ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и АПУ»

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов направления подготовки 13.03.02 «Электроэнергетика и электротехника» дневной формы обучения

Могилев 2018

УДК 004.42
ББК 32.973-018.1
Я 41

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и АПУ» «6» февраля 2018 г., протокол № 7

Составитель ст. преподаватель В. Т. Вишнеревский

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации к лабораторным работам по дисциплине «Языки программирования» предназначены для студентов направления подготовки 13.03.02 «Электроэнергетика и электротехника» дневной формы обучения.

Учебно-методическое издание

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Ответственный за выпуск Г. С. Леневский
Технический редактор А. А. Подошевко
Компьютерная верстка Н. П. Полевничая

Подписано в печать .Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 46 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

1 Лабораторная работа № 1. Изучение среды компьютерной математики MATLAB

1.1 Ход работы

Целью лабораторной работы является:

1) изучение среды математических расчетов MATLAB;
2) получение практических навыков проведения в среде MATLAB инженерных расчетов.

Вначале следует получить вариант индивидуального задания по лабораторной работе у преподавателя, ее проводящего. После выполнения задания оформить отчет по лабораторной работе.

1.2 Порядок работы со средой MATLAB

Для начала работы с персональным компьютером следует в диалоговом окне ввести имя пользователя группы и пароль доступа. В случае успешного подключения на экране компьютера отобразится рабочий стол Windows и станет доступным каталог группы на компьютере.

Запуск оболочки интегрированной среды Matlab в Windows производится одним из следующих способов:

1) с помощью клика по соответствующему ярлыку на рабочем столе Windows;
2) выбором одноименного раздела в Стартовом меню Windows;
3) запуском на выполнение загрузочного файла c:\matlab\bin\matlab.exe.

В случае успешного запуска на экране раскроется окно среды MATLAB соответствующей версии.

В среде MATLAB команды следует вводить в окне Command Window.

В окне Workspace отображается список используемых в рабочей области переменных MATLAB с указанием их размера.

В окне Command History показывается история набранных команд среды MATLAB. При необходимости можно выбрать из приведенного перечня нужную команду с помощью мыши.

Перед началом работы следует установить с помощью кнопки (…) (*Browse for folder*), расположенной слева от выпадающего списка Current Directory, текущим свой рабочий каталог (папку) на компьютере. При этом раскроется диалоговое окно, в котором необходимо с помощью мыши указать нужную папку во внешней памяти ПК. Не рекомендуется непосредственно работать с флэшкой.

Если система MATLAB запускается на ПК повторно, тогда можно выбрать имя своего рабочего каталога из выпадающего списка Current Directory.

Скалярные переменные или скаляры (не имеющие внутренней структуры) задаются с помощью оператора присваивания (=) в следующем формате:

Имя_переменной=значение

При задании действительных данных дробная часть от целой отделяется символом «точка».

Пример задания значения скаляров a = 1.23 и b = –4 · 10^{-21} приведен на рисунке 1.1.

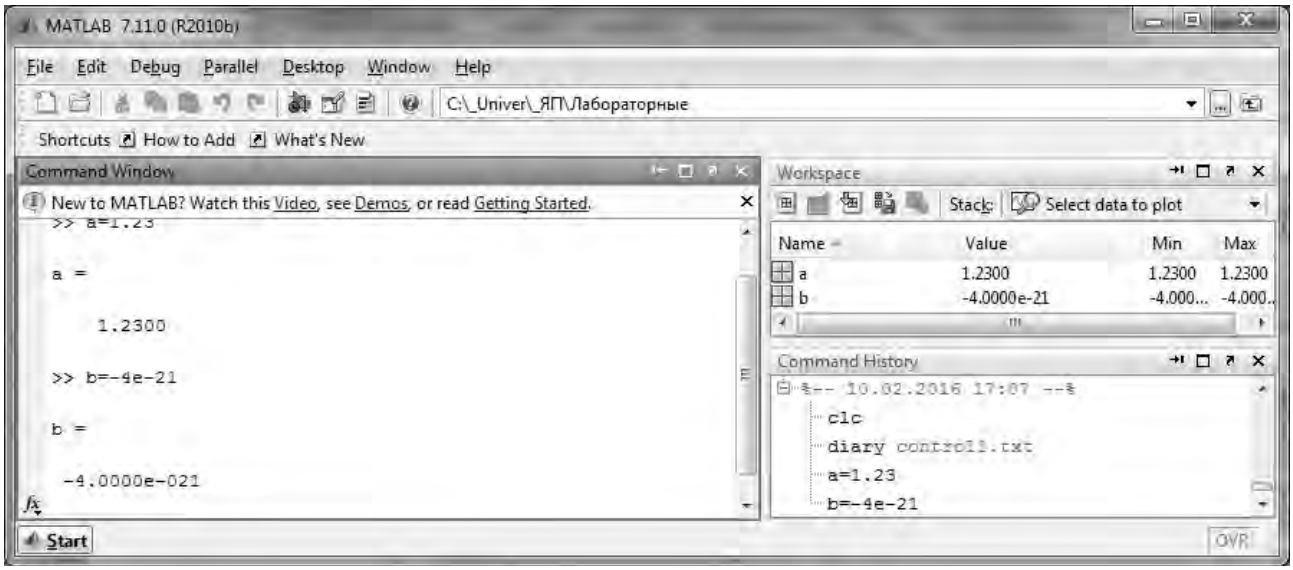

Рисунок 1.1 – Пример создания скаляров в среде MATLAB

В соответствии с заданием нужно создать две скалярные переменные. Первой присвоить значение дня рождения студента, второй – номер месяца. Далее выполнить следующие операции:

– сложение двух переменных;

– разность двух переменных;

– произведение двух переменных;

– деление первой переменной на вторую.

Для ввода команд можно использовать непосредственно командную строку из командного окна. Однако рекомендуется сразу создать m-файл, в котором впоследствии будет формироваться последовательность команд решения задания, которые можно легко редактировать. Окно редактора/отладчика вызывается с помощью раздела Главного меню Файл/New/m-file или комбинации клавиш Ctrl + N.

Структура окна аналогична ранее изучаемым встроенным редакторам IDE языков программирования.

Следует сразу сохранить m-файл в своем рабочем каталоге, нажав соответствующую кнопку или выбрав раздел File/Save As … . Имя файла должно задаваться только латинскими символами и не содержать пробелов.

В m-файле рекомендуется каждую команду записывать на отдельной строке в соответствии с синтаксисом, рассмотренным в разд. 4. Для запуска на выполнение команды следует использовать функциональную клавишу F5 (раздел Debug/Run). Это автоматически сохраняет все изменения в файле и выполняет его содержимое в командном окне MATLAB.

Результаты выполнения команд отразятся в командном окне Matlab.

Значение переменных можно посмотреть как с помощью командной строки, так и в рабочей области системы.

В начале m-файла рекомендуется выполнить команду очистки командного окна **clc**, а также очистки ранее созданных переменных в рабочей области **clear all**.

Все дальнейшие команды выполняются только в m-файле.

Построение графика.

Для дальнейшей работы с заданной функцией рационально оформить ее в виде отдельного m-файла в своем рабочем каталоге. Имя m-файла должно совпадать с именем объявляемой функции, например func.m. Для создания нового m-файла в окне редактора (рисунок 1.2) выбрать пункт File/New/m-file или нажать клавиши Ctrl + N. Сразу же следует сохранить файл под именем функции, выбрав раздел File/Save.

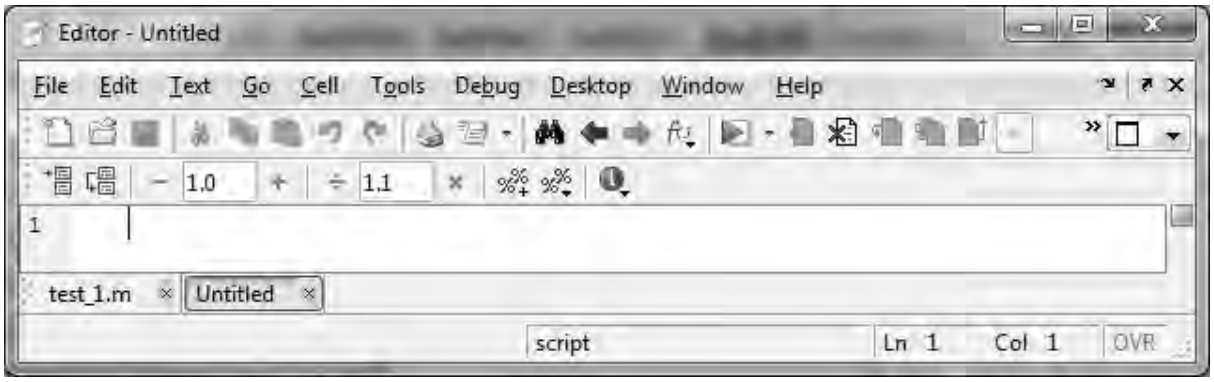

Рисунок 1.2 – Окно m-файла

Структура внешней m-функции в MATLAB имеет формат

```
function выходная_переменная = имя_функции(входная_переменная)
% текст комментария
выходная_переменная = выражение_функции
```

При записи выражения функции следует для операций с входной переменной указывать символ поэлементной обработки (символ «точка» перед символом операции). Пример реализации ранее рассмотренной функции в m-файле

```
function yy = func(xx);
% Выражение заданной функции
yy=xx.*xx-1;
```

Для построения графика функции предварительно рассчитываются два вектора: значений аргумента и функции. Значения массива-строки аргумента задаются в следующем формате:

```
вектор_аргумента = первое_значение : шаг_изменения : последнее_значение
```

Значение вектора функции рассчитывается с использованием описанной m-функции в формате

вектор_функции = имя_функции(вектор_аргумента)

Пример

```
X=0:0.02:2
Y=func(X)
```

Двухмерные графики в MATLAB строятся с помощью команды **plot**(). В списке параметров функции указывается как минимум одна переменная-вектор или расчетная функция, график которой должен быть построен. Для построения двухмерного графика рассчитанной функции используется следующий формат:

plot(вектор_аргумента, вектор_функции)

После этого на экране раскроется окно с изображением графика функции (пример на рисунке 1.3). Для отображения линий сетки следует использовать команду **grid on**. Пример построения графика в среде MATLAB

```
plot(X,Y)
grid on
```

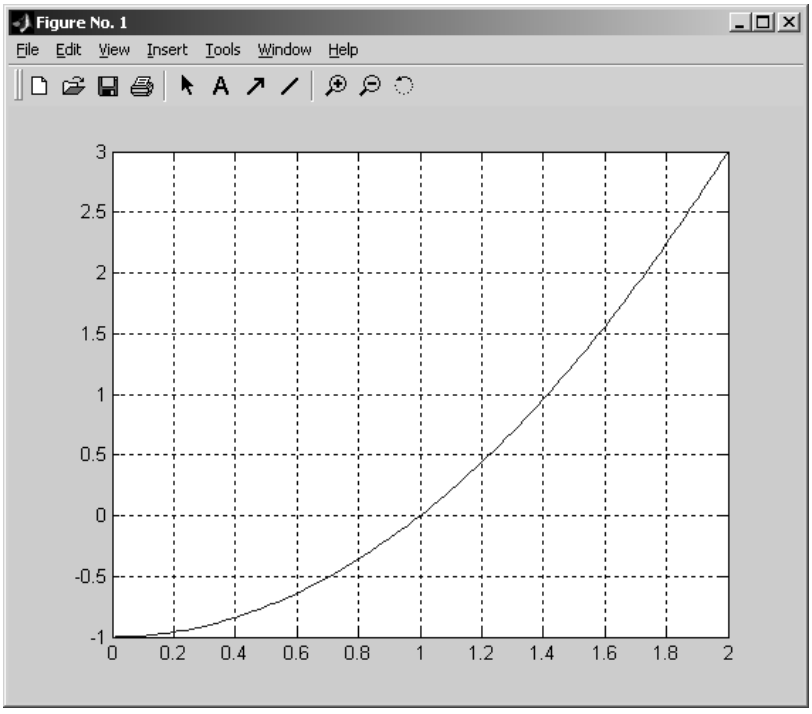

Рисунок 1.3 – Пример графика

Экспорт данных из MATLAB.

Экспорт из MATLAB численных данных можно проводить следующими способами:

– выделение нужной области текста в командном окне MATLAB и копирование его в буфер обмена Windows (клавиши Ctrl + C) с последующей

вставкой в документ отчета;

– сохранение переменных рабочей области MATLAB в mat-файле для повторного использования при следующем сеансе работы со средой MATLAB;

– использование команд **save** для сохранения значений отдельных переменных в текстовых файлах формата ANSII;

– копирование данных отдельных переменных рабочей области через буфер обмена из окна просмотра данных (рисунок 1.1).

Экспорт графических данных.

Для экспорта графических данных из окна графики MATLAB (рисунок 1.3) можно использовать:

1) сохранение графики в отдельный файл формата emf (векторный) или bmp (растровый), используя раздел окна графики File/Save Ass;

2) копирование графика через буфер обмена Windows, тогда следует воспользоваться разделом Edit/Copy Figure. При этом предварительно рекомендуется установить тип копирования графики как метафайл Windows (wmf), используя раздел File/Preferences/Copy Options (рисунок 1.4).

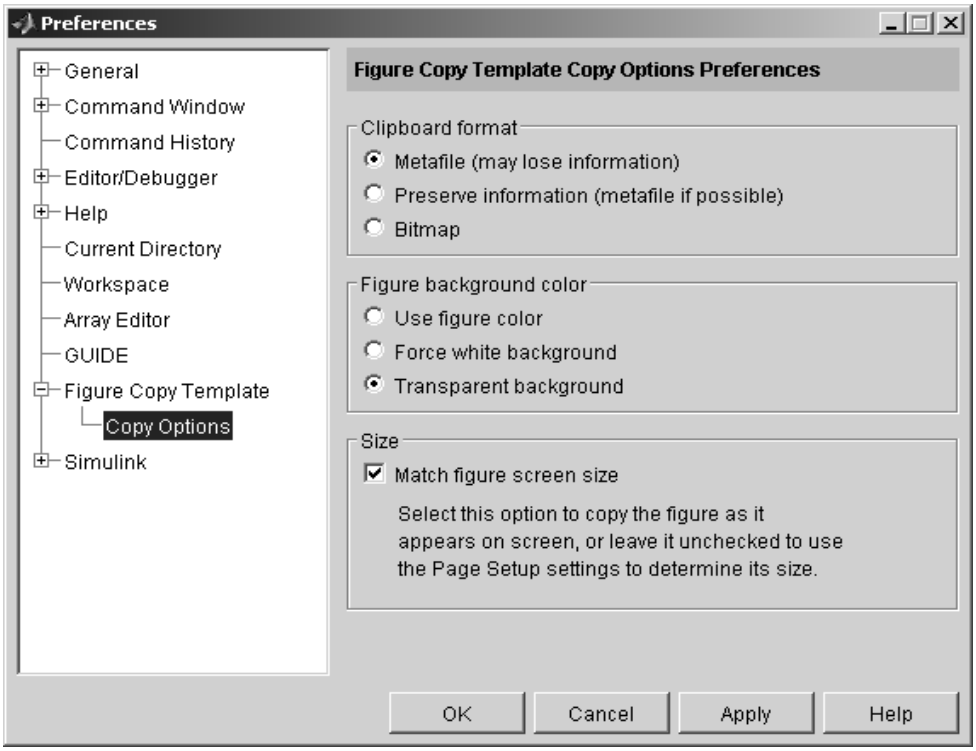

Рисунок 1.4 – Настройка параметров копирования графических данных

Окончание работы в среде MATLAB.

Перед окончанием работы следует сохранить результаты расчета в файлах и проверить сохранение m-файла.

По окончании работы производится выход из MATLAB одновременным нажатием клавиш Alt + F4 или Ctr + Q, а также вводом команды **exit** или **quit** в командном окне. Предварительно следует сохранить m-файл на жестком магнитном диске компьютера.

Окончание сеанса работы с персональным компьютером.

Перед тем как закончить сеанс с персональным компьютером, следует удалить ненужные файлы из своего рабочего каталога и провести резервное копирование данных на внешний накопитель.

Для окончания сеанса работы в сети следует закрыть рабочий каталог на сервере в среде Windows нажатием кнопки **Пуск** и указанием раздела Завершение сеанса … или *Закончить работу*.

1.3 Основные теоретические сведения

Назначение пакета MATLAB.

MATLAB (от matrix laboratory – матричная лаборатория) – это интерактивная вычислительная среда со специализированным математическим языком, предназначенная для выполнения научных и технических расчетов. Она включает:

– математические вычисления;
– создание алгоритмов;
– моделирование;
– анализ данных, исследования и визуализацию;
– научную и инженерную графику;
– разработку приложений, включая создание графического интерфейса.

Основным элементом данных MAQTLAB является матрица.

Система MATLAB состоит из пяти основных частей:

1) язык высокого уровня MATLAB – это язык матриц и массивов с управлением потоками, функциями, структурами данных, вводом-выводом и особенностями объектно-ориентированного программирования;

2) среда MATLAB – набор инструментов и приспособлений, с которыми работает пользователь. Включает в себя средства для управления переменными в рабочем пространстве MATLAB, вводом и выводом данных, а также создания, контроля и отладки m-файлов и приложений MATLAB;

3) управляемая графика – графическая система MATLAB, которая включает в себя команды высокого уровня для визуализации двух- и трехмерных данных, обработки изображений, анимации и иллюстрированной графики. Она также включает в себя команды низкого уровня, позволяющие полностью редактировать внешний вид графики, также как при создании Графического Пользовательского Интерфейса (GUI) для приложений MATLAB;

4) библиотека математических функций – коллекция вычислительных алгоритмов, охватывающая все разделы математики;

5) программный интерфейс – библиотека, которая позволяет писать программы на Си и Фортране, взаимодействующие с MATLAB. Включает средства для вызова программ из MATLAB (динамическая связь), вызывая MATLAB как вычислительный инструмент и для чтения-записи mat-файлов.

Ввод матриц и операции над ними в среде MATLAB.

Можно вводить матрицы в MATLAB несколькими способами:

– вводить полный список элементов;
– загружать матрицы из внешних файлов;

– генерировать матрицы, используя встроенные функции;

– создавать матрицы с помощью ваших собственных функций в m-файлах.

При вводе матрицы по строкам как списка элементов с клавиатуры необходимо соблюдать основные условия:

– отделять элементы строки пробелами или запятыми;

– использовать точку с запятой «;» для обозначения конца каждой строки;

– окружать весь список элементов квадратными скобками [ ].

Пример ввода в переменную среды A матрицы Дюрера:

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1].
```

Если взять сумму элементов вдоль какой-либо строки или столбца или вдоль какой-либо из двух главных диагоналей, то всегда получается одно и тоже число. В MATLAB суммирование матриц выполняет команда sum():

```
sum (A)
```

Когда выходная переменная не определена, то используется переменная *ans* для хранения результатов вычисления.

Арифметические операторы и функции MATLAB.

В отличие от большинства языков программирования в системе MATLAB практически все операторы являются матричными и предназначены для выполнения операций над матрицами.

Каждый оператор имеет аналогичную по назначению функцию.

Приоритет логических операторов выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше приоритета сложения и вычитания. Для изменения приоритета операций в математических выражениях используются круглые скобки. Степень вложения скобок не ограничивается.

Операторы и функции сравнения в MATLAB.

Операторы отношения служат для сравнения двух величин, векторов или матриц. Все операторы отношения имеют два операнда (x и y).

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера и возвращают значение 1 (True), если элементы идентичны, и значение 0 (False) – в противном случае. Примеры использования:

```
>> eq(2,2)
ans = 1
>> 2==2
ans = 1
>> ne(l,2)
 ans = 1
>> 2 ~- 2
ans = 0
```

Следует отметить, что операторы <, <=, > и >= при комплексных операндах используют для сравнения только действительные части операндов – мнимые отбрасываются. В то же время операторы == и ~= ведут сравнение с учетом как действительной, так и мнимой частей операндов.

Если один из операндов – скаляр, происходит сравнение всех элементов второго операнда-массива со значением этого скаляра. В общем случае операторы отношения сравнивают два массива одного размера и выдают результат в виде массива того же размера.

Применение операторов отношения в системе MATLAB расширено, поскольку операндами являются не только числа, но и векторы, матрицы и массивы. Возможно применение операторов отношения и к символьным выражениям. В этом случае символы, входящие в выражения, представляются своими ASCII-кодами. Строки воспринимаются как векторы, содержащие значения кодов.

Логические операторы MATLAB.

Логические операторы и соответствующие им функции служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов.

Примеры использования:

```
>>A=[1 2 3];
>>B=[1 0 0];
>> and(A,B)
ans = 1
0
0
>> or(A,B)
ans = 111
>> A&B
ans = 100
>> A|B
ans= 111
>> not(A)
 ans = 000
>> all(B)
ans = 0
>> and('abc','012')
ans = 111
```

Аргументами логических операторов могут быть числа и строки. При аргументах-числах логический нуль соответствует числовому нулю, а любое отличное от нуля число воспринимается как логическая единица. Для строк действует правило – каждый символ строки представляется своим ASCII-кодом.

Математические функции.

В MATLAB определены следующие математические функции:

1) **abs**(X) – возвращает абсолютную величину для каждого числового

элемента вектора X. Для комплексных чисел вычисляется модуль каждого числа;

2) **exp**(X) – возвращает экспоненту для каждого элемента X;

3) G=**gcd**(A,B) – возвращает массив, содержащий наибольшие общие делители соответствующих элементов массивов целых чисел A и B;

4) [G,C,D]=**gcd**(A,B) – возвращает массив наибольших общих делителей G и массивов C и D;

5) **lcm**(A,B) – возвращает наименьшие общие кратные для соответствующих парных элементов массивов A и B. Массивы A и B должны содержать положительные целые числа и иметь одинаковую размерность;

6) **log**(X) – возвращает натуральный логарифм элементов массива X;

7) **log2**(X) – возвращает логарифм по основанию 2 элементов массива X;

8) **log10**(X) – возвращает логарифм по основанию 10 для каждого элемента X. Область функции включает комплексные числа, что способно привести к непредвиденным результатам при некорректном использовании;

9) **mod**(x,y) – возвращает остаток от деления X на Y;

10) **pow2**(Y) – возвращает массив X, где каждый элемент есть 2^Y;

11) **sqrt**(A) – возвращает квадратный корень каждого элемента массива X.

Тригонометрические функции MATLAB.

В системе MATLAB определены следующие тригонометрические и обратные тригонометрические функции:

1) **acos** (X) – возвращает арккосинус для каждого элемента X;

2) **acsc**(X) – возвращает арккосеканс для каждого элемента X;

3) **asec**(X) – возвращает арксеканс для каждого элемента X;

4) **asin**(X) – возвращает арксинус для элементов действительных значений X в области [–1, 1], asin(X) возвращает действительное число из диапазона [–π/2; π/2], для значений X вне области [–1, 1] возвращает комплексное число;

5) **atan**(X) – возвращает арктангенс для каждого элемента X. Для действительных значений X ответ находится в области [–π/2; π/2];

6) **atan2** (Y, X) – возвращает вектор, содержащий арктангенсы отношения вещественных частей Y и X, игнорируя мнимые части, в интервале [–π; π];

7) **cos**(X) – возвращает косинус для каждого элемента X;

8) **cot**(X) – возвращает котангенс для каждого элемента X;

9) **csc**(X) – возвращает косеканс для каждого элемента X;

10) **sec**(X) – возвращает массив той же размерности, что и X, состоящий из секансов элементов X;

11) **sin**(X) – возвращает синус для каждого элемента X;

12) **tan**(X) – возвращает тангенс для каждого элемента X.

Функции вычисляются для каждого элемента массива. Входной массив допускает комплексные значения. Все углы задаются в радианах.

Специальные функции.

В MATLAB функция expint обозначена как expint(X) и возвращает интегральную показательную функцию для каждого элемента X.

Пример

```
>> d=expint([2,3+7i])
d = 0.0489 -0.0013 -0.00601
```

Определитель и ранг матрицы.

Для нахождения определителя (детерминанта) и ранга матриц в MATLAB имеются следующие функции:

1) **det**(X) – возвращает определитель квадратной матрицы X. Если X содержит только целые элементы, то результат – тоже целое число;

2) **rank**(A) – вычисления ранга, возвращает количество сингулярных чисел, которые являются большими, чем заданный по умолчанию допуск;

3) **rank**(A,tol) – возвращает количество сингулярных чисел, которые превышают tol.

Вычисление следа матрицы.

След матрицы A – это сумма ее диагональных элементов.

Функция trace(A) возвращает след матрицы.

Пример

```
>> a=[2 3 4; 5 6 7; 8 9 1]
a = 234
567
891
>> trace(a)
ans = 9
```

Обращение матриц.

Обратной называют матрицу, получаемую в результате деления единичной матрицы E на исходную матрицу X. Таким образом, X^-1=E/X. Следующие функции обеспечивают реализацию данной операции:

1) **inv**(X) – возвращает матрицу, обратную квадратной матрице X;

2) B=**pinv**(A) – возвращает матрицу, псевдообратную матрице A (по Муру-Пенроузу). Результатом псевдообращения матрицы является матрица B того же размера, что и A', и удовлетворяющая условиям A*B*A=A и B*A*B=B.

Численное решение инженерных задач в MATLAB.

СЛАУ может быть представлена в матричном виде как

$$A \cdot X = B,$$

где A – матрица коэффициентов уравнений;

X – искомый вектор неизвестных;

B – вектор свободных членов.

MATLAB имеет два различных типа арифметических операций – поэлементные и для массивов (векторов и матриц) в целом. Матричные арифметические операции определяются правилами линейной алгебры.

Арифметические операции сложения и вычитания над массивами

выполняются поэлементно. Знак точки «.» отличает операции над элементами массивов от матричных операций. Однако, поскольку операции сложения и вычитания одинаковы для матрицы и элементов массива, знаки «.+» и «.–» не используются.

Для решения СЛАУ чаще всего используется операция обращения x = inv(A)*B или оператор матричного деления x = A\b. Эта операция использует метод исключения Гаусса без явного формирования обратной матрицы.

Вычисление нулей (корней) функции одной переменной.

Задача решения нелинейного уравнения используется для функций вида f(x) = 0 или f1(x) = f2(x), которые можно свести к виду f(x) = f1(x) – f2(x) = 0.

Задача сводится к нахождению значений аргумента x функции f(x) одной переменной, при котором значение функции равно нулю. Функции MATLAB:

1) **fzero**(@fun,x_o) – возвращает уточненное значение x, при котором достигается нуль функции fun, представленной в символьном виде с начальным значением x_o. Возвращенное значение равно NaN, если решение не найдено;

2) **fzero**(@fun,[xl x2]) – возвращает значение x, при котором fun(x) = 0 с заданием интервала поиска с помощью вектора x = [xl x2], у которого знак fun(x(1)) отличается от знака fun(x(2));

3) **fzero**(@fun,x,tol) – возвращает результат с заданной погрешностью tol;

4) **fzero**(@fun,x,tol, trace) – выдает информацию о каждой итерации;

5) **fzero**(@fun,x,tol,trace,P1,P2,...) – предусматривает дополнительные аргументы, передаваемые в функцию fun(x,Pl,P2,...). При задании пустой матрицы для tol или trace используются значения по умолчанию.

В случаях, когда функция имеет несколько действительных корней, рекомендуется строить график функции f(x) для приближенного отделения корней и интервалов. Пример содержимого m-файла f:

```
function f=funl(x)
%Функция, корни которой ищутся
f=0.25*x+sin(x)-1;
» x=0:0.1:10;
» plot(x,funl(x));grid on;
```

Для решения систем нелинейных уравнений следует также использовать функцию solve из пакета Symbolic Math Toolbox. Эта функция способна выдавать результат в символьной форме, а если такого нет, то она позволяет получить решение в численном виде.

Пример

```
>> solve('0.25*x + sin(x) -1)
ans =  .89048708074438001001103173059554
```

Поиск локального минимума функций.

Двухпараметрическая функция описывает некоторую поверхность в трехмерном пространстве и записывается в виде

$$Z=f(x,y),$$

где x, y – независимые переменные (аргументы);

z – зависимая переменная (значения функции).

В MATLAB более удобна следующая форма записи много-параметрических функций:

$$Z=f(X_1,X_2,\ldots,X_n),$$

где X_1, X_2, …, X_n – независимые переменные (аргументы);

Z – зависимая переменная (значения функции).

Для определения локального минимума функции в MATLAB используется стандартная функция **fmins** или **fminsearch** со следующим синтаксисом:

1) xmin = **fmins**(‘имя_функции‘, x0) – возвращает вектор xmin, соответствующий значению локального минимума функции в окрестности точки x0;

2) xmin = **fmins**(‘имя_функции‘, x0, options) – использует вектор управляющих параметров options, который включает 18 компонентов, соответствующих функции foptions. Функция fmins использует только четыре из этих параметров: options(1), options(2), options(3), options(14);

3) [xmin,options]=**fmins**(‘имя_функции‘, x0, options, [ ], arg1,..., arg10) – позволяет передать до 10 параметров; четвертый входной аргумент необходим, чтобы обеспечить совместимость с функцией fminu из пакета Optimization Toolbox. Возвращается вектор управляющих параметров options, которые использовались алгоритмом: параметр options(10), фиксирующий количество выполненных итераций; параметр options(8), содержащий минимальное значение функции.

Содержание отчета

Отчет выполняется индивидуально каждым студентом на листах формата А4. Текст отчета оформляется в соответствии с требованиями ГОСТ 2.105–95. В состав отчета включаются:

– титульный лист;

– индивидуальное задание;

– распечатка электронного m-файла с индивидуальным заданием;

– результаты решения поставленных задач индивидуального задания.

Контрольные вопросы

1 Для каких целей используется система MATLAB?

2 Какой состав имеет главное меню системы MATLAB?

3 Каким образом определяются корни алгебраических уравнений в системе MATLAB?

4 Как решить нелинейное уравнение в MATLAB?

5 Как решить систему линейных уравнений в MATLAB?

6 Каким образом строятся графики функций в MATLAB?

7 Каким образом выполняется поиск определенного интеграла в MATLAB?

8 Как взять производную от заданной функции в MATLAB?

9 Как определить выражение неопределенного интеграла в MATLAB?

10 Как построить график однопараметрической функции в среде MATLAB?

11 Каким образом строится трехмерный график двухпараметрической функции в среде MATLAB?

12 Каким образом в среде MATLAB найти локальный минимум двухпараметрической функции?

13 Каким образом экспортировать расчетные данные из MATLAB в текстовые документы?

Варианты индивидуальных заданий

Индивидуальные задания выдаются преподавателем, проводящим лабораторную работу. В задании следует выполнить следующие операции:

1) найти определитель, обратную матрицу и вектор собственных значений для заданной в таблице 1.1 матрицы A;

2) решить систему линейных алгебраических уравнений (СЛАУ), которая задается в виде матриц A и B из таблицы 1.1;

3) решить нелинейное уравнение f(x) = 0, заданное в виде функции в таблице 1.2;

4) построить график функции y = f(x) на заданном в таблице 1.2 интервале;

5) найти определенный интеграл для подынтегральной функции и в границах заданной в таблице 1.2;

6) найти значение первой и второй производных (dy/dx и d^2y/dx^2) для функции y = f(x) в заданных точках a и b согласно таблице 1.2;

7) решить алгебраическое уравнение, заданное в виде коэффициентов степенного полинома в таблице 1.3;

8) найти локальный минимум двухпараметрической функции согласно данным таблицы 1.4.

Таблица 1.1 – Варианты заданий для решения СЛАУ

Вариант	Матрица A			Вектор B
1	2	4	–2	–4
	1	5	3	10
	1	3	2	5
2	1	1	6	7
	–1	2	9	2
	1	–2	3	10
3	2	–2	5	6
	2	3	1	13
	–1	4	–4	3

Окончание таблицы 1.1

Вариант	Матрица A			Вектор B
4	–5	2	–1	–1
	1	0	3	5
	3	1	6	17
5	–2	1	5	15
	4	–8	1	21
	4	–1	1	7
6	5	–1	1	10
	2	8	–1	11
	–1	1	4	3
7	2	8	–1	11
	5	–1	1	10
	–1	1	4	3
8	1	–5	–1	–8
	4	1	–1	13
	2	–1	–6	–2
9	4	1	–1	13
	1	–5	–1	–8
	2	–1	–6	–2
10	1	0	1	2
	–1	1	0	0
	1	2	–3	0

Таблица 1.2 – Варианты функций для решения нелинейных уравнений

Вариант	Функция	Интервал поиска решения	
		начало	конец
1	$f(x) = x^2 - e^x$	–2	2
2	$f(x) = x - \cos(x)$	–2	2
3	$f(x) = \sin(x) - 2\cdot\cos(x)$	–2	2
4	$f(x) = \cos(x) + (1+x^2)^{-1}$	–2	2
5	$f(x) = (x - 2)^2 - \ln(x)$	0,5	4,5
6	$f(x) = 2x - \mathrm{tg}(x)$	–1,4	1,4
7	$f(x) = x^2 - 20\cos(x + 1)$	–5	0
8	$f(x) = x^2 - 1 - 5\lvert 2x + 4\rvert^{1/2}$	–5	0
9	$f(x) = 20\sin(\mathrm{arctg}(x) + 1)$	–5	0
10	$f(x) = 20\ln(x^2 + 1) - 0.1x^3$	5	15

Таблица 1.3 – Коэффициенты уравнения $a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0 = 0$

Номер	Коэффициент уравнения				
	a_4	a_3	a_2	a_1	a_0
1	0,0005	0	0,07	0,12	1
2	0,0008	0,002	0,5	0,1	1
3	0,0002	0,005	0,4	0,3	1
4	0,0003	0,006	0,1	0,09	1
5	0,00006	0,001	0,02	0,1	1
6	0,0001	0,004	0,5	0,7	1
7	0,0004	0,004	0,06	0,1	1
8	0,00007	0,003	0,05	0,2	1
9	0,00009	0,1	0,04	0,7	1
10	0,00003	0,0007	0,02	0,1	1

2 Лабораторная работа № 2. Изучение особенностей программирования микроконтроллеров на языке С

2.1 Ход работы

Целью работы является изучение особенностей программирования микроконтроллеров с помощью языка программирования С. Также нужно составить программу по заданию преподавателя, перевести ее в машинные коды, записать в память программ микроконтроллера и выполнить.

Перед началом лабораторных занятий студентам необходимо изучить следующие вопросы:

– язык программирования С;

– особенности разработки программ в среде Atmel Studio;

– использование среды Proteus для моделирования микропроцессорных систем.

В ходе выполнения работы:

– изучить электрическую принципиальную схему к лабораторной работе;

– разработать программу в соответствии с индивидуальным заданием;

– отладить программу в среде Atmel Studio;

– произвести моделирование в среде Proteus;

– оформить отчет по лабораторной работе.

После выполнения работы необходимо ответить на контрольные вопросы.

Разработка программы в среде Atmel Studio.

Окно программы Atmel Studio после запуска представлено на рисунке 2.1.

В открывшемся окне нужно создать новый проект, как показано на рисунке 2.2.

Затем следует произвести настройки, как показано на рисунке 2.3.

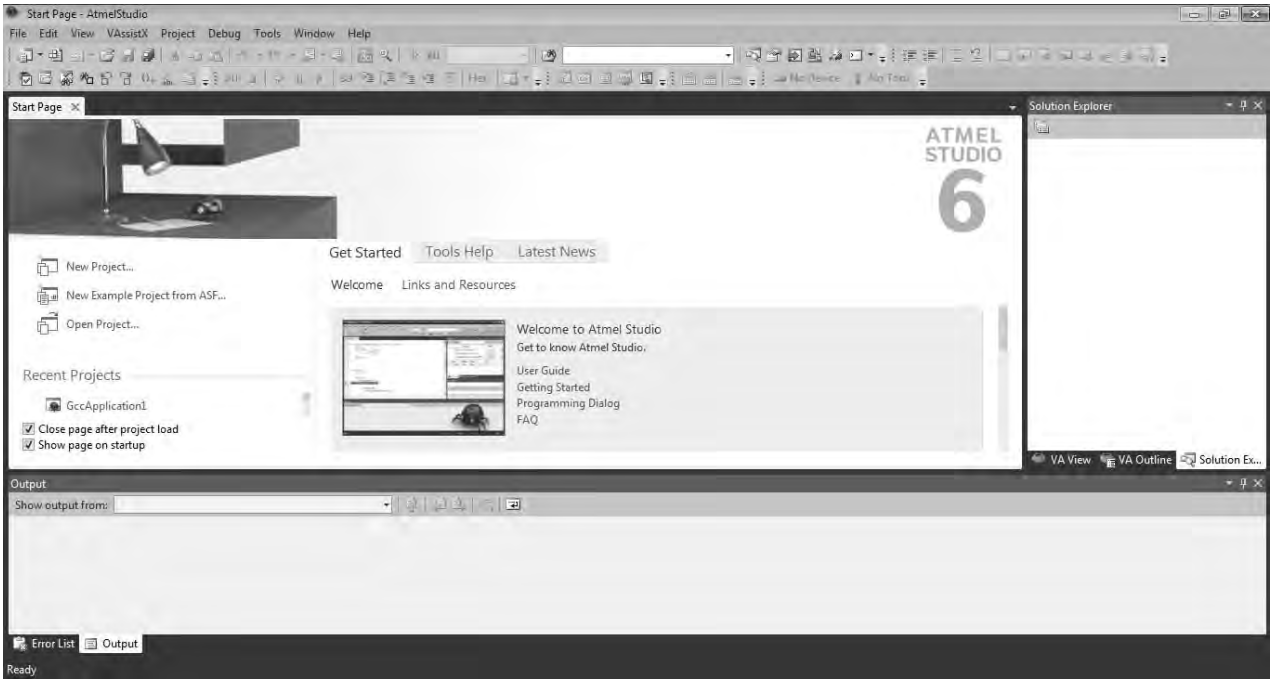

Рисунок 2.1 – Окно программы Atmel Studio после запуска

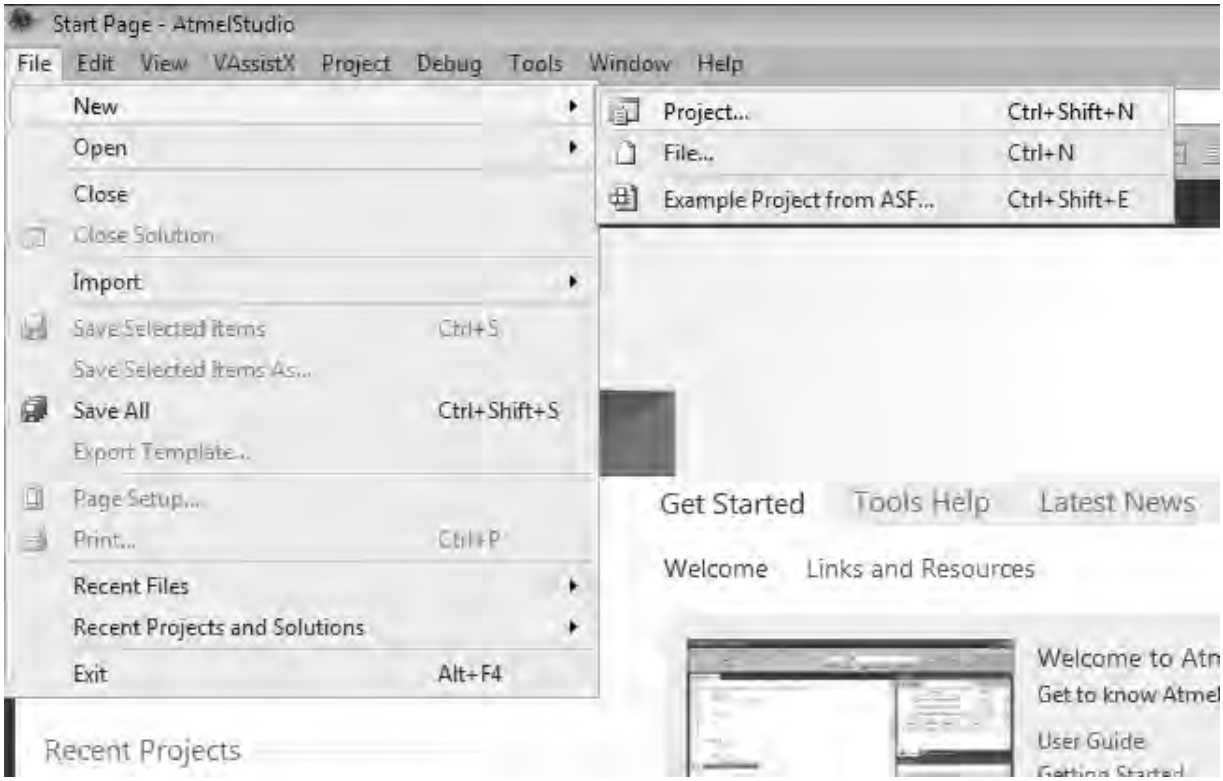

Рисунок 2.2 – Вкладка создания нового проекта в среде Atmel Studio

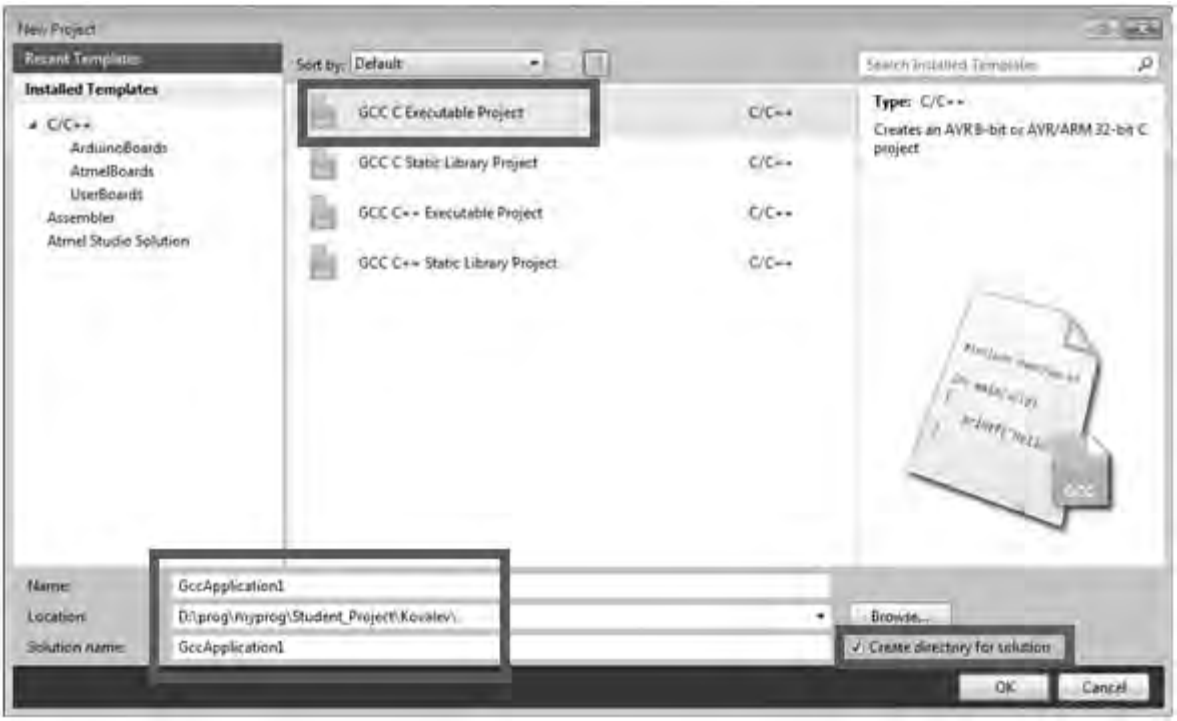

Рисунок 2.3 – Настройка проекта

Далее нужно выбрать микроконтроллер ATmega16A (или иной из семейства megaAVR), как показано на рисунке 2.4.

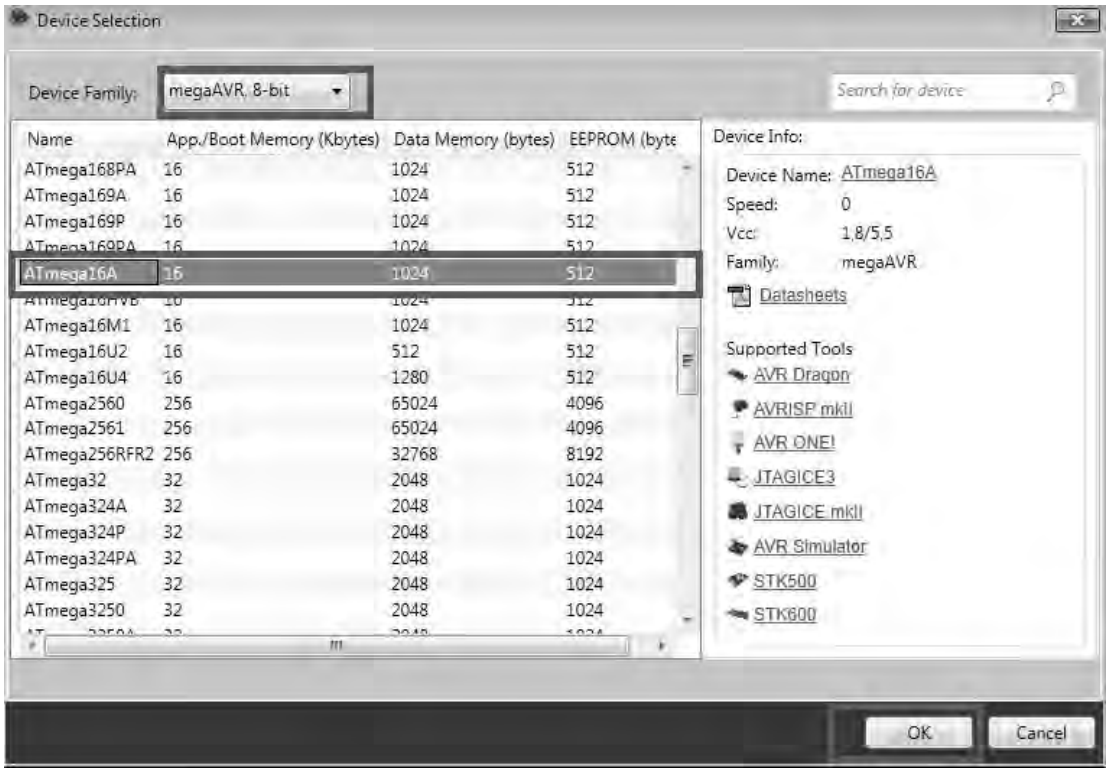

Рисунок 2.4 – Выбор микроконтроллера

После этого откроется созданный проект. Здесь следует обратить внимание на функцию int main(void) и цикл while(1) (рисунок 2.5).

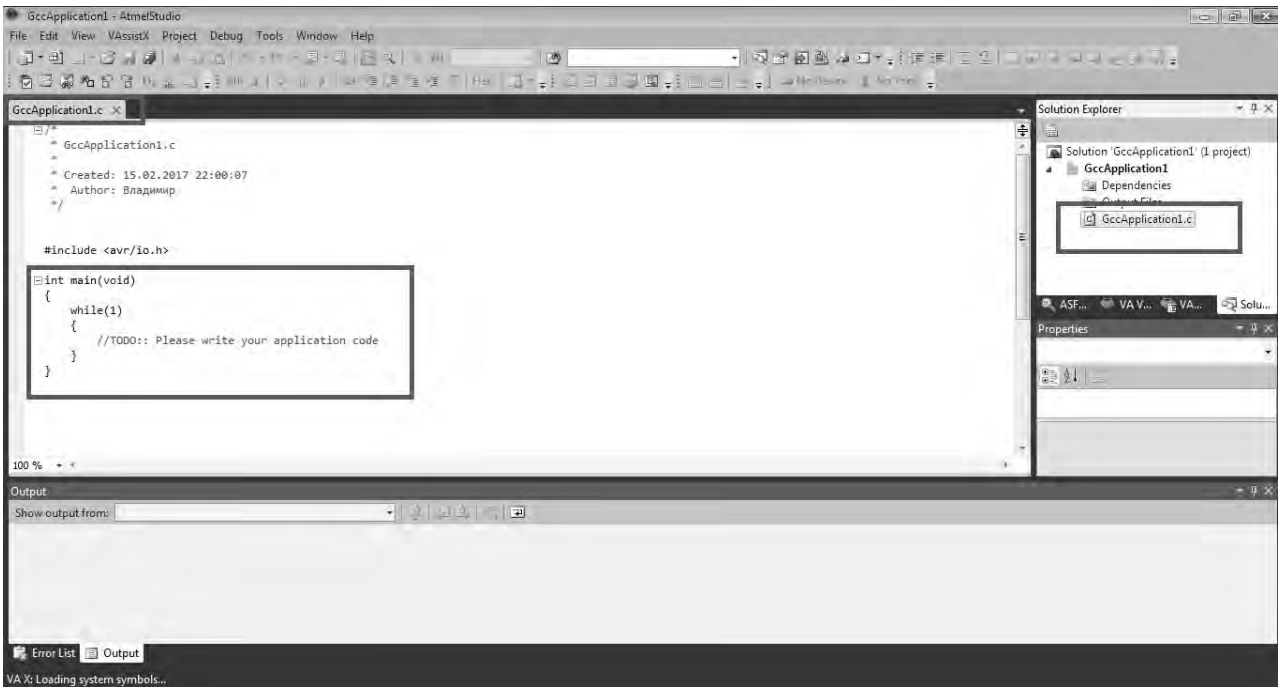

Рисунок 2.5 – Окно созданного проекта

Пример кода программы:

```
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
uint8_t a = 0;
uint8_t b = 0;
DDRD = 0xFF;
DDRC = 0x00;
PORTC = 0xFF;

    while(1)
    {
        //TODO:: Please write your application code
a = PINC;
b = a;
a = a & 0x0F;
b = b & 0xF0;
b = b >> 4;
PORTD = b | a;
    }
}
```

После написания и компиляции программы следует произвести моделирование в среде Proteus.

Окно программы Proteus после запуска представлено на рисунке 2.6.

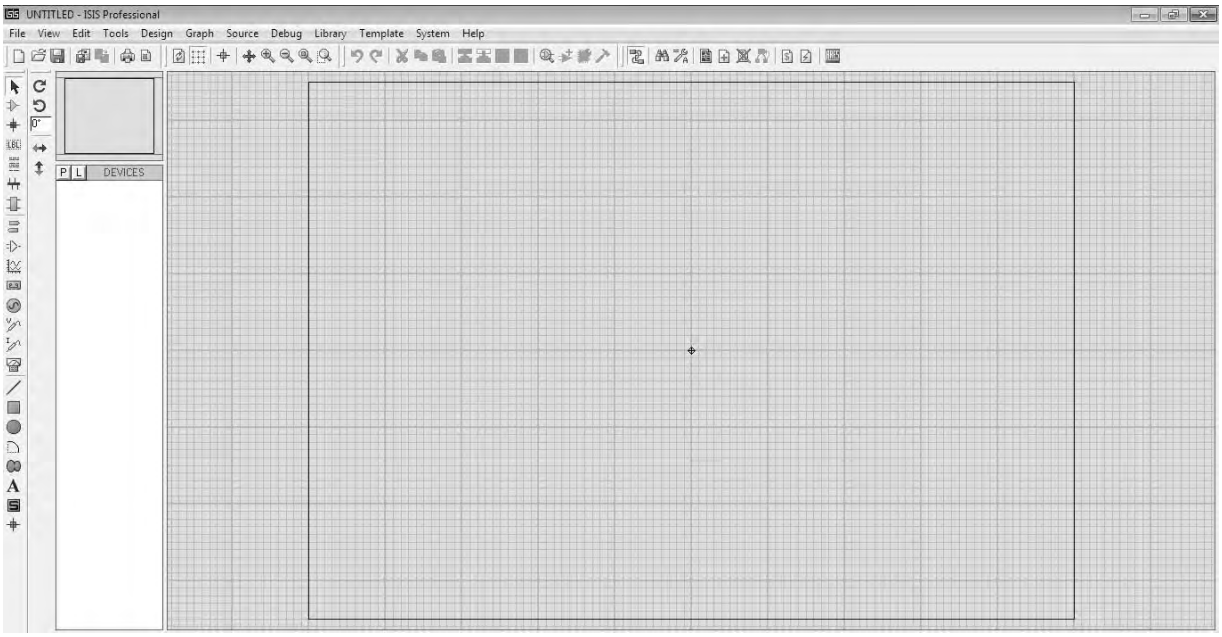

Рисунок 2.6 – Окно программы Proteus

Пример схемы в среде Proteus приведен на рисунке 2.7.

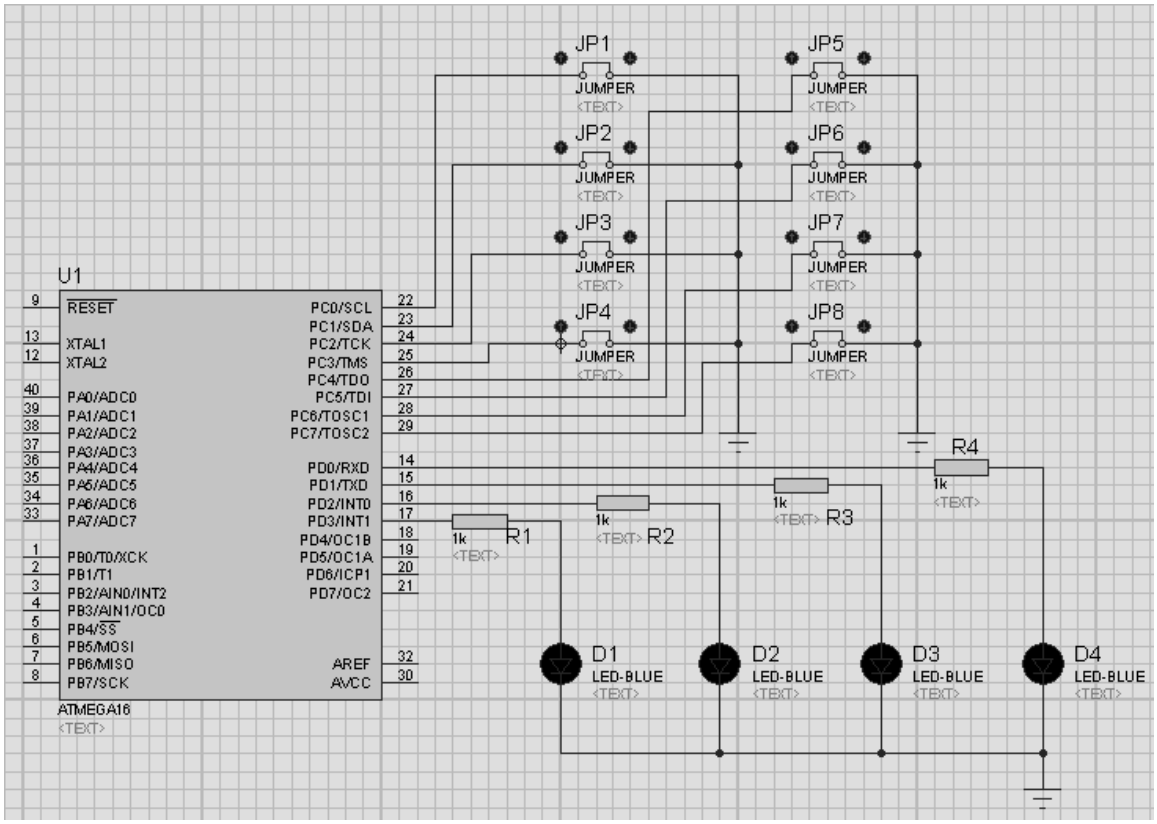

Рисунок 2.7 – Пример схемы в среде Proteus

Для того чтобы собрать схему, представленную на рисунке 2.7, необходимо разместить на рабочем поле следующие компоненты из стандартной библиотеки среды Proteus:

– микроконтроллер ATmega16;
– элемент Jumper;
– резистор 1 кОм;
– светодиод BLUE-LED.

Элемент «Земля» можно найти во вкладке, показанной на рисунке 2.8.

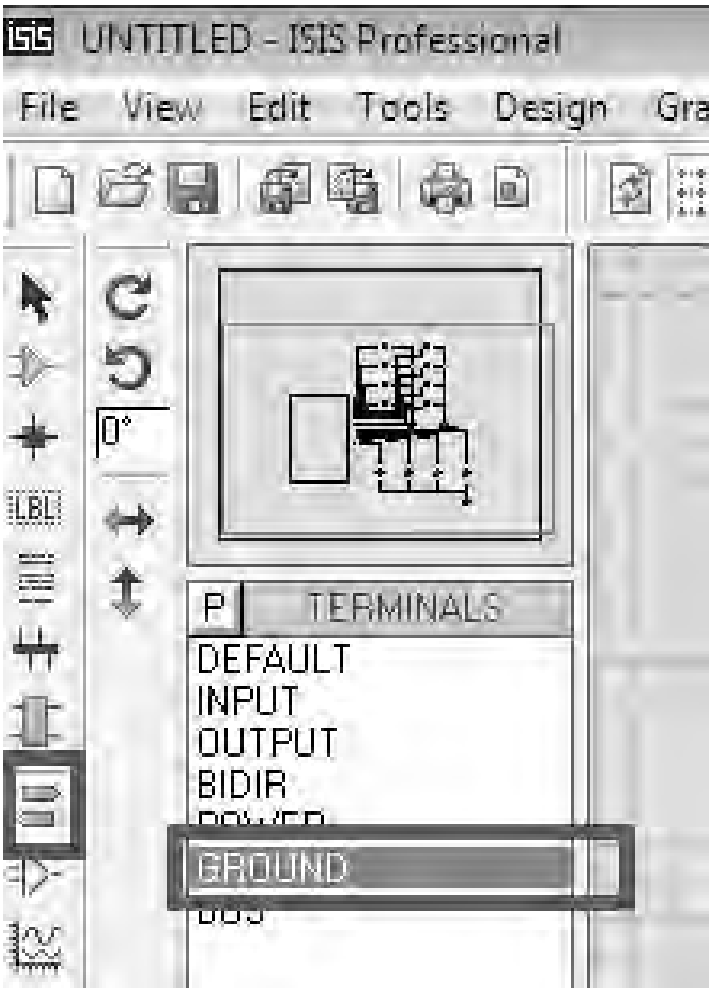

Рисунок 2.8 – Выбор элемента «Земля»

Для осуществления настройки модели микроконтроллера нужно дважды кликнуть на его изображении на рабочем поле среды моделирования Proteus (рисунок 2.9).

В открывшемся окне следует указать путь к файлу прошивки, полученному в результате компиляции проекта в среде Atmel Studio. Данный файл имеет расширение .hex и находится в папке /debug.

Также нужно выбрать частоту работы микроконтроллера, в данном случае 8 МГц.

После того как модель собрана и указан путь к файлу прошивки, для начала моделирования следует нажать соответствующую клавишу,

как показано на рисунке 2.10.

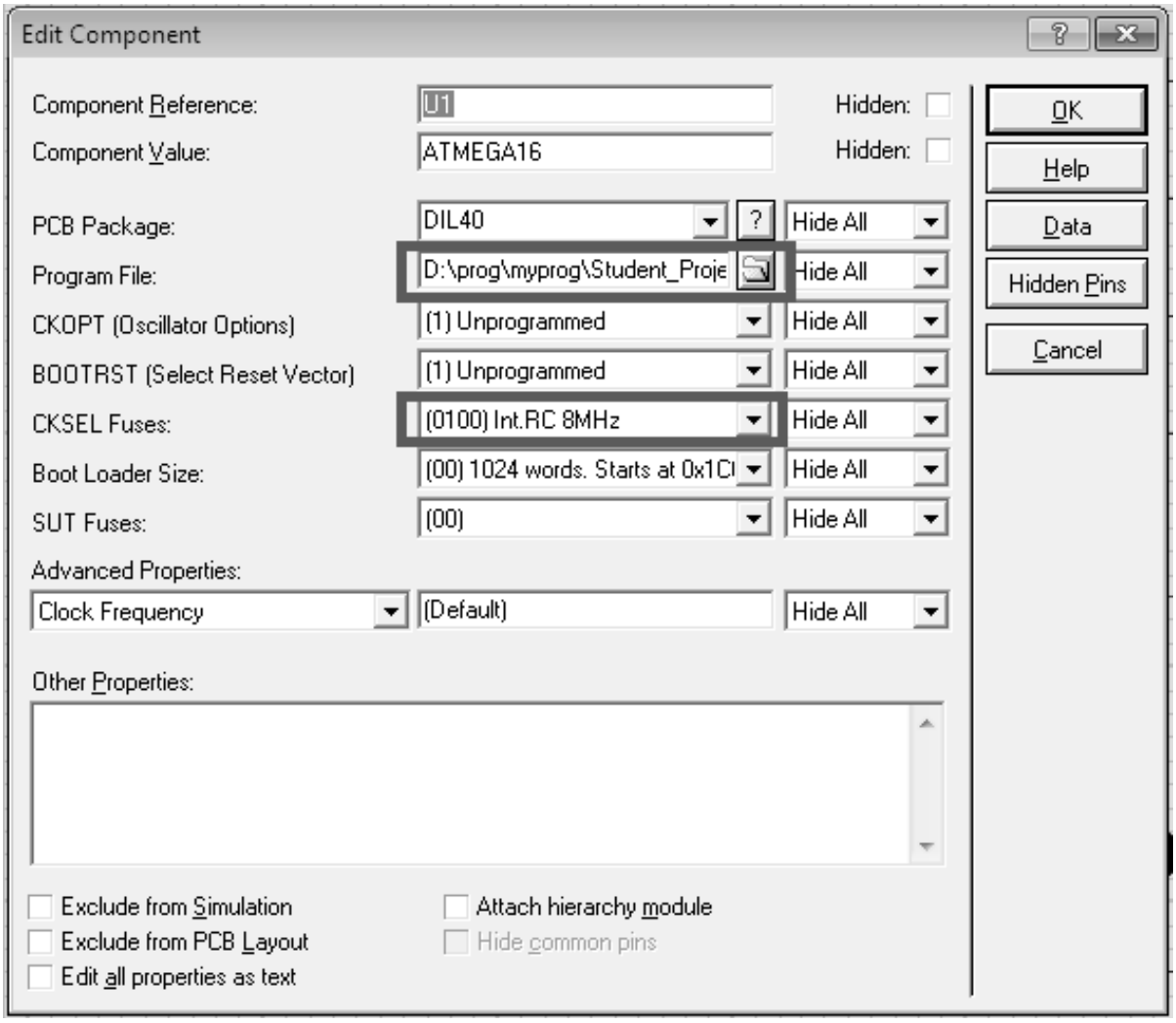

Рисунок 2.9 – Окно настроек модели микроконтроллера

Рисунок 2.10 – Клавиши управления моделированием

Варианты индивидуальных заданий к лабораторной работе

1 Реализовать логическую функцию «И» для двух тетрад порта микроконтроллера.

2 Реализовать логическую функцию «ИЛИ» для двух тетрад порта микроконтроллера.

3 Реализовать логическую функцию «И-НЕ» для двух тетрад порта микроконтроллера.

4 Реализовать логическую функцию «ИЛИ-НЕ» для двух тетрад порта микроконтроллера.

5 Реализовать логическую функцию сложения для двух тетрад порта микроконтроллера.

6 Реализовать логическую функцию вычитания для двух тетрад порта микроконтроллера.

7 Реализовать логическую функцию умножения для двух тетрад порта микроконтроллера.

8 Реализовать логическую функцию деления для двух тетрад порта микроконтроллера.

9 Реализовать логическую функцию возведения в степень для двух тетрад порта микроконтроллера.

Содержание отчета

Отчет выполняется индивидуально каждым студентом на листах формата А4. Текст отчета оформляется в соответствии с требованиями ГОСТ 2.105–95. В состав отчета включаются:

– титульный лист;
– индивидуальное задание;
– код программы;
– принципиальная схема к лабораторной работе.

Контрольные вопросы

1 Что такое микроконтроллер?
2 Что такое порт микроконтроллера?
3 Почему для программирования микроконтроллеров используется язык С?
4 Для чего нужен файл с расширением .hex?
5 Для чего нужен симулятор Proteus?

3 Лабораторная работа № 3. Изучение особенностей программирования микроконтроллеров на языке С. Управление шаговым двигателем

3.1 Ход работы

Целью работы является изучение особенностей программирования микроконтроллеров с помощью языка программирования С. Также нужно составить программу по заданию преподавателя, перевести ее в машинные коды, записать в память программ микроконтроллера и выполнить.

Перед началом лабораторных занятий студентам необходимо изучить следующие вопросы:

– язык программирования С;
– особенности разработки программ в среде Atmel Studio;

– использование среды Proteus для моделирования микропроцессорных систем;

– основы функционирования шагового двигателя.

В ходе выполнения работы:

– изучить электрическую принципиальную схему к лабораторной работе;

– разработать программу в соответствии с индивидуальным заданием;

– отладить программу в среде Atmel Studio;

– произвести моделирование в среде Proteus;

– оформить отчет по лабораторной работе.

После выполнения работы необходимо ответить на контрольные вопросы.

3.2 Порядок выполнения работы

Для выполнения данной работы нужно собрать в среде моделирования Proteus схему, представленную на рисунке 3.1. На рисунке 3.1 приняты следующие обозначения:

1 – микроконтроллер Atmega16;

2 – микросхема ULN2003;

3 – шаговый двигатель (Library: MOTORS, Device: MOTOR_STEPPER);

4 – источник постоянного напряжения;

5 – переключатель.

Для того чтобы выбрать источник постоянного напряжения, нужно перейти во вкладку, как показано на рисунке 3.2, *а*. Для задания величины напряжения дважды кликнуть по изображению источника напряжения (рисунок 3.2, *б*).

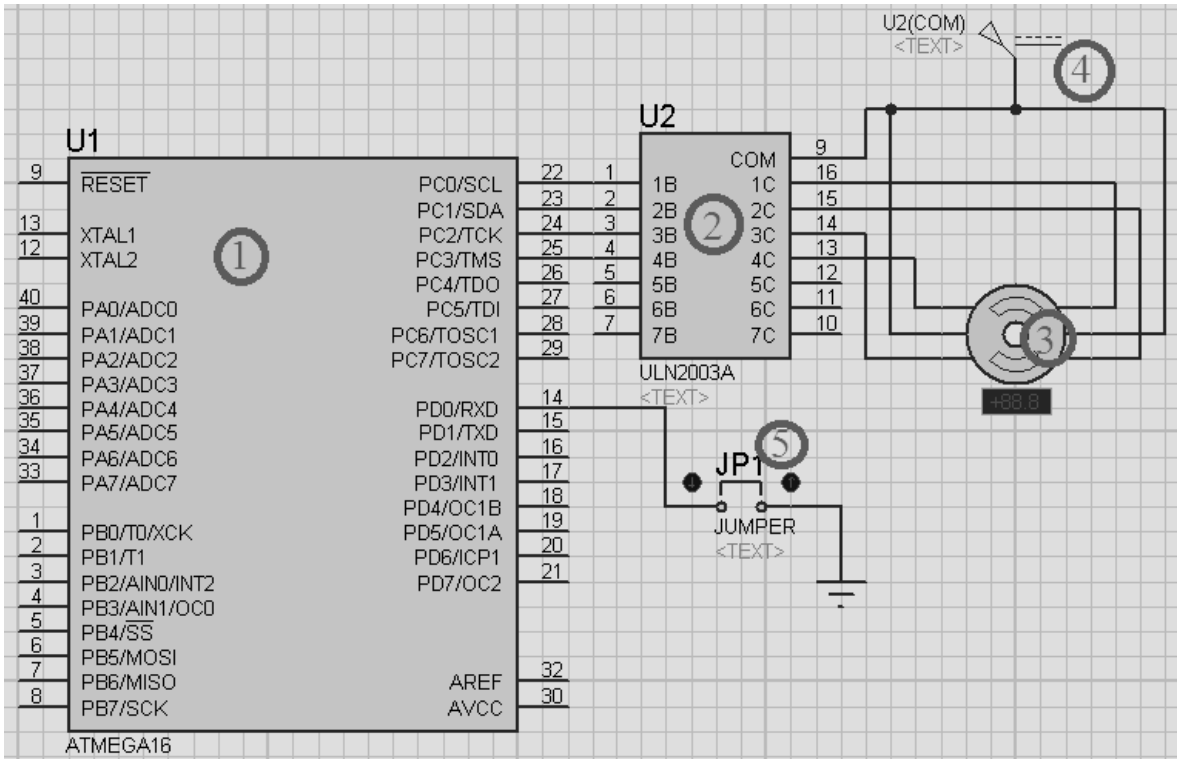

Рисунок 3.1 – Схема к лабораторной работе

а) *б*)

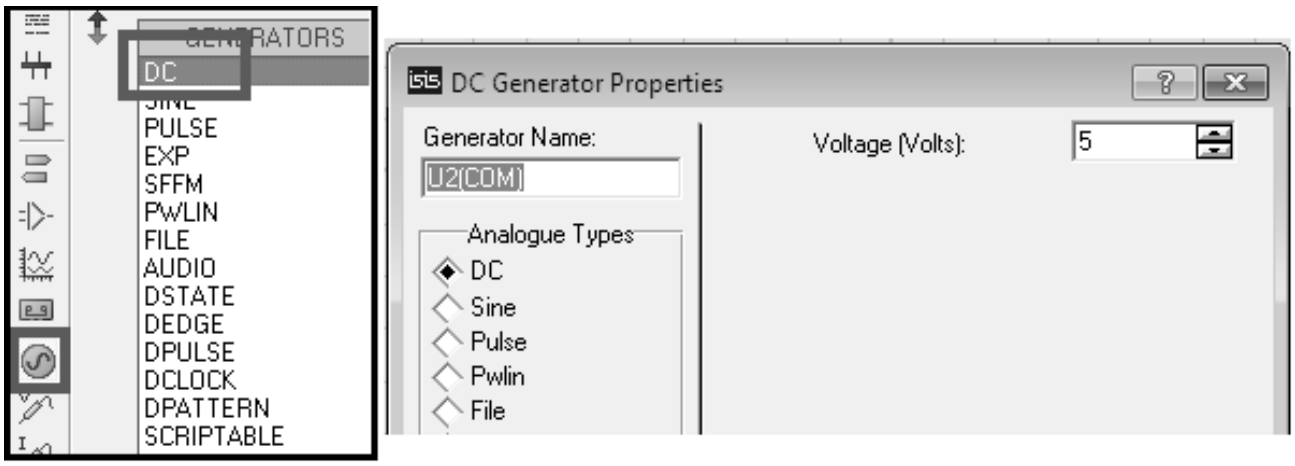

Рисунок 3.2 – Источник постоянного напряжения и его настройки

Пример программы микроконтроллера для управления шаговым двигателем:

```
/*
 * GccApplication1.c
 *
 * Created: 15.02.2017 22:00:07
 *  Author: Владимир
 */
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
uint8_t a = 0;

DDRA = 0x00;
PORTA = 0xFF;
DDRC = 0xFF;

   while(1)
   {
      //TODO:: Please write your application code
a = PINA;
a &= 1;
if (a == 0)
{
PORTC = 0b0001;
_delay_ms(1000);
PORTC = 0b0010;
_delay_ms(1000);
PORTC = 0b0100;
```

```
_delay_ms(1000);
PORTC = 0b1000;
_delay_ms(1000);
}

   }
}
```

Данная программа проверяет нажатие кнопки, подключенной к порту А микроконтроллера. С том случае, если кнопка нажата, шаговый двигатель приходит во вращение.

В следующем примере кода демонстрируется использование ссылки для передачи переменной в функцию, что позволяет запоминать положение вала шагового двигателя:

```
/*
 * GccApplication1.c
 *
 * Created: 15.02.2017 22:00:07
 *  Author: Владимир
 */
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#include <avr/io.h>
#include <util/delay.h>

void STEP(uint8_t *step_num)
{
static uint8_t counter = 1;

if (*step_num == 1)
{
PORTC = 0b0001;
}
if (*step_num == 2)
{
PORTC = 0b0010;
}
if (*step_num == 3)
{
PORTC = 0b0100;
}
if (*step_num == 4)
{
PORTC = 0b1000;
}
*step_num += 1;
if (*step_num == 5)
{
```

```
*step_num = 1;
}
counter++;
PORTA = counter;
//return step_num;

}

int main(void)
{
uint8_t a = 0;
uint8_t s = 1;

DDRD = 0x00;
PORTD = 0xFF;
DDRC = 0xFF;
DDRA = 0xFF;

   while(1)
   {
      //TODO:: Please write your application code
a = PIND;
a &= 1;
if (a == 1)
{
STEP(&s);
_delay_ms(500);
}


   }
}
```


Варианты индивидуальных заданий к лабораторной работе

1 Написать программу для регулирования скорости вращения шагового двигателя.

2 Написать программу для изменения направления вращения шагового двигателя.

3 Написать программу для задания числа оборотов шагового двигателя.

4 Написать программу для управления шаговым двигателем с использованием указателя.

5 Написать программу для выбора коэффициента дробления шага шагового двигателя.

Содержание отчета

1 Цель работы.

2 Постановка задачи.

3 Схема электрическая принципиальная к лабораторной работе.
4 Блок-схема алгоритма пользовательской программы.
5 Текст пользовательской программы.

Контрольные вопросы

1 В чем особенность шагового двигателя?
2 Как осуществляется управление шаговым двигателем?
3 Для чего используются ссылки в языке С?
4 Как программно управлять скоростью шагового двигателя?
5 Как программно осуществляется дробление шага шагового двигателя?

4 Лабораторная работа № 4. Изучение особенностей программирования микроконтроллеров на языке С. Вывод информации на семисегментный индикатор

4.1 Ход работы

Целью работы является изучение особенностей программирования микроконтроллеров с помощью языка программирования С. Также нужно составить программу по заданию преподавателя, перевести ее в машинные коды, записать в память программ микроконтроллера и выполнить.

Перед началом лабораторных занятий студентам необходимо изучить следующие вопросы:

– язык программирования С;
– особенности разработки программ в среде Atmel Studio;
– использование среды Proteus для моделирования микропроцессорных систем;
– семисегментные индикаторы.

В ходе выполнения работы:

– изучить электрическую принципиальную схему к лабораторной работе;
– разработать программу в соответствии с индивидуальным заданием;
– отладить программу в среде Atmel Studio;
– произвести моделирование в среде Proteus;
– оформить отчет по лабораторной работе.

После выполнения работы необходимо ответить на контрольные вопросы.

4.2 Порядок выполнения работы

Для выполнения данной работы нужно собрать в среде моделирования Proteus схему, представленную на рисунке 4.1. На рисунке 4.1 приняты следующие обозначения:

1 – микроконтроллер Atmega16;

2 – семисегментный индикатор;
3 – источник постоянного напряжения;
4 – переключатель.

Для того чтобы выбрать источник постоянного напряжения, нужно перейти во вкладку, как показано на рисунке 4.2, *а*. Для задания величины напряжения дважды кликнуть по изображению источника напряжения (рисунок 4.2, *б*).

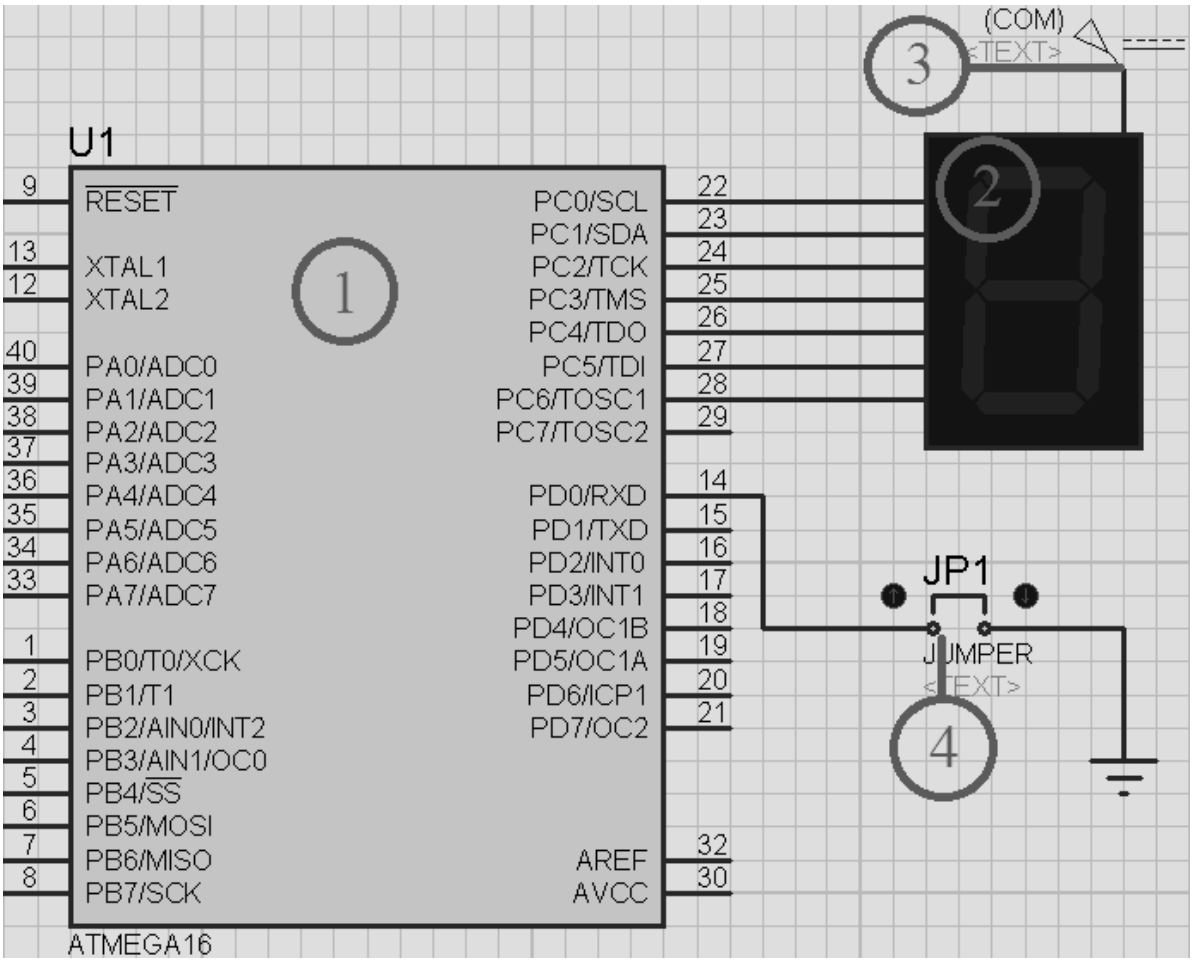

Рисунок 4.1 – Схема к лабораторной работе

а) *б*)

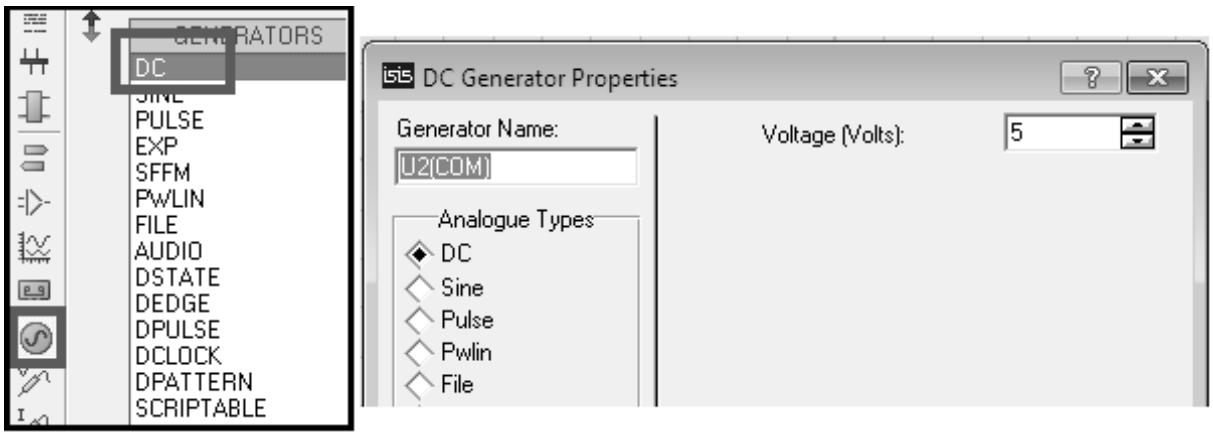

Рисунок 4.2 – Источник постоянного напряжения и его настройки

Пример программы микроконтроллера для вывода информации на семисегментный индикатор:

```
/*
 * GccApplication1.c
 *
 * Created: 15.02.2017 22:00:07
 *  Author: Владимир
 */
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
uint8_t a = 0;
uint8_t s = 1;

DDRD = 0x00;
PORTD = 0xFF;
DDRC = 0xFF;

    while(1)
    {
        //TODO:: Please write your application code
a = PIND;
a &= 1;
if (a == 1)
{
PORTC = 0b11111001; // 1
_delay_ms(500);
PORTC = 0b10100100; // 2
_delay_ms(500);
PORTC = 0b10110000; // 3
_delay_ms(500);
PORTC = 0b10011001; // 4
_delay_ms(500);
PORTC = 0b10010010; // 5
_delay_ms(500);
PORTC = 0b10000010; // 6
_delay_ms(500);
PORTC = 0b11111000; // 7
_delay_ms(500);
PORTC = 0b10000000; // 8
_delay_ms(500);
PORTC = 0b10010000; // 9
_delay_ms(500);
PORTC = 0b11000000; // 0
_delay_ms(500);
}
```

```
    }
}
```

Данная программа проверяет нажатие кнопки, подключенной к порту D микроконтроллера. В том случае, если кнопка нажата, на индикатор последовательно выводятся цифры от 0 до 9.

Варианты индивидуальных заданий к лабораторной работе

1 Вывести на семисегментный индикатор слово «СОН».
2 Вывести на семисегментный индикатор слово «Start».
3 Вывести на семисегментный индикатор слово «Stop».
4 Вывести на семисегментный индикатор слово «Error».
5 Вывести на семисегментный индикатор результат арифметической операции.
6 Вывести на семисегментный индикатор слово при нажатии на клавишу.

Содержание отчета

1 Цель работы.
2 Постановка задачи.
3 Схема электрическая принципиальная к лабораторной работе.
4 Блок-схема алгоритма разработанной программы.
5 Текст разработанной программы.
6 Выводы по работе.

Контрольные вопросы

1 Что такое семисегментный индикатор?
2 Как работает семисегментный индикатор?
3 Как осуществляется управление семисегментным индикатором?
4 Как выводить буквы на семисегментный индикатор?
5 Какие бывают разновидности семисегментных индикаторов?

5 Лабораторная работа № 5. Изучение особенностей программирования микроконтроллеров на языке С. Вывод информации на жидкокристаллический индикатор

5.1 Ход работы

Целью работы является изучение особенностей программирования микроконтроллеров с помощью языка программирования С. Также нужно составить программу по заданию преподавателя, перевести ее в машинные коды, записать в память программ микроконтроллера и выполнить.

Перед началом лабораторных занятий студентам необходимо изучить следующие вопросы:

– язык программирования С;

– особенности разработки программ в среде Atmel Studio;

– использование среды Proteus для моделирования микропроцессорных систем;

– знакосинтезирующие жидкокристаллические индикаторы.

В ходе выполнения работы:

– изучить электрическую принципиальную схему к лабораторной работе;

– разработать программу в соответствии с индивидуальным заданием;

– отладить программу в среде Atmel Studio;

– произвести моделирование в среде Proteus;

– оформить отчет по лабораторной работе.

После выполнения работы необходимо ответить на контрольные вопросы.

5.2 Порядок выполнения работы

Для выполнения данной работы нужно собрать в среде моделирования Proteus схему, представленную на рисунке 5.1. На рисунке приняты следующие обозначения:

1 – микроконтроллер Atmega16;

2 – жидкокристаллический индикатор.

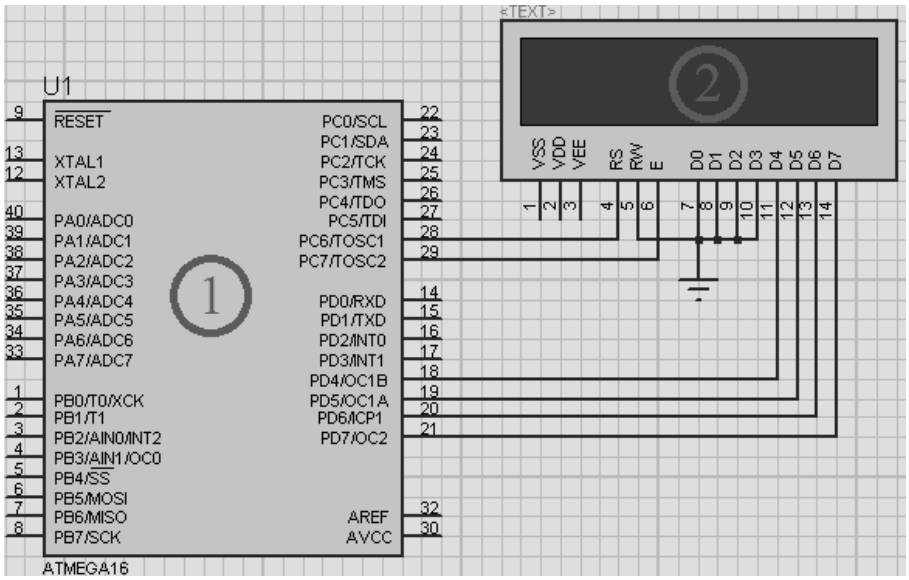

Рисунок 5.1 – Схема к лабораторной работе

Для того чтобы найти модель индикатора, нужно ввести в строке поиска компонентов «LM016L».

Для того чтобы получить возможность вывода информации на индикатор, необходимо подключить к своему проекту файл **lcd.h** (имеется у преподавателя).

Для подключения файла нужно кликнуть правой кнопкой мыши на директории проекта (рисунок 5.2), затем выбрать Add >> Existing Item и найти

предварительно скопированный в директорию проекта файл.

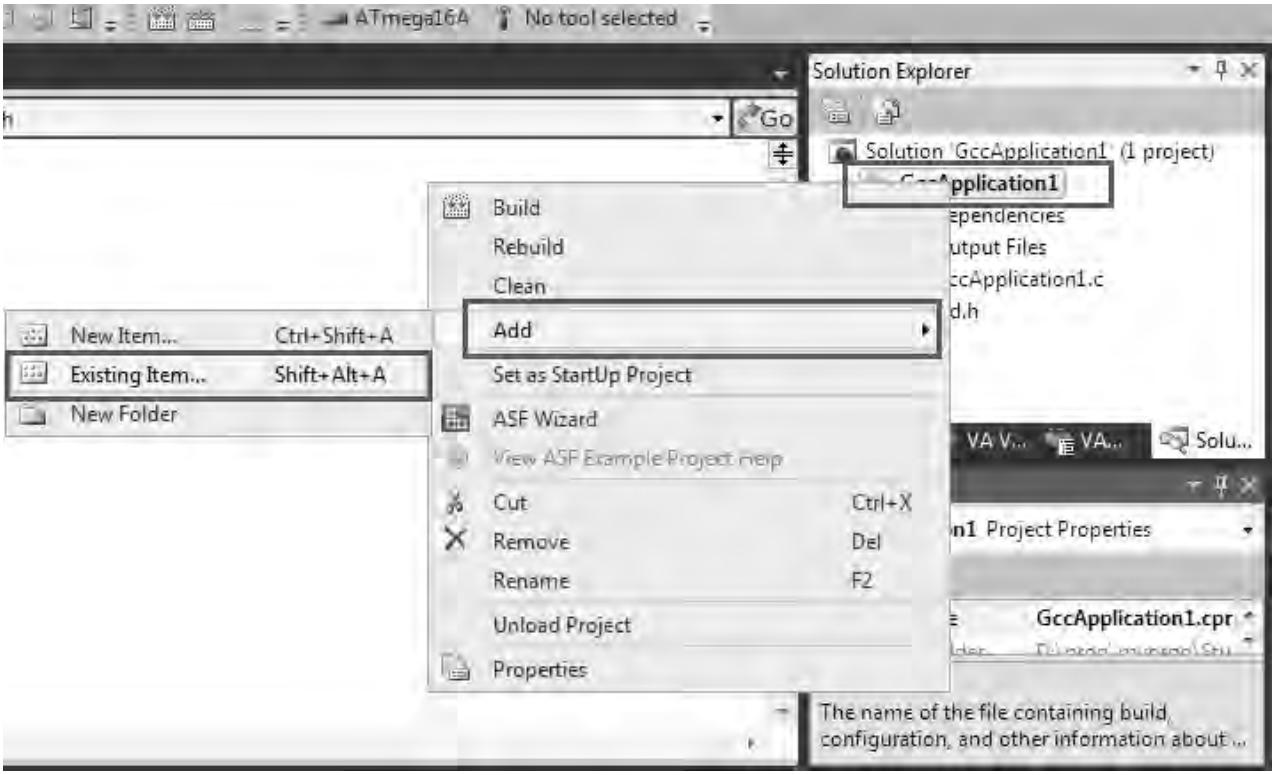

Рисунок 5.2 – Подключение файла к проекту

В файле lcd.h содержатся несколько полезных функций для работы с индикатором. Необходимо обратить внимание на следующие функции:

– void Lcd4_Clear();
– void Lcd4_Set_Cursor(char a, char b);
– void Lcd4_Init();
– void Lcd4_Write_Char(char a);
– void Lcd4_Write_String(char *a).

Пример программы микроконтроллера для вывода информации на семисегментный индикатор:

```
#ifndef F_CPU
#define F_CPU 8000000UL // 16 MHz clock speed
#endif
#define D4 eS_PORTD4
#define D5 eS_PORTD5
#define D6 eS_PORTD6
#define D7 eS_PORTD7
#define RS eS_PORTC6
#define EN eS_PORTC7

#include <avr/io.h>
#include <util/delay.h>
#include "lcd.h"
```

```
int main(void)
{
DDRD = 0xFF;
DDRC = 0xFF;
int i;
Lcd4_Init();
while(1)
{
Lcd4_Set_Cursor(1,1);
Lcd4_Write_String("LCD Hello World");
_delay_ms(1000);
Lcd4_Clear();
_delay_ms(1000);

}
}
```

Данная программа выводит на индикатор строку "LCD Hello World" и очищает индикатор с периодичностью в 2 с.

Варианты индивидуальных заданий

1 Вывести на индикатор свое имя.
2 Вывести на индикатор свою фамилию.
3 Вывести на индикатор номер группы.
4 Сделать мигающую надпись.
5 Сделать бегущую строку.

Содержание отчета

1 Цель работы.
2 Постановка задачи.
3 Схема электрическая принципиальная к лабораторной работе.
4 Блок-схема алгоритма разработанной программы.
5 Текст разработанной программы.
6 Выводы по работе.

Контрольные вопросы

1 Принцип работы знакосинтезирующего индикатора.
2 Подключение библиотеки к проекту.
3 Функции вывода символа и переноса каретки.

Список литературы

1 **Опейко, О. Ф.** Микропроцессорные средства в автоматизированном электроприводе : учебное пособие / О. Ф. Опейко, Ю. В Петренко. – Минск : Амалфея, 2008. – 340 с.

2 **Джозеф, Ю.** Ядро Cortex M3 компании ARM. Полное руководство / Ю. Джозеф. – Москва : Додэка XXI, 2012. – 552 с. : ил.

3 **Иванов, В. Н.** Электроника и микропроцессорная техника : учебник / В. Н. Иванов, И. О. Мартынова. – Москва : Академия, 2016. – 288 с.

4 **Гуров, В. В.** Микропроцессорные системы : учебное пособие / В. В. Гуров. – Москва : ИНФРА-М, 2017. – 336 с.