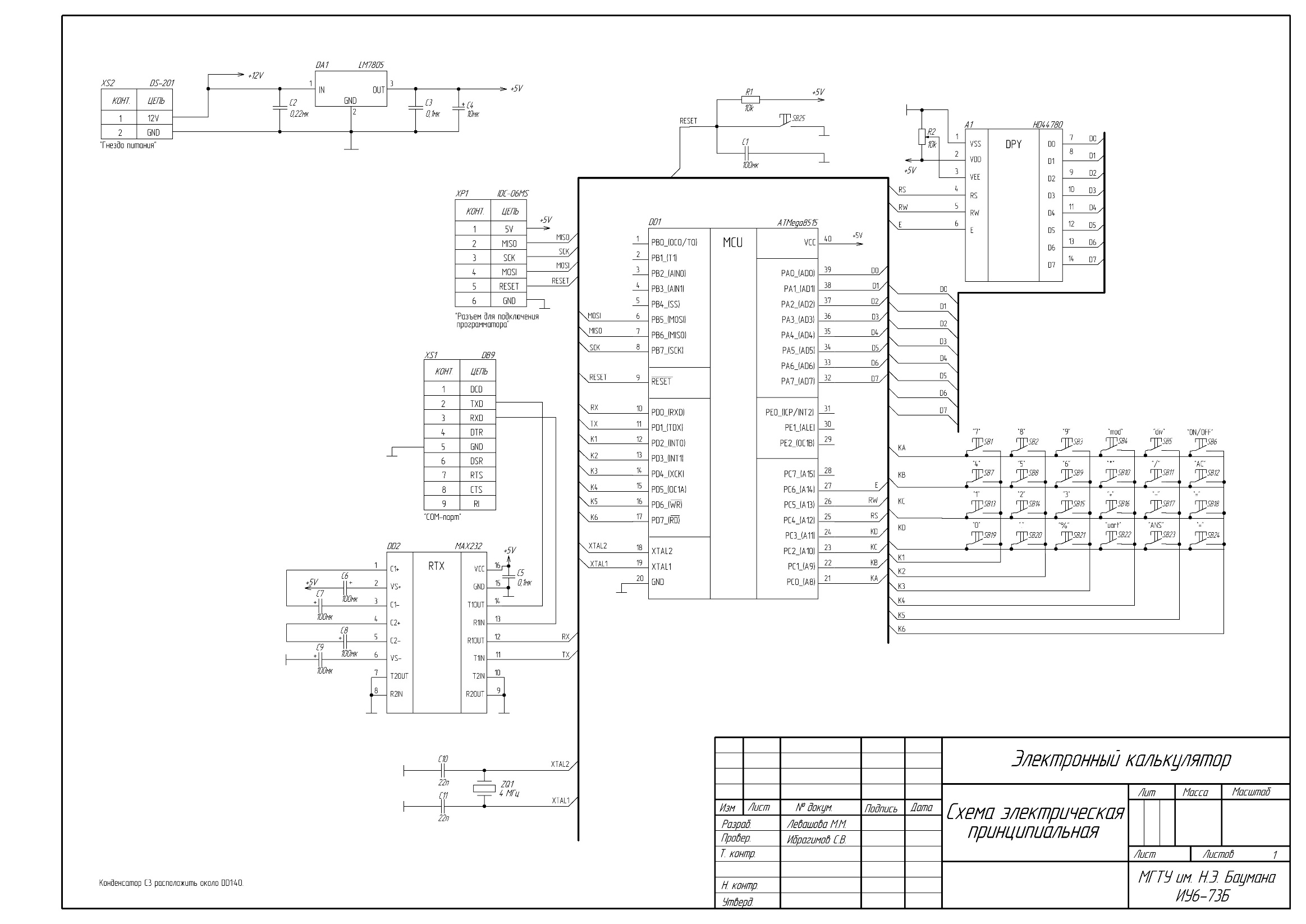
# ПРИЛОЖЕНИЕ Б

**Графическая часть**

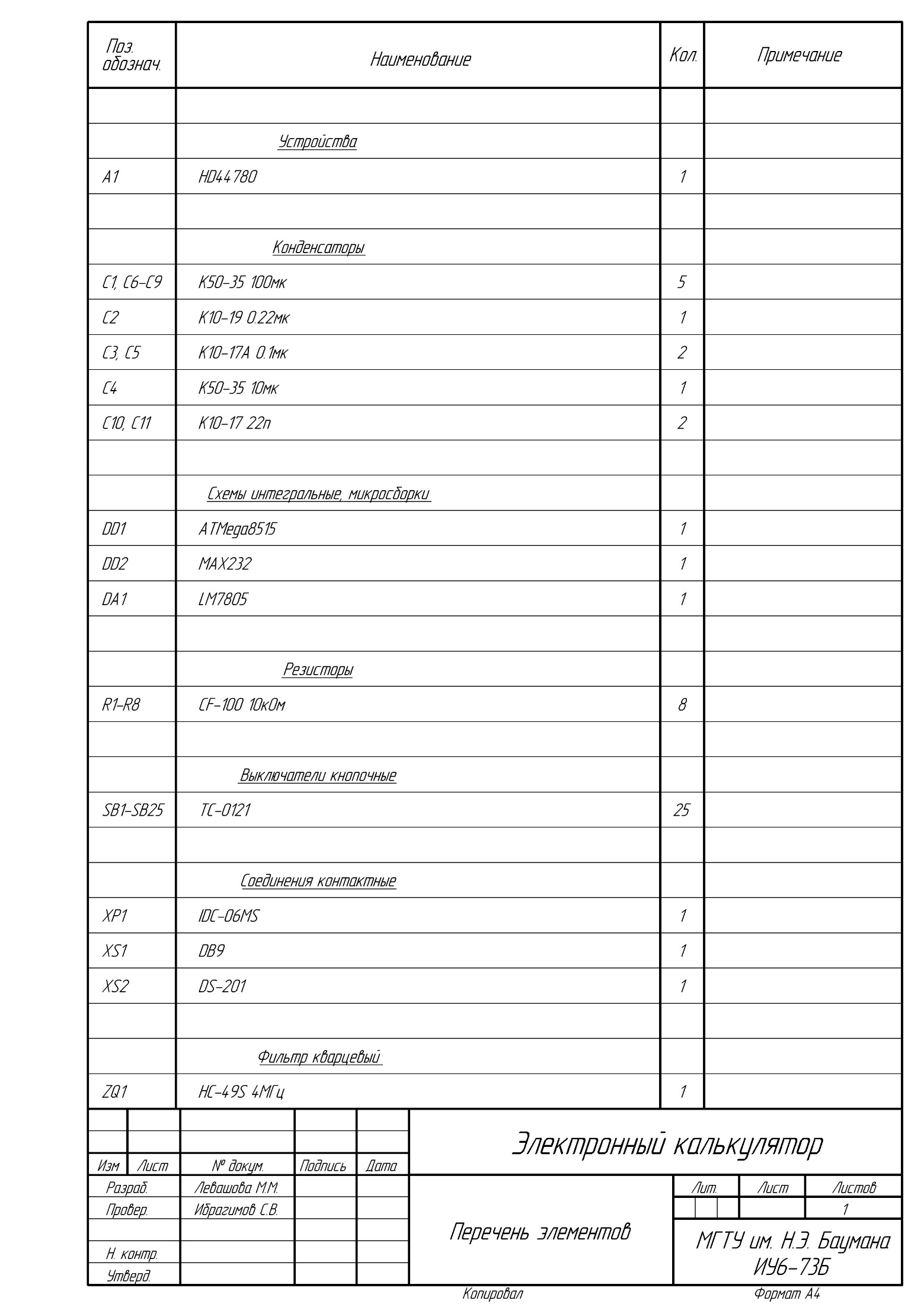
**Электрическая схема функциональная**



**Электрическая схема принципиальная**



# ПРИЛОЖЕНИЕ В

**Перечень элементов**