МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к курсовому проекту

по дисциплине «Микропроцессорные средства и системы»

по теме «Проектирование микропроцессорных устройств

на базе однокристальных микроконтроллеров»

Вариант № 4

Выполнил: студент гр. 6403

Елфимов А.Г.

Руководитель проекта:

Иоффе В.Г.

Дата сдачи:

Оценка:

Самара 2018

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Спроектировать микропроцессорное устройство, предназначенное для раскроя листового материала и реализующее следующие функции: прием сигнала запуска – положительный импульс длительностью 1мс и амплитудой 27В, подсчет числа импульсов, характеризующих длину листа. Один импульс соответствует 1мм, импульсы положительные, амплитуда 5В, длительность – 1мс. За 0.5м до конца листа сформировать импульс, включающий замедление скорости протяжки, а затем сформировать импульсы «Конец листа» и «окончание раскроя». Импульсы положительные, длительностью 3мс, амплитуда – 5В.

Ввод данных с клавиатуры о размере листа (до 10м), количестве листов (до 50), общей длине исходного материала (до 500м), длине остатка, начале и конце обработки. В процессе ввода проверять корректность вводимых данных (соответствие требуемого количества листов, размеров с длиной исходного материала).

Вывод информации на цифровой индикатор: количество листов, длина листа, признаки начала и конца технологического процесса.

По внешнему запросу состояние технологического процесса передавать по RS-485 во внешний приемник на скорости 19200бит/с.

**РЕФЕРАТ**

Курсовой проект.

Пояснительная записка: 41 страница, 13 рисунков, 3 источника, 2 таблицы.

МИКРОКОНТРОЛЛЕР, 7-СЕГМЕНТНЫЙ ИНДИКАТОР, ИНДИКАТОРЫ, КНОПКИ, КЛАВИАТУРА.

В соответствии с техническим заданием спроектирована система на основе ОМК ATmega 8535 семейства AVR фирмы Atmel, предназначенная для раскроя листового материала, вывода на цифровые индикаторы состояния технологического процесса, а также вывода через RS-485 этой информации по внешнему запросу. Разработан алгоритм функционирования системы, структурная схема, а также программное обеспечение.

Система реализована средствами разработки и отладки Proteus 8 с использованием компилятора WinAVR.

Выполнена отладка и проверка системы на работоспособность.

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧНИЙ, ЕДИНИЦ И ТЕРМИНОВ**

МПС – микропроцессорная система;

МПУ – микропроцессорное устройство;

ОМК – однокристальный микроконтроллер;

ПБ – процессорный блок;

ПЗУ (ROM) – постоянное запоминающее устройство;

ОЗУ (RAM) – оперативно запоминающее устройство;

ГСА – граф-схема алгоритма;

Рг – регистр;

7-сегментный индикатор – семисегментный индикатор;

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc500633058)

[1 СТРУКТУРНАЯ ОРГАНИЗАЦИЯ 7](#_Toc500633059)

[2 АЛГОРИТМЫ ОБРАБОТКИ И УПРАВЛЕНИЯ 10](#_Toc500633060)

[3 ВЫБОР ЭЛЕМЕНТНОЙ БАЗЫ 17](#_Toc500633061)

[3.1 Выбор ОМК 17](#_Toc500633062)

[3.2 Выбор 7-сегментных индикаторов 18](#_Toc500633063)

[3.3 Выбор клавиатуры 19](#_Toc500633064)

[3.4 Выбор преобразователя уровня 19](#_Toc500633065)

[3.5 Выбор нормализатора 19](#_Toc500633066)

[3.6 Выбор счетчика-таймера 20](#_Toc500633067)

[3.7 Выбор дешифратора 20](#_Toc500633068)

[4 ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА 21](#_Toc500633069)

[5 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 22](#_Toc500633070)

[6 АЛГОРИТМ ТЕСТИРОВАНИЯ 22](#_АЛГОРИТМ_ТЕСТИРОВАНИЯ)

[7 ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ КОНТРОЛЛЕРА 26](#_Toc500633071)

[ЗАКЛЮЧЕНИЕ 30](#_Toc500633072)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………………32](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)

[ПРИЛОЖЕНИЕ А 33](#_Toc500633073)

[ПРИЛОЖЕНИЕ Б 34](#_Toc500633075)

[ПРИЛОЖЕНИЕ В 35](#_Toc500633076)

# ВВЕДЕНИЕ

В рамках курсового проекта необходимо реализовать систему, предназначенную для раскроя листового материала, а также произвести тестирование программы и создать пакет необходимой документации.

Для решения поставленной задачи эффективно использовать МПС, реализуемую на базе однокристального микроконтроллера (ОМК). Это позволит автоматизировать требуемый технологический процесс.

ОМК обладают многими преимуществами – небольшая стоимость и энергопотребление, а также высокий функционал, достаточный для поставленной задачи. ОМК могут содержать в своем составе порты ввода/вывода, аналого-цифровые и цифро-аналоговые преобразователи, счетчики-таймеры, аналоговые компараторы и другие периферийные устройства, что избавляет от необходимости их отдельной покупки. Высокая скорость обработки данных позволяет использовать их для решения инженерных задач широкого спектра.

Использование ОМК позволит запрограммировать технологический процесс и автоматизировать его работу, минимизировав усилия, требуемые от инженера, работающего за оборудованием, а возможности по расширению и изменению программы, заложенной в ОМК, позволят минимизировать затраты на какие-либо изменения, связанные с модернизацией оборудования или изменения его назначения.

Особенностью данного объекта управления является то, что его использование позволяет без непосредственного контакта с устройством, выполняющим раскрой листового материала, запрограммировать его работу, что является удобным и безопасным способом взаимодействия с оборудованием.

# СТРУКТУРНАЯ ОРГАНИЗАЦИЯ

Особенностью разработки структуры МПУ на основе ОМК является наличие в составе любого типа контроллера функционально полного процессорного блока ПБ [1].

Входные данные вводятся с помощью клавиатуры. Эти данные представляют собой числа, следовательно, матрица клавиатуры должна позволять ввод десятичных цифр (0-9), а также иметь кнопки подтверждения и отмены. Для такой задачи достаточна матрица 3х4, управление которой выполняется с помощью трехразрядного выходного регистра и четырехразрядного входного регистра.

Вывод данных (состояние вкл/выкл, количество листов, длина исходного материала и длина одного листа) производится на 7-мисегментные индикаторы по переполнению счетчика-таймера. Состояние процесса занимает 1 знакоместо, количество листов – 2, длина исходного материала – 6, а длина одного листа – 5. Из этого следует, что необходим один индикатор с одним знакоместом, один индикатор с двумя знакоместами и два индикатор с шестью знакоместами. Выбор конкретной линии индикаторов производится с помощью дешифратора, необходимого для сокращения необходимого количества линий ввода/вывода МК.

Запуск алгоритма работы происходит по запросу INT2. Так как входной сигнал имеет высокое напряжение, есть потребность в нормализаторе, понижающем напряжение до необходимого.

После запуска алгоритма работы кнопкой INT2, результат будет выведен на те же индикаторы. Если пришел запрос от кнопки INT0, текущее состояние технологического процесса будет передано через RS-485 в случае корректности данных.

В соответствии с требованиями технического задания составлена структурная схема используемого контроллера (рисунок 1).

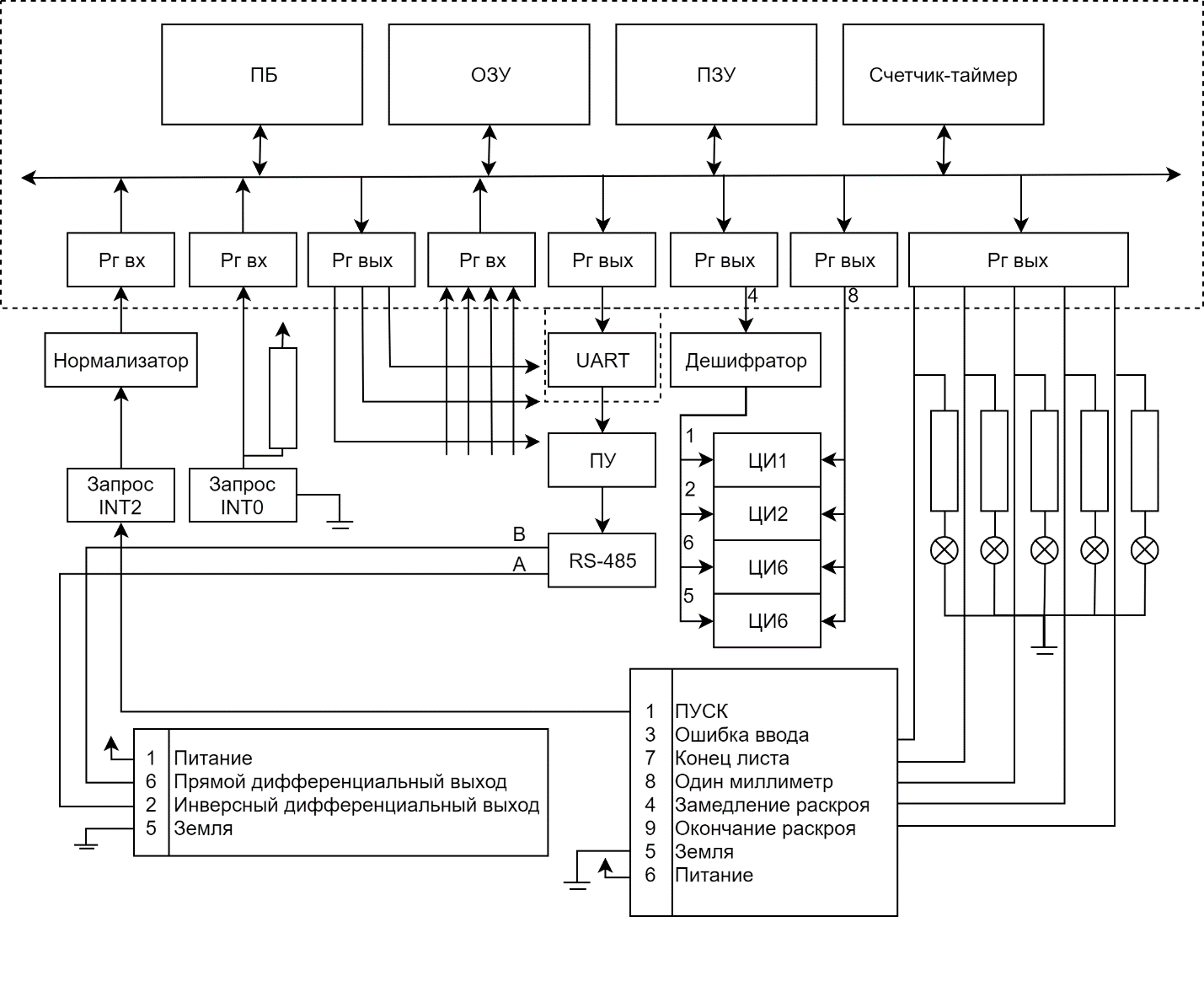


Рисунок 1 – Структурная схема контроллера

Схема состоит из следующих элементов:

* ПБ, выполняющий основные функции по обработке и управлению;
* ОЗУ для хранения входных, выходных данных и промежуточных результатов;
* ПЗУ содержит программы обработки данных;
* Рг вх используется для принятия сигнала прерывания от кнопок INT0, INT2;
* Нормализатор используется для понижения входного напряжения до 5В;
* ПУ – преобразователь уровня;
* Рг вых для вывода данных через UART;
* RS-485 для приема данных, приходящих через UART через ПУ;
* 7-сегментные индикаторы, используемые для отображения данных;
* Дешифратор для выбора конкретной линии 7-сегментных индикаторов;
* Счетчик-таймер используется для вывода значений на цифровые индикаторы;
* Рг вых для вывода данных на выбранную линию 7-сегментного индикатора;
* Рг вых для передачи сигналов во внешнее устройство и включения индикаторов, сигнализирующих о состоянии технологического процесса;
* INT0, INT2 кнопки используются для имитации источников запросов прерываний;
* Рг вых для формирования «бегущего нуля» для опроса клавиатуры;
* Рг вх для приема данных с клавиатуры;
* 9-канальный разъем предназначен для удобного подсоединения внешних источников сигналов и вывода результирующих сигналов.

# АЛГОРИТМЫ ОБРАБОТКИ И УПРАВЛЕНИЯ

Разработка алгоритма функционирования МПУ выполняется на основе технического задания и структурной схемы. Одним из наиболее распространённых средств разработки алгоритмов является проектирование на основе обобщенной граф-схемы алгоритма ГСА [1].

*Граф-схема алгоритма* представляется совокупностью последовательно исполняемых операторов: так называемых операторных вершин, отображающих преобразование данных, и условных вершин, отражающих проверки входов и результатов исполнения предыдущих шагов с целью выбора пути продолжения исполнения.

Алгоритм, работающий в соответствии с заданием по данному курсовому проекту, работаем следующим образом:

1. Ввод данных с клавиатуры;
2. Проверка корректности введенных данных;
3. Обработка данных;
4. Вывод результата.

На рисунке 2 представлена обобщенная граф-схема алгоритма.

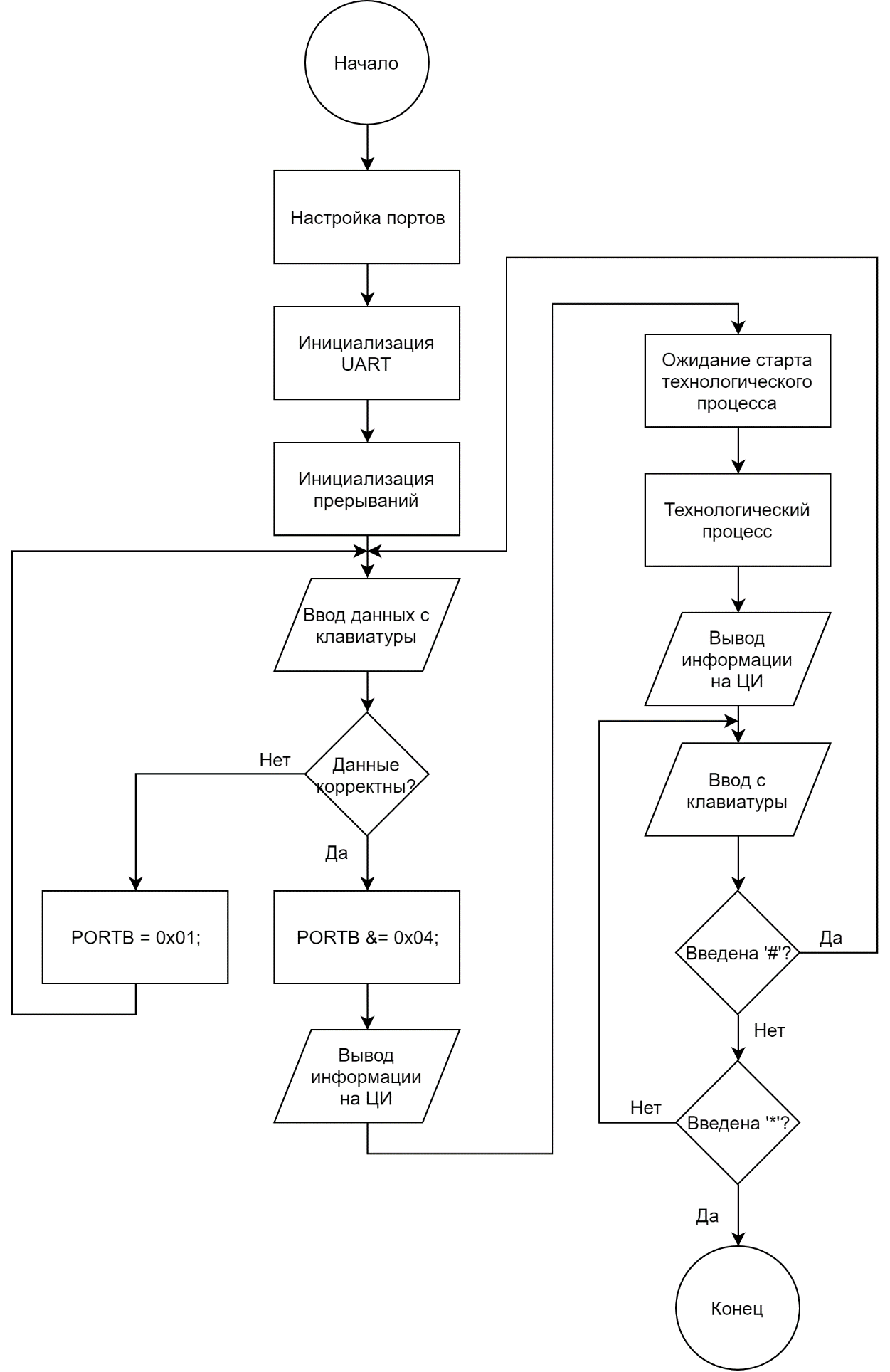


Рисунок 2 – Обобщенная граф-схема алгоритма

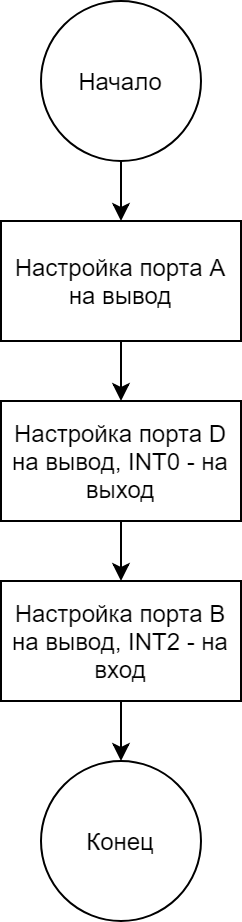


Рисунок 3 – Граф-схема настройки портов

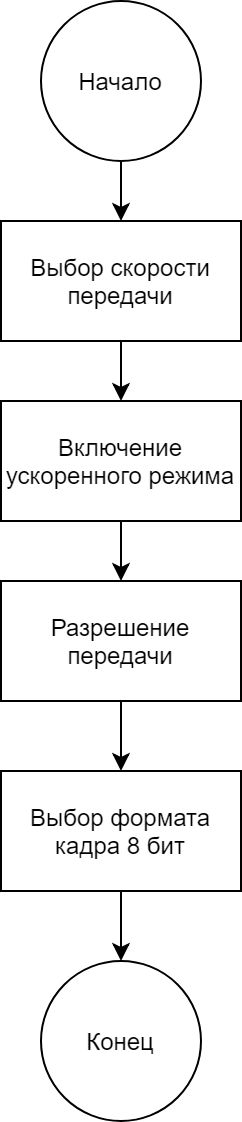


Рисунок 4 – Граф-схема инициализации UART

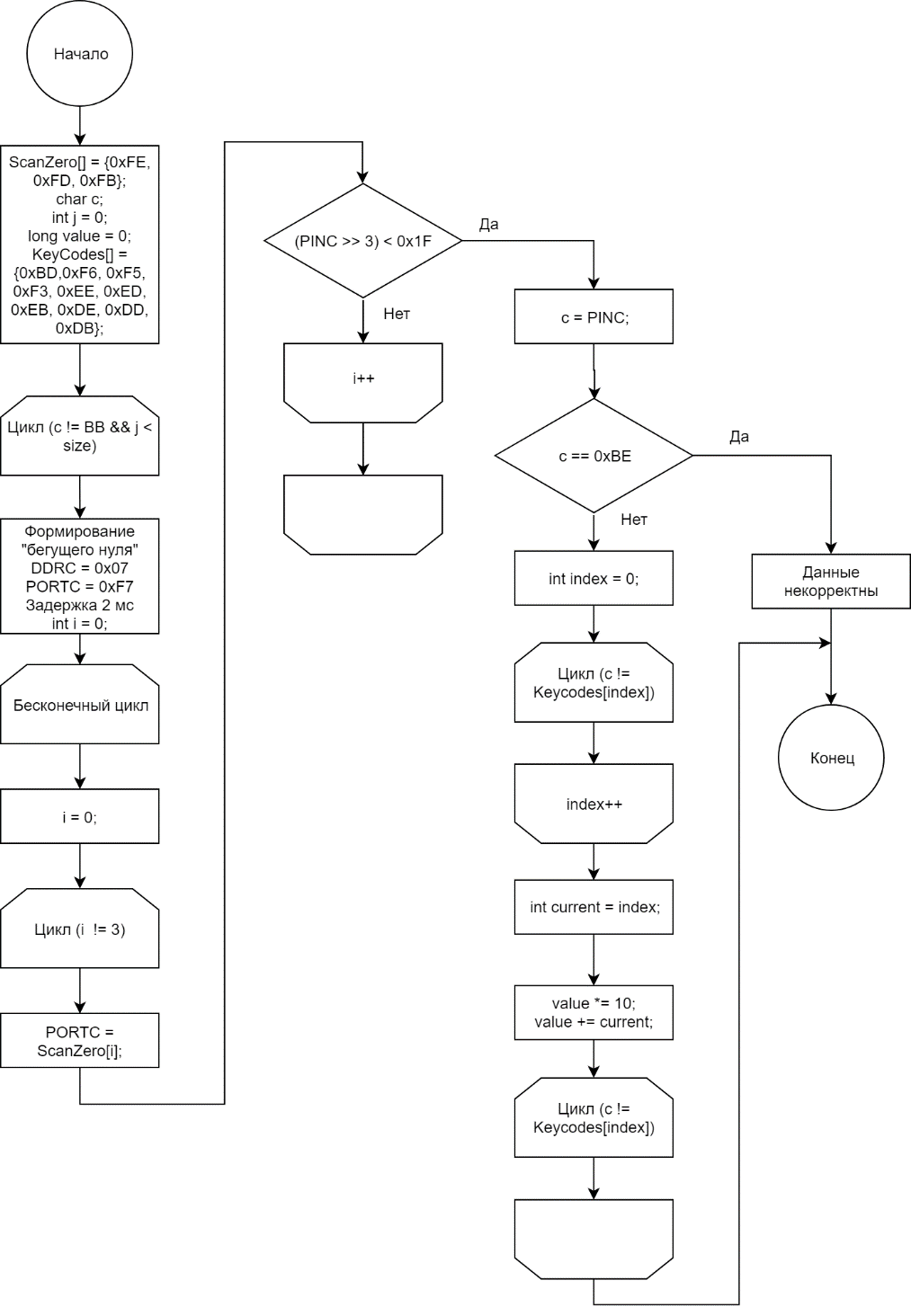


Рисунок 5 – Граф-схема ввода данных с клавиатуры

В ГСА используются следующие обозначения:

* ScanZero[] – массив сканирующего нуля;
* KeyCodes[] – массив возможных значений с клавиатуры;
* Value – значение, введенное с клавиатуры;

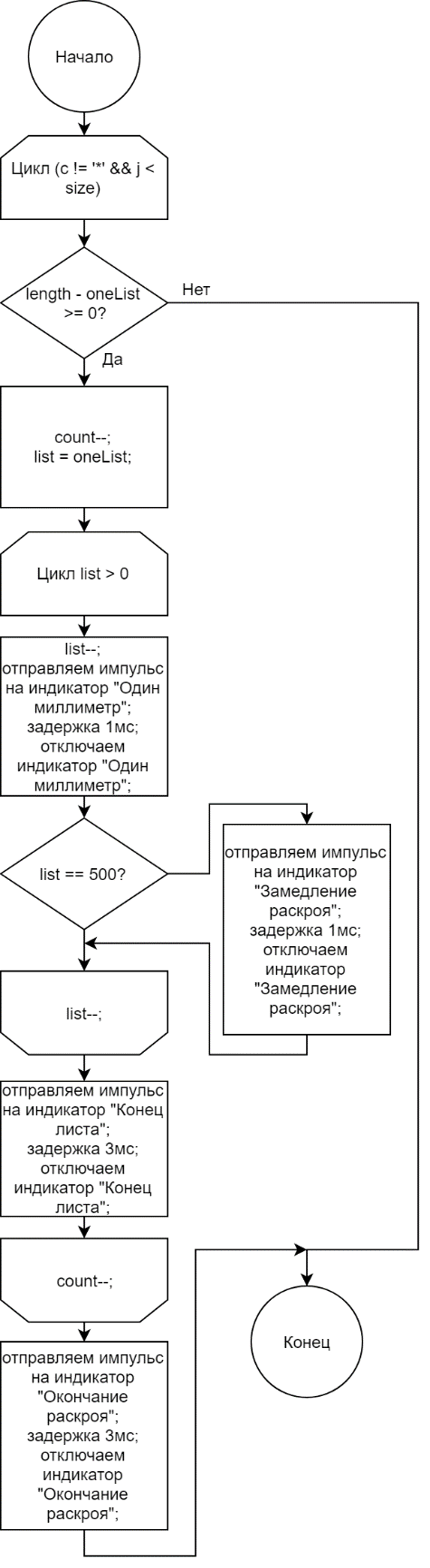


Рисунок 6 – Граф-схема технологического процесса

В ГСА используются следующие обозначения:

* Count – количество листов;
* OneList – длина одного листа;
* Length – общая длина исходного материала;

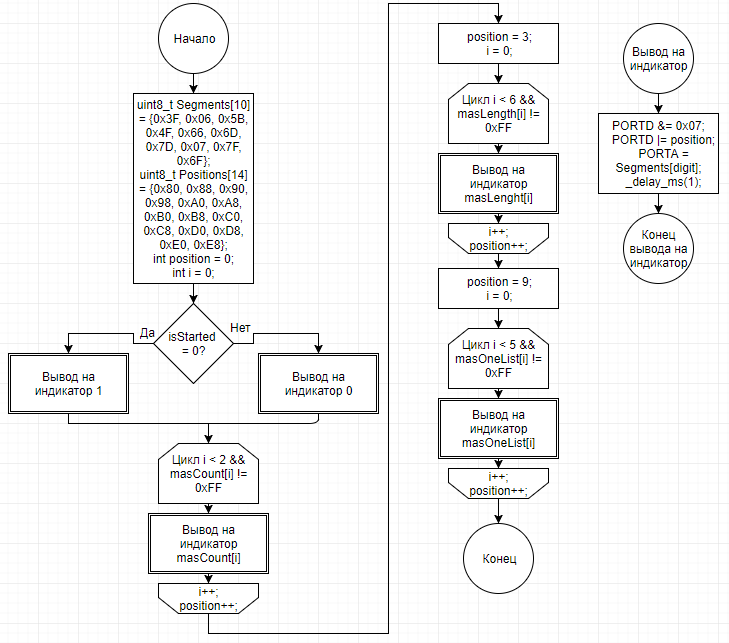


Рисунок 7 – Граф-схема вывода информации на 7-мисегментные индикаторы

В ГСА используются следующие обозначения:

* Segments[] – массив сегментов 7-мисегменотного индикатора для каждой цифры;
* Positions[] – массив позиций для дешифрации и последующего выбора позиции 7-мисегментного индикатора;
* isStarted – значение, характеризующее текущее состояние технологического процесса (запущен/остановлен);
* Position – текущая позиция на цифровых индикаторах

# ВЫБОР ЭЛЕМЕНТНОЙ БАЗЫ

## Выбор ОМК

В ходе выполнения курсового проекта необходимо выбрать ОМК в соответствии с требованиями, которые налагаются содержанием технического задания, обладать функциональной полнотой, чтобы минимизировать число внешних компонентов, необходимых для реализации требуемого МПУ, иметь минимальное количество неиспользуемых линий ввода-вывода, минимальные энергопотребление и стоимость [1].

Исходя из задания и структурной схемы, требуется подобрать ОМК, в состав которого входят следующие элементы:

* ОЗУ;
* ПЗУ;
* контроллер прерываний;
* 30 линий ввода-вывода (7 линий для подключения клавиатуры, 1 линия для передачи информации через RS-485, 2 линии для внешних прерываний, 13 линий для вывода данных на 7-сегментные индикаторы, 5 линий для вывода информации на индикаторы и на внешнее устройство, 1 линия для подключения RS-485);
* счетчик-таймер для вывода на цифровые индикаторы значений.

Анализ имеющихся на рынке ОМК показал, что достаточно выгодно использовать в системе ОМК ATmega 8535 семейства AVR фирмы Atmel, сочетающий в себе функциональность, небольшое количество линий и стоимость.

Основные характеристики Atmega 8535:

* RISC-архитектура;
* встроенный Master/Slave SPI;
* 130 различных команд: арифметических операций, логических операций, операций с битами, пересылками данных, передачи управления и управления системой, в основном выполняемых за один машинный такт;
* 32 восьмиразрядных регистра общего назначения;
* До 16 MIPS производительность при 16 МГц Встроенный 2-х тактный умножитель;
* частота тактового генератора от 0 до 16 МГц;
* 512 байт ОЗУ данных;
* 8 Кбайт Flash-ПЗУ команд;
* EEPROM объемом 512 байт (100000 циклов перезаписи);
* 32 восьмиразрядных линии ввода-вывода общего назначения (4 восьмиразрядных порта ввода-вывода);
* 2 восьмиразрядных таймер/счетчика (могут работать в режиме сравнения) и 1 шестнадцатиразрядный счетчик/таймер (имеет режимы сравнения и захвата);
* Система внутренних и внешних прерываний (до 45 внутренних и до 30 внешних);

Из приведенного перечня характеристик видно, что ATmega 8535 удовлетворяет всем требованиям к системе, перечисленных выше, и как правило, модель ATmega 8535 содержится во всех средствах компиляции и отладки, поэтому выбор этого ОМК позволит упростить процесс разработки программного обеспечения, а наличие оставшихся элементов (EEPROM, SPI и счетчики) позволит расширить систему в случае необходимости.

## 3.2 Выбор 7-сегментных индикаторов

7SEG-MPX1-CC, 7SEG-MPX2-CC, 7SEG-MPX6-CC светодиодные индикаторы – устройства отображения цифровой информации на 1, 2, и 6 цифр соответственно. Это – наиболее простая реализация индикатора, которой может отображать арабские цифры. По сравнению с жидкокристаллическими индикаторами светодиодные имеют свои преимущества, это контрастность отображения информации, малое потребление энергии. Индикаторы нельзя напрямую присоединить к выводам МК, так как в силу большой разности подаваемого напряжения питания и рабочего напряжения индикатора – 2 В можно сжечь не только индикатор, но и порт микроконтроллера. В связи с этим необходимо ставить токоограничительные резисторы, чтобы ограничить ток через них для повышенного напряжении питания. Учитывая, что рабочий ток составляет 10 мА и используя закон Ома, определим значение сопротивления для транзисторов:

## Выбор клавиатуры

Для ввода данных в программу требуется ввод всех десятичных цифр, а также кнопок подтверждения и сброса ввода. Таким образом необходимо 12 кнопок, что достигается использованием матричной клавиатуры 4x3. Достаточно будет использовать Телефонную клавиатурную матрицу KEYPAD-PHONE.

## Выбор преобразователя уровня

RS-485 — стандарт физического уровня для асинхронного интерфейса. Для передачи информации через этот интерфейс необходимо будет преобразовать ее в специальном приемопередатчике – MAX487 – маломощный полудуплексный трансивер для обслуживания RS-485 и RS-422 сетей, содержащий раздельные драйверы приемника и передатчика. Особенностью данного приемопередатчика является ограничение скорости нарастания выходного напряжения, что сводит к минимуму электромагнитные помехи и отражения сигнала, вызванные неправильной нагрузкой кабелей связи.

## Выбор нормализатора

Так как входной сигнал, служащий началом запуска для технологического процесса, имеет значение напряжения 27В, необходимо его понизить, так как ОМК не может работать с таким большим значением. Для этого был использован диод Зенера (стабилитрон) с напряжением стабилизации в 4,7В, позволяющий из повышенного напряжения получать стабильное напряжение определённого уровня, соединенный как показано на рисунке 8. На диод подается через резистор с сопротивлением 450 Ом запирающее напряжение, которое превышает напряжение пробоя диода. При этом через диод протекает ток, и избыток напряжения падает на резисторе. Такое соединение позволит снизить напряжение до 5В, что является сигналом логической единицы для данного микроконтроллера.

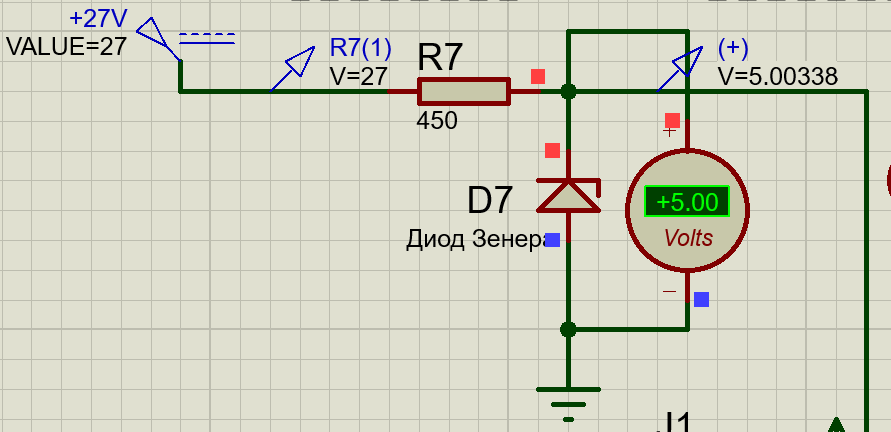


Рисунок 8 – Нормализатор напряжения

## Выбор счетчиков-таймеров

Для вывода информации на цифровые индикаторы требуется с некоторой периодичностью выполнять вывод значений на индикаторы, чтобы человеческий глаз не замечал мерцания значений на экране. Для этого используется восьмиразрядный счетчик-таймер, работающий в режиме переполнения с частотой в 64 раза меньше частоты работы ОМК.

Для отсчета 1-3 мс временных импульсов используется 8битный счетчик-таймер CT0 в режиме таймера работающей на частоте OSC/64.

## Выбор дешифратора

Для вывода данных на цифровые индикаторы требуется вывод на 14 позиций. Использование 14-ти линий ОМК является нецелесообразным, поэтому требуется дешифратор. Требуемое количество выходов дает дешифратор 4-16, для работы которого требуется всего 5 линий (4 линии для выбора и 1 линия для сигнала включения), что сильно сокращает занимаемые линии ОМК.

# ПРИНЦИПИАЛЬНАЯ ЭЛЕКТРИЧЕСКАЯ СХЕМА

Принципиальная схема МПУ представлена на рисунке в приложении А, спецификация элементов в приложении – Б.

Центральным блоком на схеме является ОМК ATmega 8535, работающий от кварцевого генератора частотой 4МГц для устранения погрешности при передачи данных через RS-485 на скорости 19200бит/с.

Выход TXD используется для передачи данных через RS-485.

Кнопки INT0, INT2 подключены к линиям PD2, PB2 соответственно, с помощью них осуществляется старт технологического процесса и процесс передачи информации через RS-485.

К входам PC0-PC6 ОМК подключена клавиатура, которая используется для ввода данных.

К входам PA0-PA7 ОМК подключены линии, необходимые для отображения информации на 7-мисегментные индикаторы.

К входам PD2-PD7 ОМК подключен дешифратор, необходимый для выбора позиции на 7-мисегментных индикаторах.

К входам PB0-PB1, PB3-PB5, PB7 ОМК подключены линии, необходимые для вывода информации на индикаторы и на внешнее устройство.

Вход PB6 используется для включения RS-485.

Виртуальный терминал используется для проверки работоспособности системы, выводя на экран значения, передаваемые через интерфейс RS-485.

+выходы и ПГ для проверки работоспособности системы

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

В качестве языка программирования выбран язык Си – контролируемый статически типизированный язык программирования общего назначения, – так как он обеспечивает более высокую скорость разработки и существенно ее упрощает в сравнении с машинно-ориентированным языком низкого уровня ассемблером.

В качестве инструментального средства для разработки и отладки программного обеспечения ОМК был выбран Proteus 8. Этот пакет программ для автоматизированного проектирования электронных схем. Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов.

Proteus 8 поддерживает МК: PIC, 8051, AVR, HC11, ARM**7**/LPC2000 и другие распространенные процессоры. Более 6000 аналоговых и цифровых моделей устройств. Работает с большинством компилятором и ассемблерами, поддерживает работу с языком Си и имеет развитый интерфейс пользователя.

При разработке программ под МК AVR на СИ, может использоваться любая среда разработки, на выходе которой формируются файлы с расширениями, поддерживаемыми Proteus 8. (HEX, ASM, COF, ELF/DWARF2, UBROF и т.д.).

Выбранная среда разработки находится в бесплатном доступе с некоторыми ограничениями на работу на официальном сайте производителя

Листинг программы приведен в приложении В.

# АЛГОРИТМ ТЕСТИРОВАНИЯ

Известно, что средства диагностирования позволяют МПС самостоятельно локализировать неисправность при условии исправности диагностического ядра, т.е. той части аппаратуры, которая должна быть заведомо работоспособной до начала процесса диагностирования.

Наиболее широкое распространение получил «метод раскрутки», или принцип расширяющихся областей, заключающийся в том, что на каждом этапе диагностирования ядро и аппаратура уже проверенных исправных областей устройства представляют собой средства тестового диагностирования, а аппаратура очередной проверяемой области является объектом диагностирования.

Средства тестового диагностирования (СТД) выполняют следующие функции:

* Загрузку диагностической информации;
* Подачу тестовых воздействий на вход проверяемого блока;
* Опрос ответов с выхода, проверяемого блока;
* Сравнение полученных ответов с ожидаемым (эталонным);
* Анализ и индикацию результатов.

*Тестирование ПЗУ*

Для способа проверки содержимого ПЗУ наиболее целесообразно использовать метод CRC. Он подразумевает подсчет CRC для ПЗУ и сравнение этого значения с CRC, хранящимся в заголовке. hex-файла программы. Подсчет состоит в считывании каждого байта и пропускании его через 16-битовый сдвиговый регистр, выполняющий операцию исключающего или над входным битом ПЗУ и 6, 8, 11 и 15 битами сдвигового регистра. При подаче на регистр каждого бита значение в регистре изменяется. После подачи всех байт из ПЗУ на выходе регистра получается 16-ричное число, которое после деления на характеристический полином даст нам значение CRC ПЗУ. Если это значение совпадает с тем, которое хранится в .hex-файле, ПЗУ считается исправным.

*Тестирование ОЗУ*

В отличие от ПЗУ содержимое ОЗУ не фиксировано, поэтому простой тест суммы для них не применим. Самые известные – тест «шахматная доска», «бегущая единица».

Тест «шахматная доска» заключается в записи шахматной последовательности по адресам памяти, сначала заменяя все биты на значение 01010101, а затем заменяя все биты на значение 10101010. Затем этот тест выполняет считывание каждого значения бита адреса. Если значения битов для данного адреса являются несоответствующими, в протоколе теста отмечается сбой. Тест выполняется циклически от младших до старших адресов.

Тест «бегущей единицы» начинается с установки всех битов в байте в нулевое значение. Поочередно справа налево каждый бит устанавливается в единичное значение, и затем возвращается в ноль. Затем тест выполняет считывание всех битов в байте. Если значения битов в байте являются несоответствующими, в протоколе теста отмечается сбой. Тест выполняется циклически от младших до старших адресов.

*Тестирование портов*

Для этого достаточно соединять порты попарно между собой, и через один порт выводить данные, а через другой – принимать. Это позволяет определить неисправность конкретных линий порта.

*Тестирование USART*

Данное тестирование выполняется «эхоконтролем». Требуется участие внешних приемопередатчиков или дополнительных линий портов, на входе/выходе которых реализуются программно требуемые протоколы обмена.

*Тестирование клавиатуры и индикаторов*

Данные тестирования выполняются одновременно. Необходимо проверить все символы, которые могут потребоваться в ходе работы программы. Для этого последовательно производится нажатие на символы клавиатуры и их отображение на цифровом индикаторе. Если все значения совпали – оба устройства считаются исправными. Если нет – необходимо проверить, какое именно устройство работает неправильно. Для этого мы снова проверяем все символы клавиатуры, последовательно записывая символы нажатых клавиш в файл и считывая их через USART. Если все символы совпадут – клавиатура считается исправной.

*Тестирование работы программы*

Для тестирования работоспособности программы используется виртуальный терминал, который отображает значения, выводимые через интерфейс RS-485.

# ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ КОНТРОЛЛЕРА

Для начала работы необходимо запустить среду Proteus 8 и открыть приложенный файл проекта.

После запуска программы начинается опрос клавиатуры (рисунок 9). Необходимо ввести с клавиатуры необходимые данные: количество листов, длину исходного материала и длину одного листа. В случае некорректного ввода загорится индикатор некорректности введенных данных (рисунок 10), и весь ввод будет необходимо начать с начала. Для отмены введенных чисел следует нажать знак «\*», для подтверждения – «#».

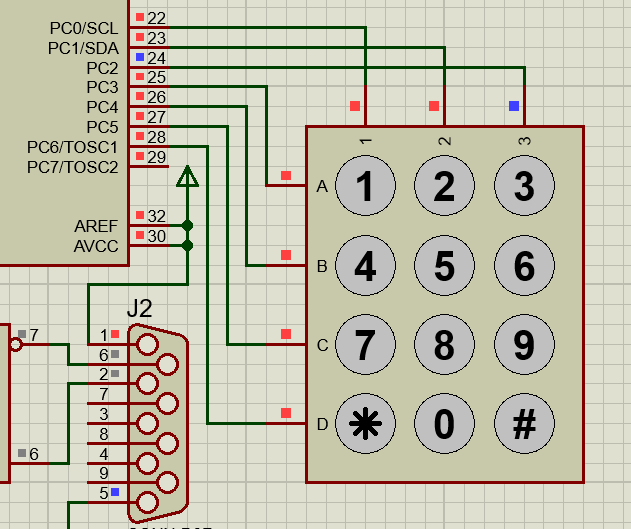


Рисунок 9 – Опрос клавиатуры

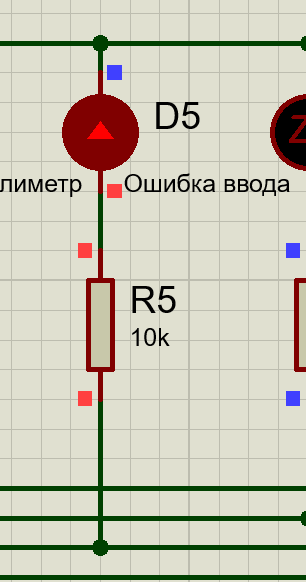


Рисунок 10 – Индикатор некорректно введенных данных

После ввода данных программа ожидает запуска кнопкой INT2, при этом на 7-мисегментные индикаторы выводятся введенные значения (рисунок 11).



Рисунок 11 - Отображение данных на 7-сегментном индикаторе

После прихода сигнала запуска программа выполняет основной алгоритм, по завершению которого выводит на 7-мисегментные индикаторы результат работы и ожидает ввода с клавиатуры следующих значений: «#» для продолжения работы программы и ввода значений с клавиатуры с начала или «\*» для завершения работы программы.

По нажатию на кнопку INT0 в случае корректно введенных данных и при простое системы через RS-485 будут выведены данные о технологическом процессе. Так же эти данные будут выведены на виртуальный терминал для проверки работоспособности системы. Символом-разделителем является значение FF.

**Тестовый пример**

Для тестового примера используем следующие данные: 11 листов по 50мм, длина исходного материала – 7363мм.

После ввода данных и нажатия на кнопку INT2 получим следующий результат (рисунок 12).

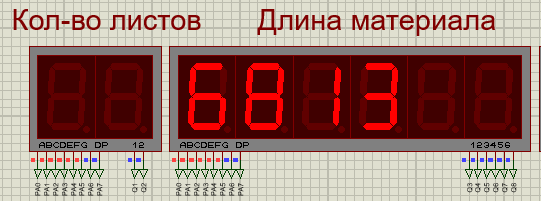


Рисунок 12 – Результат работы программы

Оценивая результат работы системы, данные переданные на 7-сегментный индикатор, можно сказать, что система работает корректно.

|  |  |
| --- | --- |
| Входные данные | Результат работы программы |
| Количество листов – 22  Длина материала – 35142  Длина листа - 41 | Остаток материала - 34240 |
| Количество листов – 88 | Сигнал ошибки ввода |
| Количество листов – 50  Длина материала – 100  Длина листа - 9 | Осталось сделать листов – 39  Остаток материала - 1 |
| Количество листов – 34  Длина материала – 524000 | Сигнал ошибки ввода |
| Количество листов – 20  Длина материала – 300000  Длина листа - 15000 | Сигнал ошибки ввода |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта был разработан алгоритм работы, структура и принципиальная схема системы, предназначенная для раскроя листового материала и вывода состояния технологического процесса на светодиодные 7-сегментные индикаторы или передача его через RS-485.

Для разработки системы потребовались следующие внешние элементы:

* Дешифратор 4-16 для уменьшения количества используемых линий для вывода информации на семисегментные индикаторы. Вместо необходимых 13 линий задействовано всего 5.
* Нормализатор напряжения, состоящий из диода Зенера и резистора, который необходим для понижения напряжения входного сигнала с +27В до +5В.
* Четыре семисегментных индикатора, позволяющие выводить на экран состояние технологического процесса.
* Кнопка INT0 для имитации внешних источников прерываний.
* Матричная клавиатура 4x3 для ввода данных в программу.
* Преобразователь уровня MAX487 для вывода данных через RS-485.

Программа была написана на языке программирования Си, с помощью программы Proteus на языке WinAVR с использованием ОМК Atmega 8535.

Частота работы ОМК равна 4 МГц, что необходимо для обеспечения наибольшей точности передачи через RS-485 на скорости 19200б/с.

В процессе работы системы используется:

* 3743 байта ПЗУ для хранения выполняемой программы;
* 66 байта ОЗУ для хранения временных переменных и промежуточных значений программы (замерено средствами Atmel Studio 7);
* 8-миразрядный счетчик-таймер CT2, работающий в режиме переполнения;
* CT0
* последовательный порт UART в асинхронном режиме, 8 битные данные без проверки четности;
* линии параллельных портов A, B, C и D. Количество ножек

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Иоффе В.Г. Проектирование микропроцессорных устройств на базе однокристальных микроконтроллеров [Электронный ресурс] : [метод. указания] / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева ; сост. В. Г. Иоффе. - Самаpа, 2015. - Режим доступа: http://lib.ssau.ru/ - on-line (дата обращения 18.09.2018)
2. ATmega8535 Datasheet [Электронный ресурс] - http://www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr/atmega8535.htm (дата обращения 11.10.2018)
3. Иоффе В.Г. Разработка и отладка микропроцессорных устройств в виртуальной среде моделирования Proteus [Электронный ресурс] : [метод. указания] / М-во образования и науки Рос. Федерации, Самар. нац. исслед. ун-т им. С. П. Королева (Самар. ун-т) ; сост. В. Г. Иоффе. - Самара : Изд-во Самар. ун-та, 2017. Режим доступа: http://lib.ssau.ru/ - on-line (дата обращения 15.10.2018)

# 

# ПРИЛОЖЕНИЕ А

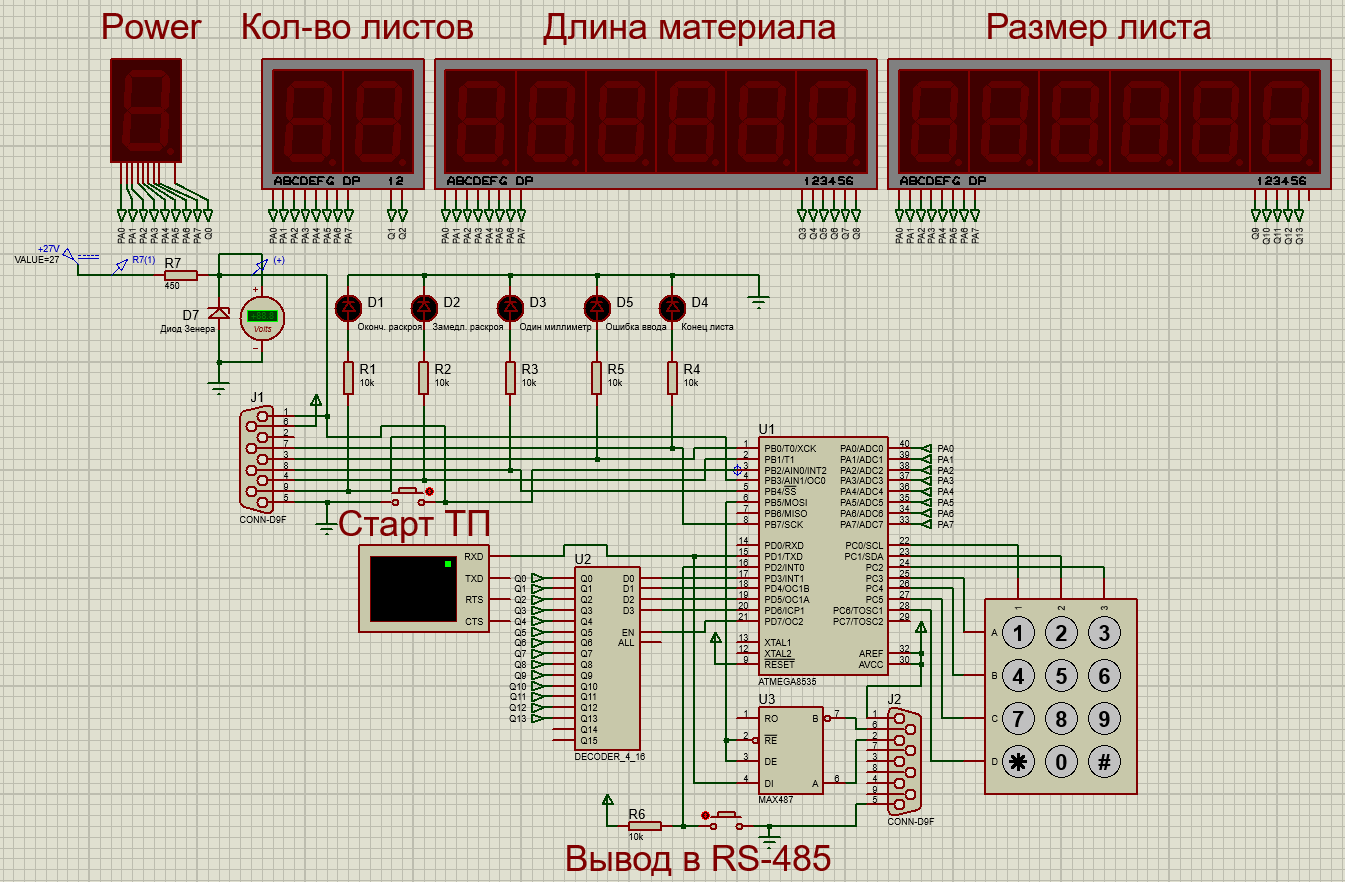


Рисунок А.1 – Принципиальная схема МПУ, выполненная в Proteus

# ПРИЛОЖЕНИЕ Б

**Спецификация элементов**

Таблица Б.1 – Спецификация элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение | Наименование | Кол-во | Описание |
| U1 | ATMEGA8535 | 1 | Однокристальный микроконтроллер |
| U2 | DECODER\_4\_16 | 1 | Дешифратор |
| U3 | MAX487 | 1 | RS-485/RS-422 приемопередатчик |
| J1, J2 | CONN-D9F | 2 | 9-канальный разъем |
| R1-R6 | RESISTOR | 6 | Резистор 10к Ом |
| R7 | RESISTOR | 1 | Резистор 450 Ом |
| INT0 | BUTTON | 1 | Кнопка прерывания INT0 |
| INT2 | BUTTON | 1 | Кнопка прерывания INT2 |
|  | 7SEG-MPX1-CС | 1 | 7-сегментный индикатор на 1 цифру |
|  | 7SEG-MPX2-CС-BLUE | 1 | 7-сегментный индикатор на 2 цифры |
|  | 7SEG-MPX6-CС-BLUE | 2 | 7-сегментный индикатор на 6 цифр |
| D7 | BZX55C4V7 | 1 | Диод Зенера (стабилитрон) |
| D1-D5 | LED-RED | 5 | Светодиод |

# ПРИЛОЖЕНИЕ В

**Листинг программы**

#include <inttypes.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/sleep.h>

#include <util/delay.h>

uint8\_t KeyCodes[10]= {0xBD, 0xF6, 0xF5, 0xF3, 0xEE, 0xED, 0xEB, 0xDE, 0xDD, 0xDB}; //комбинации цифр

uint8\_t Segments[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F}; //сегменты 7-сегментного индикатора

uint8\_t ScanZero[3] = {0xFE, 0xFD, 0xFB}; //бегущий ноль

volatile uint8\_t massiveLength[6]; //массив размера материала

volatile uint8\_t massiveCount[2]; //массив количества листов

volatile uint8\_t massiveOneList[5]; //массив размера одного листа

volatile uint8\_t isStarted; //запущен ли технологический процесс

volatile uint8\_t isCorrect; //корректность данных

volatile long length; //размер материала

volatile int count; //количество листов

volatile int oneList; //размер одного листа

//окончание работы алгоритма

volatile uint8\_t endAlgh;

//Формирование бегущего нуля

uint8\_t InputDigit(){

DDRC = 0x07;

PORTC = 0xF7;

\_delay\_ms(2);

uint8\_t i;

while(1){

i = 0;

while(i != 3){

PORTC = ScanZero[i++];

if((PINC >> 3) < 0x1F){

return PINC;

}

}

}

}

//перевод числа в десятичное

uint8\_t GetDigit(scan){

uint8\_t index = 0;

while(scan != KeyCodes[index]){

index++;

}

return index;

}

//конвертирование числа

long InputValue(uint8\_t size, volatile uint8\_t\* array){

long value = 0;

uint8\_t j = 0;

uint8\_t c;

//при считывании числа предыдущее умножается на 10 и складывается с текущим

while((c = InputDigit()) != 0xBB && j < size){

if(c == 0xBE)

return -1;

array[j] = GetDigit(c);

value \*= 10;

value += array[j++];

while((PINC >> 3) != 0x1F);

}

while((PINC >> 3) != 0x1F);

return value;

}

//вывод числа на определенную позицию ЦИ

void PrintDigit(position, digit){

PORTD &= 0x07;

PORTD |= position;

PORTA = Segments[digit];

\_delay\_ms(1);

}

//позиции ЦИ

uint8\_t Positions[14] = {0x80, 0x88, 0x90, 0x98, 0xA0, 0xA8, 0xB0, 0xB8, 0xC0, 0xC8, 0xD0, 0xD8, 0xE0, 0xE8};

//определение размера числа

int sizer(long value){

int i = 0;

for(;value > 0; i++)

value /= 10;

return i;

}

//Очистка массива

void ClearArray(volatile uint8\_t\* array, int size){

uint8\_t i = 0;

while(i < size){

array[i++] = 0xFF;

}

}

//заполнение массива числом

void FillArray(long value, volatile uint8\_t\* array, int size){

ClearArray(array, size);

int newSize = sizer(value);

uint8\_t i = 0;

long del = 1;

for(; i < newSize - 1; i++)

del \*= 10;

i = 0;

while(i < newSize){

array[i++] = value / del;

value = value % del;

del /= 10;

}

}

//вывод всех значений на ЦИ

void PrintAll(){

uint8\_t pos = 0;

uint8\_t i = 0;

if(isStarted == 0)

PrintDigit(Positions[pos++], 0);

else

PrintDigit(Positions[pos++], 1);

while(i < 2 && massiveCount[i] != 0xFF)

PrintDigit(Positions[pos++], massiveCount[i++]);

pos = 3;

i = 0;

while(i < 6 && massiveLength[i] != 0xFF)

PrintDigit(Positions[pos++], massiveLength[i++]);

pos = 9;

i = 0;

while(i < 5 && massiveOneList[i] != 0xFF)

PrintDigit(Positions[pos++], massiveOneList[i++]);

}

//прерывание компоратора для вывода значений на ЦИ

ISR(TIMER2\_COMP\_vect){

switch(OCF2){

default:{

PrintAll();

}

}

}

//отправка значения в RS-485

void send(uint8\_t value){

while(!(UCSRA & (1<<UDRE))); //очистка буфера

PORTB |= (1<<PB5);

PORTB &= ~(1<<PB6);

UDR = value;

\_delay\_ms(5);

PORTB &= ~ (1<<PB5);

PORTB |= (1<<PB6);

}

//вывод состояния технологического процесса через RS-485

void sendAll(){

uint8\_t i = 0;

while(massiveCount[i] != 0xFF && i < 2)

send(massiveCount[i++]);

i = 0;

send(0xFF); // пробельный символ между числами

while(massiveLength[i] != 0xFF && i < 6)

send(massiveLength[i++]);

send(0xFF);

i = 0;

while(massiveOneList[i] != 0xFF && i < 5)

send(massiveOneList[i++]);

send(0xFF);

}

//основной алгоритм обработки

void Alghoritm(){

isStarted = 1;

long val;

//до тех пор, пока количество листов положительное и еще хватает листового материала происходит раскройка

while(count > 0){

if((val = (length - oneList)) >= 0){

length = val;

count--;

//на каждый миллиметр листа посылается сигнал длительностью 1 мс

long list = oneList;

while(list > 0){

list--;

PORTB |= 0x10;

\_delay\_ms(1);

PORTB &= 0xEF;

//если до окончания листа остается 500мм, посылается сигнал замедления раскроя

if(list == 500){

PORTB |= 0x02;

\_delay\_ms(3);

PORTB &= 0xFD;

}

}

//сигнал окончания листа

PORTB |= 0x80;

\_delay\_ms(3);

PORTB &= 0x7F;

}

else{

//выход из алгоритма, если условие не выполняется

break;

}

}

//заполнение массивово новыми числами

FillArray(length, massiveLength, 6);

FillArray(count, massiveCount, 2);

//сигнал окончание работы

PORTB |= 0x08;

\_delay\_ms(3);

PORTB &= 0xF7;

isStarted = 0;

endAlgh = 1;

}

//прерывание для запуска технологического процесса

ISR(INT2\_vect){

switch(INTF2){

default:{

if(isCorrect && !isStarted)

Alghoritm();

}

}

}

//прерывание для вывода данных через RS-485

ISR(INT0\_vect){

switch(INTF0){

default:{

if(isCorrect)

sendAll();

}

}

}

//ввод массивов с помощью клавиатуры

uint8\_t EnterData(){

count = InputValue(2, massiveCount);

//проверка корректности

if(count <= 0 || count > 50){

PORTB = 0x01;

return 0;

}

length = InputValue(6, massiveLength);

//проверка корректности

if(length <= 0 || length > 500000){

PORTB |= 0x01;

return 0;

}

oneList = InputValue(5, massiveOneList);

//проверка корректности

if(oneList <= 0 || oneList > 10000){

PORTB |= 0x01;

return 0;

}

PORTB &= 0x04;

return 1;

}

//инициализация RS-485

void initRS485(){

UBRRH = 0;

UBRRL = 25;

UCSRA = (1<<U2X); //ускоренный режим

UCSRB |= (1<<TXEN)|(1<<RXEN); //разрешение передачи

UCSRC |= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1); //формат 8 бит

}

int main()

{

//настройка портов и прерываний

DDRA = 0xFF;

DDRD = 0xFB;

PORTD = 0x04;

\_delay\_ms(2);

DDRB = 0xFB;

PORTB = 0x04;

TIMSK = 0b10000000;

TCNT2 = 0x00;

OCR2 = 0xFF;

TCCR2 = 0b01000011;

initRS485();

MCUCR = (1<<ISC01)|(0<<ISC00)|(1<<ISC2);

GICR |= (1<<INT0)|(1<<INT2);

asm("sei");

while(1){

endAlgh = 0;

isStarted = 0;

//ввод значений с клавиатуры

do{

isCorrect = 0;

ClearArray(massiveLength, 6);

ClearArray(massiveCount, 2);

ClearArray(massiveOneList, 5);

isCorrect = EnterData();

}

while(isCorrect == 0);

while(endAlgh == 0);

//ожидание выбора пользователся - продолжить или прервать работу программы

uint8\_t value;

do{

value = InputDigit();

}

while(value != 0xBB && value != 0xBE);

if(value == 0xBE)

return 0;

}

}