Инженерно-технический институт имени А. Айтмухамбетова
Кафедра математики и физики

Ю.П. Мартынюк

Лабораторные работы по микропроцессорной технике

Учебно-методическое пособие

Костанай, 2021

УДК 004.3
ББК 32.973.26-04
М29

Автор:
Мартынюк Юрий Петрович - старший преподаватель кафедры математики и физики инженерно-технического института имени А. Айтмухамбетова КРУ им.А.Байтурсынова

Рецензенты:
Моисеенко Олег Викторович - к.т.н., профессор кафедры «Транспорт и сервис» инженерно-технологического факультета Костанайского инженерно-технического университета имени М. Дулатова
Поезжалов Владимир Михайлович - к.ф.-м..н., профессор кафедры математики и физики инженерно-технического института имени А. Айтмухамбетова КРУ имени.А.Байтурсынова;
Ибрагимова Светлана Викторовна - к.т.н., ассоциированный профессор кафедры электроэнергетики инженерно-технического института имени А. Айтмухамбетова КРУ имени.А.Байтурсынова;

Мартынюк Ю.П.
М29 Лабораторные работы по микропроцессорной технике. Учебно-методическое пособие. Костанай: КРУ им.А.Байтурсынова, 2021.- 90с.

ISBN 978-601-356-086-1

В учебно-методическое пособие включены лабораторные работы по основным разделам курса «Микропроцессорная техника». Работы предназначены для проведения лабораторных занятий в группе студентов, поэтому снабжены таблицами для выбора вариантов. Задания рассчитаны на микроконтроллер ATMega8 и программы WinAVR (AVR-GCC), Proteus Design Suite 7.7 и SimulIDE

Учебно-методическое пособие предназначено для студентов вузов, колледжей, изучающих дисциплину «Микропроцессорная техника» и другие подобные дисциплины.

ББК 32.973.26-04
М29

Утверждено и рекомендовано к изданию Учебно-методическим советом Костанайского регионального университета имени А.Байтурсынова, 30.11.2021 г. протокол № 7

978-601-356-086-1

Содержание

Введение

Широкое применение микроконтроллеров в различных областях техники, науки, и даже в бытовой технике, а также большой интерес, проявляемый учащимися к программированию микроконтроллеров требует более глубокой подготовки будущих специалистов в этой области.

Современная микропроцессорная техника – это область науки и техники, находящаяся на стыке микроэлектроники и программирования. Основная задача, решаемая с помощью микропроцессоров – это управление автоматизированными устройствами, системы сбора данных, контроллеры робототехнических систем.

Тем не менее, несмотря на достаточно широкое распространение микропроцессорных систем и литературы по данной тематике, студенты очень часто сталкиваются с проблемами практического характера при изучении программирования для микроконтроллеров и проектирования систем на них. Одной из часто встречающихся проблем является сложность в отладке устройств, выполнять которую на реальном макете устройства не всегда возможно. Отчасти это решается с помощью всевозможных отладочных плат, например, Arduino [1], но зачастую многие подробности функционирования микропроцессорных систем «ускользают» от учащихся. Одним из решений данной проблемы, а также – серьезным подспорьем при изучении микроконтроллеров, может стать моделирование, которое и является основным средством отладки устройств, применяемым в данном пособии. В каждой лабораторной работе приводятся инструкции по моделированию разрабатываемых устройств средствами одной из двух программ – Proteus Design Suite 7.7 и SimulIDE[2]. В качестве основного компилятора применяется AVR-GCC, входящий в пакет программ WinAVR[3]. Такой набор программ позволяет использовать для работы по пособию компьютер как под управлением Windows, так и Linux.

Для удобства восприятия информации вставки программного кода в тексте заданий выделены шрифтом Arial.

Перед выполнением каждого задания следует создать отдельную папку для него, название которой должно отражать содержимое работы. При использовании программы SimulIDE имя папки должно содержать только английские буквы (для этого лучше расположить папку в корневом каталоге диска C или D). Все файлы задания нужно сохранять в эту папку.

Целью данного пособия является помощь в освоении и систематизации практических знаний в области разработки программ для микроконтроллеров и моделирования микропроцессорных устройств.

Задачи: способствовать более глубокому освоению материала, научить учащихся чертить и читать схемы микропроцессорных устройств, разрабатывать программы для микроконтроллеров, адаптировать существующие программы под заданные условия, производить отладку устройств с микроконтроллерами.

1 Лабораторная работа №1 – Создание базового проекта

1.1 Цель и задачи работы

Цель – научиться создавать базовый микроконтроллерный проект, компилировать программу и запускать моделирование устройства.

Задачи работы: создание папки проекта, создание makefile, создание файла программы, компиляция программы, запуск моделирования устройства.

1.2 Задание 1 - Создание Makefile

Файл Makefile содержит инструкции по переводу программы на языке С в машинный код, то есть, по компиляции программы. Его наличие в папке проекта обязательно для работы компилятора языка С. Если этот файл отсутствует, то компилятор выдаст сообщение об ошибке ***«No rule to make target»***.

Сначала создадим папку для проекта. Откройте приложение «Проводник», перейдите в папку, в которой вы собираетесь расположить папку проекта. Щелкните на пустом месте папки правой кнопкой мыши и в появившемся контекстном меню выберите пункт «Создать -> папку». Создайте новую папку, в которой будут располагаться все файлы вашего проекта, как показано на рисунке 1:

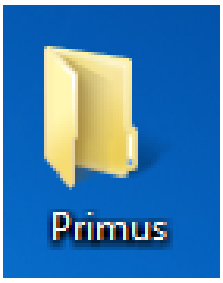

Рисунок 1 – новая папка проекта

Название для папки выберите произвольно (желательно, чтобы оно отражало содержимое, в данном случае – «Primus» в переводе с латыни означает «Первый»). Если вы используете программу SimulIDE, то имя папки и путь к ней не должно содержать русских букв, то есть, ее нельзя размещать на рабочем столе или в библиотеке «Мои документы», так как путь к ним содержит русские буквы. В таком случае рекомендуется разместить папку в корневом каталоге любого диска.

После того, как вы создали папку, запустите программу Mfile (Makefile Generator) из пакета программ Winavr. Эта программа предназначена для создания файлов Makefile, которые хранят инструкции для компилятора по переводу вашего текста программы в машинный код (например, под какой процессор компилировать программу). Окно программы выглядит так, как показано на рисунке 2.

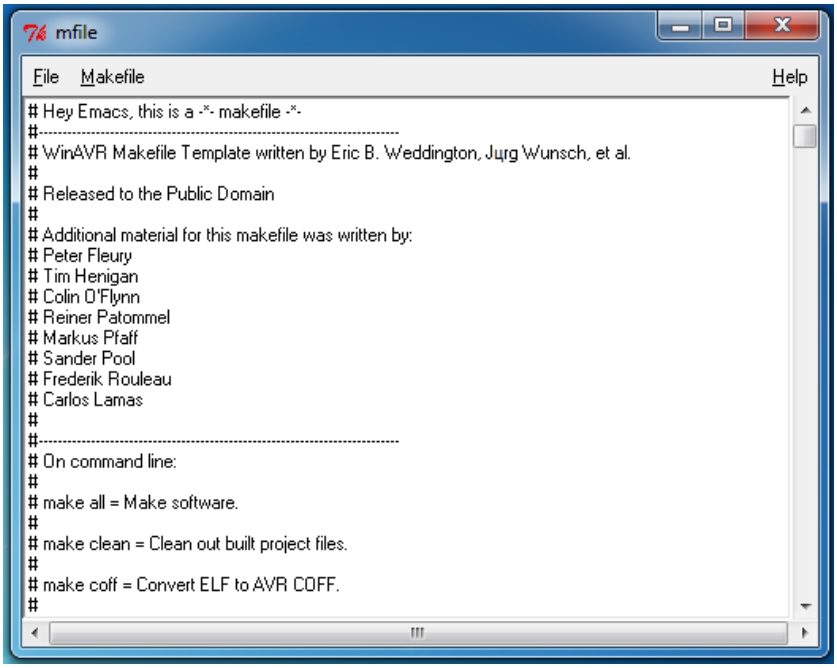

Рисунок 2 – окно программы Makefile

Нам нужно указать, для какого микроконтроллера будет выполняться компиляция. В нашем случае это ATMega8. Выберем его в меню, как показано на рисунке 3.

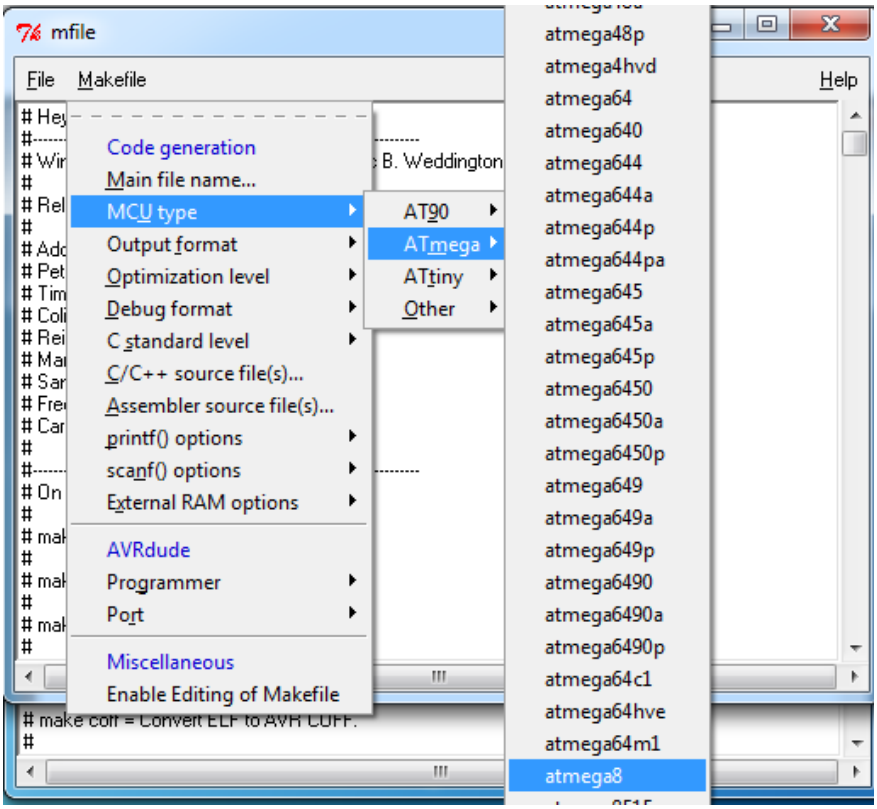

Рисунок 3 – Выбор типа микроконтроллера

Сохраните полученный файл в папку вашего проекта с помощью команды меню FILE-SAVE AS, как показано на рисунке 4

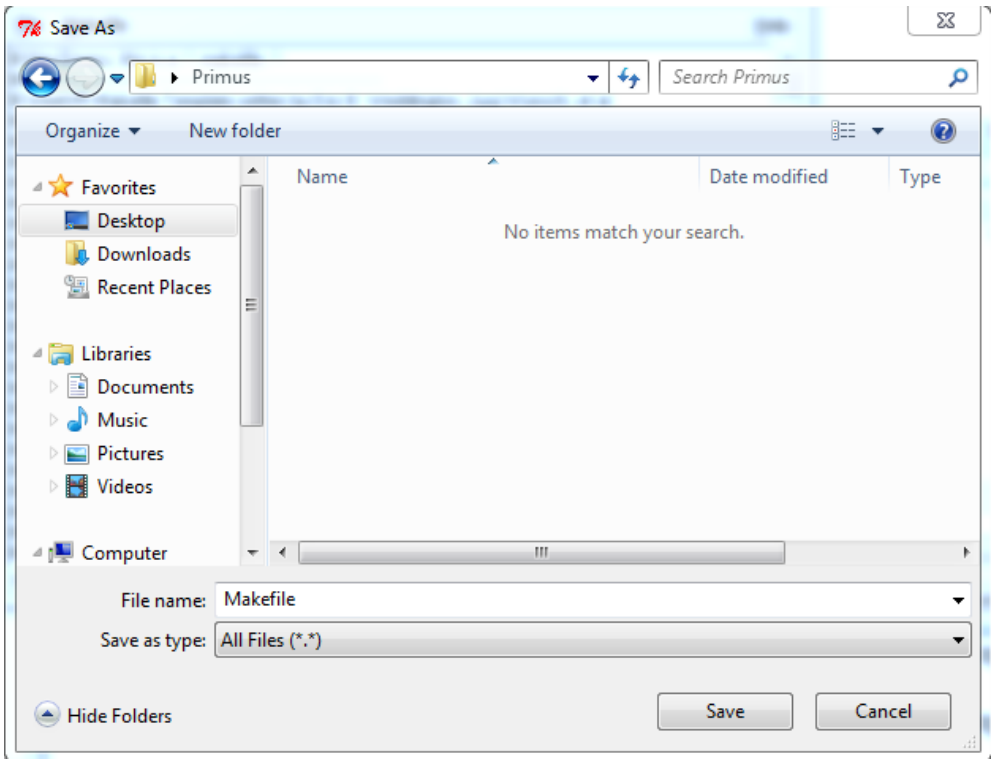

Рисунок 4 – сохранение файла Makefile в папке проекта

Имя файла изменять нельзя! После сохранения закройте программу Mfile

1.2 Задание 2 - создание пустой программы

Запустите программу «Programmers notepad» из пакета программ Winavr. Ее окно выглядит, как показано на рисунке 5

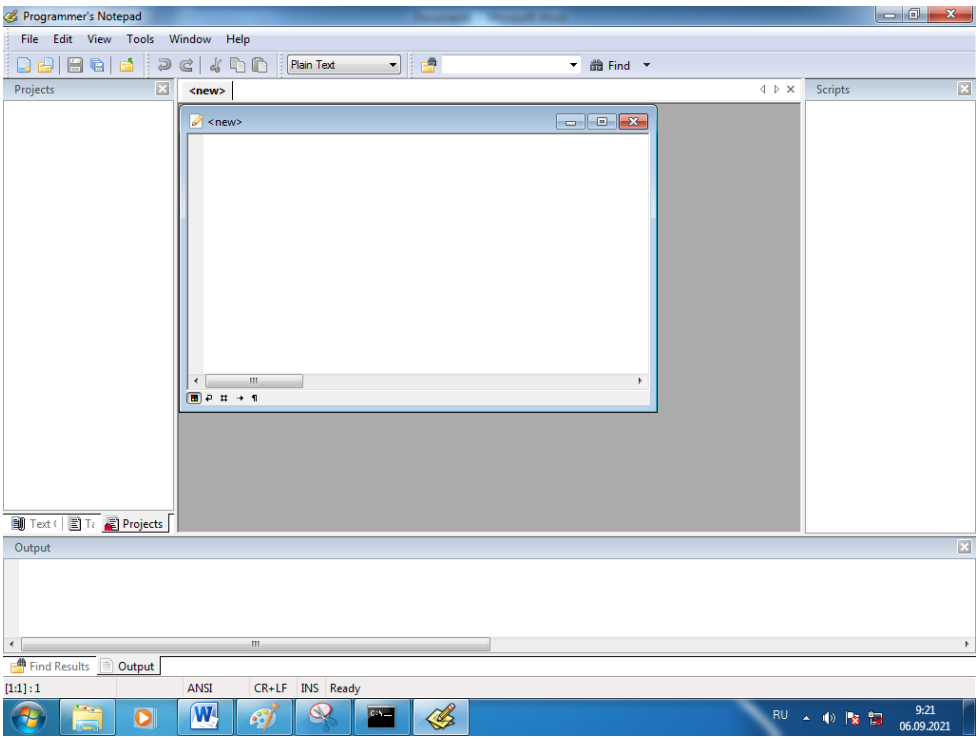

Рисунок 5 – главное окно программы Programmers Notepad

Выберите пункт меню File-Close All. Также, дополнительно закройте все окна, кроме нижнего (Output), в результате окно программы примет вид, показанный на рисунке 6.

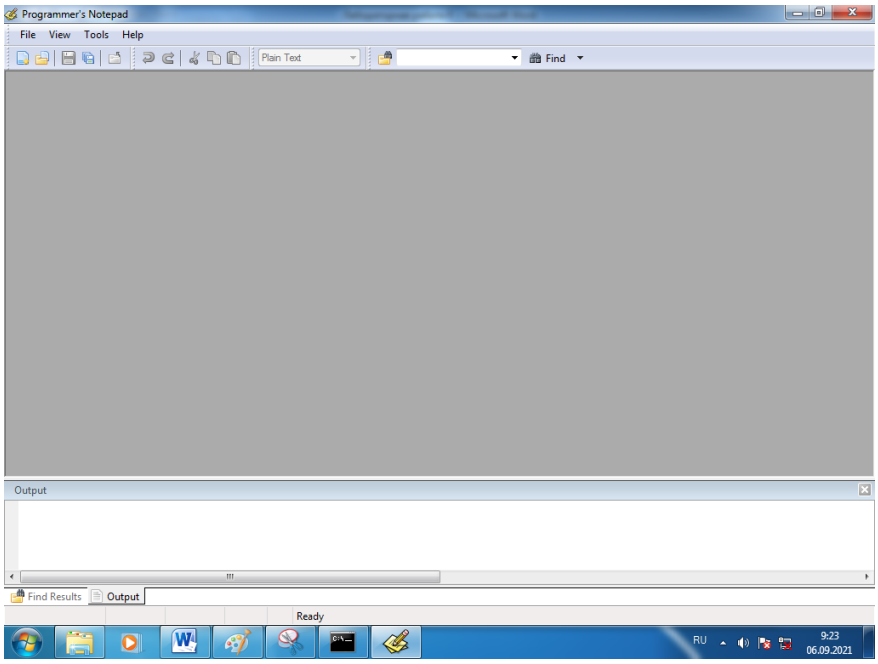

Рисунок 6 – Программа Programmers Notepad после закрытия окон

Выберите пункт меню «File-New-C/C++», в результате откроется новое окно, в которое нужно будет ввести текст программы.

Введите в него текст, показанный на рисунке 7.

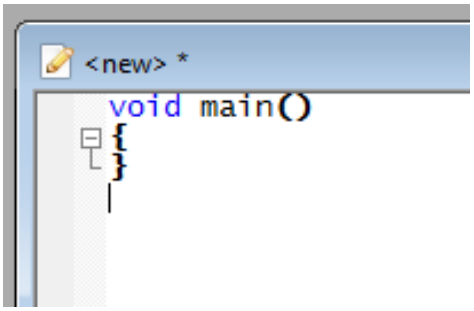

Рисунок 7 – текст пустой программы на языке С

Сохраните полученный файл в папку проекта под именем main.c (выбирать другое имя нельзя), выбрав пункт меню File-Save As. Имя файла необходимо ввести с клавиатуры самостоятельно, как показано на рисунке 8

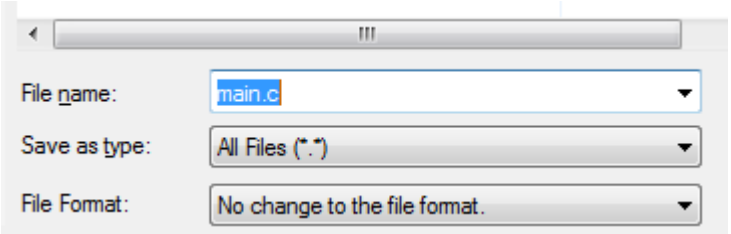

Рисунок 8 – ввод имени файла

Скомпилируйте программу, выбрав пункт меню «Tools- Make all», при этом в окне «Output» должен появиться отчет об успешной компиляции программы, как показано на рисунке 9.

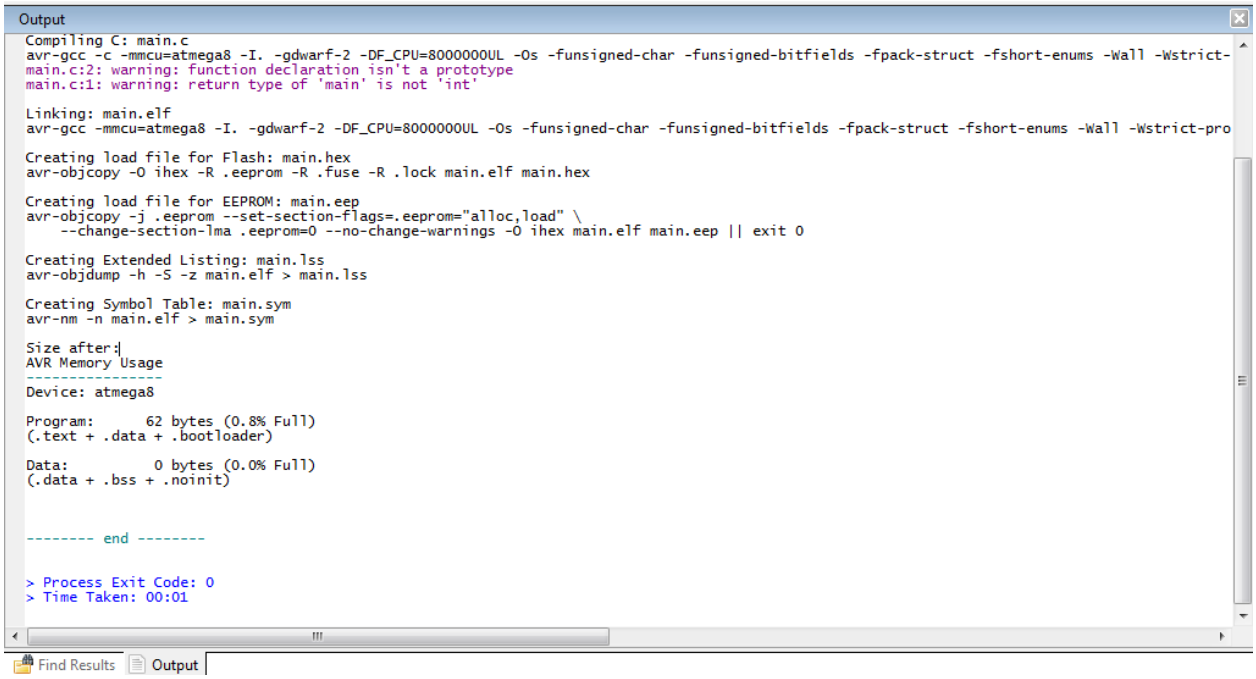

Рисунок 9 – отчет об успешной компиляции программы

Для того, чтобы просмотреть весь отчет, растяните окно Output.

В отчете содержится информация об объеме занятой памяти программ (62 байта) и данных (0 байт) и самое важное – код ошибки равен нулю (> Process Exit Code: 0). Если код ошибки не равен нулю, значит компиляция не была завершена успешно.

После успешной компиляции программы в папке проекта должно появиться несколько дополнительных файлов, и самое главное – файл main.hex с машинными кодами вашей программы. Открыв его в программе «Блокнот», можно увидеть машинный код, как показано на рисунке 10

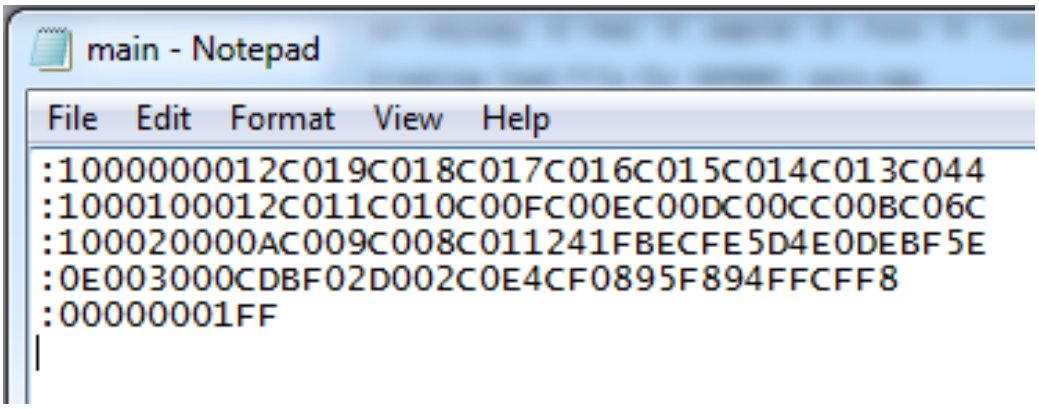

Рисунок 10 – машинный код пустой программы

Это и есть ваша программа в машинных кодах.

1.4 Задание 3 - моделирование проекта

1.4.1 Моделирование в программе SimulIDE

Запустите программу Simulide, выберите в панели компонентов контроллер «Atmega8» и перетащите его на схему, как показано на рисунке 11

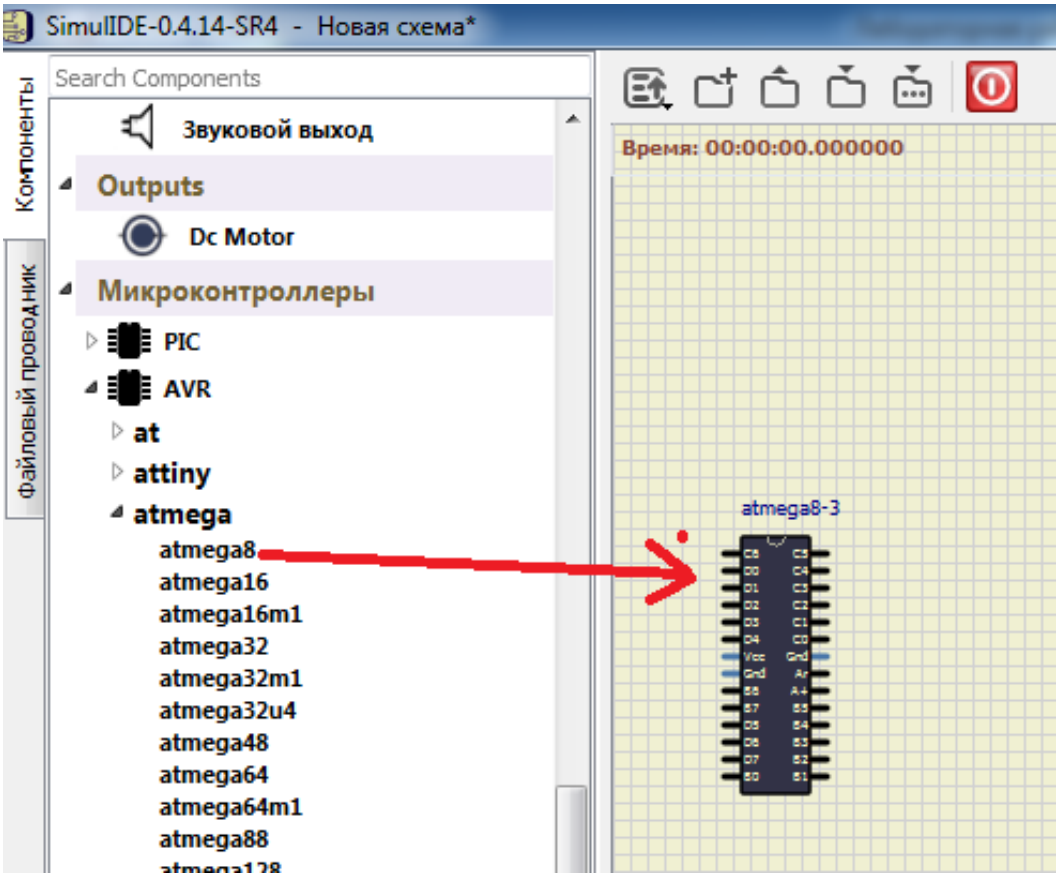

Рисунок 11 – размещение микроконтроллера на чертеже

Нажмите кнопку «Сохранить схему» на панели инструментов, как показано на рисунке 12.

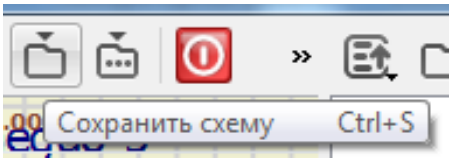

Рисунок 12 – кнопка сохранения схемы в программе Simulide

Сохраните схему в вашей папке проекта, выбрав произвольное имя файла. Затем загрузите прошивку в контроллер, для этого щелкните по нему правой кнопкой и выберите пункт «Загрузить прошивку». Выберите файл «Main.hex». Если при этом возникает ошибка, попробуйте переименовать и переместить папку проекта так, чтобы путь не содержал русских букв. Нажмите на кнопку «Запустить симуляцию»  , при этом не должно возникать ошибок.

1.4.2 Моделирование в программе Proteus

Запустите программу «ISIS» из пакета программ Proteus. Выберите пункт меню «File-Save as »и сохраните файл под любым именем в папке проекта.

Выберите контроллер ATMega8 из списка, как показано на рисунке 13.

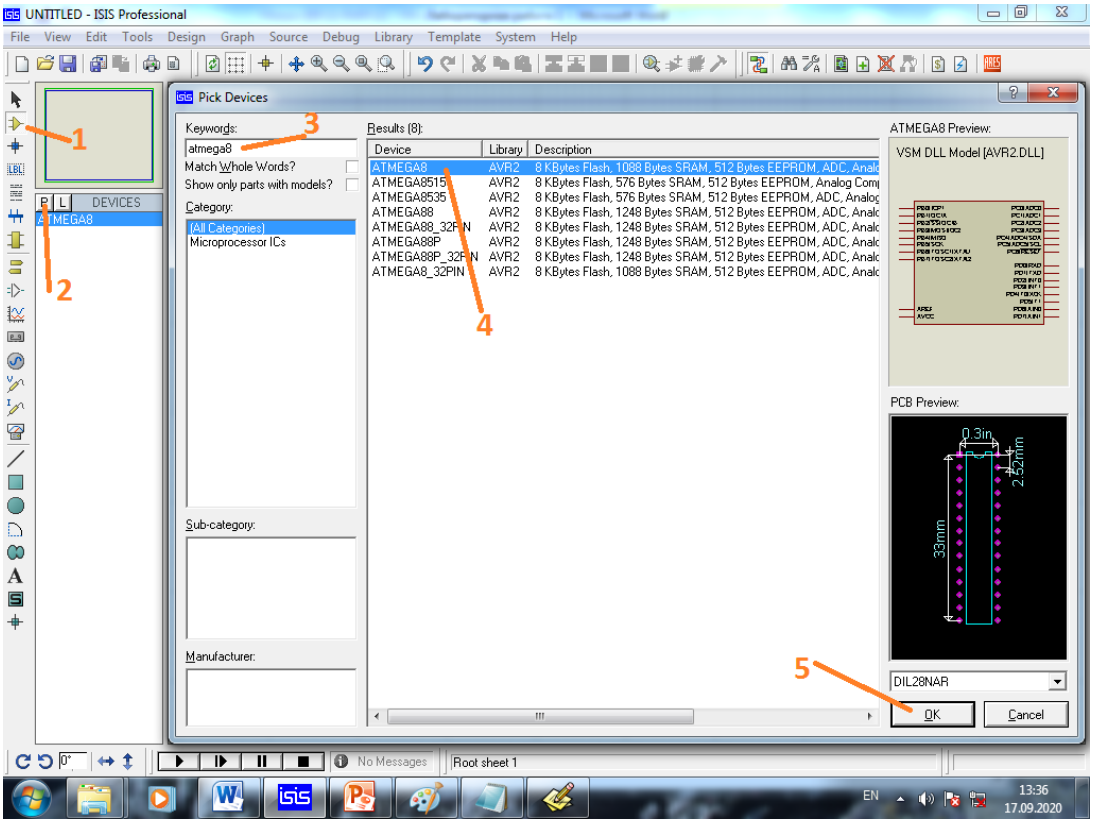

Рисунок 13 – выбор микроконтроллера Atmega8 из библиотеки

Расположите контроллер на схеме, как показано на рисунке 14

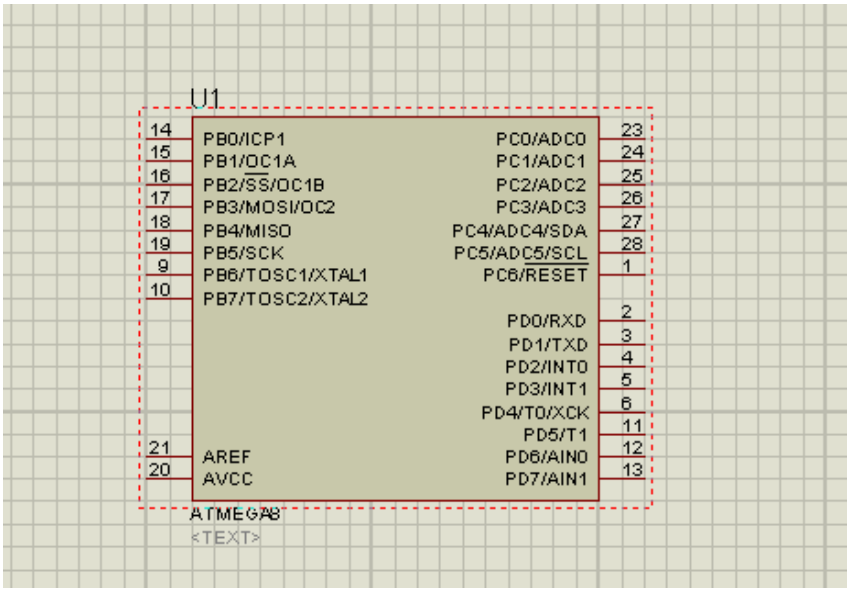

Рисунок 14 – Микроконтроллер после размещения на схеме

Теперь настройте контроллер, для этого дважды щелкните по нему левой кнопкой «мыши» и в открывшемся окне установите свойства, как показано на рисунке 15.

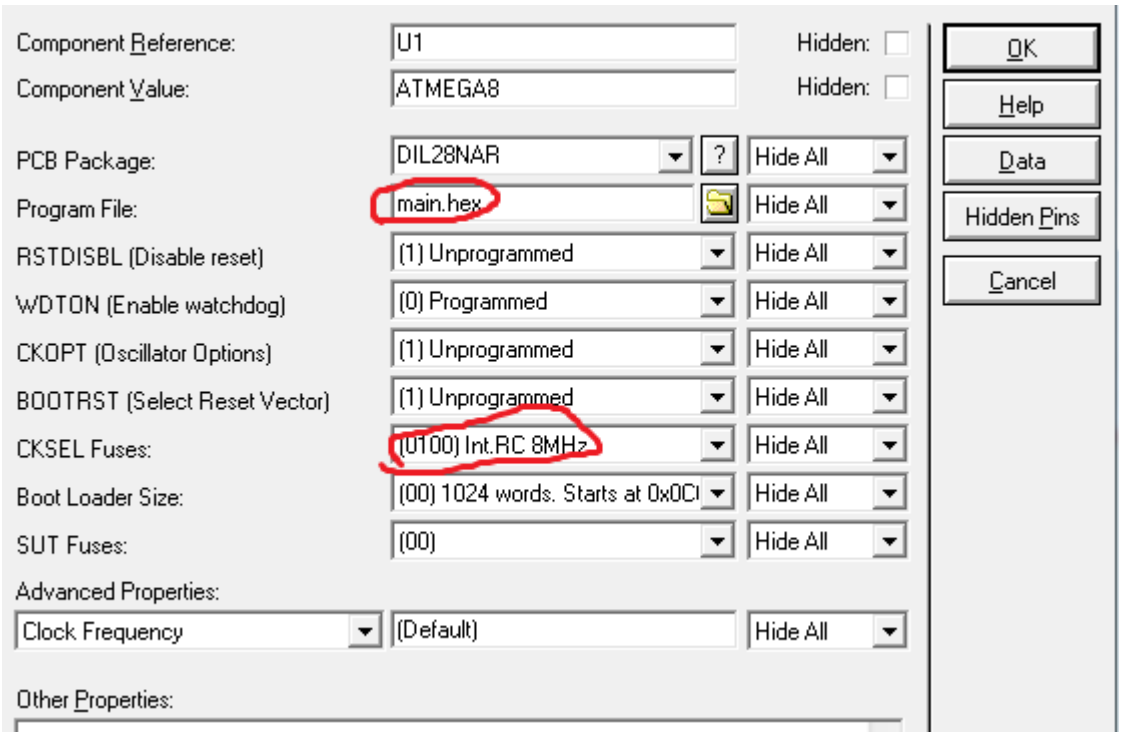

Рисунок 15 – Установка свойств микроконтроллера

Для установки параметра «Program file» необходимо нажать на кнопку с изображением папки около него и выбрать файл «main.hex» из папки проекта

Нажмите кнопку «Play» в нижней части окна. При этом не должно отображаться сообщений об ошибках.

Полученный вами проект – пустой, он не содержит никакого полезного программного кода. Тем не менее, его можно использовать для быстрого создания новых проектов, чтобы не выполнять описанные в работе действия каждый раз . Для этого сохраните папку с пустым проектом и каждый раз, когда вам нужно будет создать новый проект, создавайте копию пустого.

1.5 Контрольные вопросы

1. Для чего нужны языки программирования высокого уровня?
2. Что такое компилятор? Какую функцию он выполняет?
3. Для чего нужно моделировать устройства на микроконтроллерах?

2 Лабораторная работа №2 - программирование линейных алгоритмов

2.1 Цели и задачи работы

Цели – научиться разрабатывать и программировать простейший линейный алгоритм и моделировать работу устройства на его основе, научиться настраивать порты контроллера и производить операции вывода данных.

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

2.2 Задание 1

2.2.1 Разработка программы

Задание - разработать программу, которая мигает светодиодом с указанной частотой, подключенным к указанной линии, как показано в таблице 1 и смоделировать проект. Проект рекомендуется создавать на базе проекта из предыдущей работы (для этого проще всего сделать копию папки из лабораторной работы №1 и выполнять задание новой папке). Перед началом работы запустите все используемые программы и уже в них откройте файлы проекта.

Таблица 1 – варианты лабораторной работы

Вариант	Вывод контроллера, к которому подключать светодиод	Частота, Гц
Пример	PB0	1
1	PB1	0,9
2	PB2	0,8
3	PB3	0,7
4	PB4	0,6
5	PB5	0,5
6	PC0	1,1
7	PC1	1,2
8	PC2	1,3
9	PC3	0,6
10	PC4	0,5
11	PC5	1,1
12	PD1	1,2
13	PD2	1,3
14	PD3	1,4
15	PD4	1,5
16	PD5	1,6

Разработаем алгоритм, как показано на рисунке 16.

Рисунок 16 – алгоритм мигания светодиодом

В данном алгоритме задержка в 0,5 с, примененная дважды, соответствует частоте 1 Гц.

Затем напишем программный код для каждого блока алгоритма.

Настройка порта: Светодиод по условию подключаем к линии PB0. Она относится к порту B. Обратимся к таблице 2, в которой показаны регистры ввода-вывода микроконтроллера и выберем из нее нужный нам регистр.

Таблица 2 – регистры портов ввода-вывода

Порт	Регистр ввода	Регистр вывода	Регистр управления
B	PINB	PORTB	DDRB
C	PINC	PORTC	DDRC
D	PIND	PORTD	DDRD

Нам нужно настроить линию PB0 на вывод. Для настройки порта будем записывать данные в регистр управления порта B DDRB. Составим таблицу, для данного регистра (чтобы узнать, какое число будем записывать в этот регистр), 0 в ней будет соответствовать вводу данных, 1 – выводу, как показано в таблице 3

Таблица 3 – настройка регистра управления

PB7	**PB6**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**
0	0	0	0	0	0	0	1

Получилось двоичное число 00000001. Его и запишем в регистр DDRB:

```
DDRB=0b00000001;
```

Включение светодиода – это операция вывода данных (так как контроллер управляет светодиодом, а не наоборот). Для ее выполнения будем использовать регистр вывода данных порта B PORTB (см таблицу выше). Для того, чтобы включить светодиод, нам нужно выдать логическую единицу на линию PB0. Снова составим таблицу (таблица 4), чтобы определить, какое число записывать в регистр вывода

Таблица 4 – вывод логический единицы на линию PB0

PB7	**PB6**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**
0	0	0	0	0	0	0	1

Получилось двоичное число 00000001. Его и запишем в регистр DDRB:

```
PORTB=0b00000001;
```

Для выполнения программной задержки (чтобы подождать 0,5 с), используем функцию задержки:

```
_delay_ms(500);
```

Время здесь задается в миллисекундах!

Выключение светодиода – это тоже операция вывода данных (так как контроллер управляет светодиодом, а не наоборот). Для ее выполнения будем использовать регистр вывода данных порта B PORTB (см таблицу выше). Для того, чтобы включить светодиод, нам нужно выдать логический 0 на линию PB0. Снова составим таблицу (таблица 5), чтобы определить, какое число записывать в регистр вывода

Таблица 5 – вывод логического нуля на линию PB0

PB7	**PB6**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**
0	0	0	0	0	0	0	0

Получилось двоичное число 00000000. Его и запишем в регистр DDRB:

```
PORTB=0b00000000;
```

Для перехода в начало алгоритма используем оператор безусловного перехода goto:

Разместим метку, чтобы обозначить, КУДА будет происходить переход. Придумаем имя метки:

my_label:

В том месте, откуда будет происходить переход, поставим оператор goto и укажем в нем имя метки my_label.

goto my_label;

Припишем строки программы, которые получились у нас, к алгоритму, как показано на рисунке 17

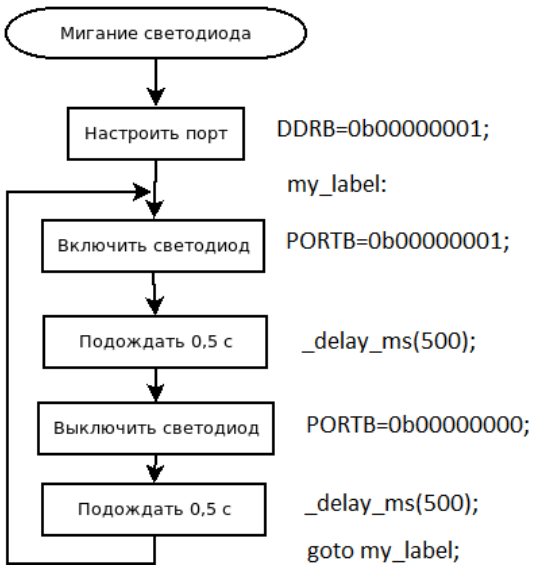

Рисунок 17 – алгоритм с программным кодом

Оформим программу согласно правилам оформления программ и снабдим комментариями:

```
#include<avr/io.h> //библиотека ввода-вывода
#include<avr/delay.h> //библиотека с функцией задержки _delay_ms
void main () //заголовок главной функции программы
{
DDRB=0b00000001;//настраиваем линию PB0 на вывод
my_label: //метка, куда переходить
PORTB=0b00000001;//включаем светодиод
_delay_ms(500); //ждем 500 мс
PORTB=0b00000000;//выключаем светодиод
_delay_ms(500); //ждем 500 мс
goto my_label; //переходим в начало алгоритма
}
```

Затем соберем программу. Для этого нужно выбрать пункт меню tools-make all. Если ошибок не было – внизу появится сообщение «>

Process Exit Code: 0», как показано на рисунке 18. Если код не равен 0, то в программе есть ошибка – ее надо устранить и выбрать пункт меню tools-make all заново, пока ошибки не будут устранены

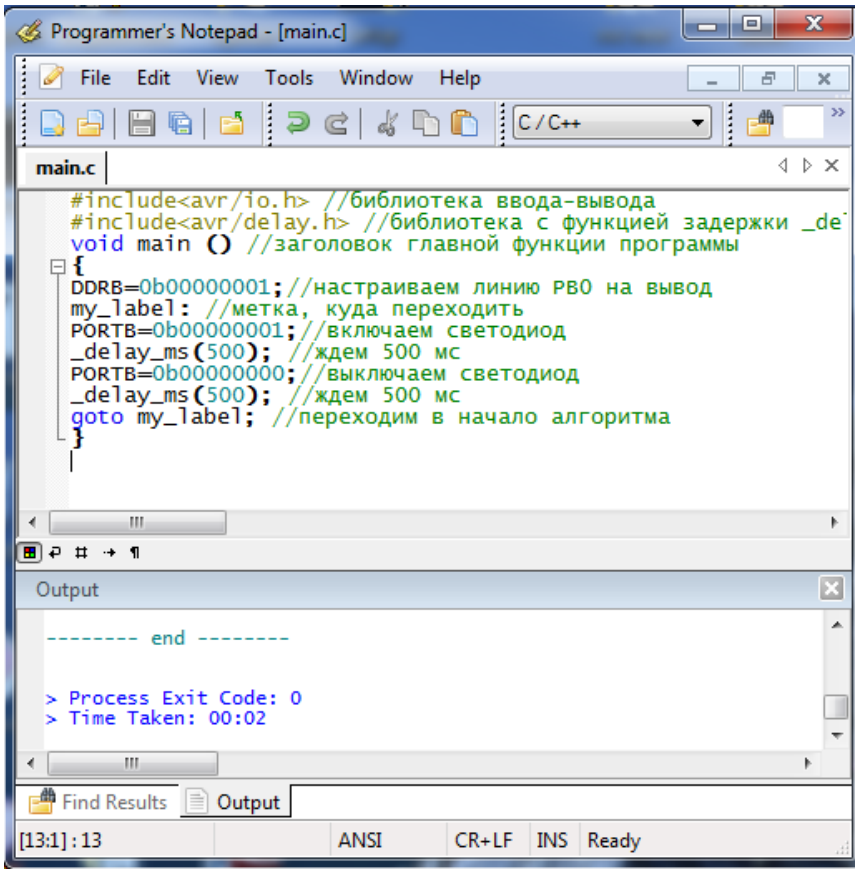

Рисунок 18 – пример успешной компиляции программы

2.2.2 Разработка схемы

2.2.2.1 Разработка схемы в программе SimulIDE

Добавьте на схему светодиод, резистор и терминал общего провода (земли), как показано на рисунке 19

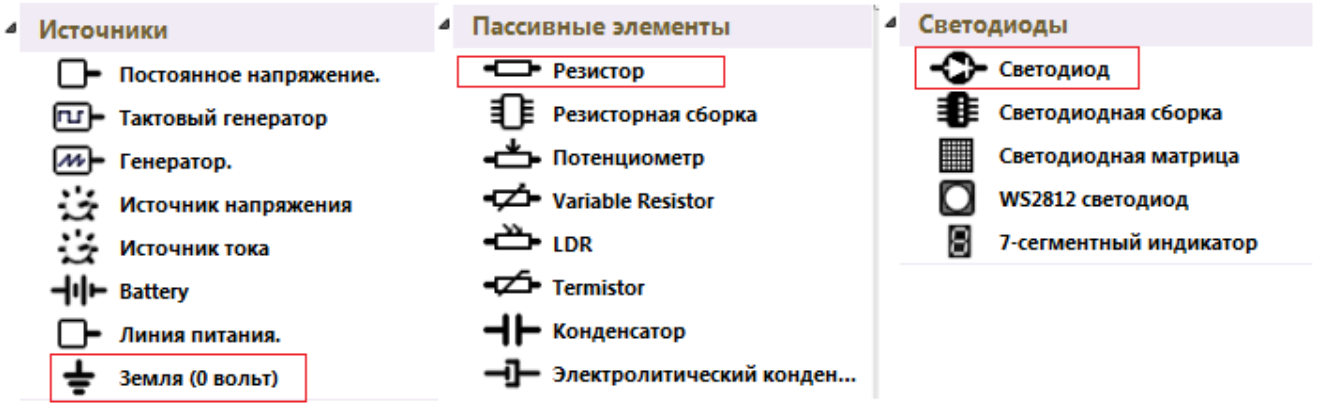

Рисунок 19 – выбор элементов схемы

Они должны выглядеть так, как показано на рисунке 20 (вместе с контроллером, оставшимся от предыдущей работы).

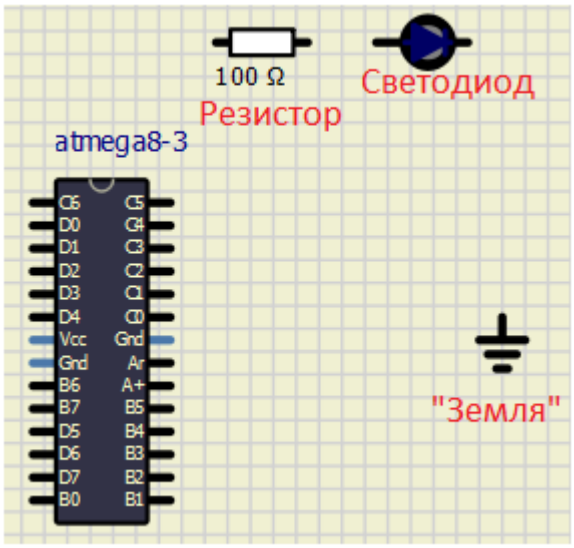

Рисунок 20 – элементы схемы

Соедините элементы в схему, как показано на рисунке 21 (**Обратите внимание, что в вашем варианте будет использоваться другой вывод микроконтроллера, чем в примере**).

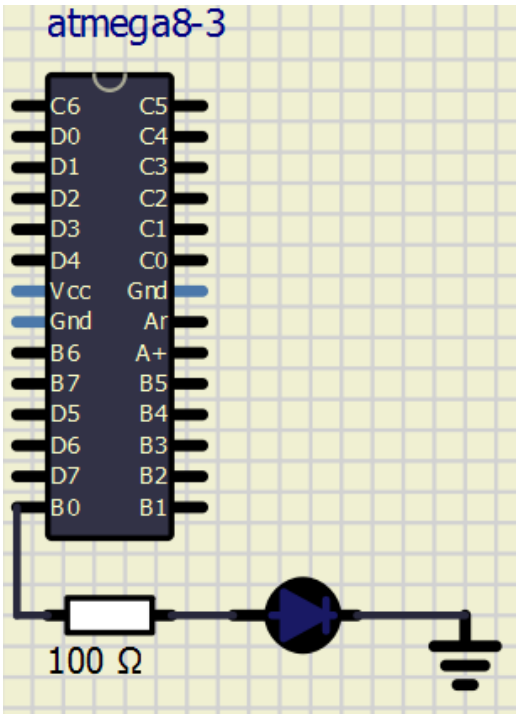

Рисунок 21 – смонтированная схема

2.2.2.2 Разработка схемы в программе Proteus

Для разработки схемы нам понядобятся: микроконтроллер, светодиод, резистор . Выберем сначала контроллер, как показано на рисунке 13, если его

еще нет на схеме. Аналогично выберем резистор (в поле keywords нужно ввести RES вместо Atmega8) и светодиод (в поле keywords нужно ввести LED-RED).

Также нам понадобится «земля» (общий провод), которую следует выбрать из панели терминалов, как показано на рисунке 22.

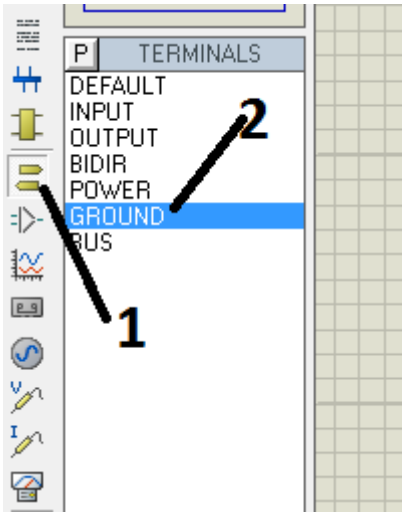

Рисунок 22 – выбор терминала общего провода

Выполняем соединения, как показано на рисунке 23

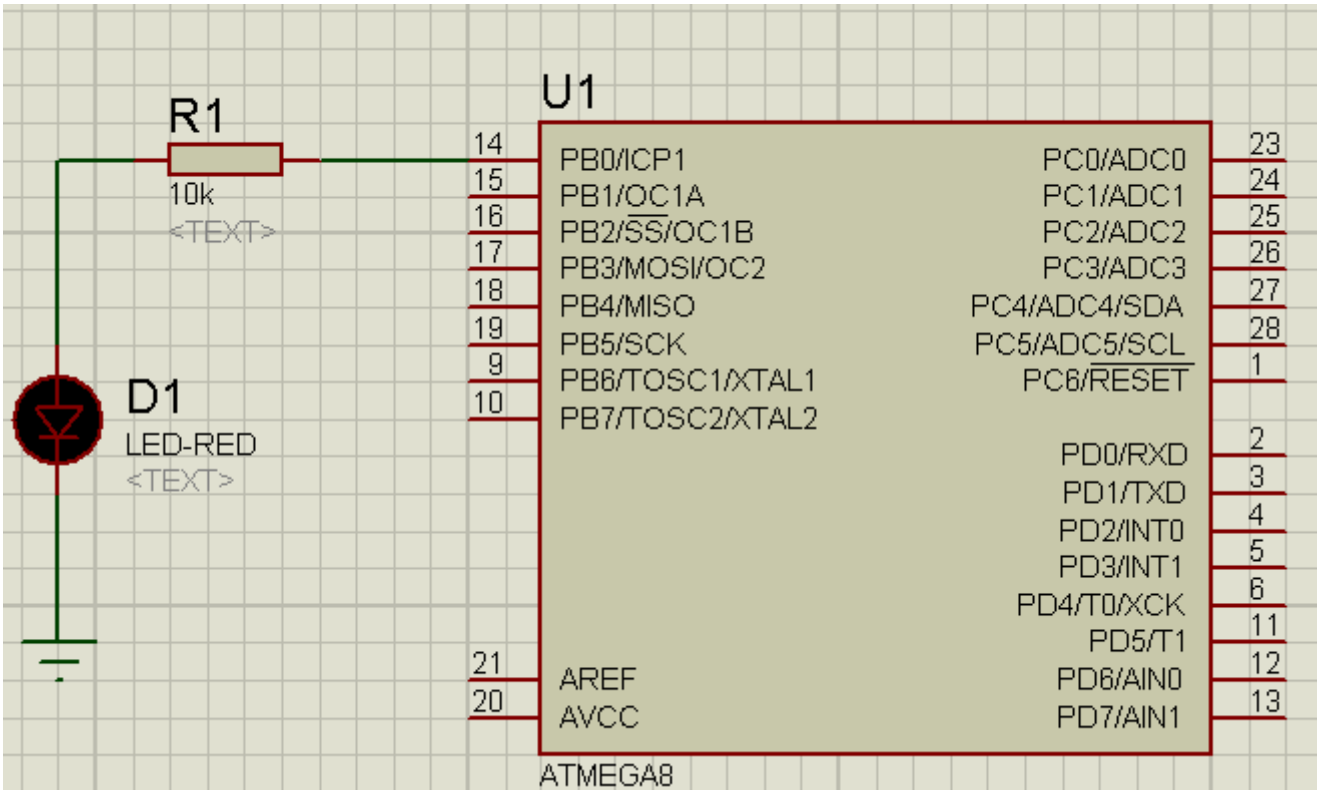

Рисунок 23 – смонтированная схема

Затем нужно настроить резистор (дважды щелкнуть по нему и 10 К поменять на 1 К) и светодиод (10 мА поменять на 1 мА, иначе он не будет светиться в этой схеме).

Сохраним схему в папку проекта.
Нажимаем кнопку “Play” и наслаждаемся миганием светодиода!

2.3 Задание 2

Разработать и смоделировать программу, которая мигает двумя светодиодами поочередно (подключение светодиодов показано в таблице 6).

Таблица 6 – варианты задания 2

Вариант	Вывод контроллера, к которому подключать светодиоды
1	PB0,PC0
2	PB0,PC1
3	PB0,PC2
4	PB0,PC3
5	PB0,PC4
6	PB0,PC5
7	PB1,PC0
8	PB1,PC1
9	PB1,PC2
10	PB1,PC3
11	PB1,PC4
12	PB1,PC5
13	PB2,PC0
14	PB2,PC1
15	PB2,PC2
16	PB2,PC3

2.4 Задание 3

Разработать и смоделировать программу, которая зажигает 8 светодиодов поочередно, а потом поочередно их гасит .

2.5 Контрольные вопросы

1. Какие виды алгоритмов вы знаете? Какие из них были использованы в данной лабораторной работе?
2. Чем режим ввода данных отличается от режима вывода данных?
3. Для чего светодиод подключается через резистор?

3 Лабораторная работа №3 - Ввод данных

3.1 Цель и задачи работы

Цель – научиться программировать операции ввода двоичных данных, использовать условный оператор и битовые операции.

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, разработка схемы, моделирование устройства.

3.2 Задание 1

3.2.1 Разработка программного кода

Разработать программу, которая мигает светодиодом с частотой 1 Гц, подключенным к указанной линии, если нажата кнопка, подключенная к заданной линии, как указано в таблице 7 и смоделировать проект. Кнопку подключить таком образом, чтобы при нажатии кнопки на микроконтроллер поступал уровень логического нуля, а когда кнопка не нажата – логической единицы (то есть, включить кнопку между линией микроконтроллера и общим проводом схемы, и дополнительно подключить резистор между линией питания и той же самой линией контроллера).

Таблица 7 – список вариантов

Вариант	Вывод контроллера, к которому подключать светодиод	Вывод контроллера, к которому подключать кнопку
Пример	PB0	PB1
1	PB1	PB2
2	PB2	PB3
3	PB3	PB4
4	PB4	PB5
5	PB5	PC0
6	PC0	PC1
7	PC1	PC2
8	PC2	PC3
9	PC3	PC4
10	PC4	PC5
11	PC5	PD0
12	PD0	PD1
13	PD1	PD2
14	PD2	PD3
15	PD3	PD4
16	PD4	PD5

Разработаем алгоритм, как показано на рисунке 24:

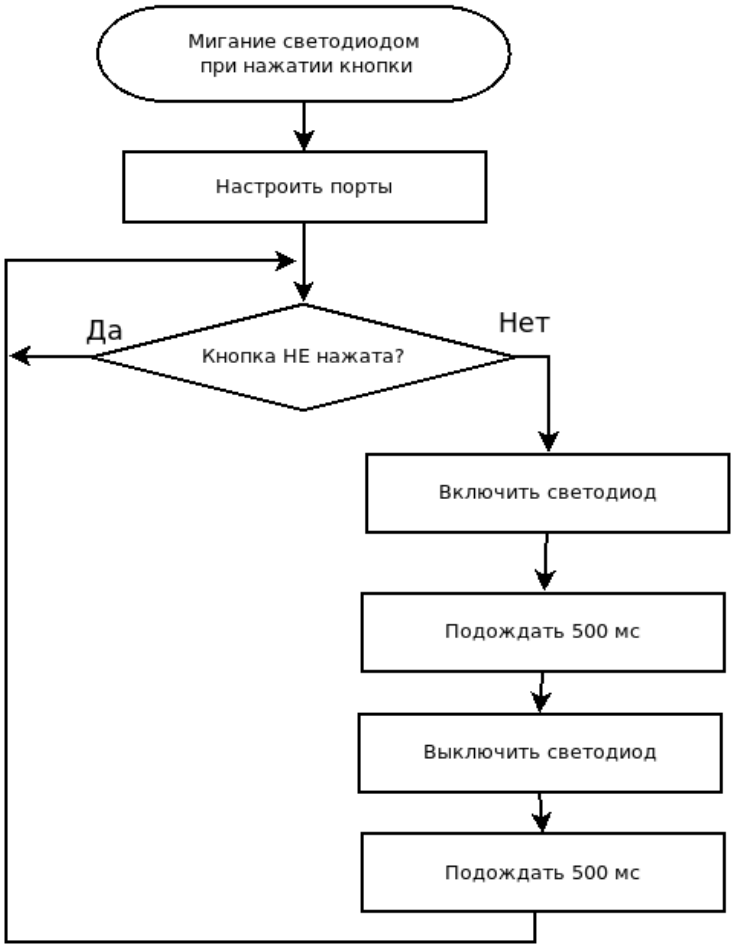

Рисунок 24 – Алгоритм первого задания

В данном алгоритме задержка в 0,5 с, примененная дважды, соответствует частоте 1 Гц.

Напишем программный код для каждого блока алгоритма

Настройка порта: Светодиод по условию подключаем к линии PB0, а кнопка – к PB1. Нам нужно настроить линию PB0 на вывод, а PB1 – на ввод Для настройки порта будем записывать данные в регистр управления порта B DDRB. Сделаем это так же, как это было сделано в предыдущей лабораторной работе:

```
DDRB=0b00000001;
```

То же самое, но более просто и наглядно можно было бы выполнить следующим образом, с использованием битовых операций:

```
DDRB=1<<PB0;
```

Включение светодиода – это операция вывода данных (так как контроллер управляет светодиодом, а не наоборот). Для ее выполнения будем использовать

регистр вывода данных порта B PORTB. Для того, чтобы включить светодиод, нам нужно выдать логическую единицу на линию PB0. С использованием битовых операций этот код будет выглядеть так:

```
PORTB=PORTB|(1<<PB0);
```

Для выполнения программной задержки (чтобы подождать 0,5 с), используем функцию задержки:

```
_delay_ms(500);
```

Время здесь задается в миллисекундах!

Выключение светодиода – это операция вывода данных (так как контроллер управляет светодиодом, а не наоборот). Для ее выполнения будем использовать регистр вывода данных порта B PORTB (см таблицу выше). Для того, чтобы выключить светодиод, нам нужно выдать логический 0 на линию PB0. С помощью битовых операций это можно сделать следующим образом:

Если включение светодиода выполнялось командой

```
PORTB=PORTB|(1<<PB0);
```

То для выключения мы поменяем операцию ИЛИ на И и применим операцию НЕ (~) к маске:

```
PORTB=PORTB&~(1<<PB0);
```

Для переходов внутри алгоритма используем оператор безусловного перехода goto. Разместим метку, чтобы обозначить, КУДА будет происходить переход. Придумаем имя метки

```
my_label:
```

В том месте, откуда будет происходить переход, поставим оператор goto и укажем в нем имя метки my_label.

```
goto my_label;
```

Внимание: Если у вас в программе несколько меток, то им нужно давать разные имена, например my_label1, my_label2

По условию задачи, кнопка при нажатии соединяет линию PB1 с общим проводом, а когда не нажата – на эту линию поступает напряжение питания через резистор. Это значит, что когда кнопка нажата, на PB1 будет 0, а когда не нажата – 1. Так как порт используется также для светодиода, то нам придется выделить из порта только один бит, отвечающий за PB1 с помощью операции И:

PINB&(1<<PB1)

Соответственно, условие «Кнопка не нажата» будет выглядеть так:

if((PINB&(1<<PB1))!=0)

Припишем строки программы, которые получились у нас, к алгоритму, как показано на рисунке 25:

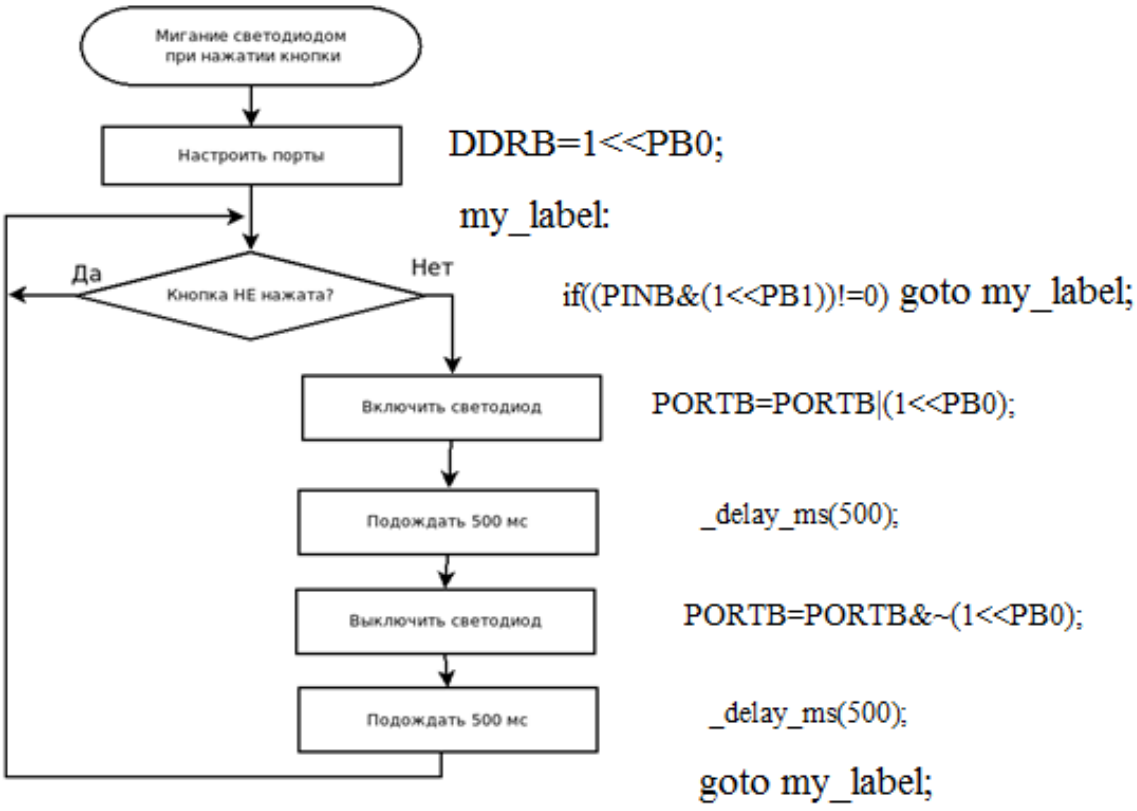

Рисунок 25 - соответствие строк программы алгоритму

Оформим программу согласно правилам оформления программ и снабдим комментариями:

```
#include<avr/io.h> //библиотека ввода-вывода
#include<avr/delay.h> //библиотека с функцией задержки _delay_ms
void main () //заголовок главной функции программы
{
DDRB=1<<PB0;//настраиваем линию PB0 на вывод
my_label: //метка, куда переходить
if((PINB&(1<<PB1))!=0)goto my_label; //переходим назад, если кнопка не нажата
PORTB=PORTB|(1<<PB0);//включаем светодиод
```

```
_delay_ms(500); //ждем 500 мс
PORTB=PORTB&~(1<<PB0);//включаем светодиод
_delay_ms(500); //ждем 500 мс
goto my_label; //переходим в начало алгоритма
}
```

Соберем программу . Для этого нужно выбрать пункт меню tools-make all. Если ошибок не было – внизу появится сообщение «> Process Exit Code: 0». Если код не равен 0, то в программе есть ошибка – ее надо устранить и выбрать пункт меню tools-make all заново, пока ошибки не будут устранены

3.2.2 Моделирование проекта

3.2.2.1 Моделирование в среде SimulIDE

Подключим светодиод, как мы это делали в предыдущей лабораторной работе, как показано на рисунке 26.

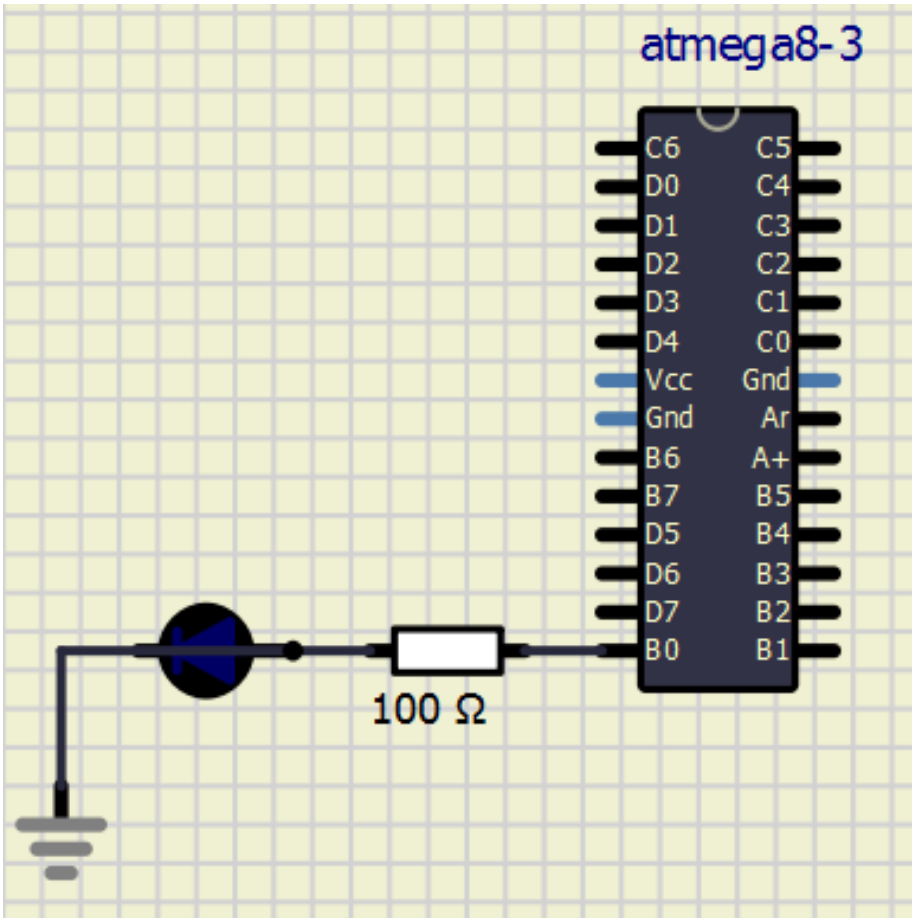

Рисунок 26 – подключение светодиода к микроконтроллеру

Выберем элементы «Линия питания» и «Кнопка», показанные на рисунке 27 , а также – еще один резистор.

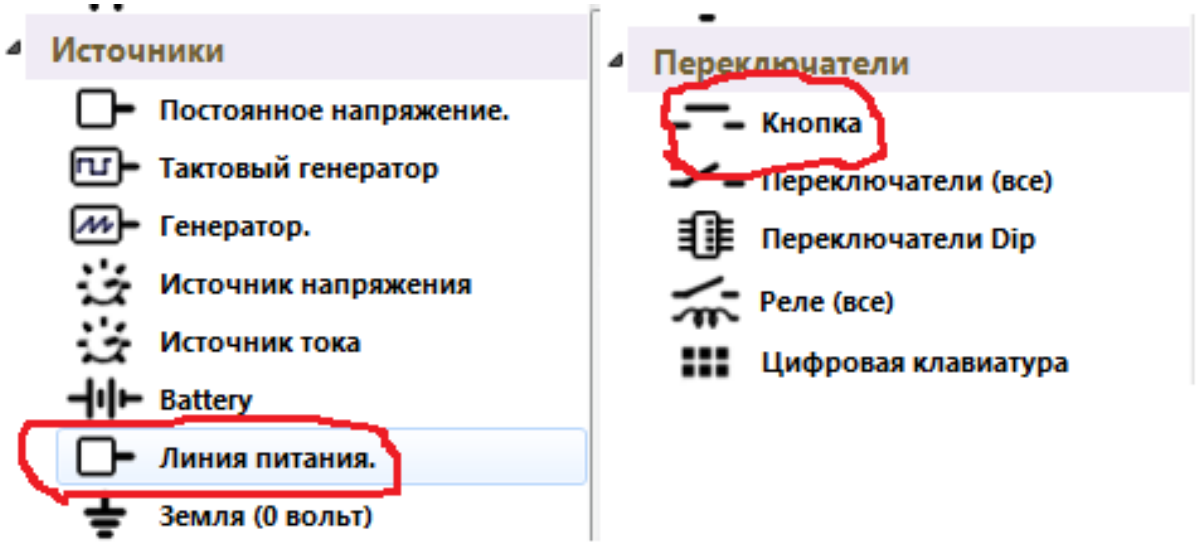

Рисунок 27 – Линия питания и кнопка на панели элементов

Затем разместим эти элементы на схеме, как показано на рисунке 28.

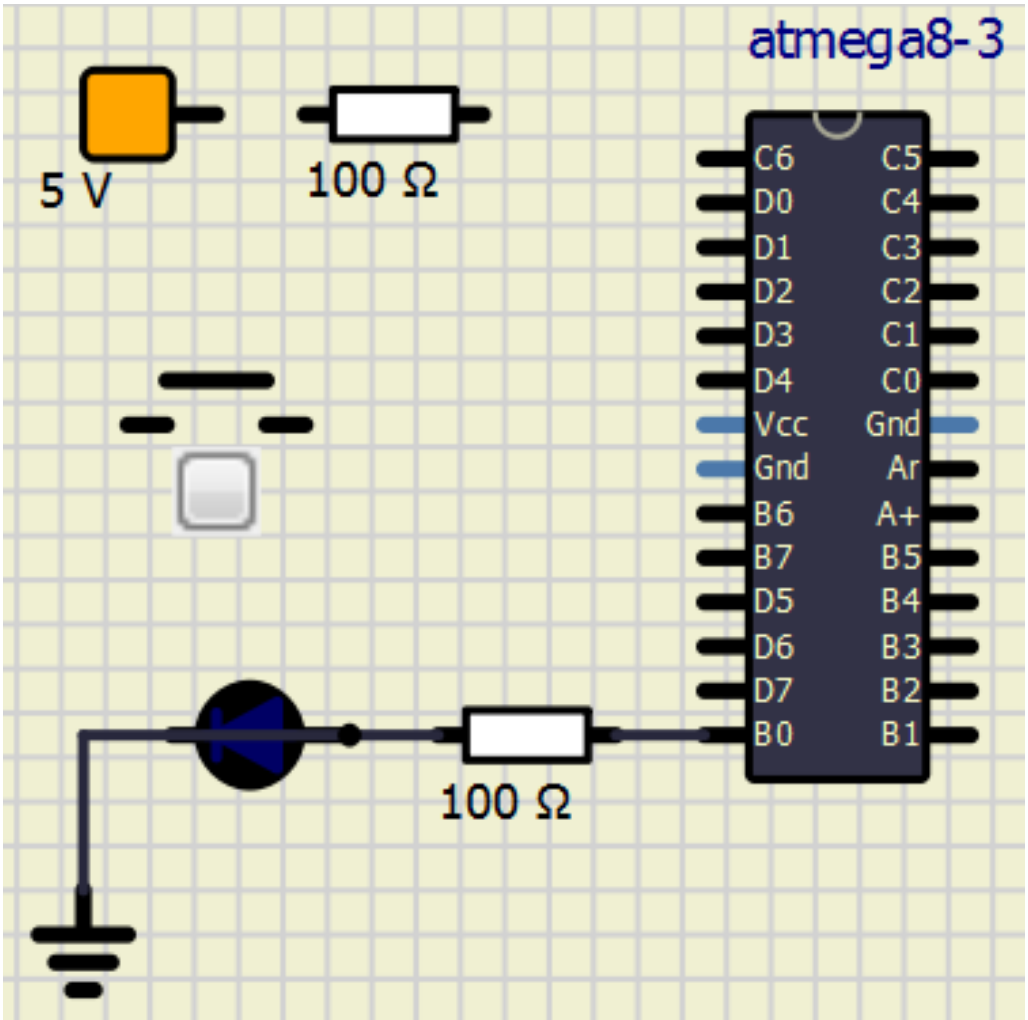

Рисунок 28 – размещение элементов на схеме

Соединим эти элементы и подключим к линии B1 контроллера согласно варианту, как показано на рисунке 29.

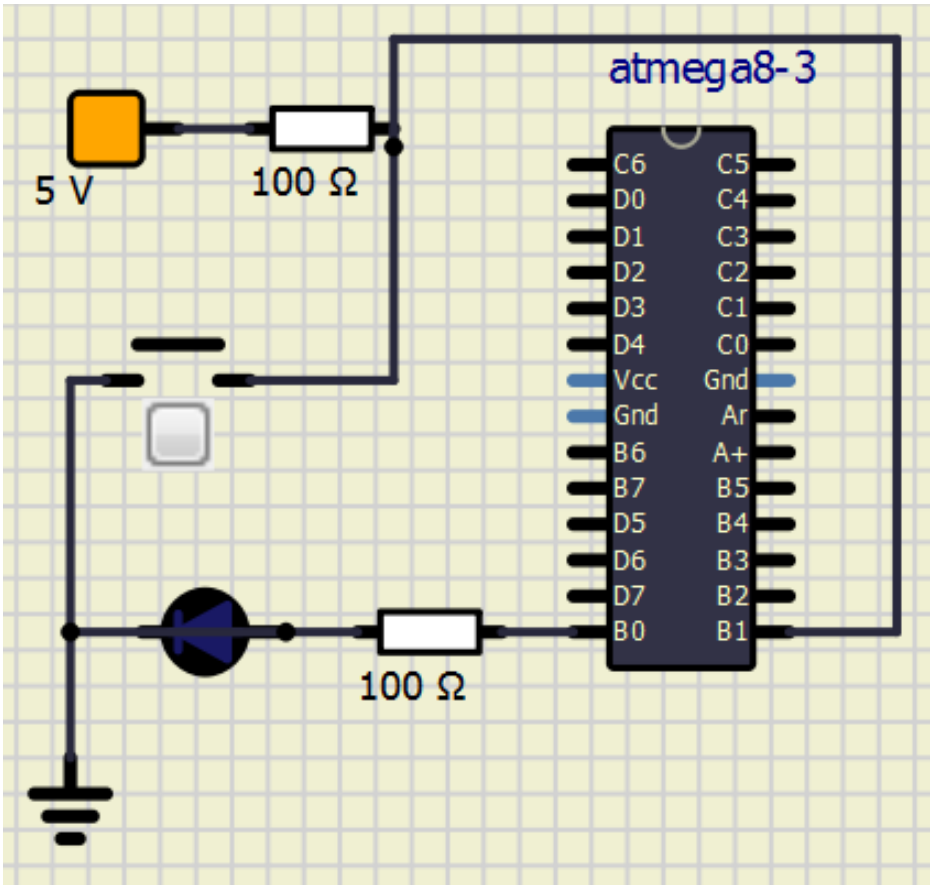

Рисунок 29 – соединение элементов устройства

3.2.2.2 Разработка схемы в программе Proteus

Выберем необходимые компоненты
-Терминал (шина) питания +5В, как показано на рисунке 30

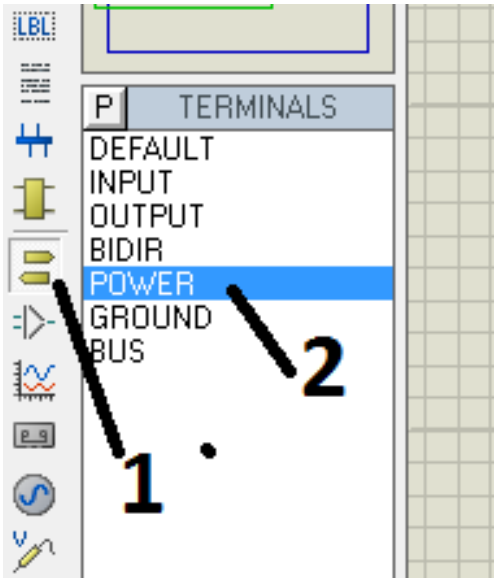

Рисунок 30 – выбор линии питания

-Кнопку, как показано на рисунке 31.

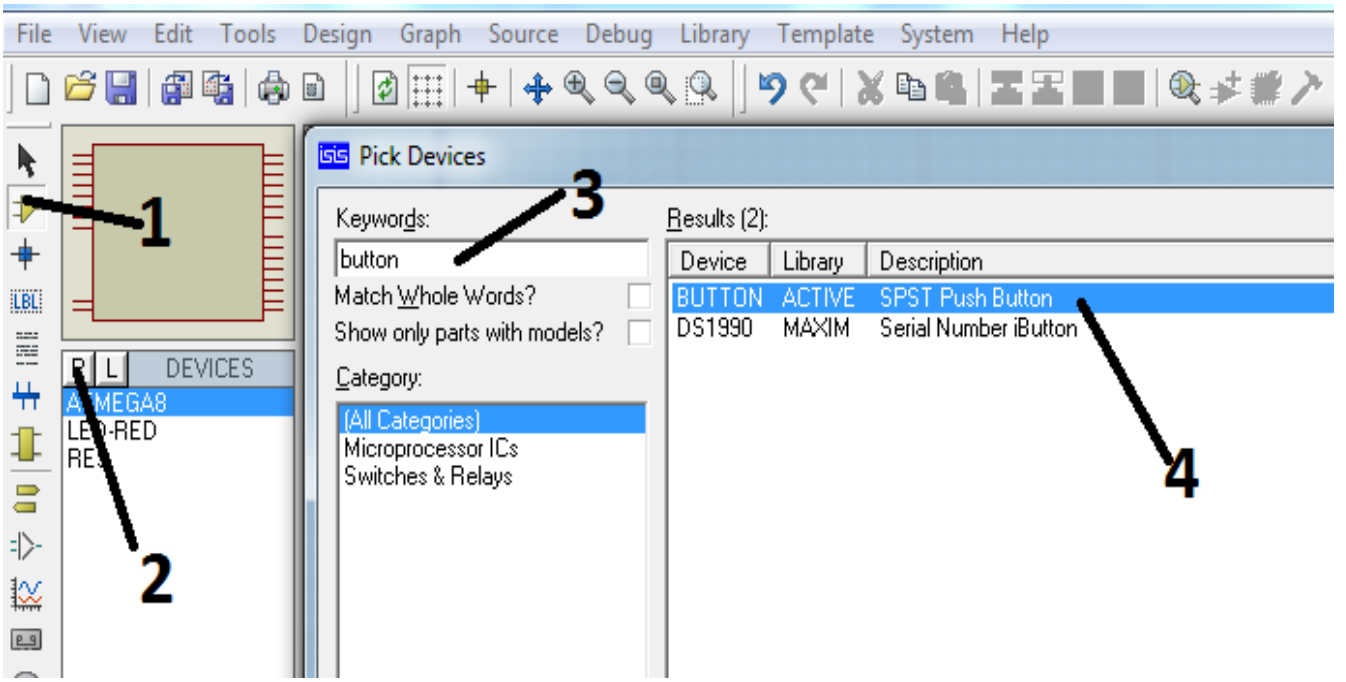

Рисунок 31 – выбор кнопки

Также нам понадобится еще один резистор. Подключим кнопку согласно схеме, как показано на рисунке 32.

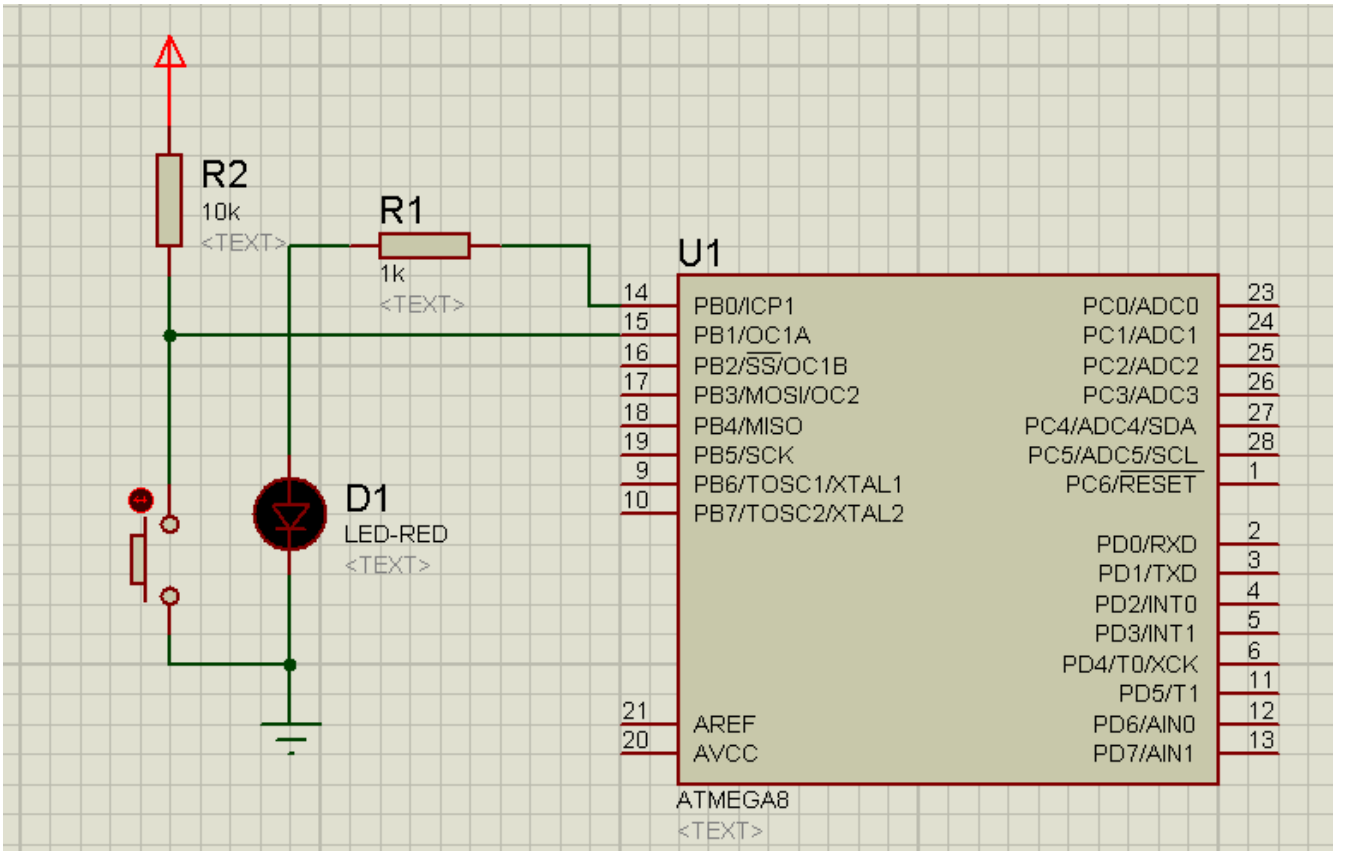

Рисунок 32 – итоговая схема устройства.

3.3 Задание 2

Реализовать алгоритм (показан на рисунке 33), который переключает состояние светодиода при каждом нажатии кнопки

Для проверки, нажата ли кнопка, использовать аналогичный оператор if, только не равно (!=) заменить на равно (==)

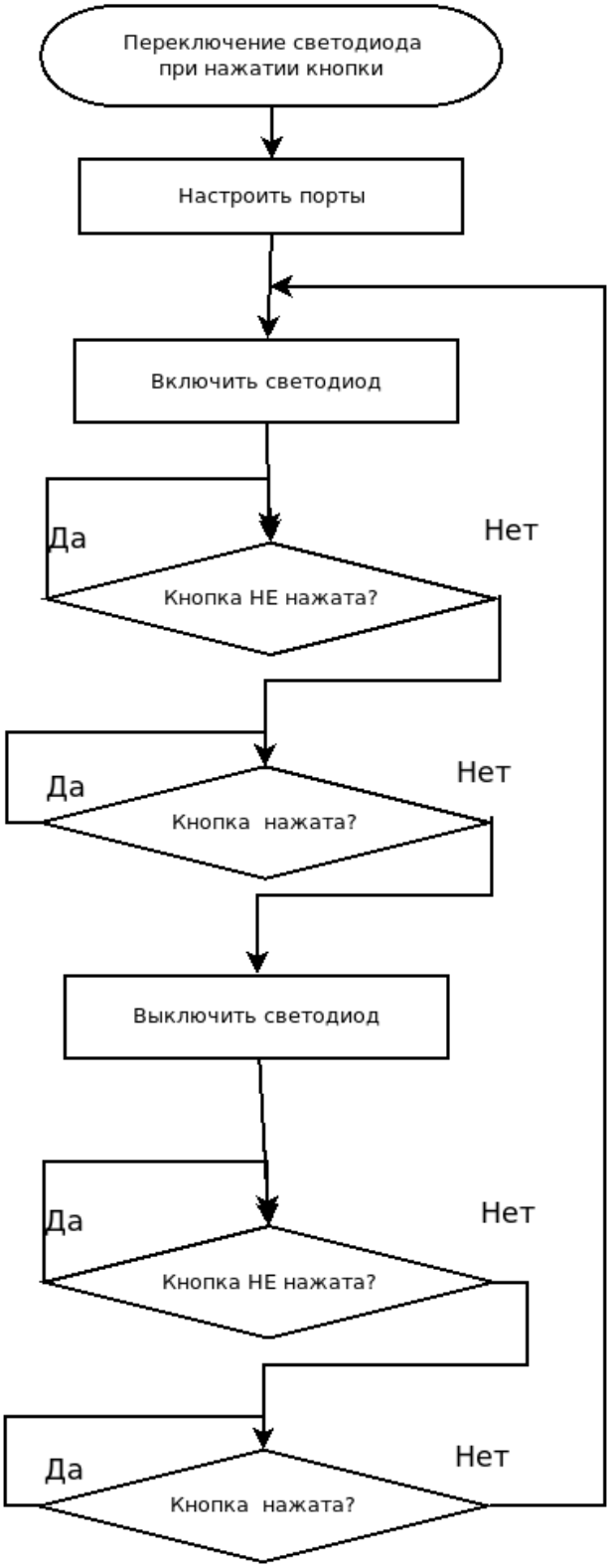

Рисунок 33 – Алгоритм для второго задания

3.4 Контрольные вопросы

1. Что значит «Включить бит» и «Выключить бит»?
2. Как с помощью битовых операций включить произвольный бит в регистре?
3. Как с помощью битовых операций выключить произвольный бит в регистре?
4. Как с помощью битовых операций проверить, равен ли произвольный бит в регистре нулю?
5. Как с помощью битовых операций проверить, равен ли произвольный бит в регистре единице?

4 Лабораторная работа №4 - Работа с семисегментным индикатором

4.1 Цели и задачи работы

Цель – научиться работать с семисегментным светодиодным индикатором.

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

4.2 Задание 1 – вывод символа на индикатор

Разработать программу, которая выводит заданный символ на семисегментный индикатор с общим катодом, подключенный к порту D (символ взять из таблицы 8).

Таблица 8 – варианты лабораторной работы

Вариант	Пример	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Символ	0	1	2	3	4	5	6	7	8	9	A	C	d	b	E	F	H

4.3 Разработка схемы устройства

4.3.1 Разработка схемы в среде SimulIDE

Выберите индикатор на панели компонентов и перетащите его на схему, как показано на рисунке 34.

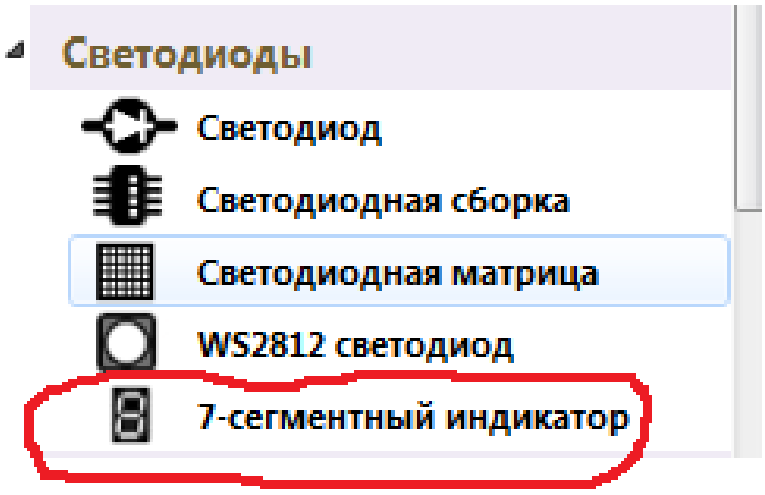

Рисунок 34 – светодиодный индикатор на панели компонентов

Соедините индикатор с контроллером, как показано на рисунке 35. Для удобства работы можно развернуть индикатор на 90 градусов против часовой стрелки, для чего по нему нужно щелкнуть правой кнопкой «мыши» и выбрать соответствующий пункт в контекстном меню.

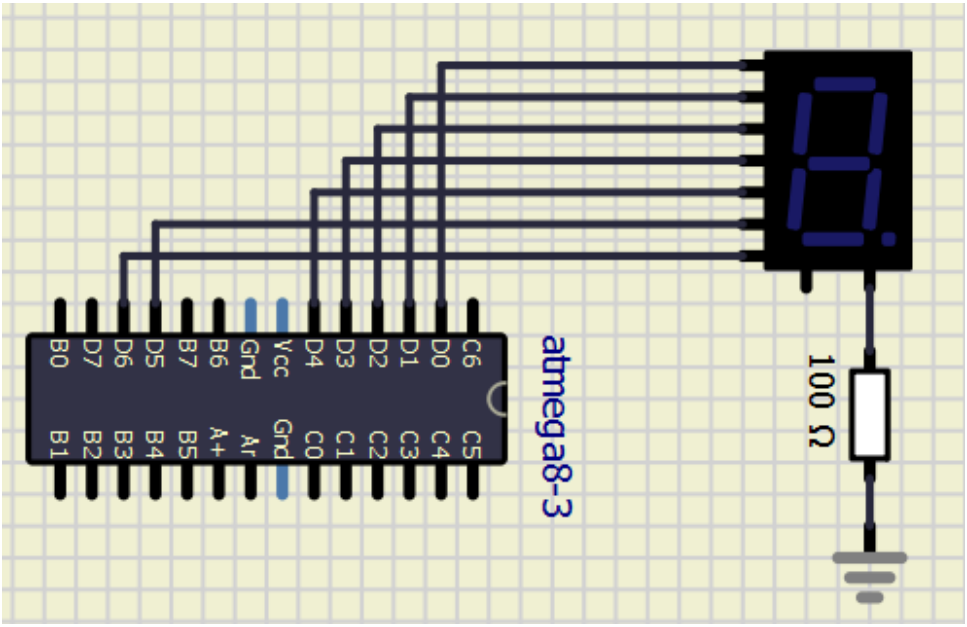

Рисунок 35 – соединение индикатора с микроконтроллером

4.3.2 Разработка схемы в программе Proteus

Сначала выберите индикатор, как показано на рисунке 36

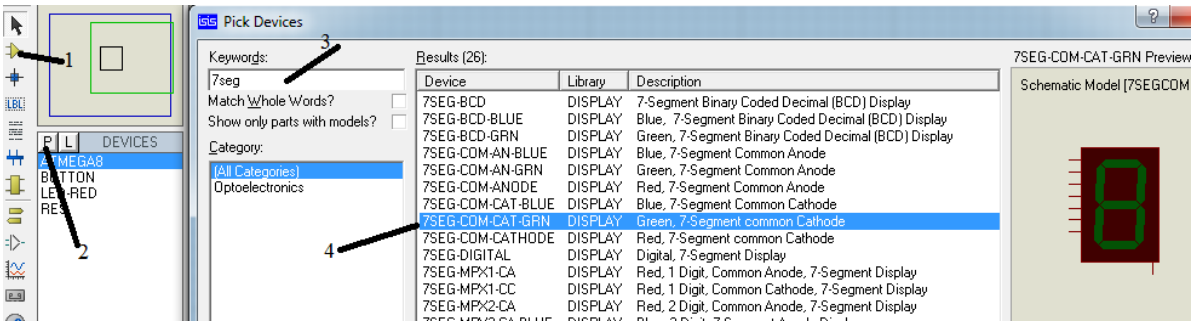

Рисунок 36 – выбор индикатора из списка

Затем подключим его к контроллеру, как показано на рисунке 38.

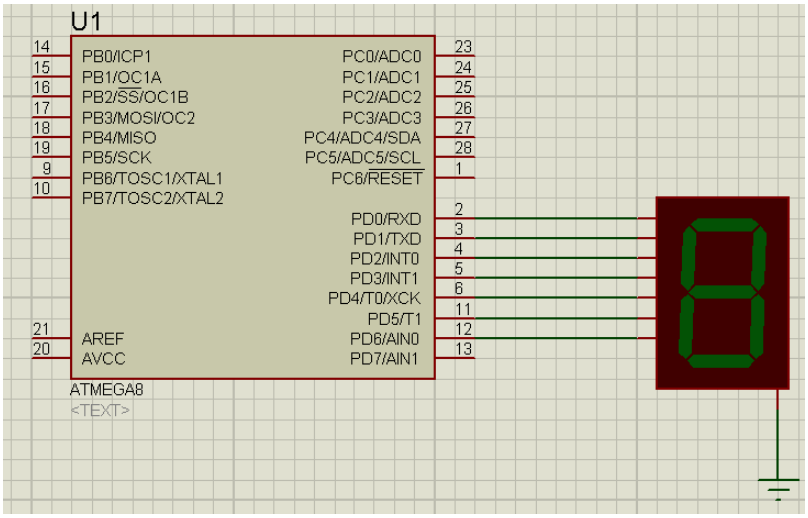

Рисунок 38 - соединение индикатора с микроконтроллером

4.4 Разработка программы

Разработаем алгоритм (в данном случае он очень прост и его можно задать в словесном виде):

1. ***Настроить порт***
2. ***Выдать в порт число, задающее цифру***
3. ***Перейти к пункту 3 (зациклиться)***

Напишем программный код для каждого блока алгоритма

Настройка порта: Индикатор подключен к линиям PD0-PD6.

Нам нужно настроить линии PD0-PD6 на вывод Для настройки порта будем записывать данные в регистр управления порта D DDRD:

```
DDRD=0b01111111;
```

Задание цифры (управление сегментами)

Включение сегментов – это операция вывода данных (так как контроллер управляет индикатором, а не наоборот). Для ее выполнения будем использовать регистр вывода данных порта D PORTD.

По условию нам нужно вывести цифру 0. Обратимся к условному обозначению индикатора, которое показано на рисунке 39.

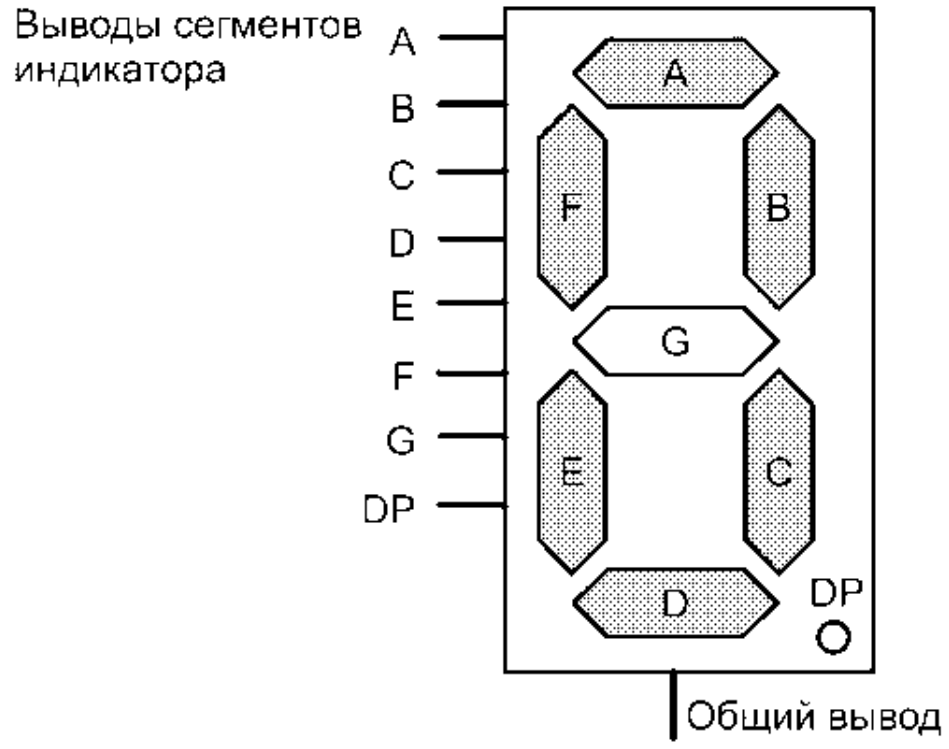

Рисунок 39 – Условное обозначение сегмента индикатора

Для вывода цифры 0 нам нужно включить сегменты A,B,C,D,E и F. Обратимся к участку схемы, который показан на рисунке 40.

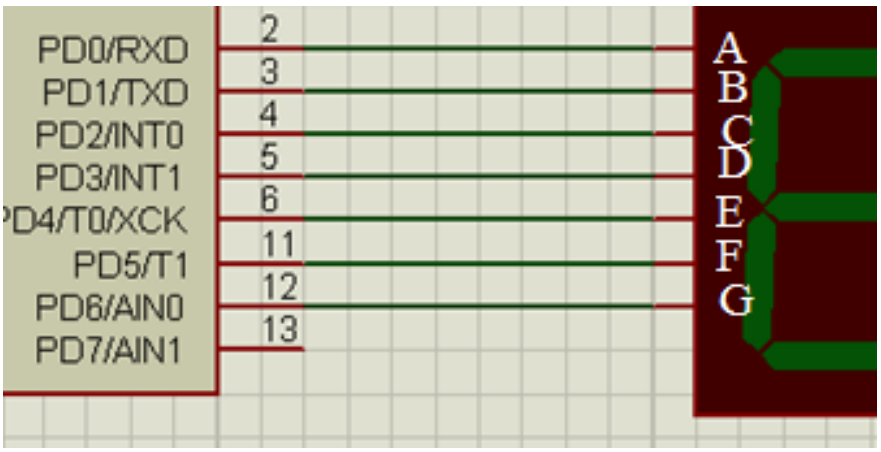

Рисунок 40 – подключение индикатора к контроллеру

Нам нужно выдать логические единицы на линиях PD0…PD5, которые соответствуют сегментам A…F, как показано в таблице 9

Таблица 9 – вывод цифры 0 на индикатор

PD7 ---	PD6 G	PD5 F	PD4 E	PD3 D	PD2 C	PD1 B	PD0 A
0	0	1	1	1	1	1	1

Получилось двоичное число 00111111, его и выдаем в порт D:

```
PORTD=0b00111111;
```

Для выполнения зацикливания используем оператор goto:

```
my_label:
goto my_label;
```

Припишем строки программы, которые получились у нас, к алгоритму и оформим программу согласно правилам оформления программ и снабдим комментариями:

```
#include<avr/io.h> //библиотека ввода-вывода
void main () //заголовок главной функции программы
{
DDRD=0b01111111;//Настроить порт
PORTD=0b00111111;Выдать в порт число, задающее цифру
my_label: //метка, куда переходить
goto my_label; //зацикливаем программу
}
```

Соберем программу ,для этого нужно выбрать пункт меню tools-make all. Если ошибок не было – внизу появится сообщение «> Process Exit Code: 0». Если код не равен 0, то в программе есть ошибка – ее надо устранить и выбрать пункт меню tools-make all заново, пока ошибки не будут устранены

Нажимаем кнопку “Play” и проверяем работу программы

4.5 Дополнительные задания

Задание 2: Выдать ту же цифру, но на индикатор с общим анодом (для этого замените на схеме индикатор и в строке для выдачи данных в порт нули замените на единицы и наоборот)

Задание 3: Написать программу, которая по очереди выводит на индикатор цифры вашей даты рождения

Задание 4: Написать программу, которая при нажатии на кнопку меняет цифру 0 на цифру 1 на индикаторе и наоборот

4.6 Контрольные вопросы

1. Как соединяются светодиоды внутри семисегментного индикатора?
2. Чем отличаются индикаторы с общим анодом от индикаторов с общим катодом?
3. Каким образом можно выключить все сегменты на индикаторе (погасить его)?

5 Лабораторная работа №5 - Динамическая индикация

5.1 Цели и задачи работы

Цель – научиться работать с семисегментным светодиодным индикатором в режиме динамической индикации, освоить оператор выбора и применение табличной выборки

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

5.2 Задание 1

Вывести вашу дату рождения (месяц и день) на четырехразрядный семисегментный индикатор, который подключен к микроконтроллеру, как показано в таблице 10

Таблица 10 – подключение индикатора к контроллеру

Линия индикатора	A	B	C	D	E	F	G	DP/ точка	1	2	3	4
Линия контроллера	PD0	PD1	PD2	PD3	PD4	PD5	PD6	PD7	PC0	PC1	PC2	PC3

5.3 Разработка схемы

5.3.1 Разработка схемы в программе SimulIDE

Разместите на схеме контроллер и семисегментный индикатор, аналогично предыдущей лабораторной работе. Щелкните правой кнопкой «мыши» по индикатору и выберите пункт «Свойства» в появившемся меню. Откроется окно, в котором нужно ввести цифру «4» в поле «Количество разрядов». После этого индикатор примет вид, показанный на рисунке 41.

Рисунок 41 – четырехразрядный индикатор

Подключите индикатор к контроллеру, как показано на рисунке 42

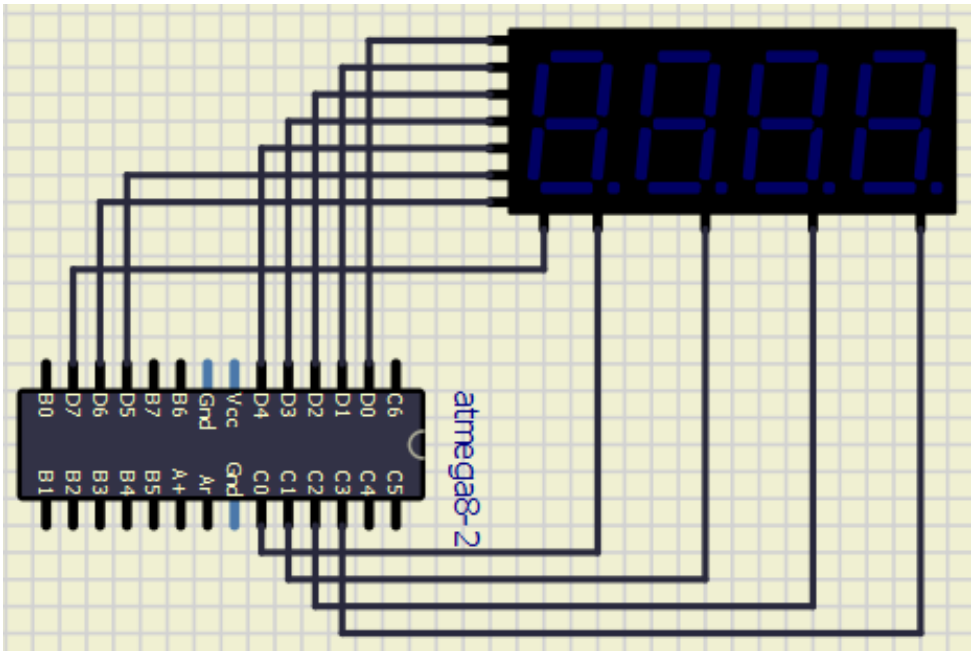

Рисунок 42 – подключение индикатора

5.3.2 Разработка схемы в программе Proteus

В отличие от программы SimulIDE, в программе Proteus четырехразрядный индикатор – это отдельный компонент. Выберите его. Как показано на рисунке 43

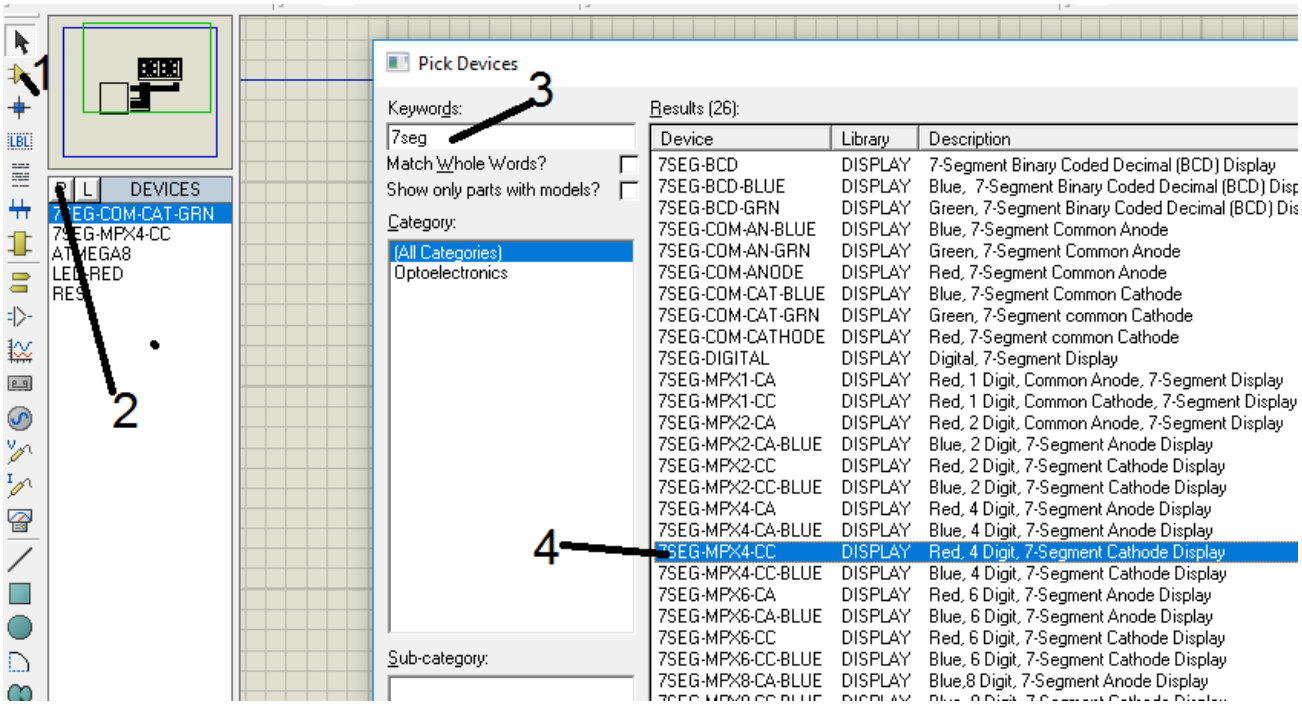

Рисунок 43 – выбор индикатора из списка компонентов

Затем разместите его на схеме, как показано на рисунке 44.

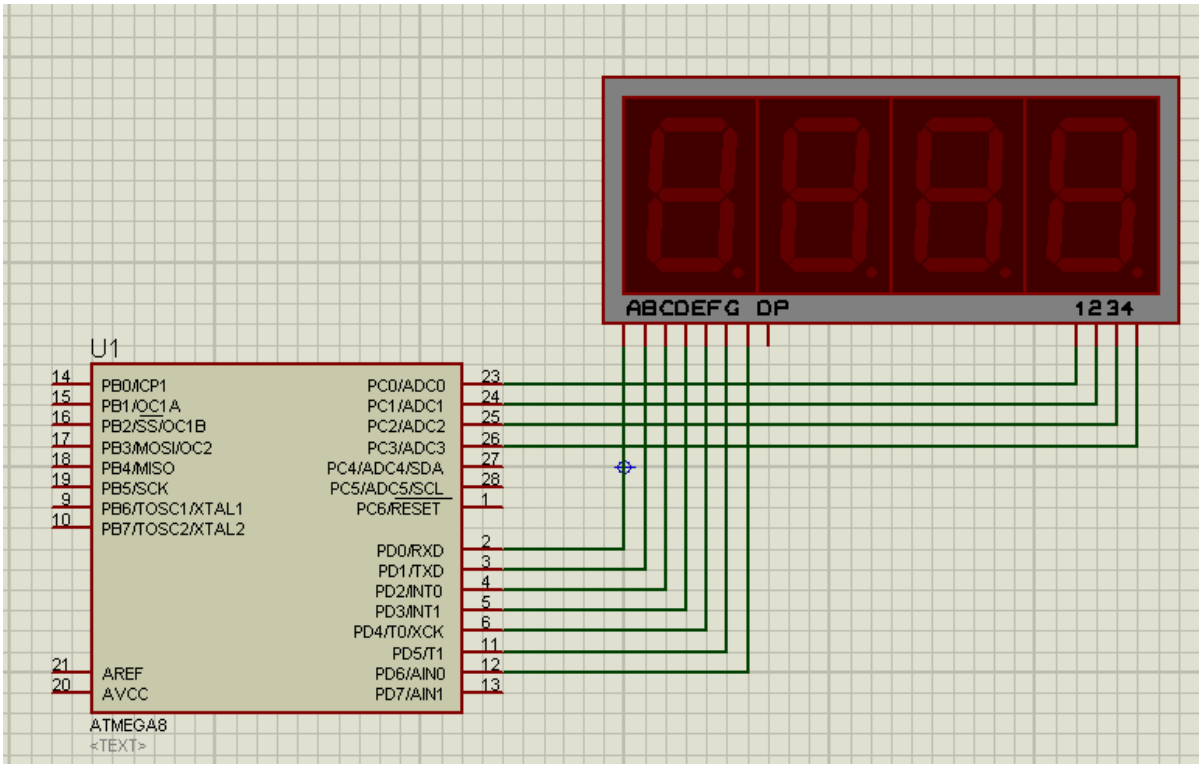

Рисунок 44 – подключение индикатора к контроллеру

5.4 Разработка программы.

В данном случае мы реализуем алгоритм динамической индикации, который описан ниже:

1. Выключить все разряды индикатора
2. Выдать код первой цифры на выводы сегментов
3. Включить первый разряд
4. Подождать 5 мс
5. Выключить все разряды индикатора
6. Выдать код второй цифры на выводы сегментов
7. Включить второй разряд
8. Подождать 5 мс
9. Выключить все разряды индикатора
10. Выдать код третьей цифры на выводы сегментов
11. Включить третий разряд
12. Подождать 5 мс
13. Выключить все разряды индикатора
14. Выдать код четвертой цифры на выводы сегментов
15. Включить четвертый разряд
16. Подождать 5 мс
17. Перейти к пункту 1

Для примера выведем число 1988. Составим таблицу (таблица 11) для вывода всех цифр на индикатор (имена сегментов показаны на рисунке 39)

Индикатор с общим анодом, поэтому 1 – сегмент светится, 0 – сегмент не светится.

Таблица 11 – коды для вывода цифр на индикатор

цифра	PD7 H	PD6 G	PD5 F	PD4 E	PD3 D	PD2-C	PD1-B	PD0 A
0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1
6	0	1	1	1	1	1	0	1
7	0	0	0	0	0	1	1	1
8	0	1	1	1	1	1	1	1
9	0	1	1	0	1	1	1	1

Выпишем из таблицы коды для вывода нужных нам цифр:

```
1: PORTD=0b00000110;
9: PORTD=0b01101111;
8: PORTD=0b01111111;
```

Для управления разрядами (которые у нас подключены к порту С) составим отдельную таблицу 12. Учтем, что логические уровни для управления разрядами противоположны уровням для управления сегментами, поэтому включенному разряду будет соответствовать 0, а выключенному – 1[4].

Таблица 12 – управление разрядами индикатора

Состояние	PC3/Разряд 4	PC2/Разряд 3	PC1/Разряд 2	PC0/Разряд 1
Все разряды выключены	1	1	1	1
Включен первый разряд	1	1	1	0
Включен второй разряд	1	1	0	1
Включен третий разряд	1	0	1	1
Включен четвертый разряд	0	1	1	1

То есть, для выключения всех разрядов нужно выполнить команду PORTC=0b1111; для включения первого разряда – PORTC=0b1110 и так далее

Подпишем к пунктам алгоритма программный код:

```
#include<avr/io.h> //библиотека ввода-вывода
#include<avr/delay.h>
void main () //заголовок главной функции программы
{

// настраиваем порты
DDRD=0b1111111;
DDRC=0b1111;
metka:
PORTC=0b1111;// Выключить все разряды индикатора
PORTD=0b00000110;// Выдать код первой цифры на выводы сегментов
PORTC=0b1110;// Включить первый разряд
_delay_ms(5);// Подождать 50 мс
PORTC=0b1111;// Выключить все разряды индикатора
PORTD=0b01101111;// Выдать код второй цифры на выводы сегментов
PORTC=0b1101;// Включить второй разряд
_delay_ms(5);// Подождать 50 мс
PORTC=0b1111;//Выключить все разряды индикатора
PORTD=0b01111111;// Выдать код третьей цифры на выводы сегментов
 PORTC=0b1011;//Включить третий разряд
_delay_ms(5);// Подождать 50 мс
PORTC=0b1111;// Выключить все разряды индикатора
PORTD=0b01111111;// Выдать код четвертой цифры на выводы сегментов
PORTC=0b0111;// Включить четвертый разряд
_delay_ms(5);// Подождать 50 мс
goto metka; //Перейти к пункту 1
 }
```

5.5 Задание 2

Модифицировать проект так, чтобы он выводил не строго заданное число, а число из переменной n (в эту переменную записать ваш год рождения).

Для этого опишем переменные (n-для самого числа и n0…n3 для его разрядов):

```
int n;//переменная для хранения числа
```

```
char n0,n1,n2,n3;//переменные для хранения цифр числа
```

Зададим значение n

```
n=1988;
```

Затем разделим число на отдельные цифры, используя операцию вычисления остатка от деления:

```
//Разделяем число на отдельные цифры:
n0=n%10;//выделяем разряд единиц
n1=(n/10)%10;//выделяем разряд десятков
n2=(n/100)%10;//выделяем разряд сотен
n3=(n/1000)%10;//выделяем разряд тысяч
```

Так как нам нужно выводить не какую-то конкретную, а произвольные цифры, которые хранятся в переменных n0…n3, то для вывода цифры реализуем следующий алгоритм, показанный на рисунке 45.

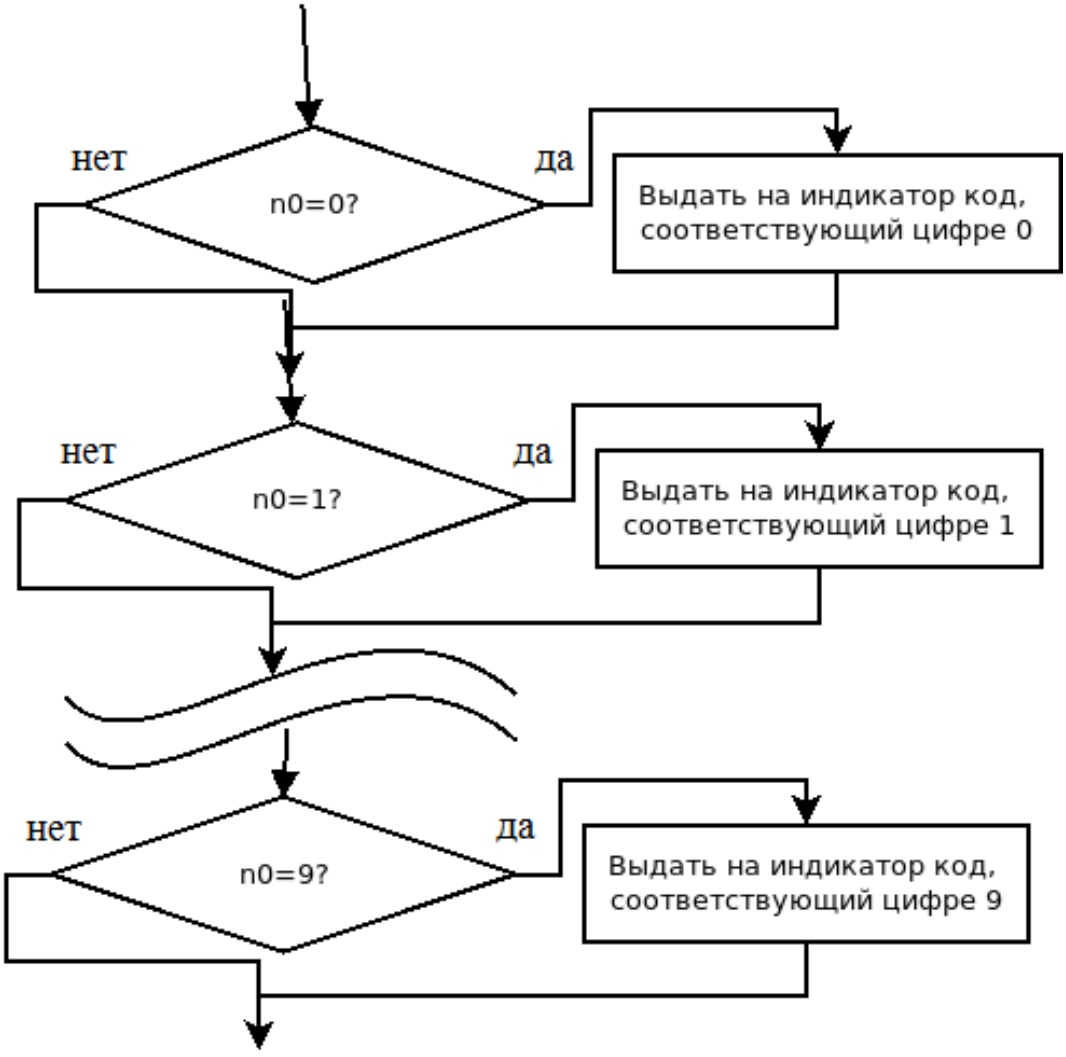

Рисунок 45 – алгоритм вывода произвольной цифры

Он содержит в себе 10 операторов IF. В нашем случае реализация этого алгоритма на языке С будет выглядеть так:

```
if(n3==0)PORTD=0b00111111;//если цифра равна 0, то выводим в порт код нуля для 7-сегментного индикатора
if(n3==1)PORTD=0b00000110;
if(n3==2)PORTD=0b01011011;
if(n3==3)PORTD=0b01001111;
if(n3==4)PORTD=0b01100110;
if(n3==5)PORTD=0b01101101;
if(n3==6)PORTD=0b01111101;
if(n3==7)PORTD=0b00000111;
if(n3==8)PORTD=0b01111111;
if(n3==9)PORTD=0b01101111;
```

Заменим в четырех местах код для вывода цифры в предыдущем проекте на этот код и в итоге получим программу следующего вида (не забудьте в нужных местах заменить n3 на n0…n2):

```
#include<avr/io.h> //библиотека ввода-вывода
#include<avr/delay.h>
void main () //заголовок главной функции программы
{
int n;//переменная для хранения числа
char n0,n1,n2,n3;//переменные для хранения цифр числа

n=5*4*3*2*1;
//Разделяем число на отдельные цифры:
n0=n%10;//выделяем разряд единиц
n1=(n/10)%10;//выделяем разряд десятков
n2=(n/100)%10;//выделяем разряд сотен
n3=(n/1000)%10;//выделяем разряд тысяч
// настраиваем порты
DDRD=0b1111111;
DDRC=0b1111;
metka:
PORTC=0b1111;// Выключить все разряды индикатора
//Выводим разряд тысяч
if(n3==0)PORTD=0b00111111;//если цифра равна 0, то выводим в порт код нуля для 7-сегментного индикатора
if(n3==1)PORTD=0b00000110;
if(n3==2)PORTD=0b01011011;
if(n3==3)PORTD=0b01001111;
if(n3==4)PORTD=0b01100110;
```

```
        if(n3==5)PORTD=0b01101101;
        if(n3==6)PORTD=0b01111101;
        if(n3==7)PORTD=0b00000111;
        if(n3==8)PORTD=0b01111111;
        if(n3==9)PORTD=0b01101111;
        PORTC=0b1110;// Включить первый разряд
        _delay_ms(5);// Подождать 50 мс
        PORTC=0b1111;// Выключить все разряды индикатора
        //Выводим разряд сотен
        if(n2==0)PORTD=0b00111111;//если цифра равна 0, то выводим в порт код нуля для 7-сегментного индикатора
        if(n2==1)PORTD=0b00000110;
        if(n2==2)PORTD=0b01011011;
        if(n2==3)PORTD=0b01001111;
        if(n2==4)PORTD=0b01100110;
        if(n2==5)PORTD=0b01101101;
        if(n2==6)PORTD=0b01111101;
        if(n2==7)PORTD=0b00000111;
        if(n2==8)PORTD=0b01111111;
        if(n2==9)PORTD=0b01101111;
        PORTC=0b1101;// Включить второй разряд
        _delay_ms(5);// Подождать 50 мс
        PORTC=0b1111;//Выключить все разряды индикатора
        //Выводим разряд десятков
        if(n1==0)PORTD=0b00111111;//если цифра равна 0, то выводим в порт код нуля для 7-сегментного индикатора
        if(n1==1)PORTD=0b00000110;
        if(n1==2)PORTD=0b01011011;
        if(n1==3)PORTD=0b01001111;
        if(n1==4)PORTD=0b01100110;
        if(n1==5)PORTD=0b01101101;
        if(n1==6)PORTD=0b01111101;
        if(n1==7)PORTD=0b00000111;
        if(n1==8)PORTD=0b01111111;
        if(n1==9)PORTD=0b01101111;
         PORTC=0b1011;//Включить третий разряд
        _delay_ms(5);// Подождать 50 мс
        PORTC=0b1111;// Выключить все разряды индикатора
        PORTD=0b1100110;// Выдать код четвертой цифры на выводы сегментов
        //Выводим разряд единиц
        if(n0==0)PORTD=0b00111111;//если цифра равна 0, то выводим в порт код нуля для 7-сегментного индикатора
        if(n0==1)PORTD=0b00000110;
```

```
if(n0==2)PORTD=0b01011011;
if(n0==3)PORTD=0b01001111;
if(n0==4)PORTD=0b01100110;
if(n0==5)PORTD=0b01101101;
if(n0==6)PORTD=0b01111101;
if(n0==7)PORTD=0b00000111;
if(n0==8)PORTD=0b01111111;
if(n0==9)PORTD=0b01101111;
 PORTC=0b0111;//Включить четвертый разряд
_delay_ms(5);// Подождать 50 мс
goto metka; //Перейти к пункту 1
}
```

5.6 Задание 3

Замените в предыдущем проекте оператор if на оператор case.
Пример:

```
switch(n0)
{
        case 0:PORTD=0b00111111;break;
        case 1:PORTD=0b00000110;break;
        case 2:PORTD=0b01011011;break;
        case 3:PORTD=0b01001111;break;
        case 4:PORTD=0b01100110;break;
        case 5:PORTD=0b01101101;break;
        case 6:PORTD=0b01111101;break;
        case 7:PORTD=0b00000111;break;
        case 8:PORTD=0b01111111;break;
        case 9:PORTD=0b01101111;break;
}
```

5.7 Задание 4 - использование табличной выборки

Замените блок операторов If из второго задания на выборку из таблицы
Для этого проделаем следующие шаги:

1. Опишем массив – таблицу символов и введем наши коды в этот массив:

```
char digits[10]={ 0b00111111,0b00000110,0b01011011 ……..  }
```

(нужно через запятую указать коды всех цифр от 0 до 9)

2. Замените блок операторов if на следующую строку:

```
PORTD=digits[n3];
```

*где необходимо, вместо n3 указать n2…n0

5.8 Контрольные вопросы

1. Для чего нужна динамическая индикация?
2. Как соединены светодиоды во многоразрядном индикаторе?
3. Как быстро нужно переключать цифры при динамической индикации?
4. Для чего нужно гасить индикатор при выдаче кода очередной цифры?

6 Лабораторная работа №6 - работа с матричной клавиатурой

6.1 Цели и задачи работы

Цель – научиться работать с матричной клавиатурой, выполнять программный опрос клавиатуры

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

6.2 Задание работы

Подключить к контроллеру матричную клавиатуру 3*3 и одноразрядный семисегментный индикатор. Разработать программу, которая выводит номер клавиши при нажатии на клавишу (если ни одна клавиша не нажата – вывести 0)

Примерный алгоритм программы:

1. Настроить порты
2. Бесконечно выполнять в цикле следующие действия

2.1 Провести опрос клавиатуры, записать результат в переменную

2.2 Вывести число из переменной на индикатор

Индикатор с общим катодом подключить к порту D (A-PD0, B-PD1….G-PD6). Клавиатуру подключить согласно варианту, как показано в таблице 13.

Таблица 13 – варианты лабораторной работы

Вариант	Строка 1	Строка 2	Строка 3	Столбец 1	Столбец 2	Столбец 3
Пример	PC0	PC1	PC2	PC3	PC4	PC5
1	PC5	PC0	PC1	PC2	PC3	PC4
2	PC4	PC5	PC0	PC1	PC2	PC3
3	PC3	PC4	PC5	PC0	PC1	PC2
4	PC2	PC3	PC4	PC5	PC0	PC1
5	PC1	PC2	PC3	PC4	PC5	PC0
6	PC5	PC4	PC3	PC2	PC1	PC0
7	PC0	PC5	PC4	PC3	PC2	PC1
8	PC1	PC0	PC5	PC4	PC3	PC2
9	PC2	PC1	PC0	PC5	PC4	PC3
10	PB0	PB1	PB2	PB3	PB4	PB5
11	PB5	PB0	PB1	PB2	PB3	PB4
12	PB4	PB5	PB0	PB1	PB2	PB3
13	PB3	PB4	PB5	PB0	PB1	PB2
14	PB2	PB3	PB4	PB5	PB0	PB1
15	PB1	PB2	PB3	PB4	PB5	PB0
16	PB5	PB4	PB3	PB2	PB1	PB0

6.3 Разработка схемы

6.3.1 Разработка схемы в программе SimulIDE

Начертите схему, как показано на рисунке 46

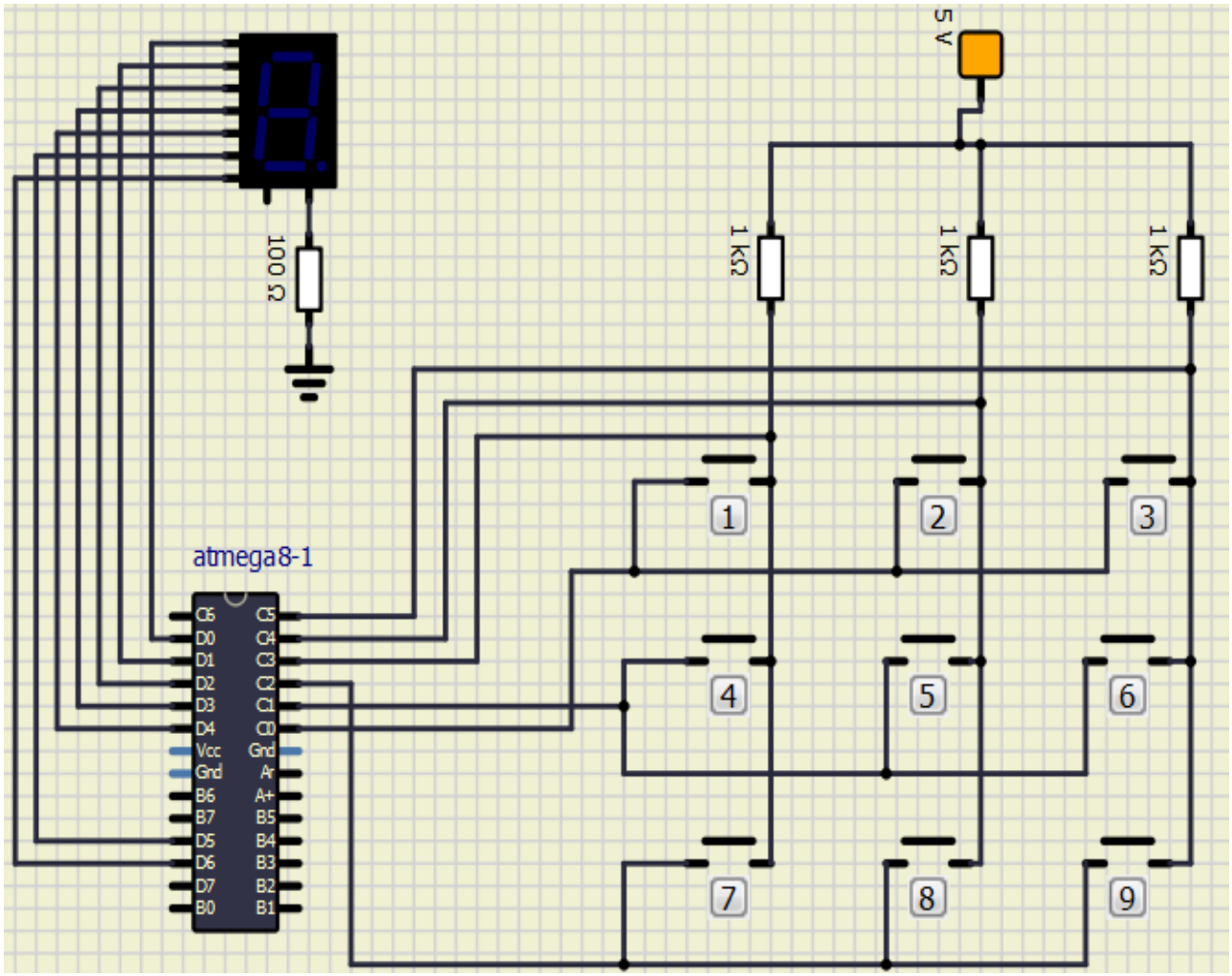

Рисунок 46 – схема устройства

Для данной схемы понадобятся компоненты, которые уже встречались в лабораторной работе №2: линия питания (показана на рисунке 46 оранжевым цветом) и кнопка. Для того, чтобы разместить подписи на кнопках, необходимо навести указатель «мыши» на кнопку так, чтобы указатель принял форму ладони. Далее следует нажать на правую кнопку «мыши», и в появившемся контекстном меню выбрать пункт «Свойства». Откроется окно свойств, в котором нужно ввести третуемую цифру в поле «Надпись».

При соединении кнопок в матрицу следует отратить внимание на то, есть ли символы соединения линий (в виде кружочков) в тех местах, где линии должны быть соединены.

Также не забудьте подсоединить столбцы (вертикальные линии) и строки (горизонтальные линии) матричной клавиатуры к контроллеру согласно вашему варианту задания.

6.3.2 Разработка схемы в программе Proteus

Начертите схему, как показано на рисунке 47

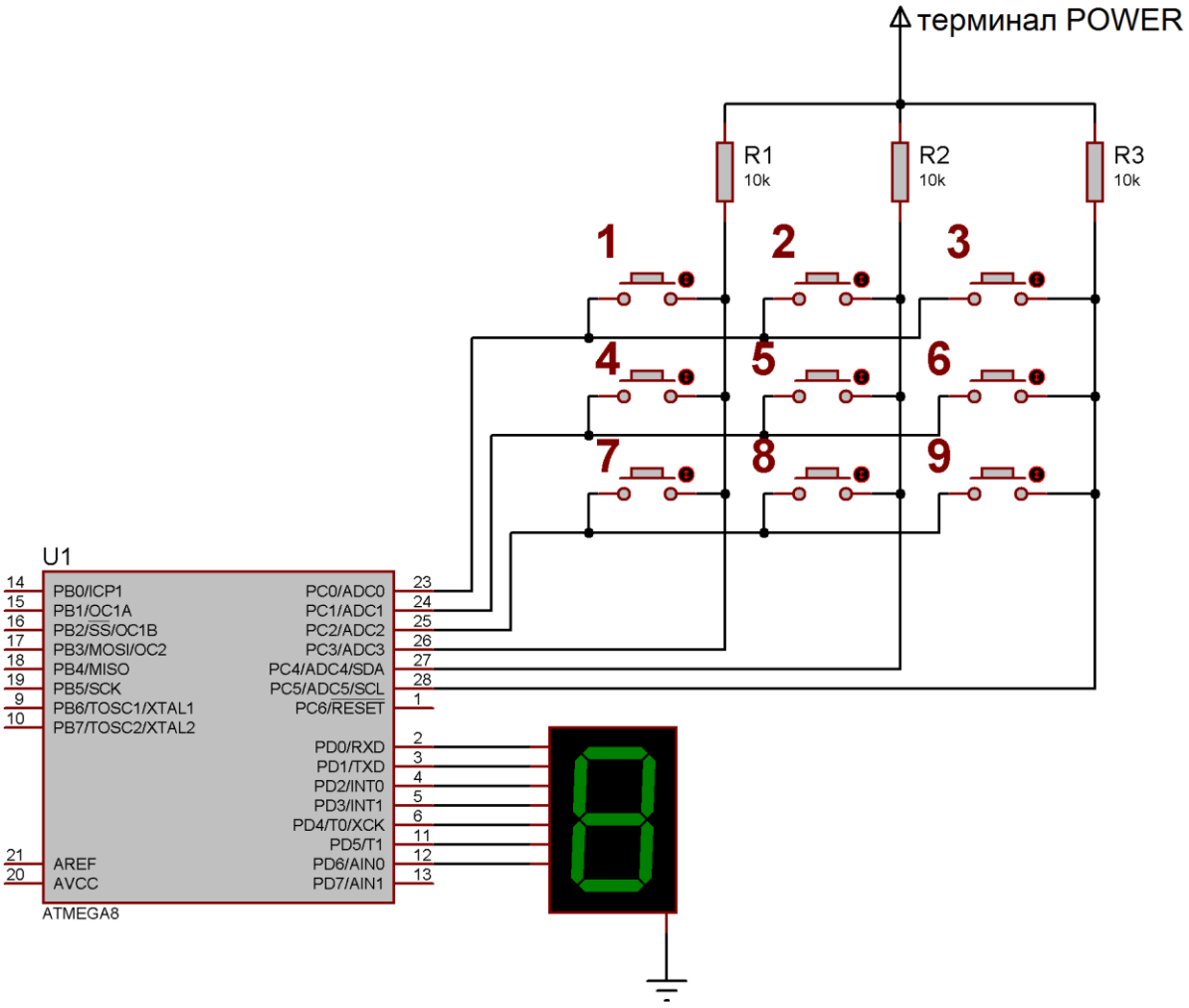

Рисунок 47 – принципиальная схема в программе Proteus

6.4 Разработка программы

Так как кнопки объединены в матрицу, и каждая из кнопок соединяет один столбец с одной строкой клавиатуры, а резисторы задают уровень логической единицы, если ни одна кнопка не нажата, то будем использовать следующий алгоритм опроса клавиатуры:

1. Выдать логический «0» на первую строку клавиатуры
2. Если логический «0» появился на первом столбце, значит нажата клавиша 1

3. Если логический «0» появился на втором столбце, значит нажата клавиша 2
4. Если логический «0» появился на третьем столбце, значит нажата клавиша 3
5. Выдать логический «0» на вторую строку клавиатуры
6. Если логический «0» появился на первом столбце, значит нажата клавиша 4
7. Если логический «0» появился на втором столбце, значит нажата клавиша 5
8. Если логический «0» появился на третьем столбце, значит нажата клавиша 6
9. Выдать логический «0» на третью строку клавиатуры
10. Если логический «0» появился на первом столбце, значит нажата клавиша 7
11. Если логический «0» появился на втором столбце, значит нажата клавиша 8
12. Если логический «0» появился на третьем столбце, значит нажата клавиша 9

Разработаем функцию настройки портов

В ней настроим на вывод линии, к которым подключается индикатор и к которым подключаются строки матрицы клавиатуры

```
void nastroyka_portov()
{
DDRD=0b01111111;//PD0-PD6, куда подключен индикатор – на вывод
DDRC=0b00000111;//PC0-PC2, куда подключены строки матрицы – на вывод

}
```

Разработаем функцию опроса клавиатуры

```
char key()//Функция опроса клавиатуры
{
PORTC=0b00000110; //Выдать лог. 0 на Строку 1, лог. 1 – на остальные строки
_delay_us(500);
if((PINC&0b00001000)==0)return 1;//Если на линии Столбец 1 лог. 0, то нажата кнопка SB1
if((PINC&0b00010000)==0)return 2;//Если на линии Столбец 2 лог. 0, то нажата кнопка SB2
```

```
    if((PINC&0b00100000)==0)return 3;//Если на линии Столбец 3 лог. 0,
то нажата кнопка SB3
    PORTC=0b00000101; //Выдать лог. 0 на Строку 2, лог. 1 – на
остальные строки
    _delay_us(500);
    if((PINC&0b00001000)==0)return 4;//Если на линии Столбец 1 лог. 0,
то нажата кнопка SB4
    if((PINC&0b00010000)==0)return 5;//Если на линии Столбец 2 лог. 0,
то нажата кнопка SB5
    if((PINC&0b00100000)==0)return 6;//Если на линии Столбец 3 лог. 0,
то нажата кнопка SB6
    PORTC=0b00000011; //Выдать лог. 0 на Строку 3, лог. 1 – на
остальные строки
    _delay_us(500);
    if((PINC&0b00001000)==0)return 7;//Если на линии Столбец 1 лог. 0,
то нажата кнопка SB7
    if((PINC&0b00010000)==0)return 8;//Если на линии Столбец 2 лог. 0,
то нажата кнопка SB8
    if((PINC&0b00100000)==0)return 9;//Если на линии Столбец 3 лог. 0,
то нажата кнопка SB9
    return 0;//Если ни одна из кнопок не нажата, то не сработает ни
одно условие и программа дойдет сюда - возвращаем 0
    }
```

Разработаем функцию вывода цифры на индикатор

```
    void out_digit(char n)
    {
    char digits[10]={0b00111111, 0b00000110,0b01011011, 0b01001111,
0b01100110,0b01101101,0b01111101,0b00000111,0b01111111,
    0b01101111};//массив(таблица), где заданы коды всех цифр
    PORTD=digits[n];//выдаем код n-й цифры в порт, на индикатор
    }
```

Разработаем главную функцию программы (по алгоритму)

```
    void main()
    {
```

```
char knopka;//переменная для хранения номера нажатой кнопки
nastroyka_portov();//Настроить порты
while(1)                //Бесконечно выполнять в цикле следующие действия

{
knopka=key();//Провести опрос клавиатуры, записать результат в
переменную
out_digit(knopka);Вывести число из переменной на индикатор
}
}
```

6.5 Контрольные вопросы

1. Для чего нужно объединять кнопки в матрицу?
2. Что произойдет, если нажать на две или три кнопки одновременно?
3. Где применяются матричные клавиатуры на практике?

7 Лабораторная работа №7 - работа с таймером

7.1 Цели и задачи работы

Цель – научиться работать с таймером на примере микроконтроллера ATMega8

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

7.2 Задание 1

Воспроизвести мелодию, номера и длительности нот которой приведены в таблице 14

Таблица 14 – ноты мелодии

№	1	2	3	4	5	6	7	8	9	10
Длительность	3	1	4	4	4	8	3	1	4	4
Номер ноты	37	37	39	37	42	41	37	37	39	37
№	11	12	13	14	15	16	17	18	19	20
Длительность	4	8	3	1	4	4	4	4	4	3
Номер ноты	44	42	37	37	49	46	42	41	39	47
№	21	22	23	24	25					
Длительность	1	4	4	4	8					
Номер ноты	47	46	42	44	42					

Длительность нот дана в 1/10 с.
Для пересчета номера ноты в частоту применять формулу (1):

$$f = f_0(\sqrt[12]{2})^n \quad (1)$$

Где f0 = частота первой ноты (32,7 Гц), n – номер ноты
Тактовая частота контроллера указана в таблице 15

Таблица 15 – список вариантов

Вариант	Пример	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
fт, МГц	8	7,8	7,6	7,4	7,2	7	6,8	6,6	6,4	6,2	6	5,8	5,6	5,4	5,2	5	4,8

7.3 Сборка схемы устройства

7.3.1 Схема в программе SimulIDE

Разместите на схеме микроконтроллер, элемент «Земля» и «Звуковой выход» (последний находится в секции «Выходы» панели компонентов) и соедините их, как показано на рисунке 48

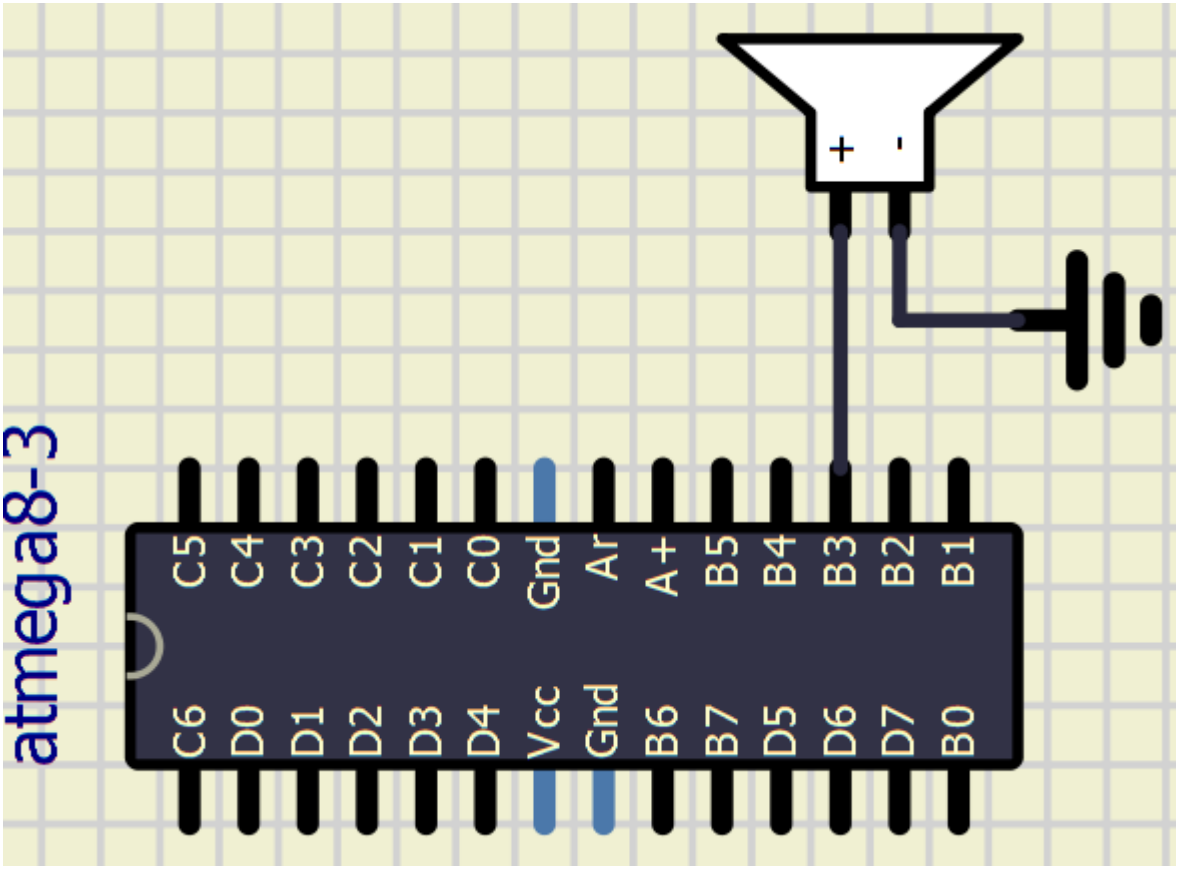

Рисунок 48 – принципиальная схема устройства

Затем щелкните правой кнопкой «мыши» по контроллеру, выберите пункт «Свойства» и в открывшемся окне введите тактовую частоту, как показано на рисунке 49

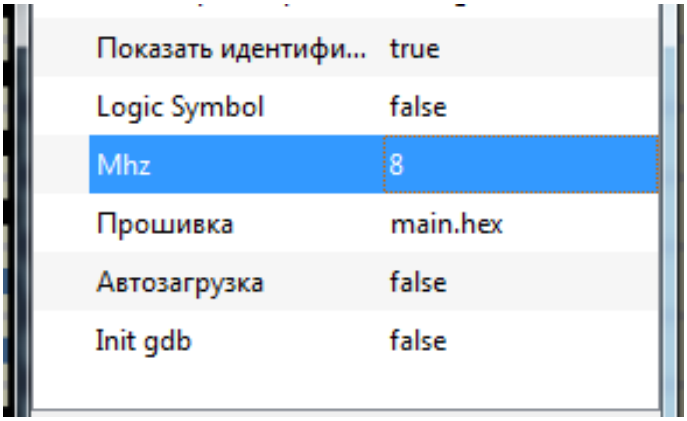

Рисунок 49 – установка частоты контроллера

7.3.2 В программе «Proteus»

Можно использовать схему от примера «Мигание светодиода». Удалим лишние компоненты и разместим компонент SOUNDER, выбор которого показан на рисунке 50. Подключим его к выводу OC2, так как используется таймер 2, как показано на рисунке 51

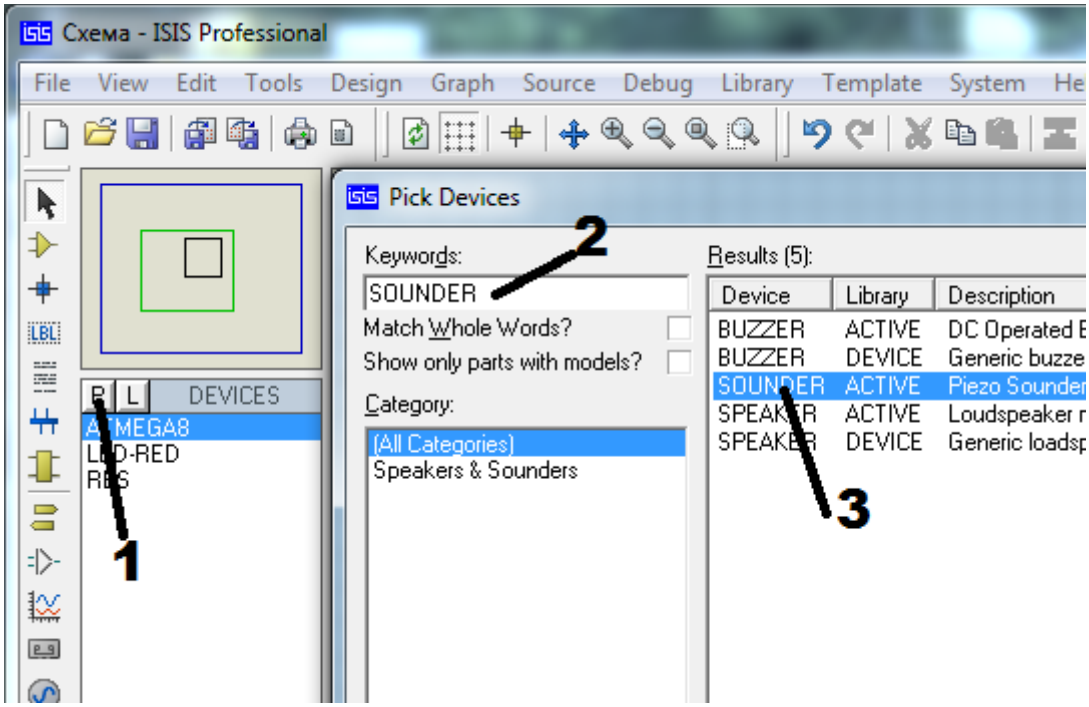

Рисунок 50 – выбор компонента «Sounder».

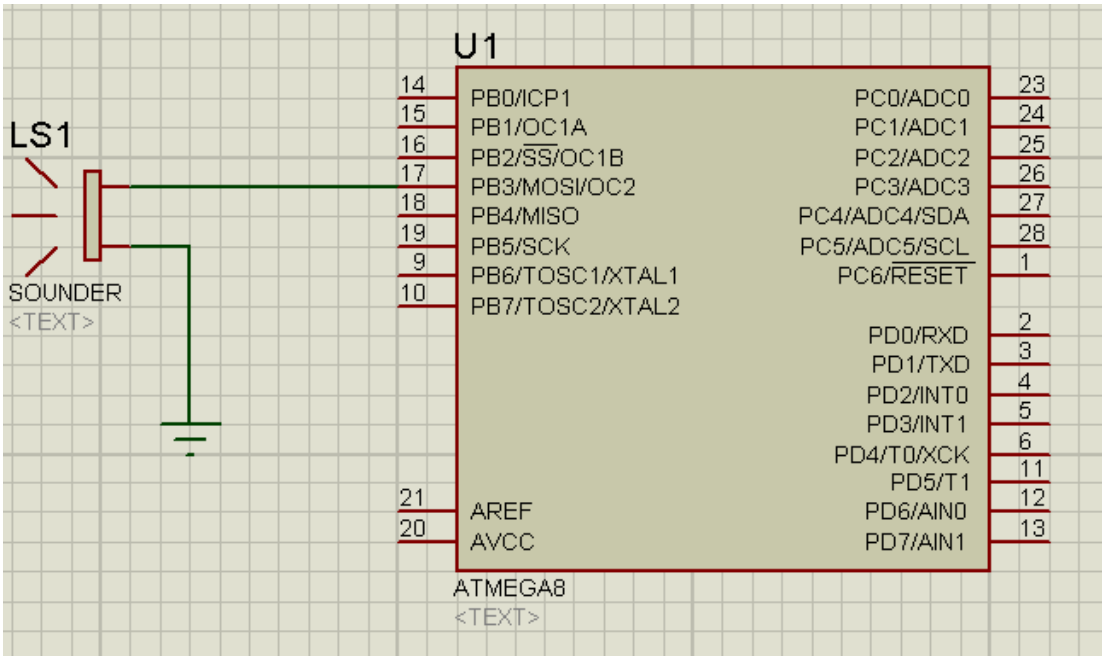

Рисунок 51 – принципиальная схема устройства

Установим частоту контроллера согласно варианту, как показано на рисунке 52

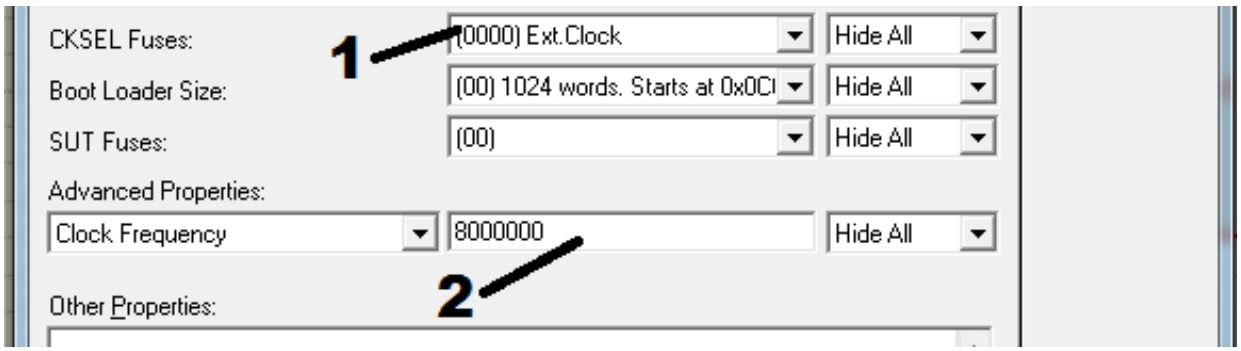

Рисунок 52 – установка тактовой частоты

7.4 Разработка программы

Предлагаемый алгоритм программы показан на рисунке 53

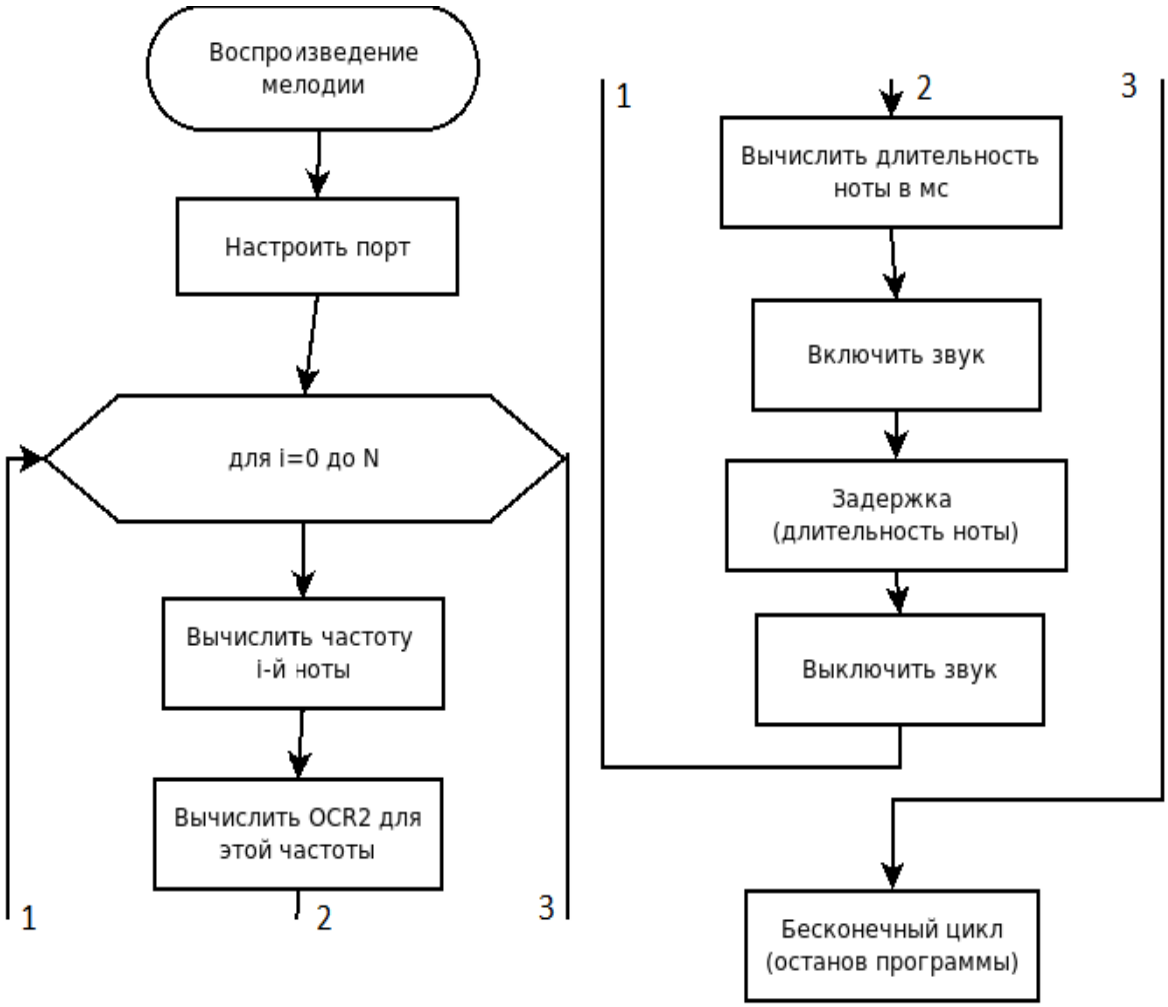

Рисунок 53 – алгоритм программы.

Запишем код для каждого блока:

Настройка порта: нам нужно настроить вывод контроллера PB3 (OC2) на вывод:

```
DDRB=0b00001000;
```

Настройка таймера: Мы будем использовать таймер в режиме деления частоты. Настройки таймера показаны на рисунке 54

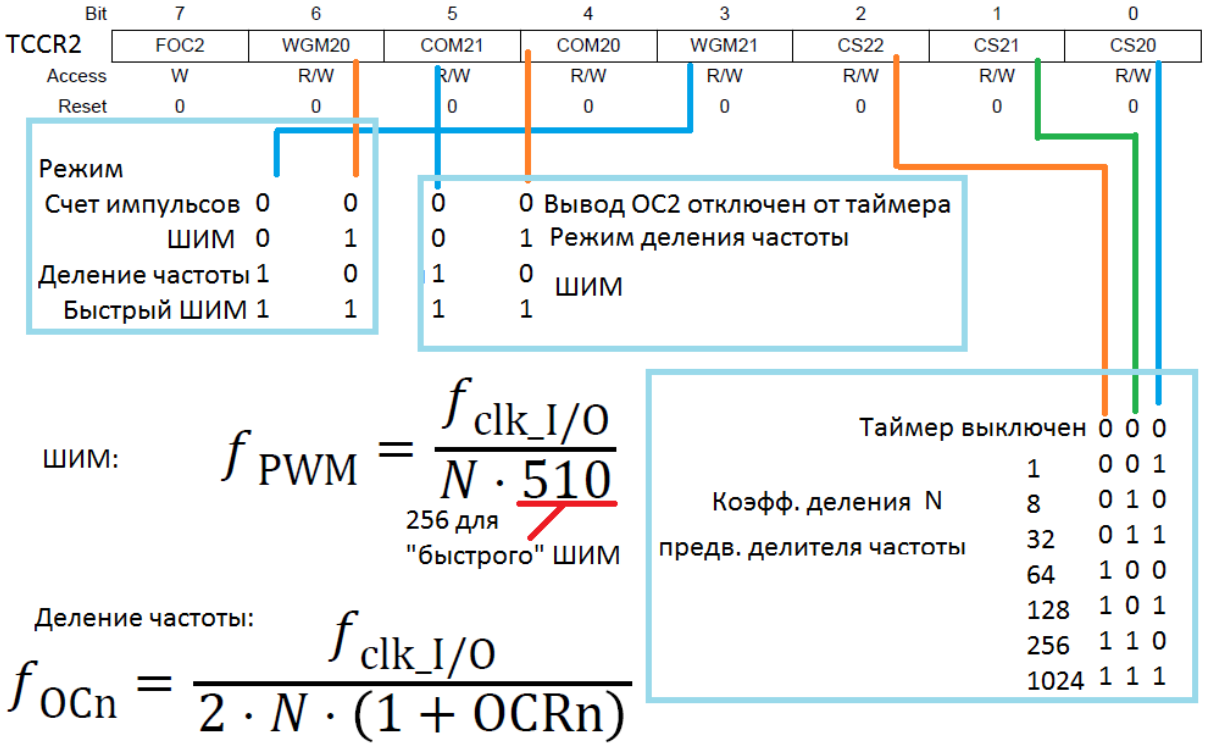

Рисунок 54 – биты конфигурации таймера

Тогда значения битов в регистре TCCR2 будут следующими:
FOC2=0
WGM20=0
COM21=0
COM20=1
WGM21=1

Теперь определимся с коэффициентом деления предварительного делителя N. Выберем его таким, чтобы требуемый для воспроизведения мелодии коэффициент OCR2 был максимальным, но не превышал 255.

Для этого рассчитаем по формуле (1) минимальную и максимальную частоты нот в мелодии (их номера – 37 и 49)

Частоты равны 277 и 554 Гц соответственно

Выразим OCR2 из формулы для деления частоты:

$$f = \frac{f\text{такт}}{2N(1+OCR2)} \qquad (2)$$

$$OCR2 = \frac{f\text{такт}}{2Nf} - 1 \quad (3)$$

Где Fтакт – тактовая частота контроллера

f – частота выходного сигнала

N – коэффициент деления предварительного делителя (см таблицу по настройке таймера)

OCR2 – число, которое нужно записать в регистр OCR2 для получения частоты f на выходе

Нам доступны значения N от 1 до 1024. Рассчитаем для каждого из них, какое число нужно записывать в регистр OCR2 для воспроизведения максимальной и минимальной частот мелодии, как показано в таблице 16.

Таблица 16 – расчет коэффициента деления таймера

fтакт=	8000000	Гц	
	min	max	
f=	277	554	
			Примечание
N	Значение OCR2		
1	14439,43	7219,217	>255, не подходит
8	1804,054	901,5271	>255, не подходит
32	450,2635	224,6318	>255, не подходит
64	224,6318	111,8159	Наилучший вариант
128	111,8159	55,40794	
256	55,40794	27,20397	
1024	13,10199	6,050993	

Выбираем N=64

Тогда биты CS22-CS20 будут равны 100 для включения звука и 000 для выключения звука (определим это по рисунку 54)

И настройка таймера будет выглядеть так:

Для включения звука

```
TCCR2=0b00011100
```

Для выключения звука

```
TCCR2=0b0001000
```

Внесем коды мелодии в массив:

```
char dlitelnost[25]={3,1,4,4,4,8,3,1,4,4,4,8,3,1,4,4,4,4,4,3,1,4,4,4,8};
```

```
char nomer[25]={37,37,39,37,42,41,37,37,39,37,44,42,37,37,49,
46,42,41,39,47,47,46,42,44,42};
```

И напишем код программы

```
#include<avr/io.h> //библиотека ввода-вывода
#include<avr/delay.h> //библиотека с функцией задержки _delay_ms
#include<math.h>
char dlitelnost[25]={3,1,4,4,4,8,3,1,4,4,4,8,3,1,4,4,4,4,4,3,1,4,4,4,8};
char nomer[25]={37,37,39,37,42,41,37,37,39,37,44,42,37,37,
49,46,42,41,39,47,47,46,42,44,42};
void main () //заголовок главной функции программы
{
DDRB=0b00001000;//настройка PB3 на вывод
char i;//переменная цикла
unsigned int frequency;//переменная для частоты ноты
float f0=32.7;//частота первой ноты, Гц
int N=64;//коэффициент деления предварительного делителя (см.
выше)
int t;//переменная для длительности ноты
for(i=0;i<=24;i++)//цикл для воспроизведения нот
{
frequency=f0*pow(1.05946309,nomer[i]);//вычисляем частоту ноты
OCR2=((8000000/(2*N))/frequency)-1;//вычисляем коэффициент
деления для этой частоты
t=100*dlitelnost[i];//вычисляем длительность ноты в мс
TCCR2=0b00011100;//включение звука
_delay_ms(t);//задержка
TCCR2=0b00011000;//выключение звука
}
for(;;);//бесконечный цикл – останов программы
```

7.5 Задание 2

Разработать программу для воспроизведения мелодии, описанной в таблице 17.

Номер ноты 0 означает паузу. Это значит, что если номер ноты равен нулю, то звук включать не нужно

Таблица 17 – мелодия для задания 2

№	1	2	3	4	5	6	7	8	9	10
Длительность	0	0	48	53	56	55	53	60	58	54
Номер ноты	4	4	4	6	2	4	8	4	12	12

Таблица 17 (продолжение)

№	11	12	13	14	15	16	17	18	19	20
Длительность	53	56	54	52	54	48	48	53	56	55
Номер ноты	6	2	4	8	4	20	4	6	2	4
№	21	22	23	24	25	26	27	28	29	30
Длительность	53	60	63	62	61	57	61	60	59	47
Номер ноты	8	4	8	4	8	4	6	2	4	8
№	31	32	33	34	35	36	37	38	39	40
Длительность	56	53	0	56	60	56	60	56	61	60
Номер ноты	4	12	8	4	8	4	8	4	8	4
№	41	42	43	44	45	46	47	48	49	50
Длительность	59	55	56	60	59	47	60	56	60	60
Номер ноты	8	6	2	4	8	4	20	4	4	4
№	51	52	53	54	55	56	57	58	59	60
Длительность	56	60	56	63	62	61	57	62	60	59
Номер ноты	4	8	4	8	4	8	4	2	2	4
№	61	62	63	64						
Длительность	47	56	53	0						
Номер ноты	8	4	24	4						

7.6 Контрольные вопросы

1. Для чего может использоваться таймер?
2. Как задать требуемую частоту на выходе таймера?
3. Как включить и выключить таймер?

8 Лабораторная работа №8 - Разработка электронных часов

8.1 Цели и задачи работы

Цель – научиться работать с прерываниями таймера на примере микроконтроллера ATMega8

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

8.2 Задание 1

Разработать электронные часы с шестиразрядным семисегментным индикатором на микроконтроллере atmega8. Индикатор подключить к контроллеру согласно таблице 18.

Таблица 18 – варианты работы

Вариант	A	B	C	D	E	F	G	DP	1	2	3	4	5	6
Пример	0	1	2	3	4	5	6	7	0	1	2	3	4	5
1	1	2	3	4	5	6	7	0	1	2	3	4	5	0
2	2	3	4	5	6	7	0	1	2	3	4	5	0	1
3	3	4	5	6	7	0	1	2	3	4	5	0	1	2
4	4	5	6	7	0	1	2	3	4	5	0	1	2	3
5	5	6	7	0	1	2	3	4	5	0	1	2	3	4
6	6	7	0	1	2	3	4	5	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6	1	2	3	4	5	0
8	0	1	2	3	4	5	6	7	2	3	4	5	0	1
9	1	2	3	4	5	6	7	0	3	4	5	0	1	2
10	0	1	2	3	4	5	6	7	3	4	5	0	1	2
11	1	2	3	4	5	6	7	0	4	5	0	1	2	3
12	2	3	4	5	6	7	0	1	5	0	1	2	3	4
13	3	4	5	6	7	0	1	2	0	1	2	3	4	5
14	4	5	6	7	0	1	2	3	1	2	3	4	5	0
15	5	6	7	0	1	2	3	4	2	3	4	5	0	1
16	6	7	0	1	2	3	4	5	3	4	5	0	1	2

Линии сегментов в таблице 18 относятся к порту D, а линии разрядов индикатора – к порту С.

Начальное время (при запуске часов) задать внутри программы. При работе часов использовать прерывания от таймера

Примерный список переменных программы:

- n – массив из 6 элементов, для отображаемых цифр
- n[0]- десятки часов
- n[1]- единицы часов
- n[2]- десятки минут

- n[3]- единицы минут
- n[4]- десятки секунд
- n[5]- единицы секунд
- s – счетчик секунд
- m- счетчик минут
- h- счетчик часов
- periods – счетчик периодов таймера (значения – от 0 до 675)
- i – номер текущего отображаемого разряда

Так как будет использовано прерывание от таймера, то в программе будет два раздельных алгоритма: основная функция программы и прерывание от таймера

Основная функция (main):

1.Задать начальное значение всех переменных

2. Настроить порты

3. Настроить таймер 2 (режим деления частоты, предварительный делитель на 64, включить прерывания от таймера)

4. Включить прерывания в контроллере

5. Зациклиться (бесконечный цикл)

Функция прерывания:

1. Выполнить цикл счета времени
2. Разделить время на десятичные разряды
3. Выполнить цикл динамической индикации

8.3 Разработка схемы

Начертите схему, как показано на рисунке 55, если вы используете программу SimulIDE, или как на рисунке 56 для программы Proteus.

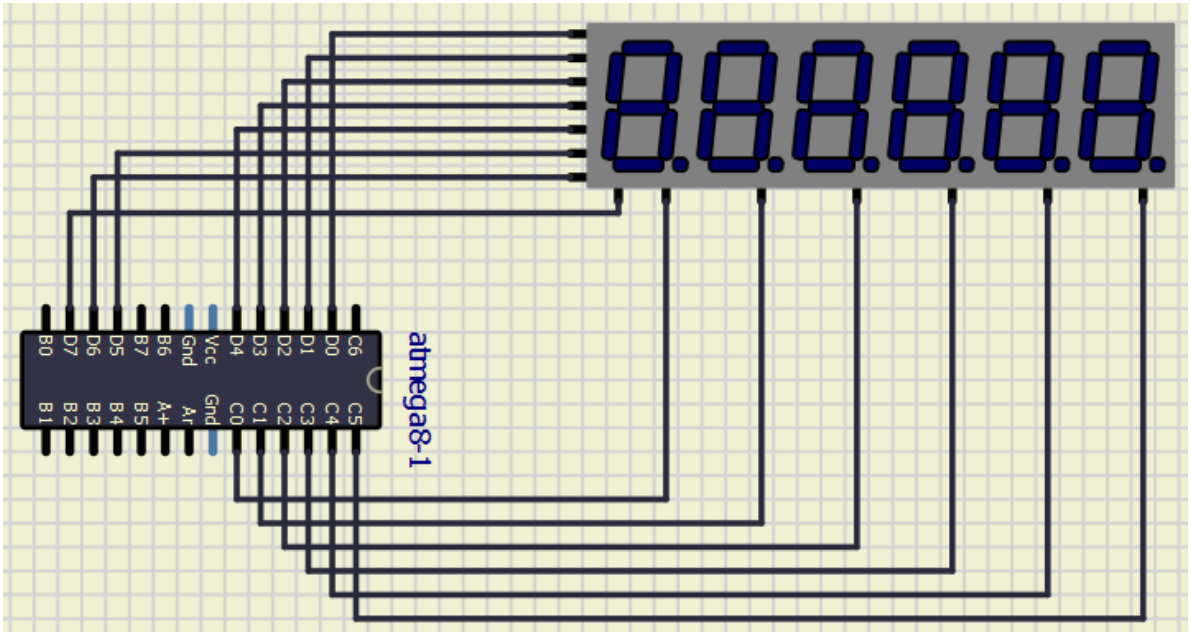

Рисунок 55 – принципиальная схема часов в программе SimulIDE

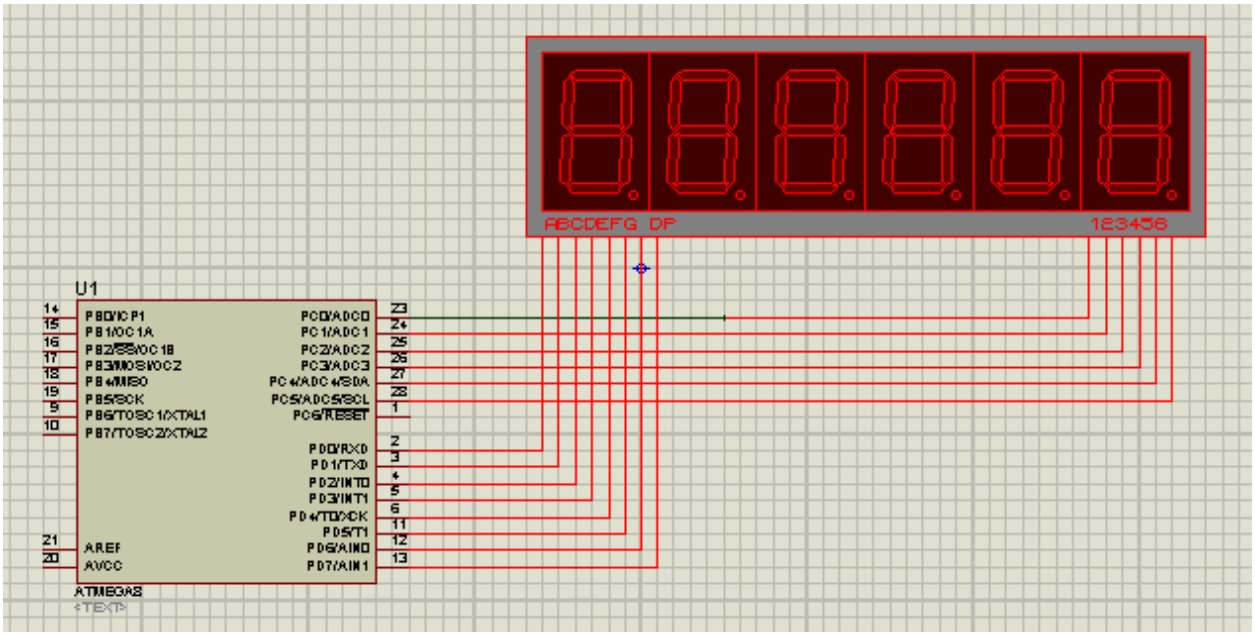

Рисунок 56 – принципиальная схема часов в программе Proteus

Схема в вашем проекте может отличаться из-за другого варианта (см. таблицу)

Дважды щелкните по контроллеру (или щелкните правой кнопкой «мыши» и выберите пункт «Свойства») для программы SimulIDE и установите тактовую частоту 11059200 Гц. Эта частота легко делится нацело на многие степени двойки (и на нее выпускаются кварцевые резонаторы), и из нее будет более удобно получать частоту 1 Гц, необходимую для работы часов[5]

8.4 Разработка программы

Разработаем основную функцию:
Настроим порты:

```
DDRD=0b11111111;//Все линии порта D на вывод
DDRC=0b00111111;//6 младших линий порта C на вывод
```

Настроим таймер согласно таблице, как показано на рисунке 54.

```
TCCR2=0b00000100;// режим деления частоты, предварительный делитель на 64,OC2 отключен от таймера
```

Для включения прерывания от таймера включим бит TOIE2 в регистре TIMSK:

```
TIMSK=0b01000000;//вкл прерывание по переполнению таймера 2
```

Рассчитаем частоту, с которой будет срабатывать прерывание от таймера

F=fтакт/(256*N)=11059200/(256*64)=675 Гц. Это значит, что прерывание по переполнению будет срабатывать с частотой 675 раз в секунду.

Основная функция будет выглядеть примерно так:

```
void main()
{
s=0;//Задаем начальные значения переменных
m=0;
h=0;
i=0;
periods=0;
DDRD=0b11111111;//Все линии порта D на вывод
DDRC=0b00111111;//6 младших линий порта C на вывод
TCCR2=0b00000100;// режим деления частоты, предварительный делитель на 64,OC2 отключен от таймера
TIMSK=0b01000000;//вкл прерывание по переполнению таймера 2
sei();//Включаем прерывания контроллера
while(1);//Зацикливаемся
}
```

Разработаем функцию динамической индикации

Примерный алгоритм функции:

1. Увеличить номер текущего разряда i на 1
2. Если он равен 6, обнулить его
3. Выключить все разряды индикатора
4. Выдать в порт D код i-й цифры (i – номер текущего разряда)
5. Если i=0, то включаем 1-й разряд
6. Если i=1, то включаем 2-й разряд
7. Если i=2, то включаем 3-й разряд
8. Если i=3, то включаем 4-й разряд
9. Если i=4, то включаем 5-й разряд
10. Если i=5, то включаем 6-й разряд

```
void dyn_ind()
{
char digits[10]={0b00111111, 0b00000110,0b01011011, 0b01001111, 0b01100110,0b01101101,0b01111101,0b00000111,0b01111111, 0b01101111};//массив(таблица), где заданы коды всех цифр
i++;//Увеличить номер текущего разряда i на 1
if(i==6)i=0;//Если он равен 6, обнулить его
PORTC=0b00111111;//Выключить все разряды индикатора
PORTD=digits[n[i]];//Выдать в порт D код i-й цифры (i – номер текущего разряда)
//Управление точками
```

```
if((i==1)//Если текущий разряд - первый
||((i==3)&&(periods<337)))//или если третий разряд и первая половина секунды
PORTD+=0b10000000;//то включаем точку на индикаторе
if(i==0)PORTC=0b00111110;//Если i=0, то включаем 1-й разряд
if(i==1)PORTC=0b00111101;//Если i=1, то включаем 2-й разряд
if(i==2)PORTC=0b00111011;//Если i=2, то включаем 3-й разряд
if(i==3)PORTC=0b00110111;//Если i=3, то включаем 4-й разряд
if(i==4)PORTC=0b00101111;//Если i=4, то включаем 5-й разряд
if(i==5)PORTC=0b00011111;//Если i=5, то включаем 6-й разряд
}
```

Разработаем функцию подсчета времени

Примерный алгоритм функции

1. Увеличить значение счетчика периодов на 1
2. Если прошло 675 периодов (1 секунда), то увеличить счетчик секунд на 1 и обнулить счетчик периодов
3. Если прошло 60 секунд, увеличить счетчик минут на 1 и обнулить счетчик секунд
4. Если прошло 60 минут, увеличить счетчик часов на 1 и обнулить счетчик минут
5. Если прошло 24 часа, то обнулить счетчик часов

```
void count_time()
{
periods++;//Увеличить значение счетчика периодов на 1
if(periods==675)//Если прошло 675 периодов (1 секунда), то
{
        periods=0;//обнулить счетчик периодов
        s++;//увеличить счетчик секунд на 1
}
if(s==60)//Если прошло 60 секунд
{
        m++;//увеличить счетчик минут на 1
        s=0;//и обнулить счетчик секунд
}

if(m==60)//Если прошло 60 минут
{
        h++;//увеличить счетчик часов на 1
        m=0;//и обнулить счетчик минут
}
```

```
if(h==24)h=0;//Если прошло 24 часа, то обнулить счетчик часов
}
```

Разработаем функцию перевода времени в десятичный вид

Примерный алгоритм функции:

1. Выделить десятки часов
2. Выделить единицы часов
3. Выделить десятки минут
4. Выделить единицы минут
5. Выделить десятки секунд
6. Выделить единицы секунд

Код этой функции может выглядеть примерно так:

```
void convert_to_dec()
{
	n[0]=(h/10)%10;//выделить десятки часов
	n[1]=h%10;//Выделить единицы часов
	n[2]=(m/10)%10;//Выделить десятки минут
	n[3]=m%10;//Выделить единицы минут
	n[4]=(s/10)%10;//Выделить десятки секунд
	n[5]=s%10;//Выделить единицы секунд

}
```

Разработаем функцию для обработки прерывания:

```
ISR(TIMER2_OVF_vect) //Прерывание по переполнению таймера 2
{
count_time();//Выполнить цикл счета времени
convert_to_dec();//Разделить время на десятичные разряды
dyn_ind();//Выполнить цикл динамической индикации
}
```

После этого соберите проект и проверьте работу часов.

8.5 Задание 2

Добавьте функцию установки времени в часы

Для этого подключите две кнопки к линиям PB0 и PB1 согласно схеме, показанной на рисунке 57.

Добавьте следующий код в функцию счета времени (в тело оператора if, в котором происходит увеличение переменной s на единицу)

```
if((PINB&0b00000001)==0)
```

```
{
h++;
if(h==24)h=0;//Если прошло 24 часа, то обнулить счетчик часов

}
if((PINB&0b00000010)==0)
{
m++;
if(m==60)m=0;//Если прошло 60 минут, то обнулить счетчик минут

}
```

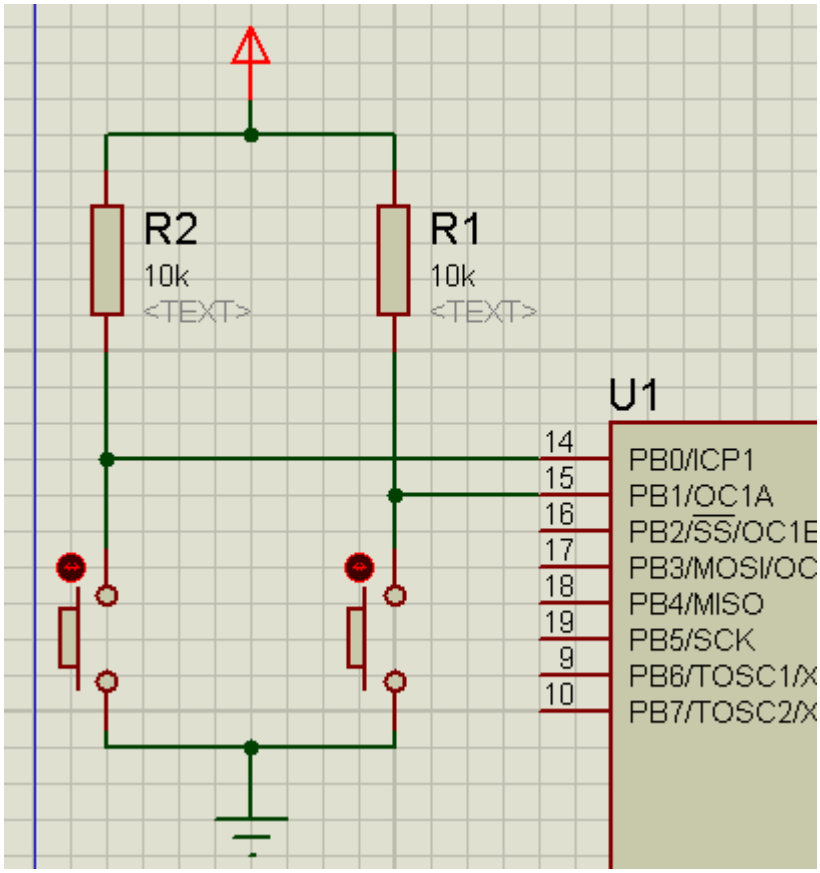

Рисунок 57 – подключение кнопок установки времени

8.6 Контрольные вопросы

1. Почему деление частоты осуществлялось до 675 Гц, а не сразу до 1 Гц?
2. Как бы вы добавили к проекту функцию будильника?
3. Каким образом происходит включение точек на индикаторе?

9 Лабораторная работа №9 - работа с индикатором 1602

9.1 Цели и задачи работы

Цель – научиться отображать информацию на жидкокристаллическом индикаторе 1602

Задачи работы: разработка алгоритма, перевод алгоритма на язык С, моделирование устройства.

9.2 Задание 1

Вывести на экран индикатора ваше имя (буквами английского алфавита)
Линии D0-D7 индикатора подключите к порту D контроллера.
Линии RS и E подключите согласно таблице 19

Таблица 19 – список вариантов работы

Вариант	Куда подключать RS	Куда подключать E
Пример	PC0	PC1
1	PC1	PC2
2	PC3	PC2
3	PC4	PC2
4	PC5	PC2
5	PC1	PC3
6	PC2	PC3
7	PC4	PC3
8	PC5	PC3
9	PC1	PC4
10	PB1	PB2
11	PB3	PB2
12	PB4	PB2
13	PB5	PB2
14	PB1	PB3
15	PB2	PB3
16	PB4	PB3

С подробной документацией на индикатор можно ознакомиться в инструкции[6].

9.3 Разработка схемы

9.3.1 Разработка схемы в программе SimulIDE

1. Соберем схему, как показано на рисунке 58.

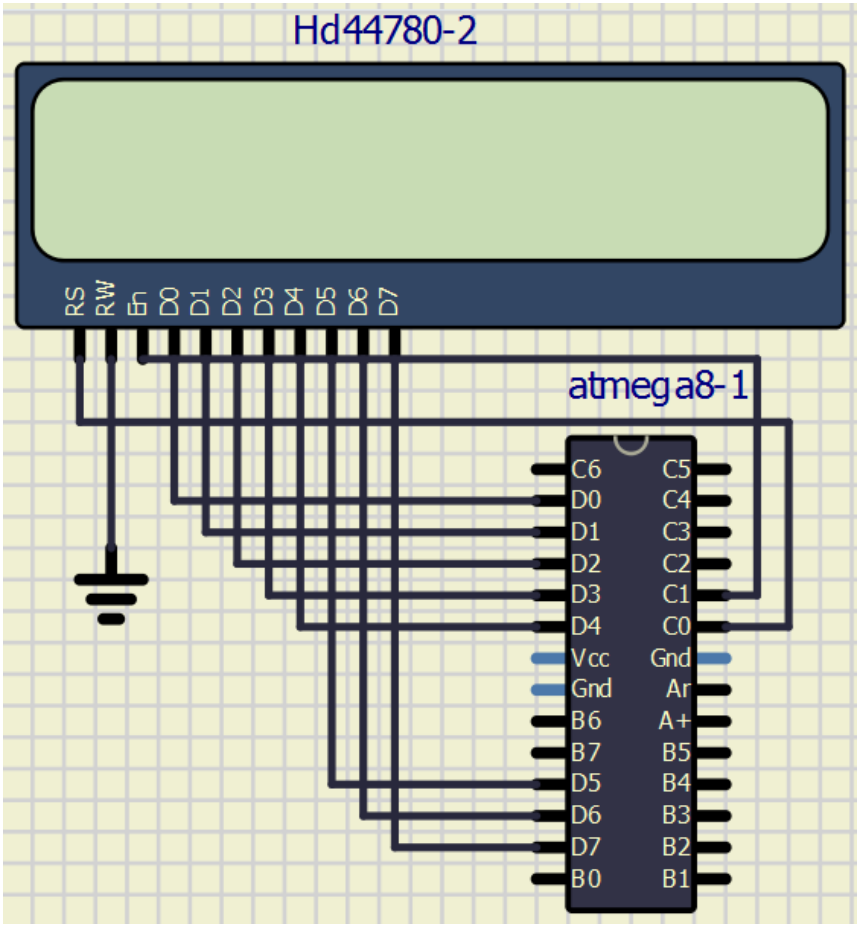

Рисунок 58 – подключение индикатора к микроконтроллеру

Используйте компонент HD44780 в качестве индикатора.

9.3.2 Разработка схемы в программе Proteus

Начертите схему, как показано на рисунке 59

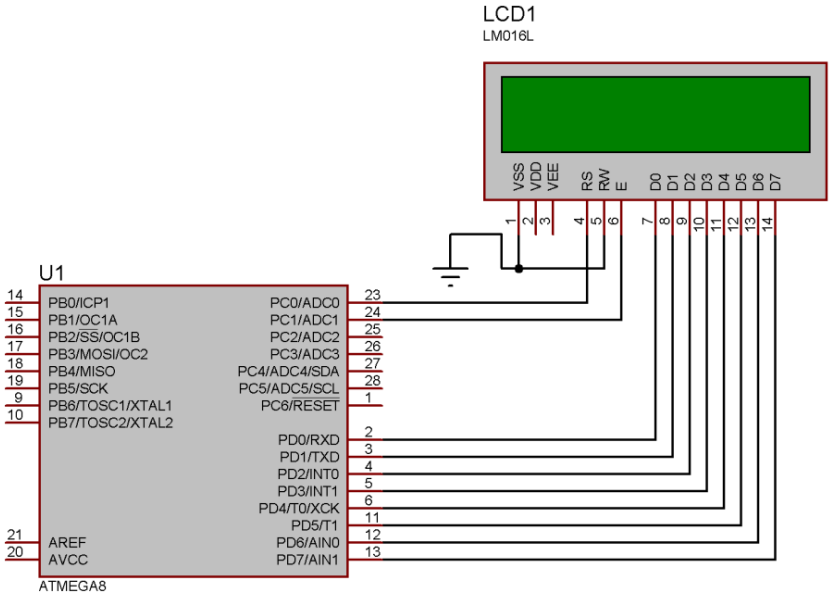

Рисунок 59 – подключение индикатора к контроллеру

Индикатор искать в библиотеке по слову LM016L.

9.4 Разработка программы

В программе должны быть:
-Функция отправки команды на индикатор
-Функция отправки данных на индикатор
-Функция начальной настройки (инициализации)
-Главная функция

Функция отправки команды на индикатор

Для того, чтобы отправить команду на индикатор, нужно выполнить следующие действия:
-Установить линию RS=0
-Выдать код команды на линии данных индикатора
-Установить линию E=1
-Через время >500 нс установить линию E=0
Функция может выглядеть следующим образом:

```
void send_command(char code)
{
PORTC=PORTC&0b11111110;//RS=0;
PORTD=code;//вывод кода на индикатор
PORTC=PORTC|0b00000010;//E=1
_delay_us(2);//задержка на треб. длительность импульса E
PORTC=PORTC&0b11111101;//E=0
_delay_us(20);
}
```

Числа, выделенные жирным шрифтом, будут меняться в зависимости от подключения индикатора в вашем варианте

Функция отправки данных на индикатор

Для того, чтобы отправить данные на индикатор, нужно выполнить следующие действия:
-Установить линию RS=1
-Выдать код команды на линии данных индикатора
-Установить линию E=1
-Через время >500 нс установить линию E=0
Функция может выглядеть следующим образом:

```
void send_data(char code)
{
PORTC=PORTC|0b00000001;//RS=1;
PORTD=code;//вывод кода на индикатор
PORTC=PORTC|0b00000010;//E=1
```

```
_delay_us(2);//задержка на треб. длительность импульса E
PORTC=PORTC&0b11111101;//E=0
_delay_us(200);
}
```

Числа, выделенные жирным шрифтом, будут меняться в зависимости от подключения индикатора в вашем варианте

Функция инициализации индикатора

Для инициализации индикатора требуется выполнить следующие действия:

-Подождать 20 мс

-выдать команду 0x30 для включения 8-битного режима

-выдать команду 0x01 для очистки экрана

-выдать команду 0b1101 для включения экрана

Тогда функция инициализации будет выглядеть следующим образом:

```
void init_lcd()
{
_delay_ms(20);
send_command(0x30);//вкл 8-битный режим
send_command(0x01);//очистка экрана
send_command(0b1101);//вкл экран и курсор
}
```

Главная функция программы

В главной функции следует выполнить следующие действия:

-Настроить на вывод линии порда D и используемые для работы с индикатором линии порта C

-Вызвать функцию инициализации индикатора

-Подождать 5 мс

-Отправить по очереди все символы слова

-Зациклиться (бесконечный цикл)

```
void main () //заголовок главной функции программы
{
DDRD=0b11111111;//Весь пори D на вывод
DDRC=0b00000011;//PC0 и PC1 на вывод
 init_lcd();//Инициализация ЖКИ
_delay_ms(5);
send_data('H');//Выводим слово
send_data('e');
send_data('l');
send_data('l');
send_data('o');
```

```
send_data('!');
for(;;);//Зацикливаемся
}
```

После запуска проекта на индикаторе должно отобразиться слово, как показано на рисунке 60

Рисунок 60 – слово «Hello» на индикаторе

9.5 Контрольные вопросы

1. В чем отличие применяемого в данной работе индикатора от семисегментных индикаторов, изученных нами ранее?
2. Предложите способ вывода текстовой информации из строки (массива символов типа char) на индикатор
3. Как можно уменьшить количество линий, соединяющих индикатор и контроллер?

10 Лабораторная работа №10 - работа с АЦП

10.1 Цели и задачи работы

Цель – научиться работать с модулем АЦП микроконтроллера

Задачи работы: настройка АЦП и проведение измерения, расчет делителя напряжения, отображение результатов на индикаторе.

10.2 Задание 1

Разработать вольтметр для измерения напряжения от 0 до Umax В. Использовать канал АЦП, указанный в таблице 20

Таблица 20 – список вариантов работы

Варианты	Пример,6,12	1,7,13	2,8,14	3,9,15	4,10,16	5,11
Канал АЦП	ADC0	ADC1	ADC2	ADC3	ADC4	ADC5
Варианты	Пример,1-5	6-11	12-16			
Umax,B	10,23	102,3	1023			

Используйте внутренний источник опорного напряжения 2,56 В

10.3 Разработка схемы

10.3.1 Разработка схемы в программе SimulIDE

Подключим элементы схемы, как показано на рисунке 61.

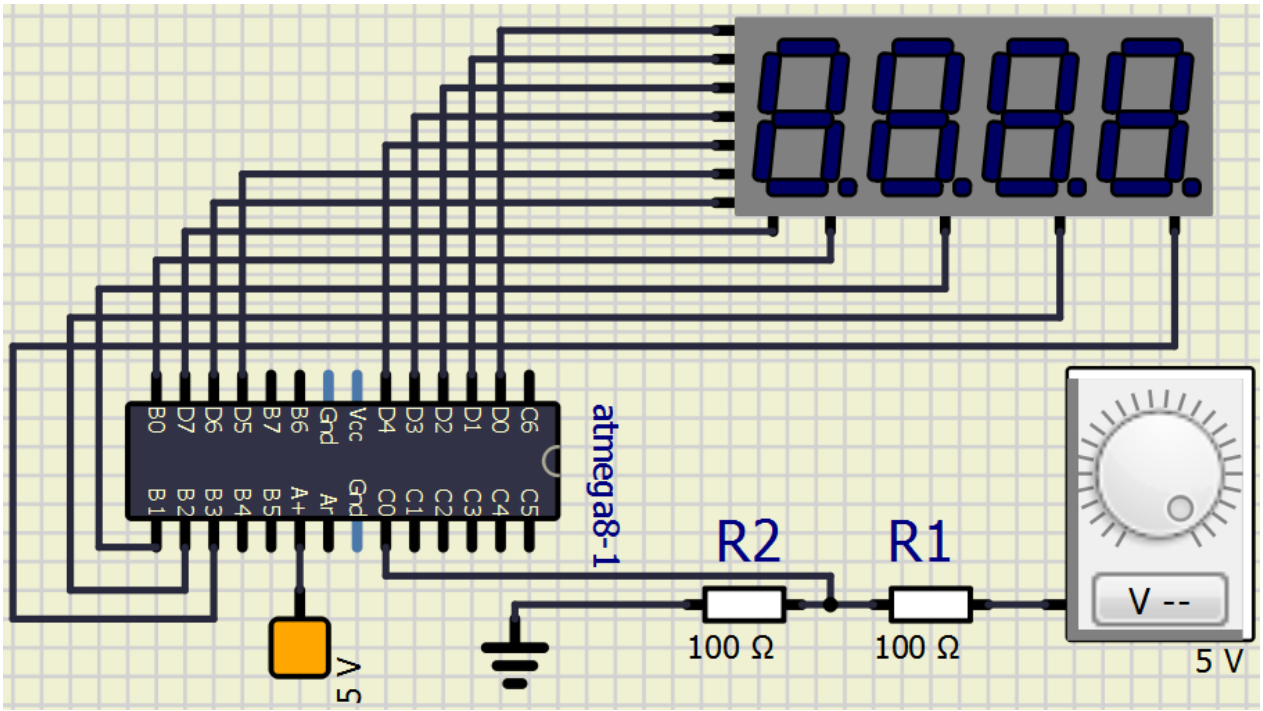

Рисунок 61 – принципиальная схема цифрового вольтметра

Используйте в этой схеме компоненты «Линия питания» и «Источник напряжения».

10.3.2 Разработка схемы в программе «Proteus»

Подключим элементы схемы, как показано на рисунке 62.

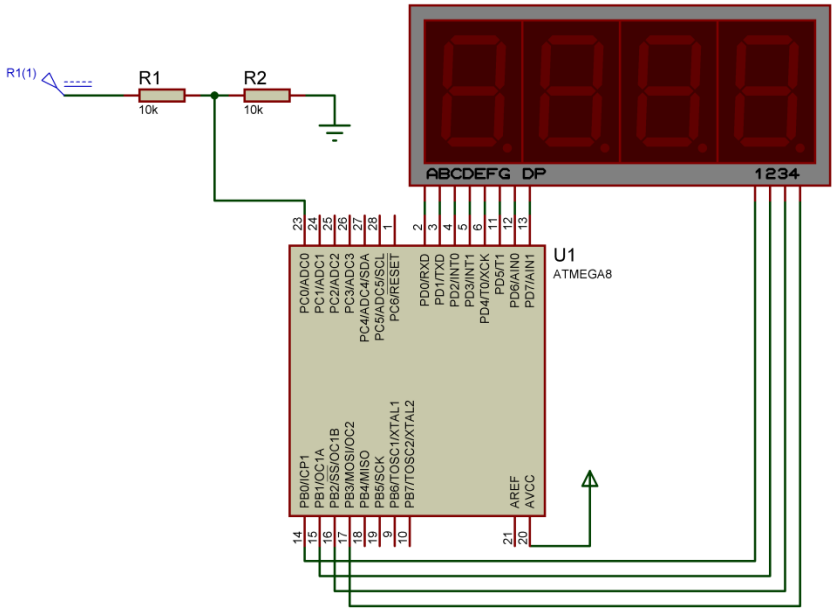

Рисунок 62 – принципиальная схема цифрового вольтметра

Линию AVCC контроллера подключите к терминалу «Power», как показано на рисунке 63. Подключим делитель напряжения к выводу PC0 контроллера и подсоединим к нему источник постоянного напряжения (установим напряжение на источнике), как показано на рисунке 64.

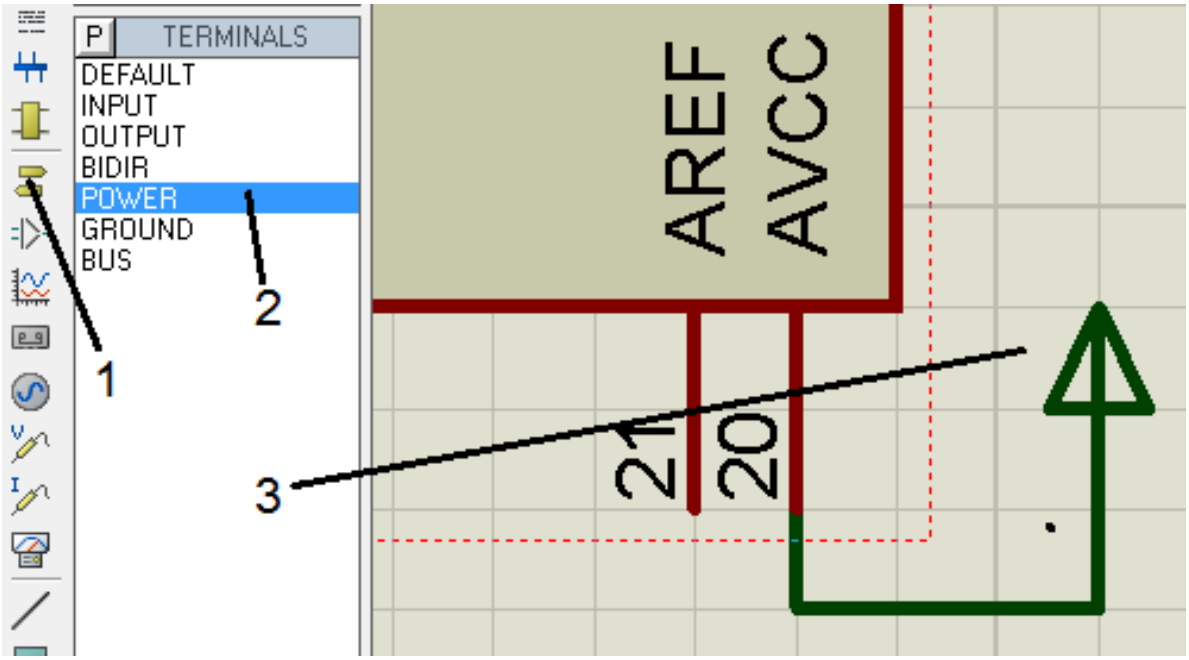

Рисунок 63 – подключение терминала Power

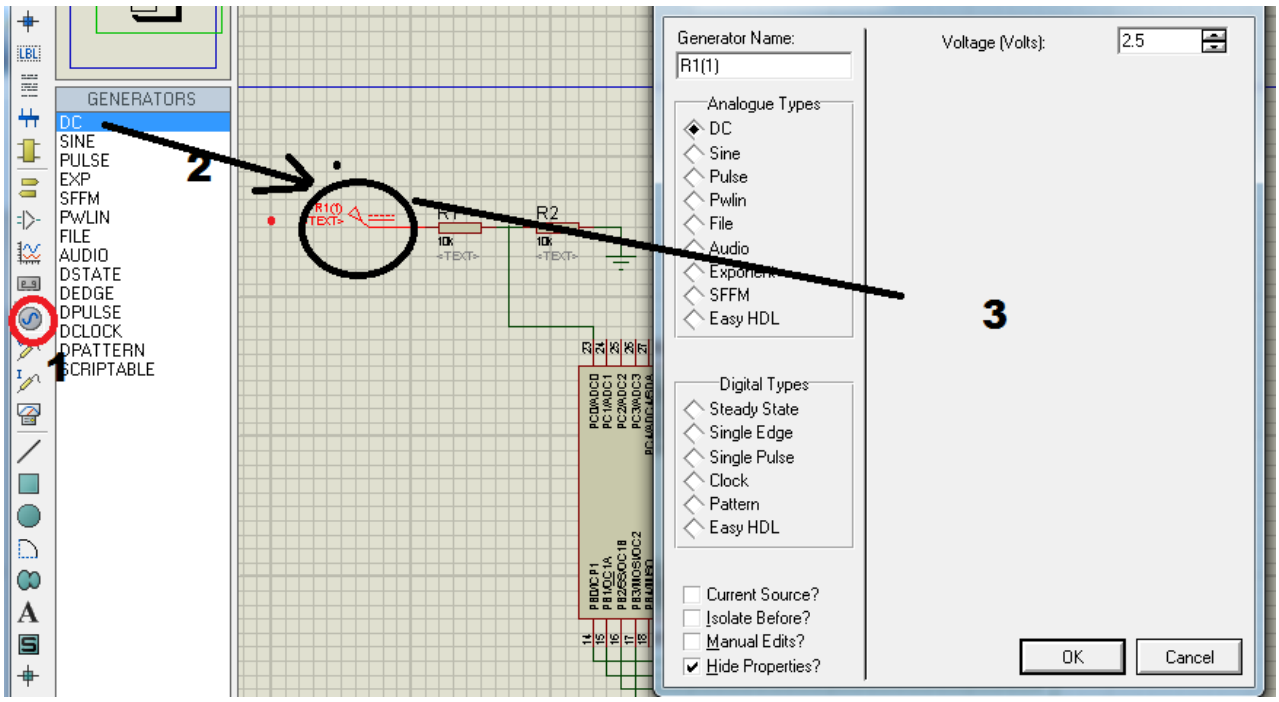

Рисунок 64 – подключение источника напряжения

Не забудьте, что схема подключения должна соответствовать таблице из вашего варианта.

10.3.3 Расчет делителя напряжения

Рассчитаем делитель напряжения. Примем R2=1 K, тогда должна выполняться пропорция(4):

$$\frac{U_{max}}{U_{ref}} = \frac{R1+R2}{R2} \qquad (4)$$

В нашем случае, Umax=10.23 В, Uref=2.56 В (одно из стандартных значений), тогда R1 получается равным 2996 Ом. Рассчитайте по этой формуле R1 и установите на схеме сопротивления резисторов, как показано на рисунке 65

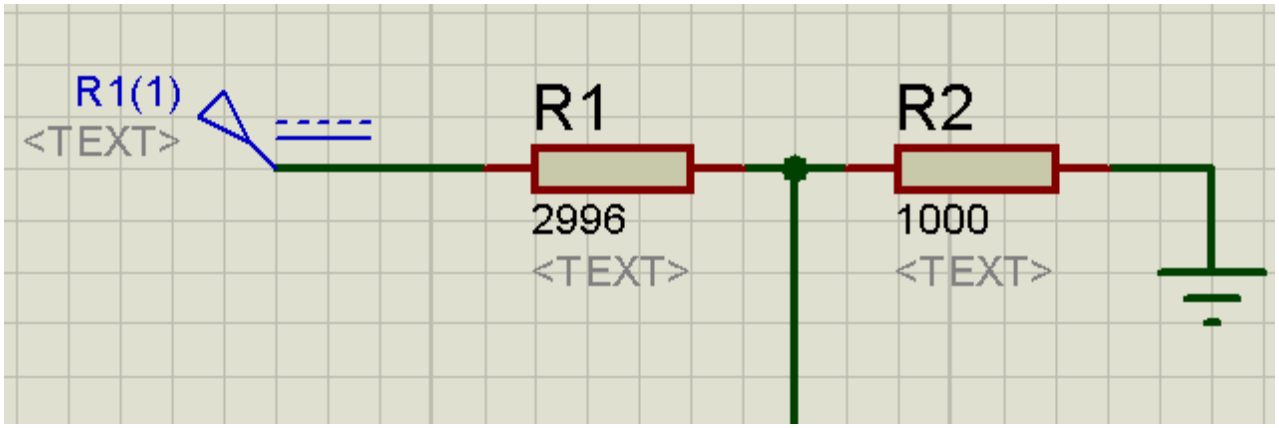

Рисунок 65 – установка сопротивлений резисторов

10.4 Разработка программы

Напишем программу. Она должна состоять из следующих пунктов:

1. Настройка портов
2. Настройка АЦП
3. Запуск измерения
4. Ожидание окончания измерения.
5. Считывание результатов измерения
6. Разделение результата (числа) на десятичные цифры (разряды)
7. Динамическая индикация результата

Настройка портов Все линии порта D (кроме PD7) используются на вывод, поэтому в регистр DDRD нужно записать 7 единиц в младших разрядах и 0 в старшем:

```
DDRD=0b01111111;
```

У порта B используются только 4 младшие линии:

```
DDRB=0b00001111;
```

Настройка АЦП.Для настройки АЦП обратимся к заводской документации на контроллер. Нас интересуют регистры ADMUX, биты которого показаны на рисунке 66 для настройки мультиплексоров АЦП и ADCSRA для управления режимами работы АЦП.

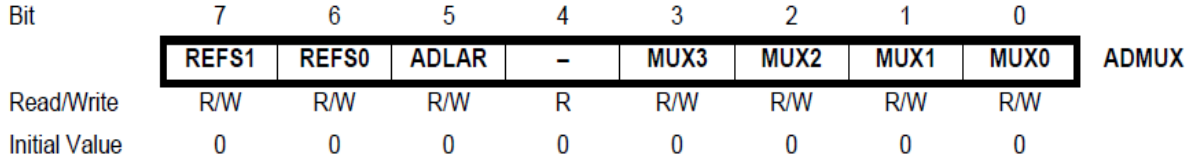

Рисунок 66 – биты регистра ADMUX

Выберем значения младших битов MUX3…0. Они выбирают входной канал АЦП, как показано в таблице 21.

Таблица 21 – выбор канала АЦП

Биты MUX3..0	Канал АЦП
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

По условию используется канал ADC0 (в вашем варианте может быть другой), поэтому значение этих битов будет 0000

Выберем значения битов REFS0..1 по таблице 22:

Таблица 22 – выбор источника опорного напряжения

Бит REFS1	Бит REFS0	Источник опорного напряжения
0	0	Вывод AREF, внешний источник
0	1	Линия AVCC с внешним конденсатором на линии AREF
1	0	Зарезервировано
1	1	Внутренний источник 2,56В с внешним конденсатором на линии AREF

Они выбирают источник опорного напряжения. Нам требуется последний вариант, поэтому значения битов будут 11. Значение остальных битов в регистре – нулевые.

Поэтому строка для настройки этого регистра будет выглядеть так:

```
ADMUX=0b11000000;
```

Настроим регистр ADCSRA. Биты этого регистра показаны на рисунке 67.

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 67 – регистр ADCSRA

Бит 7 ADEN включает модуль АЦП, поэтому он должен быть равен 1. Бит ADSC запускает измерение. Нам пока не нужно запускать измерение, поэтому он будет равен 0, но впоследствии для запуска измерения нужно сделать его равным 1.

Бит 5 ADFR включает режим непрерывного преобразования. Так как мы планируем работать в режиме одиночного преобразования, то он должен быть равен нулю.

Биты 4-3 ADIF и ADIE управляют прерываниями от АЦП. Так как мы не используем их, то сделаем эти биты равными нулю.

Биты ADPS2-ADPS0 задают, во сколько раз будет делиться тактовая частота контроллера перед подачей на АЦП.

Выберем максимальный коэффициент – он обеспечит самое точное (но медленное) измерение. То есть, эти биты будут равны 111

Учитывая всё это, строка для настройки данного регистра будет выглядеть следующим образом:

```
ADCSRA=0b10000111;
```

Для запуска измерения будем использовать ту де строку, но шестой бит ADSC сделаем равным 1:

```
ADCSRA=0b11000111;//Запуск преобразования АЦП
```

Пока измерение идёт, установленный нами бит ADSC будет оставаться равным 1, но после окончания измерения он станет равным 0. Для того, чтобы приостановить программу на это время, можно использовать цикл, который будет исполняться, пока этот бит равен 1. Для выделения бита используем логическую операцию побитового И с маской, в которой требуемый бит равен 1, а остальные – нулю. Так как количество повторений цикла неизвестно, используем цикл while

```
while((ADCSRA&64)!=0);
```

Результаты преобразования находятся в регистрах ADCL (младшая часть) и ADCH (старшая часть). Их нужно объединить в одно число

```
n=256*ADCH+ADCL;//Считываем результаты измерения
```

Значение, хранящееся в старшем регистре умножается на 256, это эквивалентно сдвигу числа на 8 разрядов влево.

«Сборка» числа из двух частей проиллюстрирована на рисунке 68

Рисунок 68 – объединение старшего и младшего байтов

(Значения чисел взяты случайным образом, для демонстрации принципа)

Для хранения отображаемых цифр нам понадобится массив из четырех элементов:

```
char N[4];//переменные для хранения цифр числа
```

Разделим число на десятичные разряды:

```
N[0]=n%10;//выделяем разряд единиц
N[1]=(n/10)%10;//выделяем разряд десятков
N[2]=(n/100)%10;//выделяем разряд сотен
N[3]=(n/1000)%10;//выделяем разряд тысяч
```

Для выполнения индикации опишем массив, хранящий коды цифр от 0 до 9:

```
char digits[10]={ 0b00111111,0b00000110, 0b01011011,0b01001111, 0b01100110, 0b01101101,0b01111101, 0b00000111, 0b01111111, 0b01101111 };
```

Код для выполнения индикации аналогичен тому, что применялся в предыдущих работах и большого интереса не представляет:

```
metka:
PORTB=0b1111;// Выключить все разряды индикатора
PORTD=digits[N[3]]; //Выдаем код цифры
PORTB=0b1110;// Включить первый разряд
_delay_ms(5);// Подождать 5 мс
PORTB=0b1111;// Выключить все разряды индикатора
PORTD=digits[N[2]]+0b10000000; //В этом разряде включаем точку
PORTB=0b1101;// Включить второй разряд
_delay_ms(5);// Подождать 5 мс
PORTB=0b1111;//Выключить все разряды индикатора
PORTD=digits[N[1]];
 PORTB=0b1011;//Включить третий разряд
_delay_ms(5);// Подождать 5 мс
PORTB=0b1111;// Выключить все разряды индикатора
PORTD=digits[N[0]];
PORTB=0b0111;// Включить четвертый разряд
_delay_ms(5);// Подождать 5 мс
goto metka; //Перейти к пункту 1
```

Следует отметить, что в одной строке к коду цифры прибавляется 0b10000000. Это включает точку на индикаторе в нужном разряде. Возможно, вам придется включить ее в другом разряде, чем указано в примере, чтобы напряжение отображалось правильно.

Для проверки вольтметра задавайте разные напряжения источника, дважды щелкнув по нему, и задавайте требуемое напряжение в окне (для программы proteus) или установите максимальное напряжение из вашего

варианта в свойствах и вращением движка установите его произвольно (для программы SimulIDE).

10.5 Контрольные вопросы

1. Для чего нужно опорное напряжение в АЦП?
2. Как устроен и как работает АЦП?
3. Предложите, как можно задействовать жидкокристаллический индикатор из предыдущей работы в данной схеме.
4. Попробуйте перенести метку в цикле индикации так, чтобы показания обновлялись непрерывно.

11 Лабораторная работа №11 - работа с USART

11.1 Цели и задачи работы

Цель – научиться работать с модулем USART микроконтроллера
Задачи работы: настройка USART, передача данных, прием данных.

11.2 Задание 1 – передача данных

Вывести с помощью USART строку символов (ваше имя), на скорости передачи данных, которая указана в таблице 23.

Таблица 23 – список вариантов работы

Варианты	Пример,7,14	1,8,15	2,9,16	3,10	4,11	5,12	6,13
Скорость, бит/с	1200	2400	4800	9600	19200	38400	57600
Варианты	Пример,1-6	7-13	14-16				
Тактовая частота, мГц	1	2	4				

11.3 Разработка схемы

11.3.1 Разработка схемы в программе SimulIDE

Расположите микроконтроллер на схеме, щелкните по нему правой кнопкой и выберите пункт «Открыть Serial Monitor». Откроется окно терминала, как показано на рисунке 69.

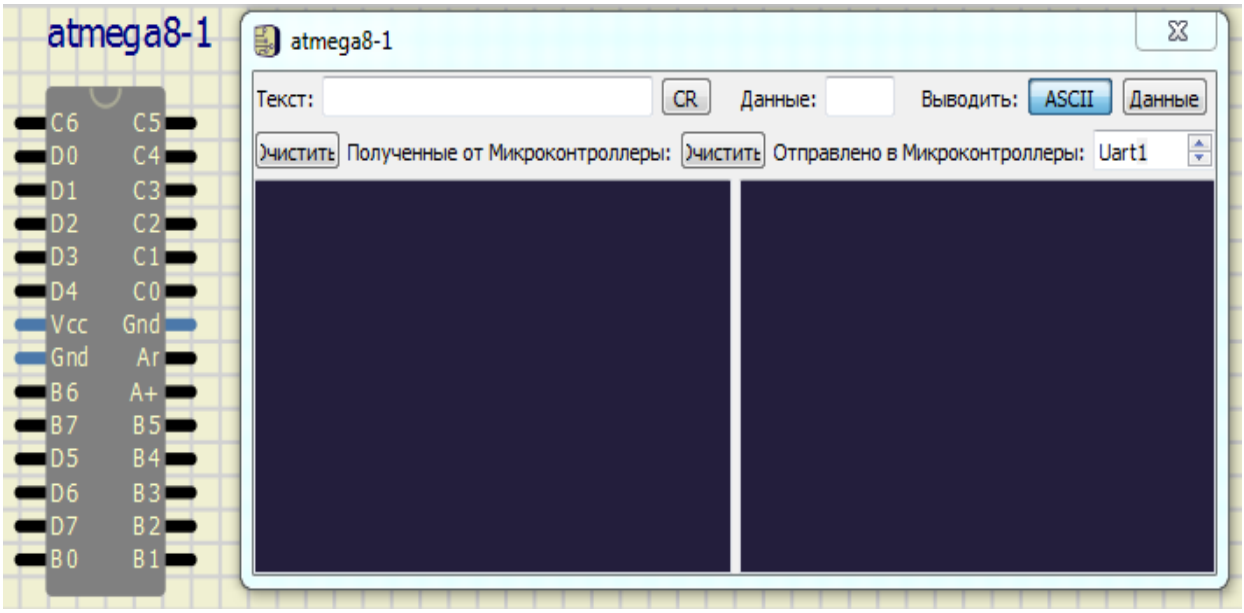

Рисунок 69 – микроконтроллер с терминалом

11.3.2 Разработка схемы в программе Proteus

Добавьте на схему виртуальный терминал – устройство, которое позволяет принимать и передавать текстовую информацию по протоколу RS-232. Для этого выберите режим измерительных приборов (1), выберите виртуальный терминал (2),разместите его на схеме (3) дважды щелкните по нему и задайте скорость передачи из вашего варианта (4), как показано на рисунке 70

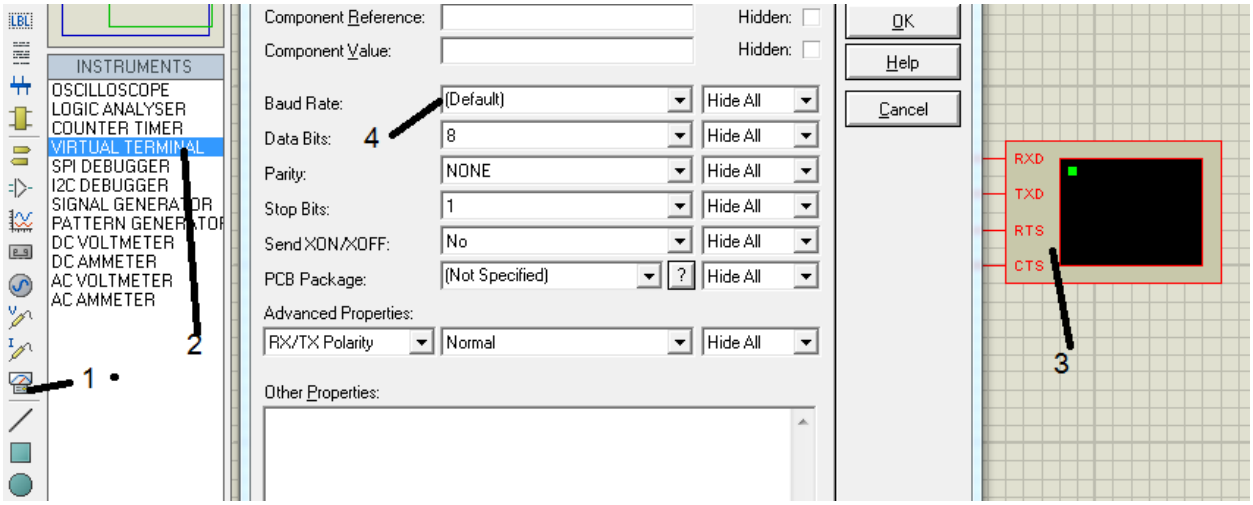

Рисунок 70 – размещение виртуального терминала на схеме

Разместите контроллер, если его нет, дважды щелкните по нему и задайте частоту. Соедините элементы согласно рисунку 71.

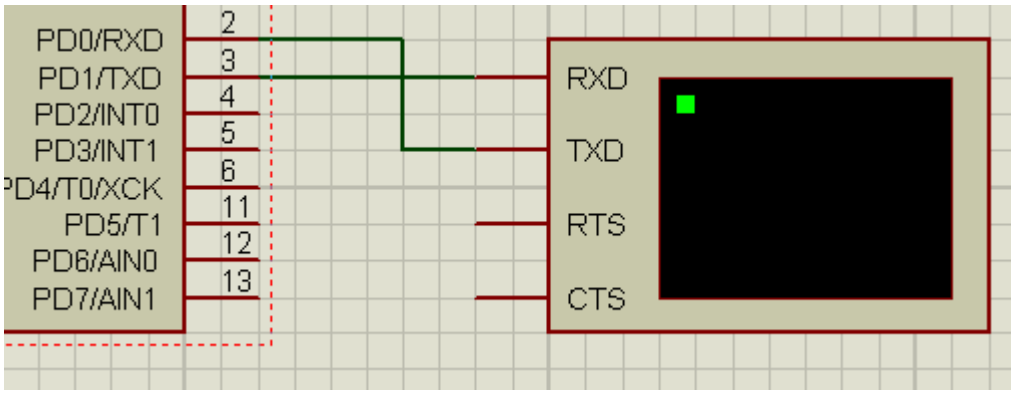

Рисунок 71 – соединение терминала с микроконтроллером

11.4 Разработка программы

Программа должна включать в себя следующие функции:
-Главная функция программы
-Функция настройки приемопередатчика
-Функция отправки символа

Начнем с настройки приемопередатчика. Зададим скорость передачи данных с помощью регистра UBRRL/H. Рассчитаем число, которое нужно поместить в этот регистр по формуле(5):

$$UBRR = \frac{F_{\text{такт}}}{16B} - 1 = \frac{1\ 000\ 000}{(16*1200)}\text{-1=51,08} \qquad (5)$$

Где B – скорость передачи данных, Fтакт – тактовая частота контроллера. Округлим его до ближайшего целого – 52. Строка для настройки скорости будет выглядеть следующим образом:

```
UBRR=52;
```

Биты регистра UCSRA представлены на рисунке 72:

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Рисунок 72 - регистр UCSRA

Бит 7 (RXC) – равен 1, когда есть принятые данные
Бит 6 (TXC) – равен 1, когда передача данных закончилась
Бит 5 (UDRE) – равен 1, когда приемопередатчик готов к передаче новых данных
Бит 4 (FE) – равен 1, когда произошла ошибка (принятый стартовый бит не равен 0)
Бит 3 (DOR) – равен 1, когда поступили новые данные, а предыдущие еще не были считаны контроллером
Бит 2 (PE) – равен 1 при возникновении ошибки четности
Бит 1 (U2X) – включает режим удвоенной скорости
Бит 0 – включает режим межпроцессорной коммуникации в синхронном режиме
Мы не будем ничего менять в этом регистре, поэтому в него можно записать нули:

```
UCSRA=0;
```

Биты регистра UCSRB представлены на рисунке 73:

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 73 – регистр UCSRB

Бит 7 (RXCIE)- Включает прерывание по окончанию приема данных

Бит 6 (TXCIE) – Включает прерывание по окончанию передачи данных

Бит 5 (UDRIE) – включает прерывание по готовности передавать новые данные

Бит 4 (RXEN) – включает приемник данных

Бит 3 (TXEN) – Включает передатчик данных

Бит 2 (UCSZ2) – Управление количеством бит данных (см. следующий регистр)

Бит 1 (RXB8) – 9-й принятый бит при 9-битном режиме

Бит 0 (TXB8) - 9-й передаваемый бит при 9-битном режиме

В этом регистре установим в единичное состояние биты RXEN и TXEN, которые включают приемник и передатчик данных:

```
UCSRB=(1<<RXEN)+(1<<TXEN);
```

Биты регистра UCSRC представлены на рисунке 74.

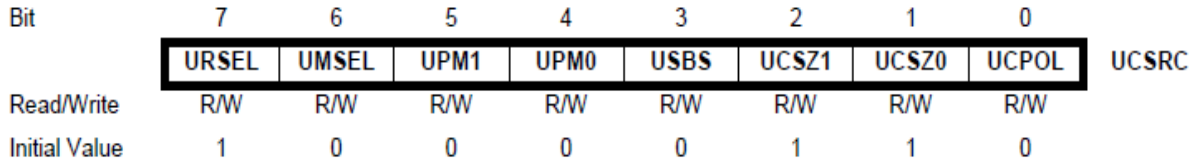

Рисунок 74 – регистр UCSRC

Бит 7 (URSEL) – выбор регистра, должен быть равен 1

Бит 6 (UMSEL) – включает синхронный режим

Бит 5 (UPM1) – вкл контроль четности

Бит 4 (UPM0) – режим контроля четности (0 – чет, 1 – нечет)

Бит 3 (USBS) – 0 : 1 стоп-бит, 1: 2 стоп-бита

Бит 2 (UCSZ1) Бит 1 (UCSZ0) – задают количество бит данных (см. далее)

Бит 0 (UCPOL) – полярность тактового сигнала в синхронном режиме

Количество бит данных задается по таблице 24.

Таблица 24 – задание количества бит данных

UCSZ2	UCSZ1	UCSZ0	Количество бит данных
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Зарезервировано
1	1	1	9

В этом регистре нужно установить биты URSEL, UCSZ0 и UCSZ1 в единичное состояние:

```
UCSRC=(1<<URSEL)+(1<<UCSZ1)+(1<<UCSZ0);
```

Тогда функция настройки приемопередатчика будет выглядеть так:

```
void init_uart()
{
	unsigned int UBRR;
	UBRR=51;//Задаем скорость передачи
	UBRRL=UBRR&0b11111111;//Младший байт скорости передачи
	UBRRH=(UBRR>>8)&0b11111111;//Старший байт скорости передачи
	UCSRA=0;
	UCSRB=(1<<RXEN)+(1<<TXEN);//Включаем приемник и передатчик
	UCSRC=(1<<URSEL)+(1<<UCSZ1)+(1<<UCSZ0);//задаем объем передаваемых данных (8 бит)
}
```

Функцию передачи данных возьмем из документации к контроллеру:

```
void UART_Transmit(char data)
{
	while(!(UCSRA&(1<<UDRE)));//ожидание готовности передатчика
	UDR=data;//отправляем данные
}
```

В главной функции сначала вызовем функцию настройки приемопередатчика, затем передадим символы слова по очереди, после чего зацикливаем программу:

```
void main () //заголовок главной функции программы
{
	init_uart();//Настраиваем приемопередатчик
	UART_Transmit('H');//Передаем символы слова по очереди
	UART_Transmit('e');
	UART_Transmit('l');
	UART_Transmit('l');
	UART_Transmit('o');
	UART_Transmit('!');
```

```
    for(;;);//Бесконечный цикл
}
```

Соберите программу и запустите проект. Должно открыться окно терминала и в нем должен появиться текст, как показано на рисунке 75

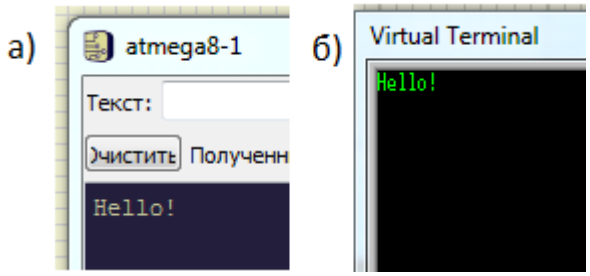

Рисунок 75 – принятый текст в терминале (а – в программе SimulIDE, б – в программе Proteus)

Если окно терминала не появилось, щелкните по контроллеру правой кнопкой и выберите пункт «Открыть Serial Monitor» в программе SimulIDE, или щелкните по терминалу правой кнопкой (после запуска проекта) и выберите в меню пункт “Virtual Terminal” В программе Proteus

11.5 Задание 2 – прием данных

Добавьте в проект функцию, которая включает светодиод, подключенный к выводу контроллера, указанному в таблице, если по терминалу принят символ «1» и выключает, если принят символ «0». Список вариантов приведен в таблице 25

Таблица 25 – список вариантов ко второму заданию

Вариант	Куда подключать светодиод	Вариант	Куда подключать светодиод
Пример	PB0	9	PC3
1	PB1	10	PC4
2	PB2	11	PC5
3	PB3	12	PD2
4	PB4	13	PD3
5	PB5	14	PD4
6	PC0	15	PD5
7	PC1	16	PD6
8	PC2		

Для этого добавим в программу функцию настройки порта:

```
DDRB=1<<PB0;//Настройка линии PB0 для светодиода на вывод
```

Также добавим функцию приема данных (используем стандартную функцию из документации):

```
char UART_Receive()
{
	while(!(UCSRA&(1<<RXC)));//ожидание готовности данных
	return UDR;//отправляем данные
}
```

В теле бесконечного цикла реализуем следующий алгоритм:

1.Принять байт

2. Если байт равен «0», выключить светодиод

3. Если байт равен «1», включить светодиод

```
char received;//Переменная, в которую помещаются принятые данные
for(;;)//Бесконечный цикл
{
	received=UART_Receive();//Принимаем данные в переменную
	if(received=='0')PORTB=0b00000000;//выключаем светодиод
	if(received=='1')PORTB=0b00000001;//включаем светодиод
};
```

Для проверки этого задания в программе SimulIDE следует вводить символы 0 и 1 в правое поле окна «Serial monitor», а в программе Proteus нужно щелкнуть правой кнопкой по окну терминала и выбрать пункт «Echo typed characters», после этого щелкнуть левой кнопкой по черному полю окна терминала, чтобы установить туда курсор и вводить символы «0» и «1» с клавиатуры. Светодиод должен загораться и гаснуть.

11.5 Контрольные вопросы

1. Как происходит передача данных с помощью USART?
2. Какую максимальную скорость USART можно установить при тактовой частоте 20 МГц?
3. Для чего может применяться USART?

Заключение

В данном учебно-методическом пособии рассмотрены основные вопросы, касающиеся разработки программ для микроконтроллеров на языке С, показано, как осуществляется ввод-вывод данных на примере простейших устройств. Рассмотрена работа с семисегментным и жидкокристаллическим индикаторами, матричной клавиатурой, работа с таймером и универсальным синхронно-асинхронным приемопередатчиком.

Также были рассмотрено моделирование и отладка микропроцессорных устройств средствами двух различных программ.

Одна из целей, которая ставилась при написании данного пособия – сделать изучение микропроцессорной техники более доступным и не слишком сложным. Эта задача вполне актуальна, так как зачастую книги по микропроцессорной технике содержат довольно сложные примеры программ, иногда состоящих из нескольких модулей, использующих различные дополнительные возможности языка программирования, что требует от учащихся высокого уровня начальной подготовки по программированию. Это вызывает большие затруднения у тех учащихся, для которых программирование не является основной специализацией. В данном пособии намеренно не предусмотрены слишком сложные примеры программ, поэтому оно должно быть доступно для учащихся широкого круга специальностей. Тем не менее, для лучшего усвоения материала, текомендуется также обратить внимание на специализированную литературу [7-9].

Список использованных источников

1. https://arduino.cc
2. https://sourceforge.net/projects/winavr/
3. https://www.simulide.com/
4. http://radio-hobby.org/modules/instruction/winavr-i-avr-studio/osnovy-si
5. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf
6. http://www.gaw.ru/html.cgi/txt/lcd/chips/hd44780/start.htm
7. 1 Белов А.В. Микроконтроллеры AVR. От азов программирования до создания практических устройств М.: Наука и техника, 2016 544 с.
8. 7 Белов А.В. - Разработка устройств на микроконтроллерах AVR: шагаем от чайника до профи М.: Наука и техника, 2016 528 с.
9. 8 Соснин, О. М. Средства автоматизации и управления - М. : Академия, 2014 - 240 с.